Complexity and Performance Evaluation of Two Partial Reconfiguration Interfaces on FPGAs: a Case Study[1]

Heng Tan, Ronald F. DeMara, and Anuja J. Thakkar
ECE Department
University of Central Florida
Orlando, FL 32816-2450
demara@mail.ucf.edu

Abdel Ejnioui
Information Technology Dept.
University of South Florida
Lakeland, FL 33803-9807
aejnioui@lakeland.usf.edu

Jason D. Sattler
Space Photonics Inc.
700 Research Center Blvd.
Fayetteville, AR 72701
jsattler@spacephotonics.com

Abstract

Two different interfaces, namely the JTAG and SelectMAP, have become the major interfaces proposed for controlling and managing partial reconfiguration of SRAM-based Field Programmable Gate Arrays (FGPAs). Each of these interfaces exhibit distinct features in terms of hardware overhead and software support. In this paper, different strategies for design and verification are compared and contrasted based on these features. Experiments are conducted in order to quantitatively evaluate the tradeoffs between design complexity and area overhead, reconfiguration flexibility, and reconfiguration latency of these two interfaces. The results show that the SelectMAP interface is highly suitable when reconfiguration latency needs to be kept to a minimum at the expense of a large area overhead and more rigid area control. On the other hand, the JTAG interface is preferable if significant control over the placement of reconfigurable modules in the reconfigurable fabric of the FPGA chip is desired and/or more flexible area management is required. The JTAG design consumes a factor of 3 to 7 times fewer logic resources and a third of the device pins, but can incur reconfiguration latency up to 40 times longer than the SelectMAP design.

1 Introduction

FPGAs have evolved from simple Programmable Logic Devices (PLD) to fully integrated System On Chips (SOCs) containing microprocessors, embedded memory, and optimized datapaths connected to a high capacity reconfigurable fabric. As a case in point, the high-end Virtex FPGAs offered by Xilinx contains more than multi-million gate equivalent reconfigurable fabric in which several PowerPC processors, a number of RAM blocks, and dedicated multipliers are embedded.

In particular, one of the major benefits provided by FPGAs is dynamic reconfiguration ability, which involves altering the programmed design within a SRAM-based Field Programmable Gate Array (FPGA) at run-time [3]. Recently, many applications in digital signal processing, multimedia, cryptography, fluid dynamics, and evolutionary systems, increasingly require partial reconfiguration as an integral component of their execution environment for performance and efficiency purposes. As more applications can benefit from runtime dynamic reconfiguration, the combining of basic partial reconfiguration techniques with high-level algorithm designs becomes an important issue to be discussed and evaluated. In particular, this paper addresses basic design considerations when partial reconfiguration techniques are integrated into a reconfigurable system using two major interfaces respectively, namely the JTAG and SelectMAP interfaces. Spatial and temporal performances of each interface are also quantitatively evaluated herein.

This paper is organized as follows. In Section 2, the basic partial reconfiguration technique from Xilinx is overviewed. In Section 3 and Section 4, the design considerations and testing strategy with JTAG and SelectMAP interfaces are presented, respectively. Section 5 describes the obtained results and their evaluation while Section 6 concludes the paper.

2 Partial Reconfiguration in Xilinx Families

Currently, the most widely used Xilinx FPGA chips with partial reconfiguration capability are Virtex II and Virtex Pro family. For these FPGA architectures, Xilinx has proposed two standard flows for partial reconfiguration process: Difference-based flow and Module-based flow [7].

[1] This project supported in-part by USAF under contract FA8651-05-C-0221. AAC/PA 01-11-06-014, ref. no. 1267.

With a Difference-based flow, the designer must manually edit a design with low-level changes. Using a low-level editing tool, such as the FPGA Editor, small changes can be made to different components, such as lookup tabled, flip-flops, and I/O pins. After the changes are completed, the partial bitstream, which contains information only regarding modifications, is generated and stored in a file.

For the Module-based flow, the full design is partitioned into modules, some of which can be fixed while others can be reconfigurable. The reconfigurable fabric of the FPGA is partitioned into column-based rectangular regions in which the fixed and reconfigurable modules will be arranged based on specified area constrains. A *bus macro* can be used to maintain correct connections between the modules by spanning the boundaries of these rectangular regions. Fig. 1 shows the basic concept of this reconfiguration flow methodology.

