CONFIGURATION WITH SELF-CONFIGURED DATAPATH: A HIGH SPEED CONFIGURATION METHOD FOR DYNAMICALLY RECONFIGURABLE PROCESSORS

Toru Sano, Yoshiki Saito and Hideharu Amano

Department of Information and Computer Science, Keio University
3-14-1 Hiyoshi, Kouhoku-ku, Yokohama 223-8522, Japan
muccra@am.ics.keio.ac.jp

ABSTRACT

Configuration with Self-configured Data Path (CSDP) is a high speed configuration data loading method for Dynamically Reconfigurable Processors (DRPs). By using a prepared configuration data, a network for computation in DRPs can be used as a configuration data path when the computation is stalled and the controller requires the configuration data transfer. Design and implementation of a DRP called MuCCRA-3.32b with CSDP demonstrates that the time for computation can be reduced to a half with only 1.3% hardware overhead. Energy consumption for configuration data loading can be also reduced.

1. INTRODUCTION

Coarse-grained Dynamically Reconfigurable Processors (DRPs) have received an attention as a power-efficient accelerators for media-rich applications on a System-on-a-chip (SoC). Some devices are commercially available [1, 2, 3, 4, 5, 6], and some of them are widely used in digital appliances[7][8].

DRPs are classified into two types based on their dynamic configuration schemes: multi-context style and configuration data delivery. In the former method, components of DRP; Processing Elements (PEs) and Switching Elements (SEs) have their own local context memories to store multiple configuration data set, and switch the configuration with a clock cycle by reading data from them. Since the capacity of the context memory is limited from 4 to 32 in most multi-context DRPs, the configuration data for the next task must be transferred from outside the chip or centralized memory.

Another method is to deliver the configuration data to each component basically one by one from centralized configuration memory module located inside the chip. Although it often takes hundreds clock cycles to deliver the configuration data, the cost for providing a context memory in each component can be avoided.

In both methods, the configuration data transfer performance is a critical factor. Even in multi-context DRPs, the operation must be stalled until the configuration data for the next task is finished to be loaded.

Various types of methods have been proposed for hiding the latency or enhancing the speed of configuration data transfer. The overlapping time for execution and configuration data transfer has been investigated more than 15 years[9] and the mechanism is available in some chips[8]. However, in some applications, the time for configuration becomes much longer than that for the configuration transfer, the stall is still required. High speed data transfer methods using multicasting[10] is also an efficient to increase the speed but the efficiency is strongly depending on applications.

Here, a simple and powerful configuration transfer method called Configuration with Self-configured Data Path (CSDP) is proposed and evaluated. In this method, the network for the execution is configured and used for transferring the configuration data. Although the execution is completely stopped in the configuration, the time for transferring can be much reduced.

2. RELATED WORK

In order to achieve high speed dynamic reconfiguration, high speed configuration data transfer is essential even for multi-context DRPs which can switch their hardware context in a clock cycle. Since the area for the context memories often doubles the area of PE array, the context number that can be stored into the context memory is strictly limited. For example, DAPDNA-II and FE-GA provide four contexts, DRP-1 has 16 and STP engine and ADRES have 32. The loading time from centralized memory outside or inside the chip to the context memory often becomes the bottleneck of the system, since the execution must be stalled during the data transfer.

In order to address this problem, two approaches have been investigated: hiding the loading time and high speed data transfer. The former method enables the overlapping execution and data loading, that is, while the executing application with a part of contexts, the configuration data for the next task is transferred and stored to unused context memory space. If all configuration data for the next task have been loaded before finishing the current task, the task switching can be done without stall. This method, often called a virtual hardware, has been proposed in the early 90s'[9], and now is available in the commercial product[8]. The problem of this method is that the DRP is needed to provide enough size of context memory to store the number of contexts for two tasks. It is often difficult for implementing complicated tasks even with using 32 contexts[11].

