Evaluation of MuCCRA-D: A Dynamically Reconfigurable Processor with Directly Interconnected PEs

Masaru Kato, Yohei Hasegawa, and Hideharu Amano
Department School of Science and Technology, Keio University
3-14-1 Hiyoshi, Yokohama 223-8522, JAPAN
Email: muccra@am.ics.keio.ac.jp

Abstract

Coarse-grained dynamically reconfigurable processor arrays (DRPAs) have been received an attention as a flexible and efficient off-loading engine for various types of System-on-Chips (SoCs). Interconnection in these architectures is one of the important factors to be evaluated. MuCCRA-1, the first prototype of MuCCRA(Multi-Core Configurable Reconfigurable Architecture) project, uses a typical island-style interconnection in its PE array. Although the island-style interconnection is flexible, the large delay time caused by passing multiple switches and long wires often degrades its clock frequency. In this paper, MuCCRA-D, a dynamically reconfigurable processor which uses direct interconnection between neighboring PEs, is designed and evaluated. The evaluation results show that the required semiconductor area for MuCCRA-D is 12% smaller than that of MuCCRA-1 by reducing wiring resource in the interconnection. Since higher clock frequency can be used, DCT, α-Blending, Bubble-Sort and SHA-1 implemented on the MuCCRA-D are 3.84 times faster than MuCCRA-1 at maximum .

Keywords: Dynamically Reconfigurable Processor, Interconnection Network

1. Introduction

In recent years, coarse-grained dynamically reconfigurable processor arrays (DRPAs) have received an attention as a flexible and efficient off-loading engine for various types of System-on-Chips (SoCs). Some devices are commercially available [1], [2], [3], [4], and some of them have been integrated into digital appliances.

In order to achieve better area- and power-efficiency compared with traditional field-programmable devices such as FPGAs, they incorporate the following properties; (1) a simple coarse grained processor consisting of an ALU, a data manipulator, a register file and other functional modules is used as a primitive processing element (PE) of an array, and (2) dynamic reconfiguration of a PE array which enables time-multiplexed execution is introduced. Some of them provide multiple sets of configuration data called hardware contexts, and switch them in one or a few clock cycles. A system with such a mechanism is called a multi-context DRPA.

Unlike common FPGAs, in which Look-Up-Tables (LUTs) with 4 or 5 inputs and the island-style interconnection are commonly used, there exist wide design choices in DRPAs, such as the PE granularity, the number of hardware contexts which can be switched dynamically, the total amount of wiring resource, and the size of PE array itself. Our performance evaluation results revealed that the optimal PE array size considering the area and power consumption is different by each application [5]. Thus, there is no all-around architecture in DRPAs, and the structure should be configurable or customizable for its main target application. Since DRPAs are assumed to be embedded into an SoC, customization of architectures will be done at the design time like a configurable processor.

The object of Multi-Core Configurable Reconfigurable Architecture (MuCCRA) project is to develop a design methodology and framework which generate highly configurable DRPAs for various target applications. On designing an architecture, an interconnection between PEs is an important factor which influences on both performance and cost. Our prototype chips in the project; MuCCRA-1 and MuCCRA-2[6] adopted an island-style interconnection similar to FPGAs for its flexibility. However, it has a tendency to enlarge maximum delay due to a long data path. The other general interconnection between PEs is a direct interconnection. In this style, a few dedicated channels are provided between PEs to transfer data. Although current DPRAs use either of interconnection styles, the evaluation and comparison based on the real layout and application have not been well done.

In this paper, we propose directly interconnected PE array architecture called MuCCRA-D and compare with MuCCRA-1 which adopted island-style interconnection.

The rest of paper is organized as follows. Section 2 introduces typical interconnection structures used in DRPAs. Section 3 describes MuCCRA-1 architecture. Section 4 introduces detail of the directly interconnected architecture, MuCCRA-D. We present an implementation result of two architectures in Section 5 and evaluation result of two architectures in Section 6. Section 7 concludes the paper.

