IEICE TRANS. INF. & SYST., VOL.E82–D, NO.10 OCTOBER 1999

PAPER

Path-Classified Trace Cache for Improving Hit Ratio in Wide-Issue Processors

Jin-Hyuk YANG[†], **In-Cheol PARK**[†], *and* **Chong-Min KYUNG**[†], *Nonmembers*

SUMMARY In this paper, an instruction-cache scheme called Multi-Path Tracing is proposed to enhance the trace cache. Paths are classified to improve the trace cache hit ratio by reducing the path conflict and basic blocks are joined to reduce the hardware cost needed to implement the trace cache. Simulation results for various SPEC integer benchmarks show that the proposed scheme increases the hit ratio by more than 25% and the effective fetch size by 10%.

key words: *superscalar, instruction fetch, trace cache*

1. Introduction

To best utilize the high parallelism provided in the execution unit of a wide-issue superscalar processor [1], an instruction fetch mechanism that can supply sufficient instructions at a time is indispensable. However, conventional cache-based fetch mechanisms [2] are limited to supplying only one basic block per cycle as branch instructions and their targets are generally located at different cache lines. A basic block is a sequential instruction group that is bounded by branch instructions. Recently, the trace cache (T-cache) fetch mechanism [3]–[5], shown in Fig. 1, was proposed to overcome this limitation, which can fetch multiple noncontiguous basic blocks per cycle without increasing the fetch latency.

The T-cache stores dynamic instruction sequences and path information at run time, and is accessed in parallel with the instruction cache (I-cache) using the same fetch address. The I-cache is 2-way-interleaved so that two consecutive cache lines can be accessed at a time; this allows one-time fetching of sequential code that spans a cache line boundary [6], [7]. The multiple branch predictor [8] generates multiple branch predictions. If the T-cache is hit, meaning that fetch address matches the tag and branch predictions match the path information, an instruction sequence from the T-cache is fed to the decoder. On a T-cache miss, fetching proceeds normally to the I-cache. At the same time, the fill logic collects consequent basic blocks as they are fetched by the processor and merges them into a trace which will be written into the T-cache later. The merging process is completed when either a given number (n) of instructions have been merged into the trace or a given number (m) of branches have been detected in the trace. This T-cache mechanism supports one path for each starting address, which makes the trace stored in the T-cache useless if the directions of branches are changed dynamically at run time.

In this paper, we propose a new T-cache architecture called Multi-Path Tracing (MPT) to enhance the T-cache by supporting multiple paths for a starting address. The enhancement is associated with two hardware schemes, path classification and basic block joining. The previous T-cache fetch mechanism proposed by Rotenberg et al is referred to as Single Path Tracing (SPT) because it considers only one path for a starting address. Briefly speaking, the path classification reduces T-cache misses caused by path conflicts (Path conflict denotes that the branch prediction path does not agree with the path information of the trace stored in the T-cache.), and the basic block joining reduces the hardware cost necessary to implement the trace cache.

The rest of this paper is organized as follows. Section 2 presents the details of the two hardware schemes which are the architectural basis of the proposed MPT scheme. Section 3 describes the way to measure the performance of the MPT scheme. Section 4 presents the experimental results followed by concluding remarks.

2. The Architecture

Figure 2 shows the MPT mechanism equipped with the

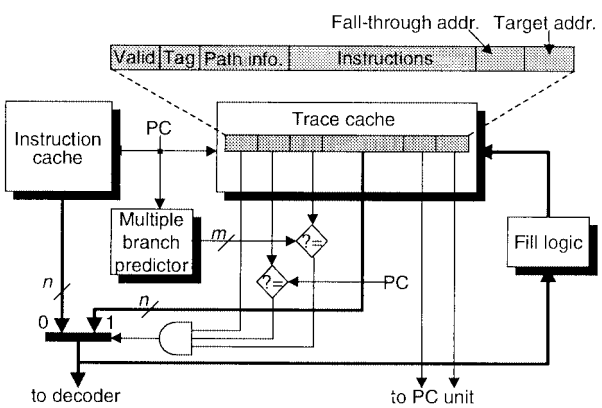


Fig. 1 Typical architecture of trace cache fetch mechanisms.

