Analytical Energy Dissipation Models For Low Power Caches*

Milind B. Kamble and Kanad Ghose
Department of Computer Science
State University of New York, Binghamton, NY 13902–6000
email: {kamble, ghose}@cs.binghamton.edu

Abstract

We present detailed analytical models for estimating the energy dissipation in conventional caches as well as low energy cache architectures. The analytical models use the run time statistics such as hit/miss counts, fraction of read/write requests and assume stochastical distributions for signal values. These models are validated by comparing the power estimated using these models against the power estimated using a detailed simulator called CAPE (*CA*ache *P*ower *E*stimator). The analytical models for conventional caches are found to be accurate to within 2% error. However, these analytical models over–predict the dissipations of low–power caches by as much as 30%. The inaccuracies can be attributed to correlated signal values and locality of reference, both of which are exploited in making some cache organizations energy efficient.

Key words: Low–power caches, energy estimation, analytical models for energy dissipation.

1. INTRODUCTION

State of the art microprocessors have one or more levels of on–chip caches [ERB+ 95, Intel 96]. The trend is to increase cache memory with the increasing transistor budget, because off–chip accesses are atleast an order of magnitude slower. By confining memory accesses on–chip, single cycle memory access latency can be achieved using static RAMs. There is the added advantage of reducing power because driving signals through high capacitance I/O pads is less frequent. An examination of the die photos of high–end microprocessors show that anywhere from 15–40% of the die area is dedicated to on–chip caches. Published reports also corroborate the fact that on–chip static RAM caches consume substantial fraction of overall chip power.

(a) The on–chip L1, L2 caches on the DEC 21164 CPU dissipate about 25% of the total chip power [ERB+ 95]

(b) In the bipolar 300MHz. CPU reported in [JBD+ 93], 50% power is dissipated in the primary caches

(c) The StrongARM SA–110 processor from Digital that boasts the best SPECmarks/watt dissipates about 43% power in the caches.

It is thus imperative to seek accurate estimates of cache power as well as implement power reduction techniques in the quest for high performance, low power microprocessors and memories. Power reduction can be achieved by device level improvisation, RAM cell organization improvements or cache architectural improvements. We concentrate on the last one.

To accurately estimate cache power, we develop a model for static RAM cell and identify its main energy dissipating components. We then develop an energy dissipation model for set associative caches based on SRAM cells. We then present architectural improvements to achieve lower energy dissipation without sacrificing access time. The analytical models for estimating cache energy dissipation are then presented based on the energy model for the cache. These require run time statistics of the cache such as hit/miss counts, fraction of read/write requests, number of dirty victims etc. and information about the cache organization such as tag width, line width, cache capacity etc. to derive the signal transition counts in various cache components. We verify the accuracy of the analytical model against results obtained through our simulator called CAPE (*CA*ache *P*ower *E*stimator). CAPE is a detailed register level CPU simulator supporting multiple cache hierarchy and it accurately tracks the *actual* transition counts in the cache components identified in our model. Our analytical model is quite detailed so as to be quite accurate in estimating power in conventional caches. However, we observe that though the analytical method might be the quickest way to estimate power, it generally overestimates when applied to the energy efficient architectures, sometimes by as much as 30%.

The rest of the paper is organized as follows: Section 2 presents the energy model of set associative caches. Section 3 provides energy saving cache organizations. Section 4 discusses the analytical model for estimating transition counts. Section 5 presents our comparative study of the accuracy of analytical model against experimental results. Section 6 presents our conclusions.

2. ENERGY DISSIPATION IN SRAM CACHES

In this section we present the energy dissipation model for set associative caches based on CMOS static RAM cells.

2.1 Set–Associative Caches

Figure 1 depicts the overall architecture of a *m*–way set associative cache. It consists of *m* RAM banks, each bank row

* This work is supported in part by the NSF through award No. MIP–9504767 and by the IEEC at SUNY–Binghamton, a national center supported by the NSF, the State of New York and Industrial Partners

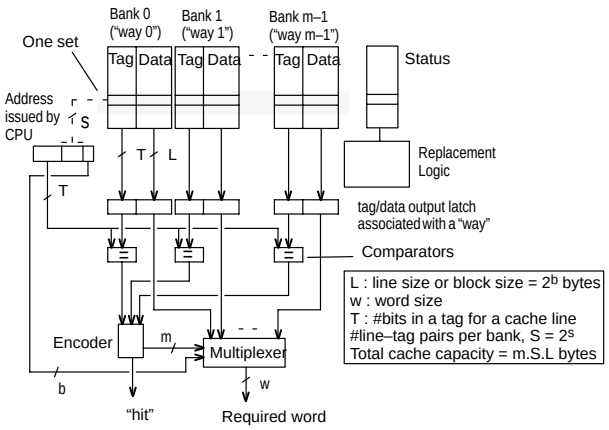


Figure 1. A m–way Set–Associative Cache

storing a contiguous block of memory words and a tag that uniquely identifies the block. The *k*–th row of all the banks constitutes the *k*–th set. A memory word at word address *A* has a block id of *A* **div** *L*, a tag value of *A* **div** (*S***L*). A block with block id *B* can be placed in at most one block frame in the set *B***mod***S*. A read access to a word in block *B* proceeds as follows:

1) Parallel Access of Tag, Data and Status RAMs: All of the tag, data and status bits in the set numbered *B* **mod** *S* are read out into latches associated with each way.

