International Journal of Networking and Computing – www.ijnc.org
ISSN 2185-2839 (print) ISSN 2185-2847 (online)
Volume 3, Number 2, pages 192–204, July 2013

# Energy Optimization using Fine-Grain Variable Stages Pipeline Processor Chip

Tomoyuki Nakabayashi, Takahiro Sasaki, Hitoshi Nakamura, Kazuhiko Ohno and Toshio Kondo
Graduate School of Engineering, Mie University
1577 Kurimamachiya-cho, Tsu, Mie, 514-8507, Japan

Received: February 19, 2013
Revised: April 18, 2013
Accepted: June 18, 2013
Communicated by Yasuaki Ito

**Abstract**

Increased energy consumption in processors caused by performance enhancement has recently become a critical problem. Many current processors employ dynamic voltage and frequency scaling (DVFS) which dynamically lowers the supply voltage and clock frequency in order to reduce energy consumption. However, it is difficult to deliver fine-grain energy optimization by using DVFS. Since a voltage regulator takes a long time for scaling the voltage and charging/discharging a power line has a large energy overhead, the useful interval of DVFS is limited to coarse-grain. To optimize energy consumption at fine-grain interval, we have proposed a variable stages pipeline (VSP) processor. VSP reduces energy consumption by dynamically varying the pipeline depth to suitable pipeline depth according to behavior of a running program. VSP can obtain finer-grained energy reduction than DVFS because pipeline scaling only requires a small overhead. In this paper, we fabricated a VSP processor chip using 180 nm technology and evaluated energy consumption of the chip. We present that the fabricated VSP chip dynamically varies the pipeline depth while a program is running and reduces the energy consumption at shorter interval than DVFS. We also analyze how to optimize the energy consumption according to system demand. Our analysis result shows that the VSP can adjust the energy consumption in the same manner for diverse program phases.

*Keywords:* Energy optimization, VLSI design, Low-energy processor architecture, Variable-depth pipeline

## 1 Introduction

The latest advances in mobile computers, such as personal digital assistants and smart phones, have led to a goal of higher computing performance with lower energy consumption. Dynamic voltage and frequency scaling (DVFS) [1, 2] which dynamically lowers the supply voltage and clock frequency is currently used in many processors to reduce energy consumption. Lowering the supply voltage effectively reduces energy consumption because energy consumption depends on the square of supply voltage.

However, it is difficult to deliver fine-grained energy reduction using DVFS technique because voltage scaling takes a long time and charging/discharging a power supply line consumes a large energy consumption. Therefore, the useful interval of DVFS is limited to coarse-grain. Reference [3] estimates the useful interval of DVFS in terms of energy reduction and it takes at least $10^5$ cycle

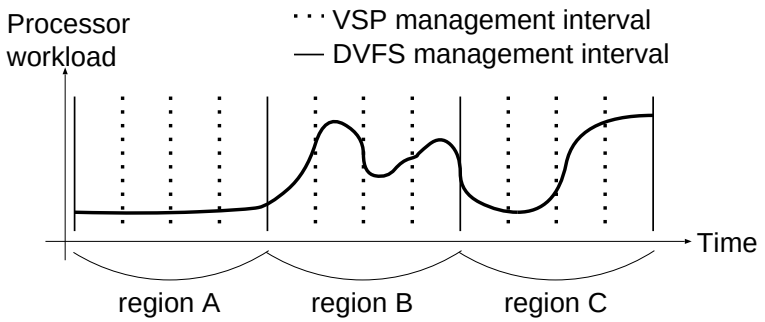


Figure 1: Relation between workload and energy management interval.

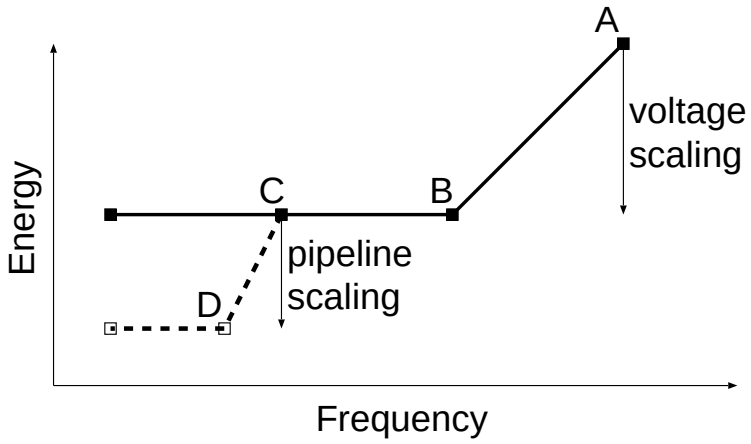


Figure 2: Combining voltage scaling and pipeline scaling.

order to reduce energy consumption. In general, it is said that monitoring the workload of a processor and scaling supply voltage take tens or hundreds of microseconds. This is shown in Fig. 1 for the lines labeled "DVFS management interval". In region A, DVFS can effectively reduce energy by lowering the voltage and frequency because the workload is light through the DVFS management interval. In region B and C, however, the workload transitions at shorter interval than the DVFS management interval. If DVFS raises the voltage and frequency for the peak workload in region B, the processor wastes energy at low workload region in region B. In contrast, if DVFS sets lower voltage and frequency than that of the peak, response time is degraded.

In addition, we point that the effectiveness of DVFS is limited because there is a lower bound on voltage of minimum operating voltage of CMOS devices. Figure 2 shows this problem. A processor that only DVFS is applied reduces energy consumption by lowering the voltage and frequency as shown point A to B in Fig. 2. However, once the voltage has reached the lowest voltage, lowering the frequency does not save further energy consumption as shown point B to C because lowering the frequency without lowering the voltage reduces only power. Therefore, developing low-energy techniques that compensate for DVFS is essential.