The latter method is to increase the speed of configuration data transfer by making the best use of multicast. A multicast configuration scheme called RoMultiC[10] for DRPs has been proposed. Similar to the configuration compression scheme using Wildcard Registers[12] proposed for FPGAs, RoMultiC exploits the fact that there exists identical configuration data of Reconfigurable Elements (REs), such as Processing Elements (PEs) and Switching Elements (SEs), in an application with high parallelism. It has been employed in MuCCRA[13] and WPPA[14], and scheduling methods to fix the order of multicasting configuration bit-map has been also proposed[15]. The size of configuration data for multicasting is also analyzed[16]. These methods are efficient only when there are a lot of the same patterns to be transferred, and highly dependent to the characteristics of applications.

3. CONFIGURATION WITH SELF-CONFIGURED DATA PATH (CSDP)

3.1. Concept

Most of DRPs provide a network, which can be used for broadcasting or multicasting data to all PEs and SEs during the execution. Such networks for forming execution datapath can be also used to transfer the configuration data. Configuration with Self-configured Data Path (CSDP) configures and uses such a network to load the configuration data. The following two operations are required for implementing the CSDP.

- A prepared configuration data must be set for SEs to form the broadcast network from the centralized configuration memory to context memory modules in each PE.
- The data selector which enables to select the configuration data from multiple configuration paths is required for each context memory.

By using the inter-PE network for transfering configuration data, application cannot be executed at the same time. However, in most cases, the stall is caused when the number of context is not enough to hold configuration data for two continuous tasks. In such cases, the overlapping mechanism

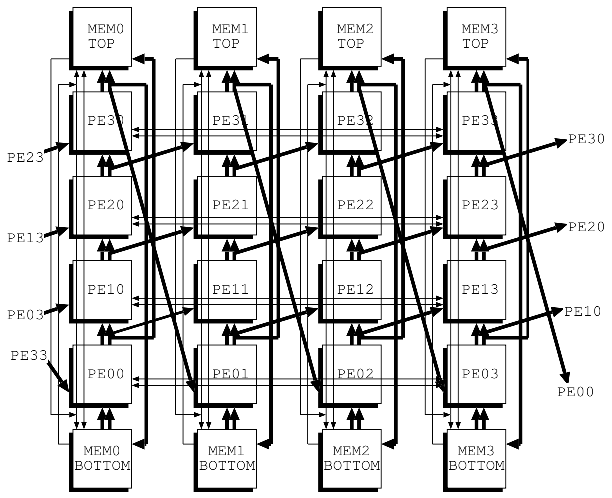

Fig. 1. PE Array Structure of MuCCRA-3.32b

is useless. The CSDP can be implemented so that the mechanism only works in the above cases. That is, the CSDP can be used complementarily with the overlapping mechanism.

3.2. Target Architecture: MuCCRA-3.32b

Although the CSDP can be implemented for various types of DRPs, we must fix the target architecture to evaluate its performance and hardware overhead. Here, MuCCRA-3.32b is adopted as the target architecture. It is based on a low power DRP called MuCCRA-3, the third prototype in MuCCRA project[13] and a real chip using Fujitsu's 65nm CMOS process is working. Only the bit-width of MuCCRA-3 is stretched from 16bits to 32bits in MuCCRA-3.32b.

3.3. PE Array Structure

Although the design of MuCCRA-3.32b is based on previous prototypes MuCCRA-1 and 2 , the structure is optimized so as to reduce the power consumption.

MuCCRA-3.32b has 4×4 PE array as shown in Fig.1. The distributed memory (MEM) modules for storing data are provided at the both edges of the array in order to relax the access conflicts frequently occurred in MuCCRA-1 and 2. Dual-bank method is introduced in order to overlap the time for computation and data transfer from external[17]. For easy debugging, both banks can be directly accessed from outside the memory.

Some programs for MuCCRA-1 or 2 used a certain number of PEs just for generating addresses for accessing MEM modules. To save such PEs, a dedicated address counter is provided to each MEM module. If the address computation is simple, the access of MEM can be done without using PEs. The reset, counting up and down of address counters are defined by the configuration data attached to MEM modules.

