Avaliação de Desempenho, Área e Energia de Caches com Controle de Poluição

Richard R de Souza, Giancarlo C Heck, Renato Carmo & Roberto A Hexsel
Departamento de Informática
Universidade Federal do Paraná
Curitiba, PR, Brasil
{rrs03,giancarlo,renato,roberto}@inf.ufpr.br

Resumo

Este artigo compara o desempenho, a área e o dispêndio de energia de quatro projetos do primeiro nível da hierarquia de memória para sistemas embarcados. Os quatro projetos são: ⟨i⟩ cache primária com mapeamento direto (MD); ⟨ii⟩ cache primária com associatividade binária; ⟨iii⟩ cache primária com MD associada a uma cache com controle de poluição; e ⟨iv⟩ cache primária com MD associada a uma cache de vítimas e controle de poluição. Os projetos foram simulados com os programas da suíte CommBench, e para estes programas, caches com 4-8 Kbytes e os projetos ⟨iii⟩ e ⟨iv⟩ tem bom desempenho; os sistemas com o projeto ⟨iv⟩ tem a melhor relação desempenho/área e energia.

1. Introdução

Memórias cache são fundamentais para esconder a latência de acesso à memória DRAM. Em sistemas embarcados, além de desempenho, consumo de energia e custo são parâmetros importantes que condicionam o projeto, e freqüentemente o projetista deve fazer escolhas não-triviais entre, ao menos, desempenho computacional, dispêndio de energia e custo, medido pela área do circuito integrado.

Em [4], diversas técnicas de projeto de memórias cache de dados são avaliadas quanto ao desempenho – redução na taxa de faltas e no número de ciclos por instrução (CPI), e à redução do espaço ocupado pela cache. Este trabalho [1] contém uma comparação, do ponto de vista de área e de energia, das técnicas de projeto mais promissoras, que são as *caches com controle de poluição.* Estas caches contém um *buffer* totalmente associativo interposto entre a memória e a cache de primeiro nível e sua função é impedir que blocos que estejam sendo usados ativamente pelo processador sejam expurgados por blocos que são referenciados uma única vez. Os sistemas foram simulados com versões modificadas do SimpleScalar [8], executando programas da suíte CommBench [11].

O texto está organizado como se segue. A Seção 2 descreve os quatro *buffers* que são comparados. A Seção 3 descreve o ambiente de simulação e os programas de teste, enquanto que a Seção 4 contém uma comparação de desempenho dos quatro projetos investigados com relação à taxa de faltas e a CPI. A Seção 5 traz as comparações de área e de energia. A Seção 6 descreve os padrões de referências à memória dos programas de teste e discute as relações entre os padrões e o ganho de desempenho obtido com os *buffers*. Finalmente, nossas conclusões e trabalhos futuros são apresentados na Seção 7.

2. Modelos dos Buffers

As quatro memórias especializadas, ou *buffers*, investigados são descritos no que se segue.

Caches não-bloqueantes Para permitir que um processador segmentado continue trabalhando durante a ocorrência de uma falta na cache, é necessário manter um registro da falta pendente para que, depois que o bloco faltante seja entregue pela memória, a palavra referenciada seja encaminhada ao processador. Feito o registro, o processador pode seguir executando outras instruções. A estrutura que mantém um registro de bloco faltante chama-se *Miss Status Holding Register* (MSHR) [6]. Todos os modelos foram simulados com 4 MSHRs, o que permite ao processador prosseguir mesmo que haja até 4 faltas concomitantes.

Cache de Vítimas Uma *cache de vítimas* (CV) é uma cache totalmente associativa com 1-4 blocos e que mantém os blocos expurgados da L1 porque foram substituídos por outros, recém-buscados [5]. Uma CV provê associatividade a uma cache L1 com mapeamento direto; em geral, 4 blocos são suficientes para eliminar uma fração considerável das faltas por conflitos de mapeamento.

Cache de Controle de Poluição Para evitar que blocos que são trazidos para a cache e que são em seguida descartados expurguem blocos que são usados ativamente, uma *cache de controle de poluição* (CCP, *pollution control cache* [9], pode ser interposta entre a memória e a cache L1. Os blocos faltantes são inicialmente carregados na CCP, e de lá a palavra faltante é encaminhada ao processador; se o bloco é referenciado uma segunda vez antes de ser expurgado da CCP, então ele é promovido para a L1. Um acerto na CCP custa dois ciclos: um para detectar a falta na L1 e o segundo para o acesso à CCP. O projeto descrito em [9] associa uma CV à L1, e outra CV à CCP; a versão simulada neste trabalho difere daquele projeto porque há somente uma CV, ligada à L1. A Figura 1 mostra um diagrama de blocos da CCP como simulada.

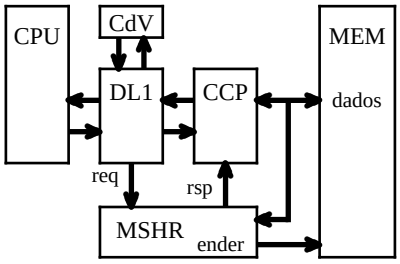

Figura 1. Cache de Controle de Poluição.

Cache de Vítimas e Controle de Poluição É possível simplificar o projeto da CCP e ao mesmo tempo obter melhor desempenho da hierarquia de memória. Uma *cache de vítimas e controle de poluição* (CVCP, *pollution control victim cache*) [4] é uma cache totalmente associativa interposta entre a memória e a L1, que opera como um misto de CCP e CV. Blocos referenciados pela primeira vez são trazidos para a CVCP, e na segunda referência são promovidos para a L1; o bloco expurgado da L1 é carregado na CVCP, no local em que estava o bloco recém-promovido. Um acerto na CVCP custa dois ciclos. A implementação da CVCP é menos complexa que a da CCP, e nos testes descritos em [4], seu desempenho é melhor. A Figura 2 mostra um diagrama de blocos da CVCP.

