Overwrite Configuration Technique in Multicast Configuration Scheme for Dynamically Reconfigurable Processor Arrays

Satoshi Tsutsumi, Vasutan Tunbunheng, Yohei Hasegawa, Adepu Parimala,
Takuro Nakamura, Takashi Nishimura, and Hideharu Amano
Department of Information and Computer Science, Keio University
Yokohama, 223-8522, Japan
muccra@am.ics.keio.ac.jp

Abstract

A new configuration scheduling algorithm in multicast configuration scheme is proposed and evaluated over reduction ratio of configuration data transfer cycles and power/energy overhead on a coarse-grained dynamically reconfigurable processor array (DRPA). As a case study, the proposed methods are applied to some real applications on a DRPA architecture MuCCRA-1. As a result, we confirmed that the proposed overwrite configuration technique for DRPAs reduced an application configuration cycles 66.5% at maximum compared to one without multicast and 20.2% compared to one without the overwrite configuration. It also decreased the configuration energy consumption 18.7% at maximum beyond the overhead of memory overwrites.

1. Introduction

Coarse-grained dynamically reconfigurable processor arrays (DRPAs) have received an attention to cope with the demands for media-rich applications on a System-on-a-chip (SoC). With its efficiency and flexibility, the DRPA as an off-loading engine will offer a low-power area-efficient design solution. Some devices are commercially available [1, 2, 3, 4, 5, 6, 7].

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[8] for DRPAs has been proposed. For FPGAs, the configuration compression scheme using Wildcard Registers[9] has been researched.

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[10] and WPPA[11] so far.

In these multicast configuration schemes, the configuration data can be overwritten, and the latest configuration data are valid. With this nature, these schemes can cut back further configuration data transfer cycles by scheduling configuration mask patterns and their order.

In this paper, we propose a new configuration scheduling algorithm using the overwrite configuration technique in the multicast configuration scheme. We evaluate the overwrite configuration technique over the reduction effect of configuration data transfer time and power/energy overhead on a coarse-grained DRPA with RoMultiC.

2. Multicast configuration scheme

RoMultiC is similar to the Wildcard Registers but different in that it requires no Wildcard Register writes. RoMultiC uses row and column multicast bits directly to specify configured REs. Configuration data is received by the REs where the row and column multicast bits are both ‘1’. By multicasting configuration data, configuration data transfer time can be substantially shortened. The configurable area is restricted in a rectangle, but by devising the transfer order, any complex configuration pattern can be configured.

3. Multicast configuration scheduling

3.1. Overwrite configuration technique

In order to explain the concept of multicast configuration scheduling, we assume $m \times n$ array with k types of configuration data. As we introduced before, a reconfigurable element such as PE is configured where both the row multicast bit and column multicast bit are ‘1’. We call a set of configured elements defined by multicast bits a configuration mask pattern.

The multicast configuration scheduling corresponds to the process, selecting appropriate configuration mask patterns and arranging them in the order of configuration.

By making the best use of overwriting configuration data, a set of configuration mask patterns can be replaced with another configuration pattern. This means that the number of configuration data transfer cycles is eliminated by the difference of the configuration mask patterns. The overwrites occur where the elements are hidden by the upper configuration mask pattern.

3.2. Pattern separation scheduling

First, we introduce the pattern separation scheduling (PSS) algorithm, which does not use the overwrite configuration technique.

Let F be a collection of all possible configuration mask patterns. Given a target configuration pattern X, we have a collection of configuration mask patterns C which cover X with the following greedy set cover algorithm[12].

```
$T \leftarrow X, C \leftarrow \emptyset$
while $T \neq \emptyset$ do
    Find a set $S \in F \setminus C$ that maximizes $|S \cap T|$
    $C \leftarrow C \cup \{S\}$
    $T \leftarrow T \setminus S$
end
return $C$.
```

3.3. Pattern composition scheduling

Next, we introduce the pattern composition scheduling (PCS) algorithm. At the beginning of process, the PCS finds basic configuration mask patterns for each configuration data type. Then, using the overwrite configuration technique, it decreases configuration data transfer cycles.