The interconnection for connecting PEs is a hybrid style consisting of the island-style network used in MuCCRA-1 and 2, and the direct interconnection network used in MuCCRA-D[18]. In Fig. 1, bold lines represent direct links, while thin lines show the island style network using switching elements (SEs). From the evaluation of MuCCRA-1 and 2, the island style network is advantageous for flexible interconnections, but tends to introduce too long delay. On the other hand, the direct network utilized in MuCCRA-D tends to increase the number of context switching because of its strict limitation in the data transfer. By using the hybrid network in MuCCRA-3.32b, direct links can be used for local communication without using SEs while the data for the distant PEs are transferred through the island style network.

MuCCAR-3 has two channels which can exchange data freely in the SE. Although the structure slightly increases the SE hardware, the flexibility of channel usage would reduce the number of context switching. For keeping the modularity especially convenient in the layout design, each SE is treated as a unit in a PE.

3.4. PE Structure

In the target architecture MuCCRA-3.32b, the data width is stretched to 32bits in order to implement various types of media applications.

The PE structure of MuCCRA-3.32b is consisting of ALU (Arithmetic Logic Unit) and RF (Register File) as shown in Fig. 2. At both input ports of ALU, selection modules (ALU_SELs) are provided to select input data. Simple shift, mask and constant generation functions are also provided in them. The operations in ALU are simplified; addition, subtract, logic operations, shift operation, comparison and data selection. Only 13 operations are provided, while MuCCRA-1 or 2 have 30 instructions. The RF modules is 8-depth 2 reads/1 write register file; the same as MuCCAR-1 or 2.

Another important feature of MuCCRA-3.32b is output registers, which store output of the ALU in each clock cycle. In previous prototypes MuCCAR-1 and 2, the output of PE can be directly forwarded to the other PEs with multiple SEs. Although this FPGA-like data transfer is flexible, the operational frequency often is degraded by the long critical paths through multiple PEs and SEs, and changes drastically depending on the application programs. In MuCCRA-3.32b, only one computation can be done in a clock cycle, although the data can be transferred to distant PEs through multiple SEs. By using this structure, the operational frequency is almost the same independent from the application programs, and pipelined execution is easy to be implemented. This

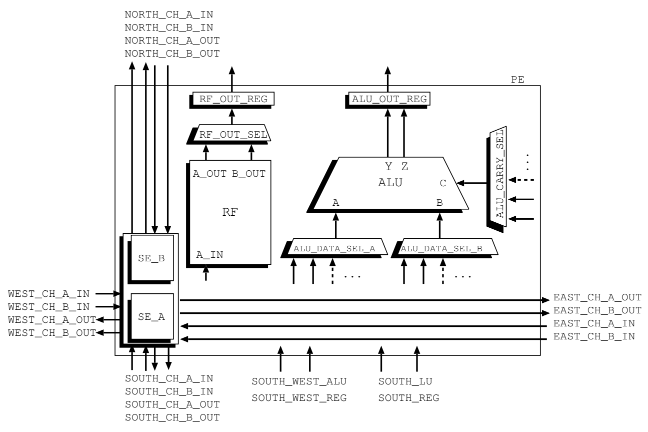

Fig. 2. PE Structure

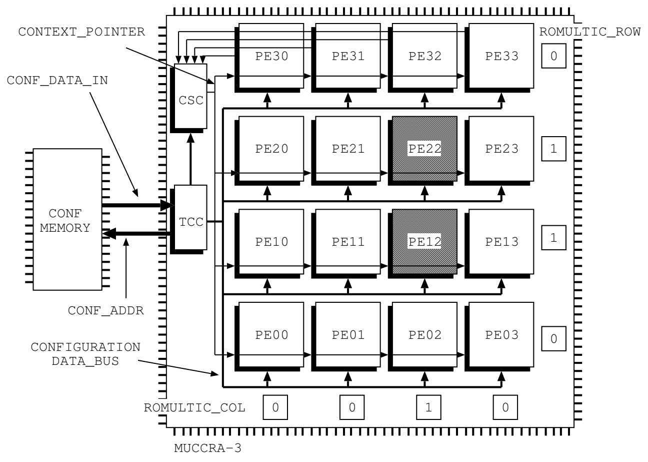

Fig. 3. Configuration Data Flow

structure also suppresses the propagation of the data switching to multiple PEs, thus saves the energy.

