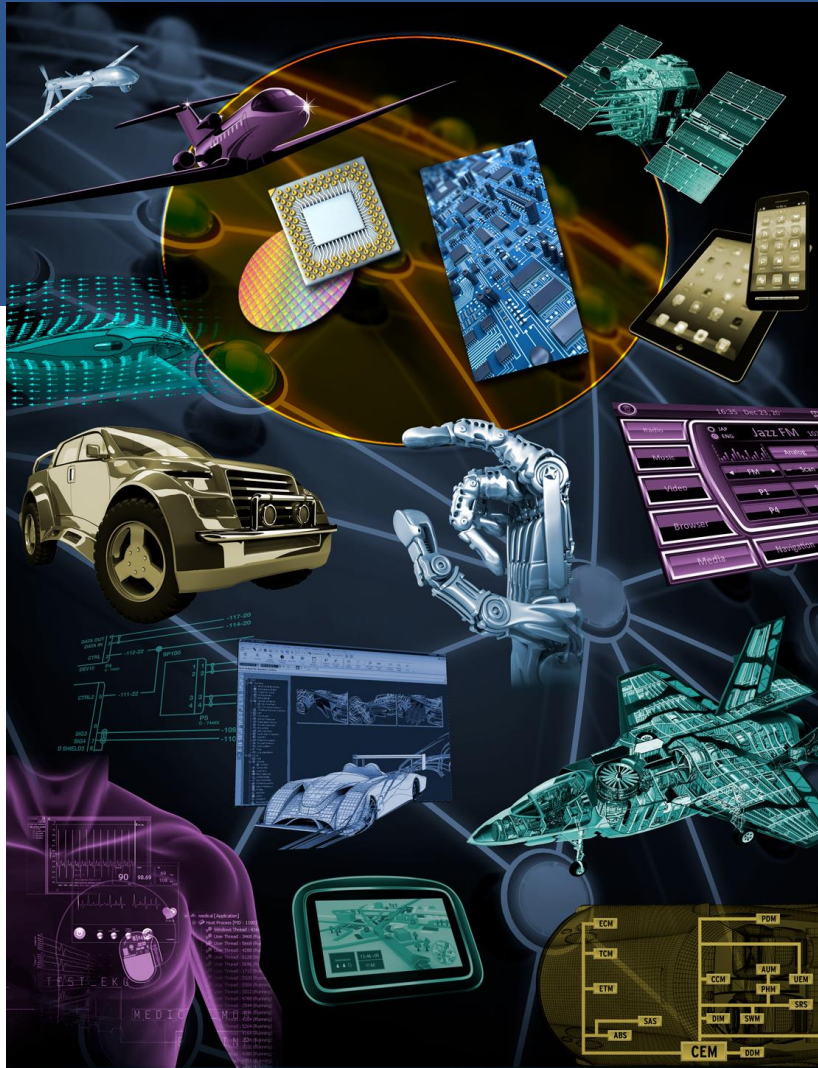


Exhaustively Verify SEU Mitigation Techniques Using Formal Verification

Mark Handover
Applications Engineer

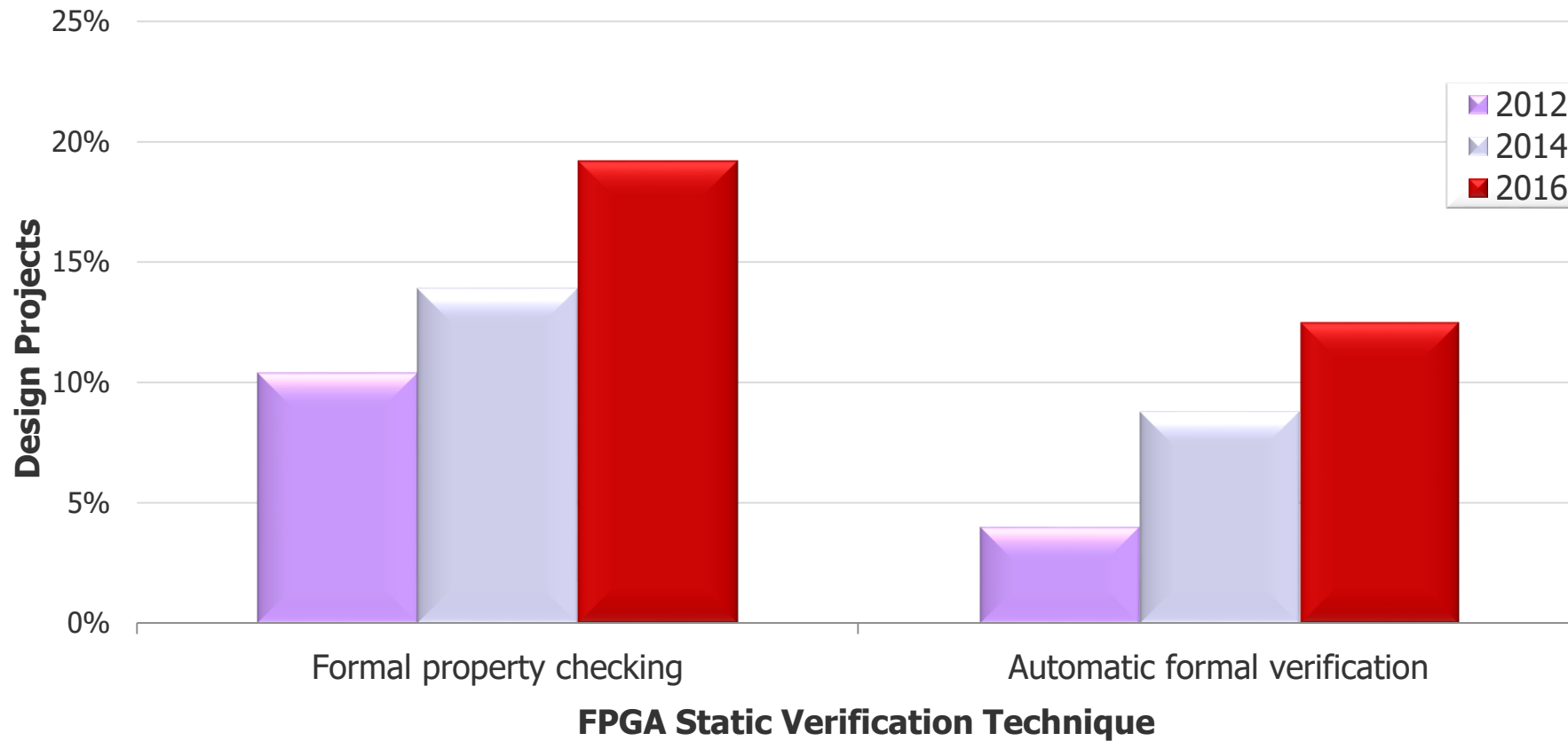
April 9, 2018



Today's Agenda

- Formal Verification – A Renaissance
- The Problem: Verifying SEUs With Simulation
- The Solution: Automated Formal Analysis
- Case Study

