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

PAULO CESAR SANTOS DA SILVA JUNIOR

SISTEMAS MULTIPROCESSADOS EM CHIP

RECONFIGURABILIDADE E HETEROGENEIDADE

ECONOMIA DE ENERGIA E COMPATIBILIDADE BINÁRIA

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Microeletrônica.

Orientador: Prof. Dr. Luigi Carro

Porto Alegre
2014

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Silva Jr., Paulo Cesar Santos da
Sistemas Multiprocessados em Chip – Reconfigurabilidade e Heterogeneidade, Economia de Energia e Compatibilidade Binária /
Paulo Cesar Santos da Silva Jr. – 2014.
90 f.:il.
Orientador: Prof. Dr. Luigi Carro.
Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Microeletrônica. Porto Alegre, BR – RS, 2014.
1. Reconfigurabilidade. 2.Processadores heterogêneos 3. Compatibilidade Binária. Sistemas multiprocessados I. Carro, Luigi II. Título.

RESUMO

As limitações resultantes do avanço das tecnologias de integração, como o crescente aumento da densidade de potência, levando à necessidade de redução da frequência de operação dos circuitos somados à necessidade de redução do consumo energético, sejam por motivos ecológicos ou para melhor suprir dispositivos portáteis, trazem a necessidade de maior intervenção e personalização do hardware em relação às exigências do software.

Em diversos níveis estas intervenções podem ser aplicadas, onde a granularidade pode variar desde elementos de processamento sendo completamente desativados até processadores tendo apenas unidades funcionais sendo desativadas, memórias cache reconfiguradas em tamanho e associatividade, etc.

Entretanto, a reconfiguração do hardware deve atingir todas as etapas destes sistemas para que seja possível atingir redução satisfatória em termos de potência e consumo de energia.

Além da integração acelerada de elementos de processamento em um mesmo circuito integrado, a crescente concentração de heterogêneas tarefas em um mesmo dispositivo, leva à integração de elementos de processamento também heterogêneos, e por consequência diferentes comportamentos variando de acordo com a aplicação.

Para justificar esta reconfigurabilidade e heterogeneidade dos elementos de processamento este trabalho apresenta um estudo que possibilita a observação da execução de diferentes aplicações em elementos de processamento amplamente reconfiguráveis. Para que a reconfigurabilidade e heterogeneidade possam ser aplicáveis, foi inserida uma ferramenta capaz de manter a compatibilidade entre o elemento de processamento mestre e os elementos de processamento aceleradores reconfiguráveis disponíveis.

Os experimentos apresentados baseiam-se na necessidade de manter a menor quantidade de silício ativa, acelerando o código fonte enquanto reduz-se o consumo de energia. Somada a redução de energia, a compatibilidade binária é levada em consideração buscando a manutenção da produtividade quando da utilização de sistemas heterogêneos reconfiguráveis.

Palavras-Chave: Reconfigurabilidade, heterogeneidade, Sistemas Multiprocessados, Compatibilidade Binária.

ABSTRACT

MULTIPROCESSOR SYSTEM ON CHIP
Reconfigurability and Heterogeneity
Energy Saving and Binary Compatibility

The limitations resulting from the advancement of integration technologies, such as the increasing power density, leading to the need to reduce the operating frequency of the circuits added to the need to reduce energy consumption, whether for environmental reasons or to better serve mobile devices, bring the need for greater intervention and hardware customization to the demands of the software.

To varying degrees these interventions can be applied where the granularity can range from processing elements being completely disabled until processors having only functional units being disabled, reset cache memories in size and associativity, etc. However, the reconfiguration of hardware should reach all stages of these systems so that you can achieve satisfactory reduction in power and energy consumption.

In addition to the accelerated integration of processing elements on a single integrated circuit, the increasing concentration of heterogeneous tasks in a same device, also leads to the integration of heterogeneous processing elements, and therefore different behavior varies according to the application.

To justify this reconfigurability and variety of processing elements this work presents a study that allows the observation of the implementation of different applications in widely reconfigurable processing elements. For reconfigurability and heterogeneity may be applicable, a tool to maintain compatibility between the master processing element and accelerators reconfigurable processing elements available was inserted.

The experiments presented are based on the need to maintain the lowest amount of active silicon, accelerating the source code while reducing power consumption. Added to energy reduction, binary compatibility is taken into consideration seeking to maintain productivity when using reconfigurable heterogeneous systems.

Keywords: Reconfigurability, heterogeneity, Multiprocessors System, Binary Compatibility

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ASIC	- Application Specific Integrated Circuit
EP	- Elemento de Processamento
flit	- Flow Control Digit
FPGA	- Field Programmable Gate Array
ILP	- Instruction Level Parallelism
ITRS	- International Technology Roadmap for Semiconductors
MMI	- Memory Management Interface
MPSoC	- Multiprocessor System on Chip
NI	- Network Interface
NoC	- Network on Chip
TLP	- Thread Level Parallelism
UF	- Unidade Funcional
ULA	- Unidade de Lógica e Aritmética
VHDL	- VLSI Hardware Description Language
VLIW	- Very Large Instruction Word

SUMÁRIO

TODA A TEORIA É TEMPORÁRIA, UMA MELHOR PODE APARECER SEM AVISO. MAS O RESULTADO EMPÍRICO É PERMANENTE, NO MÁXIMO APARECERÁ UM MAIS PRECISO.

"LATTES"

1 INTRODUÇÃO

Em diversas áreas a tecnologia tem avançado mudando a forma como vivemos nossas vidas. O mundo tornou-se dependente da tecnologia, e por consequência a tecnologia passou a ser mais difundida e exigida. Dispositivos que ofereciam poucas funções como aparelhos de TV, telefones móveis e até mesmo simples relógios de pulso, que no máximo contemplavam calendário e calculadora, hoje dispõe de mais tecnologia embarcada do que computadores de uma década atrás, como o Samsung GearS (SAMSUNG, 2014) e o Apple Watch (APPLE, 2014).

Assim, a crescente demanda por maior desempenho é ditada pelas modernas aplicações, cujo aumento na complexidade e na exigência por maior poder de processamento fez com que a tecnologia acelerasse a passos largos nos últimos anos. Aplicações como reconhecimento de voz e face, decodificadores de áudio e vídeo de alta definição, comunicação em tempo real, tanto relacionado à internet como entre dispositivos, além de monitoramento e intervenção de sinais vitais são apenas algumas das aplicações já existentes em dispositivos embarcados. Além disto, os usuários tornam-se cada vez mais exigentes, requerendo o máximo da tecnologia em diversos ambientes. Este constante impulso à tecnologia trouxe a quebra de um paradigma, a "lei de Dennard" (DENNARD, GAENSSLEN, *et al.*, 1974).

A lei de Dennard dita que a potência consumida pelos transistores reduz proporcionalmente à suas dimensões. Isto é, com a redução das dimensões dos transistores e o aumento da densidade destes, previstas na lei de Moore (MOORE, 1965), a tensão e a corrente que alimentam estes dispositivos também reduziriam, e desta forma a potência consumida reduziria significativamente a cada passo tecnológico. Entretanto, com a redução das dimensões físicas dos transistores algumas limitações foram alcançadas, dentre elas a redução da tensão não seguiu a mesma proporção, e em consequência a potência consumida por estes dispositivos não escalou suficientemente com a tecnologia. Em virtude disto, estes dispositivos não conseguem mais operar em frequência máxima teórica.

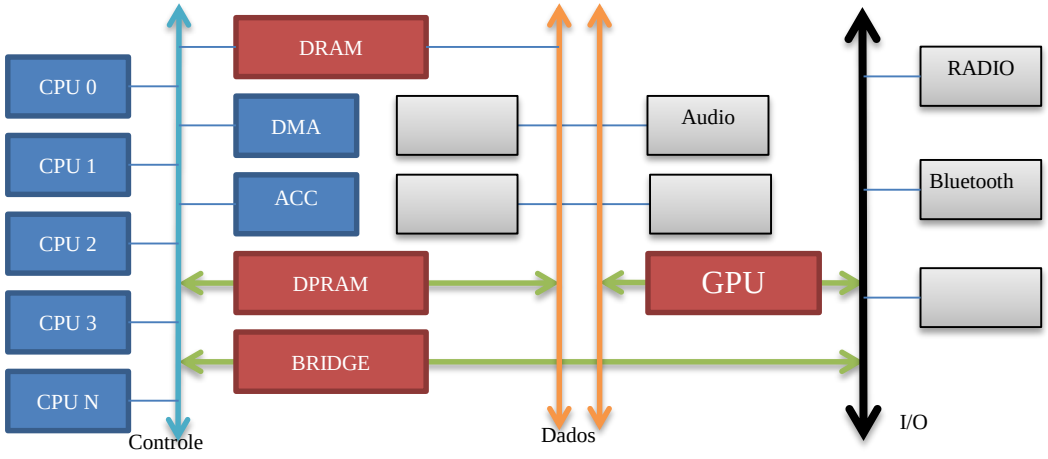
Sendo assim, desde o início dos anos 2000 os projetistas de processadores incrementam o número de núcleos de processamento, visando explorar o aumento de densidade dos transistores ditado pela lei de Moore, aumentando o poder de processamento sem a necessidade de operar na máxima frequência destes componentes. Mantendo, de certa forma, o consumo de potência em níveis aceitáveis sem danos ao desempenho. Desta forma,

devido às exigências de desempenho e o advento de novas tecnologias de manufatura de transistores (da ordem de algumas dezenas de nanômetros) um maior número de elementos computacionais passou a ser integrado em um mesmo circuito. Esta integração possibilitou o agrupamento de sistemas completos em um único chip (BENINI e DE MICHELI, 2002) (KUMAR, 2002), conhecidos como SoC (do inglês, *system-on-chip*). O nível computacional destes circuitos aumenta conforme o número de elementos. Assim, diferentes elementos de processamento (EPs) passaram a integrar estes sistemas, formando sistemas multiprocessados chamados MPSoCs (do inglês, *Multiprocessor System on Chip*). Estes sistemas multiprocessados são compostos por diversos EPs, podendo constituir MPSoCs homogêneos ou heterogêneos.

A inclusão dos vários EPs traz alguns desafios importantes, como a comunicação entre estes elementos. Apesar dos barramentos tradicionais terem evoluído (ARM, 2013) (RYU, SHIN e MOONEY, 2001) (KUMAR, 2002), o volume de dados e a quantidade de dispositivos que efetivamente comunicam-se entre si tornam a solução tradicional incapaz de satisfazer as demandas. Assim, redes em chip out NoC (do inglês *Network-on-Chip*) surgiram como solução para esta demanda (BENINI e DE MICHELI, 2002).

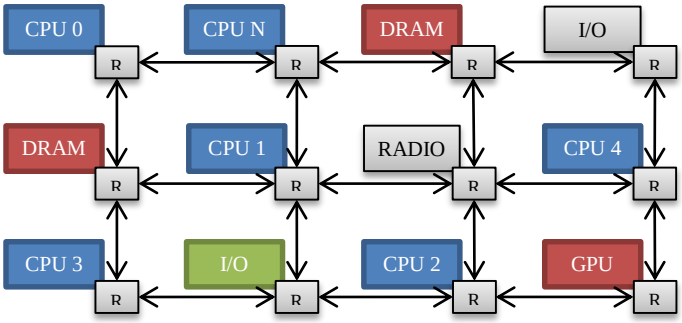
As Figuras 1.1 e 1.2 ilustram barramentos e NoCs, onde é possível visualizar a diferença entre as duas soluções. A característica que aparece de forma mais acentuada entre as duas possibilidades mostradas é a escalabilidade que surge como uma dos maiores benefícios da NoC. Fora essa característica, o reduzido consumo de energia e a menor latência são outras características importantes das NoCs (AGARWAL, 2009)

Figura 1.1: Exemplo de Barramento



Fonte: adaptado de (ARTERIS, 2005)

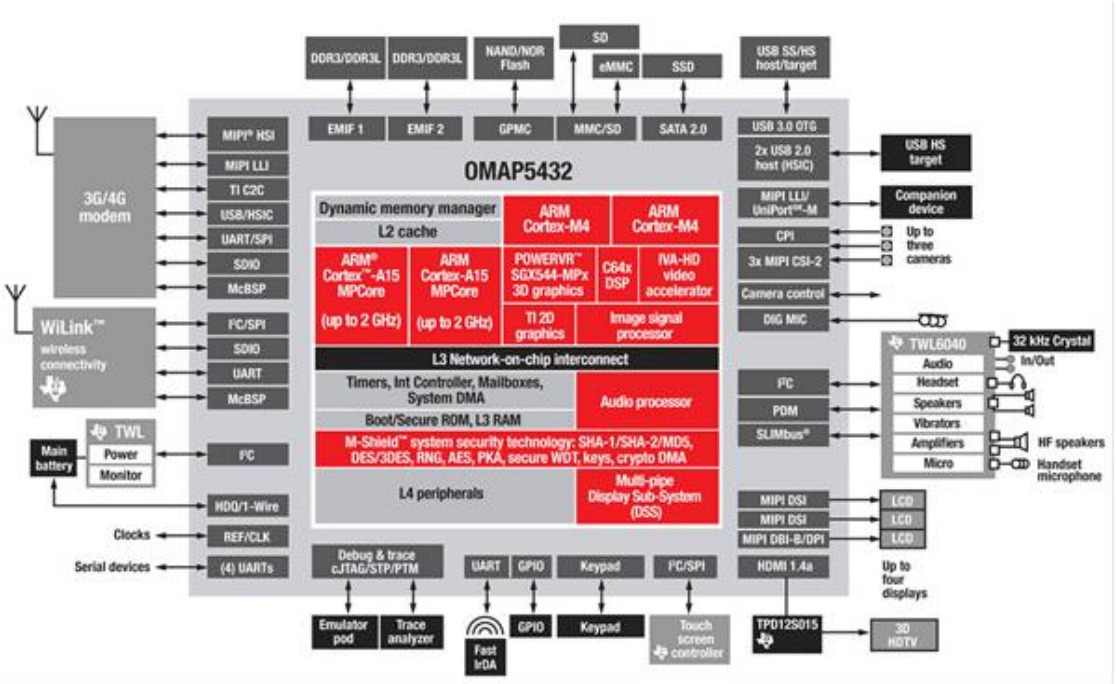
Figura 1.2: Exemplo de Rede em Chip



Fonte: adaptado de (ARTERIS, 2005)

Em virtude da difusão da tecnologia e o aparecimento de novas necessidades ditadas pela vida moderna, dispositivos móveis, como por exemplo, os *smartphones* (telefones móveis inteligentes) e *tablets*, passaram a receber novas funções e consequentemente requerer maior poder de processamento. Esta escalada tecnológica levou ao aparecimento de MPSoCs cada vez mais complexos, como o OMAP 5 da Texas Instruments (TI, 2013) e o Nvidia TegraK1 (NVIDIA, 2014). Processadores compostos por 72 núcleos já são realidade como o Tilera GX que comporta processadores VLIW (TILERA, 2012). Outros como o MPSoC Intel SCC (*Single-chip Cloud Computer*) composto por 24 processadores x86 de dois núcleos cada, surgem como uma promessa para dispositivos portáteis. A figura 1.2 mostra o diagrama do MPSoC OMAP 5432 da Texas Instruments, o qual utiliza uma NoC para conectar diversos elementos de processamento.

Figura 1.3: Diagrama do MPSoC TI OMAP5432



Fonte: (TI, 2013)

Além dos desafios em hardware provenientes da composição dos MPSoCs com diversos EPs, o software necessita acompanhar estas evoluções e tem sua complexidade aumentada proporcionalmente. Quanto mais complexo o MPSoC, mais complexo será manter a programabilidade e a compatibilidade com binários já estabelecidos. MPSoCs heterogêneos trazem a condição de que EPs diferentes precisam se comunicar, onde tamanho de palavras de instruções e dados, ordenação dos bytes, frequência de operação, protocolos de comunicação, etc., podem não ser compatíveis tornando as tarefas de programabilidade e compatibilidade críticas, o que traz a importante questão da produtividade relacionada ao aumento desta complexidade.

1.1 Descrição do Problema

A expansão do número de elementos de processamento já alcançando dezenas de unidades (INTEL, 2013), e prevista para centenas de unidades integradas em um único chip (HENKEL, 2003) somado ao avanço da tecnologia, com transistores alcançando poucos nanômetros nos próximos anos, fará com que limites críticos principalmente relacionados ao consumo de potência sejam alcançados, mesmo que a frequência de operação máxima não seja aplicada. Desta forma, será impossível não apenas trabalhar com a frequência máxima, mas até mesmo manter um razoável percentual do chip ativo. Com estes limites aproximando-se graças à queda da lei de Dennard e a manutenção da lei de Moore, a densidade de potência passa a ser um grande desafio.

Do lado da manufatura dos dispositivos básicos, cientistas estão em constantemente busca de novos materiais capazes de consumir menos potência enquanto disponibilizam o desempenho requerido. Materiais como os nanotubos de carbono, o grafeno (SHULAKER, HILLS, *et al.*, 2013) e a molibdenita (LOPEZ-SANCHEZ, LEMBKE, *et al.*, 2013) aparecem como algumas promessas. Entretanto, enquanto o custo destes novos materiais não permitir sua utilização, soluções devem ser desenvolvidas pelos projetistas de hardware principalmente no âmbito da arquitetura.

Previsões apresentadas pelo ITRS (do inglês, *International Technology Roadmap for Semiconductors*) mostram que a densidade de potência para as tecnologias futuras cresce mais do que a densidade de transistores para futuros chips. A Tabela 1 ilustra a previsão para o número de transistores por área, consumo de potência estática e a frequência de operação (ITRS, 2013).

Tabela 1.1: Previsão do consumo de potência para futuras tecnologias

Ano	**2010**	**2014**	**2016**	**2018**	**2020**	**2023**	**2026**
Nó Tecnológico (nm)	45	22	18,9	15	11,9	8,4	6
Milhões de Transistores / cm²	1701	4284	6806	10804	17150	34300	68600
Potência Estática*	1	1,27	2,18	2,91	-	-	-
Frequência (GHz)	3,74	3,89	4,55	4,97	5,33	5,99	6,74

*Normalizado para tecnologia de 45nm

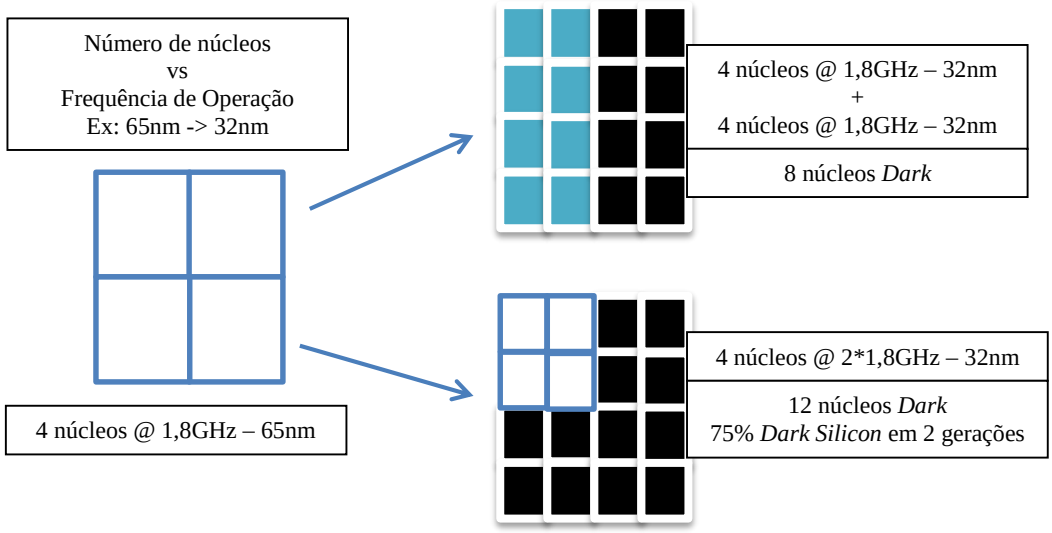
Fonte: (ITRS, 2013)

Com a densidade de potência prevista para futuras tecnologias, a quantidade de transistores que poderão ser mantidos ativos simultaneamente reduzirá para pequenos percentuais (ESMAEILZADEH, BLEM, *et al.*, 2011). A necessidade de redução da área ativa não ocorre apenas por simples consumo energético, mas também para a manutenção da vida útil do circuito integrado. Esta porção de silício (porção de circuito) que precisa ser mantido inativo é chamada de *Dark Silicon* (ESMAEILZADEH, BLEM, *et al.*, 2011). A figura 1.4 ilustra a evolução do *Dark Silicon* e a escolha da indústria quando a tecnologia escala de 65nm para 32nm.

Uma forma de contornar o problema *DarkSilicon* é a utilização agendada dos elementos de processamento, isto é, EPs muito próximos ou que interfiram entre si não podem ser ativados simultaneamente. Entretanto, esta solução pode acarretar em restrições quanto ao uso de aplicações, o que desmotiva o uso de múltiplos processadores nesta escala.

Considerando-se que, quanto ao hardware, a heterogeneidade e os múltiplos elementos de processamento alcancem um patamar aceitável na relação desempenho/energia, outra necessidade deve ser considerada: a necessidade de programar este hardware.

Figura 1.4: Influência do *Dark Silicon* na escalada de multinúcleos



Fonte: (GOULDING, SAMPSON, *et al.*, 2010)

Com a aceleração do consumo destes dispositivos e a urgência que o *time-to-market* representa para qualquer produto, novas abordagens que envolvam a utilização de muitas ferramentas ou que levem a mudanças radicais e não sejam capazes de manter uma retro compatibilidade binária não são bem vistas. Assim, em termos de software dois fatores são determinantes: a facilidade de programar novos sistemas computacionais e a manutenção da compatibilidade binária.

1.2 Escopo deste Trabalho

O presente trabalho aborda a reconfigurabilidade de elementos de hardware objetivando encontrar a melhor relação possível entre consumo de energia e desempenho. Diversas técnicas buscando alcançar este objetivo são apresentadas na literatura. Entretanto, poucos trabalhos apresentam um estudo onde sistemas completos, como os MPSoCs, utilizam técnicas de reconfiguração de hardware de forma a combinar desempenho e baixo consumo energético em todas as etapas que os compõe: Processamento, Memória e Comunicação. Além disto, este trabalho dá um foco particular à reconfigurabilidade de ASICs (do inglês, *Application Specific Integrated Circuits*), diferentemente da inserção de novos componentes hardwares baseados em FPGAs, mas não as descartando.

Além de abordar a reconfigurabilidade dos principais componentes de um MPSoC, este trabalho aborda também a programabilidade destes componentes dotados de reconfigurabilidade. Esta abordagem considera a manutenção da compatibilidade entre diferentes processadores que compõem os MPSoCs heterogêneos e apresenta uma solução onde a reconfigurabilidade é utilizada de forma inteligente para acelerar o desempenho e reduzir o consumo de potência.

1.3 Contribuições deste Trabalho

O trabalho mostra que é possível encontrar configurações ótimas de hardware para cada aplicação que se deseja executar. O trabalho mostra ainda que a aceleração de um código, previamente compilado para determinada arquitetura, pode ser acelerado por uma arquitetura diferente ao mesmo tempo em que provê economia de energia. A manutenção da programabilidade e a compatibilidade binária entre processadores com arquiteturas difundidas e processadores aceleradores heterogêneos e reconfiguráveis também é apresentada neste trabalho.

1.4 Organização deste Trabalho

O capítulo 1 introduziu o trabalho que será apresentado, além de ilustrar a problemática alvo deste estudo. O capítulo 2 apresenta uma abrangente pesquisa visando ilustrar as técnicas tradicionalmente aplicadas aos circuitos integrados objetivando redução de energia. O capítulo 3 abrange ainda mais as técnicas de redução do consumo energético apresentando algumas técnicas de reconfigurabilidade de hardware.

No capítulo 4 são apresentados os sistemas multiprocessados que dão início ao objetivo deste trabalho, mostrando ainda comparações entre sistemas estáticos e reconfiguráveis. Os capítulos 5 e 6 ampliam o ambiente de trabalho mostrado no capítulo 4, trazendo a reconfigurabilidade e a heterogeneidade para o contexto da redução energética atacando os problemas apresentados no capítulo 1. Os capítulos 5 e 6 apresentam ainda resultados dos experimentos realizados nestes sistemas. O capítulo 7 objetiva ilustrar o uso da ferramenta que facilita a utilização dos sistemas heterogêneos e reconfiguráveis, mostrando resultados e comparações relacionadas ao uso desta ferramenta.

2 TÉCNICAS PARA REDUÇÃO DO CONSUMO DE ENERGIA

A procura pela melhor relação entre desempenho e energia (e potência) é uma constante no projeto de circuitos integrados, principalmente quando os alvos são sistemas embarcados. Entretanto, um clássico problema é ainda um dos mais importantes: área ociosa. Quanto maior a área de um chip (em termos de número de transistores) maior será o número possível de unidades funcionais e maior será seu desempenho, isto é, mais capacidade de processamento. O problema se mantém principalmente quando não é possível ocupar todas as unidades em tempo integral, resultando em desperdício de energia.

A redução no consumo de potência faz-se necessária tanto objetivando redução dos custos de manufatura (encapsulamento, dissipação, etc.) quanto visando menor consumo energético (maior duração para baterias, questões ambientais, etc.). Os desafios para redução do consumo de potência aumentam juntamente com o crescimento da densidade dos dispositivos básicos, escalada da tecnologia, aumento da frequência de operação, redução da tensão de alimentação e da tensão de limiar (*threshold*) dos transistores.

Diversas abordagens são utilizadas para reduzir o consumo energético e de potência nos circuitos integrados. Com o provimento de maior integração e o crescimento da demanda por equipamentos eletrônicos portáteis cada vez mais sofisticados e complexos, estas preocupações tornaram-se mais urgentes. Assim, técnicas estáticas (estruturas das portas lógicas, por exemplo) e técnicas dinâmicas como o DVFS (do inglês, *Dynamic Voltage and Frequency Scaling),* por exemplo, são aplicadas desde a concepção do projeto.

2.1 Técnicas Estáticas

Em tempo de projeto é natural a busca por redução de custos de manufatura, como encapsulamento. Assim, limitar a dissipação (e o consumo) de potência reduz a necessidade de elementos adicionais para controle de temperatura, por exemplo, além de aumentar a confiabilidade e durabilidade do circuito.

