NRFs Identification & Signoff with GLS Validation

Rohit Kumar Sinha

Rakesh Misra, Samir Nagesh Kulkarni

Problem Statement/Introduction

NRF Bugs

Design1> Design-A0 Si hangs due to scan logic in functional path
Yield impact 8% &1 man-year debug effort
Design2→ Si punit pwrgood not getting asserted due to NRF on scan_en pathYield impact 20% 5 man-months debug effort

NRF Definition

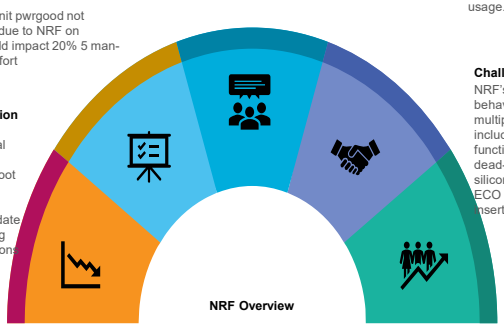
Flip-flops without a set/reset pin resulting output/q in undetermined state. Flops which can't be initialized in the HDL

Why NRF

Accounts for 20% Area Saving
Routing Congestion due to set/reset; Direct Positive impact on Area and Power with NRF usage.

Innvoation

Innovation is to identify the critical Non-resettable registers in the boot partitions in Intel Architecture. Secondly, to validate those NRFs using dynamic simulations on the netlist



NRF Overview

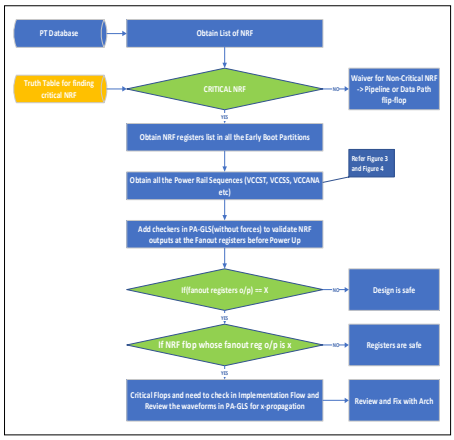
Challenges

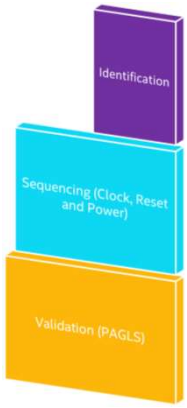
NRF's undeterminism behavior causes multiple silicon failure including few functional bugs or dead-on-arrival silicon. Mainly post-ECO and Scan-insertions

Proposed Methodology/Advantages

Domain	Item	Description
Design	SpyGlass Lint Rules	ResetFlop-ML and UnInitializedReset-ML rules
VAL	X-Prop Validation	Comprehensive x-prop verification at RTL
	GLS $deposit mechanism	NRF 0/1/random/no-value deposit in GLS for key control blocks such as punit, boot path (first step) Final step: minimum NRFs in SOC which have been reviewed and documented in HSDs
	Assertion methodology	Assertion based methodology to validate and verify small pieces of design for full functionality
DFT	Evil Validation	NRF non scan cells x-injection and observing critical signals

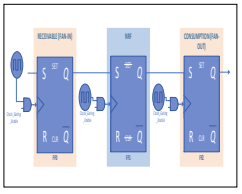
Implementation Details/Diagram





Implementation Details/Flow Chart

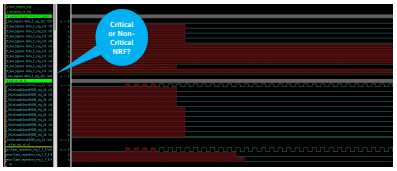
- Early detection of functional failures due to NRFs and left shift for NRF signoff at SD 0.8 / SD 1.0 milestone (SD – Structural Design Flow)
- Scales up for different Intel architectures along with different technology nodes or different foundries
- Nullifies the probability of functional failures due to NRFs
- Promotes better architecture/design optimization with the use of NRFs and its integration
- Engineering community can ramp-up quickly
- Optimizes the manual efforts to signoff the NRFs and sets a standard process



NRF Detection Strategy

Results Table

Program Name	Total NRFs	Critical NRFs	Signoff	Duration
SoC Design1	1255318	8000	YES @ SD1.0	2 Weeks
SoC Design2	1588147	78000	Analysis is under progress; SD1.0	



Impacted Area	Details
Speed of Execution	Table shows almost 0.7% are critical NRFs which were of the review and signoff for ADL-N program part.
Quality	The entire outcome of these techniques enabled Architecture and Design team to focus on critical NRFs and review them for signoff.
Integration correctness and Learnings	It is evident that having a NRF in the design is not an issue but what makes the difference is NRFs integrations

Conclusion

- PA-GLS Validation Coverage helps to identify critical NRF in boot partitions due to various logical addition and logic optimization
- Each partitions have high number of NRFs (>5000) leading to increased validation complexity. Total number of NRFs in client SoC is ~120K. Covering all combinations on NRF through Dynamic Simulation is a challenge.
- The validation of the non-resettable flops and all aspects of the verification flow enables a correct and complete integration and guarantees functionality in any SoC for 100% percent silicon success.
- X-prop enabled RTL validation uses RTL for simulations with all register elements initialized to 'X' value. Due to the way RTL construct used and coded, it is tough to create a true x-propagation behavior in RTL and PAGLS compliments x-prop feature.
- In the mentioned approach, we will start with all NRFs and run the simulations to see which flip-flops needs reset for correct functionality
- This method guarantees to generate critical list of NRFs that designers need to review and focus on those that can cause a failure in silicon, thus saving the efforts in design cycle and saving re-spins (Time & Money).

REFERENCES

References

- ADL-N Power-up Sequence: ADLN-Rail_sequencing.pptx (sharepoint.com)
- Rohit Kumar Sinha(s), Late Silicon Bug – Formal comes to Rescue, Design Automation Conference, 2020