EFFECT OF SERIALIZED ROUTING RESOURCES ON THE IMPLEMENTATION AREA OF DATAPATH CIRCUITS ON FPGAS

SEBASTIAN IP AND ANDY GEAN YE
Department of Electrical and Computer Engineering
Ryerson University
350 Victoria Street, Toronto, Ontario, M5B 2K3
CANADA
email: sebastian.ip@gmail.com, aye@ee.ryerson.ca

Abstract: - In this work, we investigate the effect of serialization on the implementation area of datapath circuits on FPGAs. With ever-increasing logic capacity, FPGAs are being increasingly used to implement large datapath circuits. Since datapath circuits are designed to process multiple-bit wide data, FPGA routing resources, which typically consist of a significant amount of FPGA area, are routinely being used to transport multiple-bit wide signals. Consequently, it is important to design efficient routing architectures for transporting multiple-bit wide signals on FPGAs. Serialization, where several bits of a signal are first time-multiplexed and then transported over a single wire, has been effectively used to increase the I/O bandwidth of FPGAs. Recent work has proposed to use serialization to increase the area efficiency of FPGA routing resources for transporting multiple-bit wide signals. Most of the work, however, has focused on circuit-level design issues. Little work has been done on the overall effect of serialization on the area efficiency of FPGAs. In this work, we investigate the overall effect of serialization on the area efficiency of FPGAs. We propose a detailed FPGA routing architecture, which contains a set of serialization routing resources, and its associated routing tool. Using the architecture and the tool, we measure the effect of serialization on active area and track count. We found that, for benchmarks that contain four-bit wide datapath circuits, serialization can achieve a maximum active area reduction of 6.4% and a routing track reduction of 29%.

Key-Words: - Field-Programmable Gate Arrays, Serial Routing Resources, Routing, Area Efficiency

1 Introduction

Field-Programmable Gate Arrays (FPGAs) are reconfigurable devices that are designed to implement a wide variety of digital systems including video processing, biomedical imaging, digital communications and high performance computing applications. Most of these applications contain a large proportion of datapath circuits that are designed to process multiple-bit wide data.

Datapath circuits usually are constructed out of highly regular bit-sliced structures. During the technology mapping and placement phase of the design flow, these regular structures are often preserved in order to take advantage of the specialized FPGA structures such as carry chains, multi-bit memory blocks, and arithmetic units [1]-[14]. The preserved regularity puts an increasing demand on FPGA routing resources to efficiently transport multiple-bit wide signals on FPGAs.

Serialization, where several bits of signals are time-multiplexed and transmitted over a single wire, potentially can be used to reduce the number of routing tracks that are required to route a datapath circuit. Since routing resources typically consist of over 50% of the total FPGA area [13], the reduction can be used to increase the area efficiency of FPGAs.

The area efficiency of FPGAs is particularly important since a significant density gap exists between FPGAs and Application Specific Integrated Circuits (ASICs). This gap manifests itself in terms of active area (the amount of area that the transistors occupy) and wiring density. In particular, FPGAs dedicate a significant amount of active area to memory for storing the configuration information and to programmable switches for configuring the logic and routing resources. At the same time, FPGAs also must provide sufficient routing tracks for the largest applications that they implement while the smaller applications often leave many tracks unused.

As the process technology shrinks, wiring area and wiring density have become an increasingly important aspect of integrated circuit design [15]. In this work, we measure the effect of serialization on both the active area and wiring density of FPGAs. In particular, we use the traditional standard of post-place-and-route minimum-width-transistor area [16]

to measure the impact of serialization on routing area. To this end, we propose a detailed serial routing architecture and a modified version of the Negotiated Congestion (NC) router [16]-[18] to efficiently serialize and route signals on FPGAs.

The remainder of this paper is organized as follows: Section 2 provides a detailed discussion on the motivation and related work; Section 3 describes the conventional FPGA routing architecture that this work is based on; Section 4 describes the design of the serialized routing architecture; Section 5 describes the modifications to the Computer-Aided Design (CAD) flow; Section 6 presents experimental results; and Section 7 concludes.

2 Motivation and Related Work

Serialization has been effectively used to increase the I/O bandwidth of FPGAs. In particular, Xilinx and Altera both offer FPGAs with gigabit-per-second serial transceivers [19] [20] and the Virtual Wires project has demonstrated the effectiveness of serialization in overcoming I/O pin limitations of FPGAs for large multi-FPGA prototyping systems [21] [22].

As the process dimension shrinks, the performance penalty of on-chip serial communication has been significantly reduced [23]-[25]. Several recent work [26]-[29] has proposed to incorporate serialization circuits into FPGA routing resources based on techniques such as wave-pipelining [30]-[32] and surfing [33]-[35]. The work, however, has focused on circuit-level designs and no investigations have been performed at the architectural level. Time-multiplexed routing architectures based on cycle-by-cycle reconfiguration have also been proposed [36]-[38]. Instead of serializing multi-bit signals, these architectures share routing resources by reconfiguring the routing resources every clock cycle. Only [38] has evaluated the area efficiency of these reconfigurable architectures and the evaluation is estimated based on global scheduling results instead of the actual post-place-and-route results.

Architectural-level investigations based on post-place-and-route results are important since, due to the reconfigurability of FPGAs, the area efficiency of FPGAs is strongly dependent on the utilization of its routing resources and, in particular, the effectiveness of FPGA CAD tools to utilize these resources. In this work, we propose a detailed serial routing architecture and its associated serialization-aware router. Based on the architecture and its supporting CAD tools, we evaluate the effect of serialization on the area efficiency of FPGAs.

3 FPGA architectural Description

We base our study on the island-style FPGA [39] shown in Figure 1. In the figure, a set of Configurable Logic Blocks (CLBs) are arranged in a grid. At the edges of the grid are IO blocks. Horizontal and vertical routing channels run between the rows and columns of the CLBs. Each channel contains a set of routing tracks and each track consists of a set of wire segments that span two CLBs. Switch blocks are used to connect the routing segments together at the intersections of the routing channels. The input and output pins of the CLBs are connected to the track segments through the input and output connection blocks, respectively.

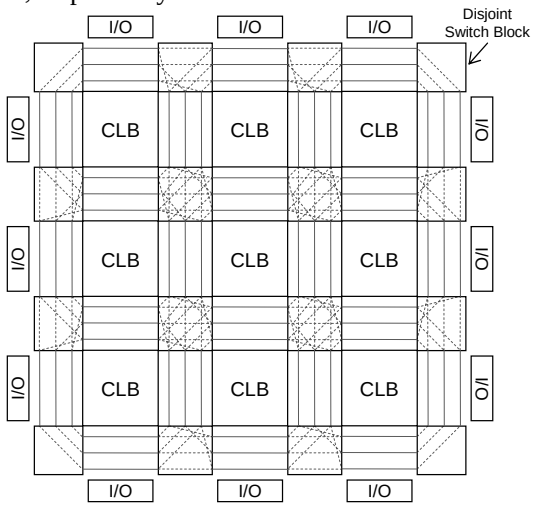

Fig. 1 FPGA Structure

As shown in Figure 2, each CLB contains four logic clusters. Each cluster contains a set of four tightly connected Basic Logic Elements (BLEs) [16]. Each BLE contains a four-input Look-Up Table (LUT) and can be used to implement either combinational or sequential logic. As shown in the Figure 2(a), each cluster contains ten logically equivalent input pins and four logically equivalent output pins.

As in [13] and [14], the four clusters are structured for implementing four neighboring bit-slices from a datapath circuit. Consequently, for each CLB, the

cluster-level inputs are grouped into ten four-bit wide input buses as shown in Figure 2(b). Similarly, the cluster-level outputs are grouped into four four-bit wide output buses.

