UNIVERSIDADE FEDERAL DE RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

Redes-em-Chip: Arquiteturas e Modelos para Avaliação de Área e Desempenho

por

CESAR ALBENES ZEFERINO

Tese submetida à avaliação
como requisito parcial para a obtenção
do grau de Doutor em Ciência da Computação

Dr. Altamiro Amadeu Susin
Orientador

Dr. Sergio Bampi
Co-orientador

Porto Alegre, junho de 2003.

CIP – CATÁLOGO NA PUBLICAÇÃO

Zeferino, Cesar Albenes

Redes-em-Chip: Arquiteturas e Modelos para Avaliação de Área e Desempenho / por Cesar Albenes Zeferino. – Porto Alegre: PPGC da UFRGS, 2003.

242 p.: il. + 1 CD-ROM

Tese (doutorado) – Universidade Federal do Rio Grande do Sul. Programa de Pós Graduação em Computação, Porto Alegre, BR-RS, 2003. Orientador: Susin, Altamiro Amadeu; Co-orientador: Bampi, Sergio.

1. Microeletrônica. 2.Sistemas Integrados. 3. Redes-em-Chip. I. Susin, Altamiro Amadeu. II. Bampi, Sergio. III. Título

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitora: Profa. Wrana Panizzi
Pró-Reitor de Ensino: Prof. José Carlos Ferraz Hennemann
Pró-Reitora Adjunta de Pós-Graduação: Profa. Jocélia Grazia
Diretor do Instituto de Informática: Prof. Philippe Olivier Alexandre Navaux
Coordenador do PGCC: Prof. Carlos Alberto Heuser
Bibliotecária-Chefe do Instituto de Informática: Beatriz Regina Bastos Haro

Agradecimentos

À todos aqueles que de uma forma ou de outra contribuíram à minha formação no Programa de Pós-Graduação em Computação da Universidade Federal do Rio Grande do Sul (UFRGS) e, principalmente, à realização desta tese.

Ao professor Altamiro Amadeu Susin, pela sua confiança, amizade, respeito e orientação na condução dos trabalhos. A ele deixo minha eterna homenagem sob a forma de uma arquitetura de rede-em-chip denominada SoCIN.

Ao professor Sergio Bampi, pela sua co-orientação e por instigar-me a estudar temas importantes que fomentaram a minha formação e a realização desta tese.

Ao professor Alain Greiner, da Universidade de Paris, pela sua confiança e por oportunizar a participação no projeto SPIN, cuja experiência que foi de vital importância para a focalização do tema desta tese.

Ao professor Luigi Carro que associou-se à nossa idéia e com sua amizade, seu espírito crítico e sua objetividade ajudou-nos a obter resultados importantes.

Aos demais professores que me orientaram no curso de minha vida acadêmica, agradeço pelos seus ensinamentos e deixo aqui registrados os seus nomes: José Renes Pinheiro, Humberto Pinheiro, Fábio Antônio Baldissera, Mário Imaguire, Hermann Adolf Harry Lücke (*in memorium*) e Márcio Cherem Schneider.

A todos os professores do Grupo de Microeletrônica (GME) da UFRGS que com seus esforços pessoais construíram um grupo de pesquisa de excelência, do qual sinto orgulho de ter participado e contribuído para a sua qualificação.

A todos os colegas do GME, alunos de graduação, mestrado e doutorado, com os quais convivi durante esses cinco anos de trabalho. Em especial, deixo meu agradecimento ao Rogério Rizzi, ao Ricardo Vargas Dorneles, ao Márcio Eduardo Kreutz e ao Edgard de Faria Corrêa, que acompanharam e apoiaram mais de perto a evolução deste trabalho em diferentes fases do seu desenvolvimento.

Ao CNPq, pelo apoio financeiro para realização da tese e do estágio sanduíche.

Aos meus pais, Elpídio e Elenita, à minha irmã Enelise, ao meu cunhado, Mauro, ao meu sobrinho Artur, à minha cunhada Soriane e aos meus sogros, Luis e Iracema, os quais sempre torceram por mim e tiveram a compreensão necessária. Agradeço, por propiciarem um ambiente familiar harmonioso e pleno de amor.

Um agradecimento especial à minha mãe que durante todo esse período orou para que eu tivesse êxito em cada etapa deste caminho.

À minha esposa, Jaine, pelo seu amor, apoio, compreensão, paciência e por sacrificar sua vida profissional em meu favor. Principalmente, agradeço a ela por existir e fazer-me feliz.

À Deus.

Sumário

Lista de Abreviaturas

AHB	Advanced High-perfomance Bus
AMBA	Advanced Microcontoller Bus Architecture
APB	Advanced Peripheral Bus
ASIM	Architecture des Systèmes intégrés et Microélectronique
aSoC	adaptive System-on-Chip
BCU	Bus Control Unit
BFW	Buffered Wormhole Switching
BIST	Built-In Self Test
bop	begin-of-packet
BPS	Backbone-Platform-System
bps	bits por segundo
CASS	Cycle-Accurate System Simulator
CBDA	Centrally-Buffered, Dynamically-Allocated
CBS	CrossBar Switch
CLICHÉ	Chip-Level Integration of Communicating Heterogeneous Elements
CMOS	Complementary Metal-Oxide Semiconductor
CPE	Codificador de Prioridade Estática
CPLD	Complex Programmable Logic Device
CPU	Central Processing Unit
DAMQ	Dynamically-Allocated, Multi-Queue
DCR	Device Control Register bus
DFT	Design For Test
DSP	Digital Signal Processor
E/S	Entrada-e-Saída
eop	end-of-packet
FCFS	First-Come-First-Served
FFT	Fast Fourier Transform
FIFO	First-In, First-Out
FPGA	Field Programmable Gate Array
GT	Gerador de Tráfego
HDL	Hardware Description Language
HLP	High Level Protocol
HOL	Head-of-Line blocking

HSL	High-Speed Link
I/F	Interface
I/O	Input/Output
IB	Input Buffer
IC	Input Controller
ICB	Input Control Block
IFC	Input Flow Controller
IIR	Infinite Impulse Response
IP	Intelectual Property
IRS	Input Rd Switch
LIP6	Laboratoire d'Informatique, Université Paris VI
LRS	Least Recently Served
MAC	Mulitply-And-Accumulate
OC	Output Controller
OCB	Output Control Block
OCN	On-Chip Network
OCP	Open Core Protocol
ODS	Output Data Switch
OFC	Output Flow Controller
OMI	Open Microprocessor Systems Initiative
OPB	On-chip Peripheral Bus
ORS	Output Rok Switch
PBD	Platform-Based Design
PC	Parte Controle
PHIT	Physical unIT
PI-Bus	Peripheral Interconnect Bus
PLB	Processor Local Bus
PO	Parte Operativa
QoS	Quality of Service
RAM	Random Acess Memory
RASoC	Router Architecture for Systems-on-Chip
RIB	Routing Information Bits
RR	Round-Robin
RSPIN	Router SPIN
RTL	Register Transfer Level

SAF	Store-And-Forward
SAFC	Statically Allocated, Fully Connected
SAMQ	Statically Allocated Multi-Queue
SoC	System-on-Chip
SoCIN	System-on-Chip, Interconnection Network
SPIN	Scalable Programmable Integrated Network
SRAM	Static Random Access Memory
UART	Universal Asynchronous Receiver/Transmitter
ULSI	Utra Large Scale Integration
VCI	Virtual Component Interface
VCT	Virtual Cut-Through
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
VLSI	Very Large Scale Integration
VSIA	Virtual Socket Interface Alliance

Lista de Figuras

Lista de Tabelas

Resumo

Com o advento dos processos submicrônicos, a capacidade de integração de transistores tem atingido níveis que possibilitam a construção de um sistema completo em uma única pastilha de silício. Esses sistemas, denominados sistemas integrados, baseiam-se no reuso de blocos previamente projetados e verificados, os quais são chamados de núcleos ou blocos de propriedade intelectual. Os sistemas integrados atuais incluem algumas poucas dezenas de núcleos, os quais são interconectados por meio de arquiteturas de comunicação baseadas em estruturas dedicadas de canais ponto-a-ponto ou em estruturas reutilizáveis constituídas por canais multiponto, denominadas barramentos. Os futuros sistemas integrados irão incluir de dezenas a centenas de núcleos em um mesmo chip com até alguns bilhões de transistores, sendo que, para atender às pressões do mercado e amortizar os custos de projeto entre vários sistemas, é importante que todos os seus componentes sejam reutilizáveis, incluindo a arquitetura de comunicação. Das arquiteturas utilizadas atualmente, o barramento é a única que oferece reusabilidade. Porém, o seu desempenho em comunicação e o seu consumo de energia degradam com o crescimento do sistema. Para atender aos requisitos dos futuros sistemas integrados, uma nova alternativa de arquitetura de comunicação tem sido proposta na comunidade acadêmica. Essa arquitetura, denominada rede-em-chip, baseia-se nos conceitos utilizados nas redes de interconexão para computadores paralelos. Esta tese se situa nesse contexto e apresenta uma arquitetura de rede-em-chip e um conjunto de modelos para a avaliação de área e desempenho de arquiteturas de comunicação para sistemas integrados. A arquitetura apresentada é denominada SoCIN (*System-on-Chip Interconnection Network*) e apresenta como diferencial o fato de poder ser dimensionada de modo a atender a requisitos de custo e desempenho da aplicação alvo. Os modelos desenvolvidos permitem a estimativa em alto nível da área em silício e do desempenho de arquiteturas de comunicação do tipo barramento e rede-em-chip. São apresentados resultados que demonstram a efetividade das redes-em-chip e indicam as condições que definem a aplicabilidade das mesmas.

Palavras-chave: Microeletrônica. Sistemas Integrados. Redes-em-Chip.

TITLE: “NETWORKS-ON-CHIP: ARCHITECTURES AND MODELS TO EVALUATE AREA AND PERFORMANCE”

Abstract

With the developing of new submicron process technologies, it has been possible to integrate an entire system on a single chip. Such kind of system is named System-on-Chip (SoC) and its design methodologies are based on the reuse of pre-designed and pre-verified components named cores or intellectual property blocks. Current SoCs include a few dozens of cores interconnected by communication architectures based on point-to-point dedicated channels or in multi-point channels (also named shared busses). However, the future SoCs will integrate up to hundreds of cores on a single billion-transistor chip. To meet the market pressures and to amortize the design costs among several systems, it is important that all the systems components are reusable, including the communication architecture. Considering the current on-chip communication architectures, only the bus-based one offers reusability. Meanwhile, their performance and power consumption degrade with the system growing. To meet the requirements of the future SoCs, a new approach has been proposed in the academia. It is named Network-on-Chip (NoC) and is based on the concepts used in interconnection networks for parallel computers. The current thesis is inserted in this context and presents a new NoC architecture and a set of models to allow the evaluation of area and performance of on-chip communication architectures. The network architecture is named SoCIN (System-on-Chip Interconnection Network) and presents scalability features which allow to the designer to tune the network dimensions in order to meet costs and performance requirements of the target application. The models allow to get a high level estimation of silicon area and performance of busses and NoCs. Several results are presented and they demonstrate the effectiveness of the NoC approach and indicate the conditions defining the applicability of a NoC.

Keywords: Microelectronics. System-on-Chip. Network-on-Chip.

1 Introdução

O advento dos processos submicrônicos tem permitido um aumento do nível de integração de transistores em uma mesma pastilha de silício, e possibilitará a sustentação da Lei de Moore por ainda muitos anos. Esse avanço na tecnologia tem possibilitado a integração de múltiplos componentes, como processadores, controladores e memória, em um único chip, resultando na integração de um sistema completo em uma mesma pastilha. Esses sistemas são denominados sistemas integrados ou SoCs (*Systems-On-Chip*).

Para atender às pressões do mercado e amortizar os custos de projeto entre vários sistemas, é importante que os componentes integrados em um SoC sejam reutilizáveis. Dessa forma, as metodologias de projeto adotadas devem ser baseadas no reuso de componentes pré-projetados e pré-verificados. Esses componentes reutilizáveis são denominados núcleos e podem ser desenvolvidos pela empresa responsável pelo projeto do sistema ou adquiridos de terceiros.

Os núcleos de um sistema integrado são interconectados por meio de uma estrutura de canais denominada arquitetura de comunicação (também referenciada pelos termos arquitetura, estrutura ou rede de interconexão). Atualmente, são utilizadas duas abordagens para a interconexão dos núcleos: canais ponto-a-ponto dedicados e canais multiponto compartilhados. A primeira alternativa é a que oferece o melhor desempenho, pois cada comunicação ocorre independentemente das demais por meio de canais exclusivos. Contudo, ela requer um projeto específico e, portanto, possui reusabilidade limitada. Já na arquitetura multiponto, conhecida por barramento, a mesma estrutura pode ser reutilizada em diferentes sistemas, reduzindo o tempo de projeto.

Nesse contexto, muitos fabricantes oferecem soluções de integração de sistema que incluem bibliotecas completas com núcleos para componentes de processamento e de comunicação. Tipicamente, a arquitetura de comunicação utilizada é o barramento (ou uma hierarquia com dois ou mais barramentos), pois oferece, como vantagem, características de reusabilidade e baixo custo de silício.

Porém, os canais multiponto do barramento são conectados a vários núcleos e compartilhados pelas diferentes comunicações entre os núcleos do sistema. Com o aumento do número de núcleos conectados aos canais do barramento, a carga capacitiva desses canais é incrementada, resultando em um aumento na energia e no tempo necessários à propagação dos sinais pelos fios do barramento. Além disso, a largura de banda do barramento é definida pelo tempo de propagação dos sinais e pelo número de bits do seu canal de dados. Desde que a largura desse canal é fixa e o tempo de propagação se degrada com o aumento do sistema, a largura de banda do barramento também sofre com esse aumento. Além disso, quanto maior for o número de núcleos capazes de iniciar uma comunicação, menor será a largura de banda disponível a cada um deles, pois os mesmos deverão competir pelo uso do barramento. Portanto, para sistemas que possuem perspectivas de serem escalados pelo acréscimo do número de núcleos, é sabido, *a priori*, que o desempenho da arquitetura de comunicação será menor nas novas gerações.

Segundo estudos do ITRS (*International Technology Roadmap for Semiconductors*) [ITR 2003], até o final desta década, estarão sendo disponibilizadas tecnologias de processo de 100 - 50 nm. Com tais tecnologias, será possível a integração de até quatro bilhões de transistores e dezenas a centenas de núcleos em uma mesma pastilha de silício [BEN 2002], o que permitirá o desenvolvimento de novas aplicações nas áreas de multimídia, telecomunicações e eletrônica de consumo. Contudo, enquanto isso abre novas oportunidades de projeto, surgem também novas dificuldades com relação à especificação, mapeamento e avaliação das opções de projeto, assim como questões associadas às arquiteturas de comunicação.

Do ponto de vista da comunicação, o problema reside no fato de que esses sistemas serão tão complexos que inviabilizarão o uso de interconexões dedicadas face às dificuldades envolvidas e à falta de reusabilidade dessa abordagem. Por outro lado, os requisitos de desempenho em comunicação, como largura de banda escalável e baixa latência, dificilmente serão atendidos pelas arquiteturas baseadas no barramento. Isso decorre da natureza arquitetural do barramento que impõe uma série de problemas já identificados. Nos futuros sistemas integrados, os comprimentos dos fios de um barramento irão se manter proporcionais ao tamanho da pastilha (*die size*) e não irão diminuir com o aumento da freqüência de relógio devida à redução dos transistores. Como foi visto acima, por utilizar conexões do tipo multiponto, quanto maior for o número de núcleos conectados ao barramento, maior será a sua capacitância parasita. Assim, o atraso na movimentação de dados ao longo dos fios irá se tornar incrivelmente significativo e limitará o desempenho do sistema. Quanto à potência, o problema é que o barramento opera por difusão e cada sinal deve chegar a todos os pontos do barramento, exigindo uma grande quantidade de energia. Além desses, existem outros problemas, como largura de banda não escalável e arbitragem centralizada, que dificultarão em muito o uso de barramentos em SoCs complexos.

Logo, é evidente a necessidade de desenvolvimento de novos sistemas de comunicação que ataquem os problemas acima enumerados. Nesse contexto, a solução proposta pela comunidade científica está no uso de redes de interconexão chaveada, como aquelas encontradas em computadores paralelos. Essas redes têm como vantagens a largura de banda escalável, o uso de conexões ponto-a-ponto curtas e o paralelismo na comunicação, entre outras. Embora tenham como desvantagem maiores custos e latência na comunicação, esses problemas serão certamente atenuados pela grande disponibilidade de transistores e por soluções arquiteturais que permitirão reduzir a latência da rede e seus efeitos no desempenho da aplicação.

Essas redes chaveadas, quando aplicadas à comunicação intrachip, recebem múltiplas denominações na literatura: *micronetworks*, *On-Chip Networks* (OCNs) e *Networks-on-Chip* (NoC). Contudo, todas elas se referem à mesma base arquitetural (redes de interconexão chaveada para computadores paralelos), sendo que, neste texto, será adotado o terceiro termo (e a sua tradução para o português: redes-em-chip).

Existem alguns trabalhos que propõem arquiteturas de comunicação intrachip baseadas nos conceitos adotados em redes de computadores. O mais antigo conhecido nessa linha data de 1992 e discute o problema da interconexão em sistemas ULSI (*Ultra Large Scale Integration*), propondo a adoção de arquiteturas semelhantes às de uma rede local, com melhorias [TEW 92]. Um trabalho recente, de 2002, propõe uma arquitetura de interconexão intrachip também com características similares às de uma rede de computadores [BRI 2002]. Essas arquiteturas diferem em muitos pontos das

redes de interconexão chaveada para computadores paralelos, as quais são projetadas visando oferecer, ao mesmo tempo, alto desempenho, baixo custo e escalabilidade, e, no entendimento deste autor, não podem ser incluídas na categoria das redes-em-chip.

Desde o ano de 2000, quando foram publicados alguns dos primeiros trabalhos conhecidos com resultados efetivos sobre redes-em-chip [GUE 2000a, LAN 2000], o interesse pelo assunto tem crescido significativamente e tem sido tema de sessões especiais em importantes conferências internacionais (DAC'2001 e DATE'2002), nas quais foram apresentadas e discutidas diversas questões sobre o assunto. Da mesma forma, o espaço em publicações periódicas tem crescido de forma importante, com edições especiais dedicadas ao tema.

Disso percebe-se a importância e a necessidade de se inserir em um contexto que oferece um grande número de oportunidades de pesquisa. A tese aqui apresentada se estabelece nesse contexto e é o resultado de uma série de estudos e atividades de pesquisa iniciadas em 1998 com o objetivo principal de tratar das questões associadas às redes de interconexão chaveada de aplicação específica em nível de chip ou de agregado de computadores. A focalização do tema foi dada após diversos estudos e, principalmente, com a participação em um dos projetos de pesquisa pioneiros na área das redes-em-chip: o Projeto SPIN (*Scalable, Programmable Integrated Network*), desenvolvido no departamento ASIM (Architecture des Systèmes intégrés et Microélectronique) *do* LIP6 (Laboratoire d'Informatique Paris 6). No contexto do Projeto SPIN, trabalhou-se na modelagem e na avaliação de arquiteturas de comunicação para sistemas integrados por meio de simulação computacional. Foram avaliadas e comparadas duas arquiteturas: o barramento PI-Bus e a rede SPIN.

Dada a quantidade de oportunidades de pesquisa em aberto na área das redes em-chip, identificou-se uma carência por instrumentos de alto nível que auxiliassem a avaliação da aplicabilidade dessas redes em sistemas integrados. Ou seja, a partir de um sistema integrado com características e requisitos conhecidos, o problema reside em determinar qual a arquitetura de comunicação mais adequada a esse sistema e a partir de qual configuração uma rede-em-chip se faz necessária. Nesse contexto, foi desenvolvido um conjunto de modelos para avaliação de área e desempenho de arquiteturas de comunicação para sistemas integrados.

Uma outra questão atacada na tese foi a especificação de uma arquitetura de rede-em-chip e o desenvolvimento de um modelo VHDL parametrizável do seu roteador. Essa rede é denominada SoCIN (*System-on-Chip Interconnection Network*) e sua característica principal, e que a diferencia das demais redes-em-chip encontradas na literatura, é que ela se baseia em um núcleo de roteador configurável (denominado RASoC – *Routing Architecture for System-on-Chip*), cuja largura de canais e profundidade dos *buffers* podem ser dimensionadas em função dos requisitos do sistema. Além disso, a descrição é altamente modularizada e permite a exploração do espaço arquitetural pela substituição das alternativas utilizadas por outras que melhor atendam aos requisitos estabelecidos. A disponibilidade desse modelo tem fomentado o desenvolvimento de muitos projetos de pesquisa.

Esta tese é organizada como segue. No Capítulo 2, é apresentada uma revisão sobre redes de interconexão chaveada para computadores paralelos que permitem familiarizar o leitor com os conceitos envolvidos nesse tipo de rede. Essa revisão não é

exaustiva e se baseia em um estudo previamente efetuado e que resultou em uma monografia de Exame de Qualificação [ZEF 99c].

No Capítulo 3, são discutidos conceitos a respeito de sistemas integrados com ênfase ao problema da comunicação em SoCs. São apresentadas algumas das arquiteturas que representam o estado da arte das redes-em-chip sem, contudo, esgotar todos os exemplos possíveis de serem descritos.

O Capítulo 4 apresenta os resultados das atividades de pesquisa realizadas junto ao Projeto SPIN. São descritos os conceitos e tecnologias estudadas, o processo de modelagem da rede SPIN e os sistemas integrados modelados visando a avaliação e a comparação do desempenho da rede SPIN e do barramento PI-Bus. O capítulo inclui, também, os resultados dessa avaliação.

No Capítulo 5, é realizada uma descrição da arquitetura da rede SoCIN, bem como resultados de síntese, incluindo custos e freqüências de operação do seu roteador e de redes básicas para diferentes configurações dos parâmetros de entrada. A arquitetura do roteador RASoC é detalhada no Apêndice 1.

O Capítulo 6 apresenta um conjunto de modelos analíticos para a estimativa do custo em silício de arquiteturas de comunicação atuais, incluindo o barramento PI-Bus, a rede SPIN e a rede SoCIN. Esses modelos permitem avaliar o impacto das configurações do sistema e da rede na sobrecarga de área de silício adicionada pelos componentes de cada arquitetura.

No Capítulo 7, são descritos modelos analíticos que permitem uma estimativa em alto nível do desempenho das arquiteturas de comunicação consideradas no Capítulo 6. São apresentados resultados de avaliações das características de desempenho de cada arquitetura para diferentes configurações de sistema.

Ao término de cada capítulo, são apresentadas considerações que fazem uma avaliação local do texto e dos resultados associados. Dessa forma, o Capítulo 8, apresenta as considerações finais da tese de modo a oferecer um apanhado geral sobre os resultados, as contribuições deste trabalho, projetos futuros e oportunidades de pesquisa em redes-em-chip.

Por fim, é preciso destacar que o escopo da tese é limitado e busca atacar questões pontuais da área estudada. É no entendimento do autor que sua contribuição principal reside no desenvolvimento e na disponibilização de modelos de simulação, síntese e estimativa de área e desempenho de redes-em-chip, assim como um conjunto de dados e análises sobre a aplicabilidade dessas redes. Acredita-se que os resultados aqui obtidos irão fomentar um número de trabalhos a serem realizados em continuidade à tese e ajudar a consolidar uma linha de pesquisa por ela introduzida no Grupo de Microeletrônica da Universidade Federal do Rio Grande do Sul.

2 Noções Fundamentais sobre Redes de Interconexão Chaveada para Computadores Paralelos

Este capítulo apresenta uma revisão dos principais conceitos relacionados às redes de interconexão chaveada para computadores paralelos, mais comumente conhecidas pela expressão “redes de interconexão” (do inglês, *interconnection networks* ou INs). São discutidos os atributos que caracterizam as redes de interconexão, bem como os mecanismos de comunicação utilizados para a transferência de dados pela rede. O capítulo inicia com a apresentação de alguns conceitos e definições básicas, após o quê as topologias de redes de interconexão são apresentadas. Em seguida, são discutidos os problemas de *starvation*, *livelock* e *deadlock*, e os principais mecanismos e características das redes de interconexão: roteamento, chaveamento, controle de fluxo, memorização e arbitragem. Por fim, são apresentados alguns exemplos representativos de redes de interconexão chaveada para sistemas de computação paralela, de modo a ilustrar as alternativas arquiteturais utilizadas nesse nível.

2.1 Definições Básicas

As redes de interconexão são tipicamente utilizadas para interconectar os nodos de um computador paralelo, sendo que um nodo pode ser simplesmente um processador, um módulo de memória ou até um computador completo com processador e memória local e uma interface de rede. Em um multiprocessador, a rede de interconexão oferece a infra-estrutura necessária para que os diversos processadores acessem os módulos de memória compartilhada. A comunicação entre os processos executados nos processadores ocorre pelo acesso a variáveis compartilhadas em memória, sendo que os acessos a essas variáveis são protegidos por mecanismos de exclusão mútua. Já em um multicomputador, a rede interconecta os computadores da máquina e, por não haver compartilhamento de memória, a comunicação ocorre pela troca de mensagens. Esse tipo de computador é também chamado de multiprocessador de memória distribuída. Existem também variações desses modelos, como os multiprocessadores de memória compartilhada distribuída, mas o estudo sobre os mesmos foge do escopo deste texto.

Uma rede de interconexão é constituída basicamente por roteadores e enlaces (*links*). Os enlaces ligam os roteadores entre si e aos nodos do computador paralelo, enquanto que os roteadores estabelecem o caminho necessário à transferência de dados pela rede. Esses dados são transferidos sob a forma de mensagens, as quais podem ser divididas em unidades menores chamadas pacotes. Uma rede de interconexão é caracterizada pela sua topologia e por um conjunto de mecanismos que definem a forma como ocorrerá a transferência de mensagens pela rede. As diferentes alternativas de topologia e de mecanismos de comunicação têm impacto direto no desempenho da rede, o qual pode ser avaliado através de algumas métricas, como a largura de banda, a vazão e a latência. Todos esses aspectos, citados neste parágrafo, são discutidos abaixo e nas subseções que se seguem.

2.1.1 Roteador

Um roteador é constituído por um núcleo de chaveamento (*crossbar*), uma lógica de controle para roteamento e arbitragem, e portas de entrada-e-saída para comunicação com outros roteadores e/ou com os nodos do computador paralelo. Essas portas de comunicação incluem canais de entrada e de saída, os quais podem possuir uma pequena memória para armazenamento temporário de informações (a qual é denominada *buffer* de memorização ou, simplesmente, *buffer*). As portas possuem ainda controladores de enlace para a implementação do protocolo físico de comunicação. Eles regulam o tráfego das informações que entram e saem do roteador.

2.1.2 Enlaces

Um enlace liga um roteador a outro roteador ou a um nodo do computador paralelo e pode possuir um ou dois canais físicos de comunicação, sendo implementado sob a forma de um cabo elétrico ou óptico. Tipicamente, os enlaces utilizados em redes de interconexão são *full-duplex*, sendo constituídos por dois canais unidirecionais opostos de modo a permitir a transferência simultânea de informação nas duas direções do enlace.

2.1.3 Mensagens e pacotes

As informações trocadas pelos nodos fonte (emissor) e destinatário (receptor) de uma comunicação são organizadas sob a forma de mensagens. Em geral, uma mensagem possui três partes: um cabeçalho (*header*), a carga útil (*payload*) e um terminador (*trailer*), sendo que o cabeçalho e o terminador formam um envelope ao redor da carga útil da mensagem. O cabeçalho inclui informações de roteamento e de controle utilizadas pelos roteadores para propagar a mensagem em direção ao nodo destinatário da comunicação. O terminador, por sua vez, inclui informações usadas para a detecção de erros e para a sinalização do fim da mensagem.

Tipicamente, as mensagens são quebradas em pacotes para transmissão. Um pacote é a menor unidade de informação que contém detalhes sobre o roteamento e seqüenciamento dos dados e mantém uma estrutura semelhante à de uma mensagem, com um cabeçalho, uma carga útil e um terminador. Um pacote é constituído por uma seqüência de palavras cuja largura é igual à largura física do canal, a qual é denominada *phit*[1]. Em outras palavras, um phit é definido pelo número de bits de dado transmitidos simultaneamente, sendo igual a 1 bit nos enlaces seriais e a *n* bits nos enlaces paralelos de *n* bits. Além dos bits de dado, o phit pode incluir sinais de enquadramento e de controle da integridade do dado.

[1] PHIT = *PHysical unIT*

2.1.4 Parâmetros de desempenho de uma rede de interconexão

O desempenho de uma rede de interconexão pode ser avaliado pela sua largura de banda, pela vazão e/ou pela latência da rede.

A largura de banda (*bandwitdh*) refere-se à taxa máxima com a qual a rede de interconexão pode propagar as informações uma vez que uma mensagem entra na rede. Tradicionalmente, no cálculo da largura de banda, são contados os bits do cabeçalho, da carga útil e do terminador da mensagem, sendo que a unidade de medida utilizada é “bit por segundo” (ou bps) [PAT 96].

A vazão (*throughput*) é definida em [DAL 91] como o número de mensagens que a rede entrega na unidade de tempo e, em [PAT 96], como a largura de banda da rede para uma dada aplicação. Essas duas definições não são muito claras, pois, na primeira, o tamanho das mensagens pode variar e, na segunda, a vazão pode ser confundida com a largura de banda máxima da rede. Em [DUA 97], é apresentado um conceito mais nítido para a vazão, a qual é definida como o tráfego máximo aceito pela rede ou, em outras palavras, a quantidade máxima de informação entregue na unidade de tempo. Para que a vazão seja independente do tamanho das mensagens e da rede, seu valor pode ser normalizado, dividindo-o pelo tamanho das mensagens e pelo tamanho da rede. Como resultado, a vazão normalizada é medida em bits por nodo por ciclo de relógio (ou por microssegundo).

A latência é o tempo envolvido desde o início da transmissão de uma mensagem até o momento em que ela é completamente recebida. Em geral, a latência individual de cada mensagem é de pouca importância, principalmente quando são utilizadas cargas sintéticas [DUA 97]. Nesse caso, o valor médio das latências (ou latência média) é mais significativo para a avaliação da rede. Porém, se algumas mensagens experimentam latências muito maiores que o valor médio, isso pode ter um impacto importante no desempenho de algumas aplicações. Logo, além da latência média, o desvio padrão pode ser importante para ajudar a identificar essas situações.

A latência é medida em unidades de tempo. Porém, como muitas comparações de alternativas de projeto são realizadas utilizando-se simuladores de rede, a latência pode ser medida em ciclos de relógio do simulador.

A latência L de uma rede com carga é dada por (2.1), na qual os quatro termos do lado direito da equação são definidos na Tabela 2.1. A ocupação do canal depende do número de enlaces utilizados na rota da mensagem, sendo que a ocupação de cada enlace é dependente da largura de banda do canal em bps. O número de enlaces utilizados é denominado distância e quanto maior for o seu valor, maiores serão os atrasos acumulados pela ocupação do canal e também pelo roteamento. A latência total depende ainda das estratégias utilizadas para roteamento e chaveamento.

$$L = S + O + A_{RC} + C \tag{2.1}$$

TABELA 2.1 – Termos da equação da latência.

Termo		Definição
S	Sobrecarga (*overhead*)	Refere-se aos tempos gastos pelos nodos fonte e destinatário para, respectivamente, injetar e retirar a mensagem da rede.
O	Ocupação do canal	É uma medida do tempo gasto para transferir a mensagem pelos enlaces utilizados na rota entre os nodos fonte e destinatário.
A_{RC}	Atrasos de roteamento e de chaveamento	É o tempo gasto para a determinação da rota a ser utilizada para a propagação da mensagem através do roteador.
C	Atraso de contenção	Refere-se aos períodos de tempo durante os quais a mensagem é impedida de avançar devido ao congestionamento da rede.

2.1.5 Características de uma rede de interconexão

Uma rede de interconexão pode ser caracterizada pela sua topologia e pelas estratégias utilizadas para roteamento, controle de fluxo, chaveamento e arbitragem que ela utiliza, conforme resume a Tabela 2.2.

TABELA 2.2 – Características de uma rede de interconexão.

Característica	Definição/Função
Topologia	Define o arranjo dos roteadores e enlaces sob a forma de um grafo.
Roteamento	Determina como uma mensagem escolhe um caminho dentro do grafo.
Chaveamento	Define como e quando um canal de entrada de um roteador é conectado a um canal de saída selecionado pelo algoritmo de roteamento.
Controle de fluxo	Lida com a alocação de canais e *buffers* para uma mensagem que atravessa o grafo.
Arbitragem	Determina qual canal de entrada pode utilizar um determinado canal de saída do roteador.
Memorização	Define como e onde serão armazenadas mensagens bloqueadas em um roteador.

2.2 Topologia

Uma rede de interconexão pode ser caracterizada pela estrutura como seus roteadores são interligados. Essa estrutura é tipicamente representa por um grafo G(N,C) no qual N representa o conjunto de roteadores da rede e C representa o conjunto de canais de comunicação. Quanto à topologia, as redes de interconexão podem ser agrupadas em duas classes principais: as redes diretas e as redes indiretas.

2.2.1 Redes diretas

Nas redes diretas, cada roteador está associado a um processador e esse par pode ser visto como um elemento único dentro do computador (o qual pode ser referenciado simplesmente pelo termo "nodo"). Cada nodo possui ligações ponto-a-ponto diretas para um determinado número de nodos vizinhos. Uma mensagem trocada entre dois nodos não-vizinhos deve passar por um ou mais nodos intermediários. Se uma mensagem recebida por um nodo é destinada a outro nodo dentro da rede, o roteador do primeiro deve repassá-la para algum dos seus nodos vizinhos para que a mensagem avance em direção ao seu destinatário. Apenas os roteadores são envolvidos nessa comunicação, sem interferência aos processadores dos nodos intermediários. Um algoritmo de roteamento é utilizado pelo roteador a fim de decidir para qual nodo vizinho a mensagem deve ser repassada.

Em termos de conectividade, a rede direta ideal é a completamente conectada. Porém, sua escalabilidade é restrita e seu custo é proibitivo para um número de nodos moderado ou grande, pois o grau do nodo (número de canais por nodo) é N. 1, no qual N é o número de nodos da rede.

Face às dificuldades de se implementar uma rede direta ideal, como a completamente conectada, diversas alternativas foram propostas na tentativa de obter uma boa relação entre desempenho e custo. A grande maioria das implementações práticas se restringe a alguns modelos de rede que possuem topologia ortogonal. Uma rede apresenta topologia ortogonal se, e somente se, seus nodos podem ser arranjados em um espaço *n*-dimensional e cada enlace entre nodos vizinhos produz um deslocamento em uma única dimensão. As topologias de redes diretas ortogonais mais utilizadas são a grelha (ou malha) *n*-dimensional, o toróide (ou *k-ary n-cube*) e o hipercubo, exemplificados na Figura 2.1. Conforme é ilustrado, um nodo é composto pelo processador e pelo roteador, além de outros componentes, como memória e entrada-e-saída (E/S).

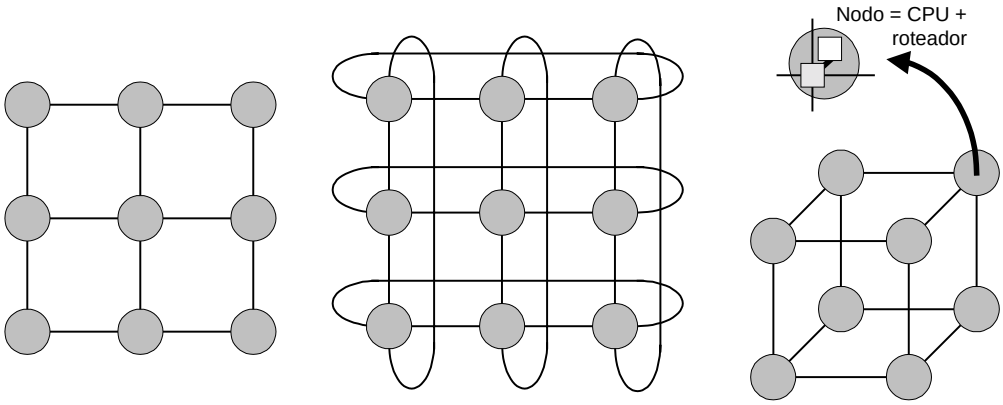


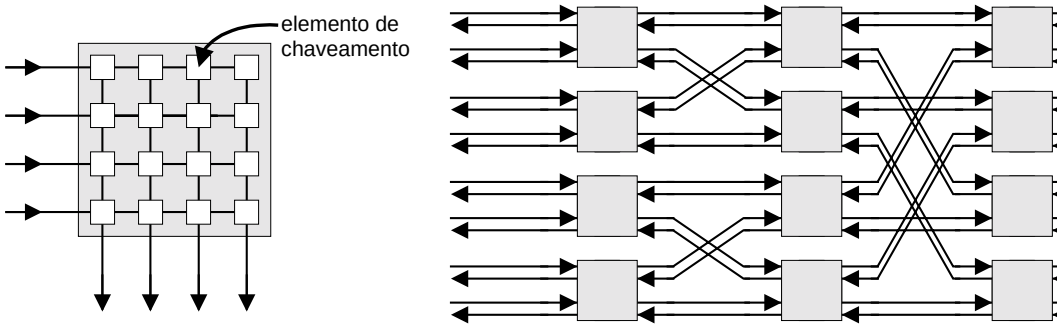
FIGURA 2.1 – Redes diretas: (a) grelha 2-D; (b) toróide 2-D; (c) hipercubo 3-D.

2.2.2 Redes indiretas

Nas redes indiretas, os roteadores não são acoplados a processadores, formando um elemento único, como nas redes diretas. Os nodos (neste caso, processadores, módulos de memória ou computadores completos) possuem uma interface para uma rede de roteadores. Cada roteador possui um conjunto de portas bidirecionais para ligações com outros roteadores e/ou com os nodos da máquina. Somente alguns roteadores possuem conexões para os nodos e apenas estes últimos podem servir de fonte ou destinatário de uma mensagem. A topologia da rede é definida pela estrutura de interconexão desses roteadores.

A topologia de uma rede indireta é vista como um grafo de roteadores e canais, não sendo necessário explicitar a presença dos nodos da máquina. Duas topologias clássicas de redes indiretas se destacam: o crossbar e as redes multiestágio. Para conexão indireta de N nodos, o crossbar é a topologia ideal, pois consiste de um único roteador com uma chave $N \propto N$. Embora seja mais econômico que uma rede direta completamente conectada (a qual necessitaria de N roteadores, cada um com uma chave crossbar $N \propto N$), o crossbar possui uma complexidade da ordem de N^2, o que torna o seu custo proibitivo para redes grandes.

Existem diversas topologias alternativas de redes indiretas. Nessas topologias, uma mensagem tem que atravessar alguns roteadores para chegar ao nodo destinatário. Em redes regulares, esses roteadores são usualmente idênticos e são organizados como um conjunto de estágios. Os estágios de entrada e de saída possuem ligações para os nodos e para os estágios internos da rede. Esses, por sua vez, são ligados aos seus estágios vizinhos através de padrões de conexão regulares. Essas redes são chamadas de redes multiestágio e podem ser caracterizadas, também, pelo número de estágios e pela forma como eles são arranjados. A Figura 2.2 mostra uma rede crossbar constituída por um roteador 4∝4 (quatro portas de entrada e quatro portas de saída) e uma rede multiestágio 8∝8 bidirecional do tipo borboleta (*butterfly*).



(a) (b)

FIGURA 2.2 – Redes indiretas: (a) crossbar 4∝4; (b) multiestágio 8∝8 bidirecional.

2.3 *Starvation, Livelock* e *Deadlock*

Uma rede de interconexão tem a função principal de oferecer o suporte físico necessário à comunicação entre os processos alocados aos nodos de processamento de um computador paralelo. A rede transporta pacotes através dos seus canais físicos e dos *buffers* dos roteadores. Uma comunicação é realizada com sucesso quando a informação enviada é devidamente recebida pelo destinatário. Entretanto, existem três situações que podem impedir que um pacote chegue ao seu destinatário: *starvation*, *livelock* e *deadlock*, as quais são descritas a seguir.

2.3.1 *Starvation*

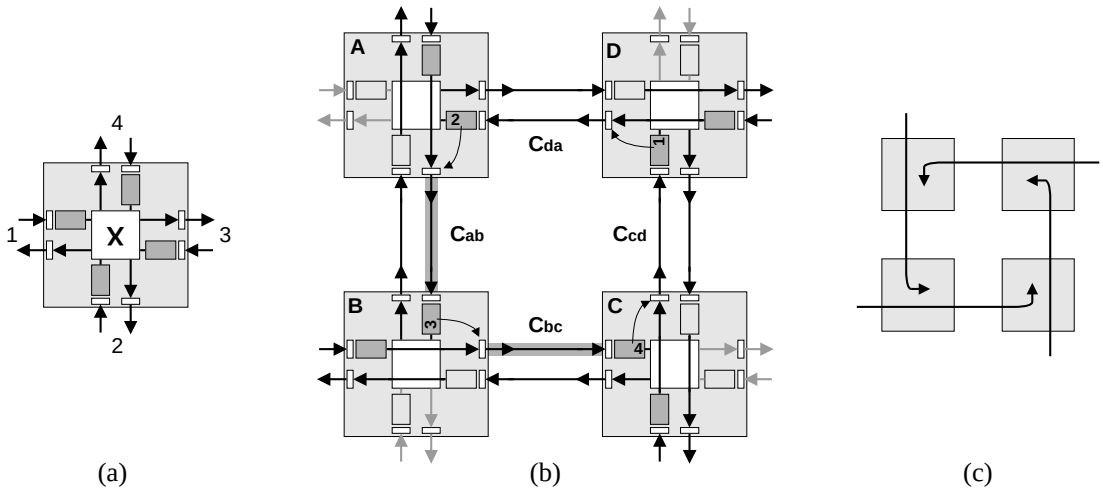
Muitas vezes, em um roteador, podem existir dois ou mais *buffers* de entrada com pacotes destinados a uma mesma saída. Quando isso acontece, uma política de arbitragem deve ser utilizada para selecionar qual dos *buffers* de entrada deve ser conectado à saída. Em geral, a arbitragem pode ser realizada através de critérios baseados em prioridades estáticas ou dinâmicas. Dependendo do esquema de arbitragem adotado, um pacote em um *buffer* de entrada, ao competir com outros pacotes por uma mesma saída, pode ficar bloqueado permanentemente se o recurso requisitado for sempre concedido a outros pacotes. Essa situação é conhecida pelo nome *starvation* e pode ser evitada pelo uso de um mecanismo de arbitragem adequado. Se alguns tipos de pacotes prioritários em relação a outros ocuparem toda a banda, a solução será reservar uma parte dessa banda para os pacotes de menor prioridade.

2.3.2 *Livelock*

Uma outra situação, conhecida pelo nome de *livelock*, ocorre quando um pacote mantém-se trafegando permanentemente pela rede porque os canais necessários para ele chegar ao nodo destinatário nunca se encontram disponíveis. Esse tipo de problema ocorre, tipicamente, em algoritmos de roteamento tolerantes a falhas, os quais utilizam caminhos não-mínimos para rotear os pacotes. A forma mais simples de se evitar o *livelock* é justamente permitir apenas o uso de caminhos mínimos, ou seja, os caminhos mais curtos até o destinatário. Entretanto, quando o uso de caminhos não-mínimos é vital para oferecer tolerância a falhas, pode-se prevenir o *livelock* através da limitação do número de operações de desvio através de caminhos não-mínimos.

2.3.3 *Deadlock*

O terceiro problema que pode impedir um pacote de atingir o seu destinatário, e o mais difícil de ser resolvido, é o *deadlock*. Esse problema ocorre quando existe uma dependência cíclica de recursos na rede, o que é melhor entendido pelo exemplo ilustrado na Figura 2.3.



(a) (b) (c)

FIGURA 2.3 – *Deadlock*: (a) roteador; (b) pacotes em *deadlock*; (c) dependência cíclica.

Na Figura 2.3.a é mostrado um modelo de roteador com quatro portas bidirecionais (1, 2, 3 e 4) conectadas a um núcleo crossbar (X). A Figura 2.3.b ilustra uma parte de uma rede de interconexão com quatro roteadores (*A*, *B*, *C* e *D*) interligados através dos canais *Cab*, *Cbc*, *Ccd* e *Cda*. A dependência cíclica ocorre porque existem quatro pacotes no segmento de rede ilustrado e cada um dos pacotes mantém um canal de enlace, mas requer outro canal já alocado para um segundo pacote. Por exemplo, o canal *Cab* (e o *buffer* a ele associado) foi reservado a um determinado pacote, o qual necessita do canal *Cbc* para avançar. Isso é indicado pelo número da porta de saída desejada (no caso, porta 3) no cabeçalho do pacote bloqueado no *buffer* de entrada da porta 4 do roteador *B*. A Figura 2.3.c ilustra com maior clareza essa dependência cíclica.

Existem três estratégias diferentes utilizadas para lidar com o *deadlock*: (i) prevenir, (ii) evitar e (iii) recuperar. Na prevenção, os recursos (canais e *buffers*) são garantidos a um pacote de modo que uma requisição nunca leve a rede ao *deadlock*. Isso pode ser obtido reservando-se todos os recursos necessários antes do início da transmissão do pacote e tornando estruturalmente impossível o *deadlock*. Na estratégia de evitar o *deadlock*, os recursos são requisitados na medida em que um pacote avança pela rede. Um recurso qualquer é garantido a um pacote apenas se sua alocação não levar a um estado global inseguro. Uma forma de evitar o *deadlock* é limitar as voltas que o algoritmo de roteamento pode realizar de modo a impedir a ocorrência de ciclos como o mostrado na Figura 2.3.c. Já na recuperação de *deadlock*, qualquer recurso é garantido a um pacote sem nenhuma verificação. Essa situação pode levar ao *deadlock* e, por isso, são necessários mecanismos para detectá-lo e para realocar alguns recursos entre os pacotes de modo a levantar o *deadlock*. Nesse caso, os pacotes que perdem recursos já alocados são, então, descartados ou re-roteados. A prevenção de *deadlock* é uma estratégia altamente conservadora e conduz a uma baixa utilização dos recursos da rede. A estratégia de evitar o *deadlock* é menos conservadora, pois os recursos são alocados na medida em que os pacotes avançam, melhorando a utilização da rede. Já a detecção de *deadlock* é uma estratégia otimista, a qual deve ser utilizada apenas quando os *deadlocks* são raros e suas conseqüências podem ser toleradas [DUA 97].

2.4 Roteamento

O roteamento é o método usado por um pacote (ou mensagem) para escolher um caminho através dos canais da rede. O desempenho da comunicação na rede de interconexão depende fortemente do algoritmo utilizado. Em geral, o algoritmo de roteamento visa atender a alguns objetivos específicos, os quais têm conseqüência direta em algumas propriedades da rede de interconexão, como:

a) Conectividade – capacidade de rotear pacotes de qualquer nodo fonte para qualquer nodo destinatário.

b) Liberdade de *deadlock* e *livelock* – capacidade de garantir que nenhum pacote ficará bloqueado ou circulando pela rede sem atingir o seu destinatário.

c) Adaptatividade – capacidade de rotear pacotes através de caminhos alternativos quando ocorrer congestionamento ou falha em algum componente do caminho em uso.

d) Tolerância a falhas – capacidade de rotear pacotes na presença de falhas em componentes. A tolerância a falhas pode ser obtida sem adaptatividade, roteando-se um pacote em duas ou mais fases.

Na literatura, existe um grande número de algoritmos de roteamento que visam atender a requisitos distintos. Esses algoritmos podem ser agrupados em diferentes classes segundo critérios específicos. Algumas taxionomias para os algoritmos de roteamento têm sido propostas [DUA 97, ASH 98] e, de uma forma geral, esses algoritmos podem ser classificados quanto ao momento da realização do roteamento, ao número de destinatários de um mesmo pacote, ao local onde as decisões de roteamento são tomadas, à implementação e à adaptatividade.

Quanto ao momento da realização do roteamento, o algoritmo pode ser classificado como dinâmico, se ele é realizado no tempo de execução da aplicação, ou estático, se o algoritmo de roteamento é executado no tempo de compilação da aplicação.

Quanto ao número de destinatários de um mesmo pacote, o algoritmo pode ser classificado como *unicast*, se cada pacote possuir um único destinatário, ou *multicast*, se um pacote pode ser encaminhado para múltiplos destinatários.

Considerando o local onde as decisões de roteamento são tomadas, o algoritmo de roteamento pode ser centralizado, fonte ou distribuído. No primeiro tipo, os caminhos são estabelecidos por um controlador central na rede. No roteamento fonte, o nodo emissor (ou fonte) define o caminho a ser seguido pelo pacote antes de injetá-lo na rede. Já no roteamento distribuído, o caminho é definido é pelos roteadores enquanto o pacote atravessa a rede.

No que tange a implementação, o algoritmo pode ser baseado em tabela, se o roteamento é feito a partir de uma consulta a uma tabela em memória, ou baseado em máquina de estado, se o roteamento é realizado a partir da execução de um algoritmo implementado em software ou em hardware.

Quanto à adaptatividade, o algoritmo de roteamento pode ser classificado como determinístico ou adaptativo. Em um algoritmo determinístico, o roteamento fornece sempre o mesmo caminho entre um determinado par fonte-destinatário. Já em uma abordagem adaptativa, o roteamento utiliza alguma informação a respeito do tráfego da rede e/ou do estado dos canais para evitar regiões congestionadas ou com falhas.

Os algoritmos adaptativos podem ainda ser classificados quanto à progressividade, à minimalidade e ao número de caminhos. No primeiro critério de classificação, o roteamento é considerado progressivo quando os cabeçalhos dos pacotes sempre avançam pela rede, reservando um novo canal a cada passo de roteamento, ou regressivo (*backtracking*) quando o cabeçalho de um pacote puder retornar pela rede, liberando canais previamente reservados. No segundo critério (minimalidade), o roteamento adaptativo é classificado como mínimo (ou *profitable*), se o algoritmo pode selecionar apenas canais de saída que aproximem cada vez mais o pacote do seu destinatário, ou não-mínimo (ou *misrouting*), se ele pode selecionar canais que levem o pacote a se afastar do seu destinatário. Por fim, no que tange o número de caminhos, um algoritmo adaptativo pode ser classificado como completo (ou total), se o algoritmo de roteamento puder utilizar todos os caminhos disponíveis, ou parcial, se apenas um subconjunto desses caminhos puder ser considerado.

Na Figura 2.4, é apresentado um resumo das classificações dos algoritmos de roteamento descritas nos parágrafos anteriores.

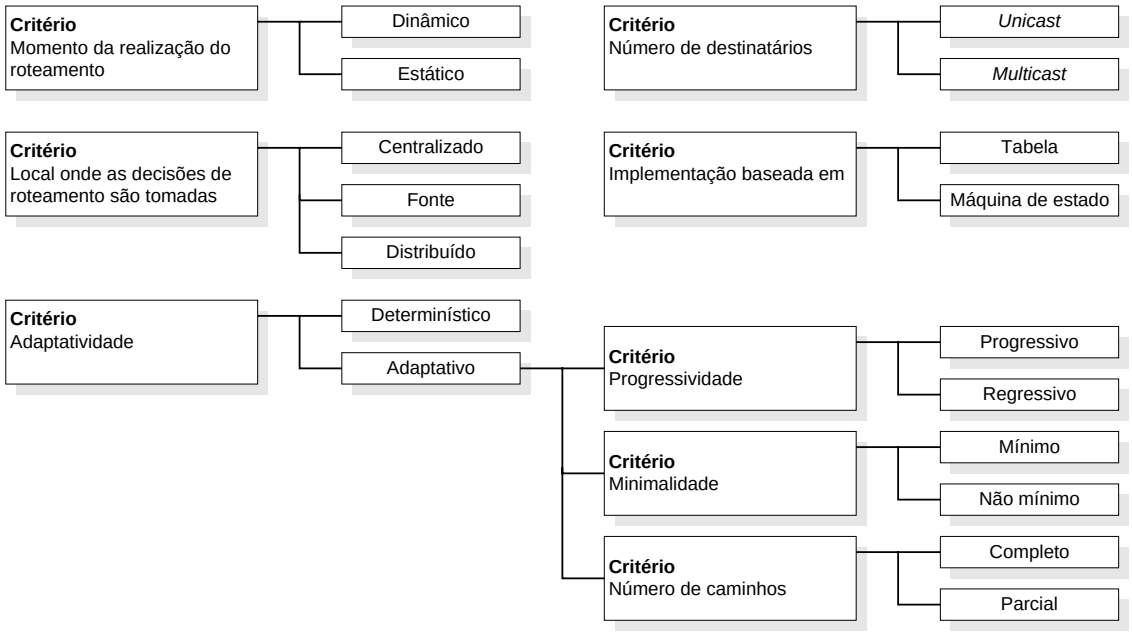


FIGURA 2.4 – Classificações para os algoritmos de roteamento.

2.5 Chaveamento

Em uma rede de interconexão, os dados são transferidos entre os nodos da rede através dos canais físicos que interligam esses nodos. O chaveamento define a forma pela qual esses dados são transferidos de um canal de entrada de um nodo para um dos seus canais de saída. Os dois tipos principais de técnicas de chaveamento são baseados ou no estabelecimento de um caminho completo entre o fonte e o destinatário da mensagem (circuito) ou na divisão das mensagens em pacotes os quais reservam seus caminhos dinamicamente na medida em que avançam em direção ao destinatário. A seguir, são descritas as principais técnicas de chaveamento.

2.5.1 Chaveamento por circuito

No chaveamento por circuito, um caminho físico fonte-destinatário completo é estabelecido para a transferência de uma mensagem, sendo mantido até o término da comunicação, a qual é realizada em duas etapas. Na primeira, o nodo fonte injeta, na rede, um cabeçalho de roteamento com o endereço do destinatário e com algumas informações de controle. Esse cabeçalho avança pela rede, reservando canais físicos para o estabelecimento do circuito. Se um canal desejado já estiver sendo ocupado por uma outra mensagem, o cabeçalho fica bloqueado até que esse canal lhe seja alocado. Quando o cabeçalho atinge o nodo destinatário, uma informação de reconhecimento é enviada ao nodo fonte através do caminho de retorno do circuito estabelecido. A partir desse momento, ocorre a segunda etapa da comunicação, a qual consiste na transferência dos dados da mensagem. O circuito pode ser desfeito durante o avanço do terminador da mensagem em direção ao destinatário. A cada roteador, a passagem desse terminador sinaliza que o recurso previamente alocado pode ser liberado.

O chaveamento por circuito tem sua origem em redes telefônicas e sua vantagem está no fato de que, uma vez estabelecido o caminho entre os nodos fonte e destinatário, a mensagem é transferida sem contenção, como se utilizasse um fio. O uso de *buffers* nos roteadores é mínimo, pois é necessária apenas uma posição para manter o cabeçalho enquanto um canal desejado encontrar-se indisponível. A maior desvantagem desta técnica é que os canais já alocados a um cabeçalho bloqueado em algum ponto da rede não podem ser utilizados por outro circuito, o que reduz a utilização da rede e aumenta o tempo para o estabelecimento dos caminhos. O uso do chaveamento por circuito se justifica apenas nos casos nos quais as mensagens são longas e pouco freqüentes. Alguns exemplos de computadores paralelos que utilizaram o chaveamento por circuito incluem o Intel iPSC/2 [NUG 88], o Intel iPSC/860 e o BBN GP 1000.

2.5.2 Chaveamento por pacote armazena e repassa (*store-and-forward*)

Quando as mensagens trocadas entre nodos da rede são curtas e freqüentes, o uso do chaveamento por circuito é injustificável, pois aumenta a contenção na rede e o tempo para o estabelecimento do circuito torna-se muito maior que o tempo envolvido na transferência dos dados. Alternativamente, pode-se dividir as mensagens em pacotes de comprimento fixo, cada um incluindo um cabeçalho com as informações necessárias para o seu roteamento pela rede. Os pacotes são enviados um a um e cada pacote reserva apenas os recursos necessários para avançar de nodo em nodo. Em cada canal de

entrada do roteador, deve ser incluído um *buffer* (tipicamente uma fila) com capacidade para armazenar um pacote inteiro. Quando um roteador recebe um pacote, ele é armazenado no *buffer* do canal de entrada pelo qual chegou ao roteador. Após, um circuito de roteamento identifica o destinatário do pacote e requisita um canal de saída. Essa requisição é escalonada por um árbitro que determina quando o cada canal de entrada será conectado ao canal de saída requisitado. Quando a conexão é realizada, os pacotes armazenados nos *buffers* de entrada são encaminhados para os roteadores adjacentes ou para o nodo de processamento local, caso este seja o destinatário. Por esse motivo, esta técnica é denominada armazena-e-repassa (do inglês, *store-and-forward* ou SAF).

Diferentemente do chaveamento por circuito, no qual os recursos são reservados mesmo quando o caminho não está completamente estabelecido, no chaveamento SAF, cada pacote aloca apenas os recursos necessários para avançar de nodo para nodo na rede. Porém, isso implica em uma sobrecarga adicional à comunicação, pois cada um dos pacotes deve transportar um cabeçalho de endereçamento e os roteadores precisam gastar um tempo para efetuar o roteamento individual de cada pacote. Desde que um pacote só pode ser repassado após ter sido completamente recebido, a latência de comunicação aumenta com o tamanho do pacote. Por fim, deve haver espaço em *buffer* suficiente em todos os roteadores para manter esses pacotes, o que aumenta o custo da rede.

O chaveamento SAF foi inicialmente utilizado em redes de comunicação de computadores. Os primeiros computadores paralelos que utilizaram este tipo de chaveamento foram: o Intel iPSC/1, o MIT *Tagged Dataflow Machine* [ARV 90] e o *Manchester Dynamic Dataflow Machine* [GUR 85, GUR 87].

2.5.3 Chaveamento por pacote com transpasse virtual (*virtual cut-through*)

O chaveamento por pacote com transpasse virtual (do inglês, *virtual cut-through* ou VCT) foi proposto por Kermani e Kleinrock [KER 79] como uma alternativa ao chaveamento SAF. Ele visa reduzir a latência na comunicação quando um pacote chega a um roteador e o canal desejado já encontra-se disponível. Na técnica SAF, quando isso ocorre, o pacote deve ser armazenado inteiramente para então ser retransmitido pelo canal de saída desejado, mesmo que ele esteja ocioso. No chaveamento VCT, quando o cabeçalho do pacote contendo as informações de roteamento chega a um roteador e o canal de saída desejado encontra-se disponível, o restante do pacote (carga útil e terminador) desvia o *buffer*, reduzindo a latência da comunicação. Um pacote só é armazenado em *buffer* se o canal desejado estiver alocado a outro pacote. Para isso, o roteador deve ter espaço em *buffer* suficiente para armazenar completamente o pacote bloqueado. No pior caso, quando a rede está muito carregada, o VCT se comporta como o chaveamento SAF. Um exemplo de aplicação do chaveamento VCT é o roteador *Chaos* [KON 91].

2.5.4 Chaveamento por pacote *wormhole*

O chaveamento por pacote *wormhole* (ou, simplesmente, chaveamento *wormhole*), introduzido por Dally e Seitz [DAL 86], é uma variação do chaveamento VCT e tem como objetivo principal reduzir a quantidade de *buffer* necessária para manter pacotes bloqueados na rede. Um pacote é dividido em *flits* que avançam pela

rede em um modo *pipeline*. Um flit[2] é a menor unidade de dados sobre a qual é realizado o controle de fluxo e pode ser pequeno e ter tantos bits quanto um phit (largura do canal físico de dados), ou ser tão grande quanto um pacote. Em geral, um flit tem o tamanho de um a quatro phits, para conter, no mínimo, a informação necessária ao roteamento do pacote, ou seja, o cabeçalho.

Nas redes baseadas nesta técnica, os *buffers* dos roteadores têm capacidade para armazenar poucos flits, de modo que os flits de um pacote bloqueado são mantidos em diferentes roteadores na rede se o tamanho do pacote for maior que o espaço livre no *buffer*. Como a informação de roteamento é incluída no flit de cabeçalho, os flits de dado devem seguir o flit do cabeçalho através da rede. Como resultado disso, não é possível realizar a multiplexação de flits de diferentes pacotes em mesmo canal lógico. Um pacote deve atravessar completamente um canal antes de liberá-lo para outro pacote. Essa é uma das principais desvantagens desta técnica de chaveamento, pois a probabilidade de ocorrer o *deadlock* é maior. Entretanto, essa restrição pode ser contornada através do uso de canais virtuais, onde múltiplos canais lógicos compartilham um único canal físico, como será descrito na seção sobre controle de fluxo.

A grande vantagem do chaveamento *wormhole* é que os requisitos de *buffer* são menores que os das abordagens SAF e VCT, possibilitando a construção de roteadores pequenos e rápidos. Em geral, o chaveamento *wormhole* é o mais vantajoso com relação à utilização da rede e ao custo dos roteadores, sendo o mais usado atualmente. Exemplos de computadores paralelos com redes de interconexão baseadas no chaveamento *wormhole* incluem o Cray T3D [KES 93], o Cray T3E [SGI 99], o IBM SP [STU 94], o Meiko CS-2 [BEE 94], o SGI Origin 2000 [LAU 97] e o Intel/ASCI Option Red [MAT 98].

2.5.5 Modelos de latência

Em [DUA 97], são apresentados modelos de latência para as diferentes técnicas de chaveamento. Nesses modelos, são desconsiderados os termos S e C utilizados em (2.1), os quais correspondem, respectivamente, à sobrecarga de comunicação dos nodos fonte e destinatário e ao atraso devido à contenção na rede (conforme definições apresentadas previamente na Tabela 2.1). Em outras palavras, os modelos assumem que a rede trabalha sem carga e leva-se em conta apenas a ocupação do canal e os atrasos de roteamento e chaveamento, ou seja, os termos O e A_{RC} em (2.1).

Na Tabela 2.3, são apresentados os parâmetros utilizados nos modelos de latência. É assumido que a mensagem (ou o pacote) possui um cabeçalho com tamanho igual a W bits (largura física do canal) e uma carga útil de tamanho igual a L bits (ou L/W palavras de 1 phit). Os canais dos enlaces da rede operam a uma freqüência de B Hz, com um tempo de propagação igual a t_w. Em cada roteador, são gastos t_r segundos para rotear o cabeçalho mais t_s segundos para propagação de um phit (palavra de W bits). Os nodos fonte e destinatário da mensagem são separados por D enlaces, sendo que o número de roteadores percorridos pela mensagem é igual a D+1.

[2] *FLIT = FLow control unIT* (unidade de controle de fluxo)

TABELA 2.3 – Parâmetros utilizados nos modelos de latência.

Parâmetro	Unidade	Definição
L	Bits	Tamanho da carga útil da mensagem
W	Bits	Tamanho do phit
B	Hz	Freqüência de transmissão
T_w	segundos	Tempo de propagação nos canais dos enlaces (= 1/B)
T_s	segundos	Tempo de propagação nos canais internos do roteador
T_r	segundos	Tempo para o roteador fazer o roteamento
D		Número de enlaces entre os nodos fonte e destinatário

Na Tabela 2.4, são resumidos os modelos de latência apresentados em [DUA 97]. A tabela inclui também o requisito de memorização mínima de cada técnica.

TABELA 2.4 – Modelos de latência.

Técnica de chaveamento	Latência sem contenção (desconsiderando a sobrecarga)	Memorização mínima
Circuito	$D \propto [\ t_r + 2.(t_s + t_w)\ ] \ + \ t_w \propto (L/W)$	1 flit
Store-and-forward	$D \propto \{\ t_r + (t_s + t_w) \propto [(L + W)/W]\ \}$	1 pacote
Virtual cut-through	$D \propto (\ t_r + t_s + t_w) + \max\{t_s, t_w\} \propto [L/W]$	1 pacote
Wormhole	$D \propto (t_r + t_s + t_w) + \max\{t_s, t_w\} \propto [L/W]$	1 flit

2.6 Controle de Fluxo

Uma rede de interconexão pode ter muitos canais e *buffers*. Os pacotes[3] que trafegam pela rede competem por esses recursos para avançarem em direção aos seus destinatários. Quando um pacote não pode prosseguir pois algum recurso que ele necessita já está sendo utilizado por um outro pacote, é dito que ocorreu uma colisão de recurso (ou, simplesmente, uma colisão). Nesse caso, alguma política deve ser usada para decidir se o pacote deve ser descartado, bloqueado no lugar onde está, recebido e armazenado temporariamente ou, então, desviado para um outro caminho. Essa política é denominada controle de fluxo e lida com a alocação de canais e *buffers* para um pacote enquanto o mesmo trafega pela rede [NI 93].

Em geral, todas as redes de interconexão realizam controle de fluxo em nível de enlace. Em cada terminal de um enlace, o nodo ou roteador a ele conectado pode possuir um *buffer* para o armazenamento do dado em transferência. Se, por algum motivo, o *buffer* de entrada do receptor estiver cheio, o transmissor deve manter o dado a ser enviado em um *buffer* local até que o receptor esteja pronto para realizar a comunicação. Um mecanismo de controle deve então ser usado para bloquear a saída de

[3] Como a maioria das redes de interconexão atuais utiliza variações do chaveamento por pacotes (principalmente o *wormhole*), a partir deste ponto, neste capítulo, será dada a preferência ao uso do termo pacote ao invés de mensagem.

dados do transmissor e o receptor deve enviar uma informação de controle ao transmissor notificando-lhe quando estiver pronto para receber novos dados.

A forma mais simples de controle de fluxo em nível de enlace é aquela na qual o receptor possui um *buffer* FIFO de entrada e uma linha de retorno ao transmissor para informar se há espaço disponível nesse *buffer*. Essa informação pode ser interpretada pelo transmissor como sendo um crédito e ele só pode enviar um dado se tiver crédito disponível. Uma abordagem semelhante baseia-se no protocolo de *handshake*, no qual o emissor informa a intenção de enviar um dado ao receptor através de uma linha de validação e o receptor confirma a disponibilidade de espaço em *buffer* para receber esse dado através de uma linha de reconhecimento (*acknowledgement*). Outras abordagens de controle de fluxo em nível de enlace incluem o uso de *slack buffers* e de canais-virtuais, discutidos a seguir.

2.6.1 Controle de fluxo baseado em *slack buffer*

O controle de fluxo baseado em *slack buffer* [SEI 93] pode ser visto como o controle de nível em um reservatório com marcas de nível baixo e de nível alto, no qual o conteúdo do mesmo deve ser mantido entre esses limites, na chamada região de histerese. Sempre que o conteúdo do *buffer* ultrapassar um desses limites, um pacote de controle (geralmente um caracter *Stop* ou *Go*) deve ser enviado pelo receptor ao transmissor de modo a regular o fluxo de dados, seja pela suspensão (caracter *Stop*) ou pelo reestabelecimento (caracter *Go*) do envio de dados pelo transmissor.

O *slack buffer* serve para ocultar a latência associada à transmissão dos pacotes de controle de fluxo. A região de histerese entre as marcas de nível alto e de nível baixo é dimensionada de modo a evitar o envio excessivo de caracteres de controle, o que consumiria uma largura de banda demasiada no canal oposto ao de entrada de dados.

Este tipo de controle de fluxo é particularmente adequado para redes que usam chaveamento *wormhole* e um exemplo de rede que utiliza essa estratégia é a Myrinet [BOD 95]. Nessa rede, o *slack buffer* possui uma região de histerese de 16 bytes e as demais regiões possuem 32 bytes cada uma.

2.6.2 Controle de fluxo baseado em canais virtuais

Em redes de interconexão com chaveamento *wormhole*, cada roteador possui um conjunto de *buffers* e uma chave. Para cada canal físico, há um *buffer* (ou fila) de entrada que armazena os flits recebidos para que depois os mesmos sejam direcionados a uma ou outra porta de saída em função do roteamento realizado. Se o primeiro pacote desse *buffer* for bloqueado por conta de uma colisão e o pacote for maior do que o espaço livre disponível no *buffer*, o canal físico também restará bloqueado e nenhum outro pacote poderá utilizar esse canal. Esse problema é conhecido como bloqueio de cabeça de linha (HOL – *Head-of-Line blocking*).

Se o *buffer* de entrada for organizado em filas de profundidade menor, denominadas canais-virtuais, obtém-se uma coleção de filas que podem ser alocadas independentemente umas das outras. Com isso, se um canal-virtual estiver bloqueado por algum motivo, o canal físico poderá ainda ser usado por um outro canal-virtual, resolvendo o problema do bloqueio HOL e aumentando a utilização do canal físico.

Essa organização pode ser chamada de multi-via, pois existem múltiplas vias virtuais em uma mesma via física. Da mesma forma, a organização anterior pode ser chamada de mono-via. Os canais-virtuais foram originalmente introduzidos por Dally e Seitz [DAL 87] para resolver o problema do *deadlock* em redes com chaveamento *wormhole*.

2.6.3 Controle de fluxo baseado em créditos

O controle de fluxo baseado em créditos é um tipo de controle que produz uma rede de comunicação que nunca descarta dados. Uma transmissão só é realizada quando o receptor tem espaço suficiente em seu *buffer* para armazenar o pacote a ser recebido [KOR 99]. Para isso, o seguinte protocolo é utilizado: (i) o receptor envia ao transmissor uma informação de créditos relativa ao espaço disponível para recepção de dados em seu *buffer*; (ii) o transmissor, por sua vez, inicia uma transmissão apenas quando possuir crédito suficiente; e (iii) o crédito do transmissor diminui com o envio de dados e aumenta com o recebimento de informações de controle apropriadas.

Esta técnica de controle de fluxo pode ser associada a canais físicos com uma única via ou com múltiplas vias. O controle de fluxo baseado em créditos com múltiplas vias baseia-se no controle com canais virtuais. Essa estratégia é utilizada na rede do multiprocessador SGI Origin 2000 [GAL 97], na qual cada canal físico possui quatro canais-virtuais. Após a inicialização do sistema, todos os receptores creditam seus pares com base no tamanho dos *buffers* de recepção dos canais virtuais. A informação a respeito do crédito disponível é enviada através do canal oposto ao canal de entrada de um mesmo enlace bidirecional.

2.7 Memorização

Todo roteador com chaveamento por pacote (seja do tipo SAF, VCT ou *wormhole*) deve ser capaz de armazenar aqueles pacotes destinados a saídas que já estejam sendo utilizadas por outros pacotes e, então, realizar o controle de fluxo de modo a evitar a perda de dados recebidos em cada canal de entrada. Isso exige a implementação de algum esquema de memorização para a manutenção dos pacotes bloqueados dentro do roteador.

Um roteador ideal deveria dispor de uma capacidade de armazenamento infinita e garantir que um pacote armazenado não seja bloqueado por outros pacotes quando a sua saída for liberada. Entretanto, nos casos reais, a memória do roteador é limitada e, dependendo da estratégia utilizada, o bloqueio de um pacote por outro é inevitável. A organização dos *buffers* de memória (independentes ou compartilhados) e suas posições (na entrada, na saída ou centralizados) afetam o desempenho do roteador. A seguir, são apresentadas algumas das estratégias de memorização possíveis de serem utilizadas em um roteador.

2.7.1 Memorização centralizada compartilhada

Nessa abordagem, um *buffer* centralizado é utilizado para armazenar os pacotes bloqueados em todos os canais de entrada e o espaço de endereçamento é dinamicamente distribuído entre os pacotes bloqueados. Esse *buffer* é denominado CBDA (*Centrally-Buffered, Dynamically-Allocated*) e, no pior caso, deve oferecer uma largura de banda igual à soma das larguras de banda de todas os canais. Ou seja, em um roteador $N \propto N$, o *buffer* deve possuir $2N$ portas de modo a permitir N acessos simultâneos de leitura e N acessos simultâneos de escrita. Por outro lado essa abordagem oferece uma utilização do espaço de memória melhor do que aquelas proporcionadas pelas abordagens nas quais esse espaço é previa e estaticamente alocado aos canais de entrada. Um problema pode surgir no caso de uma saída estar em uso por uma entrada e outra entrada com pacotes destinados a essa saída continuar a receber dados, levando ao enchimento do *buffer* e afetando as outras comunicações. Esse problema pode ser evitado pela limitação do espaço alocável a cada canal. Um exemplo de roteador que utiliza um esquema de memorização centralizada compartilhada é o roteador Vulcan do computador IBM SP-2 [IBM 99a, IBM 99b].

2.7.2 Memorização na entrada

Nesta abordagem, o espaço de memória é distribuído sob a forma de partições entre os canais de entrada do roteador. Essas partições são implementadas através de *buffers* independentes, cada um com portas de entrada próprias. Existem várias alternativas possíveis de implementação, entre as quais podem ser destacadas as estratégias *buffer* FIFO, *buffers* SAFC, *buffers* SAMQ e *buffers* DAMQ, as quais são comparadas em [TAM 92] e descritas brevemente a seguir.

Buffers FIFO

Essa é a alternativa mais simples e de menor custo. Cada *buffer* FIFO (*First-In, First-Out*) possui um espaço de memória fixo onde os dados são lidos na mesma ordem em que são escritos. Porém, o seu grande inconveniente é o problema do bloqueio HOL, no qual um pacote bloqueado na saída do *buffer* impede o avanço de outro pacote que esteja atrás dele e para o qual a saída desejada esteja disponível, o que resulta em uma subutilização das portas de saída. Na Figura 2.5, é mostrado um roteador com quatro *buffers* FIFO nos canais de entrada e um crossbar $4 \propto 4$.

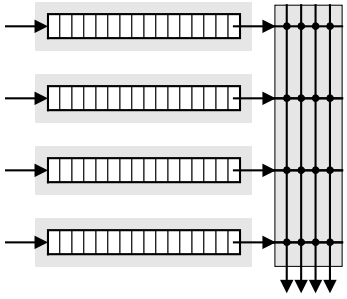


FIGURA 2.5 – Roteador com quatro *buffers* FIFO.

Buffers SAFC

Uma alternativa para contornar o problema do bloqueio HOL, que ocorre nos *buffers* FIFO, consiste em dividir cada *buffer* de entrada em *N* partições com tamanho igual a 1/*N* do tamanho do *buffer* original, conforme é mostrado na Figura 2.6.a. Cada uma dessas partições deve ser atribuída estaticamente a um canal de saída e, por isso, esta estratégia é denominada SAFC (*Statically Allocated, Fully Connected*). Isso requer o uso de um crossbar $N^2 \propto N$ ou, então, de *N* crossbars $N \propto 1$ ao invés de um único crossbar $N \propto N$. Esta estratégia apresenta uma série de inconvenientes. Primeiramente, há um custo adicional referente ao controle do crossbar e ao controle dos *N buffers* por porta de entrada. Em segundo lugar, a taxa de utilização dos *buffers* é limitada a 1/*N* do espaço de armazenamento total, ou seja, ela é pior que a dos *buffers* FIFO. Por fim, o controle de fluxo é mais complexo, pois deve ser realizado para cada *buffer* de entrada, e exige um pré-roteamento dos pacotes recebidos para que os mesmos sejam direcionados à partição correspondente à saída a ser requisitada [TAM 92].

Buffers SAMQ

A fim de simplificar o gerenciamento do crossbar, as saídas dos *buffers* de entrada atribuídos a um mesmo canal de saída podem ser multiplexadas, resultando em uma estratégia chamada SAMQ (*Statically Allocated Multi-Queue*), ilustrada na Figura 2.6.b. Essa estratégia elimina alguns dos inconvenientes da SAFC, pois reduz o custo do crossbar. Entretanto, ela ainda mantém os outros dois problemas relacionados à utilização dos *buffers* e ao controle de fluxo.

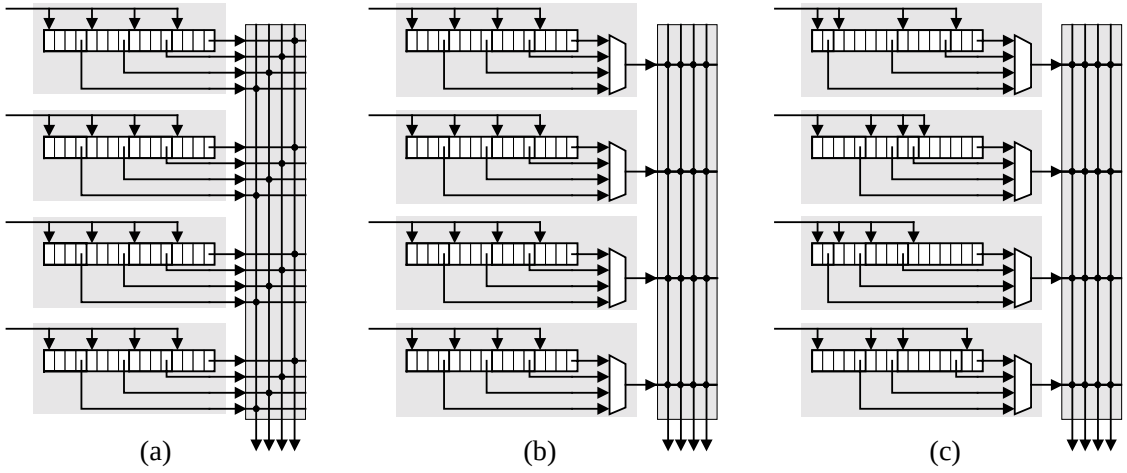


FIGURA 2.6 – Roteador com quatro *buffers*: (a) SAFC; (b) SAMQ; (c) DAMQ.

Buffers DAMQ

Com o objetivo de remediar os problemas das estratégias anteriores, Tamir [TAM 92] propôs uma arquitetura de *buffer* denominada DAMQ (*Dynamically-Allocated, Multi-Queue*), a qual é representada na Figura 2.6.c. Nessa estratégia, o espaço de memorização associado a um canal de entrada é particionado dinamicamente entre os canais de saída conforme a demanda dos pacotes recebidos. Dessa forma, evita-se o problema do bloqueio HOL dos *buffers* FIFO, aumenta-se a utilização do espaço de

memória disponível e o controle de fluxo é mais simples que nas estratégias SAFC e SAMQ, não requerendo pré-roteamento. Entretando, o gerenciamento do *buffer* DAMQ, baseado em listas encadeadas, torna a sua implementação física mais complexa.

2.7.3 Memorização na saída

Esta estratégia consiste em particionar o espaço de memorização entre as saídas, sendo que as partições podem ser implementadas como *buffers* FIFO. O problema, nesse caso, é que cada *buffer* deve ser capaz de suportar a demanda simultânea das *N* entradas. O *buffer* pode ser implementado com *N* portas de escrita ou com uma porta de escrita operando a uma velocidade *N* vezes maior que a das entradas. Além disso, esta estratégia requer um controle de fluxo interno entre portas de entrada e de saída do roteador.

2.8 Arbitragem

Enquanto o roteamento determina qual canal de saída dever ser usado por um pacote que chega a um canal de entrada, a arbitragem define qual canal de entrada (ou *buffer* de entrada) poderá utilizar um determinado canal de saída (ou *buffer* de saída). Em outras palavras, o roteamento é um mecanismo de seleção de saída e a arbitragem é um mecanismo de seleção de entrada.

A arbitragem é fundamental para a resolução de conflitos decorrentes da existência de múltiplos pacotes competindo por um mesmo canal de saída. O mecanismo de arbitragem deve ser capaz de resolver esses conflitos, selecionando um dos pacotes com base em algum critério e sem levar qualquer pacote a sofrer *starvation*, ou seja, ficar indefinidamente esperando por uma oportunidade para avançar em direção ao nodo destinatário.

O árbitro de um roteador pode ser implementado de forma centralizada ou distribuída. Na abordagem centralizada, mostrada na Figura 2.7.a, os mecanismos de roteamento e de arbitragem são implementados em um único módulo. Esse módulo recebe os cabeçalhos dos pacotes, executa o roteamento e determina o canal de saída a ser utilizado por cada pacote. A partir disso, ele faz a arbitragem, selecionando os pacotes a serem conectados em cada saída, configura o crossbar e habilita os *buffers* selecionados para os mesmos avançarem seus pacotes. Essa abordagem é utilizada em árbitros que procuram maximizar a utilização do crossbar, realizando uma arbitragem global, a qual considera todos os pacotes que estejam prontos para serem transmitidos nos *buffers* dos canais de entrada, bem como o estado presente dos canais de saída. Entretanto, essa centralização impõe maiores restrições quanto à capacidade de roteamento de pacotes no tempo.

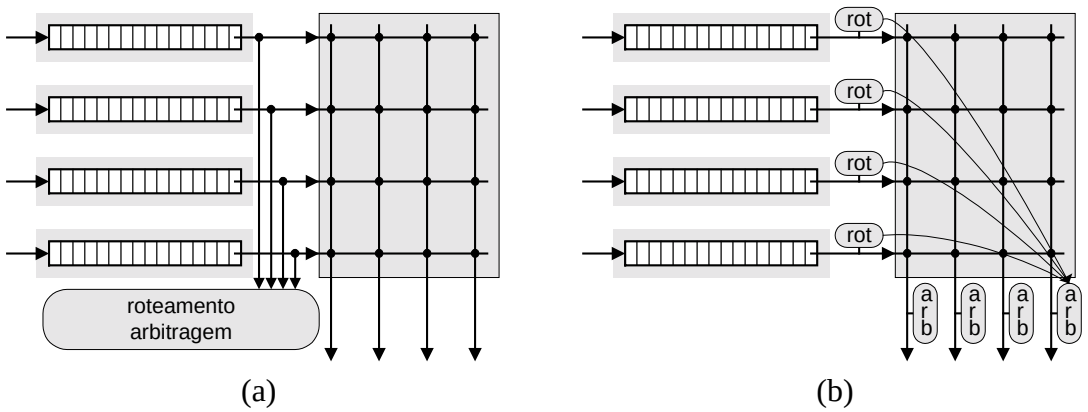


FIGURA 2.7 – Abordagens para implementação de árbitros: (a) centralizada; (b) distribuída.

Na abordagem distribuída, ilustrada na Figura 2.7.b, o roteamento e a arbitragem são realizados de forma independente para cada porta bidirecional do roteador. Cada uma das portas possui um módulo de roteamento associado ao seu canal de entrada e um módulo de arbitragem no seu canal de saída. Essa abordagem aumenta a capacidade de roteamento de pacotes mas apresenta limitações quanto à sua aplicação em redes baseadas em algoritmos de roteamento adaptativo. Isso ocorre pois o mecanismo de roteamento pode determinar múltiplas saídas candidatas e então enviar requisições a todos os árbitros selecionados [CUL 97]. Se pelo menos dois dos árbitros requisitados selecionarem a mesma entrada, uma saída será escolhida e as saídas preteridas ficarão ociosas, quando poderiam ter selecionado outras entradas, aumentando a taxa de utilização do crossbar. Para esse problema ser resolvido é preciso que o árbitro tenha uma visão global, o que só é possível em uma abordagem centralizada. Portanto, a implementação distribuída parece ser mais adequada em redes baseadas em roteamento determinístico.

O mecanismo de arbitragem pode ser baseado em diferentes critérios: prioridades estáticas, prioridades rotativas (*round-robin*), escalonamentos por idade (ou *deadline*), FCFS (*First-Come-First-Served*), LRS (*Least Recently Served*), entre outros. O esquema de prioridades estáticas é o mais simples de ser implementado, pois consiste de um circuito codificador de prioridade estático. Nesse esquema, cada requisição considerada pelo árbitro tem um nível de prioridade fixo, de modo que, dependendo do tráfego da rede, uma requisição pode ser sempre preterida em favor das outras de maior prioridade, vindo a sofrer *starvation*. Para solucionar esse problema, um esquema de prioridades dinâmicas deve ser utilizado. Contudo, é também desejável que ele ofereça uma distribuição equilibrada do recurso arbitrado entre seus requisitantes. Por exemplo, em um árbitro com prioridades rotativas, um requisitante selecionado em um ciclo de arbitragem recebe o menor nível de prioridade para o ciclo de arbitragem seguinte. Se existirem N requisitantes para o mesmo circuito, todos os requisitantes terão oportunidades iguais para utilizar o recurso requisitado. A implementação de um árbitro com prioridades rotativas é baseada no uso de um codificador de prioridade programável. Em [GUT 99], Gupta e McKeown descrevem diferentes alternativas para implementação desse tipo de codificador de prioridade programável.

2.9 Exemplos

Nesta seção, são exemplificadas algumas redes de interconexão para multiprocessadores. Elas possuem alguns pontos em comum, os quais demonstram quais soluções de projeto são mais apropriadas para a obtenção de uma alta largura de banda e de uma baixa latência de comunicação. O objetivo é apenas de ilustrar as definições e características apresentadas neste capítulo. Maiores detalhes sobre as redes aqui apresentadas podem ser encontrados nas referências indicadas na Tabela 2.5.

TABELA 2.5 – Exemplos de redes de interconexão para multiprocessadores.

