

POLITECNICO DI MILANO
MICROLAB
HPPS PROJECT PRESENTATION



Polaris
2D Relocation for Self Dynamical Run-time Reconfiguration

Tutor: Marco Domenico Santambrogio

Project Authors:
Marco Novati, Matr.: 708479

A.A. 2006-2007

Contents

1	Introduction	3
2	Xilinx Virtex-4 and Virtex-5 architecture	4
2.1	Frame Addressing	5
2.2	Configuration registers	7
2.3	Bitstream structure	11
3	BiRF Square	12
3.1	The BiRF Square Parser	13
3.2	The FAR calculation	14
3.3	The CRC calculation	14
4	Synthesis results	16
5	Simulation of BiRF Square	17
6	The Validation Phase	18
6.1	Target Device	18
6.2	Validation Architecture	19
6.3	Experimental Results	20
7	Concluding Remarks and Future Works	21

Abstract

Further to the release of a new generation of FPGA devices, like Xilinx Virtex-4 and Virtex-5, it is now possible to use a novel method of reconfiguration called 2D reconfiguration. In the context of self dynamical run-time 2D reconfiguration, relocation can be exploited, as in 1D reconfiguration, and is also a lot more critical due to the explosion in the number of possible system evolutions. The goal of this work is the creation of a solution to perform 2D relocation: particularly, the development starts from BiRF, a hardware filter exploiting 1D relocation on several FPGA families: Virtex, Virtex-E, Virtex-II, Virtex-II Pro. The aim is to develop a new version of BiRF that supports 2D relocation to be used on the new FPGAs, more specifically on Virtex-4 devices.

1 Introduction

It is possible to identify several models to manage a reconfigurable system, each of them with advantages or disadvantages. The context in which this work is based is 2D reconfiguration with a self partial dynamical run-time module-based approach: the means of these terms are explained in the following. 2D reconfiguration is a powerful technique that permits to configure a module with more flexibility, placing the functionality arbitrarily on the 2D FPGA area as long as there is enough empty area. The term Self is related to the location of the reconfiguration manager, which is on the FPGA: with this approach the system is more complex, but it becomes a stand-alone device with high performance and small size with no need of an external system for reconfiguration management. Partial is referred to the possibility of independently reconfiguring a fraction of the whole area; this is very useful, because normally only part of the configuration logic has to be substituted, i.e. a single functionality, and the reconfiguration time grows directly in the amount of logic to be configured. In a partial context, a Dynamical approach permits to reconfigure a part of the FPGA, while the rest of the system is still working. Finally, run-time means that the actual bitstream to be configured are generated only when necessary, and module-based means that every partial bitstream describes a whole functionality. In the context previously described, it is possible to use a powerful technique, called relocation, to obtain, starting from the bitstream used to completely describe a functionality in a single location, all the possible configuration files for the same functionality in different position of the FPGA. For 1D relocation, several alternatives exist in literature, either hardware or software; both have advantages and disadvantages depending on the system to which they are going

to be applied. With a software solution, there is no area occupation, but it is necessary to have a hardware processor embedded on the target FPGA or to configure a software one to perform its execution. A hardware solution, instead, even if it uses up part of the available configurable logic, has, in general, better performance, permits to free the processor from the task of relocation, and it is possible to use it in an architecture without requesting the presence of a processor in the system, neither embedded nor configured. For the reasons previously listed, a hardware solution has been chosen, to the development of a solution exploiting 2D relocation, starting from BiRF, a hardware filter that performs 1D relocation.

2 Xilinx Virtex-4 and Virtex-5 architecture

The Virtex-4 [1] and Virtex-5 [2] Family are the newest generation FPGA from Xilinx; combining a wide variety of flexible features, the Virtex-4 and Virtex-5 family enhance programmable logic design capabilities and are a powerful alternative to ASIC technology. The basic blocks are an enhancement of those found in the popular Virtex-based product families: Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X, allowing upward compatibility of existing designs. The types of block present in this family are: CLB, DSP, BRAM, IOB, and CLOCK. The Configurable Logic Blocks (CLBs) are the main logic resource for implementing sequential as well as combinatorial circuits. Each CLB element is connected to a switch matrix to access to the general routing matrix. The composition of CLB blocks are different. In Virtex-4 family, a CLB element contains four interconnected slices, each of them contains two 4-input-LUTs. In Virtex-5 family, instead, a CLB element contains a pair of slices, each of them with four 6-input-LUTs. DSP, Digital Signal Processing, is a new type of block present in Virtex-4 devices; the purpose is to deliver off-the-shelf programmable devices with the best mix of logic, memory, I/O, processors, clock management, and digital signal processing. Each DSP block contains two DSP48 slices; each of them supports many independent functions, including multiplier, multiplier-accumulator (MACC), multiplier followed by an adder, three-input adder, barrel shifter, wide bus multiplexers, magnitude comparator, or wide counter. The DSP48 slices available in all Virtex-4 family devices support new DSP algorithms and higher levels of DSP integration than previously available in FPGAs. The architecture also supports connecting multiple DSP48 slices to form wide math functions, DSP filters, and complex arithmetic without the use of generic FPGA fabric, which permits to obtain lower power usage, very high performance, and efficient silicon utilization. In Virtex-5 devices, there

