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

HENRIQUE COTA DE FREITAS

Arquitetura de NoC Programável Baseada em Múltiplos Clusters de Cores para Suporte a Padrões de Comunicação Coletiva

Tese apresentada como requisito parcial para a obtenção do grau de Doutor em Ciência da Computação

Prof. Dr. Philippe Olivier Alexandre Navaux
Orientador

Porto Alegre, junho de 2009.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Freitas, Henrique Cota de

Arquitetura de NoC Programável Baseada em Múltiplos Clusters de Cores para Suporte a Padrões de Comunicação Coletiva / Henrique Cota de Freitas. – Porto Alegre: Programa de Pós-Graduação em Computação, UFRGS, 2009.

125 p.:il.

Tese (doutorado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação. Porto Alegre, BR – RS, 2009. Orientador: Philippe Olivier Alexandre Navaux.

1.Network-on-Chip. 2.Computação Programável e Reconfigurável. 3.Padrões de Comunicação Coletiva. 4.Processadores Many-Core. I. Navaux, Philippe Olivier Alexandre. II. Título.

AGRADECIMENTOS

Agradecer parece algo relativamente fácil, afinal, é só dizer muito obrigado. Mas o verdadeiro agradecimento vem acompanhado de um sorriso no rosto, um abraço e uma felicidade em ter recebido a colaboração ou ajuda para a realização de uma tarefa. O sorriso não pode ser visto em palavras, portanto, gostaria de dizer o quanto estou feliz ao escrever os agradecimentos, citando pessoas que contribuíram de alguma forma para que este doutorado pudesse ser concluído.

Deus, obrigado pela proteção e oportunidades que me fizeram crescer como pessoa e profissional.

Ao meu filho Walter, dizer que o que faço é por você seria redundante. Mas por você sou redundante inúmeras vezes sem perceber.

Silvana, minha esposa. O que dizer? Abriu mão de uma vida pelo meu doutorado. Muitos beijos e abraços.

Aos meus pais Walter e Marlene. Sinônimos de dedicação à família. Agradeço tudo o que vocês têm feito pela minha família, e principalmente pelo Walter (neto).

Minha tia Maria Marta, mais do que uma tia. Compromete-se com a felicidade e realizações dos sobrinhos. Abraços às tias Dodora e Dione.

Túlio meu irmão, Ana Célia minha cunhada, e Ana Clara minha sobrinha. Viagens a Confins, incentivos e muita alegria que fazem minha família maior.

Aos meus tios e tias, obrigado pelos conselhos e incentivos.

Mudando de Minas Gerais para o Rio Grande do Sul, minha felicidade só foi possível graças a algumas pessoas:

Ao meu orientador Prof. Philippe Navaux, pela confiança e por acreditar em mim. Pelas conversas, reuniões, pelas oportunidades de crescimento científico, e pelo relacionamento que ultrapassa a de um orientador, mas de um amigo que se preocupa com o bem estar de todos os seus alunos.

Monica e Rafael Ávila, Tatiana e Rafael Santos. As conversas de laboratório vão deixar saudades, mas nada como uma mudança de cidade para mudar o lugar da conversa. Tatiana, a parceria com você foi um aprendizado durante o doutorado.

Marco, Eduardo, Felipe, Roberto, Vicente, Manuela e Camaratta do Lab. 201, Márcia, Lucas, Righi, Kassick, Mozart, Laércio, Danilo, Franciele e Thiago do Lab. 209. Obrigado pelas ajudas, prévias, artigos, e viagens que fizemos em conjunto. Foi muito bom trabalhar com vocês.

Aos professores e funcionários do Instituto de Informática, ao Programa de Pós-graduação em Computação, e a UFRGS, pelos conselhos, apoio e suporte durante o doutorado.

Ao CNPq pela bolsa de doutorado, que sem ela as dificuldades seriam maiores.

Voltando a Minas, obrigado PUC Minas e ao Programa Permanente de Capacitação Docente, que sem dúvida criaram as condições para que eu pudesse ir para Porto Alegre e realizar meu doutorado.

Gostaria de terminar os agradecimentos através de um provérbio chinês dito pelo Prof. David Patterson, em palestra proferida no ISCA 2008.

三个臭皮匠胜过一个诸葛亮

"A team is smarter than any single genius."

David Patterson, Beijin, China, 24 de junho de 2008

Portanto, agradeço novamente ao Prof. Navaux e à UFRGS pela oportunidade de ter assistido esta palestra. As frases ganham força quando existe uma história. Levo como bagagem para minha vida profissional o trabalho em grupo desenvolvido pelo Grupo de Processamento Paralelo e Distribuído (GPPD) da UFRGS.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS 7

LISTA DE FIGURAS 9

LISTA DE TABELAS 12

RESUMO 13

ABSTRACT 14

1 INTRODUÇÃO 15

1.1 Problema 17
1.2 Hipóteses 17
1.3 Objetivos e Meta 18
1.4 Motivações 18
1.5 Contribuição 18
1.6 Organização do Texto 19

2 ESTADO DA ARTE 21

2.1 Arquiteturas de Redes de Interconexão em Chip 21
2.1.1 Arquiteturas de Processadores com Múltiplos Núcleos 21
2.1.2 Características e Classificações das Redes de Interconexão 24
2.1.3 Principais Critérios para Avaliação das Redes de Interconexão 27
2.1.4 Networks-on-Chips (NoCs) 28
2.1.4.1 Arquiteturas de NoCs 29
2.1.4.2 Topologias de NoCs 30
2.1.4.3 Tipos de Protocolos 33
2.1.4.4 Análise da Viabilidade das NoCs 35
2.2 Padrões de Comunicação 36
2.2.1 Comunicação Coletiva 37
2.3 Computação Reconfigurável 39
2.3.1 Hardware Programável 41
2.3.2 Níveis e Tipos de Reconfiguração 42
2.4 Trabalhos Correlatos 43
2.4.1 NoCs Baseadas em Clusters de Núcleos de Processamento 44
2.4.2 NoCs com Roteadores Programáveis 45
2.4.3 NoCs Reconfiguráveis Baseadas em Dispositivos FPGA 46
2.4.4 NoCs Reconfiguráveis Baseadas em ASICs Polimórficos 48
2.4.5 Contribuição da Tese Versus Trabalhos Correlatos 49

3 METODOLOGIA DE PROJETO E AVALIAÇÃO 52

3.1 Simulação em ArchC e SystemC 54
3.2 Modelagem e Simulação Usando Redes de Petri 54
3.3 Simulação da Descrição de um Protótipo de Hardware 55
3.4 Avaliação de Desempenho de Cargas de Trabalho Paralelas 56

4 PROPOSTA DA ARQUITETURA MCNOC 60

4.1 Chave Crossbar Reconfigurável 61
4.2 Processador de Rede em Chip 66
4.2.1 Arquitetura e Conjunto de Instruções do NPoC 67
4.2.2 Suporte a Interleaved Multithreading 70
4.3 Roteador Programável 72
4.4 Arquitetura da MCNoC 76
4.4.1 Protocolos de Rede 77

5 RESULTADOS 80

5.1 Avaliação da Ocupação em FPGA 80
5.2 Avaliação do Tempo de Transmissão, e Consumo de Potência e Energia 83
5.2.1 Avaliação do Tempo de Transmissão de Pacotes 83
5.2.2 Avaliação do Consumo de Potência e Energia 86
5.3 Avaliação de Desempenho da MCNoC 89
5.4 Avaliação do Tempo de Execução de Programas no NPoC 100

6 CONCLUSÃO 105

6.1 Trabalhos Futuros 108
6.1.1 Simulador em ArchC e SystemC da MCNoC 108
6.1.2 Benchmarking Baseado em Cargas Paralelas para NoCs 108
6.1.3 Predição de Comunicação 108
6.1.4 Mapeamento de Processos e Identificação de Padrões 109
6.1.5 Modelo de Programação Híbrido Visando Reconfiguração de NoCs 109
6.1.6 Grids-on-Chips 110
6.1.7 Projeto de um Sistema Real para Experimentação 110
6.1.8 MCNoC Polimórfica 110

REFERÊNCIAS 112

ANEXO A RELATÓRIOS SUMARIZADOS DE SÍNTESES 120

ANEXO B PUBLICAÇÕES 123

LISTA DE ABREVIATURAS E SIGLAS

ALU	Arithmetic and Logic Unit
ACK	Acknowledgement
ASIC	Application Specific Integrated Circuit
ASIP	Application Specific Instruction Set Architecture
BCTU	Buffer and Crossbar Transfer Unit
BMT	Blocked Multithreading
CHNoC	Cluster-based Network-on-Chip
CPI	Ciclos Por Instrução
CMT	Chip Multithreading
CPLD	Complex Programmable Logic Device
End	Endereço
EPROM	Erasable Programmable Read Only Memory
EX	Execution
FF	Flip-Flop
FPGA	Field Programmable Gate Array
GigaNoC	Giga Network-on-Chip
GPP	General-Purpose Processor
hst	High Signal Transition
ID	Instruction Decoder
IF	Instruction Fetch
imed	Imediato
IMT	Interleaved Multithreading
IOB	Input and Output Block
IP	Intellectual Property
IPNoSys	Integrated Processing NoC System
ISA	Instruction Set Architecture
LB	Logic Block

lst	Low Signal Transition
LUT	Lookup Table
MCNoC	Multi-Cluster Network-on-Chip
MEM	Memória
MPGA	Mask Programmable Gate Array
MUX	Multiplexador
NAS	Numerical Aerodynamic Simulation
NPB	NAS Parallel Benchmark
NoC	Network-on-Chip
NPoC	Network Processor on Chip
opcode	Operation Code
PC	Program Counter
PLA	Programmable Logic Array
PNoC	Programmable Network-on-Chip
RAMP	Reconfigurable Architecture Multiprocessor
RANoC	Router Architecture for Network-on-Chip
RCS-NR	Reconfigurable Crossbar Switch for NoC Router
RISC	Reduced Instruction Set Computing
RdP	Redes de Petri
SCH	Scheduler
SMT	Simultaneous Multithreading
SoC	System-on-Chip
SoCIN	System-on-Chip Interconection Network
Val	Validação
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
WB	Write Back

LISTA DE FIGURAS

Figura 1.1: Visão geral da arquitetura de NoC programável proposta 19
Figura 2.1: Execução de múltiplas threads em um quad-core 22
Figura 2.2: Comparação entre arquiteturas. (a) superescalar, (b) multi-core 23
Figura 2.3: Arquiteturas Many-Core. Teraflops de desempenho 23
Figura 2.4: Justificativas do projeto Terascale. (a) utilização inteligente de núcleos, (b) redistribuição de carga de trabalho, (c) núcleos reservas, (d) integração de dispositivos dedicados 24
Figura 2.5: Topologias de redes de interconexão. (a) estrela, (b) árvore, (c) *mesh*, (d) *torus*, (e) hipercubo, e (f) pipeline 24
Figura 2.6: Classificação segundo tipo de chaveamento 26
Figura 2.7: Interconexões dinâmicas: barramento e chave crossbar 27
Figura 2.8: Interconexões dinâmicas multinível: (a) Banyan e (b) Omega 27
Figura 2.9: Evolução da interconexão global para NoC. (a) fio longo dominado pela resistência, (b) adição de repetidores ou buffers, (c) repetidores se tornam latches, e (d) latches evoluem para roteadores de NoC 28
Figura 2.10: Arquitetura básica de uma NoC com topologia mesh 29
Figura 2.11: Arquitetura básica de um roteador de NoC para topologia mesh 30
Figura 2.12: Topologias do projeto Terascale. (a) anéis interconectados, (b) mesh 33
Figura 2.13: Exemplos de comunicação coletiva 38
Figura 2.14: Comunicação coletiva cheia 39
Figura 2.15: Abordagens de projeto 40
Figura 2.16: Componentes básicos de um FPGA 42
Figura 2.17: Níveis de reconfiguração 42
Figura 2.18: Abordagens para NoCs com arquiteturas adaptáveis. (a) abordagens básicas, (b) abordagem derivada ASIP+ASIC, e (c) abordagem derivada FPGA+PArch 44
Figura 2.19: Abordagem ASIP+ASIC 45
Figura 2.20: Primeiro nível de reconfiguração baseado em FPGA 46
Figura 2.21: Primeiro nível de reconfiguração baseado em PASIC 48
Figura 3.1: Elementos de uma Rede de Petri 54
Figura 3.2: Visualização de padrões de comunicação através do Triva 58
Figura 4.1: Arquitetura de um chip multi-cluster 60
Figura 4.2: Arquitetura de um roteador-em-chip programável 61
Figura 4.3: Comunicação one-to-one (simultaneidade) 63
Figura 4.4: Comunicação one-to-all (broadcast) 63
Figura 4.5: Comunicação all-to-one 64
Figura 4.6: Efeito da latência através das simulações de RdPs 64

Figura 4.7: Exploração do segundo nível de reconfiguração através de protótipo em FPGA 65
Figura 4.8: Arquitetura da chave crossbar reconfigurável 66
Figura 4.9: Estágios de pipeline do processador de rede 67
Figura 4.10: Arquitetura do processador de rede em chip 68
Figura 4.11: Adiantamento de dados da instrução jump em ArchC 68
Figura 4.12: Formato de instruções do processador 69
Figura 4.13: Bancos de registradores compartilhados. (a) 4 threads, 4 bancos compartilhados, (b) 3 threads, 1 banco compartilhado 69
Figura 4.14: Fluxo de instruções do NPoC no pipeline IMT 71
Figura 4.15: Alterações no NPoC para suporte a IMT 71
Figura 4.16: Arquitetura do roteador em chip programável 72
Figura 4.17: Abordagens para identificação de padrões de comunicação 73
Figura 4.18: Modelo em RdP para identificação de padrões de comunicação 74
Figura 4.19: Funções if / else de prioridade na identificação de padrões de comunicação 75
Figura 4.20: Algoritmos para identificação de tráfego em oito buffers. (a) 1 pacote do SO requisita topologia, (b) todas as comunicações encerradas 75
Figura 4.21: Arquitetura da Multi-Cluster NoC 76
Figura 4.22: Handshake de duas vias 78
Figura 4.23: Cabeçalho de pacote de rede 78
Figura 4.24: Roteamento hierárquico 79
Figura 5.1: Tempo de transmissão one-to-one 85
Figura 5.2: Tempo de transmissão all-to-one 85
Figura 5.3: Tempo de transmissão broadcast 86
Figura 5.4: Topologias implementadas na RCS-NR. (a) Estrela, (b) Árvore Balanceada, (c) Árvore Binomial, (d) Anel, (e) Hipercubo e (f) Pipeline 87
Figura 5.5: Consumo de potência do RANoC no FPGA 2vp100ff1704-6 87
Figura 5.6: Consumo de potência da MCNoC no FPGA 2vp100ff1704-6 88
Figura 5.7: Visualização do Triva para comunicação one-to-one cheia (Traço BT) 89
Figura 5.8: Visualização do Triva de intervalo one-to-one (Traço BT). (a) one-to-two e one-to-one, (b) e (c) one-to-two e two-to-one, e (d) two-to-one 89
Figura 5.9: Visualização do Triva de intervalo one-to-one (Traço BT). (a) one-to-three e two-to-one, (b) three-to-one, two-to-one e one-to-two, e (c) three-to-one e one-to-one 90
Figura 5.10: Padrões de comunicação coletiva. (a) one-to-one, (b) one-to-all, (c) all-to-one, e (d) all-to-all 90
Figura 5.11: Métricas de avaliação de desempenho 92
Figura 5.12: Somas relativas dos tempos de transmissão. (a) transmissão e interseção removida, (b) transmissão removida, (c) interseção removida, e (d) sem remoção 93
Figura 5.13: Tempo de transmissão do padrão de comunicação one-to-one 94
Figura 5.14: Tempo de transmissão do padrão de comunicação one-to-all 95
Figura 5.15: Tempo de transmissão do padrão de comunicação all-to-one 95
Figura 5.16: Tempo de transmissão do padrão de comunicação all-to-all 96
Figura 5.17: Tempo de transmissão considerando todos os padrões de comunicação 97
Figura 5.18: Tempo de transmissão considerando todos os padrões de comunicação (pacote de tamanho máximo igual a 4096 bytes) 97
Figura 5.19: Buffers de saída para topologias mesh e torus 98
Figura 5.20: Programa ComPat em linguagem de montagem NPoC 101
Figura 5.21: Ganho de desempenho do NPoC com suporte IMT 102

Figura 6.1: Modelo híbrido de programação 109
Figura 6.2: MCNoC polimórfica 111

LISTA DE TABELAS

Tabela 2.1: Classificação segundo forma de comunicação 26
Tabela 2.2: Exemplos de topologias de NoCs 31
Tabela 2.3: Características de algumas topologias fixas 32
Tabela 2.4: Análise comparativa para uso de NoCs 35
Tabela 2.5: Comparações entre características de NoCs relacionadas 51
Tabela 3.1: Programas paralelos utilizados do NAS 58
Tabela 4.1: Exemplos de instruções de propósito geral 70
Tabela 4.2: Instruções de rede 70
Tabela 4.3: Lista de condições do modelo em RdP 74
Tabela 4.4: Lista de ações do modelo em RdP 74
Tabela 5.1: Utilização de componentes do FPGA 2vp100ff1704-6para RANoC 81
Tabela 5.2: Utilização de componentes do FPGA 2vp100ff1704-6 para MCNoC 81
Tabela 5.3: Quantidade de BRAMs utilizadas pela MCNoC do FPGA 2vp100ff1704-6 82
Tabela 5.4: Redes roteadas do FPGA 2vp100ff1704-6 82
Tabela 5.5: Utilização de componentes do FPGA 2vp100ff1704-6 para NPoC com e sem IMT 83
Tabela 5.6: Resultados de freqüência de interconexão e vazão (throughput) no FPGA 2vp100ff1704-6 84
Tabela 5.7: Consumo de energia no FPGA 2vp100ff1704-6 88
Tabela 5.8: Quantidade e tamanho das mensagens coletadas 91
Tabela 5.9: Principais características das NoCs modeladas 92
Tabela 5.10: Execução de programas em versões do NPoC 101
Tabela 5.11: Tempos entre fim e início de um novo padrão (tamanho máximo de pacotes igual a 128 *bytes*) 103
Tabela 5.12: Alterações pelo uso do ComPat na aplicação EP 103
Tabela 5.13: Alterações pelo uso do ComPat na aplicação FT 104

RESUMO

As próximas gerações de processadores *many-core* exigem que novas abordagens no projeto de arquitetura de processadores sejam propostas. Neste novo contexto, as redes de comunicação *intra-chip* são importantes para garantir o desempenho dos programas. Soluções tradicionais de interconexão possuem limites físicos que comprometem a escalabilidade e o desempenho no processamento de aplicações paralelas de diversos tipos. A alternativa apontada pelo estado da arte é a *Network-on-Chip* (NoC) composta por roteadores e outros elementos de rede capazes de prover comunicação escalável e de alto desempenho.

No entanto, as cargas de trabalho geram padrões de comunicação diferentes que podem influenciar no desempenho da rede. Existem pesquisas que abordam metodologias de projeto dedicado de NoCs em função de domínios de aplicações específicos. Apesar de uma NoC dedicada possuir um alto desempenho, cargas de trabalho paralelas geram padrões de comunicação coletiva que mudam dinamicamente. Com o objetivo de aumentar a flexibilidade de redes-em-*chip*, trabalhos correlatos utilizam conceitos de computação reconfigurável para aumentar a capacidade da arquitetura da NoC se adaptar em função de padrões de comunicação. Alguns trabalhos focam na programação de FPGAs e outros em ASICs polimórficos.

O objetivo desta tese é propor uma arquitetura de *Network-on-Chip* que suporte múltiplos *clusters* de núcleos de processamento através de roteadores programáveis e de topologias reconfiguráveis. Cada roteador é composto por uma chave *crossbar* reconfigurável capaz de implementar topologias dinamicamente através do uso de um segundo nível de reconfiguração. Os roteadores possuem processadores de rede que aumentam a flexibilidade e a capacidade da NoC se adaptar ao padrão de comunicação através de programas que monitoram e gerenciam a rede. Portanto, a contribuição da tese é a Arquitetura de NoC Programável Baseada em Múltiplos Clusters de Cores.

Os resultados baseados em modelos analíticos e de simulação, e cargas de trabalho artificiais e naturais, mostram que a arquitetura da NoC possui um alto desempenho e vazão de pacotes, proporcionados pela adaptação de topologias e redução da influência da rede na comunicação. A ocupação em FPGA mostra que os roteadores programáveis possuem tamanho similares a NoCs com arquiteturas tradicionais para gerenciamento de mesma quantidade de núcleos. A menor utilização de *buffers* de entrada resulta em uma melhor eficiência no consumo de potência e energia. Portanto, através dos modelos de projeto e avaliação foi possível verificar através dos resultados que a arquitetura da MCNoC é uma alternativa para suportar padrões de comunicações coletivas.

Palavras-Chave: Arquitetura de Network-on-Chip, Computação Programável e Reconfigurável, Padrões de Comunicação, Processadores Many-Core.

Programmable Multi-Cluster NoC Architecture to Support Collective Communication Patterns

ABSTRACT

For the next generation of many-core processors, new design methodologies must be proposed. In this context, on-chip interconnections are important to assure the program performance. Traditional approaches of interconnections have physical constraints that reduce the scalability and performance to process parallel applications. The state-of-the-art points out to the Network-on-Chip (NoC), which consists of routers and other network devices capable of increasing the communication scalability and performance.

However, workloads produce different types of communication patterns, which can influence the network performance. There are research works that explore application-specific NoC design to response the demand on specific workloads. Although a dedicated NoC has a high performance, parallel workloads have different collective communication patterns. In order to increase the flexibility of NoCs, related works use concepts of reconfigurable computing to add architecture adaptability to support dynamic communication patterns. Some works focus on FPGA-based reconfiguration and others on polymorphic ASICs.

The goal of this thesis is to propose an alternative Programmable Multi-Cluster NoC architecture. Each router consists of a reconfigurable crossbar switch capable of implementing dynamic topologies through a second reconfiguration level. The routers have network processors that increase the flexibility and the NoC adaptability through management programs in order to support different workloads. Therefore, the contribution of this thesis is the following: A Programmable Multi-Cluster NoC (MCNoC) architecture.

Based on analytical and simulation models, and artificial and natural workloads, results show the high performance and throughput for the proposed NoC architecture, due to the adaptable topologies and low network latency impact. Results based on FPGA shows a similar component utilization considering the proposed programmable NoC relative to conventional NoC architectures for the same number of processing cores. The low utilization of input buffers improves the efficiency of power and energy consumption. Therefore, through design and evaluation models, the NoC proposal was verified and the results point out the MCNoC as an alternative architecture to support collective communication patterns.

Keywords: Network-on-Chip Architecture, Programmable and Reconfigurable Computing, Communication Patterns, Many-Core Processors.

1 INTRODUÇÃO

Uma das principais motivações durante o projeto da arquitetura de um processador está relacionada à carga de trabalho e seu comportamento. A evolução dos processadores *single-core* nas últimas décadas mostrou uma intensa busca pela solução de problemas associados aos programas seqüenciais através de técnicas para o suporte ao paralelismo no nível de instrução. No entanto, devido a restrições no paralelismo de instruções, consumo de potência, além dos limites impostos pela latência do fio, o ganho de desempenho das gerações de processadores com um único núcleo (*single-core*) tem diminuído a taxas inferiores ao que se esperava (OLUKOTUN, 1996) (HO, 2001) (KEYES, 2001) (KEYES, 2008). A solução para esta desaceleração no ganho de desempenho dos processadores *single-core* está em uma nova geração de processadores com múltiplos núcleos (*multi-core*) (WOLF, 2004) (KUMAR, 2004) (KUMAR, 2005a) (OLUKOTUN, 2005) (HILL, 2008). Atualmente os núcleos de um processador *multi-core* são mais simples, alcançando uma maior eficiência no consumo de potência e energia. Porém, múltiplos núcleos aumentam a capacidade de exploração do paralelismo no nível de *thread*, representam uma organização nativa para suportar programas paralelos e troca de mensagens, mas podem ocasionar uma intensa comunicação interna.

Como conseqüência do aumento do número de núcleos, há também a necessidade de uma rede de comunicação entre os núcleos de alto desempenho e eficiente. Soluções tradicionais (ANDERSON, 1975) (AHMADI, 1989) (AGARWAL, 1991) (KUMAR, 2005b) de interconexão, e.g., barramentos e chaves *crossbar*, possuem problemas de escalabilidade e limites físicos. Portanto, aumentar o tamanho destas soluções de interconexão implica diretamente nas seguintes conseqüências: aumento da resistência do fio, aumento da latência de comunicação, e aumento da complexidade de roteamento do fio. Como conseqüência, a comunidade tem trabalhado com o objetivo de reduzir a influência dos limites físicos dos fios através de uma arquitetura de rede chamada de NoC (*Network-on-Chip*) (BENINI, 2002) (BENINI, 2005) (BJERREGAARD, 2006) (OGRAS, 2007b).

As NoCs são redes de comunicação em *chip* baseadas em troca de pacotes, que possuem os seguintes componentes básicos: roteadores, adaptadores de rede, e *links* de comunicação. A maioria das NoCs é projetada para que um roteador seja responsável pela interconexão de um núcleo à rede de comunicação. Por exemplo, em uma rede com topologia *mesh*, um roteador teria entre o núcleo ou demais roteadores adjacentes *links* curtos com baixa resistência e latência de comunicação. Em contrapartida, cada roteador aumenta em complexidade a gerência de comunicação interna, podendo ou não

aumentar a latência de comunicação. Como conseqüência do uso das NoCs, existe uma maior escalabilidade da rede de comunicação e, portanto, a possibilidade de suportar uma maior quantidade de núcleos de processamento.

Chips com milhares de núcleos ainda é uma projeção, mas os desafios para viabilizar *chips* com dezenas ou centenas de núcleos já começam a ser enfrentados e resolvidos. Os processadores com uma quantidade elevada de núcleos são chamados de *many-core*. Um exemplo deste tipo de processador de propósito geral em desenvolvimento é da linha de pesquisa Terascale da Intel (INTEL, 2007). Este processador possui oitenta núcleos de processamento e uma rede de comunicação, que inicialmente pode ser uma *mesh* ou um conjunto interconectado de anéis. No entanto, o aumento da quantidade de núcleos implica em um aumento do tráfego de pacotes podendo gerar gargalos na rede e perda de desempenho do processador.

O impacto no desempenho pelas redes de comunicação é amplamente conhecido pela comunidade de processamento paralelo (GRAMA, 1993) (CULLER, 1999) (DONGARRA, 2003) (DE ROSE, 2003). A troca de mensagens pode ser afetada pela latência da rede e com isto o desempenho pode degradar. Como exemplo, o simples aumento da quantidade de nós de processamento não significa aumento de desempenho. À medida que o número de processadores aumenta, o ganho de desempenho também aumenta, mas a partir de uma certa quantidade de processadores este ganho começa a diminuir, uma vez que a rede insere atrasos na comunicação. *Clusters* multiprocessados (BUYYA, 1999) têm sido amplamente utilizados como infra-estrutura de alto desempenho na solução de problemas paralelos devido à grande escalabilidade oferecida pela rede de comunicação. Projetar NoCs baseadas em *clusters* (LENG, 2005) (NIEMANN, 2005) pode reduzir o impacto da própria NoC no desempenho do *chip*, mas sua influência continua sendo um fator importante. Apesar da abordagem baseada em *clusters*, o crescimento da NoC pode degradar o desempenho, se a utilização dos núcleos de processamento não for otimizada em função das aplicações e dos roteadores da NoC que gerenciam os *clusters*.

Portanto, os principais problemas referentes aos limites físicos e de desempenho das *Networks-on-Chips* podem ser descritos através dos seguintes itens:

- Área ocupada: Por questões dos limites físicos impostos e em função de um melhor aproveitamento do *chip*, a NoC deve ocupar um espaço relativamente pequeno, para que a área seja melhor ocupada por uma maior quantidade de núcleos de processamento.
- Consumo de Potência e Energia: A NoC deve consumir pouca potência e energia para que o impacto no consumo do *chip* seja pequeno.
- Latência da rede: O atraso da NoC para iniciar uma comunicação ou entregar um pacote deve ser pequeno para que o desempenho do processador não seja degradado.
- Largura de banda e vazão da rede: A NoC deve possuir uma largura de banda e arquitetura que propicie uma alta vazão de pacotes.

Existem pesquisas que buscam soluções de arquiteturas dedicadas de NoCs para problemas ou aplicações específicas (BERTOZZI, 2005) (HO, 2006). No entanto, processadores de propósito geral podem executar cargas de trabalhos de diferentes tipos. Neste contexto, redes que se adaptam às características de aplicações dinamicamente podem ser mais adequadas. Portanto, seria possível explorar

processamento paralelo e distribuído em *chip* alterando topologias e aumentando a vazão de aplicações e o desempenho do processador.

Para exploração do processamento paralelo em processadores *many-core*, é necessária uma arquitetura de *Network-on-Chip* que atenda requisitos de flexibilidade e desempenho. Uma das alternativas é o uso de conceitos de computação reconfigurável (COMPTON, 2002) (TODMAN, 2005). A aplicação dos conceitos de reconfiguração para mapeamento de cargas de trabalho paralelas em NoCs, pode através da exploração da flexibilidade, aumentar o desempenho na entrega dos pacotes pela rede, reduzindo a influência dos roteadores e como conseqüência o tempo de transmissão de rede.

As próximas seções apresentam de forma detalhada a descrição do problema abordado pela tese com foco nas comunicações coletivas oriundas de programas paralelos, além das hipóteses, objetivos, motivações e contribuição, que justificam o projeto de uma Arquitetura de NoC Programável baseada em Múltiplos Clusters de Cores.

1.1 Problema

A arquitetura de rede pode beneficiar um determinado comportamento ou padrão de comunicação de uma aplicação. Considerando a alta demanda por programas paralelos para as novas gerações de processadores *many-core*, existe a necessidade de que uma arquitetura de NoC seja capaz de suportar com alto desempenho a intensa comunicação coletiva e, conseqüentemente, troca de pacotes entre todos os núcleos envolvidos no processamento paralelo.

O padrão de comunicação coletiva descreve uma possível configuração de interconexões entre origens e destinos. Isto significa que é possível ter uma comunicação mais intensa em determinados *links* da rede ou em determinados períodos de tempo. Conseqüentemente, o padrão de comunicação pode mudar ao longo do tempo e pode interferir no desempenho da rede. Neste caso, uma rede adaptável pode ganhar em desempenho em relação a uma rede estática. Da mesma forma, técnicas diferentes de reconfiguração podem tornar uma rede adaptável melhor do que outra. Portanto, o problema abordado por esta tese pode ser expresso pela seguinte afirmação:

- Comunicações coletivas demandam intensa comunicação entre núcleos, e a arquitetura de uma rede-em-*chip* adaptável pode não oferecer suporte adequado, aumentando o atraso entre as diversas comunicações.

O problema nos leva a seguinte pergunta:

- Como uma arquitetura de NoC pode se adaptar a padrões de comunicação coletiva para que seja possível reduzir o tempo de transmissão de pacotes?

1.2 Hipóteses

Para reduzir o impacto dos problemas apresentados, algumas hipóteses podem ser levantadas focando um novo projeto de *Network-on-Chip* adaptável, são elas:

- Topologias reconfiguráveis, através de um segundo nível de reconfiguração, podem aumentar o desempenho da rede de comunicação em *chip*.
- Para gerenciar o segundo nível de reconfiguração, os roteadores da NoC devem ser programáveis.

- Em função do impacto da NoC no desempenho do *chip*, um roteador de NoC deve suportar um *cluster* de núcleos de processamento, para reduzir o efeito das latências.

1.3 Objetivos e Meta

Com base nas hipóteses descritas o objetivo principal desta tese é projetar uma arquitetura de *Network-on-Chip* que suporte múltiplos *clusters* de núcleos de processamento através de roteadores programáveis e de topologias reconfiguráveis. Este objetivo principal é alcançado através dos seguintes objetivos intermediários:

- Projetar uma arquitetura de roteador programável para *clusters* de núcleos de processamento.
- Projetar uma arquitetura de processador de rede para o roteador.
- Projetar uma arquitetura de chave *crossbar* reconfigurável e *bufferizada* para o roteador.
- Projetar uma topologia de interconexão entre roteadores para a NoC.
- Definir protocolos e estratégias de gerenciamento da NoC.
- Avaliar a NoC através de cargas de trabalho e métricas específicas.

A meta é mostrar através da metodologia de trabalho que a arquitetura de NoC proposta nesta tese é viável e que pode ser uma alternativa de rede-em-*chip* adaptável para futuros processadores *many-core*.

1.4 Motivações

A próxima geração de processadores *many-core* abre uma grande possibilidade de pesquisa. O número de problemas está relacionado ao funcionamento do processador e ao suporte do mesmo aos diversos tipos de aplicações paralelas. Neste contexto, é possível explorar novos projetos de arquiteturas de *Networks-on-Chips* e como estas podem suportar com um melhor desempenho e baixo custo a demanda por processamento paralelo.

Portanto, as principais motivações para tese podem ser apresentadas em dois itens básicos:

- A próxima geração de processadores *many-core*.
- A demanda por arquiteturas de NoCs adaptáveis que suportem as características de comunicação coletiva de programas paralelos.

1.5 Contribuição

Conforme é apresentado na seção de Trabalhos Correlatos (2.4), existe um conjunto de pesquisas que busca melhorar o desempenho de redes-em-*chip* adaptáveis através do uso de arquiteturas programáveis ou reconfiguráveis. Em função dos problemas e objetivos apresentados, esta tese contribui com uma arquitetura de NoC que busca suportar características típicas de programas paralelos baseados em padrões de comunicação coletiva.

Portanto, a contribuição desta tese é a seguinte:

- Arquitetura de NoC Programável Baseada em Múltiplos *Clusters* de *Cores*, ilustrada pela Figura 1.1.

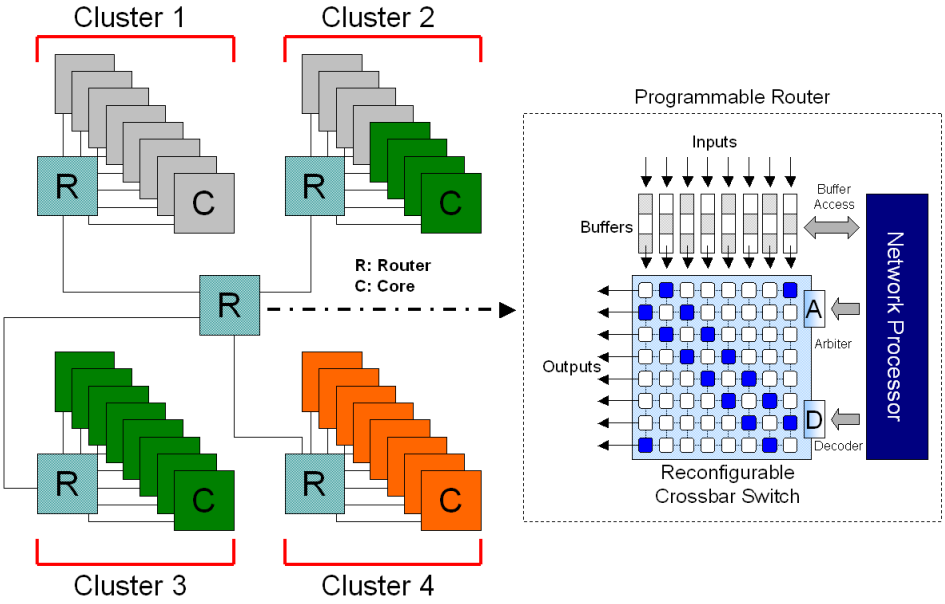


Figura 1.1: Visão geral da arquitetura de NoC programável proposta

As principais características da arquitetura da NoC podem ser apresentadas através dos blocos construtivos descritos a seguir:

- Arquitetura de roteador programável e reconfigurável para suporte a *clusters* de núcleos de processamento.
- Arquitetura de processador de rede em *chip* para roteadores de NoCs.
- Arquitetura de chave *crossbar* reconfigurável para roteadores de NoCs.
- Arquitetura de NoC com topologias adaptáveis aos padrões de comunicação coletiva.

1.6 Organização do Texto

A tese está organizada de acordo com os seguintes capítulos:

- Capítulo 2 - Estado da Arte: É apresentada uma revisão bibliográfica sobre os principais conceitos referentes ao tema da tese, além dos trabalhos correlatos.
- Capítulo 3 - Metodologia de Projeto e Avaliação: É apresentada a definição dos modelos, técnicas e ferramentas. Estes tópicos são apresentados e relacionados em etapas para desenvolvimento da proposta e obtenção dos resultados de validação.
- Capítulo 4 - Proposta da Arquitetura MCNoC: Neste capítulo a arquitetura da MCNoC e dos seus blocos construtivos são apresentados em detalhes.
- Capítulo 5 - Resultados: São definidas métricas no capítulo de metodologia que são necessárias para verificação da proposta de arquitetura. As

avaliações dos resultados baseadas nestas métricas são apresentadas neste capítulo.

- Capítulo 6 - Conclusão: Neste capítulo é feita a conclusão final da tese, relacionando os resultados, objetivos, e problemas apresentados. Com base nas conclusões são apontados também os trabalhos futuros que poderão ser realizados após a defesa da tese.

2 ESTADO DA ARTE

A rápida evolução da área de arquitetura de computadores tem mostrado que é necessária cada vez mais uma maior interação entre áreas relacionadas. Projetos de processadores *multi-core*, principalmente *many-core*, são exemplos da necessidade de um estudo mais amplo capaz de abranger áreas até certo ponto distintas, tais como: redes de comunicação de dados, caracterização de cargas de trabalho paralelas e computação reconfigurável. Além disso, o projeto de um processador *many-core* requer conhecimento de métricas de avaliação de desempenho (JAIN, 1991) (MENASCÉ, 2003) (LILJA, 2004) (LE BOUDEC, 2007) para identificação do impacto de novas alternativas e contribuições para a área.

Neste capítulo é feita uma revisão bibliográfica das principais áreas relacionadas à tese e que influenciam no projeto da arquitetura da NoC Programável baseada Múltiplos *Clusters* de *Cores*. Neste contexto, é importante um estudo dos diversos conceitos relativos aos tipos de redes de interconexões e *Networks-on-Chip*, padrões de comunicação coletiva, computação reconfigurável, além dos trabalhos correlatos à proposta de arquitetura. Portanto, as seções deste capítulo apresentam os conceitos mais relevantes relacionados à tese.

2.1 Arquiteturas de Redes de Interconexão em Chip

O objetivo desta seção está na descrição dos principais conceitos relacionados às redes de interconexão e NoCs. No entanto, a Subseção 2.1.1 faz uma breve apresentação de arquiteturas de processadores com múltiplos núcleos com o objetivo de ilustrar a necessidade dos projetos de NoCs para a próxima geração de processadores *many-core*.

2.1.1 Arquiteturas de Processadores com Múltiplos Núcleos

Apesar de processadores com múltiplos núcleos existirem desde a década passada, em sistemas dedicados, e.g., processadores de rede (INTEL, 2001) (COMER, 2003), o surgimento em arquiteturas de propósito geral tem alguns anos. Os fatores que mais motivaram o surgimento destes processadores são os limites do fio e o alto consumo de potência. A base do desempenho dos processadores se valia pelo paralelismo no nível de instrução, com *pipeline* superescalar, no máximo duas *threads* simultâneas (SMT – *Simultaneous Multithreading*) e a constante elevação da freqüência de operação. Porém, o consumo de potência já atingia limites impraticáveis, o paralelismo no nível de instrução já havia encontrado limites de desempenho e múltiplas *threads* simultâneas em um mesmo *pipeline* muitas vezes degradava o desempenho em função da

concorrência pelos mesmos recursos ou unidades funcionais. Sendo assim, era necessária uma nova abordagem para a constante evolução no aumento do desempenho, atendendo limites de consumo de potência. Os processadores com múltiplos núcleos, já usados em outras áreas, se tornaram alternativas para os computadores de propósito geral.

A Figura 2.1 mostra um exemplo de arquitetura *quad-core*, sendo que cada núcleo possui um *pipeline* superescalar de duas vias. Em comparação com outras arquiteturas, cada um dos núcleos pode ser considerado relativamente simples. Neste exemplo, o objetivo é suportar tanto paralelismo no nível de instrução quanto no nível de *thread*. No entanto, as *threads* são somente suportadas pela existência de mais de um núcleo, já que cada núcleo não é *multithreaded*, sendo responsável pela execução de apenas uma *thread*. Processadores *multi-core* que suportam múltiplas *threads* são também conhecidos como *Chip Multithreading* (CMT) (UNGERER, 2002) (UNGERER, 2003) (SPRACKLEN, 2005) (FREITAS, 2006c).

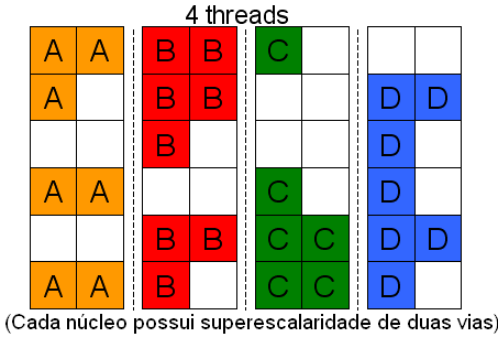


Figura 2.1: Execução de múltiplas threads em um quad-core (FREITAS, 2009a)

Existem processadores *multi-core* de propósito geral que suportam múltiplas *threads* em cada núcleo, através de *Interleaved Multithreading* - IMT (KONGETIRA, 2005) ou SMT (KALLA, 2004). O motivo para escolha de uma técnica ou outra se deve muito a característica das cargas de trabalho. A Figura 2.2 ilustra um experimento realizado em 1996 (OLUKOTUN, 1996) que mostra as vantagens de usar uma arquitetura *multi-core* em relação a uma arquitetura superescalar. O experimento mostra que a área ocupada pelas duas soluções é a mesma, e as principais características de cada arquitetura são as seguintes:

- Ambas são baseadas em arquiteturas do processador MIPS R10000.
- A arquitetura (a) é um superescalar de seis vias de execução.
- A arquitetura (b) é *multi-core*, possui quatro núcleos, sendo que cada um possui *pipeline* superescalar de duas vias.

Em resumo, os resultados mostram que uma arquitetura superescalar possui desempenho melhor se a carga de trabalho possuir alto paralelismo no nível de instrução. Por outro lado, a arquitetura *multi-core* possui melhor desempenho se a carga de trabalho tiver um alto paralelismo no nível de *thread*. Este experimento foi o início para o surgimento do processador Niagara (Ultrasparc T1) (KONGETIRA, 2005), que possui oito núcleos escalares com suporte IMT cada um. Este processador possui alta vazão de *threads* e baixa vazão de instruções. Portanto, uma arquitetura como esta não é adequada para um computador pessoal, mas adequada para um sistema de aplicações *web* e banco de dados, onde a carga de trabalho possui alto paralelismo de *threads*.

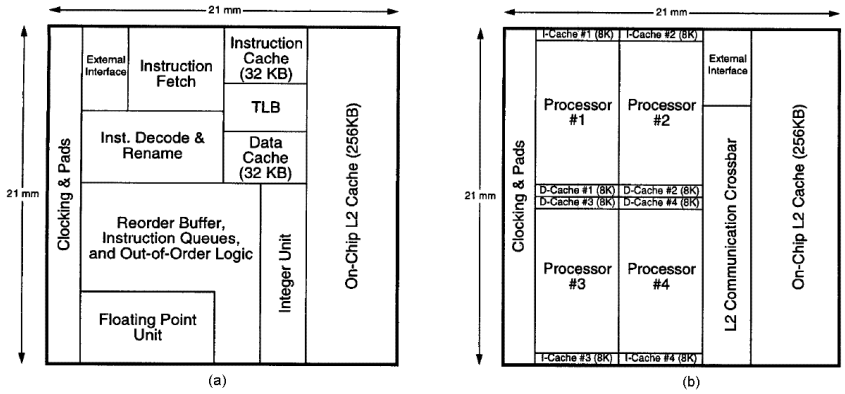


Figura 2.2: Comparação entre arquiteturas. (a) superescalar, (b) multi-core (OLUKOTUN, 1996)

Pode ser visto na Figura 2.2b, que para quatro núcleos uma chave *crossbar* é suficiente para prover a comunicação interna. A mesma alternativa foi adotada para o processador Niagara, que possui oito núcleos. No entanto, para a nova geração de processadores *many-core*, uma única chave *crossbar* possui limites físicos que impedem o aumento da quantidade de núcleos interconectados a ela. A Figura 2.3 mostra que no projeto Terascale da Intel (INTEL, 2006) (INTEL, 2007), está em desenvolvimento um processador *many-core* com oitenta núcleos e uma *Network-on-Chip*. Em realce, cada núcleo possui um roteador e, portanto, a comunicação interna é por troca de mensagens. A Subseção 2.1.4 descreve os principais detalhes das *Networks-on-Chip*.

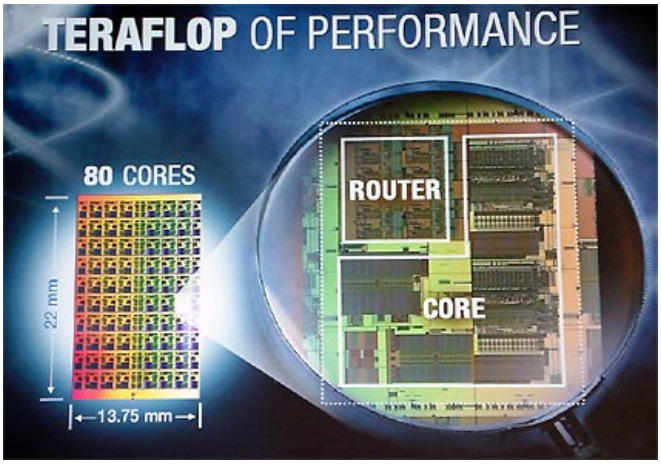


Figura 2.3: Arquiteturas Many-Core. Teraflops de desempenho (INTEL, 2006)

As principais vantagens dos processadores *many-core*, além do aumento do desempenho, podem ser ilustrados pela Figura 2.4, tais como:

- (a) Ativação somente dos núcleos necessários.
- (b) Redistribuição de carga de trabalho em função de altas temperaturas atingidas pelos núcleos.
- (c) Núcleos reservas para substituição de núcleos com problemas.
- (d) Expansão da funcionalidade com diversos dispositivos dentro do *chip*.

