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

MATEUS BECK RUTZIG

Gerenciamento Automático de Recursos Reconfiguráveis Visando a Redução de Área e do Consumo de Potência em Dispositivos Embarcados

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação

Prof. Dr. Luigi Carro
Orientador

Porto Alegre, março de 2008.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Rutzig, Mateus Beck

Gerenciamento Automático de Recursos Reconfiguráveis Visando a Redução de Área e do Consumo de Potência em Dispositivos Embarcados / Mateus Beck Rutzig – Porto Alegre: Programa de Pós-Graduação em Computação, 2008.

112 p.:il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação. Porto Alegre, BR – RS, 2008. Orientador: Luigi Carro.

1.Arquiteturas Reconfiguráveis. 2.Sistemas Embarcados 3.Exploração de Recursos I. Carro, Luigi. II. Título.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

ASIP	Application Specific Instruction Set Processor
DSP	Digital Signal Processor
MP3	Moving Picture Experts Group
JPEG	Joint Pictures Expert Group
UFR	Unidade Funcional Reconfigurável
PPG	Processador de Propósito Geral
RISC	Reduced Instruction Set Computer
CISC	Complex Instruction Set Computer
VHDL	VHSIC hardware description language
TB	Tradutor Binário
LDH	Linguagem de Descrição de Hardware
PISA	Portable Instruction Set Architecture
ISA	Instruction Set Architecture
RTL	Register Transfer Language

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

A complexidade dos sistemas embarcados está crescendo devido à agregação de funcionalidades em um único dispositivo eletrônico e a heterogeneidade de comportamento das aplicações que compõe estas funcionalidades agrava este cenário. Atualmente, os projetistas de processadores estão buscando outro paradigma de computação para ser empregado neste tipo de dispositivo. A aceleração da execução dos processadores Superescalares está estagnada, a extração do paralelismo no modelo Von-Neumann está chegando ao limite teórico. Arquiteturas Dataflow são uma possível solução para este problema, entretanto a área disponível em silício da tecnologia atual não comporta a implementação deste tipo de arquitetura. Arquiteturas reconfiguráveis aparecem como uma solução viável para a exploração de um alto nível de paralelismo, sendo factível a implementação deste tipo de arquitetura nas atuais tecnologias CMOS. Entretanto, a inserção do hardware reconfigurável ocasiona uma elevação na área ocupada e, conseqüentemente, na potência consumida. É neste cenário que este trabalho se insere. Uma arquitetura reconfigurável foi escolhida como estudo de caso, sendo acoplada a um processador MIPS R3000. Além disto, foi desenvolvida uma ferramenta que, automaticamente, constrói um hardware otimizado através da exploração de recursos necessários para obter o máximo grau de paralelismo da execução de um conjunto de aplicações. O acoplamento desta ferramenta com a técnica de tradução binária utilizada nesta arquitetura reconfigurável provê uma exploração estática/dinâmica. Estática pelo ponto de vista de construção de uma nova unidade reconfigurável otimizada em área antes da fabricação do chip. Dinâmica devido a adaptabilidade da execução do tradutor binário, após a fabricação da unidade otimizada gerada pela ferramenta, a unidade otimizada alcança as mesmas acelerações demonstradas na unidade não otimizada com uma menor área ocupada e potência consumida. Além disto, neste trabalho é demonstrado o impacto na potência consumida pelo sistema fornecido por uma técnica de desligamento de blocos da unidade funcional reconfigurável. Assim, as explorações da área e do consumo de potência demonstraram ser factível a inserção da arquitetura reconfigurável proposta em um dispositivo embarcado.

Palavras-Chave: sistemas embarcados, arquiteturas reconfiguráveis, exploração de área, redução de potência.

Automatic Reconfigurable Resources Management Aim to Reduce Area and Power Consumption on Embedded Systems

ABSTRACT

Nowadays, the large amount of complex and heterogeneous functionalities that are found on a single embedded device has driven designers to create novel solutions to increase the performance of embedded processors and, at the same time, maintain power dissipation as low as possible. While the instruction level parallelism exploitation is reaching the theoretical limit, Dataflow architectures are seen as a reasonable proposal to solve this problem. However, even for near future CMOS technologies, the price to pay for using such architectures is still too high. Reconfigurable architectures could be a possible solution to explore higher-levels of parallelism, and their deployment on current CMOS technologies is feasible. However, the fusion of a reconfigurable hardware with a general-purpose processor still implies in a high area overhead, besides the elevated power consumption. The proposal of this work is to couple static and dynamic techniques to achieve a low-power, high performance reconfigurable architecture that can show speed ups for several heterogeneous applications with the minimum possible area overhead. At design time, the static exploitation produces a new reconfigurable unit optimized in area. Thanks to the proposed dynamic reconfiguration mechanism, the optimized reconfigurable unit provides acceleration and low power dissipation, adapting to the different degrees of parallelism available in the application, and accelerating applications not foreseen at design time.

Keywords: embedded systems, reconfigurable architectures, area exploitation, power reduction.

1 INTRODUÇÃO

Atualmente, o crescimento na diversidade dos dispositivos embarcados esta produzindo uma quebra no paradigma da computação e o conceito de se possuir um único computador de mesa esta sendo alterado. Hoje, em uma residência estão distribuídos vários pequenos centros de processamento que realizam quase as mesmas funcionalidades de um computador de mesa. A fácil comunicação entre estes dispositivos agregada ao seu baixo custo atraem cada vez mais consumidores. O comércio destes dispositivos está cada vez maior. Como ilustra a Figura 1.1, até 2009 é estimado um crescimento médio anual de mais de 10% da produção destes dispositivos. Conseqüentemente, a previsão até 2009 é que o lucro das empresas alcance o dobro do obtido em 2004. (VASSILIADIS, 2006)

Inicialmente, pode-se observar que estes eram dispositivos embarcados simples, ou seja, proviam poucas funcionalidades. Exemplos como reprodutores de mp3, telefones celulares, vídeo games portáteis, câmeras fotográficas digitais eram disponibilizados no mercado separadamente. Entretanto, nos dispositivos atuais, todas as funcionalidades citadas acima estão agregadas em um único produto, fato que torna o projeto destes mais complexos de ser realizado.

A complexidade do projeto decorre do fato destes dispositivos proverem restrições, que devem ser obedecidas. Na Figura 1.2 é ilustrado o desempenho de alguns processadores da família Intel x86 e o prognóstico para as três próximas gerações desta família na execução do conjunto de benchmark SPECInt (AUSTIN, 2004). O círculo esboçado no gráfico demonstra o desempenho almejado pelos sistemas móveis embarcados em uma de suas próximas gerações (Two Gen). Nota-se que o poder de processamento requisitado por estes dispositivos ultrapassa o prognóstico dos processadores de propósito geral.

Além da questão do desempenho, a Figura 1.3 ilustra a restrição de consumo de potência de 75mW nesta mesma geração (AUSTIN, 2004). Este valor é equivalente a menos de 1% da potência total consumida por um processador de propósito geral. Estas duas características agrupadas relatam a complexidade de se desenvolver uma arquitetura para um dispositivo embarcado. Um fator importante a ser observado é a tendência de estes dispositivos diminuírem cada vez mais de tamanho. Para isto, seus componentes devem ser projetados seguindo rígidas restrições de área, o que torna o fluxo de projeto ainda mais complicado.

Agregando todas as colocações acima, um projeto de um sistema embarcado se resume em dispor um produto que disponibilize diversas funcionalidades, e estas por

sua vez necessitam de enorme capacidade de processamento. Ao mesmo tempo, os sistemas embarcados devem ser sejam leves, pequenos e possibilitar alta durabilidade de tempo da bateria.

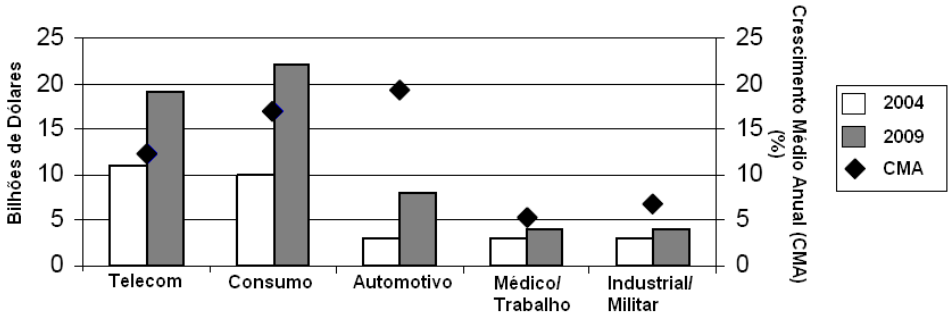

Figura 1.1: Estimativa de mercado para os Sistemas Embarcados (VASSILIADIS, 2006)

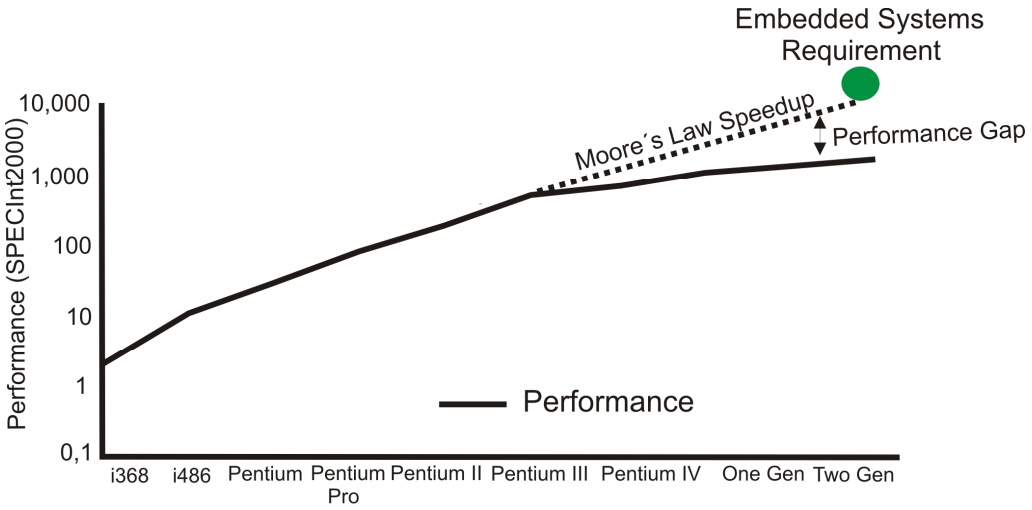

Figura 1.2: Requisito de processamento para os futuros Sistemas Embarcados. (AUSTIN, 2004)

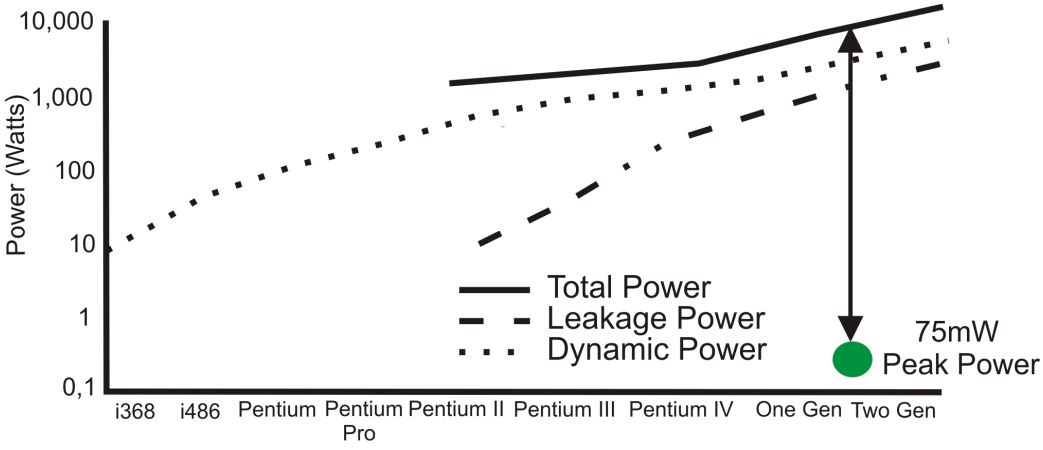

Figura 1.3: Restrição de potência imposta pelos futuros dispositivos embarcados. (AUSTIN, 2004)

1.1 ASIP

O desafio de se obter o melhor desempenho para executar de forma eficiente somente um conjunto de aplicações conduz à projetistas desenvolver uma arquitetura específica para realizar esta tarefa, um *Application Specific Instruction Set Processor* (ASIP) (JAIN, 2001). Este conceito leva a processadores não flexíveis, entretanto, de alto desempenho e baixo consumo de energia comparado aos processadores de propósito geral (PPG). Ao contrário de um ASIP, o PPG é destinado a conjuntos de aplicações que não são especificadas no início do projeto da arquitetura, e normalmente fornecem um desempenho mediano para todo o tipo de aplicação.

No segmento de sistemas embarcados é proveitosa a utilização de processadores ASIP. Devido aos compromissos de projeto relatados anteriormente, um projeto customizado para um conjunto de aplicações fornece a eficiência requisitada para um sistema embarcado (custo, tamanho do código, desempenho e potência) (KUÇUKÇAKAR, 1999). Entretanto, um projeto de uma arquitetura ASIP não é trivial, primeiramente deve-se levantar os requisitos, após especificar a arquitetura, realizar a implementação e finalmente testar. A demora para concretizar todas estas etapas não se torna ideal quando o *time-to-market* é pequeno.

Projetistas de processadores ASIP relatam que a fase que consome mais tempo no processo do projeto é a sincronização e integração do hardware com o software (ERNST, 1998). Para diminuir este gargalo de projeto e alcançar a máxima eficiência, são utilizadas técnicas de particionamento de hardware/software da aplicação. Assim, tanto projetistas de software quanto arquitetos de hardware podem sincronizar suas tarefas para otimizar e debugar o sistema. Desta forma, são descobertos erros em fases iniciais do projeto, diminuindo o custo e respeitando o *time-to-market* desejado.

Entretanto, a acelerada convergência de diversas funcionalidades para um mesmo dispositivo torna inviável a utilização de um processador ASIP para executar cada função de um futuro sistema embarcado. Um processador embarcado deve manipular de forma eficiente aplicações que possuem comportamentos opostos. Telefones celulares atuais já compõem este cenário, aplicações como decodificadores MP3 e agendas telefônicas tem de ser executadas de forma eficiente em uma plataforma. Deste modo, a versatilidade dos processadores de propósito geral pode ser vista como solução para executar as diversas aplicações que compõe um sistema embarcado atual. Para ganhar eficiência, uma expansão do conjunto de instruções no geral faz-se necessária.

Um exemplo específico de um cenário de aplicações heterogêneas é o celular desenvolvido pela Apple. O Iphone provê diversas funcionalidades: reprodutor de áudio, browser para internet, reprodutor de vídeo, agenda telefônica, além de um controle complexo de sua interface com o usuário. Entretanto, como ilustra a Figura 1.4, a Apple optou por um processador de propósito geral ao invés de um ASIP. O processador ARM11 (ARM, 2006) possui várias características que focam na aceleração dos diferentes nichos de aplicações: extensão Jazelle executa nativamente JAVA; instruções DSP foram incorporadas em seu conjunto de instruções; processamento SIMD (do inglês *Single Instruction Multiple Data*) é disponibilizado para execução eficiente de vídeos. Entretanto, a extensão no conjunto de instruções desta plataforma traz consigo a necessidade de recompilação do código fonte das aplicações executas, afetando o *time-to-market* do dispositivo pela ausência de compatibilidade de software.

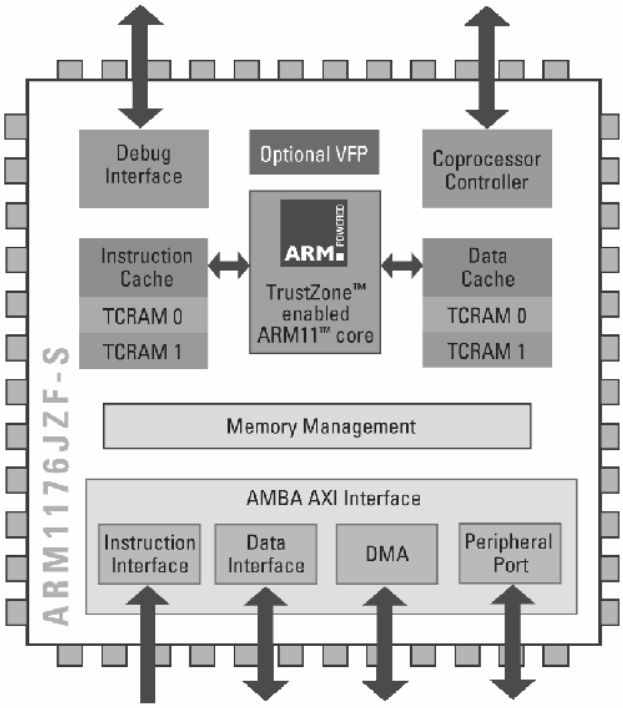

Figura 1.4: Plataforma Iphone

1.2 Processadores Superescalares

Processadores Superescalares são largamente utilizados em computadores de mesa. Entretanto, devido ao seu elevado consumo de potência, sua utilização se torna proibitiva em sistemas embarcados. A exploração do paralelismo em tempo de execução, em nível de instruções, realizada por esta abordagem, fornece um alto índice de aceleração na execução de aplicações. Assim, a recompilação do código fonte não se torna necessária quando há modificação da organização do processador. Além disto, a grande utilização destes processadores motivou a indústria e o meio científico a produzir técnicas de compilação que foquem a aceleração no desempenho desta abordagem. (BACON, 2004)

Níveis altos de exploração de paralelismo, diminuição do tempo de projeto evitando a recompilação do código e compiladores que provêm altos níveis de otimização incentivam a utilização destas arquiteturas em dispositivos embarcados. Entretanto, devido aos limites encontrados em termos de paralelismo em nível de instruções, as modificações no hardware de exploração de paralelismo dos superescalares não estão causando o impacto esperado na aceleração da execução das aplicações. Além disto, com mostrado em (WILCOX, 1999) o mecanismo de escalonamento de instruções do processador superescalar Alpha 21264, que faz parte do hardware de exploração do paralelismo em nível de instrução, abrange 55% da energia total consumida por este processador.

Em (FLYNN, 2005) é relatado que uma das soluções utilizadas para elevar o desempenho destes processadores é o aumento da freqüência de operação do circuito. Esta característica não é viável para dispositivos embarcados, visto que a potência consumida é diretamente proporcional à freqüência do circuito. A complexidade e o consumo de potência do hardware de exploração de ILP dos processadores superescalares, agregado à diversidade do grau de paralelismo das aplicações (XU,

1999; WALL, 1991), formam um cenário não adequado para a utilização destes processadores nos futuros dispositivos embarcados.

Algumas abordagens empregam a técnica VLIW (do inglês *Very Long Instruction Word*) para acelerar aplicações em dispositivos embarcados (SUGA, 2000; GRAY, 1993; NAKAMURA, 1995; KOWALCZYK, 2001). Nesta técnica o compilador é o responsável por detectar as dependências de dados entre as instruções e deixar explícito no código que estas podem ser executadas em paralelo. Assim, o hardware identifica as modificações no código e, dotado de replicações de unidade funcional, consegue executar as instruções simultaneamente. Entretanto, a obrigatória modificação do código binário quebra o conceito de compatibilidade de software, característica indispensável para diminuir o tempo de projeto e respeitar o *time-to-market* imposto pela indústria.

1.3 Exploração de novas alternativas arquiteturais para prover aceleração na execução de aplicações

A heterogeneidade, quanto ao comportamento na execução, das aplicações em um sistema embarcado torna o desenvolvimento da plataforma de execução complexa. Na literatura, as aplicações estão caracterizadas, quanto ao comportamento, como: *Control Flow* e *Data Flow* (OVIEDO, 1980). Considerando-se a equação:

Gcmp = (Número Total de Instruções Executadas)/(Número de Instruções de Controle Executadas) (1)

Analogamente, seguindo (1), aplicações que se encaixam na primeira caracterização são aquelas que, em sua execução, possuem um *Gcmp* (*Grau Comparativo*) menor, comparado a aplicações do segundo tipo. Assim, quanto menor o número de instruções de controle executadas, mais propensa à aplicação é de ser caracterizada como *data Flow*.

Como citado anteriormente, a heterogeneidade de tipos de aplicações convergindo para um sistema embarcado engloba tanto aplicações *control flow* quanto aplicações *data flow*. Assim, é necessário construir uma arquitetura que consiga manipular de forma eficiente os dois comportamentos. Processadores Superescalares obtêm ganhos nos dois comportamentos, já que técnicas como especulação de saltos e predição de desvios ajudam na aceleração de aplicações *control flow*. Entretanto, como explicitado anteriormente, o hardware de especulação agregado ao hardware de exploração do paralelismo elevam significativamente o consumo de energia.

Essencialmente, para acelerar uma dada aplicação, somente duas abordagens podem ser exploradas em hardware: aumento de freqüência de relógio e exploração do paralelismo inerente na aplicação. A primeira abordagem é simples de ser colocada em prática, mas não é favorável para projetos embarcados, sendo que a potência é diretamente proporcional à freqüência do circuito. Já a exploração do paralelismo da aplicação é o alvo principal dos projetistas na obtenção de um melhor desempenho de execução.

Atualmente, o paralelismo é explorado em diversas granularidades: arquiteturas superescalares descobrem paralelismo em nível de instruções; arquiteturas SMT (YAMAMOTO, 1994; HIRATA, 1992) executam *threads* distintas da mesma aplicação em um mesmo processador; em um grão mais grosso, arquiteturas CMP (PALACHARLA, 1996) exploram esta característica em nível de processos ou *threads*.

Entretanto, as várias abordagens de exploração do paralelismo mostradas anteriormente recaem no mesmo problema, o paradigma de programação. O modelo Von Neumann traz consigo uma limitação de exploração do paralelismo, causada pelo seu modelo *control driven*, que tem a sua execução conduzida pelo contador de programa. Novos estilos de execução já foram propostos, e máquinas *Dataflow* exploram o máximo paralelismo da aplicação utilizando um modelo *data driven* (DENNIS, 1975). Basicamente, a execução de certa operação dar-se-á quando os dados requisitados para tal estiverem disponíveis.

Atualmente, vários pesquisadores estão direcionando seus estudos para as arquiteturas reconfiguráveis. O fato que motiva a utilização deste tipo de arquitetura é que estas se localizam entre os modelos Von-Neumann e *Dataflow*. Assim, consegue-se obter uma arquitetura em silício, que explora um alto grau de paralelismo. Além disto, a complexidade de desenhar formas geométricas complexas nas futuras tecnologias CMOS forçará o projetista a utilizar estruturas regulares em seus projetos arquiteturais. A conseqüência da utilização de arquiteturas reconfiguráveis nestas novas tecnologias será a baixa taxa de defeitos de fabricação comparado às arquiteturas não regulares, visto que estas são formadas por replicações de estruturas idênticas (OR-BACH, 2001).

Uma arquitetura reconfigurável é formada por uma Unidade Funcional Reconfigurável (UFR); uma unidade capaz de realizar a reconfiguração da UFR e um processador de propósito geral (PPG). A idéia básica desta abordagem é descobrir partes do código que podem ser executadas mais eficientemente na UFR, chamadas de *Kernels* (HUTCHINGS, 1997). A descoberta de um *Kernel* pode ser realizada em tempo de compilação ou execução, dependendo da técnica utilizada. Ganhos são obtidos pela natureza de execução das aplicações, ou seja, esta técnica explora a repetição da execução na unidade reconfigurável dos *Kernels* previamente descobertos. Além disto, a utilização da lógica combinacional, ao invés de seqüencial, auxilia no aumento no desempenho.

Entretanto, a utilização desta técnica infere na inserção de um grande bloco funcional e de um sistema de roteamento para realizar a reconfiguração. Estas duas unidades trazem um enorme acréscimo de área para o sistema. O consumo de potência estática sofre, proporcionalmente, o mesmo acréscimo refletido na área ocupada. Sendo que este consumo está se tornando mais significativo na potência total consumida, fato relacionado à diminuição do tamanho do transistor nas novas tecnologias CMOS. Além disto, as unidades funcionais reconfiguráveis, em sua maioria, formam um circuito totalmente combinacional. Desta forma, não há como evitar o consumo de potência dinâmica sem a modificação da modelagem arquitetura.

O cenário descrito anteriormente torna a implementação de um sistema reconfigurável, que tem como alvo um dispositivo embarcado, um desafio. Neste âmbito, obrigatoriamente, deve-se explorar o máximo nível de paralelismo da aplicação, utilizando a menor área de silício, dotada de mecanismos que realizem a redução no consumo de potência estática e dinâmica da arquitetura.

1.4 Contribuições deste trabalho

Todas as contribuições desta dissertação estão baseadas em dois trabalhos propostos no grupo de Sistemas Embarcados da Universidade Federal do Rio Grande do Sul. Em (BECK, 2005) foi apresentada uma técnica de tradução binária que, agregada a uma unidade funcional reconfigurável, explora o paralelismo em nível de instruções em um

processador de pilha que executa nativamente Java. Neste trabalho foram relatados ganhos significativos de desempenho e redução no consumo de energia da arquitetura. Em (BECK, 2006a) foi demonstrada uma variação da técnica de tradução binária apresentada anteriormente. Neste caso, a abordagem também foi acoplada a uma unidade funcional reconfigurável, porém utilizando um modelo de um processador RISC que executa o conjunto de instruções PISA.

Esta dissertação deseja demonstrar que as inovações propostas naqueles trabalhos são factíveis de serem utilizadas no âmbito de sistemas embarcados. As rígidas restrições impostas a estes dispositivos forçam o projetista a criar métodos que agreguem tanto otimizações em tempo de compilação quanto em tempo de execução. Dependendo da aplicação, as otimizações realizadas em tempo de compilação não são eficazes e, às vezes, acabam por prejudicar alguma característica do dispositivo final (compatibilidade de software, desempenho, consumo de potência, etc) pelo fato destas otimizações não serem capaz de se adaptar à execução da aplicação.

Assim, a principal inovação deste trabalho é compor a idéia de explorar o paralelismo dinamicamente, proposta por Beck, com otimizações estáticas e dinâmicas que são cruciais para a viabilidade da implementação da arquitetura reconfigurável em um dispositivo embarcado, mas ainda mantendo total compatibilidade de software binário.

Em uma análise de um PPG para compor o sistema demonstra-se que o processador MIPS R3000 adequado para as restrições impostas pelo projeto, já que este é amplamente utilizado no âmbito embarcado. Em cima desta plataforma o primeiro objetivo desta dissertação é demonstrar que os mesmos níveis de aceleração de aplicações heterogêneas mostrados por Beck (2006,2007), podem ser alcançados em um processador embarcado.

Entretanto, como citado anteriormente, a grande área ocupada pelo circuito reconfigurável é o que impede sua a implementação nos dispositivos embarcados. Desta forma, este trabalho fornece uma ferramenta que, em tempo de compilação, porém levando em conta o comportamento da execução de uma aplicação na unidade funcional reconfigurável, explora os recursos mínimos necessários para acelerá-la e modela uma nova arquitetura otimizada em área. Entretanto, nesta exploração as principais características da proposta de Beck são conservadas: compatibilidade de software e desempenho. A adaptabilidade fornecida pelo hardware de tradução binária desta abordagem é capaz de, após a otimização de área, explorar em um mesmo nível o paralelismo disponível na aplicação. Assim, com a ferramenta aqui proposta é possível reduzir, em tempo de projeto, a área da UFR, mantendo o mesmo grau de aceleração no dispositivo fabricado.

Agregada a exploração de área em tempo de projeto, esta dissertação propõe uma exploração, em tempo de execução, da potência consumida pela UFR. Beneficiada pelo mecanismo de tradução binária, a técnica utilizada realiza o desligamento, em tempo de execução, de partes da UFR que estão ociosas, reduzindo assim tanto a potência estática quanto a potência dinâmica do circuito. Para prover a redução de potência é necessária a inserção de um transistor que, controlado pelos bits de reconfiguração já existentes da unidade reconfigurável, interrompe a alimentação para uma determinada área do circuito, sendo este o único hardware inserido para a implementação desta técnica.

Assim, o trabalho em conjunto das duas otimizações explora o mecanismo de tradução binária para fornecer o máximo desempenho da proposta de Beck, reduzindo a

área ocupada e a potência consumida pelo sistema reconfigurável. As técnicas propostas conservam a compatibilidade de software e, o mais importante, não inserem nenhum hardware adicional de controle.

Em resumo, as contribuições relevantes deste trabalho são:

- Implementar/Validar o trabalho proposto em (BECK, 2006a) em um modelo de um processador MIPS R3000;
- Estimar custos de área, potência, energia e desempenho desta implementação;
- Desenvolver uma ferramenta que, de forma automática, explore os recursos de hardware necessários para acelerar uma dada aplicação, refletindo em uma melhor exploração do paralelismo e na redução de área da unidade funcional reconfigurável;
- Estimar o impacto que esta ferramenta produziu em área, potência, energia e desempenho;
- Verificar técnicas de redução do consumo de potência estática e dinâmica nas UFR;
- Avaliar o impacto que a técnica escolhida causou na potência e energia do sistema reconfigurável;

1.5 Organização deste trabalho

Discutida a motivação e o foco deste trabalho, nas próximas seções serão explicitadas de maneira mais abrangente as implementações e resultados do mesmo. No Capítulo 2 serão apresentados e criticados alguns trabalhos relacionados às arquiteturas reconfiguráveis, além de propostas de exploração de recursos de hardware. No Capítulo 3 será apresentada a arquitetura reconfigurável utilizada como estudo de caso deste trabalho, além da implementação desta no processador MIPS R3000. O impacto em área, energia, potência e desempenho causado pela inserção da arquitetura reconfigurável será demonstrado neste capítulo.

A ferramenta ARISE desenvolvida para realizar a exploração de recursos de hardware da arquitetura reconfigurável será apresentada no Capítulo 4. Ainda, neste mesmo capítulo será explicitada a técnica de *Sleep Transistor* e os resultados estimados da incorporação desta na arquitetura reconfigurável. Finalmente, no último capítulo serão apresentados as conclusões e algumas motivações para trabalhos futuros.

2 ESTADO DA ARTE

Nesta seção serão apresentados alguns trabalhos relacionados que motivaram a realização desta dissertação. Primeiramente, serão analisadas algumas arquiteturas reconfiguráveis, demonstrando as vantagens e desvantagens de suas abordagens. Após, algumas técnicas relacionadas à exploração de recursos de hardware em arquiteturas reconfiguráveis serão explicitadas.

2.1 Arquiteturas Reconfiguráveis

Vários trabalhos já foram propostos explorando arquiteturas reconfiguráveis objetivando a aceleração de execução de aplicações. Estes demonstraram ganhos significativos desta característica. (GUPTA, 1993; GAJSKI, 1998; HENKEL, 1997; VENKATARAMANI, 2001). Além disto, pela redução do tempo de execução das aplicações, estes trabalhos mostram-se energeticamente eficiente em relação ao sistema original (HENKEL, 1999; WAN, 1998; STITT, 2002).

Na literatura, ainda não existe um senso comum de classificação de arquiteturas reconfiguráveis. Assim, para este trabalho caracterizaremos as arquiteturas baseadas em (COMPTON, 2002). Basicamente, neste trabalho as arquiteturas reconfiguráveis são distinguidas por três aspectos: acoplamento, granularidade e mecanismo de reconfiguração.

2.1.1 Acoplamento

Em um projeto de uma arquitetura reconfigurável a escolha do tipo de acoplamento entre a UFR e o processador de propósito geral reflete diretamente na eficiência do sistema. Uma UFR que é implementada como uma unidade funcional dentro do processador é chamada de fortemente acoplada. A comunicação entre a PPG e a UFR ocorre somente dentro do núcleo, resultando em um alto desempenho. Outra forma de modelar a UFR é como um co-processador do PPG. Normalmente a escolha deste tipo de acoplamento esta ligada a área disponível de silício dentro do núcleo. Quando não existe área suficiente para armazenar a UFR dentro do núcleo, este tipo de acoplamento é utilizado. Nesta última abordagem a UFR é acoplada ao PPG por um barramento externo ao núcleo, assim o custo de comunicação entre o processador e a UFR é mais elevado do que a abordagem anterior.

Existem outras técnicas de acoplamento, chamadas de fracamente acopladas, fornecendo um alto custo de comunicação. A UFR anexada é um exemplo deste tipo de abordagem, esta fica localizada no barramento que conecta a memória cache e a

interface de entrada/saída. O custo de comunicação é alto, entretanto, é menor do que a abordagem de acoplamento independente. Esta última se comunica com o UFR através do barramento de entrada/saída, dentre todas as abordagens esta é a mais fracamente acoplada.

2.1.2 Granularidade

Cada UFR pode ser implementada através de diferentes tipos e tamanhos de blocos funcionais. Por exemplo, pode-se formar uma UFR somente com somadores independentes de um bit, ou simplesmente, utilizar um somador de 32 bits como unidade básica. Este tipo de escolha de projeto é denominado granularidade da UFR.

A granularidade escolhida para os blocos funcionais tem influência no número de bits necessários para reconfiguração. Utilizando o exemplo anterior, uma UFR formada por somadores de um bit como unidade básica refletem em um elevado número de bits para realizar uma soma de 32 bits. Entretanto, se for realizado o encapsulamento de 32 somadores de um bit em uma caixa preta, abstrai-se a complexidade da configuração dos 32 somadores na reconfiguração da UFR, tornando o controle mais simples para a execução da mesma tarefa. Neste caso, denomina-se grão fino para a primeira abordagem e grão grosso para a segunda.

2.1.3 Mecanismo de Reconfiguração

Várias propostas já foram apresentadas em relação ao mecanismo de reconfiguração da UFR. Em (HUTCHINGS, 1997; HUTCHINGS, 1999) foram demonstrados mecanismos que, em tempo de compilação, extraem partes do programa que podem ser executadas de forma mais eficiente na UFR. Entretanto, estas técnicas são dependentes de algum tipo de ferramenta para realizar esta tarefa, modificando o código compilado original. A conseqüência deste tipo de abordagem é o aumento do tempo do projeto com a adição de uma nova etapa no fluxo, além de não prover compatibilidade de software, já que existe a necessidade de recompilação do código.

Stitt (LYSECKY, 2006) foi um dos pioneiros a utilizar técnicas que, em tempo de execução, detectam partes do código da aplicação que podem ser executadas de forma eficiente na unidade reconfigurável. As vantagens providas por esta abordagem é a não modificação do código compilado, provendo compatibilidade de software e a conseqüente diminuição do *time-to-market* do dispositivo.

2.1.4 Exemplos de abordagens propostas

Existem várias abordagens propostas no meio científico e comercial que utilizam um meio reconfigurável para prover aceleração e flexibilidade na execução de aplicações. Nesta subseção serão demonstrados os trabalhos mais relevantes que contribuíram na propagação do uso de arquiteturas reconfiguráveis.

O principal componente do sistema proposto em (HAUCK, 1997) é uma arquitetura reconfigurável que consiste em um FPGA projetado para suportar computações intensivas. Este sistema, chamado Chimaera, é acoplado fortemente a um processador de propósito geral que compartilha recursos, como banco de registradores, com a arquitetura reconfigurável. O acoplamento utilizado por esta abordagem torna a UFR uma unidade funcional do processador, diminuindo o tempo de comunicação e de reconfiguração do FPGA.

O processador GARP (HAUSER, 1997) estende o conjunto de instruções MIPS-II, para executá-las em uma arquitetura reconfigurável fracamente acoplada a um processador de propósito geral. A comunicação entre o PPG e a UFR é realizada através de instruções dedicadas. Um ambiente completo de processamento foi projetado para esta arquitetura, incluindo bibliotecas e memória virtual.

Em (VASSILIADIS, 2001) foi proposto a arquitetura reconfigurável Molen baseada em FPGA, sendo foco de otimização desta abordagem são as partes críticas da aplicação, ou seja, laços e sub-rotinas. A reconfiguração do FPGA é realizada em um grão fino. Além da arquitetura, Molen propõe um novo paradigma de programação que permite que o código de processadores de propósito geral e descrições de hardware sejam agregados em um mesmo código de programa. Ainda, neste trabalho é proposto um novo conjunto de instruções e uma micro-arquitetura baseada em micro-código.

PipeRench (GOLDSTEIN, 2000) propõe uma computação reconfigurável que utiliza a técnica de pipeline para diminuir a latência de reconfiguração e execução em um FPGA. O esclarecimento desta técnica, chamada virtualização, está ilustrado na Figura 2.1. Nesta Figura é demonstrada uma aplicação que foi dividida em 5 estágios de pipeline (Figura 2.1(a)), levando 7 ciclos para ser configurada e executada. Entretanto, a Figura 2.1(b) demonstra a aplicação da técnica de virtualização nesta aplicação, onde somente 3 estágios foram necessários para executá-la. Sendo que o estágio 1 da Figura 2.1(b) foi utilizado, em diferentes períodos de tempo, para executar o estágio 1 e 4 da Figura 2.1(a). Piperench propõe um acoplamento da UFR anexado ao processador, além de utilizar um compilador para realizar otimizações no código, focando o favorecimento da técnica de virtualização.

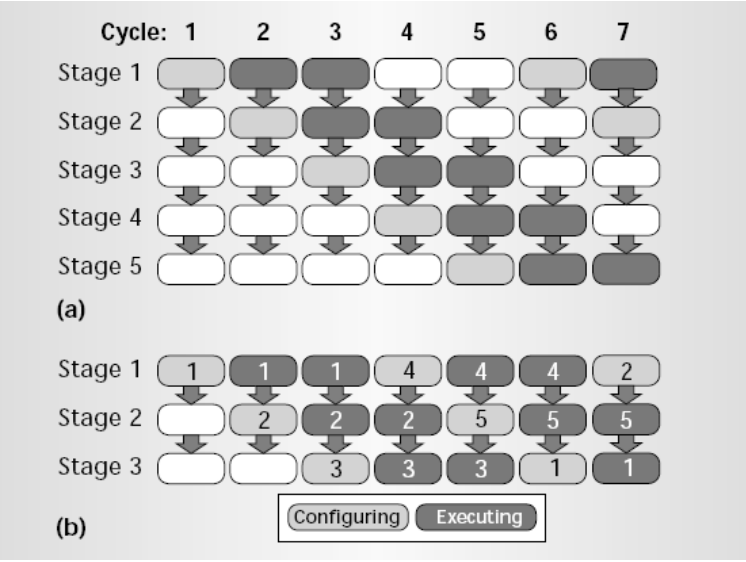

Figura 2.1: Exemplo de uma execução na arquitetura PipeRench

Os primeiros estudos sobre os benefícios de realizar dinamicamente o particionamento do software para ser executados em um hardware reconfigurável foram reportados em (LYSECKY, 2006). O *warp processing* é composto por dois microprocessadores, um sendo responsável por executar a aplicação, e outro gerenciando um algoritmo de particionamento. Além destes processadores, uma memória local e um FPGA compõem a arquitetura. Nesta abordagem existe um hardware que detecta as regiões criticas da aplicação, repassando esta para o algoritmo

de particionamento realizar a compilação para um grafo de controle. Este mesmo algoritmo sintetiza e mapeia na estrutura do FPGA o grafo previamente construído.

TRIPS (SANKARALINGAM, 2004) é um exemplo de uma arquitetura híbrida Von-Neumann/*Dataflow*, para melhor explorar o paralelismo da aplicação esta arquitetura disponibiliza um grande número de recursos, além de utilizar três diferentes modos de execução: D-Morph que explora o paralelismo em nível de instrução; T-Morph que trabalha no nível de threads; e S-Morph que é apropriada para aplicações do tipo *streaming* que apresenta um alto nível de paralelismo de dados. Uma abordagem totalmente *Dataflow* é apresentada em (SWANSON, 2003): Wavescalar, que abandona totalmente o conceito de contador de programa e a forma linear de execução de Von-Neumann, que como explicitado anteriormente, limita a quantidade de paralelismo a ser explorado na aplicação. A principal característica desta arquitetura é que ela possui centenas de centros de processamento distribuídos ao invés de uma única unidade central.

Tanto TRIPS quanto Wavescalar mostraram excelentes resultados em relação a aumento de desempenho. Entretanto, a complexidade de implementação de um compilador, causada pela complexa alocação das operações nas unidades de processamento, e a indisponibilidade atual de área de silício para implementá-la, torna esta uma abordagem inadequada para a tecnologia atual.

Em (BECK, 2005) foi proposto um algoritmo de tradução binária que dinamicamente realiza a tradução de uma seqüência de instruções em um mapeamento da unidade reconfigurável. Neste caso, a arquitetura alvo desta abordagem é um processador que executa nativamente Java. A estrutura reconfigurável é caracterizada como grão grosso por conter replicações de unidades funcionais equivalentes as unidades do processador. Além disto, a comunicação entre o PPG e UFR é desempenhada por barramento dedicado dentro do núcleo, o que a classifica como uma UFR fortemente acoplada. Entretanto, em (BECK, 2006a) foi idealiza a modificação do algoritmo de tradução binária, sendo uma arquitetura RISC o alvo de implementação deste estudo.

Este último estudo mostra-se parecido quanto ao acoplamento a Chimaera. Entretanto, em Chimaera e Garp a execução na plataforma reconfigurável necessita de uma ferramenta que, em tempo de compilação, modifica o código e adiciona instruções especiais. Este passo é evitado em (BECK, 2006) com a implementação do algoritmo dinâmico de tradução binária. Em (LYSECKY, 2006) a detecção de partes da aplicação também é realizada dinamicamente. Porém, o algoritmo de particionamento requer uma grande quantidade de memória (aproximadamente 8MB) para sua execução. Ainda, o FPGA que compõe a arquitetura provê uma longa latência de configuração, além de consumir uma quantidade considerável de área e potência estática. A latência de reconfiguração também é um problema destacado em Molen, devido a sua granularidade fina de reconfiguração.

2.2 Exploração de Recursos de Hardware

Diversos trabalhos já foram propostos sobre exploração de aplicações, com a finalidade de medir sua demanda computacional. O principal objetivo é realizar estudos sobre o comportamento das aplicações para modelar arquiteturas otimizadas em área que execute estas de maneira eficiente.