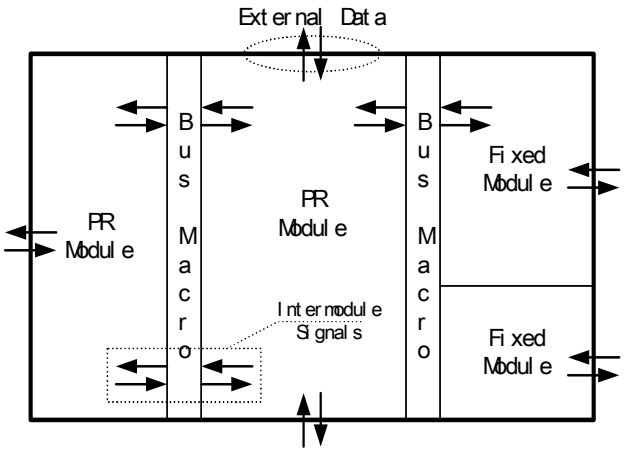

Figure 1. Design Layout with Two Reconfigurable Modules

Although the Module-based flow still involves Xilinx tools to place-and-route designs, and generate the partial reconfiguration files, these operations can be carried out in the background simply by using scripts instead of launching GUI-based tools or other forms of manual modification. This makes the Module-based flow suitable for full automation making it highly flexible than the Difference-based flow, particularly when it is considered for integration with high-level user applications. As a result, this paper chooses the Module-based flow as the primary partial reconfiguration technique for the design in the remainder of the paper's discussion.

When integrating the basic partial reconfiguration flow with high-level algorithms, several basic questions need to be considered:

- Which hardware interface should be used for the reconfiguration process?
- What fixed hardware components are required to support control logic?
- How can a communication channel be established between the low-level hardware components and the high-level applications through the chosen interface?
- How can applications be tested, particularly after the partial reconfigurations? Especially, which strategies and tools can be used in this verification process?

3 Design with SelectMAP Interface

3.1 Select MAP Interface

The SelectMAP interface provides an eight-bit bidirectional data bus interface to the FPGA configuration logic [6]. It can be used either in Master Mode with the CCLK signal considered as an output from the FPGA, or in Slave mode with the CCLK signal considered as an input. In slave mode, SelectMAP allows for both configuration and readback, while in master mode only configuration is possible.

In SelectMAP slave mode, 14 pins, including DATA pins (`D0:D7`), `CCLK`, `RDWR_B`, `BUSY`, `CS_B`, `PROG_B`, `DONE`, and `INIT_B`, are required to carry out the reconfiguration process. These pins will need to be placed in specific positions of the general-purpose I/O blocks within the FPGA by specifying user constrains.

Multiple devices can be connected on the same SelectMAP bus. To do so, the DATA pins (`D0:D7`), `CCLK`, `RDWR_B`, `BUSY`, `PROG_B`, `DONE`, and `INIT_B` are connected in common between all devices. The `CS_B` Chip Select inputs are kept separate, so that each device can be accessed individually. External control logic is required to arbitrate between devices by asserting and de-asserting the `CS_B` signals as necessary.

In this paper, all the designs are based on a Loosely Coupled architecture in which all the high-level control logic originates from an external PC host. Only one FPGA device is connected to the reconfiguration interface. The clock is setup to be generated from outside the FPGA itself while readback may be required for user verification purposes. Based on these considerations, the SelectMAP interface will be running in slave mode in the experiments described in this paper.

3.2 Prototype Hardware Design

In order to communicate with the host PC through the SelectMAP interface in slave mode, a PCI bus interface is normally used to establish the connection. This requires the instantiation of a PCI core either inside the target FPGA chip or inside a dedicated bridge chip on the FPGA board. To complete this setup, SRAM modules may be needed as buffers for the purpose of receiving data from the host PC and reconfiguring the target FPGA.

