PLFire: A Visualization Tool for Asynchronous Phased Logic Designs

K. Fazel, M. A. Thornton
Department of Computer Science and Engineering
Southern Methodist University
Dallas, TX 75275
{kfazel, mitch}@engr.smu.edu

R. B. Reese
Department of Electrical and Computer Engineering
Mississippi State University
Mississippi State, MS 39762
reese@ece.msstate.edu

Abstract

We present a visualization tool called PLFire, *which allows a user to observe the behavior of a Phased Logic (PL) circuit. Phased logic is a technique for realizing self-timed circuitry that is delay-insensitive and requires no global clock. One advantage of self-timed circuits is that throughput is based on average propagation delays and not worst-case delay. By being able to visualize the operation of a PL circuit, including the token flow, a designer gets a better understanding of what features of a design have the greatest impact on performance.*

1 Introduction

Phased Logic (PL) is an asynchronous design methodology devised in [1] that produces delay-insensitive circuits that do not require a global clock signal. An implementation of a basic PL [2, 3] gate has resulted in a circuit that is well-suited for implementation in a Field Programmable Gate Array (FPGA) type of device. A major advantage of the PL design methodology is that it allows designers to use current synthesis tools and design styles for synchronous digital circuits.

With regards to PL optimizations, as different optimization techniques are implemented for PL, it becomes increasingly difficult to gauge the effectiveness of a technique without the use of some visualization tool. This is what our visualization tool, *PLFire*, addresses. Current optimization techniques such as Early Evaluation [10] and Gated PL change the flow of data tokens with a PL circuit. Being able to visualize these changes allows a researcher to better understand how these techniques affect a circuit on a global level.

2 Phased Logic

A PL netlist can be thought of as a marked graph with data tokens flowing throughout the graph. Each data token has a phase that is either even or odd. A data token is represented by a dual-rail signal that uses *Level Encoded Dual Rail* (LEDR) encoding [4]. A PL gate has an internal state bit used to represent the gate phase and a phased logic gate fires whenever all of the phases of the inputs matches the internal gate phase.

2.1 PL Design Cycle

The methodology for mapping a clocked design to a PL design is:

1) An RTL VHDL description of a clocked circuit is generated.
2) A commercial synthesis tool (Synopsys Design Compiler) is used to synthesize the RTL to an EDIF netlist of D-flip-flops and combinational gates.
3) A mapping program, called *plmapper* [8], is used to convert the EDIF netlist to a PL netlist in VHDL format.

3 Visualization Tool

As the size of circuits that are being investigated in PL research grows, mentally keeping track of a design's PL structure and behavior becomes increasingly difficult. The purpose of the visualization tool, *PLFire*, is to help a designer visualize the behavior of a PL circuit. Fig. 1 provides a screenshot of the program.

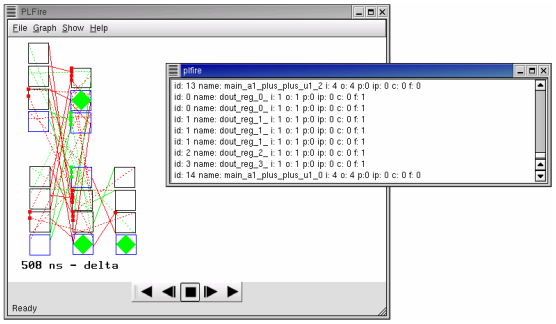

Figure 1. Screenshot of *PLFire*

Moreover, the use of *PLFire* aids in the development of optimization techniques and tools. As different optimization techniques are implemented for PL, verifying whether an optimization truly has an impact on the overall circuit is desirable. Current optimization techniques center on the ability to maximize throughput of token flow and are implemented as global and local transformations of the netlist. Being able to see these optimizations gives the researcher some understanding of

1530-1591/03 $17.00 © 2003 IEEE

their overall effect. Furthermore, this tool allows the investigation of the interoperability of these techniques.

