Nov. 2003, Vol.18, No.6, pp.833–838 J. Comput. Sci. & Technol.

Evaluation and Choice of Various Branch Predictors for Low-Power Embedded Processor

FAN DongRui (范东睿)[1], YANG HongBo (杨洪波)[2], GAO GuangRong (高光荣)[2] and ZHAO RongCai (赵荣彩)[1]

[1]*Institute of Computing Technology, The Chinese Academy of Sciences, Beijing 100080, P.R. China*

[2]*Department of Electrical and Computer Engineering, University of Delaware, Newark, DE 19716, USA*

E-mail: {fandr,rczhao}@ict.ac.cn; {hyang,ggao}@capsl.udel.edu

Received May 16, 2002; revised April 8, 2003.

Abstract Power is an important design constraint in embedded computing systems. To meet the power constraint, microarchitecture and hardware designed to achieve high performance need to be revisited, from both performance and power angles. This paper studies one of them: branch predictor. As well known, branch prediction is critical to exploit instruction level parallelism effectively, but may incur additional power consumption due to the hardware resource dedicated for branch prediction and the extra power consumed on mispredicted branches. This paper explores the design space of branch prediction mechanisms and tries to find the most beneficial one to realize low-power embedded processor. The sample processor studied is Godson-like processor, which is a dual-issue, out-of-order processor with deep pipeline, supporting MIPS instruction set.

Keywords branch predictor, embedded processor

1 Introduction

Over the past decade, branch prediction was extensively studied for its critical role in exploiting instruction level parallelism (ILP). Evolving from simple taken or not-taken, to one-bit, two-bit mechanism[1], bimodal mechanism[1], 2lev[2,3] and Hybrid[3−5] mechanisms, more and more complex branch prediction schemes[6−8] have been proposed to achieve higher prediction accuracy in high performance processors.

However, the scenario is much different in embedded systems where power consumption is becoming increasingly important. It is shown that hardware resources needed to achieve high prediction accuracy consume a notable fraction of on-chip power, more than 10%[9], presenting a challenging tradeoff for embedded systems. A good design can benefit from a comprehensive study on the performance, power and area of different branch prediction schemes.

This paper studies impacts of different prediction mechanisms on performance and power of MiBench[10,11] benchmarks as well as several sample programs with abundant branches, using SimpleScalar3.0 simulator[12] and Wattch1.03 power simulator[13]. Major results of our experiments are as follows. 1) We find that, compared with using 1K-entry *gshare*, energy-delay-product (EDP) of the processor using more advanced branch predictors like a 32K-entry *gshare* predictor on an average increase 6.1%, and that compared with using 1K-entry 3-bit wide GAs, EDP of the processor using a 32K-entry 8-bit wide GAs predictor on an average increase 5.7%, thus the aggressive predictors are not good choice in our context. 2) Our results show that a simple 2-level branch prediction scheme such as the *gshare* predictors is a good choice in terms of power for the architecture configuration we simulated. Experiments show, compared with that using 1K-entry *gshare*, EDP of the processor on an average increase 6.3% when using a combined predictor, which is composed of 8K-entry meta-table, 16K-entry bimodal and 1K-entry 8-bit wide BHT, 4K-entry PHT 2lev. 3) we observe that the differences in EDPs of different branch predictors are narrowed if clock-gating is applied. It justifies the use of more aggressive branch predictors when clock gating is employed.

2 Background

There have been considerable interests in low-

The work was supported by the Pilot Project of Knowledge Innovation Program of the Chinese Academy of Sciences under Grant No.KGCX2-109 and the National High-Tech Research and Development 863 Program of China under Grant No.2001AA111091.

power design of microarchitecture[14,15]. Most of previous work focused on particular components like cache configurations[16] and code compression to reduce bus traffic[17] rather than branch predictors. A recent work studying the power issues related to branch prediction was done by Kevin Skadron[9], whose work was based on 6-issue high-end processors, using SPEC2000 benchmarks as their workloads. Although interesting, problems remain open as to which extent the results in [9] will apply to embedded processors. In fact, the results from our experiments reported here show totally different conclusions for embedded systems.

