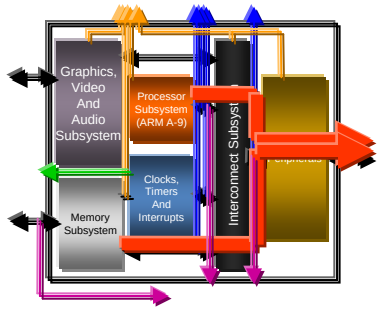


SoC-Level Formal Verification

Verification Conference November 2011
Lawrence Loh
VP of Worldwide Applications Engineering
Jasper Design Automation, Inc.

Formal Verification and SoC-Level Verification

- Formal verification
  - Traditionally operates at block level
  - Focuses on exhaustive verification of functionalities
  - Performed by designers or DV engineers of the particular design
- SoC-level verification
  - Performed at subsystem or system level
  - Focuses on how blocks work together in the system, rather than the functionality of a particular block
  - Performed by system engineers that may not be familiar with the individual components of the system

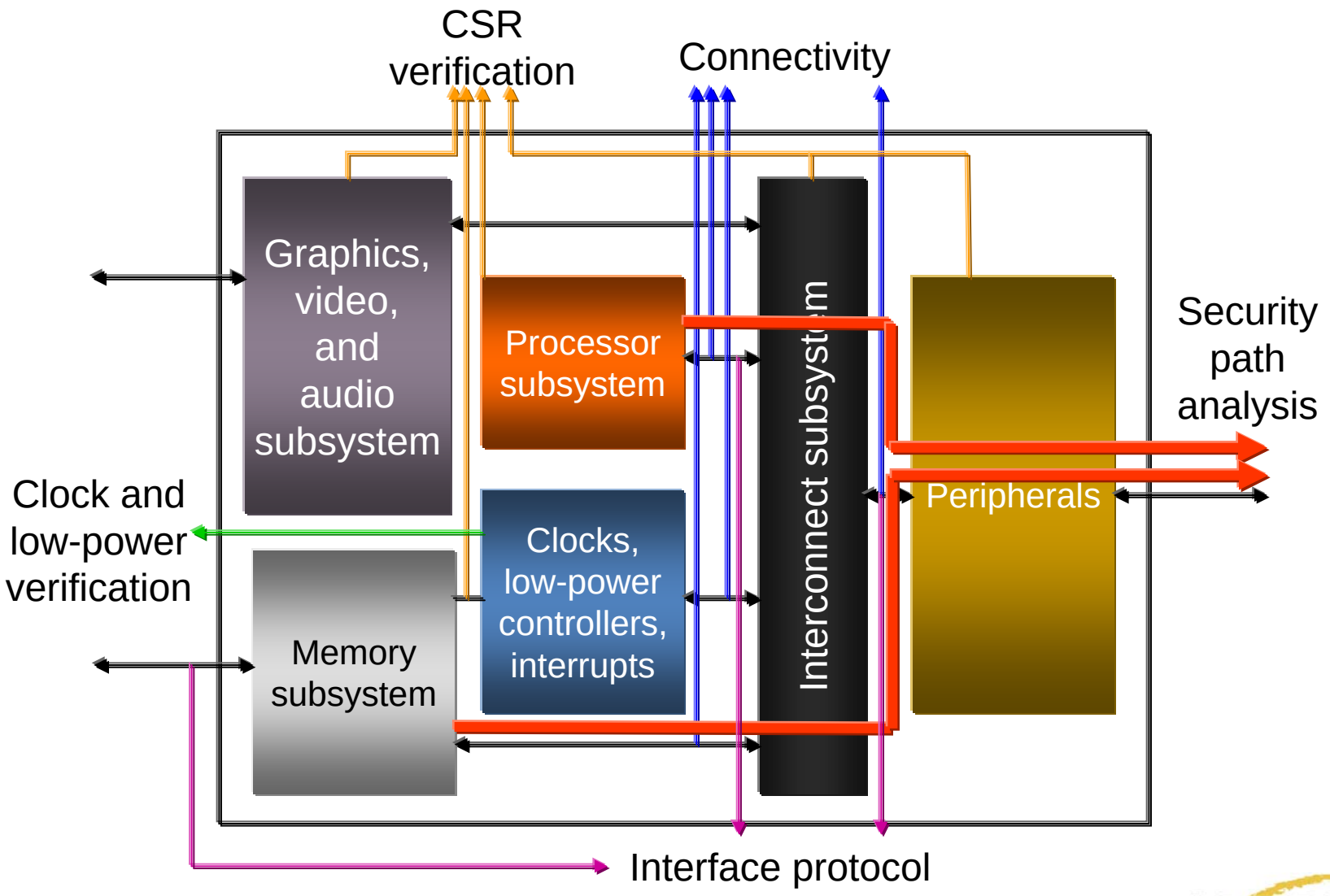
Potential Roadblocks for Traditional Formal

