Module Based Implementation Of Partial Reconfiguration In FPGA For Counters

Mahendra Dhadwe [1]Arvind Choubey [2]

[1,2]*Electronics & Communication, National Institute of Technology, Jamshedpur, India*

Abstract

Module-based partial reconfiguration of FPGAs play important role, it provides possibility for runtime flexibility. It enables hardware tasks to swap in and out the design without interruption of the entire system. This resultin increase in speed and functionality of FPGA based system. This paper presents flow of partial reconfiguration and implementation of reconfigurable modules using Planahead software on Xilinx virtex-6(XC6VLX240TFF1156-1).Planahead software specifically designs for partial reconfiguration as it has advance floorplanningcapability.This paper reduces power consumption and size by using generated partial bit file of various counters used such as ring counter, up-counter, and decade counter

1. Introduction

Field programmable gate arrays (FPGAs) are quickly becoming the usual targeted technology for many development efforts.FPGAs are programmable logic devices which allow the implementation of digital systems. They provide an array of logic cells that can be configured to perform a given functionality by means of a configuration bit-stream. Many of FPGA systems can only be statically configured. Static reconfiguration means to completely configure the device before system execution. If a new reconfiguration is required, then it is necessary to stop system execution and reconfigure the device it over again. Some FPGAs allow performing partial reconfiguration, where a reduced bitstream reconfigures only a given subset of internal components. Dynamic Partial Reconfiguration (DPR) allows the part of device be modified while the rest of the device (or system) continues to operate and unaffected bythe reprogramming.

In particular, two important benefits can be achieved by exploiting partial dynamic reconfiguration on reconfigurable hardware: (i) the reconfigurable area can be exploited more efficiently with respect to a static design; (ii) some portion of the application must change over time and react to changes in its environment.

In electronics circuit for different type of program require counter so many times. Counter is one of the main building block in various program and affect the timing access and power consumption. So implement partial reconfiguration for various counter sharply reduce thepower consumption and area and timing to perform. The counter such as ring counter, up-counter, and decade counter used to perform partial reconfiguration[1].

2. Partial Reconfiguration

Xilinx has proposed many methods to dynamic partial reconfiguration. There are two main styles of dynamic partial reconfiguration: difference-based and module-based.

A. Difference-Based partial reconfiguration

This method of partial reconfiguration is accomplished by making a small change to a design, and then by generating a bitstream based on only the differences in the two designs. It is especially useful in case of changing Look-Up Table (LUT) equations or dedicated memory blocks content. The partial bitstream contains only information about differences between the current design structure (that resides in the FPGA) and the new content of an FPGA. Switching the configuration of module from one implementation to another is very quick, as the bitstream differences can be extremely smaller than the entire device bitstream[2].

In complex designs, it is difficult to find the component you want to modify. So this methodis not suitable for large-scale complex systems.

B. Module-Based partial reconfiguration

This method is based on modular design flow. This feature allows a team of engineers to work independently on different modules of a design and merge them into one FPGA design. The complete design can be divided into modules and each of these may be independent. If all modules are independent, i.e. no common I/O except clocks then there is no need to use any bus macro for inter-module communication. The bus macro provides a fixed "bus" of inter-design communication. However, for modules that do communicate with each other, a special bus macro allows signals to cross over a partial reconfiguration boundary. The HDL code should ensure that any reconfigurable module signal that is used to communicate with another module does so only by first passing through a bus macro. Without this special consideration, inter-module communication would not be feasible as it is impossible to guarantee routing between modules.

Module-based partial reconfiguration requires performing a set of specific guidelines during the stage of design specification. For each reconfigurable module of the design, a separate bitstream is created. Such a bitstream is used to perform the partial reconfiguration of an FPGA.

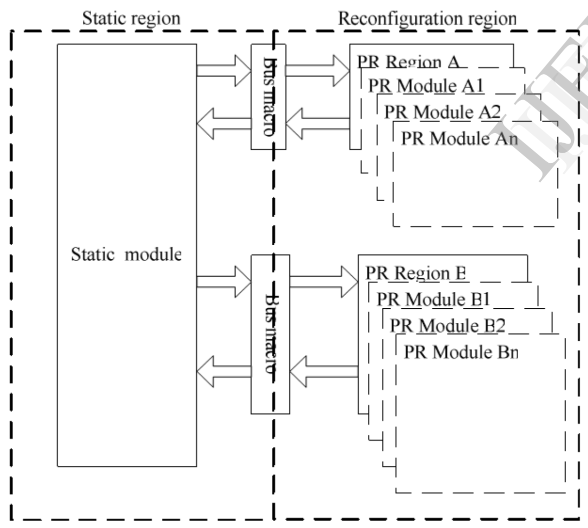


Fig1. partial reconfiguration

A module-based DPR system with two partial reconfiguration regions (i.e. PRR_A and PRR_B) and many partial reconfiguration modules (i.e. PRM_A1, PRM_ A2,…,PRM_An, PRM_B1, PRM_B2,…, PRM_Bn) is show in fig1.