Figure 2 shows the detailed FPGA hardware component design for the Loosely Coupled prototype with the SelectMAP interface developed for Xilinx Virtex II Pro platforms. This solution was based on two Commercial Off The Shelf (COTS) independent hardware units: the Avnet Virtex II Pro development board and a Pentium Host PC. The development board is equipped with the Virtex II Pro XC2VP7 chip. The latter contains one PowerPC core,

The fixed operational resources includes multiple interfaces to control the external on-board SRAM modules, the RS232 resources and the memory controllers supporting the PowerPC core in accessing the Block RAMs. In order to connect the PowerPC core with all the heterogeneous FPGA resources inside the chip, a three-segment path is also established in the fixed operational resources region, which includes an On-chip Peripheral

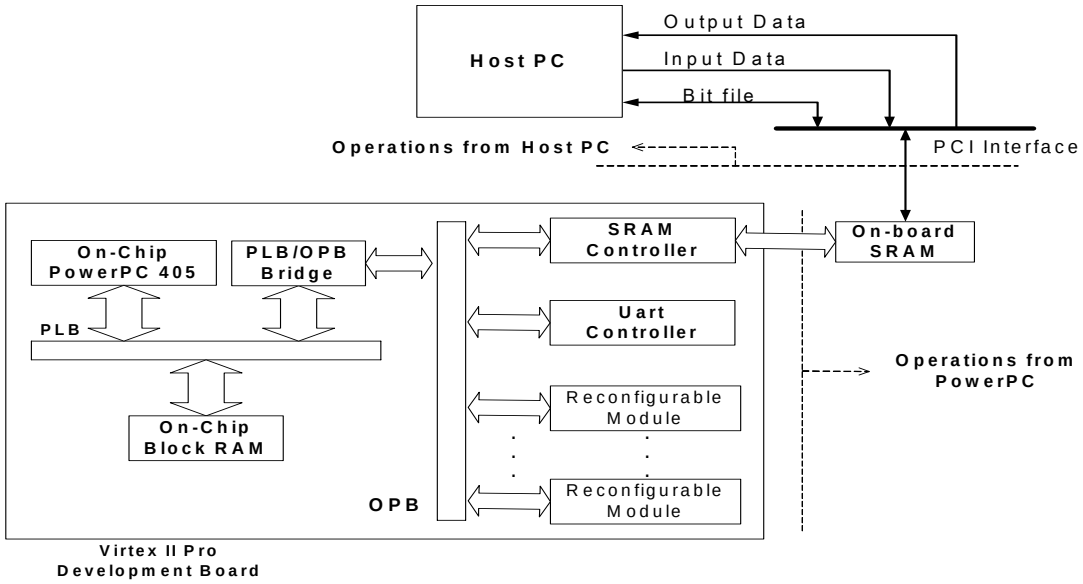

Figure 2: Loosely-Coupled System using SelectMAP Control

which can operate at the clock frequency of 300 MHz. The board communicates with the PC through the standard PCI interface. The 8MB SRAM provided is used as a buffer for the configuration process through the SelectMAP interface in slave mode.

This prototype hardware subsystem developed includes two resource subsets, *system resources* and *operational resources* [1]. The system resources are composed of an on-chip PowerPC 405 CPU core, on-chip Block RAMs, and all the onboard peripherals such as the SRAM modules and RS232 interface. Whereas the SRAM modules can be used as a shared memory by the host PC and the on-chip PowerPC, the RS232 interface can be used for monitoring and debugging purposes. The operational resources consist of the actual modules instantiated inside the Virtex II Pro chip. These modules consist of a *fixed subset* and a *reconfigurable subset*. The modules in the former are kept unchanged during the partial reconfiguration process, and are used to control the on-chip data communications and on-board peripherals. On the other hand, the modules in the latter are used for user-defined partial reconfiguration applications. These modules are all initially developed by Xilinx ISE 6.3i and Xilinx EDK 6.3i toolsets.

Bus (OPB), a Processor Local Bus (PLB) and a Bridge Core (PLB2OPB) providing access to the OPB from the PLB.

All the remaining resources in the fabric comprise the reconfigurable subset. In order to maintain correct communications with PowerPC through the OPB bus with those reconfigurable and relocated modules, simplified *Intellectual Property Interface (IPIF)* [8] and *Bus Macro* structures [7] have been adopted. This setup permits seamless control of the communication of all the modules.

