Dynamic Partial Self-Reconfiguration on Spartan-III FPGAs via a Parallel Configuration Access Port (PCAP)

Salih Bayar and Arda Yurdakul

Boğaziçi University, Department of Computer Engineering
P.K. 2 TR-34342 Bebek, Istanbul, TURKEY
{salih.bayar,yurdakul}@boun.edu.tr
http://www.cmpe.boun.edu.tr

Abstract. *This paper presents an alternative approach for dynamic partial self-reconfiguration that enables a Field Programmable Gate Array (FPGA) to reconfigure itself dynamically and partially through a parallel configuration access port (PCAP) under the control of the stand alone PCAP core within the FPGA instead of using an embedded processor. The reconfiguration process is accomplished without an internal configuration access port(ICAP), which should be used either with MicroBlaze soft core or with PowerPC hard core using HWICAP core for the On-Chip Peripheral Bus (OPB)[6]. However, the stand alone PCAP core needs neither HWICAP core nor the OPB bus interface. The PCAP core needs only 324 slices, which is approximately 16% of a Spartan-3S200 FPGA. The dynamic partial self-reconfiguration via PCAP core works up to 50Mbyte/s. This approach has been implemented on a pure Spartan-3 FPGA from Xilinx, but it can also be used for any other FPGA architectures, such as Virtex-II(Pro), Virtex-4, Virtex-5, etc.*

Key words: Run-Time Reconfiguration, Partial Reconfiguration, FPGA, Self-Reconfiguration

1 Introduction

Partial reconfiguration is the ability to reconfigure preselected areas of an FPGA anytime after its initial configuration while the design is operational. By taking advantage of partial reconfiguration, hardware can be shared between various applications and upgraded remotely without rebooting and thus resource utilization can be increased [2].

The dynamic partial self-reconfiguration (DPSR) concept is the ability to change the configuration of part of an FPGA device by itself while other processes continue in the rest of the device. Normally FPGA devices can be reconfigured numerous times at runtime via an external intelligent agent such as a microprocessor, microcontroller, computer, or tester. A pure Spartan-3 FPGA, which doesn't have any multiboot capabilities and Internal Configuration Access Port

(ICAP), cannot be reconfigured without any additional external hardware. However, some other FPGA series such as Spartan-3A(N), Virtex-II(Pro), Virtex-4, Virtex-5 FPGA series have this ICAP module on their predesigned hardware architecture[8][9][10][11][12]. Although the Spartan-3 architecture is based on the Virtex-II and Virtex-II Pro architectures, the pure Spartan-3 family does not support the ICAP interface. In spite of the lack of an ICAP module on its architecture, dynamic reconfiguration is still supported in Spartan-3 via the external SelectMAP interface or JTAG, but not through the ICAP interface. Spartan-3 FPGAs support some of the dynamic partial reconfiguration capabilities, but with some limitations compared to Virtex devices.

Up to now, the lack of ICAP module on pure Spartan-3 FPGAs makes the DPSR impossible on these architectures without using any other additional external devices. ICAP is a functional subset of external parallel SelectMAP mode and is accessible internally via a user design. It allows the user design to control device reconfiguration at run-time. It becomes available after initial configuration is complete. That's why a component should be developed for pure Spartan-3 FPGAs, which acts as an ICAP and allows partial self-reconfiguration at run-time for Spartan-3 FPGA family.

In most cases a reconfigurable FPGA system consists of three main components: an external intelligent agent, some external (non-)volatile memory and a Complex Programmable Logic Device (CPLD). Such a reconfigurable FPGA system is described in detail in [5]. In some cases, systems may not require a CPLD if the used intelligent agent has a sufficient number of general purpose I/O (GPIO) pins. For these systems, the FPGA can be (re)configured directly by the intelligent agent[5].

In this study, a custom soft PCAP core is developed within the target FPGA, which controls the partial reconfiguration flow through SelectMAP port and supplies configuration clock for reconfiguration. Because of being written entirely in VHDL, this PCAP core is highly portable and can also be used for all other Xilinx FPGA architectures. Thus it is not necessary to use an external intelligent agent to control the partial reconfiguration flow. As a result, using this PCAP reduces hardware cost and power consumption of a self reconfigurable system.

