Design of the Optimum Execution Stage of Embedded Processors

Masa-aki Fukase[1], Non-member

ABSTRACT

We describe the optimum design of an execution stage for embedded processors. The execution stage is the core of embedded processors to treat basic applications for multimedia, communication and control. Considering algorithms in running such applications, the design principle of embedded processors is to achieve reliability, precision, low power, high speed and fast throughput altogether. However, it is hard to sufficiently clear the design principle by existing microprocessor technology. The one of difficult issues is the mixed sequence of integer and FP (floating point) arithmetic instructions. The different latencies of these instructions in a processor's execution stage surely make the pipelined processing disturb and degrade throughput. The solution for this problem in this study is a wave-pipelined MFU (multifunctional unit) that is the multifunctionalization and wave-pipelining of different FUs (functional units). The high speed characteristic of wave-pipeling guarantees the reliability of processor behavior that is free from timing error. Thus, the waved MFU agrees well with overall trade-off design. In order to complement the shortage of the standard CAD tools, we explored the HS/SW (hardware/software) co-design for the waved MFU. Experimental results by using a 0.18-μm CMOS standard cell technology showed the usefulness of this approach.

Keywords: Embedded Processor, Execution Stage, Floating Point, Instruction Scheduling, Pipeline Disturbance, Wave-pipeline, Waved MFU, HW/SW Co-Design

1. INTRODUCTION

Multimedia, communication and control are the three major fields covered by embedded systems. They frequently employ applications like voice recognition, 3D graphics, image/vision processing, etc. Basic algorithms in running these applications are Fourier transformation, coding such as Huffman, Elias, etc. Then, one of the most important operations in processing such algorithms is FP (floating point) arithmetic. This is obvious from, for example, the Sobel filter, normalized correlation function and sequential similarity algorithm used by ASV (advanced safety vehicle) applications.

However, there exist fundamental issues in the execution of FP arithmetic by embedded processors. This conceives the trade-off between computational precision and power dissipation. Although FP format width tends to increase in order to achieve higher precision, it consumes much power. In view of power conscious compact design of embedded processors, power consumption has been continuously reduced by minimizing the bitwidth representation [1], [2], [3] and using the integer approximation of FP numbers [4].

Even though the best-suited bitwidth of FP format is fixed for some purpose, more fundamental approach is still required to achieve the overall design principle of embedded processor's execution stage. None of microprocessor techniques are sufficiently effective by themselves. In fact, existing microprocessor technology, for example, voltage scaling and the gating of power and clock, have both merits and demerits. They exhibit effectiveness after all when they are used as hybrid techniques. Actually, HW/SW (hardware/software) co-design is crucial for not only embedded processors but also the execution stage itself.

Considering the mixed arithmetic sequence of FP and integer numbers is a major factor of pipeline disturbance and the pipeline disturbance makes it hard to achieve high speed with low power, the one of practical approaches for the design principle of embedded processors is to sophisticate the adjustment of the latency and delay of the execution stage. Since the critical path of the execution stage depends on both data and arithmetic to be processed, HW/SW co-design is required for the delay tuning of the execution stage.

We describe in this article the optimum design of a power conscious highly performable execution stage suited for embedded processors [5]. The execution stage is a wave-pipelined or waved MFU (multifunctional unit) that is the combination of multifunctionalization and wave-pipelining of FUs (functional units). The waved MFU has the HW merit of power conscious high throughput and the SW merit free from instruction scheduling without pipeline disturbance. Although the waved MFU is not fastest, it

Manuscript received on December 12, 2012 ; revised on April 30, 2013.

Final manuscript received April 20, 2014.

[1] The author is with Graduate School of Science and Technology, Hirosaki University, Hirosaki 036-8561, Japan, E-mail: slfuka@eit.hirosaki-u.ac.jp

This work is supported in part by VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with Synopsys, Inc. and Cadence Design Systems, Inc.

is power conscious and highly performable. Thus, the waved MFU agrees well with overall trade-off design.

In spite of many superior aspects of wave-pipelining, it lacks established design procedure. Firstly, the standard CAD environment cannot be available for wave-pipelining because it is dedicated to regular pipelines. Secondly, the clocking scheme is quite different between the regular pipeline and the wave-pipeline. While the regular pipeline utilizes pipeline registers and thus the tuning of maximum delay is crucial for high performance, wave-pipelining does not use pipeline registers but depends on well-regulated propagation of waves [6].

This paper presents the HS/SW co-design approach for the development of the waved MFU and its employment in a Java compatible media pipeline. The combination of Java and FPU is practically important because it is frequently observed in embedded systems. The HW contribution in designing the waved MFU is to prepare efficient delay cells. This is carried out at both cell and transistor levels within the constraint imposed on standard cell-based ASIC [7]. The SW contribution is to show a code issue scheme so as to adjust timing between the waved MFU with longer latency and the source side of the waved MFU with shorter latency. The usefulness of the design of the waved MFU is shown by experimental results using a 0.18-μm CMOS standard cell technology.

