UNIVERSIDADE FEDERAL DE PELOTAS
Centro de Desenvolvimento Tecnológico
Programa de Pós-Graduação em Computação

Dissertação

Exploração de Arquiteturas de Memórias Híbridas para Sistemas Embarcados utilizando Memórias Não Voláteis

Lisandro Luiz da Silva

Pelotas, 2017

Lisandro Luiz da Silva

Exploração de Arquiteturas de Memórias Híbridas para Sistemas Embarcados utilizando Memórias Não Voláteis

Dissertação apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal de Pelotas, como requisito parcial à obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Júlio Carlos Balzano de Mattos

Coorientadora: Profª Drª Lisane Brisolara de Brisolara

Pelotas, 2017

Universidade Federal de Pelotas / Sistema de Bibliotecas
Catalogação na Publicação

S586e Silva, Lisandro Luiz da

Exploração de arquiteturas de memórias híbridas para sistemas embarcados utilizando memórias não voláteis / Lisandro Luiz da Silva ; Júlio Carlos Balzano de Mattos, orientador ; Lisane Brisolara de Brisolara, coorientadora. — Pelotas, 2017.

131 f.

Dissertação (Mestrado) — Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, 2017.

1. Scratchpads. 2. Memórias híbridas. 3. Consumo energético. 4. Desempenho. I. Mattos, Júlio Carlos Balzano de, orient. II. Brisolara, Lisane Brisolara de, coorient. III. Título.

CDD : 005

Elaborada por Aline Herbstrith Batista CRB: 10/1737

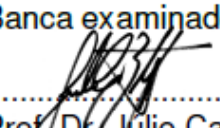
Lisandro Luiz da Silva

Exploração de Arquiteturas de Memórias Híbridas para Sistemas Embarcados utilizando Memórias Não Voláteis

Dissertação aprovada, como requisito parcial, para obtenção do grau de Mestre em Ciência da Computação, Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas.

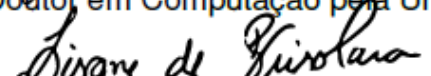
Data da Defesa: 02/05/2017

Banca examinadora:



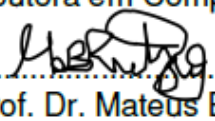
.....

Prof. Dr. Júlio Carlos Balzano de Mattos (Orientador)
Doutor em Computação pela Universidade Federal do Rio Grande do Sul



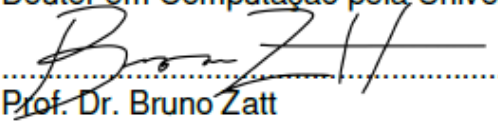
.....

Prof. Drª Lisane Brisolara de Brisolara (Coorientadora)
Doutora em Computação pela Universidade Federal do Rio Grande do Sul



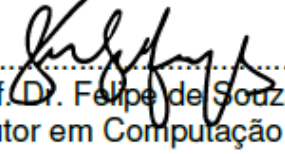
.....

Prof. Dr. Mateus Beck Rutzig
Doutor em Computação pela Universidade Federal do Rio Grande do Sul



.....

Prof. Dr. Bruno Zatt
Doutor em Microeletrônica pela Universidade Federal do Rio Grande do Sul



.....

Prof. Dr. Felipe de Souza Marques
Doutor em Computação pela Universidade Federal do Rio Grande do Sul

Dedico este trabalho aos meus pais e a todos que fizeram parte e colaboram de forma direta e indireta na minha caminhada.

Agradecimentos

Gostaria de agradecer a todas as pessoas que me ajudaram durante esses anos de estudo. Primeiramente, agradeço aos meus pais pelo apoio incondicional em permitir que eu pudesse vencer esta etapa de minha vida, eles merecem todo o meu carinho.

Agradeço à minha namorada Itana Sena, a qual sempre esteve ao meu lado durante esses cinco anos, incentivando-me a crescer e a prosperar os desejos do meu coração. O meu muito obrigado por tudo.

Agradeço ao meu orientador professor Júlio Carlos Balzano de Mattos e a minha coorientadora professora Lisane Brisolara de Brisolara que foram fundamentais no meu crescimento acadêmico, científico e profissional. Obrigado por fazerem parte da minha caminhada, pontuando, de forma construtiva, as minhas dificuldades e motivando-me a realizar este trabalho.

Aos docentes da Universidade Federal de Pelotas, os quais tive o privilégio de ser aluno, que fizeram parte do meu processo de aprendizado, contribuindo para a minha formação. Aos meus colegas de mestrado, por terem compartilhado todo aprendizado de forma humanizada durante todo esse tempo que tivemos juntos.

Por fim, agradeço a todos os colegas e amigos pelo apoio e que contribuíram, ainda que indiretamente, para a realização deste trabalho.

“Embora ninguém possa voltar atrás e fazer um novo começo, qualquer um pode começar agora e fazer um novo fim.” (CHICO XAVIER).

Resumo

SILVA, Lisandro L. **Exploração de Arquiteturas de Memórias Híbridas para Sistemas Embarcados utilizando Memórias Não Voláteis.** 2017. 131f. Dissertação (Mestrado em Ciência da Computação). - Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2017.

O consumo de energia é tão importante quanto o desempenho em sistemas embarcados alimentados a bateria, pois cada vez mais estes sistemas precisam processar computação intensiva com um baixo consumo energético. Devido à alta contribuição do acesso à memória no consumo total de energia de sistemas embarcados, a arquitetura de memória influencia fortemente os objetivos dos projetos dos dispositivos embarcados. Existem, na literatura, diversas técnicas de otimização do acesso à memória para sistemas embarcados, possibilitando por parte do projetista do sistema, uma exploração do espaço de projeto abrangente para a arquitetura de memória. Muitas dessas técnicas são propostas devido aos problemas enfrentados com o avanço da tecnologia, como por exemplo, a memória tradicional baseada em SRAM (*Static Random Access Memory*) *on-chip* tornou-se um gargalo em desempenho e consumo energético para o projeto de sistemas embarcados, devido ao seu alto *leakage* e latência de leitura. As tecnologias emergentes de memórias não voláteis (NVM, *Non-Volatile Memories*), tal como STT-RAM (*Spin-Transfer Torque RAM*) e *PCRAM (Phase Change RAM*), são soluções candidatas para os futuros sistemas de memória, pois elas possuem algumas vantagens sobre a memória SRAM tradicional, como por exemplo, um menor consumo energético para as operações de leitura. Este trabalho apresenta um estudo de exploração realizadas em memórias híbridas utilizando memórias não voláteis em sistemas embarcados. O trabalho apresenta a investigação do acesso à memória do processador embarcado ARMv5, a análise dos acessos realizados, juntamente com os impactos dos acessos à memória no consumo e no desempenho para diferentes modelos híbridos de memórias com tecnologias emergentes, para um determinado conjunto de benchmarks retirados do MiBench. Os resultados são significativos, pois, por exemplo, conseguiu-se aumentar o número de acessos da memória *scratchpad* (SPM) híbrida utilizando memória SRAM de 16KB, STT-RAM 32KB e PCM de 8KB (SPM 4) em cerca de 0,99% quando comparada com uma memória SPM tradicional utilizando SRAM de 32KB (SPM 1), além de reduzir em média 23,81% a latência e 49,24% o consumo energético e diminuir o *leakage* em 46,65% e a área em 16.29%.

Palavras-chave: *Scratchpads*, Memórias Híbridas, Consumo Energético, Desempenho.

Abstract

SILVA, Lisandro L. **Hybrid Memory Architecture Exploration for Embedded Systems using Non-Volatile Memories.** 2017. 131f. Dissertation (Master Degree em Ciência da Computação). - Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2017.

Power consumption is as important as the performance in embedded systems battery powered, as increasingly these systems need to handle intensive computing with low power consumption. Due to the high contribution of memory access in the total power consumption of embedded systems, the memory architecture strongly influences the design objectives of the embedded devices. There are in the literature, various techniques of memory access optimization for embedded systems, allowing the system designer, a comprehensive design space exploration for the architecture of memory. Many of these techniques are proposed due to problems encountered with the advancement of technology, as for example, the traditional memory based on SRAM (Static Random Access Memory) on-chip has become a bottleneck in performance and energy consumption for the design of embedded systems, due to your high leakage and read latency. The emerging technologies of non-volatile memories (NVM, Non-Volatile Memories) as STT-RAM (Spin-Transfer Torque RAM) and PCRAM (Phase Change RAM), are candidate solutions for future memory systems because they have some advantages over traditional SRAM. This work presents explorations carried out in hybrid memories using non-volatile memories in embedded systems. First, the work displays a review of memory hierarchies, caches, scratchpads, the problems faced by embedded systems and new emerging technologies to these systems. Afterward, the work presents the investigation of memory access of the embedded processor ARMv5, the analysis of the accesses performed, along with the impacts of the memory accesses in consumption and performance for different hybrid models of memories with emerging technologies, for a given set of benchmarks taken from the MiBench. The results are significant, because, for example, we were able to increase the number of accesses of the scratchpad memory (SPM) hybrid using memory SRAM 16 KB, STT-RAM 32 KB and PCM 8 KB (SPM 4) in approximately 0.99% when compared with a traditional 32 kb SRAM using SPM (SPM 1), in addition to reducing in average 23.81% the latency and 49.24% energy consumption and decrease the leakage by 46.65% and 16.29% area.

Keywords: *Scratchpads*, Hybrid Memories, Energy Consumption, Performance.

Lista de Figuras

Lista de Tabelas

LISTA DE ABREVIATURAS E SIGLAS

ABS	Anti-Lock Braking System
CPU	Central Processing Unit
DRAM	Dynamic Random-Access Memory
DVD	Digital Versatile Disc
DWM	Domain Wall Memory
EDP	Product Energy-*Delay*
eDRAM	Embedded Dynamic Random-Access Memory
FBDRAM	Floating Body Dynamic Random-Access Memory
FRAM/FeRAM	Ferro-magnetic Random-Access Memory
GPS	Global Positioning System
HMC	Híbrido Memória Cubo
IPC	Instruções Por Ciclo
MLC	Multi-Level Cell
MP	Memória Principal
MRAM	Magnetoresistive Random-Access Memory
MTJ	Magnetic tunnel junction
NVM	Non-Volatile Memories
PCM	Phase Change Memory
PCRAM	Phase Change Random-Access Memory
PMTJ	Perpendicular Magnetic Tunnel Junction
p-STT-MRAM	Perpendicular Spin-Transfer Torque Magnetic Random-Access Memory
RAM	Random-Access Memory
RDRAM	Rambus Dynamic Random-Access Memory
RISC	Reduced Instruction Set Computer
RRAM/ReRAM	Resistive Random-Access Memory
SDRAM	Synchronous Dynamic Random-Access Memory
SLC	Single-Level Cell

SMC	Smart Memory Cube
SoC's	System-On-Chip
SPM	Scratchpad Memory
SRAM	Static Random-Access Memory
SSD	Solid-State Drive
STT-MRAM	Spin-Transfer Torque Magnetic Random-Access Memory
STT-RAM	Spin-Transfer Torque Random-Access Memory
VLIW	Very Long Instruction Word
VM	Volatile Memories
VWB	Very Wide Buffer

SUMÁRIO

1 INTRODUÇÃO

Nas últimas décadas, o crescimento da tecnologia tem se mostrado abrangente em diversas áreas, como, telecomunicações, eletrônica de consumo, meios de transporte, equipamentos médicos, entretenimento, entre outras. Esse crescimento tecnológico trouxe consigo o uso de sistemas computacionais, os quais são responsáveis por facilitar o cotidiano das pessoas. Os sistemas computacionais podem ser encontrados em diversas aplicações, como exemplificação tem-se o avanço dos aparelhos celulares, os quais acabam se tornando cada vez mais complexos, possuindo diversas outras funcionalidades se comparados com aparelhos celulares de poucos anos atrás. Houve um aumento significativo da convergência de diversas funcionalidades em um único aparelho. Os primeiros aparelhos telefônicos móveis tinham a função de somente realizarem ligações, com o passar do tempo, os mesmos foram ganhando mais funcionalidades, como por exemplo, a inserção de câmera digital, MP3 player, e principalmente o acesso à internet. Os sistemas computacionais quando utilizados em sistemas ainda mais complexos, são chamados de sistemas embarcados (MARWEDEL, 2011).

A diferença entre um computador de uso geral para um sistema embarcado é que, um computador de uso geral é concebido para ser flexível e executar uma vasta gama de aplicações diferentes, enquanto que sistemas embarcados são utilizados para controlar uma determinada aplicação (GAJSKI, et al. 2009). Um sistema embarcado realiza um conjunto de tarefas predefinidas, geralmente com requisitos específicos, pois o sistema é dedicado a tarefas específicas. Também são projetados para executar funções dedicadas muitas vezes com restrições de computação em tempo real. Atualmente, as aplicações desses sistemas tornam-se cada vez mais complexas, mesmo com o aumento da capacidade de processamento dos sistemas digitais modernos, contudo, o tempo para comercialização (*time-to-market*) torna-se cada vez menor.

Os sistemas embarcados possuem características que os diferem dos demais sistemas, como por exemplo, confiabilidade, custo, restrições de tempo real, tamanho do código, desempenho, baixo consumo energético, limitações físicas

(tamanho e peso), devem ser consideradas no desenvolvimento destes sistemas. Dentre os desafios encontrados no projeto destes sistemas, destaca-se a preocupação com o consumo energético e desempenho. Estes requisitos têm motivado a investigação de técnicas de otimização que diminuem o consumo energético sem que ocorra uma degradação do desempenho original, ou que aumente o desempenho sem que ocorra um grande aumento do consumo energético do sistema. A hierarquia de memória influência fortemente o desempenho e consumo energético, pois, de acordo com a hierarquia empregada, o tempo e o consumo energético do acesso ao endereço possuirá resultados diferentes dependendo do nível da hierarquia, do tipo e do tamanho da memória empregada que o endereço está contido.

Muitas dessas técnicas utilizam memórias não voláteis (NVM, *Non-Volatile Memories*) como uma forma de otimização, visto que essas memórias emergentes possuem algumas vantagens em relação as memórias tradicionais DRAM (*Dynamic Random-Access Memory*) e SRAM (*Static Random-Access Memory*), como observado nos trabalhos de Wang et al. (2012), Li et al. (2012) e Hu et al. (2013), onde estes trabalhos admitem que a memória principal NVM pode alcançar uma economia energética expressiva com o desempenho comparável ao de uma memória principal tradicional DRAM. Como alguns exemplos de memórias emergentes podemos citar a PCM (*Phase Change Memory*) ou PCRAM (*Phase Change Random-Access Memory*), a MRAM (*Magnetoresistive Random-Access Memory*), a STT-MRAM (*Spin-Transfer Torque Magnetic Random-Access Memory*) também chamada de STT-RAM (*Spin-Transfer Torque Random-Access Memory*), a RRAM/ReRAM (*Resistive Random-Access Memory*), a FRAM/FeRAM (*Ferro-magnetic Random-Access Memory*) e a DWM (*Domain Wall Memory*).

As vendas de NVM ainda são moderadas, segundo a empresa de consultoria Développement (2016), o mercado emergente de NVM em 2015 é consideravelmente mais baixo do que os negócios voláteis dominantes da DRAM e da memória *flash* não-volátil, que possuem receitas combinadas de quase 80 bilhões de dólares, enquanto que as vendas de NVMs representam cerca de 53 milhões de dólares em 2015. Entretanto, a mesma empresa prevê que o mercado NVM emergente mundial subirá de 53 milhões de dólares em 2015, para 4.6 bilhões dólares até 2021. Apresentando assim, um impressionante crescimento maior que 110% ao ano. Sob esta nova análise, a empresa de consultoria descreve como as

tecnologias NVM emergentes serão cada vez mais utilizadas em vários mercados. Os principais segmentos do mercado que foram identificados pelos analistas da empresa são apresentados na Figura 1, juntamente com o tipo de NVM utilizada e as empresas investidoras.

Diversos sistemas embarcados possuem em seu interior um processador ARM. Esse processador tinha mais de 95% de participação no mercado de *smartphones*, 10% em computadores móveis, 35% em TVs digitais em 2010 (MORGAN, 2011). A partir do ano de 2014, a empresa ultrapassou a marca dos 50 bilhões de chips produzidos, 10 bilhões foram produzidos somente em 2013 (SHIMPI, 2014). Na Figura 2 é observado o número de unidades vendidas de chips contendo núcleos ARM entre os anos de 1997 e 2015. Assim, a arquitetura que será analisada neste trabalho é a arquitetura ARM, devido ao fato da sua grande utilização em sistemas embarcados.

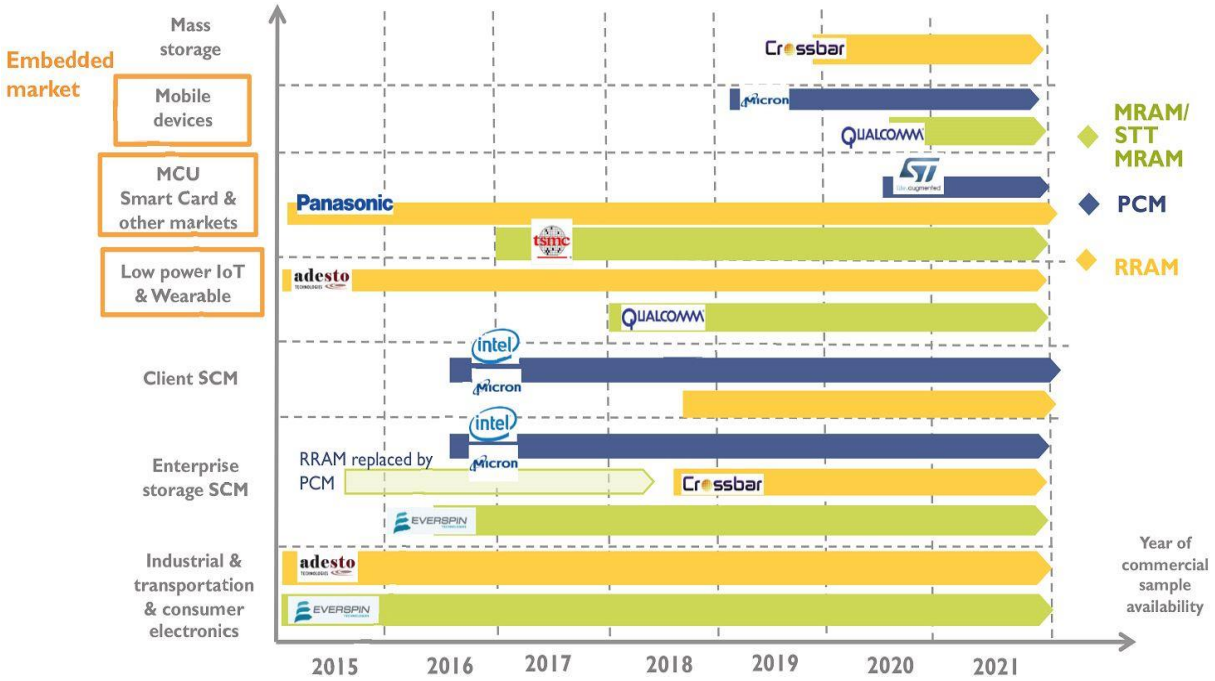


Figura 1 – Segmentos do mercado para NVM.
Fonte: DÉVELOPPEMENT, 2016.

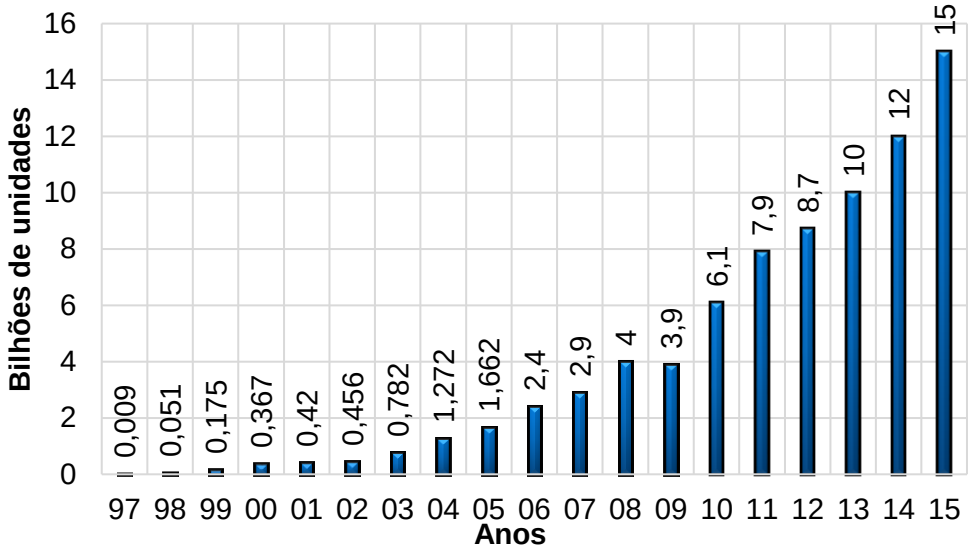


Figura 2 – Vendas de chips contendo núcleos ARM
Fonte: Adaptado de ARM (2017), HEXUS (2015) e BEDDARD (2016).

1.1 Motivação

Os sistemas embarcados caracterizam-se por apresentar muitas restrições e requisitos, como citado anteriormente. De um modo geral a descrição dos requisitos funcionais não é suficiente para o projeto de um sistema embarcado, devendo ser considerados também requisitos não-funcionais, tais como desempenho, custo, consumo de energia, tamanho físico e peso (WOLF, 2012). Estas e outras características têm um forte impacto no projeto destes sistemas, tanto no projeto do *hardware* como no projeto do *software*.

Segundo Hennessy e Patterson (2011), em um sistema de computador moderno o gargalo dominante na obtenção de alto desempenho e eficiência energética é a distância tecnológica entre o desempenho do processador e da memória tradicional, como demonstrado na Figura 3. Esta distância torna-se mais significativa em sistemas embarcados. Segundo Wolf e Kandemir (2003), o sistema de memória é um dos principais fatores de desempenho e consumo energético, especialmente nos sistemas embarcados que utilizam bateria. Como dito anteriormente, o projeto de sistemas embarcados apresenta muitas restrições e requisitos rígidos. Sendo assim, os projetistas possuem limitações em suas decisões, por causa desses requisitos não-funcionais extras, essas decisões não podem considerar apenas o desempenho, mas também o consumo energético dos dispositivos embarcados.

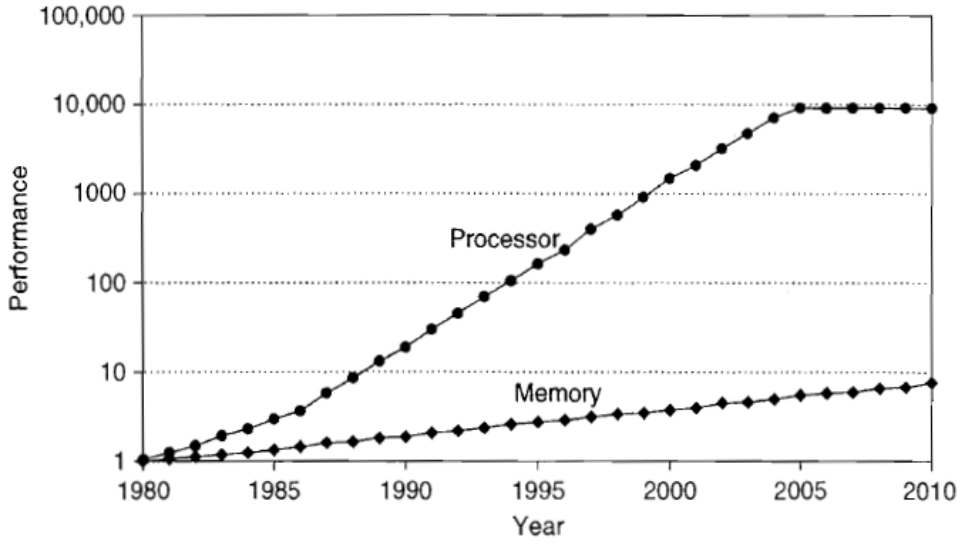


Figura 3 – Gap de desempenho existente entre Memória e Processador.
Fonte: HENNESSY e PATTERSON, 2011.

Com a alta contribuição da energia gasta para o acesso à memória no consumo total da mesma em sistemas embarcados, a arquitetura de memória possui uma forte influência sobre os objetivos do projeto em sistema embarcados. As memórias *caches* são normalmente utilizadas para melhorar o desempenho. Entretanto, o desempenho total do sistema e o consumo de energia são severamente relacionados com o tempo de acesso à memória e o consumo de energia média, o que faz com que a arquitetura de uma memória *cache* seja uma grande preocupação em projetos de processadores embarcados. Portanto, o projetista do sistema precisa realizar uma exploração do espaço de projeto abrangente para a arquitetura de memória.

Na literatura foram propostos vários tipos de otimizações em memória, como técnicas orientadas a loop, procedimento de *inlining* e o desenrolamento de laços (*loop unrolling*), como por exemplo no trabalho de Wolf e Kandemir (2003), as quais são exemplos de otimizações em *software*. Geralmente as abordagens tradicionais são otimizações de *hardware* ou de *software*, mas existem técnicas mistas que são relacionadas tanto ao *hardware* como ao *software*.

Visando explorar diferentes tecnologias, nos últimos anos diversos trabalhos estão utilizando NVMs com o intuito de melhorar os resultados das tradicionais DRAM e SRAM. Esses trabalhos mostram vantagens na utilização de memórias não voláteis em sistemas embarcados. Podemos citar os trabalhos de Wang et al. (2012 e 2013), Li et al. (2012), Hu et al. (2013) e Komalan et al. (2015), que utilizam algum tipo de NVM em seu desenvolvimento. O presente trabalho tem por motivação os

trabalhos de Wang et al. (2013) e Hu et al. (2013), em que os autores conseguiram diminuir o consumo energético e melhorar o desempenho através da comparação de abordagens híbridas de memórias com modelos tradicionais de memórias SRAMs, ambos os trabalhos utilizaram uma SPM (*Scratchpad Memory*) como um estudo de caso e apenas um tipo de memória emergente.

Sendo assim, o presente trabalho justifica-se pela necessidade de reduzir o consumo energético e aumentar o desempenho dos sistemas embarcados que utilizam a arquitetura ARM, através da utilização das novas tecnologias que estão surgindo no mercado de sistemas embarcados, as chamadas memórias emergentes, sendo de grande relevância para uma melhor eficiência energética e processamento desses sistemas.

1.2 Objetivos

Este trabalho tem como objetivo a exploração de arquiteturas de memórias híbridas para sistemas embarcados. Através dessa exploração busca-se, investigar o consumo energético e o desempenho nos acessos à memória para uma arquitetura ARM, explorando aspectos de hierarquia de memória, bem como as tecnologias de memória volátil e não volátil. O trabalho realiza experimentos comparando desempenho e consumo de soluções para SPM com memória volátil SRAM com soluções empregando modelos híbridos de memórias (SRAM, STT-RAM e PCM). Desta forma, este trabalho complementa os trabalhos existentes, pois, explora duas tecnologias de memória emergentes um único modelo de SPM.

1.3 Organização do Trabalho

O trabalho está dividido em seis capítulos. No segundo capítulo é realizada uma revisão dos aspectos dos dispositivos embarcados, suas principais características e restrições, além de apresentar uma revisão de hierarquias de memória, *caches* e *scratchpads.* Neste mesmo capítulo algumas tecnologias emergentes de memórias são revisadas, destacando suas respectivas características e as vantagens e desvantagens do uso de NVMs na hierarquia de memória.

No terceiro capítulo são revisadas as otimizações que podem ser realizadas, tanto em *hardware* quanto em *software*, ou em ambos para os sistemas de memória. Neste capítulo também são discutidos trabalhos relacionados que utilizam alguma dessas otimizações para a hierarquia de memória, incluindo trabalhos que exploram tecnologias emergentes.

No quarto capítulo é apresentada a metodologia adotada para a realização deste presente trabalho. Este capítulo apresenta as aplicações/*benchmarks* utilizados, a arquitetura avaliada e as ferramentas utilizadas para a realização do trabalho. No mesmo capítulo é apresentada a diferença entre as duas análises (estática e dinâmica) que são utilizadas ao decorrer do trabalho, com suas respectivas características, vantagens e desvantagens. Por fim, também são introduzidos os experimentos realizados neste trabalho.

No quinto capítulo são apresentados os resultados para os experimentos realizados, juntamente é realizada uma comparação dos resultados das abordagens estática e dinâmica. Neste capítulo, é possível identificar a influência das NVMs no consumo energético e no desempenho para os experimentos realizados. Por fim, o sexto capítulo apresenta as principais conclusões deste trabalho. Ainda neste capítulo são discutidos possíveis trabalhos futuros envolvendo NVMs em outros níveis hierárquicos de memória.

2 FUNDAMENTAÇÃO

Este capítulo apresenta conceitos e fundamentos importantes para a compreensão do trabalho. Primeiramente, o capítulo destaca características e restrições dos sistemas embarcados, além de revisar a hierarquia de memória em sistemas computacionais. Após, o capítulo discute algumas tecnologias emergentes de memórias.

2.1 Sistemas Embarcados

O desenvolvimento tecnológico dos últimos anos, como mencionado antes, trouxe consigo diversos aparelhos que auxiliam os seres humanos nas suas atividades diárias. Dentre estes equipamentos encontram-se: as máquinas que auxiliam nos afazeres domésticos, os aparelhos que auxiliam a medicina, podendo ajudar no controle dos pacientes ou até mesmo em cirurgias delicadas, os equipamentos que facilitam a comunicação das pessoas, ou para o entretenimento das mesmas, trazendo assim conforto e prazer para o seu dia-a-dia.

Segundo Marwedel (2011), sistema embarcado é um dispositivo no qual um sistema computacional é completamente encapsulado ou dedicado ao dispositivo ou equipamento que ele controla. Estes sistemas possuem uma funcionalidade restrita para atender uma tarefa específica em sistemas maiores nos quais estão inseridos. No mercado global 79% dos processadores utilizados são embarcados, certamente esse número deve ter sofrido um aumento significativo com os avanços dos anos, com a venda de dispositivos portáteis e produtos eletrônicos de consumo, como por exemplo, celulares, *smartphones*, televisores, câmeras digitais e DVD (*Digital Versatile Disc*). Estes processadores embarcados também podem ser encontrados em equipamentos maiores, como por exemplo, automóveis e aeronaves. Wolf (2012), destaca que um carro top de linha, ou seja, um carro com uma alta tecnologia, possui em torno de 100 processadores, enquanto um carro mais popular possui em média 60 processadores, representando assim, uma grande quantidade de eletrônica embarcada. Esses processadores são responsáveis pelo controle do

sistema de *air bag*, injeção eletrônica, ABS (*anti-lock braking system*), GPS, entre outros. No transporte aéreo são encontrados diversos dispositivos embarcados, presentes desde o controle da aeronave até os sistemas que possuem informações de cada passageiro. Muitos desses dispositivos embarcados precisam operar em tempo real para o correto funcionamento da aeronave, necessitam ser extremamente confiáveis, pois, são sistemas críticos que podem causar mortes e danos.

Com o avanço da tecnologia de semicondutores foi possível a integração de diversos sistemas mais complexos em um único chip (*system-on-chip – SoC's*), os mesmos serão cada vez mais necessários, devido ao fato do aumento da demanda de desempenho das diversas aplicações, como dispositivos portáteis, multimídia, entre outros. Os componentes de um *SoC* precisam ser específicos para cada domínio de aplicação, sendo que os elementos de processamento podem ser desde microcontroladores até microprocessadores, já por outro lado os componentes que fazem a comunicação dos elementos de controle podem ser *buffers*, memórias, barramentos, entre outros. Além disso, existe uma clara tendência dos sistemas embarcados serem implementados em *SoC's*, devido a integração de diversos componentes em um único chip, possibilitando assim a inserção de novas funcionalidades aos dispositivos portáteis. Essas novas funcionalidades agregadas aos dispositivos proporcionam um aumento significativo no consumo de energia, por isso, o mesmo tem sido enfrentado como um fator limitante às funcionalidades inseridas aos dispositivos alimentados por bateria.

Como visto, há uma diversa gama de aplicações onde sistemas embarcados estão envolvidos, e muitas delas possuem um grande número de funcionalidades que necessitam estar em um único dispositivo, um exemplo clássico é o celular de última geração ou *smartphones*. No mercado de *smartphones*, estão disponíveis diversos modelos de dispositivos com internet, câmera fotográfica, reprodutor de áudio e vídeo, *bluetooth*, entre outras funcionalidades. Outro exemplo é a eletrônica embarcada empregada atualmente em automóveis, como citado anteriormente, diversos processadores são responsáveis por controlar várias aplicações que estão inseridas nesses veículos, muitas dessas aplicações dependem dos resultados de diferentes sensores. Percebe-se assim, que os requisitos do sistema computacional por traz desses dispositivos são os mais diversos possíveis, portanto pode-se

afirmar que sistemas embarcados são naturalmente heterogêneos, pois são compostos de componentes de *hardware* (digital e analógico) e de *software*.

Em Shearer (2008), afirma-se que, a quantidade de recursos inseridos nos dispositivos portáteis teve um crescimento elevado se comparado com a tecnologia empregada na fabricação de baterias. Com isso, as funcionalidades agregadas consomem grande parte da energia fornecida pela bateria, diminuindo assim o tempo de utilização das mesmas. A Figura 4 mostra o aumento de recursos dos dispositivos embarcados em contraste com o consumo energético da bateria.

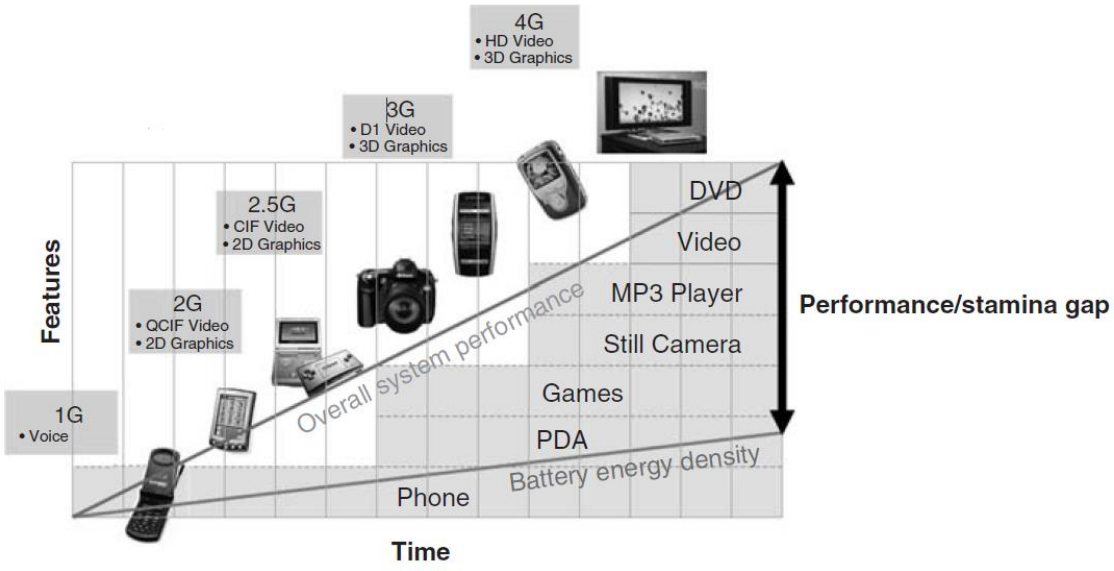


Figura 4 – Gap entre desempenho existente e energia.
Fonte: SHEARER, 2008.

Outro fator que chama a atenção é a exigência de dispositivos portáteis mais leves e menores, diminuindo assim o espaço destinado a bateria do dispositivo. Assim, um sofisticado gerenciamento de energia é a chave para aumentar o tempo de consumo das baterias, permitindo assim que o usuário do sistema disfrute ainda mais das funcionalidades oferecidas pelo dispositivo. Ou seja, o maior desafio encontrado é a realização de proporcionar o máximo desempenho em contraste com o mínimo gasto energético.

Os sistemas embarcados possuem muitas restrições e exigências, como por exemplo, baixo consumo de energia e potência, alto desempenho, tamanho e peso reduzidos. Os requisitos rígidos também são conhecidos como requisitos não funcionais, pois são frequentemente associados com o estado do sistema e não com as funcionalidades oferecidas como os requisitos funcionais. Estas e outras características possuem um forte impacto no projeto destes sistemas, tanto no ramo

de *software* quanto em *hardware*, no entanto, como destacado anteriormente, sistemas embarcados possuem uma vasta área de aplicações, onde cada uma pode ter características e requisitos diferentes. Como citado anteriormente em (WOLF, 2012) a descrição dos requisitos funcionais não é suficiente para o projeto de um sistema embarcado, devendo ser considerados também requisitos não-funcionais, tais como desempenho, custo, consumo de energia, tamanho físico e peso. Além das restrições físicas tradicionais, segundo Koopman (2007), sistemas embarcados são geralmente sistemas de segurança crítica, ou seja, a confiabilidade e a segurança são mais importantes do que o desempenho.

Algumas destas restrições são muito importantes no projeto de sistemas embarcados, tais como (MATTOS, 2007):

- Desempenho: com a crescente complexidade dos sistemas embarcados, a velocidade do sistema geralmente é requisito importante a ser alcançado;
- Tempo real: os sistemas embarcados geralmente estão envolvidos em tarefas onde os requisitos de tempo são rigorosos e devem responder a eventos externos, com uma avaliação precisa do tempo de execução;
- Dimensão física e peso: estes sistemas estão inseridos em sistemas maiores onde o requisito de portabilidade é muito importante, assim processadores embarcados devem ser pequenos e leves.
- Confiabilidade: alguns destes sistemas estão envolvidos em situações críticas onde os erros podem ter consequências dramáticas, envolvendo vidas humanas e enormes quantias de dinheiro;
- Tempo e custo de projeto: atualmente, o tempo de colocação do produto no mercado é muito importante. Assim, o tempo e custo de projeto devem ser reduzidos tanto quanto possível.
- Baixo consumo de energia: o consumo de energia apresenta uma questão crítica especialmente em dispositivos portáteis, pois estes utilizam bateria.

