8-27-2009

High-speed dynamic partial reconfiguration for field programmable gate arrays

John Hoffman

John Colby Hoffman
Candidate

Electrical and Computer Engineering
Department

This thesis is approved, and it is acceptable in quality and form for publication:

Approved by the Thesis Committee:

Major Thesis ,Chairperson

James C Hyer

James Kapalka

High-Speed Dynamic Partial Reconfiguration for Field Programmable Gate Arrays

by

John Colby Hoffman

Electrical Engineering, Texas Tech University
USA, 1998

THESIS

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Masters of Science
Computer Engineering

The University of New Mexico
Albuquerque, New Mexico

July, 2009

To my wife, son and daughter.

You are my inspiration

Acknowledgments

I heartily acknowledge Dr. Marios Pattichis, my advisor and dissertation chair, for continuing to encourage me through the years of classroom teachings and the long number of months writing and rewriting these chapters. His guidance and professional style will remain with me as I continue my career.

I also thank my committee members, Dr. James Plusquellic, and Dr. James Lyke, for their valuable recommendations pertaining to this study and assistance in my professional development. Gratitude is extended to the Xilinx, inc. for the funding to pursue this research.

High-Speed Dynamic Partial Reconfiguration for Field Programmable Gate Arrays

by

John Colby Hoffman

ABSTRACT OF THESIS

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Masters of Science

Computer Engineering

The University of New Mexico

Albuquerque, New Mexico

July, 2009

High-Speed Dynamic Partial Reconfiguration for Field Programmable Gate Arrays

by

John Colby Hoffman

Electrical Engineering, Texas Tech University
USA, 1998

M.Sc., Computer Engineering, University of New Mexico,
USA, 2009

Abstract

With dynamically and partially reconfigurable designs, it is necessary that the speed of the reconfiguration be accomplished in a time that is sufficiently small such that the operation of reconfiguration is not the limiting factor in the process. Therefore, the communication between the source of configuration and the configurable unit must be made as fast as possible. The aim of this work is to use an embedded controller internal to the FPGA to control the reconfiguration process and obtain the maximum speed at which reconfiguration can occur, with current FPGA technology.

The use of Direct Memory Access (DMA) driven operations instead of the current arbitrated bus architectures yielded a 30% increase in the speed of reconfiguration

compared to other methods such as OPB_HWICAP and PLB_HWICAP [**1**]. The use of interrupt driven partial reconfiguration was also introduced, allowing the processor to switch to other tasks during the reconfiguration operation. All of these contributions lead to significant performance improvements over current partial reconfiguration subsystems.

The configuration controller was tested using four partially reconfigurable system implementations: (i) one targeting the Hard IP PowerPC405 on Virtex-4, (ii) a second targeting the Soft IP MicroBlaze on Virtex-5, (iii) a third targeting the Hard IP PowerPC440 on Virtex-5, and (iv) a fourth system targets the Hard IP PowerPC440 on Virtex-5 capable of adaptive feedback. The adaptive feedback Virtex-5 system can use internal voltage and temperature measurements from the Xilinx System Monitor IP to dynamically increase or decrease the speed of reconfiguration and/or change other reconfigurable aspects of the system to better match the environment.

Table of Contents

List of Figures

Chapter 1

Introduction

1.0 Overview

As the speed and size of FPGA reconfigurable fabric has grown the ability to perform multiple complex parallel applications on a single device has become a reality. The BMW WilliamsF1 team is running its fifth generation vehicle control and monitoring (VCM) unit with a Texas Instruments DSP and a Xilinx Virtex family of FPGA devices to control mission critical operations [**2**]. Today, FPGAs have increased product features, decreased product time to market and given system designers abilities unavailable without the use of custom ASIC's.

An area of particular recent interest is dynamic partial reconfiguration. This feature allows the FPGA programmable fabric to change its mode of operation during run time. Effectively, Dynamic Partial Reconfiguration (DPR) allows for time division multiplexing portions of the FPGA fabric while the system is operating. For example, when a satellite is deployed, it may need to perform in a mode that requires 100% of the power available only 25% of the time. During the satellite real-time operation, it may also become necessary to perform an update to take advantage of new algorithms. If the system is developed using FPGAs with the capability to perform partial reconfiguration,

the system can take advantage of operating in a mode requiring 100% of the power when needed and 25% at other times, reducing the power supply needs of the system.

Currently, when considering partial reconfiguration in a high performance system the largest bottleneck is the time which it takes to switch hardware resources for these applications. When a device is partially reconfiguring an area of the fabric, the fabric resource is not available to the system. Therefore, increasing the speed at which the device is reconfigured increases the availability of the reconfigurable resource [**3**].

In this thesis, a custom High Speed Dynamic Partial Reconfiguration Controller (HSDPRC) core is implemented using Xilinx Virtex-4FX and Virtex-5FX devices. Figure 1.1 shows the major components of the proposed scheme. When implemented on a Virtex-5FX device, using LocalLink DMA and a System Monitor feedback to the

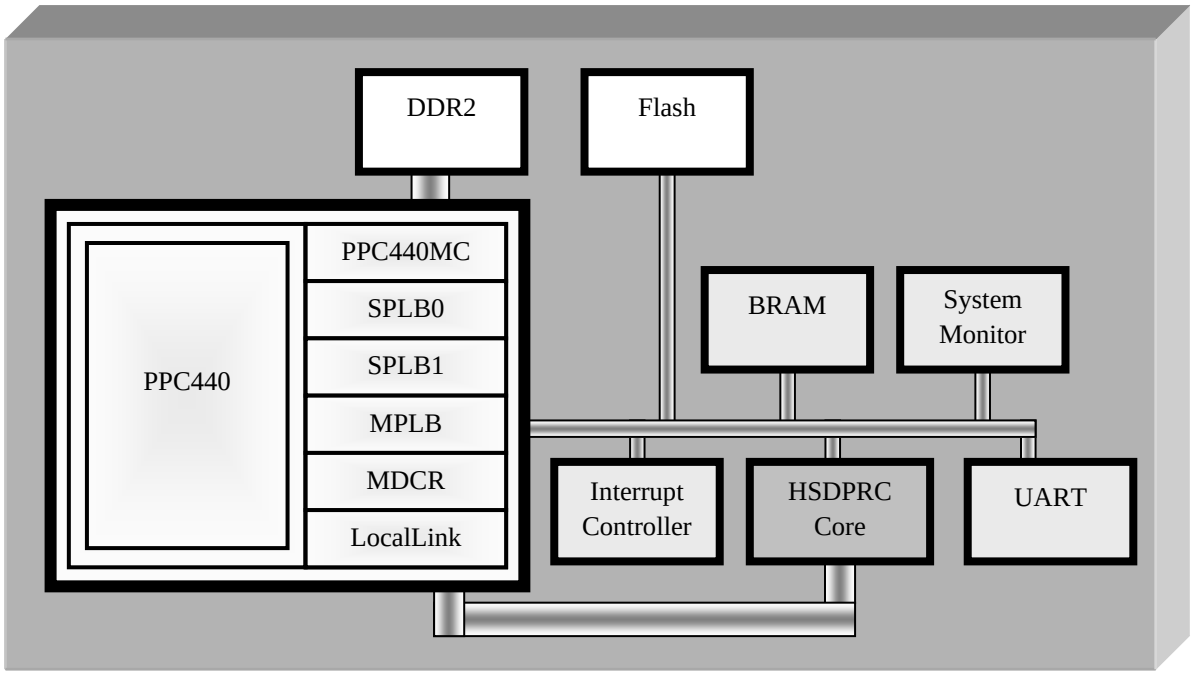

Figure 1.1: LocalLink V5 PPC440 ICAP Controller Block Diagram.

HSDPRC the system has been tested to perform Partial Reconfiguration at over 3.2 Gb/s.

The HSDPRC core was tested with:

- The PowerPC (PPC440) system with a Xilinx ML507 evaluation board,
- The soft core MicroBlaze system with a Xilinx ML507 evaluation board, and
- The PowerPC (PPC405) system with a Xilinx ML410 evaluation board.

The architecture presented can be implemented on all of the Xilinx Virtex-4, Virtex-5 and Virtex-6 family of devices. The design can also easily be ported to Spartan devices that have Partial Reconfiguration software support.

1.1. Thesis goals

The primary objective of this thesis is to maximize the speed of internally directed partial reconfiguration of Field Programmable Gate Arrays, and reduce the processor overhead associated with the process. This is done by designing and implementing a partial reconfigurable system, comprised of a new dedicated DMA driven reconfiguration soft IP core, and several systems with multiple reconfigurable blocks.

In the process, a set of new techniques, modified design flows and hardware cores must be created for designs that incorporate run-time reconfiguration. The effectiveness of the solution is tested by benchmarking the performance of the system using the Advanced Encryption Standard (AES), implemented in the FPGA reconfigurable matrix. The performance of the solution is compared to alternative solutions. The results are

presented in a form that allows us to predict the benefits of this technique in different applications, as well as in the context of future; improved devices.

1.2. Innovations and Contributions

A summary of the primary innovations and contributions includes:

- Obtained the current maximum theoretical performance of the ICAP port (3.2Gb/s)
- New IP Development Flow
- Ability to merge PR regions
- Abstract basic FPGA ICAP function from the user
- Ability to move a bitstream from static storage to FPGA fabric
- Ability to efficiently store and implement multiple partial bitstreams for a single PR region.
- Ability to manipulate partial bitstreams from within the FPGA fabric via the embedded processor.
- Ability to perform interrupt driven partial reconfiguration.

1.3. Thesis Outline

This thesis is organized as follows:

- Chapter 2 presents a Background of FPGAs, Virtex-4 and Virtex-5 platform FPGAs, Dynamic Partial Reconfiguration, Xilinx tools, Xilinx Partial Reconfiguration Flow and Xilinx ML507 and ML410 evaluation boards.
- Chapter 3 presents the related work and motivation.
- Chapter 4 explains the hard and soft-core processor architectures of the implemented Dynamic Partial Reconfiguration systems. This chapter will also cover the software for the embedded processors.
- Chapter 5 presents the methodology of the experiment.
- Chapter 6 presents the results, a discussion of the results and Design considerations, and
- Chapter 7 provides the conclusion for this thesis and future work.

Chapter 2

Basic FPGA Background on Partial Reconfiguration

This section presents basic FPGA architecture, the process used to configure an FPGA and the Xilinx IP used in this thesis.

2.0 FPGA Architecture

Field Programmable Gate Arrays (FPGAs) are programmable semiconductor devices that are based around a matrix of configurable logic blocks (CLBs) connected via programmable interconnects. As opposed to Application Specific Integrated Circuits (ASICs) where the device is custom built for the particular design, FPGAs can be programmed for the desired application or functionality requirements. Fundamentally, an FPGA's CLBs can be configured to implement complex binary / Boolean logic, arithmetic and memory storage functions. FPGAs are SRAM based which can be reprogrammed as the design evolves [**4**].

2.1. Common FPGA Features

FPGAs use reconfigurable logic and interconnect, as well as incorporate hard IP (ASIC type) blocks of commonly used functionality such as RAM, clock management, DSP and

microprocessors. In what follows, brief descriptions of the basic FPGA components are given. Figure 2.1 shows how the blocks can be distributed in the configurable matrix.

2.1.1. Configurable Logic Block (CLBs)

The CLB is the basic logic unit in an FPGA. The Virtex IV devices are made up of CLBs consisting of a configurable switch matrix with 4-6 inputs, some selection circuitry (MUX, etc), and flip-flops. The switch matrix is highly flexible and can be configured to handle combinatorial logic, shift registers, or RAM.

2.1.2. Interconnect

While the CLB provides the logic capability, flexible interconnect routing routes the signals between CLBs, hard IP blocks, and to and from I/Os. Routing comes in several flavors, from that designed to interconnect between CLBs to fast horizontal and vertical long lines spanning the device to global low-skew routing for Clocking and other global signals.

2.1.3. IO (IOBs)

FPGAs provide support for many I/O standards, thus providing the ideal system interface bridge. I/O in FPGA devices is grouped in banks with each bank independently able to support different I/O standards.

2.1.4. Memory

Embedded Block RAM memory is available in most FPGAs, which allows for on-chip memory in a design. The memory allows for a low latency path for other on chip resources.

2.1.5. Clock Resources

FPGAs provide support for complex clocking. The Digital Clock Managers (DCMs) and global-clock multiplexer buffers provide a complete solution for designing high-speed clock networks. Up to twenty DCM blocks are available. To generate deskewed internal or external clocks, each DCM can be used to eliminate clock distribution delay. The DCM also provides 90°, 180°, and 270° phase-shifted versions of the output clocks. Fine-grained phase shifting offers higher resolution phase adjustment with fraction of the clock period increments. Flexible frequency synthesis provides a clock output frequency equal to a fractional or integer multiple of the input clock frequency. Virtex-4 devices can have up to 32 global-clock MUX buffers. The clock tree is designed to be differential. Differential clocking helps reduce jitter and duty cycle distortion.

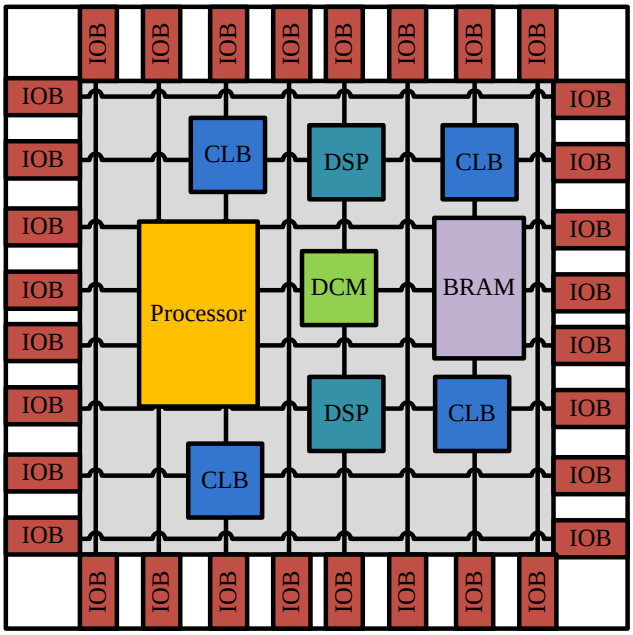

Figure 2.1: Basic FPGA Elements.

2.1.6. Hard IP vs. Soft IP

FPGA designs are described using a high level design language (VHDL, Verilog, or C). Typically, a large FPGA design is a composite of custom user logic, pre designed & verified Commercial off the Shelf (COTS) soft and hard IP macro functions. Soft IP can be placed inside the FPGA at many different locations. Hard IP are system functions which have been optimized for lowest power, fastest throughput and smallest footprint on the die. Hard IP functions within a Xilinx FPGA are embedded 405 or 440 Power PC processors, Ethernet Media Access Control (MAC) cores, Block RAM, DSP Blocks, Clock managers, etc. Hard IP blocks are always present on the FPGA silicon die regardless if the user is using them in implementing specific functions.