2. EMBEDDED PROCESSORS

Before describing the waved MFU itself, this section overviews its target or embedded processors. It is made clear from the overview of the design of embedded processors that the execution stage is the core of embedded processors. Then, execution stages normally used in these processors, a basic execution stage and a MFU are described in view of hardware aspect. Besides, unfolding is briefly described that is one of software techniques for pipelined execution stages.

2.1 Design Principles

Fig. 1 overviews design principles of embedded processors, techniques and issues corresponding to individual principles. While various schemes for embedded processors improve each of power consumption, reliability, precision and throughput to some extent, they alternatively involve another issue. Although various microprocessor techniques are self consistent, none of them are sufficiently effective by themselves. Then, a solution preferable to regular techniques is also shown in Fig. 1.

Power conscious techniques shown in Fig. 1 are complementary to each other, each of which is insufficient to individually achieve low power. Since voltage scaling tends to increase latency, the clock cycle time, t, also increases and thus throughput decreases. Gating ranks third to suppressing clock speed and voltage scaling. The power gating saves static power [8] and the clock gating saves dynamic power [9].

The voltage scaling increases delay [10]. This violates the clock constraint, which is given by

$$\text{Critical path delay} \ < \ t. \tag{1}$$

The increase of t surely deteriorates throughput given by

$$\text{GIPS} \ = \ \text{giga instructions} \ / \ t. \tag{2}$$

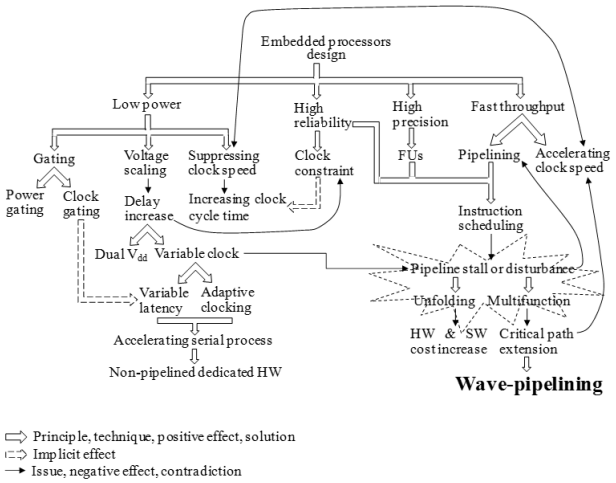

Fig.1: *Design principles and a solution of embedded processors.*

In order to avoid the increase of delay caused by the voltage scaling, a provisional solution is a variable clock technique that relaxes or varies the clock condition when a computation takes longer latency. It is a cycle time-variant clock scheme divided into adaptive clocking [11] and transient gating called variable latency [12], [13]. Although the clock gating and variable latency do not have the same motivation or the design principle to be cleared, they are based on the same control scheme of gating excepting the length of the clock-disable period. Commonsensically, the clock gating disables clocks for a long time during the unemployed period of a combinational logic circuit. On the other hand, the variable latency disables clocks transiently for several clocks.

Since the suitability of the variable clock technique is limited to serial processing, it is not important practically. The variable clock causes pipeline disturbance and reduces throughput. Thus, the burden of design principles of embedded processors is shifted onto execution stage. Complicated pipeline controls do not always enhance throughput. One of provisional solutions for the pipeline disturbance is to introduce MFU. This is free from instruction scheduling, but the scale up due to multifunctional-

ization brings about critical path extension in turn. Doubtlessly, this is a formidable effect in view of clock speed because the clock speed is generally determined as closely to critical path delay as possible. The critical path extension resulting from the multifunctionalization is apt to cause timing error in which critical path delay exceeds clock cycle time.

In order to achieve the reliability of processor behavior free from timing error, wave-pipelining is very effective due to its high speed characteristic [14]. Thus, the practical solution for the design of an optimum execution stage is wave-pipelining. This satisfies the design principles altogether. In spite of many characteristics superior to regular pipelines, wave-pipelining has not been a major processor technique. This is due to the lack of established design procedure. The HW/SW co-design of the waved MFU is described in Section 3.

2.2 Basic Execution Stage

One of regular execution stages is a basic execution stage shown in Fig. 2. Such a bottom-up structure at the gate-level is common in processors. Actually, the development of efficient arithmetic units like adders [2], [12] and multipliers [3] are developed and placed independently. Case studies of spuriously merging independent unifunctional components are ALUs of MIPS and UltraSparc processors, scalar processing units of CRAY-I, NEC SX, etc.

In view of high speed clock, Fig. 2 schemes variable clocking. The variable clock approach releases clock design from the worst case criterion. Considering the worst condition of critical path rarely happens, the variable clock approach employs smaller t than maximum latency.