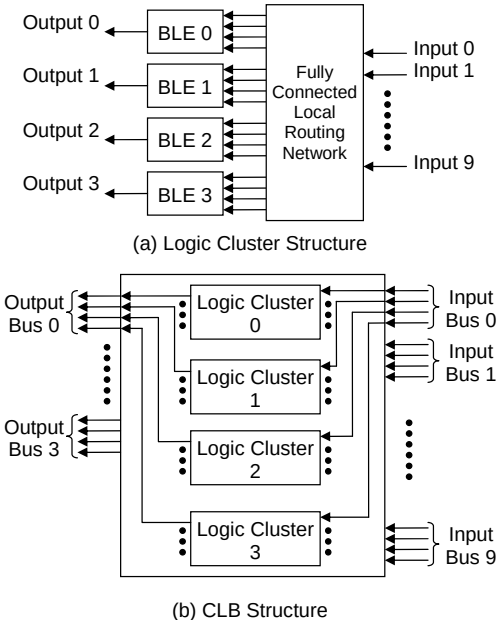

Fig. 2 CLB Structure

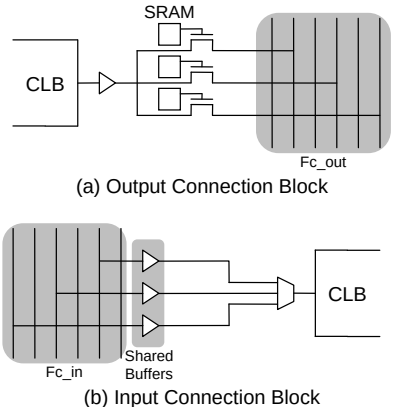

Fig. 3 Output and Input Connection Block Structure

The connection blocks are used to connect the input and output pins of a CLB to the routing tracks. In this work, the same types of connection blocks from [16] are used. In particular, as shown in Figure 3(a), an output connection block connects an output pin to its neighboring routing tracks. It consists of a shared output buffer along with a set of SRAM controlled pass transistor switches. An input connection block, on the other hand, connects an input pin to its neighboring routing tracks. It consists of a multiplexer and a set of shared isolation buffers as shown in Figure 3(b).

As in [16], the connection blocks are specified by two architectural parameters Fc_in and Fc_out. In particular, Fc_in specifies the percentage of routing tracks that a CLB input pin connects to and Fc_out specifies the percentage of routing tracks that a CLB output pin connects to. Finally, the disjoint topology [40] is used for each switch block.

Note that four-input LUTs, four BLEs per cluster, ten input pins per cluster, four output pins per cluster, and four clusters per CLB are shown to be efficient for implementing datapath-oriented FPGAs in [13]. The segment length of two has been shown to be efficient for both datapath-oriented [13] and conventional FPGAs [16]. All transistors in CLBs, connection blocks and switch blocks are sized based on the transistor sizing methodology from [13] and [16].

4 Serial Routing Architecture

We augment the conventional routing architecture with serializers and deserializers. A serializer converts a four-bit wide signal into a one-bit wide, time-multiplexed, serial signal. The serial signal can be more efficiently routed through the FPGA routing resources since it requires considerably less routing tracks than the original four-bit wide signal. Deserializers are used to convert the serialized signals back into their original four-bit wide parallel form.

4.1 Serializer and Deserializer Connectivity

Figure 4 shows the connectivity of the serializers and deserializers. As shown in Figure 4(a), each serializer can be connected to a number of CLB output buses. Dedicated connections similar to the input connection blocks are used to connect the CLB outputs to the serializers. In particular, each serializer contains four input pins. Each pin corresponds to one of the four bits that are to be serialized by the serializer. The pin is connected to a corresponding bit of a CLB output bus through a multiplexer. The output of the serializer is then connected to the routing tracks by a dedicated output connection block. Note that, as shown in Figure 4(a), the CLB outputs also retain their direct connections to the routing tracks through their own output connection blocks. Consequently, the signals

can bypass the serializers and be directly connected to the tracks. (For clarity the pass transistor switches shown in Figure 3(a) are not shown in Figure 4(a)).

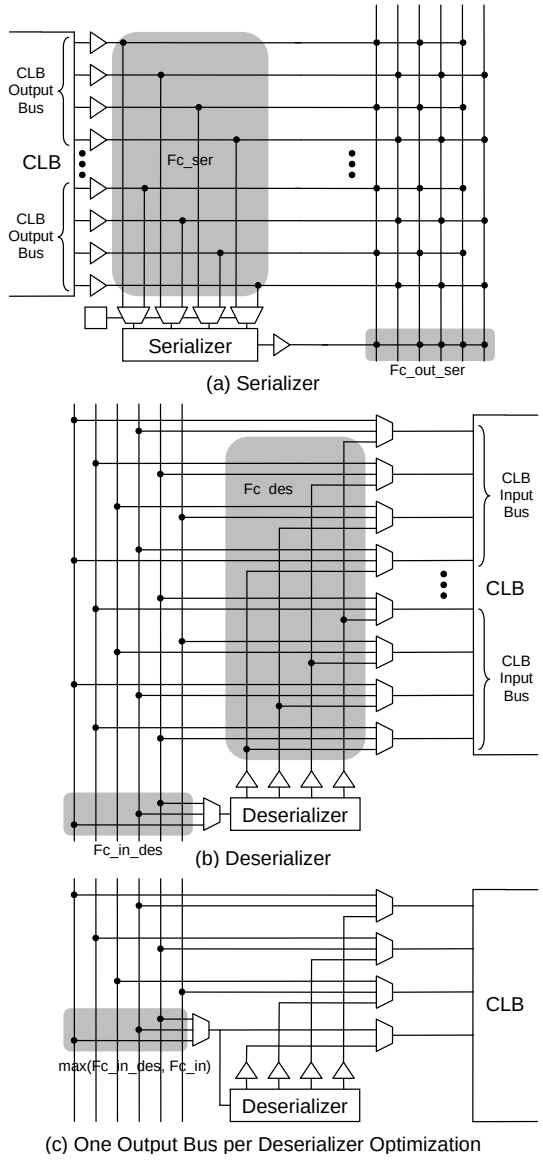

Fig. 4 Serializer and Deserializer Connectivity

As shown in Figure 4(b), a deserializer contains a single input that receives the serialized signals. The input is connected to the routing tracks by a single input connection block. The output of a deserializer consists of four bits – each representing one bit of the original four-bit wide parallel signal. As shown in the figure, these bits are connected to the input connection blocks of the corresponding CLB input buses. As for the serializer, each output bit of the deserializer is connected to a corresponding bit from a CLB input bus. Note that, as shown in the figure, the CLB inputs also retain direct connections to the routing tracks through the input connection blocks. (For clarity the isolation buffers shown in Figure 3(b) are not shown in Figure 4(b)).

Note that when a deserializer is connected to only one CLB input bus, the deserializer, as shown in Figure 4(c), can share its input connection block with the input connection block of a CLB input pin from the bus. The sharing requires an additional 2:1 multiplexer to connect the output of the shared input connection block to the corresponding CLB input pin. The sharing, however, also reduces the number of input connection blocks from five to four. Since the overall implementation area of the deserializer is reduced, the sharing is always assumed in this work.

As shown in Figure 4, the connectivity of the serializers and the deserializers can be quantified by six architectural parameters. In particular, Fc_ser is equal to the percentage of four-bit wide CLB output buses per CLB that can be connected to a serializer. Fc_des is equal to the percentage of four-bit wide CLB input buses per CLB that can be connected to each deserializer. Fc_out_ser is equal to the percentage of tracks that the output pin of a serializer can connect to and Fc_in_des is equal to the percentage of tracks that the input pin of a deserializer can connect to. Finally, Num_ser and Num_des are equal to the number of serializers and deserializers per CLB, respectively.

