Utilização dos simuladores do MPSoCBench para o ensino e aprendizagem de Arquitetura de Computadores

Liana Duenha
Universidade Federal de Mato Grosso do Sul
UFMS - Campo Grande, MS, Brasil
Email: lianaduenha@facom.ufms.br

Rodolfo Azevedo
Universidade Estadual de Campinas
UNICAMP - Campinas, SP, Brasil
Email: rodolfo@ic.unicamp.br

Abstract

Este artigo propõe uma metodologia de ensino complementar às disciplinas de Arquitetura de Computadores fundamentada em projetos de exploração arquitetural, experimentação e análise de resultados. Adotou-se o MPSoCBench como ferramenta de simulação, o qual possui modelos funcionais de sistemas multiprocessados e fornece estimativas de desempenho, temporização e consumo de potência para principais unidades funcionais dos sistemas computacionais.

1. Introdução

Arquitetura de Computadores é a disciplina responsável por desenvolver uma compreensão mais profunda do ambiente de hardware sobre o qual a computação é realizada e está presente no currículo dos cursos de Ciência da Computação, Engenharia da Computação e cursos afins. Nessa disciplina, os alunos devem conhecer o projeto dos principais componentes funcionais de um sistema computacional, entender como interagem entre si e estudar métodos para avaliação do desempenho dessas unidades funcionais e do sistema como um todo.

A abordagem comumente adotada nas disciplinas de Arquitetura de Computadores é essencialmente teórica, baseada em exposição de conteúdo e discussões. Uma abordagem complementar é fazer uso de simuladores de sistemas para realização de atividades práticas baseadas em experimentação, com o objetivo de realizar avaliação dos diversos componentes arquiteturais que, consequentemente, resulta em fixação do conteúdo e amadurecimento de conceitos. Entretanto, o conteúdo programático de ambas as disciplinas é extenso e é comum que o professor não consiga, na carga horária proposta, explorar o conteúdo em atividades práticas complementares. Nesse caso, pode-se utilizar uma disciplina complementar com enfoque puramente experimental e prático.

O objetivo deste trabalho é propor uma nova metodologia de ensino complementar às disciplinas de Arquitetura de Computadores, principalmente àquelas fundamentadas nas arquiteturas RISC. Especificamente, este trabalho descreve o uso de uma ferramenta de simulação de código aberto para apoio pedagógico às disciplinas Arquitetura de Computadores e afins, como forma de complementar a metodologia de ensino adotada pelo docente.

A ferramenta sugerida é o MPSoCBench [4], um *framework* de simulação de sistemas multiprocessados composto por um conjunto escalável de modelos de plataformas para avaliação e exploração de componentes arquiteturais, aplicação de novas metodologias, desenvolvimento de software paralelo e otimização de hardware.

Este trabalho está organizado da seguinte maneira: a Seção 2 descreve os principais componentes e funcionalidades do *framework* MPSoCBench; a Seção 3 propõe uma metodologia baseada em projetos para aproveitar os recursos de simulação do MPSoCBench no ensino de Arquitetura de Computadores e disciplinas afins; a Seção 4 conclui este artigo.

2. MPSoCBench

No contexto deste trabalho, uma *plataforma virtual* é um modelo funcional de um sistema completo, contendo a descrição dos seus componentes em alto nível de abstração. Cada plataforma virtual modelada e simlada pelo MPSoCBench contém um conjunto de 1 a 64 núcleos de processamento, *caches* de instruções e dados, conectados a uma memória externa e a outros componentes por meio de mecanismos de interconexão, os quais podem ser tão simples quanto um barramento (*crossbar*), ou tão elaborados quanto as redes em chip (*NoCs*). A modelagem e simulação são baseadas em SystemC [2] e TLM2 [1], que são extensões de C/C++ especializadas para modelagem em nível de sistema, cujo *kernel* de simulação é baseado em eventos discretos.