Manuscript received November 9, 1998.
Manuscript revised March 17, 1999.

[†]The authors are with the Department of Electrical Engineering, Korea Advanced Institute of Science and Technology, 373–1, Kusong-dong, Yusong-gu, Taejon, 305–701, Korea.

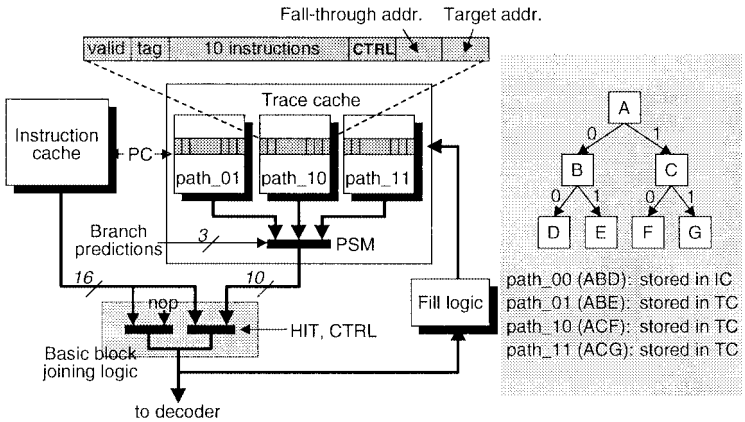


Fig. 2 Overall architecture of the proposed MPT fetch mechanism adopting the path-classified T-caches and basic block joining logic. (PSM in trace cache stands for path selection multiplexers.)

proposed hardware techniques. The major differences between the SPT [3] and the MPT are the organization of T-cache and the instruction selection logic. In the MPT, the T-cache is organized to support the path classification, and the instruction selection logic is designed to support the basic block joining.

Throughout this paper, as was assumed in the SPT scheme [3], we also assume that the fetch size is limited to 16 instructions ($n = 16$) and 3 basic blocks ($m = 3$).

2.1 Path Classification: Reducing Path Conflicts

In the SPT, T-cache can store only one path from a certain starting address. However, there are four possible paths that have the same starting address (See the control flow graph of Fig. 2.). Since the paths with the same starting address have to be placed at the same entry in the T-cache, many conflicts are inevitable. In general, the path conflicts exert a negative influence on the T-cache hit ratio. To investigate the effects of path conflicts on the T-cache hit ratio, we simulate the previous T-cache fetch mechanism for SPEC integer benchmarks. Figure 3 represents the percentage of path conflict misses with respect to the total misses and shows that a large portion of T-cache misses is caused by path conflicts. Furthermore, the portion increases as the size of T-cache increases; 25% of misses are caused by the path conflicts for 4 KB T-cache, 40% for 16 KB, and 52% for 64 KB. To reduce the path conflicts, we explicitly classify paths from a certain starting address, and store them in different locations.

As shown in Fig. 2, the MPT scheme contains three separate traces corresponding to three different paths of two consecutive branches, i.e., paths 01, 10, and 11. Since the basic blocks in path_00 can be directly supplied by the I-cache, it is not stored in the T-cache. The path_01 cache contains basic blocks B and E in the control flow graph of Fig. 2. To form the path_01 trace, the

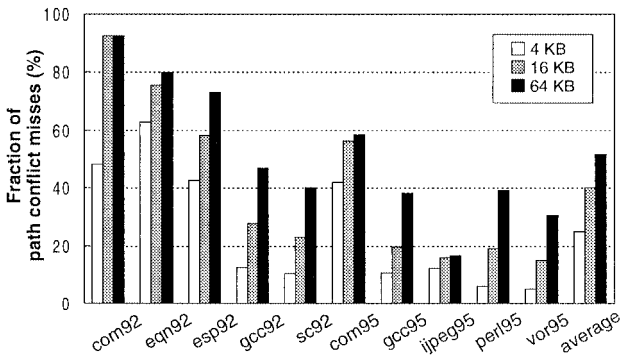


Fig. 3 Percentage of path conflict misses with respect to the total T-cache misses for various SPEC integer benchmark programs.

