A Novel Reconfigurable Processor Using Dynamically Partitioned SIMD for Multimedia Applications

Chun-Gi Lyuh, Jung-Hee Suk, Ik-Jae Chun, and Tae Moon Roh

In this paper, we propose a novel reconfigurable processor using dynamically partitioned single-instruction multiple-data (DP-SIMD) which is able to process multimedia data. The SIMD processor and parallel SIMD (P-SIMD) processor, which is composed of a number of SIMD processors, are usually used these days. But these processors are inefficient because all processing units (PUs) should process the same operations all the time. Moreover, the PUs can process different operations only when every SIMD group operation is predefined. We propose a processor control method which can partition parallel processors into multiple SIMD-based processors dynamically to enhance efficiency. For performance evaluation of the proposed method, we carried out the inverse transform, inverse quantization, and motion compensation operations of H.264 using processors based on SIMD, P-SIMD, and DP-SIMD. Experimental results show that the DP-SIMD control method is more efficient than SIMD and P-SIMD control methods by about 15% and 14%, respectively.

Keywords: Reconfigurable processor, SIMD, array processor, multimedia, DSP, H.264.

Manuscript received May 6, 2009; revised Oct. 12, 2009; accepted Oct. 15, 2009.

This work was supported by the IT R&D program of MKE/IITA, Rep. of Korea (2006-S-006-04, Components/Module technology for Ubiquitous Terminals).

Chun-Gi Lyuh (phone: +82 42 860 1248, email: cglyuh@etri.re.kr), Jung-Hee Suk (email: jhsuk@etri.re.kr), Ik-Jae Chun (ijchun@etri.re.kr), and Tae Moon Roh (tmroh@etri.re.kr) are with the Convergence Components & Materials Research Laboratory, ETRI, Daejeon, Rep. of Korea.

doi:10.4218/etrij.09.1209.0021

I. Introduction

H.264 is an open, licensed standard which supports most of the efficient video compression techniques available today. In addition, the higher performance in video compression and decompression is required, the more complex hardware is needed as in [1]. Various hardware implementation methods for H.264 have been proposed [2]-[6]. Some partial or full H.264 encoders or decoders in [2]-[4] were developed by using an application-specific integrated circuit (ASIC) design method. Although these encoders or decoders implemented by ASIC are very good for high performance, they have limitations in flexibility when some modifications are needed. On the other hand, an application specific instruction set processor (ASIP) [5] and configurable processor [6] design methods have some flexibility, but their performance is not as good as that of the ASIC design method.

Figure 1 shows the relations between flexibility and performance according to the implementation methods. A general purpose processor is flexible enough to be applied to various systems in various fields, but its performance is low. An ASIP can adopt a more optimized instruction set than a

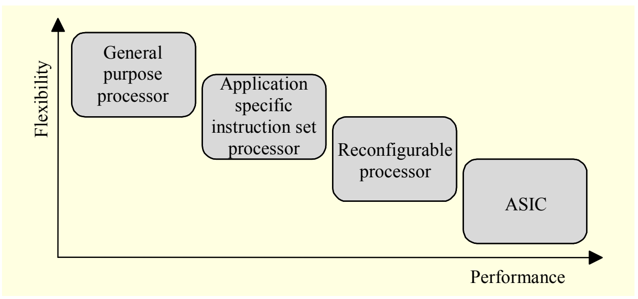

Flexibility
General purpose processor
Application specific instruction set processor
Reconfigurable processor
ASIC
Performance

Fig. 1. Trade-off between flexibility and performance.

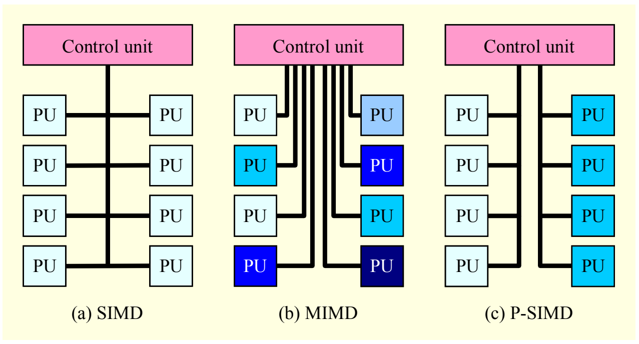


Fig. 2. Various control methods.

general purpose processor can for performance enhancement. However, an ASIP also has the same disadvantage in performance that the general purpose processor has. Although an ASIC has higher performance than other circuits due to its optimized hardware implementation for each application, it cannot be altered flexibly for various applications. A reconfigurable processor takes advantage of the general purpose processor and ASIC to achieve high performance with reasonable flexibility. A reconfigurable architecture was first proposed by Estrin [7], [8]. The reconfigurable architecture is a hybrid computer architecture composed of fixed and variable blocks and supports flexibility in software and the speed of the hardware. Research on the flexibility and performance of the reconfigurable architecture has been pursued.

