Building Asynchronous Circuits With JBits

Eric Keller

Xilinx Inc.
2300 55^{th} Street
Boulder, CO 80301
Eric.Keller@xilinx.com

Abstract. Asynchronous logic design has been around for decades. However, only recently has it gained any commercial success. Research has focused on a wide variety of uses, from microprocessor design to low power circuits. The fact that design tools and designer experience are geared more towards synchronous circuit design has limited the acceptance and popularity of asynchronous design. The JBits™ API provides low level access to the configuration of resources in a Xilinx® FPGA. Because of the control given to the user, the JBits API is an ideal design environment for implementing asynchronous circuits on mainstream FPGAs. An asynchronous full adder was implemented on a Virtex® FPGA. The design of the circuit is described as well as modifications to the design tools to simplify asynchronous circuit specification.

1 Introduction

As ASICs increase in size, asynchronous circuits are often cited as a possible alternative to standard synchronous circuit designs. Clock distribution can be a problem in larger ASICs, whereas with an asynchronous circuit there is no clock. While synchronous circuits currently are the main design technique, asynchronous circuits have been designed that are either faster or consume less power. The main problem with asynchronous design is the fact that commercial design tools are geared towards synchronous circuits.

The JBits API provides direct access into the configuration bitstream of a Xilinx FPGA. It provides abstraction layers that provides a design environment similar to a structural HDL while concurrently affording low-level control. The level of control is ideal for asynchronous circuits. Using the JBits API, an asynchronous full adder was implemented in a Virtex device.

2 Asynchronous Logic

Asynchronous logic has been actively researched for several decades [6] [8] [9]. However it has yet to reach mainstream use. Recently there has been a renewal of interest among the research community. Commercial products are slowly coming to market [5] [3] [7]. The advantages afforded by asynchronous logic are making it an increasingly viable alternative to synchronous design as ASICs continue to grow in size. The advantages generally cited are modularity, average-case performance, low power consumption, and no clock distribution.

3 Asynchronous Design with JBits

The JBits API provides access into a Xilinx FPGA configuration bitstream. There is access to every resource, such as routing programmable interconnect points (PIPs), LUT values, and CLB configurations. Because of the level of control, the JBits tool kit provides an ideal environment to design asynchronous circuits in mainstream FPGAs. Research has either focused on implementing asynchronous circuits on mainstream FPGAs [10] or on creating architectures better suited for asynchronous designs [11] [4]. Payne [4], for example, suggests an architecture that has a programmable delay element. However, with the JBits API, a programmable delay element can be implemented by directly controlling the routing. Althouse [13] created a programmable delay in the XC4000™ devices with a 50 ps resolution using the JBits API.

3.1 NCL Full-Adder

NULL Convention Logic™ [3] (NCL™) is a technique similar to Delay Insensitive Minterm Synthesis (DIMS) [12]. It uses four-phase signaling with a dual-rail communication protocol. While DIMS uses C-elements and OR gates, NCL provides for optimizations by allowing M-of-N gates. Using M-of-N gates results in smaller circuits than what would be created with DIMS. The M-of-N gate works by transitioning to a high value when M of the inputs are high. The transition to a low value does not occur until all of the inputs are low. Figure 1 shows the implementation of a 2-of-3 gate in a Virtex LUT. While it may appear that using feedback by going through general purpose routing may cause a problem, it does not. Because of the signaling protocol, the feedback only needs to have a shorter delay than the output going through circuitry that requests the NULL or DATA phase, and through the circuitry of the previous stage that sends the DATA or NULL signal. Also, hazards do not cause a problem because the transitions that can occur on the LUT and the LUT configuration do not result in any situations that cause hazards.

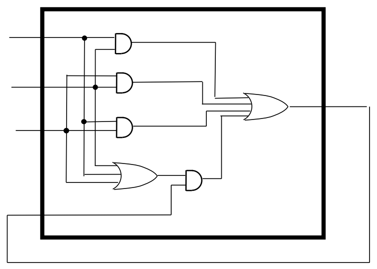


Fig. 1. Diagram showing the implementation of a 2-of-3 NCL gate in a Virtex LUT.

