2021 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE OCTOBER 26-27, 2021

Detection of glitch-prone clock and reset propagation with automated formal analysis

Kaushal Shah (Kaushal_Shah@mentor.com),
Sulabh Kumar Khare (Sulabh-kumar_khare@mentor.com)
DVT, Siemens EDA, Noida

***Abstract*—Modern SoCs employ various complex reset and clock architectures to handle ever increasing demand for timing and power efficiency. Advanced clock and reset architectures can lead to complex circuitry on the clock and reset paths. The logic on the clock and reset paths is prone to propagation of glitches. This paper presents a unique method based on identification of unique combinational expressions on the clock and reset path and use automatic formal analysis to zero-in on real clock and reset glitch paths that can make your chip unpredictable if not addressed early. Appropriate debug aids based on combinational expression analysis of the clock or asynchronous reset path logic are presented to quickly zoom to the logic that's responsible for introducing the glitch.**

***Keywords*—Clock, Reset, Glitch, RTL**

I. INTRODUCTION

With the increase in the complexity of the System-on-Chips (SOC) with every new generation, the number of clocks and resets also increases to meet the requirement of high performance and low power. However, this leads to various issues related to glitch on the clock and reset paths that may occur in the system with the use of multiple asynchronous clocks and reset signals. The traditional verification techniques like Functional Simulation, Timing analysis, and complex formal techniques cannot guarantee silicon without the domain crossing issues and glitch on these clocks and reset paths. The gate-level simulation will also play a major role in unearthing glitches that would have crept in during synthesis. Even though issues get flagged, major challenges are huge run times and difficulty in debug at netlist. The present designs usually contain multiple IPs from different sources, and once they integrate to form a complete design, glitch prone logic can be introduced unintentionally and may cause the designs to fail.

Glitches are unwanted spikes in signals that can propagate through the combinational logic. At times, signals display an in-correct intermediate value before reaching the final, steady state value; this results in a glitch. Fig 1 shows glitches in combinational logic. A signal may temporarily change its value while it is supposed to remain static at a logic 1 or 0 value, or it may oscillate while changing value from 0 to 1 or 1 to 0 before reaching the final value.

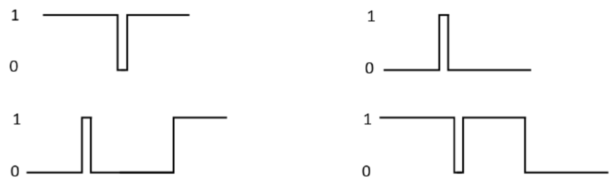

Fig 1: Examples of glitches on combinational logic

In combinational logic, a glitch can occur if there are multiple paths with opposite polarity converging at an AND or, OR gate and the path delays from the source of the signal to the converging point do not match. Glitches can occasionally cause functional errors that may lead to chip failures. Typically, glitches can be introduced during synthesis due to glitch prone implementation of combinational logic. Fig 2 shows a simplified synthesis flow; it takes an RTL description of the design in an HDL language like Verilog or VHDL with design constraints and converts it to a gate-level netlist. The synthesis process performs some optimizations to satisfy the design constraints and meet power, area, and frequency requirements. During the synthesis process, it may implement basic RTL elements like multiplexers in a way which is prone to glitches. Moreover, the low power artifacts from UPF can interfere with the clock and reset and introduce glitch prone logic.

These glitches can cause sources of worry across clock domains. Generally, any combinational logic may be having short-lived glitches; this effect usually is 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. The design may, therefore, receive a glitch as a pulse, causing a functional failure. In a similar manner, clock glitch can create unnecessary clocking events and functional problems and reset glitch can cause the design to reset and cause unexpected behavior. Digital Systems with glitch increase the number of signal transitions. As a result, the power consumption of design increases.

