Hardware-Software Monitoring Techniques for Dynamic Partial Reconfigurable Embedded Systems

Wang Xiao-Wei\textsuperscript{1,2}, CHEN Wei-Nan\textsuperscript{1}, PENG Cheng-Lian\textsuperscript{1}, You Hong-jun\textsuperscript{1}

\textsuperscript{1}Department of Computer and Information Technology, Fudan University, CHINA
\textsuperscript{2}Shanghai Branch of Navy Equipment& Technology Institute, Shanghai, CHINA
\{xiaoweiwang,wnchen,clpeng@fudan.edu.cn\}

Abstract

Dynamic partial reconfigurable embedded system includes at least one reconfigurable device, which is emerging as the new paradigm for satisfying the simultaneous demand for application performance and flexibility. But it brings difficulties in debugging, testing and performance measurement at the same time. Monitoring is an effective approach in debugging, testing and performance measurement in such complex systems. Dynamical partial reconfigurable monitor module based on the reconfigurablity of the hardware platform and the implement methods were proposed in this paper, which decrease the chip’s area consumption and reconfiguration time cost. The experiments about hardware-software monitoring approach showed expected results.

Key Words: Reconfigurable embedded system, hardware-software monitoring, reconfigurable monitoring module, partial reconfiguration

1. Introduction

Dynamic partial reconfiguration offers countless benefits across multiple industries. It can be an important component to any design or application—allowing designers more capabilities and resources than meets the eye. Partial reconfiguration is the ability to reconfigure select areas of an FPGA anytime after its initial configuration. You can do this while the design is operational and the device is active (known as active partial reconfiguration) or when the device is inactive in shutdown mode (known as static partial reconfiguration).

By taking advantage of partial reconfiguration, we will gain the ability to: adapt hardware algorithms, share hardware between various applications, increase resource utilization, provide continuous hardware servicing upgrade hardware remotely [1].

However, designing dynamic partial reconfigurable applications has been prohibitively difficult. Recent advances in FPGA architectures and implementation tools better support dynamic hardware module development[2][3]. At the same time, dynamic partial reconfiguration brings difficult in debugging and testing, few people carry about it. Monitoring is an effective way for debugging and testing complex systems such as distribute parallel and distributed system, dynamic partial reconfigurable embedded system. It is also an important means used in fault tolerant system and high reliable system. A special monitor system based on the reconfigurablity of FPGA was proposed in this paper, which decrease the chip’s area and reconfiguration time cost while satisfies the requirements of monitoring.

The remainder of this paper is organized as follows. Section 2 describes the related work. Section 3 outlines our application scenario and platform. Section 4 outlines our monitor module architecture. Section 5 reviews the method used to access the monitor architecture via the FPGA Boundary Scan interface. Section 6 presents an experiment result to show the monitor capabilities of our approach. Finally, Section 7 presents our conclusions.

2. Related Works

System monitor has traditionally been undertaken through one of three approaches: hardware, software and hybrid monitoring. The hardware monitor typically consist of connecting physical probes from the device as logic analyzer and signal scope. Software monitoring consists of adding instrument into the source code like soft probes. This approach collect hardware independent information but have significant performance penalty due to additional software overhead. Hybrid monitoring attempts balance the two approaches by providing a few software instructions to access selected hardware counters. However, portability can be issue with this implementation-specific approach. Triggering is the critical step in monitoring to ensure the accuracy of information.
There are two main types of monitors based on their trigger schemes. Time-driven and Event-driven monitors [11].

For reconfigurable system monitor is a new problem. Because hardware can be load to execute by reconfiguration through reconfiguration port like software’s executable file or thread be load to the program memory. That is to sat hardware is more like software in reconfigurable system, while hardware executes in parallel in nature. Debugging, testing and performance measure such systems include parallel executing hardware and software is a big problem for the developer.

Delon levi and Steven developed an interactive debug tool for Xilinx FPGA-based hardware—boardscope, using JBits tool [13]. But the tool only for hardware debug and JBits tool only support to Xilinx virtex II devices. None of the tool based on JBits was commercialized. And a lot papers published about FPGA testing by constructing testing structure on FPGA, but they could not testing the configuration hardware module that will run on the chip[17].

