Reconfigurable OPB Coprocessors for a Microblaze Self-Reconfigurable SOC Mapped on Spartan-3 FPGAs

Enrique Cantó, Francesc Fons
Dep. Electronic, Electrical and Automatic Engineering
University Rovira i Virgili
Av. Països Catalans 26
43007 Tarragona, SPAIN
canto@etse.urv.es

Mariano López
Dep. Electronics Engineering
Universidad Politécnica Cataluña
Av. Victor Balaguer s/n
08800 Vilanova i la Geltrú (Barcelona), SPAIN
lopezg@eel.upc.es

Abstract—Dynamically reconfigurable FPGAs are usually based on internal SRAM configuration memory, that can be fully or partially written from an external device. One of their applications is to map reconfigurable coprocessors, so an external microprocessor can change during run-time the coprocessor mapped on a FPGA. Coprocessors can execute the time-critical tasks of an algorithm, while the general purpose microprocessor executes the rest of computations and controls the FPGA reconfiguration. Microblaze is a soft-core 32-bit microprocessor designed to be implemented as a part of a System-On-Chip (SOC) mapped on Xilinx FPGAs. The Xilinx EDK software allows designers to map a SOC composed of a Microblaze plus several OPB (On-chip Peripheral Bus) peripherals. But the EDK was not designed to allow reconfigurable OPB peripherals mapped on the FPGA device. This paper demonstrates that it is possible to design a self-reconfigurable SOC mapped on a low-cost Spartan-3 FPGA, where an area section is devoted to map several reconfigurable OPB coprocessors in a time-multiplexed way.

I. INTRODUCTION

The shell of applications implemented on FPGA devices is continuously growing, since they offer several advantages when they are compared with ASIC counterparts, such as rapid-prototyping and lower NRE costs. Moreover, reconfigurable FPGAs can be updated during the lifetime of the applications, allowing the design upgrading without changing the device. A more advanced architecture is a dynamically reconfigurable FPGA, where the configuration memory is usually an internal RAM that can be fully or partially written from an external non-volatile memory or controller, changing the behaviour of the device at run-time. They can cover a large range of applications such as evolvable, self-reconfigurable hardware. Likewise, by using a temporal partitioning, they permit mapping several coprocessors multiplexed on time on a single device. When the application executed by the microprocessor requires a new coprocessor, a new bit-stream is downloaded from an external memory or controller.

A SOC is composed of a microprocessor and peripherals. Peripherals are dedicated to specific tasks, such as communications, timers, or others, while the microprocessor executes the rest of tasks. Usually the microprocessor is attached to coprocessors in order to speed-up critical computations. But the SOC area can be greatly increased if the number of coprocessor is large, because all of them are physically placed in the device. However just one of the coprocessors may be active during a time slot, while the rest are inactive. SOCs mapped on partially reconfigurable FPGAs can be self-reconfigurable, in order to map several reconfigurable coprocessors in the same devoted area, in a time-multiplexed way.

Xilinx FPGAs can be partially reconfigured during run-time, but their EDK software for SOC designs does not support this feature. The Xilinx EDK software allows designers to build a SOC mapped on a FPGA, composed of a soft-core 32-bit general-purpose microprocessor connected to peripherals through an OPB (On-chip Peripheral Bus) bus [1]. When the bit-stream is obtained from the tools, the FPGA can be fully reconfigured to map a static SOC. EDK does not offer support to map reconfigurable peripherals on the designed SOC, although this feature would very useful in order to map reconfigurable coprocessors.

Xilinx FPGAs can be partially reconfigured on the fly using the SelectMAP [2] external interface, designed to be connected to a programmer or to a platform configuration memory. Moreover Virtex-II family of devices includes an internal ICAP [3] reconfiguration plus internal tri-state routing resources [4] to easy self-reconfiguration capabilities. Unfortunately these features are not present in the low-cost Spartan-3 [5] family of FPGAs, but the SelectMAP is still available.

The paper presents a self-reconfigurable SOC for a Spartan-3 FPGA, composed of a Microblaze microprocessor, some OPB peripherals, and a reconfigurable OPB coprocessor. The SOC was designed to test the viability of reconfigurable OPB coprocessors on a self-reconfigurable SOC mapped on a low-cost FPGA. The organisation of the paper is as follows. Section II introduces some of the most important reconfiguration capabilities of the Xilinx FPGAs. Next section explains the top-level architecture of the self-reconfigurable SOC. Section IV briefly comments the main steps used to complete the design flow. Finally, section V describes the experimental results and conclusions.