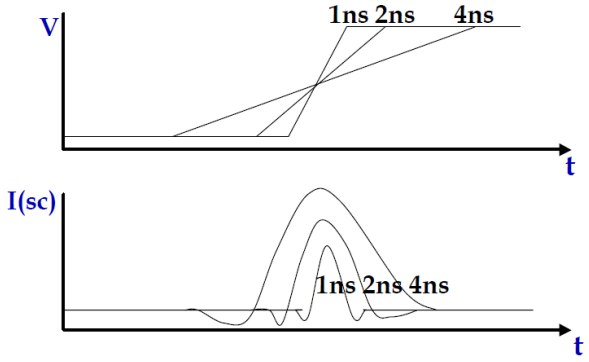
2.1.1 Dimensionamento dos Dispositivos básicos

O adequado dimensionamento dos transistores buscando a redução no consumo da potência é uma das técnicas utilizadas. Com o correto dimensionamento é possível prover chaveamento com menor *slack time* (diferença entre tempo requerido e tempo de subida de

um sinal na saída de uma porta lógica), o que provê redução no consumo de potência dinâmica.

Outra questão importante que pode ser minimizada com o correto dimensionamento dos transistores é o *slope signal* (inclinação do sinal de entrada). Quanto menor o tempo de subida de um sinal (de zero à um lógico) menor será o *slope* e consequentemente menor o consumo de potência por curto-circuito (RABAEY, 2005). A figura 2.1 ilustra tal variação, mostrando que a corrente consumida é diretamente relacionada ao *slope*.

Figura 2.1: Influência da inclinação do sinal de entrada no atraso e consumo



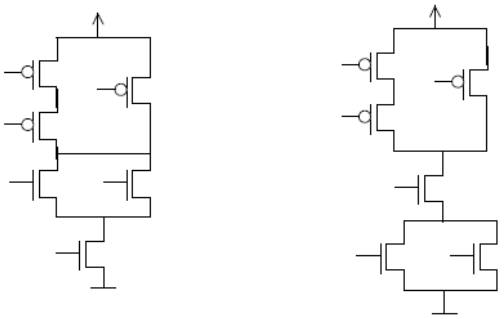
Fonte: adaptado de (RABAEY, 2005)

Além disto, fazendo com que os transistores que chaveiam com maior frequência tenham suas dimensões reduzidas, reduz-se a potência dinâmica por chaveamento.

2.1.2 Reestruturação das portas lógicas

A correta reestruturação das portas lógicas e das redes de portas lógicas é outra forma de modificar as características de atraso e potência de um circuito. A Figura 2.2 ilustra portas lógicas equivalentes que possuem diferentes características elétricas, e por consequência de *timing*.

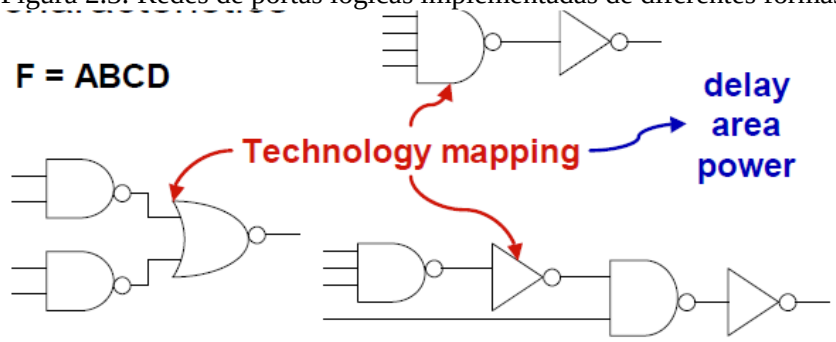
Figura 2.2: Portas lógicas implementadas de diferentes formas



Fonte: adaptado de (RABAEY, 2005)

Aumentando a granularidade dos dispositivos, na Figura 2.3 pode ser visto que redes de portas lógicas também podem ser reestruturadas buscando melhor relação desempenho/potência. Desta forma análises e ajustes ainda em tempo de síntese podem contribuir com a redução no consumo de potência.

Figura 2.3: Redes de portas lógicas implementadas de diferentes formas

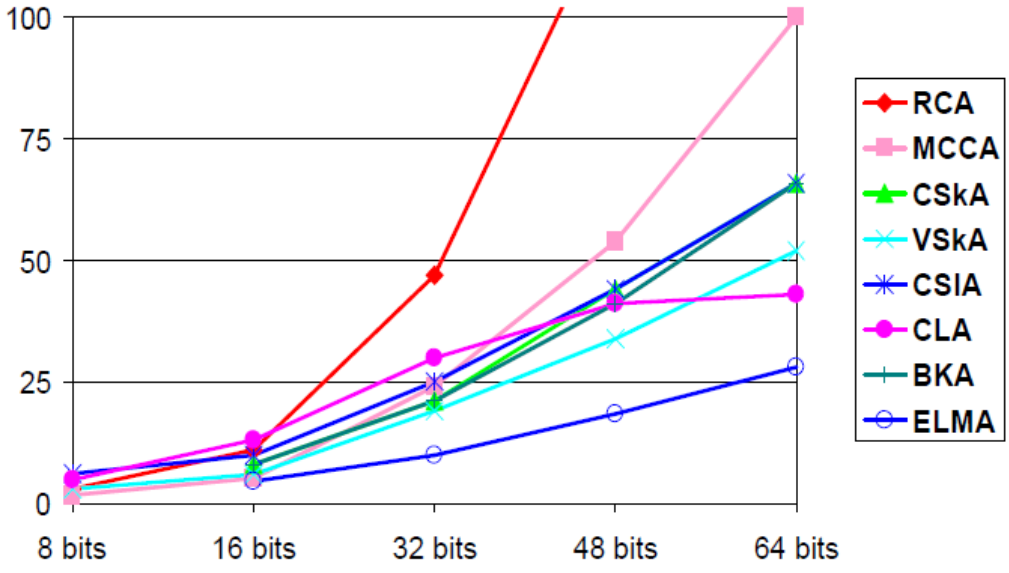


Fonte: adaptado de (RABAEY, 2005)

Dispositivos como *flipflops* e *latches*, assim como os componentes das unidades lógicas e aritméticas (ULA) e multiplicadores também podem ser implementados de forma a consumir menor potência (KUMRE, SOMKUWAR e AGNIHOTRI, 2013). A Figura 2.4 ilustra o produto entre potência e atraso – (PDP do inglês *power delay product*) para diferentes implementações de ULAs.

Na Figura 2.4 é possível notar que com o aumento da largura de dados aumenta também o PDP para todas as implementações do exemplo, entretanto a arquitetura CLA varia pouco a partir de 48bits. Mostrando que dependendo da necessidade, diferentes decisões devem ser tomadas em tempo de projeto para amenizar ao máximo o consumo de potência.

Figura 2.4: Produto de potência e atraso para diferentes arquiteturas de somadores



Fonte: adaptado de (NAGENDRA, IRWIN e OWENS, 1996)

2.1.3 Memórias

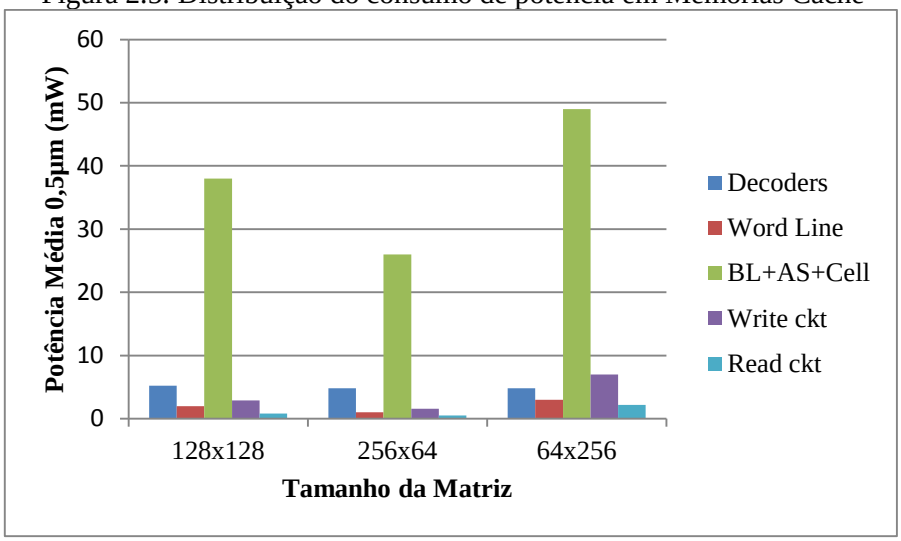
Memórias são críticas para os sistemas embarcados, principalmente referindo-se às memórias cache. Primeiramente por causa do desempenho: é necessário que esta memória seja capaz de acompanhar o processador para que não seja vista como gargalo do sistema. Por outro lado, como normalmente sua área é grande, o consumo de potência de todo o sistema acaba sendo influenciado por suas características.

Para que as memórias cache sejam capazes de acompanhar o desempenho dos processadores ao qual são acopladas, estas são baseadas em células SRAM (do inglês, *Static Random-Access Memory*). Estas memórias são capazes de alto desempenho, mas em contrapartida são grandes consumidoras de potência.

Assim, em tempo de projeto, algumas técnicas devem ser aplicadas para extrair o máximo destes dispositivos com o menor consumo possível, principalmente quando em operação. A Figura 2.5 ilustra as regiões críticas nas memorias SRAM para uma tecnologia de 0,5µm. Para tecnologias atuais, da ordem de 65nm, as proporções se mantêm válidas (KANDEMIR, YEMLIHA, *et al.*, 2010).

As técnicas mais aplicadas às memórias cache baseadas em SRAM são a partição dos bancos de memória, a divisão das linhas de *word*, segmentação da linhas de *bit*, a aplicação de linhas de *word* pulsadas e além do isolamento das linhas de *bit*.

Figura 2.5: Distribuição do consumo de potência em Memórias Cache



Fonte: adaptado de (CHANG e GARCIA-MOLINA, 1997)

Analisando brevemente a figura 2.5 é possível perceber que o conjunto linha de bits (BL), o *sense amplifier* e as células de memória são os que demandam maior consumo de potência, desta forma técnicas que atacam estes elementos devem ser levadas em consideração. A técnica que abrange estes pontos é a partição dos bancos de memória, isto é, dividindo-se a memória em bancos menores leva à ativação de menos células por vez. A ativação de menos células aumenta o desempenho, reduz o consumo de potência e a capacitância nas linhas de *word*.

2.1.4 Barramentos

Os barramentos são amplamente utilizados tanto para possibilitar comunicação *intrachip* (dento de um mesmo chip, como entre o núcleo do processador e memórias de nível L1) como para efetivar comunicação entre processadores e dispositivos exteriores, como memórias RAM e outros componentes externos ao processador.

Apesar da larga utilização dos barramentos, estes são uma significante fonte de consumo de potência, devido ao chaveamento, que ocorre toda vez que uma nova informação é transmitida, e à grande capacitância formada pelos mesmos.

Diversas técnicas são utilizadas para minimizar o consumo que a transmissão de dados impõe aos circuitos integrados:

- Redução da atividade de chaveamento nos barramentos através de codificação das informações;
- Redução da variação da voltagem utilizando sinalização diferencial
- Diferentes estruturas de barramentos
- Minimizar o tráfego de dados através de compressão de código

As técnicas aplicadas aos barramentos minimizam o consumo de potência destes elementos, entretanto quanto maior o número de dispositivos acoplados, menor será a eficiência destas técnicas e do próprio barramento.

2.2 Técnicas Dinâmicas

Além das técnicas estáticas, apresentadas na Seção 2.1, técnicas dinâmicas surgem como auxiliares na redução no consumo de potência. Estas técnicas são aplicadas em tempo de execução, isto é, durante o funcionamento do circuito modificações de características de funcionamento são alteradas resultando em redução no consumo energético.

2.2.1 DVFS

Com o controle dinâmico de voltagem e de frequência de operação dos circuitos (DVFS, do inglês, *Dynamic Voltage and Frequency Scale*) é possível reduzir o consumo de energia de um grande leque de componentes de hardware. A frequência de operação tem impacto linear sobre o consumo de energia, enquanto a voltagem tem impacto quadrático, sendo o fator com maior relação com o consumo energético. Assim, quando estas duas variáveis são atacadas, atinge-se uma maior redução no consumo.

Com o avanço tecnológico reduzindo a largura de canal dos transistores a níveis nanométricos surge também o aumento da integração de componentes, fazendo com que a aplicação de técnicas como o DVFS torne-se cada vez mais importante. Entretanto, o avanço da tecnologia resulta também em maior dificuldade de manter as especificações precisas durante os processos de fabricação, causando variações nos parâmetros físicos e elétricos dos circuitos. Estas variações podem resultar em falhas no que tange a relação frequência/potência e voltagem/potência (HERBERT e MARCULESCU, 2009). Além das dificuldades em manter estes processos precisos, a redução da largura dos canais dos transistores reduz também a tensão de alimentação destes componentes. Esta tensão de alimentação vem aproximando-se, nominalmente, da tensão de limiar destes transistores, deixando pouca margem para que o DVFS possa, em termos de tensão, auxiliar na redução do consumo de potência e energia (GARG, MARCULESCU, *et al.*, 2009).

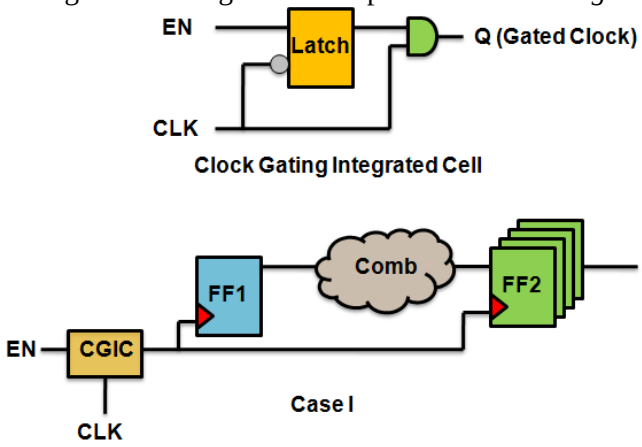
Apesar da redução da tensão de alimentação influenciar negativamente no desempenho, é importante a realização de análises e até mesmo alterações na estrutura das portas lógicas para que seja possível tal redução já que existe uma relação quadrática entre tensão e potência.

2.2.2 Clock Gating

A árvore de relógio é o sinal cujo chaveamento é o de maior frequência em um circuito integrado. Este chaveamento é necessário para que dispositivos permaneçam sincronizados, como os *flip-flops*. Entretanto, devido ao longo caminho percorrido pelo sinal de relógio e a capacitância formada por este caminho, a potência dinâmica consumida por este sinal pode chegar a 50% da potência total de um sistema em *chip* (KATHURIA, AYOUBKHAN e NOOR, 2011).

Mesmo quando etapas de um circuito integrado não estão em uso, o sinal de relógio mantém incessantemente o chaveamento aumentando o consumo energético. Assim, surge a necessidade da aplicação de técnicas que reduzam o consumo de potência destes sinais.

Figura 2.6: Diagrama Exemplo do *Clock Gating*



Fonte: adaptado de (GUPTA, 2009)

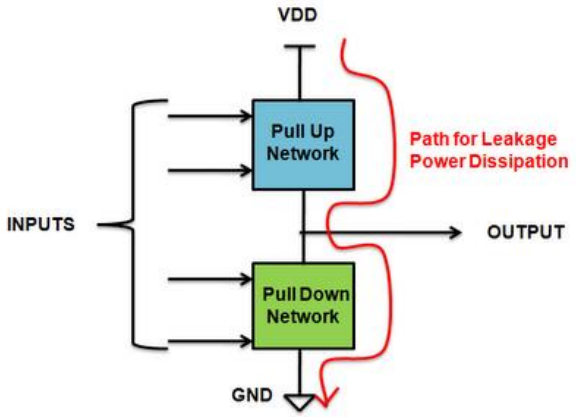
O C*lock Gating* é uma destas técnicas que especificamente controla o sinal de relógio do circuito, possibilitando que este chegue apenas aos dispositivos que necessitam em um dado momento.

A Figura 2.6 ilustra de forma simplificada o controle do caminho de relógio. Através do sinal EN (*enable*) é possível controlar a propagação do sinal de relógio (clk) fazendo com que este chegue a componentes ou mesmo grupo de componentes apenas quando estes forem utilizados.

2.2.3 Power Gating

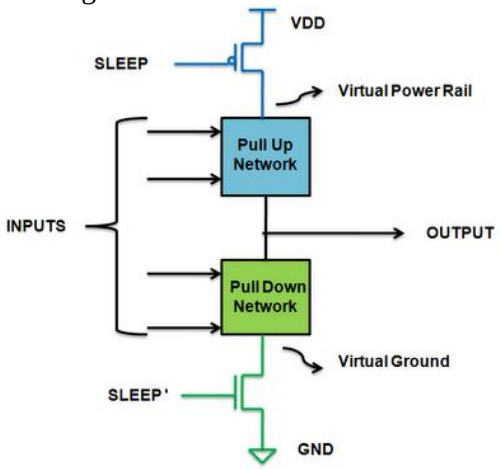
Power Gating, ou chaveamento da alimentação, é outra efetiva técnica empregada em projetos de baixa potência. Diferentemente do C*lock Gating*, que se ocupa da redução de potência dinâmica, o *Power Gating* visa redução de potência estática. A potência estática torna-se cada vez mais crítica juntamente com a redução dos canais dos transistores, principalmente a partir de poucas dezenas de nanômetros.

Figura 2.7: Diagrama de circuito básico sem *Power Gating* – ênfase para a potência



Fonte: adaptado de (GUPTA, 2009)

Figura 2.8: Diagrama de circuito básico com *Power Gating*



Fonte: adaptado de (GUPTA, 2009)

Basicamente todo o circuito que utiliza lógica digital CMOS consiste de uma rede de transistores PMOS (rede *Pull-Up*) e outra rede constituída de transistores NMOS (rede *Pull-Down*), como mostrado na Figura 2.7.

Em algum momento, se existir um caminho direto entre os terminais da fonte de tensão (*VDD* e *Ground*), por exemplo, o circuito dissipará potência. Esta potência é gerada pela corrente que circula diretamente entre os terminais da fonte, e é conhecida como *leakage* (ou potência de fuga). O controle destas redes, visando minimizar esta potência de fuga, é a razão do *Power Gating* (LUNGU, BOSE, *et al.*, 2009).

A Figura 2.8 mostra de forma simples como transistores estrategicamente aplicados podem contribuir com o controle destas redes.

Basicamente, o funcionamento deste circuito se dá da seguinte forma:

- Durante a operação normal, o sinal *SLEEP* é carregado com valor Zero. Ambos os transistores de controle *SLEEP* ficam ativos, provendo uma fonte virtual para as redes PMOS e NMOS que funcionam normalmente.
- Em períodos de baixa atividade o sinal *SLEEP* é carregado com valor UM, desativando os transistores de controle. Assim, as redes de transistores PMOS e NMOS são desabilitadas cessando a potência de fuga.

É importante notar que, durante a operação normal, os transistores *Sleep* contribuem com um consumo extra de potência estática, já que estes transistores permanecem sempre ativos. Embora isto ocorra, a potência estática destes transistores é extremamente menor quando comparado às redes em que são ligados, além disto são projetados com alto valor de tensão de limiar (*threshold*), o que reduz o consumo de potência estática em modo normal de utilização.

2.3 Considerações

Existem diversas outras técnicas que são apresentadas na literatura, assim como utilizadas na indústria. Entretanto, o objetivo deste capítulo é apenas apresentar as técnicas mais comuns e já difundidas tanto no meio acadêmico quando no meio industrial. Assim, apesar das inúmeras técnicas existentes, é possível observar que o avanço tecnológico e a demanda por maior desempenho com menor consumo energético trazem necessidade de outras abordagens, como brevemente explicado no capítulo introdutório. Um das abordagens que tem surgido de forma interessante é a reconfigurabilidade de elementos, apresentado no Capítulo 3.

3 HARDWARE RECONFIGURÁVEL

As técnicas apresentadas no capítulo 2 são, em sua maioria, aplicadas em tempo de projeto e estáticas. Apesar de essenciais para a economia de energia, o dinamismo das aplicações, a crescente integração e a aceleração na miniaturização dos transistores exigem técnicas dinâmicas de redução do consumo. A partir desta percepção, diversas técnicas buscando um funcionamento mais eficiente dos sistemas eletrônicos foram elaboradas.

Com o avanço da tecnologia algumas técnicas de redução de potência chagam ao limite de sua eficiência, como o caso do DVFS. Desta forma a simples redução da tensão e/ou da frequência de operação destes componentes não é suficiente. Surge assim a necessidade de desativar parcialmente ou integralmente estes componentes. Algumas técnicas vão além, transferindo todo o contexto que está em execução em um processador maior para um processador menor reduzindo assim o consumo energético (ARM, 2013).

3.1 Processadores Reconfiguráveis

Em contraste com processadores estáticos, os processadores reconfiguráveis tem a arquitetura de hardware, assim como as instruções suportadas, capaz de ser modificada dinamicamente. Isto significa que o tipo e a quantidade de circuitos que implementam algumas instruções podem ser modificados após a fabricação do processador, inclusive durante seu funcionamento.

O projeto de processadores reconfiguráveis obedece principalmente a três paradigmas: processador anexo, coprocessador e unidades funcionais. O paradigma do processador anexo refere-se à logicas reconfiguráveis conectadas ao processador principal via barramento de entrada e saída (I/O), por exemplo, o barramento PCI. O processador principal controla a operação da lógica reconfigurável via barramento, sendo possível o tráfego de dados (BARAT e LAUWEREINS, 2000) (GOLDSTEIN, SCHMIT, *et al.*, 2000). No paradigma do coprocessador a lógica reconfigurável é conectada diretamente ao processador principal, semelhante a um coprocessador de ponto flutuante (BARAT e LAUWEREINS, 2000) (HAUSER e WAWRZYNEK, 1997). Quando a lógica reconfigurável funciona como uma unidade funcional disposta dentro do processador, temos o paradigma das unidades funcionais (BARAT e LAUWEREINS, 2000) (ISELI e SANCHEZ, 1993) (RAZDAN e SMITH, 1994).

3.1.1 Superescalar

Processadores superescalares são capazes de executar diversas instruções de forma simultânea graças à quantidade de unidades funcionais disponíveis, entretanto estas unidades são fixas e previamente conhecidas. Os processadores superescalares reconfiguráveis tem sido propostos visando diferentes finalidades: desde aumento de desempenho à confiabilidade (KOTTKE e STEININGER, 2007). Em (NIYONKURU e ZEIDLER, 2004) um processador superescalar de uso geral é proposto com o objetivo de manter compatibilidade de software enquanto acelera aplicações específicas cujas restrições de tempo são cruciais. O processador de uso geral não é capaz, por si só, de executar tais aplicações específicas dentro dos limites requeridos, desta forma um FPGA é anexado ao caminho de dados do processador, como uma unidade funcional. Basicamente, a unidade FPGA é reconfigurada em tempo de execução, disponibilizando unidades funcionais capazes de atender os requerimentos da aplicação.

3.1.2 VLIW

Os processadores VLIW (*Very Large Instruction Word*) são capazes de explorar paralelismo em nível de instruções, assim como os processadores superescalares, graças às diversas unidades funcionais independentes que o constituem. Entretanto, diferente dos superescalares, não é capaz de atribuir as operações dinamicamente para as unidades de execução disponíveis, possuindo grande dependência do compilador.

Diversos trabalhos apresentam processadores VLIW reconfiguráveis. Em (KOESTER, LUK e BROWN, 2008) é apresentada uma técnica de descoberta da melhor configuração para um VLIW *softcore* e em seguida é gerado o código deste VLIW que será implementado em FPGA. O objetivo é "montar" o processador VLIW ideal para determinada aplicação, consumindo a menor quantidade de recursos e reduzindo o consumo de energia. De forma semelhante, (BROST, YAN e PAINDAVOINE, 2007) apresentam um algoritmo que automaticamente encontra e gera um modelo em VHDL, que consome o mínimo de recursos para executar a aplicação alvo. Este VHDL gerado é baseado em um processador VLIW previamente conhecido, e os módulos gerados são implementados em FPGA compatíveis com a ISA do VLIW original.

O uso de FPGA em processadores reconfiguráveis torna-se custoso em virtude do tempo de reconfiguração destes dispositivos. Em (LODI, TOMA, *et al.*, 2003) é apresentado um processador VLIW com uma unidade funcional reconfigurável. Esta unidade é composta por células lógicas reconfiguráveis fazendo com que o processador seja capaz de suportar

instruções originalmente não suportadas. O processador, juntamente com as células reconfiguráveis, foi sintetizado com a tecnologia de 180nm, isto é, evitando as limitações de uma reconstrução completa de todo o processador em FPGA. Entretanto, apesar de acelerar a execução de algumas aplicações com significativa redução de energia, as células reconfiguráveis, assim como as FPGAs, tem sua frequência de operação limitada.

3.2 Memórias Reconfiguráveis

Memórias são dispositivos com alto consumo de potência (MALIK, MOYER e CERMAK, 2000) (SEGARS, 2001). As memórias cache de instruções e dados são especialmente mais críticas, principalmente para os sistemas modernos. Nos sistemas modernos, assim como nos processadores, minimizar a área de silício é um dos focos, mas ao mesmo tempo procura-se reduzir a distância entre importantes dispositivos com o objetivo de melhorar desempenho e minimizar o desperdício de potência despendida na comunicação entre eles, como no caso de processadores e memórias cache que atualmente são integrados em um mesmo chip. Esta integração, que busca a manutenção do desempenho, gera uma maior densidade de potência. Nos processadores modernos memórias cache de níveis 1, 2 e até mesmo de nível 3 são integrados juntamente com o processador, aumentando a densidade de transistores ativos e consequentemente a densidade de potência.

Seguindo o raciocínio dado aos processadores na Seção 0, a simples redução da tensão e da frequência de operação destes elementos de hardware não é suficiente para uma redução satisfatória do consumo de potência. Assim, ajustes em tempo de execução ou previamente ao início da execução de uma aplicação surgem como possibilidades interessantes.

3.2.1 Memórias Cache

Em 1999, Reinman e seus colegas (REINMAN e JOUPPI, 1999) apresentaram um estudo mostrando que a potência de uma memória cache organizada com mapeamento direto pode ser de apenas 30% de uma memória de mesmo tamanho com associatividade de 4 vias. Assim, diversos estudos posteriores surgiram com o intuito de alcançar uma configuração para memórias cache que atingisse a melhor relação desempenho/potência. Entretanto, diferentes aplicações possuem diferentes comportamentos, e para que estas aplicações utilizem as caches de forma ideal (ou próximo da ideal), estas requerem diferentes configurações.