2.1.7. System On a Programmable Chip (SoPC)

Tying all of these elements together, using an FPGA, one can realize an entire system on a single programmable device where previous systems required many discrete devices and required more board space, cost and power.

2.1.8. FPGA Packaging

The FPGA is realized as a silicon chip or “die” soldered to a small printed circuit board (PCB) rigid laminate also called a “carrier”. The IO ports of the silicon die are connected to solder balls under the carrier PCB which are used to make connections to the target application’s larger PWB. The composite carrier PWB and silicon die is lastly encased in a material such as plastic or ceramic. Riding on the top is a square piece of metal acting as the package top whose function is to act as a means to extract heat from the silicon die and transfer it to the ambient environment or to an additional mechanical heat sink. Finally, the FPGA is soldered to the target PWB [**4**] [**5**]. Figure 2.2 shows a side view of FPGA packaging.

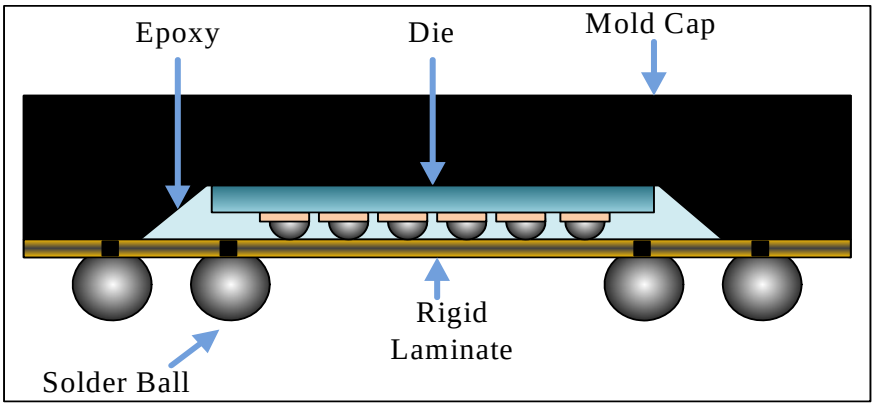

Figure 2.2: FPGA Packaging.

2.2. Standard FPGA Design Flow

To define the behavior of the FPGA, the designer utilizes a hardware description language such as VHDL and creates source code files. Then, using software tools such as Xilinx's ISE or Synplicity, the VHDL code is interpreted (synthesized), logic functions are mapped to FPGA resources (CLBs, IOBs, DSP, memory, etc.), locations for these are selected on the die, and the interconnects to these resources are determined. Finally, the output file is converted into an executable Configuration bitstream file which is used to configure the FPGA. The designer can simulate, verify and validate the resulting output files before actually loading the FPGA on a target system [**6**]. The typical FPGA design flow can be seen in Figure 2.3.

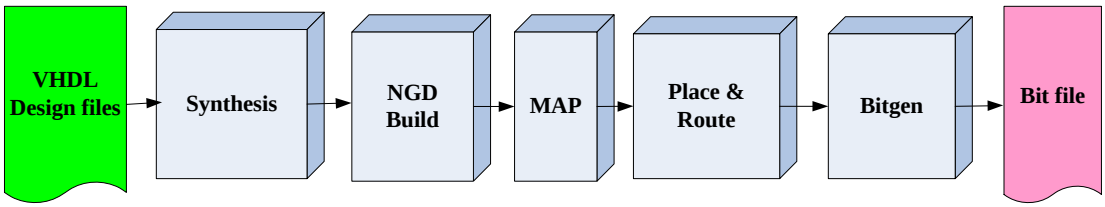

Figure 2.3: FPGA Design Flow.

2.3. Configuration of FPGAs

FPGA devices are configured by loading application-specific configuration data, referred to as "the bitstream", into internal memory referred to as "Configuration Memory". All programmable elements, including the routing resources, are controlled by values stored in the Configuration Memory cells. These values are loaded during configuration and can be reloaded to change the functions of the programmable elements. Because Xilinx

FPGA configuration memory is Static Random Access Memory (SRAM) based, it must be configured each time it is powered-up. The bitstream is loaded into the device through special configuration pins [**6**].

2.4. Full and Partial Configuration

Unique to Xilinx FPGAs, loading of configuration memory can be performed across the entire device or in smaller partial blocks called Partial Reconfigurable Modules (PRMs). The ability to implement PRMs allows the FPGA to stay active and process an existing user application while another application is being loaded and configured [**7**]. This increases a systems availability and capability, because resources may not need to be reinitialized and the system has more resources available for time multiplexing.

2.5. Xilinx FPGA Configuration

The FPGA configuration 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 (SelectMAP, JTAG, or Serial) to the appropriate register. The registers control all other aspects of configuration.

2.5.1. Configuration Memory

The Xilinx configuration memory is arranged in frames that are tiled about the device. Frames are the smallest addressable segment in the configuration memory. A frame consists of 41 32-bit words or 1312 bits.

The Configuration array size equals the number of configuration frames times the number of words per frame. The Configuration overhead consists of commands in the bitstream that are needed to perform configuration, but do not themselves program any memory cells. Configuration overhead contributes to the overall bitstream size.

2.5.2. FPGA Addressing

The Frame Address Pointer (FAR) is a register that indicates the address the configuration controller will write or read from. The addressing in the FPGA is not linear; but rather (starting from the equator of the device) an address comprising of the top/bottom bit, block type, row address, column address, and minor address. A detailed explanation of the addressing for each device can be found in the device Configuration User Guide [**8**].

2.6. Xilinx IP

Xilinx offers many built in cores that were used when developing and testing the HSDPRC. A list of the cores and there description follows.

2.6.1. Embedded Processor Cores

This section outlines all of the Xilinx Embedded processor cores that are available in the Virtex-4 and Virtex-5 devices. The HSDPRC was developed to work with and was tested with all of the available processors from Xilinx.

2.6.1.1. PowerPC 405 Hard IP Core

The PowerPC405 is available in the Virtex-2 Pro Virtex-4 FX family of devices. The PowerPC 405D5 processor is embedded into the Virtex-2 Pro devices while the 405F6 processor core is in the Virtex-4. This embedded processor is capable of 450 MHz clock frequency and over 684 Dhrystone MIPS in the fastest Virtex-4 FX speed grade device. The PowerPC 405 embedded processor includes a memory management unit (MMU), separate instruction and data cache units, JTAG, debug, and trace logic, and timer facilities [**9**]. The PowerPC405 does not have an integrated crossbar switch. To implement a cross bar switch the Soft Core Multi Port Memory controller can be used.

2.6.1.2. PowerPC 440 Hard IP Core

The PowerPC 440 is only available in the Virtex-5 FX family of devices. This embedded processor is capable of 550 MHz clock frequency and 1000 Dhrystone MIPS in the fastest Virtex-5 FX speed grade device. The processor is a seven-stage pipelined PowerPC processor, which consists of a three-stage, dual-issue instruction fetch and decode unit with attached branch unit, together with three independent, four-stage pipelines for complex integer, simple integer, and load/store operations, respectively. Also included is a memory management unit (MMU), separate instruction and data cache units, JTAG, debug, and trace logic, and timer facilities [**10**].

The PowerPC 440 processor block includes a crossbar switch that acts as a central arbitration module accepting master requests from up to five groups of master devices and redirects the transactions to and from one of two groups of slave devices. All data passing from any master device to any slave device within the embedded processor block in Virtex-5 FPGAs passes through the crossbar. Figure 2.4 shows how the PPC440 and crossbar are connected.

The crossbar forms the main interface into or out of the CPU. The crossbar is also the main connection and switch point for any devices instantiated within the FPGA logic that need to communicate with the processor or external memory visible to the processor [**10**].

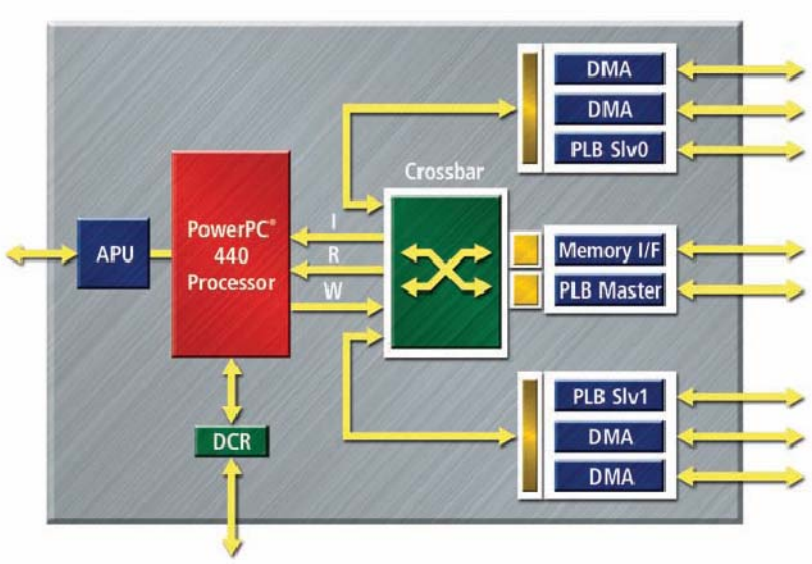

Figure 2.4: PowerPC 440 Block Diagram [11**].**

2.6.1.3. MicroBlaze Soft IP Core

The embedded MicroBlaze Soft IP Core is a flexible MicroBlaze 32-bit soft processing core. The MicroBlaze processor offers memory management and FPU configuration options enabling commercial grade RTOS support, unique for a soft processor. MicroBlaze is highly configurable. The fixed feature set of the processor includes: thirty-two 32-bit general purpose registers, 32-bit instruction word with three operands and two addressing modes, 32-bit address bus and a single issue pipeline [**12**].

2.6.2. LocalLink DMA

The LocalLink interface defines a high-performance, packet-oriented, synchronous and point-to-point interface. The LocalLink interface is full duplex in that, it has separate interfaces for receive and transmit that can operate simultaneously. In addition, the

LocalLink interface can be used for Scatter-Gather DMA (SGDMA) and both Hard DMA (HDMA) or Soft DMA (SDMA). The HDMA uses a 128 bit bus compared to the SDMA 64 bit bus. The HDMA is typically much faster than SDMA and only available using devices with the HDMA crossbar located in the V5FX PPC 440 block. All other devices with PPC405 (V2P, V4FX) or MicroBlaze use SDMA [**13**]. Figure 2.5 shows how the SDMA and HDMA memory crossbars acts as switches, allowing multiple system resources to directly access the external Shared Access Memory (SAM).

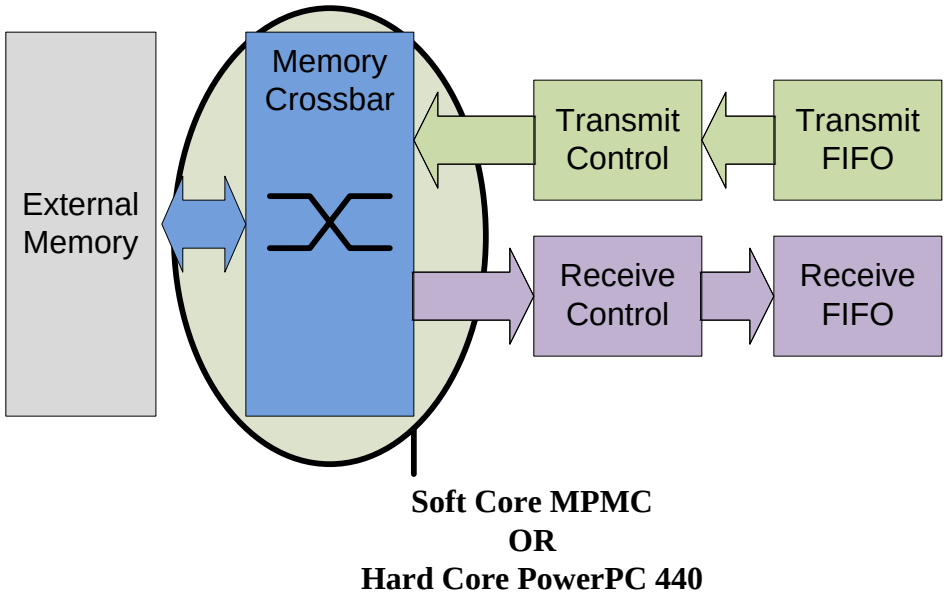

Figure 2.5: LocalLink Block Diagram.

2.6.3. DDR Memory Controller IP

Xilinx offers several Soft IP memory controller cores that allow a user to connect hard and soft IP internal to the FPGA to external memory. The memory controllers abstract the user from performing a device initialization sequence upon power-up and reset conditions, auto-refresh cycles, and single-beat and burst transactions. The memory

interfaces currently offered for DDR are PLB Double Data Rate (DDR) Synchronous DRAM (SDRAM) Controller, Multi Port Memory Controller (MPMC) and Power PC 440 Memory Controller (PPC440MC) is available only on Virtex-5FX devices.

2.6.4. PLB Memory Controller

The PLB DDR SDRAM controller is a Soft IP core that connects the PLB bus to DDR SDRAMs external to the FPGA. The benefit of this memory controller (excluding Virtex-5FX) when compared to the MPMC, is a reduction in the amount of logic resources needed. The drawback to this controller is that in order for the processor to access DDR2 memory it must communicate through the PLB bus, which may be heavily arbitrated. Additionally, if a DMA operation is requested it is also arbitrated through the PLB bus [**14**]. Figure 2.6 depicts the large number of peripherals that must be arbitrated on a PLB centric system.

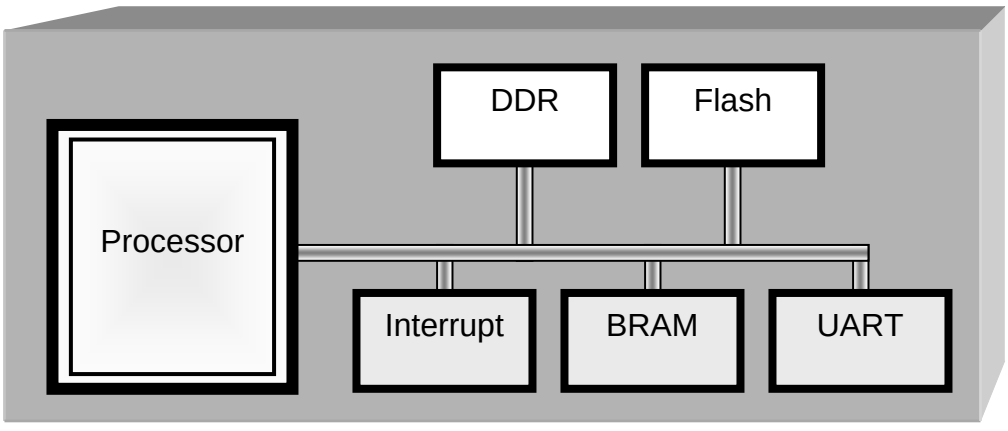

Figure 2.6: PLB Memory Controller Block Digram.

