Yao J, Miwa S, Shimada H *et al.* A fine-grained runtime power/performance optimization method for processors with adaptive pipeline depth. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 26(2): 292–301 Mar. 2011. DOI 10.1007/s11390-011-1132-9

A Fine-Grained Runtime Power/Performance Optimization Method for Processors with Adaptive Pipeline Depth

Jun Yao[1] (姚 骏), *Member*, *IEEE*, Shinobu Miwa[2], Hajime Shimada[1]
and Shinji Tomita[3,4], *Member*, *ACM*, *IEEE*

[1]*Graduate School of Information Science, Nara Institute of Science and Technology, Ikoma 630-0192, Japan*

[2]*School of Engineering, Tokyo University of Agriculture and Technology, Tokyo 183-8538, Japan*

[3]*Graduate School of Informatics, Kyoto University, Kyoto 606-8501, Japan*

[4]*Institute for Integrated Cell-Material Sciences, Kyoto University, Kyoto 606-8501, Japan*

E-mail: {yaojun, shimada}@is.naist.jp; smiwa@cc.tuat.ac.jp; tomita@icems.kyoto-u.ac.jp

Received August 6, 2009; revised November 15, 2010.

Abstract Recently, a method known as pipeline stage unification (PSU) has been proposed to alleviate the increasing energy consumption problem in modern microprocessors. PSU achieves a high energy efficiency by employing a changeable pipeline depth and its working scheme is eligible for a fine control method. In this paper, we propose a dynamic method to study fine-grained program interval behaviors based on some easy-to-get runtime processor metrics. Using this method to determine the proper PSU configurations during the program execution, we are able to achieve an averaged 13.5% energy-delay-product (EDP) reduction for SPEC CPU2000 integer benchmarks, compared to the baseline processor. This value is only 0.14% larger than the theoretically idealized controlling. Our hardware synthesis result indicates that the proposed method can largely decrease the hardware overhead in both area and delay costs, as compared to a previous program study method which is based on working set signatures.

Keywords dynamic optimization, energy saving, fine-grained, pipeline stage unification, workload analysis

1 Introduction

In recent years, power consumption has become a major restriction in the design of modern processors, especially for the portable and mobile platforms where the battery life and thermal problem are dominant considerations. To relieve this performance scaling restriction from the power consumption, many mechanisms have been designed to reduce the certain parts of the CMOS power function $P = \alpha CfV^2 + VI_{\text{leak}}$. As an example, an adaptive depth pipeline (ADP) method changes processor pipeline configurations which accordingly reduces the "αCf" part under a light workload, and thus saves the dynamic part of energy.

Pipeline stage unification (PSU)[1-2] and dynamic pipeline scaling (DPS)[3] are two specific ADP implementations. A detailed study indicated that a pipeline depth reconfiguration in PSU processors will only introduce a nanosecond-level switch penalty, which makes it eligible for a fine-grained energy optimization. To make best use of PSU's fast switch feature, we designed a low-cost method in this paper to categorize program behaviors at a very fine granularity. In this method, the processor throughput, as represented by instructions per cycle (IPC), is studied continuously to predict suitable processor configurations that follow short-term program phase behaviors. The control hardware has been extensively optimized to meet the fine-grained controlling target.

As compared to the static baseline processor execution without an adaptive pipeline depth, the fine-grained IPC-based mechanism achieves a 13.5% EDP reduction, according to the energy-delay-product (EDP) metric[4] which is commonly used for mobile platforms. Meanwhile, this final averaged EDP result is only 0.14% larger than the theoretically idealized execution, in which the processor is instructed from the statistical profiling data. This EDP reduction is slightly better than the predictor using signature history table (SHT) to identify the program phases[5]. A detailed

Regular Paper

This work was originally created in Kyoto University, Japan and then continued in Nara Institute of Science and Technology (NAIST), Japan. This work is partially supported by VLSI Design and Education Center (VDEC), University of Tokyo with the collaboration with Synopsys Corporation.

©2011 Springer Science + Business Media, LLC & Science Press, China

analysis indicates that the fine-grained analyzer has the following major advantages, as compared to the SHT-based method:

1) Better toleration of influences from very large in-program code diversity;

2) Far smaller hardware complexity in implementation. The workload analysis only requires a one-cycle delay by a well-tuned hardware design.

The remainder of the paper is organized as follows. Section 2 describes the PSU framework as the baseline technique and introduces some related work. Section 3 proposes the fine-grained workload study method for the suitable PSU degree prediction and gives its hardware implementation. Simulation methodology is described in Section 4. In Section 5, the prediction accuracies and energy efficiency results are presented. Section 6 concludes the paper.

2 Baseline Technique and Related Work

Currently, dynamic voltage/frequency scaling (DVFS) is commonly used in commercial microprocessors[6] to reduce processor working energy consumption by decreasing the supply voltage V and the frequency f as a sequence under a light workload. However, the voltage change in a modern processor[6] will introduce an execution delay of tens of microseconds, which may be ineligible to apply a very fine-grained energy control.

Alternatively, Shimada *et al.*[1-2] and Koppanalil *et al.*[3] presented a different method, which was expressed as pipeline stage unification (PSU) or dynamic pipeline scaling (DPS). Its main purpose is to reduce the processor's energy consumption via bypassing/inactivating part of pipeline registers and use shallow pipelines under a light workload. As described in [1], a pipeline of 20 stages was employed as the baseline processor, following a similar scheme of current microprocessors. Three PSU degrees were assumed as different pipeline configurations.

1) PSU degree 1 (U_1): the normal mode without bypassing any pipeline registers.

2) PSU degree 2 (U_2): merge every pair of adjacent pipeline stages by bypassing and inactivating the pipeline register between them. The new pipeline has 10 stages.

3) PSU degree 4 (U_4): based on U_2, merge the adjacent stages one step further. The pipeline now contains 5 stages.