The structure of paper is organized as follows: In section 2, we briefly explain the main types of partial reconfiguration and give a few developed samples associated with DPSR concept. Section 3 presents the structure and functionality of our PCAP core. Section 4 gives an example, where a run-time DCM reconfiguration via PCAP is implemented. Finally, section 5 presents our conclusions.

2 Dynamic Partial Self-Reconfiguration (DPSR) Concept in XILINX FPGAs

Partial reconfiguration is the ability to reconfigure a portion of the target FPGA while the remainder of the design is still operational. Partial reconfiguration is only possible through either serial JTAG interface or parallel slave SelectMAP mode. Since parallel slave SelectMAP interface has higher performance than the serial JTAG interface, the SelectMAP port is used in this study. Parallel SelectMAP port is used for either complete configuration or partial reconfiguration for applications, where the performance is the most important consideration. To be able to work with the SelectMAP interface, the SelectMAP pins should be persisted after the device is configured, which allows the SelectMAP interface to be used for reconfiguration[7].

There are mainly two types of partial reconfiguration: module-based and difference-based. The reconfiguration speed is directly proportional to the length of the reconfiguration bit information. In this project the bit length was very important because of storing the bitstream in the BlockRAM within the FPGA, which is normally very small for storing bit file. As a result of memory limitation for storing bit file, the difference-based method is used in this project, which is usually accomplished by making small changes in FPGA Editor and thus comes up with a very small bit length.

To be able to perform dynamic partial self-reconfiguration on a FPGA-based system, there should be either an internal configuration access port or an equivalent port. Some FPGAs such as Spartan-3A(N), Virtex-II(Pro), Virtex-4, Virtex-5 from Xilinx, which have ICAP on their hardware, support self-reconfiguration without using an external intelligent agent. The fact that the pure Spartan-3 does not have such an internal configuration port, has rendered impossible the self-reconfiguration without using an external intelligent agent up to now apart from a few new studies, which are discussed below.

In [3], a soft ICAP, known as JCAP, has been developed in order to realize the self reconfiguration. As a reconfiguration interface they use serial JTAG interface which is very slow compared to parallel SelectMAP port. The reconfiguration via ICAP, which supports a parallel internal interface and allows the FPGA application to access configuration registers, works with 66Mhz [8]. Though the ICAP on Virtex-II or Spartan3A devices have a reconfiguration speed 66MByte/s, JCAP only achieves a reconfiguration rate of 2Mbits/s. The reason of this huge performance difference between the ICAP and JCAP is the serial JTAG interface for JCAP.

In our study we have used parallel SelectMAP port instead of serial JTAG interface, thus we have developed a self-reconfigurable system on pure Spartan-3 series which should work at least 8 times faster than the developed system in [3]. Since a serial configuration method is used in [3], they achieved to send one bit per configuration clock cycle. However, in our study we use a parallel configuration method, hence we send 8 bits at each configuration clock cycle.

In addition to the reconfiguration rate comparison with the system in [3], we have used BlockRAM within the FPGA in order to store partial bitstream instead of using other 8 or 32-bit width registers, which should be controlled by a softcore within the target FPGA. Using BlockRAMs to store the partial bitstream reduces the access time compared to other external memories. Moreover, the partial bitstream flow is also controlled only by PCAP core, which makes a processor-independent run-time reconfigurable system available. Furthermore the PCAP core developed in this study can be associated with any softcore and thus a run-time reconfigurable softcore can be developed, which is very small and flexible.

In [1], a self-reconfiguration system on pure Spartan-3 has been developed. They have solved the lack of ICAP on Spartan-3 FPGAs by adding an external loopback, therefore they have used a GPIO core on MicroBlaze and 11 external wires to accomplish the interface through SelectMAP port. In order to store initial configuration bitstream and generate configuration clock signal they have also used a XCF configuration flash PROM. Under the control of GPIO core of MicroBlaze they reconfigure the target FPGA through SelectMAP port. Though they have achieved a speed as in ICAP, they have used an external PROM to store initial configuration bitstream, MicroBlaze soft core to control the configuration flow and a TFTP server and onboard SDRAM to store the partial bitstreams. Although we have also used 11 external wires and the SelectMAP port as a reconfiguration interface, we have used BlockRAM to store partial bitstream, PCAP core to control the reconfiguration flow. Since they designed a MicroBlaze-based system, they used 4198 slices of an Spartan-3S2000 FPGA. Applying such a system to small FPGAs (e.g. to a Spartan-3S200) is impossible. However, our PCAP core is very small, which is 324 slices and only 16% of a Spartan-3S200. Thus, as mentioned above, we have accomplished a processor-independent run-time reconfigurable system and presented a very new approach, which is storing partial bitstreams on BlockRAM within the FPGA.