3. Application Scenarios and Platform

3.1. Application Scenario

It is a MicroBlaze-based processor system which communicates with AC97 CODEC controller to capture audio samples, process it through two reconfigurable filter regions (one for each channel), and output back the filtered sample. The two filters (one for each channel) are placed in reconfigurable regions and rest of the design is placed in the static region. The two reconfigurable regions can be reconfigured with one of the four filters (all pass, band pass, high pass, low pass) or with a blank bit stream under the control of user. The design is developed on Xilinx XUPV2P board which has CODEC chip on the board. The CODEC chip samples audio signal through Line In jack at 48000 Hz and sends sampled data serially to AC97 controller. AC97 controller is interfaced with OPB interface so it can be interfaced with the OPB bus. The AC97 controller presents sample in 16-bit parallel data form for each channel, first of left channel followed by right channel. The application passes the sample through a user desired filter (lowpass, highpass, bandpass, allpass) and presents the filtered sample back to the AC97 controller, which in turn sends sample serially to the CODEC. The output of the CODEC is fed to the Line Out jack on the board. The system diagram is shown below.

As shown in the figure 1, the system is a MicroBlaze-based system. Dual port 64KB block memory is used as LMB (Local Memory Bus) memory to hold the application program and data, and is connected to MB using ILMB (Instruction side LMB) and DLMB (Data side LMB) controllers. OPB MDM debug module is needed when System ACE controller is used. UARTLite core is used to interface with user using serial communication at 115200 baud, 8 data bits, and no parity. HyperTerminal or equivalent terminal emulation program can be run on a host system. System ACE controller is instantiated to interface with on-board ACE chip which communicates with the Compact Flash memory card. EMC Controller is used to provide additional RAM for a bigger program and/or large data. In this design it is not used. AC97 controller is used to communicate with on-board CODEC chip. ICAP is required to configure the reconfigurable region as and when desired.

This application includes software running on Microblaze—a risc 32-bit embedded CPU. The software managed the dynamic reconfigurable hardware module whose functions are filters. The hardware module looked as hardware task on the views of operating system which resident in partial reconfiguration region (PRR) in FPGA. Microblaze manages the partial dynamic hardware and blank reconfiguration region. When it needs to reconfiguration, the CPU load the bitstream of the dynamic hardware task and download to the FPGA through ICAP (Internal configuration Access Port) port.

Hardware task interface is the generalized interface between CPU and hardware task. It include necessary bus to communicate information and control signals to manage the hardware task, which is general to the most common hardware task used in reconfiguration.
3.2. Partial Reconfigurable Device

SRAM based FPGAs, like Xilinx’s Virtex family, enables reconfiguration of only a part of the reconfigurable fabric while other parts continue to operate. This way, different tasks of an application, or several independent applications, can be mapped to the reconfigurable hardware in sharing fashion exploiting the parallel processing potential. From an application point of view, partial reconfiguration allows it to change portions of its functionality without halting the entire application or without requiring extra hardware functionality (i.e. chip area)[4]. We use Xilinx’s XCV2P30 FPGA as the main device in the application.

3.3. Platform

Our prototype platform is XUP V2Pro development system (XUP-V2Pro) composed of XCV2P30 FPGA, Ethernet card, AC97 audio card, RS232, VGA interface and GPIOs, system ACE, Virtex™-II Pro FPGA with PowerPC™ 405 cores, Up to 2 GB of Double Data Rate (DDR) SDRAM, System ACE™ controller and Type II Compact Flash™ connector for FPGA Configuration and data storage, Embedded Platform Cable USB configuration port, High-speed SelectMAP FPGA configuration from Platform Flash In-System Programmable Configuration PROM, Support for “Golden” and “User” FPGA configuration bitstreams, On-board 10/100 Ethernet PHY device, RS-232 DB9 serial port, 2 PS-2 serial ports, AC-97 audio CODEC with audio amplifier and speaker/headphone output and line level output Microphone and line level audio input, On-board XSGA output, up to 1200 x 1600 at 70 Hz refresh, Three Serial ATA ports, two Host ports and one Target port, Off-board expansion MGT link, with user-supplied clock[5].

4. Monitor Module structure

As figure 2 shows, the application system’s structure and monitoring module. The application system includes CPU-Microblaze, ICAP, UART, EMC, GPIO, CODEC interface, dynamic hardware task interface.

