# Relatório de Bolsa

David Ricardo Jorge dos Santos Gil

Técnicas de Software para a optimização de energia em arquitecturas de computadores.

## Índice:



## Resumo:

Durante o curso do ano de 2003, no âmbito da minha colaboração como bolseiro no projecto COOLCHIPS: ambiente de projecto e analise de sistemas de baixo consumo desenvolvi as seguintes actividades:

Fase I – Pesquisa sobre o trabalho já desenvolvido e publicado.
Fase II – Pesquisa sobre Modelos de estimação de Energia/Desempenho.
Fase III – Estudo e Implementação de um modelo.

Na primeira fase o objectivo foi ficar a par do trabalho nesta área efectuado pela comunidade cientifica internacional. Em resumo, concluiu-se que já há trabalho muito relevante realizado nesta área, nomeadamente por Tiwari et al. [1][2][3][4][5][8] que no seu artigo [4] introduziu a estimação de energia/desempenho ao nível da instrução. Esta técnica muito eficaz permitiu em [5] desenvolver técnicas de minimização (até 70%) de consumo muito eficazes (como re-agendamento de instruções) para um processador muito simples DSP (Digital Signal Processor) mas amplamente utilizado especialmente em telecomunicações móveis e sistemas de controlo. Contudo Russel e Jacome [6] vieram provar que estas técnicas de estimação e minimização de potência ao nível da instrução não são aplicáveis a processadores mais complexos, e que é preferível utilizar técnicas que já eram utilizadas para optimização de desempenho, como o enlinhamento de funções (Function Inlining) e o desenrolamento de loops (Loops Unroling).

O objectivo deste trabalho é desenvolver técnicas de software para minimizar o consumo de energia num sistema computacional (CPU + periféricos), para isso é

necessário ter um ambiente de teste que podem ser tanto físicos [5][6] como simulados em computador [7][8][9][10][11][12][13][14][15]. Por tanto, na segunda fase, o objectivo foi pesquisar especificamente modelos de estimação. Concluiu-se que apesar de existirem muitos simuladores de processadores/arquitecturas, são muito poucos aqueles que fornecem resultados de energia consumida e menos ainda os que simulam uma arquitectura completa com CPU, memórias CACHE e memória principal. Desta pesquisa chegou-se à conclusão que os simuladores existentes mais aptos para a adopção como plataforma de estimação de energia/desempenho são o Avalanche[7], Wattch[8] e o SimplePower [14] e o ARM Power Analyser [15]. O mais completo é o SimplePower [14] e foi o utilizado inicialmente, mas devido à pouca fiabilidade de resultados e dificuldades de utilização, passou-se a utilizar o Wattch [8] que tem a grande vantagem de ser executável em computadores máquinas com sistema operativo Linux mais rápidas e disponíveis. Contudo nenhum dos simuladores pode ser utilizado como ferramenta única, foi necessário liga-los por exemplo a simuladores de memória CACHE, como o Dinero IV [10] e respectivos estimadores de potência: CACTI [16].

## Fase I – Pesquisa sobre o trabalho já desenvolvido e publicado.

A tendência inicial para estimação de energia consumida em computadores é fazer uma simulação ao nível do circuito, isto é, ao nível da porta lógica ou mais ainda, ao nível do transístor, recorrendo a ferramentas como o SPICE ou VHDL. Esta técnica, apesar de precisa, é impraticável quando o alvo de exploração é o Software, visto que um programa mesmo dos mais simples precisa de milhares de instruções para ser executado o que torna uma simulação de tão baixo nível muito demorada, complexa e pouco prática.

A tendência seguinte proposta em [4] foi a propor um modelo de estimação baseado em medições físicas. Ligando aparelhos de medida de potência eléctrica aos terminais de um CPU simples (um DSP), e analisando os seus dados, discretizou-se o consumo de energia no CPU ao nível da instrução e pormenorizou-se a técnica até se descobrir o efeito de instruções encadeadas bem o como o efeito dos seus parâmetros no consumo de energia. Em [4] verifica-se ainda que depois de estudado o consumo de energia no processador, é então possível estimar com 10% de erro o consumo de energia durante a execução de qualquer programa desde que se conheça o seu código fonte. A exploração adequada deste modelo levou a poupanças de energia até 70% verificados experimentalmente [5]. Este tipo de modelo tem a desvantagem de ser aparatoso, pois não dispensa a existência física do Processador e sistema de medida.

Contudo, baseando-se ainda em medições físicas, [6] veio provar que num processador de alta-desempenho de 32 bits, ou seja num processador complexo, todas as instruções consomem aproximadamente a mesma energia e que as instruções que as precedem ou lhes seguem têm influencia reduzida.

