# Mecanismo de Otimização de Consumo de Energia e Desempenho baseado na Variação de Parâmetro da Memória Cache e do Processador NIOSII

Abel Guilhermino Silva Filho[1], Sidney Marlon Lopes de Lima[2]
*Universidade Federal de Pernambuco (UFPE) – Centro de Informática[1]*
*Universidade de Pernambuco (UPE) – Departamento de Sistemas e Computação[2]*
*agsf@cin.ufpe.br , smll@dsc.upe.br*

## Resumo

*O consumo de energia de uma hierarquia de memória cache pode atingir cerca de 50% de um sistema microprocessado[1]. Este projeto propõe: (i) um fluxo para estimar consumo de energia e desempenho computacional usando uma sistema SOC baseado em FPGAs, e (ii) um mecanismo de exploração de arquitetura com base na variação dos parâmetros da memória cache, utilizando o microprocessador NIOSII. Resultados, baseados nos benchmarks Mibench e Xirisc demonstraram que, em média, com exploração de 10% do espaço de busca, uma redução do consumo de energia de 28% pode ser alcançada, além de um aumento de 7% no desempenho para as 5 aplicações avaliadas. Adicionalmente, observou-se que foram encontrados resultados ótimos em 60% dos casos analisados.*

## 1. Introdução

Pesquisas indicam que 80% do total de processadores existentes no mercado são sistemas embarcados [18], que podem contemplar soluções baseadas em microcontroladores, DSPs, FPGA, dentre outros, inclusive misturas entre estas soluções, como são os casos dos SoCs.

Nos últimos anos, FPGAs estão sendo cada vez mais utilizados em ambientes computacionais embarcados para aumentar o desempenho do sistema[11]. FPGAs têm apresentado um grande desenvolvimento nos quesitos de densidade, velocidade e capacidade de armazenamento. Essas características tornaram possível a construção de sistemas complexos formados por um ou mais processadores (*soft-core*) e hierarquia de memória. Já se sabe, no entanto, que com o crescimento das funcionalidades em FPGAs, acompanhado do aumento do nível de sua complexidade, mais portas lógicas são necessárias, implicando em um aumento na energia dissipada pelo sistema[3][4].

Energia dissipada pode ser expelida para o ambiente em forma de calor. Excessivo calor pode diminuir a confiabilidade do sistema, afetar o encapsulamento dos circuitos e reduzir o tempo de vida útil da plataforma[5]. Está claro, portanto, que energia dissipada é um item essencial a ser analisado em sistemas embarcados. Medidas, no entanto, devem ser tomadas para não haver a degradação do desempenho do sistema.

Atualmente a energia consumida por uma hierarquia de memória pode atingir até 50% do total da energia gasta por um sistema [1]. Isto tem guiado muitos pesquisadores e projetistas de processador a analisar e entender a relação entre as várias configurações, estruturas de memória e o consumo de energia envolvido. A composição da memória pode conter memória principal e memória auxiliar tal como memória cache. Trabalhos relatam que o ajuste dos parâmetros de uma memória cache, para uma aplicação específica, pode economizar em média 60% do consumo de energia do sistema [20] e um aumento médio no desempenho de cerca de 30% [2].

Encontrar uma configuração adequada (tamanho total da cache, tamanho da linha e associatividade) para uma aplicação específica não é uma tarefa fácil e pode levar um tempo computacional muito elevado para análise e simulação. Por esta razão, a exploração exaustiva de todas as possíveis configurações de cache no espaço de projeto não é uma solução adequada. Com isso, estratégias para explorar o espaço de projeto de configurações devem ser adotadas para reduzir o consumo de energia sem degradação do desempenho e com custo computacional reduzido.

O uso de heurísticas, associadas com um ambiente capaz de analisar o comportamento da arquitetura pode alcançar resultados que reduzem o espaço de exploração da aplicação. Neste trabalho, está sendo proposto (i) um fluxo para estimar o consumo de

energia e desempenho com base em tecnologia FPGAs e (ii) um mecanismo de exploração de arquitetura OMCF-FPGA (*Optimization Mechanism based on Cost Function and FPGA*) com base na variação de parâmetros de uma hierarquia de memória. O processador utilizado pela pesquisa foi o NIOSII [17], um *soft processor* baseado na arquitetura RISC.

## 2. Trabalhos Relacionados

Simulações em nível RTL permitem a extração de informações sobre consumo de energia e desempenho com precisão satisfatória. Maior precisão, no entanto, não implica necessariamente rapidez na obtenção dos resultados das análises. Simulações em nível de sistema, por exemplo, gastam um tempo consideravelmente menor quando comparado às análises em nível RTL. Esta é uma das razões para que grande parte dos mecanismos de exploração de arquitetura sejam propostos no nível de sistema, a exemplo dos trabalhos de Gordon-Ross[2] e Silva-Filho[19].