3 Methodology

In this paper, our architectural model is a Godson-like[18] processor, and Table 1 shows the configuration of the simulated processor, in which the FP ALU and FP multiplier are fully pipelined.

Table 1. Configuration of the Simulated Processor

Issue width	2 instructions per cycle
Instruction window	RUU=8
Pipeline length	8 cycles
Functional units	1 Int ALU; 1 Int mult/div
	1 FP ALU; 1 FP mult/div
L1 D-cache	8KB; 2-way; 32B cache block
L1 I-cache	8KB; 2-way; 32B cache block
L1 I-cache latency	1 cycle
L1 D-cache latency	1 cycle
L2 cache	None; 2 memory ports
Memory latency	18 cycles
TLB size	48-entry
TLB miss penalty	30 cycles

We use Wattch version 1.03 on Mandrake 8.1 Linux OS to simulate the power and performance of the Godson processor. Wattch is a simulator for estimating microprocessor power dissipation at the architecture level. It is a cycle-by-cycle simulator and has been demonstrated to be fast and accurate[12,13]. We use the virtual PISA instruction set which is a superset of MIPS since Godson, the target processor we are studying, supports MIPS instruction set. We use two sets of benchmarks: (1) *dijkstra*, *stringsearch* and *rijndael* from MiBench, and (2) sample applications with abundant branch instructions, e.g., *fabnc, queen* and *sort*. The reason we choose these benchmarks is to include a set of diverse applications of embedded systems, covering network processing, data security, and office automation.

4 Experiments and Results

We choose seven types of branch predictors, including *bimodal*[1], *gshare*, GAg, GAs, PAg, PAs[2,19] and Comb[1,5], and we compare more than fifty configurations to find out the relationship between power and performance. The configurations contain those referred to in [2, 9, 19]. The structure of Comb is like Hybrid, but it is the combination of a bimodal and a 2lev. Comb_1 is composed of 1K-entry meta-table, 2K-entry bimodal and 512-entry 2-bit wide BHT, 512-entry PHT 2lev predictor. Comb_2 is composed of 2K-entry meta-table, 4K-entry bimodal and 1K-entry 7-bit wide BHT, 1K-entry PHT 2lev predictor. Comb_3 is composed of 8K-entry meta-table, 16K-entry bimodal and 1K-entry 8-bit wide BHT, 4K-entry PHT 2lev predictor.

Major results of our experiments are as follows.

We find that, compared with using 1K-entry *gshare*, EDP of the processor using more advanced branch predictors like a 32K-entry *gshare* predictor on an average increase 6.1%, and that compared with using 1K-entry 3-bit wide GAs, EDP of the processor using a 32K-entry 8-bit wide GAs predictor on average increases 5.7%, thus the aggressive predictors are not good design choice in our context.

Our results show that a simple 2-level branch prediction scheme such as the *gshare* predictors is a good choice in terms of power for the architecture configuration we simulated. Experiments show, compared with using 1K-entry *gshare*, EDP of the processor on an average increases 6.3% when using a combined predictor, which is composed of 8K-entry meta-table, 16K-entry bimodal and 1K-entry 8-bit wide BHT, 4K-entry PHT 2lev.

We observe that the differences in EDPs of different branch predictors are narrowed if clock gating is applied. It justifies the use of more aggressive branch predictors when clock gating is employed.