Como alegado anteriormente, sistemas embarcados são dedicados para uma determinada aplicação ao contrário dos dispositivos convencionais. No projeto de *hardware* para sistemas embarcados os fatores mais interessantes para serem

considerados são, o desempenho, custo (projeto e fabricação), tamanho físico, peso, consumo de energia e potência. Estes últimos são fatores limitantes na funcionalidade oferecida para alguns sistemas embarcados, principalmente os portáteis (SHEARER, 2008), que operam por bateria, como visto anteriormente, pois cada vez mais os dispositivos embarcados necessitam de mais funcionalidades, alto desempenho para um mesmo consumo energético.

2.2 Sistemas de Memória

Segundo Wolf e Kandemir (2003), a memória é um dos principais fatores de desempenho e consumo energético, portanto, o tamanho de memória é importante, principalmente nos dispositivos portáteis, visto que precisam possuir um tamanho pequeno de memória. Enquanto isso, os trabalhos de Catthoor et al. (2010), Verma e Marwedel (2007), e Hennessy e Patterson (2011) demostram que a arquitetura de memória possui uma relevância importante em circuitos integrados, pois influenciam na área e no consumo energético do circuito. Como descrito por Catthoor et al. (2010), estas arquiteturas representam cerca de 40% a 60% do total de energia consumida por um processador embarcado para um conjunto de instruções.

Em sistemas de computadores modernos, o principal empecilho para a obtenção de alto desempenho e eficiência energética, é a diferença tecnológica existente entre o desempenho do processador e da memória tradicional (HENNESSY e PATTERSON 2011), que é ainda mais significativa em sistemas embarcados. Enquanto a velocidade da memória DRAM está aumentando em um fator de 1,07 por ano, as velocidades dos processadores possuem um fator de aumento de 1,5 a 2 por ano (MARWEDEL, 2011). Atualmente, as velocidades dos processadores em relação as velocidades das memórias ainda estão aumentando, entretanto, essa diferença se comparada aos anos anteriores não possui o mesmo crescimento (Figura 3). Apesar disso a diferença de desempenho entre o processador e a memória é cada vez maior, conforme demonstrado na Figura 5, onde é apresentado o crescimento da velocidade do processador em contraste com o crescimento da velocidade da memória, com o passar dos anos. Com isso, é importante a utilização de memórias menores e mais rápidas, que atuem como um *buffer* entre a memória principal e o processador. Estas novas memórias são

inseridas na hierarquia de memória com o intuito de melhorar o desempenho entre o processador e a memória principal.

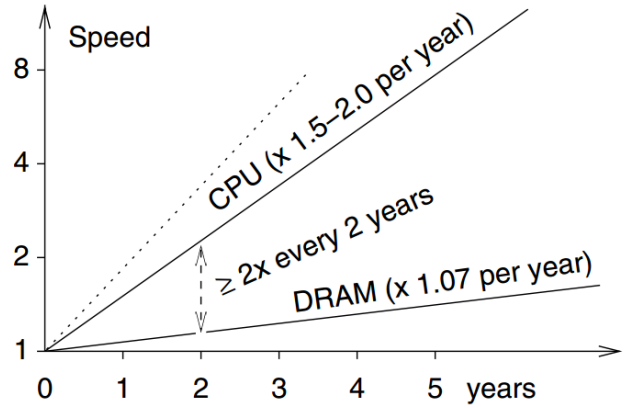


Figura 5 – Distância crescente entre as velocidades do processador e da memória.
Fonte: MARWEDEL, 2011.

Na literatura existe um leque muito grande de diversos tipos de otimizações de memória, as mesmas estão focadas não só em *hardware* ou *software*, mas podem ser uma combinação das duas abordagens, *hardware* e *software* como mencionado antes.

Em Acevedo e Jimenez (2002), mostra-se a contribuição de cada tipo de instrução para o consumo de energia do sistema. Dado um programa qualquer, conforme a instrução é chamada, partes específicas do microprocessador são ativadas, ou seja, a escolha correta da instrução poderia gerar uma redução do consumo de energia.

Os dados, informações, endereços e configurações devem ser armazenados em algum tipo de memória. No entanto, isto deve ser feito de alguma forma eficiente, para que a latência da busca do dado pelo processador seja a mínima possível, ou seja, o desempenho da busca deve ser o maior possível. Para melhorar o sistema original, realiza-se otimizações em código, ou até mesmo melhorias na hierarquia de memória. Possuindo assim resultados melhores em tempo de execução e eficiência energética, pois grandes memórias requerem mais energia para a busca do dado requerido pelo processador, além de demorarem mais tempo do que as memórias menores. A Figura 6 ilustra o tempo gasto para a busca de uma informação na memória, juntamente com a energia consumida, quando varia-se o tamanho de uma memória *cache* formada por SRAM. Pelo gráfico da Figura 6, percebe-se que a partir do ponto de 8k bytes haverá um aumento significativo no consumo energético, devido a fatores limitantes na tecnologia empregada para as memórias.

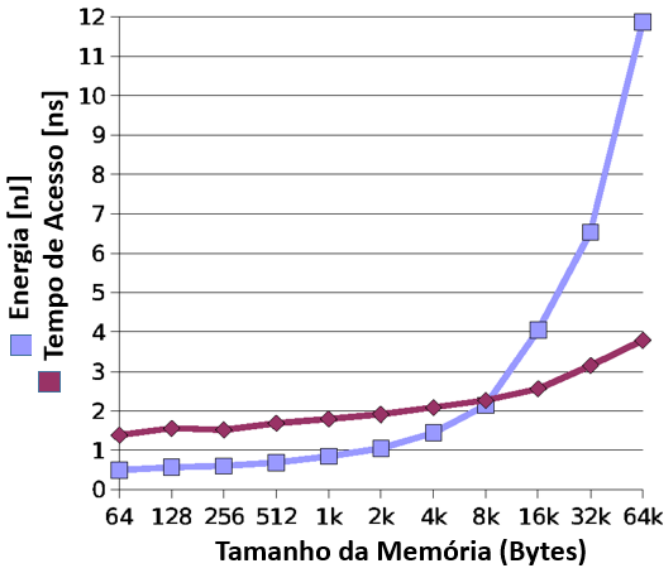


Figura 6 – Comportamento energético e tempo gasto em acesso a memória conforme o tamanho da mesma aumenta.
Fonte: MARWEDEL, 2014.

Existem praticamente dois tipos de memórias, as memórias não voláteis (NVM) e as memórias voláteis (VM). Como o próprio nome diz as memórias voláteis são aquelas que após o desligamento do sistema, ou seja, retirada a fonte de energia, perdem os dados armazenados, isto não acontece nas memórias não voláteis, onde quando é retirada a alimentação elas permanecem com os dados guardados. As memórias não voláteis são implementadas em muitos dispositivos embarcados, pois possuem características desejáveis, como por exemplo, tolerância a radiação, economia de energia e armazenamento dos dados garantidos, mesmo após o desligamento do aparelho (fonte de energia), sendo assim, não é necessária energia para manter os dados na memória, portanto, sua energia estática é muito pequena. Ainda as memórias não voláteis são mais densas do que as memórias voláteis, permitindo assim que, para uma mesma área de memória, os projetistas coloquem na memória não volátil uma quantidade maior de dados/informações.

2.2.1 Hierarquia de Memória

Os principais motivos para explorar o nível hierárquico das memórias é reduzir a energia gasta por acesso e o tempo do acesso à memória. Segundo Marwedel (2011), memórias maiores gastam mais energia por acesso e são mais lentas do que memórias menores. Com isso, busca-se obter uma boa eficiência

energética com um bom tempo de execução através da exploração hierárquica de memórias. Entretanto, o uso de memórias *caches* provocam alguns problemas para aplicações embarcadas, principalmente nas questões de previsibilidade. Além disso, as memórias rápidas possuem um custo elevado se comparado com memórias lentas, graças a isso uma hierarquia de memória é organizada em vários níveis, onde as memórias que possuem alta velocidade ficam nos níveis mais altos de processamento, e a cada nível abaixo, a memória é mais lenta. Sendo assim, tem-se no topo da hierarquia, perto do processamento, memórias de alta velocidade, entretanto com tamanho reduzido e de alto custo, e após, em cada nível desta hierarquia tem-se memórias maiores e custos decrescentes, entretanto possuem sua velocidade reduzida.

O conjunto de dados disponíveis na hierarquia mais alta é geralmente um subconjunto dos dados do nível inferior, e assim por diante. Desta forma, o objetivo é esconder as latências da memória não volátil e obter um sistema de alta velocidade de memória, através dos níveis utilizados na hierarquia de memória, com o menor custo possível, ou seja, busca-se um sistema de memória com a velocidade do nível superior possuindo um custo tão pequeno quanto o nível inferior. Os níveis de hierarquia de memória são implementados utilizando três tecnologias atualmente.

Primeiramente são encontradas as memórias que estão mais próximas ao processador, elas são chamadas de *caches*, utilizam a tecnologia de memória estática de acesso aleatório (SRAM). A tecnologia envolvida possui elevado desempenho para leituras e gravações, entretanto, é a tecnologia mais cara de toda a hierarquia de memória. Sendo assim, possui sua utilização em pequenos tamanhos, seguindo a tendência do mercado que quanto mais rápida, mais cara e menor. Além disso, esta memória é volátil, ou seja, não retém os dados quando desligada, necessitando permanecer sempre ligada para que seus dados sejam mantidos.

Posteriormente em um nível abaixo é encontrado as memórias principais de trabalho que utilizam a tecnologia DRAM, elas são maiores, mais lentas e mais baratas quando comparadas com as memórias *caches*. A memória do tipo DRAM, assim como a SRAM, precisa permanecer ligada para preservar seus dados.

O último nível de memória, conhecido como memória secundária, utiliza tecnologias de armazenamento não voláteis, ou seja, após o desligamento do sistema a memória irá manter os seus respectivos dados. Possuem um tamanho

total de armazenamento elevado, pois guardam uma grande quantidade de informações do sistema. Se faz necessário o emprego de uma memória deste tipo em pelo menos um nível da hierarquia de memória. As memórias deste nível geralmente utilizam meios magnéticos que são escritos e gravados por dispositivos mecânicos, esses dispositivos possuem um péssimo desempenho. Alternativamente, o último nível pode empregar memórias do tipo SSD (*solid-state drive*), elas são mais rápidas quando comparadas com os meios magnéticos, entretanto, os SSD's possuem custos mais elevados.

Os principais elementos que compõem uma hierarquia de memória são os registradores, a memória *cache*, a memória principal e a memória secundária. A Figura 7 apresenta uma ilustração com diferentes níveis de hierarquia de memória e seus respectivos tempos de acesso e tamanho físico.

O conceito de hierarquia de memória explora o princípio da localidade (HENNESSY e PATTERSON, 2011). Este princípio ocorre no tempo (localidade temporal) e no espaço (localidade espacial). O princípio pode ser visto como a necessidade do processador enxergar somente uma parte da memória durante a execução de uma aplicação, essa parte é referente aos dados e as instruções necessárias para a computação da tarefa ou aplicação. Este princípio ainda pode ser explicado, como sendo o comportamento dos programas em repetir trechos de código utilizando dados repetidos ou acessar repetidamente dados próximos aos que estão sendo utilizados. Sendo assim, o principal objetivo da hierarquia de memória é melhorar o desempenho.

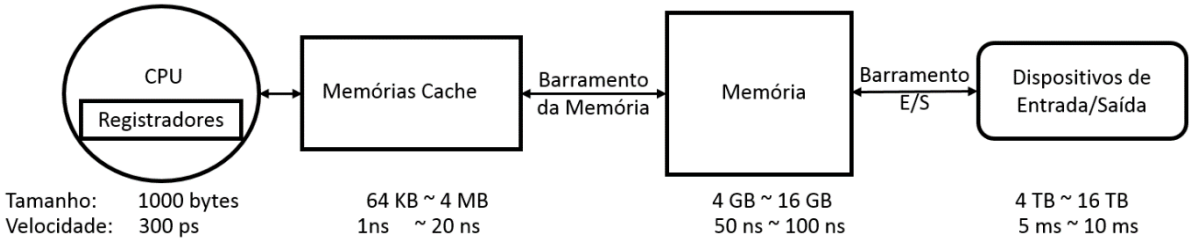


Figura 7 – Diferentes níveis da hierarquia de memória em computadores embarcados, desktop e servidores.
Fonte: HENNESSY e PATTERSON, 2011.

O princípio de localidade pode ser classificado em dois tipos diferentes:

- Localidade Temporal: como o próprio nome diz, essa localidade se refere à necessidade temporal dos dados, ou seja, se um item é

referenciado, a probabilidade dele ser referenciado novamente é elevada. Sendo assim, existe a necessidade de manter os dados recentemente acessados na hierarquia que está mais próxima ao processador.

- Localidade Espacial: a mesma é referente ao local de armazenamento da memória, ou seja, se um item é referenciado, os itens de endereços próximos possuem a probabilidade de serem referenciados em breve. Sendo assim, existe a necessidade de não apenas trazer para a hierarquia mais próxima ao processador os dados referenciados, mas sim o bloco de dados, que contém os dados de endereços próximos.

2.2.2 Caches

São chamadas de memórias *caches*, as hierarquias de memórias que estão localizadas entre o processador e a memória principal de trabalho, ou seja, geralmente são memórias que estão perto dos processadores. Atualmente os computadores de uso geral possuem pelo menos um nível de hierarquia de *cache*.

O funcionamento das memórias *caches* pode ser definido em dois passos, dado um endereço de acesso, que foi gerado pelo processador para a busca de algum dado na memória. Primeiramente a *cache* irá verificar se possui uma cópia da posição de memória solicitada do processador, se possuir, ela verificará a posição correspondente dentro da *cache* que contenha o endereço solicitado e o retornará ao processador, caso contrário, se a *cache*, não possuir o endereço solicitado, ela irá trazer o conteúdo da memória principal e escolherá uma posição onde a cópia do dado será armazenado, e retornará uma cópia ao processador.

2.2.3 Scratchpads

Uma memória *scratchpad* (SPM) é um *array* de memória com a parte de lógica de decodificação (BANAKAR, et al., 2002). A ideia por traz de uma *scratchpad* é manter objetos de memória mapeados pelo compilador no seu último estágio. A SPM ocupa uma parte do espaço de endereçamento e o restante é ocupado pela memória principal.

A grande vantagem das SPM's é que não existe necessidade de checagem da disponibilidade do dado ou instrução como nas memórias *caches*, eliminando o emprego de rótulos (*tags*) e comparadores existentes nas *caches*. Isso contribui significativamente na redução da área e consumo de energia. Segundo Banakar et al. (2002) uma cache utilizando memória SRAM normalmente possui um consumo energético entre 25% a 45% do consumo total do chip. Os autores utilizaram uma memória *scratchpad* com o objetivo de reduzir o consumo de energia e de área, os resultados do trabalho de Banakar et al. (2002), apresentaram uma redução de 40% no consumo energético e uma redução em média de 46% na área. Assim, a memória *scratchpad*, tem sido amplamente adotada em muitos sistemas embarcados devido a sua menor área e menor consumo energético. Além disso, a *scratchpad* pode proporcionar muitas vezes uma melhor previsibilidade e sincronização em dispositivos de tempo real, por ser uma memória gerenciada por software (LI, GAO e XUE, 2005).

A memória *scratchpad* apesar de ser uma técnica de otimização de memória por *hardware*, pode possuir também técnicas com abordagens em *softwares*. O trabalho de (LI, GAO e XUE, 2005) propõe uma metodologia para o problema de gerenciamento de memória *scratchpad*, o qual pode ser resolvido por um algoritmo de coloração de grafos para a alocação de registradores.

Uma *scratchpad* pode conter instruções ou dados alocados em seu interior. A Figura 8 demonstra as informações que podem ser alocadas em uma *scratchpad* em uma determinada hierarquia de memória.

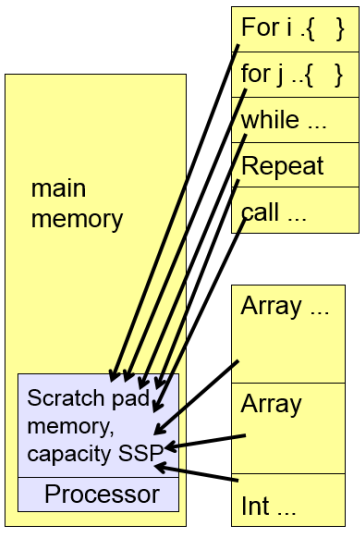


Figura 8 – Exemplo de alocação de dados/instruções em uma *scratchpad*.
Fonte: MARWEDEL, 2014.

A Figura 9 demonstra um sistema embarcado genérico com as memórias *on-chip* e *off-chip*. O sistema de memória principal pode estar contido parcialmente dentro do chip (*on-chip*) e parcialmente fora do chip (*off-chip*). A maioria dos sistemas embarcados utiliza somente um único nível de *cache*, como observado na Figura 9, porém com a necessidade de aumento de desempenho, podem existir outros níveis de *cache*.

Pela Figura 9, as memórias *scratchpads* (SPM) podem ser utilizadas juntamente com memórias *caches*, como memórias de alta velocidade *on-chip*. O tipo tradicional de memória empregada nas SPMs e *caches* são as memórias SRAMs. Na forma *off-chip*, memórias de tecnologias diversas podem ser utilizadas, inclusive memórias SDRAM (*Synchronous Dynamic Random-Access Memory*) ou RDRAM (*Rambus Dynamic Random-Access Memory*). As memórias *on-chip* e *off-chip* influenciam o desempenho e o consumo de energia em sistemas embarcados.

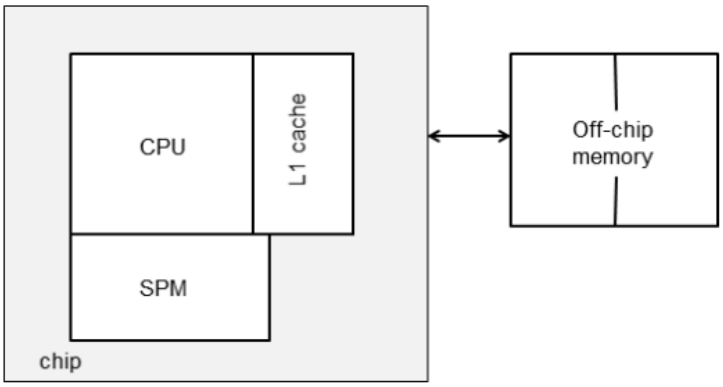


Figura 9 – Vista centrada em memória de um sistema computacional embarcado.
Fonte: WOLF e KANDEMIR, 2003.

2.3 Tecnologias Emergentes em Memórias

Com o avanço da tecnologia, a memória tradicional baseado em SRAM *on-chip* tornou-se um gargalo para o projeto de eficiência energética em sistemas embarcados, devido ao seu alto *leakage* (corrente de fuga) e tempo de acesso. As tecnologias emergentes de memória não volátil, tal como STT-RAM, PCRAM, RRAM e DWM são as possíveis soluções para os sistemas de memória futuras.

No trabalho de Mittal, Vetter e Li (2015), os autores realizam um breve resumo das tecnologias de memória e classificaram diversos trabalhos que utilizam essas

memórias em seus experimentos, na Seção 2.3.1 encontra-se um resumo destas tecnologias juntamente com as suas características e os desafios na sua utilização.

2.3.1 Embedded DRAM

Embedded DRAM, ou (eDRAM, do inglês, *Embedded Dynamic Random-Access Memory*) é uma RAM dinâmica baseada em um capacitor que pode ser integrado no mesmo molde que o processador. Este modelo pode conter um capacitor e um transistor (1T1C) convencional DRAM na célula, ou uma lógica compatível de ganho celular (CHANG, et al., 2013) e (AGRAWAL, et al., 2013), ambos usam alguma forma de capacitor para armazenar os dados. Como a carga armazenada vaza gradualmente, a atualização periódica é necessária para preservar seu valor e evitar a decadência.

A exigência da atualização periódica é dos maiores obstáculos no uso de eDRAM. Estas memórias empregam transistores de lógica rápidos, que possuem um *leakage* maior do que a DRAM convencional e, portanto, sua exigência de atualização também é maior do que da DRAM. Em Barth et al. (2008) os autores relatam que o período de retenção da eDRAM é de 40 ms, em comparação com o período de retenção de 64ms de uma DRAM tradicional (WILKERSON, et al., 2010). Além disso, o período de retenção da informação reduz ainda mais devido a fatores como o aumento da temperatura, processo de variação e dimensionamento da tecnologia (CHANG, et al., 2013) e (ALIZADEH, et al., 2012). Atualizar as operações consome uma fração significativa do consumo energético da *cache*, além de interferir no acesso da *cache* e reduzir a disponibilidade da *cache*. Além disso, para pequenos períodos de retenção e *caches* de grandes dimensões, um número muito grande de blocos precisa ser atualizado em uma pequena quantidade de tempo. A eDRAM é criticada por possuir desafios de escalabilidade devido à dificuldade de colocação de carga precisa e dados de sensoriamento. Por fim, uma memória *cache* de 4MB empregando eDRAM possui como latência de leitura/gravação um intervalo entre 3 e 5 nanosegundos (CHANG, et al., 2013) e (CACTI, 2014).

2.3.2 STT-RAM

Segundo Smullen et al. (2011), a STT-RAM utiliza uma junção de túnel magnético (*Magnetic Tunnel Junction,* MTJ) como armazenamento de memória. A

Figura 10 exibe uma célula STT-RAM, nota-se pela figura que um transistor de acesso é ligado a um elemento de memória implementado usando um MTJ, que contém duas camadas ferromagnéticas separadas por uma camada isoladora de óxido. A direção da magnetização de uma camada ferromagnética é fixa enquanto que a outra camada ferromagnética pode ser alterada pela passagem de uma corrente. A resistência do MTJ é determinada pela direção de magnetização relativa dessas duas camadas. Se as duas camadas têm direções diferentes, a resistência do MTJ é alta e vice-versa. Usando essa propriedade, um valor binário é armazenado em uma célula STT-RAM. Para ler o valor armazenado, uma pequena tensão é aplicada entre os terminais MTJ. A corrente que flui através do dispositivo é detectada, e o estado de magnetização é determinado como um resultado.

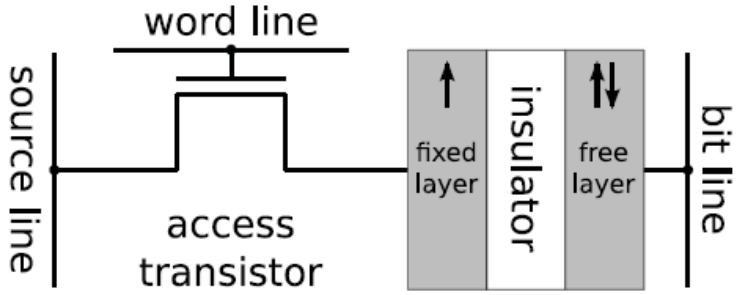


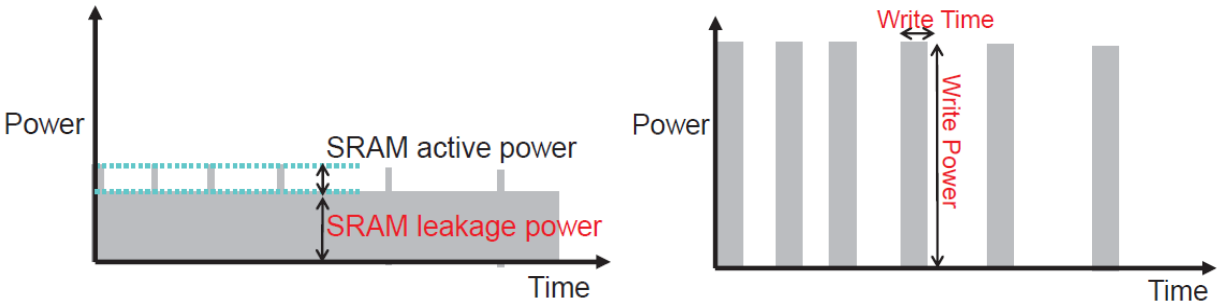
Figura 10 – Célula STT-RAM.
Fonte: RODRÍGUEZ, TOURIÑO e KANDEMIR 2014.

Embora a STT-RAM possui densidade menor do que a PCM e a RRAM, e maior latência e energia para a operação de escrita do que a SRAM, essa memória foi amplamente utilizada para a concepção de *caches* devido a seu alto *endurance*. O *endurance* é o número de ciclos de gravações que podem ser aplicados a um bloco de memória *flash*, antes que a mídia de armazenamento torne-se inconfiável. No entanto, apesar de um valor de *endurance* de 10^{15} foi estimado, o melhor resultado do teste de *endurance* até agora é inferior a 4×10^{12} (HUAI, 2008). Outra vantagem da STT-RAM é que a sua não-volatilidade pode ser negociada para melhorar sua energia de gravação e latência.

Em Smullen et al. (2011), os autores modificam o tempo de retenção encolhendo a área planar do MTJ, enquanto que o trabalho de Jog et al. (2012) consegue isto diminuindo a espessura da camada livre e baixando a saturação da magnetização que reduz a barreira térmica do MTJ. Como exemplo, Jog et al. (2012)

mostra que para a frequência de 2 GHz, os valores de latência de gravação de uma STT-RAM de 4 MB para períodos de retenção de 10 anos, 1 segundo e 10 milissegundos são respectivamente 22, 12 e 6 ciclos. Assim, com base na característica da aplicação e no nível da hierarquia da *cache*, um *designer* pode escolher um valor apropriado de período de retenção.

O modelo de memória p-STT-RAM (*Perpendicular Spin-Transfer Torque Magnetic Random-Access Memory*) possui uma maior probabilidade para substituir a SRAM do que a STT-MRAM, pois a p-STT-RAM apresenta uma maior velocidade de acesso. Com os avanços da tecnologia CMOS, com a fabricação de transistores cada vez menores, o *leakage* da SRAM cresceu consideravelmente, tornando-se uma parte significativa do consumo energético total em diversos chips de semicondutores. A Figura 11 (a) mostra a energia típica de uma SRAM durante o estado ativo, indicando que a energia total gasta por uma SRAM é predominantemente o *leakage*. Como mostrado na Figura 11 (b), idealmente, não há *leakage* para STT-MRAM. No entanto, para reduzir a energia total (energia total = ativa + *leakage*, na Figura 11), substituindo uma SRAM por uma p-STT-MRAM, o ponto mais importante é a redução no consumo energético e/ou a redução da latência para operações de escrita da STT-MRAM.



(a) SRAM convencional. (b) p-STT-MRAM.

Figura 11 – Comparação da potência com o tempo para SRAM e STTMRAM durante o estado ativo.
Fonte: FUJITA et al., 2015.

2.3.3 RRAM

Uma RRAM com comutação unipolar usa um dielétrico isolador (LI e CHEN, 2009). Quando é aplicada uma tensão suficientemente elevada, é formado um filamento ou um percurso condutor no dielétrico isolador. Depois disso, aplicando

tensões adequadas, o filamento pode ser ajustado para *set* (o que leva a uma baixa resistência) ou *reset* (o que leva a uma alta resistência).

Comparado a SRAM, uma cache RRAM tem alta densidade, latência de leitura comparável e possui um valor muito inferior de gasto energético referente ao *leakage*. No entanto, as limitações da RRAM são a seu baixo *endurance*, cerca de 10^{11} (KIM, et al., 2011), alta latência e consumo energético para as operações de escrita. Por exemplo, uma *cache* típica de 4 MB de RRAM tem uma latência de leitura entre 6 e 8 nanosegundos e uma latência de escrita entre 20 e 30 nanosegundos (DONG et al., 2012).

2.3.4 PCM

A PCM usa um material de mudança de fase chamado GST, que é uma liga de germânio, antimônio e telúrio. Quando a liga é aquecida a uma temperatura muito alta e rapidamente resfriada, transita em uma substância amorfa com alta resistência elétrica (*reset*). Por outro lado, quando a liga é aquecida a uma temperatura entre a cristalização e o ponto de fusão e resfriada lentamente, cristaliza até um estado físico com menor resistência (*set*). A Figura 12 ilustra as operações *set* e *reset.* Na Figura 12 para a operação *set*, quando o GST é aquecido a uma temperatura entre a temperatura de cristalização (~ 300 ° C) e a temperatura de fusão (~ 600 ° C) durante um período de tempo, GST transforma-se no estado cristalino que corresponde a um Lógica '1'. Para a operação *reset*, quando o GST é aquecido acima do ponto de fusão e extinguido rapidamente, GST transforma-se no estado amorfo que corresponde ao '0' lógico. Esta propriedade física é usada para armazenar um valor binário em uma célula PCM.

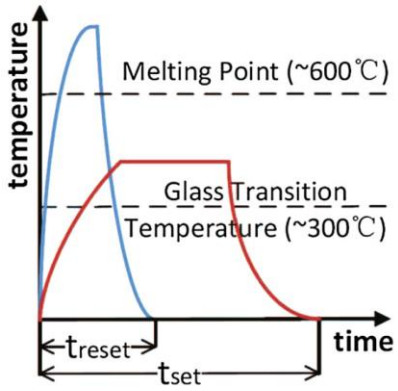


Figura 12 – As operações *set* e *reset* da PCM.
Fonte WANG et al., 2015.

Em um chip PCM, as células são organizadas na matriz bidimensional tal como ilustrado na Figura 13. Nesta figura percebe-se a estrutura básica de uma célula da PCM, essa estrutura consiste em um transistor NMOS e um dispositivo de mudança de fase.

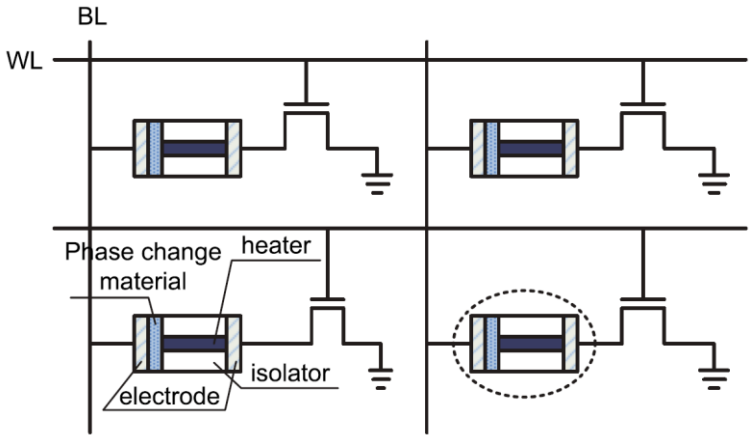


Figura 13 – Uma matriz de células PCM e a estrutura de uma célula PCM.
Fonte: WANG et al., 2015.

Os dois desafios mais graves no uso de PCM para projetar *caches on-chip* são sua resistência limitada de escrita e alta latência de gravação. Uma vez que o tráfego de escrita para uma *cache* é muito mais pesado do que para uma memória principal e a *endurance* da PCM é apenas perto de 10^8 escritas (WANG, et al., 2013) e (JOO, et al., 2010), para diversas aplicações, uma *cache* que utiliza PCM pode falhar em menos de uma hora. Uma típica *cache* de 4 MB utilizando PCM possui uma latência de leitura entre 15 e 20 nanosegundos e uma latência de escrita entre 150 e 170 nanosegundos (DONG et al., 2012) e (JOO, et al., 2010). Assim, a PCM é adequada para a memória principal ou hierarquias inferiores de *cache*, por exemplo, *cache* L3 ou mesmo *cache* L4 (WU, et al., 2009), onde a sua latência elevada pode ser tolerada e a alta densidade pode ser utilizada para evitar acessos fora do chip.

2.3.5 DWM

A DWM funciona controlando a parede do domínio (*Domain Wall*, DW) em nanofios ferromagnéticos (VENKATESAN, et al., 2012). O fio ferromagnético pode ter múltiplos domínios que são separados por paredes de domínio. Esses domínios podem ser individualmente programados para armazenar um único bit (na forma de

uma direção de magnetização) e assim, o DWM pode armazenar múltiplos bits por célula de memória.

Logicamente, uma macrocélula DWM aparece como uma fita, que armazena múltiplos bits e pode ser deslocada em qualquer direção. O desafio na utilização de DWM é que o tempo consumido no acesso a um bit depende da sua localização em relação à porta de acesso, o que leva a uma latência de acesso não uniforme e torna o desempenho dependente do número de operações de deslocamento necessárias por acesso. Comparado com outras NVMs, DWM é mais recente, e ainda está em fase de pesquisa e protótipos.

2.3.6 Comparativo entre as Tecnologias Emergentes

Ao contrário de memórias baseadas em carga, como DRAM, NVMs armazenam seus dados através da mudança do estado físico. Uma vez que uma operação de escrita para NVM envolve a mudança de seu estado físico, uma operação de gravação para NVM possui maior latência e consumo energético do que uma operação de leitura, levando a assimetria leitura-escrita (BISHNOI, et al., 2014a). Da mesma forma, a latência de escrita e o consumo energético da transição da lógica 1 para 0 é maior do que a de 0 para 1, levando a assimetria de escrita de 0/1 (BISHNOI, et al., 2014b). As NVMs também permitem o armazenamento de múltiplos bits de dados em uma única célula de memória. Isto é referido como armazenamento de células de múltiplos níveis (*Multi-Level Cell*, MLC) e conduz a um aumento significativo na densidade de armazenamento das NVMs.

Comparando com a memória tradicional SRAM, as memórias emergentes STT-RAM e PCRAM proporcionam um *leakage* menor e uma maior densidade. Comparando-as entre si, STT-RAM possui uma menor latência de acesso e potência dinâmica, enquanto PCRAM possui maior densidade. Segundo Wu et al. (2009) e Sun et al. (2009) a memória STT-RAM é mais adequada para memórias de último de nível, enquanto que a PCRAM é promissora como uma alternativa para DRAM na memória principal (LEE, et al., 2009).

Embora STT-RAM apresente muitas características atraentes, como o baixo *leakage* e alta densidade, esse tipo de memória possui alguns problemas. Ao contrário da SRAM, em que ler e escrever operações consomem o mesmo tempo e energia, uma operação de gravação de STT-RAM precisa de muito mais tempo de

latência e maior energia do que uma operação de leitura. Além disso, a latência e a energia de operações de escrita convencionais em uma STT-RAM são várias vezes maiores do que os da SRAM para um mesmo tamanho de memória. Como a memória *on-chip* é mais próxima à CPU na hierarquia de memória, reduzir a latência de acesso a memória *on-chip* é fundamental para o aumento do desempenho em sistemas embarcados.

Novos modelos de STT-RAM foram desenvolvidos para diminuir os problemas envolvidos com as operações de escrita. Segundo Khalili et al. (2011) e Tadisina et al. (2010), as PMTJ (*Perpendicular Magnetic Tunnel Junction*) foram desenvolvidas para alcançar uma baixa corrente de comutação, mantendo uma alta estabilidade térmica para as STT-RAM. Já em Xu et al. (2011), os autores conseguiram diminuir significativamente os problemas de gravação em STT-RAM SPM, graças a um cuidadoso processo de cootimização entre os dispositivos da arquitetura.

A Figura 14 exibe a capacidade, a velocidade dos acessos e o *endurance* de diversos modelos de memórias. Nesta figura nota-se modelos de memórias com acessos rápidos como SRAM, STT-RAM, p-STT-RAM e DRAM, com acessos moderados como PCM e ReRAM e com acessos lentos, como por exemplo, discos rígidos e memória flash. Observa-se que o modelo p-STT-RAM possui velocidade de acesso a memória superior ao modelo STT-RAM.

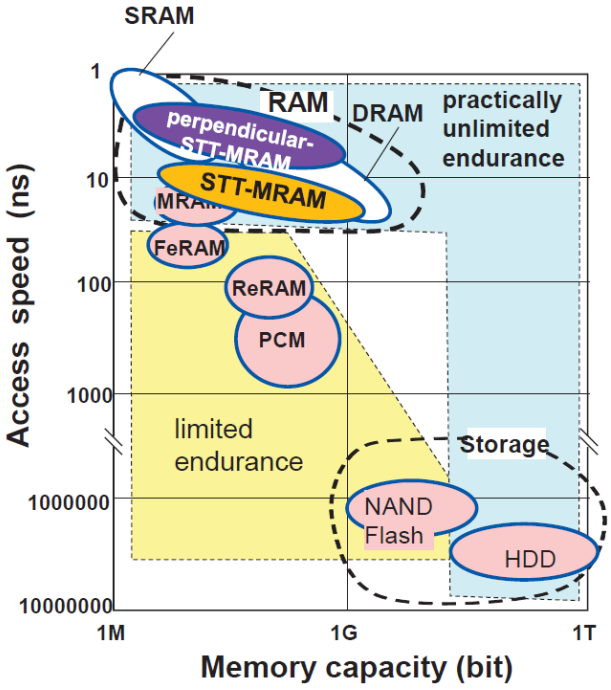


Figura 14 – Capacidade e velocidade de acesso de diversas memórias.
Fonte: FUJITA et al., 2015.

O trabalho de Mittal, Vetter e Li (2015), apresenta uma avaliação comparativa das propriedades das diferentes tecnologias de memórias (Tabela 1). Os valores apresentados podem não ser satisfatórios, entretanto com a pesquisa contínua, os valores das propriedades podem ser melhorados significativamente.

Tabela 1 – Características das diferentes tecnologias de memórias

	SRAM	EDRAM	STT-RAM	RRAM	PCM	DWM
Tamanho da Célula (F^2)	120 - 200	60 – 100	6 - 50	4 - 10	4 - 12	> 2
Endurance	10^{16}	10^{16}	4×10^{12}	10^{11}	10^8 - 10^9	10^{16}
Velocidade (L/E)	Muito Veloz	Veloz	Veloz/Lento	Veloz/Lento	Lento/Muito Lento	Veloz/Lento
Leakage	Alto	Médio	Baixo	Baixo	Baixo	Baixo
Energia Dinâmica (L/E)	Baixo	Médio	Baixo/Alto	Baixo/Alto	Médio/Alto	Baixo/Alto
Período de retenção	N/A	30 - 100 µs	N/A (Quando Relaxado)	N/A	N/A	N/A

Fonte: MITTAL, VETTER e LI, 2015.

