A Unified Formal Framework for Analyzing Functional and Speed-path Properties

Oswaldo Olivo Sandip Ray
University of Texas at Austin
{olivo,sandip}@cs.utexas.edu

Jayanta Bhadra Vivekananda Vedula
Freescale Semiconductor Inc.
{RA9113,B35209}@freescale.com

***Abstract*— We develop a formal tool for speed-path analysis and debug. We encode speed-path requirements in a formal hardware description language providing the semantics of both the functional behavior and timing constraints, and the disciplined use of an SMT solver to analyze speed-path requirements. We are applying our framework for speed-path analysis of several RTL designs from Opencores.**

I. Introduction

A critical part of the chip design is to push higher the clock frequency. The process is normally iterative, with each iteration involving debug engineers to improve clock frequency through speed-path debug. Unfortunately, the process is heavily manual. Furthermore, extant speed-path debug and diagnosis techniques are only applicable during post-silicon verification. A major reason for the emphasis towards post-silicon rather than pre-silicon is the availability of accurate timing information at the latter phase. However, a consequence of the emphasis is the postponing of this important facet of design verification until the (preliminary) silicon is available. Furthermore, post-silicon verification induces significant complexity due to limited observability: only a few of the internal signals are available during normal chip operation. Thus, current speed-path analysis requires significant human expertise to solve the complex problem of *inferring* possible timing constraints from a partially observed execution trace, and isolate them from functional bugs and manufacturing defects. The problem is exacerbated in the context of asynchronous SoC designs, where multiple components with different clock domains are integrated in a single chip; the non-determinism introduced by the difference in clock domains as well as asynchronous communication makes it difficult to isolate a speed-path error without full observability of the design.

In this paper, we develop a speed-path debug and analysis framework as part of *pre-silicon verification*. We develop a formal model of hardware that includes both functional and timing behavior. To account for the fact that accurate timing information is unavailable during pre-silicon analysis, our timing model is parameterized on timing parameters. Based on the model, we develop a tool that determines the necessary timing relations among different circuit elements for the appropriate functional behavior. Some of the timing parameters introduced in our model are inspired by results from static timing analysis [1]; however, through a unified formal model for functionality and timing, we can make use of functional invariants for speed-path analysis. Our framework is compositional; individual modules can be separately analyzed and their speed-path constraints accounted for while analyzing modules that contain their invocations. To our knowledge, our tool provides the first formal infrastructure for compositional, parameterized, pre-silicon analysis for speed-path constraints.

II. Tool Overview

Our tool has the following key ingredients.

- A formalized hardware description language (HDL)
- A formal, parameterized timing semantics
- A tool for reducing speed path constraints to SMT formulas
- Connection to an external SMT solver to discharge the above constraints

Hardware Description Language

The HDL used in the framework is EMOD [2], a hierarchical, occurrence-oriented hardware description language for expressing gate-level designs. The EMOD analysis framework [3] includes a translator from synthesizable Verilog, and a variety of reasoning techniques including symbolic simulation, Boolean satisfiability solving, and theorem proving. The language has a formal semantics, through a deep embedding in the logic of the ACL2 theorem prover. The language has been used in formal functional verification of industrial hardware designs, *e.g.*, in the verification of floating-point units of the VIA Nano^{TM} microprocessor [4].

Timing Semantics

We have developed a model and associated analysis toolsuite for timing constraints on top of the EMOD analysis infrastructure. Timing constraints in combinational circuits arise from gate delay, which restricts the amount of time for which a signal value persists at a wire. In order to extend the EMOD infrastructure for speed-path analysis, we develop a formal model of design gates, parameterized with both functional behavior and timing constraints. More precisely, a gate G is viewed as a pair (d_G, op_G), where d_G is the gate delay and op_G is the type of operation computed by G. An *execution* of a circuit is an assignment of a sequence of *signals* to each gate of the design. A signal x at the input to gate G is described by the 3-tuple $x = (A, H, V)$, where A and H are non-negative integers representing *arrival time* and *hold*

time respectively, and V is a Boolean. Informally, the arrival time of x at gate G is an integer that represents the time point when the value of x is available at G. The hold time is the number of time units that the signal must persist in order for the output to be reliably computed, and V is the value of the signal. The arrival time is an absolute time number, while the hold time is in relative time units: informally, an arrival time of 7 represents arrival of the signal x at 7 time units after the start of execution, while a hold time of 5 indicates that the value of the signal persists for 5 units after arrival.