Fig. 2 (a) shows the organization of the basic execution stage together with datapath and the clocking circuit. Short-latency operands generalize integer arithmetic instructions and long latency operands do floating point arithmetic instructions. Then, FU_A and FU_B are the generalization of IU (integer unit) and FPU (FP unit), respectively.

The output frequency of the variable clock source,

$$f_{var} = {t_{var}}^{-1} \tag{3}$$

is set as follows. Firstly, the frequency of basic clocking,

$$f_{basic} = {t_{basic}}^{-1} \tag{4}$$

is given by Eqs. (5) and (6), keeping the clock constraint to guarantee the reliability of processor behavior as shown in Fig. 1.

$$\text{Clock cycle time} \geq \text{critical path delay} \tag{5}$$

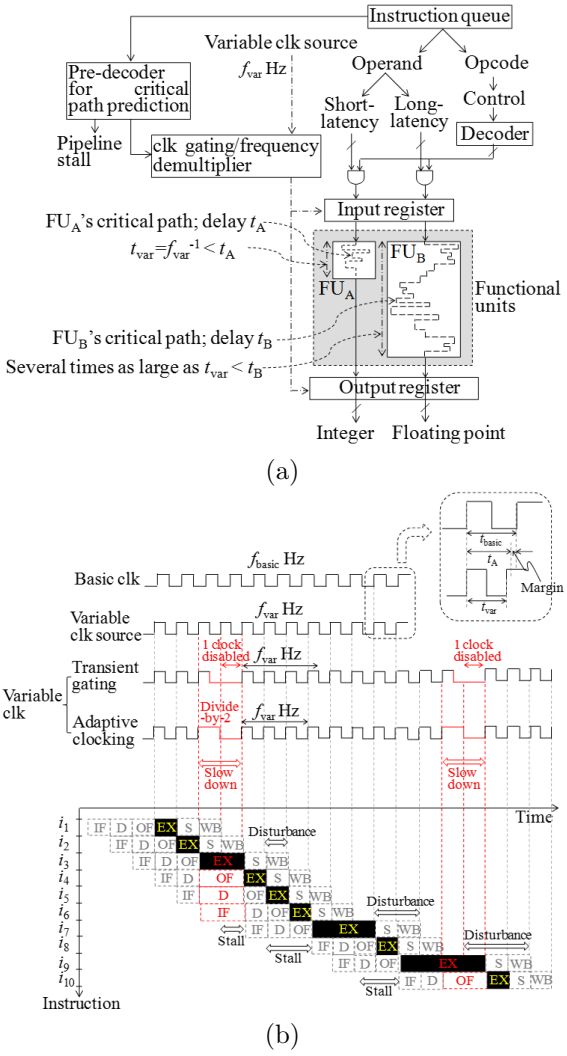

Fig.2: *Basic execution stage (a) Organization (b) Timing chart.*

$$t_{basic} \geq t_A. \tag{6}$$

Taking off the clock constraint so as to be

$$f_{var} > f_{basic}, \tag{7}$$

$$f_{var} = {t_{var}}^{-1} \tag{8}$$

is set up for FU_A by

$$t_{var} < t_A. \tag{9}$$

Similarly to Eq. (9),

$$\text{Several times as large as} t_{var} < t_B \tag{10}$$

is set up for FU_B. After all, t_{var} is given by

$$t_{var} < t_A \leq t_{basic} \tag{11}$$

from Eqs. (6) and (9).

The essence of the variable clock approach is the countermeasure of the worst case. When propagation delay violates the timing constraint of Eq. (11), the reliability is achieved by the slowdown of the clock speed. The clock slowdown occurs when instructions issued from the instruction queue activate paths in the following.

- Paths in FU_A with delays larger than t_{var}.
- Paths in FUB with delays larger than several times as large as t_{var}.

The necessity of the clock-slowdown is dynamically predicted by the unit of pre-decoder for critical path prediction. The one of the output of the pre-decoder is connected to the clock gating unit implementing the transient gating or to the frequency demultiplier implementing the adaptive clocking. The other output of the pre-decoder is connected to a pipeline stall unit.

Fig. 2 (b) exemplifies the timing chart of a 6-stage instruction pipeline whose EX (execution) stage is the basic execution stage shown in Fig. 2 (a). The extent and the effect of the clock-slowdown are also shown in Fig. 2 (b). Although the rough estimation of the clock slowdown is f_{basic} from Eq. (6), the clock speed less than f_{basic} is also allowable in the case of FUs shown in Fig. 2 (a), where the inputs of FU_A and FU_B are not triggered by different clocks issued from the same clock source. Practically, the slowdown is carried out by one clock cycle.

As for the effect of the clock-slowdown, the transient slowdown of clocking surely causes pipeline disturbance. The pipeline disturbance is such phenomena that the pipeline's output, WB (write back) stage, is made delayed owing to some reason. This results in

$$\text{IPC (instructions/clock cycle)} < 1. \tag{12}$$

Although the variable clock scheme is effective for the speed-up of normal pipelining, it cannot avoid the deterioration of IPC. Corresponding to the disturbance at the output-side, the input-side, IF (instruction fetch) stage, is stalled.