MPSoCBench é um framework baseado em Linux que depende da instalação prévia do SystemC [2] e do ArchC [3] e pode ser baixado através da página `http://www.archc.org/benchs/mpsocbench`, assim como os seus tutoriais de instalação e utilização. A configuração dos modelos é automatizada por *scripts*, que permitem a criação de centenas de configurações distintas em uma única linha de comando.

2.1. Modelos de Processadores

O MPSoCBench contém quatro modelos funcionais de processadores: ARM, MIPS, PowerPC e SPARC, que correspondem às versões de 32 bits do conjunto de instruções ARMv5e, RISC MIPS-I, SPARC-V8 e PowerPC, respectivamente, todos com emulação de chamadas ao sistema operacional. Os modelos são descritos utilizando a Linguagem de Descrição de Arquiteturas (ADL) ArchC [3]. Esses modelos contêm informações de ciclos requeridos para executar cada instrução do conjunto. O modelo de temporização adotado é baseado em implementações reais dos processadores ARM9E-S, MIPS32 M5150, PowerPC 405TM Core e UT699 LEON 3FT/SPARCTM V8. Além de poder modificar a quantidade de ciclos para cada instrução do ISA utilizando o método `setCycles`, o usuário também pode escolher a frequência do processador utilizando o método `set_proc_freq`. Assim, ao final da simulação, é fornecido o valor de tempo de execução da aplicação no sistema simulado. Para que esse valor seja o mais preciso possível, é necessário que a modelagem dos demais componentes da plataforma também contenham informação sobre suas latências.

2.2. Memória

A maioria dos simuladores funcionais utiliza uma memória compartilhada simples com latência fixa, o que pode ser considerado insuficiente para estudos mais realistas, em que a dinâmica entre processadores e memória influencia fortemente o desempenho do sistema. O MPSoCBench oferece duas abordagens de memória: uma memória compartilhada simples, pré-configurada com 512MB, suficiente para a execução de todas as aplicações disponíveis no *framework*, considerando a maior quantidade de núcleos de processamento. A segunda abordagem consiste de um modelo de memória mais preciso utilizando o simulador DRAMSim [9], um simulador de memórias DRAM com precisão de ciclos, que oferece descrição detalhada de diversos tipos de memória e provê, ao final da simulação, estatísticas de memória que incluem latência e potência média por *ranks* e por banco de memória. O uso do DRAMSim é opcional e é habilitado por meio de uma *flag* definida no arquivo *defines.h* no diretório raiz do MPSoCBench.

2.3. Caches

O modelo de memórias caches oferecidas no MPSoCBench contêm os seguintes parâmetros de configuração: a quantidade de blocos de cache; a largura do bloco em bytes, a associatividade, políticas de escrita (*write through* ou *write back*) e de **substituição** de blocos (aleatória, FIFO e LRU). O último nível de *cache* é conectado a um diretório compartilhado, que implementa o algoritmo MSI de coerência de *cache*, habilitando computação paralela com recursos compartilhados. Na versão atual do *framework*, o protocolo de coerência de caches está implementado apenas para caches com política de escrita *write-through*. Para estender o mecanismo de coerência para caches com política de escrita *write-back* seria necessário um mecanismo mais sofisticado de comunicação entre as caches, o qual não está disponível no *framework* atualmente.

2.4. Interconexões

Há três mecanismos de interconexão: um barramento (*crossbar*) muito simples e útil para exploração funcional; e duas implementações de interconexão baseada em redes (ou NoCs, do inglês, *Network on Chip*), uma com temporização relaxada (NoC-LT) e outra com temporização aproximada (NoC-AT). Embora ambas contenham um mecanismo de temporização, a NoC-AT é mais precisa pois sua implementação e temporização são baseadas em uma modelagem de NoC RTL [8] e a suas interfaces de comunicação obedecem protocolos de comunicação TLM2 não-bloqueantes [1]. Considerando diferenças fundamentais de precisão e desempenho dos mecanismos de interconexão apresentados, a escolha do mecanismo de comunicação mais apropriado pode ser feita com base no caso de uso, como mostrado na Tabela 1:

2.5. Modelos de Consumo de Energia

O *framework* possui modelos de consumo de energia baseados nos processadores LEON3 e PLASMA, que são versões RTL de código aberto compatíveis com os processadores SPARC e MIPS do MPSoCBench. O processo de caracterização de consumo médio por instrução é baseado no método Tiwari e foi realizado por Guedes et al. [5] utilizando diferentes tecnologias de caracterização baseadas em FPGAs (Xilinx Spartan, Xilinx Virtex, Altera Cyclone e Altera Stratix) e ASICs (OpenPDK 45nm) e diferentes frequências (40MHz, 50MHz, 125MHz, 250MHz e 400MHz). Os dados resultantes do processo de caracterização foram acoplados ao MPSoCBench para habilitar estimativas de consumo dinâmico dos processadores das plataformas *multicore*. Pode-se também habilitar um componente para gerenciamento global dos níveis de

Tabela 1. Interconexão mais apropriada para cada caso de uso do MPSoCBench

Issues	Cross-bar	NoC-LT	NoC AT
Estudos sobre caches e memórias	✔	✔	
Estudos sobre DVFS	✔	✔	
Exploração arquitetural		✔	✔
Verificação funcional de hardware	✔	✔	✔
Vazão das instruções	✔	✔	
Algoritmos de roteamento e tráfego na rede		✔	✔
Otimizações do processador	✔	✔	
Avaliação de consumo do processador e do sistema	✔	✔	
Desenvolvimento e avaliação de software paralelo	✔	✔	
Aspectos relativos à temporização			✔

voltagem e frequência em que cada core opera durante a simulação (DVFS) com o objetivo de estudar técnicas de escalabilidade dinâmica de voltagem e frequência para otimizar o consumo de energia na plataforma; entretanto, essa funcionalidade não foi explorada nesse trabalho. Mais detalhes podem ser obtidos a partir de [4].

2.6. Aplicações

Existem 17 aplicações paralelas disponíveis na ferramenta, adaptadas a partir do ParMiBench [6] e SPLASH2 [10] que estão descritas em [4]. Para evitar portar um sistema operacional completo, a ferramenta oferece a biblioteca acPThread.h, onde que são emuladas as principais funcionalidade da biblioteca POSIX PThread.h.

3. Metodologia

Esta seção propõe uma metodologia de ensino de Arquitetura de Computadores baseada em projetos, os quais contemplam a exploração de diversos recursos arquiteturais de sistemas computacionais por meio dos simuladores gerados pelo MPSoCBench. A proposta consiste em dividir as atividades da disciplina em projetos que, em conjunto, cubram parte significativa do conteúdo programático de Arquitetura de Computadores. Considerando a particularidade dos recursos presentes nos simuladores do MPSoCBench, é possível realizar a exploração de vários aspectos arquiteturais, desde os mais elementares como análise de desempenho dos processadores até técnicas mais avançadas de escalabilidade dinâmica de voltagem e frequência ou exploração do espaço de projeto utilizando ferramentas auxiliares para estimativa de parâmetros físicos como, por exemplo, área do *chip*. Selecionamos oito projetos que serão descritos nas seções a seguir.

3.1. Análise comparativa entre modelos de processadores

Há diferenças fundamentais entre os modelos de processadores disponíveis na ferramenta; espera-se, consequentemente, diferentes desempenhos ao executar o mesmo conjunto de aplicações variando-se apenas o modelo do processador. Nesse projeto, os alunos podem explorar aspectos fundamentais da arquitetura, como descrição do banco de registradores, formato e comportamento das instruções, bem como avaliar como estas características arquiteturais influenciam no desempenho do processador. A Figura 1 ilustra a quantidade de instruções executadas por cada um dos processadores ao executar o mesmo conjunto de aplicações. Embora todas as aplicações possam ser executadas em um ambiente com mais núcleos de processamento, elas foram executadas em plataformas com apenas um núcleo para evitar que instruções relativas ao controle de concorrência afetassem os resultados (já que parte dos mecanismos de sincronização é baseada em espera ocupada).