2.6.5. Multi Port Memory Controller (MPMC)

The Xilinx Multi Port Memory Controller (MPMC) is a Programmable memory controller that supports memory such as Synchronous Dynamic Random Access Memory (SDRAM), Double-Data-Rate Synchronous Dynamic Random Access Memory (DDR) and higher speed DDR memory (DDR2). In addition, the MPMC can support Error Correcting Code (ECC) and Performance Monitoring (PM). The MPMC connects to the external memory creating a Soft IP cross bar with up to eight Personality Interface Modules (PIMs) between the memory and user peripherals. The Xilinx MPMC core can be used in all Xilinx Virtex and Spartan FPGAs. The MPMC supports a Soft Direct Memory Access (SDMA) controller that provides a full-duplex, high-bandwidth, LocalLink interfaces into memory [**15**]. Figure 2.7 shows a typical MPMC system.

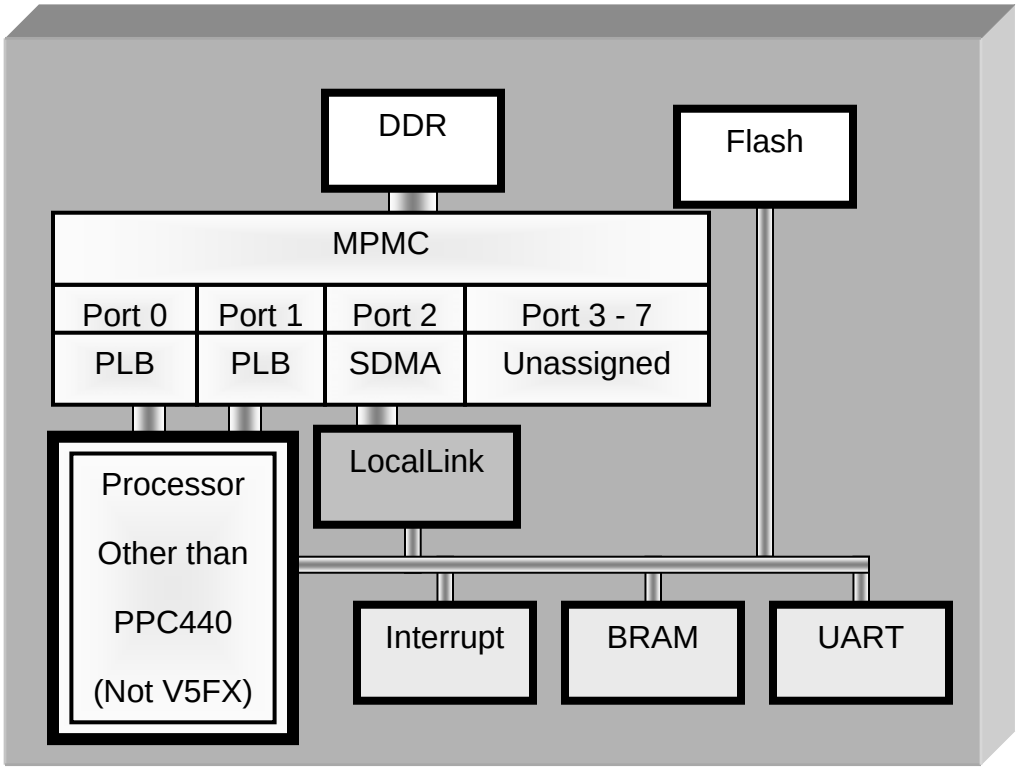

Figure 2.7: LocalLink MPMC Block Digram.

2.6.6. Power PC 440 Memory Controller (PPC440MC)

The Xilinx Power PC 440 (PPC440MC) is a Programmable memory controller that supports Double-Data-Rate Synchronous Dynamic Random Access Memory (DDR2). The PPC440MC can support Error Correcting Code (ECC) and Performance Monitoring (PM). The PPC440MC Connects directly to the hard Crossbar Switch, Memory Controller Interface (MCI), located on the PPC440. The PPC440MC supports a Hard Direct Memory Access (HDMA) controller that provides a full-duplex, high-bandwidth, LocalLink interfaces into memory. Power PC440 and PPC440MC are only available on Virtex 5FX devices [**16**]. Figure 2.8 shows a system utilizing the PPC440MC memory core with an HDMA crossbar switch.

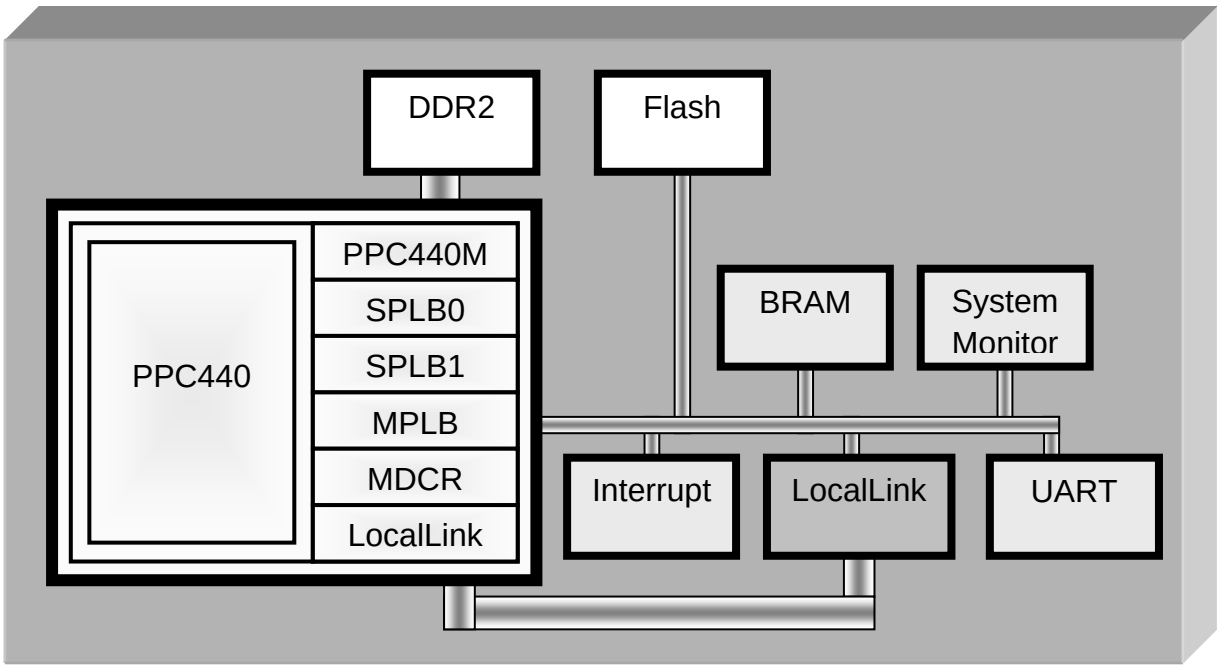

Figure 2.8: LocalLink PPC440MC Block Digram

2.6.7. ICAP

The Internal Configuration Access Port (ICAP) is an FPGA primitive that provides access via the FPGA fabric to the configuration control logic. The Fabric can perform readback and reconfiguration via the ICAP interface. The ICAP port can be used as a 32b or 8b interface and is tested to operate with the device voltage and temperature spec for writes 100MHz and reads 80MHz for Virtex-2pro, Virtex-4, and Virtex-5.

2.6.8. System Monitor

V5 and V6 devices all have a System Monitor. The System Monitor is a 10-bit, 200-kSPS (kilo samples per second) Analog-to-Digital Converter (ADC). The System Monitor is used with sensors embedded in the device that measure FPGA physical operating parameters such as Power (voltage) and Temperature across the entire FPGA die. In addition, the System Monitor incorporates dedicated analog-input pair (VP/VN) and 16 that are user selectable analog inputs. The additional inputs can be used to measure FPGA external analog sources.

Additionally, the System Monitor can be configured to signal alarms when an Over-Temperature (OT) condition occurs (> 125°C) or under voltage, (device specific). The over-temperature alarm signal will then deactivated when the device temperature falls below a user-specified lower limit. If the FPGA power down feature is enabled, the FPGA enters power down when the OT signal becomes active. All of the System Monitor user configurable settings can be set when the System Monitor is instantiated in the users

design or changed during operation using the Dynamic Reconfiguration Port (DRP) and the System Monitor control registers.

A System Monitor IP core is currently available from Xilinx, XPS SYSMON ADC (v1.00b). The XPS SYSMON ADC core connects to both the PPC440 and MicroBlaze processor cores using the Processor Local Bus (PLB) V4.6. The XPS SYSMON ADC core also supports user configurable interrupts that can be set to inform the user if the device is out of a predefined range. Figure 2.9 shows a high level block diagram of the System Monitor hard IP block.

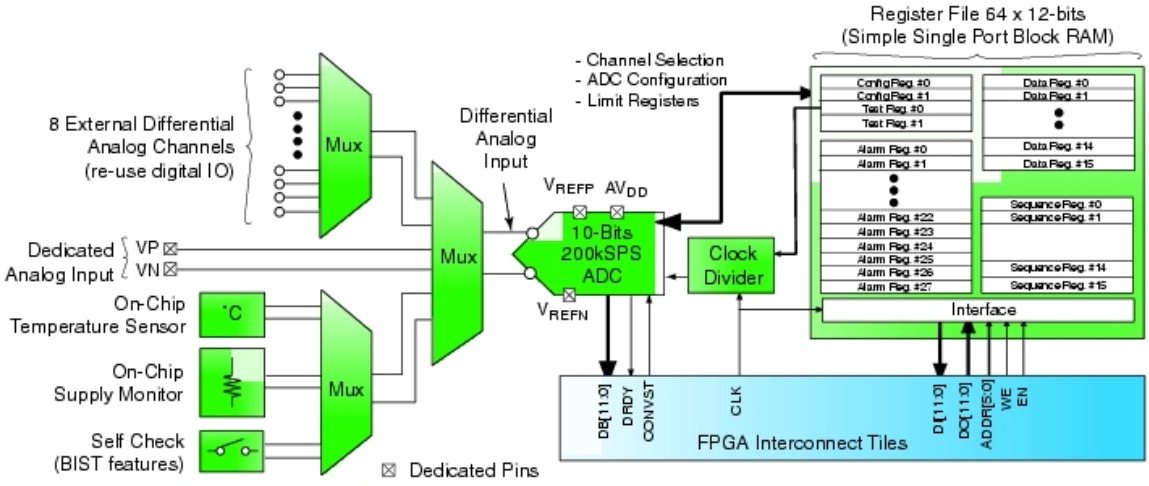

Figure 2.9: System Monitor Block Diagram

The System Monitor is a great tool that can be used as a feed back to the system to dynamically change the device behavior such as to speed, reduced power consumption and total device environment monitoring capabilities. With the System Monitors advanced device physical environment monitoring capabilities and dynamic reconfiguration, the system has the ability to dynamically tailor its operation to match the environment it is in [**17**].

2.7. Xilinx Virtex-4 and Virtex-5 FPGAs

The Xilinx Virtex-4 and Virtex-5 FPGAs are both comprised of three platform families LX, FX, and SX offering multiple feature choices and combinations to address complex applications. LX devices are target at logic intensive applications including more CLB resources that the other device families. SX devices are targeted at DSP applications that need more resources such as DSP hard IP and Block RAM memory. The FX family has a plethora of resources targeted at platform SoC designs; these devices have Embedded PowerPC and Multi gigabit transceiver (MGT) Hard IP Cores. Virtex-4 devices are the predecessors to Virtex-5. Many enhancements including the PowerPC, CLBs, Block RAM memory, DSPs and interconnect have been improved in the Virtex-5.

The main differences between the Virtex-4 and Virtex-5 families that are important when considering a design for PR are given in Figure 2.10 [6][15].

	Virtex-4	**Virtex-5**
Embedded CPU	PowerPC 405 IBM (uses MPMC for DMA) Operating up to 450 MHz	PowerPC 440 IBM Integrated crossbar - (used for LocalLink DMA) Operating up to 550 MHz
CLBs	1 type	2 types
LUTs	4 input	6 input
Clocking	DCM	DCM and PLL
Block RAM	18 Kb that is programmable from 16K x 1 to 512 x 36	36-Kbit that is programmable from 32K x 1 to 512 x 72

Frame Addressing	See: Virtex-4 FPGA Configuration User Guide [**8**]	See: Virtex-5 FPGA Configuration User Guide [**18**]
System Monitor	Unsupported	Supported

Figure 2.10: Comparison of Virtex-4 and Virtex-5.

2.8. Xilinx Software

2.8.1. ISE

The ISE Design Suite comprises of all the tools needed to design, test and implement a Xilinx FPGA. ISE is offered with many licensing options such as Logic, Embedded and DSP. In order to implement a Partial Reconfiguration design a separate PR license must be obtained. The standard ISE design flow is:

- Design Entry – Creating the source files for the project.
- Design Synthesis – synthesizing a design is the process of creating a netlist files from the various source files. The netlist files can serve as input to the implementation module.
- Design Verification (simulation) – functional simulation is accomplished using a simulator to verify the functionality of a design. Timing simulation is run after the design is implemented, it must have context of the actual placement and routing in order to obtain the exact speed and timing of the circuit.
- Design Implementation – After synthesis, implementation will convert the logical netlist into a physical netlist that can be converted to a bitstream and downloaded to

the target device Design implementation involves three steps: Translating the netlist, Mapping and Place&Route.

- Device Configuration – This refers to the actual programming of the target FPGA by downloading the programming file to the Xilinx FPGA.

2.8.2. Embedded Development Kit (EDK)

The Xilinx Embedded Development Kit (EDK) provides an integrated development environment for connecting to and integrating Xilinx MicroBlaze and PowerPC processors. EDK includes Xilinx Platform Studio (XPS) and the Software Development kit (SDK).

- XPS includes a Graphical IDE and command line interface for developing hardware platforms for embedded applications. Including a Processing IP catalog, including a wide variety of processing peripheral cores for customizing embedded systems.
- SDK includes a GNU C/C++ compiler and debugger.

2.8.3. PlanAhead Software

The PlanAhead software is a Xilinx FPGA graphical constraints entry and implementation interface. The software can be used for simplified pin planning, implementation management and incremental design flows. Included with the software are tools such as signal integrity analysis, and timing analysis.

For Partial Reconfiguration designs the software will automatically setup the directory structure and scripting environment, estimate the ratio of logic needed in a region compared to the resources available and perform Design Rule Checking (DRC). Due to long runtimes associated with PR designs these features can greatly decrease the time needed to implement a design.

The software uses many of the sub-module programs that are integrated into the ISE environment. To use PlanAhead for Partial Reconfiguration designs a PR license must be obtained from Xilinx.