4.2 Pipelined Routing Tracks and Synchronization

In this work, as in [41]-[44], the routing tracks are pipelined to increase the throughput of the serial transmission. In particular, in each switch block, the buffers are replaced by latches. All the latches in the serializers, the deserializers and the switch blocks are clocked by a single clock, the global serial clock.

Due to the pipelined routing tracks, the number of clock cycles that it takes for a signal to travel from its serializer to its deserializer varies depending on the number of switch blocks that the signal must go through. Consequently, serialized signals can reach their deserializers at different times. For example, if one serialized signal must travel through three switch blocks to reach its deserializer, the deserializer should start the deserialization process three cycles after the

first bit of the signal is transmitted by the serializer. Another signal, however, might travel through ten switch blocks to reach its deserializer. This deserializer must start its deserialization process ten cycles after the first bit of the signal is transmitted by the serializer. Consequently, each deserializer must be individually synchronized with respect to its serializer.

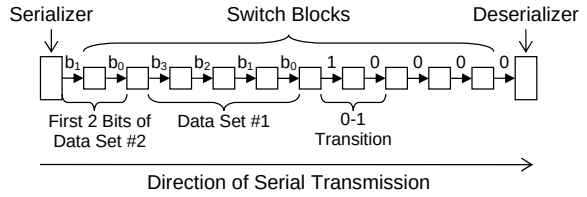

Fig. 5 Preamble of 0-1 Transition in Serial Transmission

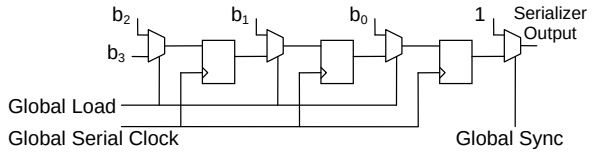

Fig. 6 Serializer

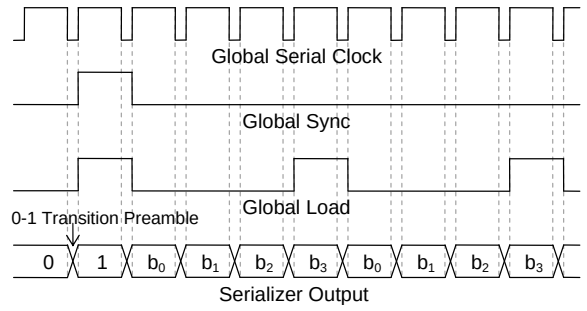

Fig. 7 Serializer Operation

To synchronize the operation of the serializers and the deserializers, all latches in the switch blocks are first initialized to 0. As shown in Figure 5, each serializer then transmits a preamble of 0-to-1 transition before transmitting its first serialized signal. When the preamble reaches the deserializer, it is used by the deserializer to trigger the deserialization process.

4.3 Serializer Design

The detailed design of the serializer is shown in Figure 6. It consists of three latches and four 2:1 multiplexers. The serializer is controlled by three global clock signals – the global synch, the global load and the global serial clock. The operation of the serializer is shown in Figure 7. First, the global synch signal is asserted to create a preamble of 0-to-1 transition at the serializer output. The global load signal is asserted at the same time to load the first set of CLB output values into the serializer. These values are then shifted out of the serializer output one bit at a time in four clock cycles. After four cycles, the global load signal is asserted again to load the next set of CLB output values into the serializer.

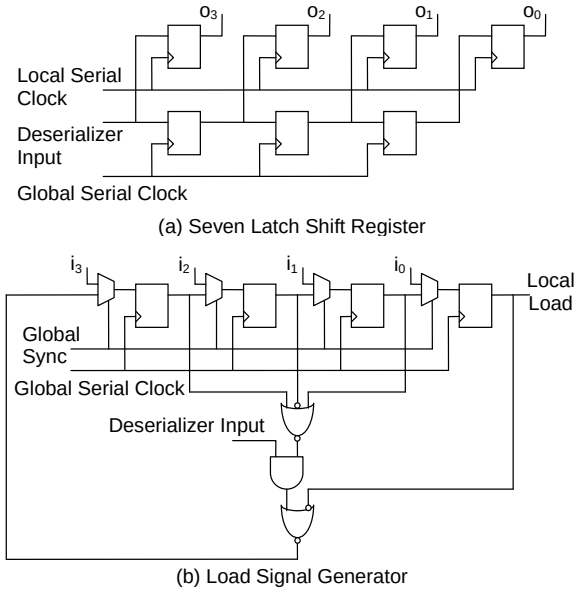

Fig. 8 Deserializer

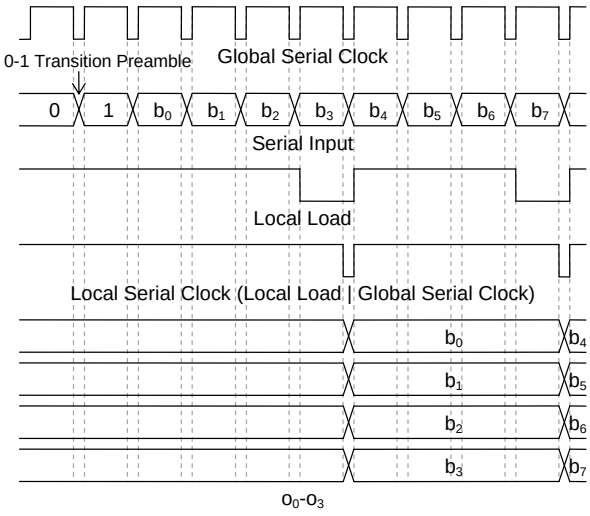

Fig. 9 Deserializer Operation

4.4 Deserializer Design

The design of the deserializer is shown in Figure 8. It consists of a seven-latch shift register as shown in Figure 8(a) and a load signal generator as shown in Figure 8(b). The operations of the shift register and the load signal generator are shown in Figure 9. As shown the latches are arranged in two banks. The three latches in the lower bank are controlled by the global serial clock. The input of the first latch is directly connected to the input of the deserializer.

The inputs of the four latches in the upper bank are connected to the corresponding outputs from the lower bank and the deserializer input as shown in Figure 8(a). The latches in the upper bank are controlled by the local serial clock (which is equal to the output of the load signal generator, local load, ORed with the global serial clock). The outputs of the upper bank, labeled o_0-o_3, are directly connected to a CLB input bus. Note that the lower bank of latches are used to de-serialize the input signal from the deserializer input and the top bank of latches are used to isolated any signal changes during the deserialization process from the CLB input bus.

The load signal generator is designed to synchronize the operation of the serializers and the deserializers. In particular, the same global synch signal that creates the preamble of 0-to-1 transition at the serializer also initializes the load signal generator (as shown in Figure 8(b)) of the deserializer by loading a set of initialization values (i_0 to i_3) into the latches. A combinational circuit is then used to detect the 0-to-1 transition at the input of the deserializer. Consequently, as shown in Figure 9, the output of the load signal generator (local load) remains 1 until a 0-to-1 transition is detected by the combinational circuit. The load signal generator then pulses the local load signal every four clock cycles in order to load a new set of deserialized data into the upper bank of the seven-latch shift register.