FPGA Developers Are Adopting More Formal



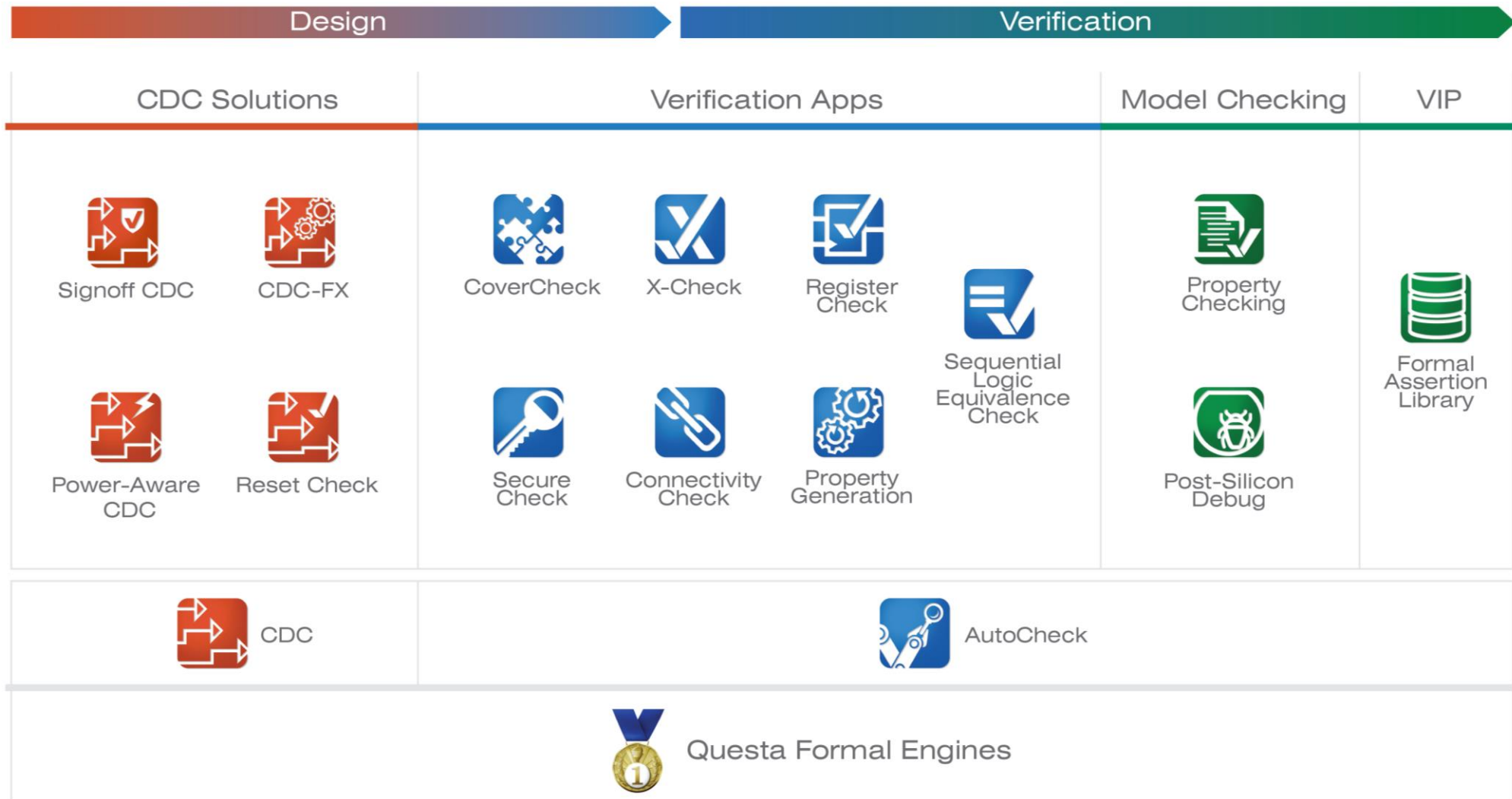
Source: Wilson Research Group and Mentor Graphics, 2016 Functional Verification Study

What's Driving This?

- More customers are demanding exhaustive results
- Tight schedules demand verification as early as possible
- Automated formal apps enable any engineer to use formal's power without having to learn formal
 - A formal-based tool focused on a specific, high-value verification challenge
 - Leverage the power of exhaustive formal algorithms without having to learn formal
 - Use “the best tool for the job”
 - increasing your verification effectiveness & throughput

Questa Formal Solutions & Apps

Automated, Exhaustive Verification For Complex Challenges

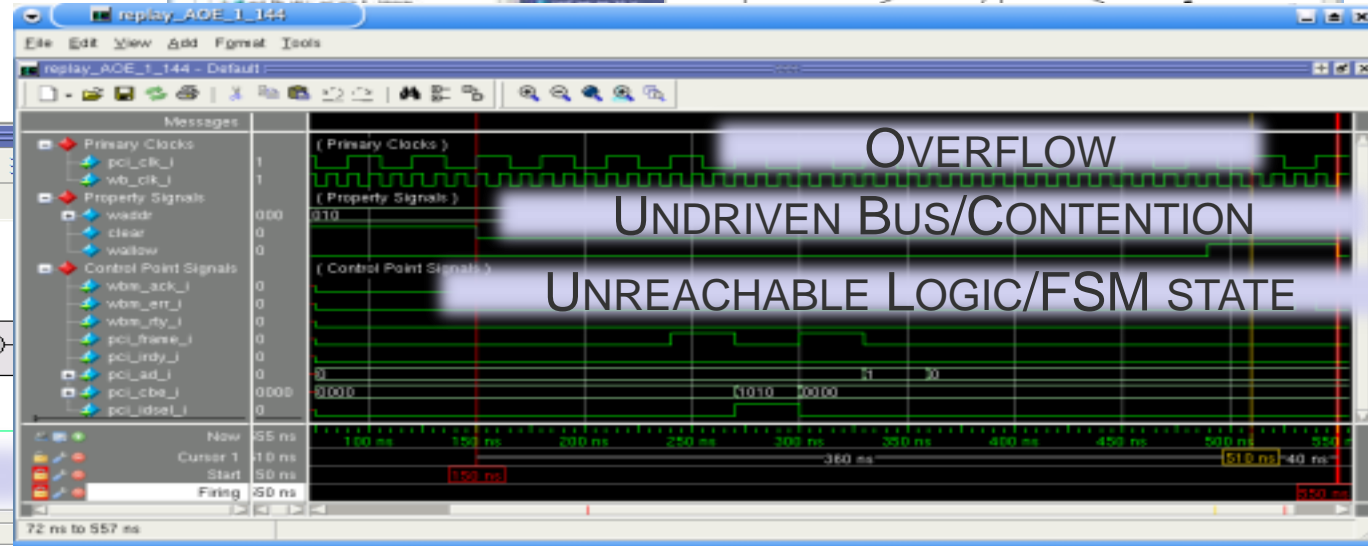
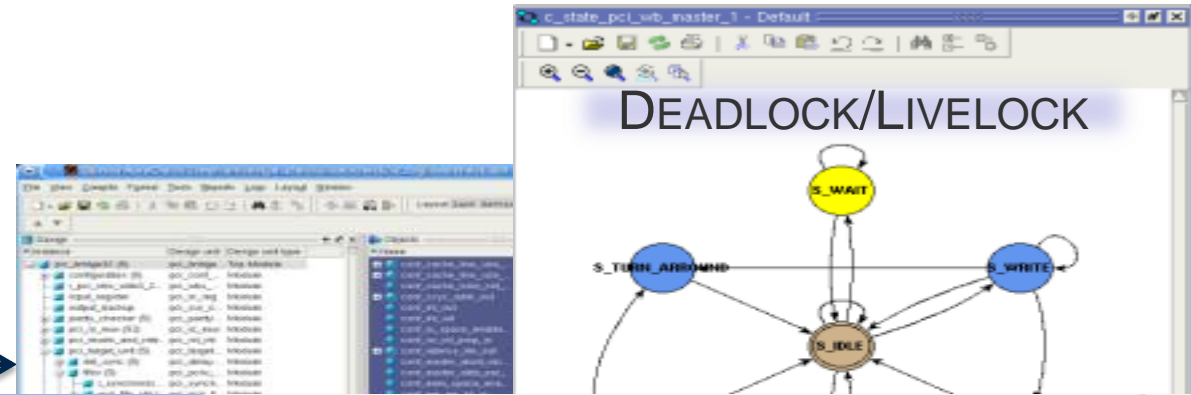
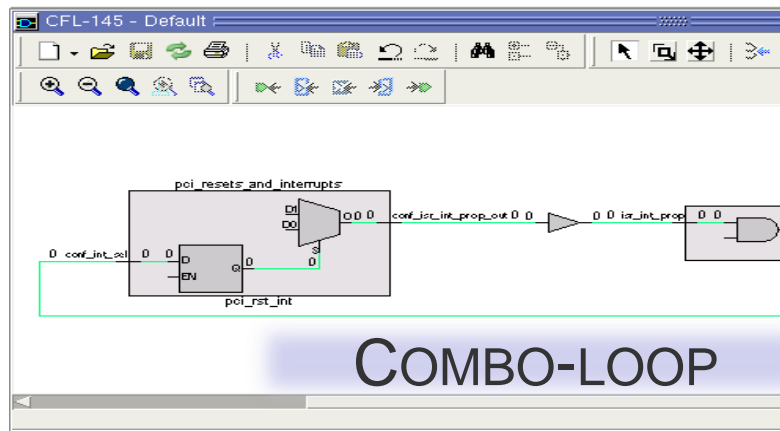
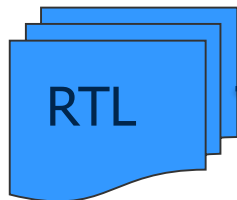