# Fase II – Pesquisa sobre Modelos de estimação de Energia/Desempenho.

Um dos primeiros Simuladores que surgiu foi o WARTS [10] que é um colecção de ferramentas para analisar a execução de programas bem como o seu traço de acesso à memória, o que é perfeitamente adequado para explorar o sistema de memória. Uma das ferramentas que esta colecção fornece é o Dinero III [10] um simulador de memória CACHE baseado no traço de acesso à memória, o que traz a vantagem de fazer a medição rápida e precisa de dados estatísticos como a Taxa de Faltas e outros, muito importantes para o desempenho, que, como veremos adiante possibilita também a estimativa da energia consumida no sistema de memória durante a execução do programa.

Os modelos anteriores restringiam-se ao núcleo de processamento, o CPU. A malha *Avalanche* [7] veio propor uma malha de estimação de energia para um computador completo, com CPU, memória CACHE e memória principal. Este modelo utiliza componentes que não são do domínio público e a sua exploração limita-se à variação de parâmetros da memória CACHE e análise dos seus efeitos.

O modelo proposto em [9], o *Platune* cumpre os objectivos de estimação de potencia e desempenho mas de uma forma simplista adequada apenas, como defendida pelos autores para o desenho preliminar de uma arquitectura, já que apenas permite a varição de cerca de uma vintena de parametros da arquitectura, como se pode ver pela fig.1 .

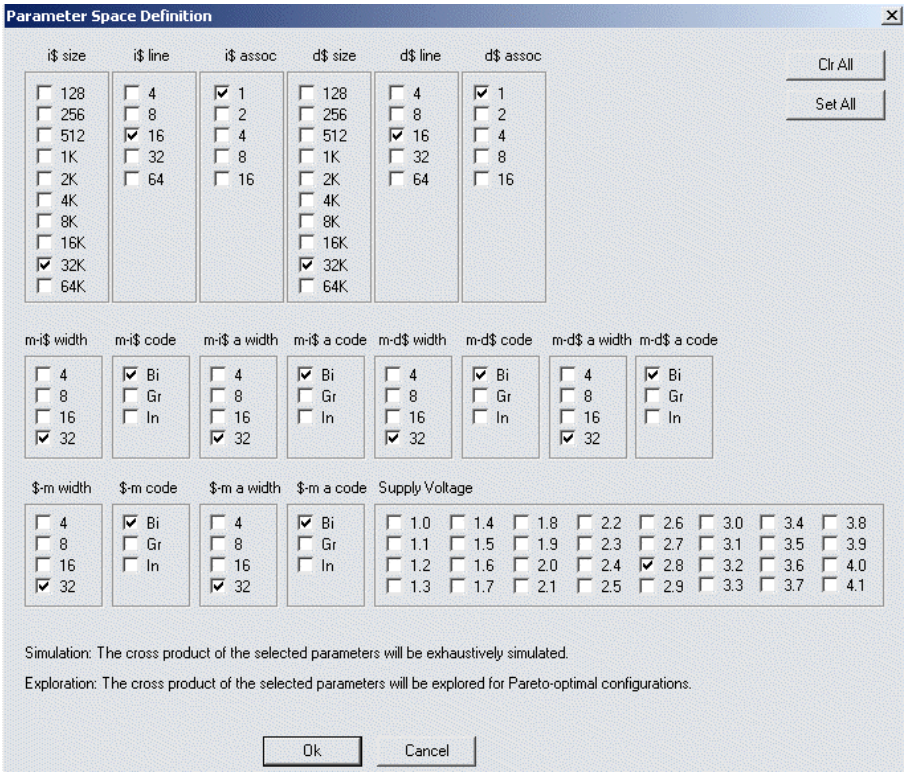



**fig. 1 - Janela de Configuração da Simulação do Platune**

Este simulador tem ainda as desvantagens de não permitir a configuração do compilador e de não aceitar código fonte em linguagem de baixo nível tipo Assembly.

O SimpleScalar [11] é uma ferramenta muito versátil que serve para simular a execução de programas em variadissimas arquitecturas 100% configuráveis. Tem tambem a grande vantagem de oferecer um leque de ferramentas que permitem obter os compiladores, assembladores, loaders, etc. para essas arquitecturas. Este simulador é amplamente utilizado por investigadores e pela industria, mas por si só, não oferece ferramentas de estimação de energia.

O SimICS [12] e o SimOS [13] são emuladores muito detalhados que podem emular vários tipos de máquinas inclusivamente com multiprocessadores, podem emular a execução de sistemas operativos completos como Linux, Windows e MacOS. Estes simuladores medem variadíssimos parâmetros estatísticos mas não estão vocacionados para a estimação de consumo de energia nem do desempenho, servem antes, tal como a maioria das ferramentas existentes, para dar uma ideia ao investigador do comportamento da sua arquitectura dado uma carga de trabalho realista.

O WATTCH [10] extende as capacidades do SimpleScalar [11] para a estimação de energia, feita analiticamente, contudo concentra-se unicamente no Processador.