A reconfigurable processor generally has an array architecture for its operation performance and requires a smart control method for its array processors in order to improve its operation efficiency. We can classify the control methods into three main groups: single-instruction multiple-data (SIMD), multiple-instructions multiple-data (MIMD), and parallel SIMD (P-SIMD) control methods. These methods are shown in Fig. 2. In Fig. 2, PU denotes processing unit (PU), and thick lines denote the instructions to operate in each PU. The SIMD control method has a short length instruction, and all the PUs work using the same operation. Therefore, when various independent operations are needed, the method causes a reduction in processing performance. Also, the MIMD control method can improve the processing performance by using different instructions in each of the PUs. However, in reconfigurable processor architecture, only one control unit (CU) controls every PU for the different instructions. That is, a PU cannot work independently. As a result, the operations are inefficient if the instruction cannot be predefined according to the condition of the data. The method also has problems of long instructions, such as area overhead.

To solve the problems of the SIMD and MIMD control methods, we can use a P-SIMD control method. The parallel processors are partitioned into several SIMD groups when they are designed, and each partitioned SIMD group uses the same instructions. For example, MorphoSys, one of well-known reconfigurable architectures, uses the P-SIMD control method to control 64 reconfigurable cells [9]. REMARC uses both the P-SIMD and MIMD control methods to control 64 nanoprocessors [10]. However, in P-SIMD control methods, the operations are still inefficient if the instructions cannot be predefined according to the condition of the data. Therefore, we propose a dynamically partitioned SIMD (DP-SIMD) control method and implement an architecture supporting the control method to solve the problems of the existing SIMD, MIMD, and P-SIMD control methods.

The rest of this paper is organized as follows. In section II, a DP-SIMD control method is proposed. Section III presents the reconfigurable processor based on the proposed control method. Section IV shows the experimental results. Finally, section V concludes this paper.

II. Dynamically Partitioned SIMD Architecture

1. Dynamically Partitioned SIMD Control Method

The H.264 decoder is composed of variable length decoding (VLD), inverse transform and quantization (ITQ), motion compensation (MC), intra prediction (IP), and a deblocking filter (DF) [11], [12]. Among these, VLD is not compatible with a parallel process, and ITQ can promote operation performance using a SIMD control method. However, the IP, MC, and DF change operations according to conditions such as prediction mode and boundary strength. These conditional operations selected according to the conditions reduce the efficiency of the SIMD operation.

Figure 3 shows an example of the conditional operations to obtain the variables a and d using different operands and operators depending on the condition.

Because all of the PUs composed in parallel for the SIMD control method generally perform the same operation at the same time, all of the PUs have to perform all operations of each

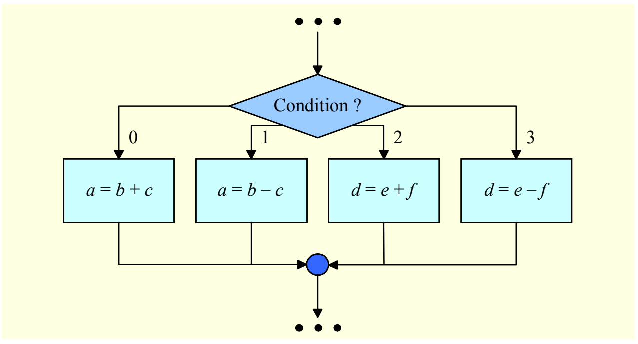


Fig. 3. An example of conditional operations.

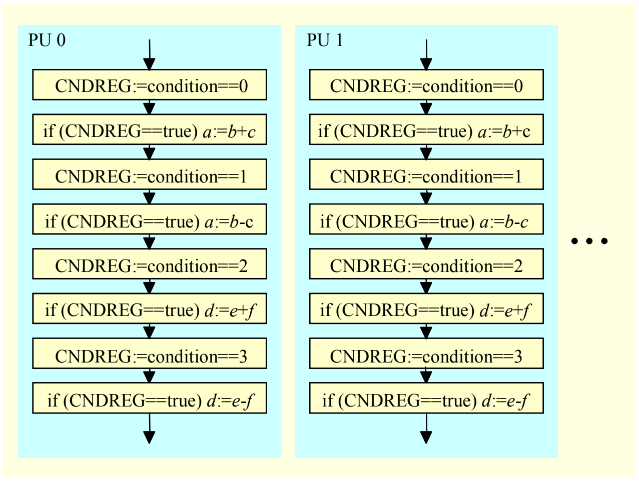
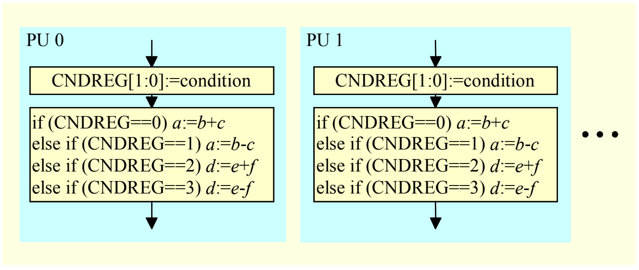


Fig. 4. Conditional operation of the SIMD control method.