Característica	**Intel/ASCI**	**SGI Origin**	**IBM SP**	**Cray T3E**
Topologia	Grelha 3-D	Hipercubo	Multiestágio	Toróide 3-D
Roteamento	Fonte e determinístico	Distribuído e determinístico	Fonte e determinístico	Fonte e determinístico; Adaptativo
Chaveamento	*Wormhole*	*Wormhole*	*Wormhole**	*Wormhole*
Controle de fluxo	Créditos e canais virtuais	Créditos e canais virtuais	Créditos	Créditos e canais virtuais
Memorização	Entrada	Entrada	Entrada e Centralizada	Entrada
Arbitragem	(Não disponível)	*Deadline*	LRS	Round-Robin
Referências	[MAS 98] [CAJ 96]	[LAU 97] [GAL 97]	[IBM 99a] [IBM 99b]	[SGI 99]

* BFW = *Buffered Wormhole Switching*

Analisando-se as redes mostradas na tabela, observa-se que as topologias com estruturas regulares são preferidas porque facilitam o roteamento. Em uma topologia *n*-dimensional, como a grelha, por exemplo, o roteamento fonte é implementado com bastante facilidade a partir de deslocamentos relativos nos eixos da rede. A maioria dessas redes utiliza topologias diretas, com exceção da rede do IBM SP, a qual foi baseada em uma rede indireta do tipo multiestágio.

Quanto ao roteamento, todas as redes utilizam roteamento dinâmico, sendo que a abordagem mais utilizada é a baseada no roteamento fonte e determinístico, o qual oferece o menor custo de implementação. Nas redes com topologia direta, o ordenamento por dimensão é utilizado. Nas redes com topologia indireta (IBM SP), o cabeçalho inclui o endereço da porta de saída em cada chave no caminho do pacote. Essa abordagem possui o inconveniente de aumentar o tamanho do cabeçalho com a distância do nodo destinatário, o que não ocorre no roteamento ordenado pela dimensão. O grande problema do roteamento fonte é que ele não oferece adaptatividade para falhas na rede. A rede do SGI Origin oferece uma espécie de adaptatividade estática. Ao invés de utilizar roteamento fonte, ele usa roteamento distribuído. O cabeçalho do pacote inclui o endereço do destinatário, o qual é utilizado para uma consulta em tabela em cada roteador a fim de determinar a porta de saída. Essas tabelas incluem rotas estáticas as quais podem ser reconfiguradas em função de alterações na rede. O grande problema do uso de tabelas distribuídas é que, com o aumento do tamanho da rede, o número de entradas em cada tabela cresce, aumentando, também, o tempo de consulta à tabela e piorando a latência de roteamento. No SGI Origin, esse problema foi contornado pela organização da rede em uma forma hierárquica e pelo particionamento da tabela em

duas partes, sendo que apenas uma delas é consultada em cada roteamento. A rede do SGI Origin inclui ainda um mecanismo de roteamento fonte, o qual é utilizado para a configuração das rotas nas tabelas distribuídas e para o gerenciamento da rede. Ainda sobre o roteamento no SGI Origin, a liberdade de *deadlock* é garantida através da programação adequada das tabelas de roteamento, evitando-se a ocorrência de ciclos nas rotas estáticas. Por fim, das redes descritas acima, apenas o Cray T3E inclui uma alternativa de roteamento adaptativo com a inclusão de um canal virtual adicional exclusivamente para esse fim.

Com relação ao método de chaveamento, todas as redes descritas utilizam chaveamento *wormhole*. Isso se justifica pois essa é a técnica que consegue oferecer, ao mesmo tempo, uma baixa latência de chaveamento e um baixo custo de implementação, já que os *buffers* são menores e os flits de um pacote podem ser bloqueados em diferentes roteadores. Entretanto, um pacote bloqueado reserva um grande número de recursos da rede, o que aumenta o risco do *deadlock*. A solução mais utilizada para minimizar esses efeitos é a implementação de múltiplos canais virtuais em cada canal físico. Tipicamente, costuma-se utilizar quatro canais virtuais, como, por exemplo, no Intel/ASCI e no SGI Origin. O Cray T3E adota ainda um quinto canal virtual, o qual é usado na implementação de roteamento adaptativo. O IBM SP não implementa canais virtuais, mas, alternativamente, ele inclui uma fila central utilizada para remover pacotes bloqueados na rede. Dependendo das condições de tráfego, o mecanismo de chaveamento se comporta como o *wormhole* ou como o chaveamento por pacote e, por esse motivo, ele é denominado de chaveamento *buffered wormhole*.

Quanto ao controle de fluxo, a técnica mais utilizada é a baseada em créditos combinada com a de canais virtuais. Nas redes do Intel/ASCI, do SGI Origin e do Cray T3E, vários créditos são transferidos de uma única vez, informando o total de espaço disponível no *buffer* de recepção. Já na rede do SP2, cada crédito é transferido individualmente. As duas abordagens têm as suas vantagens e desvantagens. Na primeira, as linhas de dados do canal oposto do enlace podem ser utilizadas para transferir os créditos, o que dispensa o uso de linhas adicionais e permite que mais de um crédito seja transferido de uma única vez. Entretanto isso consome uma parte da largura de banda do enlace e, em alguns casos, como no Origin 2000, uma banda lateral é reservada mesmo que não seja necessário atualizar os créditos, degradando a largura de banda disponível. Na segunda abordagem, um sinal é reservado para o controle de fluxo, o que dispensa o uso do canal de dados do enlace. Contudo, esse sinal único permite que apenas um crédito seja transferido por vez. Para incrementar a quantidade de créditos em uma mesma transferência seria necessário incluir fios adicionais, o que aumentaria o custo do enlace.

Quanto à memorização todas as redes utilizam memorização na entrada, sendo que o IBM SP inclui ainda um *buffer* central para armazenar os *flits* de pacotes bloqueados e evitar o bloqueio HOL nos *buffers* de entrada. Em geral, os espaço de armazenamento implementado em cada roteador situa-se entre 3,5 Kbytes e 6 Kbytes.

Finalmente, quanto ao árbitro, todas as redes acima utilizam algum tipo de arbitragem com prioridades dinâmicas de modo a garantir a ausência de *starvation*.

2.10 Considerações

Neste capítulo, foram apresentados os conceitos fundamentais sobre redes de interconexão e foram discutidas as principais características e técnicas de projeto. O texto procurou oferecer apenas uma visão geral sobre o assunto, evitando um detalhamento excessivo. Referências mais completas, como [DUA 97], discutem esses temas com mais profundidade. Conforme pôde ser observado, as principais topologias de redes de interconexão utilizadas atualmente são as redes diretas ortogonais (grelha, toróide e hipercubo) e as redes indiretas com múltiplos estágios. Com relação ao roteamento, foi apresentada uma classificação para os diferentes tipos de algoritmo disponíveis conforme alguns critérios estabelecidos. Os algoritmos de roteamento mais simples e, portanto mais baratos são também os mais utilizados. Quanto ao chaveamento, foram apresentadas as principais técnicas utilizadas, sendo que a técnica de chaveamento por pacote *wormhole* é a mais adotada atualmente pois permite a construção de roteadores menores, mais baratos e mais rápidos. Quanto ao controle de fluxo, foram vistas diferentes técnicas, sendo a baseada em créditos a mais utilizada, combinada ou não com o controle de fluxo por canais virtuais. Quanto à memorização, foi mostrado que podem ser utilizados *buffers* na entrada, na saída ou *buffers* centralizados compartilhados. Porém, em geral, a primeira abordagem é a mais adotada. Além disso, foram feitos alguns comentários a respeito da arbitragem, sua importância para evitar o problema de *starvation* e os tipos de critérios utilizados. Outros dois aspectos relativos ao projeto de uma rede de interconexão consite na garantia da ausência de *livelock* e *deadlock*, sendo que esses problemas são evitados através do uso de uma estratégia de roteamento adequada.

No capítulo a seguir, será apresentada uma revisão da literatura a respeito de sistemas integrados e do estado da arte das arquiteturas de comunicação utilizadas nesses sistemas.

3 A Comunicação em Sistemas Integrados

O advento das tecnologias de processo submicrônico tem permitido o projeto de sistemas completos em um único chip, os quais são denominados sistemas integrados, sistemas-em-ship ou SoCs (*Systems-on-Chip*)[4]. Neste capítulo, primeiramente, são tratados diferentes aspectos relacionados aos sistemas integrados de modo a estabelecer um cenário para o foco principal do texto. Após, enfatizam-se as questões relacionadas à comunicação em SoCs, com uma discussão sobre as alternativas de arquiteturas de comunicaçao, vantagens e desvantagens de cada alternativa e exemplos representativos de arquiteturas utilizadas nos SoCs atuais e daquelas que estão sendo desenvolvidas com o objetivo de atender aos requisitos de comunicação dos futuros sistemas integrados.

3.1 Os Sistemas Integrados em um Único Chip

As metodologias de projeto de sistemas integrados são, em geral, baseadas no reuso de blocos previamente projetados e verificados, os quais são denominados núcleos (em inglês, *cores*). Conforme definem Gupta e Zorian [GUP 97], um núcleo é um bloco de circuito pré-projetado e pré-verificado que pode ser usado na construção de uma aplicação maior ou mais complexa em uma pastilha de material semicondutor (*chip*), sendo que os diferentes tipos de núcleo podem ser classificados em função da forma como são implementados e disponibilizados ao projetista do sistema integrado[5]:

a) *Soft-core* – consiste de uma descrição em linguagem de descrição de hardware (HDL . *Hardware Description Language*) que pode ser mapeada para diferentes processos de fabricação;

b) *Firm-core* – contém mais informação, normalmente um *netlist* ou até informações sobre o posicionamento e o roteamento; e

c) *Hard-core* – inclui o leiaute e informações referentes à temporização do circuito para uma determinada tecnologia e já está pronto para ser utilizado no sistema.

Normalmente, os núcleos são produtos de tecnologia, software e experiência que são sujeitos a patentes e a direitos autorais. Em outras palavras, um núcleo representa uma propriedade intelectual (IP . *Intelectual Property*) que o construtor licencia ao usuário. Por isso, os núcleos são também denominados blocos de propriedade intelectual, ou, em inglês, *IP Blocks*.

Um sistema integrado não é constituído apenas por núcleos. Conforme Madisetti e Shen [MAD 97], ele é formado por um conjunto de núcleos integrados através de uma arquitetura de comunicação, incluindo, ainda, um controlador e uma interface para o mundo externo (Figura 3.1). Entretanto, conforme será visto posteriormente a própria arquitetura de comunicação pode ser um bloco reutilizável. No que tange a terminologia, na literatura, a arquitetura de comunicação de um SoC é também denominada rede de comunicação ou rede de interconexão (seja ela chaveada ou não).

[4] Neste texto, será usado tanto o termo “sistema integrado” quanto o acrônimo “SoC” (SoCs no plural).
[5] Nessa classificação, mantém-se a terminologia em inglês apresentada por Gupta e Zorian [GUP97].

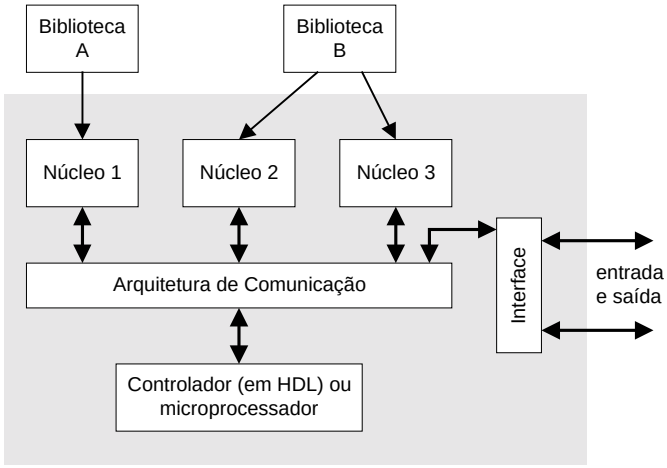


FIGURA 3.1 . Arquitetura genérica de um sistema integrado [MAD 97 – p.44].

Os núcleos utilizados em um SoC podem ser novos ou herdados de projetos já existentes, e podem também ser obtidos de uma ou mais bibliotecas. Se os núcleos são provenientes de fontes independentes, a integração e o teste podem ser difíceis, requerendo, possivelmente, o reprojeto do núcleo para adequá-lo a um protocolo de interface comum.

Para facilitar o projeto de SoCs, existem fabricantes que oferecem ferramentas de projeto e bibliotecas completas com núcleos baseados em uma interface definida para uma arquitetura de comunicação disponível juntamente com a biblioteca. Esse tipo de metodologia é denominado “projeto baseado em plataforma” (PBD – *Platform-Based Design*) [VAH 2001, KEU 2000].

Uma plataforma é uma arquitetura de hardware e software específica para um domínio de aplicação [DEM 2001, DUT 2001, PAU 97], mas altamente parametrizável. Ela inclui ferramentas de apoio ao projeto e bibliotecas com componentes de hardware (eg. processadores, memórias, dispositivos de E/S) e de software (eg. *drivers* de dispostivos, sistemas operacionais de tempo-real e códigos de aplicação), os quais podem ser parametrizados para atender aos requisitos do sistema alvo. A idéia básica é que uma arquitetura adequada e eficiente para uma aplicação também será adequada e eficiente para muitas aplicações similares. O uso da mesma arquitetura para o desenvolvimento de uma aplicação não apenas acelera o projeto da aplicação como também reduz o seu tempo de verificação [KUM 2002].

Segundo Bergamaschi [BER 2002], os SoCs baseados em plataforma são desenvolvidos a partir de uma arquitetura de referência que satisfaz um conjunto de restrições arquiteturais e permite o reuso de componentes de hardware e de software. Na prática, um SoC baseado em uma plataforma é construído removendo-se componentes desnecessários da arquitetura de referência e adicionando-se componentes necessários que não são oferecidos nessa arquitetura. O custo inicial do desenvolvimento da plataforma é alto. Os projetos subsequentes, entretanto, desenvolvidos com base na mesma plataforma, necessitarão muito menos recursos pela reutilização de componentes de hardware e de software. Além da redução do custo de projeto, o tempo para a realização do mesmo também é reduzido devido ao reuso dos componentes.

3.1.1 Iniciativas da comunidade em sistemas integrados

É justamente no sentido de evitar os problemas de interligação entre núcleos de diferentes fabricantes que a VSIA[6] (*Virtual Socket Interface Alliance*) tem desenvolvido padrões abertos para o interfaceamento de núcleos, como, por exemplo, o padrão VCI (*Virtual Component Interface*) [VSI 2000]. O VCI consiste de uma especificação de interface para interligação ponto-a-ponto entre núcleos de um SoC, independentemente da arquitetura da comunicação utilizada para interconectar esses núcleos. O estabelecimento desse padrão é baseado na premissa de que somente será possível explorar completamente o poder do reuso de projeto se forem suportadas descrições comuns de propriedade intelectual.

Uma segunda iniciativa na mesma direção do VCI é o OCP[7] (*Open Core Protocol*) [OCP 2001]. Desenvolvido pela empresa Sonics Inc.[8], ele define uma interface de alto desempenho entre núcleos, a qual independe da arquitetura de comunicação utilizada. O OCP apresenta, como diferencial ao VCI, a inclusão de sinais de teste (*scan* e JTAG) e de controle (eg. interrupção) em sua interface, os quais não são contemplados pelo padrão proposto pela VSIA.

O OCP e o VCI exigem que o núcleo disponha de uma interface compatível com o padrão ou que o mesmo seja envolvido por uma lógica de adaptação de protocolo, caso o núcleo possua uma interface diferente. Essa lógica é denomina *wrapper* e no contexto da comunicação ela pode ser entendida como um adaptador de protocolo de comunicação. Caso a arquitetura de comunicação alvo utilize uma interface diferente daquela usada pelo núcleo (baseado ou não em um protocolo padrão), é necessário um adaptador adicional para fazer a compatibilização das diferentes interfaces Esse conjunto de adaptadores exige uma área adicional no chip e acrescenta uma latência na comunicação. Contudo, seu uso facilita em muito a integração de um sistema, sobretudo se os núcleos forem provenientes de diferentes fabricantes.

Um SoC genérico no qual o padrão VCI é utilizado como interface (I/F) de integração é mostrado na Figura 3.2. Conforme é ilustrado, os fornecedores dos núcleos podem disponibilizar versões compatíveis com o padrão VCI ou acompanhadas de adaptadores de protocolo, caso o núcleo possua uma interface diferente da VCI (proprietária ou não). De maneira semelhante, o fornecedor da arquitetura de comunicação pode disponibilizar ao integrador do sistema os adaptadores necessários para realizar a conversão entre a interface da sua arquitetura e o padrão VCI. A vantagem dessa abordagem para o fornecedor da arquitetura de comunicação é que ela permite que a mesma arquitetura seja reutilizada na integração de núcleos com diferentes interfaces. Ao mesmo tempo, isenta o fornecedor do núcleo de disponibilizar uma versão para cada arquitetura de comunicação à qual seus produtos podem ser conectados. Sob o ponto de vista do integrador, essa abordagem evita que o mesmo tenha que conhecer várias interfaces e implementar os adaptadores necessários à compatibilização dos núcleos e da arquitetura de comunicação a serem integrados em seu sistema.

---

[6] A VSIA é uma coalisão internacional de companhias formada em 1996 com o objetivo de estabelecer uma visão unificada para a indústria de sistemas integrados, bem como definir os padrões técnicos necessários para a integração de núcleos de diferentes fontes. Informações disponíveis por WWW em http://www.vsi.org (acesso em 20 de julho de 2003).

[7] Informações disponíveis por WWW em http://www.ocpip.org/home (acesso em 20 de julho de 2003).

[8] Informações disponíveis por WWW em http://www.sonicsinc.com (acesso em 20 de julho de 2003).

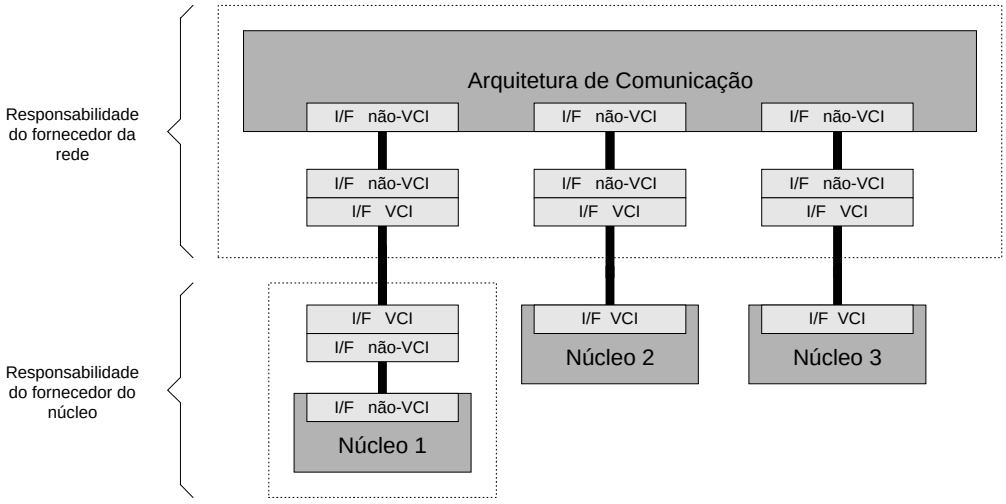


FIGURA 3.2 . Um sistema genérico integrado via padrão VCI (onde I/F = interface).

Além das iniciativas acima destacadas, existem comunidades organizadas a respeito do tema sistemas integrados. Por exemplo, a TechOnLine mantém uma comunidade *on-line* dedicada ao projeto de SoCs, tecnologias de processo, componentes virtuais e propriedade intelectual em silício. Essa comunidade é denominada SoCnet[9] e, em sua publicação *on-line*, oferece boletins e anúncios, calendário de eventos, artigos e banco de dados para consulta.

3.1.2 O teste em sistemas integrados

Um dos principais aspectos relacionados ao desenvolvimento de um sistema integrado diz respeito ao teste. Em [GUP 97], Gupta e Zorian discorrem sobre a questão do teste em sistemas integrados baseados no reuso de núcleos. Dois tipos de teste são geralmente necessários, o teste interno para a verificação do núcleo e o teste externo para a verificação da integração do sistema no que tange à interconexão entre os núcleos integrados no SoC.

Como o integrador do sistema tem pouco ou nenhum conhecimento a respeito da estrutura interna do núcleo utilizado (exceto para os *soft-cores*), é o construtor do núcleo quem deve preparar o teste interno. Esse teste interno consiste de estruturas DFT (*Design For Test*) e dos padrões de teste associados, os quais podem ser gerados internamente por BIST (*Built-In Self Test*) ou externamente através de vetores de teste. Os vetores são aplicados ou capturados na periferia do núcleo sob teste. Porém, o construtor do núcleo determina os requisitos desse teste interno sem conhecer o processo e a aplicação alvos. Ele não sabe qual método de teste deve adotar, o tipo ou tipos de falhas (estática, dinâmica ou paramétrica) ou, ainda, o nível de cobertura de falhas desejado. Como resultado, o construtor pode não fornecer um teste de qualidade adequada. Além disso, o construtor do núcleo precisa preparar um teste interno que seja descrito apropriadamente, portado e que esteja pronto para ser utilizado, ou seja, para a interoperabilidade com o sistema de teste. Para isso, o teste deve ser descrito em um padrão aceito comumente.

[9] Informações disponíveis por WWW em http://www.techonline.com (acesso em 20 de julho de 2003).

Um núcleo típico é montado em um SoC de modo que o acesso físico direto à sua periferia não é normalmente disponível. Como resultado disso, os engenheiros de teste precisam de um mecanismo de acesso eletrônico, ou seja, lógica e fios adicionais para conectar a periferia do núcleo aos recursos de teste. Para estabelecer um mecanismo de acesso eletrônico, o projetista deve fornecer dois elementos básicos: um conjunto de caminhos de acesso e uma rede de controle para habilitar e desabilitar o caminho de acesso desejado. Esse caminho de acesso contém as estruturas DFT necessárias de todas as periferias dos núcleos para o chaveamento entre os modos normal e de teste. Existem vários tipos de caminhos de acesso, sendo que o mais utilizado é baseado na serialização dos padrões de teste. Por exemplo, vários fornecedores utilizam a cadeia de *boundary scan* IEEE Std. 1149.1 [TEX 97].

Além dos modos de teste e de operação normal, um sistema baseado em núcleos requer, freqüentemente, vários outros modos relacionados ao teste. Um deles é o modo de teste externo para a detecção de falhas de interconexão (estáticas ou dinâmicas) entre núcleos. Adicionalmente à aplicação e à captura de padrões de teste durante os modos de teste interno e externo, os engenheiros de teste podem usar caminhos de acesso periférico para isolar os núcleos. Isso leva o núcleo a um estado seguro, se for necessária uma isolação no lado da entrada do núcleo, ou coloca as saídas do núcleo no modo de terceiro estado. Algumas vezes, outros modos de isolação do núcleo são necessários, seja para colocá-lo nos modos de espera, de baixa dissipação de potência ou de desvio, ou para controlar uma fonte de corrente do núcleo para teste I_{DDQ}.

3.1.3 Bibliotecas de núcleos para o desenvolvimento de sistemas integrados

Existem diversos fabricantes de circuitos integrados que oferecem bibliotecas de núcleos para a construção de SoCs. Em geral essas bibliotecas são desenhadas visando uma família de processadores e os seus núcleos utilizam uma interface padrão baseada no processador de referência. A biblioteca também pode incluir núcleos para a arquitetura de comunicação, geralmente um barramento hierárquico multi-mestre.

A IBM, por exemplo, provê a biblioteca BlueLogic com núcleos de processadores e periféricos para a família IBM PowerPC4*xx* (eg. 401, 405 e 440), além de núcleos para DSPs (eg. IBM C54XDSP) e processadores de outros fabricantes (eg. ARM ARM7TDMI). Inclui também diversos núcleos para os diferentes tipos de controladores de entrada e saída, como controladores de rede (eg. Ethernet 10/100), enlaces de alta velocidade (eg. HSL), interfaces seriais (eg. UART, USB), interfaces de barramento (eg. AGP4X, PCI), compressão de dados (eg. JPEG), entre outros. A biblioteca inclui ainda uma arquitetura de comunicação padrão, denominada CoreConnect, e ferramentas de projeto que facilitam a integração do sistema [IBM 2000b].

3.1.4 Ferramentas de auxílio ao projeto de sistemas integrados

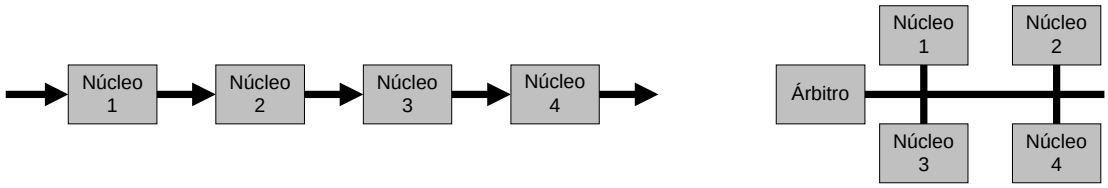
A implementação de um sistema complexo em um SoC é ainda uma tarefa bastante árdua. Bergamaschi e Lee [BER 2000] enumeram as diversas razões que dificultam o projeto de um SoC. Primeiramente, o projetista deve escolher os componentes a serem utilizados, definir os requisitos de desempenho do sistema e, ainda, conhecer as características, as funcionalidades e as interfaces dos componentes a serem utilizados. Uma vez completada essa etapa, é necessário realizar a integração do sistema. Porém, mesmo que cada núcleo tenha sido pré-projetado e pré-verificado individualmente, o projeto físico e a verificação do sistema são dificultados pelo surgimento de efeitos imprevisíveis, como ruídos e acoplamentos capacitivos. Além disso, a falta de padrões estabelecidos e adotados amplamente pela indústria torna ainda mais difícil a integração de núcleos fornecidos por diferentes fabricantes. Por fim, há ainda o problema da integração hardware-software, a qual é geralmente feita em uma fase posterior do projeto e afeta diretamente o tempo de lançamento do produto no mercado. Ainda, segundo Bergamaschi e Lee, há poucas ferramentas de auxílio ao projeto de sistemas integrados que permitam a integração e a configuração facilitadas de núcleos em um sistema. A complexidade dos SoCs atuais e a falta de ferramentas de alto nível apropriadas tornam difíceis a reusabilidade e a aplicação direta de núcleos.

Recentemente, tem-se verificado uma série de esforços para solucionar ou minimizar as limitações acima descritas. Com relação ao interfaceamento, conforme já foi citado, existem algumas propostas de interface padrão (como VCI e OCP) que ainda não estão sendo amplamente utilizadas. Contudo, acredita-se que uma delas venha a se tornar um padrão de fato no projeto de sistemas integrados. Quanto às ferramentas de apoio ao projeto, existem diversos trabalhos de pesquisa em desenvolvimento, porém, a discussão desses trabalhos foge do escopo deste texto.

3.2 Arquiteturas de Comunicação dos SoCs Atuais

Nos sistemas integrados atuais, a interconexão entre os núcleos é realizada através de canais ponto-a-ponto ou de canais multi-ponto (Figura 3.3). Na primeira abordagem, tipicamente utilizada em sistemas baseados no modelo de comunicação de fluxo de dados (eg. codificadores/decodificadores de áudio ou de vídeo), os núcleos são interligados por canais dedicados, sendo que cada canal é constituído por um conjunto de condutores ligando dois núcleos. Já na segunda abordagem, geralmente usada em sistemas orientados ao modelo de comunicação de espaço de endereçamento (eg. processador–memória–E/S), a arquitetura de comunicação é estruturada sob a forma de um barramento compartilhado ao qual os núcleos do sistema são conectados [GUE 2000a]. Essas duas abordagens podem ser comparadas quanto ao paralelismo, consumo de energia[10], freqüência de operação, escalabilidade, área e reusabilidade.

[10] Nos textos técnicos em inglês o termo mais utilizado é *power consumption*, o qual, segundo Rabaey [RAB96], determina as quantidades de energia consumida e de potência (ou calor) dissipada por um circuito.



(a) (b)

FIGURA 3.3 . Arquiteturas de comunicação atuais: (a) ponto-a-ponto; (b) multi-ponto.

Quanto ao paralelismo, os canais ponto-a-ponto são independentes entre si e permitem que múltiplas comunicações sejam realizadas simultaneamente. Já em um barramento, apenas uma comunicação pode ocorrer a cada momento, e os múltiplos núcleos do sistema concorrem pelo uso do barramento, o qual possui um controlador central para escalonar a sua utilização.

Com relação ao consumo de energia, ele é maior em canais com fios mais longos e que alimentam múltiplas cargas. Isso ocorre porque esses fatores impactam diretamente na capacitância parasita do fio e o consumo de energia aumenta com essa capacitância [RAB 96]. Em uma arquitetura baseada em canais ponto-a-ponto, os fios tendem a ser mais curtos e com uma única carga. Já em uma arquitetura multiponto, os fios tendem a ser mais longos e carregados por todos os núcleos a ela conectados. Além disso, o consumo de energia é proporcional à freqüência de mudança de estado. Como um barramento é responsável por todas as comunicações, a sua taxa de utilização é próxima dos 100% [LIA 2000] e a atividade de chaveamento nos seus fios é intensa. Por outro lado, em canais ponto-a-ponto, a taxa de utilização é muito menor e, portanto, o consumo de energia em cada segmento é bem inferior.

No que tange a freqüência de operação, os tempos de transição (subida e descida) nos fios aumentam com a carga capacitiva associada. Esses tempos definem o ciclo de operação e, portanto, a freqüência de operação dos fios de um canal de comunicação. Quanto maior for a carga capacitiva, maiores serão os tempos de transição e menor será a freqüência de operação dos canais. Com base no que foi visto no parágrafo acima, pode-se avaliar que as arquiteturas com canais ponto-a-ponto podem operar em freqüências maiores que as arquiteturas baseadas em canais multi-ponto, pois as cargas capacitivas aos quais os fios são submetidos são menores. Para lidar com esse problema nos barramentos, são utilizam *drivers* maiores e que tenham capacidade de suprir uma maior corrente elétrica de modo a reduzir os tempos subida e de descida dos fios. Contudo, isso aumenta ainda mais o consumo de energia no barramento.

Quanto à escalabilidade, uma arquitetura de comunicação é dita escalável se a sua largura de banda cresce com o tamanho do sistema [HWA 93]. Em uma arquitetura com canais ponto-a-ponto, cada núcleo adicionado ao sistema requer a implementação de novos canais para a comunicação com os núcleos já existentes. A largura de banda desses novos canais é então agregada à largura de banda já disponível. Em uma arquitetura multiponto, tipicamente, um núcleo adicionado ao sistema é conectado aos canais já compartilhados pelos demais núcleos sem aumentar a largura de banda da arquitetura. Pelo contrário, esse núcleo irá competir pela banda disponível, sendo que a largura de banda por núcleo será ainda menor. Além disso, esse novo núcleo contribuirá

para aumentar a carga capacitiva, aumentando o consumo de energia e reduzindo a freqüência de operação.

Com relação à área, o custo de uma arquitetura de comunicação é dado pelo custo do metal para a implementação dos canais físicos e pelo custo de silício para a realização de funções lógicas como arbitragem, decodificação de endereços, chaveamento, entre outras. Em uma abordagem ponto-a-ponto, cada canal físico é constituído por fios unidirecionais e pode ser otimizado independentemente (eg. largura do canal físico, inserção de *drivers* etc). Já em uma abordagem multi-ponto, a largura física dos canais do barramento deve ser dimensionada de modo a atender aos requisitos de latência e de largura de banda de todos os núcleos do sistema, sendo que a otimização de um barramento compartilhado com múltiplas cargas pode ser muito difícil [HU 2002]. Em geral, em um sistema baseado em canais ponto-a-ponto, os núcleos que se comunicam são posicionados o mais próximo possível uns dos outros de modo a reduzir o comprimento dos canais de comunicação. Já em sistema baseado em uma arquitetura multi-ponto, os canais de comunicação do barramento devem alcançar todos os núcleos do sistema, sendo que, tipicamente, assume-se que o comprimento total do barramento é igual à metade do perímetro do chip [LAN 2000]. Disso poder-se-ia concluir que o barramento possui um custo de metal superior ao de uma arquitetura baseada em canais ponto-a-ponto. Contudo, na abordagem ponto-a-ponto, cada núcleo pode ter múltiplos canais de comunicação para outros núcleos do sistema, o que aumenta os custos de metal e de silício devido à implementação de múltiplas portas. Além disso, dependendo da complexidade da estrutura lógica de comunicação, pode ser difícil posicionar os núcleos de modo que todos os canais dos núcleos que se comunicam tenham um comprimento mínimo, tornando o roteamento dos fios uma tarefa ainda mais árdua ao projetista. Nesses casos, torna-se necessário o uso de metodologias que permitam a obtenção automática de arquiteturas de comunicação ponto-a-ponto otimizadas quando ao custo e, também, ao consumo de energia [HU 2002].

Ainda no que tange a área, é sabido que o custo de silício em uma abordagem ponto-a-ponto é dado pelas estruturas associadas às múltiplas portas de comunicação em cada núcleo, como *drivers* (ou *buffers*) de sinal e filas de sincronismo de comunicação (ou *buffers* de memorização). Eventualmente, esse custo de silício pode ser ainda maior se for realizada a multiplexação do canal lógico de modo a reduzir o número de fios no canal físico. Já em um sistema baseado em barramento, cada núcleo possui uma única porta de comunicação cuja interface deve ser compatível ou adaptada ao protocolo do barramento. Como em um barramento existem múltiplos núcleos que podem injetar sinais nos seus fios, é preciso implementar *buffers tri-state* nas interfaces dos núcleos com o barramento, além de eventuais registradores ou filas de sincronismo para o armazenamento temporário dos dados transferidos. O barramento exige o estabelecimento de um protocolo para acesso ao meio físico, o qual é normalmente implementado sob a forma de um árbitro central que recebe requisições dos núcleos do tipo mestre, os quais possuem habilidade para iniciar uma transferência de dados, aplica um critério de prioridade para selecionar uma entre múltiplas requisições e envia um sinal de confirmação ao mestre selecionado. Como em um barramento todos “escutam” o que um “fala”, é preciso que seja implementado um mecanismo de seleção do núcleo destinatário da mensagem (o escravo). Essa seleção pode ser feita de diferentes formas (centralizada no árbitro ou distribuída entre os escravos), mas, em qualquer uma, ela implica em um custo adicional de silício. Em resumo, a área de silício de uma abordagem ponto-a-ponto depende do número de núcleos, do número de portas por

núcleo e do tamanho dessas portas. Já em um barramento, o número de portas por núcleo e o tamanho delas é fixo, porém o aumento do número de núcleos implica no aumento do custo do árbitro e do(s) circuito(s) de seleção. Dadas essas dependência, uma avaliação de qual abordagem ocupa a maior área depende da avaliação do tamanho e da complexidade do sistema.

A partir das características avaliadas acima, poder-se-ia concluir que a melhor solução para interconexão de núcleos em SoCs consistiria no uso de canais ponto-a-ponto, pois oferecem paralelismo, consomem menos energia, operam em freqüências maiores, oferecem escalabilidade e, dependendo do sistema, podem ocupar menos área. Contudo, resta ainda avaliar uma última característica das arquiteturas de comunicação para SoCs: a reusabilidade. Essa característica é na verdade um requisito fundamental no mercado de microeletrônica, o qual é altamente competitivo e exige que o tempo para lançamento de um produto (do inglês, *time-to-market*) seja o menor possível de modo a garantir o seu sucesso e a conquista do maior número de clientes. O reuso de núcleos pré-projetados e pré-verificados reduz o tempo de projeto e ajuda a atingir essa meta pois, a cada novo sistema, o projeto do SoC consistirá na integração de núcleos já validados. Além disso, o reuso permite amortizar os custos de projeto de um componente entre vários sistemas baseados nesse componente [KUM 2002]. Nessa direção, se a arquitetura de comunicação e os componentes a ela associados forem também reutilizáveis, a integração do sistema será ainda mais facilitada, ajudando a atingir os requisitos de tempo e custo de projeto do sistema. Contudo, uma arquitetura de comunicação baseada em canais ponto-a-ponto dedicados é fundamentada em uma abordagem *ad-hoc* e o seu reuso em sistemas diferentes é bastante restrito. Além disso, com o aumento da complexidade dos SoCs, o tempo para o projeto da arquitetura de comunicação tende a ser ainda maior [DAL 2001], o que inviabiliza essa abordagem como solução de interconexão em sistemas complexos com requisitos restritos de tempo de projeto.

A disponibilidade de uma arquitetura de comunicação reutilizável é de vital importância para abreviar o tempo de projeto de um sistema. Arquiteturas multi-ponto, como o barramento, são totalmente reutilizáveis e, por conta disso, têm sido a abordagem preferida para a construção dos SoCs atuais, apesar das suas desvantagens em relação à abordagem ponto-a-ponto. Como exemplo, podem ser citadas diversas arquiteturas de comunicação em barramento usadas atualmente: OMI PI-Bus, IBM CoreConnect, ARM AMBA, Silicore Corp. WISHBONE, PalmChip CoreFrame, Mentor Graphics FISPbus, Motorola IP Interface, Sonics SiliconBackplane, Sonics Micronetwork, IDT Peripheral Bus e University of Manchester MARBLE[11].

Para contornar as limitações inerentes de uma arquitetura em barramento, a principal solução utilizada atualmente consiste no uso de hierarquias com múltiplos barramentos. Por exemplo, através do uso de uma arquitetura com dois barramentos interconectados por um circuito ponte, conforme ilustrado na Figura 3.4, consegue-se aumentar o número de transações simultâneas, reduzir a carga capacitiva de cada barramento (e, portanto, a energia consumida e o ciclo de operação) e aumentar a largura de banda do sistema. Além disso, os dois barramentos são implementados com características diferentes. Um é destinado à interconexão de núcleos com requisitos de

[11] A página "*Summary of SoC Interconnection Buses*", mantida pela empresa Silicore, apresenta resumos e *links* para páginas a respeito de cada um desses barramentos. Informações disponíveis por WWW em http://www.silicore.net/uCbusum.htm (acesso em 20 de julho de 2003).

grande largura de banda (como processadores e controladores de memória) e o outro é usado para a conexão de núcleos lentos ao sistema (como controladores de portas seriais e paralelas). Portanto, isso requer um mapeamento adequado dos núcleos ao sistema de modo que aqueles mais lentos fiquem isolados daqueles mais rápidos, evitando que núcleos rápidos fiquem parados esperando que os núcleos lentos completem suas operações de comunicação.

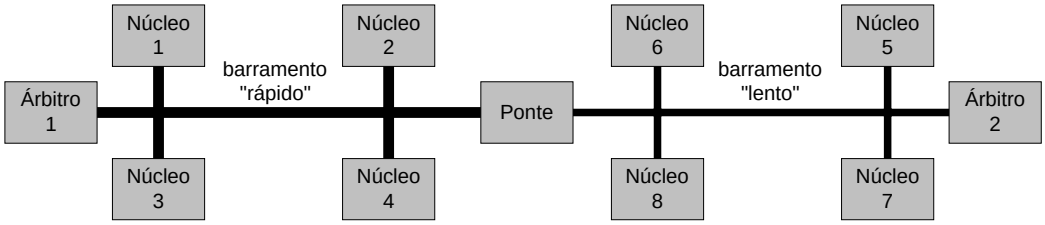


FIGURA 3.4 . Exemplo de arquitetura hierárquica com dois barramentos.

Essa solução pode ser ainda melhorada, porém existem limites. Por exemplo, podem ser implementados barramentos mais largos de modo a aumentar o número de bits transferidos a cada ciclo, mas um barramento muito largo não é eficiente para a transferência de palavras de dado menores. Como solução, pode-se concatenar mais de uma transação, mas isso leva a um aumento da latência. Uma outra alternativa para se obter uma maior largura de banda seria o aumento da freqüência do relógio, mas isso não pode ser feito arbitrariamente devido a restrições elétricas decorrentes das capacitâncias parasitas associadas às unidades conectadas ao barramento e também ao longo comprimento de suas linhas [GUE 2000a].

Como os barramentos constituem na abordagem mais utilizada para a interconexão de núcleos nos SoCs atuais, nas seções a seguir, são apresentadas definições básicas sobre esses barramentos e alguns exemplos representativos de barramentos utilizados em SoCs comerciais.

3.2.1 Definições básicas sobre barramentos para sistemas integrados

Um barramento de sistema para SoCs é constituído por vias exclusivas para dado, endereço e controle, além de uma unidade de controle responsável pela arbitragem do barramento, sendo, por isso, também denominado “árbitro”, conforme mostra a Figura 3.5. O árbitro é necessário nos sistemas onde mais de um componente precisa iniciar a transferência de dados entre ele e outro componente do sistema. Ao componente que tem a habilidade de iniciar uma transferência, dá-se o nome de “mestre”, enquanto que ao seu parceiro, dá o nome de “escravo”. Em sistemas com múltiplos mestres, cada um dos mestres envia uma requisição ao árbitro toda vez que precisa realizar uma transferência de dados com algum escravo. Se houverem várias requisições concorrentes, o árbitro deve utilizar algum critério para eleger uma delas e então notificar o mestre selecionado através de uma linha de confirmação. O critério de arbitragem pode ser baseado em prioridades estáticas ou dinâmicas (eg. aleatórias, fila circular etc). Desde que o árbitro é também um recurso compartilhado, o incremento do número de mestres no sistema aumenta o atraso na arbitragem [GUE 2000a].

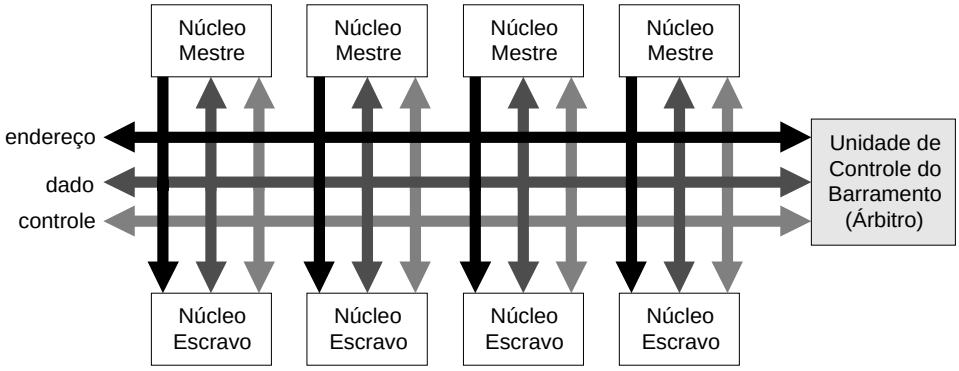


FIGURA 3.5 . Arquitetura genérica de um SoC baseado em barramento.

Além de cumprir o papel de árbitro, em alguns sistemas, a unidade de controle do barramento também integra a função de decodificar o endereço fornecido pelo mestre da transação em curso e selecionar o escravo que deve responder a essa transação. Essa centralização da arbitragem e da decodificação é uma das características específicas que distinguem os barramentos de sistema dos barramentos de periféricos, os quais podem ser mais facilmente estendidos por utilizarem decodificação distribuída, na qual o escravo é responsável pela decodificação do endereço. Contudo, existem barramentos em chip para interconexão de periféricos que utilizam decodificação centralizada, como, por exemplo, o PI-Bus [SIE 94].

As transferências de dados entre um mestre e um escravo são denominadas “transações”, sendo que existem dois tipos básicos. Na transação de escrita, o mestre envia dados ao escravo, enquanto que, na transação de leitura, o mestre solicita dados do escravo, o qual os envia ao mestre. Essas transações podem ser realizadas em um modo denominado “rajada” (*burst*), o qual permite que sejam transferidos vários dados de endereços adjacentes entre o mestre e o escravo em uma única transação e de forma atômica. Esse tipo de modo de operação é tipicamente utilizado em acessos a caches de instrução e de dado.

Segundo Guerrier e Greiner [GUE 2000a], além do fato de ser reutilizável, o barramento apresenta outras vantagens que justificam o seu uso como arquitetura de comunicação nos sistemas integrados atuais:

a) Suporta diretamente o modelo de comunicação de endereçamento em memória, sendo compatível com a maioria dos núcleos disponíveis;

b) A latência da comunicação é zero, uma vez que o mestre tenha sido autorizado pelo árbitro a utilizar o barramento;

c) Existe uma cultura bem estabelecida no uso do barramento em sistemas digitais; e

d) O custo de silício é pequeno.

3.2.2 A arquitetura de comunicação CoreConnect

A arquitetura de comunicação IBM CoreConnect consiste de uma estrutura de interconexão reutilizável organizada em uma hierarquia de barramentos e integra a biblioteca de núcleos denominada BlueLogic. Os núcleos dessa biblioteca são pré-projetados e pré-verificados para trabalharem com os protocolos da arquitetura CoreConnect, possibilitando o reuso de chip para chip [IBM 99].

A arquitetura CoreConnect provê três barramentos para a interconexão de núcleos: PLB (*Processor Local Bus*), OPB (*On-chip Peripheral Bus*) e DCR (*Device Control Register bus*), os quais são ilustrados na Figura 3.6 e descritos a seguir.

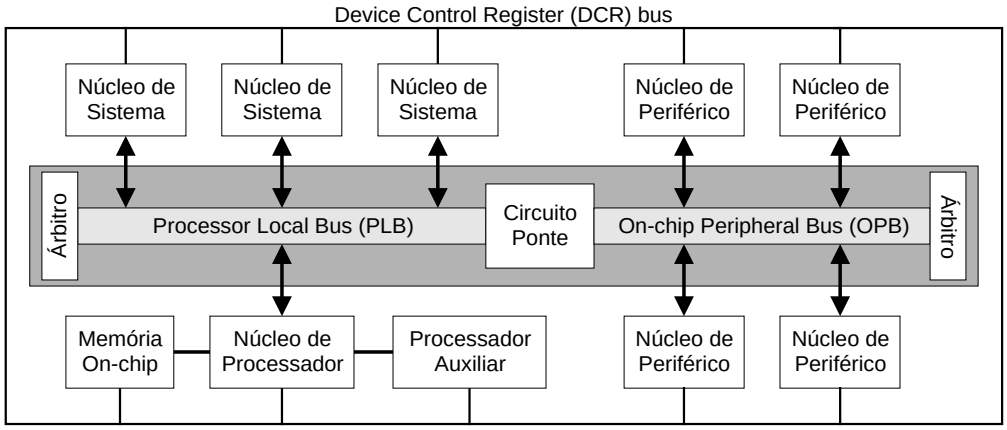


FIGURA 3.6 . Sistema genérico baseado na arquitetura CoreConnect.

PLB - *Processor Local Bus*

O PLB é dedicado à interconexão de dispositivos com grande largura de banda como núcleos de processadores, interfaces de memória externa e controladores de DMA. Os barramentos podem ser de 32, 64 e 128 bits (extensíveis em até 256 bits) e existem vias separadas para leitura e escrita de dados, o que possibilita a realização de duas transferências simultâneas (de leitura e de escrita) por ciclo de relógio. A latência da rede é reduzida pelo endereçamento em *pipeline*, o qual permite a sobreposição de três requisições de leitura com uma leitura em execução e uma requisição de escrita com um acesso de escrita em andamento. Além disso, possibilita a sobreposição de uma transferência em andamento com a arbitragem do barramento para futuras transferências. A largura de banda máxima estimada em até 23,2 Gbit/s para uma arquitetura de 128 bits operando a 183 MHz [IBM 2000b].

A interconexão de diversos núcleos mestres e escravos pelo PLB requer o uso de uma macrofunção chamada macro PLB (Figura 3.7), a qual é um núcleo reutilizável. Essa macrofunção oferece interfaces de controle e endereço, assim como portas separadas para leitura e escrita de dados para cada mestre do sistema. Ela suporta até oito mestres, mas implementações específicas podem suportar um número maior. Pelo lado dos escravos, a macro PLB oferece uma única interface compartilhada. Entretanto, independentemente do número de interfaces mestre, o PLB limita o número de transações simultâneas em cada ciclo de relógio a uma transação de escrita e outra de leitura.

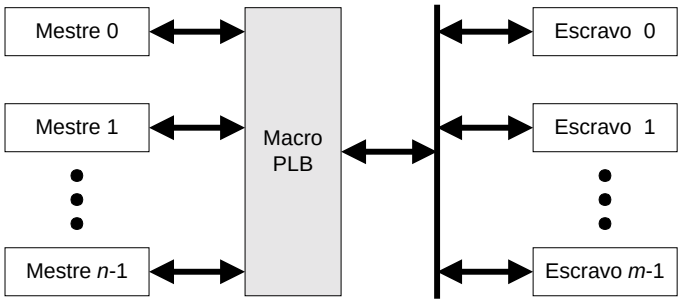


FIGURA 3.7 . Interfaces da macro PLB.

É importante destacar que o aumento do número de mestres e/ou escravos conectados à macro PLB afeta diretamente a freqüência de operação máxima alcançável. Para contornar essa limitação, a arquitetura CoreConnect prevê a implementação de múltiplos barramentos locais separados, interconectados por uma macro chamada PLB CBS (PLB *CrossBar Switch*). Na Figura 3.8, a macro PLB CBS é colocada entre os árbitros PLB e seus barramentos escravos. Quando um mestre inicia uma transação, a macro PLB CBS utiliza o endereço associado para selecionar o barramento escravo apropriado. A vantagem do uso dessa macro é que, por ser uma chave crossbar, permite a realização de múltiplas transferências de dados simultâneas utilizando um esquema de arbitragem para manipular requisições concorrentes para um mesmo barramento escravo. Essa solução resulta em uma arquitetura híbrida, pois inclui uma arquitetura de comunicação chaveada no sistema de comunicação baseado em barramentos.

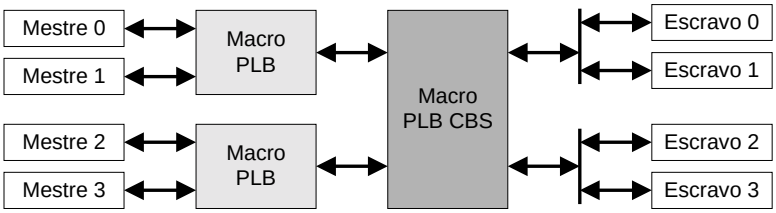


FIGURA 3.8 . Um sistema com múltiplas macros PLBs e uma macro PLB CBS [IBM 99].

OPB - *Peripheral On-chip Bus*

O OPB é um barramento secundário projetado para aliviar os gargalos de desempenho do sistema através da redução da carga capacitiva no PLB. Os tipos de periféricos que são passíveis de serem conectados no OPB incluem portas de comunicação serial e paralela, UARTs (*Universal Assynchronous Receiver/Transmitter*) e outros periféricos com baixos requisitos de largura de banda. Diferentemente do PLB, existe apenas um barramento de dados de 32 bits, o qual permite o dimensionamento dinâmico da largura do barramento (8, 16 e 32 bits), suporta transferências em rajadas e a conexão de múltiplos mestres ao barramento.

DCR – *Device Control Register bus*

O DCR é um barramento de baixo desempenho com estrutura em anel utilizado apenas para a transferência de informações de configuração e de estado entre o mestre desse barramento (um núcleo tipo processador) e os demais núcleos.

3.2.3 A arquitetura de comunicação AMBA

Um outro exemplo representativo de arquitetura de comunicação é a arquitetura AMBA (*Advanced Microcontoller Bus Architecture*), uma especificação padrão e aberta de barramento em chip desenvolvida pela ARM para a integração de núcleos em um SoC. Ela apresenta um barramento de sistema de alto desempenho (AHB – *Advanced High-perfomance Bus*) e um barramento de periféricos (APB – *Advanced Peripheral Bus*), os quais são interconectados por um circuito ponte [ARM 2003].

As arquiteturas AMBA e CoreConnect apresentam muitas características em comum. Os barramentos de alto desempenho das duas arquiteturas possuem vias separadas para leitura e escrita de dados, as quais podem ser de 32, 64 e de 128 bits, porém o PLB da arquitetura CoreConnect pode ser estendido a até 256 bits. Ambos os barramentos suportam operação com múltiplos mestres, transações divididas, transferências em *pipeline*, em rajadas e em linha. Já com relação aos barramentos de periféricos, o APB da arquitetura AMBA suporta apenas um mestre conectado ao barramento, enquanto que o OPB da arquitetura CoreConnect suporta a conexão de múltiplos mestres.

Contudo, a principal diferença entre as arquiteturas CoreConnect e AMBA está nas alternativas de implementação utilizadas para contornar as restrições ao desempenho do sistema de comunicação. Enquanto que a CoreConnect prevê a inclusão de um crossbar a fim de isolar os diferentes barramentos e possibilitar múltiplos acessos simultâneos, na arquitetura AMBA, os barramentos são implementados como multiplexadores sob a forma de um OU-por-fio (ver Figura 3.9). Isso isenta os núcleos de implementarem saídas em terceiro estado e transforma as conexões dos barramentos em ligações ponto-a-ponto, evitando o problema da degradação do desempenho decorrente de ligações multi-ponto, típico dos barramentos.

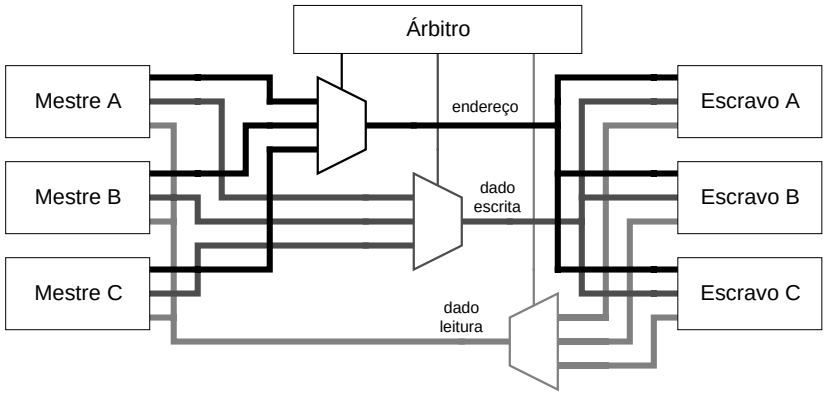


FIGURA 3.9 . O barramento de sistema da arquitetura de interconexão AMBA.

3.3 Arquiteturas de Comunicação para os Futuros SoCs

Os SoCs atuais podem integrar diversos tipos de núcleos com requisitos de comunicação na faixa de Gbit/s por núcleo, como processadores escalares, DSPs (*Digital Signal Processors*), processadores de vídeo, memória e dispositivos de E/S de alta largura de banda.

Um sistema integrado com uma dezena de núcleos operando em modo multi-mestre pode requerer uma largura de banda agregada maior que 10 Gbit/s e a expectativa é que os futuros SoCs apresentem requisitos de comunicação ainda maiores. Por exemplo, para explorar o paralelismo em nível de tarefa entre núcleos de processamento com uma comunicação concorrente reconfigurável, é necessária uma vazão agregada de 50 Gbit/s [GUE 2000a]. Aplicações com requisitos de comunicação como esse incluem os processadores de rede e sistemas para televisão de alta definição. Por exemplo, os processadores de rede OC-768 [KAR 2001] a serem utilizados nos *backbones* da Internet trabalharão com uma taxa de dados de 40 Gbit/s, requerendo uma capacidade de processamento só alcançável através de arquiteturas paralelas e um desempenho em comunicação maior do que o oferecido pelos barramentos atuais.

Como foi visto na seção anterior, o barramento tem sido a alternativa de arquitetura de comunicação preferencial nos SoCs atuais devido à sua reusabilidade. Contudo, como também foi destacado, o ele apresenta sérias limitações quanto ao consumo de energia e, principalmente, ao desempenho em comunicação. Sua freqüência de operação se degrada com o crescimento do sistema, ele não oferece paralelismo em comunicação e a sua largura de banda é compartilhada pelos núcleos conectados ao barramento.

Algumas das limitações do barramento podem ser amenizadas com a implementação de barramentos separados, interconectados por pontes. Entretanto, isso exige um agrupamento específico para cada caso, levando à perda da flexibilidade e da generalidade do barramento [GUE 2000a]. Além do mais, uma transação entre sub-unidades em barramentos diferentes ocupa diversos recursos do sistema de comunicação (todos os barramentos e pontes atravessadas), bloqueando outras comunicações. Somando-se a isso, essa alternativa exige um hardware adicional, o que implica no aumento do custo em silício do barramento.

Uma forma de se obter uma arquitetura de comunicação reutilizável e com desempenho escalável consiste no uso de redes de interconexão chaveada, semelhantes àquelas utilizadas em computadores paralelos. Essas redes baseiam-se em ligações ponto-a-ponto, e apresentam boas características quanto ao paralelismo, consumo de energia, freqüência de operação e escalabilidade, atendendo ainda ao requisito de reusabilidade.

Como o uso de tais redes na interconexão de núcleos em sistemas integrados é um tema emergente, não existe ainda uma terminologia única estabelecida nas comunidades acadêmica e industrial para referenciar esse novo tipo de arquitetura de comunicação intra-chip. Três termos diferentes têm sido utilizados na literatura: *micronetworks*, *On-Chip Networks* (OCNs) e *Networks-on-Chip* (NoCs), sendo que o último, por apresentar um acrônimo similar a SoC, tem tido a maior aceitação. Neste

texto, serão utilizados indistintamente os termos em rede-em-chip e o acrônimo NoC para referenciar essa nova abordagem.

As primeiras redes-em-chip utilizadas em SoCs consistiam basicamente de um crossbar central interligando os núcleos do sistema. Em geral, esses sistemas eram pequenos, com até uma dezena de núcleos, e a chave crossbar possuía um número de portas limitado. Um exemplo é o processador TMS320C80 MVP da Texas Instruments [KIM 97], um multiprocessador heterogêneo em um único chip combinando um processador RISC, quatro DSPs LIW, um controlador de DMA, dois controladores de vídeo e múltiplos módulos de memória interna, todos interconectados por um crossbar com uma largura de banda agregada de 19,2 Gbit/s.

Atualmente, existem diversos projetos em andamento que visam a construção de redes-em-chip. Essas NoCs são formadas por um conjunto de roteadores interligados por meio de canais ponto-a-ponto. Cada roteador possui um crossbar (centralizado ou distribuído entre os canais de saída), além de *buffers* de memória (ou filas) e circuitos de controle que implementam os mecanismos necessários à transferência de dados.

Dois projetos podem ser destacados por desenvolverem redes-em-chip com características diferentes: o SPIN e o aSoC. A rede SPIN utiliza topologia indireta e roteamento dinâmico, feito em tempo de execução, enquanto que a rede aSoC baseia-se em uma topologia direta e utiliza roteamento estático, realizado em tempo de compilação. Essas duas redes são descritas a seguir, de maneira sistemática, buscando caracterizá-las com base nas taxionomias apresentadas no capítulo anterior. Após, são dadas breves descrições sobre NoCs apresentadas mais recentemente na literatura.

3.3.1 Arquitetura de interconexão SPIN

A rede experimental SPIN (*Scalable Programmable Integrated Network*), em desenvolvimento no departamento ASIM (Architecture des Systèmes intégrés et Microélectronique) do Laboratório de Informática da Universidade Paris VI (LIP6 - Laboratoire d'Informatique Paris 6) é resultado de um projeto de pesquisa denominado Projeto SPIN[12]. Esse projeto focaliza os aspectos relacionados às arquiteturas de interconexão visando oferecer melhores escalabilidade, desempenho e facilidade de manufatura para os futuros sistemas integrados desenvolvidos com tecnologias de processo submicrônico. Ela foi inicialmente descrita em [GUE 99] e os primeiros resultados experimentais foram apresentados em [GUE 2000] e [GUE 2000a]. Um estudo comparativo entre o desempenho em comunicação da rede SPIN e do barramento PI-Bus foi desenvolvido no escopo deste trabalho [AND 2003].

Topologia

A arquitetura SPIN é uma rede indireta com topologia em árvore-gorda quaternária, com roteadores nos nodos e núcleos (ou terminais) nas folhas [GUE 2000a], conforme é ilustrado na Figura 3.10. Essa topologia pode também ser classificada como uma rede multiestágio bidirecional [DUA 97]. Os enlaces que interligam os elementos da árvore (nodos e folhas) são bidirecionais *full-duplex*,

[12] Página do Projeto SPIN disponível por WWW em http://asim.lip6.fr/~adrijean (acesso em 20 de julho de 2003).

implementados por dois canais unidirecionais *simplex*. Cada canal *simplex* é constituído por 32 bits para dados e quatro bits de banda lateral (do inglês, *sideband*) para enquadramento do pacote (bits marcadores de início e fim), sinalização de paridade da palavra de dado e sinalização de erro de paridade. Ou seja, o tamanho do phit (largura física do canal) é igual a 36 bits, porém, além desses, cada canal *simplex* inclui um par de sinais necessários ao controle de fluxo, os quais não são contabilizados no cálculo do phit do canal.

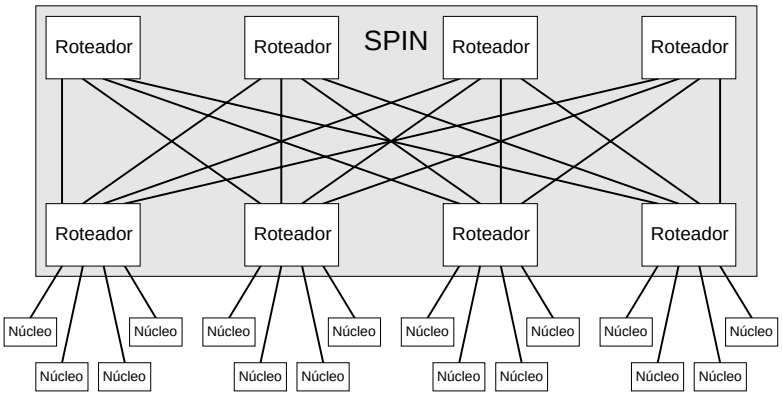


FIGURA 3.10 . SoC com 16 núcleos interconectados por uma rede SPIN.

Roteamento

O mecanismo de roteamento utilizado na rede SPIN varia com o sentido de deslocamento de um pacote pela rede. Quando um cabeçalho de pacote chega em um roteador, é feita a comparação de um conjunto de bits do endereço do destinatário contidos no cabeçalho com a própria identificação do roteador. Em função do resultado dessa comparação, o roteador decide se o pacote deve “subir” na rede, afastando-se das folhas da árvore, ou “descer”, aproximando-se do seu destinatário. Para os pacotes que sobem, o mecanismo de roteamento é adaptativo e qualquer canal de saída em direção ao nível superior da rede pode ser utilizado, conforme a disponibilidade dos mesmos e em função de um mecanismo de escolha aleatória. Para pacotes que descem, o roteamento é determinístico e o canal de saída a ser utilizado em direção ao nível inferior está identificado no endereço do destinatário, sendo, portanto, pré-determinado pelo emissor do pacote. Com relação ao local onde as decisões são tomadas, o roteamento é distribuído para os pacotes que sobem e fonte para os pacotes que descem na rede, pois, no primeiro caso são os roteadores que escolhem os canais de saída a serem utilizados, enquanto que, no segundo caso, são as interfaces de rede dos nodos fontes que tomam essa decisão. Em ambos os casos, os mecanismos de roteamento são dinâmicos, pois são realizados em tempo de execução. Observando-se a Figura 3.10, pode-se concluir que as comunicações entre núcleos conectados a um mesmo roteador jamais utilizarão o roteamento adaptativo, enquanto que as comunicações entre núcleos conectados a diferentes roteadores deverão “subir” pela árvore de maneira adaptativa.

Um problema do roteamento adaptativo aplicado aos pacotes que sobem pela rede é a possibilidade de perda do ordenamento original dos pacotes enviados, pois cada pacote poderá tomar um caminho diferente e chegar no destinatário em uma ordem diversa daquela na qual foram emitidos, pois a latência pode variar de um caminho para outro. Esse problema deve ser resolvido pelo protocolo de comunicação, conforme será discutido no final desta subseção.

Chaveamento

A rede SPIN utiliza uma variação do chaveamento *wormhole*, a qual é semelhante à técnica denominada *buffered wormhole switching* utilizada na rede de interconexão do multiprocessador IBM SP [STU 94]. Essa técnica combina aspectos dos chaveamentos por pacote armazena-e-repassa (ou SAF) e *wormhole*, sendo que seu nome é justificado pelo fato de que cada roteador possui um *buffer* central para absorver pequenos pacotes bloqueados na rede, mas o controle de fluxo é feito em nível de flit. Dessa forma, dependendo das condições de tráfego e do tamanho dos pacotes, ele pode se comportar de forma análoga ao *wormhole* ou ao chaveamento por pacotes. Na rede SPIN, são implementados dois *buffers* centrais um para cada lado do roteador (superior e inferior) e quando um pacote é bloqueado em um canal de entrada, seus flits são movidos para o *buffer* central correspondente, conforme a disponibilidade de espaço para absorvê-los. A primeira vantagem dos *buffers* centrais é que seu uso ameniza o problema do bloqueio de cabeça de linha[13], pois quando um pacote bloqueado é completamente movido para um *buffer* central, a saída do *buffer* de entrada em que ele estava armazenado fica liberada para o pacote seguinte na fila. Uma segunda vantagem é que, no esquema de arbitragem, descrito logo a seguir, a prioridade maior é dada aos pacotes armazenados nos *buffers* centrais. Isso permite reduzir a latência de um pacote na rede e também estabelecer um certo grau de qualidade de serviço, pois os pacotes são classificados em dois tipos: os que podem ser movidos para os *buffers* centrais e os que não podem. Os pacotes do primeiro tipo terão um atendimento mais privilegiado que os do segundo e serão submetidos uma latência média menor para atingir seus destinatários.

Desde que o chaveamento da rede SPIN é um variante do chaveamento *wormhole*, os pacotes são divididos em flits[14] (unidades de controle de fluxo), sendo que cada flit possui o tamanho de um phit (largura do canal), ou seja, 36 bits.

Formato do pacote

O primeiro flit de um pacote carrega o cabeçalho, no qual oito dos 32 bits da parte de dado são utilizados para identificar um entre 256 destinatários possíveis. Dos 24 bits restantes da parte de dado do cabeçalho, um é usado identificar se o pacote está habilitado ou não a utilizar os *buffers* centrais e os demais são reservados ao protocolo de nível mais alto, como, por exemplo, para enumerar os pacotes emitidos de modo que o receptor possa reordená-los em caso de recebimento fora da ordem de emissão. A carga útil do pacote pode ter qualquer tamanho e o terminador é um flit no qual o bit marcador de fim de pacote (*eop*) da banda lateral é estabelecido em um. É previsto que um pacote típico tenha um tamanho médio correspondente ao número de flits necessários para a transferência de uma linha de cache com quatro palavras de 32 bits, ou seja, um flit de cabeçalho e quatro pares de flits para endereço e dado, totalizando nove flits.

[13] Ver definição nas subseções 2.6.2 (Controle de fluxo baseado em canais virtuais) e 2.7.1 (Memorização centralizada compartilhada).
[14] Ver definição na subseção 2.5.4 (Chaveamento por pacote *wormhole*).

Roteador

O bloco construtivo básico da rede SPIN é o roteador RSPIN mostrado na Figura 3.11. Ele possui quatro portas bidirecionais dirigidas ao nível inferior da rede e quatro portas bidirecionais dirigidas ao nível superior da rede, as quais são denominadas, respectivamente, portas inferiores e portas superiores. Nos canais de entrada de cada porta, há um *buffer* tipo FIFO com capacidade para armazenar quatro flits. Além desses, existem dois *buffers* (ou filas) centrais compartilhados com capacidade para armazenar 18 flits de pacotes bloqueados nos *buffers* dos canais de entrada, minimizando o problema do bloqueio de cabeça de linha. Um *buffer* central é dedicado às portas inferiores e o outro às portas superiores, sendo que a capacidade de cada *buffer* central permite o armazenamento de dois pacotes do tamanho de uma linha de cache de 32 bits. As saídas dos *buffers*, os canais de saída e as entradas dos *buffers* centrais são interconectados por uma chave crossbar 10∝10 parcial, na qual nem todas as conexões são implementadas devido a restrições impostas pelo algoritmo de roteamento. Essa chave crossbar é construída de forma distribuída por meio de multiplexadores baseados em *buffers tri-state.* Cada multiplexador é alocado a um dos recursos compartilhados do roteador, ou seja, aos canais de saída e às filas centrais.

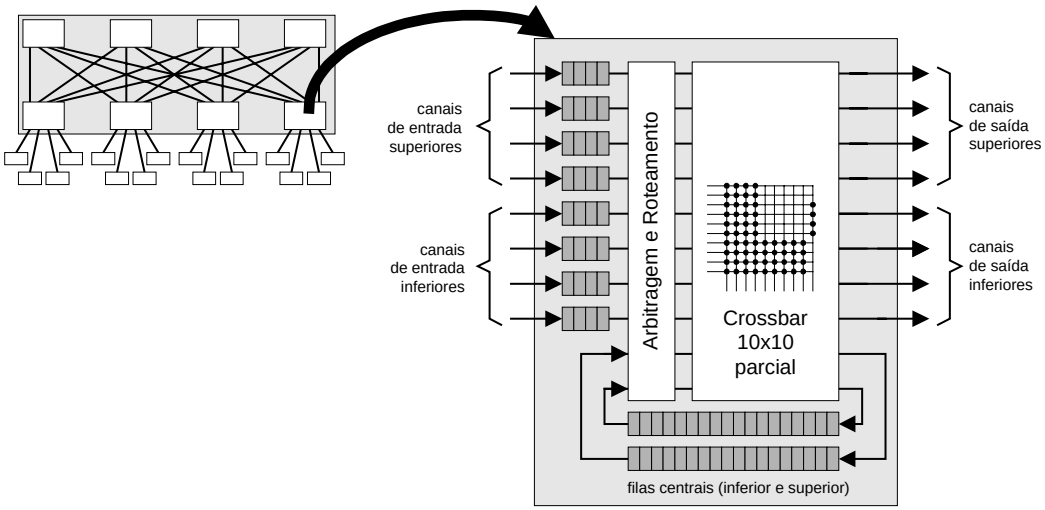


FIGURA 3.11 . Roteador RSPIN [GUE 2000a].

Quanto ao bloco de arbitragem e de roteamento, ele também é implementado de maneira distribuída. Cada canal de entrada possui um bloco de roteamento que detecta a presença de um cabeçalho de pacote na saída do *buffer* de entrada, aplica o algoritmo de roteamento e requisita o canal de saída necessário para o encaminhamento do pacote. Se o canal de saída estiver ocupado, o bloco de roteamento então requisita o *buffer* central associado ao tipo de canal do *buffer* de entrada (desde que o pacote esteja habilitado a ser encaminhado para um *buffer* central). As filas centrais também possuem seus blocos de roteamento para a seleção do canal de saída. Já os blocos de arbitragem são associados a cada recurso compartilhado do roteador, ou seja, canais de saída e entradas dos *buffers* centrais. Eles aplicam um algoritmo de arbitragem baseado em prioridades dinâmicas rotativas de modo a oferecer uma utilização balanceada desses recursos. Contudo, os *buffers* centrais possuem prioridade maior em relação aos canais de entrada.

No que tange a realização física, o roteador RSPIN é implementado como uma macrocélula de hardware (um *hard-core*) e possui 9144 portas lógicas, ocupando uma área de 4400µ ∝ 4560µ. Em um processo .25ν CMOS com seis camadas de metal, a área é de 0,8 mm^2, sendo reduzida para 0,45 mm^2 em um processo .18ν CMOS. Da área total, 29,6% é ocupada pela parte de memorização (*buffers* de entrada e centrais).

Estudos preliminares sugerem que o custo em área para uma rede SPIN com 32 terminais é menor que 13 mm^2 em um processo .25ν CMOS. Estima-se que, em uma área de silício com menos de 20 mm^2, seja possível implementar uma rede com 64 terminais em um processo .18ν CMOS ou com 128 terminais em um processo .13ν CMOS [GUE 2002b]. Nessas estimativas, é assumido o uso do roteador RSPIN com canais de dados de 32 bits e com as mesmas profundidades de *buffer*.

Memorização

Em função da técnica de chaveamento *buffered wormhole*, a rede SPIN utiliza memorização na entrada e memorização centralizada compartilhada. Cada roteador RSPIN inclui 144 bytes nos oito *buffers* de entrada com capacidade de 4 flits (8 ∝ 4 ∝ 36 bits = 1152 bits) e 162 bytes nos dois *buffers* centrais com capacidade de 18 flits (2 ∝ 18 ∝ 36 bits = 1296 bits), totalizando 306 bytes de memorização. Nesses dados, não estão contabilizados os registradores necessários ao armazenamento dos estados dos autômatos do roteador.

Desempenho

O ciclo de relógio obtido por simulação elétrica para uma rede SPIN com 32 terminais em um processo .25ν CMOS é de 5 ns (200 MHz). Para essa rede, a largura de banda agregada máxima estimada será de 204.8 Gbit/s (32 ∝ 200 MHz ∝ 32 bits). Apenas para comparação, a largura de banda máxima estimada para o barramento de sistema (PLB) da arquitetura CoreConnect é de 23,2 Gbit/s para uma arquitetura de 128 bits [IBM 99]. Já em um rede SPIN com 128 terminais, a largura de banda agregada máxima estimada é de 1,82 Tbit/s, considerando um processo .13ν CMOS [GUE 2000a].

Controle de fluxo

A rede SPIN utiliza controle de fluxo baseado em créditos. Um crédito inicial é dado ao transmissor e esse crédito é decrementado a cada flit enviado. Quando o receptor consome esse flit, ele notifica o transmissor, enviando-lhe um crédito através de uma das linhas de controle de fluxo. Quando o sistema inicia seu funcionamento, os créditos dos emissores são estabelecidos no valor correspondente à capacidade dos *buffers* de entrada dos receptores (quatro posições).

Reusabilidade e interconectividade

O protocolo de comunicação utilizado na rede SPIN é baseado no modelo de troca de mensagens, o qual difere dos protocolos usados na maioria dos núcleos disponíveis para a construção de SoCs. Esses núcleos utilizam ou o modelo de espaço

de endereçamento ou o modelo de fluxo de dados (eg. processamento de vídeo), sendo que este último e algumas versões do primeiro não admitem o recebimento de pacotes em uma ordem diferente daquela enviada [GUE 2000a], o que pode ocorrer na rede SPIN devido ao uso de roteamento adaptativo.

Para oferecer a capacidade de reordenamento de pacotes, é necessário que, primeiramente, o emissor enumere os pacotes emitidos, utilizando os bits de protocolo de alto nível disponíveis no cabeçalho do pacote SPIN. Após, é preciso que o receptor implemente um *buffer* para armazenar esses pacotes e entregá-los na ordem correta. Esse protocolo deve ser implementado pelo adaptador que conecta o núcleo à rede.

A identificação dos pacotes se faz necessária também em SoCs que utilizem processadores *multi-threaded*. Se uma *thread* executando em um processador envia um pacote de requisição para a leitura de dados em uma memória remota por exemplo, uma nova *thread* poderá ser escalonada enquanto a primeira aguarda pelo recebimento do pacote de resposta. Quando esse pacote chegar, é preciso identificar a qual *thread* ele se refere a fim garantir o funcionamento correto do sistema. Embora, nesse caso, o problema não decorra do roteamento adaptativo, é preciso assegurar que o protocolo de comunicação permita estabelecer a identificação dos pacotes em função das *threads* às quais eles estão associados.

Para garantir a reusabilidade e a interconectividade com núcleos de diferentes fabricantes, a rede SPIN prevê o uso de interfaces de rede (também denominadas adaptadores ou *wrappers*) para realizar a tradução entre o protocolo nativo SPIN e o padrão VCI (*Virtual Component Interface*). O adaptador SPIN/VCI é colocado entre o núcleo e a rede, sendo que existem dois tipos de adaptadores: um para componentes que operam como mestres da comunicação ou iniciadores de requisições (eg. processadores) e outro para componentes escravos ou alvos de requisições (eg. memória). O núcleo conectado ao adaptador SPIN/VCI deve possui uma interface VCI ou, se sua interface for proprietária, utilizar um segundo adaptador para fazer a tradução entre o seu protocolo e o protocolo VCI. Os adaptadores SPIN/VCI utilizam transações divididas (protocolo *split*) para permitir que um mestre envie uma série de pacotes de requisição a um ou vários escravos mesmo antes do recebimento do primeiro pacote de resposta. Se esse recurso não fosse usado, a latência da rede iria impor restrições significativas à comunicação, pois o mestre teria que aguardar o retorno do pacote de resposta para enviar um novo pacote de requisição. Como os pacotes de resposta podem ser recebidos em uma ordem diferente daquela na qual os pacotes de requisição foram emitidos, um *buffer* é utilizado para garantir a entrega das respostas ao núcleo na mesma ordem das requisições por ele enviadas.

Como o mecanismo de controle de fluxo baseado em créditos garante que os flits de pacotes só são emitidos se o receptor tiver capacidade de absorvê-los, a possibilidade de perda de pacotes é mínima, salvo falhas na rede. Mecanismos de retransmissão podem ser implementados nos adaptadores seja para reenviar um pacote para o qual foi detectado um erro de paridade na transmissão ou para retransmitir um pacote perdido devido a uma falha na rede.

Limitações da arquitetura SPIN

Um problema muito importante em sistemas multiprocessados com compartilhamento de dados em memória é o da coerência de cache. Enquanto o barramento oferece suporte em hardware para garantir a coerência das cópias dos dados compartilhados em memória, a rede SPIN não oferece nenhum mecanismo específico para esse fim. A ausência desse suporte é justificada pela complexidade e pelo custo de sua implementação em sistemas embarcados e pelo fato de que esses sistemas são multiprocessadores preferencialmente heterogêneos e assimétricos, com primitivas de sincronização simples e invalidação explícita, as quais são facilmente suportadas por protocolos mais simples.

Uma segunda limitação diz respeito ao fato de não se ter garantido a ausência de *deadlock* na rede SPIN. Uma solução aplicada foi reservar metade dos canais de saída das portas superiores do roteador a pacotes que transportam uma requisição (eg. requisição de leitura) e a outra metade a pacotes que transportam uma resposta (eg. dados lidos). Essa solução confirmou ser efetiva por simulação, mas até o momento da publicação desta tese, ainda não se havia provado formalmente se ela garante a liberdade de *deadlock*.

3.3.2 A arquitetura de interconexão aSoC

A arquitetura aSoC (*adaptive System-on-Chip*)[15] é uma rede de interconexão chaveada experimental com largura de banda escalável dirigida a sistemas orientados ao fluxo de dados. Essa rede difere da arquitetura SPIN por utilizar roteamento estático definido em tempo de compilação [LIA 2000].

Topologia

A topologia da rede de interconexão aSoC é do tipo direta, sendo que cada nodo da rede é constituído pelo núcleo e seu roteador. O roteador possui quatro portas bidirecionais para conexão com nodos vizinhos e uma interface bidirecional para conexão com o núcleo local, sendo que o canal possui uma largura de 32 bits (tamanho do phit).

A estrutura da rede é determinada pelo projetista em função dos requisitos do sistema, entretanto ela é limitada pelo grau do nodo, o qual permite que cada nodo seja conectado a no máximo outros quatro nodos. Com isso, pode-se implementar topologias como a grelha 2-D ou uma topologia irregular qualquer. Na Figura 3.12, é apresentado um sistema integrado aSoC genérico utilizando uma rede de interconexão com topologia em grelha 2-D.

[15] Página do Projeto aSoC disponível por WWW em http://vsp2.ecs.umass.edu/vspg/ASOC/index.html (acesso em 20 de julho de 2003).

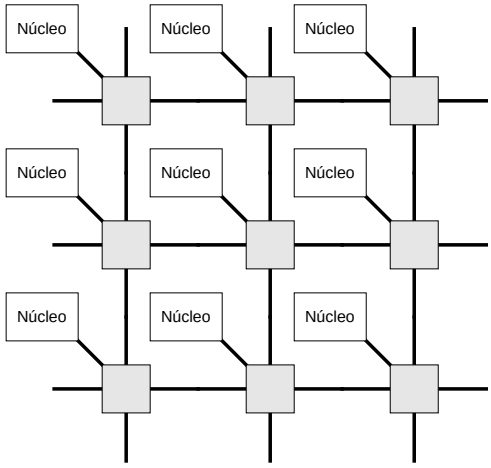
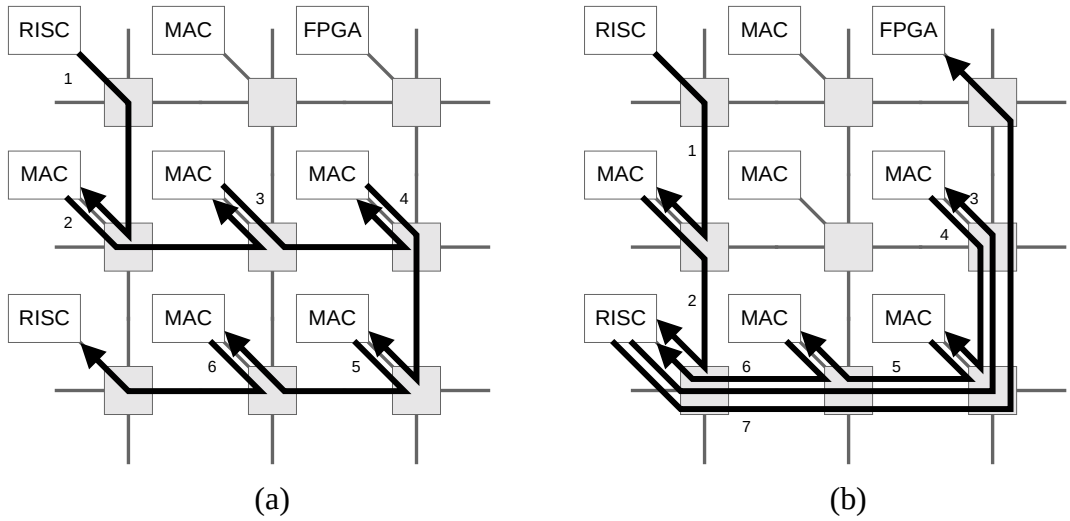


FIGURA 3.12 . Sistema com 9 núcleos interconectados por uma rede em grelha 2-D.

A arquitetura aSoC foi projetada para uso em sistemas reconfiguráveis heterogêneos, como os exemplos ilustrados no sistema da Figura 2.13. No sistema mostrado, existem três tipos de núcleos: FPGA, processador RISC (*Reduced Instruction-Set Computer*) e multiplicador-acumulador (MAC). Na Figura 3.13.a, são mostradas as comunicações necessárias para a execução de uma aplicação de filtro IIR (*Infinite Impulse Response*) de cinco estágios. Primeiramente, um núcleo RISC realiza um pré-processamento e, após, cinco MACs executam os estágios de multiplicação e acumulação. O resultado final é colhido pelo segundo processador RISC, sendo que nem todos os núcleos disponíveis no sistema são utilizados por essa aplicação. No entanto, eles podem ser aproveitados em outras aplicações. O FPGA, por exemplo, pode ser utilizado para implementações específicas à aplicação, como o decodificador Viterbi da Figura 3.13.b. Nas figuras, cada passo de comunicação está identificado por um número de seqüência. Observa-se que o modelo de comunicação utilizado por essas duas aplicações é o de fluxo de dados e que o padrão de comunicação de cada aplicação pode ser completamente determinado em tempo de compilação.



(a) (b)

FIGURA 3.13 . Padrões de comunicação da execução de duas aplicações em um SoC heterogêneo com nove núcleos: (a) filtro IIR; (b) decodificador Viterbi.

Roteamento

O mecanismo de roteamento usado na rede aSoC pode ser classificado como estático, centralizado e determinístico. A configuração de cada roteador, para cada ciclo de execução da aplicação, é determinada pelo escalonador durante a compilação e carregada em uma memória interna do roteador, a qual é denominada memória de interconexão. Um contador de programa indica a posição de memória a ser utilizada para configurar o crossbar em cada ciclo.

Esse escalonamento da comunicação em tempo de compilação é feito com base na premissa de que a maior parte da comunicação em aplicações intensivas em dados pode ser determinada em tempo de compilação. Isso permite uma redução do hardware e um aumento do desempenho da comunicação ao custo da perda de flexibilidade. O escalonador maximiza a quantidade de mensagens trocadas simultaneamente, colocando, em um mesmo ciclo, comunicações que não requerem o uso dos mesmos recursos de comunicação (eg. canais). Essa abordagem tem sido utilizada em diversos sistemas de processamento paralelo, como o iWarp [BOR 90], o NuMesh [SHU 96] e o RAW (*Reconfigurable Architecture Workstation*) [WAI 97].

Chaveamento

A rede aSoC utiliza uma variação do chaveamento por circuito. A cada ciclo de relógio, são estabelecidos caminhos diretos entre pares fonte-destinatário, sendo que cada caminho pode ocupar um ou mais canais da rede. O algoritmo de escalonamento analisa o perfil de comunicação da aplicação e determina os circuitos que podem ser estabelecidos simultaneamente, maximizando o paralelismo nas comunicações. A diferença principal com o chaveamento por circuito clássico é que o uso de roteamento estático permite que os circuitos sejam determinados antes da execução da aplicação. Isso evita a necessidade de uma fase inicial de envio de um cabeçalho para o estabelecimento do caminho fonte-destinatário.

Roteador

O roteador da arquitetura aSoC, mostrado na Figura 3.14, consiste de um crossbar 5∝6 no qual uma das interfaces é usada para conexão com o núcleo local e as outras quatro para conexão com os nodos vizinhos. Uma sexta saída do crossbar é utilizada internamente para configurar um contador usado em transferências de mensagens de tamanho arbitrário. O roteador inclui ainda uma memória de interconexão, um contador de programa, *buffers* de entrada e *buffers* FIFO para comunicação com o núcleo local. A memória de interconexão armazena diferentes padrões de configuração determinados pelo compilador, enquanto que o contador de programa seleciona a configuração a ser utilizada em cada ciclo, selecionando uma posição da memória de interconexão.