2) Tag Comparison and Data Steering: The request tag is compared simultaneously with the contents of the tag latches. If a tag match occurs, and if the block frame is valid (indicated by an associated valid bit), it is a cache hit and the desired word is steered out to the requester. Otherwise, a miss signal is generated, stalling the requester until the missing block is installed, after evicting a victim block chosen by a replacement algorithm [Smith 82].

Cache writes are handled as a normal cache read followed by a write. Two cache variations result, based on where the update is performed on a write – the write–through cache or a write back cache [Smith 82].

2.2. Energy Dissipation in SRAM Cells

Figure 2 shows some key components of a CMOS static RAM cell. A detailed description of the operations of all the components of a static RAM can be found in [WeEs 93]. The dominant energy dissipation in CMOS circuits is due to output transitions of gates which charge or discharge the load capacitance. The energy dissipated per transition is given by:

$$E_t = 0.5 \cdot C \cdot V^2 \quad (1)$$

where C is the capacitive load and V is the supply voltage as well as the output voltage change.

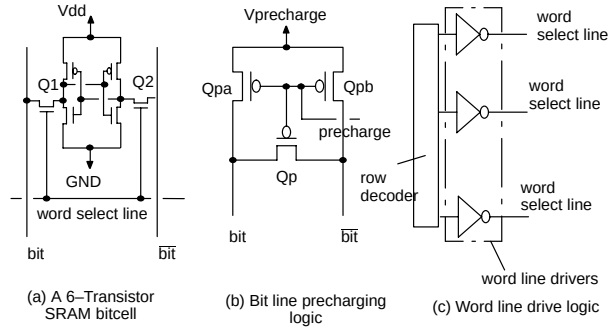


Figure 2. Some Components of a Static RAM

The major components of the CMOS SRAM that dissipate energy whenever the cell is accessed for read or write are:

Bit–line dissipations: caused due to precharging in preparation for an access and then during the actual read or write. Based on [WiJo94] we derived the following equations:

$$C_{bit,\,pr} = N_{rows} \cdot (0.5 \cdot C_{d,Q1} + C_{bit}) \quad (2)$$

$$C_{bit,\,r/w} = N_{rows} \cdot (0.5 \cdot C_{d,Q1} + C_{bit}) + C_{d,Qp} + C_{d,Qpa} \quad (3)$$

where $C_{bit,pr}$, $C_{bit,r/w}$ are the effective load capacitance of the bit lines during precharging, and read/write to the cell. $C_{d,Qx}$ is the drain capacitance of transistor Q_x and C_{bit} is the bitwire capacitance over the extent of a single bit cell. We assume a $\frac{1}{2}V_{dd}$ voltage swing on the bit lines.

Word–line dissipations: caused due to assertion of the word select line by the word line drivers to perform the read or write

$$C_{wordline} = N_{columns} \cdot (2 \cdot C_{g,\,Q1} + C_{wordwire}) \quad (4)$$

where $C_{wordline}$ is capacitive load of the driver, $C_{wordwire}$ is the word select line capacitance across the extent of a bit cell.

Dissipation in output lines: caused due to driving signals on the interconnects external to the cache.

Input line dissipations: caused due to transitions on the input lines and input latches.

2.3. Energy Dissipation in Set–Associative Caches

We now enumerate the energy dissipated within a m–way set associative cache, with a total data capacity of *D* bytes, a tag size of *T* bits and a line size of *L* bytes. Let *St* denote the number of status bits per block frame. These status bits are implemented as a row of $m \cdot St$ bits in a status RAM bank. The number of sets, S, is $D/(L \cdot m)$. The main components of energy dissipations are:

Energy dissipated in the bit lines, E_{bit}, due to precharging, readout and writes is given by:

$$N_{rows} = S \qquad N_{columns} = m \cdot (8 \cdot L + T + St)$$

$$E_{bit} = 0.5 \cdot V_{dd}^2 \cdot [N_{bit,\,pr} \cdot C_{bit,\,pr} + N_{bit,\,w} \cdot C_{bit,\,r/w} + N_{bit,\,r} \cdot C_{bit,\,r/w} + m \cdot (8 \cdot L + T + St) \cdot CA \cdot (C_{g,Qpa} + C_{g,Qpb} + C_{g,Qp})] \quad (5)$$

where $N_{bit,pr}$, $N_{bit,r}$, $N_{bit,w}$ are the total number of bit line transitions due to precharging, reads and writes, CA is the total number of cache accesses.

