# Synergies Between Delay Test and Post-silicon Speed Path Validation: A Tutorial Introduction

Sandip Ray[1] and Arani Sinha[2]

[1]ECE Department, University of Florida, Gainesville, FL 32611. USA. sandip@ece.ufl.edu
[2]Intel Corporation, Hillsboro, OR 97124. USA. arani.sinha@intel.com

***Abstract*—The goal of speed path validation is to identify frequency limiting paths in a fabricated IC. It is a complex and expensive activity, requiring significant manual expertise. This paper provides a tutorial overview of speed path validation, focusing primarily on the state of the practice and its limitations. This paper also discusses delay test and discusses synergies between the two disciplines.**

***Index Terms*—Silicon debug, Timing errors, Frequency limiting paths, Delay Test**

## I. Introduction

Post-silicon validation makes use of fabricated (pre-production) silicon implementation of an integrated circuit (IC) as the platform to run tests. Post-silicon validation enables exploration of many design characteristics that cannot be exercised in pre-silicon (*e.g.*, RTL, netlist, or FPGA) validation. In particular, the silicon runs at target clock speed, while a cycle-accurate simulator is about $10^6$ times slower. Consequently, post-silicon tests can explore *deep* design states and exercise the design on realistic workloads. Furthermore, post-silicon validation can explore physical and electrical characteristics of the design, effect of temperature and physical stress, etc. Post-silicon validation remains one of the most critical and most expensive components in the modern IC validation flow.

Several papers have been published recently on aspects of post-silicon validation, including a number of surveys and tutorials [9], [8], [13]. However, literature has primarily focused on post-silicon *functional validation*, *i.e.*, the use of the silicon to identify, isolate, and diagnose design errors. Of course the scope of post-silicon validation is much larger: in addition to functional validation, one performs *compatibility validation* (*i.e.*, whether the hardware works with the plethora of peripherals, operating systems, and applications), *electrical validation* (*i.e.*, ensuring that the electrical characteristics such as power draw, voltage droop, etc. are within margin, even under noise), marginality validation (*i.e.*, behavior of the design under high clock speed), validation of the design under high thermal and physical stress, and many others.

In this paper, we provide a tutorial introduction to circuit marginality validation (CMV) activity in post-silicon validation. Commonly referred to as *speed path validation*, the goal is to ensure that the silicon runs at a fast enough speed under a variety of operating conditions. Speed path validation is a complex activity, involving a variety of sophisticated instruments together with subtle use of on-chip debug and testing architecture. We discuss the challenges involved in this activity, current practice and its limitations, and recent research in this area. While we discuss some research solutions, the focus of the paper is on industrial practice.

One of our goals is to situate speed path validation in the context of traditional delay test. Speed path validation relies on re-purposing Design-For-Test (DFT) hooks for identifying the failing paths. Similarities, differences, and synergies between delay test and speed-path validation will be covered.

## II. What is Speed Path Validation and Why Do We need It?

A speed path is a frequency-limiting critical path which affects the performance of a chip. A speed path that violates timing constraints during post-silicon validation is called a *failing speed path*. The goal of speed path validation is to identify failing speed paths, and determine the reason for the failure (*e.g.*, a slow transistor). Even if there is no failure, speed path validation is used to find and reduce speed bottlenecks in the circuit; this enables subsequent spins of the silicon to run at higher clock speed and sold as faster products.

Obviously, a significant effort is put in pre-silicon static timing analysis (STA) to create reasonably accurate measurement of circuit timing [7]. Nevertheless, a number of failing speed paths escape to post-silicon. In particular, STA often uses simplified delay models, which can result in mis-correlation between STA and post-silicon timing behavior [15]. Furthermore, STA generally cannot account for logical interactions between signals. For instance, a long path in a circuit may not be activated because of a subtle functional invariant. For these reasons, there is often discrepancy between the estimated timing from STA and actual timing measurements for silicon. Kaiss [2] estimates about 5% of chip frequency is achieved by identifying and fixing frequency-limiting paths through post-silicon speed path validation.

## III. Delay Test

Delay testing has traditionally been done for screening defects and marginalities that prevent a fabricated die from operating at targeted speed [3]. Many categories of test patterns generated for testing for excessive delay for a circuit can be used in speed path validation.

