Reducing Energy Consumption via Low-Cost Value Prediction

Toshinori Sato[1,2] and Itsujiro Arita[1]

[1] Department of Artificial Intelligence
[2] Center for Microelectronic Systems
Kyushu Institute of Technology
tsato@ai.kyutech.ac.jp

Abstract. Power consumption is becoming one of the most important constraints for microprocessor design in nanometer-scale technologies. Device engineers, circuit designers, and system architects are faced with many challenges. In the area of mobile and embedded computer platforms, power has already been a major design constraint. However, it is also a limiting issue in general-purpose microprocessors. In order to manage the impact of increasing microprocessor power consumption, some architectural-level techniques are required as well as circuit-level design improvements. In this paper, we propose to make any instruction in the program execution flow non-critical by using a low-cost value predictor in order to improve energy efficiency. Based on simulations, we find that up to 11.4% of energy reduction in functional units can be attained by utilizing value prediction.

1 Introduction

The increasing popularity of portable and mobile computer platforms such as laptop PCs and smart cell phones is a driving force in investigation of high-performance and power-efficient microprocessors. For example, Java-2 MicroEdition (J2ME) works on cell phones[9], allowing users to download several applications such as 3D-animated games and play them on their cell phones. Guides for travellers and flight ticket reservations are also available, and mobile banking and trading are provided. Therefore, it is required that embedded processors have high performance, and their designers have begun to include features that are traditionally found in general purpose processors. For example, modern embedded microprocessors support out-of-order execution[10]. As the computing power of microprocessors for mobile devices increases, however, their power consumption also increases. In addition, while power is already a major design constraint in the area of mobile and embedded computer platforms, it has also become a limiting issue in general-purpose microprocessors.

The active power P_{active} and gate delay t_{pd} of a CMOS circuit are given by

$$P_{active} = fC_{load}V_{dd}^2 \quad (1)$$

$$t_{pd} \propto \frac{V_{dd}}{(V_{dd} - V_{th})^{\alpha}} \tag{2}$$

where f is the clock frequency, C_{load} the load capacitance, V_{dd} the supply voltage, and V_{th} the threshold voltage of the device. α is a factor dependent upon the carrier velocity saturation and is approximately 1.3–1.5 in advanced MOSFETs[6]. Based on Eq.(1), it can easily be found that a power-supply reduction is the most effective way to lower power consumption. However, Eq.(2) tells us that reductions in the supply voltage increase gate delay, resulting in a slower clock frequency, and thus diminishing the computing performance of the microprocessor.

In order to mitigate the performance loss, we can exploit information regarding circuit criticality. There are a lot of device-level techniques for the design tradeoffs such as transistor size optimizations[5], multiple supply voltages[20], and multiple[22] and variable[6] threshold approaches. Power reduction without performance loss is achieved by selecting non-critical paths as candidates for low-power design. In other words, performance-oriented design is used only in speed critical paths. The same philosophy can be applied to architecture-level design. Only instructions on critical paths are executed on fast and power-hungry functional units, and non-critical instructions can be executed on slow and power-efficient units[13, 15, 17]. In this paper, we propose to translate any instructions into non-critical ones by using a low-cost value predictor[16] and to execute the translated non-critical instructions slowly. It is expected that processor performance is not be diminished if the value prediction accuracy is considerably high.

The organization of the rest of this paper is as follows: Section 2 surveys related work. Section 3 explains how to improve energy efficiency by generating non-critical instructions. Section 4 describes our evaluation environment. Section 5 shows simulation results. Finally, Section 6 concludes the paper.

