A Configuration Data Multicasting Method for Coarse-Grained Reconfigurable Architectures

Takuya Kojima and Hideharu Amano
Dept. of Information and Computer Science, Keio University, Yokohama, Japan

***Abstract*—This paper proposes a novel configuration data compression technique for coarse-grained reconfigurable architectures (CGRAs). The proposed technique is based on a multicast configuration technique called RoMultiC, which reduces the configuration time by multicasting the same data to multiple PEs(Processing Elements) with two bit-maps. Scheduling algorithms for an optimizing the order of multicasting have been proposed. In general, configuration data for CGRAs can be divided into some fields like machine code formats. The proposed scheme confines a part of fields for multicasting so that the possibility of multicasting more PEs can be increased. This paper analyzes algorithms to find a configuration pattern which maximizes the number of multicasted PEs. We implemented the proposed scheme to CMA (Cool Mega Array), a straight forward CGRA as a case study. Experimental results show that the proposed method achieves 40.0% smaller configuration for an image processing application at maximum. Furthermore, it achieves 35.6% reduction of the power consumption for the configuration with a negligible area overhead.**

I. Introduction

Coarse-grained reconfigurable architecture (CGRA) is a remarkable platform for embedded systems including IoT devices because of its high degree of energy efficiency and programmability. CGRAs have an array of reconfigurable processing elements (PEs) for efficient parallel execution of compute-intensive application. Each PE operates according to configuration data which specify behaviors of components in the PE, and the size of configuration data for the whole PE array is often quite large.

Considering that a CGRA is connected to a host CPU with a general purpose interface bus like 32-bit width as an accelerator, a critical issue is a long time to transfer the configuration data from the CPU to the CGRA, in other words, reconfiguration time. That is why an efficient method of reducing the configuration data is required.

Multicasting is one of techniques for reducing the reconfiguration time with simple hardware. It enables the multiple PEs to be configured simultaneously by exploiting a fact that a certain number of PEs use the same configuration data. The multicasting is effective, especially when the same data-flow-graph mapping is repeated on the PE array to utilize data-level parallelism of an application. RoMultiC is a multicasting method using two bit-maps for the rows and the columns of the PE array[1][2]. Other multicasting methods are also proposed in [6] and [7]. If the configuration data are allowed to be overwritten, the latest configuration for each PE is actually used for its operation. A scheduling of multicasting proposed in [1] can reduce the time for multicasting.

In general, the configuration data can be divided into some fields such as an opcode and operands for ALU. In RoMultiC[1], the PEs which have an identical configuration for all fields are allowed to be multicasted. However, the grain size of multicasted fields has not been well investigated. When the partial fields are allowed to be written, the number of PEs to be multicasted is increased. Here, we propose a fine grain multicasting method considering a practical configuration data format. In this work, we compare two scheduling algorithms: 1) *Espresso*[3]-base algorithm and 2) Integer-Linear-Program (ILP)-base algorithm for the fine grain multicasting. Implementation overhead is also important as well as reducing the configuration time. Thus, the proposed schemes are implemented considering a real chip CGRA, and compared with the previous method from the viewpoint of reconfiguration speed and hardware overhead.

