Optimal Pipeline Depth with Pipeline Stage Unification Adoption

Jun Yao, Hajime Shimada, and Shinji Tomita
Graduate School of Informatics,
Kyoto University, Kyoto 606-8501, Japan
{yaojun,shimada,tomita}@lab3.kuis.kyoto-u.ac.jp

Shinobu Miwa
Graduate School of Law,
Kyoto University, Kyoto 606-8501, Japan
miwa@lab3.kuis.kyoto-u.ac.jp

***Abstract*—To find the optimal pipeline design point by considering both performance and power objectives has been one focus of interest in recent researches. However, we found that previous papers did not consider deepening or shrinking pipeline depth dynamically during the program execution. In this paper, with the adoption of the earlier 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 shown in this paper illustrate that the PSU adoption can achieve good efficiency for platforms which concern both energy and performance, even after the utilization of complex clock gating.**

I. 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 the 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 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 studied workloads.

However, all of the researches above 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 efficiencies in certain programs, while inevitably experiencing some penalties in other programs with different behaviors. In addition, even for a single program, our results show that various runtime periods require different optimal pipeline depths due to the changes in program characteristics along the whole execution.

There have been some researches on dynamically changing the pipeline design during the program execution in recent years. Shimada et al.[4] 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 thus constructing a shallow pipeline for the microprocessor. This method was called Pipeline Stage Unification (PSU) in Shimada's research. Since the application of PSU actually provides a pipeline with multi-usable depths, we are able to adapt the pipeline depth to the program characteristics, in order to achieve better performance or less power consumption. Therefore, with the adoption of the PSU method, we may have different findings in the study of optimal pipeline depth, as compared to paper [3].

In this paper, we are focusing on the study of optimal pipeline design in PSU enabled platforms which concern both energy and performance. To study the efficiency of our proposed PSU mechanisms, we used power/performance metrics such as PDP, EDP, and EDDP, as defined by Gonzalez [6].

Our study of SPEC CPU2000 integer benchmarks demonstrates that for metric EDP, the average optimal depth for a fixed-depth pipeline is 12 stages, with 14.2 FO4 per each pipeline stage. By applying PSU, more EDP reduction is obtained in the deep pipelines with more than 16 stages. Among them, the 24-stage PSU enabled pipeline is the most efficient one in reducing EDP. Compared to the optimal 12-stage pipeline among fixed-depth pipelines, the deep 24-stage pipeline with ideal PSU enabling can gain 6.5% more EDP reduction, even after the utilization of complex clock gating which usually lower the chances of other energy saving technologies. By considering EDDP, which puts more emphasis on performance, the 24-stage pipeline design is still the best design after ideal PSU adoption. Averaged from the benchmarks we have studied, it achieves 8.29% more EDDP reduction than the 18-stage pipeline, which is the optimal design for a fixed-depth pipeline that adopts EDDP as metric.

The rest of the paper is organized as follows. Section II describes the main proposal of this paper. Simulation methodology to evaluate power/performance of different pipeline depths can be found in Section III. In Section IV we will show the experiment results, together with some analyses. Section V concludes the paper.

II. Power/performance vs. pipeline depth

A. Basic power/performance model

Considering a given technology, the clock frequency varies due to the changes in the pipeline design. It can be calculated as $\frac{1}{t_o+t_p/n}$, where n is the number of pipeline stages, t_p serves for the total logic delay of pipeline, and t_o represents the latch overhead per stage. t_p and t_o are usually expressed in the number of FO4 inverter delays. With the increase of n, data passes through fewer logic units in one cycle and the clock frequency becomes progressively larger consequently.

Performance is usually presented as delay. Averagely, the time consumed by each interval can be calculated as the product of Cycles Per Instruction (CPI) and the cycle period. From paper [3], together with our results in Section IV, we found that the average CPI could be regarded as linear to the number of pipeline stages, in most of the benchmarks. Roughly, we can express the CPI in the following equation:

$$CPI = CPI_0 \times (1 + \gamma n) \tag{1}$$

In this equation, CPI_0 is the mathematically predicted value when n is set to zero, and γ is the slope of the $\frac{CPI}{CPI_0}$ vs. n curve. Both of these two values can be obtained by fitting the CPI curves from our simulation results. The fitting data depicts that both CPI_0 and γ are positive.

