The Optimization of Interconnection Networks in FPGAs

Xiaolei Chen, Yajun Ha

Department of Electrical and Computer Engineering
National University of Singapore
Singapore
elehy@nus.edu.sg

Abstract. Scaling technology enables even higher degree of integration for FPGAs, but also brings new challenges that need to be addressed from both the architecture and the design tools side. Optimization of FPGA interconnection network is essential, given that interconnects dominate logic. Two approaches are presented, with one based on the time-multiplexing of wires and the other using hierarchical interconnects of high-speed serial links and switches. Design tools for both approaches are discussed. Preliminary experiments and prototypes are presented, and show positive results.

Keywords. field-programmable gate array, architecture, computer-aided design

1 Introduction

With the technology scaling enabling ever higher degree of integration, field-programmable gate arrays (FPGAs) nowadays contain rich logic, as well as a variety of customized functional blocks, such as block memory, digital signal processing (DSP) blocks and Ethernet controller. Hence, FPGA has also become an important medium to implement user designs. But the scaling technology also brings new challenges to the FPGA community. The challenges include, but are not limited to, the followings: 1) Leakage power needs to be contained; 2) Process variations need to be considered so that the FPGAs are variation- and fault-tolerant; 3) Scalable FPGA architectures are vital to fully exploit the substantially more transistors with each new generation of technology.

These challenges have a better chance to be addressed by combining innovations from both the architecture side and the computer-aided design (CAD) tools side. It is important to remember that FPGA architecture and FPGA CAD tools are highly interactive. On the one hand, CAD tools, especially the physical synthesis tools, must be optimized for the architecture, so that the architectural features can be fully exploited. On the other hand, the necessity of controlling the complexity of CAD tools and reducing compile time means that a more regular architecture is preferred.

Dagstuhl Seminar Proceedings 10281
Dynamically Reconfigurable Architectures
http://drops.dagstuhl.de/opus/volltexte/2010/2842

The ongoing research the authors present in this paper mainly targets on the optimization of interconnection network in FPGAs. This is motivated by the fact that logic and interconnect are un-balanced in FPGAs. It is a well-known fact that the programmable interconnects in FPGAs dominate the logic as the main contributor to the area, delay and power consumption. To quantify this un-balance, the authors used VPR 5 [1] to map 20 MCNC benchmarks [2] to island-style FPGAs [3], assuming PTM 90nm CMOS process [4]. Fig. 1 shows the profile of interconnects versus logic area ratio for the 20 benchmarks. It can be observed that, for all but three of the benchmarks, the area consumed by interconnects is at least 10x that of the logic. Fig. 2 profiles the circuit critical path delay breakdown between logic and interconnects. Again, it is found that, for all but three of the benchmarks, interconnects account for at least 40% of the critical path delay. The similar un-balance can be found for the power. It has been reported in [5] that 52% of the dynamic power consumption in a commercial FPGA is due to interconnects.

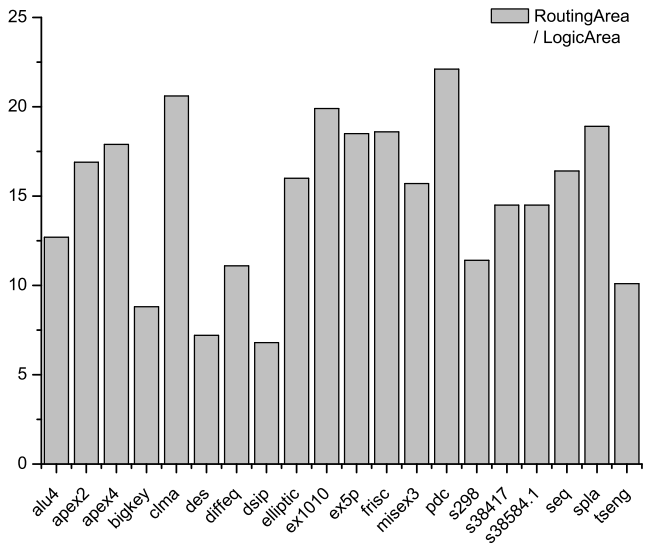

Fig. 1. Ratio of interconnects area versus logic area of 20 MCNC benchmarks

This paper presents two approaches which focus on the optimization of interconnection network in FPGAs. The first approach, which is covered in Section 2, centers on the idea of time-multiplexing the FPGA interconnects. The second approach, which is covered in Section 3, proposes using a hierarchical network