PU 0
CNDREG[1:0]:=condition
if (CNDREG==0) a:=b+c
else if (CNDREG==1) a:=b-c
else if (CNDREG==2) d:=e+f
else if (CNDREG==3) d:=e-f
PU 1
CNDREG[1:0]:=condition
if (CNDREG==0) a:=b+c
else if (CNDREG==1) a:=b-c
else if (CNDREG==2) d:=e+f
else if (CNDREG==3) d:=e-f
• • •
Fig. 5. Conditional operation of the DP-SIMD control method.

of the conditions including other conditions that are not related to a given processing flow. This is the reason for the low operation efficiency of the SIMD control method. Figure 4 shows the operation method when the previous example is carried out under a parallel processor of the existing SIMD control method. First, the condition flag register CNDREG is set true after the decision of whether the condition is zero is made, and the operation result is then reflected according to the CNDREG while the operation $a := b + c$ under first condition is performed. After that, the second, third, and fourth conditions are applied in the same manner. All PUs should perform all these operations, and it takes 8 steps including 4 steps for computations. Although the P-SIMD and the MIMD control methods can enhance the performance when the categories of operations for each PU can be predecided before their operation, they achieve almost the same performance as SIMD in other cases.

On the other hand, PUs in DP-SIMD can select the instructions according to the data condition while their operations are executed. Therefore, the parallel processors can be dynamically partitioned into multiple identical instruction groups of SIMD at any time. This is why we named the proposed method *dynamically partitioned SIMD*. In DP-SIMD processors, the performance can be enhanced because the PUs do not have to process useless operations. Figure 5 shows an operation method when applying the previous example using the proposed DP-SIMD control method. First, each PU estimates the condition value through the SIMD operation and then sets the CNDREG value. After that, it performs the operation through the DP-SIMD control method according to the CNDREG value. In this case, two steps are sufficient for the processing time, including only one step for computation. When four instructions can be selected at a time in an implemented DP-SIMD processor, the operation efficiency of DP-SIMD would be enhanced 4 times compared to SIMD in the ideal.

2. Dynamically Partitioned SIMD Architecture

By applying the DP-SIMD control method in a parallel processor, the problem of low operation efficiency caused by the existing SIMD control method is solved. The instruction for the DP-SIMD is created by combining various instructions into one, as in the case of a very long instruction word (VLIW) processor. But, unlike the VLIW, which determines the order according to the function block, the PU of a DP-SIMD directly chooses an instruction using the conditions and then performs its operation.

Figure 6(a) shows a processor structure of the DP-SIMD control method. Several PUs are composed in parallel, and a CU is used to control them. The CU reads and transmits the instructions that the PUs will carry out from the program memory and controls the entire operation.

There are three possible means of control in the DP-SIMD control method. First, an instruction based on the value calculated by a PU can be chosen. This is called *data-based partitioned SIMD control*. An example for this step is shown in Fig. 6(b). In the case of the previous example for the conditional operation, a PU sets up the CNDREG register according to the condition value. The CU assigns D (data) to the DPS signal and activates the PU. At this time, the InstrMux sends a selected instruction to the PU according to the CNDREG register value.

Second, an instruction based on the value determined in advance by the PU can be chosen. This is called *position-based partitioned SIMD control*. An example for this step is shown in Fig. 6(c). Based on the type and structure of the operation, the CU can previously save a value for selecting an instruction in the POSREG register, which is allotted 2 bits by each of the PUs if the PU determines the instruction in advance. The CU assigns P (position) to the DPS signal and activates the PU. At this time, the InstrMux sends a selected instruction to the PU according to the *POSREG* register value. The implementation of the P-SIMD control method is possible when this method is used.

Finally, all of the PUs can operate the same instruction by the SIMD control method. This is called *SIMD control*. Figure 6(d) shows this example. The CU assigns S (SIMD) to the DPS signal for the SIMD control, sets a selected value to the SI signal for the selection of the instruction, and activates the PUs. At this time, the InstrMux sends a selected instruction to the PUs. All the PUs receive the same instruction.

III. ETRI Reconfigurable Processor

We have developed a DP-SIMD-based ETRI reconfigurable processor (ERP) that has a high data capacity for video CODECs.

