Extending the SIMPPL SoC Architectural Framework to Support Application-Specific Architectures on Multi-FPGA Platforms

David Dickin and Lesley Shannon
School of Engineering Science, Faculty of Applied Sciences
SFU, Burnaby, B.C.
email: {drdickin, lshannon}@sfu.ca

Abstract

Process technology has reduced in size such that it is possible to implement complete application-specific architectures as Systems-on-Chip (SoCs) using both Application-Specific Integrated Circuits (ASICs) and Field Programmable Gate Arrays (FPGAs). However, the reconfigurable nature of an FPGA results in lower logic density, such that large, complex applications require multi-FPGA implementation platforms.

Although designing SoCs is challenging, SoC models such as Systems Integrating Modules with Predefined Physical Links (SIMPPL) exist to facilitate the design process. SIMPPL leverages defined physical interfaces and communication protocols to enable rapid system-level integration for application-specific architectures. This paper presents a "SIMPPL Repeater" that enables the SIMPPL SoC architectural framework to be used for systems spanning multiple FPGAs. The SIMPPL Repeater abstracts inter-chip communication, allowing designers to treat a multi-FPGA platform as a single large reconfigurable fabric and focus on their application-specific architecture.

1. Introduction

Field Programmable Gate Arrays (FPGAs) are commonly used as platforms for hardware acceleration via application-specific architectures. However, for large, complex applications such as molecular dynamics [1], seismic imaging [2], and logic emulation [3], a single FPGA does not provide sufficient resources for a complete implementation in hardware. This has led to the development of scalable multi-FPGA platforms that provide sufficient resources for implementing these types of systems [4] [5].

The design of application-specific architectures targeting a single FPGA is non-trivial. Targeting a multi-FPGA platform complicates the design even further. Designers must appropriately partition the system across multiple chips and properly implement inter-FGPA communication. These challenges increase the time and effort required to implement the application. Thus, it becomes prudent to consider time-efficient design methods as the size and popularity of multi-FPGA systems grow.

Systems Integrating Modules with Predefined Physical Links (SIMPPL) is an architectural framework for SoC designs that reduces the integration time of an application-specific architecture by more than an order of magnitude [6], [7]. SIMPPL makes use of the *Computing Element (CE)* abstraction, whereby a module's datapath is separated from its system-level control and communication. This separation enables the use of a customizable, lightweight controller that can be tailored to the requirements of each module in the system. Only the controller's program needs to change in order to integrate the module into a new system, a feature that greatly improves the opportunities for Intellectual Property (IP) reuse. SIMPPL was developed for SoC designs, i.e. a single Application Specific Integrated Circuit (ASIC) or FPGA, and was not intended for multi-chip platforms.

This paper presents the *SIMPPL Repeater (SR)*, which enables the use of SIMPPL on multi-FPGA platforms while allowing efficient inter-FPGA communication. Although this work focuses on designs for multi-FPGA platforms due to the lower logic density of FPGAs, the concepts are also applicable to multi-ASIC platforms. The SR replaces the lightweight controller in a typical CE to create a *SIMPPL Repeater Computational Element (SRCE)* that extends the SIMPPL architectural framework to multi-FPGA platforms. This extension to SIMPPL is accomplished by abstracting the off-chip interface such that when the SRCE is integrated with normal CEs in a system, the designer can treat a multi-FPGA platform as one large FPGA. Treating the platform as a single FPGA allows designers to focus on their application-specific architectures instead of inter-FPGA communication. A simple multi-FPGA system is used to demonstrate the SR and the SRCE. The SR consumes 13% of the SRCE's Flip Flops (FFs) and 40% of its Look-Up-Tables (LUTs), maintaining the lightweight aspect of other SIMPPL controllers.

The rest of the paper is organized as follows. Section 2 presents background on multi-FPGA systems and the SIMPPL architectural framework. Section 3 describes the SR and the SRCE and how they fit into the SIMPPL system model. Section 4 presents an SRCE used in a simple demonstration

system and the results garnered from the system. Finally, we draw conclusions and outline areas for future work in Section 5.

