

Data Memory Design and Exploration for Low Power Embedded Systems

Wen-Tsong Shiue,
Sathishkumar Udayanarayanan,
Chaitali Chakrabarti
Arizona State University, Tempe, Arizona

Categories and Subject Descriptors: B.3.2 [Memory Structures]: Design styles - cache memory; B.3.3 [Memory Structures]: Performance analysis and design aides - simulation; D.3.4 [Programming Languages]: Processors - memory management.
General Terms: Algorithms, Design, and Performance.

Abstract - In embedded system design, the designer has to choose an on-chip memory configuration that is suitable for a specific application. To aid in this design choice, we present a memory exploration procedure based on three performance metrics, namely, cache size, the memory access time and the energy consumption. We show the importance of including energy in the performance metrics, since an increase in the cache size and line size reduces the memory access time but does not necessarily reduce the energy consumption. The memory exploration procedures enable us to find the cache configuration (cache size, line size) that satisfies the area and time constraints while minimizing the energy consumption, and the cache configuration that satisfies the area and energy constraints while minimizing the memory access time. The exploration procedures for cache configuration is very efficient since it considers only a selected set of candidate points. Finally, we validate our exploration procedures by running simulation experiments on MediaBench applications.

1. INTRODUCTION

In systems that involve multidimensional streams of signals such as images or video sequences, it has been shown that the majority of the area and power cost is not due to the datapath or controllers but due to the global communication and memory interactions [Cathoor et al. 1994; Cathoor et al. 1998]. In fact, in embedded applications for real-time signal processing, 50-80% of the power cost is due to memory traffic caused by transfers between the ASIC and the off-chip memories. Even general-purpose processors such as the 21164 DEC Alpha chip or the StrongArm SA-110 processor dissipate 25-30% of the total power in the cache. Clearly, in order to reduce the system-level power consumption, it is very important to focus on design strategies that would reduce the power consumption due to memory traffic.

In embedded processor-based design, the processor core is decoupled from the on-chip memory, and it is upto the system designer to choose an on-chip memory configuration that is suited for a particular application. While traditionally the system constraints have been area and execution time, in embedded systems used in low power or battery operated devices, energy is an equally important design constraint. To aid the system designer explore the memory design space efficiently, a good exploration strategy is clearly needed. In this paper, we present an efficient exploration strategy for determining an on-chip data memory

architecture that satisfies the system requirements of cache size, the average memory access time (which is related to the execution time) and the energy consumed in the memories. We add energy to the traditional performance metrics, since the variation in the memory access time is quite different from the variation in the energy consumption. The proposed procedure chooses the cache configuration (line size and cache size) that satisfies the system constraints. The exploration procedure is very efficient since it looks at only a subset of candidate points. In this paper, we focus only on the data cache exploration (the instruction cache size is not varied). We also assume that the on-chip cache and off-chip memories are single port, single module memories. Finally, we validate the exploration procedure by performing simulation experiments on a variety of applications including image and video coders, speech and audio coders, etc. from the MediaBench suite of applications. We have used the cache simulator in the SimpleScalar tool set to aid in the exploration. The work presented in this paper is an extension of our earlier work in [Shiue and Chakrabarti 1999a; Shiue and Chakrabarti 1999b]. The main contributions of this paper are

- Analyzed the differences between the variation in the memory access time and the variation in the energy for different cache configurations. This helped justify the inclusion of energy in the performance metrics.
- Developed an energy model that emulates the model in [Kamble and Ghose 1997] but is simpler.
- Developed memory exploration procedures that determine the minimum energy cache configuration that satisfies the area and access time bound and the minimum access time cache configuration that satisfies the area and energy bound. These procedures consider only a selected set of candidate points, thereby reducing the search time significantly.

Memory optimization for embedded systems has been addressed extensively in [Panda et.al. 97a, Panda et.al. 97b, Panda et.al. 99]. The performance metrics of the system are data cache size and number of processor cycles. In addition, an excellent method for off-chip data placement is proposed that reduces the number of conflict misses significantly. In [Dutta et.al. 98], memory system design for video processors has been studied; the performance metrics are area, cycle time and utilization.

In recent years, the focus has been on system-level based approaches that consider the interdependencies between the different system parts. For instance, in [Kirovski et.al. 97], a heuristic procedure to select an area-minimal processor core and cache configuration that satisfies the performance requirements (in terms of the hit rate) has been proposed. Different data and instruction cache sizes for fixed line size and associativity have been considered. In [Li and Henkel 98], tradeoffs between energy and performance for a system with CPU and caches has been studied. Different sizes and associativities of the instruction and data cache have been considered. In [Givargis et.al. 99], power, performance and area tradeoffs have been studied for a system comprising of not only the CPU and caches but also the bus interfaces. An analysis of the results shows that the CPU-to-cache bus size is the major component in determining the system's power

and execution time metrics. Also, with deep submicron technologies, the interface power would be dominant. Finally, cycle-accurate simulation of energy dissipation in embedded systems have recently been proposed in [Simunic et.al. 99][Ye et.al. 00]. These simulators will certainly enhance the study and analysis of system-level energy-performance trade-offs.

The rest of the paper is organized as follows. Section 2 describes the access time and energy models used in our procedure. Section 3 describes how the memory access time and energy consumption vary for different cache configurations. Section 4 describes the efficient memory exploration procedures. Section 5 concludes the paper.

2. PRELIMINARIES

2.1 Experimental Set Up

Seven different applications from MediaBench [Lee et.al 1997] are simulated and their characteristics are analyzed. The applications considered are: EPIC, JPEG, and MPEG2, which are image and video coders, GSM, G.721, and ADPCM, which are speech and audio coders, and PEGWIT, which is a public key encryption algorithm. Simulations are performed using the cache simulators available in SimpleScalar tool release 2.0 [Austin et.al. 2000]. The simulator *sim-cache* is used for the memory exploration procedure. Only the L1 data cache is considered. The cache is assumed to be direct-mapped. Line sizes of 16 bytes to 128 bytes and cache sizes of 1KB to 128KB are considered. The simulator *sim-outorder* is used to compute the exact number of instruction cycles. The instruction cache is set to the default value in all the simulations.

2.2 Performance Metrics

In this section we describe the three performance metrics of our memory exploration system, namely, cache size, average (memory) access time and energy consumed in the memories.

