DESIGN METHODOLOGY AND TRADE-OFFS ANALYSIS FOR PARAMETERIZED DYNAMICALLY RECONFIGURABLE PROCESSOR ARRAYS

Yohei Hasegawa, Satoshi Tsutsumi, Vasutan Tanbunheng, Takuro Nakamura, Takashi Nishimura, and *Hideharu Amano*

Department of Information & Computer Science, Keio University
3-14-1 Hiyoshi, Kohoku-ku, Yokohama, 223-8522 Japan
E-Mail: muccra@am.ics.keio.ac.jp

ABSTRACT

In this paper, we propose a Dynamically Reconfigurable Processor Array (DRPA) generator which can generate various types of DRPAs. Our target DRPA architecture is fully parameterized. By specifying architectural parameters, it can automatically generate RTL model, simulation environment, and finally chip layout. In our DRPA generator, although the fundamental design of a processing element (PE) and an inter-PE connection is fixed, the array size, PE granularity, and connection flexibilities of intra/inter PE are selectable. In this paper, we have generated various types of DRPAs and evaluated semiconductor area and speed by using the ASPLA/STARC 90-nm CMOS technology. From evaluation results, fundamental trade-offs between architectural parameters and area/delay are analyzed.

1. INTRODUCTION

In recent years, coarse grained dynamically reconfigurable processor arrays (DRPAs) have been received an attention as a flexible and efficient off-loading engine for various types of System-on-Chips (SoCs). Some devices are commercially available [1, 2, 3, 4], and some of them have been integrated into digital appliances [5].

In order to achieve better area- and power- efficiency compared with traditional field-programmable devices such as FPGAs, they incorporate the following properties: (1) a simple coarse grained processor consisting of an ALU, a data manipulator, a register file and other functional modules is used as a primitive processing element (PE) of an array, and (2) dynamic reconfiguration which enables PE array to perform time-multiplexed execution.

Unlike common FPGAs which are based on Look-Up-Tables (LUTs) and island-style interconnection, there exist wide design space in DRPAs, such as PE granularity, the number of hardware contexts which can be switched dynamically, the total amount of wiring resource, and PE array size itself. Our previous work revealed that the optimal PE array size considering area and power consumption is different for each application [6]. Thus, we believe that there is no all-around architectures in DRPAs, and the structure should be configurable or customizable for its main target applications. Since DRPAs are embedded into an SoC, their architectures should be customized at design time.

The object of our project, Multi-Core Configurable Reconfigurable Architecture (MuCCRA) project, is to develop a design methodology and framework which generate highly configurable DRPAs for various target applications. In this paper, as the first step of the project, we develop a flexible architecture generator and target DRPAs are modeled and parameterized. And then, the impact of architectural parameters on area and delay is analyzed.

2. DESIGN ENVIRONMENT

Fig.1 depicts our design environment of parameterized DRPAs. Our final goal is generating both chip layout of a DRPA and its programming environment based on designer's demands. The basic DRPA architecture template is fixed, and designers can generate their desired DRPAs by controlling parameters.

At first, our DRPA generator loads architectural parameters and generates a synthesizable Verilog-HDL model of the DRPA. They can be logically and physically synthesized without any modifications. Since a simple testbench is also generated, it is feasible to verify the DRPA immediately. In addition, a DRPA compiler which generates configuration data from C-like description is also created from a retargetable compiler generator. But, the DRPA compiler is under construction and is out of scope in this paper.

3. TARGET ARCHITECTURE TEMPLATE

3.1. PE Architecture

The basic building unit of DRPA is a Processing Element (PE) shown in Fig.2. Each PE has a programmable PE Core, connection blocks, and a context memory. In the PE Core,

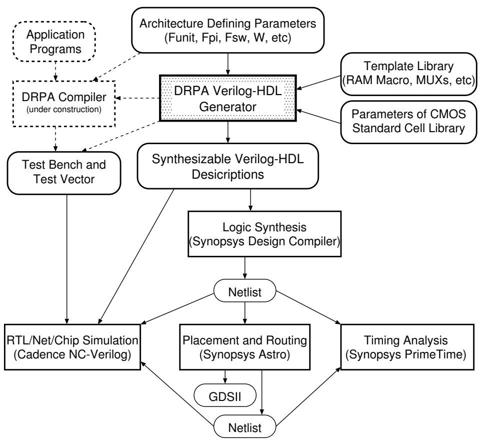
Application Programs
Architecture Defining Parameters (Funit, Fpi, Fsw, W, etc)
Template Library (RAM Macro, MUXs, etc)
DRPA Compiler (under construction)
DRPA Verilog-HDL Generator
Parameters of CMOS Standard Cell Library
Test Bench and Test Vector
Synthesizable Verilog-HDL Desicriptions
Logic Synthesis (Synopsys Design Compiler)
Netlist
RTL/Net/Chip Simulation (Cadence NC-Verilog)
Placement and Routing (Synopsys Astro)
Timing Analysis (Synopsys PrimeTime)
GDSII
Netlist

