Design and Analysis of Low-Power Cache Using Two-Level Filter Scheme

Yen-Jen Chang, *Member, IEEE*, Shanq-Jang Ruan, *Member, IEEE*, and Feipei Lai, *Senior Member, IEEE*

***Abstract*—Power consumption is an increasingly pressing problem in modern processor design. Since the on-chip caches usually consume a significant amount of power, it is one of the most attractive targets for power reduction. This paper presents a *two-level filter* scheme, which consists of the *L1* and *L2 filters*, to reduce the power consumption of the on-chip cache. The main idea of the proposed scheme is motivated by the substantial unnecessary activities in conventional cache architecture. We use a single block buffer as the L1 filter to eliminate the unnecessary cache accesses. In the L2 filter, we then propose a new *sentry-tag* architecture to further filter out the unnecessary way activities in case of the L1 filter miss. We use *SimpleScalar* to simulate the *SPEC2000* benchmarks and perform the *HSPICE* simulations to evaluate the proposed architecture. Experimental results show that the *two-level filter* scheme can effectively reduce the cache power consumption by eliminating most unnecessary cache activities, while the compromise of system performance is negligible. Compared to a conventional instruction cache (32 kB, two-way) implemented with only the L1 filter, the use of a two-level filter can result in roughly 30% reduction in total cache power consumption. Similarly, compared to a conventional data cache (32 kB, four-way) implemented with only the L1 filter, the total cache power reduction is approximately 46%.**

***Index Terms*—Block buffer, filter scheme, low-power cache, power consumption, unnecessary cache activity.**

I. Introduction

SINCE AN on-chip cache can effectively reduce the speed gap between processor and main memory, almost modern microprocessors employ it to boost system performance. For high clock frequency, these on-chip caches are implemented using arrays of densely packed static random-access memory (SRAM) cells. The number of transistors devoted to the on-chip caches is often a significant fraction of the total transistor budget for the entire chip. As the on-chip cache size keeps increasing, the power dissipated by the on-chip caches becomes significant (e.g., 25% of the total chip power in the DEC 21164 [1], 43% of the total power in the SA-110 [2]). This trend will likely continue as processors become more sophisticated and provide higher performance.

Manuscript received June 24, 2002; revised October 14, 2002.

Y.-J. Chang is with the Computer Science and Information Engineering Department, National Taiwan University, Taipei, Taiwan 106, R.O.C. (e-mail: d88017@csie.ntu.edu.tw).

S.-J. Ruan was with the Electrical Engineering Department, National Taiwan University, Taipei, Taiwan 106, R.O.C. He is now with Synopsys Inc., Taipei, Taiwan 110, R.O.C.

F. Lai is with the Computer Science and Information Engineering and Electrical Engineering Departments, National Taiwan University, Taipei, Taiwan 106, R.O.C. (e-mail: flai@cc.ee.ntu.edu.tw).

Digital Object Identifier 10.1109/TVLSI.2003.812292

As mentioned above, cache is one of the most attractive targets for power reduction. There have been several techniques for reducing the power consumption of on-chip caches. Against the traditional concurrent access flow, Hasegawa *et al.* [3] proposed a phased cache with a serial access scheme, where tag comparison is followed by the data arrays read so that only the required data is actually read out from the data array. However, the phased cache suffers from longer cache hit time. A new tag architecture and tag-skipping technique proposed by Choi *et al.* [4] reduces the number of unnecessary tag lookups and, thus, the power consumption in the embedded system-on-chip (SOC) design. Filter cache [5], L-cache [6], block buffering [7], and multiple line buffer [8] attempt to reduce power consumption by placing a small cache (i.e., L0 cache) or output latches between the processor and the L1 cache. If the L0 cache or output latches can serve most L1 cache requests, then L1 cache activity can be greatly reduced, thereby saving power. Cache sub-banking was also presented in [7] and [8], in which the data memory array of the cache is divided into several sub-banks. In each cache access, only those sub-banks that contain the desired data can be read out. In [9], Albonesi exploited the subarray partitioning of set associative caches and proposed a selective cache ways method that can disable a subset of cache ways during periods when full cache is not required to achieve good performance. In addition to hardware modification, however, the selective cache ways method requires a lot of software supports, including special instructions and some specific software for analyzing application cache requirements. The way-predicting set-associative cache [10] reduces the power dissipation by accessing only a single predicted cache way instead of accessing all the cache ways. Since the entire cache would be activated as a conventional set-associative cache in case of prediction miss, the performance/power efficiency of the way-predicting cache largely depends on the accuracy of the way prediction.

In this paper, we are interested in exploring low-cost solution to reduce the cache power consumption, which is software independent and requires a little hardware overhead as well as slight architecture modification. We propose a *two-level filter* scheme that combines a *block buffer* with a new *sentry-tag* architecture. In a level-one (L1) filter, the block buffer is used to exploit the spatial locality of reference to reduce the unnecessary cache accesses. The use of a block buffer is a well-known technique, but it is only beneficial for the application with superior spatial locality. Consequently, in a level-two (L2) filter, we propose the *sentry tag* to filter out the unnecessary way activities in case of the L1 filter miss. By using the L2 filter to access only those possible hit ways, instead of accessing all the ways, the cache power consumption can be further reduced. To understand the effect of

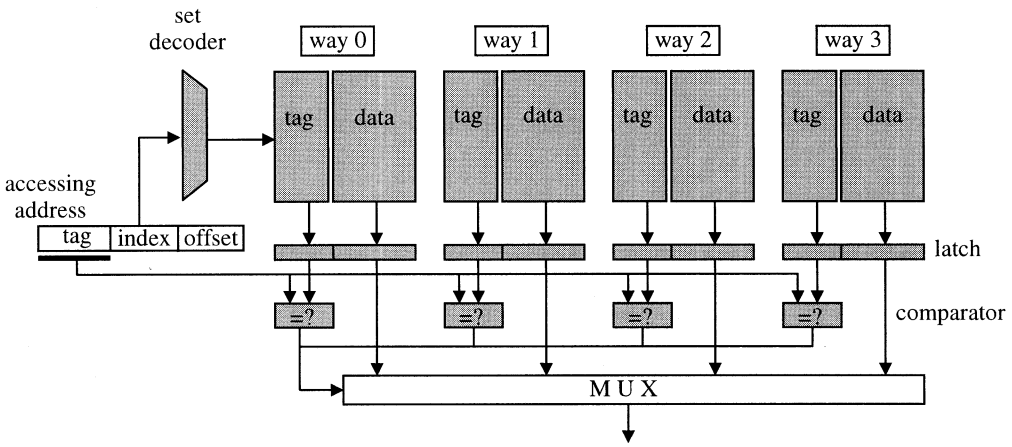


Fig. 1. Conventional four-way set-associative cache architecture. (The gray blocks represent the active components.)

the L2 filter, we develop an analytic model, and verify it with experimental results. Compared to the previous work [9], [10], the proposed *two-level filter* scheme does not require any software support and retains the fixed cache access time. The major differences from the preliminary version of this study [11] are that we further develop an analytic model to evaluate the efficiency of the proposed scheme, and provide a more detailed power and performance estimation in this paper.

The remainder of this paper is organized as follows. Section II identifies the problems of the conventional implementation of the set-associative caches. Next, in Section III, we describe the details of the cache architecture with our proposed *two-level filter* scheme, and provide an analytic model for the filter performance. In Section IV, we give a detailed power estimation model for the proposed architecture. Experimental results are given in Section V, and Section VI offers some conclusions.

II. Conventional Set-Associative Cache

Fig. 1 shows the general implementation of a conventional four-way set-associative cache. The CPU issues an address to the cache consisting of three parts, i.e., tag, index, and offset. Consider an A-way set associative cache, with a size of C bytes and a block size of B bytes. Since the number of sets is $S = C/(B \times A)$, the length of index is $\log_2(S)$ bits, which is used to index the set from which the data will be retrieved. The length of offset is $\log_2(B/4)$ bits, which is used to select the appropriate word (1 word = 4 B) within a block. Finally, the tag part is used to check whether the current access is hit or miss.

To further minimize the access delay, the data arrays of the cache are accessed concurrently with the tag arrays, and then the result of tag comparison is used to select the required block. In other words, in a four-way set-associative cache, there are always four-way activities per cache access, as shown by the gray blocks in Fig. 1. The conventional parallel access scheme used in the set-associative cache is good for the performance, but it is not optimized from the viewpoint of power consumption. This is because the parallel data arrays access before knowing the result of tag comparison would result in a lot of unnecessary way activities and, thus, large power consumption.

For example, suppose that the tag of accessing address is "$x, \ldots, x1$." The selected set contains four blocks (i.e., it is a four-way set-associative cache) and the contents of the tag array are "$x, \ldots, x0$," "$x, \ldots, x0$," "$x, \ldots, x0$," and "$x, \ldots, x1$," respectively. It is obvious that the required data is not within *ways* 0, 1, and 2 since the least significant bit of tag for these ways is "0," which is not equal to that of the accessing address (in this case, "1"). Thus, the *ways* 0, 1, and 2 are *unnecessary way activities* in this access. If we know this result before starting the conventional cache access, we may only enable *way* 3 to be accessed instead of accessing the entire cache. As the degree of associativity becomes larger, the number of unnecessary way activities tends to increase, and thus, so does the power consumption.

