HOLOS
ISSN 1807 - 1600

ACELERADORES RECONFIGURÁVEIS NO PROJETO *MULTICORE*: UMA ANÁLISE DE CUSTO *VERSUS* BENEFÍCIO

A. S. B. LOPES[1], M. M. PEREIRA [2]
Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande de Norte[1], Universidade Federal do Rio Grande do Norte[2]
ORCID ID: https://orcid.org/0000-0002-2016-055X[1]
alba.lopes@ifrn.edu.br[1]

Submetido 15/04/2020 - Aceito 27/08/2020

DOI: 10.15628/holos.2020.9924

RESUMO

A crescente evolução do *software* através de novas técnicas tem permitido o desenvolvimento de diversas soluções para atender a demanda da sociedade. Diferentes soluções de *hardware* têm surgido nos últimos anos para atender a essa demanda. Dentre essas soluções, os sistemas com múltiplos núcleos, chamados de *multicores*, estão entre as principais tendências. Porém, a busca por soluções de *hardware* não visa só o alto desempenho. É preciso levar em consideração outros aspectos como a eficiência energética e a área. Nesse cenário, a combinação de processadores com aceleradores reconfiguráveis tem sido amplamente explorada pelo fato destes últimos proporcionarem ganho de desempenho com redução do consumo de energia. Neste trabalho, pretendemos colaborar com o projeto de *multicores* investigando diferentes combinações de processadores com aceleradores reconfiguráveis. Como estudo de caso, combinamos processadores superescalares com arquiteturas reconfiguráveis de granularidade grossa e avaliamos três cenários. O primeiro é uma combinação de processadores e aceleradores que alcançam o mais alto desempenho possível para um conjunto de aplicações. O segundo é uma combinação de processadores e aceleradores definido por um limite de desempenho e o terceiro é limitado pelo consumo de energia. Os experimentos mostram que é possível obter uma aceleração de mais de 2,8x para determinadas aplicações; economizar mais de 11% de energia com perda de 10% de aceleração e reduzir 30% de área com economia de 20% no consumo de energia.

PALAVRAS-CHAVE: *multicore*, acelerador reconfigurável, exploração de espaço de projeto, desempenho, energia.

RECONFIGURABLE ACCELERATORS IN MULTICORE DESIGN: A TRADEOFF ANALYSIS

ABSTRACT

The continuous evolution of software through new techniques has allowed the development of several solutions. To keep up with the demand for high performance, several hardware solutions have emerged in recent years. Among these solutions, systems with multiple cores, called multicores, are among the main trends. However, the search for hardware solutions is not only about high performance. It is necessary to take into account other aspects such as energy efficiency and area. In this scenario, the combination of processors with reconfigurable accelerators has been widely explored for providing performance gains with reduced energy. In this work, we intend to collaborate with this topic by investigating different combinations of multicore design with reconfigurable accelerators. As a case study, we combined superscalar processors with coarse-grained reconfigurable architectures and evaluated three scenarios. The first is a combination of processors and accelerators that achieve the highest possible performance for a set of applications. The second is a combination of processors and accelerators defined by a performance limit and the third is limited by energy. The experiments show that it is possible to obtain an acceleration of more than 2.5x for some applications; save more than 11% energy with a 10% loss of acceleration and reduce 30% of area with 20% energy savings.

KEYWORDS: multicore, reconfigurable accelerator, design space exploration, performance, energy.

HOLOS
ISSN 1807 - 1600

1 INTRODUÇÃO

A evolução das aplicações através do surgimento e do aprimoramento das técnicas como aprendizado de máquina, computação de borda, 5G, 8k, dentre outros, tem gerado uma grande demanda por sistemas de computação com alto poder de processamento (Mao et al., 2017). De processadores usados em sistemas embarcados aplicados em robótica a supercomputadores usados para o processamento de aplicações computacionalmente intensivas, a busca tem sido por projetos de *hardware* que ofereçam não somente alto desempenho, mas que também apresentem eficiência energética, e atendam outros aspectos como restrições de tamanho físico (Butko et al., 2019).

Nesse cenário, sistemas heterogêneos que combinam diferentes processadores de propósito geral com aceleradores especializados se tornaram uma opção promissora (Cong et al., 2014). Esses processadores podem diferir entre si considerando um conjunto de características fundamentais, como as arquiteturas de conjunto de instruções (*Instruction Set Architectures* - ISAs) e a microarquitetura, por exemplo, em ordem e fora de ordem (Gao et al., 2015). Na indústria, essa abordagem já é difundida em arquiteturas que combinam tecnologias de CPUs (*Central Processing Units*) e GPUs (*Graphics Processing Units*) para aceleração gráfica e, mais recentemente, arquiteturas dedicadas para inteligência artificial e aprendizado de máquina (Neshatpour et al., 2018). No entanto, os benefícios potenciais de tais sistemas heterogêneos apresentam alta complexidade de projeto.

Devido a uma maior simplicidade de projeto, processadores *multicore* heterogêneos de uma única ISA surgem como uma alternativa atraente (Van Craeynest & Eeckhout, 2013). Esses processadores combinam núcleos, ou *cores*, com diferentes características microarquiteturais, mas compartilhando uma ISA comum. Essa organização ganhou popularidade no mercado de dispositivos móveis, onde a eficiência energética é uma questão muito importante. Os *chips Nvidia Tegra* (Nvidia, 2020) e *Samsung Exynos Octa* aproveitam a tecnologia *ARM big.LITTLE* (Samsung, 2020) e são exemplos do uso desse tipo de modelo. Além dos núcleos grandes e pequenos, a aceleração de *hardware* personalizado para algoritmos ou aplicativos específicos também se espalhou amplamente (Cong et al., 2014). Espera-se que esses aceleradores ofereçam ordens de magnitude em benefícios de desempenho e energia em comparação com soluções de uso geral. No entanto, o desenvolvimento de aceleradores de *hardware* específicos é oneroso tanto em tempo, quanto em mão de obra (Smit et al., 2002).

Uma solução candidata para lidar com esse desafio é o uso de arquiteturas reconfiguráveis (ARs) como aceleradores. As ARs são capazes de otimizar partes da aplicação (Li et al., 2017), pois são plataformas flexíveis devido à sua capacidade de reconfiguração e exploração de paralelismo (Compton & Hauck, 2008), podendo economizar energia e alcançar alto desempenho. Uma questão importante ao combinar processadores e ARs é se a combinação pode fornecer melhorias de desempenho e energia conforme o esperado em detrimento ao aumento no tamanho final do circuito (área). Responder à pergunta acima requer uma grande exploração do espaço de projeto, pois a carga de trabalho influencia diretamente a combinação de núcleos que devem ser usados.

HOLOS
ISSN 1807 - 1600

Com base nisso, neste trabalho, apresentamos três projetos de *multicore* compostos por processadores superescalares combinados com arquiteturas reconfiguráveis de granularidade grossa (*Coarse-Grained Reconfigurable Architectures* - CGRAs) para atender os aspectos de desempenho, energia e área. Os projetos foram gerados seguindo uma metodologia que executa vários experimentos em um simulador de microarquitetura que contém um processador superescalar fora de ordem acoplado a um CGRA. Como simulador de alto nível usamos *gem5* (Binkert, 2011) e, para resultados de área e energia, usamos as ferramentas McPAT (Li et al., 2009) e *Synopsys RTL* (Synopsys, 2020). Os experimentos demonstraram que os *multicores* otimizados atingiram uma aceleração de até 2,8x. Além disso, com uma sobrecarga de 10% em desempenho, foi possível economizar mais de 35% em área e 11% em consumo de energia.

Este artigo está organizado da seguinte maneira: seção 2 apresenta os trabalhos relacionados, abordando o projeto de *multicores* heterogêneo, envolvendo arquiteturas reconfiguráveis. Na seção 3 apresentamos nossa metodologia de exploração, incluindo as ferramentas utilizadas para realizar nossos experimentos. A Seção 4 apresenta os resultados dos experimentos considerando vários cenários de execução e os *multicores* heterogêneos gerados após o emprego da metodologia. Seção 5 apresenta as conclusões e discute os trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

Antes de discutir alguns dos trabalhos relacionados ao projeto desenvolvido, é necessário apresentar brevemente os principais conceitos abordados neste trabalho: *multicores* heterogêneos e aceleradores reconfiguráveis.