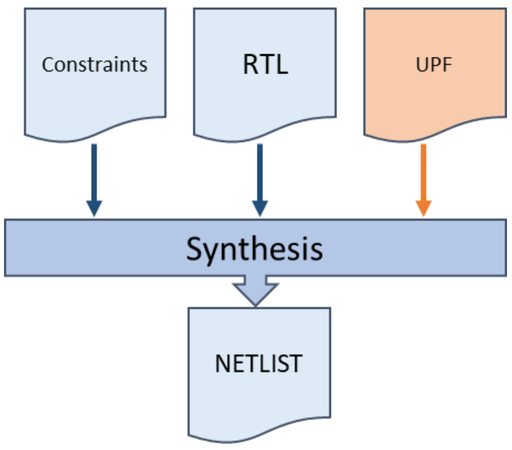

Fig 2: Synthesis may produce glitch prone logic when it converts RTL to a gate-level netlist

II. GLITCH PROBLEM ON CLOCK OR RESET PATH

As an example of how a glitch can be introduced in the asynchronous reset path, consider the logic in Fig 3. This logic can be generated after the synthesis of multiplexer logic in the RTL. A glitch can be introduced and propagated to reset path whenever an asynchronous reset or asynchronous set pins are driven by converging signals with differing inversion counts. Consider the logic in Fig 3, which is an example of an asynchronous reset path with converging signals. The combinational logic in the schematic has the potential to glitch under certain conditions. For example, when rst_a and rst_b (active low resets) are inactive, which means holding value 1, and sel transitions from 1 to 0, there is a possibility of delay through the inverter and the output may momentarily become 0. So even when the resets are inactive, the converged signal can glitch when the sel value changes and may put the driver register to de-asserted state. Now, if the clock's sampling edge comes during the reset glitch, the unwanted value is sampled and propagated which may result in functional failure.

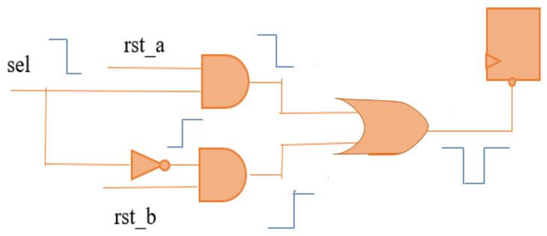

Fig 3. static-0 glitch introduced on the asynchronous reset path

III. CHALLENGES

With the increasing complexity of SoCs, multiple and independent clocks are essential in the design. The design specifications require system level multiplexing of some of these clocks before they are sent to actual IP. Also, to save power, clock gating cells are inserted in clock paths. While implementing these multiplexing and gating cells, a designer may unintentionally design logic that can lead to glitches. A glitch on a clock signal exposes a chip (or a section of a chip) to asynchronous behavior. A glitch-prone clock signal driving a flip-flop, memory, or latch may result in incorrect, unstable data.

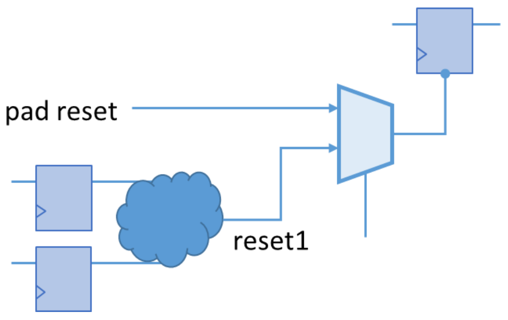

Fig 4: Multiplexing and combinational logic on the reset path

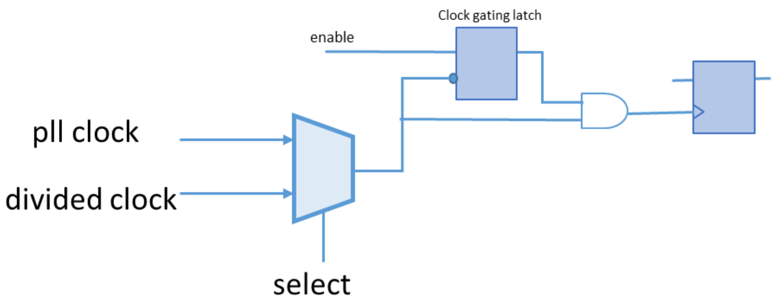

Fig 5: Multiplexing and gating on the clock path

Existing method of verification rely on RTL level structural checks which just check presence of logic on clock and reset paths. Analysis of the results of these structural checks is manual and often the logic is considered safe and checks are waived since structurally it looks good and no problem is seen. Also, RTL simulation does not show any issues. However, during synthesis, the logic may be converted to a logic which generates glitches under certain conditions. The generated glitch may propagate to different parts of the design causing unexpected behaviors.

Design and verification engineers have attempted to identify the presence of combinational logic capable of inducing the glitches in signals traversing to asynchronous reset paths and clock paths. Each such method has some limitations in terms of the accuracy it provides, and effort required to identify the true glitch paths. Below are some examples of methods tried by engineers and the challenges faced.

