Effect of parallel processing on a generic SoC platform using DPR

Hiroyuki Tanabe, Kohei Miyata, Akira Yamawaki*

Kyushu Institute of Technology, 1-1 Sensui-cho, Tobata-ku Kitakyushu-shi Fukuoka, Japan

*Akira Yamawaki: yama@ecs.kyutech.ac.jp

Abstract

Electronic products become more multifunctional and diversified. Thus, the SoC is needed to be larger and more diversified, be smaller and more high-performance. To satisfy these demands, enormous effort and cost of development are needed. However, such enormous effort and cost increase the cost of products. Therefore, a generic SoC platform which is able to be used in various applications is needed to reduce the burden of developers and cost. So far, the Dynamic partial reconfiguration (DPR) of hardware is one of the candidates to tackle such problem mentioned above. We have proposed a generic SoC platform, dynamic partial reconfiguration platform (DyREP), which uses the DPR nature. In this paper, we confirm an effect of parallel processing on DyREP, where the multiple reconfigurable modules can be executed simultaneously. Consequently, we attempt to clarify that the parallel processing on the DyREP can improve the performance.

Keywords: System on Chip, FPGA, Integrated Circuit.

1. Introduction

A system on Chip (SoC) mounted on electronic products plays important role of such products. Recently, electronic products become more multifunctional and diversified. Thus, the SoC is needed to be larger and more diversified. In addition, the SoC is requested to be smaller and more high-performance. To satisfy these demands, enormous effort and cost of development are needed. However, such enormous effort and cost increase the cost of products. Therefore, a generic SoC platform which is able to be used in various applications is needed to reduce the burden of developers and cost.

The Dynamic partial reconfiguration (DPR) of hardware is one of the candidates to tackle such problem mentioned above. The DPR can implement several signal processing hardware on a same hardware platform by reconfiguring the

hardware modules. Since the hardware modules can be reconfigured partially while the whole hardware system is running, the hardware reconfiguration does not affect the total performance. In addition, the DPR can handle many hardware modules that realize one signal processing by small die since the hardware modules are swapped on the small die.

Several researches have proposed their own SoC architecture using DPR feature[1][2]. Those architectures are specialized to the dedicated application for their researches. Also, the reconfigurable hardware modules have been specialized to those dedicated SoC architectures. That is, those hardware modules cannot be reused across several applications and SoCs. To use DPR feature for development of SoCs in high-mix low-volume products, a standardized form of the DPR hardware that is able to be mounted on the any SoC is needed.

We have proposed a generic SoC platform, dynamic partial reconfiguration platform (DyREP), which uses the DPR nature [3]. In our previous research[3], through the case study that binarizes an image and uses a prototype system of DyREP with an FPGA, we have shown that DyREP can evaluate the processing time, the reconfiguration time and the hardware size in order to make an optimum reconfiguration scenario while performing trade-off among them. Also, we have indicated that the small die of DyREP can efficiently realize one signal processing using several hardware modules while swapping them.

In this paper, we confirm an effect of parallel processing on DyREP, where the multiple reconfigurable modules can be executed simultaneously. Consequently, we attempt to clarify that the parallel processing on the DyREP can improve the performance.

The rest of paper is organized as follows. Section 2 explains the DyREP briefly. Section 3 discusses about results.

2. DyREP

Fig.1 shows the development flow of the DyREP. The reconfigurable block (RB) is a block that reconfigures hardware on the DyREP. The number of RBs can be appropriately set according to the application and hardware resource.

Figure1. Concept of DyREP

At Phase1, the user selects the partial reconfigurable module (PRM) from library which is the HDL hardware modules are implemented on the RBs. The user makes a scenario that shows how to reconfigure PRMs on the RBs by using the Gantt chart.