The reconfigurable monitor module includes: one monitor controller, one to 8 probes modules can vary according to the signals monitored, one data memory controller, one data memory, one bus signals capturing probe and one software monitor module. All the sub modules work at the same clock that is the same as the system’s clock.

Figure2. System structure with monitor module

Probe module is composed of 64 signal probes, one signal comparator and one interface of data storage memory. Signal probes are signals multiplexers connect to the wires to be monitored, which do not modify its input signals. The width of probe signals and trigger conditions can be configured by monitor controller. When the signals be monitored satisfy the condition the memory controller will trigger the data memory to store the signal. The comparator is used to compare the signals monitored with the condition configured by the monitor controller.

The data memory controller manages the memory to store the data monitored. It accepts the command from the monitor controller, and then generates address for the data to store and necessary signals for the others modules read or write memory.

The data memory is used to store the data from the wire signals monitored and the event data from the CPU for software monitoring. It is controlled by the memory controller and can be transfer out of chip through JTAG port. Its width is 32-bit, depth vary according to the requirements and the chip’s area.

The bus signal monitor module is composed of trigger input logic, output logic, data capture logic. It is similar with Xilinx’s IBA OPB/PLB used to detect and record user-defined events under the control of the monitor controller.

The software monitor module controller connected to CPU through FSL (fast simplex link) bus accepts the command of the embedded CPU mainly used to monitor software threads. Xilinx FPGA has special instruction using FSL bus. The software code read from FSL is like” microblaze_bread_datafsl”, and write to FSL bus is like “microblaze_bwrite_datafsl”.

115
Monitor controller module accepts the command from monitor HOST computer through JTAG port. It sends instructions to sub modules such as probes module, data memory controller, bus signal monitor module to set the signal width and storage memory depth.

5. Hardware Monitor

Hardware monitor mainly capture interested signals, store the required data and show the signals wave. According to whether the monitor module loaded via partial reconfiguration, the hardware monitor approach classifies Integrate monitor and resident monitor. That is to say, our monitor module has two application modes: integrate monitor mode and resident monitor mode.

5.1 Integrate Monitor

Integrate monitor approach is using a number of simple and powerful probes module as described above coupled to critical point and interested wires within an application prior to insertion into reconfigurable embedded system. The probes module sample the via sample enable signal through monitor controller. In many cases, integrate monitor satisfy the requirements. But integrate monitor was inserted into the application fabric before the system implementation. It occupies the limited chip area and it lacks flexibility. If the users want to monitor other signals that were not connected before the implementation of the system, they have to implement the system again from the beginning. Because the probes can not change during the runtime of the system the probes can not move in the chip, more signal need to be monitored more probes modules needed and more chip area need in the limited embedded system. Resident monitor approach based the reconfigurability will satisfy the requirement without the drawback described above.

5.2 Resident Monitor

Resident monitor approach is that the system’s monitor module constantly loaded on the FPGA which application loaded via partial reconfiguration attach. A resident monitor module provides a standard user interface through which users can monitor signals of interest. It is an emerging technology play a great role in partial dynamic reconfigurable embedded system. But the implementation is more difficult than integrate monitor.

Xilinx provide two flows for partial reconfiguration: module based or difference based[6]. In the process of design the reconfigurable embedded system, it is better to use module based flow, because it have better tools support such as “Early Access Partial Reconfiguration Flow” which includes ISE partial reconfiguration patch and PlanAhead as a platform for reconfiguration design [7]. But when have few to modify such as probes move place or add a connection to the probe, difference based flow is more efficient in the process of implementation and reconfiguration.

6. Software Monitor

Monitor classifies time-driven and event driven monitor generally. The time driven monitor can only provide statistic information about the system, while the event-driven can record the information of the monitor points respective in detail, which can be used to debug and test during the design and to analyze and optimize the performance.

Software monitor generally implemented by inserting probe code into the program and recording the event information, which disturbs the program executing and needs memory to store the event data. But embedded systems have strict limitation in size and power consumption. The means used in reconfigurable embedded system are different from the usual distributed system. We use hybrid monitor mode, which uses special hardware as described in section 3 record respective event information and uses software probe code define the events in source of the program. The events are defined by program address, data address, and internal data in CPU. Thus, all kinds of events can be defined flexibly and status of the software runtime can be recorded fully while the program running process is disturbed little.

