O Consumo de Energia da Arquitetura DTSVLIW

Felipe Thomaz Pedroni[1], Fernando Líbio L. Almeida e Alberto F. De Souza
Departamento de Informática, Universidade Federal do Espírito Santo
felthomaz@hotmail.com
{flibio, alberto}@inf.ufes.br

Resumo

Neste trabalho apresentamos um estudo sobre o consumo de energia da arquitetura DTSVLIW. Nós implementamos uma versão do nosso simulador DTSVLIW capaz de medir tanto o consumo de energia dinâmico quanto o estático. Comparamos estes resultados com os de simuladores da arquitetura do processador Alpha 21264, como o Wattch e o Hotleackage. Os experimentos mostraram que a arquitetura DTSVLIW consome consideravelmente menos energia que o processador Alpha 21264.

1. Introdução

Até recentemente, consumo de energia e dissipação de calor eram questões relevantes apenas para os arquitetos de processadores de computadores portáteis. Porém, prevê-se que o consumo de energia e a dissipação de calor tornar-se-ão fatores limitantes da performance de qualquer microprocessador no futuro [10].

O número máximo possível de transistores em um CI (próprio para fabricação de processadores) já passa de 100 milhões e em breve será possível implementar processadores com até um bilhão de transistores [5]. Entretanto, o chaveamento em alta freqüência de um grande número de transistores resulta em elevado consumo de corrente e dissipação de grande quantidade de calor. Processadores de alto desempenho atuais precisam de ventilação forçada ou até mesmo de mecanismos mais elaborados de resfriamento. Assim, existe hoje uma crescente demanda pelo desenvolvimento de processadores de alto desempenho e de baixo consumo.

Neste trabalho examinamos o consumo de energia dinâmico e estático da arquitetura *Dynamically Trace Scheduled Very Long Instruction Word* (DTSVLIW) [6, 7], a qual ainda não foi implementada fisicamente. A fim de permitir uma melhor compreensão dos resultados no contexto dos processadores existentes atualmente, realizamos uma análise comparativa experimental do consumo de energia do processador Alpha 21264 [2] com o consumo de uma implementação hipotética da arquitetura DTSVLIW equivalente, no seu hardware, a este processador. Para a realização dos experimentos, implementamos uma versão do nosso simulador DTSVLIW com medidores de consumo de energia. Para a avaliação do consumo de energia dinâmico do processador Alpha 21264 utilizamos o simulador Wattch [4], enquanto que, para a avaliação do consumo estático, utilizamos o simulador Hotleakage [17], ambos disponíveis publicamente. Tanto nosso simulador DTSVLIW quanto os simuladores Wattch e Hotleakage são baseados no simulador SimpleScalar-3.0 (www.simplescalar.com), o que permite uma comparação apropriada entre os resultados obtidos.

2. A Arquitetura DTSVLIW

A Figura 1 mostra um diagrama de blocos da arquitetura DTSVLIW, a qual possui dois modos de execução: um escalar e um VLIW. Sempre que um trecho de código é encontrado pela primeira vez, ele é executado no modo escalar pelo Processador Primário, um processador *pipelined* simples capaz de executar no máximo uma instrução por ciclo – instruções que necessitam de mais de um ciclo impedem que instruções subseqüentes avancem para o estágio de execução do *pipeline* do Processador Primário (Figura 1). No modo escalar, o código é trazido da Cache de Instruções pelo Processador Primário e, quando suas instruções são enviadas para o estágio de execução, elas são também enviadas para a Unidade de Escalonamento (linha tracejada na Figura 1), que as escalona, dentro da Lista de Escalonamento, em blocos de instruções VLIW. Estes blocos são salvos na Cache VLIW, sendo que o endereço de cada bloco é igual ao da primeira instrução do trecho de código do qual o bloco se originou. Se um mesmo trecho de código precisa ser executado novamente, ele pode estar presente na Cache VLIW. Isto é detectado pela Unidade de Busca (Figura 1) que, neste caso, traz instruções VLIW da Cache VLIW e as envia para execução na Máquina VLIW, que as executa em modo VLIW. No diagrama de blocos da Figura 1, instruções com latência superior a um ciclo permanecem mais de um ciclo no estágio de execução VLIW, que pode ou não ser totalmente pipelined, dependendo da instrução.

[1] Bolsista de Iniciação Científica – PIBIC/UFES-CNPq.