Silva-Filho et al [19] propõe a heurística denominada TECH-CYCLES, que explorando cerca de 2% do espaço de busca total reduz o consumo de energia, em média 41%, e aumenta o desempenho em cerca de 25% quando comparado com a heurística TCaT proposta por Gordon-Ross et al.[2]. Já a heurística TEMGA, também de Silva-Filho et al.[14] e baseada em algoritmos genéticos, apresentou bons resultados para cache de dados, obtendo uma redução média do consumo de energia em cerca de 15% quando comparada a heurística TCaT. Zhang et al. [15] apresentou uma heurística para ajuste de cache reconfigurável que busca parâmetros da cache visando reduzir consumo de energia. Ghosh et al.[16] por sua vez apresentou uma heurística que, através de um modelo analítico, determina uma configuração de cache com base nas restrições de desempenho do projetista.

Por outro lado, sistemas utilizando *soft-core*, em nível RTL, estão sendo desenvolvidos. Thomas Wicent [6] propôs duas arquiteturas. A primeira com núcleo de processador NIOS II e hierarquia de memória cache, já a segunda continha núcleo de processador MIPS adaptado e nomeado de Plasma CPU [7]. O estudo avaliou o impacto de células lógicas e bits de memória em um FPGA APEX. A CPU MIPS demanda 2861 células lógicas, enquanto a CPU NIOS necessita apenas de 1500. Além disso, o modelo de memória do MIPS usa 70% bits a mais de memória quando comparado ao modelo de memória do NIOS. Este foi um dos motivos que levou ao uso do NIOS para o fluxo proposto.

Ferramentas RTL, com suas limitações, possibilitam investigar parâmetros como consumo de energia e desempenho, baseadas no sistema alvo e nas características da plataforma. Já ferramentas em nível de sistema, usualmente são baseadas em modelos que podem ser analíticos, empíricos, probabilísticos entre outros. Isto pode levar a uma significativa imprecisão na analise dos resultados coletados.

Então para a obtenção de uma boa configuração de memória cache, um ambiente que associe a precisão das ferramentas RTL acrescido de um mecanismo de exploração que evite a busca exaustiva tornam-se necessários. Neste sentido, este é o foco principal deste trabalho, e a metodologia proposta que provê tais aspectos, será explicada nas próximas seções.

## 3. Metodologia Proposta

### 3.1. Fluxo para Estimativa de Energia e de Desempenho

O fluxo de projeto para realizar estimativa de consumo de energia e de desempenho da aplicação foi subdividido em quatro partes:

1. Construir a arquitetura de um sistema computacional.
2. Carregar a aplicação na memória RAM do sistema.
3. Avaliar o número de ciclos necessários à execução da aplicação.
4. Obter o total de energia consumida pelo sistema.

A figura 1 mostra em detalhes qual é a seqüência de passos deste fluxo. Na etapa 1, constrói-se a arquitetura do sistema, utilizando a ferramenta SOPC Builder [8]. Ela gera toda a arquitetura do sistema (será descrito na seção 3.2) em linguagem VHDL. Além disso, ela cria um arquivo *.ptf o qual contém todas as informações quanto a configuração dos componentes inerentes à arquitetura.

Na segunda parte, a aplicação (ex: Fir) é compilada, através da ferramenta NIOSII IDE [8]. Por fim, a arquitetura é compilada, neste momento com a aplicação carregada na memória RAM.

Na etapa 3, o ModelSim-Altera[8] é utilizado para obter o número de ciclos necessários à execução da aplicação. Ao término da execução da aplicação, é gerado um arquivo *.vcd. Este arquivo conterá informações quanto ao comportamento dos sinais pertencentes à arquitetura e servirá como parâmetro para a estimativa do consumo de energia.

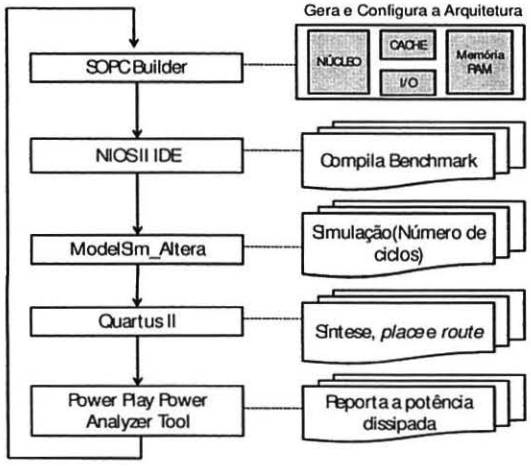


Figura 1: Fluxo proposto

O objetivo da etapa 4 é a obtenção do consumo de energia do sistema, mas antes disso é necessário a síntese e o posicionamento da arquitetura. Estas tarefas são realizadas pela ferramenta QuartusII [8]. A última ferramenta a ser utilizada pelo fluxo é o PowerPlay Power Analyzer Tool [8], ela tem como função analisar a potência dissipada pelo sistema. Para isto ela utiliza o arquivo *.vcd gerado na etapa 3. A potência dissipada associada ao número de ciclos gera o consumo de energia do sistema, através da equação (1).

