社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－ARC－170 (8)
2006／11／29

An EDP Study on the Optimal Pipeline Depth for Pipeline Stage Unification Adoption

JUN YAO,† HAJIME SHIMADA,† YASUHIKO NAKASHIMA,‡
SHIN-ICHRO MORI* and SHINJI TOMITA†

To find optimal pipeline design point by considering both performance and power objectives has been one hot spot in recent researches. However, we found that previous papers did not consider deepening or shrinking pipeline depth dynamically during program execution. In this paper, with the adoption of the previously proposed Pipeline Stage Unification (PSU) method, we studied the relationship between power/performance and pipeline depth in processors with a pipeline of multi-usable depths. Our evaluation results of SPECint2000 benchmarks show that the 24-stage PSU enabled pipeline can achieve the smallest Energy-Delay-Product (EDP). Moreover, it can obtain more EDP reduction, compared to the processors with fixed 12-stage pipeline, which is the optimal pipeline depth among fixed depth pipelines.

1. Introduction

In the recent years, increasing the clock frequency has provided most part of the microprocessor performance improvements. With a given technology, the effective way to increase the frequency is to make deeper pipelines, i.e., to contain fewer gates in each clock period. M. S. Hrishikesh, et al.[1] has discussed that the optimal logic depth per pipeline stage is 6 to 8 fan-out-of-four (FO4) inverter delays for integer benchmarks from SPEC 2000, in order to achieve the optimum performance.

As the consideration of power dissipation becomes more and more important in modern microprocessor design, a performance-only objective will be less competitive for processors in which the thermal dissipation or the battery life is the dominant problem, such as the mobile phones and laptops. Several researches have been subjected to revealing the relationship between the pipeline depth and the power/performance metrics[2),3)]. A. Hartstein's study[3)] show that the design point to obtain optimum Energy-Delay-Delay-Product (EDDP) occurs at an 8-stage (20 FO4 per each stage) pipeline design point, averaged over all of the 55 workloads he studied. In that paper, he also proposed a theoretical method to find the trade-off between power and performance.

However, all of the above studies were assuming a fixed pipeline depth during the program execution. Our studies in this paper show that the characteristics of individual program will cause the optimal pipeline depth to occur at quite different design points. Thus, using a single fixed pipeline depth will show efficiency in certain programs, while inevitably experiencing some penalty in other programs with different behaviors. Moreover, even for a single program, our results show that various runtime periods require different optimal pipeline depths due to the changes in program characteristics through the whole execution.

There are some studies on dynamically changing the pipeline design during the program execution in recent years. Shimada et al.[4),11),12)] and Koppanalil et al.[5)] have presented us a method to reduce the processor power consumption via in-activating and bypassing some of the pipeline registers and using a shallow pipeline during the program execution, which is called Pipeline Stage Unification (PSU). According to this characteristics, PSU can be regarded as a dynamic method to change the pipeline depth during runtime. With the adoption of this method, we may have different findings in the study of optimal pipeline depth since PSU method can adapt the pipeline depth to the program characteristics.

Our research described in this paper is focusing on revealing the relationship between the pipeline depth to both the processor power consumption and the performance under PSU adoption. We used the Energy-Delay-Product (EDP) defined by Gonzalez[6)] to serve as the power/performance metric because it can mostly fit for our target platforms which include light-weighted workstations and laptops.

From our study of SPECint2000 benchmarks, we found that the average optimal depth for a fixed depth pipeline is 12 stages, with 14.2 FO4 per each pipeline stage. With the PSU utilization, an n-stage pipeline can change to an n/2-stage or an n/4-stage pipeline dynamically. We can achieve smaller EDP in PSU enabled deep pipelines with more than 16 stages. Among them, we found that the 24-stage PSU enabled pipeline is the most efficient one in reducing EDP. Our results also show that compared with the optimal 12-stage pipeline among fixed depth pipelines, the deep 24-stage pipeline with PSU enabling can achieve about 9% more EDP reduction, averaged from 8 SPECint2000 benchmarks. Even after complex clock gating which lowers opportunities for other energy saving technologies, this 24/12/6-stage pipeline design can still save 6.4% more EDP than the 12-stage fixed depth pipeline.