O trabalho de Mittal e Vetter (2016) também discute propriedades aproximadas para diferentes tecnologias de memória (Tabela 2). Pode-se notar que estes valores devem ser considerados simbólicos, pois, futuras pesquisas podem levar a mudanças nestes parâmetros. Na Tabela 2, a granularidade de acesso refere-se à quantidade mínima de dados que são lidos / escritos em cada acesso e a *endurance* refere-se ao número de gravações que o bloco de memória pode suportar antes de se tornar não confiável. A propriedade comum a todos as NVMs é que sua latência / energia de escrita é significativamente maior do que a latência / energia de leitura. Além disso, em condições normais, as NVMs podem reter dados durante vários anos sem a necessidade de qualquer energia de *standby*.

Tabela 2 – Propriedades aproximadas em nível do dispositivo das tecnologias de memórias

	Tamanho da Célula	Granularidade de Acesso	Latência de Leitura	Latência de Escrita	Latência de Apagar	Endurance	Energia de Espera
HDD	N/A	512 B	5 ms	5 ms	N/A	$> 10^{15}$	1W
SLC Flash	4 – 6 F^2	4 KB	25 µs	500 µs	2 ms	10^4- 10^5	0
DRAM	6 – 10 F^2	64 B	50 ns	50 ns	N/A	$> 10^{15}$	Energia de atualização
PCM	4 – 12 F^2	64 B	50 ns	500 ns	N/A	10^8- 10^9	0
STT-RAM	6 – 50 F^2	64 B	10 ns	50 ns	N/A	$> 10^{15}$	0
ReRAM	4 – 10 F^2	64 B	10 ns	50 ns	N/A	10^{11}	0

Fonte: MITTAL e VETTER, 2016.

3 TRABALHOS RELACIONADOS

Este capítulo apresentará alguns trabalhos relacionados com hierarquia de memória em sistemas embarcados. Conforme citado anteriormente, os sistemas de memória são os principais responsáveis pelo desempenho e consumo de energia em sistemas embarcados. Assim, o consumo energético da memória assume uma importante relevância em sistemas embarcados que utilizam bateria.

Segundo Benini et al. (2003), para ter um projeto em memória com o objetivo em eficiência energética é necessário reduzir o custo de acessar a memória, respeitando o desempenho e as restrições impostas ao projeto. Uma forma é utilizar organizações hierárquicas para reduzir o gasto de energia nas memórias, permitindo assim acessos a bancos menores. Em Benini et al. (2003) uma taxonomia das abordagens de projeto em memória de baixa potência é apresentada. Essa taxonomia se divide em três grupos maiores, sendo eles, o *hardware* centralizado (objetivo principal é o *hardware*), *software* centralizado (objetivo central é o *software*), e o conjunto de *software* e *hardware*, onde ambas as abordagens são exploradas.

Dado um acesso à memória, na categoria onde o *hardware* é o central ponto de estudo, existe uma redução do custo do acesso, produzindo uma hierarquia personalizada de memória. No entanto, quando o *software* é o central ponto de estudo, é dada uma arquitetura de memória fixa, então modifica-se os requisitos de armazenamento e os padrões de acesso, para que se busque uma otimização da hierarquia empregada. No método onde ambos são estudados (*hardware* e *software*), temos que simultaneamente otimizar os custos e os padrões dos acessos na memória.

Segundo Benini et al. (2003), as otimizações em *hardware* foram classificadas nas seguintes abordagens:

- Técnica de exploração: como o próprio nome diz explora o fato de que o projeto do espaço de memória pode ser parametrizado e discretizado para permitir a busca exaustiva ou perto da mesma.

- A separação da memória: é uma solução típica orientada para o desempenho e redução de energia, pois reduz a latência ao acessar os blocos de memória menores, e a energia gasta pode ser reduzida para alguns padrões de acesso específicos.
- Extensão da hierarquia de memória: o particionamento da memória se estende ao tamanho da hierarquia de memória por divisão, com ou sem replicação.
- Abordagem orientada para desempenho: soluções arquiteturais para a otimização de desempenho são empregadas para ajudar na redução de energia. As técnicas que tentam reduzir a taxa de *cache-miss* pelo projeto de uma *cache* adequada, por exemplo, ao reduzirem o número de *cache-misses* diminuem também o custo do acesso dos dados na memória.
- Otimização de largura de banda: esta otimização está se tornando cada vez mais significativa para dispositivos embarcados, devido ao aumento de paralelismo em nível de instrução gerado pelos processadores superescalares ou VLIW (*Very Long Instruction Word*), e devido à densidade de integração que permite latências mais curtas.

Segundo Wolf e Kandemir (2003), há vantagens em otimizar o software para um sistema embarcado, pois pode-se ajustar o software para características detalhadas da arquitetura, como tamanho e a organização da cache. O software pode ser otimizado como um todo, ao invés de otimizar as partes de forma independente.

As otimizações de software realizadas por compiladores que visam melhorar a localidade de dados podem ser divididas em: otimizações que visam reduzir a latência de memória e as otimizações que visam esconder a latência da memória.

Primeiramente, estão as abordagens que alteram o padrão de acesso ao programa, *layout* de memória de variáveis, ou ambos. Várias aplicações em embarcados utilizam loops e matrizes, dentre as transformações mais conhecidas estão permutação de loop e fusão de loop, segundo Wolf e Kandemir (2003).

Posteriormente, estão as abordagens que tentam esconder as latências, em particular, o atraso na leitura de dados da memória. Dentre elas, a expansão em linha (*inline expansion*), é uma forma de otimização do compilador, a qual consiste no processo de substituição de uma chamada da sub-rotina ou função, com o corpo da sub-rotina ou função que está sendo chamada. Essa otimização normalmente

melhora o tempo e o espaço de utilização, ao custo de aumentar o tamanho final do programa, entretanto, em alguns casos pode diminuir o tempo de execução ou diminuir o tamanho final do programa.

Nos sistemas embarcados, *hardware* e *software* podem ser especialmente projetados e otimizados para aplicações específicas, sendo assim, para otimizar o consumo energético de um sistema, é preciso resolver vários problemas, como por exemplo, quais são os melhores tamanhos de PCM (*Phase Change Memory*) e DRAM, como alocar estaticamente dados e programa, e como migrar dados entre PCM e DRAM.

Em Panda (2001) os autores realizaram otimizações em *software* e *hardware*. Primeiramente, otimizações independentes da arquitetura na forma de transformações de código são examinadas. Logo após, um amplo espectro de técnicas de otimização, que tratam das arquiteturas de memória em diferentes níveis de granularidade, são avaliadas. Estas técnicas vão desde os arquivos de registro de memória *on-chip*, como *caches* de dados até a memória dinâmica (DRAM).

Tecnologias de memórias não voláteis tem sido estudadas nos últimos anos e muitos trabalhos apontam vantagens na utilização destas memórias em sistemas embarcados. Os trabalhos anteriores de Wang et al. (2012), Li et al. (2012), Hu et al. (2013), confirmam que a memória principal NVM pode realizar economia energética significativa com desempenho comparável ao de uma memória principal tradicional DRAM. Mais recentemente, no trabalho de Mittal e Vetter (2016), são apresentadas técnicas de software propostas para explorar as vantagens e reduzir as desvantagens de NVMs quando usadas para projetar sistemas de memória e, em particular, o armazenamento secundário (por exemplo, drive de estado sólido) e a memória principal. No trabalho de Mittal e Vetter (2016), são revisadas diversas obras que utilizam algum tipo de memória não volátil. Essas obras são classificadas de acordo com o tipo da NVM utilizada e em qual nível da hierarquia de memória ela está inserida. Assim, os autores conseguem destacar as semelhanças e diferenças de cada abordagem utilizada.

Nesta dissertação, para um melhor entendimento, os trabalhos revisados foram divididos em duas classes: aqueles que realizam otimizações em *caches* (Seção 3.1) e aqueles que realizam otimizações em *scratchpads* (Seção 3.2). Alguns desses trabalhos realizam otimizações em níveis arquiteturais e outros trabalhos empregam técnicas no nível de software para melhorar os acessos ocorridos à

memória. Recentemente, tecnologias de memória não volátil começaram a ser exploradas em diversos trabalhos, como por exemplo, os trabalhos de Komalan et al. (2015), Wang et al. (2013) e Hu et al. (2013). Enquanto Komalan et al. (2015) aborda o emprego de memórias NVM em caches, Wang et al. (2013) e Hu et al. (2013) as empregam em *scratchpads*.

3.1 Otimizações realizadas em *Caches*

Muitos autores exploram a hierarquia de *cache* em processadores embarcados. Em Kwak e Choi (2010), por exemplo, os pesquisadores usam uma *cache* filtro, onde existe uma pequena memória *cache* auxiliar entre a CPU e o nível 1 (L1) da *cache*. A mesma é uma *cache* uniforme, ou seja, contém dados e instruções. Por causa das faltas ocorridas na *cache* filtro, há uma perda considerável e inevitável de 20% no desempenho, contudo possui uma redução de mais de 50% no consumo energético. Já em Dash e Srikanthan (2010), os autores se dedicaram em colocar uma *cache* de instruções com o objetivo de reduzir o consumo da energia da aplicação multitarefa, sem se preocuparem muito com o desempenho. Essa técnica foi empregada em sistemas multiprocessados, e o impacto no desempenho da *cache foi avaliado*. Os resultados apontaram que a interferência da *cache,* devido aos segmentos de código, era demasiada baixa para afetar o processo de ajuste de *cache*.

No trabalho de Alipour, Salehi e Moshari (2011), foi feita uma exploração de *cache* em processadores embarcados, considerando o desempenho e o consumo de energia como fatores primordiais. No trabalho, foram utilizados diversos *benchmarks* a fim de descobrir o efeito da diversidade dos acessos à *cache* no desempenho e consumo energético dos processadores embarcados. Os resultados mostraram que embora o emprego de *caches* maiores possa melhorar o desempenho em processadores embarcados, observa-se uma saturação e após um limite do aumento da *cache*, a melhoria em desempenho começa a decair. Com isso, foi introduzida uma série de hierarquias de *cache*, baseadas em penalidades de desempenho, para identificar a avaliação máxima possível no desempenho, ainda conseguiram mostrar que, apesar das *caches* possuírem tamanhos menores, elas podem possuir um desempenho aceitável. O mesmo trabalho também introduz um modelo para calcular o consumo de energia da hierarquia de cache para identificar o

mínimo e o máximo da avaliação de eficiência energética com base no desempenho por consumo de energia. Este modelo é empregado pelos autores de Qadri e Mcdonaldo-Maier (2009) e Silva et al. (2011). Em Qadri e Mcdonaldo-Maier (2009), os autores avaliam um modelo de consumo de energia, entretanto focaram apenas em uma *cache* de dados. Trabalhos como este que analisam o consumo energético e a taxa de transferência de uma *cache* de dados em determinadas aplicações, fornecem para o projetista de *hardware* e *software* um auxílio para ajustar a *cache* ou a aplicação para um determinado consumo energético.

Ainda em Alipour, Salehi e Moshari (2011), os resultados mostram que nem sempre será verdade que para melhorar o consumo energético o desempenho deve ser degradado, como são demonstrados em pesquisas anteriores, o trabalho de Alipour, Salehi e Moshari (2011) afirma que consumo energético e desempenho estão fortemente relacionados com a natureza da aplicação. Com base nestes resultados, os processadores embarcados podem empregar *cache* L1 de 16 a 64 KB e a L2 pode variar de 32 até 128 KB de tamanho. Estes valores servem de recomendação para qualquer projetista determinar o tamanho da *cache* com configuração ideal para processadores embarcados, isto se for considerado o desempenho por energia.

No trabalho de Mittal, Vetter e Li (2015), os autores apresentam diversas pesquisas sobre abordagens arquiteturais propostas para a concepção de sistemas de memória, especificamente *caches* utilizando memórias com tecnologias emergentes. Neste trabalho de revisão, os autores apresentam uma classificação dessas tecnologias e abordagens arquiteturais baseadas em suas características principais e conseguem realçar as semelhanças e diferenças dessas abordagens.

Em Komalan et al. (2015), os autores utilizaram uma memória não volátil para substituir as memórias SRAM em *caches* no nível L1. No artigo, os autores substituíram a cache de dados L1 SRAM por uma memória STT-MRAM e exploraram a penalização de leitura da cache L1 de dados com o emprego da NVM em um processador ARM. O impacto da substituição da memória SRAM por uma STT-MRAM foi avaliado quanto ao desempenho de *benchmarks*. Primeiramente, observou-se uma sobrecarga de 54% nas penalidades de desempenho dos *benchmarks* analisados, associadas ao emprego da memória STT-MRAM quando comparada com a SRAM. Estas penalidades se devem a latência de leitura de uma STT-MRAM ser bem superior se comparada com a latência de uma SRAM. Após

foram realizadas algumas modificações na arquitetura da cache de dados empregada, que resultaram em uma redução destas penalidades, através da utilização de um VWB (*Very Wide Buffer*) entre a STT-MRAM e o processador. A Figura 15 ilustra a utilização do VWB.

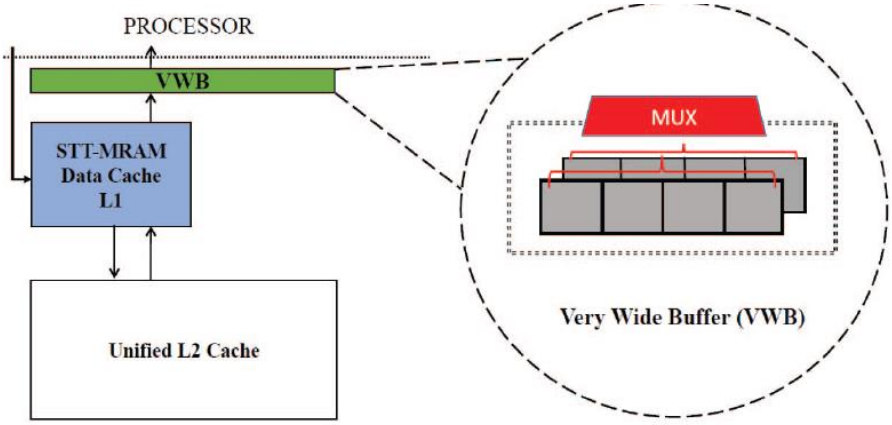


Figura 15 – Modificação da organização D-cache L1 SRAM com uma D-cache L1 STT-MRAM, utilizando o VWB.
Fonte: KOMALAN et al., 2015.

Após a realização da modificação arquitetural, Komalan et al. (2015), ainda propuseram e realizaram otimizações de software, ou seja, transformações e otimizações de código, exploraram o paralelismo através de vetorizações, alinhamentos de loop, e reduções do número de decisões de desvios. Todas estas otimizações foram automaticamente realizadas e conseguiram reduzir ainda mais as penalidades de leitura.

Para avaliar a eficácia das modificações propostas, os autores implementaram o projeto via o simulador GEM5 (BINKERT, et al., 2011) e utilizaram um conjunto de *benchmark* do PolyBench (POUCHET e YUKI, 2011). Através de experimentos, verificaram a influência do tamanho do VWB, na redução das penalidades. Os experimentos apontaram que ao aumentar o tamanho do VWB, reduz-se as penalidades. Entretanto, aspectos de custo e energia associados ao aumento do VWB também devem ser considerados, assim como o roteamento do *layout*, pois o *layout* do projeto torna-se mais complexo a medida em que o VWB é aumentado. Assim, os autores indicaram que o tamanho ideal para o VWB é cerca de 2KBits.

Ainda no trabalho de Komalan et al. (2015), são realizadas comparações entre a solução arquitetural, com outros trabalhos relacionados. A solução

arquitetural oferece melhor desempenho (menor porcentagem de penalidades), devido as otimizações de software e modificações em hardware. Porém, quando as mesmas transformações são realizadas em uma SRAM, a solução com SRAM apresentou um desempenho melhor em relação a solução com a STT-MRAM, cerca de 8%. Entretanto, vale destacar que essa comparação não leva em conta as vantagens óbvias oferecidas pela cache com NVM (STT-MRAM), como uma área e gasto energético menores e não volatilidade. Assim, conclui-se que, pelo trabalho de Komalan et al. (2015), as explorações realizadas juntamente com as transformações e otimizações de código podem reduzir a penalidade de desempenho introduzido pela utilização da NVM que inicialmente era de 54% para níveis toleráveis, cerca de 8%, mesmo para os piores casos testados.

3.2 Otimizações realizadas em *Scratchpads*

Uma otimização muito utilizada em sistemas embarcados é o uso de memórias *scratchpads*, pois apresentam vantagens quando comparadas com as memórias *caches,* como discutido anteriormente. Pesquisam vem sendo realizadas nas quais são avaliados o emprego de tecnologias emergentes em memórias *scratchpads*, como em Wang et al. (2013) e Hu et al. (2013).

Em Wang et al. (2013), os autores focaram em substituir uma memória *scratchpad* (SPM) empregando SRAM por uma *scratchpad* empregando STT-RAM. A memória SST-RAM foi escolhida dentre outras NVMs por apresentar vantagens em alguns aspectos, como por exemplo, melhores resultados para as operações de leitura e escrita (latência e consumo energético), para determinados tamanhos de memórias. Os resultados de ambas versões da SPM*,* com SST-RAM e com SRAM, foram comparados, os critérios observados na comparação foram o consumo energético e desempenho para operações de escrita e leitura juntamente com a área e o *leakage* de cada tipo de memória empregada. Posteriormente, os autores também avaliaram uma SPM híbrida, a qual emprega ambas as tecnologias, SRAM e STT-RAM, onde os dados mais escritos são alocados na SRAM e os dados mais lidos são alocados na STT-RAM.

Para a realização do trabalho, os autores utilizaram uma plataforma de simulação construída sobre a ferramenta SIMICS juntamente com o GEMS. Os *benchmarks* selecionados para o trabalho foram retirados do *MiBench* e a tecnologia

empregada foi de 45nm (nanômetros). Para estimar as latências e o consumo de energia para operações de leitura / escrita para um determinado tamanho de memória não volátil foi utilizado o simulador NVsim (DONG, et al. 2012). Com a ferramenta NVsim, três tipos diferentes de otimizações, latência otimizada, produto da energia-*delay* (EDP) otimizado, e energia otimizado (WANG, et al., 2013) foram extraídos.

Como mencionado anteriormente, os autores exploraram duas abordagens diferentes para *scratchpad*, a primeira abordagem é uma substituição direta de uma memória SRAM por uma memória STT-RAM, a segunda abordagem é um *design* híbrido (SRAM e STT-RAM) para a memória *scratchpad*. Na abordagem híbrida, foram utilizadas diferentes áreas (proporções) de SRAM e STT-RAM.

Na primeira abordagem, a substituição direta de uma memória SRAM por uma memória STT-RAM, no quesito desempenho (IPC, Instruções Por Ciclo), em média a opção de otimização de energia resultou em uma degradação de cerca de 8%, enquanto que a latência otimizada e o produto energia-*delay* melhoraram cerca de 24% e 10%, respectivamente. Ainda na substituição direta, no quesito consumo energético, em média a latência otimizada sofreu uma degradação, cerca de 20%, enquanto que o produto energia-*delay* e a energia otimizada melhoraram cerca de 18% e 31%, respectivamente.

Como a otimização produto energia-*delay* (EDP) obteve os melhores resultados, tanto para desempenho como para consumo energético, na primeira abordagem (substituição direta), foi utilizada a otimização da EDP para as diversas proporções da SPM híbrida (segunda abordagem). Os autores fixaram a área de uma SPM de 64KB SRAM como linha base para realizarem a comparação de diversas proporções do modelo híbrido. Por exemplo, os autores utilizaram a proporção de 1:1 de área para a SPM híbrida, o que significa que a SPM híbrida possui 32KB de SRAM e 128KB de STT-RAM, enquanto o SPM linha de base tem 64 KB de SRAM. Segundo Wang et al. (2013) a memória STT-RAM pode ser projetada cerca de quatro vezes mais densa que a memória SRAM e estas duas SPMs de densidades diferentes possuíram uma área total semelhante de silício, devido a variação na área requerida por cada tecnologia.

Na segunda abordagem, *design* híbrido (SRAM e STT-RAM), na comparação de desempenho para as diversas proporções para a SPM híbrida, a proporção 2:1 (SRAM:STT-RAM) possui os melhores resultados, cerca de 36% em média. Na

comparação do consumo energético para as diversas proporções para a SPM híbrida, o modelo ALLSTT-RAM (empregando somente memória STT-RAM), apresentou os melhores resultados, cerca de 29% em média. Na comparação do produto do desempenho e consumo energético para as diversas proporções da SPM híbrida, a proporção 2:1 (SRAM:STT-RAM) apresentou os melhores resultados, cerca de 45% em média.

Pelo trabalho de Wang et al., (2013), percebe-se que, através das explorações realizadas juntamente com as otimizações demonstradas, que a arquitetura SPM híbrida pode superar SPM's constituídas somente por SRAM ou STT-RAM. Devido as características atraentes da STT-RAM, como o baixo *leakage* e de alta densidade, esta é apontada como uma memória promissora para modelos híbridos de memória ou para a substituição da SRAM.

Em Hu et al. (2013), os autores utilizaram como memória emergente uma PCM. O trabalho teve como o estudo de caso uma SPM híbrida composta de 16KB SRAM e 64KB PCM, a qual foi comparada com uma SPM de 32KB SRAM. Ainda nesse trabalho, os autores exploraram diferentes algoritmos para a alocação dos endereços nos tipos de memórias utilizadas, propondo um novo algoritmo para alocação em memórias híbridas. Os benchmarks empregados pelos autores para a realização do trabalho foram selecionados do *MiBench*. De acordo com os resultados experimentais, com a ajuda do algoritmo proposto pelos autores, o modelo híbrido de arquitetura SPM reduziu o tempo de acesso à memória em 18.17%, a energia dinâmica em 24.29%, e o *leakage* em 37.34% quando comparada com uma SPM contendo SRAM com a mesma área.

Ambos os trabalhos citados, Hu et al. (2013) e Wang et al., (2013), utilizaram o NVsim (DONG, et al. 2012) com tecnologia 45nm, para a obtenção dos parâmetros das memórias PCM, STT-RAM e SRAM, como as latências de acesso e o consumo energético para as operações de escrita e leitura. Os dois trabalhos conseguiram reduzir significativamente o consumo energético e melhorar o desempenho para diversos benchmarks executados em um processador ARM, através do emprego de tecnologias emergentes de memória em SPMs.

Os trabalhos anteriores exploraram uma tecnologia de NVM (somente STT-RAM ou PCM). O presente trabalho explora duas tecnologias de NVMs, a STT-RAM e PCM. Quando comparadas com outras NVMs, a STT-RAM e PCM apresentaram os melhores resultados extraídos pelo NVSim, como por exemplo, melhor consumo

energético ou desempenho para as operações de leitura e/ou uma maior densidade. Utiliza-se *scratchpads* em vez de *caches* como estudo de caso, pois as *scratchpads* possuem algumas vantagens em relação a *caches*, como por exemplo, menor área, menor consumo energético e podem proporcionar diversas vezes uma melhor previsibilidade e sincronização em dispositivos de tempo real. Além disso, *caches* possuem problemas de penalidades de desempenho quando a memória SRAM é substituída por uma NVM, como citado anteriormente no trabalho de Komalan et al. (2015).

4 AVALIAÇÃO DE HIERARQUIA DE MEMÓRIA UTILIZANDO MEMÓRIAS NÃO VOLÁTEIS

Este capítulo apresenta primeiramente uma proposta de hierarquia de memória usando NVMs. Neste capítulo, também a metodologia adotada na avaliação da hierarquia proposta é detalhada, apresentando os *benchmarks e* a arquitetura avaliada. Além disso, o capítulo revisa as ferramentas utilizadas para a obtenção dos *traces* da arquitetura avaliada e dos parâmetros das memórias voláteis e não voláteis empregadas. O capítulo reforça também a diferença entre as análises realizadas (estática e dinâmica) e detalha os experimentos realizados neste trabalho, cujos resultados são apresentados e discutidos no Capítulo 5.

4.1 Hierarquia de Memória Proposta

A Figura 16 ilustra a visão geral da hierarquia de memória proposta. Nesta hierarquia é empregada uma SPM híbrida, pois a SPM possui uma memória SRAM e pelo menos uma NVM. Na Seção 4.2, será demonstrado experimentos utilizando diferentes modelos de SPMs, esses modelos são diferentes no tamanho ou nos tipos de memórias empregadas. Pela Figura 16, quando um dado é requerido pela CPU (*Central Processing Unit*), a CPU pode buscar esse dado na SPM ou na MP (Memória Principal). Todas as memórias da arquitetura podem trocar dados, ou seja, por exemplo, a memória NVM pode trocar informações com a SRAM ou com a MP (formada por uma memória DRAM) e vice-versa. Essas trocas de informações são determinadas com base no número de acessos ocorridos para cada tipo de operação (leitura e escrita) de um determinado endereço e as características (vantagens ou desvantagens) do tipo da memória utilizada no armazenamento deste endereço. Por exemplo, na PCM são armazenados os endereços que possuem os maiores acessos para a operação de leitura sem nenhuma operação de escrita, enquanto que na SRAM são armazenados os endereços que possuem os maiores números de acessos (leitura + escrita). Neste caso, se um endereço está contido na PCM e ao decorrer da execução da aplicação, este endereço realiza uma operação de escrita, ele será desalocado e inserido na SRAM (se possuir espaço disponível).

As características de cada tipo de memória empregada neste trabalho são apresentadas na Tabela 4 e a Seção 4.2.1 apresenta a ordem e as características das alocações nas memórias.

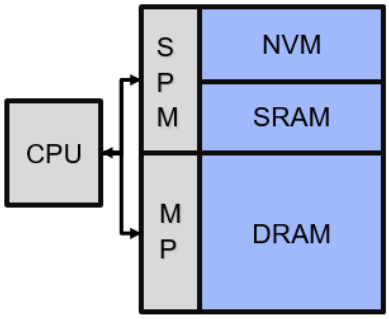


Figura 16 – Visão geral da hierarquia de memória.

4.2 Metodologia

Considerando o acesso à memória em sistemas embarcados, a arquitetura de memória possui uma forte influência sobre os objetivos do projeto destes sistemas, por causa da latência e da quantidade energética gasta por acesso à memória. Neste contexto, este trabalho tem por objetivo investigar o desempenho e o consumo de energia dos acessos à memória para arquitetura ARM e realizar comparações entre SPMs possuindo diferentes tipos e tamanhos de memórias, com o intuito de melhorar o desempenho dessa arquitetura sem que haja grandes perdas no consumo energético.

A metodologia deste trabalho se baseia em: através da arquitetura ARM, dos *benchmarks* e de uma ferramenta capaz de simular a arquitetura selecionada (Simics), é obtido os *traces* de dados para as diferentes aplicações avaliadas nesse trabalho. Após, é realizada a verificação dos acessos totais ocorridos para cada endereço, além de observar a quantidade de operações de leitura/escrita nesses endereços. Assim, obtém-se o número total de endereços juntamente com o número total de operações de leitura e escrita para cada *benchmark* analisado. Para a realização destas tarefas foram utilizados dois *scripts*, um para a geração dos *traces* de dados de cada uma das aplicações (*script trace*) e outro para a realização da análise desses *traces* (*script análise*).

Através dos resultados obtidos até esse momento, é realizado a análise do tamanho de memória necessária para suportar cada uma das aplicações

(experimento 1 - verificação dos endereços e acessos), com o objetivo de verificar qual é a maior memória necessária para suportar uma determinada aplicação/*benchmark.* Paralelamente, é utilizado o NVsim capaz de obter os parâmetros das NVMs e da SRAM, como a latência e consumo energético para as operações de escrita e leitura.

Com os resultados obtidos do NVsim e com a análise feita até esse momento, são realizados os experimentos utilizando SPMs híbridas (experimento 2 - comparação entre as diferentes SPMs). Para isso, o *script análise* foi modificado (*script análise alocação)*. Além da verificação dos acessos ocorridos, esse último *script* agora é responsável por alocar os endereços nas diferentes memórias utilizadas, de acordo com as vantagens e desvantagens de cada tipo de memória e considerando a quantidade dos acessos ocorridos para as operações de leitura e escrita para cada endereço.

A partir deste ponto (experimento 2), o trabalho pode ser dividido em duas análises diferentes, a análise estática e dinâmica. Cada análise possui seu respectivo *script* responsável por realizar a alocação dos endereços nas memórias utilizadas (*script análise alocação estática* e o *script análise alocação dinâmica).* Alguns parâmetros de entrada desses *scripts* são, o tipo, o tamanho e a latência das operações de escrita e leitura para cada memória utilizada. A Seção 4.2.1 explica a diferença entre essas análises, juntamente com os fluxogramas dos *scripts*.

Através do *script de alocação* (estático ou dinâmico), é realizado a comparação entre diferentes tipos de SPMs (experimento 2 - comparação entre as diferentes SPMs), para cada tipo de análise (estática e dinâmica). Para essa comparação são utilizadas cinco SPMs diferentes como estudo de caso, as quais são apresentadas na Tabela 3.

Tabela 3 – Configurações das *scratchpads*

Configuração da SPM	Tamanho das Memórias (KB)		
	SRAM	STT-RAM	PCM
1. SPM	32	-	-
2. SPM	16	64	-
3. SPM	16	32	-
4. SPM	16	32	8
5. SPM	16	64	8

No experimento 2, uma SPM com 32KB de SRAM é empregada como linha base para a realização das comparações. Com essas comparações é possível identificar o impacto da inclusão das memórias não voláteis na SPM. Na Seção 5.2 é explicado o porquê da utilização dessas cinco SPMs.

Um novo experimento é realizado para identificar a influência das SPMs no desempenho e no consumo energético total das aplicações (experimento 3 – comparação entre as diferentes SPMs utilizando MP). Para isso, a latência e o consumo energético das operações de leitura e escrita da DRAM (tipo de memória utilizada na MP) foram estimados através da ferramenta Cacti. Para o experimento 3, foram utilizadas as mesmas arquiteturas propostas no experimento 2, a diferença entre esses experimentos é que o experimento 3 considera os valores de desempenho consumo energético da memória principal (DRAM). A linha base para as comparações do experimento 3 é uma arquitetura de memória sem a utilização de nenhuma SPM, ou seja, a linha base desse experimento possui somente a memória DRAM. Os resultados dos experimentos são apresentados e discutidos no Capítulo 5. A Figura 17, ilustra detalhadamente o fluxo da metodologia para a obtenção dos resultados dos experimentos realizados neste trabalho.

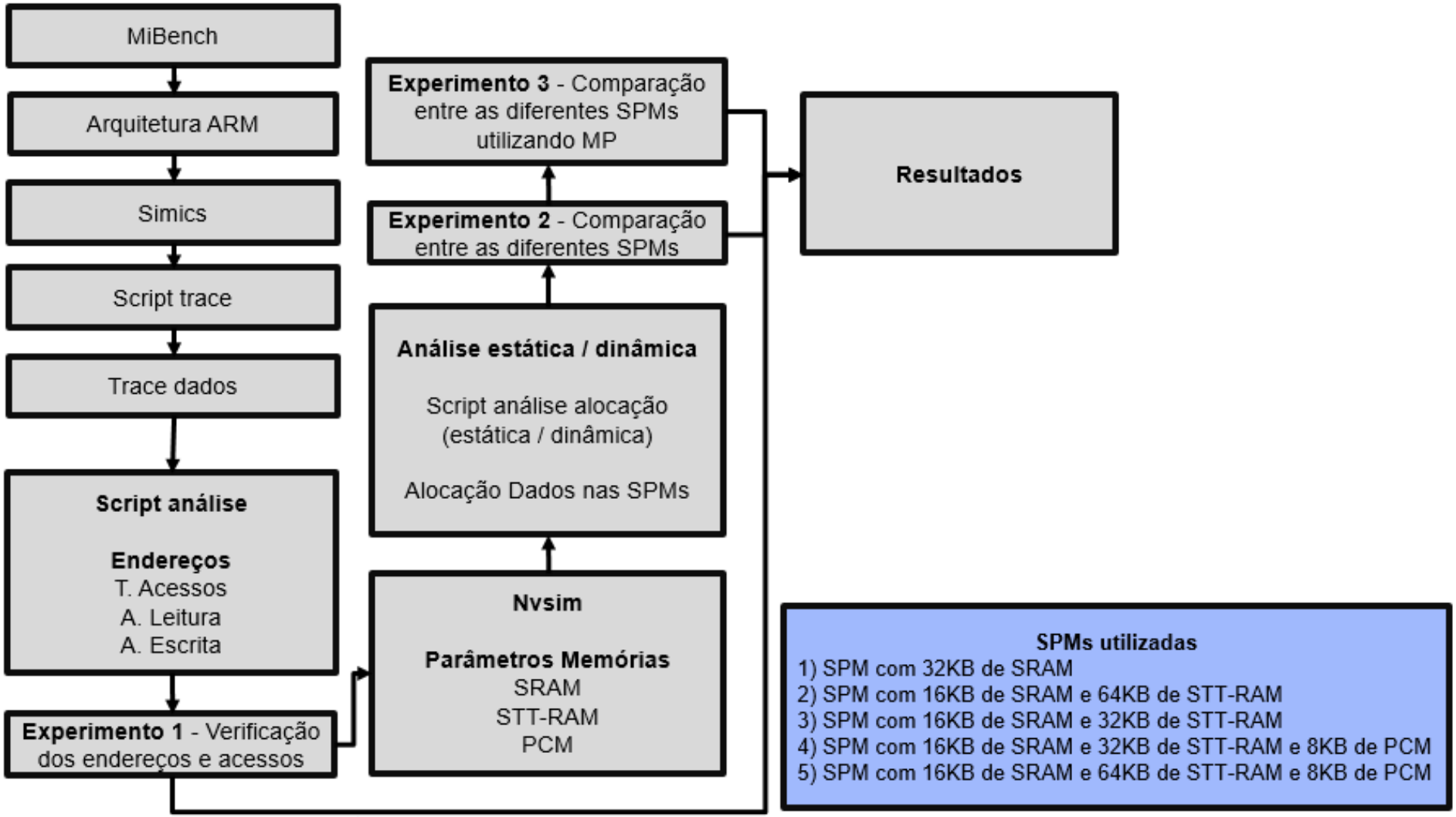


Figura 17 – Fluxo da metodologia.

O primeiro passo foi compilar todos os *benchmarks* selecionados para gerar binários através de um *cross-compiler* da arquitetura ARM. Para o segundo passo é utilizada a ferramenta Simics juntamente com um *script* (*scritp trace*) para a geração automática do *trace* de um determinado *benchmark.* A Figura 18 apresenta um fragmento do arquivo gerado do *trace* de dado do *benchmark* FFT para a arquitetura ARM.

```
data: [         1] CPU  0 <v:0xdf8c0088> <p:0x02f8c0088> Read   4 bytes  0x5b
data: [         2] CPU  0 <v:0xdf8c0088> <p:0x02f8c0088> Write  4 bytes  0x5c
data: [         3] CPU  0 <v:0xc036c35d> <p:0x01036c35d> Read   1 bytes  0x0
data: [         4] CPU  0 <v:0xc036c334> <p:0x01036c334> Read   4 bytes  0xf8010000
data: [         5] CPU  0 <v:0xc036c380> <p:0x01036c380> Read   4 bytes  0xb5
data: [         6] CPU  0 <v:0xc036c380> <p:0x01036c380> Write  4 bytes  0xb6
data: [         7] CPU  0 <v:0xf8010004> <p:0x0e0010004> Read   4 bytes  0x3
data: [         8] CPU  0 <v:0xdf8c0088> <p:0x02f8c0088> Read   4 bytes  0x5c
data: [         9] CPU  0 <v:0xdf8c0084> <p:0x02f8c0084> Read   4 bytes  0x5c
data: [        10] CPU  0 <v:0xc036c368> <p:0x01036c368> Read   4 bytes  0xdf8c0000
data: [        11] CPU  0 <v:0xdf8c0000> <p:0x02f8c0000> Read   4 bytes  0xdf8cb000
data: [        12] CPU  0 <v:0xc0311ec8> <p:0x010311ec8> Write  4 bytes  0xc036c32c
data: [        13] CPU  0 <v:0xc0311ecc> <p:0x010311ecc> Write  4 bytes  0xdf8c0000
data: [        14] CPU  0 <v:0xc0311ed0> <p:0x010311ed0> Write  4 bytes  0x0
data: [        15] CPU  0 <v:0xc0311ed4> <p:0x010311ed4> Write  4 bytes  0xc01678f4
```

Figura 18 – Fragmento do *trace* de dados gerado do *benchmark* FFT para a arquitetura ARM.

Neste *trace*, ilustrado na Figura 18, pode-se observar a ordem dos dados executados, o endereço virtual, denotado por “v”, o endereço físico denotado por “p”, o tipo de acesso a memória (leitura ou escrita), o tamanho do endereço em bytes e o conteúdo que o endereço possui.