1. Overall Architecture

Figure 7 shows the overall architecture of the ERP. The CU is for an external I/O and control of the internal PU array (PUA). It is a 32-bit custom processor which controls the *processing unit array configuration memory* and the *processing unit array data memory* in parallel with the execution of a program for self operation using the *control unit program memory* and the *control unit data memory*.

The architecture of the PUA is depicted in Fig. 8. The PUA has a two-dimensional array architecture for parallel processing. It has four rows and can also be designed using a variable dimensional value according to the performance requirement. In this paper, the PUA has a 4×16 array. In addition, arrays of 4×4, 4×8, 4×32, and so on are available. Two types of connections among the PUs are supported for an efficient data transfer between them. First, Fig. 8(a) shows a mesh-type interconnection architecture for a basic data transfer between the PUs comprising a two-dimensional array, and Fig. 8(b) shows a ring-type interconnection for various data exchanges.

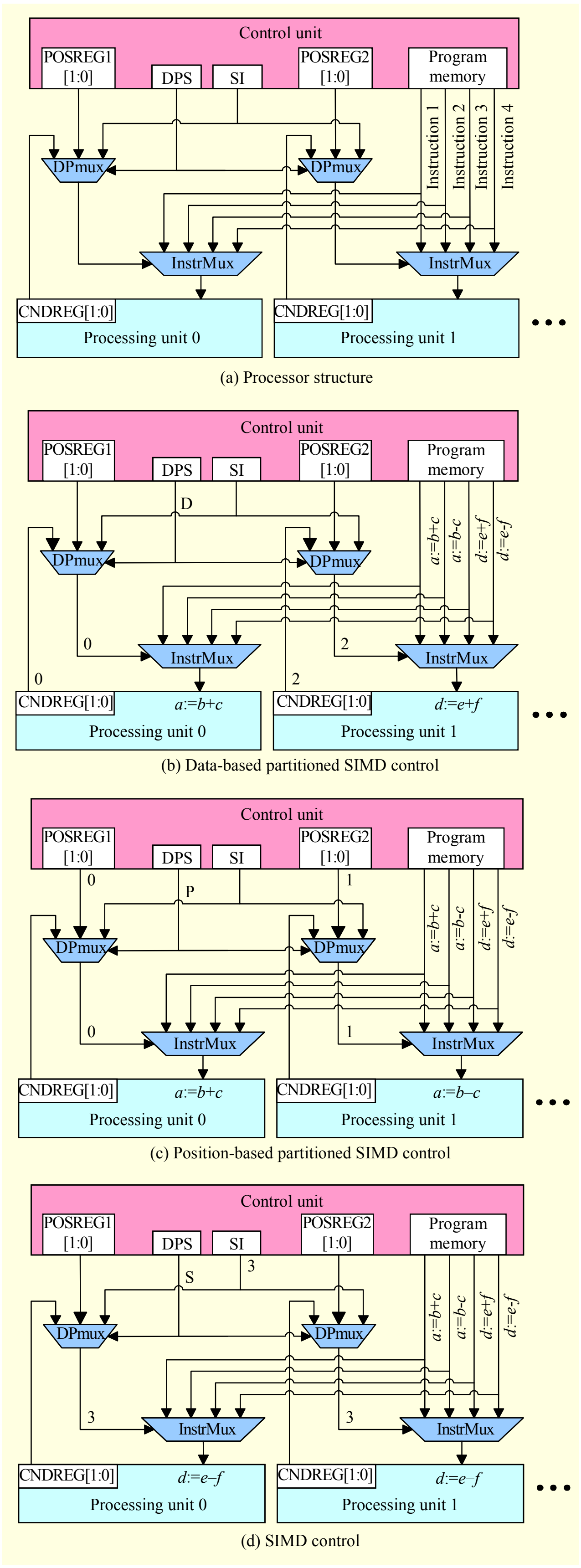
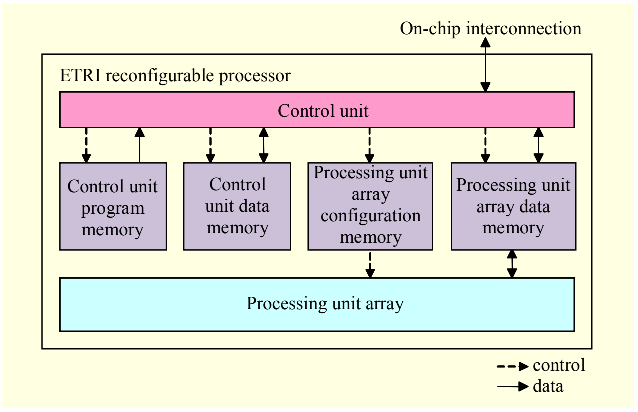


Fig. 6. Dynamically partitioned SIMD processor architecture.



