Dynamic Reconfiguration: Core Relocation via Partial Bitstreams Filtering with Minimal Overhead

Fabrizio Ferrandi
DEI - Politecnico di Milano
e-mail: `ferrandi@elet.polimi.it`

Massimo Morandi
DEI - Politecnico di Milano
e-mail: `massimo.morandi@microlab-mi.net`

Marco Novati
DEI - Politecnico di Milano
e-mail: `marco.novati@microlab-mi.net`

Marco D. Santambrogio
DEI - Politecnico di Milano
e-mail: `marco.santambrogio@polimi.it`

Donatella Sciuto
DEI - Politecnico di Milano
e-mail: `sciuto@elet.polimi.it`

ABSTRACT

Partial reconfiguration is a relatively new feature of FPGAs and it allows the modification of hardware functionalities at runtime, providing the possibility for great improvements in the concept of reconfigurable computing. However, this new approach also creates some problems in the implementation phase of modules and in their placement. By restricting the form of single functions to arrays of whole columns communicating on a single horizontal bus, the problem can be significantly simplified. Column-wise partial reconfiguration can be realized by means of a space allocation manager that determines the columns where single modules should be placed and a component which modifies the bitstream to place it in the correct position. This paper describes the development of such component in the form of the Bitstream Relocation Filter, BiRF, that allows the relocation of a partial bitstream with minimal overhead during the download process. The proposed solution is thought starting from the REPLICA filter, trying to adapt it to work within the fixed side of the Caronte architecture.

1. INTRODUCTION

One of the most important problems in current computing is the continuous search for the best compromise between flexibility and performance: generally every step in the direction of versatility tends to be a step back in terms of efficiency and vice-versa. A possible solution to this problem is offered by Field Programmable Gate Arrays, from now on FPGAs. FPGAs allow the creation of reconfigurable systems [1], which can be considered a good compromise between special purpose hardware - like Application Specific Integrated Circuits (ASICs) - and general purpose computers. This paper presents a space allocation manager that determines the columns where single modules should be placed on the physical device. This component is used to modifie at runtime the partial reconfiguration bitstream to place it in the correct position.

This paper is organized as follow. Section 2 describes the concept of FPGA reconfiguraton with a focus on its dynamic version [1, 2] along with the problems it creates in the design of a system and the solutions developed to solve them, again focusing on a specific approach: bitstream relocation. All the work presented in this paper is based on the Xilinx Virtex series FPGAs [3] whose features are described in section 3. This section explains the structure and use of Xilinx Virtex FPGAs, configuration bitstreams and configuration registers. The Bitstream Relocation Filter is inspired on the REPLICA filter, as proposed in [4], whose basic assumptions and ideas are described in Section 2. Section 4 presents the development and implementation process of the BiRF filter, with a description of the whole system and an explanation of all its components. Finally in Section 6 the synthesis results of the filter are shown with possible extensions and improvements of the work in the future.

2. STATE OF THE ART

The main feature of a reconfigurable system is the possibility to change repeatedly its configuration; on these systems, functions can be implemented as independent hardware modules which can be downloaded on the chip only when needed, replacing other modules. This approach has two main advantages: in terms of versatility, a reconfigurable system is more flexible than an Application Specific Integrated Circuit, while in terms of performance it generally offers better latency and smaller area requirements than a general purpose computer. Unfortunately, reconfigurable systems also have a major drawback in the high time overhead caused by the reconfiguration process itself; the creation and download of a whole configuration bitstream on the FPGA is a time consuming operation, and doing it repeatedly causes unacceptable waste of computing time. A possible solution to such a problem is bitstream compression: reducing the amount of data before the download helps minimizing the time overhead but, with this method, the board still has to be wholly reconfigured every time a new function is needed, which is the main drawback.