NCL is a case of a Quasi Delay Insensitive circuit. The delay sensitivity lies in the orphan of a circuit [2]. However, the timing constraints are a lot less stringent than an isochronic fork. In most cases the circuit can be considered delay insensitive. To analyze the timing constraints, only partial circuits must be analyzed.

Pipelined circuits can be created using asynchronous registers. In NCL this involves a circuit checking that the input has a complete data set. The register will pass the data through once an acknowledge signal is sent from the circuit that will receive the data. Once the data is passed through, the register will send a signal to the previous circuit requesting a return-to-zero (or NULL) transition. Figure 2, shows an example of an NCL register.

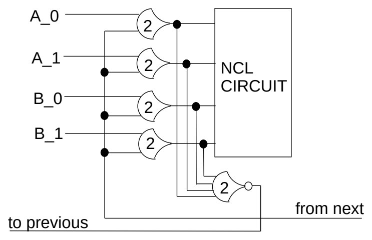


Fig. 2. Diagram showing an 2 bit asynchronous register in NCL.

Using the JBits design tool, an NCL full adder was created. The full adder is parameterizable by allowing variable width inputs and requires three Virtex slices per bit. The M-of-N gates were each made into cores and required a LUT with feedback. Interconnection between the M-of-N gates was done through the use of JRoute [1]. While JRoute does not support the ability to specify a delay for a given net, this control is not necessary when using NCL. Only a subset of the circuit needs to be analyzed for timing, and if a problem is found it is handled in a special case, such as a programmable delay element core. The special case would modify the net routed by JRoute with the use of templates to add delay. Shown in Figure 3 is a single stage of an NCL full adder circuit.

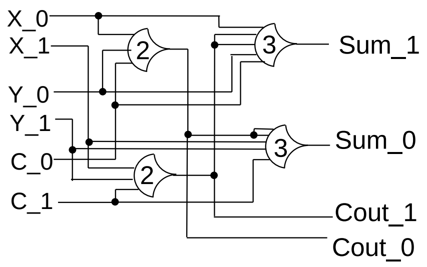


Fig. 3. Diagram showing the implementation of a full adder using NCL.

3.2 Modification to RTP Core Specification

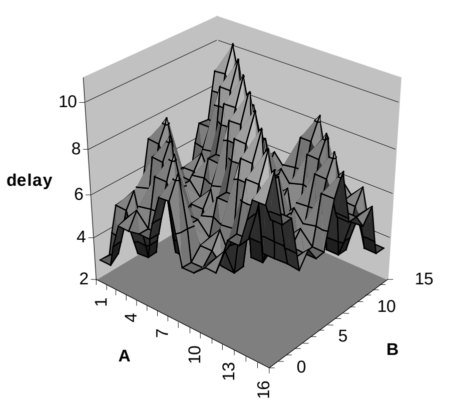
The JBits run-time parameterizable (RTP) core specification provides a means for abstracting away the low level configuration calls, thereby creating an environment similar to traditional hardware design languages (HDLs), while concurrently affording the ability to make bitstream level modifications. However, the RTP Core specification did not provide clean support for dual-rail asynchronous circuits. The problem occurred when connecting high level cores; the Net and Bus classes were meant for single signals. However, in dual-rail logic a net is equivalent to a bus with two nets. For this a DualRailNet was created. It extends the Bus class and can only be of width two. A DualRailBus was also created and is similar to the Bus class, except that DualRailNet was used in place of Net. A method to traverse the hierarchy of cores to the physical pins was created. JRoute is then used to make the physical connections.

