**International Journal of Combined Research & Development (IJCRD)**
**eISSN:2321-225X;pISSN:2321-2241 Volume: 2; Issue: 2; February - 2014**

# Detailed Design Flow for Partial Reconfiguration

Kunal Yogeshkumar Parikh
Asst. Professor, Department of ECE
AMC Engineering College
18th K. M, Bannerghatta Road,
Bangalore-560083, Karnataka, India
parikhykunal@gmail.com

## ABSTRACT

With the rapid development of VLSI Technology, FPGA exposed many limitations at the area, speed, power, and chip capacity. Newer Xilinx FPGAs (few Spartan & Virtex series) provides the possibility to be reconfigured by Dynamic Partial Reconfiguration (DPR) technique. DPR is defined as the ability of a single system to get reconfigured to perform multiple applications. The main contribution of this paper is in proposing a complete design flow of the partial reconfiguration technology which analyzed on Xilinx Virtex -5 ML507 Board. It is shown in the experiment that flexibility of the system was improved greatly by the dynamic partial reconfiguration.

## Keywords

FPGA; Dynamic Partial Reconfiguration; Virtex-5 ML507 Board

## 1. INTRODUCTION

FPGAs have come a long way from mere glue-logic applications of interconnecting discrete components to get higher performance from reconfigurable processors. Today, FPGAs are even used in high energy physics experiments like ALICE1 at CERN2 (although they are not very radiation tolerant), due to the possibility to easily do future upgrades of the electronics [1]. Beyond that, newer Xilinx FPGAs (few Spartan & Virtex series) provides the possibility to be reconfigured by Partial Reconfiguration technique.

Reconfigurable systems can be designed using various methods with different ways of getting the system reconfigured. Reconfigurable computing is first classified into two categories which are Full reconfiguration and Partial reconfiguration. Partial Reconfiguration again classified into another two categories which are, Static and Dynamic Partial Reconfiguration. Static partial reconfiguration allows a portion of FPGA to be reconfigured, but during this process the remaining part of FPGA is in shutdown mode and Dynamic partial reconfiguration allows a portion of FPGA to be reconfigured while the remaining part of FPGA is under operation without any loss of data.

In this paper Xilinx Virtex5 ML507 FPGA platform as the hardware platform in the research and design the reconfigurable module and assembled the system functions. Besides, by the use of bus macro technic solves the communication problem of the static module and reconfigurable, and based this way can more effectively control the implementation of the reconfiguration operation [Flow Paper]. During the experiment, we observed the key steps of the partial reconfigurable technique and proposed the detailed design flow of the Same on Xilinx Virtex 5 Family Chip

## 2. RELATED WORK

### 2.1 Reconfigurable System with FPGA

Reconfigurable system is defined as the ability of a single system to get reconfigured to perform multiple applications

#### *2.1.1 FPGA*

In 1985, the first commercially viable field programmable gate array (FPGA) was invented by Xilinx. It had programmable gates and programmable interconnects between gates. The most important difference to CPLDs is that FPGAs are based on lookup tables (LUTs) instead of logic arrays. The LUTs are realized via static random access memory (SRAM) cells because of its volatile nature, the SRAM cells must be loaded with configuration data each time an FPGA powers up. The configuration of the FPGAs could be changed in order to enhance or change the functionality of the chip and even to remove bugs [2]

The CLB is the basic logic unit in a FPGA. Exact numbers and features vary from device to device, but every CLB consists of a configurable switch matrix with 4 or 6 inputs, multiplexer circuitry etc, and flip-flops. The switch matrix is highly flexible and can be configured to handle combinatorial logic, shift registers or RAM. While the CLB provides the logic capability, flexible interconnect routing routes the signals between CLBs and to and from I/Os. Routing comes in several flavors, from that designed to interconnect between CLBs to fast horizontal and vertical long lines spanning the device to global low-skew routing for Clocking and other global signals. The detailed Architecture of FPGA is shown in Figure 1.