A significant improvement can be obtained introducing the partial reconfiguration [2], a feature provided by systems like the FPGAs of the Xilinx Virtex Series [5], which represents a great step forward in the field of reconfigurable systems; partial reconfiguration means that the FPGA does not need to be always completely rewritten but can be modified only when and where needed and, most of all, while the rest of the system keeps working. This is a particularly powerful approach because, usually only a small portion of the total FPGA has to be changed, possibly a single function, and reconfiguration time is directly tied to the size of the reconfiguration bitstream. Partial reconfiguration presents also a second problem in the pre-synthesis phase: if the configuration has to change dynamically, the actual position of modules cannot be known during the implementation process; however, each pre-synthesized module bitstream can only be implemented in a fixed location.

The bitstream relocation problem can be solved in two ways: by restricting the possible positions where a module can be placed, that implies the fact that all the allowed bitstreams for every function have to be created in advance, or by creating the bitstreams for each different module in a fixed arbitrary position and relocating them afterwards. Relocation is the approach which will be further discussed in this paper. Two existing solutions based on the said approach are the PARBIT tool [6] and the REPLICA filter [4]. The former is a software solution which can relocate any part of a complete bitstream in an arbitrary position, a powerful tool that, unfortunately, suffers from excessive reconfiguration overhead due to its nature; it has to be run on an external computer, so the bitstream must be relocated on the computer and only after the relocation process it can be downloaded on board. The filter proposed in [4], instead, is an hardware solution, created with the goal of limiting and, possibly eliminating, the computational overhead by performing relocation during the normal download process; this approach eliminates the need for an external computer to perform relocation by means of a filter implemented on a board interposed between the computer which manages reconfiguration and the target FPGA. The REPLICA solution offers great improvements in terms of time overhead, and it basically eliminates them by taking advantage of the inevitable latency, caused by data transfer, and using that time to perform its manipulations on the bitstream word by word as they get transferred on the board.

3. XILINX VIRTEX ARCHITECTURE

Virtex FPGAs [3] are composed of several parts: Configurable Logic Blocks (CLBs), Input Output Blocks (IOBs), Ram Blocks and a set of configurable resources for interconnection between different blocks. Virtex devices can be configured through SelectMAP interface, master/slave serial interface and Boundary-Scan interface. SelectMAP is a 8-bit interface which allows to write eight bits of configuration bitstream per clock cycle; Virtex devices can be configured to retain the SelectMAP pins, allowing further re-configuration via those pins or, if further re-configuration is not required, those pins can be configured as user I/O. With master/slave serial or Boundary Scan interface, only one bit of configuration bitstream per clock cycle can be written. Configuration data can be read using the SelectMAP interface or the Boundary Scan interface, while the master/slave serial interface can be only used to write.

3.1 Memory Configuration

The Virtex configuration memory is a rectangular array of bits, grouped into vertical frames that are one-bit wide and extend from the top of the array to the bottom. Frames are grouped together into larger units called columns. Each device contains one central column that configures the four global clock pins. Two IOB columns represent configuration for all of the IOBs on the left and right edges of the device. The main part of the device is made of CLB columns which contain CLBs in the middle and an IOB on each of the edges. The remaining column type is used for the RAM Block and it is divided in two parts; the first used for the content and the second one used for the interconnections.

3.2 The Structure of a Configuration Bitstream

A bitstream is a binary file that contains all the device data configuration, which is uploaded through one of the available interfaces. A bitstream is composed of a series of configuration commands and configuration data, organized in 32-bit words. The main commands which can be found in a bitstream are read and write commands that are executed when found in, or stored into, a configuration register. A command is organized as a packet with a header word followed by options and/or data words. An Header word contains: a type field, an operator field, a register-address field and a word-count field, as described in details in [3].

3.3 Configuration Registers

Configuration logic is manipulated via a series of configuration registers, used to store each command or data word. Two of these configuration registers contain commands which are involved in the bitstream relocation process: the FAR and the CRC. The FAR register contains the address of the current frame in the FPGA which is composed of three parts: block type, major address and minor address. Block type discriminates between CLB and RAM, major address identifies the column in the addressing space, minor address indicates a specific frame in the column. The integrity of configuration data is granted by Cyclic Redundancy Checks (CRC) that must be executed before the device initialization. When a data is written to any configuration register except LOUT, a CRC value is computed using the register data and address bits. CRC check is done after a writing in the CRC register, comparing the computed value with the one that is stored in the register. If the two values are different, the device is put in an error state.

