6 WALTARI, M., and HALONEN, K.: 'Timing skew insensitive switching for double-sampled circuits'. Proc. IEEE Int. Symp. Circuits and Systems (ISCAS), May 1999, Vol. 2, pp. 61–64

7 SUMANEN, L., WALTARI, M., and HALONEN, K.: 'A 10-bit 200MS/s CMOS parallel pipeline A/D converter'. Proc. 26th European Solid-State Circuits Conf. (ESSCIRC), Sept. 2000, pp. 440–443

8 PARK, Y.-I., SOUNDARAPANDIAN, K., TSAY, F., and BARTOLOME, E.: 'A 1.8V, 10-bit, 100MS/s CMOS pipelined ADC'. 2001 IEEE Int. Solid-State Circuits Conf. (ISSCC), Feb. 2001

Instruction cache organisation for embedded low-power processors

Changwoo Jung and Jihong Kim

A low-power I-cache architecture is proposed that is appropriate for embedded low-power processors. Unlike existing schemes, the proposed organisation places an extra small cache in parallel alongside the L1 cache. Since it allows simultaneous accesses to both caches, the proposed scheme introduces little performance degradation. Using simple hardware logic (for sequential accesses) and a compiler transformation (for loop accesses), most L1 cache requests are served by a small cache, so that the amount of energy consumed by the L1 cache is significantly reduced. Experimental results show that for the SPEC95 benchmarks, the proposed organisation reduces the energy-delay product on average by 67.2% over a conventional cache design and 16.8% over the filter cache design.

Introduction: For high-performance embedded microprocessors, a large on-chip instruction cache is necessary to satisfy the need for fast instruction access. Although a large on-chip cache is effective in meeting this requirement, it dissipates a significant amount of energy. Since many high-performance embedded microprocessors are used for time-critical applications (where performance cannot be sacrificed for low energy consumption), if a low-power instruction cache design technique were to be useful, it must maintain the same performance level of the original processor while reducing the power consumption. However, existing low-power instruction cache design techniques, which are based on the vertical extension of the memory hierarchy (as with the filter cache [1] and the loop cache [2]) do not achieve this conflicting requirement in a satisfactory fashion.

In this Letter, we propose a novel low-power instruction cache organisation that reduces the energy consumption significantly with little or no impact on performance. The proposed organisation differs from existing schemes in that a small cache is located not between the CPU and L1 cache, but alongside the L1 cache. With an appropriate compiler support based on cache trace information, our scheme reduces the energy-delay product significantly over the filter cache and loop cache organisations as well as the conventional cache design.

Enhanced mini-cache (EM-cache) organisation: A popular design approach for a low-power on-chip cache is to introduce a small cache (in addition to a normal L1 cache) into a memory hierarchy. By placing a small cache between the CPU and L1 cache, a large portion of the processor's instruction requests is served by the small (thus energy-efficient) cache. Although it improves the overall energy efficiency of the memory hierarchy, the vertical extension of the memory hierarchy incurs additional delay cycles because the small cache is bound to miss some instruction requests.

In the proposed scheme, an extra small cache is located alongside the L1 cache, as shown in Fig. 1. We call this small cache the EM-cache. By putting the EM-cache alongside the L1 cache, there is no performance degradation, since both caches can be accessed simultaneously. To reduce the energy consumption, we selectively disable accesses to the L1 cache by the *L1_Deact* signal, which is set either by hardware logic (that sets *L1_Dynamic_Deact*) or by special instructions (that set *L1_Static_Deact*). The instruction selection logic in the ID stage chooses the appropriate instruction between the IR_{EM} and IR_{L1}. In the proposed scheme, since *L1_Deact* is set to 1 for most of the L1 cache requests, the instruction in IR_{EM} is generally used in the ID stage, leaving the L1 cache in the deactivated state most of the time. Since the instruction selection is made in the ID stage, there is no increase in the cache access time that may be on the critical path.