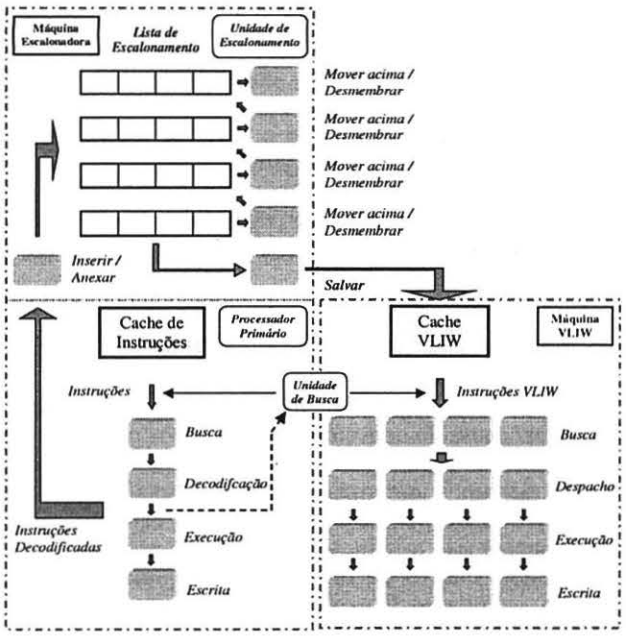

Figura 1: A Arquitetura DTSVLIW

A arquitetura DTSVLIW é capaz de executar código seqüencial comum em modo VLIW, sendo o grau de paralelismo de sua Máquina VLIW dependente apenas da tecnologia usada na sua implementação. Em um processador DTSVLIW, a Máquina Escalonadora provê compatibilidade de código objeto e a Máquina VLIW provê desempenho VLIW e simplicidade.

A arquitetura DTSVLIW é uma alternativa à arquitetura Super Escalar e pode ser dimensionada para disponibilizar paralelismo de hardware igual ou superior àquele que pode ser extraído pela arquitetura Super Escalar. Contudo, com a crescente importância do consumo de energia na decisão sobre qual arquitetura utilizar, a escolha da arquitetura DTSVLIW no lugar da arquitetura Super Escalar só ocorrerá se a primeira também oferecer vantagens no quesito consumo de energia. No entanto, uma comparação quantitativa entre o consumo de energia destas arquiteturas ainda não havia sido feita. Assim, a principal motivação para a realização deste trabalho foi investigar o potencial da arquitetura DTSVLIW como uma possível alternativa à arquitetura Super Escalar no quesito consumo de energia.

Neste trabalho realizamos uma comparação quantitativa entre o consumo de energia da arquitetura Super Escalar do processador Alpha 21264 e o de um processador com arquitetura DTSVLIW e com níveis equivalentes de complexidade de hardware e de desempenho. O resultado deste trabalho nos permitiu quantificar os benefícios da adoção da arquitetura DTSVLIW no lugar da arquitetura Super Escalar.

3. Consumo de Energia

3.1. Freqüência e tensão de operação

A Equação I, abaixo, mostra a interdependência entre a freqüência de operação e a tensão de alimentação de um transistor CMOS [10]:

$$f \propto (V - V_{th})^{\alpha} / V \quad (Equação\ I)$$

Na Equação I, V é a tensão fornecida ao transistor, V_{th} é sua tensão de *threshold* e o expoente α é uma constante derivada experimentalmente que, para as tecnologias atuais, é aproximadamente igual a 1,3.

Usando dados da indústria, é possível modificar a Equação I de modo obter uma nova equação relacionando freqüência e tensão própria para o momento tecnológico atual. Para isso, definimos uma tensão de operação V_{norm} e uma freqüência f_{norm}, as quais são normalizadas para a máxima tensão de alimentação, V_{max}, e máxima freqüência de operação, f_{max}. Podemos, então, definir uma relação linear entre tensão e freqüência na forma da seguinte equação [10]:

$$V_{norm} = \beta_1 + \beta_2 \cdot f_{norm} \quad (Equação\ II)$$

onde as constantes $\beta 1 = V_{th} / V_{max}$ e $\beta 2 = 1 - \beta 1$.

Através da Equação II, que prediz resultados obtidos recentemente pela indústria, é possível ver que quando $f =$ 0, $V_{norm} = V_{th} / V_{max}$. V_{th} / V_{max} (V_{norm}) é aproximadamente igual a 0,3 para as tecnologias atuais. A Equação II também indica que a redução da freqüência de operação de uma percentagem, p, de f_{max}, permite que a tensão de alimentação seja reduzida por uma percentagem menor que p. Por exemplo, se nós assumirmos $\beta 1$ = 0,3, a redução da freqüência de 50% (p = 50%, isto é, f_{norm} = 0,5) permite a redução da tensão de alimentação em apenas 35% (V_{norm} = 0,65). Igualmente, a redução da tensão pela metade (V_{norm} = 0,5) força uma redução da freqüência de operação por mais da metade ($f_{norm} \approx 0{,}3$).