2. Background

This work's focus is on extending the SIMPPL SoC architectural model to multi-FPGA platforms. As such, a summary of how SIMPPL works is provided along with a brief description of other efforts to ease system implementation on multi-FPGA systems

2.1 Overview of SIMPPL

SIMPPL is an SoC architectural model that facilitates rapid integration of modules in an application-specific architecture. Modules in a SIMPPL system are termed Computing Elements (CEs) and are implemented using either dedicated hardware (hardware CE) or as software running on a processor (software CE). The CE abstraction provides uniform physical interfaces and communication protocols between CEs. CEs are connected together using unidirectional point-to-point links. These links are implemented as asynchronous First-In First-Out buffers (FIFOs) Figure 1 a) illustrates a high-level view of a system modelled using SIMPPL.

The hardware CE abstraction separates the communication and system-level control from a hardware module's datapath, as shown in Figure 1 b). Hardware CEs are composed of:

1. A *Processing Element (PE)*, which encapsulates the module's datapath (e.g. a 32-bit multiplier
2. A *SIMPPL Controller*, which executes a local program and provides system-level control for the PE's operation and inter-CE communication
3. A *SIMPPL Control Sequencer (SCS)* that contains the local program for the controller.

SIMPPL Controllers are lightweight microcontrollers that have a configurable Instruction Set Architecture (ISA), allowing them to operate with a wide variety of PEs. The tradeoff for using a SIMPPL Controller, as opposed to custom control logic, is an increase in resource usage and a possible increase in the module's latency due to the controller's fetch, decode, and execute operations. However, the module's throughput is unaffected.

There are several variants of SIMPPL Controllers that have been optimized for different PE types. The Full Controller supports the full ISA to allow sending and receiving data to the PE (e.g. memory). The Producer has a reduced ISA and only supports sending data from the PE, whereas the Consumer only handles receiving data from the PE. The Producer and Consumer are commonly used in conjunction with each other for pipelined PEs (e.g. a multiplier). This allows data to be simultaneously sent and received, whereas the Full Controller is only capable of executing one operation at a time.

Communication between CEs is packet-based. An example of a SIMPPL packet is shown in Figure 2. In this paper, each SIMPPL packet is composed of a 32-bit program word and a control bit, making the effective word size of the packet 33 bits. If the control bit of a packet is asserted, then the program word is an instruction word. All SIMPPL packets start with an instruction word, which contains the opcode of the SIMPPL instruction and the number of data words (NDW) in the packet. If the control bit is not asserted then the program word is either a data word or an address word. Only some of the SIMPPL instructions require an address word, which follows directly after the instruction word in the packet.

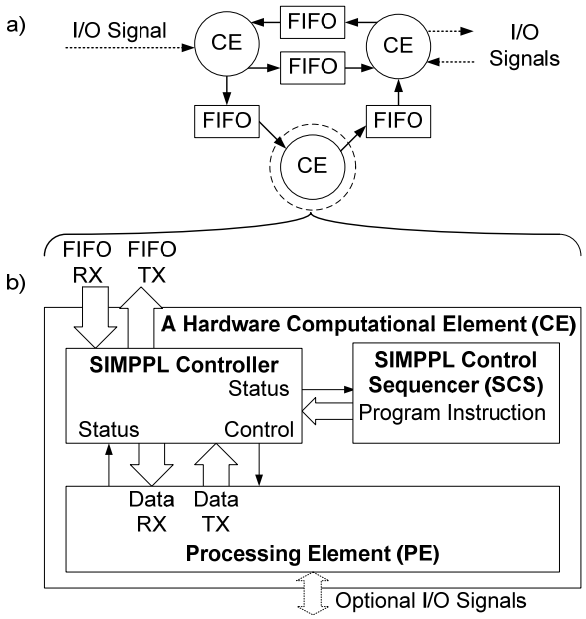

Figure 1. a) An SoC modelled using SIMPPL. b) Hardware CE block diagram.

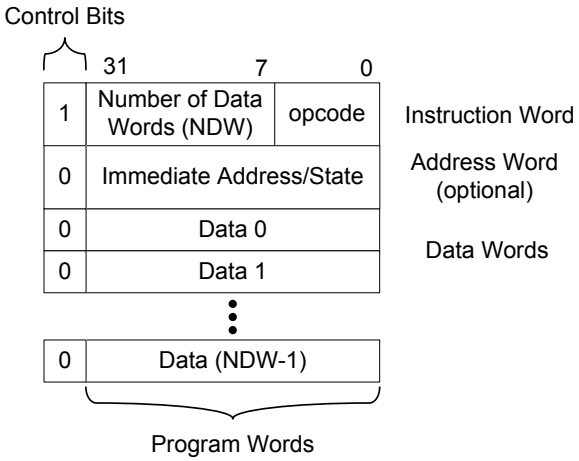

Figure 2. SIMPPL packet format.