We can then derive the delay of executing a program, as:

$$D(n) = \frac{N_I CPI}{f} = N_I CPI_0 (1 + \gamma n)(t_o + t_p/n) \tag{2}$$

The parameters are the same as defined in previous formulas except that N_I represents the instruction number in executing the program.

The power consumed in microprocessors is given by:

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

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 according to the 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 form. 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 power estimation method 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 the supply voltage gating method on the disabled pipeline registers. Besides, to reduce the dynamic power is also the main target of other energy saving technologies such as Dynamic Voltage Scaling (DVS) method, which scales down supply voltage when experiencing low workload.

The power/performance metrics, defined in paper [6], have the form of the following equation:

$$Metric(m, n) = P(n) \times D(n)^m \tag{4}$$

In this equation, m usually takes the value of 1, 2, or 3, which represents metric PDP, EDP, or EDDP, respectively. Combining these four equations, we can see that the metric is actually a function of n. And we can take the derivative of this metric function with respect to n, to find the theoretical optimal pipeline depth. It is a mathematically lengthy problem. Moreover, some parameters like CPI_0, γ and α in equation (1)-(3) are still uncertain, and will vary during the program execution. For these reasons, we use simulation results in the study of seeking the optimal number of pipeline stages.

B. Employing PSU, a pipeline with multi-usable depths

As described in paper [2], [3], several researches have been carried out to find the optimum pipeline depth for power/performance consideration based on a similar model in Section II-A. However, although they compared the different pipeline depths, they were using a predetermined pipeline depth during a whole execution. And their proposed optimal depth was averaged among a studied workload set. Under these assumptions, the differences between different programs or periods can not be noticed and hence may be eliminated. As we discuss in Section IV, it is 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.

For this purpose, we employed the earlier proposed Pipeline Stage Unification (PSU) mechanism, designed by Shimada [4]. In paper [4], PSU is designed as a method to unify adjacent pipeline stages via bypassing and in-activating some of the pipeline registers. Other than its original objective to save power consumed in the gated pipeline registers, PSU is rather a pipeline reconfiguration method. As shown in paper [4], a PSU-enabled pipeline was assumed to have the following three unification degrees:

(1) Unification Degree 1 (U1): The normal mode without bypassing any pipeline registers. It has a pipeline with 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 with $n/2$ stages.

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

According to this 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. It means that we need to dynamically find the best result from $Metric(m, n)$, $Metric(m, \frac{n}{2})$, and $Metric(m, \frac{n}{4})$ based on the history information, and use the corresponding unification degree for next bulk of instructions. Such an online PSU control mechanism has been detailedly studied in our other research [7]. In the latter part of this paper, we will demonstrate the relationship between the power/performance metric and a pipeline with multi-usable depths by the utilization of a profile based ideal PSU control mechanism.

III. Simulation methodology

To study the effect of different pipeline depths on power/performance, we varied the pipeline depth of a modern super-scalar architecture similar to current processors, which have relatively deep pipelines. This section describes the simulation framework and methodology we utilized 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 SPEC CPU2000, with train inputs. 1.5 billion instructions were simulated after skipping the first billion instructions.

A. Wattch tool set

We used Wattch tool set 1.02 [9] to collect the energy consumption results. In modern processors, α in equation (3) will vary a lot for different programs during different runtime periods because of the widely used clock gating. We used *cc3* method in Wattch to provide a complex clock gating simulation. In this clock gating method, power is scaled linearly with the 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 employ 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 does 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.

B. Processor parameters

In our simulation, we selected a processor with 20-stage fixed-depth pipeline, which is similar to the Pentium 4 architecture, to serve as the baseline processor. Table I shows the baseline processor configuration.

To define the detailed processor parameters that vary along with the pipeline depth, we assumed that the latches in pipeline approximately consume 30% of the total processor power in the 20-stage pipeline. This value is same with 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

TABLE I
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	32K-entry gshare, 13-bit history, 4K-entry BTB,32-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	32-entry I-TLB, 64-entry D-TLB, 128 cycles miss latency

TABLE II
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

to be 140 FO4 and 2.5 FO4 respectively, as described in paper [3]. Assume that the baseline 20-stage pipeline has a clock frequency of 2GHz, the frequencies of other pipelines with different stages can be calculated as:

$$f(n) = \frac{t_o + t_p/20}{t_o + t_p/n} \times 2GHz \qquad (5)$$

We used Cacti 3.0 tool set [10] to calculate the cache latency of the baseline processor configuration. For example, the L1 data cache of $90nm$ 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.

