Low-power L2 cache design for multi-core processors

C.-M. Chung and J. Kim

A low-power set-associative L2 cache design for a multi-core processor is proposed. Since this way-predicting L2 cache (WP-L2) predicts a destination way and accesses only the predicted way, it consumes less energy than a conventional set-associative L2 cache. Exploiting access patterns of an L2 cache, WP-L2 is based on two prediction logics; a look-ahead buffer (LAB) predicts the next sequential cache block and a way-affinity table (WAT) records the way number of the previous L2 cache access. Combining the logics, WP-L2 predicts correct ways for about 83% of L2 cache accesses and reduces about 22% of access latency and 44% of energy consumption compared to the conventional eight-way set-associative L2 cache.

Introduction: According to the development of architecture and VLSI technology, multiprocessor system-on-a-chips (MPSoCs) are now widely used in high-performance mobile embedded systems. Since the general power source of mobile embedded systems is a battery, low-power consumption is increasingly important in the design of MPSoCs.

For high capacity and data sharing, modern MPSoCs prefer large L2 caches. To achieve a low miss ratio, they employ set-associative caches. However, set-associative caches are energy inefficient because the requested data has only one way and other ways are unnecessary to be accessed. To enhance the energy efficiency of set-associative caches, way-predicting L1 caches have been proposed [1]. They remember the MRU way number for each set and the way is accessed first in the next L1 cache access. The MRU heuristic works fine for L1 caches because the L1 caches have a high degree of locality. However, in L2 caches, most of accesses with high locality are filtered by L1 caches, thus making the MRU heuristic less efficient.

In this Letter we propose a way-predicting L2 cache (WP-L2) for MPSoC. Our proposed techniques are based on two observations of L2 cache access patterns. First, many L2 cache accesses are sequential. We call this 'the sequentiality property'. The sequential accesses can be found in fetching instructions and loading/storing data blocks. Block-level processing, which is common for multimedia applications or matrix operations, requires multiple sequential L2 cache accesses when the data block size is larger than the L2 cache block size. The cache blocks in the same way of sequential sets tend to be accessed successively, because the cache blocks are mapped to the same data block. We call this 'the way-affinity property'. WP-L2 takes account of the properties in predicting the next way to be accessed.

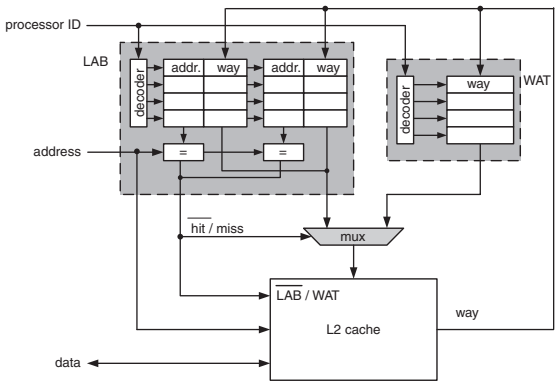

Fig. 1 *Logic diagram of way-predicting L2 cache (with 2-way LAB and WAT) for 4-way MPSoC*

A way-predicting L2 cache (WP-L2): Fig. 1 shows a logic diagram of our WP-L2. The way prediction logics of WP-L2 consist of a look-ahead buffer (LAB), a way-affinity table (WAT), and a mux. LAB, which predicts the next way based on the sequentiality property, stores the way numbers of the next cache blocks. After an L2 cache access, the way number of the next sequential cache block is looked-ahead and stored in LAB, for use in the next L2 cache access. LAB is implemented as a small set-associative cache with a set size equal to the number of processors. Way size of LAB is configurable. The address of the cache block is used as the tag. If the address is matched, LAB returns a way number. On the other hand, WAT, which exploits the way-affinity property, stores the way numbers of the *last* L2 cache accesses. Since there is one last access for each processor, WAT is implemented using a register array with a size equal to the number of processors. Each register stores a way number.

If we estimate the area size of L2 cache by CACTI [4], in a case of ARM11 MPCore [3], the original L2 cache requires about 61.85 mm^2 and WP-L2 requires about 61.93 mm^2. So, WP-L2 requires only 0.14% of L2 cache area to increase.