2 Related Work

Value prediction[11] is a speculative technique which executes instructions using predicted data values. It breaks data dependence chains speculatively, resulting in critical path reduction. Many studies have proposed value prediction mechanisms, some of which achieve predictability rates as high as 80%[21]. However, these predictors such as 2-level and hybrid predictors require considerable hardware cost for realizing their high predictabilities. Some studies were performed to reduce hardware cost[2, 12, 16]. Morancho et al.[12] proposed to reduce hardware cost by classifying instructions based on their value predictability. Easily predictable instructions use simpler predictors, whose hardware cost is low. High-cost predictors are used only for hard-to-predict instructions. Calder et al.[2] proposed filtering instructions based on their level of criticality. Only instructions regarding critical paths are held in the value predictor, resulting in capacity saving. We proposed 0/1-value predictor[16], which generates only values 0 and 1. Since 22.6% of dynamic instructions on average generate the value 0 or 1, processors benefit from the 0/1-value predictor.

The critical path is a chain of dependent instructions, which determines the number of cycles executing the program. And thus, the performance of the processor is limited by the speed at which it executes the instructions along the critical path. If we can identify which instructions are critical, we can accelerate their execution by any means. Critical path prediction[4, 19] is such a technique for identifying critical instructions dynamically. Combination of the critical path predictor and the value predictor enables to break critical paths, resulting in boosting processor performance. Fields et al.[4] found that mispredictions can be reduced by up to 500% by using the information regarding critical instructions.

Exploiting information regarding instruction criticality is effective not only for improving processor performance but also for reducing energy consumption[3, 13, 15, 17]. Casmira et al.[3] studied the potential benefit of exploiting instruction criticality, which they call slack, for power reduction. They found there is significant availability of non-critical instructions, but did not mention any practical mechanism that utilize the results for power reduction. Pyreddy et al.[13] use profiled-based heuristics proposed by Tune et al.[19] for identifying critical instructions. From a profile run, each instruction is marked as critical or non-critical. When the program is executed, the critical instructions are executed on fast and power-hungry functional units and the non-critical ones are executed on slow and power-efficient units. Unfortunately, they could attain little power savings and caused significant performance loss. In contrast, Sato et al.[15] and Seng et al.[17] utilized a dynamic mechanism. They proposed to use the critical path predictor to identify non-critical instructions. Seng et al.[17] focused on eliminating hot spots of power density in general-purpose high-performance microprocessors. They reported significant gains in the ratio of performance and power density, but did not mention how energy consumption can be reduced. We evaluated the potential of energy reduction via this criticality-based instruction scheduling[15] because we are interested in embedded processors as well as general-purpose processors, and found that approximately 40% of energy consumed in functional units can be reduced.

3 Energy Reduction

This section explains how to reduce energy consumption by value prediction.

Because every value prediction requires verification whether it is correct or not, the number of instructions that are executed and committed is not reduced. On the contrary, it might increase when mispredictions are considered. Therefore, only utilizing value prediction can not reduce energy consumption. The key idea of energy reduction via value prediction is the verification should not be fast if it is expected that its prediction is highly accurate. In other words, predicted instructions are not on critical paths and thus they can be executed on low frequency functional units. Such units can be operate by low supply voltage, resulting in significant power reduction according to Eq(1). If a prediction is correct, there are no penalty on execution cycles and thus energy consump-

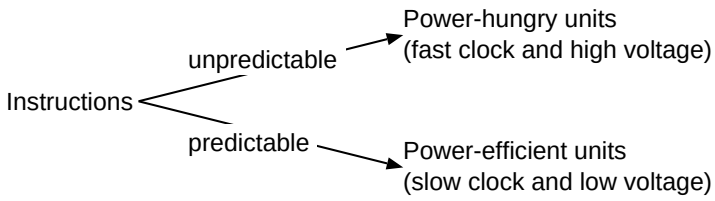


Fig. 1. Instruction execution policy

tion is reduced. Otherwise, there are considerable penalty due to long latency of each mispredicted instruction and re-execution of each misspeculated instruction, resulting in degrading energy efficiency. In summary, only unpredictable instructions are executed on fast and power-hungry functional units, and predictable instructions can be executed on the slow and power-efficient units, as shown in Figure 1.

