Exploring the optimal size for multicasting configuration data of Dynamically Reconfigurable Processors

T. Nakamura, T. Sano, Y. Hasegawa, S. Tsutsusmi, V. Tunbunheng, and H. Amano
Department of Information & Computer Science, Keio University
3-14-1 Hiyoshi, Kohokuk-ku, Yokohama, 223-8522 Japan
email: muccra@am.ics.keio.ac.jp

Abstract

The configuration data transfer time of a dynamically reconfigurable processor often bottlenecks the hardware context switching time and degrades its computation performance. In order to reduce data transferring time from a central memory to hardware context memory modules in all Processing Elements (PEs) and Switching Elements (SEs), a multicasting mechanism called RoMultiC (Row-Muticast Configuration) was proposed. However, the original RoMultiC used the whole PE or SE as a unit of multicast, the reduction of transfers is limited. Here, the trade-off between the granularity of multicast and hardware increase are evaluated, and the best way to make the multicast bit-map is explored. Evaluation results show that time for transfer is reduced up to 42% compared with the original RoMultiC with only 2% hardware overhead.

1. Introduction

Coarse-grained dynamically reconfigurable processor arrays (DRPAs) have received an attention as a cost-efficient off-loading engines for media-rich applications on a System-on-a-chip (SoC). Some devices are commercially available, and some of them are widely used in digital appliances[1].

In such SoCs integrating an MPU and DRPA into a small chip area, a high speed dynamic configuration scheme of the DRPA, that is, changing the configuration data for each processing element and interconnect mechanisms of the DRPA, is essential to accommodate a variety of applications. Particularly, configuration data transfer time may often be a bottleneck of the system performance in such reconfigurable systems.

To address this problem, a multicast configuration scheme called RoMultiC (Row-Multicast Configuration)[2] for DRPAs has been proposed. Simular to the configuration compression scheme using Wildcard Registers[3] proposed for FPGAs, RoMultiC exploits the fact that there exist identical configuration data of Reconfigurable Elements (REs), such as Processing Elements (PEs) and Switching Elements (SEs), in an application with high parallelism. RoMultiC has been employed in MuCCRA[4] and WPPA[5], and scheduling methods to fix the order of multicasting configuration bit-map has been also proposed[6].

There is another possibility to improve the efficiency of RoMultiC by making the best use of PE structure in DRPAs. Unlike the FPGA, each PE is consisting of several components; ALU, data manipulator and register files. In conventional RoMultiC, the configuration data is only transferred to multiple PEs at a time only when all these components require the exact same configuration data. By using separated bit-map for each component, the possibility of multicast will increase. On the other hand, the extra buses and multiplexers needed for increasing number of bit-maps may increase the chip area.

In this paper, the trade-off between the granularity of multicast and increasing of the hardware is evaluated, and the best way to give the multicast bit-map is explored.

2. RoMultiC

All reconfigurable devices are needed to transfer configuration data to the configuration memory modules. Unlike FPGAs, dynamically reconfigurable processors, in which configuration data is needed to be sent quickly tends to provide a common large configuration memory inside the chip. Configuration data is transferred from such common configuration memory to configuration memory modules each of which is provided in each PE or SE.

Most dynamically reconfigurable processors employ a sequential configuration scheme giving a serial address to configuration data memory and transferring configuration data to the address in order. For example, DRP-1[7], a coarse-gained dynamically reconfigurable processor core

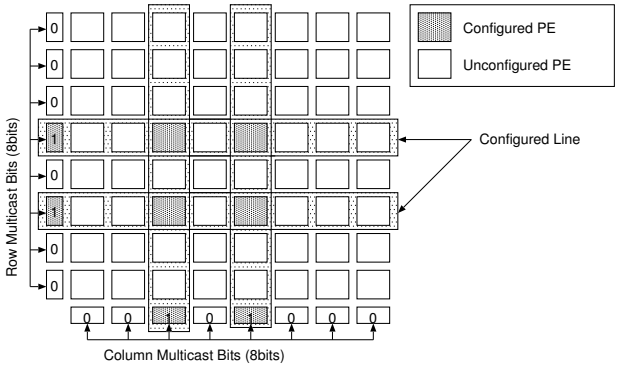

Figure 1. RoMultiC

released by NEC Electronics, equips multicontext configuration scheme and has a distributed context memory in each PE. The context memory has a unique address with which configuration data is written through the configuration data bus sequentially. In this method, configuration speed is limited with the configuration data bus width.