III. Two-Level Filter Scheme

In this section, we propose a simple and effective *two-level filter* scheme to reduce the number of unnecessary cache activities and, thus, the corresponding unnecessary power consumption. Instead of direct access [as shown in Fig. 2(a)], we use two filters concurrently to reduce the number of unnecessary cache activities [as shown in Fig. 2(b)], in which the level-one (*L1*) filter and level-two (*L2*) filter are a *single block buffer* and *sentry tag*, respectively.

A. Level-One (L1) Filter: Single Block Buffer

In the conventional cache architecture described in the previous section, the unit of cache access is a block. The range of block size is usually from 4 to 16 words in current processors. For applications with spatial locality, the next access data are likely to be located in the same block as the last access. We can take advantage of spatial locality to add one output latch to reduce the number of unnecessary cache access. In other words, if the cache block being accessed currently is still resident in the block buffer, the required data can be fetched from the block buffer directly without the normal cache access.

Caches with a single block buffer were introduced by Su and Despain [7], and extensive research [8] had shown that the use of a small number of block buffers is very efficient in reducing the power consumption of caches. They showed that a power saving of 40%–50% can be easily achieved by using eight block buffers. The decrease in power consumption with the increased number of block buffers is as expected, but using beyond one

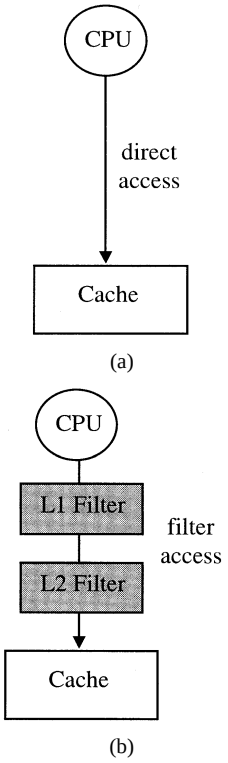


Fig. 2. (a) Conventional cache architecture. (b) Cache architecture with our proposed two-level filter scheme.

block buffer, the power saving is not as much as the use of one block buffer and might complicate the implementation of replacement. In fact, the use of one block buffer can result in roughly 40% reduction in cache power consumption, thus, we decide to use a single block buffer in this paper.

B. Level-Two (L2) Filter: Sentry Tag

From both power-saving and performance-improvement aspects, the use of a block buffer is indeed efficient, but the amount of power saving strongly depends on the program behavior. The higher spatial locality the access stream possess (e.g., instruction reference), the larger the amount of power that can be saved. This characteristic is not good for those programs with poor spatial locality. The key idea of our proposed *L2 filter* architecture is to reduce the unnecessary way activities in the case of block buffer miss, i.e., *L1 filter* miss. Thus, the cache power consumption can be further decreased.

The *sentry bit* is defined as an identifier for each cache block. We first choose some tag bits to be sentry bits and then remove them from the tag array to the *sentry-tag* storage. By pre-comparing the sentry bits of the accessing address with the sentry-tag contents stored in the selected set, this *L2 filter* scheme can effectively identify which way activities are unnecessary and then disable these cache ways in the following cache access. The content of the sentry tag would be updated when the required block is reloaded from the lower level memory during a cache miss.

For example, let the sentry bit of the current access address be "1," and the selected set contains four blocks (i.e., it is a four-way set-associative cache), which sentry bits are "0," "1," "0," and "0," respectively. Clearly, there is an impossible hit in *way* 0, *way* 2, and *way* 3 so we can disable these three ways. For *way* 1, since the sentry bit is "1," this match implies that *way* 1 potentially contains the required data. Consequently, in this case, we can reduce the number of way activities from 4 to 1 and save the power consumption corresponding to the unnecessary way activities. Unlike [9], where the selective cache ways method needs software support for analyzing application cache requirements and enabling cache ways appropriately, our proposed scheme does not require any software support. Thus, we can apply the proposed sentry tag to the processors without modifying the existing operating system and instruction set architecture (ISA).

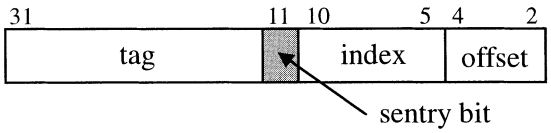


Fig. 3. Address space partition of a 8-kB four-way cache with one sentry bit.

Note that more than one match in the sentry bit comparison is possible. This means that our scheme does not guarantee the elimination of all unnecessary way activities. Any tag bits can be used as the sentry bits. Due to the spatial locality property of references, the lower order bits of the tag is more sensitive than the higher order bits in detecting the reference address variation. The simplest choice is to use the least significant bit of the tag part as the 1-b sentry (e.g., A[11] in Fig. 3). The more bits are used as sentry bits, the more accurate in filtering out the unnecessary way activities. In the following section, we will evaluate the impact on L2 filter performance for various numbers of sentry bits.

C. Analytic Model for Sentry Tag

Ideally, given the number of sentry bits (S), the way activities in each access (average way activities W_{Ave}) can be expressed in terms of the hit ratio (HR) and the number of cache ways (W), as shown in (1). For each access, there are two possible results: hit or miss. First, in the cache hit, the hit way must be activated, and there should be $(W-1)/2^S$ activated ways in the remainder $(W-1)$ ways due to the number of sentry bits S. Thus, the number of average way activities is $\mathrm{HR}\times(1+(W-1)/2^S)$ in the cache hit, which is the first part of (1). Similarly, in case of miss, the number of average way activities is $\mathrm{MR}\times W/2^S$ [as shown in the second part of (1)], in which the miss rate (MR) is equal to $(1-\mathrm{HR})$. For example, if S is zero, the average way activities is W. That is, all of the ways should be accessible in the caches without the proposed sentry tag. In another case, if $S=1$ and $W=4$, the average way activities is $2+0.5\,\mathrm{HR}$. Thus, we can save $4-(2+0.5\,\mathrm{HR})$ unnecessary way activities in one access as follows:

$$\begin{aligned} W_{\mathrm{Ave}} &= \mathrm{HR}\times\left(1+(W-1)\times\frac{1}{2^S}\right)+(1-\mathrm{HR})\times W\times\frac{1}{2^S} \\ &= \mathrm{HR}+\frac{\mathrm{HR}\times W}{2^S}-\frac{\mathrm{HR}}{2^S}+\frac{W}{2^S}-\frac{\mathrm{HR}\times W}{2^S} \\ &= \frac{W}{2^S}+\left(1-\frac{1}{2^S}\right)\times\mathrm{HR}. \end{aligned} \tag{1}$$

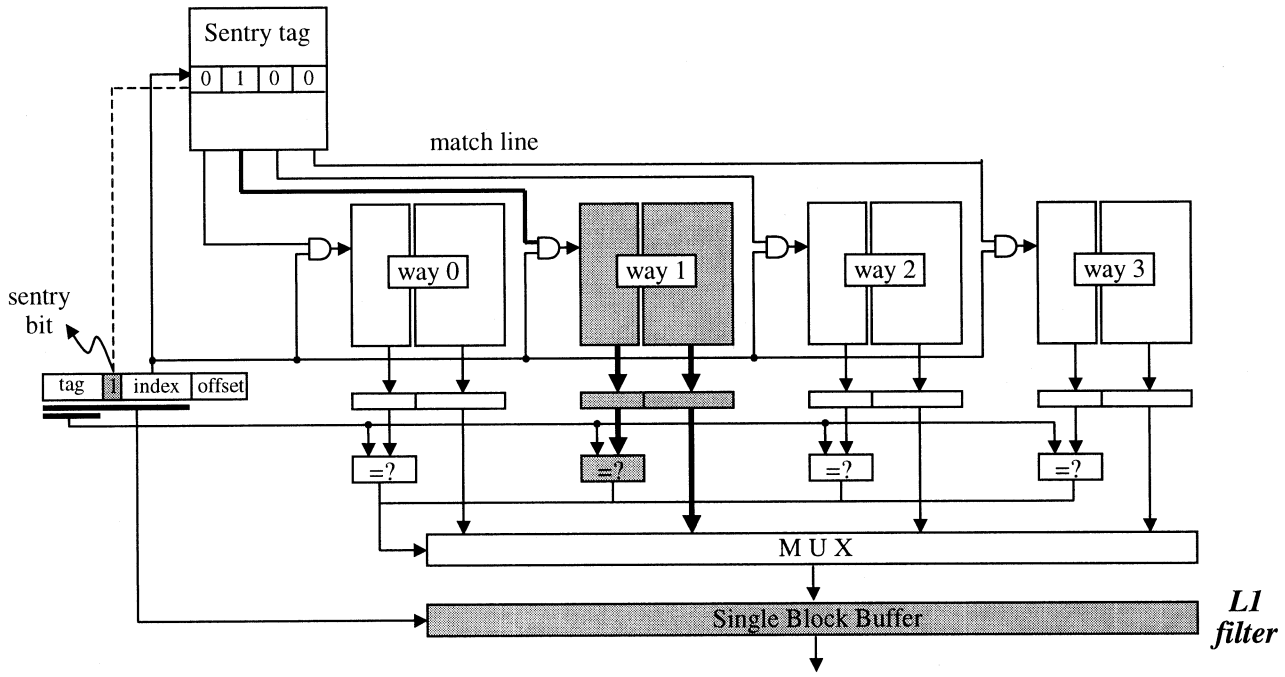


Fig. 4. Two-level filter scheme. A four-way set-associative cache architecture with a block buffer and a 1-b sentry tag. (The gray blocks symbolize an active component.)