first basic block A supplied by the I-cache is joined with basic block B and E supplied from the path_01 cache by using basic block joining logic which will be described in Sect. 2.2. Likewise, the path_10 cache contains basic blocks C and F, and the path_11 cache contains basic blocks C and G. All paths are accessed in parallel with the current starting address, and a correct path is selected by using the branch prediction bits.

2.2 Basic Block Joining: Optimizing Hardware Costs

In the SPT, the instruction sequence that is supplied by the T-cache consists of 3 basic blocks from a starting address. If the T-cache is hit, instructions from I-cache are entirely discarded without any use. This is a redundancy of the T-cache fetch mechanism because the first basic block can be directly supplied by the I-cache.

Therefore, in our MPT scheme, only the second and the third basic blocks are stored in the T-cache, which are concatenated to the first basic block obtained from I-cache at run time. Hence, the T-cache needs to store only the second and third basic blocks, resulting

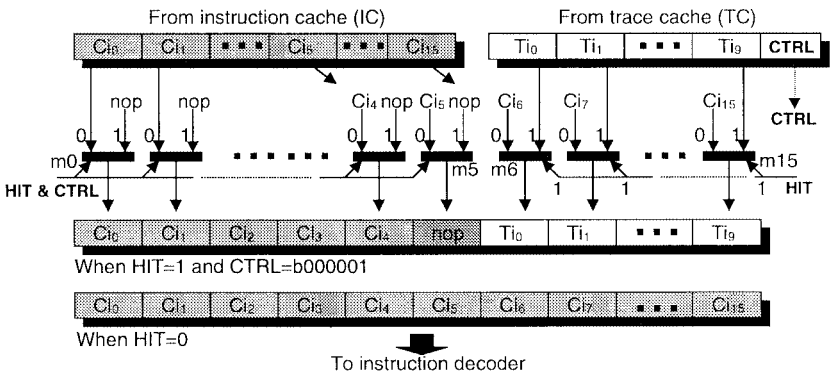


Fig. 4 Basic block joining in the MPT scheme, where a number of instructions from the I-cache as determined by the masking bits (HIT & CTRL) are joined with two basic blocks from the T-cache.

in the reduction of storage area by one third compared to the SPT scheme.

Figure 4 shows the basic operation of basic block joining. Instructions are selected from the I-cache and T-cache using multiplexers that are controlled by the HIT and CTRL signals. The HIT signal is activated, i.e., set to 1, if T-cache is hit. The CTRL signals represent the actual size of the first basic block and are provided by the T-cache. The size of the first basic block is limited to 6, as determined from simulation to achieve a good performance/cost trade-off. If the number of instructions in the first basic block is less than or equal to 6, the T-cache stores only the second and third basic blocks, and 'no-op' instructions are inserted between the first basic block from I-cache and the second basic block from T-cache at run time to make the selection logic simple. Otherwise, i.e., if the first basic block contains more than 6 instructions, the 7th and the following instructions in the first basic block are stored in the T-cache together with the second and the third basic blocks. The first basic block is selected by multiplexers, m0 to m5, which are controlled by the logical-AND of the HIT signal and the CTRL signals. The second and third basic blocks are selected by multiplexers, m6 to m15, which are controlled by the HIT signal.

2.3 Timing Comparison

The MPT scheme is intended to increase the T-cache hit ratio. If the MPT scheme increases the cycle time, the performance benefits obtained by the scheme will be diminished by the increased cycle time. Hence, it is important to guarantee that the proposed scheme does not increase the cycle time.