3.5. Context Switching and Configuration

Like MuCCRA-1 and 2 and other DRPs, MuCCRA-3.32b is a multi-context style DRP which switches multiple sets of configuration called contexts. All PEs and MEMs provide their context memory modules (CONTEXT_MEMORY), and read out the configuration data according to the context pointer in every clock cycle.

The configuration data in each context memory must be transferred before computation from the memory module outside the chip. Task Configuration Controller (TCC) manages it. Fig.3 shows the configuration and context control mechanism in MuCCRA-3.32b.

First, TCC gives addresses (CONF_ADDR) of required configuration data in the external memory, and reads out the configuration data (CONF_DATA_IN). TCC, PE and MEM modules are connected with a shared bus called CONFIGU-

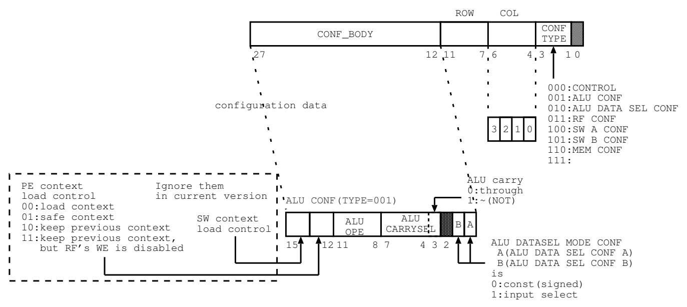

Fig. 4. Configuration Code (ALU Operation)

RATION_DATA_BUS, and TCC transfers the configuration data through the bus with identifiers that indicate receivers.

Each PE or MEM module receives the data if needed, and stores it in its CONTEXT_MEMORY. Like previous prototypes, we adopted multicasting mechanism called RoMultiC[10]. In this mechanism, the configuration data is stored in the CONTEXT_MEMORY if both the multicast bits for row (ROMULTIC_ROW) and column (ROMULTIC_COLUMN) are set to be active high signal. In the example shown in Fig.3, the same configuration data are multicasted to PE12 and PE22. Since the same configuration data often used in multiple modules, this method can reduce the configuration time and total required storage efficiently. The configuration data of MuCCRA-3.32b is designed with the concept of fine-grained RoMultiC[16], and divided into 16bits for each module. Seven format types are used for ALU, ALU_SEL, SE_A, SE_B, RF, MEM and control. Fig. 4 shows a format for ALU. Although this kind of short format increases the clock cycles for configuration data transfer, the opportunities of multicast are increased, and the efficient data transfer is achieved in total. Including the multicast bitmap and configuration types, 28bits data must be transferred.

After transferring all configuration data, Context Switch Controller (CSC) gives CONTEXT_POINTER to all modules for reading the context memory as shown in Fig.5. In MuCCRA-1 and 2, the configuration data is about 100bits, and stored in a single memory. On the contrary, each PE in MuCCRA-3.32b has fragmented CONTEXT_MEMORY modules for ALU, ALU_SEL_A, ALU_SEL_B, SE_A, SE_B and RF. The total configuration bits are reduced into 80bits and multicasted through a single 16bits bus. This approach called fine-grained RoMultiC increases the possibility of multicast and reduce the total energy for configuration data transfer.

Although CONTEXT_POINTER is usually incremented in a clock cycle, the conditional branch and jump according to the computation result in PE30, PE31, PE32 and PE33 are supported.