XPP[8] from PACT XPP Technologies, Inc. is also a coarse-grained dynamically reconfigurable processor consisting of 2D PE array. XPP employs the Wave Reconfiguration technology which configures PEs according to the data flow on the array. Configuration data are transferred from memory modules in parallel to decrease configuration time, and it supports partial configuration for unused PEs. Although the method allows the overlap between computation and configuration, it is hard to be used in multicontext DRPAs.

In the template reconfiguration[9], configuration data in a centralized configuration memory is shared with multiple contexts. Although it both saves the required memory and time for configuration delivery, an address transform table which often forms the critical path is needed to share the configuration data.

RoMultiC (Row-Multicast Configuration) uses row and column multicast bits in order to specify configured targets instead of mapping a sequential address to PEs. As shown in Fig.1, configuration data is received by the PEs where the row and column multicast bits are both '1'. By multicasting configuration data, configuration data transfer time can be substantially shortened.

The amount of multicast bits is greater than that of address bits for serial addresses, and hence, the required memory size seems to become large. In most cases, however, since the configuration data transferred with multicast are shared with multiple PEs, the total configuration data will be greatly reduced compared to the case that whole configuration data are held without sharing.

The configurable area is restricted in a rectangle, but by devising the transfer order, any complex configuration pattern can be configured because the configuration data can be overwritten and the latest configuration data are valid. With this feature, RoMultiC can reduce further configuration data transfer cycles by scheduling their order and configuration patterns. Scheduling methods to find the efficient order of multicasting bit-maps has been proposed[6]. By using such methods, it can reduce the number of cycles for multicasting 40% at maximum. From the viewpoint of implementation, RoMultiC does not require large extra hardware compared with the common configuration delivery method, since only a wire from the bit-map register is required in a row and a column of the PE array. Thus, it is adopted in MuCCRAs[10] and WPPA[5].

3. The granularity of RoMultiC

There is another possibility to improve the required clock cycles for multicasting in RoMultiC. In coarse-grained DRPAs, a PE is consisting of several components; ALU, data manipulator register files and multiplexers for connecting components, and the configuration data commonly becomes 50 bits - 200 bits. RoMulTiC can multicast configuration data only when all configuration bits are completely the same. For PEs with long configuration bits, it is difficult to find the completely matched pattern. In practical applications, a few bits for multiplexers sometimes differ even if all other components work similarly.

In order to address this problem, we separate each field of configuration data and search the same pattern to be multicasted as shown in Fig 2. The possibility to find the same pattern is increased when small field is used. Configuration data for PEs whose function is almost the same but only interconnection to outside is different can be efficiently done by multicasting the common part first, then sending different small field later. The requirement of the common memory entries used to store the configuration data can be also saved, since shared data is not needed to stored in multiple entries.

On the other hand, as also shown in Figure 2, multicast bitmap is needed for all fields which are separately multicasted. Since the size of multicast bitmap is increased as the size of array, the increasing of multicast bitmap for each field requires hardware overhead especially for a large PE array. Thus, the granularity to attach the bitmap must be carefully selected considering the reduction of configuration data transfer and the increase of the multicast bitmap.

4. A target architecture: MuCCRA-2.32b

In order to evaluate the trade-off on the granularity of multicast, practical and typical DRPA must be assumed.