## 3.1 Modelo Inicialmente Implementado

Depois de uma pesquisa extensa no meio, quer em proceedings de conferencias quer na Internet, chegou-se à conclusão que não havia um modelo práctico orientado para a exploração do Software, mas que existem ferramentas publicadas baseadas no SimpleScalar [11] como o WATTCH [10], o SimplePower [14] e o ARM PAnalyzer [15] que quando montadas com outras como o Dinero [10] e o CACTI [16] podem servir o objectivo.

O modelo implementado tem como base um simulador de execução de processadores, denominado *SimplePower*[14], que para alem de simular a execução do programa a testar, fornece uma estimativa da energia consumida pelo CPU, o numero de ciclos de processador que o programa demora a ser executado e ligando-se ao Dinero [10] faz ainda a simulação funcional da memória CACHE, devolvendo no final da simulação valores estatísticos sobre a mesma.

Para completar a estimação foram acrescentados modelos das restantes partes da arquitectura: barramentos, memória principal e memória CACHE, formando no seu conjunto a arquitectura representada na fig.2

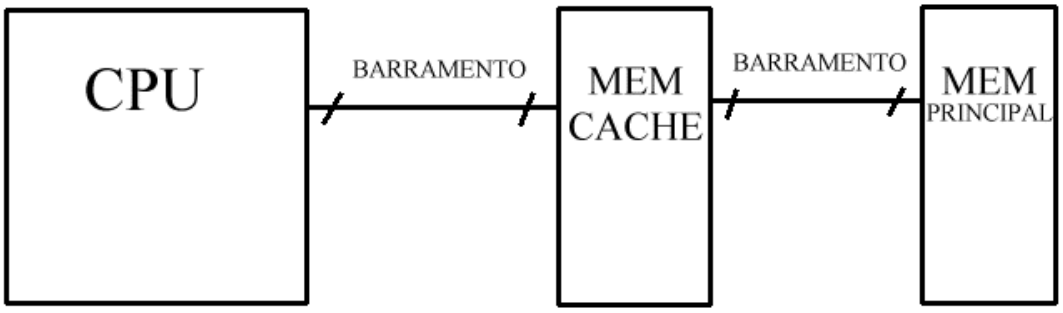



**fig. 2 - Esquema da Arquitectura Simulada.**

Algumas das ferramentas utilizadas, devolvem no final da simulação o valor da capacidade equivalente do componente que simulam, que conhecendo-se a tensão de alimentação deve-se utilizar (1) para obter o valor da energia consumida nesse componente.

$$E_i = \frac{1}{2} C V_i^2 \tag{1}$$

A ligação entre as várias componentes do modelo pode ser vista na fig.3, a qual mostra que no final da simulação, para obter o consumo total de energia, soma-se a energia consumida nos vários componentes:

$$E = E_{CPU} + E_{CACHE} + E_{MEM} + E_{BUS,CPU\text{-}CACHE} + E_{BUS,CACHE\text{-}MEM} \tag{2}$$

**NOTA:** que em geral, a soma (2) não é equivalente à soma da capacitancia equivalente dos vários componentes, visto que a sua tensão de alimentação pode não ser igual.

Eis a ligação entre os vários componentes do modelo:

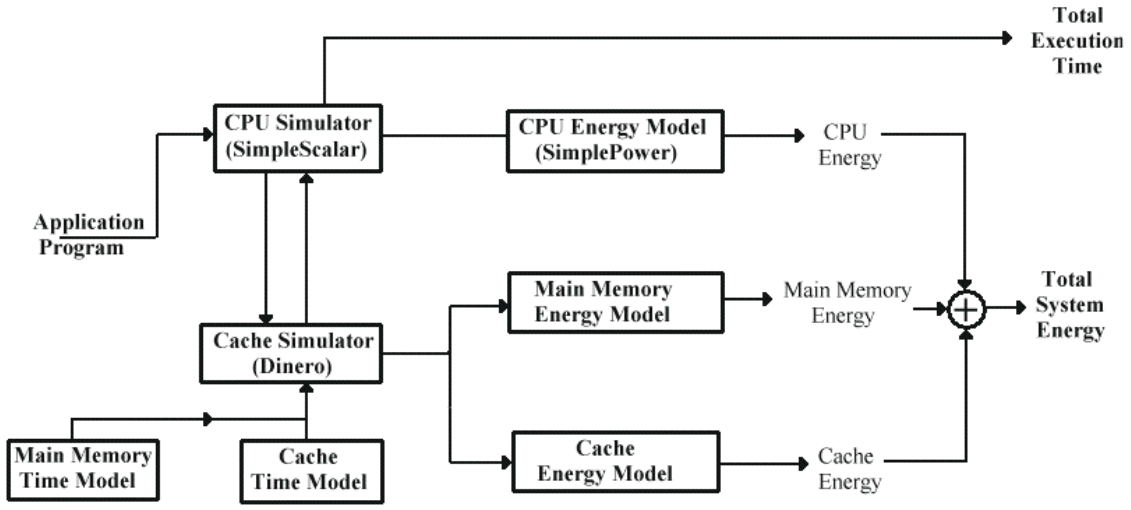



**fig. 3 - Esquema do modelo implementado**

O programa a simular é fornecido em linguagem C ou em Assembly próprio do Simplescalar[11]. O simulador é do tipo Cycle-Accurate, e interage com o simulador de Cache, ou seja, a simulação acontece ciclo-a-ciclo e em simultâneo com a simulação da Cache, e se por exemplo, houver uma falta no acesso à Cache, isso afecta a simulação do CPU.