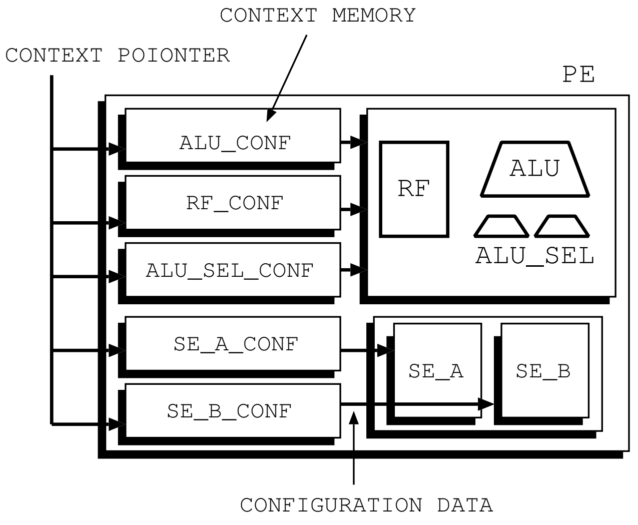

Fig. 5. MuCCRA-3.32b Context Management

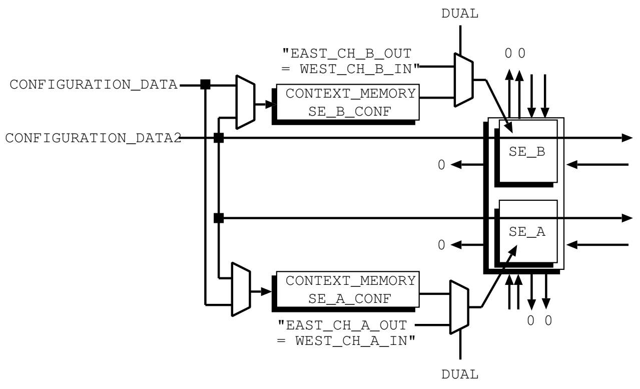

Fig. 6. CSDP Mechanism

3.6. The CSDP in MuCCRA-3.32b

In MuCCRA-3.32b, the configuration data is transferred by using a dedicated CONFIGURATION_DATA_BUS, and the CSDP is implemented for increasing its bandwidth by using network for computation as the second configuration bus. Fig. 6 shows the CSDP implementation in MuCCRA-3.32b. When the computation is stalled and the TCC requires the next configuration, the TCC sends "DUAL" signal to all PEs. When a PE receives "DUAL" signal, the PE changes the configuration data of SE_B into the prepared configuration data for transferring the data from WEST to EAST. By using this configuration, the datapath using SE_B changes into the broadcasting bus CONFIGURATION_DATA_BUS2 for transferring configuration data as shown in Fig. 7. Since the bit-width of links in MuCCRA-3.32b is 32bits, it can carry the whole 28bits data for configuration with a lane.

The outside CONF_MEMORY reads two configuration data (CONF_DATA_IN and CONF_DATA_IN2) in parallel,

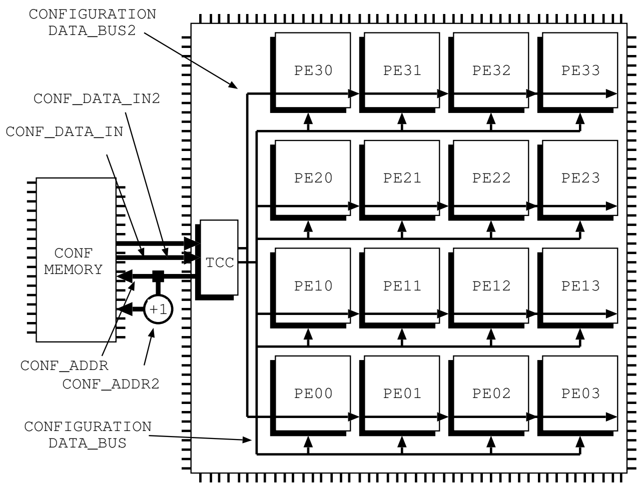

Fig. 7. CSDP in MuCCRA-3.32b

Table 1. Tools Used for Evaluation

Steps	Tools
Simulation	Cadence NC-Verilog 8.1
Synthesis	Synopsys Design Compiler 2007.12-SP3
Power Analysis	Synopsys Prime Time 2007.12-SP3

and transfers by using both configuration buses. Each context memory selects the data from two configuration buses based on the bitmap as shown in Fig. 6. Here, two configuration data must be transferred to different components. The assignment that satisfies this limitation can be found easily in the common configuration codes.