Delay failures in a design can be classified as gross delay failures or small delay failures, and delay test fault models are based on these assumptions. Gross delay failure assumption is

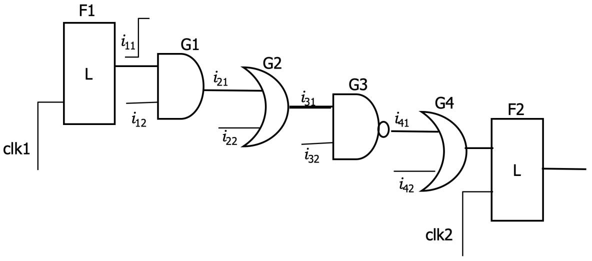

Fig. 1. A Logical Path Example

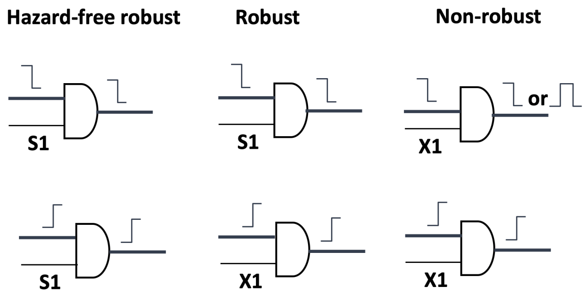

Fig. 2. Categories of Path Delay test

incorporated in the development of the transition delay fault model. This model assumes that the delay size on a node is high so that the fault can be propagated to a scan flop independent of the delays on the paths that are used to excite or propagate the fault effect. Small delay fault assumption has led to the development of multiple fault models. Two principal fault models are (1) small delay defect fault model or the timing aware fault model [5], and (2) path delay fault model. Timing aware fault models use knowledge of gate delays in the design to guide the excitation and propagation to and from the node which has the delay fault. Timing aware fault model assumption can be used to create test patterns which excite longer delays in the design relative to transition delay fault model. The path delay fault is discussed in more details below.

A physical path can be defined a sequence of contiguous gates from a launching flop to the capturing flop. For every physical path, we can define two logical paths: one that starts with a rising transition on the launching flop and another that starts with a falling transition on the launching flop. We refer to inputs that are on the logical path of interest as *on-path inputs*; others are referred to as *side inputs*. This is illustrated in Fig. 1. $P$ is a flop-to-flop physical path; $LP$ is a logical path that starts with a rising transition on $P$; on-path inputs are $i_{11}$, $i_{21}$, $i_{31}$, $i_{41}$; and side inputs are inputs of path gates not on P, *i.e.*, $i_{12}$, $i_{22}$, $i_{32}$, $i_{42}$.

Path delay tests can be classified in different categories based on sensitization at the side inputs. These categories are (i) hazard-free robust, (ii) robust, and (iii) non-robust. The conditions are shown in Fig. 2 for an AND gate. In this figure, S1 implies a static 1 value and X can be either 0 or 1. Similar conditions can be derived for all Boolean gates. Robustness is an important property of path tests as they guarantee that delay on adjacent paths, ones through side inputs of the path of interest, do not prevent detection of any potential defect on the path. If a path test fails to meet any of the conditions outlined here, then the path may only be a functionally sensitizable path, *i.e.*, it may propagate the delay effect as a glitch.

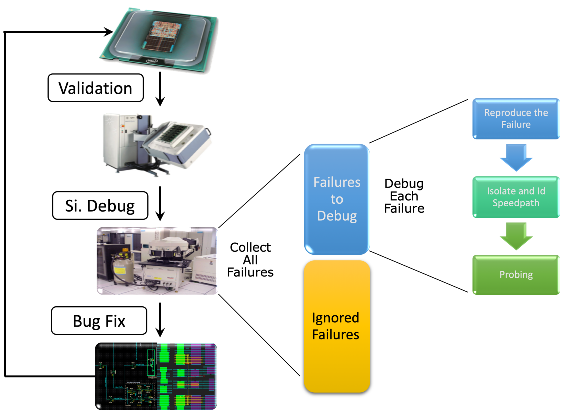

Fig. 3. Speed Path Validation Flow

Delay tests require at least two vectors $v_1$ and $v_2$. Vector $v_1$ sets up the condition for test, and the response of the circuit needs to settle down before $v_2$ is applied. After $v_2$ is applied, the response of the circuit is captured at the operational frequency of the design. There are different methods of application of delay test. The two most important ones are the launch-of-capture and launch-of-shift methods.

