DVCon 2012

Design & Verification Conference & Exhibition

February 28 – March 1, 2012

Systematically Achieving CDC Verification Closure based on Coverage Models and Coverage Metrics

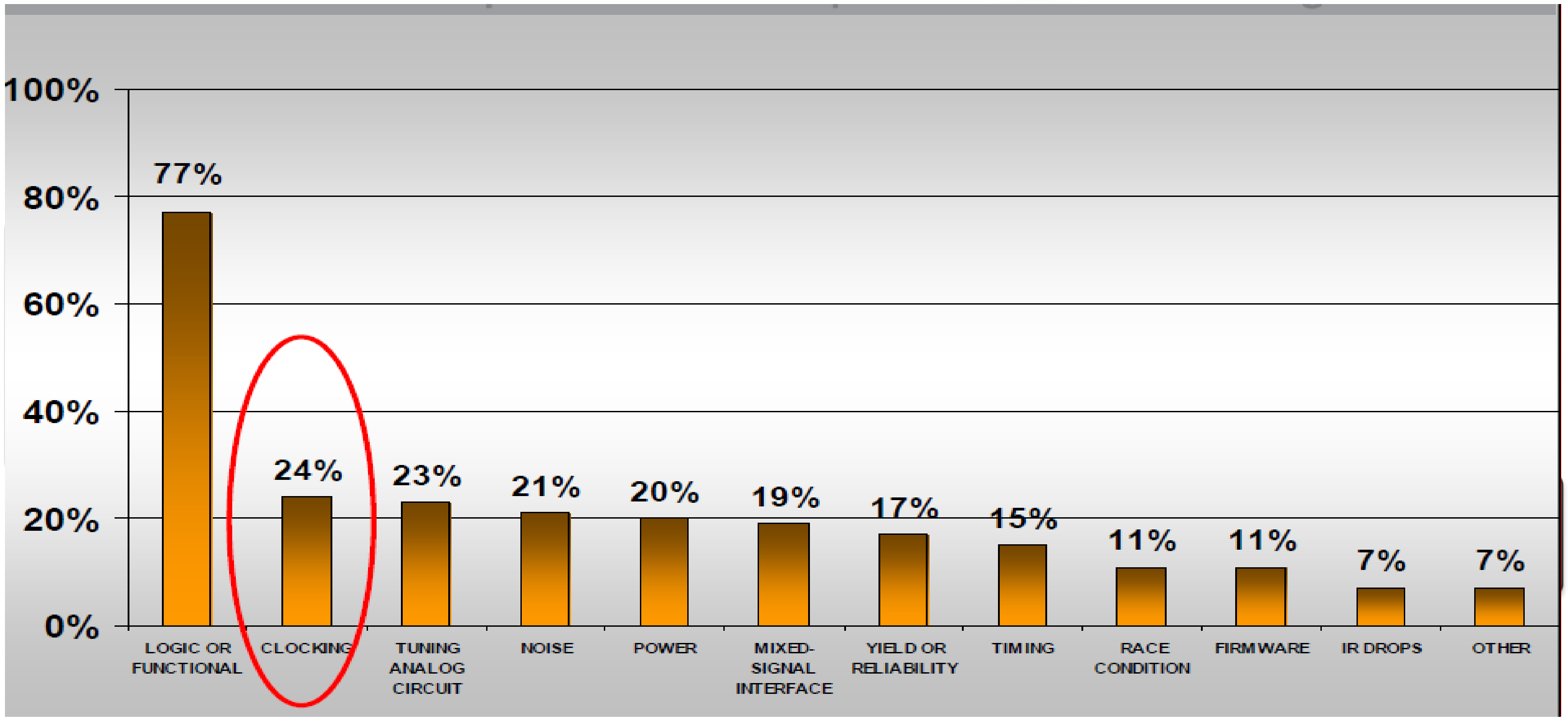
by