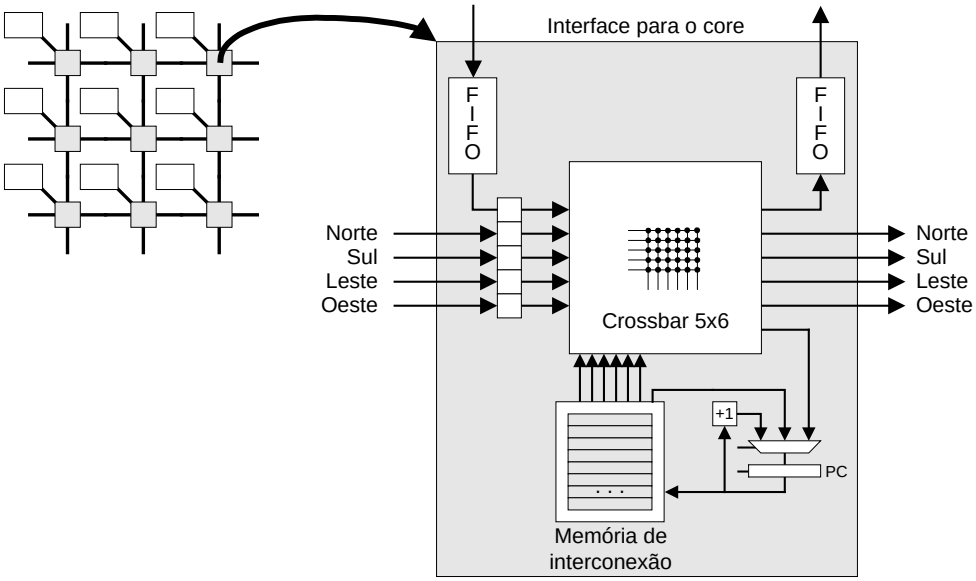


FIGURA 3.14 . O roteador aSoC.

O roteador aSoC possui 50.000 transistores ocupando uma área de 6000μ ∝ 5000μ, sendo que o crossbar ocupa uma área de 4000μ ∝ 2800μ e possui 2.300 transistores. Nenhuma informação adicional sobre a realização física do roteador foi encontrada na literatura disponível.

Memorização

A rede aSoC utiliza um esquema de memorização particular. Em função de seus mecanismos de roteamento e chaveamento, seus roteadores incluem uma memória de interconexão de tamanho variável, conforme as aplicações-alvo do sistema. Por exemplo, a implementação realizada em [LIA 2000] utilizou uma memória de 256∝32 bits (1 Kbyte). A memorização de entrada é mínima (5∝4 bytes = 20 bytes, para uma palavra de 32 bits), enquanto que os *buffers* FIFO da interface com o núcleo local possuem uma profundidade de quatro palavras (2 ∝ 4 ∝ 4 bytes = 16 bytes) e o contador de programa ocupa 16 bits (2 bytes). Com isso, um roteador aSoC com uma memória de 256 posições terá um total de 1062 bytes (1024 + 20 + 16 + 2).

Desempenho

Estimativas indicam que em uma implementação usando um processo .3 νm CMOS, o período mínimo de operação do relógio será de 3 ns (333 MHz)[16]. Uma rede aSoC com 16 terminais usando uma topologia em grelha 2-D terá uma largura de bisecção igual a 8, considerando dois canais em cada enlace. Com a tecnologia citada acima, essa rede terá uma largura de banda agregada máxima estimada de aproximadamente 85,2 Gbit/s (8 ∝ 333 MHz ∝ 32 bits).

[16] Esse valor refere-se a uma estimativa conservadora, a qual foi feita sobre um valor escalado dos resultados de simulação obtidos para um roteador projetado com tecnologia de 1.2 νm CMOS (caminho crítico igual a 8,5 ns obtido por simulação via SPICE [LIA 2000]).

Uma rede SPIN equivalente, com 16 terminais, usando um processo .25νm CMOS, terá uma largura de banda máxima estimada de 102,4 Gbit/s ($16 \propto 200$ MHz $\propto 32$ bits), considerando o mesmo ciclo de relógio estimado para a rede com 32 terminais. Nesse caso, mesmo que a rede aSoC opere em uma freqüência mais alta, a rede SPIN possuirá uma largura de banda agregada maior em função da largura de bisecção da sua topologia em árvore gorda ser maior que a da grelha.

Controle de fluxo

Na interface entre o roteador e o núcleo local, os *buffers* FIFO provêem um meio de compatibilizar a diferença de velocidade entre o núcleo e o roteador, sendo, neste caso, utilizado um controle de fluxo do tipo FIFO. O núcleo só pode escrever no *buffer* de entrada do roteador se o mesmo não estiver cheio e só pode ler do *buffer* de saída se o mesmo não estiver vazio. O estado de cada *buffer* é indicado por um bit de controle.

Interconectividade e reusabilidade

O roteamento estático e determinístico utilizado na rede aSoC a torna bastante adequada para uso em aplicações orientadas ao fluxo de dados, como, por exemplo, aplicações em processamento digital de sinais, com a vantagem de oferecer uma estrutura adaptável ao padrão de comunicação de cada aplicação.

As informações disponíveis na literatura a respeito da rede aSoC dão a entender que a interface entre o núcleo e o roteador é constituída apenas pelos FIFOs de entrada e de saída, utilizando o mecanismo de controle de fluxo tipo FIFO. Ou seja, para conectar qualquer núcleo ao roteador aSoC, é preciso adicionar um adaptador para esse tipo de interface. É possível que essa limitação seja decorrente do aparente uso restrito da rede ao escopo do projeto. Enquanto a rede SPIN já inclui adaptadores de protocolo para o padrão VCI, a literatura sobre a rede aSoC não menciona a disponibilidade de adaptadores para interfaces padrão (eg. VCI ou OCP).

Limitações

Em [LIA 2000], são apresentados resultados de simulação de sistemas com 9 e 16 núcleos executando três aplicações de processamento digital de sinais. Utilizando o simulador NSIM [MET 92], foram modeladas versões desses sistemas usando três tipos de arquitetura de comunicação: barramento, rede aSoC e rede chaveada com roteamento dinâmico e chaveamento por pacotes (os autores não oferecem detalhes sobre a arquitetura dessa rede). Os resultados finais mostram que ambas as redes-em-chip proporcionam uma redução no tempo de execução das aplicações quando comparadas com o barramento. A rede aSoC mostrou-se mais eficiente, com menores tempos de execução e taxas de utilização de enlace. Contudo, os resultados obtidos com a rede chaveada com roteamento dinâmico são bastante próximos aos da aSoC. Embora os resultados do trabalho supracitado mostrem que a rede aSoC com roteamento estático oferece um desempenho melhor que o de uma rede com roteamento dinâmico para as aplicações simuladas, é justamente a falta de roteamento em tempo de execução que é identificada pelos autores como uma limitação significativa da rede aSoC. Considerando a necessidade desse tipo de roteamento, eles propõem, como abordagem alternativa, realizar alguns ciclos do escalonamento em tempo de execução. Nessa abordagem, os cabeçalhos dos pacotes são examinados pelo roteador e a informação

resultante do roteamento é armazenada em uma pequena memória. Transferências de dados subseqüentes podem então utilizar essa informação armazenada para encaminhar os dados em direção ao destinatário correto.

Outra limitação comentada pelos autores diz respeito ao tamanho da memória de interconexão, a qual deve ser grande o suficiente para acomodar um espectro largo de aplicações, o que pode encarecer o roteador. Finalmente, assim como na rede SPIN, nenhum mecanismo em hardware é oferecido para suportar a coerência de cache, o que provavelmente deve ser justificado pelo tipo de sistemas e aplicações visadas pelo projeto aSoC.

3.3.3 Outras redes-em-chip

As redes-em-chip descritas acima constituem dois exemplos de alternativas ao barramento que utilizam características arquiteturais diferentes. Enquanto a SPIN adota roteamento dinâmico e um tipo de chaveamento por pacotes, a rede aSoC utiliza roteamento estático e uma variante do chaveamento por circuito. A primeira abordagem é bastante questionada por muitos pesquisadores face à latência de comunicação decorrente do roteamento dinâmico. Entretanto, ela é a solução mais flexível e genérica, permitindo seu uso em sistemas baseados em diferentes modelos de computação e de comunicação, sobretudo em aplicações com tráfego dinâmico. Tal afirmativa pode ser confirmada pelo aumento recente de trabalhos baseados nessa abordagem. Nesta subseção, são apresentadas algumas arquiteturas de redes-em-chip mais recentes de modo a ilustrar opções de projeto mais utilizadas atualmente.

O toróide dobrado de Dally e Towles

Em [DAL 2001], Dally e Towles propõem o uso das redes-em-chip baseadas em roteamento dinâmico e chaveamento por pacotes do tipo *wormhole* como alternativa de interconexão em sistemas integrados. Segundo os autores, o uso dessas redes no lugar de conexões multi-ponto *ad-hoc* permite a aplicação de circuitos de sinalização agressivos de modo a reduzir a dissipação de potência e aumentar a velocidade de propagação, reduzindo a latência e aumentando a largura de banda. Em resumo, a idéia é estruturar um SoC como uma matriz de blocos de tamanho regular que se comunicam pela troca de pacotes através de uma rede de interconexão chaveada com topologia direta do tipo toróide dobrado (uma variação do toróide 2-D). Os autores apresentam um exemplo de rede que apresenta um custo adicional de silício estimado em 6,6 %. Essa rede possui muitas características semelhantes às da rede SPIN, contudo, algumas diferenças devem ser destacadas:

a) O SoC é organizado como uma matriz de blocos quadrados, denominados *tiles*. Cada núcleo é mapeado em um desses blocos e os roteadores são então implementados de forma distribuída ao redor dos blocos.

b) O canal de comunicação é extremamente largo, com um campo de dado de 256 bits mais 38 bits para sinais de controle e identificação da rota, entre outros (phit = 294 bits). Com isso, em um único flit pode ser transferida uma linha de cache completa ou mesmo todos os sinais lógicos de uma interface de um núcleo qualquer com até 256 sinais de entrada-e-saída;

c) Para cada canal físico, existem oito canais virtuais que possibilitam o provimento de maior qualidade de serviço para alguns tipos de pacotes, como, por exemplo, pacotes pré-escalonados referentes a aplicações que apresentam um perfil de comunicação quase estático, com grandes fluxos de dados e que exigem alta largura de banda e latência reduzida. Além disso, os canais virtuais minimizam o problema do bloqueio de cabeça de linha e permitem que se evite o congelamento da rede por *deadlock*.

d) Cada roteador requer aproximadamente 6000 bytes de memorização (cerca de 20 vezes mais que no RSPIN). Tal requisito deve-se ao tamanho do phit e ao número de canais virtuais.

Na Figura 3.15, é ilustrada a organização da rede proposta, mostrando-se sua topologia, as interfaces do roteador distribuído (denominadas *N*, *E*, *S*, *W* e *L*) e a organização interna da interface *E* (leste).

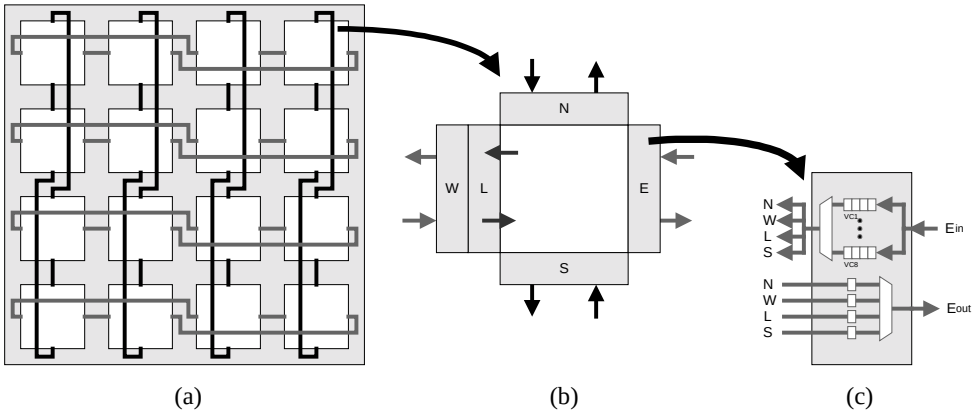
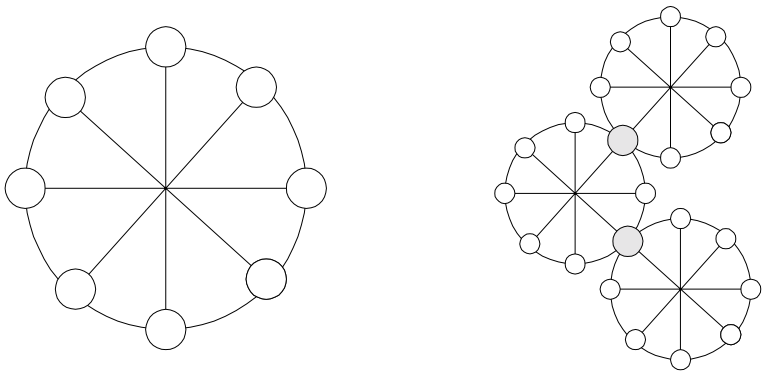


FIGURA 3.15 . Rede proposta em [DAL 2001]: (a) topologia; (b) roteador distribuído; (c) interface leste do roteador.

A rede Octagon

Em [KAR 2002], Karim, Nguyem e Dey descrevem uma rede de interconexão chaveada para um processador de rede OC-768 destinado a operar como roteador de *backbone* da Internet. Com base, nos requisitos de processamento e de comunicação intra-chip da aplicação, os autores afirmam que um roteador OC-768 requer uma arquitetura multiprocessada com memória distribuída capaz de executar mais de 57 GIPS e uma arquitetura de comunicação que suporte uma taxa de dados da ordem de 40 Gbit/s [KAR 2001]. A arquitetura ideal para tal aplicação seria o crossbar, mas tal arquitetura possui uma escalabilidade cara – $O(n^2)$. Dessa forma, os autores propõem uma rede de interconexão chaveada cujo custo cresce linearmente com o número de nodos. A configuração básica dessa rede é um agregado com oito nodos, denominado Octagon, e utiliza uma topologia direta do tipo anel-cordal [HWA 93]. Cada nodo é constituído por um roteador 4∞4 e um processador e/ou memória, conforme é ilustrado na Figura 3.16. A principal característica dessa topologia é que a distância máxima entre dois nodos é igual a dois. Ou seja, para nodos conectados a um mesmo agregado, esse é o número máximo de enlaces entre roteadores a serem percorridos por um pacote qualquer.



(a) (b)

FIGURA 3.16 . Rede Octagon [KAR 2002]: (a) topologia básica com 8 nodos; (b) escalabilidade.

A rede Octagon utiliza roteamento dinâmico e distribuído, e pode operar com chaveamento por circuito ou por pacotes. No chaveamento por circuito, um mecanismo é utilizado para escalonar comunicações que não ofereçam conflitos de recursos e que possam ser estabelecidas simultaneamente. Segundo uma análise de desempenho, para sistemas equivalentes, a vazão oferecida pela Octagon é oito vezes maior que a do barramento e duas vezes maior que a do crossbar. Além disso, a arquitetura Octagon apresenta escalabilidade linear através do uso de nodos-ponte para interconectar agregados (conforme ilustrado na Figura 3.16.b) ou pela inclusão de nodos com uma porta adicional para interconexão de cada nodo com quatro nodos vizinhos, abordagem que produz uma arquitetura mais compacta quanto ao número de enlaces e à área do leiaute físico.

A rede CLICHÉ

Em [KUM 2002], Kumar et al. propõem uma rede-em-chip baseada em uma topologia em grelha 2-D e denominada CLICHÉ (*Chip-Level Integration of Communicating Heterogeneous Elements*), a qual é mostrada na Figura 3.17. Segundo a terminologia utilizada pelos autores, a rede possui $m \propto n$ chaves (roteadores), nas quais são conectados os recursos do sistema (núcleos ou agregados de núcleos). Os recursos são posicionados nas lacunas (*slots*) disponíveis entre as chaves, sendo que recursos maiores que a área de uma lacuna podem ocupar múltiplas lacunas organizadas sob a forma de regiões. Uma região é uma área dentro da rede que pode ter diferentes topologias e mecanismos de comunicação, sendo conectada à rede através de um adaptador de comunicação (*wrapper*). Essa abordagem permite separar o desenvolvimento, o gerenciamento e a instanciação das diferentes regiões. Os autores prevêem que a área de um recurso será limitada à máxima região síncrona de uma dada tecnologia. Com avanço dos processos de fabricação essa área será reduzida e o número de recursos irá aumentar, assim como a largura de banda das comunicações entre as chaves e entre os recursos e as chaves. Porém, os protocolos de comunicação da rede não serão afetados.

Para uma tecnologia 60 nm CMOS, esperada para 2008, os autores prevêem que, em um chip com área de $22\ \text{mm} \propto 22\ \text{mm}$, será possível construir um sistema que acomode $10 \propto 10$ recursos, cada um com uma área de $2\ \text{mm} \propto 2\ \text{mm}$. Quanto à infra-

estrutura da rede, os autores prevêem que cada chave ocupará $30\,\nu m \propto 30\,\nu m$ e os canais de comunicação terão uma largura de $30\,\nu m$, com a disponibilidade de 300 fios por canal, sendo que 256 desses fios serão reservados aos dados e os outros 44 fios serão alocados para as funções de enquadramento, controle de fluxo e roteamento.

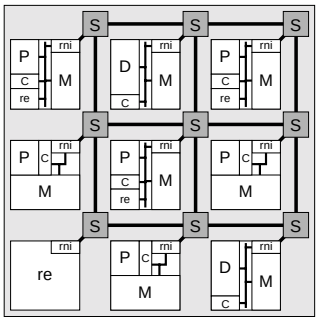


FIGURA 3.17 . Sistema baseado na rede CLICHÉ [KUM 2002].

Além da rede CLICHÉ, os autores também propõem uma metodologia de projeto baseada em plataforma denominada BPS (*Backbone-Platform-System*). Ela é baseada na idéia de se ter uma plataforma de aplicação específica baseada em uma "espinha dorsal" (*backbone*) na qual as aplicações finais possam ser mapeadas como software ou hardware configurável. A metodologia estabelece três fases diferentes no projeto de um SoC baseado na rede CLICHÉ. Durante a fase denominada "desenvolvimento do *backbone*", o foco do projeto é a rede-em-chip (chaves, interfaces, canais e protocolos de comunicação) e a definição dos tipos de região. Na fase denominada "desenvolvimento da plataforma", é criada a plataforma de computação para uma dada área de aplicação, envolvendo o escalamento da rede, a definição das regiões, o projeto dos recursos e a definição do sistema de controle. Por fim, na fase denominada "desenvolvimento da aplicação", a funcionalidade da aplicação é mapeada para os recursos. Em resumo, a metodologia estabelece a criação de uma arquitetura a partir de um modelo geral de rede e o mapeamento da aplicação na arquitetura.

3.4 Considerações

Neste capítulo, procurou-se tratar das questões relativas à comunicação em sistemas integrados. Após a apresentação de alguns conceitos, classificações e generalidades a respeito de SoCs, buscou-se aprofundar o tema proposto através de estudos de caso, identificando-se limitações e soluções encontradas para o problema da comunicação. Pôde-se ver que, embora o barramento seja a solução de melhor relação custo/desempenho para os sistemas integrados atuais, os futuros SoCs, com dezenas a centenas de núcleos, terão requisitos de comunicação difíceis de serem suportados por estruturas de tempo compartilhado, como o barramento. Para tais sistemas, a solução que atende a todos os requisitos identificados consiste nas redes-em-chip, pois são reutilizáveis, possuem largura de banda escalável, oferecem paralelismo em comunicação e apresentam melhores características quanto ao consumo de energia e à freqüência de operação. No capítulo a seguir, é apresentada uma metodologia para avaliação do desempenho de comunicação de redes-em-chip, a qual foi aplicada durante na modelagem e na avaliação da rede SPIN.

4 Modelagem e Simulação de Redes-em-Chip

Este capítulo apresenta uma metodologia para avaliação do desempenho em comunicação de redes-em-chip, a qual se baseia no uso de um simulador para sistemas integrados com precisão de ciclos denominado CASS (*Cycle-Accurate System Simulator*). Nessa metodologia, a rede a ser avaliada é modelada utilizando uma extensão da linguagem C para a descrição de sistemas síncronos no nível de transferência entre registradores. Após, a rede é instanciada em conjunto com outros modelos de núcleos, disponíveis em uma biblioteca de componentes, para a construção de um modelo de sistema integrado. Esse sistema é simulado e obtém-se o dados para a avaliação da rede, como, por exemplo, o número de ciclos para completar a execução da aplicação ou a latência média de comunicação. O desempenho medido pode então ser comparado ao obtido com outras arquiteturas de comunicação, como, por exemplo, um barramento central.

Essa metodologia foi aplicada durante estágio realizado no Departamento ASIM do LIP6 sob a orientação do professor Alain Greiner. Foram desenvolvidos um modelo CASS do roteador RSPIN, modelos de sistemas integrados baseados na rede SPIN e *benchmarks* para validação dos modelos e avaliação de desempenho desses sistemas. Os resultados obtidos foram então comparados com medidas de execuções dos *benchmarks* em modelos de sistemas equivalentes baseados no barramento PI-Bus. Todos os componentes de processamento utilizados na modelagem desses sistemas possuem interface VCI e são utilizados adaptadores para conectá-los às arquiteturas de comunicação. Alguns dos resultados obtidos foram publicados nos *Designer's Forum* da conferência *Design Automation and Test on Europe* – DATE'2003 [AND 2003].

Na organização deste capítulo, primeiramente, são apresentados aspectos relacionados ao simulador CASS, ao padrão VCI e ao barramento PI-Bus. Após, é mostrado o processo de modelagem do roteador, descrevendo-se a organização do roteador e a estrutura do modelo desenvolvido. Finalmente, são descritos os sistemas utilizados na avaliação de desempenho e os resultados obtidos.

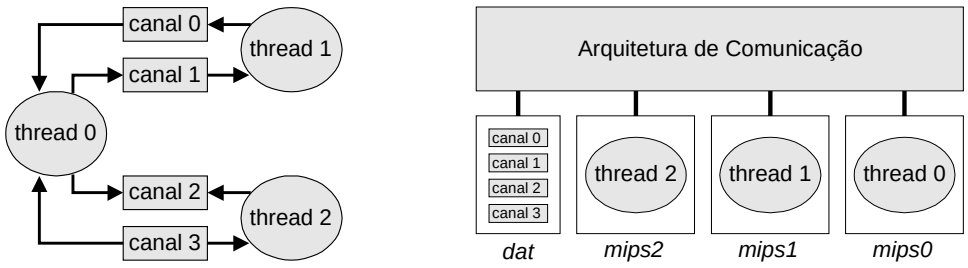
4.1 O Simulador CASS

O simulador CASS (*Cycle Accurate System Simulator*) é um simulador ciclo-a-ciclo para modelagem e simulação de sistemas síncronos [PET 97, GRE 2001]. Ele é baseado no modelo de autômatos comunicantes e permite a descrição de uma arquitetura como um conjunto de módulos interconectados por sinais, sendo que os módulos são descritos no nível de transferência entre registradores (RTL – *Register Transfer Level*).

No CASS, a arquitetura do sistema é especificada através de um arquivo VHDL estrutural (extensão *.vst*), no qual são instanciados componentes disponíveis nas bibliotecas (arquivos com extensão *.a*) e definidas as interconexões entre as instâncias. No arquivo *.vst*, pode-se instanciar várias vezes um mesmo componente, como, por exemplo, o modelo de um processador ou de uma memória RAM. Cada módulo deve ser descrito em linguagem C e compilado de modo a obter-se a biblioteca correspondente [AND 2003].

Atualmente, as bibliotecas de componentes do CASS incluem modelos para a construção de sistemas mono ou multiprocessados baseados no processador MIPS R3000. Elas incluem também modelos para módulos de memória RAM, controladores de E/S, geradores de tráfego e dois tipos de arquitetura de comunicação: o barramento PI-Bus e a rede SPIN (cujo modelo foi implementado no contexto deste trabalho). As bibliotecas incluem versões de componentes com interfaces compatíveis com o barramento PI-Bus e com o padrão VCI. Além disso, são disponibilizados adaptadores de comunicação para a conversão de protocolo PI-Bus/VCI e SPIN/VCI, pois os componentes compatíveis com o padrão VCI requerem o uso desses adaptadores para serem conectados a uma das arquiteturas de comunicação.

O simulador CASS dispõe ainda de um sistema operacional *multithread* denominado Mutek, o qual implementa as funções *thread Posix*. No Mutek, as aplicações podem ser implementadas como *threads* comunicantes que trocam mensagens através de acessos de escrita e leitura em canais de comunicação. As *threads* podem ser mapeadas em múltiplos processadores de modo a acelerar a computação. Os canais de comunicação, por sua vez, são mapeados em um componente de memória RAM compartilhada pelas *threads*. Por exemplo, na Figura 4.1.a, é mostrada uma aplicação hipotética implementada por três *threads* comunicantes (numeradas de 0 a 2), as quais trocam mensagens utilizando quatro canais (ou FIFOs) de comunicação. Essa aplicação é mapeada para um sistema igualmente hipotético (Figura 4.1.b), sendo que as threads são mapeadas para três instâncias do modelo de processador MIPS R3000 e os canais de comunicação são mapeados para uma instância de um modelo de memória RAM (denominada *dat*), na qual também são colocados os dados do sistema operacional e das *threads* da aplicação. Por ser irrelevante à explicação, na figura, é omitido o tipo de arquitetura de comunicação utilizado no sistema.



(a) (b)

FIGURA 4.1 – Implementação de aplicações *multi-threaded* no CASS: (a) *threads* comunicantes; (b) mapeamento da aplicação em um sistema multiprocessado.

4.1.1 Fluxo de modelagem e simulação

Na Figura 4.2, é ilustrado o fluxo de modelagem e simulação de sistemas usando o CASS. A simulação é executada a partir do arquivo de descrição do sistema (*.vst*), das bibliotecas de componentes (*.a*) e do arquivo executável da aplicação (*.out*) gerado pelo compilador/ligador C com base na descrição da aplicação (arquivos *.c* e *.h*). Ao executar a simulação, o CASS pode gerar dois tipos de arquivo de saída. Se o sistema incluir uma instância do modelo TTY, será possível imprimir mensagens em um arquivo de saída através do uso da função C *printf*. De fato, esse componente é um controlador de saída que imprime mensagens em uma janela do monitor do computador, as quais são

também registradas em um arquivo denominado *tty.out*. O segundo arquivo de saída, denominado *simulation.txt*, permite o registro do estado dos registradores e dos sinais das interfaces dos componentes a cada ciclo de relógio, o que facilita a depuração dos modelos utilizados.

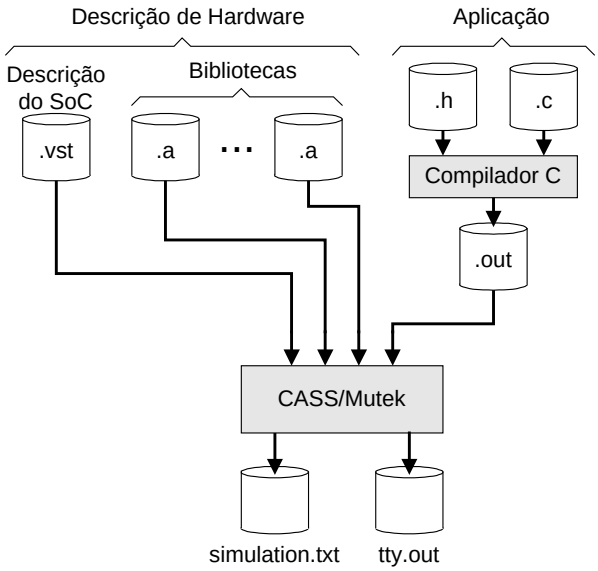


FIGURA 4.2 – Fluxo de modelagem e simulação usando o CASS.

Em sistemas que não utilizem modelos de componentes programáveis (eg. processador MIPS R3000), não há aplicação a ser desenvolvida e carregada. Um exemplo de sistema desse tipo pode incluir apenas instâncias dos modelos de gerador de tráfego e de memória RAM, como será visto mais tarde.

4.1.2 Descrição de um modelo CASS

Conforme apresentado em [GRE 2001], um modelo CASS de um componente denominado *XXX* [17] é descrito em linguagem C através dos arquivos *XXX.h* e *XXX.c*, ilustrados na Figura 4.3. O primeiro arquivo (*XXX.h*) contém a estrutura de dados *XXX*, a qual inclui os parâmetros estruturais do componente, sua interface (portas de entrada e de saída) e os registradores que representam o estado interno de uma instância do modelo. O segundo arquivo (*XXX.c*) inclui de duas a três funções utilizadas para a criação de uma instância e para a descrição dos comportamentos seqüencial e combinacional do componente. Essas funções são denominadas, respectivamente: *CreateXXX*( ), *SequentialXXX*( ) e *CombinationalXXX*( ), sendo que esta última não é implementada quando o componente é baseado em um autômato de Moore. O arquivo *XXX.c* deve incluir os arquivos *access.h* e *beginner.h*, os quais definem as primitivas utilizadas para acessar a estrutura de dados, permitindo a leitura e escrita nas portas e registradores do modelo. Essas funções são denominadas *ReadPORT*( ), *ReadREGISTER*( ), *WritePORT*( ) e *WriteREGISTER*( ).

[17] *XXX* é o nome de um módulo genérico qualquer (eg. *router*, *vciram*).

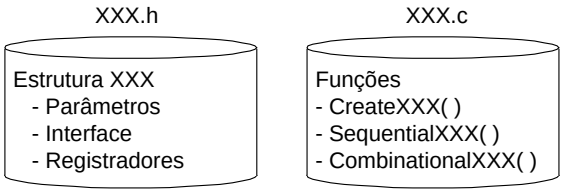


FIGURA 4.3 – Arquivos de um modelo CASS.

A função *CreateXXX*( ) cria uma instância particular do modelo *XXX* e deve declarar todas as portas de entrada e saída por meio do uso da primitiva *DeclareModelPort*( ). Ela é chamada uma única vez no início da simulação e carrega os parâmetros estruturais da instância.

A função *SequentialXXX*( ) inclui as funções de transição e de geração de Moore. Ela é chamada uma vez em cada ciclo de simulação e é constituída por três seções:

a) Descrição da função de transição – nesta seção, podem ser utilizadas as primitivas *ReadPORT*( ), *ReadREGISTER*( ) e *WriteREGISTER*( ). Podem ser feitas várias leituras de uma mesma porta ou de um mesmo registrador. Contudo, só é permitida uma única escrita em cada registrador, sendo que as escritas nos registradores são realmente efetuadas após a chamada da função *UpdateREGISTER*( ). Dessa forma, duas leituras efetuadas antes e após uma escrita em um registrador retornarão o mesmo valor se elas forem posicionadas acima da chamada *UpdateREGISTER*( ) no arquivo que descreve o modelo. Isso permite a descrição facilitada de vários autômatos operando em paralelo, sem a preocupação com a ordem na qual são calculados os novos valores dos registradores. Por fim, a utilização da primitiva *WritePORT*( ) é proibida nesta seção.

b) Atualização dos registradores – nesta seção, é implementada a barreira de sincronização a partir da qual as escritas realizadas nos registradores serão realmente efetivadas. Isso é feito pela chamada à função *UpdateREGISTER*( ).

c) Descrição da função de geração de Moore – nesta seção, são feitas as escritas nas portas de saída através da utilização da primitiva *WritePORT*( ). Os valores a serem escritos são calculados a partir de leituras dos registradores atualizados na seção anterior. Qualquer variável intermediária que dependa do valor desses registradores deve ser recalculada. As primitivas *ReadPORT*( ) e *WriteREGISTER*( ) não podem ser usadas nesta seção.

A função de geração de Mealy é descrita na função *CombinationalXXX*( ), a qual deve ser vazia quando é descrito um autômato de Moore. Contudo, no caso de um autômato de Mealy, ela pode ser chamada várias vezes em um mesmo ciclo de simulação. Logo, é importante que se limite o número de sinais de Mealy para melhorar o desempenho da simulação. Nessa função, que tem uma única seção, devem ser utilizadas as primitivas *ReadREGISTER*( ) e *ReadPORT*( ) para ler o conteúdo dos registradores e das portas de entrada. As saídas de Mealy são atualizadas através da primitiva *WritePORT*( ), sendo que é proibido o uso da primitiva *WriteREGISTER*( ).

4.2 O Padrão VCI

O VCI (*Virtual Component Interconnect*) é um padrão de interface para a interconexão de núcleos em sistemas integrados [VSI 2000], sejam esses sistemas baseados em canais ponto-a-ponto, em um barramento ou uma rede-em-chip. Ele estabelece conexões lógicas ponto-a-ponto entre os núcleos iniciador-VCI e alvo-VCI da comunicação. O iniciador é aquele que tem a capacidade de iniciar a comunicação e enviar uma requisição ao alvo, o qual executa a requisição e envia uma resposta ao iniciador. Ao iniciador, só é permitido o envio de requisições e, ao alvo, o envio de respostas. Uma conexão VCI é então formada por dois canais lunidirecionais, sendo um para o iniciador enviar requisições e outro para o alvo enviar as respostas (Figura 4.4). As larguras das vias de endereço e dado são definidas pelos requisitos dos componentes ligados pela conexão VCI. Exemplos de requisição VCI e resposta VCI incluem, respectivamente, uma requisição para leitura de um dado de uma posição do alvo e uma resposta incluindo o dado lido.

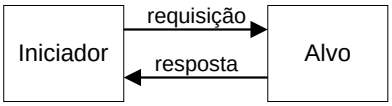


FIGURA 4.4 – Conexão VCI ponto-a-ponto.

Em uma transação VCI, as informações são transferidas sob a forma de pacotes de requisição e de resposta. Por definição, um pacote VCI é o objeto de transporte pelo qual são transferidas uma ou mais células VCI, de forma ordenada e atômica, através de um canal VCI. (transferências simples e em rajada). Uma célula VCI corresponde à unidade de informação básica transferida no canal VCI em um único ciclo, sendo definida pela largura física em bytes da via de dados implementada no canal. Um pacote de resposta deve conter o mesmo número de células do pacote de requisição ao qual ele se refere. Além disso, um protocolo de *handshake* deve ser aplicado a cada célula do pacote de modo a garantir o controle do fluxo de dados no canal VCI.

Cada núcleo compatível com o VCI deve implementar o tipo de interface correspondente à sua funcionalidade: iniciador-VCI ou alvo-VCI, conforme os exemplos da Figura 4.5.a e da Figura 4.5.b, respectivamente. Se um determinado núcleo possui as duas funcionalidades (por exemplo, um coprocessador), ele deve implementar os dois tipos de interface (Figura 4.5.c). Contudo, se a interface do núcleo não for compatível com o VCI, um circuito adicional deve ser utilizado para realizar a adaptação dos dois protocolos (Figura 4.5.d). Esse circuito é denominado adaptador, sendo que ele deve implementar o mínimo de armazenamento de endereço e de dado, enquanto que a latência por ele acrescentada deve ser a menor possível. Em inglês, esse tipo de circuito é referenciado pelo termo *wrapper* (envolvedor ou envelopador) por envolver o circuito original.

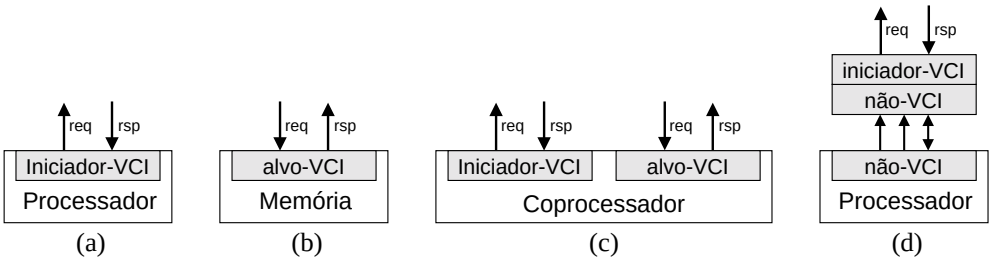


FIGURA 4.5 – Uso da interface VCI: (a) processador; (b) memória ; (c) coprocessador; (d) processador não-VCI com adaptador iniciador-VCI.

Para interconectar núcleos compatíveis com o padrão VCI, é necessário que a arquitetura de comunicação a ser utilizada ofereça terminais também compatíveis com o padrão VCI. Em geral, as arquiteturas de comunicação utilizam protocolos proprietários e cabe ao seu desenvolvedor oferecer os adaptadores necessários à tradução entre o protocolo proprietário e o protocolo VCI a fim de facilitar o trabalho do integrador.

O padrão VCI estabelece três níveis de complexidade visando aplicações com diferentes requisitos: VCI periférico, VCI básico e VCI avançado. O primeiro estabelece uma interface mais simples e de menor custo. O segundo define uma interface adequada à maioria das aplicações. Já o último inclui características adicionais, as quais incluem o suporte necessário a aplicações de alto desempenho, como, por exemplo, identificação das *threads* emissora da requisição VCI.

O VCI utiliza o conceito de transação divida (protocolo *split*), na qual as fases de requisição e de resposta são desvinculadas uma da outra. Assim, um iniciador pode enviar tantas requisições quantas forem necessárias, sem precisar aguardar pelas respostas. Esse protocolo é utilizado na versão básica e na versão avançada, sendo que, na primeira, a ordem das respostas deve ser a mesma das requisições. Já no VCI avançado, as requisições podem ser rotuladas com identificadores, o que permite que a ordem das respostas possa ser recuperada, caso seja perdida. Além disso, esse recurso possibilita que requisições de diferentes *threads* sejam intercaladas e as respostas sejam entregues corretamente às *threads* requisitoras. Contudo, tal recurso requer que a interface VCI do iniciador inclua um *buffer* de reordenação que permita a recuperação da ordem das respostas.

4.3 O Barramento PI-Bus

A arquitetura de referência nos trabalhos de modelagem e avaliação de arquiteturas de comunicação intra-chip apresentados neste texto é o barramento PI-Bus (*Peripheral Interconnect Bus*) [SIE 94]. Esse barramento é resultado de um trabalho de especificação realizado pela entidade OMI (*Open Microprocessor Systems Initiative*), integrante do programa ESPRIT da Comunidade Européia. Entre as empresas envolvidas nesse trabalho de especificação, estão incluídas a ARM, a INMOS, a Philips e a Siemens.

O PI-Bus é um protocolo de barramento para ser usado em SoCs com foco principal nos requisitos de comunicação de vários tipos de periféricos integrados no sistema. Ele foi especificado para a realização de transferências mapeadas em memória

entre seus agentes, sendo que cada agente deve possuir ao menos um tipo de interface: mestre ou escravo. Alguns tipos de agentes, como coprocessadores, devem incluir os dois tipos de interface.

Para usar o PI-Bus, o sistema deve possuir um controlador de barramento denominado BCU (*Bus Control Unit*) para realizar as funções de arbitragem do barramento, decodificação de endereços (para a seleção do agente escravo de cada transferência) e controle de *time-out*.

O PI-Bus suporta a conexão de múltiplos mestres ao barramento e possui arquitetura independente de processador e vias separadas para endereço e dado, as quais podem ser escaladas para até 32 bits. As transferências de dados podem ser simples ou em rajadas, sendo que a via de dados pode ser suportar palavras de 8, 16 ou 32 bits, porém sem dimensionamento dinâmico do barramento. Além disso, o protocolo permite operações de leitura e escrita em uma mesma transferência para a realização de operações atômicas do tipo leitura-modificação-escrita, tipicamente utilizadas em acessos a semáforos.

Na Figura 4.6, é ilustrado um sistema PI-Bus genérico e são mostrados alguns dos sinais do barramento: via de endereços, via de dados, linha de requisição do mestre X (REQ_X), linha de confirmação do mestre X (GNT_X) e linha de seleção do escravo Y (SEL_Y). Diversos outros sinais são omitidos na figura para tornar mais clara a visualização da estrutura do barramento PI-Bus. Quando um mestre deseja realizar uma transferência de dados com um escravo, ele deve, primeiramente, solicitar ao BCU o direito de uso do barramento, ativando a sua linha de requisição (REQ). O árbitro do BCU recebe essa requisição e as de outros mestres e aplica um mecanismo de arbitragem para selecionar uma delas. Uma vez concluída a arbitragem e quando o barramento estiver disponível, o árbitro ativa a linha de confirmação (GNT) do agente mestre selecionado, informando-lhe que o barramento lhe foi concedido e que ele pode iniciar a transação. Quando o mestre inicia uma transferência de dados, a primeira fase da transação refere-se ao endereçamento do escravo alvo. Nesse momento, é de responsabilidade do BCU a decodificação dos bits mais significativos do endereço e a seleção do escravo, o que é feito com a ativação da linha SEL do escravo selecionado. A especificação do barramento PI-Bus não determina o esquema de arbitragem a ser utilizado no BCU, ficando a critério do desenvolvedor a escolha do esquema mais adequado ao seu sistema (eg. prioridades estáticas, dinâmicas etc).

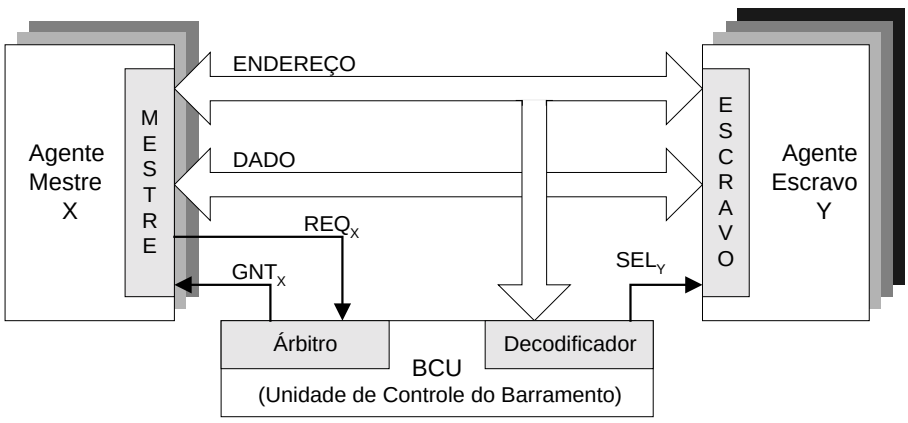


FIGURA 4.6 – Sistema integrado baseado no PI-Bus.

O simulador CASS inclui bibliotecas de modelos de núcleos orientados ao barramento PI-Bus, incluindo o processador MIPS R3000 com caches de instrução e de dado, módulo de memória RAM e controladores de E/S com interfaces mestre e/ou escravo PI-Bus. Inclui também bibliotecas de modelos de núcleos equivalentes com interface VCI e adaptadores VCI/PI-Bus para a conexão desses modelos ao barramento PI-Bus.

4.4 Modelagem da Rede SPIN no CASS

Enquanto um barramento pode ser modelado basicamente através de sua unidade de controle, como, por exemplo, o BCU do PI-Bus, uma rede-em-chip é modelada pela descrição de seus roteadores. Durante o estágio realizado no Departamento ASIM do LIP6, foi efetuada a modelagem da rede SPIN através da descrição do modelo CASS do roteador RSPIN. A descrição desse modelo foi feita com base em um modelo VHDL sintetizável do roteador. Esses dois modelos são equivalentes ciclo-a-ciclo, contudo, o modelo CASS apresenta uma simplificação no que tange a não inclusão das estruturas relacionadas ao teste do roteador. A seguir, são apresentadas a organização do roteador modelado e a estrutura do modelo CASS.

4.4.1 Organização do roteador RSPIN

O roteador RSPIN é o bloco construtivo básico de uma rede SPIN, na qual os roteadores são conectados entre si e com os núcleos do sistema através de enlaces SPIN constituindo uma topologia em árvore gorda. Cada roteador possui oito portas externas, compatíveis com o protocolo físico definido por esse enlace, e, internamente, é organizado em unidades que incluem os *buffers*, multiplexadores e autômatos que realizam a transferência de pacotes pelo roteador.

O enlace SPIN

O enlace SPIN é constituído por dois canais físicos unidirecionais. A largura do canal (tamanho do phit) é de 36 bits, com 32 bits para dado e 4 bits de banda lateral dedicados ao enquadramento do pacote (marcadores de início e de fim) e sinalização de paridade e de erro. No modelo CASS, essas duas partes (de 32 e de 4 bits) e os recursos a elas associados são descritos em campos separados, denominados *data* e *fanion*. Tal partionamento é necessário pois o CASS trabalha com variáveis inteiras de 32 bits para implementar qualquer sinal, seja um bit ou uma palavra com *n* bits. Logo, um canal físico de 36 bits não pode ser representado por uma única variável.

Cada canal do enlace SPIN inclui dois sinais adicionais usados pelo mecanismo de controle de fluxo: *dv* (*data valid*) sinaliza a presença de um dado válido no canal e *cr* (*credito return*) indica a liberação de uma posição no *buffer* de entrada do receptor sob a forma de um crédito, como é mostrado na Figura 4.7 e na Tabela 4.1. A largura de banda do enlace SPIN é de 12,8 Gbit/s para uma freqüência de operação de 200 MHz ($2 \propto 32$ bits $\propto 200$ MHz), ou seja, 6,4 Gbit/s por canal.

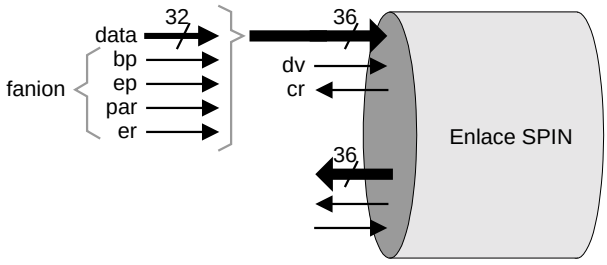


FIGURA 4.7 – O enlace SPIN.

TABELA 4.1 – Significado de cada sinal do enlace SPIN.

Sinal	**Largura**	**Definição**
data	32-bit	Dados
bp	1-bit	Marcador de início do pacote (ativado somente no flit de cabeçalho)
ep	1-bit	Marcador de final do pacote (ativado somente no flit terminador)
par	1-bit	Paridade sobre os 36 bits
er	1-bit	Sinalizador de erro
dv	1-bit	Validação de dado
cr	1-bit	Retorno de crédito

OBS: Os sinais *par* e *er* não são avaliados pelo roteador. Cabe aos adaptadores das interfaces dos núcleos realizar a geração e a verificação dos sinais de paridade e erro, respectivamente

A interface do roteador RSPIN

A interface do roteador RSPIN, ilustrada na Figura 4.8, é composta por oito portas bidirecionais compatíveis com o enlace SPIN. Existem quatro portas inferiores (D0-3), utilizadas para conectar o roteador a roteadores de um nível inferior na rede ou a adaptadores SPIN, e quatro portas superiores (U0-3), reservadas exclusivamente para a conexão com roteadores situados um nível acima na rede SPIN. Cada porta possui dois canais unidirecionais em oposição: o canal de entrada e o canal de saída.

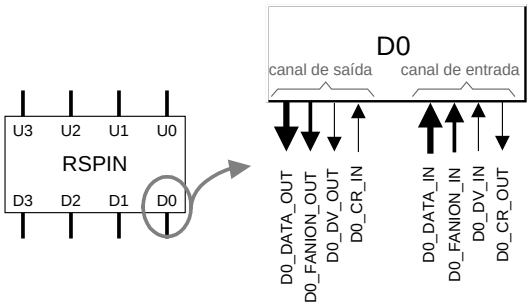


FIGURA 4.8 – A interface do roteador RSPIN e os sinais da porta D0.

Conforme o tamanho da rede, a topologia em árvore-gorda não inclui conexões entre os roteador do nível superior da rede. Nesses roteadore, as portas superiores não são sintetizadas, reduzindo o custo de silício desse nível da rede.

A estrutura interna do roteador RSPIN

Internamente, o roteador RSPIN modelado sob CASS é constituído por dez unidades básicas que implementam os *buffers*, multiplexadores e autômatos do roteador, sendo que existem três tipos de unidade. Nas unidades tipo DN (0 a 3), são implementados os canais de entrada das portas inferiores e os canais de saída das portas superiores. As unidades tipo UP (0 a 3) implementam os canais de entrada das portas superiores e os canais de saída das portas inferiores. Os nomes desses dois tipos de unidade estão diretamente ligados aos canais de entrada aos quais elas estão associadas, conforme é mostrado na Figura 4.9.

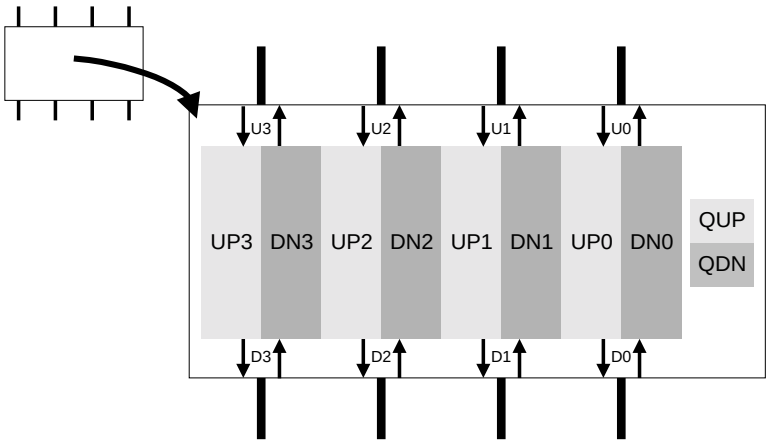


FIGURA 4.9 – As unidades do modelo RSPIN-CASS.

Como foi visto na descrição da rede SPIN (subseção 3.3.1), o roteador RSPIN inclui duas filas centrais compartilhadas utilizadas para armazenar pacotes bloqueados nos *buffers* de entrada do roteador, liberando-os para serem utilizados por outros pacotes. Existe uma fila central para cada lado do roteador e elas são implementadas e gerenciadas pelas unidades tipo Q (QDN e QUP).

O crossbar do roteador RSPIN

O crossbar do roteador RSPIN é realizado de forma distribuída nas unidades do roteador e atende às restrições impostas pelo algoritmo de roteamento, a qual restringe o uso dos canais de saída e das filas centrais, conforme segue:

a) Canais de saída superiores – só podem ser utilizados por pacotes provenientes dos canais de entrada inferiores. Ou seja, nem os canais de entrada superiores nem as filas centrais podem requerê-los[18];

b) Canais de saída inferiores – podem ser utilizados por pacotes provenientes de todos os canais de entrada e de todas as filas centrais;

c) Fila central superior – pode ser utilizada apenas por pacotes bloqueados nos *buffers* dos canais de entrada superiores. Ao retirar pacotes bloqueados nos *buffers* de entrada, as filas centrais incrementam a prioridade desses pacotes para que eles

[18] O algoritmo de roteamento usado na rede SPIN estabelece que um pacote que entra por uma porta superior só pode deixar o roteador através de uma porta inferior, não lhe sendo permitido realizar o retorno por um mesmo enlace.

sejam privilegiados no momento de competir por uma saída inferior na rede. O objetivo é que os pacotes permaneçam o menor tempo possível dentro da rede de modo a reduzir a latência média da comunicação;

d) Fila central inferior – pode ser utilizada apenas por pacotes bloqueados nos *buffers* dos canais de entrada inferiores que sejam destinados a canais de saídas inferiores.

Considerando essas restrições, o crossbar implementa apenas as conexões permitidas pelo algoritmo de roteamento, conforme é mostrado na Figura 4.10, o que leva a uma simplificação do crossbar e a uma conseqüente redução do seu custo.

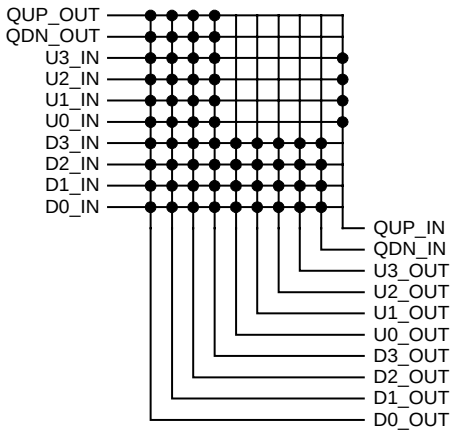


FIGURA 4.10 – Conexões possíveis no crossbar do roteador RSPIN.

A parte operativa do roteador RSPIN

O roteador RSPIN é organizado conforme o modelo parte controle – parte operativa (PC–PO) [DAV 83, CLA 73, FLE 80]. Nesse modelo, a parte operativa inclui o conjunto de registradores, operadores e conexões, enquanto que a parte controle inclui a(s) máquina(s) de estado(s) que realiza(m) o seqüenciamento das operações, selecionando a rede de conexões, os registradores e os operadores que devem atuar em cada passo do algoritmo implementado [CAR 2001].

A parte operativa do roteador RSPIN modelado sob CASS é representada na Figura 4.11, na qual são mostrados os *buffers* e multiplexadores de chaveamento associados a cada unidade, bem como os canais de dado que interligam cada um desses componentes. Comparando-se esta figura com a anterior, pode-se verificar que cada coluna da Figura 4.10 corresponde a um multiplexador da Figura 4.11, enquanto que cada entrada de multiplexador nesta última figura corresponde a um ponto de conexão na Figura 4.10.

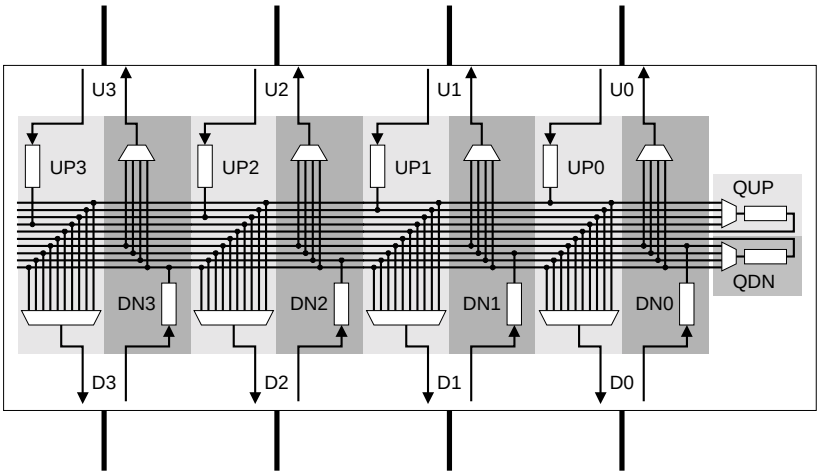


FIGURA 4.11 – A parte operativa do modelo RSPIN-CASS.

A parte controle do roteador RSPIN

A parte controle do roteador RSPIN é implementada de forma distribuída. Cada unidade possui um conjunto de autômatos responsáveis pelos mecanismos de roteamento, chaveamento, arbitragem e controle de fluxo. Esses mecanismos são integrados em dois blocos de controle principais, chamados ICB (*Input Control Block*) e OCB (*Output Control Block*). Um terceiro bloco de controle é responsável pelo gerenciamento do acesso (escrita e leitura) ao *buffer* local.

O ICB é conectado diretamente à saída do *buffer* local da unidade à qual ele é associado. Esse bloco realiza o roteamento dos pacotes, identificando a presença de um cabeçalho de pacote, executando o algoritmo de roteamento e enviando uma requisição à saída ou à fila central selecionada pelo algoritmo. Uma requisição será enviada à fila central quando o canal de saída desejado já estiver ocupado e, se, além disso, o pacote estiver habilitado a utilizar esse recurso através da sinalização em um bit específico no cabeçalho. Existem três tipos de ICBs, um para cada tipo de unidade (DN, UP e Q), os quais podem requisitar apenas os recursos definidos pelo algoritmo de roteamento.

Para implementar o roteamento adaptativo para os pacotes direcionados às saídas superiores, o ICB-DN utiliza um mecanismo que permite selecionar qualquer saída superior disponível (UP0-3), minimizando a contenção na rede. A seleção da saída superior a ser requisitada é feita utilizando um árbitro cuja ordem das prioridades varia a cada ciclo de relógio, sendo a seleção de saída realizada de forma aleatória. Para isso, é utilizado um contador módulo quatro que serve de índice para a seleção da ordem de prioridades a ser aplicada. Assim, a cada ciclo de relógio, uma ordem de prioridades diferente é usada. A garantia da minimização da contenção é dada inciando-se o contador de cada unidade com o índice da sua identidade. Com isso, em um mesmo ciclo de relógio, se todas as saídas superiores estiverem disponíveis e cada entrada inferior tiver um pacote direcionado a um nível superior na rede, cada pacote irá requisitar uma saída diferente. A contenção irá surgir quando o número de saídas superiores implementadas for menor que o número de pacotes requerendo esse tipo de saída.

O outro bloco de controle, denominado OCB, é responsável pelas funções de arbitragem, chaveamento e controle de fluxo. Ele recebe requisições dos blocos ICB, aplica a política de arbitragem adequada para eleger uma das requisições e comanda o multiplexador para efetuar o chaveamento de modo a conectar a saída do *buffer* selecionado ao recurso por ele gerenciado (canal de saída, nas unidades tipo UP e DN, ou fila central, nas unidades tipo Q). A política de arbitragem varia com o tipo de unidade associada, mas todas elas aplicam, ao menos, uma fila circular (função *round-robin*) para avaliar um subconjunto das requisições recebidas, conforme segue:

a) OCB-DN – aplica uma política de arbitragem de dois níveis. Para requisições provenientes das unidades UP e DN, o árbitro seleciona uma requisição aplicando a função *round-robin*. Após, realiza a seleção final utilizando um esquema de prioridades estáticas, conforme o seguinte critério[19]:
$P_{ICB\text{-}QUP}$? $P_{ICB\text{-}QDN}$? $P_{round\text{-}robin(ICBs\text{-}UP)}$? $P_{round\text{-}robin(ICBs\text{-}DN)}$

b) OCB-UP – aplica uma política de arbitragem de um único nível, utilizando a função *round-robin* para selecionar uma entre as quatro requisições permitidas, as quais são oriundas dos ICBs-DN;

c) OCB-Q – aplica uma política de arbitragem idêntica àquela utilizada pelos OCBs-UP. Contudo, para a unidade QUP, as requisições analisadas são as provenientes dos ICBs-UP e, para as unidades QDN, são consideradas as requisições oriundas dos ICBs-DN.

Na Figura 4.12, são ilustrados os controladores ICB e OCB de todas as unidades. Pode-se observar que o ICB lê a saída do *buffer* local e envia requisições aos OCBs (na figura, são mostradas as linhas de requisição possíveis para o ICB-UP1). O OCB, por sua vez, recebe as requisições dos ICBs e comanda o multiplexador local para efetuar o chaveamento.

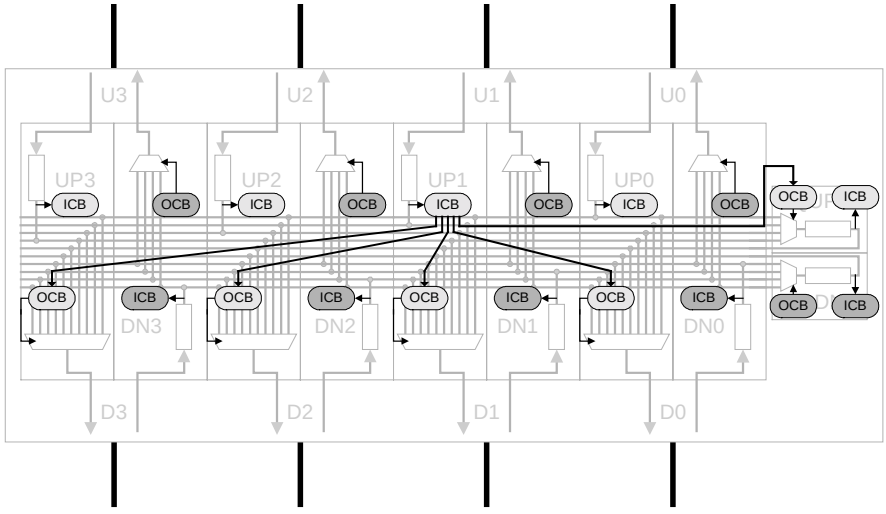


FIGURA 4.12 – Controladores ICB e OCB do RSPIN.

[19] $P_{ICB\text{-}X}$ significa "prioridade da requisição ICB do tipo X", onde X pode ser QUP, QDN, UP ou DN; e $P_{round\text{-}robin(ICBs\text{-}X)}$ significa "prioridade da requisição selecionada pela função *round-robin* aplicada às requisições ICBs do tipo X.

Latência do roteador RSPIN

A latência mínima para um pacote atravessar o roteador RSPIN é determinada pela quantidade de ciclos necessária para:

a) Registro do cabeçalho do pacote no *buffer* do canal de entrada;

b) Execução do algoritmo de roteamento pelo ICB do canal de entrada e o conseqüente envio de requisição ao OCB selecionado; e

c) Arbitragem das requisições recebidas pelo OCB do canal de saída e estabelecimento do caminho entre o *buffer* do canal de entrada selecionado pelo OCB e o canal de saída (chaveamento).

Como uma dessas tarefas consome um ciclo de relógio, a latência mínima seria de três ciclos. Contudo, o mecanismo utilizado para implementar o roteamento adaptativo produz um problema denominado: “problema da dupla requisição”. Quando um pacote é destinado a uma porta superior, o algoritmo de roteamento utiliza um contador para selecionar o canal de saída a ser requisitado. Esse contador é incrementado, ciclicamente, a cada ciclo de relógio e, se o algoritmo de roteamento é executado em todos os ciclos, a informação de referência para a seleção do canal de saída a ser requisitado será diferente a cada ciclo. Com isso, pode ocorrer de o árbitro de um canal de saída superior selecionar uma requisição enviada por canal de entrada em um ciclo anterior, enquanto que o ICB responsável por essa requisição esteja enviando uma nova requisição para um outro canal de saída apontado pelo contador no ciclo corrente. Essa segunda requisição poderia provocar o estabelecimento de um caminho para um segundo canal de saída, levando ao funcionamento incorreto do roteador.

A solução encontrada pelos arquitetos da rede SPIN foi estabelecer ciclos separados para a execução dos algoritmos de roteamento e arbitragem. Um autômato de dois estados rotula os ciclos do relógio em ciclos dos tipos “ímpar” e “par”. O roteamento é executado somente nos ciclos ímpares e a arbitragem nos ciclos pares. Assim, enquanto as requisições estão sendo arbitradas, nenhuma nova requisição pode ser emitida e, no ciclo de roteamento seguinte, os *buffers* selecionados pelos árbitros não conterão mais cabeçalhos de pacotes, os quais já terão sido encaminhados aos canais de saída. Isso evita que esses mesmos pacotes requisitem duas saídas diferentes.

Contudo, a limitação dessa solução reside no fato de que se o cabeçalho de um pacote é registrado durante um ciclo ímpar, o ciclo seguinte será um ciclo de arbitragem (par). Assim, o pacote terá de esperar um ciclo a mais para ser roteado. Logo, a latência mínima para um pacote atravessar o roteador irá variar de três a quatro ciclos, sendo, na média, igual a 3,5 ciclos.

A latência calculada acima é considerada a latência mínima média sem contenção. Ou seja, quando os pacotes não competem pelo mesmo recurso no roteador. Esse valor serve de referência para calcular a latência da comunicação em uma rede SPIN na qual um pacote deve atravessar um ou mais roteadores para chegar ao seu destinatário. Nas referências sobre a rede SPIN, o ciclo referente ao registro do cabeçalho do pacote é omitido e a latência informada é de 2,5 ciclos. Contudo, segundo o entendimento deste autor, é importante deixar claro que existe o custo de um ciclo de relógio para registrar o cabeçalho de um pacote a cada roteador em uma rede-em-chip.

Modelos de rede SPIN

Na Figura 4.13, são apresentados três exemplos de rede SPIN com 8, 16 e 32 terminais. Na rede com 8 terminais (Figura 4.13.a), as comunicações entre núcleos conectados ao mesmo roteador são submetidas a uma latência mínima média de 3,5 ciclos na ausência de contenção. Já as comunicações entre núcleos conectados a diferentes roteadores são submetidas a uma latência mínima média de 7 ciclos. Essas latências levam em conta apenas o tempo entre a injeção do cabeçalho do pacote na rede e a ejeção desse cabeçalho no terminal destinatário. A latência total depende ainda do tamanho da carga útil transportada no pacote e dos atrasos referentes aos adaptadores de comunicação.

Na rede com 16 terminais (Figura 4.13.b), a latência mínima média das comunicações entre núcleos conectados a roteadores diferentes é correspondente à latência necessária para atravessar três roteadores, ou seja, 10,5 ciclos. A rede com 32 terminais (Figura 4.13.c) é constituída por duas subredes (ou agregados) de 16 terminais interconectadas através de enlaces SPIN ligados às portas superiores dos roteadores do segundo nível de cada subrede. As comunicações entre núcleos conectados a roteadores de subredes diferentes são submetidas a uma latência mínima média de 14 ciclos.

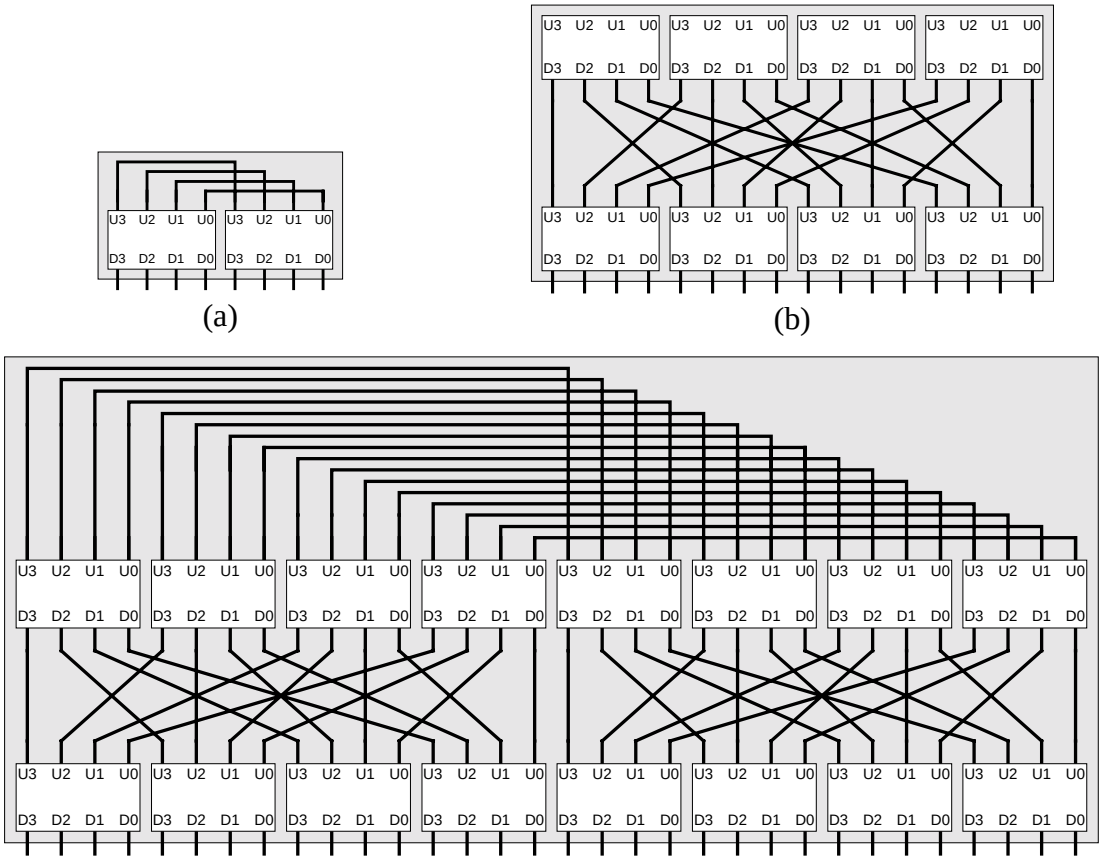


FIGURA 4.13 – Redes SPIN com: (a) 8 terminais; (b) 16 terminais; e (c) 32 terminais.

A escalabilidade da rede SPIN é indicada por Guerrier e Greiner em [GUE 2000a] e reproduzida na Tabela 4.2. Na tabela, é mostrado o número necessário de roteadores RSPIN para cada tamanho de sistema, onde *n* indica o número de terminais da rede.

TABELA 4.2 – Número de roteadores RSPIN na rede SPIN.

n	**Número de RSPINs**
4	1
8	2
16	8
32	16
64	48
128	96

4.4.2 Estrutura do modelo CASS

O modelo CASS do roteador RSPIN é constituído por dois arquivos denominados *router.h* e *router.c*. O primeiro arquivo é uma descrição em linguagem C com 370 linhas de código que realiza a declaração dos parâmetros, interface e registradores do roteador. O segundo arquivo é uma descrição com cerca de 3800 linhas de código C (incluindo comentários) no qual são feitas as definições de todas as funções do modelo.

A modelagem do roteador foi realizada utilizando-se uma abordagem hierárquica de modo a manter a compatibilidade estrutural com o modelo VHDL. Dessa forma, para cada entidade VHDL, foi criada uma função C, sendo que algumas entidades foram agrupadas em uma única função e algumas funções adicionais foram definidas. No nível mais alto da hierarquia, a qual é mostrada na Figura 4.14, estão as funções obrigatórias em qualquer modelo CASS: a função de criação de uma instância (chamada *CreateRouter*) e a função que define o seu comportamento seqüencial (denominada *SequentialRouter*). O modelo não inclui nenhum autômato de Mealy e, portanto, dispensa a implementação de uma função para o comportamento combinacional.

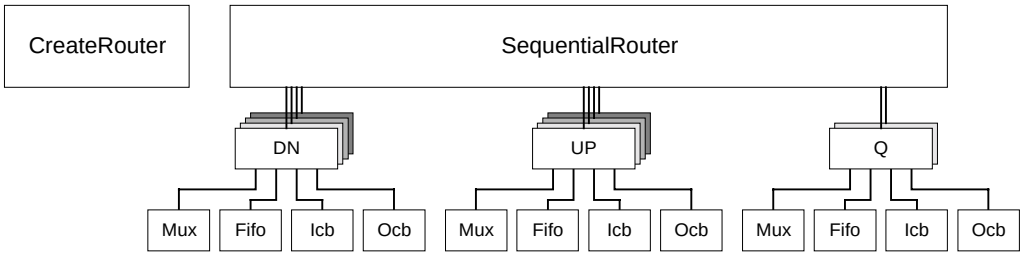


FIGURA 4.14 – Diagrama de hierarquia de funções do modelo CASS do roteador RSPIN.

Conforme é mostrado na Figura 4.14, a função *SequentialRouter* chama um conjunto de funções que implementam as unidades do roteador, sendo que cada uma dessas funções faz uso de funções que definem os seus controladores (*Icb* e *Ocb*), *buffer*

local (*Fifo*) e multiplexador (*Mux*). Outras nove funções são utilizadas nos diferentes níveis, mas elas não são mostradas na Figura 4.14 de modo a conferir uma maior legibilidade ao diagrama. As funções UP, DN e Q são referenciadas múltiplas vezes, o que é representado pela repetição e superposição de blocos.

A estrutura básica da função *SequentialRouter* é ilustrada na Figura 4.15. Ela constitui um laço que é executado a cada ciclo de relógio para determinar os estados dos registradores e das portas de saída do modelo. A execução desse laço é explicada logo a seguir.

```
Se RESET ativado:
÷  Reinicializa todos os registradores

Se RESET não-ativado:
 1. Calcula variáveis intermediárias dependentes do estado corrente
    dos registradores. Essas variáveis são utilizadas pelas funções
    das unidades Q, DN e UP.

 2. Executa a função da unidade  QUP  e escreve nos seus registradores
 3. Executa a função da unidade  QDN  e escreve nos seus registradores
 4. Executa a função da unidade  DN0  e escreve nos seus registradores
 5. Executa a função da unidade  DN1  e escreve nos seus registradores
 6. Executa a função da unidade  DN2  e escreve nos seus registradores
 7. Executa a função da unidade  DN3  e escreve nos seus registradores
 8. Executa a função da unidade  UP0  e escreve nos seus registradores
 9. Executa a função da unidade  UP1  e escreve nos seus registradores
10. Executa a função da unidade  UP2  e escreve nos seus registradores
11. Executa a função da unidade  UP3  e escreve nos seus registradores

12. Executa função que atualiza os registradores com os valores
    escritos após a execução de cada uma das funções anteriores
    (passos 2 a 11)

13. Recalcula as variáveis intermediárias utilizando os estados
    atualizados dos registradores. Essas variáveis são utilizadas no
    cálculo dos valores a serem escritos nas portas de saída.

14. Escreve nas portas de saída da interface externa do roteador.
```

FIGURA 4.15 – Estrutura da função *SequentialRouter*.

Durante a simulação, se o sinal RESET é ativado, o simulador reinicializa todos os registradores do roteador. Caso contrário, ele calcula os valores de um conjunto de variáveis intermediárias, as quais correspondem a sinais internos do roteador e dependem do estado dos registradores no início do ciclo (passo 1). Posteriormente, são executadas as funções das unidades do roteador para calcular os novos valores dos registradores (passos 2 a 11), sendo que, após a execução de cada função, os valores calculados são escritos nos registradores associados à unidade executada. Contudo, esses valores são atualizados apenas com a execução da função *UpdateREGISTER* (passo 12). No passo 13, são recalculados os sinais internos do roteador com base no novo estado dos seus registradores. Finalmente, no passo 14, é feito o calculo e a atualização dos conteúdos das portas de saída do modelo (passo 14).

A validação do modelo CASS do roteador foi feita através da modelagem de diferentes sistemas integrados utilizando redes SPIN com 4, 8, 16 e 32 terminais. Sobre esses sistemas foram executadas diferentes aplicações que permitiram verificar a correção do modelo descrito.

4.5 Avaliação de Desempenho

Nesta seção, são apresentados os resultados de experimentos realizados utilizando o simulador CASS para a simulação de sistemas baseados no barramento PI-Bus e na rede SPIN. Essas simulações foram realizadas visando a avaliação e a comparação do desempenho em comunicação dessas arquiteturas em diferentes configurações de tamanho de sistema, tamanho de mensagem e carga de comunicação. Nos primeiros experimentos, foram utilizados geradores de tráfego parametrizáveis que permitiram a execução de dois tipos de avaliação. Incialmente, fez-se variar o tamanho do sistema, mantendo-se fixo o tamanho dos pacotes VCI e mediu-se o número de ciclos necessários para a entrega de todas as mensagens trocadas pelos núcleos. Após, fixou-se o tamanho do sistema e fez-se variar a taxa de geração de pacotes a fim de determinar os pontos de saturação das arquiteturas de comunicação para diferentes tamanhos de pacote. Em um terceiro conjunto de experimentos, foram modelados sistemas programáveis multiprocessados baseados em múltiplas instâncias de modelos do processador MIPS R3000 de memórias RAM. Além desses sistemas, foi desenvolvido um *benchmark* paralelo baseado na Transformada Rápida de Fourier (FFT – *Fast Fourier Transform*). Essa aplicação foi executada sobre os sistemas baseados nas duas arquiteturas de comunicação e foi medido o número de ciclos gastos na execução da aplicação em cada sistema.

4.5.1 Avaliação da escalabilidade da arquitetura de comunicação

Nesta avaliação, foram modelados sistemas integrados com diferentes tamanhos com o objetivo de avaliar e comparar a escalabilidade do PI-Bus e da rede SPIN. Foram considerados sistemas com 4, 8, 16 e 32 núcleos, sendo metade do núcleos do tipo iniciador-VCI e a outra metade do tipo alvo-VCI. Os núcleos do tipo iniciador-VCI são instâncias de um modelo de gerador de tráfego (GT) que envia uma requisição VCI de tamanho configurável a cada um dos alvos-VCI do sistema. Os núcleos do tipo alvo-VCI são instâncias de um modelo de memória RAM que espera por requisições VCI e retorna respostas VCI com o tamanho das requisições recebidas.

Na Figura 4.16, é ilustrado um sistema integrado com quatro núcleos denominados *iniciador*0, *iniciador*1, *alvo*0 e *alvo*1, sendo que os dois primeiros são do tipo iniciador-VCI e os dois últimos são do tipo alvo-VCI. Ambos os núcleos são conectados à arquitetura de comunicação através de adaptadores que realizam a conversão entre o protocolo VCI e o protocolo da arquitetura utilizada. Na Figura 4.16.a, podem ser visualizadas as requisições VCI enviadas pelos núcleos iniciadores, enquanto que as respostas enviadas pelos núcleos do tipo alvo são ilustradas na Figura 4.16.b.

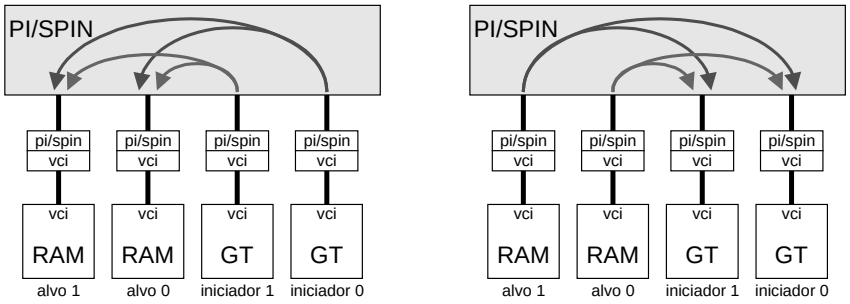


FIGURA 4.16 – Mensagens emitidas em um sistema com quatro núcleos: (a) originadas pelos iniciadores; (b) originadas pelos alvos.

Na carga de comunicação definida, cada iniciador envia uma requisição VCI a cada um dos alvos do sistema e estes últimos, após receberem uma requisição, devem enviar uma resposta ao núcleo iniciador requisitante. O objetivo então é medir o número de ciclos gastos pela arquitetura de comunicação para entregar todas as requisições e respostas VCI enviadas pelos núcleos do sistema.

Na Tabela 4.3, são mostradas as quantidades de pacotes de requisição e de resposta, bem como a quantidade de flits transferidos em cada arquitetura de comunicação para pacotes VCI transportando uma única célula. Como pode ser observado, embora o número de pacotes VCI gerados seja o mesmo em ambas as arquiteturas, o número de flits transferidos é maior na rede SPIN. Isso ocorre porque enquanto que no barramento PI-Bus as palavras de endereço e dado são demultiplexadas, na rede SPIN, o flit é limitado a 32 bits e as palavras de endereço e dado devem ser multiplexadas e, além disso, serem precedidas por um cabeçalho de roteamento.

TABELA 4.3 – Pacotes VCI e flits transferidos no barramento PI-Bus e na rede SPIN.

Número de núcleos	**Pacotes de requisição VCI**	**Pacotes de resposta VCI**	**Flits transferidos no PI-Bus**	**Flits transferidos na SPIN**
4	2∝2 = 4	2∝2 = 4	4+4 = 8	3∝(4+4) = 24
8	4∝4 = 16	4∝4 = 16	16+16 = 32	3∝(16+16) = 96
16	8∝8 = 64	8∝8 = 64	64+64 = 128	3∝(64+64) = 384
32	16∝16 = 256	16∝16 = 256	256+256 = 512	3∝(256+256) = 1536

Nesta avaliação, os sistemas baseados na rede SPIN foram modelados com base na pior condição de posicionamento em termos de latência de comunicação. Todos os iniciadores foram colocados em uma metade da rede, enquanto que os alvos foram posicionados na outra metade, conforme é ilustrado no exemplo da Figura 4.17. Nessa configuração, a latência mínima (sem carga) é dada em função da maior distância entre dois núcleos na rede.

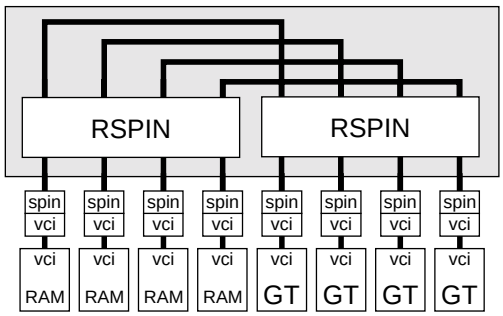


FIGURA 4.17 – Posicionamento de iniciadores e alvos em uma rede SPIN com oito terminais.

Resultados experimentais

Na Figura 4.18, são apresentados os resultados de experimentos de simulação realizados com os sistemas descritos acima. A figura mostra um gráfico relacionando o número de ciclos gastos (em escala logarítmica) pelo barramento PI-Bus e pela rede SPIN para diferentes tamanhos de sistema. Como pode ser observado, para essa carga de comunicação, a rede SPIN supera o barramento PI-Bus em sistemas com um pouco mais de oito de núcleos, se for considerado o ponto de intersecção das curvas interpoladas. Contudo, deve-se levar em conta que o tamanho de mensagem utilizado pode ser considerado o pior caso para a rede SPIN. Para pacotes VCI com um maior número de células como, por exemplo, em uma transferência de linha de cache, o desempenho da rede SPIN será ainda melhor. Isso porque o custo da sobrecarga de comunicação devida ao cabeçalho de roteamento será menor com o aumento da carga útil do pacote.

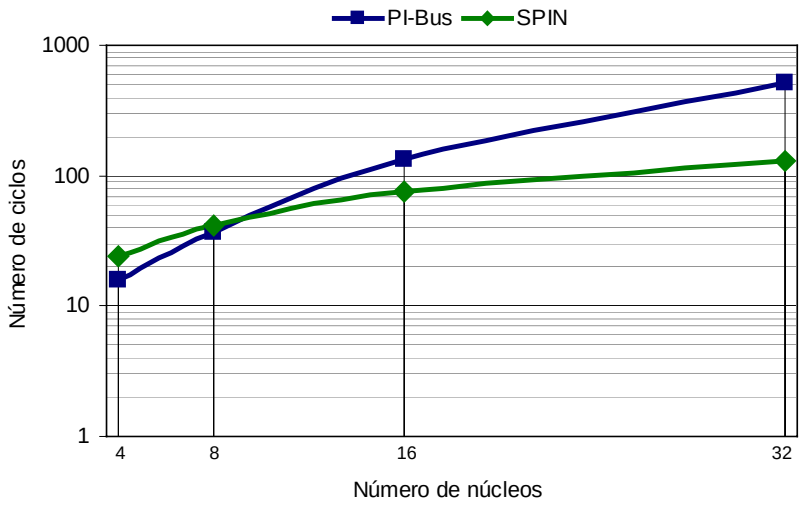


FIGURA 4.18 – Número de ciclos do PI-Bus e da SPIN para pacotes com uma célula VCI.

Ainda analisando os resultados mostrados na Figura 4.18, destaca-se que o melhor desempenho obtido pela rede SPIN se deve ao uso do protocolo *split*, o qual permite a realização de transações divididas. Ou seja, os iniciadores-VCI podem enviar novas requisições antes do recebimento das respostas para as requisições pendentes,

assim escondendo a latência da rede. Para ilustrar isso, na Figura 4.19, são comparados os resultados de experimentos de simulação realizados com e sem o uso do protocolo *split*. Como pode ser visto, quando o protocolo *split* não é utilizado, o número de ciclos gastos no sistema baseado na rede SPIN é maior que o número de ciclos gastos nos sistemas baseados no barramento PI-Bus, para qualquer tamanho de sistema.

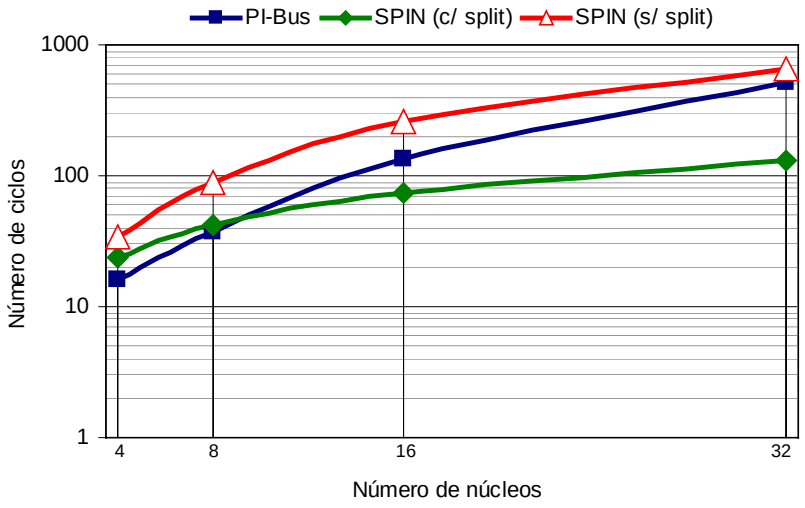


FIGURA 4.19 – Benefícios do protocolo *split* para a rede SPIN.