$$E(joules) = P_{dis}(watts) \times T(seconds) \quad (1)$$

O fluxo é repetido tantas vezes quanto o mecanismo de exploração requisitar. Por isso, a importância em minimizar o espaço de exploração. Este mecanismo utiliza o consumo de energia e o número de ciclos, obtidos respectivamente nas etapas 4 e 3, como argumentos. Ele é explicado em detalhes nas seções 3.3 e 3.4. A seguir será informado o ambiente experimental usado neste trabalho.

## 3.2. Ambiente Experimental

A arquitetura sugerida neste trabalho é composta por um núcleo de processador NIOS II com freqüência de 50MHz, tensão de alimentação de 1.2 Volts, um nível de hierarquia de memória cache, contendo cache de instruções e cache de dados. A arquitetura possui uma memória principal RAM de 128KB. Além disso, são utilizados o JTAG UART: um periférico de entrada e saída, o *Performance Counter*: um contador de número de ciclos, o *System ID Peripheral:* um periférico o qual evita a permutação entre dispositivos e por fim o *Interval Timer:* um contador de tempo.

Apenas o tamanho da cache é utilizado como parâmetro configurável pela heurística a ser explicada na seção 3.4. Tanto a cache de instruções quanto a cache de dados podem variar de 512Bytes a 64Kbytes, incrementada em potência de 2, resultando um total de 8 diferentes tamanhos de cache. O espaço total de exploração é de 64 possíveis configurações da arquitetura sugerida. Os tamanhos da linha da cache de instruções e da cache de dados possuem respectivamente 32bytes e 4bytes, para todas as configurações. A associatividade é fixada em 1, tanto para a cache de dados quanto para cache de instruções.

O estudo proposto utiliza 5 diferentes aplicações extraídas dos benchmarks suítes: MiBench[9] e XiRisc[10]. A tabela 1 mostra o nome delas com os seus respectivos números de instruções executadas. Algumas mudanças foram necessárias: (i) reduzir o tamanho do conjunto de entradas, e (ii) eliminar os arquivos de entrada e saída, colocando-os diretamente no código fonte da aplicação. Ambas as medidas têm como intuito diminuir o tempo de simulação. Caso essas modificações não fossem empregadas uma única simulação poderia demorar dias, tornando inviável o projeto proposto.

Tabela 1: Avaliação das aplicações utilizadas

| Suíte | Benchmark | Modif. | Nº Instruções |
|---|---|---|---|
| MiBench | Stringsearch | R | 15912 |
| XiRisc | Bubble_Sort | E/S | 2890 |
| | Des | E/S | 2021 |
| | Fir | E/S | 2567 |
| | Iquant | E/S | 6230 |

R Reduzido o conjunto de entradas
E/S Arquivo(s) de entrada e saída eliminado(s)

A síntese realizada pelo QuartusII utilizou técnicas balanceadas de otimização para velocidade e área. Para calcular a potência dissipada foi usado o PowerPlay Power Analyzer Tool (ferramenta interna ao Quartus II). Na avaliação da potência dissipada, foram usados parâmetros *default* tais como: temperatura ambiente (25°C), dissipador térmico de 23mm com 200 LFpM de taxa de circulação de ar e *toggle rate* de 12,5%.

As arquiteturas considerando diferentes configurações de memória cache foram simuladas em uma máquina Athlon XP com 512MB de memória RAM. A família e o dispositivo alvo de todas as simulações foram *Cyclone III* e EP3C120F780C8 respectivamente.

### 3.3. Estudos Preliminares

A heurística de exploração OMFC-FPGA (*Optimization Mechanism based on Cost Function and FPGA*) consiste em obter a melhor configuração de hierarquia de memória cache com base na variação do tamanho da cache, em relação ao consumo de energia e número de ciclos. A OMCF-FPGA considera um nível de memória cache, dividida em cache de instruções e dados.

Como dito na seção 3.2 os possíveis tamanhos da memória cache variam em potência de dois, de 0.5kbytes a 64 kbytes. A configuração inicial, em negrito na figura 2, adotada pela heurística possui o tamanho da cache de instruções e dados com os seus valores mínimos (0.5kbytes). Tanto o tamanho da cache de instruções quanto da cache de dados varia do menor para o maior valor (64 kbytes).