The potential of energy reduction is estimated as follows. We decide that clock frequency and supply voltage for the slow units are half of those for the fast units. We assume that half of instructions are predictable and will be executed on the slow units. Energy consumption is calculated as follows. The number of total instructions are unchanged, and thus the percentage of instructions executed on the slow units determines the energy reduction. When the original energy consumption is defined as E, the fast units consumes $\frac{1}{2}E$, because its power consumption is unchanged and the number of instructions executed there becomes half. On the other hand, the slow units consumes $2 * \frac{1}{2} * \frac{1}{2} * (\frac{1}{2})^2 E$, because the number of instructions executed there, its clock frequency, and its supply voltage are all half while execution time is 2 times increased. Thus, total energy consumption is $\frac{5}{8}E$. In other words, 37.5% of energy reduction is attained.

It should be noted that the value predictor consumes power. If its power consumption is considerably large, the proposal is not effective on energy reduction. Therefore, we should use any low-cost value predictor whose power consumption is significantly small. In this study, we use the 0/1-value predictor[16] as a very low-cost predictor.

4 Methodology

In this section, we describe our evaluation environment by explaining a processor model and benchmark programs.

4.1 Processor Model

We use two types of simulators for this study. One is a functional simulator for evaluating predictability and the other is a timing simulator for evaluating processor performance. The timing simulator models wrong path execu-

Table 1. Execution latency

class	cycles	class	cycles
IntALU	1	FloatADD	3
IntMUL	3	FloatCMP	3
IntDIV	12	FloatCVT	3
		FloatMUL	3
		FloatDIV	12
		FloatSQRT	12

tion caused by misspeculations. We implemented the simulators using the SimpleScalar/PISA tool set (ver.3.0a)[1]. SimpleScalar/PISA instruction set architecture (ISA) is based on MIPS ISA.

The timing simulator models a 2-way out-of-order execution embedded processor, which is primary based on NEC VR5500[10]. Due to unavailability of the detailed information and to limitation of the tool set, several configurations of our processor model does not completely match those of VR5500. Our model has three integer ALUs, a floating-point unit, and a load-store unit. Branch instructions are executed on the ALUs, while VR5500 has a dedicated branch unit. The latency for execution is summarized in Table 1. We assume that the ALUs dynamically change its execution mode between fast and slow mode. This would be impractical assumption. However, we use this assumption as the first step toward exploiting dual-voltage pipeline. In the fast mode, they can execute most integer operations in one cycle as shown in Table 1, and in the slow mode they execute operations in two cycles. We assume that the ALUs are not pipelined in the slow mode to maintain their throughput in the fast mode. Thus, they diminishes their throughput by half in the slow mode.

A single-port, non-blocking, 32KB, 32B block, 2-way set-associative L1 data cache is used for data supply. It has a load latency of 1 cycle after the data address is calculated and a miss latency of 24 cycles. No L2 cache is used. A memory operation that follows a store whose data address is unknown cannot be executed. A 32KB, 32B block, 2-way set-associative L1 instruction cache is used for instruction supply. For control prediction, a simple branch predictor that has 4K 2-bit saturation counters, and an 8-entry return address stack are used. The branch predictor is updated at the instruction commit stage. A mispredicted branch generates a 6 cycle penalty. Out-of-order execution is based on a register update unit which has 32 entries, while VR5500 uses 20-entry reservation station.

For value prediction, we use the 0/1-value predictor[16]. Figure 2 presents the block diagram of the 0/1-value predictor. Its main structure is the Value History Table (VHT). The VHT is a direct-mapped table and has 1-bit `Data Value` and 2-bit `Conf` fields. As determined by a previous study[16], the 0/1-value predictor does not have tag field because it does not always contribute to predictor performance. The `Data Value` field stores the last result of the associated instruction. When an instruction produces the value 0 or 1, the `Data`

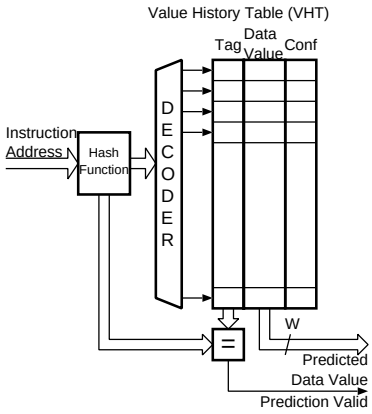


Fig. 2. 0/1-value predictor

Table 2. Supply voltage and frequency scaling

	clock frequency / supply voltage
TM5800	400MHz/0.93V — 800MHz/1.3V
XScale	500MHz/1.15V — 1.0GHz/1.8V