Além da densidade de potência, outra questão importante quanto às memórias cache é a taxa de *hit*. Quanto maior o tempo de espera por parte do processador por dados ou instruções maior será o desperdício de energia, então a redução do número de *miss*, tanto para instruções quanto para dados, deve ser reduzido. Desta forma, devido à dinâmica comportamental das aplicações e a busca pelo uso ótimo destas memórias, surgiram caches reconfiguráveis em tempo de execução.

A reconfiguração de memórias cache em tempo de execução aparece em diferentes estudos, e basicamente são apresentadas de forma a modificarem alguns parâmetros como tamanho, associatividade e largura dos blocos.

3.2.1.1 Tamanho

Utilizar apenas os transistores essenciais para executar determinada aplicação visando redução de potência é intuitivo, assim como manter ativa apenas a quantidade de memória cache necessária para necessária para determinada aplicação. Quando caches muito grandes estão disponíveis e pouco desta memória é efetivamente utilizada ocorre um grande desperdício de energia. Assim, a reconfiguração da quantidade de memória ativa pode reduzir o consumo de energia de forma significativa desde que não reduza a relação desempenho/potência (BENITEZ, MOURE, *et al.*, 2010). A reconfiguração do tamanho da memória cache pode ocorrer independentemente de outros parâmetros.

3.2.1.2 Associatividade

As diferentes associatividades possíveis de serem implementadas em memórias cache tendem a beneficiar um ou outro tipo de aplicação. Isto é, as aplicações podem constituir uma maior localidade espacial ou temporal, assim caches diretamente mapeadas ou caches totalmente associativas contribuiriam com uma ou outra solicitação. Entretanto as aplicações diferem entre si, sendo assim, ou trabalha-se com uma organização média capaz de ser apenas suficiente para a maioria das situações ou utiliza-se de recursos onde é feita uma constante busca pela organização ideal.

Assim, um método para possibilitar que uma memória seja compatível com qualquer aplicação e ao mesmo tempo economize energia sempre que possível é provendo o desligamento de vias. Este desligamento pode reduzir o consumo de potência já que desativaria comparadores e não mais seria necessário acionar as linhas de bit e escrita das vias

desativadas. Se esta redução de potência for mais significativa do que o aumento de *miss* então uma economia de potência real ocorre.

O processador MCORE da Motorola (MALIK, MOYER e CERMAK, 2000) traz em conjunto com seu circuito integrado (IC, do inglês *Integrated Circuit*) uma cache unificada com associatividade de 4 vias que pode ser reconfigurada, definindo o número de vias ativos e utilizada como cache de instruções, dados ou ambas. A desativação em tempo de execução também é possível, provendo uma configuração ótima por demanda (ALBONESI, 2000).

3.2.1.3 Tamanho da linha (tamanho dos blocos)

Tradicionalmente, as linhas de memórias cache possuem tamanhos que variam entre 16 e 128 bytes para processadores modernos. Como os tamanhos das palavras que trafegam nestes processadores é de algumas dezenas de bits, cada linha contem diversas palavras. Assim, a localidade espacial pode ser explorada já que diversas palavras adjacentes são transferidas entre diferentes níveis de memória. Entretanto, quando a aplicação em execução não proporciona a exploração deste tipo de localidade ocorre grande desperdício de energia, já que uma significativa quantidade de dados será transferida sem necessidade.

Desta forma, o adequado ajuste do tamanho da linha pode acarretar em uma significativa redução no consumo de energia. Diversas abordagens são apresentadas na literatura. De forma estática (ZHANG, VAHID e NAJJAR, 2003), quando as configurações são definidas em tempo de projeto, é possível definir o tamanho das linhas de uma memória cache baseado em *profiles* das aplicações alvo (NICOLAESCU, JI, *et al.*, 2000), por exemplo. Entretanto, maior flexibilidade surge quando estas configurações ocorrem de forma dinâmica, possibilitando que uma maior gama de aplicações seja atendida de forma mais eficiente. Em tempo de compilação, é possível extrair algumas informações pertinentes para a configuração do tamanho de linhas (WITCHEL e ASANNOVIC, 2001). Ainda de forma dinâmica, e em tempo de execução (VEIDENBAUM, TANG e GUPTA, 1999) a cache pode ser adaptada de acordo com o comportamento da aplicação.

3.3 NoC – Network on Chip

Com a escalada da tecnologia em direção a sistemas cada vez mais complexos, como os MPSoCs compostos por grande quantidade de elementos de processamento, os barramentos tradicionais são incapazes de manter a demanda requerida (GUERRIER e GREINER, 2000). Além da questão desempenho, a energia consumida por barramentos

aumenta proporcionalmente com o número de EPs conectados. Assim, redes em chip ou NoC surgem como promessa de solução para estas interconexões (BENINI e DE MICHELI, 2002). Como as conexões das NoCs são ponto a ponto, estas não sofrem com os mesmos problemas dos barramentos, sanando as dificuldades com desempenho e aumento de consumo energético quando muitos elementos são conectados.

Entretanto, assim como os processadores e memórias, componentes da NoC responsáveis pela comunicação entre estes elementos tendem a ser subutilizados dependendo da aplicação em execução. Os diferentes componentes destas redes devem então ser capazes de se adequar para que o mínimo de energia seja utilizado.

A reconfiguração de forma dinâmica principalmente dos roteadores da NoC busca a manutenção do desempenho com o menor consumo energético.

3.3.1 Roteadores

A possibilidade da utilização da NoC em substituição aos barramentos trouxe a inserção de mais componentes ativos. A arquitetura das NoCs são baseadas em chaveamento, ou de pacotes ou de circuitos. Assim, estas bases levam à construção de novos e eficientes princípios na construção de roteadores. Estes roteadores podem ser simples comutadores ou abranger comportamentos ditos inteligentes dotados de várias funcionalidades.

Roteadores necessitam de diferentes recursos para capacitar a rede de forma adequada. Protocolos de roteamento, por exemplo, definem a adequada comunicação entre os roteadores. Algoritmos de roteamento determinam quais caminhos serão utilizados para que dois ou mais roteadores possam receber determinada informação. Outo item importante são os buffers que auxiliam na sincronização dos dados que trafegam pela NoC, e são essências quando o chaveamento é feito por pacote, já que a comunicação por *burst* é comum nestas redes. A largura dos barramentos que promovem a comunicação entre estes roteadores é outra questão importante, pois tem direta influência na quantidade de *buffers* ativos nos roteadores, que consequentemente provoca variação no consumo de potência.

Todas estas etapas são apresentadas na literatura tanto de forma estática, como de forma dinâmica, isto é, reconfigurados. Em termos de *buffers*, além da desativação de alguns canais quando diferentes larguras de dados são providas, a reconfiguração da sua profundidade é possível apresentando significativo aumento na eficiência energética (MATOS, CONCATTO, *et al.*, 2011) e (RAHMANI, LILJEBERG, *et al.*, 2010).

O chaveamento é outro alvo de amplo estudo, já que dependendo do tipo de aplicação este quesito torna-se prioridade. É possível que uma aplicação necessite de alta prioridade, requerendo um chaveamento por circuito, o que reduziria a latência durante a comunicação, ou quando múltiplas aplicações estão ativas e concorrem por caminhos para chegar aos seus EPs, o chaveamento por pacotes torna-se uma escolha (AHMAD e ARSLAN, 2005).

3.3.2 Interfaces de rede

As interfaces de rede ou NI (do inglês, Network Interface) são responsáveis por empacotar e desempacotar os pacotes que trafegam pela rede, quando esta está configurada para chaveamento por pacote. Diversas implementações aparecem na literatura, inclusive provendo estudos quando estas tarefas são realizadas via hardware dedicado ou software (BHOJWANI e MAHAPATRA, 2003).

Estas NIs são responsáveis por diversas tarefas indispensáveis para efetivar a comunicação entre os roteadores da NoC e os elementos de processamento, dentre elas:

- Controlar o fluxo de dados: disponibilizar corretamente os dados tanto no sentido EP→NoC quanto no sentido NoC→EP, sem perdas de informação.
- Traduzir a informação: as informações trafegadas na NoC são adicionadas de protocolos e informações que auxiliam no controle de fluxo e no direcionamento das mensagens, entretanto estas informações não são compreendidas (nem necessárias) para o EP.
- Reconstituir e dividir as informações: a largura dos dados na NoC pode ser diferente da largura de dados do EP, desta forma a NI deve ser capaz de reagrupar ou dividir estes dados de acordo com a largura do canal disponível.

Os distintos EPs que formam os sistemas heterogêneos podem utilizar diferentes protocolos de comunicação, e através destas interfaces os protocolos podem ser unificados, sendo traduzidos nas próprias NIs, por exemplo. Estas interfaces devem acompanhar de forma dinâmica as mudanças dos roteadores da NoC quando estes são reconfigurados. Principalmente no que tange à largura dos barramentos que levam ao desligamento de *buffers* e outros componentes.

3.4 Considerações

Duas abordagens são importantes quanto aos hardwares reconfiguráveis: a necessidade de fazer com que uma mesma área de silício seja capaz de executar mais de uma tarefa é uma das buscas - compondo mais de uma arquitetura ou padrão de comportamento em um mesmo espaço, como a reconfigurabilidade baseada em FPGAs que buscam implementar o hardware necessário, esbarrando na velocidade de execução do conjunto encontrar melhor hardware-escrever hardware-executar aplicação.

Outra questão é o uso de ASICs para implementar múltiplas funções com recursos estritamente necessários para a aplicação pertinente. Esta tarefa é mais árdua quando se leva em conta a rigidez dos ASICs, entretanto deve ser levado em conta que esta abordagem é capaz de prover maior desempenho, principalmente por que o hardware já existe e não precisa ser “escrito”.

4 SISTEMAS MULTIPROCESSADOS RECONFIGURÁVEIS

Sistemas Multiprocessados em Chip (MPSoCs) têm emergido como uma tecnologia capaz de prover as necessidades de crescimento de desempenho e a integração de diferentes características em um único circuito integrado. A exploração de paralelismo em diferentes níveis tem sido crucial para o sucesso destas plataformas. Para tal, os Elementos de Processamento (EPs) que compõem os MPSoCs podem ser constituídos de diferentes arquiteturas como processadores de uso geral, processadores VLIW, DSPs, processadores vetoriais e outros aceleradores, além de memórias cache locais com diferentes organizações. Para efetivar a comunicação entre estes diversos EPs, sejam processadores, memórias ou conjunto destes, redes-em-chip (NoCs) têm sido adotadas, provendo reusabilidade, escalabilidade, paralelismo e grande capacidade de vazão de dados, como apresentado na Seção 3.3.

Entretanto, a busca em espaço de projeto não visa apenas desempenho, mas objetiva também a redução do consumo de potência e energia. Assim, técnicas capazes de reduzir de forma significativa o consumo de potência são essenciais para possibilitar o uso eficiente do agressivo avanço da tecnologia.

Algumas abordagens são apresentadas na literatura, como em Atitallah (ATITALLAH, NIAR, *et al.*, 2006) que apresenta um *framework* onde é possível inserir a descrição dos componentes de hardware que compõe um MPSoC e a partir deste ponto definir a melhor relação entre desempenho e consumo energético. Entretanto, este *framework* deve ser utilizado em tempo de projeto, ou seja, não beneficiando possibilidades dinâmicas que ocorram em tempo de execução.

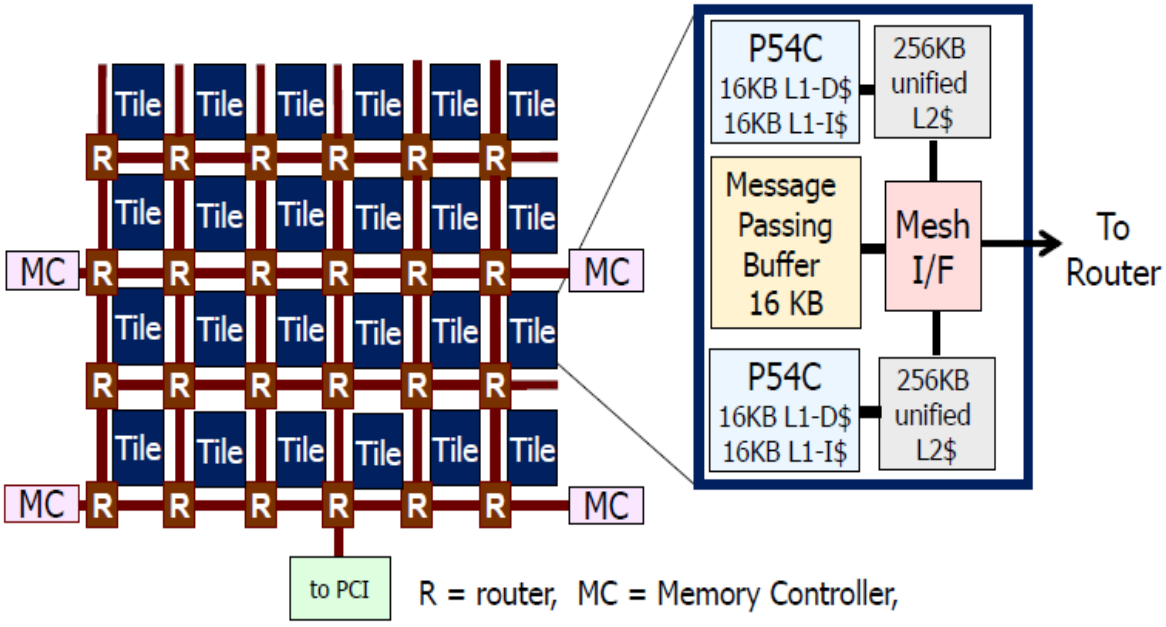
Outra abordagem pode ser vista em Mandelli (MANDELLI, OST e CARARA, 2011) onde é apresentado um mapeamento dinâmico de tarefas para MPSoCs baseados em NoC preocupando-se com o consumo energético quando deste mapeamento. Novamente, em Huang (HUANG e QIANG, 2010), uma abordagem preocupando-se com energia e confiabilidade buscando o melhor escalonamento e alocação de tarefas em MPSoCs é tomada.

Apesar da preocupação com consumo de energia em MPSoCs mostrada em vários trabalhos, poucos lidam com a questão da reconfigurabilidade em diferentes granularidades dos diferentes elementos. As técnicas mostradas no capítulo 3 são, invariavelmente, trabalhadas de forma individual impossibilitando uma análise mais adequada quando todos os componentes que compõe o sistema estão ativos.

As técnicas de redução de potência apresentadas na Seção 2 são amplamente utilizadas. Outras, como as descritas na Seção 3.1.2, mostram a capacidade de alterações na arquitetura dos processadores, neste caso processadores VLIW. A modificação da quantidade de linhas de execução em processadores VLIW altera sua arquitetura, reconfigurando o número e o tipo de unidades funcionais paralelas disponíveis, possibilitando alcançar a melhor combinação entre número de unidades funcionais ativas, desempenho e consumo energético. Outro beneficio do controle da quantidade de unidades funcionais ativas em processadores VLIW é em relação à memória de instruções. Como a largura da porta da memória de instruções está diretamente ligada à largura das instruções do processador, é possível reduzir o consumo de potência também nesses componentes, redimensionando a largura de dados.

A comunicação entre os diferentes EPs de um MPSoC e os roteadores da NoC é feita, em muitos casos, utilizando espaços em memória destes, como mostrado na Figura 4.1. Então, a variação da quantidade de unidades funcionais de um processador VLIW pode variar a vazão e a largura de dados requerida da memória de instruções, e por consequência dos roteadores da NoC. É possível perceber que estas alterações refletem diretamente no consumo de potência de todo o sistema. Como as alterações em largura de dados são causadas pela reconfiguração do processador VLIW, memórias e roteadores da NoC planejados em tempo de projeto podem acarretar em redução do desempenho (quando baseado no caso médio) ou consumo desnecessário de potência (quando baseado no pior caso).

Figura 4.1: Intel SCC - espaço em memória para comunicação com roteadores da NoC



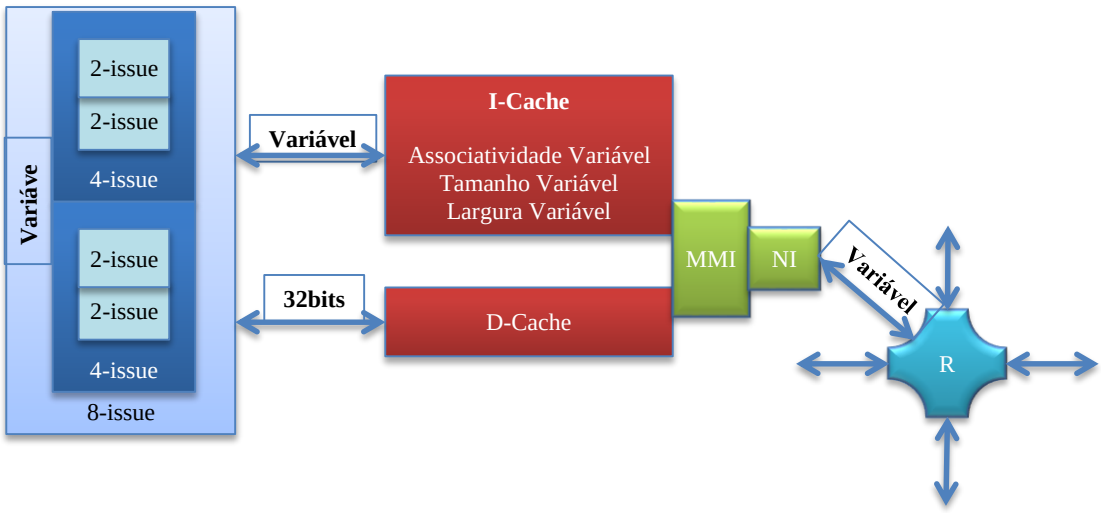
Fonte: (INTEL, 2011)

4.1 Ineficiência dos Sistemas parcialmente reconfiguráveis

Em elementos de processamento constituídos de processadores e memórias cache, se o processador apresenta técnicas possibilitando a redução do consumo de potência, a potência das memórias cache torna-se proporcionalmente mais crítica para todo o sistema. Assim, técnicas para redução do consumo de potência devem ser aplicadas também às memórias cache, acompanhando as mudanças no processador, em busca da melhor relação desempenho/potência. Por exemplo, se um processador tem sua frequência de operação reduzida de 500MHz para 200MHz, a memória cache pode ter também sua frequência reduzida, desde que não apresente redução de desempenho, acentuando a economia energética. Esta seção apresenta um estudo mostrando consumos de potência do processador VLIW, memória cache de instruções e roteador da NoC em diferentes casos, objetivando mostrar que tais variações são significativas.

A figura 4.2 representa o hardware confeccionado para este estudo. Este nó é totalmente funcional (o mesmo será explanado na seção 5.1), e foi elaborado utilizando a linguagem de descrição de hardware VHDL. Após ter seu funcionamento verificado, foram extraídos dados de potência desta descrição para diversas configurações. Assim, tomando como referência o nó da figura 4.2 o seguinte experimento foi feito: é suposto que existe uma necessidade de tráfego de 19.2Gbps sobre o processador, assim são analisados todos os elementos que compõe o nó referência.

Figura 4.2: Elemento de Processamento conectado ao roteador



Fonte: Autor

As configurações utilizadas para estes testes são apresentadas na Tabela 4.1, onde são também informados os valores iniciais do nó correspondente. As potências do processador, interfaces e roteador da NoC foram extraídas utilizando a ferramenta Cadence RTL Compiler para um nó tecnológico de 65nm. Para a memória de instruções, a ferramenta CACTI 6.0 (CACTI) foi utilizada para a mesma tecnologia (SANTOS, NAZAR, *et al.*, 2013).

O nó referência é inicializado com valores de configuração iniciais, sendo que o processador VLIW configurado com 8 linhas de execução e frequência de operação de 500MHz, memória cache de instruções na com 8 vias de associatividade 256bits de largura de porta e 32KBytes de tamanho operando também a 500 MHz. O roteador da NoC com largura de dados de 256bits e 500MHz de frequência.

Num primeiro momento, apenas o processador possui a capacidade de reconfiguração, neste caso, reduzir a quantidade de linhas ativas, consequentemente unidades funcionais e a largura de instruções. Além do número de linhas, este processador teve também a frequência de operação variada. Quando estas alterações são feitas no processador tem-se a redução da potência, entretanto apenas no componente alterado, já que o restante do circuito continua estático e ativo.

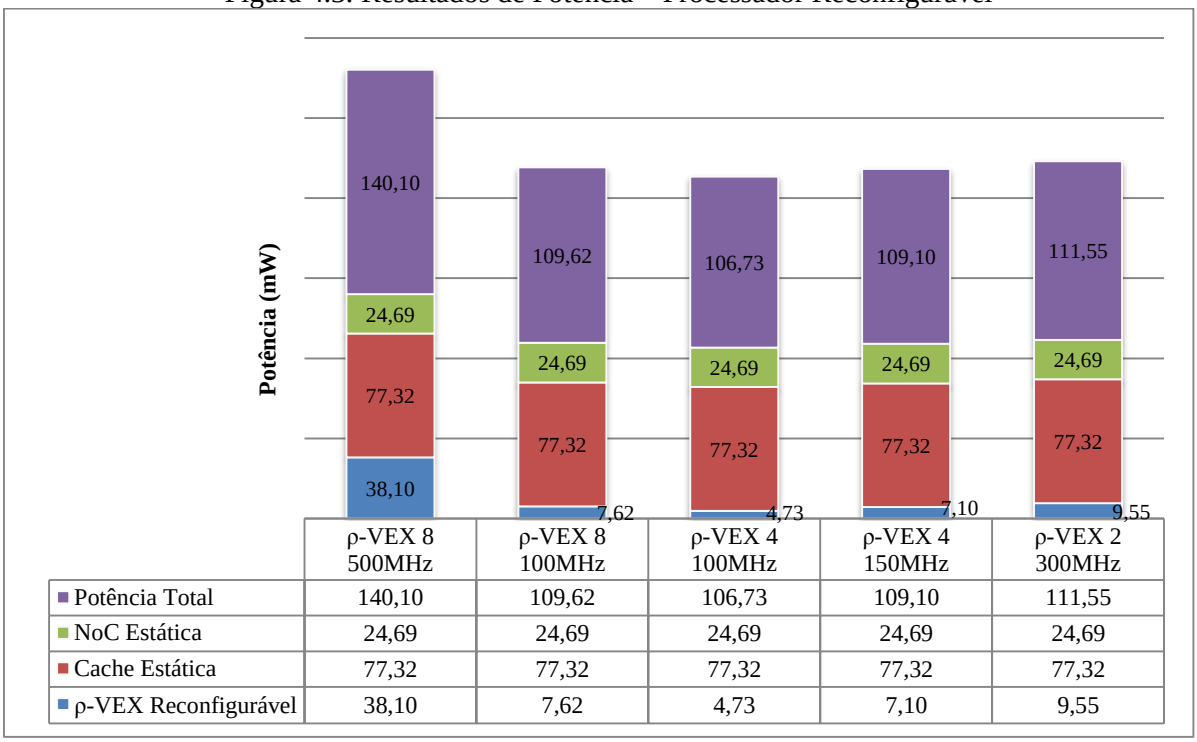
A Figura 4.3 apresenta valores de potência para o nó referência, quando apenas o processador é reconfigurado. A figura ilustra apenas potências para memória cache de instruções, sendo que a memória de dados permanecerá estática e não é alvo deste estudo.

Tabela 4.1: Parâmetros utilizados no experimento

Parâmetros	Valor inicial	Variação
ρ-VEX – Número de linhas ativas	8	2/4/8
ρ-VEX – Frequência (MHz)	500	100 a 500 (passo 50)
Cache – Largura da porta (bits)	256	64/128/256
Cache – Frequência (MHz)	500	100 a 500 (passo 50)
Cache – Tamanho (Kbytes)	32	16/32
Cache – Associatividade (Ways)	8	1/2/4/8
NoC – Frequência (MHz)	500	100 a 500 (passo 50)
NoC – Largura de Dados (bits)	256	16/32/64/128/256
Potência - Valores Iniciais (mW)	140,10	---

Fonte: Autor

Figura 4.3: Resultados de Potência – Processador Reconfigurável



	ρ-VEX 8 500MHz	ρ-VEX 8 100MHz	ρ-VEX 4 100MHz	ρ-VEX 4 150MHz	ρ-VEX 2 300MHz
Potência Total	140,10	109,62	106,73	109,10	111,55
NoC Estática	24,69	24,69	24,69	24,69	24,69
Cache Estática	77,32	77,32	77,32	77,32	77,32
ρ-VEX Reconfigurável	38,10	7,62	4,73	7,10	9,55

Fonte: (SANTOS, NAZAR, *et al.*, 2013)

Inicialmente a capacidade de vazão de dados é de 128Gbps (primeira coluna da Figura 4.3), isto é, maior do que os 19,2Gbps requeridos. Além disto, com a configuração inicial tem-se o consumo de potência de 140,10mW para o nó referência. Buscando atender as requisições de vazão de dados, o processador tem sua frequência de operação reduzida para 100MHz mantendo a configuração com 8 linhas de execução, alcançando a vazão de dados de 25,6Gbps e o consumo de potência total do sistema de 109,62mW, ou seja, houve uma redução de 80% na capacidade de processamento (vazão de instruções) e uma redução de apenas 22% da potência total do nó. No passo seguinte, ainda analisando o gráfico da figura 4.3, o número de linhas de execução disponíveis no processador são reduzidas de 8 para 4 e a frequência de operação é mantida em 100MHz.

A capacidade do processador cai então para 12,8Gbps, inferior à requerida de 19,2Gbps. Desta forma pode-se aumentar a frequência do processador para 150MHz, atingindo a meta de 19,2Gbps e levando a potência para o valor de 109,mW. Apesar da redução de potência de 38,1mW para 7,1mW por parte do processador, a redução total de potência do sistema é de aproximadamente 23%, como pode ser calculado a partir do gráfico apresentado na Figura 4.3.

