Secure Partial Reconfiguration of FPGAs

Amir Sheikh Zeineddini and Kris Gaj
George Mason University
4400 University Drive, Fairfax, VA 22030 USA
{asheikhz, kgaj}@gmu.edu

Abstract

SRAM FPGAs are vulnerable to security breaches such as bitstream cloning, reverse-engineering, and tampering. Bitstream encryption and authentication are two most effective and practical solutions to improve the security of FPGAs. In this paper, we investigate a method to perform a secure dynamic partial reconfiguration of SRAM FPGAs using embedded processor cores. Two schemes based on hard-wired PowerPC processor core and the MicroBlaze soft processor core have been compared and contrasted in terms of speed and FPGA resource usage. A practical experiment, demonstrating feasibility, performance, and flexibility of both schemes has been conducted using Xilinx ML310 board with Xilinx Virtex-II Pro FPGA.

1. Introduction

As the performance gap between FPGAs and ASICs decreases [1], platform FPGAs with various configurable elements and embedded blocks provide new solutions for high density and high-performance embedded system designs. These platforms not only enable system architects to design and develop complex custom systems using embedded processors and interoperable IP cores, but also provide technologies such as dynamic reconfiguration of part of an FPGA while other areas of the device remain operational. There are many advantages in partial dynamic reconfiguration especially for applications that require adaptive and flexible hardware such as mobile communication applications and real-time embedded systems. Deploying dynamic run-time reconfiguration in systems results in reduced chip area and power consumption.

Considering the wide range of features, platform FPGAs address many new application areas. An increase in their popularity makes the need for design security mechanisms even more important especially in high-security areas where SRAM FPGAs might not otherwise be acceptable. The design security must protect the design against cloning and reverse engineering. A survey in [2] analyzes possible attacks against FPGAs. In the case of SRAM FPGAs this survey is directly concerned with protection of a bitstream especially during configuration and reconfiguration. Bitstream encryption as a solution increases the level of security and makes the configuration bitstream secure against attackers.

The Xilinx [3] security solution, known as SecureChip technology, uses CAD tools for bitstream encryption and an embedded hard-wired internal circuit for decryption [4]. One of the drawbacks of this scheme is that the partial reconfiguration capability of FPGA is disabled and therefore a device configured with an encrypted bitstream cannot be partially reconfigured.

In this paper, we propose a method that uses embedded microprocessor cores to achieve bitstream security, specifically for designs that benefit from partial reconfiguration. This method is capable of performing secure partial reconfiguration of the FPGA after the initial configuration. It provides the flexibility of using arbitrary algorithms for authentication and encryption of partial bitstreams. It can also facilitate secure remote partial reconfiguration for vendor updates and feature upgrades in the field.

The rest of the paper is organized as follows. Section 2 presents the related previous work and background. In Section 3 an overview of Xilinx EDK tools and evaluation board is presented. Section 4 explains the hardware architecture of the implemented self-reconfiguring systems for both hard and soft processor cores. Section 5 presents the methodology of the experiment. Section 6 presents the obtained results and the discussion of the results. In Section 7 conclusions and future work is provided.

2. Related Work and Background

2.1. Related Work

The Xilinx SecureChip technology is simple and efficient. All Virtex-II family devices (Virtex-II, Virtex-II Pro, and Virtex-II Pro X FPGAs) use the

Triple DES encryption scheme [4]. In Virtex-4 devices, Triple DES has been replaced with AES to increase security and throughput. The scheme exploits software support of Xilinx ISE CAD tools for both encryption of the bitstream and key generation. **Figure 1** shows the Xilinx security system.

For decryption, it uses an on-chip decryptor along with the internal decryption keys stored in a dedicated memory. Either an externally-connected battery or an auxiliary power supply (V_{CCAUX}) is the source of power for volatile storage of the keys. The keys are erased if there is a tampering with the device.

The problem with this scheme is the extra area and cost needed for the external battery, lack of flexibility, and the disablement of partial reconfiguration for encrypted bitstreams.

A method proposed by Algotronix [6] removes the need for an external battery by finding another way of storing the secret key on the FPGA such as use of laser to engrave the key. This will make it necessary for the FPGA to contain both encryption and decryption circuits, and hence there is no need for the software to support encryption. This solution uses even more FPGA silicon area than Xilinx

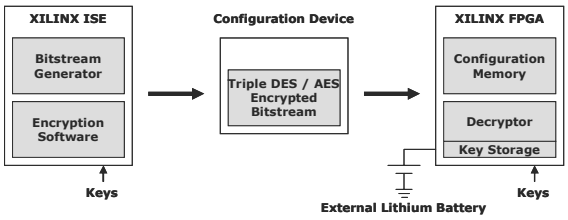


Figure 1. Xilinx SecureChip Technology

scheme and also lacks flexibility since the encryption and decryption circuits are fixed with no possibility of upgrade or use of another algorithm.