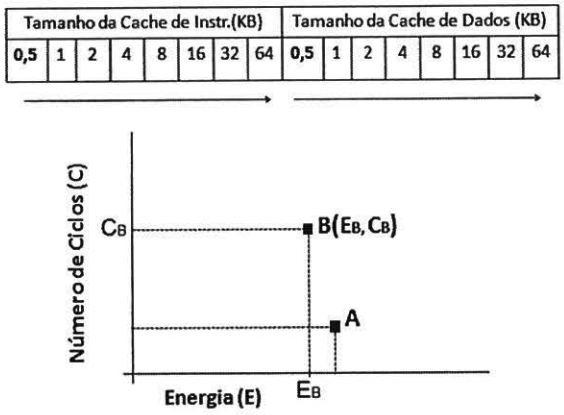


Figura 2: Escopo do espaço de busca da OMCF-FPGA

A figura 2 mostra também a forma como o espaço de busca é explorado. Assumindo que a primeira configuração avaliada pela heurística seja a configuração B ($E_B$, $C_B$). Os pontos $E_B$ e $C_B$ são os resultados obtidos pelo fluxo para estimativa do consumo de energia e número de ciclos respectivamente.

A heurística OMCF-FPGA aceita a busca de novas configurações que possuam função de custo (FC) menores do que a configuração B, de acordo com a seguinte equação (2). Em outras palavras, a heurística proposta visa minimizar a função de custo FC através da exploração do parâmetro tamanho da cache.

$$FC = E(mJ) \times Ciclos(10^3) \qquad (2)$$

Ao contrário de outras heurísticas, como a AEM-FPGA[12] que propõe um mecanismo que visa valores de energia e ciclos cada vez menores, sob pena de descartar determinadas configurações, a heurística que está sendo proposta, OMCF-FPGA, visa minimizar uma Função de Custo que também usa como argumentos o consumo de energia e o desempenho da aplicação. Neste sentido, possibilita que outras configurações sejam testadas viabilizando novas reduções no espaço de projeto e uma melhoria nos resultados obtidos.

Considere a configuração A. Esta configuração possui consumo de energia maior do que a configuração B. No entanto, a configuração A possui função de custo menor que a configuração B. A configuração A passa, então, a ser a melhor configuração examinada pela OMCF-FPGA. Se o mecanismo de exploração da heurística AEM-FPGA fosse considerada neste caso, a configuração A seria descartada tendo em vista que o valor de energia da configuração A é maior que o da configuração B. No entanto, observe que apesar do valor de energia da configuração A ser um pouco acima do valor de energia da configuração B, o número de ciclos da configuração A é muito menor que o da configuração B. Neste sentido, observamos que este diferencial proporcionava novas soluções próximas do Pareto[13] resultando em uma melhoria do mecanismo que será demonstrada através das aplicações que foram avaliadas nas próximas seções.

### 3.4. Mecanismo de Exploração OMCF-FPGA

O mecanismo de exploração proposto varia apenas o tamanho da cache. O fluxo de estimativa de energia e número de ciclos é realizado no nível RTL, onde o tempo de análise é consideravelmente maior quando comparado a análises no nível de sistema, no entanto, oferece resultados mais precisos nas análises. Em média uma simulação com o fluxo proposto demora cerca de 5 horas. Considerando o espaço de exploração total descrito na seção 3.2, seria necessário cerca de 1600 horas em uma busca exaustiva pela melhor configuração para as 5 aplicações descritas na tabela 1. Este fato tornaria o processo cansativo e demasiadamente lento. O mecanismo de exploração proposto avalia, em média, cerca de 9% do espaço de exploração total.

A figura 3 apresenta o algoritmo do mecanismo de exploração proposto neste trabalho. Cada configuração é composta por dois parâmetros, o tamanho da cache de instruções e o tamanho da cache de dados,

denominados respectivamente de TI e TD, além de três argumentos *FC*(Função de Custo), *Energia* e Número de *Ciclos*. A configuração inicial possui uma tupla com TI e TD iguais a 0.5 kbytes.

Durante a primeira iteração da heurística, todo o fluxo detalhado na seção 3.1 é executado. Após isto, as variáveis locais *FCmin, Emin e Cmin* recebem respectivamente os argumentos *FC*, *Energia* e *Ciclos* da configuração inicial.

A partir deste ponto, o parâmetro avaliado, neste momento TI, sofre variação, enquanto que o outro parâmetro, TD, é mantido fixo. Da segunda iteração em diante, caso esta nova configuração obtenha um argumento FC menor do que aquele que está armazenado na variável local *FCmin*, todas as variáveis locais são atualizadas com os argumentos da configuração atual, e o processo se repete. Caso contrário, a configuração atual é descartada e retorna-se a configuração imediatamente anterior, fixando TI. Em seguida, é iniciada a avaliação de TD, até se encontrar a melhor configuração.