Therefore, PSU has provided a pipeline with an adaptive depth. Its shallow modes, as U_2 and U_4, consume less energy than normal mode U_1 from the gated pipeline registers. In addition, [7-8] stated that the variabilities in programs usually drive the optimal pipeline depth to quite different depths when putting the emphasis on both energy and performance. As a solution to the stated problems, PSU's adaptive pipeline depth is expected to fit a larger range of different programs.

A further study of the PSU mechanism reveals that the latency of a PSU degree switch from U_i to U_j can be decomposed into a pipeline flush and a frequency scaling. Researches in [9-10] give designs to reduce the dynamic frequency scaling to zero or nanosecond level. Therefore, the latency of PSU degree switch becomes comparable to a pipeline flush, which can be finished in tens cycles. Compared to the relatively long voltage scaling penalty in DVFS applications[6], PSU is eligible to be scheduled more frequently.

As can be expected, a finer granularity based controller may have efficiency in workload optimization. Specifically, Fig.1 shows the ideal EDP reductions of several different PSU degree control intervals, under our employed environment which will be introduced in Section 4. Four relatively complex benchmarks are used in this figure. It can be observed that a finer granularity tends to have better EDP reduction results. Benchmark bzip2 even displays negative EDP reductions under large intervals.

However, the very small granularity introduces a new restriction on the complexity of the workload analyzer itself. Methods in [11-13] are effective to detect shifts of workload behaviors while their accurate but complex analyzing procedures are expected to be covered by a millisecond-level control interval. Another previously proposed signature history table based workload analyzer[5] will require a period of 50 clock cycles to get the program interval clustering results. When the granularity shrinks to a time slice of less than 100 cycles, the relatively complex method can no longer be applicable. Therefore, to effectively use the fast PSU degree switching feature, a low latency PSU control mechanism is required, which serves as our goal in this research.

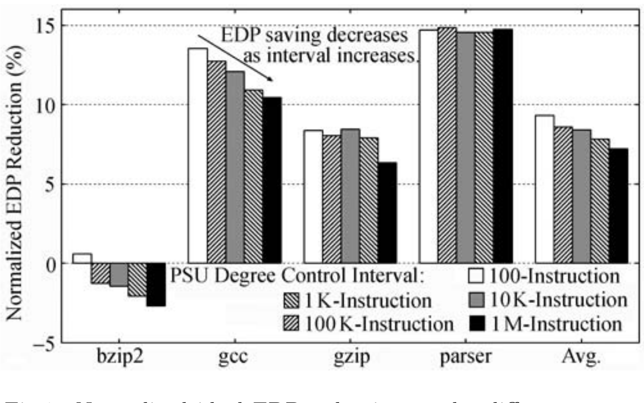

Fig.1. Normalized ideal EDP reductions under different execution intervals.

3 A Fine-Grained PSU Control Method

3.1 Analytical Model

Analyzing workload at a very fine granularity requires a representative but easy-to-get characteristic metric to balance accuracy and overhead. In this subsection, a rough analytical model is derived from previous researchers' work for the selection of a simple program characteristic representation.

Processor energy consumption is affected by both performance and working power. First, we derive the relationship between performance metric of cycles per instruction (CPI) and pipeline depth from the analysis in [7-8, 14]. After abstraction, CPI can be expressed as:

$$CPI(n) = CPI_0 + \beta n, \tag{1}$$

where n is the number of pipeline stages and can be controlled by PSU method. CPI_0 is defined to measure the instruction committing rate of a hazard free pipeline and is thus independent of the pipeline depth. The term "βn" represents the delay introduced by hazards. This hazard delay increases linearly when pipeline takes more stages due to the increased stalled cycles before the hazard is resolved. The parameter β denotes the rate of CPI increasing with n.

To verify (1), Fig.2 gives the CPI results (y-axis) versus the number of three pipeline depths (x-axis) from our practical simulation. The "0-stage" value in Fig.2 is the intersection points between CPI line and the y-axis, which represents CPI_0 in (1). This figure gives a rough vision of β and CPI_0 for different programs.

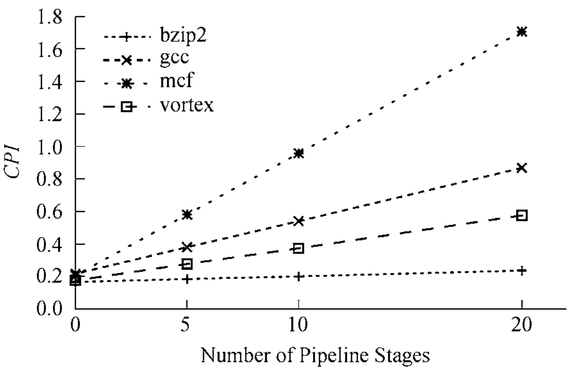

Fig.2. CPIs of different PSU degrees from simulation.

Similar to CPI, the power consumption in PSU processor can also be expressed as a function of the number of pipeline stages. Supposing that each pipeline register has a capacitance of C_p, the total gate capacitance is now $C_o + nC_p$, where C_o is from units other than pipeline registers. We can now extract the power function of $Power = \alpha CfV^2$ into (2). Note that only the dynamic part of power is considered in this model.

$$Power(n) = \alpha(C_o + nC_p)f(n)V^2. \tag{2}$$

where α is the averaged activity factor of processor units. $f(n)$ represents the different clock frequencies under different PSU degrees. We follow the same design as [1] that the frequency scales down linearly as the PSU degree goes from U_1 to U_2 and U_4. Accordingly, $f(n)$ can be calculated as $n \cdot f_{\text{base}}/n_{\text{base}}$, where the "base" subscript indicates the values from the baseline processor.