Energy dissipated in the word lines, E_{word}, including energy expended in driving the gate of the row driver

$$E_{word} = V_{dd}^2 \cdot CA \cdot m \cdot (L \cdot 8 + T + St) \cdot (2 \cdot C_{g,\,Q1} + C_{wordwire}) \quad (6)$$

Energy dissipated in the output lines, E_{output}, is the energy dissipated when driving interconnect lines external to the cache towards the cpu side or the memory side.

$$E_{aoutput} = 0.5 \cdot V_{dd}^2 \cdot (N_{out,\,a2m} \cdot C_{out,\,a2m} + N_{out,\,a2c} \cdot C_{out,\,a2c})$$

$$E_{doutput} = 0.5 \cdot V_{dd}^2 \cdot (N_{out,\,d2m} \cdot C_{out,\,d2m} + N_{out,\,d2c} \cdot C_{out,\,d2c})$$
$$E_{output} = E_{aoutput} + E_{doutput} \quad (7)$$

where $E_{aoutput}$, $E_{doutput}$ are the address and data lines dissipations, $N_{aout,a2m}$, $N_{out,d2m}$ are the number of transitions on the memory–side address and data line drivers, $C_{out,a2m}$ and $C_{out,d2m}$ are their corresponding capacitive loads. Similarly $N_{out,a2c}$, $N_{out,d2c}$ and $C_{out,a2c}$, $C_{out,d2c}$ are the corresponding terms for the cpu–side interconnect. Following [WiJo 94], the capacitive load for on–chip destinations is 0.5pF and for off–chip destinations, it is 20pF.

Energy dissipated in the address input lines, E_{ainput} is the energy expended in the input gates of the row decoder,

$$E_{ainput} = 0.5 \cdot V_{dd}^2 \cdot N_{ainput} \cdot [(m + 1) \cdot 2 \cdot S \cdot C_{in,dec} + C_{awire}] \quad (8)$$

where N_{ainput} is number of transitions in the address input lines, $C_{in,dec}$ is the gate capacitance of first level of decoder, and C_{awire} is the wire capacitance of the common address lines feeding the decoder in each RAM bank.

Energy dissipations occur in the sense amps due to output loading of the sense amps by the inputs of the data output drivers and internal short circuit current. Tag comparators, cache control logic and address comparators in writeback queue also contribute to enery consumption. These components have been ignored for the studies presented here because they are relatively small. The total energy dissipation in a CMOS SRAM based set associative cache is thus given by

$$E_{cache} = E_{bit} + E_{word} + E_{output} + E_{ainput} \quad (9)$$

The actual value of capacitive coefficients is derived from [WiJo 94] assuming a 0.8 micron implementation. We also adapt the energy dissipation equations of the different cache components for various non–standard organizations presented further.

3. LOW POWER CACHE DESIGNS

We now discuss the architectural power reduction techniques that we have used in our studies.

3.1 Block Buffering

Caches exploit spatial and temporal localities in programs. It is thus very likely that a requested word is confined to the block (or one of the blocks of the set) that was last accessed. In this situation, which we refer to as *block buffer hit*, the tag/data and status arrays need not be read out thereby saving on the precharge and readout energy. To implement such a scheme, we need an extra latch and a comparator that stores and compares the set address of the last access with that of the current address. The block buffering scheme suggested in [SuDe 95] performs the block buffer hit comparison *first* and only on a miss is the usual cache probe started. This adds to the cache access latency, affecting performance. We propose a slightly modified version that implements block buffering without adding to latency (using two phase clocks).

(Modified) Step 1:

Clock phase 0: Perform tag, set address and valid bit comparison. Start precharging tag, data and status RAMs.

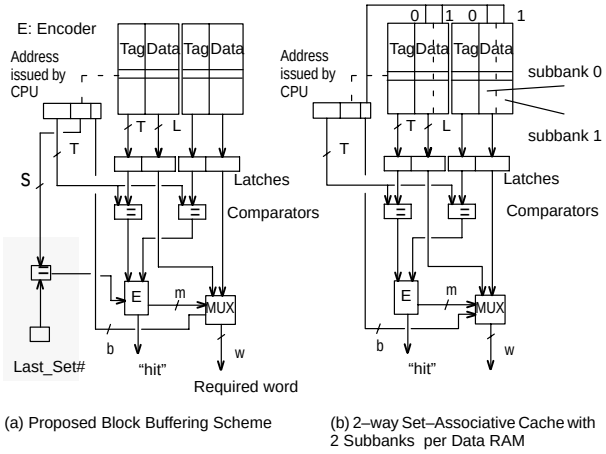
Clock phase 1: Read out cells if a block buffer miss occurs.

Step 2: (phase 0 not performed if hit occurs in Step 1)

Clock phase 0: Perform tag and valid bit comparison.

