The Design and Implementation of Hardware Task Configuration Management Unit on Dynamically Reconfigurable SoC

Xiao-Wei WANG,, Wei-Nan CHEN, Ying WANG, Hong-Jun YOU,Cheng-Lian PENG
Computer Science and Technology School
Fudan University
Shanghai, CHINA
Email :{ 051021035, wnchen, clpeng} @fudan.edu.cn

Xiao-Wei WANG
Shanghai Branch
Navy Equipment& Technology Institute
Shanghai, CHINA

***Abstract*—The advantages and the flexibility introduced into the hardware implementation by partial dynamic reconfiguration have rapidly changed the design flow of embedded systems. Configuration management is an important issue in operating system for dynamically reconfigurable system-on-chip. Reconfiguration overhead affects the performance of reconfigurable system. This paper presents a hardware implemented efficient configuration management unit. Wet define our operating system framework based on unified multitask programming model for reconfigurable system-on-chip at first. Then the detailed design and implementation of the configuration management unit are given. Finally a use case is presented, which shows the efficiency of the configuration.**

Key Words: Dynamic Partial Reconfiguration, System-on-Chip, Operating System, Dynamic Hardware Task, Configuration Management Unit

I.. INTRODUCTION AND RELATED WORK

Dynamically Reconfigurable System-on-Chip (DRSoC)integrates dynamically reconfigurable hardware with embedded processor on the same die [1].It is always used for data stream driven, mixed hardware/software applications like data encryption, signal processing and so on [2][3][4]. The system is a combination of software and hardware, which strongly control-dominated at a system level and need real-time operating system (RTOS) services for synchronization, communication and configuration management. RTOS for HW management has been recently introduced. Proofs of concepts are exhibited in [6] and [7]. These experiments show that RTOS level management of reconfigurable architectures can be considered as available from a research perspective. In [6]， the author proposed communication API based on message passing where communication between HW and SW tasks are handled with a Hardware Abstraction Layer. In [7], the RTOS is mainly dedicated to the management (placement / communication) of HW tasks. In [8], more details are given about a Network-On-Chip communication scheme. The creation of hardware task is implemented by configuration, so our paper focuses on configuration management.

The main task for configuration management is to select the proper dynamic hardware tasks and write the corresponding files into configuration memory of the FPGA. The configuration overhead is the main performance bottleneck of dynamic partial reconfigurable system. One of the important jobs of the configuration managements is to reduce configuration overhead. After the design is placed and routed,there are three methods to short the reconfiguration time: reduce bitstream file size; optimize the way; optimize the bitstream transfer from memory to the configuration port like Internal Access Configuration Port(ICAP) of Xilinx's virtex family FPGA.

Most of the papers about reconfiguration overhead considered that configuration time only related with the size of the bitstream because the transfer speed was a constant. Based on the assumption that the configuration port can process incoming data every clock cycle, the maximum theoretical throughput can be calculated by the clock frequency and the width of the port. In fact, practical configuration speed is much lower than theoretical value. We will discuss it in section3. We propose the new configuration management unit implemented in hardware to enhance the practical configuration speed.

This paper is organized as follows: next section will make a brief view on operating system framework based on the unified software/hardware multitask model. In Section 3, we will analyze practical configuration speed of Xilinx FPGA. Section 4 illustrates the design and implementation of the configuration management which is implemented in hardware. The experiment results will be presented in Section 5, which shows the efficiency of configuration. Finally, Section 6 concludes the paper.

978-0-7695-3678-1/09 $25.00 © 2009 IEEE
DOI 10.1109/ICESS.2009.83

II. Operating System Framework Based on Unified Multitask Model

Due to the lack of effective uniform programming model for reconfigurable system, the migration costs of a function implementation from software to hardware are normally high. Even a small task migration needs an excessive modification, because it relates to different design teams. But the recent developments in configurable devices have increasingly blurred the traditional line between hardware and software. Using this excite characteristic, it seems that we can reduce the migration cost greatly.