Outro detalhe que pode ser percebido pela Figura 2.4 é a necessidade de uma rede de comunicação eficiente e que possibilite as vantagens apresentadas. Na subseção seguinte, algumas abordagens são descritas.

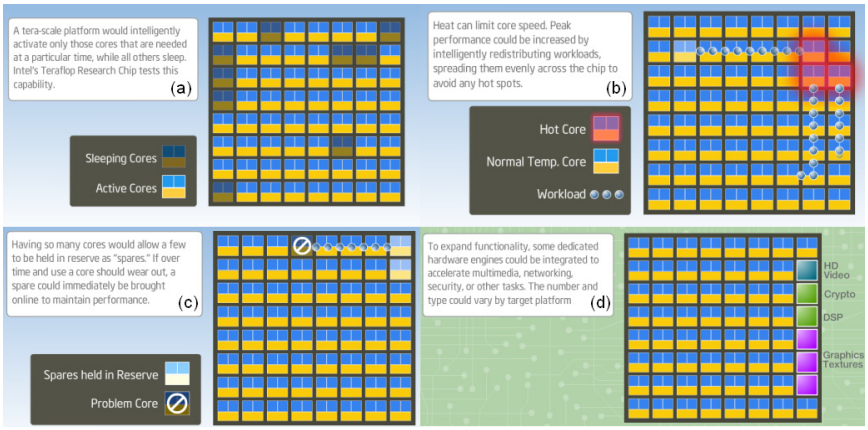


Figura 2.4: Justificativas do projeto Terascale. (a) utilização inteligente de núcleos, (b) redistribuição de carga de trabalho, (c) núcleos reservas, (d) integração de dispositivos dedicados (INTEL, 2007)

2.1.2 Características e Classificações das Redes de Interconexão

As redes de interconexão em *chip* são extremamente importantes para o desempenho final de um processador *multi-core* ou *many-core*. Estas redes de comunicação são responsáveis pelo tráfego de dados ou pacotes e, portanto, podem representar um gargalo ou o grande diferencial de desempenho entre propostas de arquitetura de processadores.

As Classificações das redes de interconexão (ANDERSON, 1975) (AHMADI, 1989) (DUATO, 2002) podem ser aplicadas tanto no contexto *on-chip* quanto *off-chip*. Nesta seção não será feita qualquer distinção do contexto da rede. Isto significa que apesar de todos os conceitos serem apresentados para o contexto *on-chip*, estes podem ser empregados também no contexto *off-chip*.

Um dos primeiros critérios no projeto de uma rede de interconexão é a topologia. A Figura 2.5 ilustra alguns exemplos: (a) estrela, (b) árvore, (c) *mesh*, (d) *torus*, (e) hipercubo, e (f) *pipeline*. Na definição de uma topologia, duas métricas devem ser observadas (DUNCAN, 1990) (DE ROSE, 2003): conectividade e adequação ao domínio de aplicação. Estas métricas podem indicar inicialmente a capacidade da topologia responder ao tráfego de pacotes ou dados, apresentando mais ou menos gargalos, contenções ou atrasos.

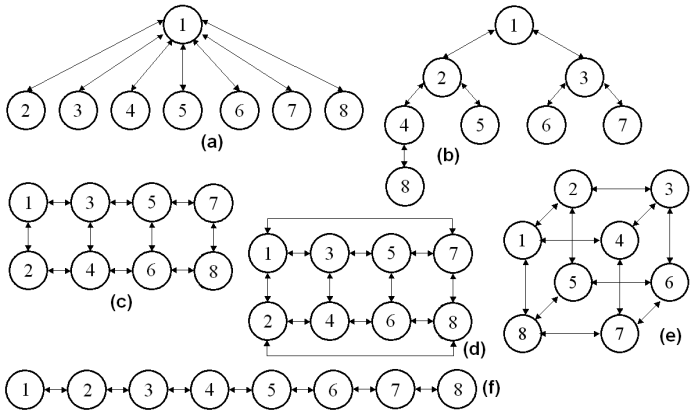


Figura 2.5: Topologias de redes de interconexão. (a) estrela, (b) árvore, (c) *mesh*, (d) *torus*, (e) hipercubo, e (f) pipeline

A conectividade pode ser explicada através de teoria de grafos (CALVERT, 1997). Esta métrica indica o quanto um vértice (nó) do grafo (topologia) está conectado aos demais. Uma topologia completa, onde todos os nós estão conectados entre si, possui nós com conectividade máxima e isto pode representar uma quantidade menor de saltos entre origem e destino de uma transmissão. Esta quantidade de saltos é outra métrica importante, pois pode representar um caminho mais curto para o caminho de dados. No entanto, para atingir um melhor desempenho, é necessário que haja pouco tráfego e poucas contenções no caminho. Um caminho curto com alta contenção pode não ser a melhor rota para a comunicação. Questões relativas às técnicas de roteamento são descritas na Subseção 2.1.4.

A adequação da topologia ao domínio de aplicação (LEMEIRE, 2008) representa o quanto uma topologia mapeia um padrão de comunicação definido por uma aplicação (programa) em execução. Este mapeamento pode significar a redução de contenções através de um projeto de topologia que diminua a concorrência por nós intermediários (gargalos). Em (DUNCAN, 1990) são descritas algumas características de aplicações e as melhores topologias para cada uma delas. Existem trabalhos correlatos apresentados na Seção 2.4 que abordam o projeto de NoCs com foco em padrões de comunicação.

As redes de interconexões podem ser classificadas segundo alguns critérios (ANDERSON, 1975) (AHMADI, 1989) (DUATO, 2002). Nesta seção é dada ênfase a dois tipos: Forma de comunicação e tipo de chaveamento. Nestes dois tipos as topologias são usadas como bases para diferenciar as estruturas e através delas são feitas as classificações. A Tabela 2.1 ilustra a classificação segundo forma de comunicação. Neste caso são definidas três estratégias descritas a seguir:

- A Estratégia de Transferência pode ser definida por direta ou indireta. As redes diretas são compostas por um conjunto de nós em que cada um está diretamente conectado a outro. Nas redes indiretas, os nós possuem conexões indiretas, ou seja, um conjunto de nós é responsável apenas por interconectar outros nós. Sendo assim, nas redes diretas cada nó possui, conectado a ele, um elemento de processamento e nas redes indiretas os nós intermediários não possuem este elemento de processamento. Segundo a Tabela 2.1 apenas as topologias Estrela e Árvore seriam indiretas. Neste caso, o nó central da Estrela é apenas intermediário e não possui um elemento de processamento. No caso da Árvore existe um subconjunto de nós responsáveis por interconectar os nós folhas, que possuem elementos de processamento interconectados. Nos outros exemplos de topologias, haveria um elemento de processamento interconectado a cada um dos nós, por isso estes exemplos são de redes diretas.

- O Método de Controle define o tipo de roteamento utilizado. Em todas as redes diretas apresentadas, o roteamento é descentralizado, o pacote é roteado a cada salto durante o caminho. No caso das redes indiretas, os nós intermediários da Árvore também descentralizam o roteamento. A exceção é a topologia Estrela, que possui um nó central responsável pelo roteamento.

- A Estrutura do Caminho define se o pacote possui um caminho dedicado ou compartilhado entre origem e destino. Em todos os exemplos os caminhos são dedicados. Um exemplo que pode ser ilustrado como exceção seria uma topologia Estrela onde o nó central não fosse baseado em uma matriz de conexão (e.g., chave *crossbar*), mas em um barramento. O barramento é uma

estrutura compartilhada que reduziria a largura de banda e escalabilidade da rede.

Tabela 2.1: Classificação segundo forma de comunicação

Topologia	Estratégia de Transferência	Método de Controle	Estrutura do Caminho
Estrela	Indireta	Roteamento centralizado	Dedicado
Árvore	Indireta	Roteamento Descentralizado	Dedicado
Mesh	Direta	Roteamento Descentralizado	Dedicado
Torus	Direta	Roteamento Descentralizado	Dedicado
Hipercubo	Direta	Roteamento Descentralizado	Dedicado
Pipeline	Direta	Roteamento Descentralizado	Dedicado

Outro tipo de classificação (DE ROSE, 2003) é segundo o tipo de chaveamento, que pode ser estático ou dinâmico. No chaveamento estático a interconexão entre os nós é fixa, e por isso é também chamada de interconexão espacial. No caso do chaveamento dinâmico, a interconexão é realizada segundo demanda de conexão, e por isso também é chamada de interconexão temporal. A Figura 2.6 ilustra alguns exemplos destes dois tipos de chaveamento.

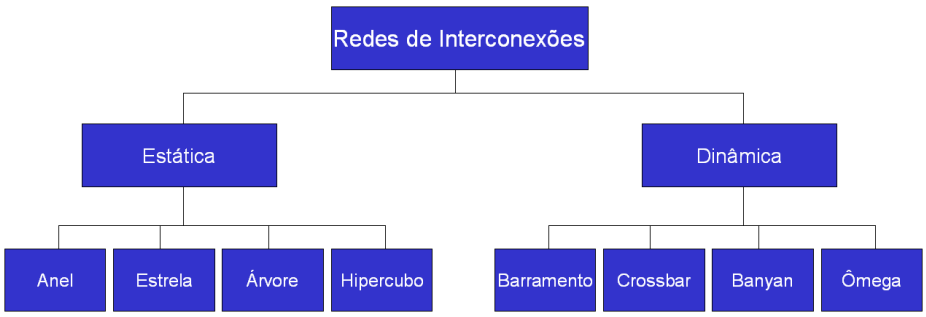


Figura 2.6: Classificação segundo tipo de chaveamento

Os exemplos do chaveamento estático são baseados em topologias já apresentadas nesta seção, mas com características de conexão fixa ponto-a-ponto. Os exemplos de chaveamento dinâmico podem ser utilizados de forma independente, como será visto nas Figuras 2.7 e 2.8, mas também podem ser utilizados em nós de roteamento das topologias já apresentadas.

Dois dos principais tipos de interconexões dinâmicas (barramento e chave *crossbar*) são apresentados na Figura 2.7. Todos os dois tipos possuem problemas de escalabilidade se ambas são usadas para suportar o aumento do número de nós interconectados a elas. Este problema de escalabilidade está relacionado aos problemas e limites dos fios já descritos. No entanto, ambas possuem um bom desempenho em se tratando de uma comunicação *broadcast* ou *multicast*. A característica de múltiplos barramentos interconectados por chaves faz com que a chave *crossbar* tenha um desempenho muito melhor do que um barramento, já que é possível ter N (número de nós) comunicações distintas simultâneas. Ao contrário, o barramento é um meio compartilhado e por isso suporta apenas uma comunicação por vez. Comunicações distintas devem esperar *slots* de tempo para acesso ao barramento, reduzindo a largura de banda disponível. Esta é uma característica das redes chamadas de bloqueantes. A vantagem do barramento é o menor custo em relação à chave *crossbar*.

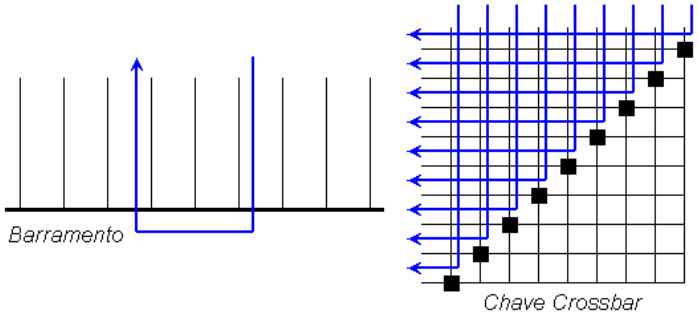


Figura 2.7: Interconexões dinâmicas: barramento e chave crossbar

A Figura 2.8 ilustra redes multinível baseadas no chaveamento dinâmico. Neste caso, existem pequenas matrizes de conexão (e.g., chaves *crossbar* 2x2) interconectadas em vários níveis consecutivos. As chaves *crossbar* não são bloqueantes, mas a interligação delas nas redes Banyan e Omega torna estas redes bloqueantes, já que existe apenas um caminho entre entrada e saída. Por outro lado, um único caminho entre entrada e saída faz com que o roteamento seja eficiente, podendo ser feito de forma descentralizada.

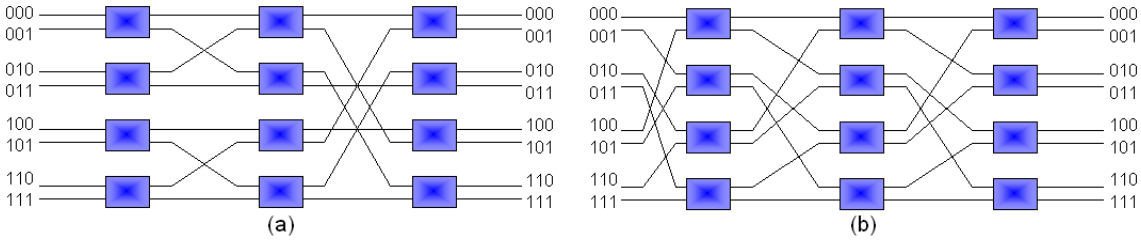


Figura 2.8: Interconexões dinâmicas multinível: (a) Banyan e (b) Omega

2.1.3 Principais Critérios para Avaliação das Redes de Interconexão

Os conceitos e classificações apresentados são necessários para o projeto de uma NoC. Nesse sentido, as métricas para avaliação de uma rede-em-*chip* são as mesmas de uma estrutura de interconexão já apresentada. Sendo assim, nesta subseção são apresentadas as principais métricas (JAIN, 1991) (HWANG, 1998) (DE ROSE, 2003) que devem ser consideradas para projeto e avaliação de uma rede de comunicação em *chip*, conforme a seguir.

Escalabilidade: Capacidade de uma rede de interconexão interligar uma maior quantidade de elementos de processamento ou periféricos. Pode ser medido através de, por exemplo, limites impostos pela resistência do fio ou degradação/amplitude do sinal.

Desempenho: Capacidade que a rede possui para entregar um determinado dado ou pacote em tempo adequado. As principais métricas são: distância a ser percorrida (saltos), latência do meio (fio e componentes) e largura de banda (tamanho dos dados transmitidos por unidade de tempo). Estas métricas podem ser influenciadas por contenções em elementos de rede intermediários, ineficiência de um algoritmo de roteamento e tamanho do dado ou pacote transmitido. A métrica mais utilizada para avaliar desempenho é o tempo final de transmissão. Afinal, independente do grupo de métricas utilizado para comparação entre redes, ou da influência maior ou não do meio, o que vale é o quanto uma rede é mais rápida (menor tempo) do que outra. Portanto, é possível relacionar algumas das principais métricas conforme a Equação 1:

$$T = L + \frac{P}{B} \qquad \text{Equação 1}$$

T = Tempo de transmissão, L = Latência do meio,
P = Tamanho do pacote, B = Largura de Banda.

Custo: Em função do aumento do tamanho para suportar escalas diferentes ou novas interligações, além do aumento da largura de banda, o custo em potência consumida ou área aumenta proporcionalmente. As principais métricas neste caso seriam: potência consumida, área ocupada ou componentes utilizados.

Confiabilidade: Uma rede é confiável se existe uma garantia maior na entrega dos dados ou pacotes. Redundância de *links* de transmissão é um exemplo para aumentar a tolerância a falhas. A quantidade de *links* redundantes ou a porcentagem de acerto na entrega de pacotes são possíveis métricas para avaliação de confiabilidade.

Funcionalidade: Além de transmitir dados ou pacotes, funcionalidade é a capacidade de uma rede desempenhar outras funções. Estas funções podem ser exemplificadas como a capacidade de armazenar pacotes em *buffers*, reordenar pacotes, rotear pacotes, ou se adaptar a um determinado padrão de comunicação. Uma possível métrica para avaliar a funcionalidade está na quantidade de funções que cada rede é capaz de suportar.

2.1.4 Networks-on-Chips (NoCs)

De acordo com os limites de escalabilidade impostos pelos fios, as tradicionais soluções de interconexão largamente utilizadas em arquiteturas *multi-core*, tais como barramento e chave *crossbar*, são impraticáveis para arquiteturas *many-core*. A solução que vem sendo estudada e proposta através de várias pesquisas é a *Network-on-Chip* (BENINI, 2002) (BENINI, 2005) (BJERREGAARD, 2006) (OGRAS, 2007b). A Figura 2.9 ilustra a evolução focada no problema relativo aos fios até uma arquitetura inicial de NoC, como forma de sumarizar o obstáculo no uso de fios globais para interconexão de grande número de núcleos, conforme descrito nas subseções anteriores e capítulo de Introdução.

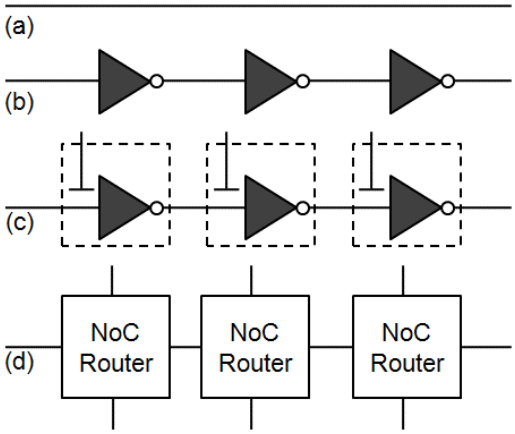


Figura 2.9: Evolução da interconexão global para NoC. (a) fio longo dominado pela resistência, (b) adição de repetidores ou buffers, (c) repetidores se tornam latches, e (d) latches evoluem para roteadores de NoC (CIDON, 2009)

Portanto, nesta subseção é dada uma ênfase às NoCs apresentando as principais abordagens relativas às arquiteturas da rede ou roteador, topologias e protocolos, além

de uma análise da viabilidade em comparação com as alternativas de interconexões tradicionais.

2.1.4.1Arquiteturas de NoCs

Uma *Network-on-Chip* é composta por três elementos básicos: roteador, interface e *links* de comunicação. O roteador é o elemento principal responsável pela interconexão da rede, pela definição de rotas, pelo controle de fluxo, qualidade de serviço e, portanto, pela garantia de entrega do pacote de dados. Por se tratar de uma rede de comunicação composta por roteadores, o mecanismo de entrega de dados é através de passagem de mensagem ou pacotes de rede. Interligando os roteadores existem os *links* de comunicação. Estes *links* são os fios responsáveis pela existência do caminho a ser percorrido pelos pacotes. A forma como os roteadores estão interconectados pelos *links* dá origem à topologia da rede. A subseção seguinte apresenta detalhes e comparações entre topologias propostas para NoCs. O último elemento que compõe uma NoC é a interface de rede. Esta interface também é chamada de adaptador ou *wrapper*, sendo necessária para garantir a correta comunicação entre a rede (roteadores da NoC) e os núcleos ou periféricos que estão interconectados. Esta interface garante que haja uma correta comunicação entre protocolos diferentes (NoC, núcleo, memória, etc.).

A Figura 2.10 ilustra um exemplo de NoC baseada na topologia *mesh*. Neste caso, a NoC é uma *mesh* 3x3 que interconecta três núcleos de processamento através de três interfaces de rede. Estas interfaces estão interconectadas, cada uma, a um roteador diferente da NoC e, portanto, cada roteador é específico para um núcleo de processamento. Os demais roteadores da NoC nesta figura não estão interconectados a elementos externos, mas poderiam ser memórias, ou qualquer *hardware* dedicado.

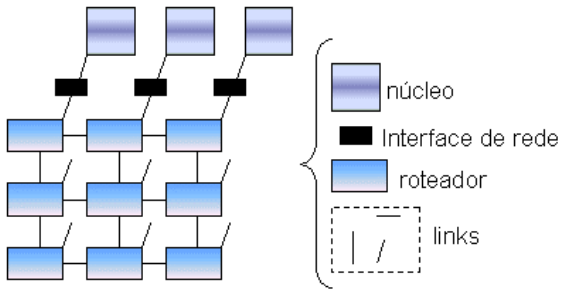


Figura 2.10: Arquitetura básica de uma NoC com topologia mesh (FREITAS, 2009a)

Um *chip* composto por uma NoC, núcleos de processamento, memórias e outros *hardwares* dedicados representam um sistema de computação. Portanto, para este caso existe um termo muito comum conhecido como Sistema-em-*Chip* ou *System-on-Chip* (SoC). Os projetos de arquiteturas de processadores *many-core* caminham para uma heterogeneidade de componentes em *chip* que os aproximam desta classificação.

A Figura 2.11 ilustra uma arquitetura típica de roteador para uma NoC *mesh* apresentada na Figura 2.10. Estes roteadores trabalham com roteamento XY e, portanto, possuem quatro saídas para norte, sul, leste e oeste. Além disso, é necessária uma saída para o núcleo de processamento através da interface de rede. Para a interconexão entre as saídas, o roteador se baseia em uma arquitetura simples de chave *crossbar*, neste exemplo, 5x5, normalmente sem nenhum adicional de complexidade, como, por exemplo, suporte a *broadcast*. Para definição das conexões que devem ser realizadas, é necessário um Árbitro. A função principal deste mecanismo de arbitragem está na solução de conflitos na utilização da chave *crossbar* e, por conseqüência, na liberação

de pacotes dos *buffers* de entrada. Neste exemplo de roteador, os *buffers* estão presentes apenas nas entradas. É importante ressaltar, que a arquitetura deste roteador ilustra um exemplo de circuito dedicado e não programável.

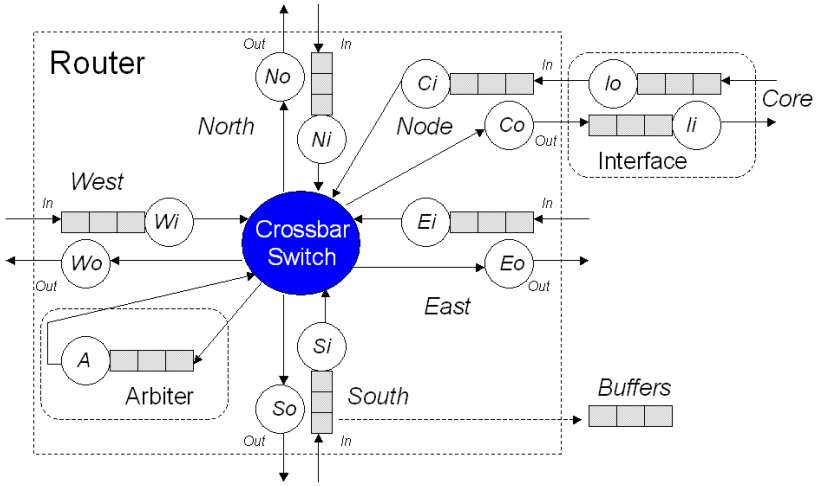


Figura 2.11: Arquitetura básica de um roteador de NoC para topologia mesh (FREITAS, 2009a)

O impacto da utilização de *buffers* pode estar relacionado ao tamanho, quanto maior a profundidade do *buffer*, maior o tamanho da NoC, ou pelos seguintes motivos relacionados à chave *crossbar* (AHMADI, 1989):

- *Buffers* de entrada: As técnicas de arbitragem são relativamente simples, possuem uma melhor relação de área e potência, além de proporcionar um melhor desempenho para a chave *crossbar*.
- *Buffers* de saída: Em função de N entradas conectadas a cada um dos *buffers* de saída, a chave *crossbar* precisa ser N vezes mais rápida. A adoção de *buffers* de saída não é a mais adequada para alto desempenho. No entanto, existem vantagens em se tratando da eliminação do bloqueio de pacotes que não receberam permissão de envio, porque o primeiro pacote da fila ainda não teve liberação de uma determinada saída. Este problema é conhecido como *head of the line blocking* e pode acontecer nas soluções com *buffers* de entrada.
- *Buffers* de *crosspoint*: Cada ponto de conexão da chave *crossbar* possui um *buffer*. É utilizada a técnica de roteamento chamada de *self-routing*. Neste caso, em cada *crosspoint* seria necessário, além do *buffer*, um decodificador para decisão de envio ou não do pacote. Esta solução aumenta o tamanho e a potência consumida da chave *crossbar*.

2.1.4.2Topologias de NoCs

A definição de uma topologia de NoC, tal como de um roteador, está relacionada à carga de trabalho que será executada. A Tabela 2.2 ilustra as principais NoCs e suas respectivas topologias. Nesta seção é feita uma análise das principais características de cada uma destas topologias.

Tabela 2.2: Exemplos de topologias de NoCs

Network-on-Chip	Topologia
SPIN (ANDRIAHANTENAINA, 2003)	Árvore gorda.
ASoC (LIANG, 2000)	*Mesh* 2D.
Dally (DALLY, 2001)	*Torus* 2D.
Nostrum (MILLBERG, 2004)	*Mesh* 2D.
Sgroi (SGROI, 2001)	*Mesh* 2D.
Octagon (KARIM, 2002)	Anel Cordal.
Marescaux (MARESCAUX, 2003)	*Torus* 2D.
AEtheral (RIJPKEMA, 2003)	*Mesh* 2D.
Eclipse (FORSELL, 2002)	*Mesh* 2D hierárquica.
Proteo (SIGÜENZA-TORTOSA, 2002)	Anel bi-direcional.
Hermes (MORAES, 2004)	*Mesh* 2D.
SoCIN (ZEFERINO, 2003)	*Mesh/Torus* 2D.
SoCBUS (WIKLUND, 2003)	*Mesh* 2D.
QNoC (BOLOTIN, 2004)	*Mesh* 2D.
T-SoC (GRECU, 2004)	Árvore gorda.
Bouhraoua (BOUHRAOUA, 2006)	Árvore gorda.
BENoC (MANEVICH, 2009)	Barramento global integrado com *mesh* 2D.
BiNoC (LAN, 2009)	*Mesh* 2D com canais bidirecionais.
CHNoC (LENG, 2005)	*Clusters* com topologia irregular.
GigaNoC (NIEMANN, 2005)	*Clusters* com topologia regular *mesh* 2D.
IPNoSys (FERNANDES, 2008)	Programável. Roteadores ativos. *Mesh*, *torus* e *butterfly 2D*.
PNoC (HILTON, 2006)	Reconfigurável. Baseado em FPGA.
Bartic (BARTIC, 2005)	Reconfigurável. Baseado em FPGA.
CoNoChi (PIONTECK, 2008)	Reconfigurável. Baseado em FPGA.
Wireless NoC (WANG, 2007)	Sem fio irregular.
RECONNECT (JOSEPH, 2008)	*Honeycomb mesh*. Topologia fixa usada para suportar o conceito *Polymorphic ASIC*.
ReNoC (STENSGAARD, 2008)	Reconfigurável. Interface entre os roteadores e links para alteração/ativação de topologias. Usa o conceito *Polymorphic ASIC*.
Polymorphic NoC (MERCALDI-KIM, 2008)	Reconfigurável. Conjunto de Chaves *crossbar* interconectadas para reconfiguração de topologias.
MCNoC (FREITAS, 2008c)	Reconfigurável e Programável. Segundo nível de reconfiguração baseado em chaves *crossbar* para *clusters* de núcleos.

Através da Tabela 2.2 é possível citar três tipos de topologias: fixas, sem fio e reconfiguráveis. As topologias fixas são alternativas clássicas de adoção de uma determinada forma de interconexão que privilegie um comportamento específico de uma determinada carga de trabalho (BERTOZZI, 2005) (HO, 2006). Topologias sem fio são alternativas recentes para eliminar as limitações do fio no projeto de NoCs através de uma tecnologia chamada de *Radio-on-Chip* (CHANG, 2001). Por fim, as topologias

reconfiguráveis utilizam plataformas programáveis para que sejam realizadas adaptações na forma de interligações em função de mudanças no padrão de comunicação das cargas de trabalho. Espera-se através da reconfiguração um aumento na flexibilidade de topologias da NoC focando no ganho de desempenho em relação a uma solução de topologia fixa.

As topologias fixas descritas pela Tabela 2.2 podem ser analisadas segundo alguns critérios (STOJMENOVIC, 1997) (HWANG, 1998) (DE ROSE, 2003), conforme Tabela 2.3:

Tabela 2.3: Características de algumas topologias fixas

	Número de ligações	**Grau do nó**	**Diâmetro**
Árvore Binária*	$2^{h+1} - 2$	3	2h
Anel Cordal	2n	4	2
Anel Bi-direcional	n	2	n/2
Mesh 2D	2n-2r	4	2(r-1)
Torus 2D	2n	4	$n^{1/2}$-1
Honeycomb mesh 2D	g *n/2	3	$1{,}63\sqrt{n}$
Totalmente conectada**	$(n^2 – n) / 2$	n - 1	1

*Árvore Gorda é uma alternativa tolerante a falhas baseada na replicação de conexões ou uso de conexões com maior vazão nas ligações perto da raiz.
**Totalmente conectada serve como referência para análise.
h = altura, n = número de nós, r = linhas, g = grau do nó.

A topologia totalmente conectada é apresentada na Tabela 2.3 como referência e alternativa para alcançar melhor desempenho em uma rede. Neste caso, as características desta topologia mostram que ela possui um alto número de ligações, um alto grau do nó, e um baixo diâmetro (maior distância entre dois nós). Estas são características que descrevem um baixo número de saltos e alta redundância de ligações. Quanto menor o número de ligações e grau do nó, maior é o diâmetro da rede. O grande problema de uma topologia totalmente conectada é o custo, tal como, problemas para roteamento dos fios.

As topologias mais encontradas na literatura são baseadas em *Mesh* e *Torus*. A principal característica está relacionada à capacidade de suportar aplicações cujos problemas podem ser particionados (e.g., operações com matrizes e processamento de imagens). As topologias *Honeycomb mesh* ou *mesh* hexagonal são consideradas da mesma família e possuem desempenho similar. A vantagem de uma topologia *honeycomb* em relação à *mesh* está na diminuição do custo em função de uma menor área ou quantidade de *links*, além de facilitar o mapeamento de aplicações.

A topologia Anel possui o menor custo entre as apresentadas, mas em contrapartida possui um diâmetro que cresce de forma linear em função do número de nós. Uma alternativa que mantêm um baixo custo é a topologia Anel Cordal que possui caminhos alternativos, aumentado o grau do nó e diminuindo o diâmetro. Esta alternativa é mais tolerante a falhas do que a versão original (Anel) e, portanto, mais confiável.

A topologia Árvore Binária é uma solução interessante para aplicações baseadas em algoritmos de divisão e conquista. O diâmetro cresce de forma linear em relação à altura, e a confiabilidade é relativamente baixa, já que a perda de um nó pode separar a

topologia em duas partes. Outro problema está relacionado ao nó raiz que é um gargalo entre as subárvores da esquerda e direita. A solução para este problema está na Árvore Gorda. Esta solução é mais tolerante a falhas, pois possui um maior número de ligações entre os nós próximos da raiz. Outra solução é aumentar a largura de banda destas ligações, reduzindo as contenções de comunicação do efeito concentrador / gargalo da raiz da árvore.

Encontrar uma topologia, que atenda os melhores requisitos de desempenho, escalabilidade, confiabilidade e custo não é tão trivial (KREUTZ, 2005). No projeto Terascale da Intel (Figura 2.12) duas topologias estão em teste: anéis interconectados e *mesh*. Devido ao grande número de diferentes domínios de aplicação ou padrões de comunicação, algumas propostas apontam para o uso de conceitos de reconfiguração para aumentar a adaptabilidade da rede (BJERREGAARD, 2006) (OGRAS, 2007a). Na Seção 2.2 é feita uma relação entre os padrões de comunicação e possíveis topologias de NoCs. No mesmo caminho, a Seção 2.3 apresenta conceitos de computação reconfigurável que podem ser utilizados para aumentar a flexibilidade e adaptabilidade das topologias de NoCs.

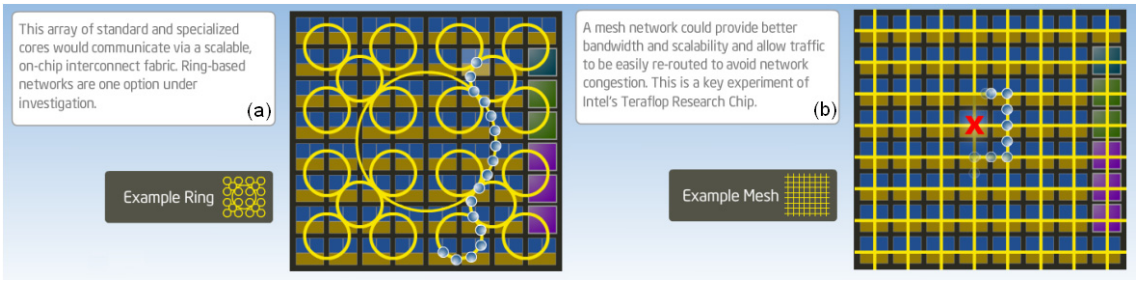


Figura 2.12: Topologias do projeto Terascale. (a) anéis interconectados, (b) mesh (INTEL, 2007)

2.1.4.3Tipos de Protocolos

Políticas e estratégias de transporte de dados em uma NoC são de responsabilidade dos protocolos. Os protocolos descrevem as principais características de funcionamento da rede. Neste sentido, os protocolos são capazes de garantir a entrega dos dados, a confiabilidade da rede, a melhor rota, além do melhor desempenho, entre outras características. Os principais aspectos dos protocolos para NoCs (BJERREGAARD, 2006) são apresentados a seguir:

- Chaveamento por circuito: Existe uma rota dedicada para o caminho do dado. Esta rota (circuito) permanece reservada até que a transmissão do dado acabe.
- Chaveamento por pacote: Pacotes trafegam pela rede e o caminho é definido através de roteamento por salto. Não há reserva de caminho e, portanto, vários pacotes podem compartilhar o mesmo meio.
- Orientada a conexão: Existe uma conexão lógica entre origem e destino. Portanto, a transmissão só inicia após a confirmação de um estado de pronto entre o transmissor e receptor.
- Não orientada a conexão: Não existe uma conexão lógica. A comunicação é estabelecida sem o acordo entre origem e destino.

- Roteamento determinístico: A definição da rota é feita pela origem e destino. Um exemplo utilizado pelas topologias *mesh* e *torus* é o roteamento XY. Primeiro são percorridas as linhas e depois as colunas até o destino.
- Roteamento adaptativo: A definição da rota é feita a cada salto de roteamento. Neste caso, um roteador intermediário pode alterar o caminho que o pacote seguirá.
- Roteamento mínimo ou não mínimo: O roteamento é mínimo se sempre é feita a escolha do menor caminho entre origem e destino.
- Atraso versus perda: No caso de um protocolo baseado em atraso, o pior caso é um pacote atrasado. No caso de um protocolo baseado em perda, é possível excluir um pacote da rede, portanto, uma retransmissão seria o pior caso.
- Controle central ou distribuído: Pelo controle central, o roteamento é feito de forma global. No caso do distribuído, cada roteador define a melhor rota para o pacote.

Existem três técnicas de encaminhamento de pacotes conforme descrição a seguir:

- *Store-and-Forward*: Todo o pacote é armazenado em *buffers* do roteador para que o cabeçalho seja analisado. Em seguida o pacote é encaminhado.
- *Wormhole*: Enquanto o pacote é recebido seu cabeçalho é analisado. Definida a rota, todo o pacote é encaminhado para o nó seguinte sem a necessidade de armazenamento temporário em *buffers*.
- *Virtual cut-through*: Segue o mesmo mecanismo do *wormhole*, mas o nó antes de encaminhar espera uma confirmação do próximo nó destino para envio do pacote.

Os mecanismos para controle de fluxo são responsáveis por garantir o funcionamento da rede. Entre os principais benefícios estão:

- Garantir que pacotes não sejam descartados.
- Evitar retransmissões de pacotes.
- Diminuir o congestionamento da rede ou contenções nos roteadores.
- Otimizar o uso de recursos da rede.
    - *Buffers* menores.
    - Menor número de transmissão de pacotes.
    - Menor uso dos *buffers* e roteadores.
    - Menor consumo de potência e energia.
    - Melhor desempenho da rede.

Um dos conceitos básicos do controle de fluxo é ajustar a taxa de saída de dados de um transmissor para um receptor. Três abordagens clássicas são descritas a seguir:

- *Handshake*: também chamado de “aperto de mão”, consiste em um acordo entre transmissor e receptor através de linhas adicionais de controle para liberação do envio de pacotes. Transmissor envia sinal de solicitação e receptor envia sinal de *buffer* livre.

- Créditos: o receptor envia a quantidade de espaço livre no *buffer* de entrada para que o transmissor saiba quantos créditos estão disponíveis para envio de pacotes.
- Canais virtuais: técnica para eliminar o problema *head of the line blocking* já descrito na Subseção 2.1.4.1, em que o primeiro pacote bloqueia os demais da fila. Através de canais virtuais é possível criar diversas saídas do *buffer* de entrada, como se este tivesse várias pequenas filas.

Apesar do projeto de protocolos visar a entrega dos dados, existem problemas que podem ocorrer e impedir que pacotes cheguem ao destino. Estes problemas são conhecidos como *deadlock*, *livelock* e *starvation*, e devem fazer parte das preocupações de um projetista de NoC.

- *Deadlock*: O *deadlock* é a representação de uma dependência cíclica. Neste caso, um pacote não consegue progredir e fica restrito a um subconjunto de estados ou roteadores.
- *Livelock*: O *livelock* é a representação de uma contínua retransmissão do pacote sem atingir o nó destino. Comum em protocolos de roteamento.
- *Starvation*: O *starvation* é a representação da não alocação de um recurso devido a postergação indefinida de acesso ao mesmo. Comum em protocolos de arbitragem.

2.1.4.4Análise da Viabilidade das NoCs

A Tabela 2.4 apresenta as vantagens e desvantagens da adoção de uma NoC em relação às soluções tradicionais (barramento e chave *crossbar*) mais utilizados atualmente nos processadores *multi-core*.

Apesar das vantagens das soluções tradicionais no que diz respeito à simplicidade, compatibilidade e latência, os limites físicos impostos pelo fio, questões relacionadas à escalabilidade e largura de banda apontam para a NoC como a melhor alternativa para futuras gerações de processadores *many-core*. Seguindo esta tendência, nos capítulos de metodologia, proposta e resultados, é apresentada a proposta de NoC que une algumas características das chaves *crossbar* para arquiteturas de roteadores programáveis com foco no aumento do desempenho em *chips many-core.*

Tabela 2.4: Análise comparativa para uso de NoCs (adaptado de BJERREGAARD, 2006)

Tipo de Interconexão		**Prós (+) e Contras (-)**	
Barramento	Fio	O aumento do fio aumenta a resistência degradando o desempenho.	-
Chave *Crossbar*		O aumento do fio aumenta a resistência degradando o desempenho.	-
Network-on-Chip		Os fios são ponto-a-ponto entre roteadores e o desempenho não degrada em função do aumento de nós.	+
Barramento	Árbitro	O árbitro é um gargalo à medida que o número de nós aumenta.	-
Chave *Crossbar*		O árbitro pode ser centralizado ou descentralizado e não é o fator principal para degradação do desempenho em função do aumento dos nós.	+-
Network-on-Chip		As decisões de roteamento são distribuídas e não representam um gargalo.	+

Tipo de Interconexão		Prós (+) e Contras (-)	
Barramento	Largura de banda	A largura de banda é limitada e compartilhada por todos os nós.	-
Chave *Crossbar*		Cada interconexão é independente e a largura de banda de comunicação por conexão não é afetada pelas demais.	+
Network-on-Chip		A largura de banda não é afetada pelo aumento da rede.	+
Barramento	Latência	Latência é afetada pelo fio.	+
Chave *Crossbar*		Latência é afetada pelo fio.	+
Network-on-Chip		Latência é afetada pelas contenções em roteadores	-
Barramento	Compatibilidade	Em sua maioria são compatíveis com qualquer IP (*Intelectual Property*) incluindo os softwares.	+
Chave *Crossbar*		Em sua maioria são compatíveis com qualquer IP (*Intelectual Property*) incluindo os softwares.	+
Network-on-Chip		São necessários adaptadores (*wrappers*) entre os IPs, e os softwares precisam de sincronização em sistemas *multi-core*.	-
Barramento	Complexidade	Conceitos simples e bem compreendidos.	+
Chave *Crossbar*		Conceitos simples e bem compreendidos.	+
Network-on-Chip		Projetistas precisam de uma reeducação em função dos novos conceitos.	-

2.2 Padrões de Comunicação

Um projeto de processador deve ter como objetivo seu contexto de utilização. Portanto, devem ser levadas em consideração quais as cargas de trabalho que serão executadas. O projeto de um processador *many-core* possui como característica forte a escolha de uma rede de comunicação interna para suportar a intensa transmissão e troca de pacotes. Sendo assim, o estudo e análise da carga de trabalho (JAIN, 1991) são essenciais para o projeto da rede-em-*chip*, já que esta pode ser o diferencial de desempenho ou gargalo de funcionamento do processador.

Conforme já foi iniciado na subseção que trata especificamente de topologias, algumas delas possuem formatos que atendem ou mapeiam determinados domínios de aplicação. Alguns estudos apontam a necessidade de uma topologia específica para aumentar o desempenho da rede. Outros já apontam a necessidade da rede ser adaptável para atender as diferentes necessidades ou aplicações. Na Seção 2.4 (Trabalhos Correlatos) é feita uma análise de diferentes propostas para *Networks-on-Chip*.

De acordo com (DUNCAN, 1990) algumas topologias fixas possuem uma melhor capacidade de mapear programas. Portanto, é possível fazer as seguintes relações:

- Topologia Árvore: programas de divisão e conquista.
- Topologia Hipercubo: programas científicos 3D.
- Topologias *Mesh* e *Torus*: operações de matrizes e processamento de imagem.

- Topologias *Pipeline* e Anel: processamento de protocolos de rede.
- Topologia Estrela: programas mestre / escravo.

Considerando que os processadores *many-core* possuem uma característica nativa de paralelismo (diversos núcleos de processamento), é de se esperar que programas paralelos sejam freqüentemente executados. A Subseção 2.2.1 aborda questões referentes às comunicações coletivas (típicas em programas paralelos) e faz uma relação com possíveis topologias que melhor mapeiam o padrão de comunicação.

2.2.1 Comunicação Coletiva

Um dos aspectos que merecem atenção especial em programação paralela é o fato de que um grupo de processos ou *threads* gera uma intensiva movimentação de dados. Esta movimentação de dados implica em uma comunicação e troca de informação de forma coletiva e isto pode afetar o desempenho do sistema. Portanto, a comunicação coletiva é o resultado da demanda de operações que envolvem dados globais e que precisam ser transmitidos e recebidos através de um modelo de passagem de mensagem (DUATO, 2002). Conseqüentemente, a rede de comunicação de dados tem um papel importante no desempenho dos programas paralelos.

A programação paralela que se destina a *clusters* ou máquinas multiprocessadas também se aplica a processadores *multi-core* e *many-core*. A principal diferença se deve ao fato de que a carga gerada é executada em *chip*. Como conseqüência, muda-se a dimensão do problema, influenciado pelas novas latências, largura de banda e disponibilidade de recursos, em especial da rede-em-*chip*.

A Figura 2.13 ilustra exemplos de comunicação coletiva e seus respectivos padrões de comunicação. As linguagens de programação paralela são baseadas em serviços que tem como objetivos simplificar e facilitar a modelagem ou descrição da comunicação coletiva que deve ser executada. Estes serviços podem ser relacionados com padrões regulares de comunicação, conforme cardinalidades de cada exemplo da Figura 2.13 descrito a seguir (DUATO, 2002):

One-to-one

(a) Comunicação circular. Apenas uma mensagem é enviada usando as primitivas *send* e *receive*.

(b) Comunicação em *pipeline*. Apenas uma mensagem é enviada usando as primitivas *send* e *receive*. Neste caso, a origem (P_1) apenas envia e o destino (P_n) apenas recebe.

One-to-all

(c) Comunicação pelo serviço *broadcast*. A mesma mensagem é enviada para todos os nós da rede.

(d) Comunicação pelo serviço *scatter*: A origem envia diferentes mensagens para diferentes nós da rede. Pode ser considerado como um *broadcast* personalizado.

All-to-one

(e) Comunicação pelo serviço *reduce*: Diferentes mensagens de várias origens são combinadas em uma única mensagem para um determinado nó da rede.

(f) Comunicação pelo serviço *Gather*: Diferentes mensagens de várias origens são concatenadas para um determinado nó da rede seguindo uma ordem de identificação (ID) de cada mensagem.

All-to-all

(g) Comunicação pelo serviço *All-broadcast*: Todos os processos enviam seu próprio *broadcast*. Normalmente as N mensagens são concatenadas baseadas no ID das origens.

(h) Comunicação pelo serviço *All-scatter*: Todos os processos enviam seu próprio *scatter*. As N mensagens concatenadas são diferentes para diferentes destinos.