4.5.2 Avaliação da capacidade da arquitetura de comunicação

Em qualquer arquitetura de comunicação, existe um limite de saturação que surge quando o tráfego gerado pelos iniciadores é maior que a carga suportada pela arquitetura. Em outras palavras, dada uma carga oferecida à arquitetura de comunicação, se essa carga estiver abaixo do seu ponto de saturação, ela irá aceitá-la e a latência média das mensagens estará dentro de um limite considerado igualmente aceitável (eg. até 100 ciclos). Contudo, se a carga aplicada estiver acima do ponto de saturação, a arquitetura de comunicação não será capaz de aceitar toda a carga aplicada e a latência média das mensagens crescerá de modo significativo. Nos experimentos de avaliação da capacidade da arquitetura de comunicação, fixou-se o tamanho do sistema em 32 núcleos (16 iniciadores e 16 alvos), fez-se variar a carga oferecida para diferentes tamanhos de mensagem e foi medida a latência média das mensagens para cada carga a fim de determinar o ponto de saturação da arquitetura para essa configuração de sistema. Os núcleos iniciadores utilizados são instâncias de um modelo de gerador de tráfego que emite uma grande quantidade de pacotes VCI (eg. 10 mil) para endereços distribuídos randomicamente, sendo que cada memória RAM é responsável por uma faixa de endereços.

Os parâmetros de entrada dos experimentos são o número de pacotes VCI a serem emitidos pelos iniciadores (N_{packet}), a quantidade de células em cada pacote VCI (N_{cell}) e o número médio de ciclos entre o envio de dois pacotes de requisição sucessivos (G_{av}). Este último serve como parâmetro de controle para a variável G, a qual determina um intervalo (*gap*) randômico entre dois pacotes sucessivos. A carga oferecida (L_{off}) define o percentual de largura de banda de canal usado por cada iniciador, ou seja:

$$L_{off} = N_{cell} / (N_{cell} + G_{av}) \quad (4.1)$$

Uma carga oferecida de 100% corresponde à configuração na qual $G_{av} = 0$. Quanto maior for o valor de G_{av}, menor será a carga oferecida. Como exemplo, se N_{cell} é igual a 4 e G_{av} é igual a 8, então a carga oferecida é de 33,33%.

Nos geradores de tráfego, para cada pacote de requisição a ser enviado é atribuído um rótulo de tempo. A latência correspondente à transação iniciada pelo pacote é medida pela diferença entre o momento da chegada do pacote de resposta correspondente à requisição emitida e o valor do rótulo de tempo do pacote de requisição. Cada gerador de tráfego possui uma tabela na qual ele guarda as informações a respeito dos pacotes emitidos de modo a poder calcular a latência da comunicação.

Na avaliação da rede SPIN, os núcleos foram distribuídos de uma maneira diferente em relação à avaliação anterior. Em cada roteador do nível mais baixo da rede, foram conectados dois iniciadores e dois alvos de maneira intercalada, conforme é ilustrado no exemplo da Figura 4.20. Com isso, cada iniciador pode estar separado de um alvo na rede por um, três ou quatro roteadores, caso eles estejam conectados no mesmo roteador, no mesmo agregado de 16 terminais ou em agregados diferentes, respectivamente. Essa configuração se fez necessária pois esses experimentos foram conduzidos em uma fase posterior aos primeiros, após o que adotou-se, como solução para o problema do *deadlock*, a divisão reservada dos canais de saídas superiores do roteador RSPIN entre pacotes de requisição e de resposta (conforme comentado no item “Limitações da arquitetura SPIN”, na subseção 3.3.1).

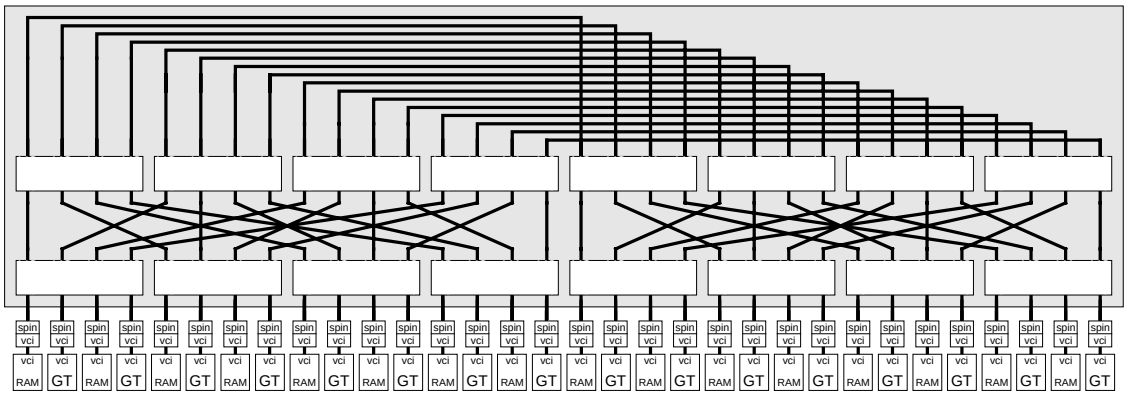


FIGURA 4.20 – Posicionamento de iniciadores e alvos em uma rede SPIN com 32 terminais.

Resultados experimentais

A seguir, são apresentados resultados que mostram a carga máxima aceita pelo barramento PI-Bus e pela rede SPIN para um tamanho fixo de pacote (N_{cell}). Essas simulações foram feitas considerando a realização de 160 mil transações do tipo requisição-resposta (10 mil por iniciador).

A Figura 4.21 apresenta curvas que ilustram os pontos de saturação do barramento PI-Bus e da rede SPIN para pacotes com apenas uma célula VCI. No eixo das abcissas é mostrada a carga aceita por cada arquitetura e no eixo das ordenadas a

latência média para a execução das transações requisição-resposta. Como pode ser observado, o barramento PI-Bus aceita somente 3% da carga máxima, acima da qual a latência média ultrapassa o nível de 100 ciclos de relógio para execução de transações requisição-resposta. Já para a rede SPIN, o limite de carga aceita fica em 14%, porém, sua a latência média para as cargas muito leves (menor que 3%) é cerca de quatro vezes maior que a latência do barramento PI-Bus. Considerando a apenas a latência, esses resultados ilustram que o barramento possui desempenho satisfatório em aplicações não intensivas em comunicação. Contudo, para aplicações com comunicação intensiva, a aplicabilidade do barramento é limitada e, embora a latência da rede SPIN seja maior, ela aceita uma carga de comunicação muito maior que o barramento.

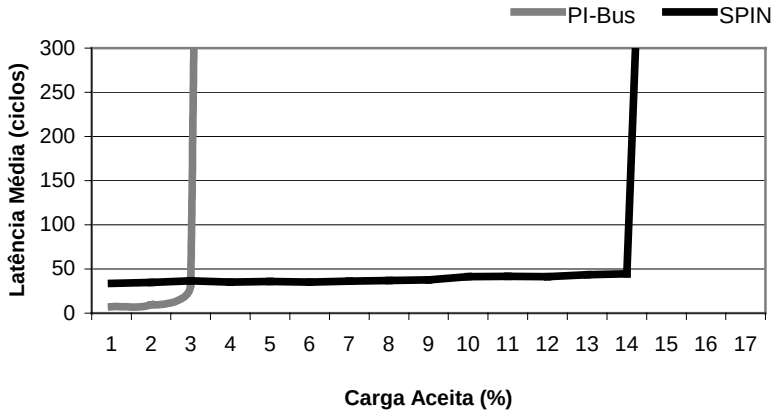


FIGURA 4.21 – Latência média do PI-Bus e da SPIN para N_{cell} = 1.

A Figura 4.22 apresenta a latência média para pacotes com duas células VCI. Observa-se que o ponto de saturação das duas arquiteturas é estendido. O barramento satura com uma carga oferecida de 4% enquanto que a rede SPIN satura com uma carga próxima aos 20%. Esse efeito é justificado pelo fato de que o aumento do tamanho do pacote reduz o número de requisições simultâneas para os recursos da arquitetura, atenuando os conflitos de arbitragem. Um outro fator que favorece a rede-em-chip é que as mensagens são transferidas em modo *pipeline*. Segundo [DUA 97], o tempo de estabelecimento do caminho pelos cabeçalhos é amortizado entre mais flits quando a mensagem é mais longa. Uma vez que o caminho é estabelecido pelo cabeçalho, os flits da carga o seguem avançando mais rapidamente do que este.

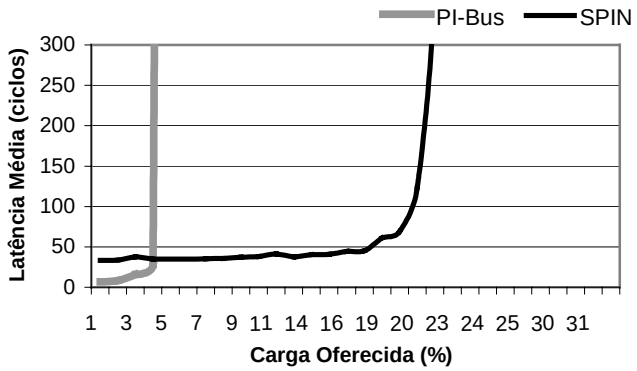


FIGURA 4.22 – Latência média do PI-Bus e da SPIN para N_{cell} = 2.

Na Tabela 4.4, são listados os limites de carga oferecida suportada pelas duas arquiteturas de comunicação para diferentes tamanhos de pacotes. Foi considerado como ponto de saturação latências superiores a 100 ciclos. Observa-se que o aumento do tamanho do pacote permite estender muito pouco a capacidade de carga do barramento. Entretanto, para a rede SPIN, pacotes com 2 e 4 células VCI resultam em um ponto de saturação cerca de 33% maior que o ponto de saturação correspondente a pacotes com apenas uma célula. Para pacotes maiores, o ponto de saturação é estendido ainda mais. Porém, a partir de pacotes com 16 células, o ponto de saturação é limitado a 31%. Acima dessa configuração de tamanho de pacote, o efeito produzido é uma extensão do ponto de saturação se forem consideradas latências médias de até 300 ciclos. Por exemplo, para pacotes com 64 células, a latência média é de 236 ciclos para uma carga oferecida de 38%, enquanto que, para pacotes com 16 células, qualquer carga oferecida superior a 34,7% conduz a uma latência média maior que 300 ciclos.

TABELA 4.4 – Saturação das arquiteturas de comunicação para diferentes tamanhos de pacote.

N_{cell}	PI-Bus	SPIN
1	4%	15%
2	5%	20%
4	5%	20%
8	6%	28%
16	6%	31%
32	6%	31%
64	6%	31%

4.5.3 Avaliação de um sistema multiprocessado executando a FFT

Esta subseção descreve o desenvolvimento de modelos de sistemas multiprocessados baseados no barramento PI-Bus e na rede SPIN. Esses sistemas foram modelados com o objetivo de comparar o desempenho das duas arquiteturas com relação à execução de um *benchmark* tipicamente utilizado na avaliação de redes de interconexão. O *benchmark* escolhido foi a FFT (*Fast Fourier Transform*), sendo que, além da descrição dos modelos dos sistemas, foi necessário implementar uma versão paralela da FFT com base nas primitivas de comunicação suportadas pelo sistema operacional *Mutek* do simulador CASS.

Arquitetura de referência

Na avaliação de desempenho baseada na FFT, primeiramente foi estabelecida uma arquitetura de um sistema integrado monoprocessado definida como arquitetura de referência para o desenvolvimento dos sistemas que seriam avaliados. Nessa arquitetura, ilustrada na Figura 4.23, todos os núcleos de processamento são compatíveis com o padrão VCI e são conectados à arquitetura de comunicação através de adaptadores que realizam a conversão ente o protocolo VCI e o protocolo da arquitetura utilizada (PI Bus ou SPIN).

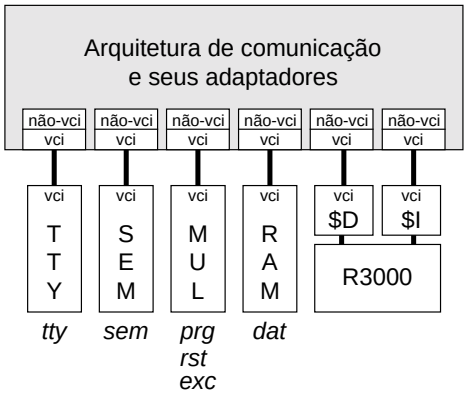


FIGURA 4.23 – Sistema integrado de referência para a avaliação de desempenho das arquiteturas de comunicação.

Os modelos de núcleos incluídos nesse sistema são os seguintes:

a) R3000 – é um modelo de processador MIPS sem unidade de ponto flutuante com interfaces para conexão a caches separadas para dados em instrução.

b) $D – cache de dado com interface iniciador-VCI.

c) $I – cache de instrução com interface iniciador-VCI.

d) RAM – é um modelo de memória RAM que é instanciado para a implementação de memórias destinadas ao armazenamento da zona de dados (*dat*) do espaço de endereçamento do MIPS R3000. Este modelo possui interface alvo-VCI.

e) MUL – é um modelo de memória RAM que suporta o mapeamento de três zonas de memória não contíguas. Ele é utilizado para o mapeamento das zonas de código de programa (*prg*), reinicialização (*rst*) e exceção (*dat*) do espaço de endereçamento do MIPS R3000. Este modelo possui interface alvo-VCI.

f) SEM – trata-se de um modelo especializado de memória RAM que implementa semáforos binários utilizados para a implementação de exclusão mútua no acesso a dados compartilhados (eg. canais de comunicação). Esses semáforos são mapeados em uma zona de endereços denominada *sem*. Este modelo possui interface alvo-VCI.

g) TTY – é um modelo utilizado para a impressão de mensagens em uma janela no monitor do computador e registro em um arquivo de saída. Sua zona de endereços é referenciada pelo termo *tty*. Este modelo possui interface alvo-VCI.

Mapa de endereços

Cada um dos núcleos com interface alvo-VCI é mapeado em uma ou mais zonas de endereços mostradas na Tabela 4.2. Todas as zonas possuem um tamanho de 1 Mbyte. Contudo, o núcleo TTY mapeia quatro zonas contíguas de 1 Mbyte, as quais são referenciadas por um único nome: *tty*.

TABELA 4.5 – Mapa de endereços dos núcleos do sistema.

Zona	Endereços
dat	0x10000000 – 0x100FFFFF
prg	0x00400000 – 0x004FFFFF
exc	0x80000000 – 0x800FFFFF
rst	0xBFC00000 – 0xBFCFFFFF
sem	0xB0000000 – 0xB00FFFFF
tty	0xA0000000 – 0xA03FFFFF

Na zona *dat* são mantidas a pilha e os dados do sistema operacional e da aplicação. Por exemplo, os canais utilizados para comunicação entre *threads* são estruturas de dados manipuladas pelo sistema operacional e alocadas nessa zona.

Benchmark

A aplicação escolhida como *benchmark* para a avaliação e comparação das redes acima descritas foi a Transformada Rápida de Fourier (FFT – *Fast Fourier Transform*). O algoritmo utilizado foi baseado naquele apresentado em [QUI 94]. Na versão implementada, a aplicação conta com *threads* (denominadas *fft*) que realizam o processamento da FFT em paralelo e uma *thread* adicional para colher o resultado final do processamento realizado pelas demais e imprimir esses resultados no terminal de saída, sincronizando o término da aplicação (denominada *sync*). O número de *threads fft* foi limitado em quatro, o qual corresponde ao número de processadores a utilizados na avaliação.

Na Figura 4.24, é mostrado o fluxo de dados para uma FFT de oito pontos processada através de quatro *threads fft* e uma *thread sync*. Conforme é ilustrado na figura, o algoritmo é executado em quatro estágios: (i) permutação de pontos entre as *threads fft*; (ii) iterações com processamento e sem comunicação; (iii) interações com processamento e comunicação; e (iv) sincronização. A primeira etapa pode ser eliminada se os pontos forem lidos pelas *threads fft* na ordem permutada.

Durante o estágio de processamento com comunicação, a cada iteração, as *threads fft* devem trocar informações sobre todos os seus pontos com uma *thread fft* parceira. Isso é feito pela leitura e escrita em canais de comunicação cuja profundidade é proporcional ao tamanho do vetor de pontos a ser trocado (igual ao número de pontos por *thread*). Da mesma forma, ao término do estágio de processamento com comunicação, as *threads fft* enviam seus resultados finais para a *thread sync* escrevendo em canais de comunicação reservados para essa finalidade.

O código dessa versão da FFT paralela é escalável e o número de *threads fft* pode ser aumentado se o sistema integrado alvo permitir o aumento do paralelismo da execução. Ou seja, teoricamente, quanto maior for o número de processadores disponível no sistema, maior será a capacidade de processamento paralelo instalada e os pontos da FFT poderão ser distribuídos entre um número maior de *threads fft*. Contudo, esse número deverá escalar de forma que o número de *threads fft* seja uma potência de dois.

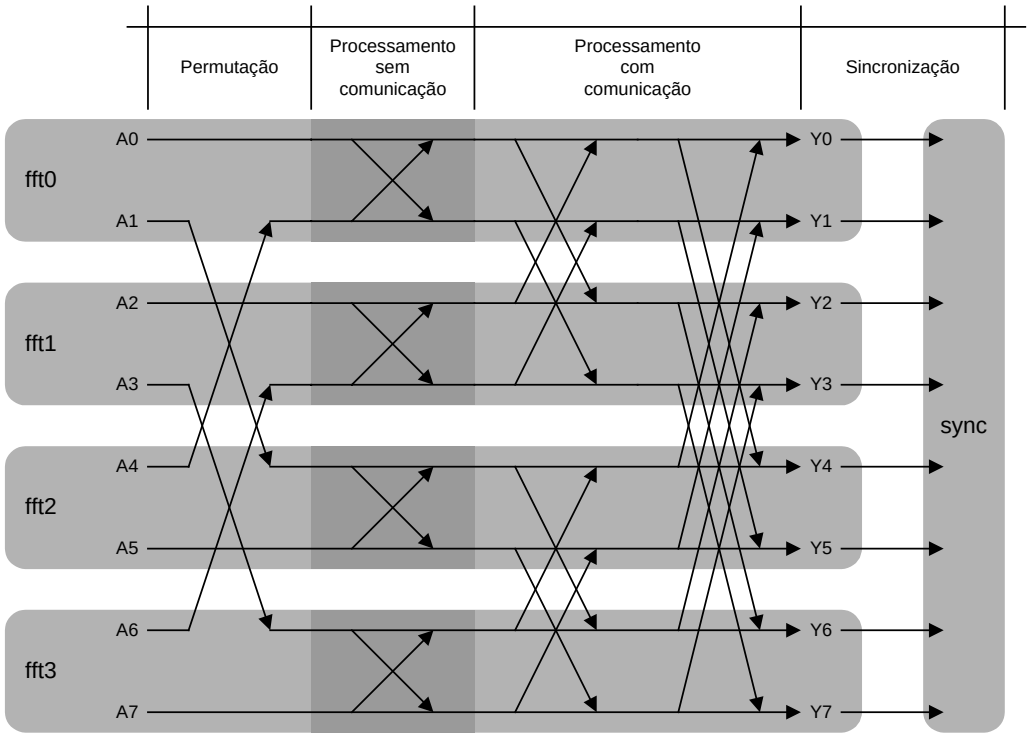


FIGURA 4.24 – Fluxo de dados em uma FFT com oito pontos processada por quatro *threads fft*.

Sistemas multiprocessados

Em sistemas multiprocessados, a vantagem do paralelismo da rede SPIN sobre o PI-Bus será verificada se, e somente se, os dados puderem ser distribuídos entre diferentes RAMs de dados. Caso contrário, apesar de a rede oferecer múltiplos caminhos paralelos, uma RAM de dados única iria seqüencializar as comunicações. Com esse objetivo, para cada *thread fft*, foi definida uma estrutura para manter os dados privativos de cada *thread* em uma RAM particular. Cada uma dessas memórias pode receber as estruturas de dados de uma ou mais *threads*, conforme o número de RAMs instanciadas. Essa alocação é feita estaticamente através de ponteiros que apontam para endereços sabidamente não utilizados pelo sistema operacional dentro da zona de dados de `0x100A0000` a `0x100DFFFF`.

Em um sistema com quatro processadores (Figura 4.25), cada *thread fft* deve ser carregada pelo sistema operacional em um processador diferente. Havendo cinco RAMs de dados, a estrutura de dados de cada *thread* é colocada em uma RAM exclusiva e a quinta memória é utilizada para manter os dados do sistema operacional e a pilha. O objetivo dessa distribuição manual das estruturas de dados das *threads* em diferentes memórias seria contornar uma característica do sistema operacional que refere-se à alocação de todos os dados de todas as *threads* em uma mesma zona de endereços, a qual é mapeada em única memória de dados. Contudo, conforme será apresentado abaixo, essa solução mostrou-se infrutífera.

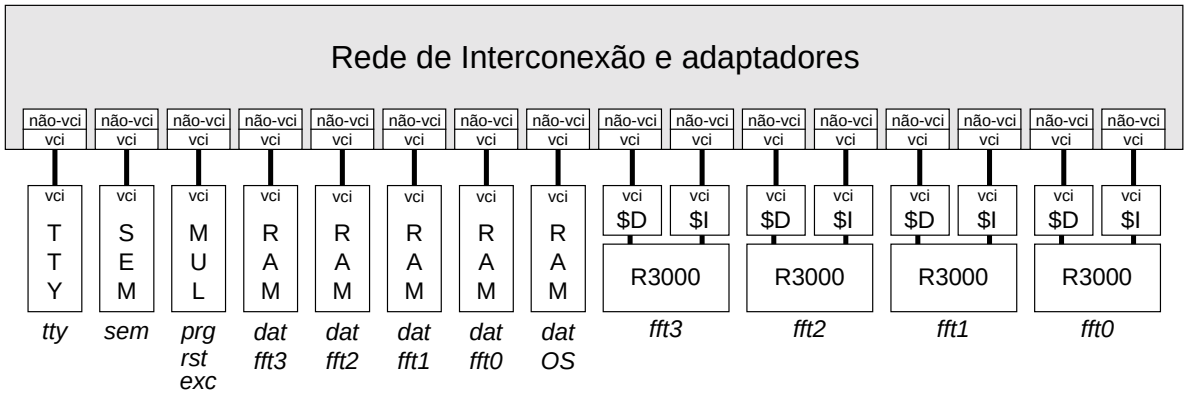


FIGURA 4.25 – Sistema multiprocessado para a execução da FFT.

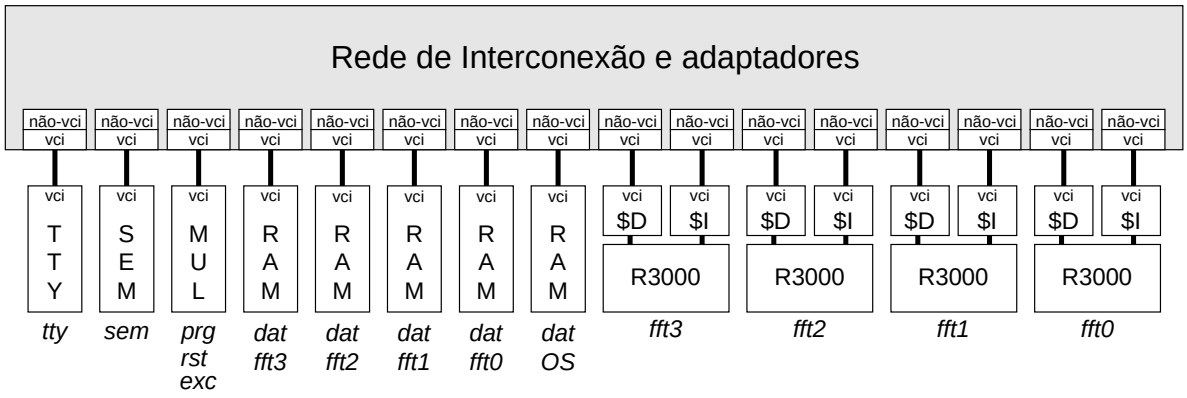


FIGURA 4.25 – Sistema multiprocessado para a execução da FFT.

Limitações da plataforma de simulação

Considerando-se como plataforma de simulação o conjunto formado pelo simulador CASS, pelo sistema operacional Mutek e pelas bibliotecas de componentes para a modelagem de sistemas integrados (processadores, memórias, adaptadores e arquiteturas de comunicação), foram identificadas três limitações que impedem a avaliação adequada do desempenho da rede SPIN quando aplicada à modelagem de sistemas integrados multiprocessados como o descrito na Figura 4.25.

A primeira limitação refere-se à ausência de mecanismos de hardware ou de software que suportem a coerência de cache para os dados compartilhados entre as *threads*, problema que não ocorre com os modelos orientados ao barramento PI-Bus. Para contornar essa limitação e tentar obter uma carga de comunicação semelhante para as duas arquiteturas, nas avaliações realizadas, a cache de dados foi desabilitada para ambas as arquiteturas.

A segunda limitação decorre de duas características do simulador. Primeiramente, o sistema operacional Mutek foi desenvolvido visando sistemas baseados no barramento PI-Bus e todos os dados das aplicações e do sistema são alocados em uma mesma zona de endereços mapeada para uma mesma memória RAM (denominada *dat OS* na Figura 4.25). Em segundo lugar, o sistema operacional não

oferece primitivas que propiciem a alocação dos dados de cada *thread* em memórias físicas diferentes. A solução apresentada anteriormente baseia-se na organização dos dados de uma *thread* em uma estrutura alocada para uma zona de memória desejada. O procedimento utilizado consiste em criar um ponteiro para essa estrutura e fazer esse ponteiro apontar para uma zona de dados desejada. Contudo, embora se tenha controle sobre o mapeamento dos dados da *thread*, o ponteiro para a estrutura de dados continua sendo alocado e mapeado pelo sistema operacional na memória *dat OS*. Essa memória será acessada por todas as *threads* que precisarem acessar os dados de suas estruturas mapeadas em memórias diferentes e se tornará um gargalo para o sistema.

A terceira limitação está fortemente associada à limitação descrita no parágrafo acima. As comunicações entre as *threads* são também realizadas por meio do acesso a estruturas de dados alocadas e mapeadas pelo sistema operacional (os canais lógicos de comunicação). Essas estruturas são todas mapeadas para a memória *dat OS* e todas as comunicações entre as *threads* devem passar por ela. Dessa forma, embora a rede ofereça largura de banda e paralelismo superiores ao barramento, o fato de sua latência ser maior e de haver um ponto de estrangulamento no sistema leva a rede SPIN a ter um desempenho inferior ao do barramento.

Em resumo, a modelagem dos sistemas multiprocessados orientados ao barramento PI-Bus e à rede SPIN, bem como o desenvolvimento do *benchmark* paralelo, foram atividades que consumiram tempo e esforço e cujos resultados não atenderam às expectativas. Contudo, permitiram a obtenção de uma experiência na modelagem e avaliação de sistemas reais e na identificação das limitações do ambiente de simulação. Assim, destaca-se que o ambiente CASS/Mutek requer a implementação de suporte à coerência de cache à alocação e mapeamento dos dados da *threads* e dos canais de comunicação em memórias distribuídas para que uma solução multiprocessada baseada na rede SPIN tenha um desempenho efetivo.

Resultados experimentais

Os resultados mostrados a seguir ilustram as questões destacadas acima. São apresentadas curvas que indicam o número de ciclos gastos na execução da FFT utilizando-se quatro *threads* mapeadas no sistema multiprocessado da Figura 4.25 com quatro processadores (com caches de dado desabilitadas). Cada curva mostra o tempo gasto na solução da FFT para diferentes quantidades de pontos (4, 8, 16, 32 e 64). Conforme pode ser observado, o desempenho da rede SPIN é bastante inferior ao do barramento PI-Bus, o qual apresenta uma latência significativamente menor que a da rede SPIN.

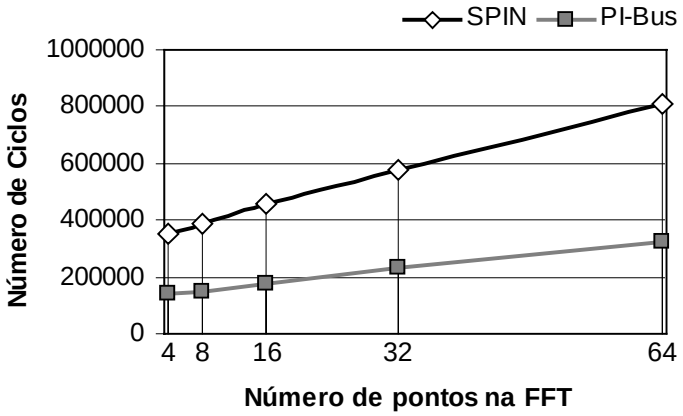


FIGURA 4.26 – Número de ciclos para a execução da FFT com diferentes números de pontos no barramento PI-Bus e na rede SPIN.

4.6 Considerações

Este capítulo apresentou as atividades e os resultados associados ao estágio sanduíche realizado no departamento ASIM do LIP6 sob a orientação do professor Alain Greiner. Como foi destacado, o estágio focou o estudo de conceitos associados à avaliação de arquiteturas de comunicação para sistemas integrados, incluindo o simulador CASS, o padrão VCI, o barramento PI-Bus e a própria rede SPIN. Desse estudo, desenvolveu-se um modelo CASS do roteador RSPIN, assim como sistemas integrados baseados no barramento PI-Bus e na rede SPIN e um *benchmark* paralelo para a avaliação de sistemas reais. Os modelos foram aplicados na avaliação do desempenho dessas arquiteturas.

Os experimentos baseados em geradores de tráfego demonstraram a efetividade da rede SPIN. Foi possível observar que a rede SPIN se mostra superior em desempenho ao barramento PI-Bus em sistemas com mais de oitos núcleos, considerando-se a carga de comunicação modelada. Para sistemas com 32 núcleos, verificou-se que a rede SPIN se apresenta como a única das duas arquiteturas a sustentar uma latência média aceitável (até 100 ciclos) para valores de carga oferecida superiores a 6% (ou até menores, dependendo do tamanho dos pacotes). Esses resultados foram divulgados em artigo publicado no *Designer's Forum* da conferência DATE'2003 [AND 2003]. Já os experimentos baseados em modelos de sistemas reais não atenderam as expectativas devido a limitações no simulador CASS. Contudo, acredita-se que caso não houvessem tais limitações, os resultados confirmariam o melhor desempenho da rede-em-chip relação ao barramento.

No período de estágio realizado no departamento ASIM foi possível verificar-se a efetividade da metodologia de modelagem e avaliação de arquiteturas de comunicação para sistemas integrados utilizando-se o simulador CASS. Como destacado acima, o simulador mostrou-se efetivo na avaliação de sistemas sintéticos, mas foi bastante restritivo quando da avaliação de modelos de sistemas reais. Acredita-se que as limitações apresentadas são passíveis de serem contornadas pela inclusão de chamadas de função no sistema operacional Mutek que permitam a alocação e o mapeamento de

dados privativos das *threads* e dos canais de comunicação em diferentes memórias distribuídas do sistema.

Durante o processo de estudo e modelagem do roteador RSPIN foi possível colaborar com os pesquisadores do Projeto SPIN no sentido de identificar soluções arquiteturais para o roteador, como, por exemplo, a definição da arquitetura dos árbitros do roteador, e também auxiliar na depuração de modelos de componentes do simulador CASS (eg. processador e adaptadores de comunicação). Essa interação foi mantida quando do retorno ao Brasil, uma vez que as atividades de experimentação e composição do artigo publicado no *Designer's Fórum* do DATE'2003 foram conduzidas após a conclusão do estágio.

A experiência obtida com esses estudos foi fundamental para o desenvolvimento dos projetos de pesquisa realizados posteriormente, principalmente na definição de uma arquitetura alternativa de rede-em-chip com vistas à síntese de redes de menor custo. Essa arquitetura de rede, denominada SoCIN (*System-on-Chip Interconnection Network*) é descrita no capítulo a seguir.

5 Uma Arquitetura de Rede-em-Chip para Sistemas Integrados

Este capítulo apresenta a arquitetura de uma rede-em-chip denominada SoCIN (*System-on-Chip, Interconnection Network*), a qual possui topologia direta e pode ser configurada como uma grelha 2-D ou um toróide 2-D (ou um toróide dobrado). Ela se baseia em controle de fluxo do tipo *handshake*, roteamento do tipo fonte e determinístico, chaveamento por pacotes do tipo *wormhole*, arbitragem dinâmica distribuída e memorização de entrada.

O bloco construtivo básico da rede SoCIN é o roteador RASoC (*Router Architecture for Systems-on-Chip*) e trata-se de um *soft-core* VHDL parametrizável em três dimensões: largura dos canais de comunicação, profundidade dos *buffers* e largura da informação de roteamento no cabeçalho do pacote. Além disso, o roteador RASoC pode ser configurado quanto ao número de portas a serem sintetizadas. Contudo, essa parametrização deve ser feita no momento da modelagem da rede, aterrando-se as entradas das portas não utilizadas e deixando em aberto as saídas dessas portas. Essas características oferecem um nível de customização à rede que permite que a mesma seja dirigida a diferentes aplicações.

As alternativas arquiteturais utilizadas no roteador RASoC buscam atender a diferentes objetivos, algumas estão entre as de menor custo, enquanto que outras estão entre as de melhor desempenho. A exploração do espaço de projeto foi limitada, pois entende-se que, em um primeiro momento, é mais importante dispor de um modelo validado que estabeleça as bases para tal exploração. A estrutura do roteador é distribuída e modularizada, sendo que cada módulo possui uma interface que permite o reuso do bloco na implementação de outras alternativas arquiteturais.

Devido à limitação de acesso a tecnologias para a validação do modelo e obtenção de resultados de síntese, a modelagem do roteador RASoC foi orientada a dispositivos lógicos programáveis de alta densidade, também denominados CPLDs (*Complex Programmable Logic Devices*) ou FPGAs (*Field Programmable Gate Arrays*). Contudo, o modelo pode ser facilmente portado para a síntese em outras tecnologias.

Neste capítulo, são apresentados detalhes e discussões a respeito da escolhas de projeto para a rede (topologia, enlaces, modelo de comunicação nativo, chaveamento, rotamento, formato do pacote, memorização, controle de fluxo e arbitragem) e uma visão geral da organização do roteador RASoC (os detalhes da organização e da implementação do modelo VHDL são apresentados no Apêndice 1). Este capítulo também apresenta alguns dos principais diagramas de formas de onda utilizados na validação do modelo, assim como resultados da síntese do roteador e de redes de pequena escala em FPGA.

5.1 Topologia

A rede SoCIN utiliza como topologia topologia de referência a grelha 2-D, a qual é mostrada na Figura 5.1. Na figura, as caixas representam roteadores e os arcos entre os roteadores representam enlaces de comunicação. O arco no canto superior esquerdo de cada caixa representa um enlace terminal para conexão de um núcleo à rede. Do ponto de vista do roteador, esse enlace terminal é denominado porta local.

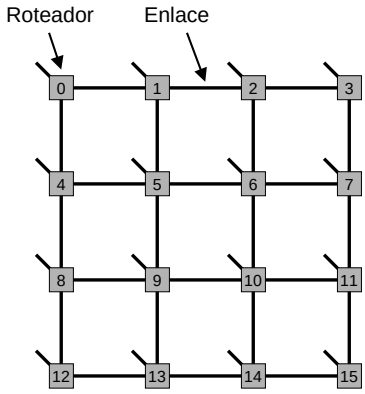
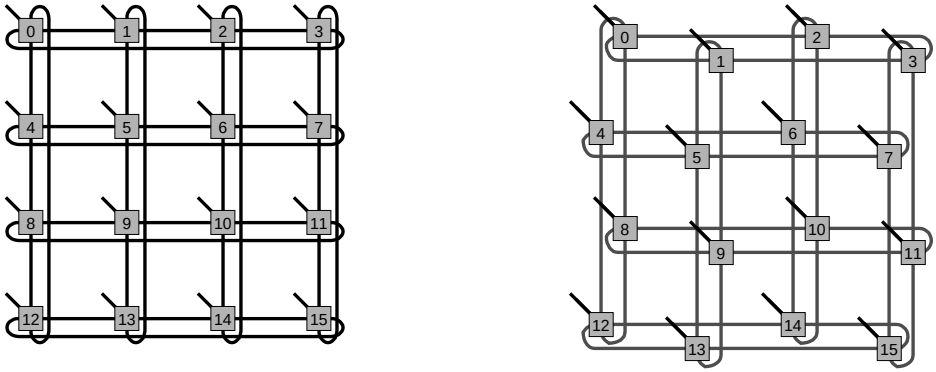


FIGURA 5.1 – Topologia em grelha 2-D utilizada na rede SoCIN.

A escolha dessa topologia deve-se ao fato de que as estruturas 2-D constituem-se nas mais fáceis de se implementar com as tecnologias de fabricação correntes [TAY 99], pela possibilidade de crescimento da rede sem modificações aos roteadores e pela capacidade dessa topologia em explorar a localidade espacial das aplicações, oferecendo, juntamente com a árvore-gorda, as melhores oportunidades de ganho de desempenho para um conjunto maior de aplicações [BAR 99].

Uma característica da grelha 2-D é que ela pode ser facilmente estendida para um toróide 2-D (Figura 5.2.a) pela inclusão de enlaces interligando os roteadores da periferia da rede. O toróide tem a vantagem de reduzir o diâmetro da rede, um parâmetro determinado pelo caminho mais curto entre os nodos mais distantes na rede [HWA 93]. Contudo, essa topologia possui um conjunto de enlaces mais longos que os da grelha, sendo que o maior comprimento desses canais resulta em uma carga capacitiva maior e uma freqüência de operação menor. No que tange a topologia, essa limitação pode ser amenizada utilizando-se variantes como o toróide dobrado ilustrado na Figura 5.2.b. Nessa topologia, os enlaces mais longos são menores que os correspondentes no toróide convencional, o que resulta em uma carga capacitiva menor e uma freqüência de operação maior. Na figura, os enlaces entre os roteadores são representados em um tom mais claro para uma maior legibilidade.



(a) (b)

FIGURA 5.2 – Topologias alternativas para a rede SoCIN:
(a) toróide 2-D; (b) toróide dobrado 2-D.

As topologias ilustradas nas Figuras 5.1 e 5.2 possuem estrutura regular. Contudo, tipicamente, os núcleos de um sistema integrado têm dimensões diferentes e, para não desperdiçar área de silício, é preciso que a rede assuma uma estrutura irregular como, por exemplo, a da Figura 5.3. Um aspecto interessante que pode ser observado nessa figura é que o roteador 4 não possui nenhum núcleo conectado ao seu enlace terminal, mas ele é incluído devido a requisitos do algoritmo de roteamento. Por exemplo, se o algoritmo determinar que um pacote deve percorrer primeiramente os enlaces na direção X para só depois deslocar-se na direção Y (sem poder retomar um enlace da direção X), seria impossível o envio de um pacote do núcleo conectado ao roteador 3 para o núcleo conectado ao roteador 7, por exemplo.

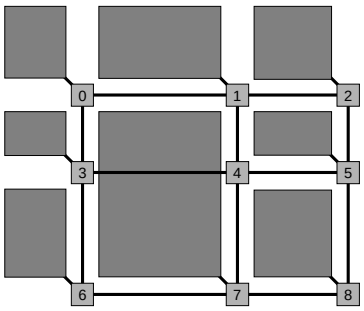


FIGURA 5.3 – Um sistema heterogêneo baseado em uma rede em grelha 2-D.

Em [ZHA 99], foi investigado o problema da interconexão de núcleos heterogêneos nos futuros sistemas integrados baseados em redes-em-chip. Esse estudo foi realizado no âmbito de um sistema com núcleos reconfiguráveis e, com base na heterogeneidade dos núcleos, os autores propuseram o uso de uma estrutura de interconexão baseada em uma grelha 2-D com leiaute irregular. Um problema que surge da irregularidade da rede é que seus canais são diferentes e precisam ser otimizados individualmente para atender aos requisitos de conectividade de cada núcleo [ZHA 99]. A principal limitação de um projeto heterogêneo é a sua falta de generalização para uma metodologia de projeto organizada. Uma alternativa para se garantir a regularidade da rede e ainda a latência da rede em grelha para longas distâncias consiste no uso de uma rede com estrutura hierárquica. Essa hierarquia é constituída por uma grelha 2-D regular no topo da hierarquia e por arquiteturas de comunicação locais (eg. em barrramento ou

em grelha) interconectando os núcleos organizados sob a forma de agregados (*clusters*), os quais são conectados aos terminais da rede do topo [TAY 99].

Consideradas as restrições para o acesso a tecnologias de fabricação de circuitos integrados, a rede SoCIN foi modelada e validada visando sua aplicação inicial em sistemas integrados sintetizados em dispositivos do tipo FPGA [ROS 93], os quais constituem-se em uma tecnologia acessível para a integração de sistemas digitais. Entretanto, é sabido que essa abordagem é limitada a sistemas experimentais e não é utilizada em sistemas integrados comerciais. De fato, os FPGAs constituem em partes integrantes de muitos sistemas integrados, mas são utilizados para a síntese de partes reconfiguráveis das aplicações [LIA 2000, KUM 2002], não para a síntese da arquitetura de comunicação. Por isso, o modelo da rede SoCIN foi planejado de modo a ser facilmente portado para outras tecnologias, além do FPGA.

5.2 Enlaces

Os enlaces da rede SoCIN são similares aos da rede SPIN [GUE 2000a] e são implementados por dois canais unidirecionais *simplex* (Figura 5.4). Cada canal *simplex* é constituído por n bits de dados e dois bits de banda lateral, sendo n+2 o valor da largura física do canal (phit). Os dois bits de banda lateral correspondem aos sinais usados para o enquadramento do pacote, os quais constituem os marcadores de início de pacote (*bop – begin-of-packet*) e fim de pacote (*eop – end-of-packet*). O parâmetro n deve ser definido em função dos requisitos do sistema e pode incluir, além dos sinais necessários para o transporte dos dados do pacote, sinais adicionais a serem manipulados por um protocolo de mais alto nível. Um exemplo seria a implementação de sinalizadores de paridade e de erro de paridade, como na rede SPIN [GUE 2000]. Tal protocolo seria transparente aos roteadores da rede, mas visível aos adaptadores de comunicação. Além dos n+2 bits, cada canal *simplex* inclui um par de sinais necessários ao controle de fluxo (*val* e *ack*), os quais não são contabilizados no cálculo do phit da rede, pois não atravessam os roteadores e não armazenados em seus *buffers*. O uso desses bits de enquadramento do pacote e de controle de fluxo será melhor explicado em seções que seguem.

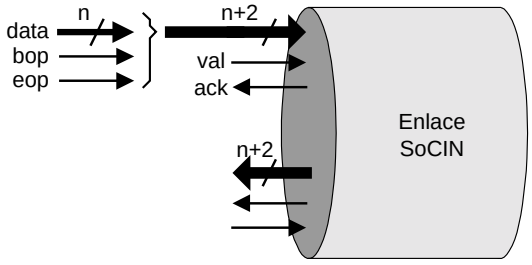


FIGURA 5.4 – Enlace da rede SoCIN.

5.3 Modelo de Comunicação Nativo

O modelo de comunicação nativo da rede SoCIN é o de troca de mensagens. Em um sistema integrado baseado na rede SoCIN, os núcleos comunicam-se entre si enviando e recebendo pacotes de requisição e de resposta. Os núcleos com capacidade de iniciar uma comunicação e enviar um pacote de requisição são denominados iniciadores-SoCIN (eg. processador), enquanto que os que devem responder às requisições enviando pacotes de resposta são denominados alvos-SoCIN (eg. memórias). Essa terminologia é derivada daquela utilizada no padrão VCI [VSI 2000].

A maioria dos núcleos para sistemas integrados baseia-se em modelos de comunicação diferentes do modelo nativo da rede SoCIN, utilizando, tipicamente, o modelo de comunicação de fluxo de dados (*dataflow*) ou modelo do espaço de endereçamento [GUE 2000a]. Na verdade, é sabido que os futuros sistemas integrados irão incluir um conjunto diversificado de tarefas heterogêneas [KUM 2002], integrando núcleos orientados a um ou a outro modelo de comunicação. Dessa forma, padrões de tráfego com comportamentos dinâmico e estático irão coexistir em um mesmo sistema.

Qualquer rede-em-chip que se proponha ser reutilizável para diferentes domínios de aplicação deve ser capaz de suportar os modelos de comunicação utilizados por essas aplicações. Esse suporte pode ser dado através da implementação de adaptadores de comunicação, os quais têm a função de realizar a adaptação entre os modelos de comunicação dos núcleos e da rede. Alternativamente, os núcleos podem ser implementados incluindo uma interface compatível com o protocolo da rede. Contudo, é mais provável que os núcleos a serem integrados em um sistema não ofereçam tal compatibilidade, exigindo o uso de adaptadores.

Na rede SoCIN, os adaptadores de comunicação devem ter um número de identificação exclusivo correspondente à posição do roteador ao qual está conectado, e uma tabela de roteamento a ser utilizada para determinar a rota a ser tomada por cada pacote enviado de um núcleo a outro. Após consultar a tabela, o adaptador constrói o cabeçalho do pacote, nele incluindo a informação de roteamento correspondente ao caminho a ser tomado pelo pacote e a própria identificação do adaptador. Essa identificação é necessária para que o adaptador do núcleo alvo da comunicação possa determinar o emissor do pacote de requisição e estabelecer a rota a ser tomada pelo pacote de resposta.

Na Figura 5.5, é ilustrado um exemplo de sistema baseado em uma rede SoCIN com topologia em grelha 2×2. Nesse sistema existem dois iniciadores-SoCIN e dois alvos-SoCIN (caixas escuras) conectados à rede através de adaptadores de comunicação (caixas claras). Na figura, assume-se que os adaptadores dos iniciadores implementam tabelas de roteamento para os alvos e vice-versa.

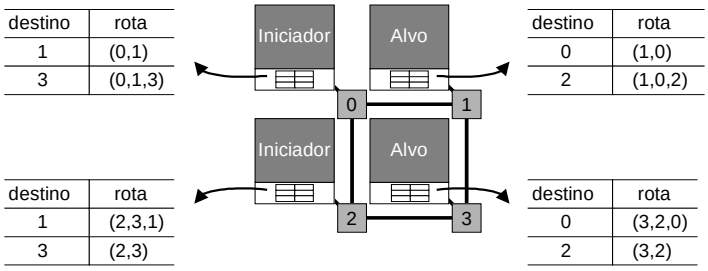
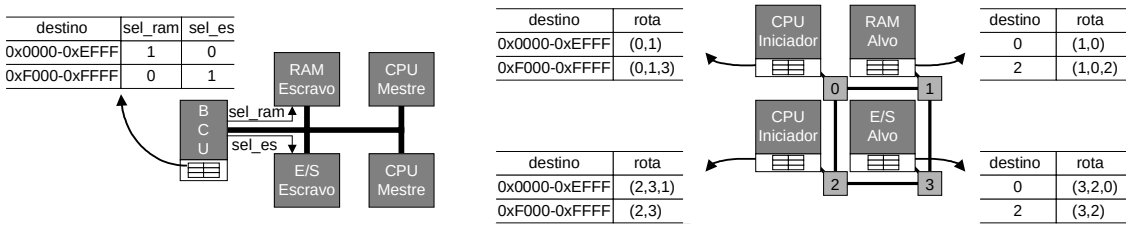


FIGURA 5.5 – Um sistema genérico baseado na rede SoCIN.

A adaptação dos diferentes modelos de comunicação ao modelo nativo da rede SoCIN consiste na implementação dos circuitos de conversão dos protocolos e das interfaces, bem como a configuração adequada das tabelas de roteamento. Por exemplo, na Figura 5.6.a é mostrado um sistema integrado multiprocessado baseado no modelo de comunicação de espaço de endereçamento. Ele é composto por dois núcleos de processamento (CPUs – *Central Processing Units*), um controlador de E/S e uma quantidade de memória RAM interconectados por um barramento controlado pelo BCU (BCU – *Bus Controller Unit*). O BCU possui uma tabela de mapeamento utilizada na decodificação de endereços para a seleção do escravo alvo de cada comunicação. A implementação de um sistema semelhante baseado na rede SoCIN exigiria o uso de adaptadores de comunicação cujas tabelas de roteamento dos adaptadores incluíssem o mapeamento necessário para o estabelecimento das rotas a serem incluídas em cada pacote (Figura 5.6.b). Nos adaptadores dos iniciadores-SoCIN, as tabelas efetuam o mapeamento de zonas de endereços do espaço de endereçamento de memória para rotas a serem tomadas pelos pacotes de requisição (no exemplo, assume-se que a E/S é mapeada no espaço de memória). Já nos adaptadores dos alvos-SoCIN, as tabelas realizam o mapeamento entre endereços de rede e as rotas a serem tomadas pelos pacotes de resposta.



(a) (b)

FIGURA 5.6 – Exemplo de um SoC multiprocessado: (a) baseado em um barramento; (b) baseado na rede SoCIN.

Procedimento análogo ao mostrado na Figura 5.6 deve ser utilizado para a interconexão de núcleos em aplicações baseadas no modelo de comunicação de fluxo de dados.

5.4 Chaveamento

A rede SoCIN utiliza o chaveamento por pacotes do tipo *wormhole*. Os pacotes são constituídos por flits sendo que cada flit possui o tamanho de um phit, o qual é igual a *n*+2 bits. O primeiro flit do pacote constitui o seu cabeçalho e inclui as informações necessárias ao estabelecimento do caminho do pacote na rede. Os demais flits incluem a informação a ser transferida pelo pacote, sendo que eles seguem o flit de cabeçalho pela rede em um modo *pipeline*. Cada roteador da rede tem capacidade para armazenar poucos flits de um pacote bloqueado. Assim, quando o cabeçalho de um pacote chega a um roteador e a porta de saída necessária ao seu encaminhamento não está disponível, o roteador absorve os flits possíveis de serem armazenados no espaço disponível em seu *buffer* e os demais flits são mantidos nos *buffers* dos roteadores anteriores no caminho do pacote.

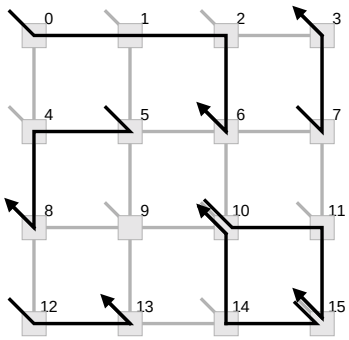
5.5 Roteamento

A rede SoCIN utiliza roteamento baseado no ordenamento por dimensão [DUA 97]. Com base na classificação das técnicas de roteamento apresentada na seção 2.4, essa técnica pode ser classificada como dinâmica, *unicast*, fonte, baseada em tabela e determinística. Em outras palavras, as rotas a serem utilizadas pelos pacotes são determinadas em tempo de execução, sendo que cada pacote pode ter apenas um destinatário e a rota a ser seguida é definida pelo emissor do pacote por meio da consulta a uma tabela de roteamento. Por ser determinístico, os pacotes transferidos de um dado emissor para um dado receptor utilizam sempre o mesmo caminho, sem qualquer capacidade de adaptação às falhas ou ao tráfego na rede.

As técnicas de roteamento baseadas no ordenamento por dimensão são muito utilizadas em redes de interconexão com topologias dos tipos grelha, toróide e hipercubo *n*-dimensionais. Elas recebem diferentes denominações, como roteamento XY para as redes 2-D ou *e-cube* para as redes hipercúbicas [DUA 97]. A grande vantagem do roteamento baseado no ordenamento por dimensão é que ele garante a liberdade de *deadlock* a um baixo custo de implementação, porém, reduz a utilização da rede, conforme será explicado posteriormente.

No roteamento XY utilizado na rede SoCIN, um pacote deve percorrer totalmente uma linha na direção X até chegar à coluna na qual se situa o destinatário. Após isso, ele deve percorrer essa coluna até atingir o roteador ao qual o núcleo destinatário está conectado, quando então ele é entregue a esse núcleo. Uma vez que um pacote toma a direção Y, ele não pode mais tomar a direção X. Se os núcleos fonte e destinatário estiverem na mesma linha, o pacote não precisa ser roteado na direção Y. De maneira análoga, se eles estiverem na mesma coluna, o pacote não precisa ser roteado na direção X.

Na Figura 5.7, são apresentados alguns exemplos de rotas permitidas pelo roteamento XY em uma rede com topologia em grelha 2-D (a tabela ao lado lista a rotas ilustradas na figura). Destaca-se a comunicação entre os núcleos conectados aos roteadores 10 e 15. Como pode ser observado, o caminho tomado pelos pacotes emitidos por cada um dos dois núcleos é diferente, pois a direção Y só pode ser utilizada após serem esgotados os deslocamentos na direção X.



Origem	Destino	Rota
0	6	0-1-2-6
5	8	5-4-8
7	3	7-3
12	16	12-13
10	15	10-11-15
15	10	15-14-10

FIGURA 5.7 – Exemplos de rotas permitidas no roteamento XY.

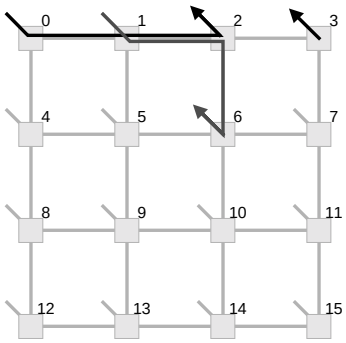
Conforme foi discutido na subseção 2.3.3, o *deadlock* surge quando ocorre uma dependência cíclica entre quatro pacotes na rede. Uma dependência cíclica pode surgir em uma grelha 2-D quando o algoritmo de roteamento permitir a ocorrência de qualquer um dos dois ciclos abstratos ilustrados na Figura 5.8.a. Cada um desses ciclos é constituído por quatro voltas (ou curvas) e se o roteamento permitir que todas essas voltas sejam realizadas, então esses ciclos poderão ocorrer e provocar dependências cíclicas, levando a rede ao estado de *deadlock*.

(a) (b)

FIGURA 5.8 – Dependências cíclicas: (a) ciclos abstratos (b) voltas permitidas no roteamento XY [DUA 97].

O roteamento XY garante a liberdade de *deadlock* proibindo a ocorrência das voltas “Y-para-X”, ilustradas por linhas tracejadas na Figura 5.8.b. Como somente são permitidas as voltas do tipo X-para-Y, ilustradas por linhas contínuas na mesma figura, os ciclos abstratos da Figura 5.8.a jamais irão ocorrer e a rede nunca entrará em *deadlock*. Contudo, ao limitar os tipos de volta permitidos em uma rota, o roteamento também limita as rotas possíveis de serem utilizadas entre os núcleos conectados à rede. Essa questão é ilustrada na Figura 5.9. Dada uma grelha 4∝4 com roteadores identificados pelos números de 0 a 15, se, por exemplo, o núcleo conectado ao roteador 0 estiver enviando um pacote longo (que ocupe muitos canais simultaneamente) ao núcleo conectado ao roteador 2, os canais e os *buffers* associados à rota 0-1-2 estarão reservados para essa comunicação até que o terminador do pacote avance por esse caminho liberando os recursos alocados. Se, durante o envio desse pacote, o núcleo conectado ao roteador 1 tiver um pacote para ser enviado ao núcleo conectado ao roteador 6, ele terá que esperar a liberação do canal 1-2 pelo pacote em curso para poder

avançar ao seu destinatário utilizando a rota 1-2-5. Se não houvesse a restrição de voltas do tipo “Y-para-X”, a rota 1-5-6 poderia ser utilizada, reduzindo a contenção na rede.



Origem	Destino	Rota
0	2	0-1-2
1	6	1-2-6

FIGURA 5.9 – Exemplo de limitação do uso dos canais da rede com roteamento XY.

Em resumo, a garantia da ausência de *deadlock* por restrição de voltas tem um preço a ser pago: a redução do número de rotas possíveis e, portanto, a redução da taxa de utilização da rede e da largura de banda agregada disponível. Contudo, existem alternativas de algoritmos de roteamento livres de *deadlock* menos restritivos com relação aos tipos de volta permitidos. Por exemplo, o modelo de volta (*turn model*), proposto por Glass e Ni [GLA 92], proíbe apenas uma volta em cada ciclo abstrato, propiciando diferentes algoritmos de roteamento parcialmente adaptativos e com uma melhor taxa de utilização da rede. Já os algoritmos de roteamento totalmente adaptativos oferecem uma taxa de utilização da rede ainda maior. Contudo, a garantia da liberdade de *deadlock* nesses algoritmos é mais onerosa, especialmente em redes com chaveamento por pacotes do tipo *wormhole*, nas quais a solução consiste no uso de controle de fluxo baseado em canais virtuais [DAL 87]. Estudos apresentados em [VAI 2001] mostram que o uso de roteamento adaptativo e canais virtuais em redes de interconexão para computadores paralelos só é justificável se a aplicação apresentar uma demanda de largura de banda de comunicação que compense o custo adicional para a implementação da rede. No nível das redes-em-chip, as restrições de custo podem ser ainda maiores conforme a quantidade de silício disponibilizada para a implementação da rede. Por isso, das redes-em-chip conhecidas que utilizam topologia em grelha e chaveamento por pacotes do tipo *wormhole* [KUM 2002], o roteamento XY tem sido a opção preferencial, pois permite a construção de roteadores de baixo custo e de redes livres de *deadlock*.

5.5.1 Implementação do roteamento XY na rede SoCIN

O roteamento XY implementado na rede SoCIN é do tipo fonte e baseado em endereçamento relativo. Em outras palavras, o adaptador de comunicação do núcleo emissor do pacote consulta sua tabela de roteamento e obtém a informação que define a quantidade de deslocamentos a serem realizados nas direções X e Y para o pacote chegar no seu destinatário (ou seja, o número de enlaces em cada direção). Na medida em que o cabeçalho do pacote avança pela rede, de roteador em roteador, essa informação deve ser atualizada no cabeçalho para refletir o número de deslocamentos que restam ser realizados. A informação de roteamento é utilizada em cada roteador para determinar a porta de saída a ser requisitada para o encaminhamento do pacote. Enquanto o valor do deslocamento em X for não nulo, o pacote é encaminhado na direção X. À proporção em que ele avança nessa direção, esse valor é decrementado até

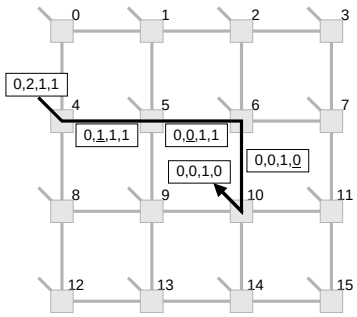
chegar a 0. No roteador seguinte, se o valor do deslocamento em Y for não nulo, o pacote é roteado nessa direção. Ao passo em que cabeçalho do pacote avança pela rede na direção Y, esse valor é decrementado até chegar a 0. Quando um roteador recebe um cabeçalho de pacote com os valores de deslocamento em X e em Y nulos, o pacote é entregue ao núcleo conectado à porta local do roteador.

A informação de roteamento é constituída por quatro campos que definem a quantidade de deslocamentos a serem dados nas direções X e Y, assim como o sentido a ser tomado. Esses campos são denominados *Xmod*, *Xdir*, *Ymod* e *Ydir*, sendo que *Xmod* e *Ymod* informam o número de enlaces a serem percorridos pelo pacote em X e em Y, respectivamente. Já os campos *Xdir* e *Ydir* definem se o pacote deve ser dirigido nos sentidos leste ou oeste, para a direção X, e norte ou sul para a direção Y, conforme mostra a Tabela 5.1.

TABELA 5.1 – Campos constituintes da informação de roteamento.

Campo	**Tipo**	**Significado**
Xmod	Natural	Quantidade de deslocamentos em X (número de enlaces em X)
Xdir	Bit	Sentido do deslocamento em X: 0 = leste, 1 = oeste
Ymod	Natural	Quantidade de deslocamentos em Y (número de enlaces em Y)
Ydir	Bit	Sentido do deslocamento em Y: 0 = norte, 1 = sul

Na Figura 5.10, é exemplificada a transferência do cabeçalho de um pacote pela rede, destacando-se a atualização da informação de roteamento em cada roteador. Um pacote destinado ao núcleo conectado ao roteador 10 é injetado na rede pelo adaptador de comunicação do núcleo conectado ao roteador 4. Como pode ser observado, a informação de roteamento no momento da injeção do pacote na rede é igual “0,2,1,1” e indica que o pacote deve percorrer dois enlaces em direção ao leste e um enlace em direção ao sul. O exemplo é explicado a seguir.



Origem	**Destino**	**Xdir,Ymod,Ydir,Ymod**
4	10	0,2,1,1

FIGURA 5.10 – Exemplo de roteamento de um pacote na rede SoCIN.

No exemplo da Figura 5.10, o roteador 4 avalia o campo *Xmod* e, após identificar que o mesmo é não nulo, avalia o campo *Xdir* para determinar se o pacote será encaminhado em direção ao oeste ou ao leste (única opção possível para esse roteador). Desde que *Xdir* é igual a 0, o pacote é encaminhado em direção ao leste e o campo *Xmod* é decrementado, passando a valer 1. Procedimento semelhante se repete no roteador 5, o qual atualiza o campo *Xmod* tornando-o nulo. O roteador 6, por sua vez, verifica que *Xmod* é igual a 0 e então avalia o campo *Ymod* para verificar se o mesmo é não nulo. No exemplo, como *Ymod* é igual a 1, o roteador avalia o campo *Ydir* para determinar se o pacote deve ser encaminhado em direção ao norte ou ao sul. Antes de

enviar o pacote pela saída sul (pois *Ydir* é igual a 1), o roteador 6 atualiza o campo *Ymod* decrementando-o para 0. O roteador 10 recebe esse pacote e, após determinar que tanto *Xmod* quando *Ymod* são nulos, entrega o pacote ao núcleo conectado à sua porta local. O algoritmo de roteamento XY baseado em endereçamento relativo é ilustrado no código C apresentado na Figura 5.11.

```
÷ Entradas:    módulo e direção em X e em Y (Xmod, Xdir, Ymod, Ydir)
÷ Saídas:      porta de saída selecionada (Sel)
÷ Constantes: portas de saída (West, East, North, South, Local)
```

```
int RoutingXY(int Xdir, int *Xmod, int Ydir, int *Ymod)
{
       int Sel;
       if (*Xmod != 0) {
              if (Xdir = 1)
                     Sel = West;
              else
                     Sel = East;
              *Xmod--;
       }
       else {
              if (*Ymod != 0) {
                     if (Ydir = 1)
                            Sel = South;
                     else
                            Sel = North;
                     *Ymod--;
              }
              else
                     Sel = Local;
       }
       return Sel;
}
```

FIGURA 5.11 – O algoritmo de roteamento XY baseado em endereçamento relativo.

Uma outra alternativa para a implementação do roteamento XY é baseada no endereçamento absoluto [DUA 97], no qual os roteadores possuem coordenadas XY e a informação de roteamento contém as coordenadas do destinatário. Cada roteador compara suas coordenadas com as do destinatário e determina a porta de saída a ser utilizada. Essa alternativa tem a vantagem de dispensar a atualização do cabeçalho e de ter uma informação de roteamento mais curta, pois não é necessário codificar os bits indicadores do sentido do deslocamento.

5.6 Formato do Pacote

No protocolo da rede SoCIN, o pacote é composto por um flit de cabeçalho e por um número irrestrito de flits que compõem a carga útil do pacote, sendo que o último flit da carga útil é também o terminador do pacote. Cada flit possui n+2 bits, dos quais n bits são reservados ao transporte de informação (ou dado) e dois bits são destinados ao enquadramento do pacote. Esses bits de enquadramento são denominados *bop* (*begin of-packet*) e *eop* (*end-of-packet*), correspondendo, respectivamente, aos marcadores de início (cabeçalho) e de fim (terminador) do pacote. Conforme pode ser observado na Figura 5.12, o cabeçalho é marcado pelo bit *bop* em 1, enquanto que o terminador é marcado pelo bit *eop* em 1.

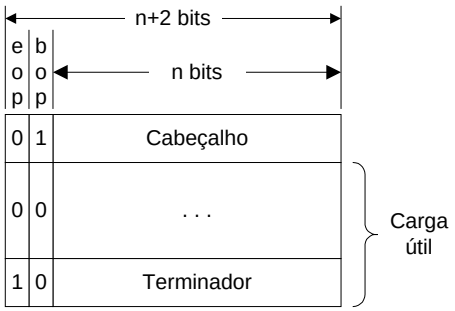
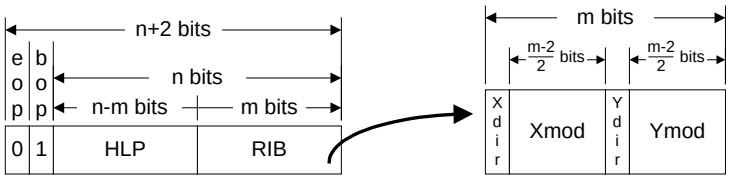


FIGURA 5.12 – Formato do pacote SoCIN.

No cabeçalho de um pacote SoCIN, *m* bits são reservados à informação de roteamento. Esse parâmetro pode ser configurado em função do tamanho da rede a ser sintetizada, sendo que *m* deve ser menor ou, no máximo, igual a *n*. Na terminologia da rede SoCIN, esses bits são denominados RIB (*Routing Information Bits*) e incluem os campos relacionados ao deslocamento em X e em Y. Dos *m* bits do RIB, (*m*–2)/2 são reservados ao campo *Xmod* e (*m*-2)/2 são reservados ao campo *Ymod*. Os outros dois bits são reservados aos campos *Xdir* e *Ydir*, conforme é ilustrado na Figura 5.13. Os demais bits do cabeçalho não são manipulados pelos roteadores e são reservados ao protocolo de alto nível (HLP – *High Level Protocol*) dos adaptadores de comunicação. Esses bits podem ser utilizados, por exemplo, para a identificação do emissor do pacote, identificação da *thread* que enviou o pacote em um processador *multithreaded* ou para implementar protocolos de detecção e sinalização de erro de paridade.



(a) (b)

FIGURA 5.13 – Formato do cabeçalho do pacote SoCIN: (a) HLP e RIB; e (b) campos do RIB.

O valor definido para *m* limita o tamanho da rede pois o número máximo de deslocamentos em cada direção é igual a:

$$Xmod_{max} = Ymod_{max} = 2^{(m/2-1)} - 1 \qquad (5.1)$$

Já a dimensão máxima da rede será $k_{max} \backsim k_{max}$, onde:

$$k_{max} = 2^{(m-2)/2} \qquad (5.2)$$

A partir de (5.1) e (5.2), se *m* for igual a 10, por exemplo, cada campo de módulo terá 4 bits com valor máximo igual a 15. A dimensão máxima de uma rede com topologia em grelha 2-D será 16⇜16, com 15 enlaces em cada direção. Essa parametrização da largura da parte de dados do pacote e da informação de roteamento (parâmetros *n* e *m*) permite a síntese de redes com custo mais adequado aos requisitos do sistema e, ao mesmo tempo, oferece escalabilidade ao modelo do roteador.

5.7 Memorização

A rede SoCIN utiliza memorização na entrada, abordagem na qual os espaço de memória para o armazenamento de flits de pacotes em cada roteador é distribuído sob a forma de partições entre os canais de entrada do roteador. Essas partições são implementadas sob a forma de *buffers* FIFO (*First-In, First-Out*) e os flits são lidos das partições na mesma ordem em que são escritos. Essa abordagem apresenta como limitação o problema do bloqueio de cabeça de linha. Contudo, é a mais econômica para a implementação de roteadores de baixo custo. Existem soluções para amenizar esse problema, as quais são baseadas na implementação de múltiplos *buffers* FIFO em cada canal de entrada (tipicamente, um para cada saída possível de ser utilizada), mas essas soluções são mais caras, pois, além dos múltiplos *buffers*, requerem *crossbars* de maior dimensão [TAM 92].

Nos roteadores da rede SoCIN, cada entrada possui um *buffer* FIFO com capacidade de armazenar p flits de n+2 bits, sendo p um parâmetro ajustado em função dos requisitos do sistema. Quanto maior é o seu valor, mais profundos são os *buffers* e menor é a contenção na rede, pois os pacotes tendem a ocupar menos canais quando são bloqueados em algum ponto da rede. Contudo, o custo dos roteadores aumenta com p, sendo que, tipicamente, em redes de interconexão baseadas em chaveamento *wormhole*, p é igual a quatro [DUA 97].

Como o roteador da rede SoCIN têm até cinco portas de entrada, conforme a topologia implementada e a posição do roteador na rede, o número máximo de bits de memória por roteador na rede SoCIN é dado por:

$$Mem_{max/rot} = 5 \propto p \propto (n+2) \quad \text{(bits)} \tag{5.3}$$

5.8 Controle de Fluxo

A técnica de controle de fluxo utilizada na rede SoCIN é baseada no protocolo de *handshake*, no qual o emissor informa a intenção de enviar um dado ao receptor através de uma linha de validação e o receptor confirma a disponibilidade de espaço em *buffer* para receber esse dado através de uma linha de reconhecimento (*acknowledgement*). De fato esse sinal de reconhecimento sinaliza ao emissor que o flit por ele injetado no canal de comunicação será consumido pelo receptor no ciclo seguinte do relógio. O reconhecimento é dado sob a condição de haver uma validação no canal e o *buffer* de entrada do receptor não estiver cheio.

Cada canal de um enlace SoCIN inclui um par de linhas de controle de fluxo denominadas *val* e *ack*. Essas linhas são conectadas a circuitos de controle em cada extremidade do canal de modo a implementar o protocolo de *handshake* que regula o fluxo de dados no canal. A linha *val* é alimentada pelo emissor do canal, o qual a ativa quando possuir um flit pronto para ser enviado ao receptor. A linha *ack*, por sua vez, é alimentada pelo receptor do canal, o qual a ativa quando detecta *val* em um e possui espaço no seu *buffer* de entrada para absorver o flit a ser enviado pelo emissor. Esse mecanismo é bastante conservador garante que nenhum flit será descartado pois a transmissão só é realizada após um acordo entre os envolvidos na negociação (emissor e receptor).

Em sua implementação, o protocolo de *handshake* é derivado do protocolo FIFO associado aos *buffers* de entrada. Como é ilustrado na Figura 5.14, a linha *val* é conectada diretamente à saída de estado *rok* do *buffer* de entrada do emissor, a qual sinaliza se o *buffer* possui algum dado para ser lido. No receptor, a linha *val* é conectada à entrada de controle *wr* do seu *buffer* de entrada, a qual comanda a escrita do flit recebido. Essa escrita só será realizada se houver espaço disponível para tal, o que é indicado pelo sinal *wok*, do qual é derivado o sinal *ack* retornado ao emissor. Na figura, são ilustrados dois autômatos de Mealy que representam o comportamento do controle de fluxo *handshake*. Em cada autômato, o único estado utilizado representa a máquina de estados do *buffer* FIFO ao qual está associado. No caso do emissor, quando *rok* for igual a 1 (transição superior), a saída *val* (valor no denominador da expressão) também será igual a 1. Por outro lado, se *rok* for igual a 0 (transição inferior), *val* será igual a 0. No caso do receptor, a saída *ack* será ativada em 1 somente quando *wok* e *val* forem simultaneamente iguais a 1. Do contrário, *ack* será igual a 0.

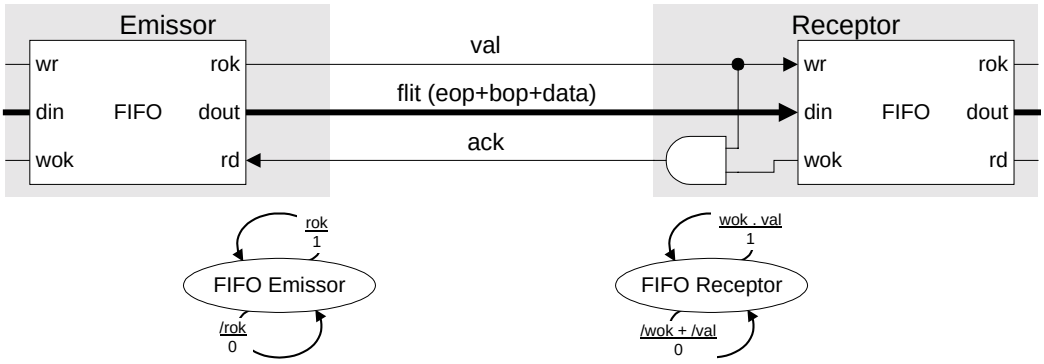


FIGURA 5.14 – Controle de fluxo baseado no protocolo de *handshake*.

O esquema da Figura 5.14 é uma abstração que busca ilustrar os sinais envolvidos no controle de fluxo de dados em um canal. São omitidos detalhes a respeito do chaveamento desses sinais no lado do emissor, já que podem haver até quatro *buffers* de entrada possíveis de serem conectados ao canal de saída de um dado emissor.

Uma alternativa à técnica de controle de fluxo ilustrada acima seria aquela baseada em créditos. Nessa técnica, cada emissor possui um contador que é inicializado com o valor correspondente à capacidade de armazenamento de flits do *buffer* do receptor. A cada flit enviado, esse contador é decrementado e o emissor não poderá enviar nenhum flit quando os créditos se esgotarem (contador igual a zero). Quando o receptor consome um flit armazenado em seu *buffer*, repassando-o a outro roteador, ele envia um bit de crédito ao emissor, o qual incrementa o seu contador de créditos. Essa técnica de controle de fluxo é aplicada na rede SPIN [GUE 2000a].

5.9 Arbitragem

A rede SoCIN utiliza um esquema de arbitragem distribuída no qual cada porta de um roteador possui um árbitro que seleciona o *buffer* de entrada do roteador a ser conectado ao canal de saída dessa porta. Os árbitros utilizam um critério baseado em prioridades dinâmicas rotativas (*Round-Robin*) que garante que nenhum pacote sofrerá com o problema de *starvation* (discutido na subseção 2.3.1).

Na estrutura do roteador, cada *buffer* de entrada possui um controlador de entrada que detecta a presença de um cabeçalho de pacote na saída do *buffer* e avalia a informação de roteamento desse cabeçalho para determinar a porta a ser utilizada para o encaminhamento do pacote em direção ao seu destinatário. Após concluída a seleção, o controlador de entrada informa o árbitro da porta selecionada através de um sinal de requisição. Esse árbitro recebe outros três sinais de requisição oriundos de outros controladores de entrada do roteador. Se o canal de saída estiver livre, o árbitro aplica o critério de prioridades corrente para selecionar uma das requisições. Após completar a arbitragem, o árbitro ativa um sinal de confirmação que é utilizado para comandar circuitos de chaveamento do roteador a fim de conectar a saída do *buffer* de entrada selecionado ao canal de saída da porta. Além disso, o árbitro atualiza o critério de prioridade de modo que o canal de entrada selecionado na arbitragem corrente terá o menor nível de prioridade na arbitragem seguinte, o que proporciona uma distribuição mais igualitária no uso dos canais de saída.

O árbitro de cada canal de saída é integrado junto a um circuito controlador de saída. Esse circuito é responsável por controlar a temporização dos sinais de confirmação que comandam os circuitos de chaveamento. A conexão entre o *buffer* de entrada selecionado e o canal de saída deve ser mantida até que o pacote que gerou a requisição seja completamente transferido pelo canal. Isso é sinalizado pela passagem do último flit do pacote pelo canal de saída e pelo recebimento de um sinal de reconhecimento enviado pelo canal de entrada do receptor do pacote.

5.10 Visão Geral da Arquitetura do Roteador RASoC

O roteador da rede SoCIN é denominado RASoC (*Router Architecture for System-on-Chip*). Ele consiste de um *soft-core* implementado sob a forma de um modelo VHDL parametrizável para síntese de redes-em-chip em sistemas integrados. O modelo VHDL foi desenvolvido com base na arquitetura dos FPGAs da Altera, os quais apresentam a restrição de não oferecerem *buffers tri-state* internos. Com isso, os circuitos de chaveamento são modelados de forma a serem sintetizados através de circuitos multiplexadores baseados em arranjos AND-OR. Contudo, a portabilidade para outras estruturas (como árvores de multiplexadores 2∝1, *buffers tri-state*, arranjos NAND-NAND etc) pode ser feita com poucas modificações no código VHDL.

Esta versão do roteador RASoC possui como parâmetros de síntese os mesmos parâmetros arquiteturais descritos nas seções anteriores: a largura do canal de dados (*n*), a largura da informação de roteamento (*m*) e a profundidade dos *buffers* de entrada (*p*). No código VHDL, esses parâmetros recebem outras denominações, sendo chamados, respectivamente *DATA_WIDTH*, *ROUTE_WIDTH* e *FIFO_DEPTH*.

5.10.1 A interface do roteador RASoC

Conforme é ilustrado na Figura 5.15.a, o roteador RASoC possui cinco portas bidirecionais denominadas *L* (*Local* ou local), *N* (*North* ou norte), *E* (*East* ou leste), *S* (*South* ou sul) e *W* (*West* ou oeste). Cada porta é compatível com os sinais do enlace SoCIN e possui dois canais unidirecionais: canal de entrada (sufixo *in*) e canal de saída (sufixo *out*), conforme é ilustrado na Figura 5.15.b. Como pode ser observado, internamente, o roteador RASoC possui um crossbar 5∝5 parcial que implementa

apenas 20 das 25 conexões que poderiam ser realizadas em um crossbar com essas dimensões. Isso se deve ao fato de que não é permitido a um canal de entrada de uma porta ser conectado ao canal de saída associado à mesma porta. Em outras palavras, um pacote que chega ao canal de entrada da porta *W* de um roteador, por exemplo, não pode ser encaminhado ao canal de saída dessa porta. Nesse caso, os únicos canais de saída possíveis de serem utilizados por esse pacote são aqueles associados às portas *L*, *N*, *E* e *S*.

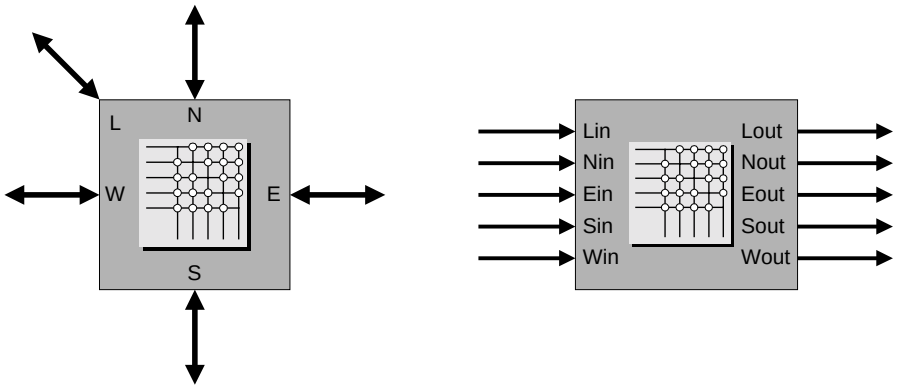


FIGURA 5.15 – Interface do roteador RASoC: (a) portas bidirecionais (b) representação alternativa mostrando os canais unidirecionais.

5.10.2 A organização interna do roteador RASoC

O roteador RASoC tem uma organização baseada em uma abordagem distribuída. Tanto o crossbar quanto os controladores que realizam as funções de roteamento, arbitragem e controle de fluxo são distribuídos em módulos associados aos canais de entrada e de saída de cada porta bidirecional. Conforme é ilustrado na Figura 5.16.a, existe um módulo associado a cada canal do roteador e esses módulos são interconectados por sinais internos de dado, estado e controle. Na Figura 5.16.b, são ilustrados alguns dos sinais de saída do módulo *Lin* (canal de entrada da porta local). As linhas mais finas representam sinais de requisição para os árbitros dos módulos dos canais de saída das outras portas e a linha espessa representa o canal de dados usado para a transferência de flits para esses canais de saída.

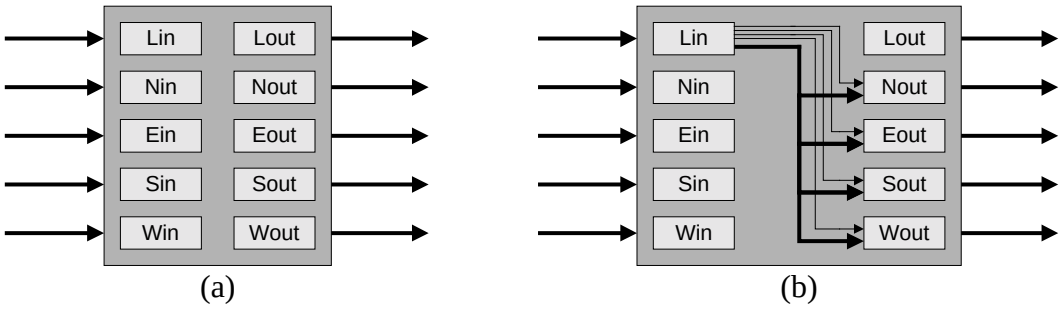


FIGURA 5.16 – Organização do roteador RASoC: (a) módulos associados aos canais; (b) exemplos de sinais internos do canal *Lin*.

No momento da modelagem da rede, o compilador pode ser orientado a não realizar a síntese de uma ou de outra porta do roteador. Isso é feito conectando-se as entradas dos canais associados ao nível 0 e deixando-se as saídas desses canais desconectadas. Como será visto, na seção a respeito dos resultados de síntese, isso permite a geração de redes de menor custo quando as mesmas não utilizam todas as portas do roteador (eg. grelha 2-D).

Os módulos mostrados na Figura 5.16 são baseados em dois circuitos denominados *Input Channel* e *Output Channel*. Esses circuitos são descritos brevemente a seguir, sendo que maiores detalhes são fornecidos no Apêndice 1.

O módulo *Input Channel*

O módulo *Input Channel* é ilustrado na Figura 5.17. Como pode ser observado, ele é composto por quatro blocos que são denominados *Input Flow Controller* (IFC), *Input Buffer* (IB), *Input Controller* (IC) e *Input Rd Switch* (IRS). O bloco IFC realiza o controle do fluxo de entrada, adaptando o protocolo de *handshake* do enlace SoCIN ao protocolo FIFO. O bloco IB, por sua vez, executa a memorização dos flits dos pacotes que chegam pelo canal de entrada até que eles possam ser encaminhados através de um canal de saída. Já o bloco IC realiza o roteamento dos pacotes que chegam ao bloco IB. Ele verifica a presença de um cabeçalho na saída de dado do bloco IB, aplica o algoritmo de roteamento com base no campo RIB do cabeçalho, envia uma requisição à saída selecionada por esse algoritmo e atualiza o valor do RIB para computar o deslocamento já realizado. Desde que cada módulo de canal de entrada é conectado a quatro módulos de canal de saída, é preciso realizar a conexão dos sinais de controle de fluxo da saída do bloco IB com os sinais de controle de fluxo do canal de saída selecionado pelo algoritmo de roteamento. Essa conexão é realizada, em parte, pelo bloco IRS, o qual conecta os sinais de controle de fluxo no sentido saída-entrada. Ele recebe os sinais de comando de leitura (*rd*) dos quatro canais de saída possíveis de serem requisitados e, baseado em sinais de confirmação enviados por esses canais de saída (*gnt*), conecta o sinal *rd* do canal de saída solicitado ao sinal *rd* do bloco IB. A conexão dos sinais de controle de fluxo no sentido entrada-saída é feita no módulo *Output Channel*.

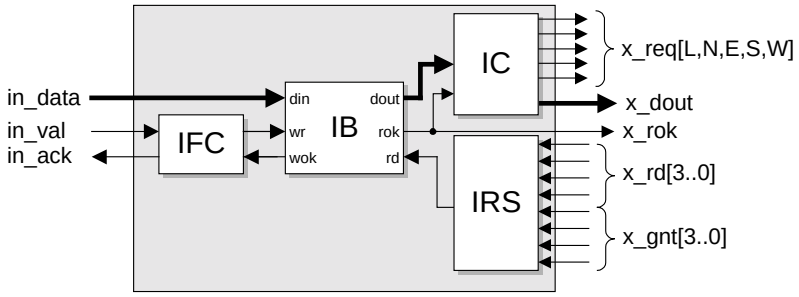


FIGURA 5.17 – Interface e estrutura do módulo *Input Channel*.

Na Figura 5.17, os nomes dos sinais conectados à interface externa do roteador utilizam o prefixo "*in_*", enquanto que os sinais conectados aos módulos de saída do roteador utilizam o prefixo "*x_*". Os sinais de interface com prefixo *in_* são compatíveis com o enlace SoCIN. Os demais sinais incluem: os sinais de requisição enviados aos

módulos *Output Channel* das outras portas (*x_req*); o canal de dados com *n*+2 bits (*x_dout*); o sinal de estado que informa a disponibilidade de um flit a ser lido do bloco IB (*x_rok*); os sinais de comando de leitura dos canais de saída (x_rd_i, onde *i* varia de 3 a 0); e os sinais de confirmação gerados pelos módulos *Output Channel* (x_gnt_i).

