A Ubiquitous Processor Built-in a Waved Multifunctional Unit

Masa-aki Fukase[1] and **Tomoaki Sato**[2], Non-members

ABSTRACT

In developing cutting edge VLSI processors, parallelism is one of the most important global standard strategies to achieve power conscious high performance. These features are more critical for ubiquitous systems with great demands for multimedia mobile processing. Then, one of most important issues for ubiquitous systems is instruction scheduling, because floating point units indispensable for multimedia mobile applications take longer latency than integer units. Although software parallelism has been inevitable to fully utilize hardware parallelism between regular scalar units, it has been really awkward. Thus, we describe in this article a double scheme to achieve instruction scheduling free ILP (instruction level parallelism) and apply the double scheme to a ubiquitous processor HCgorilla we have so far developed. The double scheme is the multifunctionalization of scalar units and making a resultant multifunctional unit (MFU) wave-pipeline. The multifunctionalization frees the instruction scheduling, and the wave-pipelining recovers the reduction of clock speed to be caused by the scale up of a multifunctional circuit. HCgorilla built-in the waved MFU is promising for wide-range dynamic ILP at a rate higher than regular processors.

Keywords: Ubiquitous processor, floating point, instruction scheduling, multifunctional unit, wave-pipelining

1. INTRODUCTION

Considering an increasing number of cellar phones and PDAs, multimedia computing on ubiquitous network is one of most remarkable trends of next generation information technologies and communications. Then, some strategies to achieve power conscious high performance and high precision computing is the essence to cover various requirements for mobility, usability, security, reality, real time responsibility, etc. Practically, an overall solution for these requirements is to take hardware approach and VLSI implementation. Yet, software dependent supervision is still necessary to deal with sophisticated requirements for ubiquitous environment. Thus, the development of powerful hardware in conjunction with software is crucial for further spread of ubiquitous systems.

Since VLSI trend in recent years is to exploit not higher speed but power conscious high performance, parallelism is really the global approach for the development of contemporary VLSI processors. Parallelism is crucial in view of both hardware and software aspects. Hardware parallelism is covered by multicore and multiple pipeline architectures. Even ubiquitous systems have introduced multicore architectures in recent years. In order to fully utilize hardware parallelism, software support like TLP (thread level parallelism) and ILP (instruction level parallelism) is also inevitable. Although much emphasis has been put on TLP, ILP is still an important subject even for multithreaded processors. Actually, multithreaded processors include scalar units that execute arithmetic instructions in parallel.

Since floating point (FP) instructions indispensable for multimedia applications take longer time than integer instructions, instruction scheduling is inevitable for the extraction of ILP from these instructions. But, software tools for instruction scheduling are not suited to ubiquitous platforms, because they need larger computer resources. Thus, a hardware approach for the implementation of FP operations is a matter of urgent subject. However, FP hardware generally occupies large area and consumes much power. Compactness is indispensable to keep the mobility of ubiquitous devices. A compact FPU (FP number processing unit) we developed has only 5 stages, and it works at 400 MHz [1]. Yet, it still requires awkward instruction scheduling so far as it is used in conjunction with IU, because it takes less latency than the resultant 5-stage FPU.

In order to progress the overall status of ubiquitous and processor techniques, it is really worthwhile to exploit hardware parallelism free from awkward instruction scheduling [2]. Thus, we have explored a double scheme to solve both issues, instruction scheduling and scale up [3]. The double scheme is the multifunctionalization and wave-pipelining of scalar units. The essence of the double scheme is to complement the drawback of multifunctionalization with wave-pipelining. Incorporating a multifunctional unit (MFU) in the execution (EX) stage of an instruction pipeline frees the instruction scheduling, because it takes the same latency to execute any func-

[1]The author is with Graduate School of Science and Technology, Hirosaki University, Hirosaki 036-8561, Japan Tel.+81-172-39-3630, Fax:+81-172-39-3645, E-mail: slfuka@eit.hirosaki-u.ac.jp.

[2]The author is with C&C Systems Center, Hirosaki University, Hirosaki 036-8561, Japan Tel.+81-172-39-3723, Fax:+81-172-39-3722, E-mail: tsato@cc.hirosaki-u.ac.jp.

tion. Then, the reduction of clock speed to be caused by the scale up of a multifunctional circuit is recovered by wave-pipelining, because the clock speed of a wave-pipeline is determined by the difference between the critical path delay and the minimum path delay of the waved circuit. We have further exploited the application of the waved MFU to HCgorilla, which we have so far developed for ubiquitous systems [4].