2.2 Multi-FPGA System Design Overview

There is considerable previous work focused on board level routing architectures for multi-FPGA platforms and automated methods of implementing a single, large system design across the multiple FPGAs

[8]-[11]. Inter-chip communication typically has increased latency and decreased throughput when compared to intra-chip connections. Thus, multi-FPGA implementations suffer from performance bottlenecks if the system is partitioned poorly or the platform architecture is ill-conceived.

The research listed above is concerned with designing multi-FPGA platforms and tools that intelligently implement systems on multi-FPGA platforms. SIMPPL provides a framework for SoC architectures and focuses on reducing system design time and facilitating IP reuse. The extended SIMPPL model can be used in conjunction with multi-FPGA platform specific CAD tools to provide a flexible design framework for different reconfigurable platforms.

SIMPPL is contrasted with other IP reuse frameworks and SoC models in [6]. The proposed extension to SIMPPL abstracts multi-FPGA platforms as a single reconfigurable fabric, allowing SIMPPL to remain platform agnostic. At the time of writing the authors are unaware of any other attempts to extend SIMPPL beyond its SoC origins or other system architectural frameworks that abstract multi-FPGA platforms as a singular reconfigurable platform. Instead, applications implemented on multi-FPGA platforms use a custom architecture to avoid the overhead associated with using a generalized framework such as SIMPPL.

3. The SIMPPL Repeater

In this section, we present the new SR and SRCE and how they extend the SIMPPL architectural framework to multi-FPGA platforms.

3.1 SIMPPL I/O

All communication with I/O peripherals in a SIMPPL system is encapsulated using CEs. This allows other CEs to use the SIMPPL protocol to communicate with off-chip peripherals. The I/O encapsulating CEs, termed I/O CEs, act as interpreters for on-chip CEs communicating with off-chip peripherals.

Figure 3 a) is provided as an example of a SIMPPL system that includes an off-chip peripheral in the form of an external memory chip. The on-chip CE A can send SIMPPL packets to the I/O CE requesting reads and writes from memory using the SIMPPL protocol. The I/O CE interprets these requests and treats the external memory as a typical PE whose ports happen to be I/O pins on the FPGA. This configuration allows the external memory to be modelled as an on-chip resource, as in Figure 3 b). Incorporating off-chip peripherals into SIMPPL systems as I/O CEs is necessary when the off-chip peripherals are not capable of interpreting the SIMPPL packet communication protocols.

3.2 SIMPPL Inter-FPGA Communication

Consider a system that consists of two CEs, A and B, that need to be connected together in a system as shown in Figure 4 a). If CEs A and B are too large to be implemented together on a single FPGA, then a multi-FPGA platform is required. Obviously, this will necessitate inter-FPGA communication. The I/O CEs C and D are added to encapsulate the PEs that handle transferring data between the two FPGAs. The multi-FPGA implementation, shown in Figure 4 b), is consistent with SIMPPL's method of incorporating I/O into the system. However, the fact that the off-chip peripheral is an FPGA with CEs gives rise to a special situation.

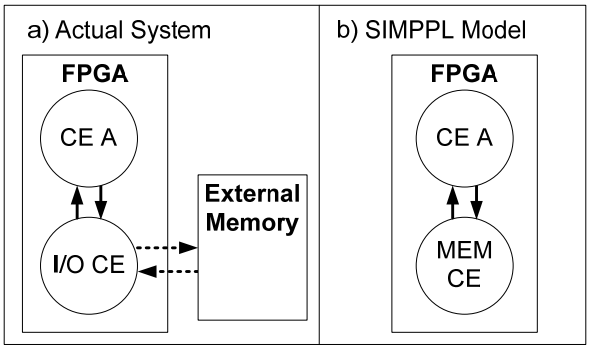

Figure 3. a) The actual structure of a SIMPPL system containing an I/O CE. b) The SIMPPL model view of the same system.