Este processo leva a que se gere a estimativa da energia consumida pelo CPU e também os dados do desempenho da Cache, como por exemplo, o número total de acessos, taxa de faltas, etc. Estes dados são fornecidos ao modelo de energia da memória principal e Cache para calcular a energia aí consumida.

É ainda de realçar que a simulação é toda ela escalavel, ou seja, todos os componentes da simulação podem ser alterados. Por exemplo, barramentos, a memória Cache e a memória principal, muito facilmente, pois é só mudar manualmente os seus parâmetros de consumo e desempenho. O CPU também pode ser alterado, quer o modelo de energia quer o próprio funcionamento interno.

## 3.2 Modelo de energia e desempenho do CPU

Para este fim, foi utilizada a ferramenta *SimplePower*, que consiste no conhecido simulador *SimpleScalar*, modificado, para conter tambem um modelo de consumo de energia, ou seja, ao simulador *SimpleScalar*, foi acrescentado código e dados que permitem estimar a energia consumida pelo CPU durante a execução do programa.

Nesta ferramenta, os consumos são sempre descritos por capacidades, aplicando-se a conhecida formula (1) para obter o consumo de energia.

Esta energia (E) é consumida cada vez que ocorre uma transição numa linha ou porta, sendo V o valor dessa transição em unidades de tensão eléctrica.

Como se sabe, o CPU divide-se em várias unidades que se dividem em vários blocos funcionais.

Esta ferramenta tem a grande vantagem de já trazer pré-configurada uma arquitectura para simulação com um Pipeline de 5 andares, o que embora longe dos CPUs de alto-desempenho, já representa um CPU com uma elevada complexidade.

Os cinco andares são:

- IF (instruction Fetch)
- ID (Instructon Decode)
- EXE (Execution)
- MEM (Memory access)
- WB (Write Back)

A cada ciclo de relógio, é simulada a execução de todas as instruções activas e o simulador identifica quais os blocos funcionais activados, para cada um dos quais a capacidade equivalente é extraída directamente de uma tabela que tem como entradas os

valores lógicos das entradas do bloco funcional em questão: somadores, ALU, multiplicadores, unidades de deslocamento, unidades de controlo, ficheiro de registos, registos de pipeline e multiplexers.

Estas tabelas são obtidas para cada tecnologia através de várias simulações em SPICE dos esquemas dos circuitos integrados do processador.

Assim a capacidade equivalente associada à execução de uma instrução é obtida somando as capacidades de cada bloco funcional activado durante a execução desta instrução.

No final, o SimplePower mostra informação variada como: estado final do ficheiro de registos, número de ciclos da execução, número de transições nos barramentos internos do CPU, capacidades para cada andar da Pipeline, capacidades para os blocos funcionais, e capacidade total.

Uma vez que o *SimplePower* e o *SimpleScalar* são na realidade o mesmo simulador, os executáveis são compatíveis entre os dois simuladores.
Para se obter executáveis foi necessário compilar as ferramentas de desenvolvimento cruzado entre plataformas que se mostram na fig.4 (compilador,assembler,loader) a partir das ferramentas do *SimpleScalar*. Opcionalmente é possível seguir manualmente todos os passos da fig.4, mas por defeito, o gcc faz todos os passos automaticamente e de forma transparente.

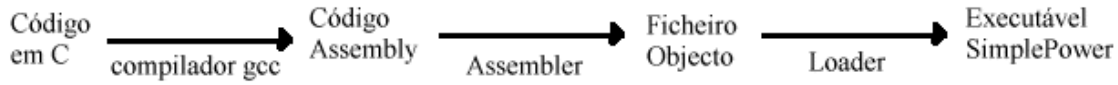



**fig. 4 - Obtenção de um binário Simulável**

## 3.3 Modelo de energia e desempenho da memória CACHE

O simulador *SimplePower* é distribuído conjuntamente com um simulador de memória CACHE. *Dinero IV*, o qual, recebendo como parâmetros as características da memória CACHE. Por exemplo, tamanho, nível de associatividade, tempos de espera em caso de falha e de sucesso, etc. simula a memória CACHE durante o tempo de execução, impondo ao processador os ciclos de espera associados às falhas (misses) ou aos sucessos (hits), conforme a situação.

Contudo o Dinero IV não estima a potência consumida na memória cache nem os tempos de acesso. Para esse fim foi utilizada a ferramenta *CACTI*, que tem com entradas os seguintes parametros:

- Tamanho da Cache em bytes,
- Tamanho do Bloco (ou Cacheline) em bytes,
- Nível de Associatividade e
- Tamanho da tecnologia.