O módulo *Output Channel*

O módulo *Output Channel* é ilustrado na Figura 5.18. Ele é composto por quatro blocos denominados *Output Controller* (OC), *Output Data Switch* (ODS), *Output Rok Switch* (ORS) e *Output Flow Controller* (OFC). O bloco ODS conecta a saída de dado do bloco IB do canal de entrada selecionado ao canal de dados do canal de saída. O bloco ORS, por sua vez, seleciona o sinal *rok* do bloco IB do canal de entrada conectado ao canal de saída. Esse sinal é dirigido ao bloco OFC, o qual realiza o controle do fluxo de saída de dados, adaptando o protocolo o FIFO ao protocolo de *handshake* do enlace SoCIN. O bloco OC recebe as requisições oriundas dos canais de entrada, seleciona uma das requisições e envia um sinal de confirmação ao canal de entrada selecionado. Esse sinal de confirmação é utilizado como sinal de comando pelos blocos ODS e ORS, assim como pelo bloco IRS do canal de entrada selecionado, para efetuar o chaveamento dos sinais internos do roteador. Como é o sinal de confirmação que comanda os blocos de chaveamento, o bloco OC monitora a transferência do último flit ao receptor conectado no outro extremo do enlace associado ao canal de saída. Uma vez que essa transferência tenha sido completada, o bloco OC desativa a linha de confirmação de modo a desfazer a conexão previamente estabelecida.

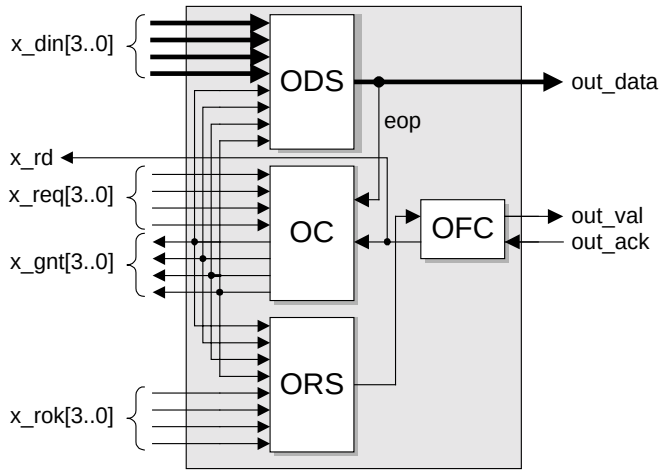


FIGURA 5.18 – Interface e estrutura do módulo *Output Channel*.

Na interface do módulo *Output Channel*, mostrada na Figura 5.18, os nomes dos sinais conectados à interface externa do roteador utilizam o prefixo "*out_*", enquanto que os sinais conectados aos módulos de entrada do roteador utilizam o prefixo "*x_*". Os sinais de interface do tipo *out* são compatíveis com o enlace SoCIN. Os demais sinais incluem: os canais de dado dos blocos IB que podem ser conectados ao canal de saída (x_din_i, onde *i* varia de 3 a 0); o comando de leitura para o bloco IB selecionado (*x_rd*); os sinais de requisição de uso do canal de saída (x_req_i); os sinais de confirmação de seleção para uso do canal de saída (x_gnt_i); e os sinais de estado que informam a disponibilidade de um flit a ser lido desses blocos IB (x_rok_i).

5.11 Estrutura do modelo VHDL do roteador RASoC

O modelo VHDL do roteador RASoC é composto por três níveis de entidade que descrevem a arquitetura do roteador. No primeiro nível (nível mais alto), é definida a entidade *rasoc*, a qual descreve a estrutura do roteador baseada na interconexão de instâncias das entidades *input_channel* e *output_channel*. Essas entidades constituem o segundo nível da descrição e definem a estrutura dos módulos de entrada e de saída, respectivamente. No terceiro nível da descrição VHDL, são definidas as entidades que implementam os blocos que compõem os módulos de entrada (IC, IFC, IB e IRS) e de saída (OC, OFC, ODS e ORS), conforme é ilustrado na hierarquia apresentada na Figura 5.19.

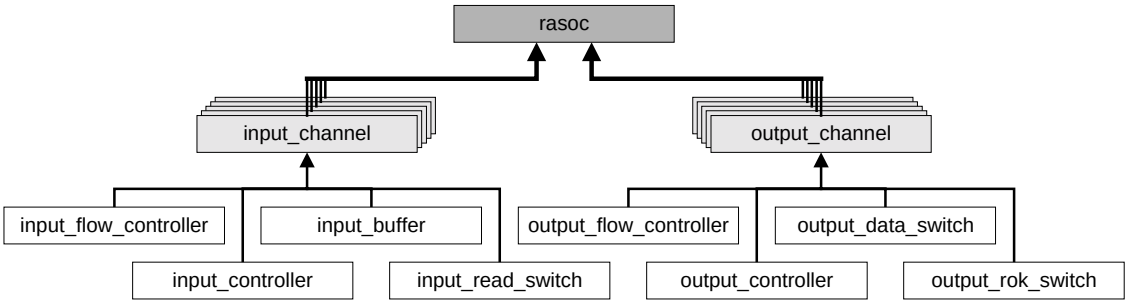


FIGURA 5.19 – Estrutura do modelo VHDL do roteador RASoC.

A entidade RASoC possui três parâmetros de entradas: FIFO_DEPTH (*p*), DATA_WIDTH (*n*) e ROUTE_WIDTH (*m*), descritos na Tabela 5.2. Esses parâmetros são transmitidos às demais entidades de modo a configurar as dimensões de seus componentes. Por exemplo, a entidade *input_buffer* recebe os dois primeiros parâmetros listados acima e modela um *buffer* com FIFO_DEPTH palavras de DATA_WIDTH+2 bits. Já a entidade *output_data_switch* recebe apenas o parâmetro DATA_WIDTH e modela um multiplexador 4∝1 com canais de DATA_WIDTH+2 bits.

TABELA 5.2 – Parâmetros do roteador RASoC.

Parâmetro	**Descrição**
FIFO_DEPTH	Capacidade dos *buffers* FIFO quanto ao número de flits (*p*)
DATA_WIDTH	Largura do canal de dados (*n*)
ROUTE_WIDTH	Largura do campo RIB do cabeçalho (*m*)

A arquitetura da entidade *rasoc* consiste de uma descrição estrutural baseada nos diagrama de bloco ilustrado na Figura 5.20. Esse diagrama apresenta apenas as instâncias das entidades *input_controller* (*Lin*, *Nin*, *Ein*, *Sin* e *Win*) e *output_controller* (*Lout*, *Nout*, *Eout*, *Sout* e *Wout*) e inclui, também, os sinais que interconectam essas instâncias. Para oferecer uma melhor legibilidade à figura, esses sinais são agrupados em cinco barramentos denominados *Lbus*, *Nbus*, *Ebus*, *Sbus* e *Wbus*.

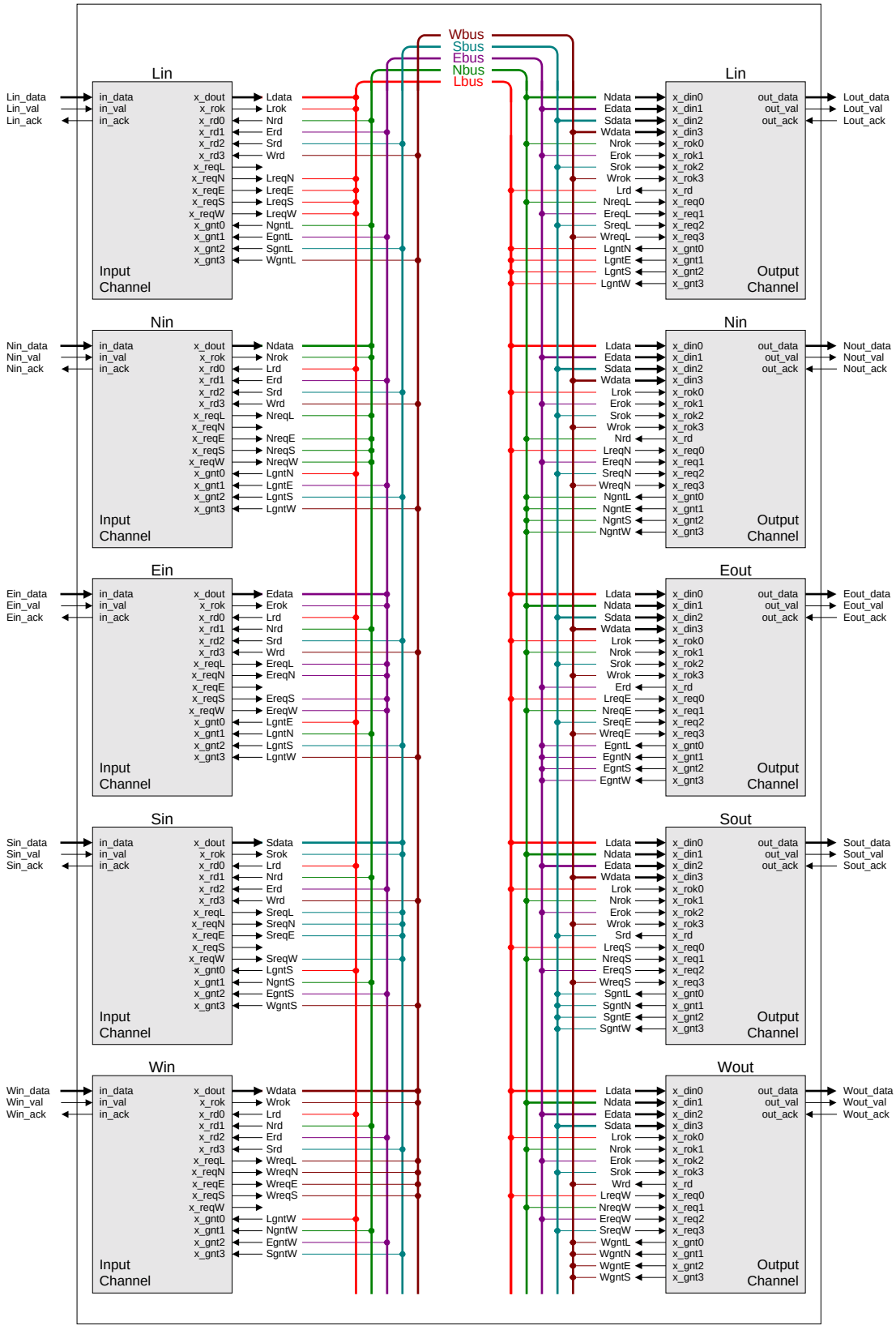


FIGURA 5.20 – Instâncias e sinais da entidade *rasoc*.

5.12 Simulação

A validação do modelo VHDL do roteador RASoC for realizada por simulação utilizando-se o simulador do ambiente Quartus II da Altera [ALT 2003]. A validação foi efetuada em diferentes níveis, simulando-se, primeiramente, cada bloco componente do roteador. Após, foram simulados e validados os módulos dos canais de entrada e de saída, bem como a interface de um roteador completo com cinco portas. Por fim, foram simuladas redes de pequena escala submetidas a uma carga de comunicação fornecida por um modelo de gerador de tráfego. Nesta seção, é apresentado um subconjunto dos diagramas de formas de onda obtidos nessas simulações, incluindo a simulação da interface do roteador e de uma rede em grelha 2×2.

5.12.1 Simulação da interface do roteador RASoC

Para ilustrar a validação e o funcionamento do modelo VHDL do roteador RASoC foram utilizados vetores de entrada baseados na situação representada na Figura 5.21, na qual pode-se observar que quatro pacotes atravessam o roteador concorrendo por seus recursos. Os pacotes chegam ao roteador pelos canais de entrada *Nin*, *Ein*, *Sin* e *Win*, e devem ser encaminhados aos canais de saída *Sout*, *Sout*, *Eout* e *Lout*, respectivamente. Como os pacotes que chegam pelos canais *Nin* e *Ein* precisam compartilhar o canal de saída *Sout*, o mecanismo de arbitragem deve escalonar o uso desse canal pelos pacotes requisitantes.

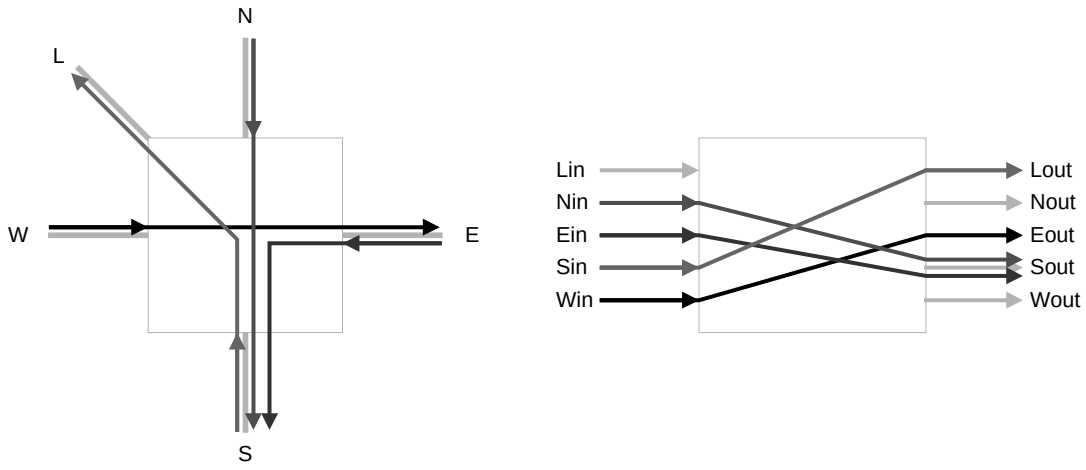


FIGURA 5.21 – Configuração de tráfego de pacotes em um roteador.

Para garantir uma maior legibilidade aos diagramas de forma de onda, as simulações baseiam-se em um roteador com os seguintes parâmetros:

a) Largura física canal de dados (parâmetro *n* ou `DATA_WIDTH`) = 8 bits;

b) Largura da informação de roteamento (parâmetro *m* ou `ROUTE_WIDTH`) = 8 bits; e

c) Capacidade dos *buffers* (parâmetro *p* ou `FIFO_DEPTH`) = 2 flits.

Com essa configuração, principalmente para os parâmetros *n* e *m*, torna-se mais clara a identificação do conteúdo dos pacotes e, principalmente, dos seus cabeçalhos.

Cada canal físico possui *n*+2 bits para incluir os bits de enquadramento (*bop* e *eop*). Nos diagramas de formas de onda, cada flit será composto por três dígitos hexadecimais (eg. `131h`), sendo que o dígito mais significativo pode assumir os seguintes valores: `1` (no cabeçalho do pacote), `2` (no terminador do pacote) e `0` (nos demais flits da carga útil do pacote). Com relação ao cabeçalho, a informação de roteamento ocupa todos os *n* bits do canal de dados (pois *m* é igual *n*). Dessa forma, nesta configuração, não existem bits reservados para um protocolo de mais alto nível. Com isso, o dígito hexadecimal menos significativo do cabeçalho é destinado à identificação da rota na direção Y enquanto que o dígito intermediário é destinado à identificação da rota na direção X, conforme é ilustrado na Figura 5.22. Destaca-se que, embora seja uma palavra separada, o terminador é parte integrante da carga útil do pacote.

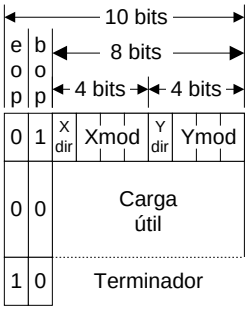


FIGURA 5.22 – Formato do pacote para *n* = 8 e *m* = 8.

Na Tabela 5.3, são apresentados alguns exemplos de palavras de cabeçalho para configuração de parâmetros a ser utilizada (não estão incluídos os bits de enquadramento).

TABELA 5.3 – Exemplos de palavras de cabeçalho para *m* = 8.

Palavra	**Significado**
`00h = 0000.0000b`	pacote deve ser encaminhado pela porta local
`10h = 0001.0000b`	pacote deve percorrer 1 enlace em direção ao leste (*Xdir* = 0)
`90h = 1001.0000b`	pacote deve percorrer 1 enlace em direção ao oeste (*Xdir* = 1)
`01h = 0000.0001b`	pacote deve percorrer 1 enlace em direção ao norte (*Ydir* = 0)
`09h = 0000.1001b`	pacote deve percorrer 1 enlace em direção ao sul (*Ydir* = 1)
`2Ah = 0010.1010b`	pacote deve percorrer 2 enlaces em direção ao leste (*Xdir* = 0) e 2 enlaces em direção ao sul (*Ydir* = 1)

Nas simulações realizadas para a obtenção dos diagramas de formas de onda mostrados a seguir, foi considerado que todos os pacotes possuem o mesmo tamanho (4 flits). Embora o conteúdo da parte de dados da carga útil de um pacote seja irrelevante para o roteador, foram definidos valores exclusivos para cada flit da carga útil de cada pacote a fim de facilitar a identificação dos mesmos. Na Tabela 5.4, são apresentados os pacotes utilizados nas simulações para a configuração de tráfego de pacotes em um roteador mostrada previamente na Figura 5.21. Em cada linha da tabela, é mostrado o canal de entrada através do qual o pacote chega ao roteador, o canal de saída pelo qual ele deve ser encaminhado e os conteúdos do cabeçalho e dos quatro flits da carga útil, incluindo o terminador.

TABELA 5.4 – Pacotes utilizados na simulação.

No. do pacote	**Canal de entrada**	**Canal de saída**	**Cabeçalho** *bop* = 1 *eop* = 0	**Flit0** *bop* = 0 *eop* = 0	**Flit1** *bop* = 0 *eop* = 0	**Flit2** *bop* = 0 *eop* = 0	**Flit3** *bop* = 0 *eop* = 1
0	*Win*	*Eout*	132h	000h	001h	002h	203h
1	*Nin*	*Sout*	10Ah	010h	011h	012h	213h
2	*Ein*	*Sout*	10Bh	020h	021h	022h	223h
3	*Sin*	*Lout*	100h	030h	031h	032h	233h

Na Tabela 5.5, é explicado o roteamento a ser realizado por cada um dos pacotes mostrados na Tabela 5.4. São apresentados os campos do cabeçalho antes e depois do roteamento, assim como a porta de saída utilizada para encaminhar os pacotes.

TABELA 5.5 – Conteúdo do RIB dos pacotes antes e após o roteamento.

	Cabeçalho antes do roteamento						**Cabeçalho depois do roteamento**				
No. do pacote	**RIB**	**X dir**	**X mod**	**Y dir**	**Y mod**	**Canal de saída**	**RIB**	**X dir**	**X mod**	**Y dir**	**Y mod**
0	32h	0	011	0	010	*Eout*	22h	0	010	0	010
1	0Ah	0	000	1	010	*Sout*	09h	0	000	1	001
2	0Bh	0	000	1	011	*Sout*	0Ah	0	000	1	010
3	00h	0	000	0	000	*Lout*	00h	0	000	0	000

Nos diagramas de forma de onda a seguir, obtidos pela simulação do modelo VHDL no ambiente Quartus II da Altera, é assumido que o núcleo e os roteadores conectados aos canais de saída do roteador sendo simulado possuem *buffers* de entrada com profundidade infinita. Ou seja, os destinatários imediatos dos pacotes que atravessam o roteador sendo simulado possuem capacidade de consumir imediatamente todos os flits dos pacotes a eles encaminhados. Essa consideração foi tomada para poder identificar pontualmente a latência devida a um único roteador.

Na Figura 5.23, é ilustrado o roteamento do pacote 0 que entra pelo canal *Win* e sai pelo canal *Eout*. Analisando-se cada onda do diagrama, observa-se que o sinal *Win_val* é mantido alto até que todo o pacote seja absorvido pelo *buffer* do canal de entrada. O sinal *Win_ack* é ativado apenas quando o *buffer* de entrada tem capacidade de absorver algum flit, sendo que apenas nas transições de subida do relógio nas quais *Win_val* = *Win_ack* = 1 é que o flit em *Win_data* é efetivamente transferido. O sinal *Eout_val* é ativado quando o caminho entre *Win* e *Eout* é estabelecido, sendo mantido alto enquanto houver algum flit do pacote na saída do *buffer* de entrada. A conexão entre os canais de entrada e de saída se mantém até que o terminador do pacote (flit 203h) seja completamente transferido (*eop* = 1 e *Eout_ack* = 1).

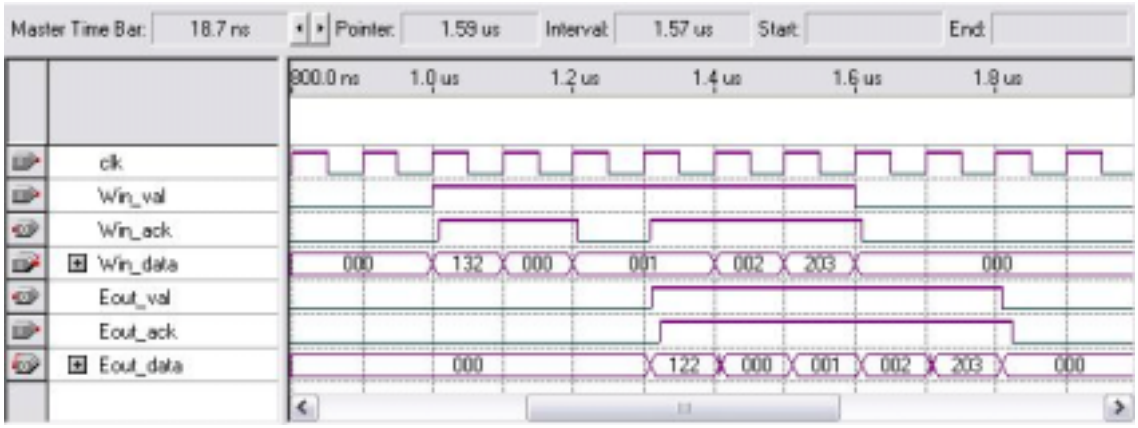


FIGURA 5.23 – Transferência do pacote 0 entre os canais *Win* e *Eout*.

Ainda a respeito da Figura 5.23, destaca-se que, devido à pouca profundidade do *buffer* do canal de entrada (igual a dois) e à latência de registro, roteamento, arbitragem e chaveamento (igual a três), o sinal de controle de fluxo *Win_ack* é mantido alto por apenas dois ciclos de relógio, correspondentes ao registro do flit de cabeçalho (`132h`) e do primeiro flit de dado (`000h`) do pacote, é desativado até que o flit de cabeçalho seja encaminhado liberando espaço no *buffer* para o recebimento de um novo flit.

Conforme foi antecipado, é assumido que o *buffer* de entrada do roteador conectado ao outro extremo do enlace associado à porta leste do roteador sendo simulado tem profundidade infinita. Por isso, a linha *Eout_ack* é ativada tão logo a linha *Eout_val* o seja e se mantém em nível alto durante toda a transferência. Finalmente, observa-se que o campo de módulo da direção X é decrementado (de 3 para 2) de modo a computar o deslocamento realizado. A latência para transferência do pacote 0 pelo roteador é igual a sete ciclos de relógio, dos quais três são devidos ao encaminhamento do cabeçalho (registro, roteamento, arbitragem e chaveamento) e quatro são devidos ao tamanho da carga útil.

Na Figura 5.24, é ilustrado o encaminhamento dos pacotes 1 e 2 pelo canal de saída *Sout* do roteador. Como os dois pacotes chegam simultaneamente aos canais de entrada *Nin* e *Ein*, respectivamente, um desses canais é selecionado pelo árbitro associado ao canal *Sout* e conectado a essa saída. O outro canal de entrada é mantido em estado de espera até que o pacote do primeiro canal selecionado seja completamente encaminhado ao próximo roteador. Na figura, observa-se que o primeiro pacote encaminhado é o pacote 1, o qual chega ao roteador pelo canal de entrada *Nin*. A temporização da transferência desse pacote é idêntica àquela ilustrada no exemplo anterior. Por outro lado, a temporização do pacote 2, o qual chega ao roteador pelo canal *Ein*, tem como particularidade a espera pela liberação do canal de saída e conexão do canal de entrada *Ein* ao canal de saída *Sout*. A latência de contenção sofrida por esse pacote é igual a sete ciclos de relógio. Incluindo-se as latências devidas ao cabeçalho (três) e à carga útil (quatro), a latência do pacote 2 para atravessar o roteador é igual a 14 ciclos de relógio.

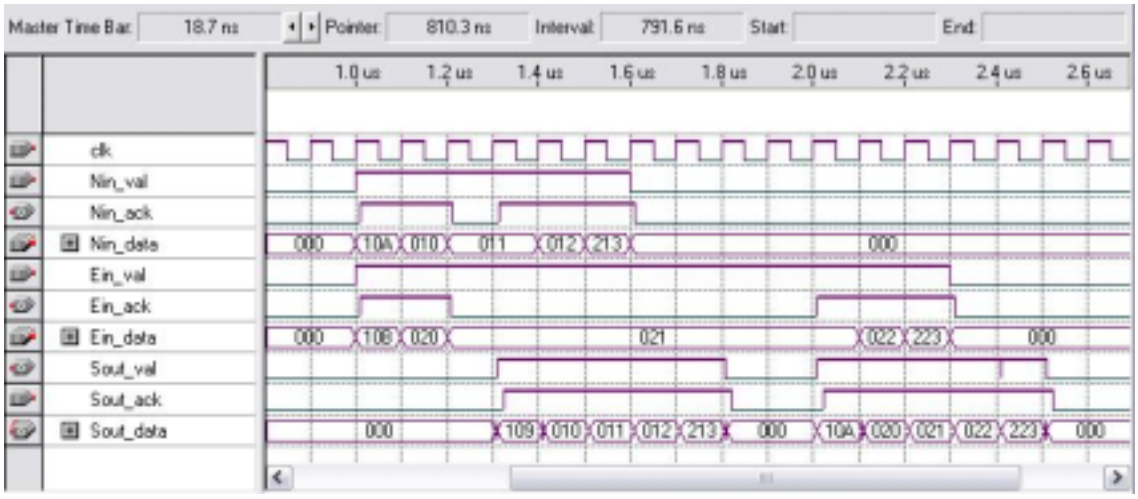


FIGURA 5.24 – Transferência dos pacotes 1 e 2 entre os canais *Nin*, *Ein* e *Sout*.

Na Figura 5.24, é ilustrado o roteamento do pacote 3, o qual entra pelo canal *Sin* e sai pelo canal *Lout*. A temporização desse pacote é idêntica àquela do pacote 0, mostrada na Figura 5.23. Porém, destaca-se que o cabeçalho não é atualizado, pois, quando o cabeçalho do pacote 3 chega ao canal *Sin*, seus campos de módulo são ambos iguais a 0, o que indica que não há mais deslocamentos a serem realizados nas direções X e Y e que o pacote deve ser encaminhado ao pelo canal *Lout*.

FIGURA 5.25 – Transferência do pacote 3 entre os canais *Sin* e *Lout*.

5.12.2 Simulação de redes SoCIN de pequena escala

Após a validação individual do roteador RASoC, foram modelados sistemas integrados de pequena escala utilizando redes SoCIN com topologias em grelha e em toróide 2∝2. Os núcleos de processamento desses sistemas foram baseados em um modelo de gerador de tráfego parametrizável desenvolvido especialmente para essa validação e que realiza a geração de pacotes para envio a diferentes destinatários. As simulações desses sistemas permitiram confirmar a correção do modelo do roteador, pois todos os pacotes gerados foram entregues aos seus destinatários sem a ocorrência de bloqueios na rede.

Arquitetura do gerador de tráfego

O modelo de gerador de tráfego (GT) desenvolvido para a validação do roteador RASoC e da rede SoCIN trata-se de um *soft-core* VHDL parametrizável com interface compatível com o protocolo SoCIN. Sua arquitetura utiliza um conjunto de subunidades baseadas em um modelo de gerador de tráfego simples (GTS). Cada subunidade GTS é responsável por um padrão de geração de pacotes definido pelos seguintes parâmetros:

a) `SOURCE_ID` – identificador da instância de GT ao qual o GTS está associado;

b) `TARGET_ID` – identificador da instância de GT em direção à qual serão encaminhados os pacotes gerados pelo GTS;

c) `ROUTE_WIDTH` – largura da informação de roteamento (RIB) a ser incluída no cabeçalho do pacote;

d) `DATA_WIDTH` – largura do canal de dados da rede;

e) `RIB` – informação de roteamento a ser colocada no cabeçalho dos pacotes. Ela define a rota a ser utilizada para o encaminhamento dos pacotes ao nodo destinatário;

f) `PACKET_LENGTH` – número de flits da carga útil dos pacotes a serem gerados pelo GTS;

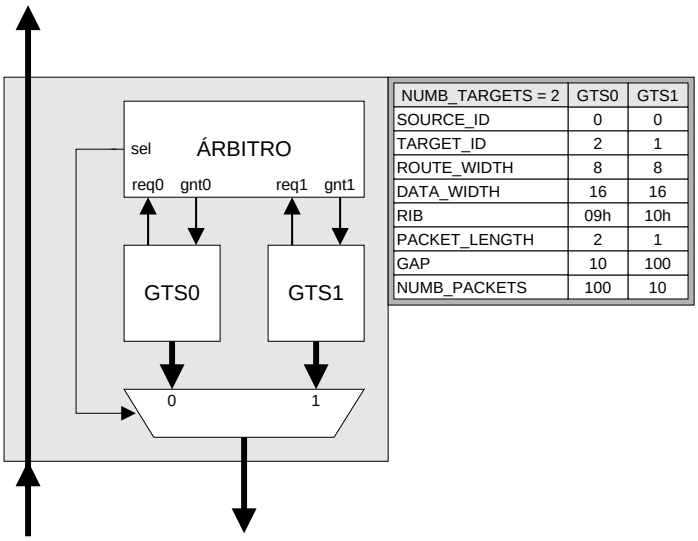
g) `GAP` – intervalo de geração de pacotes que define o número de ciclos de relógio entre o envio do terminador de um pacote e a geração do pacote seguinte a ser enviado ao destinatário definido pelo parâmetro `RIB`; e

h) `NUMB_PACKETS` – número de pacotes a serem gerados.

Em cada instância do modelo GT pode haver mais de uma instância do modelo GTS, confome o número de padrões de geração de pacotes necessários. Os parâmetros `SOURCE_ID`, `ROUTE_WIDTH` e `DATA_WIDTH` serão iguais para todas as instâncias de GTS de uma mesma GT, porém, os demais parâmetros podem diferir de modo que cada GTS descreva um padrão exclusivo de geração de pacotes.

As múltiplas instâncias de GTS de um mesmo GT concorrem pelo uso da porta do roteador à qual o GT está conectado. Essa concorrência é gerenciada por um árbitro que recebe requisições dos GTSs, aplica um critério de arbitragem quando existirem múltiplas requisições simultâneas, seleciona um dos módulos e comanda um multiplexador para conectar o módulo selecionado à porta local do roteador. Para garantir o uso balanceado dessa porta pelos diversos GTSs, o árbitro utiliza uma política de prioridades baseada em uma fila circular (*round-robin*) que determina que o GTS selecionado em um ciclo de arbitragem receba o menor nível de prioridade na no ciclo de arbitragem seguinte.

Uma representação simplificada de uma instância de modelo GT é apresentada na Figura 5.26. Ela inclui duas instâncias de GTS, cada qual responsável pela geração de um padrão de tráfego. Na figura, é mostrada a tabela que que descreve a configuração de cada GTS. Por exemplo, o GTS0 é configurado para gerar 100 pacotes (`NUMB_PACKETS`) com 2 flits na carga útil (`PACKET_LENGTH`) em intervalos de geração de 10 ciclos (`GAP`). Esses pacotes serão encaminhados ao GT de identidade 2 (`TARGET_ID`) por meio da rota descrita pelo `RIB` de 8 bits (`ROUTE_WIDTH`) igual a 09h (um enlace em direção ao sul).



NUMB_TARGETS = 2	GTS0	GTS1
SOURCE_ID	0	0
TARGET_ID	2	1
ROUTE_WIDTH	8	8
DATA_WIDTH	16	16
RIB	09h	10h
PACKET_LENGTH	2	1
GAP	10	100
NUMB_PACKETS	100	10

FIGURA 5.26 – Arquitetura simplificada de um gerador de tráfegos para dois padrões diferentes de geração de pacotes.

Além do canal de saída, pelo qual são encaminhados os pacotes gerados, o módulo GT também possui um canal de entrada. Esse canal recebe pacotes e os repassa a uma interface de saída de modo a retirar da rede os pacotes encaminhados ao GT. Esses pacotes não são processados pelo GT e a interface de saída é utilizada somente para observar os pacotes recebidos por cada GT após a simulação.

Esse modelo de gerador de tráfego tem algumas limitações e sua arquitetura requer ainda um número de melhorias a fim de estender a sua funcionalidade e permitir o seu uso não apenas para a validação da rede, mas, também, para a obtenção de parâmetros de desempenho (eg. latência média de comunicação e vazão).

Simulação de sistemas integrados

Para validar o roteador RASoC, foram modelados SoCs baseados em redes SoCIN com topologias diretas dos tipos grelha 2-D e toróide 2-D. Um dos sistemas utilizados é ilustrado na Figura 5.27. Ele é constituído por quatro módulos GT interconectados através de uma grelha 2×2.

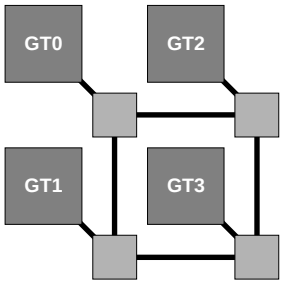


FIGURA 5.27 – Sistema com quatro GTs interconectados por uma grelha 2×2.

Um dos sistemas simulados foi baseado no padrão de comunicação ilustrado na Figura 5.28. Conforme é mostrado, existem quatro tarefas comunicantes, sendo que cada tarefa comunica-se com outras duas. Por exemplo, a Tarefa 0 envia mensagens com 1 palavra a cada 100 ciclos de relógio (1/100) à Tarefa 2 e mensagens com 2 palavras a cada 10 ciclos de relógio (2/10) à Tarefa 1. Essas quatro tarefas foram mapeadas em um sistema semelhante ao da Figura 5.27 e seus geradores de tráfego foram configurados utilizando os parâmetros resumidos na Tabela 5.6.

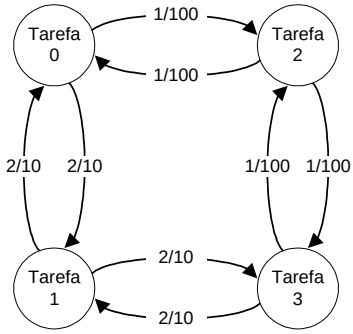


FIGURA 5.28 – Exemplo de padrão de comunicação simulado.

TABELA 5.6 – Resumo dos parâmetros utilizados na configuração do padrão de comunicação da Figura 5.28.

GT	GTS	TARGET_ID	PACKET_LENGTH	GAP	NUMB_PACKETS
0	0	1	2	10	100
	1	2	1	100	10
1	0	0	2	10	100
	1	3	2	10	100
2	0	0	1	100	10
	1	3	1	100	10
3	0	1	2	10	100
	1	2	1	100	10

Esse sistema integrado foi então validado por simulação e os resultados confirmaram a correção dos modelos do roteador e do gerador de tráfego. A Figura 5.29 apresenta a visão geral da simulação do sistema descrito anteriormente. O sinal denominado `CoreXY_data` (onde XY é igual a 00, 01, 10 ou 11) representa o canal de dados da interface de saída do gerador de tráfego XY pela qual são observados os pacotes recebidos pelo gerador. Já o sinal `CoreXY_rok` informa a disponibilidade de um flit a ser lido nessa interface. Observa-se, pela densidade de pulsos do sinal `rok` associado que o gerador de tráfego 01 é o que recebe o maior número de mensagens, enquanto que o gerador de tráfego 10 é o menos requisitado. O término da simulação é indicado pelo sinal `eot`, o qual é ativado quando todos os geradores de tráfego concluem o envio de suas mensagens. O diagrama conta ainda com um sinal denominado `ticks` que rotula cada período de relógio a fim de auxiliar a avaliação da simulação.

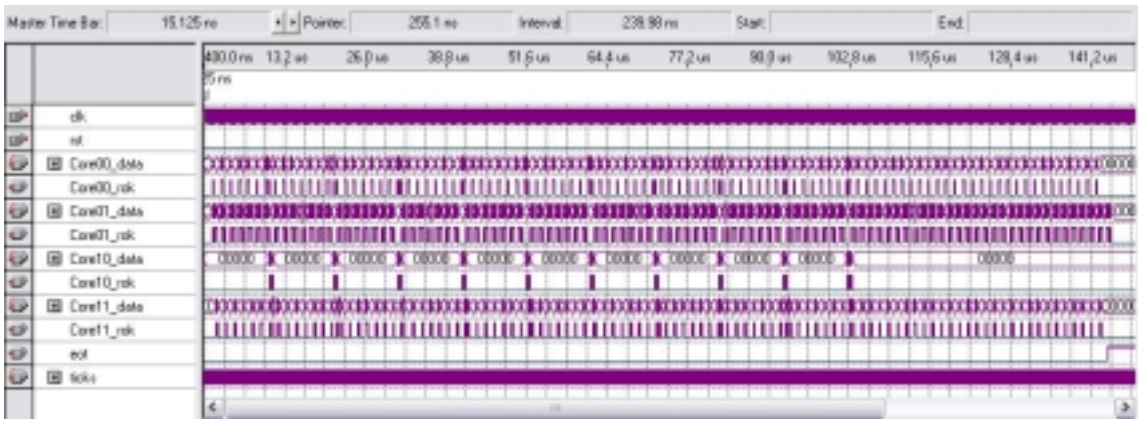


FIGURA 5.29 – Diagrama de formas de onda geral da simulação de um SoC 2∝2.

A Figura 5.30 ilustra uma região do diagrama de formas de onda acima apresentado. Nele, pode-se observar, por exemplo, que o gerador de tráfego 00 recebe uma mensagem com uma palavra de dado emitida pelo gerador 10 (indicado pelo número 2 no dígito central do cabeçalho) e outra com duas palavras de dado emitida pelo gerador de tráfego 01.

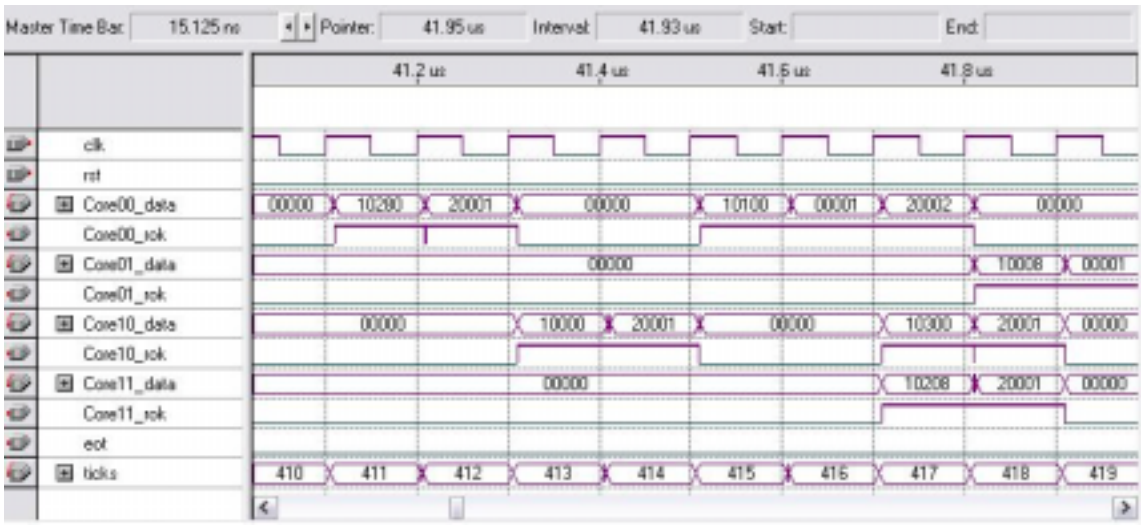


FIGURA 5.30 – Aproximação do diagrama de formas de onda da simulação.

5.13 Síntese

Nesta seção, são apresentados resultados da síntese do roteador RASoC para diferentes configurações dos parâmetros *n*, *m* e *p* (denominados *DATA_WIDTH*, *ROUTE_WIDTH* e *FIFO_DEPTH* no modelo VHDL, respectivamente). Os resultados são baseados na síntese do modelo VDHL em FPGA utilizando-se o ambiente QUARTUS II da Altera [ALT 2003]. São indicadas as quantidades de células lógicas consumidas em cada configuração, bem como parâmetros de desempenho obtidos através da ferramenta análise de temporização. Nos resultados apresentados, o compilador foi orientado à realização de síntese sem otimizações e o FPGA alvo foi o dispositivo EPF10K200SFC672-1 [ALT 2003a]. A escolha por tal dispositivo deve-se à sua disponibilidade de pinos para a síntese de roteadores com canais largos (eg. 32 bits) e também ao fato que os núcleos de processamento utilizados como referência para comparação de área são apresentados, na literatura, com base em disposivtios dessa família de FPGA (FLEX10K). Com relação ao número de pinos, destaca-se que, no momento da implementação da rede, é tipicamente desnecessário externar os pinos dos roteadores, e o requisito do número de pinos será definido em função das interfaces dos núcleos de processamento.

5.13.1 Síntese do roteador RASoC

Avaliação do impacto da profundidade dos *buffers* e da largura dos canais no custo do roteador

No gráfico de colunas da Figura 5.31, são apresentados os resultados de área da síntese do roteador RASoC no FPGA EPF10K200SFC672-1 para diferentes configurações de largura de canal e profundidade de *buffers*. Cada padrão de coluna expressa o número de células lógicas (LCs – *Logic Cells*) necessárias para uma determinada largura de canal (*n* igual a 8 16 ou 32 bits) e para quatro valores de profundidade de *buffer* (*p* igual a 1, 2, 3 e 4 flits), sendo que a largura do campo RIB do cabeçalho foi fixada em 8 bits (parâmetro *m*). Na configuração de menor custo (*n* = 8 e *p* = 1), o roteador ocupa 463 LCs, enquanto que, na configuração mais cara (*n* = 32 e *p* = 4), o seu custo é de 1830 LCs.

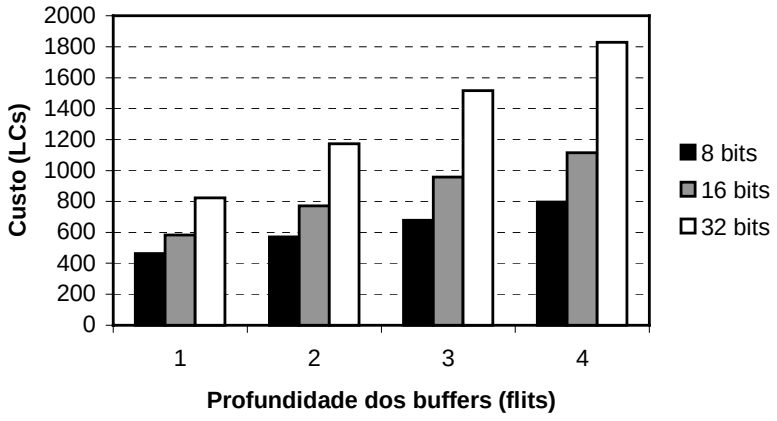


FIGURA 5.31 – Impacto da profundidade dos *buffers* na área do roteador RASoC para

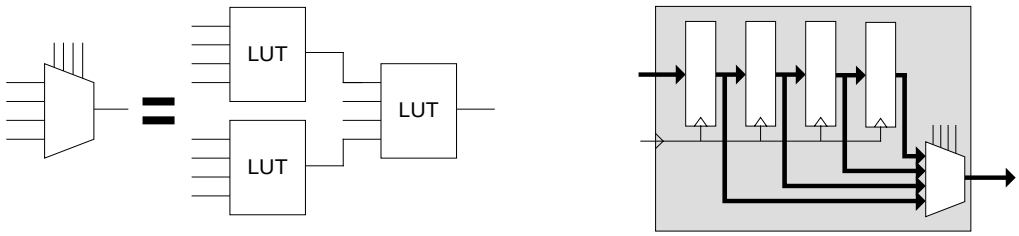
diferentes configurações de largura de canal (8, 16 e 32 bits).

Esses valores podem ser comparados com os resultados de área para núcleos de processamento de 8, 16 e 32 bits apresentados por Mattos e Carro em [MAT 2002]. Nesse trabalho, os autores comparam os custos em LCs para a síntese de dois núcleos de microcontrolador (FemtoJava e MCS8051) e um núcleo de processador escalar (RISCO) em FPGAs da família FLEX10K da Altera. O FemtoJava [ITO 2000] é um microcontrolador baseado em pilha que roda *bytecodes* Java. Suas características incluem: conjunto de instruções *bytecode* reduzido, arquitetura *Harvard*, ortogonalidade de execução e facilidade de inserção e remoção de instruções conforme os requisitos da aplicação alvo. O núcleo do microcontrolador MCS8051 é uma versão reduzida da arquitetura original desse microcontrolador, pois implementa um subconjunto de seu conjunto de instruções. O RISCO é um microprocessador RISC de 32 bits com um pipeline de três estágios. Uma diferença importante entre o FemtoJava e os demais núcleos é que ele inclui um hadware para manipulação de interrupções, o que implica em um custo adicional ausente no MCS8051 e no RISCO. A Tabela 5.7 apresenta um comparativo dos resultados de síntese desses processadores com o roteador RASoC para os tamanhos de palavra disponíveis.

TABELA 5.7 – Custo em LCs de núcleos de processamento e do roteador RASoC.

	FemtoJava	**MCS8051**	**RISCO**	**RASoC (*p* = 1)**	**RASoC (*p* = 4)**
8 bits	1481	659	. ∝.	463	795
16 bits	1979	. ∝.	. ∝.	583	1115
32 bits	. ∝.	. ∝.	1271	823	1830

Para *buffers* de profundidade mínima (p = 1), o custo do roteador RASoC em relação aos custos dos processadores varia de 29,5% (para o FemtoJava de 16 bits) a 70,3% (para o MCS8051). Já para p = 4, as relações de custo mínima e máxima são iguais a 53,7% (para o FemtoJava de 8 bits) e 144% (para o RISCO), respectivamente. Nota-se, então, que, para núcleos com essas granularidades, o custo do roteador implementado em FPGA pode ser bastante significativo. Isso ocorre porque, nos FPGAs da Altera, os circuitos de chaveamento são sintetizados sob a forma de LUTs (*Look-Up Tables*), o que onera a implementação do roteador (Figura 5.32.a). Em outras tecnologias (eg. *standard-cell*), as chaves poderiam ser implementadas utilizando-se estruturas mais simples, o que reduziria o custo associado ao chaveamento. Além disso, os *buffers* FIFO foram descritos no modelo do roteador RASoC de modo a não utilizarem blocos de memória RAM do FPGA, deixando-os livres para a síntese de núcleos de processamento. Com isso, sua implementação baseada em flip-flops e chaves de seleção (multiplexadores), representada na Figura 5.32.b, possui um custo bastante elevado. Em outras tecnologias, esse custo seria menor, pois o FIFO poderia ser implementado utilizando-se bits de memória SRAM CMOS.



(a) (b)

FIGURA 5.32 – Estruturas sintetizadas em FPGA: (a) multiplexador baseado em LUTs; (b) *buffers* FIFO.

Avaliação do impacto da profundidade dos *buffers* e da largura dos canais no desempenho do roteador

Na Figura 5.33, são apresentados os resultados da medição de freqüência máxima do roteador após a síntese. Pode-se observar que, nos resultados para larguras de canal de 8 bits, o aumento da profundidade dos *buffers* resulta em uma redução da freqüência máxima de operação. Contudo, para as outras configurações esse comportamento não se reproduz. Por exemplo, para as configurações de 16 bits, a diferença entre as freqüências de operação dos roteadores com *buffers* com um e com dois flits é muito menos significativa que na configuração de 8 bits. Já na configuração de 32 bits, há um aumento da freqüência quando a profundidade do *buffer* é aumentada de duas para três posições. Além disso, não há como relacionar a largura de canal a uma maior ou menor freqüência de operação, pois, em cada configuração de profundidade de *buffer*, uma ou outra configuração de largura de canal atingiu o maior desempenho, efeito que pode ser atribuído ao compilador e à arquitetura do FPGA.

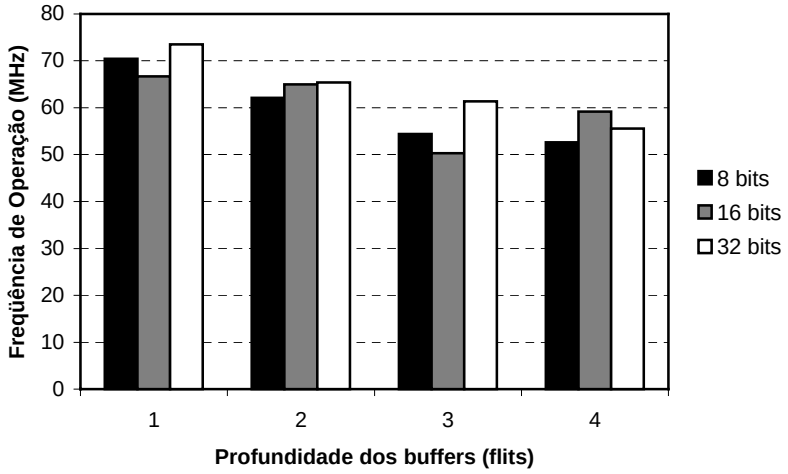


FIGURA 5.33 – Impacto da profundidade dos *buffers* no desempenho do roteador RASoC para diferentes configurações de largura de canal (8, 16 e 32 bits).

Avaliação da largura do campo RIB no custo e no desempenho do roteador

Na Figura 5.34, são apresentados resultados de área e freqüência para roteadores de 32 bits com duas posições nos *buffers* de entrada ($n = 32$, $p = 2$). O parâmetro m foi variado de 6 a 12 bits, o que permitiria a construção de redes com 8×8 a 32×32 roteadores. Como pode ser observado, na medida em que a largura do campo RIB é aumentada, o custo do roteador (gráfico de linha com eixo das ordenadas à direita) também cresce e sua freqüência de operação (gráfico de colunas com eixo das ordenadas à esquerda) diminui, pois o caminho crítico do circuito de atualização do cabeçalho torna-se maior.

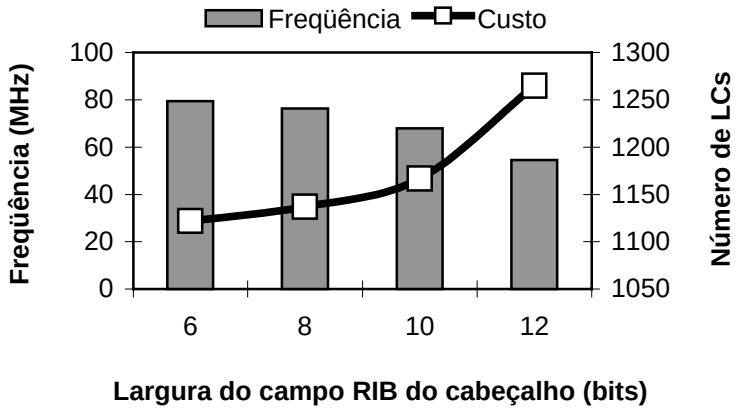
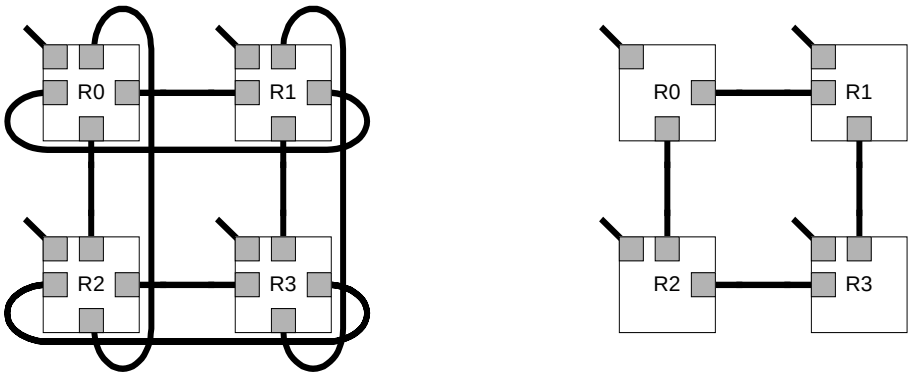


FIGURA 5.34 – Impacto da largura do campo RIB do cabeçalho no custo e no desempenho de roteadores de 32 bits com *buffers* de profundidade igual a dois.

5.13.2 Síntese de redes SoCIN

Nesta subseção são apresentados alguns resultados a respeito da síntese de duas redes SoCIN 2×2, uma com topologia toroidal e outra com topologia em grelha. O objetivo básico é comparar a diferença entre custo e a freqüência de operação de redes que implementam todas as portas do roteador (toróide) ou apenas parte dessas portas (grelha), como é ilustrado na Figura 5.35. Na grelha 2-D, cada roteador implementa apenas três portas, sendo que a geração do roteador com menos portas é realizada automaticamente pelo compilador. Contudo, na descrição VHDL, é preciso "aterrar" as entradas dos canais que não serão sintetizados, mapeando-se zeros à essas entradas na interface do roteador. Além disso, deve-se conectar as saídas não utilizadas a sinais que não são conectados a nenhuma entrada de outro roteador.



(a) (b)

FIGURA 5.35 – Redes 2∝2: (a) toróide; (b) grelha.

Na Figura 5.36, são apresentados os resultados de síntese dessas redes para configurações com *m* e *p* fixos (iguais a 8 bits e a 2 flits, respectivamente) e *n* variável (8, 16 e 32 bits). No eixo da ordenadas à esquerda, são mostrados os custos em LCs para a síntese do toróide 2∝2 e da grelha 2∝2 no FPGA EPF10K200SFC672-1 [ALT 2003a]. No eixo da ordenadas à direitas, apresenta-se redução pecentual de área proporcionada pela grelha calculada pela expressão indicada na legenda. Observa-se que para ambas as configurações, a grelha apresentou uma redução percentual significativa (de 49,4 a 52,7%). Contudo, com o aumento das dimensões das redes, essa redução será menor, pois a quantidade de roteadores internos, que implementam todas as portas, cresce quadraticamente com o tamanho da rede, enquanto que a quantidade de roteadores de periferia, os quais, na grelha, não implementam todas as portas, cresce linearmente.

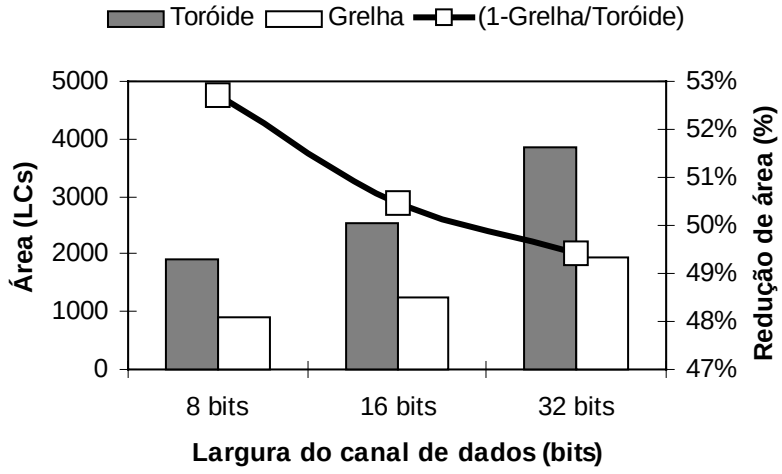


FIGURA 5.36 – Comparativo da área das redes toróide 2∝2 e grelha 2∝2.

Quanto à freqüência de operação, os resultados obtidos mostram que, no toróide, o enlace entre os roteadores da periferia produz uma perda de desempenho importante. Por exemplo, para as configurações de 8 bits, a freqüência máxima de operação da grelha (51,3 MHz) é três vezes maior que a freqüência do toróide (16,4 MHz).

Também foi considerado o impacto da inclusão de núcleos de processamento na síntese das redes de modo a avaliar se os ganhos percebidos anterioremente eram reproduzidos. Nesses experimentos, foram modelados SoCs baseados nas redes apresentadas acima e em instâncias do modelo de gerador de tráfego descrito anteriormente. Configurando-se os parâmetros dessas instâncias, foi possível gerar núcleos de processamento de diferentes tamanhos e SoCs homogêneos e heterogêneos. Em um SoC homogêneo, todas as instâncias do gerador de tráfego ocupam, aproximadamente, o mesmo número de LCs, enquanto que, nos SoCs heterogêneos, cada instância ocupa um número diferente de LCs. Foi tomado, como referência, o custo do roteador da grelha para uma configuração com n = 8 bits, m = 8 bits e p = 2 flits, e foram criadas instâncias do gerador de tráfego com área aproximadamente igual a 1∝, 2∝, 4∝ e 8∝ esse valor. Os resultados obtidos mostraram que a relação entre as áreas dos SoCs baseados na grelha e no toróide é aproximadamente a mesma, variando de 54,4% a 54,8% nos diferentes tipos de sistema (homogêneo ou heterogêneo) e tamanhos de instâncias (1∝, 2∝, 4∝ e 8∝), para um mesmo valor de n.

5.14 Considerações

Neste capítulo, foi apresentada a arquitetura da rede SoCIN e do seu bloco construtivo básico o roteador RASoC, assim como considerações a respeito da modelagem em VHDL, validação por simulação e síntese em FPGA. Embora o texto deste capítulo não tenha apresentado os detalhes a respeito dos componentes do roteador RASoC, é importante destacar que os mesmos baseiam-se em arquiteturas que almejam atender ou a requisitos de baixo custo ou a requisitos de melhor desempenho, ou a ambos. Contudo, a modelagem dos componentes foi orientada pelas limitações impostas pela arquitetura da tecnologia disponível (FPGA).

Quanto ao controle de fluxo, a técnica utilizada (*handshake*) está entre as de menor custo. Além disso, a modularidade dos blocos IFC e OFC permite a substituição facilitada dessa técnica por uma outra que implemente um protocolo de controle de fluxo baseado em dois fios, como, por exemplo, a baseada em créditos.

Quanto ao roteamento, a estratégia utilizada (roteamento XY) é destacada na literatura como aquela que melhor permite a obtenção de roteadores de baixo custo e redes totalmente livres de *deadlock*. Contudo, destaca-se que ela limita a utilização dos canais da rede. Alternativamente, poderiam ter sido utilizadas técnicas de roteamento baseadas no modelo de volta, por exemplo. Contudo, procurou-se limitar a exploração do espaço de projeto no escopo desta tese de modo a permitir a realização de outros estudos, como os que serão apresentados nos capítulos a seguir.

No que tange à arbitragem, a arquitetura de árbitro *round-robin* utilizada não está entre as de menor custo, mas é uma das que oferece o menor caminho crítico e, portanto, propicia a realização de árbitros e roteadores mais rápidos. Outras alternativas menos custosas foram estudas e modeladas. Contudo, optou-se por uma arquitetura mais rápida, pois o gargalo no desempenho de um roteador está justamente em seu árbitro.

Com relação ao chaveamento, a técnica de chaveamento por pacotes do tipo *wormhole* é a mais utilizada em redes de interconexão de computadores paralelos por propiciar a construção de roteadores de baixo custo. Outras técnicas, como o

chaveamento por circuito, também apresentam um baixo custo de implementação, mas o desempenho da rede é inferior quando as mensagens são curtas e freqüentes. Quanto à implementação dos elementos de chaveamento, a modelagem foi dirigida à síntese em FPGA, tecnologia disponível para validação do modelo. Porém procurou-se oferecer uma portabilidade facilitada para outras tecnologias de fabricação.

Quanto à estratégia de memorização adotada, o uso de *buffers* FIFO nos canais de entrada é a abordagem mais utilizada em redes de interconexão de computadores paralelos. Outras alternativas, como memorização centralizada ou de saída, também podem ser utilizadas para melhorar o desempenho, mas o custo do roteador é aumentado. Da mesma forma, o particionamento do *buffer* de entrada em múltiplas filas reduz o problema do bloqueio de cabeça de linha, mas também encarece o roteador. Conforme é discutido na literatura [VAI 2001], tais soluções são válidas somente se a aplicação demandar um desempenho superior ao oferecido com uma abordagem convencional. Com relação à arquitetura de *buffer* FIFO utilizada, destaca-se que seu custo cresce de maneira indesejada com a profundidade do *buffer*. Uma alternativa adequada, no caso de uma implementação *full-custom*, por exemplo, seria uma implementação baseada em células de memória SRAM, cujo custo é bem inferior.

Com relação à validação do modelo, foram mostrados diagramas de formas de onda extraídos de simulações que auxiliaram na confirmação da sua correção. As análises de custo e desempenho também permitiram a obtenção de informações a respeito da viabilidade da aplicação desse modelo na síntese de redes-em-chip, sobretudo integradas em FPGA. Dado o custo de implementação de algumas estruturas particulares do roteador (multiplexadores e *buffers*), acredita-se que sua aplicação nos FPGAs atuais se restrinja a redes baseadas em palavras de largura igual a 8 bits ou 16 bits (no máximo). Ainda, com relação à profundidade do *buffer*, estima-se que a mesma deva ser limitada a duas palavras tanto por causa do custo como pelo desempenho do roteador. Porém, é importante lembrar que *buffers* menos profundos provocam uma contenção maior na rede quando os pacotes são maiores que a profundidade do *buffer*. Por exemplo, um pacote bloqueado que tenha um tamanho igual a quatro vezes a profundidade dos *buffers* irá bloquear três canais de comunicação da rede. Entretanto, com a evolução dos processos de fabricação e o aumento da densidade lógica dos FPGAs, acredita-se que poderão ser sintetizadas redes com configurações maiores.

A rede SoCIN tem sido utilizada como rede de referência para diversas atividades de pesquisa do Grupo de Microeletrônica (GME) da UFRGS do grupo de Concepção de Sistemas Embarcados e Distribuídos (CSED) da Universidade do Vale do Itajaí (UNIVALI). Alguns trabalhos já foram publicados tendo como base a arquitetura da rede SoCIN e sua descrição VHDL. Em [ZEF 2002] e [ZEF 2002a] foram apresentados modelos analíticos para estimativa de desempenho de redes-em-chip, sendo que uma das redes consideradas foi a rede SoCIN. Em [COT 2003], a arquitetura da rede SoCIN serviu de base para a realização de experimentos de validação de uma proposta de reuso da rede-em-chip de um sistema integrado para a realização do teste de seus núcleos. Em [BEC 2003], a arquitetura SoCIN foi utilizada em um estudo comparativo do desempenho e do consumo de energia de um barramento e de duas redes-em-chip (toróide e árvore-gorda). Em [CAS 2002], é realizada uma análise sobre aplicações mapeadas para um sistema integrado baseado na arquitetura da rede SoCIN. Em [COR 2003], foram discutidos métodos para o teste de adaptadores de comunicação para um sistema integrado baseado na rede SoCIN. Em [ESP 2002], foram descritas e comparadas duas alternativas de implementação do árbitro *round-robin* do roteador

RASoC, enquanto que, em [ESP 2003], são descritos modelos parametrizáveis de árbitros distribuídos de baixo custo a serem utilizados em versões de mais baixo custo do roteador RASoC. Por fim, em [ZEF 2003], é feito o detalhamento da arquitetura da rede SoCIN e de seu roteador.

Nos capítulos a seguir, serão apresentados modelos analíticos desenvolvidos com o objetivo de oferecer estimativas de alto nível sobre o custo e o desempenho de arquiteturas de comunicação para sistemas integrados.

6 Modelos Analíticos para a Estimativa da Área de Arquiteturas de Comunicação para Sistemas Integrados

Este capítulo apresenta um conjunto de modelos analíticos que foram desenvolvidos para propiciar estimativas de alto nível para avaliação e comparação do custo de arquiteturas de comunicação para sistemas integrados. Esses modelos visam, especialmente, a comparação entre uma arquitetura de referência em barramento e duas arquiteturas de redes-em-chip de modo a determinar para qual tamanho de sistema cada rede apresenta um custo dentro de um limite de sobrecarga de área definido. Primeiramente, são apresentadas as arquiteturas de comunicação e considerações gerais a respeito da avaliação do custo de redes-em-chip. Após, são apresentados os modelos de Langen, Brinkmann e Rückert [LAN 2000] para a estimativa de alto nível da área de um barramento central e de um crossbar central. Esses modelos são avaliados e, a partir deles, são propostas extensões que permitem a modelagem do número equivalente de portas lógicas dos circuitos que compõem a parte operativa de um barramento típico para sistemas integrados (PI-Bus) e de duas redes-em-chip (SPIN e SoCIN). Esses modelos são aplicados para a avaliação da sobrecarga de área de silício para diferentes configurações de sistema, nas quais são variados os tamanhos dos núcleos do sistema, o número de núcleos e a largura da palavra de dados dos canais de comunicação. Por fim esses modelos são comparados para configurações de sistemas baseadas em palavras de 32 bits.

6.1 Arquiteturas de Comunicação

As arquiteturas de comunicação consideradas neste trabalho foram um barramento central e duas redes-em-chip, uma com topologia indireta e outra com topologia direta. O barramento central é baseado nas características da arquitetura do barramento PI-Bus [SIE 94]. Ele possui vias de dado e de endereço demultiplexadas, cada uma com largura de 32 bits e suporte a múltiplos mestres, sendo que o controle do acesso é realizado pela unidade de controle do barramento (BCU – *Bus Controller Unit*), a qual também realiza a decodificação de endereços e a seleção do escravo alvo de cada transação.

A rede-em-chip com topologia indireta é baseada na rede SPIN [GUE 2000], a qual foi descrita no Capítulo 4. Ela possui topologia em árvore-gorda quaternária e canais bidirecionais de 32 bits, nos quais as palavras de endereço e de dado de 32 bits devem ser multiplexadas no tempo. A rede SPIN é baseada no modelo de comunicação de troca de mensagens e utiliza chaveamento por pacotes do tipo *wormhole* e controle de fluxo baseado em créditos. Os núcleos conectados à rede comunicam-se enviando e recebendo pacotes de requisição e de resposta, sendo que cada pacote é constituído por um cabeçalho e uma seqüência de palavras de 32 bits. A rede SPIN utiliza roteamento adaptativo, o qual permite que pacotes trocados entre um mesmo par emissor-receptor utilizem caminhos diferentes na rede de modo a se adaptarem ao tráfego da mesma.

A rede direta é baseada na rede SoCIN, a qual foi descrita no Capítulo 5. Ela possui canais de largura parametrizável (eg. 8, 16, 32 bits), utiliza o modelo de

comunicação de troca de mensagens, chaveamento por pacotes do tipo *wormhole* e controle de fluxo baseado no protocolo *handshake*. O roteamento é determinístico e baseia-se no ordenamento por dimensão. Um pacote é primeiramente roteado em uma linha na direção X e, somente após atingir a mesma coluna do destinatário, ele pode ser encaminhado na direção Y. Uma vez tomada essa direção o pacote não pode mais ser roteado na direção X. A limitação dessa técnica de roteamento é que, embora existam múltiplos caminhos entre um par fonte-destinatário, apenas um pequeno subconjunto desses caminhos pode ser efetivamente utilizado. Isso reduz em muito a utilização da rede, mas, pelo fato de impedir o surgimento de ciclos, garante a ausência de *deadlock* a um baixo custo.

6.2 Avaliação do Custo das Redes-em-Chip

Segundo Guerrier e Greiner [GUE 2000a], o custo em silício de um barramento central é mínimo se comparado ao custo de uma rede-em-chip. Isso se deve ao fato de que a lógica associada ao barramento é constituída apenas por um árbitro que seleciona um entre múltiplos mestres passíveis de iniciar uma comunicação, pelos circuitos de decodificação de endereço para a seleção do escravo alvo de cada comunicação e pelos *buffers tri-state* que conectam e isolam os núcleos do barramento. Já em uma rede-em-chip, cada roteador é composto por *buffers* de memorização, multiplexadores de chaveamento, árbitros para seleção de entrada, decodificadores para a seleção de saída (circuitos de roteamento) e controladores de fluxo de entrada e de saída.

Em [HWA 93], Kai Hwang apresenta um quadro comparativo sobre os custos para implementação de diferentes redes de interconexão para computadores paralelos em termos da complexidade de chaveamento (*switching complexity*) e da complexidade de fiação (*wiring complexity*). A complexidade de chaveamento diz respeito aos custos relacionados aos circuitos lógicos necessários para a transferência das mensagens, enquanto que a complexidade de fiação reflete os custos associados aos fios que compõem os canais físicos de comunicação necessários à transferência dessas mensagens.

Quanto à complexidade de chaveamento, Hwang situa o barramento em um extremo e o crossbar em outro. Segundo ele, a complexidade do barramento cresce linearmente com o número de nodos (n) conectados ao barramento, ou seja, é $O(n)$. Já a complexidade de chaveamento do crossbar cresce quadradicamente com o número de nodos no sistema, ou seja, é $O(n^2)$. Entre esses dois extremos o autor coloca as redes de interconexão constituídas por arranjos de múltiplos roteadores baseados em crossbars de dimensões limitadas (tipicamente, 4 a 8 portas por roteador). Em uma rede multi-estágio, por exemplo, a complexidade de silício é $O(n \propto \log_k n)$, onde 2^k define o número de portas em cada roteador, enquanto que em um toróide 2-D é $O(n)$. Quanto à complexidade de fiação, Kai Hwang estima que, em um barramento, ela é $O(w)$, onde w define o número de fios paralelos que compõem os canais físicos de comunicação. Por outro lado, ele estima que a complexidade de fios de um crossbar é $O(w \propto n^2)$, enquanto que é $O(w \propto n \propto \log_k n)$ em uma rede multi-estágio e $O(w \propto n)$ em um toróide 2-D.

A complexidade de chaveamento de uma rede de interconexão para computadores paralelos pode ser entendida como custo de silício em uma arquitetura de comunicação intrachip (barramento ou rede-em-chip). Esse custo de silício pode ser

medido pelo número equivalente de portas lógicas necessárias à implementação dos circuitos lógicos associados à transferência de mensagens ou, então, pela própria área ocupada por essas portas lógicas. Já a complexidade de fiação pode ser vista como custo de metal nas arquiteturas intrachip. Ou seja, a área gasta na implementação dos canais físicos de comunicação, a qual é função do número de canais físicos, do comprimento dos canais físicos, do número de fios por canal e da largura desses fios.

Nas tecnologias atuais, a área de silício consumida na implementação da arquitetura de comunicação (ou sobrecarga[20]) implica em um custo adicional ao sistema integrado. Isso é devido ao fato de que os circuitos lógicos associados à arquitetura de comunicação são montados sobre a mesma camada do chip na qual são montados os circuitos lógicos relacionados aos núcleos do sistema. Portanto, é desejável que essa sobrecarga de silício seja mínima, sendo que, em [BRI 2002] é estabelecido um limite de máximo de 10%, o qual, segundo os autores, permite garantir a obtenção de um sistema bem balanceado. Contudo, deve-se considerar que cada sistema possui requisitos particulares e, embora neste trabalho também se utilize esse valor referência para fins comparativos, o limite de sobrecarga de área de silício associada à arquitetura de comunicação será específico ao sistema alvo.

Por outro lado, as tecnologias correntes dispõem de múltiplas camadas de metal (eg. 4 ou mais) e, tipicamente, a interconexão entre os núcleos é feita nas camadas de metal mais elevadas. Dessa forma, o roteamento dos fios associados aos canais de comunicação é feito sobre a área de silício devida aos núcleos e aos componentes da arquitetura de comunicação. Nessa condição, pode-se assumir que, na maioria dos casos, a área de metal associada à arquitetura de comunicação não implicará em uma sobrecarga na área do *die*.

Embora as dimensões dos canais físicos possam não ter impacto direto na área do chip, elas são de fundamental importância para o desempenho da comunicação. A freqüência de operação de um canal de comunicação depende da capacitância e da resistência dos seus fios e essas características físicas são função das dimensões desses fios (comprimento e largura).

Neste capítulo, o foco principal é dado à estimativa do impacto da arquitetura de comunicação no custo do sistema integrado. Dessa forma, nas seções a seguir, será considerado apenas o custo de silício devido a essas arquiteturas. Contudo, no Capítulo 7, será considerado o impacto das interconexões de metal no desempenho da da arquitetura de comunicação.

[20] O termo sobrecarga deriva do inglês *overhead* e, no contexto deste capítulo, indica a área adicional gasta com a arquitetura de comunicação em relação à área devida aos núcleos do sistema integrado.

6.3 Modelos de Referência para Estimativa de Área de Arquiteturas de Comunicação Intrachip

Em [LAN 2000], Langen, Brinkmann e Rückert apresentam modelos analíticos para a estimativa de alto nível da área e do consumo de energia de três tipos de arquiteturas de comunicação intrachip centralizadas: barramento, crossbar e multiplexador. Esses modelos são apresentados a seguir e servirão de referência para os modelos a serem desenvolvidos para a estimativa da sobrecarga de área das redes-em-chip[21]. Os modelos apresentados por Langen, Brinkmann e Rückert foram desenvolvidos assumindo-se as seguintes condições:

1. Foi modelada a infra-estrutura mínima necessária para habilitar a transferência de dados entre dois núcleos (ou módulos, na terminologia dos autores), de modo que os protocolos de controle de fluxo e arbitragem não são considerados nos modelos. Os autores assumem que o espaço de projeto para protocolos e árbitros é bastante amplo e que os mesmos podem ser considerados em uma segunda camada de modelagem;

2. Apenas os bits de endereço necessários para identificar todos os n núcleos são codificados/decodificados. Do número total de W fios paralelos em um canal físico, $\log_2 n$ são reservados ao transporte de endereços e W–$\log_2 n$ ao transporte de dados;

3. Os núcleos são quadrados e possuem o mesmo tamanho, sendo que o comprimento de cada aresta dos núcleos é referenciado pela letra B.

Feitas essas considerações, os autores expressam a área total consumida a partir da soma das áreas devidas às portas lógicas e aos canais físicos:

$$A_{total} = (A_{gate} \propto N_{gates}) + (L \propto width) \qquad (6.1)$$

onde A_{gate} é a área de uma porta lógica básica (eg. NAND com duas entradas), N_{gates} é o número equivalente de portas lógicas básicas, L é o comprimento do canal físico e $width$ é a largura de cada fio do canal físico.

Como pode ser observado, em (6.1), o modelo inclui a área de metal na estimativa da área total ocupada pela arquitetura de comunicação, o que pode ser atribuído ao fato de que os autores consideraram uma tecnologia 0.6 νm CMOS com duas camadas de metal. Contudo, conforme foi afirmado na seção anterior, os modelos desenvolvidos neste capítulo visam tecnologias com um número maior de camadas de metal, as quais permitem a sobreposição dos canais físicos da arquitetura de comunicação à área de silício ocupada pelos núcleos e pelos componentes da arquitetura. Apesar disso, nas subseções as seguir, as quais descrevem os modelos de referência, será reproduzida a modelagem completa efetuada em [LAN 2000] para oferecer uma visão clara sobre esses modelos.

[21] No decorrer deste capítulo, os modelos de área apresentados por Langen, Brinkmann e Rückert [LAN 2000] serão citados pelo termo “modelos de referência”.

6.3.1 Modelo para a estimativa da área do barramento central

Na modelagem do barramento, os autores assumem que os fios do barramento cruzam todo o chip e que seu comprimento total pode ser estimado como sendo igual à metade do perímetro do *die*. Desde que cada aresta do *die* é dada pelo somatório das arestas (B) dos $n^{1/2}$ núcleos posicionados em cada lado do chip, o comprimento total de fio para os W fios que compõem o barramento é definido por:

$$L_{bus} = W \propto 2 \propto B \propto n^{1/2} \tag{6.2}$$

Com relação ao número de portas lógicas, os autores consideram que cada um dos W fios do barramento pode ser alimentado por diferentes núcleos e é necessário incluir *buffers tri-state* em cada saída de núcleo. Além disso, cada núcleo requer um decodificador de endereços cujo custo é dado pelo parâmetro $N_{gates,addr}$. Assim, o número de portas lógicas do barramento é dado por:

$$N_{gates,bus} = (n \propto W) + N_{gates,addr} \tag{6.3}$$

onde $N_{gates,addr}$ é estimado em função do número de bits de endereço que devem ser decodificados. Os autores assumem que são necessárias $\log_2 n$ portas de duas entradas para comparar o endereço fornecido com o endereço do núcleo mais uma porta de saída para a ativação do sinal de seleção. Para um sistema com até 64 núcleos, o endereço tem no máximo 8 bits e as portas de comparação de endereço podem ser ligadas por uma única porta AND. Como existem n núcleos conectados ao barramento, o número total de portas consumidas pelos circuitos de decodificação de endereços é assumido como sendo dado por:

$$N_{gates,addr} = n \propto (\log_2 n + 1) \tag{6.4}$$

Associando-se (6.1) a (6.4), obtém-se o modelo que estima o custo total do barramento:

$$A_{total,bus} = A_{gate} \propto \{(n \propto W) + [n \propto (\log_2 n + 1)]\} \quad + \quad (W \propto 2 \propto B \propto n^{1/2} \propto width) \tag{6.5}$$

6.3.2 Modelo para a estimativa da área do crossbar central

Enquanto que no modelo do barramento os *buffers tri-state* e os decodificadores são distribuídos entre os núcleos conectados ao barramento, no modelo do crossbar, essas estruturas são centralizadas em um único componente. Esse componente é denominado distribuidor e o modelo considera que os núcleos do sistema são posicionados ao redor do distribuidor. Langen, Brinkmann e Rückert transformam a área total do chip em uma área circular, a qual é dividida em duas partes iguais por um círculo com o raio médio do crossbar dado por:

$$\theta \propto r_{av}^{\,2} = 0{,}5 \propto n \propto B^{\,2} \tag{6.6}$$

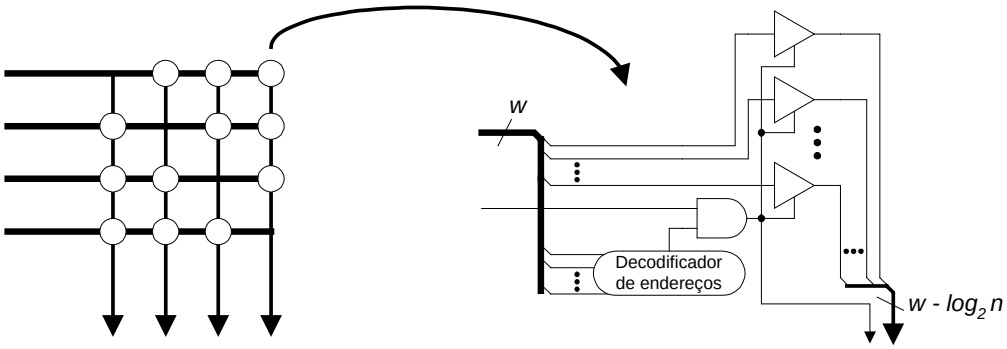
da qual se obtém:

$$r_{av} = (n^{1/2} \propto B) \propto (2 \propto \theta)^{1/2} \tag{6.7}$$

Para cada núcleo do sistema, o distribuidor associa dois canais unidirecionais: um canal de entrada, composto por *W* fios (endereço e dado), e um canal de saída, composto por $W–\log_2 n$ fios (apenas para os dados). Além desses fios, os autores assumem a necessidade de um sinal adicional para cada núcleo na entrada do distribuidor, o qual é utilizado para indicar que o núcleo está escrevendo um dado, e um sinal de saída para indicar que o núcleo foi endereçado. Além disso, é assumido que nem sempre é possível rotear um canal diretamente do distribuidor para o núcleo. Sendo assim, para a tecnologia utilizada (0.6 νm CMOS, metal-duplo), é considerado um fator de multiplicação para o raio médio, o qual é igual a 2. Dessa forma, o comprimento total de fio para um sistema baseado no crossbar central é dado por:

$$L_{crossbar} = n \propto [(2 \propto W) – \log_2 n + 2] \propto 2 \propto r_{av} \quad (6.8)$$

O distribuidor é modelado por uma matriz na qual as linhas e as colunas são conectadas por *buffers tri-state*. É assumido que a saída de um núcleo não pode ser conectada à entrada desse núcleo. Sendo assim, como pode ser observado na Figura 6.1.a, os pontos de chaveamento que conectariam os canais de entrada e de saída de um mesmo núcleo não são implementados e, dessa forma, existem $(n – 1)^2$ pontos de chaveamento.



(a) (b)

FIGURA 6.1 – Modelo de crossbar: (a) matriz de chaveamento; (b) estrutura de um ponto de chaveamento.

Como pode ser observado na Figura 6.1.b, cada ponto de chaveamento é constituído por $W–\log_2 n$ *buffers tri-state*. Além disso, os *buffers tri-state* de um mesmo ponto de chaveamento devem ter um sinal de habilitação comum, o qual depende da ativação simultânea da saída de seleção do decodificador de endereços associado ao canal de entrada e do sinal de indicação de escrita daquele canal de entrada. Essa função de habilitação é implementada por uma porta lógica AND e, dessa forma, existe uma porta adicional de controle para cada um dos $(n – 1)^2$ pontos de chaveamento do crossbar. Por fim, é necessário associar um decodificador de endereços a cada ponto de chaveamento para indicar se aquela conexão está sendo requerida. A partir disso, a equação que determina o número de portas lógicas no crossbar é:

$$N_{gates,crossbar} = [(n – 1)^2 \propto (W – \log_2 n + 1)] \quad + \quad n \propto N_{gates,addr} \quad (6.9)$$

ou, então:

$$N_{gates,crossbar} = [(n – 1)^2 \propto (W – \log_2 n + 1)] \quad + \quad [n^2 \propto (\log_2 n + 1)] \quad (6.10)$$

na qual é possível se observar claramente que o custo de silício do crossbar cresce quadradicamente com o número de núcleos a ele conectados. A área total do crossbar é então dada por:

$$A_{total,crossbar} = A_{gate} \propto \{[(n-1)^2 \propto (W - \log_2 n + 1)] \quad + \quad [n^2 \propto (\log_2 n + 1)]\} \qquad (6.11)$$
$$+ \{n \propto [(2 \propto W) - \log_2 n + 2] \propto 2 \propto r_{av}\} \propto width$$

6.3.3 Avaliação dos modelos de referência para a estimativa de área

Em [LAN 2000], os modelos de área para o barramento e para o crossbar foram comparados com resultados da síntese dessas arquiteturas, descritas em VHDL, utilizando uma tecnologia 0.6 νm CMOS com duas camadas de metal. Foram avaliados sistemas baseados em diferentes tamanhos de núcleos, com mil, dez mil e cem mil portas lógicas, sendo que todos os núcleos possuíam portas com largura de 16 bits. O erro médio dos valores estimados de área em relação aos valores obtidos com a síntese foi de 25% após uma etapa de calibração. Essa calibração foi realizada para incluir *drivers* não modelados pelas equações de área e potência.

Uma limitação dos modelos de referências é que seus autores consideraram que as diferentes portas e *buffers tri-state* possuem o mesmo custo de uma porta lógica NAND de duas entradas. Essa simplificação inclui um erro adicional à estimativa que pode ser evitado utilizando-se fatores de multiplicação que expressem a relação entre a área da porta considerada e a área de uma porta NAND para uma tecnologia alvo. Por exemplo, na Tabela 6.1, são listadas as áreas de algumas portas utilizadas nesses modelos com base na biblioteca de células desenvolvida pela AMS (austriamicrosystems) para a sua tecnologia 0.35 νm CMOS [AUS 2001a]. As células apresentadas possuem dimensões mínimas, o que é indicado pelo termo (1∝). A tabela também inclui o número equivalente de portas NAND com duas entradas para cada uma das células. Como pode ser observado, o custo de um *buffer tri-state* é 2,7 vezes maior que o de uma porta NAND com duas entradas.

TABELA 6.1 – Exemplos de células da biblioteca AMS 0.35 νm CMOS [AUS 2001a].

Célula	**Descrição**	**Área (νm²)**	**Qtd. NA2**
NA2	Porta NAND com 2 entradas (1∝)	55	1,0
AND2	Porta AND com 2 entradas (1∝)	73	1,3
AND8	Porta AND com 8 entradas (1∝)	237	4,3
BT1	*Buffer tri-state* com habilitação em nível alto (1∝)	146	2,7

Uma outra limitação dos modelos diz respeito ao fato de que, ao ignorarem os *buffers* que reforçam o sinal injetado nos fios, os autores não consideraram que, em um barramento, são necessários *buffers* maiores que no crossbar devido à maior capacitância dos fios que compõem os canais do barramento, que são mais longos que os do crossbar. Em [BEC 2003], Beck Filho e Carro efetuam a comparação de diferentes topologias de arquiteturas de comunicação para SoC. Segundo os autores, os transistores dos *buffers* de um núcleo conectado a um barramento devem ter um canal quatro vezes mais largo que os dos transistores dos *buffers* de um núcleo conectado a uma rede-em-chip para que os canais do barramento possam operar na mesma

freqüência que os canais da rede-em-chip. Se isso fosse levado em conta nos modelos apresentados em [LAN 2000], a sobrecarga de área do barramento seria maior do que a modelada.

Desde que, no modelo de área (6.1), os autores consideram a área de metal no cálculo da área total do chip, eles incorrem no erro de assumir que os canais do barramento e do crossbar utilizam fios com a mesma largura. Tipicamente, para que os canais do barramento possam atingir freqüências de operação do mesmo nível dos canais ponto-a-ponto do crossbar (e de uma rede-em-chip) é preciso reduzir a resistência dos fios através do aumento da largura dos mesmos. Conforme os resultados apresentados em [BEC 2003], com uma largura de linha de 4 νm (8 vezes maior que o valor mínimo), o período de operação do barramento chega próximo a 2 ns quando se utilizam *buffers* com transistores PMOS de largura de canal superior a 70 νm. Esse mesmo período de operação é obtido nos canais de uma rede-em-chip utilizando fios de largura mínima (0.5 νm) e *buffers* com transistores PMOS de largura de canal inferior a 18 νm. Além disso, para a uma configuração de largura de linha de 4 νm e largura de canal do transistor PMOS superior a 70 νm, os canais da NoC atingiriam um período de operação de 0,75 ns, aproximadamente.