Em (CLARK, 2002) um compilador é utilizado para identificar partes críticas da aplicação. Após isto, estas partes são analisadas e assim, são determinados quais são os requisitos de hardware necessários para acelerar sua execução. Nesta análise algumas características são levadas em conta: desempenho, área, requisitos de entrada e saída. Desta forma, o código da aplicação é modificado com a inserção de novas instruções. Como estudo de caso desta proposta foi utilizado uma arquitetura reconfigurável de grão grosso e fortemente acoplada a um processador ARM.

Clark (2003) apresentou um aperfeiçoamento da exploração realizada em (CLARK 2002), limitando a exploração sobre um subconjunto de grafos de instruções. Neste trabalho é demonstrada uma técnica de escolha de grafos candidatos a exploração, a aplicação é analisada e assim é identificado o conjunto de sub-grafos que possuem computação significante. Desta forma, estes sub-grafos são transformados em novas instruções, sendo estas adicionadas ao código da aplicação. Como estudo de caso este trabalho utilizou um processador VLIW de quatro palavras baseado no conjunto de instruções ARM7.

Similar a (CLARK, 2002), o trabalho apresentado em (ARNOLD, 2001) propõe um método automático que explora a extensão de um conjunto de instruções baseados em um processador VLIW. Um compilador é usado para detectar ocorrências freqüentes de padrões de operações e calcular o número de recursos necessários que devem ser disponibilizados para executar tal aplicação. Então, estas instruções são construídas e incorporadas ao conjunto de instruções do processador.

Em (HUANG, 1994) é apresentado um ambiente de automatização de projeto chamado ASIA (do inglês, *Automatic Synthesis of Instruction-Set Architecture*). Este método recebe como entrada uma aplicação, representada como micro-operações; uma função objetivo; e um modelo de máquina pipeline. A função objetivo é fornecida pelo usuário, onde é definido o tempo de execução da aplicação, o tamanho do conjunto de instruções, e o custo do hardware. O modelo da máquina especifica a configuração dos estágios do pipeline. Com estas informações, a ferramenta explora a aplicação, fornecendo como saída um conjunto de instruções e os recursos de hardware necessários para otimizar a função objetivo repassada. Além disto, disponibiliza o código que é compilado de forma otimizada para o conjunto de instruções gerado.

Um sistema que, a partir de uma dada aplicação, automaticamente modela um processador ASIP é proposto em (GOODWIN, 2003). AutoTIE verifica um largo conjunto de ASIPs gerados e, levando em conta restrições de custos de hardware, elege o processador que fornece o melhor compromisso entre custo e desempenho. O primeiro passo realizado pela ferramenta é a escolha de um conjunto de instruções para a aplicação. Utilizando um compilador C/C++ são identificadas as regiões críticas da aplicação. Durante esta análise, técnicas de otimizações comuns em compiladores são realizadas. Após esta fase, é gerado um conjunto de instruções, adicionais à arquitetura, projetadas especificamente para a aplicação em questão. Durante este passo são realizadas estimativas de desempenho e custos de hardware para cada instrução adicional. Variando recursos de hardware e desempenho, é possível criar diversas arquiteturas que fornecem níveis diferentes de aceleração da aplicação. Assim, é possível escolher o ASIP que melhor se encaixa nos custos de hardware repassados.

Todas as abordagens demonstradas acima realizam a exploração de recursos de hardware através da construção de grafos de dependências de dados de uma dada aplicação. Entretanto, todas recaem no mesmo problema, a exploração é realizada com

o suporte de um compilador, assim o comportamento da aplicação não é levado em conta na geração da arquitetura. Ainda, o resultado da exploração é um código binário modificado, o que não produz compatibilidade de software.

Como contribuição para este trabalho foi desenvolvida a ferramenta ARISE, que será explicitada na Seção 4.1, esta modifica o paradigma de exploração de recursos na formação de uma arquitetura. O gerenciamento de recursos proposto em ARISE também constrói os grafos de dependências de dados de uma dada aplicação, entretanto estes grafos são gerados a partir da execução da aplicação. Desta forma, consegue-se extrair informações que influenciam na modelagem da arquitetura. O reuso dos grafos é uma característica que reflete diretamente na aceleração da aplicação, sendo que esta característica não pode ser extraída em tempo de compilação.

Outra característica desta ferramenta é que após o término da execução, nenhuma modificação do código é realizada, a compatibilidade de software é provida, causando uma diminuição no tempo de projeto do dispositivo embarcado. Ainda, o resultado fornecido é uma melhor exploração do paralelismo da aplicação, gerando uma arquitetura otimizada em área e potência consumida.

Resumindo, é provida uma ferramenta que fornecida uma aplicação, extrai os recursos necessários para acelerá-la. Disponibilizando, como resultado, um hardware reconfigurável otimizado que consegue explorar de maneira *dataflow* o paralelismo disponível na aplicação.

2.3 A técnica de redução de potência

O funcionamento de um dispositivo embarcado, na sua maioria, é dependente de alimentação por bateria. A preocupação sobre consumo de potência nestes dispositivos surge a partir desta realidade. Várias técnicas de redução do consumo de potência em circuitos digitais já foram propostas. Entretanto, poucas delas são destinadas a arquiteturas reconfiguráveis. Assim, para este trabalho foi realizado um estudo em algumas técnicas, para selecionar a que melhor se beneficia da reconfigurabilidade disponibilizada por este tipo de arquitetura. Técnicas como *Clock Gating* e *Dinamic Voltage Scaling* são largamente utilizadas em circuitos convencionais (JEJURIKAR, 2004; BENINI, 1997; TELLEZ, 1995). Entretanto, estas não são alternativas ideais para as novas tecnologias, visto que, focam na redução de potência dinâmica e não na potência estática.

Sleep transistor (TSCHANZ, 2003) foi proposto para atacar o consumo de potência estática. Basicamente, estes transistores são inseridos para controlar a alimentação de uma determinada parte do circuito. A utilização desta técnica é viável em partes do circuito que permanecem uma grande faixa de tempo sem realizar computações. Entretanto, alguns efeitos no desempenho do circuito podem ser introduzidos com o uso desta técnica, além de ocasionar um aumento da área total da arquitetura.

Para o estudo da modelagem da técnica de *Sleep Transitor* na arquitetura reconfigurável deste trabalho, foi idealizado um mecanismo que ataca não apenas o consumo de potência estática. A interligação do sistema de tradução binária e da técnica em questão soluciona o elevado consumo de potência dinâmica do circuito reconfigurável. A conseqüência da formação da unidade funcional reconfigurável por somente lógica combinacional é o constante chaveamento de bits em todas as unidades funcionais, mesmo naquelas que não estão sendo utilizadas para computação. O

transistor de desligamento se beneficia dos bits de configuração existentes e os utiliza para realizar o corte de alimentação de uma determinada parte da arquitetura. Assim, no momento da execução, os blocos que não estão com recursos alocados não recebem alimentação. Desta forma, estes não irão consumir tanto a potência estática, pois não existe alimentação, quanto potência dinâmica, pois não há chaveamento de bits.

3 O SISTEMA RECONFIGURÁVEL

O sistema reconfigurável utilizado como estudo de caso deste trabalho foi proposto no laboratório de Sistemas Embarcados da UFRGS. Em (BECK, 2006a; BECK 2006b) foi explorada a utilização deste sistema no simulador Simplescalar (BURGER, 1997), executando o conjunto de instruções PISA. A partir desta implementação demonstraram-se ganhos significativos no desempenho do processador.

No simulador Simplescalar é possível escolher um vasto número de parâmetros para a arquitetura do processador: número de unidades funcionais disponíveis; atraso de cada unidade. Entretanto, torna-se árduo o trabalho de modificar a arquitetura em si, ou seja, modificar o seu conjunto de instruções.

Um dos objetivos deste trabalho foi disponibilizar o sistema reconfigurável em uma plataforma real. Assim, seria possível obter resultados em relação ao desempenho do sistema, e também demonstrar o consumo de potência e a área ocupada. Para isto foi escolhido o processador MIPS R3000 como alvo de implementação do sistema reconfigurável (BECK, 2008), e o principal motivo desta escolha é a ampla utilização deste processador no domínio de sistemas embarcados.

A Figura 3.1 ilustra um diagrama de blocos do sistema completo, os blocos em cinza escuro representam os estágios do pipeline do processador MIPS R3000. Em cinza claro, estão os blocos que simbolizam as unidades, que foram acopladas ao processador, responsáveis por realizar a execução de forma reconfigurável: hardware de tradução binária (TB), unidade funcional reconfigurável (UFR) e a cache de reconfigurações. Nas próximas seções deste capítulo cada um dos componentes do sistema será explorado.

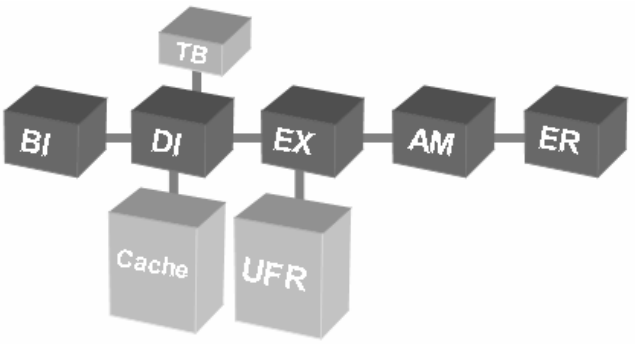

Figura 3.1: Diagrama de blocos do Sistema Reconfigurável (BECK, 2008)

3.1 Arquitetura do Processador MIPS R3000

O primeiro processador da família MIPS foi idealizado por Hennessy em 1985, chamado R2000. Este processador possui um pipeline tradicional de cinco estágios. Motivado pela grande complexidade do bloco de controle dos processadores CISC da época, desenvolveu-se um processador simples, com um conjunto de instruções reduzido e de fácil decodificação. Estas características formam um modelo de arquitetura chamado RISC (do inglês *Reduced Instruction Set Computer*).

O processador MIPS R3000 possui as mesmas características arquiteturais do seu predecessor R2000. Entretanto, diferencia-se deste último no suporte a memória cache, podendo endereçar até 64 Kb para dados e 64Kb para instruções. Outra característica inovadora deste processador é a inserção de unidade de gerenciamento de memória dentro da seu circuito integrado.

O conjunto de instruções que o processador MIPS R3000 utiliza é diferenciado em apenas três tipos. As instruções chamadas de tipo-R permitem realizar operações entre dois registradores fontes e enviar o resultado da operação para um registrador destino. Já as instruções do tipo-I utilizam um operando imediato de 16 bits e mais um registrador como fonte, enviando o resultado para um registrador destino. Finalmente, instruções do tipo-J possuem um campo de 26 bits para um operando imediato, e este tipo somente é utilizado pelas instruções de salto que alocam o valor do contador de programa nestes bits.

O pipeline do MIPS R3000, ilustrado na Figura 3.2, possui cinco estágios, divididos em:

- Busca de instruções (do inglês *Instruction Fetch*) – neste primeiro estágio é realizada a busca da instrução na memória e é efetuado o cálculo do endereço da próxima instrução a ser buscada.
- Decodificação (do inglês *Instruction Decode*) – no segundo estágio do pipeline, a instrução repassada pelo estágio de busca é decodificada. Assim, já descobertos quais registradores serão utilizados pela mesma, neste mesmo estágio seus respectivos operandos são buscados no banco de registradores.
- Execução (do inglês *Execution*) – com a instrução já decodificada e os operandos já disponíveis, neste estágio é realizada a execução da operação especificada na instrução. Instruções de acesso à memória calculam neste estágio o endereço de destino do acesso.
- Acesso à Memória (do inglês *Memory Access*) – repassado o endereço de acesso à memória pelo estágio de execução, neste estágio as instruções de acesso à memória efetivamente realizam a busca do dado na mesma.
- Escrita do Resultado (do inglês *Write Back*) – o último estágio realiza a escrita do resultado da instrução no banco de registradores para que as próximas instruções utilizem nas suas computações.

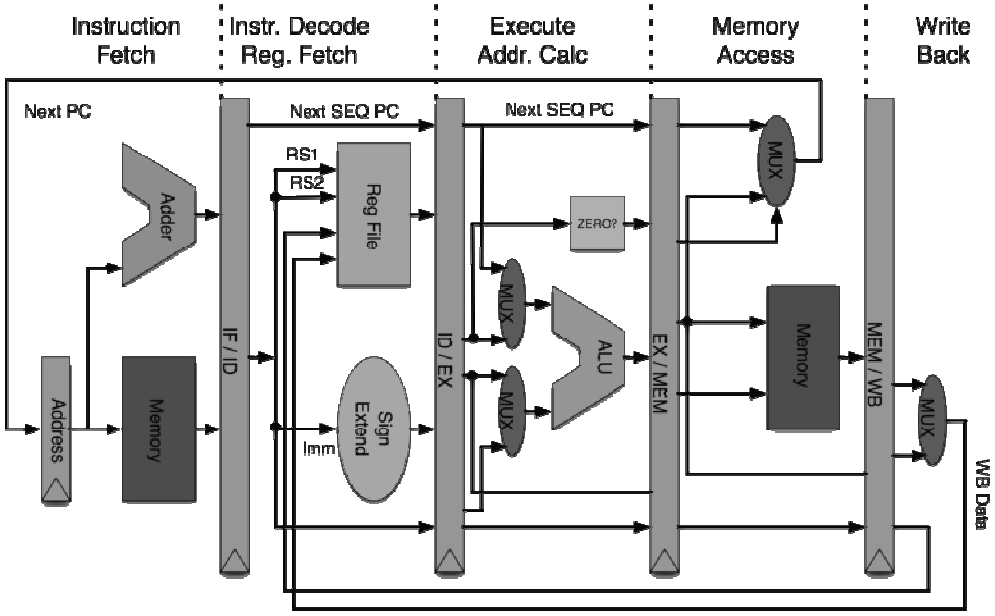

Figura 3.2: Diagrama de blocos do pipeline da arquitetura MIPS R3000. (HENNESSY, 1990)

As instruções suportadas pelo modelo de arquitetura do processador utilizado neste trabalho estão descritas na Tabela 3.1. É importante salientar que o modelo utilizado não suporta instruções de ponto flutuante, assim todas as aplicações executadas somente possuem instruções que manipulam dados inteiros.

Tabela 3.1: Instruções suportadas pelo MIPS R3000

Categoria	*Sintaxe*
Aritméticas	*add, addu, sub, subu, addi, addiu, mul, div*
Transferência de Dados	*lw, lh, lhu, lb, lbu, sw, sh, sb, lui, mfhi, mflo*
Lógicas	*and, andi, or, ori, xor, nor, slt, slti*
Deslocamento de Bit	*sll, srl, sra*
Saltos Condicionais	*beq, bne*
Saltos Incondicionais	*j, jr, jal*

Com o objetivo de tentar manter o fluxo do pipeline contínuo em toda execução da aplicação o modelo do processador utiliza a técnica de retardo de saltos (do inglês *delay slot*). Em tempo de compilação, uma instrução anterior, que não possua dependência com a instrução de salto, é escolhida para ser posicionada, no código, após o salto. Sendo esta técnica suportada pelo hardware, sempre a instrução após o salto será executada, mesmo que a condição para o salto seja verdadeira. A utilização desta técnica evita a inserção, pelo compilador, de instruções NOP (do inglês *NO oPeration*) no código da aplicação, em conseqüência do cálculo do endereço do desvio e da inserção deste endereço no contador de programa. Utilizando esta técnica o desempenho do processador é elevado, causando diminuição do tempo de execução e do número de instruções efetivas executadas.

Outra característica importante sobre o modelo utilizado é que o mesmo possui, dentro de sua organização, um banco de registradores de 32 bits, além de disponibilizar dois registradores especiais utilizados para executar instruções de multiplicação e divisão.

3.2 Arquitetura da Unidade Funcional Reconfigurável

Como explicitado na Seção 2.1, toda a unidade funcional reconfigurável possui uma classificação que depende de algumas características de implementação. A abordagem aqui utilizada propõe uma arquitetura que se classifica como: dinâmica, de grão grosso e fortemente acoplada.

A utilização de um hardware de tradução binária, que será demonstrado na Seção 3.3, torna possível a realização da reconfiguração da arquitetura em tempo de execução, por este motivo o sistema reconfigurável é classificado como dinâmico. A arquitetura proposta é formada pelas unidades funcionais descritas na Tabela 3.2, e pode-se observar que são unidades funcionais simples. O grão mínimo de reconfiguração é uma unidade funcional, assim a UFR é classificada como arquitetura de grão grosso. Finalmente, como pode ser observado na Figura 3.1, a unidade funcional reconfigurável (UFR) é acoplada, por barramento, no estágio de execução do processador. Portanto, é implementada como uma unidade funcional adicional do processador, evitando acessos externos ao núcleo e, conseqüentemente, diminuindo o tempo de reconfiguração e execução. Para este modelo classifica-se a arquitetura proposta como fortemente acoplada.

Tabela 3.2: Unidades funcionais da arquitetura e suas operações

Unidades Funcionais	*Operações Executadas*	*Instruções Alocadas*
Grupo 1: Unidade Lógica e Aritmética	Soma, Subtração, Deslocamento, Comparação, Lógica de Bit	*add, addu, sub, subu, addi, addiu, and, andi, or, ori, xor, nor, slt, slti, sll, srl, sra, beq, bne, j, jr, jal, lui, mfhi, mflo*
Grupo 2: Load/Store	Leitura/Gravação de dado na memória	*lw, lh, lhu, lb, lbu, sw, sh, sb*
Grupo 3: Multiplicador	Multiplicação	*mul*

3.2.1 Estrutura

Um exemplo de uma possível estrutura da UFR é ilustrado na Figura 3.3. Para auxiliar no entendimento, a Figura 3.3 foi dividida verticalmente em níveis, cada nível corresponde a um ciclo de relógio do processador MIPS R3000. Cada nível foi dividido em linhas, assim um nível abrange três linhas. É importante salientar que não existem barreiras temporais entre os níveis, ou seja, a execução dentro da estrutura reconfigurável é realizada de forma combinacional.

A estrutura da UFR foi modelada de forma que se possa explorar ao máximo o paralelismo de instruções existente nas aplicações. Como pode ser observado na Figura 3.3, é possível realizar dois tipos de execução na estrutura: paralelo e seqüencial. O primeiro corresponde à disposição horizontal das unidades funcionais, ou seja, quando as instruções são alocadas lado a lado horizontalmente na estrutura, elas iniciarão suas execuções no mesmo instante de tempo. Entretanto, o segundo tipo demonstra a execução de forma vertical na estrutura. Assim, instruções alocadas em linhas diferentes iniciarão suas execuções em instantes distintos.

Para simplificar a alocação das instruções na estrutura reconfigurável, as unidades funcionais foram classificadas em grupos. No exemplo ilustrado na Figura 3.3, representando o grupo um, existem quatro unidades lógicas e aritméticas dispostas horizontalmente. Assim, é possível executar até quatro operações deste tipo em paralelo. Após o cálculo do caminho crítico do processador MIPS R3000, foi verificado que, em um ciclo de relógio do processador, é possível realizar três operações lógicas e aritméticas. Conseqüentemente, em um nível (respectivo a um ciclo de relógio do processador) da arquitetura reconfigurável foram encadeadas três unidades funcionais deste grupo.

De forma análoga às unidades funcionais do grupo um, a Figura 3.3 ilustra um alinhamento horizontal de duas unidades de *Load*/*Store*, representando o grupo dois. Neste exemplo, a arquitetura é capaz de executar até dois acessos à memória em um mesmo ciclo de relógio. O número de unidades dispostas horizontalmente deste grupo é dependente da disponibilidade de portas existente no modelo de memória utilizado. Finalmente, para o grupo três, no exemplo utilizado somente uma multiplicação pode ser executada por nível da arquitetura.

Na Figura 3.3, ao lado das unidades funcionais, existe um bloco com os bits de configuração da estrutura. Para cada linha da mesma é necessária uma tabela de bits para armazenar informações sobre quais registradores estão sendo escritos nesta linha. Esta tabela é utilizada pelo tradutor binário, no momento da alocação das instruções, para verificar as dependências entre as instruções.

Em conseqüência da regularidade da arquitetura, a modelagem da abordagem possibilita que o projetista facilmente realize parametrizações no número de unidades funcionais dentro da arquitetura reconfigurável. No âmbito de sistemas embarcados este fator é de extrema importância, visto que em tempo de projeto é necessário verificar os requisitos de desempenho, área e potência do dispositivo para posterior fabricação.

3.2.2 Interconexão

Basicamente, as unidades funcionais que compõe a estrutura reconfigurável são interligadas através de multiplexadores e demultiplexadores. A Figura 3.4 ilustra o esquema de interconexão da arquitetura. A esquerda da Figura 3.4 existe o barramento que conecta o contexto de entrada ao contexto de saída, abrangendo todos os níveis da estrutura reconfigurável. O objetivo do barramento é fornecer os operandos, advindos do contexto de entrada, para unidades funcionais e receber os resultados das operações para posterior gravação destes no contexto de saída.

Os dois contextos são formados por um número de registradores pré-definido pelo projetista, sendo que este número deve ser igual ao número máximo de registradores utilizados para cada execução na estrutura reconfigurável.

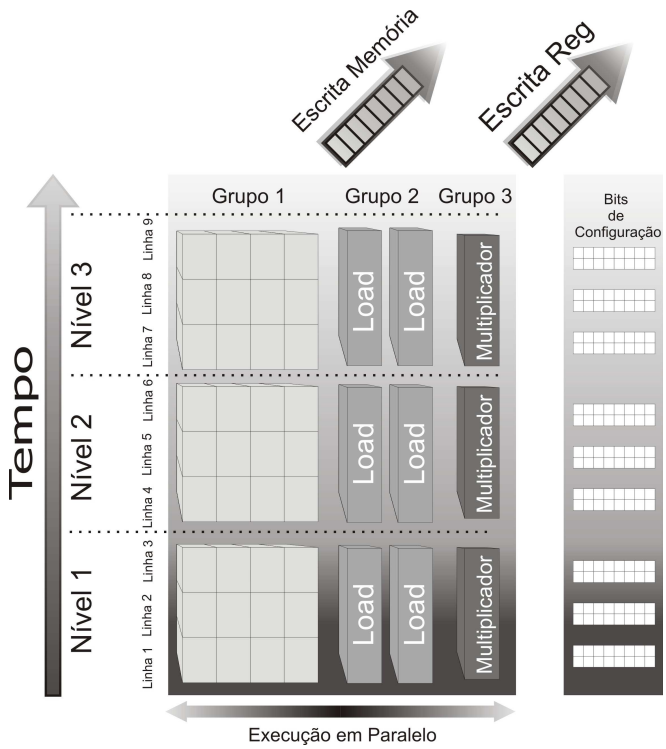

Figura 3.3: Estrutura da Arquitetura Reconfigurável (BECK, 2006a)

Ao iniciar a execução na arquitetura reconfigurável todos os operandos estão armazenados no contexto de entrada. Para realizar o roteamento destes, até as entradas das unidade funcionais, são necessários dois multiplexadores para cada uma destas unidades, como ilustrado à direita da Figura 3.4. Estes selecionam os registradores corretos a serem carregados do barramento e repassam seus valores à unidade funcional. O número de entradas de cada um destes multiplexadores é igual ao número de registradores que compõe o contexto de entrada.

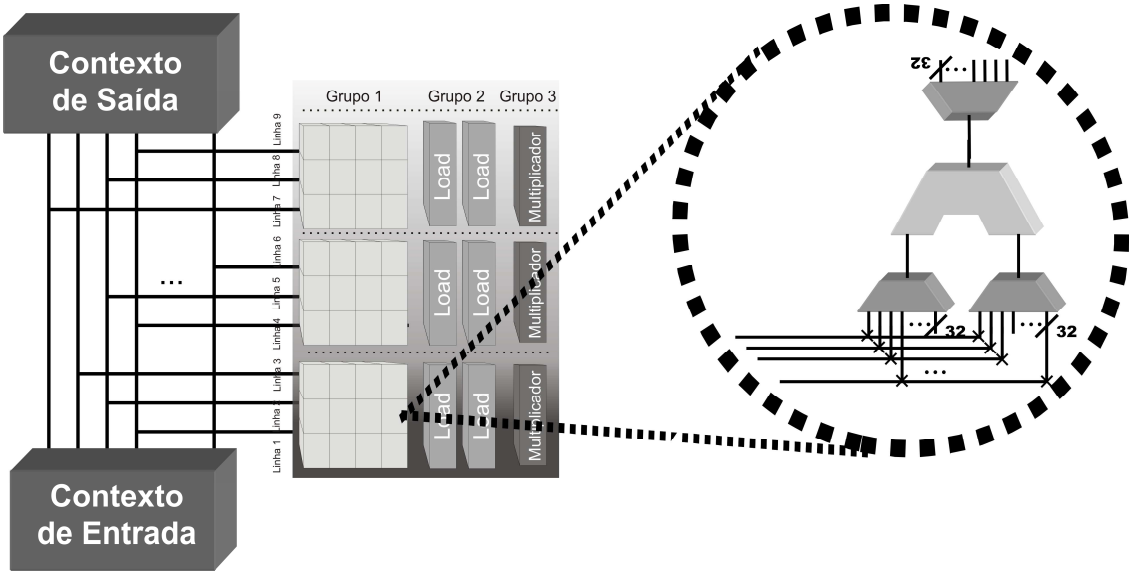

Figura 3.4: Esquemático do modelo de interconexão da estrutura reconfigurável

Após o término da execução da instrução na unidade funcional, deve-se realizar a escrita do resultado no registrador destino. Assim, um demultiplexador é inserido na saída de cada unidade funcional, seu papel é selecionar a linha do barramento onde o resultado deve ser propagado, para finalmente este ser escrito no registrador destino no contexto de saída. Análogo ao multiplexador de entrada, o número de saídas dos demultiplexadores é igual ao número de registradores que compõem o contexto de saída.

A interconexão completa da arquitetura reconfigurável provê uma grande flexibilidade na alocação das instruções na estrutura. Como será demonstrado na Seção 3.3, o hardware de tradução binária, responsável pela alocação das instruções nas unidades funcionais, é simples de ser implementado. A ampla comunicação existente entre as unidades funcionais da estrutura acarreta um grande consumo de área e , conseqüentemente, de potência em relação ao total ocupado pela arquitetura.

3.3 Tradução Binária

Uma das restrições impostas pelo projeto de um dispositivo embarcado é o tempo de projeto. A concorrência das empresas para disponibilizar no mercado o primeiro dispositivo com novas funcionalidades força o projeto embarcado ser cada vez mais curto. Assim, os projetistas devem utilizar técnicas que ajudem a modelagem e reaproveitamento do software já escrito para o dispositivo anterior.

A tradução binária é largamente utilizada em processadores de propósito geral para prover compatibilidade de software. A utilização desta técnica em sistemas embarcados foi proposta em (BECK, 2005), neste trabalho foi demonstrado que com a utilização de um tradutor binário e uma unidade reconfigurável, acoplado a um processador Java eleva-se o desempenho e diminui-se a energia consumida pelo sistema. A maior característica herdada deste trabalho foi à manutenção da compatibilidade de software.

Em (BECK, 2006a), foi demonstrado que a utilização da técnica de tradução binária também alcança ótimos resultados de desempenho e energia em um processador que executa o conjunto de instruções PISA (BURGER, 1997). Assim, para este estudo, utilizando o processador MIPS R3000, foi herdada a técnica aplicada neste último trabalho.

A idéia básica do mecanismo é prover compatibilidade de software a partir da tradução binária de seqüências de instruções, em tempo de execução, para que as mesmas possam ser futuramente executadas num mecanismo mais eficiente, neste caso, em uma unidade funcional reconfigurável.

3.3.1 Detecção e Execução

O tradutor binário (TB) proposto em (BECK, 2006a) trabalha em paralelo ao processador. Como contribuição para esta dissertação foi implementado este mecanismo no processador MIPS R3000, o mesmo foi descrito em VHDL, uma linguagem de descrição de hardware.

Basicamente, o mecanismo avalia cada instrução executada pelo processador, agrupando-as em blocos chamados de configuração da UFR. A cada instrução executada pelo processador é verificada a possibilidade da execução desta na unidade reconfigurável. Caso positivo, a mesma é alocada na configuração corrente da UFR. Ao final da construção de uma configuração da UFR, esta é armazenada em uma cache de

reconfigurações, indexada pelo valor do contador de programa da primeira instrução. Quando este valor novamente for alcançado pelo contador de programa, o mecanismo reconfigurável é ativado seguindo os seguintes passos:

- Efetua-se a busca da configuração, da seqüência de instruções em questão, na cache de reconfigurações;
- Configura-se a UFR com os bits fornecidos pela cache;
- Os valores dos registradores necessários para executar a configuração são carregados no contexto de entrada da UFR;
- A execução é realizada na UFR;

Após o término da execução na UFR o valor do contador de programa é atualizado, assim como os valores dos registradores modificados no banco de registradores. Do mesmo modo as escritas na memória de dados são realizadas. É importante destacar que enquanto o mecanismo reconfigurável está ativo o processador não realiza nenhuma operação.

Diferentemente dos processadores superescalares, esta abordagem não realiza repetidas vezes, para a mesma seqüência de instruções, a verificação das dependências entre as instruções. A utilização da técnica de reuso de rastros (do inglês *trace reuse*) evita a repetição desta tarefa. Portanto, se não houver falha na cache de reconfiguração, o mecanismo de TB somente será aplicado uma única vez em cada seqüência de instruções.

Em relação a mecanismos utilizados em outros sistemas reconfiguráveis, que somente aplicam as técnicas de reconfiguração em partes mais executadas da aplicação, a abordagem de TB descrita neste trabalho fornece uma maior flexibilidade. A aplicação inteira sofre análise do TB, não se limitando a apenas partes específicas da aplicação.

O hardware de TB é composto basicamente por tabelas e mapas de bits que armazenam temporariamente dados da configuração corrente. O algoritmo é composto por seis tabelas e dois mapas de bits que serão especificadas a seguir:

- Mapas de Escrita – a função deste mapa é armazenar o número do registrador de escrita de cada instrução alocada na UFR. Este mapa é utilizado na verificação das dependências entre as instruções no momento de alocação das mesmas na arquitetura. Em cada linha da UFR existe um mapa de escrita, sendo o número de bits igual ao número de registradores existentes no contexto de entrada. Este mapa de bits não será armazenado na configuração final, pois não tem utilidade no momento da execução.
- Mapas de Recurso – este mapa foi inserido no mecanismo para gerenciar a alocação de recursos na UFR. Assim, no momento da inserção de uma nova instrução em uma configuração, é realizada a busca por uma unidade funcional ociosa neste mapa. Analogamente ao mapa de escrita, os dados do mapa de recurso não serão inseridos na configuração.
- Tabela de Contexto Atual – armazena uma referência a todos os registradores que serão utilizados pelas instruções executas por uma configuração.
- Tabela de marcadores de contexto atual – faz a caracterização dos registradores da tabela de contexto atual, distinguindo entre registradores de leitura e de escrita.

- Tabela de Contexto Inicial de Leitura – nesta tabela são inseridas as referências a todos os operandos de leitura armazenados na tabela de marcadores de contexto atual. A inserção é feita na posição correspondente à posição do operando na tabela de contexto atual. A função desta tabela é indicar quais os registradores devem ser carregados no contexto de entrada no inicio da execução.
- Tabela de Imediatos – as instruções do tipo-I e do tipo-J trazem em seu corpo operandos imediatos. Para que estas possam ser executadas na UFR é necessário armazenar o valor destes operandos. Assim, a função desta tabela é armazenar os valores imediatos, para que no momento da execução, sejam carregados para o contexto de entrada.
- Tabela de Leituras – a função desta tabela é armazenar quais registradores serão entradas de cada unidade funcional. Especificamente, os dados desta tabela serão inseridos, na hora da execução, nos bits de controle dos multiplexadores de entrada de cada unidade funcional. O modo que esta tabela indica os registradores é referenciando a posição dos mesmos na tabela de contexto inicial.
- Tabela de Escrita – analogamente a tabela de leituras, a tabela de escrita armazena a referência para a coluna em que o recurso foi alocado na UFR. A definição da posição de escrita nesta tabela corresponde à mesma posição em que este operando está na tabela de contexto. Estes valores servirão como controle dos demultiplexadores alocados após as unidades funcionais, com o objetivo de realizar a escrita nos registradores do contexto de saída.

A Figura 3.5 demonstra o hardware de tradução binária acoplado ao processador MIPS R3000. O hardware foi dividido em quatro estágios para não infringir o caminho crítico do processador. Os estágios que estão em cinza escuro são estágios do processador e os estágios que estão em cinza claro são estágios do hardware de TB. Cada instrução executada pelo processador é analisada pelo TB que realiza a alocação destas nas tabelas e mapas de bits que compõe o hardware de detecção. Abaixo é demonstrada a análise de execução de cada estágio do algoritmo. A descrição realizada em VHDL do TB para esta dissertação ocupou o total de 1566 linhas de código, a área ocupada em hardware por esta descrição será explicitada na Seção 3.6.5.

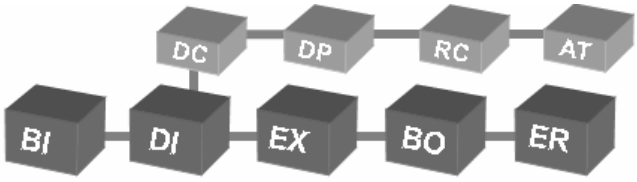

Figura 3.5: Estágios do Tradutor Binário anexados ao pipeline do processador

- Decodificação (DC) – neste estágio é realizada a decodificação dos campos da instrução, estas informações servirão para identificar características como:
    - Grupo da Instrução: em qual grupo da unidade reconfigurável a instrução deve ser alocada.

- Tipo e Função da Instrução: em qual unidade funcional do grupo a instrução deve ser alocada, além de qual função a unidade em questão deve desempenhar para executar a instrução.
- Registradores de Leitura e Escrita: identificam quais são os registradores de leitura e escrita da instrução.
- Operandos Imediatos: identifica, se a instrução possuir, operandos imediatos.

A descrição em VHDL deste estágio ocupou 412 linhas de código.

- Dependência (DP) – após a instrução estar decodificada e seus campos identificados, o papel deste é verificar as dependências de dados existentes entre as instruções. Este trabalho será útil para o próximo estágio alocar as instruções nas unidades funcionais. Dependências de dados verdadeiras são bastante comuns entre instruções em um fluxo de execução, então se deve tomar o devido cuidado na execução destas garantindo a consistência dos dados. Sua descrição em VHDL ocupou 184 linhas de código.
- Recursos (RC) – no estágio anterior são detectadas as dependências verdadeiras existentes no fluxo de execução entre as instruções. Neste estágio é verificada a disponibilidade de unidades funcionais na UFR e realizada a alocação de uma destas para que a instrução em questão possa ser executada. Sua descrição em VHDL ocupou 202 linhas de código.
- Atualização de Tabelas e Mapas de Bits (AT) – neste estágio todas as tabelas e mapas demonstrados anteriormente são atualizados. Assim, a cada instrução adicionada à UFR, é realizada a atualização nas tabelas com as devidas informações necessárias para que a instrução seja executada. Deste modo, é garantido que, no momento da execução, a instrução seja alocada, executada e o seu resultado seja armazenado no registrador destino. Ao final da construção de uma configuração, todos os dados das tabelas necessárias serão empacotados, e a mesma será armazenada na cache de reconfigurações. Sua descrição em VHDL ocupou 768 linhas de código.

3.3.2 Especulação

Como mostrado na Seção 3.3.1, a detecção de um bloco de instruções para montagem de uma configuração ocorre de forma seqüencial, em momento de execução. Entretanto, alguns fatores podem interromper a formação deste bloco e, conseqüentemente, a montagem de uma configuração. O primeiro fator são as instruções que não tem suporte de execução na UFR, exemplos destas são as instruções que realizam operações de divisão. No momento em que este tipo de instrução é encontrado pelo TB, a configuração corrente é concluída, e armazenada na cache de reconfigurações. Posteriormente, quando uma instrução com suporte de execução na UFR é encontrada, uma nova configuração é iniciada.

Outro fator que interrompe a formação de uma configuração são as instruções de salto. Assim, a cada instrução de salto executada pelo processador a configuração corrente é concluída e uma nova configuração é iniciada (BECK, 2006a). Em (BECK, 2006b) foi proposto um mecanismo de especulação de saltos, este cria árvores de execução que são utilizadas na formação das configurações da UFR.

A utilização de especulação possibilita a formação de configurações que ultrapassam a execução de instruções de salto, impactando diretamente no desempenho da execução da aplicação pela inclusão de um número maior de instruções em uma única configuração. O número de saltos executados em uma única configuração da UFR pode ser definido pelo projetista. Entretanto, a penalidade por erro de especulação é proporcional ao crescimento do nível especulado.

3.4 Cache de Reconfiguração

O sistema reconfigurável, fundamentalmente, explora três técnicas amplamente difundidas no meio científico: reconfigurabilidade, tradução binária e reuso. Esta última explora a natureza de execução do modelo Von-Neumann, onde o contador de programa é o elemento que dirige o fluxo de execução. Assim, a existência de um laço ou de saltos remete a repetição de certo trecho de código da aplicação.

A abordagem proposta explora esta característica das aplicações, realizando tradução binária destes trechos de códigos e armazenando as configurações realizadas em uma cache, evitando a repetitiva análise de código realizada pelos processadores superescalares.

Cada posição da cache de reconfigurações armazena dados necessários para a execução de uma configuração na UFR. Como já explicitado anteriormente, cada configuração armazena os bits necessários para controlar os multiplexadores e demultiplexadores, bits de controle das funções de cada unidade funcional, além dos dados imediatos contidos nas instruções.

Na extração dos resultados de (BECK 2006a, BECK 2006b) o algoritmo de substituição FIFO (do inglês *first in, first out*) foi utilizado. Como contribuição para esta dissertação uma maior exploração desta característica foi realizada. Algoritmos tradicionais de substituição foram implementados; LRU (do inglês *least recently used*); LFU (do inglês *least frequently used*) e o algoritmo randômico. Em alguns casos o algoritmo LRU e FIFO obtiveram o mesmo número de faltas na cache. Entretanto, na maioria das aplicações o algoritmo FIFO demonstrou melhores resultados, ou seja, este algoritmo é capaz de abranger a região exata da execução temporal das configurações.

Nos trabalhos anteriores, foi verificado que, devido a restrições de área e potência dos sistemas embarcados, 512 blocos é o tamanho máximo ideal para armazenar as configurações na cache. Visando aperfeiçoar o desempenho da cache e verificando que 512 é um número razoavelmente pequeno de blocos, a cache é implementada com o modelo totalmente associativo. Apesar do maior consumo de área e potência deste modelo, em relação a outros propostos, este foi escolhido devido ao seu rápido tempo de envio da configuração à UFR.

3.5 Metodologia

A metodologia da primeira contribuição relevante deste trabalho, a implementação do sistema reconfigurável na plataforma MIPS R3000 será explicitada nesta seção, além disto, irá ser apresentado os fluxos de extração de resultados de desempenho, potência e área do sistema acoplado ao processador.

3.5.1 ArchC

Como citado anteriormente, os trabalhos anteriores exploraram o sistema reconfigurável no simulador Simplescalar, executando o conjunto de instruções PISA. Como contribuição deste trabalho foi realizada a modelagem e validação da arquitetura reconfigurável na linguagem de descrição de hardware (LDH) ArchC (RIGO, 2004).

A LDH ArchC é uma linguagem de descrição de hardware, baseada em SystemC, que permite, em alto nível de abstração, modelar e verificar uma arquitetura. Após a modelagem é gerado, de forma automática, um conjunto de ferramentas capaz de simular e verificar a modelagem realizada. A vantagem de modelar o comportamento de uma arquitetura em alto nível de abstração é evidente em projetos embarcados, a redução no tempo de escrita e validação da arquitetura é refletida diretamente no tempo em que o dispositivo final é disponibilizado para o mercado.

Para este trabalho foi utilizada uma modelagem, em ArchC, *instruction accurate* da arquitetura do processador MIPS R3000. Como citado anteriormente, após a realização modelagem do sistema reconfigurável na mesma linguagem, os dois sistemas foram acoplados e validados.

Desta modelagem, como poderá ser observado na Seção 3.6, foram extraídos resultados de desempenho do sistema, executando um conjunto de aplicações do conjunto MiBench (GUTHAUS, 2001).

3.5.2 Fluxo de Potência

O consumo de potência de um circuito digital pode ser estimado em diferentes níveis de implementação. Níveis mais detalhados, como especificações SPICE do circuito, fornecem resultados precisos, porém seu tempo de simulação se torna proibitivo. Uma alternativa para este tipo de projeto é a elevação do nível de abstração. Assim, foi desenvolvido um fluxo de estimativa de potência em nível de descrição de circuito RTL (do inglês *Register Transfer Level*), onde se agrega duas características vantajosas: curto tempo de simulação e relativa precisão na estimativa. (LANDMAN, 1995)

Neste trabalho foram escolhidas algumas ferramentas amplamente difundidas no meio comercial que agregadas caracterizam um fluxo de estimativa de potência (Figura 3.6). Como entrada para fluxo é repassada uma descrição RTL em VHDL do circuito digital. A ferramenta Design Compiler, desenvolvido pela Synopys Inc., sintetiza este circuito para *standard cell* utilizando uma biblioteca de tecnologia que é disponibilizada por indústrias de semicondutores. Além disto, esta ferramenta fornece um arquivo chamado "*Forward Annotation File*", que é gerado a partir da biblioteca de células. Este contém informações, necessárias para a etapa posterior, sobre dependências de caminho entre células lógicas.