2.2.1 Cache Size

Given the area constraint, we find the largest possible cache size C_{\max} that satisfies this constraint. The memory exploration procedure searches for the best cache configuration among cache sizes $\leq C_{\max}$. The cache sizes are chosen to be powers of 2. If the cache size, C , is increased or if the number of cache lines, L , is increased, the miss rate is reduced. While the reduction in miss rate directly translates to reduction in the average memory access time, it does not necessarily result in reduction of energy consumption. Thus choosing the largest cache size is not necessarily the best solution. Performance trade-offs have to be studied before choosing the best cache configuration.

2.2.2 Average Memory Access Time

The average memory access time is defined by

$$\text{Average_memory_access_time} = \text{hit_time} + \text{miss_rate} * \text{miss_penalty}$$

We adopt the model used in [Hennessey and Patterson 1996] and assume that the hit_time or the number of cycles per hit is 1, and that the miss_penalty or the number of cycles per miss is 42, 44, 44 and 48 for line sizes of 16, 32, 64 and 128 respectively. Thus increasing the line size reduces the miss_rate but increases the miss_penalty.

The execution time (in cycles) is another important performance metric. If the execution time of a task assuming no cache misses is known, then the execution time (in cycles) is given by [Li and Henkel 98].

$$\begin{aligned} \text{Cycles} = & \text{Cycles_w/o_cache} + \text{N_miss_rd} * \text{read_penalty} + \text{N_miss_wr} * \text{write_penalty} \\ & + \text{N_miss_fetch} * \text{fetch_penalty} \end{aligned}$$

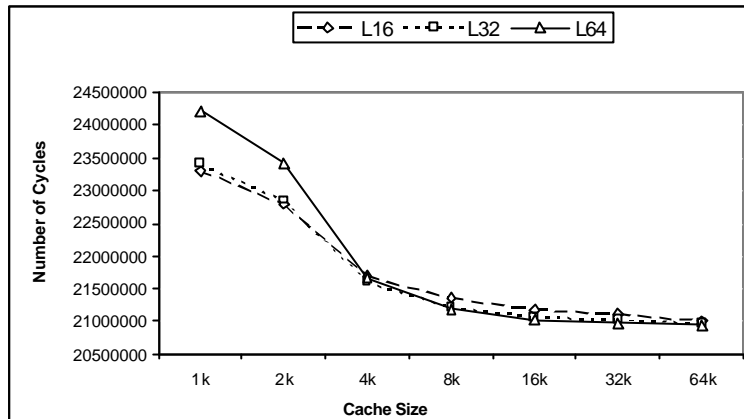
where Cycles_w/o_cache is the execution time of a program running on the processor core without cache misses, N_miss_rd is the number of read misses, N_miss_wr is the number of write misses, N_miss_fetch is the number of instruction fetches, read_penalty is the number of cycles for a read miss, write_penalty is the number of cycles for a write miss and fetch_penalty is the number of cycles for an instruction cache miss.

For a fixed size instruction cache, the term N_miss_fetch * fetch_penalty does not change with the data cache configuration and can be considered as a constant. If we assume that for the data cache, the read and write penalties are the same, then the number of cycles can be approximated to

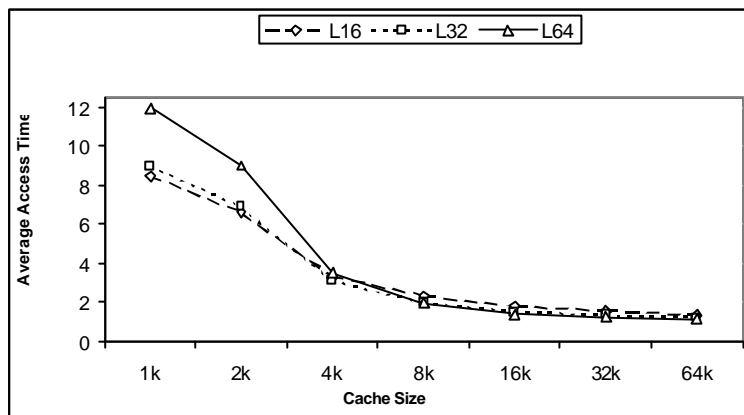
$$\text{Cycles} = \text{Cycles_w/o_cache} + \text{N_misses} * \text{miss_penalty} + \text{constant},$$

where N_misses is the sum of read and write misses.

If the application is memory-intensive, then it is reasonable to assume that the actual number of cycles and the memory access time are related. Figure 1(a) shows a plot of the number of cycles for different cache and line sizes for the JPEG encoder. The number of cycles was obtained using *sim-outorder* simulator from SimpleScalar. Figure 1(b) shows the plot of the average memory access time for the same example. These two plots indicate that the trend in the curves for different cache and line sizes is very similar. This is a trend that we observed in all the examples that we looked at. This leads us to conclude that the trade-offs between average memory access time, energy and cache size are very similar to the tradeoffs between number of cycles, energy and cache size. Since the amount of time that it takes to run the cache simulator (*sim-cache*) is a fraction of the time (<10%) that it takes to run a timing simulator (*sim-outorder*), in the rest of this paper we have used the average memory access time as the performance metric rather than the number of execution cycles.



(a)



(b)

Figure 1. JPEG encoder. (a) Number of cycles and (b) Average memory access time for different cache sizes and line sizes.

2.3 Energy Consumed in the Memories

There exist several energy models for caches [Cattloor et al. 1998; Kamble and Ghose 1997; Wilton and Jouppi 1994; Su and Despain 1995; Kin et al. 1997]. The model that we have used here is based on the analytical model developed by [Kamble and Ghose 1997]. Their model combines memory traffic, process features such as capacitance, and architectural factors including cache line size, set associativity, and capacity. The process models are based on measurements reported by [Wilton and Jouppi 1994] for a 0.8 μm process technology. An intuitive and simple model for power consumption due to hits in a cache has been developed by [Su and Despain 1995]. While the energy model in [Kamble and Ghose 1997] is comprehensive and exact, the energy model in [Su and Despain 1995] achieves faster estimation at the expense of reduction in accuracy. We have combined the energy models in [Kamble and Ghose 1997] and [Su and Despain 1995] to develop a model that leads to fast estimation as in [Su and Despain 1995] and yet matches the energy values in [Kamble and Ghose 1997] quite closely. Furthermore, we not only consider the energy consumed in the host

processor but also consider the energy consumed in the off-chip memory. The energy model presented here is more accurate than that used in our earlier work [Shiue and Chakrabarti 1999a].