2.9. Xilinx Partial Reconfiguration Design Flow

The Xilinx Partial Reconfiguration design flow is a module based design process where the design is broken into hierarchy with sub-modules. The modules can be static or dynamic. Additionally, a complex User Constraint File (.ucf) must be created to constrain the Xilinx implementation tools.

Once the design is broken into sub-modules the Top design and the modules are separately synthesized, translated and mapped. The result of the Map process is a native circuit description (.ncd) file that physically represents the design mapped to the components available in the target Xilinx FPGA.

The next stage in the process is to run place and route, using the <design>.ucf, top.ncd and the sub-module (.ncd) files. First, the top.ncd file is placed and routed; the output is a placed and routed top level netlist (.ncd) and a static.used file. The static.used

file is a list of all the logic resources now used by the top level design. The next step is to place and route each of the sub-module (.ncd) files. Using the static.used and <design>.ucf file the modules are separately placed and routed creating individual (.ncd) files.

The final process is to perform a merge of the individual (.ncd) files and create all of the bitstreams for the design.

2.10. Non Xilinx Software

2.10.1. ModelSim SE

ModelSim SE is a verification and debug environment for ASIC and FPGA designs. The simulators provide support for VHDL, Verilog, SystemC and SystemVerilog IEEE Standards. The simulation environment includes a complex wave form viewer for analyzing complex digital systems.

2.10.2. Synplify Premier

Synplify Premier Software is an advanced synthesis tool from Synplicity. The synthesis engine is technology independent in that a user can target multiple FPGA vendors and device families. Advanced synthesis tools are available such as graph-based physical synthesis, DSP Optimization, Timing Closure and an advanced RTL debug tool Identify.

2.10.3. Graph-based physical synthesis

Graph-based physical synthesis is used to give more accurate timing estimation by back annotating physical aspects of the FPGA into the synthesis process. This allows for accurate estimation of routing delays prior to actual placement and routing. To perform physical synthesis the synthesis engine must have context of the pre-existing wires, switches and placement sites for the specific FPGA architecture. Therefore, a massive database, represented as a detailed routing resource graph is used. The graph provides the information necessary to estimate timing based on actual delay and availability of resources as opposed to just proximity.

2.10.4. Synplify Identify

Synplify Identify gives a developer the ability to directly debug RTL source code. Acting as an In Circuit Emulator (ICE) for FPGAs the Identify tool allows a developer to quickly add probes and trigger conditions to source code prior to implementation.

Once the design is implemented the probes and triggers can be accessed vis-à-vis the FPGA JTAG interface. After a trigger condition is set and a trigger condition occurs the user can then down load the captured data. Using a built in viewer or a viewer such as Mentor Graphics ModelSim the data can be read in a graphical format.

The advantage over traditional FPGA simulation is due to the fact that many of today's FPGAs do not provide complete simulation models for all of the primitive's (Hard IP) available to the user. A debugger that has the ability to capture signals and

displays them to the user for review allows for quick and accurately debugging of a design.

2.11. Xilinx Development Boards

Xilinx offers a wide variety of evaluation platforms to accelerate the design process. The evaluation platforms include software, reference designs, cables, and programming hardware. The evaluation system listed below were chosen for the available Virtex FX device, memory architecture, user interface and System ACE CompactFlash (CF) controller available to store multiple Partial Reconfiguration bitstreams.

2.11.1. ML410 Evaluation Board

The ML410 Evaluation platform is designed for experimenting with the architectural features of Virtex-4 FPGAs and as a platform to create user designs. The ML410 is populated with the Virtex-4 XC4VFX60 device that incorporates two PowerPC 405 processor blocks. A list of all the features available on the ML410 can be found in the ML410 Evaluation Platform User Guide [**19**].

2.11.2. ML507 Evaluation Board

The ML507 Evaluation platform is designed to for experimenting with the architectural features of Virtex-5 FPGAs and as a platform to create user designs. The ML507 is

populated with the Virtex-5 XC5VFX70T device that incorporates a PowerPC 440 processor block with integrated DMA engines and a multi-port crossbar switch. A list of all the features available on the ML507 can be found in the ML505/ML506/ ML507 Evaluation Platform User Guide [**20**].

Chapter 3

Related Work and Motivation

Historically, Field Programmable Gate Arrays (FPGAs) were comprised of a small number of lookup tables connected with a programmable interconnect array. The programmable arrays could then be configured to do tasks such as Glue Logical Application Specific Integrated Circuit (ASICs) devices together “Glue Logic” and simple logic processes.

As physical semiconductor device structures shrunk, the number and speed of the programmable lookup tables has increased. It has also allowed for the addition of hard logic blocks, such as Multi-Gigabit Transceivers, block of dedicated memory, and processor blocks to name a few. The added capabilities have moved the FPGA from being the “Glue Logic” of the system, to one of the main system blocks. In many cases FPGAs are replacing the ASICs they once connected together.

As the amount of programmable logic has scaled, creating more system functionality, so has the need for dynamically controlling power, cooling and system resources. The one solution that can solve the need for dynamic control of power, cooling and system resources is the use of Dynamic Partial Reconfiguration (DPR). DPR is the process of changing the operation of a portion of an FPGA during operation. The partially reconfigured portion of the device can then be timeshared to do multiple operations.

One of the main issues when developing a system that incorporates DPR is the availability of the resource. If a resource is system critical, and used 100% of the time, it does not make sense to partially reconfigure the resource. In addition, when considering time sharing a resource, it is imperative to consider not only the time the resource is in use, but also the time it takes to reconfigure the resource.

In this Thesis, using advanced techniques available with current FPGA architectures', the speed at which Dynamic Partial Reconfiguration can be accomplished is optimized. Thus, the proposed approach further reduces the time a resource is offline due to the reconfiguration process.

3.0 FPGAs and Partial Reconfiguration

FPGAs are fast becoming the platform of choice for many of today's System On a Chip (SoC) deigns. With advanced hard IP blocks such as Block Memory, Multi Gigabit Transceivers (MGTs), embedded processors and DSP cores (Digital signal Processing) the logic applications that FPGAs can be used for are only limited to the size of the FPGA and the power available.

Historically, the FPGA fabric has been configured statically, in that during the board manufacturing process the FPGA's configuration is loaded on a configuration prom and the PCB is shipped. Then when the FPGA is powered by the user, the configuration is loaded by the configuration prom and the device is functional. This approach works well for systems that do not need the FPGA to perform multiple operations or do not need the system to dynamically change operation. Figure 3.1 shows a typical non PR flow where

the configuration prom is programmed during the manufacturing process and does not change for the life of the product.

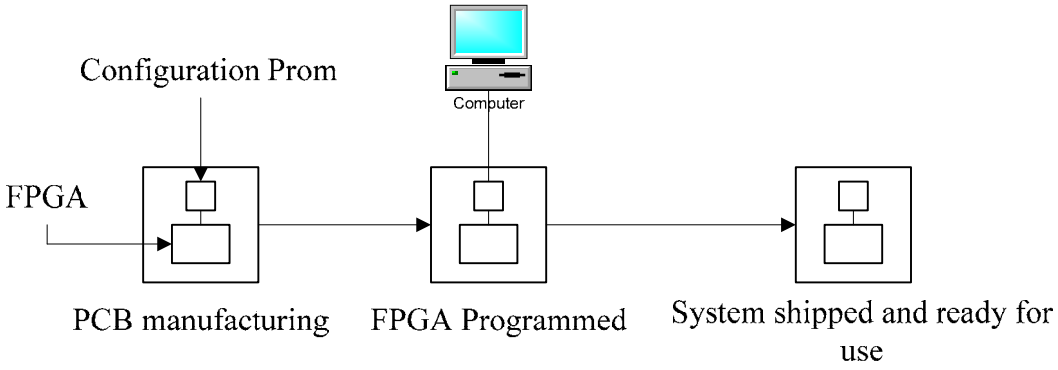

Figure 3.1: FPGA Configuration Process.

Current Xilinx FPGAs have the ability to be Partially Reconfigured. This feature allows the FPGA to adapt to environment conditions, and take advantage of advanced design techniques that were not available before. Examples include:

- Dynamic Power control – With Dynamic power control the FPGA can dynamically reduce the power consumption by reducing the clock speed of system resources or changing the output driver current to minimize power. This approach has also been successfully implemented in Laptop Computers. When the Laptop is connected to an external power source the processor is clocked at its full clock speed. However, when on battery power the clock speed is reduced to save power. This same technique is available with FPGAs, but with much higher granularity of control than the two states available with the laptop application.

- Dynamic Temperature control – Like dynamic Power control, many systems need the ability to change with the environment. With Dynamic Temperature control the system can reduce resources to adapt to the system environment.
- Dynamic Performance – Dynamic hardware resources coupled with an embedded processor, provide for a capability to assign resources at run time. This allows the system to actively maintain the optimal balance between applications preformed in software or hardware, depending on the demand on the system at a particular time.
- High Performance Reconfigurable Computing (HPRC) – Using Dynamic High Speed algorithm acceleration, FPGAs coupled with High-end Processors or DSP's take advantage of an FPGAs parallel algorithm acceleration, and the ability of High-end Processors operating systems to process large amounts of data quickly. These systems have been proven effective with many applications, such as financial modeling and other large problems with parallel data structures.
- High Performance Embedded Reconfigurable Computing (HEPRC) – With high performance embedded computing the system needs to perform high end functions at a fraction of the size weight and power of HPRC. Some applications are automotive control systems [**21**], Software Defined Radios [**22**] and video processing [**1**].
- DISC: The Dynamic Instruction Set Computer (DISC), first proposed by Wirthlin and Hutchings [32], used a medium grain, Configurable Logic ArraY (CLAY) device from National semiconductors. This computer had an instruction set made

up of independent hardware units configured into the device as needed. The architecture's performance was limited by the reconfiguration time overhead.

From a hardware perspective, the main issue with dynamic reconfiguration is the availability of the fabric to the system. This is due in part to the bottleneck imposed on the data path to the fabric.

3.1. Current Dynamic Reconfiguration Solutions

Currently, there are two FPGA-based solutions to run-time reconfiguration: Partial Reconfiguration (PR) employed by Xilinx and Software Programmable Reconfiguration (SPR) used by Altera. The more ambitious PR approach necessitates an FPGA architecture designed to support reconfiguration zones. With the SPR approach, FPGA components are created as highly flexible building blocks controlled and manipulated through embedded software code running on an embedded processor or even through host software running on a general-purpose processor (GPP) [**23**].

3.2. Xilinx XPS HWICAP Controller

Xilinx currently offers a Processor core XPS HWICAP (v1.00.a) that enables Partial reconfiguration of the FPGA with an embedded processor. The core connects to the Processor Local Bus V46 (PLBV46) giving burst mode access to and from the FPGAs Internal Configuration Access Port (ICAP).

With the XPS HWICAP, the FPGAs partial bitstream must be stored in main memory before accessed to reconfigure the FPGA. This creates a requirement that the system have enough external memory to store all of the partial bitstream needed to perform a partial reconfiguration operation. The system also needs to have static storage area to store partial bitstreams when the system power is removed.

In addition to writing to the ICAP, the XPS HWICAP also provides the ability to read back the FPGA configuration bits. During an ICAP read the data frames are read back and stored in a Read FIFO buffer in the FPGA fabric. After the read operation is compete, the CPU can read the frame data directly from the Read FIFO. The read function is a very useful feature of the XPS HWICAP. When coupled with the FRAM ECC primitive, it can be used to detect Single Event Upsets (SEUs).

The XPS HWICAP core provides a bust mode interface to the ICAP port. The interface has a bottle neck created by the arbitration of the PLBV46. It is also processor intensive in that it needs to have a significant amount of processor cycles to initialize and perform a reconfiguration process [**24**].

3.3. PLB DMA ICAP Controller

A method of partial reconfiguration proposed by [**1**], targets the Xilinx Virtex-2P and Virtex-4 family of devices. By creating a custom logic core that instantiates the ICAP and connects it to the Processor Local Bus (PLB), the authors were able to improve data throughput over the existing XPS HWICAP controller by a factor of 58%. Additionally, a method to calculate the expected reconfiguration throughput and reconfiguration time

was introduced, which enables the user to decide if DPR is fast enough for a certain application.

The main drawback of using the PLB bus is the latency due to the arbitration of the bus. There is also a reduction in the processor performance during reconfiguration.

3.4. Other Proposed solutions

In a solution proposed by [**25**], using a Xilinx Virtex-2P FPGA the configuration is controlled with a “configuration controller”. The controller is used to partial reconfigure a device while at the same time perform authentication and encryption or decryption of the partial bitstream.

This method of reconfiguration is not a high speed approach of performing reconfiguration. However, the ability to perform reconfiguration on secure systems will become increasingly important as the technology is adopted in the future.

Chapter 4

Implementation Methodology

4.0 Overview

In order to obtain the current maximum theoretical performance for Internally Directed Partial Reconfiguration (IDPR), it is necessary to provide non-interrupted data to the ICAP port at the maximum speed at which data can be written. With current Virtex-4, Virtex-5 and Virtex-6 devices, the maximum ICAP data path is 32 bits and the frequency is defended as 100MHz across the voltage and temperature spec. Therefore, the maximum achievable bandwidth of the ICAP is at 3.2Gb/s. To obtain this speed, it is necessary to create an interface between a device storing the partial bit stream and the ICAP port with at least such a bandwidth. While such an interface is the simplest, it lacks the requirement to perform IDPR. Recent implementations of IDPR systems were reported in [38, 39, 40, 41, 42].

A simple point to point interface also does not have the ability to move bit streams from one location in the system to another. For example, a point to point interface cannot move information from a slower access storage device to a faster device. This would be the case for most static storage devices that could be used to store the bit stream when the system is powered down. Therefore, to create a system that is more robust than a simple point to point interface, with the ability to move bitstreams stored in static storage to high speed storage, an embedded processor is necessary.

The embedded processor approach also has the ability to control the switching of the partial bit streams. Using the embedded processor to actively control the use of the fabric abstracts many of the mundane tasks associated when reconfiguring the FPGA fabric. This allows the user / designer of the system to have a more abstracted view of system interactions and resources.

In Xilinx V4, V5 and V6 FPGAs, the fastest method to move data to and from processor memory space, is by using a LocalLink DMA interface. The next sections describe the design and development of a custom HSDPRC soft IP core capable of performing IDPR of a target FPGA with an embedded processor.

4.1. HSDPRC Core Design

The HSDPRC is a custom soft IP core specifically developed for LocalLink DMA to and from the Virtex-4, Virtex-5 and Virtex-6 32b-ICAP primitives. The core was developed to interface between both the HDMA or SDMA memory controllers and the FPGA ICAP. To allow for ease of use, the controller was designed with a set of internal ICAP functions. The functions allow the user to perform useful tasks such as reading a specific address in the FPGA and setting internal read and write masks. The following section will present the approach used to develop the HSDPRC.