Path tests can also be derived from functional tests by extracting relevant vectors from a functional testbench that can excite delays on paths of interest in a design. This requires a detailed understanding of the design and manual test development.

## IV. Speed Path Activity Overview

It goes without saying that isolating speed paths and diagnosing their root cause is a challenging operation. A modern IC can include billions of transistors; furthermore, tests can be between $100,000$ and $1M$ cycles long and any of the cycles can entail activity of a slow transistor. Consequently, isolating a speed path can appear a bit like finding a needle in the haystack. Nevertheless, unlike other post-silicon validation activities such as functional validation or compatibility validation, speed path validation follows a relatively structured process. Fig. 3 provides a high-level overview of the overall flow. The process of debugging a speed failure starts by applying test patterns until an error is detected. The activity is performed on dedicated testers, and the test patterns are targeted to stress both the logical combinations of gates and transistors, and voltage droop, capacitive and inductive noise, and other electrical characteristics (Section V). The process is applied on dedicated testers, although some "system level" tests are employed with specialized validation boards. A significant amount of on-chip instrumentation is employed to assist in the debugging process (Section VII). The fix of such a failure typically involves modifying the circuit either by replacing a cell/gate with a faster (or slower) one, or by performing design retiming operations.

## V. Tests in Speed Path validation

Obviously, the efficacy of any post-silicon validation activity critically depends on the tests employed. For speed path validation, tests applied at the tester include both structural tests (*e.g.*, targeting stuck-at faults) and functional tests [1], [11]. Functional tests include pre-silicon functional validation tests created to exercise specific design features or tests known to have been effective at detecting electrical issues in previous generations of the circuit [10]. For processor cores these tests can include tests from random instruction generators: the goal is to exercise the various micro-architectural features in ways not conceived by human. These tests of course are not specifically targeted to stress the timing and marginal behaviors of the current chip. Tests to cover specific speed paths during debug are written for a small set of serious issues by human experts. In addition, longer system level tests are employed with the IC installed on validation board, that are closer to the eventual application run by the customer.

Note that test generation as well as the process of debugging from the tests can be expensive and challenging. First, automated structural and functional tests are not very effective in finding marginality issues, since they are not created to target marginal behaviors. Marginal behaviors can be created by electrical phenomena such as voltage droop and crosstalk. Such electrical behavior can also be aggravated by process variation. Such circuit behavior depends on workload and can be intermittent i.e. they may not be reproducible reliably. It is impossible to simulate during pre-silicon validation methods the combination of parameters that cause marginalities induced by a combination of electrical and process parameters. Second, writing manual tests when necessary for isolating specific speed paths during debug is extremely resource-intensive. Third, while system level tests represent realistic workloads and correspondingly can illustrate frequency limiting functionality when the chip is deployed, debugging the issues exhibited in these tests is difficult and unreliable due to non-determinism in system execution. Note that it would be helpful to make the failures reproducible in a tester, since the tester has significant more debug "hooks" or instrumentation than a validation board. On the other hand, long tests cannot be effectively used in a tester given the limited tester memory.

## VI. Debugging a Failure

Given the results of the tests, the debugging process entails identifying and isolating failing conditions. A typical approach is to use a 2-dimensional shmoo plot of voltage ($V_{dd}$) with frequency. Fig. 4 shows a representative plot. The overall idea is to identify the region (voltage and frequency combinations) that correspond to the passing tests. A failure obtained in that region would be a candidate for a failing speed path.

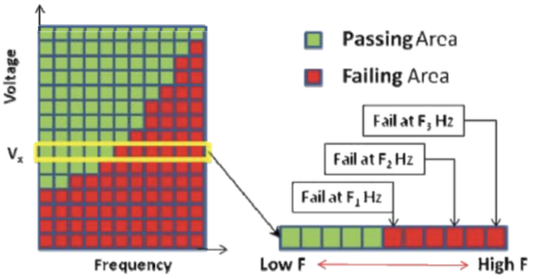

Fig. 4. Typical Shmoo Plot in Speed Path Validation. The red region represents that $V_{dd}$ and frequency combination that corresponds to failing tests, and the green region represents the region for passing tests.