is an extension of the Virtex-4 slices, DSP48E slice; the new enhancements provide improved flexibility and utilization, improved efficiency of applications, reduced overall power consumption, increased maximum frequency, and reduced set-up plus clock-to-out time. The high performance allows designers to implement multiple slower operations in a single DSP48E slice using time-multiplexing methods. The new block RAMs are similar to the older ones: each block RAM stores 18K bits of data in a Virtex-4 device, while Virtex-5 block RAM stores up to 36K bits of data and can be configured as either two independent 18 Kb RAMs, or one 36 Kb RAM. Write and Read are synchronous operations; the two ports are symmetrical and totally independent, sharing only the stored data. Each port can be configured in one of the available widths, independently from the other port; the read port width can be different from the write port width for each port. The memory content can be initialized or cleared by the configuration bitstream. During a write operation the memory can be set to have the data output either remain unchanged, reflect the new data being written or the previous data now being overwritten. All Virtex-4 and Virtex-5 FPGAs have configurable high-performance SelectIO drivers and receivers, supporting a wide variety of standard interfaces. The robust feature set includes programmable control of output strength and slew rate, and on-chip termination using Digitally Controlled Impedance (DCI). Each IOB contains both input, output, and 3-state SelectIO drivers. These drivers can be configured to various I/O standards. Differential I/O uses the two IOBs grouped together in one tile. For clocking purposes, each Virtex-4 and Virtex-5 device is divided into regions: the number of regions varies with device size, eight regions in the smallest device to 24 regions in the largest one. Each device has 32 global clock lines that can clock all sequential resources on the whole device (CLB, block RAM, and I/O), and also drive logic signals.

2.1 Frame Addressing

A frame is the smallest amount of configuration information that can be accessed. It is formed by a vertical stack of 1312 bits that spans the whole height of a row. Each configuration frame in the FPGA has a unique 32-bit address that can be divided into Block type, Top/Bottom indicator, Row address, Major address, Minor address.

The row address identifies a row in the FPGA. The device is divided in two part, top and bottom; the rows are numbered from 0 (up to 9) in the top and bottom halves of the FPGA, starting from the center, as shown in Figure 1. During configuration, the row addresses are scanned from 0 upward in the top half, and then from 0 downward in the bottom half. The largest Virtex-

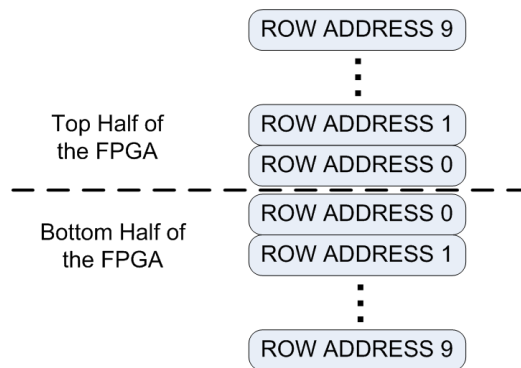


Figure 1: Row Addressing

5 device has a total of 12 rows (6 in each half), even though the hardware design can support up to 20 rows (10 in each half).

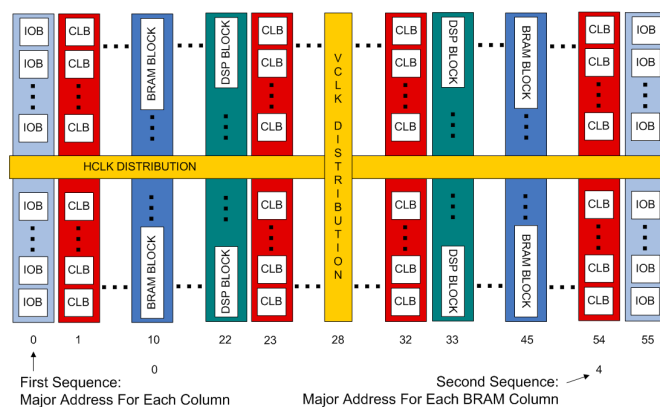


Figure 2: Column Addressing

Figure 2 shows the structure of a row in Virtex-4 or Virtex-5 devices; a row consists of a stack of basic blocks with a row of HCLK tiles passing through the middle. Each row is divided into the same number of columns, where a column corresponds to a block in the array (CLB, DSP, block RAM, IOB, etc.). Table 1 shows the number of blocks in the different column of a row for Virtex-4 and Virtex-5 FPGA. The major addresses are numbered from left to right starting with 0.

There are two sequences of major address per row: the first sequence assigns a major address to each column, and it is used to select a column for normal configuration, while the second sequence assigns a major address to each block RAM column, which is used to access configuration frames that are specific to the block RAM contents. Consequently, block RAMs have two

Column Type	Virtex-4	Virtex-5
CLB	16	20
DSP	8	8
BRAM	4	4
IOB	16	20

Table 1: Number of Blocks per Column in a Row

major addresses: one to access their normal configuration, and the other to access their contents.