4.1.1. HSPR Development

The development of the HSDPRC soft IP core required an advanced two part development flow, splitting the development into a traditional cycle accurate RTL simulation and the “hardware in the loop” simulation flow. The purpose for the flow was necessary since the Xilinx ICAP simulation model does not provide a complete behavioral model, in that it does not allow writing or reading internal FPGA registers or configuration address. Therefore, it is not possible without actual hardware testing to see how the device ICAP will behave with a core. The following sections outline the development flow and embedded processor driver description. Figure 4.1 shows the steps used to implement this flow.

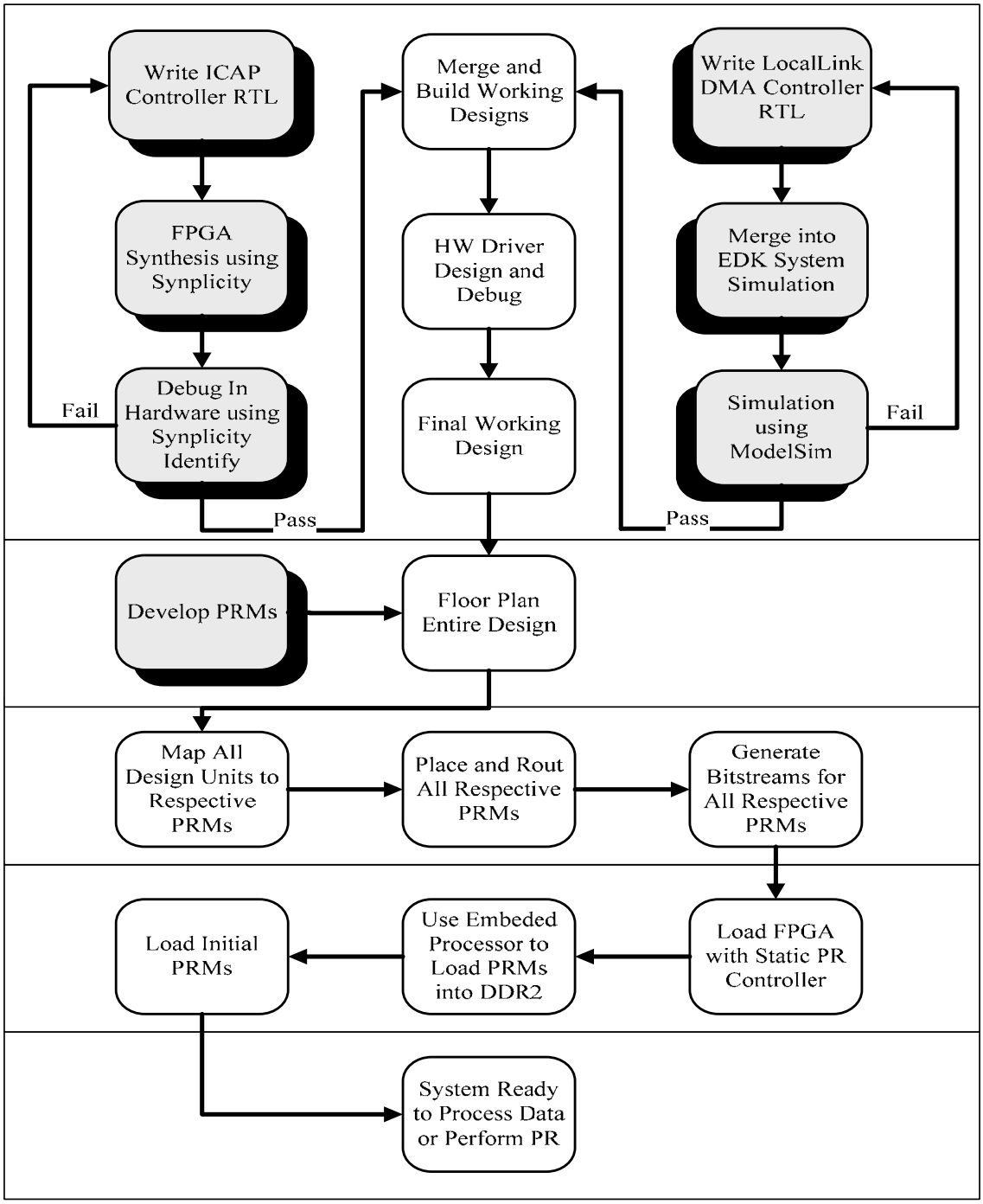

Figure 4.1: HSDPRC Development Flow.

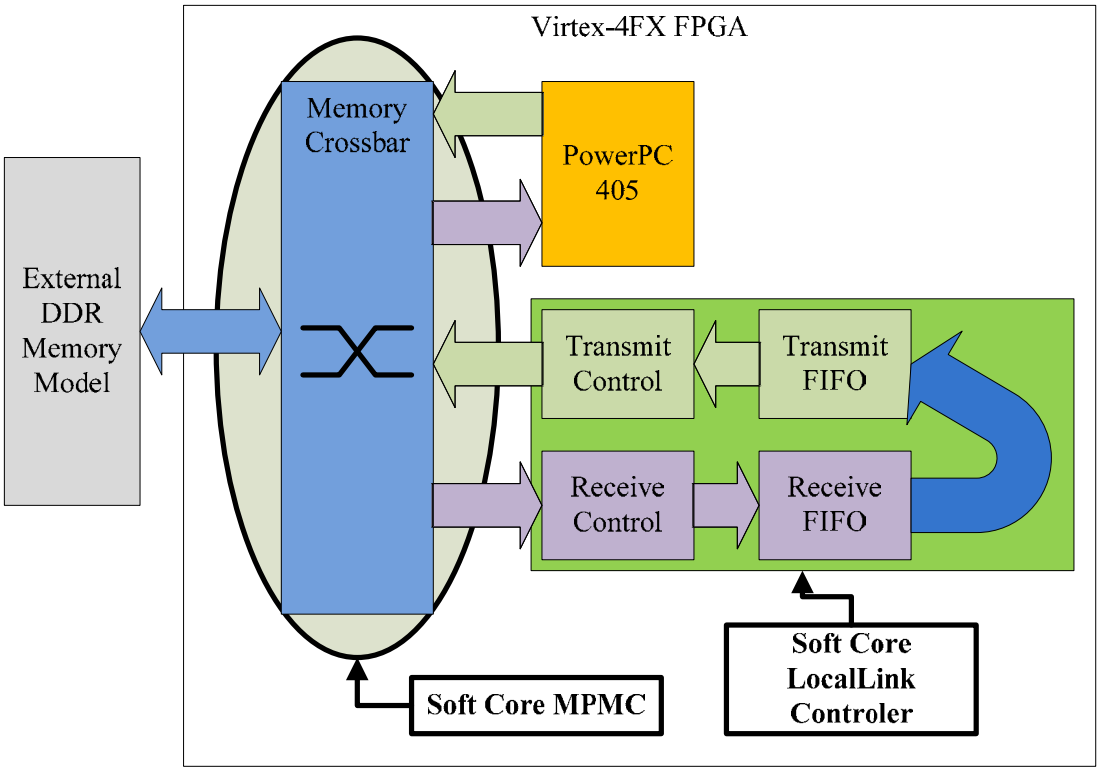

Figure 4.2: Virtex-4FX ModelSim Simulation Model.

4.1.2. ModelSim Simulation

The ModelSim simulation model developed for the LocalLink core consists of a complete ML410 PR subsystem simulation model. The model included the software and driver used to implement the system, DDR memory and a loopback core. This allowed for development and testing of the behavioral VHDL model as well as testing the driver used to configure the LocalLink core. Two ModelSim simulations were run, a DMA test to verify the full function of the LocalLink DMA and a Register test to verify the ability of the processor to access the registers internal to the HSDPRC core.

Figure 4.3 depicts the model developed for the ModelSim simulation. The external DDR model consists of a 64 bit wide data bus running at 200MHz connected to the MPMC memory controller core. The data path between the MPMC and the Transmit and Receive FIFOs are 32-bits wide running at 100MHz. Controlling the system is the Virtex-4 PowerPC 405 simulation model. The PowerPC 405 simulation model is running 300MHz and is used to initialize the external memory, and setup and initiate the DMA transfer.

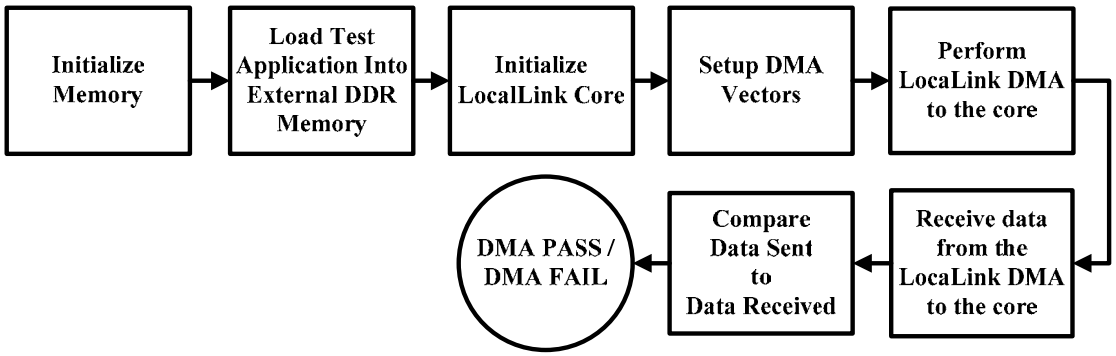

Figure 4.3: ModelSim Simulation Stateflow Diagram.

4.1.3. ModelSim DMA Simulation

Figure 4.3 shows the ModelSim Simulation Stateflow Diagram. The DMA simulation starts by initializing the external DDR memory. After initialization, the DDR is loaded with the software and drivers needed to perform the DMA. Once the software is loaded, the simulation model takes the PowerPC subsystem out of reset and the application code starts running.

The application code used in the simulation is a basic data compare test. First, a known data pattern is generated in the external DDR memory model. Then the application sets up the transmit and receive DMA vectors.

The transmit vector is used to instruct the DMA controller to move data from the external memory models base address, where the known data starts, to the HSDPRCs receive FIFO address. The receive DMA vector instructs the DMA controller to move data from the HSDPRCs Transmit FIFO to a predefined address in the external memory model. The vectors are then sent to the DMA controller and the DMA operation is started.

The data pattern is now sent to the HSDPRC core via a LocalLink DMA. The data is then received in the HSDPRC cores receive FIFO. Once the data is received it is then transmitted back to the DDR memory by writing to the HSDPRC Transmit FIFO.

Once the DMA operation has finished, the known transmit data and the receive data are compared. If the data matches the DMA, the test is successful; else the system needs further analysis.

4.1.4. ModelSim Register Simulation

The HSDPRC internal register simulation is used to ensure the processor has the intended access to the appropriate registers. Table 1 below lists all of the HSDPRC internal registers.

Register Name	(offset from C_BASEADDR)	Access	Size in Bits	Description
Control Register	0x0	Read/Write	32	Bit 31 - Invert Payload Data Bit
Status Register	0x2	Read	32	Bit 31 - Done Bit 30 - Busy Bit
Total Tx Frames Sent	0x4	Read	32	32-bit counter value.
Total Rx Frames Received	0x8	Read	32	32-bit counter value.
Total Tx Bytes Sent	0x10	Read	32	32-bit counter value.
Total Rx Bytes Received	0x20	Read	32	32-bit counter value.
ICAP Control Register	0x40	Read/Write	32	Bit - 31 ICAP_RST Bit - 30 ICAP_EN Bit - 29 ICAP_RW Bit - 28 Disable_loopback Bit - 27 - 24 HW ICAP FUNCTIONS Bit - 23 - 2 Currently Unused Bit - 1 START_PROCESS Bit - 0 PROCESS_DONE
ICAP Single Word Read Register	0x80	Read	32	32-bit value
ICAP TYPE2 WRITE REG	0x100	Read/Write	32	32-bit value
ICAP FAR ADDRESS REG	0x200	Read/Write	32	32-bit value

Table 1: HSDPRC Internal Registers.

This simulation is a simple write followed by read and compare. Here, a known 32 bit data word is written to the target register and then read back and compared to the written value. This simulation can only be performed on Read/Write registers. All of the other registers were simulated by reading the target register and comparing the value to the expected value. If the data matches the expected value the test passes. Otherwise, further system analysis is necessary.

4.1.5. Identify Hardware in the Loop Development

The Identify Hardware in the loop development flow ensures the HSDPRCs ICAP interface operates as expected. The flow required a test fixtures used to stimulate the core and an Identify project used to readback the results and display them for analysis. The Identify test procedure ensures that both ICAP read and write functionality work as expected. The Hardware-in-the-loop simulation was run at both 80MHz and 133MHz. The individual states are listed below. The NOOP commands are needed to push data through the internal ICAP data pipeline.

1. Write the Synchronization word
2. Write one NOP command
3. Write the RCFG command to the CMD register
4. Write one NOP command
5. Write the Starting Frame Address to the FAR (0x00000000)
6. Write the read FDRO register packet header
7. Read the FDRO register count times. See Formula below

8. Write one NOP instruction.
9. Write the DESYNCH command.

The Frame Data Register (FDRO) is a read-only internal FPGA register The FRDO provides readback data for configuration frames starting at the address specified in the Frame Address Register (FAR) register. The read length of the FDRO is:

$$\text{FDRO Read Length} = 41 \times (\text{frames to read} + 1) + 1$$

The extra one in the formula above is used to read and write additional frame. This is to account for the frame buffer. The 41 is the number of words in a frame.

4.1.6. Merging the EDK Simulation with the Identify Project

After the EDK simulation and the Identify hardware-in-the-loop worked independently as expected, the two systems were combined. This was accomplished by removing the transmit and receives FIFOs used in the EDK system and replacing them with the ICAP wrapper used in the Identify system. The final HSDPRC soft IP core was then integrated into an IDPR controller design.

4.1.7. Software Driver

The software API developed for the HSDPRC soft IP core gives the user the ability to read and write all of the cores internal registers. A list of the registers is given in the HSDPRC control Register (see table 1).There is no API for directly initiating a PR DMA operation. The DMA operation is setup using the DMA controller and is application specific. Therefore, it is up to the user of the HSDPRC to set up the DMA operations in the user application code.

Chapter 5

Implementation Results

5.0 Test Platform Overview

The HSDPRC core was designed to target Virtex-4, Virtex-5 and Virtex-6 FPGAs along with all of the Processors offered by Xilinx for these devices. To accomplish this, a cross platform verification approach was necessary. This included developing multiple embedded systems implemented on the target devices. The ML410 and ML507 evaluation boards were the chosen platforms for the Virtex-4 and Virtex-5 devices they are populated with, their availability and extensive documentation. The new Virtex-6 FPGA could not be tested because of its current limited availability. However, the ICAP interface for V5 is the same as for the Virtex-6 FPGA, so it should behave as such. Figure 5.1 is a block diagram summarizing the four implementations.

Techniques developed for testing this architecture have been presented in international conferences ([99], [40] [100], [101]).