Em função das características físicas da CACHE, o CACTI calcula os tempos de acesso e a energia consumida por acesso. Os tempos de acesso são convertidos para unidades de período de relógio do CPU e introduzidos directamente como parâmetros do simulador de CACHE *Dinero*.

$$Cache_{stall\_Cycles\left(\frac{read}{write}\right)} = T_{acesso\left(\frac{read}{write}\right)} \times f_{clock\_CPU} \tag{3}$$

Por fim, foi escrito um programa que mistura as informações da simulação (numero de acessos à cache) com as fornecidas pelo *CACTI*, para calcular qual a energia consumida pela memória CACHE.

$$E_{CACHE} = N_{readmiss} \times \left(E_{readmiss} + E_{writehit}\right) + N_{readhit} \times E_{readhit} + N_{writemiss} \times \left(E_{writemiss} + E_{writehit}\right) + N_{writehit} \times E_{writehit} \tag{4}$$

$$C_{CACHE} = \frac{2 * E_{CACHE}}{V_{dd\,CACHE}^{\,2}} \tag{5}$$

Em as energias por acesso ($E_{readmiss}$, $E_{writehit}$, $E_{writemiss}$ e $E_{readhit}$) são fornecidas pelo CACTI, os números de acessos ($N_{readmiss}$, $N_{readhit}$, $N_{writemiss}$ e $N_{writehit}$) são resultados dos simuladores e a tensão de almientação da CACHE, $V_{ddCACHE}$ é calculada pelo CACTI.

## 3.4 Modelo de energia e desempenho da memória principal

Após uma ampla pesquisa, conclui-se que não há nenhuma ferramenta de software que estime a energia consumida nem o tempo de acesso de uma memória principal. Por isso e seguindo a recomendação do artigo [11], optou-se por se escolher no mercado uma memória e retirou-se os referidos dados do catalogo do fabricante.

A memória escolhida foi a HYB 39L256160AC-8, trata-se de uma memória RAM de 256-Mbit com 22 bits de endereçamento, e 16 de dados, que funciona a 3.3V e optimizada para sistemas móveis.

Os dados do catalogo que se considerou relevantes para o modelo de desempenho foram os seguintes:

- Corrente consumida pela memória em modo de espera
- Corrente consumida pela memória em modo de operação
- Largura dos barramento de endereços e de dados
- Capacidade dos barramentos de dados, endereços, pinos de controlo e de sinal de relógio.
- Tensão da fonte de alimentação e das linhas da memória.

O programa criado para calcular a potência consumida pela memória começa por determinar o tamanho do Burst, ou seja por o numero de palavras (words) que são lidas em acesso da memória CACHE à memória principal.

$$burst = \frac{number\_of\_sets_{CACHE}}{number\_of\_bits\_stored_{MEM}} \quad (6)$$

De seguida, determina-se o numero de ciclos de memória ocorridos durante a execução do programa:

$$Ncycles_{MEM} = Ncycles_{CPU} \times \frac{f_{MEM}}{f_{CPU}} \quad (7)$$

Assumindo que, no que diz respeito ao numero de transições no barramento entre a CACHE e a memória, as estatísticas são idênticas às do barramento entre o CPU e a memória CACHE, calcula-se a capacidade associada ás linhas de acesso à memória principal:

$$C_{address\_bus} = P_{address}(transition) \times burst \times addresslength \times C_{address} \times cache\_misses \quad (8)$$

$$C_{data\_bus} = P_{data}(transition) \times burst \times databuslength \times C_{data} \times cache\_misses \quad (9)$$

$$C_{CLK} = 2 * Ncycles_{MEM} \times C_{CLK\_line} \quad (10)$$

$$C_{MEM\_LINES} = C_{CLK} + C_{data\_bus} + C_{address\_bus} \quad (11)$$

Por outro lado temos ainda a energia consumida pela fonte alimentação vem dada pela equação:

$$E = \int V.Idt \quad (12)$$

Particularizando, calcula-se a energia consumida pelo simples facto da memória estar ligada durante a execução do programa:

$$E_{static} = V_{DD} \times I_{nop} \times \frac{Ncycles_{MEM}}{f_{MEM}} \quad (13)$$

Ao qual se acrescenta a parcela consumida durante os acessos à memória, obtem-se a energia que a fonte de alimentação fornece à memória

$$E_{dynamic} = V_{DD} \times (I_{op} - I_{nop}) \times \frac{(N_{read} \times (latency + burst) + N_{write} \times burst)}{f_{MEM}} \quad (14)$$

Note-se que a terceira parcela desta multiplicação, à semelhança do que ocorre na equação da energia estática, representa o tempo, neste caso o tempo durante o qual a memória está activa. A parcela $N_{read}$ x (latency+burst) representa o numero de ciclos de

relógio durante o qual a memória está em modo de leitura, $N_{write}$ x burst, o numero de ciclos de relógio em que a memória esteve a receber informação (a escrever). Para converter a soma destes números de períodos de relógio para unidades de tempo é preciso multiplicar pelo período de relógio da memória, que é equivalente a dividir pela frequência do mesmo.

