2018 13th International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS)

Dynamic Partial Reconfiguration Verification Using Assertion Based Verification

Islam Ahmed[1], Hassan Mostafa[2,3], Ahmed Nader Mohieldin[2]
[1]IC Verification Solutions, Mentor Graphics, a Siemens Business, Cairo, Egypt
[2]Electronics and Communications Engineering Department, Cairo University, Giza 12613, Egypt
[3]Nanotechnology Program at Zewail City of Science and Technology, Cairo, Egypt
{islam_ahmed@mentor.com, hmostafa@uwaterloo.ca, anader2000@yahoo.com}

***Abstract*—Dynamic Partial Reconfiguration (DPR) on Field Programmable Gate Arrays (FPGAs) allows reconfiguration of some of the logic at runtime while the rest of the logic keeps operating. This feature allows the designers to build complex systems such as Software Defined Radio (SDR) in a reasonable area. However, utilizing DPR needs extra care to be taken for new issues such as waiting for running computations on a module before reconfiguring it, isolation of the reconfigurable modules during the reconfiguration process, and initialization of the reconfigurable module after the reconfiguration process is done. This paper proposes a technique to verify these newly introduced issues using Assertion Based Verification (ABV). The proposed technique proves effectiveness in finding issues on real designs that utilize DPR technique.**

I. Introduction

Dynamic Partial Reconfiguration (DPR) [1] on Field Programmable Gate Arrays (FPGAs) allows reconfiguration of some of the logic at runtime while the rest of the logic keeps operating. It allows the implementation of complex circuits such as Software Defined Radio (SDR) and Internet of Things (loT) applications within reasonable area on the FPGA, and consequently the power consumption of the circuit is reduced.

In DPR, the design consists of a number of Reconfigurable Modules (RMs), each module has modes that are changed during runtime according to the system operating modes. A Reconfigurable Region (RR) is a location on the FPGA in which the reconfigurable module is implemented on. The dynamically reconfigurable systems extend the design flexibility through mapping of multiple reconfigurable modules to the same physical reconfigurable region, which reduces the design cost and the resource usage.

This paper is organized as follows: Section II presents the motivation and the research idea, Section III presents the proposed verification methodology for DPR logic. Section IV demonstrates a case study for the proposed flow. Finally, Section V draws the paper's conclusion.

II. Motivation and Research

Using DPR technique for FPGA designs adds a new challenge in the design and verification of FPGA designs, as designers must add extra logic in their DPR designs for 1) isolating the RM during the reconfiguration process, 2) initializing the RM after the reconfiguration process is done, and 3) delaying reconfiguration requests till the computations done by the RM is completed. The added logic for these tasks should be verified on the RTL to make sure it is working as expected, and any bugs are caught as early as possible in the design cycle. The detection of real reconfiguration issues is very challenging especially in the early design stages, if such errors are not tackled and verified early in the design cycle, they may cause functional errors during on-chip verification which are hard to debug. In this paper, a new methodology is proposed to verify the DPR logic using Assertion Based Verification (ABV) [2], we consider the DPR flow for Xilinx FPGAs in this paper [1].

The typical structure of designs that utilize DPR is shown in Fig. 1, the Internal Configuration Access Port (ICAP) [1] is used to read or write to the FPGA configuration memory. A controller is needed for the ICAP to handle the reconfiguration requests and monitor its status. The output port of the ICAP can be used to monitor its status. The yellow blocks in Fig. 1 are added by the designer to have a correct operation for the design during and after the reconfiguration process [1]. Our verification approach is to model the functionality of the DPR logic using SVA [3] properties, then verify these properties on the design using formal or simulation methods.

III. ABV for Dynamic Partial Reconfiguration

During the reconfiguration process of the RM, the values of newly downloaded bitstream may drive incorrect values to the static logic side, so designers add isolation logic for all the ports of the RM to prevent the propagation of the data from the RM to the static logic during the reconfiguration process.

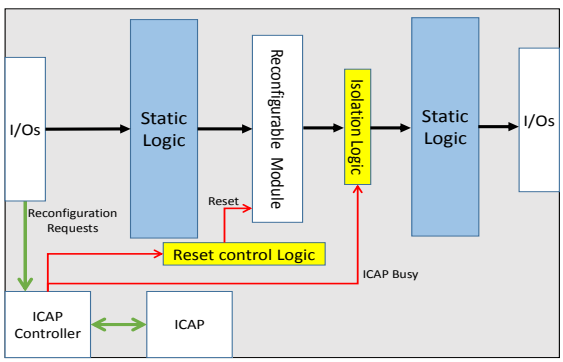

Fig. 1. Typical structure of a design that utilizes DPR.