Fig. 1. Design Environment

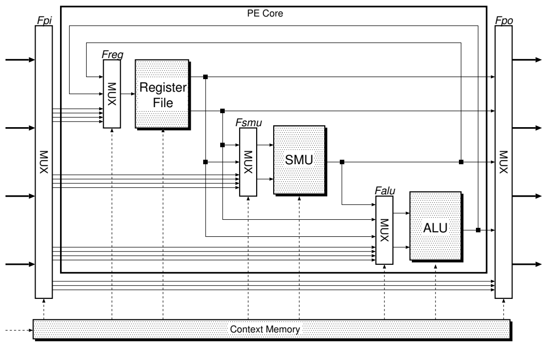

Fig. 2. PE Architecture and Intra-PE Flexibilities

like most existing DRPA devices, a data manipulator called Shift & Mask Unit (SMU), an Arithmetic Logic Unit (ALU), and a register file are provided. In this paper, it is assumed that each ALU supports a multiplication operation.

The most fundamental parameter of DRPA is granularity of PE given by G. G specifies the data width treated in a PE and interconnection. G is set from 4 to 32 in the most cases.

The flexibility of interconnection within a PE Core can be defined with the number of selectors provided on inputs and outputs of functional units such as ALU. Each functional unit of a PE Core has an input selector, and the number of input channels which can be selected by the unit is an important parameter. As shown in Fig.2, the input channel number for SMU, ALU, and register file are represented by $F_{\mathrm{smu}}, F_{\mathrm{alu}}, F_{\mathrm{reg}}$ respectively. These parameters are corresponding to the flexibility of intra-PE local routing.

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 the PE Core. We define the number of inputs and outputs that can be connected to the connection blocks as F_{pi} and F_{po}. If the connection blocks can get the data from global routing wires in 4 directions, the number of connections in

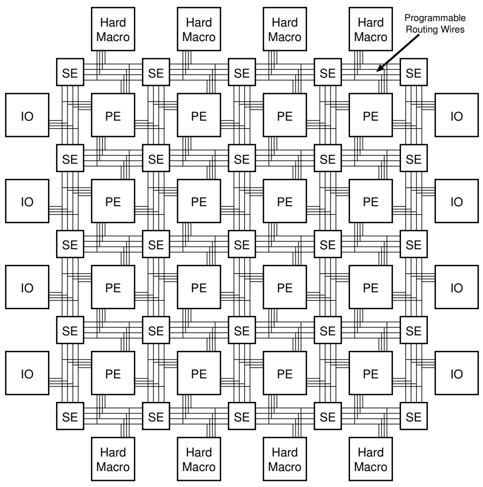

Fig. 3. DRPA Architecture with 4×4 PEs

each direction is $F_{\mathrm{pi}}/4$. In this work, F_{po} is defined by the total number of outputs of functional units, i.e., 4. The operations of each functional unit and local intra-PE connection are statically defined by configuration data called a context.

3.2. Array Architecture

Our DRPAs have a two-dimensional PE array, and its size is denoted by (M, N). And, an island-style interconnection structure like traditional FPGAs is adopted. Fig.3 shows an example of the DRPA with $(4, 4)$. As shown in this figure, each PE is surrounded by programmable routing wire segments. And, connection blocks in each PE mediate the connection between PEs and global routing resources.

On the intersection of a vertical and horizontal channel, 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. The number of channels in global routing resources is denoted by W, and each SE provides W independent switches. For each switch, an entering link can be connected to F_{sw} other links, where F_{sw} means the SE flexibility or the flexibility of inter-PE global routing.

In many released DRPAs, a certain number of distributed memory modules are installed for some group of PEs. For example, NEC Electronics' DRP-1 has 8 distributed memory modules called VMEMs and HMEMs per 8×8 PE array. Here, we similarly assume that distributed memory modules are provided for a certain number of PEs.

3.3. Context Switching Mechanism

Each PE and SE in a DRPA equips its context memory in which the configuration data for a particular operation is held. The central controller broadcasts a context pointer to all of reconfigurable elements including PEs and SEs. The context is read from the context memory according to the context pointer, and all reconfigurable elements are reconfigured in parallel. This type of dynamic reconfiguration is called a multicontext scheme, and a lot of current devices support it. In multicontext devices, the dynamic reconfiguration can be done in only one clock cycle by distributing the context memory into each reconfigurable element. In this paper, the number of contexts is defined with the parameter C.