Static module is the design remains in operation during the partial reconfiguration process. Partial reconfiguration module (PRM) is the design module that can be swapped in and out of the device on the fly, multiple PRMs can be defined for a specific region. Partial reconfiguration region (PRR) is the part of the FPGA that is set aside for partial reconfigurable modules. More than one PRR can be set on the chip[3],[4].

3. Implementation Using Planahead [5]

The Xilinx partial reconfiguration design flow is managed by the PlanAhead application included in the Xilinx IDE. This is the tool that allows you to define the physical placement of the static and PR regions on your target FPGA. The netlists generated using synthesis tool ISE (13.2) in the previous sections must be imported into a PlanAhead project and used to implement the design for the targeted FPGA

A. Implementation flow

Step1: start with the HDL description of the design. Synthesize the static part and reconfigurable modules usingxilinx 13.1(ISE) synthesize tool

Step2: placing and routing (PAR) and mapping.

Step 3: creating aplanahead project. (a) Specify synthesized (EDIF or NGC) netlist.(b) Set PR project

Step4: Set the location of the static netlists. (a)specify the top netlist file.(b) specify the UCF file. Step5: select the targeting device i.e. virtex6, familyXC6VLX240T.

B.Floor planning Partial Reconfigurable Partition:

Step1: create netlist design

Step2: set the partition, (a) set the partition is reconfigurable (b) add reconfigurable module as black box without netlist. Step3: assign pblock mode, drawa rectangle on FPGA die.

C. Adding Reconfigurable Instances to the Partial

Reconfiguration Partition:

Step1: Add up counter as reconfigurable module.

Step2: Again ring counter as reconfigurable module.

Step3: Again decade counter as reconfigurable module.

Create Design Instances for Implementation:

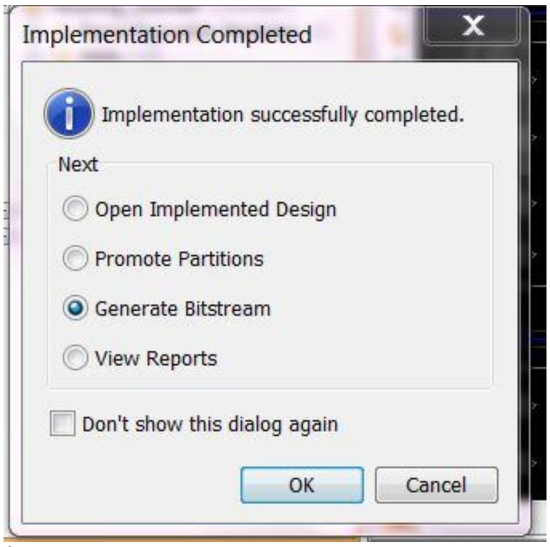
Step1: In the Design Run window, click on Create New Run

D. Implement Designs:

Step1: Right-click „config_1" in the „Design Runs" pane and

select „Make Active". Step2:Right-click „config_1" and select „Launch Runs".

Step3: When the implementation completes, select generate bitstream, shown in fig2.


Implementation Completed

Implementation successfully completed.

Next

Open Implemented Design [✓ Generate Bitstream] Promote Partitions View Reports

Don't show this dialog again

OK Cancel

fig2. Implementation complete

E. Generating Bitstream

Step1: Right-click„config_1"and select „Generate Bitstreams".

4.Result

TABLE I

Bitstream name	Type	size
config_1.bit	Full bitstream	9017KB
config_1_reconfig_counter_counter _partial.bit	Partial bitstream	139 KB
config_1_reconfig_decade_decade _partial.bit	Partial bitstream	116 KB
config_1_reconfig_ring_ring_partia l.bit	Partial bitstream	116 KB

As shown result of bit size in table1,partial reconfiguration utilizes a smaller bitstream than a full bitstream for the FPGA. The size of the bitstream is directly proportional to the number of resources being configured,The direct benefit is less space needed for storing the necessary configurations for operation.As reconfiguration times are highly dependent on the size and organization of the PRRs, an additional benefit is that the reconfiguration time is shorter.

5. Conclusion

In this paper, we have illustrated the clear advantage of module- based partial reconfiguration. The advantages of module based partial reconfiguration are show by implementing various counters.

REFERENCES

[1]Ian Kuon, Russell Tessier, and Jonathan Rose,*FPGA Architecture: Surveyand Challenges,*Boston –Delft ,now Publishers Inc. 2008

[2]Wang Lie,WuFeng-yan,*"Dynamic Partial Reconfiguration on Cognitive Radio Platform"*, 978-1-4244-4738-1/09,IEEE

[3]R.V. Kshirsagar and S. Sharma, "*difference based partial reconfiguration",*IJAET ,ISSN: *2231-1963,*

[4]Xilinx,Inc.XAPP290:*"Two Flows for Partial Reconfiguration: Module Based or Difference Based "* available:http :/ / www. xilinx. Com/

[5]PlanAhead User Guide, (UG632), available: http://www.xilinx.com/support/documentation/sw_manual s/xilinx13_2/PlanAhead_UserGuide.pdf/