Após a obtenção do *trace*, o próximo passo é a verificação de quantos endereços foram acessados no *trace* de dados e o número de vezes que cada operação foi executada para todos os endereços para cada *benchmark* selecionado no trabalho. Para automatizar esta análise, foi criado um *script* (*script análise*) para a realização desta tarefa. Essa verificação é o primeiro experimento do trabalho (experimento 1 – verificação dos endereços e acessos), detalhado na Seção 4.2.5.1. A Figura 19 ilustra um recorte da saída do *script análise* para a aplicação FFT. As informações contidas na Figura 19 estão sendo esclarecidas na Seção 4.2.1.1.

```
Existem 2513599 linhas no arquivo
Todo Arquivo Pesquisado? Sim
Linha Inicial de Pesquisa = 1
Linha Final de Pesquisa = 2513599
Existem 2513599 linhas executadas/pesquisadas no arquivo
Existe no maximo 2513599 enderecos diferentes
Existem 278470 enderecos diferentes
O maior numero de operacoes consecutivas foi de 80917
Endereco|Quantidade_Acessos|Op_Leitura|Op_Escrita|Maximo_
0xdf8c0088|1603|1308|295|15|1|15|R|1|R|8
0xc036c35d|1705|1705|0|1705|0|1705|R|1705|R|1705
0xc036c334|1705|1705|0|1705|0|1705|R|1705|R|1705
0xc036c380|590|295|295|1|1|1|R|1|W|1
0xf8010004|714|714|0|714|0|714|R|714|R|714
0xdf8c0084|1110|1013|97|42|1|42|R|6|R|2
0xc036c368|285|285|0|285|0|285|R|285|R|285
0xdf8c0000|188|188|0|188|0|188|R|188|R|188
0xc0311ec8|242|121|121|1|1|1|W|1|R|1
0xc0311ecc|251|125|126|1|2|2|W|1|R|1
0xc0311ed0|261|127|134|1|3|3|W|1|R|1
0xc0311ed4|261|127|134|1|3|3|W|1|R|1
0xdf8cb09c|971|877|94|40|1|40|R|3|R|4
0xc0311ebc|53|25|28|1|2|2|W|1|R|1
0xc0311ec0|47|26|21|6|1|6|W|1|R|1
0xc0311ec4|217|108|109|1|2|2|W|1|R|1
0xdf8cb0e4|1203|401|802|1|2|2|R|1|W|2
0xc0311eb0|35|16|19|1|2|2|W|1|R|1
0xc0311e8c|12|6|6|1|1|1|W|1|R|1
0xc0311e90|14|7|7|1|1|1|W|1|R|1
0xc0311e94|28|14|14|1|1|1|W|1|R|1
```

Figura 19 – Recorte da saída do *script análise* para a aplicação FFT.

A ideia por trás do experimento 1 é identificar a quantidade de memória necessária para alocar os endereços das aplicações. O tamanho da memória necessária é definido pela aplicação que possui o maior número de endereços diferentes (pior caso). Após a realização deste primeiro experimento, determinou-se que seria necessário um tamanho de 2MB para a memória principal.

Após o NVSim é utilizado, para aquisição dos parâmetros das NVMs e da SRAM. Em nossos experimentos, a tecnologia de 45nm com otimização balanceada entre área e latência foi empregada através da configuração do NVSim. A Tabela 4 exibe os resultados dos parâmetros encontrados pelo NVsim para os diferentes tipos e tamanhos de NVMs e SRAMs.

Tabela 4 – Parâmetros para a NVM e SRAM

	SRAM		STT-RAM		PCM
Parâmetros	16K	32K	32K	64K	8K
Área (um^2)	41478,678	81170,765	24325,955	43712,02	2143,109
Latência Leitura (ns)	2,887	3,41	3,113	3,232	0,473743
Latência Escrita (ns)	2,887	3,41	7,227	7,257	-
RESET Latência (ns)	-	-	-	-	40,409
SET Latência (ns)	-	-	-	-	150,409
Energia Leitura (pJ)	8,570	15,672	6,159	6,273	6,194
Energia Escrita (pJ)	1,159	1,701	11,017	13,162	-
RESET Energia (pJ)	-	-	-	-	3603
SET Energia (pJ)	-	-	-	-	3603
Leakage (mW)	18,934	37,652	0,839373	1,44	0,314516

Fonte: NVSim.

Com os resultados obtidos com o NVSim, o *script análise* foi modificado para que alocasse os endereços nos tipos de memórias utilizadas (MP/SRAM/STT-RAM/PCM), de acordo com suas respectivas vantagens e benefícios. Nessa etapa, duas análises diferentes, a análise estática e dinâmica, são realizadas. Cada uma dessas análises emprega *scripts* diferentes para a alocação dos endereços (*script análise alocação estática* e *script análise alocação dinâmica*). Os parâmetros de entrada dos *scripts* de alocação responsáveis por influenciarem as alocações ocorridas nas memórias são os parâmetros do tipo, do tamanho e da latência (escrita e leitura) para cada memória utilizada. A Seção 4.2.1 explica as abordagens estática e dinâmica, juntamente com os fluxogramas dos *scripts*. A Figura 20 apresenta um fragmento da saída do *script análise alocação dinâmica* para o *benchmark* FFT quando empregada uma SPM híbrida, contendo PCM, STT-RAM e SRAM. As informações contidas na Figura 20 estão sendo explicadas na Seção 4.2.1.2.

Após realiza-se a comparação entre as SPMs utilizadas nesse trabalho (experimento 2 - comparação entre as diferentes SPMs). Neste experimento, a SPM 1 é usada como linha base para a análise comparativa. Como mencionado anteriormente, a Tabela 3 resume as SPMs que foram analisadas, juntamente com os tipos e tamanhos de suas respectivas memórias.

```
0xdf8c0088|1603|1308|295|15|1|15|R|1|R|8|STT|1603|0|1601|2|4|1599|1602|1|0|1|1|1
0xc036c35d|1705|1705|0|1705|0|1705|R|1705|R|1705|PCM|1705|0|1705|0|1705|0|1|1704|0|0|0|1
0xc036c334|1705|1705|0|1705|0|1705|R|1705|R|1705|PCM|1705|0|1705|0|1705|0|1|1704|0|0|0|1
0xc036c380|590|295|295|1|1|1|R|1|W|1|SRAM|590|0|2|588|590|0|589|1|0|1|0|1
0xf8010004|714|714|0|714|0|714|R|714|R|714|PCM|714|0|714|0|714|0|1|713|0|0|0|1
0xdf8c0084|1110|1013|97|42|1|42|R|6|R|2|STT|1110|0|1110|0|7|1103|1104|6|0|0|1|1
0xc036c368|285|285|0|285|0|285|R|285|R|285|PCM|285|0|285|0|285|0|1|284|0|0|0|1
0xdf8c0000|188|188|0|188|0|188|R|188|R|188|PCM|188|0|188|0|188|0|1|187|0|0|0|1
0xc0311ec8|242|121|121|1|1|1|W|1|R|1|SRAM|242|0|1|241|242|0|242|0|0|1|0|0
0xc0311ecc|251|125|126|1|2|2|W|1|R|1|SRAM|251|0|1|250|251|0|251|0|0|1|0|0
0xc0311ed0|261|127|134|1|3|3|W|1|R|1|SRAM|261|0|1|260|261|0|261|0|0|1|0|0
0xc0311ed4|261|127|134|1|3|3|W|1|R|1|SRAM|261|0|1|260|261|0|261|0|0|1|0|0
0xdf8cb09c|971|877|94|40|1|40|R|3|R|4|STT|971|0|971|0|4|967|968|3|0|0|1|1
0xc0311ebc|53|25|28|1|2|2|W|1|R|1|SRAM|53|0|1|52|53|0|53|0|0|1|0|0
0xc0311ec0|47|26|21|6|1|6|W|1|R|1|SRAM|47|0|1|46|47|0|47|0|0|1|0|0
0xc0311ec4|217|108|109|1|2|2|W|1|R|1|SRAM|217|0|1|216|217|0|217|0|0|1|0|0
0xdf8cb0e4|1203|401|802|1|2|2|R|1|W|2|SRAM|1203|0|2|1201|1203|0|1202|1|0|1|0|1
0xc0311eb0|35|16|19|1|2|2|W|1|R|1|SRAM|35|0|1|34|35|0|35|0|0|1|0|0
0xc0311e8c|12|6|6|1|1|1|W|1|R|1|MP|12|0|1|11|12|0|12|0|1|1|0|0
0xc0311e90|14|7|7|1|1|1|W|1|R|1|MP|14|0|1|13|14|0|14|0|1|1|0|0
0xc0311e94|28|14|14|1|1|1|W|1|R|1|MP|28|0|1|27|28|0|28|0|1|1|0|0
```

Figura 20 – Saída do *script análise alocação dinâmica* para a aplicação FFT.

Por último, é realizado o terceiro experimento (experimento 3 – comparação entre as diferentes SPMs utilizando MP), o qual consiste em utilizar as SPMs e avaliar a influência que elas possuem sobre a latência e o consumo energético total de cada aplicação. A ferramenta Cacti é utilizada para adquirir os parâmetros da memória DRAM, considerando também a tecnologia de 45nm. A Tabela 5 exibe os resultados dos parâmetros encontrados com a ferramenta Cacti para a DRAM de tamanho 2MB.

Tabela 5 – Parâmetros para a DRAM

	DRAM
Parâmetros	2M
Área (um^2)	9744873,729
Latência Leitura (ns)	4,392
Latência Escrita (ns)	4,392
Energia Leitura (pJ)	164,753
Energia Escrita (pJ)	164,753
Leakage (mW)	31,109

Fonte: CACTI.

Para a realização do experimento 3, são empregadas as mesmas SPMs do experimento 2. Para linha base deste experimento é utilizada uma solução com somente uma memória DRAM. A Seção 4.2.5 exibe e explica todos experimentos realizados.

Com os resultados obtidos pelos *scripts* de alocações o consumo energético e desempenho para uma aplicação, pode ser calculado utilizando as fórmulas

apresentadas na Tabela 6. Onde, por exemplo, a latência leitura de um endereço é calculada pelo número de acessos que este endereço possui em uma das memórias (DRAM, SRAM, STT-RAM e PCM), multiplicado pelo parâmetro de latência de leitura desta memória que o endereço está contido. A mesma ideia é utilizada para a latência de escrita e para as energias (leitura/escrita) calculadas. Para obter o valor total do desempenho e consumo energético de uma aplicação, pela Tabela 6 para o desempenho total por exemplo, o cálculo é a soma da latência de leitura e da latência de escrita de todos os endereços da aplicação, a mesma ideia é utilizada para o cálculo da energia total para uma aplicação. Após obter o desempenho e consumo energético de todas as aplicações analisadas neste trabalho, calcula-se a média, o pior e o melhor valor para o consumo energético e desempenho de todas as aplicações para os experimentos 2 e 3. Os resultados destes experimentos estão sendo ilustrados na Seção 5.2.

Tabela 6 – Fórmulas para os cálculos das latências e energias

Número	Fórmula
1.	$Latência\_Leitura_{end} = Numero\_Acessos\_Leitura\_End_{Mem} * Latência\_Leitura_{Mem}$
2.	$Latência\_Escrita_{end} = Numero\_Acessos\_Escrita\_End_{Mem} * Latência\_Escrita_{Mem}$
3.	$Energia\_Leitura_{end} = Numero\_Acessos\_Leitura\_End_{Mem} * Energia\_Leitura_{Mem}$
4.	$Energia\_Escrita_{end} = Numero\_Acessos\_Escrita\_End_{Mem} * Energia\_Escrita_{Mem}$
5.	$Latência_{Total} = \sum_{n=0}^{n=num\_end} Latência\_Leitura_n + Latência\_Escrita_n$
6.	$Energia_{Total} = \sum_{n=0}^{n=num\_end} Energia\_Leitura_n + Energia\_Escrita_n$

4.2.1 Análise Estática e Dinâmica

Esta seção descreve as abordagens de análise estática e dinâmica empregadas no trabalho. Durante a execução do trabalho, a primeira abordagem realizada foi a estática, devido ao fato de ser a abordagem mais simples de ser implementada, além de apresentar um tempo menor para a obtenção de resultados.

A Figura 21 exibe um resumo dos procedimentos envolvidos em cada uma das abordagens estudadas, visando a obtenção dos seus respectivos resultados (alocação de endereços em diferentes memórias utilizadas). Pela Figura 21, percebe-se que, as duas abordagens recebem como entrada o *trace* de uma dada

aplicação e aplicam os *scripts* de alocação específicos para a obtenção dos resultados das análises. Por exemplo, para a abordagem estática, são utilizados o *script análise* e o *script análise alocação estática*, enquanto que a abordagem dinâmica utiliza somente o *script análise alocação dinâmica*.

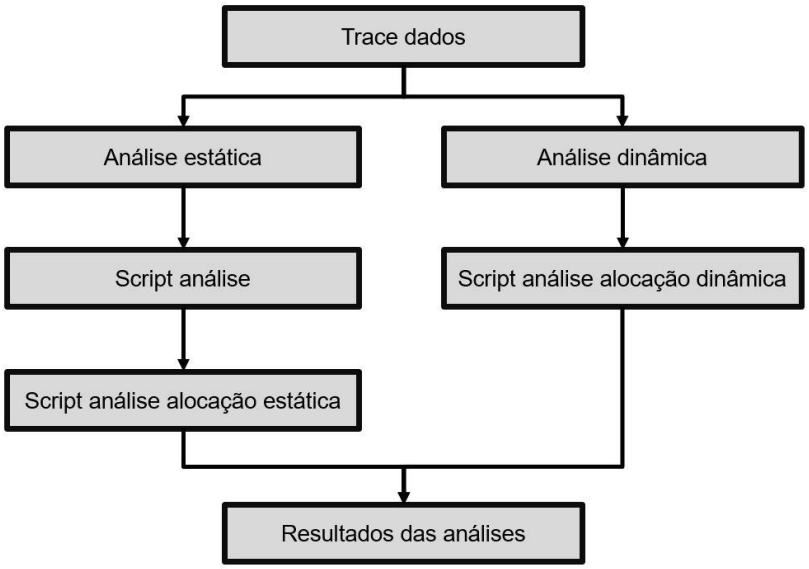


Figura 21 – Procedimentos das análises estática e dinâmica.

A Figura 22 apresenta a ordem empregada na alocação dos endereços nas memórias, considerando as tecnologias PCM, STT-RAM, SRAM e DRAM. As alocações dos endereços das duas abordagens seguem a mesma ordem. Dado um endereço, primeiramente ele será inserido na PCM, se o endereço respeitar as vantagens da PCM, caso contrário, este endereço será inserido na STT-RAM se respeitar as vantagens dessa memória, senão o endereço será inserido na SRAM se respeitar as condições dessa memória. No último caso, se o endereço não for inserido em nenhumas das memórias anteriores, ele será inserido na MP (DRAM). As memórias podem migrar endereços entre elas, de acordo com as características de cada memória.

Os *scripts* utilizados para a realização das alocações dos endereços, foram criados considerando os valores dos parâmetros obtidos pelo NVsim (Tabela 4). Assim, para cada tipo de memória foi definida uma característica, que determina a alocação ou não de um dado endereço nesta memória, de acordo com vantagens deste tipo de memória (Tabela 4).

A característica para alocar o endereço na PCM é que o endereço não deve possuir nenhum acesso de escrita, assim, os endereços que são inseridos nessa

memória são os que possuem os maiores números de acessos somente para a operação de leitura.

A característica para os endereços serem alocados na STT-RAM é que o total de acessos da operação de leitura deve ser maior que o total de acessos da operação de escrita multiplicada pela razão entre as latências de escrita por leitura (latência escrita dividida pela latência de leitura da memória STT-RAM). Assim, os endereços que possuírem um elevado número acessos da operação de leitura comparado com as operações de escritas serão inseridos na STT-RAM.

A característica para os endereços serem inseridos na memória SRAM, é que o número de acessos total (acessos das operações de escrita somado com os acessos das operações de leitura) devem ser os maiores possíveis, assim, os endereços que possuírem um elevado número de operações de escrita e de leitura serão inseridos na SRAM. Se o endereço a ser alocado não for inserido em nenhuma memória anterior, ele será alocado na DRAM.

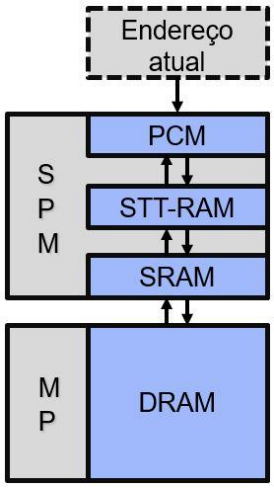


Figura 22 – Ordem das alocações nas memórias.

4.2.1.1 Análise Estática

Com o *trace* de uma determinada aplicação para a obtenção dos resultados das alocações nas memórias, utilizando a abordagem estática, primeiramente é necessário realizar uma pré-análise do *trace*. Essa pré-análise pode ser feita utilizando o *script análise*. Um fragmento da saída desse *script* para a aplicação FFT já foi demonstrado na Figura 18.

Pelas Figuras 18 e 19, percebe-se que, a ordem que os endereços diferentes aparecem são as mesmas em ambas as figuras (a partir da primeira linha da Figura 18 e da décima linha da Figura 19). Entretanto, na Figura 19 o endereço aparece

somente uma vez, pois o *script análise* é responsável por armazenar todos os endereços diferentes da aplicação, ou seja, os endereços aparecem em ordem cronológica, enquanto que na Figura 18 os endereços aparecem na ordem real da execução da aplicação.

A Figura 19 exibe as características de cada endereço, essas características são separadas por campos (valores/caracteres entre os "|"). A letra R significa operação de leitura (*Read*) e a letra W operação de escrita (*Write*), um exemplo das características de cada endereço é demonstrado na Tabela 7, para a linha 10 da Figura 19, onde observa-se o significado de cada campo.

Tabela 7 – Significados dos campos para o *script análise*

Linha 10		"0xdf8c0088|1603|1308|295|15|1|15|R|1|R|8"
Onde, cada campo significa:		
Número do Campo	Valor Contido no Campo	Significado do Campo
1	0xdf8c0088	Endereço
2	1603	Total de acessos (leitura + escrita)
3	1308	Total de acessos leitura
4	295	Total de acessos escrita
5	15	Maior valor consecutivo de leituras
6	1	Maior valor consecutivo de escritas
7	15	Maior valor consecutivo entre leituras e escritas
8	R	Primeira operação realizada
9	1	Maior valor consecutivo da primeira operação
10	R	Última operação realizada
11	8	Maior valor consecutivo da última operação

A saída do *script análise* serve de entrada para o *script análise alocação estática* (Figura 21). Este último script tem o objetivo de alocar os endereços nas memórias utilizadas, de acordo com as vantagens e desvantagens de cada tipo de memória e do número de acessos referentes às operações de leitura e escrita identificados anteriormente pelo *script análise.*

A Figura 23 ilustra um trecho da saída do *script análise alocação estática,* considerando as memórias PCM, STT-RAM, SRAM e DRAM para a aplicação FFT. Observa-se neste trecho os mesmos endereços existentes que foram demonstrados na Figura 19, e como estes foram alocados para as diferentes memórias. As linhas anteriores a 44 foram suprimidas da Figura 23, pois estas descreviam informações sobre as memórias e outras características da análise.

```
44  0xdf8c0088|1603|1308|295|15|1|15|R|1|R|8|STT|0|0|1|0
45  0xc036c35d|1705|1705|0|1705|0|1705|R|1705|R|1705|PCM|0|0|0|1
46  0xc036c334|1705|1705|0|1705|0|1705|R|1705|R|1705|PCM|0|0|0|1
47  0xc036c380|590|295|295|1|1|1|R|1|W|1|SRAM|0|1|0|0
48  0xf8010004|714|714|0|714|0|714|R|714|R|714|PCM|0|0|0|1
49  0xdf8c0084|1110|1013|97|42|1|42|R|6|R|2|STT|0|0|1|0
50  0xc036c368|285|285|0|285|0|285|R|285|R|285|PCM|0|0|0|1
51  0xdf8c0000|188|188|0|188|0|188|R|188|R|188|PCM|0|0|0|1
52  0xc0311ec8|242|121|121|1|1|1|W|1|R|1|SRAM|0|1|0|0
53  0xc0311ecc|251|125|126|1|2|2|W|1|R|1|SRAM|0|1|0|0
54  0xc0311ed0|261|127|134|1|3|3|W|1|R|1|SRAM|0|1|0|0
55  0xc0311ed4|261|127|134|1|3|3|W|1|R|1|SRAM|0|1|0|0
56  0xdf8cb09c|971|877|94|40|1|40|R|3|R|4|STT|0|0|1|0
57  0xc0311ebc|53|25|28|1|2|2|W|1|R|1|SRAM|0|1|0|0
58  0xc0311ec0|47|26|21|6|1|6|W|1|R|1|SRAM|0|1|0|0
59  0xc0311ec4|217|108|109|1|2|2|W|1|R|1|SRAM|0|1|0|0
60  0xdf8cb0e4|1203|401|802|1|2|2|R|1|W|2|SRAM|0|1|0|0
61  0xc0311eb0|35|16|19|1|2|2|W|1|R|1|SRAM|0|1|0|0
62  0xc0311e8c|12|6|6|1|1|1|W|1|R|1|MP|1|1|0|0
63  0xc0311e90|14|7|7|1|1|1|W|1|R|1|MP|1|1|0|0
64  0xc0311e94|28|14|14|1|1|1|W|1|R|1|MP|1|1|0|0
```

Figura 23 – Recorte da saída do *script análise alocação estática* para a aplicação FFT.

Na Figura 23, nota-se as características inseridas pelo *script análise alocação estática* a partir do campo 12 (campos anteriores já foram mencionados pelo resultado obtido no *script análise)* para as linhas 44 a 64. A Tabela 8 exibe as características inseridas pelo *script análise alocação estática*, utiliza-se como exemplo, o endereço contido na linha 44 da Figura 23 (mesmo endereço apresentado anteriormente na linha 10 da Figura 19).

Tabela 8 – Significados dos campos para o *script análise alocação estática*

Linha 44		“0xdf8c0088|1603|1308|295|15|1|15|R|1|R|8|STT|0|0|1|0”
Onde, o campo 12 e os seus subsequentes significam:		
Número do Campo	Valor Contido no Campo	Significado do Campo
12	STT	Tipo de memória que o endereço está alocado
13	0	Número de alocações ocorridas pelo endereço na MP
14	0	Número de alocações ocorridas pelo endereço na SRAM
15	1	Número de alocações ocorridas pelo endereço na STT-RAM
16	0	Número de alocações ocorridas pelo endereço na PCM

A saída do *script análise alocação estática* de todos os *traces* empregados neste trabalho, são responsáveis pelos resultados da alocação estática. O fluxograma detalhado do *script análise alocação estática* é ilustrado pela Figura 24.

Essa figura dá uma ideia abrangente do funcionamento do *script* para a realização das alocações dos endereços nas memórias, segundo a abordagem estática.

Pela Figura 24, por exemplo, dado um endereço e utilizando todas as memórias empregadas no trabalho. Nota-se que, o endereço atual será inserido primeiramente na PCM se satisfazer as características/vantagens dessa inserção nesta memória. Caso contrário, o endereço atual será inserido em outra memória seguindo a ordem ilustrada na Figura 22 ou pela Figura 24, respeitando as características para as inserções nas memórias (mencionadas na Seção 4.2.1) e as condições da Figura 24. Ainda nesta figura, percebe-se que, quando o endereço atual respeitar as características para a inserção em alguma das memórias e está memória estiver sem espaço para a alocação deste endereço. O *script* percorre a memória buscando o endereço que possui a menor vantagem/característica desta memória. Se o endereço atual possuir uma vantagem maior que o endereço que está inserido na memória, o endereço atual é inserido no lugar do endereço que possui a menor vantagem, e esse endereço que foi substituído deve ser alocado em um nível inferior na hierarquia de memória. Após a inserção do endereço substituído e do endereço atual em algum tipo de memória, ou quando somente o endereço atual for inserido em alguma memória, o *script* atualiza o endereço atual com o próximo endereço (novo endereço, Figura 24) do *trace* que está sendo analisado.

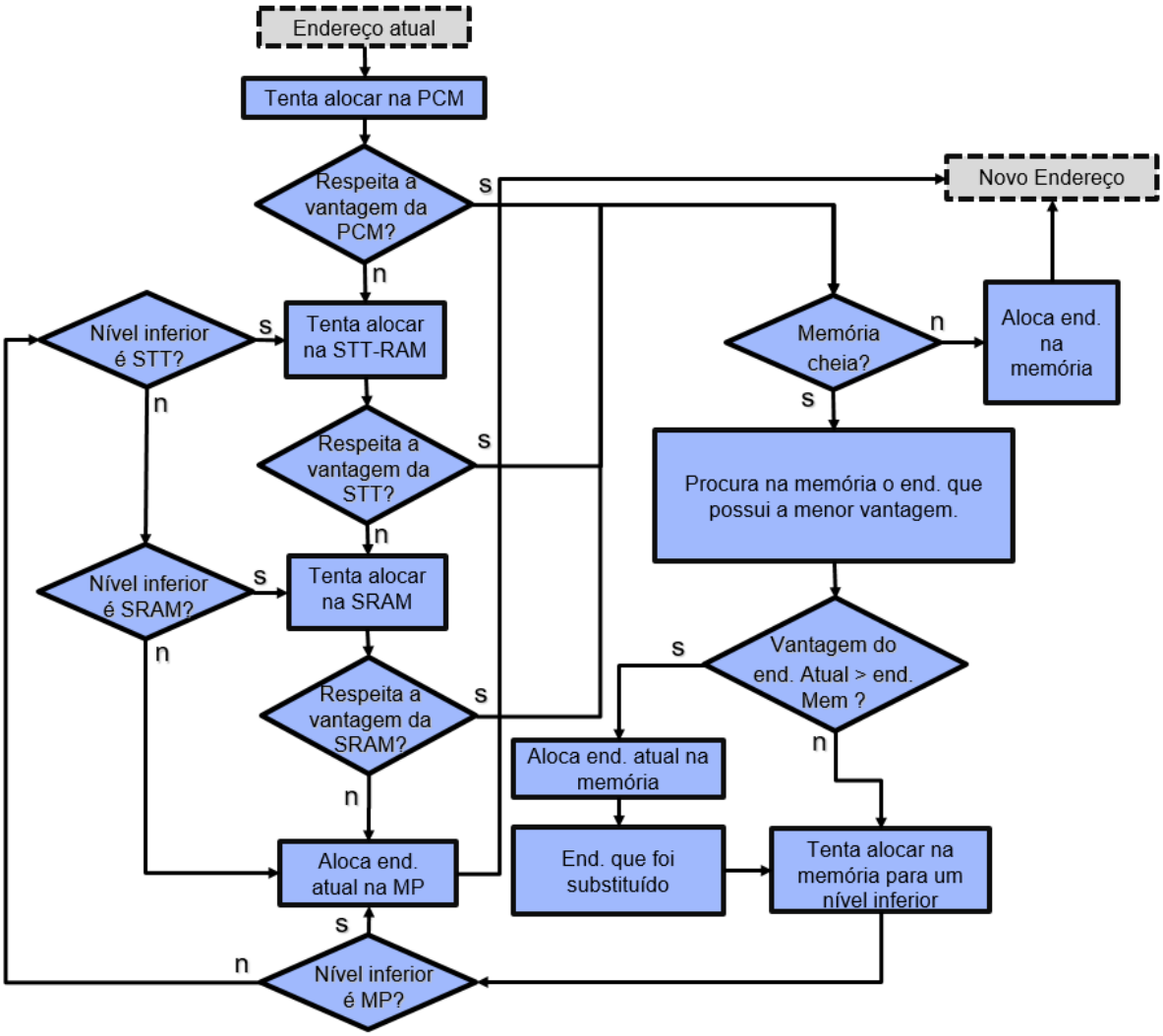


Figura 24 – Fluxograma do *script análise alocação estática*.

4.2.1.2 Análise Dinâmica

Utilizando o *trace* de dados de um *benchmark* para realizar as alocações ocorridas na hierarquia de memória, utilizando a abordagem dinâmica, é necessário somente a utilização de um *script* (*script análise alocação dinâmica)* como pode ser observado na Figura 21.

O *script análise alocação dinâmica* tem o objetivo de alocar os endereços nas memórias utilizadas, de acordo com as vantagens e desvantagens de cada tipo de memória e do número de acessos das operações de leitura e escrita que está sendo atualizado ao decorrer da execução da aplicação pelo próprio *script análise alocação dinâmica.* Assim, as alocações dos endereços nas respectivas memórias utilizadas ocorrem em tempo de execução do *benchmark* para a arquitetura ARM, ao contrário da análise estática que precisava de um *script* intermediário. Através disso, essa

análise precisa de um período maior para a obtenção dos resultados quando comparada com a análise estática.

A Figura 20 exibe um fragmento da saída do *script análise alocação dinâmica* para a aplicação FFT. A saída desse *script* possui algumas características a mais para os endereços quando comparada com a saída do *script análise alocação estática*. Na abordagem estática já foram apresentados os campos inferiores ao 12 (no *script análise)*. A Tabela 9 demonstra as características do *script análise alocação dinâmica*, utiliza-se como exemplo, o endereço contido na linha 44 da Figura 20. As diferenças entre o *script análise alocação estática* e o *script análise alocação dinâmica* é a inserção dos campos 13 ao 20 (Tabela 9) no *script* da abordagem dinâmica.

Tabela 9 – Significados dos campos para o *script análise alocação dinâmica*

Linha 44		"0xdf8c0088|1603|1308|295|15|1|15|R|1|R|8|STT|1603|0|1601|2|4|1599|1602|1|0|1|1|1"
Onde, os campos 13 ao 20 significam:		
Número do Campo	Valor Contido no Campo	Significado do Campo
12	STT	Tipo de memória que o endereço está alocado
13	1603	Número de miss na MP
14	0	Número de hit na MP
15	1601	Número de miss na SRAM
16	2	Número de hit na SRAM
17	4	Número de miss na STT-RAM
18	1599	Número de hit na STT-RAM
19	1602	Número de miss na PCM
20	1	Número de hit na PCM
21	0	Número de alocações ocorridas pelo endereço na MP
22	1	Número de alocações ocorridas pelo endereço na SRAM
23	1	Número de alocações ocorridas pelo endereço na STT-RAM
24	1	Número de alocações ocorridas pelo endereço na PCM

Os *misses* e *hits* das memórias foram utilizados para os testes realizados, com o intuito de identificar em qual memória o endereço está contido naquele momento da execução da aplicação/*benchmark.* Percebe-se que, o número de (*miss* + *hit*) para cada memória é igual ao número total de acessos, pois, quando ocorre a busca do endereço pela CPU nas memórias, a procura pelo endereço ocorre paralelamente entre as memórias utilizadas.

A saída do *script análise alocação dinâmica* de todos os *traces* utilizados neste trabalho, são responsáveis pelos resultados da alocação dinâmica. O fluxograma do *script análise alocação dinâmica* é descrito a seguir pela Figura 25.

Essa figura é uma ideia abrangente do funcionamento do *script* para a realização das alocações dos endereços nas memórias.

A Figura 24 tem uma forte semelhança com a Figura 25. A exemplificação utilizada na Figura 24 pode ser adotada para a Figura 25, entretanto, como essas figuras são de abordagens diferentes, possuem implementações diferentes de seus *scripts* de alocações. O fluxograma da Figura 25 possui algumas condições inseridas para as alocações dos endereços nas memórias quando comparada com o fluxograma da Figura 24. Os endereços que chegam para serem alocados (endereço atual), para abordagem estática são todos diferentes, enquanto que para a abordagem dinâmica podem ser iguais. Assim, o fluxograma da Figura 25 é mais complexo, pois dado um endereço para ser alocado (endereço atual), esse endereço deve ser testado, já que pode estar contido em alguma memória utilizada.

Na Figura 25, se o endereço atual já estiver em alguma memória, os dados deste endereço devem ser atualizados (total de acessos, totais de escritas e leituras). Após a atualização dos dados, se o endereço atual ainda satisfazer as características da memória que ele está contido, o endereço atual permanecerá na memória. Caso contrário, o endereço atual deve ser desalocado e inserido em uma memória de nível hierárquico inferior. Para a inserção do endereço atual (desalocado) em uma memória de mais baixo nível, primeiramente o endereço atual deverá satisfazer as características da memória do nível inferior. Caso contrário, o endereço deve ser alocado em um nível mais inferior da hierarquia de memória.

Se o endereço atual (desalocado) satisfazer os requisitos da memória mais baixa, o *script* de alocação percorre esta memória inferior, buscando o endereço que possui a maior vantagem/característica para o tipo de memória que o endereço atual (desalocado) estava contido anteriormente. Se o *script* encontrar um endereço na memória inferior que satisfaça as condições da memória superior (do endereço atual desalocado), o endereço da memória inferior é transferido para a memória superior na posição do endereço atual (desalocado) e o endereço atual (desalocado) é transferido para o local do endereço do nível inferior da memória. Se o *script* não encontrar um endereço na memória inferior que satisfaça as condições da memória superior (do endereço atual desalocado), o endereço atual (desalocado) deve ser alocado em um nível mais inferior da hierarquia de memória.

Por fim, o endereço atual deve ser atualizado para o próximo endereço (novo endereço, Figura 25) do *trace* que está sendo analisado. Para isso a análise

dinâmica utiliza uma abordagem semelhante ao da análise estática (mencionado anteriormente na Seção 4.2.1.1), entretanto a análise dinâmica possui algumas diferenças para a atualização do endereço atual. Além das características mencionadas na análise estática, a análise dinâmica pode atualizar o endereço atual para um novo endereço do *trace*, quando o endereço atual já estiver em alguma memória utilizada. Para isso, o endereço atual que está nessa memória pode ser desalocado da memória e inserido em outra memória que está em um nível hierárquico menor, ou o endereço atual é somente atualizado na memória que está contido inicialmente.

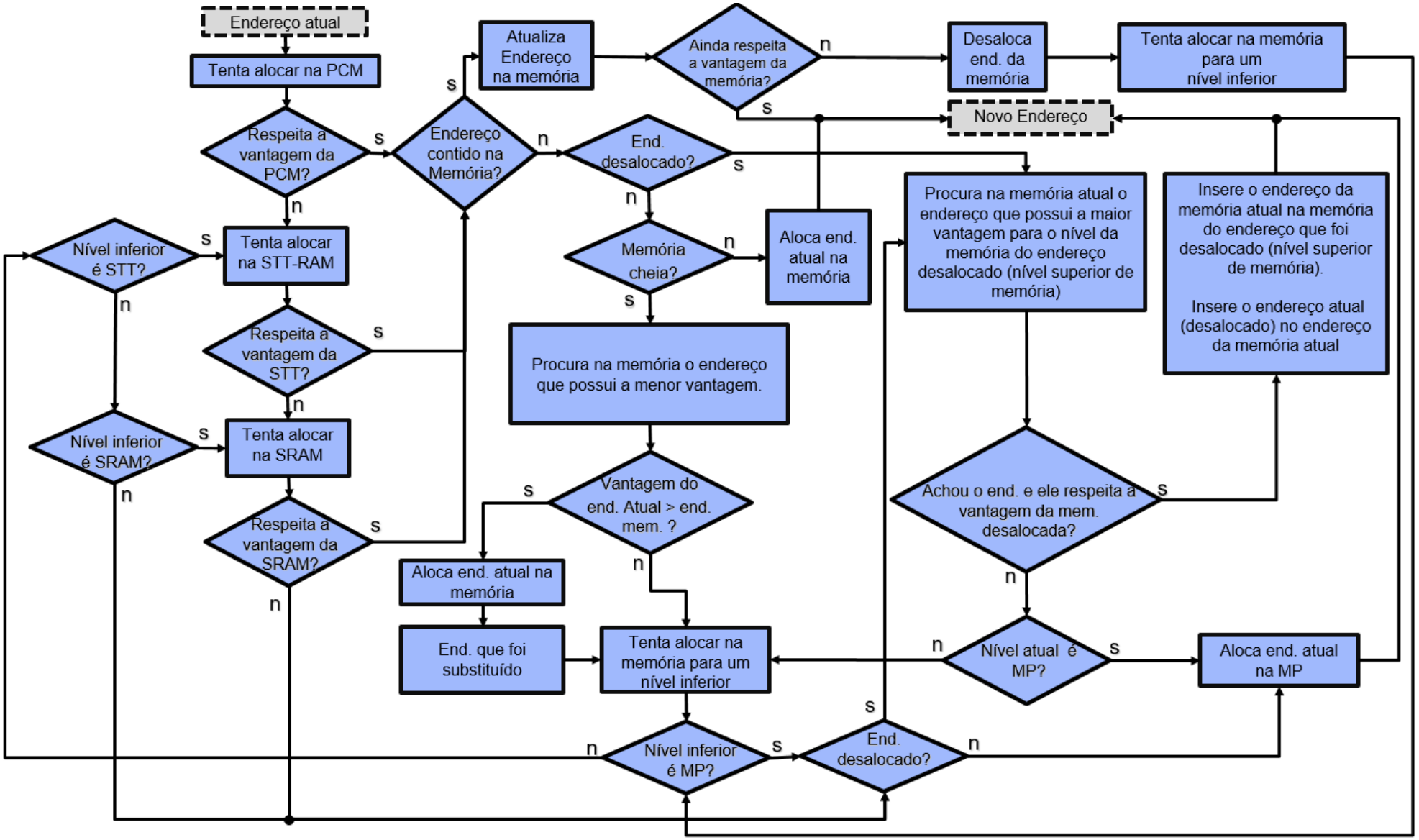


Figura 25 – Fluxograma do *script análise alocação dinâmica*.

4.2.1.3 Diferença entre a Análise Estática e Dinâmica

A diferença principal entre as análises é que a abordagem dinâmica acontece durante da execução da aplicação, enquanto que a abordagem estática acontece depois da execução da aplicação (como fosse uma análise off-line prévia). Assim, a abordagem estática emprega um *script* (*script análise*) para a realização de uma pré-análise da aplicação antes da alocação.

Comparando os fluxos detalhados nas Figuras 24 e 25, percebe-se que a abordagem dinâmica é mais complexa. Isso acontece porque, nessa abordagem o endereço que chega para ser alocado (endereço atual) já pode estar alocado em alguma memória, assim ele deve ser atualizado e se não respeitar mais a característica da memória que ele está inserido, o endereço deve ser desalocado e inserido em outra memória. Como a análise estática realiza uma pré-análise do *trace* da aplicação, os endereços que chegam para serem alocados nas memórias são todos diferentes, tornando essa última abordagem mais simples de ser implementada.

São apresentados no Capítulo 5 primeiramente os resultados da abordagem dinâmica para os experimentos realizados, e no final do capítulo é realizada a comparação dos resultados das duas análises para os experimentos abordados.

4.2.2 Benchmarks Analisados

Em GUTHAUS et al. (2001) é apresentado o conjunto de *benchmarks MiBench*, os quais são divididos em seis categorias, sendo: automóveis e controle industrial, rede, segurança, dispositivos de consumo, escritório e telecomunicações. Estas categorias oferecem características diferentes, que permitem que pesquisadores avaliem a eficiência de seus projetos para um segmento específico do mercado. A Tabela 10 lista todos os *benchmarks* do *MiBench*, classificando-os nas categorias anteriormente mencionadas.

