AUTOMATED GENERATION OF PARTIALLY RECONFIGURABLE MODULES AND THEIR COMMUNICATION INFRASTRUCTURE

Plamen Shterev

Department of Computer Science 12, University of Erlangen-Nuremberg, Erlangen, Germany,
E-mail: plamensht@abv.bg

This paper presents SlotComposer, a software tool for automated synthesis flow and communication infrastructure generation for partially reconfigurable hardware modules. SlotComposer inserts slice-based bus macros between adjacent modules or if specified connects partially reconfigurable modules to the Reconfigurable Multiple Bus (RMB) communication infrastructure. At the same time it optimises the usage and placement of bus macros and instantiates all intermediate communication signals and generates all the necessary constraint files to synthesise the project using Xilinx Partial Reconfiguration Flow tool.

Keywords: FPGA, partial reconfiguration, bus macros, inter-module communication

1. INTRODUCTION

Advances in FPGA technologies have enabled the implementation of reconfigurable computers (RC) based on partially reconfigurable FPGAs. The promise is to meet the computational demand without the need of extra logic gates. However, a few hardware application developers want to use the partial reconfiguration feature due to substantial difficulties currently involved.

There are three main dilemmas commonly faced while developing partially reconfigurable modules for a reconfigurable computing platform:

1) The lack of user-guided tool flow automation for the PR Flow makes it very hard to compile even a simple user application.
2) The lack of an inter-module communication support for non-adjacent reconfigurable modules makes it very hard to make use of an automated tool flow and requires the developer to invest time into the design of a communication infrastructure. Moreover, the instantiation of hundreds of signals between two partially reconfigurable modules by hand is error prone and time consuming.
3) The lack of optimal resource utilization limits the number of possible inter-module signals because the number of connections is limited by the type of the bus macros used and the height of the FPGA. Also, the optimal use of the area group constraints together with the physical placement of bus macros can be automated. The need for high level language as well as methodology and tool support is an unresolved issue that has drawn numerous research interest [2], [4].

The primary goal of this work is to give the FPGA application designer a user-guided tool for the automation of repetitive tasks and the execution of the PR Flow. It is defined user-guided as the ability of the developer to modify the work done automatically at any stage of the design flow.

2. INTER-MODULE COMMUNICATION

One of the central limiting factors for the wide usage of partial dynamic reconfiguration is the problem of inter-module communication. There are three most frequently used ways for communication among different modules. The first one is a direct communication using bus-macros between adjacently placed modules (Fig. 1); providing fixed communication channels that help to keep the signal integrity upon reconfiguration. A standard bus macro can pass eight signals. Assuming the ideal case, when all the signals are either in one and the same direction or their number is multiple of eight, then for ***n*** signals we will need ***n/8*** bus macro modules. The second way is shared memory communication using SRAMs or BlockRAMs (Fig. 2). This is particularly useful in applications in which each module must process a large amount of data, which is then sent to the next module. Each SRAM bank can be accessed by the module placed below as well as by the direct neighbours placed to the right and left. A controller is used to manage the SRAM access. Depending on the application, the user may set the priority of accessing the SRAM for these three modules.

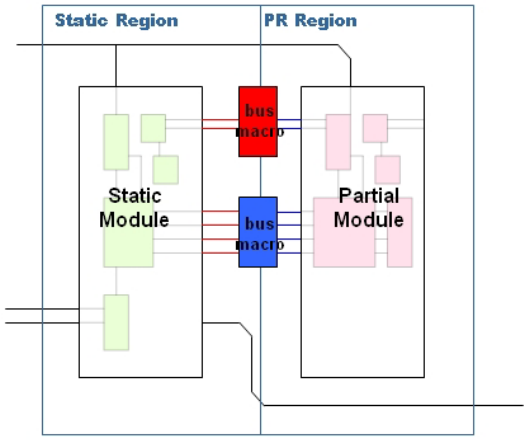

Fig.1. Direct communication using bus macros

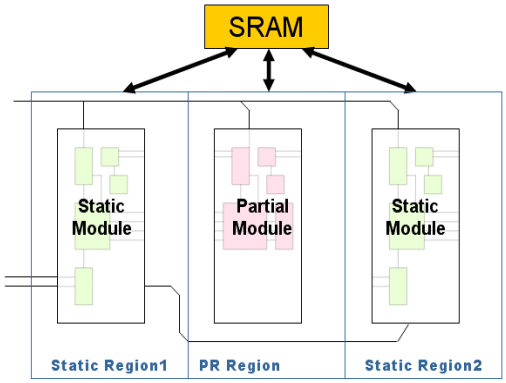

Fig.2. Module communication via SRAM

The disadvantage is that only adjacent modules can use these two communication modes. For non-adjacent modules it is recommended to use a dynamic signal switching communication architecture called Reconfigurable Multiple Bus (RMB) (Fig.3.) [3], [7], [8]. This is a one dimensional structure which connects each of set of *N* modules with one of its *N* processing elements called cross point. Between two cross points several horizontal bus line segments exists. Each cross point consists of a set of switches which are used to connect two bus line segments with each other.