Both CEs A and B communicate using the SIMPPL protocol. Therefore, there is no need for the I/O CEs C and D to act as interpreters. Doing so complicates communication between A and B. In order for A to send a packet to B, the packet must first go through both C and D. The SIMPPL Controllers in C and D will assume any packets they receive contain instructions that should be executed by them. This execution is not desirable as the original packet from A, and the instruction it includes, is intended for B. Therefore, both C and D must be specially configured to ensure that they do not execute the instructions contained in packets they receive. Instead, they should forward all received packets. Forwarding abstracts the inter-FPGA communication, such that A and B appear to be connected together on a single FPGA using the SIMPPL model depicted in Figure 4.

While it is possible to implement this forwarding behavior in C and D by using a custom SIMPPL instruction or additional logic, these approaches are inefficient. Employing a programmable microcontroller to implement a dedicated function introduces latency into the forwarding operation and consumes more FPGA resources than necessary. In the event that a SIMPPL system must span several FPGAs, it becomes prudent to consider a more efficient method of inter-FGPA communication.

3.3 The Role of the SIMPPL Repeater

The SR is used to replace the SIMPPL Controller and SCS in an I/O CE that encapsulates an inter-FPGA communication PE, i.e. CEs C and D from Figure 4 b). This new type of I/O CE containing an SR is termed a SIMPPL Repeater Computational Element (SRCE), and is included in a SIMPPL system as seen in Figure

4 c). Similar to the function of a repeater in telecommunications, the SRCE receives an input signal and then retransmits it such that the output can travel long distances.

Unlike the SIMPPL Controller, an SR does not interpret the instruction word of every packet it receives. If a packet is received, then the appropriate action for the SR is to forward it as soon as possible. By implementing a dedicated forwarding function, the SIMPPL Repeater trades-off the flexibility of the combined SIMPPL Controller and SCS for a decrease in latency and a reduction in resource usage. Fewer resources are required as the programmable aspects of the SIMPPL Controller and SCS are discarded. In addition, the fetch, decode, and execute operations of a SIMPPL Controller are removed, reducing the latency of operation.

The SRCE abstracts the inter-FPGA communication details from the rest of the SIMPPL system. The SIMPPL model of two CEs on separate FPGAs with SRCEs becomes identical to that of the two CEs implemented on the same FPGA. Due to this generalization, the SRCE transparently extends SIMPPL to multi-FPGA platforms. From the modelling perspective, CEs do not need to be aware that they are communicating off-chip. They communicate in the same manner as they would with any other on-chip CE. The only noticeable difference during operation is the increased latency between CEs using inter-FPGA communication.

3.4 SIMPPL Repeater Computational Element

A block diagram of the SRCE is shown in Figure 5. The two sub-modules of the SRCE are the SR and the PE. The SR's function is to handle transferring data between the FIFOs and the PE, as well as recognizing instructions words to ensure packets are properly transmitted. The PE performs the actual inter-FPGA communication and I/O specific protocols.

The SR is comprised of FIFO read/write logic customized for the SIMPPL packet format. The read logic is contained in the *RX Block* from Figure 5, while the write logic is implemented in the *TX Block*. Both the RX Block and TX block have small state machines to control transferring data between the FIFOs and the PE. The RX Block and the TX Block operate independently to allow the simultaneous flow of data from the RX FIFO to the PE and from the PE to the TX FIFO. This is necessary to support a full-duplex communication channel.

The SR and PE are connected using only a few signals, as shown in Figure 5. There are PE and SR versions of the *Rdy*, *Valid*, *SoF* (Start-of-Frame), and *EoF* (End-of-Frame) control signals. The *Rdy* signals show that a core can accept new data. *Valid* indicates that the data the core is outputting is valid. The assertion of *SoF* identifies the beginning of a SIMPPL packet, and thus the control bit of a SIMPPL packet word, while *EoF* identifies the end of the packet. In addition to these control signals, there are the Data RX and Data TX data busses that are used to transfer the SIMPPL packet program words.