Tabela 10 – Classificação dos *benchmarks* do *MiBench* dentro de suas correspondentes categorias

Auto/Industrial	Consumer	Office	Network	Security	Telecomm.
Basicmath *	jpeg *	ghostscript	dijkstra *	blowfish enc.	CRC 32 *
Bitcount *	lame	ispell *	patrícia *	blowfish dec.	FFT *
qsort *	Mad *	rsynth	(CRC 32)	pgp sign	IFFT
susan (edges) *	tiff2bw *	sphinx	(sha)	pgp verify	ADPCM enc.
susan (corners)	tiff2rgba *	stringsearch *	(blowfish)	rijndael enc. *	ADPCM dec.
susan (smoothing)	tiffdither			rijndael dec.	GSM enc.
	tiffmedian			sha *	GSM dec.
	typeset *				

Fonte: GUTHAUS et al. 2001.

Dentre todos os *benchmarks* listados na Tabela 10, foram escolhidos para este trabalho alguns aleatoriamente, pois o tempo requerido para a obtenção dos resultados (tempo de compilação) inviabilizou a avaliação de todo o conjunto. Visando uma certa representatividade, a preferência na escolha foi de avaliar pelo menos um *benchmark* por categoria. Sendo assim os escolhidos foram: *basicmath, bitcount, crc32, dijkstra, FFT, ispell, jpeg, mad, patrícia, qsort, rijndael, sha, stringsearch, susan, tiff2bw, tiff2rgba e typeset.* Esses benchmarks foram destacados com asteriscos na Tabela 10.

4.2.3 Arquitetura Analisada

A arquitetura escolhida para rodar os *benchmarks* selecionados foi a ARM. A arquitetura ARM é conhecida primeiramente como *Acorn* RISC (*Reduced Instruction Set Computer*) *Machine*, mas atualmente possui o nome de *Advanced* RISC *Machine*. O ARM possui uma arquitetura de processador original de 32 bits, foi desenvolvido para apresentar o melhor desempenho possível, ocupar pouca área e ter baixo consumo energético. A versão mais recente desta arquitetura usa palavras de 64 bits.

Os processadores ARM como são arquiteturas RISC, possuem um número reduzido de instruções para a programação. São encontrados em diversos aparelhos como por exemplo, em celulares, calculadores, automóveis, aplicações industriais. Suas características principais são:

- Arquitetura *Load-Store*: as instruções processarão somente operações aritméticas dos valores que estiverem nos registradores e sempre armazenarão os resultados em algum outro registrador.
- Formato de instruções de três endereços: dois registradores operandos e um registrador resultado.
- Instruções fixas de 32 bits de largura (em exceção as instruções *Thumb* compactadas de 16 bits) alinhadas em 4 bytes consecutivos da memória. Instruções com execução condicional, deslocamentos, carga e armazenamento de múltiplos registradores.
- Possui 15 registradores de 32 bits, para o uso geral.
- Manipulação de periféricos de I/O como dispositivos mapeados na memória com suporte à interrupção.
- Pipeline de 3 e 15 estágios.
- Baixo consumo energético.
- Tamanho do núcleo reduzido.

Segundo Virtutech (2007) o Simics, ferramenta empregada neste trabalho, suporta diversos tipos de arquiteturas de processadores, segue abaixo a arquitetura analisada neste trabalho:

- ARM SA1110: Modela um sistema monoprocessado com processador ARMv5 (*Intel Strong ARM*), oferece suporte nativo ao Linux *kernel* 2.4.12. Este modelo de processador ARMv5 é genérico com implementações mínimas.

4.2.4 Ferramentas de Simulação e Estimativa

Para a avaliação do impacto da hierarquia de memória no desempenho bem como no consumo energético relacionado à execução de uma dada aplicação, é necessário o emprego de ferramentas de simulação e estimativa. Esta seção apresenta as ferramentas utilizadas no trabalho. O Simics foi utilizado para a obtenção dos *traces* dos *benchmarks* para a arquiteturas ARM, o NVSim e Cacti foram utilizados para a obtenção dos parâmetros das memórias, como por exemplo, latência, energia, *leakage* e área.

4.2.4.1 Simics

O Simics (MAGNUSSON et al., 2002) possui muitos aspectos relevantes para as simulações das arquiteturas de processadores embarcados, a seguir será explicado a origem desses aspectos relevantes e os benefícios de sua utilização. O Simics é uma plataforma de desenvolvimento, modelagem e simulação de arquiteturas completas de microcomputadores (MAGNUSSON et al., 2002), fabricado originalmente pelo Instituto de Ciência da Computação da Suécia, e após desenvolvido pela *Virtutech* fundada em 1998. A *Virtutech* foi adquirida pela Intel em 2010 e o Simics agora é comercializado pela *Wind River Systems*, subsidiária da Intel.

A ferramenta provê hardware virtual, que executa os mesmos binários de um hardware físico. Com isso, é possível selecionar uma máquina qualquer, desenvolver algum software, e realizar simulações, avaliando o desempenho de uma arquitetura virtual configurável quando executando o referido software. Isso tudo reduz custos e agiliza o processo de desenvolvimento, dado que não é necessário ter o hardware físico para realizar estas simulações.

Segundo Engblom e Ekblom (2006), o Simics é um simulador *full-system*, toda a arquitetura modelada nele é programável, como por exemplo, processadores, memórias, conjunto de instruções, entre outras. Além disso, o pacote original do programa já possui diversos *targets*, arquiteturas programáveis prontas, como ARM, Malta, PowerPC, entre outras. Essas arquiteturas podem ser modificadas, com isso é possível modelar uma arquitetura do zero ou alterar algum *target* para construir a arquitetura desejada.

O simulador não modela todos os componentes de um hardware real, pois este foca em simulação no nível de instruções. Segundo Virtutech (2007) normalmente a execução no simulador é de uma instrução por ciclo (parâmetro definido pelo IPC padrão da ferramenta), enquanto uma máquina real consegue gerenciar a execução de mais instruções em apenas um ciclo, devido ao fato de utilizarem a superescalaridade e outros componentes de desempenho.

Além disso, diversos outros componentes não podem ser totalmente simulados, como é o caso dos comportamentos do sistema e barramentos, como por

exemplo, os gargalos de acesso aos recursos. Assim, fica claro que o objetivo do simulador é simular máquinas no nível de conjuntos de instruções e não realizar simulações exatas de todos os dispositivos que englobam o hardware em questão. Vale destacar que a simulação suportada pelo Simics não é adequada para a comparação direta entre uma máquina real e uma máquina simulada, devido ao fato que a diferença entre uma máquina real e uma máquina simulada depende de diversos fatores. Desta forma, deve-se considerar que haverá uma diferença de tempos de execução entre o sistema real e o simulado.

Entretanto, a ferramenta Simics é útil para a comparação de cargas de trabalho em diferentes configurações, que são modeladas dentro do próprio simulador, uma vez que as tendências ocorridas nas simulações devem se repetir em sistemas reais. Além disso, as arquiteturas modeladas pelo usuário podem ser executadas rapidamente com o auxílio de um *script*, que altere a configuração do hardware virtual da ferramenta. Logo, a ferramenta fornece um ambiente controlado e com determinismo controlado, propício para avalição de sistemas computacionais futuros.

A ferramenta Simics é utilizada por diversas empresas como Intel, AMD e outros fabricantes para a realização de simulações e testes de desempenho em suas máquinas, devido a facilidade que a ferramenta proporciona para as simulações. Algumas empresas fornecem aos programadores os *scripts* de novas arquiteturas que ainda estão sendo desenvolvidas, com o objetivo realizarem testes com antecedência em programas e ferramentas que irão executar nestas novas arquiteturas.

Para a simulação mais detalhada, a ferramenta Simics apresenta alguns modos de operações, *fast* (rápido), *stall* (lento) e *mai* (microarquitetura). Sendo assim, é possível obter informações a respeito de tempo, passos de execução, ciclos de execução, registradores em diferentes graus de complexidade, detalhamento e velocidades de simulação, endereços virtuais ou físicos.

4.2.4.2 Ferramenta NVSim

Enquanto existem muitas ferramentas disponíveis para as simulações de memórias SRAMs e DRAMs, as ferramentas semelhantes para as memórias NVMs

atualmente são poucas. Para as simulações de NVMs, destacam-se as ferramentas NVSim (DONG et al., 2012) e NVMain (POREMBA, ZHANG e XIE, 2015), que espera-se que ajudem a aumentar as pesquisas envolvendo as NVM em diferentes níveis da arquitetura de memória. Segundo Dong et al. (2012), o NVSim vem sendo validado com sucesso com protótipos NVM industriais e por esta razão, como mencionado anteriormente, o NVSim será o simulador empregado neste trabalho.

O NVSim modela a área, o tempo, a energia dinâmica e o *leakage* das memórias PCM, STT-RAM, ReRAM, FBDRAM (*Floating Body Dynamic Random-Access Memory*) e SLC (*Single-Level Cell*) NAND *flash*. O NVSim é uma extensão generalizada do PCRAMsim e usa os mesmos princípios de modelagem que o Cacti, mas possui como base um novo *framework*, com maior flexibilidade. Além das memórias mencionadas, assim como o Cacti, o NVSim também tem a capacidade de modelar SRAMs.

4.2.4.3 Ferramenta Cacti

Cacti é uma ferramenta que simula *caches* e memórias principais (THOZIYOOR, et al., 2008). A ferramenta demonstra o tempo de acesso à memória, o número de ciclo por acesso, a área da memória, o consumo energético por acesso, entre outros parâmetros. Cacti facilita e ajuda a entender as melhores vantagens e desvantagens de desempenho das organizações do sistema de memória para arquiteturas de computadores.

A ferramenta está disponível em duas formas: uma versão baseada na web, e uma versão com código-fonte (C ++). A versão com interface na web é frequentemente atualizada com as últimas versões e correções de *bugs*. A interface web da versão do CACTI 5.3 pode ser acessada facilmente através de um link (CACTI, 2014), assim permitindo, que a ferramenta Cacti seja acessível a um elevado número de usuários. A versão da web visa atender às necessidades da maioria dos usuários, enquanto a versão do código fonte, pode atender demandas mais específicas, pois através de *scripts*, parâmetros de simulação podem ser redefinidos.

A ferramenta está sendo continuamente atualizada, com isso, os resultados das simulações para uma determinada configuração de memória ou tecnologias de

caches específicas podem sofrer alterações à medida que as novas versões são lançadas e seus *bugs* corrigidos. O Cacti conta com uma atuante comunidade internacional, a qual através de fóruns de discussão, soluciona dúvidas de seus usuários.

4.2.5 Experimentos realizados

Neste presente trabalho são realizados três experimentos, o primeiro é descrito na Seção 4.2.5.1 e os outros dois experimentos são descritos na Seção 4.2.5.2 Para cada experimento realizado são analisados 18 *traces* de diferentes aplicações.

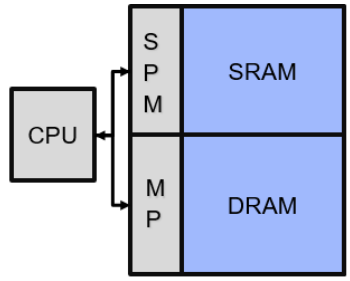
4.2.5.1 Experimento 1 - Verificação dos endereços e acessos

Para primeiro experimento foi verificada a necessidade de identificar o maior tamanho de memória necessária para alocação dos endereços da aplicação que possui o maior número de endereços (pior caso). Cada endereço possui 4 bytes (no pior caso), assim, identificando o número de endereços diferentes para uma determinada aplicação é possível identificar o tamanho de memória necessária para armazenar todos os endereços dessa aplicação.

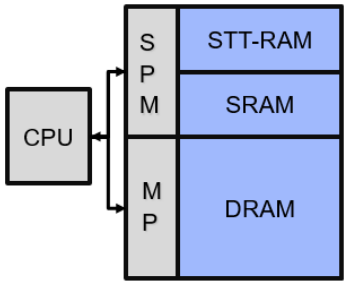
Além disso, nesse experimento é realizado a comparação de memórias de 1Kbytes até 4 Mbytes que possuem os endereços mais acessados, assim, sabe-se o percentual de acessos que um determinado tamanho de memória possui para os endereços com maior frequência de acessos. Também é verificada a quantidade de operações de escrita e leitura, com o intuito de identificar qual a maior operação ocorrida e o percentual de cada uma delas.

4.2.5.2 Experimento 2 e 3 - Exploração de diferentes SPMs

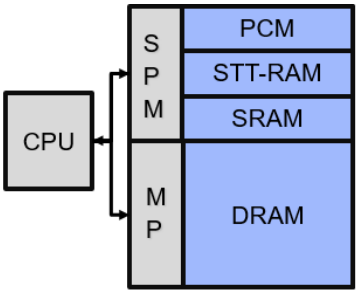
Para o segundo e terceiro experimentos são utilizados diversos tipos de SPMs, as mesmas SPMs foram mencionadas anteriormente na metodologia deste trabalho. As arquiteturas utilizadas nesses experimentos estão sendo exibidas na Figura 26.



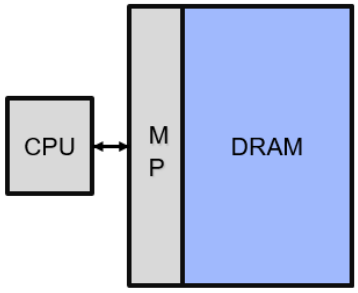
(a) Arquitetura com a SPM utilizando SRAM.



(b) Arquitetura com a SPM utilizando SRAM e STT-RAM



(c) Arquitetura com a SPM utilizando SRAM, STT-RAM e PCM.



(d) Arquitetura sem a utilização de SPM

Figura 26 – Arquitetura analisadas no trabalho.

O segundo experimento consiste em realizar a comparação entre diversos tipos de SPMs, para isso, foi utilizada as arquiteturas (*a), (b)* e *(c)* da Figura 26. Através dessas comparações entre as SPMs é possível identificar as influências das NVMs inseridas nas SPMs no consumo de energia e no desempenho. As arquiteturas (*a*), (*b*) e (*c*), possuem diferentes configurações de tamanhos de memórias. Para o segundo experimento como estudo de caso são utilizadas cinco SPMs diferentes, essas SPMs estão sendo demonstradas na Tabela 3.

Para a comparação entre as cinco versões de SPMs é utilizada a SPM com 32KB de SRAM como linha base para o experimento 2. Para a realização do experimento 3 são utilizadas as arquiteturas (*a), (b), (c) e (d)* da Figura 26. No experimento 3, são utilizadas as mesmas SPMs do experimento 2, no entanto, a linha base para esse último experimento (experimento 3) é a arquitetura sem a utilização de SPMs, letra *(d)* da Figura 26. O experimento 3 consiste em realizar a comparação das aplicações avaliadas neste trabalho utilizando as SPMs e sem a utilização dessas memórias, assim, é possível identificar a influência das SPMs no desempenho e no consumo energético total das aplicações.

5 RESULTADOS EXPERIMENTAIS

Neste capítulo são apresentados os resultados obtidos a partir da execução das aplicações selecionadas utilizando a metodologia da Seção 4.2. O capítulo possui três grandes seções: verificação dos endereços e acessos (experimento 1), comparação entre as diferentes SPMs (experimento 2 e 3) e a comparação entre os resultados da análise dinâmica e estática. Nos experimentos 2 e 3, optou-se pela utilização da abordagem dinâmica, pois nesta análise as alocações dos endereços nas memórias utilizadas ocorrem durante a execução das aplicações analisadas, possibilitando assim, resultados mais legítimos do que a abordagem estática. Na análise estática além das alocações dos endereços nas memórias ocorrerem após a execução da aplicação, para a realização desta abordagem é necessária a utilização de *script análise*, responsável por realizar uma pré-análise da aplicação antes da alocação dos endereços nas memórias.

5.1 Experimento 1 - Verificação dos endereços e acessos

O primeiro experimento consiste na verificação dos endereços ocorridos para os *benchmarks* analisados. A verificação possui como objetivos a identificação do número de endereços, o tipo de acesso desse endereço e a quantidade desses acessos para cada aplicação. Todos os resultados desse experimento podem ser encontrados nos apêndices A e B.

A Figura 27 ilustra resultados referentes a quantidade de endereços diferentes, considerando todas as aplicações. Esta figura demonstra o menor valor, o maior valor e o valor médio de endereços diferentes dos *benchmarks* analisados. O *benchmark* que obteve o menor número de endereços diferentes foi o *stringsearch* e a que obteve o maior número de endereços diferentes foi o *qsort*.

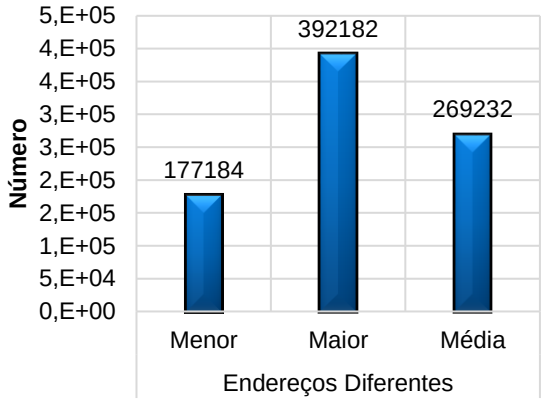


Figura 27 – Quantidade de endereços diferentes para todas as aplicações.

A Figura 28 apresenta a média do total de acessos a endereços realizados pelas aplicações, assim como menor valor e maior valor de total de acessos. A figura, também demonstra o número de acessos para as operações de leitura e de escrita. Percebe-se, que o número de acessos para a operação de leitura é superior ao da operação de escrita em todos os casos (menor valor, maior valor e valor médio) desses acessos.

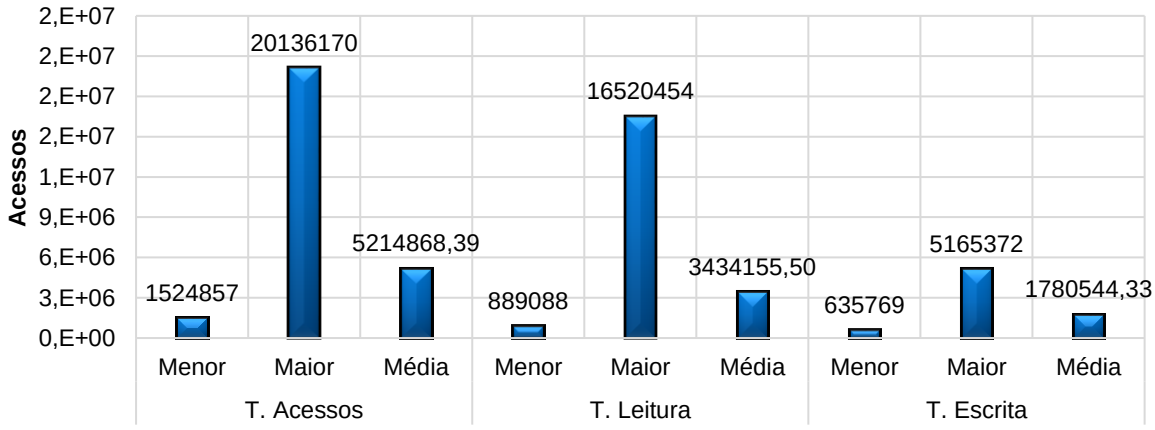
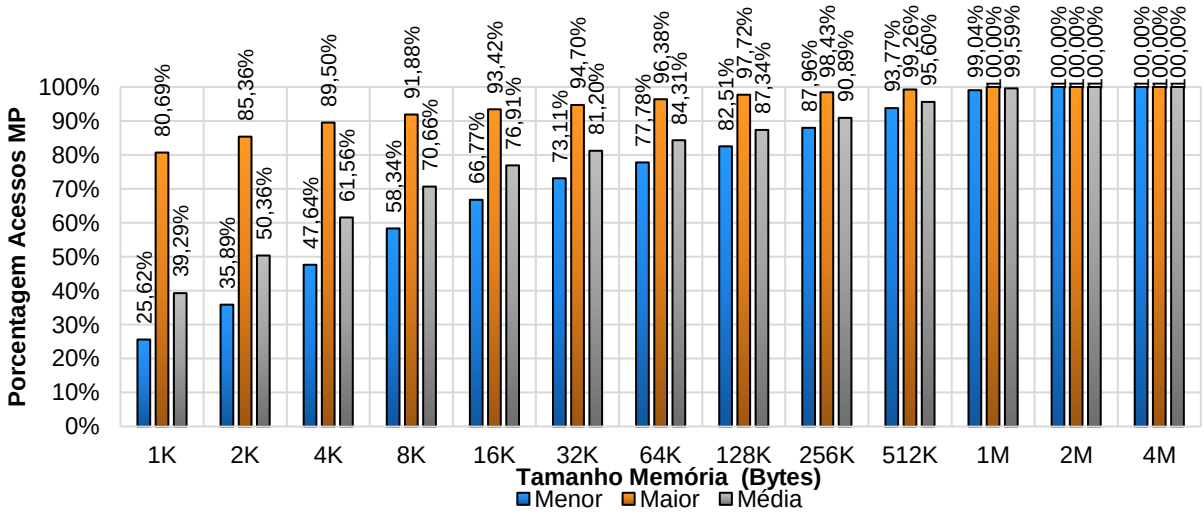


Figura 28 – Total de acessos, total das operações de escritas e leituras.

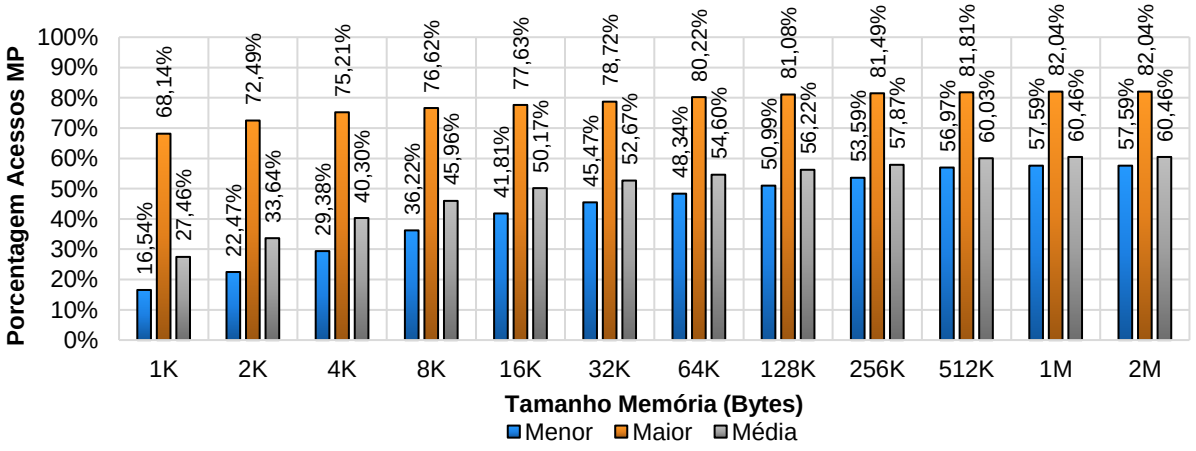
Utilizando os endereços que possuem o maior número de acessos, é realizada a comparação de memórias com tamanho entre 1Kbytes até 4Mbytes. Os endereços que possuem uma maior frequência de acessos são inseridos nas memórias menores, posteriormente, os endereços que não possuem uma frequência elevada de acessos são inseridos nas próximas memórias. A Figura 29 (a) demostra diversos tamanhos de memórias com a respectiva porcentagem de acessos que essas memórias possuem. Por exemplo, na Figura 29 (a) uma memória de apenas

16KB é responsável por possuir em média 76,91% de todos os acessos realizados por todas as aplicações analisadas.

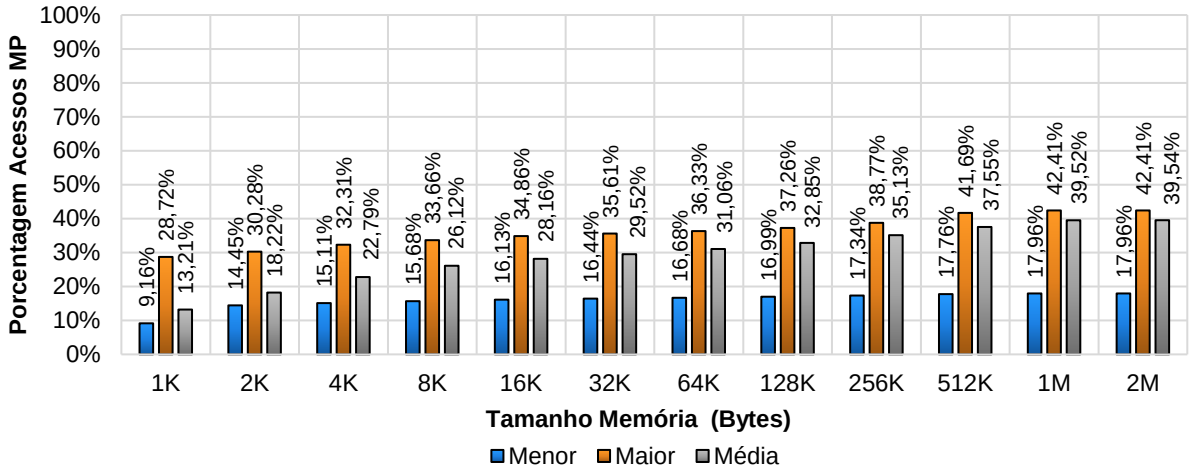
Na Figura 29 (a), nota-se, que houve uma normalização dos resultados após o tamanho de 2MB para a memória. Assim, uma memória possuindo 2MB, já é responsável por armazenar todos os endereços de cada aplicação. Isso, também pode ser verificado pela Figura 27, onde, o maior valor de endereços diferentes quando multiplicado pelo tamanho de cada endereço 4bytes (pior caso), representa uma memória superior a 1MB e inferior a 2MB. Através disso, como o exemplo da Figura 29 (a), é realizada também a comparação de memórias de 1Kbytes até 2Mbytes para os endereços que possuem os maiores acessos das operações de leituras e escritas. A Figura 29 (b) representa o percentual dos endereços mais frequentemente acessados para os diferentes tamanhos de memórias para a operação de leitura e a Figura 29 (c) para as operações de escrita.



(a) Endereços mais acessados (operação de leitura e operação de escrita).



(b) Endereços mais acessados para a operação de leitura.



(c) Endereços mais acessados para a operação de escrita.

Figura 29 – Comparação de memórias utilizando os endereços que possuem uma maior frequência de acessos.

A Figura 30 apresenta somente os valores médios das Figuras 29 (b) e (c), só que desta vez, é realizada a comparação de memórias com tamanho entre 4 bytes (menor possível) até 4 Mbytes. Pelas figuras 30 e 29 (b, c), observa-se que para todos os tamanhos de memórias, que os percentuais de acessos para as operações de leituras são superiores aos percentuais das operações de escrita, anteriormente foi verificado na Figura 28, que o total de acessos de leitura é superior ao total de acessos de escrita. Isso é uma motivação para o uso de NVMs, devido ao fato, que as NVMs possuem vantagens nas operações de leitura quando comparadas com memórias tradicionais como SRAMs e DRAMs. Assim, NVMs podem ser empregadas como uma forma de otimização no acesso à memória para a arquitetura avaliada (ARM). Nos experimentos 2 e 3, cujos resultados são discutidos na Seção 5.2, este emprego será avaliado.

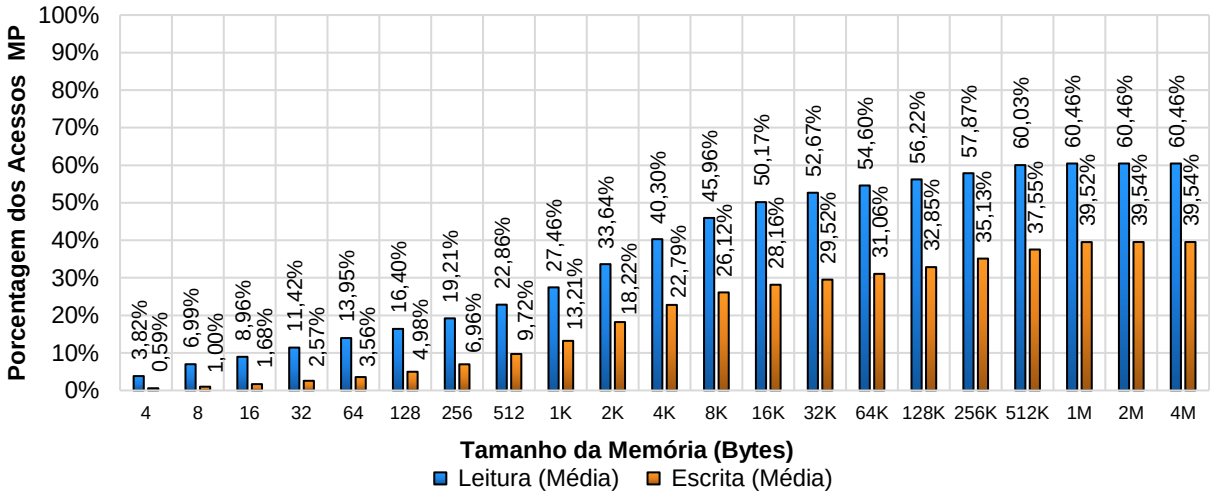


Figura 30 – Comparação dos endereços com maior frequência de acessos para a operação de leitura e escrita.

5.2 Experimentos 2 e 3 - Comparação entre as diferentes SPMs

Nesta seção são apresentados os resultados dos experimentos 2 e 3, nos quais emprega-se NVMs em diferentes tipos de SPMs (estudo de caso). O segundo e terceiro experimentos foram detalhados na Seção 4.2.5.2, as configurações das SPMs utilizadas foram apresentadas na Tabela 3. Lembrando que, para a realização do experimento 2 a linha base para as comparações é a SPM 1 (composta de 32 KB de SRAM) e para o experimento 3, a linha base das comparações é a arquitetura apresentada na Figura 26 (d), que utiliza somente a memória principal (DRAM), ou seja, sem o uso da SPM.

O trabalho tem como objetivo verificar o consumo energético e desempenho de diversas aplicações para arquiteturas híbridas de memória, como estudo de caso são utilizadas as SPMs. Partindo da linha base (SPM 1), a SPM 2 foi definida, com a inclusão de uma memória SST-RAM combinada a SRAM. Entretanto, verificou-se que a área da SPM 2 é maior que a linha base (Figura 31 (a)), com isso é utilizada a SPM 3, que possui um tamanho de memória STT-RAM menor do que o modelo anterior (SPM 2). Possibilitando assim, que a área total da SPM 3 seja inferior ao modelo da SPM 1. Na configuração SPM 4, uma memória PCM foi combinada às memórias SRAM e STT-RAM, e possuindo uma área menor quando comparada com a linha base. Sem a preocupação com a área, é criada a SPM 5, que além de possuir o mesmo tamanho de STT-RAM do modelo da SPM 2 possui também uma PCM como forma de otimização. Esta configuração (SPM 5) foi criada com o intuito

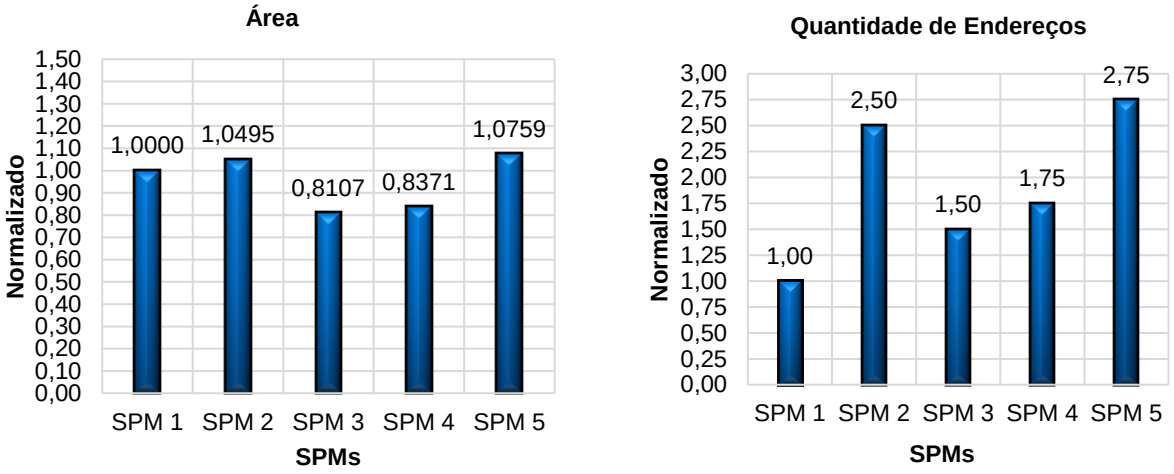
de examinar se existe vantagem em ampliar a área para adquirir melhores resultados no consumo energético ou no desempenho. Por esse motivo são realizadas comparações da SPM 5 com os modelos anteriores de SPMs. Através disso, busca-se identificar a ocorrência de grandes diferenças entre os resultados encontrados para as comparações dessas SPMs, justificando-se assim o aumento da área na SPM 5.

5.2.1 Experimento 2 - Comparação entre as diferentes SPMs

Nesta seção será apresentada a comparação entre as diferentes SPMs utilizadas no trabalho. Como linha base para esse experimento é utilizada a SPM 1 que possui 32 KB de SRAM para a comparação com as demais SPMs. Através dessa comparação é possível identificar as características de desempenho e consumo energético das NVMs inseridas nas SPMs quando comparadas com a linha base (SPM 1). Todos os resultados desse experimento podem ser encontrados no apêndice C.

A Figura 31 (a) ilustra a área para as diferentes configurações de SPMs empregadas no trabalho. Os resultados indicaram que a SPM 2 e SPM 5 possuem uma área maior, cerca de 4,95% e 7,59% respectivamente, quando comparadas com a SPM 1. Os modelos SPM 3 e SPM 4 possuem áreas menores quando comparadas com a linha base, cerca de 18,93% e 16,29%, respectivamente.

A diferença entre a SPM 2 e a SPM 5, é que a última SPM possui a memória PCM, sendo assim a sua área é maior que o modelo da SPM 2. A SPM 4, que também emprega memória PCM, apresenta maior memória quando comparada a SPM 3. O aumento de área referente ao emprego da SPM (nas configurações SPM5 e SPM4) pode ser observado claramente na Figura 31 (a), onde a SPM 5 e SPM 4 aumentaram suas porcentagens quando comparadas com a SPM 2 e SPM 3, respectivamente.



(a) Área das SPMs. (b) Número de endereços nas SPMs.

Figura 31 – Experimento 2 - Área e número de endereços das SPMs.

Na Figura 31 (b), a porcentagem dos endereços inseridos, para as diferentes SPMs empregadas no trabalho, são comparados com a linha base. Nesta figura percebe-se que, mesmo possuindo áreas menores, as SPMs (3 e 4), possuem respectivamente 50% e 75% a mais de endereços contidos nas memórias do que a SPM 1 (linha base). Este resultado está relacionado ao fato das NVMs possuírem densidades maiores do que os modelos tradicionais de memórias (SRAMs e DRAMs).

A Figura 32 ilustra o percentual do número total de acessos que uma determinada SPM possui em relação a linha base. Sabe-se que os endereços mais importantes, com uma maior frequência de acessos são inseridos em níveis mais altos da hierarquia. Assim a Figura 31 (b) e a Figura 32 possuem uma forte relação, pois, conforme as SPMs possuem um número maior de endereços contidos em suas memórias o número de acessos total a esses endereços tendem a ser maiores. Isso é verificado na Figura 32, onde nota-se que, em todos os casos (menor valor, maior valor e valor médio) e para todas as SPMs, o número total de acessos são superiores ao modelo da SPM 1.

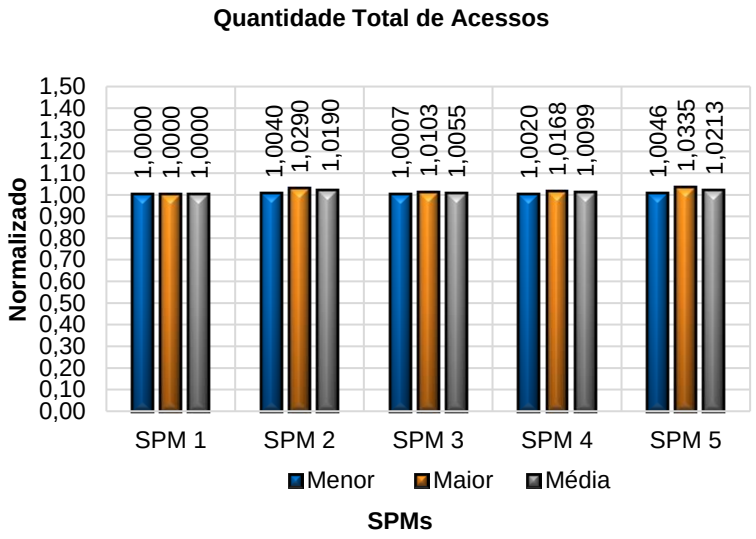


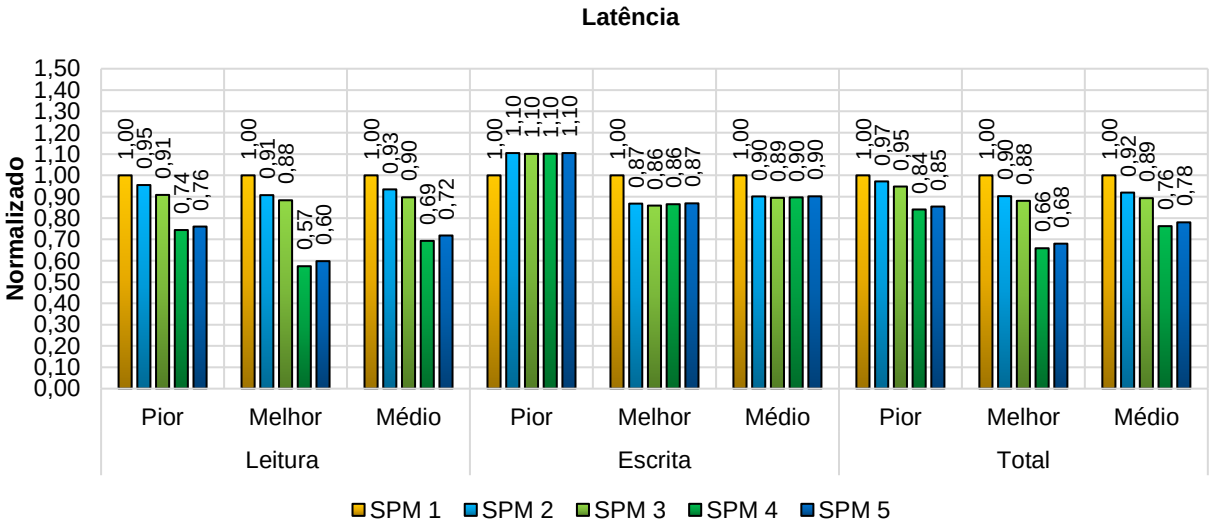
Figura 32 – Experimento 2 - Número total de acessos nas SPMs.