- Capacity
  - Full system may be millions (or 10's of millions) of gates, which is considered beyond most normal formal capacity
- Setup for verification
  - SoC-level verification environment is often very involved and stimuli are generated at a much higher level
- Applicability
  - Many tasks in system-level verification may seem unrelated to what traditional formal is achieving
- Expertise

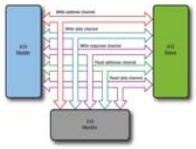
SoC-Level Verification – "Scalable" Formal

- Verification planning
  - What are the components and functionalities that need to be verified above and beyond what block-level verification has done?
  - Contract between two or more IPs (blocks) - how to ensure that adjacent IPs work together properly (protocol, functional dependencies, etc.)
  - System-level functionalities such as low-power schemes, clocking structures
  - Potential automation to reduce human error and human time investment
  - Mission critical functionalities
- IP comprehension
  - Key challenge when the integrator is not familiar with the IPs
  - Even though verification at this level is black-box verification, debugging failures requires some understanding of IPs
  - Explore potential paths through the system (security, interrupts, etc.)
  - Explore power-up sequence, clock structures, low-power schemes, latencies
- Automation and verification on selected items
- System-level debugging, assisting emulation, prototyping

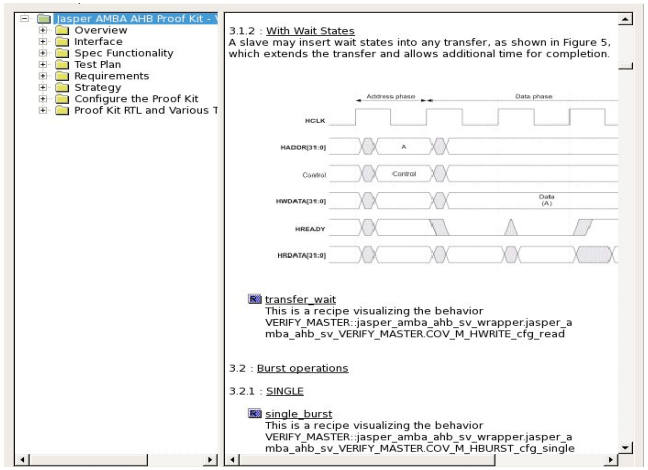
Example of an SoC



Interface Verification

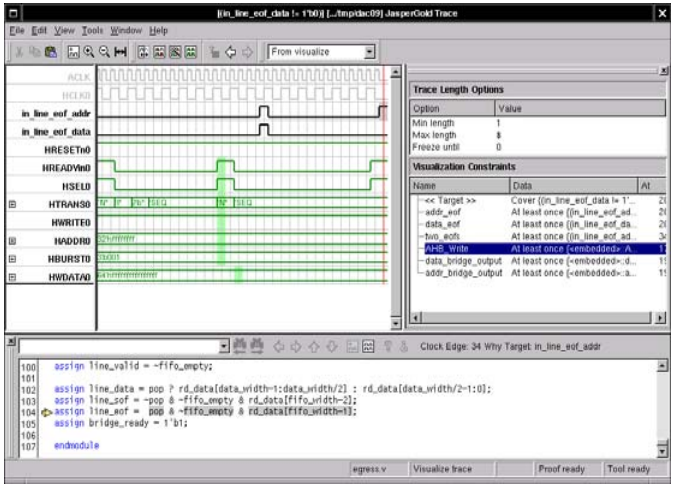


- Protocol certification
  - Protocol checker to allow exploration possibilities on the specification
  - Ensure full-compliance, or expose limitations
- Configurations and setup
  - Automate the connection and setup of environment
  - Protocol checker can serve as the setup for other verification tasks