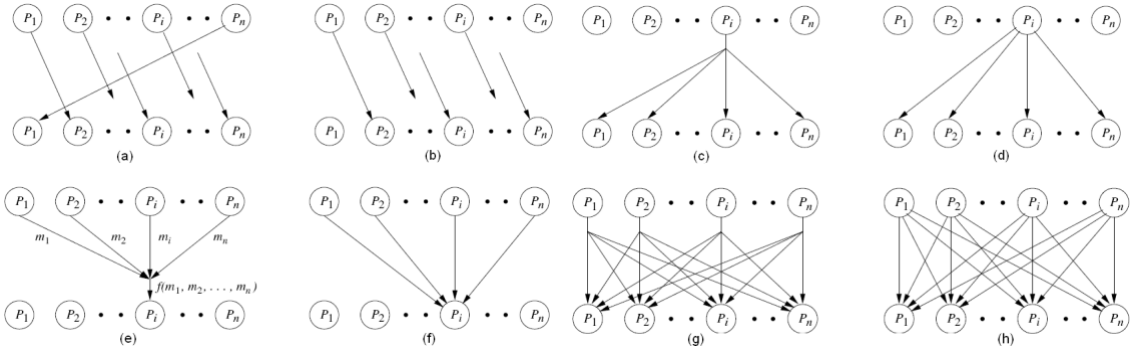


Figura 2.13: Exemplos de comunicação coletiva (DUATO, 2002)

O comportamento de diversos tipos de domínios de aplicação gera um determinado tipo de padrão de comunicação entre os processos ou *threads*. A literatura mostra que existem topologias que melhor se aplicam a problemas específicos. O que algumas pesquisas procuram são alternativas que possam aumentar a flexibilidade sem perder desempenho, em específico, das redes-em-*chip*. Uma das alternativas é a aplicação de conceitos de computação reconfigurável, conforme é descrito na Seção 2.3.

Outra característica relacionada às aplicações paralelas, que pode ser usada para definição de projeto de uma arquitetura de rede-em-*chip* é a Função de Isoeficiência. A Função de Isoeficiência (GRAMA, 1993) é uma métrica capaz de expressar a escalabilidade de um sistema para funcionar de forma eficiente, de acordo com o tamanho do problema a ser processado, em relação à quantidade de processadores disponíveis. A função de isoeficiência leva em conta os seguintes parâmetros: número de processadores, velocidade e largura de banda de interconexão de rede. A Equação 2 expressa o problema da escalabilidade de acordo com as seguintes métricas: Eficiência (E), *Overhead* total (To), Tamanho do problema (W) e custo de execução de cada operação (Tc). Se o tamanho do problema (W) é constante e o número de processadores aumenta, a eficiência diminui, porque o *overhead* total (To) aumenta diretamente ao aumento de processadores. Ao contrário, se o tamanho do problema (W) aumenta e o número de processadores é constante, a eficiência (E) aumenta porque o *overhead* total (To) cresce a taxas menores do que o tamanho da aplicação (W).

$$E = \frac{1}{1 + \frac{To}{WTc}} \qquad \text{Equação 2}$$

Considerando como foco a rede-em-*chip*, se há um aumento do número de núcleos interconectados para processar uma aplicação, há um aumento no tamanho da rede. O

aumento no tamanho da rede faz crescer o impacto no tempo de transmissão de pacotes, e como conseqüência há uma redução na eficiência da própria rede no processamento de uma aplicação que possui tamanho constante. Ao contrário, se o número de processadores for constante, não há aumento no tamanho da rede, a influência na variação do tamanho da aplicação neste caso resulta em um impacto menor na eficiência da rede-em-*chip*.

O desempenho da NoC à intensa comunicação coletiva demandada pelos programas paralelos está diretamente ligado ao problema e objetivos definidos nesta tese. A Figura 2.14 ilustra toda a comunicação ocorrida durante o processamento de uma aplicação paralela. Todos os nós se comunicam, mas em períodos de tempo existe um padrão de comunicação, de acordo com as cardinalidades apresentadas pela Figura 2.13. O que se propõe nesta tese é uma arquitetura de NoC Programável baseada em Múltiplos *Clusters* de *Cores* capaz de se adaptar aos padrões de comunicação coletiva e com isso aumentar o desempenho na transmissão de pacotes entre núcleos. Conforme é descrito nas seções seguintes, conceitos de computação reconfigurável são utilizados para aumentar a flexibilidade de NoCs proporcionando a capacidade de adaptação a domínios de aplicações diferentes.

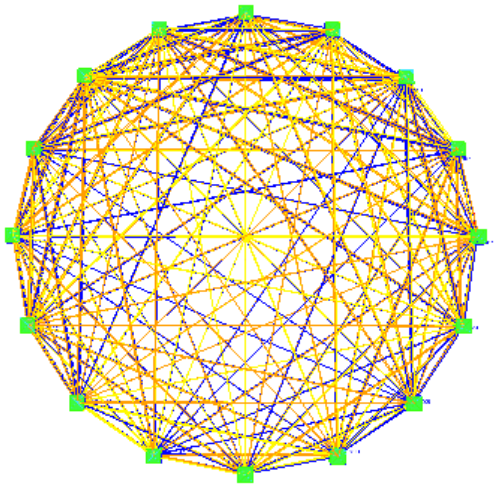


Figura 2.14: Comunicação coletiva cheia

2.3 Computação Reconfigurável

Através da Computação Reconfigurável (COMPTON, 2002) (MARTINS, 2003) (VERBAUWHEDE, 2004) (TODMAN, 2005) é possível alterar as propriedades de uma determinada arquitetura tornando-a mais flexível e com maior capacidade de se adaptar às novas demandas e necessidades. Computação Reconfigurável é um paradigma independente de dispositivo de *hardware*, mas depende inicialmente de um *hardware* programável para que se possa reconfigurar a arquitetura alvo.

Nesta seção o foco é a relação *hardware* versus arquitetura e programação versus reconfiguração. A Figura 2.15 ilustra algumas abordagens de projeto que possuem como objetivo o aumento de desempenho ou flexibilidade. Nesta figura, é apresentada uma comparação entre dispositivos de *hardware* e suas principais características, ressaltando a arquitetura reconfigurável como uma boa alternativa para agregar desempenho e flexibilidade, alcançando ganho final no processamento.

Portanto, um processador GPP (*General-Purpose Processor*) teria maior flexibilidade, uma vez que este é projetado para executar qualquer tipo de *software*. Em contrapartida, esta alta flexibilidade limita o desempenho, já que a complexidade adicional para flexibilidade exige, por exemplo, alta freqüência de operação. Do outro lado, um *hardware* ou arquitetura dedicada também chamada de ASIC (*Application Specific Integrated Circuit*) é mais simples, atinge alto desempenho operando em baixas freqüências, mas não possui flexibilidade, sendo útil apenas para um determinado tipo de aplicação. Co-processadores ou processadores dedicados conhecidos como ASIP (*Application Specific Instruction Set Processor*) procuram aliar desempenho ou flexibilidade, mas ambos possuem limitações na flexibilidade do *hardware* e do *software*, já que pós-fabricação não é possível alterar o projeto. Uma alternativa para alteração de projeto pós-fabricação está no uso de um dispositivo FPGA (*Field Programmable Gate Array*). O *hardware* projetado não muda, é um FPGA, mas este *hardware* é programável, oferecendo liberdade para alterar a arquitetura neste implementada.

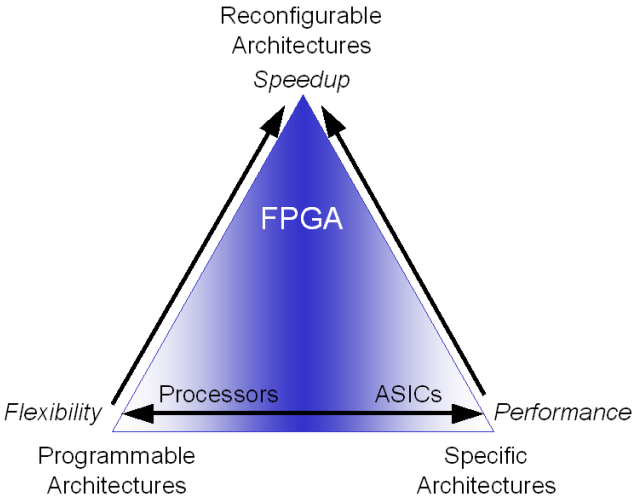


Figura 2.15: Abordagens de projeto

Cada uma destas abordagens apresenta vantagens ou desvantagens em relação à outra em se tratando de flexibilidade, desempenho e contexto de aplicação. Em resumo, existem contextos onde é necessário o uso de um GPP ou ASIC, mas para certas cargas de trabalho, a possibilidade de reconfiguração através do uso de um FPGA aumenta o desempenho do sistema, tal como os seguintes exemplos:

- FPGA aliado ao GPP: Trechos de *softwares* que possuem um alto custo de computação podem ser implementados em FPGA. A computação em *hardware* (FPGA) em freqüência mais baixa do que o *software* em GPP atinge desempenho superior. O principal motivo se deve ao circuito dedicado que é menos complexo que um processador, e assim, executa em tempo inferior o problema demandado.
- FPGA aliado ao ASIC: Um ASIC é um circuito dedicado não reconfigurável, portanto, só executa uma aplicação. Se para ganhar em desempenho todas as aplicações fossem executadas em apenas um ASIC, teríamos um *hardware* com várias alternativas e caminhos de execução de custo muito elevado e praticamente inviável. A solução é o FPGA que pode ser programado para implementar uma arquitetura dedicada sempre que houver uma nova demanda.

As seções seguintes exploram com mais detalhes os tipos de *hardwares*, e os tipos e níveis de reconfiguração.

2.3.1 Hardware Programável

Os *hardwares* reconfiguráveis seriam capazes de mudar sua forma, a quantidade e a localização física dos componentes. Este tipo de *hardware* ainda não existe, e o mais próximo que se pode chegar é através da reconfiguração da arquitetura implementada em um *hardware* programável. Os primeiros *hardwares* programáveis eram capazes de implementar funções lógicas e a partir destas implementações novas configurações eram obtidas. Os principais exemplos destes *hardwares* são (MARTINS, 2003): EPROM (*Erasable Programmable Read Only Memory*) e PLA (*Programmable Logic Array*). Com a crescente demanda por configurações mais complexas, novos *hardwares* programáveis surgiram, tais como: CPLD (*Complex Programmable Logic Device*), MPGA (*Mask Programmable Gate Array*) e o FPGA (*Field Programmable Gate Array*).

O FPGA tem sido largamente utilizado pela indústria e academia pela facilidade de prototipação e implementação de novas arquiteturas. Um bom exemplo é o projeto RAMP (*Reconfigurable Systems and a Research Accelerator Architecture Multiprocessor*) (KRASNOV, 2007), que tem como objetivo criar condições necessárias para testes de novos projetos voltados para a nova geração de processadores *many-core*. Vários FPGAs são interconectados, processadores e outros dispositivos são implementados nos FPGAs, e a partir desta infra-estrutura flexível e adaptável, são feitas emulações de *chips many-core* e, e.g., testes de sistemas operacionais, compiladores, linguagens de programação, além da própria arquitetura *many-core*.

A estrutura básica de um FPGA (ORDONEZ, 2003) (MARTINS, 2003) (SO, 2006) é ilustrada pela Figura 2.16. Um FPGA é composto por elementos lógicos programáveis (LB – *Logic Block*), interconectados por uma rede programável e por dispositivos de entrada e saída (IOB – *Input and Output Block*). Os blocos lógicos programáveis, responsáveis pela computação, são constituídos de três componentes básicos:

- LUT (*Lookup Table*): É uma representação direta de uma tabela verdade. Uma LUT de n entradas implementa, através de configurações, qualquer lógica combinacional de n entradas.
- FF (*Flip-Flop*): Capaz de adicionar lógica seqüencial ao circuito implementado inicialmente pelas LUTs.
- MUX (Multiplexador): Faz a seleção entre os resultados da LUT e do FF. É responsável por adicionar mais flexibilidade de implementação de circuitos aos LBs.

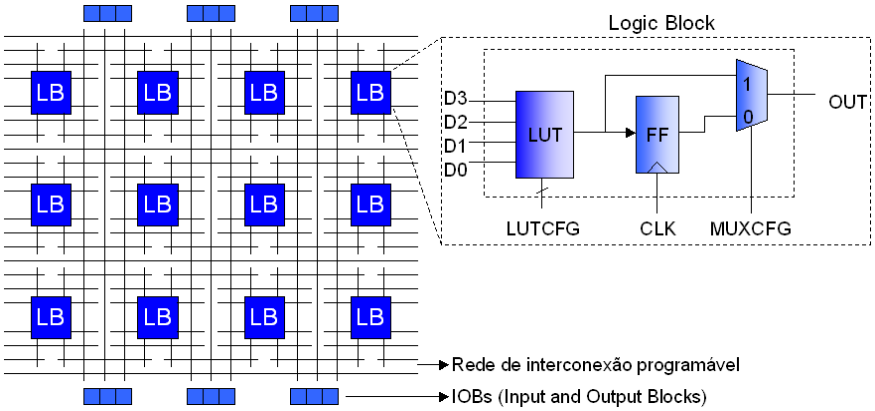


Figura 2.16: Componentes básicos de um FPGA (adaptado de SO, 2006)

Através de um FPGA ou de outro tipo de *hardware* programável é possível reconfigurar a arquitetura prototipada resultando na adaptação do novo *hardware* configurado a uma nova demanda ou contexto de aplicação. Os níveis ou tipos de reconfiguração são descritos na seção seguinte.

2.3.2 Níveis e Tipos de Reconfiguração

A abordagem de níveis de reconfiguração (FREITAS, 2006a) procura explorar mais de uma possibilidade de adaptação do *hardware*, modificando parâmetros que alteram diferentes características da arquitetura implementada. A exploração de níveis de reconfiguração pode resultar inclusive em uma independência de plataforma de *hardware* para configurar uma nova arquitetura. Através da Figura 2.17 é possível descrever as relações entre programação versus reconfiguração e *hardware* versus arquitetura.

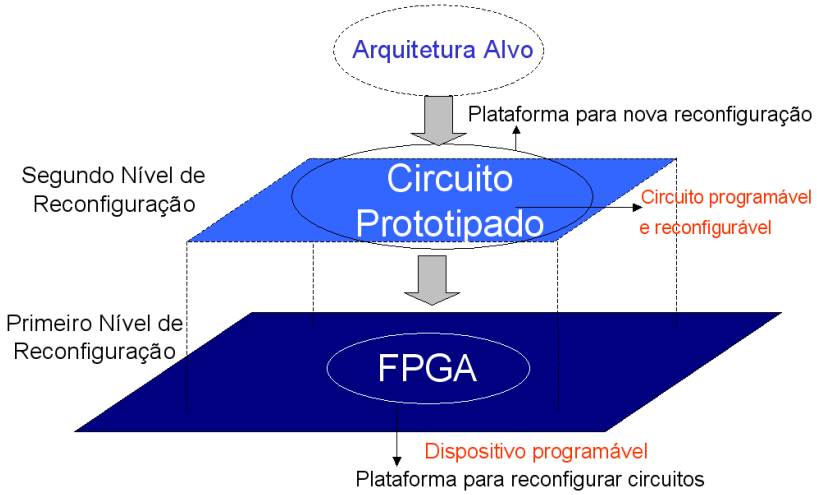


Figura 2.17: Níveis de reconfiguração

A Figura 2.17 ilustra dois níveis de reconfiguração, conforme descrito a seguir:

- Primeiro nível: O primeiro nível é suportado por um *hardware* programável, neste exemplo, um FPGA. Este *hardware* é a plataforma para reconfigurar circuitos implementados. A mudança do circuito implementado pode alterar as funcionalidades e objetivos da arquitetura alvo prototipada.
- Segundo nível: O segundo nível é baseado em um circuito programável capaz de suportar uma reconfiguração da arquitetura alvo. Neste caso, o

circuito implementado é a nova plataforma de reconfiguração da arquitetura independente do *hardware* programável, no exemplo desta figura, FPGA.

O primeiro e segundo níveis podem trabalhar de forma integrada, mas é possível ter um circuito programável e reconfigurável prototipado em um dispositivo não programável, por exemplo, um ASIC. A utilização integrada ou independente pode possibilitar um conjunto de tipos de reconfiguração (ORDONEZ, 2003), tais como:

- Reconfiguração estática: o funcionamento do *hardware* programável é interrompido e nova configuração é realizada.
- Reconfiguração dinâmica: a nova configuração é realizada em tempo de execução, com o *hardware* em funcionamento.
- Reconfiguração parcial: parte do *hardware* ou arquitetura é reconfigurada.
- Reconfiguração total: todo o *hardware* ou arquitetura é reconfigurada.
- Reconfiguração intrínseca: cada FPGA que compõe o sistema é parcialmente reconfigurado.
- Reconfiguração extrínseca: o sistema pode ser parcialmente reconfigurado considerando apenas os FPGAs que o compõe como unidade atômica de reconfiguração.

Um dos grandes problemas da reconfiguração dinâmica no primeiro nível é a latência. Dependendo do contexto, esta latência pode ser alta o suficiente para inviabilizar a flexibilidade desejada da arquitetura, resultando em um baixo desempenho do *hardware*. Uma das soluções seria manter a reconfiguração estática no primeiro nível e realizar a reconfiguração dinâmica no segundo nível explorando a baixa latência que pode ser baseada na mesma freqüência de operação do *hardware*. A maior parte dos Trabalhos Correlatos focam na reconfiguração de NoCs apenas no primeiro nível. Conforme está descrito no Capítulo 4, o foco principal da aplicação dos conceitos de reconfiguração nesta tese está no uso do segundo nível, tornando-a independente da plataforma de prototipação.

2.4 Trabalhos Correlatos

Um dos principais alvos de estudo e pesquisa para a próxima geração de processadores *many-core* são as *Networks-on-Chip*. Existem diversas propostas de arquiteturas de NoC, a maior parte com topologias fixas, com uma tendência comum de um roteador por núcleo de processamento. Estas arquiteturas podem impor caminhos longos, compartilhados e com contenções, aumentando o tempo de transmissão de pacotes. Um espelhamento mais adequado às redes de computadores faria com que os roteadores de NoCs fossem responsáveis por um maior número de núcleos. Os roteadores seriam programáveis e ativos, podendo inclusive processar parte das aplicações. De acordo com as cargas de trabalho, uma NoC flexível, com capacidade de adaptação aos padrões de comunicação também seria mais interessante. Os principais trabalhos correlatos buscam o aumento da flexibilidade com foco no aumento do ganho de desempenho. A Figura 2.18 ilustra as abordagens de programação e reconfiguração utilizadas em protótipos de NoCs e que são detalhadas nas subseções seguintes. O principal conceito ainda não apresentado é o PASIC (*Polymorphic* ASIC). Neste caso, o projeto da NoC utiliza um ASIC programável capaz de prover adaptação de topologias aos padrões de comunicação.

Além do uso de reconfiguração na arquitetura da NoC, outras pesquisas (Subseção 2.4.1) buscam alternativas para aumentar o desempenho focando, e.g., em aglomeração de núcleos por roteador. Outra abordagem para aumento da flexibilidade da topologia é através do uso da tecnologia *Radio-on-Chip* (CHANG, 2001) utilizada pela *Wireless NoC* (WANG, 2007). Apesar das características de topologias dinâmicas e irregulares, além de um menor custo de caminho de pacotes, a *Wireless* NoC não é programável e reconfigurável. As seções seguintes descrevem os trabalhos correlatos que buscam encontrar respostas e viabilizar NoCs que atendam a aglomeração de núcleos e o aumento da flexibilidade através do uso de conceitos de reconfiguração.

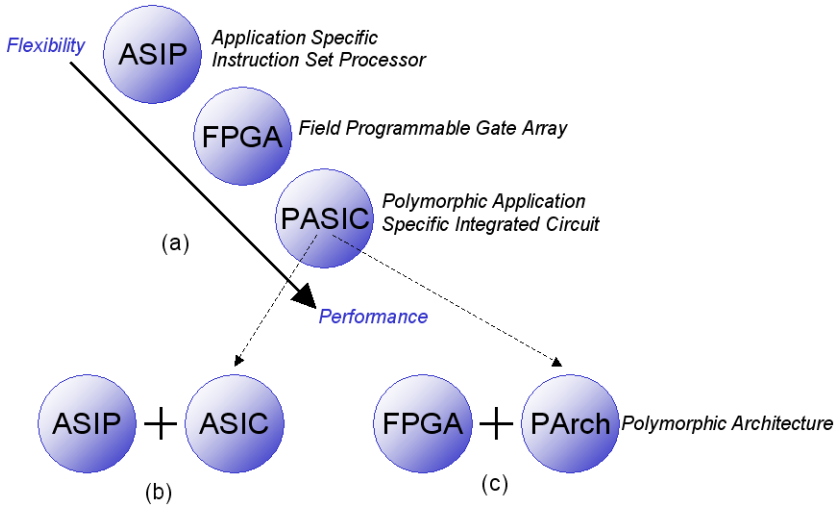


Figura 2.18: Abordagens para NoCs com arquiteturas adaptáveis. (a) abordagens básicas, (b) abordagem derivada ASIP+ASIC, e (c) abordagem derivada FPGA+PArch

2.4.1 NoCs Baseadas em Clusters de Núcleos de Processamento

A CHNoC (*Cluster-based Hierarchical NoC*) (LENG, 2005) é uma proposta de NoC indireta baseada em uma topologia irregular e *clusters* de núcleos de processamento e memórias. A arquitetura é composta por roteadores de dois tipos: *core* e *edge*. *Core Routers* são os roteadores intermediários responsáveis pela interconexão da rede. *Edge Routers* são responsáveis pela interconexão de um *cluster* ao restante da rede. Cada *cluster* é composto por um sistema de memória, núcleos de processamento, um barramento de interconexão e uma interface de rede para conexão com um *Edge Router*. Os roteadores são circuitos dedicados com *buffers* de saída. A argumentação por este tipo de *buffer* é feita com base nos seguintes pontos: melhor aproveitamento da largura de banda e redução do efeito de latência. O protocolo utilizado é *wormhole* como uma estratégia para otimizar o uso dos *buffers*. A técnica de roteamento é baseada em uma abordagem distribuída e parcialmente adaptativa. O experimento de validação da CHNoC foi através de descrição em SystemC e mapeamento de uma aplicação *MPEG-4 decoder*. Por se tratar de uma topologia dedicada e irregular, projetada para mapear um domínio específico de aplicação, constatou-se um bom desempenho da proposta.

A proposta GigaNoC (NIEMANN, 2005) é baseada em uma arquitetura de NoC direta com topologia regular para o contexto de múltiplos Processadores de Rede. A arquitetura é composta pelos seguintes componentes: *Switch Box* (SB - equivalente ao roteador), barramento do *cluster*, *link* de comunicação entre SBs. Cada *cluster* é composto por múltiplos núcleos de processamento, memórias ou dispositivos dedicados.

Os componentes do *cluster* são interconectados por um barramento que também está conectado ao *Switch Box*. As topologias utilizadas para ilustrar a proposta GigaNoC são: *mesh*, *torus* e *butterfly*. Cada *Switch Box* é composto por duas partes principais: i) Portas de entrada / saída e chave *crossbar*. As portas de entrada são compostas por *buffers*. ii) Estrutura de controle entre os núcleos e a rede. A proposta foi verificada através de descrição em VHDL (*VHSIC Hardware Description Language*) obtendo bons resultados de área ocupada e freqüência em FPGA.

O grande problema da aglomeração de núcleos em uma topologia fixa está na capacidade desta topologia em mapear os diversos domínios de aplicação (BJERREGAARD, 2006). Portanto, para um caso específico esta aglomeração pode resultar em alto desempenho e em outros casos um baixo desempenho. Como conseqüência desta baixa flexibilidade, algumas pesquisas exploram os conceitos de computação reconfigurável, com foco principal em FPGA, para aumentar a capacidade da NoC de se adaptar a novos padrões de comunicação. Uma alternativa é aumentar a flexibilidade do roteador para que este possa processar também partes dos programas em execução. As seções seguintes apresentam os trabalhos correlatos que exploram essas alternativas.

2.4.2 NoCs com Roteadores Programáveis

Aumentar o desempenho, através do processamento de instruções da aplicação nos roteadores da NoC (FERNANDES, 2008), é a alternativa apresentada pela IPNoSys (*Integrated Processing NoC System*). De acordo com Nguyen (2007) uma arquitetura *dataflow* pode ser explorada através do uso de NoCs. Instruções dentro dos laços de repetições são executadas pelos roteadores da NoC conforme o grafo de fluxo de dados gerado durante a compilação da aplicação. Conforme apresentado pela Figura 2.18, as arquiteturas de NoCs semelhantes à IPNoSys se enquadram no uso de ASIPs+ASICs. A Figura 2.19 ilustra este exemplo.

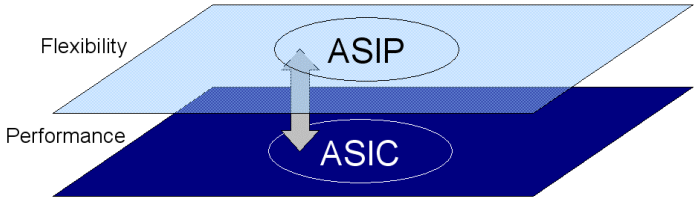


Figura 2.19: Abordagem ASIP+ASIC

Baseado em Nguyen (2007), a IPNoSys é uma proposta de NoC com roteadores ativos, ou seja, roteadores que processam um pacote de instruções da aplicação. As principais características da IPNoSys são: topologia *mesh* ou *torus* 2D, roteamento XY, técnicas de encaminhamento *virtual cut-through* combinado com *wormhole*, dois canais virtuais, controle de fluxo baseado em crédito, arbitragem distribuída e memorização na entrada. Os roteadores são compostos pelos seguintes blocos: uma unidade lógica e aritmética e uma unidade de sincronização. Os núcleos de processamento foram substituídos por núcleos de acesso à memória. Portanto, a proposta IPNoSys é uma arquitetura totalmente *dataflow*, e executa todas as instruções na rede e não somente as pertencentes aos laços de repetições. O modelo de computação da IPNoSys é baseado em passagem de pacotes entre roteadores, caracterizando um *pipeline* e explorando o paralelismo das transmissões. A IPNoSys foi descrita em SystemC no nível de precisão de ciclos e para sua simulação foi necessário o projeto de uma linguagem de descrição de pacotes. A dependência de dados foi realizada através do grafo de fluxo de dados

utilizado como linguagem intermediária para a descrição de pacotes. De acordo com os resultados, a IPNoSys possui um bom desempenho executando aplicações que tratam situações de *deadlock* e executam a transformada discreta do cosseno.

A IPNoSys é uma proposta com topologias fixas e a pesquisa ainda não aponta o estudo dos padrões de comunicação para possível adaptação da arquitetura. As próximas subseções descrevem a reconfiguração sem o uso de roteadores programáveis como forma de aumentar a flexibilidade da NoC.

2.4.3 NoCs Reconfiguráveis Baseadas em Dispositivos FPGA

FPGAs são alternativas para aumento de flexibilidade das NoCs. Com foco nos níveis de reconfiguração apresentados pela Figura 2.17, a Figura 2.20 ilustra o uso do FPGA como primeiro nível de reconfiguração. Os trabalhos apresentados nesta subseção focam apenas no primeiro nível.

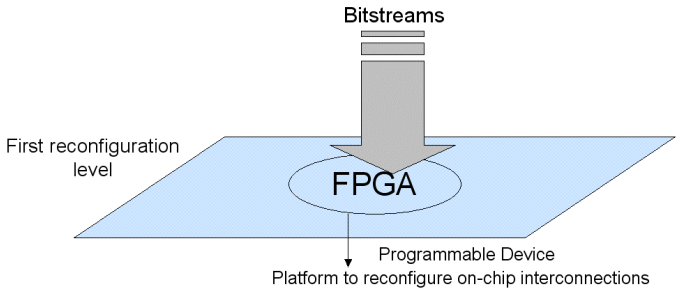


Figura 2.20: Primeiro nível de reconfiguração baseado em FPGA

Em (BARTIC, 2005) são apresentadas vantagens nas reconfigurações de topologias em função de um determinado domínio de aplicação. Uma das principais características do projeto é a arquitetura do roteador capaz de interconectar mais de um componente ou núcleo de processamento. Esta liberdade é conseguida através da nova implementação do roteador na plataforma FPGA. A técnica de encaminhamento de pacotes é baseada em *virtual cut-through* e o algoritmo de roteamento é determinístico com uma capacidade limitada de adaptabilidade. Cada roteador possui uma chave *crossbar* que utiliza técnica de arbitragem por saída para resolver problemas de conflitos entre as entradas. As portas de saída possuem *buffers* para alocação dos pacotes. A escolha foi feita em função do problema de *head of the line blocking*. O controle de fluxo é baseado na técnica *handshake* com linhas de requisição e confirmação. A arquitetura foi descrita em VHDL, e para cada protótipo foi implementada uma topologia de rede diferente, conforme relação a seguir: hipercubo, *mesh*, *torus*, árvore e *crossbar*. Os resultados e conclusões apontam para a escolha da topologia em função dos seguintes parâmetros: área, largura de banda, latência, coeficiente de saturação da rede e limitações físicas.

Em (HUR, 2007) é feito um estudo da adaptação da chave *crossbar* em função de domínios de aplicação. Ressalta-se neste trabalho que a chave *crossbar* é um dos blocos principais utilizados em roteadores e NoCs de diversos tipos, por este motivo, a importância do estudo. A técnica se baseia na implementação de chaves *crossbar* parciais. Uma chave *crossbar* completa possui n multiplexadores por porta, enquanto que na chave *crossbar* parcial o número de multiplexadores é variável dependendo do grafo de topologia da aplicação. A implementação dessa proposta é baseada no protocolo de controle de fluxo *handshake* e na técnica de arbitragem *round-robin*. O protótipo foi implementado em FPGA, e avaliado através de alguns grafos de aplicações. O objetivo principal era avaliar a redução de área e o desempenho da chave

crossbar parcial em relação à completa. Em todos os resultados houve redução de área e melhor desempenho.

PNoC (*Programmable NoC*) (HILTON, 2006) é um projeto que explora a programação do FPGA para reconfiguração das interconexões da NoC. A topologia de rede proposta consiste de uma série de subredes que contém roteadores, núcleos de processamento, entre outros elementos. PNoC é baseada em chaveamento por circuito e controle de fluxo baseado em *handshake*. Os roteadores trabalham com tabelas de roteamento, *buffers* nas portas de entrada, e um árbitro central. Cada roteador é parametrizado em número de portas e largura dos dados e endereços das linhas de conexão de cada porta. O protótipo foi implementado em FPGA e os resultados principais, baseados em área ocupada e freqüência de operação, apontam a PNoC mais eficiente em comparação realizada com barramentos.

A reconfiguração parcial e dinâmica é explorada através da proposta de rede CoNoChi (PIONTECK, 2008). A CoNoChi (*Configurable Network-on-Chip*) suporta mudanças de topologias de rede em tempo de execução adicionando ou removendo elementos de rede sem parar seu funcionamento. A arquitetura da CoNoChi consiste em fatias de reconfiguração independentes que possuem *switches*, *links* ou unidades de processamento. O roteamento é realizado através de tabelas de rotas, que são distribuídas por uma unidade central para cada fatia, e o encaminhamento de pacotes está baseado no algoritmo *virtual cut-through*. A pilha de protocolos é baseada em endereços físicos e lógicos. Os endereços físicos são usados pelos elementos de rede para roteamento, os endereços lógicos pelas aplicações. O árbitro de cada *switch* funciona de forma adaptativa em função de parâmetros de cabeçalho, levando em consideração questões como qualidade de serviço, que também é suportado pela CoNoChi. Como a reconfiguração dinâmica e parcial não é trivial, o FPGA foi utilizado para verificar a viabilidade de implementação. A avaliação da proposta foi feita com o auxílio de uma simulação em SystemC.

A remoção de *links* de comunicação da NoC é a proposta feita por Wang (2008). Neste trabalho o objetivo é reduzir o custo de uma NoC através de uma análise prévia do tráfego de rede. Se em determinado *link* houver um tráfego baixo ou inexistente, é feita a remoção do *link*. A proposta também se baseia na redução do custo associado às políticas para tratamento a *deadlocks*, uma vez que com um número menor de *links* de comunicação, este tratamento fica mais fácil. A NoC proposta consiste de uma topologia *mesh*, roteamento *wormhole*, roteadores com portas de entrada *bufferizadas*, chave *crossbar* 5x5 e módulo de arbitragem. À medida que os *links* de comunicação da NoC são removidos, a topologia irregular se adapta ao tráfico da rede. Os principais resultados do trabalho se baseiam na redução de área e, principalmente, no aumento do desempenho através da NoC modelada em um simulador escrito em C++.

Em um caminho contrário ao proposto em (WANG, 2008), em (KIM, 2005) é proposto o aumento da largura de banda dos barramentos da chave *crossbar* para atender a necessidade de maior vazão em função do aumento do tráfego entre dois nós comunicantes. Este aumento da largura de banda se faz através de *links* adicionais para prover a comunicação. No mesmo caminho de (HUR, 2007) o estudo é focado na chave *crossbar* usada para o projeto de NoCs. Outras propostas de chaves *crossbar* reconfiguráveis (EGGERS, 1996) (FEWER, 2003) não abordam o contexto NoCs.

2.4.4 NoCs Reconfiguráveis Baseadas em ASICs Polimórficos

Metodologias para projeto de topologias de NoCs em função de domínios de aplicações são apresentadas em (BERTOZZI, 2005) e (HO, 2006). Estes trabalhos não propõem a reconfiguração de topologias, mas uma metodologia de avaliação da carga de trabalho para projeto específico de topologia em ASIC. No entanto, é possível encontrar na literatura propostas de NoC polimórficas baseadas em plataformas ASIC programáveis. A Figura 2.21 ilustra o uso do PASIC como plataforma de reconfiguração de primeiro nível para as NoCs, conforme é apresentado pelos trabalhos desta subseção.

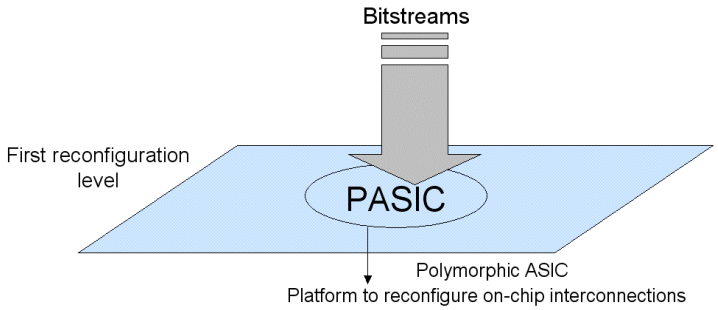


Figura 2.21: Primeiro nível de reconfiguração baseado em PASIC

Em (MERCALDI-KIM, 2008) é apresentado a *Polymorphic On-Chip Network*. A principal motivação do trabalho está na adaptação da NoC, sem o uso de FPGAs, em função de padrões de tráfego. A NoC consiste de uma coleção de blocos de rede configuráveis, tais como *buffers* e *crossbar*. Os parâmetros adaptáveis são: largura de banda do *link*, *buffers* e topologia da rede. As principais características da NoC avaliada são: topologias *butterfly*, *fat tree*, *flattened butter*, *mesh* e *ring*, algoritmo de roteamento determinístico e algoritmo de encaminhando *store-and-forward* e *wormhole*. A reconfiguração é baseada na repetição de padrões de blocos chamados de Regiões e *Crossbars*. Cada conexão dentro da *crossbar* é estaticamente configurável e cada região consiste de múltiplas fatias da rede compostas pelos seguintes elementos: *buffers* de entrada, roteadores, conexões *crossbar* e árbitros. A reconfiguração é baseada na ativação de conexões entre Regiões e *Crossbars*. A política de reconfiguração não está definida, mas propõe-se que seja feita através do sistema operacional ou alguma aplicação em tempo de execução. Os resultados foram validados através de um simulador de rede (WANG, 2003) executando aplicações de Pareto. A área é reduzida em função da adaptação da topologia da rede sem perder em desempenho.

Seguindo a mesma idéia de adaptação da topologia, em Stensgaard (2008) é apresentado o projeto da ReNoC. A proposta se baseia na inserção de uma nova camada entre os roteadores e os *links*. Esta nova camada consiste de interfaces responsáveis por ativar os *links* entre os roteadores. Desta forma, é possível alterar a topologia de uma *mesh* para uma nova topologia que segue o padrão de comunicação identificado na carga de trabalho. A adaptação está baseada em uma configuração de inicialização resultante de um grafo de topologias das aplicações. Os resultados da ReNoC focam em aplicações baseadas em *Video Object Decoder* para verificar área, e consumo de potência e energia.

Ao contrário das propostas anteriores, em (JOSEPH, 2008) não é feita uma adaptação da topologia através de interconexões, mas um mapeamento de nós de computação através do grafo da aplicação. A NoC chamada de RECONNECT é um ASIC composto de fatias de *arrays* regulares baseado na topologia *honeycomb*

(STOJMENOVIC, 1997). Cada aplicação é fragmentada em subestruturas chamadas de *HyperOps*. Um *HyperOp* é um sub-grafo do fluxo de dados da aplicação em execução. Em tempo de execução, os elementos de computação, que constituem as fatias de *array*, são agregados para mapear as operações do *HyperOp*. Esta agregação e mapeamento é a configuração proposta para que a NoC se adapte ao domínio de aplicação. A RECONNECT consiste da seguinte estrutura: interface de rede entre roteador e elementos de computação, roteadores com 4 saídas para a rede (norte, sul, leste, oeste), algoritmo de roteamento XY, chave *crossbar* de roteador com arbitragem de saída (algoritmo *round-robin*), *buffers* nas portas de entrada e mecanismo de controle de fluxo por canais virtuais. Os resultados verificaram uma redução da área e potência em relação a topologia *mesh* e o aumento do desempenho através da execução de *kernels* de aplicações baseadas em multimídia e processamento de sinais digitais.

Duas das propostas apresentam o termo Polimorfismo em ASICs aplicado a NoCs quase simultaneamente, Mercaldi-Kim (Estados Unidos) em junho de 2008 e Joseph (Índia) em julho de 2008. No entanto, em termos de adaptação de topologias, apenas o primeiro trabalho satisfaz a característica esperada de mudança da forma ou das interconexões realizadas capazes de mapear um padrão de comunicação. Em Stensgaard (2008), apesar do termo polimorfismo não aparecer, a característica da NoC se enquadra nos conceitos PASIC.

Por fim, uma característica comum às NoCs PASIC é o fato da flexibilidade existir, mas não ser tão alta. Afinal, em uma plataforma ASIC não é possível adicionar elementos ao projeto implementado, mas alterar e remover conexões seguindo restrições estabelecidas pelo projeto inicial implementado no ASIC. Por este motivo, as propostas que utilizam FPGA são mais flexíveis, podendo usar em novas configurações, componentes ainda não utilizados do dispositivo. Além disso, o fato do FPGA possuir inúmeros blocos lógicos, a realocação das unidades construtivas da NoC após uma reconfiguração possui um grau de liberdade maior, devido a alta flexibilidade do FPGA. A grande vantagem do PASIC é justamente o desempenho e o menor tempo de reconfiguração. Esta conclusão pode ser ilustrada pela Figura 2.18 através da relação entre flexibilidade, desempenho e dispositivos.

2.4.5 Contribuição da Tese Versus Trabalhos Correlatos

A arquitetura da NoC apresentada nesta tese e as tomadas de decisão referentes ao projeto e avaliação são descritos em detalhes nos Capítulos 3 e 4. No entanto, ao fim deste capítulo é possível relacionar as características da NoC Programável baseada em Múltiplos *Clusters* de *Cores* apresentadas na Seção 1.5 com os trabalhos correlatos com o intuito de verificação do ineditismo da arquitetura proposta:

- Característica 1: Arquitetura de roteador programável e reconfigurável para suporte a *clusters* de núcleos de processamento.
    - (LENG, 2005) e (NIEMAN, 2005) apresentam *clusters* com topologias fixas e roteadores não programáveis.
- Característica 2: Arquitetura de processador de rede em *chip* para roteadores de NoCs.
    - Os trabalhos correlatos não apresentam roteadores programáveis baseados em Processadores de Rede. (NGUYEN, 2007) e

(FERNANDES, 2008) apresentam roteadores ativos baseados no modelo de fluxo de dados.

- Característica 3: Arquitetura de chave *crossbar* reconfigurável para roteadores de NoCs.
  - (KIM, 2005) e (HUR, 2007) apresentam chaves *crossbar* reconfiguráveis baseadas na programação do FPGA.
  - A proposta desta tese se baseia na reconfiguração de topologias sobre a chave *crossbar* usando o segundo nível de reconfiguração. O FPGA é utilizado como plataforma de prototipação.
- Característica 4: Arquitetura de NoC com topologias adaptáveis aos padrões de comunicação coletiva.
  - Os trabalhos (BARTIC, 2005), (HILTON, 2006), (PIONTECK, 2008), (WANG, 2008) focam apenas na programação do FPGA, ou seja, primeiro nível de reconfiguração.
  - O trabalho (JOSEPH, 2008) foca apenas no primeiro nível, com foco no ASIC, mas não muda a forma de interconexão ou topologia, não possui roteador programável e não explora *clusters* de núcleos de processamento.
  - Os trabalhos (MERCALDI-KIM, 2008), (STENSGAARD, 2008) focam no primeiro nível de reconfiguração baseado em PASIC, não possuem roteadores programáveis e não exploram *clusters* de núcleos de processamento.

A Tabela 2.5 sumariza as comparações realizadas. A proposta de NoC Programável baseada em Múltiplos Clusters de Cores é chamada de MCNoC (Multi-Cluster NoC – nome original em inglês), conforme publicações da pesquisa. MCNoC é também a sigla citada nos próximos capítulos em referência a arquitetura de NoC apresentada nesta tese. É importante ressaltar que o FPGA (coluna Reconfigurável) é utilizado como dispositivo de prototipação e reconfiguração estática, sendo o segundo nível de reconfiguração responsável pela arquitetura polimórfica (PArch), Figura 2.18c. A reconfiguração dinâmica no segundo nível é abordada pela tese, que está baseada na chave *crossbar* reconfigurável e no processador de rede em *chip*. Portanto, através da programação dos roteadores da MCNoC, é possível suportar novos padrões de comunicação coletiva, aumentando o desempenho.

Outro ponto importante diz respeito ao fato de que nenhuma das arquiteturas de NoCs relacionadas nesta subseção foram propostas com foco em comunicações coletivas. Portanto, o projeto da MCNoC é a única com este propósito.

Para finalizar, o trabalho desenvolvido por Vassiliadis (2007), chamado de FLUX, utiliza o FPGA para alterar topologias configuradas de acordo com padrões de comunicação que são identificados. No entanto, não é mencionado NoC como foco da pesquisa, mas arquiteturas multi-processadas baseadas em *clusters*. Porém, nada impede que a técnica seja avaliada para o contexto *intra-chip*.

Tabela 2.5: Comparações entre características de NoCs relacionadas

NoC	Topologia	Clusterizada	Programável	Reconfigurável
MCNoC	**Adaptável**	**Sim**	**Sim**	**Sim (FPGA+PArch)**
(LENG, 2005)	Fixa	Sim	Não	Não
(NIEMAN, 2005)	Fixa	Sim	Não	Não
(NGUYEN, 2007)	Fixa	Não	Sim	Não
(FERNANDES, 2008)	Fixa	Não	Sim	Não
(KIM, 2005)	Adaptável	Não	Não	Sim (FPGA)
(HUR, 2007)	Adaptável	Não	Não	Sim (FPGA)
(BARTIC, 2005)	Adaptável	Não	Não	Sim (FPGA)
(PIONTECK, 2008)	Adaptável	Não	Não	Sim (FPGA)
(WANG, 2008)	Adaptável	Não	Não	Sim (FPGA)
(HILTON, 2006)	Adaptável	Não	Não	Sim (FPGA)
(MERCALDI-KIM, 2008)	Adaptável	Não	Não	Sim (PASIC)
(STENSGAARD, 2008)	Adaptável	Não	Não	Sim (PASIC)
(JOSEPH, 2008)	Fixa	Não	Não	Sim (PASIC)

3 METODOLOGIA DE PROJETO E AVALIAÇÃO

A metodologia de trabalho busca atender alguns requisitos necessários na avaliação e validação da proposta de arquitetura da MCNoC. Neste sentido, a definição dos modelos utilizados contribui para a especificação de métricas adequadas para geração de resultados, avaliação e validação (JAIN, 1991) (MENASCE, 2003) (LILJA, 2004) (LE BOUDEC, 2007). Os três modelos para avaliação de desempenho são: modelo analítico, modelo de simulação e modelo baseado em experimentação real ou medição.

A medição é realizada a partir da existência física do protótipo. No entanto, em uma metodologia de desenvolvimento de uma nova arquitetura, o protótipo ou *hardware* implementado ainda não existe. Além disso, antes da implementação física se faz necessário o uso de outros modelos de avaliação que possam identificar falhas e possibilitar correções com um menor custo de reprojeto. Considerando esta restrição referente ao modelo de medição, as avaliações realizadas nesta tese são feitas com base no modelo analítico e de simulação. Estes modelos são utilizados para projeto e obtenção de resultados, além da verificação da arquitetura proposta.

Para avaliação do funcionamento, comportamento e desempenho são utilizadas cargas de trabalho artificiais e naturais. O modelo artificial não utiliza trechos ou programas da carga de trabalho real. Neste modelo são utilizados programas específicos e parâmetros de descrição. O modelo natural consiste de componentes básicos (programas ou trechos) da carga real, ou de rastreamentos baseados em arquivos de registros. Os arquivos de registros (e.g., *logs*, rastros ou traços) são compostos por dados em seqüência cronológica além de eventos específicos que representam algum tipo de ocorrência no sistema. Considerando os dois tipos de modelos de cargas de trabalho, a avaliação adotada para a arquitetura proposta se divide em dois tipos:

- Funcionamento e comportamento da MCNoC baseados no modelo artificial. Os modelos artificiais são adequados para testar e verificar funcionamento e comportamento específicos de um determinado sistema.
- Desempenho baseado no modelo natural. Os modelos artificiais, por não representarem cargas de trabalho reais, não são adequados para avaliação de desempenho.

Portanto, a primeira fase de projeto é dividida nas seguintes etapas:

1. Estudo da literatura e estado da arte relacionada à tese.
2. Identificação do problema e contribuição para o estado da arte.