In [7], a new solution is proposed with no implementation available at the moment. The scheme selects and places the cores for encryption and decryption in the FPGA, and then removes them to free the chip area. A dedicated configuration controller manages both the encryption and decryption schemes by relying on partial and self-reconfiguration. This configuration scheme also uses an embedded key instead of an externally-powered storage for the secret key.

The method is flexible and adjusts the security level to application needs but is relatively complex considering the limitations of partial reconfiguration imposed by the FPGA manufacturers and the CAD tools. To our best knowledge, this method has never been implemented in practice.

The work presented in this paper focuses on a more specific case in which only secure partial reconfiguration after initial configuration is considered. By using an embedded microprocessor as a configuration controller inside of the chip, secure partial reconfiguration can be achieved. The next section provides a background about the flows for partial reconfiguration along with the overview of the Virtex-II Pro platform FPGA and the tools for creating a self-reconfiguring design.

2.2. Background

The Virtex-II Pro device used in our designs is the first platform FPGA capable of implementing flexible, high performance, and low-cost system-on-a-chip designs by combining a variety of features embedded in the FPGA fabric with specially developed hardware/software IP cores [8]. Particularly, it incorporates fully embedded IBM PowerPC405 processor core which is an implementation of the PowerPC embedded environment architecture [9]. The embedded PPC405 core is a 32-bit Harvard architecture processor with functional units such as cache unit and memory management unit (MMU). It operates in a five-stage pipeline, and most instructions execute in a single cycle. It is capable of more than 300 MHz clock frequency and 420 Dhrystone MIPS, and is contained in a processor block. Processor block also contains on-chip memory controllers and integration circuitry compatible with IBM CoreConnect bus architecture [10] that enables the compliant IP cores to integrate with this block. The CoreConnect architecture provides three buses for interconnection of hard and soft IP cores. The key features of CoreConnect are the Processor Local Bus (PLB), On-chip Peripheral Bus (OPB) and Device Control Register (DCR) Bus.

Virtex-II Pro is configured by delivering the bitstream through one of the configuration interfaces (JTAG, SelectMAP, or Slave/Master Serial). Configuration memory is arranged in a rectangular array of bits. One-bit wide vertical frames are the smallest addressable segments of the Virtex-II Pro configuration memory space [11]. Data is loaded on a column-basis and each column contains the number of frames dependent on the specific FPGA device. The Virtex-II Pro configuration control logic consists of a packet processor, a set of registers, and global signals that are controlled by the configuration registers. The packet processor controls the flow of data from the configuration interface to the appropriate register. The registers control all other aspects of configuration.

The Virtex-II Pro configuration architecture features an Internal Configuration Access Port (ICAP) that provides the user logic with access to FPGA configuration interface and therefore access to memory bits of configuration memory [11]. The interface is similar to SelectMAP interface but it cannot be used for full configuration. With no handshaking mechanism ICAP interface can be clocked up to the maximum frequency of 66MHz [4]. In Virtex-4 devices this interface can be used for readback and reconfiguration when the device is initially configured with an encrypted bitstream.

In active partial reconfiguration, new data can be loaded through ICAP to dynamically reconfigure a particular area of FPGA while the rest of the FPGA is still operational. Xilinx introduces two design flows for active partial reconfiguration [12]:

1) Module-Based: This flow is suitable for partially reconfiguring a large portion of the design and is based on the Xilinx Modular Design methodology [13]. A special bus macro is required for inter-module communications. It is used to establish unchanging routing channels between modules.

Bus macros use fixed routing resources and are currently implemented using 3-state buffers associated with dedicated segmentable horizontal routing resources. Xilinx will introduce a new implementation with the release of ISE 8.1 since Virtex-4 devices do not contain 3-state buffers (TBUF). **Figure 2** shows a bus macro used for inter-module communication and its implementation with 3-state buffers.

Reconfigurable modules in the design need to have specific properties in terms of their width, height, and placement. Available resources are also limited only to those encompassed by the width of the module and communications with other modules (both fixed and reconfigurable) should take place through bus macros.

HDL coding and synthesis process follow some general guidelines in terms of the structure of top-level design, instantiation of bus macros, shared signals, and synthesis attributes.