Ashish Hari, Yogesh Badaya

Mentor Graphics

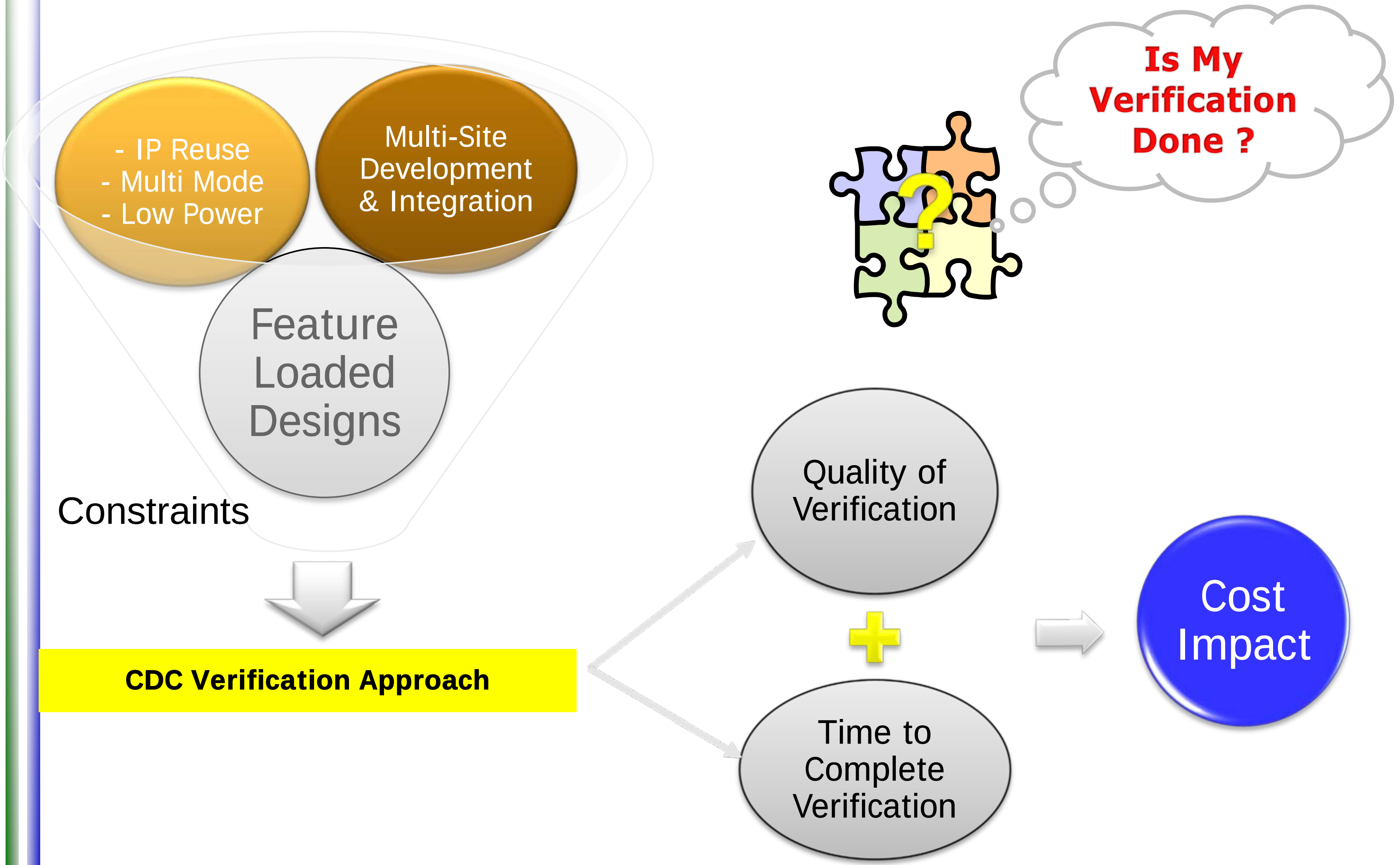
Mentor Graphics®

Motivation for CDC Verification



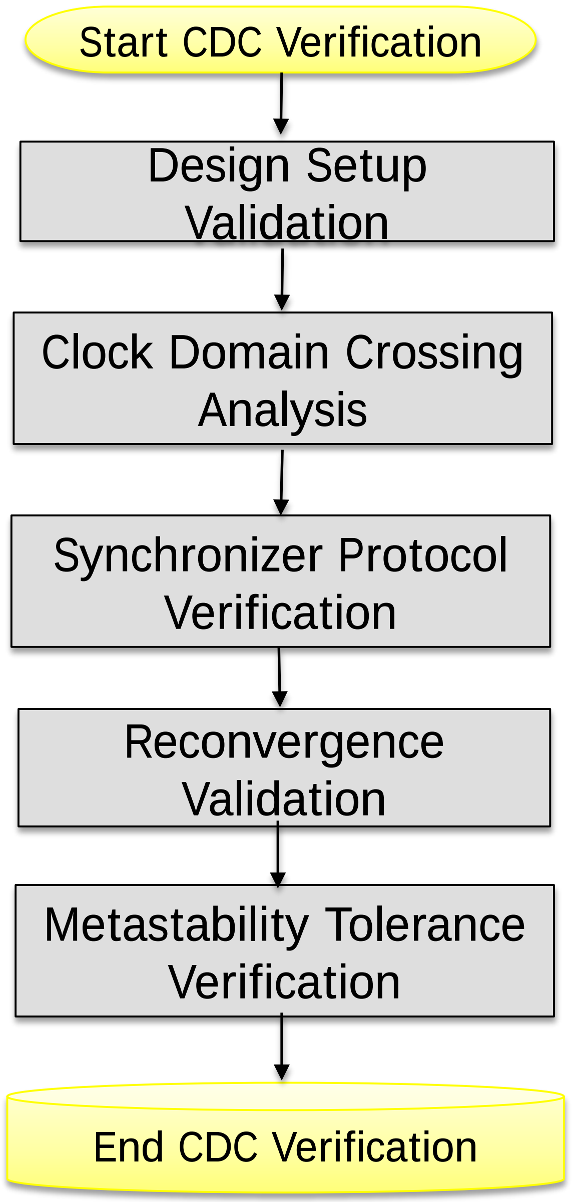
CDC Issues are #2 Reason for silicon respins !!