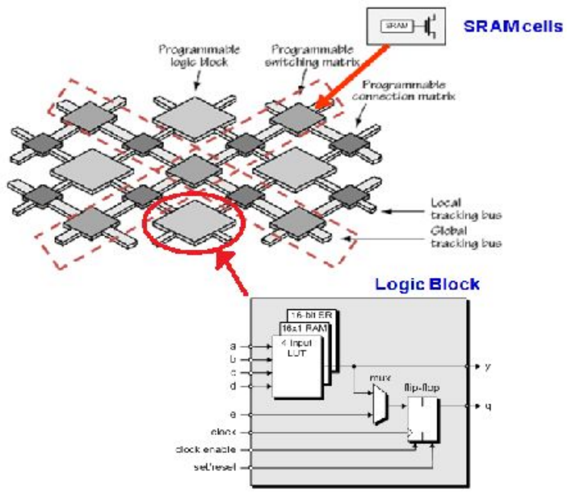

**Figure 1: FPGA Architecture**

The logic and routing elements in an FPGA are controlled by programming points, which may be based on Static Random Access Memory (SRAM), Antifuse, or Flash technology. After a design has been compiled, designer can program the FPGA to perform a specified computation by loading the bitstream into it. The SRAM-based FPGAs can be configured any number of times to provide additional implementation flexibility for further tailoring the implementation to lower cost and make better use of the device. Most current commercial FPGAs use volatile static-RAM (SRAM) bits connected to configuration points to configure the FPGA [3]

### *2.1.2 SRAM Technology*

The most widely used method for storing configuration information in commercially available FPGAs is volatile static RAM (SRAM). This method has been made popular because it provides fast and infinite reconfiguration in a well-known technology.

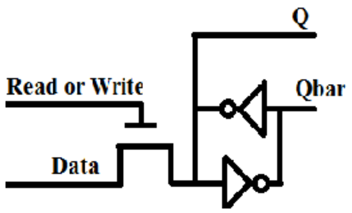

**Figure 2: Programming Bit for SRAM based FPGA**

For reconfigurable computing, SRAM-based FPGAs are used. In these devices, every routing choice and every logic function is controlled by a simple memory bit. With all of its memory bits programmed, by way of a configuration file or bitstream, an FPGA can be configured to implement the user's desired function. Thus, the configuration can be carried out quickly and without permanent fabrication steps, allowing customization at the user's electronics bench, or even in the final end product.

## 2.2 Approaches for Partial Reconfiguration

There are three approaches to design Reconfigurable Systems
1. Reconfiguration using JTAG
2. Reconfiguration using RS232
3. Auto Reconfiguration

### *2.2.1. Reconfiguration using JTAG*

Figure 3 illustrates the external reconfiguration of FPGA using JTAG, wherein it is not mandatory to use a soft core processor like Microblaze. Though PlanAhead is used to allocate same hardware resources for different modules, but whenever there is a need to reconfigure the FPGA, the corresponding bitstream file is loaded through JTAG onto FPGA.

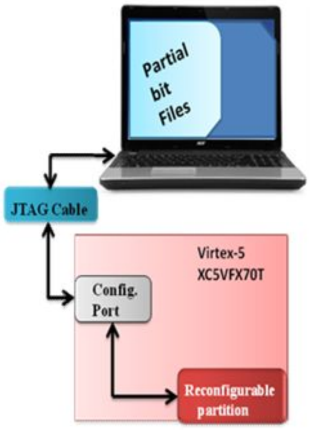

**Figure 3: Reconfiguration using JTAG [4]**

### *2.2.2 Reconfiguration using RS232*

The RS-232, is a legacy, full duplex, wired, Asynchronous Serial Communication Interface. It extends the UART communication signals for external data communication. RS 232 interface defines various handshaking and control signals for communication apart from transmit and receive signal lines for data communication.

Figure 4 illustrates the reconfiguration of FPGA using external RS232 interface, which can be either from another microcontroller or desktop PC. Commands for reconfiguring the FPGA to perform different functionalities are passed through RS232 interface. For this case, Microblaze soft core processor is embedded into the FPGA to perform the operations of reconfiguration based on inputs received through RS232 port and hardware Internal Configuration Access Port (ICAP) provides access to load the partial bitstream files from compact flash onto FPGA.

### *2.2.3. Auto Reconfiguration*

Figure 5 illustrates the most commonly sought form of reconfiguration, self or auto reconfiguration of FPGA. In this case, the FPGA reconfigures itself to perform functionality, either after predefined time duration or based on a status flag. Microblaze soft processor and ICAP interface are used for this case too, but the use of UART interface is not mandatory.

