

Energy Efficient Caching-on-Cache Architectures for Embedded Systems

HUNG-CHENG WU, TIEN-FU CHEN, HUNG-YU LI^{*}
AND JINN-SHYAN WANG⁺

Department of Computer Science

⁺*Department of Electrical Engineering*

National Chung Cheng University

Chiayi, 621 Taiwan

As the number of applications for embedded and real-time systems has grown, running systems at a lower power dissipation level has become an important issue, especially for the battery-based systems. Past studies have shown that the cache is responsible for a large part of the power dissipation in a system chip. Thus, reducing the power dissipation in the cache has become an important research topic.

In this paper, we present a cache architecture, the *CoC*, that reduces cache energy dissipation in embedded systems. The *CoC* can reduce the access frequency at wordlines and bitlines, which are the main power consuming components, in both tag and data arrays. In addition, we use a hierarchical address comparison (*HAC*) scheme to optimize energy reduction and the comparison time. Experimental results show that the *CoC* is practical in reducing energy dissipation in the L1 cache, and that the *HAC* scheme can effectively minimize the comparison penalty in the *CoC*.

Keywords: low power cache, data reuse, CAM, power evaluation, hierarchical address comparator

1. INTRODUCTION

The performance of processors has substantially improved in recent years as process technology has made significant progress. System performance depends on not only the processor, but also the cache system. This situation has been more obvious when the system is running multimedia applications. In particular, as SIMD-like enhancements are added to the CPU for multimedia processing, the bandwidth and power requirements become more important issues in designing memory systems that support these applications. In chip power dissipation and evaluation studies [13], the on-chip cache has been found to be responsible for a significant part of the power dissipation in most modern processors.

To reduce the power dissipation caused by frequent data access in the cache, several techniques has been proposed in the literature. These techniques for reducing cache power consumption can be categorized as follows: (1) those that improve the cache architecture, such as filter caching [9], victim buffering [1], block buffering [8, 17], multiple line buffering [4], and way-predicting set-associative caching [6]; (2) those that operate in a software environment, such as dynamic cache management [3] and the code

Received January 16, 2001; revised August 26, 2002; accepted November 7, 2002.

Communicated by Lionel M. Ni.

transformation technique [11]; (3) those that involve VLSI circuit design, such as low-power encoding [15], MDM [10], and domino tag comparators [5]; and (4) the compilation techniques [18]. The approach we propose in this study is based on design enhancement of cache architectures.

In this paper, we propose two methods for reducing cache power dissipation. The first method is *Caching on Cache (CoC)*, which is a small size of sub-cache in L1 cache. The purpose of the *CoC* is to minimize the frequency of cache access and thus to reduce overall cache power consumption. When a cache access request occurs, the *CoC* disables the wordlines in both the data and tag arrays if the *CoC* hits. When a *CoC* miss happens, the wordlines in both the data and tag arrays are enabled, and the cache performs a normal cache operation. Since accessing the *CoC* is an extra overhead, we propose a hierarchical address comparison (*HAC*) scheme to reduce the access cost in the *CoC*. In the *HAC* operation, we divide the address comparison step into several different segments and thus reduce unnecessary address comparison when a mismatch is detected at an earlier stage. To evaluate the proposed scheme, we have employed trace-driven simulation and developed a power evaluation model to compare the traditional approach with our proposed scheme. The simulation results show that the *CoC* with four entries and the *HAC* scheme with an index segment can significantly reduce the number of cache operations.

The remainder of this paper is organized as follows. In section 2, we present our main ideas for the *CoC* and *HAC*. We present a performance evaluation of the power evaluation model in section 3. The overall simulation environment and performance results are also presented. Then, we review related works on reducing power dissipation for on-chip caches in section 4. Section 5 concludes this paper.

2. PROPOSED METHOD

2.1 Traditional Cache Access Model

In the original cache operation, the requested address is first split into three parts: the block, index, and tag. First of all, the index is decoded, and the associated word-lines in both the data and tag arrays are enabled. A row of memory cells is selected, and the content of the memory cells is read out after the bit-lines are pre-charged. The sense amplifiers scale up these signals in the bit-lines. In the tag arrays, the signals that are scaled up are compared with the tag part of the requested address. If tag comparison results in a match, the cache sends the request data to the CPU. Otherwise, it needs to fetch data from the next level of the memory hierarchical system. Both the tag and data signals must pass through the address decoder, word-line, bit-line, and sense amplifiers. Obviously, a high level of power dissipation will be generated by the tag and data arrays. Here, the *CoC* is used to reduce the access frequency in both the tag and data arrays. The modified cache hierarchical system is shown in Fig. 1.

In *CoC*, power dissipation is caused by address comparison and register accessing. The *HAC* scheme is used to perform partial *CoC* address comparison. Using the *HAC* scheme, we can reduce the amount of unnecessary address comparison without sacrificing the hit ratio of the *CoC*. In the following sections, we will describe all in the detail *CoC* and the *HAC* scheme.

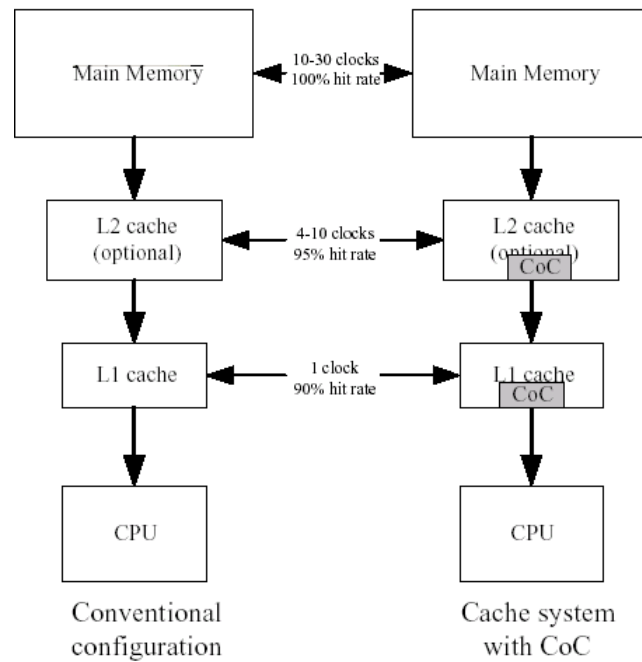


Fig. 1. Traditional and CoC embedded memory systems.