CDC Verification Challenge



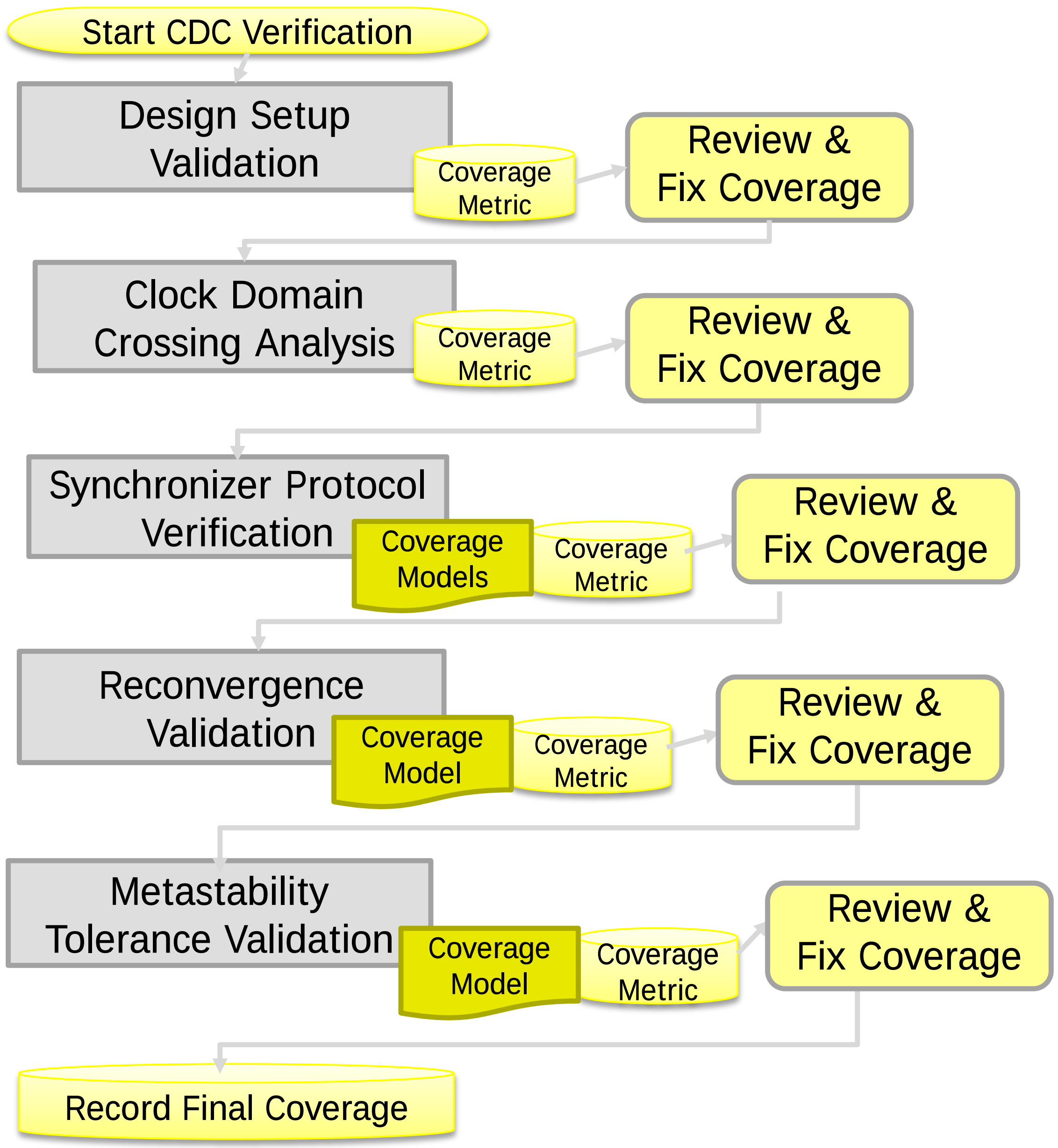
- IP Reuse
- Multi Mode
- Low Power
Multi-Site Development & Integration
Feature Loaded Designs
Constraints
CDC Verification Approach
Is My Verification Done ?
Quality of Verification
Time to Complete Verification
Cost Impact

CDC Verification Flow


Start CDC Verification
Design Setup Validation
Clock Domain Crossing Analysis
Synchronizer Protocol Verification
Reconvergence Validation
Metastability Tolerance Verification
End CDC Verification

- ❑ Sequential Flow : Proceed to next step only after completing prior step
- ❑ Decision, when to Proceed to next step based on:
  - Judgment of Verification Team
  - Time Available for Verification?
- ❑ Missing Link
  - Targeted Coverage
  - Know when to proceed to next step
- ❑ Solution
  - Coverage Metrics
  - Coverage Models

Coverage Based CDC Verification Flow



Goal – Achieve Targeted Coverage for Every Step

Coverage Metric and Coverage Model Objectives

- ❑ Comprehensively model each step of CDC Verification Flow
- ❑ Provide crisp information about verification progress
- ❑ Assist in directing verification engineer to focus on right areas by fixing design problems or adding directed test-cases