3.2. Consumo global de energia

A Equação III, abaixo, define o consumo global de energia de um circuito CMOS como sendo igual à soma da potência dinâmica com a estática [10].

$$P = f \cdot A \cdot C \cdot V_{DD}^{\ 2} + V_{DD} \cdot I_{leak} \quad (Equação\ III)$$

O primeiro termo da Equação III modela o consumo de energia dinâmico, resultado do carregamento e descarregamento das cargas capacitivas existentes no circuito, enquanto que o segundo termo modela o consumo estático devido à corrente de fuga, I_{leak}. Nós ignoramos na Equação III o consumo de energia devido à corrente de curto-circuito, a qual momentaneamente flui entre a alimentação e o terra quando um circuito CMOS

comuta. Tal consumo é relativamente pequeno e o primeiro termo da equação pode absorvê-lo.

3.3. Análise do consumo dinâmico

Um inversor CMOS, mostrado simbolicamente na Figura 2, permite ilustrar as causas do consumo de energia dinâmico. O elemento básico de todo circuito implementado com tecnologia CMOS – tecnologia empregada no projeto dos circuitos digitais que implementam a grande maioria dos processadores atuais – é o inversor lógico. Uma vez que o consumo de energia num circuito inversor seja claramente formulado, esta formulação poderá ser empregada em circuitos mais complexos.

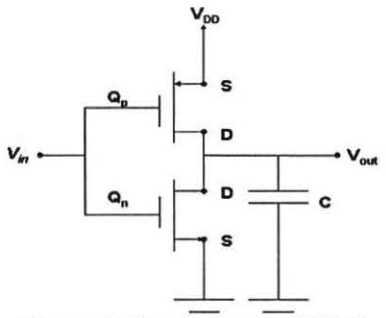

Figura 2: Um inversor CMOS

Um inversor CMOS utiliza dois transistores MOSFET tipo enriquecimento; um com *canal n* e um outro com *canal p*. Na Figura 2, V_{DD} é a tensão de alimentação do circuito (V da Equação I) e C representa a soma das capacitâncias internas do MOSFETs, Q_n e Q_p, a capacitância das linhas de interconexão entre o nó de saída do inversor e entrada das outras portas lógicas, mais as capacitâncias destas outras portas de carga.

Suponha que no instante imediatamente anterior a t = 0 V_{in} = 0, V_{out} = V_{DD} e que a energia armazenada no capacitor seja igual a $½*C*V_{DD}^2$ (C completamente carregado). Se em t = 0 V_{in} assume um valor igual a V_{DD}, Q_p corta e Q_n conduz. O transistor Q_n descarrega o capacitor e, no final do intervalo de descarga, a energia $½*C*V_{DD}^2$ é removida de C e dissipada em Q_n.

Vamos agora considerar a outra metade do ciclo, quando V_{in} vai para zero. Neste caso, o transistor Q_n corta e Q_p conduz, recarregando capacitor C. Seja a corrente instantânea fornecida ao capacitor por Q_p representada por i. Essa corrente vem, certamente, da fonte V_{DD}. Portanto, a energia consumida da fonte durante o período de carregamento será igual a:

$$\int V_{DD} \cdot i \cdot dt = V_{DD} \int i \cdot dt = V_{DD} \cdot Q \quad (Equação\ IV)$$

onde Q é a carga fornecida ao capacitor, que é igual a $C*V_{DD}$. Portanto, a energia consumida pela fonte durante o intervalo de carregamento é $C*V_{DD}^2$. No fim do intervalo de carregamento, a tensão no capacitor será igual a V_{DD} e, portanto, a energia armazenada nele será novamente igual a $½*C*V_{DD}^2$. Segue que, durante o intervalo de carregamento, metade da energia fornecida pela fonte, $½*C*V_{DD}^2$, é dissipada em Q_p.

Da discussão anterior, conclui-se que há uma dissipação total de potência no inversor de $C*V_{DD}^2$ a cada ciclo de operação. Agora, se o inversor for chaveado a uma taxa de $f*A$ ciclos por segundo (onde f é a freqüência de *clock* e A é o fator de atividade, que é menor ou igual a 1, pois nem todas as portas comutam a cada ciclo), a dissipação de potência dinâmica será igual a (ver Equação III) [10]:

$$P_{dinamica} = f \cdot A \cdot C \cdot V_{DD}^{\ 2} (Equaçao \quad V)$$

Atualmente, a potência dissipada dinamicamente é a principal causa do consumo de energia dos processadores, tanto que, em tecnologias de fabricação menos agressivas, nós poderíamos aproximar o consumo total ao consumo dinâmico, dado pela Equação V. Nesta equação, o fator V_{DD}^2 sugere que a redução da tensão de alimentação é o meio mais efetivo de diminuir o consumo de energia. Porém, a Equação II mostra que a redução da tensão de alimentação pela metade força uma redução da freqüência de operação por mais da metade.

3.4. Análise do consumo estático

Com os avanços tecnológicos recentes, hoje já é possível fabricar transistores com largura de canal inferior a 100nm. Infelizmente, a redução da largura do canal dos transistores (e conseqüentemente seu tamanho total) exacerba seu consumo estático de energia, o que tem feito com que a potência estática comece a dominar a potência total consumida pelos microprocessadores. Vale citar que, em 2001, o *International Techology Roadmap for Semiconductors* [14] predisse que, em futuras gerações de processadores (com tecnologia de fabricação abaixo de 65nm), a dissipação de potência dinâmica será ultrapassada pela de potência estática.