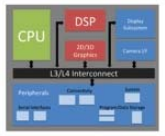
Protocol spec

Comprehend and certify the Interface



Design behavior

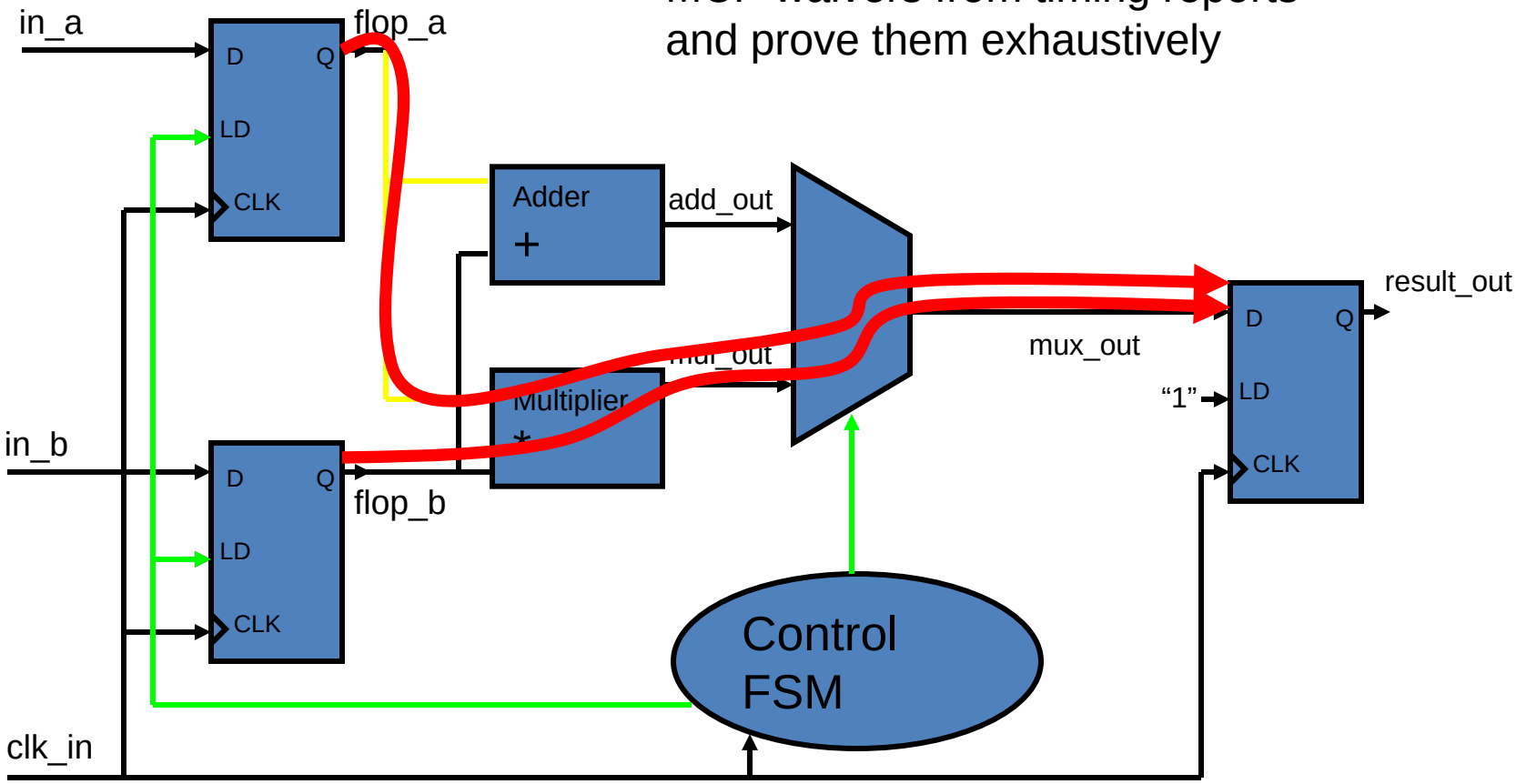
SoC Integration Solution – Some Examples