Operating system is a reasonable solution because it is the traditional boundary between hardware and software. Although commercial RTOSs available for popular embedded processors provide significant reductions in design time, they typically do not take advantage of the intrinsic parallelism of hardware tasks, probably because FPGAs have historically been treated as hardware accelerators, for which there are only device drivers provided by the operating system.

To cope with this problem, we adopted a uniform multi-task (thread) model and implemented a RTOS prototype which is proposed for hybrid chips containing both CPU and FPGA components in one chip. We extend this model into the embedded system design that is composed of a host processor and several reconfigurable devices. This programming model allows hardware tasks on reconfigurable devices to execute in a truly-parallel multitasking manner, which are organized like software tasks, and substantially decreases the migration time for a task from SW implementation to HW implementation. The operating system framework prototype is shown in figure1. As figure 1 show, one of the most important services of the OS for reconfigurable system is to manage the HW tasks. The communication controller manages the data transfer between HW tasks and processor in communication layer. The creation and remove of hardware tasks (dynamic module) are implemented by the reconfiguration operations, which are the important operation for tasks scheduling. The configuration management unit is based on the devices' IPs, such as Xilinx's OPB-HWICAP and XPS-HWICAP. Conventional configuration overhead is calculated theoretical by the width and clock frequency of the configuration port, but it is much lower than the theoretical value in practice.

III. Conventional Hardware Task Configuration Management Unit

A . Hardware Task Configuration Port and Theoretical Configuration Bandwidth

Partial reconfiguration of Virtex devices can be accomplished in either Slave SelectMAP mode or Boundary-Scan mode (JTAG). Instead of resetting the chip and doing a full configuration, new data is loaded into a specified area of the chip, while the

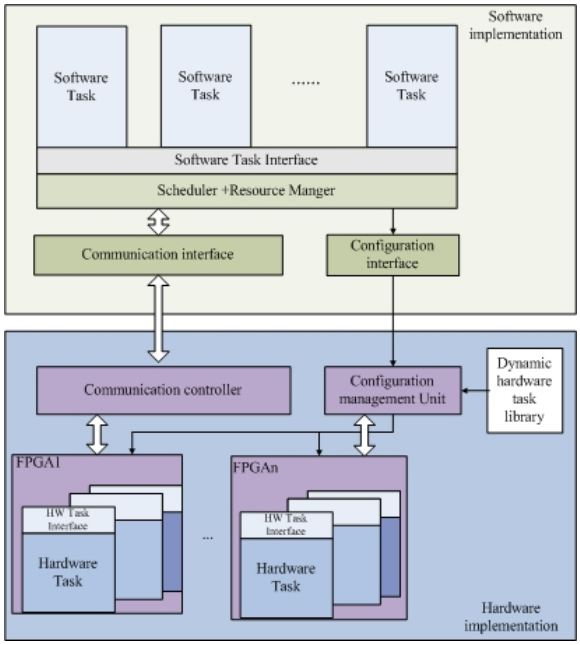

Figure1.The framework of OS based on unified multitask model

rest of the chip remains operating. ICAP is the subset of SelectMAP for Internal configuration access port, so its characters are the same as SelectMAP's. For self-reconfiguration system-on-chip, it uses ICAP IP cores like OPB-HWICAP or XPS-HWICAP to configure the devices, which connected through bus and controlled by CPU. The selectMAP ports of Virtex5 and virtex4 devices are 32-bit wide. So the theoretical configuration bandwidth (TCB):

$$TCB = DIW \times f_{cclk} \qquad (1)$$

DIW is data input width; fcclk is the frequency of the configuration clock.

TABLE I. THEORETICAL CONFIGURATION BANDWIDTH

Devices Family	JTAG 1MHz	ICAP/SelectMAP	
		fmax	*TCWmax*
Virtex2/Pro	125Kbytes/s	100MHz	100Mbytes/s
Virtex4	125Kbytes/s	100MHz	400Mbytes/s
Virtex5	125Kbytes/s	125MHz	500Mbytes/s

B Practical Configuration Bandwidth

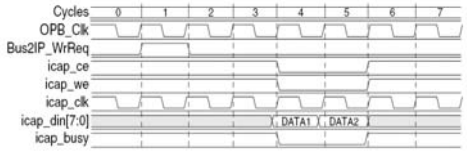

Figure2. opb-hwicap write cycle

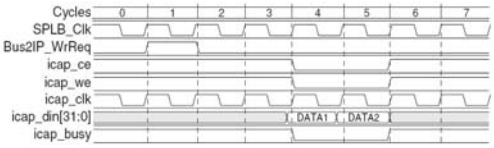

Figure3. xps-hwicap write cycle

The classic configuration timing diagram is shown in figure2 and figure3 [19][20]. The theoretical configuration bandwidth is calculated according the figures. But, in reality, it is not possible for the ICAP in virtex family devices to process new data every clock cycle.