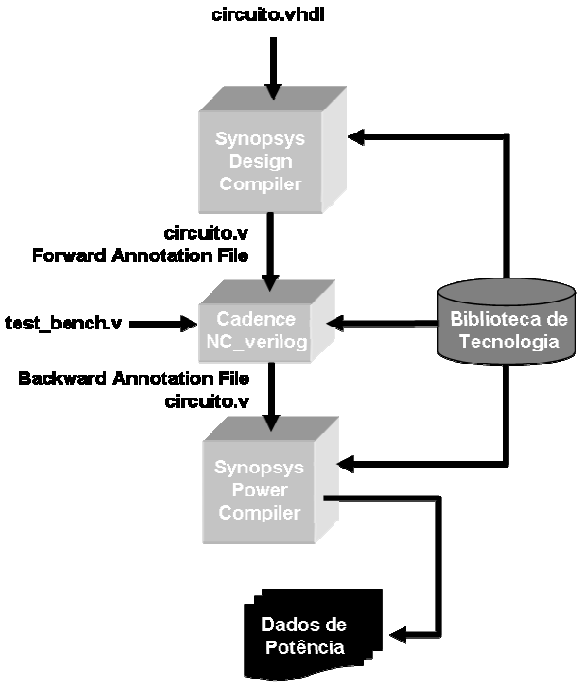

Figura 3.6: Fluxo de estimativa de potência

Após a síntese do circuito para células lógicas da biblioteca é realizada a simulação do circuito para extração da atividade de chaveamento. NC-Verilog, desenvolvido pela Cadence Design Systems, é o simulador de circuitos digitais utilizado pelo fluxo. Como entrada, para este simulador é fornecido o circuito mapeado anteriormente para a biblioteca específica, assim como o arquivo "*Forward Annotation File*". Além destes, também é repassado um arquivo, gerado pelo usuário, que produz a excitação do circuito e a biblioteca de células padrão em que o este foi mapeado.

Após a simulação do circuito, obtém-se como saída um arquivo chamado "*Backward Annotation File*" que contém dados sobre a atividade de chaveamento. Como já explicitado anteriormente, o consumo de potência estática, nas novas tecnologias CMOS, está se tornando cada vez mais significativo em relação ao consumo de potência total do circuito. Ciente disto, o principal requisito de seleção das ferramentas que iriam compor o fluxo foi o suporte à estimativa de potência estática.

No âmbito deste trabalho, para estimativa de potência foram realizadas as descrições RTL, em VHDL, de todos os componentes do sistema reconfigurável, abrangendo: unidade funcional reconfigurável, mecanismo de tradução binária e cache de reconfigurações. A descrição RTL, também em VHDL, do processador MIPS R3000 foi obtida através do repositório Opencores.

3.6 Resultados

Nesta seção será demonstrado o impacto que a inserção do sistema reconfigurável causou no desempenho, consumo de potência e na área ocupada pelo processador MIPS R3000.

3.6.1 Caracterização da Infra-estrutura

Como já explicitado anteriormente, os resultados de desempenho do sistema reconfigurável foram obtidos, através de simulação, a partir de uma modelagem baseada

em SystemC na linguagem de descrição de hardware ArchC. Em relação à extração de resultados de área, foi utilizada a ferramenta Leonardo Spectrum, desenvolvida pela Mentor Graphics. Esta ferramenta recebe como entrada o circuito, descrito na linguagem VHDL, e utiliza uma biblioteca de "*standard cell"* para sintetizá-lo e, assim, estimar a área ocupada pelo circuito. É importante ressaltar que o circuito de Tradução Binária utilizado para extração de área e potência não foi considerada a técnica de especulação.

A partir da execução das aplicações no simulador ArchC foi possível verificar a taxa média de chaveamento de bits nas entradas das unidades funcionais da UFR. Demonstrou-se em (GEGLER, 2007) que a taxa média de chaveamento na entrada de uma unidade funcional entre dois dados consecutivos é de três bits no processador MIPS R3000. Resultados estimados de potência consumida pelo sistema foram obtidos através do fluxo descrito na Seção 3.5.2 utilizando a taxa de chaveamento de três bits com uma freqüência de 200 MHz. Para ambos os fluxos, área e potência, é necessário a presença de uma biblioteca de *standard cell*. Assim, nos resultados que serão apresentados a seguir foi utilizada a biblioteca TSMC 0.18um.

Um conjunto de aplicações é necessário para avaliar o impacto desempenho e potência do sistema reconfigurável. *Mibench* foi o conjunto escolhido pelo fato deste disponibilizar aplicações com comportamentos de execução distintos. Como pode ser observado na Figura 3.7, o número de instruções executadas entre saltos variam drasticamente dentro deste conjunto. A aplicação *ADPCM* (do inglês *Adaptative Differential Pulse-Code Modulation*), que é uma técnica de conversão de dados de som em informação binária, possui um comportamento muito orientado a controle. Em outro extremo, o algoritmo de criptografia *Rijndaeld* é o que possui mais instruções executadas entre saltos, ou seja, comportamento orientado a dados. Conseqüentemente, com este conjunto heterogêneo de aplicações pode-se avaliar de uma forma justa o desempenho do sistema reconfigurável.

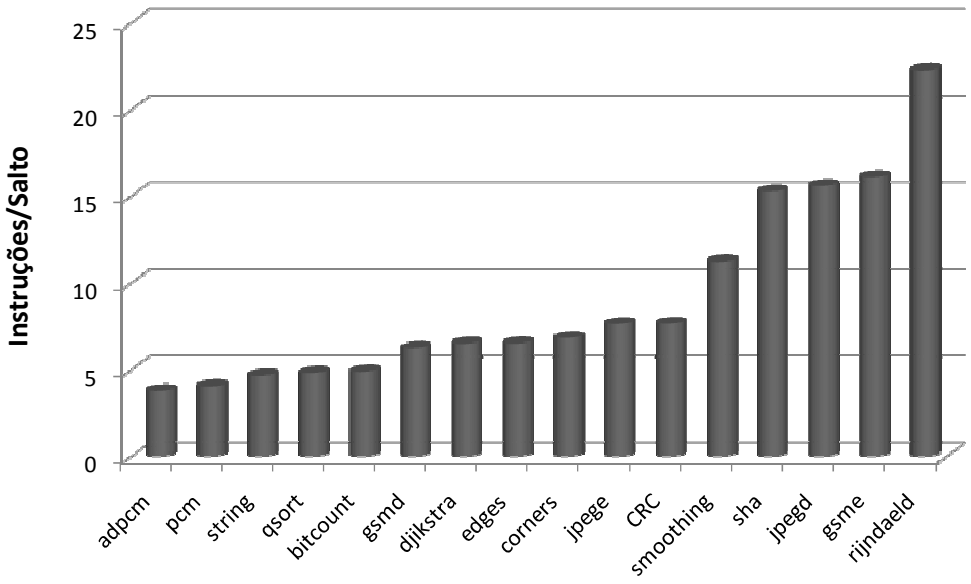

Figura 3.7: Diferentes comportamentos das aplicações utilizadas

Como já explicitado anteriormente, o número de unidades funcionais disponíveis na UFR pode ser parametrizado. Assim, para o conjunto de aplicações acima foram

utilizados cinco modelos diferentes de UFR, estes seguem a forma da Figura 3.3. O número de unidades funcionais de cada modelo é demonstrado na Tabela 3.3. Ressaltando que o Modelo 4 é composto pelo menor número de unidades funcionais necessárias para alcançar o potencial máximo, em desempenho, da técnica proposta sem a utilização de especulação de saltos. Analogamente, acontece com Modelo 5 utilizando a técnica de especulação de saltos.

O modelo de memória utilizado na extração de consumo de potência e energia é descrito em (PUTTASWAMY, 2002). Implementado em 0.25 um, este modelo possui 0.5 MB de capacidade, suficiente para executar todas as aplicações em questão. O circuito da memória trabalha em 200 MHz e é alimentado em 5 Volts.

Tabela 3.3: Modelos de UFR

	Modelo 1	Modelo 2	Modelo 3	Modelo 4	Modelo 5
#Total Linhas	24	48	150	189	255
#Total Colunas	11	16	20	203	185
#ULA/Linha	8	8	12	55	50
#Mul/Linha	1	2	2	14	15
#Load/Linha	2	6	6	134	120

Nas seções seguintes será apresentado o impacto da inserção do sistema reconfigurável no desempenho, potência, energia e área do processador. Estes resultados foram obtidos através da execução das aplicações citadas anteriormente, dentro de cada seção serão demonstrados os respectivos resultados na presença e ausência da técnica de especulação. A profundidade de especulação utilizada é de dois saltos, estruturada através do algoritmo de escolha bimodal. (SMITH, 1981)

Para melhor visualização dos gráficos e tabelas desta seção foram selecionadas cinco aplicações do conjunto MiBench para demonstrar os resultados: *pcm*, *dijkstra*, *smoothing*, *CRC* e *rijndaeld*. Entretanto, todos os dados de todas as aplicações podem ser encontrados no Apêndice A. A escolha das cinco aplicações foi baseada no comportamento das mesmas, como pode ser observado na Figura 3.7, estas possuem comportamentos heterogêneos. *Pcm* e *dijkstra* são orientadas a controle, *rijndaeld* é orientado a dados, enquanto *smoothing* e *CRC* são aplicações mistas. Assim, há uma cobertura de todos os comportamentos nos resultados demonstrados nesta seção.

3.6.2 Desempenho

3.6.2.1 Sem especulação

A inserção do sistema reconfigurável no processador MIPS R3000 demonstra acelerações significativas no desempenho das aplicações. A Figura 3.8 ilustra esta aceleração provida pela inserção dos diferentes modelos de UFR no processador em questão. Para este experimento foram disponibilizados 512 blocos na cache de reconfigurações. Importante ressaltar que, sem o uso da técnica de especulação, os limites das configurações passam a ser regrados pelas instruções de salto.

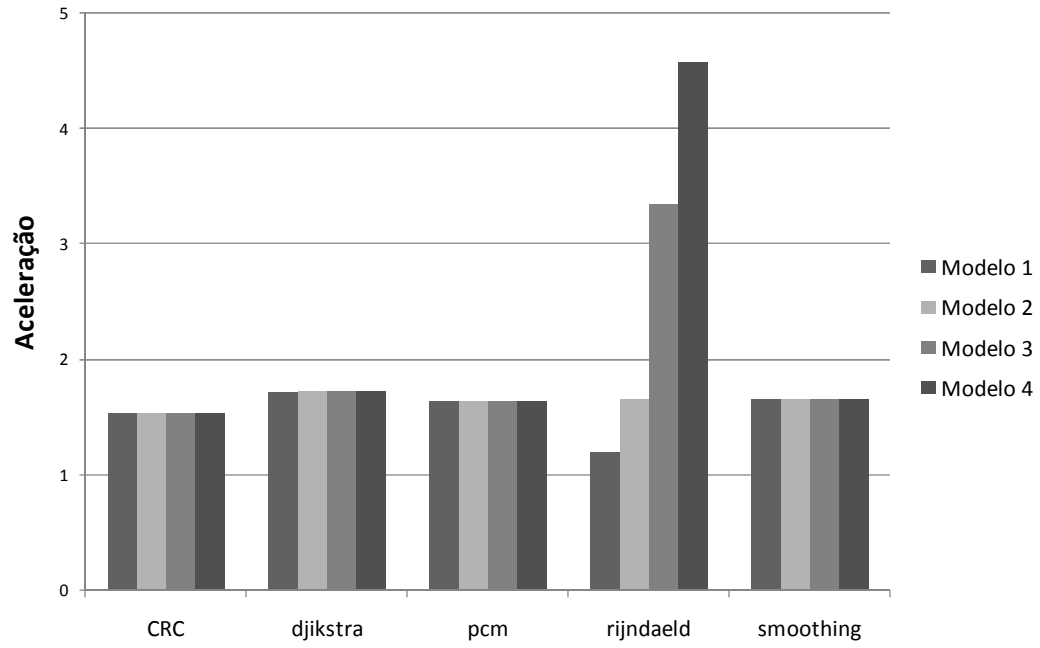

Figura 3.8: Aceleração no desempenho da arquitetura em diferentes modelos de UFR

O fator de aceleração alcançado pelo Modelo 1 foi, na média total das aplicações, de 1.68. Sendo este fator elevado para 2.05 no Modelo 3, próximo ao alcançado pelo Modelo 4 que apresentou 2.15 em média. Com ajuda da Tabela 5.1 do Apêndice A pode-se observar que, na maioria das aplicações, o número de unidades funcionais disponibilizadas no Modelo 2 já é o suficiente para explorar o máximo paralelismo existente nas aplicações. Somente os algoritmos mais orientados a dados, como o *rijndaeld* obtêm maiores acelerações de desempenho no Modelo 3 e 4. Como pode ser observado na Figura 3.8 a aceleração desta aplicação passou de 1.6 para 3.3 do Modelo 2 para o Modelo 3, elevando-se para 4.6 no Modelo 4. Este fato mostra a heterogeneidade do grau paralelismo, em nível de instruções, existente nas aplicações executadas.

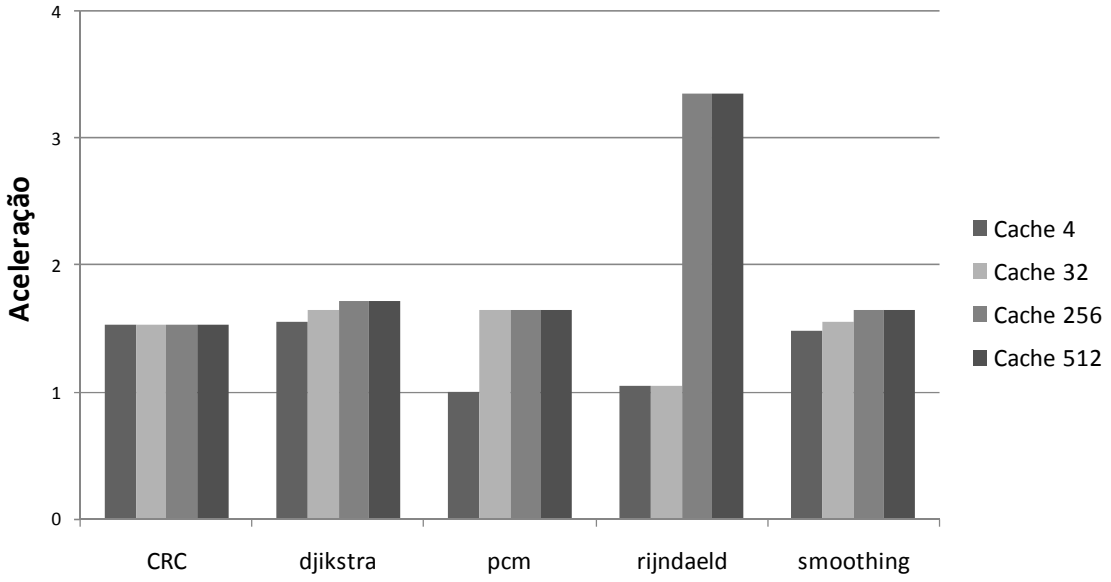

Figura 3.9: Influência do tamanho da cache de reconfigurações na aceleração

O número de blocos disponíveis na cache de reconfigurações tem um grande impacto na aceleração final da aplicação. A Figura 3.9 ilustra a aceleração das aplicações em diferentes tamanhos de cache para o Modelo 3, sendo proporcional para os outros modelos de UFR. Como pode ser observada nesta Figura, a aceleração no desempenho de algumas aplicações é bastante comprometida pela utilização de somente quatro blocos na cache, com este número de blocos disponível a aplicação *pcm* não demonstra nenhuma aceleração. A utilização de poucos blocos, para armazenar as configurações realizadas pelo TB, ocasiona um grande número de faltas na cache, forçando o sistema a executar as instruções no processador ao invés de acelerá-las na UFR. Entretanto, para quase todas as aplicações 256 blocos na cache é o suficiente para alcançar a aceleração máxima do modelo de UFR. Como pode ser observado na Tabela 5.1, a aplicação *gsmd* é a única que necessita de 512 blocos para alcançar a aceleração máxima neste modelo, o fato desta ter em seu código um elevado número de configurações que podem ser executados na UFR, agregado ao reuso da maior parte das configurações esclarece a necessidade de um número maior de blocos na cache.

3.6.2.2 Com especulação

A Figura 3.10 demonstra a aceleração obtida na execução em todas as aplicações, utilizando a técnica de especulação proposta em (BECK, 2006b). Observa-se que a inserção da técnica elevou ainda mais o desempenho das aplicações, comparado aos resultados demonstrados sem especulação (Figura 3.8). Esta técnica explora a reconfigurabilidade além das instruções de salto, ou seja, as configurações não são interrompidas quando este tipo de instrução é encontrado.

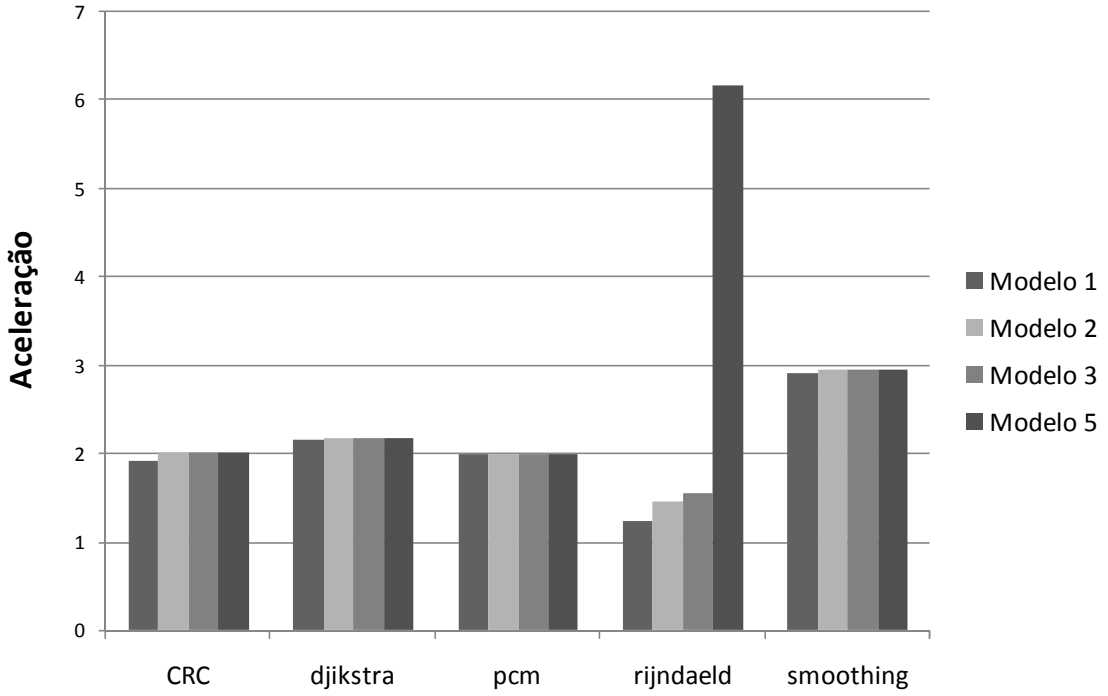

Figura 3.10: Aceleração no desempenho em diferentes modelos de UFR utilizando a técnica de especulação

A média de aceleração no desempenho provida pelo sistema sem a utilização de especulação no Modelo 1 foi de 1.68 sendo elevada para 2.03 no uso da técnica. Esta última aceleração ultrapassa a média provida pelo Modelo 3 sem o uso da técnica, ou

seja, conseguem-se fatores de aceleração maiores com um menor número de unidades funcionais disponíveis na UFR. A aceleração apresentada pela abordagem no Modelo 3 foi de 2.63, um grande aumento em relação a aceleração de 2.05, fornecido pelo mesmo modelo sem o uso da técnica. O Modelo 5, relativo ao máximo potencial alcançado pelo sistema reconfigurável utilizando a técnica de especulação, demonstrou em todas as aplicações uma média de aceleração de 3.01. Assim, o potencial máximo de aceleração do sistema reconfigurável elevou-se em 40% com a inserção da técnica de especulação.

A utilização de especulação eleva o número de faltas na cache de reconfiguração. Quando o hardware de TB detecta um erro na especulação de um salto, a configuração correspondente é eliminada da cache de reconfigurações. Assim, uma nova configuração é realizada pelo TB e inserida na cache. Por este motivo, quando a especulação é utilizada, o número total de configurações realizadas pelo TB é maior, comparado a ausência da técnica. Para a execução da aplicação *jpege*, sem a utilização de especulação são criadas 1317 configurações, este número cresce para 27321 com o uso da técnica. Entretanto, como pode ser observado na Figura 3.11 e na Tabela 5.11, o algoritmo de substituição FIFO gerencia de maneira eficiente o grande número de configurações realizadas, o que reflete em um alto nível de aceleração das aplicações no Modelo 3. Para a aplicação *jpege*, 32 posições são necessárias para se obter a máxima aceleração neste modelo. Em todas as aplicações, exceto na *edges*, os mesmos 256 blocos utilizados na ausência de especulação foram suficientes para alcançar a aceleração máxima deste modelo.

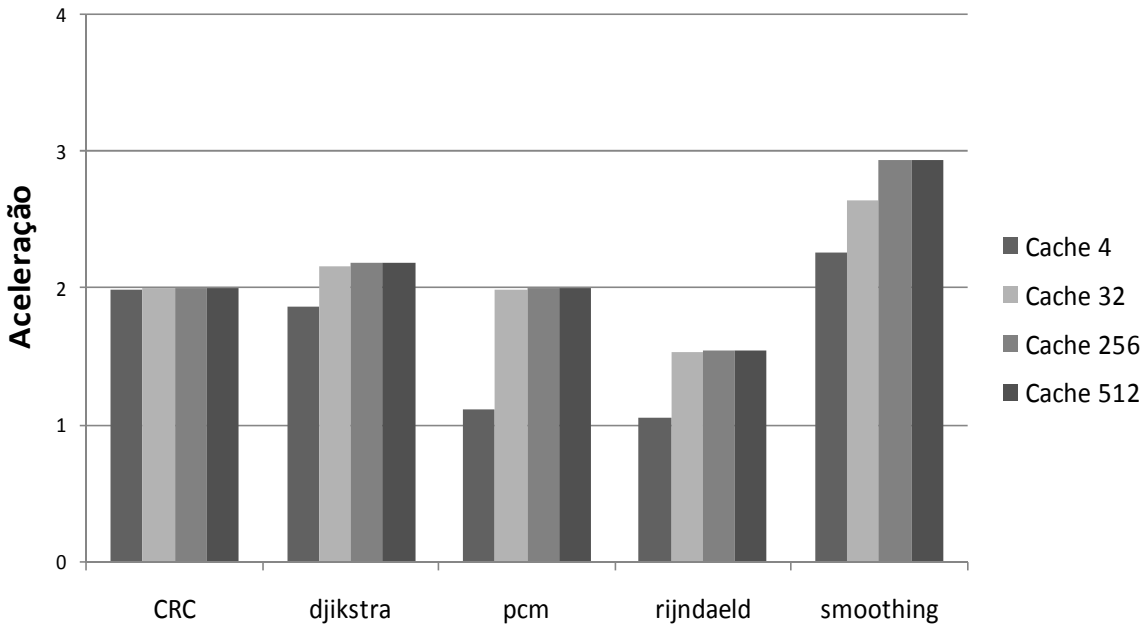

Figura 3.11: Influência do tamanho da cache de reconfigurações na aceleração utilizando a técnica de especulação

3.6.3 Potência

O consumo de potência é uma restrição imposta pelos dispositivos embarcados, visto que, a maioria destes é alimentada por bateria. Nesta seção serão apresentados os

valores de potência consumida pelo sistema reconfigurável acoplado ao processador MIPS R3000.

3.6.3.1 Sem especulação

A Figura 3.12 ilustra o consumo de potência das aplicações, em diversas configurações de cache, no modelo Modelo 2 sem a utilização da técnica de especulação. Nesta mesma Figura, é demonstrado o consumo de potência do processador MIPS desprovido do sistema reconfigurável. Observa-se que o consumo do sistema reconfigurável eleva-se na mesma proporção em que um maior número de blocos disponíveis na cache é disponibilizado. A menor taxa de faltas de configurações, em caches que provêm um maior número de blocos, acarreta uma maior freqüência de ativação da UFR. A elevação da complexidade do hardware da cache em relação à disponibilidade de blocos disponíveis é outro fator que explicita o aumento proporcional do consumo de potência.

Ao comparar a Figura 3.13 e Figura 3.12 é observado que, devido ao aumento do número de unidades funcionais disponíveis na UFR do Modelo 2 para o Modelo 4, o consumo de potência eleva-se em grandes proporções. A média deste consumo para todas as aplicações e tamanhos de cache de reconfiguração no Modelo 2 é de 11.4 Watts. Entretanto, esta média para o Modelo 4 eleva-se para 61.92 Watts. Sendo a potência média do processador MIPS 1.63 Watts, a utilização do Modelo 4 ocasiona um aumento de 38 vezes no consumo de potência do sistema.

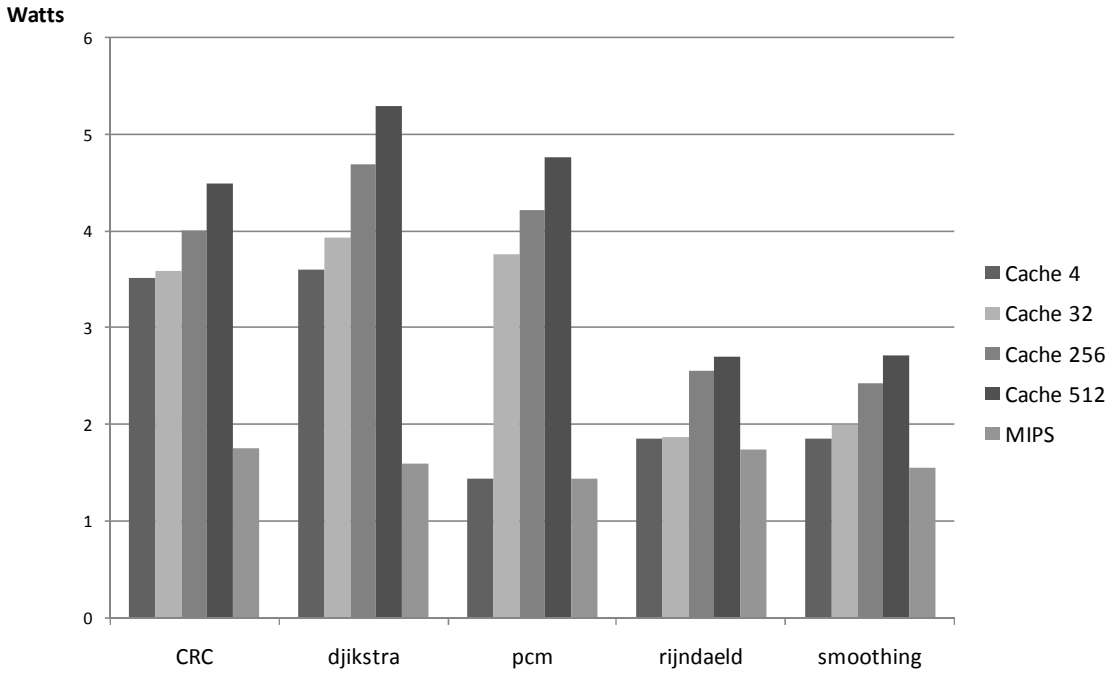

Figura 3.12: Consumo de potência das aplicações do Modelo 2 e do processador MIPS

A preocupação advinda deste grande impacto no consumo de potência estimulou a exploração de técnicas de redução. Como contribuições deste trabalho, no próximo capítulo serão demonstradas técnicas aplicadas ao sistema que impactam diretamente na redução deste consumo e de área ocupada pelo sistema reconfigurável.

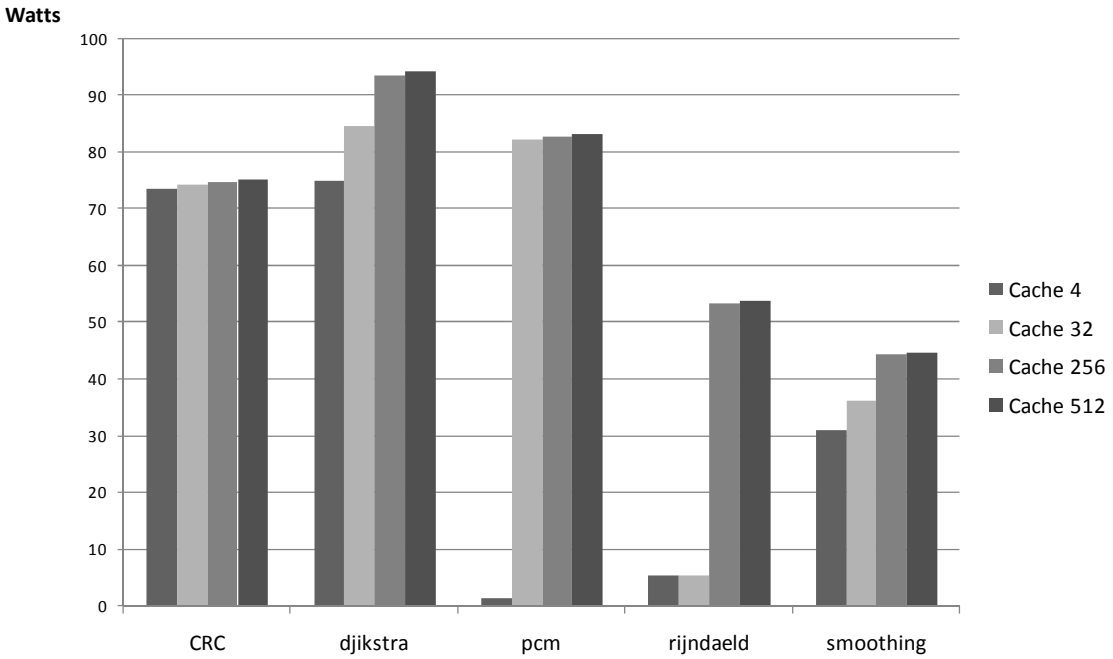

Figura 3.13: Consumo de potência das aplicações no Modelo 4

3.6.3.2 Com especulação

Na Figura 3.14 é demonstrado o consumo de potência de todas as aplicações no Modelo 2, utilizando a técnica de especulação. Este consumo é reduzido em relação à execução deste modelo sem a utilização da técnica. A inserção da técnica impacta na redução de acessos a memória cache, desta forma, a potência média consumida é reduzida. Um exemplo é a execução da aplicação *jpege*, sem especulação são realizados 5.766.867 acessos à cache, a inserção da técnica reduz este valor para 2.973.009.

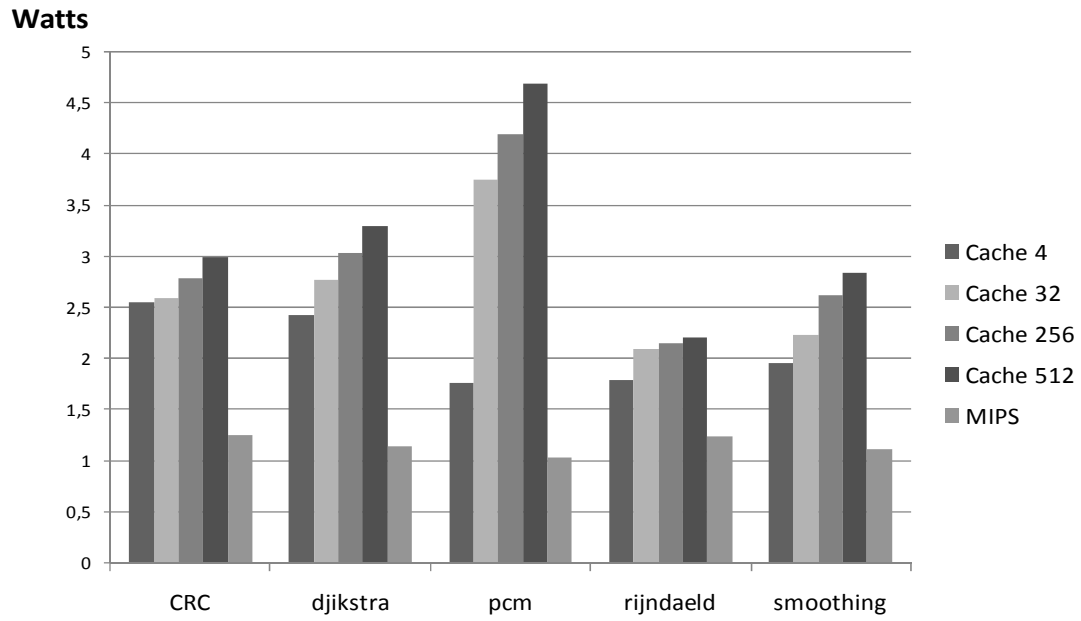

Figura 3.14: Consumo de potência das aplicações do Modelo 2, utilizando especulação, e do processador MIPS

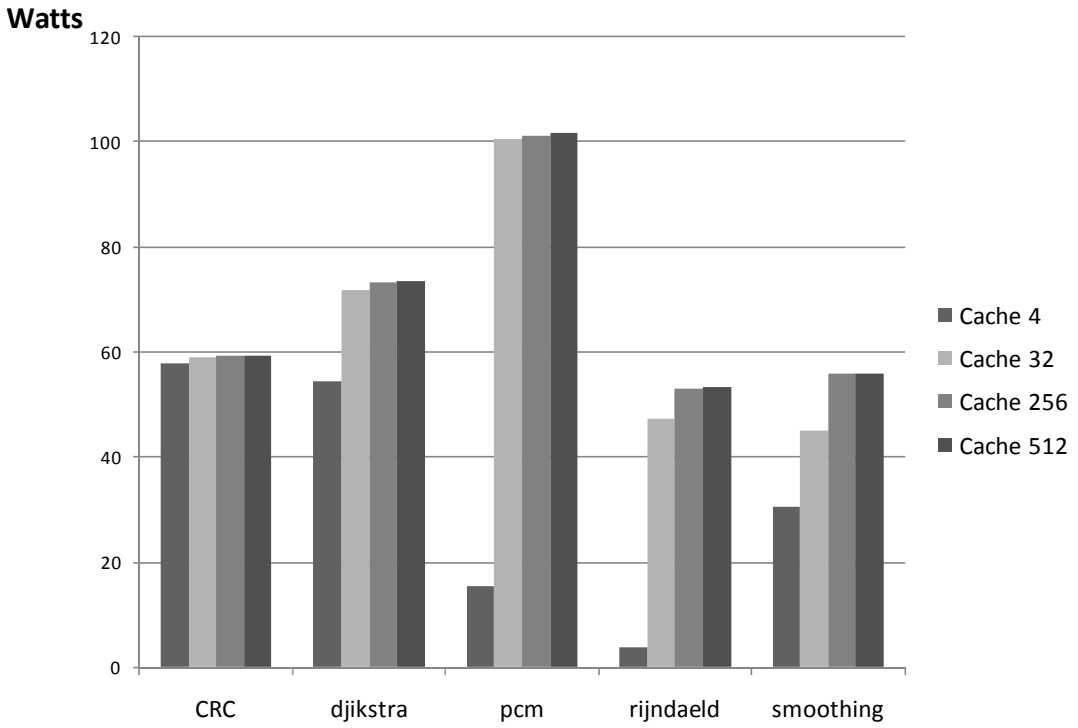

Figura 3.15: Consumo de potência das aplicações no modelo Modelo 5 com a utilização de especulação

O mesmo fato acontece com o Modelo 5, a Figura 3.15 ilustra a potência consumida por este modelo utilizando a técnica de especulação. Comparando esta com a Figura 3.13, nota-se em algumas aplicações uma redução no consumo de potência. Apesar do Modelo 5 conter mais unidades funcionais que o Modelo 4, a potência média consumida por este último modelo é maior, visto que a utilização de especulação no Modelo 5 reflete em menos ativações da UFR nestas aplicações.

Assim, conclui-se que além de prover aceleração ao sistema reconfigurável, a técnica de especulação impacta, em algumas aplicações, diretamente na redução do consumo de potência. Entretanto, mesmo com a redução advinda desta técnica, o consumo de potência é um fator restritivo de implantação desta abordagem em um dispositivo embarcado.

3.6.4 Energia

3.6.4.1 Sem especulação

De forma análoga ao consumo de potência, a inserção do sistema reconfigurável na plataforma MIPS causa um aumento no consumo de energia. A Figura 3.16 ilustra o consumo de energia na execução das aplicações no Modelo 2 com a disponibilidade de 512 blocos na cache de reconfigurações. Além disto, esta Figura apresenta o consumo de energia das aplicações executadas somente no processador MIPS R3000. Observa-se que a inserção do sistema reconfigurável eleva a energia consumida. Em média, o consumo de energia na execução das aplicações no Modelo 2 é de 258 mjoules. Sendo o consumo médio do processador 193 mjoules, a inserção da abordagem reconfigurável acarreta um aumento de 1.34 vezes o consumo de energia do sistema.

É importante salientar o dado demonstrado pela aplicação *rijndaeld* na Figura 3.16. A disponibilidade de um maior número de blocos na cache acarretou um drástico aumento na aceleração desta aplicação (Figura 3.9). Desta forma, o tempo de computação foi reduzido na mesma proporção e a energia resultante foi menor que a energia consumida pela execução da aplicação no processador MIPS R3000.

A mesma análise anterior é feita para o Modelo 4 na Figura 3.17, com o aumento do número de unidades funcionais disponíveis na UFR a energia cresce significativamente em relação ao Modelo 2. Entretanto, comparado com o consumo de potência deste modelo, demonstrado na Figura 3.13, o consumo de energia não é conduzido na mesma proporção. Na subseção anterior concluiu-se que a potência média consumida na execução das aplicações utilizando o Modelo 4 foi 38 vezes maior que potência do processador. Entretanto, como pode ser observado na Figura 3.17, em média o consumo de energia deste mesmo modelo é de 4.38 Joules, sendo o consumo médio do processador 193 mjoules, o crescimento de energia é somente de 22.7 vezes. A aceleração da aplicação provida pelo sistema reconfigurável reduz o tempo de execução da aplicação, o que impacta diretamente no declínio do fator energia em relação ao fator potência.

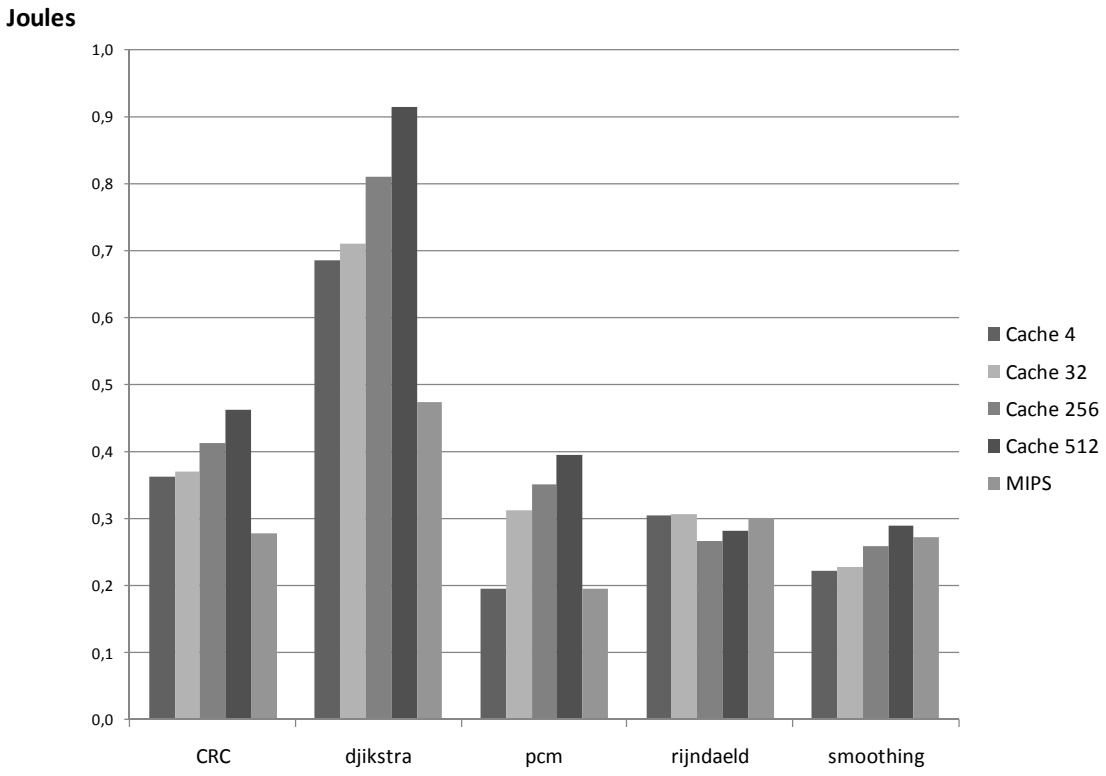

Figura 3.16: Energia consumida pela execução das aplicações utilizando o Modelo 2

Contudo, o elevado consumo de energia demonstrado não é viável ao domínio embarcado. Se o sistema alvo desta arquitetura dependesse de bateria para sua alimentação, a duração desta ficaria comprometida, seu tempo seria reduzido em 22.7 vezes do tempo original. Como já citado anteriormente, técnicas de redução desta característica serão demonstradas no próximo capítulo.