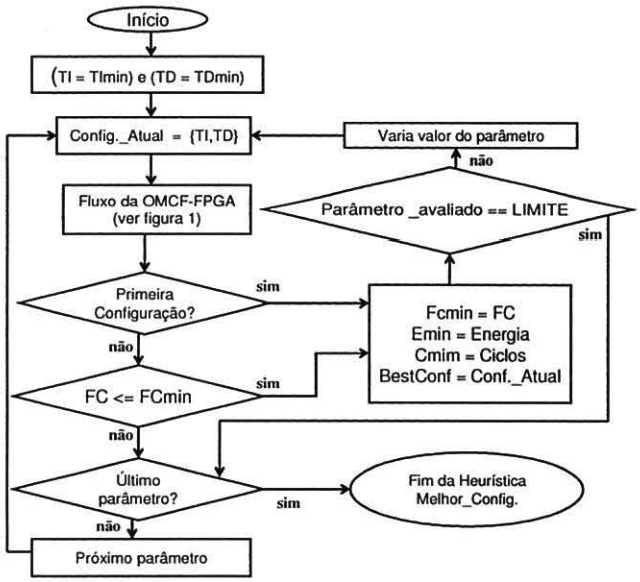


Figura 3: Algoritmo da heurística OMCF-FPGA

Caso a heurística alcance os valores limites dos parâmetros (TI=64 kbytes e TD=64 kbytes), ela finalizará. Após a conclusão, a melhor configuração encontrada através da heurística é obtida através das informações armazenadas na variável *BestConf* indicada na figura 3.

## 4. Resultados

Apesar da exploração exaustiva não ser necessária nesta pesquisa, ela foi realizada para demonstrar a eficiência da heurística OMCF-FPGA. Alguns *scripts* foram utilizados para automatizar a exploração total do espaço de busca correspondente a todas as possíveis configurações descritas na seção 3.2 envolvendo as 5 aplicações descritas na tabela 1.

O espaço de exploração total (busca exaustiva) da aplicação Iquant do benchmark Xirisc é detalhada na figura 4. Nesta figura, os resultados da heurística OMCF-FPGA foram adicionados para demonstrar a diferença entre eles para esta aplicação. Esta mesma figura relaciona a configuração de memória cache (eixo x), o número de ciclos (eixo y, à esquerda) e o consumo de energia (eixo y, à direta). A figura pode ser vista como uma junção de 8 conjuntos, com 8 configurações cada, atingindo um total de 64 configurações, correspondentes ao espaço de exploração total do projeto. Um dos conjuntos pode ser exemplificado através do retângulo na figura 4.

O consumo de energia e o número de ciclos estão dispostos na posição vertical na figura 4 com a sua respectiva configuração no eixo x. Por exemplo, a configuração com o tamanho da cache de instruções e dados ambos iguais a 512bytes (indicados na linha tracejada no gráfico da figura 4), possui consumo de energia igual a $E_P$ e número de ciclos igual a $C_P$.

A primeira configuração possui valor de 0,5kb para cache de instruções e 0,5kb para cache de dados. A segunda configuração possui valor de 0,5kb para cache de instruções e 1kb para a cache de dados. As próximas seis configurações possuem cache de instruções fixa em 0,5kb e a cache de dados continua sendo incrementada, em potencia de 2, até alcançar o valor de 64kb, encerrando o primeiro dos 8 conjuntos da figura.

No segundo conjunto, indicado na figura através do retângulo, a configuração inicial possui valor de 1kb para a cache de instruções e 0,5kb para a cache de dados. Assim como ocorreu no primeiro conjunto, a cache de instruções se mantém fixa e a cache de dados, desta vez, é incrementada em potência de 2 até atingir 64kb.

Esta forma de combinação se repete até ser atingida a configuração limite, correspondente a última configuração da oitava parte, que possui cache de instruções e dados ambas iguais a 64kb.

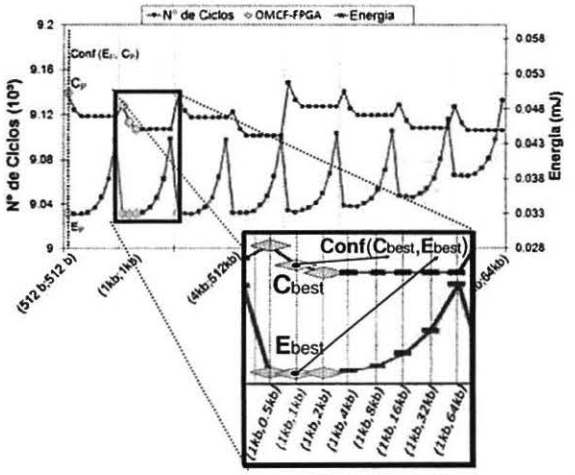


Figura 4: Consumo de Energia X Número de Ciclos X Configuração da Memória Cache(Instrução, Dado)

Os losangos representam quais foram os valores de consumo de energia e número de ciclos atingidos pela heurística proposta. Para melhor entendimento, é feito um zoom no segundo conjunto da figura 4, a parte mais povoada pela OMCF-FPGA. Neste zoom é mostrado 3 das 5 configurações que a heurística avaliou. O tamanho da cache de instruções é fixado em 1kb e o tamanho da cache de dados vai de 0.5kb a 64kb. A linha tracejada serve para interligar cada configuração com o seu respectivo consumo de energia e o número de ciclos.