In our energy model, the total energy is given by $\text{Energy} = E_{\text{dec}} + E_{\text{cell}} + E_{\text{io}} + E_{\text{main}}$, where the E_{dec} is the energy in the decoder block, E_{cell} is the sum of the energy in the cell arrays, E_{io} is the energy consumed in the address pads and data pads of the host processor, E_{main} is the energy consumed in off-chip memory. E_{main} consists of two components – the energy consumed in the address and data pads in the off-chip memory and the energy of accessing the off-chip memory.

The energy model is based on those cache components that dominate overall cache power consumption. For instance, in the address decoding path, the capacitance of the decoding logic is less than that of the address bus, and so we consider E_{dec} to be only the energy consumed in the address buses. Similarly, in cell arrays, we consider E_{cell} to be the energy consumed by the pre-charged cache word/bit lines. During a miss, the dominant components are the energy consumed in the I/O pads of the host processor and off-chip memory and the energy consumed in accessing the off-chip memory. For our experiments, we have used the SRAM CY7C1326-133 from Cypress as our main memory. The SRAM is of size 2M bits, has an access time of 4ns, voltage of 3.3V, current of 375 mA, and has energy consumption of 4.95 nJ per access. While we choose an SRAM for our system, a DRAM (equipped with an energy model) would be sufficed. In summary,

Energy = Edec+Ecell+Eio+Emain

- $E_{\text{dec}} = \alpha * (\text{Add_bus_bs}) * (C/L)$
- $E_{\text{cell}} = \beta * (\text{Word_line_size}) * (\text{Bit_line_size} + 4.8) * (\text{Nhit} + \text{Nmiss})$
- $E_{\text{io}} = \gamma * (\text{Data_pad_bs} * 8L + \text{Add_pad_bs})$
- $E_{\text{main}} = \gamma * (\text{Data_pad_bs} * 8L + \text{Add_pad_bs}) + E_m * 8L * \text{Nmiss}$

and

- $\text{Add_bus_bs} = \text{Pr}_1 * (\text{Nhit} + \text{Nmiss}) * \text{Wadd}$
- $\text{Add_pad_bs} = \text{Pr}_1 * (\text{Nmiss}) * \text{Wadd}$
- $\text{Data_pad_bs} = \text{Pr}_2 * (\text{Nmiss})$
- $\text{Word_line_size} = m * (8L + T + \text{st})$
- $\text{Bit_line_size} = C / (m * L)$
- $\alpha = 7.89e-17; \quad \beta = 1.44e-14; \quad \gamma = 5.45e-11; \quad E_m = 4.95e-9 \text{ J}$

Where

- C = cache size.
 - L = cache line size.
 - m = m-way set associative cache.
 - T = tag size of T bits
 - St = number of status bits per block frame.
 - Pr_1, Pr_2 = the probability of a 0 to 1 transition
 - Nhit = number of hits.
 - Nmiss = number of misses.
 - Wadd = the width of address bus.
 - Add_bus_bs = number of bit switches on address bus.
 - Add_pad_bs = number of bit switches on address pads.
 - Data_pad_bs = number of bit switches on data pad.
 - Word_line_size = Number of memory cells in a word line.
 - Bit_line_size = Number of memory cells in a bit line.
 - E_m = Energy consumption of a main memory access
- α, β and γ are used for 0.8 μm CMOS technology

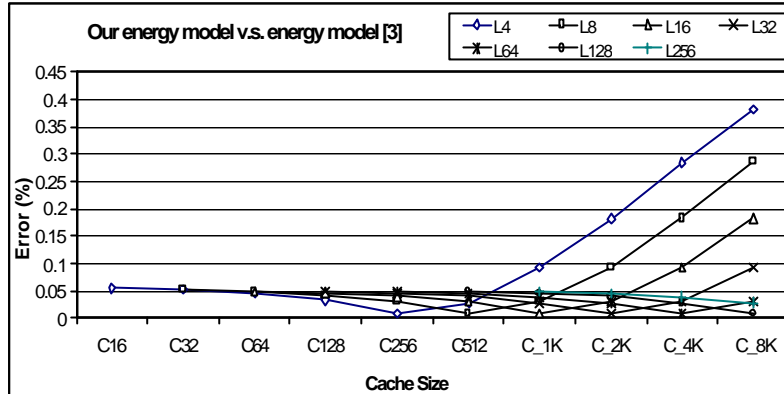


Figure 2. Example Compress. Errors for different cache sizes and line size.

In our experiments, we assume Pr_2 is 0.25, since it is very unlikely that more than $\frac{1}{4}$ of the data bits will undergo transitions. Since the size of main memory is 2Mbits, W_{add} is $\log(2M/8) = 18$ bits. The tag size is calculated as $W_{add} - \log(C) - \log(m)$, where C is the cache size and m is the degree of set associativity. As mentioned earlier, our energy model results in very fast estimates. This was made possible by simplifying the expressions used in [Kamble and Ghose 1997]. Figure 2 plots the % error of our model with respect to [Kamble and Ghose 1997] for different cache and line sizes. A maximum error of 0.38% indicates that our model is very close to that of [Kamble and Ghose 1997].

3. TRADEOFFS BETWEEN ENERGY CONSUMPTION AND THE AVERAGE MEMORY ACCESS TIME

In this section, we describe the differences between the variation in the energy consumption and the variation in the average memory access time for different cache configurations. It is this difference in the variation that motivated us to include energy consumption in the performance metrics. We first describe the miss rate variation since it helps us analyze the variations in the memory access time and energy consumption.

Variation in the miss rate: For a specific line size, as the cache size increases, the miss rate drops. The absolute value of the miss rate varies from program to program. The drop in the miss rate is low as the cache size increases ($>8K$). For instance, for the ADPCM encoder example shown in figure 3(a), for $L=16$, the miss rate is 0.01454 at $C=4KB$ and drops to 0.001861 at $C=8KB$. For very large cache sizes ($C>8KB$), the miss rate varies very little with increase in the cache size. For the ADPCM encoder example, for $L=16$, the miss rate varies from 0.001190 at $C=16KB$ to 0.001159 at $C=64KB$.

For low cache sizes, the miss rate for $L=16$ is lower than the miss rate for $L=128$. As the cache size increases, the miss rate for $L=16$ may become higher than the miss rate for $L=128$. For instance for the ADPCM encoder example (Figure 3(a)), the cross over happens between $C=4K$ and $C=8K$.