2. Related Work

The existing DRPAs adopt either of two styles for interconnection networks in their PE array; an island-style connection and a direct interconnection. The island-style has a switching element on the intersection of vertical and horizontal global channels and it controls connectivity according to the configuration data. Data exchange between a PE and a global channel is done by using a connection block. This style is generally used in conventional FPGAs. For DRPAs, Chameleon CS2112[7], NEC Electronics' Dynamically Reconfigurable Processor (DRP)[2] and a part of PACT XPP[4] interconnection are categorized into this type. Although the island-style has an advantage of flexibility on PE-to-PE connections, there are two major problems: (1) the area of global routing and switches become large, and (2) the maximum operating frequency is degraded by the long maximum delay.

Interconnecting PEs directly is also widely used. Hitachi's Flexible Engine/Generic ALU array (FE-GA) [8] has 32 PEs including arithmetic logic units (ALU) and multipliers (MLT) which forms a 2-dimensional array, that is, each PE is connected with neighboring PEs directly. IMEC's Architecture for Dynamically Reconfigurable Embedded System (ADRES)[1] also has direct interconnections which connect 32-bit Reconfigurable Cells (RCs). ADRES has special channels that connect remote PEs. Since the direct interconnection style is able to transfer data to the connected PE quickly with a constant delay, the architecture adopting this style can operate with a high clock frequency. However, the direct interconnection has disadvantage of taking a few cycles to transfer data to remote PEs.

3. MuCCRA-1

In this section, we introduce the first prototype chip of Multi-Core Configuration Reconfigurable Array (MUCCRA) project called MuCCRA-1 uses an island style interconnection architecture.

3.1. Overview of MuCCRA-1 Architecture

Figure 1 shows PE array structure of MuCCRA-1. For multi-media processing, the granularity of the whole architecture is 24 bits, that is, all functional units and channels

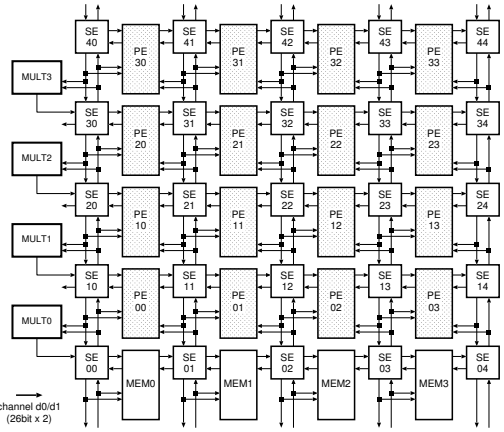

Figure 1. PE Array Architecture of MuCCRA-1

treat 24-bit data except wires for 2-bit carry. MuCCRA-1 has a 4×4 PE array, four multipliers (MULT) on the left side of array and four distributed memories (MEM) which has 24 bit $\times$ 256 entries on the bottom of array. An island-style interconnection structure like traditional FPGAs is adopted. That is, each PE is surrounded by programmable routing wire segments. Connection blocks are provided between PEs and global routing channels for sending or receiving to or from PEs. On the intersection of vertical and horizontal channels, a Switching Element (SE) is placed. The SE is a set of simple programmable switches in which an entering channel is connected to the other SEs. There are two channels for the global routing resources.

3.1.1. PE Structure of MuCCRA-1

Each PE has a programmable PE Core, connection blocks, and a context memory. In the PE Core as shown in Figure 2, like a lot of existing DRPA devices, a data manipulator called Shift & Mask Unit (SMU), an Arithmetic Logic Unit (ALU), and a register file (RFile) are provided. RFile has 26-bit (24 data bits + 2 carry bits) $\times$ 8 entries. A context memory which is 64-bit $\times$ 64-entry holds the configuration data that is distributed from the configuration memory at the beginning of the execution.

Each PE is connected with global routing wires via connection blocks. The connection blocks pick up the data from global routing wires, and distribute to all functional units of a PE Core. The operation of each functional unit and local intra-PE connection are statically defined by configuration data called a context.