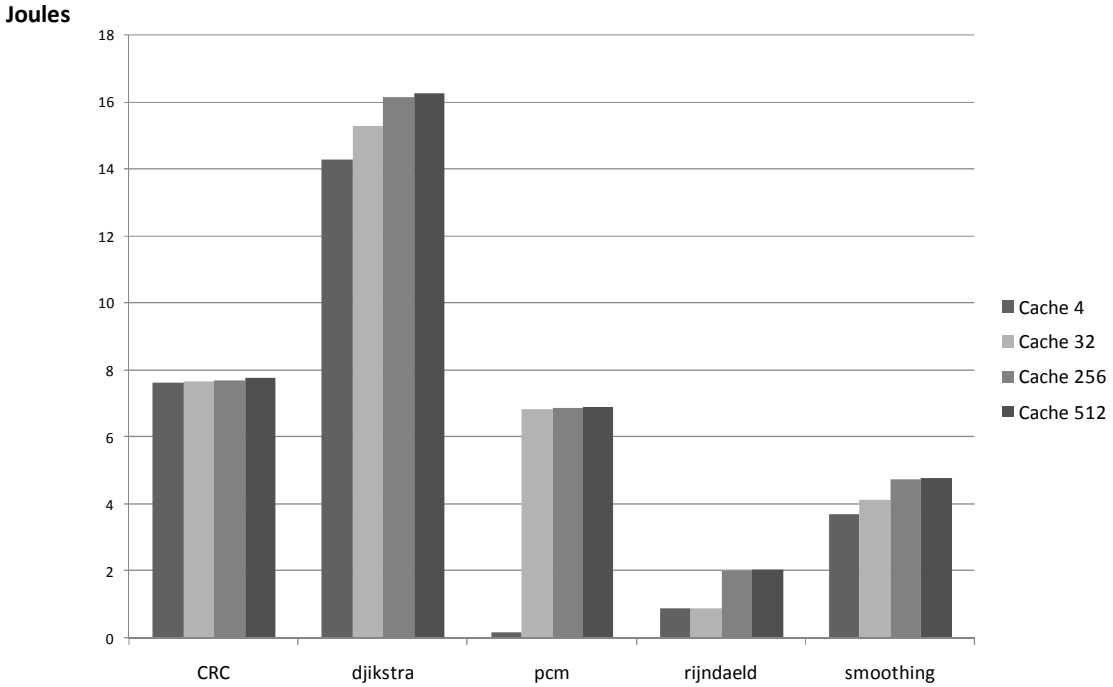

Figura 3.17: Energia consumida pela execução das aplicações utilizando o Modelo 4

3.6.4.2 Com especulação

A utilização da técnica de especulação impacta diretamente na redução do consumo de energia do sistema reconfigurável. A Figura 3.18 ilustra a energia consumida pelo sistema na utilização do Modelo 2. Comparando a energia consumida por este mesmo modelo sem a utilização de especulação, Figura 3.16, nota-se um declínio na energia consumida. Três fatores influenciam nesta redução: a maior aceleração no desempenho provida pela técnica de especulação (redução no tempo de computação); como demonstrado na seção anterior, a diminuição da potência consumida na inserção desta técnica; e, a diminuição no número de acessos a cache de reconfigurações com a inserção da técnica de especulação. A média de energia consumida por este modelo sem especulação foi de 258 mjoules, sendo reduzida para 179 mjoules com a inserção da técnica.

Para o Modelo 5, a Figura 3.19 ilustra a energia consumida na execução das aplicações com a utilização de especulação. Este modelo apresentou uma menor energia consumida em relação ao modelo que oferece o máximo potencial da abordagem sem a utilização de especulação (Modelo 4). Apesar do maior número de unidades funcionais, a média da energia consumida pelo Modelo 5 foi de 3.33 Joules, uma redução de 24% da energia total do sistema em relação ao Modelo 4 sem o uso de especulação.

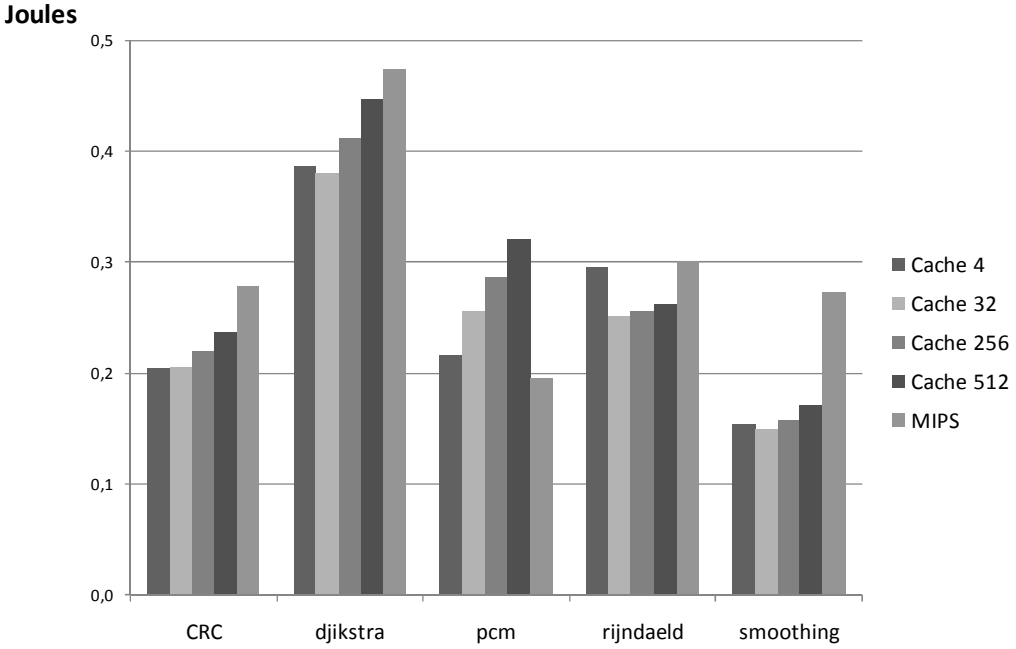

Figura 3.18: Energia consumida pela execução das aplicações utilizando o Modelo 2 com a técnica de especulação

Entretanto, mesmo com a porcentagem de redução demonstrada com a inserção da técnica de especulação, a energia consumida pelo Modelo 5 é, em média, 17 vezes maior que a energia consumida pela execução das aplicações somente no processador. Assim, ainda a duração da bateria fica comprometida pela grandeza do aumento da energia consumida.

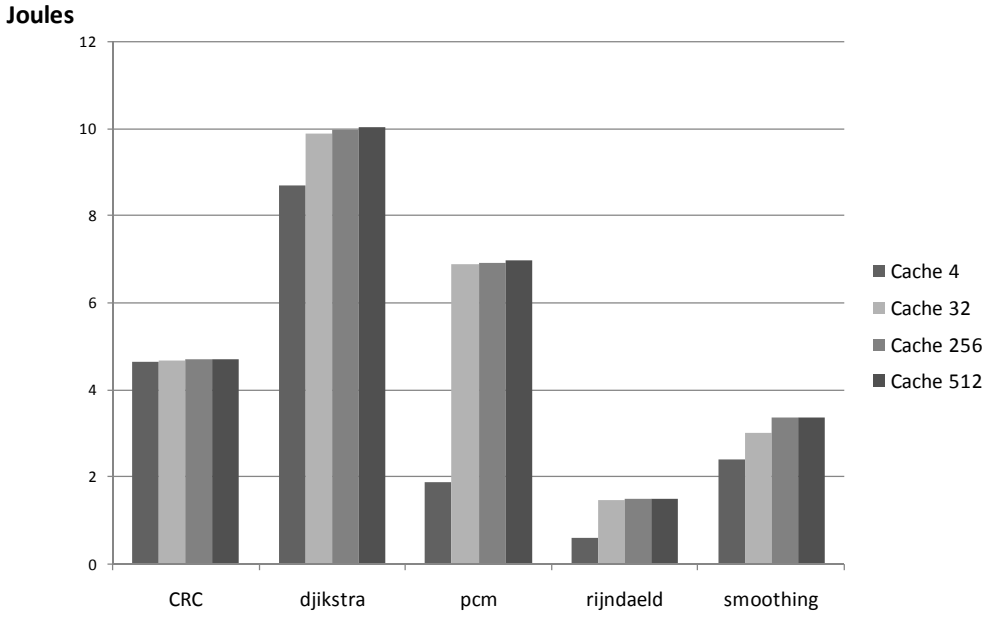

Figura 3.19: Energia consumida pela execução das aplicações utilizando o Modelo 5 utilizando a técnica de especulação

3.6.5 Área

A inserção do sistema reconfigurável no processador MIPS R3000 acarreta um aumento significativo na área. A Tabela 3.4 demonstra a área ocupada por cada componente, em portas lógicas, dos modelos propostos anteriormente. A área do processador MIPS R3000, extraída a partir da descrição VHDL é de 26.886 portas. Assim, o acréscimo de área na plataforma para o Modelo 1 é de 29.22 vezes em relação à área original do processador. Para o Modelo 3 este valor cresce para 251.97 vezes.

Tabela 3.4: Área ocupada, em portas lógicas, pelos componentes da UFR

Componente	Modelo 1	Modelo 2	Modelo 3	Modelo 4	Modelo 5
ALU	294.032	600.576	2.815.200	11.870760	19.941.000
Load	34.112	31.488	98.400	2.021.792	3.345.600
Multiplicador	20.067	214.048	668.900	4.307.716	8.528.475
Multiplexador	378.780	657.408	2.824.800	18.487.032	31.104.900
Demultiplexador	58.752	117.504	367.200	337.824	624.240
TOTAL (em portas)	*785.743*	*1.621.024*	*6.774.500*	*37.025.124*	*63.544.215*

A partir da descrição do hardware de tradução binária em VHDL foi possível verificar a sua área ocupada. A simplicidade do hardware é verificada nesta característica, sendo 1.204 portas a área ocupada pelo mesmo. Entretanto, como explicitado anteriormente, o tradutor binário é baseado também em tabelas e mapas, na Tabela 3.5 são encontrados os dados de área consumida, em bytes, destes componentes. É possível observar nesta Tabela que o custo de área das tabelas e mapas é considerado baixo e factível de ser implementado nas tecnologias atuais de memória.

Tabela 3.5: Área, em bytes, das tabelas e mapas

Tabela/ Mapa	Modelo 1	Modelo 2	Modelo 3	Modelo 4	Modelo 5
Escrita (M)	32	32	32	32	32
Recurso (M)	125	268	1.212	8.717	10.678
Leitura (T)	216	512	2.200	19.719	24.225
Escrita (T)	72	144	450	567	765
Contexto Inicial (T)	5	5	5	5	5
Contexto Atual (T)	5	5	5	5	5
Imediatos (T)	16	16	16	16	16
TOTAL (em Bytes)	*439*	*950*	*3.888*	*29.029*	*35.694*

A Tabela 3.6 contém os dados de área ocupada pela cache de reconfiguração nos diversos modelos de UFR propostos. Diferentemente das tabelas que compõe o tradutor binário, dependendo do número de blocos disponíveis na cache, não é interessante a possibilidade de implementação da cache na tecnologia atual. A redução do número de blocos seria uma solução viável. Entretanto, como demonstrado nas Figura 3.9 e Figura 3.11, esta solução ocasionaria uma diminuição na aceleração máxima alcançada pelo sistema reconfigurável. Assim, será demonstrado no próximo capítulo técnicas que reduzem o número de unidades funcionais necessárias dentro da UFR e impactam diretamente no tamanho da cache de reconfigurações.

Tabela 3.6: Área, em bytes, da cache de reconfigurações

Blocos (em Bytes)	Modelo 1	Modelo 2	Modelo 3	Modelo 4	Modelo 5
2	910	1.932	7.809	58.091	71.420
4	1.756	3.800	15.554	116.118	142.776
8	3.637	7.868	32.320	240.954	296.231
16	7.024	15.200	62.216	464.474	571.106
32	14.264	30.912	126.632	948.667	1.166.437
64	28.096	60.800	248.864	1.857.896	2.284.424
128	56.264	121.744	498.178	3.716.359	4.569.613
256	112.384	243.200	995.456	7.431.584	9.137.696
512	224.773	486.405	1.990.917	14.863.173	18.275.397

4 MODELO DE EXPLORAÇÃO DE RECURSOS DE HARDWARE E POTÊNCIA CONSUMIDA

No capítulo anterior foram demonstrados ganhos significativos de desempenho de um processador MIPS R3000, derivados da inserção do sistema reconfigurável, na execução de um conjunto de aplicações com comportamento heterogêneo, evidenciando o potencial da abordagem proposta em (BECK, 2006a). Entretanto, como comprovado neste mesmo capítulo, a inserção do sistema reconfigurável acarretou uma grande elevação na área, potência e energia consumida da plataforma. Como já citado anteriormente, os dispositivos embarcados mantém rígidas restrições sobre estas características.

Os fatores citados anteriormente motivaram a exploração do sistema reconfigurável, focando na redução do consumo de área, potência e energia. Entretanto, o sistema é dotado de vários componentes, todos passíveis de otimização. A verificação do componente que causa maior impacto nestas características foi alvo de primeiro estudo. A Figura 4.1 demonstra o consumo de potência, em percentagem, de cada componente da abordagem, nos modelos utilizados no capítulo anterior.

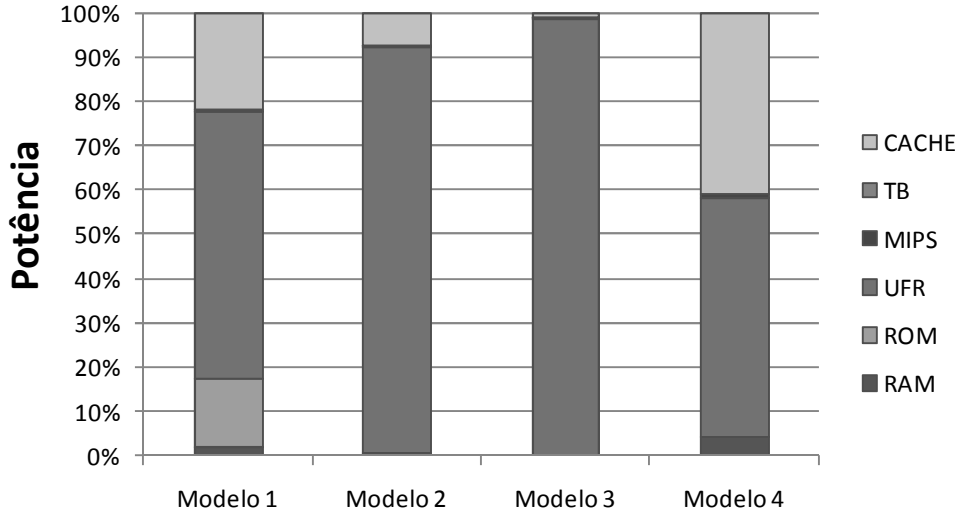

Figura 4.1: Consumo relativo de potência nos componentes do sistema reconfigurável

Nota-se nesta Figura que a unidade funcional reconfigurável é responsável por 89% do consumo total de potência, em média, em todos os modelos. Desta forma, conclui-se

que o componente alvo de otimização deve ser a UFR. O consumo de potência estática está se tornando cada vez mais significativo nas novas tecnologias CMOS, este consumo é diretamente proporcional a área ocupada pelo circuito. Deste modo, primeiramente o foco de exploração deve ser a área da UFR.

A otimização da área consumida pela arquitetura causa um impacto direto no consumo de potência. Entretanto, ainda é possível reduzir o consumo de potência evitando que partes do circuito, que não estão realizando computação em um dado instante, recebam alimentação. Para verificar o potencial da utilização do desligamento de partes do sistema reconfigurável foi selecionada uma abordagem proposta em (TSCHANZ, 2003). Esta técnica utiliza um transistor que realiza o corte de alimentação de partes do circuito, evitando o consumo de potência estática.

Nas próximas seções deste capítulo serão apresentadas a ferramenta de exploração de recursos de hardware e a técnica de desligamento, assim como os resultados obtidos através da implantação das mesmas nas UFR propostas no capítulo anterior.

4.1 ARISE

Atualmente, o projeto de um processador é realizado sem a verificação da quantidade de recursos necessários para acelerar uma dada aplicação. Ainda, são adicionados cada vez mais recursos que não fornecem a esperada contribuição na aceleração destas aplicações. Conseqüentemente, o processador se torna mais complexo elevando o consumo de potência, área e energia.

O cenário anterior ilustra a necessidade de modificar o fluxo de projeto de um processador. Primeiramente, é obrigatório verificar os requisitos da aplicação para posteriormente modelar o hardware. Entretanto, para sistemas embarcados este novo fluxo é problemático, visto que causa a inserção de uma fase que pode comprometer o *time-to-market*.

A questão central é como verificar os requisitos de hardware de uma aplicação de forma eficiente e rápida. Uma abordagem é retirar ao máximo o grau de intervenção humana e utilizar ferramentas que, de forma automática, identifique e construa uma arquitetura otimizada em recursos para a execução da aplicação.

Assim, como contribuição para este trabalho foi desenvolvida a ferramenta ARISE (do inglês *Automatic Resources Investigation System based on application Execution*) que, de forma automática, explora a execução de uma aplicação e fornece o exato número de recursos necessários para acelerá-la, diminuindo a área ocupada e a potência consumida do sistema reconfigurável alvo. (RUTZIG, 2008)

O desenvolvimento desta ferramenta teve como principal objetivo explorar, de forma eficiente, o paralelismo disponível na aplicação e, conseqüentemente, medir os recursos mínimos de hardware necessários para tornar a sua execução orientada a dados (*dataflow*). Como pode ser observado na Tabela 3.4, as unidades funcionais, em média, ocupam 46% da área total da UFR, sendo o restante da área ocupada pelos multiplexadores e demultiplexadores. Apesar da menor área, o gerenciamento de recursos realizado pela ferramenta ocasionará grande impacto na área da UFR, visto que, a diminuição do número destas impacta diretamente na área ocupada pelos outros componentes. A interconexão também terá um declínio na área, assim como a cache de reconfigurações.

Em uma versão preliminar desta ferramenta, em (RUTZIG, 2008) foi demonstrado que a exploração de recursos do sistema reconfigurável, sem a utilização de especulação, no simulador Simplescalar executando o conjunto de aplicações MiBench alcançou uma redução de um fator cinco na área ocupada pela UFR com somente 5.8% de perda de desempenho em média.

O trabalho anterior demonstra a principal contribuição da ferramenta ARISE, a agregação de duas técnicas: tradução binária e exploração de recursos levando em conta a execução da aplicação. O resultado desta agregação é a perfeita exploração do paralelismo das aplicações com a mínima área ocupada pela arquitetura e, o mais importante, refletindo em perdas insignificantes de desempenho devido a adaptabilidade do mecanismo de tradução binária na execução pós otimização da UFR.

Para esta dissertação a ferramenta de exploração foi desenvolvida na linguagem C++ e acoplada à descrição ArchC do sistema reconfigurável. Toda a sua descrição ocupou 1345 linhas de código C++, foram modelas 3 classes contendo 12 métodos. A implementação foi realizada de forma modular, possibilitando a migração da mesma para outras plataformas, ou seja, a ferramenta não é dependente do processador utilizado como estudo de caso.

4.1.1 Funcionamento

Basicamente, a ferramenta ARISE realiza a exploração de recursos em um grão grosso, ou seja, toda a sua investigação na concepção de uma UFR é realizada através de grafos de dependência de dados de instruções (GDD). Estes grafos são formados através das configurações realizadas pelo tradutor binário. Assim, para cada configuração construída pelo TB existirá o seu GDD correspondente. A partir de cada GDD, são extraídas várias informações relevantes para explorar perfeitamente o paralelismo e reduzir o número de recursos disponíveis na UFR. Finalmente, as informações de todos os GDD são correlacionadas. Com os dados produzidos desta computação a ferramenta concebe uma UFR otimizada para o sistema reconfigurável.

Como já explicitado anteriormente, a principal diferença desta técnica de exploração de recursos, em relação a todas as outras propostas, é que a mesma constrói os GDDs a partir da execução de uma aplicação, refinando ainda mais a otimização da arquitetura. O fluxo de execução da ferramenta é ilustrado na Figura 4.2, nesta são demonstrados os seis passos necessários para construir e explorar os GDDs. A tarefa de cada passo é descrita a seguir:

- Passo 1 – o início do fluxo da ferramenta está ligado ao mecanismo de tradução binária, ou seja, não pertence à exploração da UFR. Entretanto, a ferramenta depende deste para receber de entrada uma configuração da UFR. No momento em que o TB conclui uma configuração, as instruções que compõe a mesma são repassadas para a ferramenta de exploração.
- Passo 2 – de posse das instruções da configuração corrente a ferramenta constrói o seu grafo de dependência de dados (GDD).
- Passo 3 – após o término da construção do GDD, é realizado o armazenamento deste em um banco de dados. Para que, após o fim da execução da aplicação as informações de todos os GDDs sejam correlacionadas.

- Passo 4 – este passo corresponde a repetição dos três passos anteriores, ou seja, para cada configuração concluída é construído seu GDD, sendo este armazenado no banco de dados. Somente esta etapa e as anteriores se mantém ativas durante a execução da aplicação.
- Passo 5 – a ativação deste passo é realizada no momento do término da execução da aplicação. Neste instante, todos os GDDs pertencentes à aplicação estão armazenados no banco de dados. Desta forma, o módulo explorador extrai de cada GDD informações necessárias (altura, largura, fator de reusabilidade) para a construção da UFR otimizada.
- Passo 6 – após a extração das informações de todos os GDDs, estas são comparadas e, como resultado, a ferramenta produz uma arquitetura otimizada para a UFR. A principal função deste passo é realizar a intersecção de todas as informações dos GDDs.

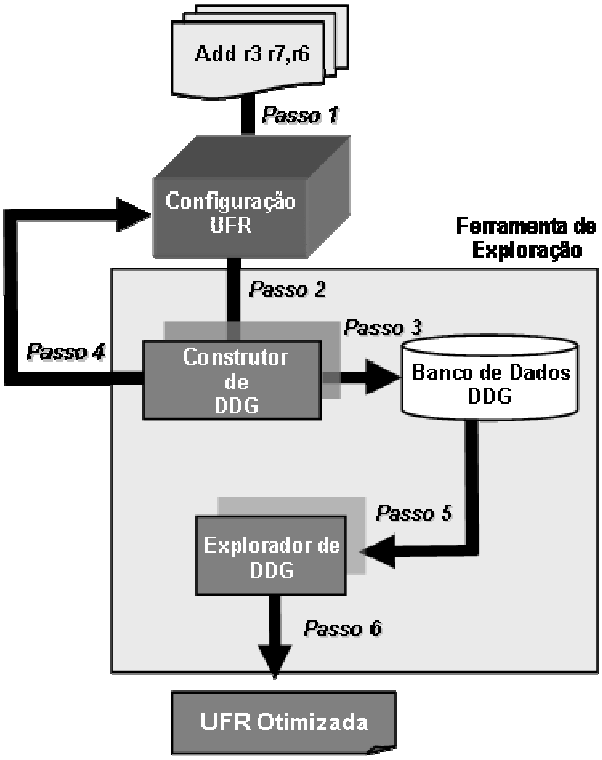

Figura 4.2: Fluxo da ferramenta de exploração

Na exploração dos GDD várias informações são extraídas para a concepção da nova UFR: altura e largura dos grafos; número e tipo de instruções contidas em cada grafo; tipo de dependência entre as instruções, etc. Entretanto, dois fatores necessitaram de uma maior atenção neste trabalho, pelo fato destes possuírem grande influência na caracterização da arquitetura. Estes são: paralelismo de instruções existentes nos grafos e o fator de reuso de cada GDD.

4.1.1.1 Paralelismo

A principal razão de executar partes de uma aplicação em uma unidade funcional reconfigurável é explorar o seu paralelismo, no estudo de caso deste trabalho, o paralelismo em nível de instruções. Como já citado anteriormente, para alcançar os níveis de aceleração demonstrados na Seção 3.6.2 um número elevado de unidades

funcionais devem ser alocadas horizontalmente na UFR. Em (BECK, 2006a) foi proposto um modelo retangular, ilustrado na Figura 3.3, desta unidade funcional reconfigurável. O desenho deste modelo pressupõe que em uma execução de uma aplicação há um mesmo nível de paralelismo em todas as linhas da UFR, pois existe o mesmo número de unidades funcionais disponíveis nestas.

Entretanto, como mostrado em (WALL, 1991) em uma aplicação existem diferentes níveis de paralelismo. Em (BECK, 2006a), o número exato de unidades funcionais que deve ser alocado em cada linha da UFR é desconhecido. Para refletir o máximo potencial do paralelismo disponível na aceleração da aplicação, com a mínima de área ocupada, a ferramenta ARISE provê uma relação entre o paralelismo existente na aplicação e a alocação das unidades funcionais em paralelo em cada linha da UFR. Desta forma, o número exato destas unidades será alocado em cada linha da arquitetura, reduzindo a demasiada área ocupada pela UFR retangular.

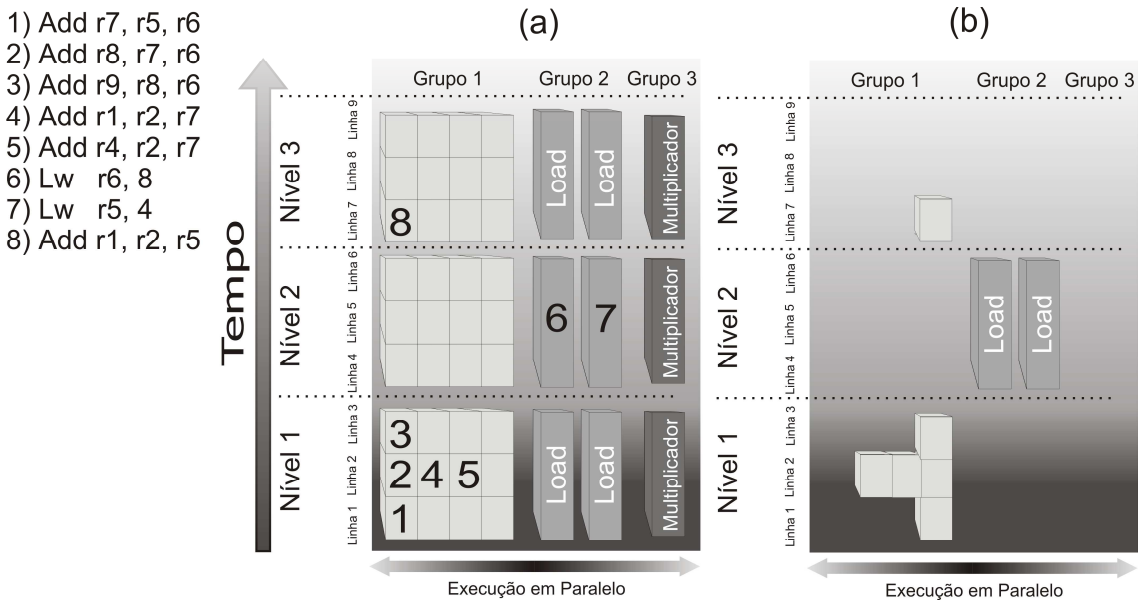

Figura 4.3: Exemplo de alocação de uma seqüência de instruções na UFR

Um exemplo de alocação de uma seqüência de instruções (configuração) executadas pela arquitetura reconfigurável é ilustrada na Figura 4.3(a). Pode ser observado nesta figura, que o nível de paralelismo é variável nas linhas da UFR. Devido ao fato da primeira instrução da seqüência possuir dependência de leitura após escrita (do inglês *read after write dependence*) com as instruções de número dois, quatro e cinco; o grau de paralelismo na primeira linha da UFR é reduzido. Desta forma, a maioria das unidades funcionais contidas nesta linha se mantém ociosas durante esta execução.

Outro exemplo importante a se observado na Figura 4.3 são as dependências que ocorrem entre as instruções de número três, seis, sete e oito. A cascata de dependências criada por estas instruções impede que ocorra a alocação de instruções nas unidades funcionais do Grupo 1 contidas no segundo nível da UFR. Este fato causa, além de uma grande ociosidade destas unidades, um desperdício de área ocupada que, conseqüentemente, eleva o consumo de potência do sistema. Para solucionar esta grande ociosidade, a ferramenta realiza uma realocação das unidades funcionais dentro na arquitetura da UFR, levando em conta a execução da aplicação, para que estas sejam úteis na exploração do paralelismo.

Assim, na Figura 4.3 (b) é demonstrado que, para o exemplo em questão, existe um potencial de redução de área. Nesta figura é ilustrada a perfeita alocação de recursos para este exemplo, a redução de área é diretamente relacionada ao paralelismo e a maneira como as instruções são alocadas dentro da arquitetura. O paralelismo pode ser calculado em tempo de projeto e este valor irá ter um forte impacto na área e no consumo de potência da arquitetura reconfigurável. Nesta ferramenta o paralelismo é calculado em tempo de projeto, porém levando em conta a execução da aplicação. Ainda, as futuras execuções desta aplicação na UFR da Figura 4.3(b) não apresentarão perda no desempenho, visto que, o mecanismo de tradução binária se adaptará a arquitetura resultante da ferramenta ARISE.

4.1.1.2 Fator de reusabilidade

Diferentemente de outras abordagens, que utilizam o compilador para otimizar a arquitetura, a abordagem proposta neste trabalho realiza esta tarefa dinamicamente, realizando a execução da aplicação em um simulador que descreve o comportamento do hardware. Portanto, a ferramenta consegue observar vários fatores de execução que impactam diretamente na otimização da UFR.

Um exemplo simples é que, em tempo de execução, é possível obter informações sobre o reuso de cada GDD, ou seja, o número de vezes que este foi executado. De posse destes dados, a ferramenta fornece valores a cada GDD proporcionais à sua taxa de reuso. Assim, GDDs com valores mais elevados terão maior influência na caracterização da UFR otimizada. Com a utilização de técnicas dinâmicas de decisão obtêm-se um maior grau de otimização para a UFR.

Em tempo de compilação não é possível detectar os GDD que não são executados. Assim, um GDD pode ter um grande impacto na caracterização da área da arquitetura, mas não ser executado na UFR. A dinamicidade da ferramenta proposta trata este problema descartando da caracterização da UFR os GDD que não são executados.

4.1.2 Exploração e Conclusões Preliminares

Para melhor explicitar o modo que a ferramenta realiza a concepção da UFR, será demonstrado através de um exemplo de uma configuração extraída da aplicação *dijkstra* disponibilizada pelo conjunto de aplicações MiBench. A Figura 4.4 ilustra a configuração encontrada pelo hardware TB sem a utilização da técnica de especulação. Devido a sua simplicidade e por cobrir os principais motivos que influenciam a caracterização da UFR esta configuração foi selecionada. A escolha desta configuração também foi fundamentada no seu grande fator de reusabilidade, o que diretamente impacta na aceleração do desempenho da aplicação. Além disto, com ajuda da própria ferramenta foi relatada que o seu GDD, ilustrado na Figura 4.5, possui uma estrutura muito significativa, sendo esta comum em todas as aplicações propostas na Seção 3.6.1.

```
lw      r2 (6)
andi    r2  2
srlv    r2  r2 r17
sw      r2 (3)
lw      r2 (28)
addiu   r5  0
lw      r3 (28)
lw      r4 (28)
sllv    r5  r5 r17
addiu   r2  2
sw      r2 (28)
lw      r2 (28)
addiu   r3  3
subu    r4  r4  r5
sw      r3 (28)
sw      r4 (28)
addu    r2  r5  r2
sw      r2 (28)
```

Figura 4.4: Seqüência de instruções do algoritmo de Dijkstra

Pode ser observado na Figura 4.5 que existe um grande grau de paralelismo a ser explorado neste grafo. Na primeira linha é possível executar cinco instruções em paralelo. Entretanto, o grau de paralelismo se reduz gradativamente no decorrer das linhas do grafo, nas últimas duas linhas somente existe a possibilidade de executar uma instrução.

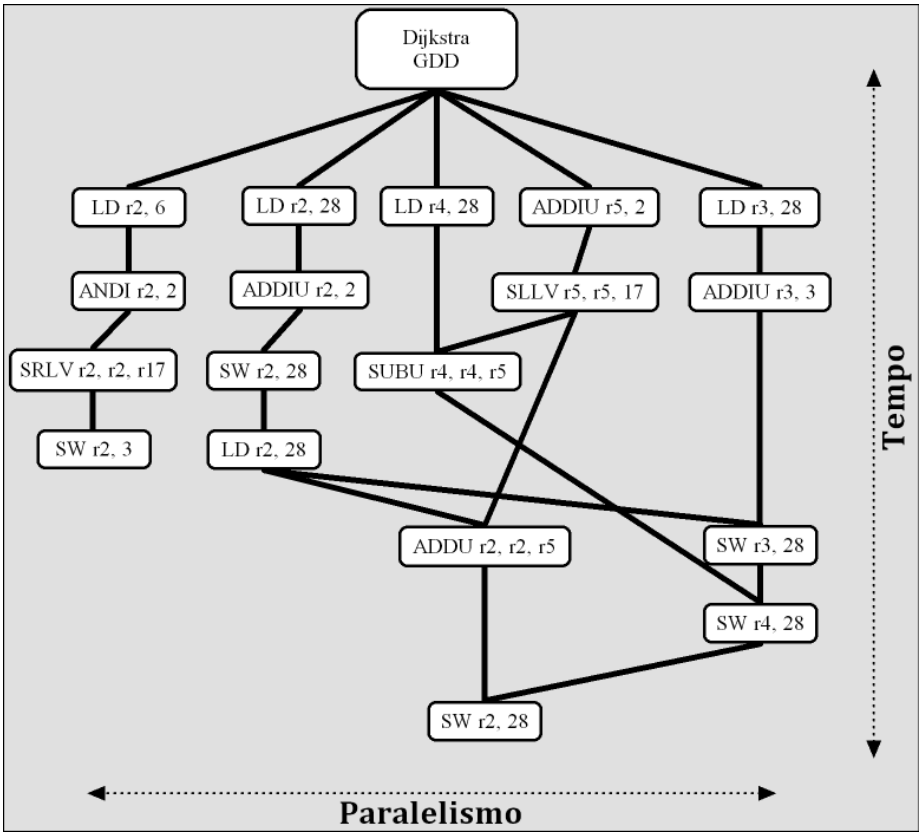

Figura 4.5: Grafo de dependência de dados entre as instruções da Figura 4.4

A computação tradicional de um processador RISC ocorre da seguinte forma: primeiramente os operando são carregados da memória para os registradores; após a

computação é realizada com os operandos previamente carregados; e finalmente, o resultado é gravado na memória. Uma inovação demonstrada na abordagem proposta por (BECK, 2006a), em relação a todas às outras propostas reconfiguráveis, é a utilização de unidades funcionais que acessam a memória na UFR. Com a ferramenta de exploração foi possível verificar o impacto da inserção deste tipo de unidade funcional no desempenho da arquitetura, desta forma explorou-se as aplicações demonstradas na Seção 3.6.1. Assim, concluiu-se que os GDD construídos se tornam extremamente pequenos sem a utilização de unidades que acessam a memória. Devido à natureza de execução RISC explicitada anteriormente, a reconfiguração fica limitada somente a computação do dado em si, diminuindo drasticamente a aceleração no desempenho das aplicações. Sem a utilização das unidades de acesso a memória na UFR a ferramenta construiu GDDs pequenos, o maior grafo obteve altura cinco. Entretanto, com a inserção destas unidades a altura do maior grafo foi vinte e quatro, sendo este pertencente à aplicação *rijndaeld*.

O modo de computar dos processadores RISC influi diretamente na construção da UFR otimizada. Como pode ser observado na Figura 4.5, existe um maior número de instruções do tipo *Load* alocadas na primeira linha deste grafo. Assim, as instruções que carregam os registradores com os dados para uma futura computação podem, na maioria dos casos, realizar a execução, de forma paralela, no início da UFR. Esta característica não é um fato isolado deste exemplo, foi verificado este evento na maioria dos grafos criados pela ferramenta. Devido a esta característica, a ferramenta ARISE, automaticamente, concentra um grande número de unidades de acesso a memória nas primeiras linhas da UFR.

Outro fato importante a ser explorado na Figura 4.5 é a concentração de instruções que realizam a computação em si nas linhas centrais do grafo. Conseqüentemente, a ferramenta identifica este comportamento e aloca um grande número de unidades funcionais que realizam a computação nestes locais da UFR. Finalmente, como regra a computação RISC, se verifica neste grafo que as instruções que gravam na memória os resultados das computações estão concentradas nas últimas linhas do grafo. Desta forma, um maior número de unidades funcionais será alocado, pela ferramenta, nestes espaços da UFR. Apesar de ocorrer uma concentração de unidades funcionais nas linhas centrais do grafo. Na maioria das linhas, este tipo de unidade funcional esta presente, existindo a necessidade de alocação destas unidades funcionais em todas as linhas da UFR.

O comportamento das dependências entre as instruções é um fator que influi na formação de uma UFR. A ferramenta ARISE explorou os GDDs das aplicações propostas na Seção 3.6.1 e verificou a existência da heterogeneidade desta característica. A aplicação *pcm* apresentou, na média de todas as execuções, 88% de dependências de dados entre instruções alocadas nas unidades funcionais do Grupo 1. Entretanto, somente 11% de dependências entre instruções alocadas no Grupo 2 e 1, e menos de 0,1% entre instruções alocadas no Grupo 1 e 2. O processamento desta aplicação é baseado na tradução de um sinal analógico para um sinal digital, seguido de uma compressão deste último. Portanto, a computação é muito intensa, a busca dos dados da memória é irrelevante em relação ao montante de computação que esta realiza. Devido a sua forma de computar, esta aplicação fornece uma grande porcentagem de dependências entre instruções alocadas no Grupo 1.

Em oposição ao *pcm*, os GDDs da aplicação *djikstra* apresentaram, em média, 32 % de dependências entre instruções alocadas no Grupo 1, 29% entre instruções alocadas

no Grupo 1 e 2 e 37% entre instruções alocadas no Grupo 2 e 1. O objetivo desta aplicação é encontrar, em um grafo, o menor caminho entre dois vértices. Na realização desta tarefa, a aplicação busca na memória, individualmente, cada vértice do grafo para realizar a computação, o que impacta no elevado número de dependências entre instruções alocadas no Grupo 2 e 1. Posterior à computação, o vértice computado é armazenado na memória, impactando no número de dependências entre o Grupo 1 e 2.

É importante ressaltar a heterogeneidade das aplicações quanto à proporção de tipos de dependências de dados existente. Assim, é demonstrado que, para otimizar a UFR, foi utilizado um conjunto de aplicações que envolvem diferentes comportamentos desta característica.

Entretanto, a ferramenta ARISE verificou que a média, em todas as aplicações, de dependências entre instruções alocadas no Grupo 1 foi de 55%; 17% das dependências ocorreram entre instruções alocadas no Grupo 1 e 2; 18% no Grupo 2 e 1; 5% entre instruções alocadas no Grupo 2; e os 3% restantes ocorreram entre instruções alocadas em multiplicadores com todas as outras unidades funcionais.

4.2 A técnica de *Sleep Transistor*

Sleep transistor foram propostos para atacar o consumo de potência estática. Basicamente, estes transistores são inseridos para controlar a alimentação de uma determinada parte do circuito. A utilização desta técnica é viável em porções do circuito que permanecem uma grande faixa de tempo sem realizar computações. Entretanto, alguns efeitos no desempenho do circuito podem ser introduzidos com o uso desta técnica, além de ocasionar um aumento da área total da arquitetura (TSCHANZ, 2003). A inserção de um bloco de controle é necessária para controlar o corte de fornecimento, de alimentação para o circuito.

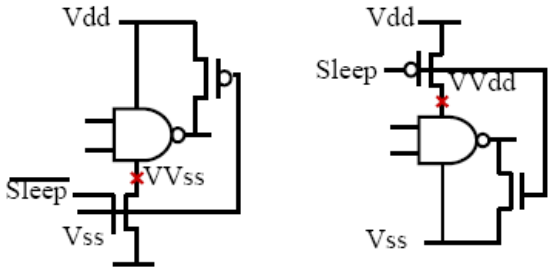

Figura 4.6: Tipos de implementação de *Sleep Transistor* (SHI, 2006)

A Figura 4.6 ilustra as possíveis implementações da técnica utilizando o *Sleep Transistor* para controlar a alimentação de uma porta NAND (SHI, 2006). À esquerda desta Figura é demonstrada a técnica empregando um transistor NMOS para realizar a controle da alimentação do circuito, à direita da mesma Figura um transistor PMOS é utilizado para realizar esta tarefa. Além do transistor de controle, outra chave deve ser inserida na saída da porta para evitar a sua flutuação. No exemplo da esquerda, quando a porta estiver em modo inativo, a saída se estabilizará em Vdd. Entretanto, no exemplo da direita, a inatividade da porta causará o valor resultante na saída igual à Vss. Desta forma, é responsabilidade do projetista verificar a viabilidade do valor da saída do bloco controlado.

4.2.1 Verificação do potencial da abordagem no sistema reconfigurável

A inserção do sistema reconfigurável, como foi demonstrada na Seção 3.6, causa um grande aumento na área e potência consumida. A exploração dos mínimos recursos necessários para acelerar uma aplicação é realizada pela ferramenta ARISE, proposta na Seção 4.1, de forma proporcional o consumo de potência também foi reduzido. Entretanto, este consumo após a exploração de área ainda continuou elevado.

O grande consumo de potência da UFR é conseqüência das características de modelagem da arquitetura proposta. Como já explicitado anteriormente, a arquitetura da UFR é formada por um circuito totalmente combinacional, uma execução realizada nesta, obrigatoriamente, acarreta o chaveamento de todas as suas unidades funcionais. Portanto, mesmo as unidades não alocadas por instruções computam dados irrelevantes e fornecem resultados desnecessários à execução.

Foram pesquisadas várias técnicas de redução de consumo de potência para unidade funcional reconfigurável. Inicialmente, a técnica de *Sleep Transistor* não se mostrou adequada para a abordagem em questão, pois seu foco principal é a redução do consumo de potência estática, sendo que a potência dinâmica é a que produz um acentuado impacto no consumo da UFR implementada na tecnologia 0.18um.

Entretanto, foi verificado que associação do tradutor binário à técnica de *Sleep Transistor* indiretamente interrompe tanto o consumo de potência estática como o consumo de potência dinâmica na abordagem reconfigurável proposta. O sistema reconfigurável, como explicitado anteriormente, possui três etapas em sua execução. Na primeira etapa, o hardware de TB realiza a tradução binária da seqüência de instruções e armazena a configuração da UFR na cache. A segunda etapa é relacionada à busca da configuração na cache e envio para a UFR para posterior execução, esta última ação constitui a terceira etapa.

Com a utilização do algoritmo de TB não é necessária a inserção do hardware que controla o desligamento dos blocos da UFR. Na primeira etapa (construção da configuração), o TB verifica os blocos funcionais que não possuem nenhuma instrução alocada e modifica o controle do *Sleep Transistor* deste bloco para “modo inativo”. Na segunda etapa (configuração da UFR), o transistor configurado para “modo inativo” interromperá a alimentação do respectivo bloco. Finalmente, no momento da execução não ocorrerá alteração no valor dos bits no bloco inativo. Além disso, por não estar recebendo alimentação, não irá consumir potência estática durante a execução.