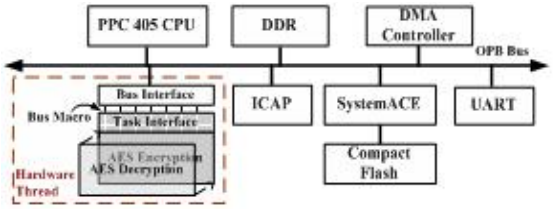

Figure4. the architecture of self-reconfiguration data encryption and decryption system-on-chip

When the clock frequency of ICAP is over 50MHz, it is necessary to respond to the ICAPs handshaking (busy) signal[]. This signal indicates whether the ICAP has accepted the incoming data or not. In Figure 5 the BUSY_ICAP signal is shown recorded from a reconfiguration sequence using the configuration management unit. It can be seen that ICAP is busy quite often which is indicated by the handshaking signal BUSY_ICAP [17]. Thus the maximum theoretical throughput cannot be achieved.

We created a self-reconfiguration application system for data encryption and decryption on XUPV2P board with XC2VP30 FPGA as figure 4 shows. We tested the reconfiguration speed, and the results are shown in table 2, which are far lower than theoretical value in table 1. For virtex-5, it is similar with virtex-2 and we will show the experiment result in section 5.

TABLE II. PRACTICAL TEST RESULT OF CONFIGURATION OVERHEAD ON XUPV2PRO BOARD

Name	size	Reconfigure time	throughput
AES Encryption	307525 Bytes	821.90ms	374.57Bytes/ms
AES Decryption	293821 Bytes	776.63 ms	378.63Bytes/ms

IV. THE DESIGN AND IMPLEMENTATION OF THE HARDWARE IMPLEMENTED CONFIGURATION MANAGEMENT UNIT

The configuration overhead is critical in many real time applications. For example, the auto vision system deals with video processing for driver assistance systems, where hardware tasks should be exchanged. The following scenario should describe why it is necessary to reduce the reconfiguration time. Using a frame input rate of 25 frames per second results in a maximum allowable time of 40 ms to process one image. If the image processing can be done in 35 ms, 5 ms are left over for reconfiguration. Assuming a reconfiguration can be accomplished within this timeframe, no frames must be dropped [10].

Many researchers consider that the configuration throughput is constant as the theoretical values; so their researches concentrate on how to reduce the bitstream size of the dynamic module to accelerate the configuration process. But they reduced bitstream file at most 20% as they reported[]. Compared with the practical values in table 2, it can not satisfy the requirements of applications in many real time situations.

The theoretical values are ideal and it can satisfy the requirements of the most real time applications. As section3.2 analyzed, the main affect factor is the BUSY_ICAP signal .CPU manages the configuration process; at least 4 cycles are consumed to get valid handshaking signals when the BUSY_ICAP signal appears. As figure 5 shows, the signals appeared so frequently. The configuration management unit for conventional hardware tasks in DRSoCs is implemented in software. Most of the configuration time is consumed by waiting for valid handshaking signals because of BUSY_ICAP signal.

Hardware implemented management unit can get efficient result in many situations. We implemented a hardware task configuration management unit in hardware, which is shown in figure 6.