3.1.2. SE Structure of MuCCRA-1

Each SE consists of two multiplexer-based programmable switches (SWs) as shown in Figure 3 and a context memory containing configuration data which specifies a destination of each SW. In Figure 3, the SW transfers an

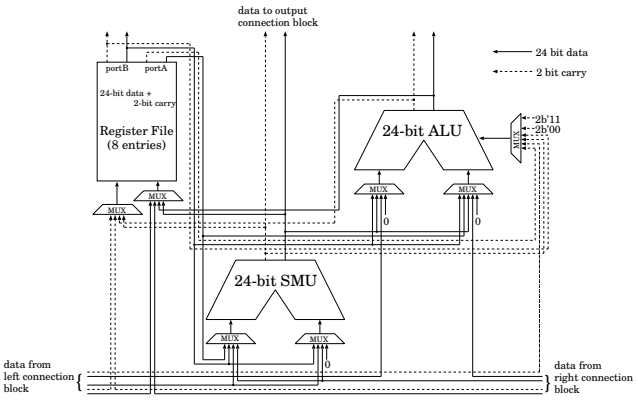

Figure 2. PE Core Architecture of MuCCRA-1

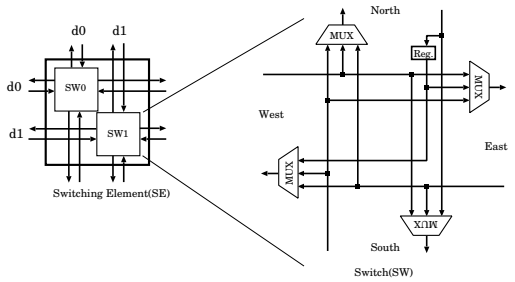

Figure 3. SE and SW Architecture of MuCCRA-1

entering data to desired output direction for each channel (d0, d1). Note that an input from the North is latched into a register in order to avoid combinatorial loops in the interconnection network, while inputs from other directions are unrestricted. The detail structure of the inter-PE connection network of the MuCCRA-1 is mentioned in the next subsection. In general, a multiplexer-based switch consumes larger area than a programmable switch of FPGAs in which a bidirectional data transfer using transfer gates. However, it is commonly used for island-style dynamically reconfigurable processor since short-term conflicts of outputs at dynamic reconfiguration can be avoided. The configuration data for a context of SE is 16-bit wide, and each context memory of SE can hold up to 64 contexts as similar to PE.

3.1.3. Inter-PE Connection Network

The inter-PE connection network of the MuCCRA-1 is illustrated in Figure 4. Each PE can select and take data from vertical global routing resources (d0, d1) in both sides of the PE via an internal input connection block called PICKIN. On the other hand, all outputs of ALU, SMU, and RFile of a PE can be transferred to horizontal channels in any direction via the output connection block (PICKOUT).

In order to avoid combinatorial loops, this network uses the same policy for deadlock avoidance as the Turn model [9]. The connection turns from the down direction to horizontal direction is only latched in an internal register of the

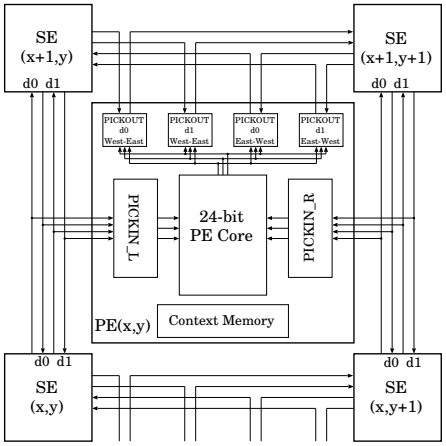

Figure 4. Inter-PE Connections

destination SE, while other connections are allowed without any restrictions. Similarly, data from an upper PE to a lower PE must be stored in a RFile of the lower PE, while outputs of the lower PEs are allowed to connect to inputs of upper PEs without any restrictions.

These restrictions are naturally fulfilled if the following typical datapath structure is mapped on to the PE array. The source data is read out from distributed memories, processed upward with several PEs, and computation results are written into RFiles or MEMs via feedback lines from the topmost SEs.