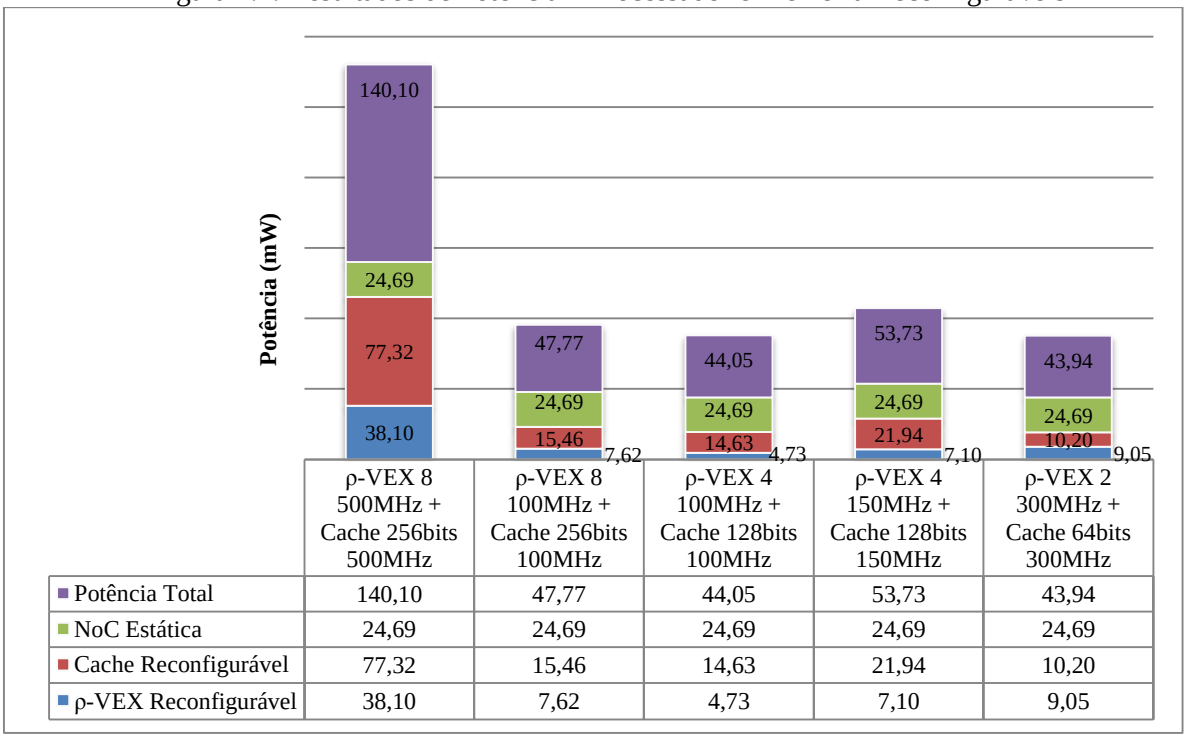
Desta forma, outros componentes de hardware precisam ser atingidos pelo processo de redução de potência. Quando o processador VLIW tem seu número de linhas de execução

variada, a largura necessária da porta da memória de instruções também pode ser alterada. Assim, quando o processador é modificado e passa a utilizar instruções com largura de 128bits, onde anteriormente era de 256bits, a memória cache de instruções deve acompanhar tal alteração, assim como alterar sua frequência de operação para que seja capaz de manter a melhor relação desempenho/potência possível.

Como apresentado na Seção 3.2, as memórias caches podem ter diversos parâmetros reconfigurados, como tamanho de blocos e associatividade, reduzindo o consumo de potência e/ou incrementando o desempenho do elemento de processamento (GIVARGIS e VAHID, 2002) enquanto acompanha as reconfigurações aplicadas ao processador, conforme discutido na Seção 3.1. A Figura 4.4 mostra resultados de potência adicionando memórias caches de instruções ao grupo de hardwares reconfiguráveis.

Nota-se que processador e memória reconfigurados conseguem reduzir a potência do nó, que inicialmente é de 140,1mW para 43,94mW quando o processador VLIW com 2 linhas de execução rodando a 300MHz é utilizado. Apesar do processador não ter a menor potência para a vazão requerida, a redução da largura de dados para 64bits possibilita o uso de uma memória cache de instruções com menor consumo de potência, promovendo uma redução de potência ainda maior no consumo total.

Figura 4.4: Resultados de Potência – Processador e Memória Reconfiguráveis



	ρ-VEX 8 500MHz + Cache 256bits 500MHz	ρ-VEX 8 100MHz + Cache 256bits 100MHz	ρ-VEX 4 100MHz + Cache 128bits 100MHz	ρ-VEX 4 150MHz + Cache 128bits 150MHz	ρ-VEX 2 300MHz + Cache 64bits 300MHz
Potência Total	140,10	47,77	44,05	53,73	43,94
NoC Estática	24,69	24,69	24,69	24,69	24,69
Cache Reconfigurável	77,32	15,46	14,63	21,94	10,20
ρ-VEX Reconfigurável	38,10	7,62	4,73	7,10	9,05

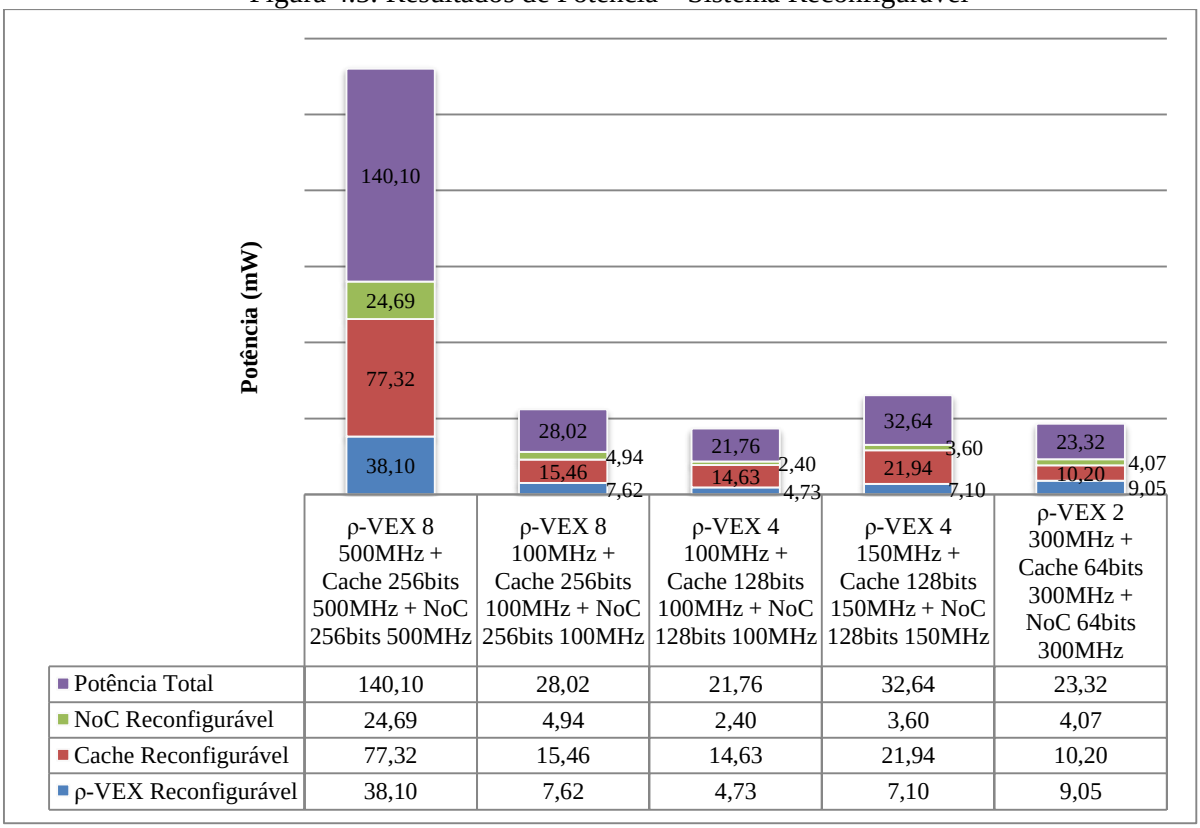
Fonte: (SANTOS, NAZAR, *et al.*, 2013)

4.2 Sistemas Totalmente reconfiguráveis

Embora a reconfiguração de alguns parâmetros de processador e memória sejam capazes de reduzir o consumo de potência destes elementos, chegando a algumas ordens de grandeza (como apresentado na Seção 4.1), com o crescimento dos sistemas e o uso de soluções de comunicação *intrachip* mais complexas, como as NoCs, estes também devem ser considerados. Os roteadores de uma NoC, assim como os barramento que os interligam, normalmente trabalham com frequências e largura de dados fixos, entretanto, após a reconfiguração de processadores e memórias as NoCs podem se tornar ineficientes, consumindo potência desnecessária ou tornando-se um gargalo para o sistema.

Tomando como exemplo um roteador que tem sua frequência de operação de 400MHz e largura de dados de 128bits interconectado à uma memória cuja largura de dados é de 256bits operando à 200MHz, estes estão funcionando com máxima eficiência. Entretanto, se a memória for reconfigurada e passar a operar com frequência de 100MHz e 128bits de largura de dados, o roteador da NoC passa a ter um consumo desnecessário. Por outro lado, se a memória for reconfigurada para 400MHz e 256bits, o roteador passa a ser um fator limitante de desempenho.

Figura 4.5: Resultados de Potência – Sistema Reconfigurável



	ρ-VEX 8 500MHz + Cache 256bits 500MHz + NoC 256bits 500MHz	ρ-VEX 8 100MHz + Cache 256bits 100MHz + NoC 256bits 100MHz	ρ-VEX 4 100MHz + Cache 128bits 100MHz + NoC 128bits 100MHz	ρ-VEX 4 150MHz + Cache 128bits 150MHz + NoC 128bits 150MHz	ρ-VEX 2 300MHz + Cache 64bits 300MHz + NoC 64bits 300MHz
Potência Total	140,10	28,02	21,76	32,64	23,32
NoC Reconfigurável	24,69	4,94	2,40	3,60	4,07
Cache Reconfigurável	77,32	15,46	14,63	21,94	10,20
ρ-VEX Reconfigurável	38,10	7,62	4,73	7,10	9,05

Fonte: (SANTOS, NAZAR, *et al.*, 2013)

A falta de combinação entre os diferentes elementos que compõe os MPSoCs tende a limitar a capacidade de reduzir de forma mais agressiva o consumo de potência destes sistemas. Assim, a reconfiguração simultânea dos três eixos, processamento, memória e comunicação passam a ser crucial.

A Figura 4.5 ilustra o consumo de potência quando todo o sistema é reconfigurado. É possível notar que quando a reconfiguração ocorre em todos os eixos, a redução de potência é ainda mais significativa. Assim, seguindo o exemplo da Seção 4.1, quando o objetivo é manter uma vazão de dados no processador em 19.2Gbps, com a reconfiguração apresentada em todos os eixos a redução de potência cai de 140,1mW para 23,32mW, alcançando a melhor relação desempenho/potência.

4.3 Conclusão

A análise de consumo de potência deve ser considerada em todos os eixos que compõem os sistemas. No capítulo 4, foi apresentado um breve estudo mostrando que quando um ou outro componente é tratado de forma que consuma apenas a potência necessária para determinada tarefa, apesar da redução de potência alcançada, mantém-se o desperdício em outros componentes do sistema. Assim, é imprescindível que, para atingir a melhor relação potência/desempenho, o tratamento destes componentes deve ser realizado de forma individual, buscando um grão mais fino quando destes tratamentos.

5 HETEROGENEIDADE E RECONFIGURABILIDADE

O número de aplicações que se deseja executar de forma simultânea cresce à medida que aumenta a variedade destas aplicações e amplia-se a diversidade de uso para um mesmo dispositivo. Assim, por exemplo, é comum que através de um equipamento móvel estejamos conectados a redes sociais onde câmeras e microfones estejam em constante conexão via redes sem fio enquanto fazemos uso de aplicativos gráficos que demandam grande quantidade de processamento através de decodificações e outros. Entretanto, apesar da necessidade de suportar tamanho processamento, não é necessário que todo o hardware esteja ativo o tempo todo. Em verdade não é mais possível que todo o hardware esteja ativo em tempo integral para futuras tecnologias, em virtude da previsão de alta densidade de transistores e elevado grau de integração.

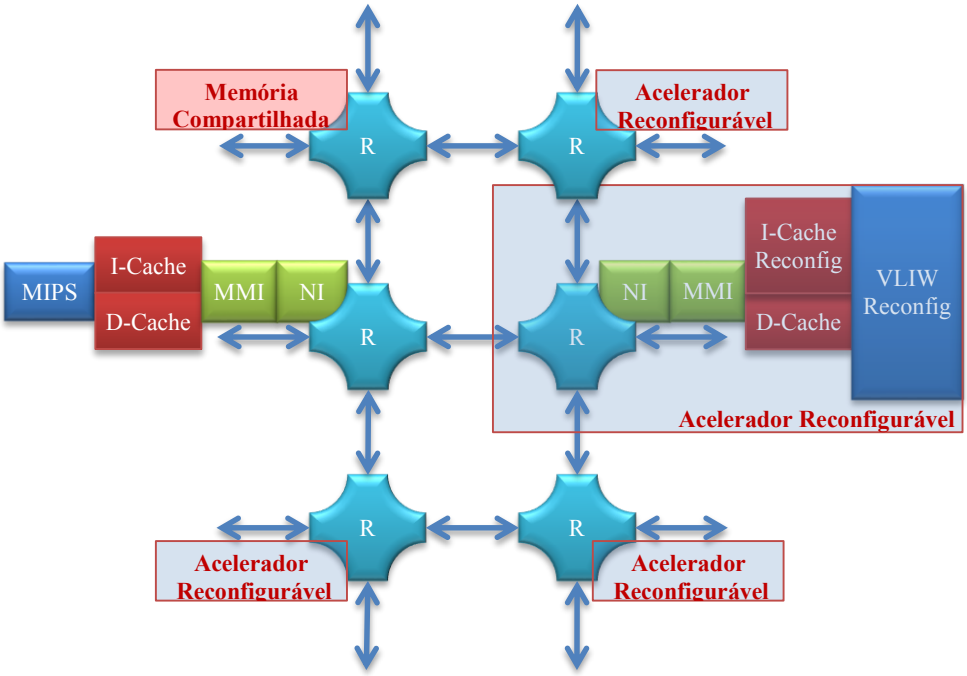
Muitas soluções desativam elementos inteiros, como a desativação de um acelerador gráfico quando este não for necessário, reduzindo assim o consumo de potência. Apesar de esta solução trazer algum benefício, uma grande porção do silício acaba sendo mantida inoperante, quando parte deste poderia compor solução para execução de outras tarefas ou aplicações.

Com o objetivo de manter uma relação ótima entre desempenho e potência este trabalho propõe o uso de um EP onde todos os elementos que o compõe são amplamente reconfigurados (e não simplesmente desativados), adaptando-se às requisições de determinada aplicação. Entretanto, para alcançar este objetivo, esta adaptação deve ser feita em todos os eixos: processamento, memória e comunicação.

Além da reconfigurabilidade nos três eixos, a heterogeneidade torna-se importante não apenas visando a redução no consumo de potência, mas também sendo capaz de proporcionar a compatibilidade com binários já estabelecidos. Em verdade a própria reconfigurabilidade promove heterogeneidade, já que pode induzir a modificação de parâmetros como suporte a diferentes instruções, largura de instruções, disponibilidade de UFs e volume de dados.

A Figura 5.1 ilustra o MPSoC com foco no processador VLIW reconfigurável, nas memórias cache de instruções adaptáveis e nos roteadores da NoC, assim como seus barramentos, adaptáveis à demanda. As interfaces entre memórias cache e roteadores da NoC, MMI e NI, possibilitam a comunicação entre estes elementos independentemente da frequência de operação de ambos, capacitando o sistema trabalhar de forma não sincronizada, isto é, memórias cache em uma determinada frequência e os roteadores da NoC em outra. Esta capacidade amplia as combinações de funcionamento.

Figura 5.1: MPSoC apresentando os EPs



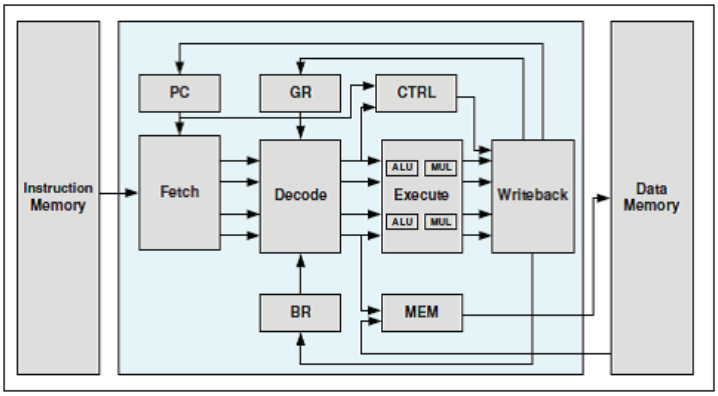
Fonte: Autor

5.1 Processador VLIW ρ-VEX Reconfigurável

O processador VLIW utilizado é baseado no processador *softcore* ρ-VEX descrito em linguagem VHDL que por sua vez, foi desenvolvido por Wong (WONG, VAN AS e BROWN, 2008) e baseia-se na arquitetura VEX (*VLIW Example*) que implementa a ISA da família Lx (FARABOSCHI, BROWN, *et al.*, 2000) de processadores VLIW embarcados promovidos pela parceria entre as empresas HP e STmicro. Apesar do desenho original do processador VEX suportar agrupamentos (*clusters*) de diversas unidades de processamento, cada qual com distinto número de unidades funcionais, apenas a arquitetura de uma única unidade de processamento é levada em consideração no processador ρ-VEX.

O processador ρ-VEX padrão, assim como cada unidade de processamento VEX padrão, é formada por 4 unidades de ULAs, 2 unidades de Multiplicadores, 1 unidade de acesso à memória e 1 unidade de *branch*, distribuídas em 4 linhas de execução (ou *issues*) Além disto, possui 64 registradores de 32bits de uso geral e 8 registradores de 1bit para *branch*. Cada instrução é formada por múltiplas subinstruções ou sílabas, sendo cada sílaba podendo ser vista como uma simples instrução RISC. A Figura 5.2 ilustra a organização do processador padrão 4-sílabas.

Figura 5.2: Diagrama do processador ρ-VEX



Fonte: (WONG, VAN AS e BROWN, 2008)

O processador ρ-VEX pode ser configurado em tempo de síntese de forma a constituir um processador capaz de executar 2 sílabas VEX paralelas (2-*issue*), 4 sílabas (4-*issue*) ou ainda executar 8 sílabas em paralelo (8-*issue*). A Tabela 5.1 ilustra a distribuição das unidades funcionais no processador para cada uma das suas três variações disponíveis.

A modificação feita para este trabalho possibilita que estas configurações sejam feitas em tempo de execução, isto é, sem a necessidade de refazer a síntese para o processador, seja esta destinada para ASIC ou FPGA. Nesta versão é possível ativar e desativar unidades funcionais, formando um processador capaz de executar 2, 4 ou 8 sílabas por instrução VLIW, conforme a Tabela 5.1. Deve ser considerado que a reconfiguração deste processador é realizada utilizando as técnicas de *clock gating* e *power gating*, abrangendo os problemas de potência dinâmica e potência estática, respectivamente. Para tanto, alguns sinais de controle foram adicionados, através dos quais se determina o número de linhas de execução (ou issue) que permanecem ativos. Sendo assim, para fins de cálculo de potência e energia, é considerado que quando o processador é reconfigurado, as unidades desativadas consomem potências próximas de zero.

Diferentemente da proposta apresentada por Anjam (ANJAM, NADEEM e WONG, 2011), onde o mesmo processador é reconfigurado baseado em uma versão 2-issue destinando-se a implementações em FPGA, a proposta neste trabalho visa a implementação em ASIC (*Application Specific Integrated Circuit*). Desta forma, a base é o processador com maior número de recursos, e a partir deste componente, seções são desativadas efetivando a configuração desejada.

Tabela 5.1: Organização do processador ρ-VEX em suas diferentes configurações

2-issue	4-issue	8-issue
ULA/CTRL/MUL	ULA/CTRL	ULA/CTRL
	ULA/MUL	ULA/MUL
		ULA/MUL
		ULA
		ULA
		ULA/MUL
	ULA/MUL	ULA/MUL
ULA/MEM/MUL	ULA/MEM	ULA/MEM

Fonte: Adaptado de (WONG, VAN AS e BROWN, 2008)

5.2 Memória Cache de Instruções Reconfigurável

As variações promovidas pela reconfiguração do processador ρ-VEX afetam diretamente a largura da memória de instruções, já que alterando a quantidade de linhas em processadores VLIW, altera-se também a largura de suas instruções. De certa forma, independentemente da natureza do processador que compõe o elemento de processamento, seja ele Superescalar, VLIW, RISC, etc, capacidades de reconfiguração causarão variações nos requerimentos quanto à vazão de instruções e dados. Além da simples alteração da quantidade de sílabas processadas por instrução proveniente da reconfiguração citada, a reorganização ou disponibilidade de recursos (UFs) também contribui para que as memórias cache sejam exigidas de diferentes formas. Assim, a variação destes requerimentos promove modificações na escolha da melhor configuração das memórias.

As memórias caches de instruções utilizadas neste trabalho podem ser ajustadas visando a melhor relação entre desempenho e potência, e foram baseadas nos trabalhos de Benitez (BENITEZ, MOURE, *et al.*, 2010) e Zhang (ZHANG, VAHID e NAJJAR, 2003). A meta é manter a configuração ideal de memória cache para determinada aplicação, variando tamanho total de memória, largura de dados e associatividade, mantendo o menor consumo de potência sem danos ao desempenho. Adicionalmente, a frequência de operação também varia, acompanhando a adaptação feita no processador, compondo um EP completamente reconfigurável capaz de constituir maior desempenho com menor consumo de potência possível.

Memórias de dados e memórias de instruções podem ser ajustadas de forma independente, mas a frequência de operação para ambas as memórias é a mesma, já que estas

se comunicam com o roteador através de uma mesma interface (MMI). Entretanto, vale salientar que este trabalho abrange apenas memórias cache de instruções, pois apresentam número significativo de variações de comportamento quando das alterações de processador já citadas.

A métrica é manter o menor número de *misses* possíveis na memória de instruções para determinada aplicação. Apesar de esta memória ter como base o trabalho de Benitez (BENITEZ, MOURE, *et al.*, 2010), suas configurações são feitas remotamente no início da execução de cada diferente aplicação, não possibilitando a auto-reconfiguração apresentada no mesmo trabalho.

Para este estudo, uma descrição comportamental deste hardware foi confeccionada utilizando a linguagem VHDL. Esta memória é constituída de grupos de memórias que são ativados ou desativados conforme a necessidade. Além da quantidade de memória disponível, estes blocos de memórias podem ser interligados de forma a alterar sua associatividade e largura de dados.

A reconfiguração das memórias cache ocorre de forma individual e apenas na memória de instruções, sendo que os parâmetros ótimos para a memória cache de instruções não necessariamente serão os mesmos para a memória cache de dados. O único parâmetro mantido fixo é o tamanho de cada linha, contendo blocos com duas vezes a largura da palavra de instruções. Então, quando o processador tiver o tamanho da palavra de instruções configurado para 64bits, ou seja, processador com duas linhas de execução, a memória cache de instruções terá 16Bytes de linha, quando a instrução tiver 256bits o tamanho da linha será de 64Bytes. A Tabela 5.2 apresenta as configurações possíveis para as memórias apresentadas.

A memória cache de dados para este trabalho é apresentada como uma RAM já contendo os dados pertinentes, isto é, os dados das aplicações são previamente carregados nesta memória, e possuem o tamanho adequado para que não acarretem *misses*. Dentre os motivos para não considerar estas caches no estudo apresentado, está o fato de que os dados destas memórias não tem comportamento variável no decorrer da execução da aplicação, assim como não variam de tamanho ou comportamento quando o número de linhas é modificado.

Tabela 5.2: Configurações possíveis para a memória cache de instruções.

	Cache de Instruções
Tamanho (KBytes)	4/8/16/32
Associatividade	Direto/2/4/8
Linha (Bytes)	2 * largura das instruções
Largura (bits)	64 / 128 / 256

Fonte: Autor

5.3 NoC Reconfigurável

A rede-em-chip que compõe o MPSoC em estudo é a NoC SoCIN (ZEFERINO, 2003) e utiliza os roteadores RaSoC (do inglês, *Router Architecture for System-on-Chip*) (ZEFERINO, KREUTZ e SUSIN, 2004).

O roteamento desta NoC utiliza o algoritmo XY determinístico e sua unidade básica de transmissão é chamada de *flit* (do inglês, *flow control* digit) adotando chaveamento por pacotes. O primeiro *flit,* chamado *header* ou cabeçalho, possui dados responsáveis pelo roteamento e informa o início de um pacote. O *flit* chamado *payload* ou corpo do pacote transporta apenas os dados propriamente. O último é chamado *tail* ou *flit* de término, contendo informações relacionadas ao término do pacote em transmissão.

Os roteadores RaSoC possuem *buffers* em cada uma das cinco portas de entrada. Para fins de redução no consumo de energia, a NoC transitará diferentes larguras de dados, sendo assim, foram necessárias modificações nos roteadores: substituiu-se os *buffers* com largura de dados configurável em tempo de síntese, por 16 conjuntos de *buffers* de 16bits para cada porta de entrada. Estas portas podem então ser configuradas para 16, 32, 64, 128 ou 256bits de largura, em tempo de execução, mantendo ativo apenas os *buffers* pertinentes para aquela largura de canal. A ideia é capacitar o roteador a consumir menos potência quando a largura ativa do barramento variar seja entre roteadores ou entre roteadores e EPs.

Para que estas modificações fossem possíveis, dois bits do *flit* de cabeçalho são utilizados como *bits* de controle, acionando o número de *buffers* necessários de acordo com a configuração requerida.

5.4 NI e MMI

Os componentes responsáveis pela comunicação entre os elementos de processamento e os roteadores da NoC são as interfaces de rede e as interfaces de gerenciamento de memória. As interfaces de rede ou NIs (do inglês, *Network Interfaces*) são responsáveis pelo empacotamento e desempacotamento dos pacotes transmitidos. Para tal, devem ser capazes de identificar os diferentes *flits* que trafegam pela rede.

A NI utilizada neste trabalho foi adaptada de (MATOS, 2010). Originalmente a NI utilizada lida com três tipos de *flits:* o cabeçalho que indica o início de um novo pacote e possui informações sobre o roteamento, carga contendo dados propriamente ditos e o *flit* de término de pacote. Assim, um barramento de *2+nbits* é utilizado, onde *n* é a largura do canal de transmissão e 2 bits contemplam a identificação dos *flits*.

Entretanto, algumas modificações foram necessárias para possibilitar este trabalho. A primeira modificação necessária foi quanto à variação da largura de dados, capacitando a NI seguir as variações de largura de dados nos roteadores. A importância da reconfiguração da NI dá-se principalmente quanto ao número de *buffers* ativos, assim reduzindo o consumo de potência. A segunda modificação foi a inserção de um terceiro bit de identificação, fazendo com que os barramentos da NoC passassem de 2+*nbits* para 3+*nbits.* Sendo assim, passam a existir cinco tipos de *flits.* A Tabela 5.3 apresenta os tipos de *flits* e seus conteúdos.