2.1 Referencial Teórico

Por muitos anos, os fabricantes de *hardware* replicaram as estruturas dos processadores para criar múltiplos caminhos de dados, permitindo que múltiplas instruções executassem de maneira concorrente. A duplicação de unidades aritméticas e unidades de ponto flutuante são exemplos dessa metodologia (Scogland et al., 2008). Os processadores *multicore* são o passo seguinte dessa replicação, onde duas ou mais unidades de execução independentes são integradas no mesmo circuito. Os *multicores* podem ser simétricos (homogêneos) ou assimétricos (heterogêneos). Nos *multicores* homogêneos, todos os núcleos têm a mesma arquitetura, tamanho, poder de processamento e consomem a mesma quantidade de energia. Já nos *multicores* heterogêneos, existem núcleos que são computacionalmente mais poderosos do que outros (Hill & Marty, 2008). Assim, tarefas que demandam maior desempenho podem executar nos núcleos com maior poder de processamento ao custo de maior consumo de energia. Já as demais tarefas podem executar nos núcleos com menor poder de processamento e menor consumo energético.

A heterogeneidade dos sistemas *multicore* pode ser classificada em funcional e de desempenho (Reddy et al., 2011). Na heterogeneidade funcional, os núcleos são muito diferentes funcionalmente, possuindo arquiteturas com ISAs distintas. Nesse modelo de heterogeneidade, a vantagem se dá pelos padrões específicos de execução, a fim de atender aos requisitos desempenho e consumo de energia. Já na heterogeneidade de desempenho, os núcleos possuem a mesma ISA,

mas com características de desempenho e consumo de energia diferentes. Exemplo dessa heterogeneidade pode ser encontrada nos processadores *ARM* denominados *big.LITTLE* (Greenhalgh, 2011). Essas arquiteturas combinam núcleos com maior poder de processamento (*big*) e núcleos menores, com um menor poder de processamento (*LITTLE*) que são projetados para reduzir o consumo energético. De acordo com ARM (2020), com esta solução, é possível se ajustar ao padrão de uso dinâmico dos dispositivos embarcados, como *smartphones*, *tablets*, dentre outros. Esse ajuste é possível porque a arquitetura *big.LITTLE* se ajusta a momentos de alto poder de processamento, como o requisitado em jogos em dispositivos móveis, alternadamente com longos períodos de baixa demanda por processamento, como em envio de mensagens, visualização de *e-mail* e áudio.

Em adição a núcleos grandes e pequenos em sistemas *multicore* heterogêneos, a aceleração em *hardware* em forma de caminhos de dados e circuitos de controle customizados para algoritmos específicos também se tornou vastamente utilizada (Cong et al., 2014). É esperado que esses aceleradores tragam benefícios em ordens de magnitude significativamente superiores quando comparados aos processadores de propósito geral. Entretanto, o desenvolvimento de aceleradores específicos em *hardware* é de alto custo em termos financeiros, de tempo, e de mão de obra. Uma forma de reduzir o impacto do custo dos aceleradores de *hardware* é através da adoção de arquiteturas reconfiguráveis. A tecnologia de reconfiguração de *hardware* é composta por um conjunto consolidado de metodologias, técnicas e ferramentas que têm como objetivo combinar a flexibilidade do *software* e a eficiência do *hardware* (Compton & Hauck, 2002).

Arquiteturas reconfiguráveis são classificadas de acordo com a sua granularidade. As arquiteturas de granularidade mais fina, como os FPGAs (*Field-Programmable Gate Arrays*), oferecem alta flexibilidade através de um vasto número de blocos lógicos. A forma de interconexão entre esses blocos pode ser configurada de forma a definir a funcionalidade que o *hardware* irá desempenhar. Considerando a grande quantidade de blocos lógicos existentes, o posicionamento e roteamento desses blocos é uma tarefa bastante complexa. Por outro lado, as CGRAs (*Coarse-Grained Reconfigurable Architecture*) trabalham em um nível de granularidade mais grossa. No lugar de blocos lógicos, essas arquiteturas possuem blocos reconfiguráveis similares às unidades lógico-aritméticas (ULAs) dos processadores. A maior vantagem destas últimas é apresentar uma considerável redução na complexidade necessária para realizar posicionamento e roteamento dos blocos reconfiguráveis, bem como redução no tamanho de memória e de tempo de configuração. Assim, por meio da customização do caminho de dados, as arquiteturas reconfiguráveis podem oferecer alto desempenho e redução no consumo de energia (Koeplinger et al., 2016).

A combinação de processadores de propósito geral com arquiteturas reconfiguráveis como aceleradores tem sido vastamente utilizada (Kuon et al., 2008; Tehre & Kshirsagar, 2012; Kareemullah, Janakiraman & Kumar, 2017; Nguyen, Le-Van & Tran, 2018). Nessa combinação, os processadores de propósito geral são responsáveis por controlar a lógica reconfigurável e executar o código do programa que não pode ser eficientemente acelerado (Compton & Hauck, 2002). Essa combinação também tem sido adotada em cenários *multicore*. E considerando que encontrar o número ideal dos elementos de processamento no projeto de uma arquitetura reconfigurável atrelado aos demais aspectos não é uma tarefa trivial (Suh et al., 2012), esse problema se intensifica

HOLOS ISSN 1807 - 1600

quando essas arquiteturas são inseridas nesse cenário de *multicores*, expandindo o tamanho do espaço de projeto a ser investigado.

2.2 Trabalhos relacionados

Na literatura é possível encontrar diferentes tipos de exploração de heterogeneidade em projetos *multicore*. Um tipo de arquitetura heterogênea consiste em integrar múltiplos núcleos homogêneos com aceleradores de *hardware* (Rossi et al., 2013; Dehyadegari et al., 2012). Outro tipo de configuração de arquitetura heterogênea consiste em núcleos de propósito geral que são assimétricos nas características de desempenho e potência (Cong et al., 2014; Kamdar & Kamdar, 2015; Mishra & Tripathi, 2014; Butko et al., 2019). Também é possível encontrar uma configuração de arquitetura heterogênea que consiste em núcleos e aceleradores assimétricos e especializados, incluindo arquiteturas reconfiguráveis (Koutras et al., 2017; Duhem et al., 2015; Souza et al., 2016). No caso dos *multicores* com arquiteturas reconfiguráveis é possível encontrar tantos projetos que utilizam FPGAs (Watkins & Albonesi, 2010; Bouthaina et al., 2013), quanto CGRAs (Souza et al.; 2016; Hussain, 2014), ou até mesmo ambas (Koenig et al., 2010).

Em (Butko et al., 2019), os autores apresentam um estudo do impacto da heterogeneidade na arquitetura *multicore* de ISA homogênea, aumentando o nível de heterogeneidade dos núcleos assimétricos. Neste trabalho, os autores variam a quantidade de grandes e pequenos núcleos no sistema e avaliam o impacto da inclusão de um núcleo intermediário, chamado de *middle*, na arquitetura. Já o projeto *multicore* proposto em (Cong et al., 2014) consiste em múltiplos núcleos grandes e pequenos, incluindo FPGAs. Com base nas prioridades, as aplicações podem ser escalonadas em núcleos de tamanhos diferentes. Os FPGAs podem ser usados para sintetizar aceleradores e permitir a execução em módulos que foram reconfigurados para computação específica de domínio.

O trabalho apresentado em (Koenig et al., 2010) apresenta uma arquitetura reconfigurável composta por diferentes modelos de processadores e interconexão. Um alto nível de heterogeneidade é explorado pelo sistema. Ele é composto de blocos reconfiguráveis grossos e finos, fortemente integrados, suportando vários tipos de arquitetura de conjunto de instruções (ISA). A arquitetura pode lidar com processadores RISC (*Reduced Instruction Set Computer*), VLIW (*Very Large Instruction Word*) e EPIC (*Explicitly Parallel Instruction Computer*). Ele também apresenta uma estrutura de software, fornecendo uma ferramenta de tempo de projeto e ferramentas de tempo de compilação. Depois de lidar com diferentes ISAs, é necessária uma etapa adicional para preparar a plataforma para execução.

