Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2011, Article ID 439072, 10 pages
doi:10.1155/2011/439072

Research Article

A High-Speed Dynamic Partial Reconfiguration Controller Using Direct Memory Access Through a Multiport Memory Controller and Overclocking with Active Feedback

John C. Hoffman and Marios S. Pattichis

Department of Electrical and Computer Engineering, The University of New Mexico, MSC01 1100, 1 University of New Mexico, Albuquerque, NM 87131-0001, USA

Correspondence should be addressed to Marios S. Pattichis, pattichis@ece.unm.edu

Received 23 November 2010; Revised 12 April 2011; Accepted 7 June 2011

Academic Editor: Eduardo Marques

Dynamically reconfigurable computing platforms provide promising methods for dynamic management of hardware resources, power, and performance. Yet, progress in dynamically reconfigurable computing is fundamentally limited by the reconfiguration time overhead. Prior research in the development of dynamic partial reconfiguration (DPR) controllers has been limited by its use of the Processor Local Bus (PLB). As a result, the bus was unavailable during DPR. This resulted in significant time overhead. To minimize the overhead, we introduce the use of a multiport memory controller (MPMC) that frees the PLB during the reconfiguration process. The processor is thus allowed to switch to other tasks during the reconfiguration operation. This effectively limits the reconfiguration overhead. An interrupt is used to inform the processor when the operation is complete. Therefore, the system can multitask during the reconfiguration operation. Furthermore, to maximize performance, we introduce the use of overclocking with active feedback. During overclocking, the use of active feedback is used to ensure that the device voltage and temperature are within nominal operating conditions. All of these contributions lead to significant performance improvements over current partial reconfiguration subsystems. The portability of the system, demonstrated on the Virtex-4 and the Virtex-5, consists of four different hardware platforms.

1. Introduction

As the speed and size of FPGA reconfigurable fabric has grown, the ability to perform multiple complex parallel applications, using a single device, has become a reality. For example, as early as 2003, the BMW Williams F1 team was 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 [1]. Today, FPGAs have increased product features and decreased product time to market and given system designers abilities that were only possible with the use of custom ASICs.

Dynamic partial reconfiguration 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. Future systems are likely to benefit from the development of effective systems that use DPR to provide for dynamic performance and power control.

Currently, when considering partial reconfiguration, the largest bottleneck is the time it takes to switch hardware resources. When a device is partially reconfiguring an area of the fabric, the fabric resources in the area are not available to the system. Therefore, increasing the speed at which the device is reconfigured increases the percentage of time of availability of the reconfigurable resource. As a result, we can explore applications that require high dynamic reconfiguration rates. An application of DPR in a dynamic arithmetic architecture has been recently reported in [2]. A recent application in DSP can be found in [3].

In what follows, we will focus our attention on DPR on Xilinx FPGA devices. More recently, DPR has also been made available on Altera FPGAs. While we will not discuss DPR on Altera FPGAs, we believe that the basic ideas introduced

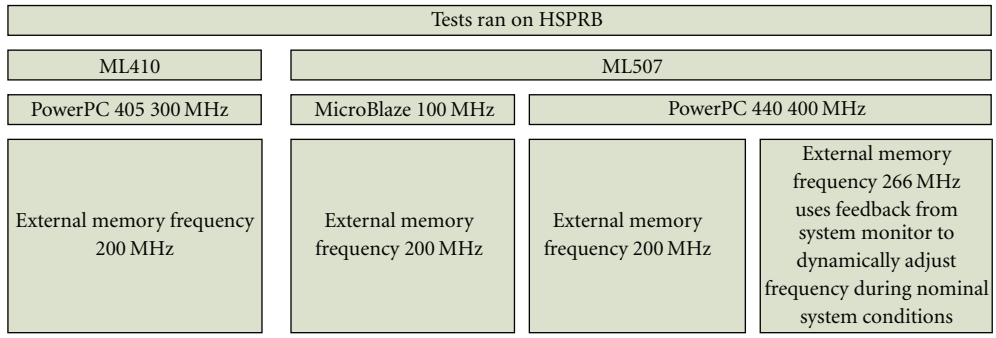

Figure 1: High-speed dynamic partial reconfiguration controller platforms.

here should be applicable to the future generations of FPGAs. Nevertheless, as with most high-performance hardware applications, we recognize that substantial effort may be needed to reimplement these ideas in future FPGAs.

To achieve DPR on Xilinx devices, the goal is to provide a high-speed interface that transfers the DPR bitstreams into the Internal Configuration Port (ICAP). Here, the ICAP port is used to transfer the bitstreams into the FPGA fabric. The most direct approach for implementing this interface is through the use of the Processor Local Bus (PLB) that provides an interface between the ICAP, the local processor, and several other peripherals in the Virtex-4 and Virtex-5 devices. Examples of high-performance implementations that make use of the PLB bus can be found in [4, 5]. Alternatively, in [6], the authors report on the use of the PLB bus for downloading partial bitstreams through Ethernet. On the other hand, Shelburne et al. [7] implemented a custom MetaWire interface to emulate a Network-on-Chip (NoC) approach that does not make use of the PLB bus. The approach makes use of expensive BRAM resources and over-clocking (144 MHz versus the Xilinx recommended 100 MHz clock) without monitoring proper FPGA operation. As a result, over-clocking may cause the system to lock up unexpectedly or cause other erratic behaviors. Liu et al. [8] also provided a BRAM-based approach. As in [7], this approach is limited by the size of the BRAM.