The rest of the paper is organized as follows. Section 2 describes the main idea of this paper. Sim-

† Graduate School of Informatics, Kyoto University
‡ Graduate School of Information Science, Nara Institute of Science and Technology
* Graduate School of Engineering, Fukui University

ulation methodology to evaluate EDP of different pipeline depths can be found in Section 3. In Section 4 we show the experiment results, together with some analyses. Section 5 concludes the paper.

2. EDP vs. pipeline depth

2.1 Basic processor EDP model

Considering a given technology, the clocking frequency varies due to the changes in the pipeline design. Such a relationship can be expressed as follows:

$$f = \frac{1}{t_o + t_p/n} \quad (1)$$

It gives that the frequency (f) is a function of these parameters: the number of pipeline stages (n), the total logic delay of pipeline (t_p), and the latch overhead per pipeline stage (t_o). t_p and t_o are usually expressed in the number of FO4 inverter delays. From equation (1), we can see that with the increase of n, data passes through fewer logic units in one cycle and the clock frequency increases consequently.

Performance is usually presented as delay. It is quite sensitive to the frequency, as shown in the following formula.

$$D(n) = \frac{N_c}{f} = N_c \times (t_o + t_p/n) \quad (2)$$

The parameters are the same with equation (1) except that N_c represents the number of cycles required in executing the program.

Power can be expressed as follows:

$$P(n) = \alpha C_{total} f V^2 = \frac{\alpha(nC_{latch} + C_o)V^2}{t_o + t_p/n} \quad (3)$$

This is the dynamic power part of the total processor power, where α represents the average activity, C_{total} refers to the total capacity, f denotes the clock frequency, and V serves as the supply voltage. Since both C_{total} and frequency vary due to pipeline design, we can extract the formula a little further to take the influence of stage number into account. The latter part of equation (3) is the extracted result. C_{latch} is the latch capacity per pipeline stage and C_o is the capacity of other processor units.

Note that this equation is different from the previous power estimation in paper 3) in two aspects:

(1) The power equation in paper 3) claims that the majority power consumption is in pipeline latches. We added other parts power consumption together, including register files, cache, and so on. These units have a total capacity represented by C_o in equation (3).

(2) For simplicity, only dynamic power is considered in this paper, although the pipeline stage unification (PSU) method, which we used in our research, can also decrease leakage power by applying supply voltage gating to disabled pipeline registers. Besides, to reduce the dynamic power is also the main target of current DVS like technologies which scale down supply voltage under low workload.

Energy-Delay-Product (EDP) can be expressed as follows:

$$EDP(n) = P(n) \times D(n) \times D(n) \quad (4)$$

To find the optimal depth, we need to take the derivative of this EDP equation with respect to n. It is a mathematically lengthy problem. Moreover, some parameters like N_c and α in equation (2) and (3) are still uncertain, and will vary during the program execution. For these reasons, we use simulation results to find the optimal number of stages in this paper.

2.2 Employing PSU: a changeable pipeline depth

As described in paper 2), 3), several researches have been carried out to find the optimum pipeline depth for power/performance consideration with a similar model in Section 2.1. However, although they compared the energy metrics of different pipeline depths, they were using a fixed pipeline depth during a whole execution. This means that they eliminated the differences between different programs and different periods. And their proposed optimal depth was averaged among a studied workload set. As we discuss in Section 4, we found it almost impossible to define a certain optimal depth for all the benchmarks. This inspired us to launch our study based on a changeable pipeline depth.

In our research we employed the previously proposed design by Shimada[4] for this purpose. In his paper, Shimada proposed an energy consumption reduction method called PSU to reduce the power consumption in mobile processors. Other than its original purpose to reduce power consumption by bypassing the pipeline registers, PSU is rather a pipeline reconfiguration method. As shown in paper 4), pipeline depth under PSU is called unification degree. We assumed following three unification degrees, as defined in paper 4).

(1) Unification Degree 1 (U1): The normal mode without bypassing any pipeline registers. It has a pipeline of n stages.

(2) Unification Degree 2 (U2): Merge every pair of adjacent pipeline stages by in-activating and bypassing the pipeline register between them. It has a pipeline of $n/2$ stages.

(3) Unification Degree 3 (U4): Based on U2, merge the adjacent stages one step further. It has a pipeline of $n/4$ stages.

According to the PSU proposal, if we start from an n-stage pipeline, we can choose one suitable pipeline design point from n, $n/2$, and $n/4$ stages. As we are using EDP to be the power/performance metric, we need to find a best one from $EDP(n)$, $EDP(\frac{n}{2})$, and $EDP(\frac{n}{4})$ based on the history information, and use the corresponding unification degree for next bulk of instructions, which is shown in our previous research 7). In the latter part of this paper, we will show the relationship between the EDP and a pipeline with multi-usable depths.