Em (Souza et al., 2016), os autores propõem uma solução que permite criar arquiteturas *multicore* heterogêneas compostas por núcleos homogêneos acoplados à lógica reconfigurável de granularidade grossa. O sistema projetado é homogêneo na arquitetura, permitindo compatibilidade binária, e heterogêneo na organização. Segundo os autores, a heterogeneidade é feita acoplando a lógica reconfigurável com diferentes números de unidades funcionais para cada núcleo.

HOLOS
ISSN 1807 - 1600

O trabalho proposto nesse artigo contribui com a literatura ressaltando as divergências com os trabalhos relacionados nos aspectos listados a seguir. Em contraste com (Cong et al., 2014), nosso trabalho propõe o uso de arquiteturas reconfiguráveis de granularidade grossa com o objetivo de reduzir o tempo de reconfiguração. Em vez de avaliar o custo *versus* benefício entre processadores em ordem e fora de ordem como (Butko et al., 2019), nosso trabalho apresenta a melhor combinação de processadores superescalares fora de ordem e arquiteturas reconfiguráveis para otimizar um determinado requisito não funcional, por exemplo, desempenho e consumo de energia. Ao contrário de (Koenig et al., 2010), em nosso trabalho, exploramos o uso de uma ISA única. Portanto, nem modificações no código fonte nem bibliotecas adicionais são necessárias. Usamos tradução binária que lida com o processo de geração de configurações, e a heterogeneidade é completamente transparente para o programador ou o sistema operacional. Adicionalmente a (Souza et al., 2016), nosso trabalho expande a heterogeneidade explorada por este trabalho, que considera a lógica reconfigurável com número diferente de unidades funcionais para cada núcleo, também explorando e alterando as características microarquiteturais dos processadores superescalares.

Assim, neste trabalho, somos capazes de gerar três tipos de *multicores* heterogêneos otimizados. O primeiro tipo é uma combinação de superescalar e CGRA que alcança o mais alto desempenho possível para um conjunto de aplicações (*benchmarks*). O segundo tipo é uma combinação de superescalar e CGRA limitado por um critério de desempenho, e o terceiro tipo é limitado por um valor máximo de energia as ser consumida pelo sistema.

3 METODOLOGIA

Para realizar a exploração de espaço de projeto foi necessário definir os modelos arquiteturais a ser explorados; escolher as ferramentas para obtenção dos dados de desempenho, energia e área; e definir os *benchmarks* para realizar os experimentos. Em seguida, aplicar uma metodologia descrita a seguir e resumida na Figura 1:

1. Coleta dos resultados de desempenho dos *benchmarks* através do uso do simulador de alto nível *gem5* (Binkert, et al, 2011), que simula o caminho de dados de um processador superescalar;
2. Coleta dos resultados de desempenho dos mesmos *benchmarks* através do simulador *gem5*, simulando os processadores superescalares acoplados às CGRA*s*;
3. Medição do consumo de energia de cada um dos modelos arquiteturais para cada um dos *benchmarks* através do simulador *McPAT* (Li, et al, 2009);
4. Estimativa de área de cada processador superescalar e cada acelerador reconfigurável usando como base a ferramenta *Synopsys RTL* (Synopsys, 2020);
5. Proposta de sistemas *multicore* compostos pelas combinações de processadores superescalares e aceleradores do tipo CGRA que atendam aos requisitos do projetista.

HOLOS
ISSN 1807 - 1600

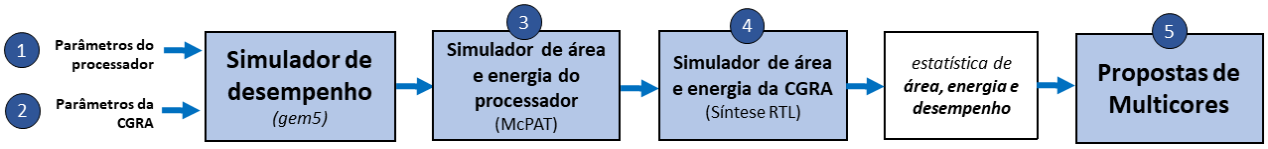


Figura 1: Metodologia adotada na exploração de espaço de projeto

Nas subseções a seguir, descreveremos as ferramentas que usamos para coletar desempenho, estimar a área e medir o consumo de energia dos processadores e o CGRA usado para realizar nossos experimentos.

3.1 Desempenho

Para obter resultados de desempenho, empregamos o simulador *gem5* (Binkert et al., 2011). O *gem5* é um simulador capaz de modelar uma ampla variedade de modelos arquiteturais. Ele fornece um conjunto de modelos de processadores, porém, a extensão desses modelos com novos recursos também é possível. Além disso, para obter resultados de desempenho do processador acoplado aos aceleradores, um projeto de arquitetura reconfigurável de granularidade grossa (CGRA), com base na proposta de (Brandalero & Beck, 2017), foi modelado no caminho de simulação *gem5*. Os processadores e a organização do CGRA usados neste trabalho são descritos a seguir.

Características do processador: Um processador superescalar implementa o *ILP* (*Instruction Level Parallelism*) executando mais de uma instrução durante um ciclo de relógio. Usando o *gem5*, podemos simular vários cenários de processadores variando um conjunto de parâmetros como a quantidade de execuções de instruções iniciadas por ciclo (*issue)*, tamanho do *buffer* de reordenamento (*Reorder Buffer* - RoB), tamanho da fila de leitura/escrita na memória, hierarquia de memória, tamanho da memória *cache* etc.

Arquitetura reconfigurável de granularidade grossa: CGRA*s* têm sido usadas como aceleradores para acelerar a execução de partes computacionalmente intensivas do código que possuem grandes quantidades de *ILP* (Hartenstein, 2001). As arquiteturas CGRAs fornecem várias conexões diretas entre as UF*s* que permitem que os dados sejam enviados de uma UF para outra sem a necessidade de armazenamento num banco de registradores (BR). Como resultado, há uma maior economia de energia pois uma menor quantidade de dados precisa ser transferida para dentro e para fora dos BRs. Em Brandalero (2019), são apresentados resultados que demonstram uma redução do consumo de energia de até 1,59x quando uma arquitetura reconfigurável é acoplada a um processador superescalar. O ganho mais expressivo ao usar o CGRA é na aceleração de laços. Outro fator importante para economizar energia ao usar CGRAs é que a arquitetura pode ser configurada no início de um laço e permanecer fixa durante toda a execução do laço.

A CGRA usada no estudo de caso explorado neste trabalho está fortemente acoplada ao *pipeline* do processador superescalar. O sistema, ilustrado na Figura 2(a), é composto pelo processador base, a *cache* de configuração, a CGRA e o módulo de tradução binária (TB).

HOLOS
ISSN 1807 - 1600

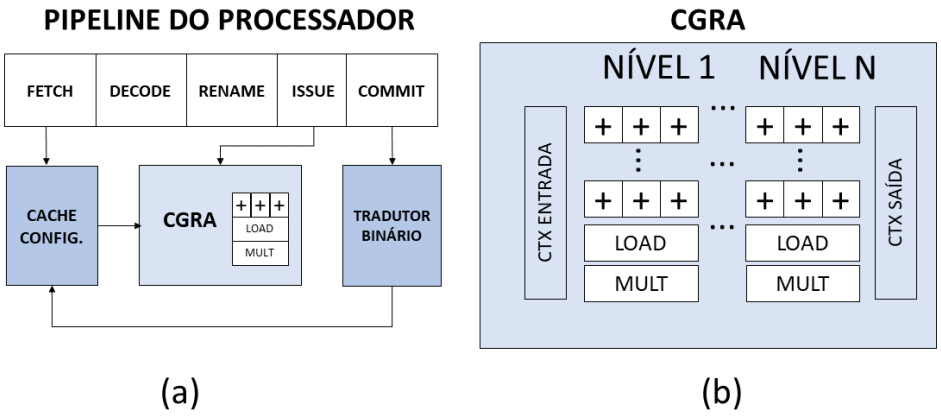


Figura 2: Modelo arquitetural utilizado no estudo. (a) processador superescalar com CGRA acoplada (b) detalhes arquiteturais da CGRA

O fluxo de execução do sistema é o seguinte: na primeira vez, a sequência de instruções é executada no *pipeline* do processador superescalar. Enquanto é executada, a sequência de instruções é transformada pelo módulo de tradução binária em configurações CGRA e armazenadas na *cache* de configuração. O processador superescalar continua executando as próximas instruções. A cada sequência de instruções, é feita uma busca na *cache* de configuração para verificar se há uma configuração que corresponda à instrução a ser executada. Caso seja encontrada uma correspondência, a configuração é carregada no CGRA, onde a execução continua da maneira mais otimizada.