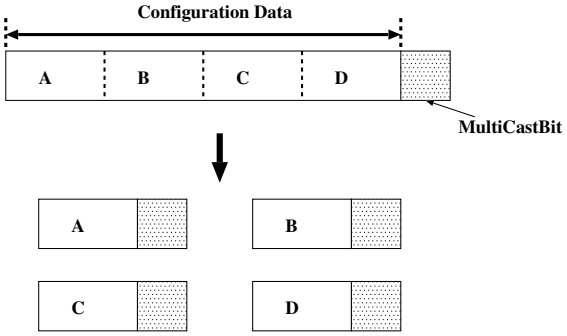

Figure 2. An example of fine grain RoMultiC

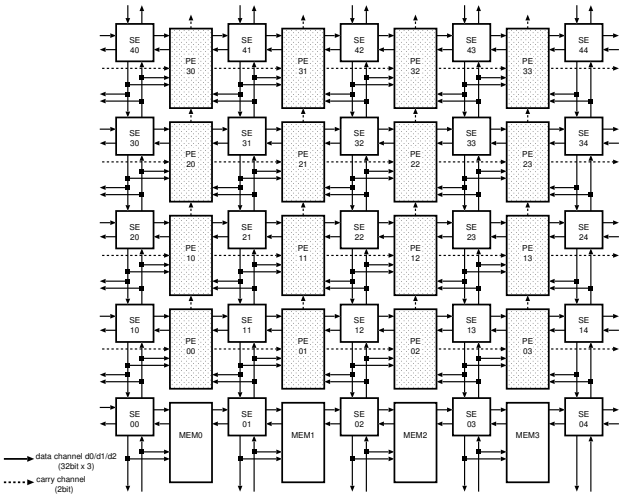

Figure 3. PE Array Architecture

Here, a model architecture, MuCCRA-2.32b (Multi-Core Configurable Reconfigurable Array-2.32b) is introduced for this purpose. MuCCRA-2.32b is a small scale, multicontext DPRA designed for analyzing various kinds of trade-off on DRPA architectures. For example, in [11], the consuming power of MuCCRA-2.32b is analyzed with some improvement methods. It is almost the same as MuCCRA-2[10] which is now working on a real chip, but chip dependent design optimization is eliminated for analyzing a common design. The evaluation and modification can be done using the real layout of the chip design described later.

4.1. PE Array

As shown in Figure 3, MuCCRA-2.32b has a 4 $\times$ 4 PE array and four distributed memories (MEMs) which have 32 bit $\times$ 256 entries on the bottom of array. For multi-media processing, the granularity of the whole architecture is 32 bits, that is, all functional units and channels treat 32-bit

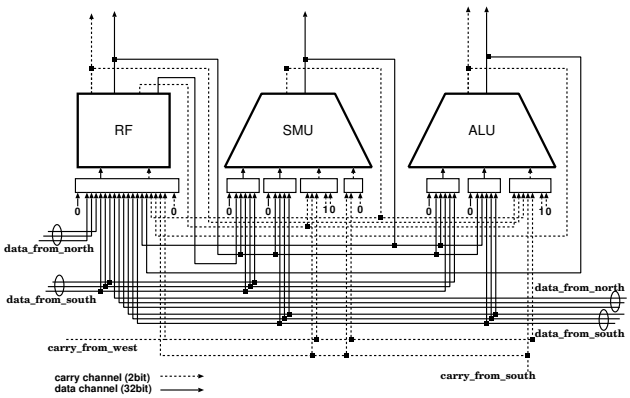

Figure 4. PE Core Architecture

data except wires for 2-bit carry. Task Configuration Controller (TCC) and Context Switching Controller (CSC) are provided to manage task and context switching.

An island-style interconnection structure like traditional FPGAs is adopted for connecting PEs and MEMs. 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 link is connected to the other SEs. There are three channels for the global routing resources.

4.2. PE Structure of MuCCRA-2.32b

Each PE is consisting of a programmable PE Core, connection blocks, and a context memory. In the PE Core as shown in Figure 4, 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 34-bit (32 data bits + 2 carry bits) $\times$ 8 entries. A context memory which is 64 bits $\times$ 32 entries 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 defined by configuration data stored in the context memory.

4.3. Interconnection Network

The inter-PE connection network of the MuCCRA-2.32b consists of Connection Blocks and SEs. Each PE can select and take data from vertical global routing resources (d0, d1,

d2) 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 links in any direction via the output connection block (PICKOUT).

Output data from a PICKOUT of a PE is transferred to PICKINs of other PEs through SEs. Each SE consists of four multiplexer-based programmable switches (SWs) and a context memory containing configuration data which specifies a destination of each SW. The SW transfers an entering data to desired output direction for each link (d0, d1, d2). 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 configuration data for a context of SE is 15-bit wide, and each context memory of SE can hold up to 32 contexts as similar to PE.