Clock phase 1: Steer matching data and update replacement bits, contents of *last_set_addr* latch etc.; generate hit/miss signal, activate replacement logic in case of a miss.

Figure 3(a) shows our scheme for a 2–way set–associative



(a) Proposed Block Buffering Scheme

(b) 2–way Set–Associative Cache with 2 Subbanks per Data RAM

Figure 3. Block Buffering and Cache Subbanking

cache, with the components needed for block buffering. Compared to the scheme of [SuDe 95], our scheme spends some energy in precharging bit lines on a block buffer hit, but does not cost in latency. Besides, precharging would be necessary anyway for subsequent accesses and successive precharging of bit lines without intervening readouts does not drain much energy.

3.2 Cache with Data RAM Subbanking

Since the word offset into the block frame is already available *before* the cell readout is performed, power savings can be achieved by reading out only the words at the required offset from all the ways. This technique is called subbanking. Figure 3(b) depicts a 2–way set associative cache with two subbanks in each data RAM. A subbank address decoder is needed to enable only the desired subbank before the word line is asserted. The energy efficiency of single level subbanked cache was also studied in [SuDe 95]. We extend it to two level cache hierarchy.

3.3 Other approaches

We have also studied the effectiveness of bus invert coding, proposed in [StBu 95] for reducing I/O power. This involves the use of one extra line to indicate the encoding of the value on the other lines (logically inverted or non–inverted).

4. ANALYTICAL MODELS FOR DISSIPATIONS

We present the formulation of the analytical models to estimate power dissipation in static RAM based caches. These models use the traditional cache statistics such as hit/miss counts, fraction of read/write requests, number of dirty victims etc. These statistics are obtained from our simulation tool CAPE (*CA*che *P*ower *E*stimator). The transition counts

for the various energy components discussed in Section 2.3 are derived using the cache organizational parameters and assuming an uniform, independent and identical probability distribution function for signal transitions.

4.1 Transition Counts in a Set–Associative Cache

The transition counts in the major energy dissipation components of set associative caches, discussed in Section 2.2 are now derived based on run time statistics and organizational parameters. The notations $Pr_{0\rightarrow1}$, $Pr_{1\rightarrow0}$, $Pr_{0\rightarrow1,1\rightarrow0}$ denote the probabilities of signal transitions between logical 0 and 1. We make the assumption that all signal values are independent and have uniform probability of 0 or 1 value, making the above terms equal to 0.5. We have used the following notations: N_{rhit}, N_{rmiss} are the number of read hits and misses, N_{whit}, N_{wmiss} are the number of write hits and misses, N_{hit}, N_{miss} are the total number of hits and misses, N_{wb_req} are the number of write back requests and $N_{bitlines}$ are the number of bit lines

$N_{bit,\ pr}$: In all the cache organizations, the bit lines are precharged for every access made to the cache. Total number of bit line transitions due to precharging will be

$$N_{bit,\ pr} = Pr_{0\rightarrow1} \cdot (N_{hit} + N_{miss}) \cdot N_{bitlines} \quad (10)$$

$N_{bit,\ r}$: These are energy dissipating transitions that occur when cell contents are read out onto the *precharged* bit lines. The total number of bit line read transitions will be

$$N_{bit,\ r} = Pr_{1\rightarrow0} \cdot N_{array\_acc} \cdot N_{bitlines} \quad (11)$$

$N_{bit,\ w}$: These are energy dissipating transitions that occur in bit lines when updating contents of the cache. A write occurs when there is a write–hit or when a missing line is installed. Let $W_{avg,\ data}$ be the average data width of a write request. The total number of bit line write transitions will be

$$N_{bit,\ w} = Pr_{0\rightarrow1,1\rightarrow0} \cdot N_{whit} \cdot (St + W_{avg,\ data}) \cdot 2 + Pr_{0\rightarrow1,1\rightarrow0} \cdot N_{rmiss} \cdot (T \cdot 1 + St + 8 \cdot L \cdot 1) \cdot 2 \quad (12)$$

The number of cache access CA in Equations 5, 6 actually means the number of array accesses denoted by N_{array_acc}

$$CA = N_{array\_acc} \quad (13)$$

$N_{out,a2m}$: The memory–side address bus is driven on a read miss and on write–back or write–through requests (depending on the write policy of the cache). We denote the width of the address bus by W_{addr_bus}. The total number of transitions on the memory–side address bus will be

$$N_{out,a2m} = Pr_{0\rightarrow1,1\rightarrow0} \cdot (N_{rmiss} + N_{wmiss} + N_{wb\_req}) \cdot W_{addr\_bus} = Pr_{0\rightarrow1,1\rightarrow0} \cdot (N_{rmiss} + N_{whit} + N_{wmiss}) \cdot W_{addr\_bus} \quad (14)$$

for write–back and write–through caches respectively.

