Identifying Soft-Resets in RTL Design using RDC Flow

Megha Hansaliya
Sushovan Kunti

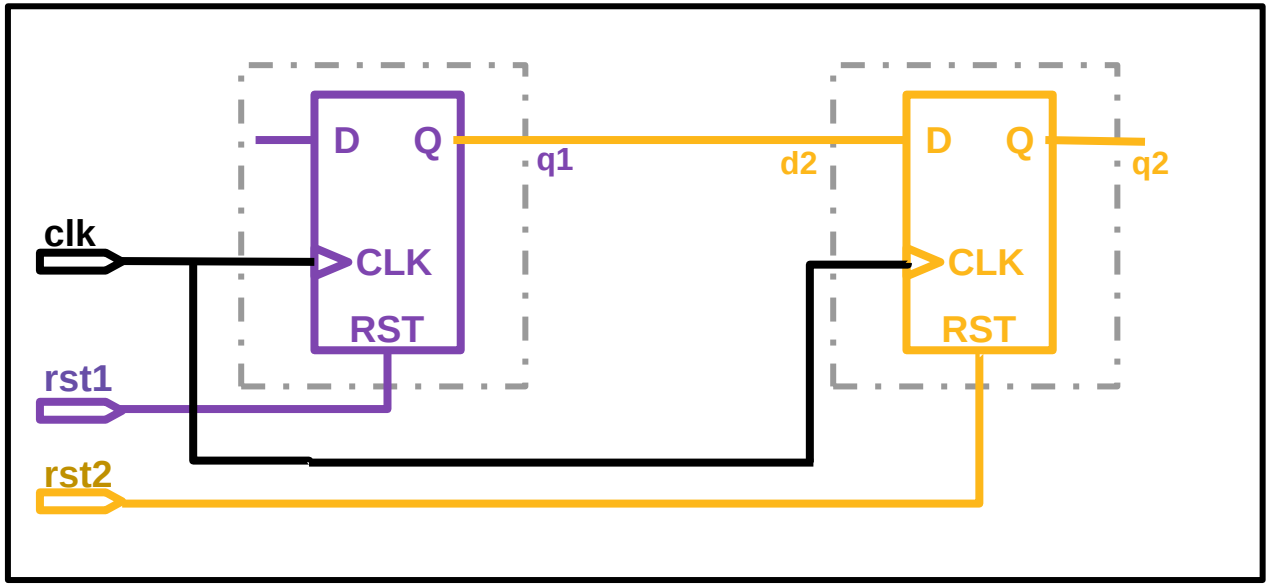
Google

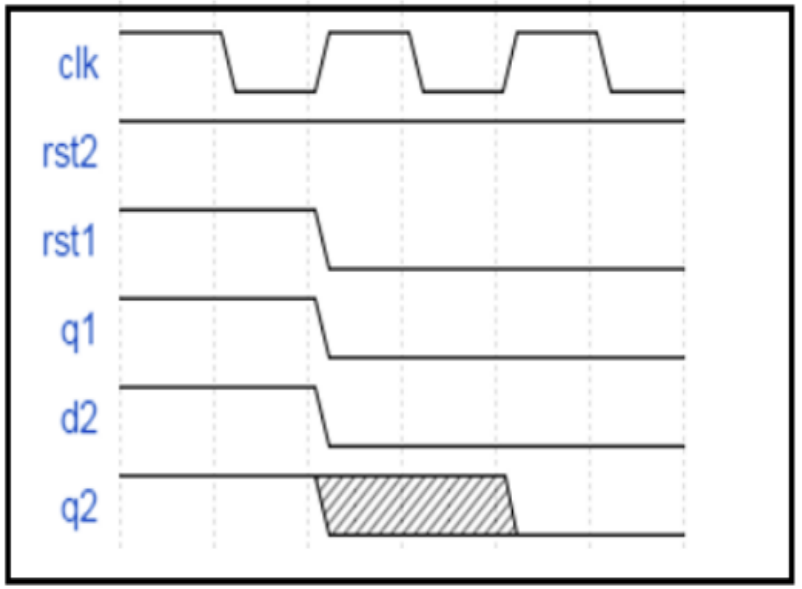
AGENDA

- ❑Introduction to RDC
- ❑Problem Statement
- ❑Methodology
- ❑Implementation
- ❑Challenges and Workaround
- ❑Results
- ❑Conclusions
- ❑Future Scope of Work