2.2 Caching on Cache (CoC)

The *CoC* is located in front of a normal cache as shown in Fig. 2. The *CoC* is composed of registers and a CAM. All the data are stored in registers and addresses put in the CAM or in the combination of register cells and CAM. As the *CoC* is imbedded in the cache, it does not contribute much penalty to the L0 cache. When the *CoC* hits, data are read out from the *CoC* data register line. A *CoC* miss will enable the cache index decoder to read out the tag and data as in a normal cache operation.

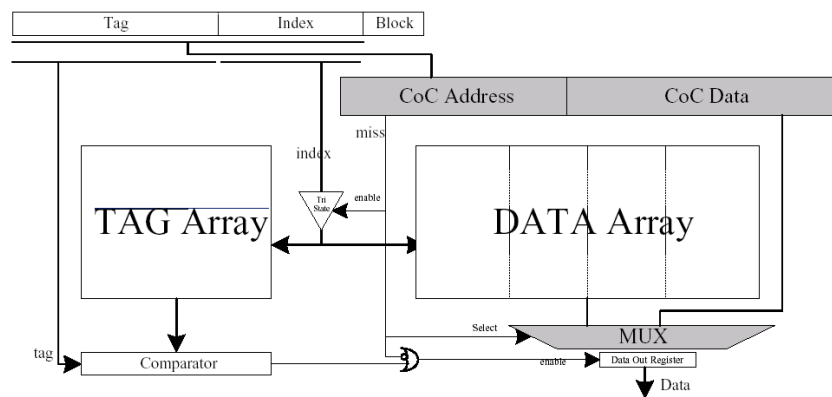


Fig. 2. A cache with the CoC architecture.

We summarize the energy consumption in traditional caches in Eq. (1) and that in a cache with the *CoC* in Eq. (2):

$$E_{normal\ cache} = E_{C_access} + R_{C_miss} \times E_{C_update} \quad (1)$$

$$E_{Cw/CoC} = E_{CoC} + R_{CoC_miss} \times (E_{C_access} + E_{CoC_update}) + R_{C_miss} \times E_{C_update}, \text{ where} \quad (2)$$

- E_{CoC} , E_{CoC_update} : the energy consumed by a *CoC* access and that consumed by an update operation of one *CoC* entry, respectively;
- E_C , E_{C_update} : the energy consumed by one cache hit and that consumed by updating cache operations, respectively; E_{C_update} also includes the energy consumed by accessing the next level of the memory system;
- R_{CoC_miss} , R_{C_miss} : the ratios of a *CoC* miss and a cache miss, respectively.

The objective of designing an efficient on-chip cache is to achieve a low miss ratio. Without an index decoder, the *CoC* has to compare the entire address to verify if the cache is hit or miss. With higher spatial locality for data access, the most significant part (MSB) of the request address does not change frequently in a sequential cache reference. On the contrary, the least significant part of address varies more than the MSB. Here we separate the address into two main parts: the LSB and MSB. To reduce the cache access time that combines the *CoC* execution, we list the following conditions:

- *CoC* hit: When the *CoC* hits, the cache array is disabled. The access time is sourced from the *CoC* address comparison result and data readout.
- *CoC* miss: When a *CoC* miss happens, the cache is enabled, and the requested data is read out as in a normal cache access.
 - Cache hit: The access time includes both the cache hit operation and *CoC* address comparison.
 - Cache miss: The access time depends on the next level of the memory hierarchy. We compare the L1 and our *CoC* with L1 cache systems in Table 1.

Table 1. L1 v.s. L1/*CoC* access time comparison.

	Cache Hit		Cache Miss	
	<i>CoC</i> Hit	<i>CoC</i> Miss	<i>CoC</i> Hit	<i>CoC</i> Miss
L1	T_C		T_{Next}	
L1/ <i>CoC</i>	$< T_C$	$> T_C$	<i>null</i>	$\approx T_{Next}$

In Table 1, T_C is the access time for the cache, and T_{Next} is that for the next level of the memory system. *Null* in the table indicates that a cache miss will not happen when the *CoC* hits. When the cache hits and *CoC* misses, the access time is larger than that in any other situation. Furthermore, the cache performance is much worse when the *CoC* miss ratio is high.

2.3 Hierarchical Address Comparison (*HAC*)

The property of spatial locality can be found by the fact that requested addresses are usually sequential. This means that addresses may vary in the least significant bits, and

that the other part will often be the same. When the requested data is all in the same page, the least significant bits may be located near the block bits.

To reduce the comparison time in the CoC, we decompose the address comparison process into a hierarchical address comparison step that compares the address partially. As the LSB varies more than the MSB, the first step is to compare the LSB to determine if it is necessary to perform MSB comparison in the CoC. If the LSB does not match, we do not have to compare the MSB in the next step. In this way, by partitioning the address comparison step in a hierarchical manner, we can reduce the energy that is consumed by performing a full address comparison. However, this may incur a small delay as the address comparison is performed by two steps. This results in a trade-off between the comparison time and energy dissipation.

In Fig. 3, the address is separated into two MSB's and two LSB's. The higher MSB and LSB are divided according to the range of the page size, and the lower ones are determined by the block size. In the HAC operation, the LSB is first compared and then MSB is compared. If the LSB part is large enough, it is likely to eliminate the number of MSB comparisons in the CoC.