The implementation flow takes place in three phases after the design entry. In initial budgeting phase, the design is floor planned and constrained based on the properties of each module. The result is a file with extension '.ucf' that is used for active implementation phase. This phase places and routes each module separately in the context of the top-level logic and constraints. The final assembly phase uses all placed and routed modules generated from the previous phase to combine them into a complete FPGA design. To maintain the performance of each module, placement and routing for each module are preserved.

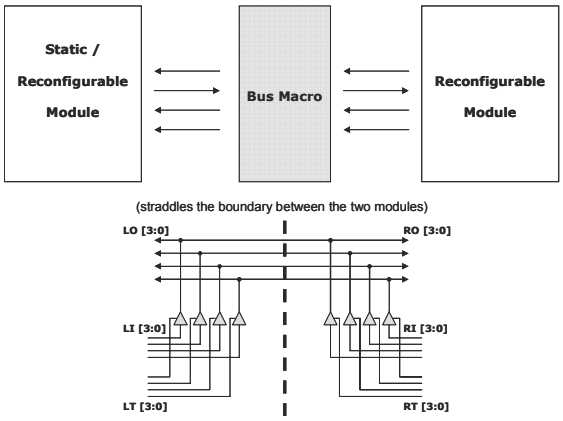


Figure 2. Physical Implementation of a 4-bit Bus Macro by Xilinx

At present, bitstreams generated for the full design require that the initial bitstream includes at least one variation of any partially reconfigurable module. This means that the initial bitstream should be a complete design since all global resources such as clocking logic need to be placed and properly constrained. Bitstream frames for clocks are separate from other frames. This imposes a limit in which a completely separate module cannot be added to an initial design with module-based partial reconfiguration flow.

2) Difference-Based: Using this flow the design can change either at the front-end or the back-end. For changes in HDL code or schematics at the front-end, the design must be re-synthesized and re-implemented, while for back-end changes the FPGA Editor tool can be used to modify sections of the design. Many different types of changes can be made using this tool, including routing information, LUT programming, changing BRAM contents and I/O standards.

The bitstream generator BitGen, used with the proper options setting, can create a partial bitstream that contains only the difference between the modified design and the initial bitstream. In other words, BitGen produces a partial bitstream that only configures the frames that are different between the two designs. The produced partial bitstream is small and quick to load. A partial bitstream can be loaded only after the device power up and loading an initial bitstream. The design must take into account the transition time of the reconfigurable module(s) and other modules, and should not rely on the state of the signals connected to the reconfigurable module.

3. Xilinx Embedded Development Kit and ML310 Evaluation Board

Multiple embedded software tools, PowerPC and MicroBlaze infrastructure, and peripheral IP cores

included in Xilinx Embedded Development Kit (EDK) provide a framework for design of hardware/software components of the embedded processor systems on programmable logic [14]. Utilizing the appropriate tool for each stage of the design facilitates hardware/software partitioning, design reuse, and shorter time-to-market.

Embedded system tools in EDK consist of Xilinx Platform Studio (XPS), GNU software development tools, hardware/software development tools, board support packages, and embedded operating systems.

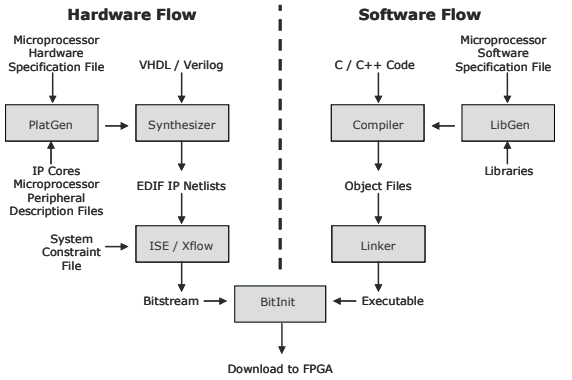


Figure 3. EDK Tools Flow

Figure 3 presents an overview of the tools flow.

In a typical design of an embedded processor system, the first step is to create a hardware platform followed by the creation of software platform and optionally verification platform. XPS IDE with its underlying tools integrates all the processes from design entry to design debug and verification [15].

Creating a basic hardware system involves assembling a system containing processor, buses, and peripherals, generating an HDL netlist, and implementing the design using ISE implementation tools to generate a bitstream.

Creating the software platform involves building libraries, compiling C applications, initializing bitstreams with the application, downloading applications onto external memories, and debugging applications using debugger. XPS calls GNU compiler tools provided for both hard and soft processors for compiling and linking user application executables. The Bitstream Initializer (BitInit) tool can then initialize the bitstream with the executable in the instruction memory of processors on the FPGA. The bitstream can be downloaded using Xilinx Microprocessor Debugger (XMD), bootloader programs, or System ACE controller. XMD is the underlying engine to communicate to processor targets and provides an interface for both hardware system debug and software running on hardware. This tool with GNU debugger is used for software debugging.