A. *Gate-level simulation to capture glitch*

For each of the clock and reset paths with combinational logic, the receiver value can be sampled in a gate-level simulation. Sampled values then can be used to identify the presence of glitch logic in these paths. Due to the asynchronous nature of signal, many simulators have difficulty sampling the signal paths at the times when glitches may be present. Even if the simulator catches a glitch path and a waveform is present, it is difficult to manually identify the source of the glitch path.

B. *Assertion to identify glitch*

Some verification engineers have elected to write assertions to check if the simulators are sampling the signal paths at the correct time, but the results of sampling checks are often not accurately captured by the simulator. Assertions also have some semantic limitations when it comes to asynchronous clocks, and all simulators may not handle these kinds of assertions. So, writing assertions may itself becomes a difficult task.

C. *Static checks at the netlist level*

Verification engineers have also performed static checks on the circuit design to identify portions of the circuit that may be susceptible to inducing glitches in signals traversing to clock or reset pins. These static checks typically include divergence and convergence check, combinational glitchy logic in clock or reset path, clock gating enable signal not driven by latch check, asynchronous source merges with different clock domain. Flagging every clock or reset path that contains combinational logic in its path for manual inspection by the verification engineers is a mammoth effort. While manual inspection can identify combinational logic capable of inducing signal glitches, the procedure is error-prone and time-consuming

IV. SOLUTION: GLITCH DETECTION ON CLOCK AND RESET PATH

The glitch minimization techniques in the digital circuits are well studied in the literature and numerous approaches have been proposed like reducing switching activity, gate freezing minimizes power dissipation by eliminating glitching, gate sizing technique used for path balancing and multi-threshold technique used in path balancing for glitch reduction. Although there are numerous approaches for glitch minimization, the detection of glitch becomes more critical.

The methodology proposed in this paper is illustrated in Fig. 5. This flow shown can be divided into different sections as below.

A. Design Setup

For design setup, gate-level design files and initial constraints should be provided. The initial constraint file consists of information about top-level clocks, resets, design modes, constants, etc.

B. Detect clock and reset paths

Identification of clock and reset networks (trees) and marking all registers in the design with corresponding clock or reset domain.

C. Static analysis related to Glitch

This stage is the main process in the glitch detection. It innovatively identifies the unique glitch-prone combinational expressions common to many clock paths or reset paths. The next step is to do a comprehensive expression analysis of the combinational logic tree to identify potential glitch candidates that can cause glitch to propagate.

D. Formal analysis

In this step, the glitch list is further pruned to identify the scenarios under which the glitch really propagates. To accomplish this, we utilize the formal engines to verify the glitch propagation conditions and conclusively give counter examples of scenarios under which the glitch will propagate. The counter example helps the verification engineer convince the IP team of the exact scenario under which the glitch can propagate and cause a silicon failure.

E. Debug

To analyze the glitchy logic, its source and where actually the glitch occurs (converging point) can be visualized using the schematic in the GUI.

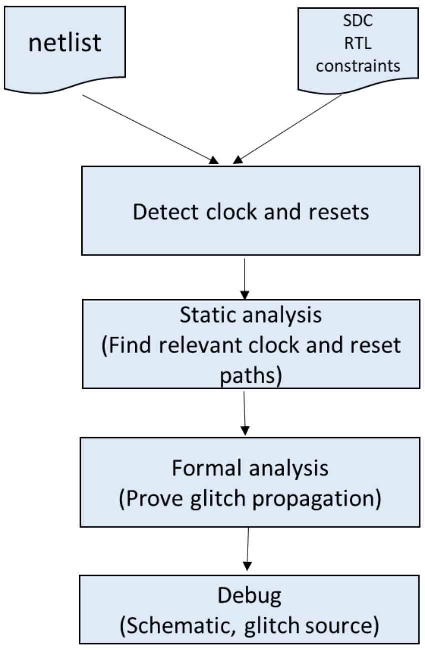
netlist
SDC
RTL
constraints
Detect clock and resets
Static analysis
(Find relevant clock and reset paths)
Formal analysis
(Prove glitch propagation)
Debug
(Schematic, glitch source)

Fig 6: Formal analysis-based glitch propagation detection

V. RESULTS

The proposed methodology has been implemented on couple of SoCs to detect glitches on clock and reset paths. For reference, lets name the designs as Design 1 and Design 2. Both the designs were setup for clock and reset tree detection and top-level constraints were applied. Design 1 is of the order with 10 million gates and Design 2 with 2 million gates. Table I and Table II below shows the result of the clock and reset glitch analysis on both the designs respectively.