Step1: Design Setup Validation

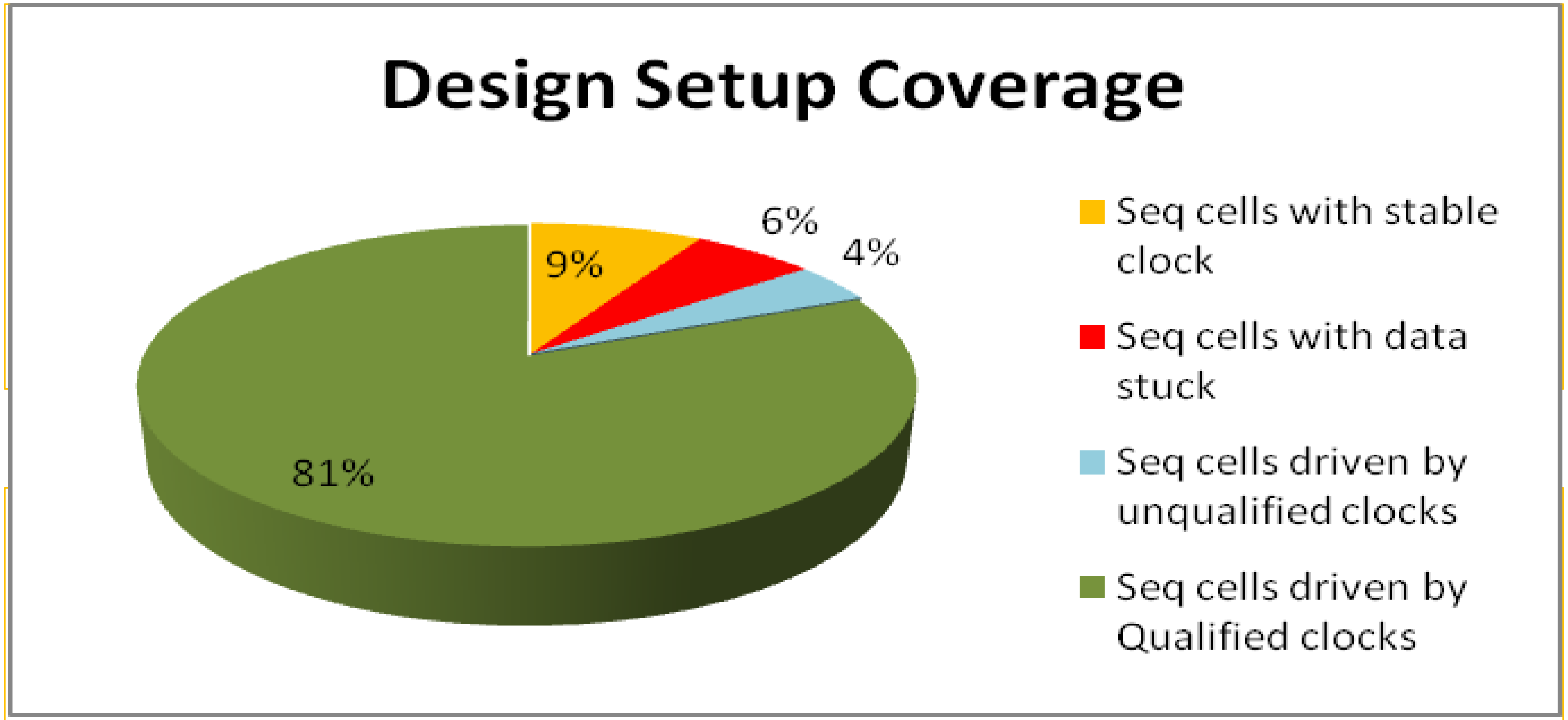
❑ Objective: Validate Clock Tree and Tune Design Configuration

Clock Tree Verification | Black-Box Qualification | Design Component Classification

1. **Clock Tree Verification**
   - All flops should be clocked by user specified clocks
   - Inferred clocks can be:
     1. Primary clocks : Review and Qualify if valid
     2. Gated Clocks : Mark enable signals as stable
     3. Muxed clocks : Appropriately constraint Mux select as per design mode.

Verification Target : ALL gated, muxed and inferred clocks should be resolved, or waived after qualification by the verification engineer.

Step1: Design Setup Validation ...

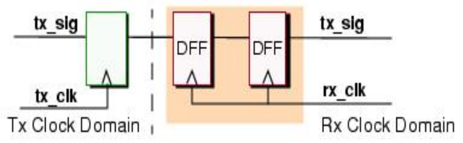


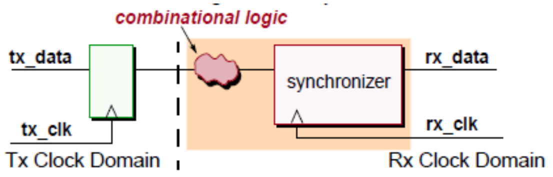
Verification Target : ALL portions excluding sequential cells driven by qualified clocks should reduce close to 0%

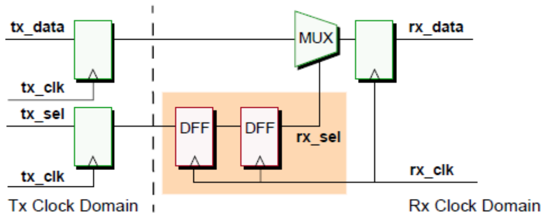
Step2: Clock Domain Crossing Analysis

Key Information identified at this step :

- Correctly Synchronized Crossings
- CDC with missing synchronizers
- CDC with incorrect synchronizers







Coverage for this step is defined as:

$$C2 = \frac{Xt - (k1 * Xm + k2 * Xi)}{Xt} * 100$$

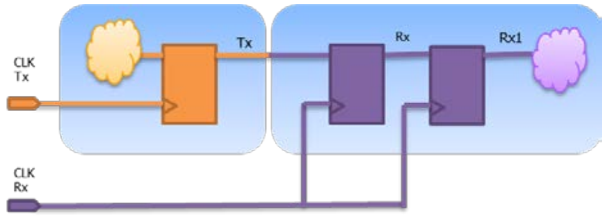
Where,
Xt: Total Clock domain crossings
Xm: Crossings with missing synchronizers
Xi: Crossings with incorrect synchronizers
k1, k2: weights, to be adjusted based on application or design requirements

