Reconfiguration Management on FPGA Platform for Cognitive Radio

Xiguang Wu, Sumit J. Darak, Pierre Leray, Jacques Palicot, Honggang Zhang

SUPELEC/IETR
Avenue de la Bouliaie CS 47601, F-35576
Cesson-Sevigne CEDEX, France
Email: {xiguang.wu, sumit.darak, pierre.leray, jacques.palicot, honggang.zhang}@supelec.fr

Abstract

Cognitive Radio (CR) is considered as a promising technique for future wireless communications due to the capability of adapting its behavior to its operating environment. Generally, a management architecture is necessary to efficiently manage the CR features and functionalities. Taking into account of the capability of dynamic partial reconfiguration of FPGA equipment, in this paper, we introduce the implementation of reconfiguration management on a FPGA platform. Based on this kind of reconfiguration management, all reconfigurable radio processing elements (PEs) can be reconfigured efficiently and flexibly, which also makes CR system more practicable and energy-efficient.

1. Introduction

In order to efficiently manage the Cognitive Radio (CR) equipments [1], our team has proposed a management architecture called Hierarchical and Distributed Cognitive Radio Architecture Management (HDCRAM) [2], [3], which is illustrated in Fig. 1. In order to practically implement the Cognitive Radio, efficiently reconfigurable and flexible hardware platforms are necessary.

Field Programmable Gate Array (FPGA) is a preferable platform due to its good performance and flexibility. Besides, some FPGA families have integrated embedded processors and have provided Dynamic Partial Reconfiguration (DPR) technique [5], which provide more flexibilities. DPR is the ability to dynamically reprogram a subset of logic in an operating FPGA by downloading a partial configuration bitstream while the remaining logic continues to operate without interruption [6]. Benefiting from these features and advantages, FPGA is more suitable for developing CR equipments. In this paper, we focus on the reconfiguration management part of HDCRAM on FPGA platform.

2. HDCRAM Architecture

A radio equipment can be considered as a set of functional components that are connected with each other. These components, illustrated as operators at the bottom of Fig. 1, can be either software or hardware. To become a CR equipment, a management part is needed to efficiently manage these operators. HDCRAM has been previously proposed to take charge of the management of the CR equipments. The representative HDCRAM architecture with three levels is depicted in Fig. 1.

HDCRAM has two aspects of management: cognitive radio management (CRM) and reconfiguration management (ReM). The CRM part is responsible for sensing and decision making in order to adapt the equipment to the environment. When a CRM unit makes a decision, it sends reconfiguration parameters to its associated ReM unit. It uses a bottom-up method in the CRM part. Sensing information is transmitted from lower level to upper level. The ReM part is in charge of the reconfiguration management. It takes a top-down approach in the ReM part. Reconfiguration commands are sent from upper level to lower level.

In general, HDCRAM is composed of three hierarchy levels. Each level has the Cognitive Radio Management Unit (CRMU) and its associated reconfiguration management unit (ReMU). At level 1, there can only be one CRM manager and one ReM manager, because this is the top level. At level 2 and level 3, multiple couples of CRMU and ReMU can exist. CRMU sends commands to its associated ReMU transversely at each level.

With this hierarchical management, it is possible to reconfigure an operator in different cycles with three scales: 1) a locally small cycle that only includes the operator and its associated level 3 management; 2) a medium cycle that
3. Reconfiguration Management Architecture on FPGA Platform

The reconfiguration management architecture comprises one PC and one FPGA, as shown in Fig. 2. The level 1 HDCRAM is unique and implemented on a PC. Therefore, on the FPGA side, the highest level is level 2. The Xilinx ML506 board is connected to PC by an Ethernet cable. The communication among different platforms of HDCRAM follows the User Datagram Protocol (UDP), which makes the communication easier and flexible. By this method, different devices do not have to be placed together very near to each other. They are connected with each other via Ethernet only requiring their IP addresses. It makes the system scalable so that we can add new devices easily, and need not change those devices that have already existed. Various components are necessary, and all these components work together, enabling the implementation of reconfiguration management on FPGA platform.

A. Hardware UDP Core

Based on the Embedded Hard Tri-Mode Ethernet MAC [7] provided by Xilinx, we have developed a hardware UDP CORE, which works at 1Gbits/s and thus provides a high speed transmission of data and partial bitstreams [8]. In addition to UDP protocol, it also supports Address Resolution Protocol (ARP). The reason of including the ARP protocol is that it allows FPGA to change its IP address thus to dynamically build up communication with different devices.

B. Demultiplexer and Arbiter

There are several different types of data, which should be correctly sent to the corresponding components. Depending on the destination port of the incoming UDP packet, we classify the incoming data into three kinds: command, processing data, and partial bitstream. But the UDP core has only one receiver, so a demultiplexer is necessary to switch the data path depending on the incoming data type. If the incoming packet is a command, it should be sent to level 2 ReMU implemented in Microblaze; if the incoming packet is processing data, it should be transmitted to processing elements (PEs); if the incoming package is a partial bitstream, it should be stored in SRAM. Likewise, an arbiter decides what kind of data should be sent to transmitter when the FPGA sends data to PC.