A granularidade do bloco funcional controlado por um *Sleep Transistor* pode ser modificada de acordo com a necessidade do circuito. A verificação do potencial desta técnica, na unidade funcional reconfigurável, foi idealizada avaliando o desligamento a partir de dois grãos diferentes. Em primeira análise foi inserido um *Sleep Transistor* para controlar a alimentação de cada nível da UFR. Assim, o nível que não apresentar nenhuma instrução alocada será alterado para o “modo inativo”. A segunda análise realizada diminui o grão de controle para unidade funcional. Para cada unidade funcional existe um *Sleep Transistor* que realiza o controle de alimentação. Desta forma, as unidades funcionais que não foram designadas pelo TB para realizar computação serão configuradas para o “modo inativo”, ou seja, não receberão alimentação.

4.3 Metodologia

A ferramenta ARISE foi utilizada para conceber automaticamente uma arquitetura ideal, em relação ao número de unidades funcionais e o paralelismo disponível nas aplicações. Como citado anteriormente, a ferramenta caracteriza a UFR levando em conta a execução da aplicação. Assim, as mesmas aplicações e modelos de UFR, demonstradas na Seção 3.5, serão utilizadas para avaliar o impacto na redução da área ocupada provida pela ferramenta. No restante deste trabalho a arquitetura reconfigurável retangular proposta em (BECK, 2006a) será denominada de modelo não otimizado.

Na utilização da ferramenta ARISE foram empregadas duas abordagens distintas para explorar os recursos da UFR. A primeira abordagem, que chamaremos de "abordagem ótima", oferece o número exato de unidades funcionais e suas respectivas alocações na UFR, com o mesmo desempenho demonstrado com a UFR retangular (não otimizada) (Seção 3.6.2). A segunda abordagem, que chamaremos de "abordagem de baixo custo", fornece maior redução de área que a primeira. Entretanto, pode causar um impacto no desempenho das aplicações.

Na abordagem ótima, a ferramenta explora todos os GDD do conjunto de aplicações, verificando o grau de paralelismo disponível em cada linha da UFR. Conseqüentemente, constrói uma UFR otimizada em área com o número exato de unidades funcionais para cada linha relacionando o grau de paralelismo previamente encontrado. Assim, é construída uma UFR que explora o máximo paralelismo da aplicação, fornecendo o desempenho ideal para o modelo de UFR explorado. É importante ressaltar que somente os grafos reusados são levados em conta nesta caracterização, o que ressalta a vantagem de realizar a exploração dinamicamente.

Na abordagem chamada de baixo custo, a ferramenta também explora os GDD da aplicação e verifica o grau de paralelismo disponível em cada linha da UFR. Entretanto, o projetista fornece à ferramenta a restrição máxima de perda de desempenho que o dispositivo final pode apresentar em relação ao sistema original. Esta restrição máxima dependerá de requisitos de projeto do dispositivo embarcado. Desta forma, a ferramenta remove da caracterização da UFR os GDDs que possuem menor influência na aceleração do desempenho da aplicação. O número de GDD removidos da caracterização dependerá da restrição de perda de desempenho fornecida pelo projetista à ferramenta. Assim, em decorrência da redução do número de grafos a explorar caracterizar a UFR, a ferramenta reduz o número de recursos disponíveis na UFR em relação abordagem ótima.

A verificação do potencial da técnica de *Sleep Transistor* será demonstrada na Seção 4.4 através da execução das aplicações utilizadas na Seção 3.5. Desta maneira, é possível verificar o impacto que o desligamento dos blocos ocasionará na potência e, por conseqüência, na energia consumida pela aplicação. As duas metodologias de desligamento demonstradas na Seção 4.2.1 serão aplicadas. Duas granularidades diferentes de desligamento serão utilizadas para verificar qual destas é a mais viável de ser empregada no dispositivo embarcado.

O impacto das técnicas reunidas será demonstrado na próxima seção, a forma em que estas se complementam é a principal motivação de agregá-las. A ferramenta de exploração realiza sua tarefa estaticamente, ou seja, antes da fabricação do dispositivo é realizada a caracterização da UFR. Desta forma, é realizada uma melhor exploração do paralelismo e ao mesmo tempo é reduzida a área do chip. Entretanto, após sua fabricação, novas aplicações com características diferentes poderão ser executadas na

plataforma e o desempenho destas poderá ser penalizado pela caracterização estática da UFR. Entretanto, a técnica de *Sleep Transistor* agrega dinamicamente, ou seja, em tempo de execução, a redução do consumo de potência na UFR. Com a utilização de ambas as técnicas, em um mesmo projeto embarcado, é possível levantar os requisitos/restrições do dispositivo e balancear a otimização da UFR em tempo de projeto e tempo de execução, não subestimando requisitos de futuras aplicações.

4.4 Resultados

Nesta seção serão apresentados os dados resultantes da exploração de recursos e desligamento de blocos funcionais da UFR. Para melhor visualização dos resultados foram escolhidas as mesmas cinco aplicações do capítulo anterior, a heterogeneidade no comportamento da execução das mesmas foi o principal motivo da seleção. De todo modo, todos os resultados do conjunto de aplicações Mibench podem ser visualizados nas tabelas do Apêndice A e do Apêndice B.

4.4.1 Exploração de Recursos

Como já citado anteriormente, para otimizar a unidade funcional reconfigurável a ferramenta de exploração de recursos baseia-se no paralelismo disponível na aplicação. Nesta seção serão utilizados os modelos propostos na Seção 3.6.1, exceto o Modelo 1 que comparado ao Modelo 2, disponibiliza quantidades semelhantes de unidades funcionais.

Para demonstrar os resultados foram utilizadas duas metodologias que impactam em diferentes restrições dos sistemas embarcados. A primeira, chamada de metodologia ótima, garante que o desempenho seja, no mínimo, igual ao da UFR retangular. Porém, na metodologia de baixo custo, é fornecida à ferramenta uma porcentagem máxima de perda de desempenho, em relação à UFR retangular.

A primeira metodologia é voltada à dispositivos embarcados que priorizam o desempenho e não possuem rígidas restrições de área. Entretanto, o alvo da segunda são os dispositivos que possuem restrições de área e que a perda de desempenho não afeta o seu funcionamento. Para os resultados demonstrados nesta seção a ferramenta foi calibrada para obter no máximo 20% de perda no desempenho da execução das aplicações na metodologia de Baixo Custo.

Como citado anteriormente, é possível acelerar a execução de aplicações, utilizando o sistema reconfigurável, com a presença ou ausência da técnica de especulação. A exploração de recursos obtém diferentes resultados nas duas abordagens. Como visto anteriormente, o número de instruções inseridas em uma única configuração eleva-se com a inserção da técnica de especulação. Isto impacta diretamente a necessidade de maior disponibilidade de recursos na UFR. Assim, como será demonstrada a seguir, existe a necessidade de realizar a exploração dos recursos na presença e na ausência desta técnica.

4.4.1.1 Sem Especulação

Na verificação do impacto que a ferramenta de exploração de recursos foram utilizados os modelos 2, 3 e 4. Sendo o último, o modelo que fornece o desempenho máximo na utilização do sistema reconfigurável sem a técnica especulação. A Tabela 4.1 demonstra o número de unidades funcionais em cada modelo explorado. Nesta tabela, para cada um destes é fornecido o número de unidades funcionais que compõe a

UFR retangular. Também é demonstrado o número de unidades funcionais resultantes da exploração de recursos com a metodologia ótima de exploração e a metodologia de baixo custo.

Tabela 4.1: Número de unidades funcionais nos modelos explorados

	Modelo 2			Modelo 3			Modelo 4		
	Retangular	**Ótimo**	**Baixo Custo**	**Retangular**	**Ótimo**	**Baixo Custo**	**Retangular**	**Ótimo**	**Baixo Custo**
Alu	384	466	409	1800	1003	930	10395	1161	1084
Ld	96	263	216	300	580	453	8442	640	502
Mul	32	48	18	100	88	49	882	94	49
Total	*512*	*777*	*643*	*2200*	*1671*	*1432*	*19719*	*1895*	*1635*

Observa-se na Tabela 4.1 que as metodologias de exploração, no Modelo 2, disponibilizaram um número maior de unidades funcionais na UFR do que a forma retangular. O procedimento da ferramenta de exploração é de verificar o nível de paralelismo de cada linha dos GDDs e desenhar o formato perfeito de uma UFR, para que esta explore o paralelismo máximo desta aplicação. Assim, os resultados demonstram que este modelo de UFR, em sua forma retangular, se apresenta sub-dimensionado para explorar o grau de paralelismo disponível nas aplicações executadas. Será explicitado a seguir, que o acréscimo de área neste modelo elevará o desempenho de algumas aplicações devido à perfeita exploração desta característica.

Entretanto, nos outros modelos explorados obtêm-se uma diminuição no número de unidades funcionais, como no Modelo 4, conclui-se que o nível máximo de paralelismo não é constante em todas as linhas da UFR, como propõem a forma retangular apresentada na Figura 3.3.

A Tabela 4.2 demonstra o fator de redução/aumento de área que as metodologias produziram após a exploração. O Modelo 2, explorado pela metodologia Ótima, acresce 28% na área em relação ao modelo retangular, mas a grandeza deste valor é diminuída para 9% quando é utilizada a metodologia de Baixo Custo. Esta última fornece 27% de redução de área, em relação à UFR resultante da metodologia Ótima.

Nesta Tabela também é demonstrado que no Modelo 3 a metodologia Ótima reduz em 39% a área da arquitetura, e a metodologia de Baixo Custo produz uma redução de 64%. Entretanto, o dado mais relevante é apresentado no Modelo 4, com a utilização da ferramenta são alcançados 5.67 vezes de redução de área, empregando a metodologia Ótima, ou seja, sem perda de desempenho. Com a metodologia de Baixo Custo, eleva-se a economia em área, demonstrando um fator de 6.8 vezes de redução.

Tabela 4.2: Fator de redução de área em relação a UFR retangular

Fator	Modelo 2		Modelo 3		Modelo 4	
	Ótimo	Baixo Custo	Ótimo	Baixo Custo	Ótimo	Baixo Custo
Em relação ao Retangular	+0.28	0.09	-0.39	-0.64	-5.67	-6.80
Em relação ao Ótimo		-0.27		-0.17		-0.17

A Figura 4.7 ilustra a aceleração fornecida pela inserção do sistema reconfigurável nos modelos submetidos à exploração, na extração dos dados desta seção foram disponibilizados 512 blocos na cache de reconfigurações. Analisando os gráficos desta figura, em todos os modelos, as aplicações mantiveram as mesmas acelerações no desempenho em relação à UFR retangular. Entretanto, a aplicação *rijndaeld* obteve uma maior aceleração nas UFR otimizadas comparado à forma retangular nos Modelos 2 e 3. A melhor exploração do paralelismo, resultante da ferramenta ARISE, implicou em um crescimento no desempenho desta aplicação. Como pode ser observada no Apêndice A, a aplicação *gsmd* também se beneficiou deste fato, aumentando sua aceleração de 1.6 para 2.3 no Modelo 2 e de 3 para 3.3 no Modelo 3.

Outro fator importante a ser notado é que a maior redução de área produzida pela exploração, utilizando a metodologia de Baixo Custo, não resultou em perda alguma de desempenho em relação à forma retangular. Além disto, obtiveram-se os mesmos resultados de acréscimo na aceleração da metodologia ótima na aplicação *rijndaeld*. Como citado anteriormente, este é a principal motivação da abordagem de exploração de recursos utilizada, a dinamicidade de execução do tradutor binário reflete neste resultado, os grafos removidos da caracterização da UFR na metodologia de Baixo Custo conseguem se adaptar a forma da UFR resultante e, conseqüentemente, executar as aplicações sem perda de desempenho.

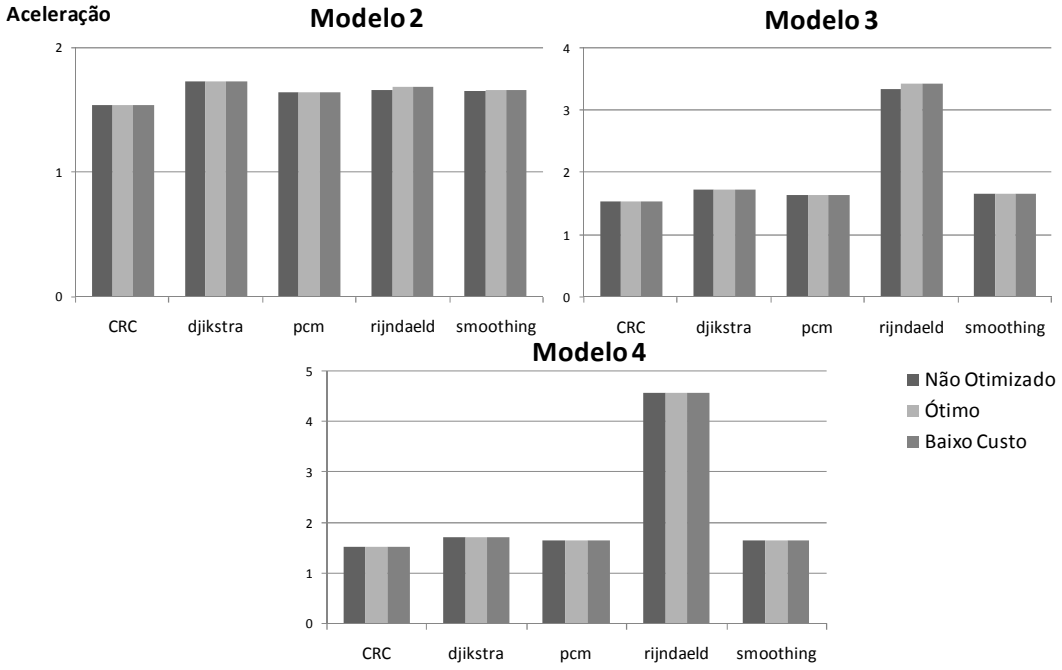

Figura 4.7: Aceleração do desempenho após a exploração de área

Relacionando a Tabela 4.1 e a Figura 4.7, conclui-se que além de fornecer uma redução de área nos Modelos 3 e 4, respectivamente 64% e 6.8 vezes, a metodologia de baixo custo consegue manter a aceleração máxima provida pelo sistema reconfigurável sem a utilização de especulação. Além disto, na Tabela 5.1,Tabela 5.2 e Tabela 5.3 do Apêndice A pode ser constatado que na parametrização da cache reconfiguração e dos modelos de UFR foram realizadas 192 diferentes execuções, sendo que somente duas

destas execuções apresentaram perda de desempenho utilizando a UFR explorada pela metodologia de Baixo Custo.

A redução no número de unidades funcionais impacta diretamente na potência consumida pela UFR. Esta redução é refletida tanto na potência estática quanto na potência dinâmica, devido ao seu circuito ser formado somente de lógica combinacional. A Figura 4.8 ilustra o consumo de potência das aplicações para cada modelo explorado, além da potência consumida pelo processador MIPS desprovido do sistema reconfigurável. Observa-se nesta figura que as UFRs, do Modelo 2, resultantes das metodologias de exploração de recursos fornecem um maior consumo de potência. Entretanto, a metodologia de exploração de Baixo Custo se aproxima do consumo da forma retangular. Como explicitado anteriormente, o maior consumo reflete, pela sua perfeita exploração do paralelismo, em maiores acelerações nas aplicações do que a versão retangular da UFR.

Entretanto, nos outros modelos a diminuição drástica de área impactou no consumo de potência. Na Figura 4.8 pode ser observado que, em todas as aplicações houve uma queda brusca na potência consumida. Na aplicação *rjindael*, a utilização da metodologia ótima reduziu a potência de 8.75 watts para 6.29 watts no Modelo 3. Com o emprego da metodologia de baixo custo, a redução ainda foi maior, alcançando o valor de 5.98 watts. Além disto, como explicitado anteriormente, a UFR resultante proveu nesta aplicação uma perfeita exploração do nível de paralelismo e elevou seu desempenho. A média de redução no consumo de potência, para todas as dezesseis aplicações no Modelo 3, utilizando a metodologia Ótima foi de 35%, elevando-se para 40% na metodologia Baixo Custo.

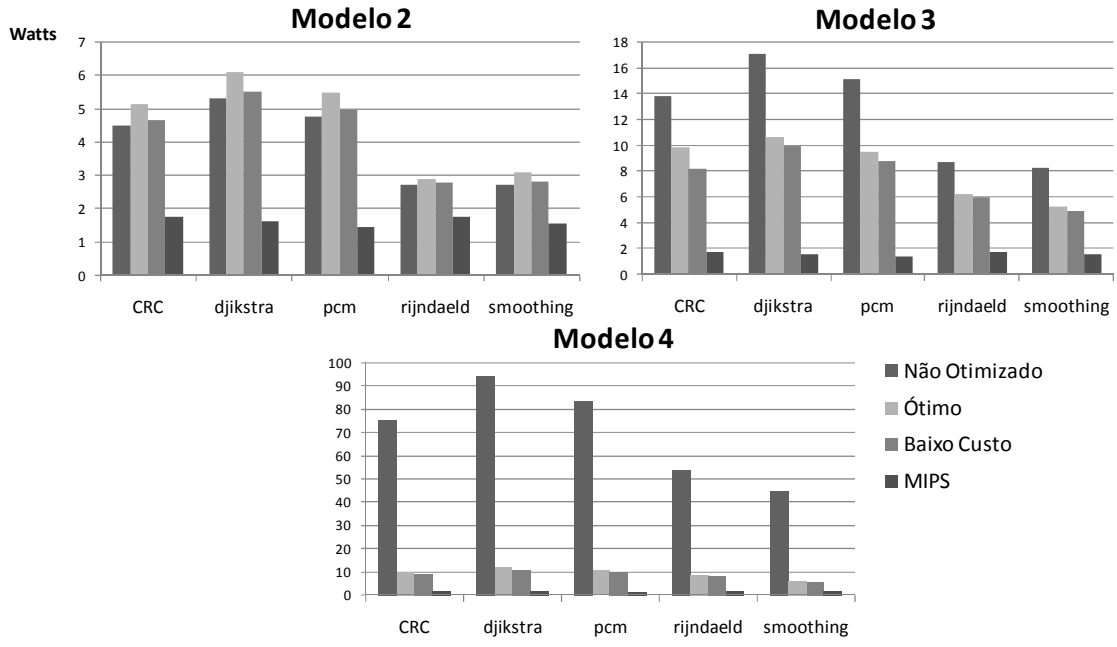

Figura 4.8: Potência consumida após a exploração de área

No Modelo 4 o consumo de potência sem a exploração de recursos torna a implementação deste modelo não factível para um sistema embarcado. A aplicação *smoothing*, no modelo retangular, apresenta um consumo médio de 44.65 watts. A Figura 4.8 demonstra que após a exploração de área, utilizando a metodologia Ótima, a potência desta aplicação foi reduzida para 5.88 watts. Sem perda de desempenho, a

exploração realizada com a metodologia de Baixo Custo diminui ainda mais a potência, apresentando 5.49 watts para esta aplicação. Este fato demonstra a importância da exploração da área da UFR e, conseqüentemente, a viabilidade de implementação da UFR de máximo potencial nos dispositivos embarcados atuais.

A Figura 4.9 ilustra a energia consumida pela execução das aplicações nos modelos submetidos à exploração da ferramenta ARISE. A redução da energia destes modelos é análoga à redução de potência demonstrada na Figura 4.8. No Modelo 2 ocorreu uma elevação no consumo de energia, devido ao acréscimo de unidades funcionais. Entretanto, nos demais modelos, houve uma redução drástica na energia. No Modelo 3 a aplicação *rijndaeld* explorada com a metodologia de Baixo Custo consome 285 mjoules, menor do que os 301 mjoules consumidos pelo processador desprovido da unidade reconfigurável, fato explicado pela aceleração de 3.4 vezes na execução provida pela inserção do sistema reconfigurável.

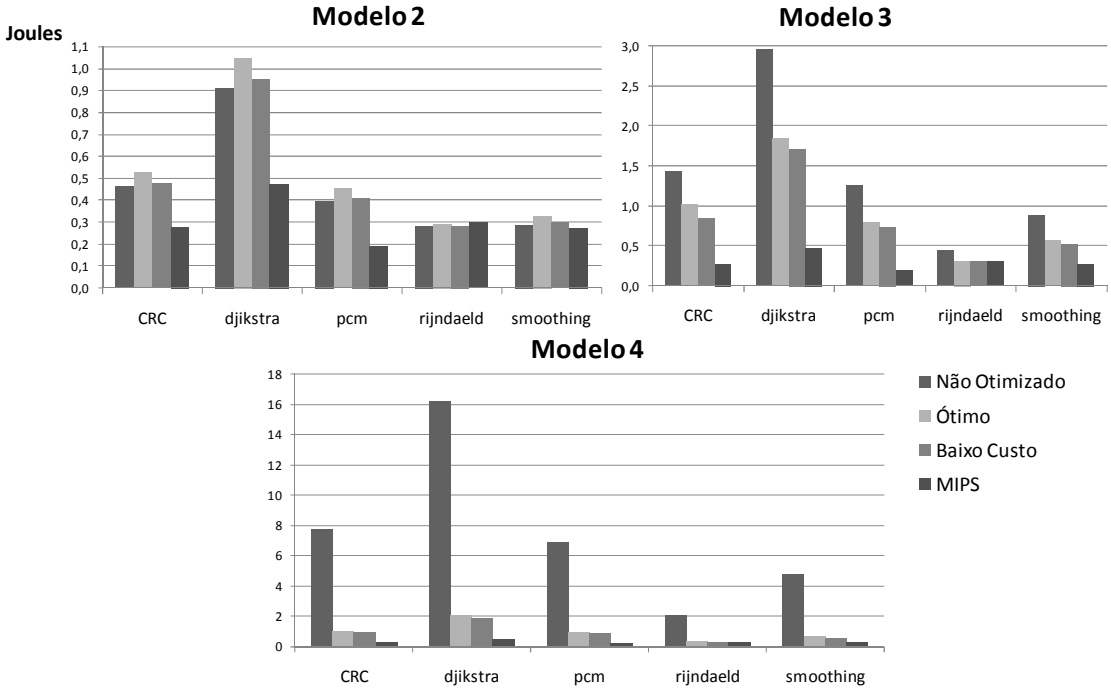

Figura 4.9: Energia consumida após a exploração de área

No Modelo 4 a execução da aplicação *dijkstra* consome, na forma retangular, 16.25 joules. Energia reduzida para 2.07 joules, sem perda de desempenho, na exploração de área realizada com a metodologia ótima. Fato que ilustra o quão necessário é a exploração de recursos em um sistema reconfigurável.

4.4.1.2 Com Especulação

A utilização da técnica de especulação proporciona uma maior exploração do grau de paralelismo disponível na execução. O maior número de instruções inseridas, na utilização desta técnica, em uma única configuração reflete em GDDs mais profundos e largos. Assim, as unidades funcionais reconfiguráveis resultantes da exploração apresentam modificações em relação às apresentadas sem a utilização desta técnica. Neste caso, os modelos 2, 3 e 5 foram escolhidos como exemplares para exploração de recursos. Lembrando que o último representa a UFR que fornece o potencial máximo da abordagem reconfigurável empregando esta técnica.

A Tabela 4.3 ilustra o número de unidades funcionais resultantes, nos modelos explorados, das duas metodologias de exploração. Comparando os resultados desta Tabela com os mostrados na Tabela 4.1, nota-se uma elevação no número de unidades resultantes. A utilização da técnica de especulação implica em uma maior concentração de instruções em uma configuração da UFR. Desta forma, o grau de paralelismo explorado eleva-se, refletindo na necessidade de um maior número de unidades funcionais em relação à mesma exploração sem a utilização desta técnica.

Tabela 4.3: Número de unidades funcionais resultantes da exploração com especulação

	Modelo 2			Modelo 3			Modelo 5		
	Retangular	**Ótimo**	**Baixo Custo**	**Retangular**	**Ótimo**	**Baixo Custo**	**Retangular**	**Ótimo**	**Baixo Custo**
Alu	384	514	443	1800	1029	964	12750	1506	1434
Ld	96	269	197	300	570	438	10200	752	600
Mul	32	51	26	100	74	56	1275	74	56
Total	*512*	*834*	*666*	*2200*	*1673*	*1458*	*24225*	*2332*	*2090*

A Tabela 4.4 demonstra o fator de redução na área resultante da exploração pela ferramenta ARISE nas duas metodologias de exploração propostas. Comparado aos valores demonstrados nesta tabela com a Tabela 4.2, nota-se uma elevação da área ocupada em decorrência da inserção da técnica de especulação. O emprego da técnica de especulação ocasionou um aumento de 5% na área ocupada no Modelo 2 comparada a mesma metodologia de exploração sem o uso de especulação. Da mesma forma, dados do Modelo 3 desprovido de especulação demonstram uma redução de 64% de área na exploração pela metodologia de Baixo Custo, percentagem reduzida para 59% na inserção da técnica.

No modelo que explora o máximo potencial da abordagem reconfigurável dotada de especulação (Modelo 5) a ferramenta produziu reduções significativas da área da UFR. A exploração utilizando a metodologia ótima resultou em uma área 8.46 vezes menor do que a UFR retangular, valor elevado para 9.38 vezes com a metodologia de Baixo Custo. Na Tabela 4.4, a utilização da metodologia ótima refletiu em uma elevação de 33% na área do Modelo 2. Como explicitado anteriormente, em decorrência da ineficiente exploração do grau de paralelismo deste modelo na sua forma retangular, a ferramenta adicionou unidades funcionais para aperfeiçoar esta característica.

Tabela 4.4: Fator de redução de área em relação a UFR retangular com a utilização da técnica de especulação

Fator	Modelo 2		Modelo 3		Modelo 5	
	Ótimo	Baixo Custo	Ótimo	Baixo Custo	Ótimo	Baixo Custo
Em relação ao Retangular	+0.33	+0.25	-0.41	-0.59	-8.46	-9.38
Em relação ao Ótimo		-0.27		-0.13		-0.10

A Figura 4.10 ilustra a aceleração das aplicações nos modelos explorados pela ferramenta. Pode ser observado nesta figura, que a redução de unidades funcionais não produziu danos no desempenho das aplicações. Ainda, a utilização da metodologia de Baixo Custo não acarretou perdas de desempenho. Em contraponto, como pode ser observado na Tabela 5.10 a exploração utilizando a metodologia Ótima no Modelo 2 refletiu em uma maior aceleração da aplicação *jpege*, elevando a sua aceleração de 2.91 para 3.73.

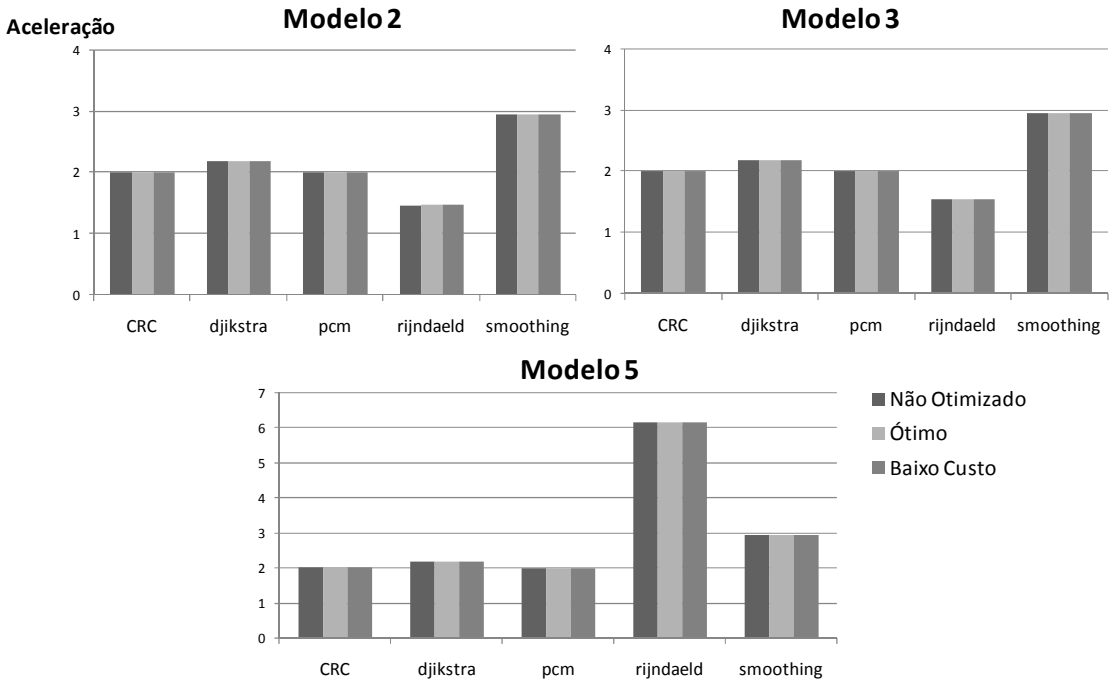

Figura 4.10: Aceleração das aplicações após a exploração de área

A utilização da exploração da metodologia de Baixo Custo mostrou-se altamente eficiente com a utilização de especulação. Como pode ser observado na Tabela 5.10, Tabela 5.11 e Tabela 5.12, na parametrização de blocos na cache e nos modelos de UFR foram totalizadas 192 execuções, sendo que somente 6 destas mostraram alguma perda de desempenho na utilização desta metodologia, 122 apresentaram as mesmas acelerações e 64 execuções alcançaram maiores acelerações do que a forma retangular.

A exploração de área da UFR repercutiu em um mesmo nível de redução de potência consumida pelos modelos. Na Figura 4.11 pode ser observado um aumento no consumo do Modelo 2, devido ao aumento do número de unidades funcionais. Todavia, nos Modelos 3 e 5 houve uma redução significativa no consumo de potência. Principalmente no último, onde a média de consumo de potência de todas as aplicações foi reduzida em um fator sete. No Modelo 5 com a metodologia de exploração Ótima a aplicação *pcm* reduziu a potência de 101.74 watts para 13.19 watts. Ainda, sem nenhuma perda no desempenho a exploração de Baixo Custo consumiu somente 12.56 watts em média.

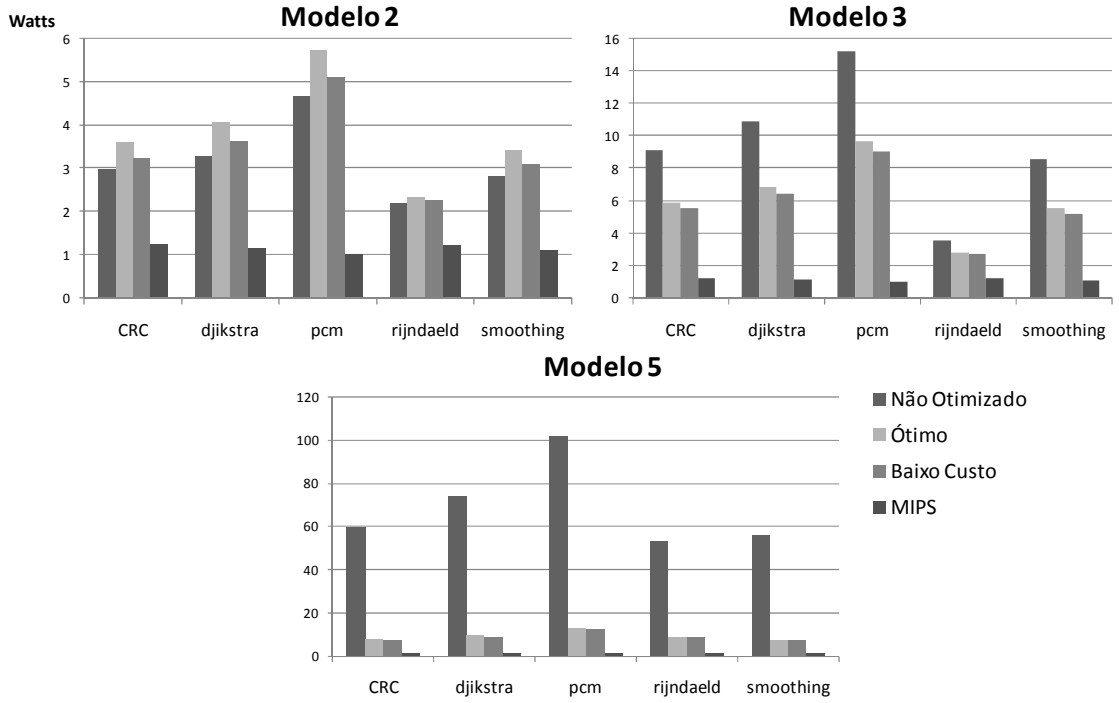

Figura 4.11: Consumo de potência dos modelos explorados pela ferramenta

A energia consumida, na execução das aplicações, também obteve uma redução significativa em decorrência da exploração realizada. A Figura 4.12 ilustra a diminuição do consumo da arquitetura. A média de redução de todas as aplicações executadas no Modelo 3 foi de 35% utilizando a metodologia Ótima, redução elevada para 39% na metodologia de Baixo Custo. O maior fator de redução de energia ocorreu no Modelo 5, sendo este de 86.75% na utilização da metodologia Ótima e 87.30% resultante da metodologia de Baixo Custo.

No Modelo 3 a exploração com a metodologia Ótima da aplicação *dijkstra* reduziu a energia consumida de 1.48 joules para 0.93 joules, na utilização da metodologia de Baixo Custo a energia consumida foi de 0.88 joules. Ainda, no Modelo 5, sem afetar a aceleração desta mesma aplicação, a exploração com a metodologia Ótima diminui a energia de 10 joules para 1.28 joules.

Entretanto, como pode ser observado na Figura 4.12, mesmo com a utilização da ferramenta ARISE a energia consumida pelos Modelos 3 e 5 na execução das aplicações é proibitiva para um sistema embarcado alimentado por bateria. Ainda, comparado à execução destas mesmas aplicações no processador MIPS R3000 o sistema reconfigurável eleva a energia consumida em ordens de grandeza nos modelos otimizados. No Modelo 3, a aplicação *pcm* explorada pela metodologia de Baixo Custo consome 621 mjoules, sendo que a execução desta mesma aplicação no processador consome 195 mjoules, ou seja, uma elevação em um fator três no consumo de energia. Já no Modelo 5 com as mesmas configurações anteriores, a aplicação consome 859 mjoules em sua execução, quatro vezes maior que o consumo do processador MIPS R3000. Desta forma, soluções que evitem o consumo de potência e, conseqüentemente, de energia devem ser adotados em sistemas reconfiguráveis para que estes sejam factíveis de serem introduzidos em um dispositivo embarcado.

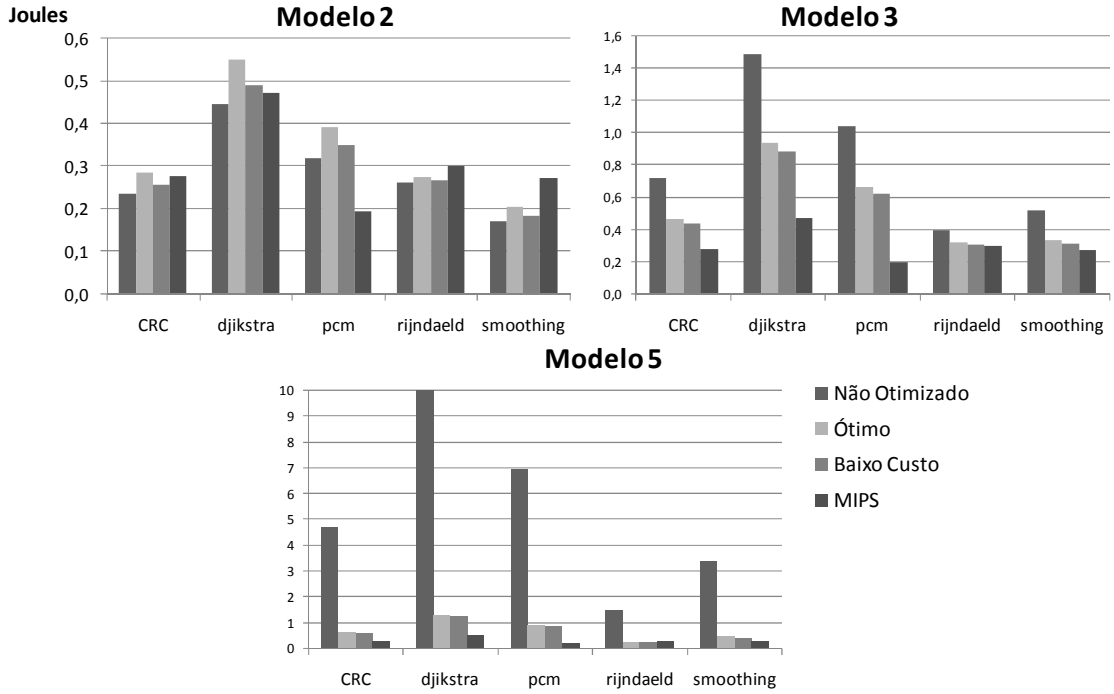

Figura 4.12: Energia consumida pelas aplicações nos modelos explorados

4.4.2 Exploração de Potência

Na Seção 4.4.1 foi demonstrado o impacto ocasionado pela exploração da ferramenta ARISE na área, potência e energia consumida pelo sistema reconfigurável. Analisando os resultados obtidos através das unidades funcionais reconfiguráveis concebidas desta exploração, foi verificado que é possível implementar o sistema reconfigurável nas tecnologias atuais. Entretanto, como pode ser observado nas figuras da seção anterior que relatam a potência e a energia dos modelos de UFR otimizados, este sistema reconfigurável ainda é restritivo para dispositivos embarcados.

A inserção da UFR Modelo 4, otimizado pela metodologia Ótima, no processador MIPS R3000 eleva, em média, 5.03 vezes a potência da arquitetura. Este fator sobe para 5.08 vezes na inserção do Modelo 5 com as mesmas características.

Dispositivos embarcados que são alimentados por bateria não comportam tal nível de consumo de potência. Assim, nesta seção será demonstrada a verificação do potencial da técnica de *Sleep Transitor* nos modelos concebidos anteriormente pela ferramenta de ARISE. A motivação para a utilização desta técnica é tentar reduzir o consumo de potência das UFRs na tentativa de alcançar fatores aceitáveis para dispositivos alimentados por bateria.

Na verificação do potencial desta abordagem foram utilizados dois grãos de desligamento para o sistema reconfigurável: desligamento de nível e desligamento de unidades funcionais. O primeiro possui um grão mais grosso, ou seja, é inserido um *sleep transistor* para cada nível da UFR. Neste grão o controle é reduzido e são inseridos poucos transistores na UFR. O segundo fornece um grão mais fino de desligamento, cada unidade funcional possui um *sleep transistor* para controlar a sua

alimentação. O impacto no aumento da área é maior, além de tornar o controle da técnica pelo TB mais detalhado.

Para avaliar a técnica proposta foram executas as dezesseis aplicações propostas na Seção 3.6.1, entretanto para melhor visualização dos resultados foram escolhidas as mesmas aplicações da seção anterior, pelos mesmos motivos relatados. Entretanto, os resultados de todas as aplicações podem ser visualizados nas tabelas do Apêndice B. Todos os valores demonstrados nos gráficos desta seção levam em conta a disponibilidade de 512 blocos na cache de configurações, utilizando as UFR resultantes da metodologia Ótima de exploração de área.

4.4.2.1 Sem Especulação

A Figura 4.13 ilustra a redução na potência consumida na inserção da técnica de *Sleep Transistor* nos Modelos 2 e 4, sem a utilização de especulação. A abordagem de desligamento de níveis no Modelo 2 resultou em uma redução de 6.08 Watts para 2.82 Watts na aplicação *dijkstra*, um fator de redução maior que dois. Esta mesma abordagem de desligamento no Modelo 4 reduziu a potência desta aplicação de 12.03 Watts para 3.01 Watts, elevando o fator de redução para quatro.

Entretanto, como pode ser observada nesta mesma Figura, esta abordagem de desligamento resulta, em média no Modelo 4, em um consumo três vezes maior comparado a arquitetura desprovida do sistema reconfigurável. Assim, a abordagem de desligamento de unidades funcionais foi verificada. Desta forma, o consumo da aplicação *dijkstra* no Modelo 2 recua para 2.07 Watts. Portanto, nesta abordagem de desligamento, a inserção do sistema reconfigurável no processador MIPS R3000, reflete em uma elevação de somente 29.82% no consumo de potência na execução da aplicação *djikstra*.

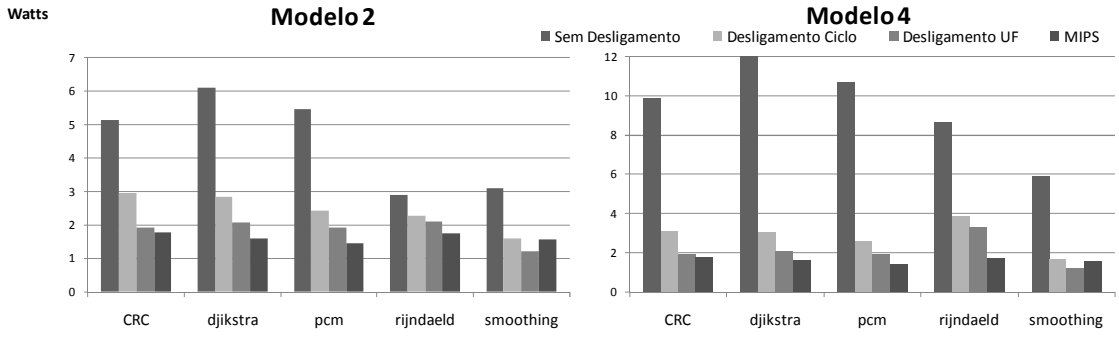

Figura 4.13: Redução no consumo de potência na inserção da técnica de Sleep Transistor

É importante ressaltar que a utilização da abordagem de desligamento de unidade funcional fornece valores parecidos para os diversos modelos de UFR, pois as unidades que não estão alocadas com instruções não consomem nenhum dos dois tipos de potência (estática e dinâmica). Assim, a execução da aplicação em questão no Modelo 4, utilizando a abordagem de desligamento de unidades funcionais, eleva o consumo de potência nos mesmos 29.82% relatados no Modelo 2. Fato que explicita que, esta aplicação não explorou um grau maior de paralelismo no Modelo 4, assim as mesmas unidades funcionais foram desligadas no Modelo 2 e 4 na execução desta aplicação.