II. RECONFIGURATION ON XILINX FPGAS

Xilinx FPGAs offer two external interfaces for reconfiguration: the JTAG and the SelectMAP. They were designed to be connected to a programmer or a platform configuration memory, and both interfaces can be used for partial reconfiguration. Virtex-II family has an embedded ICAP circuitry, designed to allow self-reconfiguration. It is very similar to the SelectMAP interface, but with internal ports that can be interconnected to other FPGA resources. The EDK software also includes an OPB peripheral, composed of the ICAP plus internal BRAM (Block RAM) memory, and an API (Application Programming Interface) to...
facilitate programming of applications that require self-reconfiguration.

In the other hand, Virtex-II family also features internal tri-state routing resources, that can be used as interconnection busses between reconfigurable area sections, in order to correctly communicate logic resources through areas between reconfigurations.

Although these Xilinx FPGAs offer support for self-reconfiguration, the EDK tools are not designed to easily map reconfigurable peripherals on a self-reconfigurable SOC. Moreover, low-cost FPGAs, such as the Spartan-3 family, lack of the ICAP and tri-state routing resources. Moreover, the minimum reconfiguration unit on a Spartan-3 FPGA is a column, instead of a frame on Virtex and Virtex-II devices, which contains the reconfiguration bits of an entire FPGA column, from the top to the bottom. We have developed a design methodology, based on [6], but modified to be applied on reconfigurable OPB coprocessors, instead of FSL (Fast Simplex Link) coprocessors. The FSL is a specifically designed link between one Microblaze and one coprocessor, while the OPB is an industry standard SOC bus. OPB peripherals or coprocessors can be attached in SOC systems with Microblaze or PowerPC microprocessors. Another difference of an OPB coprocessor is that the microprocessor uses regular read/write memory instructions to access the coprocessor registers, instead of the specific Microblaze FSL instructions. The disadvantages of an OPB peripheral are that the OPB is a more complex and slower bus interface.

III. SELF-RECONFIGURABLE SOC

The SOC is divided in two areas: the static area, and the reconfigurable area. The static area maps the Microblaze microprocessor, and some peripherals, while the reconfigurable area maps a reconfigurable coprocessor. Both areas are interconnected using bus-macros. The bus-macros are not composed of tri-state routing resources, because Spartan-3 FPGAs lack of them. They were developed different length bus-macros, (1-bit, 2-bit, 4-bit and 8-bit wide) to connect OPB signals between the static and reconfigurable areas. They were developed two types of bus macros, depending on the signal direction: from static to reconfigurable area denoted as bmX_mblaze2drip, or from reconfigurable to static area denoted as bmX_drip2mblaze (X is the bits width). Each bit of a bus-macro employs a LUT on the edge of the static area, a LUT placed on the edge of the dynamic area, and a routing net connecting the output of a LUT to an input of the other one, crossing both areas. The configuration of the routing resources and LUTs of a bus-macro are not changed when the reconfigurable area is reconfigured, ensuring the correctness connection between both areas.

The static area is composed of the microprocessor, plus its internal 8 KB BRAM memory, and four OPB peripherals: a UART (Universal Asynchronous Receiver-Transmitter), a GPIO (General Purpose Input-Output), a MDM (Microprocessor Debug Module), and a 7-segments peripheral. The MDM is used to debug applications executed by Microblaze, the 7-segments peripheral is used to show status on a display, and the UART to connect the system with an external PC. The external ports of the GPIO are physically connected with the external SelectMAP interface, using a very simple PCB board. This way, Microblaze can modify the port values in order to perform a self-reconfiguration of the SOC. In order to map a new coprocessor in the reconfigurable area, Microblaze reads a new partial bit-stream and drives the GPIO ports accordingly. Ideally, the partial bit-stream should be stored in a memory, to read it fast. Unfortunately, internal BRAM is too small to hold even one partial bit-stream. Another solution could be an external SRAM connected to the SOC using an OPB memory controller. This solution has not been tested because we used the Digilent Spartan-3 prototyping board, providing a 512K x 32 SRAM, besides a xc3v200 FPGA. The problem is that some SRAM pins are connected to the FPGA pins in the reconfigurable area. In the other hand, the minimum reconfiguration unit on Spartan-3 FPGAs is a frame. These facts do no permit to correctly maintain reads from the external SRAM while reconfiguring. To overcome this issue, Microblaze reads partial bit-streams from an external PC using the RS-232 communication channel and the UART peripheral with its ports placed in the static area. Although this way of reading bit-streams is very slow, it can be used to test the auto-reconfiguration capabilities of the SOC and the reconfigurable coprocessors.