When an L2 cache access is requested, a processor ID and an address are sent to LAB and WAT to predict a target way. LAB searches a way number using a requested address, and WAT reads the way number at the same time. The mux selects one out of ways from LAB and WAT. The way from LAB is preferred if LAB finds a hit because LAB has a lower miss penalty (we will describe the miss penalty later). After prediction, the L2 cache accesses only the predicted way. If L2 cache finds a cache hit, it feeds back the way number to WAT for updating the way. If L2 cache finds a cache miss, after performing a victim replacement operation, the replaced way number is sent to WAT. After L2 cache access, L2 tags are checked again to find the way number of the next sequential cache block. If the block is found, the address and the way number are stored in LAB.

In WP-L2, operations of way prediction, L2 cache access, and way information update consume energy to get data from L2 cache. Thus, the average energy consumption of WP-L2 is modelled as follows (symbols are defined in Table 1)

$$AE_{WP-L2} = E_{predict} + AE_{L2} + E_{update} + AE_{leakage}$$

where

$$\begin{aligned} E_{predict} &= E_{LAB} + E_{WAT}, \\ AE_{L2} &= r_{hit} \times E_{1-way} + r_{miss} \times (E_{1-way} + E_{n-way}), \\ E_{update} &= E_{tag} + E_{LAB} + E_{WAT}, \\ AE_{leakage} &= AL_{WP-L2} \times (P_{LAB} + P_{WAT} + P_{n-way}) \end{aligned}$$

Table 1: Definition of variables in energy model

Variable	Definition
E_{LAB}, E_{WAT}, $E_{1\text{-}way}$, $E_{n\text{-}way}$, E_{tag}	Dynamic energy values of LAB, WAT, 1-way L2 cache, n-way L2 cache, and L2 cache tag lookup
P_{LAB}, P_{WAT}, $P_{n\text{-}way}$	Leakage power values of LAB, WAT, and n-way L2 cache
$AL_{WP\text{-}L2}$	Average latencies of WP-L2
L_{LAB}, L_{WAT}, $L_{1\text{-}way}$, $L_{n\text{-}way}$	Latency of LAB, WAT, 1-way L2 cache, and n-way L2 cache
r_{hit}, r_{miss}	Way prediction hit and miss ratio

The latency of WP-L2 is variant according to prediction hit or miss [1]. If LAB or WAT finds a correct way, L2 accesses only the target way. Otherwise, L2 re-accesses all ways again. Thus, the average latency of n-way WP-L2 is modelled as follows:

$$\begin{aligned} AL_{WP-L2} = \text{MAX}(L_{LAB}, L_{WAT}) + L_{L2,1-way} \\ + L_{L2,n-way} \times r_{miss} \end{aligned}$$

However, if only LAB is used in WP-L2, all ways are accessed directly in the case of a prediction miss, and the average latency model is as follows:

$$AL_{WP-L2} = L_{LAB} + L_{L2,1-way} \times r_{hit} + L_{L2,n-way} \times r_{miss}$$

Experimental results: In order to evaluate energy efficiency of WP-L2, we executed parallel applications in SPLASH-2 and MiBench on CATS [2]. We modified the simulator to include WP-L2 and the energy model. We configured CATS according to the specification of ARM11 MPCore. The energy parameters were derived from CACTI and Wattch [5].

Table 2 shows prediction hit ratios of way-predicting caches. Since WP-L2 consists of LAB and WAT, WP-L2 is evaluated with three combinations. In the aspect of hit ratio, WP-L2 outperforms MRU heuristic in all the benchmark programs. WP-L2 utilising the both LAB and WAT showed the best performance, because it had the advantage of complementary cooperation of them. On average, WP-L2 predicted correct ways for about 83% of L2 cache accesses. Although the average hit ratio of WAT was a little higher than LAB's, LAB resulted in a higher hit ratio for half of the applications. It is because the prediction hit ratios of LAB and WAT are affected by the L2 cache access pattern of the target application.