In terms of timing, there are two differences between the proposed MPT scheme and the SPT scheme. The first is that the MPT scheme, compared to the SPT scheme, has additional delay caused by the path selection multipelxer. However, since the number of entries of the I-cache is generally larger than that of the T-cache, the instruction fetch cycle is determined by the I-cache. This means the additional delay due to the path selection multiplexer does not increase the instruction fetch time. The second is that the behavior of basic block joining logic is more complex than the instruction selection logic of the SPT scheme. However, the delay in basic block joining logic is almost the same as the delay of the instruction selection logic of the SPT scheme, because they are composed of only one multiplxer.

3. Experiment

In this section, we present the experimental method used to investigate the effectiveness of the proposed MPT scheme. The simulation methodology and the parameters for the simulation models are described in Sect. 3.1, and the performance metrics are described in Sect. 3.2.

3.1 Simulation Methodology and Models

To evaluate and compare the performance, we constructed trace-driven simulation models for the MPT scheme and Rotenberg's SPT scheme [3] and simulated them with the T-cache size of 4 KB, 16 KB, and 64 KB using instruction traces of 10 SPEC integer benchmarks; five SPECint92 benchmarks and five SPECint95 benchmarks. The benchmarks were compiled using GNU C compiler with the compiler options "-O4-static." SPARC instruction traces were generated using the QPT (Quick Profiler and Tracer) [9].

Table 1 summarizes simulation parameters for T-cache fetch mechanisms used in this experiment, where MPT stands for Multi-Path Tracing and SPT stands for Single-Path Tracing, i.e., MPT is the enhanced T-cache fetch mechanism equipped with the proposed hardware techniques, while SPT is the Rotenberg's T-cache fetch mechanism [3] which can store only one path for a cer-

Table 1 Simulation parameters for SPT (Single-Path Tracing) and MPT (Multiple-Path Tracing) scheme. The parameters of the two TC fetch mechanisms were chosen such that the same total silicon area is used.

SIMULATION PARAMETERS			TC FETCH MECHANISM	
			SPT	MPT
Fetch limit			16 insts. or 3 basic blocks	
Instruction cache	size		8 KB – 128 KB	
	line size		64 byte	
	associativity		direct mapped	
	miss penalty		10 cycles	
Multiple branch predictor	length of BHR		14 bit	
	size of PHT		2^{14} 2-bit counters	
	#predictions per cycle		≤ 3	
Trace cache	line size		16 insts.	10 insts.
	#entries	4 KB	64	32/path
		16 KB	256	128/path
		64 KB	1024	512/path

tain starting address.

Since the T-cache size of MPT needed to store the three paths is roughly twice of the T-cache size of SPT for each entry because in the proposed MPT scheme only two basic blocks are stored instead of three, i.e., $(2/3) \times 3 = 2$, only half as many entries as SPT are used in MPT to fairly compare the two schemes, i.e., the performance of two schemes is compared with assuming the same total silicon area for the T-cache. For example, the number of entries of the MPT scheme for a 4 KB T-cache is 32, while that of SPT is 64.

To focus on the performance of the T-cache, we modeled the I-cache and the multiple branch predictor to have the same parameters for both schemes. The I-cache is direct mapped and its line size is 64 bytes. Its miss penalty is fixed to 10 cycles but its size is changed from 8 KB to 128 KB. The multiple branch predictor used in this experiment is an extended model of the GAg scheme [10] that has a 14-bit length of Branch History Register (BHR) and 2^{14} 2-bit counters in Pattern History Table (PHT).

3.2 Performance Metrics

To evaluate and compare the performance of T-cache fetch mechanisms, T-cache hit ratio and effective fetch size (EFS) are used as the performance metrics. EFS indicates the average number of correct instructions per cycle, denoting that the discarded instructions due to incorrect branch prediction are not counted in calculating EFS.