Introduction on RDC

Introduction to RDC

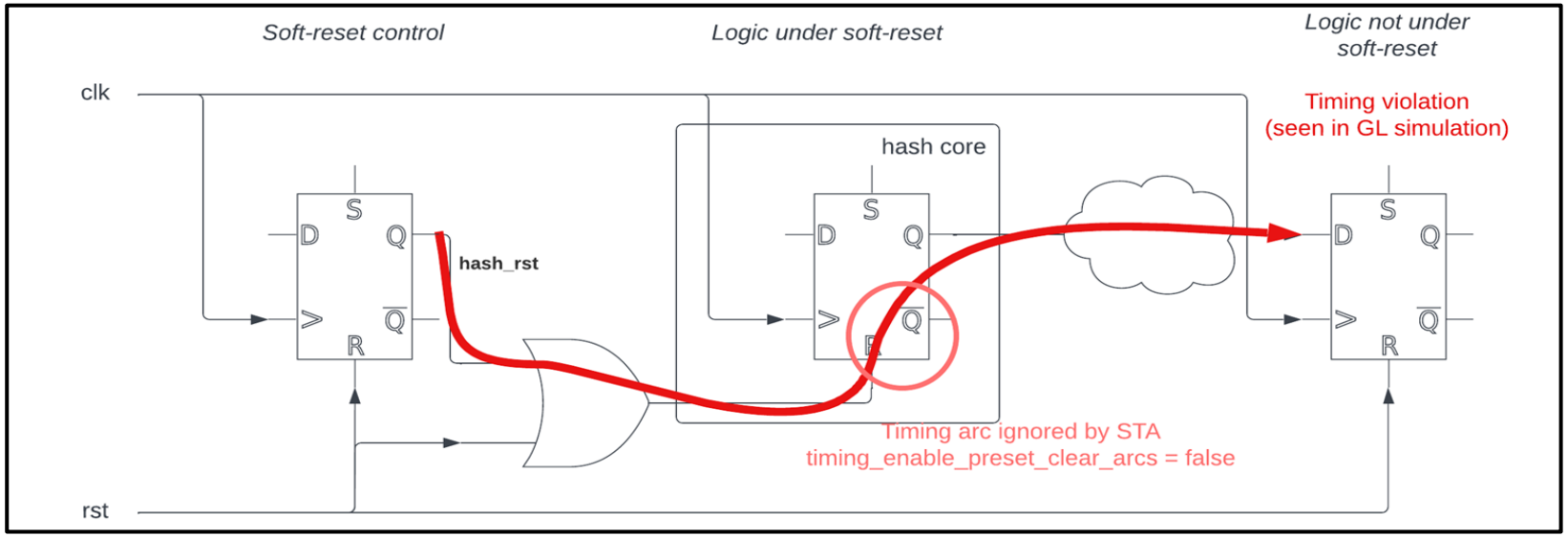




- Today's SoCs see **increasing reset domains and very complex reset architecture**
- **Multiple resets** are designed to cater to various requirements
- Part which asynchronously goes in reset state, **corrupts the other part not in reset state**
- Results into **metastability, glitch, and functional failure** leading to silicon failure
- In the above schematic, due to the asynchronous assertion of rst1, the d2 input may change within the setup or hold window of that flop, **leading to metastability**

Problem Statement

Problem Statement



- Synchronous reset coming from internal logic or F/Fs is **combined with global asynchronous resets**
- Together when they **drive the async reset port** of some internal logic it **causes metastability**
- As a result the **internal logic goes under soft-reset**
- **Difficult to find the exact point** where **reset-crossing** is happening between **software reset and asynchronous reset**
- Require **multiple iterations** of RTL Reset Constraints releases
- Industry-standard tools often lack direct visibility into reset crossings between soft-resets and asynchronous resets leads to **post-silicon bugs** in the chip