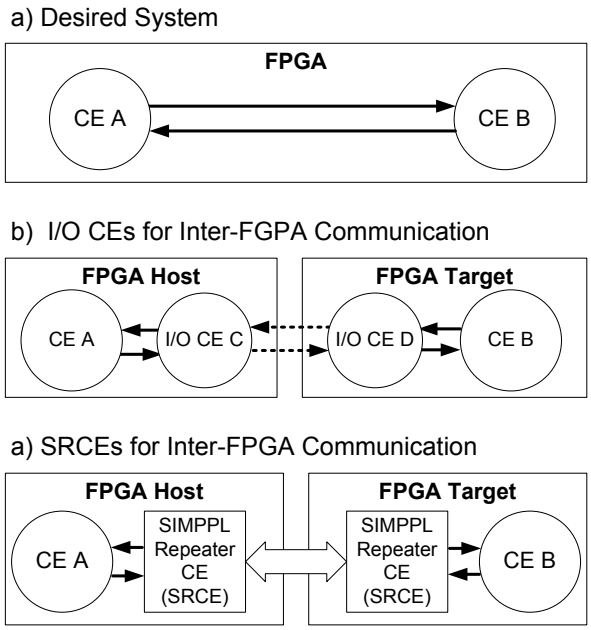

Figure 4. a) Desired SIMPPL system. b) Using I/O CEs to span the system across multiple FPGAs. c) Using SRCEs instead of I/O CEs.

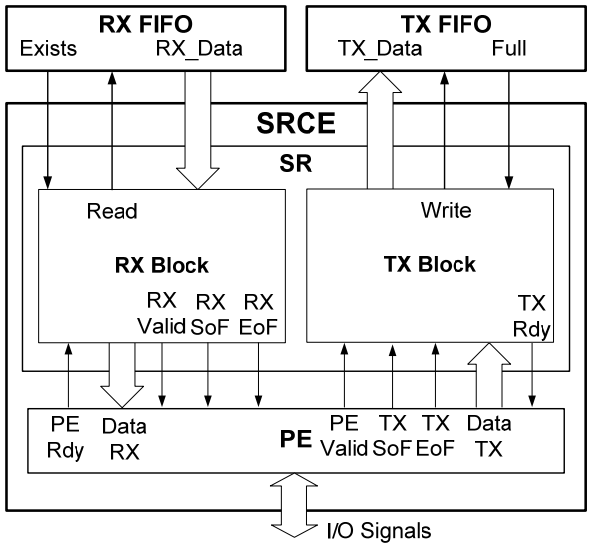

Figure 5. SIMPPL Repeater Computational Element (SRCE) block diagram.

There are many possible communication protocols and physical interfaces that can be used to transfer data between FPGAs and it is the PE (see Figure 5) that implements these I/O specific protocols. Thus, the architecture of the PE is dependent upon the I/O type used. The SR-PE interface was designed to be simple and generic. The onus is placed on the PE to support the interface. If this is not possible, then the SR can be customized to support any special requirements the PE might have for a specific I/O type. In general, this customization is not anticipated

as the sole function of the SRCE's PE is to transfer data between FPGAs.

4. Results

In this section, a sample multi-FPGA system demonstrating the SIMPPL Repeater is presented.

4.1 Multi-FPGA Verification System

The system used to evaluate the SIMPPL Repeater is shown in Figure 6. The implementation platform consists of two XUPV2P development boards [12] connected using a 50 cm SATA cable. Each XUPV2P board contains a Xilinx XC2VP30 Virtex™-II Pro FPGA [13]. One FPGA is referred to as the *Host FPGA,* the other as the *Target FPGA*. The Host FPGA contains a *Test CE*, and an SRCE. The Test CE is connected to the SRCE using a Fast Simplex Link (FSL), a Xilinx specific FIFO [14]. The Target FPGA contains an SRCE and an FSL configured to loopback any data it receives from the SRCE.

In both SRCEs, the PE implementing the inter-FPGA communication channel contains a Multi-Gigabit Transceiver (MGT). MGTs are commonly used in current multi-FPGA platforms [4], [5] as they provide excellent data throughput with low latency and few I/O pins. This MGT PE operates at 75 MHz and provides a full-duplex channel that can ideally transmit and receive data at 1.2 Gigabits per second (Gbps). The PE is composed of a RocketIO™ Transceiver (An MGT) [15], an Aurora Core [16], and some additional circuitry for datapath manipulation and buffering received data.

Two different Test CEs were used in the evaluation system. A Latency Measurement Core (LMC) is used to measure the latency of the channel. The latency of a one word SIMPPL packet's round-trip time is measured using a counter in the LMC. The counter is started when the LMC writes the word to an FSL, and is stopped once LMC reads the word back out of an FSL. The counter value is then halved to give the one-way trip time. The other Test CE is a SIMPPL Packet Generator (SPG). The SPG generates the full range of SIMPPL packets with variable sizes to provide realistic traffic for the SRCEs. The SPG checks for discrepancies between the data it transmits and receives to ensure no errors occurred during transit. The throughput of the system was measured using a counter (not shown) that counts the number of data words sent to the PE in a set amount of time. During this set time period the SRCE input is saturated to avoid a stall.