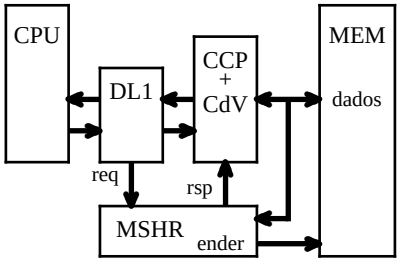

Figura 2. Cache de Vítimas e Controle de Poluição.

Projetos de hierarquias de memória Os quatro *buffers* podem ser combinados para melhorar o desempenho da hierarquia de memória. Efetuamos a comparação de desempenho, área e consumo de energia de quatro projetos distintos para o primeiro nível de uma hierarquia de memória. Nosso foco é em caches de dados para sistemas embarcados e portanto pesquisamos caches primárias pequenas e simples. Os quatro projetos que investigamos são: ⟨i⟩ cache primária com mapeamento direto ($L1_{MD}$); ⟨ii⟩ cache primária com associatividade binária (*2-way set associative*, $L1_{2w}$); ⟨iii⟩ cache primária com MD e cache com controle de poluição, com 4, 8, 16 e 32 blocos (CCP_4 a CCP_{32}); ⟨iv⟩ cache primária com MD e cache de vítimas e controle de poluição, com 4, 8, 16 e 32 blocos ($CVCP_4$ a $CVCP_{32}$). Para limitar o espaço de projeto investigamos somente o primeiro nível da hierarquia – os parâmetros usados para o segundo nível são representativos de sistemas de memória relativamente ambiciosos, ou de sistemas com uma cache de segundo nível.

3. Ambiente de Simulação

As simulações foram executadas sobre versões modificadas do SimpleScalar *(sim-outorder)* [3, 2], descritas em [4]. O processador modelado é um processador superescalar com largura 2 (busca, decodifica, executa e completa duas instruções por ciclo). O *buffer* de reordenação tem 16 elementos e a fila de memória *(load-store queue)* tem 8 elementos, e há uma unidade de multiplicação-divisão de ponto flutuante. Estes parâmetros representam um bom compromisso entre desempenho, complexidade de projeto e custo do processador.

A cache primária de dados (L1) foi simulada com capacidades de 1, 2, 4, 8, 16 e 32 Kbytes, com blocos de tamanhos 32 bytes, e com mapeamento direto e associatividade binária. A latência de acerto *(hit)* é 1 ciclo; a penalidade por falta é de 18 ciclos para a primeira palavra, mais 2 ciclos por palavra adicional; o barramento tem 8 bytes de largura – a carga de um bloco na cache custa 18+3·2=24 ciclos. Todos os dados foram coletados com uma cache de instruções (L1i) com 32 Kbytes e associatividade binária para minimizar sua interferência nos resultados.

O desempenho dos projetos foi avaliado com execução direta no SimpleScalar dos programas da suíte *CommBench* [11]. Estes são programas relativamente pequenos e são núcleos de aplicações típicas de processadores de rede. O conjunto contém programas de *processamento de cabeçalho*, que examinam e processam o cabeçalho de mensagens, e de *processamento de conteúdo*, que acessam e/ou modificam os conteúdos de um fluxo de pacotes. Os programas de processamento de cabeçalho empregados nas medições são RTR (*Radix-Tree Routing*), FRAG (fragmentação de pacotes IP), e DRR (escalona-

mento *Deficit Round Robin*). Os programas de processamento de conteúdo empregados são CAST (cifração *CAST-128*), ZIP (compressão Lempel-Ziv), REED (codificação Reed-Solomon) e JPEG (algoritmo de compressão de imagens). Todas as simulações foram executadas com os dados distribuídos com a suíte CommBench. A Tabela 1 mostra as contagens de instruções dinâmicas, de *loads*, *stores*, e a fração de todas as instruções que são referências à memória.

Tabela 1. Caracterização dos programas [$\times 10^6$]

programa	instr	*loads*	*stores*	(ld+st)/ins
`cast_dec`	141,45	29,69	13,01	0,302
`cast_enc`	141,57	29,70	13,01	0,302
`drr`	22,31	11,03	2,25	0,596
`frag`	179,09	31,03	21,03	0,291
`jpeg_dec`	223,68	43,41	25,26	0,307
`jpeg_enc`	317,31	55,78	28,17	0,265
`reed_dec`	1309,74	164,30	72,31	0,181
`reed_enc`	719,91	117,46	39,07	0,217
`rtr`	906,86	280,62	60,77	0,376
`zip_dec`	41,39	9,07	2,00	0,267
`zip_enc`	248,03	51,90	16,38	0,275

4 Desempenho *vs* Capacidade

O desempenho dos quatro projetos de hierarquia de memória é expresso em *ciclos por instrução* (CPI). Em geral, CPI é inversamente proporcional à capacidade da cache e este comportamento característico será usado adiante. Os valores apresentados são as médias para os 11 programas da suíte CommBench.