3. Simulation methodology

To study the effect of different pipeline depths on power/performance, we varied the pipeline depth of a modern superscalar architecture similar to current processors, which have relatively deep pipelines.

This section describes our simulation framework and the methodology we used to perform this study.

We used a detailed cycle-accurate out-of-order execution simulator SimpleScalar Tool Set[8] to collect the runtime performance information. We ran 8 integer benchmarks (bzip2, gcc, gzip, mcf, parser, perlbmk, vortex, and vpr) from SPEC2000, with train inputs. 1.5 billion instructions are simulated after skipping the first billion instructions.

3.1 EDP estimation methods

Based on this simulator, we have used two ways to estimate the runtime power dissipation and the EDP in processors:

(1) Simple estimation

By extracting equation (4) we can get the following formula to calculate the energy consumption in processor.

$$EDP(n) = \alpha(nC_{latch}+C_o)V^2{N_c}^2(t_o+t_p/n) \quad (5)$$

Each parameter is the same as we have already defined in section 2.

In defining the detailed value for each parameter, we select the processor with 20-stage fixed depth pipeline which is similar as the Pentium 4 architecture, to serve as the baseline processor. For simplicity, we assume that α will keep as a constant for different pipeline designs and for different programs intervals during runtime. And we assumed that the latches in pipeline approximately consume 30% of the total processor power in a 20-stage pipeline. This value is same with previous paper 4). From this assumption, we can get the latch capacity for each stage (C_{latch}) to be about 1.5% of total capacity, with C_o providing the left 70% of total capacity. t_p and t_o are set to be 140 FO4 and 2.5 FO4 respectively, as described in paper 3).

After simplification, equation (5) shows that EDP directly depends on the pipeline design and the performance.

(2) Modified Wattch toolset

The previous method is a bit rough since we simply assume the activity α of each breakdown processor unit to keep unchanged during the runtime. In modern processors, this activity will vary a lot for different programs during different runtime periods because of the widely used clock gating. Wattch toolset[9] has provided several ways to estimate runtime energy with clock gating utilization. We use *cc3* method in Wattch toolset to estimate this power in clock gating applied processors. In this clock gating method *cc3*, power is scaled linearly with port or unit usage, except that unused units dissipate 10% of their maximum power. The factor 10% exists because it is impossible to turn off a unit totally when it is not needed, in the practical circuits.

Since Wattch 1.02 used a fixed traditional 8 stage pipeline, we modified it to adopt deeper pipelines, such as 20 stages. Furthermore, because of how Wattch provides the dissipated power, it is not easy to determine the breakdown power consumed in pipeline latches, since it do not explicitly have a part related to pipeline registers. Thus we made an approximate assumption that the power consumed in pipeline registers is a proportion to the total clock power, and this proportion varies linearly according to the change of pipeline depth.

Table 1 Processor configuration

Processor	4-way out-of-order issue, 128-entry RUU, 64-entry LSQ, 4 int ALU, 2 int mult/div, 4 fp ALU, 2 fp mult/div, 4 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/4-way
Memory	64 cycles first hit, 2 cycles burst interval
TLB	16-entry I-TLB, 32-entry D-TLB, 128 cycles miss latency

Table 2 Some Detailed Processor Parameters

n	$t_o + t_p/n$	Freq.	IL1	DL1	L2	ALU	MP.
4	37.5	0.5067	1	1	4	1	4
8	20	0.95	2	2	7	2	8
12	14.2	1.3412	2	2	10	2	12
16	11.3	1.6889	3	3	13	3	16
20	9.5	2.0	4	4	16	3	20
24	8.33	2.28	4	4	20	4	24
28	7.5	2.533	5	5	23	4	28
32	6.88	2.7636	5	5	26	5	32
36	6.34	2.9739	6	6	29	5	36
40	6	3.1667	7	7	32	6	40

3.2 Processor parameters

Table 1 shows the baseline processor configuration.

Other than the configuration showed in table 1, we need to define other parameters which change according to the modification of pipeline depth.