After the brief survey of wave-pipelining, we describe in this article the design of a waved MFU more in detail to achieve instruction level hardware parallelism, instruction scheduling free pipeline, and power conscious speedup for media processing. Then, the waved MFU is used to improve the previous version of HCgorilla. The chip implementation of the improved HCgorilla is done by using a 0.18-μm standard cell CMOS process. The improved HCgorilla is promising for wide-range dynamic ILP at a rate higher than regular processors.

2. WAVE-PIPELINING

Wave-pipelining is a unique control scheme of signal propagation within a processor that does not use regular pipeline registers [5], [6]. It exploits high clock frequency and high throughput by launching as many data as possible into CLBs (combinational logic blocks) under the restriction that they do not conflict. Since such sophisticated control is done not by inserting pipeline registers among CLBs but by tuning CLBs themselves, wave-pipelining brings low power dissipation as well. Mostly wave-pipelines have been so far applied for simple unifunctional circuits such as adders [7], multipliers [8], counters [9], and DRAM controls [10].

2.1 Wave-Pipelining Techniques

Since the wave-pipeline is the potential rival of the regular pipeline, we exemplified a wave-pipeline in a multifunctional unit or ALU [11]. But, it was rare. The lack of sufficient power of hitherto CAD tools was a likely reason why wave-pipelines had been applied out of complicated circuits. Wave-pipelines have not been the target of design tools developed for regular pipelines.

The disappointing tendency that wave-pipelines have not so far applied widely has been mainly due to the lack of mature design tools. Except the need of manual dependent design process, the wave-pipeline has no potential drawback. Thus, we have explored several design techniques and application dedicated to wave-pipelines [12].

2.2 Scale Up

Wave-pipelining comprehends scale up issues. They take two aspects. The one is concerned with delay tuning. Since the tuning of wave-pipelinefs clock speed is done by approaching the shortest path delay as closely as possible to the critical path delay, the shortest path surely elongates. In addition, this is repeated for the second shortest path and so on. Yet, the scale up of CLB circuits due to delay tuning is sufficiently cancelled by the removal of pipeline registers. They are really area-consuming.

The other aspect of wave-pipeline vs. scale up issue is concerned with multifunctionalization. A possible way to release instruction scheduling in running processors is to merge the parallel structure of regular pipelines and to make them completely multifunctional. This surely executes every function with the same latency. However, the increase of circuit scale accompanied with the multifunctionalization elongates the critical path. This results in the degradation of clock speed. Thus, the simply merging of regular pipelines does not always promise the total enhancement of processor performance. In order to completely unify hardware units without deteriorating clock speed, wave-pipelining is really promising. This is discussed more in detail in the next chapter.

2.3 Heterogeneous Pipeline

Although wave-pipelining is a promising control scheme of processors, a pessimistic view was taken about wave-pipelining the entire region of a processor. This is because it was inferred that general purpose registers cannot be removed and some of them would interrupt wavesf propagation [13]. Practically considering the status of wave-pipelining techniques, it is expedient to introduce wave-pipelines into usual pipelines in part. Actually, a hybrid approach was taken for a 3-segment router where each segment was wave-pipelined [14]. In addition, a 14-segment microprocessor, ULTRASPARC-III was developed by wave-pipelining the second and third instruction fetch segments [15].

We studied the wave-pipelining of every segment of a 12-segment processor [16]. A heterogeneous pipeline is an instruction pipeline whose segments are wave-pipelined as shown in Fig. 1. In this case, the 3rd and 5th stages are wave-pipelined by tuning without registers. Pipeline registers are allocated among segments. Making the entire region of a processor heterogeneous promises higher throughput, higher clock frequency, and less power dissipation. A usual instruction pipeline is a special case of a heterogeneous pipeline constructed by 1-wave segments. The strong point of a heterogeneously pipelined processor is discussed more in detail in the next chapter.