A Figura 3 contém os medidas de desempenho, expressas como *CPI* $\times$ *capacidade da L1* para os quatro projetos. Como esperado, o projeto mais simples é o de pior desempenho: a $\mathrm{L1}_{\mathrm{MD}}$ tem o pior desempenho em todas as capacidades. A cache com mapeamento associativo $\mathrm{L1}_{\mathrm{2w}}$ tem desempenho um pouco melhor que a $\mathrm{L1}_{\mathrm{MD}}$, mas pior que os outros dois projetos. O desempenho das CCP_4 e CCP_{32} é similar ao da CVCP_4. A CVCP_{32} tem o melhor desempenho, e este é uniforme para todos os tamanhos de L1.

A Figura 4 mostra as curvas de taxa de faltas *na cache primária L1* para os quatro projetos. As curvas para os projetos $\mathrm{L1}_{\mathrm{MD}}$ e $\mathrm{L1}_{\mathrm{2w}}$ representam a taxa de falta medida no processador, enquanto que as curvas para as CCP e CVCP representam as taxas de faltas na interface da L1 e portanto são *maiores* do que as medidas no processador. Nestes dois projetos mais sofisticados, a taxa de faltas na L1 é pior porque o *buffer* filtra uma fração considerável das faltas – uma falta na L1 freqüentemente é um acerto no *buffer*.

A taxa de faltas medida na L1 pode ser elevada, mesmo quando a taxa de acertos no primeiro nível da hierarquia é

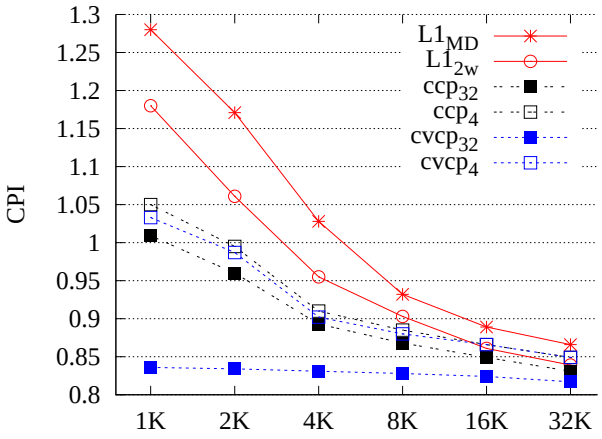

Figura 3. CPI $\times$ capacidade da L1.

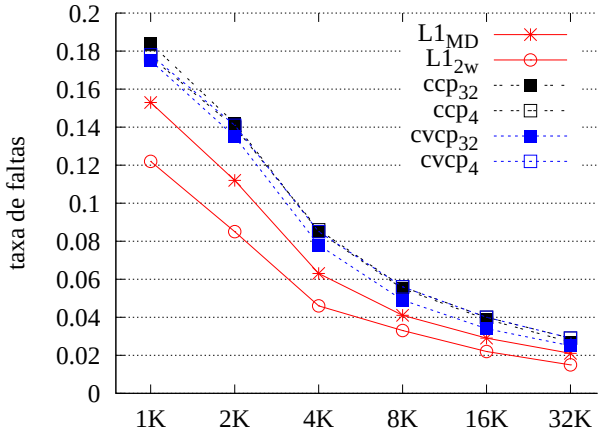

Figura 4. Taxa de faltas na L1 $\times$ capacidade da L1.

alta porque um bloco faltante na L1 pode ser encontrado em qualquer dos *buffers*. A Figura 5 mostra a taxa de acertos em todo o primeiro nível da hierarquia de memória – L1 e *buffers* – que é a taxa de acertos efetiva para o processador. Para os programas e conjuntos de dados do CommBench, a CVCP_{32} sofre quase que somente faltas compulsórias, acomodando os conjuntos de dados dos programas. Uma cache L1 de 32 Kbytes contém, quase que completamente, os conjuntos de trabalho dos programas, como indicam as medidas, especialmente aquelas para a $\mathrm{L1}_{\mathrm{MD}}$.

5. Área & Energia

Para efetuar as comparações de desempenho das quatro organizações da hierarquia de memória, as caches foram modeladas com CACTI-3.2 [10, 7], para estimar a área e o dispêndio de energia de cada uma delas. As quatro

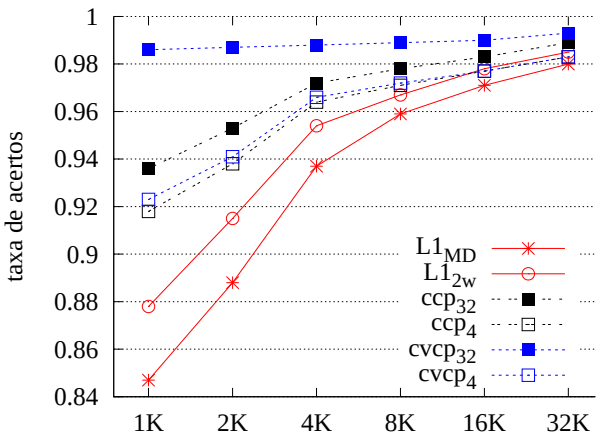

Figura 5. Taxa de acertos global × capac. da L1.

organizações são: ⟨i⟩ cache L1 com mapeamento direto (MD); ⟨ii⟩ cache L1 com associatividade binária *(2-way)*; ⟨iii⟩ L1 com MD e CCP; e ⟨iv⟩ L1 com MD e CVCP.

5.1 Modelagem com CACTI

A tecnologia das memórias empregada na versão 3.2 do CACTI é 180 nm, que é a versão mais recente que permite a simulação de memórias com capacidade tão pequena quanto aquelas estudadas neste trabalho.