C. Microblaze

A level 2 ReMU is implemented in Microblaze, which is a soft processor core embedded in FPGA. The level 2 ReMU controls the level 3 ReMUs, both software ReMUs and hardware ReMUs. Multiple software ReMUs and software PEs can be created in Microblaze. They are software coded with C or C++ language, for instance a function in a class.
D. Hardware PE Controller

It is easy to implement such software ReMUs and PEs. Generally, software is flexible, but hardware has good performance. Therefore, we expect to have the hardware PE that is as flexible as software and at the same time keeps its performance. With this aim in mind, we have developed hardware PE controller, namely hardware level 3 ReMU, which is connected to Microblaze, as shown in Fig. 3.

![Fig. 3. Hardware PE controller.](image)

The interface between PE controller and Microblaze has several signals such as address, input, and output. In this way, we can have enough parameters only using these signals. As shown in Table I, address 0 corresponds to parameter 1; address 4 corresponds to parameter 2, and so on. Depending on the different values of the address signal, we can easily change the parameters of the hardware PE. For example, when it needs to reconfigure a parameter of the hardware PE, Microblaze first writes the address value corresponding to this parameter into the address signal, then writes the new value of the parameter into the input signal of the PE controller.

<table>
<thead>
<tr>
<th>Address</th>
<th>Parameter</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>P_1</td>
<td>S_1</td>
</tr>
<tr>
<td>4</td>
<td>P_2</td>
<td>S_1</td>
</tr>
<tr>
<td>8</td>
<td>P_3</td>
<td>S_3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>4(x-1)</td>
<td>P_x</td>
<td>S_x</td>
</tr>
</tbody>
</table>

There is one address signal, and in order to avoid conflict, only Microblaze can write values into this address signal. The hardware PE controller is not allowed to write values to the address signal, but it can read the value of the address signal. When the Microblaze wants to read the metrics of the hardware PE, it writes the address value corresponding to the metric into the address signal. Then PE controller writes the value of the metric into the output signal, and waits Microblaze to read it.

The hardware PE supports Dynamic Partial Reconfiguration. When it needs only to change the general parameters, it uses the method discussed above. When it needs to change the overall functionality of the hardware PE, it is better to choose the Dynamic Partial Reconfiguration approach. Besides, it is also possible to delete the hardware PE by downloading its corresponding black partial bitstream. The DPR feature makes our platform more flexible.

E. Bitstream Controller

When a partial bitstream is arriving, controlled by the Bitstream Controller, it is transmitted from the UDP core to the SRAM. Meanwhile, the Bitstream Controller sends the length of the incoming partial bitstream to Microblaze. Because Microblaze is more flexible than hardware logic, we choose Microblaze to manage the base address and the length of the partial bitstream. The base address of the first partial bitstream begins with 0, which is the beginning address of SRAM. In this way, we can calculate the base address of the next partial bitstream if we know the length of the current partial bitstream. In order to efficiently manage the partial bitstreams, Microblaze makes the base address and the length of a partial bitstream as a pair so that it can find the corresponding partial bitstream correctly when performing a partial reconfiguration.

F. Icap Controller and ICAP

Internal Configuration Access Port (ICAP) [6] is responsible for reconfiguring the specific portion of the FPGA. The Icap Controller is connected to Microblaze. When a hardware PE needs to perform a partial reconfiguration, Microblaze sends its corresponding base address and the length of the partial bitstream to the Icap Controller, then, according to the base address and the length, the Icap Controller reads the partial bitstream from the SRAM and sends it to ICAP. Finally ICAP reconfigures the region of this hardware PE.
Partial bitstreams are downloaded and stored in a 1MB SRAM so that it can reduce the reconfiguration time, because we can reuse the partial bitstreams many times after downloading and storing them in SRAM thus do not need to download them every time. The functionality of SRAM is similar to a local software library.

4. Discussion

The work in this paper uses the same partial bitstreams download approach presented in our previous work [8], which can reach 125Mbytes/s without considering the overhead, and nearly 120.6Mbytes/s taking into account the overhead. The reconfiguration management can achieve the maximum theoretical throughput (400Mbytes/s) of the ICAP during partial reconfiguration [6], [9].

In addition to the high performance, the reconfiguration management architecture provides more flexible approach to efficiently manage the PEs, thus enables the implementation of some potential CR and green scenarios. For example, the same functionality can be implemented with less resource, at the same time, reducing the power consumption. It is possible to implement the current existing system in a smaller device, which has several advantages. Small devices consume less industrial materials, thus potentially reduce production costs and CO₂ emissions. This also maybe makes small devices much cost-effective.

5. Conclusion

In this paper, we mainly explain how the reconfiguration management part of HDCRAM could be implemented on FPGA platform, what kinds of components are developed and used, and how they work together to achieve the functionality of reconfiguration management.

We will improve the reconfiguration management in our following work. Actually, it has already had some kind of cognitive features in the present work, since it can send information of FPGA to level 1 HDCRAM. Moreover, we are going to implement the full cognitive management of HDCRAM on the FPGA platform, which will be applied to some useful scenarios.

6. Acknowledgments

This work has received a French state support granted to the CominLabs excellence laboratory and managed by the National Research Agency in the “Investing for the Future” program under reference Nb. ANR-10-LABX-07-01. The authors would also like to thank Region Bretagne, France, for its support of this work.

7. References