Finalmente, somando

$$E_{MEM\_SUPPLY} = E_{static} + E_{dynamic} \tag{15}$$

Para obter a energia total consumida pela memória é então necessário somar a energia consumida nas linhas de interface da memória com aquela que a fonte de alimentação teve de fornecer directamente à memória.

$$E_{MEM} = E_{MEM\_SUPPLY} + \frac{V_{dd,mem} \times \left(C_{address\_bus} + C_{data\_bus} + C_{CLK}\right)}{2} \tag{16}$$

## 3.4 Modelo de energia dos barramentos.

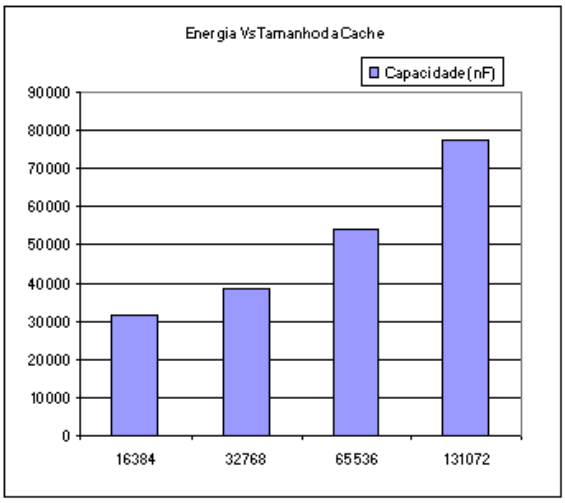
No modelo considerado, existem dois barramentos, um entre o CPU e a memória CACHE e outro entre a memória CACHE e a memória principal.
Assim, temos simplesmente:

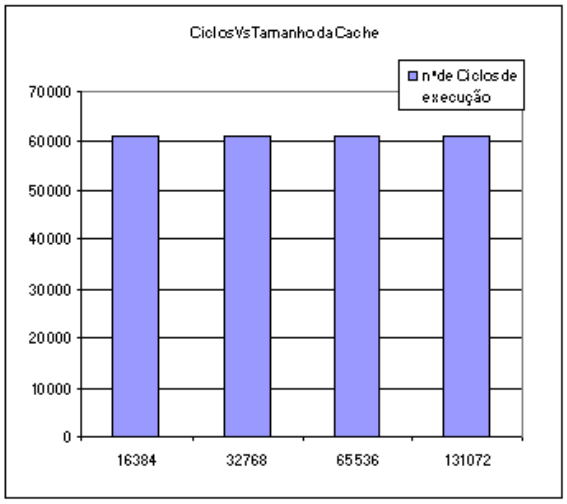
$$C_{CACHE\_BUS} = Total\_Transitions \times C_{CACHE\_LINE}$$
$$C_{MEM\_BUS} = C_{MEM\_LINE} \times N_{misses} \times \left(P_{address}(transition) \times address\_length + P_{data}(transition) \times data\_length\right) \tag{17) (18}$$

Nota: os parâmetros Total_Transitions, $P_{address}(transition)$ e $P_{data}(transition)$ são resultados da simulação, o segundo representa a probabilidade de ocorrer uma transição numa linha.

## 3.5 Resultados

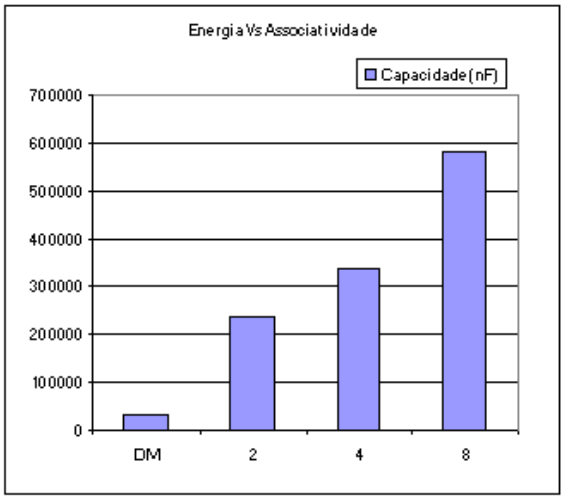
Alguns resultados obtidos com este modelo, quando se simulou um programa muito simples, que consistia num contador com módulo 1000.