We then define the average filter rate (FR) as the ratio of the average unnecessary way activities to the number of cache ways. By definition, the average filter rate is given by (2). The higher FR means that the sentry tag is more efficient in filtering out the unnecessary way activities. From (2), with the given HR and W, the filter rate will increase with the number of sentry bits. It certifies that the more bits are used as sentry bits, the more accurate in filtering out the unnecessary way activities. Suppose, for example, the hit ratio is 0.98, the average filter rate of a four-way cache with a 2-b sentry tag is 0.56. If we increase the number of sentry bits to 3 b, the filter rate would be increased to 0.66. In Section V, the accuracy of this analytic model for the average filter rate would be verified with the experimental results as follows:

$$\begin{aligned}\mathrm{FR} &= \frac{W - W_{\mathrm{Ave}}}{W} \\ &= 1 - \frac{\frac{W}{2^S} + \left(1 - \frac{1}{2^S}\right) \times \mathrm{HR}}{W} \\ &= \left(1 - \frac{1}{2^S}\right) \times \left(1 - \frac{\mathrm{HR}}{W}\right). \end{aligned} \tag{2}$$

D. Cache Architecture With Two-Level Filter

Fig. 4 depicts a four-way set-associative cache with the proposed *two-level filter*. Compared to the conventional set-associative caches, the hardware augmentations include a single block buffer, a sentry tag, and the control circuit. We use the transistor number as measurement in the following hardware (or area) overhead analysis.

1) In the block buffer, we use a 9T content addressable memory (CAM) cell to implement the tag part (the width is 27 b), and the data part can be implemented with the 8T latch, in which the width is the same as the block size (i.e., 256 b fixed in this paper). Hence, the area overhead of this block buffer is roughly $(9{*}27) + (8{*}256) = 2291$ transistors.
2) We must remove the sentry bits from the tag array to the sentry-tag storage. To minimize the comparison delay, we use the 9T CAM cell to implement the sentry tag. Thus, the area overhead of the sentry tag is $3{*}N_{\mathrm{ST}}{*}N_{\mathrm{B}}$ transistors, in which value 3 is the difference between the 9T CAM cell and 6T SRAM cell. N_{ST} is the number of sentry bits and N_{B} is the number of cache blocks.
3) We need an additional control circuit to enable/disable the cache way. In the conventional cache shown in Fig. 5(a), the cache way is accessible when the word line is asserted. Note that the word line is derived from the set decoder directly. We can add an AND gate to control whether the selected word line should be asserted or not. As shown in Fig. 5(b), the cache way is accessible when the decoder line and the match line are asserted concurrently. The match line is the output of sentry tag, as shown in Fig. 4. The number of AND gates used in our architecture is $S{*}A$, in which S is the number of cache sets and A is the cache associativity and, thus, the area overhead is approximately $6{*}S{*}A$ transistors.

For a 32-kB two-way cache with a block size of 32 B, the cache area spent in the tag and data arrays is approximately $(1024{*}18{*}6) + (1024{*}256{*}6) = 1\,683\,456$ transistors. The value 1024 is the block number. If the number of sentry bits is three, based on the above area analysis (a)–(c), the area overhead in our two-level filter scheme is approximately $2291 + (3{*}3{*}1024) + (6{*}512{*}2) = 17\,651$ transistors. Since

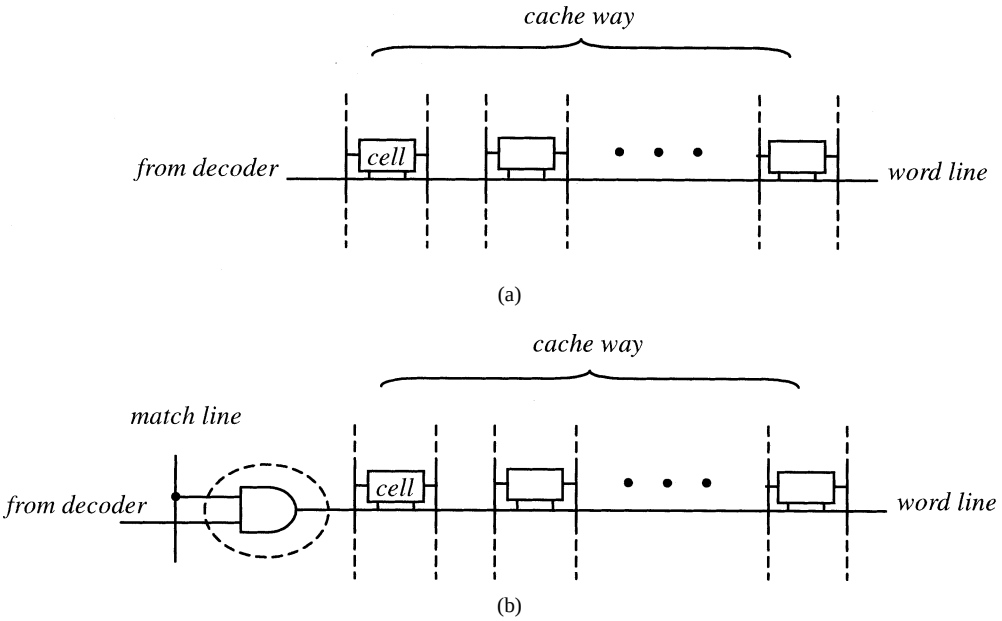


Fig. 5. Control circuit of a: (a) conventional cache and (b) cache with a sentry tag.

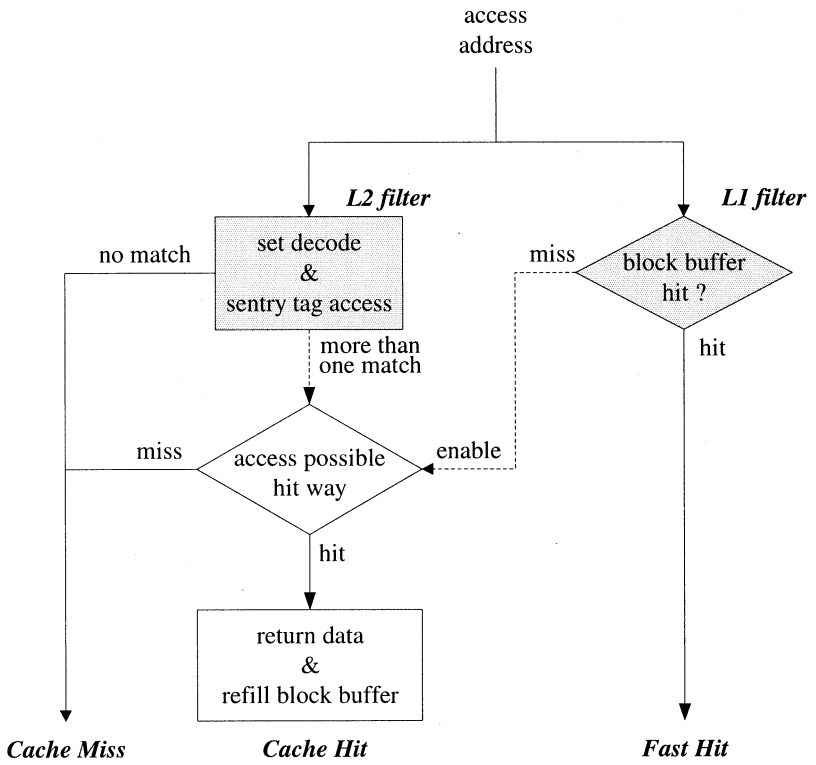


Fig. 6. Access flow in the cache architecture with two-level filter scheme.

the overhead is around 1% of the cache area, it is negligible. The access flow of an A-way cache with a two-level filter scheme is shown in Fig. 6 and is described in the following steps.

Step 1) The access address is concurrently fed into the L1 and L2 filters. We use the L1 filter to check whether the required data is still resident in the block buffer. At the same time, the set decoding and the sentry bits comparison in the L2 filter are also completed in order. Here, to minimize the delay penalty in filtering out the unnecessary cache accesses and way activities, we overlap the L1 filtering with the L2 filtering.

Step 2) *Case 1:* If a hit occurs in the L1 filtering, this is a fast hit. We can skip this cache access, and the required data is directly read from the block buffer.