Branch misprediction resolution latency is defined as the time when the actual outcome of the branch is known. This value also varies for different designs. However, researches in paper [1], [11], [12] illustrate that this latency play a minor role in the final Instructions Per Cycle (IPC). In our simulation, we roughly assume that it changes linearly to the number of pipeline stages.

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 configurations in table II. The notations in table II list as follows:

- n: number of pipeline stages;

- $t_o + t_p/n$: FO4 inverter delay per stage (in FO4);
- Freq.: Clock frequency (in GHz);
- IL1, DL1, L2: latency of L1 instruction, L1 data, and L2 cache (in cycles);
- ALU: integer ALU latency (in cycles);
- MP.: latency through the branch resolution path (in cycles).

In our simulation, integer ALU latency in table II 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 [12].

IV. Results and analyses

In this section, we will study the relationship between power/performance metrics and pipeline depth, for pipelines with either a predetermined depth or multi-usable depths, by the help of simulation results.

A. Energy-only or Performance-only Consideration

If we set m in equation (4) to be 1, it represents the PDP metric, which is actually an energy-only metric. Roughly assuming a processor without clock gating, in which α of the $P(n)$ equation (3) remains to be a constant, we can get a degraded version of equation (4). After simplification, it takes the form of:

$$Metric(1,n) = A_2 n^2 + A_1 n + A_0 \quad (6)$$

All the coefficients in this equation are positive. Therefore, we can get the mathematical conclusion that the smallest PDP is always achieved in a one-stage pipeline without clock gating application. Dashed lines in figure 1 depict the results we obtained from SPEC CPU2000 integer benchmarks. Each $PDP(n)$ in this figure is normalized by $PDP(20)$ of the corresponding benchmark. "$PDP(20)$"-like notation stands for the PDP result of the processor with a fixed-depth pipeline of 20 stages. Similar notations are used in the latter parts of this paper. For simplicity, only 4 from those 8 benchmarks are listed. It shows that even with the application of sophisticated clock gating, the conclusion that shallow pipelines consume less energy is still applicable. Therefore, if PDP is used as the power/performance metric, there is no need to apply PSU mechanisms, since a 4-stage pipeline always achieves the best efficiency in our tests.

Another utmost situation occurs when we set m in $Metric(m,n)$ to be ∞. It eliminates the influence of power, and thus is a performance-only metric.

Solid lines in figure 1 demonstrate the delay results we got from SPECint2000 benchmarks. It shows a quite different trend than the energy-only metric. Delay shrinks dramatically from 4 to 20 stages design point, while keeps plain among deep pipelines which have more than 20 stages. Since the best

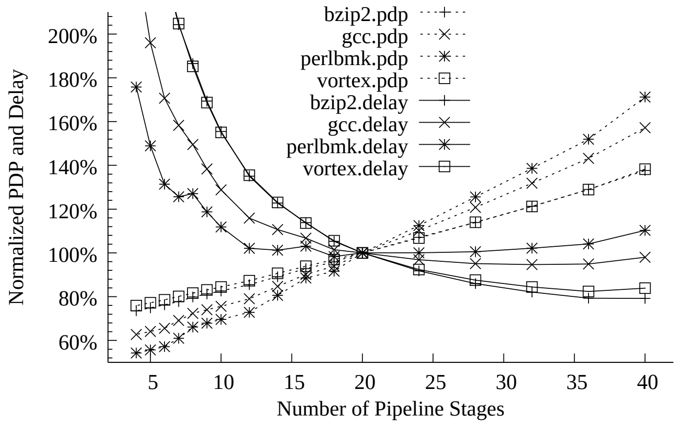

Fig. 1. Normalized PDP and Delay of fixed-depth pipelines

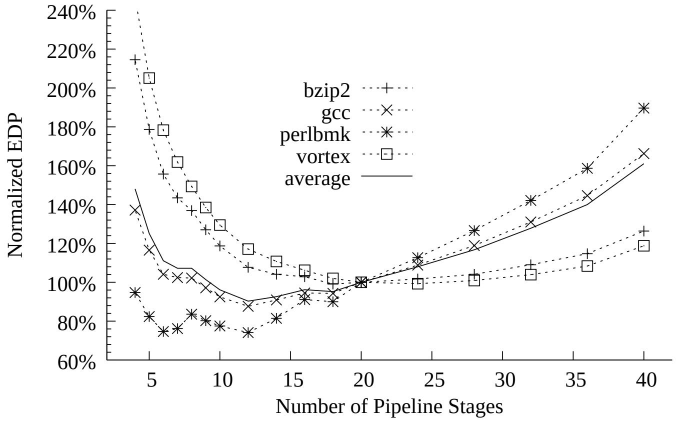

Fig. 2. Normalized EDP of fixed-depth pipelines

performance concentrates at the deep pipeline area, we can gain little achievement by applying a pipeline of changeable depths. Therefore, the efficiency of PSU will also be limited in platforms which focus on performance most.