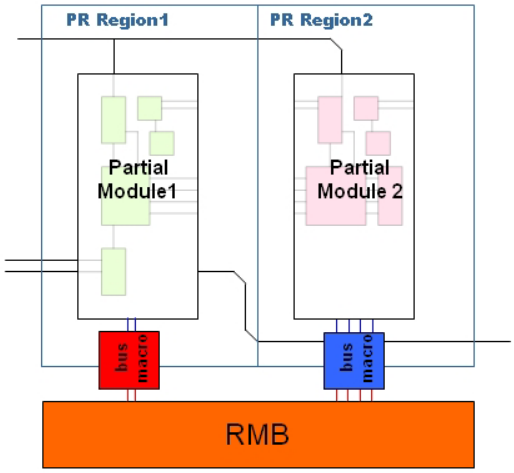

Fig.3. Reconfigurable Multiple Bus communication

3. BUS MACRO COMMUNICATION

To facilitate communication across reconfigurable module boundaries, yet still conform to the Partial Reconfiguration requirement that routing resources across such boundaries to be completely fixed and static, the use of a special bus macro is required. To support communication between two modules A and B, a special bus macro has to be used. Partial Reconfiguration requires the signals used as communication paths between or through reconfigurable modules must use fixed routing resources. That is, the routing resources used for such inter-module signals must not change when a module is reconfigured. Some design flows have realized this using tree-state buffers that are hard wired into the FPGA architecture. This practice is nowadays discouraged because they have fixed locations and are dispersed across the architecture. Practical design experience highlighted the need for greater concentrations of connections between static and dynamic regions and more flexibility in the places where they have to be located. A better solution is pre-routed slice-based bus macros to be used. They are implemented as relationally placed macros. Each of them consists of two configurable logic blocks (CLBs), one on either side of the module boundary between the area groups of module A and module B (Fig. 4).

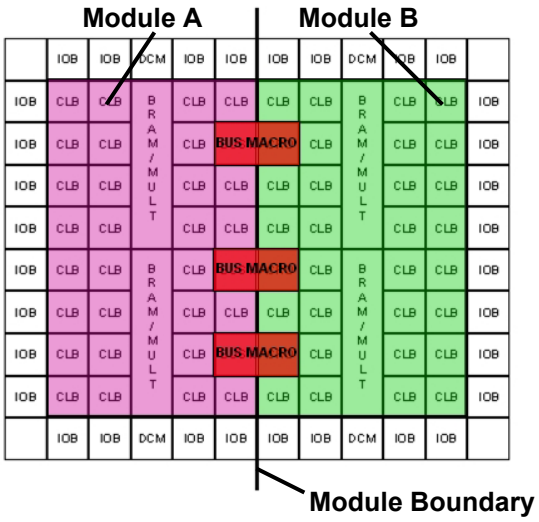

Fig.4. Slice-based bus macros

4. SLOTCOMPOSER FUNCTIONALITY

SlotComposer is multifunctional designer software. It serves three main functions during the FPGA design process and these are: (Fig. 5):

- insertion of bus macro communication between the reconfigurable modules
- creation of synthesis scripts with all the project configuration settings to be synthesised at any later time using XST – the command line mode of Xilinx software environment
- creation of partial flow reconfiguration script which can be run in command line mode at any later time

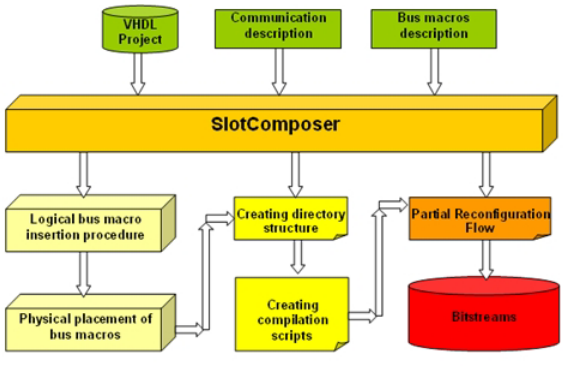

Fig.5. Design flow with SlotComposer

In the following sections the key features of the tool are presented.