4.4. Context Switching and Task Control

In MuCCRA-2.32b, 32 hardware context are controlled by CSC (Context Switching Controller) which uses a simple context counter. Reconfigurable modules including PEs and SWs load configuration data from context memory according to a context pointer generated from the context counter in the CSC. It is also reconfigurable module, and the configuration data for context control is also loaded for itself. The context switching is done as follows; (1) the next context number is fixed, (2) the context pointer is transferred to PE array, and (3) the configuration data in the context memory is read out. Since MuCCRA-2.32b has 32 entries in each context memory, 32 hardware context can be stored and switched without delay.

The context counter is simply incremented when the branch is not specified or not taken in the context. In MuCCRA-2.32b, the branch address and branch condition signals are computed in the PE array and sent from the special PE to the CSC. If the branch is taken, the branch address is added with the context pointer in the CSC. By using the mechanism, a table jump according to the computation results can be implemented as well as simple loop structures.

Since MuCCRA-2.32b is a multicontext DRPA, the context switching is done with a clock cycle. However, the configuration data must be transferred from the configuration memory to each context memory module before the application starts.

TCC (Task Configuration Controller) provides 1K depth central configuration memory for storing configuration data for each task, and the configuration code is multicast using RoMultiC to the context memory modules in PEs and SEs. During the configuration transfer, the task cannot start and it can be a bottleneck of the system. Although virtual hardware technique enables to overlap the execution and configuration data transfer, the configuration data transfer time often cannot be perfectly hidden[12] in some applications.

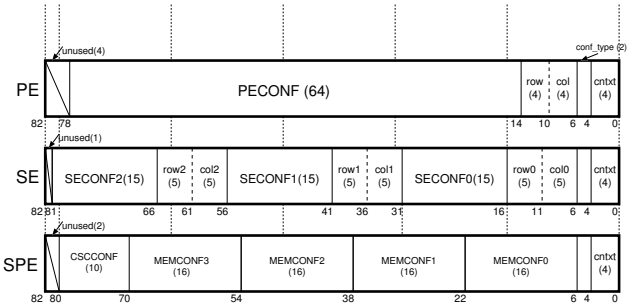

Figure 5. Configuration data types

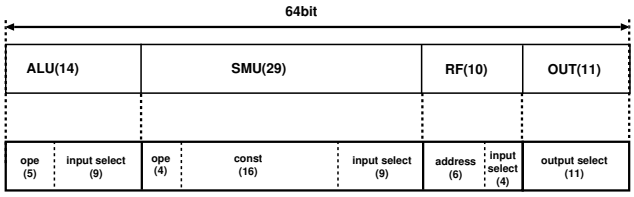

Figure 6. Field structure of PE configuration data

4.5. Configuration data

As shown in Figure 5, three different types of configuration data are used; for PE, for SE and for SPE. The configuration data for SPE defines the context control for conditional branches of the context pointer. The configuration data is followed by the bits for identification of types (conf_type), context number (cntxt), and the multicast bit map for RoMultiC (row/col) described lator. SE needs only 15bits configuration data for each module, thus three sets are transferred in parallel in order to save the number of data transfer. The detail field of configuration data for PE is shown in Figure 6.

The configuration data for PE is consisting of 64bits; 14bits for ALU operation, 29bits including 16bits constant for SMU operation, 10bits for RFile unit, and 11bits for output selection (OUT).

4.6. Implementation

MuCCRA-2.32b was designed for a 5.00-mm square die in Aspla 90nm CMOS technology. The area of MuCCRA-2.32b core is almost 4mm $\times$ 4mm. The RTL model is described in Verilog-HDL. Synopsys Design Compiler

Table 1. Specification of target applications

	CNTXT	PE util	PE conf	RoMultiC
DCT	29	63.8	277(67.1)	413
SHA-1	8	73.4	88(65.2)	135
DWT	8	21.2	38(52.8)	72
FFT	13	49.5	94(63.5)	148
AES	32	73.0	325(63.6)	511

2006.06-SP2 and Synopsys Astro 2007.03-SP3 are used for logic synthesis and layout, respectively.