A melhor configuração, de acordo com zoom da figura, foi de 1kb para a cache de instruções e 1kb para a cache de dados. Na figura o consumo de energia dessa configuração é denominado de $E_{best}$ e o número de ciclos de $C_{best}$. Analisando a figura 4, observa-se que a heurística, de uma maneira geral, avalia boas configurações, demonstrando a sua eficiência. A melhor configuração, para o *Iquant,* foi alcançada com apenas 5 simulações, isto corresponde a 8% do espaço de busca total. A redução do consumo de energia foi avaliado em relação à ***cache base*** de 0,5kbytes e 64kbytes para instruções e dados, respectivamente. Uma redução de cerca de 24% foi observada.

A mesma análise em termos de consumo de energia e desempenho das aplicações foi realizada para as demais aplicações com o intuito de validar a heurística OMCF-FPGA.

Os gráficos 5, 6, 7, 8 e 9 comparam os resultados da heurística proposta (OMCF-FPGA) com outra heurística existente, também baseada em FPGA (AEM-FPGA). Apesar de não ser necessário na abordagem proposta, os resultados exaustivos são incluídos nos gráficos para todas as aplicações avaliadas visando mensurar a eficácia do mecanismo de otimização proposto.

O gráfico da figura 5 indica resultados da heurística para a aplicação Bubble-sort do benchmark XiRisc. Embora a maioria das configurações obtidas não serem tão boas em termos de quantidade de ciclos, a heurística OMCF-FPGA obteve resultado ainda melhor que a AEM-FPGA tanto em termos de energia quando de quantidade de ciclos da aplicação.

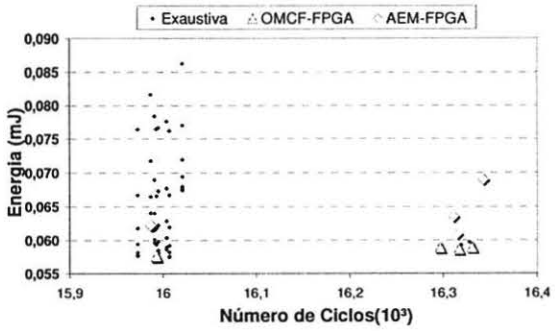


Figura 5: Bubble_Sort: Consumo de Energia X Número de Ciclos

Podemos observar que no gráfico da figura 6 para a aplicação Des do XiRisc, a heurística OMCF-FPGA obteve resultados próximos do Pareto, mostrando assim a eficácia do mecanismo para tal aplicação.

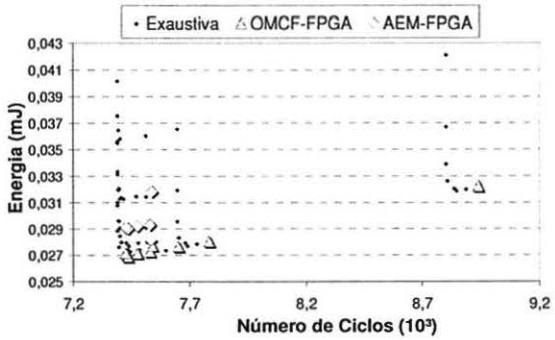


Figura 6: Des: Consumo de Energia X Número de Ciclos

Por outro lado, observa-se na figura 7 que resultados obtidos para a aplicação Fir do benchmark XiRisc, apesar dos bons resultados em termos de energia, a quantidade de ciclos necessário para execução da aplicação obtidos pela heurística OMCF-FPGA não foram tão bons quando comparados aos da heurística AEM-FPGA.

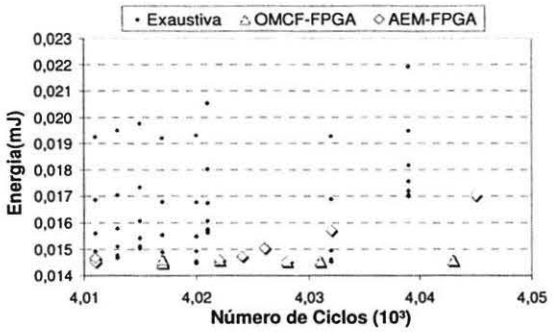


Figura 7: Fir: Consumo de Energia X Número de Ciclos

As figuras 8 e 9 mostram que a heurística proposta obteve bons resultados em termos de energia e quantidade de ciclos para as aplicações Iquant e StringSearch dos benchmarks XiRisc e Mibench respectivamente.

