2019 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION INDIA

Our Experience of Glitches at Clock Trees, CDC Paths and Reset Trees

Jebin Mohandas, Intel Technology India (p) Ltd., Bengaluru, India. (jebin.mohandas@intel.com)

***Abstract*— Implementation of a design from RTL to netlist introduces glitches on clock trees, Clock domain crossings (CDC) paths and reset trees. CDC if not verified appropriately, cause chip re-spins if detected late in the design cycle. To prevent this, implementing a rigorous approach to detect these potential problems early in the design cycle at the RTL level, and post synthesis at the gate level is mandatory. For low power designs, clock gating and reset gating are used extensively. When IPs from multiple sources are integrated, glitch-causing combinational logic can be introduced unintentionally. At the same time, CDC paths with multiplexers or combinational logic are prone to glitch defects introduced by synthesis. These glitches, pulses of very short duration, may be captured by a register when crossing clock domains, causing a functional failure. In this paper, we review the cause of these challenges and introduce a methodology to overcome these difficulties.**

.

Keywords—glitch; clock integrity; reset integrity;

I. INTRODUCTION

CDC verification ensures that signals that cross asynchronous clock boundaries arrive complete and coherent in the receiving domain. A flip-flop whose data or asynchronous reset input changes too close to the clock edge will enter a transitional indeterminate state whose duration is probabilistic. This transient indeterminate state is known as a metastable state. Many clock-boundary synchronization schemes exist to ensure that simple one-bit signals, all the way to multi-bit data busses, and even signaling protocols, can be transferred accurately despite this metastable behavior. The proper design of these synchronization mechanisms to ensure that the indeterminate state is not sampled more frequently than system failure rates allow are well documented and out of the scope of this paper.

Flip-flop storage devices also have requirements for minimum active pulse-widths for clock and reset signals that, if violated, will result in metastable behavior. Synchronization designs do not typically ensure correct behavior when this condition occurs as these types of issues are not limited to clock domain crossings.

Traditional CDC verification evaluates the design at a Register Transfer Level (RTL), representing registers (flip-flops) and the functional logic between them. At this level, clocks and resets can be easily specified, constrained and even inferred. Formal-based CDC algorithms exhaustively evaluate the RTL and identify CDCs, infer synchronization schemes and ensure that the design has integrity. However, traditional CDC verification also tends to focus specifically on the clock domain crossing and the synchronization schemes themselves, and does not focus on the integrity of the clock or reset signal specifically.

Our designs tend to be extremely complex. Aggressive power management techniques are employed that impact clocks and resets via gating functions as well as via power domain isolation techniques. IPs designed for generic re-use were designed with different methodologies in mind. Not to mention about the legacy IP's which are not verified for glitch. Performance and latency requirements cause us to minimize the number of clock cycles required to move data. With many clocks in the design, clock domain crossings designed for minimal latency, and a highly-complex functional environment, the challenges of CDC at the RTL level are significant enough. But generally speaking, RTL-based CDC analysis will not find all issues that can arise from this complexity. Why? RTL is an abstraction and isn't built in silicon. There are several steps to production from RTL. Designs are synthesized into gates, modified to add power reduction or isolation schemes, and to add test circuitry. Levels of abstraction are removed, exposing potential abstraction modeling inaccuracies. The implementation process is constrained to minimize power and area, and maximize performance. In the end, the design targeted to be built doesn't entirely match the one verified in RTL even though all possible efforts are made.

Yet RTL analysis is the fastest, most efficient and easiest to debug. The objective therefore is to identify constructs at the RTL level that are susceptible to implementation-based issues later so that these issues can be found and fixed in RTL. We run checks to identify issues in clock integrity, reset integrity in RTL. The glitches in data transitioning through clock domain crossings can be unearthed on running netlist CDC.

A. *Clock Integrity*

We specifically analyze clocks looking for functional clock manipulations which may later be implemented by gates due to synthesis that can create glitches. Examples of these types of issues are,

- Combination feedback loops in clock logic created by the interconnection of unrelated IP, clock control and gating circuitry, and the like.
- Clock enable signals generated by combinational logic and not connected to a latch and therefore not held during the active phase of a clock
- Forked and re-converged clock signals through combination logic ultimately driving a storage element.
- Clock and reset pins of a storage element being driven by the same signal
- Clocks driven by inappropriate combinational gates known to generate glitches such as XOR or XNOR gates.