Unfortunately, the use of the PLB bus during the reconfiguration process will also prohibit its use by the processor or other peripherals. Thus, it is preferable to develop a DPR interface that avoids the use of the PLB bus. This allows the processor to perform other work while working with other peripherals.

We propose a new high-speed dynamic partial reconfiguration system that does not use the PLB bus during the DPR bitstream download to the FPGA fabric. Instead, we use interrupts to initiate the DPR process and also to inform the processor when the DPR bitstreams have been downloaded to the FPGA fabric. This is accomplished through the use of a multiport memory controller (MPMC) memory controller. Furthermore, for maximizing performance, we also introduce over-clocking with active feedback to ensure that the device voltages and temperatures are within nominal values. To avoid the limitations of BRAM memory, the proposed approach is based on DDR memory. Our approach leads to a bandwidth of 3.4 Gb/s for both reads and writes to the ICAP port.

For testing the system, we study an application of DPR in cryptography. The need for DPR in cryptography comes from applications that require an effective hardware architecture that can provide secure communications at different classification levels or from the need to accommodate several users simultaneously. Here, the basic idea is that each classification level (or user) will be effectively implemented in hardware by downloading its unique DPR bitstream to the FPGA. Then, the DPR system will be required to adapt the FPGA fabric to the dynamically varying requirements in classification levels and the number of users. Naturally, applications in dynamic arithmetic (e.g., [2]) or in Digital Signal Processing are also possible (e.g., [3]). On the other hand, the cryptography application considered here also allows us to validate our DPR system at the single bit level for the standard set of bit test vectors provided by the National Institutes of Standards and Technology (NIST) (see [9] for details).

The remaining of this paper is organized as follows. In Section 2, we provide basic background information on available FPGA platforms, their components, and their use in the proposed high-speed dynamic partial reconfiguration controller (HSDPRC) soft IP core. We the present related work in Section 3. In Section 4, we describe the design and development of the proposed high-speed dynamic partial reconfiguration controller (HSDPRC) soft IP core. Results are given in Section 5. Concluding remarks are given in Section 6.

2. Background

In this section, we provide basic background information on the partial reconfiguration controller facilities available on the Virtex-4 (ML410 board) and the Virtex-5 (ML507 board). Figure 1 gives a block diagram summarizing the four test systems that will be considered. For techniques associated with testing the systems considered here, we refer to [10–12].

2.1. FPGA Evaluation Boards. The evaluation boards are based on the Virtex-4 and Virtex-5. Virtex-4 includes

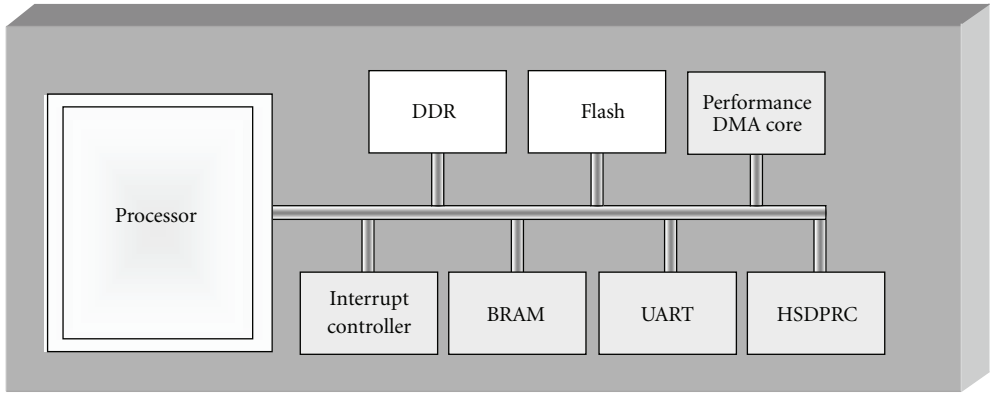

Figure 2: Common architecture components.

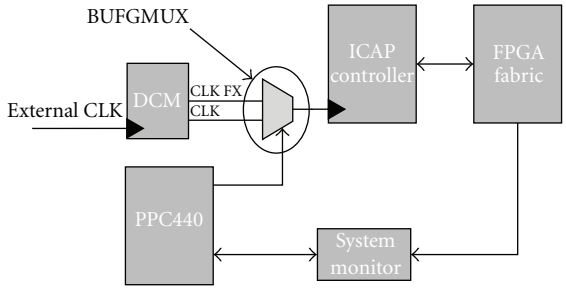

Figure 3: System monitor active feedback circuit.

a PowerPC PPC405 microprocessor. Virtex-5 includes a PowerPC PPC440 microprocessor, a hard IP crossbar memory controller, and System Monitor. Both can implement the uBlaze soft IP processor and soft IP crossbar memory controller. In what follows, we provide more details on the architecture.

2.2. 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.

2.2.1. 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, and DDR memory. Figure 3 shows how the blocks connect via the Processor Local Bus (PLB). For the proposed system, we also add the HSDPRC core and an interrupt controller.

2.2.2. Processor Local Bus (PLB). The PLB for Virtex-4 is 64-bits wide and 128-bits wide on Virtex-5 devices. The PLB 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. In addition, the PLB bus has an optional Device Control Register (DCR) slave interface that provides access to its bus error status registers.