O tratamento dos *flits* de Cabeçalho, Carga-Dados e Término é feito normalmente pela NI, isto é, informações de roteamento, montagem dos pacotes em dados e acionamentos de *flags* de término. Entretanto, outra interface chamada MMI recebe diretamente os *flits* Configuração e Carga-Instruções acompanhados dos *bits* de identificação. Isto é necessário, pois dependendo do conteúdo do campo Configuração é que estes dados serão montados a partir dos pacotes transmitidos.

A interface de gerenciamento de memória ou MMI (do inglês, *Memory Management Interface*) identifica o tipo de dado proveniente dos *flits* Carga-Dados e Carga-Instruções e escreve estes dados nas memórias correspondentes. O *flit* Configuração é interpretado e a partir daí o processador e a memória cache de instruções são configurados de acordo.

Tabela 5.3: Tipos de *flits* interpretados pela NI e MMI

	Tamanho mínimo do *flit* (*bits*)										
Tipo	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Cabeçalho	0	0	1	Configuração do roteador		Informações de roteamento					
Configuração	1	1	1	Configuração do processador		Configuração da memória cache de instruções					
Carga – Dados	0	1	1	Dados							
Carga – Instruções	1	0	0	Instruções							
Término	0	1	0	Dados ou Instruções							

Fonte: Autor

5.5 Considerações

A interface MMI é capaz de identificar se o pacote contém dados ou instruções, entretanto, neste trabalho apenas pacotes contendo instruções são trafegados fazendo com que a MMI não utilize este recurso.

A reconfiguração de todos os componentes de um sistema é possível e apresenta benefícios claros, como apresentado no capítulo 4. Apesar da necessidade de aumento de hardware, principalmente no que tange ao controle, a economia de energia alcançada supera a inclusão do pequeno número de *bits* de controle e multiplexadores. Obviamente existe um aumento de área, o que não é foco deste trabalho.

6 ORGANIZAÇÃO E VALIDAÇÃO DO EXPERIMENTO

Este capítulo apresenta os resultados extraídos dos experimentos quando a reconfigurabilidade é aplicada. O experimento é baseado na execução de aplicações reais no hardware apresentado na Figura 5.1. Este hardware é capaz de ter seus elementos configurados conforme apresentado no Capítulo 5.

Para ilustrar a importância da reconfigurabilidade completa dos sistemas multiprocessados, aplicações com diferentes exigências de hardware, como diferente ILP e diferentes comportamentos quanto à memória cache de instruções foram escolhidas.

As aplicações foram compiladas para as três variações do processador VLIW ρ-VEX apresentado na Seção 5.1, conforme as unidades funcionais dispostas na Tabela 5.1. A partir deste ponto é feito um cruzamento entre as várias configurações possíveis de processador, memória cache de instruções e roteadores da NoC, possibilitando a extração da melhor relação entre desempenho e energia consumida para todo o sistema.

A Tabela 6.1 apresenta as aplicações executadas, além de informar o tamanho de cada memória de instruções geradas para as diferentes organizações do processador VLIW. O tamanho da memória do programa varia dependendo dos parâmetros escolhidos para a compilação e do número de unidades funcionais disponíveis pelo processador. Esta variação influencia diretamente no desempenho e na energia consumida pelo elemento de processamento, assim como na escolha das melhores configurações de hardware para cada aplicação.

Tabela 6.1: Aplicações experimentadas no hardware reconfigurável

	Tamanho da memória de instruções (KB)		
Aplicação	**ρ-VEX 2-linhas**	**ρ-VEX 4-linhas**	**ρ-VEX 8-linhas**
ADPCM encode	34,7	61,4	113,6
StringSearch	29,5	50,2	94,1
OddEven	2,3	4,1	7,9
Multiplicação de Matrizes	7,2	7,4	13,6
CRC	10,7	15,3	25,6
FIR	4,7	9,4	19,4
x264 decode	9,4	11,2	21,9

Fonte: Autor

Seguindo a sistemática apresentada no Capítulo 4 e considerando que todos os componentes descritos no capítulo 4 foram escritos em linguagem de descrição de hardware VHDL, um sistema padrão foi elaborado e a partir deste sistema as reconfigurações foram

aplicadas a cada componente – processamento, memória e comunicação – buscando uma configuração ótima para cada aplicação. Para todas as aplicações, o sistema padrão é o sistema inicial e conta com o máximo de recursos disponível, isto é: o processador VLIW conta com 8 linhas de execução, memória cache de instruções com 32KBytes de tamanho, 8 vias de associatividade e 256bits de largura de dados e roteadores da NoC com 256bits.

Cada execução é iniciada a partir da memória compartilhada, podendo esta ser considerada uma memória RAM ou uma memória cache de nível mais alto contendo todo o programa, e são considerados tempos de latência para *Miss*. Além disto, o compilador gera tamanhos de programas diferentes de acordo com a configuração do processador, fazendo com que o tamanho da memória cache de instruções assim como outros parâmetros tenham adequada importância.

Todas as aplicações foram compiladas com a utilização das ferramentas promovidas pela HP/ST (HP LABS, 2010) e foram utilizados parâmetros de compilação que fornecessem o maior desempenho (-O4 -H4). Os dados de potência do processador VLIW, das interfaces MMI e NI e dos roteadores da NoC foram estimadas com auxílio da ferramenta *CADENCE RTL Compiler* com taxa de chaveamento de 50%. Quanto à memória cache de instruções, a ferramenta CACTI 6.5 (CACTI) foi utilizada. Todos os resultados foram extraídos para tecnológica de 65nm. Os experimentos aqui realizados foram apoiados na ferramenta Menthor ModelSim onde o sistema construído foi simulado e avaliado.

Para fins de cálculo de energia, as potências do roteador da NoC e das interfaces MMI e NI são apresentadas somadas, já que em termos energéticos as interfaces apresentam pequena adição frente aos roteadores.

As Seções seguintes apresentam resultados para as aplicações da Tabela 6.1 quando executadas nas diferentes configurações do sistema. A principal métrica utilizada é o produto da energia pelo tempo de execução ou EDP (do inglês, *Energy Delay Product*), que representa a energia gasta durante a execução de uma determinada aplicação. O tempo de execução também é apresentado visando compreender o comportamento do desempenho frente à redução de consumo energético quando a reconfiguração é aplicada na execução de cada aplicação.

As potências estimadas com as ferramentas mencionadas são utilizadas para o cálculo da energia total consumida através da equação $E = P_T*(1/f)$, onde P_T é a potência dinâmica somada a potência estática e f a frequência de operação deste circuito. A métrica EDP é então

calculada a partir da energia *E* calculada e multiplicada pelo número de ciclos *#ciclos* que determinada aplicação levou para ser executada, tal qual a equação *EDP = E * #cliclos*, conforme (SENGUPTA e SALEH, 2005).

6.1 ADPCM, StringSearch e OddEven

Inicialmente a aplicação ADPCM é executada na configuração padrão do sistema apresentada na Tabela 6.2, e sequencialmente são aplicadas as reconfigurações conforme a mesma tabela.

Primeiramente o processador VLIW com 8 linhas de execução é utilizado com diferentes configurações de memória cache de instruções e NoC, a partir da configuração padrão. A Figura 6.1 apresenta os resultados para a aplicação ADPCM, onde é possível observar que o processador com 8 linhas tem a memória cache de instruções configurada buscando uma redução de EDP com o menor dano possível ao desempenho. Como a alteração do tamanho da memória cache de 32KB para 16KB piora EDP e desempenho quando aplicadas as configurações 1 e 2, a próxima alteração ocorre no processador.

Tabela 6.2: Configurações de Hardware para ADPCM

	Processador	**Cache de Instruções**				**Roteador NoC**
		Tamanho	**Vias**	**Linha**	**Dados**	
Configuração Padrão	ρ-VEX 8i	32KB	8	64B	256b	256b
Configuração 1	ρ-VEX 8i	16KB	8	64B	256b	256b
Configuração 2	ρ-VEX 8i	16KB	1	64B	256b	256b
Configuração 3	ρ-VEX 4i	32KB	8	64B	256b	256b
Configuração 4	ρ-VEX 4i	32KB	8	32B	128b	128b
Configuração 5	ρ-VEX 4i	16KB	4	32B	128b	128b
Configuração 6	ρ-VEX 4i	8KB	1	32B	128b	128b
Configuração 7	ρ-VEX 2i	8KB	1	32B	128b	128b
Configuração 8	ρ-VEX 2i	8KB	1	16B	64b	64b
Configuração 9	ρ-VEX 2i	16KB	2	16B	64b	64b
Configuração 10	ρ-VEX 2i	32KB	2	16B	64b	64b

Fonte: Autor

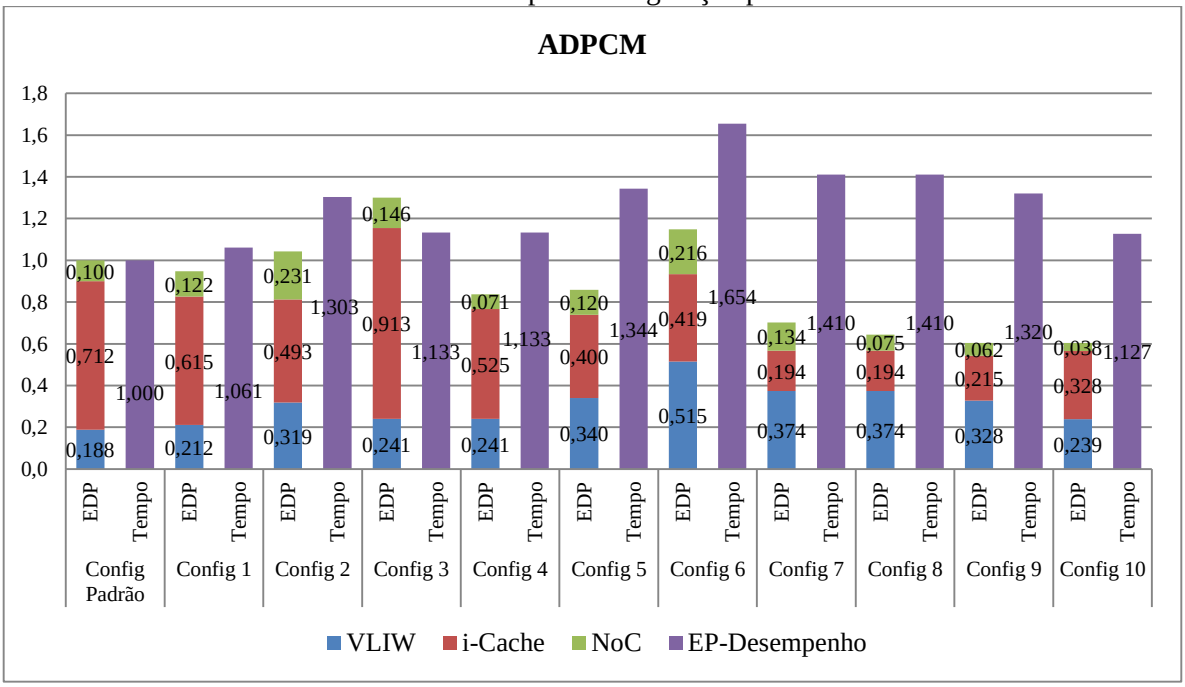
Na configuração 3, apenas o processador é alterado para 4 linhas, mantendo a memória de instruções e a NoC nos termos da configuração padrão, isto é, cache com 32KB de tamanho e 256bits de largura de dados, assim como a NoC com 256bits de largura de dados. Observando a Figura 6.1 pode-se notar que nesta configuração o EDP de todos os componentes cresce quando comparado à configuração padrão, chegando a um acréscimo de 30% com uma penalização de 13% ao desempenho. Entretanto, na configuração 4, quando

apenas a largura de dados dos componentes de memória e comunicação são alterados para melhor casar com o processador de 4 linhas, ou seja, 128bits de largura, o EDP é reduzido em 16% mantendo a penalização em 13% no tempo de execução. A redução na largura dos roteadores da NoC abrangem, além do barramento, a redução da quantidade de *buffers* existentes nos roteadores e interfaces, reduzindo a potência consumida de forma desnecessária.

As configurações 5 e 6 mantém o processador com 4 linhas de execução alterando apenas tamanho da memória cache de instruções. Estas configurações pioram tanto a métrica EDP quanto o tempo de execução da aplicação, sugerindo uma nova alteração na configuração do processador.

As configurações 7 em diante contam com o processador com 2 linhas de execução. As configurações 6 e 7 diferem apenas no processador mantendo a mesma configuração de memória de instruções e da NoC, e é possível observar que, na configuração 7, mesmo com estes componentes superdimensionados ocorre uma redução no consumo de energia (29% comparado a configuração padrão) e um aumento no desempenho em relação a configuração 6 que comporta o processador com 4 linhas. Isto ocorre devido ao menor tamanho e menor quantidade de *misses* do código compilado para o processador com 2 linhas de execução, utilizando de forma mais eficiente a memória de 8KB apresentada nestas configurações.

Figura 6.1: Resultados para aplicação ADPCM baseados na Tabela 6.2 normalizados para configuração padrão



Fonte: Autor

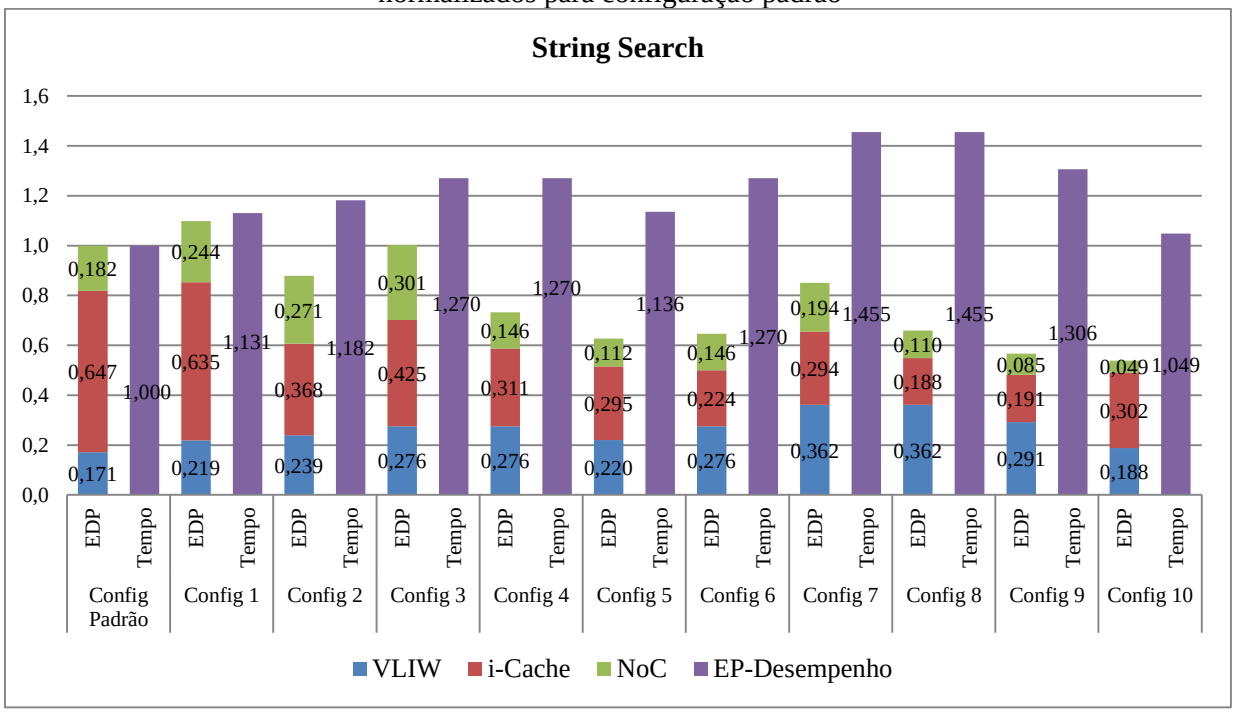
Tabela 6.3: Configurações de Hardware para StringSearch

	Processador	Cache de Instruções				Roteador NoC
		Tamanho	Vias	Linha	Dados	
Configuração Padrão	ρ-VEX 8i	32KB	8	64B	256b	256b
Configuração 1	ρ-VEX 8i	16KB	8	64B	256b	256b
Configuração 2	ρ-VEX 8i	16KB	1	64B	256b	256b
Configuração 3	ρ-VEX 4i	16KB	1	64B	256b	256b
Configuração 4	ρ-VEX 4i	16KB	1	32B	128b	128b
Configuração 5	ρ-VEX 4i	32KB	2	32B	128b	128b
Configuração 6	ρ-VEX 4i	8KB	1	32B	128b	128b
Configuração 7	ρ-VEX 2i	8KB	1	32B	128b	128b
Configuração 8	ρ-VEX 2i	8KB	1	16B	64b	64b
Configuração 9	ρ-VEX 2i	16KB	2	16B	64b	64b
Configuração 10	ρ-VEX 2i	32KB	2	16B	64b	64b

Fonte: Autor

Quando a memória e a NoC são alterados para 64bits, a redução no consumo de energia equipara-se ao consumo da configuração padrão, apesar de 41% de penalidade no tempo de execução.

Figura 6.2: Resultados para aplicação StringSearch baseados na Tabela 6.3 normalizados para configuração padrão



Fonte: Autor

Em busca da redução do EDP com menor dano ao desempenho, as configurações de memória são alteradas para 16KB e 2vias, melhorando o desempenho e reduzindo o consumo energético. Na configuração 10 a memória conta com 32KB e 2 vias de associatividade. Nesta

configuração atinge-se 39% de redução do EDP com penalidade de 12% no desempenho quando comparado a configuração padrão, sendo esta a melhor configuração em termos de EDP.

A aplicação ADPCM possui alta dependência de dados na maior parte do código, desta forma reduzido ILP pode ser extraído pelo compilador. Com esta reduzida capacidade de paralelização, apesar do processador com duas linhas de UFs apresentar um consumo maior do que o apresentado na configuração padrão, o conjunto processador, memória e roteador da NoC atinge a melhor relação entre desempenho e consumo energético.

Para a aplicação *StringSearch* a mesma configuração de processador resultante para a aplicação ADPCM apresenta ser a melhor opção em termos de EDP. Entretanto, diferentemente da ADPCM, pouco sensível ao número de vias da memória cache, a aplicação *StringSearch* consegue explorar melhor este parâmetro. A Tabela 6.3 apresenta a sequência de configurações realizadas em busca do melhor EDP para a aplicação *StringSearch*. O gráfico da Figura 6.2 apresenta os resultados para a aplicação *StringSearch*, onde novamente pode-se observar a importância de configurar todos os componentes do EP.

Utilizando os dados da Tabela 6.1, é possível observar que o tamanho da aplicação *StringSearch* é maior do que 32KB para os processadores com 4 e 8 linhas, desta forma será maior a influência de *miss* em ambas as configurações mesmo quando a maior memória disponível estiver configurada. As configurações 1 e 2, apresentam a variação de EDP e desempenho quando a memória cache de instruções é alterada para 16KB. Nestas configurações é possível perceber que, a associatividade nestes casos promove ou melhor desempenho ou menor consumo de energia, quando é mantido o processador com 8 linhas.

Na configuração 3 e 4, o processador é alterado para 4 linhas divergindo apenas na largura da memória e da NoC. A configuração 4 e 5 provêm me consumo energético, 26% e 37% respectivamente, com 13% de redução de desempenho para a configuração 5 quando comparado a configuração padrão.

A aplicação *StringSearch* compilada para um processador com 2 linhas de execução possui 29KB, cabendo na maior memória disponível, o que reduz o número de *miss*. Apesar disto, nas configurações 8 e 9, com memórias de 8 e 16KB respectivamente, é possível alcançar uma redução de EDP na ordem de 34 e 43%. A configuração 10, contando com processador com 2 linhas e memória com 32KB apresenta o melhor resultado para EDP e

desempenho, ficando a apenas 5% do tempo de execução da configuração padrão com uma redução de 46% no consumo energético.

Tabela 6.4: Configurações de Hardware para OddEven

	Processador	**Cache de Instruções**				**Roteador NoC**
		Tamanho	**Vias**	**Linha**	**Dados**	
Configuração Padrão	ρ-VEX 8i	32KB	8	64B	256b	256b
Configuração 1	ρ-VEX 8i	16KB	8	64B	256b	256b
Configuração 2	ρ-VEX 8i	8KB	1	64B	256b	256b
Configuração 3	ρ-VEX 8i	4KB	1	64B	256b	256b
Configuração 4	ρ-VEX 4i	4KB	1	64B	256b	256b
Configuração 5	ρ-VEX 4i	4KB	1	32B	128b	256b
Configuração 6	ρ-VEX 4i	4KB	1	32B	128b	128b
Configuração 7	ρ-VEX 2i	4KB	1	32B	128b	128b
Configuração 8	ρ-VEX 2i	4KB	1	16B	64b	128b
Configuração 9	ρ-VEX 2i	4KB	1	16B	64b	64b
Configuração 10	ρ-VEX 2i	2KB	1	16B	64b	64b

Fonte: Autor

A aplicação de ordenação *OddEven* provê, principalmente, acesso de leitura e escrita em memória. Apesar disto, algumas instruções que realizam *loads* especulativos são inseridas no código pelo compilador, melhorando ligeiramente o desempenho nos processadores com maior número de recursos.

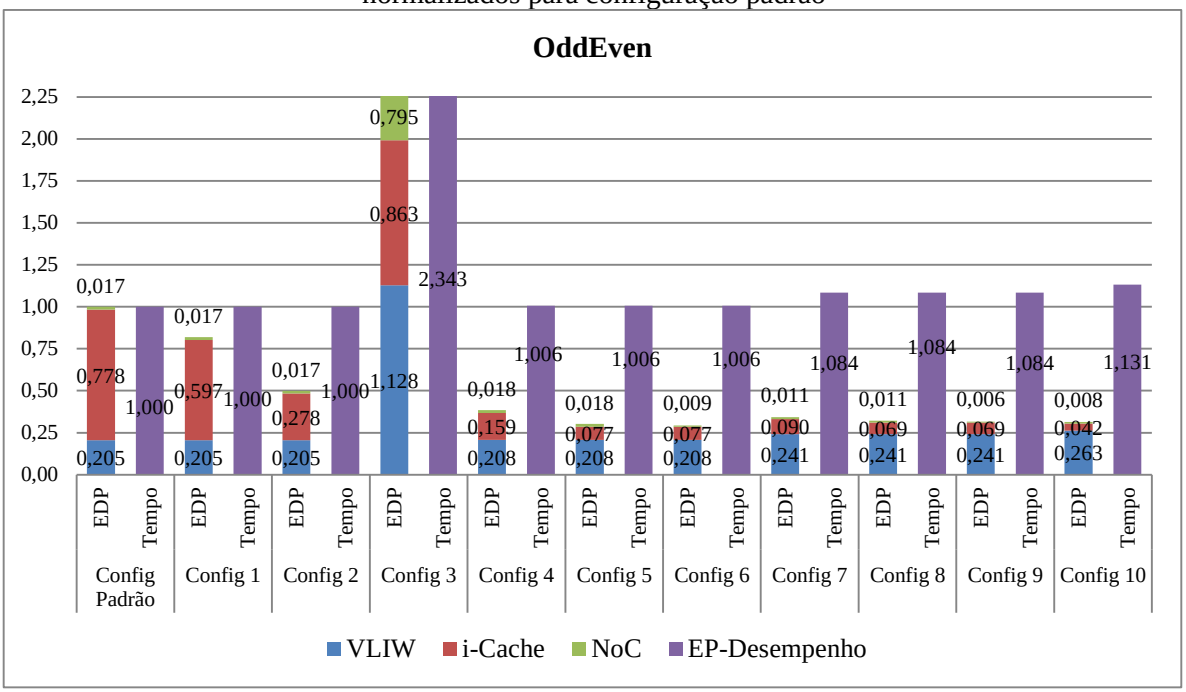
Como apresentado na Tabela 6.1, o tamanho da aplicação *OddEven* varia abaixo de 8KB, sendo assim qualquer que seja a configuração do EP, este não tirará proveito de memórias cache de instruções maiores que 8KB. A configuração padrão apresenta memória de 32KB, superdimensionada para esta aplicação, assim as configurações 1, 2 e 3 mostram o comportamento do EP quando este componente é alterado a fim de melhor se ajustar ao exigido.

As configurações padrão, 1 e 2 da Tabela 6.4, apresentam o mesmo desempenho para 3 diferentes configurações de memória, quando apenas o tamanho varia. Como o tamanho do programa para o processador com 8 linhas é menor que 8KB, a configuração 2 apresenta o menor consumo de energia sem danos ao desempenho para o processador com maior recursos, atingindo 50% de redução.

A Figura 6.3 apresenta os resultados de EDP e tempo de execução para as configurações da aplicação *OddEven*. Quando o tamanho da memória é reduzido para 4KB na configuração 3, o número de *miss* cresce de tal forma que tanto desempenho quanto energia pioram drasticamente. Os processadores com 2 linhas e 4 linhas de execução conseguem uma

redução de energia da ordem de 70%, com penalização menor que 1% ao desempenho para o processador com 4 linhas de execução e 8% para o processador com 2 linhas.

Figura 6.3: Resultados para aplicação OddEven baseados na Tabela 6.4
normalizados para configuração padrão



Fonte: Autor

6.2 Multiplicação de Matrizes e CRC

A Multiplicação de Matrizes é uma aplicação altamente paralelizável. Em virtude desta característica o compilador utilizado consegue encontrar um nível de ILP elevado, gerando códigos capazes de explorar com maior eficiência um maior número de UFs simultaneamente. A Figura 6.4 ilustra o comportamento desta aplicação quando diferentes configurações do elemento de processamento, apresentadas na Tabela 6.5, são utilizadas.