On-chip interconnection
ETRI reconfigurable processor
Control unit
Control unit program memory
Control unit data memory
Processing unit array configuration memory
Processing unit array data memory
Processing unit array
control
data

Fig. 7. Architecture of the ETRI reconfigurable processor.

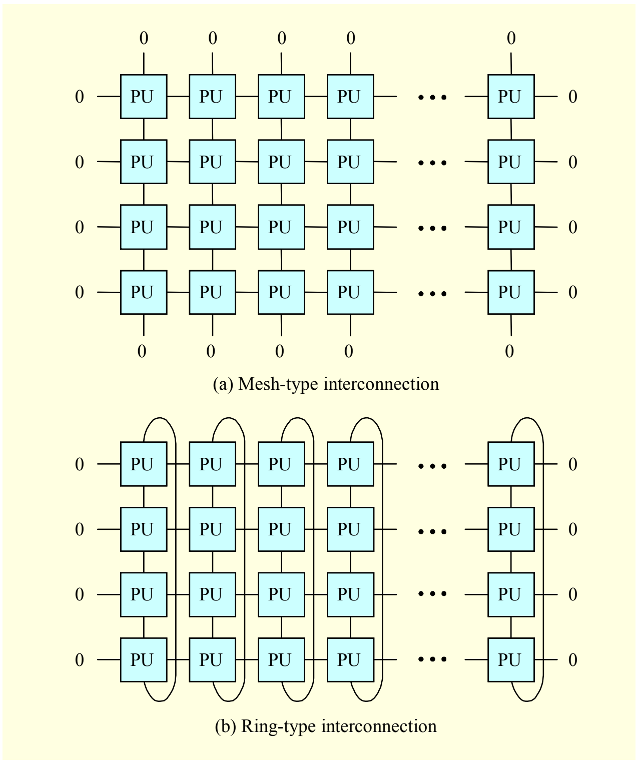


Fig. 8. Architecture of processing unit array.

2. Special Instructions for Multimedia Operation

The ERP has the basic instructions for general operation as well as special instructions considering multimedia applications.

Table 1 shows the special instructions which consist of 7 individual instructions. The MAX, MIN, and CLIP instructions reduce the operations for comparing the sizes among several values. The SUBABS instruction could be useful with subtraction and absolute operations used in motion estimation, IP, and so on of H. 264. The SRAC instruction is useful in division operations. A division operation is generally achieved with a right-shift operation. In the case of rounding off the result of a division, half of the divisor is added to the operand before division, and then the total value is divided by the shift operation. The SRAC instruction performs division in one operation.

The CMP instruction compares the sizes of two values. Because processors have PUs performing the same instructions, a conditional jump is not available. This limitation could be solved using a conditional operation such as the CMP. When the CMP is carried out, the result of the operation is saved in a condition register file, and the saved value is used to decide the execution of other operations. In addition, the value saved in the conditional register file is used for the DP-SIMD control discussed in section II.

Table 1. Special instructions.

Instruction	Description
MAX a $b > c$	$c := (a > b)\ ?\ a : b$
MIN a $b > c$	$c := (a < b)\ ?\ a : b$
CLIP a $b > c$	$c :=$ max(0, min(a, b))
SUBABS a $b > c$	$c :=$ abs(a –b)
SUBABS4 a b c $d > e$	$e :=$ abs($a - b$) + abs($c - d$)
SRAC a $b > c$	$c := a >> b +$ carry
CMP a cnd $b > c$	$c := (a$ cnd $b)\ ?\ 1 : 0$

3. Processing Unit Architecture

The PU basically performs 16-bit operations. Figure 9 shows the architecture of the PU. The PU has an arithmetic logic unit (ALU) for basic arithmetic and logical operations and includes a 16-bit multiplier and shifter. The register file consists of 32 16-bit register files that are able to input and output data through 5-read and 2-write ports in order to perform data transfers and operations simultaneously. In addition, the conditional register file is used for a conditional operation and dynamically partitioned SIMD control. The PU also has a 16-bit adder to improve the performance of the arithmetic operation and to carry out SRAC operations.