Thus, combining (1), (2) and the $f(n)$, we can get the analytical EDP value which is an energy efficiency metric[4]. As EDP is calculated as $power \times delay^2$, we can have:

$$\frac{EDP(n)}{EDP(n_{\text{base}})} = \frac{(C_o + nC_p)(CPI_0 + \beta n)^2 \cdot n_{\text{base}}}{(C_o + n_{\text{base}}C_p)(CPI_0 + \beta n_{\text{base}})^2 \cdot n}. \tag{3}$$

Here, C_o, C_p and n_{base} are predetermined values. Shimada *et al.* stated in [15] that C_o takes 70% and each C_p takes 1.5% of the capacitance of a processor with a 20-stage pipeline. CPI_0 and β are workload dependent, as shown in Fig.2. If CPI_0 and β can be predicted, we can estimate the suitable number of pipeline stages, which provides the most EDP reduction from this analytical mode. Note that for simplicity, we assumed α (in (2)) to be identical under the three PSU pipeline depths in the derivation of (3)[①].

Assuming CPI_0 takes the relatively stable value of 0.2 from Fig.2, we can have $EDP(n)/EDP(20)$ values versus β as in Fig.3(a). According to the three lines, applications with different β have their preferred PSU degrees. Specifically, PSU degree U_1 may perform best in bzip2 which has a very small β. When β increases, U_2 and U_4 will outperform U_1 consequently. In gcc and mcf which are more sensitive to the hazards, PSU degree U_4 works the best.

As a verification, the detailed EDP simulation results are presented in Fig.3(b). Only gcc in Fig.3(b) has different choice than Fig.3(a) due to the aggressive clock gating in the practical simulation. However, the relationship between best PSU degree and β has a similar trend as Fig.3(a). We can thus use this relationship in our dynamic control method.

3.2 IPC Based Dynamic PSU Utilization

Based on the theoretical analysis in Subsection 3.1, a

[①]Strictly speaking, α tends to increase when the pipeline goes towards shallow with the application of an aggressive clock gating. Latter simulations will use clock gating in Wattch Tool Set[16] to manage detailed α values.

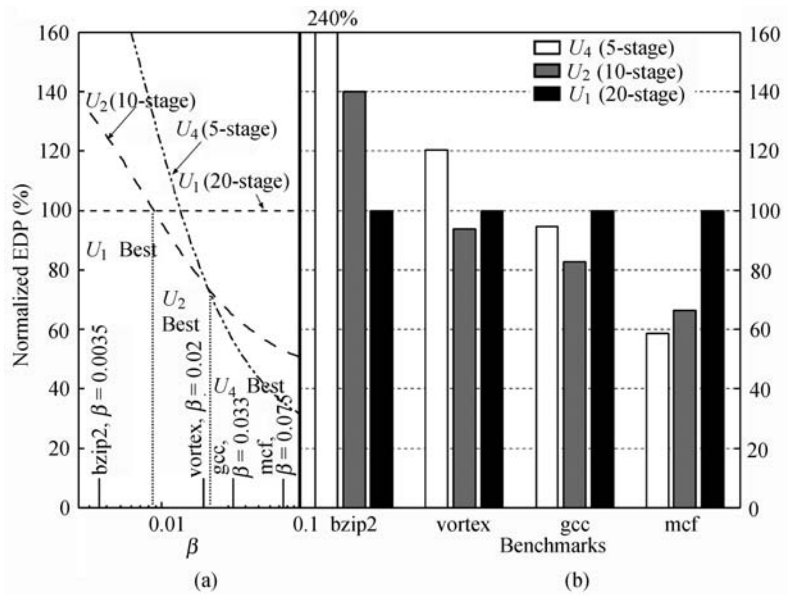

Fig.3. Estimation results of analytical model, together with simulated results. (a) Calculation from (4). (b) Simulation results (with clock gating).

rough relationship can be established between the workload characteristic *CPI* and the suitable pipeline depth configuration. When $\beta = (CPI - CPI_0)/n$ increases, larger PSU degrees — shallow pipelines tend to work better under an EDP consideration.

The values CPI_0 and β are workload specific parameters. However, as observed from Fig.2, values of CPI_0 are comparably similar among all the benchmarks. Thus at each n we can roughly use CPI/n to distinguish the program characteristic value β. Furthermore, a range of $(MIN_{\frac{CPI}{n}}, MAX_{\frac{CPI}{n}})$ that defines the boundary of different program behaviors can actually be replaced by (MIN_{CPI}, MAX_{CPI}) under different n values. Here we introduced some approximation by using *CPI* under a predefined n to represent β. This approximation can save one division operation consequently, which can help decrease the calculation time and apply a fast workload study.

Using these assumptions, we can employ a workload characteristic detection method by checking *CPI* or its reciprocal form, which is known as instructions per cycle (IPC). Since IPC is relatively easy to extract by counting the committed instructions in a certain interval, it will be used as the major characteristic indicator in this research.

Generally, a fine analyzing granularity can help use PSU's fast switching feature to an extend. However, it is also important to make the dynamic analyzing method able to tolerate the skews, especially when the granularity is in the same order of L2 cache miss or even the branch misprediction penalty. In this research, we are trying to use the idea of moving average which is widely used in the stock study field. The prediction scheme is shown in Fig.4.

By using the idea of moving average, after each small interval, the averaged IPC of the last m intervals will be studied, where m is defined to be the window size of recent IPC samples (8 is assumed to be the example of m in Fig.4). With a well selected value of m, the very short term skew of sudden IPC bursts or drops can be concealed while the long or medium term changes can still be reflected in the mean value if there are sufficient accumulation. We predict the new IPC of the next interval will follow this moving average and use the predicted IPC value to determine next PSU degree. By keeping the interval granularity at a small level, we are still able to achieve an effective tracing of workload weight shifting and thus manage possible opportunities to switch the processor configuration.