Na aplicação Multiplicação de Matrizes o processador com 8 linhas de execução é capaz de ter suas UFs ocupadas durante grande parte do tempo de execução, fazendo com que energeticamente compense sua utilização. Com o processador de 8 linhas como melhor opção, a memória cache de instruções passa a ser alvo de ajustes mais finos. Conforme ilustrado na Figura 6.4 e seguindo os parâmetros da Tabela 6.5, inicialmente o número de vias desta memória é alterado reduzindo de associatividade com 8 vias para mapeamento direto já que o código cabe integralmente na memória. Na configuração 2 o tamanho da memória é alterado para 16KB, o que resulta no menor EDP, alcançando uma redução de 41% frente a configuração padrão, e neste caso sem danos ao desempenho. Na configuração 3 uma cache

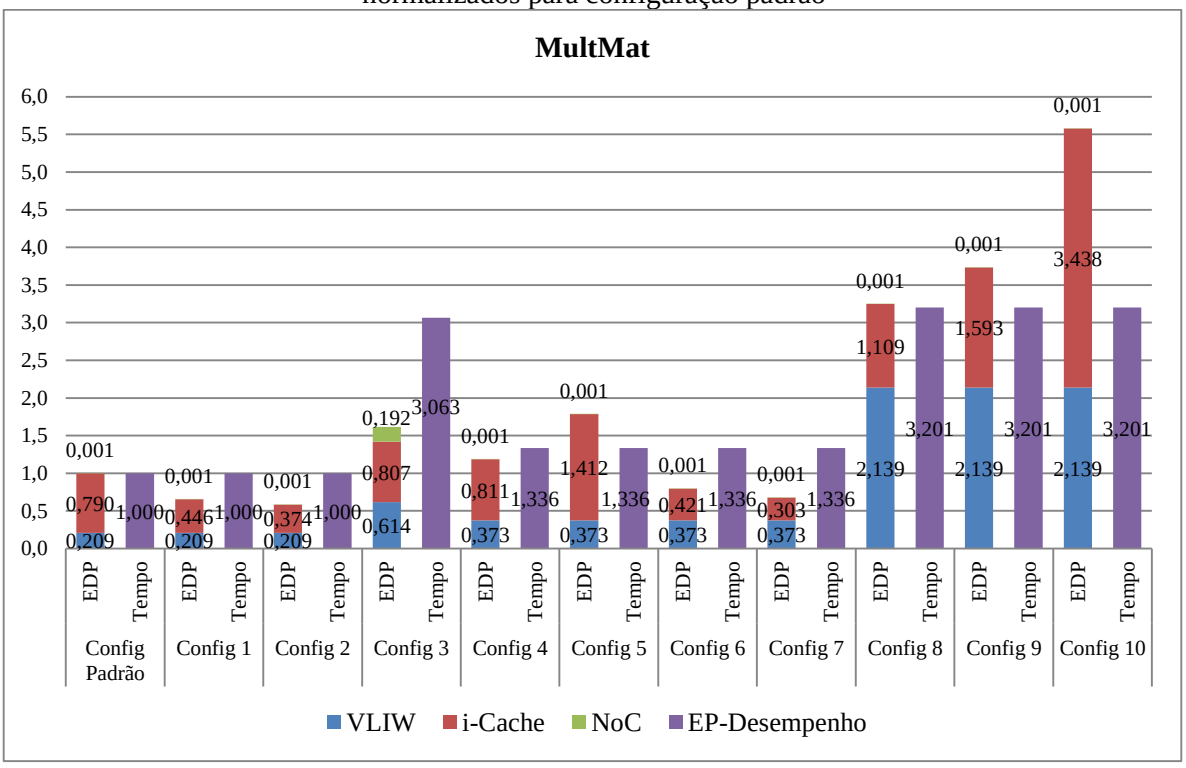
de 8KB é disponibilizada, mas provendo o pior desempenho para este processador e grande elevação do EDP.

Tabela 6.5: Configurações de Hardware para Multiplicação de Matrizes

	Processador	**Cache de Instruções**				**Roteador NoC**
		Tamanho	**Vias**	**Linha**	**Dados**	
Configuração Padrão	ρ-VEX 8i	32KB	8	64B	256b	256b
Configuração 1	ρ-VEX 8i	32KB	8	64B	256b	256b
Configuração 2	ρ-VEX 8i	16KB	1	64B	256b	256b
Configuração 3	ρ-VEX 8i	8KB	4	64B	256b	256b
Configuração 4	ρ-VEX 4i	32KB	8	32B	128b	256b
Configuração 5	ρ-VEX 4i	32KB	8	32B	128b	128b
Configuração 6	ρ-VEX 4i	16KB	1	32B	128b	128b
Configuração 7	ρ-VEX 4i	8KB	1	32B	128b	128b
Configuração 8	ρ-VEX 2i	8KB	1	16B	64b	64b
Configuração 9	ρ-VEX 2i	16KB	1	16B	64b	64b
Configuração 10	ρ-VEX 2i	4KB	1	16B	64b	64b

Fonte: Autor

Figura 6.4: Resultados para aplicação Multiplicação de Matrizes baseados na Tabela 6.5 normalizados para configuração padrão



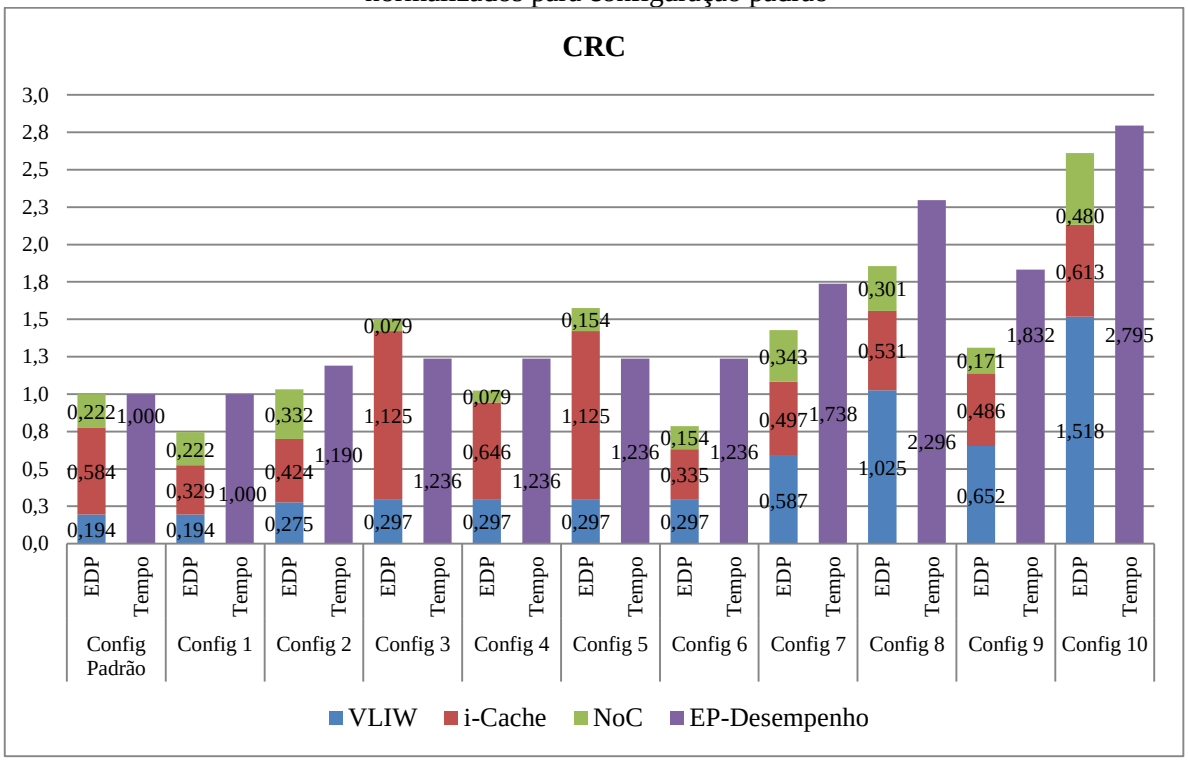
Fonte: Autor

A partir deste ponto, o processador com 4 linhas é experimentado, chegando ao elemento de processamento da configuração 7 composto pelo processador com 4 linhas de execução apresenta uma redução de 32% no EDP, utilizando uma memória cache de

instruções com tamanho de 8KB, quando comparado a configuração padrão, entretanto com um aumento de 33% no tempo de execução. O processador com 2 linhas de execução apresenta o pior desempenho em tempo de execução e energia, mostrando que mais unidades funcionais disponíveis são capazes de acelerar o código com eficiência energética.

Os resultados para a aplicação CRC são apresentados na Figura 6.5 e as configurações pertinentes aparecem na Tabela 6.6. O comportamento assemelha-se ao ocorrido na aplicação Multiplicação de Matrizes, onde o processador com 8 linhas de FUs alcança melhor relação desempenho/energia com redução de 25% do consumo energético sem danos ao desempenho. Entretanto, a diferença entre a configuração 1 com processador de 8 linhas e a configuração 6 com processador de 4 linhas é mínima em termos de consumo de energia, sendo que a configuração 6 atinge redução de 21% no consumo de energia enquanto ao configuração 1 atinge redução de 25% quanto comparado a configuração padrão. Apesar desta pequena diferença na redução do consumo de energia, o desempenho é substancialmente melhor quando a configuração 1 é utilizada, que não penaliza o tempo de execução quando comparado com a configuração padrão, enquanto a configuração 6 tem o desempenho penalizado em aproximadamente 23%.

Figura 6.5: Resultados para aplicação CRC baseados na Tabela 6.6
normalizados para configuração padrão



Fonte: Autor

Tabela 6.6: Configurações de Hardware para CRC

	Processador	**Cache de Instruções**				**Roteador NoC**
		Tamanho	**Vias**	**Linha**	**Dados**	
Configuração Padrão	ρ-VEX 8i	32KB	8	64B	256b	256b
Configuração 1	ρ-VEX 8i	32KB	1	64B	256b	256b
Configuração 2	ρ-VEX 8i	16KB	4	64B	256b	256b
Configuração 3	ρ-VEX 4i	32KB	8	64B	256b	256b
Configuração 4	ρ-VEX 4i	32KB	8	32B	128b	256b
Configuração 5	ρ-VEX 4i	32KB	8	32B	128b	128b
Configuração 6	ρ-VEX 4i	16KB	1	32B	128b	128b
Configuração 7	ρ-VEX 2i	8KB	8	32B	128b	128b
Configuração 8	ρ-VEX 2i	8KB	1	16B	64b	64b
Configuração 9	ρ-VEX 2i	16KB	1	16B	64b	64b
Configuração 10	ρ-VEX 2i	4KB	8	16B	64b	64b

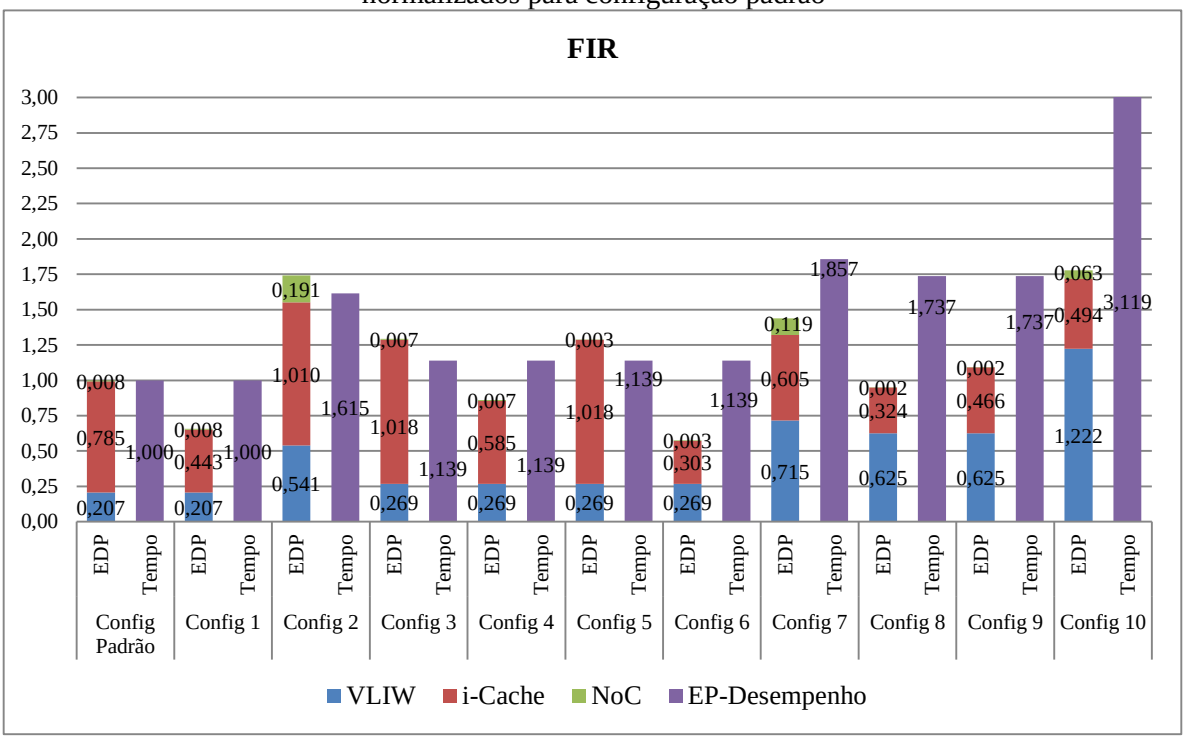
Fonte: Autor

Apesar do comportamento semelhante entre as aplicações de multiplicação de matrizes e CRC, onde ambos apresentam como melhor opção o processador com 8 linhas de execução, a importância do consumo de energia na NoC é notoriamente maior durante a execução da aplicação CRC. É possível fazer uma comparação com o resultado apresentado no gráfico da Figura 6.4, Multiplicação de Matrizes, que ilustra esta importância, onde para a configuração 2 a NoC apresenta apenas 0,1 % da energia total consumida pelo EP, enquanto a energia consumida pela NoC consome 22% do EP na configuração 1 da aplicação CRC conforme o gráfico da Figura 6.5.

6.3 FIR e x264

Algumas aplicações como FIR e x264 apresentam trechos de código onde ora atingem ILP alto ora ILP baixo, fazendo com que o processador com 4 linhas de execução seja capaz de abranger de forma satisfatória a execução destas aplicações, em termos de consumo energético. A Figura 6.6 e a Figura 6.7 apresentam este comportamento, com resultados para as aplicações FIR e x264 respectivamente.

Figura 6.6: Resultados para aplicação FIR baseados na Tabela 6.7 normalizados para configuração padrão



Fonte: Autor

A configuração 1, apresentada na Tabela 6.7 abrange apenas adequação do tamanho da memória cache de instruções, atingindo uma redução de 34% na redução do EDP, sem qualquer dano ao desempenho.

Tabela 6.7: Configurações de Hardware para FIR

	Processador	**Cache de Instruções**				**Roteador NoC**
		Tamanho	**Vias**	**Linha**	**Dados**	
Configuração Padrão	ρ-VEX 8i	32KB	8	64B	256b	256b
Configuração 1	ρ-VEX 8i	32KB	1	64B	256b	256b
Configuração 2	ρ-VEX 8i	16KB	4	64B	256b	256b
Configuração 3	ρ-VEX 4i	32KB	8	64B	256b	256b
Configuração 4	ρ-VEX 4i	32KB	8	32B	128b	256b
Configuração 5	ρ-VEX 4i	32KB	8	32B	128b	128b
Configuração 6	ρ-VEX 4i	16KB	1	32B	128b	128b
Configuração 7	ρ-VEX 4i	8KB	8	32B	128b	128b
Configuração 8	ρ-VEX 2i	8KB	1	16B	64b	64b
Configuração 9	ρ-VEX 2i	16KB	1	16B	64b	64b
Configuração 10	ρ-VEX 2i	4KB	8	16B	64b	64b

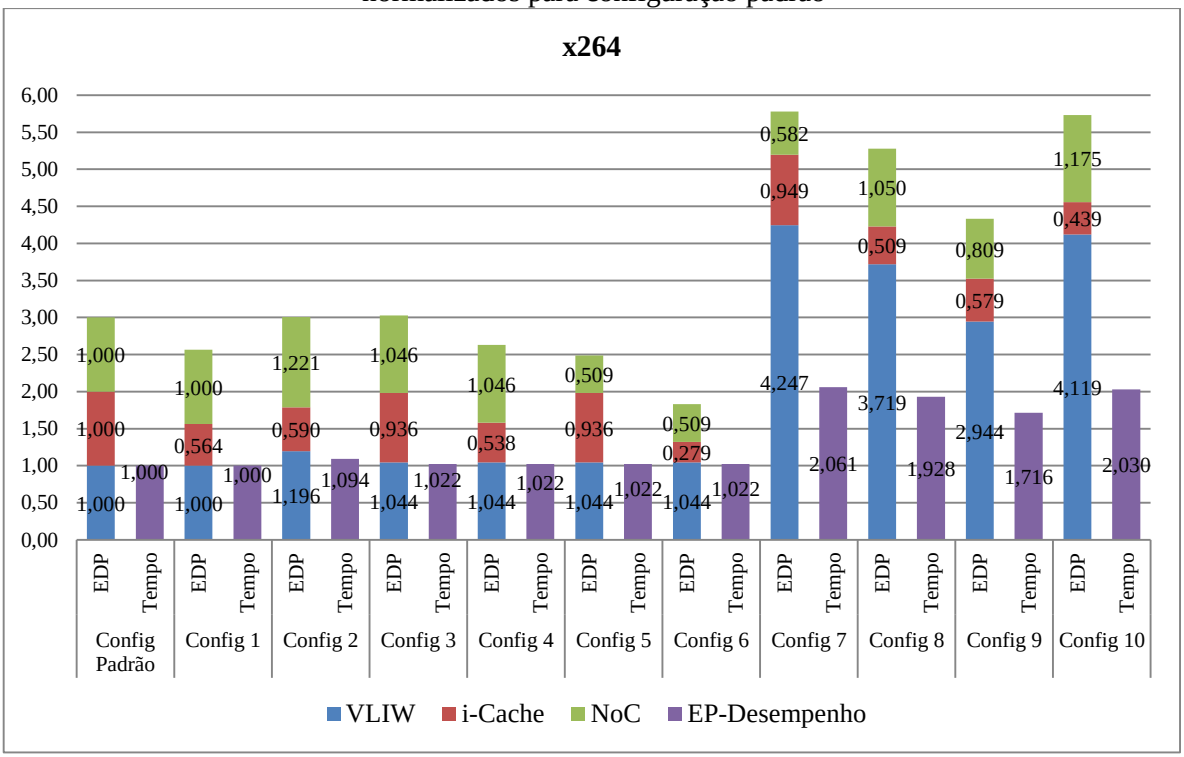
Fonte: Autor

Entretanto, se o foco for redução do EDP mesmo que ocorre mudança no desempenho, a configuração 6 da mesma tabela, abrangendo processador com 4 linhas de execução, memória cache de instruções com 16KB e mapeamento direto com NoC de 128bits, alcança uma redução de 42% de redução do EDP com 13% de aumento no tempo de execução.

A configuração 8 da Tabela 6.7, composta por processador com 2 linhas de execução e memória com tamanho de 8KB atinge um EDP menor que o EDP padrão, em aproximadamente 5%, apesar de uma penalizada de 74% no desempenho. Novamente, isto se dá devido ao tamanho da memória de instruções, que para o processador com 2 linhas é 4,5 vezes menor do que para o processador com 8 linhas.

O comportamento da aplicação FIR repete-se na aplicação x264, onde o processador com 4 linhas de execução apresenta ser a melhor opção. As configurações para a aplicação x264 são apresentas naTabela 6.8. Através da Figura 6.7 pode-se notar o comportamento da reconfigurabilidade total do sistema, onde o EP configurado com processador com 4 linhas e 16KB de memória de instruções atinge o menor resultado de EDP. Esta configuração atinge 39% de redução na métrica EDP com 2% de redução no tempo de execução. É possível observar também que um simples ajuste na associatividade da memória cache apresentado na configuração 1 é capaz de promover redução de 14% no EDP sem redução no desempenho.

Figura 6.7: Resultados para aplicação x264 baseados na Tabela 6.8
normalizados para configuração padrão



Fonte: Autor

Tabela 6.8: Configurações de Hardware para x264

	Processador	Cache de Instruções				Roteador NoC
		Tamanho	Vias	Linha	Dados	
Configuração Padrão	ρ-VEX 8i	32KB	8	64B	256b	256b
Configuração 1	ρ-VEX 8i	32KB	1	64B	256b	256b
Configuração 2	ρ-VEX 8i	16KB	4	64B	256b	256b
Configuração 3	ρ-VEX 4i	32KB	8	64B	256b	256b
Configuração 4	ρ-VEX 4i	32KB	8	32B	128b	256b
Configuração 5	ρ-VEX 4i	32KB	8	32B	128b	128b
Configuração 6	ρ-VEX 4i	16KB	1	32B	128b	128b
Configuração 7	ρ-VEX 4i	8KB	8	32B	128b	128b
Configuração 8	ρ-VEX 2i	8KB	1	16B	64b	64b
Configuração 9	ρ-VEX 2i	16KB	1	16B	64b	64b
Configuração 10	ρ-VEX 2i	4KB	8	16B	64b	64b

Fonte: Autor

Diferente da aplicação FIR, o processador com 2 linhas de execução não é capaz de apresentar qualquer redução no consumo de energia em qualquer das configurações disponíveis. Isto ocorre devido a grande aceleração alcançada no processador com 4 e 8 linhas frente ao processador com 2 linhas. Assim, mesmo com menor tamanho de programa possibilitando o uso de menor memória de instruções, este processador não é capaz de alcançar valores de redução de energia ou aceleração.

6.4 Considerações e Conclusão

O tamanho do binário do programa é crucial para a escolha do tamanho ideal da memória cache de instruções, já que se o programa couber inteiramente na memória de instruções reduzirá o número de *miss* para ZERO. Entretanto, se a aplicação resultar em poucos laços, a maior parte do código jamais se repetirá fazendo com que parte da memória fique ociosa e as instruções contidas neste espaço sejam inúteis não sendo mais acessadas. Neste ponto, a memória ideal é aquela que apresenta a menor quantidade de códigos que são referenciados apenas uma vez (ou poucas), desde que não acresça no número de misses.

Partindo do princípio que se consiga analisar o tamanho de cache ideal para cada programa, este capítulo mostrou que a reconfigurabilidade de todas as principais partes que constituem o sistema traz benefícios que abrangem diversas questões.

A redução da dissipação de potência é atingida, visto que menos área de silício fica ativa quando uma reconfiguração precisa é realizada. Esta redução de potência por área reduz

a densidade de potência, reduzindo custos com tratamento de dissipação e possibilitando uma redução no efeito de *DarkSilicon.*

A redução do consumo de energia é outra questão abrangida, visto que apenas o hardware necessário para execução de determinada aplicação é ativado, assim menor será o consumo. A aceleração do código frente ao processador mestre é outra questão importante, visto que menor será o tempo de execução das aplicações, e por consequência, menores serão os tempos em que o hardware fica ativo consumindo energia e dissipando potência.

7 SOFTWARE EM SISTEMAS HETEROGÊNEOS RECONFIGURÁVEIS

O capítulo 6 mostra a importância da reconfigurabilidade dos EPs de forma que estes se adaptem à aplicação em todos os níveis. Esta adaptação é capaz de prover significativa redução do consumo de energia alcançando melhores relações desempenho/energia. Porém, quando um sistema complexo é constituído de diversos EPs reconfiguráveis e adaptáveis à aplicação outro desafio surge: como manter o software compatível com tais adaptações destinando-o aos adequados aceleradores?

Em sistemas modernos *drivers*, *APIs*, e bibliotecas são responsáveis por possibilitar o uso de hardwares heterogêneos, onde a aplicação é apresentada previamente compilada para um determinado hardware alvo. Entretanto, quando estes hardwares são dispostos de forma dinâmica com modificações na quantidade ou disponibilidade de unidades funcionais, por exemplo, estes mecanismos tornam-se ineficientes já que o software é rígido e específico para determinado acelerador, normalmente subutilizando os recursos disponíveis, como processador, memórias cache e comunicação. Outra subutilização de recursos ocorre quando é feita uma atualização no hardware, como a instalação de um novo acelerador, possível principalmente quando um FPGA está disponível. Diferentes aplicações podem apresentar diferentes comportamentos, sejam por sua natureza ou pelas diferentes possíveis entradas, assim novamente, subutilizando recursos.

Para auxiliar na demonstração onde a reconfigurabilidade dos EPs pode ser utilizada com diversos simultâneos focos, este capítulo apresenta uma proposta que, ao mesmo tempo em que possibilita a utilização de um processador VLIW reconfigurável como um eficiente acelerador, busca automatizar esta reconfigurabilidade e manter a compatibilidade binária auxiliando na produtividade.

A proposta é baseada em um tradutor binário estático, de forma que o binário original é traduzido para o binário alvo. O objetivo deste tradutor binário é ser utilizado como ferramenta para auxiliar na demonstração da capacidade do hardware reconfigurável ser utilizado conforme o comportamento demonstrado e experimentado no capítulo 6, obtendo aceleração e redução do consumo de energia.

7.1 Tradutor Binário

Tradutores binários são amplamente apresentados na literatura, e tem uma de suas primeiras publicações datada da década de 1960 com interpretadores do tipo JIT (do inglês, *Just in Time*) (UMAP, 1966). Esta primeira aparição apresenta uma anotação contida no manual do IBM 7090, um computador transistorizado destinado a aplicações cientificas. Estas anotações explicitam que o *assembler* e o *loader* deste processador poderiam traduzir um código para a máquina em questão durante a execução. Em 1968, Thompson mostra a compilação de expressões regulares para o processador IBM 7094 durante a execução (THOMPSON, 1968).

Diferentemente de compiladores, os tradutores binários normalmente não possuem a mesma disponibilidade de recursos de tempo para efetuarem a tradução. Os tradutores necessitam traduzir o binário de uma arquitetura em outra arquitetura de forma que o tempo de execução do binário resultante seja ao menos satisfatório, no caso de simples compatibilidade binária, ou mais rápido quando o objetivo é acelerar a aplicação.

Os tradutores podem utilizar de diversos recursos e basicamente subdividem-se em emuladores, tradutores estáticos e tradutores dinâmicos, onde:

- Emuladores interpretam aplicações em tempo de execução. Entretanto, após completada a análise pelo tradutor o código já executado não é mantido em memória para uso futuro e é normalmente descartado. Os emuladores tem pouco espaço para otimizações de código, por outro lado são mais simples de serem implementados.
- Tradutores binários estáticos convertem um código de uma arquitetura para outra basicamente lendo o binário original sem a necessidade de executá-lo. A tradução estática é realizada uma única vez por código, desde que a arquitetura do processador alvo não seja modificada, sendo assim possível a execução de tal aplicação novamente, sem a necessidade de nova tradução. Esta forma de tradução tem algumas limitações devido ao fato de que todo o código é traduzido de uma única vez, isto é, sem qualquer tipo de *feedback*, tornando o desempenho critico para o tradutor e para o binário resultante. Assim a construção de um tradutor estático deve ser realizada com atenção: um tempo muito longo entre o início da tradução e a conclusão da execução do binário alvo pode penalizar o desempenho de forma significativa. A implementação de tradutores binários estáticos são desafiadoras, não apenas em virtude do

desempenho, mas também quanto à otimização do código alvo. Apesar disto, a tradução estática tem a vantagem de poder ler o código fonte antes de executá-lo.