4.1 A Simple Predictor May Be a Good Choice

First, for each individual predictor, our results in Fig.1 show a general trend: the more complex the increasing size, the higher EDP it has. And from Fig.2, we focus on *bimodal* and *gshare* to reveal this trend. That means choosing aggressive branch predictors is not suitable for the embedded processors. At the same time, we find this phenomenon is more apparent when the processor processes moderate-size applications, which often constitute the major workloads of embedded processors. So we believe

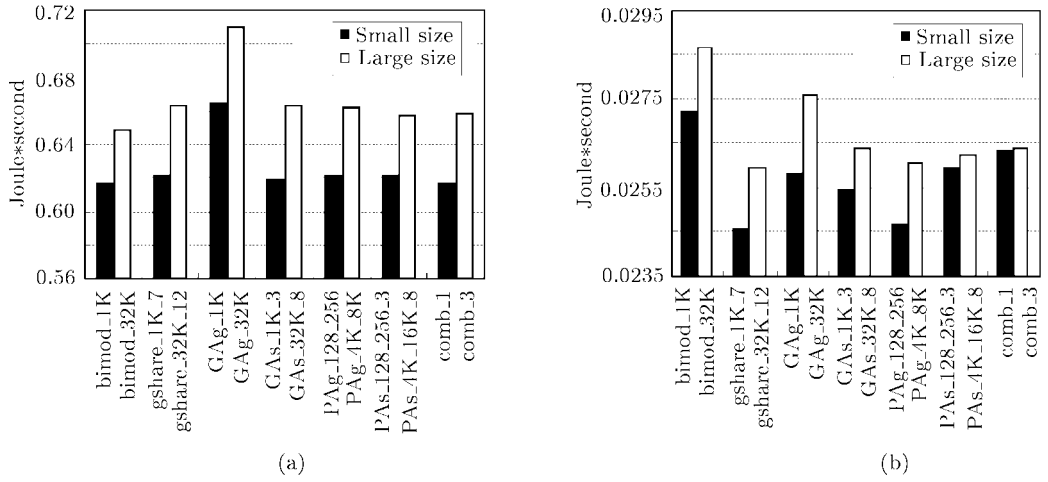

Fig.1. (a) Average EDP of *dijkstra, rijndael* and *stringsearch*. (b) Average EDP of *fabnc, queen* and *sort*.

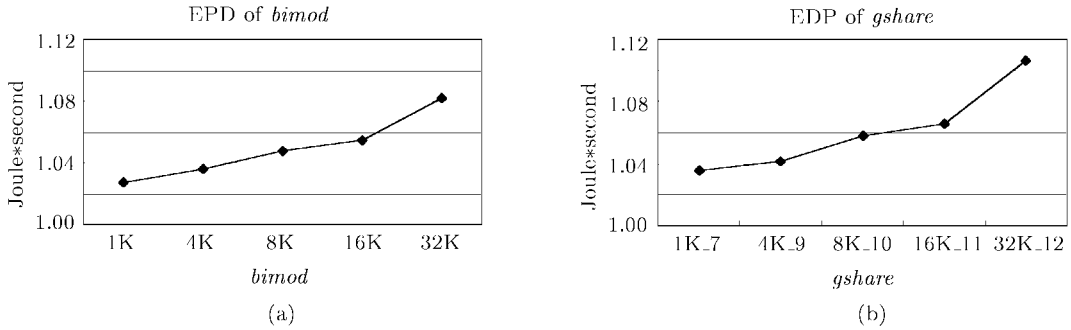

Fig.2. Average EDP of *dijkstra, rijndael* and *stringsearch*. (a) For *bimodal* predictors of different sizes. (b) For *gshare* predictors of different sizes.

choosing modest branch prediction mechanism can meet low power demand of embedded architectures.

But there seems to have inconsistency as below. As we know, higher prediction accuracy always means lower latency and less power waste because the processor can focus its resources on most of the necessary work in a certain time, and it can avoid unnecessary latency caused by misprediction, that is, saving power. On the other hand, complex and large predictors always bring on higher prediction accuracy[4−6]. But this is inconsistent with Fig.1 and Fig.2. These two figures show that complex prediction mechanisms are not good choice when EDP is considered for the architecture we simulate. We will analyze the reason for this inconsistency as below.