Outra limitação dos modelos está na simplificação do número de fios utilizados para o endereçamento dos núcleos. Em geral, os barramentos re-utilizáveis destinados a interconexão em SoCs possuem vias de endereço e dado demultiplexadas, sendo que a via de endereços implementa tantos fios quanto o número de bits associados ao espaço de endereçamento (eg. 32 bis no PI-Bus).

6.3.4 Considerações para a extensão dos modelos de referência para arquiteturas de comunicação atuais

Em [BRI 2002], Brinkmann et al. estendem o modelo por eles apresentados em [LAN 2000] a fim de modelar a área ocupada por uma arquitetura de comunicação intrachip por eles proposta. Essa arquitetura possui estrutura baseada em uma hierarquia com dois níveis de rede. No nível mais baixo, os núcleos são interconectados por arquiteturas centralizadas, como o barramento o crossbar ou o multiplexador. No nível mais alto, as arquiteturas centralizadas são interligadas por roteadores distribuídos, organizados em uma topologia arbitrária (eg. malha, toróide, multiestágio etc). O modelo de comunicação adotado é da troca de mensagens com chaveamento por pacotes do tipo armazena-e-repassa (SAF – *Store and Forward*). Além disso, em cada roteador, a seleção do caminho a ser tomado por cada pacote é realizada através da consulta a uma tabela de roteamento. Os *buffers* de entrada tem capacidade de armazenar pacotes completos, sendo que, em uma configuração avaliada, os autores consideraram que cada roteador deveria ser capaz de armazenar até 80 pacotes com 1 Kbit cada (ou seja, 10 KBytes de memorização por roteador). Ainda, quando um pacote chega a um roteador e não há condições para o seu encaminhamento ou armazenamento, o pacote deve ser descartado.

Como pode ser observado acima, a arquitetura proposta em [BRI 2002] baseia-se em características arquiteturais de roteadores para redes de computadores. Contudo, as arquiteturas de rede-em-chip desenvolvidas atualmente baseiam-se nos conceitos e nas características de redes de interconexão para computadores paralelos. Tipicamente, a técnica de chaveamento utilizada é a *wormhole*, o roteamento é do tipo

fonte e os *buffers* tem capacidade de armazenar poucos flits de um pacote, entre outras características que visam a obtenção de roteadores rápidos, de baixo custo e com consumo reduzido de energia. Logo, o modelo desenvolvido para a estimativa de área do roteador dessa rede tem um escopo limitado e não considera características arquiteturais de redes-em-chip atuais, como a SPIN, a Octagon, a SoCIN e a CLICHÉ, entre outras.

Disso e a partir da análise das limitações dos modelos apresentados em [LAN 2000], propõe-se, a seguir, uma extensão desses modelos visando propiciar uma estimativa da sobrecarga de silício de arquiteturas de comunicação para a integração de SoC atuais e das próximas gerações, as quais, são baseadas em redes de interconexão de computadores paralelos. Destaca-se que os modelos aqui propostos, não se baseiam na extensão apresentada em [BRI 2002], mas sim em uma análise crítica das limitações do modelo original e das características das arquiteturas de comunicação alvo. Nesse contexto, serão apresentados modelos para um barramento de referência similar ao PI-Bus e para as redes SPIN e SoCIN.

Algumas considerações realizadas em [LAN 2000] são mantidas para ambos os modelos, porém, outras considerações são introduzidas:

1. É modelada a infra-estrutura mínima necessária para habilitar a transferência de dados entre dois núcleos (ou mais núcleos) e, portanto, os protocolos de controle de fluxo e de arbitragem também não são levados em conta;

2. Os *buffers* de sinal (ou *drivers*) dos núcleos conectados a um barramento possuem canais 4 vezes mais largos que os dos *buffers* de sinal dos núcleos conectados a uma rede chaveada;

3. A via de endereços do barramento inclui todos os bits correspondentes ao espaço de endereçamento e possui a mesma largura da via de dados;

4. No barramento, a decodificação de endereços para a seleção do escravo de cada comunicação é feita considerando-se apenas os bits correspondentes ao número de núcleos escravos no sistema. No limite máximo, esse número será igual a n–1 se for considerado que o sistema possui apenas um núcleo mestre. Contudo, da mesma forma que em [LAN 2000], assume-se que todos os núcleos do sistema podem operar como mestre ou escravo do barramento e, portanto, o número de bits necessários à decodificação de endereço e seleção de escravos será igual a $\log_2 n$.

5. Nas redes-em-chip, assume-se que, para cada ponto de chaveamento no crossbar, existe um circuito de decodificação de endereços e que a largura da palavra de endereço a ser decodificada é definida pela capacidade de endereçamento da rede (eg. 8 bits para 256 núcleos).

6. No barramento, a decodificação de endereços para seleção dos escravos é feita pelo controlador do barramento e não pelos próprios núcleos. Assim, deve existir uma linha de seleção entre o controlador e cada núcleo escravo do sistema. Desde que foi assumido que todos os núcleos podem ser mestre ou escravo, o número de linhas de seleção será igual a n;

7. Para simplificar a modelagem, também se considera que os núcleos são quadrados e possuem o mesmo tamanho, sendo que o tamanho de um lado do quadrado é referenciado pela letra *B*. Contudo, destaca-se que os núcleos de um sistema integrado são tipicamente heterogêneos.

8. Considera-se o custo dos diferentes tipos de portas lógicas utilizadas a partir do número equivalente de portas lógicas NAND para uma tecnologia alvo.

Os modelos serão baseados na tecnologia AMS 0.35 νm CMOS [AUS 2001], a qual apresenta variações com três e quatro camadas de metal. A partir das informações contidas no manual sobre a biblioteca de células correspondente a essa tecnologia [AUS 2001a], é determinado o número equivalente de portas lógicas NAND para cada tipo de célula. A célula de referência corresponde à porta NAND de duas entradas com dimensões mínimas, a qual é denominada NA2. Na Tabela 6.2, são listados alguns exemplos de células da biblioteca referida com a área correspondente em νm^2 e o número equivalente de células NA2. As células com dimensões mínimas são indicadas pelo termo (1∝). As células com capacidade maior de injetar corrente (*strength*) são indicadas pelos termos (2∝), (3∝), (4∝) e (8∝).

TABELA 6.2 – Células da biblioteca da AMS 0.35 νm CMOS [AUS 2001a].

Célula	**Descrição**	**Área (νm^2)**	**Qtd. NA2**
NA2	Porta NAND com 2 entradas (1∝)	55	1,0
AND2	Porta AND com 2 entradas (1∝)	73	1,3
AND8	Porta AND com 8 entradas (1∝)	237	4,3
BT1	*Buffer tri-state* com habilitação em nível alto (1∝)	146	2,7
BT14	*Buffer tri-state* com habilitação em nível alto (4∝)	164	3,0
BU1	*Buffer* (1∝)	73	1,3
BU2	*Buffer* (2∝)	73	1,3
BU3	*Buffer* (3∝)	91	1,7
BU4	*Buffer* (4∝)	91	1,7
BU8	*Buffer* (8∝)	127	2,3
DF8	Flip-flop tipo D (1∝)	273	5,0
IMU2	Multiplexador 2∝1 com saída invertida (1∝)	109	2,0
MU2	Multiplexador 2∝1 (1∝)	127	2,3
MU4	Multiplexador 4∝1 (1∝)	273	5,0
MU8	Multiplexador 8∝1 (1∝)	619	11,3
NO2	Porta NOR com 2 entradas (1∝)	55	1,0
IN1	Porta NOT (1∝)	36	0,7
OR2	Porta OR com 2 entradas (1∝)	73	1,3
EO1	Porta XOR com 2 entradas (1∝)	146	2,7

6.4 Modelos para Estimativa da Área de Silício de Barramentos Similares ao PI-Bus

Assumindo-se um barramento com características similares às do PI-Bus [SIE 94] e levando-se em conta as considerações apresentadas na seção anterior, desenvolve-se, a seguir, um modelo para estimativa da área ocupada por um barramento centralizado.

Quanto ao número de portas lógicas dos decodificadores de endereços, assume-se que cada decodificador possui uma estrutura baseada em um arranjo de portas lógicas NOT-NOR [RAB 96]. Considera-se que, na média, cada decodificador utiliza $(\log_2 n)/2$ portas lógicas NOT, pois existem $\log_2 n$ bits de endereço e, no espaço de endereçamento correspondente, cada um dos $\log_2 n$ bits assume valor 0 para metade dos endereços e 1 para a outra metade dos endereços. Como cada célula correspondente à porta lógica NOT possui um custo de 0,7∝NA2, o custo devido a essas portas lógicas é dado por 0,35∝$(\log_2 n)$∝NA2 em cada decodificador. Ainda, a porta NOR de saída possui $\log_2 n$ entradas e seu custo é estimado por $[1 + 0{,}73 \propto (\log_2 n - 2)]$∝NA2. Essa expressão computa o custo das entradas adicionais em relação à porta lógica NOR com duas entradas para cada tamanho de decodificador. O erro médio da estimativa de área da porta lógica NOR com $\log_2 n$ entradas dada por essa expressão é de 7,53%, se forem consideradas portas com 3 a 8 entradas. Dessa forma, o custo total devido aos decodificadores associados aos n núcleos do sistema será:

$$N_{gates,addr} = n \propto [0{,}35 \propto \log_2 n \; + \; 1 + \; 0{,}73 \propto (\log_2 n - 2)] \tag{6.12}$$

ou

$$N_{gates,addr} = n \propto (1{,}08 \propto \log_2 n - 0{,}46) \tag{6.13}$$

Na Tabela 6.3, aplica-se (6.13) para determinar as quantidades equivalentes de células NA2 e as áreas correspondentes em mm^2 para os decodificadores em sistemas de diferentes tamanho. A área da célula NA2 é igual a 55 νm^2

TABELA 6.3 – Custos dos decodificadores para diferentes tamanhos de sistema.

n	$N_{gates,addr}$	A_{addr} (mm^2)
4	6,8	0,0004
8	22,2	0,0012
16	61,8	0,0034
32	158,1	0,0087

A quantidade de *buffers tri-state* no canal de saída da porta de comunicação de cada núcleo é dado por duas vezes a largura da via de dados (2∝W_{data}). Desde que foi assumido que os *buffers* posuem uma capacidade de corrente quatro vezes maior que a dos *buffers* de dimensões mínimas, é considerado o uso da célula BT14, a qual equivale a três células NA2. Dessa forma, o custo de *buffers tri-state* para o barramento é igual a 6∝W_{data}∝NA2 para cada núcleo do sistema, enquanto que o número equivalente de portas lógicas do barramento será:

$$N_{gates,Bus} = (n \propto 6 \propto W_{data}) + [n \propto (1{,}08 \propto \log_2 n - 0{,}46)] \quad (6.14)$$

na qual o primeiro termo define o número de *buffers tri-state*, enquanto que o segundo computa o custo dos decodificadores de endereços. Disso, o modelo simplificado de área de silício para o barramento é dado por:

$$A_{total,Bus} = A_{gate} \propto \{(n \propto 6 \propto W_{data}) + [n \propto (1{,}08 \propto \log_2 n - 0{,}46)]\} \quad (6.15)$$

Esse modelo é aplicado a seguir para avaliar a sobrecarga de área de silício do barramento em sistemas baseados em núcleos com dez mil, cinqüenta mil e cem mil portas lógicas, dimensões tipicamente encontradas nos núcleos dos sistemas integrados atuais e das próximas gerações. São apresentados resultados para diferentes larguras de vias de dado, dos quais deve-se destacar os resultados correspondentes à largura de 32 bits, a qual é utilizada no barramento PI-Bus [SIE 94]. São destacados, também, os resultados cuja sobrecarga de área de silício é igual ou inferior a 10% da área de silício devida aos núcleos, a qual é dada pelo produto do número de núcleos no sistema pela área de cada núcleo. Esse percentual foi extraído de [BRI 2002], onde, segundo os autores, tal valor permitiria a obtenção de um sistema bem balanceado. Contudo, destaca-se que, neste texto, ele serve apenas como um valor de referência para as análises a serem realizadas, pois se entende que a definição da sobrecarga de área máxima aceitável para a arquitetura de comunicação varia de aplicação para aplicação, conforme os requisitos da aplicação alvo (eg. custo, desempenho, potência etc).

Na Figura 6.2, são mostrados os resultados para sistemas baseados em núcleos com 10 mil portas lógicas. Observa-se que, na maioria das configurações de tamanho de palavra de dado (W_{data}), a sobrecarga de área de silício é inferior a 10% da área devida aos núcleos do sistema. A exceção reside nos sistemas em que W_{data} = 256 bits. Contudo, destaca-se que essas configurações são irreais, pois, tipicamente, núcleos na faixa de 10 mil portas lógicas baseiam-se em palavras de dado de 8 ou 16 bits (eg. microcontrolador 8051). Observa-se, também, que com o aumento do número de núcleos, a sobrecarga percentual se mantém quase constante, pois a área devida ao barramento cresce na mesma proporção que a área devida aos núcleos. Contudo, aumentando-se apenas a largura do barramento, a sobrecarga percentual de área cresce na proporção desse aumento, pois o número de *buffers* necessários é maior, mas a área devida aos núcleos é mantida constante. Finalmente, destaca-se que a sobrecarga estimada para um barramento similar ao PI-Bus (W_{data} = 32 bits) é inferior a 2%.

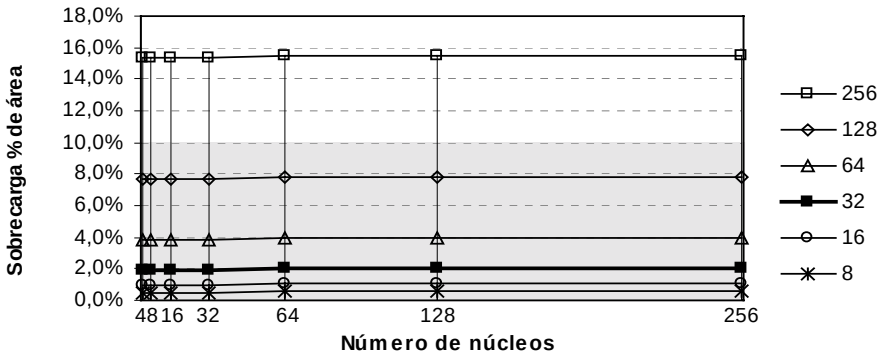


FIGURA 6.2 – Sobrecarga % de área de silício do barramento para núcleos com 10 mil portas lógicas (A_{core} = 0,55 mm^2).

Na Figura 6.3 e Figura 6.4, são mostrados os resultados para sistemas baseados em núcleos com 50 mil e 100 mil portas lógicas, respectivamente. Observa-se que, a sobrecarga de área de silício é inferior a 10%, para qualquer configuração, sendo próxima a zero para sistemas com 100 mil portas lógicas e palavra de dado de 32 bits, o que confirma a afirmação por Guerrier e Greiner [GUE 2000a] de que o custo de silício do barramento é quase nulo.

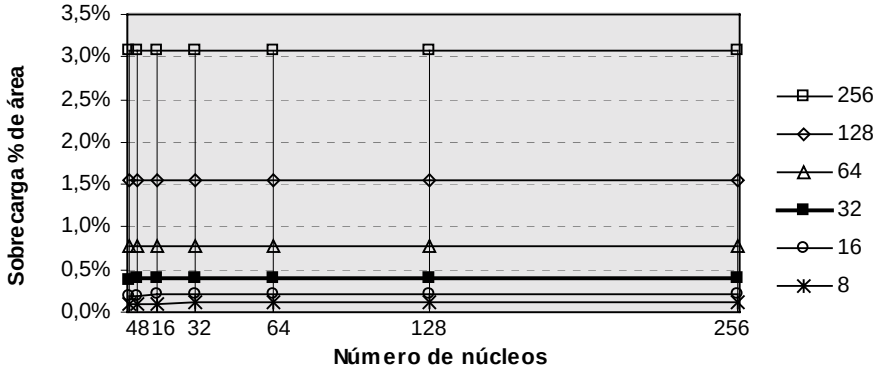


FIGURA 6.3 – Sobrecarga % de área de silício do barramento para núcleos com 50 mil portas lógicas (A_{core} = 2,75 mm^2).

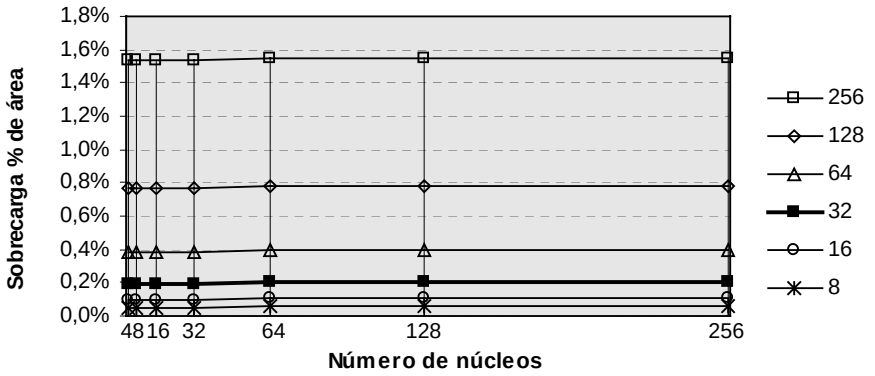


FIGURA 6.4 – Sobrecarga % de área de silício do barramento para núcleos com 100 mil portas lógicas (A_{core} = 5,50 mm^2).

Com relação aos sistemas baseados em núcleos com 100 mil portas lógicas e área de 5,50 mm^2, exemplificados na Figura 6.4, é importante destacar que alguns deles são irrealizáveis. Segundo manual de referência do CMP (Circuit Multi-Projets) [CIR 2003], na tecnologia AMS 0.35 νm CMOS, o tamanho máximo de *die* é igual a 2 cm^2, ou 200 mm^2. Isso comportaria sistemas integrando no máximo 36 núcleos com área de 5,50 mm^2. Além disso, a própria conexão de várias dezenas ou centenas de núcleos através de um barramento resultaria em um baixíssimo desempenho e um alto consumo de energia. As configurações com 64, 128 e 256 núcleos apresentadas nas figuras acima foram incluídas com o intuito de estabelecer uma referência para comparação com as redes-em-chip, as quais possuem maior escalabilidade, e, também, para ilustrar o impacto do aumento do número de núcleos na sobrecarga de área de silício do barramento.

6.5 Modelos para Estimativa da Área de Silício da Rede SPIN

Os modelos para estimativa de área da rede SPIN, apresentados a seguir, baseiam-se na arquitetura apresentada no Capítulo 4 e em informações sobre a implementação da rede contidas em [AND 2001]. Assumindo-se que as interconexões de metal entre os roteadores da rede e entre os núcleos e os roteadores são realizadas nas camadas de metal mais elevadas, então, em uma abordagem simplificada, considera-se apenas a área de silício da rede para a avaliação do seu impacto na área do sistema. Primeiramente, desenvolve-se uma equação para a estimativa do número equivalente de portas lógicas no roteador RSPIN, após, obtém-se uma equação que determina o número de roteadores em diferentes configurações de rede. Essas duas equações são então associadas para a definição do modelo de estimativa de área da rede SPIN.

6.5.1 Modelo para a estimativa do número de portas lógicas no roteador RSPIN

O roteador RSPIN, cuja arquitetura foi apresentada no Capítulo 4, é constituído, basicamente, por um crossbar 10∝10 parcial, uma coleção de *buffers* de armazenamento e controladores que comandam a operação de cada recurso do roteador (crossbar e *buffers*). No modelo apresentado a seguir, são considerados os custos do crossbar, dos *buffers* e dos decodificadores associados à seleção da saída a ser utilizada com base no tamanho de endereço definido no cabeçalho do pacote SPIN (8 bits na versão atual do roteador [AND 2003]). O custo total do roteador em portas lógicas é então estimado a partir de detalhes da implementação do roteador disponibilizados em [AND 2001].

O roteador RSPIN é constituído por um crossbar de dados e um crossbar de controle. O crossbar de dados, por sua vez, é formado por estruturas baseadas em árvores de multiplexadores 2∝1 (com saída invertida ou direta), *buffers tri-state*, inversores e portas NAND. Para facilitar a identificação do custo do crossbar, a parte operativa do roteador é reapresentada na Figura 6.5. A estrutura dos multiplexadores que compõem o crossbar de dados é mostrada na Figura 6.6 [AND 2001], na qual são identificadas as células correspondentes ao se considerar um projeto baseado na biblioteca da AMS [AUS 2001a].

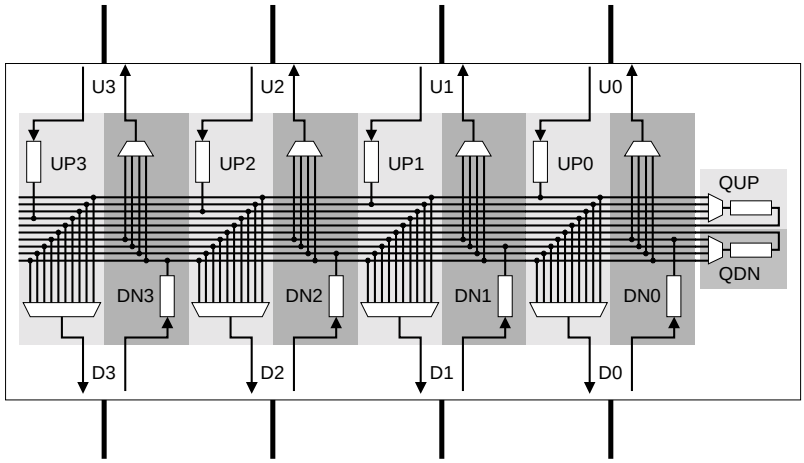


FIGURA 6.5 – A parte operativa do roteador RSPIN.

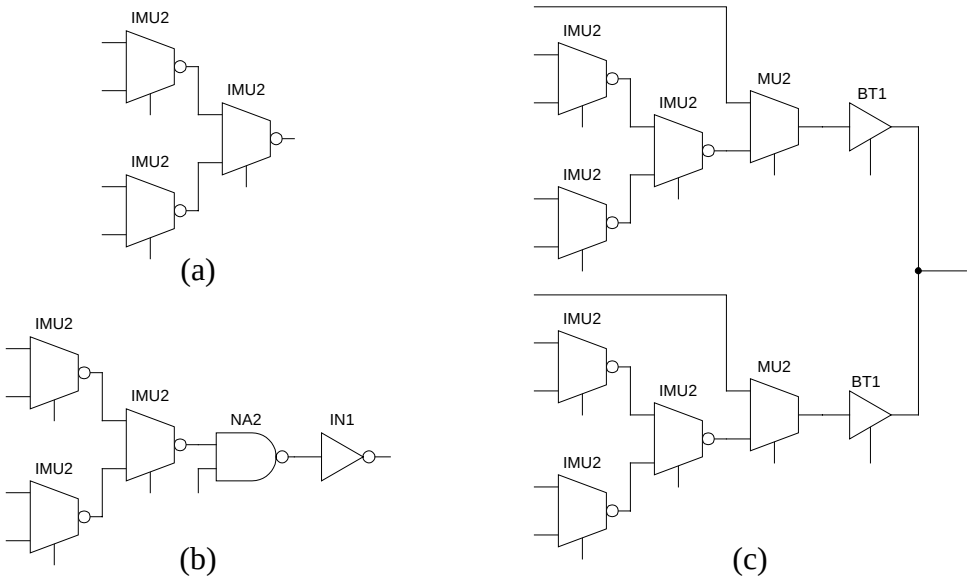


FIGURA 6.6 – Multiplexadores do roteador RSPIN: (a) *buffers* centrais QUP e QDN; (b) canais de saída superiores U0-3; (c) canais de saída inferiores D0-3 [AND 2001].

Na Tabela 6.4, é computado o custo para cada bit do crossbar de dados. Na primeira linha da tabela, são listadas as células utilizadas e a quantidade equivalente de células NA2. Na coluna mais à esquerda, são listadas as quantidades de cada tipo de multiplexador usado no crossbar. No interior da tabela, é indicada a quantidade de cada tipo de célula utilizada em cada tipo de multiplexador, assim como o custo equivalente em células NA2. Por exemplo, o multiplexador de cada canal de saída superior utiliza três células IMU2, cada qual correspondendo a duas células NA2. Assim, como existem quatro canais superiores, o número equivalente de células NA2 é igual a $4 \propto 3 \propto 2{,}0 = 24{,}0$. O custo do crossbar de dados é igual a 130,8 células NA2 por bit de dado.

TABELA 6.4 – Custo por bit do crossbar de dados do roteador RSPIN.

	NA2 (1,0)	**IN1 (0,7)**	**IMU2 (2,0)**	**MU2 (2,3)**	**BT1 (2,7)**	**Custo parcial (Qtd. NA2)**
2 *buffers* centrais			3 ⇒ 12,0			12,0
4 canais superiores	1⇒ 4	1 ⇒ 2,8	3 ⇒ 24,0			30,8
4 canais inferiores			6 ⇒ 48,0	2 ⇒ 18,4	2 ⇒ 21,6	88,0
						T = 130,8

O crossbar de controle realiza o chaveamento dos sinais de controle de fluxo internos do roteador e é constituído, basicamente, por *buffers tri-state*. Para cada ponto de chaveamento do crossbar representado na Figura 6.7, existe um par de *buffers tri-state*. Sendo assim, o número total de *buffers tri-state* é dado por $2 \propto 64 = 128$. Como cada *buffer tri-state* corresponde a 2,7 células NA2, o número total de células NA2 no crossbar de controle é igual a 345,6.

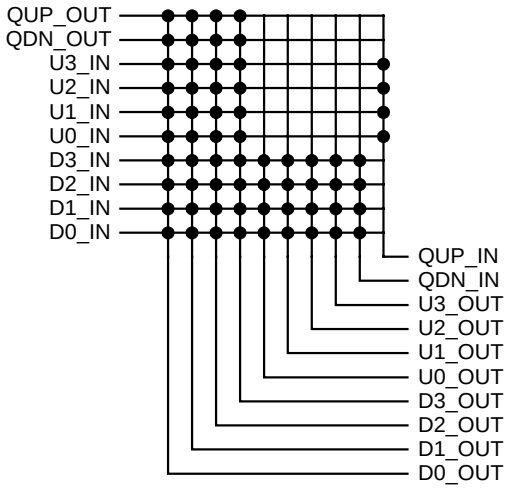


FIGURA 6.7 – Conexões possíveis no crossbar do roteador RSPIN.

Dado o custo por bit do crossbar de dado e o custo total do crossbar de controle, o número equivalente de células NA2 gastas com o chaveamento de dados no roteador RSPIN é:

$$N_{gates,RSPINswitch} = 345{,}6 \quad + \quad 130{,}8 \propto (W_{data} + 4) \tag{6.16}$$

na qual são incluídos os quatro bits de banda lateral que atravessam o roteador. Embora o tamanho da palavra de dado do roteador RSPIN seja conhecido (= 32 bits), essa informação é mantida como um parâmetro configurável para as avaliações a seguir.

Quanto à área de silício gasta na implementação dos *buffers* de memorização de entrada e central, o custo dos mesmos é estimado como uma função da profundidade (p_{in} e p_{cq}, respectivamente) e da largura da palavra de dados (W_{data}). Considera-se que os *buffers* são construídos utilizando-se uma arquitetura de memória RWM (*Read-Write Memory*) com acesso do tipo FIFO (*First-In*, *First-Out*) [RAB 96] e que cada célula de um bit dos *buffers* de memorização é implementada como uma célula de memória SRAM CMOS. Tal escolha baseia na afirmativa de Carro [CAR 99], segundo o qual o uso de flip-flops ou latches para implementar bits de memória é possível, mas aumenta significativamente o custo do circuito. De fato, uma célula de flip-flop tipo D na biblioteca da AMS possui um custo equivalente ao de cinco portas células NA2 [AUS 2001a]. Em uma abordagem simplificada, assume-se que cada bit de memória implementado através de uma célula SRAM CMOS tenha um custo equivalente ao de uma célula NA2 da biblioteca considerada, de onde se obtém:

$$N_{gates,RSPINmem} = (8 \propto p_{in} \; + \; 2 \propto p_{cq}) \propto (W_{data} \; + \; 4) \tag{6.17}$$

Continuando, desde que o modelo do barramento considera o custo do endereçamento, no modelo da rede SPIN esse custo seria dado pelo circuito de roteamento associado à saída de cada *buffer* de armazenamento. Contudo, para simplificar a modelagem, assume-se que o circuito de roteamento equivale a um decodificador de endereços para um endereço de largura W_{route}, a qual consiste na largura do campo de roteamento do cabeçalho do pacote SPIN (8 bits). Como existem 64 pontos de chaveamento no crossbar e para cada um deles deve ser atribuído um circuito de roteamento, a partir de (6.13), obtém-se:

$$N_{gates,RSPINaddr} = 64 \propto (1{,}08 \propto W_{route} - 0{,}46) \quad (6.18)$$

Finalmente, o número equivalente de portas lógicas do roteador RSPIN é dado por:

$$N_{gates,RSPIN} = N_{gates,RSPINswitch} + N_{gates,RSPINmem} + N_{gates,RSPINaddr} \quad (6.19)$$

ou seja:

$$\begin{aligned} N_{gates,RSPIN} \quad &= [345{,}6 \; + \; 130{,}8 \propto (W_{data} + 4)] \\ &+ [(8 \propto p_{in} \; + \; 2 \propto p_{cq}) \propto (W_{data} \; + \; 4)] \\ &+ [64 \propto (1{,}08 \propto W_{route} - 0{,}46)] \end{aligned} \quad (6.20)$$

Na tabela abaixo, são apresentadas as estimativas de custo do roteador RSPIN para a configuração utilizada pelos seus projetistas [GRE 2003]: W_{data} = 32 bits, p_{in} = 4 flits, p_{cq} = 18 flits e W_{route} = 8 bits.

TABELA 6.5 – Custo do roteador RSPIN em células NA2 da biblioteca da AMS.

	Qtd. NA2	**% $N_{gates,\ RSPIN}$**
$N_{gates,RSPINswitch}$	5054	63,0%
$N_{gates,RSPINmem}$	2448	30,5%
$N_{gates,RSPINaddr}$	523	6,5%
$N_{gates,\ RSPIN}$	8025	100%

Conforme indicado em [AND 2001], o roteador RSPIN com as características listadas acima possui 9144 portas lógicas. Com isso, o erro percentual do valor estimado por (6.20) é de 12,23%. Esse erro pode ser atribuído à não inclusão dos custos dos circuitos de controle de acesso aos *buffers* de memorização e dos blocos controladores de saída (OCBs), os quais implementam os árbitros do roteador. Uma segunda fonte de erro está na consideração de que os blocos controladores de entrada (ICBs) que implementam os circuitos de roteamento sejam equivalentes a agregados de decodificadores de endereços semelhantes aos utilizados no barramento. Finalmente, uma outra fonte de erro está na relação utilizada para indicar o número equivalente de células NA2 para cada porta lógica do modelo, a qual foi tomada a partir da relação entre as áreas das células na tecnologia da AMS. Essa relação varia entre diferentes bibliotecas e tecnologias. Contudo, apesar das diferentes fontes de erro, pode-se considerar aceitável o erro percentual do valor estimado, dado o nível de abstração utilizado. Além disso, na coluna mais à direita na Tabela 6.5, nota-se que cerca de 30,5% das portas lógicas são reservadas aos *buffers* de memorização. Conforme indicado em [AND 2001], essa relação é de 29,6% no roteador sintetizado. Concluindo, pode-se afirmar que a abordagem utilizada para a determinação do modelo para estimativa do número equivalente de portas lógicas para o roteador RSPIN pode ser aplicada para a obtenção de modelos para roteadores com características similares.

6.5.2 Modelo para a estimativa do número de portas lógicas na rede SPIN

O número de portas lógicas na rede SPIN é dado pelo produto do número de portas do roteador RSPIN pelo número de roteadores utilizados na rede:

$$N_{gates,SPIN} = N_{gates,RSPIN} \propto N_{routers,RSPIN} \tag{6.21}$$

Segundo Guerrier e Greiner [GUE 2002b], o número de roteadores em uma rede SPIN com n terminais cresce com $n \propto (\log n)/8$, onde n é dado por 2^m, e m é um número natural. Contudo, como pode ser observado na Tabela 6.6, essa expressão não é válida quando $\log_4 n$ também não é um número natural (ou seja, quando n = 8, 32 e 128). Nesses casos, a expressão a ser utilizada para a determinação do número de roteadores é dada por $n \propto (\log n - 1)/8$. Dessa forma, o número de roteadores na rede SPIN ($N_{routers,\mathrm{SPIN}}$) seria dado por:

$$N_{routers,SPIN} = n \propto (\log n)/8 \qquad \text{quando } \log_4 n \notin \mathrm{O} \tag{6.22}$$

$$N_{routers,SPIN} = n \propto (\log n - 1)/8 \quad \text{quando } \log_4 n \angle \mathrm{O} \tag{6.23}$$

TABELA 6.6 – Número de roteadores na rede SPIN.

n	$N_{routers,SPIN}$ **Indicado em [GUE 2000b]**	$N_{routers,SPIN}$ **Calculado por (6.22)**	$N_{routers,SPIN}$ **Calculado por (6.22) e (6.23)**
4	1	1	1
8	2	3	2
16	8	8	8
32	16	20	16
64	48	48	48
128	96	112	96
256	256	256	256

Contudo, quando $\log_4 n$ é um número natural (n é dado por uma potência de 4), as portas de comunicação superiores dos roteadores do nível mais alto da árvore não são utilizadas. Por isso, tanto essas portas como a fila central superior de cada um desses roteadores não precisam ser sintetizadas, permitindo uma redução do custo desses roteadores. De uma maneira simplificada, pode-se afirmar que tal custo é dado pela metade do custo de um roteador completo. Assim, (6.22) deve ser reescrita de modo que quando $\log_4 n$ for número natural, $[(n \propto \log n)/8 - n/4]$ roteadores serão completamente implementados e $n/4$ roteadores terão seus custos reduzidos em 50%. Disso obtém-se um modelo igual ao modelo especificado para a condição de $\log_4 n$ não pertencer aos números naturais. Portanto, pode-se assumir (6.23) como o modelo final para a determinação do número equivalente de roteadores para ambas as configurações tratadas nos modelos (6.22) e (6.23), ou seja:

$$N_{routers,SPIN} = n \propto (\log n - 1)/8 \tag{6.24}$$

Aplicando-se (6.24) no cálculo do custo do número equivalente de portas lógicas para cada tamanho de rede SPIN, obtém-se os valores mostrados na Tabela 6.7, destacando-se que foi utilizado o valor estimado para o número de portas lógicas, o qual possui um erro de 12,23%.

TABELA 6.7 – Número de portas lógicas na rede SPIN.

n	$N_{routers,SPIN}$	$N_{gates,SPIN}$
4	0,5	4013
8	2	16052
16	6	48156
32	16	128415
64	40	321037
128	96	770488
256	224	1797806

De fato, em uma rede SPIN com quatro núcleos, todos os núcleos são conectados às portas inferiores de um único roteador e as portas superiores não são utilizadas. Por outro lado, em uma rede com oito núcleos, os dois roteadores utilizados são interconectados por meio de suas portas superiores.

Finalmente o custo em silício de uma rede SPIN é estimado por:

$$A_{total,SPIN} = A_{gate} \propto N_{gates,RSPIN} \propto N_{routers,RSPIN} \tag{6.25}$$

Aplicando-se (6.25) na estimativa da sobrecarga da área de silício da rede SPIN para diferentes configurações de tamanho dos núcleos, número de núcleos no sistema e largura do canal de dados, obtém os gráficos a seguir. As profundidades dos *buffers* de entrada (p_{in}) e central (p_{cq}) foram definidas em 4 e 18 flits, respectivamente. Destacam-se as configurações com palavra de dado de 32 bits (largura efetivamente utilizada na rede SPIN), bem como os resultados de sobrecarga inferiores ou iguais a 10% da área de silício devida aos núcleos do sistema.

Na Figura 6.8 e na Figura 6.9, são mostrados os resultados para sistemas baseados em núcleos com 10 mil portas lógicas. Observa-se que poucas configurações apresentam a sobrecarga de área de silício inferior a 10%, pois a área do roteador é muito próxima da área dos núcleos. Apenas em sistemas com até oito núcleos e com palavra de dados de inferior a 64 bits foi possível atender ao limite de sobrecarga de área desejado.

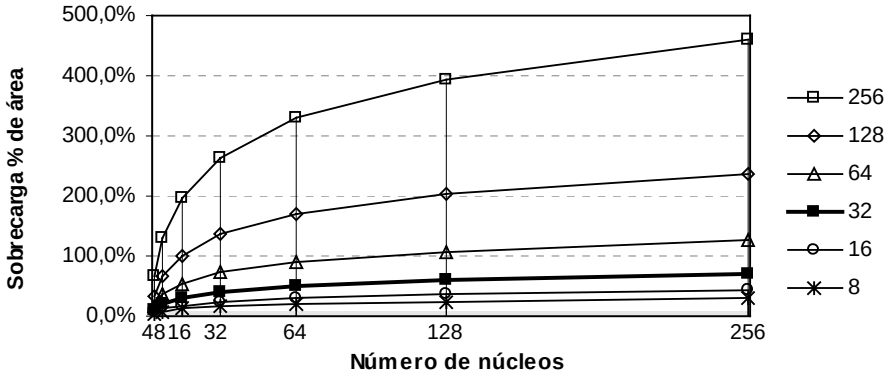


FIGURA 6.8 – Sobrecarga % de área de silício da rede SPIN para núcleos com 10 mil portas lógicas (A_{core} = 0,55 mm^2).

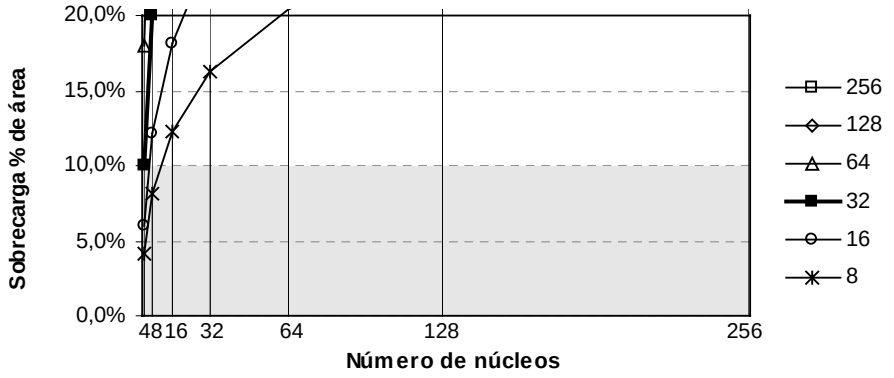


FIGURA 6.9 – Sobrecarga % de área de silício da rede SPIN para núcleos com 10 mil portas lógicas (A_{core} = 0,55 mm^2) . visualização da faixa de 10%.

Nos sistemas baseados em núcleos com 50 mil portas lógicas (Figura 6.10), o número de configurações com sobrecarga na faixa dos 10% é maior. Em todos os sistemas de 8 e 16 bits esse limite foi respeitado. Nas configurações de 32 bits, é possível a construção de sistemas com até 64 núcleos e ainda manter uma sobrecarga de área de até 10%. Em sistemas com palavras mais largas, o número de núcleos permitidos seria muito limitado (8 ou 4 núcleos).

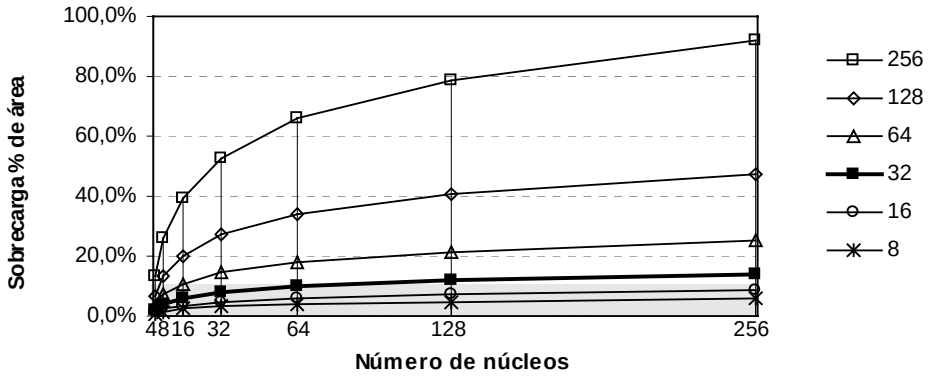


FIGURA 6.10 – Sobrecarga % de área de silício da rede SPIN para núcleos com 50 mil portas lógicas (A_{core} = 2,75 mm^2).

Nos sistemas baseados em núcleos com 100 mil portas lógicas, todas as configurações com palavra de 8 a 32 bits e a maioria das configurações com palavra de 64 bits atendem ao requisito de sobrecarga de até 10%. Novamente, sistemas baseados em palavras mais largas apresentam um sobrecarga muito grande com o aumento do número de núcleos no sistema.

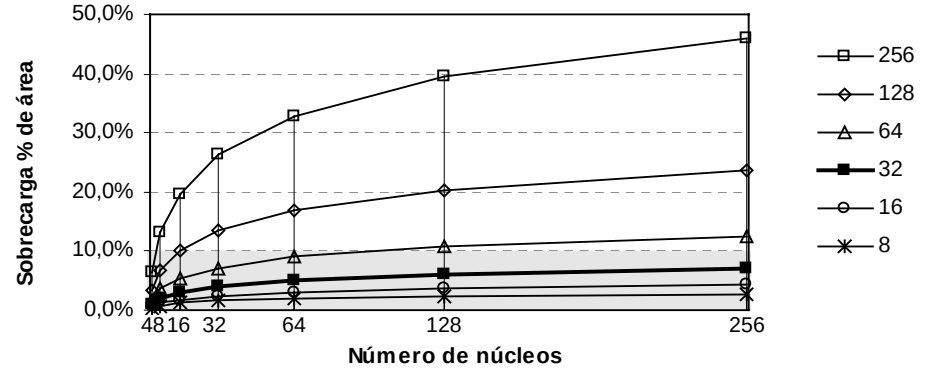


FIGURA 6.11 – Sobrecarga % de área de silício da rede SPIN para núcleos com 100 mil portas lógicas (A_{core} = 5,50 mm^2).

Os resultados obtidos com a aplicação do modelo de estimativa de área de silício da rede SPIN para as configurações utilizadas apontam para algumas conclusões a respeito da sua aplicabilidade em sistemas integrados. As características do roteador RSPIN tornam a rede SPIN muito cara para a construção de sistemas baseados em núcleos com até 10 mil portas lógicas. As únicas configurações possíveis encontrariam no barramento a melhor solução em termos de custo e o desempenho seria satisfatório devido a pouca quantidade de núcleos (até oito unidades). Contudo, os resultados também mostram que a sobrecarga da rede SPIN torna-se aceitável em sistemas com núcleos maiores, sendo que, para a palavra de referência (32 bits), a rede SPIN poderia ser utilizada em quase todas as configurações, atendendo ao requisito de sobrecarga.

6.6 Modelos para Estimativa de Área da Rede SoCIN

A determinação dos modelos para a estimativa de área da rede SoCIN adota uma abordagem análoga à utilizada para a rede SPIN. Ou seja, é determinado um modelo para estimativa do número equivalente de portas lógicas do roteador RASoC com base na biblioteca de células da AMS e, após isso, é determinado o modelo para a estimativa da área de silício da rede SoCIN.

6.6.1 Modelo para a estimativa do número de portas lógicas no roteador RASoC

O roteador RASoC, cuja arquitetura foi apresentada no Capítulo 5, possui cinco portas de comunicação bidirecionais, cada uma composta por um canal de entrada e um canal de saída. Internamente, ele é constituído por um crossbar 5∝5 parcial, cinco *buffers* de armazenamento nos canais de entrada e controladores que comandam a operação de cada recurso do roteador (crossbar e *buffers*). No modelo apresentado a seguir, são considerados os custos do crossbar, dos *buffers* e dos circuitos de roteamento associados à seleção da saída a ser utilizada. O custo total do roteador em portas lógicas é então estimado pela soma dos custos das instâncias de cada um desses componentes.

O crossbar do roteador RASoC é implementado de maneira distribuída por meio de chaves de multiplexação baseadas em um circuito de chaveamento com quatro canais de entrada de 1 bit. Para o chaveamento dos canais de dados são necessárias W_{data}+2 chaves 4∝1 de 1 bit para cada porta do roteador (onde W_{data} corresponde ao parâmetro *DATA_WIDTH* do modelo VHDL do roteador). Portanto, para o chaveamento dos canais de dados, são necessárias 5∝(W_{data} + 2) chaves 4∝1 de 1 bit. Já com relação ao chaveamento dos sinais de controle de fluxo interno, cada uma das cinco portas de comunicação do roteador requer um par de chaves 4∝1 de 1 bit, o que resulta em 5∝2 chaves 4∝1 de 1 bit.

Com relação ao custo de cada chave 4∝1 de 1 bit, seu valor depende da estrutura a ser utilizada na síntese do roteador. Por exemplo, a seguir são ilustradas três alternativas de implementação com base na biblioteca de células da AMS [AUS 2001a], utilizada nas modelagens anteriores.

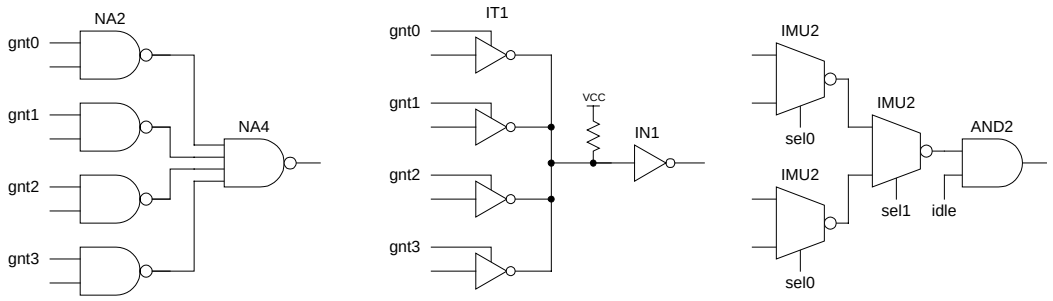


FIGURA 6.12 – Estruturas de chaves 4∝1 de 1 bit baseadas na biblioteca de células da AMS: (a) arranjo de NANDs; (b) *buffers tri-state*; (c) multiplexadores 2∝1.

Como pode ser observado na Figura 6.12, as chaves são projetadas de modo a garantir uma saída estável em 0 quando nenhuma entrada é selecionada. Na estrutura da Figura 6.12.a, a seleção é feita diretamente a partir dos sinais de confirmação (*gnt*) e saída é garantida em 0 pela própria lógica quando nenhum dos sinais de confirmação é ativado. Já na segunda estrutura (Figura 6.12.b), as saídas dos *buffers* flutuam quando nenhuma confirmação é dada e o resistor de *pull-up* assegura o nível estável necessário. Por fim, na estrutura da Figura 6.12.c, é utilizada uma árvore de multiplexadores 2∝1 e uma porta lógica AND para garantir um nível lógico 0 na saída quando nenhuma confirmação é dada.

O número equivalente de portas NA2 para cada alternativa é mostrado na Tabela 6.8. Como pode ser observado, a necessidade de uma porta lógica AND de duas entradas torna a solução baseada em multiplexadores 2∝1 um pouco mais cara que a solução baseada em um arranjo de dois níveis de portas NAND. Além disso, observa-se a necessidade de uma lógica adicional para a geração dos sinais *sel1*, *sel0* e *idle*, a qual pode ser implementada junto ao crossbar ou gerada pelo próprio controlador associado. Disso, nos modelos que seguem, será considerado que cada chave 4∝1 de 1 bit tem um custo equivalente a sete portas lógicas NAND.

TABELA 6.8 – Número equivalente células NA2 em diferentes estruturas de chaves de multiplexação baseadas na biblioteca de células da AMS [AUS 2001a].

	NA2 (1,0)	**NA4 (3,0)**	**IT1 (3,3)**	**IN1 (0,7)**	**IMU2 (2,0)**	**AND2 (2,0)**	**Qtd. NA2**
Arranjo de NANDs	$4 \Rightarrow 4$	$1 \Rightarrow 3$					7,0
Buffers tri-state			$4 \Rightarrow 13{,}2$	$1 \Rightarrow 0{,}7$			13,9
Multiplexadores 2∝1					$3 \Rightarrow 6$	$1 \Rightarrow 1{,}3$	7,3

Finalmente, o número equivalente de portas lógicas associadas aos circuitos de chaveamento do crossbar é dado por:

$$N_{gates,RASoCswitch} = [5 \propto (W_{data} + 2) + 5 \propto 2] \propto 7 \quad (6.26)$$

ou

$$N_{gates,RASoCswitch} = 35 \propto (W_{data} + 4) \quad (6.27)$$

Por exemplo, aplicando-se (6.27) para o cálculo do custo do crossbar de um roteador RASoC com canais de dado de 32 bits (W_{data}) obtém um total de 1260 células NA2.

Quanto à área de silício gasta na implementação dos *buffers* de memorização dos canais de entrada, o custo desses *buffers* é estimado como uma função da profundidade (p_{in}) e da largura da palavra de dados (W_{data}). Considerando-se, novamente, que cada célula de um bit dos *buffers* de memorização é implementada como uma célula de memória SRAM CMOS e que a mesma possua área equivalente ao de uma porta NAND com duas entradas, o número total de células NA2 dos *buffers* de memorização do roteador RASoC é dado por:

$$N_{gates,RASoCmem} = 1{,}0 \propto (W_{data} + 2) \propto p_{in} \propto 5 \quad (6.28)$$

Por exemplo, o número equivalente de portas lógicas associadas aos *buffers* de um roteador RASoC com palavra de dado de 32 bits e *buffers* com capacidade de armazenar até quatro flits seria igual a 680 células NA2.

Com relação aos custos associados ao roteamento, foram comparados os resultados obtidos por (6.13), assumindo-se um decodificador de W_{route} bits para cada um dos 20 pontos de chaveamento, com uma estimativa efetuada a partir do circuito de roteamento descrito no Capítulo 5. O erro entre as duas estimativas foi de aproximadamente 40%. Dessa forma, tendo-se disponível um modelo baseado diretamente no circuito projetado, optou-se por utilizar esse modelo na estimativa do número de portas lógicas do roteador RASoC. A equação (6.29) calcula o custo dos cinco circuitos de roteamento, sendo que cada um deles possui um custo dado por $(7{,}73 \propto W_{route} - 8{,}28)$.

$$N_{gates,RASoCaddr} = 5 \propto (7{,}73 \propto W_{route} - 8{,}28) \tag{6.29}$$

Por exemplo, se W_{route} é igual a 10 bits, largura suficiente para se construir uma rede em grelha quadrada com até 256 roteadores, o número equivalente de portas lógicas dos circuitos de roteamento é igual a 345,1, enquanto que o valor calculado por (6.13) é igual a 206,8 (erro de 40,0%).

O número equivalente de portas lógicas do roteador RASoC é então dado por:

$$N_{gates,RASoC} = N_{gates,RASoCswitch} + N_{gates,RASoCmem} + N_{gates,RASoCaddr} \tag{6.30}$$

ou, substituindo-se (6.27), (6.28) e (6.29) em (6.30):

$$\begin{aligned} N_{gates,RASoC} &= [35 \propto (W_{data} + 4)] \\ &+ [(W_{data} + 2) \propto 5 \propto p_{in}] \\ &+ [5 \propto (7{,}73 \propto W_{route} - 8{,}28)] \end{aligned} \tag{6.31}$$

A Tabela 6.9 apresenta as estimativas de custo do roteador RASoC para uma configuração equivalente à utilizada na versão atual do roteador RSPIN [ADR 2003]: W_{data} = 32 bits, p_{in} = 4 flits, e W_{route} = 10 bits. O valor de W_{route} consiste na largura necessária para se rotear um pacote por uma rede SoCIN com capacidade de interconectar tantos núcleos quanto o número de núcleos possíveis de serem interconectados por uma rede SPIN com W_{route} = 8 bits, ou seja, 256 núcleos. Como pode ser observado, comparando-se o número equivalente de portas lógicas estimado para o roteador RSPIN (8205) e para o roteador RASoC (2285), o segundo roteador apresenta um custo estimado aproximadamente 3,5 vezes menor que o do primeiro.

TABELA 6.9 – Custo do roteador RASoC em células NA2 da biblioteca da AMS.

	Qtd. NA2
$N_{gates,RASoCswitch}$	1260
$N_{gates,RASoCmem}$	680
$N_{gates,RASoCaddr}$	345
$N_{gates,\ RASoC}$	2285

Assumindo-se que o modelo apresenta uma margem de erro semelhante à obtida com o modelo da rede SPIN, pode-se estimar que, levando-se em conta os circuitos de

controle, o número equivalente de portas lógicas do roteador RASoC seja aproximadamente igual a 2564.

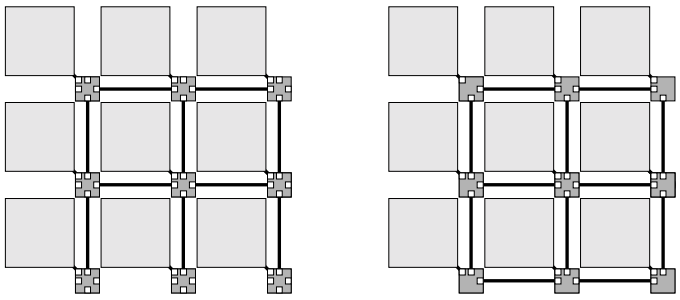
6.6.2 Modelo para a estimativa do número de portas lógicas na rede SoCIN

O número de portas lógicas na rede SoCIN é dado pelo produto do número de portas lógicas do roteador RASoC pelo número de roteadores utilizados na rede:

$$N_{gates,SoCIN} = N_{gates,RASoC} \propto N_{routers,RASoC} \tag{6.32}$$

Considerando-se uma topologia em grelha, algumas abordagens alternativas de síntese da rede podem ser tomadas, as quais conduzem a modelos variantes de (6.32) que resultam em diferentes custos. Em ambas abordagens, é considerado que todos os núcleos são conectados diretamente a um roteador RASoC, mas, para um mesmo número de núcleos, podem ser obtidas redes com tamanhos e custos diferentes, dependendo da quantidade de núcleos conectada a um mesmo roteador, conforme segue.

Na primeira abordagem, cada núcleo é conectado à porta de comunicação local de um roteador e existem tantos roteadores quantos são os núcleos do sistema (ou seja, n). Dessa abordagem, podem ser derivadas duas implementações. Em uma delas, ilustrada na Figura 6.13.a, todos os roteadores são baseados em um mesmo modelo e as portas de comunicação não utilizadas são desperdiçadas. Porém, a topologia pode ser facilmente estendida para um toróide simples ou dobrado. Na outra, ilustrada na Figura 6.13.b, são implementadas apenas as portas de comunicação necessárias à conexão com roteadores vizinhos. Nesse caso, em uma abordagem simplificada (como foi adotado na rede SPIN), assume-se que cada porta de comunicação do roteador custa $N_{gates,RASoC}/5$ e o custo de um roteador na periferia da rede seria igual a 60 ou 80% do custo de um roteador completo, conforme a sua posição. Deve-se destacar que as figuras apresentadas abaixo não representam o posicionamento real de núcleos e roteadores, nem o roteamento dos canais físico em um chip real.



(a) (b)

FIGURA 6.13 – Implementações da rede em grelha: (a) roteadores da periferia sintetizados completamente; (b) roteadores da periferia sintetizados parcialmente.

Considerando-se apenas as topologias de grelha quadradas com k roteadores em cada dimensão e fazendo-se o número de roteadores na rede ($r = k \propto k$) igual ao número de núcleos no sistema (n), pode-se expressar o número equivalente de portas lógicas em cada implementação por:

$$N_{gates,SoCIN} = N_{gates,RASoC} \propto r \tag{6.33}$$

na primeira alternativa e

$$N_{gates,SoCIN} = N_{gates,RASoC} \propto (r \ . \ 4/5 \propto r^{1/2}) \tag{6.34}$$

na segunda alternativa de implementação. Essa equação expressa que $r^{1/2}$ portas de comunicação de cada um dos quatro lados da rede não devem ser implementadas, sendo que cada porta de comunicação apresenta um custo igual a 1/5 do custo do roteador.

Aplicando-se (6.33) e (6.34) para redes de diferentes tamanhos obtém-se a Tabela 6.10, na qual observa-se que a redução percentual de área decresce com o tamanho da rede devido ao crescimento quadrático da quantidade de roteadores internos que possuem maior custo que os roteadores periféricos, cuja quantidade cresce linearmente com o tamanho da rede. Em uma rede com capacidade de conectar 256 núcleos, a redução é de apenas 5%.

TABELA 6.10 – Relação entre o número de portas lógicas para diferentes implementações da rede.

k	***r***	$N_{gates,SoCIN}$ **(6.33)**	$N_{gates,SoCIN}$ **(6.34)**	**Redução % 1 – (6.33)/(6.34)**
2	4	9140	5484	40,0%
3	9	20565	15081	26,7%
4	16	36560	29248	20,0%
5	25	57125	47985	16,0%
6	36	82260	71292	13,3%

Em uma abordagem alternativa, todas as portas de comunicação periféricas não utilizadas para interconexão entre roteadores da rede são alocadas para a conexão direta de núcleos. Assim, poderiam ser conectados de dois a três núcleos em um mesmo roteador, reduzindo o número de roteadores necessários. Dessa forma, a relação $r = n$ não seria mais válida e r seria determinado pela capacidade da rede em oferecer n portas de comunicação para a conexão de núcleos. Por exemplo, um sistema com nove núcleos poderia ser baseado em uma grelha $2 \propto 2$ (Figura 6.14), sendo que essa rede teria capacidade de interconectar até 12 núcleos. Destaca-se que a figura não representa o posicionamento real de núcleos e roteadores, nem o roteamento dos canais físicos em um chip real.

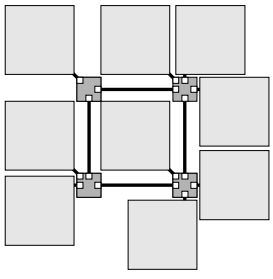


FIGURA 6.14 – Implementação alternativa da rede em grelha.

Assumindo-se uma topologia quadrada com k roteadores em cada dimensão e um total de r roteadores (onde $r = k \propto k$), a capacidade de conexão (c) de núcleos aos roteadores da rede é dada por:

$$c = r \quad + \quad (4 \propto r^{1/2}) \tag{6.35}$$

ou seja, a rede possui r portas locais para conexão de núcleos mais $r^{1/2}$ portas em cada um dos quatro lados do roteador.

Na Tabela 6.11, são listadas as capacidades de conexão de topologias em grelha 2-D com diferentes números de roteadores.

TABELA 6.11 – Capacidade da rede conectando-se núcleos às portas periféricas.

k	***r***	***c***
1	1	5
2	4	12
3	9	21
4	16	32
5	25	45

Em uma abordagem simplificada, estima-se o custo para implementar uma grelha quadrada com r roteadores para a conexão de n núcleos, sendo que são descontados os custos das portas de comunicação não utilizadas. O parâmetro r é definido por $(n^{1/2} \; . \; 1)^2$ de tal forma que r seja o menor número de roteadores (em uma rede quadrada $k \propto k$ com k roteadores em cada dimensão) e $c(r)$ seja suficiente para a conexão direta dos n núcleos do sistema. Em outras palavras, $c(r_{min}) \times n$. A partir do valor determinado para r, calcula-se a capacidade da rede aplicando-se (6.35). Desse valor, descontam-se as portas de comunicação não utilizadas fazendo-se $(c \; . \; n)$. Por fim, do custo devido aos r roteadores, é descontado o custo associado às $(c \; . \; n)$ portas de comunicação não implementadas, sendo que cada uma dessas portas possui um custo igual a 1/5 do custo do roteador. Disso obtém-se o modelo de custo que estima o número equivalente de portas lógicas para a rede SoCIN segundo esta abordagem, o qual é dado por:

$$N_{gates,SoCIN} = N_{gates,RASoC} \propto [r \; . \; (c \; . \; n)/5] \tag{6.36}$$

ou

$$N_{gates,SoCIN} = N_{gates,RASoC} \propto [(5 \propto n) \; . \; (12 \propto n^{1/2}) + 8]/5 \tag{6.37}$$

A redução percentual do número de portas lógicas dessa abordagem em relação às outras abordagens é mostrada na Tabela 6.12. Como pode ser observado, conforme o modelo utilizado como referência, a redução do custo pode ser considerada significativa (maior ou igual a 50%) para sistemas com até 16 núcleos. Destaca-se que o valor da redução do número de portas lógicas decresce com o tamanho do sistema, pois o número de roteadores com capacidade de conectar apenas um núcleo cresce quadradicamente com n, enquanto que o número de roteadores com maior capacidade de conexão cresce linearmente com n. Em uma rede com capacidade de conectar 256 núcleos, a redução é de 14,4% no primeiro caso e de 9,9% no segundo.

TABELA 6.12 – Relação entre os custos de diferentes alternativas de redes SoCIN.

n	$N_{gates,SoCIN}$ (6.33)	$N_{gates,SoCIN}$ (6.34)	$N_{gates,SoCIN}$ (6.37)	Redução % 1– (6.37)/(6.33)	Redução % 1 – (6.37)/(6.34)
4	9140	5484	1828	80,0%	66,7%
9	20565	15081	7769	62,2%	48,5%
16	36560	29248	18280	50,0%	37,5%
25	57125	47985	33361	41,6%	30,5%
36	82260	71292	53012	35,6%	25,6%

Avaliando-se os modelos desenvolvidos, conclui-se que a melhor alternativa de construção de uma rede SoCIN em termos de área é a baseada no modelo (6.37). Contudo, se além do custo for considerada a largura de banda agregada da rede e sua escalabilidade, a melhor solução é a baseada no modelo (6.34). Além, disso, deve-se destacar que a mesma arquitetura de roteador pode ser utilizada em ambos os modelos, o que atende aos requisitos de reusabilidade dos sistemas integrados. Por exemplo, em uma primeira avaliação para a exploração do espaço projeto para a construção de um determinado SoC, pode-se partir de uma arquitetura baseada no modelo (6.37) e, se a mesma não satisfazer aos requisitos de comunicação da aplicação, tem-se como alternativa utilizar uma rede SoCIN baseada no modelo (6.34). No Capítulo 7, serão apresentados modelos de estimativa de alto nível do desempenho de comunicação da rede que deverão servir de apoio para esse tipo de avaliação.

Finalmente o custo em silício de uma rede SoCIN é estimado por:

$$A_{total,SoCIN} = A_{gate} \propto N_{gates,RASoC} \propto N_{routers,RASoC} \quad (6.38)$$

Aplicando-se (6.34) e (6.38) no calculo da área de silício para diferentes configurações de tamanho de núcleos, número de núcleos no sistema e largura do canal de dados, são obtidos os gráficos mostrados a seguir, os quais expressam a sobrecarga de área da rede SoCIN para cada configuração. São consideradas as configurações de redes com $k \propto k$ roteadores, onde k varia de 2 a 16, com destaque às configurações com palavra de dado de 32 bits (largura utilizada no barramento PI-Bus e na rede SPIN), bem como aos resultados de sobrecarga inferiores ou iguais a 10% da área de silício devida aos núcleos do sistema. As profundidades dos *buffers* de entrada (p_{in}) foram definidas em 4 flits e campo W_{route} foi estabelecido igual a 10 bits.

Na Figura 6.15 e na Figura 6.16, são mostrados os resultados para sistemas baseados em núcleos com 10 mil portas lógicas. Observa-se que apenas as configurações baseadas em palavras de 8 bits e o menor sistema com palavra de 16 bits apresentam a sobrecarga de área de silício inferior a 10%. Além disso, pode ser notado que a sobrecarga é menor nas configurações com mais núcleos na periferia do que no interior da rede. Com o aumento do número de núcleos no sistema, essa relação se inverte a sobrecarga satura em um valor próximo à relação entre os números de portas lógicas do roteador e de um núcleo (eg. 23,3% para as configurações de 32 bits).

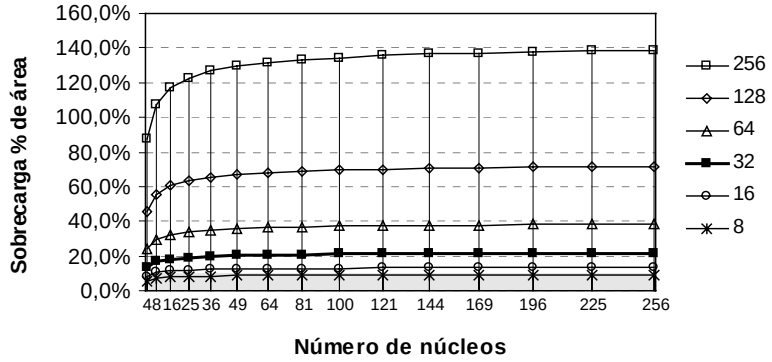


FIGURA 6.15 – Sobrecarga % de área de silício da rede SoCIN para núcleos com 10 mil portas lógicas (A_{core} = 0,55 mm^2).

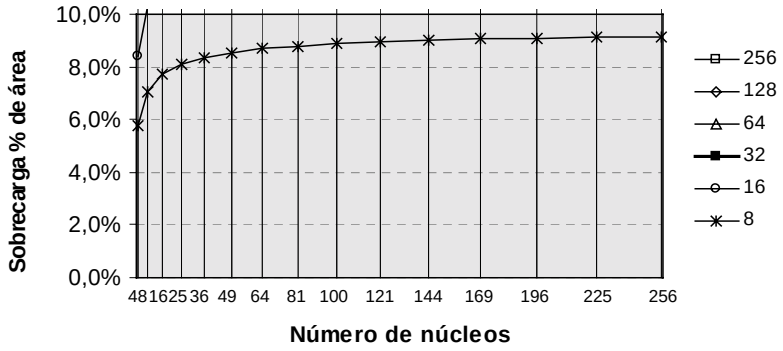


FIGURA 6.16 – Sobrecarga % de área de silício da rede SoCIN para núcleos com 10 mil portas lógicas (A_{core} = 0,55 mm^2) . visualização da faixa de 10%.

Para sistemas baseados em núcleos com um maior número de portas lógicas (eg. 50 mil), as configurações com tamanhos de palavra maiores atendem ao requisito de sobrecarga de área estabelecido. Por exemplo, na Figura 6.17, todos os sistemas com palavra de dado igual ou inferior a 64 bits apresentaram sobrecarga inferior a 10%.

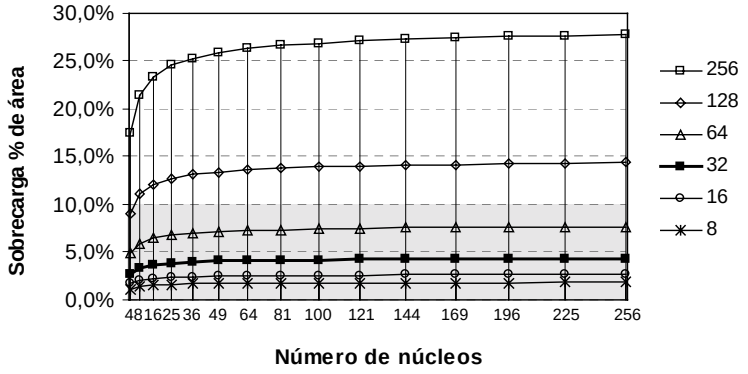


FIGURA 6.17 – Sobrecarga % de área de silício da rede SoCIN para núcleos com 50 mil portas lógicas (A_{core} = 2,75 mm^2).

Já na Figura 6.18, observa-se que inclusive os sistemas com tamanho de palavra de 128 bits possuem sobrecarga dentro do nível desejado.

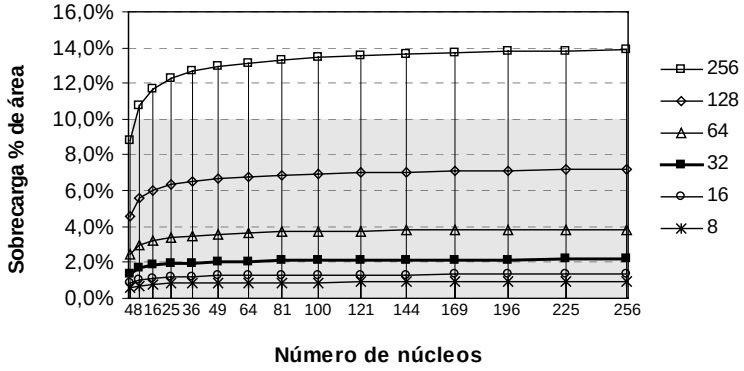


FIGURA 6.18 – Sobrecarga % de área de silício da rede SoCIN para núcleos com 100 mil portas lógicas (A_{core} = 5,50 mm^2).

Analisando-se os três casos representados nos gráficos acima, observa-se que a rede SoCIN apresenta um nível de sobrecarga aceitável para uma larga faixa de configurações em sistemas com 50 e 100 mil portas lógicas. Contudo, sua aplicabilidade é restrita no caso de sistemas com 10 mil portas lógicas, se for considerado o limite de sobrecarga estabelecido. Entretanto, se for considerada a abordagem alternativa, na qual os núcleos podem ser conectados a diferentes portas dos roteadores periféricos, é maior o número de configurações com sobrecarga inferior a 10%. Como pode ser observado, na Figura 6.19, diversas configurações de 16 bits e algumas de 32 e 64 bits atingem essa meta.

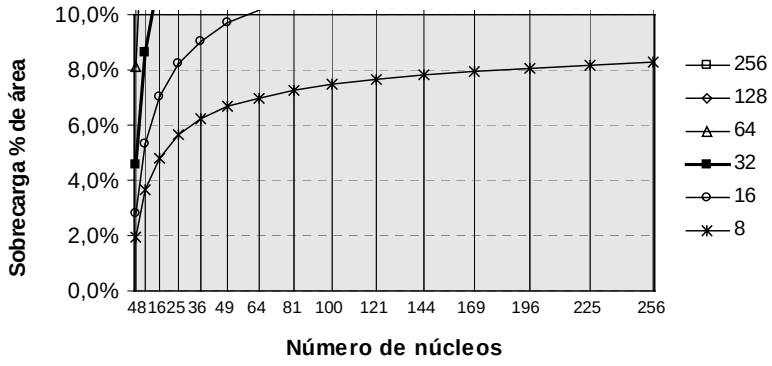


FIGURA 6.19 – Sobrecarga % de área de silício da rede SoCIN para núcleos com 10 mil portas lógicas (A_{core} = 0,55 mm^2) na abordagem alternativa . visualização da faixa de 10%.

6.7 Comparação das Sobrecargas de Área do Barramento e das Redes-em-Chip

Os gráficos a seguir resumem os resultados ilustrados nas tabelas de sobrecarga de área de silício para o barramento PI-Bus e para as redes-em-chip SPIN e SoCIN. São apresentadas as sobrecargas percentuais dessas arquiteturas para sistemas baseados em uma palavra de dados de 32 bits em configurações com 4, 16, 64 e 256 núcleos. Para a rede SoCIN, são utilizados os modelos (6.34) e (6.37) para o cálculo do número equivalente de portas lógicas na rede.

Como pode ser observado na Figura 6.20 e na Figura 6.21, para sistemas baseados em núcleos com dez mil portas lógicas, o barramento é a única arquitetura com sobrecarga de área de silício inferior a 10% em todas as configurações (valor de referência estabelecido). Comparando-se as duas arquiteturas de redes-em-chip, percebe-se que o número equivalente de portas lógicas da rede SoCIN satura próximo do valor correspondente à relação entre a área do roteador RASoC e a área de um núcleo. Por outro lado, a sobrecarga de área da rede SPIN mantém uma taxa de crescimento com o aumento do tamanho do sistema.

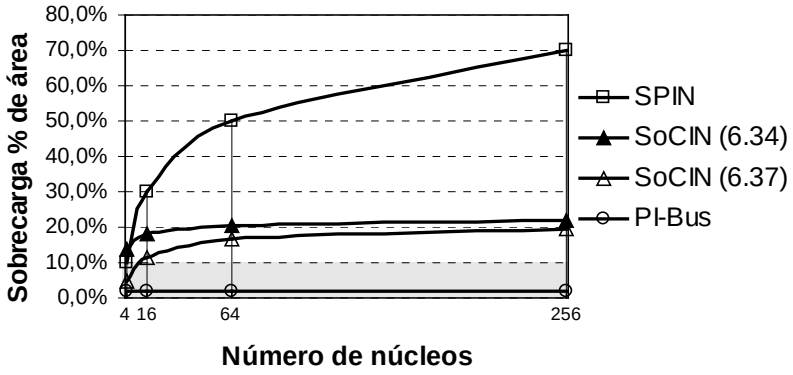


FIGURA 6.20 – Comparativo da sobrecarga de área do PI-Bus e das redes SPIN e SoCIN para sistemas baseados em núcleos com 10 mil portas lógicas (A_{core} = 0,55 mm^2) e palavra de 32 bits.

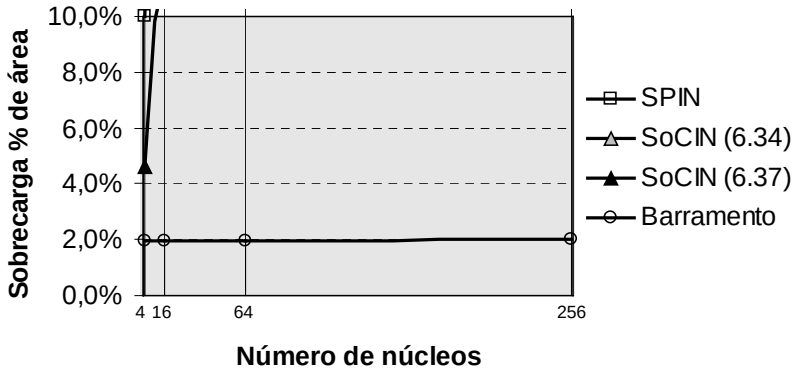


FIGURA 6.21 – Comparativo da sobrecarga de área do PI-Bus e das redes SPIN e SoCIN para sistemas baseados em núcleos com 10 mil portas lógicas – aproximação da faixa de 10%.

Com relação ao valor de referência de sobrecarga de área (10%), como pode ser observado na Figura 6.21, as redes SPIN e SoCIN atingem esse valor apenas em sistemas com quatro núcleos, sendo que a rede SoCIN só obtém esse resultado se for utilizada a abordagem alternativa de conexão dos núcleos à rede, obtida com (6.37), a qual exige apenas um roteador RASoC.

Para sistemas baseados em núcleos com 50 mil portas lógicas (Figura 6.22), a rede SoCIN mantém uma sobrecarga abaixo de 10% em todas as configurações, enquanto que a rede SPIN ultrapassa esse limite para sistemas com mais de 64 núcleos. Nesse caso, destaca-se que se houver um requisito rígido com relação a uma sobrecarga de área nessa faixa de valores, a rede SoCIN se apresenta como a solução mais adequada de rede-em-chip (sem levar em conta o desempenho das redes).

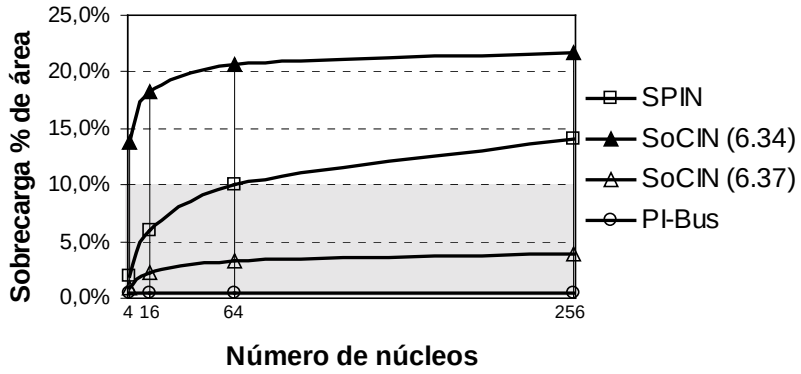


FIGURA 6.22 – Comparativo da sobrecarga de área do PI-Bus e das redes SPIN e SoCIN para sistemas baseados em núcleos com 50 mil portas lógicas (A_{core} = 2,75 mm^2) e palavra de 32 bits.

Por fim, para sistemas baseados em núcleos com 100 mil portas lógicas (Figura 6.23), ambas as redes-em-chip oferecem sobrecargas inferiores ao limite de 10%, sendo que a sobrecarga percentual de área de silício do barramento é próxima de zero, a da rede SoCIN é limitada a 2% e da rede SPIN cresce com o tamanho do sistema.

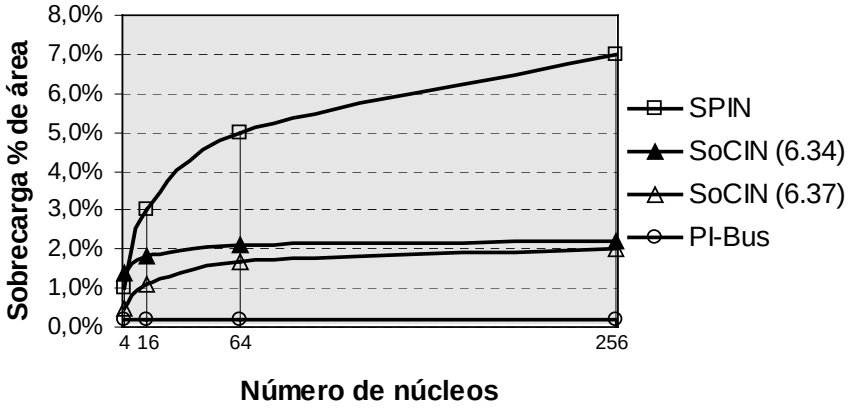


FIGURA 6.23 – Comparativo da sobrecarga de área do PI-Bus e das redes SPIN e SoCIN para sistemas baseados em núcleos com 100 mil portas lógicas (A_{core} = 5,5 mm^2) e palavra de 32 bits.

6.8 Considerações

Neste capítulo, foi apresentado o desenvolvimento de um conjunto de modelos analíticos para a estimativa da área de arquiteturas de comunicação para sistemas integrados. Os modelos foram baseados em algumas considerações que buscaram oferecer um grau de abstração, focalizando nos custos relacionados à parte operativa das arquiteturas, a qual, tipicamente, representa a seção mais cara de um sistema como os considerados no texto.

No que tange a modelagem da área das redes-em-chip, não foram considerados os custos referentes aos adaptadores de comunicação, dado que esses adaptadores são fortemente dependentes da interface de cada núcleo conectado ao sistema. Para a inclusão desses custos, três hipóteses poderiam ser consideradas: todos os núcleos possuírem interface compatível com o protocolo da rede, todos os núcleos implementarem uma interface padrão (eg. VCI ou OCP) ou cada núcleo possuir uma interface diferente (eg. proprietária, PI-Bus, VCI, OCP etc). No primeiro caso, não existe a necessidade do uso de adaptadores, enquanto que, no segundo caso, é preciso utilizar adaptadores para efetuar a tradução entre o protocolo padrão e o protocolo da rede. Por fim, no terceiro caso, cada núcleo poderia requerer de um a dois adaptadores, conforme o tipo de interface utilizado e os adaptadores disponíveis. Para se estender os modelos de estimativa de área de modo a considerar os adaptadores de comunicação, seria preciso avaliar os protocolos envolvidos e o modelo de comunicação da aplicação alvo de modo a determinar a infra-estrutura necessária, a qual pode incluir circuitos para compatibilização de tamanho de palavra, *buffers* do tipo FIFO e *buffers* de reordenamento de pacotes.

Quanto à aplicação dos modelos desenvolvidos neste capítulo, eles permitiram uma avaliação dos custos das redes consideradas, para diferentes configurações de sistema, comprovando que o custo de silício do barramento é mínimo e que a rede SoCIN apresenta uma sobrecarga de área menor que a da rede SPIN, exceto para uma configuração com quatro núcleos. Os modelos mostram-se uma ferramenta útil para a obtenção de uma estimativa inicial do custo de um sistema baseado em uma dessas redes.

7 Modelos Analíticos para a Estimativa de Desempenho de Arquiteturas de Comunicação para Sistemas Integrados

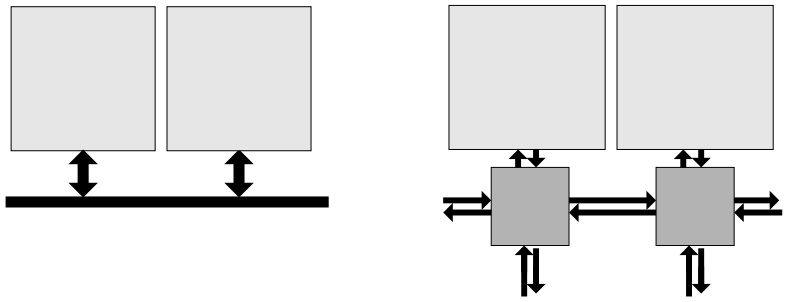
Este capítulo apresenta um conjunto de modelos analíticos que foram desenvolvidos com intuito de estabelecer as condições de contorno que determinam a aplicabilidade de uma ou outra arquitetura de comunicação para uma dada configuração de sistema. Os modelos são baseados em uma abordagem de alto nível que propicia uma primeira estimativa a respeito do desempenho em comunicação das arquiteturas consideradas. O primeiro conjunto de modelos visa estimar a carga capacitiva associada aos canais de comunicação, já que a freqüência de operação desses canais é função do valor dessa carga. É apresentado um exemplo de aplicação dos modelos e os resultados ilustram os efeitos do tamanho do sistema e das características dos componentes dos canais de comunicação (*buffers* e fio) na carga capacitiva dos fios desses canais. Após, descreve-se o segundo conjunto de modelos, o qual visa oferecer uma estimativa de alto nível do desempenho em comunicação da rede através da modelagem da latência da rede para executar cargas de comunicação sintéticas, as quais não representam o perfil de comunicação de alguma aplicação. Os modelos são aplicados a sistemas baseados no barramento e na árvore-gorda e as estimativas são comparadas com dados experimentais obtidos por simulação no CASS. Os resultados indicam para quais configurações de sistema e carga de comunicação o barramento e a árvore-gorda apresentam o melhor desempenho.

7.1 Modelos para a Estimativa do Desempenho dos Canais de Comunicação

O desempenho dos fios de um canal de comunicação é determinado pelos atrasos de propagação aos quais são submetidos os sinais aplicados a esses fios. Tais atrasos são função das cargas capacitivas dos fios, as quais dependem do número de núcleos conectados aos fios e dos comprimentos desses fios. A seguir são apresentados modelos para estimativa desse atraso de propagação.

7.1.1 Estruturas dos canais de comunicação

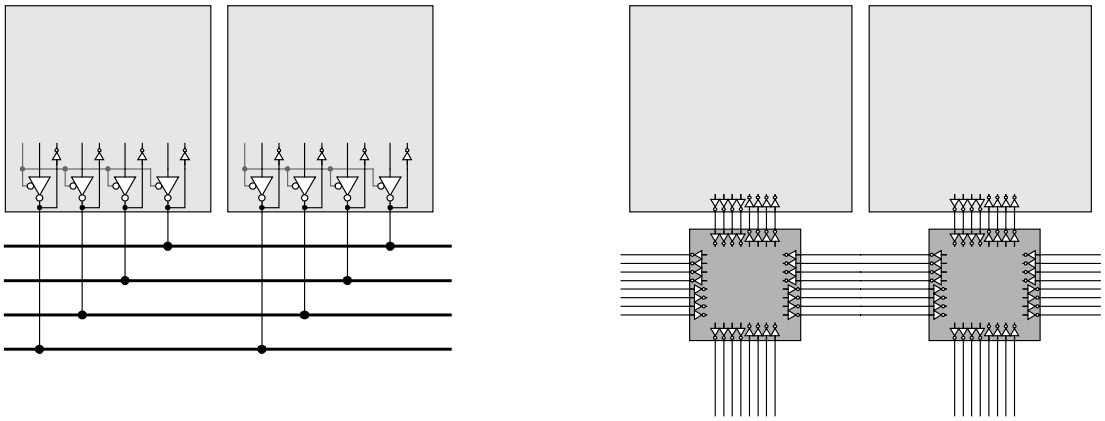
É sabido que os *n* núcleos de um sistema baseado em um barramento central compartilham os canais de comunicação do barramento e que os núcleos são conectados aos fios do barramento através de conexões multiponto bidirecionais, nas quais vários núcleos do sistema têm capacidade de injetar sinais em um mesmo fio (Figura 7.1.a). Já nas redes-em-chip, os núcleos são interligados por meio de canais chaveados de conexão ponto-a-ponto (Figura 7.1.b). Os canais entre um núcleo e um roteador (ou entre dois roteadores) são na verdade compostos por um par de canais unidirecionais em oposição, sendo que cada fio de um canal possui uma única fonte (núcleo ou roteador) injetando sinal no fio.