A Figura 33 exibe as comparações das SPMs com a SPM 1, para os parâmetros de latência (a) e energia (b). Para ambos as métricas são discutidos o pior caso, melhor caso e o caso médio. A latência na Figura 33 (a), foi dividida em três grupos, latência de leitura, latência de escrita e latência total, isso também foi adotado para o parâmetro de energia (Figura 33 (b)). Pela Figura 33 (a) percebe-se que, o único caso que não se obteve melhora na comparação com a SPM 1 foi o pior caso da latência de escrita, os demais casos para as diferentes latências obtiveram melhora no desempenho. Pela Figura 33 (a), na latência total, percebe-se que a inserção da STT-RAM (SPM 2 e SPM 3) já melhorou o desempenho em comparação com a SPM 1 e a otimização feita utilizando uma memória PCM melhorou ainda mais esse desempenho nas SPMs (4 e 5). Por exemplo, antes da utilização da PCM no caso médio para a latência total as SPMs (2 e 3) possuem respectivamente 8,08% e 10,69%, após o inseridas as PCM nas SPM esse ganho de desempenho aumentou para respectivamente para 22,03% e 23,81% (SPMs 5 e 4).

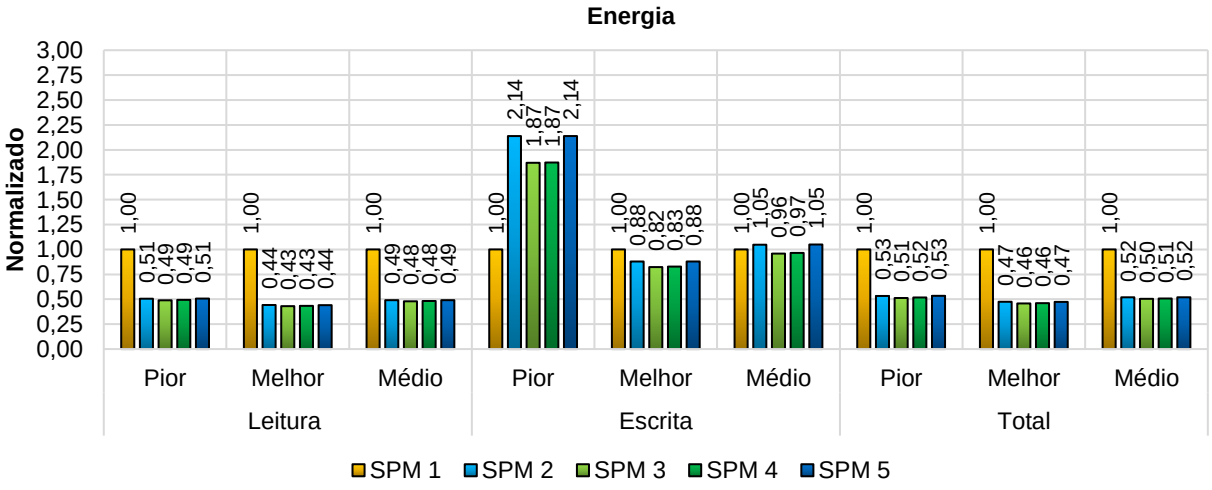
Na Figura 33 (b), verifica-se que os únicos casos que não se obteve ganho com a comparação com a linha base, foram observados na latência de escrita no pior caso e no caso médio. No pior caso todas as SPMs utilizadas apresentaram um maior consumo de energia e no caso médio as SPMs (2 e 5) possuem respectivamente 4,74% e 4,95% de aumento no consumo energético. Nos demais casos, observa-se ganhos na redução do consumo energético através da utilização das NVMs em comparação com a SPM 1. Mesmo perdendo em alguns casos da

energia de escrita, a energia total foi reduzida para todos os casos, ou seja, obteve-se ganhos em todos os casos (pior caso, melhor caso e caso médio) na energia total.

Por exemplo, para o caso médio da energia total, verifica-se que se obteve um ganho entre 48,03%(SPM 5) e 49,57%(SPM 3) de consumo energético quando comparada com a linha base. Ainda é observado que, a inserção da PCM nas SPMs (4 e 5) aumentou pouquíssimo o consumo energético, por exemplo, na SPM 3 o ganho de energia está com 49,57% após a inserção da PCM (SPM 4) o ganho diminuiu para 49,24%, um valor menor ainda é observado para as SPMs (2 e 5), onde respectivamente o ganho estava em 48,04% e após o uso da PCM passou para 48,03%.



(a) Latência (ns)



(b) Energia (pJ)

Figura 33 – Experimento 2 - Latência e energia para as SPMs.

A Figura 34 apresenta o produto da energia, do *delay* e da área (EDPA), para as SPMs utilizadas no trabalho. Verifica-se no caso médio que a SPM 4 possui o melhor ganho, cerca de 68%, enquanto que a SPM 3 possui um ganho de 63%. A inserção de uma PCM no modelo da SPM 4 quando comparada com a SPM 3, melhorou o resultado em 5%. Já a inserção de uma PCM na SPM 5 quando comparada com a SPM 2 melhorou o resultado em 6%.

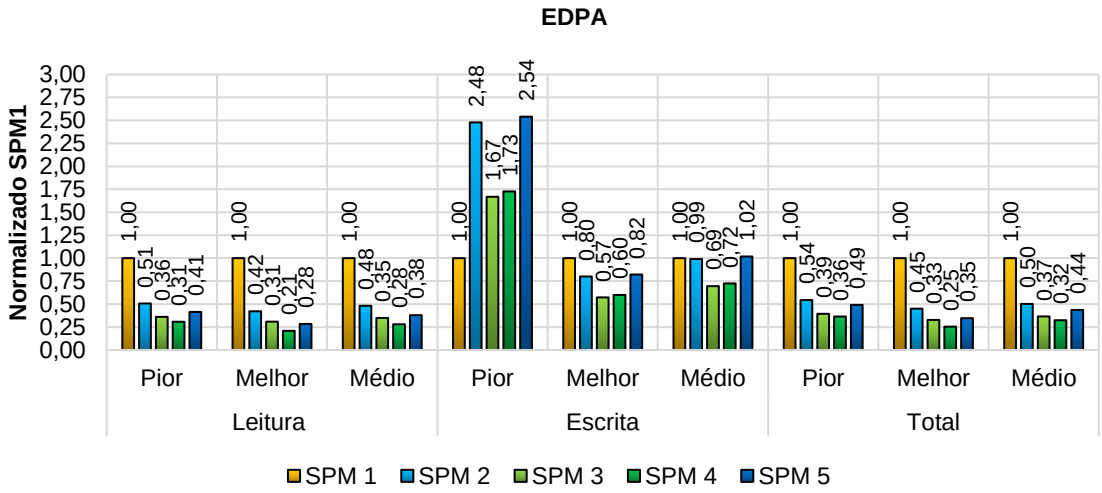


Figura 34 – Experimento 2 – Produto da energia, do *delay* e da área para as SPMs.

A Figura 35 apresenta a comparação do *leakage* das SPMs com a configuração linha base. Percebe-se que o melhor ganho foi obtido com SPM 3, que melhorou cerca de 47,48%, e após a inserção da PCM o ganho reduziu para 46,65% (SPM 4). Verifica-se pelas Figuras 31 (a) e 35, que mesmo possuindo áreas maiores as SPM (2 e 5) apresentam melhores resultados para o *leakage* do que a linha base. Pela Figura 35, observa-se reduções de *leakage* em todos as SPMs utilizadas na comparação com a SPM 1. Esses ganhos são ocasionados devido a utilização das NVMs, uma vez que, essas memórias caracterizam-se por possuírem um *leakage* menor quando comparadas com as memórias tradicionais.

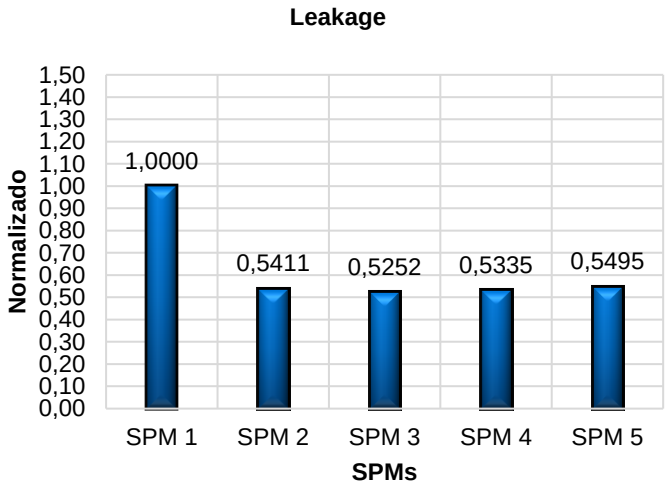


Figura 35 – Experimento 2 - Leakage para as SPMs.

5.2.2 Experimento 3 - Comparação entre as diferentes SPMs utilizando MP

Nesta seção será apresentada a comparação das diferentes SPMs avaliadas neste trabalho com a arquitetura da Figura 26 (d), utilizada como linha base nesse experimento. Esta arquitetura possui apenas uma memória principal (DRAM) e não utiliza SPM. Através dessa comparação é possível identificar a influência das SPMs no desempenho e no consumo energético total das aplicações. Todos os resultados desse experimento podem ser encontrados no apêndice D.

A Figura 36 apresenta os resultados de área para esse experimento, demostrando o aumento da área total da arquitetura da Figura 26 (d), com da inserção das diferentes SPMs utilizadas no trabalho. O menor aumento de área é encontrado para a SPM 3, cerca de 0,68% e o segundo menor aumento foi encontrado com a SPM 4. Entretanto, a SPM 4 possui um desempenho superior a SPM 3 (verificado no experimento 2) e possui uma diferença de apenas 0,02% na área total quando comparada com a SPM 3. Sendo assim, se a área não for um requisito crítico, é recomendável a utilização da SPM 4 para melhorar o desempenho do sistema.

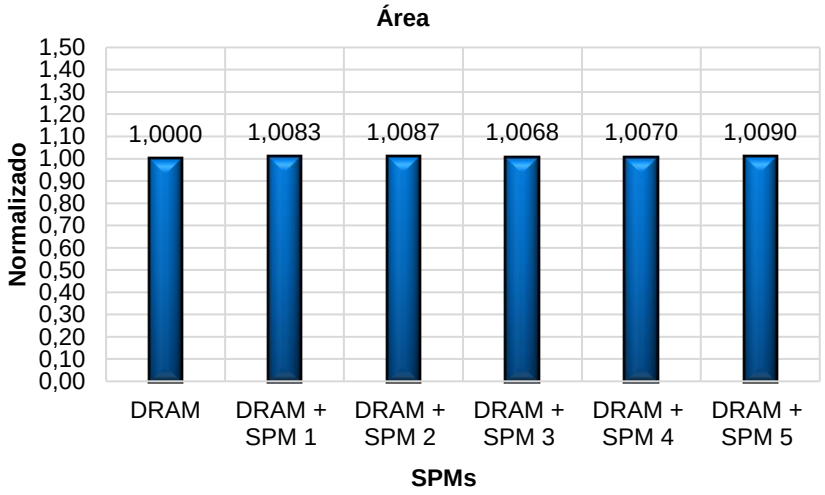


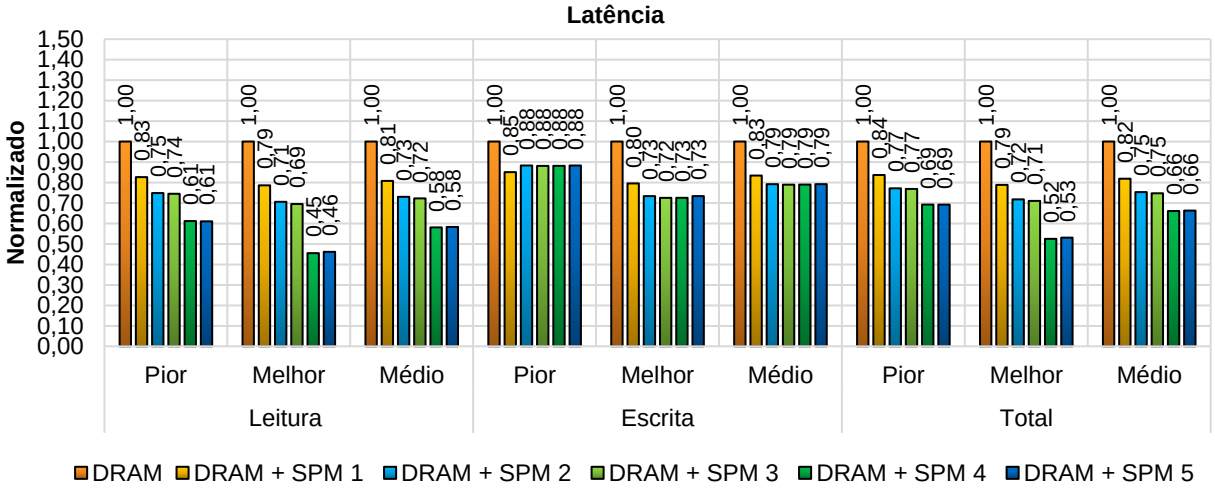
Figura 36 – Experimento 3 - Área total da arquitetura da figura 26 (d), após as inserções das SPMs.

A Figura 37 exibe as comparações das SPMs quando inseridas na arquitetura da Figura 26 (d) (linha base), para os parâmetros de latência (a) e energia (b). Neste gráfico, ambos os parâmetros são representados pelo pior caso, melhor caso e o caso médio. A latência na Figura 37 (a), foi dividida em três grupos, latência de leitura, latência de escrita e latência total, isso também foi realizado para o parâmetro de energia (Figura 37 (b)). Percebe-se que, para a Figura 37 (a) e (b), que em todos os casos (pior caso, melhor caso e caso médio) para todas as SPMs utilizadas obtém-se um significativo ganho, no desempenho e no consumo energético com a inserção das SPMs na arquitetura da Figura 26 (d). Por exemplo, o melhor ganho obtido no caso médio para a latência e o consumo energético são demonstradas pelas SPM 3 e SPM 5, essas SPMs possuem respectivamente 33,94% para latência e 80,14% para o consumo energético.

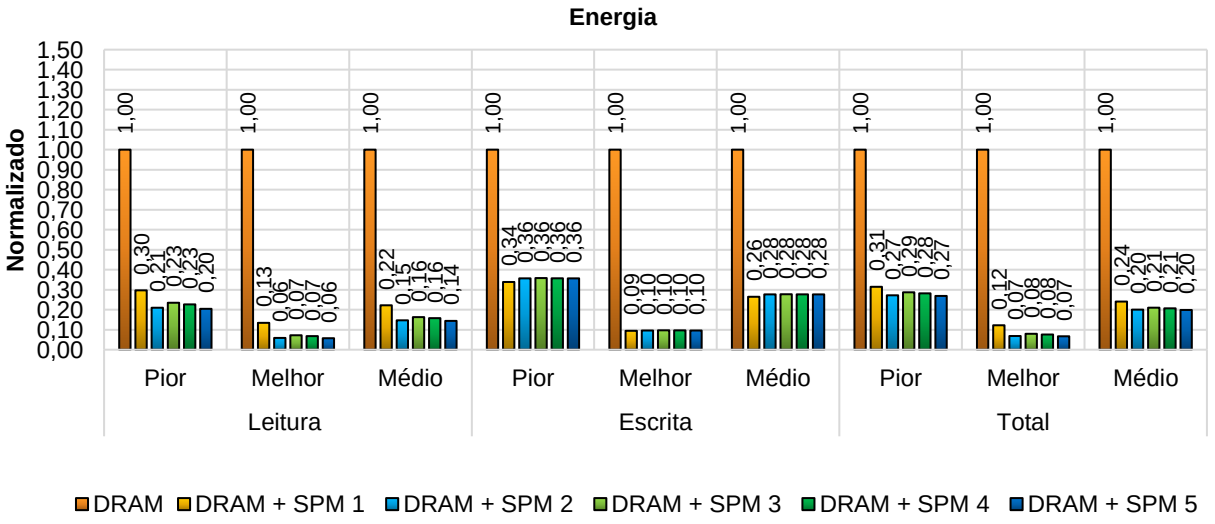
Nos dados ilustrados na Figura 37 (a), pode-se observar também o impacto positivo no desempenho referente a inclusão da PCM. A SPM 3 sem a utilização de PCM apresentou uma melhoria de desempenho de 25,27% comparada a linha base, enquanto que a SPM 4, com utilização da PCM apresentou uma melhoria de desempenho de 33,94%. Comparando essas duas SPMs (3 e 4) na Figura 37 (b) no caso médio, verifica-se que a inserção da PCM também reduziu o consumo energético total, onde a SPM 3 apresentou cerca de 78,99% de redução no consumo energético, enquanto que a SPM 4 apresentou 79,32%, quando comparadas com a linha base (DRAM). Essas melhorias de desempenho e consumo

de energia podem ser também verificadas entre as SPM 2 (sem a utilização de PCM) e a SPM 5 (com utilização da PCM).

Verifica-se pela Figura 37 (a), que o único caso que a SPM 1 possui resultados melhores as demais SPMs é no pior caso para a latência de escrita, em todos os demais casos a SPM 1 possui ganhos inferiores às outras SPMs utilizadas. Por outro lado, na Figura 37 (b), no consumo energético para a operação de escrita, em todos os casos (pior caso, melhor caso, caso médio) a SPM 1 possui resultados superiores as demais SPMs. Entretanto, para a o consumo energético para a operação de leitura e o consumo de energia total, a SPM 1 possui ganhos inferiores às outras SPMs utilizadas.



(a) Latência (ns)



(b) Energia (pJ)

Figura 37 – Experimento 3 - Latência e energia para a arquitetura da figura 26 (d), após as inserções das SPMs.

A Figura 38 apresenta o produto da energia, do *delay* e da área (EDPA). Verifica-se no caso médio que a SPM 5 possui o melhor ganho, cerca de 87%, enquanto que a SPM 2 possui um ganho de 85%. A inserção de uma PCM no modelo da SPM 5 quando comparada com a SPM 2, melhorou o resultado em 2%.

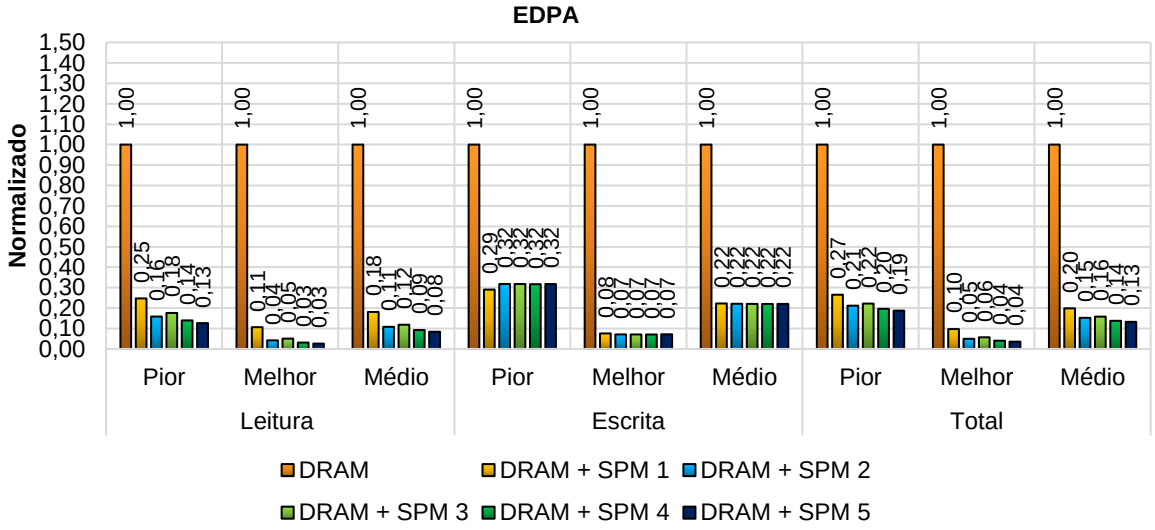


Figura 38 – Experimento 2 - Produto da energia, do *delay* e da área para a arquitetura da figura 26 (d), após as inserções das SPMs.

A Figura 39 apresenta a comparação do *leakage* da arquitetura base com suas as demais variações da arquitetura (após as inserções das SPMs). Como são inseridos na arquitetura base, os modelos de SPMs que possuem outros tipos de memória em seu interior, o *leakage* resultante na arquitetura base possui resultados maiores. Percebe-se pela Figura 39 que a inserção da SPM 1 possui o pior resultado, cerca de 121,03% de aumento do *leakage*, enquanto que as SPMs com modelos híbridos de memórias possuem valores entre 63,56% e 66,50% de aumento do *leakage*.

Percebe-se pela Figura 36, que as arquiteturas compostas das SPMs 2 e 5 possuem uma área total maior, entretanto essas arquiteturas possuem um *leakage* menor quando comparadas com a arquitetura que possui a SPM 1 inserida (Figura 39). Este resultado aponta que as NVMs, presentes nas variações SPM2 a SPM5, possuem vantagens no *leakage* quando comparadas com as memórias tradicionais.

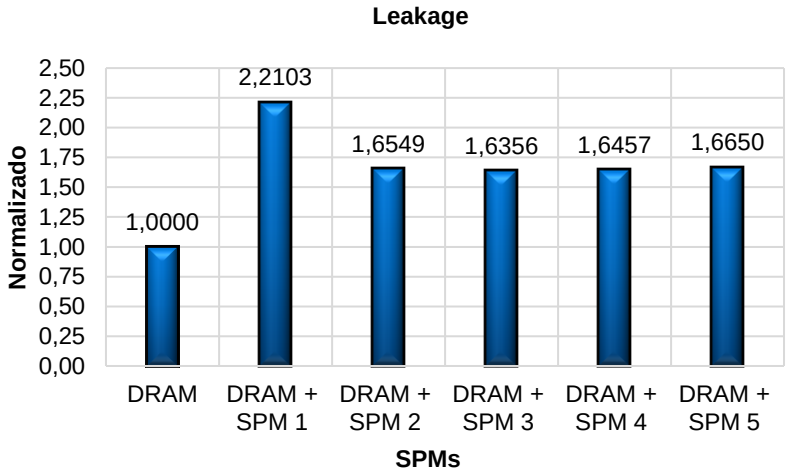


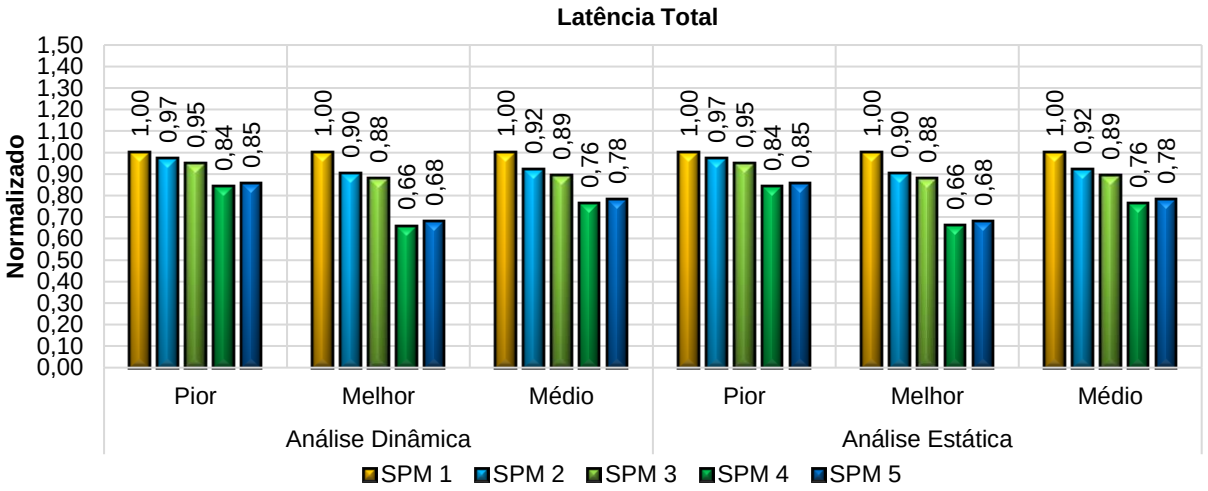
Figura 39 – Experimento 3 - Leakage da arquitetura da figura 26 (d) após as inserções das SPMs.

5.3 Comparação entre os resultados da Análise Dinâmica e Estática

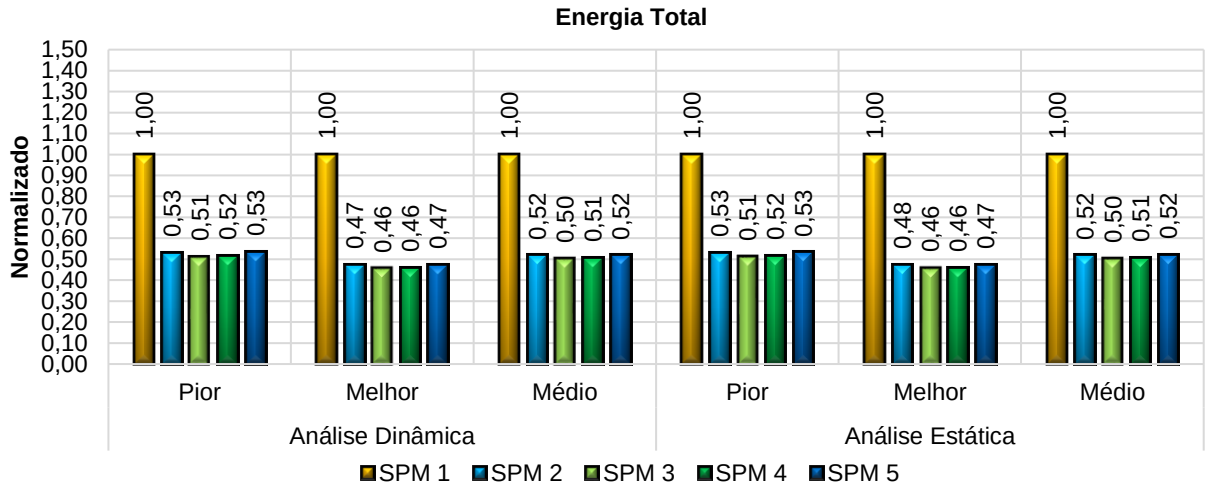
Nesta seção, os resultados dos experimentos 2 e 3, obtidos empregando a abordagem dinâmica, são comparados aos resultados obtidos aplicando a abordagem estática nos mesmos experimentos. Ambas as análises possuem melhores resultados para o consumo energético e desempenho, quando comparadas com as linhas bases do segundo e terceiro experimento.

A Figura 40 demonstra a comparação entre as análises para o experimento 2, enquanto que a Figura 41 exibe a comparação entre as abordagens para o experimento 3. Ambas as figuras apresentam os resultados dos parâmetros de latência e energia totais dos experimentos realizados, na forma de pior caso, melhor caso e caso médio.

Percebe-se pela Figura 40 (a) e (b), que para todos os casos, os resultados encontrados para latência nas diferentes soluções (SPM1 a SPM5) possuem valores bem próximos entre as duas abordagens. Por exemplo, comparando o caso médio de latência encontrado para a análise dinâmica na SPM 4 com a linha base (SPM1), observa-se um ganho de 23,81%. Enquanto, que o ganho da SPM4 comparado a SPM1, considerando caso médio de latência, para a análise estática é de 23,79%. Neste exemplo, a análise dinâmica apresentou o melhor resultado para latência.

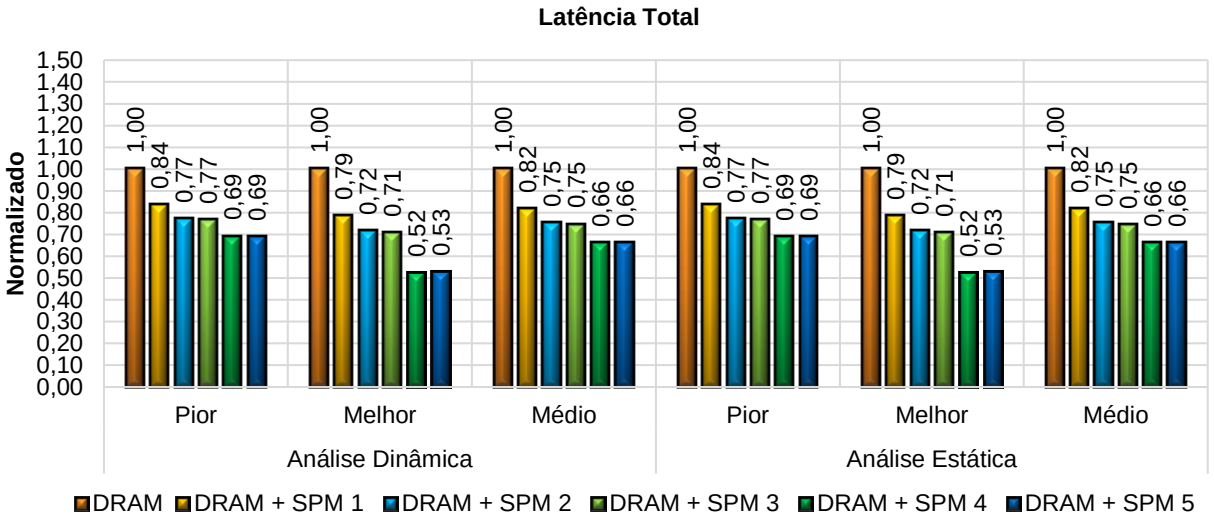


(a) Comparação das análises estática e dinâmica, para a latência do experimento 2.

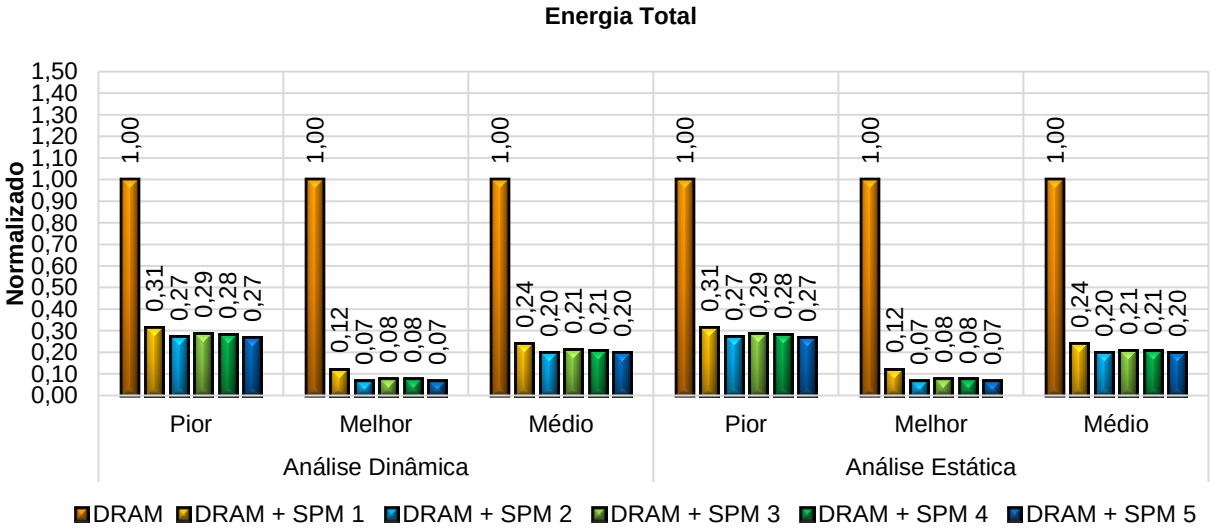


(b) Comparação das análises estática e dinâmica, para a energia do experimento 2.
Figura 40 – Comparação das análises estática e dinâmica, para o experimento 2.

A mesma situação ocorre na comparação das latências encontradas pelo emprego das abordagens estática e dinâmica no terceiro experimento. Nota-se pela Figura 41 (a) e (b), que os resultados encontrados entre as análises também possuem valores próximos. Na Figura 41 (a), a latência do caso médio obtida pela SPM 4 foi reduzida em cerca de 33,94%, com o emprego da abordagem dinâmica, enquanto na abordagem estática a mesma SPM apresentou uma melhoria de 33,95%, ambas comparadas a SPM1. Neste exemplo, onde as soluções não consideram apenas a SPM, a análise estática possui um melhor resultado em latência.



(a) Comparação das análises estática e dinâmica, para a latência do experimento 3.



(b) Comparação das análises estática e dinâmica, para a energia do experimento 2

Figura 41 – Comparação das análises estática e dinâmica, para o experimento 3.

6 CONCLUSÕES E TRABALHOS FUTUROS

A diferença de um sistema computacional de uso geral, para um sistema embarcado, é que um sistema embarcado possui diversas restrições e requisitos específicos, geralmente são utilizados em uma tarefa predefinida, ao contrário dos sistemas computacionais de propósito geral, que executam uma vasta gama de aplicações. Em um sistema computacional de uso geral o gargalo dominante para a obtenção de alto desempenho e eficiência energética é a distância da tecnológica entre o desempenho do processador e o da memória tradicional, isto se torna ainda mais significativo em sistemas embarcados, pois muitos sistemas embarcados possuem um tamanho reduzido e utilizam como fonte de energia uma bateria que alimenta o sistema para realizar a sua funcionalidade específica. Sendo assim a arquitetura de memória possui uma forte influência em sistemas embarcados, pois as operações realizadas na memória, possuem um elevado consumo energético para todo o sistema, além de influenciarem o desempenho do sistema. Assim novas pesquisas e tecnologias estão sendo desenvolvidas para melhorarem essas operações, afim de aumentar o desempenho e manter pelo menos o mesmo consumo energético que anteriormente, ou vice-versa.

O presente trabalho revisou sobre técnicas de otimizações em memórias para sistemas embarcados, revisando desde hierarquias de memórias, *caches*, *scratchpads*, até os problemas enfrentados pelos sistemas embarcados. Além disso, foram revisadas tecnologias emergentes de memória, que possuem vantagens interessantes para o domínio de sistemas embarcados.

Este trabalho realizou explorações de arquiteturas híbridas de memórias para sistemas embarcados. Através dessa exploração, investiga-se o consumo energético e o desempenho nos acessos à memória para arquitetura ARM, explorando tecnologias de memória volátil e não volátil em uma SPM. As memórias utilizadas neste trabalho são, a DRAM, SRAM, STT-RAM e PCM. Para a obtenção dos resultados utiliza-se a metodologia descrita na Seção 4.2. Através dos *benchmarks* do *MiBench* e do Simics são extraídos os *traces* de dados das aplicações utilizadas. Utiliza-se as ferramentas NVsim e Cacti para a obtenção dos parâmetros das

memórias. Pode-se utilizar duas análises diferentes para a alocação dos endereços nas memórias, a análise estática e dinâmica, cada qual com seus respectivos *scripts* responsáveis pelas alocações dos endereços. Para investigação do consumo e desempenho das arquiteturas analisadas, realiza-se experimentos utilizando cinco arquiteturas distintas, cada qual possuindo configurações diferentes de memória voláteis e não voláteis (Tabela 3).

Com os resultados encontrados da pesquisa, conclui-se, que os modelos híbridos obtiveram bons resultados tanto para desempenho quanto para consumo. Vale destacar os resultados obtidos para o modelo de SPM com 16KB de SRAM, 32KB de STT-RAM e 8KB de PCM (SPM 4) do experimento 2, que apresentou um ganho médio de 23,81% na latência total e de 49,24% no consumo energético, quando comparada com o modelo da SPM 1 (SPM possuindo 32KB de SRAM). Além disso, a SPM 4 ainda reduziu a área e o *leakage*, respectivamente em 16,29% e 46,65%, quando comparada com a SPM 1.

Este trabalho confirmou vantagens conhecidas tanto para as VMs quanto para as NVMs (Tabela 4). As NVMs possuem vantagens nas latências e no consumo energético para as operações de leitura, enquanto que a VM possui vantagens para as operações de escrita, que puderam ser observados também nos experimentos realizados neste trabalho.

Conclui-se que a memória é um dos principais fatores que influenciam o desempenho e o gasto energético em sistemas embarcados, graças a isso a mesma possui um grande potencial para as futuras pesquisas relacionadas a esses casos, como demonstrado nesse trabalho. Entretanto percebe-se, como possibilidade de trabalhos futuros, a ampliação do estudo sobre as técnicas de memória existentes, a partir da utilização de outras ferramentas de modelagem e simulação, como o simulador GEM5 (BINKERT, et al. 2011) ou o simulador Tejas (SARANGI, et al. 2015). Para a obtenção dos parâmetros das NVMs é utilizado neste trabalho o NVsim (DONG, et al. 2012), entretanto, se for utilizada a ferramenta GEM5 nas futuras pesquisas, pode-se utilizar o simulador NVMain (POREMBA, et al. 2015), para adquirir os parâmetros das NVMs.