O consumo de energia estático é causado pela "fuga" de corrente através dos transistores quando eles estão em estado estacionário, ou seja, não estão "chaveando". A corrente de fuga, I_{leak}, é a causa do consumo de energia estático, e é o resultado da combinação das correntes de fuga *subthreshold*, I_{sub}, e *gate-oxide*, I_{ox} [10]:

$$I_{leak} = I_{sub} + I_{ox} \quad (Equação\ VI)$$

Corrente de fuga Subthreshold

A corrente de fuga *subthreshold* depende da tensão de *threshold* e da tensão de alimentação, e é dada por [10]:

$$I_{sub} = k_1 \cdot W \cdot e^{-V_{th}/nV_\theta}\left(1 - e^{-V/V_\theta}\right) \quad (Equação\ VII)$$

K_1 e n são constantes derivadas experimentalmente, W é a largura do canal do transistor e V_θ é tensão térmica. À temperatura ambiente, V_θ é aproximadamente 25mV; ela aumenta linearmente com o aumento de temperatura. Se

I_{sub} aumentar a ponto de elevar a temperatura, V_θ também aumentará, o que, conseqüentemente, aumentará I_{sub} e possivelmente causará uma corrida térmica.

A Equação VII sugere dois modos de reduzir I_{sub}. Um deles seria reduzir V, o que faz com que o fator entre parênteses tenda a zero. O outro seria incrementar V_{th}. Porém, sabemos da Equação I que incrementar V_{th} reduzirá a velocidade de operação, o mesmo ocorrendo quando da redução da tensão de operação (ver Seção 3.1).

W é outro fator que contribui para I_{sub}. Tanto que ele é utilizado por alguns arquitetos para medir a corrente total de fuga *subthreshold*.

Corrente de fuga gate-oxide

De forma simplificada, a corrente de fuga *gate-oxide* é dada pela equação [10]:

$$I_{ox} = k_2 \cdot W(V/T_{ox})^2 e^{-\alpha \cdot T_{ox}/V} \quad (Equação\ VIII)$$

onde k e α são derivados experimentalmente e T_{ox} é a espessura do dielétrico de óxido que isola o *gate* do canal. É facilmente visível que o aumento de T_{ox} provoca uma redução da corrente de fuga I_{ox}. Infelizmente isto também degrada a performance do transistor porque T_{ox} deve decrescer proporcionalmente com a tecnologia de fabricação do mesmo. Logo, incrementar T_{ox} não é uma opção.

4. Trabalhos Relacionados

4.1. Simuladores de consumo de energia

O simulador Wattch modela um processador Super Escalar e seu consumo de energia dinâmico [4]. A principal vantagem de utilizar um simulador como o Wattch no lugar de ferramentas de simulação física é que ele pode ser até mil vezes mais rápido e seu erro de estimativa situa-se na faixa de 10 a 13% [4].

O simulador Hotleakage [17] é o primeiro simulador arquitetural capaz de medir potência dissipada estaticamente. Este simulador computa o consumo estático e apresenta a limitação de apenas estimar a potência de estruturas de hardware que possuem grande quantidade de células de memória (caches e banco de registradores). Contudo, tendo em vista que estas estruturas ocupam a maior parte do chip, esta abordagem apresenta uma boa aproximação do consumo estático total.

4.2. Modelagem realizada pelo Wattch

O Wattch faz uso de modelos de potência parametrizáveis correspondentes a cada unidade de hardware presente em modernos processadores Super Escalares. Ele calcula o consumo dinâmico ciclo a ciclo a partir valores dos parâmetros da Equação V para cada dispositivo que faça parte do processador. A capacitância C é estimada a partir do tamanho dos transistores e das suas interconexões. Já V_{DD} e f dependem do processo tecnológico assumido e da escolha do arquiteto, enquanto que A depende da arquitetura de cada parte do processador e dos programas sendo executados nele. Para partes do processador onde A não é medido, assume-se um valor igual a 0,5.

O Wattch foi modelado supondo-se que as capacitâncias são similares àquelas usadas por Wilton e Jouppi [16] e Palacharla, Jouppi e Smith [12, 13]; ou seja, nele cada unidade de hardware é dividida em estágios e forma circuitos RC. Isto permite, então, estimar a capacitância de cada estágio, e pela soma destas, a capacitância de unidade inteira. O simulador ainda apresenta três estilos de *clock* condicional – mecanismo através do qual uma unidade só recebe o sinal de *clock* se for operar em um determinado ciclo. Processadores atuais fazem uso agressivo desta técnica para desabilitar partes do hardware a fim de reduzir o consumo quando elas não precisam ser usadas. Algumas unidades de Hardware modeladas pelo Wattch:

- *Array Structures*: caches, banco de registradores e tabelas de predição de desvio;
- *Content Addressable Memories* (CAM): TLBs, fila de emissão, *buffer* de reordem;
- Lógica combinacional e malhas: Unidades Funcionais, lógica de checagem dependência;
- Rede de transmissão do *clock*.