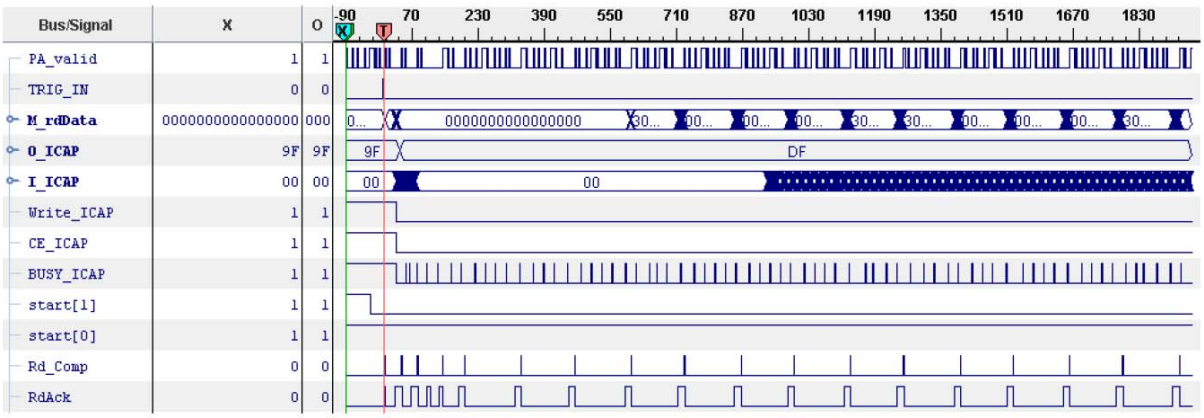

Figure5. the waveforms for hardware implemented configuration management unit using 100MHz Clock captured from ChipScope

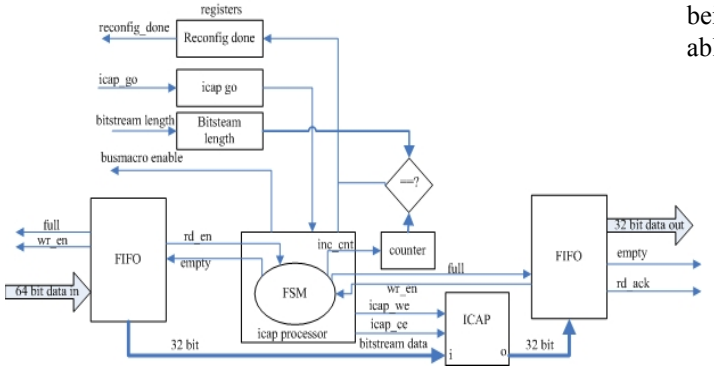

Figure6. diagram of hardware implemented configuration management unit

The configuration management unit is a PLB Master controller provides the necessary interface to transfer bitstreams to and from the ICAP. In order to avoid unnecessary overhead, the ICAP controller is equipped with 64-bit width FIFO. Not only does this minimize the amount of data which must be transferred, but also reduces the load on the CPU. After receiving the length of a bitstream from the controlling processor, the Master ICAP processor begins bursting in the required bitstream data directly from FIFO. Incoming data is stored within a FIFO, from where it can be fed one byte at a time into the ICAP. ICAP handshaking is also respected to allow for operation at the native bus speed (usually 100MHz in a system).The configuration start command will be sent along with the bitstreams size via the Device Control Register (DCR) bus directly to the ICAP controller. The controller then requests the bitstream data from memory and forwards it to the ICAP as described above. Once implemented, reading frames from the ICAP will proceed in a similar manner, except that frames will be accessed one at a time and stored internally within the ICAP processor rather than being written out to main memory. The CPU will then be able to read the frame data directly from the ICAP.