3. Definição dos modelos de desenvolvimento e avaliação de projeto.
4. Projeto da arquitetura de NoC.

Além das características descritas na Subseção 2.1.3, que podem influenciar no desempenho de aplicações paralelas, a literatura (KUMAR, 2005b) (BJERREGAARD, 2006) aponta a escalabilidade de arquiteturas tradicionais de interconexão como um problema no contexto *intra-chip*. Sendo assim, soluções como barramentos e chaves *crossbar* seriam substituídas por *Networks-on-Chip*. No entanto, como foi descrito no Capítulo 2, as principais alternativas de NoCs propostas abordam arquiteturas baseadas em 1 roteador por núcleo. Dois trabalhos correlatos (LENG, 2005) (NIEMANN, 2005) são as exceções, que propõem *clusters* de núcleos de processamento. Porém, soluções como barramentos e chaves *crossbar* possuem um alto desempenho para um tamanho pequeno de núcleos interconectados e poderiam ser exploradas para garantir a comunicação entre núcleos mais próximos da rede. Portanto, de acordo com os problemas relativos às interconexões e NoCs, é possível justificar as hipóteses apresentadas no Capítulo 1, além das definições de projeto, da seguinte forma:

- O aumento da flexibilidade e adaptação da topologia ao padrão de comunicação e os *clusters* de núcleos de processamento reduzem a quantidade de saltos. Como conseqüência, há a redução da influência da rede no tempo de transmissão e na eficiência do sistema.
- A localização dos núcleos de processamento em um único *cluster* explora a proximidade para comunicação coletiva através de chaves *crossbar* de alto desempenho.

O escopo da tese traçado pela metodologia foca especificamente nos problemas e atrasos relativos às redes-em-*chip* considerando as comunicações coletivas de programas paralelos. Portanto, os resultados de desempenho apresentam métricas relativas aos tempos de transmissão de pacotes, latência de rede, largura de banda e vazão da rede. No mesmo caminho, o custo avaliado foca somente na proposta de rede-em-*chip* através de métricas como componentes utilizados, consumo de potência e energia. Otimizações que podem estar relacionadas às aplicações paralelas, compiladores, sistemas operacionais, ou arquiteturas dos núcleos de processamento e que podem influenciar no desempenho e eficiência do sistema, não fazem parte do escopo da tese. Esta influência pode ser exemplificada através das taxas de injeção, que no caso da avaliação descrita na Seção 3.4, se baseia na arquitetura de execução dos *benchmarks* utilizados. Estas taxas de injeção também são influenciadas pelo próprio desempenho da rede, e nesse caso, se faz necessário um ambiente de simulação completo para avaliação. Portanto, todo o sistema relacionado aos processadores *many-core*, que inclui sub-sistemas de memórias e *caches*, além de outros dispositivos periféricos não foram avaliados. Para realizar avaliações deste tipo, é necessário um ambiente de simulação completo que propicie o controle de todas as variáveis do sistema. Um ambiente como este ainda não se encontra disponível.

As seções seguintes descrevem as ferramentas, o uso, as métricas e os objetivos na escolha de cada um dos modelos de projeto e avaliação. As etapas estão distribuídas e descritas em cada uma das seções a seguir.

3.1 Simulação em ArchC e SystemC

ArchC (RIGO, 2004) é uma linguagem de descrição de arquitetura de processadores capaz de gerar automaticamente simuladores baseados em SystemC. SystemC (GHOSH, 2001) é um conjunto de classes escritas em C++ que permite modelos de descrição de *hardware*. A MCNoC foi projetada para explorar a flexibilidade através de conceitos de reconfiguração, mas também por programação ou *softwares*. Neste caso, cada roteador da MCNoC é programável, conseqüentemente há um processador de rede em cada roteador da MCNoC. Ao longo das próximas seções e capítulos o processador de rede poderá ser chamado de NPoC (*Network Processor on Chip*), conforme nome original em inglês publicado em artigos.

Portanto, usando o ArchC versão 1.6 e o SystemC 2.1 é possível simular e verificar o funcionamento e desempenho da arquitetura do conjunto de instruções do processador de rede proposto através de modelos funcionais ou com precisão de ciclos. As principais métricas que podem ser obtidas através da simulação são: tempo de execução, quantidade de instruções executadas, número de ciclos, número de acessos aos bancos de registradores e memórias, ou de outros dispositivos modelados.

Portanto, as etapas da metodologia referentes a esta seção são:

5. Descrever a arquitetura do conjunto de instruções do processador de rede em ArchC modelo funcional.
6. Descrever a arquitetura do conjunto de instruções do processador de rede em ArchC modelo com precisão de ciclos.
7. Avaliar o funcionamento e desempenho da arquitetura do processador de rede.

O objetivo principal desta etapa da metodologia está em projetar e verificar a funcionalidade e desempenho do conjunto de instruções propostos para o processador de rede em *chip*. Portanto, programas testados em outras etapas são primeiramente experimentados nos modelos em ArchC.

3.2 Modelagem e Simulação Usando Redes de Petri

Através das Redes de Petri (MURATA, 1989) (MACIEL, 1996) (GIRAULT, 2002) (PENHA, 2004) é possível modelar algumas características de redes de comunicação, tais como: concorrência, controle, conflitos, sincronização e compartilhamento. As Redes de Petri (RdP) são representadas através de quatro elementos básicos ilustrados pela Figura 3.1.

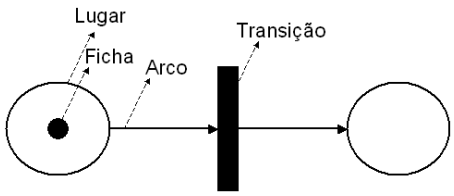


Figura 3.1: Elementos de uma Rede de Petri

- Lugares ou *Places*: representam uma condição, uma atividade ou um recurso.
- Fichas, Marcas ou *Tokens*: representam o estado de um sistema.
- Arcos: indicam os Lugares de entrada e saída para as Transições.

- Transições: representam um evento.

As etapas referentes à aplicação das Redes de Petri no projeto e avaliação da proposta de NoC são:

8. Modelar a vazão de pacotes para padrões de comunicação coletiva nas seguintes abordagens: barramento, chave *crossbar* e NoC com topologias *mesh* e *torus*. O modelo corresponde ao número de processadores de um *cluster* proposto para a arquitetura da NoC sob o impacto das latências de rede.
9. Verificar o desempenho de vazão de pacotes entre as abordagens modeladas para projeto da chave *crossbar* em roteadores de *clusters*.
10. Modelar e verificar o funcionamento entre duas abordagens para identificação de tráfegos (padrões de comunicação) usadas pelo roteador da MCNoC.

Para verificação dos modelos foi utilizado um simulador de Redes de Petri chamado VisObjNet (DRATH, 2008). Existem algumas variações para as Redes de Petri, tais como: temporizadas, estocásticas e coloridas. Em alguns dos modelos utilizados, também suportado pelo simulador, foram usadas temporizações nas Transições da RdP. Estas temporizações representam as latências impostas por cada uma das abordagens modeladas. A latência pode ser definida como o atraso imposto à primeira palavra de dados (ou célula de um *byte*) do pacote enviado. Tal como um *pipeline*, as demais células do pacote não sofrem o efeito da latência no tempo final de transmissão.

As restrições do modelo para verificação do efeito da latência no desempenho são: os pacotes possuem tamanho de uma célula, as variáveis largura de banda e tamanho da mensagem, além dos problemas relativos às contenções em roteadores de NoCs não foram modelados. O efeito das demais variáveis no tempo de transmissão é apresentado na Seção 3.4.

3.3 Simulação da Descrição de um Protótipo de Hardware

Através da descrição de um protótipo de *hardware* é possível verificar a viabilidade da MCNoC. Um dos principais objetivos com uma simulação deste tipo de descrição é avaliar métricas que podem resultar em barreiras físicas ou de desempenho na implementação final da arquitetura. Por este motivo, foi selecionado um conjunto de ferramentas (XILINX, 2008) (MENTOR GRAPHICS, 2008) para descrição e simulação do *hardware*, conforme a seguinte lista:

- ISE: Ambiente da Xilinx usado para descrição, configuração e resultados de síntese do *hardware*. Através dos resultados de síntese foi possível coletar informações de componentes utilizados no FPGA, tais como LUTs e Flip-Flops.
- ModelSim: Ambiente da Menthor Graphics utilizado para simulação do *hardware*. Através deste ambiente é possível verificar as latências impostas por cada componente. Além disso, é possível visualizar o caminho de dados e controle para correções na descrição do *hardware*.
- Xpower: Ambiente da Xilinx utilizado para geração dos resultados de consumo de potência.

Os protótipos da MCNoC foram descritos em VHDL e as sínteses e simulações foram realizadas para as seguintes famílias de FPGAs da Xilinx: 2vp30ff1152-6 e 2vp100ff1704-6.

Os processadores de rede das propostas de roteadores programáveis são descrições adaptadas do projeto Plasma (RHOADS, 2007) baseado no ISA (*Instruction Set Architecture*) do MIPS (PATTERSON, 2005). A adaptação foi realizada para aproximar o Plasma do projeto de processador de rede NPoC (*Network Processor on Chip*) da proposta apresentada na Seção 4.2. Esta adaptação está baseada na redução de instruções e registradores e acelera a verificação do roteador programável uma vez que o Plasma é descrição estável e conhecida da comunidade.

Cargas de trabalho artificiais baseadas na implementação de topologias regulares, controle de pacotes enviados, e variação dos sinais de entrada foram utilizadas para avaliação de consumo de potência. Através dos resultados de consumo de potência (Cp), e do tempo de transmissão (Tt) para as cargas artificiais, é possível expressar o consumo de energia (Ce) da proposta de NoC através da Equação 3:

$$Ce = Cp * Tt \qquad \text{Equação 3}$$

Consumos de potência e energia são importantes como métricas de verificação de viabilidade da arquitetura proposta. Portanto, consumo de potência pode ser definido como a taxa de trabalho realizado ou de energia consumida por unidade de tempo, e consumo de energia a capacidade de um sistema realizar trabalho. Quanto menores os valores encontrados para cada uma das métricas, melhor é a eficiência de consumo potência ou energia. No entanto, é possível encontrar sistemas que possuem um baixo consumo de potência, mas realizam o trabalho em um longo período de tempo, e por este motivo possuem um alto consumo de energia.

Relacionadas a esta seção, as etapas da metodologia são:

11. Descrever o *hardware* das versões propostas da MCNoC em VHDL.
12. Avaliar a síntese e simulação da MCNoC.
13. Avaliar os resultados de ocupação do FPGA, latências dos elementos da MCNoC e consumo de potência e energia.

As avaliações de desempenho não são realizadas pela simulação do *hardware*. Estes resultados são obtidos através de modelos analíticos, traços de cargas de trabalho paralelas reais, e analisadores de tráfego. O principal motivo se deve ao tamanho grande dos traços em alguns casos, demandando um tempo e uma complexidade maior de simulação. Um ambiente específico que fornece resultados mais rápidos foi desenvolvido para atividades de avaliações de desempenho, conforme descrito na seção seguinte.

3.4 Avaliação de Desempenho de Cargas de Trabalho Paralelas

Nesta etapa da avaliação da MCNoC são utilizadas modelagens analíticas e interpretações dos traços de aplicações reais, através de uma ferramenta de análise de tráfego desenvolvida em Python. A reprodução da transmissão dos pacotes (dados dos traços) segue um comportamento determinístico sem contenções. No entanto, o tempo de transmissão de pacotes pode resultar em um atraso imposto pela rede de comunicação (JAIN, 1991) (MENASCÉ, 2003) (HENNESSY, 2006) (FREITAS, 2007),

que pode diminuir o desempenho do sistema. Neste contexto, duas métricas principais da rede são: latência (L) e largura de banda (B). O tempo de transmissão depende também do tamanho do pacote (P) que deve ser transmitido e do número de pacotes (n). Portanto, o tempo final de transmissão (T) pode ser expresso da seguinte forma (Equação 4):

$$T = \left(L + \frac{P}{B} \right) * n \qquad \text{Equação 4}$$

O tempo de residência (R) de um pacote em cada elemento de rede (saltos) depende do tempo de serviço (S) ou processamento deste pacote, e do tempo de espera em fila (F). O tempo de espera em fila depende da quantidade e do tamanho dos pacotes esperando processamento. A cada nova visita (V) do pacote à fila (e.g., *buffers* de entrada), o tempo de residência final é recalculado. Um alto tempo de residência pode ser considerado como uma contenção ou gargalo de rede, e pode ser expresso pela Equação 5. Para uma rede ideal sem contenções, o tempo de residência é igual ao tempo de serviço, em alguns casos, latência do roteador.

$$R = (F + S) * V \qquad \text{Equação 5}$$

Portanto, o tempo total de transmissão (Tt) pode ser definido pela Equação 6:

$$Tt = T + R \qquad \text{Equação 6}$$

Estas equações foram usadas no estudo de padrões de comunicação coletiva para avaliação do tempo de transmissão de pacotes da MCNoC em relação às arquiteturas de NoCs baseadas nas topologias mais encontradas na literatura (*mesh* e *torus*). Uma ferramenta em Python (analisador de tráfego) foi desenvolvida para que fosse possível ler, analisar e avaliar os traços gerados pela execução dos programas paralelos escritos em MPI (*Message Passing Interface*) em uma máquina real com oito núcleos de processamento. Os programas em MPI do NAS foram compilados com a versão Mpich 1.2.7p1 usando compiladores Intel C e Fortran com otimização O3. Para obtenção dos traços os programas foram executados para 8, 16 e 32 nós de processamento sendo o Mpich instrumentado para que se tenha o controle total do procedimento de comunicação. Desta forma, todas as compilações tiveram habilitadas as opções de traço MPE (*Multi-Processing Environment*). O *benchmark* utilizado é o NPB (NPB – *NAS Parallel Benchmark*) (JIN, 1999) (NPB, 2007). A Tabela 3.1 descreve cada um dos programas do NAS (*Numerical Aerodynamic Simulation*) utilizados para avaliação de desempenho.

A Figura 3.2 ilustra a visualização de exemplos de padrões de comunicação através do *software* Triva (SCHNORR, 2008) desenvolvido durante tese de doutorado do GPPD/UFRGS. O exemplo ilustrado pela Figura 3.2a é de um padrão de comunicação híbrido baseado em topologias anéis e estrela. A Figura 3.2b ilustra o padrão de comunicação mestre / escravo hierárquico. O Triva é utilizado nesta tese para demonstrar o comportamento de cargas de trabalho paralelas aplicadas no contexto da MCNoC. A importância da visualização se deve à constatação e verificação da existência de padrões de comunicação coletiva pelos programas do NPB que podem influenciar no desempenho de *Networks-on-Chip*.

Tabela 3.1: Programas paralelos utilizados do NAS

Nome	Descrição
BT.A	Aplicação CFD (*Computational Fluid Dynamics*) simulada. Programa para resolver equações *Navier-Stokes* baseadas em diferenças finitas para o problema ADI (*Alternating Direction Implicit*), onde os resultados são blocos tri diagonais que são resolvidos seqüencialmente para cada dimensão.
CG.A	Método de Gradiente Conjugado para processar o menor valor de uma matriz não estruturada, esparsa e larga.
EP.A	*Benchmark* paralelo embaraçado que gera pares Gaussianos aleatórios. Estabelece um ponto de referência para pico de desempenho em uma dada plataforma.
FT.A	*Kernel* computacional de uma transformada rápida de Fourier (FFT – *Fast Fourier Transform*) 3D. FT executa três FFT 1D, sendo um para cada dimensão.
IS.A	Programa que testa ordenação de inteiros, importante em códigos para processamento de partículas. Este programa exercita o desempenho da ordenação e da comunicação.
LU.A	Aplicação CFD simulada que usa o método de relaxação sucessiva simétrica (SSOR – *Symetric Sucessive Over-Relaxation*) para sistema resultante da discretização de diferenças finitas de equações *Navier-Stokes* em 3D, pela separação em blocos mais baixos e superiores de sistemas triangulares.
MG.A	Método de ciclo V *multigrid* usado para solucionar equações escalares de Poisson 3D. O algoritmo funciona com *grids* finas e grossas. Ele exercita movimento de dados em distâncias longas e curtas.
SP.A	Aplicação de Dinâmicas de Fluídos Computacionais (CFD) similar ao BT. O problema é baseado em fatorização aproximada *Beam-Warming* em 3D. O sistema penta diagonal escalar resultante é resolvido seqüencialmente ao longo de cada dimensão.

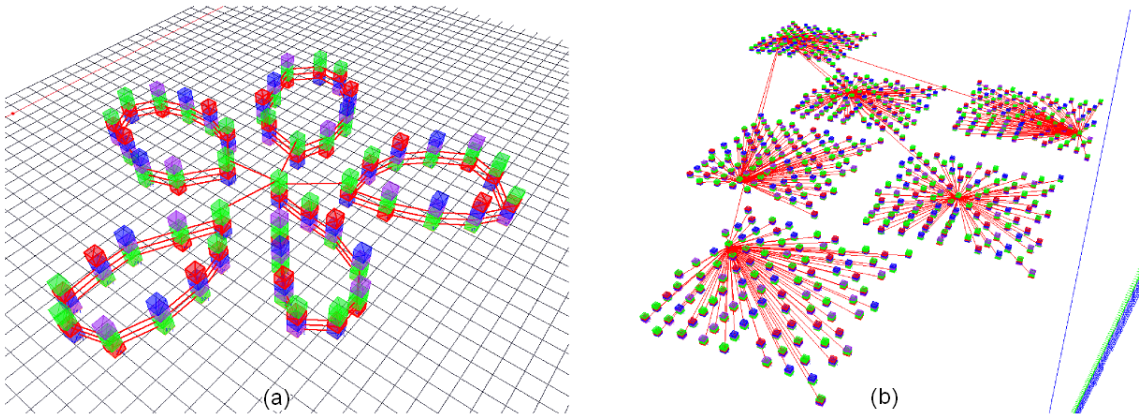


Figura 3.2: Visualização de padrões de comunicação através do Triva (SCHNORR, 2008)

Os modelos analíticos de desempenho são baseados em resultados de latência das simulações dos protótipos de *hardwares* e das características de topologias e arquiteturas das NoCs. Portanto, as etapas da metodologia relacionadas a esta seção são:

14. Especificar os modelos analíticos de desempenho e aplicar às arquiteturas de NoCs em avaliação.

15. Executar carga de trabalho paralela real (NPB) em uma máquina real, coletar e visualizar *logs* (rastros de execuções) no Triva e analisar os padrões de comunicação.
16. Desenvolver um programa em Python que processe os traços do NPB em função das arquiteturas de NoCs e retorne resultados de tempo de transmissão e características das cargas de trabalho.
17. Avaliar o desempenho dos padrões de comunicação do NPB em cada NoC através do programa analisador de tráfego desenvolvido em Python.

4 PROPOSTA DA ARQUITETURA MCNOC

Diferentemente das arquiteturas de NoC convencionais, o projeto de uma NoC com arquitetura baseada em *clusters* de núcleos de processamento objetiva um maior desempenho e vazão de aplicações no *chip*. Um *cluster* de tamanho pequeno também busca explorar o desempenho de soluções mais simples de interconexões (e.g., chave *crossbar*) sem a necessidade de uma maior quantidade de saltos de comunicação. Conseqüentemente, o uso de *clusters* de núcleos de processamento reduz o tráfego entre roteadores e possíveis contenções ou gargalos. Além disso, o acréscimo de núcleos para processamento de uma mesma aplicação sofre uma menor influência da rede e pode aumentar a eficiência e o desempenho do sistema.

A Figura 4.1 ilustra uma arquitetura de *chip multi-cluster* com ênfase na distribuição de aplicações. O *chip* exemplo pode ser definido através das seguintes características: oito *clusters* de núcleos de processamento, oito núcleos por *cluster*, um roteador por *cluster*, dois roteadores intermediários e cinco aplicações representadas por cores/sombreamentos diferentes. O mapeamento das aplicações ocorre em um único *cluster* (e.g., *cluster* 6) ou em *clusters* próximos (e.g., *clusters* 3 e 4), podendo haver compartilhamento de *clusters* (e.g., *cluster* 2). Um *chip* para suporte a *clusters* de alto desempenho está baseado em interconexões rápidas entre núcleos de um mesmo *cluster* e na menor utilização dos roteadores intermediários da rede.

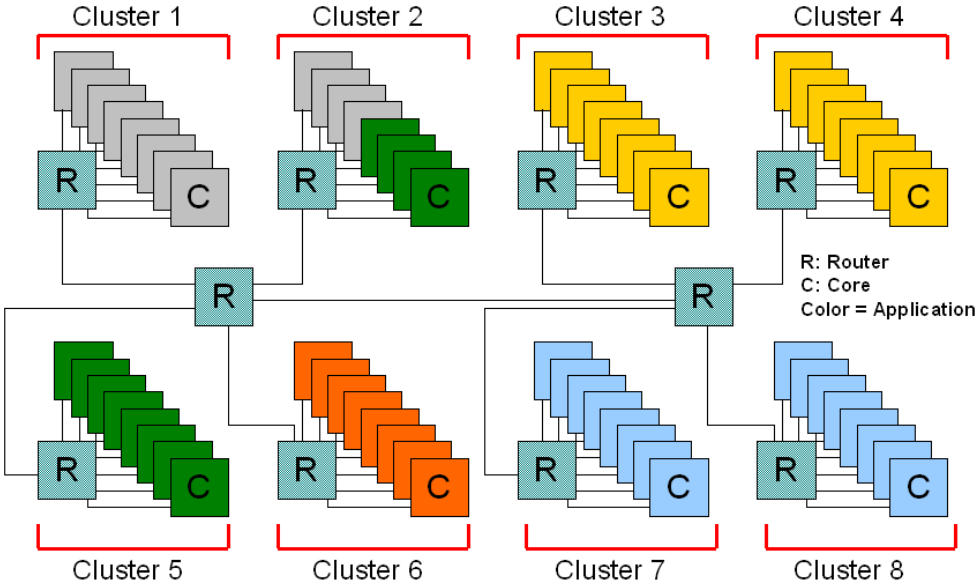


Figura 4.1: Arquitetura de um chip multi-cluster (FREITAS, 2009d)

Apesar do *cluster* ser uma solução interessante, a análise de diferentes tipos de aplicações resulta na identificação de diferentes padrões de comunicação. Além disso, os roteadores de NoC do estado da arte são circuitos dedicados e, portanto, oferecem pouca flexibilidade. Para garantir um melhor gerenciamento do tráfego de rede dentro e fora do *cluster*, é necessário aumentar a flexibilidade do roteador e da própria topologia de rede. Neste sentido, a Figura 4.2 apresenta uma arquitetura de roteador-em-*chip* programável (FREITAS, 2008b), que consiste dos seguintes elementos: um processador de rede chamado NPoC (*Network Processor on Chip*), uma chave *crossbar* reconfigurável chamada RCS-NR (*Reconfigurable Crossbar Switch for NoC Router*) e um conjunto de *buffers* de entrada.

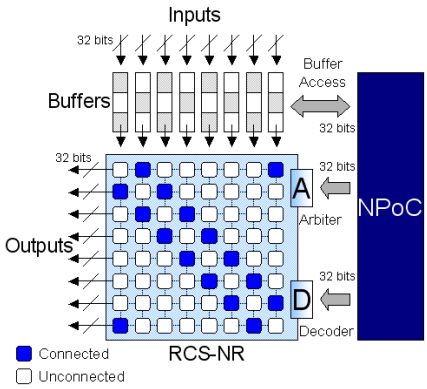


Figura 4.2: Arquitetura de um roteador-em-chip programável (FREITAS, 2009b)

Através dos roteadores programáveis interconectados é possível aumentar a flexibilidade da NoC através de programas que monitoram o tráfego pela chave *crossbar*. A monitoração do tráfego é realizado através dos programas do NPoC que alteram a topologia implementada na chave *crossbar* em função do padrão de comunicação de uma determinada aplicação. A escolha por um roteador programável ao invés de um monitoramento por núcleo (*core*) se deve também à melhor regularidade do projeto. Um roteador gerenciado por um ou mais de um núcleo dificulta a escalabilidade da NoC e gerenciamento por roteadores intermediários, podendo, inclusive, aumentar o tráfego de pacotes responsáveis apenas por gerenciamento.

As seções seguintes descrevem em detalhes os seguintes blocos construtivos da MCNoC (*Multi-Cluster NoC*) em uma visão arquitetural *bottom-up*: Chave *Crossbar* Reconfigurável, Processador de Rede em *Chip*, Roteador Programável em *Chip*, além da própria MCNoC.

4.1 Chave Crossbar Reconfigurável

Os pacotes de rede gerados pelas aplicações seguem um caminho físico que corresponde a um determinado padrão de comunicação entre origem e destino. A chave *crossbar* do roteador programável é o meio de interconexão responsável pela comunicação entre os núcleos de processamento das aplicações. Considerando que os padrões de comunicação mudam em uma mesma aplicação ou entre aplicações diferentes, aumentar a flexibilidade da chave *crossbar* pode melhorar o mapeamento do caminho físico correspondente ao padrão de comunicação e aumentar o desempenho de execução das aplicações. Neste sentido, a aplicação de conceitos de computação reconfigurável tem por objetivo aumentar as opções de topologias implementadas na chave *crossbar* para suporte aos padrões de comunicação.

O projeto da chave *crossbar* reconfigurável para roteadores de NoC é chamado de RCS-NR (*Reconfigurable Crossbar Switch for NoC Router*). O tamanho da chave 8x8 está associado a dois motivos principais: i) uma chave *crossbar* pequena é viável fisicamente, e ii) possui alto desempenho comparado com outras soluções de interconexões. Portanto, de acordo com a proposta da MCNoC, chaves *crossbar* pequenas são utilizadas em roteadores para encaminhar pacotes em, ou entre, *clusters* sem a intrusão do NPoC. Modelos em Redes de Petri (RdP) foram feitos para verificar o desempenho de pequenas *crossbars* em comparação com alternativas de interconexões baseadas em barramento e NoC. Para os modelos em RdP, o chaveamento de uma chave *crossbar* simples para envio de pacotes adiciona um atraso de 1 ciclo. O mesmo atraso pode ser definido para um barramento simples, apenas 1 ciclo. No caso de uma NoC convencional com topologias *mesh* ou *torus*, cada roteador adiciona 3 ciclos de atraso para envio de pacotes. A interface entre os núcleos origem e destino também adiciona atrasos, 1 ciclo para cada interface. Para *chips many-core*, uma única chave *crossbar* ou barramento seria inviável, já que ambos não são escaláveis, haveria um aumento do atraso associado a cada envio de pacotes. No entanto, em um *cluster intra-chip* estas soluções podem apresentar bons desempenho.

Os atrasos referentes à chave *crossbar* e barramento foram identificados em testes de projeto da RCS-NR. Os atrasos referente à NoC foram retirados da SoCIN (*System-on-Chip Interconnection*) (ZEFERINO, 2003). Nesta fase de projeto da RCS-NR, modelos em Redes de Petri foram feitos para verificar apenas o efeito da latência para escolha de uma chave *crossbar* pequena (8x8) em *cluster*. Portanto, cada pacote possui tamanho igual a 32 *bits*, ou seja, a cada ciclo um novo pacote é encaminhado por cada abordagem de interconexão modelada em RdP, e somente latência e caminhos compartilhados influenciam nos resultados de simulação dos modelos.

A Figura 4.3 ilustra um modelo em Redes de Petri para o padrão de comunicação *one-to-one* para avaliação da capacidade de simultaneidade de comunicação e vazão de pacotes. Cada modelo foi feito com base nas limitações ou características de cada abordagem de interconexão, conforme descrição a seguir:

- Barramento (Figura 4.3a): Largura de banda limitada, apenas uma comunicação por janela de tempo. Limitação representada pelo *Place Bus Arbiter*.
- Chave *Crossbar* (Figura 4.3b): Boa largura de banda, número de comunicações simultâneas iguais ao número de nós conectados.
- NoC *torus* (Figura 4.3c): Boa largura de banda, número de comunicações simultâneas iguais ao número de nós conectados.

A Figura 4.4 ilustra o padrão de comunicação *broadcast* modelado em RdP para as três abordagens de interconexões. As características de cada modelo são:

- Barramento (Figura 4.4a): Boa largura de banda, número de comunicações simultâneas iguais ao número de nós conectados.
- Chave *Crossbar* (Figura 4.4b): Boa largura de banda, número de comunicações simultâneas iguais ao número de nós conectados.
- NoC *mesh* ou *torus* (Figura 4.4c): Boa largura de banda, mas a vazão de pacotes é afetada pela quantidade de saltos existentes durante o caminho dos pacotes através do roteamento XY. O modelo representa apenas o caminho

referente ao núcleo mais distante. No entanto, os resultados consideram a influência de todos os atrasos para alcançar este núcleo.

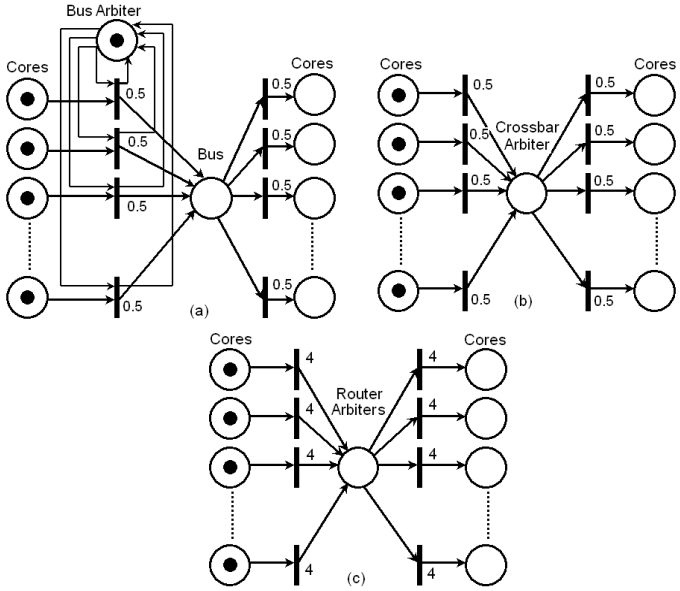


Figura 4.3: Comunicação one-to-one (simultaneidade) (FREITAS, 2008a)

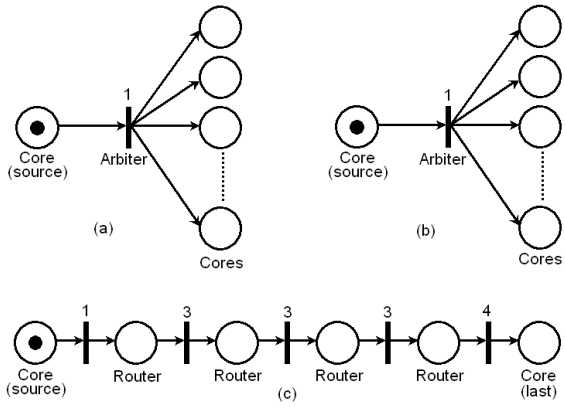


Figura 4.4: Comunicação one-to-all (broadcast) (FREITAS, 2008a)

O padrão de comunicação *all-to-one* modelado em RdP está ilustrado na Figura 4.5 para as três abordagens de interconexões. As características de cada modelo são:

- Barramento (Figura 4.5a): Largura de banda limitada, apenas uma comunicação por janela de tempo. O modelo da Figura 4.5a corresponde ao modelo da Figura 4.3a sumarizado com a latência total para último pacote enviado.
- Chave *Crossbar* (Figura 4.5b): Largura de banda limitada, apenas uma comunicação por janela de tempo. Neste caso, ocorre o número de conflitos máximo por comunicação, por se tratar de uma única saída demandada por todas as entradas. O modelo da Figura 4.5b corresponde ao modelo da Figura 4.3a sumarizado com a latência total para último pacote enviado.
- NoC *mesh* ou *torus* (Figura 4.5c): Boa largura de banda, mas a vazão de pacotes é afetada novamente pela quantidade de saltos existentes durante o caminho dos pacotes através do roteamento XY. O modelo representa apenas o caminho referente ao núcleo mais distante. No entanto, a influência de

todos os atrasos para alcançar este núcleo deve ser considerada. A Figura 4.5c é a Figura 4.4c sumarizada.

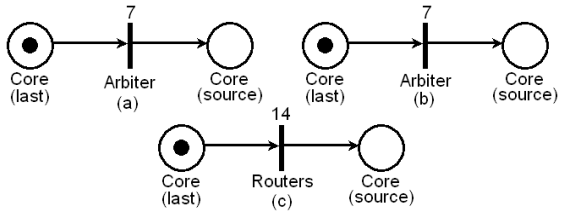


Figura 4.5: Comunicação all-to-one (FREITAS, 2008a)

Os modelos em RdP são responsáveis por uma verificação formal da decisão de uma chave *crossbar* pequena que possua alto desempenho. Para oito núcleos de processamento (oito *places*) a Figura 4.6 mostra que apesar da boa largura de banda de uma NoC *torus*, o efeito da latência imposta por cada roteador da rede reduz consideravelmente a quantidade dos pacotes entregues no mesmo intervalo de tempo. A simulação de cada modelo em RdP possui mil *tokens* iniciais e os períodos de coleta dos dados são de 500 ciclos e 1000 ciclos. Ao final verificava-se a vazão ou número de pacotes enviados. A chave *crossbar* apresentou os melhores resultados para o padrão *one-to-one*, alcançando um ganho de 4 a 8 vezes em relação a NoC e ao barramento, respectivamente. Para *broadcast*, a chave *crossbar* e o barramento alcançam os melhores resultados com um ganho de 4 a 33 vezes em relação as NoCs *torus* e *mesh*, respectivamente. Neste caso, fica ressaltado que o roteamento XY reduz consideravelmente o desempenho das NoCs em *broadcast*. No padrão *all-to-one* todas as abordagens possuem seus piores desempenhos. No entanto, a chave *crossbar* e o barramento alcançam um ganho de até 2 vezes em relação a NoC *torus*. Estes resultados consideram apenas as latências do núcleo mais distante da NoC (melhor caso da NoC) e com este melhor caso a influência do roteamento XY reduz consideravelmente o desempenho das duas versões de NoC.

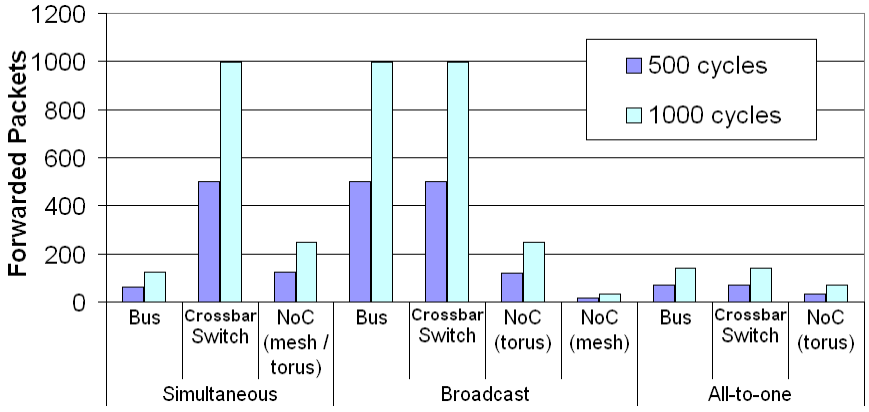


Figura 4.6: Efeito da latência através das simulações de RdPs (FREITAS, 2008a)

A Figura 4.6 apresenta apenas os resultados do efeito da latência considerando o tamanho do pacote com a mesma largura de *bits* de interconexão. Para oito núcleos, estes resultados mostram o melhor desempenho de soluções tradicionais, tais como, barramento e chave *crossbar*, em relação às duas NoCs com topologias *mesh* e *torus*. Considerando a proposta de NoC com *clusters* pequenos de núcleos de processamento, esta primeira avaliação de projeto aponta a chave *crossbar* como uma boa alternativa para o roteador programável em oposição ao barramento e aos vários roteadores dedicados de uma topologia *mesh* ou *torus*.

Como principal característica de projeto da Chave *Crossbar* Reconfigurável (RCS-NR), a Figura 4.7 ilustra a exploração do segundo nível de reconfiguração. O objetivo do segundo nível de reconfiguração é capacitar a chave *crossbar* na implementação de topologias dinamicamente em função da demanda de padrões de comunicação. Sendo assim, o primeiro nível (e.g., FPGA) só seria utilizado para reconfigurações estáticas e prototipação, que não precisam ser feitas em tempo de execução. Uma alternativa é o uso do segundo nível sem a necessidade do primeiro, ou seja, a reconfiguração de topologias seria independente da plataforma de implementação da chave *crossbar*. Por definição, o uso do segundo nível torna a reconfiguração de topologias independente do primeiro, mas o uso do primeiro nível facilita as avaliações do protótipo, devido ao aumento da flexibilidade de expansão ou redução do circuito, re-adaptando inclusive a arquitetura da NoC implementada no FPGA. Esta flexibilidade não é encontrada em PASICs, uma vez que estes não possuem a capacidade de expansão de componentes possível através de FPGAs e, portanto, pelo uso do primeiro nível em complemento ao segundo nível de reconfiguração. Esta integração de níveis é importante nas etapas de projeto e avaliação de protótipos das versões MCNoC.

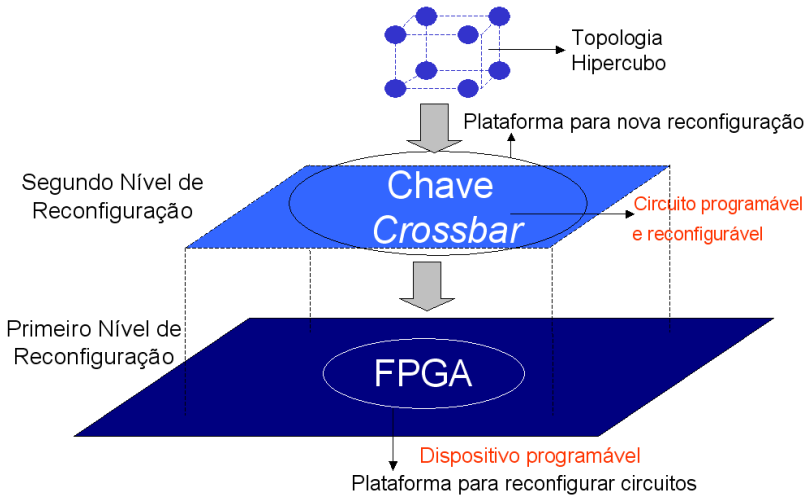


Figura 4.7: Exploração do segundo nível de reconfiguração através de protótipo em FPGA (FREITAS, 2009b)

Uma chave *crossbar* pode ser vista como uma matriz de interconexões. Esta matriz é composta por um conjunto de barramentos paralelos e perpendiculares, sendo que em cada ponto de interseção há uma chave de interconexão que provê a comunicação entre dois pontos. Algumas chaves *crossbar* possuem suporte a *broadcast* e *multicast*, mas todas possuem restrições ao chaveamento simultâneo de conexões entre várias entradas e uma única saída. Neste caso, o controle é responsável por uma seleção temporal de interconexão em função de requisições das entradas. Projetos com *buffers* e decodificadores de *crosspoint* não necessitam de linhas de requisição, mas o controle de chaveamento também é temporal.

A Figura 4.8 apresenta a arquitetura da RCS-NR, os blocos principais e um exemplo de implementação/configuração da topologia anel bidirecional. Neste projeto, todas as interconexões podem ser fechadas ao mesmo tempo, independente de haver conflitos de saída ou não. Esta implementação mapeia a comunicação de uma aplicação em uma topologia configurada na RCS-NR (Figura 4.8a) em função da demanda. Apesar de haver o conflito de saída, a solução adotada mantém a interconexão fechada e trabalha com um ordenamento de prioridade baseada em uma tabela de escalonamento. O

processador de rede do roteador é o responsável pelo gerenciamento tanto da implementação de topologia como do controle de conflitos de saída. Portanto, é possível descrever este gerenciamento atuando na RCS-NR da seguinte forma (Figura 4.8(b)):

- Implementação da topologia: o processador de rede (NPoC), através de um programa em execução, gerencia o tráfego de rede e envia uma palavra de topologia para o decodificador da RCS-NR que decodifica e implementa a topologia fechando os nós de chaveamento. Quanto maior a topologia, maior a complexidade de gerenciamento das saídas, portanto, implementar uma topologia completa na chave *crossbar* pode não ser a melhor alternativa.
- Controle de conflitos de saída: em função do tráfego demandado, o NPoC atualiza uma tabela de entradas de prioridade, que ciclo a ciclo é lida pelo árbitro da RCS-NR. Cada linha da tabela possui uma largura de 32 *bits*, sendo que cada conjunto de 4 *bits* é responsável pela seleção das entradas, liberando ou não os pacotes para uma determinada saída. Por conseqüência do suporte a topologias, a RCS-NR suporta *broadcast* e *multicast*.

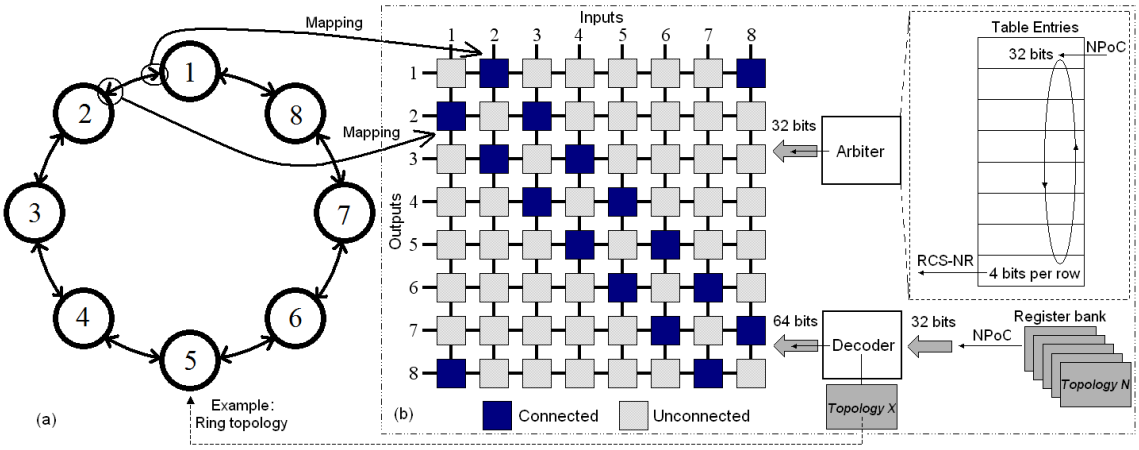


Figura 4.8: Arquitetura da chave crossbar reconfigurável (FREITAS, 2008c)

A RCS-NR possui *buffers* de entrada ilustrados na Figura 4.2. O principal motivo para esta escolha está associado a duas questões:

- Desempenho: os *buffers* de saída resolvem o problema do *head of the line blocking*, mas obrigam a chave *crossbar* ser N vezes mais rápida. A tabela de prioridades da RCS-NR pode ser atualizada em cada novo ciclo e resolver o mesmo problema para um conjunto de *buffers* de entrada.
- Economia de área e potência: os *buffers* e decodificadores de *crosspoint* são eficientes para roteamento, mas aumentam consideravelmente a área e o consumo de potência. Os *buffers* de entrada associados à implementação de topologias e tabela de arbitragem economizam em potência e possuem um alto desempenho.

4.2 Processador de Rede em Chip

O processador de rede em *chip* (NPoC) (FREITAS, 2006b) foi desenvolvido com o intuito de aumentar a flexibilidade do roteador da NoC. Como descrito na seção anterior, a responsabilidade principal é monitorar o tráfego e gerenciar a chave *crossbar* reconfigurável. Uma das diretrizes para o projeto do NPoC é a simplicidade e

regularidade das instruções de acordo com os modelos de processadores RISC (*Reduced Instruction Set Computing*) (PATTERSON, 2005). Por este motivo, o NPoC é inspirado na arquitetura dos processadores MIPS (PATTERSON, 2005), mas com mudanças nos formatos de instruções, organização do caminho de dados e controle, e no *pipeline* de instruções.

A Figura 4.9 ilustra a modificação no quarto estágio do *pipeline* de instruções. Em função da RCS-NR, as instruções do NPoC acessam os dados dos *buffers* de entrada e registradores da chave *crossbar* (BCTU – *Buffers and Crossbar Transfer Unit*), o árbitro (SCH – *Scheduler*), decodificador de reconfiguração (REC – *Reconfiguration*), além do acesso normal à memória (MEM). Os demais estágios seguem o modelo tradicional de *pipeline*:

- IF (*Instruction Fetch*): Busca de instruções.
- ID (*Instruction Decoder*): Decodificação de instruções.
- EX (*Execution*): Execução de instruções.
- WB (*Write Back*): Escrita no banco de registradores

A subseção seguinte descreve a arquitetura do NPoC.

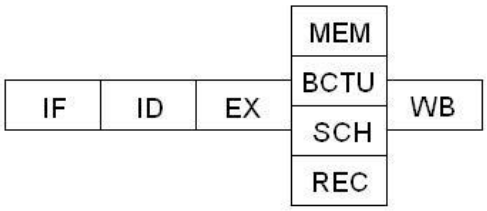


Figura 4.9: Estágios de pipeline do processador de rede (FREITAS, 2006b)

4.2.1 Arquitetura e Conjunto de Instruções do NPoC

A Figura 4.10 apresenta a organização interna dos blocos construtivos do NPoC. Através desta figura é possível descrever o caminho de dados de todas as instruções projetadas com alguns sinais de controle mais significativos, tais com o J (*jump* – desvio) e o controle dos MUXs de entrada da ALU (*Arithmetic and Logic Unit*).

O NPoC é um processador RISC, portanto, tem instruções de formato regular e fixo e com execução em apenas 1 ciclo quando o *pipeline* está cheio. No entanto, existem dependências de dados e de controle que podem impossibilitar que o *pipeline* fique cheio todo o tempo. A alternativa para reduzir o efeito destas dependências foi o adiantamento de dados (Bloco *Forwarding* – terceiro estágio) capaz de aumentar o número de vezes que o *pipeline* permanece cheio. No caso das instruções de desvio, quando de um desvio verdadeiro, as instruções presentes no *pipeline* são descartadas ocasionando bolhas. A previsão de desvio não está implementada e faz parte dos trabalhos futuros. Um exemplo do adiantamento de dados para uma instrução de desvio, descrito em ArchC, está ilustrado na Figura 4.11.