4.2 Resource Usage

The FPGA resources used by the MGT SRCE and its SR are presented in Table 1. The SR consumes 13% of the SRCE's Flip Flops (FFs) and 40% of SRCE's Look-Up-Tables (LUTs). The remainder of the SRCE's resource usage is due to the PE. The SR's two primary sub-modules are state machines with very few states. Fewer states imply that fewer FFs are required. The majority of the SR's functionality is control logic, which is implemented using LUTs. Full duplex communication is supported, and thus two state machines are required, which explains the higher percentage of LUT usage.

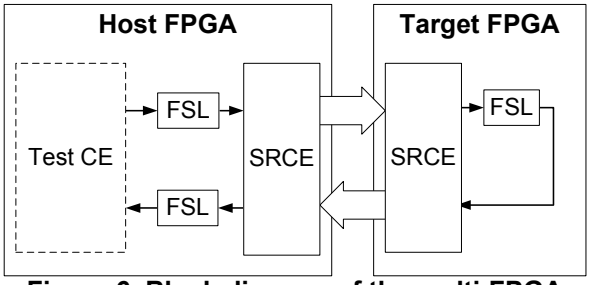

Figure 6. Block diagram of the multi-FPGA SRCE evaluation system.

Table 1. SRCE resource usage

Name	FF	% of SRCE FFs	LUT	% of SRCE LUTs
SRCE	465	100%	664	100%
PE	404	87%	398	60%
SR	61	13%	266	40%

Table 2. Comparison to SIMPPL Controllers

Name	FF	LUT	Max Freq
SIMPPL Repeater	61	266	241 MHz
SIMPPL Full Controller	115	346	283 MHz
SIMPPL Producer Controller	125	355	285 MHz
SIMPPL Consumer Controller	117	277	287 MHz

A comparison between the resource usage of the SR and the previously published SIMPPL Controllers is given in Table 2 [6]. The function of the SR, simultaneously reading and writing from FIFOs and PE, is comparable to the functionality provided by a SIMPPL Producer combined with a SIMPPL Consumer. Thus, a comparison of the SR resource usage versus the resource usage sum of a Producer and a Consumer is justifiable. The SR uses 25% the FFs and 42% of the LUTs of the combined Producer and Consumer. The smaller footprint of the SR is attributable to its dedicated functionality. The SR's maximum operating frequency of 241 MHz is much greater than the MGT PE's 75MHz, and is comparable to the other SIMPPL Controllers maximum operating frequencies.

4.3 Latency and Throughput

The latency and throughput of the inter-FPGA channel is dependent upon the type of PE in the

SRCE. The PE in the demonstration system has a theoretical maximum data throughput of 1.2 Gbps. In practice a maximum throughput of 1.1 Gbps was observed. The missing throughput is attributed to control characters for framing and flow control being sent over the channel instead of data.

The round-trip latency of the MGT channel was between 63 and 65 cycles. A round-trip latency of 64 cycles is equivalent to a one-way latency of 32 cycles, or 427 ns. The SR does not add any latency to the operation of the SRCE's PE. Data can be read from the RX FIFO into the PE in a single clock cycle. Similarly, the data received from the PE can be written to the TX FIFO in a single cycle.

Future MGTs may require a higher clock rate than the current SR's maximum frequency of 241 MHz. In this case, the SR can be pipelined to increase its maximum operating frequency. Adding a single pipeline stage to the SR should be sufficient to increase the SR's operating frequency as required. This will add one cycle of latency to the operation of the SRCE. However, the latency increase will be small compared to the latency of the channel.

5. Conclusion

This paper presents an extension to the original SIMPPL SoC model to support multi-FPGA platforms. The abstraction of inter-FPGA communication is achieved using the SIMPPL Repeater and a new CE, the SRCE, which is dedicated to forwarding SIMPPL packets through an inter-FPGA communication channel. Implementing these forwarding functions is not possible using the SIMPPL Controllers found in traditional CEs without significant modification to the ISA. In addition, the use of programmable controllers to implement a dedicated function wastes FPGA resources and increases communication latency. The SRCE provides the desired forwarding functionality while abstracting the platform as one large reconfigurable fabric from the modeling perspective.