Column Type	Frames Number
CLB	36
DSP	28
BRAM	30
IOB	54
Clock Column	4

Table 2: Frames per Column

Each column contains a certain number of frames, which are accessed using the minor address. The number of frames inside a column depends on the block type and on the type of column. Table 2 lists the number of frames (minor addresses) per column when using this block type. The frames are numbered from left to right, starting with 0.

2.2 Configuration registers

All Virtex-4 [3] and Virtex-5 [4] commands are executed by reading from or writing to the configuration register. There are two types of bitstream command header; type 1 includes a word count, defining the number of data words that follow. If word count equals 0, the following word must be a type 2 header that contains an extended word count for the following parameters.

Header Type	Op code	Register	Rsrvd	Word Count
[31:29]	[28:27]	[26:13]	[12:11]	[10:0]
001	xx	rrrrrrrrrrxxxxxx	rr	xxxxxxxxxxxx

Table 3: Type 1 Header Format

In Table 3 is shown the Type 1 packet header format, that is composed by, from left to write: three bits indicate the header type, which are 001 for Type 1 and 010 for Type 2; the two Opcode bits, which are 00 for NOP, 01 for read operation and 10 for write operation; fourteen bits for the register

addressing, even if only five of them are used; 2 bits reserved for future use; eleven bits that contain the word count field. The Type 2 data packet must follow a Type 1 packet with word count equal to zero, and is used to write long blocks, particularly when the number of data words are greater than 2048. The Type 2 packet header is shown in Table 4.

Header Type	Opcode	Word Count
[31:29]	[28:27]	[26:0]
010	rr	All used

Table 4: Type 2 Header Format

No register address is present, because it is contained in the previous Type 1 header; there are three bits that signal the type, which is 010, two bits reserved for future use, and the remaining bits that contain the word count value. Table 5 summarizes the register for Virtex-4 and Virtex-5 devices: an explanation of the registers follows.

CRC register contains the expected checksum value; while the configuration data frames are loaded, the device calculates a Cyclic Redundancy Check value from the configuration data packets. After the configuration process the configuration bitstream can contain a Check CRC instruction, followed by an expected CRC value.

If the value calculated by the device does not match the expected CRC value in the bitstream, the device is put in an error state. The CRC check is included in the configuration bitstream by default, although the designer can disable it if desired; if the CRC check is disabled, there is a risk of loading incorrect configuration data frames, or it is possible to damage the device. Virtex-4 and Virtex-5 devices use a 32-bit CRC check. The CRC32C polynomial is:

$$x^{32} + x^{28} + x^{27} + x^{26} + x^{25} + x^{23} + x^{22} + x^{20} + x^{19} + x^{18} + x^{14} + x^{13} + x^{11} + x^{10} + x^9 + x^8 + x^6 + 1$$

The Virtex-4 and Virtex-5 devices are divided into two halves: the top half and the bottom half. Frames in the bottom half mirror that in the top half with the exception of the vertical HCLK rows that contain the global and regional clocks. The HCLK title bits are in the same order in the both of the top and bottom frames. All Frames in Virtex-4 have a fixed, identical length of 1312 bits (41 32-bit words). One Frame configures one HCLK with either 4 block RAMS, 32 IOBs or 4 DSPs. The address can be written directly or can be auto-incremented at the end of each frame. The typical bitstream starts at address 0 and auto-increment to the final count. The FAR is divided into

Reg. Name	Addr.	Description
CRC	00000	CRC register
FAR	00001	Frame Address Register
FDRI	00010	Frame Data Register Input
FDRO	00011	Frame Data Register Output
CMD	00100	Command register
CTL0	00101	Control register 0
MASK	00110	Masking register for CTL0 and CTL1
STAT	00111	Status register
LOUT	01000	Legacy Output Register
COR0	01001	Configuration Option Register 0
MFWR	01010	Multiple Frame Write Register
CBC	01011	Initial CBC value register
IDCODE	01100	Device ID register
AXSS	01101	User Bitstream Access Register
COR1	01110	Configuration Option Register 1
CSOB	01111	Used for daisy chain parallel interface
WBSTAR	10000	Warm Boot Start Address Register
TIMER	10001	Watchdog Timer Register
BOOTSTS	10110	Boot History Status Register
CTL1	11000	Control Register 1

Table 5: Configuration Registers

five fields: top/bottom bit, that permits to select between top-half rows and bottom-half rows; block type, which are CLB/IO/CLK (000), block RAM Interconnect (001), block RAM content (010), CFG CLB (011), and CFG BRAM (100); row address, which selects a row of frames, for example, a row of 16 CLBs in height, with an HCLK row in the middle; column address, that selects a major column, such as a column of CLBs; minor address, that selects a memory-cell address line within a major column. The order of this parameters changes from Virtex-4 to Virtex-5, as in shown in Table 6 and 7.