2.2.3. Block RAM (BRAM). The BRAM is distributed throughout the FPGA fabric. It is thus considered to be a relatively expensive resource to use.

The proposed IDPR system uses BRAM to store the reset vector for the processor or as a memory to run the user application. The reset vector is the default location the processor will go to find the first instruction to execute after a reset (or startup). For all of the systems but the V5-PPC-266 MHz system, the BRAM stores the reset vector. For these systems after startup, the reset vector directs the processor to the user application stored in external DDR memory. For the V5-PPC-266 MHz system, to obtain optimal processor performance, the entire user application runs out of the system BRAM.

2.2.4. DDR. The DDR memory on the ML410 and ML507 boards are different. The ML410 board uses a slower 200 MHz capable DDR2 memory, while the ML507 uses a 266 MHz capable DDR2 memory. The LocalLink to memory interface ratio determines the speed of the memory interface. For both Soft IP Direct Memory Access (SDMA) and Hard IP Direct Memory Access (HDMA), the clock is a 1 to 1 or a 2 to 1 ratio of the memory clock. For all of the systems, the external memory is set to the maximum speed the memory could operate and not violate the ratio.

The DDR2 memory on both the ML410 and ML507 are standard DDR2 DIMM module with a 64-bit data bus. The ML410 PPC system DDR2 interface is configured to operate at a 100 MHz using a 1 to 1 LocalLink Clock ratio. This provides a 100 MHz clock to the LocalLink SDMA interface and HSDPRC core. For the ML507 MicroBlaze system, the DDR2 interface operates at 200 MHz, with a SDMA LocalLink to the HSDPRC core. The ML507 PowerPC system is configured with two different clocking schemes: (i) a 200 MHz DDR2 clock providing a 100 MHz LocalLink HDMA clock and (ii) a 266 MHz DDR2 clock providing a 133 MHz LocalLink HDMA clock.

2.2.5. FLASH. The flash is the static storage medium in the system, and stores the partial bitstreams when the system is not powered. During initialization, the system copies the partial reconfiguration bitstreams from the flash via the PLB into DDR memory. Compared to the DDR memory, the flash interface is significantly slower in both speed and access time. The flash is untested for the IDPR system.

2.2.6. UART. The UART in the IDPR system connects the host computer to the Processor Subsystem. The UART in the system controls the bit streams that are loaded in the partial reconfiguration regions, and to access configuration statistics. The UART is unneeded for systems that have other provisions for controlling the embedded processor.

2.2.7. Performance DMA Core. The Performance DMA core is used by the IDPRC to measure the throughput of a single HDMA using the PowerPC 440 crossbar or SDMA using the 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 set up and store the DMA performance calculations. The Performance DMA core is only needed by the IDPRC for testing, a fielded system would not need this core as it is only used to measure and document the speed of reconfiguration. To integrate the core into a design, several signals are 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.

(i) Setup_PERF_DMA sets the control register to trigger on a transaction.

(ii) Poll_Done_PERF_DMA is used to poll the status register until the DMA transaction is complete.

(iii) Tx_Transfer reports the results of the Tx transaction in Mbps.

(iv) Poll_Done_PERF_DMA reports the results of the Tx transaction in Mbps.

2.2.8. The Virtex-5 and PowerPC Platform. The Virtex-5 PowerPC 440 system utilizing the 266 MHz external DDR2 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 tested at 100 MHz, through Process, Voltage, and Temperature (PVT). On the other hand, most IC manufactures build in a tolerance when specifying the operating parameters. Taking this into account the ICAP port for this system is overclocked 33%, when the device is in nominal operating conditions.

The overclocking technique can be used in fielded systems. However, the clocking for the system would need to be controlled with a BUFGMUX primitive [11]. This primitive allows for glitchless clock switching.

2.3. Interrupt Controller. The systems interrupt controller interrupts the processor when a predefined programmable interrupt condition has occurred. For this application, the interrupt controller interrupts the processor after the LocalLink DMA operation has completed and the status registers have valid data.

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

2.4. Active Feedback Controller. The System Monitor active feedback circuit is shown in Figure 3. The circuit consists of a Digital Clock Manager (DCM), BUFGMUX, ICAP Controller, the FPGA Fabric, System Monitor, and the PowerPC 440 (PPC440) processor. The DCM produces both a 100 MHz and 133 MHz clock, from a 100 MHz reference. The BUGGMUX is a Xilinx primitive that allows the clock source driving the ICAP controller to switch without glitches, between the two DCM clocks. Before a reconfiguration operation, the FPGA System Monitor measures the voltage and temperature parameters of the FPGA and feed them back to the PPC440 processor. The PPC440 processor then, based on a predefined set of constraints, determines the clock frequency that the partial reconfiguration operation will use. Once the clock frequency is determined, the PPC440 set the BUFGMUX accordingly, before starting the partial reconfiguration operation.

3. Related Work

In this Section, we provide a summary of related research. At the end of the section, we also provide a summary of the proposed approach.

Claus et al. [4] present a dynamic partial reconfiguration system to perform DMA through the PLB ICAP controller. It is a modification of an earlier system discussed in [13] so as to allow the controller to work on any Virtex-II or Virtex-4 device. The ICAP controller acts as master to the PLB bus, allowing 64-bit data transfers in burst mode. In burst mode, the data bitstream is transferred in 16 64-bit words (256 bytes). With respect to each other, the burst mode transfers are not pipelined. The authors report maximum achievable data rates of 295.4 MB/s (see Table 1). However, the transfers are made through the PLB bus, thus not allowing use by other peripherals.