4. THE BIRF HW CORE

The work further discussed in this paper could be compared with the REPLICA filter, [4]; both these works are based on the same assumptions but the BiRF implementation is able to guarantee better performance both considering the occupation requirements on the interposed FPGA, and the throughput gained by its hardware implementation introduced into the fixed part of the Caronte architecture, [1, 2]. In this section, the basic concepts and assumptions of our approach are discussed, and they are followed, in Section 5, by a brief description of the implementation. Partial reconfiguration, while providing great improvements in terms of reconfiguration time, also causes some problems in reconfiguration management; the main of these problems is tied to the placement of functions [7] on the FPGA. Choosing an optimal - or at least good - position where to place a module is not a simple task, considering that a module can normally have arbitrary size, form and communication lines; therefore some limitations have to be introduced in the pre-synthesis phase to reduce the complexity of the problem.

4.1 Column-wise Approach

A possible solution is provided by the Xilinx Virtex series FPGAs [3], we have chosen to use this device in order to be able to plug the proposed filter into a real reconfigurable architecture such the one proposed in [1]. Those FPGAs allow reconfiguration of single columns, which means every module has to be extended on the whole board vertically, but can be arbitrarily wide horizontally; a compromise that allows to manage the reconfiguration in a one-dimensional way in opposition to the two-dimensional - and therefore more complex - generic approach [8]. With this limitation, also the problems tied to the interaction between distinct modules can be solved easily with the definition of a fixed horizontal communication infrastructure that spans the whole FPGA; in this way, communication of every module can be guaranteed even during reconfiguration.

As previously described, partial reconfiguration could imply that the position of modules is not known during the implementation process and thus the final bitstreams cannot be created in advance; each pre-synthetized module bitstream can only be implemented in a single array of columns, but there is no guarantee that the module will be needed in that specific position, and, therefore, some column-wise bitstream remapping has to be done.

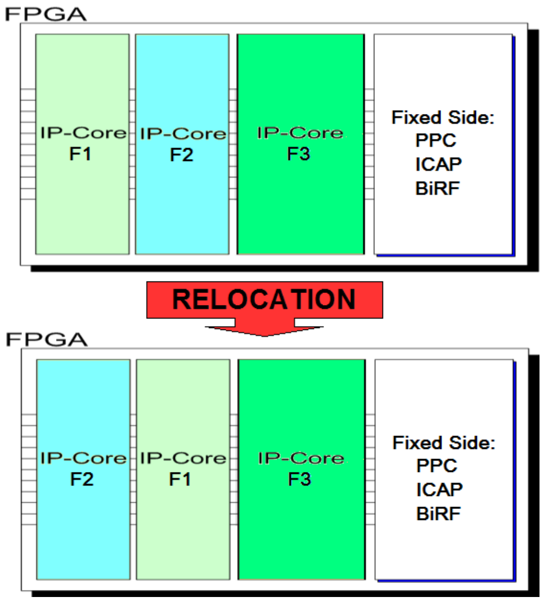

Figure 1: BiRF relocation example

5. BIRF IMPLEMENTATION

BiRF (Bitstream Relocation Filter) allows the relocation of a partial bitstream with minimal overhead during the download process; it consists of two components that perform the necessary manipulations on the bitstream and a finite state machine which controls the data multiplexer to select the correct output between the two blocks and the input. The BiRF solution is thought starting from the REPLICA filter, trying to adapt it to work with the Caronte architecture, [1, 9], and due to its little occupancy, as shown in Section 6, inserting it into the Caronte fixed side. The main inputs of the filter are: the configuration bitstream - in the form of a sequence of 32-bit words -, the encoded FPGA parameters, and the CLB column to which the bitstream has to be moved. The main output of the filter is the manipulated bitstream, which is generated out of the input bitstream and without any delay (input data is replaced only if necessary), an example of the reconfiguration via BiRF is shown in Figure 1.