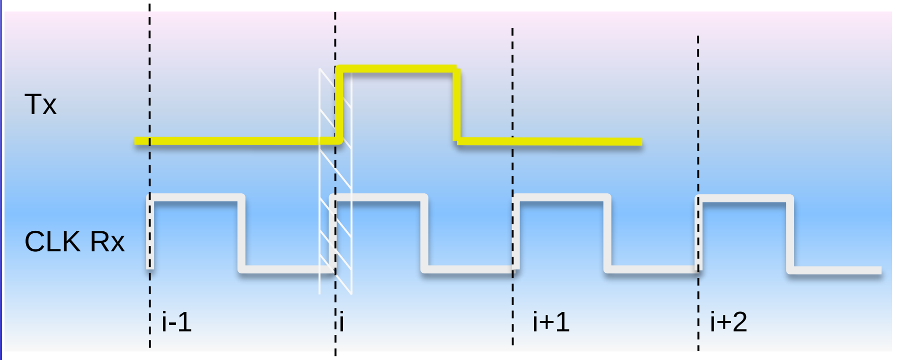
Verification Target :

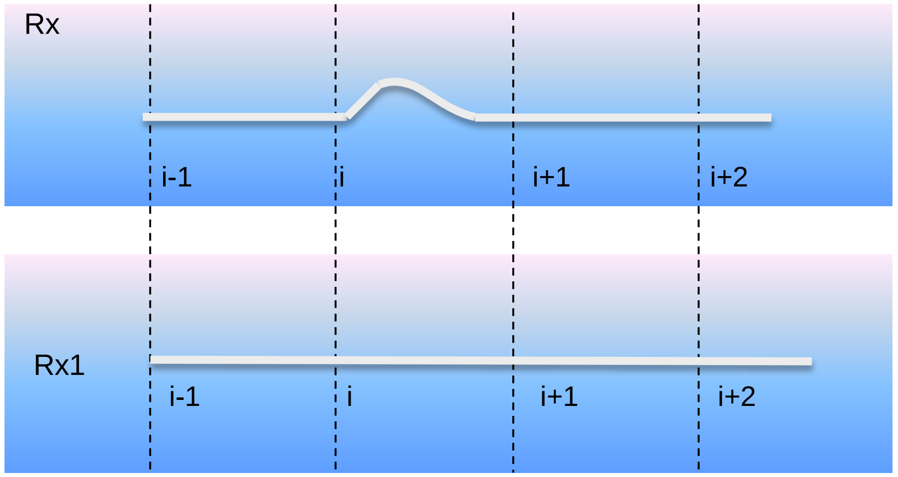
- Add missing synchronizer or waive CDC paths that have stable transmit signal
- Incorrect synchronizer structures should fixed, or qualified to be acceptable for particular protocol

Step3: Synchronizer Protocol Verification

- Every Synchronizers has assumptions → Protocols
- Protocol failure can lead to data loss
- Examples:
  - 2DFF - Data should be stable for 2 clock cycles in receiving domain
- Protocols assertions can be verified using Simulation or Formal techniques
- Coverage data collected by standard SV coverage constructs







Step3: Synchronizer Protocol Verification…

- Example of Handshake Protocol checks:
  - Data is stable when Request is asserted
  - Every Request gets Acknowledge in next 2 cycles
  - No Acknowledge without request

- **Verification Goal**
  Review these 3 types of coverage metrics:
  - **Protocol Coverage:**

    This can be defined as:

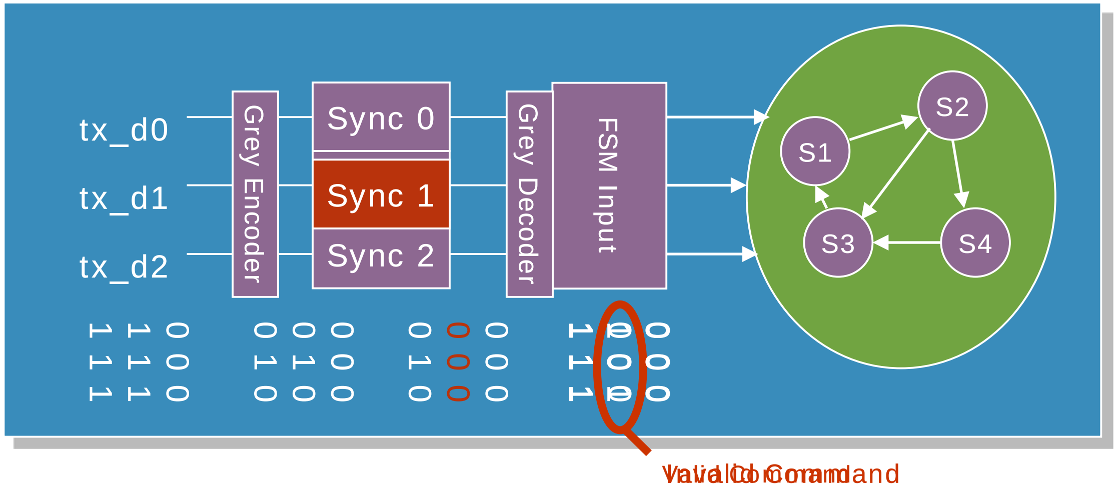
    $$C3 = \frac{Pt - Pu}{Pt} * 100$$

    Where,
    Pt : Total promoted protocols
    Pu: Uncovered Checkers

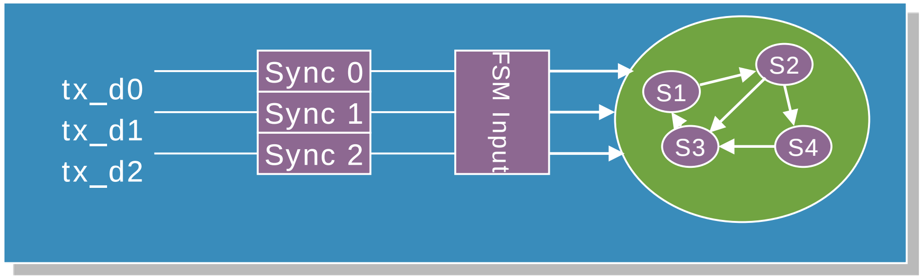
  - **Synchronizer Coverage**
    - Helps identify bugs or dead-code cases in synchronizer implementation.
  - **Check Coverage**
    - Every check of each synchronizer should be covered

```
property data_stable;
   @(posdedge clk)
      req |=>$stable(data) [*1:max]##0 ack;
endproperty  : data_stable

sequence req_ack_seq;
   @(posdedge clk)
      req ##1 !req [*1:max] ##0 ack;
endsequence : req_ack_seq

property req_has_ack;
   @(posedge clk)
      req |->req_ack_seq;
endproperty : req_has_ack

property ack_had_req;
   @(posedge clk)
      ack |->req_ack_seq.ended;
endproperty : ack_had_req


assert property (data_stable);
assert property ( req_has_ack);
assert property ( ack_had_req);
```

Step4: Reconvergence Verification



Step4: Reconvergence Verification



- Reconvergence of synchronized signals can lead to data coherency issues.
- Verification of Safe reconverence has two steps:
  1. Static analysis – If possible, reconvergence violation should be fixed structurally.
  2. If reconvergence is intentional, grey-encoding checks should be done on converging signals. The coverage for this step can be identified by standard SV checkers.

$$C4 = \frac{Rt - Ru}{Rt} * 100$$

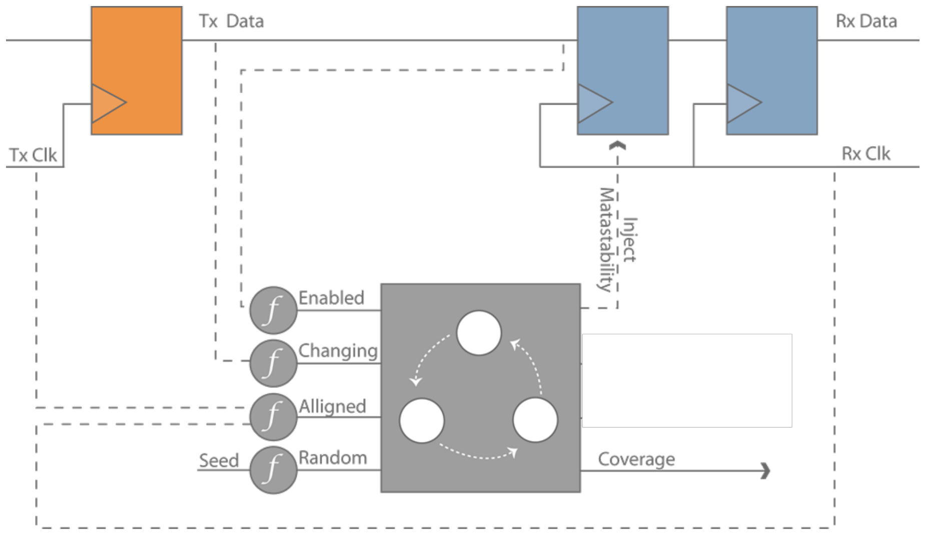
Where,
Rt: Total reconvergence conditions (excluding waived or structurally fixed cases)
Ru: Uncovered checkers for gray-encoding checks

Verification Target : Acceptable coverage ensures that there would be no unexpected data coherency issues at reconverging points leading to functional errors.

Step5: Metastability tolerance (CDC-Jitter) verification

- Synchronizers can inject a random cycle delay
- Designs should be verified for this random CDC-Jitter.
- Metastability injection models allow silicon accurate behavior on receive registers.

  Eg. Inject metastability effects on any one-bit of a synchronized bus signal
- These models are used in simulation to verify design behavior in presence of metastability effects.
- Satisfactory coverage of these assertions is critical to effective verification.



Coverage for this step can be defined as:

$$C5 = \frac{Mt - Mu}{Mt} * 100$$

Where,
Mt: Total CDC paths for which metastability model was inserted
Mu: Uncovered Checkers

Verification Goal : Acceptable coverage ensures that design is robust enough to handle metastability effect.

Overall CDC Coverage

- Step wise sequential flow based on coverage closure is the recommended approach.
- Overall CDC Coverage has significance only as a measure of verification quality, and not for debug.
- Recommended metric should reflect importance of sequential verification.