Manet et al. [5] implement an OPB ICAP controller on the Virtex-4 for using dynamic partial reconfiguration in Signal and Image Processing applications. The DPR system allows 32-bit write transfers at 100 MHz using the DMA. The implementation reported a throughput of 350 MB/s at a frequency of 100 MHz. Unfortunately, the implementation based on the OPB prohibits the use of any other peripherals during the reconfiguration writing process. Furthermore, there is no facility for DMA readback from the reconfiguration memory.

Shelburne et al. [7] implemented a custom MetaWire interface to fast DPR to emulate a Network-on-Chip (NoC) approach. This approach uses the FPGA BRAM to hold the reconfiguration bitstreams. Thus, the processor buses are avoided. This approach is interesting because it takes into

Table 1: Existing Dynamic Partial Reconfiguration Approaches.

Study	Device, bitstr. mem.	Method	Max ICAP speed
Claus et al.[4], 2008	Virtex-2P, DDR	OPB	4.77 MB/s @100 MHz
Claus et al. [4], 2008	Virtex-2P, DDR	PLB bus Custom ICAP DMA contr.	89.9 MB/s @100 MHz
Bomel at al.[6], 2009	Virtex-2P, DDR	Ethernet	50 MB/s @100 MHz
Claus et al.[4], 2008	Virtex-4, DDR2	OPB	5.07 MB/s @100 MHz
Claus et al.[4], 2008	Virtex-4, DDR2	PLB bus Custom ICAP DMA contr.	295.4 MB/s @100 MHz
Manet et al.[5], 2008	Virtex-4, ZBT SRAM or DDR	Custom ICAP DMA contr.	350 MB/s @100 MHz
shelburne et al. [7], 2010	Virtex-4, FPGA BRAM	MetaWire (custom)	219.31 MB/s @100 MHz
Bomel et al.[6], 2009	Virtex-4, DDR	Ethernet	50 MB/s @100 MHz
Liu et al.[8], 2010	Virtex-4, FPGA BRAM	PLB IP for BRAM Tx (fastest)	371.4 MB/s (max) @100 MHz

consideration many of the aspects that need to be addressed with data locality. Here, we use the term data locality to refer to the fact that the DPR bitstream is stored to the FPGA BRAM, which should be close to the FPGA fabric to be reconfigured. In addition this paper addresses the ability to overclock the ICAP port to 144 MHz, which is above the Xilinx recommended speed of 100 MHz. Unfortunately, the approach does not take into consideration the fundamental premise behind manufactures specifications, and how they relate to edge conditions the device will encounter when employed in environments outside of a controlled laboratory environment. As a result, overclocking may cause the system to lock up unexpectedly or other erratic behavior.

Bomel at al. [6] present a specific and quite simple protocol for partial reconfiguration over Ethernet. The motivation here is to create a better balanced hardware/software partitioning of hardware architectures. With no change at the protocol level, they were able to double the sustained speed over a standard 100 Mb/s (Mb = Megabits) Ethernet local Network. This is a good systems approach to DPR but lacks the ability to push the available bandwidth of the ICAP port due to the inherent speed and latency imposed by the Ethernet port resulted in a throughput of only 50 MB/s (400 Kbit/ms). The use of the Ethernet port introduces several issues associated with high network traffic. For example, packets can arrive out of order or may be dropped. The paper does not address these issues. Furthermore, Ethernet transfers use the PLB bus, a PLB/OPB bridge, and the PLB/OPB bridge to the ICAP. This approach requires a heavy burden on the processor.

Liu et al. [8] survey a variety of different designs that can achieve a range of runtime reconfiguration speeds. The first set of designs is based on the use of a slave interface for receiving control commands through the PLB bus and a PLB Master Burst interface for DMA that avoids HWICAP overhead. This first approach is similar to [4] in the use of 64-bit burst mode. The fastest approach in [8] is based on the use of BRAM PLB interfaces and achieves the maximum reconfiguration speed of 371.4 MB/s. Here, Block Memory was used to store the partial bitstream that was scheduled to be loaded into the fabric creating a virtual PR cashing technique. The approach creates the need for a look-ahead software decision point for the software algorithm to obtain the maximum throughput. In addition, the system requires a large amount of Block RAMs in the DPR controller to store the prefetch partial reconfiguration bitstream prior to loading the ICAP interface. In terms of resources, a disadvantage of the method is that the approach requires very expensive FPGA BRAM resources.

Related to our cryptographic application, we refer to [14]. In [14], the authors motivate the use of cryptographic algorithms for securing the DPR bitstreams. In a Virtex-5 implementation, the authors claimed a 1 Gbps throughput for the cryptography module (AES-GCM module) and a system throughput of about 800 Mbps. As in [8], high performance was achieved by storing the DPR bitstreams on Block RAMs on the FPGA.

The system described in this paper differs from all prior approaches in that (i) it provides interrupt support for communications between the DPR controller and the processor and (ii) uses over-clocking with active feedback to guarantee that the FPGA voltages and temperatures perform within nominal values.

The use of interrupts provides an efficient systems view of DPR. Interrupts are used for initiating DPR bitstream downloads and also to inform the processor when the PR process is complete. In the meantime, the processor will have access to other peripherals and be allowed to perform other tasks. As a result, the effective overhead of the DPR process can be substantially reduced as compared to other systems. Furthermore, this leads to an efficient application in cryptography.