We consider two types of the PCS algorithm, the PCS-L and the PCS-S, each of which has different policy on selecting basic configuration mask patterns. The PCS-S is nearly identical to the algorithm in [9].

3.3.1. Basic configuration mask pattern selection

One of the selection of basic configuration mask patterns represented with suffix L is obtained by the PSS introduced before. The other is a set of configuration mask patterns containing only one configuration data of element represented with suffix S.

3.3.2. Configuration mask pattern composition

In order to explain composition scheduling algorithm, we represent a configuration mask pattern as a matrix. A possible value of element is 1, 2, . . ., k, and ϕ (transparent).

For $m \times n$ matrix A and B, we define a relation $A \hat{\subseteq} B$ such that every element of A is covered by that of B where any value except ϕ of element in B can cover the corresponding position of the element. And also, we define a relation $C = A \hat{\cup} B$ by the following equation.

$\forall i \in \{1, 2, \ldots, m\}$ and $j \in \{1, 2, \ldots, n\}$,

$$c_{ij} = \begin{cases} a_{ij} & \text{if } b_{ij} = \phi, \\ b_{ij} & \text{otherwise.} \end{cases}$$

Given a configuration pattern X with k configuration data types and a collection of basic configuration mask patterns C_{basic}, we obtain a collection of configuration mask patterns P with the following pattern composition scheduling algorithm.

```
$T \leftarrow X, C \leftarrow C_{basic}, P \leftarrow \emptyset$
$L_i \leftarrow T_i$ where $T = \bigcup T_i, \forall i \in \{1, 2, \ldots, k\}$
while $C \neq \emptyset$ do
    Find a set $S \in F$ that maximizes $|H|$
        where $\forall i \in \{1, 2, \ldots, k\}$, $(T \hat{\cup} L_i) \hat{\subseteq} S$,
        and $H \subseteq C$ is a collection such that
        $\forall E \in H, E \subseteq S$
    $P \leftarrow P \cup \{S\}$
    $C \leftarrow C \setminus H$
    $T \leftarrow T \setminus E, \forall E \in H$
end
return $P$.
```

4. Case study

As a case study, we use MuCCRA-1[10] for evaluating the overwrite configuration technique. MuCCRA-1 is a coarse-grained dynamically reconfigurable processor incorporating a homogeneous PE array including multipliers and novel configuration schemes with 64 context memory depth.

As shown in Fig.1, the array is composed of a 4×4 24-bit PEs, and four 24-bit multipliers (MULTs) at the left edge and four 24-bit distributed memory modules (MEMs) at bottom edge of the PE array. The MULT multiplies two 24-bit data words and outputs the lower 24 bits of the product with one clock delay. MEMs are 24-bit $\times$ 256-word and 2-port SRAM modules. Like common FPGAs, an island-style interconnection network is employed in MuCCRA-1. Data are routed to destinations via SEs at the channel intersections.

The design was described with Verilog-HDL, synthesized with Synopsys's Design Compiler 2006.06-SP2, and layouted with Cadence's SoC Encouter 5.2.

5. Evaluation

We evaluate the three scheduling algorithms, PSS, PCS-L, and PCS-S, on configuration data transfer cycles.

First, we use 4×4, 6×6, and 8×8 arrays with randomly generated configuration patterns to measure configuration data transfer cycles. For each array size and the number of configuration data types, 4×4 and 6×6 with 1,000 samples respectively and 8×8 with 100 samples are examined.

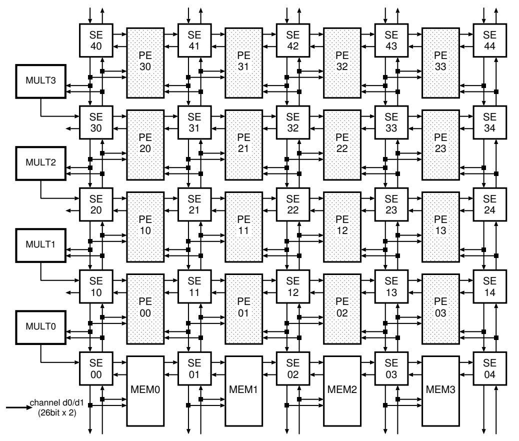

Figure 1. PE array.