The ML310 Embedded Development Platform is a Virtex-II Pro based platform suitable for rapid prototyping and system verification. The main features of ML310 include: 256 MB DDR DIMM, System ACE Compact Flash controller, FPGA UART, General Purpose IO (LEDs/LCD), PCI bus interface, and high speed I/O through RocketIO Multi-Gigabit Transceivers (MGTs). MGT blocks available in the Virtex-II Pro create high-speed serial links between devices and the FPGA. The high-speed I/O signals on the FPGA are accessible through two personality module (PM) connectors on the ML310 board. The majority of the ML310 features are accessed over the 33 MHz/32-bit PCI bus which is connected to fixed PCI devices such as Intel 10/100 PCI Ethernet NIC, ALi PCI South Bridge. The ALi South Bridge augments the ML310 with many of the basic features found on legacy PCs. The main system clock of ML310 is a 100 MHz oscillator. The FPGA generates and drives clocks required by the DDR DIMM memory and PCI bus interfaces. **Figure 4** shows the block diagram of the ML310 board.

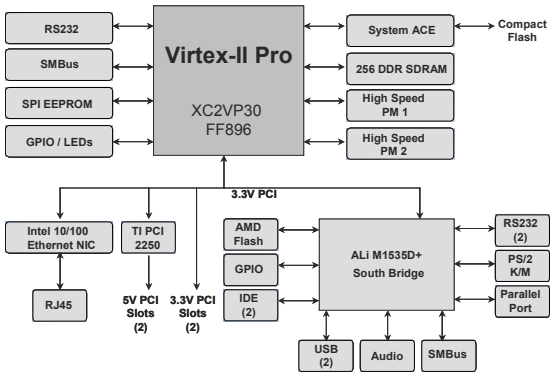


Figure 4. Block Diagram of Xilinx ML310 Embedded Development Board

4. Implemented Self-reconfiguring Systems

Figure 5 and **6** show the hardware components of the constructed self-reconfiguring platforms utilizing both embedded PowerPC and MicroBlaze soft processor cores. No cache memory is selected for the microprocessors, and both systems run at 100MHz (including PLB and OPB buses).

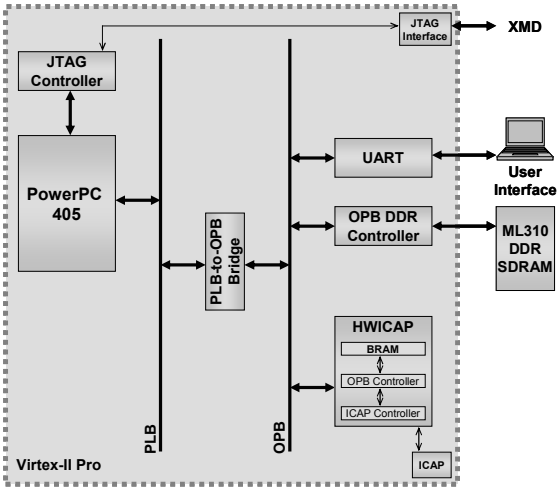
Both embedded PowerPC and MicroBlaze processor cores communicate with peripherals through one or more of the IBM CoreConnect buses, which enables compliant IP cores to integrate with embedded processor cores. PowerPC only has the PLB bus interface, and therefore OPB devices cannot directly connect to the processor. Therefore, processor and peripherals communicate over the OPB connected to the PLB through PLB-to-OPB bridge. The MicroBlaze system is configured with OPB bus and two Local Memory Buses (LMBs). The LMB is a fast and efficient local bus that connects

MicroBlaze instruction and data ports to high-speed peripherals, primarily BRAMs. Both instruction-side and data-side LMBs are connected to the same dual-port BRAM using different ports of the BRAM.

Both implemented systems require the OPB bus to instantiate the HWICAP module [16] since the current implementation of this module only connects to OPB. This module is used for reconfiguration. It enables the microprocessors to read and write the FPGA configuration memory, as well as loading partial bitstreams from system memory through ICAP. The HWICAP core consists of OPB controller, ICAP controller, and a BRAM. It uses the BRAM on OPB bus as a configuration cache and has the capability to transfer the partial bitstream from local memory to ICAP. The partial bitstream is transferred frame by frame to this BRAM and then to ICAP. The HWICAP ICAP controller connects to the ICAP block located in the lower-right corner of the logic array. ICAP interface operates at the clock rate of OPB bus.