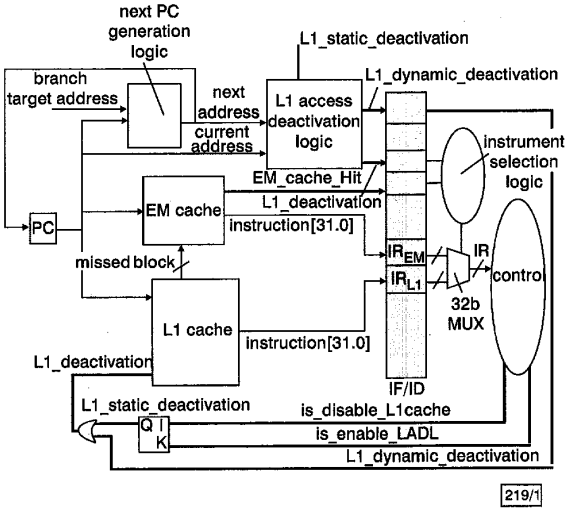

Fig. 1 *Memory hierarchy with EM-cache*

control line
data line

The energy efficiency of the memory hierarchy with the EM-cache depends on the quality of the L1 cache deactivation mechanism. The proposed L1 cache deactivation mechanism consists of two components, one in the software and the other in the hardware. The hardware deactivation component exploits the spatial locality of the sequential instruction accesses. Since many cache accesses are sequential accesses to the same cache block, we can disable the L1 cache if the next PC refers to the same cache block as the current PC does. The L1 access deactivation logic (LADL) module in Fig. 1 checks this condition for the next PC. If the next PC points to the same cache block as the current PC does, *L1_Dynamic_Deact* is set to 1, which disables the operation of the L1 cache for the next PC access. The *L1_Dynamic_Deact* signal can be set without extra time penalty while the current instruction is fetched, because the LADL module decides if the next PC will access the same cache block.

While the LADL-based hardware deactivation mechanism is simple and effective in detecting consecutive instruction fetches within the same cache block, it cannot exploit the spatial locality across multiple cache blocks. For example, many loops (common in computation-intensive programs) are likely to occupy multiple cache blocks. For such loops, the hardware deactivation mechanism cannot detect the consecutive accesses across the cache block boundary. To solve this problem, we propose an additional software deactivation mechanism based on two special instructions: *disable_L1Cache* and *enable_LADL*. The *disable_L1Cache* instruction deactivates the operation of the L1 cache while the *enable_LADL* instruction enables the LADL module as well as the L1 cache. Two special instructions are automatically inserted to appropriate basic blocks by a compiler. To insert the special instructions, the compiler classifies all the basic blocks into $D_{disable}$ or $S_{disable}$ based on cache trace information. A basic block *bb* is classified into $S_{disable}$ if the contribution from the *bb* basic block in the energy-delay product is smaller when *bb* belongs to $S_{disable}$ than when *bb* belongs to $D_{disable}$. Otherwise, a basic block is classified into $D_{disable}$.

If a basic block *bb* belongs to $D_{disable}$, the hardware deactivation component (i.e. the LADL module in Fig. 1) determines the operation status of the L1 cache. On the other hand, if a basic block *bb* belongs to $S_{disable}$, the L1 cache is deactivated while the instructions in *bb* are executed regardless of the *L1_Dynamic_Deact* signal value. For the basic blocks in $S_{disable}$, the *disable_L1Cache* instruction is inserted before the entry point of each basic block. For the successor basic block(s) bb_{succ} of a basic block $bb \in S_{disable}$, the *enable_LADL* instruction is inserted before the entry point of

bb_{succ} unless $bb_{succ} \in S_{disable}$. Fig. 2 illustrates how two special instructions are inserted into various basic block combinations.

While executing basic block $bb \in S_{disable}$, there can be some EM-cache misses with the L1 cache disabled. This is because the basic block bb may include cache blocks that were not brought into the EM-cache. Although it is small, the EM-cache misses introduce a performance penalty.

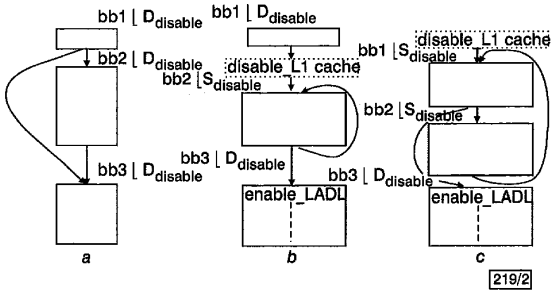

Fig. 2 *Special instruction insertions to basic blocks*

Table 1: Normalised energy-delay product relative to base case

Program	Filter cache	Loop cache	EM-cache
SPEC95 average	0.394	0.509	0.328
li	0.435	0.752	0.380
perl	0.609	0.899	0.430
hydro2d	0.363	0.422	0.303
su2cor	0.536	0.446	0.404
UTDSP average	0.375	0.374	0.293
histogram	0.545	0.497	0.401
lpc	0.501	0.425	0.327
spectral	0.163	0.170	0.166
edgedetect	0.445	0.548	0.338

Experimental results: To evaluate the effectiveness of the proposed cache organisation, we have performed experiments using the SimpleScalar microarchitecture simulator [3]. As test programs, we used the SPEC95 CPU benchmark and UTDSP benchmark. The cache energy models are based on the work described in [4]. Capacitive coefficients for the energy equations are derived from [5] assuming the 0.8μm process. For the experiments, a 32KB direct-mapped instruction cache with a block size of 32 bytes was used as the base case. For a small additional cache, a 0.5KB direct-mapped cache with a block size of 16 bytes was used.