3. WAVED MFU

3.1 Regular MFU

Integer and FP instructions are frequently used in multimedia applications. Especially, FP expressions are crucial for the expression of physical phenomena

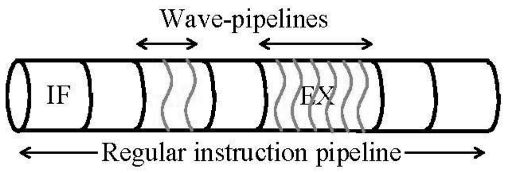

Fig.1: *Heterogeneous pipeline model.*

such as voice recognition, 3D graphics, image/vision processing, etc. A normalized correlation factor is one of actual FP expressions used in stereo matching, which is a basic obstacle detection algorithm for the image processing of ASV (advanced safety vehicle) and ITS (intelligent transport system).

In order to achieve the power conscious high performance of multimedia computing on ubiquitous platforms, we developed a compact FPU [1]. A FP format applied to this FPU is IEEE 754 compatible except the bitwidth representation. The FP data width is fixed to 16 bits according to the design principle of FPU, that is, power conscious high speed with sufficient precision and universality. The dominant factor of the power and precision is the bitwidth of FP data. Examining the necessary resolution and range of FP arithmetic used for embedded applications, it was pointed out that 9 mantissa bits and 6 to 7 exponent bits are sufficient [17]. Reducing the bitwidth of FPU as short as possible is also effective for the adjustment of latency between IU (integer unit) and FPU. This lightens instruction scheduling. The compact FPU has only 5 stages, and it works at 400 MHz.

By using the compact FPU, we designed the previous version of HCgorilla. This is HCgorilla.4 shown in Table 2. HCgorilla.4 still required awkward instruction scheduling due to the difference of the latencies of the 5-stage FPU and IU. This is similar to regular MFU that is configured by distinctive FPU and IU. Exactly, this holds for ALUs of MIPS and UltraSparc processors, scalar processing units of CRAY-I and NEC SX. The configuration of usual MFU composed of regular pipelines assures multifunctional behavior as a whole. But, it is spurious because these inner unifunctional pipelines are clearly independent and thus distinguished physically as well as logically. Although the usual configuration by regular pipelines is easy in constructing architecture, actually it does not have any advantage over the mere set of separate pipelines in view of area, speed, performance, etc. The same viewpoint holds even if unifunctional wave-pipelines are used in place of conventional pipelines. We take into account of fully merging combinational logic blocks in whole and its wave-pipelining.

3.2 Double Scheme

Although multifunctionalization might be effective for instruction scheduling, it is not always useful for overall aspects. The double scheme for scalar units solves both instruction scheduling and scale up. We apply the double scheme to EX stage. This produces waved MFU and a heterogeneously pipelined processor. The strong point of a heterogeneously pipelined processor is to combine the merit of related processors as shown in Table 1.

The wave-pipelined EX unit of a heterogeneously pipelined processor puts through any data in a single clock cycle owing to its complete merger of complicated circuits and simple circuits. Thus, a heterogeneously pipelined processor basically fulfills 1 CPI (clock cycles per instruction) supposing the immediate issue of scalar instructions. 1-CPI execution is an excellent feature that usual RISC processors do not always satisfy. The diversity of CPI of usual processors owes to the behavior and structure of their EX units. Some of them carry out complicated arithmetic like multiplication by iteratively using an ALU composed by arithmetic pipelines with the same delay. Others are composed of ALUs and more complicated arithmetic pipelines. It is intuitively understood that a heterogeneously pipelined processor occupies smaller area.

Table 1: *Comparison of a heterogeneously-pipelined processor and related processors.*

Pipelined processor		EX	Scalar processing		Vector processing
			+, -	*	
Regularly pipelined	Scalar processor	Regular MFU composed of distinctive arithmetic pipelines	1 CPI	>1 CPI	Impossible
	Base scalar processor				
	Vector processor		Impossible		Possible
Heterogeneously pipelined processor		Waved MFU completely merged	1 CPI		Possible

The double scheme for instruction scheduling free parallelism is applied to improve the ubiquitous processor HCgorilla we have so far developed for ubiquitous systems. Table 2 summarizes architectural aspects of HCgorilla familyfs dominant versions, focusing on the implementation of ILP. Improving HCgorilla.4, we have achieved HCgorilla.5.