O exemplo da Figura 4.11 pode ser expandido para todas as instruções, ressaltando que em alguns casos o adiantamento é necessário para os registradores r1 e r3, além do r2. A descrição em linguagem de arquitetura ArchC foi realizada tendo como referência toda organização da Figura 4.10.

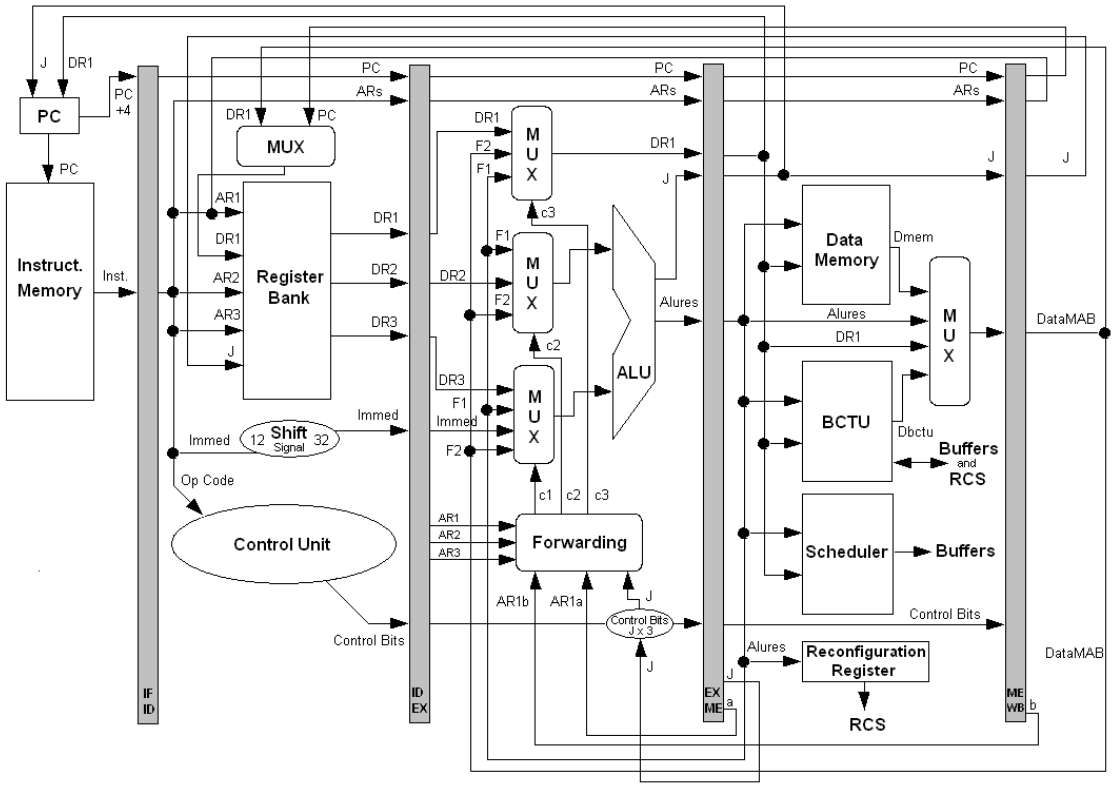


Figura 4.10: Arquitetura do processador de rede em chip (FREITAS, 2006b)

```
//forwarding for j (instruction branch)
if ( C_value == 0 ){
 //forwarding for r2
 if ( (EX_MEM.regwrite == 1) && (EX_MEM.r1 != 0 ) &&
(EX_MEM.r1 == ID_EX.r2) )
   ID_EX.data2 = EX_MEM.alures.read();
 else if ( (MEM_WB.regwrite == 1) && (MEM_WB.r1 != 0 ) &&
(MEM_WB.r1 == ID_EX.r2) )
   ID_EX.data2 = MEM_WB.datamab.read();
 else
   ID_EX.data2 = RB.read(r2); }
else {
  ID_EX.data2 = RB.read(r2);
  ID_EX.regwrite = 0; }
```

Figura 4.11: Adiantamento de dados da instrução jump em ArchC (FREITAS, 2006b)

O formato das instruções é especificado pela Figura 4.12. A palavra do NPoC é de 32 *bits* e o campo de *opcode* (*operation code*) corresponde a 6 *bits* para todas as instruções. Os últimos campos de *bits* com um número seguido por *x* não são utilizados pelas instruções. Sendo assim, instruções com três operandos registradores não utilizam os cinco últimos *bits* da palavra. Instruções que possuem um operando com valor imediato (*imed*) utilizam o bloco *shift signal* da arquitetura apresentada pela Figura 4.10, que é responsável por deslocar o sinal até o trigésimo segundo *bit* da palavra para que possa ser utilizada pela ALU. A largura de 7 *bits* dos campos de registradores possibilita um endereçamento de até 128 registradores visando o suporte a contextos de múltiplas *threads*. Neste caso, o ISA não precisa mudar com a necessidade do suporte a mais de uma *thread*, já que este ISA escala, por exemplo, até 4 bancos de 32

registradores. Registradores com endereços diferentes possibilitam acessos compartilhados, por *threads* diferentes, a dados no próprio banco de registradores. A Figura 4.13 ilustra dois tipos de compartilhamentos para possibilidades de gerenciamentos concorrentes no NPoC. A Figura 4.13a apresenta 4 bancos compartilhados por 4 *threads*, e a Figura 4.13b apresenta 1 banco compartilhado por 3 *threads* ativas. O projeto e avaliação do gerenciamento concorrente fazem dos trabalhos futuros, e estão associados a um ambiente de simulação baseado em SystemC e ArchC (Seção 6.1). Mais de um banco de registradores também pode ser útil no caso de armazenamento temporário de pacotes, conforme está descrito na avaliação de resultados da Seção 5.3.

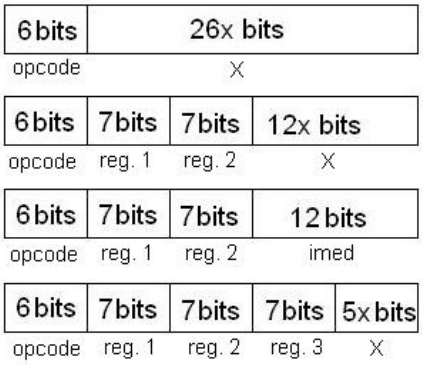


Figura 4.12: Formato de instruções do processador (FREITAS, 2006b)

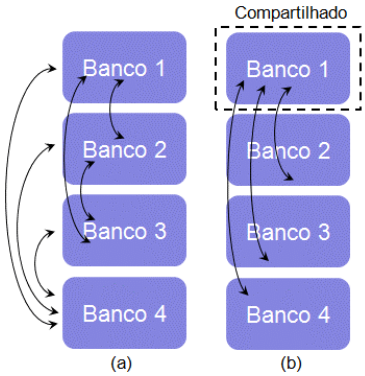


Figura 4.13: Bancos de registradores compartilhados. (a) 4 threads, 4 bancos compartilhados, (b) 3 threads, 1 banco compartilhado

A escolha das instruções está baseada em dois aspectos: i) Instruções de propósito geral para operações aritméticas e lógicas, desvios, e acessos à memória. ii) Instruções de rede específicas para acesso aos demais blocos do roteador RANoC. As instruções de rede são tão simples quanto a regularidade de um projeto RISC demanda. Portanto, não foi projetada nenhuma instrução complexa que processe um protocolo específico. Conseqüentemente, as instruções de rede são para manipulação de registradores dedicados dos demais blocos do RANoC.

A Tabela 4.1 ilustra algumas instruções de propósito geral. Existem outras instruções que não foram apresentadas e que seguem o mesmo tipo de descrição tais como: *sub, and, or, nor, nand, div, xori, addi, subi, ori, andi, jdi, jle, shiftr* e *shiftl*.

A Tabela 4.2 apresenta instruções de rede com formatos iguais às de propósito geral, porém com acessos a periféricos diferentes. Estas instruções utilizam a ALU apenas para cálculo de endereço, mas o propósito principal é o acesso aos periféricos *buffers* e

chave *crossbar*. As instruções *read* e *write* acessam os *buffers* e registradores da chave *crossbar* através da unidade de transferência (BCTU: *Buffers and Crossbar Transfer Unit*). As instruções *send, block* e *erase* especificam para o escalonador quais pacotes devem ser enviados, bloqueados ou apagados, respectivamente. A instrução *reconf* é usada para transferência de dados entre um registrador de propósito geral e o registrador de reconfiguração da chave *crossbar*. Este registrador de reconfiguração pertence ao decodificador da RCS-NR que implementa a topologia necessária.

Tabela 4.1: Exemplos de instruções de propósito geral (FREITAS, 2006b)

Instruções	Descrição
add r1, r2, r3	r1 <= r2 + r3
mul r1, r2, r3	r1 <= r2 * r3
ori r1, r2, imed	r1 <= r2 + imed
not r1, r2	r1 <= r2’
load r1, r2, imed	r1 <= conteúdo (End[r2 + imed])
store r1, r2, imed	End[r2 + imed] <= r1
jump r1, r2, imed	PC <= r2 + imed, r1 <= PC
jeq r1, r2, r3	PC <= r1, se r2 = r3

Tabela 4.2: Instruções de rede (FREITAS, 2006b)

Instruções	Descrição
read r1, r2, imed	r1 <= conteúdo (End[r2 + imed])
write r1, r2, imed	End[r2 + imed] <= r1
send r1, r2, imed	Buffer (r1), enviar pacote (r2 + imed)
block r1, r2, imed	Buffer (r1), bloquear pacote (r2 + imed)
erase r1, r2, imed	Buffer (r1), apagar pacote (r2 + imed)
reconf r1	Registrador de Reconfiguração <= r1

As instruções *send, block* e *erase* são necessárias principalmente quando o roteador trabalhar com encaminhamento de pacotes no modo *store-and-forward*, o que tecnicamente é possível sendo programável. O mesmo se aplica às instruções *read* e *write*. No entanto, para compatibilidade entre o NPoC e RCS-NR, no suporte a três tipos de instruções de escalonamento, a tabela de arbitragem deveria suportar cinco *bits* por linha (saída), três reservados para a seleção de entradas e dois para indicação das instruções. O atual projeto da RCS-NR suporta as instruções *send* e *block* para gerenciamento de conflitos de saída. Portanto, a solução para apagar um pacote é realizar uma conexão na RCS-NR e usar a instrução *send* para descarte de pacote. Técnicas de encaminhamento e tipos de protocolos são descritos nas seções seguintes.

4.2.2 Suporte a Interleaved Multithreading

Com o objetivo de aumentar a vazão de processamento e desempenho do roteador, foi projetado para a arquitetura do NPoC o suporte a quatro *threads* através da técnica *Interleaved Multithreading* (IMT). Processadores de Rede *off-chip* exploram o

paralelismo de *thread* para aumentar, primeiramente, a vazão de pacotes e depois o desempenho de aplicações que freqüentemente possuem seus contextos chaveados. Através do IMT, as grandes latências, tais como acessos à memória, influenciam menos na execução de outras instruções da mesma *thread*, já que no ciclo seguinte uma instrução de outra *thread* está em processamento. O principal motivo para não escolha da técnica BMT (*Blocked Multithreading*) está associado à latência de esvaziamento de *pipeline*, que é maior do que IMT. Em relação ao SMT (*Simultaneous Multithreading*), existe um custo muito alto (área e potência), devido à replicação de unidades funcionais do *pipeline*, além de problemas de largura de banda de acesso à memória, pela execução paralela de *threads*.

A Figura 4.14 ilustra o comportamento de execução das *threads* no *pipeline* do NPoC após o projeto do suporte a IMT. As quatro primeiras instruções no *pipeline* (direita para esquerda) são de *threads* diferentes. Somente no quinto ciclo busca-se uma nova instrução da *Thread 0* (T0). Este processo se repete, e a cada novo ciclo uma instrução de *thread* diferente da anterior é levada para o *pipeline*. As bolhas, que poderiam ser geradas por instruções de desvio, acesso à memória ou rede, são substituídas pelas execuções de outras *threads*. Esta otimização do *pipeline* aumenta a vazão de instruções por *threads* diferentes, possibilitando resultados e tomadas de decisões mais rápidas.

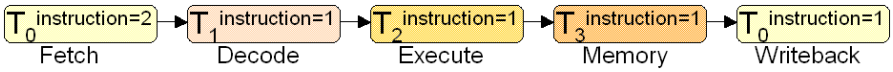


Figura 4.14: Fluxo de instruções do NPoC no pipeline IMT (FREITAS, 2009d)

As alterações no *pipeline* do NPoC estão na replicação (x4) dos bancos de registradores e contadores de programa (PC – *Program Counter*) para suportar 4 *threads*, conforme ilustrado pela Figura 4.15. Cada contador de programa possui a posição de execução de uma *thread* e cada banco de registradores guarda o contexto de uma *thread*. O chaveamento entre *threads* ocorre sem atraso de mudança de contexto, já que os contextos estão nos bancos de registradores e não precisam ser acessados na memória.

A verificação do funcionamento e desempenho do conjunto de instruções do NPoC foi feito com base nos modelos funcionais e de precisão de ciclos descritos em ArchC. Para o suporte a IMT foi realizada uma modificação no código VHDL do processador Plasma utilizado para síntese, com o objetivo de verificar a utilização de componentes e, portanto, da ocupação no FPGA dos novos blocos para suportar múltiplas *threads*.

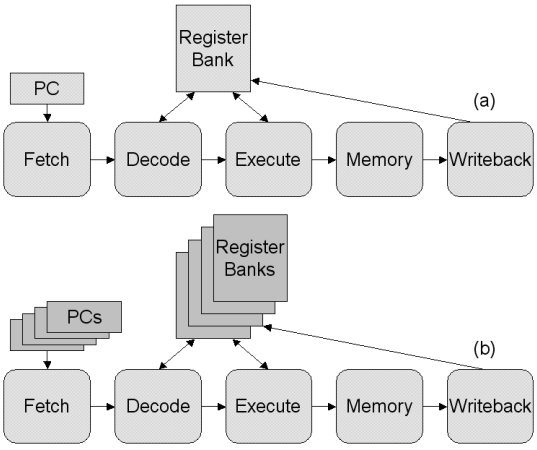


Figura 4.15: Alterações no NPoC para suporte a IMT (FREITAS, 2009d)

Para o processamento de pacotes em processadores de rede (SHAH, 2001) (COMER, 2003), existem duas abordagens: *slow path* e *fast path*. O *slow path* diz respeito ao caminho mais lento para processamento, o qual depende da intervenção do processador para processamento do pacote. Neste caso, o processador de rede retarda o envio do pacote para leitura e processamento do mesmo. O caminho *fast path* é o mais rápido, e o pacote é encaminhado sem a intervenção do processador. Neste caso, circuitos dedicados são acionados no caminho para encaminhamento rápido sem processamento do pacote. O projeto do NPoC também segue estas duas abordagens. Os pacotes normalmente são encaminhados pela chave *crossbar* reconfigurável sem intervenção do NPoC, através do caminho rápido (*fast path*). No entanto, se necessário, o NPoC pode intervir e processar um pacote, atrasando seu envio. Neste caso, o pacote segue um caminho mais lento (*slow path*), passando o processamento do cabeçalho ou conteúdo do pacote pelos registradores do NPoC.

4.3 Roteador Programável

A Chave Crossbar Reconfigurável (RCS-NR) e o Processador de Rede em Chip (NPoC) são os blocos principais do roteador programável chamado RANoC (*Router Architecture for NoC*) (FREITAS, 2008b). A Figura 4.16 ilustra a arquitetura do RANoC para um *cluster* de oito núcleos de processamento. O acesso do NPoC à RCS-NR é feito através de três interfaces (*Buffer Access*, *Arbiter* e *Decoder*) e um conjunto de instruções de rede. O bloco de acesso aos *buffers* (*Buffer Access*) não foi apresentado na Seção 4.1, mas faz parte da RCS-NR. Este bloco é responsável pela interface entre os pacotes presentes nos *buffers* de entrada e o NPoC.

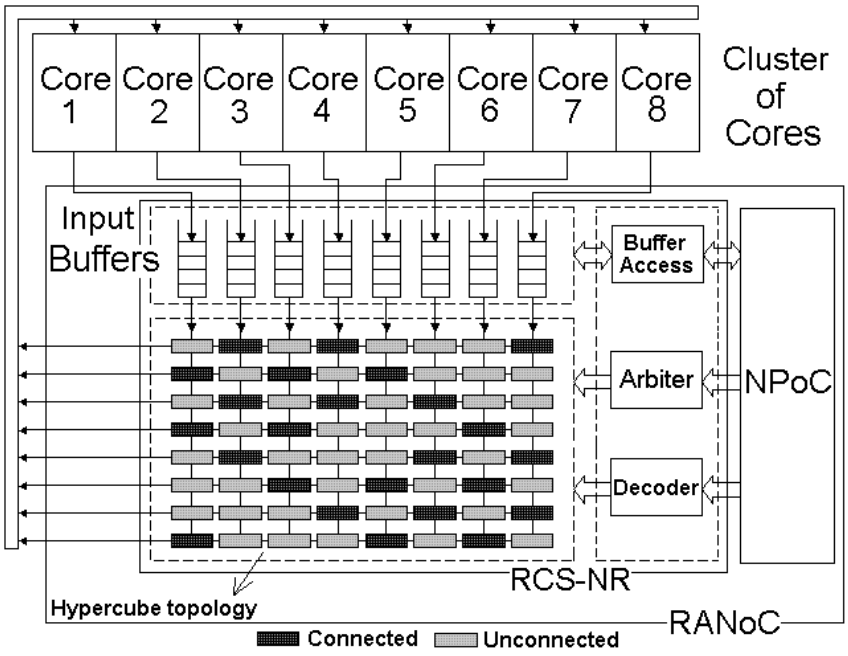


Figura 4.16: Arquitetura do roteador em chip programável (FREITAS, 2008c)

Possibilitar o acesso aos *buffers* de entrada também cria condições para que o roteador seja ativo, possa não apenas rotear pacotes, mas processar o conteúdo dos pacotes. Esta condição é possível sem retirar os pacotes dos *buffers*, através das instruções *read* and *write*. A grande vantagem do RANoC está baseada em duas características para o aumento da flexibilidade:

- Programação suportada pelo NPoC. Execução de programas para monitoramento e aumento do desempenho da rede.
- Reconfiguração suportada pela RCS-NR. Implementação de topologias para adaptação a novos padrões de comunicação.

Portanto, o gerenciamento do NPoC, no que diz respeito aos padrões de comunicação, deve ocorrer de forma que seja possível a identificação do padrão e a implementação da topologia na RCS-NR. A identificação do padrão é feita através de duas abordagens ilustradas pela Figura 4.17.

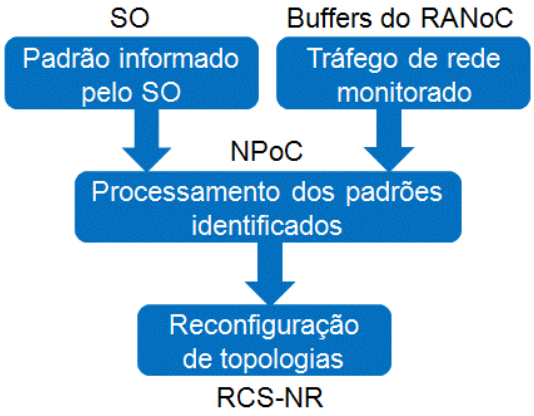


Figura 4.17: Abordagens para identificação de padrões de comunicação

A primeira abordagem está associada à informação fornecida pelo Sistema Operacional (SO) que gerencia os processos. Neste caso, ao mapear os processos em *clusters* específicos, o SO também repassa ao roteador as informações necessárias de comunicação coletiva para que seja mapeada a topologia em conformidade com a demanda de tráfego. É importante ressaltar que a informação repassada pelo SO faz parte de um trabalho em conjunto que deve ser feito em partes pelo Compilador. Em tempo de compilação, padrões de comunicação devem ser identificados e repassados ao SO para que este faça a associação ao mapear os processos. Portanto, as informações do SO são baseadas em processos do SO em execução. Estes processos enviam pacotes para iniciar o mapeamento das topologias em função dos padrões de comunicação das aplicações, ou para implementação de uma nova topologia em função de algum evento externo ou interrupção.

A segunda abordagem está relacionada ao monitoramento de tráfego feito pelo próprio roteador. Neste caso, o NPoC acessa periodicamente os *buffers* de entrada para identificação de requisições de comunicação não mapeadas. As requisições podem ser para uma única conexão ou para uma topologia inteira. Independente do tipo de requisição, a topologia é mapeada espacialmente, conforme descrito na Seção 4.1. A concorrência pelos dois tipos de abordagem pode acontecer, neste caso tem prioridade a informação fornecida pelo Sistema Operacional. O motivo se deve ao fato de que os pacotes de comunicação podem estar desatualizados em relação a eventos gerenciados pelo Sistema Operacional.

A Figura 4.18 apresenta um modelo em Redes de Petri (RdP) para verificação do funcionamento entre as duas abordagens. Em complemento ao modelo duas listas são necessárias: Lista de condições (Tabela 4.3) e Lista de ações (Tabela 4.4). Estas listas fornecem os nomes e as descrições de cada *Place* e Transição do modelo. O objetivo do modelo em RdP é ilustrar a identificação paralela dos padrões de comunicação e a

concorrência pelo recurso Decodificador, responsável por decodificar a palavra de topologia, fornecida pelo SO ou monitoramento, em *bits* de reconfiguração para a RCS-NR. Esta concorrência está modelada por um conjunto de conflitos representados pelas Transições T3, T4, T5 e T6. Os *Places* de entrada P3, P4 e P5 recebem somente um *token* por vez, e isto é insuficiente para disparar todas as transições. A escolha da transição de disparo pode acontecer aleatoriamente ou através de funções *if* / *else*. Estas funções são importantes para priorizar a abordagem de identificação de padrão fornecida pelo SO, conforme está ilustrado pela Figura 4.19. A periodicidade para identificação de padrões não é determinística, mas os *tokens* nos *Places* P1 e P2, as realimentações vindas das Transições T3, T4, T5 e T6 e a marcação inicial {1,1,0,1,0,0,0} são necessárias para que a RdP tenha vivacidade (seja livre de *deadlocks*) e tenha alcançabilidade (todas as marcações possíveis).

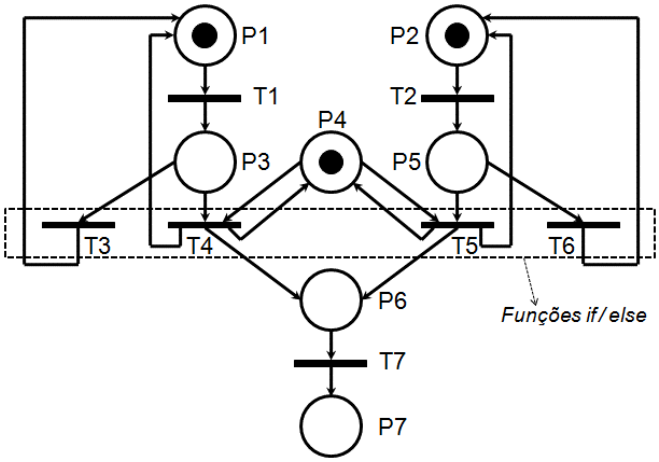


Figura 4.18: Modelo em RdP para identificação de padrões de comunicação

Tabela 4.3: Lista de condições do modelo em RdP

Place	Descrição
P1	Pacotes de requisição do Sistema Operacional.
P2	Pacotes de comunicação coletiva.
P3	Processador recebe informação de padrão de comunicação.
P4	Gerenciador de identificação de padrões.
P5	Processador processa tráfego nos *buffers* de entrada.
P6	Decodificador com palavra de topologia.
P7	Topologias reconfiguradas.

Tabela 4.4: Lista de ações do modelo em RdP

Transição	Descrição
T1	Enviar pacote de requisição.
T2	Enviar pacotes de comunicação.
T3	Descartar informação de topologia baseada no Sistema Operacional.
T4	Enviar palavra de topologia baseada na informação do Sistema Operacional.
T5	Enviar palavra de topologia baseada na informação do tráfego de requisições.
T6	Descartar informação de topologia baseada no tráfego de pacotes.
T7	Reconfigurar topologia.

A Figura 4.19 ilustra as funções definidas para o conjunto de conflitos modelados na RdP da Figura 4.18. O *Place* P3 que representa informação vinda do SO tem prioridade em relação ao *Place* P5 (tráfego). Neste caso a transição T4 dispara para envio da palavra de topologia baseada na informação do SO e se houver *token* em P5, há o disparo de T6 para descarte da informação de monitoramento, e P6 recebe *token* oriundo de P3. A mesma analogia pode ser feita quando não há *token* em P3 e a função entra no *else*. Tanto para o primeiro *if* quanto para o *else*, existem mais duas condições de disparos para T6 ou T3. Estas condições são necessárias caso um *token* seja recebido em P5 ou P3 depois das primeiras condições.

```
If P3 = token then
      T4 dispara
      If P5 = token then T6 dispara
      P6 recebe token
Else
      If P5 = token then
            T5 dispara
            If P3 = token then T3 dispara
            P6 recebe token
```

Figura 4.19: Funções if / else de prioridade na identificação de padrões de comunicação

Do ponto de vista de identificação de padrões de comunicação coletiva, a abordagem mais complexa para o roteador é baseada no monitoramento do tráfego de pacotes nos *buffers* de entrada, uma vez que a outra abordagem possui uma complexidade baseada em processo do SO em execução, sendo mais simples, neste caso, processar um pacote de requisição. Portanto, a Figura 4.20a ilustra um algoritmo que faz a varredura nos *buffers* de entrada em busca destas informações que representam requisições por comunicação do SO. Estas requisições enviam palavra de topologia baseada em conteúdo de pacote para o Decodificador. A outra abordagem (Figura 4.20b) está baseada na varredura de todos os *buffers* até que se encerrem as comunicações, ou que todos os pacotes estejam requisitando novo padrão de comunicação. Neste caso, o programa de gerenciamento atualizaria o Decodificador com uma nova palavra de topologia que já se encontra no banco de registradores, por informação baseada na inicialização do sistema, ou que tenha sido enviado previamente por pacotes da aplicação em processamento.

```
i = 1
Enquanto (1)
  Ler buffer [i]
  Se pacote requisita conexão
  então
    Enviar palavra de topologia
  i = i + 1
  Se i = 9 então
    i = 1
Fim enquanto
```

(a)

```
i = 1 e j = 0
Enquanto (1)
  Ler buffer [i]
  Se fim de comunicação então
    Limpar buffer [i]
    j = j + 1
  i = i + 1
  Se i = 9 então i = 1
  Se j = 8 então
    Enviar palavra de topologia
    j = 0
Fim enquanto
```

(b)

Figura 4.20: Algoritmos para identificação de tráfego em oito buffers. (a) 1 pacote do SO requisita topologia, (b) todas as comunicações encerradas

Apesar do roteador possuir um processador de rede (NPoC), este processador não é intrusivo, ou seja, não insere latência para o encaminhamento de pacotes no modo *fast path*. Todos os pacotes trafegam pela chave *crossbar* reconfigurável e não passam pelo

NPoC. As exceções existem quando os pacotes precisam ser processados pelo roteador, o que atrasaria o envio dos mesmos pela rede (*slow path*).

Com relação à Tabela de Arbitragem, por padrão não é dada prioridade entre as comunicações coletivas com conflito de saída, sendo aplicado o algoritmo *Round Robin* para que a cada novo ciclo uma entrada concorrente tenha acesso a saída. Esta representação é feita pela alternância dos campos nas entradas da tabela. No entanto, pode ser dada prioridade para situações, como por exemplo, em que processos comunicantes oriundos do Sistema Operacional demandam a utilização da rede. Neste caso, o campo referente à comunicação com prioridade é repetido em todas as entradas da tabela até que a comunicação se encerre, voltando a intercalação entre os processos normais (sem prioridade).

4.4 Arquitetura da MCNoC

Apesar dos oito núcleos exemplificados na Figura 4.16, interconectar roteadores significa perder entradas para compor uma topologia de rede. A Figura 4.21 ilustra uma topologia estrela entre RANoCs. Quatro RANoCs são responsáveis pelo gerenciamento de *clusters* de processamento e um RANoC é o roteador central para interligação dos demais.

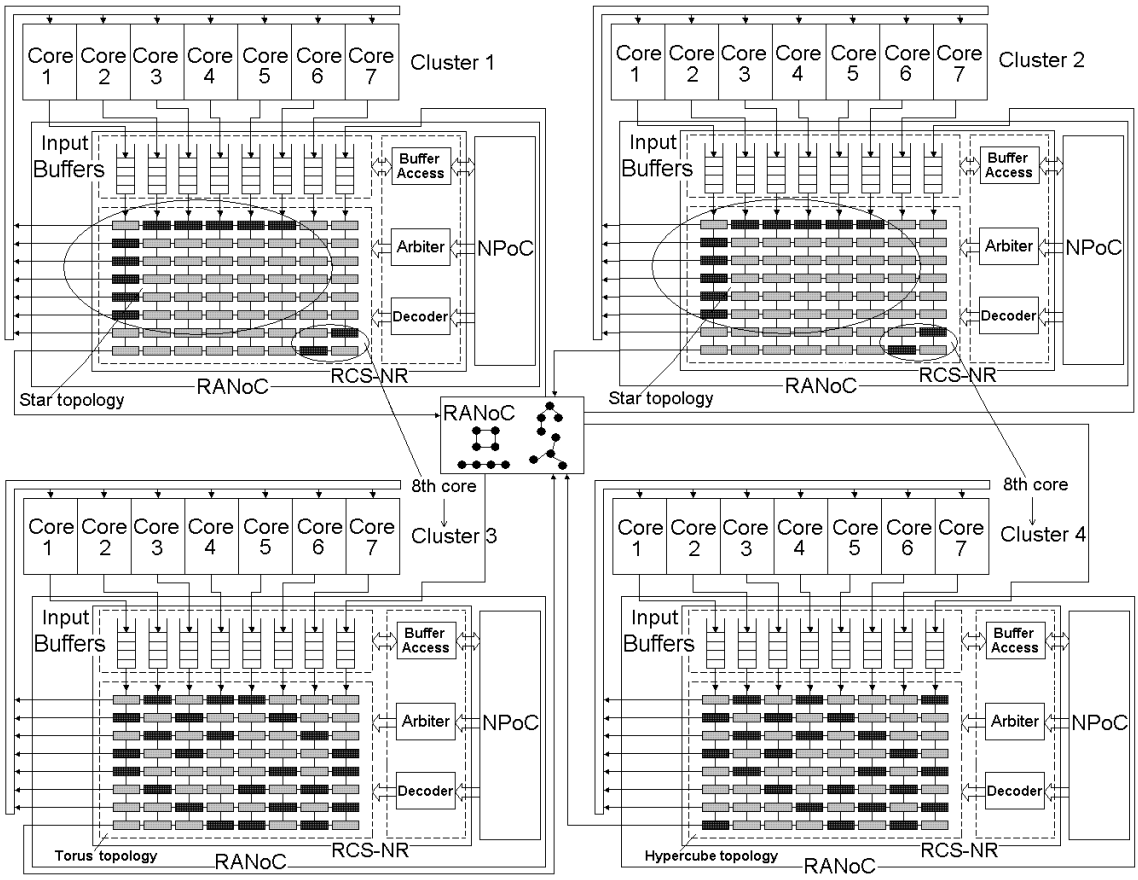


Figura 4.21: Arquitetura da Multi-Cluster NoC (FREITAS, 2008c)

Neste exemplo, a MCNoC (Multi-Cluster NoC) suporta dinamicamente a reconfiguração de topologias regulares da seguinte forma:

- Os roteadores dos *clusters* suportam topologias para comunicação entre núcleos de um mesmo *cluster*. Nos exemplos da Figura 4.21 são suportadas

as topologias estrela para o *cluster* 1 e 2, *torus* no *cluster* 3 e hipercubo no *cluster* 4. Estas topologias refletem os padrões de comunicação de possíveis aplicações em cada *cluster* e podem mudar se o padrão também mudar.

- O roteador central suporta a topologia de interligação dos demais *clusters*. Esta topologia também representa um padrão de comunicação e pode significar o acréscimo de núcleos de outros *clusters* para um determinado *cluster*. Por exemplo, núcleos dos *clusters* 1 e 2 são usados para compor os *clusters* 3 e 4, respectivamente. Através do roteador central é possível suportar topologias, tais como: estrela, árvore, *pipeline* ou anel.

O roteador central é importante para aumentar a flexibilidade de implementação de topologias entre os *clusters*. A MCNoC se baseia no uso de roteadores centrais ou intermediários para aumentar a capacidade e o suporte dinâmico de topologias, através da reconfiguração de segundo nível, em função dos padrões de comunicação. Portanto, a MCNoC pode ser considerada uma rede indireta caso os roteadores centrais não tenham núcleos a eles interconectados. Esta característica depende da implementação da MCNoC feita, por exemplo, no FPGA (primeiro nível de reconfiguração).

4.4.1 Protocolos de Rede

A arquitetura e estratégias adotadas no projeto da MCNoC estão relacionadas aos padrões de comunicação. Neste caso, as rotas consistem de topologias mapeadas para reduzir o número de saltos e a influência dos roteadores. Portanto, o algoritmo de roteamento tem um comportamento adaptativo (padrões de comunicação), já que está baseado na execução de programas. A topologia baseada no padrão de comunicação é implementada pelo RANoC através de monitoramento de rede, mas nada impede que um pacote solicite um novo padrão ou nova interconexão. Conseqüentemente, as topologias são baseadas na implementação / chaveamento de circuito.

Embora o chaveamento por pacote seja largamente usado pela maioria das NoCs convencionais, a MCNoC tem um chaveamento híbrido. O chaveamento por circuito está baseado na implementação de topologias, reduzindo o número de saltos e a influência dos roteadores para núcleos de *clusters* próximos, favorecendo o processamento de aplicações paralelas. No entanto, para *clusters* distantes e processamento de diversas aplicações distribuídas, o chaveamento de pacotes por roteadores intermediários pode aumentar as opções de roteamento, reduzindo o tráfego e aumentando o desempenho.

O roteador RANoC foi projetado com um *buffer* de entrada para cada porta para guardar pacotes por um pequeno período de tempo usando a técnica de encaminhamento *wormhole*. Para que pacotes sejam transmitidos, um algoritmo simples de *handshake* foi implementado para cada porta de entrada, permitindo que o roteador destino (receptor) possa informar se está com espaço livre no *buffer* ou não. A Figura 4.22 ilustra o protocolo *handshake* de duas vias implementado nos roteadores. O roteador transmissor envia um sinal de validação (Val) de 1 *bit* solicitando uma resposta de confirmação (Ack - *Acknowledgement*) (sinal de 1 *bit*), informando liberação (Ack = 1) ou não do *buffer* (Ack = 0).

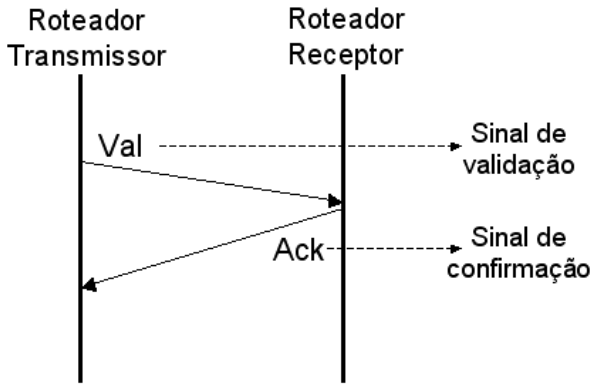


Figura 4.22: Handshake de duas vias

A Figura 4.23 apresenta o cabeçalho do pacote de rede para roteadores e núcleos de processamento. Decisões baseadas em programas dos roteadores usam a informação do cabeçalho do pacote para definir uma nova topologia ou para processar corretamente o conteúdo do pacote. Este cabeçalho foi definido com base nas características da rede proposta organizada em *clusters* de núcleos de processamento. Portanto, cada campo deste cabeçalho possui o seguinte significado:

- *Core (source)*: Campo de 3 *bits* para definir o núcleo origem do pacote.
- *Cluster (source)*: Campo de 7 *bits* para definir o *cluster* origem do pacote.
- *Core (Dest.)*: Campo de 3 *bits* para definir o núcleo destino do pacote.
- *Cluster (Destination)*: Campo de 7 *bits* para definir o *cluster* destino do pacote.
- *Total length*: Campo de 12 bits que informa o tamanho total do pacote (cabeçalho e *payload*) em *bytes*. No máximo 4096 *bytes*.

Este protocolo suporta até 128 *clusters* e 1024 núcleos, mas estes números podem diminuir já que dependem de decisões de topologias (interconexões entre roteadores).

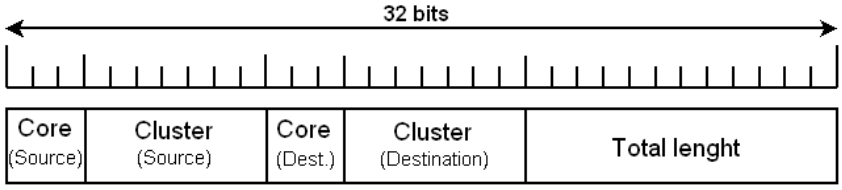


Figura 4.23: Cabeçalho de pacote de rede (FREITAS, 2008c)

A definição por um cabeçalho com cinco campos está relacionada ao fato de que um protocolo leve ou simples seja mais adequado para o processamento do roteador da NoC. No entanto, para situações em que o chaveamento por pacotes esteja sendo utilizado, principalmente em roteadores intermediários, é necessário o uso de um campo chamado *Time to Live*. Este campo é alocado nos próximos 32 bits de cabeçalho, possibilitando uma variação no tempo de vida, definida pela origem, em função da distância do roteador destino. A cada salto, os roteadores decrementam o campo para evitar que o pacote fique vagando pela rede indefinidamente e permaneça em uma situação de *livelock*.

Uma das características do cabeçalho da Figura 4.23 são os campos que indicam roteamento em dois níveis baseados em *core* e *cluster*. A Figura 4.24 ilustra a estrutura

baseada em uma hierarquia de dois níveis onde o roteador central também possui núcleos interconectados (MCNoC direta). Portanto, o roteamento entre roteadores é baseado na informação dos campos *cluster*, enquanto que internamente aos *clusters*, os campos *core* são utilizados. Isto reduz o tamanho da tabela que cada roteador deve gerenciar, uma vez que cada tabela possui informação máxima de 128 *clusters* ao invés de 1024 núcleos. A construção de cada tabela pode ser realizada no momento da implementação da topologia no FPGA (dispositivo de prototipação), ou durante a inicialização da rede através da primeira troca de pacotes entre roteadores. Neste caso, um algoritmo conhecido que pode construir as tabelas em cada roteador é o Vetor de Distância (TANENBAUM, 2003).

Com relação ao gerenciamento de *deadlocks*, sempre haverá uma saída conectada na RCS-NR evitando a situação cíclica dentro de um *cluster*, mesmo que seja uma saída apenas para descartar o pacote. No caso do envio cíclico entre roteadores / *clusters*, o campo *Time to Live* também soluciona o problema. Para finalizar, o *starvation* é solucionado pela Tabela de Arbitragem. Esta tabela é responsável por disponibilizar recursos (saídas), sem que uma comunicação permaneça indefinidamente em detrimento às demais concorrentes pela mesma saída (situação de conflito).

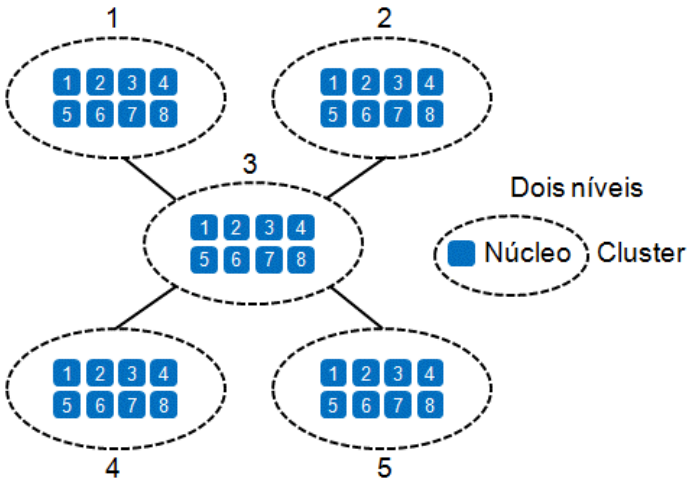


Figura 4.24: Roteamento hierárquico

O próximo capítulo apresenta os resultados alcançados na avaliação da proposta da arquitetura da MCNoC.

5 RESULTADOS

Os resultados apresentados neste capítulo verificam a utilização de componentes no FPGA, consumo de potência e energia, funcionamento e desempenho da arquitetura MCNoC, de acordo com métricas e etapas previamente apresentadas na Metodologia (Capítulo 3). Todos os resultados são baseados em modelos analíticos e simulação através de cargas de trabalho artificiais e naturais.

Para verificação e comparação dos resultados da MCNoC, a SoCIN (*System-on-Chip Interconnection Network*) (ZEFERINO, 2003) foi escolhida por apresentar um tamanho relativamente pequeno e ser baseada nas topologias *mesh* e *torus*, que são mais utilizadas entre os trabalhos relacionados às NoCs. Além disso, as avaliações foram realizadas através de famílias de FPGAs, mencionadas na Metodologia, por dois motivos: i) o FPGA é uma plataforma programável com boa relação de desempenho x flexibilidade para prototipação, e ii) a maior parte dos trabalhos correlatos também focam nesta plataforma. Considerando a grande quantidade de comunicações coletivas que demandam o uso de todos os *links* da NoC, e a alta latência para reconfiguração dinâmica usando FPGA, tanto a MCNoC como a SoCIN usam o FPGA apenas para reconfiguração estática para avaliação de novos protótipos. Conforme é apresentado na Seção 5.1, o FPGA possibilita novas configurações e versões maiores das NoCs para prototipação. No entanto, uma das características da MCNoC é o segundo nível de reconfiguração que não depende da mudança de implementação no FPGA. A exploração do segundo nível para reconfiguração dinâmica é usada pela MCNoC em relação às topologias fixas da SoCIN implementadas no FPGA. Em todas as versões de NoCs avaliadas, a profundidade dos *buffers* de entrada é igual a 3, e a largura de *bits* dos *buffers* e *links* é igual a 32.

5.1 Avaliação da Ocupação em FPGA

Esta seção tem por objetivo apresentar a avaliação dos resultados de ocupação de área. Estes resultados são importantes para viabilidade de implementação da arquitetura proposta, já que uma NoC grande pode ocupar espaço de núcleos, e / ou, aumentar consideravelmente o consumo de potência, como será visto adiante. Os resultados comparativos através de Flip-Flops e LUTs, entre as versões de NoCs, não apresentam porcentagem de ocupação no FPGA. O objetivo é avaliar o quanto a MCNoC é equivalente às topologias mais encontradas no estado da arte, baseadas em *mesh* e *torus*, sem a necessidade de um FPGA alvo para implementação, uma vez que a quantidade de *slices* utilizados pode variar entre versões de dispositivos FPGAs.

A Tabela 5.1 apresenta os resultados de ocupação do FPGA em termos de *Flip-Flops* e *LUTs* para os elementos que compõem cada versão de NoC e dos roteadores programáveis. Os resultados da RCS-NR mostram que os *buffers* são responsáveis por um aumento de aproximadamente 6 vezes no número de *Flip-Flops*. O processador Plasma teve seu tamanho reduzido (conforme metodologia) para se adequar ao projeto NPoC. A redução se deve, principalmente, no número de registradores e instruções. No entanto, existe um pequeno acréscimo de *hardware* para interfaceamento com a RCS-NR. As versões do roteador RANoC com a RCS-NR *bufferizada* e Plasma ou NPoC, mostram que a redução do processador é muito importante e significativa para o contexto em *chip*. Em comparações de área ocupada para suportar a mesma quantidade de núcleos (oito), o RANoC baseado no NPoC mostrou-se adequado possuindo componentes em quantidade menores do que a SoCIN versão *torus*, e apenas superior em termos de *Flip-Flops* comparando com a SoCIN versão *mesh*.

Tabela 5.1: Utilização de componentes do FPGA 2vp100ff1704-6para RANoC (adaptado de FREITAS, 2008c)

	Buffered version		Reduced version	
Components	**RCS-NR**	**BRCS-NR**	**Plasma**	**NPoC**
Flip-Flops	520	3344	1346	733
4-LUTs	3062	2665	5067	2174

	Programmable routers		NoCs	
Components	**RANoC 8 (Plasma)**	**RANoC 8 (NPoC)**	**SoCIN 2x4 mesh**	**SoCIN 2x4 torus**
Flip-Flops	4560	3935	3425	3948
4-LUTs	7382	4555	5464	6486

A Tabela 5.2 estende a comparação para versões da MCNoC que possuem mais de um roteador. A MCNoC 14 possui dois RANoC 8 interconectados e a MCNoC 28/32 possui cinco RANoCs 8 interconectados conforme topologia da Figura 4.21. Através da MCNoC 28/32 é possível alcançar até 32 núcleos usando as portas livres do roteador central. A MCNoC 14 utiliza componentes em quantidade similar em comparação com a SoCIN 2x7 *mesh* e *torus*, utilizando mais *Flip-Flops* e menos *LUTs* para suportar 14 núcleos, em ambas as topologias da SoCIN. Os números da MCNoC 28/32 mostram que a utilização de componentes é maior em relação a SoCIN 4x7 *mesh*, e inferior apenas em *LUTs* em relação a SoCIN 4x7 *torus*. No entanto, a MCNoC 28/32 que inicialmente equivale em quantidade de núcleos suportados, é capaz de gerenciar mais quatro núcleos através das portas livres do roteador central.