Pipeline stall or disturbance is also brought about by the coexistence of FUs with different delays. In this case, the adjustment of waiting period by using re-ordering and unfolding techniques is an effective solution. The latter is to unfold loop structures according to the pipeline degree of FPU. However, such approaches demand undesirable SW and HW costs. This is not suited to the design of compact embedded processors.

In Fig. 2 (b), the abscissa is notched by the clock cycle and the ordinate shows instruction sequence. i1, i2, and i8 are instructions that pass FU_A within t_{var}. i7 is an instruction whose delay at FU_B is less than several times as large as t_{var}. i3 is an instruction whose delay at FU_A exceeds t_{var}. In this case, the clock speed is reduced for EX by half. i_9 is an instruction with delay at FU_B larger than several times as large as t_{var}. In this case, the last clock of EX is made slowdown by 1 cycle. i_4, i_5, i_6 and i_{10} are instructions that are affected by the slowdown caused by previous instructions. Excepting FU_A, stages they pass are affected by such phenomena.

2.3 Multifunction Approach

Fig. 3 shows the multifunctionalization of the basic MFU shown in Fig. 2 (a). The employment of MFU in the EX stage is a possible way to avoid pipeline disturbance. Since MFU is a single structure or flat at the gate level description, it takes the same latency in executing any function. Thus, MFU is free from scheduling and pipeline disturbance.

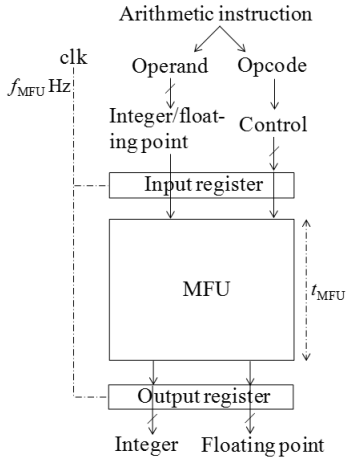

Fig.3: *Basic MFU.*

On the other hand, MFU surely results in degrading clock speed because it covers FP arithmetic with longer latency. Thus, Eq. (13) holds.

$$f_{basic} > f_{MFU}. \tag{13}$$

In addition, multifunctionalization causes the scale up of MFU's circuit. Thus, the simply merging of regular pipelines does not always promise the total enhancement of processor performance.

3. WAVED MFU

The two factors of pipeline stall or disturbance described in Section 2.2 are the transient slowdown of variable clocking and the coexistence of FUs with different delays. The waved MFU clears at least the

latter problem, the pipeline disturbance due to the coexistence of FUs with different delays. As is described in Section 4, the waved MFU is also superior to conventional clocking schemes including the variable clocking in view of throughput and power consumption.

3.1 Wave-pipeline

Fig. 4 summarizes the design flow of the wave-pipeline compared with that of the regular pipeline. Here, a wave implies a set of signals that propagate like a wave packet. Then, a wave-pipeline is a sort of a fault tolerant combinational logic circuit that propagates plural number of waves without collision. In order to realize such phenomena in a combinational logic circuit, the design principle of the wave-pipeline is as follows.

- The clock speed is not related to logic depth, but determined by the difference between the critical path delay and the minimum path delay of the waved circuit.
- The delay tuning does not use flip flops necessary for the design of regular pipelines, but relies on the adjustment of delay variation within a combinational logic.

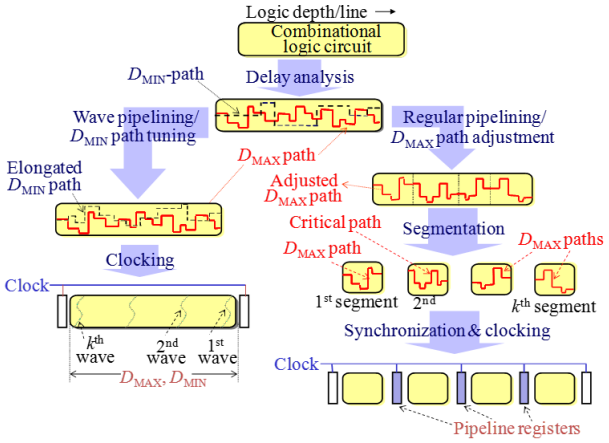

Fig.4: *Wave-pipeline vs. regular pipeline.*

Table 1 summarizes characteristic aspects of the wave-pipeline compared with the regular pipeline. Wave-pipelining has potential features of low power high speed circuit operation [14]. The wave-pipelining is useful because it recovers the reduction of clock speed caused by the scale up of a multifunctionalized circuit.

In spite of many characteristics described above, wave-pipelining has not been a major processor technique. This is due to the lack of benefits of design environment as is summarized in the following.