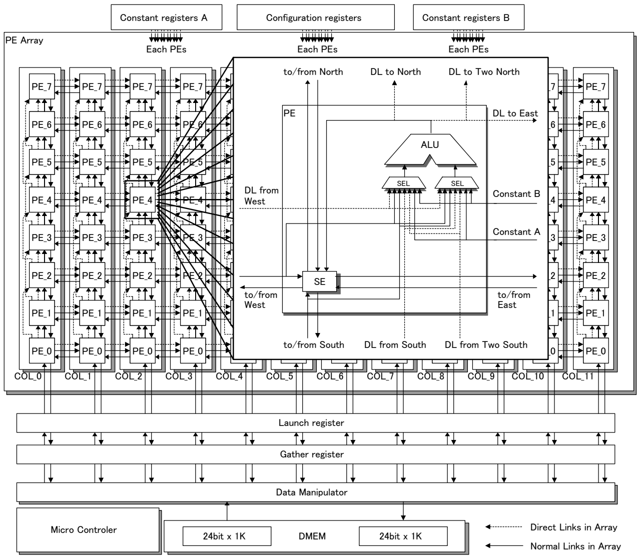


Fig. 1: Overview of the PE array

II. Background

A. Base Architecture

CMA (Cool Mega Array) is a low power straight forward style CGRA[4]. In this work, we use CC-SOTB (CMA-Cube-SOTB)[5], which is an improved version of CMA, as a base architecture for implementation of the proposed method.

Fig. 1 illustrates an array of 12×8 PEs in the CC-SOTB. Each PE consists of an ALU and a switch element (SE) as shown in Fig. 1. PEs are connected to each other with two types of interconnections: 1) direct links and 2) links provided by SEs. The main feature of the CC-SOTB is its static reconfiguration so as to cut down a large power consumption for a dynamic reconfiguration which changes the configuration clock-by-clock.

The configuration controller decodes input configuration data and writes them into the configuration registers. CC-

978-0-7695-6525-5/18/$31.00 ©2018 IEEE
DOI 10.1109/FPL2018.2018.00048

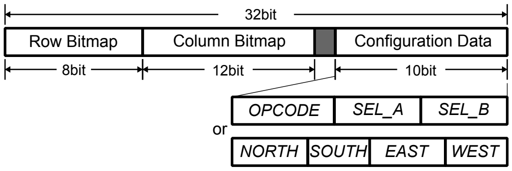


Fig. 2: Multicasting of Configuration data

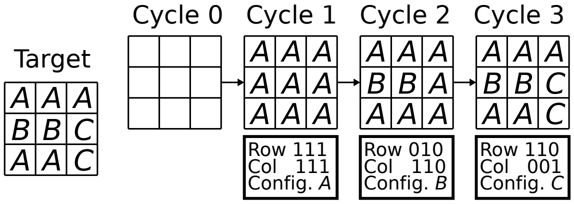


Fig. 3: Efficiency of overwriting configuration

SOTB has an external bus and a chip interface for connections to a host CPU or other accelerators. The external bus consists of a 22bit address bus and a 32bit data bus. Each module is mapped to the same address space. Thereby, the host CPU can access data in CC-SOTB modules via the interface and the external bus. The configuration data for a PE are composed of 10-bit ALU part (*OPCODE, SEL_A, and SEL_B*) and 10-bit SE part (*NORTH, SOUTH, EAST, and WEST*). Thus, the total amount of the configuration data is 20-bit × 96 = 1920-bit. A configuration of each PE is also mapped to the address space. Therefore, it takes 96 cycles to complete the reconfiguration without multicasting.

B. Multicast Configuration

In order to reduce the reconfiguration time, RoMultiC scheme [1], which is one of multicasting method, is applied to the CC-SOTB. In RoMultiC, the transferred data use two bit-maps which respectively indicate multicasted rows and columns in the PE array. As for a PE at the coordinate (x, y), the configuration data is multicasted when both x-th bit of the column bit-map and y-th bit of the row bit-map are set to "1". The data format for multicasting is shown in Fig. 2. Unlike an original format used in [1], the configuration data are written separately divided into the two parts, ALU part and SE part because there is not enough data space in the format. The bit-maps need 8 + 12 = 20 bits and then 12 bits remain for the configuration data of PEs. The part shaded gray (2 bit) in Fig. 2 is unused.