B. *CDC Data Glitches*

When implementing combinational logic functions with logic gates, the optimization algorithms of synthesis tools can create circuits that are capable of generating glitches on outputs as inputs change. This is due to the nature of different delays through logic gates and their inter-gate routes. This effect is normally not relevant in synchronous designs. However, in the case of asynchronous domain crossings, the data or reset pin on the receiving clock domain's flip-flop may change due to the glitch too close to the active edge of the receiving clock. This creates a new CDC violation in a paths that was error-free in RTL. These types of implementation issues demand that CDC analysis be done on the netlist prior to tapeout to ensure that the design will work as expected when built.

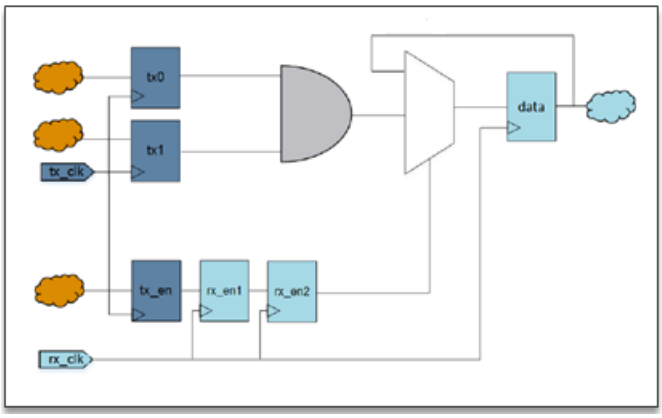

Figure 1. RTL logic before optimization

One such example is below in Figure 1 below, showing the functional RTL representation. During synthesis, the synthesis tool optimizes for power, performance and area constraints as shown below in Figure 2 by replacing the multiplexer function with an AND-OR structure and incorporating the functional AND from the RTL source into the optimized AND-OR network as well for minimal latency. This implementation is functionally identical but has induced a CDC violation due to timing-based race conditions through this new structure, in a way that, should the flip-flops tx1 and rx_en2. Our methodology uses CDC verification to identify these issues post-implementation, to ensure working silicon.

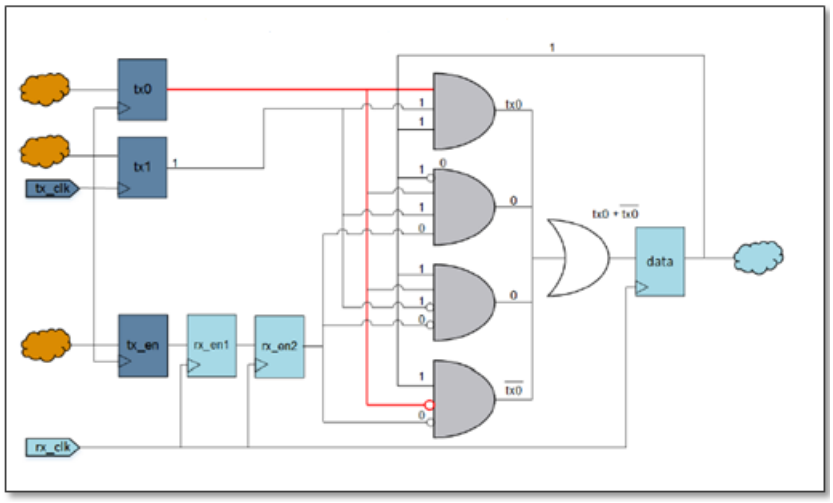

Figure 2. RTL logic after optimization

C. Reset Integrity

We specifically analyze resets looking for functional reset manipulations which may later be implemented by gates due to synthesis that can create glitches. Examples of these types of issues are,

- Resets driven by inappropriate combinational gates known to generate glitches such as XOR or XNOR gates.
- Unexpected latches or three-state logic within a reset tree.
- Combinational logic in the reset path that may generate glitches as in Figure 3 below
- Forked and re-converged reset signals through combinational logic paths that may create glitches.

II. CDC ANALYSIS - THE HIERARCHICAL WAY

Curently CDC Analysis is done bottoms up. Hierarchical models are created for IP's. Subsystems (SS) consume these IP models and generate SS models. These SS models are consumed at the SOC level run. The hierarchical approach followed is captured in Figure 3. Sometimes the IP also Sub IP's. Then the corresponding models will be consumed by IP's. Many of the IP's are of the shelf and are created in different geographies.