Table 1 summarises the normalised energy-delay product of three cache organisations. As shown in Table 1, the EM-cache organisation achieves the best energy-delay product among the three cache organisations. For the SPEC95 CPU benchmark programs, the EM-cache reduces the energy-delay product on average by 67.2% over a conventional cache design, 16.8% over the filter cache and 35.6% over the loop cache. For the UTDSP benchmark programs, the EM-cache is equally effective in reducing the energy-delay product.

Conclusion: A low-power I-cache architecture, EM-cache, for high-performance embedded processors has been described. Unlike the existing schemes, the proposed organisation places an extra small cache in parallel alongside the L1 cache. Although the EM-cache organisation allows simultaneous accesses to the EM-cache and L1 cache, most L1 cache accesses are blocked by simple hardware logic (for sequential accesses) and a compiler transformation (for loop accesses), thus reducing the energy consumption significantly. Since the L1 cache is accessed (if necessary) at the same time when the EM-cache is accessed, there is no or little performance penalty. Experimental results show that the EM-cache achieves the best energy-delay product among the existing schemes.

Acknowledgment: This research is supported in part by the Ministry of Information & Communication of Korea ('Support Project of University Foundation Research('00)' supervised by IITA).

6 November 2000
Electronics Letters Online No: 20010380
DOI: 10.1049/el:20010380

Changwoo Jung and Jihong Kim (*School of Computer Science and Engineering, Seoul National University, San 56-1, Shilim-dong, Kwanak-ku, Seoul 151-742, Korea*)

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

References

1 KIN, J., GUPTA, M., and MANGIONE-SMITH, W.H.: 'The filter cache: an energy efficient memory structure'. Proc. Int. Symp. Microarchitecture, 1997, pp. 45–49
2 BELLAS, N., HAJJ, I., POLYCHROPOULOS, C., and STAMOULIS, G.: 'Architectural and compiler techniques for energy reduction in high-performance microprocessors', *IEEE Trans. VLSI Syst.*, 2000, **8**, (3), pp. 317–326
3 BURGER, D., and AUSTIN, T.M.: 'The SimpleScalar tool set, version2.0', *Comput. Archit. News*, 1997, pp. 13–25
4 KAMBLE, M.B., and GHOSE, K.: 'Analytical energy dissipation models for low power caches'. Proc. Int. Symp. Low-Power Electronics and Design, 1997, pp. 143–148
5 WILTON, S.E., and JOUPPI, N.: 'An enhanced access and cycle time model for on-chip caches'. Technical report, DEC WRL, 1994

Switching activity evaluation of CMOS digital circuits using logic timing simulation

J. Juan-Chico, M.J. Bellido, P. Ruiz-de-Clavijo, C. Baena, C.J. Jiménez and M. Valencia

The degradation delay model is applied to accurately estimate the switching activity in CMOS digital circuits. The model overcomes the limitations of conventional gate-level logic simulators to handle the propagation of glitches, a main source of switching activity. Model results of a four-bit multiplier are within 4% with respect to HSPICE, while Verilog overestimations are up to 68%.

Introduction: Accurate estimation of the switching activity (number of logic transitions) in CMOS digital circuits is crucial in calculating the power consumption of the devices [1, 2]. It has been stated that an important component of the switching activity is due to the propagation of glitches (spurious transitory signal pulses) [2]. The contribution of glitches to the overall switching activity ranges from 15 to 70%, depending on the circuit [2]. Accurate evaluation of this switching activity has only been possible by using electrical simulators like HSPICE [3]. However, these simulators are limited to rather small circuits and consume a large amount of computational resources. In contrast, gate-level logic simulators (like VERILOG [4] or VHDL standard simulators) are able to handle very big circuits, but are not accurate at simulating glitches, mainly because the delay models they use only consider the inertial effect when dealing with narrow pulses. This is a first-order approximation which yields large overestimations of the switching activity. To solve this problem, a simplistic dynamic delay model was proposed in [2], improving the accuracy, even if this was still far from the results obtained using electrical simulators.

In previous work, we have shown the importance of digital signal degradation [5], and a very accurate delay model known as the degradation delay model (DDM) has been developed, which handles the generation and propagation of glitches very accurately [6, 7]. In this Letter, we show that the DDM, when applied to logic simulation, provides a very accurate and fast way to estimate the switching activity of digital CMOS circuits, combining the benefits of electrical and logic simulators.

Degradation delay model: The degradation delay model considers a new dynamic parameter (T) in the calculation of the propagation delay of a logic block. This parameter measures the elapsed time since the last transition of the gate's output. It has been shown in [6] that the use of this parameter makes it possible to calculate the correct propagation delay even if almost simultaneous input transitions take place arbitrarily close in time. When this happens, a