Case 2: In case of the L1 filter miss, we must use the match results in the L2 filtering to trigger the corresponding ways to read out the blocks that potentially contain the required data. There are two cases in the L2 filtering. If no match occurs in the L2 fil-

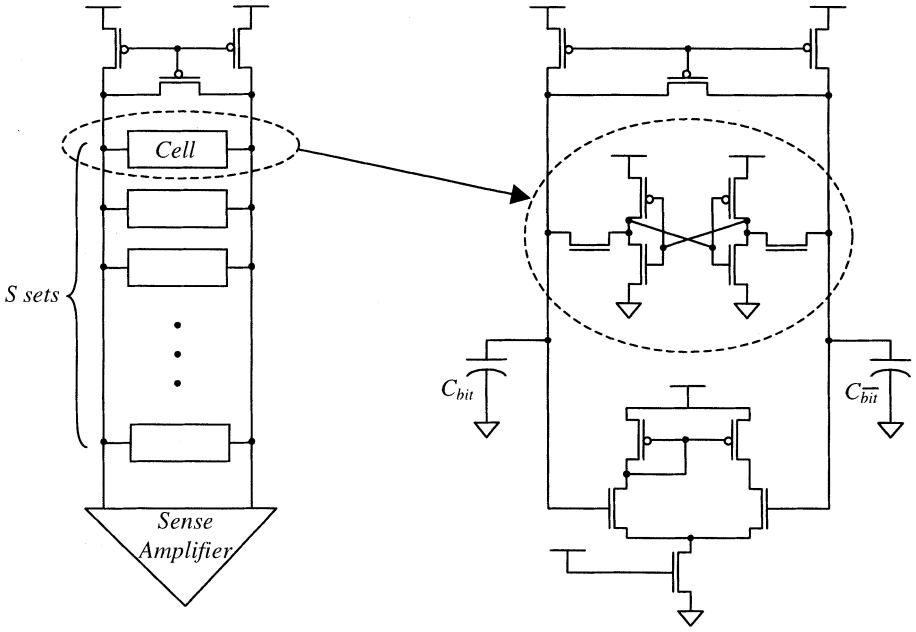


Fig. 7. Column circuit.

tering, this access must be a miss. We can then abort the following cache access and reload the required block from the lower level memory. Otherwise, step 3 must be executed.

Step 3) Perform the remainder tag comparison, as in a conventional set-associative cache. Instead of comparing the tag of the access address in parallel with the A outputs from the tag arrays (using A independent comparators), we only examine the tag of the blocks that potentially contain the required word.

Compared to the conventional access flow, our two-level scheme would induce a delay penalty because we must filter out the unnecessary cache activities before the normal cache access. The detailed analysis of the delay penalty will be addressed in Section V. Unlike the way-predicting method [10], in which the cache access time is variable, the access time of the cache with the proposed scheme is fixed. In the way-prediction method, the cache access can be completed in one cycle in case of prediction hit, but an extra cycle would be incurred in case of prediction miss. Although the high prediction-hit rate can improve the average cache access time, the penalty cannot be reduced in the worst case. By contrast, the fixed access time in our method can simplify the processor implementation.

IV. Power Estimation

In this section, we provide the detailed power estimation for various components used in the cache with the proposed two-level filter scheme. For accurate measurement of the power dissipation in various cache components, we use a 0.18-μm technology with 1.8-V voltage supply to perform the *HSPICE* simulations in the following power analysis. As shown in Fig. 4, there are three major components in our architecture, i.e., a single block buffer, sentry tag, and cache memory. Since they are independent of each other, we analyze them separately. Specifically, the power consumption of cache memory can be simplified as $P_{\text{Cache}} \approx P_{\text{way}} \times A$, in which P_{way} is the power consumption per cache way, and A is the degree of associativity (way number). According to the results in [12], the bitline and sense amplifier are by far the most power-consuming part of the cache. They contribute over 70% to the total cache power consumption. Consequently, we only consider the bitline and sense amplifier for simplification.

A. Power Consumption per Cache Way

Fig. 7(a) shows one column circuit that consists of two bitlines (*bit* and *bitbar*), S memory cells, and a sense amplifier, where S is the number of sets. Usually, S is very large, and here we do not consider the techniques of splitting horizontally data array for shorter bitlines. For simplification, instead of all memory cells, we can use an equivalent load capacitance to estimate the power dissipation of each column. Thus, Fig. 7(a) can be further reduced to Fig. 7(b). Based on [13], the effective load capacitance of the bitline during precharging, i.e., C_{bit}, is given by

$$C_{\text{bit}} = C_{\overline{\text{bit}}} = (S-1) \times \left(\frac{1}{2} \times C_{d,\text{pass}} + C_{\text{bitmetal}} \right)$$

where S is the number of sets, $C_{d,\text{pass}}$ is the drain (or junction) capacitance of the pass transistor, and C_{bitmetal} is the metal line capacitance over the extent of a single bit cell. The drain capacitance of each pass transistor is divided by two since it is shared between two vertically adjacent cells.

As the cache size increases and the degree of associativity becomes smaller, the set number S tends to increase and so does the power consumption of each column. For various set numbers, the power consumption of each column are obtained from *HSPICE* simulations and are shown in Table I. It is obvious that the read power consumption (RP_{col}) is slightly larger than the write power consumption (WP_{col}). This result can be

TABLE I
COLUMN POWER CONSUMPTION FOR VARIOUS SET SIZES

S	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}	2^{11}	2^{12}
$RP_{col}(mW)$	0.2233	0.2237	0.2246	0.2254	0.2286	0.2338	0.2454	0.2702	0.3298	0.4385	0.7025	1.3135
$WP_{col}(mW)$	0.2144	0.2148	0.2133	0.2119	0.2126	0.2174	0.2258	0.2459	0.3002	0.3946	0.6323	1.1821

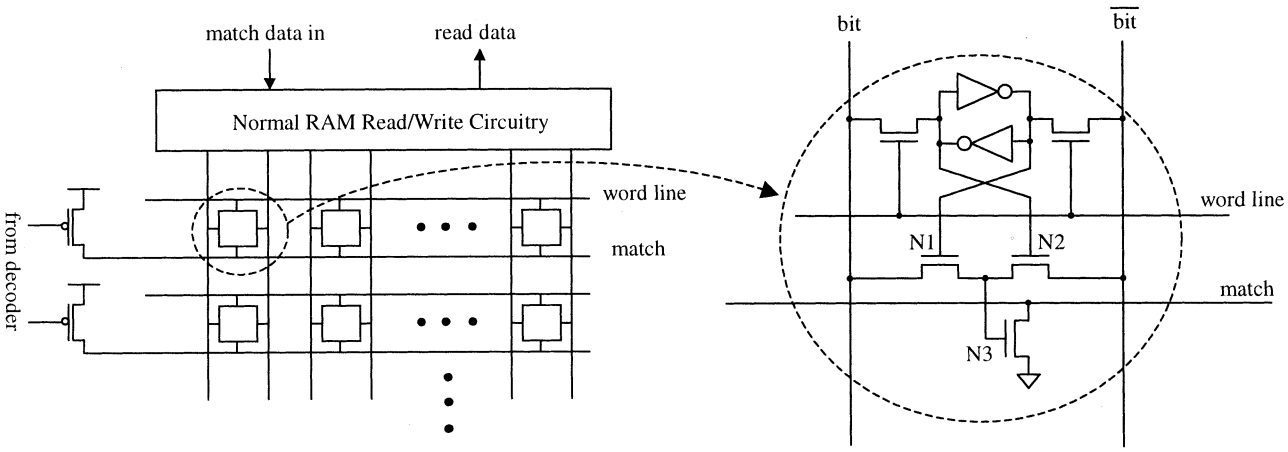


Fig. 8. Sentry-tag architecture.

confirmed by [14]. Although the power-consumption difference between read and write operations is small, for a more accurate estimation, we consider them separately in this paper. The average power consumption of one cache way for a conventional cache and our proposed architecture are given by

$$\begin{aligned} P_{\mathrm{WAY\_Conv}} &= \left[(\mathrm{RP_{col}}\times N_{\mathrm{col}})\times \mathrm{RR}\right] \\ &\quad + \left[(\mathrm{WP_{col}}\times N_{\mathrm{col}})\times \mathrm{WR}\right] \\ &= \left[\mathrm{RP_{col}}\times (T+B\times 8)\times \mathrm{RR}\right] \\ &\quad + \left[\mathrm{WP_{col}}\times (T+B\times 8)\times \mathrm{WR}\right] \qquad (3) \\ P_{\mathrm{WAY\_2L}} &= \left[(\mathrm{RP_{col}}\times N_{\mathrm{col}})\times \mathrm{RR}\right] \\ &\quad + \left[(\mathrm{WP_{col}}\times N_{\mathrm{col}})\times \mathrm{WR}\right] \\ &= \left[\mathrm{RP_{col}}\times ((T-N_{\mathrm{ST}})+B\times 8)\times \mathrm{RR}\right] \\ &\quad + \left[\mathrm{WP_{col}}\times ((T-N_{\mathrm{ST}})+B\times 8)\times \mathrm{WR}\right]. \qquad (4) \end{aligned}$$

T and N_{ST} are the number of tag bits and sentry bits and B is the block size. Note that the column number (N_{col}) includes two parts, i.e., tag and data. $\mathrm{RP_{col}}$ and $\mathrm{WP_{col}}$ are the read and write power consumption per column, respectively. RR is the rate of read operations to the total cache accesses. WR is the rate of write operations to the total cache accesses. In the instruction cache (IC), the proportion of RR to WR is $1:0$ (i.e., all cache accesses are read operation), but in the data cache (DC), the proportion of RR to WR is approximately $2:1$ [15]. Actually, the difference between these two power equations is negligible if the N_{ST} value is small.

B. Power Consumption of Block Buffer

In our proposed scheme, the use of the L1 and L2 filters would induce additional power consumption. We first analyze the power consumption in the L1 filtering, i.e., P_{L1}. In fact, P_{L1} consists of comparison power and data output power, for which values obtained from the *HSPICE* simulation are 0.6 and 7.75 mW, respectively and, thus, P_{L1} is 8.35 mW.