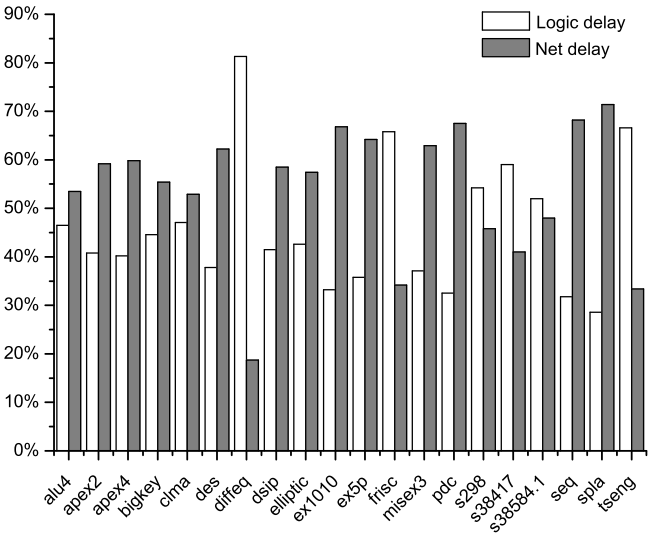

Fig. 2. Critical path delay breakdown of 20 MCNC benchmarks

of high-speed serial links and switches to build scalable FPGAs. Finally, Section 4 concludes this paper.

2 Time-multiplexed interconnects in FPGAs

This section describes an architecture and the CAD tool enhancement to support time-multiplexed interconnects for FPGAs. Preliminary results are also presented.

2.1 Motivation

An important premise of the idea of time-multiplexed interconnects is that intra-clock cycle idleness is significant at the FPGA interconnection networks. More specifically, the delay for a signal to propagate along a wire segment is only a small portion of the clock cycle. An intuitive way to understand this phenomenon is that the circuit critical path usually spans a number of logic blocks and nets, connecting these blocks, between the primary inputs and the primary outputs. If one examines one wire segment in this critical path, one may find that its delay is far less than the whole critical path delay.

As an example, the authors examined the implementation result of the largest MCNC benchmark, *clma*, to an island-style 90nm FPGA [1]. It was found that, while the circuit critical path delay after placement and routing is about 9.5 ns, delay of most nets (96.5%) are less than 1 ns. In other words, many wires are idle for a significant amount of time in a clock cycle.

2.2 Architecture

The proposed method to improve the utilization of interconnects makes use of *time-multiplex switches* (TM switches). A TM switch has multiple configurations (or contexts), as shown in Fig. 3 (b) and (c). A TM switch cycles through its multiple configurations in one clock cycle, following the micro-cycle model proposed by Trimberger in [6], as illustrated in Fig. 3 (d). As a TM switch cycles from one configuration to the next, it effectively turns on and off the connection between the two wires it connects. By replacing the conventional switches in switch blocks and connection blocks with TM switches, one can potentially achieve time-multiplexing of the wires.

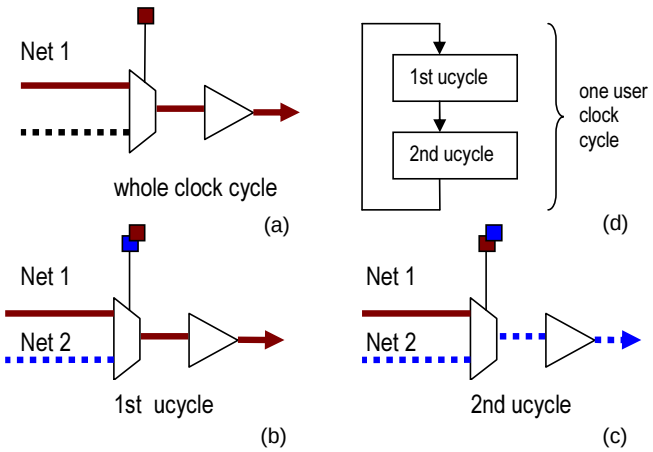

Fig. 3. (a) A conventional switch with only one configuration; (b)(c) A TM switch with two configurations; (d) A TM switch cycles through the micro-cycles in one user clock cycle