Once we see a failure in the shmoo plot, we must isolate the failure and diagnose its root cause. First, we need to make the speed path reliably reproducible. This is non-trivial because of the non-determinism involved in post-silicon validation: the same test that caused failure once can pass when repeated. Furthermore, two failing speed paths may be excited by the same test. The goal then is to fine-tune the parameters (*e.g.*, voltage, frequency, temperature, etc.) with a wide failure band so that the failure is reproducible. Once we have a reproducible failure, the goal of the exploration is to narrow down the problem by answering the following two questions:

- **Spatial root causing:** Where in the die is the slow location? Is this a transistor device or an interconnect?
- **Temporal root causing:** Where in the execution has the slowness been encountered?

## VII. Debugging Aids

As with any debug methodology, speed path diagnosis is a highly manual process. Nevertheless, there are infrastructures to aid with this activity, as described below.

### A. On-Chip Instrumentation

One of the fundamental approaches to assist debug is re-purposing the available Design-for-Test (DfT) infrastructure available on-chip. Additionally some instrumentation is introduced specifically for speed path debug.

*Scan:* A critical DfT instrumentation available in virtually any chip is scan. The key idea of re-purposing scan for speed path validation is to identify and scan out the *critical cycle* determined as follows. For each of the failing tests, the frequency is relaxed to just let the test pass. The critical cycle then is the clock cycle where the failure is latched internally. We can then record the first scanout failures after the critical cycle, to facilitate spatial root causing. For a fully scanned design, a speed path is a flop to flop path and the critical cycle is also the scanout cycle. However, note that if the design has only partial scan then there may be several cycles between the critical cycle and the scanout cycle.

*On-die Clock Shrink:* This is an interesting debug feature available in some ICs particularly when the design does not include full scan. It was introduced in the Pentium 4 microprocessor [4], and provides the capability of shrinking and expanding a specific clock cycle. Additionally the clock skew can be adjusted between clock domains. In case of partial scan, this architecture can help reduce the discrepancy between critical cycle and scanout cycle as follows. For each of the failing tests, at the frequency in which the test just passes (see above), the clock can be progressively and sequentially stretched from the failure point, to identify the critical cycle. Furthermore, clock skews are adjusted for the critical cycle. If shrinking a clock edge causes an initially failing test to pass then the driving latches of the failing path can be narrowed to that clock region; correspondingly, if a shrinking causes an initial passing test to fail then the load or receiver latches can be narrowed to the clock region.

*Process monitor:* A process monitor is used to determine the process skew of the fabricated die. A die can belong to fast, nominal, or slow skew of the process. The process skew is important information for speed path debug. A process monitor is typically a ring of buffers and inverters whose delay can be recorded. By characterizing data from process monitors sprinkled around the die, one can infer the process skew.

*Voltage droop monitor:* Another debug aid is a voltage droop monitor. A voltage droop can make a path slower than the targeted delay, and the monitor helps understand if a speed path is caused by unexpected voltage droop [12].

### B. Failure Analysis: Instruments vs CAD

While on-chip instrumentation are helpful, they are not sufficient for effective debugging of industrial speed path failures. In particular,due to the large number of gates dominated by each clock domain, one need to narrow the list of failing source candidates into a smaller group of logic gates. Following are representative technologies to enable effective failure analysis.

- *Laser Voltage Probe (LVP)* is used to monitor values, arrival times and transition times at internal signals.
- *Laser Assisted Device Alteration (LADA)* is used to speed up or slow down a device to check if it can pass under altered timing conditions [14].
- *Focused Ion Beam (FIB)* is used to "circuit-edit" a device (*e.g.*, cut and add wires, remove components, etc.) and check if it passes with altered connectivity