4.5 Transistor Sizing

Note that the latches and the multiplexers are sized according to [13] and [16]. The combinational circuit is constructed out of CMOS gates with minimum drive strength and equalized rise and fall times. Each serializer consumes 32.9 minimum-width-transistor area and each deserializer consumes 126.5 minimum-width-transistor area. In comparison, a CLB consumes 6712 minimum-width-transistor area [13]. Since a maximum of ten deserializers and four serializers can be attached to a CLB, in combination, the serializers and deserializers consume a maximum of 1395 minimum-width-transistor area. For the switch blocks, each latch requires 3.87 additional minimum-width-transistor area than the buffer that it replaces. Finally, the clock network design from [45] is used for the three clock networks required to drive the additional global clock signals (global synch, global load, and global clock). Overall, these three clock networks consume an additional 169.0 minimum-width-transistor area per CLB tile.

5 CAD Flow

To efficiently utilize the serial routing resources, we modify the Versatile Place and Route (VPR) CAD flow [16] to model the serial routing architecture. In particular, circuits are mapped onto the CLB structure as shown in Figure 2 using the Synopsys synthesis and the datapath-oriented T-VPACK packing tools as in [13] [14]. This mapping process preserves as much datapath regularity as possible in order to maximize the number of inter-CLB signals that can be grouped into four-bit wide groups and be serialized by the serializers. The VPR placer is then used to place the packed CLBs onto the serial FPGA.

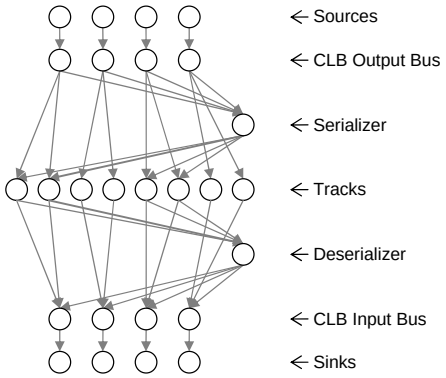

Fig. 10 Serializer and Deserializer Representation in Routing Resource Graph

Routing is performed using a modified version of the VPR router. Based on the Negotiated Congestion algorithm [18], the router selectively serializes multi-bit signals based on the congestion of the serializers and the deserializers. As shown in Figure 10, additional nodes are added to the routing resource graph to model the connectivity of the serializers and the deserializers. In particular, in the figure, a

serializer node is connected to a CLB output bus. Signals from the bus can be serialized into a bit of time-multiplexed signal by the serializer and be routed through one of the four routing tracks connected to the serializer. Similarly, the deserializer node is connected to four tracks and its outputs are connected to a CLB input bus.

To route a four-bit wide signal, we first route bit 0 of the signal (the test bit) through the routing resources. From its CLB output pin, the test bit has the option of either expanding through a serializer or be directly connected to a routing track. If the bit expands through a serializer, it represents the serialized version of the entire multi-bit signal and must enter a deserializer before reaching its target CLB. If the bit bypasses the serializers, it remains un-serialized and must bypass all deserializers. Both the serialized and un-serialized versions of the signal are expanded simultaneously. During the expansion, the version with the least accumulated congestion cost reaches the destination first and is selected as the route for the entire signal. If the serialized version is selected, the remaining bits in the four-bit wide signal are marked as routed. If the un-serialized version is selected, the remaining bits in the four-bit wide signal are routed individually as single-bit signals.

Note that the serialized version and the un-serialized version of a signal can expand onto the same routing track. Conventional routing algorithms only use a single wave-front. When two parts of the same wave-front are expanded onto the same routing resource, the part with the higher congestion cost is eliminated [46]. The serialized and un-serialized versions of the test bit, however, represent two electrically distinct signals. Replacing one with the other eliminates future expansion opportunities. For example, Figure 11(a) shows a wave-front consisting of three nodes – two gray nodes representing the un-serialized version of the expansion and one black node representing the serialized version. As shown, the un-serialized version has expanded onto the two routing tracks on the left while the serialized version is at the serializer.

During the next expansion, the serialized version expands onto the same tracks occupied by the un-serialized version. Assuming the serialized expansions have lower costs, they replace the un-serialized expansions and become part of the new wave-front as shown in Figure 11(b). Eliminating the un-serialized version, however, also eliminates future expansion opportunities. For example, in the figure, the deserializer node is already occupied. Consequently, none of nodes on the wave-front shown in Figure 11(b) can expand into the deserializer without causing congestion. As a result, the router cannot find an uncongested routing solution.

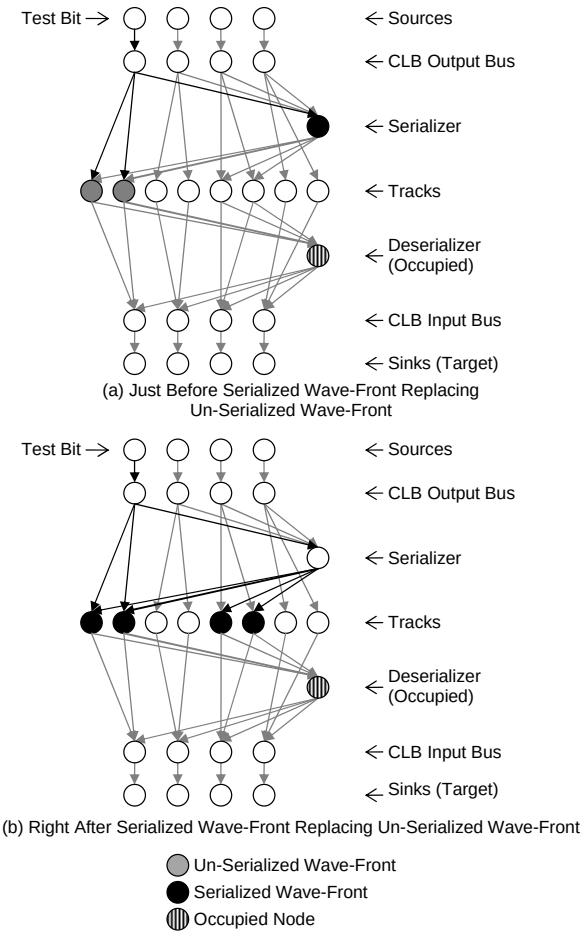

Fig. 11 Single Wave-Front Solution

In this work, we keep two distinct wave-fronts – one for the serialized version and another for the un-serialized version. The two wave-fronts do not interfere with each other and the wave-front that reaches the destination first determines the route of an entire signal. An example of the dual wave-front expansion is shown in Figure 12. In this example, when the serialized wave-front expands onto the two nodes that are already occupied by the un-serialized wave-front, both wave-fronts are kept. Consequently, the congested deserializer can be bypassed by the un-

serialized wave-front, which directly expands onto the target CLB input pin.

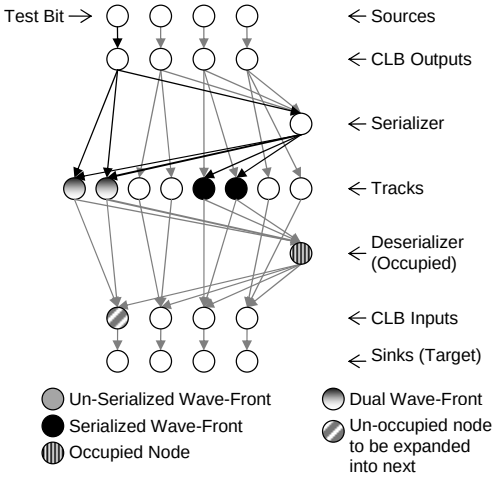

Fig. 12 Dual Wave-Front Solution

6 Experimental Results

To evaluate the effect of serialization on FPGA area and track utilization, a set of 15 benchmark circuits from the Pico-Java processor [47] were mapped onto the serial routing architecture using the CAD flow from Section 5. These benchmarks were used in several previous studies on datapath-oriented FPGAs [13] [14]. The area and track count are then measured and compared to the conventional routing architecture.