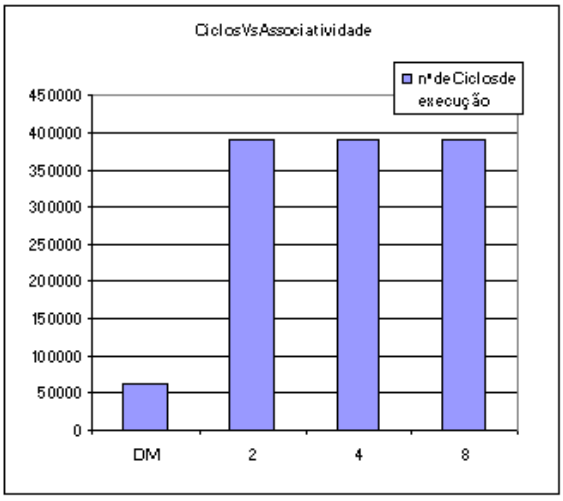



Tamanho da Cache,

Dado que este programa não precisa de um tamanho de memória muito amplo, o tamanho da não influenciou nada a não ser na energia. De facto quanto maior é uma memória cache, mais energia consome.




A lógica inerente à associatividade da Cache é relativamente pesada, quando está presente, prejudica muito o tempo de acesso da cache. Neste programa, a inclusão de um nível de associatividade diminuiu muito o numero de faltas no acesso a cache, o que é um melhoramento irrelevante visto que o desempenho e o consumo de energia saíram muito prejudicados.

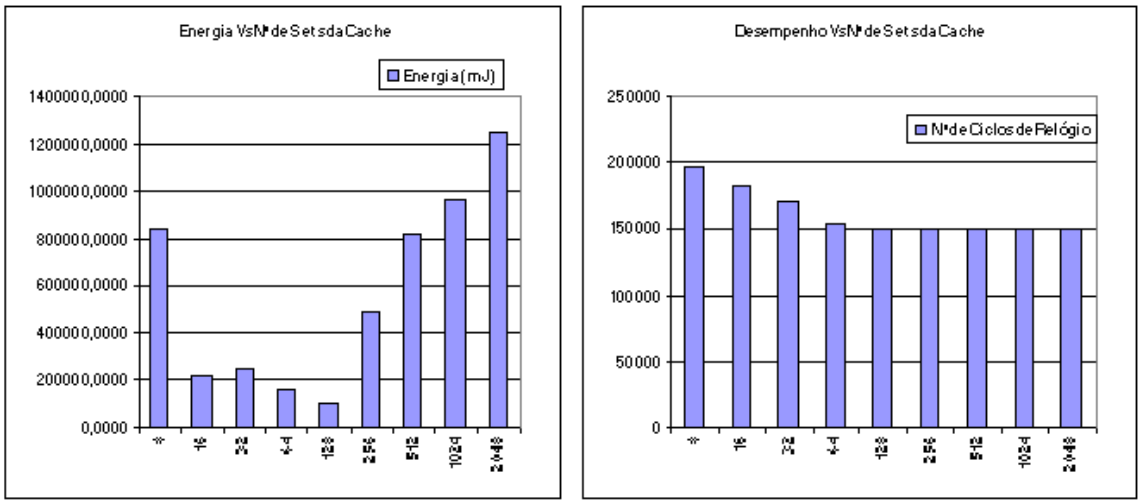
## 4 Modelo a ser Desenvolvido

Contudo, o modelo anterior demonstrou ser lenta, imprecisa e impraticável quando se simula a execução de programas não simples, por isso procurou-se alternativas.

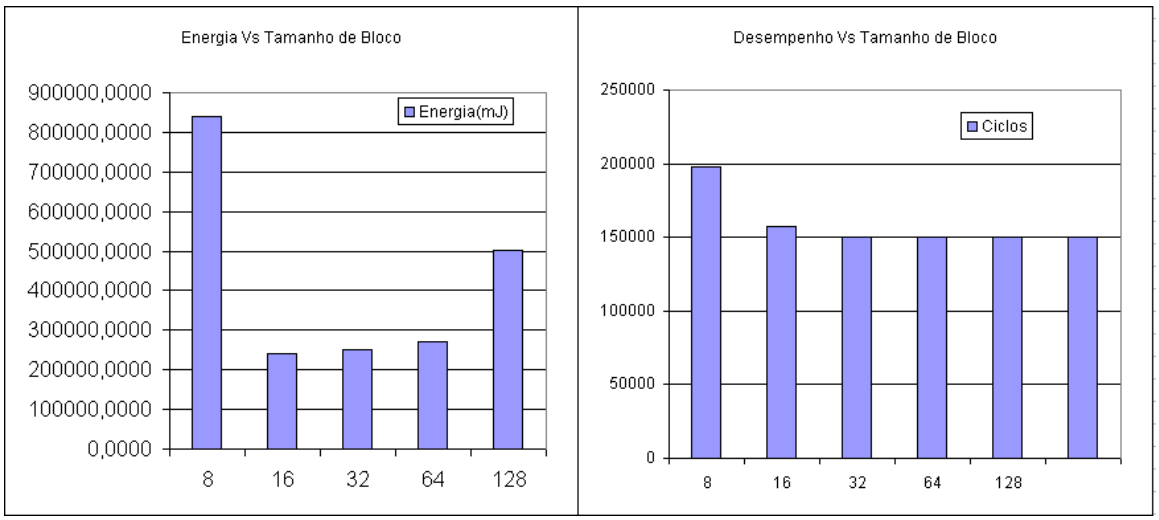
A grande diferença entre o modelo inicialmente implementado e o novo é que baseiam-se em simuladores diferentes. Enquanto que o modelo anterior partia do simulador SimplePower, o novo parte do Wattch. Ambos são extensões do SimpleScalar com estimação de energiar e partilham as mesmas ferramentas para gerar executáveis, mas a grande diferença entre os dois é o simulador de CACHE: Enquanto que o SimplePower utiliza o simulador de CACHE Dinero, o Wattch utiliza o simulador de CACHE de origem do SimplePower, que mostrou ser mais estável, fiável e sobretudo rápido.