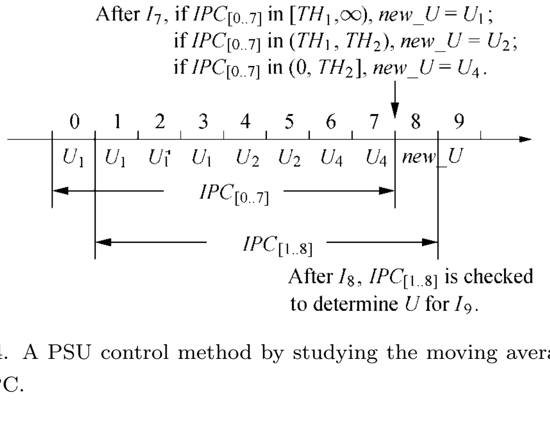
After I_7, if $IPC_{[0..7]}$ in $[TH_1,\infty)$, $new\_U = U_1$;
if $IPC_{[0..7]}$ in (TH_1, TH_2), $new\_U = U_2$;
if $IPC_{[0..7]}$ in $(0, TH_2]$, $new\_U = U_4$.
0 1 2 3 4 5 6 7 8 9
U_1 U_1 U_1 U_1 U_2 U_2 U_4 U_4 new_U
$IPC_{[0..7]}$
$IPC_{[1..8]}$
After I_8, $IPC_{[1..8]}$ is checked to determine U for I_9.

Fig.4. A PSU control method by studying the moving averages of IPC.

3.2.1 Algorithm

We can use the following equation to get the i-th sequential moving average, as MA_i.

$$MA_i = MA_{i-1} - \frac{P_{i-m}}{m} + \frac{P_i}{m}. \tag{4}$$

where P_i represents the value of performance metric at the i-th interval, and m is the size of moving window. (4) can be further simplified by taking out the division part, and now the calculation takes a summary form as in the following equation:

$$SUM_i = SUM_{i-1} - P_{i-m} + P_i. \tag{5}$$

This can be implemented within two simple carry-save adders (CSA), with an m-sized buffer caching the historical P values. After eliminating all divisions, it will introduce less overhead for the workload weight identifying procedure and is possible to satisfy the strict delay requirement in working toward the fine granularity target.

In addition, we might encounter the problem that multiple PSU degrees are used in a single sampling window, as shown in Fig.4. According to the analytical model in Subsection 3.1, although IPC can be a reference for the workload characteristics, it differs under PSU degrees U_1, U_2 and U_4 for a same program interval, since the pipeline structure has been changed after PSU degree switches. A simple conversion between the IPC_{U_x} values must be employed prior to the moving average or summary calculation.

Different to the accurate analytical model in Subsection 3.1, we use approximation from empirical data to speedup the procedure. Table 1 lists the IPC data of different pipeline depths from the first 1.0 billion instructions' execution of our simulated programs. By training these values and proper approximation, we can get two mappings between the PSU degrees U_1, U_2 and U_4, as $IPC_{U_2} = IPC_{U_1} + IPC_{U_1} \gg 1$ and $IPC_{U_2} = IPC_{U_4} - IPC_{U_4} \gg 2$, respectively. Combined with time slices of a fixed length and the corresponding frequency of each PSU degree, the division of "per cycle" can be taken out from these two equations. After simplification, the conversion function can now be expressed in the term of the number of committed instructions, as: $NI_{U_2} = NI_{U_1} - NI_{U_1} \gg 2$, and $NI_{U_2} = NI_{U_4} + NI_{U_4} \gg 1$, where NI stands for the committed instruction number. The value of U_2 is chosen to be the conversion target because the corresponding calculation can be most easily implemented by simple addition and shift.

Table 1. IPC of 1.0 Billion Instructions from Simulation

	U_1	U_2	U_4		U_1	U_2	U_4
bzip2	4.180	4.970	5.413	gcc	1.147	1.847	2.622
gzip	1.273	1.918	2.580	mcf	0.585	1.042	1.715
parser	1.067	1.705	2.389	perl.	1.077	1.754	2.536
vortex	1.725	2.672	3.602	vpr	1.370	2.064	2.729

After these preparations, Fig.5 shows the algorithm we used to determine PSU degree by using the IPC changes in a moving window as the indicator of workload characteristics. The algorithm core takes the committed instruction number per time slice as an input and convert it by defined mappings, which is performed by function `convert_to_U`$_2$`()`. Variable `num_inst_list` refers to the buffer that circularly stores the last m instruction samples. The access is via a pointer named `tail` in Fig.5. The position pointed by `tail` holds the oldest sampling value in the buffer, which is to be subtracted from current moving summary, following (5). The PSU degree is then determined by comparing the summary with two thresholds, which are designed to be the distinction points for heavy, medium and light workloads.

```
After each interval $I_k$:
num_inst_U$_x$ = sim_num_inst - last_sim_num_inst;
last_sim_num_inst = sim_num_inst;
/* num_inst_U$_x$ is the number of committed
 * instructions in interval $I_k$,
 * under PSU degree U$_x$ */
new_num_inst_U$_2$ = convert_to_U$_2$(num_inst_U$_x$,
                              psu_degree);
sum_$m$_interval_insts += new_num_inst_U$_2$;
sum_$m$_interval_insts -= num_inst_list[tail];
num_inst_list[tail] = new_num_inst_U$_2$;
tail = (tail+1)% $m$;
if (sum_$m$_interval_insts > THRESH1)
    psu_degree = UNI1;
else if (sum_$m$_interval_inst < THRESH2)
    new_psu_degree = UNI4;
else
    new_psu_degree = UNI2;
if (new_psu_degree != psu_degree)
    switch_to(new_psu_degree);
```

Fig.5. The dynamic PSU control algorithm by studying the moving summary of instruction numbers.

According to our tuning results, when the two thresholds "TH_1, TH_2" in Fig.4 are set to be around 3.2 and 0.9 in IPC_{U_2} form, the algorithm works the best. Meanwhile, there is a range of good choice for either TH_1 and TH_2. This possible range relieves the method from strict predetermination.