- CAD tools: Today's processor techniques and CAD tools have been dedicated to the development of regular pipelines. While they receive such benefit, the wave-pipelining suffers with the lack of dedicated tools.
- Chip framework: Standard cell-based ASIC is preferable for wide range application of embedded processors. However, it has not always provided with delay adjusting capability specific for wave-pipelining.
- Tuning techniques: It is one of key techniques of both regular and wave- pipelines. There have been many works about the sizing of CMOS gate and buffer [15], transistor sizing [16]. However, the viewpoints of these works have been the delay, power, drivability and clock distribution of regular circuits. They are not always sufficient for the specific delay tuning required for wave-pipelining.

Table 1: *Characteristics and issues of the wave-pipeline.*

Aspect	Wave	Regular
Clock	Fast	Moderate
Power	Low	Moderate
Area	Small	Moderate
Register	Needless	Need
CAD tool	Inopportune	Opportune
Designing effort	Cumbersome	Moderate

3.2 HW/SW Co-Design Procedure

Fig. 5 shows the structure of the waved MFU and its employment in a Java compatible media pipeline. The combination of Java and FPU is practically important because it is frequently observed in embedded systems.

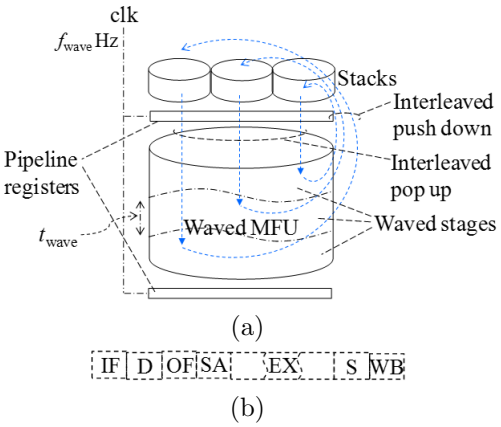

Fig.5: *Waved MFU (a) Structure (b) 7-stage Java compatible media pipeline.*

As shown in Fig. 6, the waved MFU is a hybrid microprocessor technique that combines multifunctionalization and wave-pipelining of FUs. The waved MFU receives the high speed characteristic of wave-pipelining. This is because MFU shown in Fig. 3

and the waved MFU shown in Fig. 5 (a) have basically the same logic length from Fig. 4 and Fig. 6. Thus, the wave-pipelining recovers the reduction of clock speed caused by MFU and the following relation holds for the wave-pipeline's clock speed, f_{wave} and MFUs clock speed, f_{MFU}.

$$f_{wave} > f_{MFU}. \tag{14}$$

On the other hand, Eq. (15) holds because the waved MFU is derived from not Fig. 2 (a) but Fig. 3 and IU is also scaled up during the multifunctionalization.

$$f_{basic} > f_{wave}. \tag{15}$$

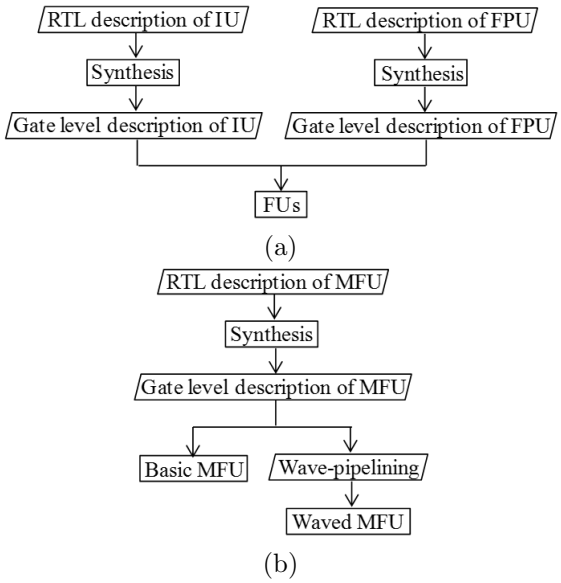

Fig.6: *Design flow of execution stages (a) Basic execution stage (b) MFU and waved MFU.*

The reason why stacks are provided in Fig. 5 (a) is because embedded processors follow the stack machine in view of following reasons.

- The stack machine has compact structure, little dependency on compilers.
- Embedded processors are provided with Java that utilizes stack operation.

Then, the reason why the stacks are parallelized in Fig. 5 (a) is because the waved MFU takes longer latency compared with source and destination units. In case of Fig. 5, the wave degree is 3 and 3 stacks are provided. The parallel stacks avoid pipeline disturbance caused by timing discrepancy between the 1-clock stack and the several-clock waved MF. If single stack is used, stack access must be stopped during the second and the third clocks when the waved MFU processes the data of the first clock. Fig. 5 (b) is a 7-stage Java compatible media pipeline whose EX stage is a waved MFU shown in Fig. 5 (a).

Fig. 7 summarizes the HW/SW design procedure of the waved MFU to achieve HW-based instruction scheduling and pipeline disturbance free parallelism. HW design is assigned to the steps L1 and L2. The remaining steps L3 to L6 are the part of SW design.