Table 2: Prediction hit ratios of way-predicting caches

	MRU	WP-L2		
		LAB only	WAT only	LAB and WAT
cholesky	0.13	0.89	0.75	0.92
cjpeg	0.23	0.70	0.69	0.81
djpeg	0.33	0.60	0.81	0.87
fft	0.03	0.34	0.47	0.53
lu (cont.)	0.07	0.93	0.89	0.95
lu (non-cont.)	0.38	0.61	0.83	0.86
raytrace	0.41	0.44	0.50	0.61
tiff	0.07	0.94	0.87	0.97
volend	0.22	0.81	0.84	0.91
Average	0.21	0.69	0.74	0.83

Tables 3 and 4 show latency and energy reduction of way-predicting caches compared with the conventional L2 cache. Similar to the prediction hit ratio, WP-L2 reduced more latency and energy than the MRU heuristic. WP-L2 using only LAB reduced the most latency (about 22%), because LAB has small miss-prediction penalty (LAB access). However, the MRU heuristic increased latency because of low hit ratio and large miss-prediction penalty (MRU list access and 1-way L2 cache access). Although WP-L2 including LAB and WAT required more leakage and dynamic energy to manage the prediction logics, because its prediction hit ratio was the highest, it reduced the most energy in all applications. On average, WP-L2 reduced about 44% of the L2 cache energy consumption. However, the MRU heuristic consumed more energy than the conventional L2 cache. Since MRU prediction logic also consumed energy and its prediction hit ratio is low, it did not reduce energy consumption of the L2 cache.

Table 3: Rates of latency reduction of way-predicting caches

	MRU (%)	WP-L2		
		LAB only (%)	WAT only (%)	LAB and WAT (%)
cholesky	−49	32	12	29
cjpeg	−40	22	7	19
djpeg	−29	17	19	24
fft	−60	5	−16	−9
lu (cont.)	−56	34	27	32
lu (non-cont.)	−25	18	20	24
raytrace	−22	9	−13	−1
tiff	−56	34	24	35
volend	−40	28	21	29
Average	−42	22	11	20

Table 4: Rates of energy reduction of way-predicting caches

	MRU (%)	WP-L2		
		LAB only (%)	WAT only (%)	LAB and WAT (%)
cholesky	−7	58	49	58
cjpeg	1	40	40	45
djpeg	08	26	38	38
fft	−16	19	24	24
lu (cont.)	−11	54	55	55
lu (non-cont.)	16	40	59	56
raytrace	14	21	22	25
tiff	−13	61	60	63
Volend	1	33	36	37
Average	−1	39	43	44

Conclusions: This Letter describes a way-predicting L2 cache (WP-L2): a new low-power set-associative L2 cache design for MPSoC. We analysed access patterns of the L2 cache and utilised two major properties: the sequentiality property and the way-affinity property. By adding simple two-way prediction logics, the look-ahead buffer (LAB) and the way affinity-table (WAT), WP-L2 predicted correct ways for about 83% of L2 cache accesses and achieved about 22% of latency and 44% of energy reduction compared to a conventional set-associative L2 cache. Experimental results show that WP-L2 is more energy and performance efficient than previous way-predicting schemes in L2 caches.

Acknowledgments: This work was supported by KOSEF grant funds (ROA-2007-000-20116-0, R33-2008-000-10095-0) and the BK21 Project in 2010. The ICT at SNU and IDEC provided research facilities.

9 March 2010
doi: 10.1049/el.2010.0642

C.-M. Chung and J. Kim (*School of Computer Science & Engineering, Seoul National University, Seoul 151-742, Korea*)

E-mail: jihong@davinci.snu.ac.kr

References

1 Inoue, K., Ishihara, T., and Murakami, K.: 'Way-predicting set-associative cache for high performance and low energy consumption'. Proc. of ISLPED, San Diego, CA, USA, 1999, pp. 273–275
2 Kim, D., Ha, S., and Gupta, R.: 'CATS: cycle accurate transaction-driven simulation with multiple processor simulators'. Proc. of DATE, Nice Acropolis, France, 2007, pp. 749–754
3 ARM: 'MPCore multiprocessor technical reference manual', ARM DDI 0360A, 2005
4 Shivakumar, P., and Jouppi, N.P.: 'CACTI 3.0: an integrated cache timing, power, and area model', *WRL Res. Rep. 2001/2*, 2001
5 Brooks, D., Tiwari, V., and Martonosi, M.: 'Wattch: a framework for architectural-level power analysis and optimizations'. Proc. of ISCA, Vancouver, BC, Canada, 2000, pp. 83–94