Tabela 5.2: Utilização de componentes do FPGA 2vp100ff1704-6 para MCNoC (adaptado de FREITAS, 2008c)

	Multi-Cluster NoCs	NoCs	
Components	**MCNoC 14**	**SoCIN 2x7 mesh**	**SoCIN 2x7 torus**
Flip-Flops	7903	6362	6909
4-LUTs	10009	10371	11733
Components	**MCNoC 28/32**	**SoCIN 4x7 mesh**	**SoCIN 4x7 torus**
Flip-Flops	19761	14491	17147
4-LUTs	24203	23683	29086

Além dos componentes principais para comparação (Flip-Flops e LUTs), as versões da MCNoC são baseadas em roteadores programáveis que utilizam outros componentes que podem não aparecer nos relatórios da SoCIN. Estes componentes são referentes em sua maioria ao processador NPoC (versão Plasma reduzido) e são apresentados em relatórios sumarizados do Anexo A.

Um componente em especial, também só utilizado pelas versões MCNoC, são os blocos de memórias BRAMs (*Block RAMs*) do FPGA para implementação das memórias acessadas pelos NPoCs . Apesar das BRAMs não fazerem parte do projeto da MCNoC, por se tratar do projeto de sistema de memória integrado com o projeto do processador *many-core*, estas informações são necessárias para ilustrar a quantidade que foi utilizada em cada projeto. Através de otimizações da ferramenta de síntese, é possível usar BRAMs para implementação das arquiteturas, reduzindo o número de LUTs, e muitas vezes aumentando o desempenho, atingindo freqüências mais altas para as versões da MCNoC. No entanto, esta otimização está desabilitada, para que as BRAMs sejam usadas apenas para implementação de memórias. O não uso da otimização faz com que as comparações entre arquiteturas realizadas nas Tabelas 5.1 e 5.2 sejam corretas em termos de Flip-Flops e LUTs. A quantidade de BRAMs em cada versão da MCNoC está ilustrada na Tabela 5.3. Cada roteador possui um conjunto de 15 BRAMs para acesso a uma memória de 4k *bytes*. A porcentagem de uso no FPGA de cada versão é relativamente baixa.

Tabela 5.3: Quantidade de BRAMs utilizadas pela MCNoC do FPGA 2vp100ff1704-6

Block RAMs	Suporte 8 núcleos		Suporte 14 núcleos		Suporte 32 núcleos	
	Usado	%	Usado	%	Usado	%
	15 de 444	3%	30 de 444	6%	75 de 444	16%

Com relação ao uso de redes de interconexões do FPGA, a Tabela 5.4 mostra que a quantidade utilizada por todas as versões está em patamares similares. Para o suporte a 8 núcleos, a versão com 1 roteador da MCNoC utiliza uma menor quantidade de rede do FPGA em relação às duas topologias *mesh* e *torus* da SoCIN. Para o suporte a 14 e 28 núcleos, a quantidade de rede do FPGA utilizada pela MCNoC é maior em relação à topologia *mesh*, de igual tamanho à topologia *torus* para suporte a 14 núcleos, e menor do que a topologia *torus* para suporte a 28 núcleos. Os números da rede do FPGA seguem a mesma tendência encontrada na soma da quantidade de Flip-Flops e LUTs usadas em cada versão.

Tabela 5.4: Redes roteadas do FPGA 2vp100ff1704-6

Redes do FPGA	Suporte a 8 núcleos			Suporte a 14 núcleos			Suporte a 28 núcleos		
	mesh	torus	mcnoc	mesh	torus	mcnoc	mesh	torus	mcnoc
	8549	9579	**8082**	15538	16946	**16946**	34757	41680	**41257**

Os resultados apresentados pelas Tabelas 5.1 a 5.4 mostram que é viável a implementação de uma NoC programável baseada em *clusters* de *cores*. A quantidade de componentes utilizados do FPGA é muito similar a uma NoC tradicional tal como a SoCIN. Componentes referentes principalmente ao NPoC (Anexo A) são em

quantidades pequenas e não acrescentam *overheads* de ocupação em FPGA que possam inviabilizar a implementação.

Para aumentar a vazão de *threads* e, portanto, de desempenho na execução de programas, o processador de rede NPoC, versão reduzida e adaptada do Plasma, foi modificado para suportar IMT. A versão sem IMT do NPoC possui um banco de 16 registradores, enquanto que as versões para duas *threads* e quatro *threads* possuem, respectivamente, 32 e 64 registradores. A Tabela 5.5 apresenta um aumento considerável de Flip-Flops e LUTs, principalmente para a versão com quatro *threads*. No entanto, comparando o roteador RANoC com a versão correspondente da SoCIN é possível verificar que com o suporte a duas *threads* o roteador possui mais Flip-Flops e menos LUTs na comparação com as versões *mesh* e *torus*. O roteador com suporte a 4 *threads* é maior do que a versão *mesh* da SoCIN, mas utiliza uma quantidade menor de LUTs em relação a versão *torus*. Apesar desta maior utilização de componentes, o roteador programável com suporte a IMT também se mostrou viável para implementação em FPGA.

Tabela 5.5: Utilização de componentes do FPGA 2vp100ff1704-6 para NPoC com e sem IMT (FREITAS, 2009d)

Prototypes		*Without IMT*	*With 2-IMT*	*With 4-IMT*
NPoC	FFs	733	1136	2167
	4-LUTs	2174	2311	3357
Router crossbar 8x8	FFs	3935	4354	5375
	4-LUTs	4555	5081	6121
Conventional NoC		*2x4 mesh*	*2x4 torus*	
SoCIN	FFs	3425	3948	
	4-LUTs	5464	6486	

5.2 Avaliação do Tempo de Transmissão, e Consumo de Potência e Energia

Nesta seção o objetivo é avaliar o tempo de transmissão de pacotes e o consumo de potência e energia das versões da MCNoC através de cargas artificiais e modelagens analíticas. A avaliação do consumo de potência e energia depende do comportamento da carga utilizada para avaliação do tempo de transmissão de pacotes descrita na subseção seguinte.

5.2.1 Avaliação do Tempo de Transmissão de Pacotes

A cada salto em uma *Network-on-Chip*, um novo roteador recebe o pacote, analisa o cabeçalho e decide a rota. Esta tarefa resulta em um tempo adicional, também chamado de latência ou atraso de envio para o primeiro *byte* (largura) do pacote. Tal como em um *pipeline* os *bytes* seguintes não sofrem efeito deste atraso. Os resultados de tempo de transmissão de pacotes levam em consideração as latências identificadas na simulação da descrição de *hardware* da MCNoC. Portanto, os resultados apresentados nesta seção consideram a latência de configuração de topologias na RCS-NR. Este atraso de configuração corresponde a dois ciclos, e após esta latência, os pacotes são enviados. A tabela de prioridade usada pelo árbitro da RCS-NR funciona como uma entrada em *pipeline* para a seleção de conflitos de saída, não adicionando atrasos condicionados às requisições de comunicação. É importante ressaltar, que topologia implementada

significa chaveamento de *crosspoint* realizado, portanto, não há atrasos para novo chaveamento na ocorrência de conflitos.

A Tabela 5.6 apresenta os resultados máximos das freqüências de interconexão apontados pela síntese da descrição em VHDL. As freqüências de interconexão da MCNoC são menores do que versões da SoCIN, uma vez que as chaves *crossbar* da MCNoC são maiores e mais complexas. No entanto, a vazão de pacotes para *broadcast* da MCNoC é significantemente maior em função do melhor desempenho da chave *crossbar* em oposição ao roteamento XY das topologias *mesh* e *torus*. Considerando que a freqüência do NPoC é a mesma da interconexão, a freqüência da MCNoC cai para 73,56 MHz, 77,44 MHz e 75,49 MHz, para suporte a 8, 14 e 28/32 *cores*, respectivamente. Para os experimentos, a freqüência utilizada por todas as versões de NoCs foi de 10 MHz, considerando que esta é também a freqüência para os roteadores programáveis da MCNoC. O primeiro motivo se deve à verificação do efeito da arquitetura no desempenho tornando a freqüência uma constante entre as arquiteturas. O segundo motivo se deve ao fato de que os protótipos da MCNoC e SoCIN se mostraram mais estáveis com freqüências mais baixas. Com a freqüência fixa, a vazão de *broadcast* da MCNoC permanece superior em relação a SoCIN.

Tabela 5.6: Resultados de freqüência de interconexão e vazão (throughput) no FPGA 2vp100ff1704-6 (FREITAS, 2008c)

	8 Cores			14 Cores			28 Cores		
	RANoC 8	SoCIN 2x4 mesh	SoCIN 2x4 torus	MCNoC 14	SoCIN 2x7 mesh	SoCIN 2x7 torus	MCNoC 28	SoCIN 4x7 mesh	SoCIN 4x7 torus
Maximum Interconnect Frequency	145.64 MHz	160.87 MHz	164.60 MHz	142.33 MHz	160.02 MHz	160.87 MHz	133.98 MHz	159.41 MHz	162.83 MHz
Maximum Broadcast Throughput	4.08 GB/s	644 MB/s	658 MB/s	7.40 GB/s	640 MB/s	644 MB/s	14.5 GB/s	638 MB/s	651 MB/s
Experimental Interconnect Frequency	10 MHz	10 MHz	10 MHz	10 MHz	10 MHz	10 MHz	10 MHz	10 MHz	10 MHz
Experimental Broadcast Throughput	280 MB/s	40 MB/s	40 MB/s	520 MB/s	40 MB/s	40 MB/s	1080 MB/s	40 MB/s	40 MB/s

Os resultados de tempo de transmissão utilizam os modelos analíticos apresentados na Seção 3.4, além das latências coletadas pela simulação do *hardware* e os valores experimentais de freqüência. Os resultados encontrados são para mil pacotes que variam de 1 *byte* a 4096 *bytes*.

A Figura 5.1 apresenta o tempo de transmissão de pacotes entre nós adjacentes para o padrão de comunicação *one-to-one*. No caso de topologias *mesh* e *torus* existem dois roteadores envolvidos na comunicação. As reduções nos tempos de transmissão de pacotes da MCNoC em relação à SoCIN são: 96,95% para pacotes de 1 *byte*, 20 % para pacotes de 128 *bytes*, e 0,78 % para pacotes com 4096 *bytes*.

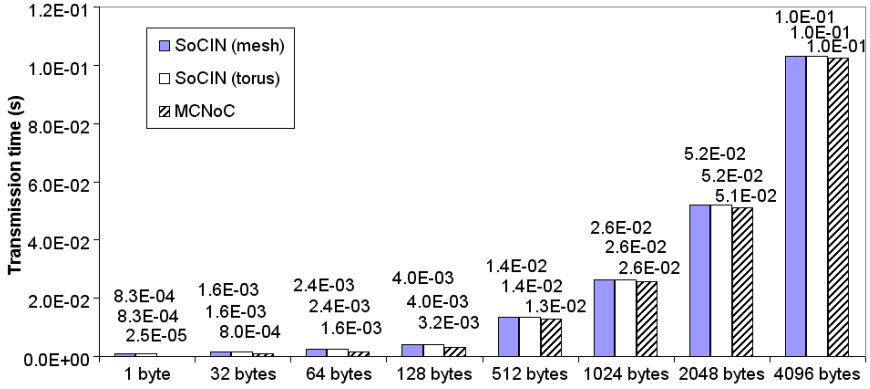


Figura 5.1: Tempo de transmissão one-to-one (FREITAS, 2008d)

A Figura 5.2 apresenta o tempo de transmissão de pacotes para o padrão *all-to-one*. Para este modelo são encontrados os maiores tempos de transmissão devido a maior concentração de pacotes em um número reduzido de roteadores, ou dos conflitos de saída na chave *crossbar* da MCNoC (topologia estrela implementada). Mesmo com um desempenho menor, existem reduções nos tempos de transmissão de pacotes da MCNoC em relação à SoCIN, conforme os seguintes dados: i) 90,66% para pacotes de 1 *byte*, 7,05 % para pacotes de 128 *bytes*, e 0,24 % para pacotes com 4096 *bytes* em comparação com a topologia *mesh*. ii) 88,88% para pacotes de 1 *byte*, 5,88 % para pacotes de 128 *bytes*, e 0,19 % para pacotes com 4096 *bytes* em comparação com a topologia *torus*.

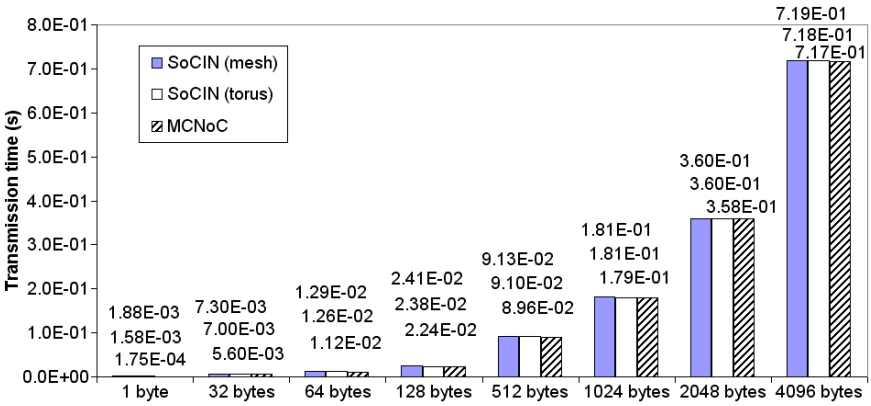


Figura 5.2: Tempo de transmissão all-to-one (FREITAS, 2008d)

A Figura 5.3 apresenta o melhor desempenho da MCNoC em relação a SoCIN. A RCS-NR explora o padrão de comunicação *broadcast* (*one-to-all*) através da capacidade de enviar simultaneamente vários pacotes para um mesmo destino. Ao contrário das topologias de NoC *mesh* e *torus*, que precisam de sucessivos saltos para alcançar todos os nós da rede. As reduções nos tempos de transmissão de pacotes da MCNoC em relação à SoCIN são: i) 98,66% para pacotes de 1 *byte*, 86,72 % para pacotes de 128 *bytes*, e 85,75 % para pacotes com 4096 *bytes* em comparação com a topologia *mesh*. ii) 98,40% para pacotes de 1 *byte*, 86,55 % para pacotes de 128 *bytes*, e 85,74 % para pacotes com 4096 *bytes* em comparação com a topologia *torus*.

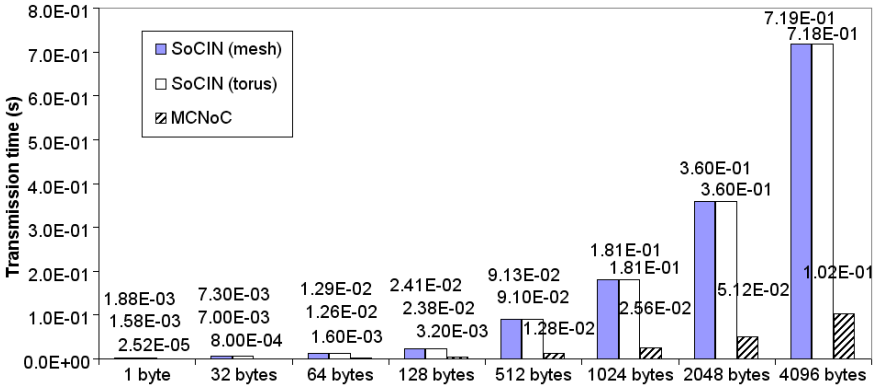


Figura 5.3: Tempo de transmissão broadcast (FREITAS, 2008d)

Um comportamento comum a todas as avaliações está relacionado ao impacto da latência dos roteadores. Quanto menor o pacote, maior a influência da latência, e quanto maior é o pacote, menor é este efeito. Portanto, o principal ganho da MCNoC em relação às topologias *mesh* e *torus* acontece com pacotes menores. A Seção 5.3 descreve que este tipo de comportamento permanece igual quando traços de cargas de trabalho paralelas reais são avaliados.

5.2.2 Avaliação do Consumo de Potência e Energia

O comportamento da carga de trabalho modelada nesta seção foi reproduzido como *testbench* para simulação do *hardware* e avaliação do consumo de potência com foco específico de identificar o impacto da transição de sinais na transmissão de pacotes (modelo artificial).

Portanto, o objetivo desta seção é apresentar a avaliação do consumo de potência e energia através de um modelo artificial de carga de trabalho que possui as seguintes características:

- Mil pacotes de 32 *bits* submetidos em cada *buffer* de entrada da MCNoC simulando utilização de máxima simultaneidade dos *links* da RCS-NR.
    - Dois tipos de carga para verificar o impacto da transição de sinais (0 para 1 ou 1 para 0) no consumo de potência dinâmica:
        - Alta transição de sinais (hst – *high signal transition*): A cada envio de pacotes, novo ciclo, 32 *bits* mudam de valor.
        - Baixa transição de sinais (lst – *low signal transition*): A cada envio de pacotes, novo ciclo, 2 *bits* mudam de valor.
    - Os dois tipos procuram identificar extremos máximos e mínimos no consumo de potência.
- O programa executado no NPoC acessa o decodificador da RCS-NR para implementação de topologias, e lê a memória constantemente através de instruções do tipo *load* e *store*.
    - As topologias implementadas através do programa do NPoC estão ilustradas na Figura 5.4.

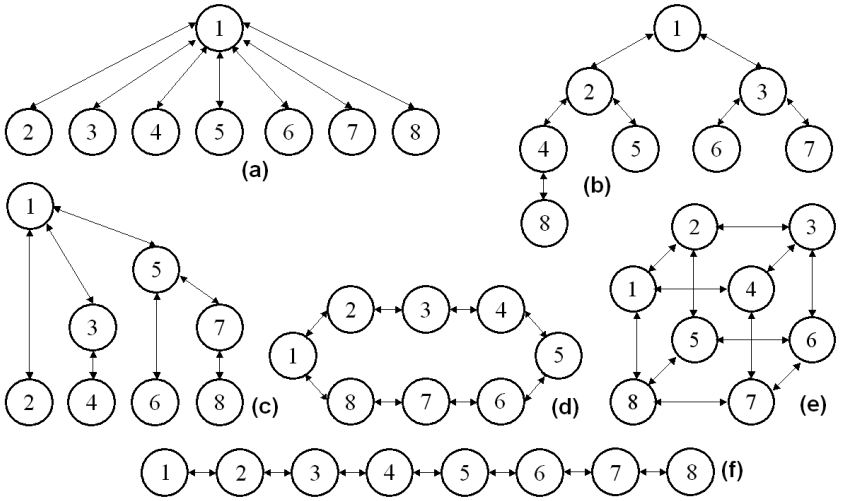


Figura 5.4: Topologias implementadas na RCS-NR. (a) Estrela, (b) Árvore Balanceada, (c) Árvore Binomial, (d) Anel, (e) Hipercubo e (f) Pipeline (FREITAS, 2008c)

A Figura 5.5 apresenta os resultados da avaliação do consumo de potência entre os roteadores RANoCs projetados com a versão original do Plasma e a versão adaptada NPoC. Apesar da redução de área ser significativa, conforme Tabela 5.1, a diferença do consumo de potência entre os dois projetos é muito pequena. Neste teste, cada RANoC gerencia oito *links* de comunicação simultânea. O RANoC baseado no NPoC tem um consumo de potência 4,93 % e 3,04 % menor em relação ao projeto baseado no Plasma, considerando baixa e alta transição de sinais, respectivamente. O grande motivo desta baixa diferença está associado às comunicações simultâneas, porque as transições de sinais nos *buffers* de entrada de cada porta têm maior influência no consumo de potência dinâmica do que o tamanho ou a execução de instruções do NPoC. Como o comportamento de cada comunicação é igual para cada teste e para cada versão da RANoC, a pequena diferença de consumo de potência está associada à grande redução de área dos processadores.

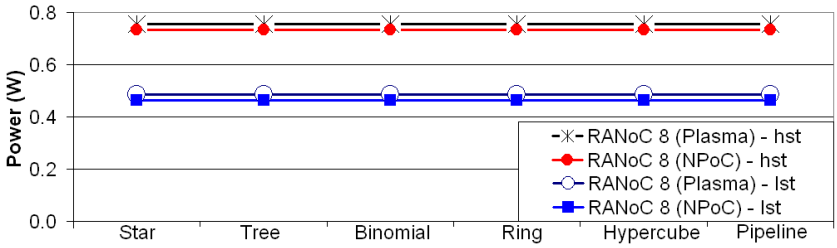


Figura 5.5: Consumo de potência do RANoC no FPGA 2vp100ff1704-6 (FREITAS, 2008c)

A mesma carga de trabalho foi submetida às versões MCNoC 14, MCNoC 28, e SoCIN 2x4, 2x7 e 4x7, sendo as versões SoCIN baseadas nas topologias *mesh* e *torus*. Considerando o comportamento constante de consumo de potência identificado através da Figura 5.5 e também pelos testes com versões da MCNoC, a Figura 5.6 apresenta um único resultado para as topologias simuladas. A MCNoC apresentou melhores resultados de consumo de potência, alcançando uma redução de até 26,96 % e 23,77 % para alta e baixa transição de sinais, respectivamente. O principal motivo para esta redução significativa está baseado na quantidade de *buffers* de entrada presentes em cada versão avaliada. No caso da SoCIN, cada roteador possui um conjunto de cinco *buffers* de entrada e no mínimo dois *buffers* por roteador são utilizados para realização

das comunicações (entradas de rede e núcleo), enquanto que cada roteador da MCNoC possui apenas um *buffer* em atividade para cada comunicação (entrada de núcleo).

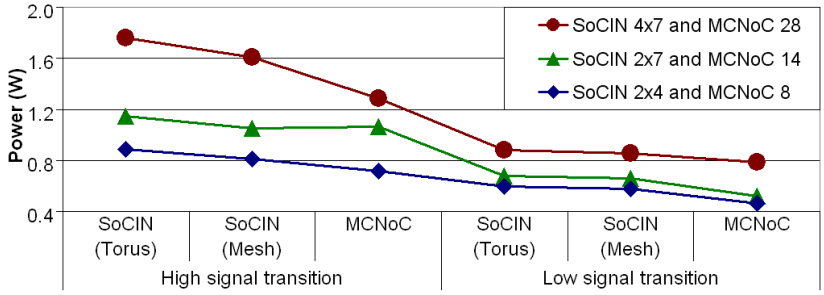


Figura 5.6: Consumo de potência da MCNoC no FPGA 2vp100ff1704-6 (FREITAS, 2008c)

O consumo de energia foi avaliado através dos dados de tempo de transmissão de pacotes de 32 *bits* para os padrões *one-to-all, all-to-one e one-to-all* (Subseção 5.2.1), para *clusters* de até oito *cores*. Portanto, para as versões de NoCs com suporte a 14 e 28 núcleos, são 2 e 4 *clusters*. Devido à baixa latência para envio de pacotes, o tempo de transmissão da MCNoC é baixo em relação ao alcançado pela SoCIN. Como visto nas figuras anteriores, o consumo de potência também é menor em comparação com a SoCIN. Por conseqüência, o consumo de energia da MCNoC também é mais baixo, alcançando uma redução de até 96,52% em relação à topologia *torus* da SoCIN para o padrão *one-to-all*, conforme dados apresentados na Tabela 5.7. De acordo com o menor consumo de potência e menor tempo de transmissão de pacotes para os padrões *one-to-one* e *all-to-one*, o consumo de energia da MCNoC para estes padrões também é menor, em comparação com a SoCIN. Nestes casos, a MCNoC atinge uma redução de até 91,88% e 75,65%, para os padrões *one-to-one* e *all-to-one*, respectivamente, em relação à topologia *torus* da SoCIN. Uma característica que pode ser observada tanto pela Tabela 5.7, quanto pelas Figuras 5.1, 5.2 e 5.3, é o comportamento parecido da MCNoC nos padrões *one-to-one* e *one-to-all*, e da SoCIN nos padrões *all-to-one* e *one-to-all*. Para a MCNoC, o custo de comunicação entre oito comunicações *one-to-one* e uma comunicação *one-to-eight* é o mesmo na chave *crossbar* RCS-NR. No caso da SoCIN, a influência do número de saltos nas topologias *mesh* e *torus* para *one-to-eight* ou *eight-to-one* é a mesma, resultando em um custo igual de comunicação.

Tabela 5.7: Consumo de energia no FPGA 2vp100ff1704-6

Consumo em mJ		8 cores			14 cores			28 cores		
		mcnoc	mesh	torus	mcnoc	mesh	torus	mcnoc	mesh	torus
One-to-one	lst	0,046	0,50	0,53	0,051	0,594	0,611	0,078	0,770	0,792
	hst	0,071	0,748	0,797	0,102	0,944	1,034	0,128	1,448	1,586
All-to-one	lst	0,322	1,344	1,243	0,362	1,584	1,426	0,548	2,054	1,850
	hst	0,499	1,997	1,861	0,743	2,518	2,413	0,900	3,862	3,700
One-to-all	lst	0,046	1,344	1,243	0,051	1,584	1,426	0,078	2,054	1,850
	hst	0,071	1,997	1,861	0,102	2,518	2,413	0,128	3,862	3,700

Finalizando, em relação ao consumo de potência, a MCNoC se mostrou viável, já que com uma quantidade menor de *buffers*, consome menos potência. Em relação ao

consumo de energia, a MCNoC por ter um baixo consumo de potência e uma alta vazão de pacotes, possui um baixo consumo de energia.

5.3 Avaliação de Desempenho da MCNoC

Conforme apresentado no Capítulo 3, as cargas de trabalho naturais baseadas em programas reais são as mais adequadas para avaliação de desempenho. Através dos traços coletados do conjunto de programas paralelos do benchmark NAS (Tabela 3.1) é possível identificar e visualizar os padrões de comunicação coletiva, analisar o comportamento da comunicação no que diz respeito aos períodos de atividade, tamanho das mensagens e avaliar o desempenho na transmissão dos pacotes.

A Figura 5.7 ilustra a visualização de parte do traço do padrão *one-to-one* obtido da aplicação BT. Em complemento a Figura 2.14, a comunicação cheia em um longo intervalo de tempo passa a impressão de que não existem padrões bem definidos e todos os núcleos comunicam entre si. Na verdade, a comunicação entre todos acontece, mas as primitivas usadas são *send* e *receive*, e representam troca de mensagens entre dois núcleos em um determinado intervalo de tempo. As Figuras 5.8 e 5.9 mostram que fechando o intervalo de tempo, é possível identificar pequenos padrões de comunicação que uma vez implementados nas RCS-NRs dos roteadores se tornam mapeamentos físicos da representação lógica de comunicação. Esta adaptação da topologia, como será descrito nesta subseção, aumenta o desempenho da NoC, reduzindo o tempo de transmissão de pacotes.

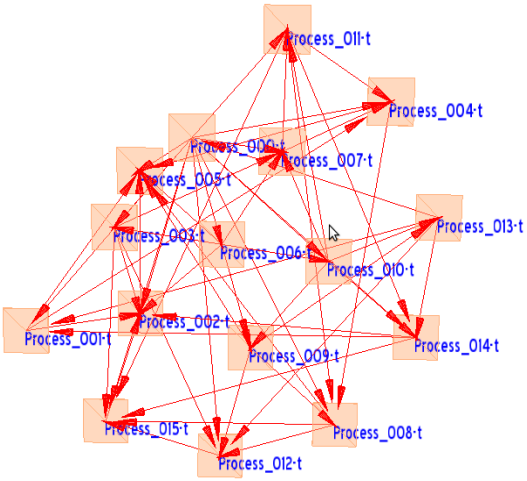


Figura 5.7: Visualização do Triva para comunicação one-to-one cheia (Traço BT)

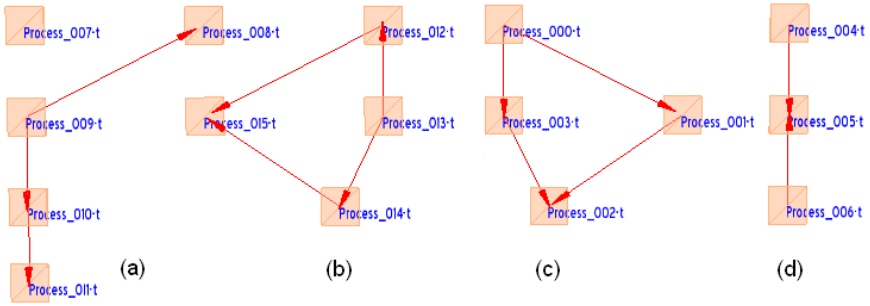


Figura 5.8: Visualização do Triva de intervalo one-to-one (Traço BT). (a) one-to-two e one-to-one, (b) e (c) one-to-two e two-to-one, e (d) two-to-one

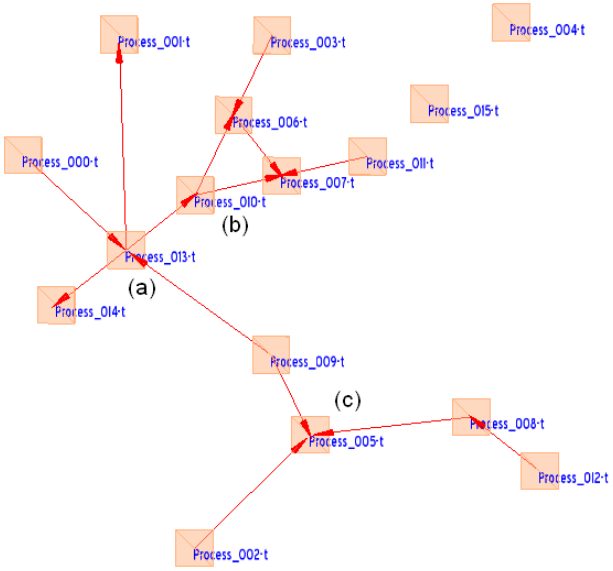


Figura 5.9: Visualização do Triva de intervalo one-to-one (Traço BT). (a) one-to-three e two-to-one, (b) three-to-one, two-to-one e one-to-two, e (c) three-to-one e one-to-one

É importante ressaltar que a implementação de uma topologia completa na RCS-NR para suportar a comunicação cheia ilustrada pela Figura 5.7 pode ser feita, mas isto requer um trabalho mais intenso do processador de rede NPoC. Com o maior número de conexões realizadas, é necessária uma alteração mais freqüente nas entradas da Tabela de Arbitragem. Considerando que em determinados intervalos de tempo, nem todas as comunicações são realizadas, e a manutenção dos campos em cada entrada que indica a habilitação de comunicação fica mais complexa. Esta complexidade pode criar lacunas de transmissões de pacotes caso o gerenciamento tenha falhas.

Para análise e avaliação dos traços coletados, foi desenvolvida uma ferramenta através da linguagem de programação Python que faz a leitura dos arquivos (traços), analisa os dados, e gera informações referentes ao tamanho e quantidade de mensagens transmitidas ao longo de todo o período de comunicação coletiva. Nesta primeira etapa de interpretação dos dados, a ferramenta classifica as comunicações coletivas em quatro padrões ilustrados pela Figura 5.10.

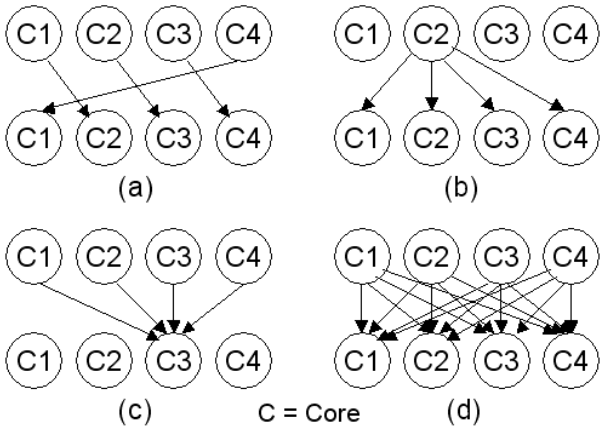


Figura 5.10: Padrões de comunicação coletiva. (a) one-to-one, (b) one-to-all, (c) all-to-one, e (d) all-to-all

A Tabela 5.8 apresenta os resultados obtidos de cada traço coletado. Inicialmente, é possível identificar uma concentração de mensagens maiores e em maior quantidade no

padrão *one-to-one*. Os demais padrões, principalmente o *one-to-all*, que obteve um alto desempenho no uso da carga artificial da Subseção 5.2.1, possuem tamanhos e quantidades pequenas de mensagens transmitidas. As exceções são as aplicações FT e IS que possuem no padrão *all-to-all*, um tamanho maior de mensagens. Estas características influenciam diretamente nos tempos de transmissão de pacotes enviados pelas NoCs, conforme é visto a partir dos resultados apresentados pela Figura 5.13.

Portanto, a segunda etapa realizada pela ferramenta desenvolvida em Python é justamente a avaliação de desempenho baseada em modelos analíticos específicos para cada NoC comparada. O grande problema na comunicação entre núcleos é justamente o atraso imposto pelos roteadores envolvidos na transmissão de pacotes. A Figura 5.11 ilustra este problema ressaltando os tempos que são mais considerados nas modelagens e avaliações de desempenho, tais como: Tempo de transmissão (Tt), Ta (Tempo de atraso ou residência), Tempo de fila ou *buffer* (Tf) e Tempo de serviço (Ts). De acordo com a metodologia, o tempo de atraso ou residência depende do tempo de espera em fila e do tempo de serviço. A cada nova visita do pacote à fila o tempo de atraso aumenta. Para uma rede ideal, ou seja, sem contenções nas filas ou *buffers*, o tempo de atraso é o tempo de serviço. No caso das NoCs modeladas, não há contenções nos *buffers* e o tempo de serviço é igual à latência para leitura dos cabeçalhos e encaminhamento de pacotes, ou mapeamento de topologias no caso da MCNoC.

Tabela 5.8: Quantidade e tamanho das mensagens coletadas

Padrões	Cores	Mensagens	BT	CG	EP	FT	IS	LU	MG	SP
One-to-one	8	Quantidade	32616	23552	0	0	7	316338	5712	65016
		Tamanho máx. (*bytes*)	58080	14000	0	0	4	81920	67600	38808
		Tamanho min. (*bytes*)	9680	4	0	0	4	128	4	16000
	16	Quantidade	77280	47104	0	0	15	759204	11024	154080
		Tamanho máx. (*bytes*)	34680	14000	0	0	4	40960	67600	30720
		Tamanho min. (*bytes*)	5780	4	0	0	4	128	4	9000
	32	Quantidade	260712	134656	0	0	31	1644936	21728	519912
		Tamanho máx. (*bytes*)	23360	7000	0	0	4	40960	34320	23360
		Tamanho min. (*bytes*)	2420	4	0	0	4	64	4	3240
One-to-all	8	Quantidade	6	0	0	2	0	9	6	4
		Tamanho máx. (*bytes*)	12	0	0	12	0	20	32	12
		Tamanho min. (*bytes*)	4	0	0	4	0	4	4	4
	16	Quantidade	5	0	0	2	0	9	6	3
		Tamanho máx. (*bytes*)	12	0	0	12	0	20	32	12
		Tamanho min. (*bytes*)	4	0	0	4	0	4	4	4
	32	Quantidade	6	0	0	2	0	9	6	4
		Tamanho máx. (*bytes*)	12	0	0	12	0	20	32	12
		Tamanho min. (*bytes*)	4	0	0	4	0	4	4	4
All-to-one	8	Quantidade	2	1	0	6	2	0	1	2
		Tamanho máx. (*bytes*)	4	4	0	4	4	0	4	4
		Tamanho min. (*bytes*)	4	4	0	4	4	0	4	4
	16	Quantidade	1	1	0	6	2	0	1	1
		Tamanho máx. (*bytes*)	4	4	0	4	4	0	4	4
		Tamanho min. (*bytes*)	4	4	0	4	4	0	4	4
	32	Quantidade	2	1	0	6	2	0	1	2
		Tamanho máx. (*bytes*)	4	4	0	4	4	0	4	4
		Tamanho min. (*bytes*)	4	4	0	4	4	0	4	4

Padrões	Cores	Mensagens	BT	CG	EP	FT	IS	LU	MG	SP
All-to-all	8	Quantidade	3	0	4	8	33	10	88	3
		Tamanho máx. (*bytes*)	20	0	40	524288	530632	20	16	20
		Tamanho min. (*bytes*)	20	0	4	524288	4	4	4	20
	16	Quantidade	2	0	4	8	33	10	88	2
		Tamanho máx. (*bytes*)	20	0	40	131072	132648	20	16	20
		Tamanho min. (*bytes*)	20	0	4	131072	4	4	4	20
	32	Quantidade	3	0	4	8	33	10	88	3
		Tamanho máx. (*bytes*)	20	0	40	32768	33488	20	16	20
		Tamanho min. (*bytes*)	20	0	4	32768	4	4	4	20

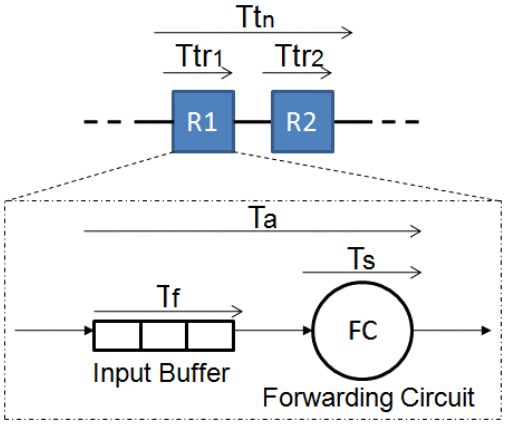


Figura 5.11: Métricas de avaliação de desempenho

Para a avaliação de desempenho, a MCNoC foi comparada com topologias tradicionais *mesh* e *torus*. As características e latências das arquiteturas foram extraídas da SoCIN e modeladas juntamente com a MCNoC na ferramenta desenvolvida em Python. Considerando as diferenças nas arquiteturas (Tabela 5.9), dois modelos foram feitos, conforme as Equações 7 e 8.

Tabela 5.9: Principais características das NoCs modeladas

	SoCIN	MCNoC
Latência do roteador	3 ciclos	2 ciclos
Técnica de chaveamento	Pacote (*wormhole*)	Circuito
Topologia	*Mesh*/*Torus*	Topologia reconfigurável baseado em estrela

$$Tt_{mesh/torus} = \left(\frac{L}{F} * ns * np\right) + \left(\frac{P}{B} * np\right) \qquad \text{Equação 7}$$

$$Tt_{mcnoc} = \left(\frac{L}{F} * ns\right) + \left(\frac{P}{B} * np\right) \qquad \text{Equação 8}$$

As principais métricas utilizadas na modelagem são: Latência (L - ciclos), Freqüência (F- 100 MHz), Número de Saltos (ns), Número de Pacotes (np), Tamanho do Pacote (P - *bytes*) e Largura de Banda (B - *bytes*/s).

A principal diferença entre os modelos, que reflete também a característica de cada NoC, diz respeito à influência das latências. Esta influência aumenta à medida que o número de saltos também aumenta. No caso da MCNoC, a clusterização de núcleos reduz o número de saltos. Outro fator importante é a quantidade de pacotes e o reflexo no tipo de chaveamento. A MCNoC possui chaveamento por circuito para o modelo em Python, devido às implementações das topologias. Por este motivo, a latência influencia apenas uma vez, independente do número de pacotes enviados na comunicação. No caso das topologias *mesh* e *torus*, as latências dos roteadores têm influência em todos os pacotes transmitidos, além de haver um número maior de saltos em cada comunicação. Esta análise feita para cada arquitetura é verificada e avaliada através dos resultados das aplicações dos modelos apresentados nesta subseção. Um fator que também agrega na perda ou ganho de desempenho é o mapeamento dos processos realizados pelo SO. Este mapeamento apresenta comunicações coletivas com maior custo, caso a distância entre os núcleos utilizados seja muito grande. O impacto de um mapeamento de alto custo tem menor efeito em uma NoC baseada em *clusters* (MCNoC), uma vez que este mapeamento resulta em uma quantidade menor de saltos.

A Tabela 5.8 mostra um tamanho máximo muito grande para as mensagens transmitidas. Os resultados preliminares baseados no modelo artificial, apresentados na Subseção 5.2.1, mostram que o tamanho do pacote influencia muito no ganho de desempenho entre as arquiteturas. Por este motivo, a ferramenta em Python quebra as mensagens em pacotes de tamanhos máximos de 128 *bytes*. Pacotes menores possibilitam uma maior vazão entre comunicações diferentes, dependendo da técnica de arbitragem, aumentando o paralelismo de comunicações. Devido à possibilidade de roteamento de pacotes pela rede, grandes pacotes tendem a monopolizar *links*, reduzindo, portanto, o paralelismo.

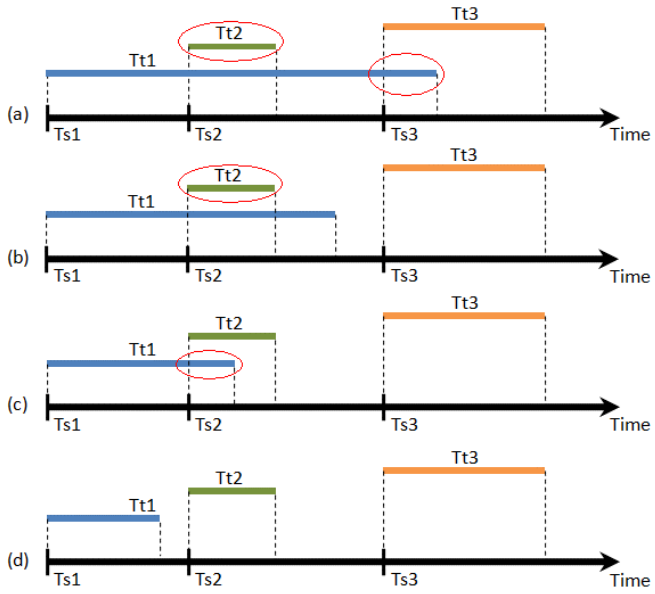
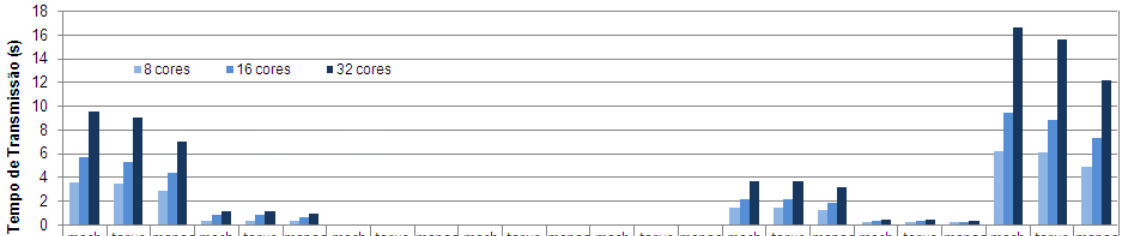


Figura 5.12: Somas relativas dos tempos de transmissão. (a) transmissão e interseção removida, (b) transmissão removida, (c) interseção removida, e (d) sem remoção

Durante a avaliação de desempenho, a ferramenta em Python analisa o comportamento das transmissões de forma a eliminar possíveis interseções temporais, devidas, por exemplo, a grandes transmissões seguidas de pequenas transmissões. A Figura 5.12 ilustra a soma relativa das transmissões que são obtidas por cada comunicação presente nos traços das aplicações paralelas. Os tempos de transmissões

são calculados com base no tamanho e quantidade de pacotes, mas o tempo de início (*Ts – Transmission start*) de cada transmissão é usado para análise devido à interseção já mencionada. Neste caso, uma transmissão inteira pode ser removida, conforme Figura 5.12 (a) e (b). A eliminação de uma interseção entre duas transmissões também ocorre, conforme Figura 5.12 (a) e (c). A avaliação é feita com base no tempo máximo para finalizar todas as transmissões, por isto, algumas podem ser descartadas por não acrescentarem tempo neste cálculo.

Os resultados a seguir são avaliados em função de cada padrão de comunicação coletiva. A Figura 5.13 apresenta os tempos de transmissão de pacotes para o padrão *one-to-one*. De acordo com a Tabela 5.8, as aplicações BT, CG, LU, MG e SP possuem o maior impacto no tempo de transmissão. Embora LU tenha mensagens em quantidade e tamanhos maiores, a aplicação SP tem um tempo de transmissão mais alto e, portanto, possui um impacto ou maior custo do ponto de vista do suporte que uma NoC deve oferecer. Este impacto ocorre em função de um maior número de mensagens grandes, além de uma quantidade maior de saltos para realizar as comunicações. Isto pode ser verificado pelo tamanho mínimo de mensagens, por exemplo, igual a 3240 *bytes* para suporte a 32 núcleos. Devido a impactos desse tipo, a MCNoC atinge um ganho de desempenho em relação às topologias *mesh* e *torus*. Como apresentado na Figura 5.13, a MCNoC que suporta 32 núcleos diminui o tempo de transmissão em relação às topologias *mesh* e *torus* (ambas que suportam 32 núcleos) de até 26% e 22% para a aplicação BT, 18% para a aplicação CG, 14% para as aplicações IS e LU, 23% e 20% para a aplicação MG, e 26% e 22% para a aplicação SP, respectivamente. Em função do comportamento similar de redução do tempo de transmissão pela MCNoC nas três arquiteturas avaliadas com suporte a 8, 16 e 32 núcleos, as porcentagens apresentadas nesta subseção, inclusive para os próximos resultados, são apenas em relação às arquiteturas de 32 núcleos.