4. Methodology

The requirement for portability of the high-speed dynamic partial reconfiguration controller (HSDPRC) led to its development as a soft IP core that uses the LocalLink DMA and Xilinx 32b-ICAP primitives. The core interface between the hard DMA (HDMA) and the ICAP uses the Xilinx Virtex-5 hard cross bar primitive. For the core interface between the Soft DMA (SDMA) and the ICAP uses the Xilinx multiport Memory Controller (MPMC) Soft IP core. To allow for ease of use, the controller has a set of internal ICAP functions. The functions allow the user to perform useful tasks such as reading specific address in the FPGA and configuring

internal read and write masks. The following section will present the approach used to develop the HSDPRC.

4.1. HSDPRC Core Design. The development of the HSDPRC soft IP core required an advanced two-part development flow (see Figure 4). The advanced flow splits the development into a traditional cycle accurate RTL simulation and a "hardware in the loop" simulation. This approach is necessary since the Xilinx ICAP simulation model does not provide a complete behavioral model; in that, it does not allow writing or reading of internal FPGA registers or configuration address. Therefore, it was not possible without actual hardware testing to see how the device ICAP will behave with a core. The following sections outline the development flow.

The simulation model consisted of a complete ML410 partial reconfiguration subsystem, including the software and driver used to implement the system. Using a complete system simulation allowed for development and testing of the behavioral VHDL model, as well as testing the HSDPRC driver. Two ModelSim simulations were run, a DMA test to verify the full function of the LocalLink DMA, and a Register test to verify the processor access to the registers internal to the HSDPRC core.

Figure 5 depicts the system model developed for the simulation. The system consists of an external DDR, PowerPC 405, MPMC, and HSDPRC model. The external DDR model has a 64-bit wide data bus running at 200 MHz. The DDR model and the MPMC crossbar core are connected. The MPMC crossbar splits the bidirectional external memory interface into 32-bit unidirectional data buses running at 100 MHz. Connected to the MPMC unidirectional data, buses are the PowerPC 405 and the Transmit and Receive FIFOs of the HSDPRC core. For the simulation, the Transmit and Receive FIFOs of the HSDPRC core loop data from the Receive FIFO to the Transmit FIFO. This configuration allowed for complete system simulation of the DMA operation and the data path to and from the HSDPRC core and external memory.

4.1.1. Hardware in the Loop Verification for the ICAP Interface. The Synplicity Identify Hardware in the loop development flow ensured the HSDPRC's ICAP interface operates as expected. The flow required a test fixtures used to simulate the core and an Identify project used to read back 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 run at both 80 MHz and 133 MHz. The sequence of individual states is as follows:

(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 0,

(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.

Here, we note that the NOP commands push the data through the internal ICAP data pipeline. In addition, the Frame Data Register (FDRO) is a read-only internal FPGA register. The FRDO provided readback data for configuration frames starting at the address specified in the Frame Address Register (FAR). The read length of the FDRO is

$$\text{FDRO\_Read\_Length} = 41 \cdot (\text{Frames\_stored} + 1) + 1, \quad (1)$$

Here, the extra 1 in the formula above represents reads and writes to an additional frame. This is to account for the frame buffer. The 41 is the number of words in a frame.

4.1.2. Connecting the HSDPRC Soft IP Core. Once the EDK cycle accurate RTL simulation and the Identify hardware-in-the-loop work independently as expected, the two systems were combined. This was accomplished by removing the transmit and receive FIFO loopback, in the cycle accurate RTL simulation. The ICAP wrapper, developed in the hardware in the loop system, was then connected to the transmit and receive FIFOs. After combining the two independent systems, the final system (the HSDPRC) was targeted to the test platforms.

4.1.3. Multimode Advanced Encryption Standard Partial Reconfiguration Module. For verifying the performance of the HSDPRC, we considered the Advanced Encryption Standard (AES). This algorithm comes with readily available National Institutes of Standards and Technology (NIST) test vectors and can be easily scaled by changing the size and modes of operation. In what follows, we provide some more details.

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 are extensively analyzed and are now used worldwide as is the case with its predecessor, the Data Encryption Standard (DES) [9].

The AES partial reconfiguration module (PRM) algorithm was designed and tested separately from the HSDPRC. For the implementations of the algorithm used, it passed all bit-for-bit test vectors provided by NIST. In addition, all of the IDPRC tests use the NIST test vectors used and verified during development of the AES algorithm. This ensured the accuracy of both the IDPRC implementation and removed possible errors in the separately developed AES implementations.