V. Case Study and Experimental Results

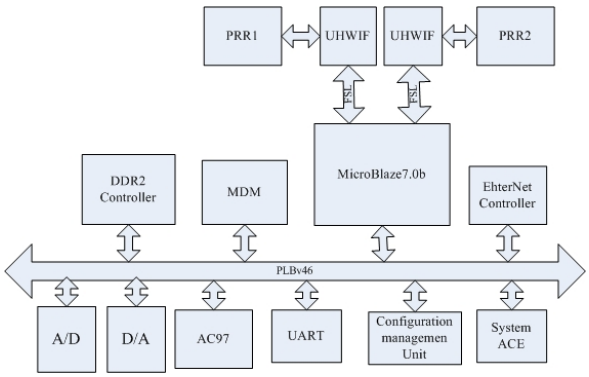

Figure 7 adaptive audio filtering system-on-chip prototypes

The prototype is built on ML505 board with XCV5LX50T FPGA as figure 6 shows. The hardware task configuration management unit implemented by hardware, which is responsible for dynamic hardware task creation and remove. The Microbalze7.0b CPU, system bus, peripherals, external DDR memory, as well as hardware tasks are all clocked at 100M Hz. We created device driver for configuration management unit in uclinux. Comparing with table2, the configuration throughput of virtex5 is nearly 7 times than that of virtex2pro FPGA when we use the original configuration unit provided by Xilinx. This is mainly due to the configuration port width because virtex-2 is 8-bit wide while virtex-5 is 32-bit wide. Comparing the data in table5 with data intable 6, hardware implemented configuration unit accelerates the configuration over 3.5 times. Comparing data in table 2 with data in table 6, the speedup of configuration is over 21 times. Because the

original configuration unit for virtex-4 and virtex-5 is xps- hwicap in XPS9.2 and

TABLE III. RESOURCE UTILIZATION AND TIMING SUMMARY OF XPS-HWICAP ON XC5VL50T

Resource name	used	total Resource	ratio
Number of Slice	594	28800	0%
Number of Flip-Flops/Latches	356	5980	0%
Number of Slice LUTs	1698	28800	0%
Number BRAM	0	7680	0%
Maximum Frequency	126MHz		

the later version of Xilinx's embedded development kit, which is a much larger and consume more internal resource than virtex-2's opb-hwicap. Comparing table 3 with talbe4, the configuration management implemented unit in hardware consume only a little more than the original configuration unit.

TABLE IV. RESOURCE UTILIZATION AND TIMING SUMMARY OF CONFIGURATION MANAGEMENT UNIT ON XC5VL50T

Resource Name	Used Resource	Total Resource	Ratio
Number of Slice	854	28800	0%
Number of Flip-Flops/Latches	686	5980	0%
Number of Slice LUTs	457	28800	0%
Number BRAM	4	7680	0%
Maximum Frequency	Minimum period: 5.830ns (Maximum Frequency: 171.527MHz)		

TABLE V. RECONFIGURATION TIME FOR THE ORIGINAL XPS-HWICAP

Dynamic Module	Bitstream Length	Configuration Time	Throughput
highpass	215KB	25.8ms	8.49KB/ms
bandpass	216KB	25.92s	8.45KB/ms
lowpass	215KB	25.8ms	8.49KB/ms
blank	161KB	19.32ms	8.46KB/ms
allpass	175 KB	21ms	8.49KB/ms

VI. CONCLUSIONS AND FUTURE WORK

In order to provide convenience for operating system to manage hardware tasks and reduce the configuration overhead of dynamic module for stream-oriented applications running on RSoC, a hardware implemented configuration management unit is proposed. The operating system framework based on multitask programming model and the theoretical are introduced, and the practical configuration time was

TABLE VI. RECONFIGURATION TIME FOR HARDWARE IMPLEMENTED CONFIGURATION MANAGEMENT UNIT

dynamic module	bitstream length	configuration time	throughput
highpass	215KB	86.6ms	2.54KB/ms
bandpass	216KB	87ms	2.53KB/ms
lowpass	215KB	86.6ms	2.55KB/ms
blank	161KB	64.85ms	2.54KB/ms
allpass	175 KB	70.49ms	2.55KB/ms

analyzed. The results tested from the using case show that the practical configuration overhead reduced over 3.5 times than the original on virtex-5 device, and over 21 times than that on virtex-2 device.

Many effective methods were proposed for virtex-2 and earlier device to reduce the size the bitstream of the dynamic partial hardware module to reduce the configuration overhead, but few can be used in virtex-4 and newer devices. We will analyze the detail of the virtex-5 and virtex-4's bitstream files, and optimize it to reduce their sizes because Xilinx's development tools no longer support virtex-2pro and earlier devices after ISE9.2 versions. We will improve our configuration unit and added DMA in it to accelerate the configuration in our operating system for reconfigurable computer.