Variation in the average memory access time: The variation in the average memory access time is very closely related to the variation in the miss rate. For a specific line size, as the cache size increases, the average memory access time drops. While the drop in the access time is large for smaller cache sizes, it is very small for larger cache sizes. In fact, the average memory access time curve is flat for large cache sizes. For instance, for the ADPCM encoder (see Figure 3(b)), for $L=16$, the average memory access time saturates at $C=8K$. An interesting point to note is that the average memory access time saturates for all values of L at $C=8K$. This phenomenon is due to the fact that the miss rate for $C=8K$ is so small, that the difference in the $miss_rate * miss_penalty$ term for different values of L is negligible.

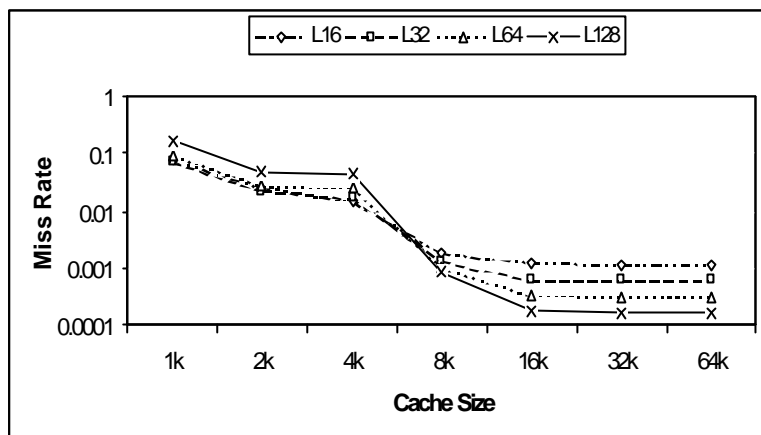
The variation in the average memory access time with line size is very similar to the variation in the miss rate with line size. If there is a cross over in the miss rate plot, there is a similar cross over in the memory access time plot. In Figure 3(b), the cross over is not visible since the access time saturates at $C=8K$.

Variation in the energy consumption: The variation in the energy consumption is different from the variation in the average memory access time. For a specific line size, as the cache size increases, the energy reduces at first and then increases. In other words, there exists a cache size (C_{dip}) for which the energy is minimum. In the ADPCM example (Figure 3(c)), the minimum energy cache size for $L=16$ is $C=8K$. The minimum energy cache size varies with the line size. For instance, for the same example, the minimum energy cache size for $L=64$ is $C=16K$. Note that the value of C_{dip} varies from program to program. Also, the dip in the energy curve may not be apparent if the maximum allowed cache size, C_{max} , is smaller than C_{dip} .

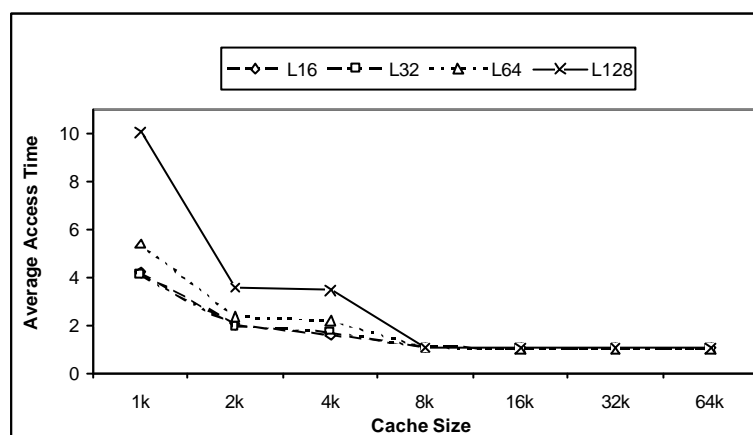
The reason for the dip in the energy curve can be analyzed by revisiting the energy equation. Since the E_{dec} and the E_{io} terms are negligible, the energy equation can be simplified to $E = E_{cell} + E_{main}$, where $E_{cell} = k_1 * (8C + 38L)$ and $E_{main} = k_2 * E_m * L * miss_rate$, where k_1 and k_2 are functions of the number of accesses and a normalization constant. Figure 4 plots E_{cell} , E_{main} and E for different cache sizes for $L=16$ for the ADPCM example. Note that while E_{main} reduces with increase in cache size, E_{cell} increases. For small cache sizes ($<8K$), E_{main} is the dominant term, and E follows E_{main} quite closely. For larger cache sizes, the $miss_rate$ is very small and the E_{cell} term dominates. As the cache size increases ($C > 8K$), the E_{cell} term increases, and E follows E_{cell} . Thus E undergoes a dip at $C=8K$. Another interesting point to note is that as E_m (the energy consumption due to a main memory access) increases, the cache size at which the energy curve dips increases.

The energy consumption increases with the increase in line size for all cache sizes upto where the curves dip. This is to be expected since the dominant term before the dip is due to E_{main} which is proportional to L . In Figure 3(c), the energy curves for different values of L converge after the dip. This is because the E_{cell} term is almost constant for all L , and the variation in the E_{main} term is negligible (due to the variation in the $miss_rate$ being negligible).

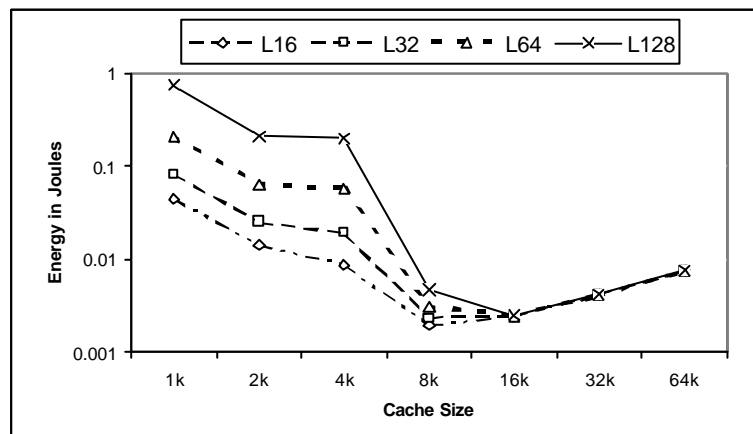
In other examples, however, the variation in the E_{main} term is small but not negligible for large cache sizes. This is because the corresponding miss rates are not that low. As a result, while the difference in the E values for different values of L is small, the energy curves do not converge.