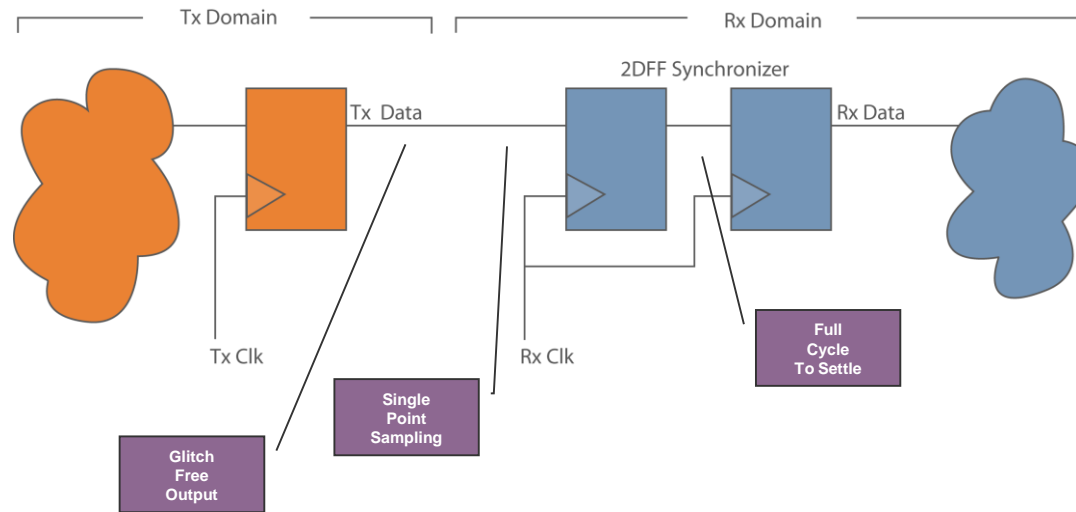
Questa AutoCheck: Push-button Formal

Eliminate bugs with low effort

- No testbench required
- No assertions required
- Rich debugging environment
- Easy to use – just provide RTL

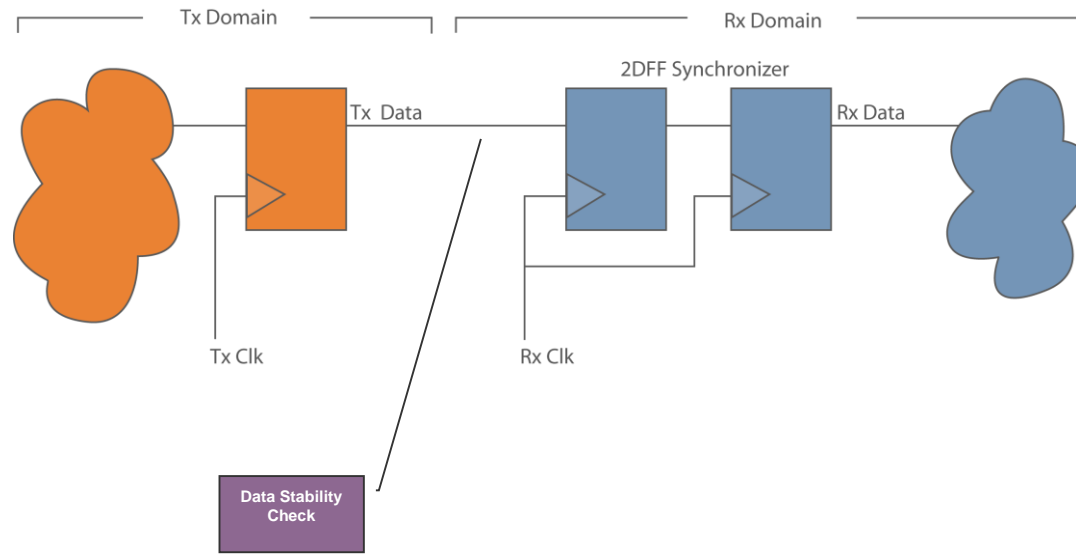


Clock Domain Crossing Analysis



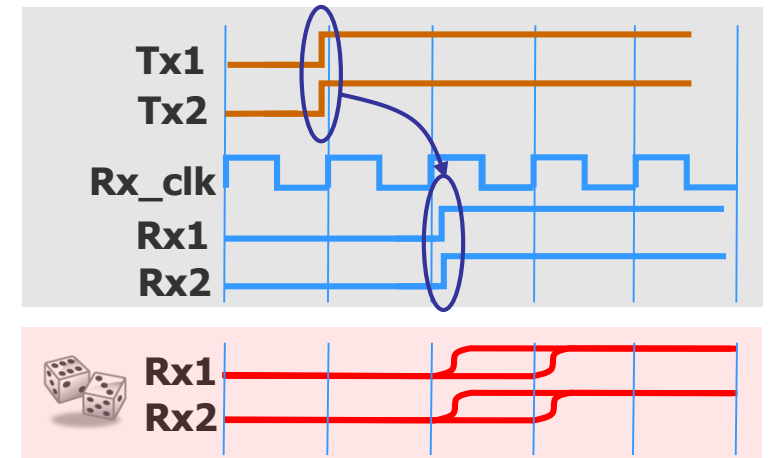
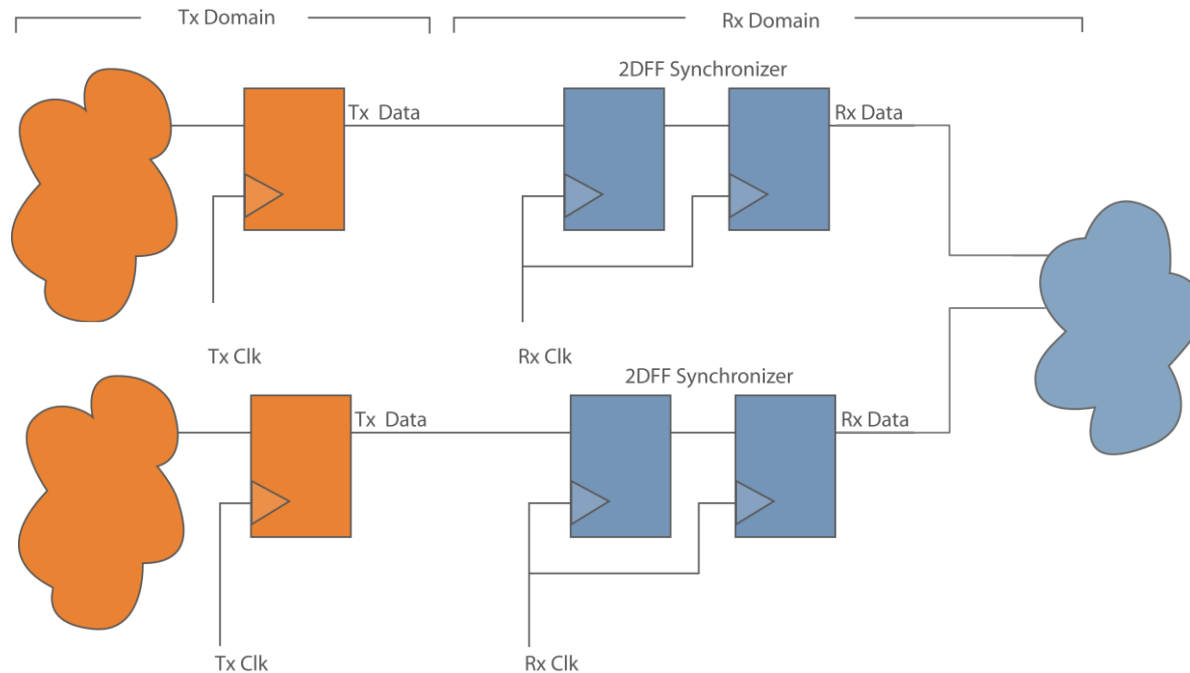
- Designers add synchronizers to reduce the probability of metastable signals
 - Are Synchronizers used in all the correct places?
 - Are they structurally correct?
 - Each synchronizer follows strict rules for that must be verified

