

# Trivializing Chip Reset Timing: A Novel Technique for SoC Backend Closure

Aradhana Kumari,  
STMicroelectronics, Crolles, France  
[aradhana.kumari01@st.com](mailto:aradhana.kumari01@st.com)

Sara Bourouga,  
STMicroelectronics, Paris, France  
[sara.bourouga@st.com](mailto:sara.bourouga@st.com)

**Abstract:** Global reset timing closure has become a significant implementation challenge in modern System-on-Chip (SoC) designs. As design complexity grows, reset signals must be distributed across large physical regions while satisfying recovery and removal timing requirements across all operating conditions. Conventional approaches rely on extensive reset-tree balancing, buffering, and iterative timing optimization, leading to increased routing congestion, area overhead, power consumption, and backend design effort. This paper presents a novel reset-closure methodology that eliminates recovery and removal timing constraints from backend optimization by delaying functional clock activation until reset propagation has completed throughout the design. Following place-and-route, the worst-case reset propagation delay is extracted and used to determine an N-cycle startup delay implemented through a synchronization and clock-gating mechanism. By guaranteeing that all sequential elements observe a stable reset state prior to clock activation, reset recovery and removal requirements are inherently satisfied. Experimental validation on a 20K-gate design implemented in 28-nm STMicroelectronics FD-SOI technology and operating at 2.4 GHz demonstrates complete elimination of recovery and removal violations. The proposed architecture converts reset timing closure from a physical-design optimization problem into a deterministic startup sequencing problem, thereby simplifying backend implementation and providing a scalable solution for future high-performance SoCs.

**Keywords:** Asynchronous reset, recovery timing, removal timing, reset-domain crossing (RDC), reset tree, physical design, backend closure, timing closure, system-on-chip (SoC).

## I. INTRODUCTION

The continued growth of System-on-Chip (SoC) complexity has transformed global reset implementation into a significant backend design challenge. Modern SoCs routinely contain hundreds of thousands of sequential elements distributed across multiple physical regions, clock domains, and functional subsystems. As a result, reset signals must be propagated across large portions of the chip while satisfying stringent recovery and removal timing requirements during timing signoff.

Figure 1 illustrates a typical global reset distribution architecture. An asynchronous reset signal (RESETN) is first synchronized and then distributed through a large fanout reset network to sequential elements throughout the design. As chip size increases, the reset distribution tree grows substantially in complexity, requiring additional buffering, routing resources, and timing optimization effort. Consequently, reset implementation

often becomes comparable in complexity to clock distribution, particularly in high-performance SoCs operating at gigahertz frequencies.

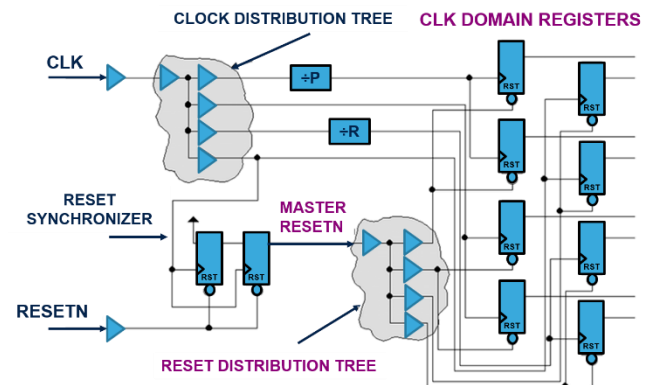


Figure.1. Conventional Global Reset Synchronization and Distribution Network

Unlike functional datapaths, reset networks contribute only during device initialization, power-on sequences, and exceptional recovery scenarios. Nevertheless, conventional implementation methodologies require recovery and removal timing closure across all reset endpoints. This process must be performed concurrently with setup and hold timing closure, increasing routing congestion, implementation runtime, power consumption, and engineering effort. In large designs, even small reset skews can generate thousands of recovery and removal violations, leading to iterative backend optimization cycles and prolonged design closure schedules.

Existing approaches primarily focus on balancing reset trees through buffering, routing optimization, and timing analysis to satisfy recovery and removal constraints. While effective, these techniques treat reset timing closure as an implementation problem that becomes increasingly difficult as design size and operating frequency continue to scale.