6.1 Architectural Parameters

The conventional architectural parameters are set to the values as shown in Table 1. As shown, each CLB input is connected to 50% of the routing tracks in its neighboring routing channel (Fc_in = 50%) and each CLB output is connected to 25% of the routing tracks (Fc_out = 25%) in its neighboring routing channel. These values were shown to be efficient in [13] and [16]. Similarly, as in Section 3, the track segment length is set to two and the disjoint switch block topology is used. The LUT size is set to four. Each logic cluster contains four BLEs, and each CLB contains four logic clusters. Finally, as shown in Figure 2, there are four four-bit wide output buses and ten four-bit wide input buses per CLB.

Table 2 shows the additional architectural parameter settings for the serial routing architecture. In particular, each serializer/deserializer is mapped to a dedicated CLB output/input bus by setting Fc_ser to 0.25 and Fc_des to 0.1. Note that, despite this one-to-one mapping of serializers/deserializers to output/input buses, serialized signals still can exit/enter a CLB through any of the serializers/deserializers due to the logic equivalency of logic cluster output/input pins. In particular, a four-bit wide output from a CLB can be routed to any of the CLB output buses by the fully connected local routing network shown in Figure 2(a) [16]. The output can then be serialized by the serializer connected to the bus. Similarly, a serialized signal can enter a CLB through any of the deserializers. Once deserialized, each bit of the signal can then be routed from the corresponding logic cluster input to any of the LUT inputs through the fully connected local routing network.

Table 1. Conventional Routing Architecture Parameters

Architectural Parameter	Value
Fc_out	0.25
Fc_in	0.5
Segment Length	2
Switch Block Topology	Disjoint
LUT Size	4
Number of BLEs per Cluster	4
Number of Clusters per CLB	4
Number of Output Buses	4
Number of Input Buses	10

Table 2. Serial Routing Architecture Parameters

Architectural Parameter	Value
Fc_ser	0.25
Fc_des	0.1
Fc_out_ser	0.25
Fc_in_des	0.5
Num_ser	4
Num_des	From 10 to 4

As shown in Table 2, Fc_out_ser is set to 0.25 and Fc_in_des is set to 0.5. These values are selected to be the same as the Fc_out and Fc_in values of the conventional routing architecture, respectively. By equating Fc_out_ser/Fc_in_des to Fc_out/Fc_in, a serialized signal has the same level of connectivity to/from routing tracks as an un-serialized signal. Consequently, serialization decisions are based entirely on the effect of serialization and are not influenced by connectivity differences to/from the routing tracks.

As shown in Table 3 when mapped onto architectures with four serializers and ten deserializers, the benchmark circuits on average utilizes 48.9% of the serializers and 30.0% of the deserializers. Since only a maximum of four

serializers are required per CLB, each serializer represents a significant proportion (25%) of the total number of serializers. Each deserializer, on the other hand, represents a much smaller proportion (10%) of the total number of deserializers. In addition, as shown in Section 4, each deserializer contains a load signal generator for synchronizing its operations with the serializers. A deserializer thus consumes significantly more area than a serializer (126.5 vs. 32.9 minimum-width-transistor area). As a result, reducing the number of serializers significantly reduces the percentage of signals that can be serialized while only achieving a minimum reduction in the total area consumed by the serializers/deserializers per CLB. Reducing the number of deserializers, on the other hand, can substantially reduce the total serializer/deserializer area while still maintaining a high rate of serialization [48]. As a result, in this work, as shown in Table 2, the number of deserializers is varied from ten to four and the number of serializers is fixed at four.

Table 3. Percentage of Serialization, Serializer Utilization and Deserializer Utilization for the 4 Serializer 10 Deserializer Configuration

Benchmark	# of Two-Terminal Multi-Bit Connections	% of Serialized Multi-Bit Connections	% of Used Serializers	% of Used Deserializers
icu_dpath	3972	99.2%	57.1%	42.6%
ex_dpath	3456	99.9%	59.2%	44.1%
ucode_dat	1608	100.0%	54.5%	41.7%
imdr_dpath	1304	100.0%	61.1%	38.3%
dcu_dpath	1220	100.0%	58.3%	40.5%
mantissa_dp	1156	99.3%	54.3%	36.1%
incmod	808	100.0%	44.9%	31.5%
multmod_dp	684	100.0%	26.7%	15.4%
smu_dpath	472	99.2%	40.3%	23.9%
pipe_dpath	436	100.0%	56.7%	21.3%
exponent_dp	428	100.0%	43.1%	24.2%
rsadd_dp	384	100.0%	52.2%	38.3%
prils_dp	280	95.7%	32.4%	21.9%
code_seq_dp	252	100.0%	42.7%	9.6%
ucode_reg	128	100.0%	50.0%	20.0%
Overall	16588	99.6%	48.9%	30.0%

6.2 Serialization Penalty

To encourage serialization, the NC router, as described in Section 5, penalizes un-serialized multi-bit signals. In particular, if a test bit is expanded from a CLB output pin directly onto a track without being serialized, the expansion cost of the track is first multiplied by a penalty factor and then accumulated into the total expansion cost of the bit. In this work, we vary the penalty factor from 1000 to $2x10^8$. The smaller values were shown to be ineffective, as many multi-bit signals were left un-serialized [48]. The higher values from $2x10^7$ to $2x10^8$, on the other hand, produce good serialization results since they force test bits to go through serializers unless the associated routing resources are highly congested.

As shown in Table 4, in this work, a set of penalty factors are selected for each benchmark circuit based on the criterion of minimizing track count. Table 3 shows that, with these values, nearly all multi-bit signals are serialized for each benchmark circuit for the architecture containing four serializers and ten deserializers per CLB.

Table 4. Penalty Factor Results

Benchmark	Penalty Factor ($x10^6$)						
	10 Des	9 Des	8 Des	7 Des	6 Des	5 Des	4 Des
code_seq_dp	80	80	20	100	100	20	80
dcu_dpath	60	60	100	120	60	100	80
ex_dpath	60	20	20	100	60	60	60
exponent_dp	100	60	120	200	120	40	60
imdr_dpath	20	60	200	120	60	20	200
incmod	60	200	100	80	20	60	100
mantissa_dp	60	80	80	80	100	120	80
icu_dpath	60	100	60	80	20	60	40
multmod_dp	40	80	80	200	120	40	60
pipe_dpath	60	40	80	60	60	20	20
prils_dp	200	20	80	80	60	60	40
rsadd_dp	120	120	80	100	80	40	80
ucode_dat	60	40	80	80	100	20	80
ucode_reg	80	60	80	60	60	60	80
smu_dpath	60	60	80	200	60	60	80

6.3 Serialization Results – Track Count

As in [49]-[51], a binary search is performed to determine the minimum channel width required to successfully route each circuit. Figure 13 shows the average number of tracks reduced per channel over the conventional routing architecture. As shown, the architecture with four serializers and ten deserializers per CLB requires the least number of tracks. As the number of deserializers per CLB decreases from ten to seven, the track reduction per channel decreases slightly from 14.9 tracks to 13.9 tracks. From six to four deserializers per CLB, tracks reduced per channel decreases significantly, where the architecture containing four serializers and four deserializers per CLB uses only 8.07 less tracks per channel than the conventional routing architecture.