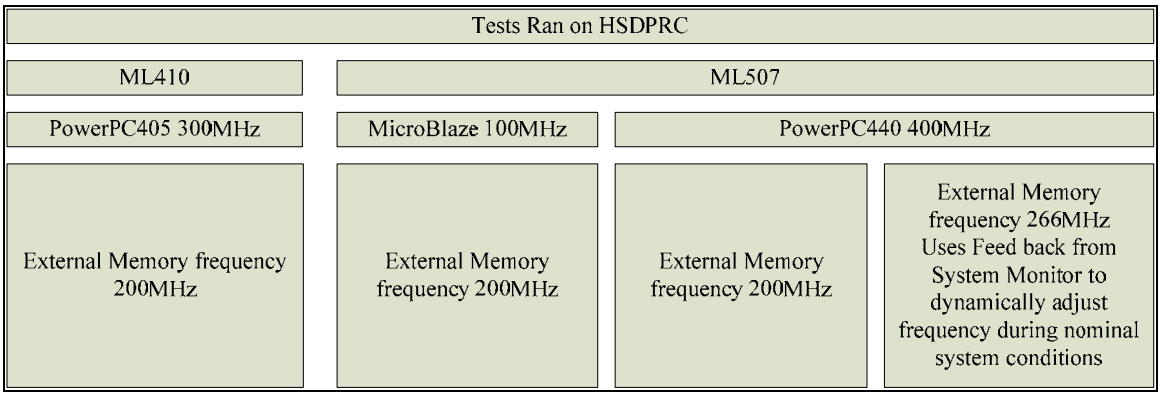

Figure 5.1: HSDPRC Test desings.

5.1. IDPR Controller Design

The following four different implementations of IDPR controllers are configured using different types, speed and configurations of external memory, processors, and devices. All of the designs are configured so as to optimize the speed of IDPR using different types of embedded processors, devices and configuration algorithms. The different devices used (Virtex-4 and Virtex-5) offered different design resources such as hard and soft crossbar memory controllers, PowerPC and MicroBlaze processors, System Monitor and speed of operation. The designs are all implemented on readily available Xilinx ML410 and ML507 evaluation boards.

5.2. Hardware Architecture

The four processor subsystems consist of common IP used across the platforms and specific IP used in order to optimize the processor subsystem for the application. The following will outline the common architecture components and the differences.

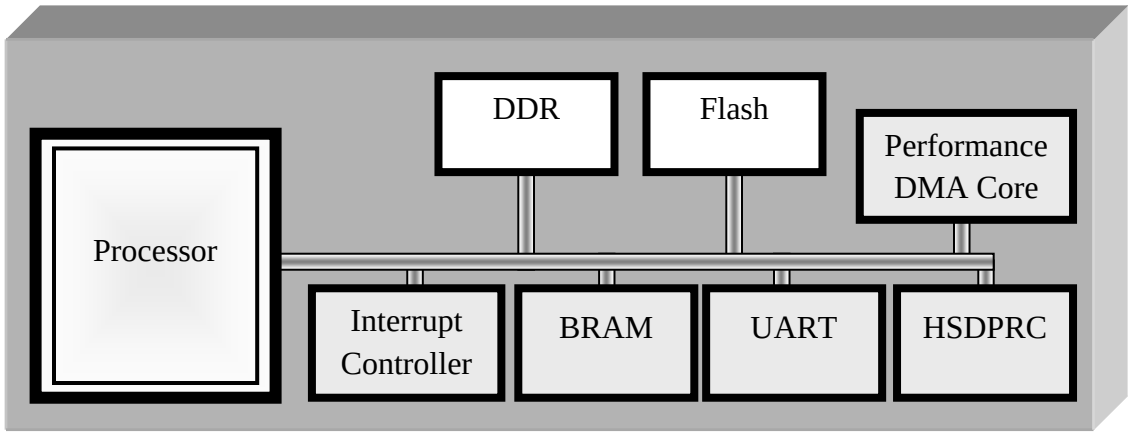

Figure 5.2: Common Architecture Components.

5.2.1. Common Architecture Components

External to the processor each of the four processor subsystems consist of several common soft IP blocks, these blocks are the universal asynchronous receiver/transmitter (UART), Block Memory (BRAM), interrupt controller, Processor Local Bus (PLB), external flash, DDR memory and HSDPRC core. Figure 5.2 shows how the blocks are connected via the PLB bus.

While the HSDPRC core is common to all four platforms, it is configured differently. The differences are outlined in the Architectural Differences section.

5.2.1.1. Processor Local Bus (PLB)

The PLB is 128-bits wide and provides a bus infrastructure for connecting an optional number of PLB masters and slaves components. It consists of a bus control unit, a watchdog timer, and separate address, write, and read data path units, as well as an optional Device Control Register (DCR) slave interface to provide access to its bus error status registers [**26**]. The IDPR design uses the PLB bus to communicate with all of the processor peripherals.

5.2.1.2. Block RAM (BRAM)

BRAM is used by the IDPR system to store the reset vector for the processor or as a place to run the user application from. The reset vector is the default location the processor will go to find the first instruction it will execute after a reset (or startup). For all of the systems but the V5-PPC-266MHz system, BRAM was used to only store the reset vector. For the V5-PPC-266MHz system the entire user application was run out of the system BRAM. Table 2 shows the different IDPR systems that were implemented.

5.2.1.3. DDR

DDR is a volatile storage medium and one of the fastest external storage devices available. In the processor subsystem Double Data Rate memory (DDR) is used to store partial bit streams during runtime. The type of DDR used and the bus width are different

in the systems implemented on the ML410 and ML507. The differences in the DDR used for each system are outlined in the respective sections. When power is removed from a system, the DDR memory will not retain state. For this purpose, a non-volatile storage medium such as Flash or an external hard drive is necessary.

The IDPR system uses the DDR memory to store the partial bitstreams used to reconfigure the device. In addition, the user application program can be run out of the DDR memory.

5.2.1.4. FLASH

The Flash in the system is a static storage medium used to store the partial bitstreams when the system is not powered. After the system is started the processor copies the statically stored partial reconfiguration bitstreams from the flash via the PLB into the much faster DDR memory. Compared to the DDR memory, the flash interface is significantly slower in both speed and access time.

A flash interface was implemented for the IDPR systems however the flash has not been tested for the system. The intent was to provide a static partial bitstream storage device when power was removed from the system.

5.2.1.5. Interrupt Controller

The systems interrupt controller is used to interrupt the processor when a predefined programmable interrupt condition has occurred. For this application, the interrupt controller is configured to interrupt the process after the LocalLink DMA operation has completed and the status registers have valid data.

The interrupt controller was used by the IDPR system to indicate when a partial reconfiguration process completed. This allows the processor to do other tasks such as processing data and/or servicing other interrupt conditions.

5.2.1.6. UART

The UART in the IDPR system is used to connect a host computer to the Processor Subsystem. This allows the user to control the bit streams that are loaded in the partial reconfiguration regions, and to access configuration statistics. The UART can be taken out of systems that have other previsions for controlling the embedded processor.

5.2.1.7. Performance DMA core

The Performance DMA core is used by the IDPR controller to measure the throughput of a single Hard Direct Memory Access (HDMA) using the PowerPC 440 crossbar or Soft Direct Memory Access (SDMA) using the Multi Port Memory Controller (MPMC) block for both the Rx and Tx channels on the LocalLink interface. The core is a slave PLB v4.6

interface with read/write registers used to setup and store the DMA performance calculations. To integrate the core into a design, several signals must be connected in the system. These include: LL_CLK, TX SOF, TX EOF, RX SOF, and RX EOF.

The embedded processor driver for this core has several functions:

- Setup_PERF_DMA - sets the control register to trigger on a transaction.
- Poll_Done_PERF_DMA - Used to poll the status register until the DMA transaction is complete.
- Tx_Transfer - reports the results of the Tx transaction in Mbps.
- Poll_Done_PERF_DMA - reports the results of the Tx transaction in Mbps.

5.2.2. Differences in IDPR Controller Hardware Architecture

The four implementations of IDPR controllers consisted of a Virtex-4 PowerPC design, Virtex-5 MicroBlaze design, Virtex-5 PowerPC design with 200MHz DDR2 and Virtex-5 PowerPC design with 266MHz DDR2. Table 2 below outlines the architectural differences with the four implementations.

IDPR Controller	**V4-PPC**	**V5-MicroBlaze**	**V5-PPC-200MHz**	**V5-PPC-266MHz**
Device	Virtex-4	Virtex-5	Virtex-5	Virtex-5
Development Platform	ML410	ML507	ML507	ML507
Processor	PowerPC 405	MicroBlaze	PowerPC 440	PowerPC 440
Processor Speed	300MHz	125MHz	400MHz	400MHz
Memory	DDR2	DDR2	DDR2	DDR2
Memory Speed	200MHz	200MHz	200MHz	266MHz
System Monitor	NONE	YES	YES	YES
Number of PR Regions	2	4	4	4
ICAP Clock	100MHz	100MHz	100MHz	133MHz
DDR Memory Controller	MPMC	MPMC	PPC440MC	PPC440MC
LocalLink Type	SDMA	SDMA	HDMA	HDMA

Table 2: Differences in IDPR Controller Hardware.

5.2.2.1. DDR Memory

The DDR memory in the two systems is very different. The ML410 board uses a slower 200MHz capable DDR2 memory while the ML507 uses a 266MHz capable DDR2 memory. DDR2 SDRAM is a double data rate synchronous dynamic random access memory. DDR2 supersedes the original DDR SDRAM specification and the two are not compatible. The primary improvement that DDR2 brings over its predecessor is the operation of the external data bus at twice the clock rate. This is achieved by operating the memory cells at half the clock rate (one quarter of the data transfer rate), rather than at the clock rate as in DDR. Consequently, DDR2 memory operating at the same external data bus clock rate as DDR will provide the same bandwidth but with higher latency, resulting in inferior performance. Alternatively, DDR2 memory operating at twice the external data bus clock rate as DDR may provide twice the bandwidth with the same latency (in nanoseconds) [**27**].

The LocalLink interface SDMA and HDMA clock can be configured to be either a 1 to 1 or a 2 to 1 ratio of the memory clock. When deciding what speed to run the external memory a decision was made to maximize the speed of the HSDPRC connected to the LocalLink SDMA or HDMA clock.

The DDR memory used on the ML410 Virtex-4 board is a standard DDR module with a 64 bit data bus. The speed at which the DDR interface on the ML410 board is configured to operate is 100MHz. using a 1 to 1 LocalLink Clock ratio provides a 100MHz clock to the LocalLink SDMA interface and HSDPRC core.

The DDR on the ML507 is a DDR2 with a 64 bit bus width capable of speed up to 266MHz. The speed at which the DDR2 interface on the ML507 board is configured to operate is 200MHz for the MicroBlaze processor system with a SDMA LocalLink to the HSDPRC core. With the PowerPC system, two clocking schemes were used: (i) a 200MHz DDR2 clock providing a 100MHz LocalLink HDMA clock and (ii) a 266MHz DDR2 clock providing a 133MHz LocalLink HDMA clock.

5.2.2.2. V5-PPC- 266MHz

The Virtex-5 PowerPC 440 system clocking the external DDR at 266 MHz is quite different from the other systems. This system uses feedback from the System Monitor IP to determine if the device voltage and temperature parameters are within nominal operating conditions. The reason this is needed is because the device specification for the Virtex-5 ICAP port is only 100MHz through Process Voltage and Temperature (PVT). However, most IC manufactures build in a tolerance when specifying the operating parameters. Taking this into account the ICAP port for this system was over clocked 33% when the device is in nominal operating conditions.

The over clocking technique can be used in fielded systems. However, the clocking for the system would need to be controlled with a BUFGMUX primitive [**28**]. This primitive would allow for glitchless clock switching when needed. A state flow diagram of the software used to perform this operation can be found in the Software test section.

5.3. Multimode AES Crypto PRM

The Advanced Encryption Standard AES algorithm is used for testing the High Speed Partial Reconfiguration Controller. The AES Partial Reconfiguration Module (PRM) algorithm was designed and tested separately from the HSDPRC. This algorithm was tested and it passed all bit for bit test vectors provided by National Institutes of Standards and Technology (NIST), for the implementations of the algorithm used for testing. The NIST test vectors are used to ensure the accuracy of implementation.

This algorithm was chosen because of its ability to be easily scaled by changing the size and modes of operation. Therefore, when switching algorithms the integrity of the reconfiguration controller could be verified by running known complex test patterns through the AES algorithm once the reconfiguration was finished. A brief description of the algorithm follows.

AES is an encryption standard adopted by the U.S. government. The standard comprises three block ciphers, AES-128, AES-192 and AES-256, adopted from a larger collection originally published as Rijndael. Each AES cipher has a 128-bit block size, with key sizes of 128, 192 and 256 bits, respectively. The AES ciphers have been analyzed extensively and are now used worldwide, as was the case with its predecessor, the Data Encryption Standard (DES) [**29**].

5.4. Partitioning the IDPR Designs

The next section will discuss how the IDPR designs were placed on the FPGA fabric. This step in the Partial Reconfiguration process is known as partitioning the design. This is true because the area of the die that will be used for the partial reconfiguration regions and the area used for the IDPR controller will be defined. In addition, the bus macro placement is locked between the regions that use the macros for communication. This process was done using the Xilinx PlanAhead tool.