Table 2: *HCgorilla family.*

Version	ISA		Pipeline					
	Java bytecode	SIMD Mode RAC	Arithmetic				Cipher
			No.	EX		Stack	
				IU	FPU		
HCgorilla.3	61	2	2	2-wave ×2	NA	1	1
HCgorilla.4	77				5-clock×2		
HCgorilla.5	58 (102)			2-wave MFU×2		2	
Remarks			Per a core	No./arithmetic pipe			No./core

The consensus of HCgorilla family is Java compatibility, symmetric double core, and multiple pipeline. Each core is composed of arithmetic media pipes and SIMD mode cipher pipes. The arithmetic pipe is supported by an LIW (Long Instruction Word) scheme we have developed. This assumes a compiler to extract ILP. The executable codes output by the LIW compiler is stored in instruction cache. The arithmetic pipe executes Java bytecodes and do stack operation following JVM style.

In order to solve the problem that HCgorilla.4fs IU and FPU take different latencies similarly to regular scalar units composed of physically divided subunits, the waved MFU is used. The resultant derivative is HCgorilla.5. The improvement from HCgorilla.4 to HCgorilla.5 owes not only the waved-MFU but also the addition of stacks. These are closely related. In order to solve the problem that ILP and stack machines like JVM are mutually exclusive [18], the HCgorillafs arithmetic media pipe has two stacks. This enhances the operation rate of the waved MFU. Corresponding to this, HCgorilla.5fs instruction set provides each stack with stack-based codes such as load, store, and arithmetic codes. As shown in Table 2, HCgorilla.5fs instruction set has 102 Java compatible media codes, which are produced from carefully selected 58 JVM codes.

3.3 Design Procedure

The application procedure of the double scheme is illustrated in Fig. 2. Fig. 2 (a) shows HCgorilla.4fs EX composed of 2-waved IU and 5-clock FPU, which is a sort of a heterogeneous pipeline. Due to the difference of the latencies of IU and FPU, the complicated instruction scheduling is inevitable. This is usually delegated to compilers. However, such approach depending on software does not always achieve good cost performance in case of ubiquitous platforms. A hardware approach to adjust the variation of latencies has been the application of variable latency pipeline to ALU [19]. Then, more straight forward approach is to reconstruct hardware units so that they take a constant latency. However, this surely takes larger scale and area. Consequently, it increases delay time, and thus degrades the clock speed of regular pipelines. A practical solution to clear the instruction scheduling is to adapt the double scheme.

The one of the double scheme, multifunctionalization is (a) Removing four arithmetic pipeline registers from FPU, (b) Logically synthesizing four kinds of integer arithmetic units of a 2-waved IU and the combinational logic of a 5-clock FPU, (c) Reducing front end instruction pipeline registers from 11 to 3, (d) Assembling five back end instruction pipeline registers, (e) Merging the control signals that distinguish one of arithmetic functions. These steps result in a non-waved or 1-waved MFU, whose clock speed is reduced due to the scale up caused by multifunctionalization.

By the wave-pipelining of the non-waved MFU, the waved-MFU shown in Fig. 2 (b) is achieved. MFU_SEL is a control signal that distinguishes one of functions. MFU_IN_A/B and MFU_OUT_A/B are the input and output registers, respectively. The waved MFU is power conscious due to the removal of arithmetic pipeline registers. Also, it is free from instruction scheduling, because it executes each function within the same latency. According to our previous work [11], the increased area of the waved MFU due to the buffers for delay tuning should be comparable with the area of the arithmetic pipeline registers used within the HCgorillafs FPU. Since our primary concern in this study is to achieve multifunctional behavior to clear instruction scheduling, area optimization is not always enough. Yet, some improvement is necessary for buffer insertion that saves more area.