This paper presents a fundamentally different approach. Rather than optimizing reset arrival times, the proposed architecture guarantees that functional clock activity begins only after reset propagation has safely completed throughout the design. By introducing a controlled clock-activation mechanism based on the worst-case reset propagation delay, the dependency between reset de-assertion and clock arrival is effectively removed. As a result, recovery and removal timing requirements become inherently satisfied, transforming reset timing closure from an iterative backend optimization challenge into a deterministic architectural design step. The main contributions of this work are summarized as follows:

- A novel timing-independent reset architecture that decouples reset propagation from functional clock activation.
- A deterministic methodology for eliminating recovery and removal timing closure requirements.
- A scalable implementation framework suitable for high-frequency and large-scale SoCs.
- Experimental validation on a 20K-gate 2.4 GHz design implemented in 28 nm FD-SOI technology, demonstrating complete elimination of reset timing violations.

## II. CONVENTIONAL RESET SYNCHRONIZATION CHALLENGES

### A. Conventional Reset Architecture

Asynchronous resets are widely used in SoC designs because they provide immediate initialization of sequential logic during power-up, fault recovery, and system restart events. However, direct distribution of an asynchronous reset signal across a clock domain may lead to metastability during reset release. To avoid this issue, reset synchronizers are commonly employed to generate a synchronized reset signal before distribution to destination registers.

Traditional reset synchronizers ensure safe de-assertion of global reset signals across clock domains but add significant routing and timing complexity.

The image in Figure 2 illustrates the behavior of a reset synchronizer during both assertion and de-assertion phases of the reset signal, along with associated timing waveforms. The circuit consists of two D flip-flops in series, forming a reset synchronizer. The RESETN signal is fed asynchronously to both flip-flops, allowing immediate reset assertion without waiting for a clock edge. This ensures that all fanout registers connected to RST\_OUT are promptly reset.

During reset de-assertion, the signal first reaches the synchronizer flops. The first flip-flop captures the de-asserted value (logic high) on the next clock edge, and the second flip-flop propagates it to RST\_OUT on the subsequent clock edge. This makes the de-assertion synchronous, and hence, recovery and removal timing checks must be met from the second stage of the synchronizer to all domain registers.

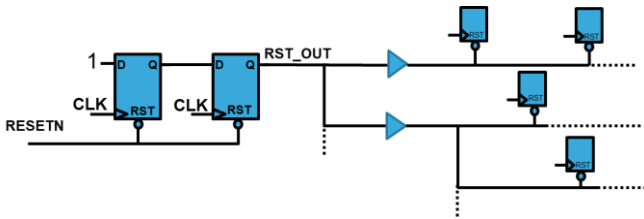


Figure.2. Conventional Two-Stage Reset Synchronizer

### B. Recovery and Removal Timing

For asynchronous reset systems, reset assertion is generally non-critical because registers are forced immediately into a known state. Reset de-assertion, however, must satisfy recovery and removal timing constraints relative to the destination clock.

Recovery time defines the minimum interval required between reset release and the next active clock edge, while removal time defines the minimum interval during which reset must remain stable after a clock transition. Violation of

either constraint can result in metastability, incorrect state initialization, unpredictable startup behavior, and potential functional failures.

Figure 3 illustrates the timing diagram that shows how removal violations can occur if RST\_OUT transitions at the same time as a clock edge, violating the required timing margin. To avoid this, RST\_OUT must be balanced across all flops in the domain, which demands meticulous timing analysis and routing effort. This process is repetitive, area-consuming, and power-intensive, especially in large designs, making reset synchronization a critical and non-trivial aspect of digital design.

Therefore, timing signoff methodologies require recovery and removal checks to be satisfied across all registers and all process, voltage, and temperature (PVT) operating corners.

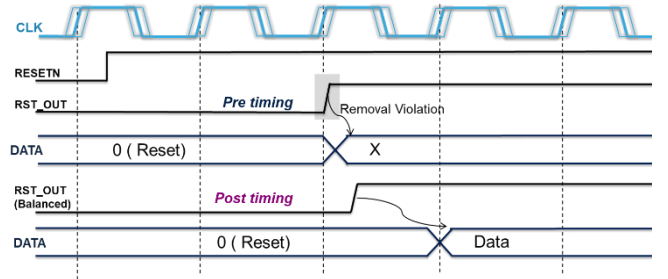


Figure.3. Recovery and Removal Timing Requirements During Reset De-assertion

### C. Scaling Challenges in Modern SoCs

The complexity of reset timing closure increases rapidly as SoCs grow in size and operating frequency. A global reset signal may drive hundreds of thousands of registers distributed across large physical distances, resulting in substantial reset-tree depth and routing complexity.