C. Power Consumption of Both Sentry Tag and Control Circuit

As to the power consumption in the L2 filtering, i.e., P_{L2}, since the sentry bits comparison is very critical in our scheme, we use CAM to implement the sentry tag. A typical CMOS CAM memory cell is shown in Fig. 8. A match operation proceeds by placing the data to be matched on the bit lines, but not asserting the word line. If they are not equal, the match line is discharged to low by N_3. Otherwise, it remains in its precharged state, i.e., high. Since all the cells in one entry share a single match line, as shown in Fig. 8, the match line remains high if and only if a "match" occurs in all the cells.

Note that the match signal is used to trigger the cache way corresponding to this sentry bits and enable it to be accessible. These additional control circuits in the L2 filter also induce power consumption. We must consider the sentry tag and the control circuit together. Fig. 9(a) shows the control logic used in our architecture. Obviously the major parts of power consumption in the control circuit are the word line (WL) and the match line (ML). The word line capacitance (C_{WL}) is approximately equal to the sum of gate capacitances of each memory cell in the row, and the match line capacitance (C_{ML}) is the sum of gate capacitances of each AND gate in the column. Thus, Fig. 9(a) can be reduced to Fig. 9(b). Depending on the cache configuration, we can calculate the capacitance C_{WL} and C_{ML}, and then estimate the power consumption of the sentry tag with a control circuit.

For each way, the power consumption of the sentry tag with control circuit $(P_{\mathrm{ST\_1}})$ are summarized in Table II. In this simulation, we use the baseline of a 32-kB two-way cache, and the number of sentry bits is varied from 1 to 8. Therefore, the total

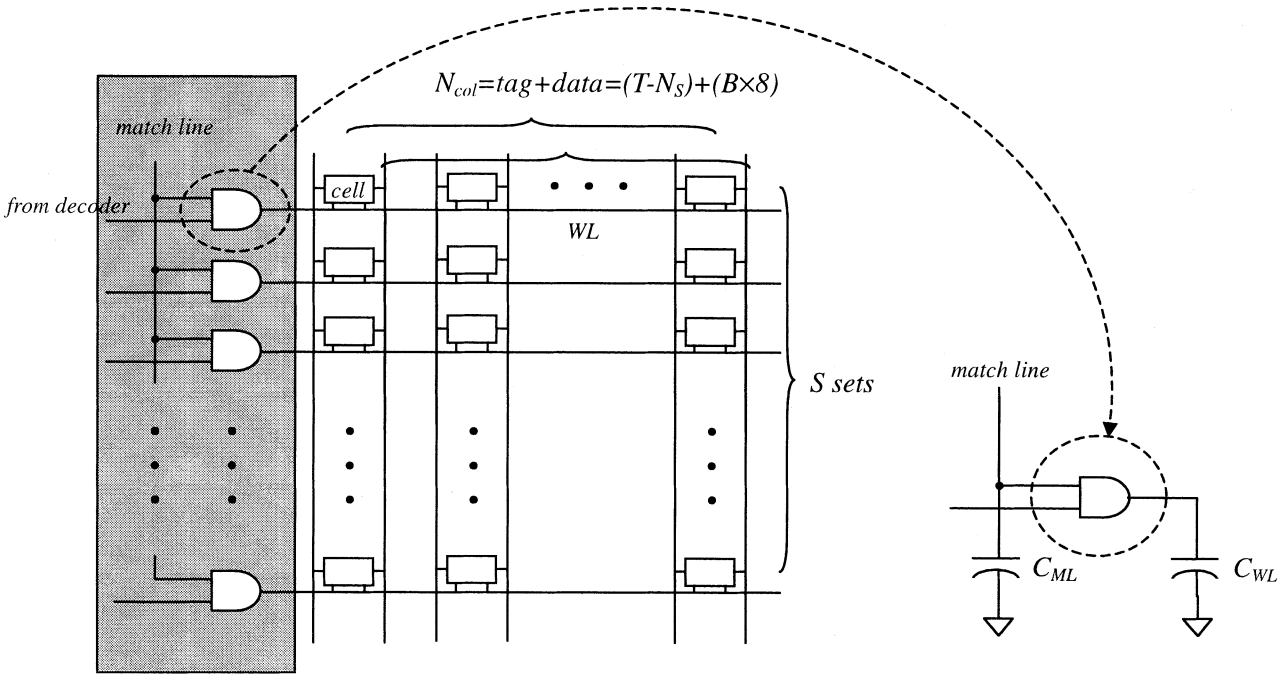


Fig. 9. Control circuit in sentry-tag architecture.

TABLE II
POWER CONSUMPTION OF SENTRY TAG WITH CONTROL CIRCUIT

N_{ST}	1	2	3	4	5	6	7	8
P_{ST_1} (mW)	0.8347	0.8378	0.8444	0.8498	0.8552	0.8606	0.8659	0.8716

power consumption of the sentry tag (P_{ST}) in a A-way cache is given by $P_{\mathrm{ST}} = P_{\mathrm{ST\_1}} \times A$.

V. EXPERIMENTAL RESULTS

In this paper, we use *SimpleScalar* [16] to simulate the *SPEC2000* benchmarks. To get a good mix of CPU- and memory-intensive loads, we randomly chose eight *CINT2000* and four *CFP2000* benchmarks. Table III summarizes the benchmarks, provides a brief description of them, and indicates the number of instructions and data simulated for each workload.

A. Baseline Cache Configurations

In this paper, we use the on-chip cache architecture with split instruction and DCs, which are a 32-kB two-way IC and a 32-kB four-way DC, respectively. The block size for both caches is 32 B. To avoid an explosion in the number of results, the address space is fixed to be 32-b wide.

B. Results and Discussions

In the following discussions, we use filter rate, average way activities, power savings, and access delay as the criteria to compare the baseline cache implemented in a conventional architecture to that implemented with the two-level filter scheme. For fair comparison, we also compare our architecture to that implemented with only the L1 filter. Since the simulation result difference between *CINT2000* and *CFP2000* is hardly noticeable, we do not present these two benchmarks separately in this paper.

Filter Rate of L1: We first define the filter rate of the L1 filter $(\mathrm{L1\_FR})$ as the ratio of the number of block buffer hits to the number of cache accesses. The higher value of $\mathrm{L1\_FR}$ means that the L1 filter is more efficient in filtering out the unnecessary cache accesses.

Fig. 10 depicts how $\mathrm{L1\_FR}$ is achieved with the addition of a single block buffer. Clearly, the value of $\mathrm{L1\_FR}$ is fixed for various cache configurations. The key observation is that, with the use of a single block buffer, we can eliminate roughly 69% and 37% of cache access for IC and DC, respectively. Due to poor locality, the single block buffer used in the L1 filter is less beneficial to DC than to IC.

Filter Rate of L2: The filter rate of the L2 filter $(\mathrm{L2\_FR})$ is the ratio of the number of unnecessary way activities to the total number of way activities in case of the L1 filter miss. The higher value of $\mathrm{L2\_FR}$ means that the L2 filter is more efficient in filtering out the unnecessary way activities. Fig. 11 shows the $\mathrm{L2\_FR}$ of both IC and DC after the L1 filtering. In this simulation, we considered two different configurations of a sentry tag for further investigation. One is a 1-b sentry tag, in which we use the least significant bit of tag portion (e.g., A[11] in Fig. 3) as a sentry bit, and the other is a 2-b sentry tag, in which we use the least two significant bits of tag portion (e.g., A[12:11] in Fig. 3) as sentry bits.

From Fig. 11, we summarize the most important aspects. First, in all cases depicted in this figure, $\mathrm{L2\_FR}$ of both IC and DC go up with the cache associativity. This trend can also be observed in the one-way case (i.e., direct-mapped cache), but it is much less pronounced. This is because the more cache ways that must be activated in one cache access, the greater the possibility that we can filter out the unnecessary way activities.

TABLE III
BENCHMARK DESCRIPTIONS

Category	Benchmark	Description	Instr. Count	Data Count
CINT2000	164.gzip	Compression	81641529094	26630126869
	175.vpr	FPGA Circuit Placement and Routing	214237074291	100616950911
	176.gcc	Programming Language Compiler	78423865621	31384113066
	181.mcf	Combinatorial Optimization	132862206988	67879944204
	197.parser	Word Processing	623496981528	333797230413
	253.perlbmk	PERL Programming Language	43730351658	20095105101
	255.vortex	Object-oriented Database	168619585094	88466806040
	256.bzip2	Compression	159926907421	80349377343
CFP2000	177.mesa	3-D Graphics Library	492137176762	246401727733
	179.art	Image Recognition/Neural Networks	181402289419	78280973858
	183.equake	Seismic Wave Propagation Simulation	597511951360	235108186746
	188.ammp	Computational Chemistry	1924889017223	542422636930