The behavior of the MPT scheme is some different from that of the SPT scheme in case that a miss occurs on the I-cache. On an I-cache miss, the MPT scheme has to stall during the miss service time, because the first basic block should be fetched from the I-cache entry, whereas the SPT scheme has no need to stall. To fairly compare the performance, the effect of instruction cache miss is considered in calculating the performance metrics. For the calculation of T-cache hit ratio, a T-cache hit means that hits are occurred on both T-cache and I-cache for the MPT scheme, whereas a hit on I-cache is not taken into account in the SPT scheme. The number of stalled cycles due to I-cache misses are also included in the calculation of EFS.

4. Experimental Results

At first, we simulated the SPT and the MPT schemes by varying the T-cache size from 4 KB to 64 KB, keeping the I-cache size constant at 128 KB. The first set of results, as shown in Table 2, represents the T-cache hit ratio for each simulated benchmark program. The proposed MPT scheme significantly improves the T-cache hit ratio. For 4 KB T-cache, the MPT scheme improves the hit ratio of T-cache by 28.8% on the average, varying from 10.1% improvement for com95 (compress in SPECint95) to 93% improvement for gcc95 (gcc in SPECint95). The average hit ratio of 16 KB and 64 KB T-cache with the MPT scheme are improved by 25.4% and 25.2%, respectively.

An observation is that the highest improvement in the T-cache hit ratio is obtained from gcc. Since the control sequence of gcc is very complex, i.e., gcc contains a large number of branch instructions and the direction of the branch instructions are rather random, the previous T-cache scheme (SPT) cannot provide high hit ratio.

Another observation is that the average T-cache hit ratio of 4 KB T-cache of MPT scheme is 61.2% and that of 16 KB T-cache of SPT scheme is 58.6%. This means that adopting the path classification is more effective than simply increasing the number of entries of T-cache in improving the T-cache hit ratio.

The increased T-cache hit ratio results in improving EFS, as shown in Table 3. For the 4 KB T-cache, the proposed MPT scheme improves EFS by 9.1% on the average; from 2.7% improvement for com95 to 16.2% improvement for gcc92 (gcc in SPECint92). The average EFS of the MPT scheme are improved by 9.6% and 10.2% compared to that of 16 KB and 64 KB T-cache, respectively.

As stated before, a disadvantage of the MPT scheme is that it can not work when a miss occurs on the I-cache, since the first basic block of a trace is fetched from the I-cache. Hence, a potential problem with the proposed MPT scheme is that its performance can be degraded when it is combined with a small size I-cache. To investigate the performance effect of I-cache size, we performed a set of simulations with varying I-cache size from 8 KB to 128 KB, keeping the T-cache size constant at 4 KB.

Figure 5 shows the average T-cache hit ratio as a function of the I-cache size for both SPT and MPT schemes. The MPT scheme provides almost constant improvement in T-cache hit ratio irrespective of the

Table 2 Trace cache hit ratio (%) for Rotenberg's SPT scheme and the proposed MPT scheme ($Improvement(\%) = (B - A)/A \times 100$).

Benchmarks	4 KB TC			16 KB TC			64 KB TC		
	SPT (A)	MPT (B)	Improvement (%)	SPT (A)	MPT (B)	Improvement (%)	SPT (A)	MPT (B)	Improvement (%)
com92	82.5	91.2	10.5	85.6	93.2	8.9	85.6	93.9	9.7
eqn92	73.8	91.0	23.3	77.5	94.2	21.5	78.2	95.6	22.3
esp92	52.9	65.6	24.0	59.5	79.4	33.4	64.5	87.4	35.5
gcc92	26.0	45.7	75.8	42.2	64.9	53.8	55.3	80.3	45.2
sc92	46.1	66.4	44.0	60.6	75.5	24.6	71.2	86.7	21.8
com95	62.3	68.6	10.1	65.4	71.8	9.8	65.5	73.4	12.1
gcc95	20.0	38.6	93.0	33.1	54.6	65.0	50.4	76.7	52.2
ijpeg95	53.1	62.4	17.5	57.2	66.6	16.4	58.6	68.0	16.0
perl95	23.6	40.8	72.9	46.3	67.6	46.0	61.9	84.5	36.5
vor95	34.5	41.5	20.3	58.1	67.1	15.5	71.5	83.1	16.2
average	47.5	61.2	28.8	58.6	73.5	25.4	66.3	83.0	25.2

Table 3 Effective fetch size for Rotenberg's SPT scheme and the proposed MPT scheme ($Improvement(\%) = (B - A)/A \times 100$).