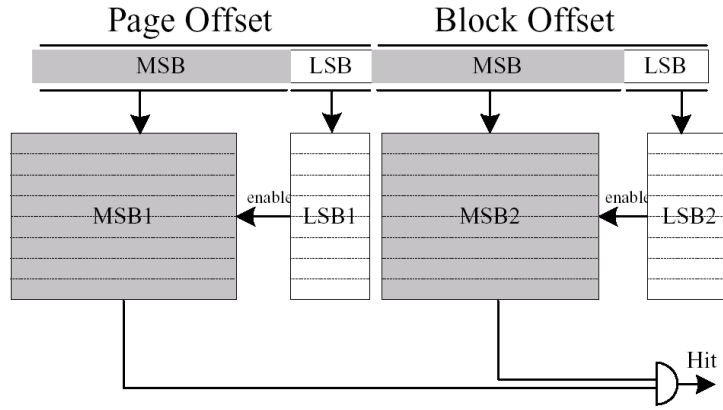


Fig. 3. Hierarchical address comparison detector.

To minimize the use of the MSB comparator, the contents stored in the CoC LSB lines should usually be different. When the number of different LSB in the CoC is reduced, the energy consumed by comparing the MSB is also minimized:

$$E_{HAC} = N_{entry} \times E_{LSB} + N_{LSB\ hit} \times E_{MSB}. \quad (3)$$

The energy consumed by an HAC operation in the CoC is expressed by Eq. (3), where E_{LSB} is the energy consumed by LSB comparison and E_{MSB} is that for MSB comparison. $N_{LSB\ hit}$ corresponds to the entry number for what LSB is matched. As described above, a larger LSB will reduce the energy consumed by MSB comparison and the value of $N_{LSB\ hit}$. However, it will increase E_{LSB} when LSB comparison is performed. In contrast, a larger MSB will increase $N_{LSB\ hit}$ and E_{MSB} . In the following, we will discuss the partition result for the MSB and LSB in the CoC.

2.4 Implementation Issues

A *CoC* may be constructed using either register arrays (REG) or content access memory (CAM), whereas data lines are stored in register cells. To buffer the most recently used data in the *CoC*, we use the FIFO policy for *CoC* replacement.

Consequently, the *CoC* behaves like a fully associative cache. To reduce the penalty due to address comparison, we use two different models in the *CoC*. A CAM bit is like a register with a comparator but more compact and with lower energy dissipation. Both models use CAM for index comparison. Here, we will discuss the two models of implementing address buffers using CAM and REG.

- CAM-CAM: In this model, both the MSB and LSB are stored in CAM. When a request address arrives, unlike a fully associative cache, both MSB and LSB comparisons are performed at the same time. As the address is separated into two parts, the comparison time for the *CoC* can be reduced by half.
- REG-CAM: The MSB is stored in REG and the LSB in CAM. In an address comparison operation, the LSB is directly compared in CAM, and the MSB is read out from REG into the comparator. When the LSB is matched, the corresponding MSB comparator is enabled. As the number of entries increases, unnecessary MSB comparison is reduced since the comparator is not enabled. The comparison operation of the CAM bit-stream is sequential. In contrast to CAM, the comparator of the bit-stream is operated in parallel. Thus, a larger CAM stream has a larger hit time, but the comparator stream gives a constant comparison time. The REG-CAM model can retain the power reduction of CAM-CAM and also reduce the comparison time. In the following section, we will discuss the effects of both implementation models.

3. PERFORMANCE EVALUATIONS

In this section, we will describe the experimental environment used in this study for performance evaluation and power estimation. Then, we will introduce the power evaluation model. Finally, we will analyze the experimental results.

3.1 Experiment Model

The experiment was based on trace-driven simulations using an object code instrumentation system called ATOM [14], which is available on the DEC Alpha AXP system. The trace files were generated on a DEC Alpha AXP 3000 workstation under DEC OSF/1 V3.2. A simulator of the cache with the *CoC* architecture was developed to collect the benchmarks statistics under various architecture configurations. Our benchmarks were composed of five applications from SPECint95 and five one from SPECfp95. Table 2 shows the benchmarks and the corresponding program characteristics of the data references generated in this study.

Three cache architectures were simulated separately: the normal L1 cache, L1 with the CAM-CAM *CoC*, and L1 with the CAM-REG *CoC*. The L1 cache configuration was direct-mapped with divided word-lines. The cache sizes that we considered were 8K,

Table 2. Benchmark characteristics.

SPECfp95			SPECint95		
Benchmark	Inst. cnt	em Data cnt	Benchmark	Inst. cnt	em Data cnt
102.swim	176,274,384	47,006,362	124.m88ksim	18,492,771	7,493,355
107.mgrid	174,225,461	48,104,890	126.gcc	80,030,168	29,109,729
110.applu	181,839,885	47,359,518	129.compress	143,482,967	20,037,295
141.apsi	133,818,460	46,659,664	132.jpeg	101,465,807	31,145,165
146.wave5	110,223,063	42,233,465	134.perl	79,683,821	34,007,394

16K, 32K, and 64K, and the block size of each cache was 32bytes. In addition, the width of the system bus was 32bits. The write policy for the *CoC* was FIFO. When the CPU sent a store request to the cache, a valid bit was modified instead of updating the *CoC* if the store was also a hit in the *CoC*.

3.2 Cache Configuration Model

The caches were mainly composed of a decoder, wordlines, bitlines and sense amplifiers. These components could vary in size as the cache size as increased or decreased. The size of the decoder is determined by the width of index bits. The wordline size was the block size. The depth of the bitline depended on the size of the decoder. Finally, the number of sense amplifiers depends on the number of wordlines. To evaluate the power dissipation in the cache, the configuration for each cache size had to be defined. In this study, we only considered the direct-map cache with divided wordlines and subbanking techniques.