Then, we have implemented some practical applications on MuCCRA-1 listed in Table 1 with their characteristics and evaluated on reduction effect of configuration data transfer cycles and power/energy overhead.

We have developed a scheduling program in C++ and run on a PC (Athlon 64 FX-57 (2.8GHz), 2GB memory, and Linux kernel-2.6.8 (64bit)). The power consumption for configuration in MuCCRA-1 is evaluated with Power Compiler 2006.06-SP2, back-annotating a SAIF file obtained by the post-layout simulation.

5.1. Reduction of configuration data transfer cycles

5.1.1. Evaluation with random patterns

The average reduction effect of configuration data transfer cycles for each array size is shown in Fig.2. We can find the most effective scheduling algorithm is the PCS-L. It reduces the configuration data transfer cycles to about 60% of the case without RoMultiC. The impact of the overwrite configuration technique can be seen in the difference between the results of the PSS and PCS-L. With the overwrite configuration, the configuration data transfer cycles are decreased 20% at maximum compared to one without overwrites. Note that although the reduction effect of the PCS-S is slightly inferior to the PCS-L, from the experiment, we confirmed that the PCS-S became superior to the PCS-L for some configuration data arrangements. In practical use, we can employ both the algorithms to get a better result.

Table 1. Application characteristics.

	BlkSize (bits)	Cntxt	Delay (ns)	ExecClks
DCT	1,024	41	40	195
α-Blender	8,192	8	24	644
SHA-1	512	12	50	418
Viterbi	16	13	34	370

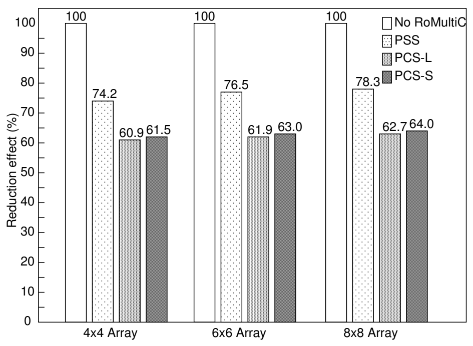

Figure 2. Reduction effect of configuration data transfer cycles in random patterns.

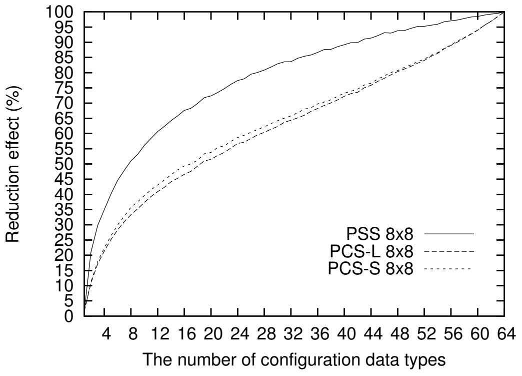

Figure 3. Reduction effect of configuration data transfer cycles (8×8 array).

Fig.3 shows the relations between average reduction effect of configuration data transfer cycles and various configuration data types normalized with the sequential configuration (100%). The results of the other array sizes show almost the same tendency. This relation implies how the reduction effect varies with the degree of parallelism.

5.1.2. Evaluation with real applications

Fig.4 depicts the impact of each configuration scheduling on the configuration data transfer cycles with real applications. It is greatly influenced by the parallelism of the application and the resource utilization.

5.2. Energy consumption overhead

Energy consumption overhead is evaluated with the no overwrite configuration scheduling PSS and the overwrite configuration scheduling PCS-L. The result of the PCS-S is almost the same as the PCS-L.

Table 2 shows the average power consumption during the configuration data transfer. The PCS-L requires a slightly larger power due to the memory overwrites.

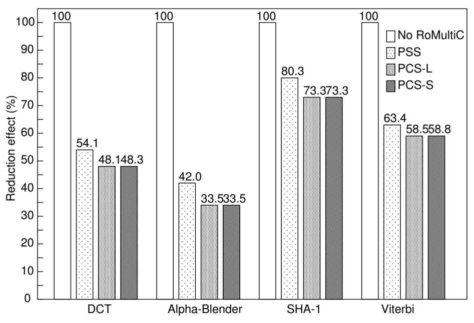

Figure 4. Reduction effect of configuration data transfer cycles in real applications.