3.1.4. Context Switching Mechanism

Each PE and SE in MuCCRA-1 equip their context memory in which the configuration data for a particular operation is held. The central controller broadcasts a context pointer to all of the reconfigurable elements including PEs and SEs. The configuration data for a context is read out from the context memory according to the context pointer, and they are reconfigured in parallel. This type of dynamic reconfiguration is called a multi-context scheme, and a lot of current devices support this scheme. In the multi-context devices, the dynamic reconfiguration can be done in only one clock cycle by distributing the context memory into each reconfigurable element.

For high speed configuration data distribution, a multicast mechanism called RoMulTiC[10] is adopted. The context control and configuration data distribution mechanism are common in all MuCCRA chips, and cannot be changed except the size of context memory which influences the area of PE and SE.

4. MuCCRA-D

Since the maximum delay path of MuCCRA-1 is various because of the difference in the length of data path on the

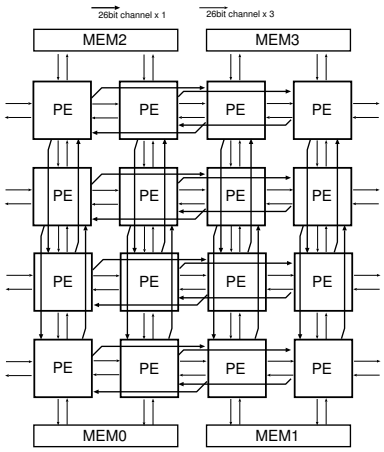

Figure 5. PE Array Architecture of MuCCRA-D

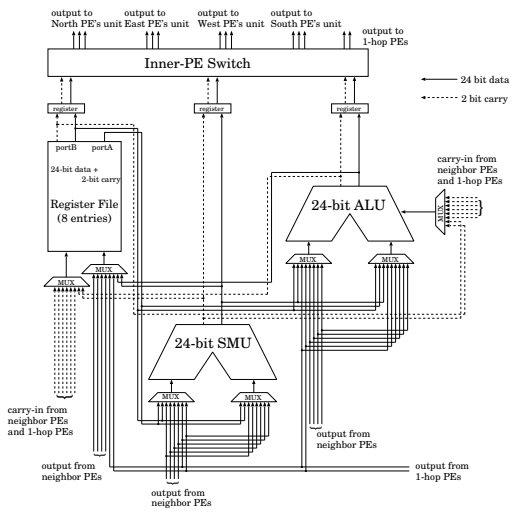

Figure 6. PE Core Architecture of MuCCRA-D

island-style interconnection, the maximum clock frequency varies by applications. In addition, it is noted that wire resource shortage occurs when PE utilization becomes high.

In this section, a dynamically reconfigurable processor MuCCRA-D with another interconnection type; direct interconnection is introduced. For comparison, MuCCRA-D has the same fundamental architecture with MuCCRA-1, but in order to fit its direct interconnection structure, some changes have been made.

4.1. PE Array Architecture

The architecture shown in Figure 5 illustrates the MuCCRA-D architecture which has a 4×4 PE-array and 4 distributed memories on top and bottom of the array. Distributed memories have 24 bit $\times$ 256 entries; that is, the same as those in MuCCRA-1.

As shown in Figure 5, each PE is connected to its nearest neighbors and also connected to the other PEs in the same row and same column to reduce the delay for transferring data to the remote PE. Unlike MuCCRA-1, every output of PE has a register, and the data to be transferred must be stored in it. Only with channels for neighboring PEs, data transfer to the distant PEs will require too many hops at the intermediate PEs. The similar channels are also provided in ADRES[1]. As described later, three independent channels are provided for the nearest neighbors while there is only a channel for one-hop distant PE.

4.1.1. PE Architecture

Each PE of MuCCRA-D consists of a programmable PE core, a context memory and an inner-PE switch that selects destination of output data from functional units.

Each PE core has the same 24-bit functional units as MuCCRA-1: ALU, SMU and RFile. Note that an ALU in the all PEs can execute a multiplier operation while MuCCRA-1 has four multipliers on an edge of the PE array. This change had made in order to reduce the overhead for transferring data to distant multipliers through direct interconnection channels.