3.2.2 Implementation and Hardware Cost

Fig.6 gives the corresponding hardware

implementation of this PSU control method. The circular buffer is used to cache the historical IPC-related values. Add_1 and Sub_1 implement `convert_to_U2()` in Fig.5. Sub_2 and Add_2 together calculate the new moving summary. The two comparisons after that help give the prediction of the suitable PSU degree for the coming interval. The ranges like "[8:2]" listed besides each unit indicate their bit-widths.

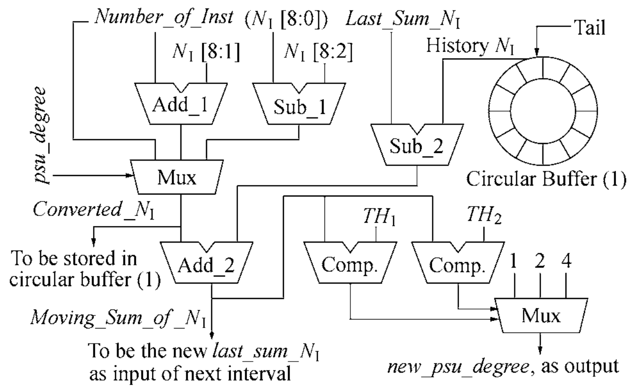

Fig.6. Hardware implementation of the algorithm in Fig.5.

The parameters in Fig.6 are defined under the rule of minimizing hardware cost without sacrificing the efficiency of the design PSU control method. After tuning, the interval is designed to be 64 cycles under PSU degree U_1 and the moving window size (as m in (5)) is set to be 64-entry. These parameters can help set the bit-widths of the hardware units, as shown in Fig.6.

We implemented this hardware design in Verilog HDL (Hardware Description Language) and synthesized it with Rohm $0.18\mu m$ library. The time delay to get the new PSU degree is about 10.7 Fan-Out-of-4 (FO4) inverters' delay, which can be finished within 1 cycle by most modern processors. This 1-cycle delay can be easily covered by normal pipeline hazards and will not impede the processor throughput.

The synthesized result also shows that the area of total logic circuits and buffers is equivalent to 1610 NAND2 gates under the Rohm 0.18μm technology. Compared to modern processors which usually have more than 1 million transistors, the area cost is trivial. Compared to the previously designed signature history table (SHT)-based PSU controller in [5], the extra area cost decreases to 14.1%.

These hardware units implement the core algorithm in Fig.5 to analyze the program behaviors and predict PSU degrees at a very low cost. Compared to those hardware event driven based processor reconfiguration methods, such as the L2 cache misses based dynamic voltage scaling[13,17], the IPC based method tends to fit a comparatively large range of applications, as the IPC has already encapsulated the factors of data dependencies, cache misses and branch mispredictions. Compared to other program phase detecting based methods[5,11,18], this IPC based method exhibits a relatively simple hardware requirement and may thus be triggered at a finer granularity. The simple hardware implementation makes it a good candidate for utilizing the fast switching feature of PSU mechanism.

4 Simulation Methodology

We used a detailed cycle-accurate out-of-order execution simulator, SimpleScalar Tool Set[19] with the Wattch Tool Set[16] to measure the processor energy and performance. The pipeline in these tool sets has been lengthened to 20 stages, following a similar scheme of Intel Pentium 4 platforms. Table 2 lists the configuration information of the baseline processor with a 20-stage pipeline. As described in Section 2, the frequency and delays of the processor units change according to the PSU degree. These values are listed in Table 3, which are similar to Shimada's proposal[1].

Table 2. Baseline Processor Configuration

Processor	8-way out-of-order issue, 128-entry RUU, 64-entry LSQ, 8 int ALU, 4 int mult/div, 8 fp ALU, 4 fp mult/div, 8 memory ports
Branch Prediction	8K-entry gshare, 6-bit history, 2K-entry BTB,16-entry RAS
L1 Icache	64KB/32B line/2way
L1 Dcache	64KB/32B line/2way
L2 Unified Cache	2MB/64B line/4way
Memory	128 cycles first hit, 4 cycles burst interval
TLB	16-entry I-TLB, 32-entry D-TLB, 144 cycles miss latency

We evaluated the dynamic mechanism on eight integers and nine floating-point benchmarks from SPEC CPU2000 suite with the train inputs. In the experiments given in Section 5, 1.5 billion instructions are simulated by fast-forwarding the first 1.0 billion instructions. Note that in latter parts, we mainly use integer benchmarks (bzip2, gcc, gzip, mcf, parser, perlbmk, vortex, vpr) to show results and analysis as they provide more variability. Results of floating-point benchmarks will be listed as supplemental data.

In this paper, we mainly studied the reduction of processor dynamic power via Wattch Tool Set[16]. Since

Table 3. Assumptions of Latencies and Penalty

	PSU Degree		
	U_1	U_2	U_4
	Clock Frequency Rate (%)		
Latencies	100	50	25
Branch Mis-Pred. Resolution	20	10	5
L1 Icache Hit	4	2	1
L1 Dcache Hit	4	2	1
L2 Cache Hit	16	8	4
int Mult	3	2	1
fp ALU	2	1	1
fp Mult	4	2	1
Mem Access (first:burst)	128:4	64:2	32:1
TLB Miss	144	72	36

glitch power in recent years has also drawn more attention as the advancement of process technology, we add the glitch ratio by following the calculation from [8], assuming a same latch factor of 60%. In addition, different from the analytical model introduced in (2), we used the *cc3* style clock gating feature from Wattch[16] to provide the detailed unit activity tracing. In *cc3*, power is scaled linearly with port or unit usage, except that unused units dissipate 10% of their maximum power. The application of clock gating usually leaves little space for other power saving methods, including the widely employed DVFS and the proposed PSU methods[5]. However, our results show that, although the chance of energy reduction in our model is lowered by the application of clock gating, it is not totally eliminated. Detailed results of dynamic PSU control is presented in Section 5.