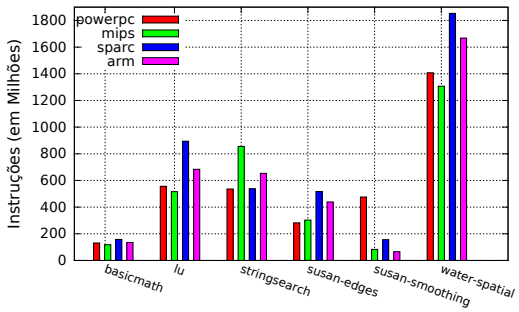


Figura 1. Instruções executadas em aplicações paralelas (em milhões).

3.2. Exploração dos mecanismos de interconexão e rede *intrachip*

Os simuladores do MPSoCBench oferecem dados de tráfego de rede *intrachip*, tornando possível avaliar mecanismos de roteamento, topologias e otimizações dos mecanismos de interconexão. Executamos aplicações em plataformas MIPS com 1 a 64 núcelos conectados por meio

de uma rede com temporização aproximada (NoC-AT) e mostramos na Figura 2 a quantidade total de requisições feitas à rede e total de *hops* (conta-se um *hop* a cada vez que um pacote é transmitido de um roteador para outro). Por meio destas ilustrações, é possível estudar a escalabilidade das aplicações. Por exemplo, no caso da aplicação SHA, à medida que aumentamos a quantidade de núcleos, aumenta também a quantidade de arquivos de entrada sobre os quais a aplicação irá processar; assim, espera-se um aumento muito maior de requisições à rede e *hops*. Por outro lado, a aplicação Susan-corners processa a mesma imagem de entrada, independentemente da quantidade de núcleos de processamento; assim, nota-se que a quantidade de requisições à rede e a quantidade de *hops* cresce lentamente à medida que aumentamos a quantidade de núcleos. Os relatórios de saída do simulador também permitem avaliar o tráfego na rede sob a perspectiva de cada roteador. Assim, é possível avaliar *gargalos* que comprometem a escalabilidade da plataforma, e a partir destes resultados, propor novos topologias para a rede e/ou novos algoritmos de roteamento.

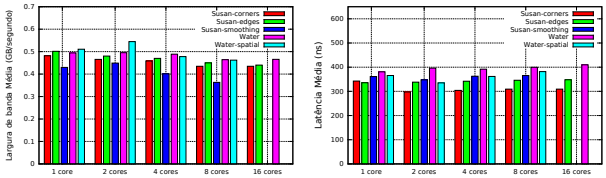
3.3. Exploração de Memória *Cache*

O MPSoCBench permite instanciar caches de dados e instruções utilizando parâmetros como associatividade, quantidade de blocos, largura do bloco, política de escrita e política de substituição de blocos, permitindo que o usuário realize exploração de desempenho das aplicações considerando diversas configurações de memórias *cache*. Para mostrar como funciona a hierarquia de memória nos simuladores do MPSoCBench, executamos 11 aplicações em plataformas com um único núcleo de processamento, no qual fixamos cache de dados com 512 blocos, 32 bytes por bloco, e variamos a associatividade (1-way, 2-way, 4-way e 8-way). As Figuras 3(a), 3(b) mostram a porcentagem de acertos de leitura e escrita na cache de dados e verifica-se que a cache 2-way obteve as melhores taxas de acerto para a maioria das aplicações. Para avaliar políticas de substituição de blocos, simulamos a aplicação Stringsearch em diversas plataformas *multicore* utilizando três algoritmos: *First In First Out* (FIFO), *Menos recentemente utilizado* (LRU) e aleatório (Random). A Figure 3(c) mostra que a quantidade de substituições de blocos descresce à medida que aumentamos a quantidade de núcleos de processamento (onde cada núcleo possui sua própria cache) e a política LRU é a que causa menor quantidade de substituições de blocos. Vale ressaltar que o uso de cache de dados em plataformas com múltiplos núcleos habilita automaticamente o uso do protocolo MSI para coerência das caches.