(a) (b)

FIGURA 7.1 – Canais das arquiteturas de comunicação: (a) multiponto no barramento; (b) ponto-a-ponto na rede-em-chip.

Conforme é ilustrado na Figura 7.2.a, a seguir, um núcleo é conectado ao barramento através de uma interface bidirecional *half-duplex*. Essa interface é constituída por um canal de entrada e por um canal de saída, sendo que ambos são conectados ao mesmo canal físico do barramento. Todos os núcleos conectados ao barramento recebem um sinal injetado por um outro núcleo e apenas um dos núcleos conectados ao barramento pode injetar sinais em um dado momento. Em outras palavras, todos os núcleos podem ler do barramento em um mesmo instante, mas apenas um núcleo pode escrever no barramento. Essa funcionalidade é garantida por meio de *buffers tri-state* colocados no canal de saída da interface do núcleo. Esses *buffers* são controlados de modo a estabelecer ou cancelar o contato elétrico do canal de saída do núcleo aos fios do barramento. O protocolo de uso do barramento garante que apenas uma dessas interfaces habilitará os seus *buffers tri-state*. Já no canal de entrada da interface dos núcleos, são incluídos *buffers* inversores que reforçam o sinal recebido e recuperam seu valor original, invertido inicialmente na saída do *buffer tri-state* do núcleo que gerou o sinal. As estruturas que implementam os canais ilustrados na Figura 7.2.a servirão de base para a modelagem do atraso devido às cargas capacitivas no barramento.

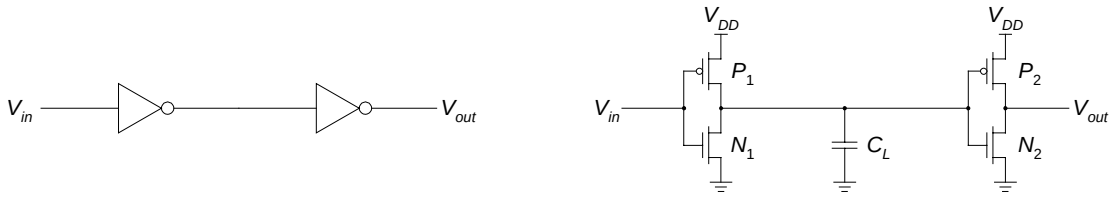


(a) (b)

FIGURA 7.2 – Canais das arquiteturas de comunicação: (a) multiponto no barramento; (b) ponto-a-ponto na rede-em-chip.

Por outro lado, em uma rede-em-chip, os núcleos são conectados aos roteadores (e estes entre si) através de uma interface bidirecional *full-duplex*, também composta por um canal de entrada e um canal de saída, mas cada um deles é conectado a um canal físico diferente. Em cada canal físico, são conectados um canal de saída e um canal de entrada de diferentes interfaces (emissor e receptor). Como não há risco de contenção, não é necessária a inclusão de *buffers tri-state*. Os canais de entrada e de saída podem incluir *buffers* (com saída invertida ou direta) para reforçar a intensidade do sinal transferido, dimensionados de acordo com o comprimento dos canais. Se a topologia da rede for irregular e os canais tiverem comprimentos variados, então caberá ao projetista dimensionar os *buffers* de modo que os canais possam operar na freqüência especificada.

Considerando uma cascata de dois inversores CMOS como o da Figura 7.3, a carga capacitiva (C_L) aplicada ao primeiro inversor é a soma das capacitâncias de dreno desse inversor (C_{dP1} e C_{dN1}) com as capacitâncias de porta do segundo inversor (C_{gP2} e C_{gN2}) e com a capacitância do fio (C_W) [RAB 96].



(a) (b)

FIGURA 7.3 – Dois inversores em cascata: (a) portas lógicas; (b) transistores NMOS e PMOS [RAB 96].

Dessa forma, obtém-se:

$$C_L = (C_{dP1} + C_{dN1}) + (C_{gP2} \text{ e } C_{gN2}) + C_W \qquad (7.1)$$

Esse modelo pode ser representado de uma maneira mais simples fazendo-se C_{d1} igual ao somatório das capacitâncias de dreno dos transistores NMOS e PMOS do primeiro inversor e C_{g2} a soma das capacitâncias de porta dos transistores NMOS e PMOS do segundo inversor. Ou seja:

$$C_L = C_{d1} + C_{g2} + C_W \qquad (7.2)$$

A freqüência máxima (f_{max}) de operação de um canal é inversamente proporcional ao período mínimo (T_{min}) suportado pelos fios desse canal. Esse período é determinado pelos tempos de propagação de subida (t_{pLH}) e de descida (t_{pHL}), os quais dependem da carga capacitiva (C_L) e de parâmetros da tecnologia utilizada (eg. mobilidade de portadores, dimensões dos *buffers*). Assumindo-se que esses parâmetros possam ser reunidos em uma única constante tecnológica (K_{tec}), define-se, então, em uma estimativa de alto nível, que o período mínimo de operação dos fios de um canal de comunicação será dado por:

$$T_{min} = C_L \propto K_{tec} \qquad (7.3)$$

Assumindo-se que K_{tec} seja o mesmo tanto para o barramento central como para as redes-em-chip, então o foco da estimativa deve ser dirigido à determinação da carga capacitiva associada a cada arquitetura, a qual irá determinar a freqüência de operação dos canais dessas arquiteturas. Dessa forma, a seguir, são apresentados modelos que visam obter uma estimava da carga capacitiva dos fios desses canais de comunicação.

7.1.2 Estimativa da carga capacitiva

Carga capacitiva devida aos *buffers*

Conforme é ilustrado na Figura 7.4, cada fio de um barramento é conectado às saídas dos *buffers tri-state* do canal de saída de cada núcleo, assim como às entradas dos inversores dos canais de entrada dos núcleos, sendo submetido às capacitância de dreno e de porta desses componentes.

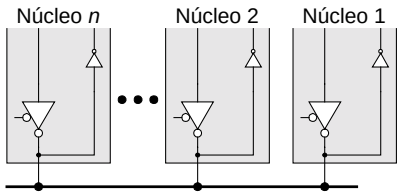


FIGURA 7.4 – Elementos conectados a um fio dos canais de um barramento.

Assumindo-se, por simplificação, que todos os n núcleos de um sistema integrado podem operar como mestre e escravo do fio considerado, existirão n capacitâncias de dreno e n capacitâncias de porta carregando cada fio. Conforme estabelecido no modelo representado na Figura 7.4, as saídas dos núcleos são conectadas aos barramentos através de *buffers tri-state*, então C_{d1} pode ser substituído por $C_{d,tri}$. Da mesma forma, como as entradas são baseadas em *buffers* de dois estados, C_{g2} pode ser substituído por $C_{g,buf}$. Assim, considerando-se todos os n núcleos do sistema, para os fios do barramento, (7.2) pode ser reescrita como:

$$C_{L,bus} = n \propto (C_{d,tri} + C_{g,buf}) + C_{W,bus} \quad (7.4)$$

Já em uma rede-em-chip, é assumido que cada fio conecta a saída de um *buffer* inversor à entrada de outro *buffer* inversor, conforme ilustrado na Figura 7.5.

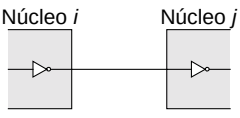


FIGURA 7.5 – Elementos conectados a um fio dos canais de uma rede-em-chip.

Dessa forma, para os fios de uma rede-em-chip, C_{d1} em (7.2) pode ser substituído por $C_{d,buf}$ e (7.2) reescrita como:

$$C_{L,noc} = C_{d,buf} + C_{g,buf} + C_{W,noc} \quad (7.5)$$

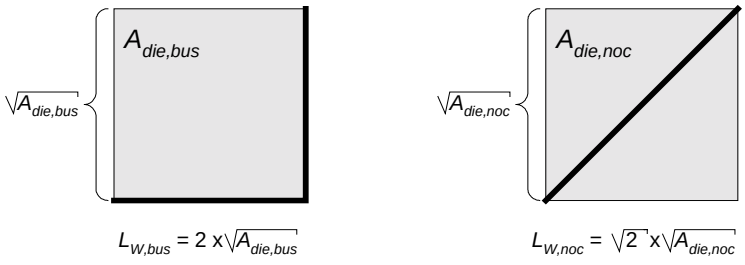
Carga capacitiva devida ao comprimento dos fios

Com relação à capacitância devida ao fio (C_W), ela é função da capacitância linear (C_m, medida em fF/νm) e do comprimento desse fio (L, medido em νm), sendo dada por:

$$C_W = L \propto C_m \tag{7.6}$$

O comprimento do fio varia com a arquitetura de comunicação utilizada e com o posicionamento dos núcleos e dos componentes da arquitetura de comunicação (roteadores, no caso das redes-em-chip). Tipicamente, assume-se que, no pior caso, o comprimento dos fios do barramento é dado por metade do perímetro do *die* [LAN 2000, ZEF 2002, ZEF 2002a]. Já o comprimento dos fios de uma rede em chip dependerá fortemente da topologia da rede e das dimensões do núcleo, fatores que terão impacto direto no posicionamento e no aproveitamento da área do *die*. Em uma grelha e em uma árvore-gorda, os canais tendem a ser curtos. Porém, em um toróide, alguns canais são bem mais longos que outros. Em uma primeira estimativa, toma-se uma condição de pior caso, na qual assume-se que o comprimento máximo dos fios dos canais de uma rede-em-chip é dado por dado por uma diagonal cruzando o *die* [ZEF 2002, ZEF 2002a]. Destaca-se que tal condição deprecia em muito a escalabilidade da rede e, na prática, o comprimento dos canais tende a ser menos sensível ao crescimento do sistema, conforme será melhor comentado posteriormente.

Sendo A_{die} a área de silício do chip, considerando-se a área devida aos núcleos e à sobrecarga associada à arquitetura de comunicação, a Figura 7.6 ilustra o caminho crítico (L) assumido para o barramento e para as redes-em-chip. Na figura, os dois *dies* são representados com o mesmo tamanho, porém, destaca-se que, para uma mesma configuração de sistema (quantidade e tamanho de núcleos), o *die* do sistema baseado em rede-em-chip será maior.



(a) (b)

FIGURA 7.6 – Caminho crítico máximo: (a) barramento; (b) rede-em-chip.

Considerando-se um sistema integrado com n núcleos homogêneos e quadrados (largura e altura iguais), cada um com aresta igual a B, então a área do *die* do sistema baseado no barramento será dada por:

$$A_{die,bus} = (n \propto B^2) + A_{bus} \tag{7.7}$$

onde A_{bus} corresponde à sobrecarga de área de silício devida ao barramento. O modelo (7.6) pode então ser reescrito como:

$$C_{W,bus} = 2 \propto [(n \propto B^2) + A_{bus}]^{1/2} \propto C_{m,bus} \tag{7.8}$$

De maneira análoga, a área do *die* para um sistema baseado em uma rede-em-chip é dada por:

$$A_{die,noc} = (n \propto B^2) + A_{noc} \quad (7.9)$$

onde A_{bus} corresponde à sobrecarga de área de silício devida aos roteadores da rede-em-chip. Dessa forma, para as redes-em-chip, considerando a condição de pior caso, (7.6) pode ser reescrita como:

$$C_{W,noc} = \{2 \propto [(n \propto B^2) + A_{noc}]\}^{1/2} \propto C_{m,noc} \quad (7.10)$$

Comparando-se (7.8) e (7.10), são identificadas duas diferenças importantes que vão impactar nas capacitâncias devidas aos fios dos canais em cada tipo de arquitetura. Considerando-se os piores casos assumidos, a capacitância da rede-em-chip é submetida a uma constante igual $2^{1/2}$, enquanto que, no barramento, essa constante é igual a 2. Contudo, destaca-se que a sobrecarga de área da rede-em-chip será maior que a sobrecarga do barramento. Conforme foi apresentado no Capítulo 6, a sobrecarga de área do barramento é quase nula para sistemas baseados em núcleos com 50 e 100 mil portas lógicas, enquanto que a sobrecarga de área das redes-em-chip não é desprezível.

Carga capacitiva total

A partir dos modelos desenvolvidos acima, (7.4) e (7.5) podem ser reescritas como:

$$C_{L,bus} = n \propto (C_{d,tri} + C_{g,buf}) \quad + \quad 2 \propto [(n \propto B^2) + A_{bus}]^{1/2} \propto C_{m,bus} \quad (7.11)$$

e

$$C_{L,noc} = C_{d,\,buf} + C_{g,\,buf} + \{2 \propto [(n \propto B^2) + A_{noc}]\}^{1/2} \propto C_{m,noc} \quad (7.12)$$

A capacitância linear (C_m), presente em (7.11) e (7.12) é dada por:

$$C_m = (width \propto C_{area}) + (0{,}4 \propto C_{peri}) + C_{coup} \quad (7.13)$$

Essa equação foi derivada de um modelo apresentado em [BEC 2003], sendo que *width* representa a largura do fio, C_{area} é a capacitância de área do fio (dada em fF/νm^2), C_{peri} é a capacitância de perímetro do fio (dada em fF/νm) e C_{coup} é a capacitância de acoplamento (dada em fF/νm). Se a largura do fio utilizada no barramento for igual à largura dos fios da rede-em-chip, então $C_{m,bus} = C_{m,noc}$.

7.1.3 Exemplo de aplicação dos modelos de estimativa de carga capacitiva

Os modelos acima foram aplicados na estimativa da carga capacitiva de sistemas baseados na tecnologia AMS .35νm CMOS com três camadas de metal [AUS 2001b] e na biblioteca de células da mesma tecnologia [AUS 2001a]. Como dados de entrada foram considerados os parâmetros obtidos da documentação disponível sobre o processo e sobre a biblioteca de células, onde:

$$C_{coup,met3\text{-}met3} = 0{,}087 \text{ fF/}\nu\text{m}$$

$$width_{min} = 0{,}5\ \nu\text{m}$$

$$C_{d,tri} = 0{,}007 \text{ pF}$$

$$C_{g,buf} = 0{,}006 \text{ pF}$$

Na documentação da biblioteca utilizada, a capacitância de dreno dos *buffers* de dois estados não é informada.

Com relação às capacitâncias de área e de perímetro, utiliza-se a abordagem adotada em [BEC 2003]. Como não é possível se saber a priori qual camada será implementada abaixo dos fios da interconexão em *Metal* 3 (poço, polisilício 1 ou 2, metal 1 ou 2), calcula-se um valor médio para essas capacitâncias através de uma media aritmética das mesmas. Essa abordagem implica na consideração de que a probabilidade de existir uma outra camada abaixo de *Metal* 3 é a igual para todas as camadas inferiores.

TABELA 7.1 – Cálculo das médias das capacitâncias para AMS *Metal* 3.

	-well	***-poly*1**	***-poly*2**	***-met*1**	***-met*2**	***-avg***
$C_{area,met3\text{-}}$	0,008	0,009	0,009	0,013	0,035	0,0148 fF/νm^2
$C_{peri,met3\text{-}}$	0,025	0,026	0,026	0,030	0,042	0,0298 fF/νm

Aplicando-se esses valores médios no cálculo da capacitância linear e considerando-se fios de largura mínima, obtém-se:

$$C_{\text{m}} = 0{,}010632 \text{ fF/}\nu\text{m}$$

Esses parâmetros foram aplicados em associação com os resultados de estimativa de sobrecarga obtidos no Capítulo 6 para sistemas com tamanho de palavra igual a 32 bis. Foram consideradas configurações com 4, 16, 64 e 265 núcleos com 50 mil portas lógicas por núcleo (1658 νm de lado).

Na Figura 7.7, é mostrado o efeito do aumento do número de núcleos na carga capacitiva dos canais do barramento central e das redes-em-chip. Primeiramente, observa-se que as curvas das redes-em-chip se sobrepõem, pois, embora elas possuam sobrecargas de área diferentes, o impacto dessa diferença de sobrecarga sobre a capacitância do fio é mínimo. Em segundo lugar, destaca-se o fato de que a carga capacitiva do barramento é significativamente maior que as das redes-em-chip. Por exemplo, para a configuração com 64 núcleos, a carga capacitiva do barramento é de 3,7 pF, ou seja, 82% maior que a das redes-em-chip. Isso pode ser traduzido em uma maior limitação à freqüência de operação e um maior consumo de energia por parte dos fios. Por isso, tipicamente, os barramentos centrais são limitados à interconexão de até 32 núcleos. Para sistemas maiores (ou até mesmo menores), costuma-se segmentar o barramento através de uma organização hierárquica na qual os barramentos são mais curtos e interligados por dispositivos do tipo ponte. Neste exemplo, são apresentados resultados para sistemas não usuais (com 64 ou 256 núcleos conectados a um mesmo barramento) apenas como uma forma de ilustrar o ganho proporcionado pelas redes-em-chip em relação ao barramento central.

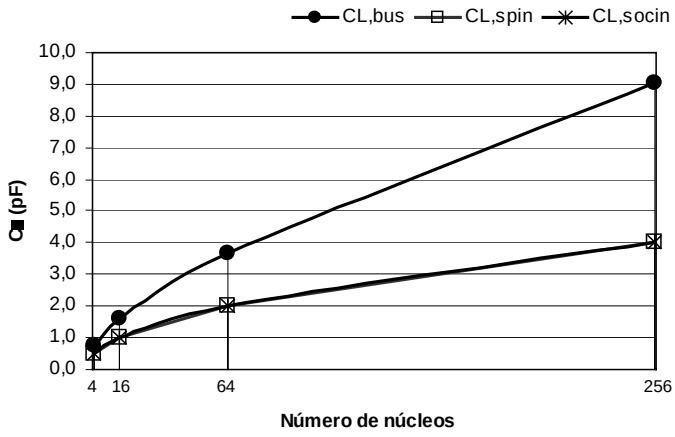


FIGURA 7.7 – Carga capacitiva dos canais do barramento e das redes-em-chip (núcleos com 50 mil portas lógicas).

Com relação ao comprimento dos canais das redes-em-chip, a consideração assumida é bastante desfavorável às NoCs. Por exemplo, em uma rede-em-chip baseada em uma grelha com topologia regular interconectando núcleos de mesmo tamanho, os canais de comunicação entre os roteadores possuem comprimento fixo, independente do número de núcleos no sistema. Essa abordagem é utilizada na rede CLICHÉ [KUM 2002], na qual os roteadores interconectam unidades denominadas regiões e cada região pode ser um único núcleo ou um subsistema com múltiplos núcleos interconectados por um barramento, conforme é ilustrado na Figura 7.8.

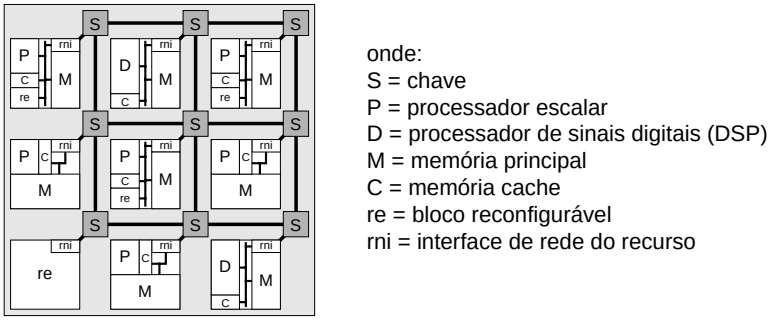


FIGURA 7.8 – Sistema baseado na rede CLICHÉ [KUM 2002].

O sistema desenhado na Figura 7.8 não representa o posicionamento real dos núcleos e canais. Contudo, ele permite estimar que, idealmente, os canais de comunicação entre roteadores possuem um comprimento quase uniforme e igual à lateral de um quadrado correspondente a uma região. Abordagem semelhante é assumida no toróide dobrado proposto por Dally e Towlles [DAL 2001], o qual é ilustrado na Figura 7.9. O sistema é organizado em regiões denominadas *tiles*, dentra dos quais são posicionados os núcleos. Os roteadores são implementados de maneira distribuída ao redor dos *tiles*, sendo que, para os autores, essa distribuição uniforme facilita o projeto dos *drivers* dos canais de comunicação e o reuso da rede. Como pode ser observado, na estrutura lógica da rede, os canais mais longos têm comprimento um pouco maior que valor correspondente ao de duas laterais de *tiles*. Novamente, a figura não representa o posicionamento físico, contudo ele deverá ser bastante próximo ao ilustrado e ainda sem o espaçamento representado entre os *tiles*.

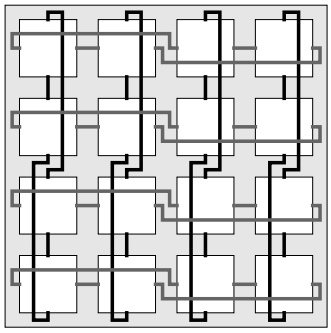


FIGURA 7.9 – Rede proposta em [DAL 2001].

As duas redes indicadas acima possuem topologia regular, o que facilita identificar o possível comprimento máximo dos fios. Mesmo para um sistema heterogêneo é possível supor que o comprimento do canal mais longo do sistema será menor do que a diagonal do *die*. Por exemplo, em um sistema heterogêneo baseado em uma grelha, poderia se considerar que o canal mais longo seria aproximadamente igual à maior dimensão (largura ou altura) do maior núcleo do sistema. Por outro lado, pode-se discutir a validade dessa consideração, argumentando-se que é difícil estimar o posicionamento real dos núcleos e dos roteadores e que, eventualmente, um núcleo pode ser posicionado longe do roteador ao qual ele está conectado. Em um exemplo, os roteadores poderiam ser posicionados no centro do *die* e os núcleos ao seu redor com alguns núcleos na periferia do *die*. Nesse caso, o comprimento máximo dos canais pode ser assumido como equivalente à metade da diagonal do *die*. Tal abordagem é sugerida na implementação de uma rede SPIN com 32 terminais SPIN [AND 2001].

Dadas as considerações acima, a Figura 7.10 ilustra o efeito do tamanho do sistema na carga capacitiva dos canais de comunicação da rede SoCIN quando os mesmos têm comprimento igual à diagonal do *die*, metade dessa diagonal, lateral do *die*, lateral de núcleo e duas laterais de núcleo (assume-se que todos os núcleos possuem 50 mil portas lógicas). Como pode ser observado, é desejável que a topologia utilizada favoreça a obtenção de um posicionamento que produza um comprimento de canal uniforme e independente do tamanho do sistema.

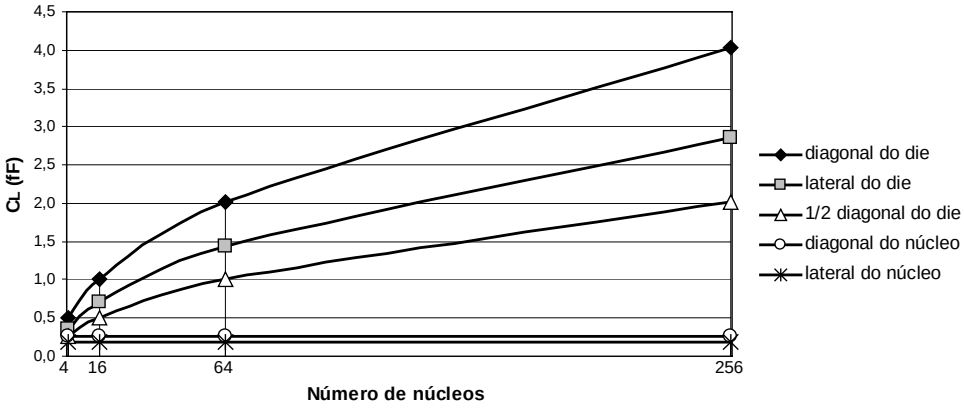


FIGURA 7.10 – Carga capacitiva dos canais de comunicação da rede SoCIN para diferentes comprimentos de canal.

Nos casos nos quais o comprimento dos canais da rede seja não-uniforme, é preciso dimensionar os *buffers* dos canais mais longos de modo a compensar a maior capacitância desses canais. Caso contrário, a freqüência de operação da rede será determinada pelo tempo de propagação do canal mais longo da rede.

Nos modelos e resultados mostrados acima, foi considerado que tanto os *buffers* de saída como a largura dos fios, em um sistema baseado em barramento, são iguais aos correspondentes em um sistema baseado em uma rede-em-chip. Porém, conforme é mostrado em [BEC 2003], para que os canais de um barramento operem na mesma freqüência dos canais de uma rede-em-chip, é necessário que suas linhas sejam mais largas e que os *buffers* de saída dos núcleos conectados ao barramento tenham uma maior capacidade de corrente (*strength*). Essa conclusão foi obtida através da aplicação do modelo de Sakurai [SAK 83] para avaliar o desempenho dos canais de um barramento e de duas redes-em-chip (toróide e árvore-gorda) interconectando sistemas com dez núcleos heterogêneos (com 7 mil a 280 mil portas lógicas, aproximadamente). Esses sistemas foram simulados via SPICE visando a tecnologia AMS .35νm CMOS com três camadas de metal [AUS 2001].

No modelo de Sakurai, um canal de comunicação é modelado como uma linha de transmissão com perdas em que, além da carga capacitiva, a resistência equivalente do amplificador (*rt*) e a resistência da linha (*R*) também são consideradas para estimar o tempo de propagação nos fios. Além disso, a resistência e a capacitância da linha são distribuídas ao longo de seu comprimento, enquanto que no modelo aqui considerado a capacitância é concentrada em um único parâmetro.

Mais importante do que a conclusão a respeito do aumento das dimensões das linhas e dos *buffers* de saída em um sistema baseado em barramento está a avaliação feita pelos autores de que existe um limite para essas melhorias. Primeiramente, para uma linha de largura mínima (0,5 νm), o alargamento do canal dos transistores dos *buffers* de 9/0,3 a 36/0,3 νm (*W*/*L* PMOS) produz um aumento na freqüência de operação. Porém, acima dessa faixa, o ganho na freqüência de operação é mínimo. Utilizando-se uma linha com largura maior (4,0 νm), a resistência dos fios é reduzida e o limite de saturação da freqüência é estendido. Entretanto, destaca-se que a freqüência de operação obtida com *buffers* maiores (72/0,3 νm) e linhas de largura igual a 4,0 νm é pouco superior à freqüência dos canais de uma árvore-gorda baseada em *buffers* quatro vezes menores (18/0,3 νm) e linhas de largura mínima (0,5 νm).

As conclusões destacadas acima são aproveitadas neste texto para ilustrar o efeito das melhorias aplicadas ao barramento na carga capacitiva das suas linhas. Aumentando-se a largura das linhas, aumenta-se também a carga capacitiva, conforme é ilustrado na Figura 7.11. A linha com largura igual a 4 νm, considerada de largura ótima em [BEC 2003], apresenta uma carga capacitiva superior, variando de 33 a 50% do valor da capacitância da linha de largura mínima (0,5 νm). Já na Figura7.12, é ilustrado o impacto do aumento da capacidade dos *buffers* (alargamento dos canais de seus transistores) em associação com o aumento da largura das linhas. A carga capacitiva da configuração de maiores dimensões (4,0 νm e 4∞) varia de 45 a 56% em relação ao valor da capacitância da configuração mínima, para os diferentes tamanhos de sistema.

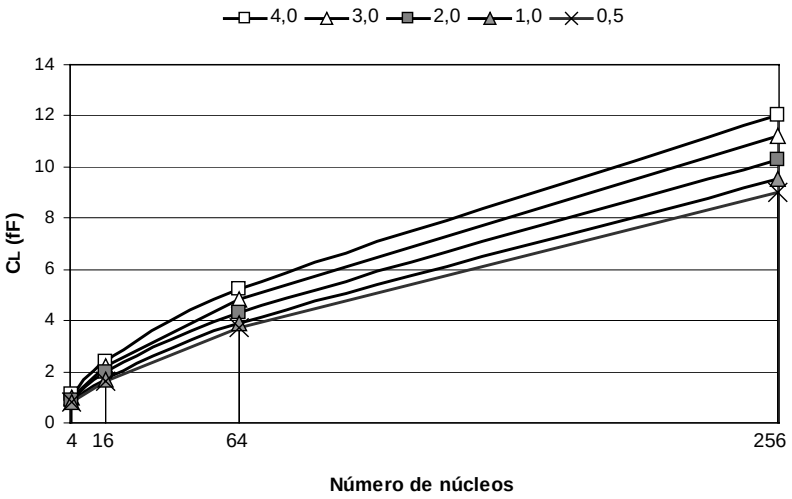


FIGURA 7.11 – Aumento da carga capacitiva do barramento com a largura das linhas (0,5 a 4,0 νm).

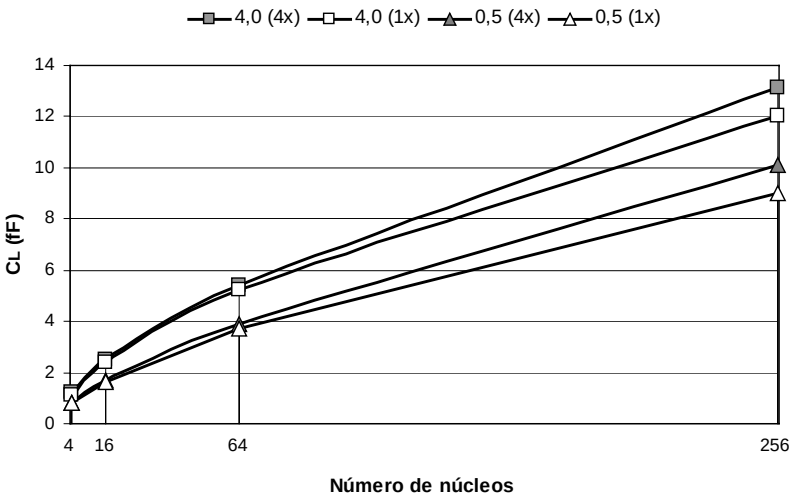


FIGURA 7.12 – Aumento da carga capacitiva do barramento com a largura das linhas (0,5 e 4,0 νm) e com a capacidade dos *buffers* (1∝ e 4∝).

Em resumo, os resultados apresentados nesse exemplo demonstram que os canais de comunicação das redes-em-chip consideradas apresentam carga capacitiva inferior em relação aos canais de um barramento. Além de propiciar uma maior freqüência de operação, uma carga capacitiva menor produz benefícios quanto a um consumo de energia mais reduzido, aspecto fundamental em muitos sistemas integrados destinados a aplicações embarcadas alimentadas por baterias. O estudo das questões relacionadas ao consumo de energia foge do escopo proposto para esta tese. No entanto, pode-se apontar trabalhos nos quais esse assunto é discutido e/ou avaliado, como o próprio relatório de pesquisa de Beck Filho e Carro [BEC 2003], alguns artigos publicados recentemente no cenário das redes-em-chip [BEN 2001, DAL 2001] e outros no contexto mais amplo da interconexão intra-chip [LAN 2000, HO 2001].

7.2 Modelos para a Estimativa da Latência

Nesta seção, são apresentados modelos que visam propiciar a obtenção de uma estimativa em alto nível do desempenho em comunicação de arquiteturas de comunicação para sistemas integrados. Esses modelos são baseados em modelos de latência com carga zero, os quais são estendidos de modo a incluir condições que consideram uma carga de comunicação particular e o paralelismo de algumas redes. Os modelos são aplicados e seus resultados são comparados com os obtidos por simulação.

7.2.1 Modelos de latência com carga zero

O modelo de latência com carga zero para o barramento central é obtido considerando-se o tempo associado à arbitragem do barramento (T_{arb}) e o tempo para a transferência da mensagem. Assumindo-se que o barramento requer um ciclo de relógio para transferir cada palavra de largura igual à largura da via de dados do barramento (W_{data}), a latência para transferir uma mensagem com m palavras no barramento é expressa por:

$$t_{msg,bus} = T_{arb} + (T_{bus} \propto m) \tag{7.14}$$

Um modelo de latência com carga zero para as redes-em-chip pode ser obtido a partir do modelo apresentado em [DUA 97] para a estimativa da latência em redes de interconexão baseadas no chaveamento *wormhole*. Segundo aquele modelo, a latência para transferir uma mensagem com M bits entre um emissor e um receptor separados por D enlaces do tipo roteador-roteador com canal de dados igual a W_{data} é dada por:

$$t_{msg,wormhole} = (t_r + t_s + t_w) \propto D \quad + \quad \max(t_s,t_w) \propto [M/W_{data}] \tag{7.15}$$

onde t_r é o tempo gasto pelo roteador para efetuar o escalonamento do pacote (roteamento e a arbitragem), t_s é o atraso de propagação devido aos canais internos do roteador e t_w é o atraso de propagação associado aos canais dos enlaces da rede. O tamanho da carga útil do pacote é definifo por M/W_{data} e indica o número de flits necessários para a transferência dos M bits do pacote. Caso o canal de dados de uma rede-em-chip seja da mesma largura do canal de dados do barramento central e for preciso incluir endereço e dado no pacote, para uma mensagem com m palavras de dado, a carga útil conterá $2 \propto m$ flits.

Em (7.15), a primeira expressão computa a latência para transferir o cabeçalho do pacote, enquanto que o segundo termo determina o tempo gasto pela carga útil do pacote para chegar ao destinatário, seguindo o cabeçalho em um modo *pipeline*. O tempo de ciclo desse *pipeline* é determinado pelo maior valor entre os tempos de propagação nos canais internos do roteador (t_s) e nos canais dos enlaces da rede (t_w).

O modelo (7.15) é estendido de modo a considerar as características de operação das redes-em-chip. Nesse novo modelo, o parâmetro D não é tomado como o número de enlaces roteador-roteador no caminho da mensagem, mas sim como o número de roteadores nesse caminho. Isso se justifica pelo fato de que a latência (em ciclos de relógio) adicionada por cada roteador é maior que a latência adicionada pelos enlaces intra-chip. Além disso, para aplicar o modelo na computação da latência com carga zero

para as redes-em-chip, é considerado que cada roteador é organizado em um *pipeline* com um ciclo de relógio (T_{noc}) definido pelo maior atraso na rede.

Em [PEH 2001], Peh e Dally descrevem um roteador *wormhole* organizado em um *pipeline* com três estágios: roteamento, arbitragem e chaveamento. Cada estágio no pipeline corresponde a um ciclo de relógio e, como é mostrado na Figura 7.13, os dois primeiros estágios (roteamento e arbitragem) correspondem ao parâmetro t_r em (7.15), enquanto que o terceiro estágio corresponde ao parâmetro t_s.

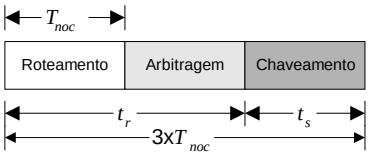


FIGURA 7.13 – Estágios do *pipeline* [PEH 2001].

Se for levado em conta o período de operação do *pipeline* (parâmetro T_{noc}) e a latência para o cabeçalho encher o *pipeline,* obtém-se um novo parâmetro denominado υ_{hoc}. Esse parâmetro representa o número de ciclos gastos pelo cabeçalho do pacote para avançar de um roteador a outro na ausência de contenção. Por exemplo, υ_{hoc} pode ser definido como igual a quatro, onde três ciclos de relógio são devidos ao *pipeline* do roteador e um ciclo é devido ao atraso no enlace entre os roteadores. Dessa forma, (7.15) pode ser reescrita como:

$$t_{msg,pipelined\text{-}wormhole} = T_{noc} \propto [(\upsilon_{hoc} \propto D) + M/W_{data}] \quad (7.16)$$

A equação (7.16) considera a latência para um pacote atravessar os roteadores da rede, porém não inclui a latência devida à inserção e à extração do cabeçalho do pacote realizadas pelas operações de empacotamento e desempacotamento. Essas operações não são normalmente necessárias em arquiteturas baseadas em um barramento, pois, tipicamente, os núcleos utilizados são compatíveis com o protocolo do barramento. Contudo, elas são comuns em sistemas baseados em redes-em-chip, pois a maioria dos núcleos é orientada a outros protocolos de comunicação. Dessa forma, para se obter uma estimativa mais realista, esses custos são incluídos em (7.16), assumindo-se que cada operação consome um ciclo de relógio para ser realizada, de onde se obtém o modelo abaixo.

$$t_{msg,noc} = T_{noc} \propto [2 + (\upsilon_{hoc} \propto D) + M/W_{data}] \quad (7.17)$$

7.2.2 Modelos de latência com carga não-nula

As equações (7.14) e (7.17) são modelos simples que descrevem a latência para transferir uma mensagem em uma condição de carga zero em arquiteturas do tipo barramento central e rede-em-chip, respectivamente. Essas equações podem ser estendidas incluindo condições de carga não-nula de modo a propiciar uma primeira estimativa do desempenho em comunicação dessas arquiteturas para diferentes configurações de tamanho de sistema e carga de comunicação. Considerando-se uma situação na qual cada um dos *n* núcleos no sistema deseja enviar uma mensagem aos outros *n*–1 núcleos, é computado o número de ciclos gastos pela rede para entregar

$n \propto (n-1)$ mensagens (ou pacotes). Considerando-se, ainda, que tal carga de comunicação represente a condição de pior caso, pode-se obter cargas mais leves aplicando-se um fator de redução (R_d) que conduz ao modelo do número de mensagens (N_{msg}) a serem entregues pela rede:

$$N_{msg} = n \propto (n-1) \propto R_d \tag{7.18}$$

Na condição de pior caso, apresentada acima, cada núcleo envia mensagens aos outros n–1 núcleos do sistema. Em uma carga de comunicação mais leve, cada núcleo enviaria mensagens a um menor número de núcleos, o que pode ser definido por um parâmetro que indique o número de médio parceiros de comunicação de cada núcleo para aquela carga (n_{load}). Disso obtém-se o modelo abaixo:

$$R_d = n_{load} / (n-1) \tag{7.19}$$

Dessa forma, a latência total requerida pelo barramento para entregar N_{msg} mensagens em um sistema com n núcleos, submetido a um fator de redução de carga definido por R_d é dada por:

$$t_{bus} = N_{msg} \propto R_d \propto [T_{arb} + (T_{bus} \propto m)] \tag{7.20}$$

Contudo, na maioria das arquiteturas de barramento central, a arbitragem pode ser realizada em paralelo com uma transferência de dado em um modo *pipeline*. Desse modo, apenas o primeiro ciclo de arbitragem precisa ser contabilizado, resultando em:

$$t_{bus} = T_{arb} \ + \ (N_{msg} \propto R_d \propto m) \propto T_{bus} \tag{7.21}$$

Substituindo-se (7.19) e (7.20) em (7.21) obtém-se o modelo de latência do barramento para o tipo de carga considerado, o qual é dado por:

$$t_{bus} = T_{arb} \ + \ (n \propto n_{load} \propto m) \propto T_{bus} \tag{7.22}$$

Para as redes-em-chip, aplicando-se o mesmo procedimento, obtém-se o modelo abaixo:

$$t_{noc} = n \propto n_{load} \propto T_{noc} \propto [2 \ + \ (\upsilon_{noc} \propto D) \ + \ M/W_{data}] \tag{7.23}$$

o qual ainda deve ser estendido para incluir o paralelismo oferecido pelas redes-em-chip e que permite a transferência simultânea de múltiplas mensagens. Esse paralelismo é expresso sob a forma de um parâmetro (σ_{noc}) utilizado como divisor do valor gerado por (7.23). Disso obtém-se:

$$t_{noc} = n \propto n_{load} \propto T_{noc} \propto [2 \ + \ (\upsilon_{noc} \propto D) \ + \ M/W_{data}] \ / \ \sigma_{noc} \tag{7.24}$$

O paralelismo realmente explorável em uma rede-em-chip depende da sua topologia, da carga oferecida pela aplicação e da localidade de comunicação da aplicação, a qual pode ser derivada da distância média entre os emissores e receptores das transferências realizadas na rede. Em [ZEF 2002a], o parâmetro σ_{noc} é estimado como:

$$\sigma_{noc} = n \propto R_d \propto D \tag{7.25}$$

7.2.3 Exemplo de aplicação dos modelos de estimativa de latência

Os modelos acima apresentados foram aplicados na estimativa da latência de comunicação de um barramento central e de uma árvore-gorda. Os resultados da estimativa foram comparados com os obtidos por meio da simulação do barramento PI-Bus e da rede SPIN no simulador CASS [PET 97]. Após, os modelos analíticos foram aplicados na comparação de estimativas de desempenho do barramento PI-Bus e da rede SPIN em diferentes configurações de sistema.

Sistemas modelados

Os sistemas modelados no CASS foram baseados em dois tipos de núcleo com interface VCI: um gerador de tráfego (iniciador-VCI) e uma memória RAM (alvo-VCI). Em um sistema com n núcleos, metade dos núcleos é de um tipo e outra metade é de outro tipo. Desde que iniciadores-VCI só podem se comunicar com alvos-VCI e vice-versa, a carga de comunicação no sistema é limitada a $n/2$ núcleos para cada núcleo do sistema (ou seja $n_{load,max} = 0{,}5 \propto n$). Logo, no modelo de carga utilizado, os geradores de tráfego são configurados a emitir uma mensagem de requisição de tamanho m a cada uma das $n/2$ memórias RAM do sistema e estas devem enviar uma mensagem de resposta com o mesmo tamanho aos $n/2$ geradores de tráfego. Logo, para a carga considerada, o número de mensagens será dado por:

$$N_{msg} = n \propto n_{load} \;=\; n \propto 0{,}5 \propto n = 0{,}5 \propto n^2 \qquad (7.27)$$

Por exemplo, em um sistema com quatro núcleos serão trocadas oito mensagens de tamanho m. Na Figura 7.14.a, são ilustrada as quatro mensagens emitidas pelos geradores de tráfego (GT), enquanto as mensagens emitidas pelas memórias RAM são mostradas na Figura 7.14.b. A configuração ilustrada na figura é válida tanto para sistemas baseados no barramento PI-Bus quanto na rede SPIN.

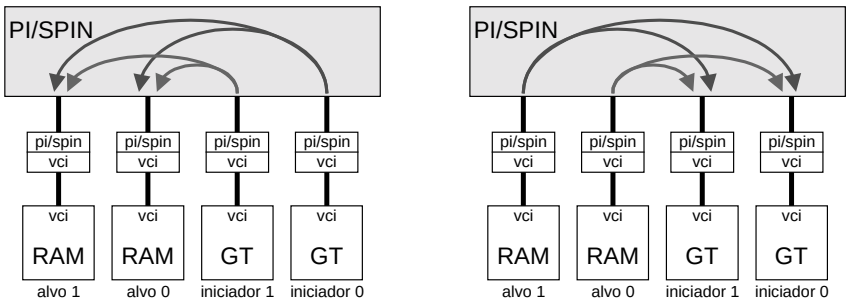


FIGURA 7.14 – Mensagens emitidas em um sistema com quatro núcleos: (a) originadas pelos iniciadores; (b) originadas pelos alvos.

Resultados analíticos ∝ resultados experimentais

Os gráficos a seguir ilustram a comparação entre os resultados experimentais e analíticos para sistemas com 4, 8, 16 e 32 núcleos trocando mensagens de tamanho mínimo (m = 1). É considerado o pior tamanho de mensagem para uma rede-em-chip, pois a sobrecarga devida ao cabeçalho é a mais significativa (50% da carga útil, a qual é composta por dois flits: endereço e dado). Na estimativa de desempenho do barramento

foi assumido o modelo em *pipeline* (7.22), com T_{arb} igual a três ciclos de relógio do barramento ($3 \propto T_{bus}$).

Na Figura 7.15, são mostrados os resultados analítico e experimental para o barramento PI-Bus. No eixo vertical à esquerda (em escala logarítmica), é indicado o número de ciclos para a entrega das mensagens, enquanto que o erro percentual absoluto entre os valores estimado analiticamente e medido por simulação é indicado no eixo vertical à direita (em escala linear). Como pode ser observado, o erro percentual é de 21% para o sistema com quatro núcleos, decrescendo até 0,4% para o sistema com 32 núcleos. Para a obtenção desse gráfico, os parâmetros utilizados foram: T_{bus} = 1, T_{arb} = 3 ciclos, W_{data} = 32 bits, m = 1 palavra de dado e n = 4, 8, 16 e 32 núcleos.

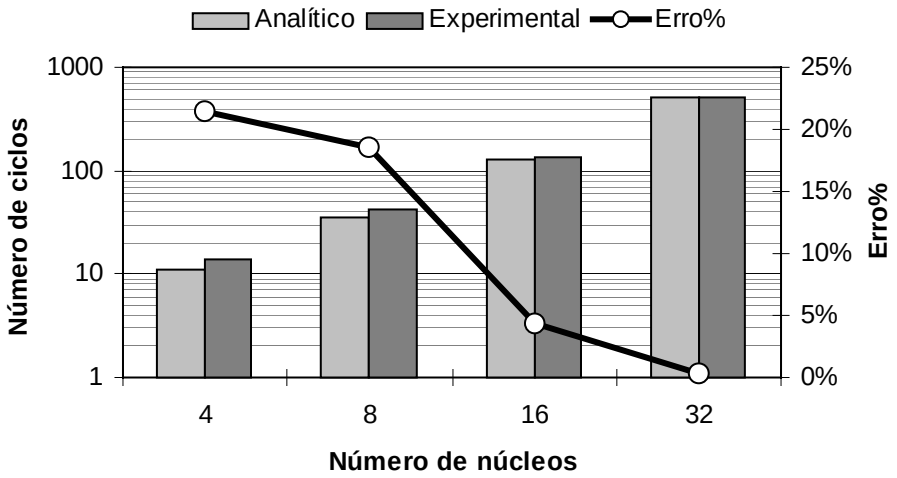


FIGURA 7.15 – Latências medida e estimada para o barramento PI-Bus.

Na comparação do modelo analítico para a rede SPIN com os resultados experimentais, foi considerado que, em um mesmo roteador, são conectados apenas núcleos do mesmo tipo (iniciador ou alvo) e que a rede é dividida em duas metades. Os iniciadores (GTs) são posicionados em uma das metades, enquanto que os alvos (RAMs) são posicionados na outra metade, conforme é ilustrado no exemplo da Figura 4.17. Essa configuração estabelece uma condição de pior caso em que a localidade na comunicação é dada pela maior distância entre dois núcleos na rede. No exemplo da figura, D é igual a 2 dois roteadores ou $\log_2(n/2)$.

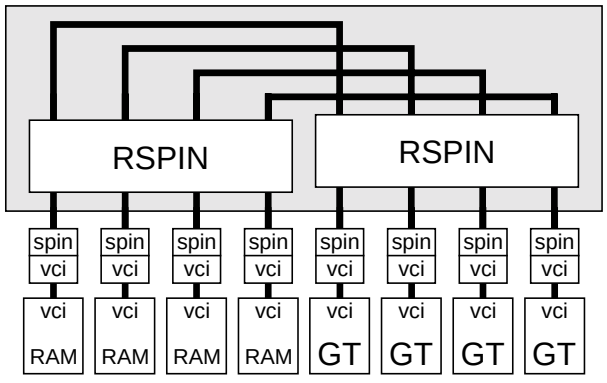


FIGURA 7.16 – Posicionamento de iniciadores e alvos em uma rede SPIN com oito terminais.

Os resultados analítico e experimental para a rede SPIN são mostrados na Figura 7.17. Nota-se que o erro percentual absoluto é nulo para sistemas com 4 e 8 núcleos. Porém, para sistemas maiores, com 16 e 32 núcleos, o erro é igual a 7 e a 18%, respectivamente. Para a obtenção desse gráfico, os parâmetros utilizados foram: $T_{noc} = 1$, $\upsilon_{noc} = 4$ ciclos, $W_{data} = 32$ bits, $m = 1$ palavra de dado, $M = 64$ bits ($M/W = 2$), $D = \log_2(n/2)$ e $n = 4$, 8, 16 e 32 núcleos.

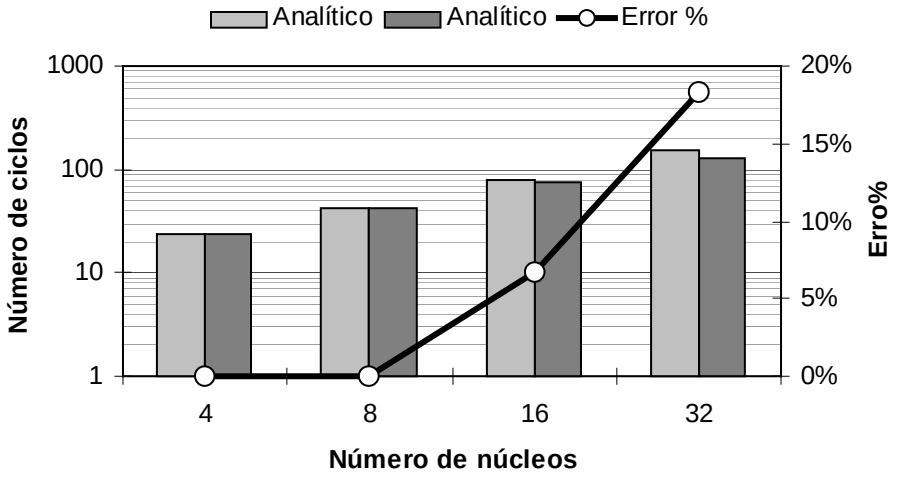


FIGURA 7.17 – Latências medida e estimada para a rede SPIN.

A Figura 7.18 ilustra os erros percentuais entre os resultados estimados e os medidos por simulação para diferentes tamanhos de mensagem (m = 1, 2, 3, 4 e 8) em sistemas com quatro e oito núcleos. O gráfico mostra que o erro percentual absoluto é inferior a 45%, sendo que, na média, seu valor é de 20,5%.

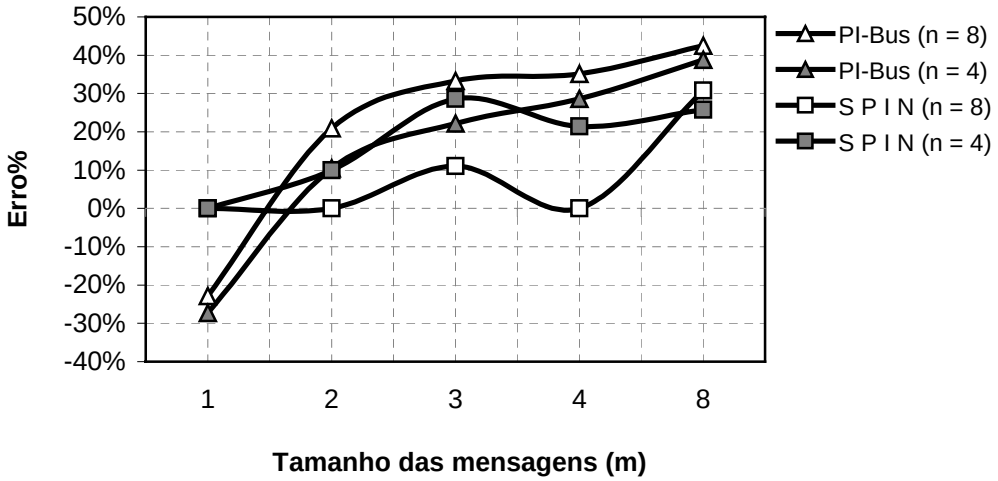


FIGURA 7.18 – Erros percentuais entre os resultados estimados e os dados medidos por simulação para diferentes tamanhos de mensagem (m) e número de núcleos (n).

Esses valores de erro percentual podem ser considerados aceitáveis, pois os modelos analíticos desenvolvidos baseiam-se em uma abordagem de alto nível. Além disso, esses modelos pretendem apenas fornecer uma estimativa do desempenho em

comunicação das redes consideradas de modo a permitir uma avaliação inicial sobre a arquitetura mais adequada para uma dada configuração de sistema e carga de trabalho.

Comparação das latências do barramento PI-Bus e da rede SPIN

Nos gráfico da Figura 7.19, é ilustrada a comparação das estimativas de latência para o PI-Bus e para a rede SPIN correspondentes às configurações apresentadas acima (sendo que $T_{bus} = T_{noc}$). Como pode ser observado, para sistemas com 4 e 8 núcleos, o barramento consome menos ciclos para a entrega de todas as mensagens. Porém, nos sistemas maiores (16 e 32 núcleos), a rede SPIN demanda um número menor de ciclos para executar a carga de comunicação aplicada.

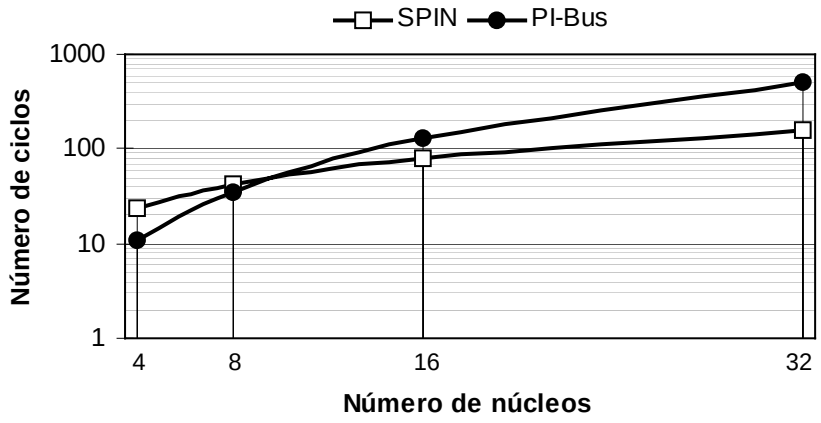


FIGURA 7.19 – Latências estimadas para o barramento PI-Bus e para a rede SPIN para diferentes tamanhos de sistema com m = 1.

Como pode ser observado, o comportamento das curvas na Figura 7.19 é bastante próximo das curvas obtidas com aplicação dessa carga em sistemas equivalentes no simulador CASS, conforme é mostrado na Figura 7.20. Destaca-se que existe uma diferença de valores entre os dois gráficos a qual decorre do erro inerente da estimativa em alto nível, conforme visto na Figura 7.15 e na Figura 7.17 .

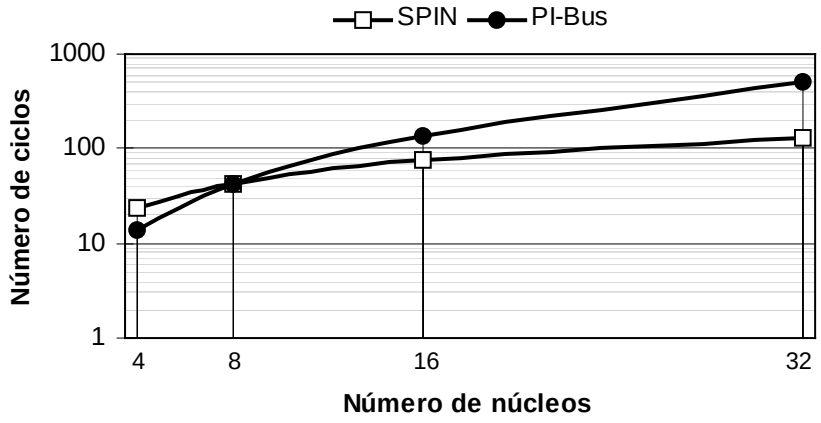


FIGURA 7.20 – Latências medidas por simulação para o barramento PI-Bus e para a rede SPIN para diferentes tamanhos de sistema com m = 1.

A Figura 7.21 apresenta uma curva que indica a relação entre as latências do barramento PI-Bus e da rede SPIN para diferentes tamanhos de sistema. Ela é obtida pela aplicação da expressão $t_{bus}/t_{noc} - 1$, de modo que seu resultado é negativo quando t_{bus} é menor que t_{noc} ou positivo quando t_{bus} é maior que t_{noc}. Com isso, é possível identificar, visualmente, em uma única curva, para qual tamanho de sistema o desempenho da rede SoCIN apresenta um ganho positivo relação ao desempenho do barramento PI-Bus, para o modelo de carga considerado. Conforme pode ser observado, o ganho de desempenho é positivo para sistemas com um número de núcleos superior a oito.

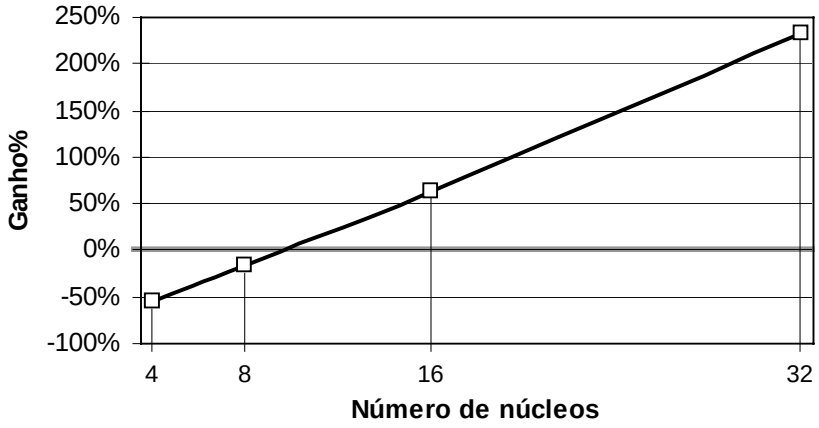


FIGURA 7.21 – Ganho percentual estimado da rede SPIN sobre o barramento PI-Bus.

Impacto do tamanho da mensagem na latência da rede SPIN

Com relação aos resultados acima, o desempenho da rede só supera o do barramento quando o paralelismo é incrementado. Isso reflete a escalabilidade da rede-em-chip, cuja largura de banda agregada aumenta com o crescimento do sistema. O desempenho da rede só não é superior pois mensagens muito curtas implicam em duas sobrecargas importantes. Primeiramente, desde que cada pacote é constituído por apenas dois flits (endereço + dado), o cabeçalho da rede gera uma sobrecarga de informação de 50%. Além disso, quanto menor é o tamanho dos pacotes transferidos em uma rede chaveada, maior é o impacto do tempo devido ao escalonamento dos pacotes nos roteadores (roteamento e arbitragem). Conforme discutido em [DUA 97], o tempo gasto no escalonamento é amortizado entre mais flits quando as mensagens são mais longas. Ainda, segundo os autores, os flits da carga útil avançam mais rapidamente que o cabeçalho, pois este tem que ser escalonado a cada roteador. Esses efeitos são ilustrados na Figura 7.22, onde são apresentadas curvas de ganho de desempenho para diferentes tamanhos de mensagem (à exceção de m, os parâmetros são os mesmos utilizados anteriormente). Como pode ser observado, o ganho de desempenho da rede SPIN é positivo em sistemas com oito núcleos (e maiores) para mensagens com tamanho igual a dois ($m = 2$, $M/W_{data} = 4$) ou maiores. Para mensagens muito curtas ($m = 1$), a latência da rede só é menor que a do barramento em sistemas com 16 ou mais núcleos.

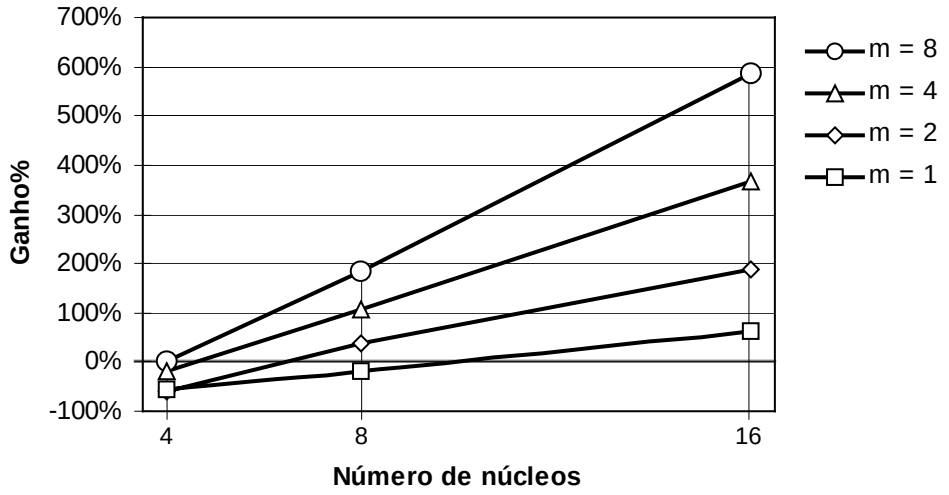


FIGURA 7.22 – Ganhos percentuais da latência da rede SPIN para diferentes tamanhos de mensagem (m = 1 a 8).

Impacto do peso da carga na latência do PI-Bus e da rede SPIN

A carga aplicada nos exemplos acima pode ser caracterizada como tendo um peso de 50%, pois cada núcleo emite mensagens para metade dos núcleos do sistema. Reduzindo-se o número de destinatários (n_{load}), cargas mais leves podem ser obtidas de modo a avaliar a latência do barramento e da rede nessas condições. Por exemplo, no gráfico da Figura 7.23, é ilustrado o ganho percentual da rede SPIN sobre o barramento PI-Bus para cargas de 50, 40, 30, 20 e 10% baseadas em mensagens com apenas uma palavra de dado (m = 1). Como pode ser observado, para cargas mais leves, o desempenho da rede e o ganho percentual em relação ao barramento diminuem com a redução da carga. Para uma carga de 10%, o barramento apresenta-se como a melhor solução em todas as configurações de sistema.

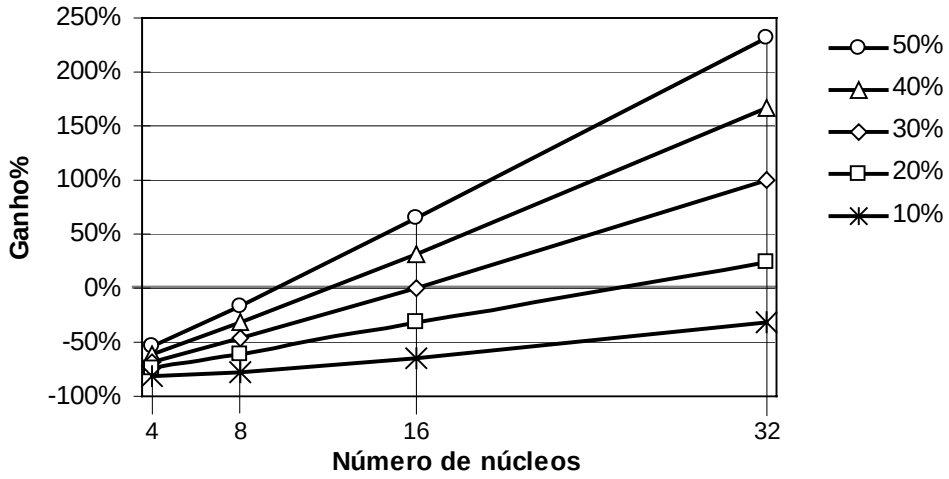


FIGURA 7.23 – Ganhos percentuais da latência da rede SPIN para mensagens de tamanho 1 e diferentes cargas (50, 40, 30, 20 e 10%).

No gráfico da Figura 7.24, é ilustrado o mesmo efeito quando as mensagens incluem quatro palavras de dado ($m = 4$). Como pode ser observado, o aumento do tamanho das mensagens incrementa o desempenho da rede, tornando-a a solução de menor latência para sistemas com 32 núcleos em qualquer configuração de carga (inclusive para a mais leve). Já para os sistemas menores (com 4 e 8 núcleos), o barramento apresenta-se como a melhor solução em desempenho para cargas de 10% e de 20%.

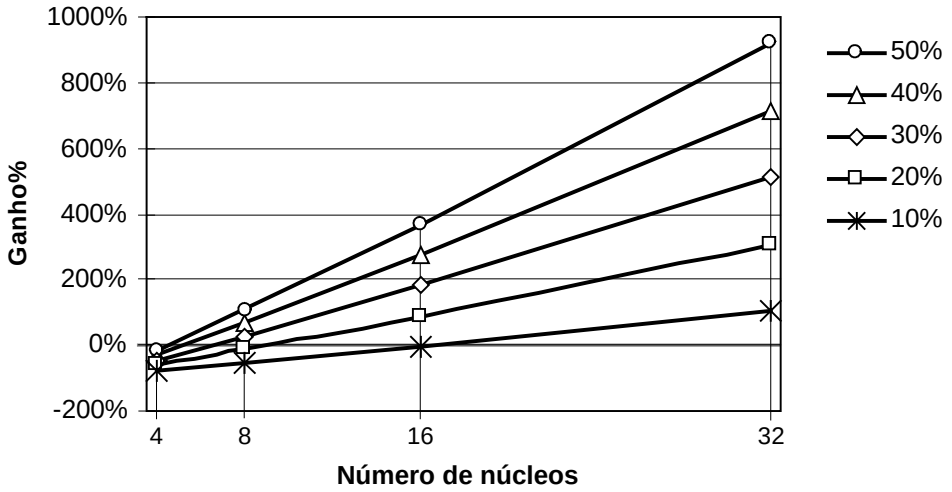


FIGURA 7.24 – Ganhos percentuais da latência da rede SPIN para mensagens de tamanho 4 e diferentes cargas (50, 40, 30, 20 e 10%).

Os resultados acima demonstram que, para configurações de carga leve e de mensagens curtas, o barramento oferece um desempenho superior ao da rede-em-chip. Contudo, com o aumento da carga e do tamanho das mensagens, a rede passa a tomar vantagem do seu paralelismo e da sua maior largura de banda.

Impacto da largura do canal de dados na latência da rede SPIN

Os modelos de latência foram também aplicados na avaliação do impacto da largura do canal da rede na execução da carga de comunicação modelada. Fazendo-se m igual a quatro (tamanho de uma linha de cache de processador RISC de 32 bits), variou-se o parâmetro W_{data} de modo a determinar qual a largura mínima de canal que permite à rede-em-chip sustentar uma latência inferior à do barramento PI-Bus com palavra de 32 bits.

No exemplo da Figura 7.25, para sistemas com quatro núcleos, apenas os canais com largura igual a 64 bits ou maior superam o desempenho do barramento, requerendo um número menor de ciclos para entregar as mensagens (ganho % positivo). Já em sistemas com oito núcleos, todas as configurações acima de 8 bits apresentam latência inferior à do barramento. Por fim, em sistemas com 16 núcleos (ou maiores), qualquer largura de canal oferece latência menor que à do PI-Bus. Esse exemplo ilustra que os modelos de latência oferecem a possibilidade de se realizar uma estimativa da configuração de menor custo da rede-em-chip que sustente um desempenho superior ao do barramento.

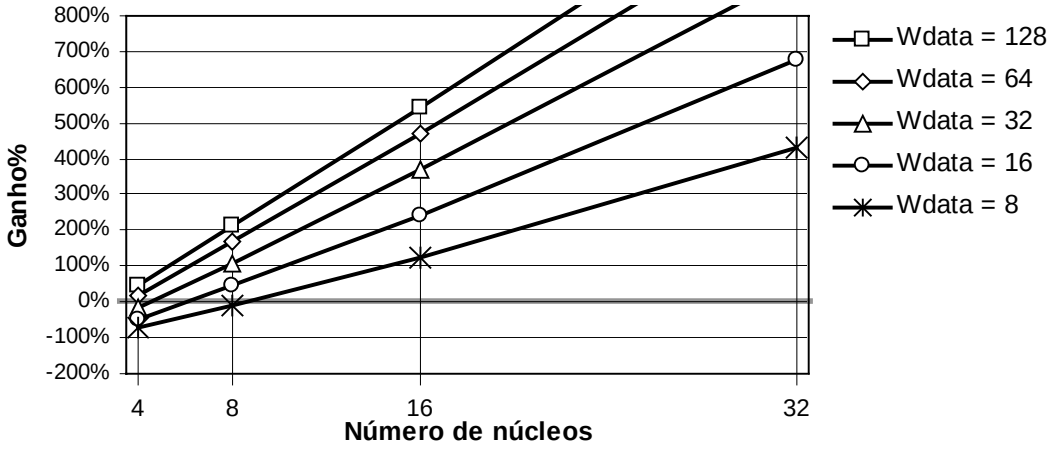


FIGURA 7.25 – Ganhos percentuais da latência da rede SPIN para diferentes larguras de canal de dados na rede (8 a 128 bits) e uma largura fixa no barramento (32 bits).

Impacto da freqüência de operação na latência do barramento PI-Bus

Em todos os exemplos ilustrados acima, foi utilizado o mesmo período de relógio para o barramento PI-Bus e para a rede SPIN ($T_{bus} = T_{noc}$). Contudo, conforme foi discutido na seção 7.1, a carga capacitiva associada ao barramento cresce com o tamanho do sistema em uma razão maior que a de uma rede-em-chip. Para ilustrar o efeito da freqüência de operação do barramento na sua latência de comunicação, o gráfico da Figura 7.26 apresenta curvas de ganho percentual de latência para relações T_{bus}/T_{noc} iguais a 1 e superiores. Nos casos ilustrados a seguir, foram consideradas mensagens com apenas uma palavra de dados ($m = 1$) e uma rede SPIN com canal de 32 bits. As curvas mostram que o aumento da capacitância e, conseqüentemente, do período de operação, torna ainda maior a latência do barramento. Por exemplo, se $T_{bus} = 2{,}0 \propto T_{noc}$, a latência do barramento é maior que a da rede-em-chip para qualquer tamanho de sistema considerado.

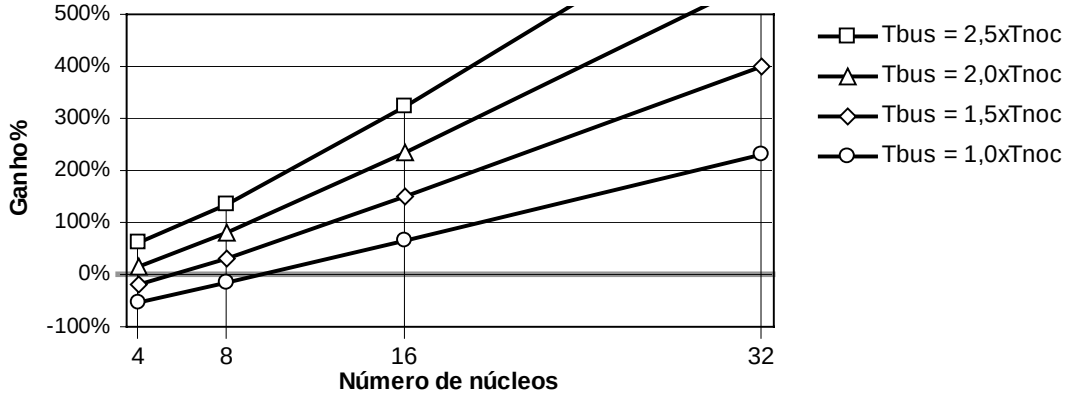


FIGURA 7.26 – Ganhos percentuais da latência da rede SPIN para diferentes períodos de operação do barramento PI-Bus.

7.3 Considerações

Neste capítulo, foi apresentado o desenvolvimento de um conjunto de modelos analíticos para a estimativa em alto nível do desempenho de arquiteturas de comunicação para sistemas integrados. Os modelos foram estabelecidos em um contexto limitado, no qual algumas variáveis não foram consideradas. Por exemplo, foi avaliado apenas o impacto da carga capacitiva como parâmetro para comparação de desempenhos dos fios dos canais de comunicação. Contudo, é sabido que a resistência do fio exerce um papel importante na determinação do tempo de propagação, sobretudo em tecnologias submicrônicas em que os fios são cada vez mais estreitos e as resistências tornam-se mais significativas. Com relação à latência, assumiu-se ser possível obter uma estimativa de desempenho de arquiteturas de comunicação através de uma análise analítica e estática. É sabido que existem vários fatores dinâmicos e não linearidades que impactam no desempenho de uma arquitetura de comunicação, como a profundidade dos FIFOs e o intervalo entre a geração de mensagens, que levam à contenção e à saturação da rede. Tais fatores são difíceis de serem capturados por modelos analíticos simples. De fato, o impacto desses fatores é tipicamente avaliado por simulação, como foi visto no Capítulo 4. Apesar dessas limitações, acredita-se que os modelos aqui desenvolvidos podem tornar-se uma ferramenta bastante útil para uma primeira análise a fim de orientar as escolhas de projeto, contudo, eles não dispensam o uso de metodologias baseadas em simulação.

Quanto à aplicação dos modelos desenvolvidos neste capítulo, eles permitiram avaliar o impacto do tamanho do sistema na carga capacitiva dos canais de comunicação para uma dada tecnologia de fabricação. Os resultados permitiram a observação de algumas questões associadas ao tamanho do sistema no desempenho dos canais. Por exemplo, foi visto que a capacitância do barramento tem um crescimento significativamente maior que a das redes-em-chip. Além disso, foi observado que as topologias de rede-em-chip que oferecem alguma regularidade no comprimento dos canais e uma certa independência com relação ao tamanho dos sistemas apresentam a vantagem de manter uma carga capacitiva constante em seus canais, facilitando o projeto dos seus *buffers*.

Com relação à latência, destaca-se que, para uma configuração típica de 32 bits e para o modelo de carga de comunicação considerada, o barramento oferece desempenho satisfatório em sistemas com até oito núcleos (ou um pouco mais) e para as cargas mais leves. Para os sistemas maiores, o paralelismo da rede permite a obtenção de uma latência de comunicação menor. Além disso, observou-se que é possível estimar uma configuração de menor custo de rede-em-chip (W_{data} menor) que consiga suprir um desempenho superior ao do barramento. Por fim, destacou-se que, se for considerado o efeito da carga capacitiva no desempenho dos fios do barramento, a latência total do barramento para a execução da aplicação será ainda mais degradada.

8 Considerações Finais

Este texto apresentou um conjunto de estudos e atividades de pesquisa que nortearam a questão da comunicação em sistemas integrados em um único chip. Fez-se uma revisão bibliográfica sobre os conceitos que fundamentam as redes de interconexão para computadores paralelos, arquiteturas base para as estruturas de comunicação que se apresentam como a melhor alternativa de interconexão intrachip para os futuros sistemas integrados: as chamadas redes-em-chip. Também foi realizado um estudo a respeito do contexto do trabalho, com conceitos e terminologias da área de projeto de sistemas integrados. Nesse estudo, focalizou-se a questão da comunicação, procurando-se identificar os principais tipos de arquiteturas de comunicação atuais, bem como as arquiteturas emergentes e que se propõem a atender aos requisitos dos futuros sistemas integrados. Após, foram identificadas as limitações das abordagens baseadas no barramento e nos canais ponto-a-ponto dedicados. Disso buscou-se oferecer uma visão sobre as arquiteturas atuais de redes-em-chip, identificando-se as características de algumas das principais arquiteturas correntes. Esse estudo não foi totalmente exaustivo, mas acredita-se ter identificado os trabalhos que apresentaram as contribuições mais significativas à área. Continuando, o texto descreveu atividades de pesquisa que envolveram o desenvolvimento de modelos de redes-em-chip e sistemas integrados para avaliação de desempenho por simulação; a especificação arquitetural, modelagem e síntese em alto nível de uma rede-em-chip; e o desenvolvimento de modelos de alto nível para a estimativa de custo e desempenho de arquiteturas de comunicação para sistemas integrados.

Ao término de cada capítulo, procurou-se apresentar um conjunto de considerações locais que tornasse a avaliação dos resultados obtidos mais próxima do texto no qual foram descritos. Dessa forma, não cabe a esta sessão fazer uma análise individual de cada capítulo, mas sim um apanhado geral sobre seus resultados, suas inter-relações e a contribuição dada ao contexto em que se insere este trabalho. Também, acreditando-se que este espaço seja o local para se fazer uma avaliação a respeito do período passado durante o desenvolvimento da tese, buscar-se-á fazer um relato sobre o histórico do acadêmico e de suas atividades de pesquisa no Curso de Doutorado do Programa de Pós-Graduação em Computação da Universidade Federal do Rio Grande do Sul (PPGC-UFRGS). Por fim, na certeza de que esta tese não conclui um projeto de vida profissional, mas sim o encaminha, serão discutidas idéias de atividades de pesquisa que se pretende desenvolver no futuro e serão listadas algumas oportunidades de pesquisa no contexto deste trabalho.

8.1 Visão Geral sobre o Trabalho e suas Contribuições

Os modelos de simulação, síntese e de estimativa de área e desempenho desenvolvidos neste trabalho permitiram a obtenção de uma série de resultados conclusivos a respeito da aplicabilidade das redes-em-chip, os quais foram apresentados no decorrer do texto. Torna-se relevante aqui emitir algumas considerações adicionais que sintetizem esses resultados.

Primeiramente, deve-se destacar que ainda existe uma série de aplicações que podem ser atendidas por uma arquitetura de comunicação baseada em um barramento central ou em uma hierarquia de barramentos, mas essa aplicabilidade está cada vez mais restrita. Com o aumento crescente da demanda por funcionalidades ainda mais sofisticadas em produtos de eletrônica de consumo, telecomunicações e de computação embarcadas, faz-se necessário o desenvolvimento de novos componentes de processamento e de comunicação. Em um mercado competitivo e com altas pressões de projeto, é necessário que o tempo de projeto de cada produto seja mínimo. Uma abordagem utilizada para suportar essas pressões consiste em acrescentar novas funcionalidades a produtos já existentes. Considerando que cada nova funcionalidade represente um núcleo adicional a uma arquitetura já existente de sistema integrado, o número de núcleos no sistema irá crescer a cada nova geração de produtos. Uma arquitetura de comunicação baseada no barramento que hoje atende aos requisitos da aplicação poderá não fazê-lo em um futuro próximo, pois sua escalabilidade é pobre e o consumo de energia e o desempenho da comunicação irão se degradar com o crescimento do sistema.

Os resultados apresentados nos Capítulos 4, 6 e 7 ajudam a determinar o escopo de aplicação de uma arquitetura em barramento e o momento a partir do qual uma arquitetura do tipo rede-em-chip se faz necessária. Se não for levada em consideração uma perspectiva futura de crescimento do sistema, o barramento central encontra utilidade em aplicações que apresentem uma baixa carga de comunicação e/ou em sistemas com menos de uma dezena de núcleos. Se for considerado que o sistema poderá crescer, mesmo que sua demanda de largura de banda não aumente, mas ele apresente fortes requisitos quanto ao consumo de energia, então o barramento possui espaço limitado e deve-se partir para uma arquitetura de rede-em-chip. Nesse contexto, a rede parametrizável aqui proposta se insere como uma solução intermediária, pois sua largura de banda é escalável em várias dimensões: tamanho do sistema, largura dos canais e profundidade dos *buffers*. Uma rede SoCIN de baixo custo pode ser construída utilizando-se canais estreitos (eg. 8 bits) e *buffers* de pouca profundidade (eg. um ou dois flits). Sua largura de banda será limitada, porém ela pode ser escalada com facilidade. Em um terceiro contexto, considerando-se aplicações com altos requisitos de largura de banda, como televisão digital de alta definição e processadores de rede (ou *network processors*), não há mais espaço para o barramento. Atualmente, dependendo do nível do paralelismo utilizado para a implementação dessas aplicações (eg. tarefa, instrução ou bit) diferentes tipos de arquiteturas de comunicação podem ser usadas, mas todas elas excluem o barramento central. Para os processadores de rede, são utilizadas soluções como canais ponto-a-ponto dedicados e hierarquias de barramento [SHA 2002]. Contudo, as redes-em-chip já começam a ser utilizadas em soluções que adotam o paralelismo em nível de tarefa [KAR 2002].

Algumas contribuições podem ser identificadas neste trabalho:

÷ O estudo comparativo entre o barramento PI-Bus e a rede SPIN, feito através de simulação com precisão de ciclo, auxiliou a demonstrar a efetividade da rede SPIN. Os resultados também permitiram identificar que de nada adianta uma arquitetura de comunicação que ofereça paralelismo se a aplicação apresentar pontos de estrangulamento que limitam a utilização do desempenho disponível.

- A rede SoCIN não apresenta nenhuma inovação arquitetural. Assim como as demais redes-em-chip, ela baseia-se em conceitos já estabelecidos de redes de interconexão para computadores paralelos. Contudo, o modelo VHDL parametrizável sintetizável do roteador RASoC oferece um diferencial à rede SoCIN, pois as arquiteturas encontradas na literatura são rígidas com relação aos parâmetros configuráveis na rede SoCIN. A flexibilidade do modelo do roteador permite a síntese de redes mais adequadas a cada aplicação, seja pela redução do custo ou pelo aumento da largura de banda da rede. Contudo, destaca-se que não foi realizada uma exploração exaustiva do espaço das redes-em-chip. Longe disso, foram escolhidas alternativas arquiteturais que oferecessem um equilíbrio entre custo e desempenho.

- Os modelos de estimativa de alto nível constituem-se em uma ferramenta útil para uma estimativa inicial sobre o custo e o desempenho de uma rede-em-chip. Não se conhece na literatura trabalhos semelhantes visando arquiteturas de redes-em-chip. Os modelos apresentados em [BRI 2002] servem ao tipo de rede proposta pelos autores, a qual difere em um número de características das redes-em-chip baseadas em redes de interconexão. Os modelos aqui apresentados serão inseridos em uma metodologia de projeto a ser desenvolvida visando à síntese automática de redes-em-chip para uma dada aplicação.

- Os estudos e as pesquisas realizados no decorrer desta tese e a difusão dos conhecimentos adquiridos fomentaram o estabelecimento de uma linha de pesquisa sobre redes-em-chip no Grupo de Microeletrônica (GME) do PPGC-UFRGS. Desde a consolidação do tema desta tese, em 2000, diversos alunos de pós-graduação se associaram à investigação de diferentes questões relacionadas ao projeto e à aplicação de redes-em-chip em sistemas integrados.

A partir das contribuições indicadas acima, identifica-se, como contribuição geral deste trabalho, o desenvolvimento e a disponibilização de modelos de simulação, síntese e estimativa de área e desempenho de redes-em-chip, assim com um conjunto de dados e análises sobre a aplicabilidade dessas redes.

8.2 Histórico da Tese

As atividades relacionadas a esta tese tiveram seu início em março de 1998 com o ingresso deste acadêmico no PPGC-UFRGS. O projeto original proposto ao programa previa o estudo e desenvolvimento de redes de aplicação específica sem determinar o nível da implementação. Durante esse ano, foram cursadas disciplinas da área de microeletrônica e foram realizados estudos sobre o tema proposto. Desse período, foi publicado artigo no Iberchip'1999 [ZEF 99] e outro no SIM'1999 [ZEF 99a]. O primeiro apresentava o projeto de uma rede de interconexão experimental baseada em um roteador *wormhole* sintetizado para FPGA, enquanto que o segundo consistia em um estudo sistemático sobre conceitos de redes de interconexão com estudos de caso.

Já no ano de 1999, avaliando-se a hipótese de se desenvolver a tese no contexto de redes para aplicações específicas integradas em um único chip, foram realizados estudos sobre conceitos na área de processamento de imagem e vídeo. Foi realizado um Trabalho Individual no qual se estudou acerca da aplicação da Transformada Discreta do Cosseno na compressão de vídeo digital [ZEF 99b]. Desse trabalho e de outros estudos, identificou-se, naquele momento, que havia pouco espaço para as soluções que se buscava explorar, pois acreditava-se que, para as aplicações vislumbradas, a melhor solução seria baseada no uso de canais ponto-a-ponto dedicados. Disso partiu-se para o estudo da interconexão de computadores em agregados, contexto no qual o acadêmico já havia trabalhado em sua dissertação de mestrado no Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Santa Catarina (PPGCC-UFSC) [ZEF 96]. Ainda em 1999, realizou-se o Exame de Qualificação cuja área de abrangência tratou de temas sobre processamento paralelo e a de parte de profundidade focalizou as redes de interconexão [ZEF 99c].

Nesse novo contexto, no ano de 2000, buscou-se uma integração com pesquisadores do Grupo de Processamento Paralelo e Distribuído (GPPD) e do Grupo de Matemática da Computação e Processamento de Alto Desempenho (GMCPAD) do PPGC-UFRGS a fim de identificar aplicações que pudessem requer uma rede de interconexão com projeto específico. Este acadêmico associou-se a um projeto de pesquisa que almejava o desenvolvimento de soluções em processamento paralelo para aplicações de simulação numérica, no qual foram publicados quatro artigos com os resultados intermediários do projeto [RIZ 2000, DOR 2000, ZEF 2000, DOR 2000a]. Nesse contexto, e vislumbrando-se o desenvolvimento de redes específicas para agregados de computadores, manteve-se contato com o professor Alain Greiner para a realização de estágio sanduíche no departamento Architecture des Systèmes Intégrés et Microélectronique do Laboratoire d´Informatique Paris 6 (ASIM/LIP6), o qual apresentava competência no desenvolvimento de um agregado de PCs chamado MPC. Nas tratativas com o professor Alain discutiu-se diferentes alternativas de projetos que acabaram por conduzir ao Projeto SPIN. Dessa forma, o contexto da tese foi novamente focado para a integração de redes em um único chip e começou-se a direcionar todos os esforços nessa área trabalhando-se em cooperação com outros pesquisadores do PPGC-UFRGS. Desses trabalhos, foram publicados artigos no IBERCHIP'2001 [KRE 2001], no SIM'2001 [KRE 2001a] e no SBCCI'2001 [KRE 2001b], os quais tratavam da análise e seleção de redes-em-chip-para sistemas integrados. Também foi publicado artigo na Revista de Informática Teórica e Aplicada [KRE 2001c].

Em 2001, já realizando estágio no LIP6, desenvolveu-se o modelo CASS do roteador RSPIN e foram efetuados os experimentos de avaliação de desempenho da rede SPIN e do barramento PI-Bus, descritos nesta tese. Desses trabalhos, foram publicados artigos no DATE'2003 [AND 2003] e no SIM'2002 [ZEF 2002c]. No final do segundo semestre de 2001, após o retorno do estágio, foi desenvolvida a primeira versão do modelo VHDL do roteador RASoC.