All functional blocks will be briefly introduced in the following paragraphs.

5.1 The BiRF Parser

The core of the BiRF filter is the parser; its objective is to identify the FAR (Frame Address Register) and CRC (Cyclic Redundancy Check) commands in the bitstream, in order to allow the modification of the values that follow them by controlling the multiplexer.

These two commands are the only ones which have to be edited when performing relocation; so, when any other command or padding is recognized, the parser simply feeds the input through the output. Therefore, a finite state machine has been created to distinguish pad words from commands, and then between different commands. If the current word is the CRC or the FAR command, the parser activates the correct line of the multiplexer feeding through output the next word - or words - which contains the new CRC or FAR calculated by the other two blocks. As the BiRF changes a part of the bitstream values after their generation, specifically MJA values, the CRC has to be recalculated, as proposed in [3], and updated as well to prevent error recognition by the FPGA and allow the download process to continue. Therefore, the task of the CRC block is to perform recalculation of the checksum on all bitstream data words changed by the rest of the BiRF. This is done by means of the parser, which enables the CRC block immediately before the first word involved in checksum and activates the correct exit of the multiplexer to substitute old data. The process used in the physical implementation of the CRC computation is based on a Xilinx algorithm proposed in [3]. The FSM also has to distinguish between single word commands and multiple word commands, which have more parameters, in order to select the correct multiplexer line and keep it the same for the right amount of clock cycles. This is necessary because the MJA and CRC blocks work on whole commands and not single words.

6. TESTS AND RESULTS

Upon completion of the modules which compose the BiRF, the final architecture has been synthesised with Xilinx ISE 7.1i and its behavior has been tested. The results of the synthesis and testing steps are described in the following sections along with a comparison with available information on the REPLICA filter.

6.1 Synthesis data

The BiRF implementation on a Virtex 2 Pro XC2VP30 takes 549 Slices out of 13.696 which corresponds to 4 per cent of the total available slices, more details are shown in Table 1. The area occupancy of the BiRF filter is relatively small if compared with the total space available on the board it allow its subsequent implementation directly in a fixed section of the target FPGA eliminating the need for an external reconfiguration module. Table 1 shows complete occupation data, in slices and in percentage on three different FPGAs such as XC2VP7, XC2VP20 and XC2VP30.

The maximum combinatorial path delay reached by BiRF

	xc2vp7	**xc2vp20**	**xc2vp30**
Slices	549 out of 4928	549 out of 9280	549 out of 13696
Slices %	11,1 %	5 %	4 %
Flip Flops	249 out of 9856	249 out of 18560	249 out of 27392
Flip Flops %	2,5 %	1,3 %	0,9 %
4 Input LUTs	983 out of 9856	983 out of 18560	983 out of 27392
4 Input LUTs %	9,9 %	5,3 %	3,6 %

Table 1: Space requirements of the BiRF

is 6,128 ns, allowing a maximum clock frequence at which the filter can operate of 112,233 MHz. This is more than acceptable because speed of the dynamical reconfiguration process does not need to exceed 100 MHz which is less than the maximum allowed by the filter.

6.2 Validation Results

The BiRF has been tested with several bitstreams given as input along with specific FPGA parameters and the target column where the bitstream itself has to be moved. The bitstreams used for testing were generated using the Caronte flow [2, 9] and include both complete and partial configurations. The filter has been tested at a frequency of 112 MHz, corresponding to a period of approximately 9 ns that is sufficiently longer than the minimal time required by the BiRF to correctly process one data word in the worst case. Tests show that the Parser effectively recognises command words and parameters given and performs the actions needed to manipulate the bitstream; also the MJA and CRC blocks perform the calculations needed to update data word which become available to the output on the clock cycle that follows the FAR and CRC commands header.