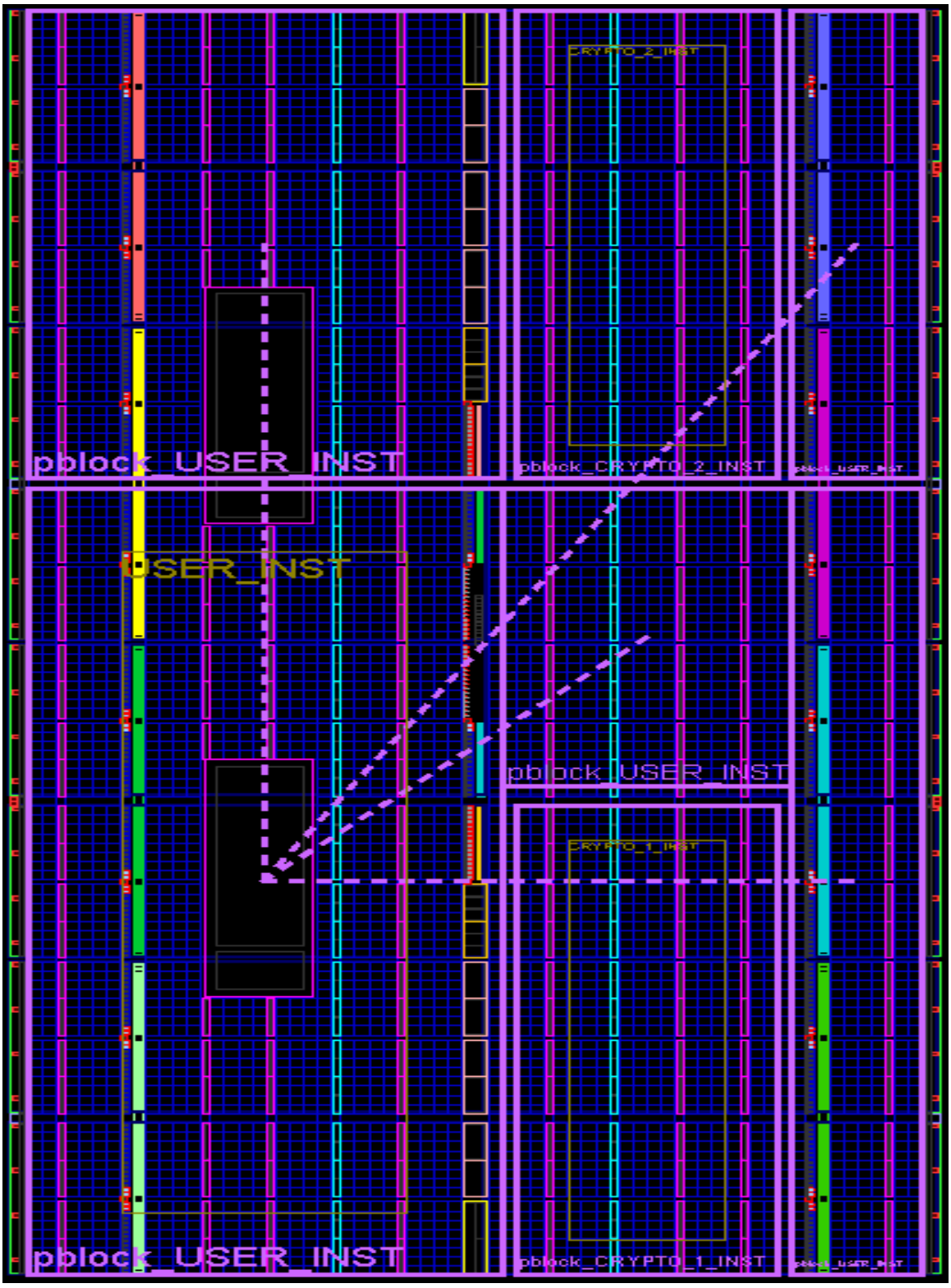

Figure 5.3: Virtex-4 PlanAhead Screen Capture 1.

5.5. Partitioning the Virtex-4 Design

The Virtex-4 IDPR design is partitioned into three regions. The design has two PR regions and the static region with the IDPR controller. The two PR regions are configured to have the same identical available FPGA resources. The PR regions containing the AES PRMs are pblock_CRYPTO_1_INST and pblock_CRYPTO_2_INST. The static region containing the IDPR controller is pblock_USER_INST.

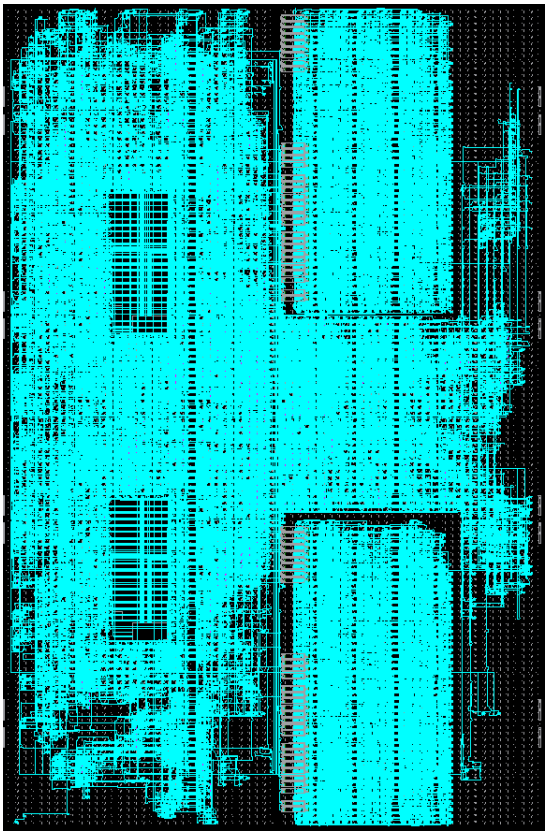

Figure 5.4: Virtex-4 FPGA Editior Screen Capture 1.

In Figure 5.4, we can see the implementation of the design after it has been placed and routed with identical AES hardware requirements in each of the PR regions. The static

design is the PowerPC 405 100MHz system. Note: Only one of the PowerPC processors is used out of the 2 available in the xc4vfx60ff1152.

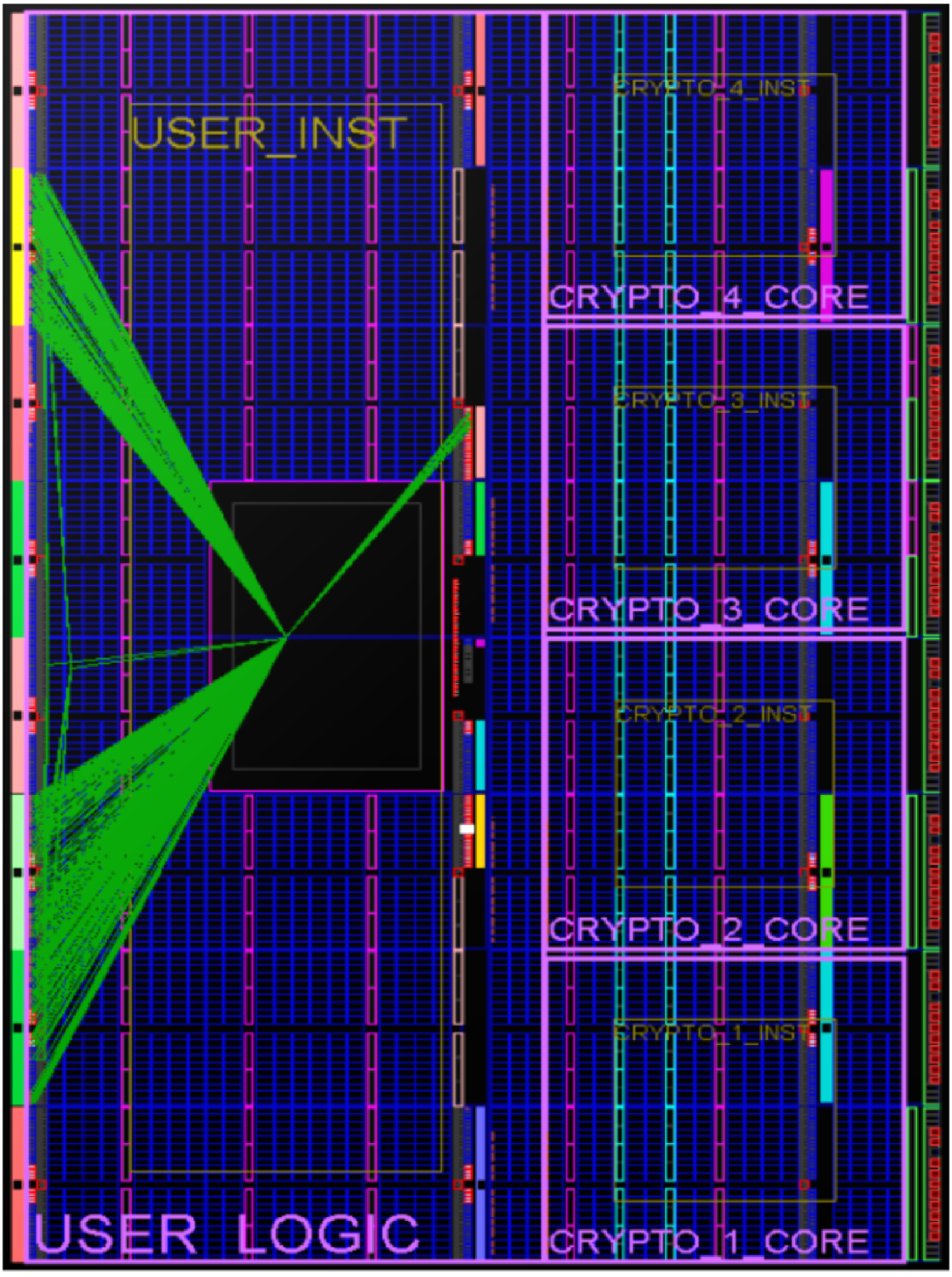

Figure 5.5: Virtex-5 PlanAhead Screen Capture 2.

5.6. Partitioning the Virtex-5 Designs

The Virtex-5 IDPR designs are partitioned in to several PR regions. The PR regions are all configured to have the same identical available FPGA resources. In the image above the PR regions on the Virtex-5 xc5v70FXT, can be seen as CRYPTO_1_CORE, CRYPTO_2_CORE, CRYPTO_3_CORE and CRYPTO_4_CORE the IDPR controller is shown as USER_LOGIC. Each of the crypto core regions can be reconfigured with different implementations of the AES algorithm. The USER_LOGIC section is Static and does not change.

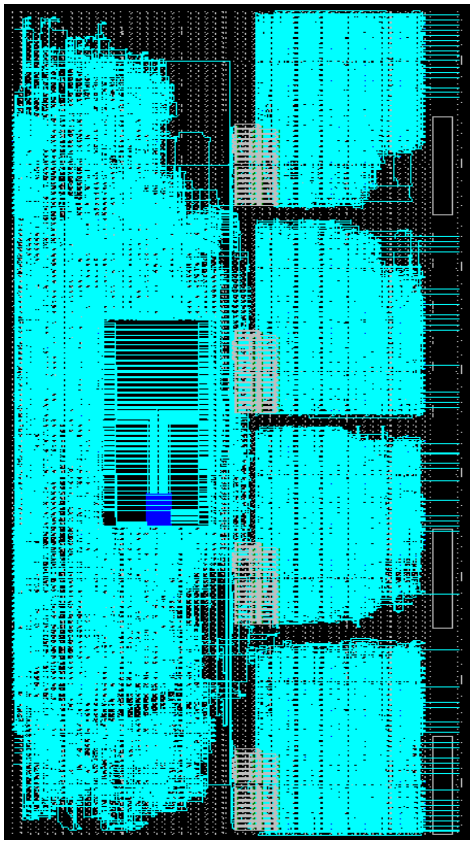

Figure 5.6: Virtex-5 FPGA Editor Screen Capture 2.

In Figure 5.6, using identical AES hardware in each PR region, we can see the results for place and route on the Virtex-5 device. The static design uses the PowerPC 440 266MHz system.

5.7. Test Procedure

The procedure used to test the HSDPRC can be seen in Figure 5.7. To test the system, first the IDPR controller bitstream was written to the FPGA development board. Once the system was loaded it was necessary to connect to the system using an EDK shell and up load the partial bitstreams to the external DDR memory. After each of the partial bitstreams was loaded, the Software test Application was loaded and started.

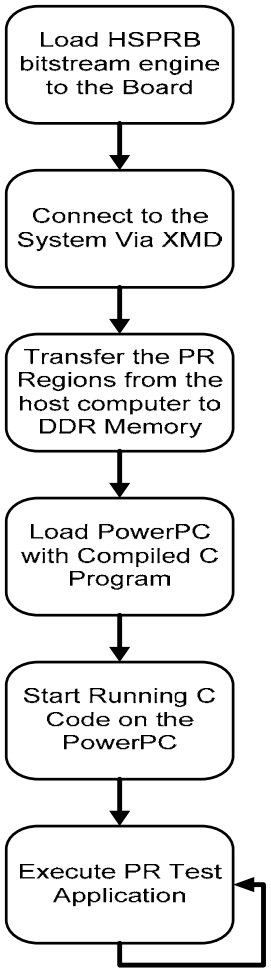

Figure 5.7: Test Procedure Stat Flow Diagram

5.8. Software Test Application

The Software Test Application allows the user to actively monitor and test the IDPR controller in the FPGA. When the system first starts, it sends a message to the user indicating the controller is operational. At this point, the controller is in an idle state until the user issues a PR command. The PR command is a number from 1 to 4. The number indicates the region that is to be reconfigured. Once the user enters a number and sends the command by issuing the enter command the process starts. The process flow can be seen in the Software Test Application State Flow Diagram.

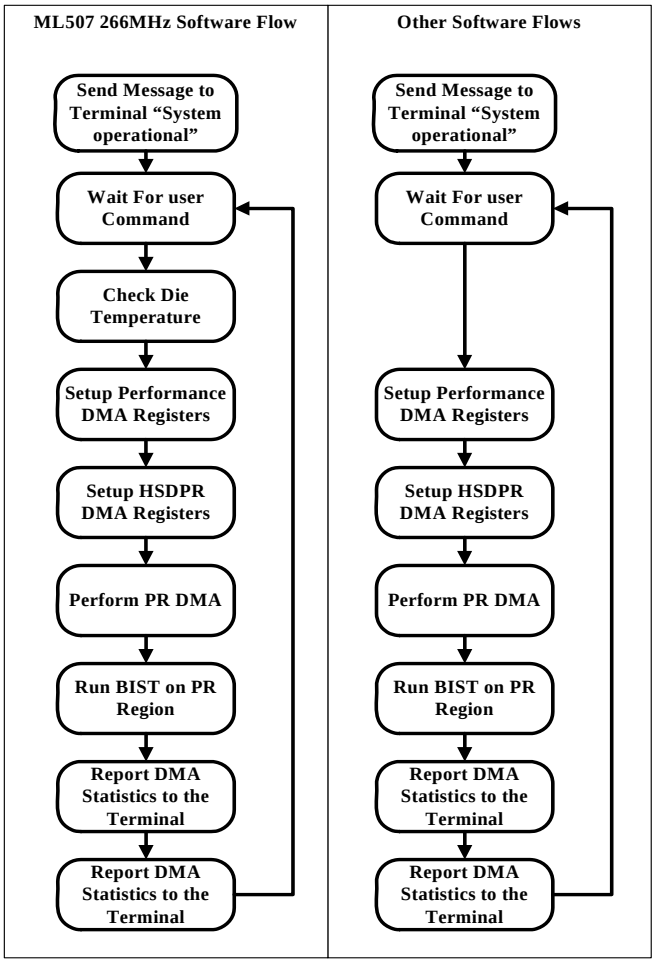

Figure 5.8: Software Test Application State Flow Diagram.

5.9. Reconfiguration Speed Measurements

Table 3 represents the previous maximum speed at which the FPGA could be configured using IDPR.

Method of Configuration	**Virtex-2P**	**Virtex-4**
Frequency	100 MHz	100 MHz
Main Memory	DDR	DDR2
Measured PLB_ICAP	89.9 Mbytes/s	295.4 Mbytes/s
Measured OPB_HWICAP	4.77 Mbytes/s	5.07 Mbytes/s

Table 3: Previously reported state of the art Calculated and Measured Values [1**].**

Table 4 represents the measured speed of ICAP writes and read throughput. The difference in the speed at which the different systems are capable of reconfiguring can be accounted for by the clock speed at which the systems external memory was ran at, the hard vs. soft memory cross bar and the width of the memory controller's fabric side bus.