Firstly, using the techniques of divided wordline and subbanking techniques, the data array can be divided into several small arrays. Based on the 8K cache, the data array for each cache size is divided into a 256×256 array, where the first 256 is the number of wordlines, and the second is the memory cell count. As a result, the 8K cache has one 256×256 data array, the 16K cache has two and so do the other configurations.

The 8K tag array is a 256×19 array, where the tag array has 256 entries and 19 bits of content. The tag array of a 16K cache is configured as 256×18 , 32K for 256×17 , and 64K for 256×16 as shown in Table 3.

Table 3. Configuration mode for each cache size.

size	data array	tag array
8K	$256 \times 256^*1$	256×19
16K	$256 \times 256^*2$	256×18
32K	$256 \times 256^*3$	256×17
64K	$256 \times 256^*4$	256×16

All the power dissipation and access time data for each component reported in this paper were obtained from the simulation results. Each building circuit except for the CAM circuit in the *CoC* was designed following the design style in the textbook [19]. To implement a low power cache, we used a recently proposed CAM circuit structure [12] in our design.

Fig. 4 shows the NAND-type matching circuit of this CAM. This low-power CAM circuit uses a typical 6-transistor memory cell [19], and an XOR gate is implemented in a CAM cell for comparing the memorized data at nodes mp and mn and the input data at nodes ip and in . The main difference between this new CAM and the other conventional CAM's is the dynamic NAND-type matching circuit adopted in this new design. The comparison result of each bit in the same row is NAND-ed to obtain the final matching result. Due to the NAND-type circuit structure, the switching activity of the match line is much less than that of the usually adopted NOR-type circuit structure, which meets the low-power requirements in this study.

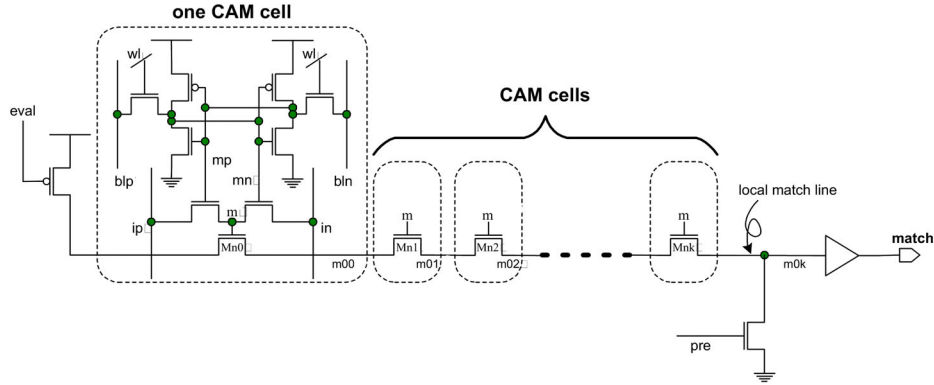


Fig. 4. NAND-type matching circuit.

3.3 Cache Power Model

The models that are typically used for power evaluation can be categorized into two groups: circuit evaluation models and high level power estimation models. Circuit evaluation uses IC design CAD tools to directly evaluate the power dissipation such like *hspice* and *power mill*. High-level evaluation can quickly estimate the power dissipation but suffers from great inaccuracy. Although circuit evaluation is more precise, it requires more time for coding and simulation. Furthermore, the circuit evaluation method is more flexible than the high-level power estimation approach. As a result, we combined the two models for our simulation. Using the cache configuration model described above, we could determine the energy consumption of each component and finally evaluate the total energy dissipation based on the cache simulation results. Based on [20] and [7], we have Eq. (4):

$$E_{total} = E_{CoC_hit} \times N_{CoC_hit} + E_{CoC_miss} \times N_{CoC_miss} + E_{L1_hit} \times (N_{L1_hit} - N_{CoC_hit}) + E_{L1_miss} \times N_{L1_miss}, \quad (4)$$

where the terms E_{CoC_hit} , E_{CoC_miss} , E_{L1_hit} and E_{L1_miss} are the energy dissipated for a single data access to caches. The *CoC* reduces the access frequency of the L1 cache when data hits in the *CoC*, which improves $N_{L1_hit} - N_{CoC_hit}$ and thus reduces the E_{total} . Each term is derived as follows:

$$\begin{aligned}
E_{L1_hit} &= E_{decoder} + E_{data\ array_r} + E_{tag\ array_r} + E_{comparator}, \\
E_{L1_miss} &= E_{decoder} + E_{tagarray_r} + E_{comparator} + E_{dataarray_w} + E_{tagarray_w}, \\
E_{CoC_hit} &= E_{LSB} \times N_{entry\#} + E_{MSB} \times N_{MSB\ hit}, \\
E_{CoC_miss} &= E_{CoC_hit} + E_{CoC_update}.
\end{aligned}$$

In above equations, we separate the energy consumption of the data bus into $E_{read\ data\ bus}$ and $E_{write\ data\ bus}$ as the levels of dissipation for reads/writes are different. In the L1 cache, the data array is accessed when the tag comparison operation is missing. $E_{L1\ cache\ miss}$ does not contain the term $E_{data\ array}$. For each tag and data array, we split the energy dissipation into the following terms:

$$E_{array} = E_{wordline} + E_{bitline} + E_{sense-amp}$$

In [20], the power dissipation for each word-line and bit-line charges according to the cache configuration model, and the power of the sense amplifier depends on the word-line length. The array energy dissipation can be divided into three terms: word-line, bit-line and sense amplifier.