- Automated register verification
  - Prove data integrity of register fields and reset values
  - Ensure non-accessible register (such as security-related registers) cannot be accidentally accessed
- Multi-cycle path verification
  - Accurately verify multi-cycle path waivers
- Chip-level connectivity
  - Exhaustively verify that RTL matches connectivity definition (spreadsheet, IP-XACT, etc.)
- Security path analysis
  - Verify that security content does not have undesirable path to outputs
  - Explore those paths to either rule out the possibility or determine how to block the paths

Multi-Cycle Paths in a Design

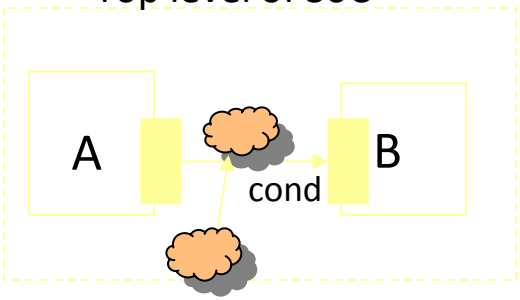
Automatically confirm validity of MCP waivers from timing reports and prove them exhaustively

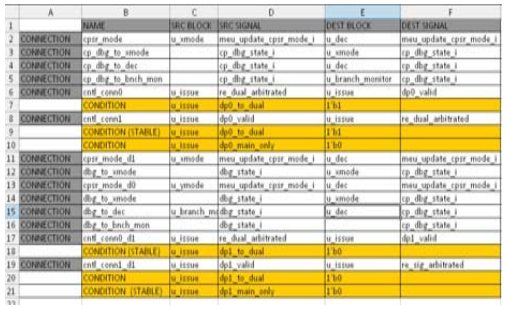


JASPER design automation

Chip-Level Connectivity Checks

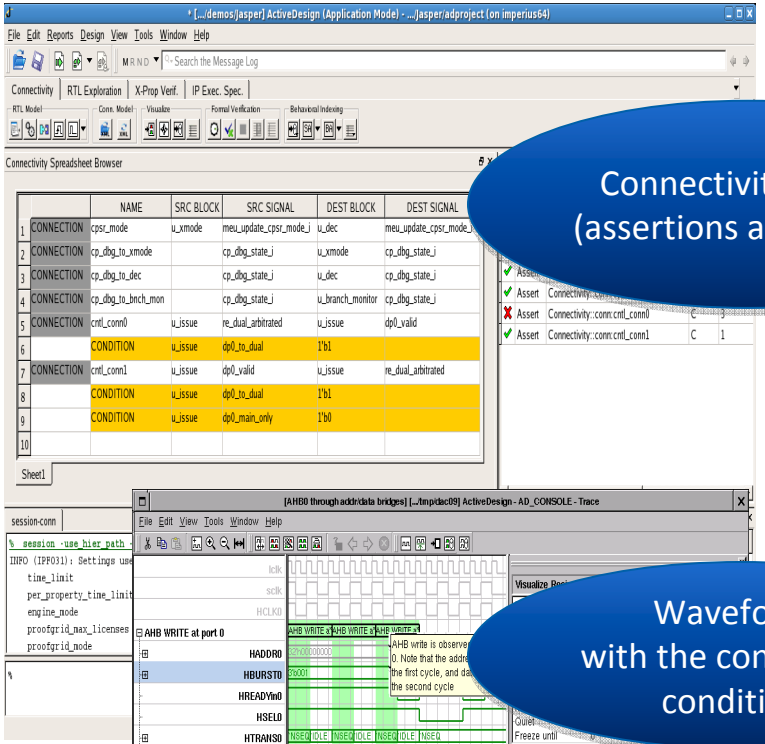
Top level of SoC



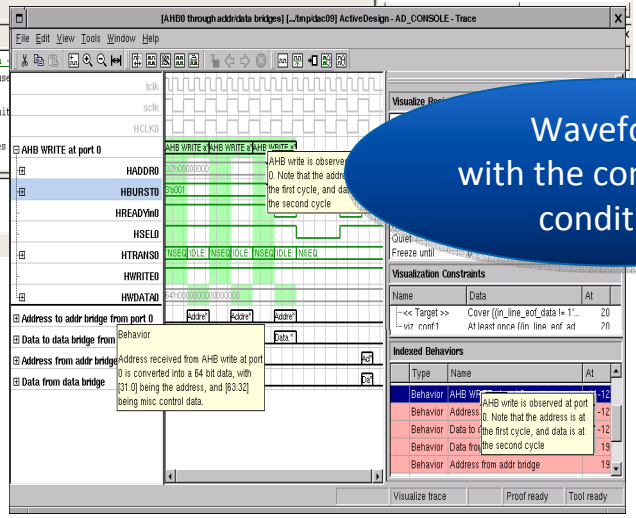


Connectivity map

RTL



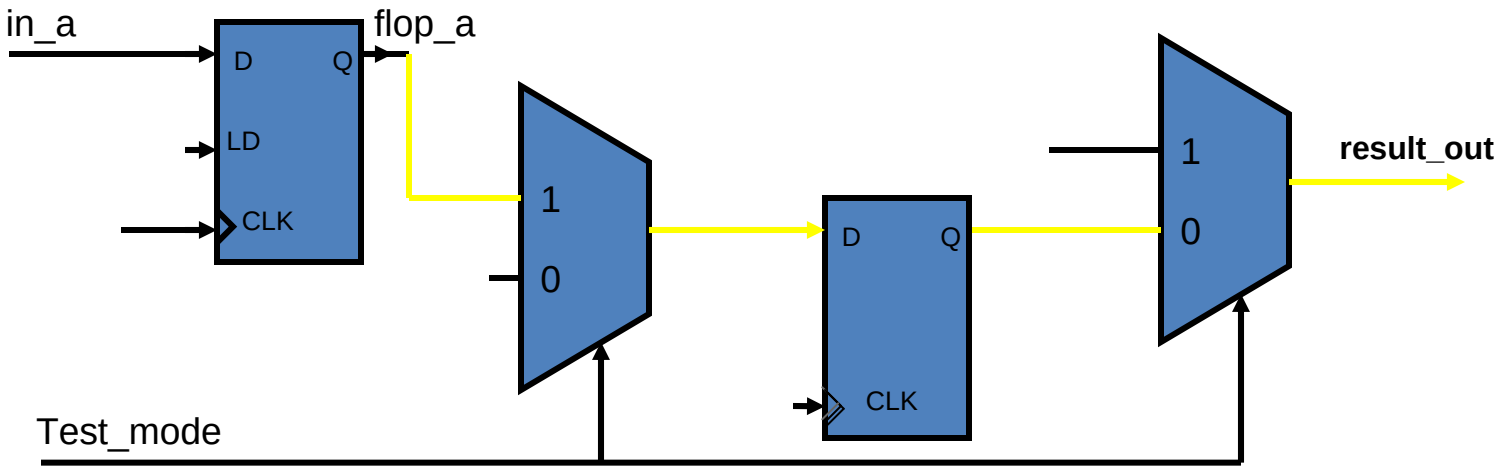
Connectivity proofs
(assertions and covers)