(a)



(b)



(c)

Figure 3. ADPCM encoder. (a) Miss rate, (b) Average memory access time, and (c) Energy consumption for different cache sizes and line sizes.

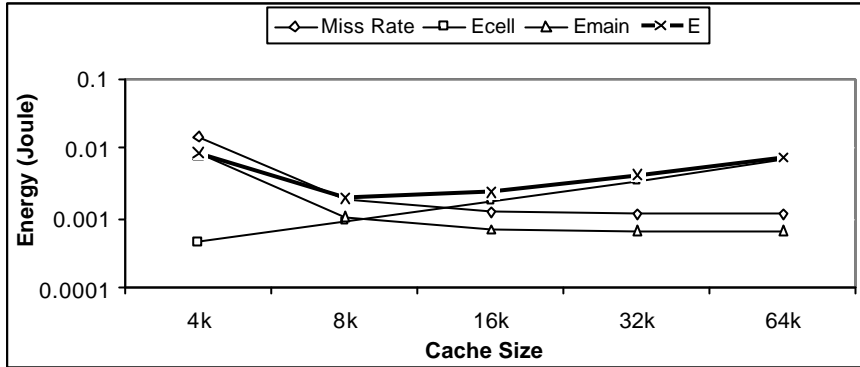


Figure 4. ADPCM encoder. Important components for energy consumption for different cache sizes for $L=16$.

4. SEARCH SPACE REDUCTION

In this section, we present efficient algorithms to find the minimum energy cache configuration given the cache size and average access time bounds, and the minimum access time cache configuration given the cache size and energy bounds. These algorithms do not look at all possible cache configurations but only a small subset of configurations. This reduces the time to search for the 'best' cache configuration significantly. We would like to point out that the exploration algorithms presented here can also be used if the execution time is used as the performance metric (instead of the average access time). In fact, the benefit of using these algorithms becomes more apparent since the time taken to run a timing simulation is significant, and so the reduction in the search space results in significant reduction in the exploration time.

4.1 Minimum Energy Cache Configuration

The algorithm for finding the minimum energy cache configuration given the cache size bound (C_{max}) and the average memory access time bound (AAT_{bound}) first identifies whether C_{max} is smaller or larger than the cache size at which the energy curve dips (C_{dip}). This is important since for cache sizes larger than C_{dip} , the energy monotonically increases and the average access time either monotonically decreases or saturates, while for cache sizes smaller than C_{dip} , both the energy and the average access time monotonically decreases. If $C_{max} < C_{dip}$, the algorithm chooses the smallest line size that meets the access time bound. This is because smaller the line size, smaller the energy. If $C_{max} > C_{dip}$, the algorithm first searches for C_{dip} (since this is not known apriori) corresponding to L_{min} , provided the access time bound is satisfied. If the time bound is not satisfied for L_{min} , it searches for higher line sizes. The algorithm is described in Figure 5. The input parameters are the range of line sizes (L_{min} to L_{max}), the range of cache sizes (C_{min} to C_{max}) and the memory access time bound (AAT_{bound}).

```

Algorithm min_energy ( $L_{\min}$ ,  $L_{\max}$ ,  $C_{\min}$ ,  $C_{\max}$ ,  $AAT_{\text{bound}}$ )
begin
1.    $C1 = C_{\max}$ ;
2.    $C2 = C1/2$ ;
3.    $mr_1 = \text{sim-cache result with } L_{\min}, C1$ ;
4.    $mr_2 = \text{sim-cache result with } L_{\min}, C2$ ;
5.    $aat1 = \text{CModel}(mr_1, L_{\min})$ ;
6.    $aat2 = \text{CModel}(mr_2, L_{\min})$ ;
7.    $E1 = \text{EModel}(mr_1, \text{acc}, L_{\min}, C1)$ ;
8.    $E2 = \text{EModel}(mr_2, \text{acc}, L_{\min}, C2)$ ;
9.    $L = L_{\min}$ ;
10.  while ( $E2 < E1$ ) do
11.      if ( $aat1 \leq AAT_{\text{bound}}$ )
12.           $C1 = C2$ ;
13.           $aat1 = aat2$ ;
14.           $E1 = E2$ ;
15.           $C2 = C1/2$ ;
16.           $mr_2 = \text{sim-cache result with } L, C2$ ;
17.           $aat2 = \text{CModel}(mr_2, L)$ ;
18.           $E2 = \text{EModel}(mr_2, \text{acc}, L, C2)$ ;
19.      else
20.           $L = 2*L$ ;
21.           $C2 = 2*C2$ ;
22.           $C1 = 2*C2$ ;
23.           $mr_1 = \text{sim-cache result with } L, C1$ ;
24.           $mr_2 = \text{sim-cache result with } L, C2$ ;
25.           $aat1 = \text{CModel}(mr_1, L_{\min})$ ;
26.           $aat2 = \text{CModel}(mr_2, L_{\min})$ ;
27.           $E1 = \text{EModel}(mr_1, \text{acc}, L_{\min}, C1)$ ;
28.           $E2 = \text{EModel}(mr_2, \text{acc}, L_{\min}, C2)$ ;
29.      end if
30.  end while
31.  if ( $E2 > E1$ )
32.      if ( $aat1 \leq AAT_{\text{bound}}$ )
33.          solution = ( $L, C1$ );
34.          return solution;
35.      else
36.          Go through all L and choose the smallest L that satisfies
37.           $AAT_{\text{bound}}$ ;
38.      end if
39.  end if
end min_energy

```

Figure 5. Algorithm to find the minimum energy cache configuration given cache size and average access time bounds.

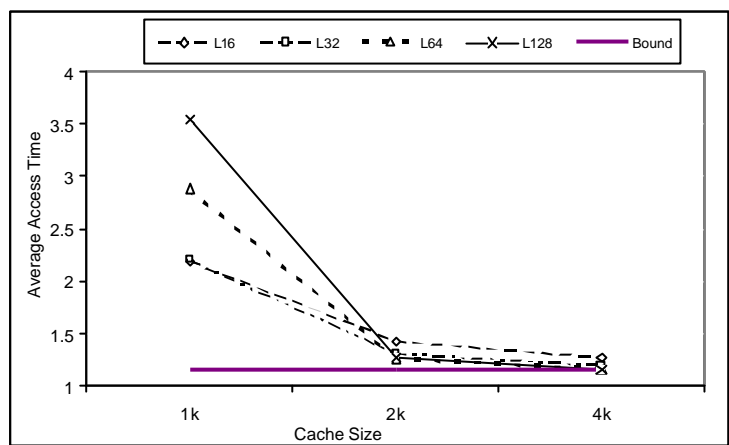
Next we trace the algorithm described in Figure 5.