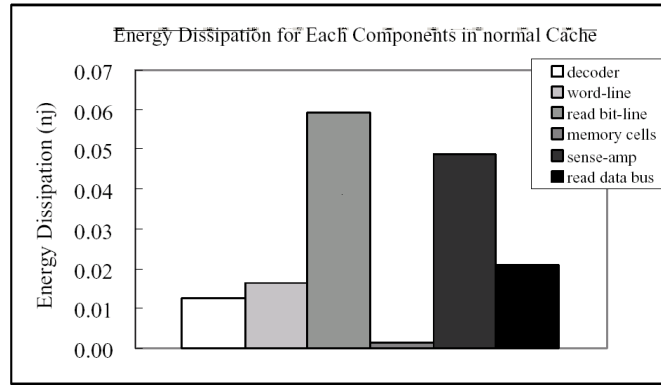


Fig. 5. Energy dissipation of each cache components.

3.4 Simulation Results

Fig. 5 summarizes the portions of power dissipation for each component when accessing the data array in the sub-banked cache architecture. These values give a relative scale when a cache read operation is performed on an 8K direct-mapped cache. They show that the bit-line and sense amplifier account for nearly 70% of the total power dissipation. Separating a large array into sub-banks (blocking the bit-line and sense-amp) may cause a large reduction of power dissipation.

Fig. 6 illustrates the power dissipation of the L1 caches and the CoC per access for each cache size. In the L1 cache, when the cache size is doubled, the energy more than doubles while dissipation of the CoC increases less significantly accordingly. This is because the number of lines in the CoC does not scale up as the L1 is cache increased. Thus the energy dissipation of the CoC is much less than that of the L1 cache.

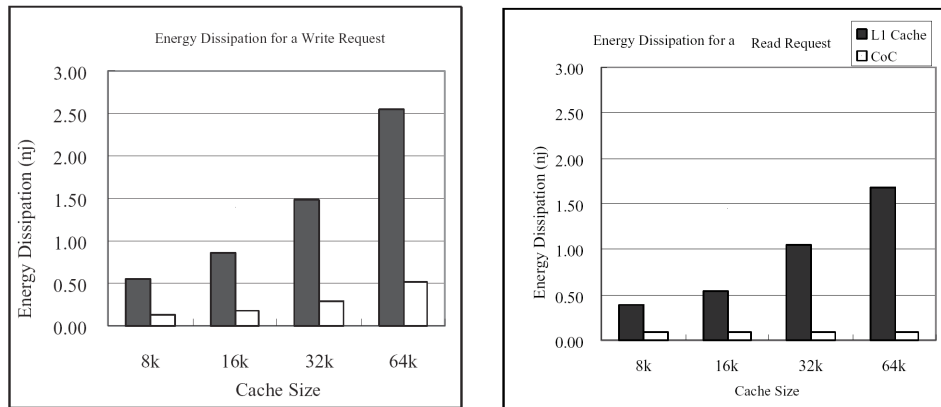


Fig. 6. Power dissipation for L1 cache and CoC.

Fig. 7 shows the energy reduction obtained with different entry numbers in the *CoC* with CAM-CAM models, and Fig. 8 compares the two CAM-CAM and CAM-REG models for an 8K cache. The amount of energy reduction is estimated for the L1 cache. As shown in Fig. 7, when the cache size increases from 8K to 64K, the amount of energy reduction is significant. A larger cache will consume more energy, but the *CoC* will not. Furthermore, the effect of varying the entry number of the *CoC* can be observed. When the entry number is one, the *CoC* can save about 20% of the energy. When the entry number exceeds four, the ratio increases only a marginally.

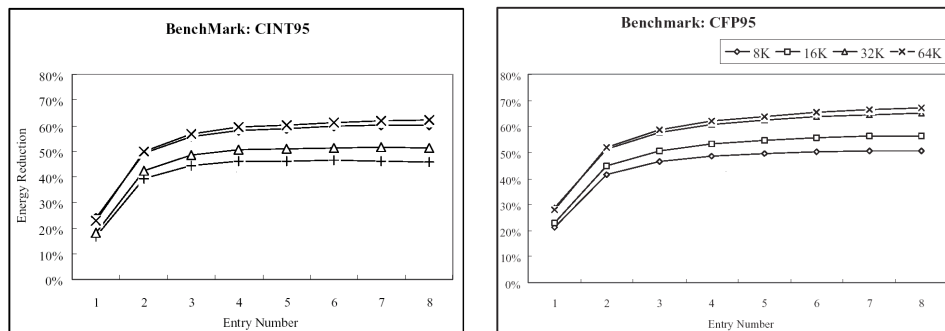


Fig. 7. CoC entry number v.s. energy reduction.

We can see that the *CoC* with four-entries is a better choice for reducing the extra hardware overhead. Fig. 8 compares the implemented *CoC* models. As the entry number increases, the energy reduction achieved with the CAM-CAM model is greater than that with the CAM-REG. In terms of the operation counts, a miss in the LSB comparison will also increase the MSB comparison in CAM-REG model. This will be more significant when the entry number of the *CoC* is large.

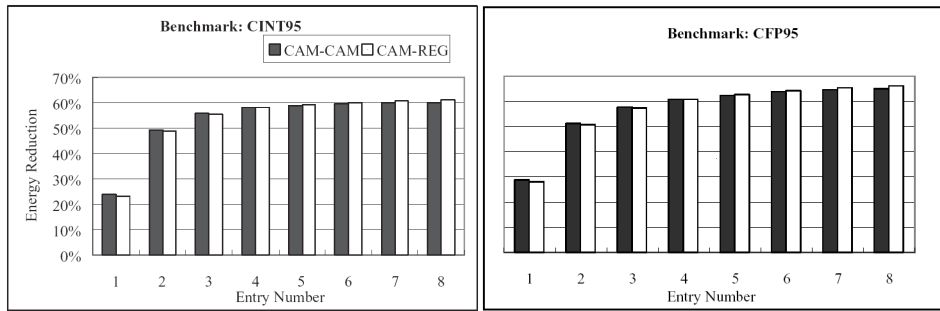


Fig. 8. Two CoC models v.s. energy reduction.