7. Experiments and Results Analysis
7.1 Experimental Setup

Several experiments have been performed to examine the monitor structures presented in this paper, special for residing monitor and software monitor. The monitor module inserted into the application described in section 2 and monitor structure as described in section 3. First, two probes to monitor the interface signals of left and right partial reconfigurable hardware task were inserted, next only one probe inserted only monitor left partial hardware interface, and then move the probes to right one. As to software monitor, monitor module as a dynamic reconfigurable module collect the events information about the time interrupt events and the interrupt server routine responding events.

The monitor module including hardware monitor and software monitor synthesized with Xilinx XST tool ISE9.1.02i PR, early access partial reconfiguration flow implemented on PlanAhead9.2.5; difference based flow was used to implement the partial bitstream of the hardware signal monitor. Modular based early access partial flow was used to implement the bitstream of the software monitor module. The configurations are run on Xilinx Virtex-II Pro 30, speed grade -7, located on the XUPV2P board. The host is run on IBM T60P with Intel Centrino Duo processor operating at 2.0 GHz, 2GB of memory.

7.2 Result Analysis

Several experiments are performed with hardware signal monitor and software monitor in partial dynamic reconfigurable embedded system running application described in section 2.

**Figure 4. Monitor PPR Left**

**Figure 5. Monitor PPR Right**

As figure 4 and figure 5 show that the difference of the two designs is very little, that is to say only move the probes in the chip. The synthesized result and partial bitstream size shown in table 1.

**Table 1. Two hardware monitor compare data**

<table>
<thead>
<tr>
<th></th>
<th>Monitor left</th>
<th>Monitor right</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slices of flip and flops</td>
<td>4405</td>
<td>4405</td>
</tr>
<tr>
<td>4 input LUT</td>
<td>435</td>
<td>435</td>
</tr>
<tr>
<td>Occupied slices</td>
<td>2624</td>
<td>2624</td>
</tr>
<tr>
<td>Bitstream size</td>
<td>1415KB</td>
<td>1415KB</td>
</tr>
<tr>
<td>Difference frames</td>
<td>109 frames</td>
<td></td>
</tr>
<tr>
<td>Partial bitstream size</td>
<td>92KB</td>
<td></td>
</tr>
</tbody>
</table>

When download through JTAG port the original download time for 1415KB bitstream is 5 second, when download the difference bitstream it takes 0.3 second.

**Table 2. Configuration time compare**

<table>
<thead>
<tr>
<th>Bitstream name</th>
<th>Size kbytes</th>
<th>Download time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>1415K</td>
<td>5 s</td>
</tr>
<tr>
<td>partial</td>
<td>92</td>
<td>0.3 s</td>
</tr>
</tbody>
</table>

The hardware monitor gets the signal wave was showed in figure 6. It show the time-sequence relation of the data-in signals, data-out signals, trigger out signal and the synchronization signal of the interface of the partial reconfigurable hardware module in the application described in section 2.

**Figure 6. Hardware signal wave from monitor module**
We used the event-driven hybrid software monitor have little disturbance to the running of the CPU. The data wave we get from the monitor as shown in figure 7. The top of the 16-bit data like “4F95” is the read in data through interface of the partial reconfigurable hardware module, and the following is the data sent to the codec interface. We insert special soft probe insert in timer interrupt and interrupt server routine as events, once the timer interrupt the count add 2 and send the count data out to the led port. The trace figure as figure8 showed.

From these data we can see hat resident monitor is better than integrated monitor.

8. Conclusions and Future Jobs

In partial reconfigurable embedded system, monitoring is an effective approach for debugging, testing and system performance measurement. To get useful information has to insert necessary hardware module in the system. Making use of the reconfigurability of the system, the monitor module can occupy smaller size of the chip, and event not occupy the chip’s area after testing and debugging if monitor module not be configured. Monitor based on the reconfigurability of such kind system can easily implement signal probes move in the chip and decrease the chip resource consumption and configuration time.

Our partial reconfigurable monitor module was implemented manual, we will develop computer aid software tools to auto generate and implement monitor module in the future.

9. Acknowledge

This work was supported in part by national natural science found project “operating system supporting reconfigurable computing” and the eleventh years’ plan project” reconfigurable computer design and implement technique”(513160703).We wish to thank the Xilinx university program(XUP) for software and development board donation.

10. References