Assume that the baseline 20-stage pipeline has a clock frequency of 2GHz, the frequency can be calculated as follows:

$$f(n) = \frac{t_o + t_p/20}{t_o + t_p/n} \times 2GHz \quad (6)$$

We have already defined t_p as 140 FO4 and t_o to be 2.5 FO4. It is easy to get frequency for each pipeline design.

Branch misprediction penalty also varies as the pipeline depth changes. We assume that it changes linearly to the number of pipeline stages.

We used Cacti 3.0 toolset[10] to calculate the cache latency of the baseline processor configuration. For example, the L1 data cache of 90um technology in the baseline processor will have a 21.14-FO4 access latency. Divided by the logic depth (latch overhead excluded), we can get the corresponding access latency in cycles. Similarly, this calculation also applies to the L2 cache latency. In this way, we derived latencies used in the simulation.

We assumed and evaluated different pipeline depths, from 4 to 40 stages. Introducing all of them is too redundant so that we only show some of the configuration in table 2. The notations in table 2 list as follows:

- n: number of pipeline stages;

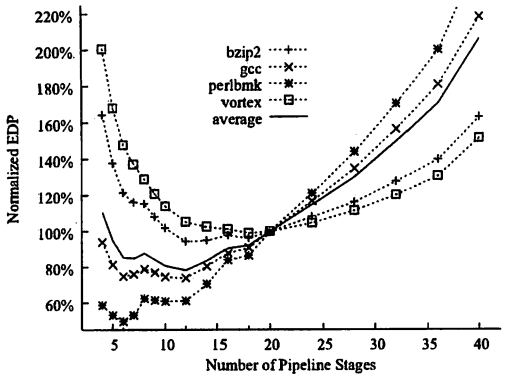

Fig. 1 Normalized EDP of fixed pipeline depth

- $t_o + t_p/n$: inverter delay per stage (in FO4);
- Freq.: Clock frequency (in GHz);
- IL1: L1 instruction cache latency (in cycles);
- DL1: L1 data cache latency (in cycles);
- L2: L2 cache latency (in cycles);
- ALU: integer ALU latency (in cycles);
- MP.: branch misprediction penalty (in cycles).

In our simulation, integer ALU latency in table 2 applies only for the calculation of dependent effective address. We assumed that the processor can issue other dependent integer executions in every cycle by using an aggressive bypassing network.

Other parameters, such as integer MULT and floating point ALU/MULT, make relatively small influences to IPC with the changes of pipeline depth, according to paper 13).

4. Results and analyses

In this section, firstly, we will present the EDP results of fixed pipeline depth, with the pipeline varies from 4 stages to 40 stages. Then, we will show the results of pipeline with changeable depth, from a PSU enabled 16-stage pipeline to a 40-stage pipeline.

4.1 Results of fixed pipeline depth

In this subsection, we follow the previous researchers' work[2),3)] under our experiments environments. It is also precedent work for the evaluations of PSU applied pipelines.

Figure 1 shows the EDP results of pipelines from 4 stages to 40 stages. The x-axis denotes the number of pipeline stages and the y-axis denotes the normalized EDP results. The energy in this series of experiments is given by the simple estimation method, described in Section 3.1. For each benchmark, all of the EDP results "$EDP(n)$" are normalized by the baseline 20-stage pipeline result "$EDP(20)$" of the same benchmark. Therefore, these curves are connected at the 20-stage point.

In this figure, the solid line represents the average values of all 8 SPECint2000 benchmarks and the dashed line is the EDPs of individual benchmark. For simplicity, we only list the results of bzip2, gcc, perlbmk, and vortex here. Other benchmarks like gzip, mcf, parser, and vpr show a similar shape to one of the benchmark drawn in figure 1.

Figure 1 clearly shows that optimal EDP result occurs at the 12-stage pipeline design point, averaged from all the 8 integer benchmarks. At this point, each pipeline stage will have 14.2 FO4 inverter delays, including 2.5 FO4 latch overhead. With the 12-stage pipeline, we can achieve an EDP reduction of 21.4%, compared with a 20-stage design, averagely for the 8 integer benchmarks.

Besides the optimal depth finding, figure 1 also shows that different benchmarks do show different behaviors at the same design point. We can see that for benchmark bzip2 and gcc, the optimal EDP can be obtained by a 12-stage pipeline, which is the same as the average line gives. For the benchmark vortex, the pipeline with 18 stages has the smallest EDP result. It is a deeper pipeline with 10.3 FO4 per each stage. For benchmark perlbmk, the optimal EDP occurs in the 6-stage pipeline design, which is a relatively shallow pipeline. Also, considering the benchmarks that are not listed in this figure, the optimal design should be 12-stage for mcf, parser, and vpr. And it should be 6-stage for gzip. Therefore, using 12-stage design to be the optimal value means that we must experience less EDP reduction in benchmarks like gzip, perlbmk, and vortex, 3 out of all 8 benchmarks.