REFERENCES

[1] K. Compton and S. Hauck, "Reconfigurable Computing: A Survey of Systems and Software", ACM Computing Surveys, vol. 34, no. 2, pp. 171–210, 2002.

[2] A. S. Zeineddini, K. Gaj, "Secure partial reconfiguration of FPGAs", Proceedings of the 2005 IEEE International Conference on Field-Programmable Technology (FPT'05): 155-162, 2005.

[3] Chang-Seok Choi, Hanho Lee. A reconfigurable FIR filter design on a partial reconfiguration platform [C]. Proceedings of the 1st IEEE International Conference on Communications and Electronics: 352-355, 2006.

[4] J. Y. Mignolet, S. Vernalde, D. Verkest and R. Lauwereins. Enabling Hardware-Software Multitasking on a Reconfigurable Computing Platform for Networked Portable Multimedia Appliances. In Proceedings of the International Conference on Engineering Reconfigurable Systems and Architecture: 116-122, 2002.

[5] C. Claus, F. H. M¨uller, and W. Stechele. .Combitgen: A new approach for creating partial bitstreams in Virtex-II Pro devices.. Workshop on reconf igurable computing Proceedings(ARCS 06), pages 122.131, March 2006.

[6] M. Vuletic, L. Pozzi, P Ienne. Seamless hardware-software integration in reconfigurable computing systems [J]. IEEE Design & Test of Computers, 22(2): 102-113, 2005.

[7] H. K.-H. So and R. W. Brodersen, "Improving Usability of FPGA based Reconfigurable Computers through Operating System

Support," in Proc. IEEE 16th Int. Conf. Field Programmable Logic and Applications, 2006, pp. 349–354.

[8] J-Y.Mignolet, V.Nollet, P.Coene, D.Verkest, S.Vernalde, and R.Lauwereins, Infrastructure for design and management of relocatable tasks in a heterogeneous reconfigurable system-on-chip," in Design, Automation and Test in Europe conf. (DATE), Munich, Germany, Mar. 2003.

[9] C. Claus, F. H. M¨uller, J. Zeppenfeld, andW. Stechele. .Using Partial-Run-Time Reconfigurable Hardware to accelerate Video Processing in Driver Assistance System, In Proceedings of the Design, Automation and Test in Europe Conference (DATE07), Nice, France, April 2007.

[10] K. Compton and S. Hauck. Reconfigurable computing: A Survey of Systems and Software. ACM Computing Surveys, 34(2):171–210, June 2002.

[11] P. Pande, C. Grecu, M. Jones, A. Ivanov and R. Saleh. Performance Evaluation and Design Trade-offs for Network-on-Chip Interconnect Architectures, IEEE Transactions on Computers, 54(8): 1025 - 1040, August 2005.

[12] Xilinx Inc. Two Flows for Partial Reconfiguration: module based or difference based. Xilinx Application Note. September 2004.

[13] The Open Group. IEEE Std 1003.1, available at http://www.opengroup.org/onlinepubs/009695399/toc.htm, 2004.

[14] Bo Zhou, Wei-dong Qiu, Yan Chen and Cheng-Lian Peng. SHUM-uCOS: A RTOS Using Multi-task Model to Reduce Migration Cost between SW/HW Tasks. In Proceedings of the 9th International Conference on Computer Supported Cooperative Work in Design (CSCWD'05): 984-989, May 2005.

[15] Xue-Gong Zhou, Ying Wang, Xun-Zhang Huang and Cheng-Lian Peng. On-line Scheduling of Real-Time Tasks for Reconfigurable Computing System. IEEE International Conference on Field Programmable Technology (FPT'06): 57-64, December 2006.

[16] Xilinx Inc. Microblaze Processor Reference Guide, Xilinx Documentation, May 2007.

[17] Xilinx Inc. OPB HWICAP, v1.00b, March 2006.

[18] xilinx Inc,UG002(v2.5)Virtex II platform FPGA User Guide,November,2007

[19] Xilinx Inc,U071 Virtex4 FPGA configuration User Guide, April 8,2008

[20] Xilinx Inc,UG191 Virtex5 FPGA Configuration User Guide,February,11,2008