Em 2002, os esforços foram dirigidos à redação e à defesa da proposta de tese e aos estudos que resultaram nos modelos analíticos apresentados nos Capítulos 6 e 7. A evolução dos modelos de estimativa de desempenho foi apresentada em artigos publicados no SBCCI'2002 [ZEF 2002], IP-Based Design'2002 [ZEF 2002a] e SIM'2002 [ZEF 2002b]. Em paralelo, foram desenvolvidos projetos com pesquisadores do GME na área de teste de sistemas integrados baseados em redes-em-chip, e com

pesquisadores da UNIVALI, na área de modelagem e de componentes para roteadores de redes-em-chip. O primeiro trabalho resultou em um artigo publicado no VTS'2003 [COT 2003] e o segundo em um artigo publicado na Revisa Hífen [ESP 2002].

No ano de 2003, foi publicado um artigo no SBCCI'2003, no qual foi apresentada a arquitetura da rede SoCIN e o detalhamento da organização do roteador RASoC [ZEF 2003].

8.3 Projetos Futuros e Oportunidades de Pesquisa

Durante o desenvolvimento do modelo do roteador RASoC, ressentiu-se de um suporte mais elaborado para a parametrização do modelo VHDL e isso restringiu o número de variáveis possíveis de serem configuradas. Dando continuidade a esta tese, será desenvolvido o projeto ParIS (***Par****ameterized* ***I****nterconnect* ***S****witch*), o qual atacará esse problema sob uma outra óptica. Serão desenvolvidos modelos de componentes para a construção de modelos de roteador. Para cada componente, serão disponibilizados modelos parametrizáveis que implementarão alternativas arquiteturais com diferentes características de custo e desempenho. Cada modelo será caracterizado quanto à área ocupada e ao seu caminho crítico para as tecnologias alvo. Eles também serão caracterizados quando às suas respostas a cargas de comunicação aplicadas por simulação. Procurar-se-á também oferecer caracterizações de alto nível baseadas em modelos analíticos de estimativa de área e desempenho. Em uma etapa posterior, serão desenvolvidas versões DFT (*Designed for Test*) desses modelos de modo a oferecer testabilidade aos mesmos.

A partir dos modelos caracterizados, será construída uma biblioteca de componentes que dará suporte a uma ferramenta computacional para a construção de modelos de roteador e de redes-em-chip. Essa ferramenta terá uma interface visual e gerará descrições em VHDL e/ou C para síntese e avaliação da rede por simulação. A ferramenta também oferecerá estimativas preliminares baseadas em modelos de alto nível. Os arquivos de saída servirão de entrada para ferramentas de síntese e simulação, como, por exemplo: o Quartus II da Altera, o CASS e/ou o SystemC.

Posteriormente, a ferramenta será integrada a uma metodologia que deverá automatizar o processo de seleção de componentes, construção do roteador, avaliação da rede de modo a atender aos requisitos de custo (área e potência) e desempenho de uma dada aplicação.

Além das oportunidades de pesquisa que serão cobertas pelo Projeto ParIS, identifica-se uma série de outras alternativas de projeto no contexto das redes-em-chip, as quais são sucintamente enumeradas abaixo:

- Avaliação da viabilidade de implementação de roteador com interface e canais internos compatíveis com alguma interface padrão: VCI ou OCP;

- Desenvolvimento de *benchmarks* a serem aplicados na avaliação de redes-em-chip com base nas características de aplicações típicas de sistemas integrados;

- ÷ Avaliação de requisitos e características de aplicações com alta demanda de largura de banda (eg. processadores de rede) para a especificação e projeto de redes-em-chip;

- ÷ Modelo híbrido roteador/ponte/árbitro para a construção de redes hierárquicas com barramentos locais de pequena capacidade na base e uma rede chaveada no topo da hierarquia, construídas a partir de instâncias um único componente que realiza a arbitragem do barramento local e o roteamento de pacotes na rede;

- ÷ Roteadores com QoS (*Quality of Service*) que possam comprometer-se com níveis qualidade de serviço para comunicações críticas do sistema integrado.

Finalmente, conclui-se esta tese destacando-se a qualidade da formação oferecida pelo PPGC-UFRGS e a reconhecida capacidade de seus professores e pesquisadores que muito colaboraram para o desenvolvimento deste trabalho.

Apêndice A Organização do Roteador RASoC

Este apêndice apresenta a descrição da organização do roteador RASoC. Conforme descrito na seção 5.10.1, o roteador RASoC é constituído, internamente, por uma estrutura distribuída baseada em dois tipos de módulo: *Input Channel* e *Output Channel*. Cada porta de comunicação é implementada por uma par de módulos de entrada e de saída e, dessa forma, um roteador completo, com cinco portas de comunicação, terá cinco instâncias do módulo *Input Channel* e outras cinco do módulo *Output Channel*. Em um roteador com um número menor de portas, os módulos associados às portas não implementadas não são sintetizados pelo compilador. Isso é feito de forma automática, desde que os sinais de entrada dessas portas sejam aterrados (conectados em 0) e os sinais de saída sejam mantidos em aberto.

A seguir, são apresentados detalhes a respeito da organização desses módulos e de seus blocos constituintes. Primeiramente, descreve-se a organização do módulo *Input Channel* e os sinais de sua interface. Após, apresenta-se a estrutura dos blocos que compõem esse módulo. Continuando, descreve-se o módulo *Output Channel* (organização e interface) e os blocos que o constituem.

A.1 O Módulo *Input Channel*

Na Figura A.1, é apresentado o símbolo do módulo *Input Channel* (que representa a sua interface) e os blocos que compõem sua estrutura interna: *Input Flow Controller* (IFC), *Input Buffer* (IB), *Input Controller* (IC) e *Input Rd Switch* (IRS). A funcionalidade desses blocos já foi descrita na subseção 5.10.2.

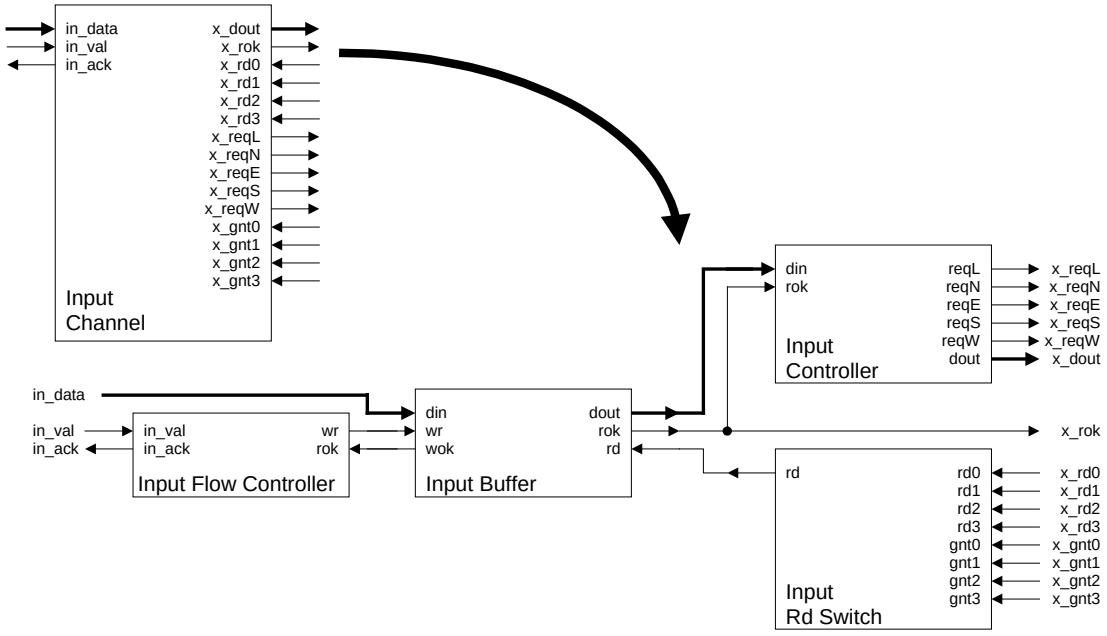


FIGURA A.1 – Interface e estrutura do módulo *Input Channel*.

Conforme foi apresentado na subseção 5.10.2, os nomes dos sinais conectados à interface externa do roteador utilizam o prefixo *in_*, enquanto que os sinais conectados aos módulos de saída do roteador utilizam o prefixo *x_*. Na Tabela A.1, é apresentada a descrição (largura e definição) de cada um dos sinais que compõem a interface do módulo *Input Channel*.

TABELA A.1 – Definição dos sinais da interface do módulo *Input Channel*.

Nome	**Largura**	**Definição**
in_data	(*n*+2) bits	Canal de dados do canal de entrada (inclui os bits *eop* e *bop*)
in_val	1 bit	Validação de flit no canal de dados do canal de entrada
in_ack	1 bit	Reconhecimento da recepção do flit pelo canal de entrada
x_dout	1 bit	Canal de dados com cabeçalho atualizado após o roteamento
x_rok	1 bit	Informa a disponibilidade de um flit a ser lido do bloco IB
*x_rd*0	1 bit	Comando de leitura enviado pelo canal de saída 0
*x_rd*1	1 bit	Comando de leitura enviado pelo canal de saída 1
*x_rd*2	1 bit	Comando de leitura enviado pelo canal de saída 2
*x_rd*3	1 bit	Comando de leitura enviado pelo canal de saída 3
x_reqL	1 bit	Requisição de uso do canal de saída da porta *Local* (local)
x_reqN	1 bit	Requisição de uso do canal de saída da porta *North* (norte)
x_reqE	1 bit	Requisição de uso do canal de saída da porta *East* (leste)
x_reqS	1 bit	Requisição de uso do canal de saída da porta *South* (sul)
x_reqW	1 bit	Requisição de uso do canal de saída da porta *West* (oeste)
*x_gnt*0	1 bit	Confirmação de seleção do canal de entrada pelo canal de saída 0
*x_gnt*1	1 bit	Confirmação de seleção do canal de entrada pelo canal de saída 1
*x_gnt*2	1 bit	Confirmação de seleção do canal de entrada pelo canal de saída 2
*x_gnt*3	1 bit	Confirmação de seleção do canal de entrada pelo canal de saída 3

Embora cada canal de entrada possa requisitar apenas quatro dos cinco canais de saída, o módulo *Input Channel* é um módulo genérico e sua interface inclui sinais de requisição para todos os canais de saída do roteador. Em cada instância do módulo *Input Channel*, o sinal de requisição não utilizado é ignorado. Também, devido à generalidade do modelo, os sinais *rd* e *gnt* possuem índices genéricos (de 0 a 3) e o significado dos mesmos varia conforme o canal de entrada implementado. Por exemplo, para o canal *Lin*, *x_rd*0 representa o sinal de comando de leitura originário da primeira porta da lista de portas que ele pode requisitar. Como essa lista é definida na ordem *N*, *E*, *S* e *W*, a porta correspondente ao sinal *x_rd*0 é a porta *N*. Da mesma forma, *x_gnt*3 representa o sinal de confirmação de seleção oriundo da quarta porta que a porta *L* pode requisitar (ou seja, a porta *W*).

Na Tabela A.2, são listados os significados dos sinais com índice genérico para cada instância do módulo *Input Channel*. Como pode ser observado, na leitura da linha correspondente à instância *Lin*, *x_rd*0 representa o sinal de comando de leitura oriundo do canal de saída da porta *N* (*Nrd*), enquanto *x_gnt*3 representa o sinal de confirmação gerado pelo canal de saída da porta *W* (*WgntL*). Deve-se destacar que existem tantos sinais de confirmação (*gnt*) quanto o número de conexões possíveis no crossbar do roteador (ou seja, 20), sendo que, na nomenclatura utilizada, *WgntL* indica uma confirmação do canal de saída *Nout* para o canal de entrada *Lin*. Por outro lado, existem

apenas cinco sinais de comando de leitura (*rd*) gerados pelos cinco canais de saída do roteador. Na terminologia do modelo, *Nrd* indica o sinal de comando de leitura gerado pelo canal de saída *Nout*, o qual deve ser conectado ao sinal *rd* do IB do canal de entrada que estiver utilizando esse canal de saída.

TABELA A.2 – Significado de x_rd_i e x_gnt_i para cada instância de *Input Channel*.

Instância	**Entradas**							
	***x_rd*0**	***x_rd*1**	***x_rd*2**	***x_rd*3**	***x_gnt*0**	***x_gnt*1**	***x_gnt*2**	***x_gnt*3**
Lin	*Nrd*	*Erd*	*Srd*	*Wrd*	*NgntL*	*EgntL*	*SgntL*	*WgntL*
Nin	*Lrd*	*Erd*	*Srd*	*Wrd*	*LgntN*	*EgntN*	*SgntN*	*WgntN*
Ein	*Lrd*	*Nrd*	*Srd*	*Wrd*	*LgntE*	*NgntE*	*SgntE*	*WgntE*
Sin	*Lrd*	*Nrd*	*Erd*	*Wrd*	*LgntS*	*NgntS*	*EgntS*	*WgntS*
Win	*Lrd*	*Nrd*	*Erd*	*Srd*	*LgntW*	*NgntW*	*EgntW*	*SgntW*

Na Tabela A.3, são indicados os sinais de requisição efetivamente utilizados em cada instância do módulo *Input Channel*. Da mesma forma que para os sinais de confirmação, existem tantos sinais de requisição quanto o número de conexões possíveis no crossbar do roteador (ou seja, 20). Na terminologia do modelo, *LreqN* indica o sinal de requisição do canal de entrada *Lin* para o canal de saída *Nout*. Observa-se que um canal de entrada de uma dada porta não pode requisitar o canal de saída da mesma porta.

TABELA A.3 – Uso das linhas de requisição nas instâncias de *Input Channel*.

Instância	**Saídas**				
	x_reqL	***x_reqN***	***x_reqE***	***x_reqS***	***x_reqW***
Lin	*–∝–*	*LreqN*	*LreqE*	*LreqS*	*LreqW*
Nin	*NreqL*	*–∝–*	*NreqE*	*NreqS*	*NreqW*
Ein	*EreqL*	*EreqN*	*–∝–*	*EreqS*	*EreqW*
Sin	*SreqL*	*SreqN*	*SreqE*	*–∝–*	*WreqW*
Win	*WreqL*	*WreqN*	*WreqE*	*WreqS*	*–∝–*

A.1.1 O bloco *Input Flow Controller*

O bloco *Input Flow Controller* (IFC) implementa o mecanismo de controle de fluxo de entrada de um canal de entrada do roteador, adaptando o protocolo de *handshake* ao protocolo FIFO. Nos dois protocolos, o controle de fluxo é realizado por meio do uso de dois sinais de controle: um de requisição (ou validação), no sentido emissor-receptor, e outro de retorno (ou reconhecimento), no sentido receptor-emissor. Como a diferença entre os dois protocolos resume-se à temporização do sinal de retorno, a lógica necessária a essa adaptação é bastante simples. No protocolo FIFO, o sinal de retorno consiste de um bit de condição que informa, antecipadamente, a habilidade de receber um novo dado, sendo ativado independentemente de haver ou não uma requisição. No protocolo de *handshake*, o sinal de retorno só é ativado após o recebimento de uma requisição e somente se o receptor for capaz de receber o dado a ser enviado. Para o emissor, a ativação desse sinal é interpretada como uma confirmação de entrega do dado sendo transmitido. Com base nisso, a lógica de adaptação dos protocolos consiste de uma operação lógica “E” que condiciona a

ativação do sinal de retorno do protocolo de *handshake* (*ack*) à ativação do sinal de requisição desse protocolo (*val*) e do sinal de retorno do protocolo FIFO (*wok*), conforme é ilustrado na Figura A.2.

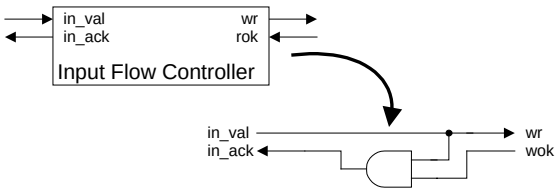


FIGURA A.2 – Estrutura do bloco IFC.

A implementação do circuito de controle de fluxo de entrada sob a forma de uma unidade independente das demais visa a permitir a substituição facilitada da técnica originalmente utilizada (*handhake*) por outra técnica baseada em dois fios (eg. controle de fluxo baseado em créditos) pela simples troca da arquitetura interna do bloco IFC. Externamente, a única diferença seria o significado do sinal *in_ack* que, no caso do controle de fluxo baseado em créditos, passaria a indicar um retorno de crétido (*in_cr*).

A.1.2 O bloco *Input Buffer*

O bloco *Input Buffer* (IB) implementa o mecanismo de memorização dos flits que chegam no canal de entrada do roteador. Ele é constituído por um *buffer* FIFO com capacidade de armazenar *p* flits de *n*+2 bits.

Internamente, o bloco IB é organizado conforme o modelo PC-PO (parte controle – parte operativa). Na Figura A.3, é apresentada a estrutura interna da PO do *buffer* e os sinais da interface PC-PO. A PO é constituída por um conjunto de registradores paralelo-paralelo identificados pela posição de 0 a *p*–1, por um multiplexador de seleção de saída e por uma porta lógica que implementa a função de escrita nos registradores.

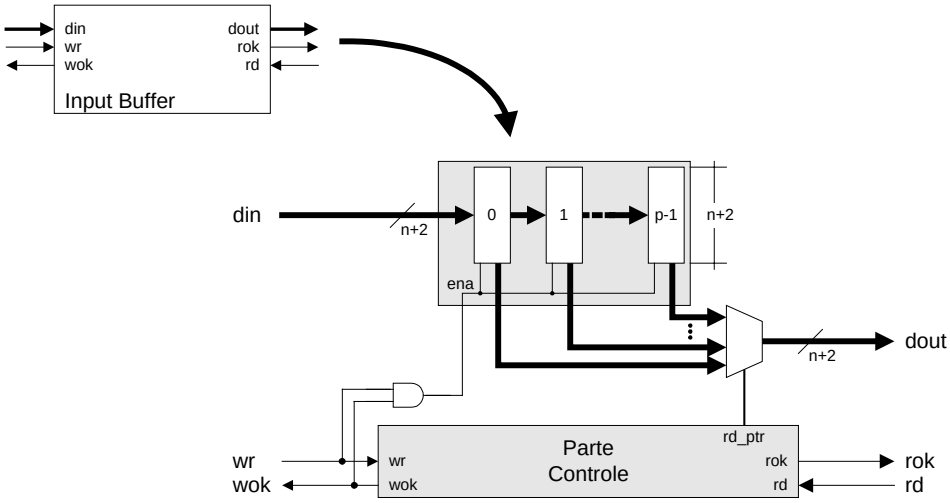


FIGURA A.3 – Estrutura do bloco IB.

A PC do *buffer* FIFO é baseada na máquina de estados de Moore ilustrada na Figura A.4. Ela possui *p*+1 estados, numerados de 0 a *p* (onde *p* é a profundidade do *buffer*), sendo que o estado 0 representa a condição de *buffer* vazio e o estado *p* representa a condição de *buffer* cheio. O estado *i*, por sua vez, representa os *p*-1 estados intermediários, nos quais existem ao menos uma posição ocupada e uma posição livre (onde *i* varia de 1 a *p*-1). Em um *buffer* com quatro posições, por exemplo, a máquina de estados tem três estados intermediários representando as condições nas quais existem uma, duas ou três posições ocupadas (ou seja, três, duas ou uma posição livre, respectivamente).

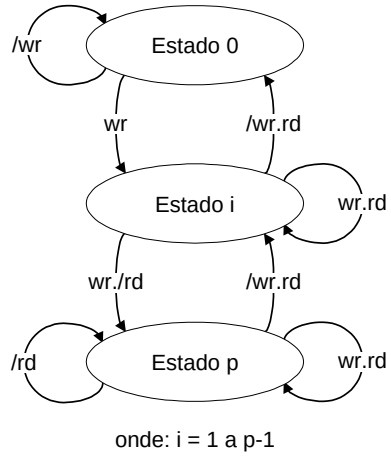


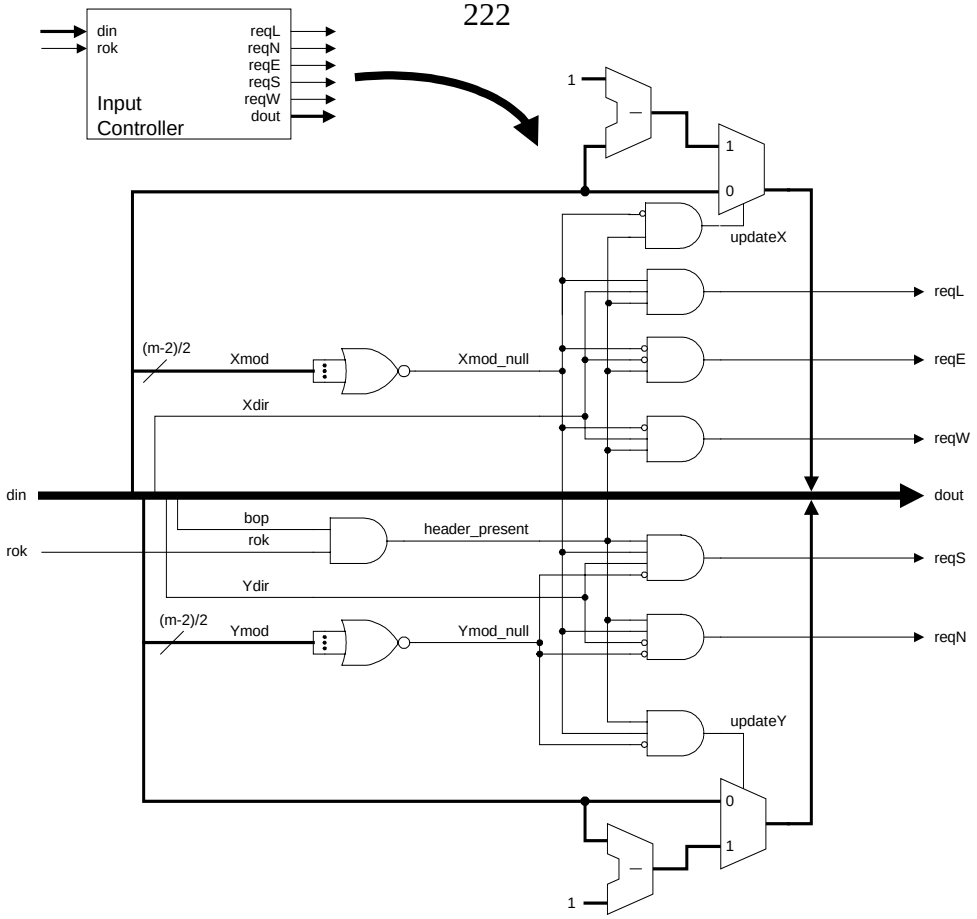
FIGURA A.4 – Máquina de estados do *buffer* FIFO.

Na Tabela A.4, são apresentados os valores das saídas da PC para cada um dos três tipos de estado. A saída *wok* sinaliza a disponibilidade de espaço no *buffer* para a escrita de novos dados, enquanto que a saída *rok* sinaliza a disponibilidade de um dado para ser lido do *buffer*. Já a saída *rd_ptr* determina o registrador a ser conectado na saída através do multiplexador. Destaca-se que, no estado *p*, o sinal *wok* dependerá da ocorrência de um sinal de *rd* durante o estado. Em outras palavras, na ausência de *rd*, o estado de *wok* é igual a 0 e não é permitida a escrita no *buffer*. Contudo se *rd* é igual a 1, então *wok* também será igual a 1, sinalizando a escrita do dado na entrada do *buffer*.

TABELA A.4 – Saídas da máquina de estados do *buffer* FIFO.

Estado	**Saídas**		
	wok	***rok***	***rd_ptr***
0	1	0	0
i	1	1	*i*–1
p	*rd*	1	*p*–1

Quanto à arquitetura utilizada neste *buffer* FIFO, destaca-se que existem soluções mais econômicas se for considerada uma implementação *full-custom* [RAB 96], como, por exemplo, a implementação de *buffers* baseados em células de memória SRAM (*Static Random Access Memory*). Contudo, para a síntese em FPGA, a arquitetura descrita acima se constitui em uma das poucas alternativas de implementação. Além dessa arquitetura, também foi implementada uma versão baseada em fila circular, a qual, porém, não é descrita neste texto.



A.1.3 O bloco *Input Controller*

O bloco *Input Controller* (IC) implementa o algoritmo de roteamento apresentado na seção 5.5. Sua estrutura é representada pelo circuito da Figura A.5. A partir da sinalização de disponibilidade de dados no bloco IB (*rok* = 1), o bloco IC verifica a presença de um cabeçalho (*bop* = 1 e *header_present* = 1) e extrai os campos *Xdir*, *Xmod*, *Ydir* e *Ymod* necessários ao roteamento do pacote. Se *Xmod* for não nulo, o roteamento é realizado na direção X. Caso contrário e se *Ymod* for diferente de zero, o roteamento é realizado na direção Y. Porém, se tanto *Xmod* quando *Ymod* forem nulos, o pacote é encaminhado pela porta local.

Um pacote a ser roteado na direção X deve ser encaminhado pela porta leste (*reqE* = 1), se *Xdir* for nulo, ou pela porta oeste (*reqW* = 1), se *Xdir* for igual a 1. De forma análoga, um pacote a ser roteado na direção Y deve ser encaminhado pela porta norte (*reqN* = 1), se *Ydir* for nulo, ou pela porta sul (*reqS* = 1), se *Ydir* for igual a 1. Além disso, se o pacote for encaminhado nas direções X ou Y, o campo de módulo dessas direções é decrementado para computar o passo de roteamento executado.

O valor de cada campo de direção pode ser entendido como um bit de sinal. Para direções positivas, como leste ou norte (+X ou +Y, respectivamente), o campo de direção é igual a 0. Para direções negativas, como oeste ou sul (. X ou . Y, respectivamente), o campo de direção é igual a 1.

FIGURA A.5 – Estrutura do bloco IC.

A.1.4 O bloco *Input Rd Switch*

O bloco *Input Rd Switch* (IRS), visto na Figura A .6.a, trata-se de um circuito multiplexador 4∝1 de um bit. Suas entradas de dado recebem os sinais de comando de leitura rd_i de quatro canais de saída do roteador (*i* varia de 0 a 3), e suas entradas de seleção recebem sinais de confirmação gnt_i enviados por esses canais de saída. Com base no valor dos sinais de seleção, o bloco IRS conecta o sinal *rd* do canal de saída requisitado pelo bloco IC ao sinal *rd* do bloco IB.

Visando a síntese do modelo em FPGAs da Altera, os quais não dispõem de *buffers tri-state* internos, a estrutura do multiplexador do bloco IRS no modelo VHDL foi descrita com base em um arranjo de primitivas (ou portas) AND-OR. O circuito equivalente a esse código é ilustrado na Figura A .6.b.

Contudo, o modelo VHDL pode se facilmente portado para tecnologias que disponham de *buffers tri-state* internos ou outras estruturas para a construção de multiplexadores (como transistores de passagem, por exemplo). Basicamente, é necessário eliminar a porta OR, trocar as primitivas AND por funções que instanciam *buffers tri-state* e incluir uma outra função para instanciar um *buffer tri-state* que garanta que a saída do multiplexador seja mantida em 0 quando nenhum dos outros *buffers* estiver habilitado, como é ilustrado na Figura A .6.c. Do contrário a saída ficaria flutuante. Alternativamente, o último *buffer tri-state* pode ser substituído por um resistor de *pull-up* e um inversor.

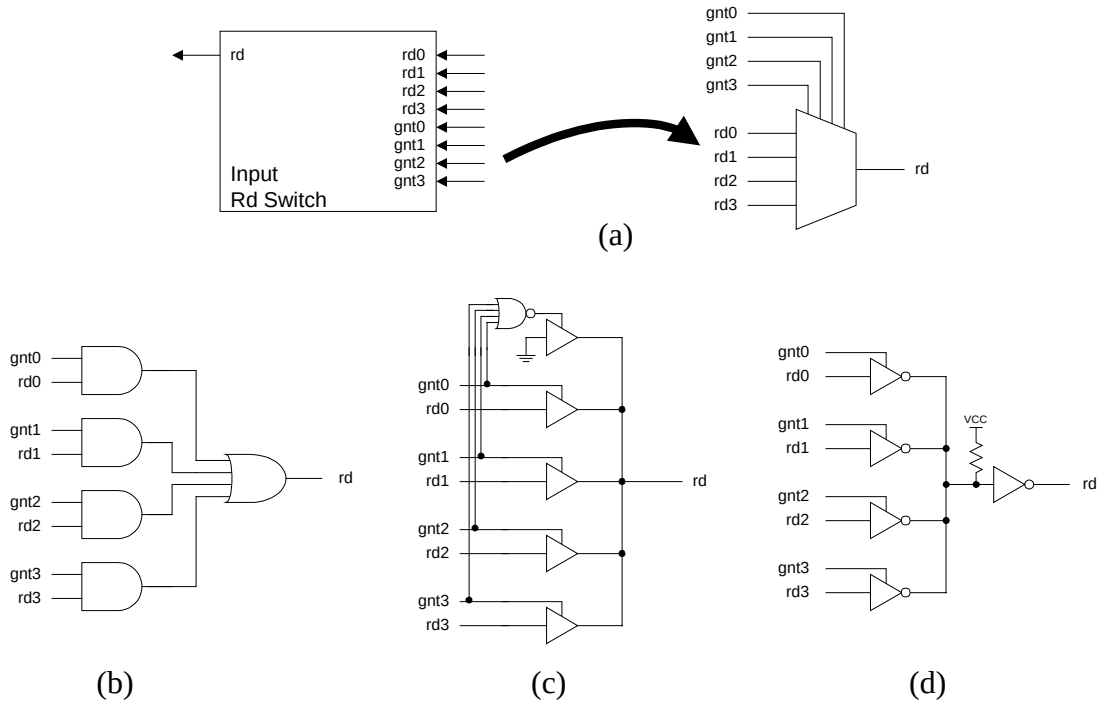


FIGURA A.6 – Estrutura do bloco IRS: (a) multiplexador 4x1; (b) arranjo AND-OR; (c) estrutura baseada em *buffers tri-state*; (d) estrutura baseada em *buffers tri-state* com resistor de *pull-up*.

A.2 O Módulo *Output Channel*

Na Figura A .7, é apresentado o símbolo do módulo *Output Channel* (que representa a sua interface) e os blocos que compõem sua estrutura interna: *Output Controller* (OC), *Output Data Switch* (ODS), *Output Rok Switch* (ORS) e *Output Flow Controller* (OFC). A funcionalidade desses blocos já foi descrita na subseção 5.10.2.

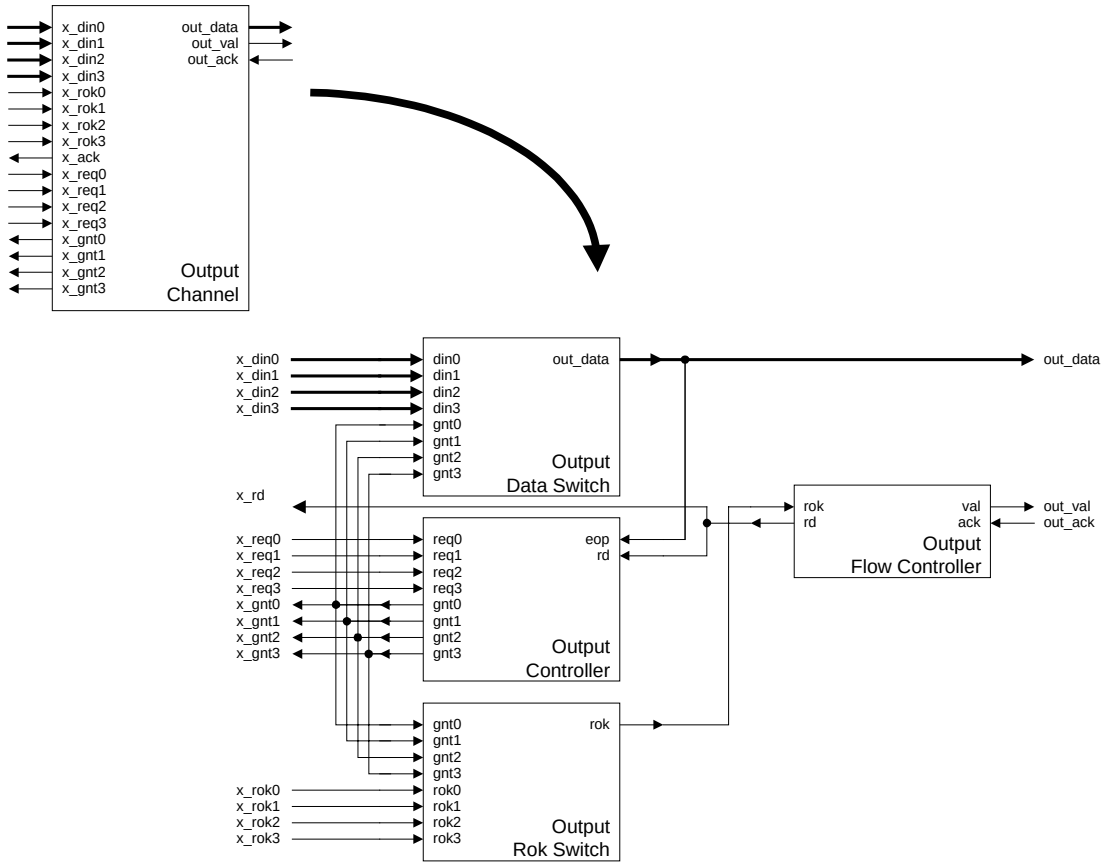


FIGURA A.7 – Interface e estrutura do módulo *Output Channel*.

Conforme foi apresentado na subseção 5.10.2, os nomes dos sinais conectados à interface externa do roteador utilizam o prefixo *out_*, enquanto que os sinais conectados aos módulos de entrada do roteador utilizam o prefixo *x_*. Na Tabela A.5, é apresentada a descrição (largura e definição) de cada um dos sinais que compõem a interface do módulo *Output Channel*.

TABELA A.5 – Definição dos sinais da interface do módulo *Output Channel*.

Nome	**Largura**	**Definição**
out_data	(*n*+2) bits	Canal de dados do canal de saída (inclui os bits *eop* e *bop*)
out _val	1 bit	Validação de flit no canal de dados do canal de saída
out _ack	1 bit	Reconhecimento da transferência do flit pelo canal de saída
x_rd	1 bit	Comando de leitura para o canal de entrada selecionado
*x_din*0	(*n*+2) bits	Canal de dados do canal de entrada 0 com cabeçalho atualizado
*x_din*1	(*n*+2) bits	Canal de dados do canal de entrada 1 com cabeçalho atualizado
*x_din*2	(*n*+2) bits	Canal de dados do canal de entrada 2 com cabeçalho atualizado
*x_din*3	(*n*+2) bits	Canal de dados do canal de entrada 3 com cabeçalho atualizado
*x_rok*0	1 bit	Informa a disponibilidade de um flit a ser lido do IB do canal 0
*x_rok*1	1 bit	Informa a disponibilidade de um flit a ser lido do IB do canal 1
*x_rok*2	1 bit	Informa a disponibilidade de um flit a ser lido do IB do canal 2
*x_rok*3	1 bit	Informa a disponibilidade de um flit a ser lido do IB do canal 3
*x_req*0	1 bit	Requisição de uso do canal de saída pelo canal de entrada 0
*x_req*1	1 bit	Requisição de uso do canal de saída pelo canal de entrada 1
*x_req*2	1 bit	Requisição de uso do canal de saída pelo canal de entrada 2
*x_req*3	1 bit	Requisição de uso do canal de saída pelo canal de entrada 3
*x_gnt*0	1 bit	Confirmação de seleção do canal de saída ao canal requisitante 0
*x_gnt*1	1 bit	Confirmação de seleção do canal de saída ao canal requisitante 1
*x_gnt*2	1 bit	Confirmação de seleção do canal de saída ao canal requisitante 2
*x_gnt*3	1 bit	Confirmação de seleção do canal de saída ao canal requisitante 3

Nas tabelas a seguir, são listados os significados dos sinais com índice genérico para cada instância do módulo *Output Channel*. Como pode ser observado na Tabela A.6, para a instância *Lout*, por exemplo, *x_din*0 e *x_rok*0 correspondem, respectivamente, aos sinais *Ndata* e *Nrok* oriundos do bloco IB do canal de entrada *Nin*.

TABELA A.6 – Significado de x_din_i e x_rok_i para cada instância de *Output Channel*.

Instância	**Entradas**							
	***x_din*0**	***x_din*1**	***x_din*2**	***x_din*3**	***x_rok*0**	***x_rok*1**	***x_rok*2**	***x_rok*3**
Lout	*Ndata*	*Edata*	*Sdata*	*Wdata*	*Nrok*	*Erok*	*Srok*	*Wrok*
Nout	*Ldata*	*Edata*	*Sdata*	*Wdata*	*Lrok*	*Erok*	*Srok*	*Wrok*
Eout	*Ldata*	*Ndata*	*Sdata*	*Wdata*	*Lrok*	*Nrok*	*Srok*	*Wrok*
Sout	*Ldata*	*Ndata*	*Edata*	*Wdata*	*Lrok*	*Nrok*	*Erok*	*Wrok*
Wout	*Ldata*	*Ndata*	*Edata*	*Sdata*	*Lrok*	*Nrok*	*Erok*	*Srok*

De forma análoga, pode-se observar, na Tabela A.7, que, para a instância *Lout*, os sinais *x_req*0 e *x_gnt*0 correspondem, respectivamente, aos sinais *NreqL* e *LgntN*.

TABELA A.7 – Significado de x_req_i e x_gnt_i para cada instância de *Output Channel*.

Instância	**Entradas**				**Saídas**			
	***x_req*0**	***x_req*1**	***x_req*2**	***x_req*3**	***x_gnt*0**	***x_gnt*1**	***x_gnt*2**	***x_gnt*3**
Lout	*NreqL*	*EreqL*	*SreqL*	*WreqL*	*LgntN*	*LgntE*	*LgntS*	*LgntW*
Nout	*LreqN*	*EreqN*	*SreqN*	*WeqN*	*NgntL*	*NgntE*	*NgntS*	*NgntW*
Eout	*LreqE*	*NreqE*	*SReqE*	*WreqE*	*EgntL*	*EgntN*	*EgntS*	*EgntW*
Sout	*LreqS*	*NreqS*	*EreqS*	*WreqS*	*SgntL*	*SgntN*	*SgntE*	*SgntW*
Wout	*LreqW*	*NreqW*	*EreqW*	*SreqW*	*WgntL*	*WgntN*	*WgntE*	*WgntS*

A.2.1 O bloco *Output Data Switch*

O bloco *Output Data Switch* (ODS) conecta a saída de dado do bloco IB do canal de entrada selecionado ao canal de dados do canal de saída. Ele recebe sinais de confirmação de seleção do bloco OC e utiliza esses sinais para selecionar um dos canais de entrada. Na Figura A .8, é ilustrada a estrutura do bloco ODS, a qual é composta por n+2 multiplexadores 4∝1 de 1 bit. A implementação desses multiplexadores varia com a tecnologia alvo disponível e, conforme foi visto na descrição do bloco IRS, pode ser baseada em arranjos AND-OR, *buffers tri-state* ou transistores de passagem.

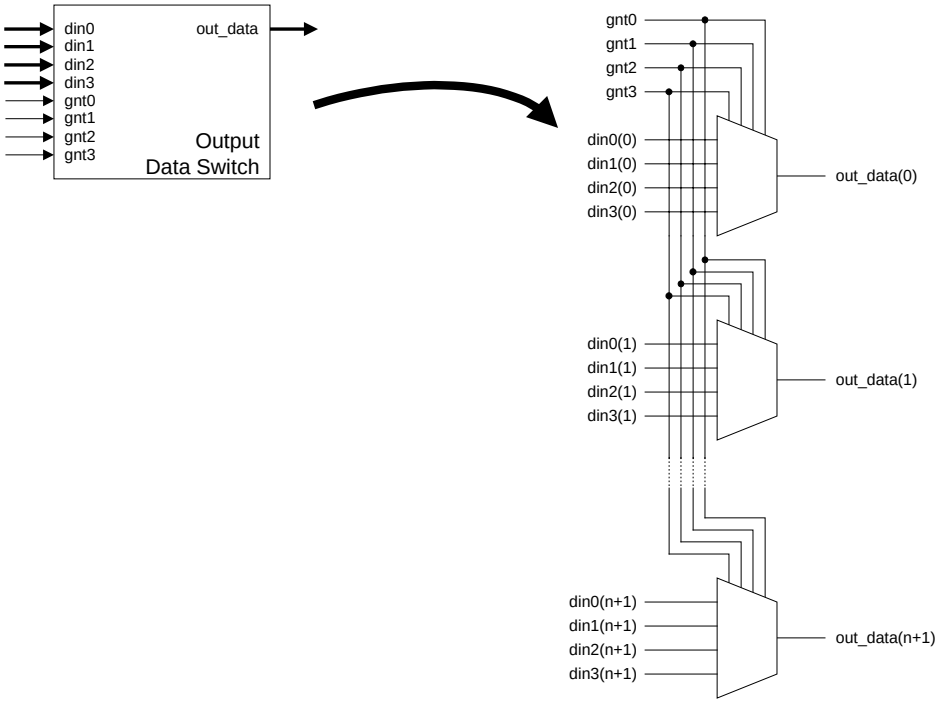


FIGURA A.8 – Estrutura do bloco ODS.

Destaca-se que, no caso de canais de dado, não há restrições quanto à flutuação do sinais de saída *out_data*. Contudo, para sinais de controle, essa flutuação deve ser evitada, já que os mesmos devem ter um valor estável definido o tempo inteiro.

A.2.2 O bloco *Output Rok Switch*

O bloco *Output Rok Switch* (ORS) seleciona o sinal *rok* do bloco IB do canal de entrada conectado ao canal de saída. Com base nos sinais de confirmação de seleção recebidos do bloco OC, ele seleciona o sinal *rok* do canal de entrada contemplado com a autorização de uso do canal de saída. Sua estrutura, mostrada na Figura A .9, é baseada em um multiplexador análogo ao apresentado para o bloco IRS. Da mesma que para os blocos IRS e ODS, a implementação desse multiplexador varia com a tecnologia alvo disponível, podendo ser baseada em arranjos AND-OR, *buffers tri-state* ou transistores de passagem. Deve ser garantido um nível estável em 0 no sinal *rok* quando nenhum sinal *de* confirmação é ativado

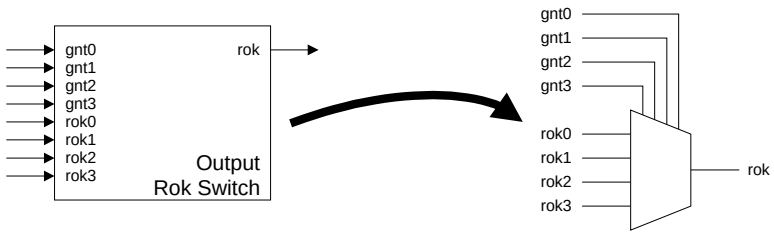


FIGURA A.9 – Estrutura do bloco ORS.

A.2.3 O bloco *Output Flow Controller*

O bloco *Output Flow Controller* (OFC) realiza o controle do fluxo dos flits emitidos através do canal de saída. Ele é responsável por adaptar o protocolo o FIFO ao protocolo de *handshake* do enlace SoCIN. Porém, como do ponto de vista do emissor, as temporizações dos sinais dos sinais de controle de fluxo no protocolo FIFO (*rok* e *rd*) e *handshake* (*val* e *ack*) são idênticas, o bloco OFC implementa apenas um par de fios que realizam a conexão direta entre os sinais *rd* e *val* e os sinais *rok* e *ack*.

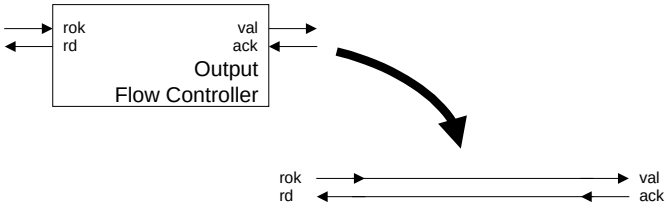


FIGURA A.10 – Estrutura do bloco OFC.

Embora a adaptação dos protocolos FIFO e de *handshake* no bloco OFC não implemente nenhuma lógica, o que poderia sugerir a irrelevância desse bloco, ele foi criado pensando-se na implementação de técnicas alternativas de controle de fluxo que requeiram a implementação de alguma lógica. Por exemplo, em uma abordagem de controle de fluxo baseada em créditos, o bloco OFC implementaria um contador de créditos para contabilizar o espaço disponível no bloco IB do canal de entrada conectado ao outro extremo do enlace SoCIN.

A.2.4 O bloco *Output Controller*

O bloco *Output Controller* (OC) recebe as requisições oriundas dos canais de entrada, seleciona uma das requisições e envia um sinal de confirmação ao canal de entrada selecionado. Esse sinal de confirmação é utilizado como sinal de comando pelos blocos de chaveamento (ODS, ORS e IRS) para efetuar a conexão do canal de dados e dos sinais de controle de fluxo do bloco IB com o canal de saída solicitado. Uma vez estabelecida essa conexão, ela é mantida até que o último flit do pacote seja enviado e sua entrega seja confirmada pelo receptor. O bloco OC monitora a linhas *eop* e *rd* para avaliar essa condição, pois a ativação do bit *eop* sinaliza o envio do terminador do pacote e a posterior ativação do sinal *rd* sinaliza que o terminador foi transferido com sucesso. Quando essa seqüência de sinais é detectada, o bloco OC desativa a linha de confirmação que havia estabelecido a conexão corrente.

O bloco OC é organizado segundo o modelo PC-PO (Figura A .11) e sua parte operativa (PO) é constituída por um árbitro *Round-Robin* do tipo exaustivo [GUT 99] e por um circuito de registro dos sinais de confirmação. Já a parte controle (PC) consiste de uma máquina de estados que monitora as linhas de requisição e o fluxo dos dados no canal de saída de modo a comandar a atualização do apontador da fila do árbitro e do registrador de sinais de confirmação.

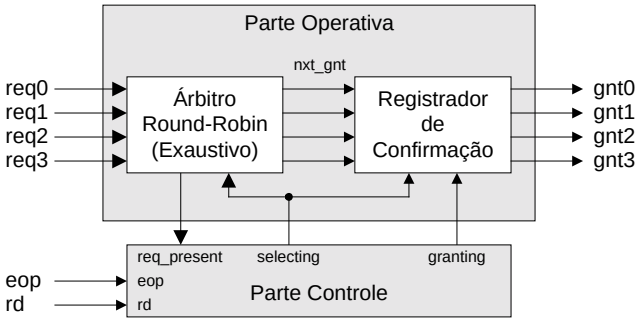


FIGURA A.11 – Estrutura do bloco OC.

A estrutura detalhada da parte operativa do bloco OC é ilustrada na Figura A.12, na qual observa-se a organização do árbitro e do registrador de confirmação. O árbitro recebe quatro sinais de requisição (req_i, onde i varia de 0 a 3) enviados pelos blocos IC dos canais de entrada que podem ser conectados ao canal de saída ao qual o bloco OC está associado. Esses sinais de requisição são encaminhados a quatro codificadores de prioridade estática (CPEs), sendo rotacionados na entrada desses codificadores de modo a gerar critérios de prioridade diferentes, cujas ordens de prioridade são indicadas pelo nome do sinal de saída de cada CPE. Por exemplo, a saída do primeiro CPE é denominada *req*0123, pois, para esse CPE, o sinal *req*0 recebe a maior prioridade e o sinal *req*3 recebe a menor prioridade. De maneira semelhante, a saída do terceiro CPE é denominada *req*2301, sendo que *req*2 recebe a maior prioridade e *req*1 recebe a menor. Ao mesmo tempo, o árbitro aplica uma lógica “OU” sobre os sinais de requisição para gerar o sinal *req_present* que é enviado à parte controle.

Após serem codificadas, as saídas dos CPEs são encaminhadas para um multiplexador que seleciona o CPE a ser utilizado com base em um ponteiro que indica a requisição que terá prioridade máxima na arbitragem corrente. A saída do multiplexador contém o índice da requisição selecionada (*sel*), o qual é encaminhado a um decodificador que gera um código de confirmação de seleção (*nxt_gnt*). Esse código é então armazenado em um registrador (*gnt_reg*) e mantido até o próximo ciclo de arbitragem. A saída desse registrador é conectada às saídas gnt_i (onde *i* varia de 0 a 3) quando o sinal *granting* da PC é igual a 1. Do contrário, todas as saídas gnt_i são mantidas em 0, sinalizando que nenhum canal de entrada está autorizado a ser conectado ao canal de saída associado ao bloco OC.

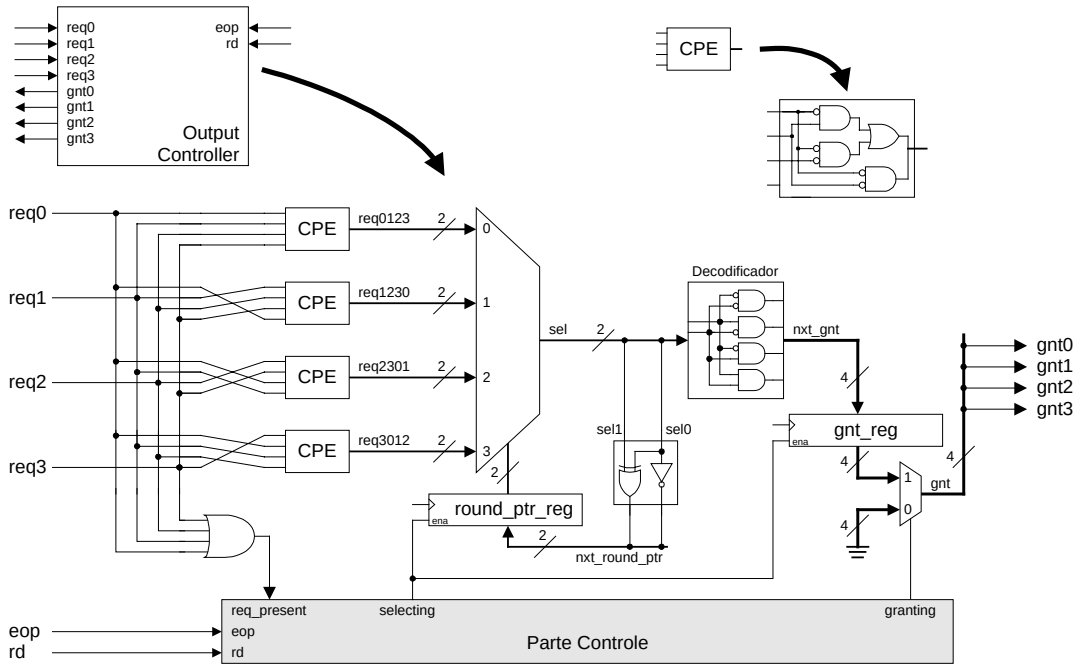


FIGURA A.12 – Estrutura da parte operativa do bloco OC.

O ponteiro que seleciona o CPE a ser utilizado na arbitragem é mantido no registrador *round_ptr_reg*. Esse registrador é atualizado a cada ciclo de arbitragem quando a saída *selecting* da PC é igual a 1. O novo valor desse ponteiro (*nxt_round_ptr*) é calculado de modo que a requisição selecionada na arbitragem corrente (indicada pelo índice *sel*) seja a de menor prioridade na arbitragem seguinte. Por exemplo, se requisição selecionada em um ciclo de arbitragem é a *req*2, no ciclo de arbitragem seguinte, o valor do *round_ptr_reg* será igual a 3 e o CPE a ser utilizado será aquele cuja saída é denominada *req*3012, no qual *req*2 tem prioridade mínima [ESP 2002].

A parte controle do bloco OC é baseada na máquina de estados de Moore ilustrada na Figura A .13, a qual possui quatro estados denominados *S_IDLE*, *S_ARB*, *S_WAIT_EOP* e *S_WAIT_LAST_RD*. Ela é inicializada no estado *S_IDLE* e nele se mantém até que o sinal *req_present* seja ativado pela chegada de alguma requisição. Quando isso ocorre, a máquina transita para o estado *S_ARB*, no qual ela ativa a saída *selecting* para notificar a parte operativa sobre a execução de um ciclo de arbitragem e comandar os registradores *round_ptr_reg* e *gnt_reg* para atualizarem seus estados. A máquina permanece nesse estado por um ciclo de relógio e então transita de maneira incondicional para o estado *S_WAIT_EOP*, ativando a saída *granting*. Esse sinal ativa o multiplexador que encaminha a saída do registrador *gnt_reg*

às saídas gnt_i, as quais comandam as chaves que conectam o canal de entrada selecionado ao canal de saída. A máquina se mantém nesse estado até que o terminador do pacote passe pelo canal de saída, o que é sinalizado pela ativação do bit de enquadramento *eop*. Se, simultaneamente a essa condição, é ativado o sinal *rd*, indicando que o terminador foi entregue ao receptor, a máquina retorna ao estado inicial (*S_IDLE*). Caso contrário, ela transita para o estado *S_WAIT_LAST_RD*, no qual se mantém até que *rd* seja ativado para então retornar ao estado *S_IDLE*. Uma vez que a máquina entra no estado *S_WAIT_EOP*, a saída *granting* é mantida ativada até que a máquina retorne ao estado inicial, conforme é indicado na Tabela A.8.

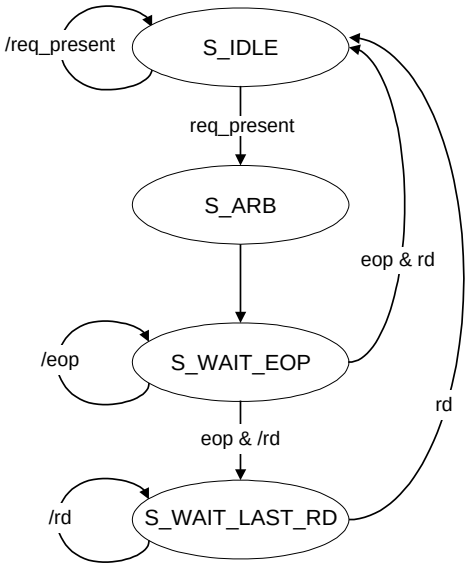


FIGURA A.13 – Máquina de estados da parte controle do bloco OC.

TABELA A.8 – Saídas da máquina de estados do bloco OC.

Estado	**Saídas**	
	selecting	***Granting***
S_IDLE	0	0
S_ARB	1	0
S_WAIT_EOP	0	1
S_WAIT_LAST_RD	0	1

Apêncice B Arquivos do CD-ROM

Na Tabela B.1 são descritos os arquivos contidos no CD-ROM em anexo.

TABELA B.1 – Arquivos contidos no CD-ROM.

Diretório	**Nome**	**Descrição**
SoCIN	socin_grid_2x2.vhd	Rede em grelha 2x2
RASoC	rasoc.vhd	Roteador RASoC
RASoC	input_channel.vhd	Canal de entrada
RASoC	input_flow_controller.vhd	Controlador de fluxo de entrada
RASoC	input_buffer.vhd	Buffer de entrada
RASoC	input_controller.vhd	Controlador de entrada
RASoC	input_rd_switch.vhd	Seletor de comando de leitura
RASoC	output_channel.vhd	Canal de saída
RASoC	output_flow_controller.vhd	Controlador de fluxo de saída
RASoC	output_controller.vhd	Controlador de saída
RASoC	output_data_switch.vhd	Seletor de comando de dado
RASoC	output_rok_switch.vhd	Selector de comando de *rok*
Traffic_Generator	traffic_generator_package.vhd	Pacote com definições de tipos
Traffic_Generator	traffic_generator_01.vhd	Gerador de múltiplos padrões de tráfego
Traffic_Generator	traffic_generator_single.vhd	Gerador de um único padrão de tráfego
Traffic_Generator	mux_Nx1.vhd	Seletor de padrão de tráfego
Traffic_Generator	ripple_rra.vhd	Árbitro *round-robin*
Traffic_Generator	ripple_ppe.vhd	Codificador de prioridades programável
Traffic_Generator	ripple_cell.vhd	Célula de arbitragem
Traffic_Generator	rrd.vhd	Deslocador de bits
SoC_SoCIN	soc_socin_grid_2x2.vhd	Grelha 2x2 e quatro geradores de tráfego

Referências

[ALT 2003] ALTERA. **Introduction to Quartus II**. ver. 3.0. San Jose, 2003. 192 p.

[ALT 2003a] ALTERA. **FLEX® 10K embedded programmable logic family data sheet**. ver. 4.3. San Jose, 2003. 128 p.

[AND 2001] ANDRIAHANTENAINA, A.; GREINER, A. **S.P.I.N.** 2001. 65 f. Relatório de Pesquisa (Doutorado) – Laboratoire d'Informatique Paris 6, Université Pierre et Marie Curie, Paris.

[AND 2003] ANDRIAHANTENAINA, A. et al. SPIN: a scalable, packet switched on-chip micro-network. In: DESIGN, AUTOMATION AND TEST IN EUROPE CONFERENCE AND EXIBITION, DATE, 2003, Munich. **Proceedings...** Los Alamitos: IEEE Computer Society, 2003. p. 70-73.

[ARM 2003] ARM. **AMBA:** the de facto standard for on-chip bus. Disponível em: <http://www.arm.com/armtech/AMBA?OpenDocument>. Acesso em: 12 maio 2003.

[ARV 90] ARVIND; NIKHIL, R. S. Executing a program on the MIT tagged token dataflow architecture. **IEEE Transactions on Computers**, New York, v. 39, n. 3, p. 300-318, Mar. 1990.

[ASH 98] ASHRAF, F. et al. Introduction to routing in multicomputer networks. **ACM Computer Architecture News**, New York, v. 26, n. 5, p. 14-21, Dec. 1998.

[AUS 2001] AUSTRIAMICROSYSTEMS. **0.35 νm CMOS design rules**. Austria, 2001. 30 p.

[AUS 2001a] AUSTRIAMICROSYSTEMS. **0.35 νm CMOS standard cell databook**. Austria, 2001. 338 p.

[AUS 2001b] AUSTRIAMICROSYSTEMS. **0.35 νm CMOS process parameters**. Austria, 2001. 55 p.

[BAR 99] BARUA, R. et al. Maps: A compiler-managed memory system for raw machines. In: INTERNATIONAL SYMPOSIUM ON COMPUTER ARCHITECTURE, ISCA, 26., 1999, Atlanta. **Proceedings...** Los Alamitos: IEEE Computer Society, 1999. p. 4-15.

[BEC 2003] BECK FILHO, A. C. S.; CARRO, L. **Comparação entre diferentes topologias de comunicação entre SOC**. 2003. 62 f. Relatório de Pesquisa (Mestrado) – Programa de Pós-Graduação em Computação, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2003.

[BEE 94] BEECROFT, J. et al. Meiko CS-2 interconnect elan elite design. **Parallel Computing**, [S.l.], v. 20, n. 10-11, p. 1627-1638, Nov. 1994.

[BEN 2001] BENINI, L.; DE MICHELI, G. Powering networks on chips. In: INTERNATIONAL SYMPOSIUM ON SYSTEM SYNTHESIS, ISSS, 14., Montreal, 2001. **Proceedings...** Los Alamitos: IEEE Computer Society , 2002. p.33-38.

[BEN 2002] BENINI, L.; DE MICHELI, G. Networks on chips: a new SoC paradigm. **IEEE Computer**, [S.l.], v. 35, n. 1, p. 70-78, Jan. 2002.

[BER 2000] BERGAMASCHI, R. A.; LEE, W. R. Designing systems-on-chip using cores. In: DESIGN AUTOMATION CONFERENCE, DAC, 37., 2000, Los Angeles. **Proceedings...** New York: ACM Press, 2000. p. 420-425.

[BER 2002] BERGAMASCHI, R. A. **The A to Z of SoCs.** 2002. 79 transparências, color., 270 mm ∝ 297 mm.

[BOD 95] BODEN, N. J. et al. Myrinet: a gigabit-per-second local area network. **IEEE Micro**, [S.l.], v. 15, n. 1, p. 29-36, Feb. 1995.

[BOR 90] BORKAR, S. et al. Supporting systolic and memory communication in iWarp. In: INTERNATIONAL. SYMPOSIUM ON COMPUTER ARCHITECTURE, ISCA, 17., 1990, Seatle. **Proceedings...** Los Alamitos: IEEE Computer Society, 1990. p. 70-81.

[BRI 2002] BRINKMANN, A. et al. On-chip interconnects for next generation system-on-chips. In: INTERNATIONAL ASIC/SOC CONFERENCE, 15., 2002, Rochester. **Proceedings...** Piscataway: IEEE Circuits and Systems Society, 2002. p. 211-215.

[CAR 2001] CARRO, L. **Projeto e protipação de sistemas.** Porto Alegre: Ed. da UFRGS, 2001. 171 p.

[CAR 99] CARRO, L. Memórias embarcadas. In: Escola de Microeletônica da SBC-Sul, 1., Pelotas, 1999. **Livro Texto**. Porto Alegre: Instituto de Informática da UFRGS, 1999. p. 201-210.

[CAS 2002] CARDOSO, R. S. et al. Analysis of applications in a grid interconnection network. In: SOUTH SYMPOSIUM ON MICROELECTRONICS, SIM, 17., 2002, Gramado. **Proceedings...** Porto Alegre: Instituto de Informática da UFRGS, 2002. p. 15-18.

[CAJ 96] CARBONARO, J.; VERHOORN, F. Cavallino: the teraflops router and NIC. In: HOT INTERCONNECTS SYMPOSIUM, 4., 1996, Stanford. **Proceedings…** Stanford: Stanford University, Aug.1996. p. 157-166.

[CIR 2003] CIRCUIT MULTI-PROJETS. **Manufacturing of ICs, Mems and MCMs**: reference manual. Grenoble, 2003. Disponível em: <http://cmp.imag.fr/CMPMan.html>. Acesso em: 15 mar. 2003.

[CLA 73] CLARE, C. C. **Designing logic systems using state machines**. New York: McGraw-Hill, 1973.

[COR 2003] CORRÊA, E. et al. Testing the wrappers of a network on chip: a case study. In: LATIN-AMERICAN TEST WORKSHOP, LATW, 4., Natal, 2003. **Proceedings...** Los Alamitos: IEEE Computer Society, 2003. p. 16-19.

[COT 2003] COTA, E. et al. The impact of NoC reuse on the testing of core-based systems. In: VLSI TEST SYMPOSIUM, VTS, 21., Napa Valley Marriot, 2003. **Proceedings...** Los Alamitos: IEEE Computer Society , 2003. p. 128-133.

[CUL 97] CULLER, D.; GUPTA, A.; SINGH, J. P. **Parallel computer architecture:** a hardware software approach. Los Altos: Morgan Kaufmann, 1998. 1100 p.

[DAL 2001] DALLY, W. J.; TOWLES, B. Route packets, not wires: on-chip interconnection networks. In: DESIGN AUTOMATION CONFERENCE, DAC, 38., LasVegas, 2001. **Proceedings...** Los Alamitos: IEEE Computer Society Press, 2002. p. 684-689.

[DAL 86] DALLY, W. J.; SEITZ, C. L. The torus routing chip. **Journal of Distributed Computing**, [S.l.], v. 1, n. 3, p. 187-196, Oct. 1986.

[DAL 87] DALLY, W. J.; SEITZ, C. L. Deadlock-free message routing in multiprocessors interconnection networks. **IEEE Transactions on Computers**, New York, v. C-36, n. 5, p. 547-553, May 1987.

[DAL 91] DALLY, W. J. Express cubes: improving the performance of *k*-ary *n*-cube interconnection networks. **IEEE Transactions on Computers**, New York, v. 40, n. 9, p. 1016-1023, Sept. 1991.

[DAV 83] DAVIO, M.; DESCHAMPS, J. P.; THAYSE, A. **Machines algorithmiques.** Lausanne: Presses Polytechniques Romandes, 1983.

[DEM 2001] DEMMELER, T.; GIUSTO, P. A. Universal communication model for an automotive system integration platform. In: DESIGN, AUTOMATION AND TEST IN EUROPE CONFERENCE AND EXHIBITION, DATE, Munich, 2001. **Proceedings...** Los Alamitos: IEEE Computer Society, 2001. p. 47-53.

[DOR 2000] DORNELES, R. V. et al. Parallel solution for shallow water equations using data decomposition. In: WORKSHOP DE SISTEMAS DISTRIBUÍDOS Y PARALELISMO, WSDP, 4., 2000, Santiago. **Memorias...** [S.l.: s.n], 2000. 1 CD-ROM.

[DOR 2000a] DORNELES, R. V. et al. PC cluster implementation of a mass transport two-dimensional model. In: SYMPOSIUM ON COMPUTER ARCHITECTURE AND HIGH PERFORMANCE COMPUTING, SBAC, 12., 2000, São Pedro. **Proceedings...** São Carlos: UFSCar, 2000. p. 191-198.

[DUA 97] DUATO, J.; YALAMANCHILI, S; NI, L. **Interconnection networks:** an engineering approach **.** Los Alamitos: IEEE Computer Society, 1997. 515 p.

[DUT 2001] DUTTA, S.; JENSEN, R.; RIECKMANN, A. Viper: a multiprocessor SOC for advanced set-top box and digital tv systems. **IEEE Design and Test of Computers**, [S.l.], v. 18, n. 5, p. 21-31, Sept.-Oct. 2001.

[ESP 2002] ESPÍRITO SANTO, F. G. M.; ZEFERINO, C. A. Projeto e avaliação de árbitros para redes-em-chip. **Hífen**, Uruguaiana, v. 26, n. 49/50, p. 81-86, 2002.

[FLE 80] FLETCHER, W. **An engineering aproach to digital design.** Englewodd Cliffs: Prentice Hall, 1980.

[GAL 97] GALLES, M. Spider: a high speed network interconnect. **IEEE Micro**, [S.l.], v. 17, n. 1, p. 34-39, Jan.-Feb. 1997.

[GLA 92] GLASS, C. J.; NI, L. M. The Turn Model for adaptive routing. In: INTERNATIONAL SYMPOSIUM ON COMPUTER ARCHITECTURE, ISCA, 19., 1992, Gold Coast. **Proceedings...** Los Alamitos: IEEE Computer Societ, 1992. p. 278-287.

[GRE 2001] GREINER, A. **Comment écrire un modèle CASS? Comment le déboguer?**: petit manuel a l'usage des debutants. ver. 0.1. Paris: Université Pierre et Marie Curie, Jan. 2001. 7p.

[GUE 2000] GUERRIER, P. **Un reseau d'interconnexion pour systèmes intégrés**. 2000. 146 f. Tese (Doutorado) – Laboratoire d'Informatique Paris 6, Université Pierre et Marie Curie, Paris, 2002.

[GUE 2000a] GUERRIER, P.; GREINER, A. A generic architecture for on-chip packet-switched interconnections. In: DESIGN, AUTOMATION AND TEST IN EUROPE CONFERENCE AND EXIBITION, DATE, 2000, Paris. **Proceedings...** Los Alamitos, IEEE Computer Society Press, 2000. p. 250-256.

[GUE 99] GUERRIER, P.; GREINER, A. A scalable architecure for system-on-chip interconnections. In: SOPHIA-ANTIPOLIS MICRO-ELECTRONICS CONFERENCE, SAME, 1999, France. **Proceedings…** Sophia Antipolis: [s.n.], 1999. p. 90-93.

[GUP 97] GUPTA, R. K.; ZORIAN, Y. Introducing core-based system design. **IEEE Design & Test of Computers**, [S.l.], v. 14, n. 4, p. 15-25, Oct.-Dec. 1997.

[GUR 85] GURD, J. et al. The Manchester prototype dataflow computer. **Communications of the ACM**, New York, v. 28, n. 1, p. 34-53, Jan. 1985.

[GUR 87] GURD, J. et al. The Manchester dataflow computing system. In: DONGARRA, J. J. (Ed.) **Experimental parallel computing systems**, Amsterdam: North-Holland, 1987. p. 177-219.

[GUT 99] GUPTA, P.; MCKEOWN, N. Designing and implementing a fast crossbar scheduler. **IEEE Micro,** [S.l.], v. 19, n. 1, p.20-28, Jan.-Feb. 1999.

[HO 2001] HO, R.; MAI, K.; HOROWITZ, M. A. The future of wires, **Proceedings of the IEEE**, [S.l.], v. 89, n. 4, p. 490-504, Apr. 2001.

[HU 2002] HU, J.; DENG, Y.; MARCULESCU, R. System-level point-to-point communication synthesis using floorplanning information. In: INTERNATIONAL CONFERENDE ON VLSI DESIGN, 15., 2002, Bangalore. **Proceedings...** Los Alamitos: IEEE Computer Society, 2002. p. 297-301.

[HWA 93] HWANG, K. **Advanced computer architecture:** parallelism, scalability, programmability. New York: McGraw-Hill, 1993. 770 p.

[IBM 99] IBM. **The CoreConnectTM bus architecture (White Paper)**. New York, 1999. 8 p.

[IBM 99a] IBM. **The RS/6000 SP high performance communication network**. New York. Disponível em: <http://www.rs6000.ibm.com/resource/technology/sp_sw1/spswp1.book_1.html>. Acesso em: 15 ago. 1999.

[IBM 99b] IBM, Interconnection Technologies for High-Performance Computing (RS/6000 SP). In: **IBM POWERparallel Technology Briefing**. New York. Disponível em: <http://www.rs6000.ibm.com/resource/technology/sp_sw2/spswp2_1.html>. Acesso em: 20 ago. 1999.

[ITO 2000] ITO, S. A.; CARRO, L.; JACOBI, R. P. Making Java work for microcontroller applications. **IEEE Design & Test of Computers**, [S.l.], v. 18, n. 5, p. 100-110, Sept.-Oct. 2001.

[ITR 2003] ITRS. **The international technology roadmap for semiconductors**. Disponível em: <http://public.itrs.net/>. Acesso em: 15 jun. 2003.

[KAR 2001] KARIM, A. N. F.; DEY, R. R. S. On-chip Communication architecture for OC-768 network processors. In: DESIGN AUTOMATION CONFERENCE, DAC, 38., LasVegas, 2001. **Proceedings...** New York: ACM Press, 2001. p. 678-683.

[KAR 2002] KARIM, A. N. F.; NGUYEN, A.; DEY, R. R. S. An interconnect architecture for networking systems on chips. **IEEE Micro**, [S.l.], v. 22, n. 5, p. 36-45, Sept.-Oct. 2002.

[KER 79] KERMANI, P.; KLEINROCK, L. Virtual cut-through: a new computer communication switching technique. **Computer Network**, [S.l.], v. 3, n. 4, p.267-286, Sept. 1979.

[KES 93] KESSLER, R. E.; SCHWARZMEIER, J. L. Cray T3D: a new dimension for Cray research. In: COMPCON, 1993, San Francisco. **Proceedings…** Los Alamitos: IEEE Computer Society, 1993. p. 176-82.

[KEU 2000] KEUTZER, K. et al. System-level design: orthogonalization of concerns and platform-based design. **IEEE Transactions on Computer-Aided Design of Integrated Circuits**, New York, v. 19, n. 12, p. 1523-1543, Dec. 2000.

[KIM 97] KIM, J.; KIM, Y. Performance analysis and tuning for a single-chip multiprocessor DSP. **IEEE Concurrency**, [S.l.], v. 5, n. 1, p. 68-79, Jan.-Mar. 1997.

[KON 91] KONSTANTINIDOU, S.; SNYDER, L. The Chaos router: architecture and performance. In: INTERNATIONAL SYMPOSIUM ON COMPUTER ARCHITECTURE, ISCA, 18., 1991, Toronto. **Proceedings…** New York: ACM Press, 1991. p. 79-88.

[KOR 99] KORNAROS, G. et al. ATLAS I: implementing a single-chip ATM switch with backpressure. **IEEE Micro**, [S.l.], v.19, n.1, p.30-41, Jan.-Feb. 1999.

[KRE 2001] KREUTZ, M. E. et al. Análise e seleção de redes de interconexão para síntese de sistemas no ambiente S³E²S. In: WORKSHOP IBERCHIP, 7., 2001, Montevideo. **Memorias...** [S.l: s.n.], 2001. 1 CD-ROM.

[KRE 2001a] KREUTZ, M. E.; ZEFERINO, C. A.; SUSIN, A. A. Trends in designing complex systems. In: MICROELECTRONICS SEMINAR, SIM, 16., 2001, Santa Maria. **Proceedings...** Porto Alegre: Instituto de Informática da UFRGS, 2000. p. 89-94.

[KRE 2001b] KREUTZ, M. E. et al. Communication architectures for system-on-chip. In: SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEM DESIGN, SBCCI, 14., 2001, Pirinópolis. **Proceedings...** Los Alamitos: IEEE Computer Society, 2001. p. 14-19.

[KRE 2001c] KREUTZ, M. E. et al. Análise e seleção de redes de interconexão para síntese de sistemas no ambiente S3E2S. **Revista de Informatica Teórica e Aplicada**, Porto Alegre, v. 8, n. 1, p. 83-101, 2001.

[KUM 2002] KUMAR, S. et al. A network on chip architecture and design methodology. In: SYMPOSIUM ON VERY LARGE INTEGRATION SCALE, Pittsburg, 2002. **Proceedings...** Los Alamitos: IEEE Computer Society, 2002. p. 105-112.

[LAN 2000] LANGEN, D.; BRINKMANN, A.; RUCKERT, U. High level estimation of the area and power consumption of on-chip interconnects. In**:** INTERNATIONAL ASIC/SOC CONFERENCE, 13., 2000, Arlington. **Proceedings...** Piscataway: IEEE Computer Society, 2000. p. 297-301.

[LAU 97] LAUDON, J.; LENOSKI; D. The SGI Origin: a ccNUMA highly scalable server. In: INTERNATIONAL SYMPOSIUM ON COMPUTER ARCHITECTURE, ISCA, 24., 1997, Denver. **Proceedings…** New York: ACM Press, 1997. p. 241-251.

[LIA 2000] LIANG, J. ; SWAMINATHAN, S.; TESSIER, R. aSOC: a scalable, single-chip communication architecture. In: INTERNATIONAL CONFERENCE ON PARALLEL ARCHITECTURES AND COMPILATION TECHNIQUES, ICPACT, 2000, Philadelphia. **Proceedings...** Los Alamitos: IEEE Computer Society, 2000. p. 37-46.

[MAD 97] MADISETTI, V. K.; SHEN, L. Interface design for core-based systems. **IEEE Design & Test of Computers**, [S.l.], v. 14, n. 4, p. 42-51, Oct.-Dec. 1997.

[MAT 2002] MATTOS, J. C. B.; CARRO, L. Efficient architecture for FPGA-based microcontrollers. In: INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, ISCAS, 2002, Scottsdale. **Proceedings…** Los Alamitos: IEEE Computer Society, 2002. v. 5, p. V-805-V-808.

[MAS 98] MATTSON, T. G.; HENRY, G. An overview of the Intel TFLOPS supercomputer. **Intel Technological Journal**. Disponível em: <http://developer.intel.com/technology/itj/q11998/articles/art_1.htm>. Acesso em: 13 abr. 2003.

[MET 92] METCALF, C. The NuMesh simulator. In: MIT LAB for Computer Science, **NuMesh Memo**, Cambridge, n. 5, Jan. 1992.

[NI 93] NI, L. M.; McKINLEY, P. K. A survey of wormhole routing techniques in direct networks. **IEEE Computer**, [S.l.], v. 26, n. 2, p. 62-76, Feb. 1993.

[NUG 88] NUGENT, S. The iPSC/2 direct connect communications technology. In: CONFERENCE ON HYPERCUBE CONCURRENT COMPUTERS AND APPLICATIONS, HCCA, 3., 1988, Pasadena. **Proceedings…** New York: ACM Press, 1988. p. 51-59.

[OCP 2001] OCP. **Open core protocol specificaction – release 1.0**. USA: OCP International Partnership, 2001. 202 p.

[PAT 96] PATTERSON, D.; HENNESSY, J. L. **Computer architecture:** a quantitative approach**.** San Francisco: Morgan Kaufmann, 1996. 760 p.

[PAU 97] PAULIN, P. et al. Embedded software in real-time signal processing systems: application and architecture trends. **Proceedings of the IEEE**, [S.l.], v. 85, n. 3, p. 419-454, Mar. 1997.

[PEH 2001] PEH, L. S.; DALLY, W. J. A delay model for router micro-architectures, In: **IEEE Micro**, [S.l.], v. 21, n. 1, p.27-34, Jan.-Feb. 2001.

[PET 97] PETROT, F. et al. Cycle-precise core based hardware/software system simulation with predictable event propagation, In: EUROMICRO CONFERENCE, 23., 1997. **Proceedings...** Los Alamitos: IEEE Computer Society, 1997. p.182-187.

[QUI 94] QUINN, M. J. **Parallel computing:** theory and practice. New York: Mc-Graw Hill, 1994. 446 p.

[RAB 96] RABAEY, J. M. **Digital integrated circuits**: a design perspective. Upper Saddle River: Prentice Hall, 1996.702 p.

[RIZ 2000] RIZZI, R. et al. Fluvial flow of the guaíba river: a parallel solution for the shallow water equations model. In: INTERNATIONAL MEETING ON VECTOR AND PARALLEL PROCESSING, VECPAR, 4., 2000, Porto. **Proceedings...** Porto: Faculdade de Engenharia da Universidade do Porto, 2000. p. 885-896.

[ROS 93] ROSE, J.; EL GAMAL, A.; SANGIOVANNI-VINCENTELLI. Architecture of field-programmable gate arrays. **Proceedings of IEEE**, [S.l.], v. 81, n. 7, p. 1013-1029, July 1997.

[SAK 83] SAKURAI, T. Approximation of wiring delay in MOSFET LSI. **IEEE Journal of Solid-State Circuits**, New York, v. 18, n. 4, p. 418-426, 1983.

[SEI 93] SEITZ, C. L.; SU, W. A family of routing and communication chips based on the Mosaic. In: SYMPOSIUM OF INTEGRATED SYSTEMS, Washington, 1993. **Proceedings…** Cambridge: MIT Press, 1993. p. 320 337.

[SGI 99] SGI. Performance of the Cray T3E multiprocessor. In: SGI, **Cray T3E White Papers**. Disponível em: <http://www.sgi.com/t3e/performance.html >. Acesso em: 27 ago. 1999.

[SHA 2002] SHAH, N.; KEUTZER, K. Network processors: origin of species. In: INTERNATIONAL SYMPOSIUM ON COMPUTER AND INFORMATION SCIENCES, ISCIS, 17., 2002, Orlando. **Proceedings...** [S.l.: s.n.], 2002.

[SHO 96] SHOEMAKER, D.; METCALF, C.; WARD, S. NuMesh: an architecture optimized for scheduled communication. **Journal of Supercomputing**, [S.l.], v. 10, n. 3, p. 285-302, 1996.

[SIE 94] SIEMENS. **OMI 324: PI-Bus – Ver.0.3d**. Munich, 1994. 35 p.

[STU 94] STUNKEL, C. B. et al. The SP-1 High Performance Switch. In: SCALABLE HIGH PERFORMANCE COMPUTING CONFERENCE, SHPCC, 1994, Knoxville. **Proceedings…** [S.l: s.n], 1994. p. 150-157.

[TAM 92] TAMIR, Y.; FRAZIER, G. L. Dynamically-allocated multi-queue buffers for VLSI communication switches. **IEEE Transactions on Computers**, New York, v.41, n.6, p. 725-737, June 1992.

[TAY 99] TAYLOR, D. et al. **On-chip interconnection network topologies: an emerging critical design.** 1999. 6 f. Project Report –Stanford University, Stanford.

[TEW 92] TEWKSBURY, S. K.; UPPULURI, M.; HORNAK, L. A. Interconnections/micro-networks for integrated microelectronics. In: IEEE GLOBAL TELECOMMUNICATIONS CONFERENCE, Orlando, 1992. **Proceedings...** Los Alamitos, IEEE Computer Society, 1992.p.180-186.

[TEX 97] TEXAS INSTRUMENTS. **IEEE Std 1149.1 (JTAG) testability primer**. Dallas, 1997. 141 p.

[VAH 2001] VAHID, F.; GIVARGIS, T. Platform tunning for embedded systems design. **IEEE Computer**, [S.l.], v. 34, n. 3, p. 112-114, Mar. 2001.

[VAI 2001] VAIDYA, A.S.; SIVASUBRAMANIAM, A.; DAS, C.R. Impact of virtual channels and adaptive routing on application performance. **IEEE Transactions on Parallel and Distributed Systems**, [S.l.], v. 12, n. 2, p. 223-237, Feb. 2001.

[VSI 2000] VSI ALLIANCE. **Virtual Component Interface Standard – OCB 2 1.0**. Los Gatos, 2000. 71 p.

[WAI 97] WAINGOLD, E. et al. Baring it all to software: raw machines. **IEEE Computer**, [S.l.], v. 30, n. 9, p. 86-93, Sept. 1997.

[ZEF 2000] ZEFERINO, C. A. et al. Parallel Simulation of the hydrodynamics of Guaíba river. In: MICROELECTRONICS SEMINAR, SIM, 15., 2000, Torres. **Proceedings...** Porto Alegre: Instituto de Informática da UFRGS, 2000. p. 22-24.

[ZEF 2002] ZEFERINO, C. A. et al. A study on communication issues for systems-on-chip. In: SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS, SBCCI, 15., 2002, Porto Alegre. **Proceedings...** Los Alamitos: IEEE Computer Society, 2002. p. 121-126. Disponível em: <http://www.inf.univali.br/~socin>. Acesso em: 13 maio 2003.

[ZEF 2002a] ZEFERINO, C. A. et al. Models for communication tradeoffs on systems-on-chip. In: INTERNATIONAL WORKSHOP ON IP-BASED SYSTEM-ON-CHIP DESIGN, 2002, Grenoble. **Proceedings...** [S.l: s.n.], 2002. p. 394-400. Disponível em: <http://www.inf.univali.br/~socin>. Acesso em: 13 maio 2003.

[ZEF 2002b] ZEFERINO, C. A. et al. Modelling communications on systems-on-chip. In: SOUTH SYMPOSIUM ON MICROELECTRONICS, SIM, 17., 2002, Gramado. **Proceedings...** Porto Alegre: Instituto de Informática da UFRGS, 2002. p. 149-152. Disponível em: <http://www.inf.univali.br/~socin>. Acesso em: 13 maio 2003.

[ZEF 2002c] ZEFERINO, C. A.; GREINER, A.; SUSIN, A. A. Evaluating On-Chip Communication Architectures. In: SOUTH SYMPOSIUM ON MICROELECTRONICS, SIM, 17., 2002, Gramado. **Proceedings...** Porto Alegre: Instituto de Informática da UFRGS, 2002. p. 136-139. Disponível em: <http://www.inf.univali.br/~socin>. Acesso em: 13 maio 2003.

[ZEF 2003] ZEFERINO, C. A.; SUSIN, A. A. A SoCIN: a parametric and scalable network-on-chip. In: SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS, 16., 2003, São Paulo. **Proceedings...** Los Alamitos: IEEE Computer Society , 2003. p. 169-174. Disponível em: <http://www.inf.univali.br/~socin>. Acesso em: 13 maio 2003.

[ZEF 96] ZEFERINO, C.A. **Projeto do sistema de comunicação de um multicomputador.** 1996. 121 p. Dissertação (Mestrado) – Curso de Pós-Graduação em Ciência da Computação, Universidade Federal de Santa Catarina, Florianópolis.

[ZEF 99] ZEFERINO, C.A.; SUSIN, A. A.; CARRO, L. Projeto de uma rede de interconexão experimental. In: WORKSHOP IBERCHIP, 5., 1999, Lima. **Memorias...** [S.l: s.n.], 1999. p. 277-284. Disponível em: <http://www.inf.univali.br/~socin>. Acesso em: 13 maio 2003.

[ZEF 99a] ZEFERINO, C. A.; BAMPI, S.; SUSIN, A. A. A study on interconnection networks for high performance parallel computers. In: MICROELECTRONICS SEMINAR, SIM, 14., 1999, Pelotas. **Proceedings...** Porto Alegre: Instituto de Informática da UFRGS, 1999. p. 33-40. Disponível em: <http://www.inf.univali.br/~socin>. Acesso em: 13 maio 2003.

[ZEF 99b] ZEFERINO, C. A. **Um estudo sobre a transformada discreta do cosseno e sua aplicação na codificação de vídeo digital**. 1999. 33 p. Trabalho Individual (Mestrado) – Programa de Pós-Graduação em Computação, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[ZEF 99c] ZEFERINO, C. A. **Redes de interconexão para multiprocessadores**. 1999. 123 f. Exame de Qualificação (Doutorado) – Programa de Pós-Graduação em Computação, Universidade Federal do Rio Grande do Sul, Porto Alegre. Disponível em: <http://www.inf.univali.br/~socin>. Acesso em: 13 maio 2003.

[ZHA 99] ZHANG, H. et. al. Interconnect architecture exploration for low-energy reconfigurable single-chip DSPs. In: ANNUAL WORKSHOP ON VLSI, Orlando, 1999. **Proceedings...** Los Alamitos: IEEE Computer Society, 1999. p. 2-8.