4. EVALUATION RESULTS AND ANALYSIS

In this section, we show the evaluation results and analyze the fundamental trade-offs of DRPAs. We generated DRPAs with the following parameters;

- Granularity $G = 8, 16, 24, 32$,
- PE Unit Flexibility $F_{\mathrm{unit}} = 4, 5, 6, 7, 8$,
- PE Input Flexibility $F_{\mathrm{pi}} = 4, 8, 12, 16$,
- SE Flexibility $F_{\mathrm{sw}} = 2, 3, 4, 5, 6$.

Because of space limitations, the other parameters are fixed; the channel number $W = 4$, the array size $(N, M) = (4, 4)$, and the number of contexts $C = 32$. In this work, the analysis is limited only in PE array, and distributed memory modules provided in the edge of PE array are excluded.

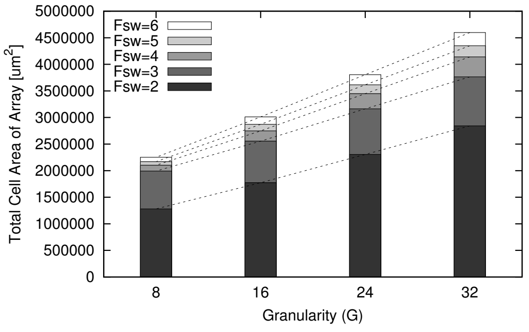

Fig. 4. Granularity vs. Total Array Area

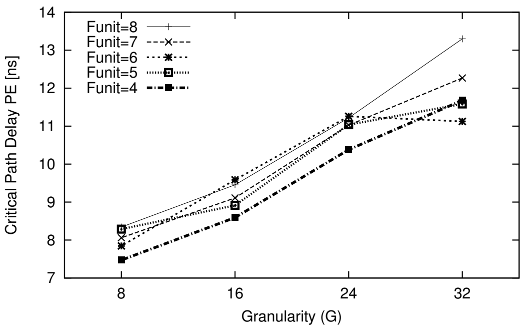

Fig. 5. Granularity vs. Critical Path Delay of PE

4.1. Granularity and Area/Delay

The PE granularity G is usually decided to match the data size mainly treated in the DRPA. Fig.4 shows the area of PE array with various G. The area is increased almost linearly with G independent of the SE flexibility (F_{sw}). The area becomes exactly double when G becomes 4 times (8bit to 32bit) with any F_{sw}. Given $G = 32$, only 1.5mm-2mm square die area is needed. This fact demonstrates that the DRPA is enough small to be used as an IP core in an SoC.

As shown in Fig.5, the critical path delay versus G is also increased with G, but the impact is rather modest compared with the case of area. If G is increased by 8bit, the delay increases about 2nsec in the 90-nm CMOS technology. This suggests that the large granularity is advantageous from the viewpoint of the critical path delay. The delay is also not so sensitive with F_{unit}.

4.2. Intra-PE Flexibility and PE Area

Fig.6 shows the total cell area of a PE for each G and F_{unit}. As prospected, the area becomes large with increasing G and F_{unit}. F_{unit} influences the area of input selectors of each functional unit and that of output selectors of connection blocks. In Fig.6, increasing area mainly comes from selectors.

Increasing F_{unit} also enlarges the area for a context memory as shown in Fig.7. However, it is not so severe compared with increasing of the cell area for a PE. That is, the increase of configuration bits is not sensitive to the increasing granularity. Moreover, additional configuration data for the PE Core is a certain constant value when G becomes double. Hence, from the viewpoint of the context memory, a large G is area-efficient.

4.3. Inter-PE Flexibility and Array Area

As discussed before, SE Flexibility or inter-PE flexibility F_{sw} gives an impact to the area of PE array. F_{sw} which shows the flexibility of the interconnection between PEs influences the area of SE and global routing resources, but does not directly related to the area of PE. Fig.8 shows the total cell area of 4×4 PE array for each G and F_{unit}. Like the case of a single PE, increasing G and F_{sw} enlarges the area of PE array.