Os detalhes da CGRA são mostrados na Figura 2(b). A lógica reconfigurável é dividida em linhas e colunas, compondo uma matriz de unidades funcionais (UFs). A UF pode ser uma unidade lógica/aritmética (ULA, representadas pelo + na Figura 2(b)), unidades de multiplicação (MULT*)* ou unidades de leitura/escrita (*LOAD*). Cada UF ocupa uma linha ou uma sequência de colunas de acordo com sua latência. Na implementação usada neste trabalho, as ULAs precisam de 1/3 de um ciclo do processador e correspondem a uma única coluna. Um nível é definido como um ciclo do processador. Instruções mapeadas para colunas podem ser executadas em paralelo. Por outro lado, instruções mapeadas em diferentes níveis são executadas em sequência.

3.2 Área e energia

Para obter resultados de área e consumo de energia, combinamos algumas ferramentas em nossa metodologia. Dividimos nossa explicação detalhando as ferramentas usadas para a estimativa de processadores superescalares e estimativas de CGRA.

Processadores superescalares: para obter resultados de área e energia dos processadores, empregamos o *McPAT*. *McPAT* é uma ferramenta que modela tempo, área e potência para permitir a exploração do espaço de projeto de diferentes processadores (Li et al., 2009). No *McPAT* é possível configurar a *issue* do processador, a profundidade do *pipeline*, o tamanho do *RoB*, o número de ULAs por núcleo, a frequência do sistema, o tamanho da *cache*, etc. Fornecendo esses parâmetros, o *McPAT* cria um modelo de *chip* interno, modelando e estimando a área de elementos arquiteturais. Para obter os resultados da área para o cenário estudado, configuramos o *McPAT* com parâmetros de microarquitetura de cada processador superescalar que pretendemos avaliar. Em

HOLOS ISSN 1807 - 1600

seguida, para estimar o consumo de energia, configuramos a ferramenta com as estatísticas de atividade dinâmica geradas por cada execução de *benchmark* no simulador de alto nível *gem5*.

Acelerador de *hardware* reconfigurável (CGRA): Como o *McPAT* não oferece suporte intrínseco aos aceleradores, obtivemos as áreas das ULAs, unidades de leitura/escrita e de multiplicadores e multiplexadores a partir da síntese RTL (*Register Transfer Level*), usando as ferramentas da *Synopsys* (Synopsys, 2020). Descrevemos a CGRA em termos de número de UFs e multiplexadores de cada tipo por cada CGRA que queremos investigar, a fim de obter a área total. Para a estimativa de energia da CGRA, usamos o McPAT para obter o pico de potência de unidades individuais de ALU e multiplicadores e usamos os resultados sintetizados estimar os componentes restantes.

Para estimar o tempo de ocupação e execução do CGRA, modificamos a implementação do *gem5* para extrair informações de mapeamento como: o número de configurações geradas para cada *benchmark*; quantas vezes cada configuração foi realizada no CGRA; quantas UFs são usadas para cada configuração; e quantos ciclos o CGRA gastou para executar cada configuração. Multiplicando os resultados individuais de potência de pico de cada UF por resultados extraídos do *gem5*, estimamos o consumo de energia para cada *benchmark* executado no CGRA acoplado ao processador superescalar.

4 RESULTADOS

Nesta seção descrevemos o estudo de caso seguindo a metodologia descrita acima. Em resumo, avaliamos 16 *benchmarks*, executando-os no simulador *gem5*, considerando 3 versões de superescalares: 2-*issue*, 4-*issue* e 8-*issue*. Em seguida, acoplamos esses processadores a três CGRAs diferentes que denominamos CGRA-P, CGRA-M e CGRA-G, de pequeno, médio e grande, respectivamente, e simulamos novamente no *gem5*, levando em consideração a aceleração fornecida pelo CGRA. Essas configurações foram escolhidas com base nos experimentos realizados em (Brandalero & Beck, 2017).

Selecionamos um subconjunto de *benchmarks* do pacote *MiBench* (Guthaus et al., 2001) que representam as cargas de trabalho típicas usadas em ambientes embarcados tais como compressão de imagem, decodificação de áudio e criptografia. Com base no número total de ciclos, discutimos os ganhos de aceleração obtidos pelos processadores superescalares e as versões em que o superescalar foi acoplado ao CGRA. Por fim, geramos três sistemas *multicores* otimizados, considerando os seguintes cenários: 1) a melhor combinação de processadores superescalares e CGRAs para oferecer o melhor desempenho; 2) a combinação de núcleos para proporcionar uma perda aceitável em 10% da aceleração, visando reduzir consumo de energia; e 3) a combinação mínima de núcleos para executar todos os *benchmarks* visando uma economia de 20% de energia.

4.1 Superescalar

Como mencionado anteriormente, em nossos experimentos, variamos a largura do *issue* do superescalar considerando o dois, quatro e oito *issues*. A Tabela 1 mostra detalhes de cada núcleo de processador usado neste estudo.

HOLOS ISSN 1807 - 1600

Tabela 1: Características microarquiteturais dos processadores superescalares

Issue	Características microarquiteturais
2-issue	Fora de ordem de largura 2; Fila de instruções: *54 ops*; *Buffer* de leitura: *72 ops*. *Buffer* de escrita: *42 ops*. Entradas no ROB: *128 ops*
4-issue	Fora de ordem de largura 2; Fila de instruções: *54 ops*; *Buffer* de leitura: *72 ops*. *Buffer* de escrita: *42 ops*. Entradas no ROB: *128 ops*
8-issue	Fora de ordem de largura 2; Fila de instruções: *54 ops*; *Buffer* de leitura: *72 ops*. *Buffer* de escrita: *42 ops*. Entradas no ROB: *128 ops*

Os experimentos foram realizados executando todos os 16 *benchmarks* nas três versões de superescalares. A Tabela 2 mostra o perfil dos *benchmarks* considerando a percentagem de instruções por tipo e o *µIPC* (Instruções por Ciclo) após executar cada *benchmark* em cada uma das versões do superescalar (2-*issue*, 4-*issue* e 8-*issue*). Observando a distribuição de instruções, *sha*, *bitcount* e *gsm-dec* possuem mais de 75% de instruções do tipo ULA. Por outro lado, *crc32*, *cjpeg*, *djpeg*, *gsm-enc*, *susan-e* e *susan-c* possuem mais de 15% de instruções de leitura da memória.

4.2 Superescalar + CGRA

O objetivo desta análise foi observar potencial de aceleração da CGRA acoplada aos processadores superescalares em comparação com as versões dos superescalares sem aceleradores descritos na Tabela 1. Para fornecer essa análise, parametrizamos diferentes versões da CGRA. Na CGRA é possível modificar o número de UFs em cada coluna, a quantidade de níveis, o tamanho da *cache* de configuração, o número de blocos básicos por configuração, entre outros parâmetros. Neste estudo, nomeamos as versões CGRA como arquiteturas pequena (P), média (M) e grande (G). Todas as versões possuem ULAs com latência de 13 ciclos, organizadas em 3 colunas por nível e 4 ULAs por coluna e cada configuração pode conter até 3 blocos básicos. A diferença entre CGRAs é a quantidade de níveis, detalhada na Tabela 3.

Tabela 2: Tempo de execução das aplicações nos processadores superescalares

Benchmark	**ULAs**	**Leitura**	**Escrita**	**Salto**	**µPC 2-issue**	**µPC 4-issue**	**µPC 8-issue**
basicmath	67,5%	12,7%	7,9%	11,8%	1,44	2,13	2,42
bitcount	83,6%	3,7%	0,9%	11,8%	0,93	1,34	1,41
cjpeg	63,6%	18,8%	9,0%	7,8%	1,40	1,74	1,84
crc32	61,2%	18,5%	8,7%	11,7%	1,69	2,82	3,05
dijkstra	71,3%	10,6%	5,8%	12,3%	1,52	2,19	2,72
djpeg	65,8%	18,6%	9,7%	4,6%	1,43	1,92	2,02
FFT	68,1%	12,8%	7,4%	11,6%	1,48	2,21	2,56
gsm-dec	77,4%	6,4%	3,5%	10,7%	1,55	2,52	2,72
gsm-enc	68,5%	15,6%	5,8%	3,6%	1,44	1,77	1,83
patricia	68,1%	12,1%	7,8%	11,8%	1,20	1,65	1,88
qsort	68,8%	12,8%	7,6%	10,9%	1,35	1,80	1,97
sha	81,0%	11,5%	4,3%	3,2%	1,22	1,59	1,62
stringsearch	67,7%	9,3%	11,1%	11,9%	1,03	1,30	1,36
susan-c	65,7%	22,3%	8,3%	3,6%	1,12	123	1,27

HOLOS ISSN 1807 - 1600

susan-e	69,0%	24,1%	3,7%	2,9%	1,03	1,10	1,11
susan-s	74,0%	13,0%	0,1%	4,5%	1,54	1,74	1,75

Tabela 3: Quantidade de unidades funcionais nos 3 tamanhos de CGRAs utilizadas.