4.2 Results of ideal PSU method

In previous subsection, we studied the optimum pipeline design point among several fixed depth pipelines by considering EDP metric. In this subsection, we will pursue our research on pipelines with PSU utilization.

From the proposal of PSU, we can see an n-stage pipeline (at U1 mode) can dynamically change to an n/2-stage pipeline under U2 mode, or to an n/4-stage pipeline unde U4 mode during the program execution. Here, U1, U2, and U4 represents the unification degree 1, 2, and 4, respectively, as we have defined in section 2.2. As shown in previous research 7), program may experience different phases during the whole execution. If PSU selects suitable unification degree for each phase, it can show better efficiency in reducing EDP compared to fixed length pipelines.

To simplify our studies and focus on the influence introduced by the pipeline design point only, in this paper, we chose the ideal PSU adoption method described in previous paper 7). For example, we apply this method on a 20-stage pipeline. Firstly, we divided program into fixed length intervals (in simulation, 10k instructions per each interval). Then we ran the program for three times, in fixed 20-stage mode, fixed 10-stage mode, and fixed 5-stage mode, similarly as we did in Section 4.1. And we recorded the EDP of each interval during the execution. After the three executions, we collected the EDP data of each interval of all U1, U2, and U4 mode for the 20-stage pipeline. After that, in the ideal PSU adoption mode, we could set the best unification degree at the beginning of an interval based on the profiling data. This method is a theoretical optimal one and

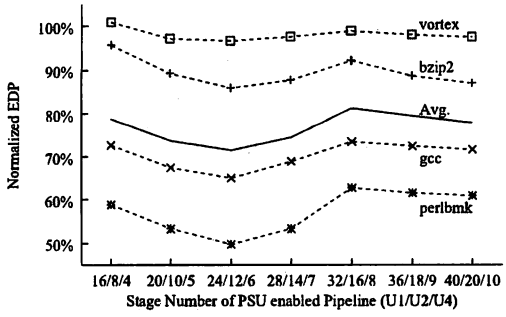

Fig. 2 Normalized EDP of PSU enabled pipelines

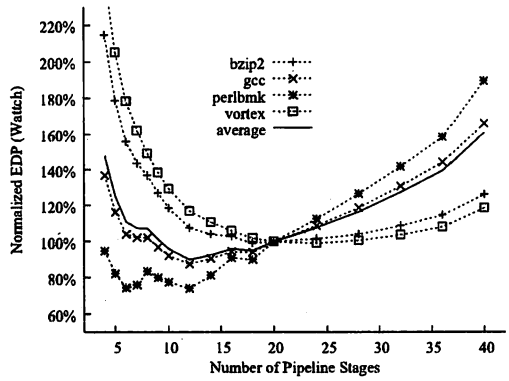

Fig. 3 Normalized EDP of fixed pipeline depth (Wattch)

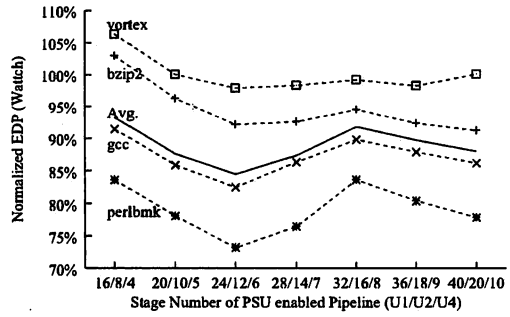

Fig. 4 Normalized EDP of PSU enabled pipeline (Wattch)

can not be achieved in real execution because it is based on the post-simulated trace analysis.

Figure 2 shows the normalized EDP of PSU enabled pipelines. The EDP is calculated by the simple estimation method. Its x-axis denotes the different pipeline configuration, from 16-stage to 40-stage pipelines with ideal PSU adoption. The notation like "20/10/5" represents a PSU enabled pipeline with 20 stages in U1, 10 stages in U2, and 5 stages in U4 mode. We will use such notation in the latter parts of the paper. The y-axis denotes the normalized EDP results. The Avg. line represents the values averaged from all the 8 SPECint2000 benchmarks. 4 lines of individual benchmark are also listed to show the influences by different program characteristics. The EDP results "$EDP(n/\frac{n}{2}/\frac{n}{4})$" of each benchmark are normalized by the 20-stage fixed depth result "$EDP(20)$" of the same benchmark.