The SR does not affect the throughput or latency of an inter-FPGA channel. There is an overhead of 13% in FFs and 40% LUTs associated with using the SR with the MGT PE presented in this paper. However, this overhead in resources does not impact performance and reduces design time as multi-FPGA system integration is simplified.

Currently, the SRCE and SIMPPL are being incorporated into a floating application to be implemented on a BEE2 platform [4]. Also, an investigation into extending the SIMPPL SRCE to support point-to-multi-point communication capabilities is also underway. This would allow multiple CEs to share the bandwidth of one inter-FPGA communication channel.

6. Acknowledgments

The authors would like to acknowledge the funding and equipment provided by NSERC, CMC Microsystems, and Xilinx.

7. References

[1] R. Scrofano and V. K. Prasanna, "Preliminary investigation of advanced electrostatics in molecular dynamics on reconfigurable computers," in *Proc. 2006 ACM/IEEE Conf. Supercomputing*, pp. 45-45.

[2] C. He, Mi Lu, C. Sun, "Accelerating seismic migration using FPGA-based coprocessor platform," in *12th Annu. IEEE Symp. Field-Programmable Custom Computing Machines (FCCM),* 2004, pp. 207-216.

[3] J. Varghese, M. Butts, J. Batcheller, "An efficient logic emulation system," in *IEEE Trans. Very Large Scale Integr. (VLSI) Syst,* vol. 1, no. 2, pp. 171-174, Jun. 1993.

[4] C. Chang, J. Wawrzynek, R. W. Brodersen, "BEE2: A high-end reconfigurable computing system," in *IEEE Design and Test of Computers*, vol. 22, no. 2, pp. 114-125, Mar. 2005.

[5] Microsoft (2008, Mar.). BEE3: Revitalizing computer architecture research. [Online]. Mar. 2008, Available: http://research.microsoft.com/projects/BEE3/

[6] L. Shannon, Paul Chow, "SIMPPL: An adaptable SoC framework using a programmable controller IP interface to facilitate design reuse," in *IEEE Trans. Very Large Scale Integr. (VLSI) Syst*, vol. 15, no 4, pp. 377-390, 2007.

[7] L. Shannon, B. Fort, S. Parikh, A. Patel, M. Saldaña, and P. Chow, "A System Design Methodology for Reducing System Integration Time and Facilitating Modular Design Verification*,"* in *Int. Conf. Field Programmable Logic and Applications*, 2006, pp. 1-6.

[8] O. Bringmann, C. Menn, W. Rosenstiel, "Target Architecture Oriented High-Level Synthesis for Multi-FPGA Based Emulation," in *Proc. Conf. Design, Automation, and Test in Europe,* 2000, pp. 326-332.

[9] S. Hauck and G. Borriello, "Logic Partition Orderings for Multi-FPGA Systems," in *Proc. 1995 ACM Third Int. Symp. Field-Programmable Gate Arrays (FPGA),* pp. 32-38.

[10] M. A. S. Khalid, J. Rose, "A Hybrid Complete-Graph Partial-Crossbar Routing Architecture for Multi-FPGA Systems," in *Proc. 1998 ACM/SIGDA Sixth Int. Symp. Field Programmable Gate Arrays,* pp. 45-54.

[11] S. Hauck, G. Borriello, "Pin Assignment for Multi-FPGA Systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol 16, pp. 956-964, Sept 1997.

[12] Xilinx (2008, Mar.). Xilinx XUP Virtex-II Pro Development System. [Online]. Available: http://www.xilinx.com/ univ/xupv2p.html

[13] Xilinx (2008, Mar.). Virtex-II Platform FPGAs: Complete Data Sheet. [Online]. Available: http://www.xilinx.com/support / documentation/data_sheets/ds031.pdf.

[14] Xilinx (2008, Mar.). Fast Simplex Link (FSL) Bus (v2.00a). [Online]. Available: http://www.xilinx.com/bvdocs/ ipcenter/data_sheet/FSL_V20.pdf.

[15] Xilinx (2008, Mar.). RocketIO Transceiver User Guide. [Online]. Available: http://www.xilinx.com/ support/documentation/user_guides/ug024.pdf.

[16] Xilinx (2008, Mar.). Aurora v2.8. [Online]. Available: http://www.xilinx.com/aurora/aurorads128.pdf.