CDC Transfer Protocols



- Synchronization between clock domains *requires* a transfer protocol
 - To ensure that data is *predictably* transferred between domains
- When protocol is violated
 - Data can be lost or corrupted
 - Simulation *may not* show a failure
 - Silicon implementation *will* eventually fail!

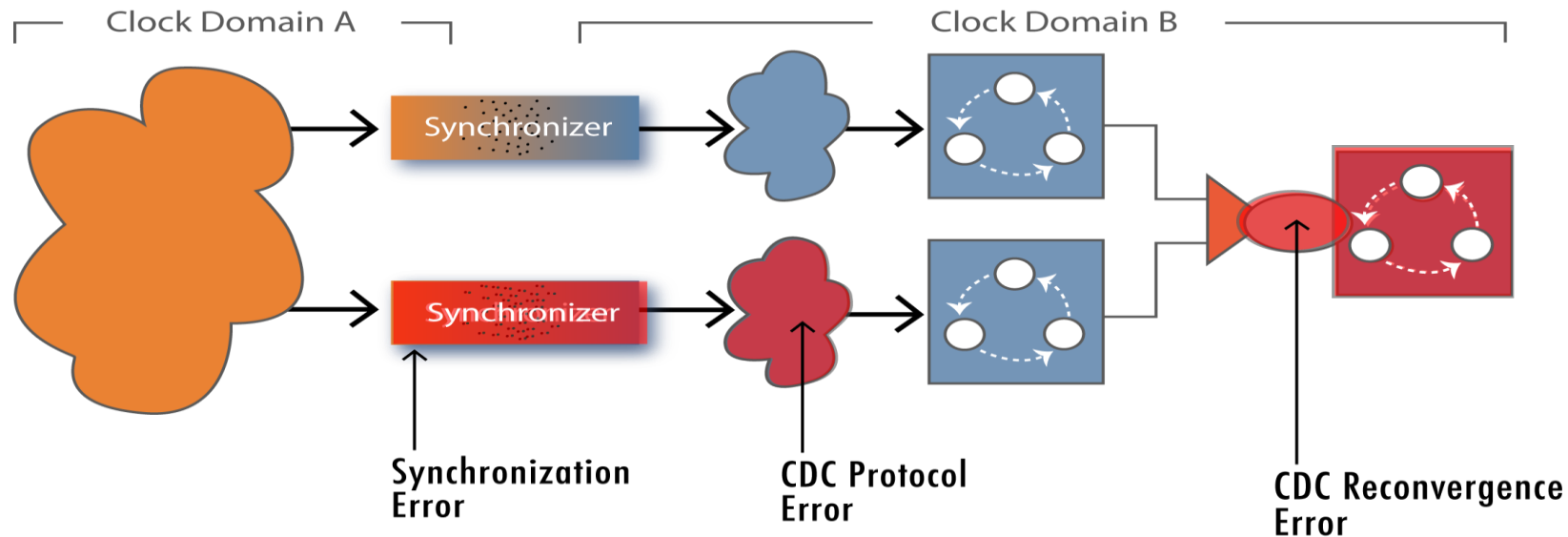
CDC Reconvergence



- Synchronizers can only resolve unpredictable values
 - Synchronizers cannot resolve *unpredictable delays*
- Timing relationships are maintained in simulation
 - *....but not in silicon*
 - If the RX domain depends on timing relationships, it will lead to a functional bug because of unpredictable delays through synchronizers

Questa CDC

The industry benchmark for CDC verification



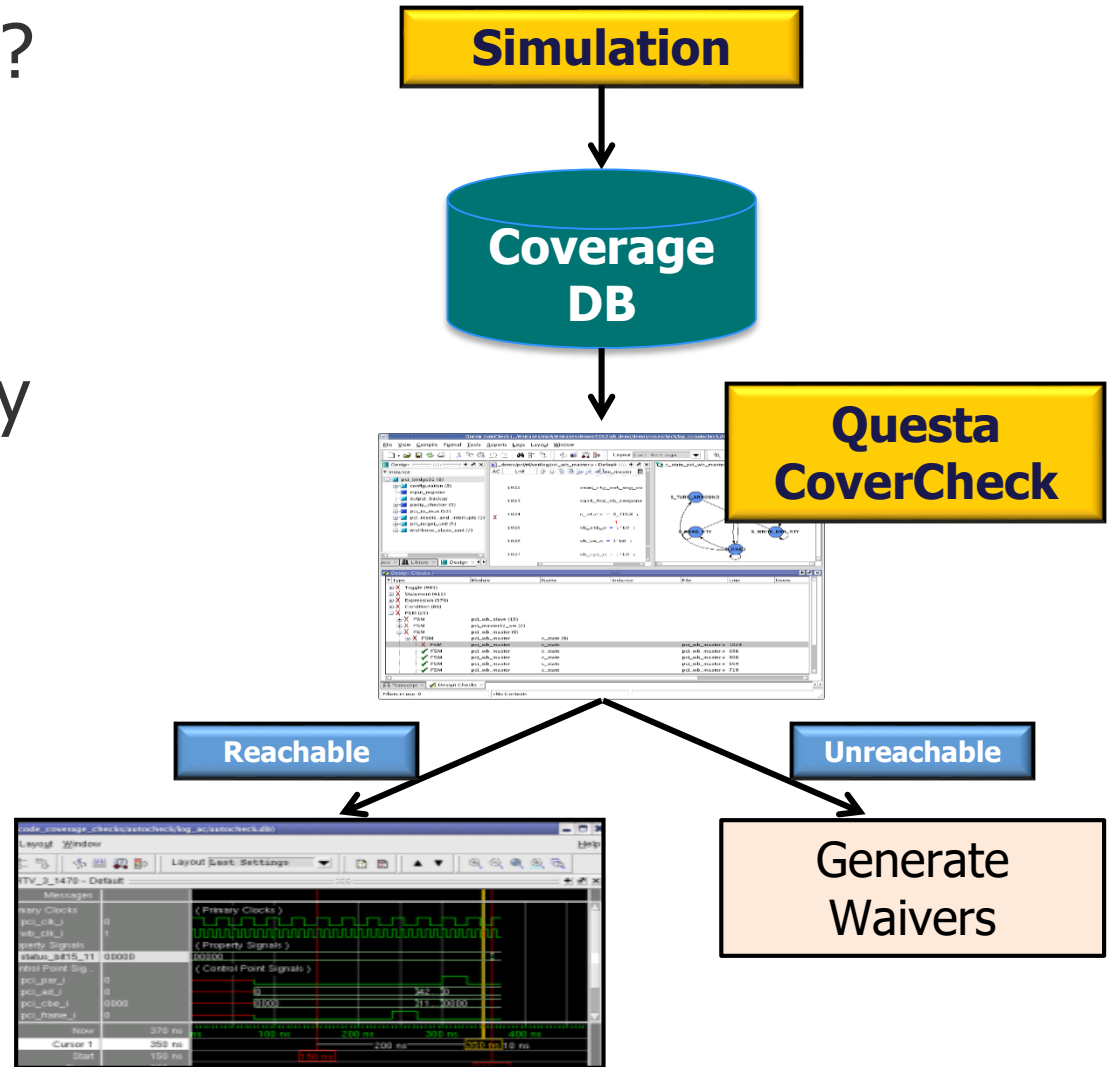
- Structural analysis of the design to ensure proper synchronization
- Complete verification of transfer protocols
- Ensuring a design is immune to the effects of metastability (testing for reconvergence problems)
- A comprehensive verification solution that combines technology, methodology, and metrics
 - Static analysis, simulation, formal and metastability effects verification