To achieve such requirement, we have proposed a variable stages pipeline (VSP) technique [4–7] that is similar technique to dynamic pipeline scaling (DPS) [12] and pipeline stage unification (PSU) [8–11]. VSP processor dynamically varies the depth of the pipeline stages and clock frequency according to the workload of the processor. Unlike DVFS, VSP processor can save energy consumption in an interval of several ten or hundred clock cycles because the penalty of varying the pipeline depth takes only a few clock cycles. Therefore, fine-grained energy optimization is

possible if VSP processor changes the pipeline depth to a suitable pipeline depth using a fine-grain controller. Some controllers for changing pipeline depth have been proposed to dynamically control the pipeline depth [10, 11]. As a pipeline depth controller to achieve highly energy-efficient computing, we have proposed a fine-grain controller that predicts a suitable pipeline depth at several tens of clock cycles and have demonstrated the effectiveness of our controller using SimpleScalar [13] processor simulator [6]. The advantage of our fine-grain control is shown in Fig. 1 for the doted lines labeled "VSP management interval". In region B and C, more aggressive energy reduction is performed by using VSP which has fine-grain controller. Furthermore, even if the workload requires middle performance of the processor, VSP can reduce energy consumption like DVFS but the useful interval is finer-grained. To reduce more energy consumption, VSP can be used with DVFS at the same time. The point D in Fig 2 shows that VSP can reduce energy consumption beyond the lower bound of DVFS. Ref. [12] points this and briefly estimates the effectiveness.

In spite of that variable-depth pipeline architecture can reduce energy consumption at finer-grained interval than DVFS, no previous works have proposed that using the approach to save short-term energy. Furthermore, all previous works have not designed and evaluated the VLSI of a hardware-implemented processor which dynamically varies the pipeline depth while a program is running.

Therefore, in this work, we fabricate the VSP processor using 180 nm technology to evaluate energy consumption in detail. This paper makes the following contributions by evaluating our fabricated VSP chip:

- The detailed energy evaluation and hardware cost to implement variable-depth pipeline architecture are presented.
- We make it clear that VSP can dynamically vary the pipeline depth according to behavior of a running program and reduce energy consumption at shorter interval than DVFS.
- VSP chip can change behavior according to trade off between energy consumption and performance in a step-by-step by changing thresholds which are used to predict suitable depth.
- We reveal how to decide the thresholds to optimize energy consumption.

## 2 Variable stages pipeline

Energy-efficient VLSI designs have become more important in recent years with low-energy and high-performance requirements. To achieve the requirements, Shimada, et.al. propose PSU [8–11], Koppanalil, et.al. propose DPS [12] and we propose VSP [4–7]. Figure 3 shows the basic concept of these approaches. Generally, if the pipeline depth is increased, such as in super-pipeline architecture, then the clock frequency, performance, and energy consumption are also increased. In contrast, if the pipeline depth is decreased, then the clock frequency, performance, and energy consumption are also decreased. When the processor workload is light, the processor lowers clock frequency and unifies plural stages to form a shallower pipeline.

However, with DPS and PSU, the number of glitches increases under low-energy mode because the scale of the combinational circuits in a pipeline stage is increased by unifying the pipeline stages. Figure 4 shows the pipeline register used in PSU processor that can vary the depth of the pipeline stages. Under high-speed mode, the DFF+MUX shown in Fig. 4 acts as a general pipeline register by selecting an upper path. Under low-energy mode, the DFF+MUX unifies the pipeline stages by selecting a lower path to connect the input port to the output port. However, glitch propagation caused by stage unification becomes a serious problem. Note that a glitch means an unnecessary transition in performing a computation. Generally, glitches are generated by irregular delays of logic gates and wires, and the frequency of glitch propagation exponentially increases in proportion to the scale of the combinational circuits. Energy dissipation caused by glitch propagation exponentially increases because multiple pipeline stages are unified and they have a large combinational circuit.

To prevent glitch propagation, VSP technique introduces the latch D-FF selector-cell (LDS-cell) shown in Fig. 5 as a pipeline register instead of the DFF+MUX. Under low-energy mode, the LDS-

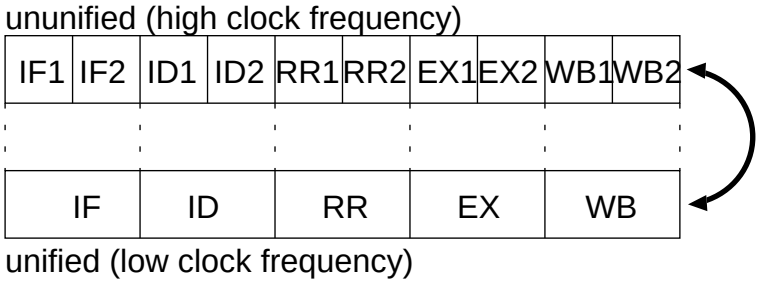


Figure 3: Variable-depth pipeline architecture.