B. EDP results

As either energy or performance only targeting seems less competitive in modern microprocessor designs, from this section, we will put emphasis on the metrics that combine both energy and performance considerations together.

1) Results of fixed pipeline depth: In this subsection, we follow the research work done in paper [2], [3] under our deployed environments. It is also the precedent work for the evaluations of PSU enabled pipelines.

Figure 2 shows the EDP results of pipelines with fixed depths, from 4 stages to 40 stages. The x-axis denotes the number of pipeline stages and the y-axis denotes the normalized EDP results. 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. Hence, 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 lines are 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 similar shapes to the benchmarks drawn in figure 2.

Figure 2 depicts 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 a 14.2 FO4 inverter delay, including 2.5 FO4 latch overhead. The 12-stage pipeline can achieve an EDP reduction of 9.70%, compared to the baseline 20-stage design, averagely for the 8 integer benchmarks.

Besides the optimal depth finding, figure 2 also shows that individual benchmark demonstrates various behaviors at the same design point. As shown in figure 2, for benchmark gcc and perlbmk, the optimal EDP can be obtained by a 12-stage pipeline, which is the same as the average line gives. For the benchmark bzip2, the pipeline with 18 stages has the smallest EDP result. It is a deeper pipeline with a 10.3 FO4 delay per each stage. And for benchmark vortex, the optimal design point occurs at even deeper pipeline design point with 24 stages and 8.33 FO4 per each stage. The other SPECint2000 benchmarks gzip, mcf, parser, and vpr demonstrate a similar behavior like the average data. Therefore, using 12-stage design to be the optimal value means that we must experience less EDP reduction in programs that have similar behaviors like bzip2 and vortex.

2) Results of ideal PSU method: In Section IV-B1, 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.

According to the proposal of PSU, an n-stage pipeline (at U1 mode) can be dynamically changed to an $n/2$-stage pipeline under U2 mode, or to an $n/4$-stage pipeline unde U4 mode during the program execution, as we have described in Section II-B. As shown in paper [7], program may experience different phases during the whole execution. If PSU selects suitable unification degree for each phase, it can achieve better efficiency in reducing EDP than the 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 paper [7]. Suppose that we apply this method on a 20-stage PSU enabled 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 the fixed 20-stage mode, the fixed 10-stage mode, and the fixed 5-stage mode, similarly as we did in Section IV-B1. 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. Thus, in the ideal PSU adoption mode, we could set the best unification degree at the beginning of an interval based on the profiling data. Although this method can not be achieved practically, it is useful to study the direct impact of different pipeline designs in PSU enabled processors.

Figure 3 shows the EDP results of processors with ideal

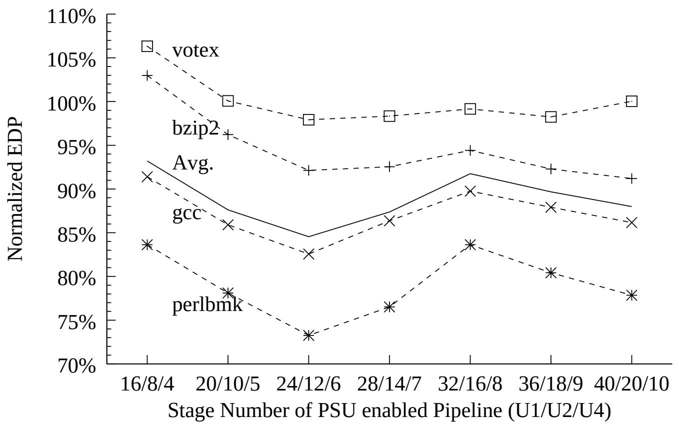

Fig. 3. Normalized EDP of ideal PSU-enabled pipelines

PSU adoption. Its x-axis denotes the different PSU-enabled pipeline configuration, from 16-stage to 40-stage. 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. The y-axis denotes the normalized EDP results. The "Avg." line is the mean EDP result from all the 8 SPECint2000 benchmarks. 4 dashed 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 3 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 delay, including 2.5 FO4 latch overhead. Furthermore, this design point is also effective for the individual benchmark consideration, i.e., most of the 8 benchmarks experience the smallest EDP under the 24/12/6 pipeline configuration with ideal PSU adoption.