First of all, it is seen how much benefit the large predictors get for the architecture we simulate. Fig.3(a) shows the delay of MiBench and Fig.3(b) shows the energy consumption of MiBench. The data in Fig.3(a) multiplying the data in Fig.3(b) are the EDPs in Fig.2. From Fig.3, we find when using more complex predictors the delay keeps nearly stable, but the power increases quickly. So we must analyze why we cannot reduce delay when using aggressive predictors. Fig.4(a) is the various average IPC values of MiBench of different-sized predictors. To our surprise, we find the IPC of aggressive predictors is only a little higher, no more than 2 percent, than the moderate ones for the embedded benchmarks–MiBench. Why does the performance using aggressive prediction mechanism increase so little? We can get the answer from Fig.4(b). Fig.4(b) is the average prediction accuracy of MiBench for different-sized predictors. It is illustrated that the prediction accuracy does not increase much as we expected when using aggressive branch prediction mechanisms. That is because the prediction accuracy increases a little when using aggressive prediction mechanism and

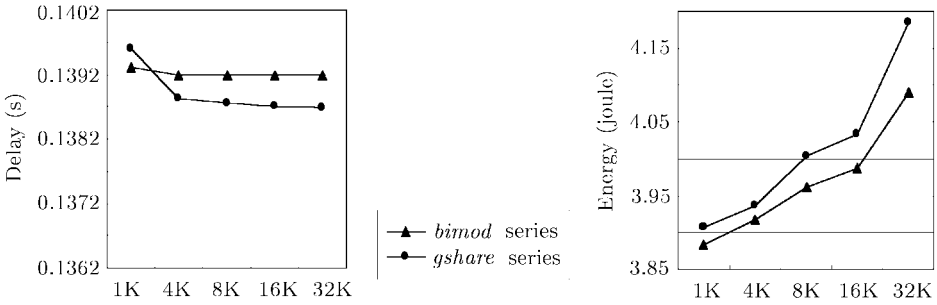

Fig.3. (a) Average delay change when using different-sized *bimod* and *gshare* predictors for MiBench. (b) Average energy change when using different-sized *bimod* and *gshare* predictors for MiBench.

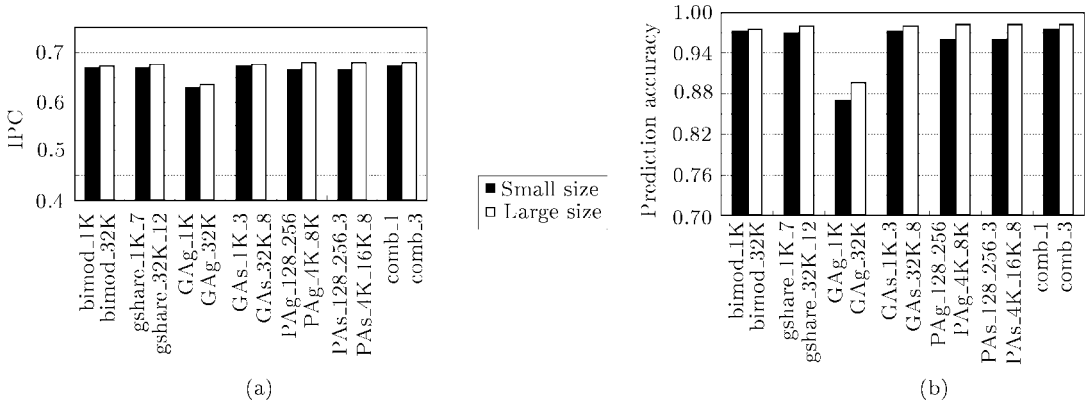

Fig.4. Performance for different-sized predictors. (a) Average IPC of MiBench. (b) Average prediction accuracy of MiBench.

this is caused by the features of embedded applications. So for the embedded processors, relatively simple predictors can satisfy the needs of embedded applications. Through the above analysis, we can make a conclusion that using simple predictors for embedded processors is a good choice.