Questa CDC: A Comprehensive Solution



Questa CoverCheck: Automatic Coverage Closure

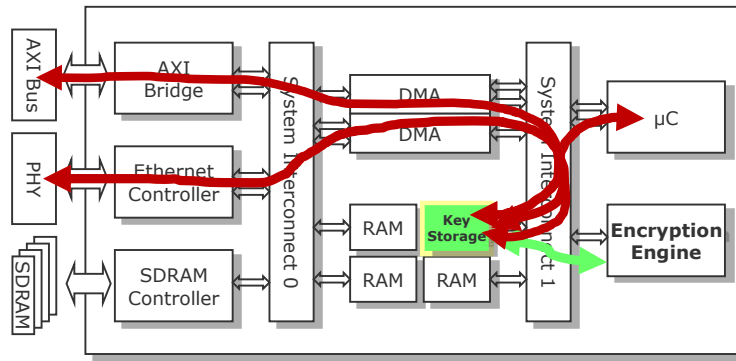
- Code Coverage less than desired?
 - How much effort to close the gap?
 - How much of the remaining is unreachable?
- Automated Coverage Reachability analysis
 - Unreachable Coverage
 - Automatic waiver generation
 - Reachable Coverage
 - Automatic witness waveforms





Questa SecureCheck – Critical Path Analysis

- Sneak path analysis
 - Is there any way for some event to happen, other than the correct way?
- Secure path analysis
 - Is the Desired/Secure Path The ONLY Path?



Waveform shows flow of Insecure data

Easily View Results

Visualize the Insecure Path

Today's Agenda

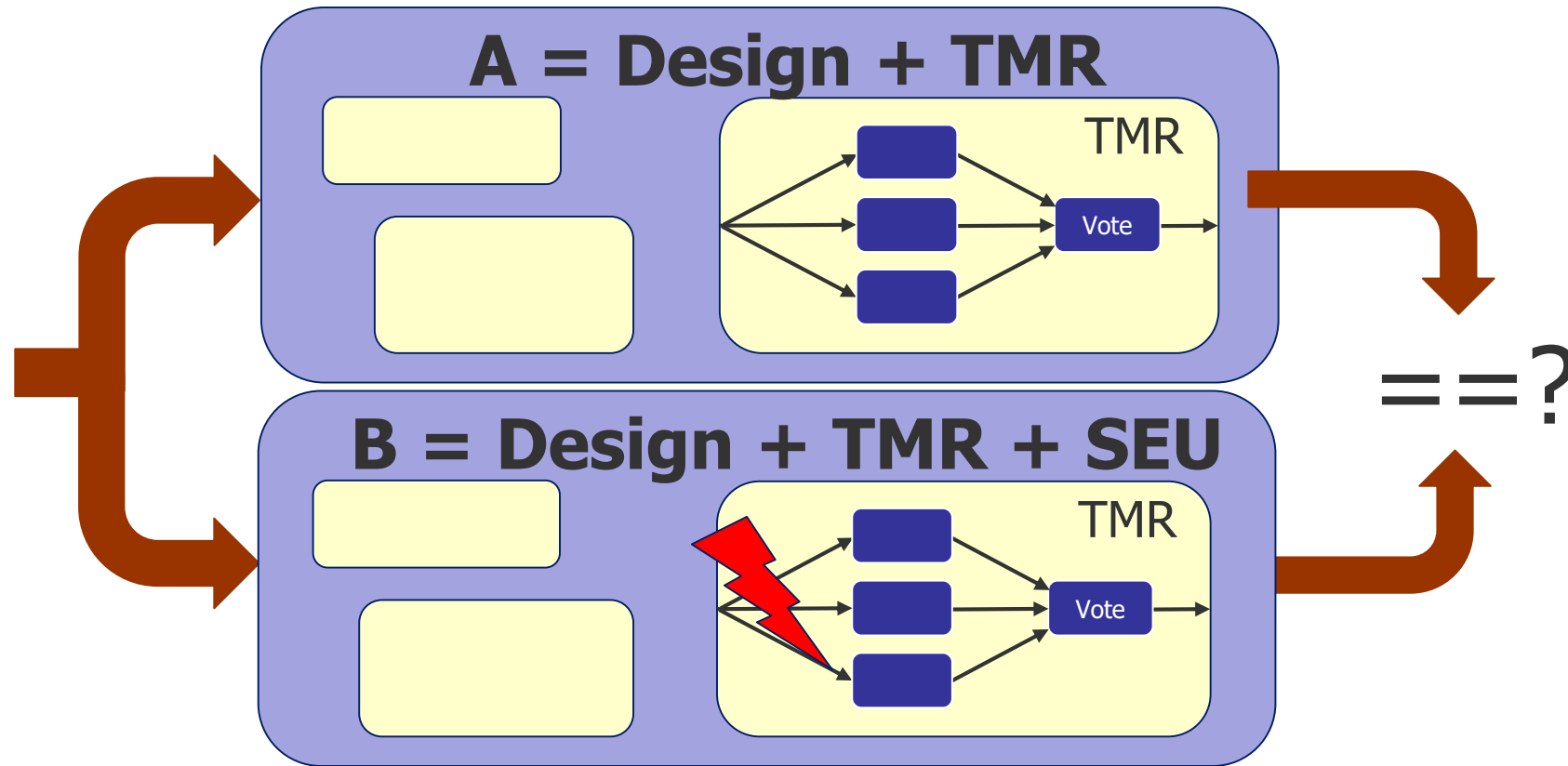
- Formal Verification – A Renaissance
- The Problem: Verifying SEUs With Simulation
- The Solution: Automated Formal Analysis
- Case Study

The Problem: Verifying SEUs with Simulation

- Simulation is only as good as the test written -- garbage in/garbage out
- Overhead: must verify “triple voters” and clock/reset domains before you can even start testing mitigation logic
- Parsing & sorting results with scripts is time consuming and error prone
- Writing tests to force values at memory elements and check for corrected results also time consuming and error prone

**Bottom line:
SEU phenomena can not be exhaustively verified
by simulation-based approaches**

The Solution: Formal Verification



**Use formal to mathematically prove outputs
A = B for the same inputs, for all time**

Questa Fault Injection and Equivalence Flow

Targeted SEU and stuck-at fault analysis without a testbench



Formal-based flow focused on validating the Safety Mechanisms' successful detection and handling of faults