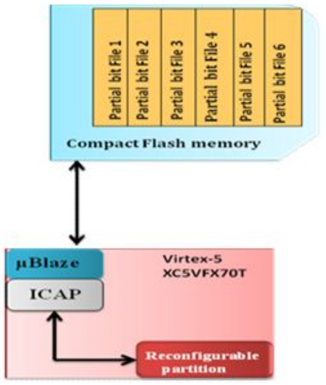

**Figure 5: Auto Reconfiguration [4]**

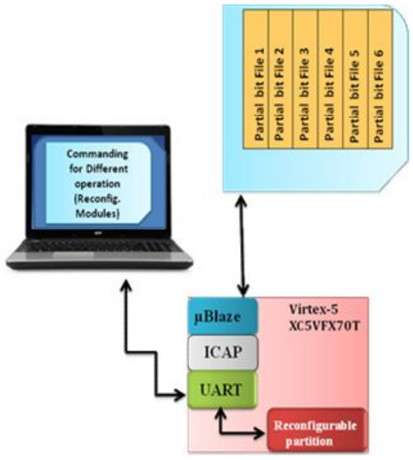

**Figure 4: Reconfiguration using RS232 [4]**

.

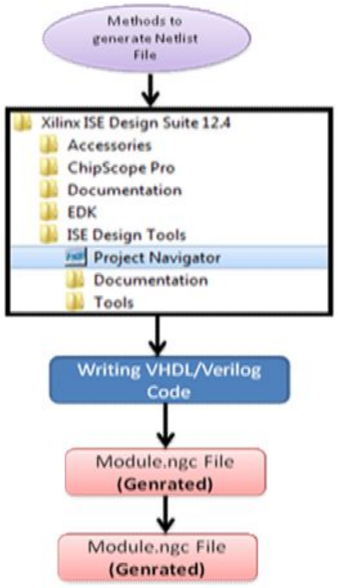
Methods to generate Netlist File
Xilinx ISE Design Suite 12.4
Accessories
ChipScope Pro
Documentation
EDK
ISE Design Tools
Project Navigator
Documentation
Tools
Writing VHDL/Verilog Code
Module.ngc File (Genrated)
Module.ngc File (Genrated)

**Figure 6: Method to generate Netlist File form Xilinx Software**

## 3. DYNAMIC PARTIAL RECONFIGURATION

## 3.1 PR Design Flow

### *3.1.1 Top Level Design Flow*

The top level design flow for Partial Reconfiguration is shown in Figure 11 and the steps are as follows

1. Create a processor system using Xilinx Platform Studio
2. Generate .elf from Software Design Kit
3. Add a processor system in ISE Create PR project and create various configurations for partial reconfigurations in PlanAhead
4. Generate full and partial reconfiguration bitstreams and system.ace files which can be stored on a Compact Flash memory

5. Configure an FPGA using a Compact Flash memory card and run user application

### 3.1.2 Detailed Design Flow

The detailed design flow for Partial Reconfiguration is shown in Figure 7 and the steps are as follow

1. First generate netlist (.ngc) file for all dynamic modules as well static module

There are two Methods to generate Netlist, which are

A). The method to generate .ngc (Netlist) file form the project navigator by writing VHDL/Verilog Code is shown in below Figure 6

The Project Navigator is also required to generate top level netlist file and user constraint file where it combines dynamic as well as static along with system components. This top level netlist is used by PlanAhead.

B). The process to generate netlist file from MATLAB is shown in below Figure 7. Here, The System Generator is a DSP design tool from Xilinx that enables the use of the MathWorks model-based Simulink design environment for FPGA design. System Generator provides a system integration platform for the design of DSP FPGAs that allows the RTL, Simulink, MATLAB and C/C++ components of a DSP system to come together in a single simulation and implementation environment. It supports a black box block that allows RTL to be imported into Simulink and co-simulated with either ModelSim or Xilinx® ISE® Simulator. System Generator also supports the inclusion of a MicroBlaze embedded processor running C/C++ programs [4]. The Module is made for particular application in MATLAB where system generator will be used to generate direct netlist file from the MATLAB. Designs are captured in the DSP friendly Simulink modeling environment using a Xilinx specific blockset. The Xilinx blockset is a library of arithmetic, logic and DSP functions under Simulink, useful for design simulation and verification [5].

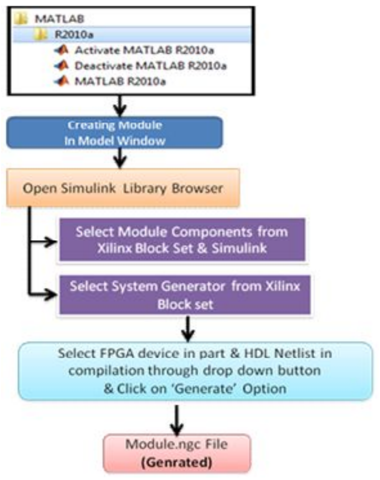
MATLAB
R2010a
Activate MATLAB R2010a
Deactivate MATLAB R2010a
MATLAB R2010a
Creating Module In Model Window
Open Simulink Library Browser
Select Module Components from Xilinx Block Set & Simulink
Select System Generator from Xilinx Block set
Select FPGA device in part & HDL Netlist in compilation through drop down button & Click on 'Generate' Option
Module.ngc File (Genrated)