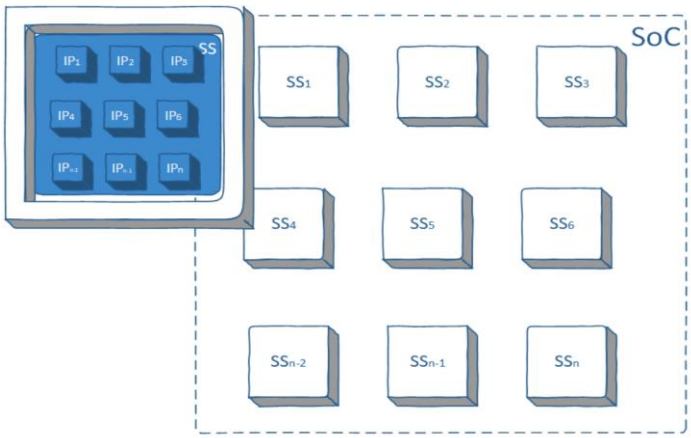

Figure 3. Bottoms UP CDC Analysis

III. METHODOLOGY TO UNEARTH CLOCK AND RESET INTEGRITY

A. Glitch Analysis at SOC level.

The methodology we adopted was to run the glitch analysis once the CDC analysis is clean at the SOC level. We flattened out the RTL without abstracts. There results were analyzed by the SOC teams and for fixing the IP teams was contacted. Subsequently the improved IP has to be plugged in to SOC.

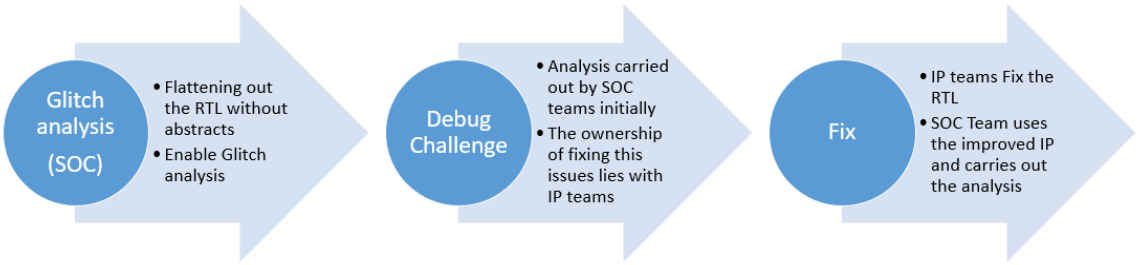

Figure 4. Glitch Analysis at SOC level.

B. Glitch analysis at SOC level(Bottoms up)

Better approach would be to carry out the glitch analysis at the IP level itself. Then the effort to go back to the IP team for fixes can be avoided. This

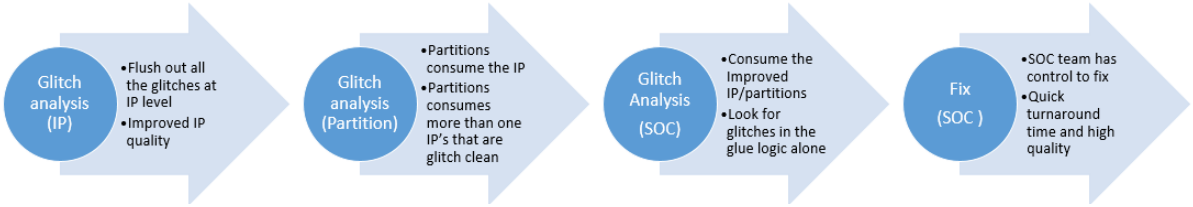

Figure 5. Glitch Analysis at SOC level (Bottoms Up)

IV. FINDINGS

Before you begin to format your paper, first write and save the content as a separate text file. Keep your text and graphic files separate until after the text has been formatted and styled. Do not use hard tabs, and limit use of hard returns to only one return at the end of a paragraph. Do not add any kind of pagination anywhere in the paper. Do not number text heads-the template will do that for you.

Finally, complete content and organizational editing before formatting. Please take note of the following items when proofreading spelling and grammar:

A. *Reset cobo Glitch*

Glitch created by combo logic on reset paths is a potential re-spin candidate. These can be captured early in the design cycle as in Figure 6 and Figure 7. Both these depict combo logic on reset paths.

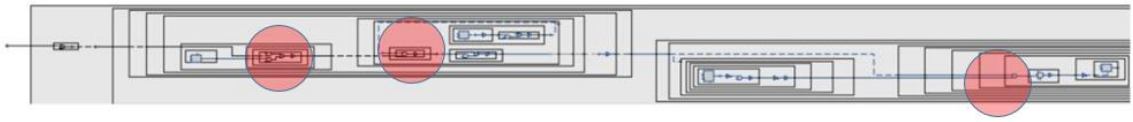

Figure 6. Glitch source in reset paths.