Inputs:

A = DUT logic + "Safety Mechanism"

B = A + faults injected

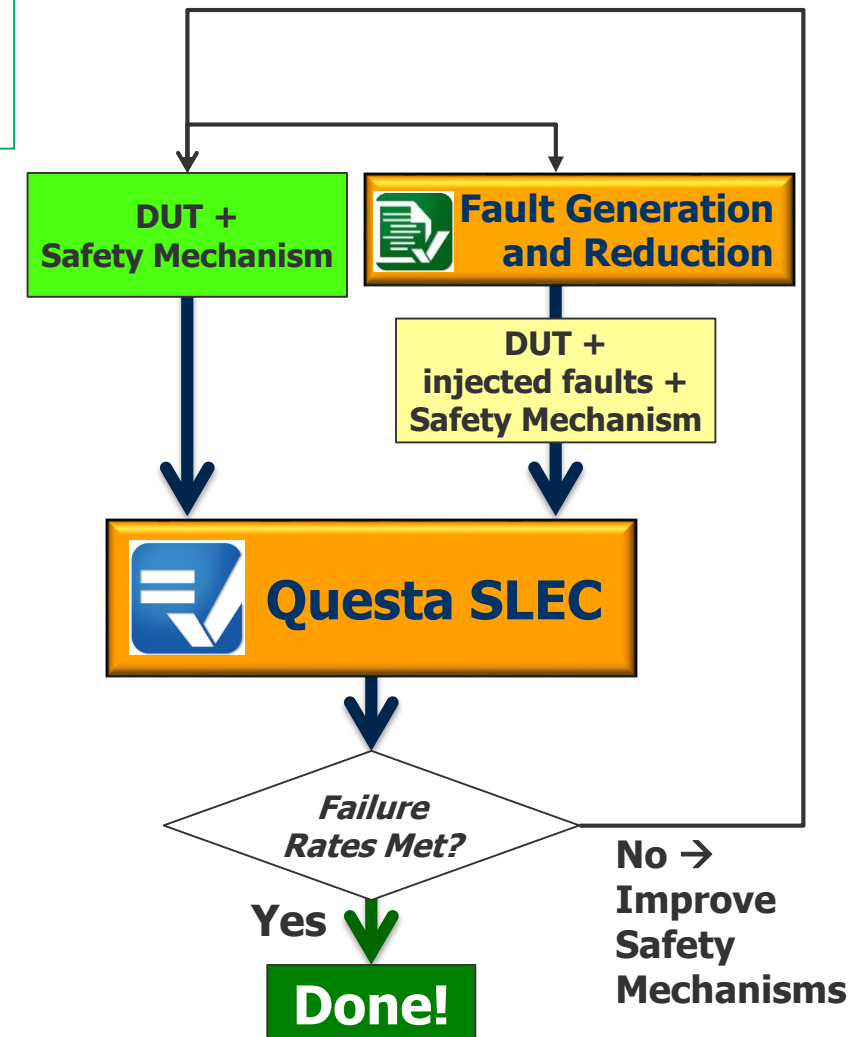
Desired output:

Does the output of the fault case match the normal case?

i.e. does Safety Mechanism detect the fault, and react properly?

Benefit:

Exhaustively prove which faults affect functional safety

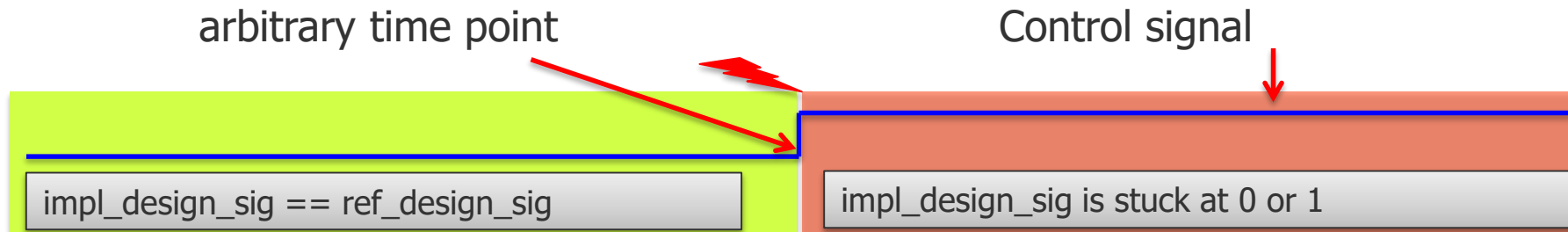


Categories for Faults

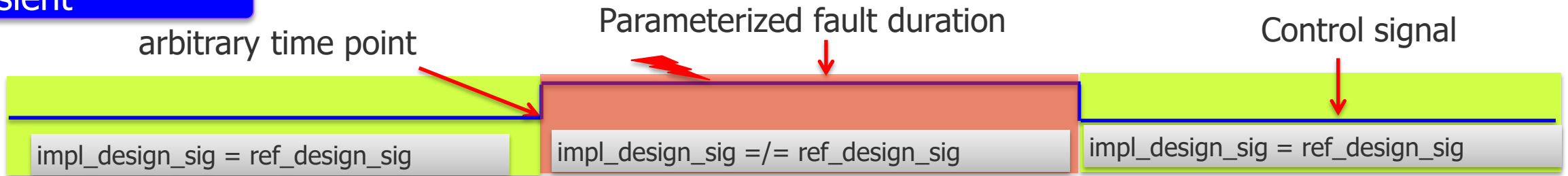
- 1. Permanent Faults** (Stuck at 0, Stuck at 1)
 - Irreversible component damage
- 2. Transient Faults** (a.k.a. soft-errors, SEU and SET)
 - Environmental Conditions
 - Cause Erroneous States in the system
 - Do not cause permanent damage
 - Hardest to detect
- 3. Intermittent Faults**
 - Caused by unstable HW
 - Often become **permanent** faults after a period of time