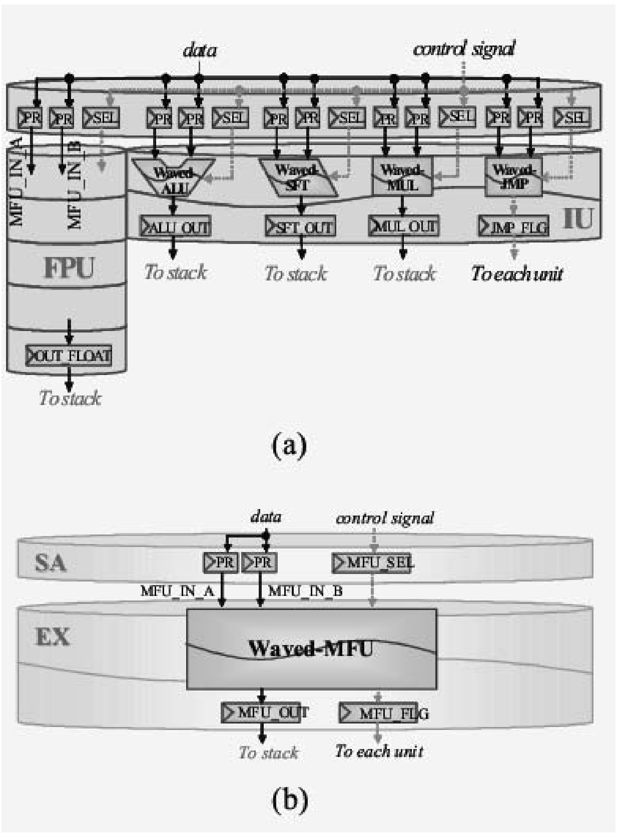

Fig.2: *Double scheme for HCgorillafs EX. (a) HCgorilla.4fs EX . (b) Waved MFU.*

4. PROCESSOR IMPLEMENTATION

The ubiquitous processor HCgorilla we have so far developed follows a symmetric double core architecture. This is dedicated to mobile use. Each core has multiple pipelines composed of media pipes and cipher pipes. Media pipes are SISD (single instruction stream single data stream) type arithmetic pipelines. These are distinguished by the EX stage that include IU and FPU. Cipher pipes are SIMD (single instruction stream multiple data stream) mode. Since the cipher pipe is occupied by the one instruction as long as the corresponding data stream continues, the multifunctionalrization scheme is exclusively applied to the arithmetic media pipefs EX stage. Table 3 summarizes the design scheme of HCgorilla.5.

Fig. 3 shows the hardware organization of HCgorilla.5. Since the arithmetic media pipe has two stacks as is described in Table 2, each core executes four arithmetic threads by making each arithmetic media pipe stagger two stacks by one clock. On the other hand, the register file is filled with pixel data by DMA transfer. The cipher pipe does double encryption during the transfer of pixel data from the register file to data cache.

Table 3: *Design Scheme of HCgorilla.5.*

Application field	Demand	Strategy		Technique	
				Hardware	Software
Multimedia	High performance	Parallelism	TLP	Multicore	Compiler
			ILP	Multiple pipe	Instruction scheduling
				Optimum scale MFU	Needless
Mobile	Wearable	Power consciousness			
				Wave-pipeline	
Internet	Quick response	High speed clock			
	Dynamic	Multithreading		Multicore	Compiler
	Diversity	Platform neutrality		Interpreter type Java CPU	API
	Security	Encryption		RNG	Cipher

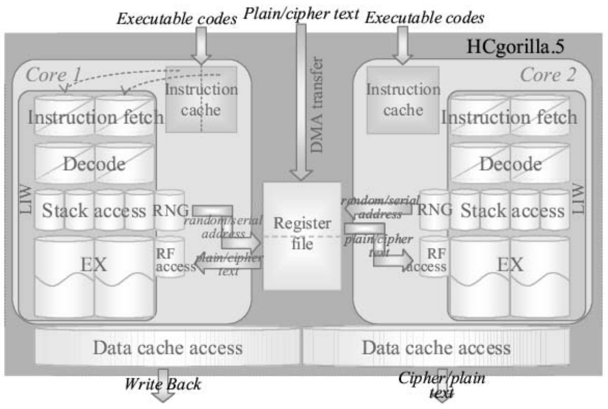

Fig.3: *Hardware organization of HCgorilla.5.*

The chip implementation of the improved HCgorilla.5 is done by using a 0.18-μm standard cell CMOS process. Table 4 summarizes specifications of HCgorilla family.

Fig. 4 shows the behavioural simulation of HCgorilla.5. Fig. 4 (a) shows a test program that adds from 1 to 128 and encrypts a standard image. HCgorilla.5fs internal behavior is also illustrated, focusing on core 1 for the sake of simple representation. Dividing the summation into four threads, these are assigned into four arithmetic media pipes.