	Níveis	ULA	MULT	LD/ST
CGRA-P	8	96	8	16
CGRA-M	16	192	16	32
CGRA-G	32	384	32	64

A execução de *benchmarks* no superescalar 2-*issue* foi usada como linha de base para comparações de desempenho. Foram realizadas 192 execuções que se referem à execução dos 16 *benchmarks* em cada um dos 12 núcleos estudados (2, 2P, 2M, 2G, 4, 4P, 4M, 4G, 8, 8P, 8M, 8G). A Figura 3 mostra a aceleração obtida considerando todos os 12 núcleos analisados e todas as 16 aplicações. Em todos os experimentos realizados, os núcleos CGRA podem acelerar a execução do *benchmark* quando comparado ao núcleo autônomo superescalar. Pode-se observar um padrão de crescimento do menor núcleo (2-*issue*) ao maior núcleo (8G), uma vez que o CGRA pode explorar mais ILP do que o superescalar. Há melhorias notáveis quando se considera o CGRA-P e o CGRA-M combinados com as três versões superescalares. O *benchmark sha*, por exemplo, alcançou uma aceleração de 1,69x quando executado no núcleo 2P, a menor versão de superescalar e de acelerador reconfigurável. A aceleração aumenta para 2,23x quando é executada no núcleo 8G. Conforme mostrado na Tabela 2, este parâmetro de referência possui um alto número de operações de ULA (81,01%) e baixa taxa de instruções por ciclo (1,22), indicando uma alta dependência entre essas operações. Outros *benchmarks* que seguem esse comportamento são *dijkstra*, *djpeg*, *fft* e *gsm-dec*.

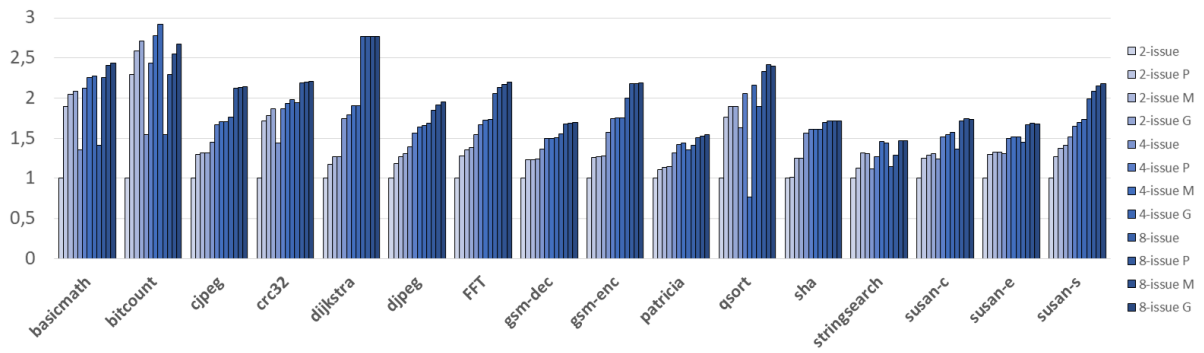


Figura 3: Comparação da aceleração de desempenho dos *benchmarks* executando em cada núcleo

Destacamos *bitcount* como a aplicação que mais se beneficia da CGRA. Esse *benchmark* tem uma baixa taxa de acesso à memória (3,7% do total) e o maior número de instruções de ULA. O CGRA pode reduzir significativamente o tempo de execução porque as UFs permitem o fluxo de dados de uma ULA para outra. Quando se observa as configurações das CGRAs geradas para esse *benchmark*, existem três configurações principais que cobrem mais de 50% do tempo de execução. Nas três configurações, uma média de 20 ULAs (quase 50% dos recursos de ULA da CGRA-P) foram ocupadas, levando apenas 3 ciclos de relógio para executar essas 20 instruções. Por outro lado, não

HOLOS
ISSN 1807 - 1600

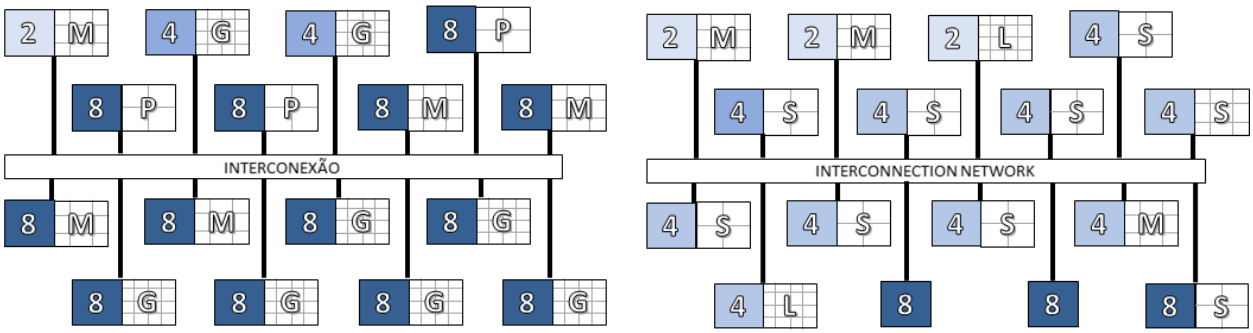
observamos melhora significativa da execução quando comparamos o núcleo 2P e o 8P, porque a quantidade de exploração de ILP desse *benchmark* já foi alcançada na versão que contém a CGRA-P.

Por outro lado, o *benchmark crc32* foi acelerado apenas 1,22x no núcleo 2P, porém, foi acelerado 1,82 no núcleo 8-*issue*. Essa aplicação é altamente dependente da memória e o caminho crítico está nas cadeias de operações de leitura da memória. Enquanto a CGRA não fornece aprimoramentos significativos de desempenho para esse *benchmark*, como se pode observar na Figura 3, o processador superescalar pode acelerar essas cadeias de operações de memória usando a fila de leitura/escrita (Brandalero & Beck, 2017).

4.3 Projetos *Multicore*

Neste trabalho, propomos a geração de três projetos *multicore*: o primeiro é a combinação de núcleos e aceleradores que atingem o maior desempenho possível para o conjunto de *benchmarks* analisado. O segundo é a combinação de núcleos e aceleradores limitados por um limite de desempenho; e o terceiro é a combinação de núcleos para operar com um limite especificado de consumo de energia.

1) ***Multicore* de melhor desempenho**: nesta proposta, combinamos 16 núcleos no *multicore*, um por *benchmark*. Para cada *benchmark*, selecionamos o núcleo que oferece o melhor desempenho. A Figura 4(a) detalha o *multicore* proposto. Como é possível observar, o *multicore* é predominantemente composto de processadores de 8-*issue* acoplados à maior versão da CGRA. Para avaliar o custo desse *multicore*, estimamos a área de cada núcleo e a energia gasta na execução para executar cada *benchmark*, conforme descrito na seção 4.2. A Tabela 4 detalha os resultados da área. Considerando os dados apresentados na tabela, o *multicore* da Figura 4(a) ocupa 39,73 mm^2. Destacamos que os resultados de área consideram apenas os componentes dos núcleos. A estimativa da área de interconexão de rede está fora do escopo deste estudo.