Benchmarks	4 KB TC			16 KB TC			64 KB TC		
	SPT (A)	MPT (B)	Improvement (%)	SPT (A)	MPT (B)	Improvement (%)	SPT (A)	MPT (B)	Improvement (%)
com92	11.0	11.5	4.5	11.2	11.5	2.7	11.2	11.5	2.7
eqn92	9.7	11.2	15.5	9.9	11.4	15.2	10.0	11.5	15.0
esp92	8.5	9.0	5.9	8.8	10.4	18.2	9.1	10.7	17.6
gcc92	6.8	7.9	16.2	7.6	8.8	15.8	8.3	9.5	14.5
sc92	8.5	9.4	10.6	9.4	10.0	6.4	10.0	10.7	7.0
com95	11.1	11.4	2.7	11.3	11.6	2.7	11.3	11.6	2.7
gcc95	6.7	7.4	10.4	7.3	8.1	11.0	8.1	9.1	12.3
ijpeg95	11.1	11.9	7.2	11.3	12.2	8.0	11.4	12.3	7.9
perl95	7.0	7.8	11.4	8.2	9.2	12.2	9.0	9.9	10.0
vor95	7.9	8.7	10.1	9.2	10.0	8.7	9.9	10.9	10.1
average	8.8	9.6	9.1	9.4	10.3	9.6	9.8	10.8	10.2

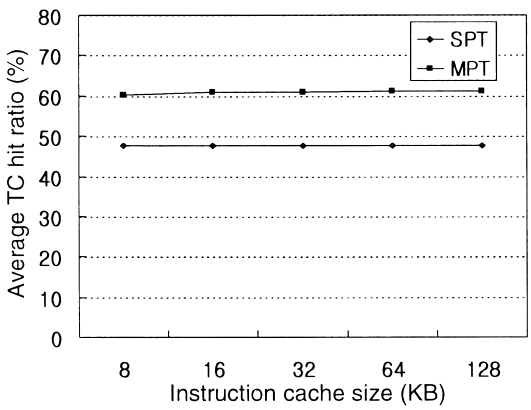


Fig. 5 Average T-cache hit ratio as a function of the I-cache size for both SPT and MPT schemes (T-cache size = 4 KB).

cache size. The average hit ratio in the MPT scheme is 60.3% for 8 KB, 60.9% for 16 KB and 32 KB, and 61.2% for 64 KB and 128 KB, while in the SPT scheme the average hit ratio is 47.5% for all sizes. Thus, the performance degraded by decreasing the cache size is not significant in the MPT scheme. This outcome is resulted from the fact that a trace is composed of the instructions fetched from the I-cache, i.e., the presence of a trace in the T-cache means that the instructions making the trace exist in the I-cache. If an entry of the I-cache is replaced by new instructions, the corresponding entry of the T-cache is also be replaced by new trace that is composed of the new instructions.

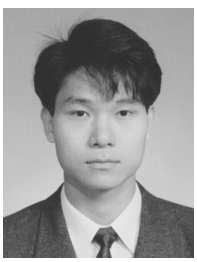
5. Conclusions

In this paper, we proposed a trace cache enhancement scheme called MPT. Paths are classified to improve the trace cache hit ratio by reducing the path conflict and basic blocks are joined to reduce the hardware cost needed to implement the trace cache. The hardware cost reduced by basic block joining technique is applied to implementing the trace classification technique to improve T-cache hit ratio by reducing path conflicts.