Address Type	Bit Index
Top/Bottom Bit	[22]
Block Type	[21:19]
Row Address	[18:14]
Column Address	[13:6]
Minor Address	[5:0]

Table 6: Virtex-4 FAR Composition

FDRI register is used to configure frame data at the frame address specified in the FAR register. The Frame Data Register (FDR) is a shift register

Address Type	Bit Index
Block Type	[23:21]
Top/Bottom Bit	[20]
Row Address	[19:15]
Column Address	[14:7]
Minor Address	[6:0]

Table 7: Virtex-5 FAR Composition

in which the data are loaded before the transfer to the configuration memory. The new configuration is written to the FPGA by the load of the CMD register with the command WCFG and the consequent write of at least two 32-bit word in the FDRI. FDRO is used to read configuration data from the FPGA, in a process called Readback; this is performed by loading the CMD register with the RCFG command and then addressing the FDRO with a read command. The CMD contains the configuration commands, that are used to manage the configuration control logic and to perform other configuration functions. The command present in the CMD register is executed each time the FAR is loaded with a new value. Table 8 gives the Command Register commands and codes.

Command	Code	Description
NULL	00000	Null command
WCFG	00001	Writes Configuration
MFW	00010	Multiple Frame Write
DGHIGH /LFRM	00011	Last Frame
RCFG	00100	Reads Configuration
START	00101	Begins Startup Sequence
RCAP	00110	Resets Capture Signal
RCRC	00111	Resets CRC
AGHIGH	01000	Asserts GHIGH B Signal
SWITCH	01001	Switches the CCLK
GRESTORE	01010	Pulses GRESTORE signal
SHUTDOWN	01011	Begins shutdown sequence
GCAPTURE	01100	Pulses GCAPTURE
DESYNCH	01101	Resets the DALIGN signal
Reserved	01110	Reserved
I PROG	01111	Internal Prog (Virtex-5 Only)
CRCC	10000	Recalculates the CRC value (Virtex-5 Only)
LTIMER	10001	Reloads watchdog timer (Virtex-5 Only)

Table 8: Command Registers

CTL0 and CTL1 – CTL in Virtex-4 devices – are used for setting the security level of the reconfiguration process, managing read and write per-

mission, the persistence option, in order to permit the reconfiguration and the readback, and the global tristate signal. The Mask Register (MASK) is used to write to the CTL register. A '1' in bit N of the mask allows that bit position to be written in the CTL register. The default value of the mask is all '0's. The STAT register contains the current value of the several control or state signals; it can be read through the SelectMAP or JTAG interfaces. The Legacy Output Register perform the daisy chaining of the configuration bitstream to other Xilinx devices. Data written to the LOUT is serialized and appears on the DOUT pin. The COR1 and COR2 registers – COR in Virtex–4 devices – are used to select specific configuration options. Multiple Frame Write Register is used for the bitstream compression option. CBC Register is used by the bitstream compression option to hold the Initial Vector (IV) for AES decryption. Any writes to the FDRI register must be preceded by a write to the IDCODE register. The provided IDCODE must match the device IDCODE. A read of this register returns the device IDCODE. AXSS Register is used to support the USR ACCESS primitive. The following registers are present only in Virtex–5 devices: CSOB Register, that permits to assert the CSO B pin for parallel daisy–chain operation; WBSTAR, which is used for warm boot start sequence; TIMER, that permits to enable to manage the Watchdog Timer; BOOTSTS, that is the Boot History Status Register.

2.3 Bitstream structure

A bitstream is a binary file that contains all the device data configuration, which is uploaded through one of the available interfaces. A configuration file is composed by 32–bit words, that contain commands for the configuration of the FPGA, additional data, padding and synchronization blocks.

Commands are composed by one header, for Type 1 command, and an additional second header, for Type 2 command; then there are the parameters, in number equal to the word count contained in the previous words. Commands of the CMD group are particular, in fact they have no parameters: they are composed by a general header, which identify them as command, followed by a word that contains the actual command to be executed. These initial informations are followed by the actual parameters, in a number equal to the one indicated in the word count field. Other than commands, there are padding words, and two particular words, called Dummy Word and Sync Word, that remark the begin of the actual bitstream. Table 9 shows the structure of an example bitstream which includes the more interesting commands for the reconfiguration process; the bitstream start is marked by the sequence of Dummy and Sync word, that can be followed by various headers

Hex Word	Explanation
...	...
FFFFFFFF	Dummy Word
AA995566	Sync Word
...	...
30008001	Type 1 write 1 word to CMD
00000007	Reset CRC
...	...
30002001	Type 1 write 1 word to FAR
xxxxxxxx	FAR value
30008001	Type 1 write 1 word to CMD
00000001	Write Configuration
...	...
30004000	Type 1 write 0 word to FDRI
50000400	Type 2 write 1024 words to FDRI
xxxxxxxx	Data word 1
...	...
xxxxxxxx	Data word n
xxxxxxxx	Data word n+1
xxxxxxxx	Data word last (1024)
...	...
30000001	Type 1 write 1 word to CRC
xxxxxxxx	CRC value
...	...