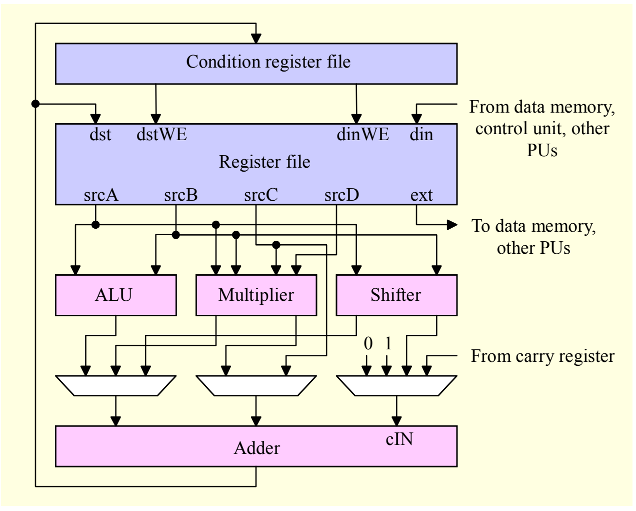
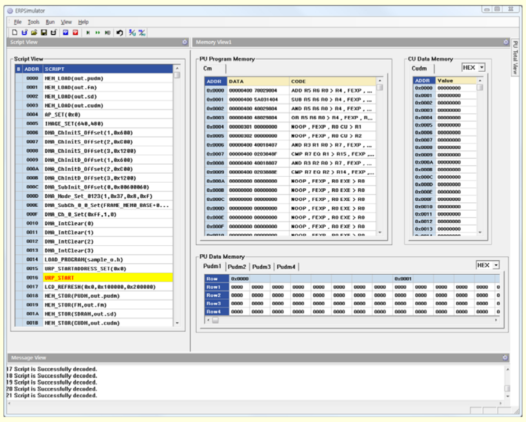


Fig. 9. Processing unit architecture.

4. Software Development Tools for ERP

A software developer tool is supported. First, an ERP assembler is available for compiling the basic assembly level construction and high-level language-like construction. It is also able to detect a hazard from the pipeline processing in the

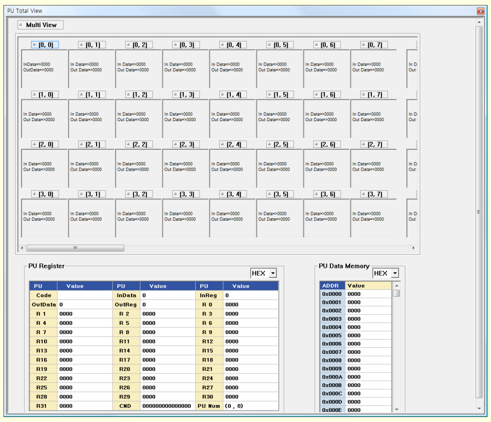


```
ADDR SCRIPT
0000 MEM_LOAD(out.pudm)
0001 MEM_LOAD(out.fm)
0002 MEM_LOAD(out.sd)
0003 MEM_LOAD(out.cudm)
0004 AP_SET(0x0)
0005 IMAGE_SET(640,480)
0006 DMA_ChInitS_Offset(1,0x600)
0007 DMA_ChInitS_Offset(2,0xC00)
0008 DMA_ChInitS_Offset(3,0x1200)
0009 DMA_ChInitD_Offset(1,0x600)
000A DMA_ChInitD_Offset(2,0xC00)
000B DMA_ChInitD_Offset(3,0x1200)
000C DMA_SubInit_Offset(0,0x00600060)
000D DMA_Mode_Set_0123(1,0x37,0x8,0xF)
000E DMA_SubCh_0_0_Set(FRAME_MEM0_BASE+0...
000F DMA_Ch_0_Set(0xFF,1,0)
0010 DMA_IntClear(0)
0011 DMA_IntClear(1)
0012 DMA_IntClear(2)
0013 DMA_IntClear(3)
0014 LOAD_PROGRAM(sample_o.h)
0015 URP_STARTADDRESS_SET(0x0)
0016 URP_START
0017 LCD_REFRESH(0x0,0x100000,0x200000)
0018 MEM_STOR(PUDM,out.pudm)
0019 MEM_STOR(FM,out.fm)
001A MEM_STOR(SDRAM,out.sd)
001B MEM_STOR(CUDM,out.cudm)
```

```
ADDR   DATA              CODE
0x0000 00000400 78029804 ADD R5 R6 R0 > R4 , FEXP , ...
0x0001 00000400 5A031404 SUB R5 R6 R0 > R4 , FEXP , ...
0x0002 00000400 40029804 AND R5 R6 R0 > R4 , FEXP , ...
0x0003 00000400 48029804 OR R5 R6 R0 > R4 , FEXP , R...
0x0004 00000301 00000000 NOOP , FEXP , R0 CU > R1
0x0005 00000302 00000000 NOOP , FEXP , R0 CU > R2
0x0006 00000400 40018407 AND R3 R1 R0 > R7 , FEXP , ...
0x0007 00000400 0203840F CMP R7 EQ R1 > R15 , FEXP ,...
0x0008 00000400 40018807 AND R3 R2 R0 > R7 , FEXP , ...
0x0009 00000400 0203888E CMP R7 EQ R2 > R14 , FEXP ,...
0x000A 00000000 00000000 NOOP , FEXP , R0 EXE > R0
0x000B 00000000 00000000 NOOP , FEXP , R0 EXE > R0
0x000C 00000000 00000000 NOOP , FEXP , R0 EXE > R0
0x000D 00000000 00000000 NOOP , FEXP , R0 EXE > R0
0x000E 00000000 00000000 NOOP , FEXP , R0 EXE > R0
0x000F 00000000 00000000 NOOP , FEXP , R0 EXE > R0
```

```
17 Script is Successfully decoded.
18 Script is Successfully decoded.
19 Script is Successfully decoded.
20 Script is Successfully decoded.
21 Script is Successfully decoded.
```

(a) Script and memory view



(b) Processing unit array view

Fig. 10. ERP simulator screenshots.