Methodology

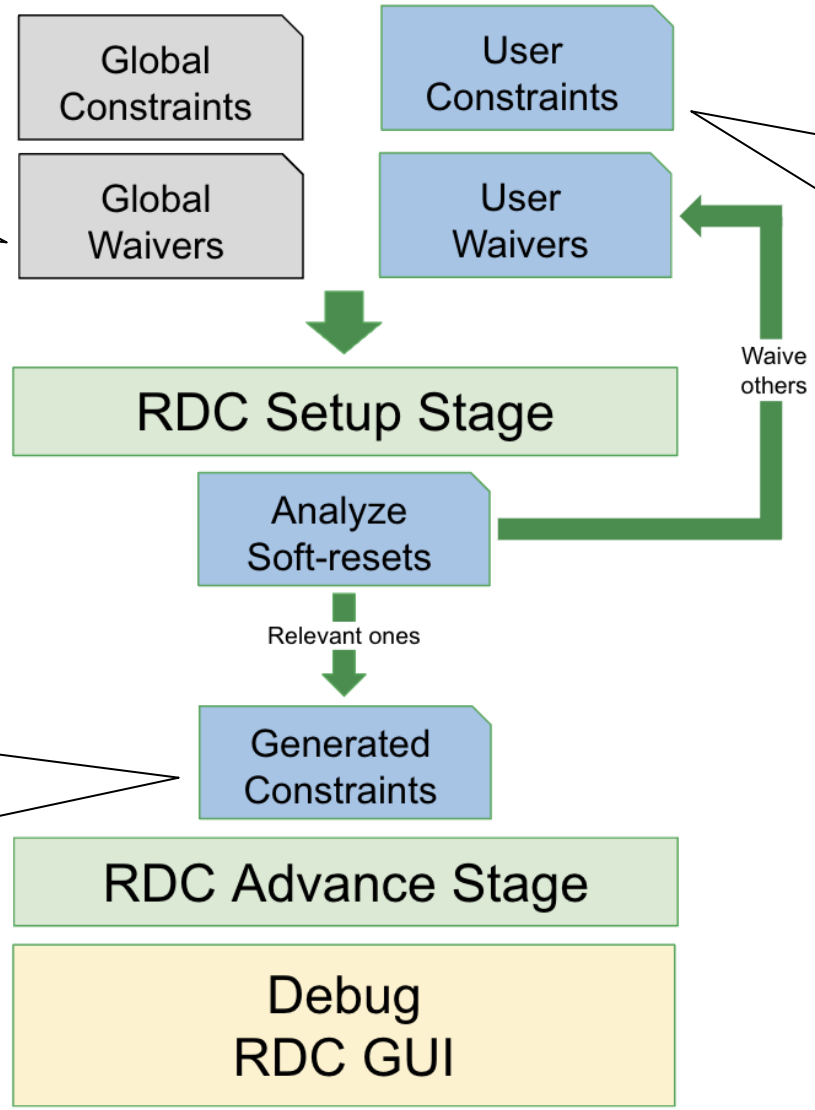
Methodology

Task 1	Task 2	Task 3	Task 4	Task 5
Designers can find all soft-resets in the design using setup stage by having **inferred soft-reset** rule	Primary inputs to the RDC tool are **reset constraints** and **reset grouping,** for resets coming from common generator module	In setup goal, find all soft-resets under a specific tag and dump them as constraints using **Custom Report Generation feature** using tool based Tcl Query	These generated constraints are used in the RDC tool with advanced goal which enables us to find **reset crossings between soft-resets and asynchronous resets**	Reset grouping enables us to **reduce noise** and only provides us with the **relevant soft-reset crossings** without having reset sequencing

Implementation

Implementation

waive_violation -tag { TAG_NAME } -status {Waived} -filter {(RdcDestResets:DestResetInfo:ResetName == "'no-reset'")} -app { rdc }



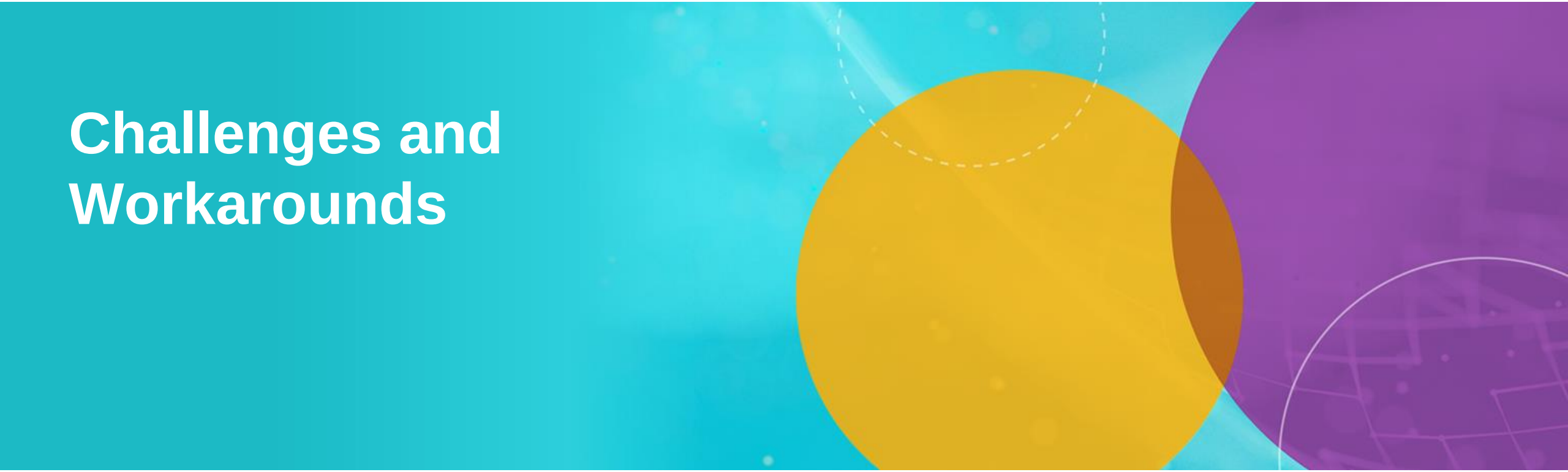
- create_reset {"rst_1"} -async -type reset -sense low
- set_reset_groups -name rst_grp_1 -group {rst_1 rst_2 rst_3}
- set_case_analysis 0 "rst_1"

report_violations -app rdc -tag <TAG_Name> -no_summary -include_waived -list -limit 0 -format "create_reset {\"%PotentialRoot%\"} -async -type reset -value low -disable_assertions_db"-file <FILE_NAME>

Important Note:

No crossings for the grouped reset will be reported in this methodology.