$N_{out,\ d2m}$: The memory–side data bus would be driven when there is a write–back request (in a write–back cache) or when there is a write request (in a write–through cache).The number of transitions on the memory–side data bus would thus be:

$$N_{out,d2m} = Pr_{0\rightarrow1,1\rightarrow0} \cdot N_{wmiss} \cdot W_{avg,\ data} + Pr_{0\rightarrow1,1\rightarrow0} \cdot N_{wb\_req} \cdot 8 \cdot L = Pr_{0\rightarrow1,1\rightarrow0} \cdot (N_{whit} + N_{wmiss}) \cdot W_{avg,\ data} \quad (15)$$

for write–back and write–through caches respectively.

$N_{out,\ d2c}$: In the first level the cpu–side data bus is driven when there is a read request and the width of data delivered, $W_{avg,data\_read}$, depends on the distribution of byte, halfword and word size LOAD instructions. For other levels, the data width will be the block size, L_{higher}, of the higher level cache.

$$N_{out,d2c} = Pr_{0\rightarrow1,1\rightarrow0} \cdot (N_{rhit} + N_{rmiss}) \cdot 8 \cdot L_{higher} \quad \text{(L2 onwards)}$$
$$= Pr_{0\rightarrow1,1\rightarrow0} \cdot (N_{rhit} + N_{rmiss}) \cdot W_{avg,\ data\_read} \quad \text{(L1)} \quad (16)$$

$N_{aninput}$: The address input line transitions cause energy dissipation whenever an address is latched into the address decoder. The total number of input line transitions will be given by

$$N_{ainput} = Pr_{0\rightarrow1,1\rightarrow0} \cdot (N_{hit} + N_{miss}) \cdot W_{addr\_bus} \quad (17)$$

4.2 Energy Dissipation of a Conventional Cache

In a conventional cache, the number of array accesses is same as the number of cache accesses, and the width of the data read out from the data cell array is the line width, L bytes, of a block. In our simulation, all LOAD instructions deliver a word–wide data from the L1 Dcache, and the required bytes are selected internally. The following are the analytical equations for transition counts in a conventional cache

$$CA = N_{array\_acc} = N_{hit} + N_{miss}$$
$$N_{bitlines} = (T \cdot m + St + 8 \cdot L \cdot m) \cdot 2$$
$$N_{bit,\ pr} = 0.5 \cdot (N_{hit} + N_{miss}) \cdot (T \cdot m + St + 8 \cdot L \cdot m) \cdot 2$$
$$N_{bit,\ r} = 0.5 \cdot (N_{hit} + N_{miss}) \cdot (T \cdot m + St + 8 \cdot L \cdot m) \cdot 2$$
$$N_{bit,\ w} = 0.5 \cdot N_{rmiss} \cdot (T \cdot 1 + St + 8 \cdot L \cdot 1) \cdot 2 + 0.5 \cdot N_{whit} \cdot (St + W_{avg,\ data}) \cdot 2$$
$$W_{addr\_bus} = 32 \qquad W_{avg,\ data\_read} = 32 \text{ for Icache, Dcache}$$
$$W_{avg,\ data} = (8 + 16 + 32)/3 = 19 \quad \text{if all caches are write through}$$
$$N_{out,a2m} = 0.5 \cdot (N_{rmiss} + N_{wmiss} + N_{wb\_req}) \cdot 32 \quad \text{(write back)}$$
$$= 0.5 \cdot (N_{rmiss} + N_{whit} + N_{wmiss}) \cdot 32 \quad \text{(write through)}$$
$$N_{out,d2m} = (N_{whit} + N_{wmiss}) \cdot 0.5 \cdot W_{avg,\ data} \quad \text{(write–through)}$$
$$= 0.5 \cdot N_{wmiss} \cdot W_{avg,\ data} + 0.5 \cdot N_{wb\_req} \cdot 8 \cdot L \text{(write–back)}$$
$$N_{out,\ d2c} = 0.5 \cdot (N_{rhit} + N_{rmiss}) \cdot 8 \cdot L_{higher} \quad \text{(L2 cache onwards)}$$
$$= 0.5 \cdot (N_{rhits} + N_{rmiss}) \cdot 19 \quad \text{(L1 cache)}$$
$$N_{ainput} = 0.5 \cdot (N_{hit} + N_{miss}) \cdot 32$$

4.3 Cache Energy Dissipation with Sub–banking

In this case, tag and status bits access is same as conventional cache, but data access is limited only to the desired word in the block. Denoting the width of a subbank by W_{word} we have

$$CA = N_{array\_acc} = N_{hit} + N_{miss}$$
$$N_{bitlines} = (T \cdot m + St + W_{word} \cdot m) \cdot 2$$
$$N_{bit,\ pr} = 0.5 \cdot (N_{hit} + N_{miss}) \cdot (T \cdot m + St + W_{word} \cdot m) \cdot 2$$
$$N_{bit,\ r} = 0.5 \cdot (N_{hit} + N_{miss}) \cdot (T \cdot m + St + W_{word} \cdot m) \cdot 2$$
$$N_{bit,\ w} = 0.5 \cdot N_{rmiss} \cdot (T \cdot 1 + St + 8 \cdot L \cdot 1) \cdot 2 + 0.5 \cdot N_{whit} \cdot (St + W_{avg,\ data}) \cdot 2$$