3.3 Testing Strategy and Communication Channel

Because there are few sophisticated testing tools available for testing the SelectMAP interface at the present time, thus the communication protocol and testing APIs for this prototype configuration are designed and built from the ground up.

For testing purposes, the communication channel needs to be bi-directional. The on-chip PowerPC is used to control the handshaking between the board and the host PC. To monitor the data exchange and reconfiguration process, the on-board RS232 is subsequently used to connect to the serial port of the host PC. After the board is

installed in a PCI slot of the host PC, it is then controlled by the debugging program running on the host PC [4].

As shown in Figure 3, the input path begins at the PC. The PC sends the data over the PCI bus to the specific address of an on-board SRAM module, and then sets the data availability flag. At the same time, the PowerPC core keeps checking the flag, reads data from the addressed SRAM module, and sends data to the hardware component for processing when the data flag is set. On the other hand, the output path data starts from the PowerPC when the latter reads the data from other hardware modules through the 3-segment bus path, and then writes it to the on-board SRAM by setting the data availability flag. In the meantime, the PC host keeps polling SRAM location for the data availability flag. When the flag is set, the PC host receives the data over the PCI bus, and saves it in its memory for further processing when ready. This bi-directional path is supported by the on-board status register, which maintains the PCI ownership control bit. The combination of the drivers running on the PC and the control program running on the PowerPC in the Virtex-II Pro fabric achieve PCI ownership of the SRAM bus to retrieve the data. After the initial boot up, the PCI ownership is controlled by the host PC until it clears the PCI ownership control bit of the status register. Thus, the PowerPC and the host PC are able to synchronize the exchange of data between each other by polling the data status flag inside the SRAM.

Table I: Communication APIs on PowerPC

API name	Host	Operation	Data Width
Initial	PC	Initializes the board	N/A
ReadBitFile	PC	Readback the bit file	File length
WriteBitFile	PC	Reconfigure the board	File length
DWordRead	PC	Reads data from the board to PC	32 bits
DWordWrite	PC	Writes data to the board from PC	32 bits
mem_dump	PowerPC	Reads the on board and on-chip memory&register	32 bits
mem_write	PowerPC	Writes to the on board and on-chip memory&register	32 bits
mem_test	PowerPC	Thorough testing on the SRAM	32 bits

The communication API, shown in Table I, has been implemented in the C programming language. This API can be divided into 3 categories: *Initialization Operations*, *Configuration File Operations*, and *On-board Memory Operations*. For the on-board memory operations, a 32-bit data width is provided as determined by the on-chip bus width. The Windriver routines [2] from Jungo Software for the PCI interface can realize part of this API functionality.

Currently, the PowerPC embedded in the FPGA fabric is running in standalone mode by automatically generating the software structure using Xilinx EDK 6.3. On top of this basic operating system, the standard communication API interface is then developed.

4 Design with JTAG Interface

4.1 JTAG Interface

The IEEE 1149.1 Test Access Port and Boundary-Scan Architecture is commonly referred to as the *Joint Test Action Group (JTAG)* interface, which is the technical subcommittee initially responsible for developing the standard. This standard provides a means to assure the integrity of individual components and the interconnections between them at the board level.

Devices containing JTAG boundary-scan logic can send data out on I/O pins in order to test connections between devices at the board level [6]. The circuitry can also be used to send signals internally to test the device-specific behavior. These tests are commonly used to detect opens and shorts at both the board and device level. In addition to testing, boundary-scan offers the flexibility for a device to have its own set of user-defined instructions. The added common vendor specific instructions, such as configure and verify, have increased the popularity of boundary-scan testing and functionality.