(a) (b)

Figura 4: Combinação de núcleos no *multicore* (a) para alcançar o melhor desempenho para cada *benchmark* (b) considerando uma perda de até 10%

Tabela 4: Estimação de área em mm^2 para todos os núcleos na tecnologia de 22nm

SUPERESCALAR			CGRA			COMBINAÇÃO DE NÚCLEOS								
2	**4**	**8**	**P**	**M**	**G**	**2P**	**2M**	**2G**	**4P**	**4M**	**4G**	**8P**	**8M**	**8G**
0,92	1,45	2,30	0,13	0,25	0,50	1,05	1,37	1,42	1,58	1,70	1,95	2,43	2,55	2,80

HOLOS ISSN 1807 - 1600

O gráfico na Figura 5 mostra os resultados de consumo de energia para cada *benchmark* em execução no respectivo núcleo cujo melhor desempenho foi alcançado. O gráfico detalha o consumo energia em percentagem, separando o consumo de energia do superescalar e o consumo de energia da CGRA. Por uma questão de simplicidade e comparação, o gráfico é normalizado e 100% representa o consumo de energia de cada *benchmark* executando apenas no superescalar. Na parte inferior de cada barra, mostramos a energia total (em *mJ*) gasta na execução do *benchmark*. Os custos de decodificação e escalonamento foram amortizados, bem como o custo da fila de leitura/escrita, pois as instruções são alocadas apenas uma vez quando a configuração é reutilizada e a execução é mais eficiente no CGRA.

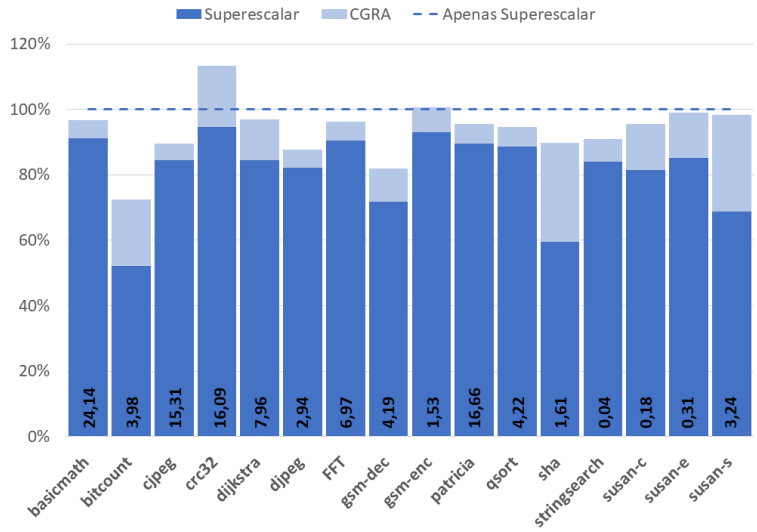


Figura 5: Consumo de energia por benchmark

Para o *benchmark bitcount*, por exemplo, o núcleo em que o melhor desempenho foi alcançado foi 8G. Na Figura 5, comparamos o ganho de energia do 8G com o superescalar de 8-*issue* sem acelerador. Pode-se observar que a execução no núcleo 8G melhora os ganhos de energia em mais de 20%. Como mencionado anteriormente, esse *benchmark* possui uma parte muito significativa do código que é acelerado no CGRA por três configurações principais que são reutilizadas várias vezes. Por outro lado, o *crc32* tem o melhor desempenho alcançado no núcleo 8M, com aceleração de 1,91 contra 1,81 do núcleo 8-*issue* sem acelerador que alcança uma melhoria muito pequena no desempenho. A energia extra gasta pelo núcleo 8M pode ser explicada pela necessidade de buscar configurações na cache de configuração.

2) ***Multicore* com limite desempenho**: nesta próxima análise, avaliamos o impacto da definição de uma taxa de aceitação de perda de desempenho. A ideia por trás deste estudo é reduzir a energia e a área, porém, limitando a perda de desempenho. Por exemplo, neste estudo de caso, definimos a mesma perda de aceleração de 10% para todos os *benchmarks*. Vale salientar que essa perda pode ser definida de acordo com as necessidades do projetista para, por exemplo, atender restrições de tempo real de uma aplicação específica. No cenário avaliado, as opções de projeto podem reduzir apenas a aceleração para cada *benchmark* em até 10% da aceleração mais alta encontrada (obtida a partir do *multicore* da Figura 4(a)). A Figura 4(b) apresenta o *multicore* resultante.

HOLOS ISSN 1807 - 1600

A Tabela 5 lista todos os *benchmarks* seguidos pela melhor aceleração e o núcleo em que essa aceleração foi alcançada (coluna "Melhor desempenho"); a redução de aceleração limitada à 10% e o núcleo em que essa aceleração foi alcançada (coluna "10% de redução"). Como é possível observar, apenas o *benchmark susan-s* permaneceu com o mesmo tipo de núcleo para obter o melhor desempenho. E, para alguns *benchmarks* como *basicmath* e *fft*, a perda de desempenho aceitável implica na seleção de um núcleo superescalar sem acelerador para executar esses *benchmarks*.

Esse novo projeto *multicore* com 16 núcleos pode economizar mais de 30% da área, enquanto se perde no máximo 10% na aceleração. A energia economizada foi de 13%. Adicionalmente, na tentativa de reduzir os custos de maneira mais drástica, também geramos um *multicore* composto por uma única unidade de cada núcleo em que a perda de desempenho foi alcançada. Isso significa que não há núcleos repetidos. Esse *multicore* mínimo possui 7 núcleos, destacados em cinza na Tabela 5. Os *benchmarks* que exigem o mesmo tipo de núcleo seriam executados sequencialmente, como o *susan-e* e o *susan-c* que exigem um núcleo 2-*issue* com CGRA-G. A área total para esse novo projeto é de 12,55 mm^2, que é menos da metade da área do projeto de 16 núcleos com 10% de perda de desempenho.

3) ***Multicore* com limite de energia**: Nesta última análise, propusemos um projeto *multicore* limitado pela energia. Nosso estudo considerou um cenário em que é necessária uma redução de 20% de energia para todos os *benchmarks*. Para esse *multicore* consideramos como base o núcleo que executou o *benchmark* com o melhor desempenho para cada *benchmark*. Em seguida, selecionamos o núcleo que mostra o melhor desempenho depois de aplicar a redução de 20% de energia. Caso esta análise indique que mais de um *benchmark* utiliza o mesmo tipo de núcleo, escolhemos apenas um núcleo por tipo no *multicore* projetado. O *multicore* gerado é mostrado na Figura 6. Como é possível observar, esse *multicore* tem dois núcleos com superescalar 2-*issue*, um acoplado a uma CGRA-M e o outro a uma CGRA-G e três núcleos 4-*issue* acoplados a cada uma das versões da CGRA. A execução de todos os *benchmarks* nesse *multicore* consumiu uma energia de 102,16 mJ. Esse consumo de energia representa 28% de ganho de energia quando comparado ao *multicore* proposto anteriormente, ou seja, ainda foi possível reduzir o consumo de energia acima dos 20% necessários.

Tabela 5: *Multicores* gerados considerando o melhor desempenho e com um limite de 10% no desempenho

Benchmark	Melhor desempenho		10% de redução		Economia de área
	Aceleração	*Núcleo*	*Aceleração*	*Núcleo*	
basicmath	1,85	8-issue G	1,68	8-issue	18%
bitcount	2,77	4-issue G	2,59	2-issue M	30%
cjpeg	1,55	8-issue M	1,47	4-issue P	38%
crc32	1,91	8-issue M	1,74	4-issue P	38%
dijkstra	2,11	8-issue G	1,96	4-issue M	39%
djpeg	1,71	8-issue P	1,55	4-issue P	35%
FFT	1,91	8-issue M	1,73	8-issue	10%
gsm-dec	2,44	8-issue P	2,31	4-issue P	35%
gsm-enc	1,36	8-issue G	1,29	4-issue P	44%

patricia	1,75	8-issue G	1,66	8-issue P	13%
qsort	1,65	8-issue M	1,51	4-issue P	38%
sha	2,23	8-issue G	2,21	4-issue G	30%
stringsearch	1,58	8-issue G	1,44	4-issue P	44%
susan-c	1,40	8-issue G	1,26	2-issue G	49%
susan-e	1,25	4-issue G	1,19	2-issue G	27%
susan-s	1,46	2-issue M	1,46	2-issue M	0%
Total da Área		39,73mm^2		26,92 mm^2	33%

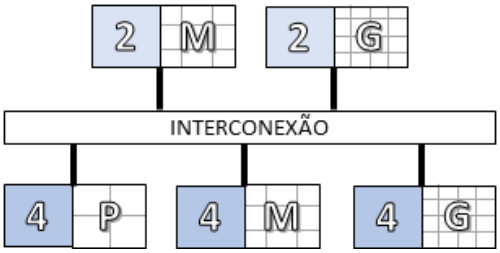


Figura 6: Combinação de núcleos para alcançar um limite de energia

4.4 Análise do tempo de simulação

Como cada uma das aplicações selecionadas foi executada em todos os núcleos, foram realizadas 192 simulações no total, o que demandou aproximadamente 16 horas de simulação. A Tabela 6 apresenta o tempo total gasto com a simulação de cada aplicação na infraestrutura do *gem5*. O pacote *Mibench* oferece dois exemplos de entradas para serem executados com seus *benchmarks*: a entrada pequena e a entrada grande, que se diferenciam pelo volume de dados que são passados para os algoritmos executarem. Nesse estudo, os *benchmarks* foram executados com a entrada pequena. Para a execução dos experimentos, foi usado como computador hospedeiro do *gem5*, um computador *Intel*(R) *Core*(TM) i7-4500U CPU @ 1.80GHz.