4.3. Modelagem realizada pelo Hotleakage

O Hotleakage é um módulo configurável acrescido ao simulador Wattch para o computo do consumo estático de energia. Este módulo avalia dinamicamente a potência estática de cada célula de interesse (por exemplo, uma célula SRAM) e esta informação é então agrupada no nível arquitetural. As funções que calculam a potência estática de cada estrutura da micro-arquitetura do processador modelado utilizam como base as equações VII e VIII.

O Hotleakage, atualmente, modela a corrente de fuga de caches e bancos de registradores. Para isso, ele faz uso dos seguintes parâmetros (equações VII e VIII):

- A espessura do dielétrico, T_{ox};
- Tensão de alimentação, V_{DD};
- Temperatura de uma estrutura específica;
- Tensão de *threshold* de transistores de estruturas específicas;
- Variação em V_{DD} devido às variações *inter-die*;
- Variação do comprimento do canal do transistor, W.

5. Modelagem do Consumo de Energia da Arquitetura DTSVLIW

Implementamos uma nova versão do nosso simulador DTSVLIW capaz de medir o consumo de energia das unidades de hardware desta arquitetura. Para fazer o simulador executar os experimentos da maneira esperada, foi necessário alterar seu código-fonte (o simulador foi

escrito em Linguagem C e roda no sistema operacional Linux), adicionando algumas funções que tornam possíveis as estimativas do consumo de potência destas unidades da arquitetura DTSVLIW. Nós usamos modelagem equivalente à utilizada nos simuladores Wattch e Hotleakage. Como todos os três simuladores são baseados no simplescalar, o trabalho de adaptação do simulador DTSVLIW foi facilitado. Na adaptação foram levadas em consideração a organização das unidades funcionais do processador Super Escalar Alpha 21264 e sua rede de distribuição de clock [3, 8], a formulação dos modelos [16, 12, 13] e implementações do Wattch [4] e Hotleakage [17], e parâmetros da tecnologia CMOS atual [10, 15, 9, 18].

6. Métodos

Nosso trabalho de implementação de um simulador DTSVLIW com medidores de consumo teve como propósito tornar possível uma avaliação prévia do consumo que uma possível implementação física desta arquitetura teria. A fim de permitir uma melhor compreensão dos resultados obtidos com a arquitetura DTSVLIW no contexto dos processadores atuais, fizemos uma comparação desta com o processador Alpha 21264 em situação equivalente. Para a realização da comparação entre o consumo DTSVLIW com o do Alpha 21264, utilizamos uma composição dos dois simuladores discutidos na Seção 4: o Wattch [4] e o Hotleakage [17].

6.1. Programas de Teste

Ambos os simuladores utilizados (DTSVLIW e Alpha 21264) recebem como entrada executáveis produzidos por compiladores comuns que geram código para a Alpha ISA. Para os experimentos que descreveremos a seguir, usamos uma parte do conjunto de executáveis do SPEC2000 (www.specbench.org) disponibilizada junto com o simulador simplescalar (www.simplescalar.com). Estes executáveis foram produzidos em uma máquina Alpha 21264 rodando o SO Digital UNIX V4.0F, e foram compilados pelo compilador DEC C V5.9-008 (Rev. 1229), ou pelo compilador Compaq C++ V6.2-024 (Rev.1229), ou, ainda, pelo compilador Compaq Fortran V5.3-915 (f77 e f90). Como entradas para os programas do SPEC2000 usamos o conjunto de entradas desenvolvido na University of Minnesota [11]. Com este conjunto de entradas, os programas do SPEC2000 selecionados pelos pesquisadores desta universidade requerem apenas cerca de um bilhão de instruções para sua execução (aproximadamente um segundo de processamento em uma máquina Alpha atual), mas este número de instruções permite capturar o desempenho do processador quando executando estes programas. Todos os programas para os quais a University of Minnesota desenvolveu entradas foram executados até terminar, ou até o limite de 2,5 bilhões de instruções. A Tabela 1 mostra os programas de teste utilizados e o número de instruções executadas em cada um deles. Uma descrição detalhada destes programas pode ser encontrada na literatura [11].

6.2. Configurações Básicas dos Simuladores

As configurações utilizadas nos experimentos são como as indicadas nas tabelas numeradas de 2 a 5. A Cache VLIW utilizada é bastante pequena e tem 36KB (Tabela 2).