At Phase 2, the Gantt chart and PRMs in the library are converted to the virtual DyREP (VDyREP) in HDL. The Gantt chart is converted to the program of a reconfiguration management processor (ReMAP). The ReMAP fetches the instruction from the ReMAP memory. The partial reconfigurable port (RP) is a port that the circuit data of PRMs are input. An external configuration memory (ECM) and the external data memory (EDM) are attached to VDiYREP. The ECM holds the circuit data of PRMs to be reconfigured on the RBs dynamically. The EDM holds the data to be processed by the PRMs on the RBs. The RB has a mailbox (MB) is a status/control register file for the PRM in the RB. The ReMAP communicates with the PRMs in the RB via the MB. The RB has a bus interface of Wishbone bus[4] to access the EDM.

Finally, at Phase 3, VDiYREP is implemented into a physical DyREP (PDyREP). An implementation tool vary

according to the used real device like FPGA and ASIC with FPGA hard-macros. The implementation toll makes configuration data for the fixed part and RBs. The fixed part is previously configured. The hardware configuration data of the RBs are stored in the ECM. When the ReMAP executes the DPR, the ReMAP loads an appropriate configuration data from the ECM and writes the data to the RP. As a result, the PRM is reconfigured on the RB.

3. EXPERIMENT AND DISCUSSION

3.1 Experimental Setup

In this paper, we have developed the prototype system (DyREP-1) as shown in Fig.2. The DyREP-1 is mounted on the Xilinx XUPV5-LX110T board that Virtex5 FPGA supporting DPR is mounted. The prototype system runs at 50 MHz. The ReMAP memory consists of the Instruction Memory (IM), the Data Memory (DM) and the Reconfiguration Managing Table (RMT). The EDM and the ReMAP memory are implemented as the BlockRAM in the FPGA. The RBs connects to the EDM by a shared bus. The ECM is implemented as the on-board SRAM. The data transfer rate of the ECM to RP is 114MB/S. The Input/Output register file (IOR) is that the DyREP-1 can communicate with a personal computer (HOST PC).

Figure3. Architecture of DyREP-1

3.2 Partial Reconfigurable Module

In this experiment, we employ an image binarization as a case study of the DyREP. The used image is 256x256 pixels with 24bit contains 8bit R, G and B. The image binarization we developed consists of three hardware modules that are gray scaling, smoothing and binarization. They are designed as PRMs shown in Fig.3. To calculate fast, EXE states of those PRMs are pipelined.

In the gray scaling, input data is converted to YCrCb color space. As a result, the luminance (Y) is grayed value.

The data path of the gray scaling shown in Fig.3 (a) decomposes the following equations to binary operations. $Y = 0.29900 \times R + 0.58700 \times G + 0.11400 \times B$.

To make the fixed point calculation, the constant values are multiplied by 2^9 . The parameters of gray scaling are image size, read address and write address. At each state from EXE1 to EXE5, each binary operation is calculated. Finally, the value of Y is outputted shifting 8 bit right.

(a).Gray Scaling

(b).Smoothing

(c).Binarization

Fig.3 Hardware Design of used PRMs

In the smoothing, we have employed moving average filter as shown in Fig.3 (b). The smoothing parameters are same as gray scaling. The three line pixels of the image are loaded to the internal buffer (DI). At EXE1, same row pixels of each line are loaded to the row data buffer (D[0][i], D[1][i] and D[2][i]). The average of three row pixels is calculated at states from EXE2 to EXE5. Then, the calculated average is stored into the output buffer (DO) in the memory.

Fig.3 (c) shows the design of the binarization. The parameters of binarization are that parameters of gray

scaling and threshold. This binarization sets the output pixel to 0 if the input pixel is lower than equal to the specified threshold value. Otherwise, the input pixel is set to 1.

Fig.4 shows execution snapshots of which all PRMs are implemented into the FPGA and executed each of them sequentially. These snapshots indicate that each PRM correctly runs.

(a).Source image (b).Gray scaling (c).Smoothing (d).Binarization

Fig.4 Image processing result of PRMs.

3.3 Experiment Result

We prepare four DPR scenarios to perform preliminary experiment as shown in Fig.5. As for Fig.5, G means grey scaling, S means smoothing and B indicates binarization.