Modelling Faults in Formal

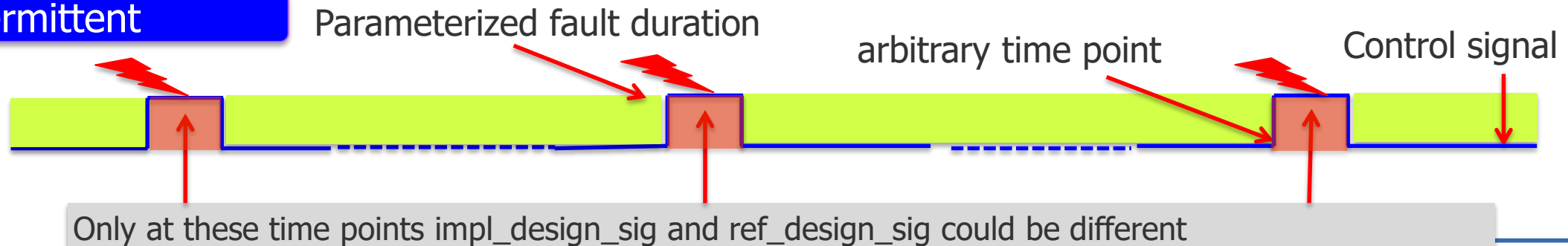
Permanent



Transient



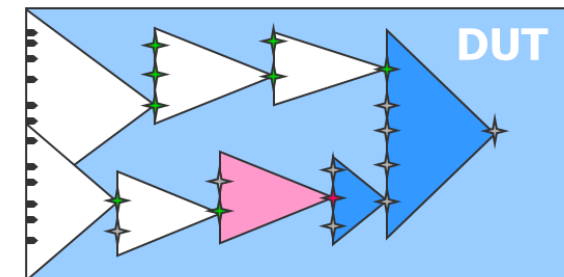
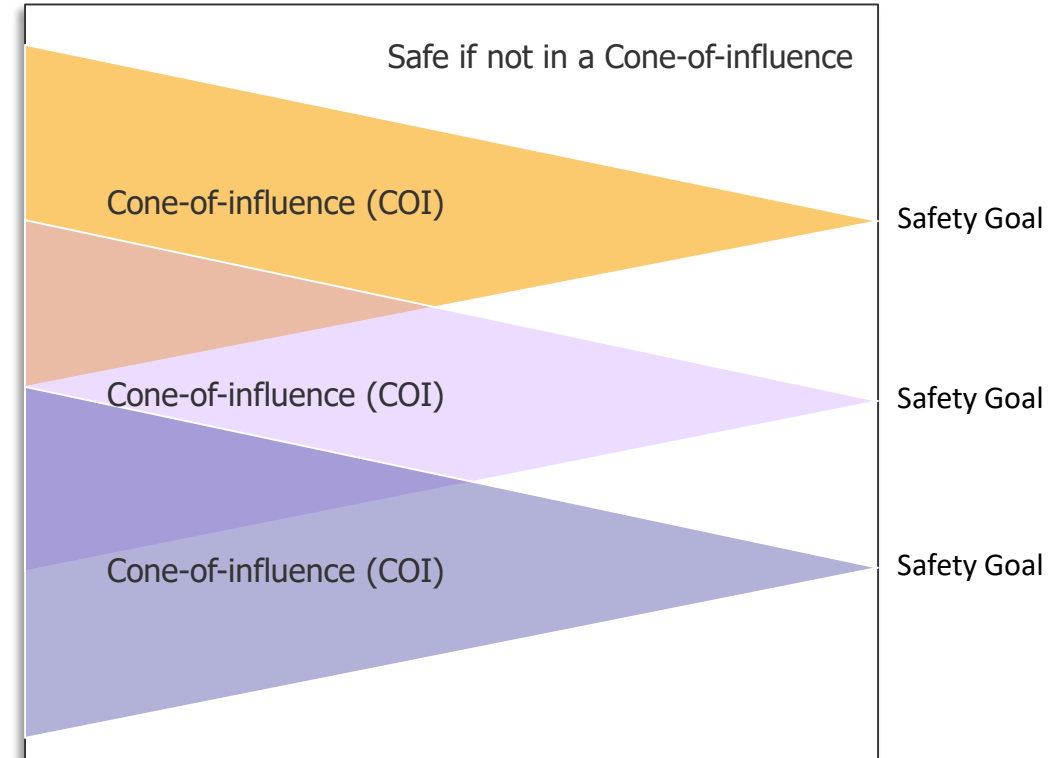
Intermittent



Questa Fault Analysis: Fault Pruning

Reducing the set of faults that need to be fault injected

- **A subset of faults**
 - Only a subset of faults in a given design will affect the safety requirement. They are in the COIs of the safety critical signals
- **Safe elements**
 - Design elements not in the COI of a safety critical signal are automatically considered safe
- **Configurations and constraints**
 - The COI can be reduced further by applying top-level constraints such as disabling DFT, debug and test, or other non-operational modes



Questa Fault Analysis: Safety Mechanism

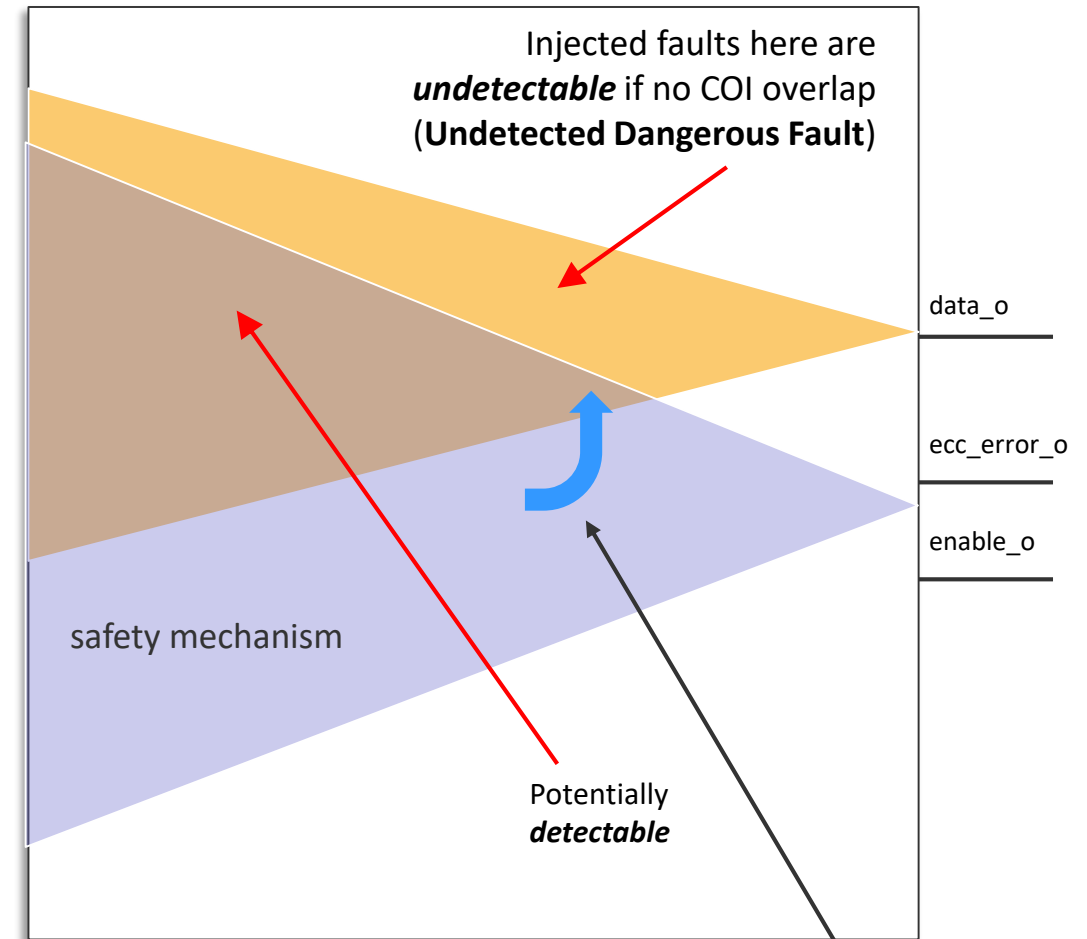
Safety Mechanisms reduce fault Injection requirements

■ Detectable fault

- Design elements in the COI of a safety requirement, and
- *Overlap* with the COI of the associated safety mechanism

■ Undetectable fault

- Design elements in the COI of a safety requirement, and
- *Not* in the COI of the safety mechanism
- Must be considered a dangerous fault



Hardening is to drive more overlap

Example Results



The screenshot displays the Questa SLEC 10.6a-INTERNAL interface. The main window shows the SLEC Properties table with columns for Name, Radix, Time, Spec Signal, and Impl Signal. A yellow callout bubble highlights the entry for SLEC_output_4, indicating that the fault path output is not equivalent between the specification and implementation.