MSHR A implementação dos MSHRs envolve mais lógica de controle do que circuitos de armazenagem e portanto não há um modelo CACTI para este bloco. Os processadores, em todos as simulações, empregam 4 MSHRs com a mesma largura do bloco da cache e portanto, a área e o dispêndio de energia destas estruturas podem ser computados como se fizessem parte do processador e não da hierarquia de memória.

Cache Primária A cache primária (L1) foi modelada de quatro formas distintas: ⟨i⟩ mapeamento direto (MD) e dois pares de portas de leitura-escrita (RD-WR), um par ligado ao processador e MSHRs, e o outro par ligado à memória; ⟨ii⟩ associatividade binária (2w) e dois pares de portas RD-WR; ⟨iii⟩ a L1 usada com a CCP tem mapeamento direto e é ligada à CV, ao processador e à memória – a L1 tem três pares de portas de leitura-escrita (Fig. 1); ⟨iv⟩ o modelo da L1 usado com a CVCP é com mapeamento direto e com dois pares de portas, um par ligado ao processador e o outro à CVCP (Fig. 2).

Cache de Vítimas O modelo da Cache de Controle de Poluição emprega uma CV ligada à L1. A CV foi simulada, e modelada, com 4 blocos, associatividade total e blocos de mesmo tamanho que na L1. Esta memória tem uma porta de leitura-escrita (RD-WR).

Cache de Controle de Poluição A CCP é uma memória totalmente associativa e seus blocos tem a mesma largura que os blocos da L1. A CCP é modelada com dois pares de portas de leitura-escrita, um par ligado à L1 e o outro à memória. Neste modelo a CV poderia estar ligada, através de um *crossbar* 3×3, à L1 e à CCP, ao invés conectar-se à terceira porta da L1. Como não dispomos de uma implementação detalhada destes dois projetos, não é possível fazermos uma comparação entre eles. Nossa conjectura é de que, para as caches L1 de pequena capacidade (até 4 Kbytes), a terceira porta da L1 ocuparia aproximadamente a mesma área que o *crossbar*; para as capacidades maiores, a terceira porta ocupa uma área maior do que o *crossbar*.

O projeto original [9] emprega duas CVs, com a segunda ligada à CCP; por isso, a CCP_{orig} também necessita de três portas: uma ligada à L1, uma para a CV, e a terceira ligada à memória. No modelo simulado somente a L1 tem três portas.

Cache de Vítimas e Controle de Poluição A CVCP é uma memória totalmente associativa e seus blocos tem a mesma largura da L1. A CVCP é modelada com dois pares de portas de leitura-escrita, um par ligado à L1 e o outro à memória. O controlador da CVCP é mais complexo que o da CCP porque a função de "cache de vítimas" está incorporada à CVCP. Esta complexidade adicional não foi considerada na modelagem com CACTI.

Interconexões Os modelos das memórias, e as estimativas obtidas com CACTI, não levam em conta as interconexões entre os vários componentes porque os custos associados às ligações com o processador/MSHRs e com a memória são os mesmos. Os projetos com CCPs e CVCPs não consideram as interligações entre a L1 e os *buffers* especializados – esta ligação representa um adicional fixo, seja em área, seja em energia/referência. No caso da CCP, se um *crossbar* for usado, ao invés de uma terceira porta na L1, as curvas mostradas adiante para área e energia desta estrutura seriam similares àquelas para a CVCP, apenas acrescidas de uma fração fixa associada à CV.

5.2 Área

Efetuamos uma comparação da área ocupada pelas diversas combinações de capacidade, associatividade e *buffers*. O lado esquerdo da Tabela 2 lista as áreas ocupadas pelos *buffers* para as capacidades de 4, 8, 16 e 32 blocos; a Tabela 3 mostra as áreas dos quatro projetos, sendo que para a CCP e CVCP são mostradas as áreas ocupadas com *buffers* de 4 e de 32 blocos. A Figura 6 mostra as áreas para as combinações de L1 simples, com mapeamento direto e associatividade binária; CCP e CVCP com capacidade de 4 e 32 blocos. A L1 foi simulada com capacidade de 1 Kbytes a 32 Kbytes.

Tabela 2. Área e energia vs capacidade [blocos]

	área [mm^2]		energia/ref [nJ]	
blocos	CV	CCP, CVCP	CV	CCP, CVCP
4	0,22	0,70	0,21	0,45
8	–	0,76	–	0,47
16	–	0,89	–	0,50
32	–	1,12	–	0,58

Tabela 3. Área [mm^2] vs capacidade [Kbytes]

capac	$L1_{MD}$	$L1_{2w}$	CCP_4	CCP_{32}	$CVCP_4$	$CVCP_{32}$
1	0,80	1,18	2,99	3,42	1,45	1,86
2	1,07	1,38	3,93	4,36	1,77	2,19
4	1,58	1,86	4,92	5,35	2,28	2,69
8	2,67	2,87	7,66	8,09	3,37	3,78
16	5,02	5,02	13,81	14,23	5,72	6,13
32	8,93	9,26	25,02	25,44	9,63	10,04

Considerando-se o modelo base como sendo a L1 com mapeamento direto ($L1_{MD}$), a L1 com associatividade binária ($L1_{2w,1K}$) ocupa 48% mais área, mas a diferença diminui com a capacidade: a $L1_{2w,32K}$ é 3% maior que a $L1_{MD,32K}$. Para uma dada combinação capacidade, tamanho de bloco e associatividade, CACTI escolhe a melhor geometria para aquele conjunto de parâmetros e por isso a diferença entre as áreas não se reduz exatamente com o inverso do quadrado da capacidade da L1.