Table 9: A bitstream example

and commands; at some point, there must be the Reset CRC command, that marks the begin of data on which the checksum has to be calculated and therefore the start of the reconfiguration phase. After that, there can be other instructions, until the FAR command, which contains the starting position for the reconfiguration, followed by the Write Configuration command, that initializes the actual write, and by the FDRI block, which describes the real logic of the functionality to be configured on the target FPGA. Finally, there is the instruction that writes the calculated CRC value on the FPGA register, in order to check the integrity of the bitstream, this can followed by other commands, on which the CRC has not to be calculated.

3 BiRF Square

This section describes the proposed approach, based on blank module reconfiguration with relocation, applied to a self partially reconfigurable architecture. The approach offers a great advantage in terms of memory usage,

because it permits to reduce the number of the partial bitstreams that are necessary to allow the configuration of a functionality in all the possible location, and that have to be saved in the device memory. BiRF Square (Bitstream Relocation Filter Square) is a hardware filter that performs relocation of partial bitstreams on Virtex-4 and Virtex-5 devices. The filter consists of two logical unit, MJA and CRC, that perform the necessary manipulations on the bitstream, and of a finite state machine which selects the correct output of the filter. The main inputs of the filter are the configuration bitstream - in the form of a sequence of 32-bit words -, and the address that specifies the location to which the partial bitstream has to be moved. The main output of the filter is the manipulated bitstream, which is provided word by word and it is available for use one clock cycle after the last input word is read. The parser and all the logical unit will be briefly introduced in the following paragraphs.

3.1 The BiRF Square Parser

The need of a parser is due to the necessity to identify the FAR (Frame Address Register) and CRC (Cyclic Redundancy Check) commands in the bitstream, in order to perform the modification of the parameters that follow them. These two commands are the only ones which have to be altered in the relocation process; for this reason, when any other command or padding is recognized, the parser has to leave the bitstream unchanged, simply feeding the input through the output. Therefore, a finite state machine has been created, depicted in Figure 3.

This FSM is an adaptation and an optimization of the BiRF one for Virtex-4 and Virtex-5 devices. In the adaptation process, the first change has been introduced: this version does not separate commands with a single parameter to commands with multiple parameters. Furthermore, some code optimizations have been done, in order to reduce the area consumption and to increase the core performance in term of frequency. The behaviour of the FSM is briefly described in the following. When BiRF Square is reset, the first fase, that performs the recognition of the beginning of bitstream commands, starts; the FSM enters the Dummy state, which waits for Dummy word that may signal the beginning of the command words. After the recognition of such word, the Parser waits for Sync word: if it is received immediatly after the Dummy word, this phase is terminated, and the commands recognition starts entering in the Wait state. If the current word is the CRC or the FAR command, the parser feeds through output the next word - or words - which contains the new CRC or FAR. When a generic command is recognized, the FSM also has to calculate the number of parameters, in order to keep the

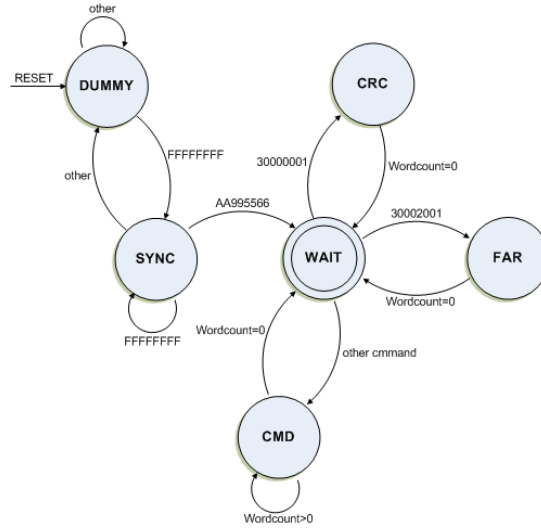


Figure 3: BiRF Square Parser FSM

bitstream the same for the right amount of clock cycles.

3.2 The FAR calculation

The FAR calculation is an important part of the relocation process; it permits to create the new parameter that will substitute the old one in the bitstream. The target address is an input of BiRF Square, in a form of a 14-bit word: the first bit is the top-bottom bit indicator, the next 5 bits contain the row address and the last 8 bits specify the column address. The calculation is different in Virtex-4 and Virtex-5 devices, as the two compositions of FAR are different too. The formulae that have been used to perform such task are described in Table 10, where “Dest” is the input of BiRF Square that provides the destination of the partial bitstream.