As mentioned before, all outputs of functional units are buffered with registers. Stored data in the register is transferred to the PE in the next clock cycle according to the configuration data. These output registers are useful to reduce the critical path on the PE array, and to prevent from forming combinatorial circuit loops through the network.

Inputs of the ALU are selected from four nearest neighbors, two 1-hop distant PEs, the SMU and RFile in the same PE. Also, the SMU inputs receive data from four nearest neighbors, two 1-hop distant PEs or RFile in the same PE. RFile inputs are selected from ALU and SMU in the same PE, and data from a connected PE. These selection is controlled according to the configuration data of the PE.

The 64-bit configuration data of each PE in the MuCCRA-D decides operations of each functional unit and destination PE to which the data from functional units is transferred. Since the configuration data for inner-PE switch and selection of direct interconnected channels is almost the same as those for interconnect control information in island style interconnection, the amount of configuration bits is the same as those for MuCCRA-1. The configuration data is stored in a 64-bit $\times$ 64-entry context memory in every PE.

4.1.2. Inner-PE Switch

The inner-PE switch in the PE transfers output data from ALU, SMU and RFile to destination PEs designated by the configuration data. Each inner-PE switch has 14 output channels as shown in the top of Figure 6. Three channels are connected to each of four neighboring PEs respectively; that is, 12 channels are used. The remaining two channels

are used for 1-hop distant PEs in the horizontal and vertical direction. Three channels to a neighboring PE are connected with the input of three functional units : ALU, SMU and RFile, respectively. Through the inner-PE switch, an output of ALU, SMU and RFile is connected to one of these 14 channels. Thus, the configuration data of the inner-PE switch decides not only the target PE but also the target functional unit in the PE. For different functional units in the target PE, output data from the different functional unit in the source PE can be transferred simultaneously. Of course, more than two outputs from the same PE can not be transferred to the same functional unit in the target PE. In the inner-PE switch, data is transferred only to one destination, that is, data broadcasting is forbidden.

Although only a channel is provided for the 1-hop neighbor, the data can be selected as input of all three functional units of destination PE.

4.1.3. Distributed Memory

Distributed memories are placed at both top and bottom of the PE array. Each MEM is connected to the nearest two PEs as shown in Figure 5. MEM is a 2-read/1-write memory, that is, two data can be read out from different addresses at the same time, but only one data can be written exclusively. This is because writing data to memory requires data and address, and there is only two PEs that are connected to each memory. Either PE will make data or address which is decided from the configuration data. Two PEs can not read data from the same address at the same time, but there is a copy function which copies and transfers the data read out from one address to two connected PEs. Whether to copy the data from memory or not is also decided by the configuration data. These implementations are made to manage data more flexibly on directly interconnected PE array.

4.1.4. Control Mechanism of MuCCRA-D

The control mechanism of MuCCRA-D is almost the same as MuCCRA-1. There is no switching elements in the PE array of MuCCRA-D, therefore, the mechanism to transfer the configuration data to switching elements is removed.

5. Implementation

MuCCRA-1 and MuCCRA-D were designed individually onto a 5.18-mm square die in Rohm 0.18um CMOS technology with 189 I/Os. The same design tools were used for both architectures. The RTL models of the both architectures are described in Verilog-HDL. Synopsys Design Compiler 2006.06-SP2 and Cadence SoC Encounter 5.2 were used for logic synthesis and layout, respectively.

Table 1. Comparison of Cell Usage and Area

	MuCCRA-D	MuCCRA-1
Num. of Cells	132624	120166
Num. of Gates	987089	1122458
Area[mm^2]	9.55	10.86

Table 2. Area of module(mm^2)

	MuCCRA-D	MuCCRA-1
Ctrl.	1.00 (10.4%)	1.09 (10.1%)
MEM ×4	1.46 (15.3%)	1.45 (13.4%)
MULT ×4	-	0.17 (1.6%)
SE ×25	-	1.80 (16.5%)
PE ×16	7.09 (74.3%)	6.35 (58.4%)

Table 1 shows the number of cells and area of both architectures. Table 2 shows area of each module and its percentage. *Ctrl.* in Table 2 shows the area for the control module in both architectures.