L1: Merge FUs.
L2: Wave-pipelining the resultant MFU.
L3: Provide the source side of the waved MFU with the same stacks as the degree of the waved MFU.
L4: Provide stack-related codes for individual stacks.
L5: Support the parallelization of stacks by a compiler that extracts ILP.
L6: Interleave those codes in the issue from instruction cache.

Fig.7: *HW/SW design procedure of a waved MFU.*

3.3 HW Design

The practical method of adjustment needed for wave-pipelining is rough tuning by buffer insertion in combination with fine tuning at transistor level design. We focus in this section on the rough tuning and delay cells efficient for the rough tuning. In order to achieve longer delay with lower power consumotion, we have investigated delay cells by utilizing ASIC. Within the constraint like the limitation of cell height imposed on this chip framework, delay cells have been varied at both cell and transistor levels. The cell structure is varied by adding another cells or capacitors [17]. The transistor sizing is done by decreasing channel width W and increasing channel length L of additional cells.

One of efficient delay cells is shown in Fig. 8 [7]. This inserts a pair of channel inverters whose channel length has been expanded. Fig. 8 (c) shows delay and power vs. channel length characteristics. The effect of transistor sizing is clear from the fact that delay is proportional to channel length. Yet, power consumption is almost constant.

3.4 Experimental Result

The two types of cell libraries are provided in the evaluation of the waved MFU by using a 0.18- m CMOS process. The one is an original standard cell library. The other is an improved standard cell library by adding a set of efficient delay cells shown in Fig. 8 into the original library. This is similar to functional ECO (Engineering Change Order) [18].

By using these libraries we have designed a waved MFU. Dominant characteristics are summarized in Table 2 compared with another waved MFU made by using the original library that does not contain efficient delay cells. The two waved MFUs have the same clock speed and functions. The 13-15% reduction of occupied area and power dissipation is due to long delay ECO cells.

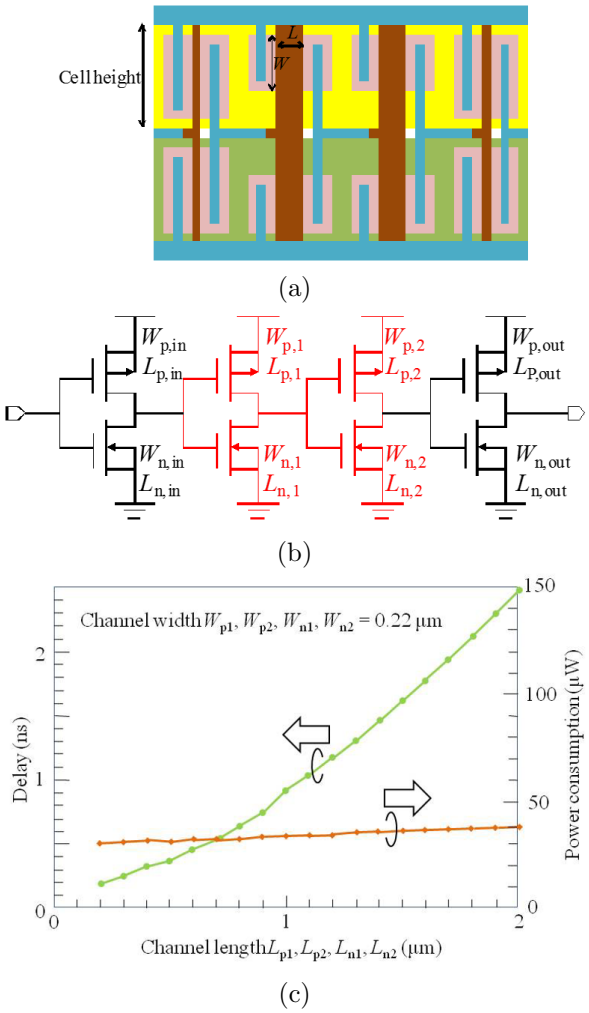

Fig.8: *An efficient delay cell with long channel length inverters (a) Cell layout (b) Transistor circuit (c) Channel length dependency of delay and power consumption.*

Table 2: *Characteristics of the Waved MFUs.*

	Library improved by using efficient delay cells	Original library without efficient delay cells
Occupied area (mm^2)	0.127	0.15
Power dissipation (mW)	28.1	32.4
Clock (MHz)	200	
Functions	Integer: 8-bit add, sub., mul. Floating point: 8-bit add, sub., mul. Logic: 4-bit shifts, compare, jump	

4. DISCUSSION

In order to compare the waved MFU with other types of execution stages, Table 3 summarizes the affinity of these components for different types of clocking. The variable clocking dynamically carried out as shown in Fig. 2 is available for the basic execution stage composed of bottom-up FUs and basic MFU. The variable clocking is also available for the waved MFU because slowing down the clock speed adjusted during the design as shown in Fig. 6 (b) does not give any effect for waves collision and thus reliability. However, the combination of the variable clocking and the waved MFU is not so beneficial. It loses the merit of the waved MFU free from pipeline disturbance.