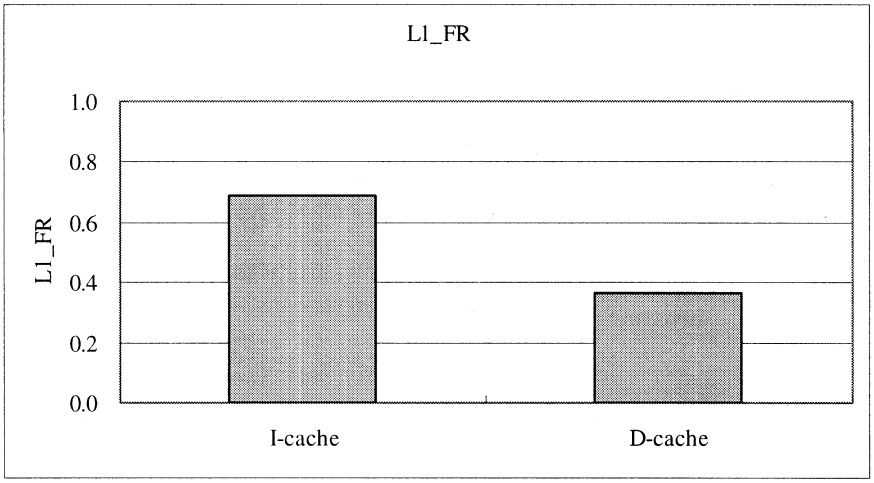
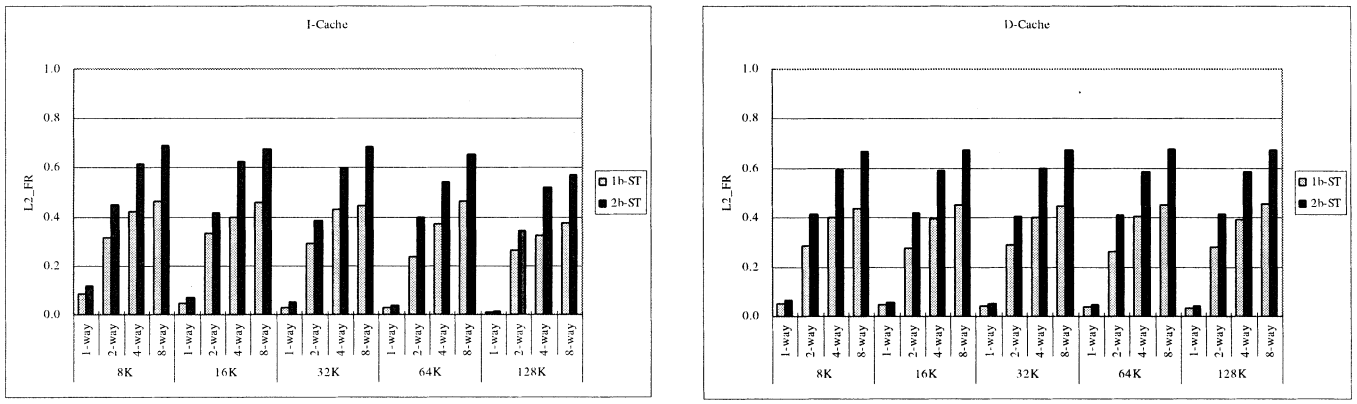


Fig. 10. L1_FR for IC and DC.

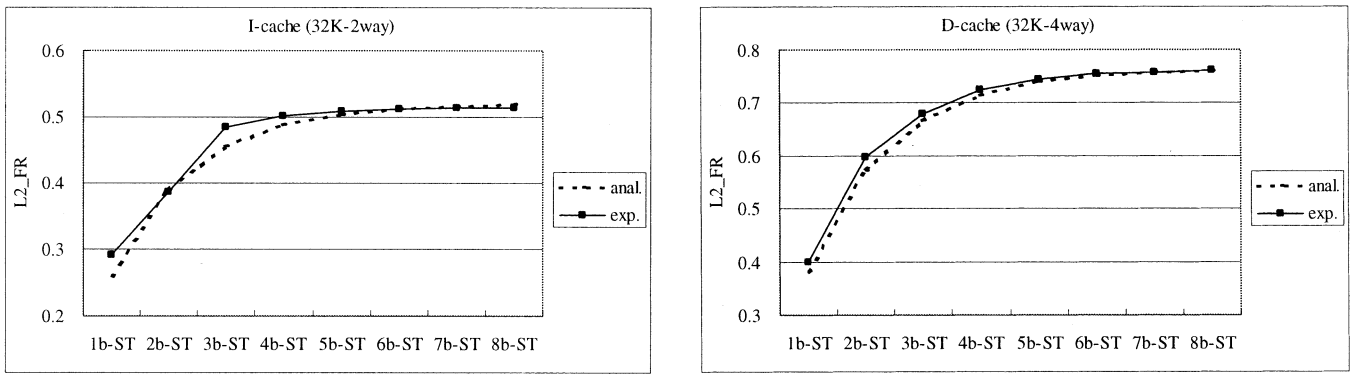


(a) (b)

Fig. 11. L2_FR of IC and DC with 1- and 2-b sentry tags. (a) IC. (b) DC.

Thus, the results suggest that the L2 filter is worthy of being implemented in the caches with associativity larger than one, especially for high associativity caches that are usually used in embedded processors, e.g., 32- and 64-way associative caches have been widely accepted [17], [18]. Second, the use of a 2-b sentry tag would lead to a higher L2_FR than the use of a 1-b sentry tag. Except for one-way caches, the former case would result in the improvement of 10%–20%+ in L2_FR for the latter case. This is a direct consequence of the inclusion property between 1-b and 2-b sentry tags, in which a 2-b sentry-tag match implies a 1-b sentry-tag match because of the sentry bits choice method described previously.

To further investigate the impact of increasing the number of sentry bits on L2_FR, the baseline caches were implemented



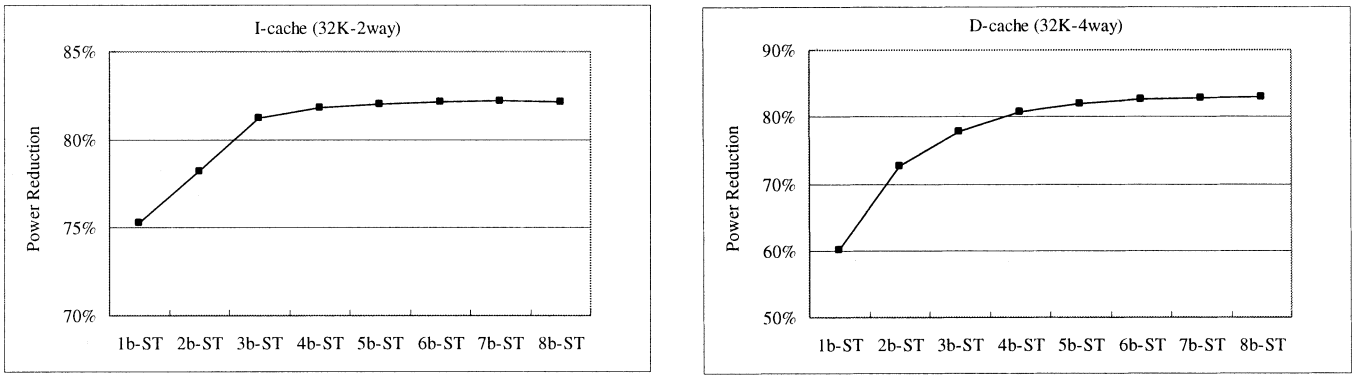
(a) (b)

Fig. 12. L2_FR for various number of sentry bits. The solid line is the experimental result and the dashed line is the analytic value obtained from (2). (a) IC. (b) DC.

TABLE IV
AVERAGE WAY ACTIVITIES OF THE BASELINE CACHE IMPLEMENTED WITH THE L1 FILTER AND TWO-LEVEL FILTER SCHEME

I-cache	L1_FR	L2_FR	W_{1LF}	W_{2LF}
1-way	0.688	0.060	0.312	0.293
2-way		0.484	0.624	0.322
4-way		0.670	1.249	0.412
8-way		0.784	2.497	0.540

D-cache	L1_FR	L2_FR	W_{1LF}	W_{2LF}
1-way	0.366	0.053	0.634	0.600
2-way		0.473	1.267	0.668
4-way		0.678	2.535	0.816
8-way		0.785	5.070	1.088



(a) (b)

Fig. 13. Power reduction for various numbers of sentry bits by (7). (a) IC. (b) DC.

TABLE V
AVERAGE PARTIAL POWER CONSUMPTION PER ACCESS FOR THE BASELINE CACHE AND THAT IMPLEMENTED WITH THE L1 FILTER AND TWO-LEVEL FILTER SCHEMES

I-cache	P_{conv}	P_{1LF}	P_{2LF}
1-way	242.2	83.9	81.1
2-way	364.4	122.1	68.8
4-way	597.0	194.7	71.6
8-way	1085.0	347.0	83.2

D-cache	P_{conv}	P_{1LF}	P_{2LF}
1-way	233.9	156.6	152.2
2-way	353.3	232.2	129.7
4-way	578.8	375.2	129.8
8-way	1055.5	677.2	155.3

TABLE VI
SUMMARY OF POWER-CONSUMPTION IMPROVEMENT FOR THE USE OF TWO-LEVEL FILTER SCHEME. THE BASELINE IC AND DC ARE 32K TWO-WAY AND 32K FOUR-WAY. Conv. IS THE CONVENTIONAL CACHE, AND Conv. + L1 IS THE CONVENTIONAL CACHE WITH L1 FILTER. (a) PARTIAL IMPROVEMENT. (b) TOTAL IMPROVEMENT

Partial Improvement	**I-cache**	**D-cache**
Compared to Conv.	81.13%	77.58%
Compared to Conv.+L1	43.68%	65.41%

(a)

Total Improvement	**I-cache**	**D-cache**
Compared to Conv.	56.79%	54.31%
Compared to Conv.+L1	30.57%	45.79%

(b)

with our two-level filter scheme, and the number of sentry bits used in the L2 filter is varied from 1 to 8. Fig. 12 shows the simulation results. We can observe the experimental results (solid line) and the analytic value (dashed line) obtained from (2) are almost the same. That validates the accuracy of the analytic model presented in Section III.