JTAG is also well-suited for downloading configuration bitstreams to FPGAs. The IEEE 1149.1 standard defines a four-wire serial interface (a fifth wire is optional) as designated in the Test Access Port (TAP) to access complex Integrated Circuits (ICs) such as microprocessors, DSPs, ASICs, and CPLDs. In addition to the TAP, a compliant IC also contains shift registers and a state machine to execute the boundary-scan functions. Data entering the chip on the Test Data In (`TDI`) pin is stored in the instruction register or in one of the data registers. Serial data leaves the chip on the Test Data Out (`TDO`) pin. The boundary-scan logic is clocked by the signal on Test Clock (`TCK`) while the Test Mode Select (TMS) signal drives the state of the TAP controller. Although it is optional, Test Reset (`TRST`) can serve as a hardware-reset signal. Multiple scan-compatible ICs may be serially interconnected on the printed circuit board, forming one or more boundary-scan chains, each chain having its own TAP. Each scan chain provides electrical access from the serial TAP interface to every pin on every IC that is part of the chain.

While the SelectMAP interface requires occupying the general purpose I/O pins in the FPGA chip, the JTAG pins

are the pre-assigned ones instead. Moreover, SRAM modules can be spared since the JTAG interface does not need them for reconfiguration.

4.2 Reconfiguration Control

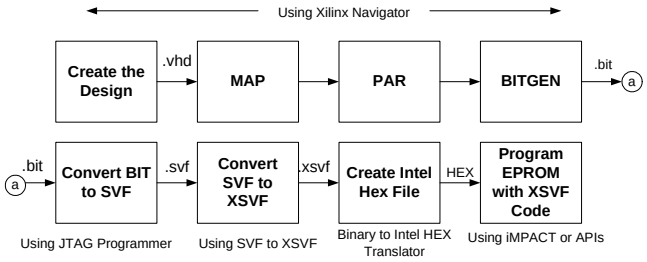

Figure 3: Programming Flow under JTAG Control

As shown in Figure 3, the FPGA programming flow that was used for JTAG-controlled partial reconfiguration begins with design entry in VHDL/Verilog. The design is first simulated using Modelsim for functional verification. Simulation is followed by synthesis, placement and routing with generation of the `.bit` file using Xilinx Navigator. The JTAG operations are recorded in the Serial Vector Format (SVF) file with the iMPACT or JTAG Programmer.

To carry out the reconfiguration operation, an API from Xilinx application note xapp058 is adopted [9]. This API, developed in the C programming language, can be ported to any microcontroller including a host PC. Instead of using a `.bit` file directly, this API is adopted into JTAG prototype using the `.xsvf` format, which needs to be converted by the Xilinx tools ahead of time. An `.xsvf` file is a compact binary file based on the `.svf` file. This format consists of scan operations and movements between different stable states on the IEEE 1149.1 state diagram [9]. `.svf` files were developed by Texas Instruments and have been adopted as a standard for data interchange by JTAG test equipment and software manufacturers [9]. The API reads and interprets the XSVF contents, generates the programming instructions, data, and control signals for the Xilinx devices, and sends them through the JTAG interface. With this API, reconfiguration process can be easily integrated into high-level control logics running on the Host PC.

4.3 Testing Strategy

Because JTAG is a very popular interface, numerous commercial tools are available for testing and debugging purposes. In our prototype, Chipscope version 6.3 from Xilinx has been chosen for verification, which eliminates the requirements for complicated bi-directional communication channels as required with the SelectMAP interface. Therefore, the hardware platform has been greatly simplified. A number of IP cores, such as the PowerPC, the Block RAM controller, the three-segment bus, and the SRAM controller can be removed from the fixed region of the FPGA chip leaving only the basic control logic. The IPIF overhead for each reconfigurable module also becomes unnecessary in the JTAG-controlled design. Only the basic functional modules and bus macros are required in the FPGA device and are arranged as shown in Figure 1.

Figure 4 shows the verification architecture of the JTAG prototype. All of the ChipScope Pro cores use the JTAG Boundary Scan port to communicate to the host PC via a JTAG download cable. The Virtual Input/Output (VIO) core is a customizable core that can both monitor and drive internal FPGA signals in real-time [10]. There are four kinds of signals available in a VIO core: (i) asynchronous inputs, (ii) synchronous inputs, (iii) asynchronous outputs, and (iv) synchronous outputs. In our prototype, synchronous outputs are used from the VIO cores as the testing pattern generator. Every VIO synchronous output has the ability to output a static 1, a static 0, or a pulse train of successive values. A pulse train is a 16-clockcycle sequence of 1's and 0's that is driven out of the core on successive clock cycles. Synchronous inputs are also used to read back the corresponding result from the output signals and displayed in the Chipscope Pro Analyzer as virtual LED indicators.