`Value` field simply holds the value. Otherwise, it keeps the value 0. In other words, the 0/1-value predictor has a priority on the value 0. This decision is proper since the value 0 is the most frequently generated value[16]. When the same instruction is encountered, the `Data Value` is used for its predicted value. The `Conf` field is a saturating up-down counter that determines whether or not the instruction should be predicted. The counter is incremented or decremented whenever a prediction is correct or incorrect, respectively. In our study, 2-bit saturating counters are used. When a misspeculation occurs, it is necessary to revert the processor state to a safe point where the speculation is initiated. We use an instruction reissue mechanism which selectively flushes and reissues misspeculated instructions. We have already proposed a practical instruction reissue mechanism[14].

We will evaluate two scalings for supply voltage and clock frequency; one is based on Transmeta TM5800[18], and the other is based on Intel XScale[7]. These scalings are summarized in Table 2.

4.2 Benchmark Programs

The MediaBench[8] is used for this study. We use original input files provided by UCLA. Table 3 lists the benchmarks and the input sets. All programs are

Table 3. Benchmark programs

program	input set	%cand
g721-decode	clinton.g721	71.7%
g721-encode	clinton.g721	72.1%
gsm-decode	clinton.pcm.run.gsm	74.5%
gsm-encode	clinton.pcm	83.3%
mpeg2-decode	mei16v2.m2v	70.2%
mpeg2-encode	options.par	78.6%

compiled by the GNU GCC (version 2.6.3) with the optimization options specified by UCLA. Each program is executed to completion. Table 3 also shows the percentage of candidate instructions predicted by the 0/1-value predictor, when the programs are executed on the timing simulator. The candidate instructions for value prediction are register-writing ones, and do not include branch and store instructions. We count only committed instructions.