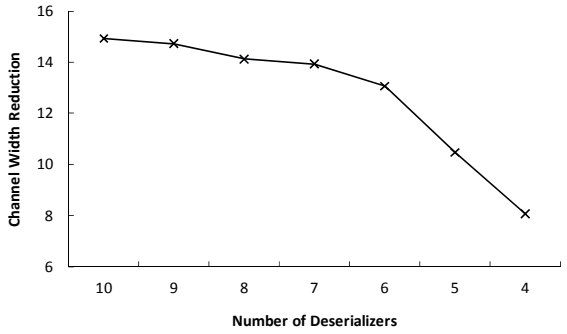

Fig. 13 Channel Width Reduction

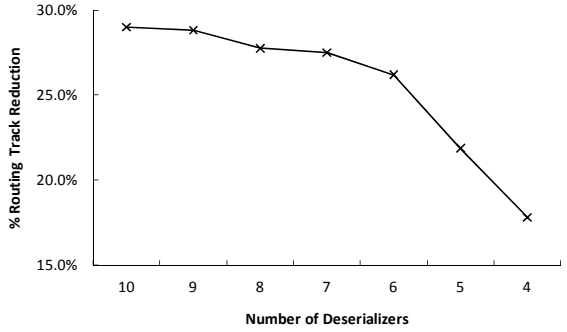

Fig. 14 Routing Track Reduction

Table 5. Per Circuit Routing Track Reduction

Benchmark	% of Routing Track Reduction						
	10 Des	9 Des	8 Des	7 Des	6 Des	5 Des	4 Des
code_seq_dp	23.7%	23.7%	23.7%	23.7%	23.7%	23.7%	23.7%
dcu_dpath	45.3%	45.3%	45.3%	45.3%	45.3%	37.7%	30.2%
ex_dpath	43.2%	40.7%	39.5%	38.3%	33.3%	24.7%	19.8%
exponent_dp	21.1%	21.1%	19.3%	21.1%	21.1%	12.3%	7.02%
imdr_dpath	24.2%	27.4%	25.8%	24.2%	21.0%	14.5%	11.3%
incmod	24.6%	24.6%	24.6%	21.1%	22.8%	19.3%	14.0%
mantissa_dp	26.1%	23.2%	23.2%	23.2%	23.2%	17.4%	11.6%
icu_dpath	28.0%	28.0%	24.0%	24.0%	16.0%	13.3%	2.67%
multmod_dp	6.56%	3.28%	6.56%	6.56%	6.56%	0.00%	0.00%
pipe_dpath	32.3%	32.3%	32.3%	32.3%	32.3%	32.3%	29.0%
prils_dp	16.2%	13.5%	10.8%	10.8%	10.8%	10.8%	10.8%
rsadd_dp	27.0%	32.4%	29.7%	29.7%	27.0%	18.9%	10.8%
ucode_dat	41.4%	41.4%	36.2%	34.5%	34.5%	27.6%	20.7%
ucode_reg	56.0%	56.0%	56.0%	56.0%	56.0%	56.0%	56.0%
smu_dpath	19.5%	19.5%	19.5%	22.0%	19.5%	19.5%	19.5%
Overall	29.0%	28.8%	27.8%	27.5%	26.2%	21.9%	17.8%

Figure 14 shows the percentage of total tracks reduced over all routing channels for all benchmark circuits. As shown, the architecture with four serializers and ten deserializers per CLB requires 29.0% less routing tracks than the conventional routing architecture. As the number of deserializers is reduced, the track reduction drops significantly from 29.0% to 17.8% with the knee of the graph occurring at six deserializers per CLB.

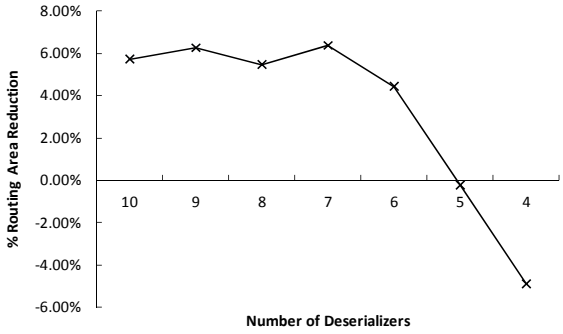

Fig. 15 Routing Area Reduction

Table 5 shows the percentage of tracks reduced per circuit. As shown, the architecture with four serializers and ten deserializers per CLB consistently achieves the highest percentage of routing track reduction since this configuration nearly serializes all multi-bit signals for each benchmark circuit. As the number of deserializers is reduced, it becomes more probably for the deserializers to become congested as multi-bit signals compete for their use. Consequently, with increased congestion, more multi-bit signals must remain un-serialized. Since each un-serialized signal requires four routing tracks instead of one, track reduction decreases across all benchmarks.

6.4 Serialization Results – Active Area

We define routing area as the total active area consumed by the serializers, deserializers, switch blocks and all types of connection blocks. Figure 15 shows the percentage of routing area reduction achieved by the serial routing architecture over the conventional routing architecture. As shown the routing area reduction follows closely with routing track reduction. From ten deserializers to six deserializers, we observed an area reduction of 4.44% to 6.37%. For five and four deserializers, on the other hand, the amount of active area reduction is quickly reduced and becomes negative. These results show that, for our benchmark set, the best number of deserializers per CLB, in terms of active area, is seven.

Table 5 shows that, with seven deserializers, there is a lower than maximum 27.5% routing track reduction. However, as shown in Section 4, the area cost of the deserializer circuit is quite significant and not all deserializers in a ten deserializer configuration are utilized. As a result, by reducing the number of

Table 6. Routing Area Reduction for 4 Serializers and 7 Deserializers

Benchmark	% of Multi-Bit Signals	Serial Routing Area					Conventional Routing Area	% of Area Reduction
		Track Only	Serializer	Deserializer	Clock Network	Total		
ucode_reg	74	3.88E+04	1.19E+03	7.93E+03	1.25E+03	4.92E+04	6.27E+04	21.6%
dcu_dpath	65	5.39E+05	8.44E+03	5.67E+04	8.87E+03	6.13E+05	7.93E+05	22.6%
ex_dpath	61	2.54E+06	2.59E+04	1.74E+05	2.72E+04	2.77E+06	3.50E+06	20.9%
ucode_dat	61	1.05E+06	1.32E+04	8.86E+04	1.39E+04	1.16E+06	1.32E+06	11.7%
icu_dpath	61	3.28E+06	2.97E+04	1.99E+05	3.12E+04	3.54E+06	3.73E+06	5.25%
rsadd_dp	61	1.96E+05	3.30E+03	2.21E+04	3.46E+03	2.25E+05	2.35E+05	4.45%
mantissa_dp	56	8.95E+05	8.44E+03	5.67E+04	8.87E+03	9.69E+05	1.01E+06	4.25%
pipe_dpath	56	2.42E+05	4.75E+03	3.19E+04	4.99E+03	2.84E+05	2.80E+05	-1.42%
imdr_dpath	53	1.01E+06	1.07E+04	7.17E+04	1.12E+04	1.11E+06	1.13E+06	2.49%
smu_dpath	51	3.03E+05	4.75E+03	3.19E+04	4.99E+03	3.71E+05	3.64E+05	-1.87%
incmod	48	7.81E+05	8.44E+03	7.17E+04	8.87E+03	8.55E+05	8.44E+05	-1.42%
exponent_dp	47	4.46E+05	4.75E+03	3.19E+04	4.99E+03	4.88E+05	4.81E+05	-1.30%
code_seq_dp	46	2.16E+05	3.30E+03	2.21E+04	3.46E+03	2.45E+05	2.38E+05	-2.91%
prils_dp	42	3.48E+05	4.75E+03	3.19E+04	4.99E+03	3.90E+05	3.35E+05	-16.4%
multmod_dp	32	1.78E+06	1.60E+04	1.07E+05	1.68E+04	1.92E+06	1.67E+06	-14.7%

deserializers per CLB, the overall active area is reduced.