4. EVALUATION

4.1. Hardware Overhead

MuCCRA-3.32b with and without the CSDP is designed by using the same process as MuCCRA-3(Fujitsu 65nm 12-layer CMOS process) with the design tools shown in Table 1. The target frequency in the synthesis was set to be 50MHz and satisfied.

Table 2 shows the hardware overhead represented with the gate numbers.

Table 2. Evaluation Results of Hardware Cost

	Gate Number	Overhead[%]
MuCCRA-3.32b	1,258,963	-
With CSDP	1,275,559	+1.3

The overhead comes from the data selectors, prepared configuration setting mechanism and extra paths for chip I/O and TCC, but it is quite small.

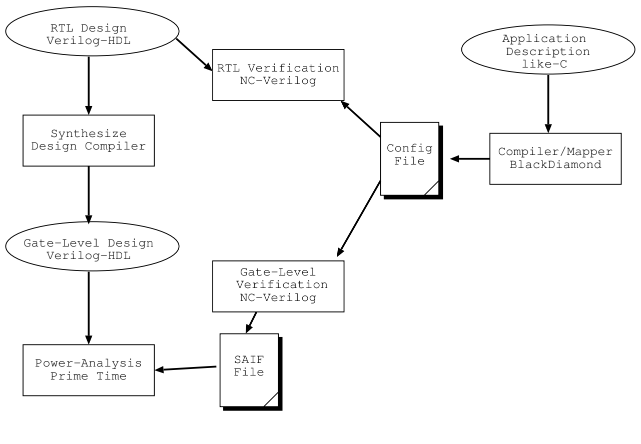
RTL Design Verilog-HDL
RTL Verification NC-Verilog
Application Description like-C
Synthesize Design Compiler
Config File
Compiler/Mapper BlackDiamond
Gate-Level Design Verilog-HDL
Gate-Level Verification NC-Verilog
Power-Analysis Prime Time
SAIF File

Fig. 8. Evaluation Environment

4.2. Execution Time vs. Configuration Data Transfer Time

Discrete Cosine Transform for JPEG coder (2D-DCT) and simple filters for image processing (α-Blender and Sepia) are executed, and configuration data loading time and execution time are evaluated respectively.

The flow of evaluation is shown in Fig.8. Verilog design for MuCCRA-3.32b was modified to provide the CSDP is synthesized by Synopsys's Design Compiler, and gate level design is generated. The design of applications was done by using a re-targetable compiler called BlackDiamond[19]. The application data is described in C-like front-end language, and the mapping, routing and configuration data generation are automatically done. The result of mapping is shown in the GUI (Fig.9), and optimization by designers can be applied.

The generated configuration data is used with the synthesized Verilog netlist, and SAIF file is obtained. The configuration data loading time, and power consumption are evaluated using the gate level simulation and Synopsys's Prime Time.

Fig.10 shows the configuration data loading time. The configuration time is reduced to be almost a half as those without the CSDP in DCT and α-Blender. In Sepia, since a part of configuration data cannot be transferred in parallel, the loading time becomes more than the half.

4.3. Power/Energy Consumption

Power consumption for transferring the configuration data is shown in Fig.11. Since the network for execution is used for configuration data transfer, the power with the CSDP is increased. However, as shown in Fig.12, from the viewpoint