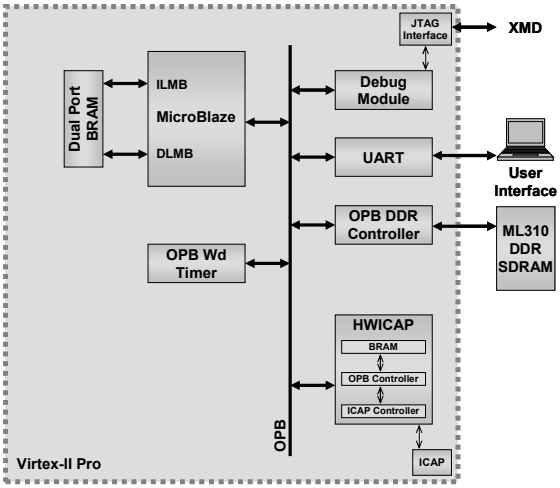
The DDR memory available on the board was selected as the external memory for storage of the partial bitstream, and therefore OPB DDR SDRAM Controller was used for both systems. JTAG port was used for both transferring the partial bitstream to DDR memory and debugging. PowerPC system requires a JTAG controller that allows the PowerPC to connect to the JTAG chain of the FPGA instantiating a JTAGPPC primitive, and directly connecting it to both PowerPC CPUs in the chip. MicroBlaze system requires a Microprocessor Debug Module on the processor OPB bus for JTAG-based debugging. This module can also be used with PowerPC405 processors. UART and additional features are not the essential parts of the self-reconfiguring systems but provide ease of use for user application. The systems were implemented in a XC2VP30 Virtex-II Pro FPGA device on the ML310 Evaluation Board with minimal footprint.

EDK automatically generated the memory map of the hardware platform as well as assigned default drivers to the processors and each of the peripherals. The program running on the processor cores was written in C to perform the following tasks: authentication, decryption, and configuration. Software cores were used for authentication and decryption. AES was used for encryption/decryption and HMAC-SHA1 was used as the authentication algorithm. Both cores were freely available implementations developed by Dr. B. Gladman [18]. These cores were ported to EDK environment so that they can be used as libraries available for the program. ICAP API [17] was used for transferring the data between the external configuration memory and HWICAP BRAM configuration cache. The ICAP API defines methods for accessing the configuration logic through ICAP port.



PLB = Processor Local Bus, OPB = On-chip Peripheral Bus, XMD = Xilinx Microprocessor Debugger, HWICAP = Hardware Internal Configuration Access Port, DDR = Double Data Rate.

Figure 5. PowerPC System



ILMB = Instruction-side Local Memory Bus, DLMB = Data-side Local Memory Bus, OPB Wd Timer = OPB Watchdog Timer, UART = Universal Asynchronous Receiver Transmitter.

Figure 6. MicroBlaze System

5. Experiment Methodology

The considered scenario for the experiment is as follows. The self-reconfiguring system reads an authenticated and encrypted partial bitstream stored in an external memory. It then verifies the authenticated partial bitstream with the stored MAC value. If the authentication is successful it decrypts the partial bitstream using the stored key and configures the device using ICAP.

To perform the experiment using the implemented self-reconfiguring systems the first step was generating a partial bitstream. The difference-based method was selected for this purpose since the module-based flow requirements were problematic

while using the ML310 board especially with regards to board pin assignments and requirements of the reconfigurable module mentioned previously.

An additional application MicroBlaze system was implemented in the form of a microcontroller as the target of partial reconfiguration. The application design only included MicroBlaze processor, buses, 8K of block RAM memory, and GPIO connected to the 8-bit LED display on the ML310 board. Using EDK this system was combined separately with each of the self-reconfiguring systems. **Figure 7** shows a simplified version of the FPGA layout.

The program running on this system was generating a pattern on LEDs. The partial bitstream changes the BRAM contents where the program running on the MicroBlaze system had been stored. That resulted in a different pattern to appear on LEDs.

FPGA Editor tool was used to modify the BRAM contents. The modified design file must be used with the initial bitstream for creating a difference-based partial bitstream. The initial bitstream of the design contained the original BRAM contents. The partial bitstream was created with BitGen program using the -r switch. BitGen set with this switch produced a bitstream that contained only the differences between the modified design file and the initial bit file. The generated bitstream (14 KB) was much smaller than the initial bitstream (1.38 MB).