An example shown in Fig. 4 illustrates this idea. Two nets pass the same routing channel between the two configurable logic blocks (CLBs). A conventional FPGA architecture would have to reserve two wires, one for each net, in this channel for these two nets (Fig. 4 (a)). In FPGAs with TM switches, one

[1] As described in Section 1, VPR 5 is used for implementation, and PTM 90 nm CMOS process is assumed.

wire is enough and can be time-shared by these two nets, if it happens that one net arrives and leaves this channel early in the clock cycle, and the other one late in the clock cycle (Fig. 4 (b)).

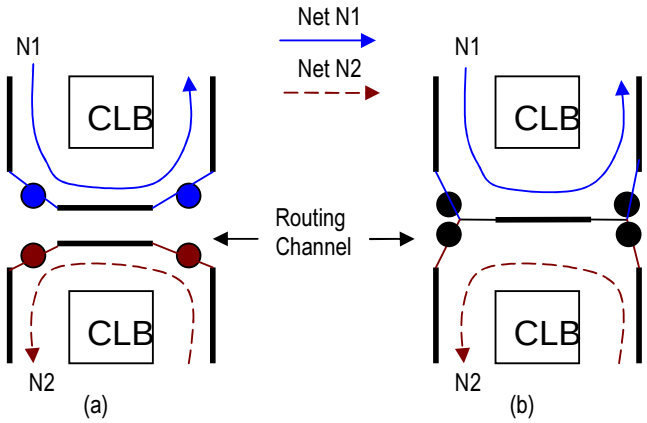

Fig. 4. Two nets pass the same routing channel: (a) Using conventional switches, two wires are needed; (b)Using TM switches, one wire suffices

2.3 CAD tool enhancement

To share a wire segment between two candidature nets, the following two conditions are necessary: 1) These two nets are not overlapping in the time domain; 2) The routes of these two nets pass exactly the same channel. The authors have incorporated these two conditions into the VPR [7] routing algorithm, so that the router can identify nets that can time-multiplex wires.

2.4 Results and Discussion

Preliminary experiments showed positive results. Minimum channel width and critical path delay are the two main metrics that are used to compare the implementation result of benchmarks on the conventional FPGA architecture with that on the proposed FPGA architecture with time-multiplexed interconnects. For the 16 MCNC benchmarks the authors used, in average 11.5% reduction in the minimum channel width was achieved by the proposed architecture, but with 1.5% increase in the critical path delay. Given the same channel width, the proposed architecture achieved 8.2% reduction in the critical path delay. More details on the experiments and results can be found in [8] and [9].

A well-understood problem with the time-multiplexed architecture modeled after the Trimberger's model is that dividing the user clock cycle into too many micro-cycles will result in excessive power consumption. On the other hand, the

authors found that two micro-cycles in one user clock cycle only resulted in limited chances for nets to time-multiplex wires. Ongoing work by the authors suggests that dividing a user clock cycle into four micro-cycles be a good choice. Future work will examine the effect of the timing model accuracy and process variations.

3 sFPGA architecture

This section describes the sFPGA architecture and its CAD flow. A preliminary prototype of the sFPGA architecture is presented.

3.1 Motivation

In current FPGAs, the switching requirement grows super-linearly with the number of logic blocks. Hence, the area requirement of interconnects will grow similarly and decrease the logic density, with the increase in gate count. In other words, the current FPGA architecture scales poorly.

This problem can be mitigated through the reuse of wires by serializing multiple nets onto a single wire. This is inspired by the way multiple FPGAs are connected. Currently, high-speed and scalable connections among multiple FPGAs are implemented using multi-gigabit transceivers (MGTs) -based serial interconnects. Furthermore, the concept of hierarchy, as it is applied in Internet, can be introduced to separate local routing from global routing.

3.2 Architecture

Fig. 5 illustrates the proposed sFPGA architecture. It is a scalable FPGA architecture using hierarchical routing network employing high-speed serial links and switches. It has two levels of hierarchy, namely the base level and the higher level. The base level consists of FPGA tiles (for example, A_0 as shown in Fig. 5). Inside one FPGA tile, the logic and interconnects are organized in the same way as in current FPGAs. The higher level consists of switch units (for example, S_0 and X_0 as shown in Fig. 5). The communications between tiles are routed by switch units using high-speed serial interconnects. More details about the switch units can be found in [10].

