Variation-Aware Instruction Rescheduling: Reducing Instruction Cache Leakage in Processor-based Embedded Systems

Goudarzi, Maziar
System LSI Research Center, Kyushu University

Ishihara, Tohru
System LSI Research Center, Kyushu University

出版情報：電気関係学会九州支部連合大会, 2007-10
バージョン：
権利関係：

Variation-Aware Instruction Rescheduling: Reducing Instruction Cache Leakage in Processor-based Embedded Systems

Maziar Goudarzi, Tohru Ishihara
(System LSI Research Center, Kyushu University, Japan {goudarzi, ishihara}@slrc.kyushu-u.ac.jp)

1 Introduction

Random Dopant Fluctuation (RDF) [1] within the same die results in changes in the threshold voltage (V_{th}) of transistors. Transistors of cache SRAM cells are even more affected due to their smaller physical channel area. The mismatch among V_{th} of transistors of a single SRAM cell results in different leakage currents depending on the value stored in the cell. Thus leakage of the cache memory can be reduced if the values with less leakage can be more often stored in each SRAM cell. To realize this idea in instruction cache, we reorder the instructions within each basic-block of the program such that they better match the less leaky state of their corresponding cache cells.

2 Value-Dependence of SRAM Leakage

When the SRAM cell is storing a 1 (Figure 1) only three transistors contribute to the total leakage (M1,2, 5); when storing a 0, the other three transistors leak.

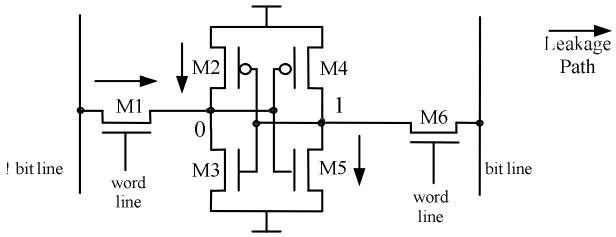

Figure 1: Different transistors leak based on cell value.

Since subthreshold leakage exponentially depends on V_{th}, total leakage can be significantly different in the two states. Figure 2 shows that at 60mv variation in V_{th}, this difference is 57% averaged over 1000 16KB caches. Moreover, the difference between the lowest and the highest total leakage of the cache is 70% on average.

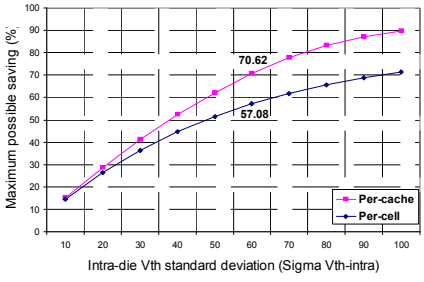

Figure 2: Leakage difference in 16KB caches.

3 Our Approach

Instructions within a basic-block can be rearranged (subject to dependency among them) to better match their corresponding SRAM cells in the instruction cache. By offline testing, the less leaky state of each SRAM cell is determined. Then, the above rearranging is applied to the binary executable of each application. We implemented two algorithms: an exhaustive search and a list-scheduling one. The former, which theoretically gives the optimum result with low efficiency, is only used to check optimality of the latter which showed 4 orders of magnitude higher efficiency with only 12% less optimality in our experiments.

4 Experimental Results

We applied the technique on five embedded benchmarks compiled with no compiler option for M32R processor. Table 1 shows the results of the list-scheduling algorithm each averaged on 1000 2KB caches with 4-byte cache-lines and 60mv standard deviation of within-die V_{th} variation; execution-times correspond to a 3.8GHz Xeon processor with 3.5GB memory. At higher variations resulting from technology scaling, the saving increases (Figure 3).

Table 1: Leakage reduction results on 2KB caches.

Cache config.		1-way	2-way	4-way	8-way
Leakage Saving (%)	FIR	11.90	12.30	11.86	11.75
	Compress	10.21	5.94	3.05	2.65
	MPEG2	9.05	6.35	4.61	3.45
	FFT	8.80	6.01	4.56	3.61
	JPEG	6.34	4.20	2.10	0.78
	Average	9.26	6.96	5.24	4.45
Avg. exec. time (s)		0.03	0.03	0.04	0.04

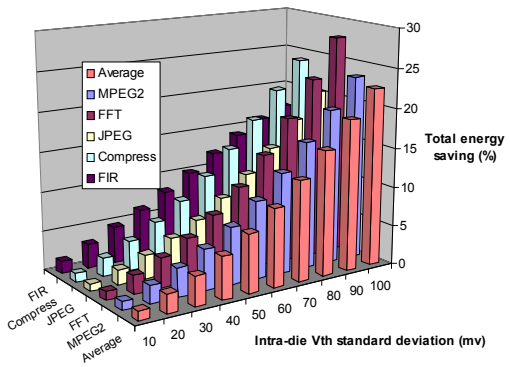

Figure 3: Saving increases with technology scaling.

4 Conclusion

We took advantage of the value-dependence of the leakage in SRAM cells due to within-die process variation and proposed a technique that reduces leakage even beyond standby mode when the cache is actively in use. Register-renaming can further improve the results.

Acknowledgement

This work is supported by CREST project of JST.

References

[1] Y. Taur, T.H. Ning, Fundamentals of Modern VLSI Devices, Cambridge University Press, 1998.