After encrypting and signing the partial bitstream, the initial FPGA bitstream was downloaded into the JTAG port of the FPGA on ML310 Evaluation Board. Then Xilinx Microprocessor Debugger was used to download the partial bitstream from the host machine (connected to the board) to an address range not used by the program in DDR memory on the board. The program running on the self-reconfiguring system successfully authenticated the partial bitstream with the stored MAC value; decryption phase would not start if the generated MAC was different than the MAC stored in the program. The program then decrypted the encrypted partial bitstream using the stored key, and dynamically partially reconfigured the other active system on FPGA.

The experiment was judged to be successful when the new pattern was displayed on the LEDs of the board. It was verifying that the new application had correctly replaced the initial program stored in the internal BRAMs of the MicroBlaze system. Section 6 presents the timing results obtained for execution of each phase of the program along with the device resource utilization summary.

We also implemented a reconfigurable design using the module-based flow. Similar to the layout shown in **Figure 7**, we partitioned the top-level design into three modules. The static module contained the configuration controller system and the reconfigurable module included the application

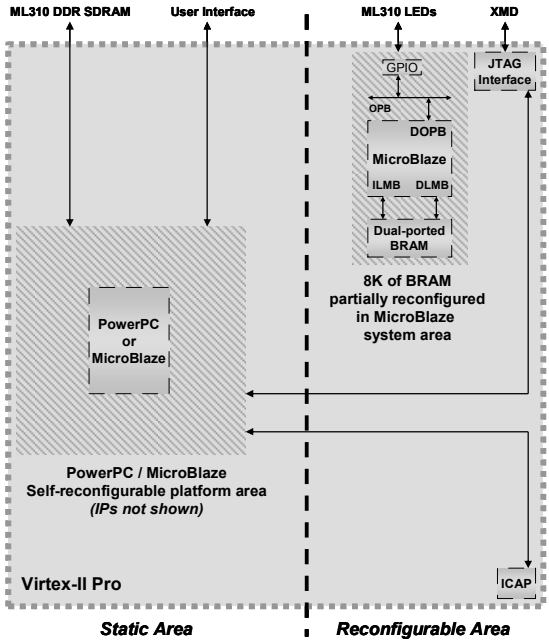


Figure 7. Simplified Layout of the Experiment Design

system. The third module was a wrapper used for instantiation of ICAP and JTAG blocks.

The configuration controller used many pins and resources of the FPGA scattered all around the chip. By placing the static module on the left side of the chip, we minimized the number of signals that passed through the reconfigurable module boundary using bus macros. A custom bus macro that spanned the full width of the reconfigurable module was used for signals connected to ICAP block to keep these signals active during reconfiguration. The generated partial bitstream for the reconfigurable boundary of 36 slice columns had a size of 321 KB. We were unable to validate the functionality of this partial bitstream because of the problems we faced in implementing another full design with an alternate version of the reconfigurable module. The problems were encountered during the final assembly phase of the implementation in the form of an undocumented error generated by the tools.

5.1. Security Analysis

One of the main advantages of using the self-reconfiguring systems is the increase of flexibility. The designer is able to partition the application according to the necessary security level and choose the suitable algorithms for the authentication and decryption. Moreover these algorithms can be upgraded to take advantage of the latest improvements of the security field without any change in the implemented partially reconfigurable design. The following considerations should be taken into account to improve the security of the scheme:

1) Partial Bitstream Storage: In our current design, authenticated and decrypted bitstream is stored in external memory before the partial reconfiguration. Storing the partial bitstream in internal memory would prevent the interception of the bitstream after authentication and decryption. The program running on the processor core should be modified in such a way that one segment of the partial bitstream (of the size of an internal block RAM) is authenticated, decrypted, and sent to ICAP at a time.

2) Key Storage: In the current design, decryption and authentication keys are embedded in the program running on the embedded microprocessor. Storing the key in a battery-powered storage or providing the key interactively by a user are among the options that can be used to increase the security of the scheme.

6. Results

1) Timing Measurements: In **Table 1** the timing result of each phase for both systems is provided. For each phase of the process (authentication, decryption, and configuration) 10 measurements were done by obtaining the number of clock cycles required for each processor to execute the functions. For PowerPC system no extra component was needed since a time-base register inside the processor is available that works with the system clock. For MicroBlaze system a watch-dog timer on OPB was used that contains a time-base register. For both systems, standard deviation from the mean value at each phase along with the percentage error is also shown in **Table 1**.