In order to show how overwriting reduces the configuration data, we use a simple example illustrated in Fig. 3. The example illustrates that the multicasting with the overwrite takes 3 cycles to complete the target configuration. First, configuration "*A*" is multicasted to the whole PEs. Next, configuration "*B*" is multicasted to two PEs at the coordinates (1, 2) and (2, 2). Lastly, configuration "*C*" is multicasted to two PEs at the coordinates (3, 1) and (3, 2). Without the overwrite, it takes 4 cycles since the configuration "*A*"s are completed with multicasting twice.

III. Motivation

A. Fine Grain Multicasting

As explained in Section II, the configuration data are multicasted ALU-by-ALU or SE-by-SE. It is not clear that the grain of multicasting is really effective in reducing the configuration data. Here, we consider to use smaller units and propose the fine grain multicasting.

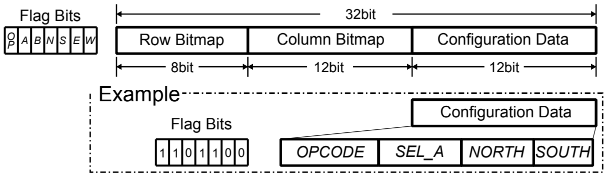


Fig. 4: Fine grain multicasting

In order to implement the scheme to CC-SOTB, a new data format for the multicasting is described in Fig. 4. In the fine grain multicasting, any combination of the fields can be multicasted unless the required data size exceeds in the available data size. Optimized combination of the fields can increase the possibility of either multicasting more PEs or utilizing the available data space in the format. Fig. 4 shows an example of the best case that the data space is fully utilized. In this case, the configuration data include 4 fields which require 12 bits totally.

Although additional flag bits are necessary to decide which fields the configuration data contain, in our case, it can be included in the address space. In other words, only 128-byte address space has to be reserved for the fine grain multicasting. Assuming n fields, 2^n-byte is needed. Of course, the flag bits are included in the data format. Nevertheless, it increases the overhead for multicasting in addition to the bit-maps.

B. Using Espresso

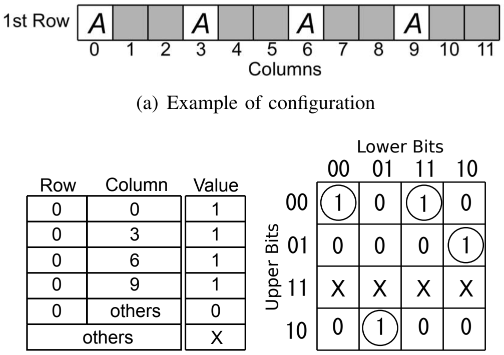
As a way to find the suitable bit-maps, a natural way is using CAD algorithms. Here, *Espresso*, a classic logic minimization algorithm, is employed like [8]. It can treat "Don't Care" denoted by **X** as well as logic **0** and **1**. A truth table whose entry corresponds to each PE in the PE array is used in *Espresso*. **1** indicates that the PE has the same configuration data, while **0** means that the PE has a different configuration data and is already written. **X** is used when a PE has a different configuration data but have not written yet.

Although *Espresso* itself is a highly efficient heuristic algorithm, sometimes it does not work well to find the bit-maps. For instance, when a target configuration shown in Fig. 5(a) is given, a truth table in Fig. 5(b) for multicasting configuration "*A*" is generated. The parts shaded gray indicate PEs whose configuration is already written. Then, a Karnaugh map associated with the row bit-map is created as described in Fig. 5(c). However, a cube which covers multiple cells cannot be found. A cube corresponds to a group of multicasted PEs. In other words, the four PEs which have the configuration "A" cannot be multicasted simultaneously.

It should be discussed how the disadvantage of *Espresso* for the fine grain multicasting has influence on the reduction of configuration data. Here, we propose another method using Integer-Linear-Program in the next section.

IV. New Multicasting Method

As mentioned in Section II-B, scheduling an order of the overwrite is important to reduce the configuration data. Here, we employ a greedy algorithm for scheduling similarly to [1][8] because of the simplicity of the algorithm. Given a target configuration, finding the bit-maps which maximize the number of written bits is repeated until the configurations of all fields in the PEs are fixed. Please note that the bits of multicasted fields which are overwritten later are not counted for maximizing.