Table 4: *Specifications of HCgorilla family.*

			HCgorilla.3	HCgorilla.4	HCgorilla.5
Design Rule			ROHM 0.18μm CMOS		
Wiring			1 polySi, 5 metal layers		
Area	Chip		5.0mm × 7.5mm		
	Core		4.28mm × 6.94mm		
Assembly	Pad	Signal	105		158
		VDD/VSS	48		32
	Package		QFP208 (Ceramic)		PGA257
Power Supply			1.8V (I/O 3.3V)		
Power consumption			241mW		274mW
Instruction cache			16bit × 32word × 2		16bit × 64word × 2
Data cache			16bit × 128word		16bit × 128word × 2
Stack memory			16bit × 8word × 4		16bit × 16word × 8
Register file			16bit × 64word		16bit × 128word
RNG			4bit		6bit
No. of . cores			2		
ILP degree			2		4
Clock frequency			330MHz	400MHz	200MHz
Current status			4.9 × 7.4-mm chip	Synthesis	4.9 × 7.4-mm chip

Fig. 4 (b) shows the result of simulation by the test program. The pipelined structure attached in the above corresponds to 200 MHz clock. A pair of instructions 0x60 fetched at the 1st clock (the actual clock counts are more than one) are integer addition by using stack_0s of the two arithmetic media pipes. Another pair of 0xe9s are integer addition by using stack_1s. Thus, each arithmetic media pipe computes two threads.

The computation of the thread 1 by the 1st arithmetic media pipe is as follows. 0x02 and 0x01 popped up from stack_0 at the 3rd clock are input into the waved-MFU. The EX stage takes two clocks and the result 0x03 are pushed on stack_0. Similarly, the computation of the thread 2 by the 1st arithmetic media pipe and stack_1 are staggered by two clocks. 0x2040 (8256 in decimal expression) stored in the data cache is the total summation of 1 to 128.

On the other hand, the register file stores the pixel data of the standard image in advance that are e2, 89, 7d, df, 89, 85 etc. 8f76 transferred from a register filefs address, which is synchronized with the actual clock counts of the 8th clock, is encrypted into fbe2 by using the RNG (random number generator) output 05. This is also the destination address of the data cache.

Fig. 5 shows the running time of simple integer summation taken by HCgorilla.5 and HCgorilla.3. Cipher pipes are idle in this case. HCgorilla.5 processes eight arithmetic threads at 200 MHz from Table 2, Fig. 3, and Table 4. HCgorilla.3 processes four threads at 330 MHz. HCgorilla. 3 is faster in the case where the summation range x is small. This is because clock speed is the dominant factor to determine the running time in the case of small x. When x is large, prevailing factor is the parallelism of arithmetic operations and thus HCgorilla.5 is faster. The effect of parallelism is useful for multimedia applications composed of iterative loops of many instructions.