The equations to compute the output driver and address input transition counts will be same as that given in Section 4.2

4.4 Cache Energy Dissipation with Block Buffering

In this case, tag, status bits and data arrays are accessed only on a miss in the block buffer. The number of bits accessed is same as that of the conventional cache. We denote the number of block buffer misses by $N_{blockbuff\_miss}$. Thus we have

$$CA = N_{array\_acc} = N_{blockbuff\_miss}$$
$$N_{bitlines} = (T \cdot m + St + 8 \cdot L \cdot m) \cdot 2$$

$N_{bit,\ pr} = 0.5 \cdot N_{blockbuff\_miss} \cdot (T \cdot m + St + 8 \cdot L \cdot m) \cdot 2$

$N_{bit,\ r} = 0.5 \cdot N_{blockbuff\_miss} \cdot (T \cdot m + St + 8 \cdot L \cdot m) \cdot 2$

$N_{bit,\ w} = 0.5 \cdot N_{rmiss} \cdot (T \cdot 1 + St + 8 \cdot L \cdot 1) \cdot 2 + 0.5 \cdot N_{whit} \cdot (St + W_{avg,\ data}) \cdot 2$

The equations to compute the output driver and address input transition counts will be same as that given in Section 4.2

4.5 Cache Energy Dissipation with Block Buffering and Subbanking :

With both, block buffering and subbanking, techniques we have to account for block buffer hits as well as smaller data width read out. Thus we have

$CA = N_{array\_acc} = N_{blockbuff\_miss}$

$N_{bitlines} = (T \cdot m + St + W_{word} \cdot m) \cdot 2$

$N_{bit,\ pr} = 0.5 \cdot N_{blockbuff\_miss} \cdot (T \cdot m + St + W_{word} \cdot m) \cdot 2$

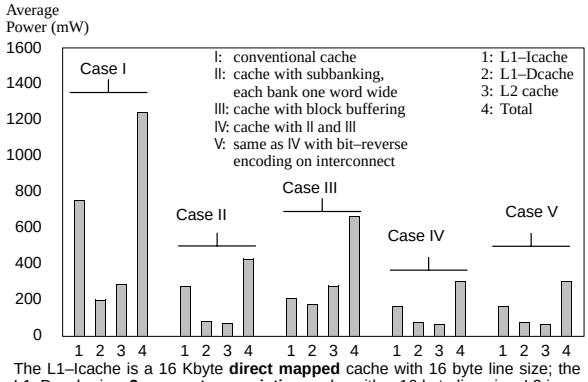
$N_{bit,\ r} = 0.5 \cdot N_{blockbuff\_miss} \cdot (T \cdot m + St + W_{word} \cdot m) \cdot 2$

$N_{bit,\ w} = 0.5 \cdot N_{rmiss} \cdot (T \cdot 1 + St + 8 \cdot L \cdot 1) \cdot 2 + 0.5 \cdot N_{whit} \cdot (St + W_{avg,\ data}) \cdot 2$

The equations to compute the output driver and address input transition counts will be same as that given in Section 4.2.

For the case study reported here, we consider a two–level cache system, where the 2nd level cache (L2) is off chip. The capacitive coefficients are derived based on the 0.8micron cache described in [WiJo 94]. We assumed $V_{dd} = 3.3V$, and the CPU and 2–stage pipelined caches were clocked at 85MHz. The 0.8micron cache has a cycle time of < 10.2 ns, for a wide range of configurations [WiJo 94], so a 85MHz clock is appropriate. We simulated the execution of several SPECint92 benchmarks on the CAPE simulator.

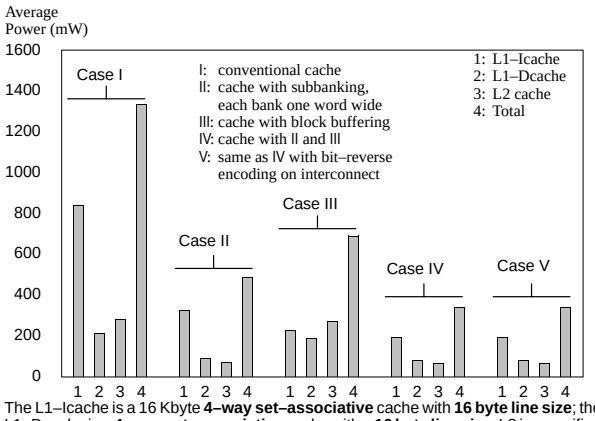
For our **base case** (Case 1), we chose a 16KB, direct–mapped L1 Icache and a 16KB, 2–way set–associative L1 Dcache each with 16byte (4 word) line size. We chose a 64KB, 4–way unified L2 cache with a 32byte line size. The interconnection bus width was 16byte data bus between L1 and L2 and between L2 and main memory. All caches were write–through with 6 deep write back buffers.