The amount of power consumption with different number of LSB-bits is shown in Fig. 9. The energy reduction ratio increase as the number of LSB-bits increases from one to six. When the LSB bit number is six, the reduction ratio reaches a saturation point. Reducing the number of LSB-bits increases the counts for the LSB comparison matching and MSB comparison operations. Thus, the amount of energy reduction achieved with the CAM-REG CoC is less significant than that achieve with the CAM-CAM CoC when the number of LSB-bits are less than four.

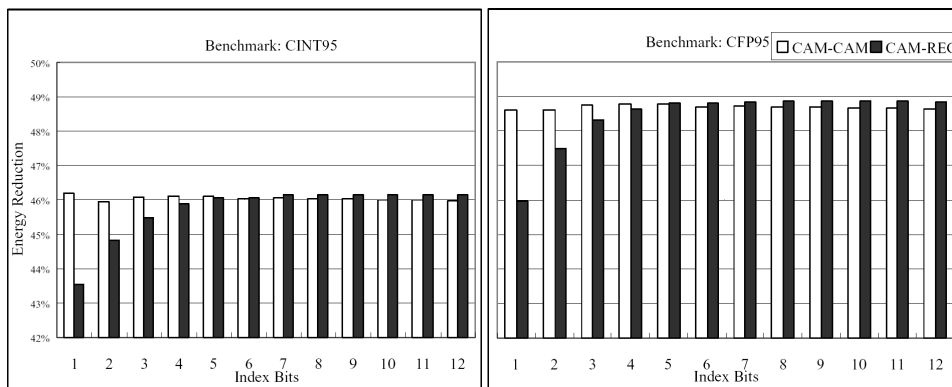


Fig. 9. Energy reduction v.s. HAC bits.

Fig. 10 shows the delay time for each LSB comparison given by either a normal comparator, CAM-CAM, or CAM-REG. The delay time is proportional to the number of LSB-bits in the CAM-REG model. On the other hand, the CAM-CAM access time decreases when the LSB-bits increases. The critical path in the CAM-CAM model is determined by the largest portion of CAM bits. As the LSB comparison time is 60% of that of the full address comparator, the access time can be reduced when an LSB miss happens.

In conclusion, from Fig. 9, the CoC with five LSB bits achieves greater energy reduction. As illustrated in Figs. 7 and 8, the CoC with 4 entries with the CAM-REG model is a better solution for cache energy reduction.

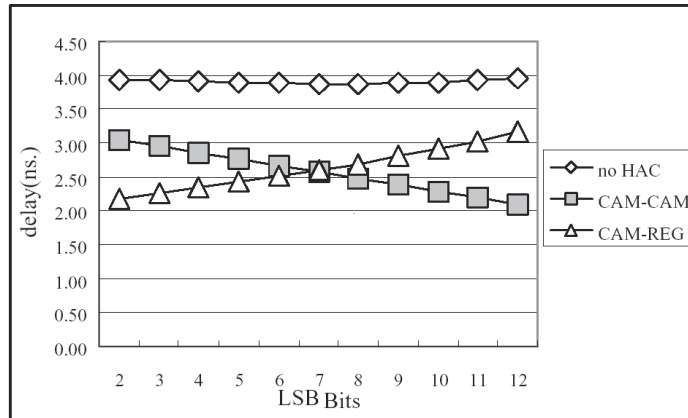


Fig. 10. HAC bits v.s. comparator delay time.

3.5 Evaluating the Delay of the CoC

The critical path of the CoC architecture is shown in Fig. 11 by the gray solid line. When data is missing in the CoC, from Fig. 11, the critical path goes through the tri-state, the TAG array, the comparator of the tag data, and the data-out registers. The extra delay of the CoC architecture is the only CAM delay for CoC address matching. We measured the delay on the critical path by running Hspice modeling on the individual components using the TSMC $.35\mu\text{m}$ process library. We obtained the access time of each component and summarize the delays in the critical path in Table 4. The overhead of the CoC architecture is only 5.6%, compared with that of the conventional cache, as shown in Table 4. In addition to the CoC address and data, the differences between the CoC and conventional cache include the tri-state and the MUX. However, the tri-state reduces the critical path delay of the CoC cache, unlike the buffer in conventional caches. This is because the tri-state delay observed is relatively smaller than the delay from the data input to the output of the buffer. The other difference is in the MUX for selecting the choice of data coming from the CoC or the normal path. Nevertheless, it is not in the critical path and the area and power are so small in the cache that they can be neglected. The CoC delay is also small compared with the accessing delay of the large capacity SRAM in the TAG array and the comparator delay. The CAM, shown in Fig. 12, used in the CoC is employed in the NOR-type structure of the match-line to achieve fast comparison. Although the energy of the NOR-type CAM is larger than that of the NAND-type CAM because of the higher switching probability, there are only four entries in the CoC, and the amount of energy in the whole cache is not significant.

Table 4. The delay penalty of the CoC path (in ns).

Critical Pth	T_{FF}	T_{CAM}	$T_{tri-state}/Buf$	$T_{SRAM}(tag)$	T_{comp}	T_{OR}	T_{FF}	Total	Penalty
Without Coc	0.8	x	0.5 (buf)	2.5	2.0	0.5	0.8	7.1	
With CoC	0.8	0.6	0.3 (tri)	2.5	2.0	0.5	0.8	7.5	5.63%

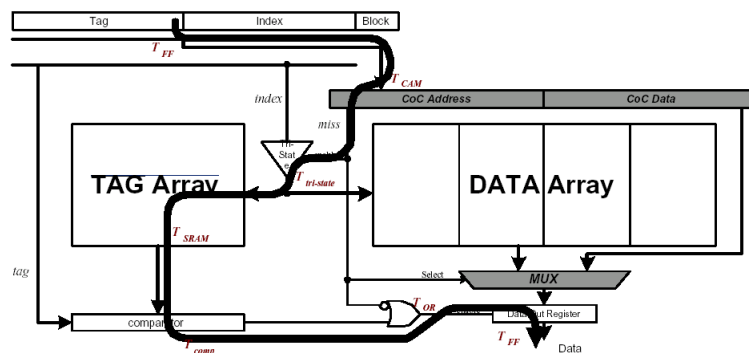


Fig. 11. The CoC architecture with critical path flow.