4.2. Partitioning the IDPRC Designs. The next section discusses how the IDPRC designs were mapped on the FPGA fabric. This step in the partial reconfiguration process is partitioning the design. Partitioning the design defines the area of the die used for the partial reconfiguration regions, the area used for the IDPRC and the bus macroplacement. The bus macros provide the communication between the

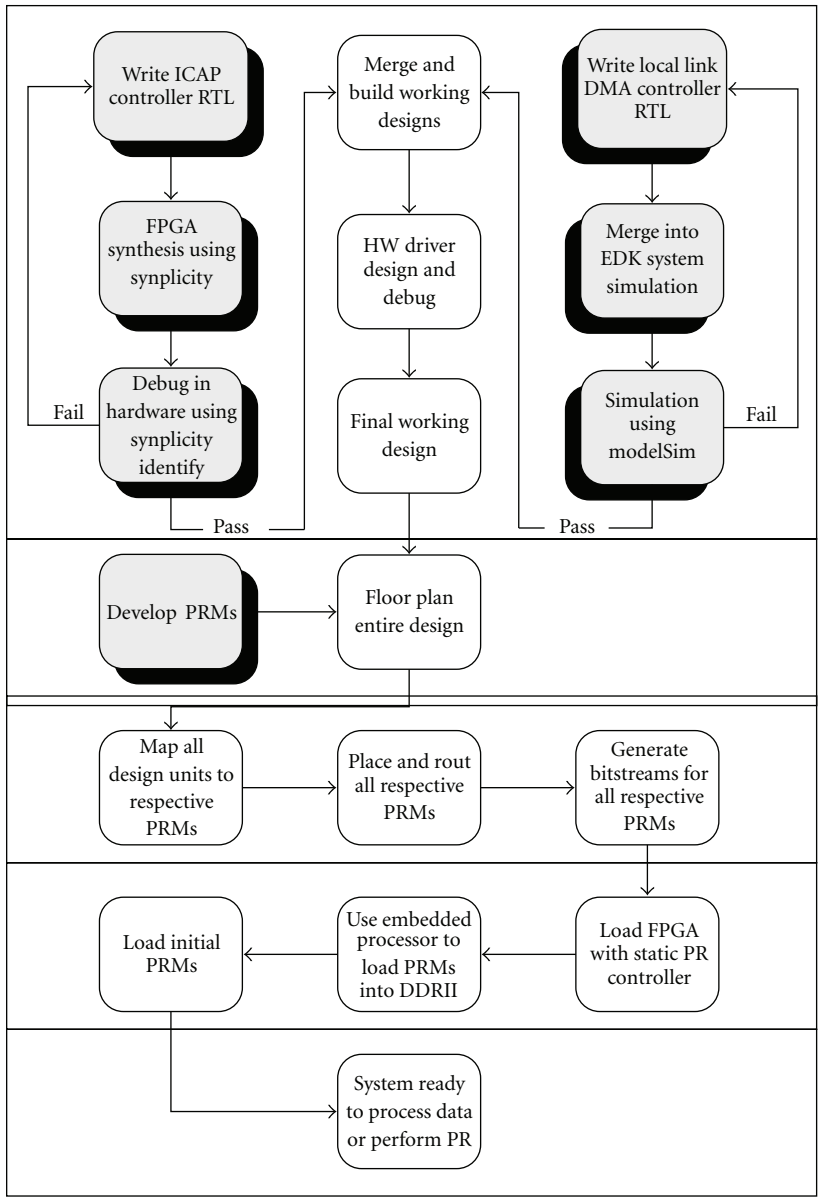

Figure 4: High-speed dynamic partial reconfiguration controller (HSDPRC) development flow.

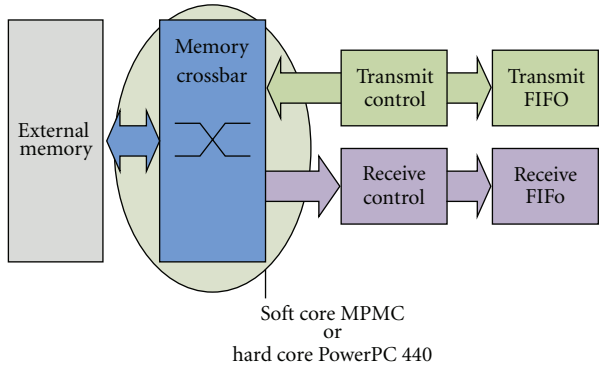

Figure 5: DMA controller verification on the Virtex-4FX.

static and dynamic regions of the FPGA fabric. This process uses the Xilinx PlanAhead tool.

4.2.1. Partitioning the Virtex-4 Design. The Virtex-4 IDPRC design consists of three regions. The design has two PR regions and a static region with the IDPRC. The two PR regions have the same identical available FPGA resources. Each of the cryptocore regions have different implementations of the AES algorithm.

Figure 6 shows the implementation of the design after it has been placed and routed. The implementation in the image has identical AES algorithms loaded into each of the PR regions. The static design is the PowerPC 405 100 MHz system. Here, we note that only one of the PowerPC processors is used out of the 2 available in the xc4vfx60ff1152.

4.2.2. Partitioning the Virtex-5 Design. The Virtex-5 IDPRC designs consist of several PR regions. The PR regions have the same identical available FPGA resources. In the image above the PR regions on the Virtex-5 xc5v70FXT, we have four cryptography cores and the IDPRC shown as a user logic core. Each of the cryptography core regions reconfigured with different implementations of the AES algorithm. The user logic core is static and does not change.

Figure 7 shows the implementation of the design after being placed and routed. The implementation in the image has identical AES algorithms loaded into each of the PR regions. The static design is the PowerPC 440 266 MHz system.

5. Results

The test procedure and software applications give the user the ability to actively monitor and test the IDPRC in the FPGA. Figure 8 shows the test process.

The first step when testing the IDPRC system is to configure the FPGA by writing the bitstream to the device. After configuration, the EDK shell connects to the FPGA subsystem. The connection to the subsystem uploads the partial bitstreams to the external DDR2 memory. After all of the partial bitstreams used for the test are loaded, the software test application uploads to the processor. In a file-based system, the load process would use a boot loader and an embedded software application. After this process, the processor and the FPGA are ready to start.