The only deviation occurs in benchmark bzip2, where the 40/20/10-stage design is the best one. However, $EDP(24/12/6)$ of bzip2 is only about 1% larger than $EDP(40/20/10)$. This observation indicates that the PSU enabled pipeline can greatly alleviate the influence introduced by the 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 3 is that the deviations between different pipeline configurations are not large. For most of the benchmarks, the deviations are 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 3 is far better than that of figure 2.

Moreover, figure 3 indicates that after applying PSU, we can obtain more EDP reduction, compared to the best fixed pipeline depth in figure 2. In figure 2, the optimal depth occurs at a 12-stage pipeline, considering all benchmarks

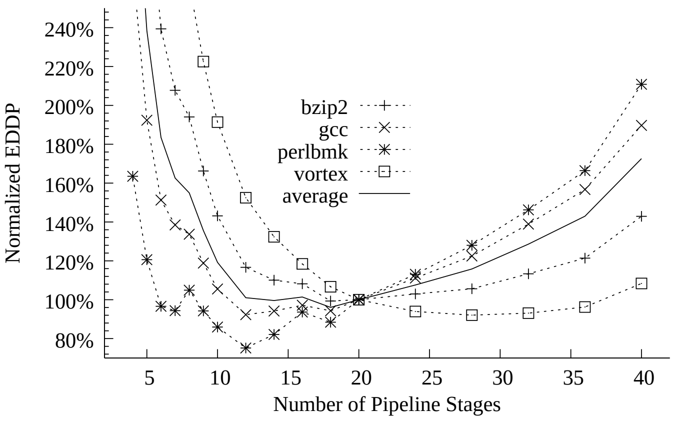

Fig. 4. Normalized EDDP of fixed-depth pipelines

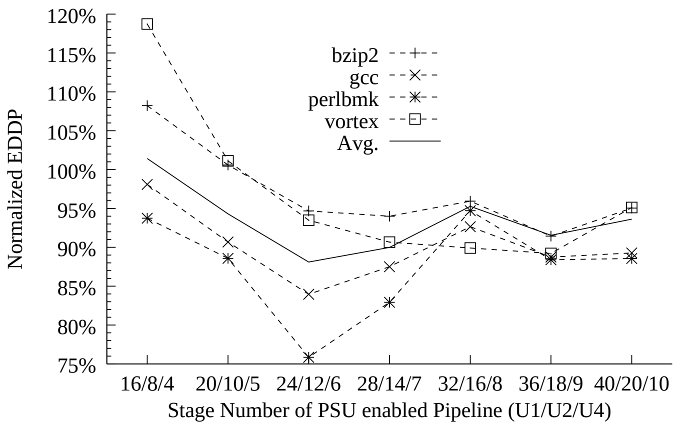

Fig. 5. Normalized EDDP of ideal PSU-enabled pipelines

averagely. Its EDP result is 90.3% of the baseline 20-stage pipeline one, while the EDP of 24/12/6-stage PSU enabled pipeline in figure 3 is 84.5% of the baseline 20-stage pipeline's EDP. This PSU-enabled pipeline saves 6.5% more EDP, compared to the fixed-depth 12-stage pipeline value. Individually, for gcc, the 24/12/6 stage achieves about 5.78% more EDP reduction than the fixed-depth 12-stage design point. This additional reduction is obtained by executing 37.5% instructions in the 24-stage, 50.9% instructions in the 12-stage, and 11.6% instructions in the 6-stage pipeline. Such an 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 the best pipeline depth is necessary and may probably show better efficiency in the modern processors, if frequency goes higher.

Usually the complex clock gating in Wattch will leave little space for other energy saving technologies. For comparison, we also did a series of experiments in which a fully switched processor is assumed. The EDP reduction by ideal PSU is about 28.5%, compared to $EDP(20)$ in processors without clock gating. Although the energy saving by PSU method is not so large after the utilization of clock gating, the chance is not completely eliminated and we can still gain 15.5% EDP reduction. For consistency, we continue to use results with clock gating in the latter parts to analyze the EDP or EDDP savings for pipelines with ideal PSU adoption.