Table 3: *Structure vs. Clocking of the Execution Stage.*

Clocking \ Structure		Bottom-up FUs	MFU	
			Basic	Waved
Steady clocking		○	○	○
Variable clocking	Transient gating	○	○	△
	Adaptive clocking	○	○	△

Table 4 summarizes the evaluation of the waved MFU compared with other types of execution stages. As for bottom-up FUs, both steady clocking and variable clocking are taken into account of. The basic and waved MFUs are evaluated in the case of steady clocking for the sake of clear discussion. As is described above, the combination of the variable clocking and the waved MFU is not practical.

Table 4: *Comparative Evaluation of Execution Stages.*

Design principles		Bottom-up FUs			MFU		Remarks
		Steady clocking	Variable clocking		Basic	Waved	
			Transient gating	Adaptive clocking			
HW	Clock speed	f_{basic}	f_{var}		f_{MFU}	f_{wave}	$f_{var} > f_{basic} > f_{wave} > f_{MFU}$
	Throughput	T_{basic}	T_{var}		T_{MFU}	T_{wave}	$T_{var} < T_{basic} < T_{MFU} < T_{wave}$
	Power dissipation	P_{basic}	P_{var}		P_{MFU}	P_{wave}	$P_{basic} > P_{var} > P_{MFU} > P_{wave}$
	Reliability	Adequate					
SW	Instruction scheduling	Necessary	Free				
	Pipeline disturbance	Present				Absent	

The evaluation for the clock speed shown in Table 4 is derived from Eqs. (7), (14) and (15). The relation between the throughput of the basic execution stage, T_{basic} and the throughput of the variable clock execution stage, T_{var} in the following is derived considering that the variable clocking enhances the clock speed but the pipeline disturbance is inevitable.

$$T_{var} < T_{basic}. \tag{16}$$

The relation between T_{basic} and the throughput of

MFU, T_{MFU} is given by

$$T_{basic} < T_{MFU}. \quad (17)$$

This reflects the fact that the multifunctionalization does not change the logic length or the maximum logic depth. The relation between T_{MFU} and the throughput of the wave-pipeline, T_{wave} is basically derived from the relation of the clock speed shown in Eq. (14).

$$T_{MFU} < T_{wave}. \quad (18)$$

Then, the relation between Tvar and TMFU is evaluated from the presence or absence of pipeline disturbance.

The power dissipation is closely related to throughput. Higher throughput makes the processing period shorter and thus the mean value of power dissipation is smaller. The most important point from Table 4 is that the waved MFU is power conscious highly performable, though it is not fastest.

5. CONCLUSION

We have studied the optimum design of a power conscious highly performable execution stage because it is the core of embedded processors. The most preferable execution stage is a wave-pipelined MFU. Although the waved MFU is not fastest, it is power conscious highly performable. However, the wave-pipelining lacks the established design procedure compared with the design of regular pipelines.

We have investigated in this study HS/SW codesign approach for the waved MFU. The HW contribution is the design of efficient delay cells by using standard cell-based ASIC. Within the constraint imposed on this chip framework, it has been made clear that a cell inserted a pair of channel inverters has desirable delay proportional to channel length with almost constant power dissipation. By adding this cell into a 0.18-μm CMOS standard cell library, we have made a waved MFU that reduces occupied area and power dissipation by 13-15%. Then, the SW contribution is the establishment of a code issue scheme so as to fully utilize the waved MFU.

The one of the next steps of this study is to achieve experimental data that shows superiority of the waved MFU to other types of execution stages. Then, the other step is to apply the waved MFU to the ubiquitous processor, HCgorilla [19]. The architecture of HCgorilla follows multicore and multiple pipelines. These are Java compatible media pipelines and cipher pipelines. Thus, the waved MFU developed in this study is really useful.

References

[1] J. Y. F. Tong, D. Nagle, and R. A. Rutenbar, "Reducing Power by Optimizing the Necessary Precision/Range of Floating-Point Arithmetic," *IEEE Trans. on VLSI Syst.*, Vol. 8, No. 3, pp. 273–286, June 2000.

[2] N. Zhu, W. L. Goh, W. Zhang, K. S. Yeo, and Z. H. Kong, "Design of Low-Power High-Speed Truncation-Error-Tolerant Adder and Its Application in Digital Signal Processing," *IEEE Trans. on VLSI Syst.*, Vol. , No. 11, pp. 1621–1624, Nov. 2010.

[3] C.-H. Chang and R. K. Satzoda, "A Low Error and High Performance Multiplexer-Based Truncated Multiplier," *IEEE Trans. on VLSI Syst.*, Vol. 18, No. 12, pp. 1767–1771, Dec. 2010.

[4] C.-Y. Kao and Y.-L. Lin, "A Memory-Efficient and Highly Parallel Architecture for Variable Block Size Integer Motion Estimation in H.264/AVC," *IEEE Trans. on VLSI Syst.*, Vol. 18, No. 6, pp. 866–874, June 2010.