Family	FAR Formula
Virtex-4	000000000 Dest(0) 000 Dest(1 to 13) 000000
Virtex-5	000000000000 Dest(0) Dest(1 to 13) 0000000

Table 10: FAR formulae for Virtex-4 and Virtex-5

3.3 The CRC calculation

Xilinx FPGAs use a Cyclic Redundancy Check mechanism to detect errors during data download; the checksum is calculated during the bitstream

generation process and, also, during data transfer to the FPGA. If, during download on board, the two CRC values do not match, the reconfiguration is stopped, because an inconsistency has been detected. As BiRF Square changes a part of the bitstream, the CRC has to be recalculated to prevent error recognition by the FPGA and allow the download process to successfully complete. From an additional examination of Virtex-4 and Virtex-5 bitstream, it has been discovered that different bitstreams generated with Xilinx tools have the same CRC values. After some deeper analysis, it emerged that there are two predefined values, one for Virtex-4 and one for Virtex-5 devices, that permit to bypass the CRC calculation; more precisely, if the parameter of the CRC command is the predefined value, the device check is always successful. Thanks to these considerations, two different versions of BiRF Square have been developed. The former does not perform the CRC calculation, substituting the CRC command parameter with the correct predefined value, which permits to obtain better performance and lower area occupation; this solution must be used in environments in which the bitstream correctness is guaranteed. The latter, which performs CRC calculation, has worse performance and consumes more space, but it is a general purpose version, and therefore it can be used in all the environments. The algorithm for the computation of the Virtex-4 and Virtex-5 CRC has been derived from the previous 16-bit-CRC algorithm, provided by Xilinx in [5]. Such algorithm analyzes the bitstream word by word, and calculates the CRC by using a part of the header and the parameters of some write commands. For these reasons, the CRC process has to be able to recognize if the current command is a read or a write one; in this last case, it is necessary to know if it is in the set of the command on which the checksum has to be calculated. So, if the current command is a write one and it is in the previously cited set, the CRC process has to store a part of the header and to perform the checksum calculation by using the parameters that will be contained in the next words; otherwise, the current command has to be ignored with also all its parameters, in order to permit the CRC process to analyze the next useful word. Regarding the actual CRC computation, the polynomial used by Xilinx Virtex-4 and Virtex-5 devices is the following:

$$x^{32} + x^{28} + x^{27} + x^{26} + x^{25} + x^{23} + x^{22} + x^{20} + x^{19} + x^{18} + x^{14} + x^{13} + x^{11} + x^{10} + x^9 + x^8 + x^6 + 1$$

In order to perform the computation of such CRC, a parallel implementation has been used, to provide the new checksum within the minimum amount of time to obtain better performances.

4 Synthesis results

The BiRF Square general purpose version has been synthesized on three different Xilinx Virtex-4 and three different Virtex-5 FPGAs. Table 11 and Table 12 respectively show area requirements on such devices.

	xc4vlx40	xc4vlx60	xc4vlx100
Slices	376	376	376
Slices %	2 %	1,4 %	0 %
Flip Flops	175	175	175
Flip Flops %	0,5 %	0,3 %	0 %
LUTs	715	715	715
LUTs %	1,9 %	1,3 %	0 %

Table 11: Space Requirements on Virtex-4

	xc5vlx50	xc5vlx85	xc5vlx110
Slice Registers	175	175	175
Slices Registers %	0,5 %	0,3 %	0,2 %
Slice LUTs	555	555	555
Slice LUTs %	1,9 %	1,1 %	0,8 %

Table 12: Space Requirements on Virtex-5

Considering the general purpose solution, implemented using a Virtex-4 with speed grade -12, it is possible to obtain a maximum frequency of 160 MHz. On a Virtex-5 device with speed grade -3, the maximum frequency at which BiRF Square can operate is 226 Mhz. Both area and timing requirements can be further improved by using the optimized version of BiRF square, previously described. Table 13 and 14 show the area requirements of the Lite version of BiRF Square on Virtex-4 and Virtex-5 devices.

	xc4vlx40	xc4vlx60	xc4vlx100
Slices	89	89	89
Slices %	0,5 %	0,3 %	0 %
Flip Flops	72	72	72
Flip Flops %	0,2 %	0,1 %	0 %
LUTs	161	161	161
LUTs %	0,4 %	0,3 %	0 %

Table 13: Space Requirements on Virtex-4 of the optimized version

	xc5vlx50	xc5vlx85	xc5vlx110
Slice Registers	72	72	72
Slices Registers %	0,2 %	0,1 %	0,1 %
Slice LUTs	134	134	134
Slice LUTs %	0,5 %	0,3 %	0,2 %

Table 14: Space Requirements on Virtex-5 of the optimized version

With those versions, the area usage is significantly smaller than the one proposed for the general purpose version, by a factor of 4. On a Virtex-4 with speed grade -12, the maximum reachable frequency is 304 MHz, while on a Virtex-5 with speed grade -3 BiRF Square can operate at a maximum frequency of 290Mhz. Again this is a noticeable improvement from the general purpose version by approximatively 80 % for the Virtex-4 version, and 32 % for the Virtex-5 one.

5 Simulation of BiRF Square

Upon completion of its development, BiRF Square has been simulated in order to test its correct behavior. Such simulation has been performed using ModelSim, a powerful tool that permits, starting from the VHDL specification, to simulate the behavior of a hardware component. In particular, by using such tool it is possible to visualize, cycle per cycle, the trend of all the signals of the target architecture; this is a very useful feature of this software, because it is very helpful in the process of detection of undesirable inconsistencies. ModelSim is distributed from Mentor Graphics, and it is available in a free version; the version used for this work is XE III 6.2c. The simulation process has been performed using several configuration bitstream, either partial or total. The functioning of the filter has been simulated at a frequency of about 150 MHz, corresponding to a period of 6,5 ns, that is enough to allow BiRF Square to process a word, even in the worst case. Simulation results show that the Parser correctly recognizes commands and the related parameters, in order to perform the modification needed for the bitstream relocation process; the filter also computes the new CRC value and the new target FAR, and correctly substitutes the old values when needed.