CU and the PU, and it achieves a simple optimization of the source code. It finally outputs image files for the CU program memory and PUA configuration memory. Second, an ERP simulator is available for the simulation of the operation result received from the images generated by the assembler. The simulator is able to show the internal memory of the CU and PUA, the register, and so on. It also can simulate the data exchange between an external device and the ERP as well as the output of the liquid crystal display controller. Figure 10(a) shows a script for the simulation of the data exchange between the input and output of the ERP and an internal memory state. Figure 10(b) shows a window to display the memory and register state of the PUA.

IV. Experimental Results

We implemented the ERP with a 4 × 16 PUA structure using Verilog HDL. Table 2 shows the synthesis results of the ERP. The total internal memory is 305 kB, total cell area is 17.5 mm^2, and the gate count is 6.2 million. The operation frequency is up to 214 MHz. For the previously described results, the ERP was synthesized using a 90 nm CMOS technology by the Synopsys Design Compiler Ultra.

Table 2. Synthesis results.

Synthesis tool		Synopsys DC Ultra
Process		90 nm CMOS
Processing unit array size		64 (4×16)
Memory size	Control unit program memory	32 kB
	Control unit data memory	4 kB
	Processing unit array configuration memory	13 kB
	Processing unit array data memory	256 kB
	Total memory area	13.4 mm^2
Total cell area		17.5 mm^2
Total gate count		6.2 M
Maximum clock speed		214 MHz

Table 3. H.264 decoder MC & ITQ processing time.

Function		SIMD (cycle)	P-SIMD (cycle)	DP-SIMD (cycle)
Type	Name			
D	GetTable	141	141	141
	Hadamard_Luma	181	181	149
	Hadamard_Chroma	63	63	47
	MemoryAlign	2,091	2,091	1,143
	HalfQuarterPel	7,048	7,048	6,724
P	ReadDC_Luma	37	31	31
	WriteDC_Luma	28	20	20
	WriteDC_Chroma	10	6	6
S	IQ_Luma	210	210	210
	IQ_Chroma	107	107	107
	IDCT	433	433	433
	MakeRecon	149	149	149
	Etc.	10	10	10
Total (instruction count)		10,508 (10,508)	10,490 (10,562)	9,170 (10,898)

Table 3 shows the processing time (operation cycles) and instruction count for the MC and ITQ operations with 16 macroblocks. The operation cycles were counted through RTL simulations using ModelSim and the function type indicates the characteristic of the parallelism in each function block. While all the three control methods, SIMD, P-SIMD, and DP-SIMD, showed the same performance for S type functions, P-SIMD and DP-SIMD reduced the required operation cycles for P type functions by 40% over SIMD, and DP-SIMD

Table 4. Comparisons of Xtensa and ERP.

Processor name	Xtensa	ERP
Cycle count	15.342M	0.227M
Gate count	25,000	Logic 1.45M Memory 4.7M

reduced the required operation cycles for D type functions by 4.8% over SIMD and P-SIMD. Even though the instruction count for DP-SIMD was increased 3.7% over SIMD and 3.2% over P-SIMD, the total processing time for DP-SIMD was improved more than 14% over SIMD and P-SIMD control methods. Consequently, the ERP consumes only 3.9 ms for 1 frame of D1 size when ITQ and MC for H.264 are operated at 200 MHz.

Table 4 compares the operation cycle and the area for the MC and ITQ operations with 1 frame of CIF size between the Xtensa processor and the ERP. The Xtensa processor has a basic instruction set [6]. The ERP gate count is 58 times larger and the operation cycle is 68 times faster than those of Xtensa. This demonstrates that the operation of ERP is more efficient than that of Xtensa.

V. Conclusion

In this paper, we proposed a DP-SIMD control method to improve the operation efficiency of the existing SIMD control method. In the ETRI reconfigurable processor we developed, the processing units performing the instructions for video data processing are implemented in an array configuration, and the array is controlled through the DP-SIMD method.

We can apply the ERP to multimedia data processing as well as various applications which need to process mass data in parallel with conditional operations. We expect the proposed control method to be applied to other parallel processor architectures to enhance performance.