In order to use the tool, two previously generated files are needed: a viznet file and a viztiming file. The viznet file contains the netlist information of the PL design. The viznet file is generated by the mapping program that converts a synchronous circuit in EDIF format to a PL circuit in VHDL format. The viztiming file contains the firing information of a PL design. The viztiming file is generated when the VHDL file is simulated.

4 Implementation

PLFire is written in C/C++ and uses the Qt windowing libraries and OpenGL. Qt is a free set of libraries provided by [5] that allows a programmer to create graphical user interface (GUI) programs for the UNIX/Windows/Macintosh platforms. OpenGL [6] is a free, standardized set of graphics libraries that many platforms support.

The current design environments are Sun Solaris using Qt 3.0.1 and Mesa3D libraries [7]. Also, Microsoft VC++ 6.0 is used for additional testing under the Windows platform.

The infrastructure of the program was designed with object-oriented techniques to allow for future revisions and additions to the code. In particular, a set of classes to represent the various aspects of a PL circuit was implemented. Using such techniques, data structures were implemented with the mindset that components should be easily modified and replaceable, and that future components and enhancements should be easily implemented.

5 Example

To illustrate the use of *PLFire*, an analysis of the behavior of a 4-bit ripple carry adder is provided. In Figure 2, the structure of the PL 4bit adder (PL4B) as displayed by *PLFire* is shown. The PL gates are grouped as input/output registers, the summing logic, and the carry out logic.

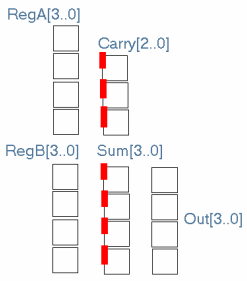

Figure 2. 4-bit adder in *PLFire*

The display shows various aspects of the circuit, including the tokens, feedback signals, and current phase of the gates and signals. It is seen that the mapper has placed the initial tokens on the sum and carry components. As the circuit simulation is visualized, the display will update these components and the tokens to reflect the current state of the circuit.

To visualize the simulation of the circuit, buttons formatted like a VCR panel are used. When the circuit is visualized, the token flow that dictates the behavior of the circuit is evident. Also, the gates are color coded to indicate which cycle they are currently in. As time advances and the colors change, one can see the rippling of data through the adder circuitry in PLB4 as expected.

6 Conclusion

In this description we introduced a phased logic visualization tool, *PLFire*, which allows a designer to visualize the behavior of a PL circuit. The visualization tool, other PL tools, and information about our research can be found at [8].

References

[1] D.H. Linder, Phased Logic: A Design Methodology for Delay Insensitive Synchronous Circuitry. PhD Dissertation, Mississippi State University, 1994

[2] R. B. Reese, M. A. Thornton, and C. Traver. Arithmetic logic circuits using self-timed bit-level dataflow and early evaluation. In Int'l Conf. on Comp. Design, pages 18-23, 2001

[3] R. B. Reese and C. Traver. Synthesis and simulation of phased logic systems. In International Workshop on Logic Synthesis, pages 255-259, Dana Point, California, 2001.

[4] M. E. Dean, T. E. Williams, and D. L. Dill. Efficient self-timing with level-encoded 2-phase dual-rail (ledr). In Advanced Research in VLSI, 1991.

[5] Trolltech Homepage. http://www.trolltech.com

[6] SGI: OpenglGL Homepage. http://www.sgi.com

[7] Mesa3D Graphics Library Homepage. http://www.mesa3d.org

[8] Phased Logic Homepage. http://www.erc.msstate.edu/mpl/projects/phased_logic

[9] M. Ligthart, K. Fant, R. Smith, A Taubin, and A. Kondratyev. Asynchronous design using commercial HDL synthesis tools. In Advanced Research in VLSI, 2000.

[10] M. A. Thornton, K. Fazel, R. B. Reese and C. Traver, Generalized Early Evaluation in Self-timed Circuits, In Proceedings of the Conference on Design, Automation and Test in Europe, pages 255-259, 2002.