Platform	**Processor**	**System Frequency**	**Memory Controller**	**ICAP Frequency**	**TX**	**RX**
ML410	PowerPC405	200MHz	MPMC	100 MHz	177.4 Mbytes/s	180.0 Mbytes/s
ML507	MicroBlaze	200MHz	MPMC	100 MHz	178.6 Mbytes/s	181.0 Mbytes/s
ML507	PowerPC440	200MHz	PPC440MC	100 MHz	335.9 Mbytes/s	340.4 Mbytes/s
ML507	PowerPC440	266MHz	PPC440MC	133 MHz	418.5 MBs	424.6 MBs

Table 4: HSDPRC Core measured Values.

The width of the MPMC internal memory bus is set at 64-bits compared to the 128-bit bus available with the PPC440MC. This alone accounts for the 2x speed increase over the systems that do not have PPC440MC. After taking this into account, the systems scales closely with respect to clock speed. Figure 5.10 shows this relation.

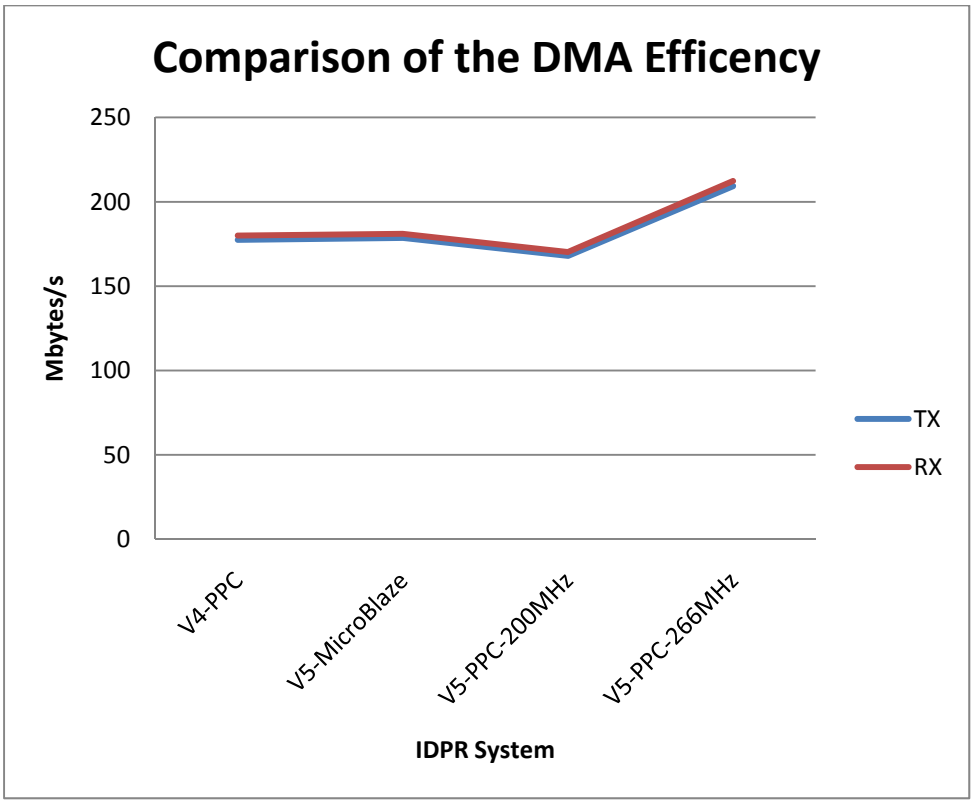

Figure 5.9: Clock Speed Comparison

5.10. Device Utilization Summary

The HSDPRC cores device utilization for a Virtex-5 can be seen in Table 5.

Number of ICAPs	1
Number of RAMB18X2SDPs	1
Number of Slice Registers	1085
Number used as Flip Flops	1082
Number used as Latches	3
Number of Slice LUTS	923
Number of Slice LUT-Flip Flop pairs	1530

Table 5: Virtex-5 Device utilzation for HSDPRC.

Chapter 6

Conclusion

In this thesis, the objective of designing and implementing a High Speed Dynamic Partial Reconfiguration Controller (HSDPRC) core that maximized the bandwidth of the ICAP was attained. The design was evaluated in terms of reconfiguration performance, and logical resources.

Additionally, the use of on chip temperature and voltage active feedback for Dynamic Partial Reconfiguration was introduced and tested. This approach created an IDPR controller that is tightly coupled with the FPGA silicon and the system behavior than previous work has accomplished.

When the ICAP core was run at the Xilinx specification over PVT, the HSDPRC soft IP core increased the speed at which a partial bitstream could be loaded into the FPGA fabric, compared to the current maximum measured speed, by over 12%. When using System Monitor active feedback the system was approximately 30% faster than the current state-of-the-art.

The HSDPRC core obtained the maximum theoretical ICAP performance achievable with Virtex-5 FXT devices. This technology and implementation is easily scalable to future generations of FPGAs.

References

[1] C. Claus, B. Zhang, W. Stechele, L. Braun, M. H¨ubner and J. Becker"A Multi-Platform Controller Allowing For Maximum Dynamic Partial Reconfiguration Throughput," *IEEE*, pp. 535-538, 2008.

[2] L. Boland, "Formula 1 Racing: The Xilinx Advantage," *Xcell Journal*, pp. 46-49, 2003.

[3] Vera, Guillermo, “A Dynamic Arithmetic Architecture: Precision, Power and Performance Considerations”, PhD Dissertation, The University of New Mexico, 2008.

[4] Xilinx, *Virtex-4 Family Overview (UG070)*. San Jose CA: Xilinx Inc., December 1, 2008.

[5] Xilinx. Xilinx. [Online]. HYPERLINK "http://www.xilinx.com/company/gettingstarted/index.htm" http://www.xilinx.com/company/gettingstarted/index.htm

[6] *The Design Warrior's Guide to FPGAs*. Elsevier Science & Technology Books, 2004.

[7] Xilinx, "Early Access Partial Reconfiguration User Guide(UG208)," 2006. [Online]. HYPERLINK "http://www12.informatik.uni-erlangen.de/esmwiki/images/f/f3/Pr_flow.pdf" http://www12.informatik.uni-erlangen.de/esmwiki/images/f/f3/Pr_flow.pdf

[8] Xilinx. (2009, Jun.) www.Xilinx.com. [Online]. HYPERLINK "http://www.xilinx.com/support/documentation/user_guides/ug071.pdf" http://www.xilinx.com/support/documentation/user_guides/ug071.pdf

[9] Xilinx. (2007, Jan.) www.Xilinx.com. [Online]. HYPERLINK "http://www.xilinx.com/support/documentation/user_guides/ug011.pdf" http://www.xilinx.com/support/documentation/user_guides/ug011.pdf

[10] Xilinx. (2009, Jan.) www.Xilinx.com. [Online]. HYPERLINK "http://www.xilinx.com/support/documentation/user_guides/ug200.pdf" http://www.xilinx.com/support/documentation/user_guides/ug200.pdf

[11] Xilinx. (2009, Jan.) www.Xilinx.Com. [Online]. HYPERLINK "http://www.xilinx.com/support/documentation/user_guides/ug200.pdf" http://www.xilinx.com/support/documentation/user_guides/ug200.pdf

[12] Xilinx. (2008, Jan.) www.xilinx.com. [Online]. HYPERLINK "http://www.xilinx.com/support/documentation/sw_manuals/mb_ref_guide.pdf" http://www.xilinx.com/support/documentation/sw_manuals/mb_ref_guide.pdf

[13] J. Lucero. (2008, Oct.) Xilinx.com. [Online]. HYPERLINK "http://www.xilinx.com/support/documentation/application_notes/xapp1121.pdf" http://www.xilinx.com/support/documentation/application_notes/xapp1121.pdf

[14] Xilinx. (2003, Oct.) Xilinx. [Online]. HYPERLINK "http://www.xilinx.com/support/documentation/ip_documentation/plb_ddr.pdf" http://www.xilinx.com/support/documentation/ip_documentation/plb_ddr.pdf

[15] Xilinx. (2008, Jun.) Xilinx.com. [Online]. HYPERLINK "http://www.xilinx.com/support/documentation/ip_documentation/mpmc.pdf" http://www.xilinx.com/support/documentation/ip_documentation/mpmc.pdf

[16] Xilinx. (April,) Xilinx.com. [Online]. HYPERLINK "http://www.xilinx.com/support/documentation/ip_documentation/ppc440mc_ddr2.pdf" http://www.xilinx.com/support/documentation/ip_documentation/ppc440mc_ddr2.pdf

[17] Xilinx. (2009, Mar.) www.Xilinx.com. [Online]. HYPERLINK "http://www.xilinx.com/support/documentation/user_guides/ug192.pdf" http://www.xilinx.com/support/documentation/user_guides/ug192.pdf

[18] Xilinx. (2009, Feb.) www.Xilinx.com. [Online]. HYPERLINK "http://www.xilinx.com/support/documentation/user_guides/ug191.pdf" http://www.xilinx.com/support/documentation/user_guides/ug191.pdf

[19] Xilinx. (2008, Dec.) www.xilinx.com. [Online]. HYPERLINK "http://www.xilinx.com/support/documentation/boards_and_kits/ug085.pdf" http://www.xilinx.com/support/documentation/boards_and_kits/ug085.pdf

[20] Xilinx. (2008, Nov.) www.Xilinx.com. [Online]. HYPERLINK "http://www.xilinx.com/support/documentation/boards_and_kits/ug347.pdf" http://www.xilinx.com/support/documentation/boards_and_kits/ug347.pdf

[21] Ju¨rgen Becker, Michael Hu¨bner, Gerhard Hettich, Rainer Constapel, Joachim Eisenmann, and Ju¨rgen Luka, "Dynamic and Partial FPGA Exploitation," *Proceedings of the IEEE Vol. 95, No. 2, February*, pp. 438-452, 2007.

[22] Xilinx. (2006, Feb.) Xilinx. [Online]. HYPERLINK "http://www.xilinx.com/prs_rls/dsp/0626_sdr.htm"

http://www.xilinx.com/prs_rls/dsp/0626_sdr.htm

[23] D. R. a. J. R. Kenny. (2008, Oct.) Two competitive FPGA methodologies for run-time reconfiguration. [Online]. HYPERLINK "http://www.dsp-fpga.com/articles/id/?3636" http://www.dsp-fpga.com/articles/id/?3636

[24] Xilinx. (2007, Aug.) Xilinx. [Online]. HYPERLINK "http://www.xilinx.com/support/documentation/ip_documentation/xps_hwicap.pdf" http://www.xilinx.com/support/documentation/ip_documentation/xps_hwicap.pdf

[25] Zeineddini, Amir, "Secure Partial Reconfiguration of FPGAS", Thesis, George Mason University, 2005.

[26] Xilinx. (2007, Aug.) Processor Local Bus. [Online]. HYPERLINK "http://www.xilinx.com/support/documentation/ip_documentation/plb_v46.pdf" http://www.xilinx.com/support/documentation/ip_documentation/plb_v46.pdf

[27] wikipedia. (2009, Jun.) wikipedia. [Online]. HYPERLINK "http://en.wikipedia.org/wiki/DDR2_SDRAM"

http://en.wikipedia.org/wiki/DDR2_SDRAM

[28] Xilinx. (2009, Jun.) www.Xilinx.com. [Online]. HYPERLINK "http://www.xilinx.com/support/documentation/user_guides/ug190.pdf" http://www.xilinx.com/support/documentation/user_guides/ug190.pdf

[29] Wikipedia. (2009, Apr.) www.Wikipedia.com. [Online]. HYPERLINK "http://en.wikipedia.org/w/index.php?title=Advanced_Encryption_Standard&action=history" http://en.wikipedia.org/w/index.php?title=Advanced_Encryption_Standard&action=history

[30] S. Gianelli. Xilinx. [Online]. HYPERLINK "http://www.xilinx.com/prs_rls/2007/end_markets/0713_v4nsa.htm" http://www.xilinx.com/prs_rls/2007/end_markets/0713_v4nsa.htm

[31] Actel. (2006) Actel. [Online]. HYPERLINK "http://www.actel.com/documents/UnderstandingFPGAPower.pdf" http://www.actel.com/documents/UnderstandingFPGAPower.pdf

[32] M.J. Wirthlin and B.L. Hutchings, "A dynamic instruction set computer," *IEEE Symposium on* FPGA*s for Custom Computing Machines*, pp. 99–107, April 1995.

[33] Alonzo Vera, Jorge Parra, Craig Kief, Marios Pattichis, and Howard Pollard, "Integrating reconfigurable logic in the first digital logic course," in *International*

Conference on Engineering Education, ICEE2006., 2006.

[34] Alonzo Vera, Daniel Llamocca, Marios Pattichis, William Kemp, Walter Sheed, Dave Alexander, and James Like, "Dose rate upset investigations on the xilinx virtex iv field programmable gate arrays," in *The Nuclear and Space Radiation Effects Conference, NSREC2007.*, 2007.

[35] Alonzo Vera, Uwe Meyer-Baese, and Marios Pattichis, "An FPGA based rapid prototyping platform for wavelet coprocessors," in *Independent Component Analyses, Wavelets, Unsupervised Smart Sensors, and Neural Networks Conference, SPIE2007*, 2007.

[36] Alonzo Vera, Jorge Parra, Craig Kief, Marios Pattichis, and Howard Pollard, "Integrating reconfigurable logic in the first digital logic course," in *International Conference on Engineering Education, ICEE2006.*, 2006.

[37] Uwe Meyer-Baese, Alonzo Vera, A. Meyer-Baese, M. Pattichis, and R. Perry., "Discrete wavelets transform fpga design using matlab/simulink," in *Independent Component Analyses, Wavelets, Unsupervised Smart Sensors, and Neural Networks Conference, SPIE2006.*, 2006.

[38] G. Mc Gregor and P. Lysaght, "Self controlling dynamic reconfiguration: A case study," *Proceedings of the 9th International Workshop on Field-Programmable Logic and Applications, FPL*, pp. 144–154, 1999.

[39] Brandon Blodget, Scott McMillan, and Patrick Lysaght, "A lightweight approach for embedded reconfiguration of FPGAs," *Proceedings of the conference on Design, Automation and Test in Europe - Volume 1 DATE '03*, 2003.

[40] Brandon Blodget et al, "A self-reconfiguring platform," *Proceedings of the 13th International Conference on Field-Programmable Logic and Applications, FPL*, 2003.

[41] John Williams and Neil Bergann, "Embedded linux as a platform for dynamically self-reconfiguring systems-on-chip," *Proc. International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA'04)*, 2004.

[42] M.J. Wirthlin and B.L. Hutchings, "A dynamic instruction set computer," *IEEE Symposium on* FPGA*s for Custom Computing Machines*, pp. 99–107, April 1995.