Here, PE_CMEM and SE_CMEM stand for the configuration memory modules used to store the configuration data for PE and SE, respectively. MEM is distributed memory modules for storing data for computation as described before.

5. Trade-off analysis

5.1. Target applications

Here, RoMultiC is applied to MuCCRA-2.32b in various grain, and the cycles needed for configuration is evaluated. Five target applications[13]: 2 dimensional DCT (Discrete Cosine Transform), SHA-1 (Secure Hash Algorithm 1), 1 dimensional DWT (Discrete Wavelet Transform) FFT (Fast Fourier Transform) and AES (Advanced Encryption Standard) are adopted.

A retargetable compiler for DRPA called Black-Diamond[14] is used for application design. It compiles programs described in a C-like language, makes place-and-routing for the target DRPA, and generates the configuration data. Note that in RoMultiC, the multicast bitmap is overwritten with the one transferred later, and the number of multicast can be reduced by scheduling the order of multicast. Here, the fundamental scheduling called Pattern Separation Scheduling[6] is used.

Table 1 shows the number of contexts (CNTXT), the average utilization of PE in a context (PE util, %), and the number of cycles for transferring the configuration data when the original RoMultiC is utilized (RoMultiC).

5.2. Fixed fielding and functional fielding

Here, two field selection methods; fixed fielding and functional fielding are tried, and the number of configuration cycles and the total amount of configuration data are evaluated. In the fixed fielding, the configuration data for each PE is divided into fixed size fields, and the multicast bitmap is attached to each field. Here, four sizes: 8bit, 16bit, 24bit and 32bit are evaluated. In the functional fielding, the configuration data is divided by the functional boundaries, and the multicast bitmap is attached. Here F1 in Figure 7 and F2 in Figure 8 are evaluated. The fine grain configuration multicast is only applied to PE configuration data. Table 2 shows the length of each field divided by F1 and F2.

Table 2. The length of each field

	ALU	CNST	SMU	RF	OUT
F1	14	-	29	10	11
F2	14	16	13	10	11

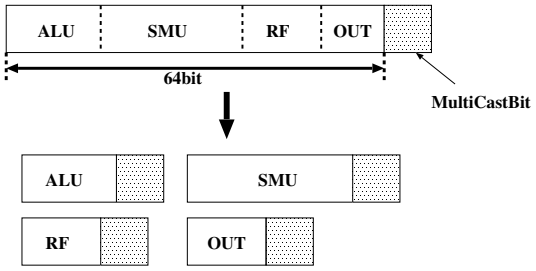

Figure 7. Functional fielding: F1

5.3. Fixed position

Here, the configuration bus is assumed to be 64bits, and a bitmap is attached to each field. Figure 9 shows an example of using 16bits fixed fielding. Since a field carries the fixed part of configuration data, this method is called the fixed position multicast (Fixed-Pos).

The cycles reduction ratio for configuration data transfer compared with the traditional RoMulTiC using 64bit field is shown in Figure 10. While the configuration data is reduced by increasing the possibility of multicast, the additional bitmap increase the total bits transferred in a cycle. The total configuration data to be transferred is shown in Figure 11.

Figure 10 shows that with 8-bit fixed fielding, the number of configuration transfer cycles can be reduced 20% at

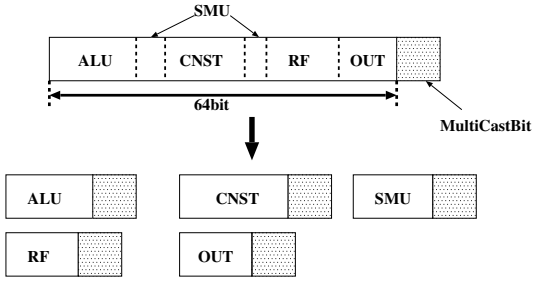

Figure 8. Functional fielding: F2

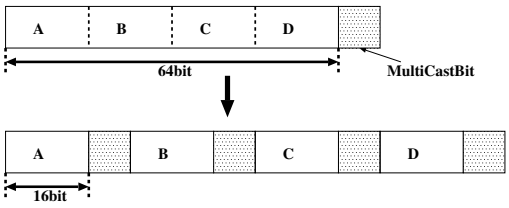

Figure 9. An example of attaching bitmaps (16bit fixed fielding)