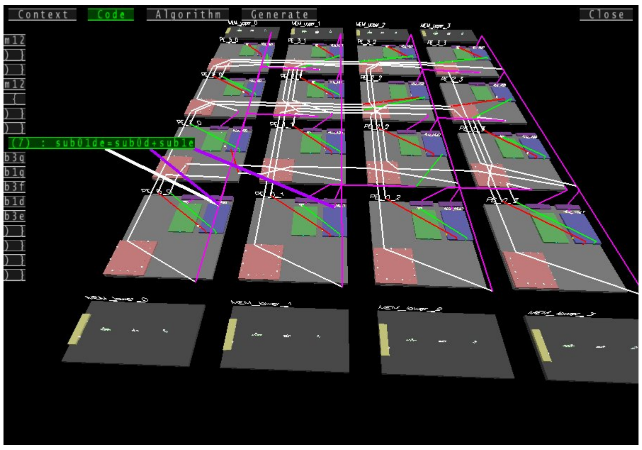

Fig. 9. GUI of BlackDiamond

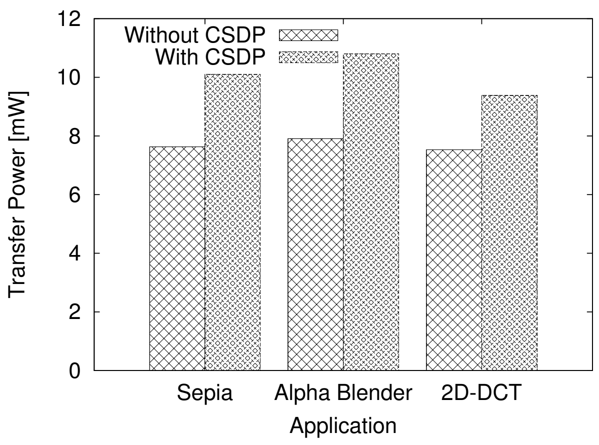

Fig. 11. Power for Configuration Data Loading

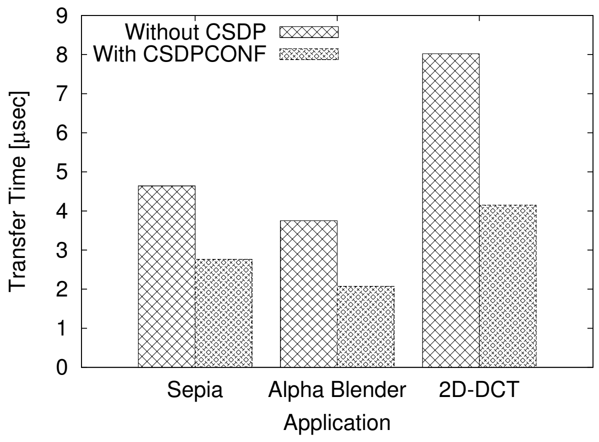

Fig. 10. Configuration Data Loading Time

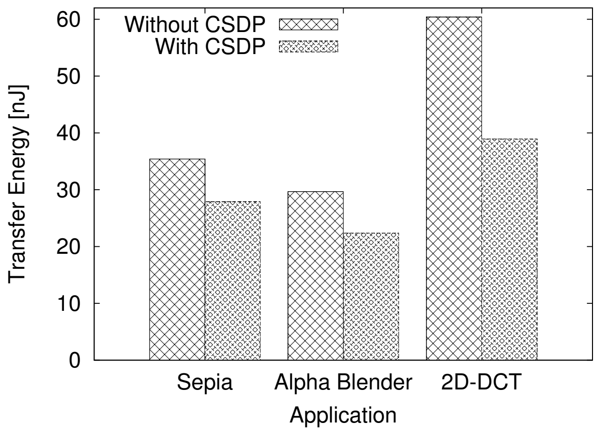

Fig. 12. Energy for Configuration Data Loading

of the energy consumption, the CSDP is better than the original configuration method since the time for configuration is much improved. This evaluation results demonstrate that the CSDP can improve the energy consumption as well as the time for configuration data loading.

The effect of introducing the CSDP is not depending on the process technology except the increasing leakage power which cannot be saved in the technique.

5. CONCLUSION

In CSDP, a network for computation in DRPs is self-configured so that it can be used as a configuration data path when the computation is stalled and the controller requires the configuration data transfer.

Design and implementation of a DRP called MuCCRA-3.32b with CSDP demonstrates that the time for computation can be reduced to a half with only 1.3% hardware overhead. Energy consumption for configuration data loading can be also reduced. The CSDP is a simple technique and so useful with various other methods for high speed configuration data loading including the overlapping.

If there are more number of pins and the configuration memory bandwidth allows, another lane consisting of SE_A can be used as the third configuration bus by using the CSDP. This style of implementation is only realistic when the configuration memory can be embedded inside the chip.

Acknowledgments: This research was performed by the author for STARC as part of the Japanese Ministry of Economy, Trade and Industry sponsored “Next-Generation Circuit Architecture Technical Development” program. The authors thank to VLSI Design and Education Center(VDEC) and Japan Science and Technology Agency(JST) CREST for their support.