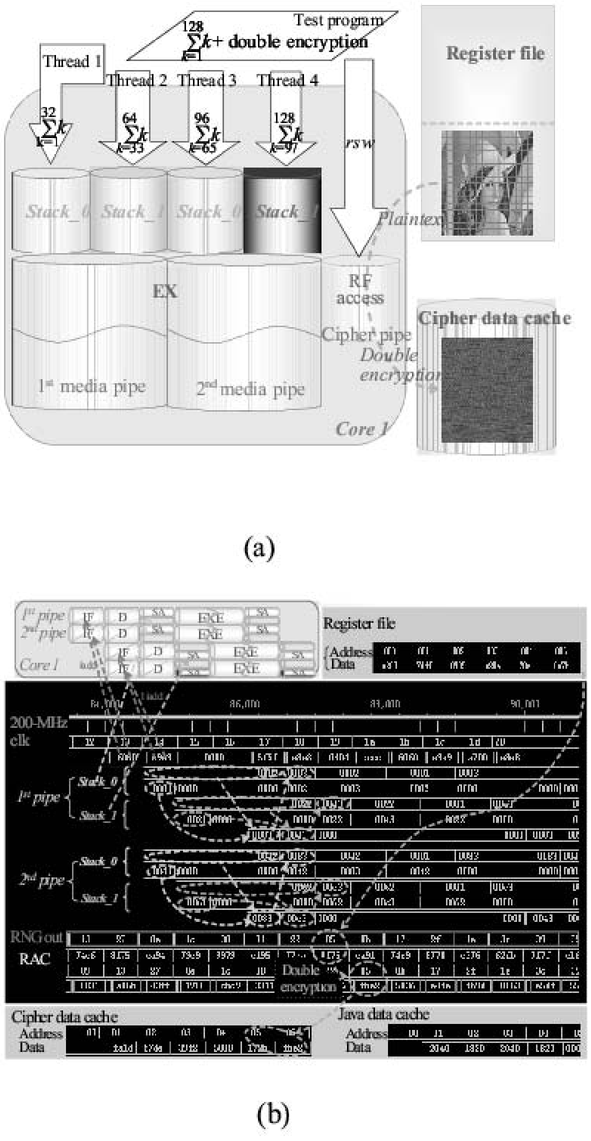

Fig.4: *Hardware organization of HCgorilla.5.*

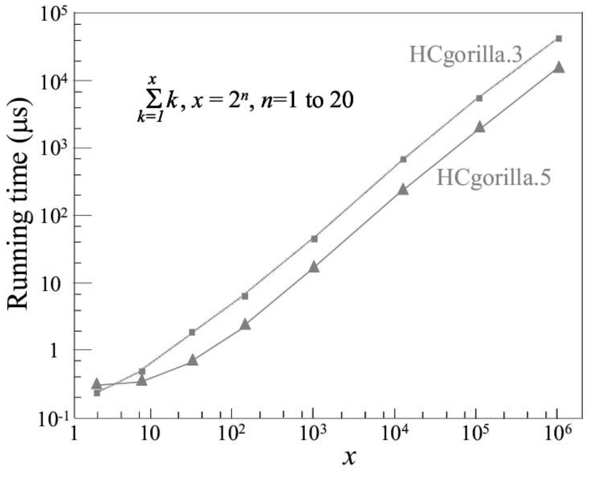

Fig.5: *Running time of HCgorilla.5 vs. HCGorilla.3.*

Fig. 6 shows the throughput of HCgorilla.5 and HCgorilla.4 in running the test programs A, B, and C. Program A sums integer k from 0 to 1024. This is the program used in Fig. 5 where x = 1024. Program B sums floating point number k from 0.0 to 1024.0, counting the loop by floating point numbers. Program C sums floating point number k from 0.0 to 1024.0. In this case, the loop count is float type. The throughput is derived from the locus of pipelined behavior on an instruction sequence vs. time space. The throughput of HCgorilla.5 is higher than HCgorilla.4, and it is almost constant in running any program. This is due to instruction scheduling free aspect. Thus, the improved HCgorilla.5 is promising for wide-range dynamic ILP at a higher rate.

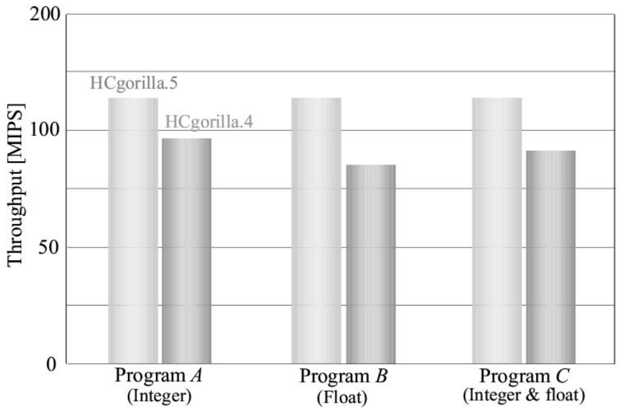

Fig.6: *Throughput of HCgorilla.5 vs. HCgorilla.4.*

5. CONCLUSION

We have studied the multifunctinalization and wave-pipelining of an EX stage. The waved MFU achieves instruction level hardware parallelism, instruction scheduling free pipeline, and power conscious speedup. Then, this has been applied to the improvement of the ubiquitous processor, HCgorilla. We have implemented HCgorilla built-in the waved MFU by using a 0.18-μ standard cell CMOS chip. This is applicable to media processing such as floating point calculation and cipher streaming.

The next step of our study will be the evaluation of the HCgorilla.5 chip. Also it is important to exploit more sophisticated wave-pipelining algorithms of delay tuning, clock multiplication, etc.

5.1 ACKNOWLEDGEMENT

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

References

[1] M. Fukase and T. Sato, Compact FPU Design and Embedding in a Ubiquitous

Processor for Multimedia Performance Enhancement, *ECTI-EEC Trans.*, Vol.6, No.2, pp.79-85, August 2008.

[2] M. Fukase, K. Noda, A. Yokoyama, and T. Sato, Enhancing Multimedia Processing by Wave-Pipelining Integer Units and Floating Point Units in Whole, *Proc. of ECTI-CON 2008*, pp.681-684, May 2008.