5 Results and Analysis

5.1 Comparison with Signature Based Method

To examine the efficiency of the proposed fine-grained IPC study based PSU degree predictor, we provide comparisons with the previously proposed signature based PSU control mechanism[5]. The signature history table (SHT) based predictor in that research uses working set fingerprints from [18, 20] to detect the program phases. By using a table structure to store historical signatures and calculating the distance between the signature of current working set with those cached ones[5], the SHT method can detect recurrences of program intervals and can thereby use the suitable PSU degree from the historical tuning information.

Fig.7 tries to give the comparisons between signature-based and fine-grained IPC-based methods. Fig.7(a) shows the program behaviors represented by a quantitative study of working set signatures, following researches in [5, 18]. The vertical axis in Fig.7(a) is the program stability, which is measured as the percentage of all stable regions in the execution. The stable region here is defined as the continuous program period in which the signatures of every two adjacent intervals (of 10 K instructions) are similar. The horizontal axis in Fig.7(a) shows the number of different signatures, which can be used as a measure of the in-program code diversity[18] since the signature is collected by tracing the program counters (PC) of executed instructions. According to these two measures, the 8 integer workloads can be roughly categorized into three groups, as demonstrated in Fig.7(a).

We use prediction accuracies in Fig.7(b) to illustrate the efficiency comparison between the two dynamic

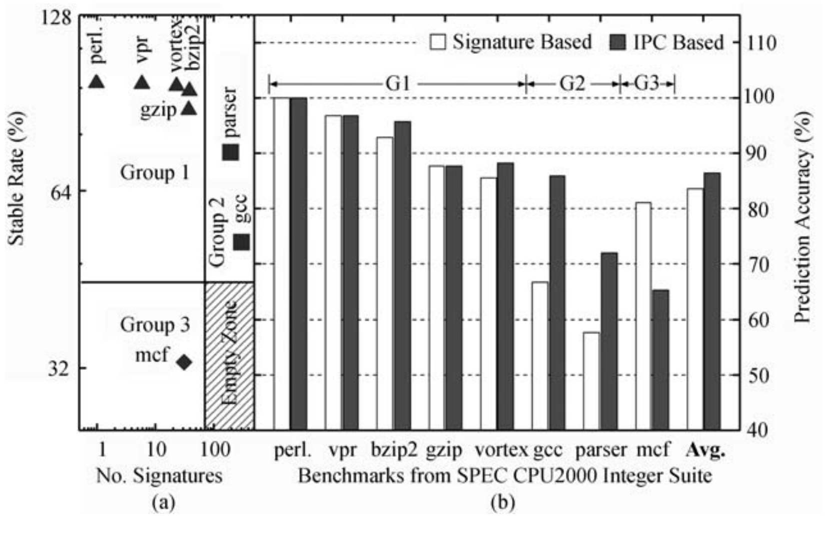

Fig.7. Benchmark grouping and comparison of signature-based and fine-grained IPC-based PSU control methods.

methods. The accuracy rates are obtained by comparing with an idealized predictor, which was similar as introduced in [5]. Based on post-simulation tracing data, the optimal predictor (denoted as ideal predictor in latter parts) has the transcendental recognition of the optimal PSU degree for the coming program block and can thus provides the most efficient processor pipeline reconfiguration prior to the actual execution.

It can be obviously observed from Fig.7(b) that the efficiencies of dynamic control methods vary due to the different characteristics. For benchmarks in Group 1 which illustrate high stabilities and low/medium code diversities, both SHT and fine-grained IPC based methods have good prediction accuracies.

The advantage of the fine-grained IPC-based predictor can be observed with the complex benchmarks in Group 2, compared with the SHT-based predictor. Gcc and parser in this group illustrate high code diversities by experiencing a large number of working set signatures, as Fig.7(a) shows. Therefore, SHT method suffers from the increased misprediction because of the complex relationships among the large number of signatures[5]. However, as the major parts of these two benchmarks are still stable, the fine-grained IPC-based method is less affected because it mainly focuses on the throughput and the changing tendency of a temporarily short-term program period. Global code diversity plays small impacts.

A different circumstance to Group 2 is shown in Group 3, where benchmark mcf demonstrates high instability while maintaining an average level of code diversity as shown in Fig.7(a). The SHT-based method outperforms fine-grained IPC method in this benchmark.

Averagely, the fine-grained IPC-based predictor manages a misprediction rate of 13.5%, in estimating the PSU degree for next small duration of cycles. This misprediction rate is slightly better than our previously designed SHT-based predictor which indicates a mean misprediction rate of 16.5%. From the studies of individual applications, we can have a vision that the two PSU degree predictor can be used in company with different sorts of applications, especially for benchmarks in Groups 2 and 3.

5.2 EDP Savings

The final target of this dynamic processor optimization framework is to achieve better power/performance results by following workload characteristics and properly scheduling PSU pipeline configurations. Fig.8 illustrates the obtained EDP results with our deployed environments. All of these values are the results after the application of clock gating, as introduced in Section 4. In Fig.8, the vertical axis lists the achieved EDP results of each benchmark, as normalized to the EDP of the same benchmark under the baseline execution. The four bars for each individual benchmark correspond to different control methods in a PSU processor, as listed in the figure.

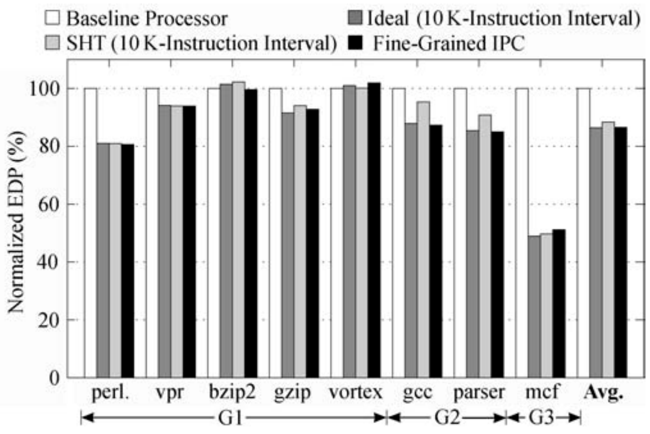

Fig.8. EDP results of dynamic PSU control methods, as normalized to the EDP of baseline processor.