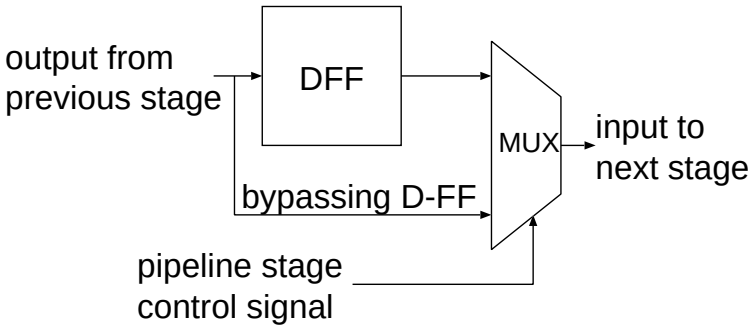


Figure 4: Circuit diagram of pipeline registers to vary pipeline.

cell outputs a master D-latch signal to prevent glitches. We have demonstrated that VSP processor consumes 13% less energy than PSU under low-energy mode [7].

To reduce energy consumption with minimum performance degradation, it is essential to correctly change the pipeline depth to a suitable pipeline depth. Therefore, we have proposed a fine-grain pipeline depth controller that stores some processor states (correlating to the workload of the processor) in the latest several tens of cycles and predicts a suitable pipeline depth using the stored information [6]. In this approach, we can choose information such as cache hit ratio, the number of cache access, instruction per cycle (IPC), and so on, to predict a suitable pipeline depth.

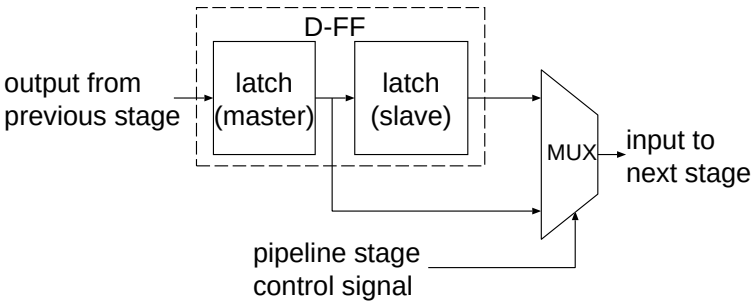


Figure 5: Circuit diagram of Latch D-FF Selector-cell (LDS-cell).

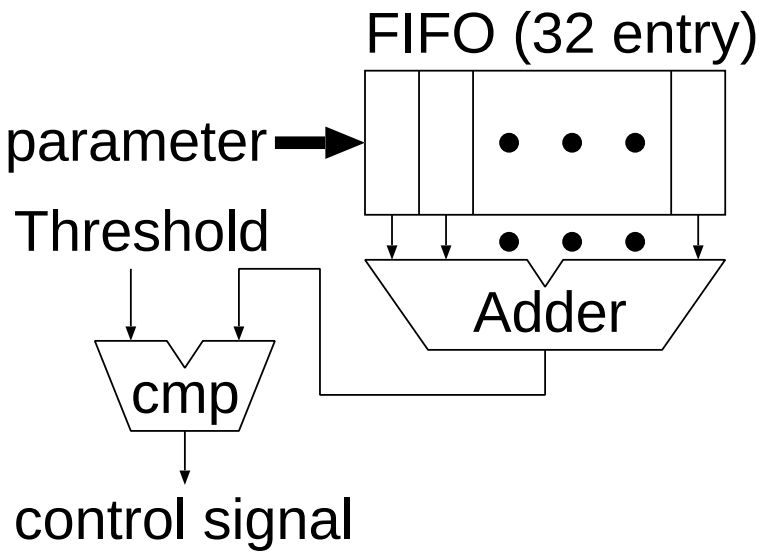


Figure 6: Block diagram of fine grain controller.

## 3 Related work

Koppanalil demonstrates that using DPS and DVFS at the same time consumes from 23% to 40% less energy than a rigid deep pipeline using DVFS. However, Koppanalil evaluates DPS using the fixed depth (deeper or shallower) on a benchmark program and does not proposed depth varying controller.

Shimada proposes combining PSU and DVFS that dynamically changes pipeline depth, voltage and frequency while a program is running [9]. However, interval of changing pipeline depth is the same as DVFS, and so fine-grained energy reduction cannot be obtained. Yao proposes a fine-grain pipeline depth controller for PSU [11] which is similar to our approach one year after we propose, but combining with DVFS is not proposed.

More importantly, all of the above works have not implemented the hardware. Therefore, an accurate evaluation for energy consumption and hardware cost has not been evaluated.

## 4 Implementation

### 4.1 Processor architecture

In this section, we describe the specific processor architecture we use in this work. We use a MIPS R3000 compatible processor which is seven stages in-order processor. The processor takes one cycle for branch instructions, two cycles for most integer ALU instructions and four cycles for complex instructions (multiplication and division). Generally MIPS processors do not adopt ALU of plural stages, however we employed such implementation to balance cycle time in each pipeline stage. As for our pipeline balancing, we hypothesize that cache system uses a high speed memory and/or is also pipelined like modern processors hence a path through instruction and data cache is not the critical path. We divided the ALU and MDU to two and four stages respectively in order to adjust the cycle time to around 5.9 ns because the longest stage except for the EX stage takes 5.9 ns. After the pipelining, the cycle times of pipelined ALU and MDU are 5.1 ns and 6.4 ns, respectively. The processor has 1 K-entry gshare branch predictor. Figure 7 shows the pipeline stages of the VSP processor. We call deeper (7-stage) pipeline *high-speed mode* and shallower (3-stage) pipeline *low-energy mode*. The features of low-energy mode are the following:

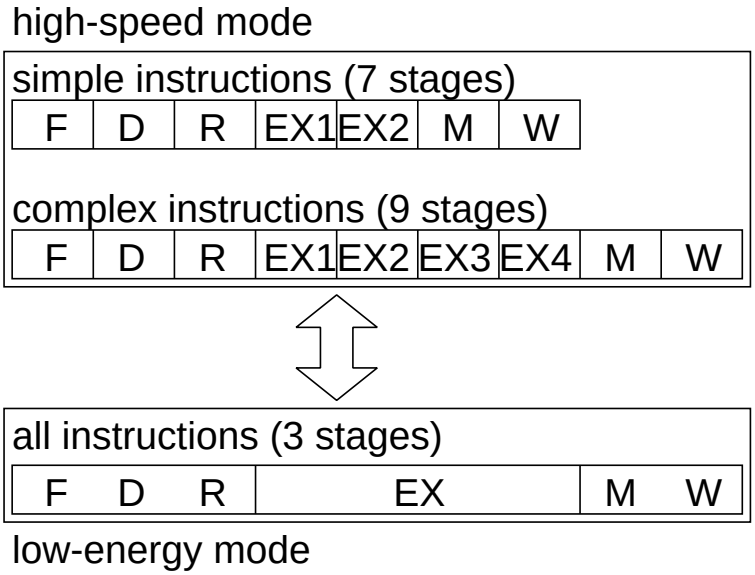


Figure 7: Block diagram of R3000 based VSP processor.

- The branch predictor is stopped because branch mis-prediction does not occur.
- No interlock occurs by data dependency between arithmetical operations.
- Clock lines for unused pipeline registers between unified stages are gated.

We use the IPC as parameter shown in Fig. 6 to predict suitable pipeline depth. Under high-speed mode, if the IPC is larger than threshold, the VSP processor remains unchanged because the pipeline effectively works. On the other hand, if the IPC is lowered, the VSP processor shifts to low-energy mode. Under low-energy mode, if the IPC is larger than threshold, then the VSP processor shifts from low-energy mode to high-speed mode, and vice versa. However, a problem occurs when the processor shifts from low-energy mode to high-speed mode. In general, a large number of branch mis-predictions degrades the IPC but this IPC degradation does not occur in low-energy mode. If the VSP processor runs under low-energy mode in branch mis-prediction intensive phase, the IPC approaches the highest IPC because branch mis-prediction does not occur due to pipeline unification and then the VSP processor shifts back to high-speed mode. As a result, the VSP processor runs an improper pipeline depth and the performance degrades. To solve the problem, we also use the number of branch instructions to predict when the processor shifts to high-speed mode because a branch mis-prediction intensive phase includes a large number of branch instructions. Therefore, we use three thresholds: $IPC_{HtoL}$ for unifying pipeline stages, $IPC_{LtoH}$ and $\#BR$ for forming deeper pipeline. These thresholds can be set by software: writing a new value to co-processor register unused by the base processor. An appropriate thresholds configuration according to workload is described in Section 6.

### 4.2 Low-overhead depth changing technique

With our proposed controller, the pipeline depth is always suitable for a running program. However, if the pipeline depth frequently changes, execution time will be increased. Forming a deeper pipeline does not have any problem. We assume that all pipeline stages execute valid instructions under low-energy mode and the processor makes deeper pipeline. In this case, to execute a program correctly, the VSP processor only has to perform the following step. The F, EX1 and M stages execute next instructions and the other stages just purge the holding instructions.

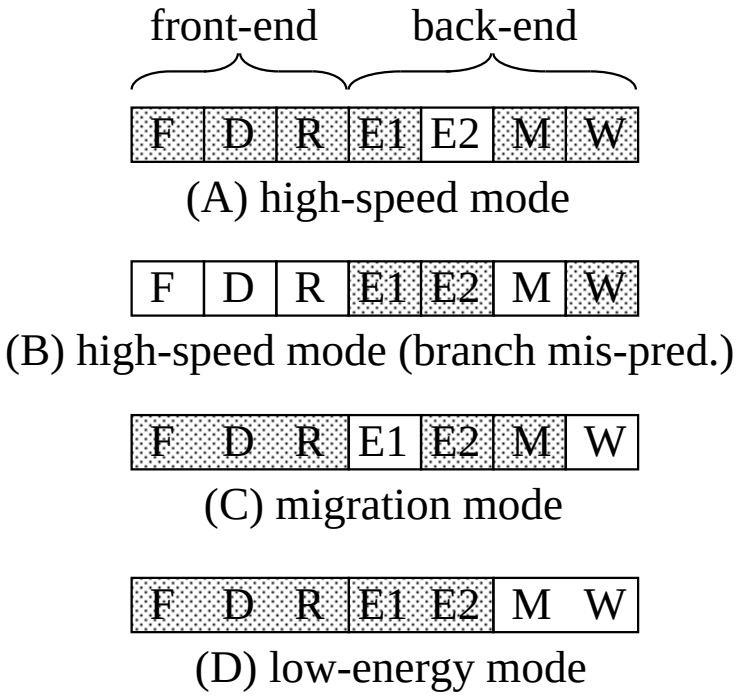


Figure 8: Change pipeline depth through migration mode.

In contrast, when unifying the stages, the processor requires a pipeline flush because following instruction destroys preceding valid instructions. For example, the F, D, and R stages in Fig. 8(A) are executing valid instructions and if they are immediately unified, the instructions in the D and R stage are destroyed by the instruction in the F stage. As a result, the processor goes into an unrecoverable state. If the processor flushes the pipeline and restarts the instructions in order to solve the problem, seven clock cycles are needed for an unification.