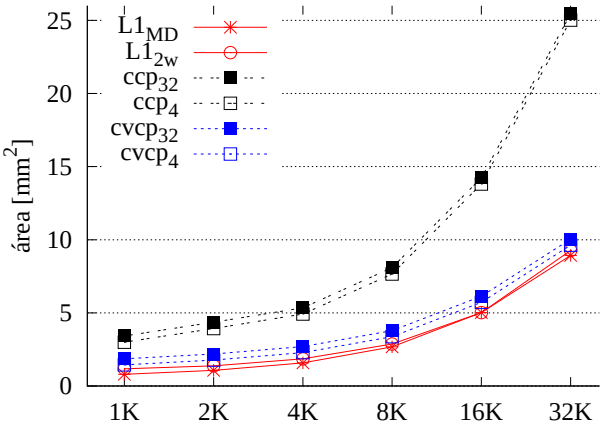

Figura 6. Área vs capacidade da L1.

Na comparação da CCP com a $L1_{MD}$, a $CCP_{4,1K}$ ($CCP_{32,1K}$) é 3,8 (4,3) vezes maior que a $L1_{MD,1K}$. A diferença cai para 2,8 (2,9) vezes para 32 Kbytes. A área da CV corresponde a uma parcela decrescente da área da CCP quando a capacidade desta aumenta; a parte mais significativa da diferença entre as áreas é devida à terceira porta da L1 usada com a CCP.

Para todas as capacidades simuladas, a CVCP ocupa mais espaço que a $L1_{MD}$, e a diferença é de um fator de 1,8 (2,3) vezes para a $CVCP_{4,1K}$ ($CVCP_{32,1K}$), decrescendo para 1,08 (1,12) vezes para 32 Kbytes. Para um determinado tamanho t da CVCP, esta estrutura acrescenta uma parcela fixa à área do conjunto $CVCP_t$+L1, que é proporcionalmente menor para as L1 de maior capacidade.

Desempenho ***vs*** **Área** A *área* ocupada pelo primeiro nível da hierarquia de memória cresce com a capacidade da L1, enquanto que a *taxa de faltas* decresce. O menor valor do produto destas duas grandezas é obtido num projeto 'ótimo', para o qual a 'derivada' do produto, com relação à capacidade, se iguala a zero. Por 'derivada' entenda-se uma aproximação na qual as linhas de tendência equivalem ao comportamento se as capacidades variassem continuamente. A Figura 7 mostra os valores dos produtos de área e taxa de faltas.

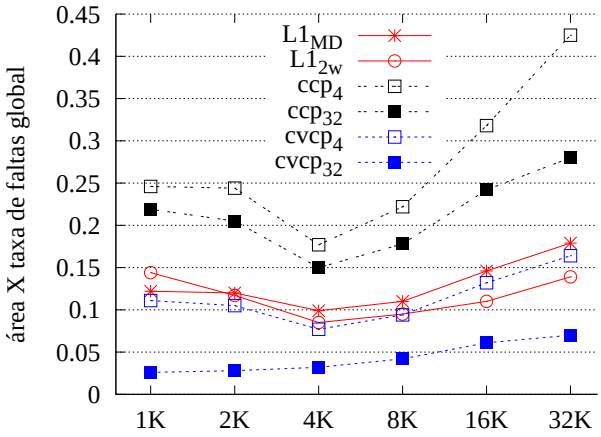

Figura 7. Área $\times$ taxa de faltas global.

A Figura 3 mostra que, para os projetos com caches de controle de poluição (exceto para a $CVCP_{32}$), o ganho de desempenho (CPI) é relativamente pequeno ao se aumentar a capacidade da L1 para além de 4-8 Kbytes, e isso é confirmado pelo produto área $\times$ taxa de faltas global, mostrado na Figura 7. Exceto para a $CVCP_{32}$, o melhor produto ocorre para a L1 com 4 Kbytes. Do ponto de vista de *taxa de faltas* $\times$ *área*, os melhores projetos são $L1_{MD,4K}$, $L1_{2w,4K}$ e $CVCP_{4,4K}$. A $CVCP_{32}$ sempre obtém melhor desempenho que os outros projetos porque, para todas as capacidades da $L1_{MD}$, a CVCP mantém o subconjunto mais ativo dos dados junto ao processador, conforme discutido na Seção 6.

5.3 Energia

Efetuamos a comparação do dispêndio de energia por referência para os quatro projetos. A Tabela 2 mostra o dispêndio de energia por referência para os *buffers*, com capacidades de 4, 8, 16 e 32 blocos; a Tabela 4 mostra o dispêndio de energia para os quatro projetos. A Figura 8 mostra as curvas de energia por referência, nas mesmas condições da Seção 5.2.