From Fig. 12, L2_FR increases with the number of sentry bits, although nonlinearly. Specifically, the case with a 3-b sentry tag is the knee of curve for both the IC and DC. In other words, when we use more than three bits as sentry bits, L2_FR continued to increase, but the increment is negligible. The key

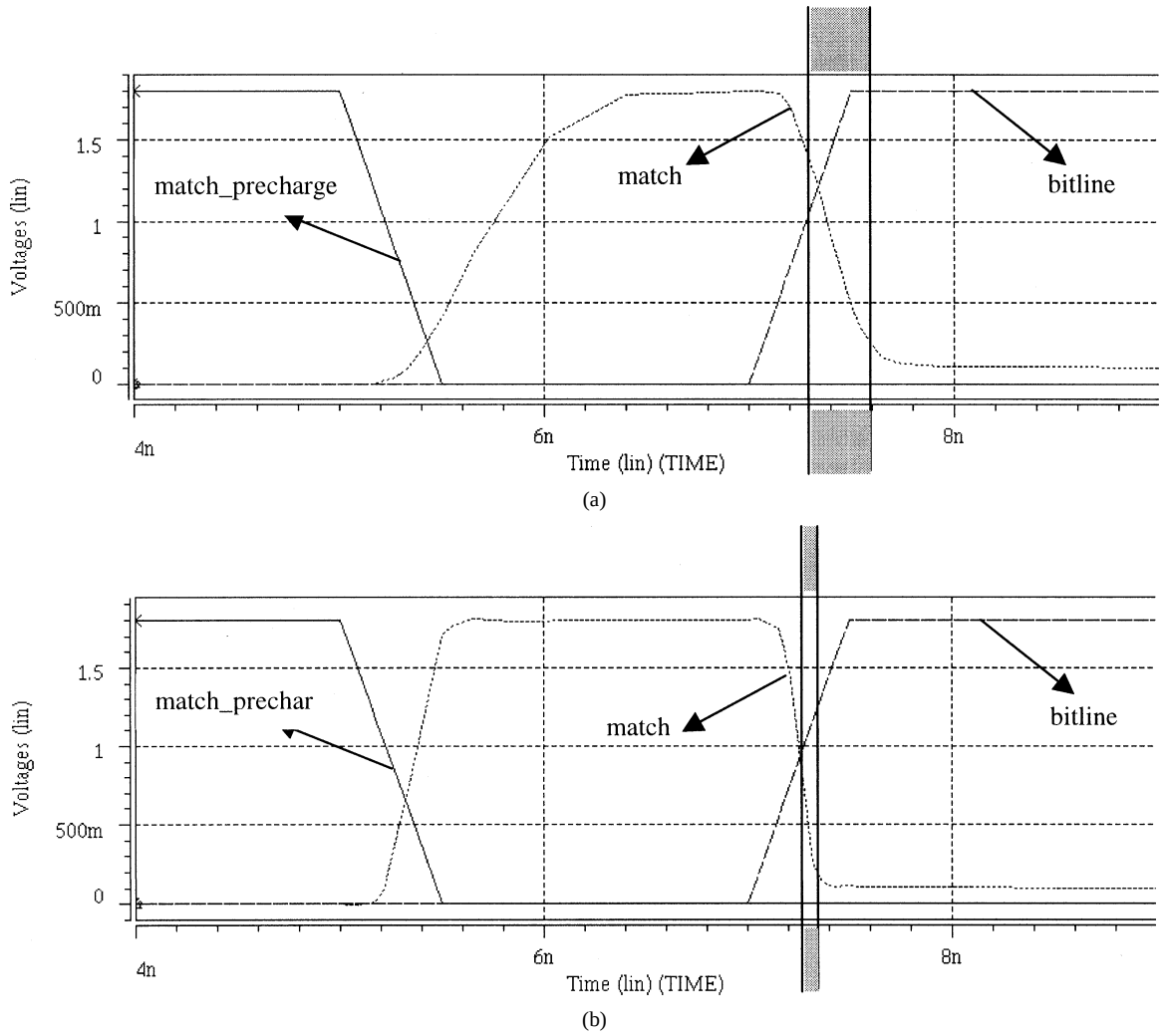


Fig. 14. *HSPICE* waveforms of: (a) L1 filtering and (b) 3-b sentry-tag comparison.

observation is that with the use of a small number of sentry bits, a large rate in filtering out the unnecessary way activities is easily achieved.

Average Way Activities: We then define the average way activities as the number of accessible ways in each cache access. Clearly, the average way activities of a conventional cache (W_{Conv}) is equal to the cache associativity, e.g., the average way activities of a conventional four-way cache is four. For a cache with only the L1 filter, the average way activities is given by (5). Similarly, for a cache with our proposed two-level filter, the average way activities is given by (6) as follows:

$$W_{1\mathrm{LF}} = \text{cache associativity} \times (1 - \mathrm{L1\_FR}) \tag{5}$$

$$W_{2\mathrm{LF}} = \text{cache associativity} \times (1 - \mathrm{L1\_FR}) \times (1 - \mathrm{L2\_FR}). \tag{6}$$

Apply the filter rate obtained from simulation to (5) and (6), the results of $W_{1\mathrm{LF}}$ and $W_{2\mathrm{LF}}$ are shown in Table IV. We observe that the use of a two-level filter is more efficient in reducing the number of average way activities than the L1 filter, especially for the DC with poor locality. For example, in a 32-kB eight-way DCs, the use of the two-level filter scheme can reduce the average way activities from 8 to 1.088, but the L1 filter only reduces the average way activities from 8 to 5.07.

Power Savings: Based on the power-consumption model described in Section IV, the average power consumption per access for the conventional cache (7) and that implemented with the L1 filter (8) and our proposed scheme (9) can be expressed by the following equations:

$$P_{\mathrm{Conv}} = W_{\mathrm{Conv}} \times P_{\mathrm{WAY\_Conv}} \tag{7}$$

$$P_{1\mathrm{LF}} = P_{\mathrm{L1}} + (W_{1\mathrm{LF}} \times P_{\mathrm{WAY\_Conv}}) \tag{8}$$

$$P_{2\mathrm{LF}} = P_{\mathrm{L1}} + \big((W_{2\mathrm{LF}} \times P_{\mathrm{WAY\_}2L}) + P_{\mathrm{ST}}\big). \tag{9}$$

$P_{\mathrm{WAY\_Conv}}$ and $P_{\mathrm{WAY}_{2L}}$ are the power consumption of one cache way for a conventional cache and our proposed architecture, respectively, and P_{ST} is the total power consumption of the sentry tag. They were described in Sections IV. P_{L1} is the power consumption of the block buffer in the L1 filtering, and the value obtained from the *HSPICE* simulation is approximately 8.35 mW. By using (9), the power-reduction curve for various numbers of sentry bits is shown in Fig. 13, which is similar to the curve of the filter rate shown in Fig. 12. This is because the sentry tag is a small storage, compared to the power consumption of one cache way $(P_{\mathrm{WAY}_{2L}})$, the value of P_{ST} is

insignificant. From Table II, even we use an 8-b sentry tag, the P_{ST} of IC and DC are 1.7 and 3.5 mW, respectively. Thus, (9) can be reduced to $P_{\mathrm{2LF}} \approx P_{\mathrm{L1}} + (W_{\mathrm{2LF}} \times P_{\mathrm{WAY\_2L}})$. Consequently, the power-reduction curve and the filter rate curve are almost the same (i.e., the knees of these two curves are the same). From the results shown in Figs. 12 and 13, we decided to use a 3-b sentry tag to implement the L2 filter for the baseline cache.

Combine (7)–(9) and the results illustrated in Table IV, the average power consumption per cache access measured in milliwatts for the baseline cache, which are implemented with the L1 filter, and the two-level filter shown in Table V. Note that the results shown in Table V are the partial cache power consumption, not the total cache power consumption, because we only consider the power consumption of the bitline and sense amplifier in our simulation (the reason for this has already been stated in Section IV).

We observe that the two-level filter scheme is very effective in filtering out the unnecessary way activities, and then large power savings can be achieved. The use of the L1 filter can reduce the power consumption, but it does not result in the kind of power savings that are realized with the use of our proposed two-level filter scheme, especially for the DC with poor locality. Consequently, the L2 filter used in the two-level filter scheme is effective in reducing the unnecessary power consumption in case of an L1 filter miss. Table VI summarizes the power-consumption improvement for the use of a two-level filter scheme. Table VI(a) shows the partial improvement of the cache power. To obtain the power consumption improvement of the entire cache, the results shown in Table VI(a) must be multiplied by 70% (underestimation). This is because the considered partial components (i.e., bitline and sense amplifier) contribute over 70% to the total cache power consumption [12]. Thus, Table VI(b) shows the power-consumption improvement of the entire cache, i.e., total improvement.