3.3 CAD flow

Fig. 6 gives an overview of the CAD flow. In the pre-processing stage (Stage I), logic synthesis is performed on the whole user design. This generates an estimation on the size (in terms of resource usage) of each module in a hierarchical design. This estimation will be passed to the design partitioning stage (Stage II), during which the whole design is partitioned into clusters. Each cluster is then mapped to one tile in sFPGA. The partitioning is constrained such that each

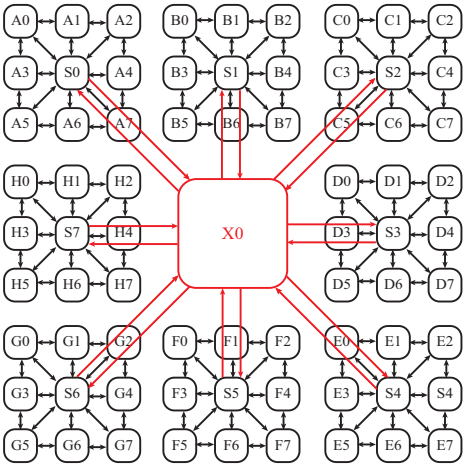

Fig. 5. sFPGA architecture block diagram

cluster can fit into one tile and signal traffic among different clusters is minimized. Then logic synthesis and optimization are performed for each tile (Stage III). Stage IV is the physical implementation, which includes two tasks, namely, tile generation and routing path generation. Tile generation corresponds to the placement and routing at each tile, while routing path generation is to route inter-tile nets. Finally, the FPGA programming file is generated (Stage V).

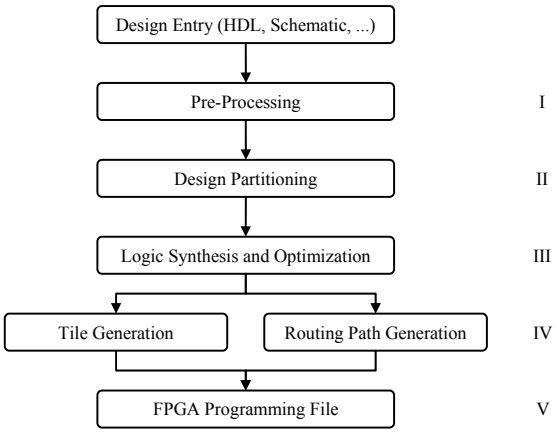

Fig. 6. CAD flow for sFPGA

3.4 Prototyping and Discussion

An instance of the sFPGA architecture was emulated using three FPGA boards. The first board emulates a switch unit, and the other two emulate two clusters each with two FPGA tiles and a switch. This is illustrated in Fig. 7. A Xilinx MGT running at 1.5Gbps using SATA interface and encapsulated by the Xilinx Aurora [11] IP with a 16-bit data interface is used to emulate the transceiver of the sFPGA architecture. The IO transceivers are implemented in FPGA fabric using Verilog HDL. Xilinx ISE10.1 is used for the implementation.

A JPEG baseline encoder was mapped to the sFPGA prototype. The JPEG encoder is partitioned into four functional blocks, namely, color space conversion (*rgb2ycbcr*), discrete cosine transform (*dct*), quantization (*quantization*) and entropy encoding (*entropy*). The serial interconnect was successfully demonstrated to be able to deliver the signals across the partitions. It was found that latency in transport is very high mainly due to high-latency transceivers thus limiting application domain to GALS (globally asynchronous and locally synchronous) designs only. However, with the advancement in transceivers, this can be extended to purely synchronous designs as well.