- Tradutores binários dinâmicos, diferentemente dos estáticos, referem-se a tradução que ocorre durante a execução do código, sendo realizada em trechos por vez. Desta forma, torna-se mais fazível encontrar melhorias para a arquitetura alvo, já que é possível analisar os diferentes comportamentos do código e contar com *feedbacks* provenientes da execução. Este tipo de tradução pode ser crítica em termos de tempo, especialmente quando for necessário que um trecho de código seja executado múltiplas vezes buscando-se possíveis melhorias.

7.2 Compatibilidade

A compatibilidade binária é um dos obstáculos que prorrogam a migração para arquiteturas novas que se apresentam mais capazes tanto em termos de desempenho como energia. A compilação de uma aplicação para cada arquitetura disponível é algo custoso, e a disponibilização desta aplicação para cada nova arquitetura, usufruindo efetivamente dos novos recursos, é inviabilizada pela dinâmica de mercado que requer produtividade virtualmente instantânea. Da mesma forma ocorre em sistemas multiprocessados heterogêneos, onde um novo acelerador pode ser adicionado ao sistema sem ser corretamente aproveitado. Ou ainda, uma determinada aplicação pode ser executada com melhor eficiência em um ou outro acelerador (ou diferentes configurações do mesmo) quando apenas um dos parâmetros desta aplicação varia, entretanto, como a aplicação já é pré-compilada não será possível utilizar de maneira mais eficiente o diferente comportamento da aplicação.

Apesar de a compatibilidade binária aparecer como limitante para melhor aproveitamento de novas arquiteturas ou tecnologias, algumas soluções utilizando a tradução binária são vistas amenizando esta limitação.

No início dos anos 2000, o processador Crusoé (TRANSMETA, 2001) foi apresentado pela Transmeta utilizando um tradutor binário para tornar o seu processador VLIW compatível com instruções x86, tornando-o compatível com binários amplamente difundidos no mercado. Nesta primeira versão do processador Crusoé, o processador VLIW contava com 128bits de largura de instruções, isto é, um processador com 4 linhas de execução de instruções 32bits cada. A compatibilidade com o conjunto de instruções x86 era obtido através de um software de tradução, batizado de *Code Morphing Software*, que tinha a função de converter as instruções x86 enviadas pelos programas em instruções RISC, mais simples e

compatíveis com o processador VLIW, ordená-las adequadamente e coordenar o uso dos registradores, tarefas que em outros processadores são executadas via hardware. O tradutor *Code Morphing* era armazenado em uma ROM integrada diretamente ao processador e utilizava espaços protegidos das memórias caches L1 e L2 tanto de dados quanto instruções além de reservar espaço na memória RAM.

A utilização dos recursos do processador e das memórias divididas entre a tradução e o processamento do código traduzido foi um dos problemas de desempenho apresentados pelo processador Crusoé, apesar de alcançar reduzido consumo energético. Mais tarde a Transmeta amenizou o problema de desempenho quando ampliou o número de linhas de execução do processador, passando de 4 para 8 linhas de execução, fazendo com que de forma dinâmica apenas o número necessário de unidades funcionais eram utilizados para tradução. Assim, mais unidades estavam disponíveis para a execução da aplicação propriamente, capacitando o processador a alcançar melhor desempenho ainda com consumo energético satisfatório. Uma possibilidade interessante do *Code Morphing* é a capacidade de atualização deste tradutor, podendo torna-lo mais eficiente ou compatível com novas ISAs.

Outras soluções para manter a compatibilidade binária são as máquinas virtuais, como ocorre com o JAVA, por exemplo, que se utiliza de uma máquina virtual, posicionando-se entre a máquina física (ou sistema que nela é executado) e o programa a ser executado pelo usuário (escrito na linguagem JAVA), interpretando ou traduzindo os códigos das aplicações de um para o outro.

7.3 Heterogeneidade e Reconfigurabilidade

Quando um sistema heterogêneo é elaborado, a variedade de EPs disponíveis neste sistema requer uma variedade de ferramentas, como compiladores, para possibilitar a execução de determinadas aplicações em seus adequados aceleradores alvo. Quanto mais distintos estes EPs, mais distintas serão estas ferramentas. Assim, em sistemas com centenas de elementos de processamentos, a heterogeneidade passa a ser uma questão crítica para a produtividade. Se estes elementos forem reconfiguráveis, o que energeticamente apresenta-se atraente, maior serão as variações destas ferramentas.

Assim, a ferramenta proposta neste trabalho prima por auxiliar na implantação da heterogeneidade e da reconfigurabilidade em sistemas multiprocessados, possibilitando o uso de aceleradores redutores de energia e aceleradores de forma simultânea, principalmente quando os processadores destes EPs são reconfiguráveis.

7.4 Tradutor Proposto

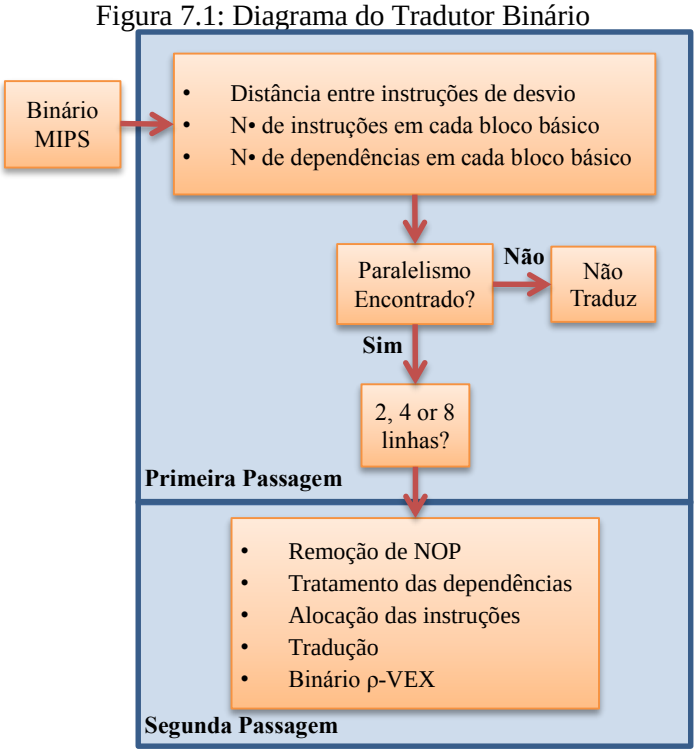
O tradutor proposto é um tradutor binário estático capaz de traduzir um binário MIPS I para o processador VEX e simultaneamente escalar a melhor configuração para este processador, para cada aplicação. Três metas são o foco deste tradutor:

- Manter a compatibilidade binária entre o processador MIPS e o processador VEX sem a necessidade de utilizar outro compilador diferente daquele destinado ao MIPS
- Acelerar o código MIPS
- Determinar a melhor configuração do processador VLIW reconfigurável para executar determinada aplicação.

O objetivo principal deste tradutor é acelerar os códigos que originalmente foram compilados para o processador MIPS, utilizando estritamente o hardware necessário para executar a aplicação, reduzindo o consumo energético sem danos ao desempenho.

A ferramenta proposta foi escrita em linguagem C, buscando o menor e mais rápido código possível, provendo fácil portabilidade e compatibilidade possibilitando futuras atualizações e implementações.

Figura 7.1: Diagrama do Tradutor Binário



Fonte: Autor

O tradutor é baseado em tabelas para que através de uma equivalência de instruções sejam substituídas as instruções MIPS por instruções VEX, diretamente quando possível, e

por um arranjo de instruções quando necessário. Assim, o tradutor funciona executando duas passagens pelo binário MIPS. Em cada uma destas passagens tarefas são executadas com o intuito de extrair desde informações a respeito de blocos básicos, números de instruções de desvio e salto, número de dependências e outras informações como endereçamento, além de efetivar a tradução propriamente dita. O diagrama mostrado na Figura 7.1 ilustra o funcionamento do tradutor.

7.4.1 Primeira Passagem

Na primeira passagem, o tradutor faz a leitura do código coletando informações que auxiliarão na composição do código alvo. Três destas informações são essenciais:

- Distância entre instruções de desvio (em números de instruções)
- Média do número de instruções em cada bloco básico
- Média do número de dependências em cada bloco básico

A distância entre instruções de desvio auxilia para determinar o número de blocos básicos. Além disto, é necessário identificar as instruções de desvio e salto para que possam ser calculados os novos endereços quando a tradução for efetivada.

A configuração ideal do processador é determinada nesta passagem, utilizando o número médio de instruções por bloco básico e a quantidade média de instruções com dependências nestes blocos. Assim, com estas informações é calculada a média de instruções passíveis de paralelismo no código. A simples divisão entre número médio de instruções e o número médio de dependências dentro de um mesmo bloco básico fornece, de forma especulativa a possibilidade de paralelismo a ser explorado. Entretanto, outras questões como disponibilidade de recursos do processador, por exemplo, restrições das unidades de acesso à memória, devem ser levadas em consideração. Estes números variam dependendo da aplicação, e obviamente não resultarão em valores exatos, sendo aplicada uma tabela de conversão, tal qual a Tabela 7.1.

Algumas instruções são críticas para o tradutor, principalmente devido à natureza reconfigurável do processador alvo. Então, durante a primeira passagem e com o auxilio das informações extraídas nela, alguns marcadores são mantidos no código. Estes marcadores são colocados em endereços destino já conhecidos por instruções de salto, e serão ajustados durante a segunda passagem.

Tabela 7.1: Valores de Conversão relacionados ao número de linhas do processador.

Tradutor Binário	Número de linhas de execução adotado
resultado < 1,5	Não traduz
1,5 =< resultado < 3,0	2
3,0 =< resultado < 6,0	4
resultado > 6,0	8

Fonte: Autor

Como o processador utilizado dispõe de 2, 4 ou 8 linhas de execução, podendo executar 2, 4 ou 8 instruções RISC, respectivamente, conforme a Tabela 5.1, o ajuste é baseado nestes recursos.

7.4.2 Segunda Passagem

Após a definição da configuração do processador e adicionar os marcadores de instruções de salto e desvio na primeira passagem, a segunda passagem inicia onde a tradução é concluída de acordo com as tabelas de tradução.

O processador ρ-VEX implementa uma ISA RISC, tornando a maioria das instruções similares às instruções MIPS. Entretanto algumas instruções do processador MIPS não possuem equivalentes na ISA do processador ρ-VEX, sendo necessário maior esforço nestas traduções. Instruções como BEQ (*branch on equal*), BNE (*branch not equal*) e outras que abrangem comparação e salto na mesma instrução não estão presentes no conjunto de instruções ρ-VEX. Assim, diversas instruções ρ-VEX são utilizadas para traduzir este tipo de instrução MIPS, como exemplificado no trecho de código da Tabela 7.2.

Outro grupo de instruções que possuem adaptações são as instruções de DIV, que abrangem divisão de inteiros e extração de resto. O processador MIPS possui, originalmente, hardware específico para estas instruções enquanto o processador ρ-VEX implementa estas instruções em software, através de uma chamada para uma função DIV que conta com 36 ciclos, não importando a configuração do processador ou o valor de suas entradas, salvo se resultar ZERO. Neste grupo de instruções, um código pré-estabelecido é inserido no binário destino para que, *inline*, sejam executadas.

Tabela 7.2: Comparação exemplo entre código MIPS e código ρ-VEX

Código MIPS	Código ρ-VEX
beq r2, r3, L0 *add r2, r0, r3*	*cmpeq b0 = r2, r3* *br b0, L0* *add r2, r0, r3*

Fonte: Autor

Como o objetivo desta ferramenta é auxiliar no uso de aceleradores heterogêneos e reconfiguráveis de forma a manter a compatibilidade binária e a produtividade, três metas são essenciais durante a tradução:

- Executar o menor número de instruções NOP possível
- Evitar a dependência de dados, tratando ou contornando com escalonamento
- Alocar as instruções de forma efetiva

Na primeira passagem as instruções de NOP são mantidas como no binário original, em virtude da manutenção dos corretos endereços de salto e desvio. Durante a segunda passagem estas instruções são removidas e apenas serão adicionadas caso não seja possível preencher todas as unidades funcionais do processador VLIW.

As dependências reais, tradicionalmente são tratadas com técnicas como renomeação de registradores, entretanto para este trabalho uma solução mais simples foi experimentada, onde estas instruções serão atrasadas ou adiantadas buscando amenizar seus impactos no desempenho. Obviamente a extração de paralelismo é ampliada quando a renomeação de registradores é utilizada, entretanto este não é o objetivo principal da ferramenta apresentada, assim foram escolhidas técnicas visando manter o tradutor o mais simples possível.

7.4.3 Resultados

Para demonstrar a capacidade de um sistema heterogêneo reconfigurável alcançar aceleração e economia de energia ao mesmo tempo, sem a necessidade de diversos compiladores, a ferramenta proposta é experimentada utilizando o mesmo hardware e conjunto de aplicações apresentados no capítulo 6.

A comparação foi realizada entre o processador MIPS e o processador VLIW reconfigurável ρ-VEX. Como o tradutor proposto encontra apenas soluções para configuração do processador, estas comparações são realizadas apenas em relação aos processadores, com os binários das aplicações gerados com o compilador HP/ST e com o tradutor binário.

Os resultados para o processador MIPS foram extraídos através do simulador GEM5 (BINKERT e BECKMANN, 2011), em conjunto com o compilador MIPS GCC versão 4.5.2. Assim como no capítulo 6, o compilador HP/ST (HP LABS, 2010) teve como parâmetros de compilação *flags* que provessem o maior desempenho possível em todas as aplicações. No caso do tradutor binário, este traduz o binário gerado inicialmente para o processador MIPS

com o uso do compilador MIPS GCC versão 4.5.2, também sendo configurado para o maior desempenho.

Tabela 7.3: Tamanho da memória de instruções para as aplicações com o tradutor

	Tamanho (KB)			
Aplicação	**MIPS**	**ρ-VEX 2-linhas**	**ρ-VEX 4-linhas**	**ρ-VEX 8-linhas**
ADPCM encode	13,5	26,99	51,23	101,96
StringSearch	5,5	10,35	19,4	38,22
OddEven	0,31	0,61	1,2	2,28
Multiplicação de Matrizes	0,76	1,55	3,06	4,59
CRC	3,65	7,4	13,0	25,75
FIR	1,9	3,63	6,73	13,4
x264 decode	7,1	13,43	24,64	49,18

Fonte: Autor

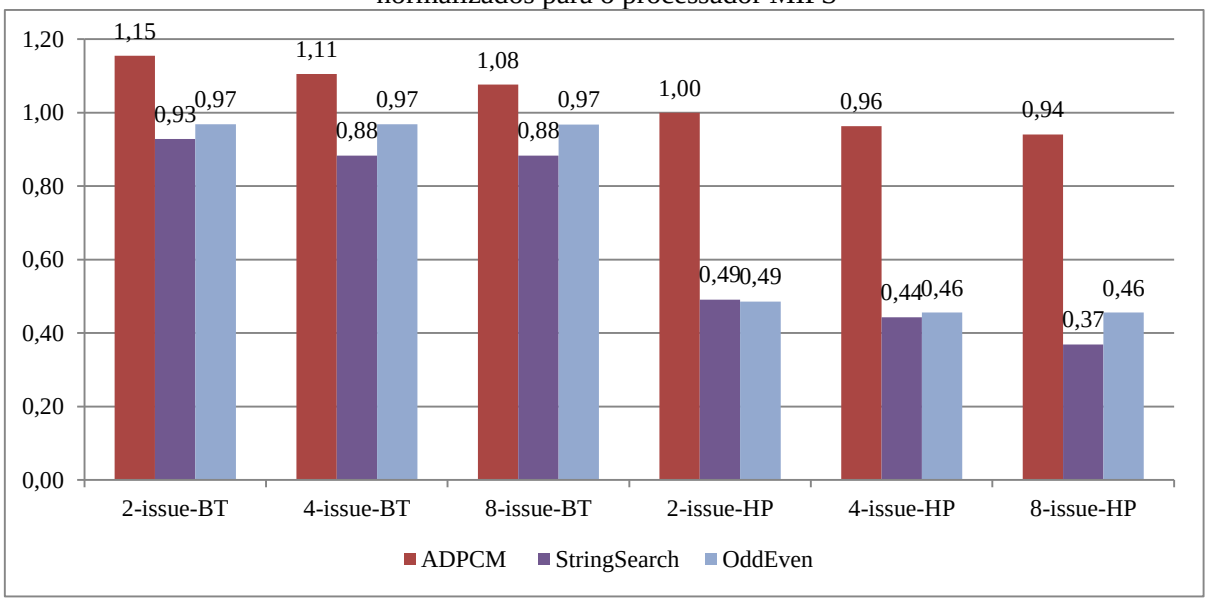
O tradutor binário é executado no próprio processador Master, neste caso o MIPS, e então o binário é disponibilizado para o processador alvo. A Tabela 7.3 apresenta as aplicações testadas, além do tamanho das memórias de instruções geradas para cada aplicação quando utilizado o tradutor proposto.

7.4.3.1 ADPCM, StringSearch e OddEven

A Figura 7.2 apresenta os resultados de desempenho para as aplicações ADPCM, StringSearch e OddEven. São apresentados resultados para os processadores VLIW com o código compilado com a ferramenta HP/ST e com o binário gerado pelo tradutor proposto. Estas aplicações são mais eficientes energeticamente quando executadas no processador com 2 linhas de execução, como visto na Seção 6.1.

A aplicação ADPCM é a mais crítica para o tradutor binário. O desempenho desta aplicação quando o processador executa o binário proveniente do tradutor é pior quando comparado com o processador MIPS, não alcançando aceleração. Como a aplicação ADPCM utiliza-se de muitos laços com dependências reais que não permitem o uso de técnicas capazes de *loop unrolling*, e o tradutor não realiza a renomeação de registradores, o desempenho acaba por ser penalizado. Outra questão que reduz drasticamente o desempenho nesta aplicação é o fato de que são utilizadas cerca de 5000 instruções de divisão por parte do MIPS. O processador MIPS executa a instrução DIV em hardware cujo número de ciclos é dependente da entrada, enquanto o processador ρ-VEX executa tais instruções em software, com penalidade fixa de 36ciclos.

Figura 7.2: Comparativo de Resultados de desempenho para *ADPCM, StringSearch e OddEven* normalizados para o processador MIPS



Fonte: Autor

O compilador HP/ST é capaz de extrair maior paralelismo, já que sua análise ocorre a partir do código fonte, sendo capaz de executar diversas técnicas durante a construção das instruções além de utilizar maior número de registradores e instruções de *LOAD especulativas*. Mesmo com a utilização destas técnicas, o compilador foi capaz de acelerar ao máximo de 6% em relação ao processador MIPS quando o processador com maior número de recursos é disponibilizado.

Nas aplicações *StringSearch* e *OddEven* foi possível acelerar o código utilizando tanto o compilador quanto o tradutor. Entretanto, a aplicação *OddEven* apresenta a mesma aceleração para as 3 configurações do processador quando o tradutor é utilizado, alcançando 3% de desempenho frente ao processador MIPS. Enquanto o compilador HP é capaz de alcançar mais de 51% de aceleração quando comparado ao MIPS.

Utilizando o binário gerado pelo tradutor, um comportamento semelhante é alcançado com a execução da aplicação *StringSearch*, com pequeno aumento na aceleração quando são ampliados os recursos do processador. Este comportamento é visível nos processadores com 4 e 8 linhas. Novamente, a capacidade do compilador HP em explorar, principalmente, o maior número de registradores disponíveis o torna mais eficiente.

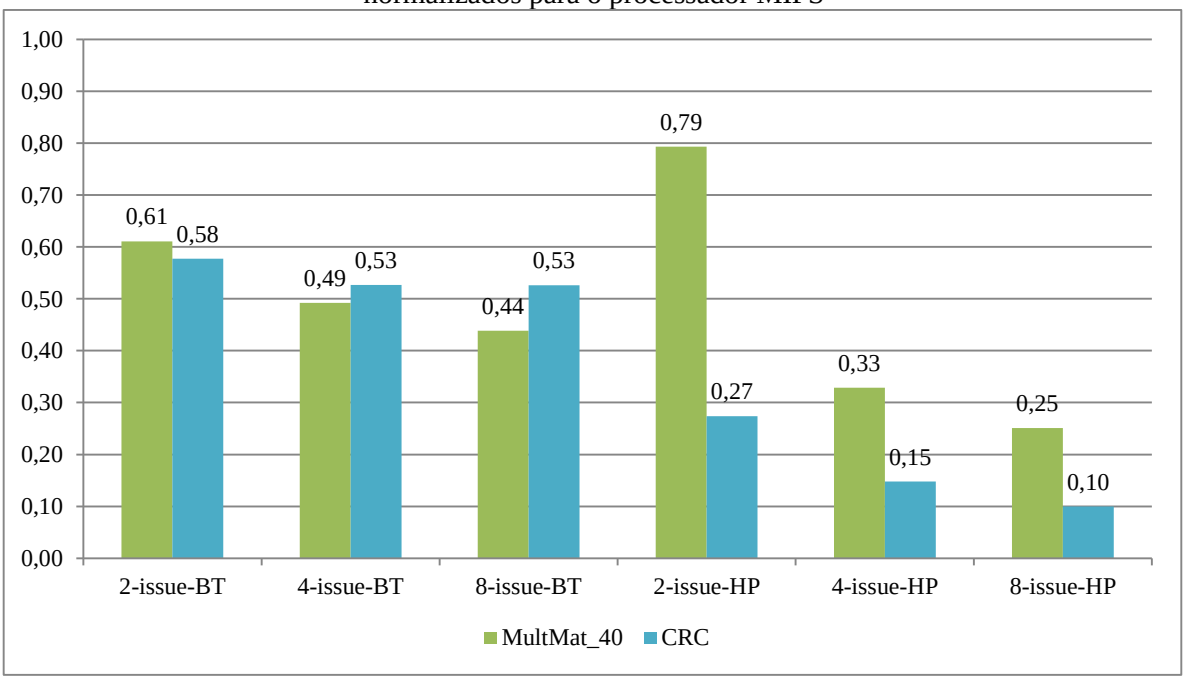
7.4.3.2 Multiplicação de Matrizes e CRC

Para as aplicações de Multiplicação de Matrizes e CRC o compilador GCC MIPS é capaz de aplicar técnicas como *loop unrolling*, mas como sua arquitetura alvo não permite ocupar mais que uma unidade funcional por vez, não é capaz de qualquer paralelismo no estágio de execução. Desta forma, o fato do compilador GCC MIPS desenrolar laços auxilia apenas em reduzir o número de instruções de desvio executadas e o número de acessos à pilha, apesar de reuso de registradores.

O reuso dos registradores, nestes casos, limita a capacidade do tradutor binário paralelizar o código, já que este apenas reordena as instruções para explorar ILP. De qualquer forma, a Figura 7.3 apresenta os resultados de desempenho para as aplicações desta seção, onde é possível observar aceleração de código.

Ambas as aplicações analisadas nesta seção são passíveis de paralelização pelo tradutor, entretanto Multiplicação de Matrizes apresenta um comportamento diferente quando o processador alvo possui 2 linhas de execução. O compilador HP é capaz de acelerar em 21% frente ao processador MIPS, já o tradutor consegue acelerar em 39% este código com o mesmo processador alvo para esta aplicação.

Figura 7.3: Comparativo de Resultados de desempenho para Multiplicação de Matrizes e CRC normalizados para o processador MIPS



Fonte: Autor

Primeiramente, é importante notar que o processador com 2 linhas de execução é configurado com unidades de multiplicadores e ULAs concorrentes, isto é, não é possível utilizar os dois multiplicadores e as duas ULAs ao mesmo tempo, além disto, as unidades de desvio e acesso a memória também dividem espaço com os multiplicadores e ULAs. Desta forma, aproxima-se mais do comportamento apresentado no processador MIPS. Outra questão que deve ser levada em consideração é o fato de que os parâmetros utilizados para o compilador HP força o uso de extremo *loop unrolling*, fazendo com que aumente a concorrência entre as UFs disponíveis no processador VLIW com esta configuração.

Quando é aumentada a disponibilidade de recursos o tradutor binário escala acompanhando este crescimento, alcançando 51% e 56% de aceleração, respectivamente para o processador com 4 e 8 linhas de execução. O mesmo comportamento é observado quando o compilador HP gera o binário, alcançando 67% e 75% respectivamente ao crescimento de recursos.

Na aplicação CRC o tradutor binário tem maiores dificuldades de extrair paralelismo principalmente quando o processador escala de 4 para 8 linhas de execução. Obtendo sua melhor relação recursos/desempenho no processador de 4 linhas. Já o compilador é capaz de extrair paralelismo em todas as diferentes configurações do processador VLIW.

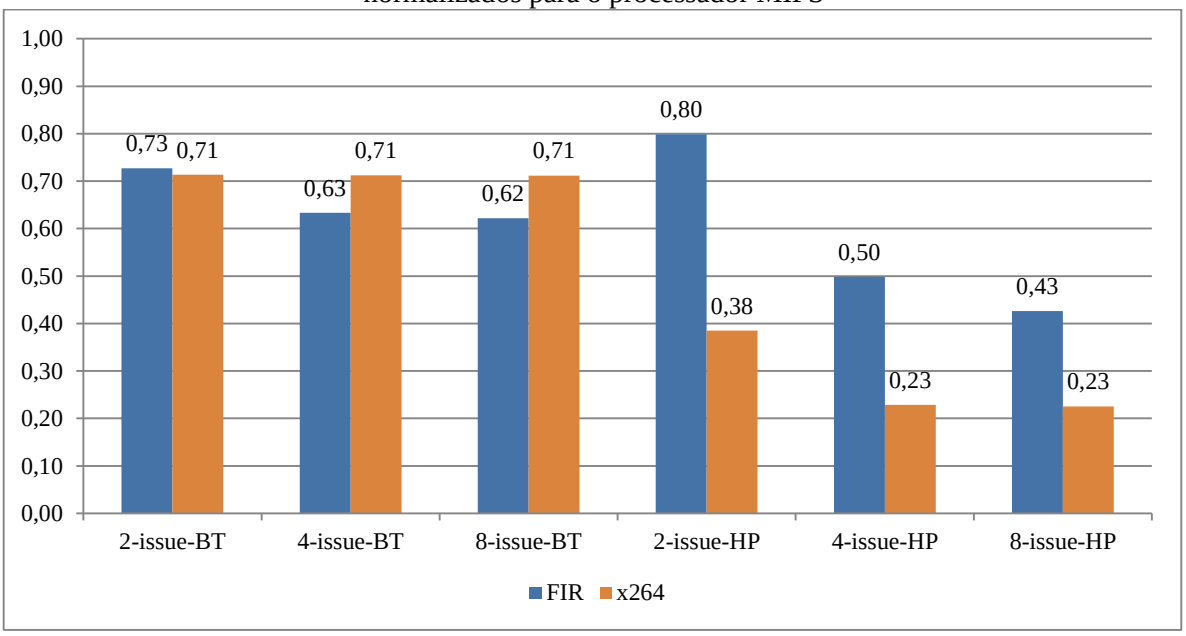
7.4.3.3 FIR e x264

Algumas aplicações apresentam comportamentos heterogêneos em termos de ILP, isto é, ora é possível extrair amplo ILP, ora o ILP aparece reduzido. A Figura 7.4 apresenta os resultados para estas aplicações.

