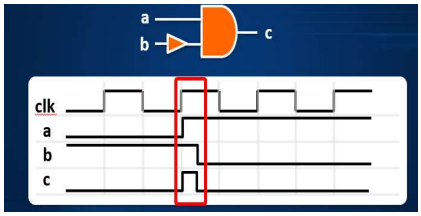
A Holistic Overview on Preventive & Corrective Action To Handle Glitches

Rohit Kumar Sinha

Parimal Das

What are Functional Glitches

- ✓Perennial problem in digital designs and functional glitches occur frequently in combination logic but are not simulated accurately.
- ✓ Difference between the event-driven simulation engine and the physical implementation of the RTL.
- ✓ Glitches in the clock domain crossing paths or glitches in the synchronous paths with exception such as set_false_path, set_multicycle_paths, set_max_delaypaths, path margins etc can also lead to Silicon failures.



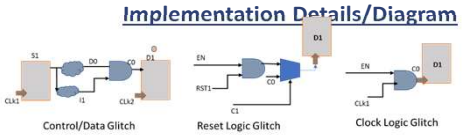
Proposed Methodology/Advantages

1. The data on the bus must always be stable before the synchronized edge-triggered qualifier signal goes high in the receive domain.

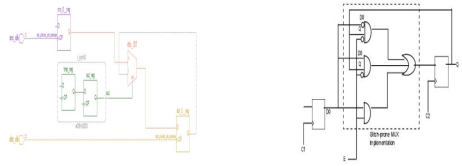
2. The data on the bus must remain stable after the qualifier has gone high, long enough to meet the hold-time requirements in the receive domain

3. Due to the uncertainty of how many receive clock cycles it will take for the qualifier to go through the synchronizer, there is a further requirement that the data on the CDC bus must be stable for at least two/three receive-domain clock cycles.

Implementation Details/Diagram



Front End – Async CDC path as no timing check. For Synchronous paths, there is no glitch issues as design is converged to correct setup/hold checks.	Front End – Spyglass, Caliber Reviews Solution – Control Logic	Front End – Spyglass, Caliber Reviews Solution- Control Logic



Implementation Details/Flow Chart

Proposed Solution: D-Mux or Recirculation Mux

FE

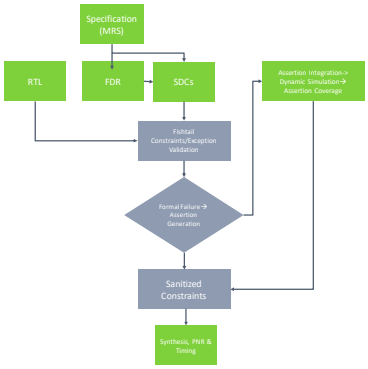
RTL Modification	Implementation Flow
• Add the pragma "infer_mux_override" • Modify the RTL with ctech instantiation • **Limitations** – Glitches inside IPs exists in CDC Hierarchical Flow Methodology	Spyglass-> VC-CDC-> Fusion Compiler where Muxes will be preserved •Map to Gtech Mux"Syn_Gen" •**Limitations** – Same as Option 1

BE

Mandatory - Static Timing Check	Paranoia for qualifier decomposition
• max_delay <= 1 RX (depending on Ctech stages) Clock Cycle (might not meet timing for long async path) • FE collateral to publish list of qualifiers and grey encoders	• Qualifier decomposition can lead to glitch. Paranoia to check non-ctech on qualifier paths. • If decomposed – to check if qualifier is blocking the async crossing

Results Table

Glitch Validation for the Synchronous Paths with Exceptions



There are multiple categories of paths reported and only glitch prone paths are identified. In cross domain crossing paths there are five ways of constraining the timing paths:

- asynchronous paths
- max_delay constraining paths
- mcp exception
- path margin based paths
- single cycle paths

Conclusion

- Unintentional glitches can alter the desired functionality of the circuit and design could fail
- For the first time both in front end and backend automation is developed to have glitch free design for clock crossing paths.
- Glitches are difficult to catch in the design flows because of its unpredictable nature
- Finding glitches and relying on back-end flows are very late and expensive, hence defining the design practices to prevent them is the only scalable solution
- In this paper a successful method of avoiding and prevention of glitches in clock cross domain paths is achieved and proposed for future use

REFERENCES

References

- Rohit Kumar Sinha(s), Left Shift Mechanism to Mitigate Gate Level Asynchronous Design Challenges, DVCon-India, 2021