**Figure 7: Method to generate Netlist File form MATLAB Software**

2. Create a processor system using Xilinx Platform Studio to generate system components

Xilinx Platform Studio (XPS) software is used to build, connect and configure embedded processor based FPGA system. The Microblaze Processor System Block Diagram is shown in below Figure 8. The PLBv46 bus and the MicroBlaze processor run at a frequency of 100 MHz and the DDR2 runs at 200 MHz in this system [6]

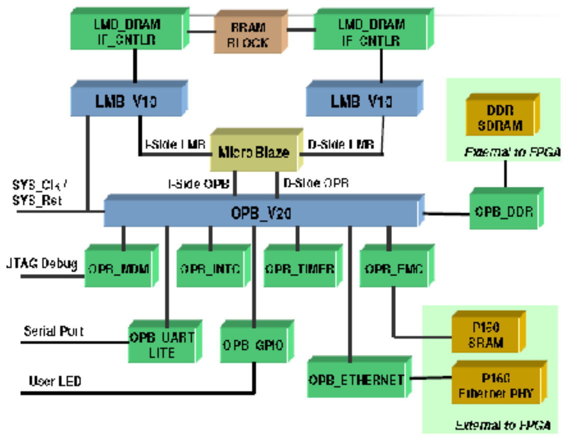

**Figure 8: MicroBlaze Processor System Block Diagram**

The system.xmp file will be created here and is used in Project navigator for top module. This will generate system_stub.bmm as well as system_stub_bd.bmm that are used in planahead and Bash Shell respectively to link with processor. It is used to generate .ngc file for top module. The complete XPS function is shown in Figure 9.

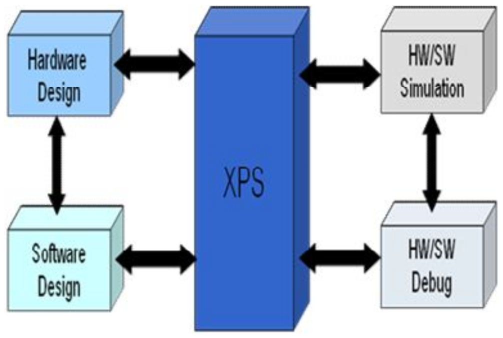

**Figure 9: XPS function [7]**

3. Open SDK from XPS to generate executable file

Xilinx Software Development Kit (SDK) is used to develop C/C++ Embedded software application where CF2ICAP command is used to fetch bit files from compact flash memory and load into FPGA using ICAP. SDK imports and compiles the provided source files and generates executable (.elf) File which will be used for generating system.ace file [7]

4. Add a processor system in ISE along with top VHDL coding and generate netlist (.ngc) file for this

- VHDL coding contains static, dynamic and System (Processor) components

5. Create a PlanAhead PR project

PlanAhead software is used to floorplan the design. It required netlist files of the dynamic, static and top level modules. It manages the modules and categorizes them into reconfigurable modules. Full and Partial bitstream files are generated using PlanAhead after verification of all reconfiguration modules. Figure 10 shows the FPGA Design planner of the reconfigurable system where reconfigurable block has not been selected. User can select only single or ‘n’ number of reconfigurable blocks depend upon the requirements.

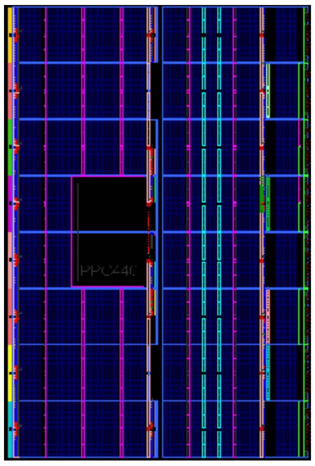

**Figure 10: FPGA Floorplan for reconfigurable systems**

6. Use bash shell

- To generate download.bit file
- To generate system.ace file, using generated download.bit file