From Table 2, area of a PE module accounts for more than half of both architectures. Since MuCCRA-D's PE provides inner-PE Switch and a multiplier inside, it leads its hardware amount larger than that of MuCCRA-1. However, MuCCRA-D provides no SEs which account for the second largest part of MuCCRA-1, therefore total area of MuCCRA-D is 12% smaller than MuCCRA-1.

Table 3 describes the length of wire segments used in both architectures. In Table 3, *Interconnection* is a total length of nets used in only the interconnection and *Average* is an average length of wire per net in the interconnection. *PE Array* is a total length of nets used in the whole architecture including interconnection, PE modules, and other modules. In regard to the segment length of interconnection on MuCCRA-D, 51% is for connection to nearest neighbors, 26% is for 1-hop connection and 23 % is for 2-hop connection. Since neighboring connections are dominant in MuCCRA-D, the average length is smaller than that in MuCCRA-1. As examined later, it enables to use a high frequency.

6. Evaluation of application

6.1. Application and Environment

This section describes an application design environment and evaluation results based on it. We implemented several

Table 3. Segment Length of Interconnection and PE array(*mm*)

	MuCCRA-D	MuCCRA-1
Interconnection	5869.1	6689.5
Average	1.14	1.58
PE Array	25163.0	23443.7

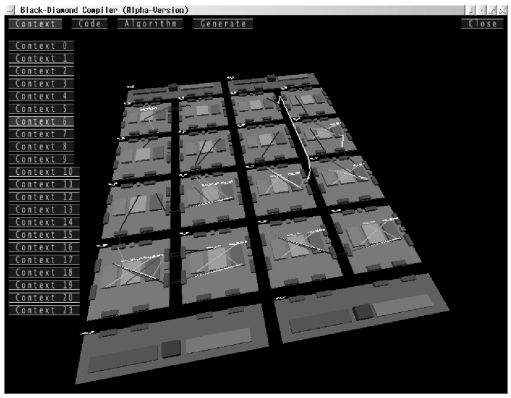

Figure 7. GUI of Black Diamond Compiler

applications on both MuCCRA-D and MuCCRA-1; discrete cosine transform (DCT) used in JPEG coder, a simple image processing program α-Blender (αBlend), a secure hash function (SHA-1) and bubble-sort (Bsort).

On the application implementation, we used GUI tool called *Comap* and *Black Diamond* compiler[11] developed for MuCCRA. In *Comap*, application implementation is done manually by selecting PE operation and connecting PEs in every context. After all selections and connections are finished, *Comap* generates configuration data for MuCCRA-D which is needed in the simulation. The Black-Diamond is a retargetable compiler which generates configuration data from a C-like description. The context scheduling, placement of operations into PEs, and routing are automatically done by the compiler. Programmers can check the results by using the GUI shown in Figure 7, and optimize them by inserting pragmas in the front-end description.

6.2. Application Implementation Results

We generated the configuration data of each application with the tools described above, and executed simulation using the post-layout netlist. Execution cycles and maximum delay are evaluated. MuCCRA-D buffers every PE outputs into the register and this guarantees a constant frequency. On the other hand, the length of data path on MuCCRA-1 varies, and this causes different frequency in each application.

Table 4 shows the number of contexts which are needed to implement each application and execution cycles (*Cycle*) with the block size (*Bsize*[bit]) of both MuCCRA-1 and MuCCRA-D. The block size is the data size which is used in each application. Figure 8 shows a performance improvement of MuCCRA-D. The performance in this figure is normalized by that of MuCCRA-1.