When the software 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 partial reconfiguration command. The command is a number from 1 to 4 (up to 2 for Virtex-4). The number indicates the region to reconfigure. Once the user enters a number and sends the command to the IDPRC, the partial reconfiguration process starts.

After performing the partial reconfiguration, a PRM verification test checks the CRC values of the individual configuration frames internal to the FPGA. The test performs a readback operation via the ICAP port. The test procedure verifies the HSDPRC, AES implementation, and IDPRC.

The final step is used to verify the operation of the AES PRM by running a Built-in self-test (BIST). The self-test

Figure 6: Virtex-4 FPGA Editor Screen Capture 1. The diagram clearly shows the two partial reconfiguration (PR) regions.

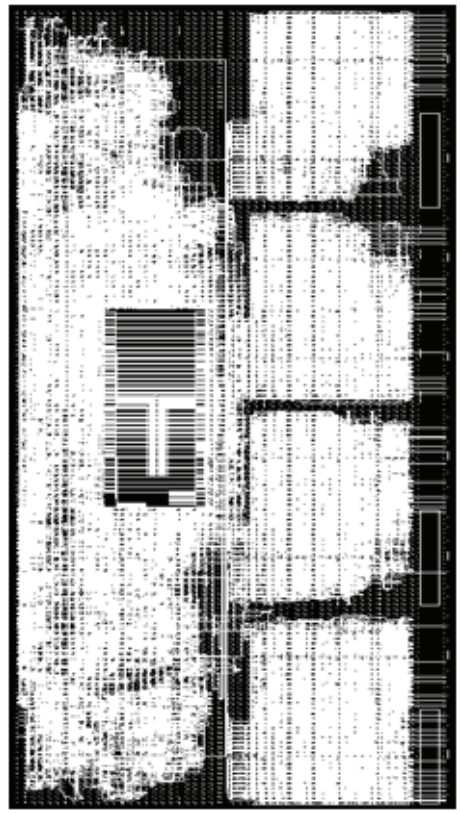

Figure 7: Virtex-5 FPGA Editor Screen Capture 2. The diagram clearly shows the four PR regions.

Table 2: HSDPRC write and read throughput. A DMA controller is used during dynamic partial reconfiguration. An interrupt is used to signal completion. Despite the fact that they share the same ICAP frequency of 100 MHz with previous devices (Virex-II and Virtex-4), the Virtex-5 results can be further improved to run at the maximum possible speed as described in Section 5.2.

Platform	Processor, System Freq	Memory Controller	TX Freq	RX Freq
ML410 (Virtex 4)	PowerPC 405, 200 MHz	MPMC	177.4 MB/s @100 MHz	180.0 MB/s @100 MHz
ML507 (Virtex 5)	MicroBlaze, 200 MHz	MPMC	178.6 MB/s @100 MHz	181.0 MB/s @100 MHz
ML507 (Virtex 5)	PowerPC 440, 200 MHz	PPC440MC	335.9 MB/s @100 MHz	340.4 MB/s @100 MHz
ML507 (Virtex 5)	PowerPC 440, 266 MHz	PPC440MC	418.5 MB/s @133 MHz	424.6 MB/s @133 MHz

writes and reads NIST encrypt and decrypt test vectors to and from the AES PRM. The vectors communicate between the IDPRC and the AES PRM through the bus macros. Here, we note that the bus macros connect between the PLB and AES PRM.

5.1. Reconfiguration Speed Measurements. Table 2 presents the speed measured for the ICAP write and read throughput using the proposed method. Dynamic partial reconfiguration uses DMA controller.

To give perspective on the results, note that the ICAP core is 4 bytes wide and operates at 100 MHz. Thus, at this frequency, the maximum bandwidth that can be achieved is at 3.2 Gbits/s or 400 MB/s. As seen in Table 2, the Virtex-5 implementation with the PPC440MC MPMC memory controller gave the best results at 100 MHz. This implementation gave 335.9 MB/s DPR speed which is at 84% of the maximum of 400 MB/s that can be achieved. Furthermore, as we will describe next, this can be significantly improved by proper over-clocking.

The HSDPRC cores' device utilization for a Virtex-5 is given in Table 3. The table shows only the size of the HSDPRC core. To implement complete systems, other resources such as the processor infrastructure and partial reconfiguration regions are needed.

5.2. Overclocking with Active Feedback. As discussed in Section 2.2.8, the Virtex-5 devices allow us to maximize the performance of the ICAP using over-clocking. This system uses feedback from the System Monitor IP to determine if the device voltage and temperature parameters are within nominal operating conditions.