Overall Coverage Metric:

$$C = k1 * C1 + k2 * C2 + k3 * C3 + k4 * C4 + k5 * C5$$

Where,
C1 to C5 are coverage figures for various verification steps
K1 to K5 are weights, such that:
$k1 \geq k2 \geq k3 \geq k4 \geq k5$, and $k1+k2+k3+k4+k5=1$

The weights may need to be adjusted based on application and verification team priorities. For our experiments, k1=0.3, k2=0.3, k3=0.2, k4=0.1, k5=0.1

Coverage Model Design

- Coverage models populate relevant coverage information for protocol checks.

- Written as separate SV modules, connected to actual signals in design through bind statements.Non Intrusive and no modification needed in design.

- Model includes the following:

1. Protocol and Coverage checks:
   - Assertion Properties for synchronizer protocol, reconvergence, Jitter
   - Coverage data collection for these properties to ensure these are triggered and verified.
2. Debug Data :
   - Statistics around synchronizer functionality to collect useful info for debugging protocol violation or coverage holes.
3. Control Flags:
   - To avoid impact on simulation performance, user may want to limit some features, so the checks can be selectively enabled or disabled.

Case-Study

Step 1. Design Setup Validation		
Sequential Cells	***St***	12352
Unconstrained flops (no qualified clock connected)	***Su***	232
Constant Clock	***Sc***	0
Constant Data with No Reset logic	***Sd***	0
Coverage (C1)	$\frac{St-(Su+Sc+Sd)}{St}$	**98.1%**
C1 Revised	*Fixed Su*	**100%**
Step 2. Clock Domain Crossing Analysis		
Missing Synchronizers	*Scalar crossing no-sync*	*526*
	Bus crossings no-sync	*257*
	Xm	**783**
Incorrect Synchronizers	*Combo-logic before sync*	3
	Mux-sel with multiple sync	1194
	Xi	**1197**
Good Synchronizers	*Mux Sync*	*330*
	FIFO	124
	DFF	47
	Handshake	316
	Xg	**817**
Total Crossings	**Xt**	**2797**
Coverage (C2)	$\frac{Xt-(k1*Xm+k2*Xi)}{Xt}$	**29.2%**
	Revised Xm	0
	Revised Xi	16
C2 revised		**99.4%**

Step 3. Protocol Verification			
Synchronizer Protocol	Total Checks	Uncovered Checks	%age Covered
Mux Synchronizer			
Data stable check	330	27	92%
Tx_min_cycle check	330	23	93%
2 Flop Synchronizer			
Data stable check	47	2	96%
handshake			
Data stable check	316	4	99%
Req_has_ack	28	2	93%
Ack_had_req	28	2	93%
fifo			
write ptr is gray-encoded	14	3	79%
read pointer is gray-encoded	14	3	79%
overflow check	14	3	79%
underflow check	14	3	79%
Total Checks (Pt)	1135		
Total Uncovered Checks (Pu)		72	
Coverage (C3)	$\frac{Pt-Pu}{Pt}$	**93.7%**	**94%**

Step 4. Reconvergence Verification		
Static reconvergence		466
Fixed structurally		295
Grey-encoding checkers	**Rt**	171
Uncovered Checkers	**Ru**	28
Coverage (C4)	$\frac{Rt-Ru}{Rt}$	**83.6%**

Step 5. Metastability tolerance		
Paths Verified	**Mt**	548
Uncovered Paths	**Mu**	42
Coverage (C5)	$\frac{Mt-Mu}{Mt}$	**92.3%**

Overall Coverage	Original	Revised
$C = k1*C1+k2*C2+k3*C3+k4*C4+k5*C5$ $C = 0.3*C1+0.2*C2+0.2*C3+0.1*C4+0.1*C5$	74.5%	96.2%

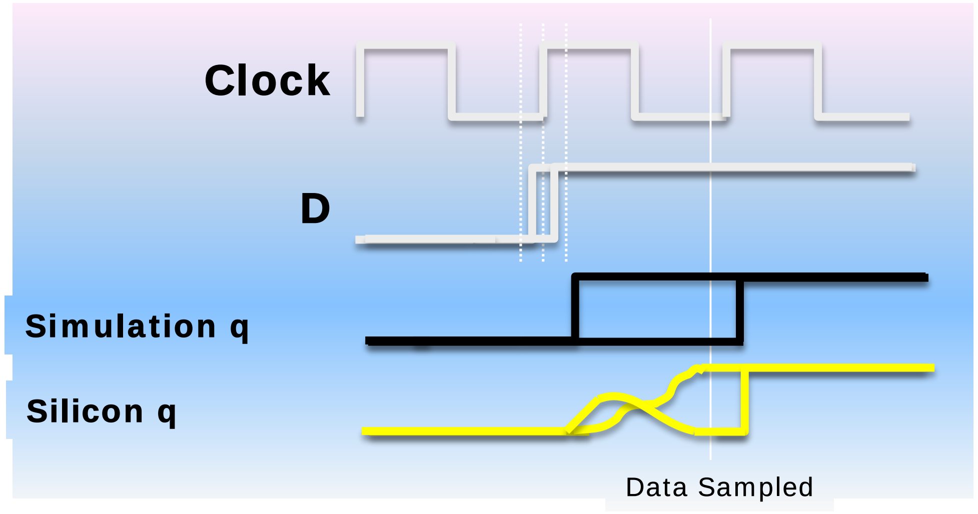
Summary

- ❑ Proposed Coverage based CDC Verification methodology helps achieve **systematic, accurate and reliable** CDC Verification closure.
- ❑ **Coverage models** accurately confirm functional verification of CDC protocols
- ❑ **Coverage metrics** enables verification teams to set **crisp sign-off targets** for each step of CDC verification flow.
- ❑ **Coverage driven methodology saves time and cost** spent on over-verification and guards against under-verification.
- ❑ Ensures that all CDC verification aspects are **covered** and verified comprehensively

Thank You!