4 Results

One characteristic of asynchronous circuits is that they exhibit average case performance. Shown in Figure 4 is a graph of the delay, in nanoseconds, through a 4-bit NCL full adder for each combination of inputs. A Virtex device was used with a -6 speed grade and the circuit requires 6 CLBs. The graph looks nearly identical with a carry in of one as opposed to a carry in of zero. For a carry in of zero, the delay values range from 3.0463 ns to 10.0507 ns with an average delay of 6.0356 ns. For a carry in value of one, the delays range from 3.0537 ns to 10.0507 ns with an average of 6.2257 ns. The variation in the delays comes from the degree to which the carry bit propagates. If it must propagate through each stage then the delay will be longer than the case when no stages produce a carry out and therefore the output will be equal to the delay of a single stage. The average performance is worse than a synchronous full adder implemented on a Virtex device because Virtex devices are designed for synchronous circuits and contain such optimizations for common circuits as the fast carry chain. However, the graph does show that average case performance is exhibited.

5 Conclusions and Future Work

An asynchronous full adder was created along with a synchronous interface. The JBits design suite is believed to be an ideal environment for asynchronous circuitry because of the level of control provided. Research in the field has been ongoing for several decades. As more commercial applications find success with asynchronous design, CAD tools will become more supportive. Further research into a combination of asynchronous design and run-time reconfiguration is being explored. Run-time reconfiguration has shown many advantages in synchronous systems that are possibly applicable to asynchronous systems. The JBits API is an important tool to enable the design of run-time reconfigurable systems.



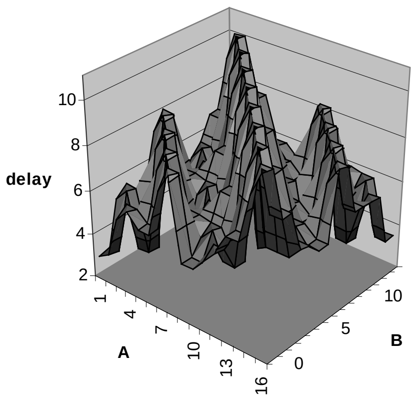


Fig. 4. Graph showing the delay, in nanoseconds, through the full adder circuit for each combination of inputs. The left graph has a carry in of 0, and right graph has a carry in of 1.

Acknowledgements

This work was supported by DARPA in the Adaptive Computing Systems (ACS) program under contract DABT63-99-3-0004.

NCL and NULL Convention Logic are trademarks of Theseus Logic, Inc.

References

1. E. Keller, "JRoute: A Run-Time Routing API for FPGA Hardware," Proc. RAW2000, pp 874-881.
2. K. Fant, R. Stephani, R. Smith, and R. Jorgenson. "The Orphan in 2 value NULL Convention Logic," Technical document, http://www.theseus.com/PDF/orph_paper.pdf.
3. K. Fant and S. Brandt, "NULL Convention Logic," Technical document, http://www.theseus.com/PDF/ncl_paper.pdf.
4. R. Payne, "Self-Timed FPGA Systems," Proc. FPL95.
5. A. Martin *et al*, "The design of an asynchronous MIPS R3000 microprocessor," *Advanced Research in VLSI*, September 1997.
6. I. Sutherland, "Micropipelines," *CACM*, 32(6):720-738, June 1989.
7. H. van Gageldonk *et al*, "An asynchronous low-power 80C51 microcontroller," *IEEE Symposium on Advanced Research in Asynchronous Circuits and Systems*, April 1998.
8. S. Unger, *Asynchronous Switching Circuits*, New York, NY: Wiley-Interscience 1969.
9. J. Udding, "A Formal Model for Defining and Classifying Delay-Insensitive Circuits and Systems," *Distributed Computing*, vol 1, no. 4, pp 197-204, 1986.
10. E. Brunvand, "Implementing Self-Timed Systems with FPGAs," Proc. FPL91.
11. S. Hauck, G. Borriello, S. Burns and C. Ebeling, "Montage: An FPGA for Synchronous and Asynchronous Circuits", Proc. FPL92.
12. H. Hulgaard and P. Christensen, "Automated Synthesis of Delay Insensitive Circuits," M.Sc. thesis (IDE 511), Dept. of Computer Science, Tec. University of Denmark, Lyngby, 1990.
13. C. Althouse, Technical Document, http://www.xilinx.com/xilinxonline/jbits_appexam.htm