Figure 4, 5 and 6 show respectively the recognition of the FAR command, of the CRC and of a generic command. As shown in the figures, the filter correctly replaces the old word when needed for the relocation process.

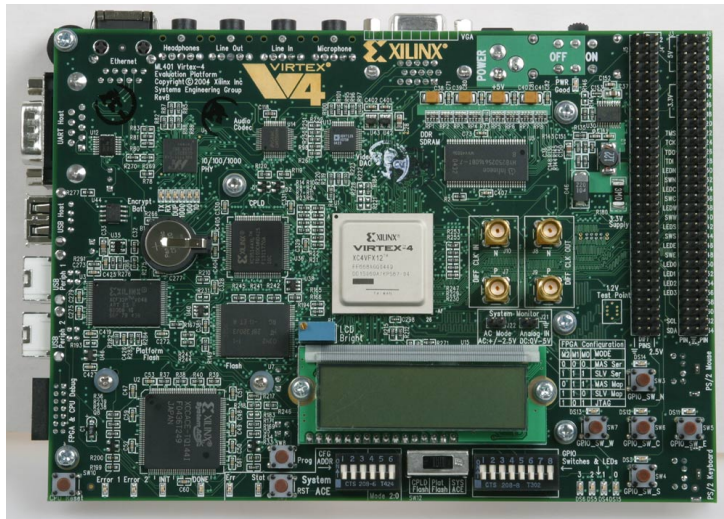


Figure 7: Xilinx Virtex-4 Evaluation Board

tors (Differential Clocks), 2 PS/2 Connectors (Keyboard/Mouse), 2 Audio (In/Out, Microphone/Head Phone), RS-232 Serial Port, 3 USB Ports (2 Peripheral/1 Host), PC4 JTAG, DB 15 VGA Display, 10/100/1000 RJ-45 Ethernet Port, 64 Bit User Expansion Connector, General Purpose I/O (Buttons and LEDs) and a 16 x2 Character LCD. The memory device embedded on the board are 64 MB DDR SDRAM, 8Mb ZBT SRAM, 64 Mb Flash, 4 kb IIC EEPROM.

6.2 Validation Architecture

Figure 8 shows the fixed part of the validation architecture. As shown in the picture, the communication is exploited through the OPB bus, a standard bus on which the IP-Cores of the fixed side are connected, along with the interfaces that permit external communication.

The components used to define core of the architecture are the Microblaze soft-processor, the internal BRAM memory and the ICAP internal reconfiguration interface; then there is the IPCM Core, which provides the system with new drivers for reconfigurable modules, the interrupt controller, for the interrupt management, the timer, that permits the throughput calculation, the RS232 Serial interface, used to transfer data to and from the FPGA, the controller of DDR-SDRAM Memory, used to save the partial bitstreams, and the BiRF Square filter, to perform relocation. The architecture has been configured on the device by means of the JTag interface. The processor code, executed by the Microblaze to perform the management of the relocation

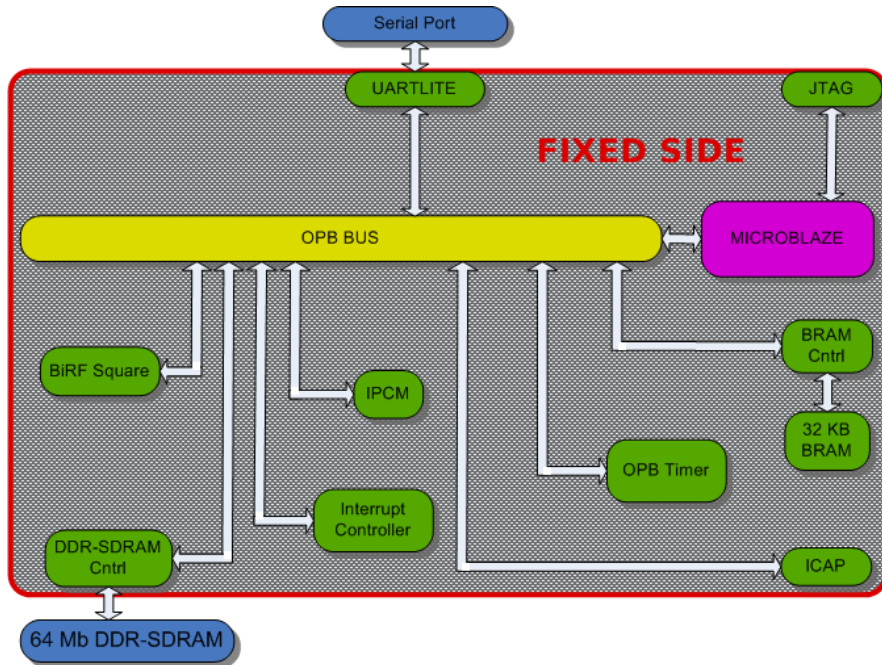


Figure 8: Validation Architecture

process, along with other software that allows external communication and internal system management was stored in the internal BRAM. In order to import BiRF Square in such architecture, it has been interfaced on the OPB bus by using the IPgen tool: table 15 shows the synthesis results of the interfaced version of the filter, in the optimized version, on the target device, xc4vFX12.