6. REFERENCES

[1] F. Veredas and M. Scheppler and 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] Y.Sugawara, K.Ide, T.Sato, "Dynamically Reconfigurable Processor Implemented with IPFlex's DAPDNA Technology," in IEICE Trans. on Inf.&Syst. Vol.E87-D, No.8. Springer, Berlin, Nov. 2003, pp. 1997–2003.

[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] B.Levine, "Kilocore: Scalable, High Performance and Power Efficient Coarse Grained Reconfigurable Fabrics," in Proc. of International Symposium on Advanced Reconfigurable Systems, 2005, pp. 129–158.

[6] Tony Stansfield, "Using Multiplexers for Control and Data in D-Fabrix," in *Proc. of Int'l Conf. on Field Programmable Logic and Application (FPL)*, Sept. 2003, pp. 416–425.

[7] Y.Kurose, I.Kumata, M.Okabe, H.Hanaki, K.Seno, K.Hasegawa, H.Ozawa, S.Horiike, T.Wada, S.Arima, K.Taniguchi, K.Ono, H.Hokazono, T.Hiroi, T.Hirano, S.Takashima, "A 90nm embedded DRAM single chip LSI with a 3D graphics, H.264 codec engine, and a reconfigurable processor ," in Hot Chips 16, 2004.

[8] Masato Motomura , "C-based Programmable-HW Core "STP Engine": Current Status and Future ," in IECE Technical Report, RECONF2008-48, 2008.

[9] X.-P. Ling, H. Amano, "WASMII: A Data Driven Computer on a Virtual Hardware," in Proc. of the IEEE Symposium on FPGAs for Custom Computing Machines(FCCM'93). Springer, Berlin, 1993, pp. 33–42.

[10] V.Tunbunheng, M.Suzuki, H.Amano, "RoMultiC: Fast and Simple Configuration Data Multicasting Scheme for Coarse Grain Reconfigurable Devices," in Proc. of IEEE FPT. Springer, Berlin, 2005, pp. 129–136.

[11] H.Amano, T.Inuo, H.Kami, T.Fujii, M.Suzuki, "Techniques for Virtual Hardware on a Dynamically Reconfigurable Processor - An approach to Tough Cases," in Proc. of the FPL 2004. Springer, Berlin, 2004, pp. 464–473.

[12] S.Hauck and 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.

[13] H.Amano, Y.Hasegawa, S.Tsutsumi, T.Nakamura, T.Nishimura, V.Tanbunheng, A.Parimala, T.Sano, M.Kato, "MuCCRA chips: Configurable dynamically-reconfigurable processors," in Proc. of the ASSCC '07. IEEE Asian, Nov. 2007.

[14] 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.

[15] S. V.Tunbunheng, Y.Hasegawa, A.Parimala, T.Nakamura, T.Nishimura, and H.Amano, "Overwrite Configuration Technique in Multicast Configuration Scheme for Dynamically Reconfigurable Processor Arrays," in *Proc. of FPT*, Dec. 2007, pp. 273–276.

[16] T. Nakamura and T. Sano and Y. Hasegawa and S. Tsutsusmi and V. Tunbunheng and H. Amano, " Exploring the optimal size for multicasting configuration data of Dynamically Reconfigurable Processors ," in *Proc. of Int'l Conf. on International Conference of Field Programmable Technology (ICFPT)*, Dec. 2008, pp. 137–144.

[17] Hideharu Amano and Shohei Abe and Katsuaki Deguchi and Yohei Hasegawa, "An I/O mechanism on a Dynamically Reconfigurable Processor - Which should be moved: Data or Configuration," in *Proceedings of International Conference on Field Programmable Logic and Applications (FPL2005)*, 2005, pp. 347–352.

[18] M.Kato, Y.Hasegawa, H.Amano, "Evaluation of MuCCRA-D: A Dynamically Reconfigurable Processor with Directly Interconnected PEs," in Proc. of The 2008 International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA'08), Aug. 2008.

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