There is a potential problem when dealing with sequential circuits: The output value depends on the state, and the feedback poses some difficulties in how to calculate the intervals of the signal. To address this operation, we formalize the latching effect separately from signal propagation delays. In particular, our formal model of latching has the following characteristics.

- There is a (parameterized) latch time.
- Let d be the latch time, and A be the arrival time of a signal in the latch. The signal is available at the output of the latch after $(A + d)$ time units.
- The output of the latch "holds" the value up to the end of the current clock cycle.

We now pose the *basic speed-path problem* for a combinational circuit as follows. Given a parameterized network of gates, determine the arrival and hold times for input signals so that the computation of each gate in the circuit is consistent with the functional semantics of the circuit without timing.

Discharging Speed-path Constraints through SMT

To solve the speed-path problem, we note that the formulation above reduces the problem to a collection of constraints in the union of two theories: Boolean Algebra and integer inequalities. Note that the solution of integer arithmetic is intractable. However, the integers arising in the problem can bounded by the number of units in the clock cycle; thus the problem can be solved by an SMT solver for Boolean and bit-vector arithmetic. In particular, we develop an interface to turn the problem into a formula solved by the Yices solver [5].

III. Results

We have implemented a tool to discharge speed-path constraints on Verilog circuits. The tool is implemented in Java, and works by (1) translating the Verilog design to EMOD (2) augmenting the EMOD design with the timing information, and (3) generating the SMT problem for speed-path which is then discharged by Yices. We have applied the tool on several publicly available hardware design benchmarks available from Opencores (`http://opencores.org`). At this writing, our tool can handle designs shown in Fig. 1. Note that the designs are taken from diverse applications. The analysis of each design requires only a few seconds with our tool.

Design	LOC	Load Time
Raggedstone PCI Spartan-3 Board	534	0.421
FPU Double VHDL	2066	9.716
Fast Hadamard Transforms	106	0.237
WC LCD Display Controller	1153	1.318
SPI Core	456	0.380

Fig. 1. Opencore designs used in Speed-path Analysis

IV. Conclusion

We have developed a formal framework for analyzing speed-path constraints together with functional behavior within the same logical framework. The approach permits pre-silicon speed-path analysis and debug, thereby providing early feedback on speed-path requirements for an upcoming design. Our initial results suggest that the approach can scale to real-world RTL designs. We are working on more subtle speed-path issues, in particular constraints that require functional invariants in their derivation. We are also extending the tool by with support for compositional verification of speed-path constraints. We are also looking for ways to integrate ATPG techniques with our constraint generation framework to generate directed tests to cover specific speed-path conditions.

References

[1] N. Maheshwari and S. Sapatnekar, *Timing Analysis and Optimization of Sequential Circuits*. Kluwer Academic Publishers, 1999.

[2] W. A. Hunt Jr. and S. Swords, "Use of the E Language," in *Hardware Design and Functional Languages (HFL 2009)*, A. K. Martin and J. W. O'Leary, Eds., Mar. 2009.

[3] W. A. Hunt, Jr. and S. Swords, "Centaur Technology Media Unit Verification," in *Proceedings of the* 21*st International Conference on Computer-Aided Verification (CAV 2009)*, ser. LNCS, A. Boujjani and O. Maler, Eds., vol. 5643. Springer-Verlag, July 2009, pp. 353–367.

[4] W. A. Hunt, Jr., S. Swords, J. Davis, and A. Slobodova, "Use of Formal Verification at Centaur Technology," in *Design and Verification of Microprocessor Systems for High-Assurance Applications*, D. S. Hardin, Ed. Springer, 2010, pp. 65–88.

[5] B. Dutertre and L. de Moura, "A Fast Linear-Arithmetic Solver for DPLL(T)," in *Proceedings of the* 18*th International Conference on Computer-Aided Verification (CAV 2006)*, ser. LNCS, T. Ball and R. B. Jones, Eds., vol. 4144. Springer, 2006, pp. 81–94.