The L1–Icache is a 16 Kbyte **direct mapped** cache with 16 byte line size; the L1–Dcache is a **2–way set–associative** cache with a 16 byte line size; L2 is an unified 64 Kbyte, **4–way set–associative** cache with 32 byte line size. *Similar enhancements are made to all caches at the same time, except that L2 was 8–way subbanked.*

Figure 4. Effects on Cache Power Dissipation Due to Various Enhancements (Case 1)

For Case 2, we maintained the same capacity of the caches. We increased the associativity of both L1 caches to 4, with the same line size. The L2 cache was not changed. Many other



The L1–Icache is a 16 Kbyte **4–way set–associative** cache with **16 byte line size**; the L1–Dcache is a **4–way set–associative** cache with a **16 byte line size**; L2 is an unified 64 Kbyte, **4–way set–associative** cache with **32 byte line size**. *Similar enhancements are made to all caches at the same time except that L2 was 8–way subbanked..*

Figure 5. Effects on Cache Power Dissipation Due to Various Enhancements (Case 2)

variations in the line size and associativity were also simulated and we compared the analytical and experimental results. Only the two cases mentioned above are presented here. Figures 5 and 6 show the experimentally estimated power dissipation in the various caches.

5. VALIDATION OF THE ANALYTICAL MODELS

For experimentally estimating the energy dissipation in static RAM caches with or without the various energy saving techniques, the values of $N_{bit,\ pr}$, $N_{bit,\ r}$, $N_{bit,\ w}$, $N_{out,\ a2cpu}$, $N_{out,\ a2m}$, $N_{out,\ d2c}$, $N_{out,\ d2m}$ and N_{ainput} are obtained by simulation and counting the *actual* transitions in the simulator. The use of *CAPE* to obtain accurate measures of transition counts by actual simulations is explained in [KaGh97]. CAPE incorporates a register–level RISC pipeline simulator, an integrated multilevel cache simulator, and a transition counting mechanism. The pipeline simulator has been adapted from the S20, MIPS–R2000 machine pipeline simulator [La 95] adapted for cycle level simulation [Ro 96].

Tables 1–4 depict the percentage error between the experimental and analytical values for each cache in the simulated system. The percentage error is computed as

$$\%Error = (Power_{expr} - Power_{analytical}) / Power_{expr} * 100$$

We observe that the error in the overall power is quite small in L1 caches in cases I, II and III. However the relative error is large when subbanking and block buffering are jointly used. This is because the main source of error is the bit line precharging transitions counts. This arises because the analytical model fails to take into account the fact that bit lines do not have to be precharged fully when a consecutive block buffer hit occurs. We also noticed a large discrepancy in the bit line write power. This is due to the fact that generally data values (in Dcache) and instruction encodings (in Icache) will have large number of bits of the same pattern. The error in output address power is also consistently large in the order of 200%, attributable to the fact that data and instruction access addresses exhibit strong locality.

The locality of reference in L2 cache is lower than the L1 caches because of the filtering occurring in L1. This reduces the deviation between analytical and experimental values of the output energy component inspite of the higher capacitive load of the off–chip pads.

Orgz	E_{bit}		E_{word}		$E_{aoutput}$		$E_{doutput}$		E_{ainput}		E_{total}	
	Case 1	Case 2	Case 1	Case 2	Case 1	Case 2	Case 1	Case 2	Case 1	Case 2	Case 1	Case 2
I	1.78	1.59	0	0	–214	–204	–40	–42	–837	–845	–1.32	–0.49
II	4.81	4.12	0	0	–214	–204	–40	–42	–837	–845	–3.66	–1.24
III	4.83	5.05	0	0	–214	–204	–40	–42	–144	–144	2.43	3.55
IV	–21.5	–22.5	0	0	–214	–204	–40	–42	–615	–615	–31.4	–28.6
V	–21.5	–22.5	0	0	–214	–204	–40	–42	–615	–615	–31.4	–28.6

Table 1 %Error in Estimating Icache Energy Dissipation

Orgz	E_{bit}		E_{word}		$E_{aoutput}$		$E_{doutput}$		E_{ainput}		E_{total}	
	Case 1	Case 2	Case 1	Case 2	Case 1	Case 2	Case 1	Case 2	Case 1	Case 2	Case 1	Case 2
I	0.52	1.02	0	0	–272	–277	–33.4	–35.2	–252	–252	–6.86	–6.03
II	1.47	2.67	0	0	–272	–277	–33.4	–35.2	–252	–252	–16.7	–14.3
III	10.6	11.1	0	0	–272	–277	–33.4	–35.2	–163	–162	2.01	2.76
IV	–1.07	0.15	0	0	–272	–277	–33.4	–35.2	–225	–225	–20.2	–17.7
V	–1.07	0.15	0	0	–272	–277	–33.4	–35.2	–225	–225	–20.2	–17.7