(a) Example of configuration

Row	Column	Value
0	0	1
0	3	1
0	6	1
0	9	1
0	others	0
others		X

(b) Truth Table

	00	01	11	10
00	1	0	1	0
01	0	0	0	1
11	X	X	X	X
10	0	1	0	0

(c) Karnaugh map

Fig. 5: Worst case with *espresso*

Algorithm 1 Algorithm for obtaining bitmaps with *Espresso*

```
Input: $C_{fix}, C_{unfix}$
Output: $B_{max,r}, B_{max,c}, f_{max}, d_{max}$
$S_{max} \leftarrow 0, B_{r,max} \leftarrow$ None, $B_{c,max} \leftarrow$ None,
    $f_{max} \leftarrow \phi, d_{max} \leftarrow$ None
$F = \{OPCODE, SEL\_A, SEL\_B,$
        $NORTH, SOUTH, EAST, WEST\}$
for all $f \in 2^F$ do
    if bit_width($f$) $\leq$ max_width then
        $\boldsymbol{D} \leftarrow$ enumerate_config_data($C_{unfix}, f$)
        for all $d \in \boldsymbol{D}$ do
            $tt$ = make_truth_table($C_{fix}, C_{unfix}, d, f$)
            $\boldsymbol{B_r}, \boldsymbol{B_c} = espresso(tt)$
            for all $b_r, b_c \in \boldsymbol{B_r}, \boldsymbol{B_c}$ do
                $S \leftarrow$ count_bits($b_r, b_c, f$)
                if $S > S_{max}$ then
                    $S_{max} \leftarrow S$, $B_{r,max} \leftarrow b_r$ , $B_{c,max} \leftarrow b_c$
                    $f_{max} \leftarrow f$ , $d_{max} \leftarrow d$
                end if
            end for
        end for
    end if
end for
```

In this work, two algorithms for finding two bit-maps are presented respectively in the subsection IV-A and the subsection IV-B. Both algorithms return bit-maps for the rows B_r and the columns B_c, the group of fields in the configuration format f and multicasted data d.

A. Espresso for finding the bit-maps

Algorithm 1 shows a solution for finding the bit-maps with *Espresso*. C_{fix} is a set of fixed configurations and they can not be overwritten except when all multicasted fields are the same as the previously written. In this case, the value of the truth table is set to **X** (Don't care).

Like Fig. 5, a truth table tt for each combination of the fields f and for each configuration data d are generated. 2^F denotes the power set of the fields F, that is, possible combinations of the fields. In our case, the number of the available combinations is 94. After *Espresso* finds cubes, bit-maps of the rows and the columns for each cube are generated. Then, the sum of written bits are calculated and the best bit-maps $B_{r,max}$ and $B_{c,max}$ are obtained. This method does not maximize the number of bits multicasted at the same time, but maximizes the number of multicasted rows and columns for given the combination of the fields. This is because *Espresso* can treat only single output logic.

Algorithm 2 Algorithm for obtaining bitmaps with ILP

```
Input: $C_{fix}, C_{unfix}$
Output: $B_{max,r}, B_{max,c}, f_{max}, d_{max}$
$S_{max} \leftarrow 0$
for all $d \in$ enumerate_config_data($C_{unfix}$) do
    $S, B_r, B_c, f \leftarrow$ find_bitmap_by_ILP($d$)
    if $S_{max} > S$ then
        $S_{max} \leftarrow S, B_{max,r} \leftarrow B_r, B_{max,c} \leftarrow B_c$
        $f_{max} \leftarrow f, d_{max} \leftarrow d$
    end if
end for
```