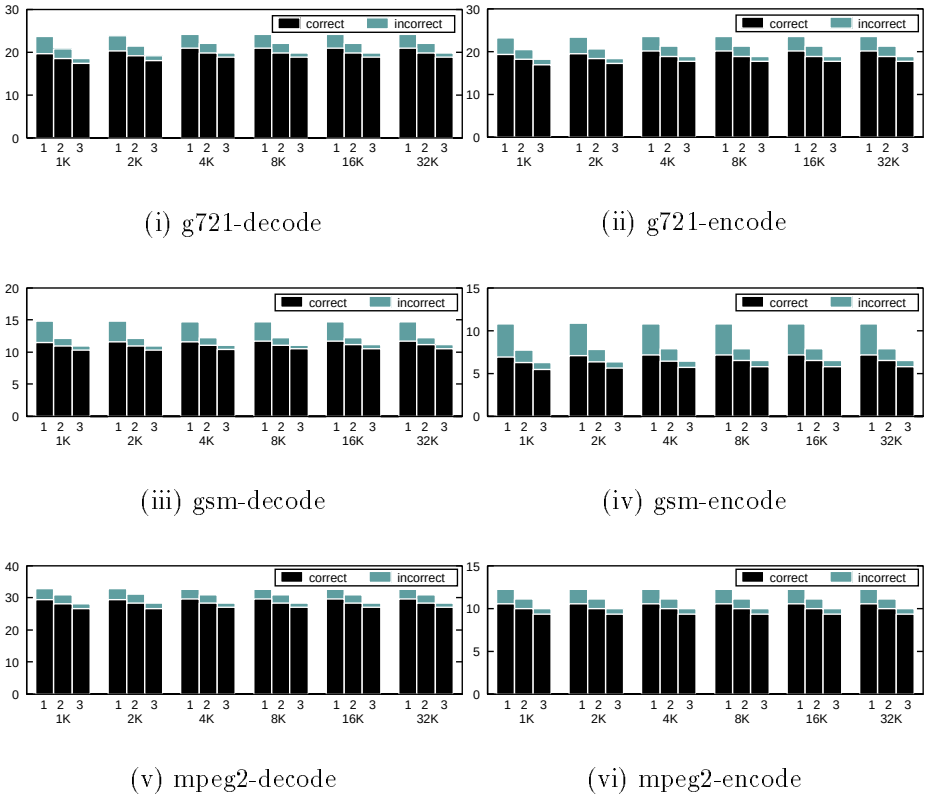
5 Results

In this section, we present simulation results. First, we evaluate predictability. We define predictability as the number of instructions that are (correctly and incorrectly) predicted by a value predictor over that of all register-writing instructions. We also define prediction coverage as the number of instructions correctly predicted over that of all register-writing instructions. On the other hand, prediction accuracy, which is the percentage of instructions correctly predicted over all predicted instructions, can be easily obtained using the following equation.

$$(Prediction\ accuracy) = \frac{(Prediction\ coverage)}{(Predictability)}$$

After that, processor performance and energy efficiency will be evaluated.

There is a tradeoff between predictability and prediction accuracy. In order to execute instructions on the slow units as much as possible, predictability should be high. On the other hand, in order not to diminish energy efficiency, prediction accuracy also should be high. In general, increasing predictability reduces prediction accuracy, because hard-to-predict instructions must be included. Therefore, we should investigate the tradeoff point carefully. In this study, we examine the threshold value for initiating prediction and the prediction table capacity. The threshold value is varied between 1 and 2, and the capacity is varied between 1K and 32K entries. Figure 3 shows the results. Note that the functional simulator is used in these evaluations. Each bar in Figure 3 is divided into two parts. The lower part (black) indicates the percentage of the instructions whose data value is correctly predicted. The upper part (gray) indicates the percentage that is mispredicted. That is, the lower part is the prediction coverage, while the sum



(i) g721-decode (ii) g721-encode

(iii) gsm-decode (iv) gsm-encode

(v) mpeg2-decode (vi) mpeg2-encode

Fig. 3. (%) Value predictability

of the two parts is the predictability. For each group of three bars, the first one indicates the result for the case where threshold value is 1. The rest bars are for the cases where threshold values are 2 and 3, respectively. From the results, it can be observed that the threshold value of 2 and the table capacity of 1K entries are in the good tradeoff point. In this configuration, hardware cost of the prediction table is only 384 byte and is less than 0.6% of that of caches. Hence, its power consumption is less considerable and will be ignored in the rest of this paper.

Next, we evaluate processor performance and energy efficiency. Now, we use the timing simulator for the evaluations. Table 4 presents the percent increase of the execution cycles. Because penalty of each misprediction becomes large, value prediction can not contribute to processor performance. However, it is fortunately that the large penalty does not have serious impact on performance. On the whole, processor performance remains the same before and after using value prediction. Figure 4 shows energy reduction of the functional units according to the assumptions described in Table 2. It is observed that energy consumption of

Table 4. (%) Increase of execution cycles

program	increase
g721-decode	0.38%
g721-encode	0.32%
gsm-decode	−0.37%
gsm-encode	0.15%
mpeg2-decode	−0.03%
mpeg2-encode	−0.21%

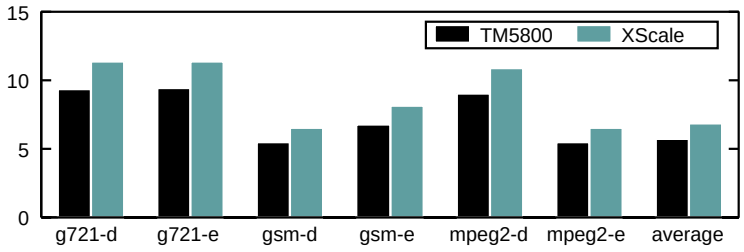


Fig. 4. (%) Energy reduction

the functional units is reduced by up to 9.4% and 11.4% in the cases of TM5800 and XScale models, respectively. These results confirm that energy efficiency can be improved by using value prediction.