In Virtex-4 devices the minimal reconfiguration unit is frame, thus a Virtex-4 FPGA can be reconfigured two dimensional by issuing some previous commands. A frame on a Virtex-4 device can be reconfigured while the rest of the device continues its normal operation. If some bits of the new frame do not change in comparison to the older one, it is guaranteed that there will be no glitches on this bits during the reconfiguration. However, on a pure Spartan-3 FPGA the minimal reconfiguration unit is a whole CLB column as shown in Figure 1. It is not guaranteed that no glitches will happen during the reconfiguration process. As a result of this problem the reconfigurable area must be comprised of whole CLB columns, from top to bottom of the device. The only issue is here the FPGA pinout, there must be no used pins in the reconfigurable area[1].

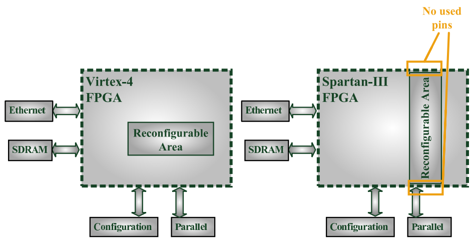

Fig. 1. Possible Reconfiguration Areas in Virtex-4 and Spartan-3

3 PCAP: A Soft Core For Dynamic Partial Self-Reconfiguration

The target FPGA Spartan-3 in our system is configured by itself through its SelectMAP port under the control of PCAP defined within the FPGA. Through the parallel SelectMAP interface, which is 8 times faster than serial JTAG interface at the same frequency, the partial reconfiguration information is accepted by the target FPGA and the reconfiguration process is executed again by the same target FPGA, where this unique FPGA acts as not only a *slave* but also a *master* at the time of reconfiguration as shown in Figure 2.

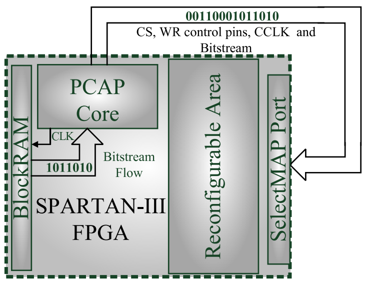

Fig. 2. Hardware architecture of whole system

As shown in Figure 3, in order to generate configuration clock frequency we have used a Digital Clock Manager (DCM) component, which is available within FPGA. Since we generate CCLK signal within FPGA, it acts as master, but at the same time we accept the CCLK signal through the SelectMAP interface into FPGA as if the signal comes from other intelligent agent, where the FPGA acts as slave. The source of CCLK is not important for the FPGA. The main point is that CCLK comes from outside. As a result of this, we should use the SelectMAP port in slave mode by setting the MODE pins as in Figure 3. The PCAP core reads a byte from the BlockRAM at each clock cycle. Under the control of CS, WRITE, CCLK signals, this byte is sent to SelectMAP interface. When the CCLK speed is set to 50 Mhz, the reconfiguration speed is 50MByte/s. Note that the reconfiguration speed is independent from the size of partial bitstream. The BUSY signal is only used if the configuration clock frequency exceeds 50 Mhz. In our study, the PCAP core can be configured to operate up to 50 Mhz. We have not exceeded 50 Mhz yet, that's why we have not used BUSY signal.However, it will be examined in the future work.