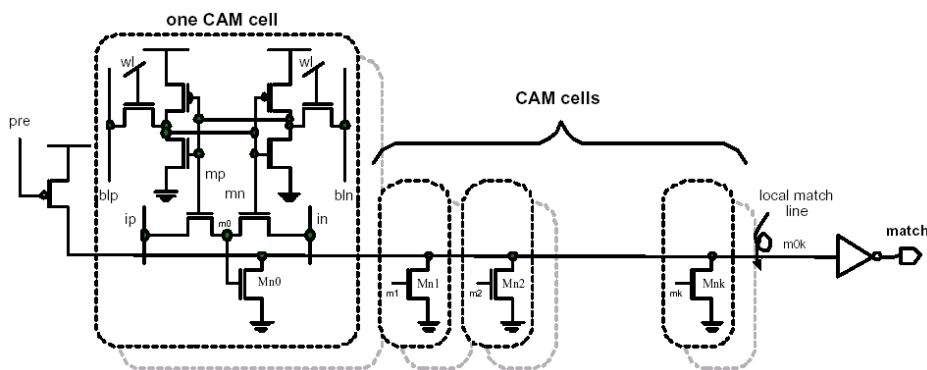


Fig. 12. The NOR-type CAM for address matching in CoC.

4. RELATED WORK

Several approaches to reducing the CMOS power consumption have been proposed. The average power consumption of CMOS technology is given by $P = \alpha C_L V_{dd}^2 f$, where C_L is the load capacitance, α is the switching activity of CMOS, V_{dd} is the supply voltage, and f is the system clock frequency. As the power consumption is quadratic proportional to the supply voltage, the most effective way to reduce power consumption is to reduce the supply voltage. The traditional power reduction approach modifies the cache architecture to reduce the switching activity or minimize the average CMOS capacitance. There is a tradeoff in the cache architecture between performance and power consumption [2].

Su and Despain proposed to reduce power consumption by using a large cache size and block buffering [16]. As most applications are executed with very high spatial locality, “buffering” the most recently fetched data blocks can reduce the number of data cache references. This is because the needed to access data accessed next is likely to be located in the same block as the most recently accessed data. Furthermore, the energy needed to access a small buffer is significantly less than that needed to access the entire cache array.

In [9], Kin and Gupta proposed a small filter that sits in front of a conventional L1 cache for the purpose of reducing cache power consumption. Because the L1 cache is accessed only when a miss happens in the filter cache, the latency when accessing the L1 cache increases when a filter cache miss happens. When 58% power reduction is achieved, cache performance is reduced by about 20%. Enlarging the filter cache size reduces the number of filter cache misses and results in a better access time for the L1 cache, but this leads to greater power dissipation.

To reduce the access latency generated by the filter cache, Ghose and Kamble proposed a line buffer for the set-associative cache [4]. Without increasing the cache access time, the proposed cache performs index decoding and number of multiple line buffers accessing at the same time. If multiple line buffers hit, the array accessing operation is canceled. In this way, the cache system performs two cache access operations at the same time but instantaneously disables either the one that becomes a miss. However, this causes a situation in which these buffer lines may lead to a larger access time than that for the index decoder. Also, those word-lines and bit-lines may have to evaluate before the disable control of the tri-state is determined.

In our proposed method, we use a tri-state to disable the index decoder before comparing addresses in the *CoC* and enabling the decoder when the *CoC* misses. In this way, the access time of the L1 cache can be increased. It is more important to reduce the access switching frequency of the tag and data arrays, which are the main components that account for power dissipation in the caches. Furthermore, to reduce the access time and power consumption overhead caused by the *CoC*, we employ a fast *HAC* method, thus taking advantage of the spatial locality property. Using *HAC*, we can optimize the power savings ratio and the access time when performing *CoC* operations. The primary difference between our approach and the solution in [9] is that our *CoC* is a linear CAM mechanism, which is coupled with the access path through tag and data arrays – a caching on the cache. Unlike the filter cache, the *CoC* does not need to provide set associative management for the CAM buffer since data enter the buffer in an FIFO fashion. The *CoC* disables wordline decoding once it matches the data in the CAM buffer. In this study, we have implemented the *CoC* controller in Verilog and measured the energy consumption and delay for each component to show the key performance differences. Previous studies did not employ the *HAC* mechanism to reduce the determination time of the tri-state control.

5. CONCLUSIONS

In this paper, we have proposed the *CoC*, an energy-efficient cache architecture for reducing the energy dissipation. To optimize the *CoC*, we use the hierarchical address comparison (*HAC*) scheme to improve the comparison time and achieve energy reduction. We have also presented two *HAC* implement models for CAM-CAM and CAM-REG. We have simulated the *CoC* in simulation experiments using the SPEC95 benchmark suites. Experimental results show that the *CoC* can effectively reduce the amount of energy dissipation by about 45%. Furthermore, using the *HAC* scheme, we can reduce the comparison time by about 40%, compared with the original *CoC* address comparison.