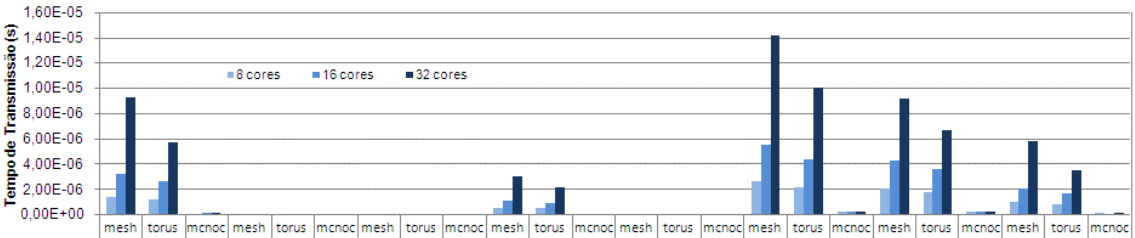
	BT			CG			EP			FT			IS			LU			MG			SP		
	mesh	torus	mcnoc	mesh	torus	mcnoc	mesh	torus	mcnoc	mesh	torus	mcnoc	mesh	torus	mcnoc	mesh	torus	mcnoc	mesh	torus	mcnoc	mesh	torus	mcnoc
8 cores	3,59	3,49	2,83	0,31	0,31	0,27	0	0	0	0	0	0	4,90	4,90	3,50	1,42	1,42	1,19	0,22	0,22	0,19	6,25	6,08	4,93
16 cores	5,68	5,29	4,4	0,8	0,79	0,66	0	0	0	0	0	0	9,10	9,10	6,90	2,16	2,16	1,86	0,31	0,31	0,25	9,47	8,83	7,36
32 cores	9,58	9,03	7,03	1,1	1,1	0,9	0	0	0	0	0	0	2,10	2,10	1,79	3,68	3,68	3,15	0,42	0,41	0,32	16,6	15,7	12,2

Obs.: Para a aplicação IS os valores da tabela devem ser multiplicados por 10^{-07} (8 e 16 cores) e 10^{-06} (32 cores).

Figura 5.13: Tempo de transmissão do padrão de comunicação one-to-one

Uma importante característica da MCNoC é o suporte a *broadcast*. Neste caso, o padrão de comunicação *one-to-all*, que possui a característica de *broadcast*, pode ser implementada pelo roteador da MCNoC através da configuração de uma topologia estrela na RCS-NR. A Figura 5.14 ilustra os resultados alcançados pelo processamento dos traços coletados, evidenciando a maior redução no tempo de transmissão pela MCNoC em relação às topologias *mesh* e *torus*. Portanto, a redução pela MCNoC em relação às topologias *mesh* e *torus* atinge até 98% e 97% para a aplicação BT, 97% e 96% para as aplicações FT, LU e MG, e 97% e 95% para a aplicação SP, respectivamente. Entretanto, a quantidade e o tamanho das aplicações para o padrão *one-to-all* é muito baixo, e por esta razão, o impacto no tempo final de transmissão de

todos os pacotes, incluindo todos os padrões de comunicação coletiva, também é muito baixo. No entanto, conforme é apresentado pelas Figuras 5.8 e 5.9, é possível identificar pequenos padrões em intervalos de tempo, quando a comunicação coletiva é baseada em *one-to-one*. Neste caso, a MCNoC consegue suportar, pela implementação da topologia, características similares tais como *one-to-three*, sem acréscimo de latência devido a requisições de comunicação que não são necessárias. Porém, o mapeamento de pequenas topologias para o padrão *one-to-one* depende de uma caracterização mais detalhada da carga de trabalho. O impacto e viabilidade de programas de gerenciamento, para este caso, requerem um estudo muito preciso, uma vez que o tempo entre padrões, conforme resultados da Tabela 5.11, são pequenos. Estudos amplos e profundos sobre caracterização de cargas de trabalho paralelas e mecanismos / algoritmos de predição de comunicação, podem ajudar na viabilização de pequenos padrões, conforme descrito na Seção 6.1 (Trabalhos Futuros).

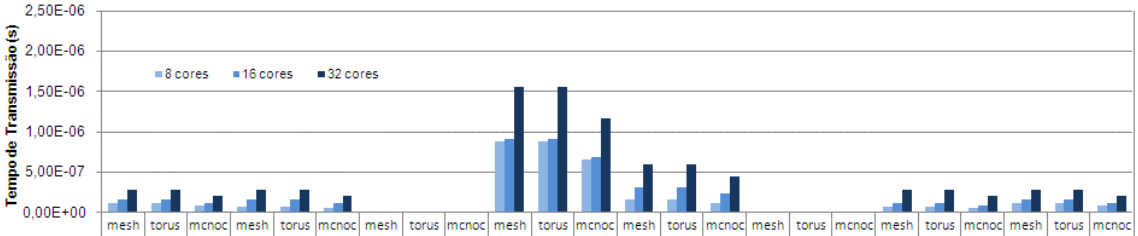


	BT			CG			EP			FT			IS			LU			MG			SP		
	mesh	torus	mcnoc	mesh	torus	mcnoc	mesh	torus	mcnoc	mesh	torus	mcnoc	mesh	torus	mcnoc	mesh	torus	mcnoc	mesh	torus	mcnoc	mesh	torus	mcnoc
8 cores	1,44	1,23	0,12	0	0	0	0	0	0	0,59	0,53	0,08	0	0	0	2,64	2,22	0,31	2,11	1,84	0,25	1,01	0,86	0,19
16 cores	3,22	2,65	0,17	0	0	0	0	0	0	1,17	0,93	0,08	0	0	0	5,59	4,42	0,31	4,32	3,60	0,25	2,06	1,67	0,11
32 cores	9,24	5,76	0,16	0	0	0	0	0	0	3,03	2,22	0,08	0	0	0	14,2	10,1	0,31	9,19	6,67	0,25	5,79	3,57	0,16

Obs.: Todos os valores nas colunas da aplicação LU devem ser multiplicados por 10^{-06}.

Figura 5.14: Tempo de transmissão do padrão de comunicação one-to-all

O padrão de comunicação que possui uma menor redução no tempo de transmissão pela MCNoC em relação às topologias *mesh* e *torus*, é o *all-to-one*, apresentado pela Figura 5.15. Neste caso, todas as entradas da RCS-NR demandam uma mesma saída, isso resulta em uma maior quantidade de conflitos e, portanto, um acréscimo de latência para término das transmissões por cada roteador RANoC. Por este motivo, a MCNoC reduz o tempo de transmissão em relação às topologias *mesh* e *torus* de até 25% para todas as aplicações que possuem a ocorrência deste padrão. Além disso, este padrão segue o mesmo comportamento do padrão *one-to-all*, ambos possuem um baixo impacto no tempo total de transmissão devido à baixa quantidade de pacotes e mensagens de tamanhos pequenos.



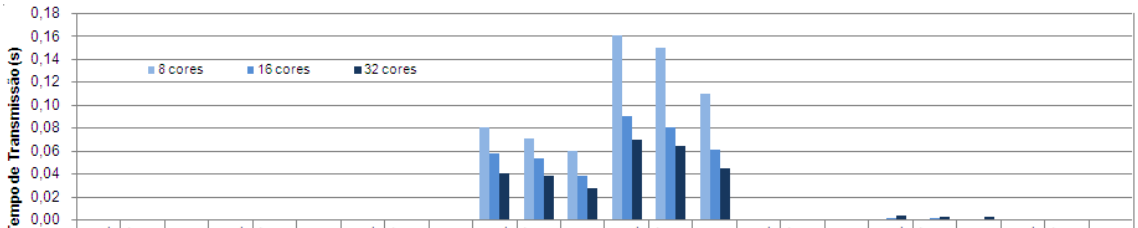
	BT			CG			EP			FT			IS			LU			MG			SP		
	mesh	torus	mcnoc	mesh	torus	mcnoc	mesh	torus	mcnoc	mesh	torus	mcnoc	mesh	torus	mcnoc	mesh	torus	mcnoc	mesh	torus	mcnoc	mesh	torus	mcnoc
8 cores	1,20	1,20	0,90	0,80	0,80	0,60	0	0	0	8,80	8,80	6,60	1,60	1,60	1,20	0	0	0	0,80	0,80	0,60	1,20	1,20	0,90
16 cores	1,60	1,60	1,20	1,60	1,60	1,20	0	0	0	9,20	9,20	6,90	3,20	3,20	2,40	0	0	0	1,20	1,20	0,90	1,60	1,60	1,20
32 cores	2,80	2,80	2,10	2,80	2,80	2,10	0	0	0	15,6	15,6	11,7	6,00	6,00	4,50	0	0	0	2,80	2,80	2,10	2,80	2,80	2,10

Obs.: Todos os valores da tabela devem ser multiplicados por 10^{-07}.

Figura 5.15: Tempo de transmissão do padrão de comunicação all-to-one

A Figura 5.16 apresenta os resultados para o padrão *all-to-all*. Este padrão possui o segundo maior impacto no tempo total de transmissão e possui um tempo relativamente alto se comparado com os padrões *one-to-all* e *all-to-one*. No entanto, o impacto está presente somente nas aplicações FT e IS, conforme Tabela 5.8, que apresentam transmissão nula e baixa, respectivamente, no padrão *one-to-one* de maior impacto. Embora no padrão *all-to-all* exista uma alta concorrência, parte do padrão é *one-to-all*. Portanto, para este padrão, os resultados mostram que a MCNoC atinge uma redução no tempo de transmissão em relação às topologias *mesh* e *torus* de até 33% e 17% para a aplicação BT, 36% e 21% para a aplicação EP, 34% e 29% para a aplicação FT, 34% e 29% para a aplicação IS, 35% e 21% para a aplicação LU, 37% e 22% para a aplicação MG, e 36% e 20% para a aplicação SP, respectivamente.

Outra característica apresentada pela Figura 5.16 diz respeito ao comportamento das aplicações FT e IS. Ambas possuem um tempo de transmissão mais alto para versões de NoCs que suportam oito núcleos. O principal motivo está relacionado ao número constante na quantidade de mensagens para 8, 16 e 32 núcleos. Devido a redução no tamanho da mensagem de 8 para 16 e 32 núcleos, o tempo de transmissão também diminui. De acordo com a Tabela 5.8, este comportamento é comum às aplicações FT e IS, sendo que o restante das aplicações possui o tamanho das mensagens igual nos três tipos de núcleos suportados.

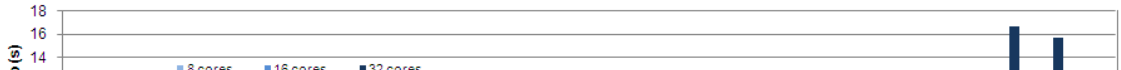


	mesh	torus	mcnoc	mesh	torus	mcnoc	mesh	torus	mcnoc	mesh	torus	mcnoc	mesh	torus	mcnoc	mesh	torus	mcnoc	mesh	torus	mcnoc	mesh	torus	mcnoc
	BT			CG			EP			FT			IS			LU			MG			SP		
8 cores	6,27	5,94	4,60	0	0	0	6,03	5,61	3,04	0,08	0,07	0,06	0,16	0,15	0,11	1,99	1,87	1,06	1,30	1,20	0,669	5,97	5,55	4,29
16 cores	2,00	1,75	1,28	0	0	0	3,55	3,02	2,04	0,057	0,053	0,038	0,09	0,08	0,061	9,28	7,85	5,31	7,60	6,40	4,00	1,99	1,73	1,28
32 cores	11,0	9,00	7,41	0	0	0	15,0	12,0	9,93	0,04	0,038	0,027	0,069	0,064	0,045	4,14	3,38	2,67	0,003	0,002	0,002	11,0	9,10	7,24

Obs.: Para a aplicação BT, EP e SP os valores da tabela devem ser multiplicados por 10^{-06} (8 cores) e 10^{-05} (16 e 32 cores). Para a aplicação LU os valores da tabela devem ser multiplicados por 10^{-05} (8 e 16 cores) e 10^{-04} (32 cores). Os valores da aplicação MG devem ser multiplicados por 10^{-04} (8 e 16 cores).

Figura 5.16: Tempo de transmissão do padrão de comunicação all-to-all

Considerando o tempo total de transmissão (soma de todos os padrões), é possível perceber a forte influência do padrão *one-to-one*, sendo que para as aplicações FT e IS, os resultados encontrados no padrão *all-to-all* prevalecem. Portanto, a MCNoC reduz o tempo de transmissão de pacotes em relação às topologias *mesh* e *torus* de até 26% e 22% para a aplicação BT, 18% para a aplicação CG, 36% e 21% para a aplicação EP, 33% e 28% para a aplicação FT, 34% e 29% para a aplicação IS, 14% para a aplicação LU, 23% e 20% para a aplicação MG, e 26% e 22% para a aplicação SP, respectivamente.



	BT			CG			EP			FT			IS			LU			MG			SP		
	mesh	torus	mcnoc	mesh	torus	mcnoc	mesh	torus	mcnoc	mesh	torus	mcnoc	mesh	torus	mcnoc	mesh	torus	mcnoc	mesh	torus	mcnoc	mesh	torus	mcnoc
8 cores	3,59	3,49	2,83	0,31	0,31	0,27	6,03	5,61	3,04	0,08	0,077	0,061	0,16	0,15	0,11	1,42	1,42	1,19	0,22	0,22	0,19	6,25	6,08	4,93
16 cores	5,68	5,29	4,4	0,8	0,79	0,66	3,55	3,02	2,04	0,057	0,053	0,038	0,09	0,08	0,061	2,16	2,16	1,86	0,31	0,31	0,25	9,48	8,83	7,36
32 cores	9,58	9,03	7,03	1,1	1,1	0,9	1,50	1,20	0,99	0,04	0,038	0,027	0,069	0,064	0,045	3,68	3,68	3,15	0,42	0,41	0,32	16,66	15,71	12,24

Obs.: Para a aplicação EP os valores da tabela devem ser multiplicados por 10^{-06} (8 cores), 10^{-05} (16 cores) e 10^{-04} (32 cores).

Figura 5.17: Tempo de transmissão considerando todos os padrões de comunicação

Para finalizar a avaliação de desempenho, a Figura 5.18 apresenta os resultados para pacotes de até 4096 *bytes*, conforme suporte do protocolo da Figura 4.23. Resultados preliminares da Subseção 5.2.1 mostram a baixa redução no tempo de transmissão pela MCNoC em relação às topologias *mesh* e *torus*, a medida que os pacotes aumentam de tamanho. Apesar desta baixa redução, é possível identificar que nem todas as aplicações possuem mensagens grandes, como é o caso da aplicação EP, que possui pacotes de 40 *bytes* de tamanho máximo. Conseqüentemente, a MCNoC que suporta 32 núcleos, reduz o tempo de transmissão de pacotes em relação às topologias *mesh* e *torus*, que suportam 32 núcleos, em até 0,85% e 0,58% para a aplicação BT, 0,54% para a aplicação CG, 36% e 21% para a aplicação EP, 1,13% e 0,80% para a aplicação FT, 1,49% e 0,94% para a aplicação IS, 0,19% para a aplicação LU, 1,22% e 0,89% para a aplicação MG, e 0,75% e 0,52% para a aplicação SP, respectivamente. Portanto, a aplicação EP possui a mesma redução apresentada pela Figura 5.17, onde o tamanho máximo dos pacotes é de 128 *bytes*.

	BT			CG			EP			FT			IS			LU			MG			SP		
	mesh	torus	mcnoc	mesh	torus	mcnoc	mesh	torus	mcnoc	mesh	torus	mcnoc	mesh	torus	mcnoc	mesh	torus	mcnoc	mesh	torus	mcnoc	mesh	torus	mcnoc
8 cores	2,85	2,85	2,83	0,27	0,27	0,27	6,03	5,61	3,04	0,061	0,061	0,061	0,12	0,11	0,11	1,21	1,21	1,19	0,19	0,19	0,19	4,97	4,97	4,93
16 cores	4,44	4,43	4,4	0,66	0,66	0,66	3,55	3,02	2,04	0,039	0,039	0,038	0,062	0,062	0,061	1,87	1,87	1,86	0,25	0,25	0,25	7,42	7,4	7,36
32 cores	7,09	7,08	7,03	0,91	0,91	0,9	1,50	1,20	0,99	0,027	0,027	0,027	0,046	0,046	0,045	3,16	3,16	3,15	0,33	0,33	0,32	12,33	12,3	12,24

Obs.: Para a aplicação EP os valores da tabela devem ser multiplicados por 10^{-06} (8 cores), 10^{-05} (16 cores) e 10^{-04} (32 cores).

Figura 5.18: Tempo de transmissão considerando todos os padrões de comunicação (pacote de tamanho máximo igual a 4096 bytes)

O tempo, em se tratando de métrica para avaliação de desempenho, é a métrica mais adequada para uma conclusão final. Apesar dos pacotes com tamanho máximo de 4096 *bytes* atingirem um tempo final menor do que os de 128 *bytes*, conforme é possível verificar pelas Figuras 5.18 e 5.17, respectivamente, pacotes grandes podem reduzir a vazão do sistema e prejudicar o processamento de aplicações com pacotes menores, tais como a aplicação EP. Em um contexto *intra-chip*, onde é possível o processamento de múltiplas aplicações distribuídas em *clusters* de processamento, o compartilhamento de

roteadores e rede por pacotes muito grandes pode ocasionar em uma reserva de *links* de comunicação por tempo muito grande, o que reduz o paralelismo que a rede pode suportar. Portanto, para sistemas com várias aplicações paralelas executando simultaneamente, a vazão de resultados pode ser muito importante, e nesse caso, pacotes menores de até 128 *bytes* podem trazer um melhor desempenho para o sistema. Isso reforça a arquitetura baseada em múltiplos *clusters* da MCNoC que atinge uma redução em tempo de transmissão de até 36% em relação às topologias *mesh* e *torus*, justamente para uma aplicação (EP) que não possui pacotes de tamanho superior a 40 *bytes*.

O grande problema para as NoCs com topologias *mesh* e *torus*, como já foi descrito, é justamente a influência das latências de roteadores para decisão de encaminhamento de pacotes. Muitos pacotes demandam muitas latências, mas uma forma de solucionar este problema seria a inclusão de *buffers* de saída ao invés, ou em conjunto, com *buffers* de entrada. Os *buffers* de saída ilustrados pela Figura 5.19 podem reduzir a latência existente entre pacotes, uma vez que todos os pacotes já se encontram em filas de saída. Através desta técnica é possível escrever o novo pacote na posição seguinte ao fim do último pacote já existente no *buffer* de saída escolhido. Conseqüentemente, haveria uma saída de pacotes em *pipeline* sem as lacunas que representam as latências de leitura de cabeçalho e encaminhamento. No entanto, como já foi descrito na Subseção 2.1.4.1, o uso de *buffers* de saída implica em uma chave *crossbar* com uma velocidade de encaminhamento mais rápida em relação à opção de *buffers* de entrada. São N entradas competindo por uma única saída, e isto pode demandar uma freqüência de operação mais alta, *buffers* de maior capacidade e, portanto, uma maior ocupação de área e consumo de potência. No entanto, existe a probabilidade do mesmo projeto, sem otimizações e com *buffers* maiores, ter a freqüência de operação reduzida.

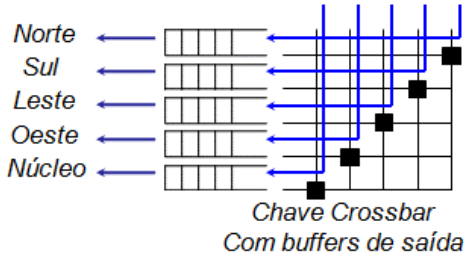


Figura 5.19: Buffers de saída para topologias mesh e torus

Os resultados encontrados são baseados em uma modelagem de avaliação onde as versões de NoCs não possuem contenção em *buffers* de entrada. Estudos relacionados às profundidades dos *buffers* são importantes inclusive para definição de tamanhos adequados de pacotes. No entanto, tamanhos de pacotes também influenciam na tomada de decisão dos tamanhos dos *buffers*. A dependência que existe para as tomadas de decisões relativas aos tamanhos de pacotes e *buffers* é justamente a caracterização e definição de cargas de trabalhos para o contexto de arquiteturas *many-core* com sistemas integrados de NoC e memória, processos de aplicações distribuídas e do Sistema Operacional. Por este motivo, a Seção 6.1 apresenta trabalhos futuros que focam justamente em estudos que possam contribuir com arquiteturas e cargas de trabalho, que ainda não estão claros no estado da arte para este novo contexto de interconexões baseadas em NoCs. Os tamanhos dos *buffers* utilizados na avaliação de área das NoCs foram escolhidos como uma constante para avaliação de ocupação do FPGA, mas através dos resultados de avaliação de desempenho é possível estimar a

profundidade dos *buffers* baseada nas cargas de trabalho NAS, através dos seguintes critérios:

- Pacotes menores de tamanho máximo de 128 *bytes* exploram melhor a capacidade de paralelismo e vazão da NoC.
- A perda de pacotes menores possui um custo menor de retransmissão.
- Os programas do NAS possuem em sua maioria pacotes de tamanho próximos a 128 *bytes* em maior quantidade do que pacotes pequenos.
- *Buffers* pequenos são mais adequados para técnica de encaminhando *wormhole* usada pelas versões de NoCs avaliadas.
- *Buffers* pequenos são mais adequados para a técnica de chaveamento por circuito utilizado pela MCNoC na implementação de topologias em *clusters* e entre *clusters* próximos. Neste caso, a necessidade de enfileiramento se faz necessário para estabelecer o início de uma comunicação, ou para processamento pelo roteador.
- *Buffers* maiores são necessários para técnica de chaveamento por pacotes, que é adotada pela MCNoC em casos de um número muito grande de roteadores intermediários.

Portanto, *buffers* de profundidade entre 3 (12 *bytes* – escolhido para avaliação de área) e 5 (20 *bytes*) podem alocar os pacotes menores, que são responsáveis pelo estabelecimento de comunicação, processamento pelo roteador, além de enfileiramento temporário. Os pacotes maiores são recebidos após confirmação de espaço em *buffer*, mas devem ser encaminhados logo em seguida seguindo a técnica *wormhole*. Se necessário, pacotes maiores podem ser alocados em bancos de registradores do NPoC para processamento de dados de aplicação. Considerando 4 bancos de registradores para suporte a IMT, é possível que apenas 1 banco possa suportar o armazenamento temporário de um pacote de até 128 *bytes* (tamanho máximo estabelecido na avaliação de programas do NAS).

Outra avaliação dos resultados pode ser feita através de uma analogia com a Função de Isoeficiência. Apesar do número de processadores do sistema ser fixo na comparação entre a MCNoC e topologias *mesh* e *torus*, a quantidade de roteadores varia. Nesse caso, o aumento da quantidade de roteadores da MCNoC para *mesh* e *torus*, faz com que as arquiteturas de NoCs tenham uma largura de banda de valores diferentes sempre que o tipo de padrão de comunicação coletiva mudar. Esta diferença pode ser explicada, por exemplo, pelo padrão *one-to-all*, onde a vazão, medida em *bytes/s* tal como a largura de banda, aumenta para a MCNoC e se mantém constante para as topologias *mesh* e *torus*. Apesar de vazão e largura de banda significarem a capacidade de escoamento de pacotes, portanto, em alguns casos são usados como sinônimos, o uso de dois *links* com iguais larguras de banda para enviar pacotes para dois destinos, simultaneamente, duplica a vazão de entrega de dados. Justamente nesta capacidade, que há uma redução na eficiência das NoCs baseadas em *mesh* e *torus*, uma vez que o aumento da quantidade de roteadores, para processar a mesma carga de trabalho, não reduz o tempo de transmissão se comparado com o tempo de transmissão da MCNoC, que usa uma quantidade menor de roteadores. Em outras palavras, as topologias *mesh* e *torus* mantêm fixo o tamanho total de pacotes transmitidos, mas aumentam a quantidade de roteadores para processar o envio dos pacotes. Este aumento dos roteadores, tal como o

aumento de processadores explicitado pela Função de Isoeficiência, reduz a eficiência da rede, uma vez que o tamanho total dos pacotes enviados permanece o mesmo.

Em oposição às desvantagens do número de roteadores na redução do tempo de transmissão, as topologias *mesh* e *torus* possuem na quantidade de roteadores e *links* a vantagem da contingência e tolerância à falhas. Defeitos ocasionados pelo tempo de uso (AGGARWAL, 2007) podem reduzir o número de roteadores, que em uma arquitetura em *cluster* pode ser mais prejudicial, uma vez que mais núcleos ficariam ilhados do restante do *chip*. Isto forçaria, por exemplo, reconfiguração de primeiro nível (FPGA), ou métodos de acessos à memória que podem degradar o desempenho no processamento de aplicações. Além disso, em situações de alto tráfego em determinados *links*, a maior quantidade destes *links*, aumentam as opções de roteamento. Porém, arquiteturas baseadas em *mesh* e *torus*, que sejam capazes de solucionar problemas de alto tráfego com novas rotas, também são mais complexas, podem ser maiores e consumirem mais potência. Um exemplo é justamente os *buffers* de saída ilustrados na Figura 5.19. Neste caso, roteadores programáveis usando o segundo nível de reconfiguração podem aumentar a flexibilidade mantendo uma melhor relação com o custo.

5.4 Avaliação do Tempo de Execução de Programas no NPoC

O objetivo desta seção é descrever a avaliação de desempenho do processador de rede NPoC com e sem IMT através da execução de alguns programas específicos. A Tabela 5.10 apresenta quatro tipos de programas executados em ArchC na arquitetura NPoC e MIPS, ambos sem IMT. Os resultados da arquitetura com suporte a quatro *threads* por IMT são baseados em um modelo analítico que possui como entradas os resultados de simulação do modelo sem IMT. Os três primeiros programas foram escritos em linguagem de montagem e simulados no NPoC, enquanto que o programa Dijkstra foi compilado e executado na arquitetura MIPS. Por se tratar de um programa relativamente grande em comparação com os demais, e da similaridade do ISA de propósito geral entre o NPoC e MIPS, pequenas diferenças na quantidade de ciclos não alteram a avaliação realizada na Tabela 5.10 e Figura 5.21. A descrição de cada programa é a seguinte:

- ComPat: Programa específico para monitorar padrão de comunicação nos *buffers* de entrada do roteador RANoC. Este programa realiza as operações principais, que estão descritas pela Figura 4.20b.
- TopPat: Este programa apresenta as instruções principais para implementação de topologias. O programa é baseado em instruções do tipo *load* (busca palavra), *reconf* (envia para Decodificador) e *jump* (repete tarefa).
- Factorial: Um programa de propósito geral para cálculo de fatorial.
- Dijkstra: Programa para calcular o caminho entre dois roteadores.

O programa ComPat pode ser ilustrado pelo exemplo da Figura 5.20 através da linguagem de montagem do processador NPoC. O funcionamento básico desta versão do ComPat é monitorar os oito *buffers* de entrada do RANoC através da instrução *read*, identificar a requisição de novo padrão ao fim das oito comunicações através de instruções do tipo *jump* condicional, e implementar a nova topologia através da instrução *reconf*. As instruções iniciais *addi* são utilizadas para iniciar registradores.

```
        addi $1, $0, 9
        addi $3, $0, 1
        addi $6, $0, 0
        addi $7, $0, 8
        addi $8, $0, 6
        addi $10, $0, L1
        addi $11, $0, L2
        addi $12, $0, 100
        addi $13, $0, 106
        addi $14, $0, L4

    L1: load $8, $12, 0
        addi $2, $0, 1
        addi $4, $0, 1

    L3: jeq $10, $1, $2
        read $5, $2, 0
        jdi $11, $5, $3
        addi $4, $4, 1
        write $6, $2, 0
        jdi $11, $4, $7
        addi $4, $0, 1
        reconf $8
        addi $12, $12, 1
        jeq $14, $12, $13
    L2: addi $2, $2, 1
        addi $13, $13, 1
        jump $9, $0, L3
    L4: ......
```

Figura 5.20: Programa ComPat em linguagem de montagem NPoC

Na Tabela 5.10 são apresentados os CPIs relativo e absoluto, o número de ciclos de execução e o tempo de execução para cada programa rodando em 50 MHz (freqüência usada pelo ArchC para avaliação de resultados de execução de programas). O CPI relativo considera a influência do chaveamento entre *threads*, e o CPI absoluto considera somente uma *thread* e as vantagens da eliminação de paradas no *pipeline* pela utilização da técnica IMT. Os programas específicos de controle da RCS-NR são muito simples e considerando que ambos possam atrasar o envio de pacotes através de uma implementação de topologia ou identificação de mudança de padrão de comunicação, o impacto é muito baixo. O impacto do uso de IMT no NPoC mostra que o CPI relativo aumenta em relação ao CPI, uma vez que todos os programas devem intercalar a execução de instruções. O CPI relativo igual a 4 considera que todos os programas estão sendo re-executados sempre que chegam ao fim. No entanto, a janela de re-execução é variável em função de regras de escalonamento. Por este motivo, o programa Dijkstra* rodando em arquitetura com IMT pode ter o CPI relativo bem próximo de 1, conforme encontrado na arquitetura sem IMT. Este efeito acontece em função da não re-execução dos demais programas.

Tabela 5.10: Execução de programas em versões do NPoC (FREITAS, 2009d)

Programs	*Without IMT*			*With 4-IMT*			
	CPI	Cycles	Time	RCPI	ACPI	Cycles	Time
ComPat	1.03	27	0.54µs	4	1	104	2.08µs
TopPat	1.20	6	0.12µs	4	1	20	0.40µs
Factorial	1.18	259	5.18µs	4	1	876	17.50µs
Dijkstra	1.00	59355000	1.18710s	4	1	237412024	4.74824s
Dijkstra*				1	1	59356000	1.18712s

Apesar do tempo de execução independente de cada programa, o uso de IMT aumenta consideravelmente o desempenho de execução das aplicações. A Figura 5.21 ilustra o efeito do chaveamento de contexto sem e com IMT na execução dos quatro tipos de programas mencionados. Devido aos quatro bancos de registradores na arquitetura IMT, todos os contextos necessários para executar os programas (*threads*) já

estão presentes. Considerando que a penalidade para troca de contexto sem IMT é de 42 ciclos (1 ciclo para cada palavra em leitura e escrita mais esvaziamento de *pipeline*), a redução no tempo final de execução de todos os programas, sem re-execução e considerando que não há *cache miss* (e.g., *load* e *store*), é praticamente inexistente, conforme é apresentado pelas Figuras 5.21a e 5.21b. No entanto, considerando que outra execução do programa Factorial está no lugar do Dijkstra, o que aproxima os programas em termos de quantidade de ciclos, a redução no tempo final é significativa, chegando a 7,1 %, conforme apresenta as Figuras 5.21c e 5.21d.

As vantagens do uso do IMT vão além da redução do tempo de execução, já que é possível aumentar a vazão de resultados dos programas. Em se tratando de processamento de rede, da existência de diversos tipos de pacotes, padrões de comunicação e da possibilidade de processamento ativo e de propósito geral, a demanda por resultados mais rápidos é grande.

Uma das características da arquitetura de roteador programável é a possibilidade de aumentar o desempenho pela flexibilidade. No entanto, um programa de gerenciamento pode também adicionar tempo ou atraso no encaminhamento de pacotes. O programa ComPat tem a função de identificar padrões de comunicação através da leitura dos *buffers* de entrada do roteador. O impacto desta identificação depende do tempo de processamento do próprio ComPat, do tempo entre o fim e início de um novo padrão, além do tempo total para transmissão de todos os pacotes. O tempo total está apresentado na subseção anterior, e o tempo entre mudanças de padrões é apresentado pela Tabela 5.11 (tamanho máximo de pacotes = 128 *bytes*).

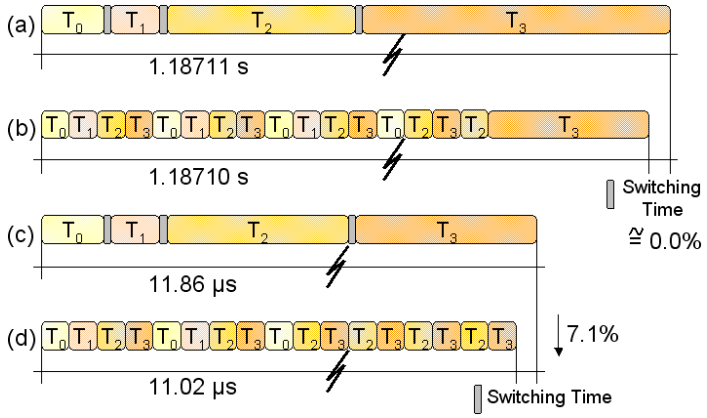


Figura 5.21: Ganho de desempenho do NPoC com suporte IMT (FREITAS, 2009d)

A ferramenta de análise desenvolvida em Python lê a informação de fim e início de transmissões entre padrões diferentes e armazena os menores e maiores valores identificados. A transição entre padrões pode ocorrer em períodos rápidos, tal como apresentado pela aplicação LU (1,95 µs), períodos longos como apresentado pela aplicação (731,69 ms), ou pela ausência de períodos de transição entre padrões, como apresentado pelas aplicações BT e SP.

O período de transição entre padrões é importante para evitar atrasos em comunicações. Portanto, a informação vinda do SO (primeira abordagem de gerenciamento) somada com as instruções de implementação de topologias deve ocorrer em tempo menor do que o período de transição. A implementação de topologias é representada pelo programa TopPat que possui um conjunto de instruções muito pequeno, conseqüentemente, poucos ciclos de execução. Considerando apenas o TopPat sendo executado em 100 MHz (freqüência das versões das NoCs com cargas NAS de

128 *bytes*), o tempo para finalizar o programa é de 0,06 µs sem IMT e de 0,2 µs com IMT. Considerando os tempos de transições presentes na Tabela 5.11, o tempo apenas do TopPat é suficiente para não adicionar atraso na comunicação. Outra forma de aproveitar os períodos de transições presentes entre padrões é a predição de comunicação. Estratégia de identificação pelo SO e por um programa do RANoC para predição de comunicação são apresentados como trabalhos futuros no Capítulo 6.

Tabela 5.11: Tempos entre fim e início de um novo padrão (tamanho máximo de pacotes igual a 128 *bytes*)

Impacto	BT	CG	EP*	FT	IS	LU	MG	SP
Tempo Menor	0 s	125,94 µs	---	228,18 ms	69,72 µs	1,95 µs	67,95 µs	0 s
Tempo Maior	47,13 ms	125,94 µs	---	731,69 ms	119,36 ms	39 ms	285,18 ms	42,13 ms

* A aplicação EP possui apenas um padrão (Tabela 5.8).

A outra estratégia de identificação do novo padrão de comunicação é através do programa ComPat. Este programa não antecipa comunicação (predição), o ComPat lê as requisições nos *buffers* para que seja feita a implementação de nova topologia em conformidade com o novo padrão de comunicação. Portanto, o ComPat adiciona atraso no envio de pacotes, caso sua atuação ocorra no início da nova comunicação. Considerando que o ComPat é executado na mesma freqüência de operação das NoCs, que suportam o envio dos pacotes das aplicações NAS, seu tempo de execução sem IMT é de 0,27 µs e com IMT é de 1,04 µs. O *overhead* do ComPat, sendo executado quatro vezes (4 padrões) para uma aplicação NAS, no tempo total de transmissão de pacotes pela MCNoC é irrisório tanto sem IMT e com IMT e mantém as taxas de redução no tempo de transmissão em relação às topologias *mesh* e *torus* praticamente iguais. A exceção seria a aplicação EP, que possui tempos baixos de transmissão de pacotes, mas possui apenas um padrão de comunicação. A Tabela 5.12 apresenta as alterações na redução de desempenho pelo uso do ComPat focando apenas os resultados para 8 núcleos, onde existe a garantia de um mapeamento em núcleos próximos e, portanto, com influência direta da gerência de apenas um roteador RANoC.

Tabela 5.12: Alterações pelo uso do ComPat na aplicação EP

8 núcleos	Fig. 5.17 (128 bytes) e Fig. 5.18 (4096 bytes)			Acréscimo ComPat sem IMT			Acréscimo ComPat com IMT		
	mesh	**Torus**	**mcnoc**	**mesh**	**torus**	**mcnoc**	**mesh**	**torus**	**mcnoc**
	6,03 µs	5,61 µs	**3,04 µs**	6,03 µs	5,61 µs	**79.5 µs**	6,03 µs	5,61 µs	**80.2 µs**

O programa ComPat só influencia na MCNoC, permanecendo as topologias *mesh* e *torus* com os mesmos valores. No entanto, o impacto na redução do tempo de transmissão é baixo quando o ComPat é adicionado ao tempo da MCNoC. Originalmente, a redução atinge 34% e 29% em relação às topologias *mesh* e *torus* (suporte a 8 núcleos), respectivamente. O ComPat sem suporte a IMT mantém uma boa taxa de redução no tempo de transmissão pela MCNoC, a qual atinge até 30% e 24% em

relação às topologias *mesh* e *torus*, respectivamente. No caso do impacto do ComPat com IMT, a MCNoC reduz o tempo de transmissão em 17% e 11% em relação às topologias *mesh* e *torus*, respectivamente. É importante ressaltar que o impacto na aplicação EP é maior devido ao fato de haver apenas um padrão com tempos muito baixos de transmissão de pacotes. As demais aplicações possuem tempos altos que superam 60 ms, o que mantém as porcentagens de redução praticamente iguais, conforme é apresentado pela Tabela 5.13. A aplicação FT possui três padrões e o programa ComPat é executado três vezes. Neste caso, as porcentagens referentes à redução no tempo de transmissão pela MCNoC em relação às topologias *mesh* e *torus* permanecem iguais em 23% e 21%, respectivamente.

Tabela 5.13: Alterações pelo uso do ComPat na aplicação FT

8 núcleos	**Fig. 5.17 (128 bytes) e Fig. 5.18 (4096 bytes)**			**Acréscimo ComPat sem IMT**			**Acréscimo ComPat com IMT**		
	mesh	**Torus**	**Mcnoc**	**mesh**	**torus**	**mcnoc**	**mesh**	**torus**	**mcnoc**
128 bytes	0,08005 s	0,07781 s	**0,06117 s**	0,08005 s	0,07781 s	**0,06117 s**	0,08005 s	0,07781 s	**0,06117 s**
4096 bytes	0,06150 s	0,06148 s	**0,06117 s**	0,06150 s	0,06148 s	**0,06117 s**	0,06150 s	0,06148 s	**0,06117 s**

No entanto, o tempo do ComPat pode não adicionar atraso se o protocolo de comunicação entre os núcleos enviar um pacote de fim de transmissão. Desta forma, o ComPat processaria pacotes de fim de padrão, conforme programa apresentado na Figura 5.20, e resultados da Seção 5.3, e não de início de requisição de padrão. Portanto, o tempo do ComPat adicionado ao fim de um padrão ocuparia o período de transição sem adicionar atraso pela implementação de topologias. As exceções ocorrem quando não há período de transição em BT e SP.

Durante a avaliação da arquitetura da MCNoC alguns pontos foram identificados como trabalhos futuros. Por esta razão, o próximo capítulo apresenta, além das conclusões da tese, descrições de possíveis atividades de pesquisa que possam agregar valor aos resultados alcançados nesta tese.

6 CONCLUSÃO

O surgimento de processadores *many-core* é uma questão de tempo. Esforços para viabilizar o crescimento das taxas de desempenho focam em arquiteturas com múltiplos núcleos explorando paralelismo no nível de *thread*. Como conseqüência, várias oportunidades de pesquisa estão surgindo para solucionar problemas derivados da inserção de vários núcleos em *chip*. Um dos principais problemas é a rede-em-*chip*, responsável por estabelecer a comunicação dos núcleos de processamento e outros periféricos. A literatura aponta a necessidade de novas abordagens de arquitetura em oposição às soluções tradicionais de interconexão, e as *Networks-on-Chip* têm se apresentado como a principal alternativa.

Embora as NoCs sejam apontadas como o caminho a seguir, explorar desempenho em um contexto nativamente paralelo (vários núcleos disponíveis) não é trivial e requer o conhecimento das variáveis que podem influenciar no ganho final do *chip*. Um dos principais motivos de queda de desempenho são as próprias redes de comunicação. À medida que são intensamente utilizadas; contenções e atrasos de pacotes podem acontecer nos roteadores. Esta latência pode acarretar em um aumento do tempo necessário para processar uma determinada aplicação. No mesmo sentido, é necessário que a rede possua uma boa largura de banda e arquitetura que privilegie a vazão de pacotes, uma vez que o tamanho do problema a ser resolvido pode acarretar na necessidade de alto desempenho de comunicação entre núcleos e periféricos.

O estado da arte relacionado às NoCs mostra que o estudo da carga de trabalho, além de ser essencial, tem sido cada vez mais abordado para o projeto de NoCs. Redes que possuam uma arquitetura dedicada para processar um determinado domínio de aplicação ganham em desempenho, mas perdem em flexibilidade. Cargas de trabalho para GPPs são de diversos tipos, e à medida que processadores com múltiplos núcleos são apresentados pela indústria, a necessidade de desenvolver programas paralelos também aumenta. Cargas de trabalho paralelas possuem características típicas de comunicação coletiva, e para cada tipo de comunicação existe um padrão de comportamento no envio e recebimento de pacotes. O projeto de NoCs mais flexíveis é importante para atender cargas paralelas diversas.

Na busca de mais flexibilidade, pesquisadores têm utilizado os conceitos de computação reconfigurável para aumentar a capacidade das NoCs de se adaptarem aos padrões de comunicação ou domínios de aplicação. A principal alternativa tem sido a utilização de FPGAs como forma de flexibilizar a implementação ou reconfiguração de novas arquiteturas de NoCs. Alternativas apontam para o uso de *wireless* e para a

exploração de PASICs. Os ASICs polimórficos são abordagens interessantes que aumentam o desempenho de reconfiguração dinâmica sem o *overhead* de um dispositivo programável, como o FPGA. No entanto, os PASICs perdem em flexibilidade e, portanto, não possuem a mesma capacidade de adicionar recursos sob demanda.

A arquitetura da MCNoC tem por objetivo explorar o segundo nível de reconfiguração para implementação de novas topologias que possam atender às mudanças nos padrões de comunicação. Aliado a esta flexibilidade, o roteador RANoC possui um processador de rede que possibilita o uso de programas de gerenciamento e monitoramento da rede. Além da flexibilidade e capacidade de se adaptar dinamicamente sem necessidade de reconfigurar o FPGA, a MCNoC também explora a arquitetura baseada em *clusters* de núcleos de processamento para reduzir a influência da rede no processamento das aplicações. Desta forma, é possível explorar o processamento paralelo em *clusters* menores ou vizinhos sem longos saltos ou influência de roteadores. Esta organização propicia futuros testes para processamento paralelo e distribuído, uma vez que o mapeamento de aplicações pode distribuir em *clusters* diferentes, aplicações paralelas diferentes (Subseção 6.1.4).

A possibilidade de reconfigurar os protótipos da MCNoC no primeiro nível (e.g., FPGA) amplia as possibilidades de avaliações das arquiteturas que podem mapear padrões lógicos de comunicação. Portanto, o uso de *hardwares* programáveis aumenta a flexibilidade e tem como principal vantagem sobre alternativas baseadas em ASIC a própria arquitetura reconfigurável. ASICs exploram arquiteturas dedicadas com altas freqüências, mas com a mudança de padrões, principalmente em contextos com um número muito grande de aplicações, pode demandar novas arquiteturas de NoCs. Neste caso, mesmo com freqüências de operações menores, NoCs reconfiguráveis baseadas em FPGAs, ou em PASICs no primeiro nível, exploram as vantagens das arquiteturas adaptáveis para vencer barreiras de freqüências e ganhar em desempenho em comparação com ASICs. Em processadores *many-core* com milhares de núcleos, os conceitos associados à *Grid-Computing* podem estar presentes. Neste caso, experimentos voltados para *Grids-on-Chips* (Subseção 6.1.6) podem demandar plataformas reconfiguráveis como o FPGA ou PASICs para se alcançar ganhos de desempenho.

Os resultados baseados nos modelos analíticos e de simulação mostram que a arquitetura da MCNoC explora a vazão de pacotes e possui alto desempenho para diferentes padrões de comunicação coletiva. Os testes também mostram que a arquitetura baseada em roteadores programáveis para *clusters* de núcleos de processamento possui uma boa ocupação de área no FPGA em comparação com duas arquiteturas tradicionais de NoCs. Devido à menor quantidade e utilização dos *buffers* de entrada, o consumo de potência também se mostrou melhor. Através dos tempos de transmissão e consumo de potência foi possível constatar uma melhor eficiência no consumo de energia. As avaliações realizadas através das cargas de trabalho paralelas do NAS mostram que a arquitetura da MCNoC possui uma boa vantagem para transmissão de pacotes em um contexto de comunicação coletiva. Pacotes menores possuem um ganho maior em relação às topologias *mesh* e *torus*. Além disso, é possível associar vazão e melhor exploração de *links* de comunicação, visando concorrência entre comunicações, através de pacotes menores. No que diz respeito ao impacto dos programas do processador NPoC no envio de pacotes, é mantida as vantagens arquiteturais da MCNoC em relação às topologias *mesh* e *torus*. Um comportamento

importante, identificado na execução das cargas de trabalho, é o período ocioso entre o fim e início de padrões de comunicação diferentes, que inviabilizam reconfiguração dinâmica em FPGA em vários casos. Este período ocioso propicia a exploração pela MCNoC para antecipar a implementação de nova topologia na RCS-NR e também de algoritmos baseados em protocolos específicos ou de predição de comunicação, conforme é descrito na Subseção 6.1.3.

Portanto, os modelos artificial e natural utilizados para verificação do funcionamento, comportamento e desempenho mostraram que a arquitetura da MCNoC é uma alternativa para suportar padrões de comunicação coletiva, com potencial para trabalhos futuros relacionados, conforme é descrito nas seções seguintes deste capítulo. Além disso, é importante ressaltar que se houver a necessidade de uma nova estratégia de gerenciamento, basta um novo programa para os roteadores da MCNoC, ao invés de uma mudança na arquitetura da NoC, que ocorre nas arquiteturas específicas e dedicadas. Conseqüentemente, a programabilidade da MCNoC torna a arquitetura mais adaptável sem necessidade de mudanças arquiteturais e novas implementações em função de novos padrões de comunicação. Portanto, o conjunto de características da arquitetura da MCNoC, que não estão presentes entre as arquiteturas apresentadas nos Trabalhos Correlatos, são: topologias adaptáveis, gerenciamento de *clusters* de *cores*, e arquitetura programável e reconfigurável. Considerando as hipóteses apresentadas na Seção 1.2, é possível concluir através dos resultados que:

- A adaptação de topologias, através do segundo nível de reconfiguração, para mapear os padrões de comunicação coletiva aumenta o desempenho.
- Os roteadores programáveis são viáveis, aumentam a flexibilidade e o desempenho no gerenciamento das topologias implementadas na chave *crossbar* reconfigurável.
- A redução do número de roteadores, através da clusterização, reduz a latência e o tempo de transmissão de pacotes.