Tabela 1: Programas de teste utilizados

Benchmarks Inteiros	Número de Instruções Executadas	Benchmarks de Ponto Flutuante	Número de Instruções Executadas
bzip2	1.819.782.161	mesa	1.688.627.786
gcc	2.500.000.000	art	1.660.422.409
gzip	1.583.267.837	mcf	794.460.163
parser	2.500.000.000	equake	1.021.625.144
twolf	972.968.480	ammp	1.247.352.121
vortex	1.154.291.894		
vpr	1.567.083.914		

Tabela 2: Configuração DTSVLIW

Processador Primário	• *pipeline* de quatro estágios (busca, decodificação, execução e escrita); • sem hardware de predição de desvios; • desvios tomados geram uma bolha de 2 ciclos no *pipeline*; • Cache de Instruções de 16KB, 2-way set associative, latência 1.
Cache VLIW	36KB, 2-way set associative, latência 1
Tamanho de uma Instrução no Cache VLIW	6 bytes
Pipeline da Máquina VLIW	3 estágios (fetch, dispach, execute)
Tamanho da Lista de Escalonamento	2 vezes o número de LIs do bloco
Registradores usados para Renomeação	Inteiros 17, Ponto Flutuante 13 e Memória 6

Tabela 3: Configuração Alpha 21264

Pipeline	• 8 estágios – *Fetch, Slot, Map, Issue, Register Read, Execute, Write-back* e *Retire*. • *Fetch, Slot* e *Map* – 4 instruções por ciclo • *Issue, Register Read, Execute e Write-back* – 6 instruções por ciclo • *Retire* – 11 instruções por ciclo
Unidades Funcionais	4 inteiras e 2 de ponto flutuante
Tamanho das *Issue queues*	Instruções Inteiras 20, Ponto Flutuante 15
Número de Registradores usados para Renomeação	Inteiros 41, Ponto Flutuante 41 e Memória 32 (load e store *queues*)
Preditor de Desvios	*Tournament branch predictor* com uma combinação de três preditores: *two level local preditor* (1024 10-bit *local history*), *path-based global predictor* (12-bit *history register* que aponta para uma tabela de 4K contadores saturados de 2 bits) e um *choice preditor* que escolhe a predição de um dos dois anteriores (4K contadores saturados de 2 bits)

Tabela 4: Latência das Instruções

Instrução	Latência
Inteira	1
Multiplicação Inteira	7
Load Inteira (Com cache hit)	3
Soma e Multiplicação de Ponto Flutuante	4
Divisão / Raiz Quadrada de Ponto Flutuante (*SP*)	12 a 18
Divisão / Raiz Quadrada de Ponto Flutuante (*DP*)	15 a 33
Load de Ponto Flutuante (Com acerto no cache)	3
Desvio incondicional	3

Tabela 5: Hierarquia de Memória

Cache de Instruções (apenas para Alpha)	64 KB, *2-way set associative*, latência 1 ciclo
Cache de Dados	64 KB, *2-way set associative*, latência 3 ciclos
Cache Nível 2, Unificado	1 MB, direct-mapped, physical-indexed, latência 7 ciclos
Memória RAM	Ilimitada, latência de 66 ciclos com *precharge* e 54 ciclos em acessos *pipelined*

7. Experimentos

Nós utilizamos quatro tipos de tecnologia de fabricação nos experimentos para medir o consumo de potência dinâmica: 70nm, 100nm, 130nm e 180nm. Porém, uma vez que a última implementação do processador Alpha 21264 foi implementada na tecnologia de 180nm, apresentamos aqui apenas os resultados para esta tecnologia.

Praticamente a totalidade da energia consumida nos processadores á transformada em calor. Assim, nós apresentamos nossas medidas de consumo em Watts dissipados na forma de calor. Os experimentos foram rodados no cluster de 64 processadores do Laboratório de Computação de Alto Desempenho da UFES (www.inf.ufes.br/~lcad).

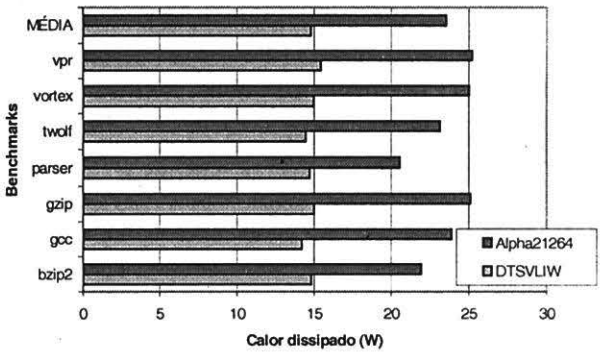

Figura 3: DTSVLIW x Alpha para 180nm – Inteiros

As figuras 3 e 4 apresentam gráficos que mostram a energia dissipada em Watts (W) de processadores Alpha 21264 e DTSVLIW configurados conforme os parâmetros das tabelas 2, 3, 4 e 5, com uma tensão de alimentação de 2V, uma freqüência de trabalho de 1GHz e uma tensão de *threshold* 0,55V. Nestas figuras, a dissipação de calor de cada unidade de hardware de nossos simuladores foi escalada linearmente com o número de acessos a elas, ou seja, se em um ciclo uma unidade á acessada a sua dissipação total é contabilizada, caso contrário, não. Porém, nós adicionamos uma dissipação equivalente a 10% da dissipação máxima de cada unidade quando a unidade não é acessada [4].