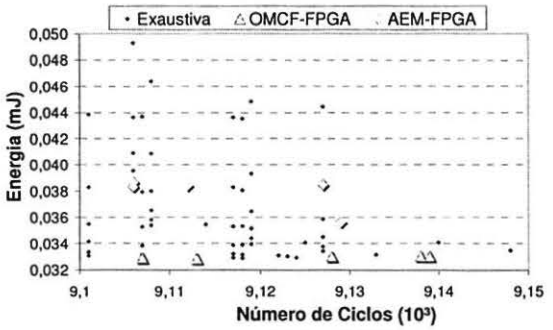


Figura 8: Iquant: Consumo de Energia X Número de Ciclos

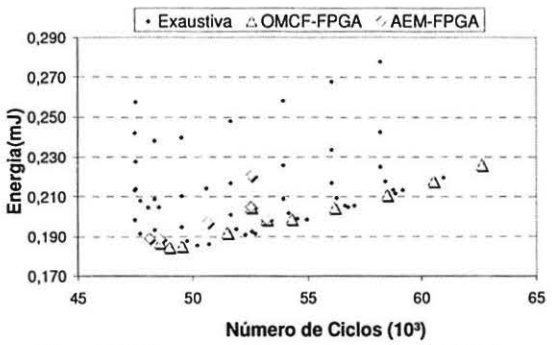


Figura 9: Stringsearch: Consumo de Energia X Ciclos

O impacto do número de simulações necessárias para concluir a execução da heurística proposta também foi avaliado. Foi observado que, em média, são necessárias apenas 7 simulações para concluir a heurística OMCF-FPGA. Quando comparado a heurística AEM-FPGA, resultados indicam que o valor médio praticamente permanece inalterado sendo necessário cerca de 10% do espaço de exploração completo.

O impacto de redução de energia foi avaliado baseado em uma ***cache base*** de 512b e 64kb para caches de instrução e dados respectivamente. Observou-se que houve um percentual de redução em torno de 28% como indicado na figura 10. Além disso, também foi avaliado o desempenho das aplicações em termos de quantidade de ciclos da aplicação e observou-se um aumento singelo de 7% no desempenho da aplicação quando considerando a mesma cache base.

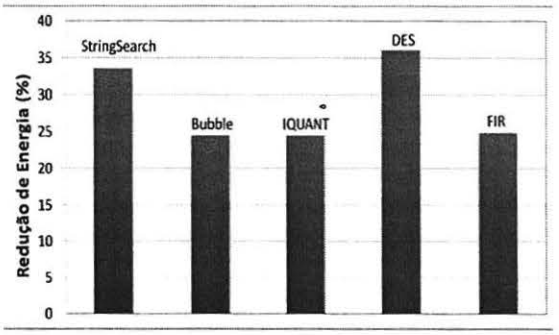


Figura 10: Comparação da porcentagem de redução do consumo de energia

Para comprovar a eficiência do mecanismo de exploração na redução de energia, os resultados obtidos foram comparados com a heurística AEM-FPGA e com os valores ótimos. A tabela 2 mostra os menores valores de consumo de energia dentre todas as configurações no espaço de busca selecionado para cada aplicação. Resultados mostrados indicam que o mecanismo OMCF-FPGA foi melhor em 100% dos casos quando comparado com a heurística AEM-FPGA em termos de consumo de energia. Além disso, os resultados na tabela 2 indicam que em 60% dos casos analisados a heurística OMCF-FPGA alcançou valores ótimos.

Tabela 2: Comparação entre AEM-FPGA, OMCF-FPGA e valores ótimos

| Aplicação | AEM-FPGA | OMCF-FPGA | *Optimal* (Exaustiva) |
|---|---|---|---|
| Stringsearch | 0.18878 | **0.18455** | 0.18455 |
| Bubble_Sort | 0.06215 | 0.05755 | 0.05749 |
| Iquant | 0.03832 | **0.03286** | 0.03286 |
| Des | 0.02904 | **0.02691** | 0.02691 |
| Fir | 0.01452 | 0.01449 | 0.01447 |

## 5. Conclusão

Neste trabalho, foram apresentados: um fluxo para realizar estimativa de energia com base em FPGAs, bem como um mecanismo de otimização (OMCF-FPGA) que avalia, em média, cerca de 10% do espaço de exploração. Cinco aplicações foram analisadas e uma redução do consumo de energia de cerca de 28% foi observada quando comparada a uma hierarquia de memória tradicional, além de um aumento no desempenho de cerca de 7%.