Finalizando, de acordo com os objetivos definidos na Introdução e a metodologia escolhida, foi possível verificar através dos resultados alcançados, que a MCNoC responde ao problema principal desta tese. A MCNoC é uma arquitetura adaptável que reduz tempo de transmissão de pacotes na busca por maior eficiência de rede no suporte aos padrões de comunicações coletivas.

Entre os principais retornos da comunidade científica a respeito da contribuição desta tese estão:

- Publicação no WSCAD (FREITAS, 2006b) sobre o projeto do processador de rede NPoC. Nesta ocasião, o artigo recebeu o prêmio de melhor artigo completo da edição WSCAD 2006.
- Publicação no IET Electronics Letters (FREITAS, 2008b) sobre o roteador RANoC, que tem por objetivo publicar idéias novas e originais.
- Publicação no FPL (FREITAS, 2008c) sobre a arquitetura da MCNoC, como alternativa de NoC adaptável para as próximas gerações de processadores *many-core*.
- Publicação de resumo no NOCS (FREITAS, 2009c) sobre a avaliação de desempenho da MCNoC com cargas paralelas NAS. O NOCS (*International Symposium on Networks-on-Chips*) é a principal conferência da área, por se

tratar de uma comunidade específica em NOCs com taxa de aprovação de artigos completos de 23% na edição de 2009.

6.1 Trabalhos Futuros

Esta seção apresenta os principais trabalhos futuros que podem contribuir com os resultados alcançados nesta tese.

6.1.1 Simulador em ArchC e SystemC da MCNoC

Simuladores de alto nível são muitas vezes mais adequados para avaliação de desempenho, uma vez que não há preocupação com questões, e.g., de implementação de circuitos que aumentam as camadas envolvidas na simulação. Por este motivo, o desenvolvimento em ArchC e SystemC de uma versão da MCNoC pode acelerar avaliações não realizadas nesta tese, que requerem um maior controle de variáveis / parâmetros da arquitetura da MCNoC. Além disso, um ambiente desenvolvido em ArchC e SystemC possibilita a interação entre núcleos e NoC, que não é encontrado em outros ambientes do estado da arte.

Alguns exemplos de avaliações que podem ser facilitadas pelo simulador:

- Identificação de gargalos em *buffers* de entrada, decorrentes de situações específicas de concorrência, avaliações de profundidade de *buffers*, perdas de pacotes e possíveis soluções de gerenciamento.
- Projeto e avaliação de novos protocolos, qualidade de serviço, e novas estratégias de gerenciamento concorrente.
- Execução de cargas de trabalho reais paralelas ou seqüenciais.
- Integração e avaliações com sistemas de memória.

6.1.2 Benchmarking Baseado em Cargas Paralelas para NoCs

Um dos grandes problemas na área é encontrar cargas de trabalho específicas para NoC. Considerando as arquiteturas de processadores *many-core* e NoCs com grande quantidade de roteadores, a demanda por programas paralelos aumenta. Por este motivo, o desenvolvimento de um *benchmarking* que represente um ambiente adequado para teste de programas paralelos teria grande utilidade para avaliação de NoCs.

Uma das alternativas iniciais seria a alteração de traços coletados de cargas reais para que estes possam ter um conjunto de informações maior que possibilite simulações específicas de arquiteturas de NoC. Um dos pontos principais está condicionado aos cabeçalhos. Extrair traços, adicionar cabeçalhos específicos e quebrar as mensagens em tamanhos de pacotes em conformidade com o contexto NoC faz aumentar as oportunidades de avaliação.

Em conjunto com o trabalho realizado nos traços, o uso de simuladores baseados em SystemC (MCNoC e outros), ajudam a criar o ambiente para teste e avaliação capaz de acelerar experimentos voltados para NoCs e cargas de trabalho paralelas.

6.1.3 Predição de Comunicação

Uma das formas para identificar um novo padrão de comunicação é através do monitoramento dos *buffers* de entrada do roteador RANoC. Durante a descrição dos

tempos relacionados aos períodos e transição entre padrões de comunicação, foi mencionado a possibilidade de exploração dos tempos ociosos para predição de comunicação (FREITAS, 2006d) (OGRAS, 2006).

A predição de comunicação tem por objetivo aumentar o desempenho da NoC sem que haja necessidade de esperar pelo encerramento ou solicitação de comunicação. A predição usaria o tempo de ociosidade de comunicação para identificar o padrão seguinte, implementar a topologia, e assim, preparar as interconexões para as novas comunicações. A predição seria interessante quando não existe informação de padrão de comunicação vindo do Sistema Operacional ou dos pacotes de rede. Há de se avaliar se o custo da predição é alto, inviabilizando o uso da mesma.

6.1.4 Mapeamento de Processos e Identificação de Padrões

Além de uma NoC de alto desempenho, o mapeamento dos processos (CARVALHO, 2007) em núcleos próximos cria condições adequadas para que o suporte da NoC adenda os requisitos de projeto. No caso da MCNoC, que tem como foco *clusters* de núcleos de processamento, um mapeamento mal feito pode resultar em comunicações distantes entre vários roteadores, eliminando as vantagens da clusterização e do mapeamento de topologias, capaz de ser implementado por cada roteador RANoC. Além disso, a identificação de padrões de comunicação por parte do sistema operacional em complemento às informações geradas em tempo de compilação, são importantes tanto para o mapeamento de processos, quanto para o processador NPoC, que em conjunto implementa topologias em conformidade com padrões de comunicação coletiva.

Portanto, estudos que envolvem compiladores e sistemas operacionais têm fundamental importância e estão diretamente ligados ao desempenho global do sistema (processador *many-core*).

6.1.5 Modelo de Programação Híbrido Visando Reconfiguração de NoCs

A MCNoC pode utilizar dois níveis de reconfiguração. A tese define o primeiro nível como estático e, portanto, usado somente quando da necessidade de alterar a arquitetura de interconexão entre roteadores, ou de blocos específicos. No entanto, a reconfiguração dinâmica usando o primeiro nível baseado em FPGAs, ou outros dispositivos programáveis, também pode ser realizada.

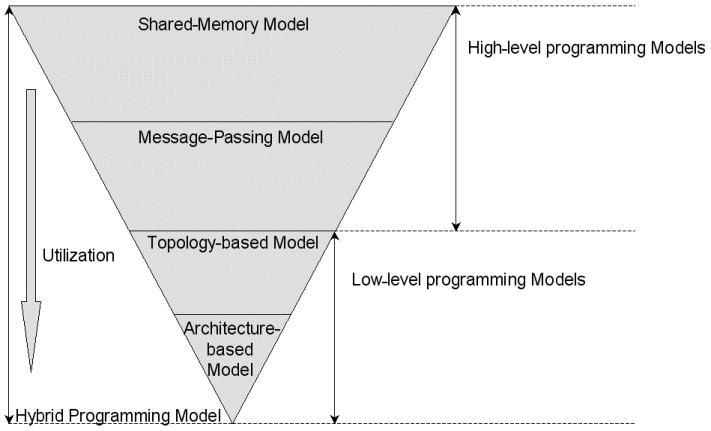


Figura 6.1: Modelo híbrido de programação (FREITAS, 2008d)

A proposta deste trabalho futuro seria um modelo de programação híbrido (FREITAS, 2008d), envolvendo os modelos convencionais de programação por passagem de mensagem e memória compartilhada, mas adicionando modelos que focam em reconfiguração da arquitetura da NoC, conforme Figura 6.1. Desta forma, seria possível explorar não só o segundo nível baseado na RCS-NR, mas também o primeiro nível de forma dinâmica. Resultados de experimentos, em dispositivos como o FPGA, são importantes como informações de entrada para o desenvolvimento deste modelo híbrido com suporte a reconfiguração dinâmica. Estes resultados servem como referências para identificação de alternativas que possam superar latências de reconfiguração.

6.1.6 Grids-on-Chips

Milhões de núcleos em um *chip* demandam uma rede-em-*chip* grande e com alto desempenho. O desenvolvimento de uma infra-estrutura de rede-em-*chip* capaz de suportar demandas de novas interconexões de forma transparente ao usuário ou programador, aumentando a quantidade de núcleos disponíveis ou aproximando estes núcleos em um *cluster* lógico ou físico, pode criar condições em que a computação em *Grid* esteja presente dentro do *chip*. O *Grid-on-Chip* (QIU, 2007) (FREITAS, 2008d) seria o equivalente ao *Grid Computing* (FOSTER, 2004) (FOX, 2007), mas em menor escala.

O uso de modelos híbridos de programação paralela (Figura 6.1), envolvendo níveis de reconfiguração, podem de forma transparente adaptar a infra-estrutura de rede-em-*chip* de acordo com a computação demandada. Isto significa que a reconfiguração da NoC pode escalar o poder de processamento à medida que serviços sejam demandados, criando ambientes *intra-chip* lógicos ou físicos, mapeando necessidades de serviços diferentes gerenciados por processos diferentes.

6.1.7 Projeto de um Sistema Real para Experimentação

Um dos grandes problemas durante o processo de avaliação de uma proposta de arquitetura é a dificuldade de se encontrar um sistema real onde seja possível aplicar o modelo de medição ou experimentação, além de comparar com os modelos analítico e de simulação. Em uma arquitetura *many-core*, existe uma quantidade muito grande de variáveis que envolvem os núcleos de processamento, além de memórias e outros periféricos que compõem o sistema em *chip*. Em complemento com o ambiente de simulação descrito na Seção 6.1.1, um ambiente composto por vários FPGAs pode prover as condições necessárias para medições, já que neste caso o sistema em avaliação existe.

Portanto, este trabalho futuro pode trazer benefícios associados ao processo de avaliação de desempenho, sendo possível medir e identificar os gargalos de projeto e funcionamento de um sistema real *many-core*. Os resultados, dentro de um processo evolutivo de projeto, contribuem para verificação de viabilidade prática da arquitetura.

6.1.8 MCNoC Polimórfica

A exploração da reconfiguração também é possível através do uso polimorfismo baseado em ASICs. Neste caso, não seriam utilizados FPGAs no primeiro nível de reconfiguração da MCNoC, mas PASICs baseados nas chaves *crossbar* reconfiguráveis (RCS-NRs) associados com os processadores de rede (NPoCs). Portanto, haveria um primeiro nível composto pela combinação de PASICs e ASIPs. A reconfiguração neste

primeiro nível depende de uma programação mais ampla da RCS-NR que realize uma integração entre RCS-NRs. A Figura 6.2 ilustra como seria a nova combinação:

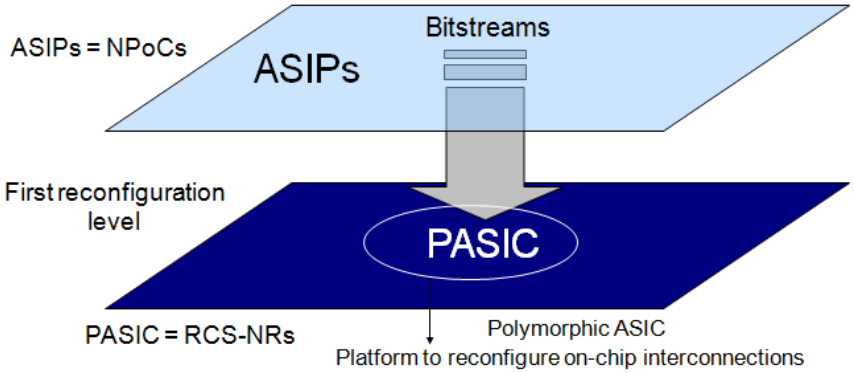


Figura 6.2: MCNoC polimórfica

REFERÊNCIAS

AGARWAL, A., Limits on Interconnection Network Performance, **IEEE Transactions on Parallel and Distributed Systems**, v.2, n.4, p.398-412, October 1991.

AGGARWAL, N., et al., Isolation in commodity multicore processors, **IEEE Computer**, v.40, n.6, p.49-59, June, 2007.

AHMADI, H., DENZEL, W. E., A Survey of Modern High-Performance Switching Technique, **IEEE Journal on Selected Areas in Communications**, v.7, n.7, p.1091-1103, September 1989.

ANDERSON, G. A., JENSEN, E. D., Computer Interconnection Structures: Taxonomy, Characteristics, and Examples, **ACM Computing Surveys**, v.7, n.4, p.197-213, December 1975.

ANDRIAHANTENAINA, A., et al., SPIN: a scalable, packet switched, on-chip micro-network, **Design Automation and Test in Europe Conference and Exhibition (DATE)**, p.70–73, March 2003.

BARTIC, T. A., et al., Topology adaptive network-on-chip design and implementation, **IEE Proc. Comput. Digit. Tech.**, v.152, n.4, p.467-472, July 2005.

BENINI, L., MICHELI, G. D., Networks on chips: a new SoC paradigm, **IEEE Computer**, v.1, p.70-78, January 2002.

BENINI, L., MICHELI, G. D., Network-on-chip architectures and design methods, **IEE Proceedings Computers & Digital Techniques**, v.152, n.2, p.261-272, 2005.

BERTOZZI, D., et al., NoC Synthesis Flow for Customized Domain Specific Multiprocessor Systems-on-Chip, **IEEE Transactions on Parallel and Distributed Systems**, v.16, n.2, p.113-129, February 2005.

BJERREGAARD, T. and MAHADEVAN, S., A Survey of Research and Practices of Network-on-Chip, **ACM Computing Surveys**, v.38, n.1, p.1-51, March 2006.

BOLOTIN, E., et al., QNoC: QoS architecture and design process for network on chip, **The Journal of Systems Architecture, Special Issue on Networks on Chip**, v.50, n.2, p.105–128, 2004.

BOUHRAOUA, A., ELRABAA, M. E., A High-Throughput Network-on-Chip Architecture for Systems-on-Chip Interconnect, **IEEE International Symposium on Systems-on-Chip**, p.1-4, 2006.

BUYYA, R., High Performance Cluster Computing – Architectures and Systems, v.1, [S.I.], **Prentice Hall**, 1999.

CALVERT, K. L., DOAR, M. B., ZEGURA, E. W., Modeling Internet Topology, **IEEE Communications Magazine**, v.35, n.6, p.160-163, June 1997.

CARVALHO, E., CALAZANS, N., MORAES, F., Heuristics for Dynamic Task Mapping in NoC-based Heterogeneous MPSoC, **IEEE International Workshop on Rapid System Prototyping**, p.43-40, 2007.

CHANG, M.-C. F., et al., RF/wireless interconnect for inter- and intra-chip communications, **Proc. of The IEEE**, v.89, n.4, p.456-466, April 2001.

CIDON, I., KOLODNY, A., GINOSAR, R., The Elements of NoC, Tutorial (Slides), **ACM/IEEE International Symposium on Networks-on-Chip (NOCS)**, San Diego, USA, 2009.

COMER, D. E., Network Systems Design Using Network Processors, [S.I.], **Prentice Hall**, 2003.

COMPTON, K., HAUCK, S., Reconfigurable Computing: A Survey of Systems and Software, **ACM Computing Surveys**, v. 34, n.2, p.171-210, June 2002.

CULLER, D. E., PAL, S. J., GUPTA, A., Parallel Computer Architecture: a hardware/software approach, [S.I.], **Morgan Kaufmann**, 1999.

DALLY, W., TOWLES, B., Route packets, not wires: on-chip interconnection networks, **38th Design Automation Conference (DAC)**, p.684–689, June 2001.

DONGARRA, J., et al., The Sourcebook of Parallel Computing, [S.I.], **Morgan Kaufmann**, 2003.

DE ROSE, C., NAVAUX, P. O. A., Arquiteturas Paralelas, [S.I.], **Sagra Luzzatto**, 2003.

DRATH, R., Visual Object Net. Disponível em <**http://www.r-drath.de/VON/von_e.htm**>. Acesso em: 18 fev. 2008.

DUATO, J., YALAMANCHILI, S., NI, L., Interconnection Networks, [S.I.], **Morgan Kaufmann**, 2002.

DUNCAN, R., A Survey of Parallel Computer Architectures, **IEEE Survey & Tutorial Series**, v.23, n.2, p.5-16, February 1990.

EGGERS, H., et al., Fast Reconfigurable Crossbar Switching in FPGAs, **6th International Workshop on Field-Programmable Logic and Applications (FPL)**, LNCS 1142, p.297-306, 1996.

FERNANDES, S., et al., IPNoSys: uma nova arquitetura paralela baseada em redes em chip, **Simpósio em Sistemas Computacionais de Alto Desempenho (WSCAD)**, p.53-60, 2008.

FEWER, C., et al., A High I/O Reconfigurable Crossbar Switch, **Annual IEEE Symposium on Field-Programmable Custom Computing Machines**, p.3-10, 2003.

FORSELL, M., A scalable high-performance computing solution for networks on chips, **IEEE Micro**, v.22, n.5, p.46–55, 2002.

FOSTER, I., KESSELMAN, C., The GRID: Blueprint for a New Computing Intrastructure, 2. Edition, [S.I.], **Morgan Kaumann**, 2004.

FREITAS, H. C., et al., Reconfigurable Crossbar Switch Architecture for Network Processors, **IEEE International Symposium on Circuits and Systems (ISCAS)**, p.4042-4045, Greece, May 2006a.

FREITAS, H. C., WAGNER, F. R., MARTINS, C. A. P. S., NAVAUX, P. O. A., Projeto de um Processador de Rede Intra Chip para Controle de Comunicação entre Múltiplos Cores, **Workshop em Sistemas Computacionais de Alto Desempenho (WSCAD)**, p.3-10, Ouro Preto, 2006b.

FREITAS, H. C., NAVAUX, P. O. A., Chip Multithreading: Conceitos, Arquiteturas e Tendências. (Desenvolvimento de material didático ou instrucional), **TI 1253**, **PPGC/UFRGS**, 2006c.

FREITAS, H. C., SANTOS, T. G. S., MAILLARD, N. B., NAVAUX, P. O. A., Previsão de Comunicação em Network-on-Chip para Arquiteturas Multi-Core, **Workshop de Processamento Paralelo e Distribuído**, Porto Alegre, RS, BRA, 2006d.

FREITAS, H. C., COLOMBO, D. M., KASTENSMIDT, F. L., NAVAUX, P. O. A., Evaluating Network-on-Chip for Homogeneous Embedded Multiprocessors in FPGAs, **IEEE International Symposium on Circuits and Systems (ISCAS)**, p.3776-3779, May 2007.

FREITAS, H. C., NAVAUX, P. O. A., Evaluating On-Chip Interconnection Architectures for Parallel Processing, **IEEE International Symposium on Scientific and Engineering Computing (SEC)**, in conjunction with the IEEE CSE 2008, p.188-193, 2008a.

FREITAS, H. C., SANTOS, T. G. S., NAVAUX, P. O. A., Design of Programmable NoC Router Architecture on FPGA for Multi-Cluster NoCs, **IET Electronics Letters**, v.44, n.16, p.969-971, July 2008b.

FREITAS, H. C., SANTOS, T. G. S., NAVAUX, P. O. A., NoC Architecture Design for Multi-Cluster Chips, **IEEE International Conference on Field Programmable Logic and Applications (FPL)**, p.53-58, 2008c.

FREITAS, H. C., NAVAUX, P. O. A., A High-Throughput Multi-Cluster NoC Architecture, **IEEE International Conference on Computational Science and Engineering (CSE)**, p.56-63, 2008d.

FREITAS, H. C., ALVES, M. A. Z., NAVAUX, P. O. A., NoC e NUCA: Conceitos e Tendências para Arquiteturas de Processadores Many-Core, **Escola Regional de Alto Desempenho (ERAD)**, Cap.3, 2009a.

FREITAS, H. C., NAVAUX, P. O. A., On the Design of Reconfigurable Crossbar Switch for Adaptable On-Chip Topologies in Programmable NoC Routers, **ACM Great Lakes Symposium on VLSI (GLSVLSI)**, p.129-132, 2009b.

FREITAS, H. C., ALVES, M. A. Z., SCHNORR, L. M., NAVAUX, P. O. A., Performance Evaluation of NoC Architectures for Parallel Workloads, **ACM/IEEE International Symposium on Networks-on-Chip (NOCS)**, p.87, 2009c.

FREITAS, H. C., MADRUGA, F. L., ALVES, M. A. Z., NAVAUX, P. O. A., Design of Interleaved Multithreading for Network Processors on Chip, **IEEE International Symposium on Circuits and Systems (ISCAS)**, p.2213-2216, 2009d.

FOX, G., Grids Challenged by a Web 2.0 and Multicore Sandwich, Keynote (Slides), **IEEE International Symposium on Cluster Computing and the Grid (CCGRID)**, 2007.

GHOSH, A., et al., System modeling with SystemC, **International Conference on ASIC**, p.18-20, 2001.

GIRAULT, C., VALK, R., Petri Nets for Systems Engineering: A Guide for Modeling, Verification and Application, [S.I.], **Springer-Verlag**, 2002.

GRAMA, A. Y., GUPTA, A., KUMAR, V., Isoefficiency: Measuring the Scalability of Parallel Algorithms and Architectures, **IEEE Parallel & Distributed Technology**, v.1, n.3, p.12-21, 1993.

GRECU, C., et al., A scalable communication-centric SoC interconnect architecture, **IEEE International Symposium on Quality Electronic Design (ISQED)**, p.343-348, 2004.

HENNESSY, J. L., D. A. PATTERSON, Computer Architecture: A Quantitative Approach, 4. Edition, [S.I.], **Morgan Kaufmann**, 2006.

HILL, M. D., MARTY, M. R., Amdahl's Law in the Multicore Era, **IEEE Computer Society**, p.33-38, July 2008.

HILTON, C. and NELSON, B., PNoC: A Flexible Circuit-Switched NoC for FPGA-based Systems, **IEE Proceedings of Computer & Digital Techniques**, v.153, n.3, p.181-188, May 2006.

HO, R., et al., The Future of Wires, **Proceedings of the IEEE**, v.89, n.4, April 2001.

HO, W. H., PINKSTON, T. M., A Design Methodology for Efficient Application-Specific On-Chip Interconnects, **IEEE Transactions on Parallel and Distributed Systems**, v.17, n.2, p.174-190, February 2006.

HUR, J. Y., STEFANOV, T., WONG, S., VASSILIADIS, S., Systematic Customization of On-Chip Crossbar Interconnects, **International Workshop on Applied Reconfigurable Computing**, p.61-72, 2007.

HWANG, K., XU. Z., Scalable Parallel Computing: Technology, Architecture, Programming, **Boston, WCB**, 1998.

INTEL, IXP1200 Network Processor Family, Hardware Reference Manual, **Intel**, December, 2001.

INTEL, 80 FPU Cores on a Chip for Teraflop Performance, **Tom's Hardware**, Disponível em: <**http://www.tomshardware.com/2006/09/27/idf_fall_2006/page2.html**>. Acesso em: 20 nov. 2006.

INTEL, 80-Core Programmable Processor First to Deliver Teraflops Performance, **Intel Platform Research,** Disponível em: <**http://www.intel.com/research/platform/index.htm**>**.** Acesso em: 25 out. 2007.

JAIN, R., The Art of Computer Systems Performance Analysis, [S.I.], **John Wiley**, 1991.

JIN, H., FRUMKIN, M., YAN, J., The OpenMP Implementation of NAS Parallel Benchmarks and its Performance, **NAS Technical Report NAS-99-011**, NASA Ames Research Center, 1999.

JOSEPH, N., et al., RECONNECT: A NoC for Polymorphic ASICs Using a Low Overhead Single Cycle Router, **IEEE**, p.251-256, 2008.

KALLA, R., SINHAROY, B., TENDLER, J. M., IBM Power5 Chip: A Dual-Core Multithreaded Processor, **IEEE MICRO**, v.24, n.2, p.40-47, 2004.

KARIM, F., NGUYEN, A., DEY, S., An interconnect architecture for network systems on chips, **IEEE Micro,** v.22, n.5, p.36-45, 2002.

KEYES, R. W., Fundamental Limits of Silicon Technology, **Proceedings of the IEEE**, v.89, n.3, p.227-239, March 2001.

KEYES, R. W, Moore's Law Today, **IEEE Circuits and Systems Magazine**, v.8, n.2, p.53-54, 2008.

KIM, D., et al., A Reconfigurable Crossbar Switch with Adaptive Bandwidth Control for Networks-on-Chip, **IEEE International Symposium on Circuits and Systems (ISCAS)**, p. 2369 – 2372, 2005.

KONGETIRA, P., et al., Niagara: a 32-way multithreaded Sparc processor, **IEEE MICRO**, v.25, n.2, p.21-29, March-April 2005.

KRASNOV*,* A., et al*.,* RAMP Blue: A Message-Passing Manycore System In FPGAs**, Proceedings of International Conference on Field Programmable Logic and Applications (FPL)***,* p.54-61, 2007.

KREUTZ, M., MARCON, C., CARRO, L., CALAZANS, N., SUSIN, A. A., Energy and Latency Evaluation of NOC Topologies, **International Symposium on Circuits and Systems (ISCAS)**, p.5866-5869, 2005.

KUMAR, R., et al., Single-ISA heterogeneous multi-core architectures for multithreaded workload performance, **31st International Symposium on Computer Architecture (ISCA)**, p.64-75, June 2004.

KUMAR, R., et al., Heterogeneous chip multiprocessors, **IEEE Computer**, v.38, n.11, p.32-38, November 2005a.

KUMAR, R., Zyuban, V., Tullsen, D.M., Interconnections in Multi-core Architectures: Understanding Mechanisms, Overheads and Scaling, **32nd International Symposium on Computer Architecture (ISCA)**, p.408-419, June 2005b.

LAN, Y.-C., LO, S.-H., LIN, Y.-C., HU, Y.-H., CHEN, S.-J., BiNoC: A Bidirectional NoC Architecture with Dynamic Self-Reconfigurable Channel, **ACM/IEEE International Symposium on Networks-on-Chip (NOCS)**, p.266-275, 2009.

LEMEIRE, J., DIRKX, E., COLITTI, W., Modeling the Performance of Communication Schemes on Network Topologies, **Parallel Processing Letters**, v.18, n.2, p.205-220, 2008.

LENG , X., et al., Implementation and Simulation of a Cluster-based Hierarchical NoC Architecture for Multi-Processor SoC, **IEEE International Symposium on Communiations and Information Technology**, p.1163-1166, 2005.

LE BOUDEC, J.-Y., Performance Evaluation of Computer and Communication Systems, **Electronics Book**, 2007. Disponível em <**http://perfeval.epfl.ch**>. Acesso em: 10 out. 2008.

LIANG, J., SWAMINATHAN, S., TESSIER, R., aSOC: A scalable, single-chip communications architecture, in: **IEEE International Conference on Parallel Architectures and Compilation Techniques**, p.37–46, October 2000.

LILJA, D. J., Measuring Computer Performance, [S.I.], **Cambridge University Press**, 2004.

MACIEL, P. R., et al., Introdução às Redes de Petri e Aplicações, **X Escola de Computação**, 1996.

MANEVICH, R., WALTER, I., CIDON, I., KOLODNY, A., Best of Both Worlds: A Bus-Enhanced NoC (BENoC), **ACM/IEEE International Symposium on Networks-on-Chip (NOCS)**, p.173-182, 2009

MARESCAUX, T., Networks on chip as hardware components of an OS for reconfigurable systems, **Field-Programmable Logic and Applications (FPL)**, p.595-605, September 2003.

MARTINS, C. A. P. S., et al., Computação Reconfigurável: Conceitos, Tendências e Aplicações, Cap.8, **Jornada de Atualização em Informática**, CSBC, 2003.

MENASCÉ, D. A., ALMEIDA, A. F., Planejamento de Capacidade para Serviços na Web, Métricas, Modelos e Métodos, [S.I.], **Campus**, 2003.

MENTOR GRAPHICS, ModelSim, **Mentor Graphics,** Disponível em: <**http://www.mentor.com/products/fv/digital_verification/modelsim_se/**>. Acesso em: 01 nov. 2008.

MERCALDI-KIM, M., Davis, J. D., Oskin, M., Austin, T., Polymorphic On-Chip Networks, **IEEE International Symposium on Computer Architecture (ISCA)**, p.101-112, 2008.

MILLBERG, M., The Nostrum backbone - a communication protocol stack for networks on chip, **Proceedings of the VLSI Design Conference**, p.693-696, January 2004.

MORAES, F., et al., HERMES: an infrastructure for low area overhead packet-switching networks on chip, **Integration the VLSI Journal**, p.69-93, 2004.

MURATA, T., Petri Nets: Properties, Analysis and Applications, **Proceedings of the IEEE**, v.77, p.541-580, 1989.

NGUYEN, H. T. L., Network-on-Chip Dataflow Architecture, **U. S. p. a. t. office, Ed. United States**, Hanoi Tran Le Nguyen (VN), p.9, 2007.

NIEMANN, J., PORRMANN, M., RÜCKERT, U., A Scalable Parallel SoC Architecture for Network Processors, **IEEE Annual Symposium on VLSI**, p.311-313, 2005.

NPB, NAS Parallel Benchmark, Disponível em: <**http://www.nas.nasa.gov/Resources/Software/npb.html>.** Acesso em: 10 set. 2007.

OGRAS, U. Y., MARCULESCU, R., Prediction-based flow control for network-on-chip traffic, **ACM/IEEE Design Automation Conference**, p.839-844, 2006.

OGRAS, U. Y. et al., Challenges and Promising Results in NoC Prototyping Using FPGAs, **IEEE Micro**, v.27, n.5, p.86-95, Sept.-Oct. 2007a.

OGRAS, U. Y. et al., On-Chip Communication Architecture Exploration: A Quantitative Evaluation of Point-to-Point, Bus, and Network-on-Chip, **ACM Transactions on Design Automation of Electronics Systems**, v.12, n.3, Article 23, p.1-20, August 2007b.

OLUKOTUN, K. et al., The Case for a Single-Chip Multiprocessor, **7th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)**, p.2-11, 1996.

OLUKOTUN, K., HAMMOND, L., The Future of Microprocessors, **ACM Queue**, v.3, n.7, p.26-29, September 2005.

ORDONEZ, E. D. M., et al., Projeto, Desempenho e Aplicações de Sistemas Digitais em Circuitos Programáveis (FPGAs), [S.I.], **Bless Gráfica e Editora Ltda**, 2003.

PATTERSON, D. A., HENNESSY, J. L., Organização de Computadores a Interface Hardware/Software, 3. Edição, [S.I.], **Campus**, 2005.

PENHA, D. O., FREITAS, H. C., MARTINS, C. A. P. S., Modelagem de Sistemas Computacionais usando Redes de Petri: aplicação em projeto, análise e avaliação, **IV Escola Regional de Informática RJ/ES**, 2004.

PIONTECK, T., et al., Adaptive Communication Architectures for Runtime Reconfigurable Systems-on-Chip, **Parallel Processing Letters**, v.18, n.2, p.275-289, 2008.

RHOADS, S., Plasma Processor, **Open Cores,** Disponível em: <**http://www.opencores.org/**>. Acesso em: 21 nov. 2007.

QIU, X., et al., High Performance Multi-Paradigm Messaging Runtime Integrating Grids and Multicore Systems, **IEEE International Conference on e-Science and Grid Computing**, p.407-414, December 2007.

RIGO, S., et al., ArchC: A SystemC-Based Architecture Description Language, **International Symposium on Computer Architecture and High Performance Processing (SBAC-PAD)**, p.66-73, October 2004.

RIJPKEMA, E., GOOSSENS, K., RADULESCU, A., Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip, **Design, Automation and Test in Europe (DATE)**, p.350–355, March 2003.

SCHNORR, L. M., HUARD, G., NAVAUX, P. O. A., 3D Approach to the Visualization of Parallel Applications and Grid Monitoring Information, **The 9th IEEE/ACM International Conference on Grid Computing (GRID)**, p.233-241, 2008.

SGROI, M., Addressing the system-on-chip interconnect woes through communication-based design, **38th Design Automation Conference (DAC)**, p.667–672, June 2001.

SHAH, N., Understanding Network Processors, **Master's thesis**, University of California, Berkeley, USA, 2001.

SIGÜENZA-TORTOSA, D., NURMI, J., Proteo: a new approach to network-on-chip, **IASTED International Conference on Communication Systems and Networks (CSN)**, September 2002.

SO, H., Introduction to Reconfigurable Computing, **CS61c sp06 Lecture (5/5/06), Machine Structures**, University of California, Berkeley, 2006.

SPRACKLEN, L., ABRAHAM, S.G., Chip Multithreading: Opportunities and Challenges, **International Symposium on High-Performance Computer Architecture (HPCA)**, p.248-252, February 2005.

STENSGAARD, M. B., SPARS, J., ReNoC: A Network-on-Chip Architecture with Reconfigurable Topology, **ACM/IEEE International Symposium on Networks-on-Chip**, p.55-64, 2008.

STOJMENOVIC, I., Honeycomb Networks: Topological Properties and Communication Algorithms, **IEEE Transactions on Parallel and Distributed Systems**, v.8, n.10, p.1036-1042, October 1997.

TANENBAUM, A. S., Redes de Computadores, [S.I.], **Campus**, 2003.

TODMAN, T. J., et al., Reconfigurable Computing: architectures and design methods, **IEE Proc.-Comput. Digit. Tech.**, v.152, n.2, p.193-207, March 2005.

UNGERER, T, et al., Multithreaded Processors, **The Computer Journal, British Computer Society**, v.45, n.3, p.320-348, 2002.

UNGERER, T., et al., A Survey of Processors with Explicit Multithreading, **ACM Computing Surveys**, v.35, n.1, p.29-63, March 2003.

VASSILIADIS, S., SOURDIS, I., FLUX interconnection networks on demand, **Elsevier Journal of Systems Architecture**, v.53, n.10, pp.777-793, 2007.

VERBAUWHEDE, I., SCHAUMONT, P., The Happy Marriage of Architecture and Application in Next-Generation Reconfigurable Systems, **ACM Computing Frontiers**, p.363-376, April 2004.

WANG, H., PEH, L.-S., MALIK, S., Power-driven design of router microarchitectures in on-chip networks. **International Symposium on Microarchitecture**, p.105–115, 2003.

WANG, Y., ZHAO, D., Design and Implementation of Routing Scheme for Wireless Network-on-Chip, **IEEE International Symposium on Circuits and Systems (ISCAS)**, p.1357-1360, May 2007.

WANG, D., et al., A Link Removal Methodology for Networks-on-Chip on Reconfigurable Systems, **IEEE International Conference on Field Programmable Logic and Applications (FPL)**, p.269-274, 2008.

WIKLUND, D., LIU, D., SoCBUS: switched network on chip for hard real time systems, **International Parallel and Distributed Processing Symposium (IPDPS)**, April 2003.

WOLF, W., The Future of Multiprocessor System-on-Chip, **Proceedings of the DAC**, June, 2004.

XILINX, ISE Design Suite, **Xilinx,** Disponível em: <**http://www.xilinx.com/ise/logic_design_prod/**>. Acesso em: 10 nov. 2008.

ZEFERINO, C. A., SUSIN, A. A., SoCIN: A Parametric and Scalable Network-on-Chip, **IEEE Symposium on Integrated Circuits and Systems Design (SBCCI)**, p.169-174, 2003.

ANEXO A RELATÓRIOS SUMARIZADOS DE SÍNTESES

RANoC 8

* Final Report *

Macro Statistics :
Registers : 299
1-bit register : 156
2-bit register : 1
3-bit register : 1
30-bit register : 1
32-bit register : 119
4-bit register : 19
6-bit register : 2
Adders/Subtractors : 1
6-bit subtractor : 1
Comparators : 22
2-bit comparator greater : 8
2-bit comparator less : 8
32-bit comparator equal : 1
33-bit comparator greatequal: 1
33-bit comparator less : 2
6-bit comparator not equal : 2
Xors : 64
1-bit xor3 : 64

MCNoC 14

* Final Report *

Macro Statistics :
Registers : 598
1-bit register : 312
2-bit register : 2
3-bit register : 2
30-bit register : 2
32-bit register : 238
4-bit register : 38
6-bit register : 4
Adders/Subtractors : 2
6-bit subtractor : 2
Comparators : 44
2-bit comparator greater : 16
2-bit comparator less : 16
32-bit comparator equal : 2
33-bit comparator greatequal: 2
33-bit comparator less : 4
6-bit comparator not equal : 4
Xors : 128
1-bit xor3 : 128

MCNoC 28/32

* Final Report *

Macro Statistics :
Registers : 1495
1-bit register : 780
2-bit register : 5
3-bit register : 5
30-bit register : 5
32-bit register : 595
4-bit register : 95
6-bit register : 10
Adders/Subtractors : 5
6-bit subtractor : 5
Comparators : 110
2-bit comparator greater : 40
2-bit comparator less : 40
32-bit comparator equal : 5
33-bit comparator greatequal: 5
33-bit comparator less : 10
6-bit comparator not equal : 10
Xors : 320
1-bit xor3 : 320

ANEXO B PUBLICAÇÕES

Neste anexo é apresentada a lista de publicações durante o doutorado.

Artigos relacionados com a tese

1. Freitas, H. C., Madruga, F. L., Alves, M. A. Z., Navaux, P. O. A., "*Design of Interleaved Multithreading for Network Processors on Chip*", IEEE International Symposium on Circuits and Systems, ISCAS 2009, Taipei, Taiwan, pp.2213-2216, 2009 (Qualis A Internacional)
2. Freitas, H. C., Alves, M. A. Z., Schnorr, L. M., Navaux, P. O. A., "*Performance Evaluation of NoC Architectures for Parallel Workloads*", ACM/IEEE International Symposium on Networks-on-Chip, NOCS 2009, San Diego, USA, pp.87, 2009 (Resumo na principal conferência da área)
3. Freitas, H. C., Navaux, P. O. A., "*On the Design of Reconfigurable Crossbar Switch for Adaptable On-Chip Topologies in Programmable NoC Routers*", ACM Great Lakes Symposium on VLSI, GLSVLSI 2009, Boston, USA, pp.129-132, 2009 (Qualis A Internacional)
4. Freitas, H. C., Alves, M. A. Z., Navaux, P. O. A., "*NoC e NUCA: Conceitos e Tendências para Arquiteturas de Processadores Many-Core*", Escola Regional de Alto Desempenho, ERAD 2009, Caxias do Sul, Brazil, Capítulo 3, 2009 (minicurso)
5. Freitas, H. C., Santos, T. G. S., Navaux, P. O. A., "*Design of programmable NoC router architecture on FPGA for multi-cluster NoCs*", IET Electronics Letters, v. 44, Issue 16, pp. 969-971, 2008 (Periódico Qualis A Internacional)
6. Freitas, H. C., Santos, T. G. S., Navaux, P. O. A., "*NoC Architecture Design for Multi-Cluster Chips"*, IEEE International Conference on Field Programmable Logic and Applications, FPL 2008, Heidelberg, Germany, pp. 53-58, 2008 (Qualis A Internacional)
7. Freitas, H. C., Navaux, P. O. A., "*A High-Throughput Multi-Cluster NoC Architecture*", IEEE International Conference on Computational Science and Engineering, CSE 2008, São Paulo, Brazil, pp. 56-63, 2008 (Qualis B ou A Internacional pelo documento de área)
8. Freitas, H. C., Navaux, P. O. A., "*Evaluating On-Chip Interconnection Architectures for Parallel Processing*", IEEE International Symposium on Scientific and Engineering Computing, SEC 2008, in conjunction with the

IEEE CSE 2008, São Paulo, Brazil, pp. 188-193, 2008 (Qualis C Internacional pelo documento de área)

9. Freitas, H. C., Colombo, D. M., Kastensmidt, F. L., Navaux, P. O. A., "*Evaluating Network-on-Chip for Homogeneous Embedded Multiprocessors in FPGAs*", In: IEEE International Symposium on Circuits and Systems, ISCAS 2007, New Orleans, USA, pp.3776-3779, 2007 (Qualis A Internacional)
10. Freitas, H. C., Navaux, P. O. A., "*The First NPoC Design*", Workshop de Processamento Paralelo e Distribuído, Porto Alegre, RS, Brazil, 2008 (evento local sem classificação)
11. Freitas, H. C., Alves, M. A. Z., Navaux, P. O. A., "*Utilização de uma Metodologia de Projeto para Arquiteturas de Processadores Multi-Core*", 8a Escola Regional de Alto Desempenho, ERAD 2008, Santa Cruz do Sul, RS, Brazil, pp.171-172, 2008 (evento regional sem classificação)
12. Freitas, H. C., Wagner, F. R., Navaux, P. O. A., Martins, C. A. P. S., "*Projeto de um Processador de Rede Intra-Chip para Controle de Comunicação entre Múltiplos Cores*", VII Workshop em Sistemas Computacionais de Alto Desempenho, WSCAD 2006, held in conjunction with the 18th International Symposium on Computer Architecture and High Performance Computing, Ouro Preto, MG, Brazil, pp.3-10, 2006, prêmio de melhor artigo (Qualis C Nacional)
13. Freitas, H. C., Navaux, P. O. A., "*Estudo e Projeto de Arquiteturas de Comunicação Intra-Chip para Processadores Multi-Core*", Escola Regional de Alto Desempenho, Porto Alegre, pp.55-56, 2007 (evento regional sem classificação)
14. Freitas, H. C., Santos, T. G. S., Maillard, N. B., Navaux, P. O. A., Previsão de Comunicação em Network-on-Chip para Arquiteturas Multi-Core, Workshop de Processamento Paralelo e Distribuído, Porto Alegre, RS, Brazil, 2006 (evento local sem classificação)

Participação em artigos do GPPD

1. Alves, M. A. Z., Freitas, H. C., Navaux, P. O. A., "*Investigation of Shared L2 Cache on Many-Core Processors*", Workshop on Many Cores, WMC 2009, in conjunction with the LNCS ARCS 2009, Delft, Netherlands, pp.21-30, 2009 (Qualis C Internacional)
2. Freitas, H. C., Alves, M. A. Z., Maillard, N. B., Navaux, P. O. A., "*Ensino de Arquiteturas de Processadores Multi-Core Através de um Sistema de Simulação Completo e da Experiência de um Projeto de Pesquisa*", Workshop sobre Ensino de Arquiteturas de Computadores, WEAC 2008, em conjunto com o SBAC-PAD, Campo Grande, pp.1-8, 2008 (Qualis C Nacional)
3. Ferreira, M. K., Freitas, H. C., Navaux, P. O. A., "*From Intel VT-x to MIPS: An ArchC-based Model to Understanding the Hardware Virtualization Support*", Workshop on Computer Architecture Education, WCAE 2008, in

conjunction with the IEEE ISCA 2008, Beijing, China, pp.9-15, 2008 (Qualis C Internacional)

4. Ferreira, M. K., Freitas, H. C., Navaux, P. O. A., "*An Overview of Memory Virtualization Techniques Based on Intel VT*", Workshop de Processamento Paralelo e Distribuído, Porto Alegre, RS, Brazil, 2008 (evento local sem classificação)
5. Alves, M. A. Z., Freitas, H. C., Navaux, P. O. A., "*Proposta de Avaliação de Memórias Cache em Processadores Multi-Core*", 8a Escola Regional de Alto Desempenho, ERAD 2008, Santa Cruz do Sul, RS, Brazil, pp.143-144, 2008 (evento regional sem classificação)
6. Ferreira, M. K., Freitas, H. C., Navaux, P. O. A., "*Modificações no MIPS Inspiradas na Intel VT-x para Suporte à Virtualização Utilizando ArchC*", 8a Escola Regional de Alto Desempenho, ERAD 2008, Santa Cruz do Sul, RS, Brazil, pp.245-248, 2008 (evento regional sem classificação)
7. Cruz, V. S., Freitas, H. C., Navaux, P. O. A., "*Avaliação do Pipeline no Processador MIPS_Robot*", 8a Escola Regional de Alto Desempenho, ERAD 2008, Santa Cruz do Sul, RS, Brazil, pp.217-220, 2008 (evento regional sem classificação)
8. Alves, M. A. Z., Freitas, H. C., Wagner, F. R., Navaux, P. O. A., "*Influência do Compartilhamento de Cache L2 em um Chip Multiprocessado sob Cargas de Trabalho com Conjuntos de Dados Contíguos e Não Contíguos*", VIII Workshop em Sistemas Computacionais de Alto Desempenho, WSCAD 2007, held in conjunction with the 19th International Symposium on Computer Architecture and High Performance Computing, Gramado, RS, Brazil, pp.27-34 , 2007 (Qualis C Nacional)
9. Cruz, V. S., Freitas, H. C., Navaux, P. O. A., "*Proposta de Expansão do Conjunto de Instruções do MIPS para Robótica*", I Concurso de Trabalhos de Iniciação Científica do Workshop em Sistemas Computacionais de Alto Desempenho (WSCAD - CTIC), Gramado, RS, 2007 (evento nacional sem classificação)
10. Cruz, V. S., Freitas, H. C., Navaux, P. O. A., "*Subconjunto de Instruções do MIPS para Suporte à Robótica*", V Workshop de Processamento Paralelo e Distribuído (WSPPD), Porto Alegre, RS, pp.73-74, 2007 (evento local sem classificação)
11. Ferreira, M. K., Freitas, H. C., Navaux, P. O. A., "*Estudo das Técnicas de Suporte a Virtualização para Projeto de Instruções no Contexto Multi-Core*", V Workshop de Processamento Paralelo e Distribuído (WSPPD), Porto Alegre, RS, pp.79-80, 2007 (evento local sem classificação)