With the comparably high prediction accuracy rates introduced in [5] and Subsection 5.1, the two dynamic PSU control methods can manage the PSU pipeline to achieve EDP results very near to the ideal scenario. The processor with the fine-grained IPC-based PSU degree predictor achieves an averaged 86.5% EDP value, as normalized by the baseline execution. This power/performance result is only 0.14% larger than the ideal execution. Compared to the priorly proposed SHT-based predictor, this fine-grained IPC-based method demonstrates a slight improvement — 2.1% decrease in the average EDP result.

Considering the individual benchmarks with various characteristics, we can find a very similar conclusion as we did in Subsection 5.1 that these two dynamic methods show different sensitivities to the specific program characteristics. Specifically, the EDP savings of fine-grained IPC-based predictor exceed the SHT-based ones in benchmarks from Group 2 because of its insensitivity to code diversities.

Fig.9 lists the normalized EDP results from floating-point benchmarks by using these PSU control methods, under their preferred working parameters. Averagely, the ideal method can reduce around 7.53% EDP as compared to the baseline processor. The two workload analyzers predict PSU degree dynamically, and achieve around 5% EDP reduction. The IPC-based method works slightly better than the previously proposed SHT-based method. However, further studies of the optimized PSU degrees in the ideal method in Fig.9

indicate that 2/3 floating-point benchmarks tend to work well under the baseline processor which uses a deep pipeline. This reduces the opportunities for power saving methods to apply low energy modes. However, necessity of dynamic PSU control can still be observed in benchmarks ammp, equake and swim. For these three benchmarks, both the two dynamic methods can detect major EDP reduction points to achieve high energy efficiencies.

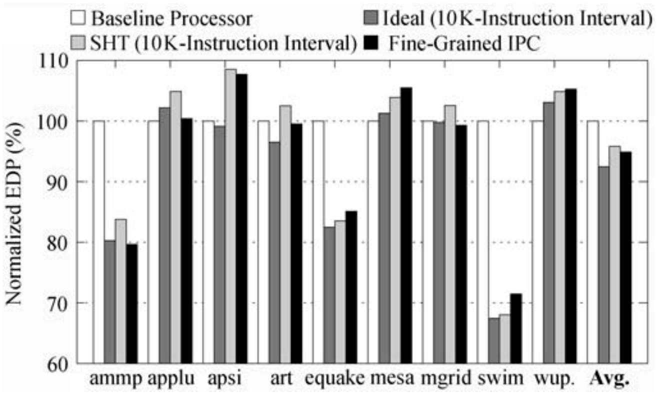

Fig.9. EDP results of floating-point benchmarks, under different PSU control methods.

5.3 Overhead of PSU Degree Switches

As for the switching overhead concerns, Table 4 lists the number of required PSU degree switches for each integer benchmark to achieve the above EDP reductions by the IPC-based method. For most of the benchmarks, this number is not large because of the stability in program behaviors. Only for the most unstable benchmark mcf, PSU degree switch will be required per 1.3 K instructions. Considering the fast switching feature of PSU, which is in the same order of a branch misprediction, we can regard this cost as negligible. Furthermore, two policies can be applied to conceal the switch overhead.

1) For mcf, our detailed study indicates that many of the PSU degree switches, especially from deep to shallow pipeline depth, come together with long pipeline hazards like cache misses or very frequent branch mispredictions. The control method can use the pipeline empty zones in these hazards to trigger a PSU degree switch, as to cover the switching overhead.

2) Modify the PSU logic to allow the degree switching in continuous cycles without waiting for an empty pipeline. Assume that pipeline is divided into zones Z_1, Z_2, Z_3 and Z_4. When instructions propagate toward Z_4, PSU degree switch can sequentially take place in Z_1, Z_2, Z_3 and finally Z_4 by enabling or disabling corresponding pipeline registers.

Table 4. Number of PSU Degree Switches in 1.5×10^9 Instructions

Bench.	No. Switches	Bench.	No. Switches
bzip2	3.39×10^4	gcc	7.78×10^4
gzip	2.43×10^5	mcf	1.14×10^6
parser	2.68×10^5	perl.	2
vortex	1.30×10^5	vpr	1

6 Conclusions

In this paper, we presented a dynamic control mechanism for a processor with pipeline stage unification support. To accompany PSU's very fast switching feature, an IPC-based fine-grained dynamic method is proposed with a well-tuned hardware design. The simulation results demonstrate that this fine-grained method can control the PSU processor to achieve a 13.5% EDP reduction for SPEC CPU2000 integer benchmarks, as compared to the baseline processor. This value is very near to the 100% ideal PSU control method. Compared to a previously designed PSU control method based on workload signature analysis, the fine-grained IPC-based PSU controller can maintain the similar EDP reduction with a largely decreased hardware overhead.

Combining these features, the PSU control method in this paper is considered to be more applicable and can meet the requirement in exploiting PSU's fast configuration switching feature.

References

[1] Shimada H, Ando H, Shimada T. Pipeline stage unification for low-power vonsumption. In *Proc. the 5th International Symposium on Low-Power and High-Speed Chips* (*COOL Chips V*), Tokyo, Japan, Apr. 18-20, 2002, pp.194-200.

[2] Shimada H, Ando H, Shimada T. Power consumption reduction through combining pipeline stage unification and DVS. *IPSJ Transactions on Advanced Computing Systems* (*ACS*), Feb. 2007, 48(3): 75-87. (In Japanese)

[3] Koppanalil J, Ramrakhyani P, Desai S, Vaidyanathan A, Rotenberg E. A case for dynamic pipeline scaling. In *Proc. the 2002 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems*, Grenoble, France, Oct. 8-11, 2002, pp.1-8.