Lines 1-8: Energy values are computed for cache configuration (L_{\min} , C_{\max}) and (L_{\min} , $C_{\max}/2$). From these energy values, we find if a dip in the energy curve occurs before C_{\max} .

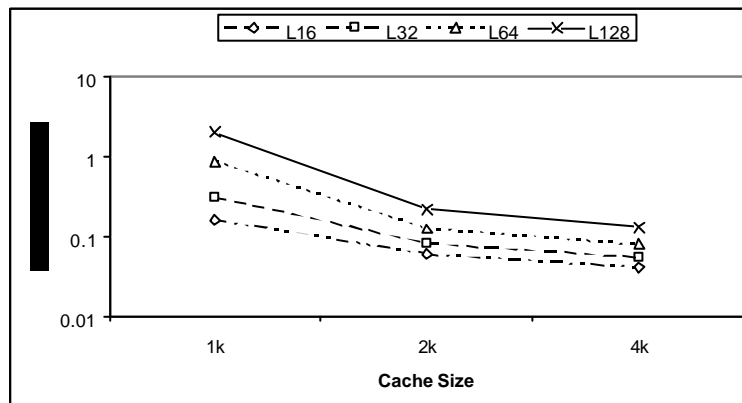
Lines 10-18: If a dip occurs before C_{\max} , then we search for the dip as long as the time bound is satisfied.

Lines 19-30: If a dip occurs before C_{\max} but the time bound is not satisfied at L_{\min} , then we search for a solution with higher line size. The cache size is multiplied by 2 so that the previous cache size is considered for the new higher line size. It is not necessary to check from C_{\max} since if the time bound is satisfied for line size $L1$, then it is satisfied for all $L > L1$. We do need to check from the next higher cache size since the cache size at which the dip occurs could vary by a factor of 2 between two neighboring line sizes.

Lines 31-38: If a dip does not occur before C_{\max} , then we choose the smallest line size that meets the time bound. For all the examples that we studied, the energy curve dips at cache size $>8K$ (if at all), and so for most of the cases, we can expect the algorithm to spend most of its time in lines 31-38.



(a)



(b)

Figure 6. GSM decoder. (a) Average memory access time and (b) Energy consumption for different cache sizes and line sizes.

Complexity Analysis: Let n_l be the number of line sizes and let n_c be the number of cache sizes considered. We measure the complexity on the basis of the number of simulations that we have to run.

Lines 1-8: 2.

Lines 10-30: Let k be the number of cache sizes above the dip. If $k > n_L$, $k - 2 + 2n_L$ simulations are run in the worst case. If $k \leq n_L$, then $n_L - 2 + 2k$ simulations are run in the worst case.

Lines 31-38: All the line sizes are decreased, so n_L , is the worst case. The total complexity, in the worst case, is $n_L + 2$ if C_{max} is below the cache size that causes the dip, and $\max\{k, n_L\} - 2 + 2 * \min\{k, n_L\}$ if C_{max} is above the dip.

Example: We illustrate the minimum energy algorithm using the GSM Decoder example (see Figure 6). The AAT_{bound} is 1.16. C_{min} is 1KB, C_{max} is 4KB, L_{min} is 16 and L_{max} is 128. Initially simulations are performed for cache sizes 4KB and 2KB with line size 16. Since E_2 is greater than E_1 , the search is restricted to cache sizes smaller than C_{dip} . Since aat_1 does not meet the AAT_{bound} , we look for the smallest line that meets the AAT bound. Simulations are performed for line sizes of 32 and 64 for $C=4KB$. Line size of 64 is the smallest line size that meets the AAT_{bound} . So the solution is $(L=64, C=4KB)$. A total of 4 simulations are required in this case. Note that 12 simulations would have been required for an exhaustive search.

4.2 Minimum Memory Access Time Cache Configuration

The algorithm for finding the minimum average memory access time cache configuration given the cache size bound and the energy bound (E_{bound}) also begins by identifying whether C_{max} is smaller or larger than C_{dip} . If $C_{max} < C_{dip}$, the algorithm chooses L_{min} if cross over has not occurred, and chooses the largest line size that satisfies the energy bound if cross over has occurred. Recall that while the minimum energy cache configuration always occurs with L_{min} , it is not true for the minimum access time configuration. If $C_{max} > C_{dip}$, and the energy bound is not satisfied at C_{max} , the algorithm tries to find a smaller cache size (corresponding to L_{min}) at which the bound is satisfied. The algorithm is described in Figure 7. The input parameters are the range of line sizes (L_{min} to L_{max}), the range of cache sizes (C_{min} to C_{max}) and the energy bound (E_{bound}).

Next we trace the algorithm described in Figure 7.

Lines 1-8: Find if the dip in the energy curve occurs before C_{max} .

Lines 9-17: If the cache size is above the cache size that causes the energy dip and if the energy bound is not satisfied, look for a lower cache size where the bound is satisfied.

Lines 18-20: Run the simulation for maximum allowed line size.

Lines 21-27: If L_{min} has the smaller miss rate choose it as the solution. If the energy bound is not satisfied at L_{min} , it will not be satisfied anywhere else (Lines 25-26).

Lines 28-30: If L_{max} has the smaller miss rate, then choose the largest L that meets the energy bound.

Complexity Analysis:

Lines 1-8: 2.

Lines 9-17: k simulations are required, where k is the number of cache sizes beyond dip.

Lines 18-20: 1.

Lines 28-29: $2n_L$, if C_{max} is after the dip; n_L , if C_{max} is before the dip.

The worst case complexity is n_L+3 , if C_{max} is before the dip, and $k+2n_L+3$, if the cache size is above the dip.