From a physical implementation perspective, reset network complexity can be approximated as:

$$Complexity_{Reset} \propto Fanout \times PhysicalDistance$$

As fanout and routing distance increase, additional buffering is required to maintain acceptable signal integrity and propagation delay. These buffers consume area and power while competing with clock and datapath routing resources. Consequently, reset-tree optimization may introduce routing congestion and prolong timing closure cycles.

The challenge becomes even more severe in GHz-frequency designs, where clock periods are extremely small and recovery/removal timing margins shrink accordingly. Even modest reset skew can generate thousands of timing violations requiring iterative optimization and repeated implementation runs.

### D. Limitations of Conventional Approaches

Existing implementation methodologies primarily address reset timing closure through reset-tree balancing, buffer insertion, routing optimization, and repeated timing analysis. While these techniques can eventually achieve signoff compliance, they treat reset timing closure as a physical implementation problem whose complexity increases with design scale.

As a result, considerable backend effort is spent optimizing a network that is active only during system initialization and recovery events. This motivates the need for an alternative architectural methodology that eliminates the fundamental dependency between reset propagation delay and clock activation, thereby simplifying reset timing closure in large high-performance SoCs.

### III. PROPOSED TIMING-INDEPENDENT RESET ARCHITECTURE

The central concept of the proposed architecture is simple: Do not optimize reset timing. Instead, delay functional clock activation until reset propagation is guaranteed to complete throughout the design.

Figure 4 presents a tunable clock delay circuit, which builds upon the conventional reset synchronizer architecture but introduces a distinct functionality. In this design, the asynchronous reset signal RESETN is directly connected to the reset tree, ensuring immediate reset assertion across all flip-flops. Upon de-assertion of RESETN, the circuit initiates a delayed activation of the clock gating control signal RST\_INT, which becomes active only after a delay of approximately 4 to 5 clock cycles.

This delay is achieved by cascading multiple flip-flops, each clocked by the system clock (CLK), allowing the reset de-assertion to propagate sequentially through the chain. The delayed RST\_INT signal can then be used to safely enable downstream logic or clock gating, ensuring that all reset-dependent elements have stabilized before resuming normal operation. This approach enhances robustness in reset recovery and mitigates timing hazards associated with premature activation of gated clocks.

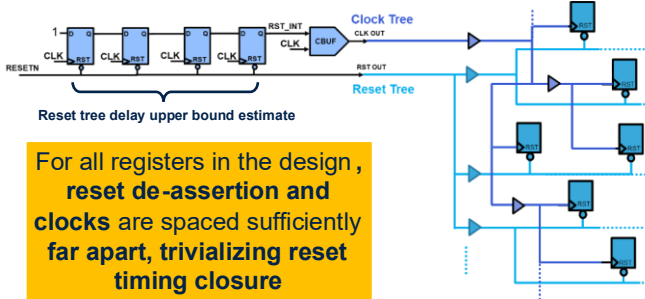


Figure.4. Proposed Reset-Aware Clock Activation Architecture

#### A. Implementation Methodology

The proposed implementation flow consists of four steps.

- **Step 1: Datapath Timing Closure**

Placement, routing, and timing closure are first completed while treating reset paths as false paths. This allows implementation tools to focus exclusively on performance-critical setup and hold timing requirements without expending effort on reset timing optimization.

- **Step 2: Worst-Case Reset Delay Extraction**

Following place-and-route, the maximum reset propagation delay from the reset source to the farthest register in the design is determined.

This delay is defined as:

$$T_{RESET(MAX)}$$

where  $T_{RESET(MAX)}$  represents the worst-case reset tree propagation delay under signoff conditions.

- **Step 3: Delay Quantization**

Express the reset delay as an integer multiple of the clock period:

$$N = \left\lceil \frac{T_{RESET(MAX)}}{T_{CLK}} \right\rceil$$

where:

- $T_{CLK}$  is the clock period.
- $N$  represents the reset delay upper-bound estimate expressed in clock cycles.

The ceiling operation guarantees that the calculated delay always exceeds the worst-case reset propagation delay

- **Step 4: Controlled Clock Activation**

An N-stage synchronizer combined with a clock-gating cell is used to delay activation of the functional clock.

The resulting output clock can be represented as:

$$CLK_{OUT} = Delay(CLK, N)$$

Only after  $N$  clock cycles have elapsed following reset release is the clock allowed to propagate to the design. This guarantees that reset propagation has completed across all sequential elements prior to functional operation.

Figure 5 illustrates the concept of reset delay upper-bound estimation and clock activation.