Therefore, we hide the penalty by overlapping a pipeline flush caused by a branch mis-prediction. Furthermore, to finely control the high-speed and low-energy modes, we divide the pipeline stages into two groups: the front-end group contains instruction fetch, instruction decode and register read stages, and the back-end group contains the latter stages. We assume that the processor runs under high-speed mode as shown in Fig. 8(A), and valid instructions exist in the shaded boxes (F, D, R, E1, M and W). A branch mis-prediction occurs at the branch instruction in the E1 stage. We note that the instruction in the R stage is a branch delay slot which must not be purged in a pipeline flush caused by a branch mis-prediction. When a branch mis-prediction occurs and pipeline depth controller is asserting pipeline unification signal, instructions in the front-end stages are purged and the branch delay slot enters the E1 stage as shown in Fig. 8(B). In the next cycle, the processor shifts to migration mode as shown in Fig. 8(C) and a new instruction is fetched. Under migration mode, the front-end stages are unified and driven at low clock frequency (in this case quarter of high-speed mode), and the back-end stages are driven at high clock frequency. It takes four high clock frequency cycles that the new fetched instruction reaches the E1 stage. Therefore, instructions in the back-end stages can be retired except for rare case.[1]. When all the back-end stages become empty, they shift to low-energy mode as shown in Fig. 8(D).

In this way, our controller can hide the penalty of changing the pipeline depth. Generally, a branch mis-prediction occurs with a probability of several percent, and a branch instruction exists with a probability of 20%. Therefore, there are sufficient opportunities to change the pipeline depth in tens or hundreds of cycles.

This technique obtains more effectiveness in a superscalar and super-pipeline processor which

[1]If the branch delay slot is complex instruction that takes multi cycles in the execution stage

Table 1: EDA environment for fabricating chip

| Phase | EDA tool |
|---|---|
| functional verification | Synopsys VCS, vers. 2009.06 |
| synthesis | Synopsys Design Compiler, vers. 2010.03-SP5 |
| place & route | Synopsys Astro, vers. 2007.03-SP12 |
| LDS-cell design | Cadence IC, vers. 5141 |
| design rule check | Mentor Graphics Calibre, vers. 2010.4 |

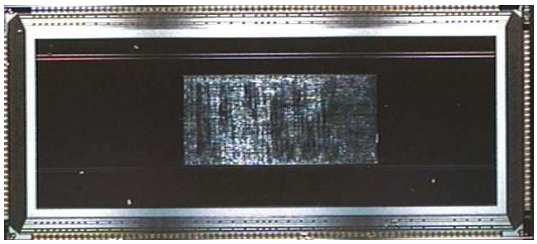

Figure 9: Chip micrograph.

have a large pipeline flushing penalty. The effectiveness of this technique is described in section 5.3.

## 4.3 Fabrication of VSP chip

We implemented the VSP processor described the previous section by using Verilog HDL. Table 1 shows the EDA tools used for designing VSP chip. We used Rohm 180 nm technology and Kyoto University standard cell library [14] except for LDS-cell. The processor was designed to operate at 100MHz for high-speed mode and 25MHz for low-energy mode.

We did not implement a cache system in the chip to evaluate an accurate energy consumed by the VSP processor. We note that this implementation does not degrade the accuracy of evaluation. In an in-order processor, the energy consumption caused by a cache miss under high-speed and low-energy modes are the same if the processor is applied clock gating when a cache miss occurs.

Figure 9 shows a micrograph of the fabricated VSP chip. The VSP processor chip has 492,742 transistors.

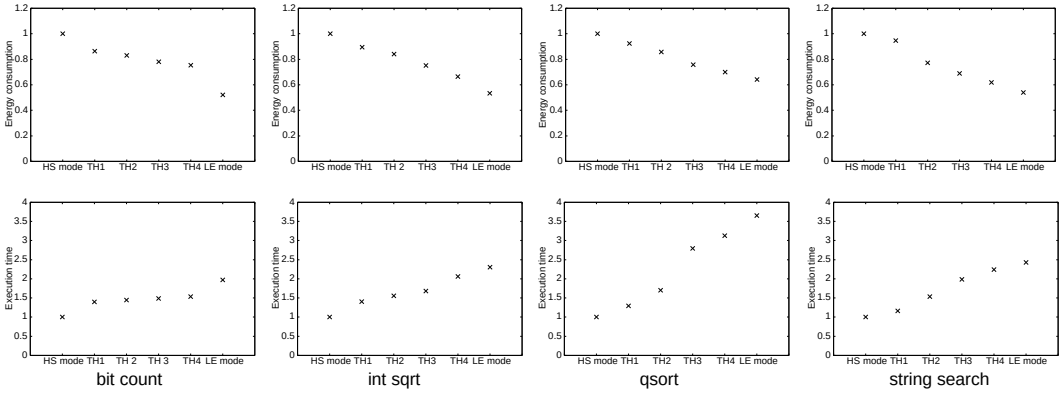


Figure 10: Energy consumption and execution time on diverse combination of thresholds.

# 5 Evaluations

## 5.1 Evaluation methodology

We used four integer benchmarks (bit count, int sqrt, quick sort and string search) from MiBench benchmark suit [15]. To clarify short-period energy reduction, we adjusted these benchmarks to finish approximately 100,000 clock cycles. Evaluations of the energy for section 5.2 were measured by using the fabricated chip. We confirmed that the fabricated chip executes the benchmarks successfully. Although we designed the chip to operate at 100 MHz for high-speed mode and 25 MHz for low-energy mode, we evaluated the chip at 35 MHz and 8.75 MHz respectively because of limitation of our test environment.