4.2 Choosing Suitable Branch Predictors

Now, let us pick the lowest EDP from each prediction scheme and compare these EDPs only. As shown in Fig.5, compare bimod_1K, bimod_2K, bimod_4K, *gshare*_1K_7, *gshare*_4K_9, GAg_1K, GAs_1K_3, GAs_2K_4, PAg_128_256, PAg_1K_2K, PAs_128_256_3, PAs_2K_4K_8, comb_1 and comb_2 to seek for a best one for Godson-like processors. From Fig.5 (a), we can see that 1K-entry bimodal, 1K-entry *gshare* with 7-bit wide history, 1K-entry GAs with 3-bit wide history, and Comb_1 have relatively similar low EDP. From Fig.5 (b), we can

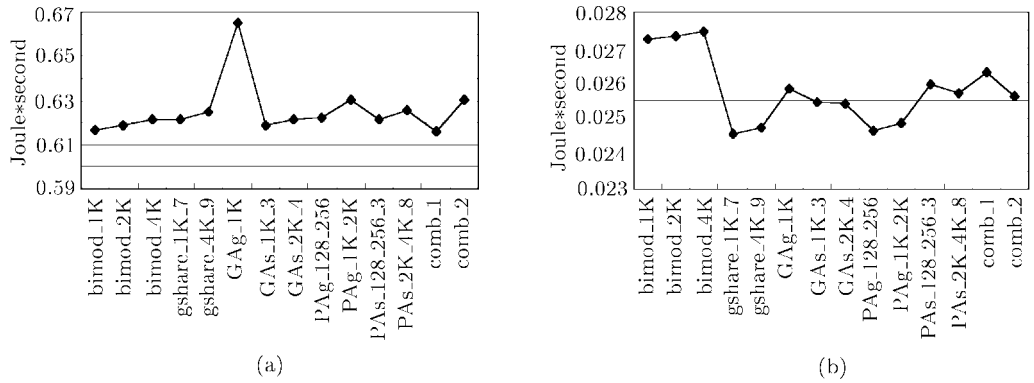

Fig.5. (a) Average EDP of *dijkstra*, *rijndael* and *stringsearch* for modest branch prediction configurations. (b) Average EDP of fabnc, queen and sort for modest branch prediction configurations.

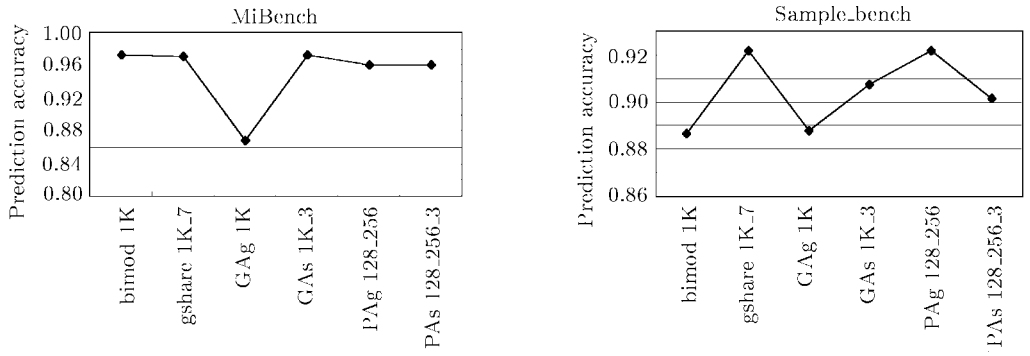

Fig.6. Compare prediction accuracy of different predictors which use nearly the same hardware consumption.