From figure 2 we can see that the optimal pipeline depth occurs at a 24-stage pipeline design point after ideal PSU adoption, averaged from 8 benchmarks. At this design point, each pipeline stage has a depth of 8.3 FO4 inverter delays, including 2.5 FO4 latch overhead. Furthermore, this design point is also the best one for the individual benchmark consideration, i.e., all of the 8 benchmarks experience the smallest EDP under the 24/12/6 pipeline configuration with ideal PSU adoption. As shown in figure 2, the lines of individual benchmark demonstrate a quite similar shape as the Avg. line, except that the EDP reduction differs due to benchmark characteristics. This observation indicates that the PSU applied pipeline can greatly alleviate the influence introduced by program behavior. And a certain design point with well-designed dynamic PSU adoption may be able to fit for a large range of programs.

Another observation from figure 2 is that the deviations between different pipeline configurations are not large. For most of the benchmarks, the deviations are approximately within 10% from the 16/8/4-stage pipeline to the 40/20/10-stage pipeline. By applying PSU dynamically, the processor can avoid "too bad pipeline configuration" efficiently so that the worst EDP in figure 2 is far better than that of figure 1.

Also, figure 2 shows that after applying PSU, we can even obtain more EDP reduction compared to the best fixed pipeline depth in figure 1. In figure 1, the optimal depth occurs at a 12-stage pipeline, considering all benchmarks averagely. And the EDP result is 78.6% of the fixed depth 20-stage pipeline one. Yet the 24/12/6 PSU applied pipeline in figure 2 can get 71.6% of $EDP(20)$, which saves about 9% more EDP, compared with the fixed depth 12-stage pipeline value. Individually, for gcc, the 24/12/6 stage can achieve about 12% more EDP reduction than the fixed depth 12-stage design point. This additional reduction is obtained by executing 32% instructions in the 24-stage, 29% instructions in the 12-stage, and 39% instructions in the 6-stage pipeline. This observation indicates that even for a single program execution, different periods will require different pipeline configuration to achieve the best power/performance result. By using PSU, dynamically predicting and adopting a best pipeline depth is necessary and may probably show more efficiency in the modern processors, if frequency goes higher.

4.3 EDP results after applying clock gating

In this section, we will show the EDP results after using Wattch to provide a sophisticated clock gating. We did the same two series of evaluations, one for fixed depth pipeline during a program execution (figure 3), the other for pipelines with PSU utilization (figure 4). The notations, x-axis, and

Table 3 n_{opt} for benchmarks (Wattch *cc3* applied)

n_{opt}	benchmarks
12	gcc, gzip2, mcf, parser, perlbmk, vpr
18	bzip2
24	vortex

y-axis are similar to the ones in figure 1 and 2, except that the EDP results are collected by Wattch toolset.

The lines in figure 3 and figure 4 show a similar shape as the ones in figure 1 and 2, respectively. Yet the EDP reduction becomes smaller after applying clock gating. This is because that clock gating can decrease the total processor energy greatly so that there is little space left for other energy saving technologies.

And for the fixed pipeline depth evaluations, the results show that the best design point moves to the deeper pipeline for some benchmarks. The optimal fixed depths for these benchmarks are shown in table 3. Still, the averagely optimal fixed 12-stage pipeline does not fit for all 8 benchmarks.

Figure 4 is the results after applying ideal PSU. We can get a similar conclusion from this figure as we did with figure 2. Even with the clock gating, the 24/12/6-stage pipeline can achieve 15.5% EDP reduction, compared with $EDP(20)$ of the baseline 20-stage fixed depth pipeline. Compared with the 12-stage pipeline which is the optimal design for a fixed depth pipeline, the 24/12/6-stage pipeline can achieve 6.4% more EDP reduction, averaged from 8 benchmarks. This indicates that the PSU utilization will have efficiency even with a sophisticated clocking gating application.