The Integrated Controller (ICON) core is used as a testing control logic core and provides a communications path between the JTAG Boundary Scan port of the target FPGA and the VIO/ILA cores. When necessary, the Integrated Logic Analyzer (ILA) core is another optional module that can be adapted to the testing system, working as customizable logic analyzer to monitor any internal signal of the design. Since the ILA core is synchronous to the monitored design, all design clock constraints that are applied to the design are also applied to the components inside the ILA core. Detailed information about applying these cores can be found in [10].

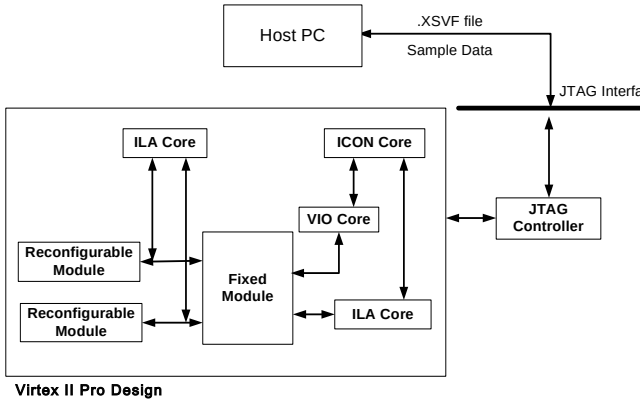

Figure 4. Loosely Coupled System for JTAG.

5 Results and Analysis

To reduce complexity for the case study, direct connecting signals between adjacent areas are taken into examination. Hence a single partial reconfiguration module is located next to the fixed modules directly. The intermodule signals are all connected by the standard bus macro provided by Xilinx. No customized bus macros or multiple PR modules currently are involved in the design.

Both prototypes for the SelectMAP and JTAG interfaces are implemented on an Avnet Virtex II Pro development board equipped with the Virtex II Pro XC2VP7 chip which is installed in a 3-GHz Pentium Host PC.

5.1 System Resource Utilization

The SelectMAP reconfigurable module is a 16-bit SECDED circuit while the JTAG design is an equivalent size 8×8 multiplier. Table II shows the design complexity for configuration and verification. In order to establish the bi-direction communication channel, the design associated with the SelectMAP interface requires eight non-trivial 32-bit wide IP cores to be instantiated inside the fixed region. On the other hand, the JTAG interface only uses four additional cores to carry out the testing with variable data width. As shown in the last four columns of Table II, the SelectMAP design consumes a factor of 3 to 7 times more various logic resources in the fabric than JTAG does.

Since SRAM modules are used as data buffer for reconfiguration purposes, the SelectMAP design uses 77 pins in the fixed region, which is three times higher than the 25 pins required in the JTAG design. In fact, because of the high pin usage spread across the fixed region, only 16 out of 68 available columns remain for the reconfigurable modules. On the other hand, at least twice the amount of reconfigurable resources are available in the case of the JTAG interface design.

In the case of the SelectMAP interface, an IPIF is required for each reconfigurable module due to the usage of the OPB bus. This results in a seven-slice area overhead. However, no such overhead is observed in the case of the JTAG interface design.

During overall operation, when the SelectMAP interface is used to establish reconfiguration and testing channels, sophisticated hardware logic is involved and excessive pins usage is incurred. These costs can limit the size and area placement flexibility of the reconfigurable modules. In this case, large capacity FPGAs, such as the Virtex-II Pro X2VP20 or above, are highly recommended. Furthermore, additional effort is also required for pins assignments and connections with customized bus macros possibly required thereby resulting in an increase in design complexity.

5.2 Timing Evaluation

Since the partial reconfiguration bitstream size vary from case to case by variety of factors, only the full bit file download time is compared, which has a fixed size of 547 KB for the Virtex II Pro XC2VP7. For JTAG prototype, since the testing data generation and detection are all carried out inside FPGA device, communication speed is not measured at this time.