In future work, we need to develop an optimized C compiler for the DP-SIMD processor and a low power consumption architecture for mobile devices.

References

[1] M. Horowitz et al., "H.264/AVC Baseline Profile Decoder Complexity Analysis," *IEEE Trans. Circuits Syst. for Video Technol.*, vol. 13, July 2003, pp. 704-716.

[2] M. Oh et al., "Design of High-Speed CAVLD Decoder Architecture for H.264/AVC," *ETRI J.*, vol. 30, no. 1, Feb. 2008, pp. 167-169.

[3] D. Yeo and H. Shin, "High Throughput Parallel Decoding Method for H.264/AVC CAVLC," *ETRI J.*, vol. 31, no. 5, Oct. 2009, pp. 510-517.

[4] K.S. Choi and S.J. Ko, "Adaptive Scanning Based on a Morphological Representation of Coefficients for H.264/AVC," *ETRI J.*, vol. 31, no. 5, Oct. 2009, pp. 607-609.

[5] Y. Kun, Z. Chun, and W. Zhihua, "Application Specific Processor Design For H.264 Baseline Profile Bit-Stream Decoding," *Proc. the 8th Int. Conf. Signal Process.*, 2006, pp. 16-20.

[6] J.H. Han et al., "Application Specific Processor Design for H.264 Decoder with a Configurable Embedded Processor," *ETRI J.*, vol. 27, no. 5, Oct. 2005, pp. 491-496.

[7] G. Estrin, "Organization of Computer Systems: The Fixed-Plus-Variable Structure Computer," *Proc. the Western Joint Computer Conf.*, 1960, pp. 33-40.

[8] G. Estrin and C.R. Viswanathan, "Organization of a Fixed-Plus-Variable Structure Computer for Computation of Eigenvalues and Eigenvectors of Real Symmetric Matrices," *J. ACM*, vol. 9, no.1, Jan. 1962, pp. 41-60.

[9] H. Singh et al., "MorphoSys: An Integrated Reconfigurable System for Data-Parallel and Computation-Intensive Application," *IEEE Trans. Computers*, vol. 49, no. 5, May 2000, pp. 465-481.

[10] T. Miyamori and K. Olukotun, "REMARC: Reconfigurable Multimedia Array Coprocessor," *IEICE Trans. Inf. Syst.*, vol. E82-D, no. 2, 1999, pp. 389-397.

[11] G. Sullivan, A. Luthra, and T. Wiegand, "Draft of Version 4 of ISO/IEC 14496-10(E)," Joint Video Team (JVT) of ISO/IEC MPEG and ITU_T VCEG, Apr. 2005.

[12] JVT H.264/AVC Joint Model Reference Software version 11.0, http://iphome.hhi.de/suehring/tml/download/old_jm/jm11.0.zip, Aug. 2007.

Chun-Gi Lyuh received the BS degree in computer engineering from Kyungpook National University (KNU), Korea, in 1998. He received the MS and PhD degrees in electrical engineering and computer science from the Korea Advanced Institute of Science and Technology (KAIST) in 2000 and 2004, respectively. He joined ETRI in 2004 as a senior researcher. His current research interests include reconfigurable processor and vision SoC platform for intelligent vehicles.

Jung-Hee Suk received the BS, MS, and PhD degrees in electronics engineering from Kyungpook National University (KNU) in Daegu, Korea, in 2001, 2003, and 2007, respectively. His doctoral research involved the H.264/AVC codec algorithm. He joined ETRI in February 2007 as a researcher in the NT Convergence Components Research Department. His current research interests include multimedia codec, parallel processing of media data, recognition algorithm, and multimedia SoC.

Ik-Jae Chun received the BS, MS, and PhD degrees in electronics engineering from Chungnam National University (CNU) in Daejeon, Korea, in 1998, 2000, and 2006, respectively. He joined ETRI in September 2006 as a senior researcher in the NT Convergence Components Research Department. His current research interests include digital system architecture, communication system design, multimedia SoC, and vision SoC. He is a member of the IEEE.

Tae Moon Roh received the BS, MS, and PhD degrees in electrical engineering and computer science from Kyungpook National University, Daegu, Korea, in 1984, 1986, and 1998, respectively. His doctoral research involved the reliability of ultrathin oxide grown by high pressure oxidation and followed by rapid thermal nitridation and hot carrier effects of MOSFET with the oxide as the gate insulator. In 1988, he joined ETRI, Daejeon, Korea. Since 1988, he has been working at Convergence Components and Materials Research Laboratory, where he has been engaged in research developing process technology for digital and analog CMOS ICs, improving the reliability of ultra-thin gate oxides, and evaluating hot carrier effects of MOSFETs. His current interests are low-power digital circuits and multimedia SoCs with reconfigurable processors. In addition, he is working on the development of a vision SoC platform for intelligent vehicles.