Uma extensão interessante para esse projeto é acrescentar aos experimentos relativos ao desempenho de *cache*, avaliação de área das *caches* utilizando o estimador de área McPAT [7]. Assim, os alunos podem realizar exploração de diversas arquiteturas de memórias *caches* utilizando um limite de área para as memórias ou para o chip como um todo e concluir, ao final do experimento, qual a melhor configuração da hierarquia de memória para otimizar o desempenho para algumas aplicações, sem extrapolar um envelope de área. O mesmo pode ser feito com limitantes de potência.

3.4. Exploração de Desempenho da memória principal

Uma vez integrado ao MPSoCBench, o DRAMSim oferece relatórios com diversos parâmetros para avaliação de memória a partir da simulação das aplicações. Nesse projeto, os alunos podem estudar distintos projetos de memória principal disponíveis na ferramenta e avaliar o impacto destas diferentes configurações de memória no desempenho das aplicações, consumo de potência e latência de memória. Simulamos cinco aplicações em plataformas com 1 a 16 núcleos de processamento, com caches de instrução e dados, 512 MB de memória RAM com um único canal e 8 bancos, caracterizada a partir do modelo DDR2 Micron. A Figura 4(a) e 4(b) mostra a largura de banda expressa em GB por segundo e a latência média expressa em nanossegundos.

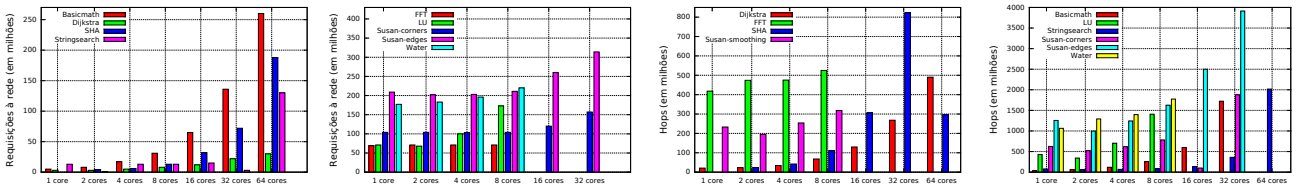


(a) (b)

Figura 4. (a) Largura de banda e (b) Latência Média de memória (média de 4 canais) executando cinco aplicações.

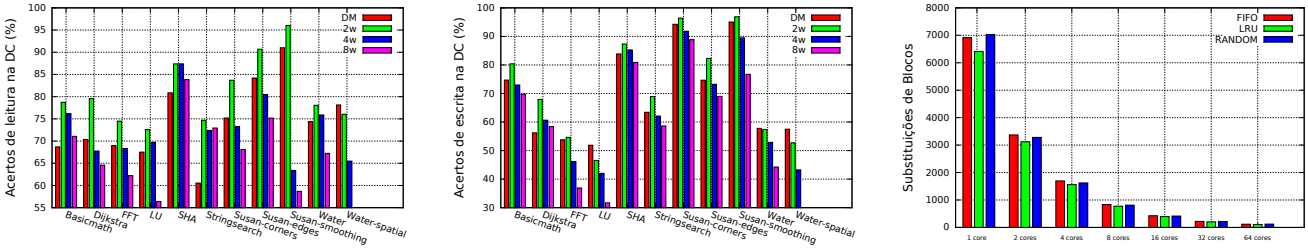
3.5. Temporização e níveis de abstração

As aplicações exploram distintas técnicas de paralelização, resultando diferenças em seus desempenhos e escalabilidade; oferecemos aqui dois exemplos na Figura 5 que ilustra o tempo de execução (simulado) das aplicações citadas em plataformas com múltiplos núcleos de processamento. A aplicação como LU apresenta aumento no tempo de execução à medida que aumentamos a quantidade de núcleos de processamento; isto ocorre pois a quantidade de dados de entrada também aumenta à medida que aumenta na quantidade de núcleos da plataforma. Já