Table III: Timing Evaluation

Interface	Full Reconfiguration	Data Communication
SelectMAP	536 msec	123 msec
JTAG	20.3 sec	N/A

As shown in Table 3, a single data processing cycle, which starts with sending the data out to the PCI bus and ends with reading the data back from the SRAM after the hardware and PowerPC processing is complete, takes up to 123 msec. On the other hand, in the case of the JTAG interface, the reconfiguration process based on the same size reconfiguration file requires 20.3 seconds to complete. Hence the reconfiguration latency observed is 40 times slower than the latency of the SelectMAP interface. This indicates the magnitude of benefit possible using SelectMAP if it is desired to keep reconfiguration latencies low.

6 Conclusion

In this paper, two major reconfiguration interfaces, namely JTAG and SelectMap, have been studied. Considerations regarding hardware component design and complexity, communication protocol development, and testing strategy have been developed and demonstrated.

Table II: Resource Utilization

Interface	# of Fixed Modules	# of Pin of Fixed Modules	Reconfigurable module overhead	Slices for Fixed Modules	BRAM for Fixed Modules	TBUF for Fixed Modules	PPC405 and On-chip Code
SelectMAP	8	77	7 slices	1352	8	352	Y
JTAG	4	25	0	473	0	48	N

The spatial and temporal performance for each interface has also been quantified. Experiments show that the reconfiguration speed of the SelectMap interface is almost 40 times faster than the speed of the JTAG interface. However, communication logic and pin assignment are comparatively simpler in the JTAG interface. This simplicity allows higher flexibility in the placement of the reconfigurable modules since the JTAG interface does not consume significant reconfigurable resources within the FPGA chip.

Currently, a fault tolerance application based on genetic repair is being developing as the top layer application [5]. This application will be integrated within the partial reconfiguration flow in addition to an area management module that will be designed on top of these two interfaces. Multiple partial reconfiguration modules, customized bus macros, and more sophisticated control flow from the application-level software will be integrated into these two testbed systems.

References

[1] Heng Tan and Ronald F. DeMara, "A Device-Controlled Dynamic Configuration Framework Supporting Heterogeneous Resource Management," in *Proceedings of Engineering of Reconfigurable System and Algorithms (ERSA 05)*, pp. 251 – 254, Las Vegas, 2005.

[2] Jungo, Inc., *WinDriver PCI/ISA/CardBus v7.01 User's Guide, 2005.*

[3] Michael Barr, "A Reconfigurable Computing Primer," *Multimedia Systems Design*, Sep. 1998, pp. 44 – 47.

[4] Nobuyuki Ohba and Kohji Takano, "An SoC Design Methodology using FPGAs and Embedded Microprocessors," in *Proceedings of the 41st annual Conference on Design Automation***,** San Diego, CA, USA, 2004.

[5] R. F. DeMara and K. Zhang, "Autonomous FPGA Fault Handling through Competitive Runtime Reconfiguration," in *Proceeding of NASA/DoD Conference on Evolvable Hardware (EH'05)*, Washington D.C., U.S.A., June 29 – July 1, 2005.

[6] Xilinx, Inc., *Virtex-II Pro Platform FPGA User Guide*, v2.4, Aug. 2004.

[7] Xilinx, Inc., *Two Flows for Partial Reconfiguration: Module Based or Difference Based*, v1.1, Nov. 2003.

[8] Xilinx, Inc., *User Core Templates Reference Guide*, v1.2, April 2003.

[9] Xilinx, Inc., *Xilinx In-System Programming using an Embedded Microcontroller*, v3.1, June 2004.

[10] Xilinx, Inc., *Chipscope Pro Software User Guide*, v6.3.1, October 2004.

This document is an author-formatted work. The definitive version for citation appears as:

H. Tan, R. F. DeMara, A. J. Thakkar, A. Ejnioui, J. D. Sattler, *"Complexity and Performance Evaluation of Two Partial Reconfiguration Interfaces on FPGAs: a Case Study," in* Proceedings of the International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA'06), *Las Vegas, Nevada, U.S.A, 2006.*