Simulações com Wattch (Complementado com o simulador de CACHE do SimpleScalar):

1º - Simulação de um programa de ordenação de uma lista de 100 elementos com algoritmo de ordenação tipo BubbleSort.



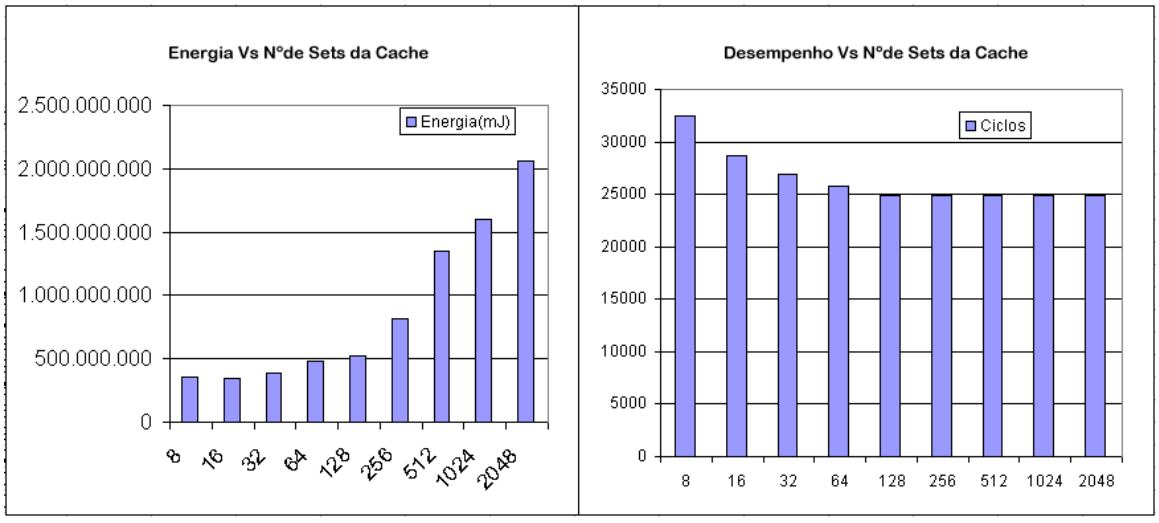

Este Modelo permite fazer simulações mais interessantes, por exemplo, esta simulação mostra claramente que o numero de Sets da CACHE ideal para executar este programa é 128. Uma CACHE maior não traz beneficios do ponto de vista de desempenho e o consumo de energia aumenta bastante. CACHES mais pequenas embora, por si próprias consumam menos energia, no global o consumo de energia é maior, pois devido à taxa de faltas ser maior, o tempo de execução aumenta (o desempenho é mais fraco).




O aumento do tamanho de Bloco da CACHE não trouxe vantagem nenhuma, visto que mantendo constante o numero de Sets, nenhum dos ensaios com um tamanho de bloco maior, trouxe beneficios à energia/desempenho, relativamente ao caso anterior.

Mesmo utilizando o melhor dos dois casos, ou seja, um numero de Sets igual a 128 e o tamanho de bloco de 16 bytes, a energia obtida foi 283719.2 mJ, que é muito maior do que os 105936.5 mJ obitdos com o tamanho de bloco igual a 8 bytes. O numero de ciclos de relógio para ambos os ensaios foi sensivlemente igual: 149840 com um tamanho de bloco de 16 bytes e 150084 com um tamanho de bloco de 8 bytes.

Ensaios de um programa de ordenação de listas mas utilizando um algoritmo de ordenação do tipo QuickSort:




Estes ensaios mostram bem que nem sempre a optimização do Desempenho e do Consumo de Energia são compatíveis. Enquanto que do ponto de vista da energia o número de Sets ideal é 16, o melhor desempenho (rapidez de execução) só é conseguido utilizando uma Cache com mais de 128 Sets.

Este modelo, baseado no Wattch terá de ser complementado com modelos de memória principal e de barramentos para servir os propósitos deste projecto que são o de estimar e optimizar o consumo de energia em microprocessadores com os respectivos periféricos.