7. Configure an FPGA using a Compact Flash memory card and run user application

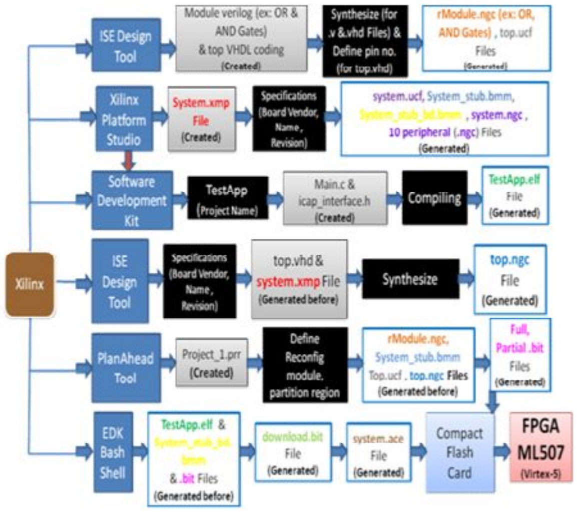
Xilinx
ISE Design Tool
Module verilog (ex: OR & AND Gates) & top VHDL coding (Created)
Synthesize (for .v &.vhd Files) & Define pin no. (for top.vhd)
rModule.ngc (ex: OR, AND Gates) , top.ucf Files (Generated)
Xilinx Platform Studio
System.xmp File (Created)
Specifications (Board Vendor, Name, Revision)
system.ucf, System_stub.bmm, System_stub_bd.bmm , system.ngc , 10 peripheral (.ngc) Files (Generated)
Software Development Kit
TestApp (Project Name)
Main.c & icap_interface.h (Created)
Compiling
TestApp.elf File (Generated)
ISE Design Tool
Specifications (Board Vendor, Name, Revision)
top.vhd & system.xmp File (Generated before)
Synthesize
top.ngc File (Generated)
PlanAhead Tool
Project_1.prr (Created)
Define Reconfig module. partition region
rModule.ngc, System_stub.bmm Top.ucf . top.ngc Files (Generated before)
Full, Partial .bit Files (Generated)
EDK Bash Shell
TestApp.elf & System_stub_bd.bmm & .bit Files (Generated before)
download.bit File (Generated)
system.ace File (Generated)
Compact Flash Card
FPGA ML507 (Virtex-5)

**Figure 11: Detailed PR Design Flow**

## 4. RESULT

The experimental hardware platform is Virtex5 XC5VFX70T FPGA ML507 development kit from Xilinx and the software development environment is Xilinx Design Suite 12.4. When the application calls from hyperlink for the hardware function, the system first checks the corresponding IP module whether configured in the FPGA, if the configuration exists and the FPGA will run it directly; otherwise the system read corresponding part of the configuration bitstream file from compact Flash memory/card to the on chip buffer at first, and then generate the hardware function after the reconfiguration by the Internal Configuration Access Port. One of the goals of a run-time configurable system is to encapsulate the reconfigurable system into a portable application which satisfies by DPR.

## 5. CONCLUSION

In the process of reconfigurable system development, drawing on the module design techniques, we have discussed in detail system hardware realization by three different approaches. The communication between reconfiguration and static module is realized by slice based on bus macro, it is better than the traditional TBUF-based bus macro. By Dynamic Partial Reconfiguration method, system reaches number of advantages which are, Application Portability. Hardware Reuse and reduces SWaP (Size-Weight-Power)

## 6. REFERENCES

[1] Norbert Abel, Sebastia, Manz, Frederik Grüll, “Increasing Design Changeability Using Dynamic Partial Reconfiguration”, *IEEE transactions on nuclear science*, VOL. 57, NO. 2, APRIL 2010.

[2] Wang Lie, Wu Feng-yan, “Dynamic partial reconfiguration in FPGAs”, *Third International Symposium on Intelligent Information Technology Application IEEE* 2009, pp 445-448.

[3] Scott Hauck, Andre Dehon, “Reconfigurable Computing”, *Morgan Kaufmann Publishers*, 2008, pp 16-20.

[4] Kunal Yogeshkumar Parikh, et al., “Design and Evalution of N module Reconfigurable System”, Springer 2014, Volume 243, pp-259-266

[5] Xilinx, Inc. UG639: System Generator for DSP User Guide, Version 14.3, October 16, 2012.

[6] Xilinx, Inc. EDK Microblaze Tutorial, Version 2.0, 9/2003.

[7] EDK overview, Xilinx Slides Presentation, August 2004

[8] Xilinx, Inc. UG744: Partial Reconfiguration of a processor peripheral tutorial user guide, Version 14.1, April 24, 2012.

[9] Xie D., “A Design Flow for FPGA Partial Dynamic Reconfiguration”, *Second International Conference on Instrumentation & Measurement*, Computer, Communication and Control, *IEEE* 2012, pp 119-123.

[10] Xilinx, Inc. UG347: ML505/ML506/ML507 Evaluation Platform User Guide, Version 3.1.2, May 2011.