Table 2 summarizes the comparison of the results for the average values and throughput. The average values of the obtained results show that PowerPC system performed faster in both authentication and decryption phases of the application. Consequently it has higher throughput in these two phases with the ratios shown in the table. Even though both systems were running at 100 MHz, the better performance of the PowerPC system could be due to the fact that its instruction set executes most of the instructions in a single cycle and is more efficient than MicroBlaze. On the other hand MicroBlaze system gives a better performance working with the HWICAP module and therefore it achieves a higher throughput for configuration. The reason might be the presence of the extra bus (PLB) and PLB-to-OPB bridge in the PowerPC system. Since HWICAP module is a slave on the OPB bus the processor should transfer the frames of the bitstream from the DDR to the HWICAP BRAM and therefore an extra bus may actually increase the time of this transfer. Thus, DMA data transfer is desirable to increase the performance of HWICAP in any system.

Table 2 also provides the time based on the unit of operation for each phase. Authentication algorithm works on bytes with a total number of 14112 bytes in the partial bitstream. Decryption works on blocks of 16 bytes in CTR mode. There were 882 blocks in the partial bitstream. Also, 32-bits words are sent to ICAP for reconfiguration with the total number of 3528.

Table 1. Timing Results for Each Phase (Clock Cycles)

PowerPC System

Phase #	Authentication	Decryption	Configuration
1	13,862,435	20,838,769	5,630,038
2	13,862,591	20,838,876	5,631,061
3	13,862,486	20,838,769	5,630,038
4	13,862,435	20,838,769	5,630,038
5	13,862,500	20,838,769	5,631,037
6	13,862,575	20,838,776	5,630,038
7	13,862,591	20,838,876	5,628,993
8	13,862,575	20,838,776	5,630,038
9	13,862,591	20,838,879	5,628,993
10	13,862,486	20,838,769	5,631,037
Std. Dev.	65	51	756
Mean	13,862,527	20,838,803	5,630,131
% Error	0.05%	0.02%	1.34%

MicroBlaze System

Phase #	Authentication	Decryption	Configuration
1	77,649,436	147,201,543	3,175,996
2	77,649,453	147,201,601	3,175,964
3	77,649,510	147,201,675	3,175,420
4	77,649,416	147,201,543	3,175,996
5	77,649,510	147,201,675	3,175,943
6	77,649,349	147,201,675	3,175,996
7	77,649,597	147,201,639	3,175,996
8	77,649,597	147,201,675	3,175,952
9	77,649,515	147,201,451	3,176,008
10	77,648,899	147,201,639	3,175,996
Std. Dev.	201	77	179
Mean	77,649,428	147,201,612	3,175,927
% Error	0.03%	0.01%	0.56%

Table 2. Comparison of the Timing Results for Each Phase

System		Authentication	Decryption	Configuration
Ave. Time (ms)	PowerPC	139	208	56
	MicroBlaze	776	1472	32
Throughput (KB/s)	PowerPC	102	68	251
	MicroBlaze	18	10	444
Ratio	PPC / MB	5.6	7.0	0.5

System	Clock Cycles / Byte	Clock Cycles / 16 Bytes Block	Clock Cycles / 4 Bytes Word
PowerPC	982	23,627	1,596
MicroBlaze	5,502	166,895	900

2) Resource Utilization Summary: Systems were designed with only the required components. It should be noted that the Xilinx MicroBlaze soft processor uses ~950 logic cells (475 Slices) in the Virtex-II Pro device but PowerPC cores are part of the FPGA fabric with no resource usage even though hard core processors in the FPGA fabric reduce the available area for logic in general.

In **Table 3** a summary of the device utilization is provided. The resource utilization is only for the configuration controllers and not the additional application system under reconfiguration. The device utilization is close for both systems. The PowerPC system used lesser amount of resources even though it required the use of extra bus and bridge but it

Table 3. Device Utilization Summary and Resource Usage of IP Cores

Device Resources	Number of Resources				
	Used by PowerPC		Used by MicroBlaze		Available in the Device
	Total	%	Total	%	
SLICEs	1,334	9	1,706	12	13,696
RAMB16s	5	3	5	3	136
MULT18X18s	0	0	3	2	136
BUFGMUXs	7	43	8	50	16
DCMs	2	25	2	25	8
JTAGPPCs	1	100	1	100	1
ICAPs	1	100	1	100	1
PPC405s	1	50	0	0	2