(a) (b) (c) (d)

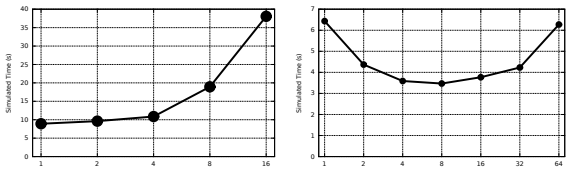
Figura 2. (a) Requisições à rede executando Basicmath, Dijkstra, SHA e Stringsearch (b) Requisições à rede executando FFT, LU, Susan-corners, Susan-edges, Water (c) *Hops* executando Dijkstra, FFT, SHA, Susan-Smoothing (d) *Hops* executando Basicmath, LU, Stringsearch, Susan-coners, Susan-edges, Water



(a) (b) (c)

Figura 3. (a) Acertos de leitura da cache de dados (b) Acertos de escrita na cache de dados (c) Substituições de blocos

a aplicação Stringsearch particiona os dados de entrada entre os núcleos disponívies, diminuindo a quantidade de processamento por núcleo; entretanto, o compartilhamento de recursos entre os cores e a sincronização interferem no tempo de execução. Neste contexto, os alunos podem, por exemplo, explorar aspectos relativos às técnicas de paralelização que melhor se adequam às aplicações, realizar análise de taxa de contenção e gargalos na comunicação que afetam o desempenho.

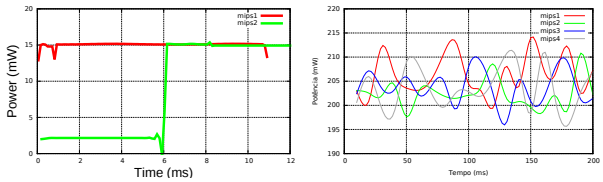


(a) LU (b) Stringsearch

Figura 5. Tempo de execução (simulado) de aplicações em plataformas com múltiplos núcleos de processamento

3.6. Exploração de Consumo dos processadores e caches

Uma vez que os processadores SPARC e MIPS possuem modelos de consumo de energia, é possível avaliar o comportamento dinâmico do consumo de energia dos núcleos de processamento presentes na plataforma. A Figura 6 mostra o perfil gerado pelo MPSoCBench executando duas aplicações paralelas (SHA e Basicmath) em plataformas com dois e quatro núcleos, respectivamente, e é possível observar as diferenças entre o perfil de consumo em cada uma delas. Durante a execução da aplicação SHA, um dos núcleos executa grande quantidade de acessos à memória para alocar e atribuir valores às variáveis, enquanto o outro núcleo aguarda em uma barreira. Essa diferença de fases é facilmente verificada na Figura 6(a). Por outro lado, todos os núcleos de processamento que executam a aplicação Basicmath realizam um conjunto de operações matemáticas sobre os dados alocados estaticamente, causando um perfil de consumo diferente da aplicação SHA, como pode ser visto na Figura 6(b).



(a) SHA (b) Basicmath

Figura 6. Potência dinâmica executando SHA (a) e Basicmath (b) em plataformas com dois e quatro núcelos, respectivamente

4. Conclusão

Este artigo propôs o uso dos simuladores do MPSoC-Bench como forma de apoio ao ensino e aprendizado de Arquiteturas de Computadores. A metodologia consiste na aplicação de um conjunto de projetos de experimentação e análise de resultados obtidos que, em conjunto, resultam no amadurecimento conceitual e fixação do conteúdo.