Fig 1. depicts an overview of the top-level architecture of the SOC. The design is divided into two modules: the mblaze module (composed of the Microblaze plus the OPB peripherals), and the drip module (containing the reconfigurable OPB coprocessor). Both modules are interconnected through a set of bmX_mblaze2drip and bmX_drip2mblaze bus-macros, to correctly communicate the OPB signals. The ports TX and RX are the serial communication ports, while the dec7seg_* is the bus connected to the 7-segments display. The drip_led is a port that drives a led showing activity in the drip module. This led is attached to a port placed in the static area, so the signal is transferred from the reconfigurable area using also a bus-macro. The bus portreconf_* is the output port of the GPIO that is externally attached to the SelectMAP interface. Finally, the sys_clk and sys_rst are the clock and reset signals of the system.

IV. DESIGN FLOW

As commented previously, EDK does not support reconfigurable SOCs, so it has been necessary to use another design flow. The Xilinx ISE tools are used to synthesise a design, executing the placement and routing, and generating a bit-stream. They can be executed in the ‘modular design’
mode [7], to execute the placement and routing of the separated modules from its synthesised netlists, and build the top-level layout from them. There are three phases in the 'modular design', starting from a set of synthesised netlists. First phase is the initial bucketing, describing the top-level design as a set of interconnected modules, and it also sets its constraints. Second phase is the active module implementation that executes basically the placement and routing separately for each module. The last phase is the final assembly, joining the layout of the modules to build the top-level design, placed and routed.

Before using ‘modular design’ it is necessary to synthesise the different modules that compose it. The OPB coprocessors were synthesised using XST or Leonardo synthesisers. All the reconfigurable coprocessors must have the same external interface signals. It has been created three coprocessors: a dummy coprocessor, a 16-bit divisor coprocessor, and an 8-bit CORDIC coprocessor. The dummy coprocessor voids the reconfigurable area when the application does not require a coprocessor, while the other two ones are used by the application to test if they are working properly after a reconfiguration. These coprocessors are quite small because the mblaze module occupies about the 90% of area in the used FPGA, while only about a 10% of area is free for the reconfigurable coprocessor.

In order to synthesise the mblaze module, EDK software is used, enabling the sub-module option. As depicted in Fig. 2, the mblaze module contains the Microblaze microprocessor and the four OPB peripherals. Microblaze also uses internal BRAM blocks for data and instructions, interconnected using the instructions or data LMB (Local Memory Bus) buses. During reconfiguration, OPB signals from the drip to the mblaze module suffer glitches that hang up the Microblaze, so the mblaze module includes an isolate circuit, named OPB2IP, to disable these OPB signals from the drip module during reconfiguration. The bus-macros will be connected in the top-level design with the OPB2IP_* and IP2OPB_* ports of this circuit, so they are declared in EDK as external ports. Many of the output OPB_* signals of the bus are directly connected to its output OPB2IP_* port. It does not transfer all OPB_* signals because the reconfigurable coprocessor does not use all them. Concretely, the 12-most significant bits of the 32-bits OPB address bus are not transferred, in order to save area resources dedicated to the bus-macros and coprocessors. Also, the OPB_Clk signal is not transferred using this component, because this signal is attached to the system clock sys_clk, and it will be routed on the dedicated FPGA clock routing. The signal OPB_Rst of the bus is transferred to the OPB2IP_Rst signal using a logical function with the IP_Halt signal, asserting OPB_Rst when IP_Halt is asserted. This way, when a reconfiguration is in process, the IP_Halt is asserted to reset the reconfigurable coprocessor. The IP_Halt signal is internally connected to the reconfiguration GPIO, in order to be controlled by the Microblaze. Finally, signals IP2OPB_* are transferred to the input OPB_Sln_* bus using also a logical function, de-asserting them when IP_Halt is asserted, avoiding a Microblaze hang-up during a reconfiguration.

The top-level is finally synthesised, declaring the mblaze and drip modules as black boxes.

Before starting ‘modular design’ it must be written a UCF (User Constraints File) defining a specific FPGA area for every module and the placement of the bus-macros and pins. This constrains file is used during all the steps of ‘modular design’ to allocate correctly logic and routing resources during the placement and routing process. Code 1 shows some of the used constrains. As it can be deducted, the drip module occupies the first 4 column of slices in the FPGA, while the mblaze module occupies the 36 rest ones. It also declares the area of the BRAM and multipliers used by Microblaze.

```
Net sys_clk LOC=T9;
Net sys_rst LOC=L14;
Net TX LOC=E13;
Net RX LOC=T13;
...
INST bm8_drip2mblaze_DBus7to0 LOC=SLICE_X2Y46;
INST bm8_drip2mblaze_DBus15to8 LOC=SLICE_X2Y42;
...
AREA_GROUP "FIXED" RANGE=SLICE_X4Y0:SLICE_X39Y47;
AREA_GROUP "FIXED" RANGE=RAMB16_X1Y0:RAMB16_X1Y5;
AREA_GROUP "FIXED" RANGE=MULT18X18_X1Y0:MULT18X18_X1Y5;
INST "mblaze" AREA_GROUP="FIXED";
AREA_GROUP "FIXED" MODE=RECONFIG;
...
AREA_GROUP "DRIP" RANGE=SLICE_X0Y0:SLICE_X39Y47;
INST "drip" AREA_GROUP="DRIP";
AREA_GROUP "DRIP" MODE=RECONFIG;
```