Table 6 shows a detailed breakdown of the results collected from our experiments for four serializers and seven deserializers. As shown, of the 15 benchmark circuits, eight produced routing area reductions and seven have an increase in routing area. In general, larger benchmarks produced significant routing area reductions due to the large number of multi-bit signals that they contain. The smaller circuits, on the other hand, produced less routing area reduction or exhibit an increase in routing area.

One exception to this trend is the multimod_dp circuit. As shown in column 2 of Table 6, the circuit contains relatively little multi-bit signals even though it ranks third in size (in terms of conventional routing area). As shown in [13], many of the inter-CLB signals contained in multimod_dp shift bit positions as they travel from a CLB output bus to a CLB input bus. These shifts cannot be captured by the current non-shifting serial routing resources and hence are not included in the percentage value shown in column 2. The result suggests that adding shifting capabilities to the serial routing resources potentially can improve the area efficiency of multimod_dp-like circuits.

Another exception to the rule is the smallest benchmark, ucode_reg, which produced a significant area reduction of 21.6%. In this circuit, 74% of the inter-CLB signals are multi-bit signals, consequently, it benefits significantly from serialization.

The table shows that the percentage of multi-bit signals per circuit (defined as the total number of two-terminal connections in multi-bit inter-CLB signals vs. the total number of two-terminal connections in all inter-CLB signals) has a significant impact on area reduction. Our results suggest that in order for a benchmark to benefit from serialization (in terms of active area reduction) the percentage of multi-bit signals in the circuit must be at least 53% or greater.

7 Conclusions

In this work, we investigated the effect of serialization on FPGA routing efficiency. Based on our benchmark circuits, we found that, for four-bit wide serial routing resources, a maximum active area reduction of 6.39% can be achieved through serialization. The configuration that achieves the maximum active area reduction contains four serializers and seven deserializers. For maximum routing track reduction, on the other hand, the serial routing architecture should contain four serializers and ten deserializers. This configuration achieves a maximum routing track reduction of 29%.

This work can be further expanded to investigate the effect of serialization on other routing architectures. In particular, this work has focused on the effect of serialization on bi-directional FPGA routing resources. It can be expanded to investigate the effect of serialization on directional routing resources [52]. Furthermore, in this work, signals are selected for serialization solely based on the congestion of the serialization routing resources. The routing tool did not consider the criticality of signals and the distances that a signal must travel. These parameters, however, can be important in maximizing routing area reduction while minimizing the effect of serialization on delay. The design of such routers is left as future work.

This work also provides a base platform for examining the efficiency of more advanced serialization techniques such as surfing and wave-pipelining [28] [29]. Finally, future studies should also investigate the effect of serialization on circuits containing wider multi-bit signals, such as 8-bit, 16-bit, and 32-bit through the use of wider serializers and deserializers.

References:

[1] D. Chen and J. Rabaey, "A Reconfigurable Multiprocessor IC for Rapid Prototyping of Algorithmic-Specific High-Speed DSP Data Paths," *IEEE Journal of Solid State Circuits*, Vol. 27, No. 12, December 1992, pp. 1895-1904.

[2] A. Yeung and J. Rabaey, "A Reconfigurable Data Driven Multi-Processor Architecture for Rapid Prototyping of High Throughput DSP Algorithms," in *Proceedings of the 1993 IEEE Hawaii International Conference on System Sciences*, Wailea, HI, January 1993, pp. 169-178.

[3] D. Cherepacha and D. Lewis, "DP-FPGA: An FPGA Architecture Optimized for Datapaths," *Journal of VLSI Design*, Vol. 4, No. 4, April 1996, pp. 329–343.

[4] C. Ebeling, "RaPiD-Reconfigurable Pipelined Datapath," in *Proceedings of the 1996 International Workshop on Field-Programmable Logic*, Darmstadt, Germany, September 1996, pp. 126–135.

[5] R. Bittner, P. Athanas, and M. Musgrove, "Cold: An Experiment in Wormhole Run-Time Reconfiguration," in *Proceedings of the 1997 ACM/SIGDA Internaltional Symposium on Field Programmable Gate Arrays*, Montery, CA, February 1997, pp. 79-85.

[6] J. Hauser and J.Wawrzynek, "Garp: A MIPS Processor with a Reconfigurable Coprocessor," in *Proceedings of the 1997 IEEE Symposium on Field-Programmable Custom Computing Machines*, Napa Valley, CA, April 1997, pp. 24–33.

[7] E. Waingold, "Baring It All to Software: Raw Machines," *IEEE Computer*, Vol. 30, No. 9, September 1997, pp. 86–93.

[8] T. Miyamori and K. Olukotun, "A Quantitative Analysis of Reconfigurable Coprocessors for Multimedia Applications," in *Proceedings of the 1998 IEEE Symposium on Field-Programmable Custom Computing Machines*, Napa Valley, CA, April 1998, pp. 2–11.

[9] A. Marshall, T. Stansfield, I. Kostarnov, J. Vuillemin, and B. Hutchings, "A Reconfigurable Arithmetic Array for Multimedia Applications," in *Proceedings of the 1999 ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, Monterey, CA, February 1999, pp. 135-143.

[10] A. Alsolaim, "Architecture and Application of a Dynamically Reconfigurable Hardware Array for Future Mobile Communication Systems," in *Proceedings of 2000 IEEE Symposium on Field-Programmable Custom Computing Machines*, April 2000, Napa Valley, CA, pp. 205–214.

[11] S. Goldstein, "PipeRench:A Reconfigurable Architecture and Compiler," *IEEE Computer*, Vol. 33, No. 4, April 2000, pp. 70–77.

[12] K. Leijten-Nowak and J. van Meerbergen, "An FPGA Architecture with Enhanced Datapath Functionality," in *Proceedings of the 2003 ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, Monterey California, February 2003, pp. 195–204.

[13] A. Ye and J. Rose, "Using Bus-Based Connections to Improve Field-Programmable Gate-Array Density for Implementing Datapath Circuits," *IEEE Transactions on Very Large Scale Integration Systems*, Vol. 14, No. 5, May 2006, pp.462-473.

[14] Ping Chen and Andy Ye, "The Effect of Multi-Bit Correlation on the Design of Field-Programmable Gate Array Routing Resources," *IEEE Transactions on Very Large Scale Integration Systems*, Volume PP, No. 99, October 2009.

[15] www.ITRS.net, 2007.

[16] V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAs*, February 1999, Kluwer Academic Publishers.

[17] R. Nair, "A simple yet effective technique for global wiring," *IEEE Transactions on Computer-Aided Design*, Vol. 6, No. 3, Mar. 1987, pp. 165-172.

[18] C. Ebeling, L. McMurchie, S. Hauck, and S. Burns, "Placement and Routing Tools for the Triptych FPGA," *IEEE Transactions on Very Large Scale Integration Systems*, Vol. 3, No. 4, December 1995, pp. 473-482.

[19] Virtex-6 Family Overview, Xilinx, 2010.

[20] Stratix IV Device Handbook, Altera, 2010.

[21] J. Babb, R. Tessier, and A. Agarwal, "Virtual Wires: Overcoming Pin Limitations in FPGA-based Logic Emulators," in *Proceedings of the 1994 IEEE Workshop on FPGAs for Custom Computing Machines*, Napa Valley, CA, April 1994, pp. 142-151.

[22] R. Tessier, J. Babb, M. Dahl, S. Hanono, and A. Agarwal, "The Virtual Wires Emulation System: A Gate-Efficient ASIC Prototyping Environment," in *Proceedings of the 1994 ACM International Workshop on Field-Programmable Gate Arrays*, Berkeley, CA, Feb. 1994.