From Table 4, the number of contexts is increased in MuCCRA-D. This is because MuCCRA-D stores every outputs of PE in the register. DCT and αBlend are stream

Table 4. Number of Contexts and Execution Cycles

		MuCCRA-D		MuCCRA-1	
	Bsize[bit]	Context	Cycles	Context	Cycles
DCT	1024	56	253	39	192
αBlend	8192	11	1027	8	644
Bsort	720	18	4441	8	1801
SHA-1	512	24	727	12	418

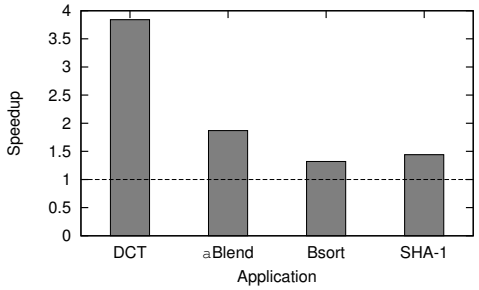

Figure 8. Application Speedup of MuCCRA-D

processing which requires a lot of multiplications, and they can be executed in the pipelined manner. So, in these applications, increasing number of contexts is not so large compared with MuCCRA-1. On the other hand, Bsort and SHA-1 need twice more contexts to implement on MuCCRA-D than MuCCRA-1. This comes from many branch operations used in both applications. In MuCCRA-D, it takes a few contexts to transfer the branch signal to special PE, which is located rightmost of the PE array, for controlling branch operation. An application like Bsort that has many branches and data moving is not effective on MuCCRA-D whose PEs are connected directly.

As the number of required context increases, the execution cycle grows. Although MuCCRA-D takes more cycles to execute than MuCCRA-1 as shown in Table 4, execution time on MuCCRA-D is smaller than that on MuCCRA-1. This is because MuCCRA-D operates with several times higher frequency than that with MuCCRA-1. The result shows MuCCRA-D is 3.84 times faster at maximum on DCT and 1.32 times faster at minimum on Bsort than MuCCRA-1. The execution speed of MuCCRA-D is about 7.75 times faster on DCT compared with the Texas Instrument's digital signal processor (TMS320C6713)[6] which works at 225MHz.

6.3. Evaluation of Energy Consumption

The energy consumption are evaluated by using Synopsys's Power Compiler with the switching probability extracted from the simulation. On the evaluation of the performance, the configuration data of each application and netlist that is generated after place and route are used.

Energy([nJ]) is a metric to compare MuCCRA-D and MuCCRA-1 which are calculated as a product of execution

time and power consumption(*Power*[mW]) in unit time. The supply voltage is set to be 1.8V.

Table 5 shows the ratio of power consumption in each module and Table 6 shows energy consumption when executing each application on both architectures. Each architecture is assumed to work at the maximum frequency clock.

Table 5. Ratio of Power Consumption in Each Module

	MuCCRA-D						
	Freq.	PE	SE	MULT	MEM	Ctrl.	Other
DCT	125MHz	89.8	-	-	2.0	1.5	6.7
αBlend	125MHz	76.1	-	-	4.7	3.3	15.9
Bsort	125MHz	73.3	-	-	4.8	3.7	18.1
SHA1	125MHz	86.8	-	-	2.4	2.3	8.5
	MuCCRA-1						
	Freq.	PE	SE	MULT	MEM	Ctrl.	Other
DCT	25MHz	69.1	12.5	3.6	3.2	1.8	9.8
αBlend	42MHz	65.0	11.8	4.1	4.0	2.3	12.8
Bsort	38MHz	63.4	13.1	4.4	3.7	2.4	13.0
SHA-1	20MHz	62.1	14.8	3.2	3.6	2.6	13.7

Table 6. Evaluation Results of Energy Consumption

	MuCCRA-D		MuCCRA-1	
	Power[mW]	Energy[nJ]	Power[mW]	Energy[nJ]
DCT	497.5	995.1	85.1	653.6
αBlend	211.0	1733.8	103.3	1583.9
Bsort	185.3	6582.7	95.9	4489.5
SHA-1	392.4	2275.7	50.6	423.0

The result shows that MuCCRA-D which has high frequency consumes much more power than that with MuCCRA-1. And this leads the high energy consumption in MuCCRA-D, although MuCCRA-D has smaller execution time than that of MuCCRA-1.