bitstream	**size [KB]**	**words**	**time [ms]**	**throughput [MB/s]**
bs1	548	140.188	1,26	424,73
bs2	1.003	256.723	2,31	424,02
bs3	1.425	362.202	3,28	424,27
bs4	4,35	785	0,01	424,80
bs5	65	16.479	0,15	424,59

Table 2: Testing Results

Table 2 shows some experimental data on the relocation process, performed on various configuration bitstreams taken as an example. Those bitstreams have different size and describe the configuration, either partial or complete, of different Xilinx FPGAs: the XC2VP7, the XC2VP20 and the XC2VP30. Figure 2 represents the time (ms) that is necessary to relocate each of those configuration bitstream.

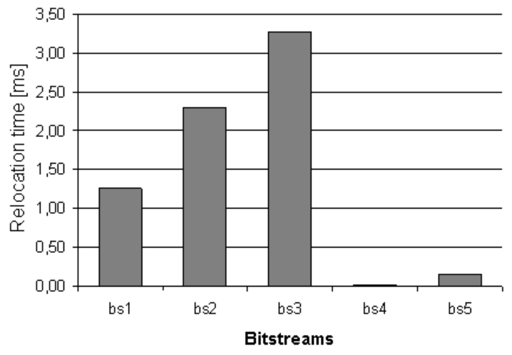

Figure 2: ***Relocation time of several bitstreams***

7. CONCLUDING REMARKS

The Bitstream Relocation Filter allows to manipulate a configuration bitstream during its downloading on an FPGA; opening the possibility for great performance improvements in partial reconfiguration. This happens by deleting the need to modify the bistream along with performing its necessary calculations on board occupation and functional requirements which become its unique concern when using a separate relocation filter and due to its small area requirements it could be easily implemented directly in the fixed part of the Caronte architecture. This could completely free the reconfiguration process from the need of being managed by an external computer, not even partially. Future improvements concern compatibility extension with other kinds of FPGAs, reduction of area requirements on board and, finally, integration of BiRF with other modules that perform the remaining tasks in the reconfiguration management process.

8. REFERENCES

[1] Alberto Donato, Fabrizio Ferrandi, Marco D. Santambrogio, and Donatella Sciuto. Exploiting partial dynamic reconfiguration for soc design of complex application on fpga platforms. In *IFIP VLSI-SOC 2005*, 2005.

[2] Alberto Donato, Fabrizio Ferrandi, Massimo Redaelli, Marco D. Santambrogio, and Donatella Sciuto. Caronte: a complete methodology to implement partially dynamically self-reconfiguring embedded systems on modern fpga. In *IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM 2005)*, 2005.

[3] S. Kelem. Virtex series configuration architecture user guide. *Xilinx XAPP151*, 2003.

[4] M. Porrmann H. Kalte, G. Lee and U. Rückert. Replica: A bitstream manipulation filter for module relocation in partial reconfigurable systems. In *The 12th Reconfigurable Architectures Workshop (RAW 2005)*, 2005.

[5] Xilinx Inc. Two flows for partial reconfiguration: Module based or small bit manipulations. *XAPP290*, May 2002.

[6] Edson Horta and John W. Lockwood. Parbit: A tool to transform bitfiles to implement partial reconfiguration of field programmable gate arrays (fpgas). *Washington University, Department of Computer Science, Technical Report WUCS–01–13*, July 2001.

[7] Markus Koester, Mario Porrmann, and Ulrich Rückert. Placement-oriented modeling of partially reconfigurable architectures. In *The 12th Reconfigurable Architectures Workshop (RAW 2005)*, 2005.

[8] U. Ruckert H. Kalte, M. Porrmann. System-on-programmable-chip approach enabling online fine-grained 1d-placement. In *18th International Parallel and Distributed Processing Symposium (IPDPS'04)*, 2004.

[9] Fabrizio Ferrandi, Marco D. Santambrogio, and Donatella Sciuto. A design methodology for dynamic reconfiguration: The caronte architecture. In *The 12th Reconfigurable Architectures Workshop (RAW 2005)*, 2005.