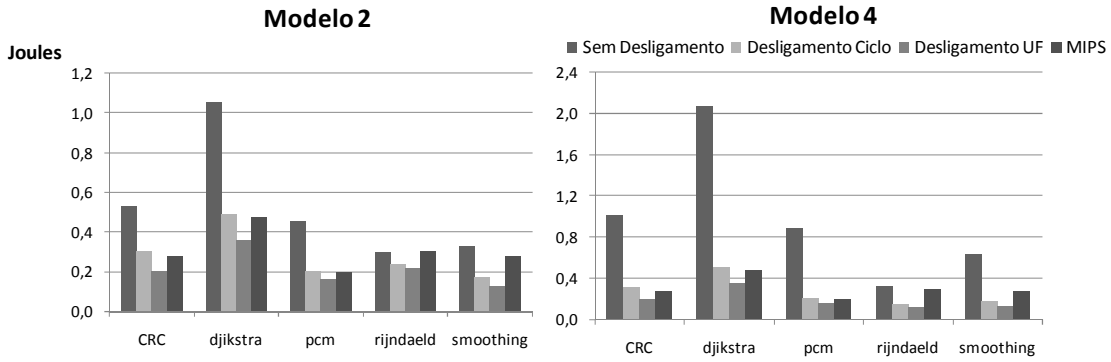

Figura 4.14: Redução no consumo de energia na inserção da técnica de Sleep Transistor

A redução da potência consumida pelo sistema reflete diretamente no consumo de energia. A Figura 4.14 demonstra a energia consumida na execução das aplicações nos modelos em questão. A energia consumida na execução da aplicação *dijkstra* no Modelo 2 é reduzida de 1.05 joules para 0.48 Joules na utilização da abordagem de desligamento de níveis. Este valor é reduzido para 0.35 Joules utilizando a abordagem de desligamento de unidade funcional, valor menor que os 0.473 Joules que o processador consome, desprovido do sistema reconfigurável, ao executar esta aplicação.

Dois fatores contribuem para o menor consumo de energia do sistema reconfigurável em relação ao consumo da mesma execução realizada no processador: diminuição do tempo de computação e redução do número de acessos a memória de programa. Como demonstrado no capítulo anterior, o sistema reconfigurável produz uma aceleração na execução do processador. Com a abordagem de desligamento de unidades funcionais obtêm-se pouca elevação no consumo de potência em relação ao processador. Assim, ao executar a aplicação em um menor tempo, conclui-se um consumo menor de energia. Ao executar uma configuração na UFR são evitados vários acessos à memória principal em função da busca das instruções. Como o acesso à memória é energeticamente oneroso, obtêm-se uma redução de energia.

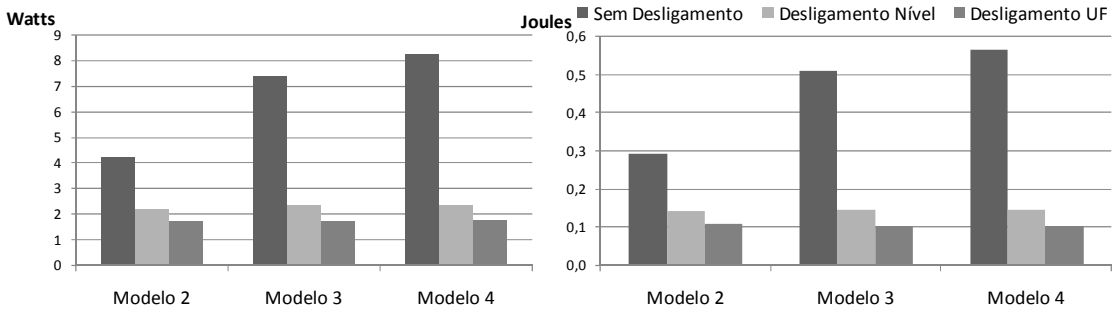

Figura 4.15: Média de redução de potência e energia em diferentes modelos de UFR

A média redução de consumo de potência e energia em todos os modelos de UFR, na execução de todas as aplicações é demonstrada na Figura 4.15. Um importante fato a ser observado nesta figura é que o desligamento por níveis fornece valores de potência e energia quase constantes em todos os modelos de UFR, devido à baixa disparidade do número de níveis ociosos nas execuções dos diferentes modelos. A inserção desta técnica de desligamento refletiu em uma redução média de potência, no Modelo 3, de 3.16 vezes, sendo elevada para 3.50 vezes no Modelo 4. Entretanto, como esperado, a abordagem de desligamento de unidades funcionais apresentou um maior nível de

redução de potência. Para o Modelo 3 a redução foi de 4.29 vezes, elevando-se para 4.74 vezes no Modelo 4.

Devido à diminuição do tempo de computação das aplicações, a média de redução da energia é maior em relação à média de redução de potência. A metodologia de desligamento de níveis no Modelo 3 impactou em 3.49 vezes de redução, crescendo para 3.88 vezes no Modelo 4. Na metodologia de desligamento por unidades funcionais o mesmo ocorreu, no Modelo 3 houve uma redução de 4.98 vezes elevando-se para 5.55 vezes no Modelo 4.

4.4.2.2 Com Especulação

Na Figura 4.16 é demonstrada a redução do consumo de potência nos Modelos 2 e 5. O desligamento de níveis demonstrou um fator de redução menor do que dois para o Modelo 2 e menor do que quatro no Modelo 4. Demonstrando que, com a utilização da técnica de especulação, não se consegue alcançar os mesmos fatores de redução de consumo de potência quando da ausência desta técnica. Devido à inserção de mais instruções em uma configuração, com a utilização de especulação, o número de níveis que permanecem ociosos é menor, provendo menos espaço para a exploração desta abordagem de desligamento.

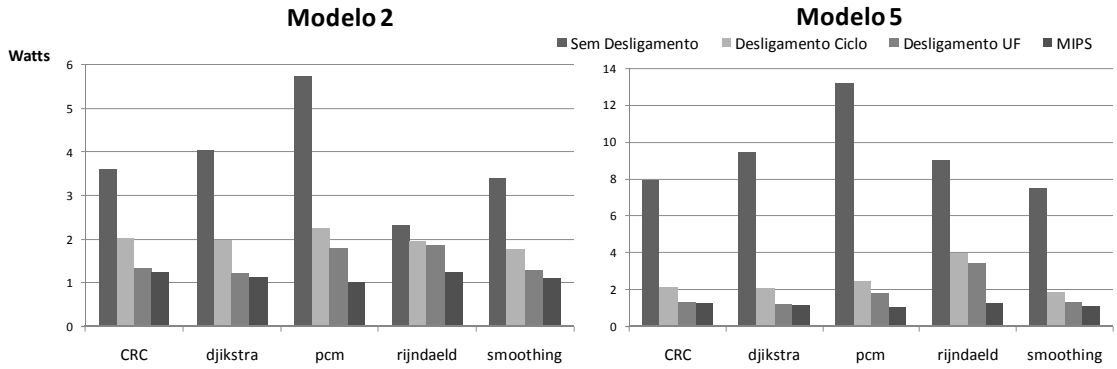

Figura 4.16: Redução no consumo de potência na inserção da técnica de Sleep Transistor na UFR utilizando especulação

No Modelo 2, a Figura 4.16 ilustra uma redução para a aplicação *dijkstra* com a utilização da metodologia de desligamento de níveis de 4.05 watts para 1.97 watts, sendo reduzido para 1.22 watts com o emprego do desligamento de níveis. No Modelo 5 a inserção da técnica de desligamento de níveis reduziu a potência consumida por esta aplicação de 9.46 watts para 2.1 watts, provendo uma redução para 1.22 watts com a inserção da abordagem de desligamento de unidades funcionais.

Entretanto, a utilização da técnica de especulação reflete em uma maior redução de energia comparado ao sistema desprovido desta técnica. Esta técnica provê uma diminuição no tempo de computação que tem um fator maior que o aumento da potência consumida assim alcança-se uma redução ainda maior na energia. A energia consumida na execução da aplicação *dijkstra* no Modelo 2 é reduzida de 0.55 Joules para 0.26 Joules na utilização da abordagem de desligamento de níveis. Este valor é reduzido para 0.16 Joules utilizando a abordagem de desligamento de unidade funcional. Sendo 0.47 Joules a energia consumida para executar esta aplicação no processador MIPS R3000, é alcançado um fator de redução maior que dois na inserção do sistema reconfigurável utilizando especulação.

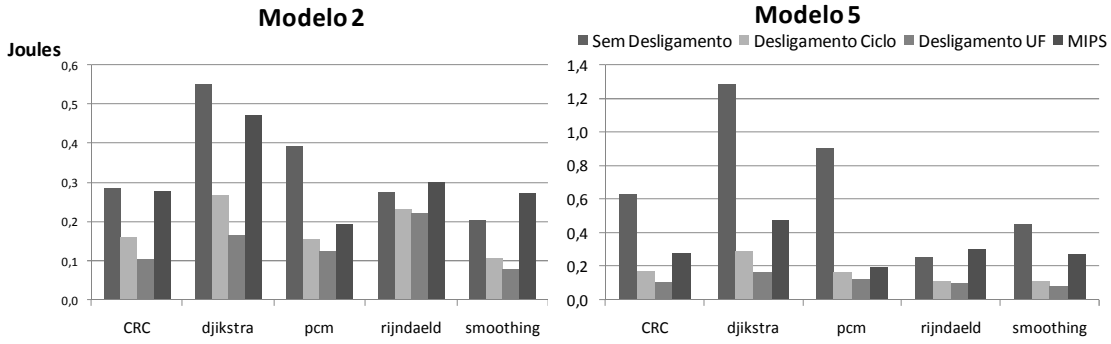

Figura 4.17: Redução no consumo de energia na inserção da técnica de Sleep Transistor na UFR utilizando especulação

Retomando os valores produzidos pela execução da aplicação *dijkstra* sem a utilização de especulação tem-se: 0.48 Joules para a abordagem de desligamento de níveis e 0.35 mJoules para a abordagem de desligamento de unidade funcional. Assim, conclui-se que a potência consumida pela especulação é maior, pelo fato de mais instruções serem alocadas em uma configuração, o que evita o desligamento das unidades funcionais. Todavia, comparando com os dados obtidos com especulação, a diminuição do tempo de computação com a utilização desta técnica é maior que a elevação da potência, desta forma obtêm-se uma redução no consumo de energia.

4.4.3 Resumo do Impacto das otimizações

As comprovações do impacto das otimizações realizadas durante este trabalho estão divididas entre os capítulos anteriores. Entretanto, esta seção irá realizar um resumo sobre o conjunto de otimizações abordadas e quais foram os resultados finais obtidos com os dois modelos que oferecem o máximo potencial da abordagem reconfigurável. O primeiro, Modelo 4, sem utilização da técnica de especulação e o segundo, Modelo 5, provido desta técnica.

Tabela 4.5: Conjunto das Otimizações realizadas neste trabalho

	Sem Especulação (Modelo 4)		Com Especulação (Modelo5)		
	Retangular	Baixo Custo	Retangular	Baixo Custo	MIPS R3000
Área (Portas)	37.025.124	4.749.805	63.544.215	6.121.960	26.886
Potência (Watts)	61,920	1,730	60,399	1,639	1,636
Energia (Joules)	4,376	0,101	3,330	0,073	0,193
Aceleração	1,8966	1,8961	2,6420	2,6419	1

Como explicitado nos capítulos anteriores, foram realizados dois tipos de otimização: otimização de recursos e otimização de potência. A primeira é realizada pela ferramenta ARISE de forma estática e resulta em uma UFR otimizada que, organiza as unidades funcionais dentro da arquitetura reconfigurável para melhor explorar o paralelismo da aplicação. Entretanto, a exploração da adaptabilidade do algoritmo de tradução binária produz as mesmas acelerações das UFR retangulares nas UFR otimizadas. A segunda também explora o mecanismo de TB e em tempo de

execução provê o desligamento de blocos funcionais da UFR que não estão sendo utilizados para computação em um dado instante de tempo.

A Tabela 4.5 apresenta informações sobre os dois modelos, a forma retangular não otimizada representa os resultados obtidos no Capítulo 3, ou seja, antes da exploração. Entretanto, a forma Baixo Custo são dados obtidos da metodologia de exploração de área de Baixo Custo utilizando *Sleep transistor* com o grão de desligamento de unidades funcionais. As informações de potência, energia e aceleração é a média da execução de todas as dezesseis aplicações apresentadas na Seção 3.6.1.

O impacto deste trabalho no Modelo 4 demonstra uma redução de 7.8 vezes na área do sistema, 35.79 vezes na potência consumida e 43.32 vezes na energia, todos estes dados apresentando uma perda de 0,026% de desempenho. No Modelo 5 demonstrou-se uma redução de 10.3 vezes na área de UFR, 36.85 vezes na potência consumida 45.61 vezes na energia, apresentando somente 0,003% de perda de desempenho.

A última coluna da Tabela 4.5 demonstra os dados relativos a área, potência e energia do processador MIPS R3000. A área dos Modelos 4 e 5 ainda é significativamente maior que a do processador. Entretanto, com as otimizações realizadas com ARISE a distância entre a área do processador e a dos Modelos foi reduzida em grandes fatores. Comparando a potência e energia consumida pelos Modelos e o processador MIPS, a utilização da técnica de Sleep Transistor produziu consumos parecidos em ambos os casos, apesar da maior área ocupada pelos Modelos. Como o sistema reconfigurável ocasionou uma aceleração na execução das aplicações, e a potência nos Modelos é parecida ao do processador MIPS, a energia consumida é menor em relação à do processador.

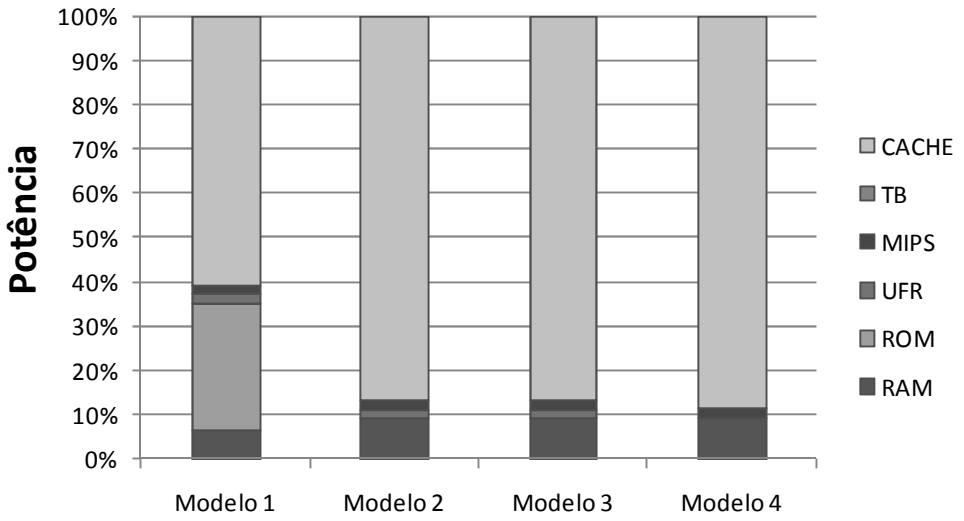

Figura 4.18: Consumo relativo de potência dos componentes do sistema reconfigurável após as otimizações de área e de potência

A Figura 4.18 demonstra o consumo relativo de potência de todos os componentes do sistema reconfigurável. Comparando a Figura 4.1 com a Figura 4.18 nota-se uma grande diminuição no consumo de potência da unidade funcional reconfigurável. Na primeira Figura a média do consumo deste componente em todos os modelos é de 89%

do consumo total do sistema, após as otimizações de área e potência esta valor é reduzido para 2,23% do consumo relativo total da arquitetura. Desta forma, a cache de reconfigurações se tornou, após as otimizações, o componente que possui a maior significância no consumo total do sistema, apresentando 61% do consumo médio em todos os modelos.

Tabela 4.6: Área, em Bytes, da cache de reconfigurações nos modelos otimizados

Blocos (em Bytes)	Modelo 4		Modelo 5	
	Retangular	Baixo Custo	Retangular	Baixo Custo
2	58.091	5.116	71.420	6.515
4	116.118	10.169	142.776	12.965
8	240.954	21.146	296.231	26.983
16	464.474	40.674	571.106	51.860
32	948.667	82.983	1.166.437	105.810
64	1.857.896	162.696	2.284.424	207.440
128	3.716.359	325.464	4.569.613	414.952
256	7.431.584	650.784	9.137.696	829.760
512	14.863.173	1.301.573	18.275.397	1.659.525

Como foi explicitado anteriormente, a exploração de recursos pela ferramenta ARISE não apenas diminui a área da unidade funcional reconfigurável, a diminuição de recursos provida por esta ferramenta impacta diretamente nos bits de configuração da UFR. Assim, o número de bits à ser armazenado na cache de reconfigurações diminui proporcionalmente ao número de recursos da UFR. A Tabela 4.6 demonstra a área da cache, em bytes, necessária para implementar os Modelos 4 e 5 com a forma retangular da UFR e após a exploração da ferramenta ARISE com a metodologia de Baixo Custo. Nota-se que a exploração da UFR impactou drasticamente na área da cache de reconfigurações, para o Modelo 4 com 16 blocos para configurações o modelo retangular necessita 464.474 bytes, valor reduzido para 40.674 bytes após a exploração da ferramenta. O mesmo ocorreu com o Modelo 5, o modelo retangular necessita de 571.106 bytes para armazenar 16 blocos de configurações, entretanto com a exploração este valor cai para 51.860 bytes, um fator de redução de onze vezes.

Desta forma, conclui-se que as explorações realizadas pela ferramenta ARISE foram de extrema necessidade para a toda plataforma reconfigurável. A impossibilidade de prototipação era visível na plataforma original não otimizada, após todas as explorações realizadas mostrou-se que é possível implementar a abordagem na tecnologia atual. Além disto, foi demonstrado que o consumo de energia médio é menor que o consumo de energia do processador desprovido do sistema reconfigurável. Assim, todas estas otimizações viabilizaram a inserção da plataforma reconfigurável em um sistema embarcado alimentado por bateria.

5 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho demonstrou o potencial de aceleração, na execução de aplicações, da arquitetura reconfigurável proposta por (BECK, 2006a) acoplada a um processador MIPS R3000. Além disto, foram aplicadas otimizações nesta arquitetura que reduziram a área ocupada pela mesma e a potência consumida. O impacto das otimizações tornou viável a inserção desta arquitetura em um dispositivo embarcado.

No capítulo 3 foi demonstrado que a utilização do mecanismo de tradução binária proposto em (BECK, 2005) acoplado a um processador MIPS R3000 e a uma unidade funcional reconfigurável provê ganhos significativos de desempenho na execução de aplicações heterogêneas. Ainda, a utilização deste mecanismo fornece compatibilidade de software, ou seja, a inserção da plataforma reconfigurável não implica em nenhuma modificação no código binário da aplicação.

O capitulo 4 agregou algumas otimizações à arquitetura proposta no capítulo 3. Os resultados providos pela ferramenta ARISE provaram a necessidade de se explorar os diversos graus de paralelismo existente na execução de uma aplicação. Ainda, a partir da diversidade desta característica é possível idealizar uma unidade reconfigurável perfeita para explorar tal paralelismo. A otimização realizada pela ferramenta reduziu drasticamente a área ocupada pela UFR e, conseqüentemente, a potência consumida pelo sistema. Agregada a adaptabilidade do mecanismo de tradução binária a ferramenta idealizou novas unidades funcionais reconfiguráveis que resultaram em mesmas acelerações no desempenho das aplicações providas pelas UFR retangulares.

Neste mesmo capítulo foi verificado o potencial de redução de potência da técnica de *Sleep Transistor* acoplada à unidade funcional reconfigurável. Mostrou-se que o impacto na redução de potência provido por esta técnica torna viável a inserção da plataforma proposta em um dispositivo embarcado dotado de bateria.

Assim, as contribuições deste trabalho tanto com ferramentas como resultados possibilitam expansões e inícios de outros trabalhos. Na próxima seção algumas idéias de trabalhos futuros serão exploradas.

5.1 Trabalhos Futuros

5.1.1 Exploração de paralelismo em nível de configurações

Em todas as abordagens, sobre arquiteturas reconfiguráveis, existe uma exploração da execução de porções da aplicação em uma unidade reconfigurável (cada porção

sendo chamada de configuração), sendo a unidade central de processamento um PPG fixo. O grau de paralelismo explorado por esta abordagem fica restrito a somente uma configuração. Nenhum trabalho ainda foi proposto em arquiteturas reconfiguráveis realizando a exploração de paralelismo sobre várias configurações. Assim, um possível trabalho futuro seria disponibilizar, em hardware, uma arquitetura reconfigurável capaz de explorar ao máximo o paralelismo existente nas aplicações, utilizando duas abordagens já conhecidas e validadas na literatura: arquiteturas reconfiguráveis e meios de comunicação utilizados em sistemas multiprocessados.

Os meios de comunicação poderiam interligar várias UFR e realizar a computação de configurações de uma mesma aplicação, sem dependências de dados, em paralelo. Ainda, pode-se explorar o paralelismo em nível de *threads* e executar estas separadamente em cada UFR. Desta forma, agrega-se a exploração do nível de paralelismo em nível de instruções dentro de cada UFR, e em nível de *threads* entre UFR.

5.1.2 Estudo de novas modelagens da UFR

A grande proporção de área ocupada pelo mecanismo de interconexão da UFR é um fato motivador para a modelagem de uma nova arquitetura reconfigurável. Como demonstrado neste trabalho, a área ocupada por multiplexadores e demultiplexadores alcança 50% da área da UFR. O mecanismo simples de tradução binária causa este impacto, na abordagem atual existem conexões entre todas as unidades funcionais, assim a comunicação e, conseqüentemente, a alocação das instruções se tornam fáceis de ser realizadas.

Como trabalhos futuros, novos modelos de UFR com restrições na comunicação entre as unidades funcionais podem ser idealizados para se obter uma menor área de interconexão. Entretanto, obrigatoriamente o hardware de tradução binária também deverá ser modificado para conseguir suportar as restrições de comunicação impostas.

5.1.3 Suporte a instruções de ponto flutuante

Atualmente, as unidades funcionais existentes na UFR têm o poder de executar instruções que manipulam dados inteiros. Assim, aplicações que possuem dados de ponto flutuante não se beneficiam da aceleração provida pela unidade reconfigurável.

Uma proposta interessante de trabalho futuro seria formalizar um hardware de tradução de dados de ponto flutuante para ponto inteiro. Assim, não ocasionaria nenhuma modificação na arquitetura reconfigurável atual. Agregar-se-ia o hardware no tradutor binário, este converteria o dado em ponto flutuante para inteiro, e finalmente executaria este na arquitetura reconfigurável. A complexidade do hardware dependerá da precisão do algoritmo implementado. Entretanto, o ponto mais importante desta abordagem é que o sistema reconfigurável continuaria provendo compatibilidade de software.

REFERÊNCIAS

ARM Homepage. Disponível em: <http://www.arm.com/products/CPUs/ARM1176.html>. Acesso em: 10 feb. 2008.

AUSTIN, T. et al. Mobile Supercomputers. **IEEE Computer**, New York, v. 37, n. 5, p. 81-83, May 2004.

ARNOLD, M.; CORPORAAL, H. Designing domain specific processors. In: INTERNATIONAL CONFERENCE ON HARDWARE SOFTWARE CODESIGN, 9.,2001, Copenhagen. **Proceedings…** New York: ACM Press, 2001. p. 61–66.

BACON, D. F.; GRAHAM, S. L.; SHARP O. J. Compiler Transformations for High-Performance Computing, **ACM Computing Surveys**, New York, v. 26, n. 4, p. 345-420, December 1994.

BECK FILHO, A. C. S.; CARRO, L. Dynamic reconfiguration with binary translation: breaking the ILP barrier with software compatibility. In: DESIGN AUTOMATION CONFERENCE, DAC, 42., 2005, Anaheim. **Proceedings…** New York: ACM Press, 2005. p. 732 – 737.

BECK FILHO, A.C.S.; CARRO, L. Automatic Dataflow Execution with Reconfiguration and Dynamic Instruction Merging. In: VERY LARGE SCALE INTEGRATION, VLSI-SOC, 2006, Perth. **Proceedings…** New York: IEEE Computer Society, 2007. p. 30–35.

BECK FILHO, A.C.S.; RUTZIG, M.B.; CARRO, L. Transparent Reconfigurable Acceleration for Heterogeneous Embedded Applications. Aceito para DATE, 2008, Munich.

BENINI, L.; MICHELI, G. Symbolic techniques of clock-gating logic for power optimization of control-oriented synchronous networks. In: DESIGN, AUTOMATION AND TEST IN EUROPE, DATE, 1997, Paris. **Proceedings…** Washington: IEEE Computer Society, 1997. p. 514–520.

BURGER, D.; AUSTIN, T.M. The SimpleScalar Tool Set. **ACM SIGARCH Computer Architecture News**, New York, v. 25, n. 3, p. 13 – 25, June 1997.

CLARK, N.; TANG, W.; MAHLKE, S. Automatically generating Custom instruction set extensions. In: WORKSHOP OF APPLICATION-SPECIFIC PROCESSORS, 2002, Istanbul, **Proceedings…** [S.l.]: IEEE Computer Society, 2002.

CLARK, N.; ZHONG, H.; MAHLKE, S. Processor Acceleration through Automated Instruction Set Customization. In: INTERNATIONAL SYMPOSIUM ON

MICROARCHITECTURE, 36., 2003, San Diego. **Proceedings…** Washington: IEEE Computer Society, 2003. p. 129.

COMPTON, K.; HAUCK, S. Reconfigurable computing: A survey of systems and software. **ACM Computing Surveys**, New York, v. 34, n. 2, p. 171- 210, June 2002.

DENNIS, J.B. First version of a data flow procedure language. In: PROGRAMMING SYMPOSIUM, 1974, Paris. **Proceedings…** London: Springer, 1974. p. 362 – 376. (Lecture Notes in Computer Science, v. 19).

ERNST, R. Codesign of embedded systems: Status and trends. **Design and Test of Computers**, Santa Barbara, v. 15, n. 2, p. 45 – 53, April 1998.

FLYNN, M.J.; HUNG, P. Microprocessor design issues: thoughts on the road ahead. **IEEE Micro**, Los Alamitos, v. 25, n. 3, p. 16 – 31, May 2005.

GAJSKI, D. et al. SpecSyn: An Environment Supporting the Specify-Explore-Refine Paradigm for Hardware/Software System Design. **IEEE Transactions on VLSI Systems**, Princeton, v. 6, n. 1, p. 84-100, March 1998.

GEGLER, W.M. **Estudo e Implementação de um Array Reconfigurável para um processador MIPS**. Trabalho de Diplomação (Curso de Engenharia da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

GUPTA, R. K.; MICHELI, G. D. Hardware-software co-synthesis for digital systems. **IEEE Design and Test of Computers**, Santa Barbara, v. 10, n. 3, p. 29 – 41, September 1993.

GUTHAUS, M.R. et al. MiBench: A Free, Commercially Representative Embedded Benchmark Suite. In: WORKSHOP ON WORKLOAD CHARACTERIZATION, 2001, Austin. **Proceedings…** Washington: IEEE Computer Society, 2001. p. 3 – 14.

GOODWIN, D.; PETKOV, D. Automatic generation of application specific processors. In: INTERNATIONAL CONFERENCE ON COMPILERS, ARCHITECTURE, AND SYNTHESIS FOR EMBEDDED SYSTEMS, 2003, San Jose. **Proceedings…** New York: ACM Press, 2003. p. 137–147.

GRAY, J. et al. VIPER: A 25MHz, 100 MIPS Peak VLIW Microprocessor. In: CUSTOM INTEGRATED CIRCUITS CONFERENCE, 1993, San Diego. **Proceedings…** Washington: IEEE Computer Society, 1993. p. 4.1.1 – 4.1.5.

HAUCK, S. et al. The Chimaera reconfigurable functional unit. In: FPGA-BASED CUSTOM COMPUTING MACHINES, 1997, Napa Valley. **Proceedings**… Washington: IEEE Computer Society, 1997. p. 87 – 96.

HAUSER, J. R.; WAWRZYNEK J. Garp: A MIPS Processor with a Reconfigurable Coprocessor. In: FPGA-BASED CUSTOM COMPUTING MACHINES, 1997, Napa Valley. **Proceedings**… Washington: IEEE Computer Society, 1997. 12-21.

HENNESSY J. L.; PATTERSON, D. A. **Computer Architecture**: A Quantitative Approach. San Mateo: Morgan Kaufmamn, 1990.

HENKEL, J.; ERNST, R. A Hardware/Software Partitioner using a Dynamically Determined Granularity. In: DESIGN AUTOMATION CONFERENCE, DAC, 42., 2005, Anaheim. **Proceedings…** New York: ACM Press, 2005. p. 732 – 737.

HENKEL, J. A low power hardware/software partitioning approach for core-based embedded systems. In: DESIGN AUTOMATION CONFERENCE, DAC, 36., 1999, Anaheim. **Proceedings…** New York: ACM Press, 2005. p. 122 – 127.

HIRATA, H et al. An Elementary Processor Architecture with Simultaneous Instruction Issuing from Multiple Threads. **ACM SIGARCH Computer Architecture News**, New York, v. 20, n. 2, p. 136 – 145, May 1992.

HUANG, I.; DESPAIN, A.M. Generating Instruction Set and Microarchitecture from Applications. In: INTERNATIONAL CONFERENCE ON COMPUTER AIDED DESIGN, 1994, San Jose. **Proceedings…** Los Alamitos: IEEE Computer Society, 1994. p. 391 – 396.

HUTCHINGS, B. L. Exploiting reconfigurability through domain-specific systems. In: INTERNATIONAL WORKSHOP ON FIELD-PROGRAMMABLE LOGIC AND APPLICATIONS, FPL, 7., 1997, London. **Proceedings…** Berlin: Springer, 1997. p.193 – 202. (Lecture Notes in Computer Science, v.1304).

HUTCHINGS, B. L. et al. A CAD suite for high-performance FPGA design. In: SYMPOSIUM ON FIELD-PROGRAMMABLE CUSTOM COMPUTING MACHINES, 1999, Napa Valley. **Proceedings…** Washington: IEEE Computer Society, 1999. p. 12–24.

JAIN, M. K.; BALAKRISHNAN, M.; KUMAR, A. ASIP design methodologies: Survey and issues. In: INTERNATIONAL CONFERENCE ON VLSI DESIGN, VLSID, 2001, Bangalore. **Proceedings...** Washington: IEEE Computer Society, 2001. p. 76 – 81.

JEJURIKAR, R.; PEREIRA, C.; GUPTA, R. Leakage aware dynamic voltage scaling for real-time embedded systems. In: DESIGN AUTOMATION CONFERENCE, DAC, 41., 2004, San Diego. **Proceedings…** New York: ACM Press, 2004. p. 275 – 280.

KOWALCZYK, A. et al. The first MAJC microprocessor: A dual CPU Systems-onchip, **IEEE Journal of Solid-State Circuits**, San Francisco, v. 36, n. 11, p. 1609- 1616, November 2001.

KUÇUKÇAKAR, K. An ASIP Design Methodology for Embedded Systems. In: INTERNATIONAL CONFERENCE ON HARDWARE SOFTWARE CODESIGN, CODES, 1999, Rome. **Proceedings…** New York: ACM Press, 1999. p. 17 – 21.

LANDMAN, P.; RABAEY, J. Activity-sensitive architectural power analysis for the control path. In: INTERNATIONAL SYMPOSIUM ON LOW POWER ELETRONICS AND DESIGN, ISLPD, 1995, Dana Point. **Proceedings...** New York: ACM Press, 1995. p. 93 – 98.

LYSECKY, R.; STITT, G.; VAHID, F. Warp Processors. **ACM Transactions on Design Automation of Electronic Systems**, New York, v.11, n. 3, p. 659 – 681, July 2006.

NAKAMURA, K.; SAKAI, K.; AE, T. Real-Time Multimedia Data Processing using VLIW Hardware Stack Processor. In: WORKSHOP ON PARALLEL AND

DISTRIBUTED REAL TIME SYSTEMS, WPDRTS, 1995. **Proceedings...** Washington: IEEE Computer Society, 1995. p. 84 – 89.

OVIEDO, E.I. Control flow, data flow and program complexity. In: INTERNATIONAL COMPUTER SOFTWARE AND APPLICATIONS CONFERENCE, COMPSAC, 1980, New York. **Proceedings...** Washington: IEEE Computer Society, 1980. p. 146 – 152.

OR-BACH, Z. Panel: (when) will FPGAs kill ASICs? In: DESIGN AUTOMATION CONFERENCE, DAC, 38., 2001, Las Vegas. **Proceedings...** New York: ACM Press, 2001. p. 321 – 322.

PALACHARLA, S.; JOUPPI, N.; SMITH, J. E. Complexity-effective superscalar processors. **ACM SIGARCH Computer Architecture News**, New York, v. 25, n. 2, p. 206 – 218, May 1997.

RIGO, S. et al. ArchC: A systemc-based architecture description language. In: SYMPOSIUM ON COMPUTER ARCHITECTURE AND HIGH PERFORMANCE COMPUTING, SBAC-PAD, 2004, Foz do Iguacu. **Proceedings...** Washington: IEEE Computer Society, 2004. p. 66 – 73.

RUTZIG, M.; BECK FILHO, A.C.S.; CARRO, L. Balancing Reconfigurable Data Path Resources According to Applications Requirements. Aceito para RAW, 2008, Miami.

SANKARALINGAM, K. et al. Trips: A polymorphous architecture for exploiting ilp, tlp, and dlp. **ACM Transactions on Architecture and Code Optimization**, New York, v.1, n. 1, p. 62 – 93, May 2004.

SHI, K.; HOWARD, D. Challenges in Sleep Transistor Design and Implementation in Low-Power Designs. In: DESIGN AUTOMATION CONFERENCE, DAC, 43., 2006, San Francisco. **Proceedings...** New York: ACM Press, 2006. p. 113 – 116.

SMITH, J. E. A study of branch prediction strategies. In: INTERNATIONAL SYMPOSIUM ON COMPUTER ARCHITECTURE, 8., 1981, Minneapolis. **Proceedings…** Los Alamitos: IEEE Computer Society, 1981. p. 135 – 148.

STITT, G.; VAHID, F. The Energy Advantages of Microprocessor Platforms with On-Chip Configurable Logic, **IEEE Design and Test of Computers**, Los Alamitos, v. 19, n. 6, p. 36 – 43, November 2002.

SWANSON, S. et al, The WaveScalar Architecture. **ACM Transactions on Computer Systems**, New York, v.25, n. 2, May 2007.

SUGA, A.; MATSUNAMI, K. Introducing the FR500 embedded microprocessor. **IEEE Micro**, Los Alamitos, v.20, n. 4, p. 21 – 27, July 2000.

TELLEZ, E.; FARRAH, A.; SARRAFZADEH, M. Activity-driven clock design for lowpower circuits. In: INTERNATIONAL CONFERENCE ON COMPUTER AIDED DESIGN, 1995, San Jose. **Proceedings…** Washington: IEEE Computer Society, 1995. p. 62 – 65.

TSCHANZ, J. et al. Dynamic-sleep transistor and body bias for active leakage power control of microprocessors. **IEEE Journal of Solid-State Circuits**, San Francisco, v. 38, n. 11, p. 1838 – 1845, November 2003.

VASSILIADIS, S.; WONG, S.; COTOFANA, S. The MOLEN ρμ-coded processor. In: INTERNATIONAL CONFERENCE ON FIELD-PROGRAMMABLE LOGIC AND APPLICATIONS, FPL, 11., 2001, Belfast. **Proceedings…** Berlin: Springer, 2001. p. 275 – 285. (Lecture Notes in Computer Science, v. 2147)

VASSILIADIS, S. **The Hipeac Embedded Systems.** Apresentação oral no Hipeac Compilation Architecture, May 2006.

VENKATARAMANI, G. et al. A Compiler Framework for Mapping Applications to a Coarse-grained Reconfigurable Computer Architecture. In: INTERNATIONAL CONFERENCE ON COMPILERS, ARCHITECTURE AND SYNTHESIS FOR EMBEDDED SYSTEMS, 2001, Atlanta. **Proceedings…** New York: ACM Press, 2001. p. 116 – 125.

XU, B.; ALBONESI, D.H. Methodology for the analysis of dynamic application parallelism and its application to reconfigurable computing. In: INTERNATIONAL SYMPOSIUM ON VOICE, VIDEO AND DATA COMMUNICATIONS, 1999, Bellingham. **Proceedings…** Bellingham: SPIE, 1999. p. 78-86.

WALL, D. W. Limits of instruction-level parallelism. In: INTERNATIONAL CONFERENCE ON ARCHITECTURAL SUPPORT FOR PROGRAMMING LANGUAGES AND OPERATING SYSTEMS, 1991, Santa Clara. **Proceedings...** New York: ACM Press, 1991. p.176 – 188.

WAN, M. et al. An Energy Conscious Methodology for Early Design Space Exploration of Heterogeneous DSPs. In: CUSTOM INTEGRATED CIRCUITS CONFERENCE, 1998, Santa Clara. **Proceedings...** Washington: IEEE Computer Society, 1998. p. 111 – 117.

WILCOX, K.; MANNE, S. Alpha processors: A history of power issues and a look to the future. In: SYMPOSIUM ON LOW POWER AND HIGH SPEED CHIPS, 1999, Kyoto. **Proceedings...** [S.l.: s.n.], 1999.

YAMAMOTO, W. et al. Performance Estimation of Multistreamed, Superscalar Processors. In: HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES, 1994, Wailea. **Proceedings...** Washington: IEEE Computer Society, 1994. p. 195 – 204.

YEAGER, K.C. The Mips R10000 Superscalar Microprocessor. **IEEE Micro**, New York, v. 16, n. 2, p. 28-40, April 1996.