978-1-5386-5291-6/18/$31.00 ©2018 IEEE

This is verified using the following SVA property for every output port of the RM:

```
property verify_isol(clock, source, destination, ICAP_BUSY);
 @(posedge clock)
 (($changed(source) && ICAP_BUSY) |=> $stable(destination));
endproperty
```

Where the *source* and *destination* signals are the output port and the register driven by that output port respectively, and the *ICAP_BUSY* is the signal which indicates that there is a reconfiguration process in progress.

After the reconfiguration process is done, the sequential elements of the RM should be reset to guarantee proper operation of the circuit. If the RM is not reset after reconfiguration, the state of sequential elements will be undefined and may be affected by erroneous values from the previous RMs that share the same physical area on the FPGA. The reset control logic is verified using the following SVA property:

```
property verify_reset(clock, RM_reset, ICAP_BUSY);
 @(posedge clock)
 ($fall(ICAP_BUSY) |-> $rose(RM_reset));
endproperty
```

When a computation is being done in the RM, the designers want to block any reconfiguration request until such computation is done. Such mechanism is required in applications like SDR, when a packet is being processed for WiFi standard as example, it should be processed completely before switching to any other standard like 3G or 4G. This is verified using the following SVA property:

```
property verify_sync(clock, RM_busy, ICAP_GO);
 @(posedge clock)
 ($rose(ICAP_GO) until $fall(RM_busy));
endproperty
```

Where *RM_busy* is the signal which indicates that a computation is being done by the RM, and *ICAP_GO* is the control signal which tells the ICAP to start a new reconfiguration process.

IV. Case Study

The approach presented in this paper is applied on an SDR chain presented in [4]. The SDR test case has four RMs, the block diagram of the design is shown in Fig. 2. The SVA properties are verified using Questa PropCheck [5] tool. The number of assertions for the isolation logic equals to the number of output ports for all the RMs, the number of assertions for the reset control logic equals to the number of RMs as each RM will have its own reset control logic, and only one assertion is generated to test the synchronization logic of the DPR controller. Table I shows the number of ports for every RM, and Table II shows the number of assertions generated for verification of the DPR logic.

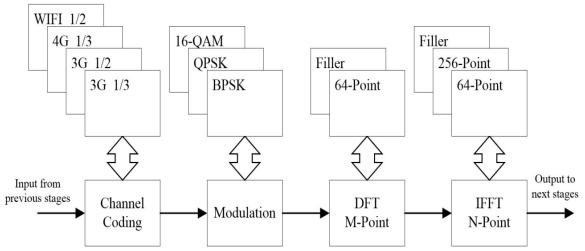

Fig. 2. Block diagram of the design under test [4].

TABLE I
Ports information for RMs of the design under test

Block	Total No. of Ports	No. of output ports
Convolutional Encoder	6	2
Modulator	7	3
DFT	7	3
IFFT	7	3

TABLE II
Generated Assertions for DPR Verification

Goal	No. of Assertions
Isolation Logic for Output Ports	2+3+3+3 = 11
Reset Control Logic	1*4 = 4
Synchronization Logic	1
Total	16

When verifying the assertions on the design under test, we were able to identify three bugs:

1) The output ports of the RMs were not isolated during the reconfiguration process.
2) The reset signals of the RMs were not activated right after the completion of the reconfiguration process.
3) The DPR controller was not handling the case in which a new reconfiguration request is received when the RM is still processing data.

V. Conclusion

In this paper we presented a verification flow for DPR using ABV. Designers can use this flow to verify their DPR designs and the dedicated logic added for DPR activities like the isolation logic, reset control logic and the synchronization logic of the DPR controller, SVA properties are used to verify these functionalities. The verification should be done on the RTL design before moving to implement it on the FPGA to make sure of the correct functionality of the design, as any error caught during verification will force the designs to restart the implementation cycle after fixing the functional errors in the design. Using a case study from literature we demonstrated how the proposed verification flow identifies three issues in the DPR logic of the design.

References

[1] Xilinx, “Partial Reconfiguration User Guide UG909" v2016.1, April 2016.

[2] M. Litterick, “Assertion-Based Verification using System Verilog”, *http://www.verilab.com/files/svug_2007_abv_litterick.pdf*.

[3] “IEEE Standard for SystemVerilog - Unified Hardware Design, Specification, and Verification Language”, *http://www.standards.ieee.org/findstds/standard/1800-2012.html*.

[4] A. Sadek, H. Mostafa, and A. Nassar,“On the use of Dynamic Partial Reconfiguration for MultiBand/MultiStandard Software Defined Radio," in *International Conference on Electronics, Circuits, and Systems (ICECS),* 2015, pp. 498-499.

[5] Mentor Graphics, “Questa CDC and Formal Functional Verification,” *www.mentor.com/products/fv/questa-formal-verification-apps*.