Slices	Slices %	Flip Flops	Flip Flops %	LUTs	LUTs %
170	3 %	227	2 %	215	1 %

Table 15: Synthesis results of the interfaced version on the target Virtex-4

From the table it emerges that these area requirements are quite good, even in this small device. Finally, after the interface creation process, the drivers have been developed, and BiRF Square has been successfully imported in the validation architecture.

6.3 Experimental Results

After the bitstream is loaded in memory, by the serial port, the relocation target column is given to BiRF Square; then the bitstream is fed word by

word to the filter, which performs the relocation by manipulating the current word, if needed. As the relocation process goes on, the individual words are immediately updated in memory and, one clock cycle after the last word is read, the relocated bitstream is available to be used. Table 16 shows the time taken by BiRF Square to perform the relocation of ten partial bitstreams and its throughput.

Bitstream	Size[KB]	Rel. Time[ms]	Throughput[MB/s]
1	1.45	0,194	7,3097
2	3.14	0,419	7,3095
3	10.61	1,417	7,3092
4	65.91	8,805	7,3097
5	94.12	12,574	7,3095
6	139.68	18,662	7,3093
7	148.79	19,878	7,3097
8	189.56	25,325	7,3097
9	570.44	76,209	7,3098
10	1026.89	137,200	7,3092

Table 16: BiRF Square Relocation Time and Throughput

As shown in the table, BiRF Square offers a throughput of approximately 7,3 MB/s, which allows its effective use in a self dynamically reconfigurable system.

7 Concluding Remarks and Future Works

BiRF Square has been developed for the purpose of applying the relocation technique in a self partially and dynamically reconfigurable system. The advantage of the filter realization is the significant reduction in the number of bitstreams to be stored in the internal memory of the system. More precisely, the memory saving grows with the number of slots available for the reconfiguration process, which is reasonable due to the fact that relocation allows to store, for each functionality, only one bitstream instead of one for each slot. In addition to the reduction of memory allowed by relocation, the results obtained by BiRF Square can be evaluated in terms of efficiency. For this purpose, particularly relevant are the area requirements and the performance reached in terms of frequency. In fact, the use of BiRF Square in a self partially and dynamically reconfigurable system does not hinder its performance; it is very important that the area increase in the fixed part of

the device and the time overhead due to relocation process are balanced by the advantages in terms of memory saving. The area requirements of the filter have been presented in section bla; the occupation ratio is relatively small, so the creation of a reconfigurable system with BiRF Square is possible with acceptable fixed area requirements. Regarding the performance of BiRF Square, the time needed for the relocation of a bitstream varies from a fraction of millisecond to a dozen of milliseconds, and it grows linearly in the bitstream dimension. The throughput, calculated as the ratio of bitstream dimension and its relocation time, is basically a constant of 7,3 MB/s. It is a very good result, considering the average size of a partial bitstream that can be used in a reconfigurable architecture on FPGA. In general, a total configuration file size is about 1 MB. Consider an architecture that uses one third of the area as fixed part and the rest as reconfigurable part divided into six slots; with such hypothesis, the average size of a partial bitstream will be about 110 KB, which corresponds to a relocation time of about 15 ms, which is more than acceptable. Regarding future work, an improvement that could be done is giving the filter direct access to the memory (DMA) to free the processor from the job of feeding the filter with the bitstream word by word and to allow BiRF Square to perform direct manipulation of the bitstream. Furthermore, it is possible to integrate the filter and the ICAP component, which is the component that performs the actual reconfiguration; with such improvement, the relocation overhead can be eliminated, due to the fact that the relocation time is much lower than the reconfiguration time.

References

- [1] X. Inc., “Virtex-4 user guide,” Xilinx Inc., Tech. Rep. ug70, March 2007. [Online]. Available: <http://www.xilinx.com/bvdocs/userguides/ug70.pdf>
- [2] —, “Virtex-5 user guide,” Xilinx Inc., Tech. Rep. ug190, February 2007. [Online]. Available: <http://www.xilinx.com/bvdocs/userguides/ug190.pdf>
- [3] —, “Virtex-4 configuration user guide,” Xilinx Inc., Tech. Rep. ug71, January 2007. [Online]. Available: <http://www.xilinx.com/bvdocs/userguides/ug71.pdf>
- [4] —, “Virtex-5 configuration user guide,” Xilinx Inc., Tech. Rep. ug191, February 2007. [Online]. Available: <http://www.xilinx.com/bvdocs/userguides/ug191.pdf>
- [5] S. Kelem, “Virtex series configuration architecture user guide,” *Xilinx XAPP151*, 2003.