APÊNDICE A RESULTADOS DE EXPLORAÇÃO DE RECURSOS

- **Sem Especulação**

Tabela 5.1: Aceleração de todas as aplicações nas metodologias de exploração de recursos (Modelo 2)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	1,000	1,000	1,000	1,607	1,607	1,607	1,607	1,607	1,607	1,607	1,607	1,607
bitcount	1,763	1,763	1,763	1,764	1,764	1,764	1,764	1,764	1,764	1,764	1,764	1,764
corners	1,295	1,295	1,295	1,642	1,642	1,642	2,172	2,174	2,174	2,172	2,174	2,174
CRC	1,528	1,528	1,528	1,534	1,534	1,534	1,534	1,534	1,534	1,534	1,534	1,534
djikstra	1,558	1,558	1,558	1,644	1,644	1,644	1,721	1,721	1,721	1,721	1,721	1,721
edges	1,374	1,378	1,378	1,567	1,572	1,572	2,189	2,200	2,200	2,189	2,200	2,200
gsmd	1,478	2,029	1,705	1,562	2,191	1,817	1,588	2,267	1,856	1,613	2,331	1,892
gsme	1,396	1,396	1,396	1,633	1,633	1,633	1,681	1,687	1,687	1,681	1,687	1,687
jpegd	1,757	1,757	1,757	2,122	2,122	2,122	2,216	2,217	2,217	2,216	2,217	2,217
jpege	2,255	2,255	2,255	2,595	2,595	2,595	2,771	2,771	2,771	2,771	2,771	2,771
pcm	1,000	1,000	1,000	1,642	1,642	1,642	1,642	1,642	1,642	1,642	1,642	1,642
qsort	1,212	1,212	1,212	1,514	1,514	1,514	1,767	1,767	1,767	1,767	1,767	1,767
rijndaeld	1,052	1,052	1,052	1,052	1,052	1,052	1,653	1,682	1,682	1,653	1,682	1,682
sha	1,724	1,724	1,724	1,906	1,906	1,906	1,909	1,909	1,909	1,909	1,909	1,909
smoothing	1,477	1,477	1,477	1,550	1,550	1,550	1,652	1,652	1,652	1,652	1,652	1,652
string	1,353	1,353	1,353	1,417	1,417	1,417	1,892	1,894	1,894	1,892	1,894	1,894

Tabela 5.2: Aceleração de todas as aplicações nas metodologias de exploração de recursos (Modelo 3)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	1,000	1,000	1,000	1,607	1,607	1,607	1,607	1,607	1,607	1,607	1,607	1,607
bitcount	1,763	1,763	1,763	1,764	1,764	1,764	1,764	1,764	1,764	1,764	1,764	1,764
corners	1,295	1,295	1,295	1,642	1,642	1,642	2,172	2,174	2,174	2,172	2,174	2,174
CRC	1,528	1,528	1,528	1,534	1,534	1,534	1,534	1,534	1,534	1,534	1,534	1,534
djikstra	1,558	1,558	1,558	1,644	1,644	1,644	1,721	1,721	1,721	1,721	1,721	1,721
edges	1,374	1,378	1,378	1,567	1,572	1,572	2,199	2,210	2,210	2,199	2,210	2,210
gsmd	2,532	2,699	2,699	2,789	2,993	2,993	2,930	3,165	3,156	3,045	3,303	3,289
gsme	1,396	1,396	1,396	1,633	1,633	1,633	1,693	1,695	1,695	1,693	1,695	1,695
jpegd	1,757	1,757	1,757	2,122	2,122	2,122	2,216	2,217	2,217	2,216	2,217	2,217
jpege	2,255	2,255	2,255	2,595	2,595	2,595	2,771	2,771	2,771	2,771	2,771	2,771
pcm	1,000	1,000	1,000	1,642	1,642	1,642	1,642	1,642	1,642	1,642	1,642	1,642
qsort	1,212	1,212	1,212	1,514	1,514	1,514	1,767	1,767	1,767	1,767	1,767	1,767
rijndaeld	1,052	1,052	1,052	1,052	1,052	1,052	3,352	3,430	3,430	3,352	3,430	3,430
sha	1,724	1,724	1,724	1,906	1,906	1,906	1,909	1,909	1,909	1,909	1,909	1,909
smoothing	1,477	1,477	1,477	1,550	1,550	1,550	1,652	1,652	1,652	1,652	1,652	1,652
string	1,353	1,353	1,353	1,417	1,417	1,417	1,892	1,894	1,894	1,892	1,894	1,894

Tabela 5.3: Aceleração de todas as aplicações nas metodologias de exploração de recursos (Modelo 4)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	1,000	1,000	1,000	1,607	1,607	1,607	1,607	1,607	1,607	1,607	1,607	1,607
bitcount	1,763	1,763	1,763	1,764	1,764	1,764	1,764	1,764	1,764	1,764	1,764	1,764
corners	1,295	1,295	1,295	1,642	1,642	1,642	2,174	2,174	2,174	2,174	2,174	2,174
CRC	1,528	1,528	1,528	1,534	1,534	1,534	1,534	1,534	1,534	1,534	1,534	1,534
djikstra	1,558	1,558	1,558	1,644	1,644	1,644	1,721	1,721	1,721	1,721	1,721	1,721
edges	1,378	1,378	1,378	1,572	1,572	1,572	2,210	2,210	2,210	2,210	2,210	2,210
gsmd	2,699	2,699	2,699	2,993	2,993	2,993	3,168	3,168	3,156	3,308	3,308	3,289
gsme	1,396	1,396	1,396	1,633	1,633	1,633	1,695	1,695	1,695	1,695	1,695	1,695
jpegd	1,757	1,757	1,757	2,122	2,122	2,122	2,217	2,217	2,217	2,217	2,217	2,217
jpege	2,255	2,255	2,255	2,595	2,595	2,595	2,771	2,771	2,771	2,771	2,771	2,771
pcm	1,000	1,000	1,000	1,642	1,642	1,642	1,642	1,642	1,642	1,642	1,642	1,642
qsort	1,212	1,212	1,212	1,514	1,514	1,514	1,767	1,767	1,767	1,767	1,767	1,767
rijndaeld	1,052	1,052	1,052	1,052	1,052	1,052	4,577	4,577	4,577	4,577	4,577	4,577
sha	1,724	1,724	1,724	1,906	1,906	1,906	1,909	1,909	1,909	1,909	1,909	1,909
smoothing	1,477	1,477	1,477	1,550	1,550	1,550	1,652	1,652	1,652	1,652	1,652	1,652
string	1,353	1,353	1,353	1,417	1,417	1,417	1,894	1,894	1,894	1,894	1,894	1,894

Tabela 5.4: Potência (Watts) de todas as aplicações nas metodologias de exploração de recursos (Modelo 2)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	1.413	1.413	1.413	4.312	5.147	4.536	4.874	5.709	5.098	5.516	6.351	5.741
bitcount	3.715	4.422	3.905	3.776	4.484	3.966	4.254	4.962	4.444	4.798	5.506	4.989
corners	2.041	2.161	2.073	2.676	2.965	2.754	4.223	4.838	4.391	4.694	5.310	4.863
CRC	3.511	4.129	3.677	3.585	4.210	3.753	4.006	4.630	4.173	4.486	5.111	4.654
djikstra	3.599	4.229	3.769	3.931	4.645	4.123	4.697	5.483	4.908	5.301	6.086	5.512
edges	2.228	2.417	2.283	2.675	2.982	2.763	4.369	5.048	4.567	4.873	5.555	5.074
gsmd	2.135	2.644	2.315	2.256	2.903	2.479	2.415	3.170	2.672	2.599	3.473	2.896
gsme	2.515	2.850	2.605	3.221	3.766	3.367	3.716	4.308	3.881	4.163	4.756	4.329
jpegd	3.096	3.609	3.234	3.789	4.530	3.989	4.070	4.879	4.289	4.164	4.973	4.383
jpege	4.516	4.866	4.610	5.358	5.811	5.479	8.003	8.518	8.142	10.633	11.149	10.772
pcm	1.435	1.435	1.435	3.759	4.452	3.945	4.226	4.919	4.412	4.759	5.452	4.945
qsort	2.323	2.492	2.368	3.439	3.911	3.566	4.806	5.526	4.999	5.360	6.081	5.554
rijndaeld	1.853	1.884	1.861	1.864	1.895	1.872	2.549	2.729	2.613	2.695	2.878	2.762
sha	2.092	2.401	2.175	2.248	2.629	2.351	2.509	2.891	2.611	2.803	3.184	2.905
smoothing	1.857	2.115	1.926	1.997	2.300	2.078	2.426	2.797	2.526	2.710	3.081	2.811
string	3.117	3.494	3.218	3.420	3.868	3.541	5.704	6.622	5.956	6.408	7.326	6.660

Tabela 5.5: Potência (Watts) de todas as aplicações nas metodologias de exploração de recursos (Modelo 3)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	1.413	1.413	1.413	16.863	10.060	9.246	17.426	10.622	9.808	18.068	11.265	10.451
bitcount	14.339	8.581	7.892	14.405	8.644	7.955	14.890	9.125	8.435	15.434	9.669	8.980
corners	3.857	2.873	2.755	7.016	4.664	4.383	13.394	8.432	7.836	13.869	8.905	8.309
CRC	12.799	8.810	7.162	12.974	8.941	7.276	13.395	9.362	7.697	13.876	9.843	8.177
djikstra	13.068	7.936	7.322	14.654	8.842	8.147	16.497	10.101	9.336	17.100	10.705	9.940
edges	4.986	3.500	3.320	7.170	4.747	4.455	14.267	8.952	8.308	14.775	9.462	8.818
gsmd	6.195	4.537	4.290	7.753	5.488	5.149	8.493	6.087	5.706	9.266	6.726	6.307
gsme	7.550	4.821	4.494	11.411	6.972	6.441	12.516	7.765	7.193	12.965	8.216	7.643
jpegd	10.799	6.624	6.124	14.900	8.881	8.160	16.188	9.624	8.838	16.284	9.719	8.933
jpege	9.777	6.926	6.584	12.162	8.474	8.033	15.737	11.546	11.044	18.368	14.177	13.675
pcm	1.435	1.435	1.435	14.177	8.530	7.855	14.644	8.997	8.321	15.177	9.530	8.855
qsort	4.869	3.489	3.324	10.541	6.691	6.231	15.636	9.766	9.063	16.190	10.320	9.617
rijndaeld	2.325	2.069	2.038	2.339	2.081	2.050	8.454	5.991	5.676	8.751	6.295	5.980
sha	6.746	4.223	3.921	7.965	4.867	4.496	8.247	5.137	4.765	8.541	5.431	5.058
smoothing	5.741	3.636	3.384	6.549	4.082	3.787	7.985	4.973	4.613	8.270	5.258	4.898
string	8.790	5.715	5.347	10.144	6.500	6.064	19.377	11.980	11.092	20.080	12.684	11.796

Tabela 5.6: Potência (Watts) de todas as aplicações nas metodologias de exploração de recursos (Modelo 4)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	1.414	1.413	1.413	98.964	11.472	10.599	99.528	12.034	11.161	100.170	12.677	11.804
bitcount	83.830	9.776	9.037	83.928	9.840	9.101	84.421	10.322	9.582	85.005	10.866	10.126
corners	15.740	3.077	2.951	35.408	5.153	4.851	73.462	9.465	8.826	73.954	9.938	9.299
CRC	73.558	8.810	8.163	74.389	8.941	8.288	74.813	9.362	8.709	75.294	9.843	9.190
djikstra	75.007	9.001	8.343	84.798	10.048	9.303	93.679	11.429	10.608	94.283	12.033	11.212
edges	23.083	3.811	3.618	36.682	5.254	4.941	79.299	10.070	9.379	79.815	10.579	9.889
gsmd	31.542	4.966	4.701	42.506	6.076	5.712	46.456	6.734	6.318	50.249	7.425	6.966
gsme	40.487	5.387	5.037	64.983	7.893	7.324	70.082	8.755	8.141	70.532	9.206	8.591
jpegd	61.185	7.491	6.955	87.614	10.132	9.358	95.496	10.988	10.145	95.605	11.083	10.240
jpege	44.192	7.518	7.152	56.674	9.240	8.766	66.330	12.416	11.878	68.970	15.047	14.509
pcm	1.436	1.435	1.435	82.323	9.702	8.978	82.792	10.169	9.445	83.325	10.702	9.978
qsort	21.525	3.775	3.598	56.998	7.490	6.996	86.476	10.984	10.231	87.030	11.538	10.785
rijndaeld	5.414	2.122	2.089	5.446	2.135	2.102	53.396	8.251	7.801	53.801	8.657	8.206
sha	37.192	4.747	4.423	45.362	5.510	5.112	45.782	5.782	5.383	46.076	6.076	5.677
smoothing	31.147	4.073	3.803	36.330	4.594	4.277	44.364	5.599	5.212	44.651	5.884	5.497
string	45.897	6.353	5.958	54.123	7.256	6.788	108.937	13.520	12.568	109.641	14.224	13.272

Tabela 5.7: Energia (Joules) de todas as aplicações nas metodologias de exploração de recursos (Modelo 2)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	0,245	0,245	0,245	0,465	0,555	0,489	0,525	0,615	0,549	0,594	0,684	0,618
bitcount	0,480	0,572	0,505	0,488	0,580	0,513	0,550	0,641	0,574	0,620	0,712	0,645
corners	0,027	0,029	0,028	0,028	0,031	0,029	0,034	0,038	0,035	0,037	0,042	0,039
CRC	0,363	0,428	0,381	0,370	0,434	0,387	0,413	0,477	0,430	0,463	0,527	0,480
djikstra	0,685	0,805	0,718	0,710	0,838	0,744	0,810	0,945	0,846	0,914	1,049	0,950
edges	0,056	0,060	0,057	0,059	0,065	0,061	0,069	0,079	0,072	0,077	0,087	0,079
gsmd	0,236	0,213	0,222	0,236	0,216	0,223	0,248	0,228	0,235	0,263	0,243	0,250
gsme	0,087	0,098	0,090	0,095	0,111	0,099	0,106	0,123	0,111	0,119	0,135	0,123
jpegd	0,260	0,303	0,271	0,263	0,315	0,277	0,271	0,324	0,285	0,277	0,331	0,291
jpege	0,087	0,094	0,089	0,090	0,097	0,092	0,126	0,134	0,128	0,167	0,175	0,169
pcm	0,196	0,196	0,196	0,312	0,369	0,327	0,351	0,408	0,366	0,395	0,452	0,410
qsort	0,138	0,148	0,141	0,164	0,186	0,170	0,196	0,225	0,204	0,219	0,248	0,227
rijndaeld	0,305	0,311	0,307	0,307	0,312	0,309	0,267	0,281	0,269	0,283	0,297	0,285
sha	0,079	0,091	0,082	0,077	0,090	0,080	0,086	0,099	0,089	0,096	0,109	0,099
smoothing	0,222	0,253	0,230	0,228	0,262	0,237	0,259	0,299	0,270	0,290	0,329	0,300
string	0,003	0,004	0,003	0,003	0,004	0,003	0,004	0,005	0,004	0,005	0,005	0,005

Tabela 5.8: Energia (Joules) de todas as aplicações nas metodologias de exploração de recursos (Modelo 3)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	0,245	0,245	0,245	1,817	1,084	0,996	1,877	1,144	1,057	1,947	1,214	1,126
bitcount	1,854	1,109	1,020	1,862	1,117	1,028	1,924	1,179	1,090	1,994	1,250	1,160
corners	0,052	0,038	0,037	0,074	0,049	0,046	0,107	0,067	0,062	0,110	0,071	0,066
CRC	1,325	0,912	0,742	1,338	0,922	0,750	1,381	0,965	0,794	1,431	1,015	0,843
djikstra	2,488	1,511	1,394	2,645	1,596	1,471	2,844	1,742	1,610	2,948	1,846	1,714
edges	0,125	0,087	0,083	0,158	0,104	0,098	0,223	0,140	0,129	0,231	0,147	0,137
gsmd	0,400	0,275	0,260	0,454	0,299	0,281	0,473	0,314	0,295	0,497	0,333	0,313
gsme	0,260	0,166	0,155	0,336	0,205	0,190	0,355	0,220	0,204	0,368	0,233	0,217
jpegd	0,906	0,556	0,514	1,035	0,617	0,567	1,077	0,640	0,588	1,083	0,646	0,594
jpege	0,189	0,134	0,127	0,204	0,142	0,135	0,247	0,181	0,173	0,288	0,222	0,215
pcm	0,196	0,196	0,196	1,176	0,708	0,652	1,215	0,747	0,690	1,259	0,791	0,735
qsort	0,289	0,207	0,198	0,502	0,318	0,296	0,638	0,398	0,370	0,660	0,421	0,392
rijndaeld	0,383	0,341	0,336	0,385	0,343	0,338	0,437	0,303	0,287	0,453	0,318	0,302
sha	0,255	0,160	0,148	0,272	0,166	0,154	0,282	0,175	0,163	0,292	0,185	0,173
smoothing	0,686	0,435	0,405	0,746	0,465	0,431	0,854	0,532	0,493	0,884	0,562	0,523
string	0,009	0,006	0,006	0,010	0,006	0,006	0,014	0,009	0,008	0,015	0,009	0,009

Tabela 5.9: Energia (Joules) de todas as aplicações nas metodologias de exploração de recursos (Modelo 4)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	0,245	0,245	0,245	10,662	1,236	1,142	10,723	1,297	1,203	10,792	1,366	1,272
bitcount	10,838	1,264	1,168	10,849	1,272	1,176	10,914	1,334	1,238	10,984	1,404	1,309
corners	0,210	0,041	0,039	0,373	0,054	0,051	0,584	0,075	0,070	0,588	0,079	0,074
CRC	7,617	0,912	0,845	7,670	0,922	0,855	7,714	0,965	0,898	7,763	1,015	0,948
djikstra	14,283	1,714	1,589	15,306	1,814	1,679	16,151	1,970	1,829	16,255	2,075	1,933
edges	0,577	0,095	0,090	0,804	0,115	0,108	1,236	0,157	0,146	1,244	0,165	0,154
gsmd	1,909	0,300	0,284	2,320	0,332	0,312	2,395	0,347	0,327	2,481	0,367	0,346
gsme	1,394	0,186	0,173	1,913	0,232	0,216	1,987	0,248	0,231	2,000	0,261	0,244
jpegd	5,133	0,628	0,583	6,084	0,703	0,650	6,349	0,731	0,675	6,356	0,737	0,681
jpege	0,852	0,145	0,138	0,950	0,155	0,147	1,041	0,195	0,186	1,082	0,236	0,228
pcm	0,196	0,196	0,196	6,831	0,805	0,745	6,870	0,844	0,784	6,914	0,888	0,828
qsort	1,280	0,224	0,214	2,712	0,356	0,333	3,527	0,448	0,417	3,550	0,471	0,440
rijndaeld	0,892	0,350	0,344	0,898	0,352	0,346	2,023	0,313	0,296	2,039	0,328	0,311
sha	1,406	0,179	0,167	1,551	0,188	0,175	1,563	0,197	0,184	1,573	0,207	0,194
smoothing	3,724	0,487	0,455	4,140	0,523	0,487	4,741	0,598	0,557	4,772	0,629	0,587
string	0,047	0,007	0,006	0,053	0,007	0,007	0,080	0,010	0,009	0,081	0,011	0,010

- **Com Especulação**

Tabela 5.10: Aceleração de todas as aplicações nas metodologias de exploração de recursos (Modelo 2)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	1,000	1,000	1,000	1,908	1,908	1,908	1,910	1,910	1,910	1,910	1,910	1,910
bitcount	1,923	1,923	1,923	1,924	1,924	1,924	1,924	1,924	1,924	1,924	1,924	1,924
corners	1,271	1,271	1,271	1,733	1,733	1,733	2,628	2,629	2,629	2,629	2,630	2,630
CRC	1,985	1,985	1,985	2,003	2,003	2,003	2,003	2,003	2,003	2,003	2,003	2,003
djikstra	1,866	1,866	1,866	2,163	2,163	2,163	2,182	2,183	2,183	2,182	2,183	2,183
edges	1,360	1,351	1,351	1,627	1,618	1,618	2,473	2,457	2,457	2,475	2,459	2,459
gsmd	1,228	1,234	1,234	1,264	1,271	1,270	1,276	1,283	1,282	1,291	1,301	1,300
gsme	1,814	1,814	1,814	2,060	2,060	2,060	2,132	2,132	2,132	2,132	2,132	2,132
jpegd	1,801	1,801	1,801	2,440	2,491	2,491	2,528	2,583	2,583	2,529	2,585	2,585
jpege	2,275	2,560	2,275	2,644	3,240	2,645	2,918	3,737	2,919	2,918	3,737	2,920
pcm	1,110	1,110	1,110	1,993	1,993	1,993	1,995	1,995	1,995	1,995	1,995	1,995
qsort	1,289	1,289	1,289	2,304	2,307	2,307	2,450	2,454	2,454	2,451	2,455	2,455
rijndaeld	1,053	1,053	1,053	1,445	1,457	1,457	1,458	1,470	1,470	1,458	1,470	1,470
sha	2,607	2,609	2,609	4,132	4,144	4,144	4,160	4,173	4,173	4,160	4,173	4,173
smoothing	2,261	2,261	2,261	2,644	2,644	2,644	2,941	2,941	2,941	2,941	2,941	2,941
string	1,471	1,471	1,471	1,617	1,617	1,617	2,686	2,689	2,689	2,686	2,689	2,689

Tabela 5.11: Aceleração de todas as aplicações nas metodologias de exploração de recursos (Modelo 3)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	1,000	1,000	1,000	1,908	1,908	1,908	1,910	1,910	1,910	1,910	1,910	1,910
bitcount	1,923	1,923	1,923	1,924	1,924	1,924	1,924	1,924	1,924	1,924	1,924	1,924
corners	1,271	1,271	1,271	1,733	1,733	1,733	2,628	2,629	2,629	2,629	2,630	2,630
CRC	1,985	1,985	1,985	2,003	2,003	2,003	2,003	2,003	2,003	2,003	2,003	2,003
djikstra	1,866	1,866	1,866	2,163	2,163	2,163	2,182	2,183	2,183	2,182	2,183	2,183
edges	1,348	1,351	1,351	1,614	1,619	1,619	2,547	2,560	2,560	2,549	2,562	2,562
gsmd	2,745	2,937	2,937	2,933	3,154	3,154	3,099	3,353	3,352	3,243	3,524	3,524
gsme	1,814	1,814	1,814	2,060	2,060	2,060	2,130	2,165	2,164	2,130	2,165	2,164
jpegd	1,801	1,801	1,801	2,490	2,491	2,491	2,582	2,583	2,583	2,584	2,585	2,585
jpege	2,560	2,560	2,560	3,239	3,240	3,240	3,734	3,737	3,737	3,735	3,737	3,737
pcm	1,110	1,110	1,110	1,993	1,993	1,993	1,995	1,995	1,995	1,995	1,995	1,995
qsort	1,289	1,289	1,289	2,304	2,307	2,307	2,450	2,454	2,454	2,451	2,455	2,455
rijndaeld	1,053	1,053	1,053	1,532	1,534	1,534	1,548	1,551	1,551	1,548	1,551	1,551
sha	2,607	2,609	2,609	4,132	4,144	4,144	4,160	4,173	4,173	4,160	4,173	4,173
smoothing	2,261	2,261	2,261	2,644	2,644	2,644	2,941	2,941	2,941	2,941	2,941	2,941
string	1,471	1,471	1,471	1,617	1,617	1,617	2,686	2,689	2,689	2,686	2,689	2,689

Tabela 5.12: Aceleração de todas as aplicações nas metodologias de exploração de recursos (Modelo 5)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	1,000	1,000	1,000	1,908	1,908	1,908	1,910	1,910	1,910	1,910	1,910	1,910
bitcount	1,923	1,923	1,923	1,924	1,924	1,924	1,924	1,924	1,924	1,924	1,924	1,924
corners	1,271	1,271	1,271	1,733	1,733	1,733	2,629	2,629	2,629	2,630	2,630	2,630
CRC	1,985	1,985	1,985	2,003	2,003	2,003	2,003	2,003	2,003	2,003	2,003	2,003
djikstra	1,866	1,866	1,866	2,163	2,163	2,163	2,183	2,183	2,183	2,183	2,183	2,183
edges	1,351	1,351	1,351	1,619	1,619	1,619	2,560	2,560	2,560	2,562	2,562	2,562
gsmd	2,937	2,937	2,937	3,154	3,154	3,154	3,353	3,353	3,352	3,524	3,524	3,524
gsme	1,814	1,814	1,814	2,060	2,060	2,060	2,165	2,165	2,164	2,165	2,165	2,164
jpegd	1,801	1,801	1,801	2,491	2,491	2,491	2,583	2,583	2,583	2,585	2,585	2,585
jpege	2,560	2,560	2,560	3,240	3,240	3,240	3,737	3,737	3,737	3,737	3,737	3,737
pcm	1,110	1,110	1,110	1,993	1,993	1,993	1,995	1,995	1,995	1,995	1,995	1,995
qsort	1,289	1,289	1,289	2,307	2,307	2,307	2,454	2,454	2,454	2,455	2,455	2,455
rijndaeld	1,053	1,053	1,053	5,663	5,663	5,663	6,162	6,162	6,162	6,162	6,162	6,162
sha	2,609	2,609	2,609	4,144	4,144	4,144	4,173	4,173	4,173	4,173	4,173	4,173
smoothing	2,261	2,261	2,261	2,644	2,644	2,644	2,941	2,941	2,941	2,941	2,941	2,941
string	1,471	1,471	1,471	1,617	1,617	1,617	2,689	2,689	2,689	2,689	2,689	2,689

Tabela 5.13: Potência (Watts) de todas as aplicações nas metodologias de exploração de recursos (Modelo 2)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	1.410	1.410	1.410	3.504	4.516	3.933	3.852	4.866	4.282	4.245	5.260	4.676
bitcount	3.179	4.071	3.557	3.226	4.119	3.605	3.596	4.489	3.975	4.016	4.910	4.396
corners	1.840	1.939	1.882	2.360	2.683	2.497	3.632	4.400	3.959	3.948	4.716	4.274
CRC	2.550	3.156	2.807	2.592	3.210	2.854	2.776	3.393	3.038	2.985	3.602	3.247
djikstra	2.428	2.998	2.670	2.771	3.529	3.093	3.024	3.795	3.352	3.288	4.058	3.616
edges	1.916	2.071	1.982	2.279	2.588	2.410	3.505	4.248	3.817	3.815	4.557	4.125
gsmd	1.834	1.966	1.894	1.878	2.032	1.948	1.949	2.110	2.022	2.035	2.206	2.114
gsme	2.336	2.895	2.573	2.602	3.329	2.910	2.918	3.691	3.246	3.203	3.976	3.531
jpegd	2.023	2.435	2.198	2.413	3.151	2.740	2.505	3.293	2.855	2.544	3.334	2.896
jpege	4.006	4.650	4.120	4.552	5.733	4.706	5.787	7.608	5.973	6.793	8.897	6.980
pcm	1.757	1.905	1.820	3.745	4.805	4.195	4.183	5.245	4.634	4.680	5.742	5.131
qsort	2.171	2.341	2.243	3.926	4.815	4.306	4.503	5.491	4.926	4.891	5.880	5.315
rijndaeld	1.792	1.812	1.800	2.087	2.218	2.147	2.147	2.282	2.209	2.200	2.336	2.263
sha	2.193	2.521	2.333	2.779	3.421	3.057	2.985	3.634	3.266	3.205	3.855	3.487
smoothing	1.961	2.275	2.094	2.235	2.703	2.433	2.616	3.197	2.863	2.835	3.417	3.082
string	2.448	2.766	2.583	2.759	3.197	2.945	4.925	6.118	5.434	5.369	6.562	5.879

Tabela 5.14: Potência (Watts) de todas as aplicações nas metodologias de exploração de recursos (Modelo 3)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	1.410	1.410	1.410	13.570	8.254	7.710	13.938	8.612	8.066	14.331	9.005	8.459
bitcount	12.051	7.365	6.885	12.103	7.415	6.935	12.480	7.788	7.307	12.900	8.209	7.728
corners	2.826	2.305	2.252	5.564	3.872	3.699	11.243	7.227	6.815	11.563	7.544	7.132
CRC	8.575	5.393	5.067	8.728	5.488	5.156	8.912	5.671	5.339	9.121	5.881	5.549
djikstra	8.090	5.100	4.794	10.290	6.322	5.915	10.668	6.634	6.220	10.932	6.898	6.484
edges	3.444	2.641	2.558	5.337	3.729	3.563	11.297	7.253	6.833	11.628	7.580	7.159
gsmd	5.093	3.904	3.745	5.842	4.381	4.186	6.367	4.803	4.589	6.949	5.274	5.039
gsme	7.890	4.957	4.657	9.834	6.015	5.623	10.593	6.625	6.200	10.877	6.915	6.489
jpegd	6.115	3.954	3.733	9.530	5.787	5.403	10.084	6.099	5.690	10.132	6.142	5.733
jpege	7.327	5.755	5.594	9.674	7.362	7.125	12.550	9.652	9.354	13.839	10.941	10.643
pcm	3.230	2.452	2.373	14.284	8.719	8.148	14.737	9.164	8.593	15.234	9.661	9.090
qsort	3.859	2.968	2.877	12.697	8.076	7.601	14.234	9.111	8.583	14.627	9.502	8.974
rijndaeld	1.987	1.884	1.873	3.369	2.701	2.632	3.482	2.788	2.716	3.539	2.845	2.773
sha	5.435	3.726	3.551	9.050	5.756	5.416	9.322	5.994	5.650	9.542	6.215	5.871
smoothing	5.079	3.432	3.264	6.888	4.431	4.179	8.392	5.342	5.030	8.614	5.562	5.250
string	5.610	3.940	3.769	7.116	4.815	4.580	16.712	10.500	9.861	17.156	10.944	10.306

Tabela 5.15: Potência (Watts) de todas as aplicações nas metodologias de exploração de recursos (Modelo 5)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	1.411	1.410	1.410	96.086	11.626	11.026	96.615	11.990	11.389	97.008	12.384	11.782
bitcount	84.773	10.337	9.808	84.864	10.389	9.859	85.300	10.764	10.234	85.721	11.185	10.655
corners	10.913	2.636	2.577	31.828	4.946	4.755	73.653	9.777	9.323	74.014	10.097	9.642
CRC	57.963	7.412	7.052	59.022	7.543	7.177	59.208	7.727	7.361	59.418	7.936	7.570
djikstra	54.502	6.997	6.659	71.943	8.841	8.393	73.352	9.196	8.740	73.616	9.460	9.003
edges	16.037	3.156	3.064	30.540	4.758	4.575	74.976	9.853	9.390	75.400	10.184	9.720
gsmd	29.443	4.884	4.710	35.772	5.586	5.372	39.238	6.125	5.889	42.911	6.719	6.460
gsme	53.420	6.818	6.487	69.120	8.438	8.006	74.725	9.241	8.770	75.015	9.530	9.059
jpegd	39.652	5.325	5.081	67.703	8.164	7.741	72.022	8.630	8.179	72.139	8.676	8.225
jpege	31.725	6.752	6.575	45.624	8.831	8.569	57.665	11.495	11.167	58.969	12.785	12.457
pcm	15.305	2.946	2.858	100.673	12.249	11.621	101.248	12.699	12.070	101.745	13.197	12.567
qsort	17.696	3.533	3.433	84.711	11.019	10.495	94.161	12.376	11.795	94.594	12.769	12.187
rijndaeld	3.581	1.949	1.937	47.367	7.874	7.593	52.998	8.787	8.473	53.224	9.014	8.699
sha	32.046	4.814	4.620	60.635	7.863	7.488	61.455	8.124	7.745	61.676	8.345	7.965
smoothing	30.636	4.477	4.291	45.035	5.990	5.712	55.739	7.277	6.933	55.981	7.498	7.154
string	31.530	4.999	4.811	42.837	6.275	6.015	113.467	14.453	13.749	113.912	14.898	14.194

Tabela 5.16: Energia (Joules) de todas as aplicações nas metodologias de exploração de recursos (Modelo 2)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	0,244	0,244	0,244	0,318	0,410	0,357	0,349	0,441	0,388	0,385	0,477	0,424
bitcount	0,377	0,483	0,422	0,382	0,488	0,427	0,426	0,532	0,471	0,476	0,582	0,521
corners	0,025	0,026	0,026	0,024	0,027	0,025	0,024	0,029	0,026	0,026	0,031	0,028
CRC	0,203	0,252	0,224	0,205	0,254	0,225	0,219	0,268	0,240	0,236	0,285	0,256
djikstra	0,386	0,477	0,425	0,380	0,484	0,424	0,411	0,516	0,456	0,447	0,552	0,492
edges	0,049	0,053	0,051	0,048	0,055	0,051	0,049	0,060	0,053	0,053	0,064	0,058
gsmd	0,244	0,260	0,251	0,243	0,261	0,250	0,249	0,269	0,258	0,257	0,277	0,265
gsme	0,062	0,077	0,068	0,061	0,078	0,068	0,066	0,083	0,073	0,072	0,090	0,080
jpegd	0,166	0,199	0,180	0,146	0,186	0,162	0,146	0,188	0,163	0,148	0,190	0,165
jpege	0,077	0,079	0,079	0,075	0,077	0,077	0,086	0,089	0,089	0,101	0,104	0,104
pcm	0,216	0,234	0,223	0,256	0,329	0,287	0,286	0,358	0,317	0,320	0,392	0,351
qsort	0,121	0,131	0,125	0,123	0,150	0,134	0,132	0,161	0,145	0,144	0,173	0,156
rijndaeld	0,295	0,298	0,296	0,251	0,264	0,256	0,255	0,269	0,261	0,262	0,276	0,267
sha	0,055	0,063	0,058	0,044	0,054	0,048	0,047	0,057	0,051	0,050	0,060	0,054
smoothing	0,153	0,178	0,164	0,149	0,181	0,163	0,157	0,192	0,172	0,170	0,205	0,185
String	0,002	0,003	0,002	0,002	0,003	0,003	0,003	0,003	0,003	0,003	0,003	0,003

Tabela 5.17: Energia (Joules) de todas as aplicações nas metodologias de exploração de recursos (Modelo 3)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	0,244	0,244	0,244	1,232	0,749	0,700	1,264	0,781	0,731	1,299	0,816	0,767
bitcount	1,429	0,873	0,816	1,434	0,879	0,822	1,478	0,923	0,866	1,528	0,972	0,915
corners	0,038	0,031	0,031	0,056	0,039	0,037	0,074	0,048	0,045	0,076	0,050	0,047
CRC	0,684	0,430	0,404	0,689	0,434	0,407	0,704	0,448	0,422	0,720	0,465	0,438
djikstra	1,286	0,811	0,762	1,412	0,867	0,811	1,451	0,902	0,846	1,487	0,938	0,882
edges	0,088	0,067	0,065	0,114	0,079	0,076	0,153	0,098	0,092	0,157	0,102	0,096
gsmd	0,303	0,217	0,208	0,325	0,227	0,217	0,335	0,234	0,224	0,350	0,244	0,234
gsme	0,209	0,131	0,123	0,229	0,140	0,131	0,239	0,147	0,138	0,245	0,154	0,144
jpegd	0,500	0,324	0,305	0,564	0,342	0,320	0,576	0,348	0,325	0,578	0,350	0,327
jpege	0,124	0,098	0,095	0,130	0,099	0,096	0,146	0,112	0,109	0,161	0,127	0,124
pcm	0,396	0,301	0,291	0,977	0,596	0,557	1,007	0,626	0,587	1,041	0,660	0,621
qsort	0,216	0,166	0,161	0,397	0,252	0,237	0,419	0,268	0,252	0,430	0,279	0,263
rijndaeld	0,327	0,310	0,308	0,381	0,305	0,297	0,390	0,312	0,304	0,397	0,318	0,310
sha	0,136	0,093	0,089	0,143	0,091	0,085	0,146	0,094	0,088	0,150	0,097	0,092
smoothing	0,397	0,268	0,255	0,460	0,296	0,279	0,504	0,321	0,302	0,517	0,334	0,315
string	0,005	0,004	0,004	0,006	0,004	0,004	0,009	0,005	0,005	0,009	0,006	0,005

Tabela 5.18: Energia (Joules) de todas as aplicações nas metodologias de exploração de recursos (Modelo 5)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	0,244	0,244	0,244	8,722	1,055	1,001	8,760	1,087	1,033	8,796	1,123	1,068
bitcount	10,050	1,225	1,163	10,057	1,231	1,168	10,104	1,275	1,212	10,154	1,325	1,262
corners	0,148	0,036	0,035	0,318	0,049	0,047	0,485	0,064	0,061	0,487	0,066	0,063
CRC	4,620	0,591	0,562	4,662	0,596	0,567	4,677	0,610	0,581	4,693	0,627	0,598
djikstra	8,666	1,113	1,059	9,869	1,213	1,151	9,973	1,250	1,188	10,008	1,286	1,224
edges	0,409	0,080	0,078	0,650	0,101	0,097	1,009	0,133	0,126	1,014	0,137	0,131
gsmd	1,637	0,272	0,262	1,852	0,289	0,278	1,911	0,298	0,287	1,988	0,311	0,299
gsme	1,416	0,181	0,172	1,613	0,197	0,187	1,659	0,205	0,195	1,665	0,212	0,201
jpegd	3,244	0,436	0,416	4,005	0,483	0,458	4,109	0,492	0,467	4,113	0,495	0,469
jpege	0,539	0,115	0,112	0,612	0,119	0,115	0,671	0,134	0,130	0,686	0,149	0,145
pcm	1,878	0,362	0,351	6,883	0,837	0,795	6,917	0,868	0,825	6,951	0,902	0,859
qsort	0,989	0,198	0,192	2,646	0,344	0,328	2,765	0,363	0,346	2,777	0,375	0,358
rijndaeld	0,590	0,321	0,319	1,451	0,241	0,233	1,492	0,247	0,238	1,498	0,254	0,245
sha	0,801	0,120	0,115	0,954	0,124	0,118	0,960	0,127	0,121	0,963	0,130	0,124
smoothing	2,392	0,350	0,335	3,008	0,400	0,381	3,347	0,437	0,416	3,361	0,450	0,429
string	0,030	0,005	0,005	0,037	0,005	0,005	0,059	0,008	0,007	0,059	0,008	0,007

APÊNDICE B RESULTADOS DE EXPLORAÇÃO DE POTÊNCIA

- **Desligamento de Nível**
  - **Sem Especulação**

Tabela 5.19: Potência consumida, em Watts, após a exploração com a metodologia de desligamento de níveis (Modelo 2)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	1.413	1.413	1.413	1.088	1.442	1.266	1.650	2.004	1.828	2.293	2.646	2.470
bitcount	1.019	1.352	1.185	1.079	1.412	1.245	1.555	1.889	1.721	2.099	2.433	2.265
corners	1.596	1.662	1.628	1.608	1.769	1.686	1.956	2.294	2.122	2.427	2.766	2.593
CRC	1.427	1.983	1.683	1.477	2.039	1.736	1.898	2.459	2.156	2.378	2.940	2.637
djikstra	1.241	1.607	1.426	1.264	1.680	1.473	1.766	2.226	1.997	2.370	2.829	2.601
edges	1.539	1.629	1.584	1.550	1.700	1.626	1.916	2.258	2.088	2.420	2.764	2.594
gsmd	1.643	1.879	1.732	1.603	1.866	1.702	1.731	2.065	1.855	1.881	2.299	2.036
gsme	1.247	1.422	1.335	1.163	1.451	1.307	1.525	1.833	1.679	1.972	2.281	2.127
jpegd	1.173	1.447	1.309	1.026	1.419	1.218	1.060	1.491	1.271	1.153	1.585	1.364
jpege	3.305	3.568	3.432	3.787	4.118	3.945	6.213	6.586	6.391	8.843	9.216	9.021
pcm	1.435	1.435	1.435	1.096	1.405	1.251	1.562	1.872	1.717	2.095	2.405	2.251
qsort	1.709	1.828	1.769	1.706	2.001	1.851	2.147	2.577	2.358	2.701	3.131	2.912
rijndaeld	1.739	1.761	1.750	1.749	1.772	1.760	1.998	2.108	2.056	2.144	2.257	2.205
sha	986	1.224	1.103	891	1.183	1.035	1.146	1.439	1.291	1.440	1.733	1.584
smoothing	911	1.088	1.001	884	1.086	986	1.063	1.304	1.185	1.348	1.589	1.469
string	1.663	1.827	1.745	1.704	1.905	1.804	2.260	2.714	2.484	2.963	3.418	3.188

Tabela 5.20: Potência consumida, em Watts, após a exploração com a metodologia de desligamento de níveis (Modelo 3)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	1.413	1.413	1.413	1.204	1.627	1.532	1.766	2.189	2.094	2.409	2.832	2.736
bitcount	1.133	1.510	1.410	1.193	1.570	1.470	1.669	2.046	1.947	2.213	2.590	2.491
corners	1.623	1.690	1.667	1.671	1.834	1.779	2.084	2.432	2.318	2.555	2.904	2.789
CRC	1.657	2.126	1.893	1.710	2.184	1.948	2.130	2.604	2.369	2.611	3.085	2.849
djikstra	1.365	1.747	1.627	1.404	1.839	1.701	1.923	2.401	2.248	2.527	3.005	2.852
edges	1.574	1.671	1.643	1.606	1.768	1.722	2.045	2.412	2.303	2.551	2.921	2.812
gsmd	1.932	2.107	2.051	1.901	2.129	2.057	2.160	2.428	2.347	2.467	2.782	2.692
gsme	1.305	1.497	1.442	1.261	1.572	1.481	1.631	1.966	1.868	2.081	2.416	2.318
jpegd	1.271	1.562	1.474	1.172	1.586	1.456	1.221	1.673	1.529	1.314	1.766	1.623
jpege	3.421	3.653	3.548	3.934	4.227	4.095	6.379	6.709	6.561	9.008	9.339	9.191
pcm	1.435	1.435	1.435	1.198	1.559	1.472	1.664	2.026	1.938	2.197	2.559	2.472
qsort	1.752	1.867	1.823	1.816	2.108	2.004	2.305	2.739	2.590	2.860	3.293	3.144
rijndaeld	1.746	1.768	1.760	1.757	1.779	1.771	2.778	2.943	2.864	3.075	3.247	3.168
sha	1.072	1.294	1.204	997	1.269	1.159	1.253	1.526	1.415	1.546	1.819	1.709
smoothing	972	1.146	1.083	953	1.154	1.083	1.146	1.387	1.303	1.431	1.672	1.588
string	1.717	1.911	1.865	1.772	2.004	1.947	2.418	2.917	2.776	3.121	3.621	3.480

Tabela 5.21: : Potência consumida, em Watts, após a exploração com a metodologia de desligamento de níveis (Modelo 4)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	1.413	1.413	1.413	2.588	1.627	1.532	3.151	2.189	2.094	3.793	2.832	2.736
bitcount	2.471	1.510	1.410	2.532	1.570	1.470	3.009	2.046	1.947	3.554	2.590	2.491
corners	1.931	1.690	1.667	2.387	1.834	1.779	3.551	2.432	2.318	4.023	2.904	2.789
CRC	4.403	2.126	1.893	4.479	2.184	1.948	4.900	2.604	2.369	5.381	3.085	2.849
djikstra	2.831	1.747	1.627	3.082	1.839	1.701	3.787	2.401	2.248	4.391	3.005	2.852
edges	1.979	1.671	1.643	2.263	1.768	1.722	3.536	2.412	2.303	4.045	2.921	2.812
gsmd	2.887	2.108	2.051	3.084	2.130	2.057	3.470	2.430	2.347	3.897	2.785	2.692
gsme	2.004	1.497	1.442	2.424	1.572	1.481	2.883	1.966	1.868	3.334	2.416	2.318
jpegd	2.430	1.562	1.474	2.859	1.586	1.456	3.074	1.673	1.529	3.167	1.766	1.623
jpege	4.773	3.653	3.548	5.637	4.227	4.095	8.285	6.709	6.561	10.915	9.339	9.191
pcm	1.435	1.435	1.435	2.415	1.559	1.472	2.882	2.026	1.938	3.415	2.559	2.472
qsort	2.265	1.867	1.823	3.104	2.108	2.004	4.163	2.739	2.590	4.717	3.293	3.144
rijndaeld	1.839	1.768	1.760	1.850	1.779	1.771	5.330	3.468	3.361	5.735	3.873	3.767
sha	2.096	1.294	1.204	2.260	1.269	1.159	2.520	1.526	1.415	2.814	1.819	1.709
smoothing	1.695	1.146	1.083	1.777	1.154	1.083	2.130	1.387	1.303	2.415	1.672	1.588
String	2.362	1.911	1.865	2.571	2.004	1.947	4.280	2.917	2.776	4.984	3.621	3.480

Tabela 5.22: Energia consumida, em Joules, após a exploração com a metodologia de desligamento de níveis (Modelo 2)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	0,245	0,245	0,245	0,117	0,155	0,136	0,178	0,216	0,197	0,247	0,285	0,266
bitcount	0,132	0,175	0,153	0,139	0,183	0,161	0,201	0,244	0,222	0,271	0,314	0,293
corners	0,021	0,022	0,022	0,017	0,019	0,018	0,016	0,018	0,017	0,019	0,022	0,021
CRC	0,148	0,205	0,174	0,152	0,210	0,179	0,196	0,254	0,222	0,245	0,303	0,272
djikstra	0,236	0,306	0,272	0,228	0,303	0,266	0,305	0,384	0,344	0,409	0,488	0,448
edges	0,039	0,041	0,040	0,034	0,037	0,036	0,030	0,035	0,033	0,038	0,043	0,041
gsmd	0,182	0,151	0,166	0,168	0,139	0,153	0,178	0,149	0,163	0,190	0,161	0,176
gsme	0,043	0,049	0,046	0,034	0,043	0,038	0,044	0,052	0,048	0,056	0,065	0,061
jpegd	0,098	0,121	0,110	0,071	0,099	0,085	0,070	0,099	0,084	0,077	0,105	0,091
jpege	0,064	0,069	0,066	0,063	0,069	0,066	0,098	0,103	0,100	0,139	0,145	0,142
pcm	0,196	0,196	0,196	0,091	0,117	0,104	0,130	0,155	0,143	0,174	0,200	0,187
qsort	0,102	0,109	0,105	0,081	0,095	0,088	0,088	0,105	0,096	0,110	0,128	0,119
rijndaeld	0,287	0,290	0,288	0,288	0,292	0,290	0,210	0,217	0,212	0,225	0,233	0,227
sha	0,037	0,046	0,042	0,030	0,040	0,035	0,039	0,049	0,044	0,049	0,059	0,054
smoothing	0,109	0,130	0,120	0,101	0,124	0,112	0,114	0,139	0,127	0,144	0,170	0,157
string	0,002	0,002	0,002	0,002	0,002	0,002	0,002	0,002	0,002	0,002	0,003	0,002