The configuration flow of an FPGA for run-time reconfiguration via PCAP is shown in Figure 4. First of all, we have generated an initial configuration bitstream with empty BlockRAM. However, in this initial configuration file, the BlockRAM is allocated for partial bitstreams. After generating the initial configuration bitstream we have generated also the partial bitstreams with the

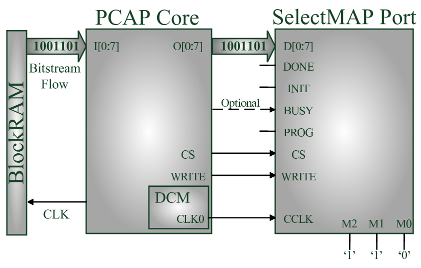

Fig. 3. PCAP Core and SelectMAP interface

help of bitgen -r flow[7]. Then we have converted the partial bitstream files to BlockRAM coefficient files. Afterwards we have loaded BlockRAM of Spartan-3 with these coefficient files, then we have generated the modified configuration bitstream that includes also partial configuration information.

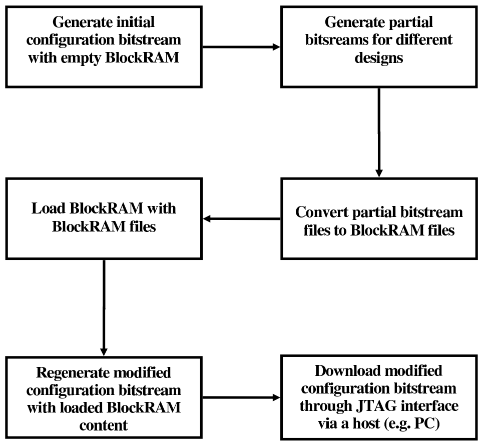
Generate initial configuration bitstream with empty BlockRAM
Generate partial bitsreams for different designs
Load BlockRAM with BlockRAM files
Convert partial bitstream files to BlockRAM files
Regenerate modified configuration bitstream with loaded BlockRAM content
Download modified configuration bitstream through JTAG interface via a host (e.g. PC)

Fig. 4. Configuration flow

The storage of partial reconfiguration bitstream requires also an additional external hardware for reconfigurable systems. In the most of reconfigurable systems the partial reconfiguration file is stored in an external non-volatile device. It can be read from there under the control of either an external intelligent agent or the FPGA itself, where FPGA acts as a *slave* and *master* respectively. Contrary to the standard methods based on storing partial reconfiguration bitstream on external non-volatile devices, the partial reconfiguration bitstream in this study is stored in BlockRAM within the target FPGA. As a result of this new approach, there is no need to use an additional external device for storing partial reconfiguration bitstream for any system, which works continuously after the initial configuration.

3.1 File Converter

To store information in a BlockRAM there is a predefined file type with extension ".coe" that can be associated with the memory coefficients. After generating partial bitstream files, we have converted these files to a suitable form with ".coe" extensions using a file converter module, which is written in Java language, in Figure 5. Note that the number of partial bitstreams needs not to be equal to the number of BlockRAMs.

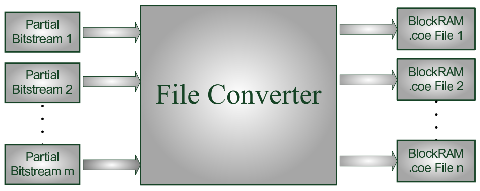

Fig. 5. File conversion from partial bitstream file to BlockRAM coefficient file

3.2 Dynamic Partial Self-Reconfiguration Flow

Since we have accomplished a dynamic partial self-reconfiguration through the SelectMAP port, the developed PCAP core behaves in our study as if it is a mirror of SelectMAP port, as same in ICAP. The following flow in Figure 6 is very similar to "SelectMAP configuration Flow Diagram" in [5] except that PROG, INIT, DONE, BUSY pins are not taken into account in our study.