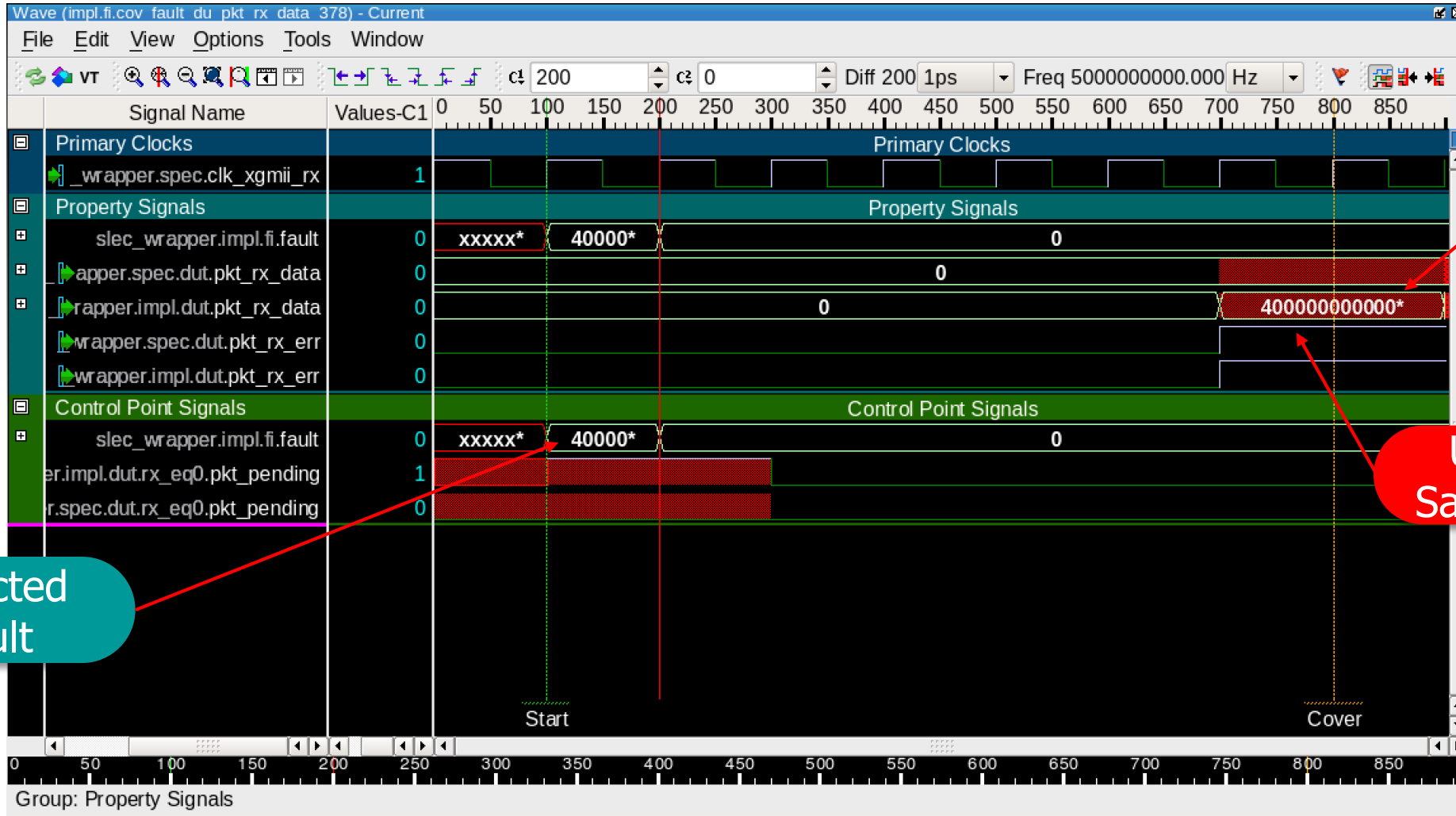
Name	Radix	Time	Spec Signal	Impl Signal
SLEC_output_4	1	1s	u_spec.BVALID_o	u_impl.BVALID_o
SLEC_output_1		2s	u_spec.ARREADY_o	u_impl.ARREADY_o
SLEC_output_2		2s	u_spec...	u_impl...
SLEC_output_3		2s	u_spec...	u_impl...
SLEC_output_5		2s	u_spec...	u_impl...
SLEC_output_6		2s	u_spec...	u_impl...
SLEC_output_7		2s	u_spec...	u_impl...
SLEC_output_8		2s	u_spec...	u_impl...
SLEC_output_9		2s	u_spec...	u_impl...
SLEC_output_10		2s	u_spec...	u_impl...
SLEC_output_11		2s	u_spec...	u_impl...

The waveform viewer (Wave (SLEC_output_4) - Current) shows signals for Primary Clocks (u_spec.ACLK_i, u_impl.ACLK_i, u_spec.PCLK_i, u_impl.PCLK_i, ipec.PCLK_i_csr, impl.PCLK_i_csr) and Property Signals (spec.BVALID_o, impl.BVALID_o). The signals are shown at time 0 and 10. The BVALID_o signal is highlighted in red, indicating a fault condition.

Only the fault path output is not equivalent

Questa Fault Injection and Equivalence

Dangerous Undetected (DU) Fault



Injected fault

Output difference detected

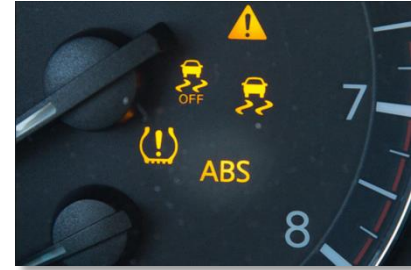
Undetected by Safety Mechanism

Questa Fault Flow: Early Adopter Success at EU Auto Systems Supplier



■ Verification Project Outline

- Exhaustively verify fault tolerance of ABS digital logic
- Small team, new to formal: an automated approach is essential




■ Partnership Setup

- Close methodology partnership with MGC and customer engineers
- Built tailored, automated solution on top of Questa SLEC app

■ Success to date

- 40,000 fault points narrowed down to ~1000 “unsafe”
- 2 hours setup, 2 days run time – **weeks faster than simulation**

For More Details



Fault Proof: Using Formal Techniques for Safety Verification and Fault Analysis

Adrian Traskov, Thorsten Ehrenberg, Sacha Loitz
Continental Teves AG & Co. oHG, Frankfurt a.M., Germany

Abdelouahab Ayari, Avidan Efody, Joseph Hupcey III
Mentor Graphics

Abstract— Safety mechanisms designed to correct/detect random hardware failures implement critical functionality but are relatively low in gate count, which often makes them an ideal application for formal verification techniques. In this paper we present a case study describing fault analysis of a Triple Modular Redundancy (TMR) element and its associated majority voter using formal. We start by describing a generic flow for fault analysis of safety mechanisms, including fault population reduction, fault injection, checking and classification, and collection of metrics. We then move on to show how formal can be used to perform each of these tasks in the context of a TMR safety mechanism. Finally we compare formal results and run time against those obtained using dynamic simulation techniques, and show how formal is able to minimize the analysis effort required.

Keywords— ISO 26262; fault analysis; formal verification, property checking, single event upset, formal verification of safety mechanism

I. INTRODUCTION

One of the cornerstones of ISO26262-compliant design is the inclusion of additional logic to implement a “safety mechanism” whose purpose is to ultimately guarantee safe behavior of the given system in the face of

DVCon Europe, October 2016

*10.2, Fault Proof: Using Formal Techniques
for Safety Verification and Fault Analysis*

<http://events.dvcon.org/events/browseproceedings.aspx?confid=211>

Summary

- Formal apps enable any engineer to use formal's power w/out having to learn formal
- Formal apps solve focused verification challenges, from Coverage closure and design checking to path analysis
- Verifying safety and flight-critical systems' vulnerability to transient (and persistent) logic faults is mandatory
- Simulation-based approaches are not exhaustive, creating some risk of bug / fault escapes
- Exhaustive SEU/fault verification with an automated, formal-based analysis greatly increases the quality of results and reduces risk

Mentor[®]
A Siemens Business

www.mentor.com