Simulation - Silicon Mismatch

- RTL simulation doesn't model metastability
- Results in *mismatch* between simulation and *silicon* behavior



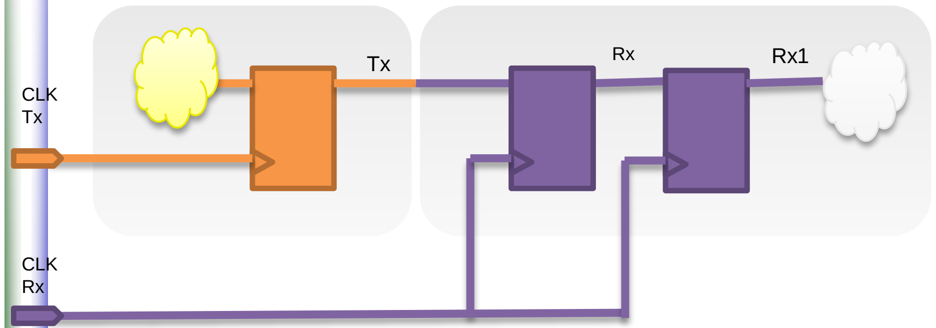
Hold Violation : Si rises to Random 1
Simulation captures 0, silicon captures 1
Silicon leads simulation by one cycle

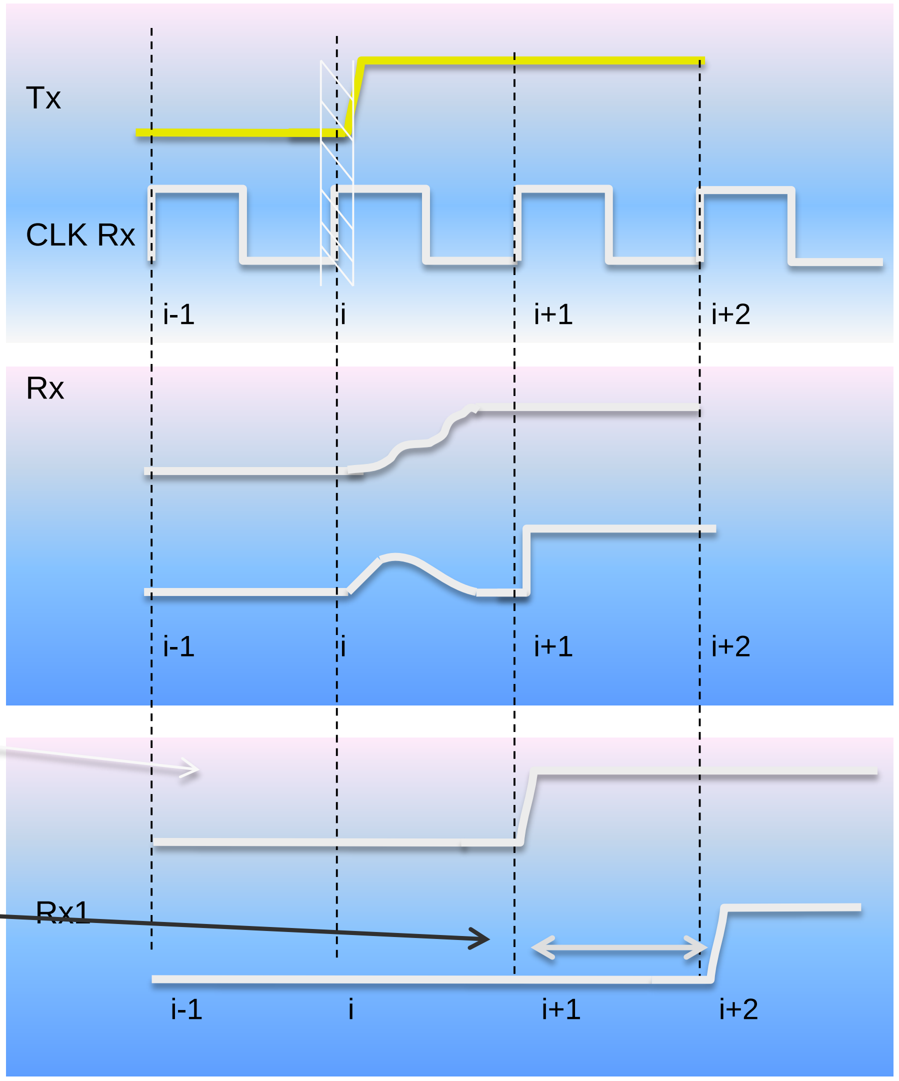
Hold Violation : Si falls to Random 0
Simulation captures 0, silicon captures 0
Silicon matches simulation

Setup Violation : Si rises to Random 1
Simulation captures 1, silicon captures 1
Silicon matches simulation

Setup Violation : Si falls to Random 0
Simulation captures 1, silicon captures 0
Silicon trails simulation by one cycle

Synchronizer effect (uncertain latency)





Synchronizer leads to stable output

Synchronizer may add uncertain delays