[23] S. Kimura, T. Hayakawa, T. Horiyama, M. Nakanishi, and K. Watanabe, "An On-Chip High Speed Serial Communication Method Based on Independent Ring Oscillators," in *Proceedings of the 2003 International Solid State Circuits Conference*, San Francisco, CA, February 2003, pp. 390-391.

[24] I. Wey, L. Chang, Y. Chen, S. Chang, and A. Wu, "A 2Gb/S High-Speed Scalable Shift-Register Based On-Chip Serial Communication Design for SoC Applications," in *Proceedings of the 2005 IEEE International Symposium on Circuits and Systems*, Kobe, Japan, May 2005, pp. 1074-1077.

[25] R. Dobkin, A. Morgenshtein, A. Kolodny, and R. Ginosar, "Parallel vs. Serial On-Chip Communication," in *Proceedings of the 2008 ACM/SIGDA International Workshop on System Level Interconnect Prediction*, Newcastle, United Kingdom, April 2008, pp. 43-50.

[26] T. Mak, C. D'Alessandro, P. Sedcole, P. Cheung, A. Yakovlev and W. Luk, "Implementation of Wave-Pipelined Interconnects in FPGAs," in *Proceedings of the 2008 ACM/IEEE International Symposium on Networks-on-Chip*, Newcastle, United Kingdom, April 2008, pp.213-214.

[27] T. Mak, P. Sedcole, P. Cheung, and W. Luk, "Wave-Pipelined Signalling for on-FPGA Communication," in *Proceedings of the 2008 IEEE International Conference on Field-Programmable Technology*, Taipei, Taiwan, December 2008, pp.9-16.

[28] P. Teehan, G. Lemieux, and M. Greenstreet, "Towards Reliable 5Gbps Wave-pipelined and 3Gbps Surfing Interconnect in 65nm FPGAs," in *Proceedings of the 2009 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, Monterey, CA, February 2009, pp. 43-52.

[29] P. Teehan, G. Lemieux, and M. Greenstreet, "Estimating Reliability and Throughput of Source-synchronous Wave-pipelined Interconnect," in *Proceedings of the 2009 ACM/IEEE International Symposium on Networks-on-Chip*, San Diego, CA, May 2009.

[30] J. Xu and W. Wolf, "Wave Pipelining for Application-Specific Networks-on-Chips," in *Proceedings of the 2002 ACM International Conference on Compliers, Architectures and Synthesis for Embedded Systems*, Greenoble, France, October 2002, pp. 198-201.

[31] J. Xu and W. Wolf, "A Wave-Pipelined On-chip Interconnect Structure for Networks-on-Chips," in *Proceedings of the 2003 IEEE Symposium on High Performance Interconnects*, Stanford, CA, August 2003, pp. 10-14.

[32] W. Burleson, M. Ciesielski, F. Klass, and W. Liu, "Wave-Pipelining: A Tutorial and Research Survey," *IEEE Transactions on Very Large Scale Intergration Systems*, Vol. 6 No. 3, March 1998, pp. 464-474.

[33] B. Winters and M. Greenstreet, "A Negative-Overhead, Self-Timed Pipeline," in *Proceedings of the 2002 IEEE International Symposium on Asynchronus Circuits and Systems*, Manchester, United Kingdom, April 2002, pp. 37-46.

[34] M. Greenstreet and J. Ren, "Surfing Interconnect," in *Proceedings of the 2006 IEEE International Symposium on Asynchronous Circuits and Systems*, Grenoble, France, March 2006, pp. 98-116.

[35] S. Yang, M. Greenstreet, and J. Ren, "A Jitter Attenuating Timing Chain," in *Proceedings of*

the 2007 IEEE International Symposium on Asynchronous Circuits and Systems, Berkeley, CA, March 2007, pp. 25-38.

[36] S. Trimberger and A. Lesea, "Programmable Logic Devices with Time-Multiplexed Interconnect," *United States Patent 6829756*, December 2004.

[37] S. Trimberger and A. Lesea, "FPGA with time-multiplexed interconnect," *United States Patent 7268581*, September 2007.

[38] R. Francis, S. Moore, and R. Mullins, "A Network of Timing-Division Multiplexed Wiring for FPGAs," in *Proceedings of the 2008 ACM/IEEE International Symposium on Networks-on-Chip*, Newcastle University, UK, April 2008, pp. 35-44.

[39] S. Brown, R. Francis, J. Rose, and Z. Vranesic, *Field-Programmable Gate Arrays*, Kluwer Academic Publishers, 1992.

[40] H. Hsieh, W. Carter, J. Ja, E. Cheung, S. Schreifels, C. Erickon, P. Freidin, L. Tinkey, and R. Kanuzawa, "Third-Generation Architecture Boosts Speed and Density of Field-Programmable Gate Arrays," in *Proceedings of the 1990 IEEE Custom Integrated Circuits Conference*, May 1990, pp.31.2.1–31.2.7.

[41] W. Tsu, K. Macy, A. Joshi, R. Huang, N. Walker, T. Tung, O. Rowhani, V. George, J. Wawrzynek, and A. DeHon, "HSRA: High-Speed, Hierarchical Synchronous Reconfigurable Array," in *Proceedings of the 1999 ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, Monterey, CA, February 1999, pp. 125-134.

[42] A. Singh, L. Macchiarulo, A. Mukherjee, and M. Marek-Sadowska, "A Novel High Throughput Reconfigurable FPGA Architecture," in *Proceedings of the 2000 ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, Monterey, CA, February 2000, pp. 22-29.

[43] A. Singh, A. Mukherjee, and M. Marek-Sadowska, "Interconnect Pipelining in a Throughput-Intensive FPGA Architecture," in *Proceedings of the 2001 ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, Monterey, CA, February 2001, pp. 153-160.

[44] D. Singh and S. Brown, "The Case for Registered Routing Switches in Field Programmable Gate Arrays," in *Proceedings of the 2001 ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, Monterey, CA, February 2001, pp. 161-169.

[45] J. Lamoureux and S. Wilton, "FPGA Clock Network Architecture: Flexibility vs. Area and Power," in *Proceedings of the 2006 ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, Montery, CA, February 2006, pp. 101-108.

[46] F. Rubin, "The Lee Path Connection Algorithm," *IEEE Transactions on Computers*, Vol. C23, No. 9, September 1974, pp. 907-914.

[47] Pico-Java Processor Design Documentation, Sun Microsystems, 1999.

[48] S. Ip, "Serial Enabled CAD Tool for the Evaluation of Serial FPGA Routing Architectures," *M.A.Sc. Thesis*, Ryerson University, Department of Electrical and Computer Engineering, Ryerson University, Ontario Canada, 2009.

[49] V. Betz and J. Rose, "Effect of the Prefabricated Routing Track Distribution on FPGA Area-Efficiency," *IEEE Transactions Very Large Scale Integration Systems*, Vol. 6, No. 3, September 1998, pp. 445-456.

[50] V. Betz and J. Rose, "FPGA Routing Architecture: Segmentation and Buffering to Optimize Speed and Density," in *Proceedings of the 1999 ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, Monterey, CA, February 1999, pp. 59-68.

[51] E. Ahmed and J. Rose, "The Effect of LUT and Cluster Size on Deep-Submicron FPGA Performance and Density," *IEEE Transactions on Very Large Scale Integration Systems*, Vol. 12, No. 3, March 2004, pp. 288-298.

[52] G. Lemieux, E. Lee, M. Tom, and A. Yu, "Directional and Single-Driver Wires in FPGA Interconnect", in *Proceedings of the 2004 IEEE International Conference on Field-Programmable Technology*, Brisbane, Australia, December 2004, pp. 41-48.