C. EDDP results

In this section, we will study the EDDP results after using Wattch to provide a complex clock gating. We did the same two series of evaluations, one for fixed-depth pipeline during a program execution (figure 4), the other for pipelines with ideal PSU utilization (figure 5). The notations, x-axis, and y-axis are similar to the ones in figure 2 and 3, except that these two figures are using EDDP to serve as the power/performance metric.

Figure 4 demonstrates that with more considerations on the performance by metric EDDP, the optimal pipeline depth of a fixed pipeline now occurs at the 18-stage pipeline design point, averaged from the 8 integer benchmarks. Individually, the optimal fixed depths for these benchmarks are shown in table III. Still, the averagely optimal fixed 18-stage pipeline does not fit for all 8 benchmarks.

TABLE III
n_{opt} FOR BENCHMARKS (EDDP)

n_{opt}	benchmarks
12	gcc, gzip, mcf, parser, perlbmk, vpr
18	bzip2, mcf, parser
28	vortex

Figure 5 is the results after applying ideal PSU. We can draw similar conclusions from this figure as we did with figure 3. After the ideal PSU adoption, the averaged optimal design point is still the 24/12/6-stage pipeline configuration. At this design point, we can achieve 11.9% EDDP saving, compared to $EDDP(20)$ of the baseline fixed-depth pipeline. Compared to the 18-stage pipeline which is the optimal design for a fixed-depth pipeline, the 24/12/6-stage pipeline achieves 8.29% more EDDP reduction, averaged from 8 benchmarks. This indicates that the PSU utilization will have efficiency in reducing the processor EDDP, even with a complex clocking gating application.

D. Considering voltage scaling

Dynamic Voltage Scaling (DVS) is a commonly used method in energy saving fields. However, as we studied the original DVS model, we found that in the recent chip technologies, it can hardly help the reduction in EDP or EDDP despite of its great saving in PDP. Simply considering a processor with one power-supply network, we can assume that the performance degrades linearly, as voltage scales down. For a bulk of instructions that is selected to execute in lowered voltage, we can approximately have:

$$\frac{Metric_{DVS}(m)}{Metric_{normal}(m)} = \left(\frac{V_{low}}{V_{dd}}\right)^2 \left(\frac{f}{f_{low}}\right)^{m-1} \quad (7)$$

Detailedly, if we are considering a processor like $90nm$

Pentium M [13], we can assume the following parameters for equation (7): $V_{dd} = 1.34V$, $f = 2GHz$ and $V_{low} = 1.1V$, $f_{low} = 1GHz$.

When $m = 1$, $\frac{PDP_{DVS}}{PDP_{normal}} = 0.6738$. It indicates that DVS can efficiently reduce the PDP.

However, when $m = 2$, $\frac{EDP_{DVS}}{EDP_{normal}} = 1.347$. It shows that DVS will experience some penalties when considering EDP metric. Similar degradations also exist when choosing metric EDDP. DVS suffers from the penalty in the delay as m becomes larger.

Moreover, there are many restrictions of voltage scaling in the current and the future process technologies (e.g. soft error, process deviation). The ineffectiveness of DVS in metric EDP and EDDP will probably become even larger in the future process technology.

Since DVS has good PDP result, we believe that it will have the best outputs if there is a deadline existing for the bulk of instructions. Under such circumstance, voltage can be lowered until successfully meeting the throughput restriction. A. Hyodo [14] has detailedly studied energy saving schemes by varying both supply voltage and pipeline depth in platforms where target throughput plays the dominant role. Yet it is different from the scope of this paper as we focus on the relationship between the best power/performance metric results and processor designs.

Recently, some researchers begin to consider hiding the DVS performance degradation under the L2 cache misses [15] with the help of multiple power-supply networks. Here, the L2 cache miss can be regarded as a special deadline for an instruction interval. However, the L2 cache misses related schemes will have some limitations: it shows the best efficiencies in memory intensive benchmarks and has small outcomes for other benchmarks. Even for memory intensive benchmarks, such outcomes will be limited by clock gating and data prefetching technologies.

For the purpose of considering both energy and performance, we do not apply voltage scaling on processors in this paper. And our results show that PSU enabled processors can still get achievements in EDP and EDDP in most benchmarks, even with the application of complex clock gating.