[5] M. Fukase, A. Kurokawa, K. Ichinohe, and T. Takaki, "Optimum Design of the Execution Stage of Embedded Processors," *Proc. of ISCIT 2012*, pp. 538-542, Oct. 2012.

[6] W. P. Burleson, M. Ciesielski, F. Klass, and W. Liu, "Wave-Pipelining: A Tutorial and Research Survey," *IEEE Trans. on VLSI Systems*, vol. 6, no. 3, pp. 464-474, Sep., 1998.

[7] A. Kurokawa, T. Takaki, and M. Fukase, "Efficient Delay Cells for Wave Pipelined Multifunctional Unit," *Proc. of SASIMI 2012*, pp. 121-126, Mar. 2012.

[8] Y. Lee, D.-K. Jeong, and T. Kim, "Comprehensive Analysis and Control of Design Parameters for Power Gated Circuits," *IEEE Trans. on VLSI Syst.*, Vol. 19, No. 3, pp. 494–498, Mar. 2011.

[9] K. Ando, "Non-Volatile Devices: Can be Normally-Off Computer Realized?," *Jour. of IEICE*, Vol. 93, No. 11, pp. 913–917, Nov. 2010.

[10] S. Chandra, A. Raghunathan, and S. Dey, "Variation-Aware Voltage Level Selection," *IEEE Trans. on VLSI Syst.*, Vol. 20, No. 5, pp. 925–936, May. 2012.

[11] S. Ghosh, D. Mohapatra, G. Karakonstantis, and K. Roy, "Voltage Scalable High-Speed Robust Hybrid Arithmetic Units Using Adaptive Clocking," *IEEE Trans. on VLSI Syst.*, Vol. 18, No. 9, pp. 1301–1309, Sept. 2010.

[12] Y. Chen, H. Li, C.-K. Koh, G. Sun, J. Li, Y. Xie, and K. Roy, "Variable-Latency Adder (VL-Adder) Designs for Low Power and NBTI Tolerance," *IEEE Trans. on VLSI Syst.*, Vol. 18, No. 11, pp. 1621–1624, Nov. 2010.

[13] Y-S. Su, D.-C. Wang, S.-C. Chang, and M. Marek-Sadowska, "Performance Optimization Using Variable-Latency Design Style," *IEEE Trans. on VLSI Syst.*, Vol. 19, No. 10, pp. 1874–1883, Oct. 2011.

[14] F. Klass and M. J. Flynn, "COMPARATIVE

STUDIES OF PIPELINED CIRCUITS," Stanford University Technical Report No. CSL-TR-93–579, July 1993.

[15] K. S. Lowe and P. G. Gulak, "A Joint Gate Sizing and Buffer Insertion Method for Optimizing Delay and Power in CMOS and BiCMOS Combinational Logic," *IEEE Trans. on CAD of IC and Syst.*, Vol. 17, No. 5, pp. 419-434, May 1998.

[16] M. Borah, R. M. Owens, and M. J. Irwin, "Transistor Sizing for Low Power CMOS Circuits," *IEEE Trans. on CAD of IC and Syst.*, Vol. 15, No. 6 , pp. 665–671, June 1996.

[17] R. Samanta, J. Hu, and P. Li, "Discrete Buffer and Wire Sizing for Link-Based Non-Tree Clock Networks," *IEEE Trans. on VLSI Syst.*, Vol. 18, No. 7, pp. 1025–1035, Jul. 2010.

[18] H.-T. Chen, C.-C. Chang, and T. T. Hwang, "Reconfigurable ECO Cells for Timing Closure and IR Drop Minimization," *IEEE Trans. on VLSI Syst.*, Vol. 18, No. 12, pp. 1686–1695, Dec. 2010.

[19] M. Fukase, "A Ubiquitous Processor Embedded With Progressive Cipher Pipelines," *International Jour. of Multimedia Technology*, Vol. 3, No. 1, pp. 31–37, Mar. 2013.

Masa-aki Fukase received the B.S., M.S., and Dr. of Eng. Degrees in Electronics Engineering from Tohoku University in 1973, 1975, and 1978, respectively. He was Research staff member from 1978 to 1979 at The Semiconductor Research Institute of the Semiconductor Research Foundation. He was Assistant Professor from 1979 to 1991, and Associate Professor from 1991 to 1994 at the Integrated Circuits Engineering Laboratory of the Research Institute of Electrical Communication, Tohoku University. He has been Professor of computer engineering since 1995 at the Faculty of Science and Technology, Hirosaki University. He served as the Director of the Hirosaki University C&C Systems Center from 2004 to 2012. He has been the Representative of Hirosaki University R&DC of Next Generation IT Technologies from 2008 to 2014. His current research activities are mainly concerned with the design, chip implementation, and application of power conscious highly performable ubiquitous processors.