A metodologia foi aplicada na disciplina Tópicos em Arquitetura de Computadores (TAC) dos cursos de graduação em Engenharia da Computação e Ciência da Computação da Faculdade de Computação (Facom-UFMS). Essa disciplina tem como requisito a prévia aprovação na disciplina Arquitetura de Computadores I. A taxa de aprovação em TAC foi de 95%, com notas iguais ou acima de 8.0 para 89% dos alunos. Atribuímos o bom desempenho, em parte, ao fato de que apenas 16,7% dos alunos não haviam realizado as disciplinas Arquitetura de Computadores II, que oferece a fundamentação teórica necessária para que os alunos estivessem conceitualmente bem preparados para a realização dos projetos experimentais desenvolvidos em TAC. De acordo com a avaliação, 90% dos alunos tiveram alguma dificuldade durante o processo de instalação da ferramenta; entretanto, 50% destes conseguiram saná-las por meio dos tutoriais das ferramentas e os demais necessitaram apoio do professor. Ainda, após indagados sobre cada um dos tópicos abordados em cada projeto, cerca de 38% dos alunos, em média, consideram que seu grau de conhecimento sobre o assunto aumentou de 30% a 50%; e 34% disseram que o aumento foi de mais de 50%.

A partir dos resultados quantitativos obtidos e da avaliações e realizadas pelo professor e pelos alunos, consideramos promissora a aplicação da metodologia baseada em experimentação, simulação e análises de resultados como forma de apoio para o ensino de Arquitetura de Computadores que, em suma, resultou no amadurecimento de aspectos conceituais e aumento do interesse pela área.

Referências

[1] OSCI TLM-2.0 Language Reference Manual. Software version: TLM 2.0.1 Document version: JA32. http://www.systemc.org.

[2] IEEE Std 1666^{TM} Standard SystemC Language Reference Manual. IEEE Computer Society, January 2012.

[3] Rodolfo Azevedo, Sandro Rigo, Marcus Bartholomeu, Guido Araujo, Cristiano Araujo, and Edna Barros. The ArchC Architecture Description Language and Tools. In *International Journal of Parallel Programming. Vol. 33, No. 5*, pages 453–484. October 2005.

[4] Liana Duenha, Guilherme Madalozzo, Thiago Santiago, Fernando Moraes, and Rodolfo Azevedo. Mpsocbench: a benchmark for high-level evaluation of multiprocessor system-on-chip tools and methodologies. In *Journal of Parallel and Distributed Computing*, volume 95, pages 138–157. Elsevier, June 2016.

[5] Marcelo Guedes, Rafael Auler, Liana Duenha, Edson Borin, and Rodolfo Azevedo. An automatic energy consumption characterization of processors using archc. *Journal of Systems Architecture*, 59(8):603 – 614, 2013.

[6] S.M.Z. Iqbal, Yuchen Liang, and H. Grahn. Parmibench - an open-source benchmark for embedded multiprocessor systems. *Computer Architecture Letters*, 9(2):45–48, Feb 2010.

[7] Sheng Li, JH Ahn, and RD Strong. McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures. In *Proceedings of the 42nd Annual International Symposium on Microarchitecture*, pages 469–480, 2009.

[8] Fernando Moraes, Ney Calazans, Aline Mello, Leandro Möller, and Luciano Ost. Hermes: an infrastructure for low area overhead packet-switching networks on chip. volume 38, pages 69–93. Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, October 2004.

[9] P. Rosenfeld, E. Cooper-Balis, and B. Jacob. Dramsim2: A cycle accurate memory system simulator. In *IEEE-Computer Architecture Letters*, ISSN 1556-6056, pages 16–19, Maryland, USA, June 2011. IEEE.

[10] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta. The Splash-2 Programs: Characterization and Methodological Considerations. In *Proceedings of the 22nd International Symposium on Computer Architecture - ISCA'95*, pages 24–36. ACM, 1995.