A Study of Dynamic Partial Reconfiguration on FPGA of High-level Synthesized Hardware

Takuma Sono and Akira Yamawaki

Kyushu Institute of Technology, Kitakyushu, Japan

*Corresponding Author: yama@ecs.kyutech.jp

Abstract

In recent years, hardware implementation of image processing is important to make embedded devices high performance and low-power. Hardware can be automatically generated by using high level synthesis (HLS) technology. The generated hardware becomes large, leading to increased chip size and price when a target image processing includes multiple functions. If we can introduce the concept of software dynamin link library (DLL) to hardware, large-scale processing software can be implemented in a small device which leads to price reduction. Our research aims to realize hardware level DLL that contains the circuit data of the HLS hardware in a library and dynamically reconfigures the required hardware module in a library on demand by Dynamic Partial Reconfiguration (DPR) on FPGA. To accomplish our research objective, this paper establishes a unified design flow from HLS hardware generation to circuit data generation. In addition, this paper develops a hardware platform supporting dynamic runtime-link of HLS hardware module. The experimental result shows that hardware runtime-link can be accomplished correctly and the reconfiguration time can be negligibly small.

Keywords: dynamic link library, high level synthesis, dynamic partial reconfiguration, FPGA.

1. Introduction

The market size of embedded image processing systems has increased and increasing. To occupy a significant market share, it is important to continuously provide high performance and low-power products quickly. Hardware implementation of image processing with high computational load is an effective way to realize high performance and low-power processing [1-2]. However,

hardware development is high burden for developer.

In order to reduce the load of hardware development, a technology for automatically generating hardware from software called high level synthesis (HLS) is used [3-5]. However, HLS has some challenges to make generated hardware efficient. A problem arises when HLS is used for large-scale processing including multiple functions. Generally, HLS cannot automatically divide the hardware into submodules, so it will generate one large-scale hardware containing all the functions. Then, a device that can implement such a large hardware is necessary, resulting in an increase in price.

On the other hand, the software conventionally can execute a large-scale program with a dynamic link library (DLL) on a small memory. The software DLL links a required function at run-time into the executable file on the memory from external storage like HDD and SSD. So the memory need not to hold the whole executable file. In addition, different executable files on the memory can share the same function in DLL. That is, all executable files need not to contain the same function individually. As a result, the memory has only to be enough large as to contain the cores of the executable files. If we can introduce the concept of this software DLL to hardware, a processing can be realized with small-sized device which is cheap and low-power, even if we convert software of large-scale processing to hardware by HLS.

Our research attempts to realize hardware DLL that contains the circuit data of the HLS hardware in a library and dynamically reconfigures the required hardware module in a library on demand by dynamic partial reconfiguration (DPR) on FPGA. This makes it possible to implement large-scale processing converted to hardware on a small-sized FPGA, which leads to price reduction.

For that purpose, we establish a unified design flow

from HLS hardware generation to circuit data generation and develop a hardware platform supporting dynamic runtime-link of HLS hardware module. Then, experiments are carried out to reveal the influence on the processing by reconfiguration.

2. Dynamic Link Library

2.1 Software Dynamic Link Library

The software library contains many functions relative to same objective like math, image processing, audio processing and so on. Generally, specified functions in the library are linked to the caller executable file by the software compiler. There are two kinds of linking methods, static link and dynamic link. Static link links the specified functions in the library to the executable file at compile time. On the other hand, dynamic link links the specified functions in the library at the run-time.

Each library file designed assuming each linking method is called static link library (SLL) and dynamic link library (DLL). Since SLL links all library functions specified into the executable file previously, the generated executable file becomes large. However, since DLL separates the library functions from executable file, the executable file with only core becomes small. That is, the memory need not to hold the whole of the executable file with library functions but just its core. As a result, the memory can be made small.