B. ILP for finding the bit-maps

In addition to the *Espresso*-based algorithm, we consider using an integer linear program (ILP) in order to obtain an optimal solution of the bit-maps. An overview of the algorithm is shown in Algorithm 2. In addition, an ILP model used in the function "find_bitmap_by_ILP" is formulated as follows.

$$isField_i = \begin{cases} 1 & \text{if the } i\text{-th field is written} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

$$isRow_j = \begin{cases} 1 & \text{if the } j\text{-th row is written} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

$$isCol_k = \begin{cases} 1 & \text{if the } k\text{-th column is written} \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

$$\max S = \sum_i \sum_j \sum_k S_{ijk} * isField_i * isRow_j * isCol_k \quad (4)$$

subject to

$$\sum_i bit\_width_i * isField_i \quad \leq \quad bit\_width_{max} \quad (5)$$

if i-th field of the PE in the j-th row and the k-th column is already fixed, then

$$isField_i = 0 \lor isRow_j = 0 \lor isCol_k = 0 \quad (6)$$

where S_{ijk} is the number of bits for i-th field of the PE in the j-th row and the k-th column, bit_width_i is the bit width of configuration data in i-th field, and bit_width_{max} is available bit width in a multicasted data. S_{ijk} is non-zero value only if i-th field of the PE in the j-th row and k-th column has the same the configuration data which are considered to be multicasted. Given a multicasted configuration data, each S_{ijk} can be calculated. Therefore, S_{ijk}s are constant values. bit_width_i and bit_width_{max} are also constant. At first glance, the objective function is not linear. However, the product of binary valuables can be replaced with an additional variable and two constraints.

V. Evaluation

A. Reduction of the configuration data

First, we develop tools which can generate data for multicasting with the three methods: 1) previous method, 2) fine grain multicasting with *Espresso* (FGM-E) and 3) fine grain multicasting with ILPs (FGM-I). The possibility of reducing the configuration data for each method is then evaluated.

The drawback of *Espresso* depends on a mapping size of an application. For instance, when many PEs are utilized,

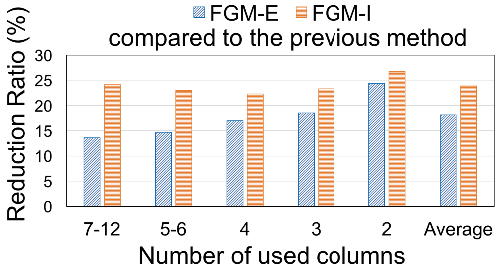


Fig. 6: Reduction ratios for each algorithm

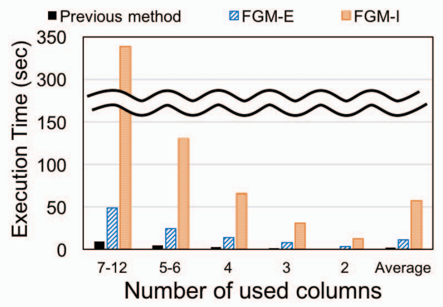


Fig. 7: Execution time for each method

Espresso is likely to fail the multicasting. In order to evaluate the influence, a lot of data-flow-graphs (DFGs) with various node size are generated randomly. Then, the multicasted data for each method are calculated. The average reduction ratio between the previous method[1] and both FGM-E and FGM-I are shown in Fig. 6. Compared to the previous method, the FGM-I achieves 23.8% reduction of configuration data in average. Originally, the previous method can reduce the configuration data by 60% in comparison with single-cast method. As expected, the larger DFG is configured, the smaller reduction ratio of FGM-E can be observed. On the other hand, FGM-I achieves similar reduction ratio for all configurations. Furthermore, three methods are applied to four image processing applications (*gray*, *sepia*, *af*, *sf*), which are used for evaluations in [5]. As the results, the reducation ratios of FGM-E and FGM-I are respectively 11.7% and 19.1% in average for the real applications. In the best case (*sepia* application), FGM-I achieves 40.0% reduction.