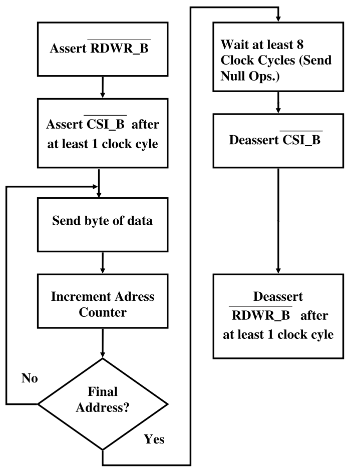
Assert $\overline{\text{RDWR\_B}}$
Assert $\overline{\text{CSI\_B}}$ after at least 1 clock cyle
Send byte of data
Increment Adress Counter
Final Address?
No
Yes
Wait at least 8 Clock Cycles (Send Null Ops.)
Deassert $\overline{\text{CSI\_B}}$
Deassert $\overline{\text{RDWR\_B}}$ after at least 1 clock cyle

Fig. 6. PCAP core Configuration Control Flow Diagram

The first three control signals PROG, INIT, DONE are only used during complete (re)configuration. BUSY signal is used if the configuration clock (CCLK) frequency is greater than 50Mhz. Due to the fact that there is only a 50Mhz oscillator available on Spartan-3 Starter Board we have chosen the CCLK for our system exactly 50Mhz, thus we do not need to use BUSY indicator signal. Under some circumstances, such as using BUSY indicator signal where it is needed, the developed design can be clocked with any other frequency values, which are supported by Spartan-3 FPGA and its SelectMAP interface. We have experimented, that our PCAP core can run safely at all frequencies up to 50Mhz. However, the performance of the PCAP core at higher frequencies will be tested after implementing the rest of the handshake signals in future.

4 Example: Run-Time DCM Reconfiguration

The soft PCAP core, which is a pure VHDL code, has been synthesized on Spartan-3S200 Starter Kit Board. In this work we have reconfigured a clock output of Digital Clock Manager (DCM), which drives the complete system, at run-time. Such a DCM reconfiguration approach is described in detail in [4].

There are various applications, where clock frequency of a system should be changed at run-time. Clock scaling method in [4] is one of them, which is mostly used to decrease FPGA power consumption by changing clock frequency of different components of system at run-time. Changing data transmission speed of a system, and other typical applications include speed drives, inverters, computers and computer controlled equipment, deep well pumps, industrial machinery, ships, aircraft.

In this work, a 4-bit up-down counter is taken as an example. This counter either works with 5 Mhz or 50 Mhz, which is accomplished by run-time DCM reconfiguration via PCAP core. The size of each partial bitstream for DCM reconfiguration is 5 KByte . The reconfiguration speed is 50 MByte/s, which means that the DCM reconfiguration via PCAP core takes approximately 0.1 ms. This counter is used solely to show that such a reconfiguration approach is possible for other reconfigurable systems where the frequency of a system or a component of system can be changed at run-time without affecting anything else in complete system.

To be able to do reconfiguration we have firstly generated complete bitstream files and then with the help of bitgen tool two fully routed NCD (Native Circuit Description) file for each different frequency value. Contrary to the approach in [4] for generating partial bitstreams, the difference based approach is used in this work.

All above mentioned components of this system can be viewed in Figure 7, which gives an overview of complete system in the FPGA-Editor. To store partial bitstreams only 6 of 12 BlockRAMs are used as shown in Figure 7, and solely 365 of 1920 slices are occupied and also 2 of 4 DCMs are used in this project. However, just for PCAP core 324 of 1920 slices, which is approximately 16% of a Spartan-3S200 FPGA, and 1 of 4 DCM are used. These results are

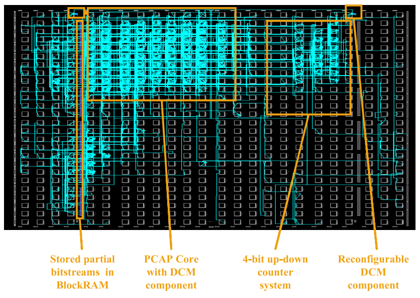
Stored partial bitstreams in BlockRAM
PCAP Core with DCM component
4-bit up-down counter system
Reconfigurable DCM component

Fig. 7. The complete system overview in FPGA Editor

shown in detail in Table 1, which summarizes the implementation cost of our study. Reconfiguring a DCM is actually reconfiguration of the single column where DCM is. As a result, it can be obviously seen that this PCAP core is much smaller than most of the soft controller.