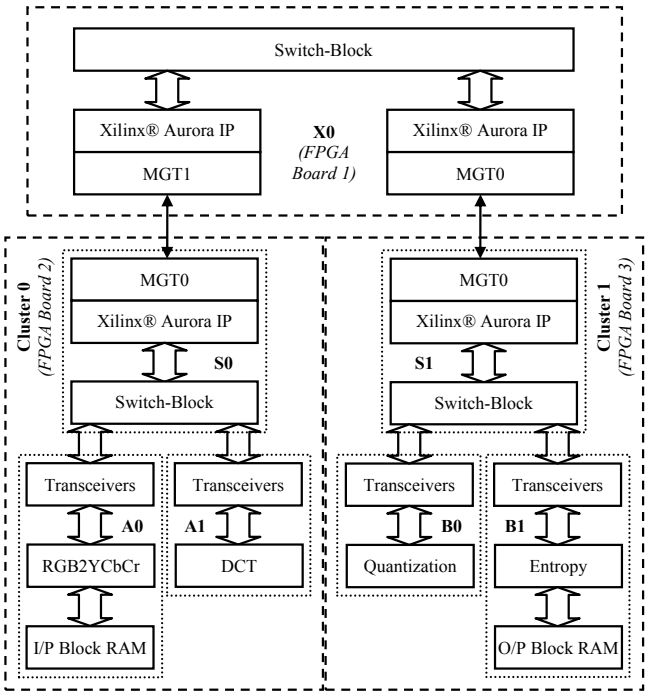

Fig. 7. Block diagram of the emulation prototype

4 Conclusions

This paper presents two approaches which aim to optimize interconnection networks in FPGAs. The proposed FPGA architecture with time-multiplexed interconnects can improve the utilization of wires, hence save interconnects area and potentially achieve better timing. The proposed sFPGA architecture improves the scalability of FPGA interconnection network by using high-speed serial links and applying the concept of hierarchy. The proposed architectures are justified by promising preliminary results. Future work needs to thoroughly investigate the impact of architecture changes.

References

1. Luu, J., Kuon, I., Jamieson, P., Campbell, T., Ye, A., Fang, W.M., Rose, J.: Vpr 5.0: Fpga cad and architecture exploration tools with single-driver routing, heterogeneity and process scaling. In: FPGA '09: Proceeding of the ACM/SIGDA international symposium on Field programmable gate arrays, New York, NY, USA, ACM (2009) 133–142
2. Yang, S.: Logic synthesis and optimization benchmarks user guide version 3.0 (1991)
3. Betz, V., Rose, J., Marquardt, A., eds.: Architecture and CAD for Deep-Submicron FPGAs. Kluwer Academic Publishers, Norwell, MA, USA (1999)
4. Cao, Y., Sato, T., Orshansky, M., Sylvester, D., Hu, C.: New paradigm of predictive mosfet and interconnect modeling for early circuit simulation. In: Custom Integrated Circuits Conference, 2000. CICC. Proceedings of the IEEE 2000. (2000) 201 –204 http://ptm.asu.edu/
5. Shang, L., Kaviani, A.S., Bathala, K.: Dynamic power consumption in virtex-ii fpga family. In: FPGA '02: Proceedings of the 2002 ACM/SIGDA tenth international symposium on Field-programmable gate arrays, New York, NY, USA, ACM (2002) 157–164
6. Trimberger, S.: Scheduling designs into a time-multiplexed fpga. In: FPGA '98: Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays, New York, NY, USA, ACM (1998) 153–160
7. Betz, V., Rose, J.: Vpr: A new packing, placement and routing tool for fpga research. In: FPL '97: Proceedings of the 7th International Workshop on Field-Programmable Logic and Applications, London, UK, Springer-Verlag (1997) 213–222
8. Liu, H., Chen, X., Ha, Y.: An architecture and timing-driven routing algorithm for area-efficient fpgas with time-multiplexed interconnects. In: Field Programmable Logic and Applications, 2008. FPL 2008. International Conference on. (2008) 615 –618
9. Liu, H., Chen, X., Ha, Y.: An area-efficient timing-driven routing algorithm for scalable fpgas with time-multiplexed interconnects. In: Field-Programmable Custom Computing Machines, 2008. FCCM '08. 16th International Symposium on. (2008) 275 –276
10. Syed, R., Chen, X., Ha, Y., Veeravalli, B.: sfpga2 - a scalable gals fpga architecture and design methodology. In: Field Programmable Logic and Applications, 2009. FPL 2009. International Conference on. (2009) 314 –319
11. Xilinx: LogiCORE IP Aurora 8B/10B v5.2. Xilinx (2010)