## 5.2 Energy consumption

Figure 10 compares energy consumption and execution time on diverse combinations of thresholds. HS mode and LE mode of the horizontal axis show results when the processor runs at fixed (high-speed/low-energy) mode and the others (TH1 to TH4) represent different combinations of three thresholds which are configured to achieve lower energy to the right. The vertical axis is normalized to result of high-speed mode. In the LE mode, the VSP processor reduces energy consumption by 34% to 48% compared with the HS mode. The energy advantage is due to higher IPC and clock gating for unused pipeline registers and branch predictor. TH1 to TH4 interpolate energy consumption and performance between high-speed and low-energy modes. This means the VSP reduces energy consumption at a short-interval (less than 100,000 cycles) according to configured thresholds even if the processor is required to achieve middle performance. In other words, the VSP can optimize trade off between the performance and energy in the region B and C shown in Fig. 1. Dynamic thresholds configuration to produce more flexible optimization is left for future work, but we present case study in Section 6.

Figure 11 shows the effectiveness of combining DVFS and VSP. We used 1.44 V (20% lower than standard voltage of hte process we used) for the lowered voltage which is calculated by the margin for 45 nm Intel Atom processor [16]. The DVFS on rigid pipeline consumes approximately 62% energy of the rigid pipeline with standard voltage. This result is consistent with that energy depends on the square of voltage. The vertical axis of Fig. 11 is normalized to the result of DVFS on rigid pipeline to make the effectiveness of combining two techniques clear. Although the VSP reduces larger energy than the DVFS except for qsort, we note that DVFS might be more energy efficient than VSP on a leakage dominant process because DVFS also is effective for leakage current. The DVFS on VSP reduces energy consumption by 38% to 52% with respect to the DVFS on rigid pipeline. It is obvious that using DVFS and VSP at the same time significantly reduces energy consumption as shown in Fig 2.

Finally, we measure energy consumed by our pipeline depth controller. The fabricated chip has an input to disable (apply clock gating for) the pipeline depth controller. When the controller is disabled, the VSP runs as fixed high-speed mode or fixed low-energy mode (the mode is specified externally). We compared controller-disabled VSP which runs as fixed high-speed mode with controller-enabled VSP which is configured its thresholds to run as fixed high-speed mode. As a result, it makes clear that the controller consumes only 2.1% energy of the VSP. This means the hardware cost of the controller is small enough.

## 5.3 Effectiveness of low-overhead depth changing technique

We briefly estimated the effectiveness of low-overhead depth changing technique we describe in section 4.2. Table 2 shows performance degradation caused by pipeline unification in case of without the low-overhead technique. We assume a pipeline flush takes 7 cycles and this penalty is required when the VSP shifts from high-speed mode to low-energy mode. The second column of the table shows combinations of thresholds shown in Fig. 10 which have the worst performance degradation in the benchmark and the third column denotes averaged interval between unifications (only shift-

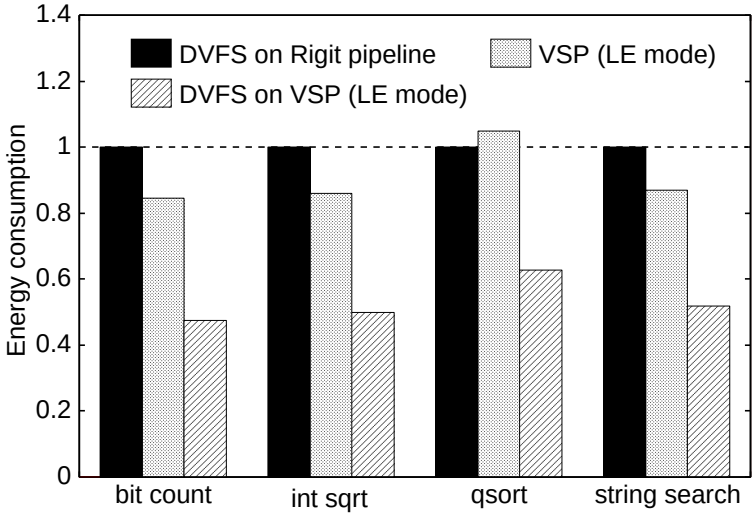


Figure 11: Measurement result of combining DVFS and VSP.

Table 2: Estimation for overhead caused by pipeline unification.

| benchmark | combination of thresholds | average interval between unifications | performance degradation |
|---|---|---|---|
| bit count | TH1 | 892 cycle | 0.8% |
| int sqrt | TH3 | 156 cycle | 4.5% |
| qsort | TH1 | 219 cycle | 3.2% |
| string search | TH4 | 142 cycle | 4.9% |

ing from high-speed mode to low-energy mode). Therefore, additional 7 cycles are required every averaged interval and this increases execution time by 0.8% to 4.9%.

In contrast, low-overhead technique hides the penalty. We confirmed that migration mode always operates for 4 high-frequency cycles, i.e. the new fetched instruction in FDR stage shown in Fig. 8 is not stalled. Therefore, the VSP unifies the pipeline stages with no penalty.

## 5.4 Hardware cost

Although we did not fabricate the base processor chip that does not apply a variable-depth pipeline architecture, we performed physical synthesis on the base processor to evaluate the additional hardware cost and energy of the VSP processor. The base processor has 456,022 transistors and VSP processor has 492,742 transistors. Therefore, modification for the VSP needs additional 8% hardware (36,720 transistors). The breakdown of the additional hardware is the following. We added approximately 2800 multiplexers to implement VSP, therefore,
$2,800\,multiplexers \times 12\,transistors = 33,600\,transistors$
are added. In addition, the depth changing controller increases 4705 transistors.

We compared energy consumption of the VSP with that of the base processor by using Synopsys Nanosim which is a fast-SPICE circuit simulator. According to the simulation result, the VSP consumes 1.5% to 5% larger energy than the base processor. These cost are reasonable compared with obtained effectiveness.