A. High-Level Design Flow

To use the PR Flow from Xilinx, the design has to follow a specific design file structure. There has to be a top-level design file, which instantiates the static and the partially reconfigurable modules (PRM), global logic, I/O-Ports and clock primitives. In this design file the communication infrastructure has also to be instantiated, such as bus macros or RMB cross points. The behaviour of the instantiated modules is described in separate High Definition Language (HDL) files. The SlotComposer eases the design process by an automated instantiation of the communication structures in the top HDL project file. It also updates the User Constraint File (UCF) for the top-level HDL file, in which the module boundaries and the location of the bus macros are described. After that it generates a batch-file which calls the required steps needed to generate the full and partial bitstreams using the PR Flow from Xilinx

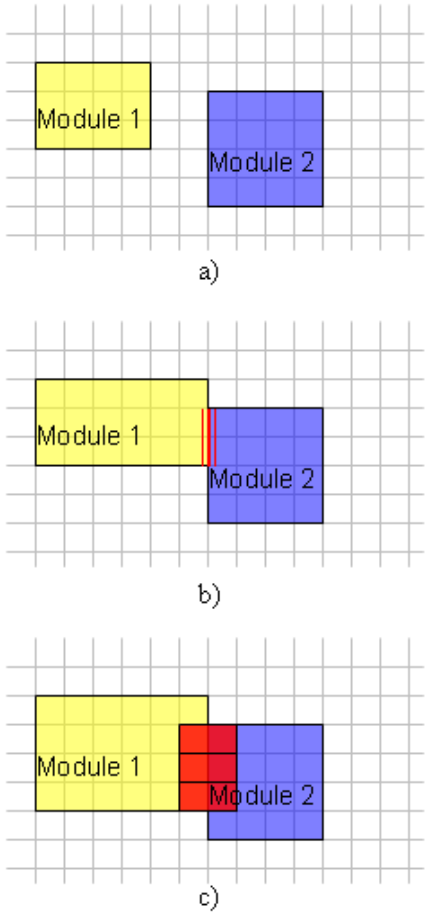

Fig.6. Algorithm for module extension and bus macro insertion
a. Check whether the modules have common boundary,
b. Extend the static module boundaries and check whether the boundary is wide enough to place all the bus macros. If not big enough extend again the boundary
c. Place bus macros.

B. Inter-Module Communication

For communication between adjacent partially reconfigurable modules (PRM) or between a PRM and an adjacent static module the top HDL file as well as UCF-file is modified, by instantiating and connecting bus macros at the boundaries of the PRM. When there are many signals crossing the reconfigurable module boundaries, many bus macros and their corresponding signal connections are required, which is a very time consuming and an error prone task, if done by hand.

C. Reconfigurable Multiple Bus Multi-Hop Communication

If a Multi-Hop Communication infrastructure is used in the project, the SlotComposer modifies the top-level HDL file and inserts bus macro communication on the cross points between the RMB and the partially reconfigurable regions. After that it inserts the location constraints of the new bus macro instances in the UCF file of the project.

D. Optimal Resource Utilization

To optimal use the given resources the SlotComposer packs as many signals as possible into a single bus macro in order use a minimal number of communication instances. Bus macros can be placed only between adjacent modules, because half of the bus macro has to be in one module while the other half has to be in the adjacent module (Fig.4). If a bus macro communication is required between non neighbouring

modules the tool checks if the space between them is free, by reading the UCF-file given from the user and if yes, it extends the boundaries of the communicating modules in such a way, that they are adjacent to each other. Also if the boundary is not big enough to insert all required bus macros SlotComposer extends the boundary and modifies the UCF-file correspondingly so that it can place all the necessary interface instances (Fig.6).

5. IMPLEMENTATION

SlotComposer is designed to be a multifunctional and adaptive tool. It is created to perform project synthesis and a partial flow design but all the settings are not hard coded in the software. Its design flow logic and sequence can be edited at any time by the user. This feature makes the tool adaptive for a project specific synthesis options and as well as flexible for the future changes in the design flows.

The whole flow configuration is described parametrically in an external file. Short set of script constants created for a flexible description of a design flow. SlotComposer supports different reconfiguration scripts for top, static and dynamic modules and some additional commands at the beginning and at the end of the flow like merging all the output files into one directory.