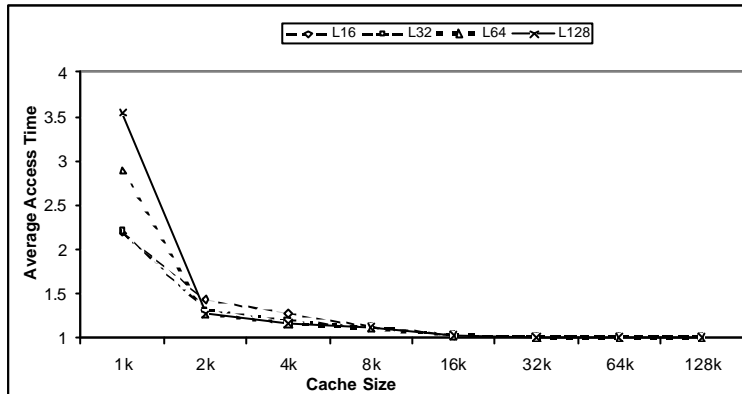
```

Algorithm min_aat( $L_{min}$ ,  $L_{max}$ ,  $C_{min}$ ,  $C_{max}$ ,  $E_{bound}$ )
begin
1.    $C1 = C_{max}$ ;
2.    $C2 = C1/2$ ;
3.    $mr_1 = \text{sim-cache result with } L_{min}, C1$ ;
4.    $aat1 = \text{CModel}(mr_1, L_{min})$ ;
5.    $E1 = \text{EModel}(mr_1, \text{acc}, L_{min}, C1)$ ;
6.    $mr_2 = \text{sim-cache result with } L_{max}, C2$ ;
7.    $aat2 = \text{CModel}(mr_2, L_{min})$ ;
8.    $E2 = \text{EModel}(mr_2, \text{acc}, L_{min}, C2)$ ;
9.   while ( $(E2 < E1) \ \&\& \ (E1 > E_{bound})$ ) do
10.     $C1 = C2$ ;
11.     $aat1 = aat2$ ;
12.     $E1 = E2$ ;
13.     $C2 = C1/2$ ;
14.     $mr_2 = \text{sim-cache result with } L, C2$ ;
15.     $aat2 = \text{CModel}(mr_2, L)$ ;
16.     $E2 = \text{EModel}(mr_2, \text{acc}, L, C2)$ ;
17.  end while
18.   $mr_2 = \text{sim-cache result with } L_{max}, C1$ ;
19.   $aat2 = \text{CModel}(mr_2, L_{max})$ ;
20.   $E2 = \text{EModel}(mr_2, \text{acc}, L_{max}, C1)$ ;
21.  if ( $mr_1 < mr_2$ )
22.    if ( $E1 \leq E_{bound}$ )
23.      solution = ( $L_{min}, C1$ );
24.      return solution;
25.    else
26.      Solution is not possible;
27.    end if
28.  else
29.    Find the largest L that meets the  $E_{bound}$ ;
30.  end if
end min_aat

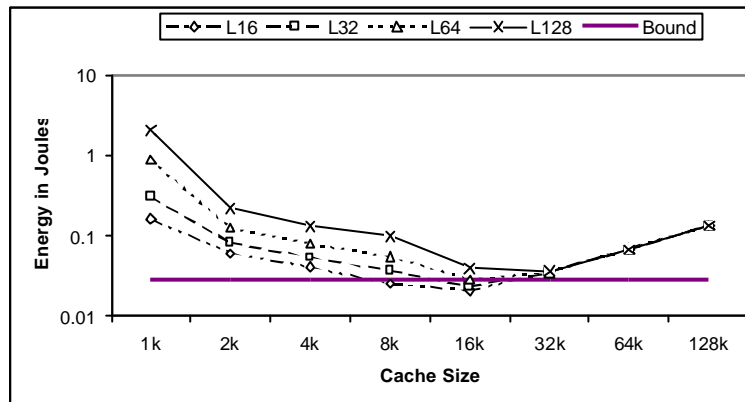
```

Figure 7. Algorithm to find the minimum access time memory configuration given cache size and energy bounds.

Example: We illustrate the minimum memory access time algorithm using the GSM Decoder example (see Figure 8). The energy bound is $3 \cdot 10^{-1} \text{J}$. C_{min} is 1KB, C_{max} is 128KB, L_{min} is 16 and L_{max} is 128. Simulations are performed for cache sizes of 128KB, and 64 KB for a line size of 16. Since $E2 < E1$, C_{max} is larger than C_{dip} . Thus simulations are performed for cache sizes of 32KB, 16KB and 8KB for a line size of 16. The miss rate is computed at line size of 16 for $C=16\text{KB}$. Since $mr_2 < mr_1$, we find the largest line size that meets the E_{bound} . Simulation is performed for line sizes of 128, 64, and 32, with $C=16\text{KB}$. It is found that above the line size of 32, the energy bound is not met. So the solution is $L=32$ and $C=16\text{KB}$. A total of 9 simulations are required using our algorithm. In contrast, an exhaustive search would require 32 simulations.



(a)



(b)

Figure 8. GSM Decoder. (a) Average memory access time and (b) Energy consumption for different cache sizes and line sizes.

5. CONCLUSION

In this paper, we have presented a data memory exploration procedure for low power embedded systems. We have considered both the memory access time and energy in the performance metrics since increasing cache size, and cache line size reduces the miss rate and the average memory access time but does not necessarily reduce the energy. We have considered the average memory access time as the “time” metric since it closely follows the variation in the execution time and is a lot faster to generate. We have developed memory exploration procedures that determine the minimum energy cache configuration that satisfies the area and average access time bound and the minimum average memory access time cache configuration that satisfies the area and energy bound. These procedures consider only a selected set of candidate points, thereby reducing the search time significantly. Our exploration procedure has been validated by performing simulation experiments on the MediaBench examples.

ACKNOWLEDGEMENTS

This work was carried out at the National Science Foundation's State/Industry/University Cooperative Research Center for Low Power Electronics (CLPE). CLPE is supported by the NSF, the State of Arizona and the following companies: Burr Brown, Conexant, Gain Technology, Intel Corporation, Medtronic, Microchip, Motorola, Raytheon, Texas Instruments and Western Design Center.