Table 2. Power consumption overhead.

	PSS (mW)	PCS-L (mW)	OH. (%)
DCT	28.731	28.916	0.647
α-Blender	48.334	49.260	1.92
SHA-1	23.408	23.573	0.705
Viterbi	33.614	33.717	0.306

As shown in Table 3, the total energy consumption, meanwhile, is decreased. This is because the portion of energy consumption for memory overwrites is not dominant whereas the left portion of it irrelevant with the transfer order is rather large. As a result of the cycle reduction, the reduced energy consumption of them can completely hide the overhead of memory overwrites and rather reduce the total energy consumption.

6. Conclusion

In this paper, we presented the new configuration scheduling algorithm using the overwrite configuration technique in the multicast configuration scheme.

As the results of evaluation with some real applications, we confirmed that our proposed overwrite configuration technique reduced the configuration data transfer cycles 66.5% at maximum compared to one without multicast and 20.2% compared to one without overwrites. It also decreased the configuration energy consumption 18.7% at maximum hiding the overhead of memory overwrites.

Table 3. Energy consumption overhead.

	PSS		PCS-L		
	Config. (clks)	Energy (nJ)	Config. (clks)	Energy (nJ)	OH. (%)
DCT	554	636.68	493	570.24	-10.4
α-Blender	84	97.441	67	79.210	-18.7
SHA-1	241	282.07	220	259.30	-8.07
Viterbi	206	235.43	190	217.81	-7.49

Acknowledgment: This work is supported in part by Japan Science and Technology Agency (JST). The authors thank to VLSI Design and Education Center (VDEC) for the design flow of the Rohm 0.18μm CMOS technology.

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 Int'l Conf. on Field Programmable Logic and Application (FPL)*, pages 106–111, August 2005.

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

[3] T. Sugawara, K. Ide, and T. Sato. Dynamically Reconfigurable Processor Implemented with IPFlex's DAPDNA Technology. *IEICE Trans. on Information & System*, E87-D(8):1997–2003, May 2004.

[4] M. Petrov et al. The XPP Architecture and Its Co-simulation within the Simulink Environment. In *Proc. of Int'l Conf. on Field Programmable Logic and Application (FPL)*, pages 761–770, August 2004.

[5] Inc. Rapport. http://www.rapportincorporated.com/.

[6] T. Kodama et al. Flexible Engine: A Dynamic Reconfigurable Accelerator with High Performance and Low Power Consumption. In *Proc. Int'l Symposium on Low-Power and High-Speed Chips (COOL Chips)*, pages 393–408, April 2006.

[7] T. Stansfield. Using Multiplexers for Control and Data in D-Fabrix. In *Proc. of Int'l Conf. on Field Programmable Logic and Application (FPL)*, pages 416–425, September 2003.

[8] V. Tunbunheng, M. Suzuki, and H. Amano. RoMultiC: Fast and Simple Configuration Data Multicasting Scheme for Coarse Grain Reconfigurable Devices. In *Proc. of IEEE Int'l Conf. on Field Programmable Technology (FPT)*, pages 129–136, 2005.

[9] S. Hauck, Li Zhiyuan, and E. Schwabe. Configuration compression for the Xilinx XC6200 FPGA. In *Proc. of IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 138–146, April 1998.

[10] Y. Hasegawa and H. Amano. Design methodology and trade-offs analysis for parameterized dynamically reconfigurable processor arrays. In *Proc. of Int'l Conf. on Field Programmable Logic and Application (FPL)*, August 2007.

[11] D. Kissler, F. Hannig, A. Kupriyanov, and J. Teich. A Highly Parameterizable Parallel Processor Array Architecture. In *Proc. of IEEE Int'l Conf. on Field Programmable Technology (FPT)*, pages 105–112, December 2006.

[12] V. Chvátal. A greedy heuristic for the set-covering problem. In *Math. of OR*, volume 4, pages 233–235, 1979.