Além disso, utilizando o GEM5, pode-se estender o trabalho utilizando o SMCSim (ERFAN, et al. 2016), este simulador também é uma extensão da ferramenta GEM5. O SMCSim modela um dispositivo *Smart Memory Cube* (SMC)

conectado a um sistema *host-on-chip* completo. O SMC é uma extensão para o padrão Híbrido Memória Cubo (HMC). A principal contribuição do trabalho de Erfan et al. (2016) é a análise completa do sistema de memória, incluindo software de alto nível para baixo nível (*firmware*) e camadas de hardware, considerando a descarga e as despesas gerais dinâmicas causadas pelo sistema operacional, coerência de cache e gerenciamento de memória.

Este trabalho utilizou como estudo de caso apenas SPMs, foram empregadas cinco configurações diferentes de SPMs possuindo memórias voláteis e não voláteis. Sendo assim, outra possibilidade para extensão deste trabalho seria a realização de otimizações em hardware ou em software para memórias *caches* utilizando memórias emergentes. Para isso, pode-se citar o trabalho de Mittal, Vetter e Li (2015) e o trabalho de Mittal e Vetter (2016). Nesses trabalhos os autores apresentam diversos trabalhos relacionados com otimizações em *caches* e memórias principais, essas otimizações são realizadas tanto em hardware quanto em software, através de projetos arquitetônicos ou técnicas voltadas ao software. Assim, como trabalho futuro, pode-se buscar a utilização de modelos híbridos de memória em outros níveis da hierarquia de memória, como por exemplo, em memórias *caches* e memórias principais. Com o intuito de melhorar o desempenho e o consumo energético dos acessos à memória dos sistemas embarcados.

REFERÊNCIAS

ACEVEDO, O; JIMENEZ, M. **A Survey of Software Optimization Techniques for Low-Power Consumption**. University of Puerto Rico. 2002.

AGRAWAL, A. et al. Refrint: Intelligent refresh to minimize power in on-chip multiprocessor cache hierarchies. **Proceedings of the 2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA).** p. 400 – 411, 2013.

ALIPOUR, M.; SALEHI, M. E.; MOSHARI, K. Cache power and performance tradeoffs for embedded applications. **International Conference on Computer Applications and Industrial Electronics (ICCAIE)**. 2011.

ALIZADEH, M. et al. Versatile refresh: Low complexity refresh scheduling for high-throughput multi-banked eDRAM. **Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE joint international conference on Measurement and Modeling of Computer Systems**. Volume. 40, no. 1, p. 247–258, 2012.

ARM**. ARM company overview** - **Company highlights.** 2017. Disponível em: < https://www.arm.com/company/> Acesso em: 16 janeiro 2017.

BANAKAR, R. et al. Scratchpad Memory: a design alternative for cache on-chip memory in embedded systems**. Proceedings of the Tenth International Symposium on Hardware/Software Codesign (CODES)**, p. 73-78, 2002.

BARTH J., et al. A 500 MHz random cycle, 1.5 ns latency, SOI embedded DRAM macro featuring a three-transistor micro sense amplifier. **IEEE Journal of Solid-State Circuits**. Volume: 43, Issue: 1, p. 86 – 95, 2008.

BEDDARD, R. **Share Sleuth: tech titan ARM on sale**. 23, february, 2016. Disponível em: <http://www.moneyobserver.com/our-analysis/share-sleuth-tech-titan-arm-sale> Acesso em: 16 janeiro 2017.

BINKERT, N. et al. The gem5 simulator. **ACM SIGARCH Computer Architecture News**, Volume 39 Issue 2, Pages 1-7, may 2011.

BISHNOI, R. et al. Avoiding unnecessary write operations in STT-MRAM for low power implementation. **15th International Symposium on Quality Electronic Design (ISQED).** p. 548 – 553, 2014a.

BISHNOI, R. et al. Asynchronous asymmetrical write termination (AAWT) for a low power STT-MRAM. **Design, Automation and Test in Europe Conference and Exhibition (DATE).** p. 1 – 6, 2014b.

CACTI 5.3. 2014. Disponivel em: <http://quid.hpl.hp.com:9081/cacti/>. Acesso em: 17 janeiro 2017.

CATTHOOR, Francky; RAGHAVAN, Praveen; JAYAPALA Murali; KRITIKAKOU, Angeliki; LAMBRECHTS, Andy; ABSAR, Javed. **Ultra-low energy domain-specif instruction-set processors**. Springer, 2010. 377 p.

CHANG, M.-T. et al. Technology comparison for large last-level caches (L3Cs): Low-leakage SRAM, low write-energy STT-RAM, and refresh-optimized eDRAM. **IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)**. p. 143 - 154, 2013.

DASH, S; SRIKANTHAN, T. Instruction Cache Tuning for Embedded Multitasking Applications. **IET Computers & Digital Techniques**, Volume: 4, Issue: 6, p. 439 – 457, november, 2010.

DÉVELOPPEMENT, Y. **Emerging Non-volatile Memor**y. July 28, 2016. Disponível em: <http://www.yole.fr/Emerging_NVM_Market.aspx#.WH6o6hsrKHu> Acesso em: 10 janeiro 2017.

DONG, X. et al. NVSim: A Circuit-Level Performance, Energy, and Area Model for Emerging Nonvolatile Memory. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, Volume: 31, Issue: 7, p. 994-1007, july, 2012.

ENGBLOM, J.; EKBLOM, D. Simics: a Commercially Proven Full-Fystem Simulation Framework. **European Space Programmers**, 2006.

ERFAN, A. et al. Design and Evaluation of a Processing-in- Memory Architecture for the Smart Memory Cube. **International Conference on Architecture of Computing Systems**. Springer International Publishing, 2016.

FUJITA, S. et al. Technology trends and near-future applications of embedded STT-MRAM. **IEEE International Memory Workshop (IMW)**. 2015.

GAJSKI, Daniel D.; ABDI, Samar; GERSTLAUER, Andreas; SCHIRNER, Gunar. **Embedded System Design: Modeling, Synthesis and Verification**. Springer, 2009. 352 p.

GUTHAUS, M.R. et al. MiBench: A free, commercially representative embedded benchmark suite. **IEEE International Workshop on Workload Characterization** (WWC-4), p. 3-14, 2001.

HENNESSY, John L.; PATTERSON, David A. **Computer Architecture – A quantitative approach.** Morgan Kaufmann, 2011. 856 p.

HEXUS**. ARM Everywhere**. 15, march, 2015. Disponível em: <http://hexus.net/static/arm-everywhere/> Acesso em: 16 janeiro 2017.

HU, J et al. Write activity reduction on non-volatile main memories for embedded chip multi-processors, **Journal ACM Transactions on Embedded Computing Systems (TECS)**, Volume 12 Issue 3, 2013.

HU, J. et al. Data Allocation Optimization for Hybrid Scratch Pad Memory with SRAM and Nonvolatile Memory. **IEEE Transactions on Very Large Scale Integration Systems** (VLSI), Volume: 21, Issue: 6, p. 1094 – 1102, june, 2013.

HUAI, Y. Spin-transfer torque MRAM (STT-MRAM): Challenges and prospects. **AAPPS Bulletin**, Key: citeulike:11401636, Vol. 18, No. 6. p. 33–40, 2008.

JOG, A. et al. Cache revive: Architecting volatile STT-RAM caches for enhanced performance in CMPs. **49th ACM/EDAC/IEEE Design Automation Conference (DAC),** p. 243 – 252, 2012.

JOO, Y. et al. Energy and endurance-aware design of phase change memory caches. **Proceedings of the Conference on Design, Automation and Test in Europe (DATE)**. p. 136 – 141, 2010.

KHALILI, A. P. et al. Switching current reduction using perpendicular anisotropy in COFEB-MgO magnetic tunnel junctions. **Applied Physics Letters**, Volume 98, Issue: 11, p. 112-507. 2011.

KIM, Y.-B. et al. Bi-layered RRAM with unlimited endurance and extremely uniform switching. **Symposium on VLSI Technology (VLSIT)**. p. 52 – 53. 2011.

KOMALAN, M. P. et al. System level exploration of a STT-MRAM based Level 1 Data-Cache. **Design, Automation & Test in Europe Conference & Exhibition (DATE)**. p. 1311-1316, march, 2015.

KOOPMAN, P. Reliability, safety, and security in everyday embedded systems. **Proceedings of the Third Latin-American conference on Dependable Computing (LADC)**, p. 1 – 2, september, 2007.

KWAK, J. W.; CHOI, J. H. Selective Access to Filter Cache for Low-Power Embedded Systems. **Proceedings of the 43rd Hawaii International Conference on System Sciences (HICSS)**. p. 1 - 8, january, 2007.

LEE, B. et al. Architecting Phase Change Memory as a Scalable DRAM Alternative. **Proceedings of the 36th annual international symposium on Computer architecture** (ISCA), p. 2 - 13. 2009.

LI, H; CHEN, Y. An overview of non-volatile memory technology and the implication for tools and architectures. **Design, Automation & Test in Europe Conference & Exhibition (DATE)**. p. 731 – 736, 2009.

LI, L.; GAO, L.; XUE, J. Memory Coloring: A Compiler Approach for Scratchpad Memory Management. **Proceedings of the 14th International Conference on Parallel Architectures and Compilation Techniques (PACT)**. p. 329 – 338, setembro, 2005.

LI, Q. et al. MGC: Multiple graph-coloring for non-volatile memory based hybrid scratchpad memory. **16th Workshop on Interaction between Compilers and Computer Architectures (INTERACT)**. p. 1 – 6, 2012.

MAGNUSSON, P. et al. Simics: A Full System Simulation Platform. **IEEE Computer**, Volume: 35, Issue: 2, p. 50-58, 2002.

MARWEDEL, Peter. 2014. **Optimizations - Compilation for Embedded Processors**. Disponível em: <http://ls12-www.cs.tu-dortmund.de/daes/media/documents/teaching/courses/ws1213/es/lecture/es-marw-7.2-optimizations.pdf> Acesso em: junho 2016.

MARWEDEL, Peter. **Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems**. 2nd ed. Springer, 2011. 333 p.

MATTOS, Júlio Carlos Balzano de. **Design Space Exploration of SW and HW IP based on Object Oriented Methodology for Embedded System Applications**. 2007. 91 f. Tese (Doutorado em Computação) – Instituto de Informática, Programa de Pós-Graduação em Computação, Universidade Federal do Rio Grande do Sul (UFRGS), Porto Alegre, 2007. Disponível em: < http://www.lume.ufrgs.br/bitstream/handle/10183/13484/000648263.pdf> Acesso em: 21 setembro 2015.

MITTAL, S.; VETTER, J. S. A Survey of Software Techniques for Using Non-Volatile Memories for Storage and Main Memory Systems. **IEEE Transactions on Parallel and Distributed Systems**, Volume: 27, Issue: 5, may 1, 2016.

MITTAL, S.; VETTER, J. S.; LI, D. A Survey of Architectural Approaches for Managing Embedded DRAM and Non-Volatile On-Chip Caches. **IEEE Transactions on Parallel and Distributed Systems**, Volume: 26, Issue: 6, june 1, 2015.

MORGAN, T. P. **ARM Holdings eager for PC and server expansion Record 2010, looking for Intel killer 2020**. 1, february, 2011. Disponível em: <http://www.theregister.co.uk/2011/02/01/arm_holdings_q4_2010_numbers/> Acesso em: 16 janeiro 2017.

PANDA, P. R. et al., 2001. Data and memory optimization techniques for embedded systems**. Journal ACM Transactions on Design Automation of Electronic Systems (TODAES)**, Volume: 6, Issue: 2, p. 149 - 206, april, 2001.

POREMBA, M.; ZHANG, T.; XIE, Y. NVMain 2.0: Architectural Simulator to Model (Non-)Volatile Memory Systems, **Computer Architecture Letters** (CAL), 2015.

POUCHET, Louis-Noel; YUKI, Tomofumi. **PolyBench**. 2011, Universidade Estadual de Ohio. Disponível em: < https://sourceforge.net/projects/polybench/> Acesso em: 19 janeiro 2017.

QADRI, M.; MCDONALDO-MAIER, K. Data Cache- Energy and Throughput Models: Design Exploration for Embedded Processors**. EURASIP Journal on Embedded Systems**, p. 20 – 27, december, 2009.

RODRÍGUEZ, G.; TOURIÑO, J.; KANDEMIR, M. T. Volatile STT-RAM Scratchpad Design and Data Allocation for Low Energy. **Journal ACM Transactions on Architecture and Code Optimization (TACO).** Volume 11, issue 4, Article 38. 2014.

SARANGI, S. R. et al. Tejas: A Java based Versatile Micro-architectural Simulator; **International Workshop on Power and Timing Modeling, Optimization and Simulation** (PATMOS) 2015.

SHEARER, Findlay. ***Power Management in Mobile Devices****.* 1 ed . Newnes, 2008. 336 p.

SHIMPI, A. L. **ARM Partners Ship 50 Billion Chips Since 1991 - Where Did They Go?**. 31, march, 2014. Disponível em: <http://www.anandtech.com/show/7909/arm-partners-ship-50-billion-chips-since-1991-where-did-they-go> Acesso em: 16 janeiro 2017.

SILVA, A. et al. An ESL approach for energy consumption analysis of cache memories in SoC platforms. **International Journal of Reconfigurable Computing,** p. 1 – 12, january, 2011.

SMULLEN, C. W. et al. Relaxing non-volatility for fast and energy-efficient STTRAM caches. **Proceedings in IEEE 17th International Symposium on High Performance Computer Architecture (HPCA)**. p. 50 – 61, 2011.

STALLINGS, William. **Computer Organization and Architecture: designing for performance***.* 10 ed. Prentice Hall PTR, 2015. 864 p.

SUN, G. et al. A Novel Architecture of the 3D Stacked MRAM L2 Cache for CMPs. **IEEE 15th International Symposium on High Performance Computer Architecture** (HPCA), p. 239 – 249, 2009.

TADISINA, Z. et al. Perpendicular magnetic tunnel junctions using Co-based multilayers. **Journal of Applied Physics**, Volume: 107, Issue: 9, 2010.

THOZIYOOR, S. et al. A Comprehensive Memory Modeling Tool and its Application to the Design and Analysis of Future Memory Hierarchies. **Proceedings of the 35th Annual International Symposium on Computer Architecture**. p. 51 - 62, 2008.

VENKATESAN, R. et al. TapeCache: a high density, energy efficient cache based on domain wall memory. **Proceedings in ACM/IEEE international symposium on Low power electronics and design.** p.185 – 190, 2012.

VERMA, Manish; MARWEDEL, Peter. **Advanced memory optimization techniques for low-power embedded processors** Springer, 2007. 188 p.

VIRTUTECH. **Simics 3.0 User Guide for Unix**. 2007. Disponível em: <http://www.cs.sfu.ca/~fedorova/Tech/simics-guides-3.0.26/simics-user-guide-unix/> Acesso em: Julho 2015.

WANG, G. et al. Energy-aware assignment and scheduling for hybrid main memory in embedded systems. **Computing (2016)**. Volume 98, issue 3, p. 279 - 301. 2015

WANG, J. et al. i2WAP: Improving non-volatile cache lifetime by reducing inter-and intra-set write variations. **IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)**. p. 234 – 245, 2013.

WANG, P. et al. Designing Scratchpad Memory Architecture with Emerging STT-RAM Memory Technologies. **Proceedings of IEEE International Symposium on Circuits and Systems** (ISCAS), Beijing, China, may, p. 1244 – 1247, 2013.

WANG, Y. et al. Loop scheduling optimization for chip-multiprocessors with non-volatile main memory. **International Conference on Acoustics, Speech and Signal Processing (ICASSP)**. p. 1553 – 1556, 2012.

WILKERSON, C., et al. Reducing cache power with low-cost, multi-bit error-correcting codes. **Proceedings of the 37th annual international symposium on Computer Architecture (ISCA)**, p. 83 – 93. 2010.

WOLF, Marilyn. **Computers as Components: Principles of Embedded Computing System Design**. 3rd ed. Morgan Kaufmann, 2012. 528 p.

WOLF, W.; KANDEMIR, M. Memory System Optimization of Embedded Software. **Proceedings of the IEEE**, Volume: 91, Issue: 1, p. 165 - 182, 2003.

WU, X. et al. Hybrid Cache Memory Architecture with Disparate Technologies. **Proceedings of the 36th annual international symposium on Computer architecture** (ISCA). p. 34-45, 2009.

XU, C. et al. Device-architecture co-optimization of STT-RAM based memory for low power embedded systems. **Proceedings of the International Conference on Computer-Aided Design (ICCAD)**, p. 463 - 470. 2011.

APÊNDICES

Apêndice A – Porcentagem dos acessos mais frequentes dos endereços para diferentes tamanhos de memórias.

Tabela 11 – Memórias de 4 bytes até 2 Kbytes para os *traces* entre os números 1 e 5

	Trace	T. Mem/Bytes	4	8	16	32	64	128	256	512	1K	2K
1	basicmath	T. Acessos	1,82%	3,60%	6,90%	11,90%	17,52%	25,37%	33,64%	42,82%	53,65%	63,48%
		T. Leitura	0,91%	1,80%	3,46%	6,45%	9,98%	14,73%	20,50%	25,80%	32,08%	38,37%
		T. Escrita	0,91%	1,80%	3,45%	5,85%	8,42%	12,02%	15,03%	18,96%	23,09%	27,41%
		M. Cons. Leitura	0,70%	1,41%	2,26%	3,50%	5,22%	6,59%	8,06%	9,85%	12,27%	14,61%
		M. Cons. Escrita	0,30%	0,33%	0,37%	0,46%	0,63%	0,90%	1,12%	1,28%	1,45%	1,64%
		M. Cons. L/E	0,70%	1,41%	2,26%	3,50%	5,38%	6,81%	8,32%	10,13%	12,69%	15,29%
2	bitcount	T. Acessos	0,91%	1,27%	1,91%	2,89%	4,67%	7,61%	11,71%	18,18%	27,00%	37,34%
		T. Leitura	0,91%	1,23%	1,73%	2,62%	4,36%	6,36%	8,66%	12,17%	17,27%	23,30%
		T. Escrita	0,18%	0,29%	0,50%	0,89%	1,55%	2,66%	4,57%	7,56%	11,47%	15,97%
		M. Cons. Leitura	0,91%	1,23%	1,73%	2,62%	4,31%	5,72%	7,24%	8,83%	10,59%	12,67%
		M. Cons. Escrita	0,07%	0,13%	0,17%	0,22%	0,30%	0,40%	0,57%	0,87%	1,23%	1,80%
		M. Cons. L/E	0,91%	1,23%	1,73%	2,62%	4,31%	5,72%	7,30%	8,93%	10,73%	12,88%
3	crc32	T. Acessos	0,97%	1,36%	2,03%	3,03%	4,69%	7,24%	11,21%	17,31%	25,62%	35,89%
		T. Leitura	0,97%	1,36%	1,94%	2,87%	4,42%	6,26%	8,43%	11,78%	16,54%	22,47%
		T. Escrita	0,15%	0,25%	0,44%	0,80%	1,39%	2,46%	4,22%	6,99%	10,71%	15,27%
		M. Cons. Leitura	0,97%	1,36%	1,88%	2,77%	4,27%	5,65%	7,10%	8,66%	10,33%	12,33%
		M. Cons. Escrita	0,07%	0,14%	0,18%	0,23%	0,30%	0,40%	0,54%	0,79%	1,10%	1,57%
		M. Cons. L/E	0,97%	1,36%	1,88%	2,77%	4,27%	5,66%	7,18%	8,77%	10,48%	12,53%
4	dijkstra	T. Acessos	11,55%	23,09%	36,03%	51,44%	61,50%	65,81%	70,74%	75,59%	80,69%	85,36%
		T. Leitura	11,55%	21,16%	31,70%	45,20%	54,49%	57,46%	60,36%	63,49%	68,14%	72,49%
		T. Escrita	2,40%	4,33%	6,58%	7,28%	8,12%	9,59%	11,40%	13,12%	13,77%	14,45%
		M. Cons. Leitura	11,55%	17,31%	24,99%	31,07%	31,98%	32,77%	33,95%	34,76%	35,41%	36,12%
		M. Cons. Escrita	0,05%	0,10%	0,20%	0,40%	0,76%	0,81%	0,83%	0,86%	0,91%	0,97%
		M. Cons. L/E	11,55%	17,31%	24,99%	31,07%	31,98%	32,77%	34,32%	35,40%	36,13%	36,86%
5	fft	T. Acessos	3,22%	6,44%	7,44%	8,25%	9,66%	12,00%	15,34%	20,58%	27,69%	39,85%
		T. Leitura	3,22%	6,31%	7,27%	8,01%	9,36%	10,98%	12,86%	15,68%	19,75%	25,88%
		T. Escrita	0,14%	0,27%	0,44%	0,76%	1,29%	2,18%	3,72%	6,14%	9,44%	15,39%
		M. Cons. Leitura	3,22%	3,95%	4,37%	5,09%	6,42%	7,59%	8,78%	10,07%	11,51%	13,21%
		M. Cons. Escrita	0,05%	0,11%	0,14%	0,19%	0,25%	0,33%	0,46%	0,69%	0,96%	1,41%
		M. Cons. L/E	3,22%	3,95%	4,37%	5,09%	6,42%	7,59%	8,83%	10,15%	11,63%	13,38%

Tabela 12 – Memórias de 4 Kbytes até 2 Mbytes para os *traces* entre os números 1 e 5

	Trace	T. Mem/Bytes	4K	8K	16K	32K	64K	128K	256K	512K	1M	2M
1	basicmath	T. Acessos	73,93%	82,60%	87,25%	90,05%	91,90%	93,57%	95,44%	98,05%	100,00%	100,00%
		T. Leitura	45,07%	50,39%	53,37%	55,03%	56,17%	57,23%	58,35%	59,45%	59,91%	59,91%
		T. Escrita	31,22%	33,58%	34,86%	35,61%	36,33%	37,26%	38,39%	39,68%	40,07%	40,07%
		M. Cons. Leitura	16,10%	16,97%	17,58%	18,14%	18,79%	19,62%	20,46%	21,55%	22,01%	22,01%
		M. Cons. Escrita	1,91%	2,05%	2,25%	2,55%	3,01%	3,64%	4,44%	5,54%	5,93%	5,93%
		M. Cons. L/E	17,16%	18,33%	19,24%	19,96%	20,75%	21,78%	23,21%	24,75%	26,51%	26,51%
2	bitcount	T. Acessos	49,15%	60,06%	68,30%	74,73%	79,31%	83,91%	88,90%	95,71%	100,00%	100,00%
		T. Leitura	30,22%	37,19%	42,70%	46,25%	49,08%	51,58%	53,90%	57,04%	57,75%	57,75%
		T. Escrita	20,85%	24,82%	27,84%	29,95%	32,42%	35,06%	38,20%	41,50%	42,25%	42,25%
		M. Cons. Leitura	14,84%	16,28%	17,60%	18,96%	20,45%	22,49%	24,49%	27,63%	28,34%	28,34%
		M. Cons. Escrita	2,51%	2,94%	3,63%	4,85%	6,22%	8,07%	10,28%	13,42%	14,17%	14,17%
		M. Cons. L/E	15,32%	17,69%	19,54%	21,29%	23,82%	26,60%	30,42%	34,35%	38,64%	38,64%
3	crc32	T. Acessos	47,64%	58,34%	66,77%	73,11%	77,78%	82,51%	87,96%	95,26%	100,00%	100,00%
		T. Leitura	29,38%	36,22%	41,81%	45,47%	48,34%	50,99%	53,59%	56,97%	57,69%	57,69%
		T. Escrita	20,13%	24,01%	27,00%	29,12%	31,51%	34,34%	37,78%	41,37%	42,31%	42,31%
		M. Cons. Leitura	14,43%	15,94%	17,26%	18,69%	20,23%	22,39%	24,64%	28,02%	28,74%	28,74%
		M. Cons. Escrita	2,20%	2,64%	3,27%	4,45%	5,86%	7,95%	10,36%	13,74%	14,68%	14,68%
		M. Cons. L/E	14,82%	17,00%	18,89%	20,66%	23,11%	26,02%	30,20%	34,52%	39,25%	39,25%
4	dijkstra	T. Acessos	89,50%	91,88%	93,42%	94,70%	96,38%	97,72%	98,43%	99,26%	100,00%	100,00%
		T. Leitura	75,21%	76,62%	77,63%	78,72%	80,22%	81,08%	81,49%	81,81%	82,04%	82,04%
		T. Escrita	15,11%	15,68%	16,13%	16,44%	16,68%	16,99%	17,34%	17,76%	17,96%	17,96%
		M. Cons. Leitura	36,84%	37,30%	37,85%	38,72%	39,83%	40,19%	40,51%	40,84%	41,07%	41,07%
		M. Cons. Escrita	1,06%	1,11%	1,16%	1,26%	1,42%	1,62%	1,89%	2,21%	2,41%	2,41%
		M. Cons. L/E	37,62%	38,12%	38,70%	39,60%	40,84%	41,34%	41,83%	42,35%	42,97%	42,97%
5	fft	T. Acessos	53,76%	64,93%	72,37%	77,49%	81,13%	84,71%	88,76%	93,98%	99,35%	100,00%
		T. Leitura	33,93%	40,87%	45,89%	48,76%	50,99%	52,80%	54,63%	57,24%	57,59%	57,59%
		T. Escrita	21,34%	25,65%	28,21%	29,91%	31,86%	33,95%	36,47%	39,08%	42,41%	42,41%
		M. Cons. Leitura	14,90%	16,00%	17,04%	18,13%	19,29%	20,74%	22,35%	24,96%	25,31%	25,31%
		M. Cons. Escrita	1,97%	2,32%	2,89%	3,85%	4,96%	6,41%	8,26%	10,87%	14,20%	14,20%
		M. Cons. L/E	15,30%	17,10%	18,55%	19,95%	21,97%	24,19%	27,04%	30,28%	35,50%	36,14%

Tabela 13 – Memória de 4 Mbytes e informações dos *traces* entre os números 1 e 5

	Trace	T. Mem/Bytes	4M	Num. Acessos		
1	basicmath	T. Acessos	100,00%	5979165		
		T. Leitura	59,91%	3581951		
		T. Escrita	40,07%	2395936		
		M. Cons. Leitura	22,01%	1316212		
		M. Cons. Escrita	5,93%	354499	Linhas Pesquisadas	5979165
		M. Cons. L/E	26,51%	1585228	Endereços diferentes	236336
2	bitcount	T. Acessos	100,00%	2087969		
		T. Leitura	57,75%	1205700		
		T. Escrita	42,25%	882269		
		M. Cons. Leitura	28,34%	591638		
		M. Cons. Escrita	14,17%	295927	Linhas Pesquisadas	2087969
		M. Cons. L/E	38,64%	806890	Endereços diferentes	220699
3	crc32	T. Acessos	100,00%	1940156		
		T. Leitura	57,69%	1119235		
		T. Escrita	42,31%	820921		
		M. Cons. Leitura	28,74%	557577		
		M. Cons. Escrita	14,68%	284819	Linhas Pesquisadas	1940156
		M. Cons. L/E	39,25%	761542	Endereços diferentes	222951
4	dijkstra	T. Acessos	100,00%	20136170		
		T. Leitura	82,04%	16520454		
		T. Escrita	17,96%	3615716		
		M. Cons. Leitura	41,07%	8269650		
		M. Cons. Escrita	2,41%	484639	Linhas Pesquisadas	20136170
		M. Cons. L/E	42,97%	8652368	Endereços diferentes	255295
5	fft	T. Acessos	100,00%	2513599		
		T. Leitura	57,59%	1447500		
		T. Escrita	42,41%	1066099		
		M. Cons. Leitura	25,31%	636105		
		M. Cons. Escrita	14,20%	357044	Linhas Pesquisadas	2513599
		M. Cons. L/E	36,14%	908538	Endereços diferentes	278470

Tabela 14 – Memórias de 4 bytes até 2 Kbytes para os *traces* entre os números 6 e 10

	Trace	T. Mem/Bytes	4	8	16	32	64	128	256	512	1K	2K
6	ispell	T. Acessos	3,68%	7,35%	8,32%	9,11%	10,43%	12,51%	15,65%	20,53%	27,86%	41,27%
		T. Leitura	3,68%	7,23%	8,21%	8,92%	10,17%	11,67%	13,41%	16,06%	19,95%	26,92%
		T. Escrita	0,12%	0,24%	0,40%	0,69%	1,16%	2,00%	3,41%	5,65%	9,20%	15,67%
		M. Cons. Leitura	3,68%	4,40%	4,83%	5,52%	6,74%	7,86%	8,99%	10,23%	11,59%	13,19%
		M. Cons. Escrita	0,05%	0,11%	0,14%	0,18%	0,24%	0,31%	0,42%	0,63%	0,87%	1,25%
		M. Cons. L/E	3,68%	4,40%	4,83%	5,52%	6,74%	7,87%	9,05%	10,32%	11,70%	13,35%
7	jpeg_C	T. Acessos	3,70%	7,40%	8,37%	9,15%	10,47%	12,49%	15,62%	20,45%	27,78%	41,22%
		T. Leitura	3,70%	7,28%	8,26%	8,97%	10,20%	11,67%	13,41%	16,04%	19,90%	26,89%
		T. Escrita	0,12%	0,24%	0,39%	0,69%	1,14%	1,99%	3,38%	5,59%	9,16%	15,62%
		M. Cons. Leitura	3,70%	4,43%	4,86%	5,54%	6,74%	7,86%	8,98%	10,21%	11,56%	13,14%
		M. Cons. Escrita	0,06%	0,11%	0,14%	0,18%	0,24%	0,31%	0,42%	0,62%	0,85%	1,22%
		M. Cons. L/E	3,70%	4,43%	4,86%	5,54%	6,74%	7,87%	9,04%	10,30%	11,67%	13,31%
8	jpeg_D	T. Acessos	4,13%	8,26%	9,21%	9,95%	11,20%	13,14%	16,13%	20,83%	28,79%	43,07%
		T. Leitura	4,13%	8,14%	9,08%	9,77%	10,94%	12,36%	14,02%	16,57%	20,61%	28,14%
		T. Escrita	0,12%	0,23%	0,39%	0,66%	1,10%	1,90%	3,25%	5,41%	9,39%	16,18%
		M. Cons. Leitura	4,13%	4,83%	5,25%	5,90%	7,05%	8,12%	9,20%	10,39%	11,69%	13,21%
		M. Cons. Escrita	0,05%	0,11%	0,14%	0,18%	0,23%	0,30%	0,40%	0,59%	0,82%	1,17%
		M. Cons. L/E	4,13%	4,83%	5,25%	5,90%	7,05%	8,13%	9,26%	10,48%	11,80%	13,36%
9	mad	T. Acessos	4,50%	8,99%	9,90%	10,62%	11,82%	13,68%	16,60%	21,24%	29,91%	44,58%
		T. Leitura	4,50%	8,87%	9,78%	10,43%	11,56%	12,93%	14,53%	17,01%	21,34%	29,16%
		T. Escrita	0,12%	0,23%	0,38%	0,64%	1,06%	1,84%	3,17%	5,37%	9,71%	16,61%
		M. Cons. Leitura	4,50%	5,16%	5,57%	6,20%	7,30%	8,34%	9,38%	10,54%	11,79%	13,25%
		M. Cons. Escrita	0,05%	0,10%	0,13%	0,17%	0,22%	0,28%	0,38%	0,57%	0,78%	1,13%
		M. Cons. L/E	4,50%	5,16%	5,57%	6,20%	7,30%	8,35%	9,43%	10,62%	11,89%	13,39%
10	patricia	T. Acessos	1,21%	2,21%	3,98%	6,70%	11,40%	18,03%	28,42%	42,21%	57,57%	70,87%
		T. Leitura	0,88%	1,77%	3,41%	5,19%	8,14%	13,07%	19,08%	26,84%	35,59%	44,52%
		T. Escrita	0,36%	0,71%	1,41%	2,73%	4,66%	7,59%	11,93%	17,72%	23,95%	28,60%
		M. Cons. Leitura	0,88%	1,77%	2,61%	4,12%	5,93%	7,38%	9,11%	11,50%	14,54%	17,55%
		M. Cons. Escrita	0,36%	0,41%	0,46%	0,54%	0,65%	0,86%	1,21%	1,29%	1,37%	1,47%
		M. Cons. L/E	0,88%	1,77%	2,61%	4,12%	6,15%	7,66%	9,43%	11,85%	14,97%	18,27%

Tabela 15 – Memórias de 4 Kbytes até 2 Mbytes para os *traces* entre os números 6 e 10

	Trace	T. Mem/Bytes	4K	8K	16K	32K	64K	128K	256K	512K	1M	2M
6	ispell	T. Acessos	54,72%	65,33%	72,43%	77,24%	80,73%	84,17%	88,30%	93,91%	99,34%	100,00%
		T. Leitura	34,71%	41,33%	46,12%	48,89%	51,03%	52,84%	54,67%	57,29%	57,66%	57,66%
		T. Escrita	21,45%	25,52%	27,88%	29,46%	31,30%	33,42%	36,05%	38,98%	42,34%	42,34%
		M. Cons. Leitura	14,80%	15,89%	16,89%	17,98%	19,13%	20,57%	22,19%	24,81%	25,19%	25,19%
		M. Cons. Escrita	1,75%	2,08%	2,58%	3,52%	4,62%	6,12%	8,30%	10,92%	14,28%	14,28%
		M. Cons. L/E	15,14%	16,80%	18,21%	19,55%	21,47%	23,70%	26,59%	30,17%	35,41%	36,07%
7	jpeg_C	T. Acessos	54,59%	65,20%	72,33%	77,14%	80,65%	84,11%	88,33%	93,98%	99,36%	100,00%
		T. Leitura	34,65%	41,24%	46,04%	48,84%	50,99%	52,79%	54,71%	57,35%	57,64%	57,64%
		T. Escrita	21,39%	25,47%	27,83%	29,42%	31,23%	33,36%	36,05%	39,00%	42,36%	42,36%
		M. Cons. Leitura	14,75%	15,84%	16,84%	17,92%	19,07%	20,52%	22,23%	24,87%	25,16%	25,16%
		M. Cons. Escrita	1,72%	2,05%	2,55%	3,45%	4,53%	6,12%	8,31%	10,95%	14,31%	14,31%
		M. Cons. L/E	15,07%	16,71%	18,12%	19,46%	21,36%	23,56%	26,53%	30,17%	35,45%	36,09%
8	jpeg_D	T. Acessos	55,93%	66,13%	72,98%	77,59%	80,94%	84,36%	88,70%	94,17%	99,38%	100,00%
		T. Leitura	35,60%	41,94%	46,56%	49,23%	51,28%	53,00%	54,84%	57,36%	57,64%	57,64%
		T. Escrita	21,72%	25,64%	27,89%	29,42%	31,19%	33,30%	36,18%	39,15%	42,36%	42,36%
		M. Cons. Leitura	14,75%	15,79%	16,75%	17,78%	18,88%	20,26%	21,89%	24,41%	24,69%	24,69%
		M. Cons. Escrita	1,65%	1,96%	2,43%	3,31%	4,45%	6,00%	8,48%	11,00%	14,21%	14,21%
		M. Cons. L/E	15,06%	16,62%	17,97%	19,25%	21,06%	23,28%	26,15%	30,02%	35,06%	35,68%
9	mad	T. Acessos	56,96%	66,80%	73,38%	77,80%	81,01%	84,31%	88,50%	94,21%	99,34%	100,00%
		T. Leitura	36,35%	42,47%	46,91%	49,48%	51,43%	53,07%	54,85%	57,24%	57,62%	57,62%
		T. Escrita	21,95%	25,72%	27,89%	29,35%	31,03%	33,11%	35,88%	39,26%	42,38%	42,38%
		M. Cons. Leitura	14,73%	15,73%	16,65%	17,63%	18,68%	19,99%	21,58%	23,97%	24,35%	24,35%
		M. Cons. Escrita	1,58%	1,88%	2,33%	3,16%	4,27%	5,79%	8,18%	10,65%	13,76%	13,76%
		M. Cons. L/E	15,03%	16,53%	17,82%	19,05%	20,77%	22,90%	25,68%	29,46%	34,24%	34,90%
10	patricia	T. Acessos	81,49%	88,67%	91,93%	93,78%	94,95%	95,96%	97,06%	98,41%	99,75%	100,00%
		T. Leitura	52,29%	56,13%	58,21%	59,46%	60,22%	60,85%	61,41%	61,93%	62,33%	62,33%
		T. Escrita	31,75%	33,47%	34,28%	34,71%	35,09%	35,60%	36,22%	37,05%	37,67%	37,67%
		M. Cons. Leitura	18,58%	19,26%	19,70%	20,02%	20,40%	20,82%	21,25%	21,77%	22,18%	22,18%
		M. Cons. Escrita	1,60%	1,67%	1,76%	1,92%	2,18%	2,52%	3,04%	3,60%	4,21%	4,21%
		M. Cons. L/E	19,74%	20,50%	21,15%	21,56%	22,00%	22,60%	23,33%	24,23%	25,28%	25,52%

Tabela 16 – Memória de 4 Mbytes e informações dos *traces* entre os números 6 e 10

	Trace	T. Mem/Bytes	4M	Num. Acessos		
6	ispell	T. Acessos	100,00%	2501059		
		T. Leitura	57,66%	1442178		
		T. Escrita	42,34%	1058881		
		M. Cons. Leitura	25,19%	629898		
		M. Cons. Escrita	14,28%	357249	Linhas Pesquisadas	2501059
		M. Cons. L/E	36,07%	902056	Endereços diferentes	278661
7	jpeg_C	T. Acessos	100,00%	2485380		
		T. Leitura	57,64%	1432689		
		T. Escrita	42,36%	1052691		
		M. Cons. Leitura	25,16%	625426		
		M. Cons. Escrita	14,31%	355619	Linhas Pesquisadas	2485380
		M. Cons. L/E	36,09%	896978	Endereços diferentes	278103
8	jpeg_D	T. Acessos	100,00%	2605109		
		T. Leitura	57,64%	1501528		
		T. Escrita	42,36%	1103581		
		M. Cons. Leitura	24,69%	643267		
		M. Cons. Escrita	14,21%	370260	Linhas Pesquisadas	2605109
		M. Cons. L/E	35,68%	929419	Endereços diferentes	278312
9	mad	T. Acessos	100,00%	2744396		
		T. Leitura	57,62%	1581361		
		T. Escrita	42,38%	1163035		
		M. Cons. Leitura	24,35%	668219		
		M. Cons. Escrita	13,76%	377744	Linhas Pesquisadas	2744396
		M. Cons. L/E	34,90%	957664	Endereços diferentes	280169
10	patricia	T. Acessos	100,00%	12555633		
		T. Leitura	62,33%	7826524		
		T. Escrita	37,67%	4729109		
		M. Cons. Leitura	22,18%	2784305		
		M. Cons. Escrita	4,21%	529178	Linhas Pesquisadas	12555633
		M. Cons. L/E	25,52%	3204499	Endereços diferentes	293103