2.2 Hardware Dynamic Link Library

In order to introduce the concept of software DLL to hardware, the DPR on FPGA is used. DPR is a function that can reconfigure a specific circuit (Reconfiguration Module, RM) without stopping the entire circuit [6-7]. Also, the circuit data to be reconfigured are stored in the external memory. Figure 1 shows the concept of Hardware DLL.

In hardware DLL, each function of software is converted to circuit data of hardware through HLS, logic synthesis and implementation. The generated circuit data are archived into

Fig 1. Hardware DLL

```
#define F0299 38 // 0.299 * 2^7
#define F0587 75 // 0.587 * 2^7
#define F0114 15 // 0.114 * 2^7
#define F0500 64 // 0.500 * 2^7

void Grayscale( *src, *dst, size ){
// Assign arguments to the bus master
#pragma HLS INTERFACE m_axi port=src offs.=direct bdl.=d
#pragma HLS INTERFACE m_axi port=dst offs.=direct bdl.=d
    L1 : for( int i=0; i<size; i++){
        r = src[i];
        g = src[i] >> 8;
        b = src[i] >> 16;
        y = F0299*r + F0587*g + F0114*b + F0500;
        y = y >> 7;
        dst[i] = y << 16 | y << 8 | y;
    }
}
```

Fig 2. Software with pragma

a library on the HDD or SSD as well as that of software. Then, the necessary hardware is brought from the memory at the time of operation, written in RM, and the processing is executed. Conventionally, FPGA having a size enough to mount whole hardware has to be prepared and the price increases. However, in hardware DLL, since the hardware necessary for execution is brought from the memory, we can use cheap and small-sized FPGA.

3. Design flow of hardware DLL

In order to make hardware development quick and easy, HLS tool is used. HLS automatically generates hardware from software. However, when generating hardware with the HLS tool, it is necessary to add instructions to the HLS tool to the software. Figure 2 shows the software example of grayscale image processing used in this research. Software assigns arguments to AXI bus master port using #pragma HLS INTERFACE. The HLS tool analyzes the pragmas and generates hardware module with physical ports corresponding to the arguments of the function.

As shown in Figure 3, the HLS hardware module has many ports inferred by the HLS tool used. This hardware module has to be assigned to the physical region on FPGA which is dynamically reconfigured by many different HLS hardware modules. This physical reconfigurable region must be specified by the floor planning of FPGA. However, due to limitation of the physical reconfigurable region, the HLS hardware with raw ports cannot be used. So, the wrapper hiding such limitations must be prepared. In addition, the wrapper should have some memory mapped registers to invoke the HLS hardware from the software.

Fig 3. Design flow of hardware DLL

Next, logic synthesis is performed on each part, and netlists are generated from the HDL program. The netlists are implemented on the FPGA by using floor planed layout annotated with the physical reconfigurable regions. Finally, the circuit data are generated. The generated circuit data are archived into the hardware library.

4. Hardware platform for hardware DLL

Figure 4 shows the hardware platform for hardware DLL. This platform dynamically reconfigures the required circuit data and performs processing.

HW_wrapper provides a memory mapped register (MMR) and AXI bus port for the HLS hardware module. Also, ICAP is used for the FPGA itself to dynamically reconfigure internally. DPR is performed by writing circuit data in ICAP. However, the wiring must be shut off before reconfiguration since the data is always exchanged between AXI HP and RM. In addition, initialization reset must be applied so that the HLS hardware module works well after

reconfiguration. Therefore, CTRL shuts off the wiring and resets the HLS hardware module.

5. Experiment

5.1 Experiment environment

We investigate whether hardware DLL is effective for hardware implementation of software including multiple functions. In addition, we investigate how the hardware DLL affects the processing. In the experiment, software and hardware DLL are processed and the processing time of each is measured to investigate.

The HLS tool used is Vivado HLS 2016.4. The platform developed by Vivado 2016.4 is built on the Xilinx ZYNQ FPGA which is equipped on Digilent ZYBO. An embedded processor in ZYNQ is used for software processing. The embedded processor runs at 650 MHz. The hardware modules run at 100 MHz. A performance counter for measuring processing time and a module for visual confirmation were added to the platform. Each software of grayscale and histogram expansion was converted to hardware and used as circuit data. The processing of grayscale and histogram expansion are sequentially performed on the image.