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

[5] Yao J, Miwa S, Shimada H, Tomita S. A dynamic control mechanism for pipeline stage unification by identifying program phases. *IEICE Transactions on Information and Systems*, Apr. 2008, E91-D(4): 1010-1022.

[6] Intel Pentium M processor on 90nm process with 2MB L2 cache datasheet. Intel Corporation, 2006.

[7] Hartstein A, Puzak T R. Optimum power/performance pipeline depth. In *Proc. the 36th Annual IEEE/ACM International Symposium on Microarchitecture*, San Diego, USA, Dec. 3-5, 2003, pp.117-128.

[8] Srinivasan V, Brooks D, Gschwind M, Bose P, Zyuban V,

Strenski P N, Emma P G. Optimizing pipelines for power and performance. In *Proc. the 35th Annual ACM/IEEE International Symposium on Microarchitecture*, Istanbul, Turkey, Nov. 18-22, 2002, pp.333-344.

[9] Geissler S, Appenzeller D, Cohen E, Charlebois S, Kartschoke P, McCormick P, Rohrer N, Salem G, Sandon P, Singer B, Reyn T V, Zimmerman J. A low-power RISC microprocessor using dual PLLs in a 0.13 μm SOI technology with copper interconnect and low-*k* BEOL dielectric. In *Proc. IEEE International Solid-State Circuits Conference*, San Francisco, USA, Feb. 3-7, 2002, Vol.1, pp.148-149.

[10] Senger R M, Marsman E D, Carichner G A, Kubba S, McCorquodale M S, Brown R B. Low-latency, HDL-synthesizable dynamic clock frequency controller with self-referenced hybrid clocking. In *Proc. IEEE International Symposium on Circuits and Systems*, Island of Kos, Greece, May 21-24, 2006, pp.21-24.

[11] Sherwood T, Perelman E, Hamerly G, Sair S, Calder B. Discovering and exploiting program phases. *IEEE Micro*, Nov./Dec. 2003, 23(6): 84-93.

[12] Isci C, Buyuktosunoglu A, Martonosi M. Long-term workload phases: Duration predictions and applications to DVFS. *IEEE Micro*, 2005, 25(5): 39-51.

[13] Isci C, Contreras G, Martonosi M. Live, runtime phase monitoring and prediction on real systems with application to dynamic power management. In *Proc. Micro 2006*, Orlando, USA, 2006, pp.359-370.

[14] Hartstein A, Puzak T R. The optimum pipeline depth for a microprocessor. In *Proc. the 29th Annual International Symposium on Computer Architecture*, Anchorage, May 25-29, USA, 2002, pp.7-13.

[15] Shimada H, Ando H, Shimada T. Pipeline stage unification: A low-energy consumption technique for future mobile processors. In *Proc. the 2003 International Symposium on Low Power Electronics and Design*, Seoul, Korea, Aug. 25-27, 2003, pp.326-329.

[16] Brooks D, Tiwari V, Martonosi M. Wattch: A framework for architectural-level power analysis and optimizations. In *Proc. the 27th Annual International Symposium on Computer Architecture*, Vancouver, Canada, Jun. 10-14, 2000, pp.83-94.

[17] Li H, Cher C Y, Roy K, Vijaykumar T N. Combined circuit and architectural level variable supply-voltage scaling for low power. *IEEE Trans. VLSI Systems*, May 2005, 13(5): 564-576.

[18] Dhodapkar A S, Smith J E. Managing multi-configuration hardware via dynamic working set analysis. In *Proc. the 29th Annual International Symposium on Computer Architecture*, Anchorage, USA, May 25-29, 2002, pp.233-244.

[19] Burger D, Austin T M. The SimpleScalar tool set, Version 2.0. *SIGARCH Computer Architecture News*, 1997, 25(3): pp.13-25.

[20] Dhodapkar A S, Smith J E. Comparing program phase detection techniques. In *Proc. the 36th Annual IEEE/ACM International Symposium on Microarchitecture*, San Diego, USA, Dec. 3-5, 2003, pp.217-227.

Jun Yao was born in 1978 and received his B.E. and M.E. degrees from Tsinghua University, China in 2001 and 2004, respectively. He received his Ph.D. degree in informatics from Kyoto University, Japan in 2009. He has been an assistant professor at Graduate School of Information Science, Nara Institute of Science and Technology, Japan since July, 2009. His research interests are computer architecture and storage area networks. He is currently a member of IPSJ, IEICE and IEEE.

Shinobu Miwa was born in 1977 and received his Ph.D. degree from Kyoto University, Japan in 2007. He has been an assistant professor in the Graduate School of Engineering, Tokyo University of Agriculture and Technology, Japan since 2008. His research interests are computer architecture and neural networks. He is a member of IEICE, IPSJ and JSAI.

Hajime Shimada was born in 1976 and received his B.E., M.E. and D.E. degrees from Nagoya University, Japan in 1998, 2000 and 2004 respectively. He was an assistant professor in Graduate School of Informatics, Kyoto University from 2005 to 2009. He is now an associate professor in Graduate School of Information Science, Nara Institute of Science and Technology, Japan since April, 2009. He is currently focusing on computer architecture related researches. He is a member of IPSJ and IEICE.

Shinji Tomita was born in 1945 in Japan. He received the B.E., M.E. and D.E. degrees from Kyoto University, Japan in 1968, 1970 and 1973 respectively. He was a professor and dean of the Graduate School of Informatics, Kyoto University from 2006 to 2009. He is currently a professor and the chief administrator of Institute for Integrated Cell-Material Sciences, Kyoto University since April, 2009. His major research interests are computer architecture and parallel processing. He is a member of IPSJ, IEICE, ACM and IEEE.