Tabela 17 – Memórias de 4 bytes até 2 Kbytes para os *traces* entre os números 11 e 15

	Trace	T. Mem/Bytes	4	8	16	32	64	128	256	512	1K	2K
11	qsort	T. Acessos	3,67%	6,95%	12,09%	19,89%	29,04%	41,61%	55,20%	67,69%	74,91%	78,62%
		T. Leitura	3,03%	5,68%	9,09%	13,92%	19,89%	26,67%	34,59%	42,70%	47,30%	49,29%
		T. Escrita	1,83%	3,01%	5,34%	7,65%	11,66%	16,59%	22,38%	27,50%	28,72%	30,28%
		M. Cons. Leitura	0,81%	1,44%	2,20%	3,25%	4,40%	6,42%	9,39%	10,99%	11,75%	12,50%
		M. Cons. Escrita	0,25%	0,38%	0,63%	1,14%	1,73%	1,78%	1,81%	1,87%	1,93%	2,03%
		M. Cons. L/E	0,81%	1,44%	2,20%	3,25%	4,52%	6,55%	10,51%	12,56%	13,41%	14,20%
12	rijndael	T. Acessos	3,08%	6,16%	7,19%	8,02%	9,44%	11,72%	15,08%	20,26%	27,24%	38,97%
		T. Leitura	3,08%	6,04%	7,06%	7,81%	9,18%	10,79%	12,66%	15,49%	19,50%	25,42%
		T. Escrita	0,14%	0,25%	0,42%	0,74%	1,24%	2,14%	3,66%	6,03%	9,24%	15,01%
		M. Cons. Leitura	3,08%	3,84%	4,28%	5,01%	6,36%	7,54%	8,74%	10,05%	11,47%	13,18%
		M. Cons. Escrita	0,06%	0,12%	0,15%	0,19%	0,25%	0,33%	0,46%	0,68%	0,94%	1,37%
		M. Cons. L/E	3,08%	3,84%	4,28%	5,01%	6,36%	7,55%	8,80%	10,14%	11,59%	13,36%
13	sha	T. Acessos	17,96%	26,05%	36,23%	52,38%	61,04%	64,89%	68,63%	73,98%	78,04%	81,82%
		T. Leitura	14,73%	22,82%	29,85%	40,91%	46,81%	49,91%	53,02%	56,77%	59,21%	61,20%
		T. Escrita	3,23%	4,80%	7,95%	13,02%	14,85%	15,49%	16,19%	17,44%	19,20%	21,03%
		M. Cons. Leitura	8,09%	9,04%	9,82%	10,82%	11,16%	11,63%	11,97%	12,32%	12,66%	13,06%
		M. Cons. Escrita	0,01%	0,03%	0,03%	0,04%	0,06%	0,08%	0,10%	0,15%	0,22%	0,31%
		M. Cons. L/E	8,09%	9,04%	9,82%	10,82%	11,16%	11,63%	11,98%	12,33%	12,69%	13,10%
14	stringsearch	T. Acessos	0,88%	1,29%	1,90%	2,96%	4,88%	7,93%	12,49%	19,39%	28,99%	40,93%
		T. Leitura	0,88%	1,18%	1,72%	2,74%	4,61%	6,68%	9,22%	12,98%	18,39%	25,14%
		T. Escrita	0,21%	0,34%	0,57%	0,98%	1,69%	2,92%	4,93%	8,09%	12,43%	17,70%
		M. Cons. Leitura	0,88%	1,16%	1,70%	2,72%	4,54%	6,12%	7,67%	9,29%	11,18%	13,48%
		M. Cons. Escrita	0,09%	0,18%	0,23%	0,30%	0,39%	0,51%	0,68%	0,99%	1,35%	1,95%
		M. Cons. L/E	0,88%	1,16%	1,70%	2,72%	4,54%	6,16%	7,78%	9,44%	11,39%	13,78%
15	susan	T. Acessos	0,85%	1,65%	2,82%	3,84%	5,50%	8,19%	11,93%	17,82%	26,03%	35,94%
		T. Leitura	0,85%	1,65%	2,68%	3,56%	5,20%	7,31%	9,51%	12,76%	17,53%	23,25%
		T. Escrita	0,18%	0,28%	0,46%	0,80%	1,37%	2,36%	4,04%	6,72%	10,31%	14,65%
		M. Cons. Leitura	0,85%	1,65%	2,18%	3,05%	4,61%	6,21%	7,78%	9,39%	11,04%	13,04%
		M. Cons. Escrita	0,06%	0,12%	0,20%	0,30%	0,38%	0,49%	0,64%	0,90%	1,23%	1,73%
		M. Cons. L/E	0,85%	1,65%	2,18%	3,05%	4,61%	6,21%	7,84%	9,53%	11,26%	13,34%

Tabela 18 – Memórias de 4 Kbytes até 2 Mbytes para os *traces* entre os números 11 e 15

	Trace	T. Mem/Bytes	4K	8K	16K	32K	64K	128K	256K	512K	1M	2M
11	qsort	T. Acessos	83,56%	87,52%	91,39%	93,76%	94,87%	95,64%	96,51%	97,65%	99,04%	100,00%
		T. Leitura	52,01%	54,69%	57,80%	59,36%	60,02%	60,48%	60,95%	61,43%	61,98%	61,98%
		T. Escrita	32,31%	33,66%	34,43%	34,81%	35,15%	35,61%	36,12%	36,70%	37,67%	38,01%
		M. Cons. Leitura	13,62%	15,24%	16,06%	16,40%	16,68%	17,02%	17,41%	17,89%	18,44%	18,44%
		M. Cons. Escrita	2,15%	2,22%	2,31%	2,46%	2,69%	2,98%	3,36%	3,84%	4,81%	5,15%
		M. Cons. L/E	15,34%	16,98%	18,04%	18,49%	18,87%	19,36%	19,98%	20,70%	21,66%	22,62%
12	rijndael	T. Acessos	53,11%	64,28%	71,77%	76,84%	80,52%	84,10%	88,35%	93,77%	99,29%	100,00%
		T. Leitura	33,56%	40,52%	45,57%	48,47%	50,72%	52,60%	54,53%	57,29%	57,69%	57,69%
		T. Escrita	21,06%	25,36%	27,89%	29,54%	31,44%	33,64%	36,26%	39,02%	42,31%	42,31%
		M. Cons. Leitura	14,87%	16,00%	17,06%	18,19%	19,39%	20,91%	22,62%	25,38%	25,77%	25,77%
		M. Cons. Escrita	1,91%	2,26%	2,78%	3,73%	4,89%	6,43%	8,34%	11,10%	14,39%	14,39%
		M. Cons. L/E	15,25%	17,04%	18,51%	19,91%	21,90%	24,23%	27,23%	30,61%	36,14%	36,84%
13	sha	T. Acessos	86,64%	89,86%	91,86%	93,34%	94,58%	95,45%	96,52%	98,01%	99,65%	100,00%
		T. Leitura	63,87%	65,85%	67,22%	68,09%	68,84%	69,37%	69,87%	70,47%	71,06%	71,06%
		T. Escrita	23,16%	24,41%	25,07%	25,61%	26,05%	26,58%	27,29%	28,10%	28,94%	28,94%
		M. Cons. Leitura	13,49%	13,86%	14,17%	14,46%	14,76%	15,15%	15,55%	16,16%	16,74%	16,74%
		M. Cons. Escrita	0,44%	0,51%	0,62%	0,78%	1,04%	1,41%	1,99%	2,60%	3,44%	3,44%
		M. Cons. L/E	13,56%	14,03%	14,48%	14,85%	15,26%	15,78%	16,54%	17,47%	18,69%	19,03%
14	stringsearch	T. Acessos	52,85%	63,07%	70,38%	76,06%	80,24%	84,78%	90,10%	96,98%	100,00%	100,00%
		T. Leitura	32,51%	39,38%	44,17%	47,33%	49,89%	52,18%	54,95%	58,31%	58,31%	58,31%
		T. Escrita	22,32%	25,81%	28,33%	30,17%	32,49%	35,08%	38,77%	41,69%	41,69%	41,69%
		M. Cons. Leitura	15,69%	17,18%	18,56%	19,96%	21,48%	23,63%	26,13%	29,49%	29,49%	29,49%
		M. Cons. Escrita	2,68%	3,15%	3,82%	5,04%	6,56%	8,71%	11,41%	14,33%	14,33%	14,33%
		M. Cons. L/E	16,34%	18,70%	20,64%	22,51%	24,96%	27,91%	32,21%	37,02%	40,04%	40,04%
15	susan	T. Acessos	47,68%	59,48%	69,01%	75,19%	79,62%	83,79%	88,44%	94,98%	100,00%	100,00%
		T. Leitura	30,04%	37,10%	43,59%	47,11%	49,82%	52,19%	54,49%	57,37%	58,31%	58,31%
		T. Escrita	19,51%	24,39%	27,46%	29,43%	31,55%	34,04%	36,95%	40,26%	41,66%	41,66%
		M. Cons. Leitura	15,16%	16,65%	17,91%	19,19%	20,59%	22,50%	24,43%	27,31%	28,26%	28,26%
		M. Cons. Escrita	2,36%	2,75%	3,29%	4,34%	5,64%	7,32%	9,41%	12,29%	13,69%	13,69%
		M. Cons. L/E	15,63%	17,93%	19,74%	21,37%	23,53%	26,17%	29,69%	33,44%	38,24%	38,24%

Tabela 19 – Memória de 4 Mbytes e informações dos *traces* entre os números 11 e 15

	Trace	T. Mem/Bytes	4M	Num. Acessos		
11	qsort	T. Acessos	100,00%	13589264		
		T. Leitura	61,98%	8422822		
		T. Escrita	38,01%	5165372		
		M. Cons. Leitura	18,44%	2505211		
		M. Cons. Escrita	5,15%	700361	Linhas Pesquisadas	13589264
		M. Cons. L/E	22,62%	3074081	Endereços diferentes	392182
12	rijndael	T. Acessos	100,00%	2372934		
		T. Leitura	57,69%	1368943		
		T. Escrita	42,31%	1003991		
		M. Cons. Leitura	25,77%	611608		
		M. Cons. Escrita	14,39%	341561	Linhas Pesquisadas	2372934
		M. Cons. L/E	36,84%	874211	Endereços diferentes	278893
13	sha	T. Acessos	100,00%	10799954		
		T. Leitura	71,06%	7673953		
		T. Escrita	28,94%	3126001		
		M. Cons. Leitura	16,74%	1808159		
		M. Cons. Escrita	3,44%	372048	Linhas Pesquisadas	10799954
		M. Cons. L/E	19,03%	2055744	Endereços diferentes	299864
14	stringsearch	T. Acessos	100,00%	1524857		
		T. Leitura	58,31%	889088		
		T. Escrita	41,69%	635769		
		M. Cons. Leitura	29,49%	449653		
		M. Cons. Escrita	14,33%	218560	Linhas Pesquisadas	1524857
		M. Cons. L/E	40,04%	610626	Endereços diferentes	177184
15	susan	T. Acessos	100,00%	2273879		
		T. Leitura	58,31%	1325993		
		T. Escrita	41,66%	947200		
		M. Cons. Leitura	28,26%	642531		
		M. Cons. Escrita	13,69%	311192	Linhas Pesquisadas	2273879
		M. Cons. L/E	38,24%	869453	Endereços diferentes	240151

Tabela 20 – Memórias de 4 bytes até 2 Kbytes para os *traces* entre os números 16 e 18

	Trace	T. Mem/Bytes	4	8	16	32	64	128	256	512	1K	2K
16	tiff2bw	T. Acessos	4,05%	8,09%	9,03%	9,79%	11,08%	13,11%	16,14%	20,90%	28,74%	42,93%
		T. Leitura	4,05%	7,97%	8,91%	9,59%	10,82%	12,28%	13,97%	16,56%	20,58%	28,04%
		T. Escrita	0,12%	0,24%	0,39%	0,68%	1,13%	1,94%	3,32%	5,51%	9,40%	16,17%
		M. Cons. Leitura	4,05%	4,74%	5,16%	5,82%	7,02%	8,11%	9,21%	10,42%	11,73%	13,28%
		M. Cons. Escrita	0,05%	0,11%	0,14%	0,18%	0,23%	0,30%	0,41%	0,61%	0,85%	1,23%
		M. Cons. L/E	4,05%	4,74%	5,16%	5,82%	7,02%	8,12%	9,26%	10,50%	11,84%	13,44%
17	tiff2rgba	T. Acessos	3,99%	7,97%	8,92%	9,69%	11,00%	13,10%	16,16%	20,98%	28,73%	42,86%
		T. Leitura	3,99%	7,85%	8,79%	9,47%	10,74%	12,24%	13,95%	16,57%	20,59%	28,00%
		T. Escrita	0,13%	0,25%	0,40%	0,69%	1,16%	1,97%	3,37%	5,59%	9,43%	16,17%
		M. Cons. Leitura	3,99%	4,68%	5,09%	5,76%	7,01%	8,11%	9,23%	10,44%	11,77%	13,34%
		M. Cons. Escrita	0,05%	0,10%	0,14%	0,18%	0,23%	0,30%	0,42%	0,63%	0,87%	1,27%
		M. Cons. L/E	3,99%	4,68%	5,09%	5,76%	7,01%	8,11%	9,28%	10,52%	11,88%	13,50%
18	typeset	T. Acessos	3,76%	7,52%	8,49%	9,26%	10,57%	12,61%	15,71%	20,52%	27,92%	41,51%
		T. Leitura	3,76%	7,41%	8,37%	9,08%	10,31%	11,79%	13,51%	16,12%	20,01%	27,09%
		T. Escrita	0,12%	0,24%	0,39%	0,68%	1,14%	1,97%	3,37%	5,56%	9,19%	15,71%
		M. Cons. Leitura	3,76%	4,48%	4,91%	5,59%	6,79%	7,90%	9,02%	10,25%	11,58%	13,16%
		M. Cons. Escrita	0,05%	0,11%	0,14%	0,18%	0,23%	0,31%	0,42%	0,62%	0,85%	1,23%
		M. Cons. L/E	3,76%	4,48%	4,91%	5,59%	6,79%	7,91%	9,07%	10,33%	11,70%	13,32%

Tabela 21 – Memórias de 4 Kbytes até 2 Mbytes para os *traces* entre os números 16 e 18

	Trace	T. Mem/Bytes	4K	8K	16K	32K	64K	128K	256K	512K	1M	2M
16	tiff2bw	T. Acessos	55,91%	66,20%	73,14%	77,80%	81,16%	84,45%	88,72%	94,22%	99,38%	100,00%
		T. Leitura	35,57%	41,98%	46,65%	49,33%	51,39%	53,11%	54,86%	57,37%	57,69%	57,69%
		T. Escrita	21,74%	25,70%	28,01%	29,53%	31,26%	33,30%	36,17%	39,10%	42,31%	42,31%
		M. Cons. Leitura	14,85%	15,89%	16,85%	17,88%	18,98%	20,36%	21,91%	24,42%	24,75%	24,75%
		M. Cons. Escrita	1,71%	2,03%	2,50%	3,36%	4,43%	5,91%	8,35%	10,86%	14,07%	14,07%
		M. Cons. L/E	15,18%	16,80%	18,15%	19,43%	21,23%	23,36%	26,16%	29,95%	34,97%	35,59%
17	tiff2rgba	T. Acessos	55,91%	66,24%	73,19%	77,84%	81,19%	84,47%	88,76%	94,23%	99,37%	100,00%
		T. Leitura	35,57%	42,01%	46,69%	49,34%	51,39%	53,10%	54,89%	57,38%	57,71%	57,71%
		T. Escrita	21,74%	25,72%	28,05%	29,56%	31,28%	33,31%	36,17%	39,09%	42,29%	42,29%
		M. Cons. Leitura	14,92%	15,95%	16,90%	17,93%	19,02%	20,39%	21,98%	24,48%	24,80%	24,80%
		M. Cons. Escrita	1,77%	2,08%	2,55%	3,40%	4,46%	5,94%	8,33%	10,83%	14,03%	14,03%
		M. Cons. L/E	15,27%	16,92%	18,25%	19,53%	21,31%	23,44%	26,23%	29,99%	34,99%	35,62%
18	typeset	T. Acessos	54,79%	65,34%	72,40%	77,17%	80,65%	84,09%	88,30%	93,97%	99,34%	100,00%
		T. Leitura	34,79%	41,35%	46,12%	48,87%	51,00%	52,78%	54,67%	57,27%	57,64%	57,64%
		T. Escrita	21,43%	25,49%	27,83%	29,40%	31,19%	33,33%	36,01%	39,03%	42,36%	42,36%
		M. Cons. Leitura	14,75%	15,82%	16,82%	17,88%	19,01%	20,44%	22,12%	24,72%	25,09%	25,09%
		M. Cons. Escrita	1,72%	2,05%	2,53%	3,43%	4,52%	6,09%	8,29%	10,88%	14,21%	14,21%
		M. Cons. L/E	15,08%	16,71%	18,10%	19,43%	21,29%	23,48%	26,42%	30,04%	35,23%	35,89%

Tabela 22 – Memória de 4 Mbytes e informações dos *traces* entre os números 16 e 18

	Trace	T. Mem/Bytes	4M	Num. Acessos		
16	tiff2bw	T. Acessos	100,00%	2609214		
		T. Leitura	57,69%	1505362		
		T. Escrita	42,31%	1103852		
		M. Cons. Leitura	24,75%	645678		
		M. Cons. Escrita	14,07%	367016	Linhas Pesquisadas	2609214
		M. Cons. L/E	35,59%	928574	Endereços diferentes	278293
17	tiff2rgba	T. Acessos	100,00%	2622117		
		T. Leitura	57,71%	1513165		
		T. Escrita	42,29%	1108952		
		M. Cons. Leitura	24,80%	650305		
		M. Cons. Escrita	14,03%	367998	Linhas Pesquisadas	2622117
		M. Cons. L/E	35,62%	934117	Endereços diferentes	278719
18	typeset	T. Acessos	100,00%	2526776		
		T. Leitura	57,64%	1456353		
		T. Escrita	42,36%	1070423		
		M. Cons. Leitura	25,09%	633890		
		M. Cons. Escrita	14,21%	359074	Linhas Pesquisadas	2526776
		M. Cons. L/E	35,89%	906868	Endereços diferentes	278796

Apêndice B – Resumo dos resultados do apêndice A para todos os benchmarks

Tabela 23 – Memórias de 4 bytes até 4 Kbytes para todos os *traces*

	T. Mem/Bytes	4	8	16	32	64	128	256	512	1K	2K	4K
T. Acessos	Menor	0,85%	1,27%	1,90%	2,89%	4,67%	7,24%	11,21%	17,31%	25,62%	35,89%	47,64%
	Maior	17,96%	26,05%	36,23%	52,38%	61,50%	65,81%	70,74%	75,59%	80,69%	85,36%	89,50%
	Média	4,11%	7,54%	10,04%	13,27%	16,44%	20,06%	24,80%	31,18%	39,29%	50,36%	61,56%
T. Leitura	Menor	0,85%	1,18%	1,72%	2,62%	4,36%	6,26%	8,43%	11,78%	16,54%	22,47%	29,38%
	Maior	14,73%	22,82%	31,70%	45,20%	54,49%	57,46%	60,36%	63,49%	68,14%	72,49%	75,21%
	Média	3,82%	6,99%	8,96%	11,42%	13,95%	16,40%	19,21%	22,86%	27,46%	33,64%	40,30%
T. Escrita	Menor	0,12%	0,23%	0,38%	0,64%	1,06%	1,84%	3,17%	5,37%	9,16%	14,45%	15,11%
	Maior	3,23%	4,80%	7,95%	13,02%	14,85%	16,59%	22,38%	27,50%	28,72%	30,28%	32,31%
	Média	0,59%	1,00%	1,68%	2,57%	3,56%	4,98%	6,96%	9,72%	13,21%	18,22%	22,79%
M. Cons. Leitura	Menor	0,70%	1,16%	1,70%	2,62%	4,27%	5,65%	7,10%	8,66%	10,33%	12,33%	13,49%
	Maior	11,55%	17,31%	24,99%	31,07%	31,98%	32,77%	33,95%	34,76%	35,41%	36,12%	36,84%
	Média	3,32%	4,27%	5,20%	6,35%	7,66%	8,89%	10,21%	11,57%	13,03%	14,68%	16,23%
M. Cons. Escrita	Menor	0,01%	0,03%	0,03%	0,04%	0,06%	0,08%	0,10%	0,15%	0,22%	0,31%	0,44%
	Maior	0,36%	0,41%	0,63%	1,14%	1,73%	1,78%	1,81%	1,87%	1,93%	2,03%	2,68%
	Média	0,10%	0,15%	0,21%	0,29%	0,41%	0,50%	0,63%	0,81%	1,03%	1,37%	1,82%
M. Cons. L/E	Menor	0,70%	1,16%	1,70%	2,62%	4,27%	5,66%	7,18%	8,77%	10,48%	12,53%	13,56%
	Maior	11,55%	17,31%	24,99%	31,07%	31,98%	32,77%	34,32%	35,40%	36,13%	36,86%	37,62%
	Média	3,32%	4,27%	5,20%	6,35%	7,69%	8,93%	10,37%	11,80%	13,30%	15,04%	16,77%

Tabela 24 – Memórias de 8 Kbytes até 4 Mbytes para todos os *traces*

	T. Mem/Bytes	8K	16K	32K	64K	128K	256K	512K	1M	2M	4M
T. Acessos	Menor	58,34%	66,77%	73,11%	77,78%	82,51%	87,96%	93,77%	99,04%	100,00%	100,00%
	Maior	91,88%	93,42%	94,70%	96,38%	97,72%	98,43%	99,26%	100,00%	100,00%	100,00%
	Média	70,66%	76,91%	81,20%	84,31%	87,34%	90,89%	95,60%	99,59%	100,00%	100,00%
T. Leitura	Menor	36,22%	41,81%	45,47%	48,34%	50,99%	53,59%	56,97%	57,59%	57,59%	57,59%
	Maior	76,62%	77,63%	78,72%	80,22%	81,08%	81,49%	81,81%	82,04%	82,04%	82,04%
	Média	45,96%	50,17%	52,67%	54,60%	56,22%	57,87%	60,03%	60,46%	60,46%	60,46%
T. Escrita	Menor	15,68%	16,13%	16,44%	16,68%	16,99%	17,34%	17,76%	17,96%	17,96%	17,96%
	Maior	33,66%	34,86%	35,61%	36,33%	37,26%	38,77%	41,69%	42,41%	42,41%	42,41%
	Média	26,12%	28,16%	29,52%	31,06%	32,85%	35,13%	37,55%	39,52%	39,54%	39,54%
M. Cons. Leitura	Menor	13,86%	14,17%	14,46%	14,76%	15,15%	15,55%	16,16%	16,74%	16,74%	16,74%
	Maior	37,30%	37,85%	38,72%	39,83%	40,19%	40,51%	40,84%	41,07%	41,07%	41,07%
	Média	17,31%	18,25%	19,21%	20,26%	21,55%	22,99%	25,15%	25,58%	25,58%	25,58%
M. Cons. Escrita	Menor	0,51%	0,62%	0,78%	1,04%	1,41%	1,89%	2,21%	2,41%	2,41%	2,41%
	Maior	3,15%	3,82%	5,04%	6,56%	8,71%	11,41%	14,33%	14,68%	14,68%	14,68%
	Média	2,10%	2,51%	3,27%	4,21%	5,50%	7,28%	9,42%	11,40%	11,42%	11,42%
M. Cons. L/E	Menor	14,03%	14,48%	14,85%	15,26%	15,78%	16,54%	17,47%	18,69%	19,03%	19,03%
	Maior	38,12%	38,70%	39,60%	40,84%	41,34%	41,83%	42,35%	42,97%	42,97%	42,97%
	Média	18,36%	19,67%	20,88%	22,53%	24,43%	26,97%	29,97%	33,79%	34,20%	34,20%

Apêndice C – Resultados das comparações entre as SPMs (linha base SPM 1)

Tabela 25 – Comparação entre as SPMs para a análise estática parte 1 (linha base SPM 1)

	SPM 1 (SRAM 32K) linha base					
Análise Estática	SPM 2			SPM 3		
Todos Benchmarks	SRAM 16KB + STT 64KB			SRAM 16KB + STT 32KB		
Parâmetros	Menor	Maior	Média	Menor	Maior	Média
Área (um^2)	-4,95%	-4,95%	-4,95%	18,93%	18,93%	18,93%
Número de Endereços	-150,00%	-150,00%	-150,00%	-50,00%	-50,00%	-50,00%
Número Acessos Total	-2,85%	-0,40%	-1,90%	-1,09%	-0,09%	-0,62%
Número Acessos Leitura	-6,43%	-0,90%	-4,02%	-3,29%	-0,49%	-2,07%
Número Acessos Escrita	-0,42%	2,94%	1,67%	0,05%	3,11%	1,83%
Número de Alocações	-109,74%	-59,45%	-82,84%	-41,00%	-13,66%	-23,92%
Latência Leitura (ns)	4,60%	9,28%	6,55%	9,10%	11,64%	10,21%
Latência Escrita (ns)	-10,44%	13,46%	9,93%	-10,08%	14,08%	10,53%
Energia Leitura (pJ)	49,47%	55,74%	50,91%	50,98%	56,89%	52,02%
Energia Escrita (pJ)	-113,75%	12,17%	-4,58%	-86,98%	17,68%	4,14%
Latência Total (ns)	2,81%	9,76%	8,09%	5,22%	11,90%	10,63%
Energia Total (pJ)	46,76%	52,50%	48,03%	48,66%	54,17%	49,52%
Leakage (mW)	45,89%	45,89%	45,89%	47,48%	47,48%	47,48%

Tabela 26 – Comparação entre as SPMs para a análise estática parte 2 (linha base SPM 1)

	SPM 1 (SRAM 32K) linha base					
Análise Estática	SPM 4			SPM 5		
Todos Benchmarks	SRAM 16KB + STT 32KB + PCM 8KB			SRAM 16KB + STT 64KB + PCM 8KB		
Parâmetros	Menor	Maior	Média	Menor	Maior	Média
Área (um^2)	16,29%	16,29%	16,29%	-7,59%	-7,59%	-7,59%
Número de Endereços	-75,00%	-75,00%	-75,00%	-175,00%	-175,00%	-175,00%
Número Acessos Total	-1,72%	-0,20%	-1,02%	-3,31%	-0,46%	-2,13%
Número Acessos Leitura	-4,27%	-0,63%	-2,67%	-7,17%	-0,98%	-4,38%
Número Acessos Escrita	-0,02%	2,90%	1,77%	-0,51%	2,80%	1,65%
Número de Alocações	-66,70%	-37,12%	-46,59%	-129,20%	-77,56%	-103,16%
Latência Leitura (ns)	25,62%	42,52%	30,67%	24,00%	40,20%	28,18%
Latência Escrita (ns)	-10,15%	13,58%	10,37%	-10,44%	13,14%	9,87%
Energia Leitura (pJ)	50,60%	56,61%	51,73%	49,32%	55,92%	50,91%
Energia Escrita (pJ)	-87,23%	17,33%	3,60%	-113,76%	12,16%	-4,84%
Latência Total (ns)	15,99%	34,12%	23,79%	14,63%	32,01%	22,03%
Energia Total (pJ)	48,20%	53,88%	49,22%	46,60%	52,66%	48,02%
Leakage (mW)	46,65%	46,65%	46,65%	45,05%	45,05%	45,05%

Tabela 27 – Comparação entre as SPMs para a análise dinâmica parte 1 (linha base SPM 1)

	SPM 1 (SRAM 32K) linha base					
Análise Estática	SPM 2			SPM 3		
Todos Benchmarks	SRAM 16KB + STT 64KB			SRAM 16KB + STT 32KB		
Parâmetros	Menor	Maior	Média	Menor	Maior	Média
Área (um^2)	-4,95%	-4,95%	-4,95%	18,93%	18,93%	18,93%
Número de Endereços	-150,00%	-150,00%	-150,00%	-50,00%	-50,00%	-50,00%
Número Acessos Total	-2,90%	-0,40%	-1,90%	-1,03%	-0,07%	-0,55%
Número Acessos Leitura	-6,48%	-0,90%	-3,99%	-3,15%	-0,47%	-1,97%
Número Acessos Escrita	-0,42%	2,85%	1,65%	-0,03%	3,19%	1,84%
Número de Alocações	-191,73%	-138,89%	-151,09%	18,93%	18,93%	18,93%
Latência Leitura (ns)	4,54%	9,27%	6,57%	-96,92%	-56,21%	-66,62%
Latência Escrita (ns)	-10,45%	13,25%	9,88%	9,17%	11,68%	10,29%
Energia Leitura (pJ)	49,46%	55,77%	50,93%	-10,07%	14,17%	10,54%
Energia Escrita (pJ)	-113,80%	12,11%	-4,74%	51,06%	56,94%	52,07%
Latência Total (ns)	2,84%	9,75%	8,08%	-86,97%	17,71%	4,18%
Energia Total (pJ)	46,71%	52,52%	48,04%	5,24%	11,93%	10,69%
Leakage (mW)	45,89%	45,89%	45,89%	48,75%	54,21%	49,57%

Tabela 28 – Comparação entre as SPMs para a análise dinâmica parte 2 (linha base SPM 1)

	SPM 1 (SRAM 32K) linha base					
Análise Dinâmica	SPM 4			SPM 5		
Todos Benchmarks	SRAM 16KB + STT 32KB + PCM 8KB			SRAM 16KB + STT 64KB + PCM 8KB		
Parâmetros	Menor	Maior	Média	Menor	Maior	Média
Área (um^2)	16,29%	16,29%	16,29%	-7,59%	-7,59%	-7,59%
Número de Endereços	-75,00%	-75,00%	-75,00%	-175,00%	-175,00%	-175,00%
Número Acessos Total	-1,68%	-0,20%	-0,99%	-3,35%	-0,46%	-2,13%
Número Acessos Leitura	-4,22%	-0,62%	-2,62%	-7,20%	-0,98%	-4,35%
Número Acessos Escrita	-0,09%	2,88%	1,75%	-0,51%	2,76%	1,63%
Número de Alocações	-177,14%	-113,10%	-127,35%	-257,77%	-200,14%	-217,65%
Latência Leitura (ns)	25,64%	42,59%	30,72%	24,00%	40,23%	28,20%
Latência Escrita (ns)	-10,16%	13,53%	10,32%	-10,46%	13,12%	9,83%
Energia Leitura (pJ)	50,64%	56,65%	51,76%	49,32%	55,93%	50,93%
Energia Escrita (pJ)	-87,25%	17,21%	3,48%	-113,81%	12,09%	-4,95%
Latência Total (ns)	15,99%	34,17%	23,81%	14,62%	32,02%	22,03%
Energia Total (pJ)	48,22%	53,93%	49,24%	46,57%	52,67%	48,03%
Leakage (mW)	46,65%	46,65%	46,65%	45,05%	45,05%	45,05%

Apêndice D – Resultados das comparações entre as SPMs (linha base DRAM).

Tabela 29 – Comparação entre as SPMs para a análise estática parte 1 (linha base DRAM)

	DRAM 2MB linha base								
Análise Estática	SPM 1			SPM 2			SPM 3		
Todos Benchmarks	SRAM 32KB			SRAM 16KB + STT 64KB			SRAM 16KB + STT 32KB		
Parâmetros	Menor	Maior	Média	Menor	Maior	Média	Menor	Maior	Média
Área (um^2)	-0,83%	-0,83%	-0,83%	-0,87%	-0,87%	-0,87%	-0,68%	-0,68%	-0,68%
Número de Endereços	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
Número Acessos Total	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
Número Acessos Leitura	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
Número Acessos Escrita	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
Número de Alocações	-7,33%	-3,78%	-4,92%	-13,26%	-4,79%	-7,01%	-10,05%	-4,26%	-5,32%
Latência Leitura (ns)	17,38%	21,39%	19,22%	25,15%	29,44%	27,01%	25,56%	30,52%	27,79%
Latência Escrita (ns)	14,94%	20,46%	16,62%	11,70%	26,66%	20,83%	11,90%	27,51%	21,06%
Energia Leitura (pJ)	70,33%	86,58%	77,79%	78,99%	94,15%	85,32%	76,66%	92,86%	83,78%
Energia Escrita (pJ)	66,13%	90,55%	73,55%	64,28%	90,39%	72,33%	64,22%	90,31%	72,29%
Latência Total (ns)	16,35%	21,17%	18,16%	22,86%	28,24%	24,71%	23,18%	29,00%	25,29%
Energia Total (pJ)	68,55%	87,78%	75,93%	72,77%	93,19%	79,97%	71,40%	92,09%	79,04%
Leakage (mW)	-121,03%	-121,03%	-121,03%	-65,49%	-65,49%	-65,49%	-63,56%	-63,56%	-63,56%

Tabela 30 – Comparação entre as SPMs para a análise estática parte 2 (linha base DRAM)

	DRAM 2MB linha base					
Análise Estática	SPM 4			SPM 5		
Todos Benchmarks	SRAM 16KB + STT 32KB + PCM 8KB			SRAM 16KB + STT 64KB + PCM 8KB		
Parâmetros	-0,70%	-0,70%	-0,70%	-0,90%	-0,90%	-0,90%
Área (um^2)	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
Número de Endereços	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
Número Acessos Total	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
Número Acessos Leitura	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
Número Acessos Escrita	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
Número de Alocações	-11,96%	-5,02%	-6,37%	-14,23%	-5,71%	-7,85%
Latência Leitura (ns)	38,85%	54,54%	41,95%	38,97%	53,88%	41,73%
Latência Escrita (ns)	11,87%	27,49%	21,03%	11,70%	26,65%	20,82%
Energia Leitura (pJ)	77,39%	93,29%	84,26%	79,55%	94,28%	85,61%
Energia Escrita (pJ)	64,29%	90,36%	72,32%	64,31%	90,40%	72,34%
Latência Total (ns)	30,82%	47,54%	33,95%	30,82%	46,94%	33,73%
Energia Total (pJ)	71,85%	92,46%	79,34%	73,10%	93,31%	80,14%
Leakage (mW)	-64,57%	-64,57%	-64,57%	-66,50%	-66,50%	-66,50%

Tabela 31 – Comparação entre as SPMs para a análise dinâmica parte 1 (linha base DRAM)

	DRAM 2MB linha base								
Análise Estática	SPM 1			SPM 2			SPM 3		
Todos Benchmarks	SRAM 32KB			SRAM 16KB + STT 64KB			SRAM 16KB + STT 32KB		
Parâmetros	Menor	Maior	Média	Menor	Maior	Média	Menor	Maior	Média
Área (um^2)	-0,83%	-0,83%	-0,83%	-0,87%	-0,87%	-0,87%	-0,68%	-0,68%	-0,68%
Número de Endereços	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
Número Acessos Total	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
Número Acessos Leitura	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
Número Acessos Escrita	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
Número de Alocações	-17,33%	-10,28%	-13,52%	-43,71%	-21,97%	-29,18%	-33,25%	-15,04%	-20,46%
Latência Leitura (ns)	17,38%	21,39%	19,22%	25,15%	29,43%	27,00%	25,51%	30,52%	27,76%
Latência Escrita (ns)	14,94%	20,46%	16,61%	11,70%	26,65%	20,82%	11,90%	27,51%	21,06%
Energia Leitura (pJ)	70,32%	86,57%	77,80%	79,02%	94,11%	85,31%	76,53%	92,78%	83,71%
Energia Escrita (pJ)	66,14%	90,55%	73,54%	64,33%	90,41%	72,34%	64,19%	90,31%	72,27%
Latência Total (ns)	16,35%	21,17%	18,16%	22,85%	28,24%	24,70%	23,15%	28,99%	25,27%
Energia Total (pJ)	68,55%	87,78%	75,93%	72,81%	93,16%	79,97%	71,31%	92,05%	78,99%
Leakage (mW)	-121,03%	-121,03%	-121,03%	-65,49%	-65,49%	-65,49%	-63,56%	-63,56%	-63,56%

Tabela 32 – Comparação entre as SPMs para a análise dinâmica parte 2 (linha base DRAM)

	DRAM 2MB linha base					
Análise Dinâmica	SPM 4			SPM 5		
Todos Benchmarks	SRAM 16KB + STT 32KB + PCM 8KB			SRAM 16KB + STT 64KB + PCM 8KB		
Parâmetros	-0,70%	-0,70%	-0,70%	-0,90%	-0,90%	-0,90%
Área (um^2)	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
Número de Endereços	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
Número Acessos Total	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
Número Acessos Leitura	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
Número Acessos Escrita	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
Número de Alocações	-43,61%	-19,98%	-26,21%	-51,89%	-27,60%	-36,31%
Latência Leitura (ns)	38,83%	54,51%	41,93%	38,97%	53,87%	41,72%
Latência Escrita (ns)	11,87%	27,48%	21,02%	11,70%	26,65%	20,82%
Energia Leitura (pJ)	77,35%	93,21%	84,22%	79,57%	94,26%	85,59%
Energia Escrita (pJ)	64,30%	90,33%	72,33%	64,34%	90,41%	72,35%
Latência Total (ns)	30,80%	47,52%	33,94%	30,81%	46,93%	33,72%
Energia Total (pJ)	71,83%	92,40%	79,32%	73,12%	93,29%	80,14%
Leakage (mW)	-64,57%	-64,57%	-64,57%	-66,50%	-66,50%	-66,50%