System Component	Resources Used					
	Slices		LUTs		FFs	
	Min	Max	Min	Max	Min	Max
Required for Both Systems						
OPB (On-Chip Peripheral Bus)	46	436	81	668	5	145
OPB HWICAP	120	128	213	224	152	155
OPB BRAM Controller	25	34	16	30	33	55
OPB DDR SDRAM Controller	332	563	353	637	314	444
Total	523	1161	590	1559	504	799
Required for PowerPC System						
PLB (Processor Local Bus)	223	1645	270	2540	59	484
PLB to OPB Bridge	595	836	535	823	547	812
Processor System Reset Module	N/A	N/A	37	57	52	82
PowerPC (Wrapper)	0	0	0	0	0	77
Total	818	2481	842	3420	658	1455
Required for MicroBlaze System						
2 x LMB (Local Memory Bus)	N/A	N/A	0	353	0	0
2 x LMB BRAM Controller	N/A	N/A	6	6	2	2
Total	-	-	6	359	2	2
Additional Features						
OPB UART Lite	N/A	N/A	88	108	48	57
OPB Timebase WDT	N/A	N/A	63	63	111	111
Microprocessor Debug Module	67	188	45	292	79	204
Total	67	188	196	463	238	372

should be considered that the resource usage for the MicroBlaze system includes the soft processor as well.

Table 3 also shows the contribution of different IP cores. The required IPs for both systems are listed on the top section of the table followed by the necessary IPs for PowerPC system and MicroBlaze system. Non-essential IPs are provided subsequently.

7. Conclusions

In this paper, we presented the implementation of a self-reconfiguring platform capable of performing secure partial reconfiguration of Xilinx FPGAs using ICAP and embedded processor cores. An application has been developed to demonstrate that FPGA can be reconfigured with an encrypted partial bitstream stored in an external memory using software cores for authentication and decryption. Improving the ICAP control logic from software to hardware planned by Xilinx will also enhance the performance of self-reconfiguring platforms since there will be less communication over the system bus and less processor involvement. An embedded OS can also facilitate the process.

Furthermore, a partial bitstream has been generated using the difference-based flow targeting an active system placed in FPGA besides the self-reconfigurable platform. Even though the difference-based flow involved none of the difficulties and restrictions of module-based flow it is not suitable for large designs where large blocks of logic are under reconfiguration. To increase the ease of use for designers and decrease the development time a simple methodology along with more support and automation from tools are needed for implementation of a partially reconfigurable design using module-based flow.

8. References

[1] S. Wong, S. Vassiliadis, and S. Cotofana, "Future directions of (programmable and reconfigurable) embedded processors," in Embedded Processor Design Challenges, Workshop on Systems, Architectures, Modeling, and Simulation—SAMOS 2002.

[2] T Wollinger, J. Guajardo, C. Paar, "Security on FPGAs: State-of-the-Art Implementations and Attacks," in ACM Transactions on Embedded Computing Systems, Vol. 3, No. 3, August 2004, Pages 534–574.

[3] Xilinx, Inc. web site. http://www.xilinx.com/.

[4] R. Krueger, "Using High Security Features in Virtex-II Series FPGAs". Xilinx Application Note XAPP766, version 1.0, Xilinx, Inc. July 2004.

[5] B. Blodget, P. James-Roxby, E. Keller, S. McMillian, and P. Sundararajan. A Self-reconfiguring Platform. In Proceedings of the International Conference on Field Programmable Logic, Lisbon, Portugal, Sept. 2003.

[6] T. Kean. Secure Configuration of Field Programmable Gate Arrays. In proceeding of 11th International Conference on Field-Programmable Logic and Applications, FPL'2001. Belfast, United Kingdom.

[7] L. Bossuet, G. Gogniat, and W. Burleson. Dynamically Configurable Security for SRAM FPGA Bitstreams. In proceeding of 11th Reconfigurable Architectures Workshop, RAW 2004. Santa Fé, USA.

[8] "Virtex-II Platform FPGA Handbook", version 2.0, Xilinx, Inc., 2004

[9] "PowerPC Processor Reference Guide", version 2.0, Xilinx, Inc., 2003

[10] IBM web site http://www.chips.ibm.com/products/coreconnect. 2003

[11] "Virtex-II Platform FPGA User Guide", version 4.0, Xilinx, Inc., 2005

[12] "Two flows for partial reconfiguration: Module based or Difference Based". Xilinx Application Note XAPP290, version 1.2, Xilinx, Inc., 2004

[13] "Development System Reference Guide", Xilinx, Inc., 2005

[14] "Embedded System Tools Reference Manual", version 3.0, Xilinx, Inc., 2004

[15] "Platform Studio User Guide", version 3.0, Xilinx, Inc., 2004

[16] "Processor IP Reference Guide", Xilinx, Inc., 2004

[17] "EDK OS and Libraries Reference Manual", version 3.0, Xilinx, Inc., 2004

[18] Dr. B. Gladman, Cryptographic Implementations, web site http://fp.gladman.plus.com/cryptography_technology/index.htm