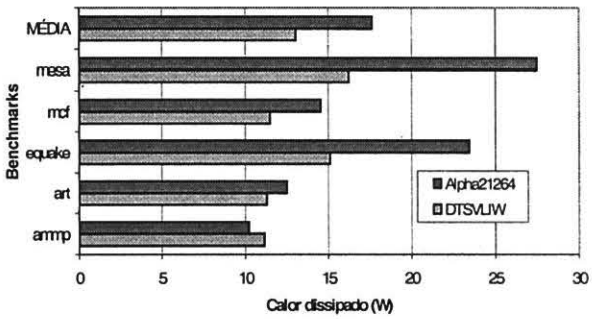

Figura 4: DTSVLIW x Alpha para 180nm – P.F.

Como os gráficos das figuras 3 e 4 mostram, o calor dissipado pelo processador DTSVLIW é significativamente menor que o dissipado pelo Alpha 21264 para programas inteiros e de ponto flutuante. O calor dissipado pelo DTSVLIW é menor que o Alpha 21264 para todos os programas inteiros, com uma média de 14,80W contra 23.54W do Alpha 21264 – ou seja, uma dissipação 37,12% menor em média. Já para os programas de ponto flutuante, o calor dissipado pelo DTSVLIW é significativamente menor que o Alpha 21264 para os programas mesa e equake, mas esta diferença é menos pronunciada nos casos de mcf, art e ammp, sendo que, no caso deste último, o Alpha 21264 supera o DTSVLIW. Contudo, em média, o DTSVLIW

dissipa 13,06W contra 17,63W do Alpha 21264 – ou seja, a quantidade de calor dissipada pelo DTSVLIW em programas de ponto flutuante foi 25,92% inferior que a quantidade dissipada pelo Alpha 21264.

A arquitetura DTSVLIW dissipa menos calor porque executa os programas em dois modos: um escalar, quando, além de executar o código no Processador Primário, sua Unidade de Escalonamento escalona e salva o código na forma de instruções VLIW; e um VLIW, quando as instruções VLIW são executadas na Máquina VLIW. A DTSVLIW executa código em modo VLIW na maioria dos ciclos de relógio; assim, sua unidade de escalonamento não recebe pulsos de *clock* e sua dissipação de calor é em muito reduzida neste modo. A arquitetura Super Escalar do Alpha 21264, por outro lado, escalona o código em todos os ciclos para extrair seu ILP; deste modo, o hardware de escalonamento deste processador precisa receber pulsos de *clock* durante praticamente toda a execução. Além de receber pulsos de *clock* praticamente todo o tempo, o hardware de escalonamento precisa receber os resultados produzidos pelas unidades funcionais e usa-los para habilitar novas instruções aguardando resultados para serem enviadas para execução. Ou seja, as três partes do hardware de escalonamento Super Escalar produzem as diferenças de dissipação de calor observadas, quais sejam: (i) a lógica de despacho de instruções vindas dos estágios de busca para as estações de reserva, (ii) a lógica usada para propagar os resultados para as estações de reserva e habilitar as instruções prontas, e (iii) a lógica de envio destas instruções prontas das estações de reserva para as unidades funcionais.

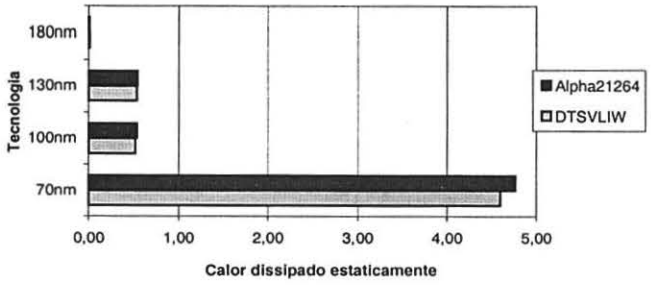

Figura 5: DTSVLIW x Alpha – dissipação estática

O gráfico da Figura 5 apresenta o calor dissipado estaticamente nos dois processadores estudados para as tecnologias usadas nos experimentos. Ele mostra que a dissipação de calor estática é muito pequena nos dois processadores para uma implementação com tecnologia de 180nm, mas começa a se tornar significativa a partir de 70nm. A quantidade de calor dissipado estaticamente nos processadores DTSVLIW e Alpha 21264 modelados é, contudo, muito próxima, como pode ser visto na Figura 5, uma vez que o hardware dos dois processadores foi feito o mais equivalente possível. A tendência para o futuro é a dissipação de calor estática superar a dinâmica [10]. Contudo, diversas técnicas de implementação de chips estão sendo estudadas para limitar a dissipação de calor estática, como dielétricos de alto *k* (*high-k dielectrics*) [10] e transistores baseados em nanotubos de carbono [1], por exemplo.