[3] M. Fukase and T. Sato, A Ubiquitous Processor Free from Instruction Scheduling, *Proc. of ISCIT*, pp.75-80, October 2008.

[4] K. Noda, A. Yokoyama, H. Takeda, M. Fukase, and T. Sato, Enhancing Multimedia Processing by Wave-Pipelining a Multifunctional Execution Unit, *Technical Report of IEICE*, Vol.107, No.8, pp.7-12, Mar 2008.

[5] L. Cotton, Maximum rate pipelining systems, *Procs. AFIPS Spring Joint Computer Conference*, pp.581-586, 1969.

[6] W. P. Burleson, M. Ciesielski, F. Klass, and W. Liu, Wave-Pipelining: A Tutorial and Research Survey, *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, Vol.6, No.3, pp.464-474, September 1998.

[7] W. Liu, C. Gray, D. Fan, T. Hughes, W. Farlow, and R. Cavin, A 250-MHz wave pipelined adder in 2-μm CMOS, *IEEE J. Solid-State Circuits,*, pp.1117-1128, September 1994.

[8] F. Klass, M. J. Flynn, and A. J. van de Goor, Fast multiplication in VLSI using wave-pipelining, *J. VLSI Signal Processing*, 1994.

[9] D. C. Wong, G. De Micheli, M.J. Flynn, R. E. Huston, A bipolar population counter using wave pipelining to achieve 2.5 x normal clock frequency, *IEEE J. Solid-State Circuits*, Vol.27, No.5, pp.745-753, May 1992.

[10] H. J. Yoo, K. W. Park, C. H. Chung, S. J. Lee, H. J. Oh, J. S. Son, K. H. Park, K. W. Kwon, J. D. Han, W. S. Min, and K. H. Oh, A 150 MHz 8-banks 256 Mb synchronous DRAM with wave pipelining methods, *Proc. ISSCC 1995* pp.250-251, 1995.

[11] M. Fukase, T. Sato, et. al, Scaling up of Wave-Pipelines, *Proc. of the 14th Inter. Conf. on VLSI Design*, pp. 439-445, January 2001.

[12] M. Fukase and T. Sato, Exploiting Design and Testing Methods of High-Speed Power Conscious Wave-Pipelines, *Proc. of NASA2007*, pp.5.1.1-5.1.6, January 2007.

[13] Y. Ikeda, Wave-pipelined architecture of a multithreaded processor, *Master Thesis of JAIST*, Febuary 1999.

[14] J. G. Delgado-Frias and J. Nyathi, A hybrid wave-pipelined network router, *Proc. of IEEE Computer Society Workshop*, VLSI 2001, pp.165-170, 2001.

[15] T. Horel and G. Lauterbach, ULTRASPARC-III: Designing Third-Genaration 64-Bit Performance, *IEEE MICRO*, pp.73-85, May 1999.

[16] M. Fukase, T. Sato, R. Egawa, and T. Nakamura, Designing a Wave-Pipelined Vector Processor, *Proc. of The Tenth Workshop on Synthesis and System Integration of Mixed Technologies*, pp.351-356, October 2001.

[17] 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.

[18] S. Nakagawa and H. Yanagi, Development of Realtime JavaTM Processor Execution Core, *OMRON TECHNICS*, Vol.40, No.1, pp.38-42, 2000.

[19] T. Sato and I. Arita, Combining Variable Latency Pipeline with Instruction Reuse for Execution Latency Reduction, *The Trans. of the IEICE D-I*, No.12, pp.1103-1113, December 2002.

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 has been the Director of the Hirosaki University C&C Systems Center since 2004. He is also Representative of Hirosaki University R&DC of Next Generation IT Technologies since 2008. His current research activities are mainly concerned with the design, chip implementation, and application of power conscious highly performable VLSI processors.

Tomoaki Sato received the B.S. and M.S. degrees from Hirosaki University, Japan, in 1996 and 1998 respectively, and the Ph.D. degree from Tohoku University, Japan, in 2001. From 2001 to 2005, he was an Assistant Professor of Sapporo Gakuin University, Japan. Since 2005, he has been an Associate Professor of Hirosaki University. His research interests include VLSI Design, Computer Hardware, and Computer and Network Security.