B. Time to schedule the configuration data

Fig. 7 depicts the execution time when the algorithms are executed on an Intel Xeon with 12 threads. Although FGM-I indicates better reduction ratio than FGM-E and the previous method regardless of the DFG size, it takes a long time to generate the multicasted data. When the large DFG is configured, FGM-E also takes a long time

C. Overheads

The CC-SOTB architecture with the configuration controller which supports the fine grain multicasting is implemented using SOTB 65 nm process technology with Synopsys Design Compiler to analyze overheads of the proposed method. Area overhead of the proposed method is shown in Table I. It also includes power consumptions of both implementations. The power consumptions are simulated with Synopsys PrimeTime and Cadence NC-Verilog. The operating frequency and the supply voltage are respectively set to 30 MHz and 0.55 V. The proposed method achieves 35.6% lower dynamic power consumption than the previous method, because writing the configuration data finely minimizes unnecessary switching in the configuration registers.

TABLE I: Comparison of area and power consumption

	area (mm^2)	overhead (%)
previous method	0.944	—
FGM	1.04	9.76

	Dynamic power (μW)	Static power (μW)
previous method	514.5	6.125
FGM	329.3	6.247

VI. Conclusion

In this work, we have introduced a new configuration multicasting scheme for CGRAs. By optimizing the grain of multicasting, the proposed method can reduce both the configuration data and the dynamic power consumption. In order to generate multicasted data, two algorithms based on *Espresso* and ILP are considered. As the experimental results, they provide a possibility of a trade-off between the reduction ratio and the execution time. When the proposed method is applied to real applications, about 40% reduction of the configuration data can be achieved in the best case.

Acknowledgment

This work is partially supported by JSPS KAKENHI S Grant Number 25220002 and JSPS KAKENHI B Grant Number 18H03215. This work is supported by VLSI Design and Education Center(VDEC), the University of Tokyo in collaboration with Synopsys, Inc and Cadence Design Systems, Inc.

References

[1] S. Tsutsumi, 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 *Field-Programmable Technology, 2007. ICFPT 2007. International Conference on*. IEEE, 2007, pp. 273–276.

[2] S. M. Jafri, A. Hemani, K. Paul, J. Plosila, and H. Tenhunen, "Compression based efficient and agile configuration mechanism for coarse grained reconfigurable architectures," in *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*. IEEE, 2011, pp. 290–293.

[3] S. Park and K. Choi, "An approach to code compression for CGRA," in *Quality Electronic Design (ASQED), 2011 3rd Asia Symposium on*. IEEE, 2011, pp. 240–245.

[4] B. Liu, W.-Y. Zhu, Y. Liu, and P. Cao, "A configuration compression approach for coarse-grain reconfigurable architecture for radar signal processing," in *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2014 International Conference on*. IEEE, 2014, pp. 448–453.

[5] R. K. Brayton, G. D. Hachtel, C. McMullen, and A. Sangiovanni-Vincentelli, *Logic minimization algorithms for VLSI synthesis*. Springer Science & Business Media, 1984, vol. 2.

[6] N. Ozaki, Y. Yasuda, M. Izawa, Y. Saito, D. Ikebuchi, H. Amano, H. Nakamura, K. Usami, M. Namiki, and M. Kondo, "Cool Mega-Arrays: Ultralow-Power Reconfigurable Accelerator Chips," *IEEE Micro*, vol. 31, no. 6, pp. 6–18, Nov 2011.

[7] Y. Matsushita, H. Okuhara, K. Masuyama, Y. Fujita, R. Kawano, and H. Amano, "Body bias grain size exploration for a coarse grained reconfigurable accelerator," in *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*, Aug 2016, pp. 1–4.

[8] S. Hauck, Z. Li, and E. Schwabe, "Configuration compression for the Xilinx XC6200 FPGA," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, no. 8, pp. 1107–1113, 1999.