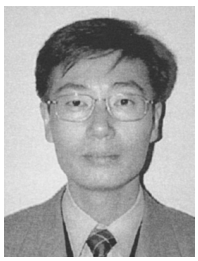
We simulated the proposed MPT scheme for various SPEC integer benchmark programs and evaluated the performance. At the same hardware cost as the SPT scheme, the MPT scheme improves the T-cache hit ratio by more than 25% and improves the effective fetch size by about 10%.

References

[1] M. Johnson, Superscalar microprocessor design, Prentice-Hall, Inc., 1991.
[2] A.J. Smith, "Cache Memories," ACM Computing Surveys, vol.14, no.3, pp.475–530, Sept. 1982.
[3] E. Rotenberg, S. Bennett, and J.E. Smith, "Trace cache: A low latency approach to highbandwidth instruction fetching," Proc. 29th Annual Int'l Symposium on Microarchitecture, pp.24–34, Dec. 1996.
[4] J.E. Smith and S. Vajapeyam, "Trace processors: Moving to fourth-generation microarchitecture," IEEE Comput., vol.30, no.9, pp.68–74, Sept. 1997.
[5] M.H. Lipasti and J.P. Shen, "Superspeculative microarchitecture for beyond AD 2000," IEEE Comput., vol.30, no.9, pp.59–67, Sept. 1997.
[6] J.S. Yim, I.C. Park, and C.M. Kyung, "SEWD: A cache architecture to speed up the misaligned instruction prefetch," IEICE Trans. Inf.& Syst., vol.E80-D, no.7, pp.742–745, July 1997.
[7] B.K. Gunther, "An integrated pre-access architecture for CMOS SRAM," IEEE J. Solid-State Circuits, vol.27, no.6, pp.901–907, June 1992.
[8] T.Y. Yeh, D.T. Marr, and Y.N. Patt, "Increasing the instruction fetch rate via multiple branch prediction and a branch address cache," Proc. 7th Int'l Conference on Supercomputing, pp.67–76, July 1993.
[9] J. Larus, "Efficient program tracing," IEEE Comput., vol.26, no.5, pp.52–61, May 1993.
[10] T.Y. Yeh and Y.N. Patt, "Alternative implementations of two-level adaptive branch prediction," Proc. 19th International Symposium on Computer Architecture, pp.124–134, May 1992.



Jin-Hyuk Yang received the B.S. degree in Electronic Engineering from Kyungpook National University, Korea in 1992, and the M.S. degrees in Electrical Engineering from KAIST (Korea Advanced Institude of Science and Technology), Korea in 1994. He is currently pursuing the Ph.D. degree in Electrical Engineering in KAIST. His current research interests include VLSI architecture for DSP and general-purpose microprocessor and design methodology for VLSI.



In-Cheol Park received the B.S. degree in Electrical Engineering from Seoul National University in 1986, the M.S. and Ph.D. degrees in Electrical Engineering from KAIST (Korea Advanced Institude of Science and Technology), in 1988 and 1992, respectively. From May 1995 to May 1996, he worked at IBM T.J. Watson Research Center, Yorktown, New York as a postdoctoral member of the technical staff in the area of circuit design. He joined KAIST in June 1996 as an Assistant Professor in the Department of Electrical Engineering. His current research interest includes CAD algorithms for high-level synthesis and VLSI architectures for general-purpose microprocessors.

Chong-Min Kyung received the B.S. degree in Electronic Engineering from Seoul National University, Korea, in 1975, and the M.S. and Ph.D. degrees in Electrical Engineering from KAIST (Korea Advanced Institute of Science and Technology), Korea, in 1977 and 1981, respectively. After graduation from KAIST, he worked at AT&T Bell Laboratories, Murray Hill, NJ, from April 1981 to January 1983 in the area of semiconductor device and process simulation. In February 1983, he joined the Department of Electrical Engineering at KAIST, where he is now a Professor. His current research interests include microprocessor/DSP architecture, chip design and verification methodology. He is Director of the IDEC (Integrated Circuit Design Education Center) established to promote the VLSI design education in Korean universities through CAD environment setup, chip fabrication services, and providing various educational materials and media related with integrated circuits and systems design.