Tabela 4. Energia/ref [nJ] vs capacidade [Kbytes]

capac	$L1_{MD}$	$L1_{2w}$	CCP_4	CCP_{32}	$CVCP_4$	$CVCP_{32}$
1	0,61	0,95	1,69	1,82	1,05	1,19
2	0,64	0,98	1,78	1,91	1,09	1,22
4	0,72	1,04	2,01	2,14	1,17	1,30
8	0,88	1,20	2,44	2,57	1,32	1,45
16	1,09	1,42	3,03	3,09	1,54	1,67
32	1,48	1,80	4,17	4,23	1,92	2,05

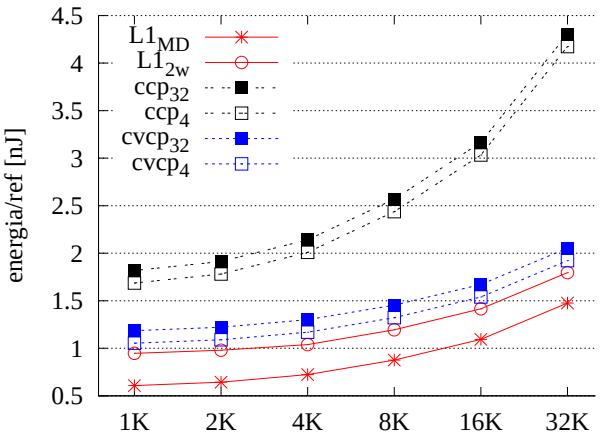

Figura 8. Energia/referência vs capacidade da L1.

O dispêndio de energia por referência da $L1_{2w}$ é de 1,56 vezes o da $L1_{MD}$, para caches de 1 Kbytes, e de 1,22 vezes para 32 Kbytes. Para a cache de controle de poluição, com todos os tamanhos de L1 simulados, uma CCP_4 consome de 2,77 a 2,83 vezes mais que a $L1_{MD}$. Para a CCP_{32}, as diferenças estão na faixa de 2,89 a 2,99 vezes. Para a CVCP, as variações são decrescentes com a capacidade da L1: a $CVCP_4$ consome 1,7 (1,3) vezes mais que a $L1_{MD,1K}$ (32 K), enquanto que para a $CVCP_{32}$ as razões são 1,95 e 1,4 vezes para $L1_{MD,1K}$ e $L1_{MD,32K}$, respectivamente.

Desempenho *vs* Energia O produto *taxa de faltas* × *energia/referência* é mostrado na Figura 9, na qual é perceptível uma inflexão para a capacidade da L1 de 4 Kbytes. Estes pontos não são os "mínimos da curva", mas indicam que acréscimos na capacidade da L1 produzem ganhos decrescentes. As CCPs tem valores piores que dos projetos simples; a $L1_{MD}$ tem desempenho mais próximo à $CVCP_4$ do que a $L1_{2w}$ porque seus circuitos são mais simples e portanto consomem menos energia que a $L1_{2w}$. Para capacidades de 4 Kbytes e além, há pouca diferença entre $L1_{MD}$, $L1_{2w}$ e $CVCP_4$. A $CVCP_{32}$ tem o melhor desempenho, por uma ampla margem.

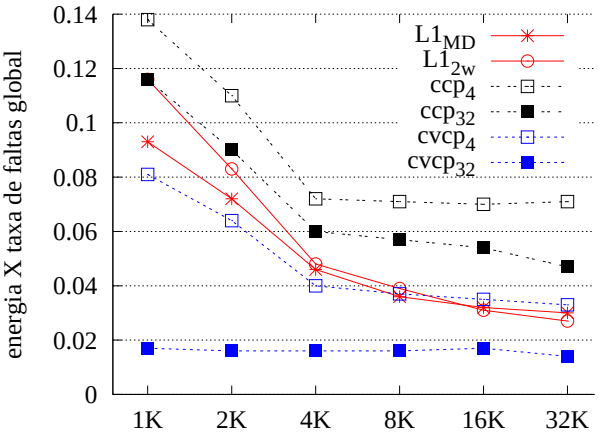

Figura 9. Energia/ref. × taxa de faltas global.

A Figura 10 mostra o produto *CPI* × *energia/referência*. Com esta métrica, os desempenhos de $L1_{2w}$, $CVCP_4$ e $CVCP_{32}$ são similares. A elevada taxa de faltas inerente à $L1_{MD}$ se traduz em elevado CPI – apesar do menor consumo de energia, este projeto é aquele com o pior desempenho global, cfe. Fig. 3. As CCPs tem desempenho (CPI) melhor que o dos projetos simples mas seu dispêndio de energia é de tal ordem que o seu produto *energia*×*CPI* é praticamente o dobro dos outros projetos.

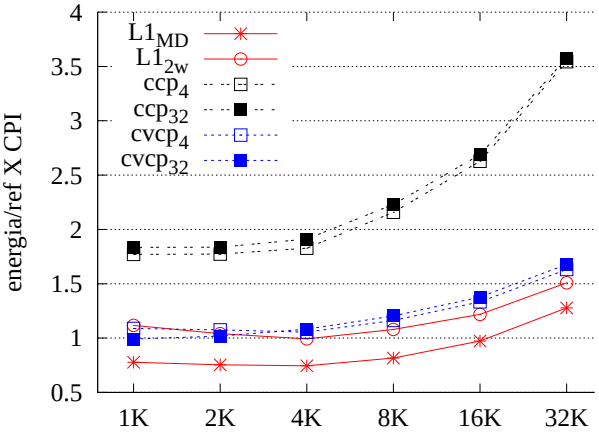

Figura 10. Energia/ref. × CPI.

6. Padrões de Referências à Memória

A *popularidade* de uma palavra de memória é definida como a razão entre o número de referências àquela palavra, e o número de endereços distintos referenciados pelo programa. Efetuamos a medição da popularidade de dados dos programas do CommBench: para 6 dos 11 programas, menos de 5% dos endereços satisfazem a mais de 70% das referências. Por exemplo, para `jpeg`, $\leq$1.2% de todos os endereços (aprox. 15.000 palavras) satisfazem a 90% de todas as referências a dados. `zip_enc` apresenta um comportamento distinto: as 12.800 palavras mais populares (17% dos endereços) satisfazem a menos de 60% das referências.