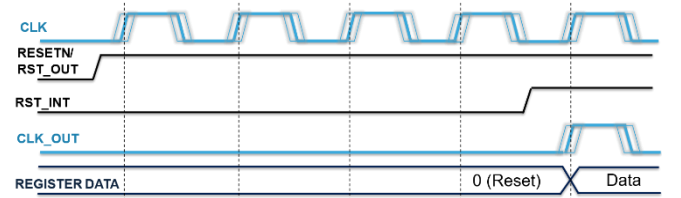


Figure.5. Timing Relationship Between Reset Propagation and Delayed Clock Activation

### IV. THEORETICAL JUSTIFICATION

For every register  $i$ ,

$$T_{RESET(i)} \leq T_{RESET(MAX)}$$

By construction:

$$N \times T_{CLK} \geq T_{RESET(MAX)}$$

Therefore:

$$N \times T_{CLK} > T_{RESET(i)}$$

for all registers.

As a result, reset de-assertion is guaranteed to precede clock arrival by at least:

$$Margin = (N \times T_{CLK}) - T_{RESET(i)}$$

Since the margin remains positive for all registers, recovery and removal constraints are inherently satisfied without dedicated reset-tree optimization.

## V. EXPERIMENTAL RESULTS

The proposed methodology was implemented on a 20K-gate high-speed design fabricated in 28 nm FDSOI technology and operating at 2.4 GHz.

Table I summarizes the measured timing results.

The results demonstrate complete elimination of both recovery and removal violations. More importantly, the approach eliminates the need for iterative reset-tree optimization activities commonly required during timing closure.

Because implementation complexity grows with design size, larger SoCs are expected to achieve proportionally larger benefits in congestion reduction, area savings, and engineering effort.

| Violation's Type | Figure of Merit  | Conventional Design Timing | Proposed New Technique |
|------------------|------------------|----------------------------|------------------------|
| Recovery         | Slack (ns)       | -0.67                      | +0.45                  |
|                  | Total Slack (ns) | -936.1                     | +3352                  |
|                  | Violating Paths  | 3427                       | 0                      |
| Removal          | Slack (ns)       | -0.08                      | +0.5                   |
|                  | Total Slack (ns) | -132.4                     | +2755                  |
|                  | Violating Paths  | 379                        | 0                      |

Table I. Recovery and Removal Timing Violations Before and After Applying the Proposed Methodology

## VI. CONCLUSION

This paper presented a novel architecture for trivializing reset timing closure in large-scale SoCs. By estimating the worst-case reset propagation delay and delaying clock activation through an N-stage synchronization mechanism, reset recovery and removal timing requirements become inherently satisfied.

Experimental validation on a 2.4 GHz 28 nm FDSOI design demonstrated complete removal of reset timing violations,

transforming a traditionally difficult implementation challenge into a deterministic and scalable methodology. In addition to reducing backend effort, the technique lowers routing congestion, area overhead, and implementation complexity, making it particularly attractive for next-generation high-performance SoCs operating at gigahertz frequencies.

## REFERENCES

- [1] J. Bhasker and R. Chadha, *Static Timing Analysis for Nanometer Designs*, Springer, 2009.
- [2] C. E. Cummings, D. Mills, and S. Golson, "Asynchronous and Synchronous Reset Design Techniques," SNUG Conference Proceedings, 2003.
- [3] N. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*, 4th Edition, Pearson, 2011.
- [4] S. Sapatnekar, *Timing Analysis and Optimization of Sequential Circuits*, Springer, 2004.
- [5] C. E. Cummings, "Synchronous Resets? Asynchronous Resets? I am So Confused! How Will I Ever Know Which to Use?," *SNUG San Jose*, 2002.
- [6] IEEE Standard for SystemVerilog Unified Hardware Design, Specification, and Verification Language, IEEE Std 1800-2023, IEEE, 2023.
- [7] IEEE Standard for Design and Verification of Low-Power Integrated Circuits, IEEE Std 1801-2024 (UPF), IEEE, 2024.
- [8] P. Alfke, "Metastability in FPGAs," *Xilinx Application Note XAPP094*, Xilinx Inc.
- [9] M. Litterick, "Pragmatic Simulation-Based Verification of Clock Domain Crossing Signals and Jitter Using SystemVerilog Assertions," *Verilab White Paper*.
- [10] Siemens EDA, *Reset Domain Crossing (RDC) Verification Methodology Guide*, 2024.
- [11] Synopsys, *Reset Domain Crossing Verification and Signoff Methodology*, Technical White Paper, 2024.
- [12] Cadence Design Systems, *Advanced Reset and CDC Verification for Large SoCs*, Technical Report, 2025.