5. Conclusions and future works

In this paper, we have studied relationship between EDP and pipeline depth for pipelines with fixed depth and ideal PSU utilization. Our results show that averagely, the optimal depth will be 12 stages for a fixed depth pipeline. But the optimal pipeline depth differs for individual programs and runtime periods, and the difference is hardly negligible. By applying ideal PSU method, we find that more EDP reduction can be achieved by deeper pipeline design. The EDP differences are with 10% between the deep pipelines, after PSU utilization. Among them, a 24/12/6 pipeline design can obtain the smallest EDP, either with simple estimation or with complex clock gating application. And it saves about 9% more EDP than the fixed 12-stage pipeline, which is the optimal one for a fixed depth pipeline design. Although the complex clock gating provided in Wattch will lower the EDP savings by this method, the effectiveness is not totally eliminated. After clock gating, this 24/12/6-stage pipeline design can still save 6.4% more EDP than the optimal 12-stage fixed depth pipeline.

Currently, only dynamic power is considered in our study. Yet related studies show that PSU can also eliminate some of leakage power by applying supply voltage gating to disable pipeline registers. Since the leakage power occupies larger and larger percentage of the total power in current high clock frequency processors, to study the optimum pipeline depth selection with taking leakage power into account will be one of our future works.

Acknowledgments This research is partially supported by Grant-in-Aid for Fundamental Scientific Research (S) #16100001 from Ministry of Education, Culture, Sports, Science and Technology Japan.

References

1) Hrishikesh, M. S., Jouppi, N. P., Farkas, K. I., Burger, D., Keckler, S. W., and Shivakumarr, P.: The Optimal Logic Depth Per Pipeline Stage is 6 to 8 FO4 Inverter Delays, *ISCA 29*, Alaska, U. S., pp. 14-24 (2002).

2) Srinivasan, V., Brooks, D., Gschwind, M., Bose, P., Zyuban, V., Strenski, P. N. , and Emma, P.G.: Optimizing Pipelines for Power and Performance, *MICRO 35*, Istanbul, Turkey, pp. 333-344 (2002).

3) Hartstein, A., and Puzak, T. R.: Optimum Power/Performance Pipeline Depth, *MICRO 36*, San Diego, U. S., pp. 117-128 (2003).

4) Shimada, H., Ando, H. and Shimada, T.: Pipeline Stage Unification: A Low-Energy Consumption Technique for Future Mobile Processors, *ISLPED 2003*, Seoul, Korea, pp. 326-329 (2003).

5) Koppanalil, J., Ramrakhyani, P., Desai, S., Vaidyanathan, A. and Rotenberg,E.: A Case for Dynamic Pipeline Scaling, *CASES 2002*, Grenoble, France, pp. 1-8 (2002).

6) Gonzalez, R. and Horowitz, M.: Energy Dissipation in General Purpose Microprocessors, *IEEE JSSC*, Vol. 31, No. 9, pp. 1277-1284 (1996).

7) Yao, J., Shimada, H., Nakashima, Y., Mori, S., and Tomita, S.: Program Phase Detection Based Dynamic Control Mechanisms for Pipeline Stage Unification Adoption, *Workshop on ALPS 2006*, Cairns, Australia, pp. 39-46 (2006).

8) Burger, D. and Austin, T. M.: The SimpleScalar Tool Set, version 2.0. *Technical Report*, CS-TR-97-1342, Univ. of Wisconsin-Madison Computer Sciences Dept. (1997).

9) Brooks, D. Brooks, Tiwari, V., and Martonosi, M.: Wattch: A Framework for Architectural-Level Power Analysis and Optimizations, *ISCA 27*, Vancouver, Canada, pp. 83-94 (2000).

10) Shivakuma, P. and Jouppi, N. P.: Cacti 3.0: An Integrated Cache Timing, Power and Area Model. *Technical Report 2001/2*, Compaq Computer Corporation, (2001)

11) Shimada, H., Ando, H. and Shimada T.: Pipeline with Variable Depth for Low Power Consumption (in Japanese), *IPSJ Technical Report*, 2001-ARC-145, pp. 57-62 (2001).

12) Shimada, H., Ando, H. and Shimada, T.: Pipeline Stage Unification for Low-Power Consumption, *Cool Chips V*, Tokyo, Japan, pp. 194-200, (2002).

13) Shimada, H., Ando, H., and Shimada, T.: Reducing Processor Energy Consumption with Pipeline Stage Unification (in Japanese), *IPSJ ACS*, Vol. 45, No. SIG 1(ACS 4), pp. 18-30, (2004).