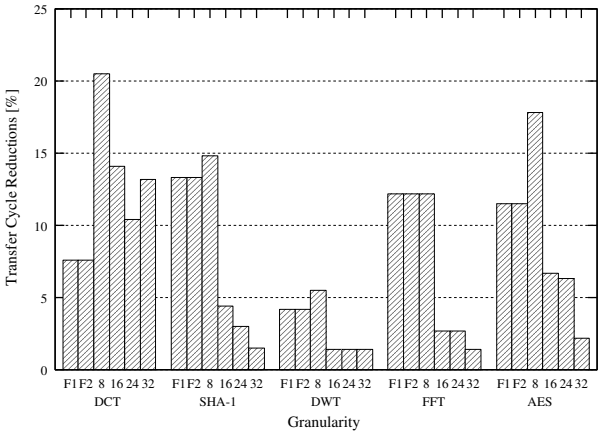

Figure 10. The transfer cycle reduction ratio(Fixed-Pos)

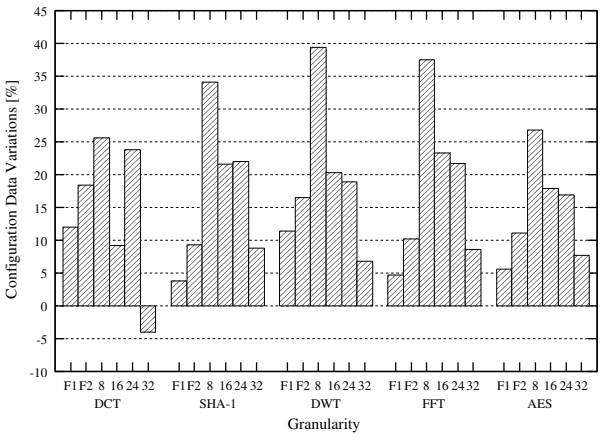

Figure 11. The size of configuration data(Fixed-Pos)

maximum. However, by using more than 16-bit fielding, the reduction ratio is less than 10% except DCT in which similar configuration pattern can be easily found. On the other hand, Figure 11 shows that the amount of configuration data is increased up to 40% except the case of DCT with 32-bit fielding.

This comes from the fact that the configuration data to be transferred is depending on the largest number to be transferred in all fields. For example, in Figure 9, even if a lot of same configuration data were found in A, B and C, the number of transferring configuration data is not reduced when the field D is different in all PEs. In this case, the configuration data, whose field D is only effective and others are filled with padding, is transferred many times.

5.4. Free position

By providing multiplexers for each part of bus corresponding to fields, the configuration data can be transferred. In order to control the multiplexer, position bits are required for each field as shown in Figure 12. This method to multicast the configuration data is called the free position here.

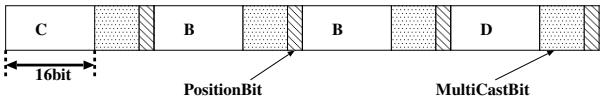

Figure 12. An example of the free position (16bit fixed fielding)

Unlike fixed position, the order and type of each field can be decided freely. In this example, fields are used for C, B, B and D, respectively. Position bits shows the location of the field in 64bits data. In functional fielding F1 and F2, the field length is set to be the largest fields. That is, in the case of F1, two 29bits are used. If the field is used for 10bit RF, the remaining 19bits just carries the padding bits. Thus, the free position accompanies some overhead for functional fielding.

Similar to the case of fixed position, cycles reduction ratio for configuration data transfer compared with the traditional RoMulTiC using 64bit field is shown in Figure 13 and the configuration data size is shown in Figure 14, respectively.

Figure 13 shows that the number of cycles can be much reduced compared with the results with the fixed position. In DCT and AES which can find a lot of similar pattern in configuration data, 8-bit fixed field achieves more than 40% reduction. Although the PE utilization of AES is larger than that of DCT, the parallel execution is efficiently done in DCT. Thus, the cycle reduction of DCT is more than that of AES. Although the reduction ratio is degraded with larger bit fields, more than 10% reduction is achieved even with 32-bit except DWT.