Para a aplicação FIR, que utiliza multiplicação de matrizes, o compilador HP é capaz de escalar em todas as configurações do processador, entretanto quando o processador com 8 linhas é disponibilizado, a aceleração cresce apenas 7% em relação ao processador com 4 linhas. Semelhante ao que ocorre com a aplicação Multiplicação de Matrizes, a aplicação FIR também apresenta melhor desempenho com o tradutor binário quando o processador com 2 linhas de execução disponibilizado.

Apesar disto, o tradutor não é capaz de acelerar com a mesma agressividade do que o compilador HP. Novamente, a quantidade de acessos à pilha por parte do compilador GCC MIPS e o limite no *loop unrolling* executado limita o trabalho do tradutor. Assim, o tradutor permite praticamente a mesma aceleração tanto no processador com 4 linhas quanto 8 linhas.

Figura 7.4: Comparativo de Resultados de desempenho para FIR e x264 normalizados para o processador MIPS



Fonte: Autor

A aplicação x264 é acelerada em 62% quando utilizado o compilador HP e o processador com 2 linhas e em 77% nos processadores com 4 e 8 linhas, mas não é capaz de escalar quando o processador é alterado de 4 para 8 linhas. O tradutor binário apresenta comportamento similar, entretanto, alcançando a mesma aceleração para as 3 configurações, atingindo aceleração de 29% frente ao processador MIPS. Isto ocorre devido à grande quantidade de repetidas instruções *load* e *store* proveniente do compilador MIPS, já que uma *string* é lida durante a execução e resultados são salvos em pilha.

7.5 Considerações e Conclusões

O processador MIPS utiliza de forma acentuada instruções de *load* e *store*, devido à reduzida quantidade de registradores disponíveis frente ao processador VLIW utilizado, levando ao intenso uso da pilha de memória. Estas instruções limitam a aceleração a partir do tradutor proposto, principalmente porque este não aplica técnicas de renomeação de registradores, reduzindo as possibilidades de ocupar as unidades funcionais disponíveis.

Apesar destas limitações, as Figuras 7.2, 7.3 e 7.4 apresentam resultados mostrando que é possível manter a compatibilidade binária, acelerando estes códigos além de possibilitar a utilização de forma eficiente dos processadores reconfiguráveis.

8 CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho abrange a utilização da reconfigurabilidade total dos sistemas heterogêneos multiprocessados, propondo que processadores, memórias cache e comunicação sejam ajustados de acordo com a exigência da aplicação.

Cumprindo estas metas de reconfigurabilidade, foi possível alcançar relações ótimas entre desempenho e consumo de energia com reduzido dano ao desempenho, e em alguns casos sem qualquer dano ao desempenho. Somado à economia de energia frente aos elementos estáticos, foi possível atingir metas de aceleração dos códigos inicialmente escritos para determinado processador.

Para auxiliar na reconfigurabilidade destes sistemas, mantendo compatibilidade binária com arquiteturas já estabelecidas, um tradutor binário constitui uma ferramenta que ao mesmo tempo em que traduz da arquitetura fonte para outra capaz de acelerar o código original, determina a melhor configuração para este processador acelerador.

O Capítulo 6 mostra que a reconfigurabilidade de todos os componentes do sistema baseado nas necessidades da aplicação, traz benefícios claros quanto ao uso de menos hardware de forma simultânea, possibilitando redução no consumo de energia e potência, ao mesmo tempo em que acelera a aplicação. O capítulo 6 mostra ainda que as 7 aplicações experimentadas se mostram exigentes quanto ao hardware ideal para executá-las, provando necessária a reconfigurabilidade personalizada com fins de economia de energia ótima.

Além da reconfigurabilidade, no Capítulo 7 são apresentados resultados utilizando a ferramenta baseada em um tradutor binário que visa manter a compatibilidade binária entre os elementos de processamento aceleradores e o elemento mestre. Esta compatibilidade é importante, pois provê produtividade quando do uso de sistemas heterogêneos reconfiguráveis. Ainda no Capítulo 7 é possível observar que o uso do tradutor perde em desempenho comparado ao compilador original do acelerador, entretanto, é capaz de acelerar o código fonte, na maioria das aplicações apresentadas, utilizando menor quantidade de memória de instruções quando comparado ao compilador. A redução do tamanho da memória de instruções representa menor área ativa por parte destas memórias, por consequência menos potência dissipada, podendo resultar em diferentes valores de consumo energético.

Os resultados alcançados neste trabalho possibilitam a abrangência dos estudos apresentados, onde distribuição ideal da área de hardware ativa do sistema, análises de

simultaneidade de aplicações em execução, diferentes roteamentos das informações trafegadas, e outros problemas podem ser abordados dando sequência aos estudos aqui apresentados quando diz respeito a densidade de potência.

Diversas outras abordagens complementariam este trabalho, como o ajuste mais fino na reconfigurabilidade de todos os componentes descritos no Capítulo 5. Por exemplo, a possibilidade de ajustar-se o processador entre 2 e 8 linhas de execução em passos de 1 linha de execução, provendo processadores com 2, 3, 4, 5, 6, 7 e 8 linhas de execução. O mesmo pode ser pensado para as memórias cache, que poderiam variar entre 2KB e 32KB em passos de 2KB, por exemplo, fazendo com que se ajustasse de maneira mais fina à algumas aplicações. Estas modificações tornariam as verificações feitas neste trabalho mais precisas, capacitando um embasamento para uma modelagem da quantidade ideal de hardware em termos exatos para cada aplicação.

Adicionado à possibilidade dos processadores aceleradores serem configurados de forma mais fina, com número de linhas de execução mais preciso, a possibilidade de tais variações ocorrerem em tempo de execução. Isto quer dizer, variar o número de linhas de execução ativas durante a execução da aplicação, fazendo com que esta não execute sílabas NOP.

Quanto à ferramenta baseada no tradutor binário, outras possibilidades surgem capacitando um melhor aproveitamento dos aceleradores como: implantação de técnicas como renomeação de registradores, *software pipeline*, escalonamento de instruções mais efetivas, capazes de trazer soluções que aproximem ainda mais os resultados do tradutor binário aos resultados apresentados pelo compilador.

Finalmente, os resultados alcançados neste trabalho possibilitam concluir que a heterogeneidade é fator importante para que um sistema seja capaz de acelerar a execução de aplicações e reduzir o consumo de energia simultaneamente, quando códigos específicos são executados em hardwares específicos, alcançando o objetivo de redução de energia. Entretanto, além da heterogeneidade, a reconfigurabilidade dos elementos de processamento apresentados, abrangendo processadores, memórias e comunicação, capacitam de forma ainda mais contundente alcançar a relação ótima entre consumo de energia e desempenho, tornando-se fundamental para atingir tais objetivos.

REFERÊNCIAS

AGARWAL, A.; ISKANDER, C.; SHANKAR, R. "Survey of network-on-chip (NoC) architectures and contributions." In: JOURNAL OF ENGINEERING, COMPUTING AND ARCHITECTURE, 2009, [S.l.]: Proceedings Scientific Journals, 2005 v.3, p. 13-27.

AHMAD, B.; ARSLAN, T. "Dynamically reconfigurable NoC for reconfigurable MPSoC in Custom Integrated Circuits." In: CUSTON INTEGRATED CIRCUITS CONFERENCE, 2005. San Jose, CA: [s.n.] 2005. p. 277 - 285.

ALBONESI, D. H. "Selective Cache Ways: On-Demand Cache Resource Allocation." In: JOURNAL OF INSTRUCTION LEVEL PARALLELISM [S.l.]: [s.n.]. Maio, 2000.

ANJAM, F.; NADEEM, M.; WONG, S. "Targeting code diversity with run-time adjustable issue-slots in a chip multiprocessor." In: DESIGN, AUTOMATION & TEST IN EUROPE CONFERENCE & EXHIBITION (DATE), 2011, [S.l.]: [s.n.], p.1,6, Março 2011

Apple Watch, 2014. Disponivel em: <https://www.apple.com/watch/>. Acesso em: set. 2014.

ARM AMBA, 2013. Disponivel em: <http://www.arm.com/products/system-ip/amba/amba-open-specifications.php>. Acesso em: 10 dez. 2013.

ARM Big/Little, 2013. Disponivel em: <http://www.arm.com/files/pdf/big_LITTLE_Technology_the_Futue_of_Mobile.pdf>. Acesso em: 2013.

ARTERIS. The Network-on-Chip Company: A comparison of Network-on-Chip and Busses. **Arteris**, 2005. Disponivel em: <http://www.design-reuse.com/articles/10496/a-comparison-of-network-on-chip-and-busses.html>. Acesso em: out. 2013.

ATITALLAH, R. B. et al. "Estimating Energy Consumption for an MPSoC Architectural Exploration." In: ARCHITECTURE OF COMPUTER SYSTEMS - ARCS, 2006. [S.l.]: [s.n.], p. 298-310, 2006

BARAT, F.; LAUWEREINS, R. "Reconfigurable Instruction Set Processors: A Survey." In: 11TH INTERNATIONAL WORKSHOP ON RAPID SYSTEM PROTOTYPING. [S.l.]: [s.n.]. p. 168-173., 2000

BENINI, L.; DE MICHELI, G. "Networks on Chips: a New SoC Paradigm." In: COMPUTER. [S.l.]: [s.n.]. vol.35, no.1. p. 70-78, Jan 2002

BENITEZ, D. et al. "A Reconfigurable Cache Memory with Heterogeneous Banks." In : DESIGN, AUTOMATION & TEST IN EUROPE CONFERENCE & EXHIBITION (DATE). Dresden: [s.n.]. 2010. p. 825-830.

BHOJWANI, P.; MAHAPATRA, R. "Interfacing cores with on-chip packet-switched networks." In: INTERNATIONAL IEEE CONFERENCE ON VLSI DESIGN, 2003. [S.l.]: [s.n.]. p. 382-387.

BINKERT, N.; BECKMANN, B. The GEM5 simulator. ACM SIGARCH COMPUTER ARCHITECTURE NEWS. [S.l.]: [s.n.]. May 2011, p.211-225.

BROST, V.; YAN, F.; PAINDAVOINE, M. "A Modular VLIW Processor." In: INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS (ISCAS'07). [S.l.]: [s.n.]. 2007. p. 3968-3971.

CACTI: An Integrated Cache and Memory Access Time,Cycle Time, Area, Leakage, and Dynamic Power Model. Disponivel em: <http://www.hpl.hp.com/research/cacti/>.

CHANG, E.; GARCIA-MOLINA, H. . "Effective Memory Use in a Media Server." In: VLDB CONFERENCE, Athens, [s.n.]: 1997. p.496-505.

DENNARD, R. H.; GAENSSLEN, F.H.; YU, HWA-NIEN; RIDEOUT, V.L.; BASSOUS, ERNEST; LEBLANC, ANDRE R., "Design of ion-implanted mosfet's with very small physical dimensions." In: IEEE JOURNAL OF SOLID-STATE CIRCUITS, 9. [S.l.]: [s.n.]. 1974.

ESMAEILZADEH, H.; BLEM, E.; AMANT, R.; SANKARALINGAM, K.; BURGER, D. "Dark silicon and the end of multicore scaling." In: 38TH ANNUAL INTERNATIONAL SYMPOSIUM ON COMPUTER ARCHITECTURE (ISCA), San Jose, California, [s.n.] Jun 2011.p. 365-376

FARABOSCHI, P.; BROWN, G.; FISHER, J.; DESOLL, G.; HOMEWOOD, F. “Lx: A Technology Platform for Customizable VLIW Embedded Processing”. 27TH ANNUAL INTERNATIONAL SYMPOSIUM OF COMPUTER ARCHITECTURE (ISCA), June 2000. p.203-213.

GARG, S.; MARCULESCU, D.; MARCULESCU, R.; OGRAS, U. "Technology-driven limits on DVFS controllability of multiple voltage-frequency island designs - A system-level perspective." DESIGN AUTOMATION CONFERENCE, 2009. DAC. [S.l.]: 46th ACM/IEEE, 26-31 Jul 2009, pp.818,821,

GIVARGIS, T.; VAHID, F. "Tuning of Cache Ways and Voltage for Low-Energy Embedded System Platforms." In: JOURNAL OF DESIGN AUTOMATION FOR EMBEDDED SYSTEMS, vol. 7, p. 35-51, 2002.

GOLDSTEIN, S. C.; SCHMIT, H.; BUDIU, M.; CADAMBI, S.; MOE, M.; TAYLOR, R.R., "PipeRench - A Reconfigurable Architecture and Compiler." In: COMPUTERS, vol 33, nº 4. [S.l.]: [s.n.]. 2000. p. 70-77.

GOULDING, N. et al. GreenDroid: A mobile application processor for a future Dark Silcon. In: HOTCHIPS, [S.l.]:[s.n.] 2010.

GUERRIER, P.; GREINER, A. "A Generic Architecture for On-Chip Packet-Switched Interconnections". In: DESIGN, AUTOMATION & TEST IN EUROPE CONFERENCE & EXHIBITION (DATE). [S.l.]: [s.n.]. 2000. p. 250-256.

GUPTA, N. http://vlsi-soc.blogspot.com.br/. **VLSI-SoC**, 2009.

HAUSER, J. R.; WAWRZYNEK, J.; "Garp -- A MIPS Processor with a Reconfigurable Coprocessor." In: 5TH ANNUAL IEEE SYMPOSIUM ON FIELD PROGRAMMABLE CUSTOM COMPUTING MACHINES. [S.l.]: [s.n.]. 1997. p. 12-21.

HENKEL, J. “Closing the SoC Design Gap”. In: COMPUTER, [S.l.]: [s.n.]. Set, 2003. p. 119-121.

HERBERT, S.; MARCULESCU, D. "Variation-aware dynamic voltage/frequency scaling." In: HIGH PERFORMANCE COMPUTER ARCHITECTURE (HPCA). [S.l.]: [s.n.]. 2009. p. 301-312.

HP LABS. http://www.hpl.hp.com/downloads/vex/. **VEX**, 2010.

HUANG, L.; QIANG, X. "Energy-efficient task allocation and scheduling for multi-mode MPSoCs under lifetime reliability constraint." In: DESIGN, AUTOMATION & TEST IN EUROPE CONFERENCE & EXHIBITION (DATE), [S.l.][s.n.],2010. p 1584-1589.

INTEL SCC 2011. Disponivel em: <https://communities.intel.com/servlet/Jive Servlet/previewBody/19269-102-1-22565/Using Intel`s-Single-Chip Cloud Computer -SCC- - Intel Mattson, Tutorial.pdf>. Acesso em: 2013.

INTEL. Intel Xeon PHI, 2013. Disponivel em: <http://ark.intel.com/pt-br/products/75800/Intel-Xeon-Phi-Coprocessor-7120X-16GB-1_238-GHz-61-core>. Acesso em: set. 2014.

ISELI, C.; SANCHEZ, E. "Beyond Superscalar Using FPGAs." In: INTERNATIONAL CONFERENCE ON COMPUTER DESIGN: VLSI IN COMPUTERS AND PROCESSORS. [S.l.]: [s.n.]. 1993. p. 486-490.

ITRS. International technology roadmap for semiconductors, 2013. Disponivel em: <http://www.itrs.net>. Acesso em: ago. 2014.

KANDEMIR, M; Yemliha, T.; MURALIDHARA, S.; SRIKANTAIAH, S.; IRWIN, M.J.; ZHNAG, Y.; "Cache topology aware computation mapping for multicores." In: CONFERENCE ON PROGRAMMING LANGUAGE DESIGN AND IMPLEMENTATION, Toronto, SIGPLAN. 2010. p.74-85.

KATHURIA, J.; AYOUBKHAN, M.; NOOR, A. "A Review of Clock Gating Techniques." In: MIT INTERNATIONAL JOURNAL OF ELECTRONICS AND COMUNICATION ENGINEERING, v. 1, n. 2, p. 106-114, August 2011.

KOESTER, M.; LUK, W.; BROWN, G. "A Hardware Compilation Flow for Instance-Specific VLIW Cores." In: 18TH INTERNATIONAL CONFERENCE ON FIELD PROGRAMMABLE LOGIC AND APPLICATIONS. [S.l.]: [s.n.]. 2008. p. 619 - 622.

KOTTKE, T.; STEININGER, A. "A Fail-Silent Reconfigurable Superscalar Processor." In: 13TH IEEE INTERNATIONAL SYMPOSIUM ON PACIFIC RIM DEPENDABLE COMPUTING. [S.l.]: [s.n.]. 2007. p. 232-239.

KUMAR, S. "A Network-on-Chip Architecture and Design Methodology". In: SYMPOSIUM ON VERY LARGE INTEGRATION SCALE, [S.l.]: [s.n.] 2002. p. 117-124.

KUMRE, L.; SOMKUWAR, A.; AGNIHOTRI, G. "Power Efficient Carry Propagate Adder." In: INTERNATIONAL JOURNAL OF VLSI DESIGN & COMMUNICATION, [S.l.]: [s.n.] v. 4, n. 3, p. 125-134, aug 2013.

LODI, A. et al. "A VLIW Processor with Reconfigurable Instruction Set for Embedded Applications". In: IEEE JOURNAL ON SOLID-STATE CIRCUITS, vol. 38, nº 11. [S.l.]: [s.n.]. p. 1876-1886. mar, 2003.

LOPEZ-SANCHEZ, O. et al. "Ultrasensitive photodetectors based on monolayer MoS2." In: NATURE NANOTECHNOLOGY. [S.l.]: [s.n.]. 2013. vol 8. p. 497-501.

LUNGU, A. et al. "Dynamic Power Gating with Quality Guarantees." In: INTERNATIONAL SYMPOSIUM ON LOW POWER ELECTRONICS AND DESIGN, San Francisco, CA: [s.n.] 2009. p1-6.

MALIK, A.; MOYER, B.; CERMAK, D. "A Low Power Unified Cache Architecture Providing Power and Performance Flexibility". INTERNATIONAL SUMPOSIUM ON LOW POWER ELECTRONICS AND DESIGN [S.l.]: [s.n.]. 2000. p. 254-259

MANDELLI, M.; OST, L.; CARARA, E. "Energy-aware Dynamic Task Mapping for NoC-based MPSoCs. In: INTERNATIONAL SYMPOSIUM ON CIRCUTS AND SYSTEMS (ISCAS), [S.l.]:[s.n.] May 2011. p. 1676-1679.

MATOS, D. "Interfaces Parametrizáveis para Aplicações Interconectadas por uma Rede-em-Chip." Dissertação (Mestrado). Universidade Federal do Rio Grande do Sul - Programa de Pós-graduação em Computação, Porto Alegre, 2010

MATOS, D.; CONCATTO, C.; KREUTZ, M.; KASTENSMIDT, F.; CARRO, L.; SUSIN, A. “Reconfigurable Routers for Low Power and High Performance”. In: VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, [S.l.]:[s.n.] 2011. p. 2045-2057.

MOORE, G. E. "Cramming more components onto integrated circuits." In: ELECTRONICS, [S.l.]: [s.n.]. vol.38. nº.8. Jun. 1965.

NAGENDRA, C.; IRWIN, M. J.; OWENS, R. M. "Area – Time –Power Tradeoffs in Parallel Adders." In: IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II - ANALOG AND DIGITAL SIGNAL PROCESSING, [S.l.]: [s.n.] vol.43 . nº 10. Oct. 1996.

NICOLAESCU, D; JI, X.; VEIDENBAUM, A.; NICOLAU, A.; GUPTA, R. "Compiler-directed Cache Line Size Adaptivity". In: INTELLIGENT MEMORY SYSTEMS (IMS). [S.l.]: [s.n.]. 2000.

NIYONKURU, A.; ZEIDLER, H. C. "Designing a Runtime Reconfigurable Processor for General Purpose Applications." In: RECONFIGURABLE ARCHITECTURES WORKSHOP -18TH INTERNATIONAL SYMPOSIUM ON PARALLEL AND DISTRIBUTED PROCESSING. [S.l.]: [s.n.]. 2004. p. 143-149.

NVIDIA Tegra K1 2014. Disponivel em: <http://www.nvidia.com.br/object/tegra-k1-processor-br.html>. Acesso em: maio 2014.

RABAEY,. J. M. "Digital Integrated Circuits: A Design Perspective." [S.l.]: Prentice Hall, 2005, 4º ed..

RAHMANI, A. M. et al. “An efficient VFI-based NoC architecture using Johnson-encoded Reconfigurable FIFOs”. In: NORCHIP, 2010. p.1-5.

RAZDAN, R.; SMITH, M. D. "A High-Performance Microarchitecture with Hardware-Programmable Funcional Units." In: 27TH ANNUAL INTERNATIONAL SYMPOSIUM ON MICROARCHITECTURE. [S.l.]: [s.n.]. 1994. p. 172-180.

REINMAN, G.; JOUPPI, N. P. "CACTI 2.0: An Integrated Cache Timing and Power Model". In: COMPAQ WESTERN RESEARCH LAB REPORT. [S.l.]: [s.n.]. vol.5, 1999.

RYU, K.; SHIN, E.; MOONEY, V. " A Comparison of Five Different Multiprocessor SoC." In: EUROMICRO SYMPOSIUM DIGITAL SYSTEMS DESIGN, 2001.p. 202-209.

SAMSUNG. Samsung Gear S, 2014. Disponivel em: <http://www.samsung.com/global/microsite/gears/index.html>. Acesso em: set. 2014.

SANTOS, P. C.; NAZAR, G.L.; CARRO, L.; ANJAM, F.; WONG, S. "Adapting Communication for Adaptable Processors: A Multi-Axis Reconfiguration Approach". In: INTERNATIONAL CONFERENCE ON RECONFIGURABLE COMPUTING AND FPGAS (RECONFIG), Cancun: [s.n.] . 2012. p.1-6.

SANTOS, P. C.; NAZAR, G.L.; ANJAM, F.; WONG, S.; MATOS, D.; CARRO, L.; "A Fully Dynamic Reconfigurable NoC-based MPSoC: The Advantages of Total Reconfiguration." In: 7TH HIPEAC-WORKSHOP ON RECONFIGURABLE COMPUTING. Berlin: [s.n.]. 2013, p. 232-237.

SEGARS, S. "Low Power Design Techniques for Microprocessors". In: INTERNATIONAL SOLID-STATE CIRCUITS CONFERENCE. [S.l.]: [s.n.]. 2001.

SENGUPTA, D.; SALEH, R. Power-Delay Metrics Revisited for 90nm CMOS Technology. In: SIXTH INTERNATIONAL SYMPOSIUM ON QUALITY OF ELECTRONIC DESIGN, (ISQED), [S.l.]: [s.n.] 2005. p. 291 - 296.

SHULAKER, M. M. et al. "Carbon Nanotube Computer." In: NATURE. [S.l.]: Nature. vol. 501. 2013. p. 526-530.

THOMPSON, K. "Regular expression search algorithm." In: COMMUN ACM, [S.l.]:[s.n.] vol.11 p.419-422. June 1968.

TI. OMAP5, 2013. Disponivel em: <http://www.ti.com/product/omap5432>. Acesso em: jul. 2014.

TILERA GX. **GX72**, 2012. Disponivel em: <http://www.tilera.com/products/processors/TILE-Gx_Family>. Acesso em: jul. 2014.

TRANSMETA. Crusoé, 2001. Disponivel em: <http://www.transmeta.com/echnology/architecture/code_morphing.html>. Acesso em: mar. 2014.

UMAP. "University of Michigan Executive System for the IBM 7090 Computer". MI: THE “UNIVERSITY OF MICHIGAN ASSEMBLY PROGRAM (UMAP), Ann Arbor: [s.n.] v. 2, 1966.

VEIDENBAUM, A.; TANG, W.; GUPTA, R.; NICOLAU, A.; JI, X. "Adapting Cache Line Size to Application Behavior". In: 13TH INTERNATIONAL CONFERENCE ON SUPERCOMPUTING (ICS). Rhodes, Grécia: [s.n.]. 1999. p.145-154

WITCHEL, E.; ASANNOVIC, K. "The Span Cache: Software Controlled Tag Checks and Cache Line Size". In: 28TH INTERNATIONAL SYMPOSIUM ON COMPUTER ARCHITECTURE (ISCA). Sweden: [s.n.]. 2001, p.1-12

WONG, S.; VAN AS, T.; BROWN, G. "ρ-VEX: A reconfigurable and extensible softcore VLIW processor." In: INTERNATIONAL CONFERENCE ON ICECE TECHNOLOGY - FPT, Taipei: [s.n.] 2008. p.369-372.

ZEFERINO, C. A. "Redes-em-Chip: Arquiteturas e Modelos para Avaliação de Área e Desempenho." Porto Alegre: Tese (doutorado) - Universidade Federal do Rio Grande do Sul - Programa de Pós-graduação em Computação, Porto Alegre, 2003.

ZEFERINO, C.; KREUTZ, M.; SUSIN, A. “RASoC: A router soft-core for networks-on-chip”. In: DESIGN, AUTOMATION & TEST IN EUROPE CONFERENCE & EXHIBITION (DATE), [S.l.]: [s.n.] 2004. p.198–203.

ZHANG, C.; VAHID, F.; NAJJAR, W. "A Highly Configurable Cache Architecture for Embedded Systems". In: INTERNATIONAL SYMPOSIUM ON COMPUTER ARCHITECTURE (ISCA) [S.l.]: [s.n.]. 2003. p. 136-146.

ZHANG, C.; VAHID, F.; NAJJAR, W. "Energy Benefits of a Configurable Line Size Cache for Embedded Systems". In: INTERNATIONAL SYMPOSIUM ON VLSI. [S.l.]: [s.n.]. 2003. p. 87-91.