8. Conclusão

Como nossos resultados experimentais mostraram, a arquitetura DTSVLIW consome menos energia por unidade de tempo (dissipa menos calor) que a Super Escalar dentro do contexto tecnológico do processador Alpha 21264. Tal resultado nos leva à conclusão de que a implementação de processadores segundo a arquitetura DTSVLIW é uma solução competitiva para atacar o crescente problema do consumo de energia nos processadores atuais.

Uma vez que a arquitetura DTSVLIW não foi implementada fisicamente, não podemos, ainda, validá-la nem verificá-la quanto ao seu consumo de energia físico real; porém, como o nosso simulador foi baseado no Wattch, esperamos que ele apresente o mesmo grau de erro deste, de 10 a 13% [4].

Referências

[1] P. Avouris, "Supertubes", IEEE Spectrum, Vol. 41, No. 8 (INT), p. 34-39, August 2004.

[2] Digital Equipment Corporation, "Alpha Architecture Handbook", Digital Equipment Corporation, 1992.

[3] D. W. Bailey, B. J. Benschneider, "Clocking Design and Analysis for a 600MHz Alpha Microprocessor", IEEE Journal of Solid-State Circuits, Vol. 33, No. 11, pp. 1627-1633, Nov. 1998.

[4] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A framework for architectural-level power analysis and optimizations", Proceedings of the 27th Annual International Symposium on Computer Architecture, pp. 83-94, June 2000.

[5] D. Burger and J. Goodman, "Billion-transistor Architectures", IEEE Computer, Vol. 30 No. 9, pp. 46-49, Sept. 1997.

[6] A. F. De Souza and P. Rounce, "Dynamically Trace Scheduled VLIW Architectures", Proceedings of the High-Performance Computing and Networking' 98 – HPCN'98, on Lecture Notes in Computer Science, Vol. 1401, pp. 993-995, Apr. 1998.

[7] A. F. De Souza and P. Rounce, "Dynamically Scheduling VLIW Instructions", Journal of Parallel and Distributed Computing, Vol. 60, No. 12, pp. 1480-1511, Dec. 2000.

[8] M. K. Gowan, L. L. Biro, and D. B. Jackson, "Power considerations in the design of the Alpha 21264 microprocessor", Proceedings of the 35th Design Automation Conference, pp. 726-731, 1998.

[9] M. B. Kamble and K. Ghose, "Analytical Energy Dissipation Models for Low Power Caches", Proceedings of the International Symposium on

Low-power Eletronics and Design, pp.343-348, 1997.

[10] N. S. Kim, T. Austin, D. Blaauw, T. Mudge, K. Flautner, J. S. Hu, M. J. Irwin, M. Kandemir and V. Narayanan, "Leakage Current: Moore's Law Meets Static Power", IEEE Computer, Vol. 36, No. 12, pp. 68-75, December 2003.

[11] A. J. KleinOsowski and D. J. Lilja, "MinneSPEC: A New SPEC Benchmark Workload for Simulation-Based Computer Architecture Research", Computer Architecture Letters, Volume 1, June, 2002.

[12] S. Palacharla, N. Jouppi, and J. Smith, "Complexity-Effective Superscalar Processors", Proceedings of the 24th Annual International Symposium on Computer Architecture, 1997.

[13] S. Palacharla, N. Jouppi, and J. Smith. Quantifying the Complexity of Superscalar Processors. University of Wisconsin Computer Science Tech. Report 1328, 1997.

[14] Semiconductor Industry Association (SAI), "International Technology Roadmap for Semiconductors", http://public.itrs.net.

[15] T. Wada, S. Rajan, and S. A. Przybylski, "An Analytical Access Time Model for On-Chip Cache Memories", IEEE Journal of Solid-State Cicuits, Vol. 27, No. 8, pp. 1147-1156, Aug. 1992.

[16] S. Wilton and N. Jouppi, "An Enhanced Access and Cycle Time Model for OnChip Caches", Technical Report No. 93/5, DEC-Western Research Lab, 1994.

[17] Y. Zhang, D. Parikh, K. Sankaranarayanan, K. Skadron, and M. Stan, "Hotleakage: A temperature-aware model of subthreshold and gate leakage for architects", Technical Report CS-2003-05, University of Virginia Department of Computer Science, Mar. 2003.

[18] R. Zimmermann and W. Fichtner, "Low-power logic styles: CMOS versus pass-transistor logic", IEEE Journal of Solid-State Circuits, Vol. 32, No. 7, pp. 1079-1090, Jul. 1997.