REFERENCES

1. G. Albera and R. I. Bahar, "Power/performance advantages of victim buffer in high-performance processors," in *Proceedings of IEEE Alessandro Volta Memorial Workshop on Low-Power Design*, 1999, pp. 43-51.
2. R. I. Bahar, G. Albera, and S. Manne, "Power and performance tradeoffs using various caching strategies," in *Proceedings of the International Symposium on Low Power Electronics and Design*, 1998, pp. 64-69.
3. N. Bellas, I. Hajj, and C. Polychronopoulos, "Using dynamic cache management techniques to reduce energy in a high-performance processor," in *Proceedings of the International Symposium on Low Power Electronics and Design*, 1999, pp. 64-69.
4. K. Ghose and M. B. Kamble, "Reducing power in superscalar processor caches using subbanking, multiple line buffers and bit-line segmentation," in *Proceedings of the International Symposium on Low Power Electronics and Design*, 1999, pp. 70-75.
5. K. Inoue, I. Ishihara, and K. Murakami, "Way-predicting set-associative cache for high performance and low energy consumption," in *Proceedings of the International Symposium on Low Power Electronics and Design*, 1999, pp. 273-275.
6. M. B. Kamble and K. Ghose, "Analytical energy dissipation models for low power caches," in *Proceedings of the International Symposium on Low Power Electronics and Design*, 1997, pp. 143-148.
7. M. B. Kamble and K. Ghose, "Energy-efficiency of VLSI caches: a comparison study," in *Proceedings of the Tenth International Conference on VLSI Design*, 1997, pp. 261-267.
8. J. Kin, M. Gupta, and W. H. Mangione-Smith, "The filter cache: an energy efficient memory structure," in *Proceedings of the 30th Annual International Symposium on Microarchitecture*, 1997, pp. 184-193.
9. U. Ko, P. T. Balsara, and A. K. Nanda, "Energy optimization of multilevel cache architectures for RISC and CISC processors," *IEEE Transactions on VLSI Systems*, 1998, pp. 299-308.
10. C. Kulkarni, F. Catthoor, and H. De Man, "Code transformations for low power caching in embedded multimedia processors," in *Proceedings of the International Parallel Processing Symposium and the Symposium on Parallel and Distributed Processing*, 1998, pp. 292-297.
11. H. Mizuno and et. al., "A 1V 100MHz 10mW cache using separated bit-line memory hierarchy architecture and domino tag comparators," *IEEE ISSCC 1996 Digest of Papers*, 1996, pp. 152-153, 434.
12. F. Shafai, K. J. Schultz, G. F. R. Gibson, A. G. Bluschke, and D. E. Sompp, "Fully parallel 30-MHz, 2.5-Mb CAM," *IEEE Journal of Solid-State Circuits*, Vol. 33, 1998, pp. 1690-1696.
13. W.-T. Shiue and C. Chakrabarti, "Memory exploration for low power embedded systems," in *Proceedings of the International Symposium on Circuits and Systems*, 1999, pp. 250-253.
14. A. Srivastava and A. Eustace, "Atom: a system for building customized program analysis tools," in *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, 1994, pp. 196-205.

15. M. R. Stan and W. P. Burlison, "Low-power encodings for global communication in CMOS VLSI," *IEEE Transactions on VLSI Systems*, 1997, pp. 444-455.
16. C.-L. Su and A. M. Despain, "Cache design tradeoffs for power and performance optimization: A case study," in *Proceedings of the International Symposium on Low Power Design*, 1995, pp. 63-68.
17. C.-L. Su and A. M. Despain, "Cache designs for energy efficiency," in *Proceedings of Hawaii International Conference on System Sciences*, 1995, pp. 306-315.
18. C.-L. Su, C.Y. Tsui, and A. M. Despain, "Low power architecture design and compilation techniques for high-performance processors," in *Proceedings of the Comcon Spring '94, Digest of Papers*, 1994, pp. 489-498.
19. N. H. E. Weste and K. Eshraghian, *Principles of CMOS VLSI Design – A Systems Perspective*, Addison Wesley Publishing Company, 1993.
20. S. E. Wilton and N. I. Jouppi, "An enhanced access and cycle time model for on-chip caches," DEC WRL Research Report 93/5, July 1994.



Hung-Cheng Wu (吳宏城) was born in Taiwan. He received the M.S. degree in National Chung Cheng University, Chiayi, Taiwan. His research interests include computer architectures and low power cache design.



Tien-Fu Chen (陳添福) was born in Kaohsiung, Taiwan, R.O.C. on August 14, 1961. He received the B.S. degree in Computer Science from National Taiwan University in 1983. After completed his military services, he joined Wang Computer Ltd., Taiwan as a software engineer for three years. From 1988 to 1993 he attended the University of Washington, receiving the M.S. degree and Ph.D. degrees in Computer Science and Engineering in 1991 and 1993 respectively. He is currently an Associate Professor in the Department of Computer Science and Information Engineering at the National Chung Cheng University, Chiayi, Taiwan. In recently years, he has published several widely-cited papers on dynamic hardware prefetching algorithms and designs. His current research interests are computer architectures, distributed operating systems, parallel and distributed systems, and performance evaluation.



Hung-Yu Li (李鴻瑜) was born in Taiwan in 1974. He received the B.S. degree in electrical engineering from the Tatung University in 1996. He is currently working toward the Ph.D. degree at the Institute of Electrical Engineering, National Chung Cheng University. His research interests include low-voltage and low-power flip-flop and memory circuits design.



Jinn-Shyan Wang (王進賢) (S'85-M'88) was born in Taiwan in 1959. He received the B.S. degree in electrical engineering from the National Cheng-Kung University in 1982 and the M.S. and Ph.D. degrees from the Institute of Electronics, National Chiao-Tung University, Taiwan, in 1984 and 1988, respectively.

During 1988-1995 he was with Industrial Technology Research Institute (ITRI), engaged in the ASIC circuit and system design and became the manager of the Department of VLSI Design. He joined the Department of Electrical Engineering, Chung-Cheng University, Taiwan, in 1995, where he is now a Full Professor. His research interests have been in low-power and high-speed digital integrated circuits and systems, analog integrated circuits, IP and SOC design, and CMOS image sensors. He has published over 20 journal papers and 40 conference papers and holds over 20 patents on VLSI circuits and architectures.