Tabela 5.23: Energia consumida, em Joules, após a exploração com a metodologia de desligamento de níveis (Modelo 3)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	0,245	0,245	0,245	0,130	0,175	0,165	0,190	0,236	0,226	0,260	0,305	0,295
bitcount	0,146	0,195	0,182	0,154	0,203	0,190	0,216	0,264	0,252	0,286	0,335	0,322
corners	0,022	0,023	0,022	0,018	0,019	0,019	0,017	0,019	0,018	0,020	0,023	0,022
CRC	0,172	0,220	0,196	0,176	0,225	0,201	0,220	0,269	0,244	0,269	0,318	0,294
djikstra	0,260	0,333	0,310	0,254	0,332	0,307	0,332	0,414	0,388	0,436	0,518	0,492
edges	0,039	0,042	0,041	0,035	0,039	0,038	0,032	0,038	0,036	0,040	0,046	0,044
gsmd	0,125	0,127	0,124	0,111	0,116	0,112	0,120	0,125	0,121	0,132	0,138	0,134
gsme	0,045	0,052	0,050	0,037	0,046	0,044	0,046	0,056	0,053	0,059	0,069	0,066
jpegd	0,107	0,131	0,124	0,081	0,110	0,101	0,081	0,111	0,102	0,087	0,117	0,108
jpege	0,066	0,070	0,068	0,066	0,071	0,069	0,100	0,105	0,103	0,141	0,147	0,144
pcm	0,196	0,196	0,196	0,099	0,129	0,122	0,138	0,168	0,161	0,182	0,212	0,205
qsort	0,104	0,111	0,108	0,086	0,100	0,095	0,094	0,112	0,106	0,117	0,134	0,128
rijndaeld	0,288	0,292	0,290	0,290	0,293	0,292	0,144	0,149	0,145	0,159	0,164	0,160
sha	0,041	0,049	0,046	0,034	0,043	0,040	0,043	0,052	0,048	0,053	0,062	0,058
smoothing	0,116	0,137	0,130	0,109	0,131	0,123	0,123	0,148	0,139	0,153	0,179	0,170
string	0,002	0,002	0,002	0,002	0,002	0,002	0,002	0,002	0,002	0,002	0,003	0,003

Tabela 5.24: Energia consumida, em Joules, após a exploração com a metodologia de desligamento de níveis (Modelo 4)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	0,245	0,245	0,245	0,279	0,175	0,165	0,339	0,236	0,226	0,409	0,305	0,295
bitcount	0,320	0,195	0,182	0,327	0,203	0,190	0,389	0,264	0,252	0,459	0,335	0,322
corners	0,026	0,023	0,022	0,025	0,019	0,019	0,028	0,019	0,018	0,032	0,023	0,022
CRC	0,456	0,220	0,196	0,462	0,225	0,201	0,505	0,269	0,244	0,555	0,318	0,294
djikstra	0,539	0,333	0,310	0,556	0,332	0,307	0,653	0,414	0,388	0,757	0,518	0,492
edges	0,049	0,042	0,041	0,050	0,039	0,038	0,055	0,038	0,036	0,063	0,046	0,044
gsmd	0,175	0,128	0,124	0,168	0,116	0,112	0,179	0,125	0,121	0,192	0,137	0,134
gsme	0,069	0,052	0,050	0,071	0,046	0,044	0,082	0,056	0,053	0,095	0,069	0,066
jpegd	0,204	0,131	0,124	0,199	0,110	0,101	0,204	0,111	0,102	0,211	0,117	0,108
jpege	0,092	0,070	0,068	0,094	0,071	0,069	0,130	0,105	0,103	0,171	0,147	0,144
pcm	0,196	0,196	0,196	0,200	0,129	0,122	0,239	0,168	0,161	0,283	0,212	0,205
qsort	0,135	0,111	0,108	0,148	0,100	0,095	0,170	0,112	0,106	0,192	0,134	0,128
rijndaeld	0,303	0,292	0,290	0,305	0,293	0,292	0,202	0,131	0,127	0,217	0,147	0,143
sha	0,079	0,049	0,046	0,077	0,043	0,040	0,086	0,052	0,048	0,096	0,062	0,058
smoothing	0,203	0,137	0,130	0,202	0,131	0,123	0,228	0,148	0,139	0,258	0,179	0,170
string	0,002	0,002	0,002	0,003	0,002	0,002	0,003	0,002	0,002	0,004	0,003	0,003

- **Com Especulação**

Tabela 5.25: Potência consumida, em Watts, após a exploração com a metodologia de desligamento de níveis (Modelo 2)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	1.410	1.410	1.410	935	1.198	1.113	1.278	1.542	1.457	1.671	1.935	1.850
bitcount	986	1.284	1.185	1.032	1.330	1.231	1.400	1.698	1.599	1.821	2.119	2.020
corners	1.608	1.641	1.629	1.595	1.709	1.671	1.808	2.081	1.994	2.122	2.396	2.308
CRC	1.233	1.623	1.460	1.250	1.648	1.482	1.434	1.831	1.665	1.643	2.040	1.875
djikstra	1.150	1.458	1.346	1.072	1.478	1.329	1.296	1.708	1.557	1.559	1.972	1.821
edges	1.548	1.606	1.587	1.535	1.648	1.613	1.699	1.969	1.886	2.007	2.275	2.192
gsmd	1.553	1.601	1.583	1.546	1.604	1.582	1.603	1.663	1.640	1.671	1.735	1.711
gsme	1.009	1.249	1.160	875	1.189	1.072	1.083	1.416	1.292	1.368	1.701	1.577
jpegd	1.042	1.196	1.152	758	1.019	935	745	1.024	934	782	1.062	972
jpege	3.476	3.950	3.584	3.827	4.678	3.968	4.911	6.268	5.076	5.917	7.557	6.083
pcm	1.376	1.410	1.398	1.057	1.334	1.245	1.491	1.768	1.680	1.989	2.266	2.177
qsort	1.785	1.877	1.841	1.887	2.311	2.153	2.239	2.708	2.534	2.626	3.095	2.921
rijndaeld	1.746	1.753	1.751	1.811	1.866	1.848	1.859	1.917	1.898	1.913	1.972	1.953
sha	1.494	1.703	1.618	1.430	1.831	1.670	1.621	2.027	1.864	1.842	2.248	2.085
smoothing	1.248	1.422	1.349	1.154	1.392	1.296	1.270	1.557	1.442	1.488	1.775	1.661
string	1.696	1.832	1.789	1.721	1.906	1.846	2.131	2.641	2.469	2.575	3.086	2.914

Tabela 5.26: Potência consumida, em Watts, após a exploração com a metodologia de desligamento de níveis (Modelo 3)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	1.410	1.410	1.410	1.036	1.360	1.332	1.379	1.704	1.676	1.773	2.097	2.069
bitcount	1.110	1.431	1.378	1.156	1.478	1.425	1.524	1.846	1.793	1.945	2.267	2.214
corners	1.625	1.658	1.651	1.650	1.763	1.742	1.938	2.211	2.161	2.252	2.526	2.475
CRC	1.399	1.729	1.593	1.419	1.755	1.617	1.602	1.938	1.801	1.812	2.148	2.010
djikstra	1.286	1.557	1.471	1.252	1.609	1.495	1.479	1.842	1.726	1.742	2.106	1.990
edges	1.575	1.632	1.621	1.586	1.701	1.681	1.843	2.130	2.081	2.160	2.450	2.401
gsmd	1.888	2.022	1.997	1.902	2.067	2.035	2.070	2.264	2.229	2.269	2.497	2.459
gsme	1.107	1.343	1.282	1.003	1.312	1.231	1.219	1.550	1.462	1.503	1.839	1.751
jpegd	1.112	1.267	1.242	875	1.142	1.092	870	1.155	1.101	909	1.193	1.139
jpege	3.855	4.006	3.939	4.547	4.761	4.668	6.107	6.372	6.259	7.395	7.660	7.547
pcm	1.388	1.433	1.430	1.164	1.503	1.474	1.598	1.938	1.909	2.095	2.436	2.406
qsort	1.824	1.906	1.879	2.071	2.462	2.346	2.442	2.875	2.748	2.829	3.263	3.136
rijndaeld	1.749	1.757	1.755	1.865	1.894	1.880	1.918	1.948	1.935	1.975	2.006	1.992
sha	1.588	1.760	1.690	1.614	1.941	1.809	1.806	2.138	2.005	2.027	2.359	2.226
smoothing	1.317	1.476	1.417	1.250	1.472	1.398	1.387	1.656	1.569	1.605	1.875	1.788
string	1.755	1.887	1.859	1.801	1.981	1.942	2.352	2.846	2.729	2.796	3.290	3.173

Tabela 5.27: Potência consumida, em Watts, após a exploração com a metodologia de desligamento de níveis (Modelo 5)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	1.410	1.410	1.410	2.087	1.361	1.333	2.432	1.705	1.677	2.826	2.098	2.070
bitcount	2.376	1.433	1.380	2.423	1.479	1.426	2.792	1.847	1.795	3.213	2.268	2.215
corners	1.798	1.658	1.651	2.196	1.764	1.742	3.223	2.213	2.162	3.538	2.528	2.477
CRC	3.124	1.731	1.595	3.173	1.757	1.619	3.357	1.940	1.803	3.566	2.150	2.012
djikstra	2.706	1.559	1.472	3.121	1.611	1.497	3.377	1.844	1.728	3.641	2.108	1.992
edges	1.853	1.633	1.622	2.113	1.701	1.682	3.167	2.132	2.083	3.488	2.452	2.402
gsmd	2.661	2.024	1.998	2.824	2.069	2.036	3.093	2.266	2.230	3.392	2.499	2.459
gsme	2.132	1.345	1.283	2.344	1.314	1.232	2.663	1.552	1.464	2.952	1.841	1.753
jpegd	1.843	1.268	1.243	2.172	1.144	1.094	2.254	1.157	1.103	2.294	1.195	1.141
jpege	4.925	4.007	3.940	6.067	4.763	4.670	7.968	6.374	6.260	9.257	7.662	7.549
pcm	1.515	1.433	1.430	2.271	1.504	1.475	2.707	1.939	1.910	3.204	2.436	2.407
qsort	2.223	1.906	1.879	3.948	2.464	2.349	4.515	2.878	2.751	4.904	3.266	3.139
rijndaeld	1.785	1.757	1.756	5.134	3.363	3.314	5.704	3.737	3.681	5.931	3.963	3.908
sha	2.557	1.761	1.691	3.488	1.943	1.811	3.700	2.140	2.007	3.920	2.361	2.228
smoothing	2.048	1.477	1.418	2.257	1.473	1.399	2.608	1.658	1.571	2.827	1.876	1.789
string	2.361	1.888	1.860	2.621	1.983	1.943	4.615	2.849	2.732	5.060	3.294	3.177

Tabela 5.28: Energia consumida, em Joules, após a exploração com a metodologia de desligamento de níveis (Modelo 2)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	0,244	0,244	0,244	0,085	0,109	0,101	0,116	0,140	0,132	0,152	0,175	0,168
bitcount	0,117	0,152	0,140	0,122	0,158	0,146	0,166	0,201	0,189	0,216	0,251	0,239
corners	0,022	0,022	0,022	0,016	0,017	0,017	0,012	0,014	0,013	0,014	0,016	0,015
CRC	0,098	0,129	0,116	0,099	0,130	0,117	0,113	0,145	0,132	0,130	0,161	0,148
djikstra	0,183	0,232	0,214	0,147	0,203	0,182	0,176	0,232	0,212	0,212	0,268	0,248
edges	0,039	0,041	0,040	0,032	0,035	0,034	0,024	0,028	0,026	0,028	0,032	0,031
gsmd	0,207	0,212	0,210	0,200	0,206	0,203	0,205	0,212	0,209	0,211	0,218	0,215
gsme	0,027	0,033	0,031	0,020	0,028	0,025	0,024	0,032	0,029	0,031	0,038	0,036
jpegd	0,085	0,098	0,094	0,046	0,060	0,055	0,043	0,058	0,053	0,046	0,061	0,055
jpege	0,066	0,067	0,069	0,063	0,063	0,065	0,073	0,073	0,076	0,088	0,088	0,091
pcm	0,169	0,173	0,172	0,072	0,091	0,085	0,102	0,121	0,115	0,136	0,155	0,149
qsort	0,100	0,105	0,103	0,059	0,072	0,067	0,066	0,080	0,074	0,077	0,091	0,086
rijndaeld	0,287	0,289	0,288	0,217	0,222	0,220	0,221	0,226	0,224	0,228	0,233	0,230
sha	0,037	0,043	0,040	0,023	0,029	0,026	0,025	0,032	0,029	0,029	0,035	0,033
smoothing	0,097	0,111	0,105	0,077	0,093	0,087	0,076	0,093	0,087	0,089	0,107	0,100
string	0,002	0,002	0,002	0,001	0,002	0,002	0,001	0,001	0,001	0,001	0,002	0,002

Tabela 5.29: Energia consumida, em Joules, após a exploração com a metodologia de desligamento de níveis (Modelo 3)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	0,244	0,244	0,244	0,094	0,123	0,121	0,125	0,154	0,152	0,161	0,190	0,188
bitcount	0,132	0,170	0,163	0,137	0,175	0,169	0,181	0,219	0,212	0,230	0,269	0,262
corners	0,022	0,023	0,022	0,016	0,018	0,017	0,013	0,015	0,014	0,015	0,017	0,016
CRC	0,111	0,138	0,127	0,112	0,139	0,128	0,127	0,153	0,142	0,143	0,170	0,159
djikstra	0,205	0,248	0,234	0,172	0,221	0,205	0,201	0,250	0,235	0,237	0,286	0,271
edges	0,040	0,042	0,041	0,034	0,036	0,036	0,025	0,029	0,028	0,029	0,033	0,032
gsmd	0,112	0,112	0,111	0,106	0,107	0,105	0,109	0,110	0,109	0,114	0,116	0,114
gsme	0,029	0,036	0,034	0,023	0,031	0,029	0,028	0,034	0,032	0,034	0,041	0,039
jpegd	0,091	0,104	0,102	0,052	0,068	0,065	0,050	0,066	0,063	0,052	0,068	0,065
jpege	0,065	0,068	0,067	0,061	0,064	0,063	0,071	0,074	0,073	0,086	0,089	0,088
pcm	0,170	0,176	0,175	0,080	0,103	0,101	0,109	0,132	0,130	0,143	0,166	0,164
qsort	0,102	0,107	0,105	0,065	0,077	0,073	0,072	0,084	0,081	0,083	0,096	0,092
rijndaeld	0,288	0,289	0,289	0,211	0,214	0,213	0,215	0,218	0,216	0,221	0,224	0,223
sha	0,040	0,044	0,042	0,025	0,031	0,028	0,028	0,033	0,031	0,032	0,037	0,035
smoothing	0,103	0,115	0,111	0,083	0,098	0,093	0,083	0,099	0,094	0,096	0,113	0,107
string	0,002	0,002	0,002	0,002	0,002	0,002	0,001	0,001	0,001	0,001	0,002	0,002

Tabela 5.30: Energia consumida, em Joules, após a exploração com a metodologia de desligamento de níveis (Modelo 4)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	0,244	0,244	0,244	0,189	0,123	0,121	0,221	0,155	0,152	0,256	0,190	0,188
bitcount	0,282	0,170	0,164	0,287	0,175	0,169	0,331	0,219	0,213	0,381	0,269	0,262
corners	0,024	0,023	0,022	0,022	0,018	0,017	0,021	0,015	0,014	0,023	0,017	0,016
CRC	0,249	0,138	0,127	0,251	0,139	0,128	0,265	0,153	0,142	0,282	0,170	0,159
djikstra	0,430	0,248	0,234	0,428	0,221	0,205	0,459	0,251	0,235	0,495	0,287	0,271
edges	0,047	0,042	0,041	0,045	0,036	0,036	0,043	0,029	0,028	0,047	0,033	0,032
gsmd	0,148	0,113	0,111	0,146	0,107	0,105	0,151	0,110	0,109	0,157	0,116	0,114
gsme	0,057	0,036	0,034	0,055	0,031	0,029	0,059	0,034	0,033	0,066	0,041	0,039
jpegd	0,151	0,104	0,102	0,129	0,068	0,065	0,129	0,066	0,063	0,131	0,068	0,065
jpege	0,084	0,068	0,067	0,081	0,064	0,063	0,093	0,074	0,073	0,108	0,089	0,088
pcm	0,186	0,176	0,175	0,155	0,103	0,101	0,185	0,132	0,130	0,219	0,166	0,164
qsort	0,124	0,107	0,105	0,123	0,077	0,073	0,133	0,084	0,081	0,144	0,096	0,092
rijndaeld	0,294	0,289	0,289	0,157	0,103	0,101	0,161	0,105	0,104	0,167	0,112	0,110
sha	0,064	0,044	0,042	0,055	0,031	0,028	0,058	0,033	0,031	0,061	0,037	0,035
smoothing	0,160	0,115	0,111	0,151	0,098	0,093	0,157	0,100	0,094	0,170	0,113	0,107
string	0,002	0,002	0,002	0,002	0,002	0,002	0,002	0,001	0,001	0,003	0,002	0,002

- **Desligamento de Unidades Funcionais**
  - **Sem Especulação**

Tabela 5.31: : Potência consumida, em Watts, após a exploração com a metodologia de desligamento de níveis (Modelo 2)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	1.413	1.413	1.413	873	873	873	1.435	1.435	1.435	2.077	2.077	2.077
bitcount	821	821	821	880	880	880	1.356	1.356	1.356	1.901	1.901	1.901
corners	1.549	1.549	1.549	1.499	1.499	1.499	1.728	1.730	1.730	2.199	2.201	2.201
CRC	970	970	970	1.017	1.017	1.017	1.437	1.437	1.437	1.918	1.918	1.918
djikstra	1.007	1.007	1.007	995	995	995	1.467	1.467	1.467	2.071	2.071	2.071
edges	1.480	1.484	1.484	1.454	1.458	1.458	1.694	1.702	1.702	2.198	2.208	2.208
gsmd	1.574	1.688	1.627	1.523	1.625	1.571	1.646	1.807	1.717	1.793	2.026	1.891
gsme	1.143	1.143	1.143	988	988	988	1.338	1.338	1.338	1.785	1.787	1.786
jpegd	995	995	995	758	758	758	764	765	765	858	858	858
jpege	3.104	3.104	3.104	3.531	3.531	3.531	5.925	5.926	5.926	8.555	8.556	8.556
pcm	1.435	1.435	1.435	913	913	913	1.380	1.380	1.380	1.913	1.913	1.913
qsort	1.625	1.625	1.625	1.492	1.492	1.492	1.837	1.837	1.837	2.392	2.392	2.392
rijndaeld	1.724	1.724	1.724	1.734	1.734	1.734	1.920	1.931	1.931	2.066	2.080	2.080
sha	843	843	843	716	716	716	971	971	971	1.264	1.264	1.264
smoothing	810	810	810	768	768	768	923	924	924	1.208	1.208	1.208
string	1.561	1.561	1.561	1.577	1.577	1.577	1.956	1.959	1.959	2.659	2.663	2.663

Tabela 5.32: Potência consumida, em Watts, após a exploração com a metodologia de desligamento de níveis (Modelo 3)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	1.413	1.413	1.413	873	873	873	1.435	1.435	1.435	2.077	2.077	2.077
bitcount	821	821	821	880	880	880	1.356	1.356	1.356	1.901	1.901	1.901
corners	1.549	1.549	1.549	1.499	1.499	1.499	1.728	1.730	1.730	2.199	2.201	2.201
CRC	970	970	970	1.017	1.017	1.017	1.437	1.437	1.437	1.918	1.918	1.918
djikstra	1.007	1.007	1.007	995	995	995	1.467	1.467	1.467	2.071	2.071	2.071
edges	1.480	1.484	1.484	1.454	1.458	1.458	1.695	1.703	1.703	2.202	2.212	2.212
gsmd	1.668	1.778	1.778	1.586	1.702	1.702	1.819	1.964	1.962	2.104	2.282	2.277
gsme	1.143	1.143	1.143	988	988	988	1.337	1.338	1.338	1.787	1.789	1.788
jpegd	995	995	995	758	758	758	764	765	765	858	858	858
jpege	3.104	3.104	3.104	3.531	3.531	3.531	5.925	5.926	5.926	8.555	8.556	8.556
pcm	1.435	1.435	1.435	913	913	913	1.380	1.380	1.380	1.913	1.913	1.913
qsort	1.625	1.625	1.625	1.492	1.492	1.492	1.837	1.837	1.837	2.392	2.392	2.392
rijndaeld	1.724	1.724	1.724	1.734	1.734	1.734	2.460	2.489	2.489	2.757	2.793	2.793
sha	843	843	843	716	716	716	971	971	971	1.264	1.264	1.264
smoothing	810	810	810	768	768	768	923	924	924	1.208	1.208	1.208
string	1.561	1.561	1.561	1.577	1.577	1.577	1.956	1.959	1.959	2.659	2.663	2.663

Tabela 5.33: Potência consumida, em Watts, após a exploração com a metodologia de desligamento de níveis (Modelo 4)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	1.413	1.413	1.413	873	873	873	1.435	1.435	1.435	2.077	2.077	2.077
bitcount	821	821	821	880	880	880	1.356	1.356	1.356	1.901	1.901	1.901
corners	1.549	1.549	1.549	1.499	1.499	1.499	1.730	1.730	1.730	2.201	2.201	2.201
CRC	970	970	970	1.017	1.017	1.017	1.437	1.437	1.437	1.918	1.918	1.918
djikstra	1.007	1.007	1.007	995	995	995	1.467	1.467	1.467	2.071	2.071	2.071
edges	1.484	1.484	1.484	1.458	1.458	1.458	1.703	1.703	1.703	2.212	2.212	2.212
gsmd	1.778	1.778	1.778	1.702	1.702	1.702	1.964	1.964	1.962	2.283	2.283	2.277
gsme	1.143	1.143	1.143	988	988	988	1.338	1.338	1.338	1.789	1.789	1.788
jpegd	995	995	995	758	758	758	765	765	765	858	858	858
jpege	3.104	3.104	3.104	3.531	3.531	3.531	5.926	5.926	5.926	8.556	8.556	8.556
pcm	1.435	1.435	1.435	913	913	913	1.380	1.380	1.380	1.913	1.913	1.913
qsort	1.625	1.625	1.625	1.492	1.492	1.492	1.837	1.837	1.837	2.392	2.392	2.392
rijndaeld	1.724	1.724	1.724	1.734	1.734	1.734	2.858	2.858	2.858	3.263	3.263	3.263
sha	843	843	843	716	716	716	971	971	971	1.264	1.264	1.264
smoothing	810	810	810	768	768	768	924	924	924	1.208	1.208	1.208
string	1.561	1.561	1.561	1.577	1.577	1.577	1.959	1.959	1.959	2.663	2.663	2.663

Tabela 5.34: Energia consumida, em Joules, após a exploração com a metodologia de desligamento de níveis (Modelo 2)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	0,245	0,245	0,245	0,094	0,094	0,094	0,155	0,155	0,155	0,224	0,224	0,224
bitcount	0,106	0,106	0,106	0,114	0,114	0,114	0,175	0,175	0,175	0,246	0,246	0,246
corners	0,021	0,021	0,021	0,016	0,016	0,016	0,014	0,014	0,014	0,018	0,018	0,018
CRC	0,100	0,100	0,100	0,105	0,105	0,105	0,148	0,148	0,148	0,198	0,198	0,198
djikstra	0,192	0,192	0,192	0,180	0,180	0,180	0,253	0,253	0,253	0,357	0,357	0,357
edges	0,037	0,037	0,037	0,032	0,032	0,032	0,027	0,027	0,027	0,035	0,035	0,035
gsmd	0,174	0,136	0,156	0,159	0,121	0,141	0,169	0,130	0,151	0,182	0,142	0,163
gsme	0,039	0,039	0,039	0,029	0,029	0,029	0,038	0,038	0,038	0,051	0,051	0,051
jpegd	0,083	0,083	0,083	0,053	0,053	0,053	0,051	0,051	0,051	0,057	0,057	0,057
jpege	0,060	0,060	0,060	0,059	0,059	0,059	0,093	0,093	0,093	0,134	0,134	0,134
pcm	0,196	0,196	0,196	0,076	0,076	0,076	0,114	0,114	0,114	0,159	0,159	0,159
qsort	0,097	0,097	0,097	0,071	0,071	0,071	0,075	0,075	0,075	0,098	0,098	0,098
rijndaeld	0,284	0,284	0,284	0,286	0,286	0,286	0,201	0,199	0,199	0,217	0,214	0,214
sha	0,032	0,032	0,032	0,024	0,024	0,024	0,033	0,033	0,033	0,043	0,043	0,043
smoothing	0,097	0,097	0,097	0,088	0,088	0,088	0,099	0,099	0,099	0,129	0,129	0,129
string	0,002	0,002	0,002	0,002	0,002	0,002	0,001	0,001	0,001	0,002	0,002	0,002

Tabela 5.35: Energia consumida, em Joules, após a exploração com a metodologia de desligamento de níveis (Modelo 3)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	0,245	0,245	0,245	0,094	0,094	0,094	0,155	0,155	0,155	0,224	0,224	0,224
bitcount	0,106	0,106	0,106	0,114	0,114	0,114	0,175	0,175	0,175	0,246	0,246	0,246
corners	0,021	0,021	0,021	0,016	0,016	0,016	0,014	0,014	0,014	0,018	0,018	0,018
CRC	0,100	0,100	0,100	0,105	0,105	0,105	0,148	0,148	0,148	0,198	0,198	0,198
djikstra	0,192	0,192	0,192	0,180	0,180	0,180	0,253	0,253	0,253	0,357	0,357	0,357
edges	0,037	0,037	0,037	0,032	0,032	0,032	0,027	0,027	0,027	0,034	0,034	0,034
gsmd	0,108	0,108	0,108	0,093	0,093	0,093	0,101	0,101	0,102	0,113	0,113	0,113
gsme	0,039	0,039	0,039	0,029	0,029	0,029	0,038	0,038	0,038	0,051	0,051	0,051
jpegd	0,083	0,083	0,083	0,053	0,053	0,053	0,051	0,051	0,051	0,057	0,057	0,057
jpege	0,060	0,060	0,060	0,059	0,059	0,059	0,093	0,093	0,093	0,134	0,134	0,134
pcm	0,196	0,196	0,196	0,076	0,076	0,076	0,114	0,114	0,114	0,159	0,159	0,159
qsort	0,097	0,097	0,097	0,071	0,071	0,071	0,075	0,075	0,075	0,098	0,098	0,098
rijndaeld	0,284	0,284	0,284	0,286	0,286	0,286	0,127	0,126	0,126	0,143	0,141	0,141
sha	0,032	0,032	0,032	0,024	0,024	0,024	0,033	0,033	0,033	0,043	0,043	0,043
smoothing	0,097	0,097	0,097	0,088	0,088	0,088	0,099	0,099	0,099	0,129	0,129	0,129
string	0,002	0,002	0,002	0,002	0,002	0,002	0,001	0,001	0,001	0,002	0,002	0,002

Tabela 5.36: Energia consumida, em Joules, após a exploração com a metodologia de desligamento de níveis (Modelo 4)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	0,245	0,245	0,245	0,094	0,094	0,094	0,155	0,155	0,155	0,224	0,224	0,224
bitcount	0,106	0,106	0,106	0,114	0,114	0,114	0,175	0,175	0,175	0,246	0,246	0,246
corners	0,021	0,021	0,021	0,016	0,016	0,016	0,014	0,014	0,014	0,018	0,018	0,018
CRC	0,100	0,100	0,100	0,105	0,105	0,105	0,148	0,148	0,148	0,198	0,198	0,198
djikstra	0,192	0,192	0,192	0,180	0,180	0,180	0,253	0,253	0,253	0,357	0,357	0,357
edges	0,037	0,037	0,037	0,032	0,032	0,032	0,027	0,027	0,027	0,034	0,034	0,034
gsmd	0,108	0,108	0,108	0,093	0,093	0,093	0,101	0,101	0,102	0,113	0,113	0,113
gsme	0,039	0,039	0,039	0,029	0,029	0,029	0,038	0,038	0,038	0,051	0,051	0,051
jpegd	0,083	0,083	0,083	0,053	0,053	0,053	0,051	0,051	0,051	0,057	0,057	0,057
jpege	0,060	0,060	0,060	0,059	0,059	0,059	0,093	0,093	0,093	0,134	0,134	0,134
pcm	0,196	0,196	0,196	0,076	0,076	0,076	0,114	0,114	0,114	0,159	0,159	0,159
qsort	0,097	0,097	0,097	0,071	0,071	0,071	0,075	0,075	0,075	0,098	0,098	0,098
rijndaeld	0,284	0,284	0,284	0,286	0,286	0,286	0,108	0,108	0,108	0,124	0,124	0,124
sha	0,032	0,032	0,032	0,024	0,024	0,024	0,033	0,033	0,033	0,043	0,043	0,043
smoothing	0,097	0,097	0,097	0,088	0,088	0,088	0,099	0,099	0,099	0,129	0,129	0,129
string	0,002	0,002	0,002	0,002	0,002	0,002	0,001	0,001	0,001	0,002	0,002	0,002

- **Com Especulação**

Tabela 5.37: Potência consumida, em Watts, após a exploração com a metodologia de desligamento de níveis (Modelo 2)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	1.410	1.410	1.410	765	765	765	1.107	1.107	1.107	1.501	1.501	1.501
bitcount	775	775	775	821	821	821	1.189	1.189	1.189	1.610	1.610	1.610
corners	1.576	1.576	1.576	1.500	1.500	1.500	1.585	1.586	1.586	1.899	1.900	1.900
CRC	926	926	926	939	939	939	1.122	1.122	1.122	1.331	1.331	1.331
djikstra	899	899	899	740	740	740	959	959	959	1.222	1.223	1.223
edges	1.502	1.506	1.506	1.448	1.453	1.453	1.485	1.493	1.493	1.793	1.798	1.798
gsmd	1.509	1.515	1.515	1.494	1.500	1.500	1.549	1.555	1.555	1.614	1.622	1.622
gsme	841	841	841	655	655	655	849	849	849	1.134	1.134	1.134
jpegd	923	923	923	545	531	531	518	502	502	555	540	540
jpege	3.312	3.574	3.313	3.612	4.148	3.613	4.657	5.617	4.659	5.663	6.906	5.665
pcm	1.356	1.356	1.356	878	878	878	1.312	1.312	1.312	1.809	1.809	1.809
qsort	1.712	1.712	1.712	1.540	1.542	1.542	1.856	1.859	1.859	2.243	2.246	2.246
rijndaeld	1.740	1.740	1.740	1.768	1.773	1.773	1.814	1.821	1.821	1.868	1.875	1.875
sha	1.352	1.354	1.354	1.157	1.160	1.160	1.345	1.349	1.349	1.565	1.570	1.570
smoothing	1.141	1.141	1.141	1.005	1.005	1.005	1.087	1.087	1.087	1.305	1.305	1.305
string	1.589	1.590	1.590	1.575	1.575	1.575	1.722	1.724	1.724	2.166	2.168	2.168

Tabela 5.38: Potência consumida, em Watts, após a exploração com a metodologia de desligamento de níveis (Modelo 3)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	1.410	1.410	1.410	765	765	765	1.107	1.107	1.107	1.501	1.501	1.501
bitcount	775	775	775	821	821	821	1.189	1.189	1.189	1.610	1.610	1.610
corners	1.576	1.576	1.576	1.500	1.500	1.500	1.585	1.586	1.586	1.899	1.900	1.900
CRC	926	926	926	939	939	939	1.122	1.122	1.122	1.331	1.331	1.331
djikstra	899	899	899	740	740	740	959	959	959	1.222	1.223	1.223
edges	1.502	1.506	1.506	1.449	1.453	1.453	1.494	1.502	1.502	1.811	1.820	1.820
gsmd	1.647	1.763	1.763	1.623	1.745	1.745	1.767	1.911	1.911	1.944	2.112	2.112
gsme	841	841	841	655	655	655	849	842	842	1.133	1.131	1.131
jpegd	923	923	923	530	531	531	502	502	502	540	540	540
jpege	3.574	3.574	3.574	4.147	4.148	4.148	5.614	5.617	5.617	6.901	6.906	6.906
pcm	1.356	1.356	1.356	878	878	878	1.312	1.312	1.312	1.809	1.809	1.809
qsort	1.712	1.712	1.712	1.540	1.542	1.542	1.856	1.859	1.859	2.243	2.246	2.246
rijndaeld	1.740	1.740	1.740	1.758	1.761	1.761	1.808	1.811	1.811	1.865	1.868	1.868
sha	1.352	1.354	1.354	1.157	1.160	1.160	1.345	1.349	1.349	1.565	1.570	1.570
smoothing	1.141	1.141	1.141	1.005	1.005	1.005	1.087	1.087	1.087	1.305	1.305	1.305
string	1.589	1.590	1.590	1.575	1.575	1.575	1.722	1.724	1.724	2.166	2.168	2.168

Tabela 5.39: Potência consumida, em Watts, após a exploração com a metodologia de desligamento de níveis (Modelo 5)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	1.410	1.410	1.410	765	765	765	1.107	1.107	1.107	1.501	1.501	1.501
bitcount	775	775	775	821	821	821	1.189	1.189	1.189	1.610	1.610	1.610
corners	1.576	1.576	1.576	1.500	1.500	1.500	1.586	1.586	1.586	1.900	1.900	1.900
CRC	926	926	926	939	939	939	1.122	1.122	1.122	1.331	1.331	1.331
djikstra	899	899	899	740	740	740	959	959	959	1.223	1.223	1.223
edges	1.506	1.506	1.506	1.453	1.453	1.453	1.502	1.502	1.502	1.820	1.820	1.820
gsmd	1.763	1.763	1.763	1.745	1.745	1.745	1.911	1.911	1.911	2.112	2.112	2.112
gsme	841	841	841	655	655	655	842	842	842	1.131	1.131	1.131
jpegd	923	923	923	531	531	531	502	502	502	540	540	540
jpege	3.574	3.574	3.574	4.148	4.148	4.148	5.617	5.617	5.617	6.906	6.906	6.906
pcm	1.356	1.356	1.356	878	878	878	1.312	1.312	1.312	1.809	1.809	1.809
qsort	1.712	1.712	1.712	1.542	1.542	1.542	1.859	1.859	1.859	2.246	2.246	2.246
rijndaeld	1.740	1.740	1.740	2.892	2.892	2.892	3.208	3.208	3.208	3.435	3.435	3.435
sha	1.354	1.354	1.354	1.160	1.160	1.160	1.349	1.349	1.349	1.570	1.570	1.570
smoothing	1.141	1.141	1.141	1.005	1.005	1.005	1.087	1.087	1.087	1.305	1.305	1.305
string	1.590	1.590	1.590	1.575	1.575	1.575	1.724	1.724	1.724	2.168	2.168	2.168

Tabela 5.40: Energia consumida, em Joules, após a exploração com a metodologia de desligamento de níveis (Modelo 2)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	0,244	0,244	0,244	0,069	0,069	0,069	0,100	0,100	0,100	0,136	0,136	0,136
bitcount	0,092	0,092	0,092	0,097	0,097	0,097	0,141	0,141	0,141	0,191	0,191	0,191
corners	0,021	0,021	0,021	0,015	0,015	0,015	0,010	0,010	0,010	0,012	0,012	0,012
CRC	0,074	0,074	0,074	0,074	0,074	0,074	0,089	0,089	0,089	0,105	0,105	0,105
djikstra	0,143	0,143	0,143	0,102	0,102	0,102	0,130	0,130	0,130	0,166	0,166	0,166
edges	0,038	0,038	0,038	0,031	0,031	0,031	0,021	0,021	0,021	0,025	0,025	0,025
gsmd	0,201	0,200	0,201	0,193	0,193	0,193	0,198	0,198	0,198	0,204	0,204	0,204
gsme	0,022	0,022	0,022	0,015	0,015	0,015	0,019	0,019	0,019	0,026	0,026	0,026
jpegd	0,076	0,076	0,076	0,033	0,031	0,031	0,030	0,029	0,029	0,032	0,031	0,031
jpege	0,063	0,061	0,063	0,059	0,056	0,059	0,069	0,065	0,069	0,084	0,080	0,084
pcm	0,166	0,166	0,166	0,060	0,060	0,060	0,090	0,090	0,090	0,124	0,124	0,124
qsort	0,096	0,096	0,096	0,048	0,048	0,048	0,055	0,055	0,055	0,066	0,066	0,066
rijndaeld	0,286	0,286	0,286	0,212	0,211	0,211	0,216	0,215	0,215	0,222	0,221	0,221
sha	0,034	0,034	0,034	0,018	0,018	0,018	0,021	0,021	0,021	0,025	0,025	0,025
smoothing	0,089	0,089	0,089	0,067	0,067	0,067	0,065	0,065	0,065	0,078	0,078	0,078
string	0,002	0,002	0,002	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001

Tabela 5.41: Energia consumida, em Joules, após a exploração com a metodologia de desligamento de níveis (Modelo 3)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	0,244	0,244	0,244	0,069	0,069	0,069	0,100	0,100	0,100	0,136	0,136	0,136
bitcount	0,092	0,092	0,092	0,097	0,097	0,097	0,141	0,141	0,141	0,191	0,191	0,191
corners	0,021	0,021	0,021	0,015	0,015	0,015	0,010	0,010	0,010	0,012	0,012	0,012
CRC	0,074	0,074	0,074	0,074	0,074	0,074	0,089	0,089	0,089	0,105	0,105	0,105
djikstra	0,143	0,143	0,143	0,102	0,102	0,102	0,130	0,130	0,130	0,166	0,166	0,166
edges	0,038	0,038	0,038	0,031	0,031	0,031	0,020	0,020	0,020	0,024	0,024	0,024
gsmd	0,098	0,098	0,098	0,090	0,090	0,090	0,093	0,093	0,093	0,098	0,098	0,098
gsme	0,022	0,022	0,022	0,015	0,015	0,015	0,019	0,019	0,019	0,026	0,025	0,025
jpegd	0,076	0,076	0,076	0,031	0,031	0,031	0,029	0,029	0,029	0,031	0,031	0,031
jpege	0,061	0,061	0,061	0,056	0,056	0,056	0,065	0,065	0,065	0,080	0,080	0,080
pcm	0,166	0,166	0,166	0,060	0,060	0,060	0,090	0,090	0,090	0,124	0,124	0,124
qsort	0,096	0,096	0,096	0,048	0,048	0,048	0,055	0,055	0,055	0,066	0,066	0,066
rijndaeld	0,286	0,286	0,286	0,199	0,199	0,199	0,203	0,203	0,203	0,209	0,209	0,209
sha	0,034	0,034	0,034	0,018	0,018	0,018	0,021	0,021	0,021	0,025	0,025	0,025
smoothing	0,089	0,089	0,089	0,067	0,067	0,067	0,065	0,065	0,065	0,078	0,078	0,078
string	0,002	0,002	0,002	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001

Tabela 5.42: Energia consumida, em Joules, após a exploração com a metodologia de desligamento de níveis (Modelo 5)

	Cache 4			Cache 32			Cache 256			Cache 512		
	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo	Retangular	Ótima	Baixo Custo
adpcm	0,244	0,244	0,244	0,069	0,069	0,069	0,100	0,100	0,100	0,136	0,136	0,136
bitcount	0,092	0,092	0,092	0,097	0,097	0,097	0,141	0,141	0,141	0,191	0,191	0,191
corners	0,021	0,021	0,021	0,015	0,015	0,015	0,010	0,010	0,010	0,012	0,012	0,012
CRC	0,074	0,074	0,074	0,074	0,074	0,074	0,089	0,089	0,089	0,105	0,105	0,105
djikstra	0,143	0,143	0,143	0,102	0,102	0,102	0,130	0,130	0,130	0,166	0,166	0,166
edges	0,038	0,038	0,038	0,031	0,031	0,031	0,020	0,020	0,020	0,024	0,024	0,024
gsmd	0,098	0,098	0,098	0,090	0,090	0,090	0,093	0,093	0,093	0,098	0,098	0,098
gsme	0,022	0,022	0,022	0,015	0,015	0,015	0,019	0,019	0,019	0,025	0,025	0,025
jpegd	0,076	0,076	0,076	0,031	0,031	0,031	0,029	0,029	0,029	0,031	0,031	0,031
jpege	0,061	0,061	0,061	0,056	0,056	0,056	0,065	0,065	0,065	0,080	0,080	0,080
pcm	0,166	0,166	0,166	0,060	0,060	0,060	0,090	0,090	0,090	0,124	0,124	0,124
qsort	0,096	0,096	0,096	0,048	0,048	0,048	0,055	0,055	0,055	0,066	0,066	0,066
rijndaeld	0,286	0,286	0,286	0,089	0,089	0,089	0,090	0,090	0,090	0,097	0,097	0,097
sha	0,034	0,034	0,034	0,018	0,018	0,018	0,021	0,021	0,021	0,025	0,025	0,025
smoothing	0,089	0,089	0,089	0,067	0,067	0,067	0,065	0,065	0,065	0,078	0,078	0,078
string	0,002	0,002	0,002	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,001