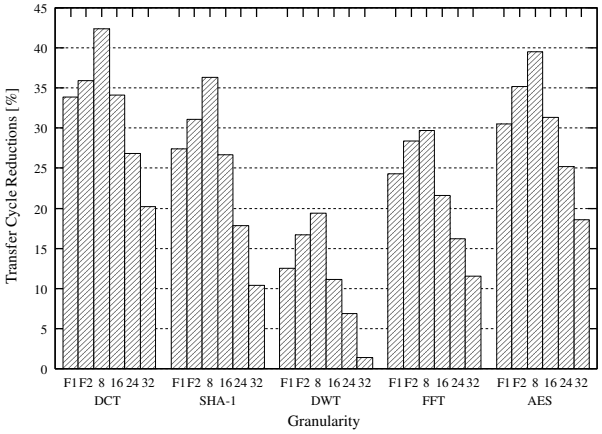

Figure 13. The transfer cycle reduction ratio(Free-Pos)

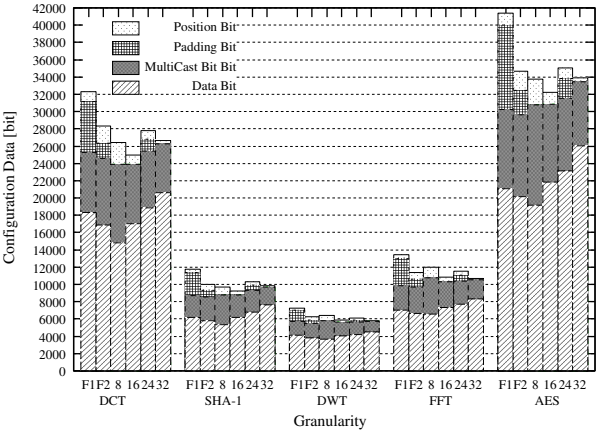

Figure 15. The breakdown of configuration bits (Free-Pos)

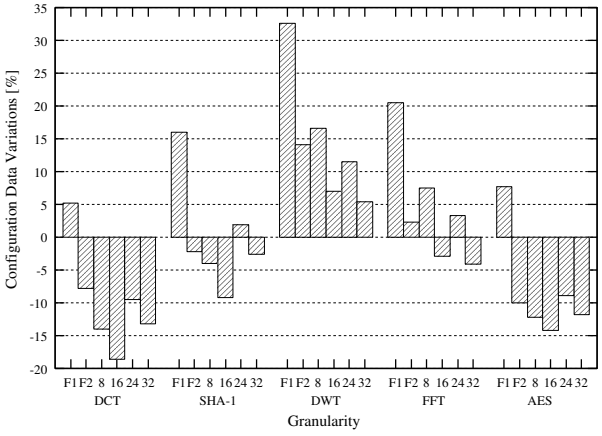

Figure 14. The size of configuration data(Free-Pos)

Although the functional fielding has a certain overhead, it achieves reasonable reduction ratio of transfer cycles. This comes from the fact that there are a lot of functional modules executing the same operation in a context. F2 which uses shorter fields is better than F1 at 4% in average. In general, when the free position is used, the shorter fields are advantageous because of its flexibility.

Figure 14 shows that configuration data can be reduced in a half cases. F1 with a large overhead of miss-aligned fields increases the configuration data in all cases. In order to analyze the data amount in detail, the breakdown of the configuration data required with the free position is shown in Figure 15. When shorter fields are used, the amount of bitmaps increases. From the viewpoint of cost per performance, 16-bit fixed field is advantageous. F1 requires a large amount of bits for padding, and F2 is advantageous clearly with the free position.

5.5. Evaluation of cost

The overhead of using short fields is as follows: (1) The bus width must be stretched for additional bitmaps, and additional logic for control is needed. (2) The multiplexers are needed to each field for selecting data in the free position. (3) The register is needed to cope with the variable field in the functional mode. (4) The amount of configuration memory modules will be increased for storing additional bitmaps.