V. Conclusions

In this paper, we have studied relationship between power/performance metric 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 if we consider metric EDP. 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 and EDDP 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. It saves 15.5% EDP, compared to the baseline 20-stage pipeline with clock gating utilization. Furthermore, such an EDP result is 6.5% smaller than the fixed 12-stage pipeline, which is the optimal one for a fixed-depth pipeline design. Similar conclusions can be obtained when considering power/performance metric EDDP.

Acknowledgment

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

References

[1] M. Hrishikesh, K. Farkas, N. Jouppi, D. Burger, S. Keckler, and P. Sivakumar, *The Optimal Logic Depth Per Pipeline Stage is 6 to 8 FO4 Inverter Delays*, Proceedings of the 29th International Symposium on Computer Architecture (ISCA 29), pp. 14–24, 2002.

[2] V. Srinivasan, D. Brooks, M. Gschwind, P. Bose, V. Zyuban, P. N. Strenski, and P. G. E. Sylvester, *Optimizing Pipelines for Power and Performance*, Proceedings of the 35th International Symposium on Microarchitecture (MICRO 35), pp. 333–344, 2002.

[3] A. Hartstein and T. R. Puzak. *Optimum Power/Performance Pipeline Depth*, Proceedings of the 36th International Symposium on Microarchitecture (MICRO 36), pp. 117–128, 2003.

[4] H. Shimada, H. Ando, and T. Shimada, *Pipeline Stage Unification: A Low-Energy Consumption Technique for Future Mobile Processors*, Proceedings of the 2003 International Symposium on Low Power Electronics and Design (ISLPED 2003), pp. 326–329, 2003.

[5] J. Koppanalil, P. Ramrakhyani, S. Desai, A. Vaidyanathan, and North E. Rotenberg, *A Case for Dynamic Pipeline Scaling*, Proceedings of the 2002 international conference on Compilers, architecture, and synthesis for embedded systems CASES 2002), pp. 1–8, 2002.

[6] R. Gonzalez and M. Horowitz, *Energy dissipation in general purpose microprocessors*, IEEE Journal of Solid-State Circuits (IEEE JSSC), Vol. 31, No. 9, pp. 1277–1284, 1996.

[7] J. Yao, H. Shimada, Y. Nakashima, S. Mori, and S. Tomita, *Program Phase Detection Based Dynamic Control Mechanisms for Pipeline Stage Unification Adoption*, Workshop on ALPS 2006, pp. 39–46, 2006.

[8] D. C. Burger, and T. M. Austin, *The SimpleScalar Tool Set, version 2.0*, Technical Report CS-TR-97-1342, Department of Computer Science, University of Wisconsin-Madison, 1997.

[9] D. Brooks, V. Tiwari, and M. Martonosi, *Wattch: A framework for architectural-level power analysis and optimization*, Proceedings of the 27th International Symposium on Computer Architecture (ISCA 27), pp. 83–94, 2000.

[10] P. Shivakumar and N. P. Jouppi. *CACT 3.0: An Integrated Cache Timing, Power and Area Model*, Technical report, WRL Research Report, 2001.

[11] A. Jagannathan, H. Honghua Yang, K. Konigsfeld, D. Milliron, M. Mohan, M. Romesis, G. Reinman, J. Cong, *Microarchitecture evaluation with floorplanning and interconnect pipelining*, Proceedings of the 2005 Asia and South Pacific Design Automation Conference (ASP-DAC 2005), pp. 8–15, 2005.

[12] H. Shimada, H. Ando, and T. Shimada, *Reducing Processor Energy Consumption with Pipeline Stage Unification (in Japanese)*, IPSJ Transactions on Advanced Computing System, Vol. 45, No. SIG 1(ACS 4), pp. 18-30, (2004).

[13] Intel Corporation, *Intel Pentium M Processor on 90nm Process with 2MB L2 Cache Datasheet*, 2006.

[14] A. Hyodo, M. Muroyama, H. Yasuura, *Variable Pipeline Depth Processor for Energy Efficient Systems*, IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer, Vol. 86, No. 12, pp. 2983–2990, 2003.

[15] Li Hai, Cher Chen-Yong, K. Roy, and T. N. Vijaykumar, *Combined Circuit and Architectural Level Variable Supply-Voltage Scaling for Low Power*, IEEE Transactions on VLSI systems, Vol. 13, No. 5, pp. 564–576, 2005.