Na Tabela 6 o tempo de simulação foi agrupado por modelo de processador. Assim, a segunda coluna agrupa o tempo de simulação (em segundos) de todos os 4 núcleos que usam o processador 2-*issue* (2, 2P, 2M e 2G). A terceira coluna agrupa o tempo de simulação de todos os núcleos que usam o processador 4-*issue* (4, 4P, 4M e 4G); a quarta coluna agrupa o tempo de todos os núcleos que usam o processador 8-*issue*. Por fim, a última coluna apresenta o tempo de simulação em horas, minutos e segundos necessários para a simulação de cada uma das aplicações.

Como é possível perceber, as aplicações *qsort*, *crc32*, *patricia*, e *basicmath*, demandaram mais de 1 hora para simular a execução nos 12 núcleos distintos, mesmo usando a entrada pequena oferecida pelo *MiBench*. O menor tempo individual considerando as 16 aplicações e os 12 núcleos distintos foi de 3,93 segundos para a aplicação *stringsearch* no núcleo 2P, enquanto o maior tempo de simulação foi para a aplicação *basicmath* que demorou 46 minutos e 41 segundos para executar no núcleo 2P. Isso reforça a importância de investir em estratégias que possibilitem otimizar o tempo de simulação de um ponto no espaço de projeto. Além do tempo de simulação no *gem5*, também foi contabilizado o tempo de simulação no *McPAT*, que totalizou cerca de 17 minutos para

HOLOS ISSN 1807 - 1600

todas as 192 simulações. Uma vez que não há significativa discrepância de tempo por aplicação, não foi apresentado o detalhamento do tempo das simulações individuais.

Tabela 6: Tempo total de simulação no *gem5* (em segundos) das aplicações nos superescalares acoplado às versões da CGRA

	2-issue	4-issue	8-issue	TOTAL (hrs:min:s)
basicmath	8.002,63 s	7.734,23 s	7.625,19 s	**6:29:22**
bitcount	737,18 s	710,74 s	725,97 s	**0:46:04**
cjpeg	488,92 s	487,56 s	490,43 s	**0:24:26**
crc32	1.676,89 s	1.552,53 s	1.521,99 s	**1:19:11**
dijkstra	1.040,24 s	973,52 s	966,26 s	**0:49:40**
djpeg	108,83 s	105,29 s	101,81 s	**0:01:41**
FFT	794,77 s	755,13 s	760,56 s	**0:38:30**
gsm-dec	206,37 s	189,83 s	193,68 s	**0:09:49**
gsm-enc	410 s	401,09 s	397,54 s	**0:20:08**
patricia	3.858,85 s	3.746,64 s	3842 s	**3:10:47**
qsort	2.296,14 s	2.258,39 s	2.345,82 s	**1:55:00**
sha	204,93 s	188,22 s	196,95 s	**0:09:50**
stringsearch	12,45 s	12,53 s	13,2 s	**0:12:42**
susan-c	20,49 s	20,35 s	20,34 s	**0:01:01**
susan-e	41,35 s	40,67 s	39,91 s	**0:02:01**
susan-s	417,89 s	427,78 s	421,75 s	**0:21:07**

5 CONCLUSÕES

Este artigo apresentou três diferentes projetos de arquitetura *multicore* otimizada, compostos por processadores superescalares e aceleradores reconfiguráveis de granularidade grossa. Os projetos foram gerados considerando aprimoramento de desempenho e restrições de consumo de energia e área. Para gerar os projetos, usamos uma metodologia que combinou três processadores de diferentes larguras de *issue* (2-*issue*, 4-*issue* e 8-*issue*) com três versões do CGRA (denominadas P, M e G) variando o número de unidades funcionais. Simulamos um conjunto de *benchmarks* e apresentamos três projetos gerados que foram criados para alcançar 1) a melhor combinação de processadores superescalares e CGRAs para oferecer o melhor desempenho; 2) a combinação de núcleos para proporcionar uma perda aceitável em 10% da aceleração; e 3) a combinação mínima de núcleos para executar todos os *benchmarks* com uma economia de energia de 20%.

Em nosso cenário de avaliação, foi possível obter uma aceleração de até 2,8x e, de acordo com os limites de área e energia, foi possível economizar mais de 30% na área e reduzir o consumo de energia em 28%. Como trabalhos futuros, pretendemos realizar um estudo de caso incluindo um cenário de multitarefa e considerar o impacto da rede de interconexão em nossos resultados. Além disso, pretendemos desenvolver uma ferramenta automatizada para exploração do espaço de projeto de processadores superescalares e CGRAs, possibilitando que outros parâmetros microarquiteturais sejam avaliados no processo.

6 REFERÊNCIAS

ARM. (2020). *Arm big.little technologies.* https://www.arm.com/why-arm/technologies/big-little. (Accessed: 2020-04-14)

Binkert, N., Beckmann, B., Black, G., Reinhardt, S. K., Saidi, A., Basu, A., ... others (2011). The gem5 simulator. *ACM SIGARCH Computer Architecture News*, *39*(2), 1–7.

Bouthaina, D., Baklouti, M., Niar, S., & Abid, M. (2013). Shared hardware accelerator architectures for heterogeneous mpsocs. In *2013 8th international workshop on reconfigurable and communication-centric systems-on-chip (recosoc)* (pp. 1–6).

Brandalero, M., & Beck, A. C. S. (2017). A mechanism for energy-efficient reuse of decoding and scheduling of x86 instruction streams. In *Design, automation & test in europe conference & exhibition (date), 2017* (pp. 1468–1473).

Brandalero, M., Shafique, M., Carro, L., & Beck, A. C. S. (2019, March). Transrec: Improving adaptability in single-isa heterogeneous systems with transparent and reconfigurable acceleration. In 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE).

Butko, A., Bruguier, F., Novo, D., Gamatié, A., & Sassatelli, G. (2019). Exploration of performance and energy trade-offs for heterogeneous multicore architectures. *arXiv preprint arXiv:1902.02343*.

Compton, K., & Hauck, S. (2002). Reconfigurable computing: a survey of systems and software. *ACM Computing Surveys (csuR)*, *34*(2), 171–210.

Compton, K., & Hauck, S. (2008). Automatic design of reconfigurable domain-specific flexible cores. *IEEE transactions on very large scale integration (VLSI) systems*, *16*(5), 493–503.

Cong, J., Ghodrat, M. A., Gill, M., Grigorian, B., Gururaj, K., & Reinman, G. (2014). Accelerator-rich architectures: Opportunities and progresses. In *Proceedings of the 51st annual design automation conference* (pp. 1–6).

Duhem, F., Muller, F., Bonamy, R., & Bilavarn, S. (2015). FoRTReSS: a flow for design space exploration of partially reconfigurable systems. Design Automation for Embedded Systems.

Gao, C., Gutierrez, A., Rajan, M., Dreslinski, R. G., Mudge, T., & Wu, C.-J. (2015). A study of mobile device utilization. In *2015 ieee international symposium on performance analysis of systems and software (ispass)* (pp. 225–234).

Greenhalgh, P. (2011). Big. little processing with arm cortex-a15 & cortex-a7: Improving energy efficiency in high-performance mobile platforms. *white paper, ARM Ltd*.