Table 1. FPGA Resources

	Occupied FPGA Slices	*Occupied BlockRAMs*	*Occupied DCMs*
PCAP Core	324 of 1920	-	1 of 4
Partial Bitstream Storage	-	6 of 12	-
Up-down Counter	41 of 1920	-	1 of 4
System	365 of 1920	6 of 12	2 of 4

As a BlockRAM on Spartan-3S200 FPGA provides 16Kbit storage for data and 2Kbit for parity bits, we were able to utilize 16Kbit of a BlockRAM to store partial bitstreams. Since the size of each partial bitstream 5 KByte (40Kbit), we have used 3 BlockRAMs to store each partial bitstream. There are 12 BlockRAMs available on a Spartan-3S200, so we can reconfigure at most 4 columns with this method.

5 Conclusion

In this paper we have discussed dynamic partial self-reconfiguration of a pure Spartan-3 FPGA through the SelectMAP port and storing different partial bitstreams on BlockRAM within the target FPGA. The most important advantage of this study is to achieve a very fast partial reconfiguration compared to other serial JTAG interfaces and using a new approach such as storing partial bitstreams on on-chip memory and reading them from there under the control of an PCAP core instead of an external intelligent agent, which reduces hardware cost and power consumption simultaneously. However, it is obvious that the number of BlockRAM units which are used for storing the partial bitstreams cannot be neglected.

This kind of implementation, using no other additional external devices apart from FPGA, is a perfect solution for almost all systems based on cost and power consumption. What's also very impressive about this implementation is that the developed PCAP core can be applied not only on a pure Spartan-3 and also on other FPGA series such as Virtex-II, Virtex-4, Virtex-5, Spartan-3A(N) and so on.

In addition to this big advantage related to size, this PCAP core can be used anywhere in the FPGA, whereas the location of the ICAP module on Virtex-II devices and two ICAP modules on Virtex-4 are fixed [9][11].

In our future work, we plan to implement other handshaking signals on PCAP for providing self-reconfiguration at frequencies higher than 50MBytes/sec. Decoupling of the memory architecture from SelectMAP interface in PCAP will also be done in order to support different memory types and reconfiguration interfaces.

Acknowledgments. This work is fully supported by The Scientific and Technological Research Council of Turkey, TÜBİTAK (Project Nr.: 104E038) and Boğaziçi University Scientific Research Projects (Project Nr.: 06M105).

References

1. Ivan Gonzalez, Estanislao Aguayo, and Sergio Lopez-Buedo. Self-reconfigurable embedded systems on low-cost fpgas. *IEEE Micro*, pages 49 – 57, July-Aug. 2007.
2. Xilinx. Cindy Kao. Benefits of partial reconfiguration. *Xcell Journal*, pages 65–67, 2005.
3. K. Paulsson, M. Hübner, G. Auer, M. Dreschmann, L. Chen, and J. Becker. *Implementation of a Virtual Internal Configuration Access Port (JCAP) for enabling Partial Self-Reconfiguration on Xilinx Spartan-III FPGAs.* FPL, Amsterdam, Netherland, August 2007.
4. Katarina Paulsson, Michael Hübner, Salih Bayar, and Jürgen Becker. *Exploitation of Run-Time Partial Reconfiguration for Dynamic Power Management in Xilinx Spartan III-based Systems.* ReCoSoc2007, Montpellier, France, June 2007.
5. Xilinx. *Using a Microprocessor to Configure Xilinx FPGAs via Slave Serial or SelectMAP Mode.* XAPP502, (v1.4) edition, November, 13 2002.
6. Xilinx. *HWICAP CORE.* DS 280, (v1.3), March, 15 2004.
7. Xilinx. *Two Flows for Partial Reconfiguration: Module Based or Difference Based.* XAPP290, (v1.2) edition, September, 9 2004.
8. Xilinx. *Spartan-3 Generation Configuration User Guide.* UG332, (v1.2) edition, May, 23 2007.
9. Xilinx. *Virtex-4 Configuration Guide.* UG071, (v1.9) edition, October, 1 2007.
10. Xilinx. *Virtex-5 FPGA Configuration User Guide.* UG191, (v2.5) edition, October, 10 2007.
11. Xilinx. *Virtex-II Platform FPGA User Guide.* UG002, (v2.1) edition, 28 March 2007.
12. Xilinx. *Virtex-II Pro and Virtex-II Pro X FPGA User Guide.* UG012, (v4.1) edition, 28 March 2007.