A Figura 11 mostra as curvas de popularidade para as duas versões de `jpeg` e de `zip`. A abcissa representa os endereços referenciados, do mais popular para o menos, e a ordenada representa o acumulado de todas as referências do programa. As curvas para `jpeg` mostram claramente a concentração de referências num subconjunto pequeno do espaço de endereçamento – há uma inflexão pronunciada próximo aos 90% das referências. Consideramos que o "limite da popularidade" é o endereço no qual a derivada da contagem cumulativa de referências é tal que cada novo endereço inserido na região de popularidade adiciona uma quantidade desprezível de referências†.

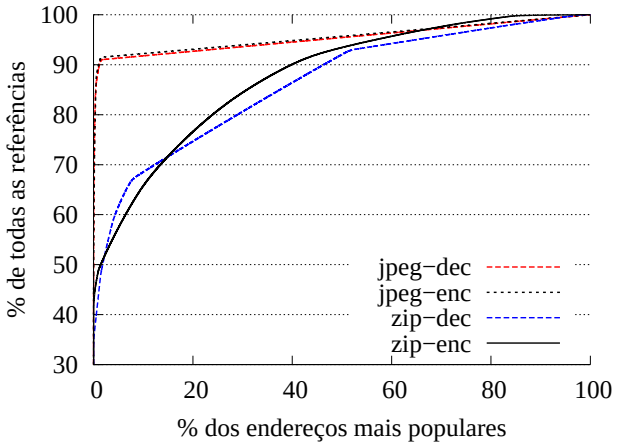

Figura 11. Popularidade de `jpeg` e `zip`.

A curva para `zip_dec` contém dois pontos de inflexão: o primeiro corresponde a 67% de todas as referências, e o segundo a 93% das referências. A curva para `zip_enc` não contém pontos de inflexão claramente visíveis mas o limite da popularidade é próximo aos 57% das referências. Os endereços referenciados por `zip` podem ser divididos em três faixas: *populares*, *semi-populares* e *impopulares*. Para se obter o melhor desempenho da hierarquia de memória, o conteúdo dos endereços populares deve ser mantido na L1, o dos semi-populares num dos *buffers* (CCP ou CVCP), e as palavras impopulares alocadas a endereços que não são buscados pela(s) cache(s) do primeiro nível.

A Tabela 5 mostra o número n de endereços distintos referenciados pelo programa, o número de endereços mais populares p, a fração $\pi = p/n$ do número de endereços mais populares sobre todos os endereços distintos, e a fração de todas as referências que são para os endereços populares. As referências são a palavras alinhadas de 32 bits. Para a maioria dos programas, $p < 512$ palavras, que é a capacidade do projeto CVCP_{32}+$\text{L1}_{\text{MD,1K}}$: 32 blocos na CVCP mais 32 blocos na L1_{MD}, cada bloco com 8 palavras. O limite da popularidade† é o ponto além do qual cada novo endereço corresponde a um acréscimo $< 1/n$.

Tabela 5. Popularidade

programa	n	p	π	refs(p)
`cast_dec`	1.974	93	0,047	0,71
`cast_enc`	1.973	75	0,038	0,70
`drr`	5.868	300	0,051	0,90
`frag`	5.591	114	0,020	0,98
`jpeg_dec`	1.002.221	15364	0,015	0,89
`jpeg_enc`	1.000.488	14835	0,015	0,90
`reed_dec`	1.451	404	0,278	0,66
`reed_enc`	1.260	297	0,236	0,72
`rtr`	1.292.378	29565	0,023	0,86
`zip_dec`	19.510	1535	0,079	0,67
`zip_enc`	76.075	12806	0,168	0,57

A Figura 5 (Seção 4) mostra a média para todos os programas da taxa de acertos global no primeiro nível da hierarquia de memória. As taxas de acerto para `jpeg` são muito próximas à media e o bom desempenho deste programa é devido à sua excelente localidade. Mesmo com $p \approx 15.000$, os *buffers* evitam que endereços impopulares poluam a L1_{MD}. Como visto na Seção 5.2, o conjunto de trabalho de `jpeg` é acomodado numa L1 de 4 Kbytes. Nas medidas para `jpeg_enc`, para L1 com capacidade maior ou igual a 4 Kbytes, a taxa de acertos fica próxima do seu máximo. Para a $\text{L1}_{\text{MD,1K}}$, o ganho de desempenho (CPI) obtido com o acréscimo de um CVCP_{32} é de 0,31, enquanto que com a $\text{L1}_{\text{MD,4K}}$, o ganho é de 0,10 – estes valores são similares às médias da Figura 3.

Com todos os tamanhos de L1 simulados, para `zip_enc` $p \approx 13.000 > |\text{L1}_{\text{MD}}|$ e o reaproveitamento das palavras trazidas para a cache é pequeno, portanto a taxa de acertos global não atinge um valor de saturação, como é o caso dos outros programas (cfe. Fig 5). A Figura 12 mostra as taxas de acerto globais dos quatro projetos, na execução de `zip_enc`. Os *buffers* com 32 blocos melhoram significativamente a taxa de acertos, mantendo parte do conjunto de trabalho na L1 e protegendo-a da poluição causada pelas referências aos endereços impopulares. O ganho no CPI, em comparação com uma $\text{L1}_{\text{MD,1K}}$, para a

combinação $L1_{MD,1K}$+$CVCP_{32}$ é de 0,25, enquanto que para L1 de 4 Kbytes o ganho é de 0,10. A $CVCP_{32}$ traz ganho de desempenho de 0,16 para a $L1_{MD,32K}$.