6 Conclusions

Power consumption is becoming one of the most important constraints for microprocessor design. In this paper, we proposed to reduce energy consumption by making any instruction in the program execution flow non-critical via the 0/1-value predictor. Predictable instructions are executed on the slow and power-efficient units, resulting in reducing energy consumed in the functional units. From the detailed simulation results, we found that up to 11.4% of energy reduction can be attained by utilizing value prediction.

Acknowledgments

This work is supported in part by a Grant-in-Aid for Scientific Research (B) (#12780273) from the Japan Society for the Promotion of Science.

References

1. Burger D., Austin T.M.: The SimpleScalar tool set, version 2.0. ACM SIGARCH Computer Architecture News, 25(3) (1997)
2. Calder B., Reinman G., Tullsen D.M.: Selective value prediction. 26th Int. Symp. on Computer Architecture (1999)
3. Casmira J., Grunwald D.: Dynamic instruction scheduling slack. Kool Chips Workshop (2000)
4. Fields B.A., Rubin S., Bodik R.: Focusing processor policies via critical-path prediction. 28th Int. Symp. on Computer Architecture (2001)
5. Hashimoto M., Onodera H.: Post-layout transistor sizing for power reduction in cell-based design. IEICE Trans. Fundamentals, E84-A(11) (2001)
6. Hiramoto T., Takamiya M.: Low power and low voltage MOSFETs with variable threshold voltage controlled by back-bias. IEICE Trans. Electronics, E83-C(2) (2000)
7. Intel Corporation: Intel$^{(R)}$ XScale^{TM} technology. `http://developer.intel.com/design/intelxscale/` (2002)
8. Lee C., Potkonjak M., Mangione-Smith W.H.: MediaBench: a tool for evaluating and synthesizing multimedia and communications systems. 30th Int. Symp. on Microarchitecture (1997)
9. Levy M: Java to go: part 1: Microprocessor Report, 15(2) (2001)
10. Levy M. NEC processor goes out of order. Microprocessor Report, 15(9) (2001)
11. Lipasti M.H., Wilkerson C.B., Shen J.P.: Value locality and load value prediction. 7th Int. Conf. on Architectural Support for Programming Languages and Operation Systems (1996)
12. Morancho E., Llaberia J.M., Olive A.: Split last-address predictor. 8th Int. Conf. on Parallel Architectures and Compilation Techniques (1998)
13. Pyreddy R., Tyson G.: Evaluating design tradeoffs in dual pipelines. Workshop on Complexity-Effective Design (2001)
14. Sato T.: Evaluating the impact of reissued instructions on data speculative processor performance. Microprocessors and Microsystems, 25(9-10) (2002)
15. Sato T., Koushiro T., Chiyonobu A., Arita I.: Power and performance fitting in nanometer design. 5th Int. Workshop on Innovative Architecture for Future Generation High-Performance Processors and Systems (2002)
16. Sato T., Arita I.: Low-cost value predictors using frequent value locality. 4th Int. Symp. on High Performance Computing, LNCS2327 (2002)
17. Seng J.S., Tune E.S., Tullsen D.M.: Reducing power with dynamic critical path information. 34th Int. Symp. on Microarchitecture (2001)
18. Transmeta corporation: Crusoe^{TM} processor model TM5800. Product Brief (2001)
19. Tune E., Liang D., Tullsen D.M., Calder B.: Dynamic prediction of critical path instructions. 7th Int. Symp. on High Performance Computer Architecture (2001)
20. Usami K., Horowiz M.: Clustered voltage scaling technique for low-power design. Int. Symp. on Low Power Design (1995)
21. Wang K., Franklin M.: Highly accurate data value prediction using hybrid predictors. 30th Int. Symp. on Microarchitecture (1997)
22. Wet L., Chen Z., Johnson M., Roy K.: Design and optimization of low voltage and high performance dual threshold CMOS circuits. Int. Design Automation Conf. (1998)