6. GRAPHICAL USER INTERFACE

SlotComposer is designed to ease designer's work and as such a tool it has a comfortable graphical interface (Fig. 7)

Using SlotComposer control panel each of the design flow steps – inserting bus macros, creating synthesis scripts and PR Flow scripts (see Fig. 5) – can be run separately. To start each process step correctly it is necessary to set all the required project data – all VHDL files with project modules, the top level design file, compiled files with bus macro entities, project user constraint file – have to be in one and the same directory.

All the settings can be done from the menus of the graphical interface. At the end of each design step SlotComposer shows a window with modules' and bus macro placement in the chosen FPGA.

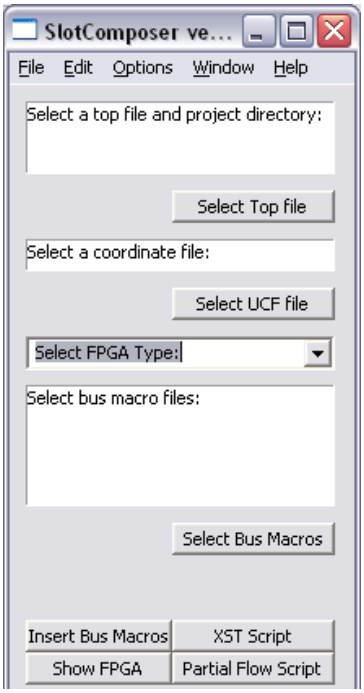

Fig.7. SlotComposer Control Panel

7. EXPERIENCE USING SLOTCOMPOSER

SlotComposer is part of the Erlangen Slot Machine (ESM) project. ESM is a flexible FPGA-based reconfigurable platform that supports re-locatable hardware modules arranged in slots. It is developed by the department of Hardware-Software-CoDesign, University of Erlangen-Nuremberg. SlotComposer is successfully used to creating communication infrastructure for VHDL applications running on the ESM. It supports different types of bus macro communication – with horizontal, vertical as

well as combination of horizontal and vertical interface modules. SlotComposer was also successfully presented on the Tutorial of ARCS 2007 Conference (12.03 – 15.03.2007, Zurich)

8. CONCLUSION

In this work a user friendly tool called SlotComposer is designed and developed which accelerates the design process of VHDL projects. It is a multifunctional tool, which can analyse VHDL project, insert communication interfaces automatically in the correct places, create a specific directory structure for the synthesis, where each dynamic module is compiled in its own directory, and write synthesis as well as partial flow scripts. The communication type which is supported by SlotComposer and can be implemented in a VHDL project is direct module communication using bus macros between the reconfigurable modules and the static part of the design as well as RMB communication via bus macros. SlotComposer has free space optimization advantage – it can route several signal through a single bus macro, providing that their direction is one and the same thus saving a lot of space in the FPGA memory. The third and also very important feature of SlotComposer is its ability to create a batch script for partial reconfiguration flow. All flow sequences and command line options are not hard coded, they are written in an external text file in a parametric way with separate flow commands for dynamic and static instances. The complete flow is generated by the tool and all variables are replaced by their current project. This parametric way of design flow input makes SlotComposer adaptive for future changes in the flow sequence and logic. A Graphical User Interface is also created for SlotComposer to make it possible for the designer comfortably to see and edit the project settings before synthesis.

9. REFERENCES

[1] Xilinx, Inc. http://www.xilinx.com.

[2] Habibi S. T. A., Design and verification of systemc transactionlevel models. IEEE Trans. VLSI Syst., 14(1):57–68, January 2006.

[3] ElGindy H. A., A. K. Somani, H. Schröder, H. Schmeck, and Spray. RMB - a reconfigurable multiple bus network. In Proceedings of the Second International Symposium on High- Performance Computer Architecture (HPCA-2), pages 108–117, San Jose, California, Feb. 1996.

[4] Rapid C. C., Design and analysis of communication systems using the bee hardware emulation environment. IEEE Rapid System Prototyping Workshop, (6), 2003.

[5] Lysaght P., B. Blodge, J. Mason, J. Young, and B. Bridgeford. Enhanced architectures, design methodologies and cad tools for dynamic reconfiguration of xilinx fpgas. FPL 2006, pages 12–17, August 2006.

[6] Majer M., J. Teich. ESM - the Erlangen Slot Machine. http://www.r-space.de.

[7] Page I., Closing the gap between hardware and software: hardware-software cosynthesis at oxford. IEE Colloquium on Hardware-Software Cosynthesis for Reconfigurable Systems, 1996.

[8] Vaidyanathan R., J. L. Trahan. Dynamic Reconfiguration: Architectures and Algorithms. IEEE Computer Society, 2003.