When the ICAP is ran at the Xilinx specification over PVT, the HSDPRC increased the speed of partial reconfiguration. To understand the performance limits using over-clocking, recall that the ICAP core is 32 bits wide that runs at the modified clock frequency. In this case, from Table 2, we have a maximum clock speed of 133 MHz. Thus, at this clock speed, the theoretically maximum possible speed is at 532 MB/s. In our system, we measured a maximum

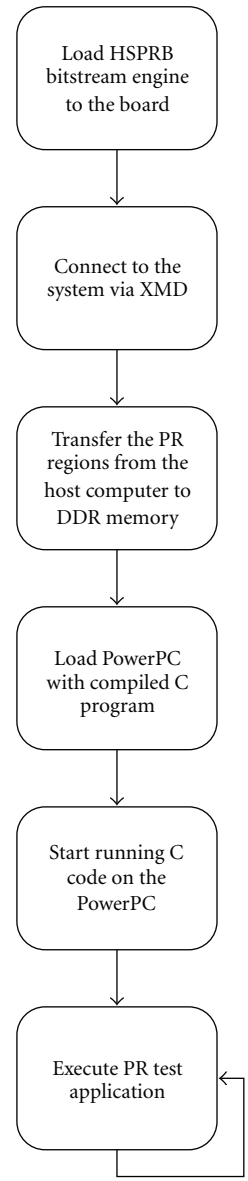

Figure 8: Test procedure state flow diagram.

Table 3: HSDPRC device utilization for Virtex-5.

Virtex-5 device utilization for HSDPRC.	
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	1530
Flop pairs	

DPR speed of 418.5 MB/s and readback at 424.6 MB/s or approximately 3.4 Gbits/s.

6. Conclusion

This paper presented a high-speed DPR system that avoids the use of the PLB bus during DPR bitstream download to the FPGA. The approach allows the processor's full access to other peripherals during most of the DPR process. Furthermore, the paper summarized an active feedback subsystem that allows over-clocking while guaranteeing that the FPGA operates within safe levels.

In terms of performance, the proposed system achieved a DPR rate of 3.4 Gigabits/s. This represents the highest DPR rate achieved so far.

Acknowledgments

The authors would like to acknowledge the support of the Xilinx Corporation for this research. They also acknowledge Steve Trimberger at Xilinx for reviewing the paper and providing helpful suggestions. They would also like to acknowledge Ken Weidele for nominating this research for the Xilinx Ross Freeman Award 2010 for Technical Innovation.

References

[1] L. Boland, "Formula 1 racing: the xilinx advantage," *Xcell Journal*, pp. 46–49, 2003.

[2] G. A. Vera, M. Pattichis, and J. Lyke, "A dynamic dual fixed-point arithmetic architecture for fpgas," *International Journal of Reconfigurable Computing*, vol. 2011, Article ID 518602, 19 pages, 2011.

[3] D. Llamocca, M. Pattichis, and G. A. Vera, "Partial reconfigurable fir filtering system using distributed arithmetic," *International Journal of Reconfigurable Computing*, vol. 2010, Article ID 357978, 14 pages, 2010.

[4] C. Claus, B. Zhang, W. Stechele, L. Braun, M. Hübner, and J. Becker, "A multi-platform controller allowing for maximum dynamic partial reconfiguration throughput," in *Proceedings of the IEEE International Conference on Field Programmable Logic and Applications, (FPL 2008)*, pp. 535–538, Heidelberg, Germany, September 2008.

[5] P. Manet, D. Maufroid, L. Tosi et al., "An evaluation of dynamic partial reconfiguration for signal and image processing in professional electronics applications," *Eurasip Journal on Embedded Systems*, vol. 2008, no. 1, Article ID 367860, 11 pages, 2008.

[6] P. Bomel, J. Crenne, L. Ye, J.-P. Diguet, and G. Gogniat, "Ultra-fast downloading of partial bitstreams through ethernet," *Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics*, vol. 5455, pp. 72–83, 2009.

[7] M. Shelburne, C. Patterson, P. Athanas, M. Jones, B. Martin, and R. Fong, "MetaWire: using FPGA configuration circuitry to emulate a network-on-chip," *Institution of Engineering and Technology Computers & Digital Techniques*, vol. 4, no. 3, pp. 159–169, 2010.

[8] M. Liu, W. Kuehn, Z. Lu, and A. Jantsch, "Run-time partial reconfiguration speed investigation and architectural design space exploration," in *Proceedings of the 19th IEEE International Conference on Field Programmable Logic and Applications, (FPL 2009)*, pp. 498–502, Prague, Czech Republic, September 2009.

[9] US Federal Government, "Advanced encryption standard," 2001, http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[10] J. Williams and N. Bergmann, "Embedded Linux as a platform for dynamically self-reconfiguring systems-on-chip," in *Proceedings of the International Conference on Engineering of Reconfigurable Systems and Algorithms, (ERSA 2004)*, pp. 163–169, Las Vegas, Nev, USA, June 2004.

[11] Xilinx, "Virtex-5 fpga user guide," 1999, http://www.xilinx.com/support/documentation/user guides/ug190.pdf.

[12] B. Blodget, P. James-Roxby, E. Keller, S. McMillan, and P. Sundararajan, "A self-reconfiguring platform," in *Proceedings of the 13th IEEE International Conference on Field Programmable Logic and Applications, (FPL 2003)*, Lisbon, Portugal, September 2003.

[13] C. Claus, F. H. Müller, J. Zeppenfeld, and W. Stechele, "A new framework to accelerate virtex-II Pro dynamic partial self-reconfiguration," in *Proceedings of the 21st IEEE International Parallel and Distributed Processing Symposium, (IPDPS 2007)*, Long Beach, Calif, USA, March 2007.

[14] Y. Hori, A. Satoh, H. Sakane, and K. Toda, "Bitstream encryption and authentication with AES-GCM in dynamically reconfigurable systems," in *Proceedings of the IEEE International Conference on Field Programmable Logic and Applications, (FPL 2008)*, pp. 23–28, Heidelberg, Germany, September 2008.