Table 3: Thresholds configuration.

| | $IPC_{HtoL}$ | $IPC_{LtoH}$ | $\#BR$ |
|---|---|---|---|
| TH0 | 15 | 18 | 6 |
| TH1 | 15 | 21 | 6 |
| TH2 | 18 | 21 | 6 |
| TH3 | 18 | 24 | 6 |

# 6 Case study

In this section, we discuss how to decide appropriate thresholds to the VSP processor. To reveal relation between combination of thresholds, execution time and energy consumption, we analyzed execution results on 88 *10-million SimPoints* [17] of four SPEC 2000 integer benchmarks (gzip, mcf, parser and bzip). Since our evaluation board can execute a program up to one million cycles, we used a MIPS R3000 simulator to perform fast-forwarding and to create a checkpoint. The VSP chip restores processor state from the checkpoint and starts execution for measuring. In this way, we ran the first 500,000 instructions of each SimPoint in order to evaluate execution time and energy consumption. According to our analysis, all 88 SimPoints are classified as two categories. Figure 12 and 13 show major SimPoints and minor SimPoints, respectively. In Fig. 12 and 13, each data indicates a SimPoint in a benchmark program (e.g., gzip_235 means the SimPoint which starts gzip program after skipping $235 \times 10\,million$ instructions). In these figures, TH0, TH1, TH2 and TH3 are configured as shown in Table 3, here $IPC_{HtoL}$ is condition to shift to low-energy mode, $IPC_{LtoH}$ and $\#BR$ are condition to make high-speed mode. Because a higher value of $IPC_{HtoL}$ means that the VSP processor is easier to unify pipeline stages and a lower value of $IPC_{LtoH}$ indicates that it is hard to shift back to deeper pipeline, the VSP tends to run under low-energy mode for a longer time as it goes TH0 to TH3. We experimentally decided these thresholds from around the average IPC under high-speed mode and these thresholds differ from thresholds which we used in Section 5.

74 SimPoints (84%) conform with the above theory. We show 16 representatives for the 74 major SimPoints in Fig. 12. As shown in Fig. 12, the energy consumption is gradually reduced as thresholds are changed as TH0 to TH3. For these applications, the VSP can adjust the energy consumption according to processor workload or required response time.

On the other hand, all remaining 14 SimPoints show the tendency shown in Fig. 13 which shows all minor trend of SimPoints. The VSP executes these SimPoints under high-speed mode most of the execution time except for fixed low-energy mode because the VSP achieves higher IPC in the minor SimPoints than in the major SimPoints. In particular, as for gzip_2 and bzip_5486 the VSP reaches almost the highest IPC even in high-speed mode. For this reason, the VSP cannot gradually reduce the energy consumption for these SimPoints by using the combinations of the thresholds. However, since a longer pipeline is effective for the minor SimPoints, pipeline unification obtains less effectiveness compared with the major SimPoints. In terms of energy consumption, only 35% energy can be reduced at a maximum in the minor SimPoints while the VSP can reduce at least 40% energy in the major SimPoints. This means that the minor SimPoints have a small impact in energy reduction, thus we consider that the behavior in the minor SimPoints is not a big matter for energy optimization.

For the major SimPoints, these thresholds are good to optimize the energy consumption. In contrast, for the minor SimPoints, although these thresholds cannot gradually reduce the energy consumption, the VSP cannot originally reduce a large energy consumption and have a small margin for energy optimization. Therefore, we conclude that deciding thresholds from around the average IPC is appropriate for energy optimization. Using thresholds such as shown in Table 3, the VSP can optimize the energy consumption in major programs.

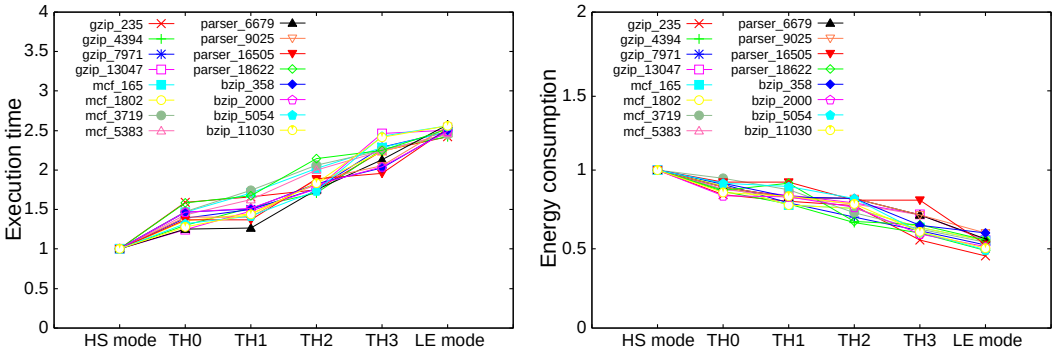


Figure 12: Major trend in 88 SimPoints.