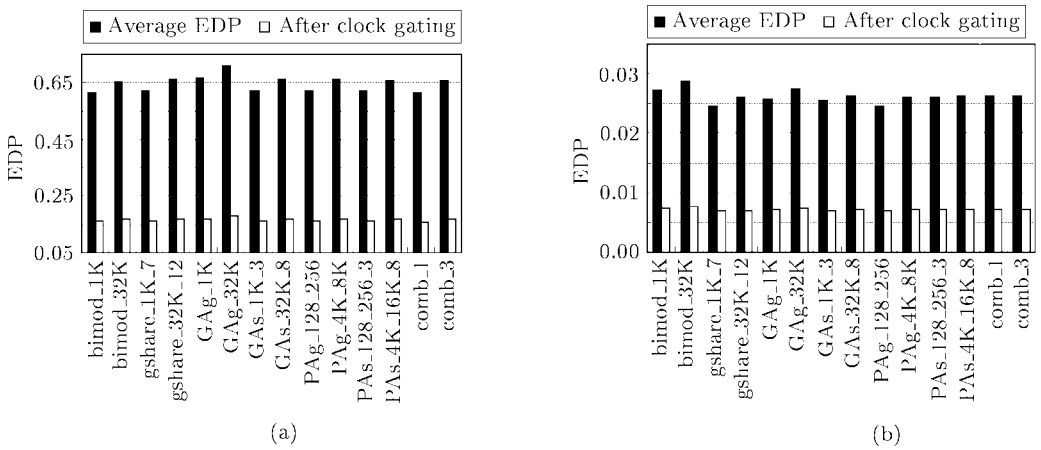

Fig.7. (a) Compare average EDP of *dijkstra*, *rijndael* and *stringsearch* with that after using clock gating. (b) Compare average EDP of *fabnc*, *queen* and *sort* with that after using clock gating.

see 1K-entry *gshare* and 128-entry 8-bit wide BHT, 256-entry PHT PAg are better. So 1K-entry *gshare* with 7-bit wide history should be a good choice for both MiBench and small sample applications in terms of EDP.

Why small *gshare* appears good? First, as we know, compared with other modest predictors it has less hardware consumption, only contains 2K bits. Less hardware consumption natively means less power consumption. On the other hand, we should analyze the prediction accuracy of *gshare*. Fig.6 is to compare the prediction accuracy of these modest predictors, which use nearly the same hardware consumption. From Fig.6, we know for both sets of benchmarks 1K-entry *gshare* has higher prediction accuracy, though it is not the best for each set of benchmarks. This also illustrates the fact that *gshare* predictor can well fit different kinds of applications. This is another reason to use *gshare* as a good candidate for the embedded processors.

4.3 Effect of Clock Gating on Branch Prediction

Fig.7 is to compare the energy-delay-product before and after using clock gating. We choose the option cc3 of the simulator, which is aggressive, non-ideal (some fraction is still consumed when disabled) conditional clocking. From Fig.7, we find two phenomena when using clock gating. First, clock gating results in considerable power saving. Second, although the conclusions from Subsections 4.1 and 4.2 are still correct after using clock gating, the difference between different schemes is reduced. The first reason for the phenomena is that idle components are gated, thus the unnecessary waste of energy is avoided. The second reason is that clock gating can reduce power consumption caused by misprediction. Since the EDPs of different-sized configurations are close when processors use clock-gating technology, choosing more aggressive branch

predictors may become a reasonable choice. When predictor of bigger size is used, accuracy and performance will increase.

5 Conclusion and Future Work

In this paper we showed impacts of different branch prediction mechanisms on performance and power for Godson-like embedded processor. Through lots of experiments and analysis, the prediction mechanisms suitable for embedded processors are figured out from so many branch prediction methods. Furthermore, we studied the relationship between branch prediction and clock gating technology as we described in this work.

There is still much remaining to be done in the future. When power consumption is considered, the relation among branch prediction and other components of processor is not very clear, and as process develops and new branch prediction methods appear, we must improve the simulator to adapt to the changes of architecture and to get more accurate data. Furthermore, today, pervasive computing is popular, modern embedded processors explore more parallelism, so we will need to investigate more closely the tradeoffs between power consumption and parallelism exploitation.