Table 2 %Error in Estimating Dcache Energy Dissipation

Orgz	E_{bit}		E_{word}		$E_{aoutput}$		$E_{doutput}$		E_{ainput}		E_{total}	
	Case 1	Case 2	Case 1	Case 2	Case 1	Case 2	Case 1	Case 2	Case 1	Case 2	Case 1	Case 2
I	0.02	–0.02	0	0	–285	–282	–41.8	–37.8	–267	–273	–4.99	–4.9
II	0.26	0.10	0	0	–285	–282	–41.8	–37.8	–267	–273	–20.1	–20.1
III	21.2	21.5	0	0	–285	–282	–41.8	–37.8	–170	–178	15.3	15.9
IV	–5.76	–5.19	0	0	–285	–282	–41.8	–37.8	–257	–261	–26.5	–25.9
V	–5.76	–5.19	0	0	–285	–282	–46.6	–42.4	–257	–261	–27.2	–26.6

Table 3 %Error in Estimating L2 Cache Energy Dissipation

Orgz	L1 I–Cache		L1 D–Cache		L2–Cache	
	Case 1	Case 2	Case 1	Case 2	Case 1	Case 2
I	–1.10	–0.46	–5.92	–5.14	–4.74	–4.66
II	–3.06	–1.17	–14.42	–12.21	–18.91	–18.77
III	3.08	3.63	1.62	2.33	16.75	15.54
IV	–30.11	–27.79	–17.76	–15.54	–25.46	–24.52
V	–30.11	–27.79	–17.76	–15.54	–27.03	–26.13

Table 4 %Error in Total Power Dissipaton

6. CONCLUSIONS

We presented analytical models for estimating the energy dissipation in conventionally–organized caches as well as caches organized for lower energy dissipation. The analytical approach presented represents a faster way of estimating the energy dissipation in caches, compared to the the more accurate approach of counting all the transitions explicitly during simulation.

These analytical models use the number of cache hits and misses, the number of cache reads and writes and similar parameters obtained from a detailed simulator for a typical cache system. In particular, the analytical models ignores the transitions caused by actual data and address values and assumes instead that transitions caused by these values are stochastically distributed. The accuracy of the analytical models were checked against the detailed transition counts, including transitions due to actual data and address values obtained from the same simulator. For the cache organization studied, the analytical models for conventionally–organized caches predict the overall energy dissipation within less than 2% of the dissipation estimated from detailed transition counts. However, for some low power cache organizations, the analytical models overestimate the power dissipation by as much as 30%, a fact that can be traced to the correlated nature of transitions within a cache.

References

[ERB+ 95] Edmondson, J. F. et al, “Internal Organization of the Alpha 21164, a 300–MHz 64–bit Quad–issue CMOS RISC Microprocessor”, Digital Technical Journal, Vol. 7, No. 1, 1995, pp. 119–135.

[Intel 96] Intel Corporation, Pentium–Pro data book, 1996.

[JBD+ 93] Jouppi, N., et al., “A 300–MHz 115–W 32–b Bipolar ECL Microprocessor”, in IEEE Journal of Solid–State Circuits, Nov. 1993, pp. 1152–1165.

[KaGh 97] Kamble, M. B. and Ghose, K., “Energy–Efficiency of VLSI Caches: A Comparative Study”, in Proc. IEEE 10–th. Int’l. Conf. on VLSI Design, Jan. 1997, pp. 261–267.

[Larus 96] Larus, J., “SPIM: A MIPS 2000 SImulator”, available form Univ. Wis., CS ftp site.

[Mon 96] Montanaro, J. et al., “A 160 MHz, 32b 0.5 W CMOS RISC Microprocessor”, in IEEE ISSCC 1996 Digest of Papers, 1996.

[Ro 96] Rogers, A., “CL–SPIM: A Cycle Level SImulator for the MIPS 2000”, available form Univ. Wis., CS ftp site.

[Smith 82] Smith, A. J., “Cache Memories”, ACM Computing Surveys, Sept. 1982, pp. 473–530.

[StBu 95] Stan, M. R. and Burleson, W. P., “Bus–Invert Coding for Low–Power I/O”, IEEE Trans. on VLSI Systems, March 1995, pp.49–58.

[SuDe 95] Su, C. and Despain, A., “Cache Design Tradeoffs for Power and Performance Optimization: A Case Study”, in Proc. of the Int’l. Sym. on Low Power Design, 1995, pp. 63–68.

[WeEs 93] Weste, N. H. E. and Eshraghian K, *Principles of CMOS VLSI Design*, 2nd edition, Addison–Wesley, 1993.

[WiJo 94] Wilton, S. E., and Jouppi, N., “An Enhanced Access and Cycle Time Model for On–Chip Caches”, DEC WRL Research Report 93/5, July 1994