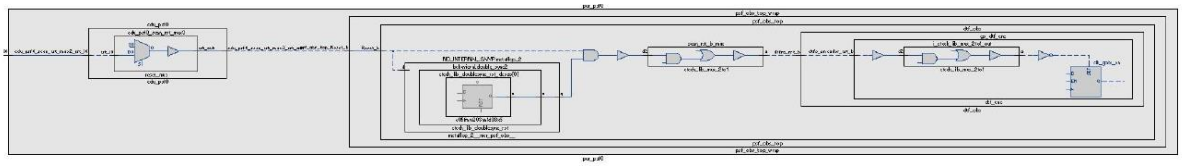

Figure 7. Glitch source in reset paths.

B. *Clock reset race*

The presence of race condition between clock and reset is flagged in Figure 8.The clock tree and reset trees can be validated .

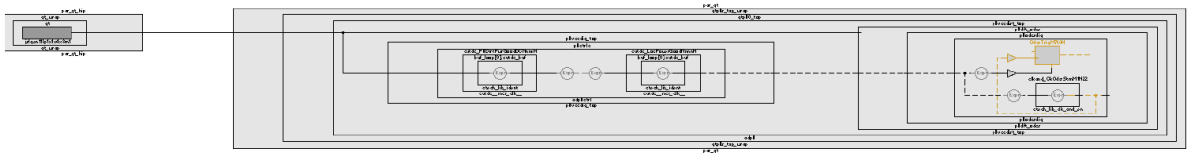

Figure 8.Clock reset race

C. *Unexpected Gate*

The presence of combo gates in clock path is flagged in Figure 9. This helps to make sure the clock tree is intact.

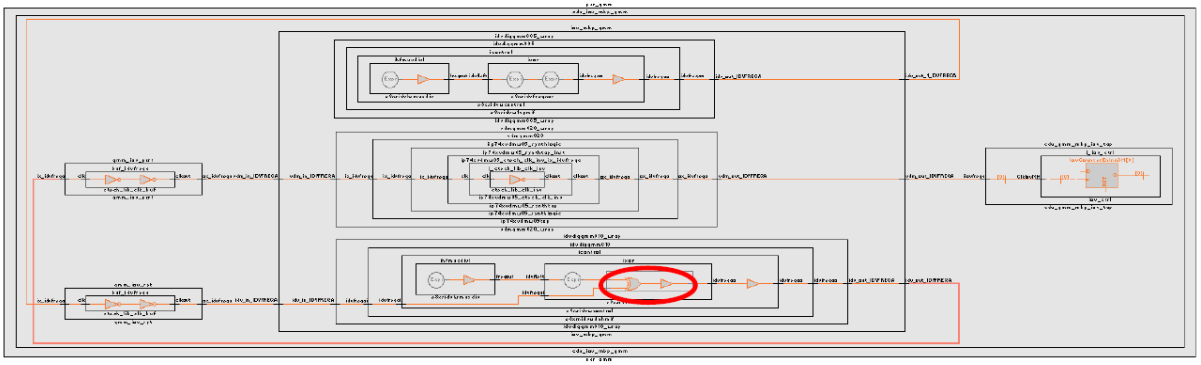

Figure 9. Combo logic in clock path

V RESULTS

With these enhanced checks for Clock Integrity, Reset Integrity we can make sure the design is free of the glitches in clock, CDC paths and in resets. Unearthing glitch earlier in the design cycle will help to speed up RTL quality and avoid costly iterations later in the design cycle. The ROI of carrying out these additional checks is justified by the confidence the design teams gains after running the checks. Table 1 captures the runtime and memory consumed for one of the SOC analysis. CDC Data glitches at the implementation level is yet to be tried.

Table I. Runtime and Memory

	Efforts		
	Checks	***Run Time***	***Memory***
1.	Clock Tree Checks	2287 sec	13 GB
2.	Reset Tree Checks	2111 sec	13 GB

REFERENCES

[1] Jackie Hsiung, et al., “Preventing Chip-Killing Glitches on CDC Paths with Automated Formal Analysis,” DVCon 2018
[2] Anwesha Choudhury, Ashish Hari, “Accelerating CDC Verification Closure on Gate-Level Designs”, DVCon 2017
[3] Ping Yeung, “Five Steps to Quality CDC Verification,” Mentor Graphics, advanced verification white paper.