References

[1] Smith J E. A study of branch prediction strategies. In *Proc. the 8th Int. Symp. Computer Architecture,* Minneapolis, 1981, pp.135–148.

[2] Yeh T Y, Patt Y N. Alternative implementations of two-level adaptive branch prediction. In *Proc. the 19th Int. Symp. Computer Architecture,* Queensland, 1992, pp.124–134.

[3] Yeh T Y, Patt Y N. Two-level adaptive training branch prediction. In *Proc. the 24th Annual Int. Symp. Microarchitecture,* Albuquerque, 1991, pp.51–61.

[4] Pan S T, So K, Rahmeh J T. Improving the accuracy of dynamic branch prediction using branch correlation. In *Proc. the Fifth Int. Conf. Architectural Support for Programming Languages and Operating Systems,* Boston, 1992, pp:76–84.

[5] Chang P Y, Hao E, Patt Y N. Alternative implementations of hybrid branch predictors. In *Proc. the 28th Annual Int. Symp. Microarchitecture,* Ann Arbor, 1995, pp.252–257.

[6] Evers M, Chang PY, Patt Y. Using hybrid branch predictors to improve branch prediction accuracy in the presence of context switches. In *Proc. 23rd Annual Int. Symp. Computer Architecture,* Philadelphia, 1996, pp.3–11.

[7] Maria-Dana Tarlescu, Kevin B Theobald, Guang R Gao. Elastic history buffer: A low-cost method to improve branch prediction accuracy. In *ICCD'97,* Austin, 1997, pp.82–87.

[8] Kevin B Theobald, Guang R Gao, Laurie J Hendren. Speculative execution and branch prediction on parallel machines. In *Proc. 1993 Int. Conf. Supercomputing,* Tokyo, 1993, pp.77–86.

[9] Parikh D, Skadron K, Zhang Y, Barcella M, Stan M. Power issues related to branch prediction. In *Proc. the 8th Int. Symp. High-Performance Computer Architecture,* Boston, 2002, pp.233–246.

[10] Matthew R Guthaus, Jeffrey S Ringenberg, Dan Ernst. MiBench: A free, commercially representative embedded benchmark suite. In *IEEE 4th Annual Workshop on Workload Characterization,* Austin, 2001, pp.1–12.

[11] MiBench Benchmarks Set. Available from http://www.eecs.umich.edu/~jringenb/mibench/

[12] Burger D, Austin T. The Simplescalar Tool Set. Version 2.0. Technical Report CSTR-97-1342, 1997, University of Wisconsin, Madison.

[13] Brooks D, Tiwari V, Martonosi M. Wattch: A framework for architectural-level power analysis and optimizations. In *Proc. the 27th Annual Int. Symp. Computer Architecture,* Vancouver, 2000, pp.83–94.

[14] David M Brooks, Pradip Bose, Stanley E Schuster *et al.* Power-aware microarchitecture: Modeling challenges for next-generation microprocessors. *IEEE Micro,* 2000, 20(6): 26–44.

[15] Gonzalez R, Horowitz M. Energy dissipation in general purpose microprocessors. *IEEE Journal of Solid-State Circuits,* 1996, 31(9): 1277–1284.

[16] Su C, Despain A. Cache design tradeoffs for power and performance optimization: A case study. *ACM/IEEE Int. Symp. Low-Power Design,* Dana Point, California, 1995, pp.63–68.

[17] Ramon Canal, Antonio González, James E Smith. Very low power pipelines using significance compression. In *Proc. the 33rd Annual Int. Symp. Microarchitecture,* Monterey, California, 2000, pp.181–190.

[18] An Introduction about the Godson project of Institute of Computing Technology, CAS. Available from http://www.ict.ac.cn/xinwen/dt011014_1.htm, 2002.

[19] Skadron K, Ahuja P S, Martonosi M, Clark D W. Branch prediction, instruction-window size, and cache size: Performance tradeoffs and simulation techniques. *IEEE Trans. Computers,* 1999, 48(11): 1260–1281.