The last item was evaluated with the configuration data size shown in the previous subsection. In order to evaluate the area overhead, MuCCRA-2.32b with proposed fielding is designed and synthesized by using the same process and tools for the original design. The results are shown in Table 3. "Original" shows the original design with a single Ratio shows the area overhead to the total area of the original MuCCRA-2.32b.

The 8-bit fixed fielding requires a certain overhead, although the cycle reduction ratio is the largest. However, the area overhead is only 2% at maximum. Note that the evaluated area dose not include for increasing wires. The impact of increasing wires is hard to be evaluated, since it often makes impossible to finish automatic place-and-routing, but sometimes there is no influence. In this case, the wire must be routed to all SEs and PEs distributed in the PE array, and impact is large. Considering this overhead of wires, 16-bit fixed fielding/free position or F2 functional fielding/free position is advantageous in most cases.

Table 3. The area overhead

Granularity	Area increased (μm^2)	Bus Width (bit)	Ratio (%)
Original	-	72	0
8-Fixed	1142	128	0.055
16-Fixed	587	96	0.0284
24-Fixed	280	88	0.0136
32-Fixed	125	80	0.00605
8-Free	45198	128	2.19
16-Free	12136	96	0.588
24-Free	4907	88	0.238
32-Free	3878	80	0.188
F1-Fixed	587	88	0.0289
F2-Fixed	644	96	0.0312
F1-Free	8417	88	0.410
F2-Free	587	96	0.0284

6. Conclusion

The trade-off between the granularity of configuration data multicast and hardware increase are evaluated with a dynamically reconfigurable processor MuCCRA-2.32b. Fixed-fielding and functional fielding are tried both with fixed position and free position. Evaluation results show that time for transfer is reduced up to 42% compared with the original RoMultiC with only 2% hardware overhead. Considering the overhead of configuration bus width, it appears that 16-bit fixed fielding/free position or F2 functional fielding/free position is advantageous in most cases.

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] K.Kurose and et al., "A 90nm embedded dram single chip lsi with a 3d graphics, h.264 codec engine, and a reconfigurable processor," in *Hot Chips 16*, 2004.

[2] 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.

[3] S.Hauck, Z.Li, and E.Schwabe, "Configuration compression for the xilinx xc6200 fpga," in *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, Vol.18, No.8*, Aug 1999, pp. 1107–1113.

[4] Y. Hasegawa and et.al, "Design Methodology and Trade-offs Analysis for Parameterized Dynamically Reconfigurable Processor Arrays," in *Proc. of FPL 2007*, Sept. 2007.

[5] D. Kissler and F. Hannig and A. Kupriyanov and J. Teich, "A highly parameterizable parallel processor array architecture," in *IEEE International Conference on Field Programmable Technology 2006 (FPT 2006)*, December 2006, pp. 105–112.

[6] S.Tsutsumi and et.al., "Overwrite Configuration Technique in Multicast Configuration Scheme for Dynamically Reconfigurable Processor Arrays," in *Proc. of FPT*, Dec. 2007, pp. 273–276.

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

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

[9] M.Suzuki and et. al, "A Cost-Effective Context Memory Structure for Dynamically Reconfigurable Processors," in *Proc. of the RAW2006*, Apr. 2006.

[10] H.Amano et al., "MuCCRA Chips: Configurable Dynamically-Reconfigurable Processors," in *Proc. of ASSCC 2007*, Nov. 2007, pp. 384–387.

[11] T.Nishimura and et. al. , "Power reduction techniques for Dynamically Reconfigurable Processor Arrays ," in *Proc. of Int'l Conf. on Field Programmable Logic and Application (FPL)*, Sept. 2008.

[12] H.Amano, S.Abe, Y.Hasegawa, K.Deguchi, M.Suzuki, "Performance and Cost Analysis of Time-multiplexed Execution on the Dynamically Reconfigurable Processor," in Proc. of the FCCM 2005. Springer, Berlin, April 2005.

[13] M.Suzuki, et.al., "Stream Applications on the Dynamically Reconfigurable Processor." Proc. on IEEE ICFPT2004, Dec. 2004.

[14] 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.