From Table 5, power consumption of PE in both architectures accounts for more than 60%. Control module (*Ctrl.*) only accounts for a few percent that is the same tendency as area, and this indicates that the power of controller (*Ctrl.*) is not dominant in both architectures. On the other hand, *other* including clock network and buffer inserted between PE and SE interconnection occupies relatively high ratios; 6 to 18%, This shows that power consumption on interconnection is not negligible in both architectures.

7. Conclusion

In this paper, we described two dynamically reconfigurable processors that have different interconnection networks; MuCCRA-D using a directly interconnection network and MuCCRA-1 using an island-style interconnection structure. Both architectures were designed and evaluated on area, wire-length, execution cycles of applications and power consumption.

Evaluation results show that MuCCRA-D is less than MuCCRA-1 by 12% on area. In regard to wire segment length, MuCCRA-D has less interconnections than that of MuCCRA-1. Considering the total wire-length, however, MuCCRA-D requires more wires. On application performance, MuCCRA-D achieved higher performance than that on MuCCRA-1, 3.84 times faster at a maximum. On the other hand, energy consumption of MuCCRA-D is larger than MuCCRA-1 due to its high operating clock frequency.

Acknowledgments:

This work is supported in part by Japan Science and Technology Agency(JST). The authors thank to VLSI Design and Education Center (VDEC).

References

[1] F. Veredas, M. Scheppler, W. Moffat, and B. Mei, "Custom Implementation of the Coarse-Grained Reconfigurable ADRES Architecture for Multimedia Purposes," in *Proc. of FPL*, Aug. 2005, pp. 106–111.

[2] M. Motomura, "A Dynamically Reconfigurable Processor Architecture," *Microprocessor Forum*, Oct. 2002.

[3] T. Sugawara, K. Ide, and T. Sato, "Dynamically Reconfigurable Processor Implemented with IPFlex's DAPDNA Technology," *IEICE Trans. on Information & System*, vol. E87-D, no. 8, pp. 1997–2003, May 2004.

[4] M. Petrov, et al., "The XPP Architecture and Its Co-simulation within the Simulink Environment," in *Proc. of FPL*, Aug. 2004, pp. 761–770.

[5] Y. Hasegawa, S. Abe, S. Kurotaki, V.M. Tuan, N. Katura, T. Nakamura, T.Nisimura, H.Amano, "Performance and Power Analysis of Time-multiplexed Execution on Dynamically Reconfigurable Processor," in *Proc. of RAW*, Apr. 2006.

[6] H.Amano, Y.Hasegawa, S.Tsutsumi, T.Nakamura, T.Nisimura, V.Tanbunheng, A.Parimala, T.Sano, and M.Kato, "MuCCRA Chips: Configurable Dynamically-Reconfigurable Processors," in *Proc. of ASSCC 2007*, Nov. 2007, pp. 384–387.

[7] X.Tang, M.Aalsma, and R.Jou, "A compiler directed approach to hiding configuration latency in Chameleon Processors," in *Proc. of FPL*, Sept. 2000, pp. 29–38.

[8] T. Takanobu et al., "Overview of Reconfigurable Processor FE-GA for Digital Media," in *Proc. of RECONF*, vol. 105, no. 451, Dec 2005, pp. 37–42.

[9] C. J. Glass and L. M. Ni, "The Turn Model for Adaptive Routing," *Proc. of Int'l Symp. on Computer Architecture*, pp. 278–287, 1992.

[10] V. Tanbunheng, M. Suzuki, and H. Amano, "RoMultiC: Fast and Simple Configuration Data Multicasting Scheme for Coarse Grain Rec onfigurable Devices," in *Proc. of FPT*, Dec. 2005, pp. 129–136.

[11] V. Tanbunheng and H. Amano, " DisCounT: Disable Configuration Technique for Representing Register and Reducing Configuration Bits in Dynamically Reconfigurable Architecture," in *Proc. of SASIMI 2007*, Oct. 2007.