6. REFERENCES

- AUSTIN T., BURGER D., KECKLER S. 2000. SimpleScalar Simulation Tools for Microprocessor and system evaluation, <http://www.simplescalar.org>.
- CATTHOOR, F., IRANSSEN, F., WUYTACK, S., NACHTERGAELE, L., MAN, H. DE. 1994. Global communication and memory optimizing transformations for low power signal processing systems. *Workshop on VLSI Signal Processing* (La Jolla, CA, Oct).
- CATTHOOR, F., WUYTACK, S., GREEF, E. D., BALASA, F., NACHTERGAELE, L., AND VANDECAPPELLE, A. 1998. Custom memory management methodology – exploration of memory organisation for embedded multimedia system design. *Kluwer Academic Publishers*. (June).
- DUTTA, S., WOLF, W. AND WOLFE, A. 1998. A methodology on evaluate memory architecture design tradeoffs for video signal processors. *IEEE Transactions on Circuits and Systems for Video Technology*, 8, 1 (Feb).
- GVARGIS, T., HENKEL, J., AND VAHID, F. 1999. Interface and cache power exploration for core-based embedded system design. *International Conference on Computer Aided Design*.
- HENNESSY, J. L. AND PATTERSON, D. A. 1996. *Computer architecture a quantitative approach*. 2nd edition Morgan Kaufman Publishers.
- KAMBLE, M. B. AND GHOSE, K. 1997. Analytical energy dissipation models for low power caches. *International Symposium on Low Power Electronics and Design*.
- KIN, J., GUPTA, M. AND MANGIONE-SMITH, W. H. 1997. The filter cache: an energy efficient memory. *30th International Symposium on Microarchitecture*. (North Carolina, November).
- KIROVSKI, D., LEE, C., POTKONJAK, M., AND MANGIONE-SMITH, W. 1997. Application-driven synthesis of core-based systems. *International Conference on Computer Aided Design*, 104-107.
- LEE, C., POTKONJAK, M., AND MANGIONE-SMITH, W.H. 1997. Mediabench: A tool for evaluating and synthesizing multimedia and communications systems . *In Proceedings of the 30th International Symposium on Microarchitecture*.
- LI, Y. AND HENKEL, J. 1998. A framework for estimating and minimizing energy dissipation of embedded HW/SW systems. *35th IEEE/ACM Design Automation Conference*, 188-193.
- PANDA, P. R., DUTT, N. D. AND NICOLAU, A. 1997a. Architectural exploration and optimization of local memory in embedded systems. *International Symposium on System Synthesis*. (Antwerp, Sept).
- PANDA, P. R., DUTT, N. D. AND NICOLAU, A. 1997b. Memory data organization for improved cache performance in embedded processor applications. *ACM Transactions on Design Automation of Electronic Systems*, 2, 4 (Oct).
- PANDA, P. R., DUTT, N. D. AND NICOLAU, A. 1999. Local memory exploration and optimization in embedded systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 18, 1 (January).

- SHIUE, W. -T. AND CHAKRABARTI, C. 1999a. Memory exploration for low power, embedded systems. *36th Design Automation Conference*, (New Orleans, LA, June), 140-145.
- SHIUE, W. -T. AND CHAKRABARTI, C. 1999b. Memory design and exploration for low power, embedded systems. *IEEE Workshop on Signal Processing Systems: Design and Implementation*. (Taiwan R.O.C., Oct).
- SIMUNIC, T., BENINI, L. AND DE MICELI, G. 1999. Cycle-accurate simulation of energy consumption in embedded systems. *36th IEEE/ACM Design Automation Conference* 867-872.
- SU, C. AND DESPAIN, A. 1995. Cache design trade-offs for power and performance optimization: a case study. *International Symposium on Low Power Electronics and Design*, 63-68.
- WILTON, S. E. AND JOUPPI, N. 1994. An enhanced access and cycle time model for on-chip caches. *Digital Equipment Corporation Western Research Lab, Tech. Report 93/5*.
- YE, W., VIJAYKRISHNAN, N., KANDEMIR, M., AND IRWIN, M.J. 2000. The design and use of SimplePower: a cycle-accurate energy estimation tool. *37th IEEE/ACM Design Automation Conference*.

AUTHORS



Wen-Tsong Shiue received his M.S. degree (1991) in Electrical and Computer Engineering Department from Western Michigan University and Ph.D. degree (2000) in Electrical Engineering Department from Arizona State University. From 1991 to 1996, Shiue worked in China Airlines, Taoyuan, Taiwan, R.O.C., where he was engaged in system and electronics design. He also was an instructor in Tamkang University, Taipei, Taiwan, R.O.C., teaching courses in logic design and electronics. From 1996 to 2000, he worked in Arizona State University where his research involved the area of memory design and exploration for low power embedded systems, low power design in high-level synthesis, and low power compiler design. From May 2000 to August 2000, Dr. Shiue worked in Motorola Inc., Austin TX, as a Senior Staff Scientist where his research involved the area of low power scheduling and binding in high-level synthesis. Since August 2000, he has been a Lead Software Engineer/Scientist in Silicon Metrics Corporation in the area of power analysis, accurate power estimation and optimization in VLSI circuits, and EDA tools development.

Dr. Shiue's current research interests include accurate power estimation for CMOS gates, low power memory design, CAD tools for high-level synthesis low power design, low power compiler design, VLSI architectures and algorithms for signal processing and image processing, and parallel processing. Dr. Shiue has authored numerous papers and publication in low power VLSI circuits and system design. He is a member of the IEEE Circuit and Systems Society, IEEE Computer Society and Center for Low Power Electronics (jointly funded by NSF, the State of Arizona and the member companies).



design.

Sathishkumar Udayanarayanan received his B.Eng. degree in Electronics and Communication Engg. from Anna University, Chennai, India in 1996. He received his M.S. in Electrical Engineering from Arizona State University, Tempe, in 2000. He is currently working in the Advanced Processor Design group at Conexant Systems, Newport Beach. His current interests are optimization algorithms, hardware architectures, and low power



faculty of Arizona State University since fall 1990.

Chaitali Chakrabarti is currently an Associate Professor of Electrical Engineering at Arizona State University, Tempe. She received her B.Tech. in Electronics and Electrical Communication Engineering from the Indian Institute of Technology, Kharagpur, India, in 1984. She received her M.S. (1986) and Ph.D. (1990) in Electrical Engineering from the University of Maryland, College Park. She has been on the

Prof. Chakrabarti's research activities in the past have focussed on algorithms and architectures for real-time media processing. Her current work is in the area of high level system design for low power applications including, memory optimization and exploration, high-level synthesis, energy-efficient compilation, and data-dependent algorithm design. She is a member of the Center for Low power Electronics (jointly funded by NSF, the State of Arizona and the member companies).