Se o projetista entender o comportamento da cache, pode então, criar mecanismos para se obter configurações próximas ao *pareto optimal* [13] sem a necessidade de avaliar um espaço de projeto muito grande. Neste trabalho foram variados apenas dois parâmetros da memória cache, e como observado, em média foram necessários 7 simulações para concluir a execução da heurística proposta. Esta abordagem, apesar de demandar tempos computacionais mais elevados, proporcionam resultados mais precisos nas análises e observou-se que em 60% dos casos analisados a heurística proposta atingiu os resultados ótimos. Com isso, estratégias de exploração, como a proposta neste trabalho, tornam-se necessárias, visando reduzir o tempo computacional nas análises.

## 6. Agradecimentos

Nossos agradecimentos à FACEPE pelo suporte financeiro fornecido ao desenvolvimento deste trabalho (APQ-0455-1.03/06).

## 7. Referências

[1] S. Segars, "Low Power Design Techniques for Microprocessors", ISSCC Tutorial note, Fevereiro 2001.

[2] Gordon-Ross, Ann, Vahid, F., Dutt, Nikil, "Automatic Tuning of Two-Level Caches to Embedded Applications". DATE, pp.. 208-213; Fevereiro 2004.

[3] George V. and Rabaey J. "Low-Energy FPGAs:Architecture and Design". MA; Kluwer, 2001.

[4] Shang L., Kaviani A., Bathala K. "Dynamic Power consumption in the Virtex-II FPGA family". In Proceedings ACM International Symposium Field-Programmable Gate-Arrays, pp. 157-164, 2002.

[5] Siozios K., Soudris D., Thanailakis A. "Efficient Power Management Strategy of FPGAs Using a Novel Placement Technique". Very Large Scale Integration, 2006 IFIP International Conference on, pp. 204-209. Outubro 2006.

[6] Wincent T. "The Design, Implementation and Evaluation of a MIPS IP-core for the Altera SOPC-Builder". Department of IMIT/LECS. Março 2005, pp.. 1 –22.

[7] Núcleo de processador Plasma CPU. Disponível em: http://www.opencores.org/projects.cgi/web/mips/overview. Acesso em 23 de junho de 2008.

[8] PowerPlay Power Analyzer Tool, Handbook Quartus II, SOPC Builder, NIOS II IDE, ModelSim-Altera. Disponível em: http://www.altera.com. Acesso em 18 de junho de 2008.

[9] Guttaus, M. R.; Ringenberg, J.S.; Ernst, D.; Austin, T.M.; Mudge, T.; Brown, R.B.; "Mibench: A free, commercially representative embedded benchmark suite". In IEEE 4th Annual Workshop on Workload Characterization, pp.1-12, Dezembro 2001.

[10]Suíte de Benchmarks XiRisc. Disponível em: http://www.micro.deis.unibo.it/~campi/XiRisc/. Acesso em 10 de Agosto de 2005.

[11] George V., Zhang H., Rabaey J. "The Design of a Low Energy FPGA" International Symposium on Low Power Electroncis and Design, pp. 188-193. Agosto 1999.

[12] Silva-Filho A.G., Lima S.L. "Mecanismo para Redução do Consumo de Energia e Ganho de Desempenho Através do Ajuste da Configuração de Caches Usando o Processador NIOS II", WSCAD-CTIC 2007. Outubro 2007

[13] Ascia, G.; Catania, V.; Palesi, M.; "An Evolutionary Approach for Pareto-optimal Configurations in SOC Platforms", In Kluwer Academic Publishers, editor, SOC Design Methodologies, pp. 157-168. Vol. 218. 2001.

[14] Silva-Filho, A.G.; Bastos-Filho, C.J.A.; Lima, R.M.F.; Falção, D.M.A.; Cordeiro, F.R.; Lima, M.P. "An Intelligent Mechanism to Explore a Two-Level Cache Hierarchy Considering Energy Consumption and Time Performance". pp. 174-184. SBAC-PAD 2007.

[15] Zhang, C., Vahid, F., Cache configuration exploration on prototyping platforms. 14th IEEE International Workshop on Rapid System Prototyping (June 2003), vol 00, p.164.

[16] Ghosh A., Givargis T. "Cache Optimization For Embedded Processor Cores: An Analytical Approach". ICCAD'03, pp. 342 – 347. Novembro 2003.

[17] Definição sobre o microprocessador NIOSII. Disponível em: www.altera.com/literature/lit nio2.jsp Acesso em 23 de junho de 2008.

[18] Tennenhouse D. "Proactive Computing". Comunications of The ACM. Volume 43, número 5, pp. 43-50. Maio 2000.

[19] A.G. Silva-Filho, F.R. Cordeiro, R.E. Sant'Anna and M.E. Lima. "Heuristic for Two-Level Cache Hierarchy Exploration Considering Energy Consumption and Performance", PATMOS, pp. 75-83, 2006.

[20] Ross A., Vahid F., Dutt N. "Fast Configurable-Cache Tuning with a Unified Second-Level Cache." IEEE/ACM International Symposium on Low Power Electronics and Design, pp.323-326. Agosto 2005.