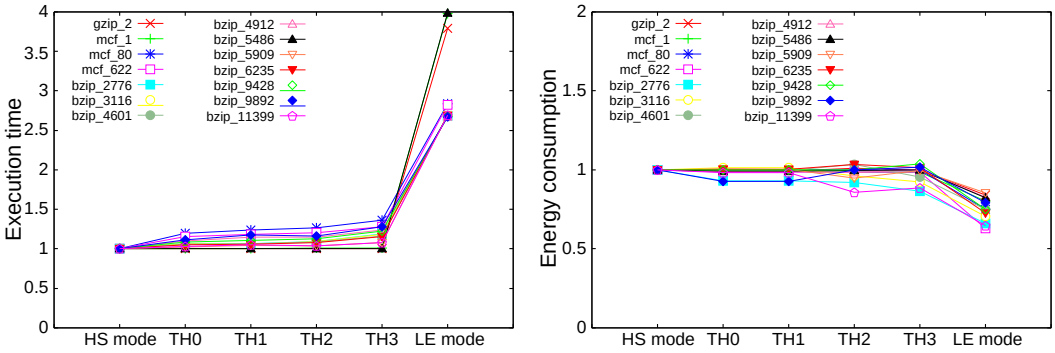


Figure 13: Minor trend in 88 SimPoints.

# 7 Summary and future work

In this paper, we presented the detailed evaluation results for the variable stages pipeline architecture which dynamically varies pipeline depth by using fabricated chip in 180 nm technology. Energy evaluation results show that the VSP processor reduces energy by 34% to 48% at fine-grained interval which DVFS cannot be used, and combining DVFS and VSP reduces energy by 38% to 52% compared with DVFS on rigid pipeline. We also evaluated required hardware cost to apply the VSP technique to base processor. As a result, we clarify that additional 8% hardware and 1.5% to 5% energy increase are required. These cost are reasonable compared with obtained effectiveness.

Finally, we revealed how to decide adjacent thresholds to optimize energy consumption by analyzing 88 SimPoints. Analysis result shows that the VSP processor follows the same behavior in major program phases, and therefore the VSP can optimize the energy consumption in many programs. Investigating more specific optimization is left for the future work.

# References

[1] I. Hong, M. Potkonjak, and M. A. Horowitz, On-line scheduling of hard real-time tasks on variable voltage processor, *Proceeding of International Conference on Computer-Aided Design*, pp. 653-656, November. 1998.

[2] J. Pouwelse, K. Langendoen, and H. Sips, Dynamic voltage scaling on a low-power microprocessor, *Proceeding of 7th ACM International Conference on Mobile Computing and Networking (Mobicom)*, July, 2001, pp. 251-259.

[3] I. Atsuki, Design Constraint of Fine Grain Supply Voltage Control LSI, *Proceeding of the 16 th Asia and South Pacific Design Automation Conference (ASP-DAC 2011)*, pp. 760-765, January, 2011.

[4] Y. Ichikawa, T. Sasaki, T. Hironaka, T. Kitamura, and T. Kondo, Low Energy Consumption by a Variable Stages Pipeline Technique, *Proceeding of International Technical Conference on Circuits/Systems Computers and Communications*, July, 2004, 0358, 6C1L-4.

[5] T. Sasaki, Y. Ichikawa, T. Hironaka, T. Kitamura, and T. Kondo, Evaluation of Low Energy and High Performance Processor using Variable Stages Pipeline Technique, *IET Journal of Computer and Digital Techniques*, April, 2008, Vol. 2, No. 3, pp. 230-238.

[6] T. Sasaki, K. Nomura, T. Nakabayashi, K. Ohno, and T. Kondo, Fine Grain Controller for Variable Stages Pipeline Processor *International Technical Conference on Circuits/Systems, Computers and Communications*, July, 2010, pp. 748-751.

[7] T. Nakabayashi, T. Sasaki, K. Ohno, and T. Kondo, Design and Evaluation of Variable Stages Pipeline Processor with Low Energy Techniques, *IET Journal of Computers and Digital Techniques*, January 2012, Vol. 6, Issue 1, pp. 43-49.

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

[9] H. Shimada, H. Ando, and T. Shimada, A Hybrid Power Reduction Scheme Using Pipeline Stage Unification and Dynamic Voltage Scaling, *Proceeding of the 9th IEEE Symposium on Low-Power and High-Speed Chips*, April 2006, pp. 201-214.

[10] J. Yao, S. Miwa, H. Shimada, and S. Tomita, A Dynamic Control Mechanism for Pipeline Stage Unification by Identifying Program Phases, *IEICE TRANSACTIONS on Information and Systems*, Vol. E91-D, No. 4, April 2008, pp. 1010-1022.

[11] J. Yao, S. Miwa, H. Shimada, and S. Tomita, A Fine-Grain Runtime Power/Performance Optimization Method for Processors with Adaptive Pipeline Depth, *Journal of Computer Science and Technology*, Vol. 26, No. 2, pp. 292-301, Mar. 2011. DOI 10.1007/s11390-011-1132-9.

[12] J. Koppanalil, P. Ramrakhyani, S. Desai, A. Vaidyanathan, and E. Rotenberg, A Case for Dynamic Pipeline Scaling, *Proceeding of International Conference on Compilers, Architecture, and Synthesis for Embedded Systems 2002*, October, 2002, pp. 1-8.

[13] T. Austin, E. Larson, and D. Ernst, SimpleScalar: An Infrastructure for Computer System Modeling, Computer, Vol. 35, No. 2, pp. 59-67, 2002.

[14] H. Onodera, A. Hirata, A. Kitamura, K. Kobayashi, and K. Tamaru, P2Lib:Process Portable Library and Its Generation System, *Journal of Information Processing*, vol.40, no. 4, pp. 1660-1669, April, 1999, (In Japanese).

[15] MiBench. http://www.eecs.umich.edu/mibench/

[16] Intel Corporation: Intel Atom Processor Z5xx Series Datasheet(2010)

[17] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder, Automatically Characterizing Large Scale Program Behavior, *10th International Conference on Architectural Support for Programming Languages and Operating Systems*, Oct. 2002, pp. 45-57.