Code 1. Examples of some constrains of the UCF file

The first phase in modular design is the initial budgeting phase. It creates the top-level design including the bus-macros, but with empty mblaze and drip modules. Second phase is the active module implementation, where it executes the placement and routing of every module, including its connection with the bus-macros and pins of the top-level design. Fig. 3(a) shows the layout of the mblaze module, while Fig. 3 (b) shows the layout of the drip module mapping the cordic coprocessor, after the execution of the second phase. Thanks to the UCF file, the layout of the bus-macros in all the reconfigurable coprocessors is the same, in order to ensure the proper interconnection with the mblaze module between reconfigurations. The third and last phase is the final assembly, joining the module layouts to build the complete layout top-level design. It must be built a top-level design for every coprocessor attached to the mblaze module, in order to extract partial bit-streams. Unfortunately, the final assembly phase does not work with the last service pack of the ISE 6.3
tools. To overcome the problem, we modified the PERL script used in [6]. The script joins the two module layouts interconnected using the bus-macros. Moreover, it deletes the routing of the sys_clk net design because it does not use a dedicated FPGA clock routing, in the drip layouts. After the execution of the script, a new routing is executed to route the sys_clk net using a dedicated clock routing. This way we obtain three top-level design layouts. Each one is composed of the mblaze module, one of the three possible coprocessors, interconnected with the bus-macros. Fig. 4 depicts the final layout of the top-level design with the cordic coprocessor.

When completed the top-level designs the bit-streams can be obtained. We created the full bit-stream that will configure the FPGA when it powers up, mapping the mblaze and the dummy module. We also created three partial bit-streams to update the reconfigurable area to the three different coprocessors.

V. EXPERIMENTAL RESULTS AND CONCLUSIONS

The application executed by Microblaze tests the divisor and cordic coprocessors, showing in the 7-segments display if they are working properly. A terminal PC program also shows the working state of the coprocessors, and permits to download a new partial bitstream to the SOC for self-reconfiguration. Obviously, the Spartan-3 prototyping board must be connected to an external board to attach the GPIO and SelectMAP interfaces. The system works properly, and also permits debugging the Microblaze application using the GDB tool between self-reconfigurations.

The main advantage of using a RS-232 serial interface as communication channel to transfer new bit-streams to the self-reconfigurable SOC is that it necessary to attach just two ports, RX and TX, to the static area. But the main disadvantage is the low bandwidth of the serial communication (115200 bps) and the size of the partial reconfiguration bit-stream (33 KB), so a reconfiguration takes a few seconds. Using an external SRAM can greatly decrease the time devoted to read the partial bit-streams, but all their interconnection ports (address, data and control busses) must be mapped in the static area. The Digilent prototyping board does not obey this constriction, so a custom-made board should be developed test this solution.

The presented SOC, composed of a soft-core Microblaze microprocessor, peripherals, and a reconfigurable OPB coprocessor, shows the viability of self-reconfigurable SOCs. Despite of the difficulties of using a low-cost Spartan-3 FPGA, and the lack of tools supporting self-reconfiguration, it can be designed successfully. Although we have presented a test self-reconfigurable SOC, due to the small FPGA size and the SRAM issues of the prototyping board, the design methodology can be used to design a SOC on industrial applications. This way, a self-reconfigurable SOC can map a large amount of reconfigurable coprocessors, without increasing their devoted area, while they are active in a time-multiplexed way. The ability of mapping a large amount of coprocessors is very profitable in complex applications,
where a static SOC, with a general-purpose embedded processor, may not have enough computational performances for the different critical tasks.

VI. ACKNOWLEDGMENT

Authors would like to thank financial support from Ministerio de Educación y Ciencia de España, under grant DIP-2003-08637-C03-03.

VII. REFERENCES

[4] Xinlinx, Two Flows for Partial Reconfiguration: Module Based or Difference Based, XAPP290 (v 1.2), Sept. 2004
[5] Xilinx, Spartan-3 Complete Data Sheet (v 2.0), Mar. 2006