The failure analysis infrastructures above are obviously heavy-weight. They are expensive (often costing more than million dollars per equipment), require human operators with special expertise, and human creativity to determine how exactly to exploit the sophistication of the machines for effective debugging. A debugging process might take hours up to weeks *per speed failure*. Since an IC can have hundreds to thousands of speed path failures, the process can take months. To ameliorate the problem, there have been recent approaches to develop light-weight, automated CAD flows. The goal is to reduce the failure analysis instruments but still exploit the on-chip instrumentation discussed in Section VII-A. McLaughlin *et al.* [6] present an automated debug mechanism, borrowing from fault simulation techniques by using delay testing for temporal root causing. In particular, faults are seeded on the receiver latch candidates after identification using clock shrink as discussed in Section VII-A, and a path tracing procedure analogous to critical path tracing is used to identify sensitized paths. If multiple paths are found then they are further analyzed through pre-silicon timing analysis as well as probing solutions. Recently, Kaiss [2] extended such procedure with symbolic analysis based on SAT-based formal reasoning. The idea is to perform symbolic pre-image computation across circuit elements using SAT, to localize a path from a failure to the source flip-flop. Kaiss reports that the tool found the same speed paths as LADA for a next-generation industrial microprocessor but was much faster than LADA-based debug which often took weeks. Such results reflect the promise of CAD solutions. Nevertheless, as of this writing, CAD approaches are still nascent in industrial practice.

## VIII. Conclusion

Speed path validation is a complex and challenging component of post-silicon validation in a modern IC. In this paper we have provided a summary of this activity, focusing on current industrial practice and limitations. We hope that the tutorial description will help de-mystify post-silicon validation in general and speed path validation in particular.

## References

[1] S. Gurumurthy, S. Vasudevan, and J. A. Abraham. Automated mapping of precomputed module-level test sequences to processor instructions. In *ITC*, pages 294–303, 2005.

[2] D. Kaiss and J. Kalechstain. Post-silicon Timing Diagnosis Made Simple Using Formal Technology. In *FMCAD*, pages 131–138, 2014.

[3] A. Krstic and K.-T. Cheng. *Delay Fault Testing for VLSI Circuits*. Springer, 1998.

[4] N. A. Kurd, J. S. Barkarullah, R. O. Dizon, T. D. Fletcher, and P. D. Madland. A multigigahertz clocking scheme for the Pentium 4 microprocessor. *IEEE Journal of Solid-State Circuits*, 36(11):1647–1653, 2001.

[5] X. Lin, K. Tsai, C. Wang, M. Kassab, J. Rajski, T. Kobayashi, R. Klingenberg, Y. Sato, S. Hamada, and T. Aikyo. Timing-Aware ATPG for High Quality At-speed Testing of Small Delay Defects. In *Asian Test Symposium*, pages 139–146, 2006.

[6] R. McLaughlin, S. Venkataraman, and K. Lim. Automated Debug of Speed Path Failures Using Functional Tests. In *VTS*, 2009.

[7] T. M. McWilliam. Verification of timing constraints on large digital systems. In *DAC*, pages 139–147, 1980.

[8] P. Mishra and F. Farahmandi, editors. *Post-silicon Validation and Debug*. Springer, 2018.

[9] P. Mishra, R. Morad, A. Ziv, and S. Ray. Post-silicon validation in the soc era: A tutorial introduction. *IEEE Design Test*, 34(3):68–92, June 2017.

[10] S. Natarajan, A. Krishnamachari, E. Chiprout, and R. Galivanche. Path coverage based functional test generation for processor marginality validation. In *ITC*, pages 1–9, 2010.

[11] S. Natarajan, S. Patil, and S. Chakravarty. Path Delay Fault Simulation on Large Industrial Designs. In *24th IEEE VLSI Test Symposium*, pages 16–22, 2006.

[12] P. Pant and J. Zelman. Understanding power supply droop during at-speed scan testing. In *VLSI Test Symposium*, 2009.

[13] K. Rahmani, S. Ray, and P. Mishra. Postsilicon trace signal selection using machine learning techniques. *IEEE Trans. Very Large Scale Integr. Syst.*, 25(2):570–580, February 2017.

[14] R. Rowlette and T. Elies. Critical Timing Analysis in Microprocessors Using Near-IR Laser-Assisted Device Alteration (LADA). In *International Test Conference*, pages 264–273, 2003.

[15] A. Shah, R. Nayyar, and A. Sinha. Silicon-Proven Timing Signoff Methodology Using Hazard-Free Robust Path Delay Tests. *IEEE Design & Test*, 37(4):7–13, 2020.