Guthaus, M. R., Ringenberg, J. S., Ernst, D., Austin, T. M., Mudge, T., & Brown, R. B. (2001). Mibench: A free, commercially representative embedded benchmark suite. In *Proceedings of the Fourth Annual IEEE International Workshop on Workload Characterization.*.

Hartenstein, R. (2001). Coarse grain reconfigurable architecture (embedded tutorial). In *Proceedings of the 2001 Asia and South Pacific Design Automation Conference* (pp. 564–570). New York, NY, USA

Hill, M. D., & Marty, M. R. (2008). Amdahl's law in the multicore era. *Computer*, *41*(7), 33–38.

Hussain, W., Airoldi, R., Hoffmann, H., Ahonen, T., & Nurmi, J. (2014). Design of an accelerator-rich architecture by integrating multiple heterogeneous coarse grain reconfigurable arrays over a network-on-chip. In 2014 IEEE 25th International Conference on Application-Specific Systems, Architectures and Processors (pp. 131-138). IEEE.

Kamdar, S., & Kamdar, N. (2015). big. little architecture: Heterogeneous multicore processing. *International Journal of Computer Applications*, *119*(1).

Kareemullah, H., Janakiraman, N., & Kumar, P. N. (2017). A survey on embedded reconfigurable architectures. In *2017 international conference on communication and signal processing (iccsp)* (pp. 1500–1504).

Koenig, R., Bauer, L., Stripf, T., Shafique, M., Ahmed, W., Becker, J., & Henkel, J. (2010). Kahrisma: a novel hypermorphic reconfigurable-instruction-set multi-grained-array architecture. In *2010 Design, Automation & Test In Europe Conference & Exhibition (DATE 2010)*.

Koeplinger, D., Prabhakar, R., Zhang, Y., Delimitrou, C., Kozyrakis, C., & Olukotun, K. (2016). Automatic generation of efficient accelerators for reconfigurable hardware. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*.

Koutras, I., Maragos, K., Diamantopoulos, D., Siozios, K., & Soudris, D. (2017). On supporting rapid prototyping of embedded systems with reconfigurable architectures. *Integration*, *58*, 91–100.

Kuon, I., Tessier, R., Rose, J., et al. (2008). Fpga architecture: Survey and challenges. *Foundations and Trends* R *in Electronic Design Automation*, *2*(2), 135–253.

Li, S., Ahn, J. H., Strong, R. D., Brockman, J. B., Tullsen, D. M., & Jouppi, N. P. (2009). Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures. In *Proceedings of the 42nd annual ieee/acm international symposium on microarchitecture* (pp. 469–480).

Li, W., Zeng, X., Dai, Z., Nan, L., Chen, T., & Ma, C. (2017). A high energy-efficient reconfigurable vliw symmetric cryptographic processor with loop buffer structure and chain processing mechanism. *Chinese Journal of Electronics*, *26*(6), 1161–1167.

Mao, Y., You, C., Zhang, J., Huang, K., & Letaief, K. B. (2017). A survey on mobile edge computing: The communication perspective. IEEE Communications Surveys & Tutorials, 19(4).

Mishra, A., & Tripathi, A. K. (2014). Energy efficient voltage scheduling for multi-core processors with software controlled dynamic voltage scaling. *Applied Mathematical Modelling*, *38*(14), 3456–3466.

Neshatpour, K., Mokrani, H. M., Sasan, A., Ghasemzadeh, H., Rafatirad, S., & Homayoun, H. (2018). Architectural considerations for fpga acceleration of machine learning applications in mapreduce. In *Proceedings of the 18th international conference on embedded computer systems: Architectures, modeling, and simulation* (pp. 89–96).

Nguyen, H. K., Le-Van, T.-V., & Tran, X.-T. (2018). A survey on reconfigurable system-on-chips. *REV Journal on Electronics and Communications*, *7*(3-4). NVIDIA. (2019). *Tegra mobile processors.* http://www.nvidia.com/. (Accessed: 2019-06-29)

Reddy, D., Koufaty, D., Brett, P., & Hahn, S. (2011). Bridging functional heterogeneity in multicore architectures. *ACM SIGOPS Operating Systems Review*, *45*(1), 21–33.

SAMSUNG. (2019). *The Samsung Reference Platform.* http://www.samsung.com/.

(Accessed: 2019-06-29)

Scogland, T., Balaji, P., Feng, W.-c., & Narayanaswamy, G. (2008). Asymmetric interactions in symmetric multi-core systems: analysis, enhancements and evaluation. In *Sc'08: Proceedings of the 2008 acm/ieee conference on supercomputing* (pp. 1–12).

Smit, G. J., Havinga, P. J., Smit, L. T., Heysters, P. M., & Rosien, M. A. (2002). Dynamic reconfiguration in mobile systems. In *International conference on field programmable logic and applications* (pp. 171–181).

Souza, J. D., Carro, L., Rutzig, M. B., & Beck, A. C. S. (2016, March). A reconfigurable heterogeneous multicore with a homogeneous ISA. In 2016 Design, Automation & Test in Europe Conference & Exhibition (DATE) (pp. 1598-1603). IEEE.

Suh, D., Kwon, K., Kim, S., Ryu, S., & Kim, J. (2012). Design space exploration and implementation of a high performance and low area coarse grained reconfigurable processor. In *2012 international conference on field-programmable technology* (pp. 67–70).

Synopsys. (2020). *RTL Synthesys.* https://www.synopsys.com/. (Accessed: 2020-04-14)

Tehre, V., & Kshirsagar, R. (2012). Survey on coarse grained reconfigurable architectures. *International Journal of Computer Applications*, *48*(16), 1–7.

Van Craeynest, K., & Eeckhout, L. (2013). Understanding fundamental design choices in single-isa heterogeneous multicore architectures. *ACM Transactions on Architecture and Code Optimization (TACO)*, *9*(4), 32.

Watkins, M. A., & Albonesi, D. H. (2010). Remap: A reconfigurable heterogeneous multicore architecture. In *2010 43rd annual IEEE/ACM International Symposium on Microarchitecture*.

COMO CITAR ESTE ARTIGO:

Lopes, A. S. B., Pereira, M.M. (2020) Aceleradores reconfiguráveis no Projeto *Multicore*: uma análise de custo *versus* benefício. Holos. 36(6), 1-20.

SOBRE OS AUTORES

HOLOS
ISSN 1807 - 1600

A. S. B. LOPES
Professora do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte, da área de Sistemas de Informação. Possui graduação em Ciência da Computação (2009) e mestrado em Sistemas e Computação (2011) pela Universidade Federal do Rio Grande do Norte. Cursando doutorado em Ciência da Computação no Programa de Pós-graduação em Sistemas e Computação (PPgSC) da Universidade Federal do Rio Grande do Norte (UFRN). Desenvolve pesquisa principalmente nas áreas de Arquiteturas Reconfiguráveis, MPSoCs, Redes em *Chip* e Sistemas Tolerantes a Falhas.
E-mail: alba.lopes@ifrn.edu.br
ORCID ID: https://orcid.org/0000-0002-2016-055X

M. M. PEREIRA
Professora Adjunta na Universidade Federal do Rio Grande do Norte. Possui graduação em Ciência da Computação (2006) e mestrado em Sistemas e Computação (2008) pela Universidade Federal do Rio Grande do Norte. Possui doutorado em Ciência da Computação pela Universidade Federal do Rio Grande do Sul (2012). Em 2010 realizou seu estágio sanduíche de doutorado no ITIV/KIT em Karlsruhe, Alemanha. Tem experiência na área de Ciência da Computação, com ênfase em Arquitetura de Sistemas de Computação e Sistemas Embarcados, atuando principalmente nos seguintes temas: sistemas embarcados, arquiteturas reconfiguráveis, e circuitos tolerantes a falhas. Desde 2017, coordena o grupo de afinidade *IEEE Women in Engineering*, atuando em projetos de pesquisa e inovação, bem como ações educativas sobre o tema.
E-mail: monicapereira@dimap.ufrn.br
ORCID ID: https://orcid.org/0000-0002-6580-1250

Editor(a) Responsável: Fábio Paiva

Pareceristas *Ad Hoc*: Max Silveira, Marco de Oliveira Domingues e Valério Medeiros Jr