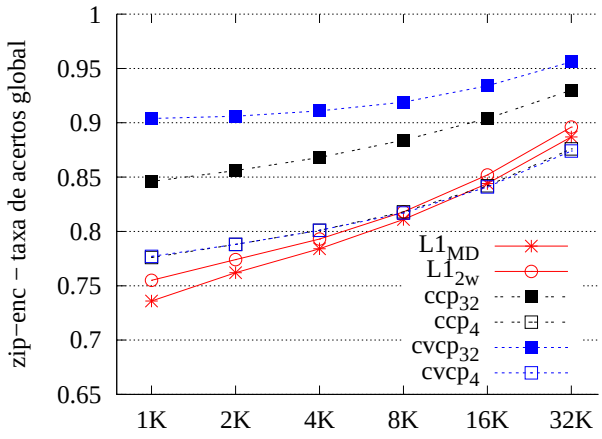

Figura 12. Taxa de acertos global de `zip_enc`.

Para programas com um padrão de referências similares ao de `zip`, a adição do CVCP é vantajosa porque o ganho de desempenho (16%) é similar ao acréscimo em área (12%) para a combinação $L1_{MD,32K}$+$CVCP_{32}$. Quanto à energia, a situação não é tão equilibrada: ganho de 16% em desempenho para 38% de acréscimo no consumo de energia. Neste caso, o ganho de desempenho deve ser pesado contra o aumento de consumo e o projetista deve decidir se a adição é compensatória. Para programas "bem comportados", CVCPs com 4 a 16 blocos podem ser vantajosos porque o acréscimo em área e energia é evidentemente menor que o de 32 blocos, enquanto que o ganho de desempenho não se reduz na mesma proporção que o número de blocos. [4] contém a avaliação de desempenho de CVCPs com capacidades de 1 a 32 blocos.

7. Conclusão

Apresentamos uma comparação de desempenho, área e dispêndio de energia de quatro projetos para o primeiro nível de uma hierarquia de memória para sistemas embarcados: uma cache com mapeamento direto ($L1_{MD}$), uma cache com associatividade binária ($L1_{2w}$), uma combinação de $L1_{MD}$ com uma cache de controle de poluição (CCP), e uma combinação de $L1_{MD}$ com uma cache de vítimas com controle de poluição (CVCP). A maioria dos conjuntos de trabalho dos programas avaliados tem alta concentração de referências nos endereços populares (boa localidade) e com tamanhos da ordem de 4-8 Kbytes – caches destas capacidades produzem ganhos de desempenho que se aproximam do máximo que se pode atingir com recursos razoáveis. Para estas capacidades, a *cache de vítimas com controle de poluição* tem bom desempenho, com acréscimos de área e dispêndio de energia que são da ordem do dobro do ganho em CPI. Para caches maiores, o ganho de desempenho é similar ao acréscimo em área e energia.

Os experimentos descritos aqui são baseados em simulações de programas que executam isoladamente. Em sistemas reais, o processador executa mais de um aplicativo e a cache é compartilhada entre os aplicativos e seus tratadores de interrupção e *drivers* de dispositivos. Estes dois últimos executam por curtos intervalos e referenciam poucos dados, mas a poluição que causam na cache desloca dados usados ativamente e reduz o desempenho dos aplicativos. Em trabalho futuro pretendemos avaliar a efetividade da CVCP em manter aquelas referências fora da cache. Também pretendemos avaliar o uso de *stream buffers* que fazem busca antecipada somente de endereços impopulares. Nossa expectativa é de que tais referências possam ser satisfeitas pelo *stream buffer* com baixa latência e sem causar poluição na cache.

Referências

[1] In *WSCAD-SSC'09: X Workshop em Sistemas Computacionais de Alto Desempenho*, pages 1–8, Out 2009.

[2] T. Austin, E. Larson, and D. Ernst. SimpleScalar: An infrastructure for computer system modeling. *Computer*, 35(2):59–67, 2002.

[3] D Burger and T M Austin. The SimpleScalar tool set, version 2.0. Technical report, University of Wisconsin-Madison and SimpleScalar LLC, 1997.

[4] G. C. Heck and R. A. Hexsel. The performance of pollution control victim cache for embedded systems. In *SBCCI'08: 21st Symp on Integrated Circuits and System Design*, pages 46–51, 2008.

[5] N. P. Jouppi. Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers. In *ISCA'90: 17th Intl Symp on Computer Architecture*, pages 364–373, 1990.

[6] D. Kroft. Lockup-free instruction fetch/prefetch cache organization. In *ISCA'81: 8th Annual Symp on Computer Architecture*, pages 81–87, 1981.

[7] P. Shivakumar and N. P. Jouppi. CACTI 3.0: An integrated cache timing, power, and area model. Technical Report 2001/2, DEC-WRL, 2001.

[8] SimpleScalar LLC, Mar 2007. `http://www.simplescalar.com/`.

[9] S. J. Walsh and J. A. Board. Pollution control caching. In *ICCD'95: Intl Conf on Computer Design*, page 300, 1995.

[10] S. J. E. Wilton and N. P. Jouppi. CACTI: An enhanced cache access and cycle time model. *IEEE Journal of Solid-State Circuits*, 31(5):677–688, Mai 1996.

[11] T. Wolf and M. A. Franklin. CommBench – a telecommunications benchmark for network processors. In *ISPASS'00: IEEE Intl Symp on Performance Analysis of Systems and Software*, pages 154–162, Abr 2000.