Table I. Clock glitch analysis result

Design name	Number of clock domains	Number of glitch signals	Number of clock paths impacted
Design 1	107	68	8437
Design 2	22	3	1

Fig 7 below shows the schematic of clock glitch on clock select latch path.

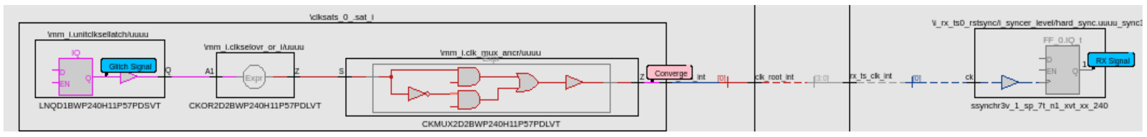
\clksats_0_.sat_i
\mm_i.unitclksellatch/uuuu
Glitch Signal
LNQD1BWP240H11P57PDSVT
\mm_i.clkselovr_or_i/uuuu
Expr
CKOR2D2BWP240H11P57PDLVT
\mm_i.clk_mux_ancr/uuuu
CKMUX2D2BWP240H11P57PDLVT
Converge
clk_root_int
rx_ts_clk_int
\i_rx_ts0_rstsync/i_syncer_level/hard_sync.uuuu_sync
FF_0.IQ_t
RX Signal
ssynchr3v_1_sp_7t_n1_xvt_xx_240

Fig 7. Schematic for clock glitch

Table II. Reset glitch analysis result

Design name	Number of reset domains	Number of glitch signals	Number of reset paths impacted
Design 1	33	2	714
Design 2	43	6	4

Fig 8 below shows the schematic of reset glitch.

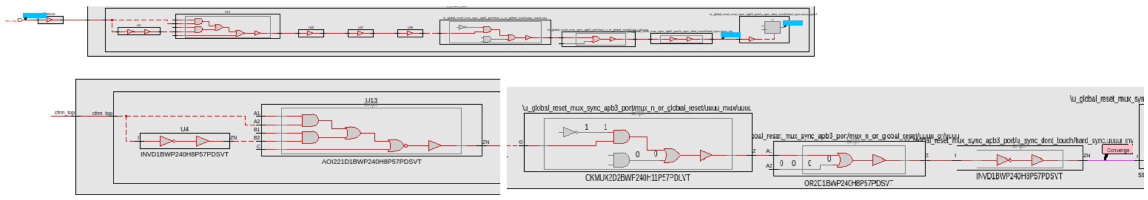

Fig 8. Schematic for reset glitch

The data where the number of glitch signals are more than the paths impacted is since multiple glitches can occur at the same path with different sources. Below are the snapshots captured for a clock glitch on the clock select latch path and a reset glitch from the glitch analysis results. The schematic shows the glitch source, converge point and the receiving register.

VI. SUMMARY

The proposed methodology intelligently identifies the unique paths which are glitch-prone and perform an automatic formal based glitch analysis to improve quality of results for glitch detection. This in result ensures that the glitches reported are with minimum noise and gives debug aids on details for the glitch source and converging point. This methodology identifies a few relevant proven glitches out of millions of clocks and reset paths and helps users focus on verifying them effectively. The generation of counter examples under which glitches would propagate makes it obvious that reported issues are genuine and pose serious threats to our next SoC. Advanced debug techniques that highlight the glitch introduction point help in fixing the glitches efficiently and reduce verification cycles and re-spin.

VII. REFRENCES

1. Jackie Hsiung, Ashish Hari, Sulabh-Kumar Khare, “Preventing Chip-Killing Glitches on CDC Paths with Automated Formal Analysis”, Mediatek, DVCon 2018.
2. Addressing the Challenges of Reset Verification in SoC Designs Chris Kwok, Priya Viswanathan, Ping Yeung, DVCon 2015
3. Jian-Hua Yan, Ping Yeung, Stewart Li, Sulabh-Kumar Khare, “Preventing Glitch Nightmares on CDC Paths: The Three Witches”
4. Clifford E. Cummings, Don Mills, Steve Golson, “Asynchronous & Synchronous Reset Design Techniques - Part Deux”
5. Prateek Gupta, Priyank Garg, “Configurable dividers for SOC- and block-level clocking – EDN”