In hardware processing, as an initial state, the grayscale circuit data is put in RM. The processing time of grayscale and histogram expansion respectively are measured. Then, the time obtained by subtracting the time of each hardware processing time from the total processing time is set as the reconfiguration time. Measurement of processing time is also performed for images of different sizes.

5.2 Experimental result

Figure 5 shows the processing time of software and

Fig 4. Hardware platform

Fig 5. Processing time

hardware DLL. Although the hardware processing time includes the reconfiguration time, the processing time of hardware is shorter than that of software at any image size. Therefore, the hardware DLL is effective for hardware implementation of software including multiple functions.

Figure 6 shows the proportion of each processing time in the hardware DLL. The reconfiguration time was about 18 milliseconds. Since the reconfiguration time depends on the size of RM, it does not change for images of any size. Therefore, the larger the image size is, the smaller the proportion occupied by the reconfiguration time becomes. For example, it is 26% for SXGA (1280×1024) and 18% for full HD (1920×1080). Therefore, the longer the processing time is, the smaller the effect of the reconfiguration time is negligibly.

6. Conclusion

Our research aims to realize hardware level DLL that contains the circuit data of the HLS hardware in a library and dynamically reconfigures the required hardware module in a library on demand by DPR on FPGA. To accomplish our research objective, this paper established a unified design flow from HLS hardware generation to circuit data generation and developed a hardware platform supporting dynamic runtime-link of HLS hardware module. In the experiment, we showed that hardware runtime-link can be achieved correctly and the reconfiguration time is negligibly small.

Future work include concealing arguments of software function arguments and shortening reconfiguration time.

References

[1] R. Karam, R. Puri, and S. Bhunia, “Energy-Efficient Adaptive Hardware Accelerator for Text Mining Application Kernels”, *IEEE Transactions on Very Large*

Fig 6. Processing time of hardware DLL

Scale Integration (VLSI) Systems, vol.24, no.12, pp. 3526-3537, 2016.

- [2] P. Zhou, T. Wang, X. Wang, and Y. Wang, “Hardware implementation of a low power SD card controller”, 2014 IEEE international Conference on Signal Processing, Communications and Computing (ICSPCC), pp. 158-161, 2014.
- [3] R. Nane, V-M. Sima, C. Pilato, J. Choi, B. Fort, A. Canis, Y. T. Chen, H. Hsiao, S. Brown, F. Ferrandi, J. Anderson, and K. Bertels, “A Survey and Evaluation of FPGA High-Level Synthesis Tools”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol.35, no.10, pp. 1591-1604, 2016.
- [4] J. Cong, B. Liu, S. Neuendorffer, J. Noguera, K. Vissers, and Z. Zhang, “High-Level Synthesis for FPGAs: From Prototyping to Deployment”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol.30, no.4, pp. 473-491, 2011.
- [5] A. Canis, J. Choi, M. Aldham, V. Zhang, A. Kammoona, J.H. Anderson, S. Brown, and T. Czajkowski, “LegUp: High-Level Synthesis for FPGA-based Processor/Accelerator Systems”, In *ACM FPGA*, pp. 33-36. 2011.
- [6] L. Pezzarossa, M. Schoeberl, and J. Sparso, “A Controller for Dynamic Partial Reconfiguration in FPGA-Based Real-Time Systems”, 2017 IEEE 20th International Symposium on Real-time Distributed Computing (ISORC), pp. 92-100, 2017.
- [7] A. Hassan, H. Mostafa, Hossam A. H. Fahmy, and Y. Ismail, “Exploiting the Dynamic Partial Reconfiguration on NoC-Based FPGA”, 2017 New Generation of CAS (NGCAS), pp. 277-280, 2017.