The FIX is the scenario that previously configures three hardware modules on three RBs and does not reconfigure dynamically as shown in Fig.5(a). It is equal to the general ASIC. The Serial is the scenario that reconfigures dynamically three hardware modules on one RB as shown in Fig.5(b). The Parallel2 is the scenario that reconfigures dynamically six hardware modules on two RBs as shown in Fig.5(c). In Parallel2, 256 x 128 pixels are processed per one RB. The Parallel4 is the scenario that reconfigures dynamically six hardware modules on four RBs as shown in Fig.5(d). In Parallel4, 256 x 64 pixels are processed per one RB.

The ReMAP program is converted from the Gantt chart is shown in Fig.6. Conf (R_{Bi}, module) reconfigures the specified RB by the specified hardware module on the ECM. Before reconfiguration, Conf disconnects the interface to the EDM and MB in order to avoid affecting to other parts. Start (R_{Bi}) sets the status and starts the specified RB and Wait_Finish(R_{Bi}) waits for completion of the specified RB. The program of the Parallel4 is omitted because it simply doubles the program of the Parallel2.

Tab.1 shows the execution time of three cases shown in Fig.5. In Tab. 1, Re means the reconfiguration time and Pr means the processing time for each PRM. Task time indicates total execution time including the reconfiguration time and the processing time as shown in Fig.6. The processing time of indicates that the process of two RBs is finished.

In the Serial, Parallel2 and Parallel4, all reconfiguration time are same. This is because that the size of all PRMs in Serial, Parallel2 and Parallel4, is same, 9.632KB.

The task time of Serial is the longest because the task time is simply summed of all reconfiguration time and processing time. Parallel2 and Parallel4 show the performance the same as FIX. This is because Parallel2 and Parallel4 can hide the reconfiguration time overlapping with the processing time, except for first reconfiguration. In addition, one RB can processes while another RB loads the date be processed from ECM.

Tab.2 shows number of used slices in three scenarios. The number of used slices of Serial is the fewest because Serial uses only one RB. The number of slices Parallel2 uses is fewer than FIX.

Thus, Parallel2 processing can achieve high performance while reducing number of used slices compared with FIX.

Fig.5 Gantt chart of three scenarios

Fig.6 ReMAP of three scenarios

Tab. 1 Reconfiguration time and Processing time [ms]

	GrayScaling		Smoothing		Binarization		Task time
	Re	Pr	Re	Pr	Re	Pr	
FIX	—	4.005	—	4.015	—	3.995	11.98
Serial	0.843	4.005	0.843	4.015	0.843	3.995	14.54
Parallel2	0.843	2.679	0.843	2.699	0.843	2.684	12.26
Parallel4	0.843	2.672	0.843	2.696	0.843	2.679	11.21

Tab.2 Number of used Slices in three scenarios

	FIX	Serial	Parallel2	Parallel4
Slice	512	220	440	880

4. Conclusions

In this paper, we confirm an effect of parallel processing on DyREP. In the experiment, we have verified the differences between serial processing and parallel processing.

Consequently, it can be said that the parallel processing can improve performance. Also, parallel processing can reduce hardware resources compared with implementing all the necessary hardware.

In the future, we are going to verify the operation in real time processing.

References

- [1] M. Taher and T. El-Ghazawi, "Virtual Configuration Management: A Technique for Partial Runtime Reconfiguration," IEEE Transactions on Computers, vol. 58, no. 10, pp. 1398–1410, 2009.
- [2] M. Majer and J. Teich, "The Erlangen Slot Machine: A Dynamically Reconfigurable FPGA-based Computer," Journal of VLSI Signal Processing, vol. 47, no. 1, pp. 15–31, 2007.
- [3] Hiroyuki Tanabe, Miyata Kohei, Yamawaki Akira, and Serikawa Seiichi : "A Generic SoC Platform using DPR Feature to Improve Social Life", 2012 IEEE/SICE International Symposium on System Integration(SII), pp. 125-128, 2012
- [4] OpenCores , CWishbone B4 WISHBONE System-on-Chip(SoC)Interconnection Architecture for Portable IP Cores, OpenCores , 2010