Challenges and Workarounds



Challenges and Workarounds

- **Larger Gate Count designs**
  - Use RDC constraints/abstract-models for lower level blocks to perform RDC in hierarchical mode

- **IPs having lots of Third Party IPs (3PIP)**
  - Black-box the third party IPs if,
    - Vendor delivers a clean RDC design
    - Boundary reset constraints are provided by the Vendor
  - The above process rules out the scenario where soft-reset from IP going into 3PIP
  - If above scenarios aren’t fulfilled then run flat RDC analysis

- **IPs with huge number of memories**
  - Ensure no external resets corrupt memory operations
  - If not true, constrain external resets driving memory for RDC analysis

Results

Results

Sub-System Design	Gate count	Performance				Violations	
		Setup (mins)	Memory (GB)	Setup+Advance (mins)	Memory (GB)	Num of Soft-resets	RDC Errors (Compressed)
Hardware Accelerator	350k	7.1	14	10.1	19	5	38
Security	200k	8.9	15	13.9	22	9	54
High Speed Peripheral	620k	12.8	20	49.1	63	140	189

Conclusions

Conclusions

- Proposed methodology **catches all the soft-reset related issues** identified by the software team
  - Reduce multiple iteration of RTL Reset Constraints releases

- Reduced turnaround **time and effort for manual review** in frontend design cycle
  - Turnaround time reduced from 1-2 week to 3-4 days

- These bugs normally found in **manual reviews at STA**
  - With catching these issues at RTL stage achieved shift left strategy

- Focuses solely on **violations from soft-reset crossings**, reducing noise compared to industry standards tools
- Can be **adopted by any user** using or planning for RDC analysis
- Addresses common industry challenges when enabling **RDC checks for soft-resets** as part of RTL Sign-Off Checks

Future Scope of Work

Future Scope of Work

- Designers **need to cleanup all CDC constraints** for seeing only reset crossings
  - Need to work on to find how to reduce constraints mapping

- For large design, observed **large amount of RDC crossings**
  - Need methods for reducing noise beyond existing suggestions
- Huge/Complex designs reports **longer runtime**
  - Needs improvement to reduce runtime for bigger design
- Analysis of **soft-reset stops at flop out, instead of going deep in the design** and reporting the exact pin
  - In figure 1, we are expecting tool should also report U1/Q as the soft-reset

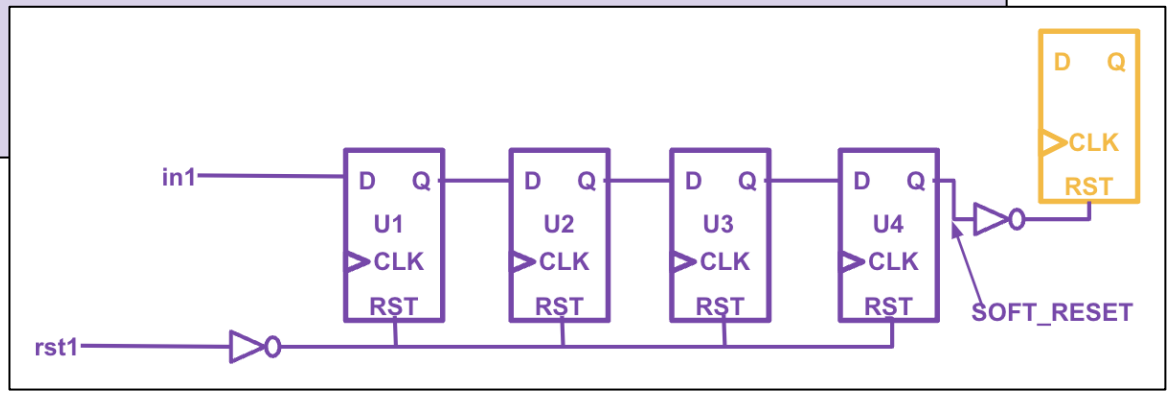


Figure 1

Google snug

Q&A

Our Technology, **Your Innovation™**

Google snug

THANK YOU

Our Technology, **Your Innovation™**