Delay Penalty: Up to this point, we have investigated the impact of power consumption of the proposed scheme. Another important factor is the cache access time. In our scheme, there must be a delay penalty due to the use of the two-level filter scheme before the normal cache access. Although the L1 filtering is followed by the L2 filtering, to minimize the delay penalty in filtering out the unnecessary cache accesses and way activities, we can overlap the L1 filtering with the L2 filtering, as described in Section III. We use the L1 filter to check whether the required data is still resident in the block buffer. At the same time, the set decoding and sentry bits comparison in the L2 filter are also completed in order.

To measure the L1 filtering time, we perform the *HSPICE* timing simulation. From the waveform shown in Fig.14(a), the time to determine an L1 filter hit is approximately 0.3 ns. As to the L2 filter time, since it consists of set decoding time and sentry bits comparison time, by using the tool *CACTI*, described in [13], we first estimate the time of set decoding are approximately 0.35 and 0.31 ns for base IC and DC, respectively, and then add the sentry bits comparison time [it is approximately 0.1 ns, as shown in Fig. 14(b)]. Consequently, the L1 filter time can be completely hidden by the L2 filtering. Since the set decoding time in the L2 filter is necessary for both the conventional cache and our scheme, the actual delay penalty due to the use of the two-level filter scheme is only the sentry bits' comparison time, which is approximately 0.1 ns in our base case.

VI. Conclusions

In modern processor design, the on-chip caches are used to boost the system performance. However, the on-chip caches usually consume a significant amount of power in processors. In this paper, we have focused on the architecture level to develop a technique for saving cache access power. The problems of the conventional set-associative cache were first identified. We then proposed a *two-level filter* scheme to reduce the unnecessary cache activities and, thus, save cache power. The proposed scheme is software independent and requires little hardware overhead, as well as slight architecture modification. In the L1 filter, we used a single block buffer to eliminate the unnecessary cache accesses and, in the L2 filter, we proposed a new *sentry-tag* architecture to further filter out the unnecessary way activities in case of the L1 filter miss. By using the result of the L2 filter to access only those possible hit ways, instead of accessing all the cache ways, the cache power consumption can be further reduced. The proposed scheme trades performance for power consumption, i.e., compared to the conventional cache, our method would result in the increase of cache access time by 0.1 ns. Experimental results show that the cache implemented with our proposed two-level filter would consume far less power than the conventional cache. Since the two-level filter scheme is based on the L1 filter architecture, for fair comparison, we compare it to both the conventional cache and that implemented with only the L1 filter. For the baseline IC (32 kB, two-way), compared to the conventional architecture implemented with the L1 filter, the use of the two-level filter can result in roughly 30% reduction in total cache power consumption. Similarly, for the baseline DC (32 kB, four-way), the total cache power reduction is approximately 46%. The maximum power saving depends on the program behavior and cache configuration, which suggests that the proposed two-level filter scheme is preferable to the DC with the poor locality, and it is worthy of being implemented in the caches with associativity larger than one, especially for high-associativity caches that are usually used in embedded processors.

References

[1] J. F. Edmondson *et al.*, "Internal organization of the Alpha 21164, a 300-MHz 64-bit quad-issue CMOS RISC microprocessor," *Digital Tech. J.*, vol. 7, no. 1, pp. 119–135, 1995.

[2] J. Montanaro *et al.*, "A 160 MHz, 32 b 0.5 W CMOS RISC microprocessor," in *IEEE Int. Solid-States Circuits Conf. Dig.*, 1996, pp. 214–215.

[3] A. Hasegawa, I. Kawasaki, K. Yamada, S. Yoshioka, S. Kawasaki, and P. Biswas, "SH3: High code density, low power," *IEEE Micro*, vol. 15, pp. 11–19, Dec. 1995.

[4] H. Choi, M. K. Yim, J. Y. Lee, B. W. Yun, and Y. T. Lee, "Low-power four-way associative cache for embedded SOC design," in *Proc. 13th Annu. IEEE Int. ASIC/SOC Conf.*, 2000, pp. 231–235.

[5] J. Kin, M. Gupta, and W. H. Mangione-Smith, "The filter cache: an energy efficient memory structure," in *Proc. 30th Int. Microarchitecture Symp.*, Dec. 1997, pp. 184–193.

[6] N. Bellas, I. N. Hajj, C. D. Polychronopoulos, and G. Stamoulis, "Architectural and compiler techniques for energy reduction in high-performance microprocessors," *IEEE Trans. VLSI Syst.*, vol. 8, pp. 317–326, June 2000.

[7] C. L. Su and A. M. Despain, "Cache design for energy efficiency," in *Proc. 28th Int. System Sciences Conf.*, 1995, pp. 306–315.
[8] K. Ghose and M. B. Kamble, "Reducing power in superscalar processor caches using subbanking, multiple line buffers and bit-line segmentation," in *Proc. Int. Low Power Electronics and Design Symp.*, 1999, pp. 70–75.
[9] D. H. Albonesi, "Selective cache ways: on-demand cache resource allocation," in *Proc. 32nd Int. Microarchitecture Symp.*, 1999, pp. 248–259.
[10] K. Inoue, T. Ishihara, and K. Murakami, "Way-predicting set-associative cache for high performance and low energy consumption," in *Proc. Int. Low Power Electronics and Design Symp.*, 1999, pp. 273–275.
[11] Y. J. Chang, F.Feipei Lai, and S. J. Ruan, "An efficient two-level filter scheme for low power cache," presented at the *IEEE/ACM 11th Int. Logic and Synthesis Workshop*, New Orleans, LA, June 4–7, 2002.
[12] G. Reinman and N. Jouppi, "An integrated cache timing and power model," Compaq, Palo Alto, CA, WRL Summer Internship, 1999.
[13] S. E. Wilton and N. Jouppi, "An enhanced access and cycle time model for on-chip caches," DEC, Palo Alto, CA, WRL Res. Rep. 93/5, July 1994.
[14] P. Shivakumar and N. Jouppi, "CACTI 3.0: An integrated cache timing, power, and area model," Compaq, Palo Alto, CA, WRL Res. Rep. 2001/2.
[15] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 2nd ed. San Mateo, CA: Morgan Kaufmann, 1995.
[16] D. C. Burger and T. M. Austin, "The SimpleScalar tool set, version 2.0," *Comput. Architecture News*, vol. 25, no. 3, pp. 13–25, June 1997.
[17] S. Santhanam *et al.*, "A low-cost 300-MHz RISC CPU with attached media processor," *IEEE J. Solid-State Circuits*, vol. 33, pp. 1829–1839, Nov. 1998.
[18] *ARM920T Technical Reference Manual*, ARM Ltd., Cambridge, U.K., 1999.

Yen-Jen Chang (M'02) received the B.S. degree in information engineering from Feng Chia University, Taiwan, R.O.C., in 1996, the M.S. degree in information and computer engineering from Chung Yuan Christian University, Taiwan, R.O.C., in 1997, and is currently working toward the Ph.D. degree in computer science and information engineering at the National Taiwan University, Taipei, Taiwan, R.O.C.

His current research interests are computer architecture systems, microprocessor architectures, low-power storage, and very large scale integration (VLSI) SOC design.

Shanq-Jang Ruan (M'00) received the B.S. degree in computer science and information engineering from Tamkang University, Taiwan, R.O.C., in 1995, and the M.S. degree in computer science and information engineering and Ph.D. degree in electrical engineering from the National Taiwan University, Taipei, Taiwan, R.O.C., in 1997 and 2002, respectively.

From July 1997 to May 1999, he was an Electronic Officer with the R.O.C. Air Force. From September 2001 to May 2002, he was a Software Engineer with the Avant! Corporation. Since June 2002, he has been a Software Engineer with Synopsys Inc., Taipei, Taiwan, R.O.C. His research interests are all aspects of low-power synthesis and *RC* extraction of VLSI physical design automation.

Feipei Lai (S'84–M'87–SM'94) received the B.S.E.E. degree from the National Taiwan University, Taipei, Taiwan, R.O.C., in 1980, and the M.S. and Ph.D. degrees in computer science from the University of Illinois at Urbana-Champaign, in 1984 and 1987, respectively.

He is currently a Professor with the Computer Science and Information Engineering Department and the Electrical Engineering Department, National Taiwan University. He is the Director of the Computer and Information Network Center, National Taiwan University. He was a Visiting Professor with the Department of Computer Science and Engineering, University of Minnesota, Minneapolis. He was also a Guest Professor with the University of Dortmund, Dortmund, Germany, and a Visiting Senior Computer System Engineer with the Center for Supercomputing Research and Development, University of Illinois at Urbana-Champaign. In 1988, he served as a consultant with ERSO, ITRI, and from August 1994 to July 1995, with the Faraday Technology Corporation. He is one of the founders of the Institute of Information and Computing Machinery. He holds four R.O.C. patents and two U.S. patents. His current research interests are SOC low-power computing, computer architecture systems, and VLSI SOC design. He is in *Who's Who in Science and Engineering* and *Who's Who in the World*.

Prof. Lai is a member of Phi Kappa Phi, Phi Tau Phi, the Association for Computing Machinery (ACM), and the Chinese Institute of Engineers. He was the five-time recipient of the 1989, 1991–1993, and 1995 Acer Award. He was also the recipient of the 1991 Taiwan Fuji Xerox Research Award.