5. REAL CHIP EXAMPLE: MUCCRA-1

As an example of DRPAs, we developed a prototype chip with a tool for application and evaluated its performance. The following parameters are selected for the prototype chip MuCCRA-1: $G = 24$, $F_{\mathrm{unit}} = 4$, $F_{\mathrm{pi}} = 4$, $F_{\mathrm{sw}} = 2$ and $C = 64$. The difference between MuCCRA-1 and DRPA architectures examined in this paper is as follows: (1) Since the 90-nm CMOS technology used in this work was not

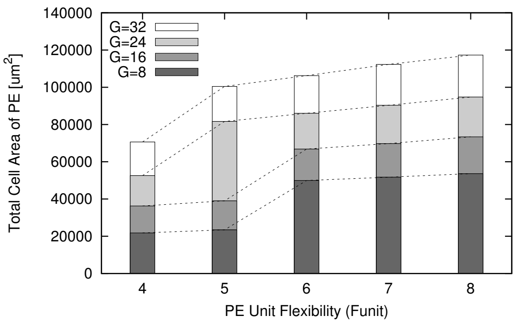

Fig. 6. PE Unit Flexibility vs. Total PE Area

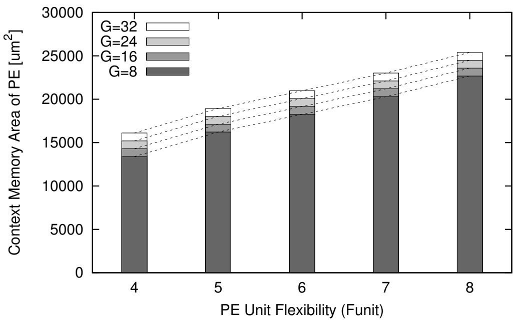

Fig. 7. PE Unit Flexibility vs. PE Context Memory Area

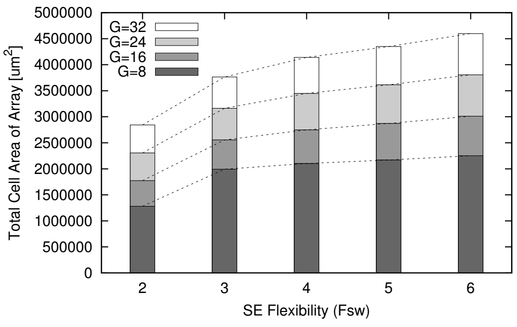

Fig. 8. SE Flexibility vs. Total Area of PE Array

available when MuCCRA-1 designed, the Rohm's 180-nm CMOS technology was used instead. It was implemented on 5.18mm x 5.18mm square die with 185 I/O pads. (2) The multipliers are separated from PEs and placed outside the PE array because of the chip limitation. In addition, in order to run large applications, virtual hardware mechanism and double buffering I/O are provided.

The MuCCRA-1 was taped out on the last November. Table 1 shows the execution time of designed applications. It works from 20MHz to 40MHz clock speed depending on the application design, and the execution speed is about twice of that of the Texas Instrument's digital signal processor (TMS320C6713) which works at 225MHz clock. This design experience demonstrates that DRPA architectures examined in this paper are practical for embedded systems.

Table 1. Execution Time for 4 Applications on MuCCRA-1

	BlockSize [bit]	ExecClocks	Delay [ns]	ExecTime [μs]
DCT	1024	195	40	7.8
α-Blender	8192	644	24	15.5
SHA-1	512	418	50	20.9
Viterbi	8	600	42	25.2

6. CONCLUSION

In this paper, we proposed a parameterized DRPA generator. By specifying architectural parameters such as PE granularity and several connection flexibilities, the generator can automatically generate a synthesizable Verilog-HDL description and verification environment.

We have generated various types of DRPAs and evaluated hardware area and speed by using the ASPLA/STARC 90-nm CMOS technology. From evaluation results, it appears that when the PE granularity changes from 8bit to 32bit, the area is doubled, and the delay time is increased about 6 nsec.

As the future work, we would like to establish the automatic design framework of application-customized DRPAs on the basis of this result. For this purpose, a re-targetable DRPA compiler is now under construction and we'll analyze architectural trade-offs based on real applications.

Acknowledgment This work is supported in part by Japan Science and Technology Agency(JST) and Japan Society for the Promotion of Science(JSPS). The authors thank to VLSI Design and Education Centor (VDEC), and Prof. Kobayashi and his colleagues in Kyoto University for their design flow of ASPLA/STARC 90-nm CMOS process.

7. REFERENCES

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

[2] 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)*, Aug. 2004, pp. 761–770.

[3] Rapport, Inc., http://www.rapportincorporated.com/.

[4] T. 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.

[5] Y. Kurose, 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*, Sept. 2004.

[6] Y. Hasegawa, et al., "Performance and Power Analysis of Time-multiplexed Execution on Dynamically Reconfigurable Processor," in *Proc. of IEEE Int'l Parallel and Distributed Processing Symposium (IPDPS)*, Apr. 2006.