Waveforms
with the connectivity
conditions

JASPER design automation

Security Path Analysis



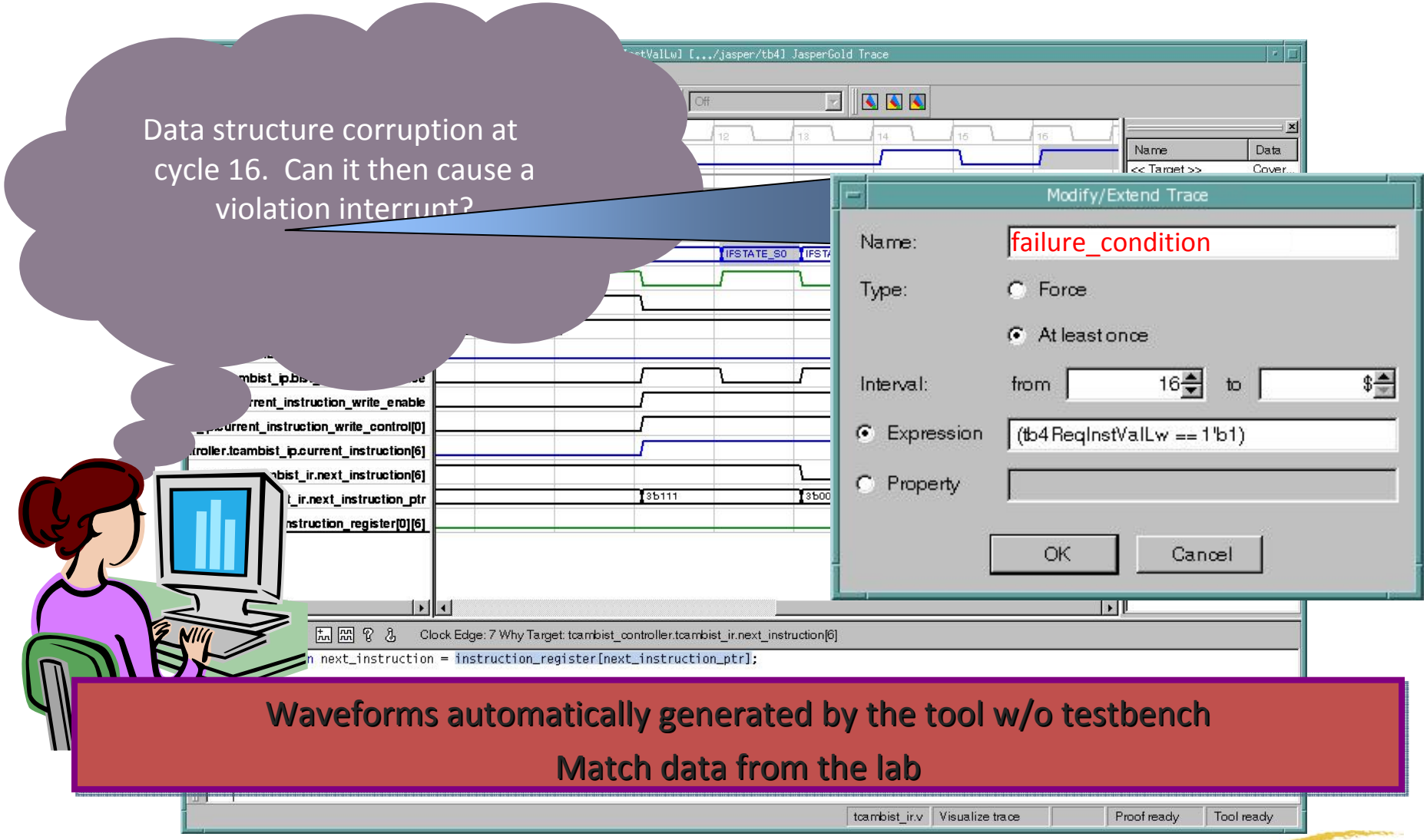
Toggling Test_mode accidentally creates a path from in_a to result_out

- Analyze all specified security points and areas
- Find all paths from those areas to SoC outputs
- Filter out allowable paths (e.g., through encryption block)
- Explore those paths to add further exceptions
- Confirm whether paths have been successfully blocked

Lab Debug Solution

- Failures observed in the lab are difficult to debug
  - Emulation, rapid prototype, post-silicon
  - Limited visibility
  - Failure signature matching
  - Reproduce failing scenario in the lab
- Root-cause isolation
  - Isolate root-cause of the bug
- Candidate-cause elimination
  - Accurately eliminate false candidate blocks and scenarios
- Validation of fixes before re-spin
  - Exhaustively verify that fix does not introduce new bugs

Lab Debug Flow



Waveforms automatically generated by the tool w/o testbench

Match data from the lab

JASPER design automation

Clock Verification

- Clock routing
  - Derived and generated clocks
  - Clock connections
  - Clock sources
- Clock-glitches verification
  - Caused by instability of clock-enable signals during the sampling edge
  - Propagation delays and input delays of the combinational logic causes the clock-enable to have glitches
  - Most CDC tools check for structural issues
  - Functional checks are difficult with simulation due to the timing delays
  - Checking waivers will involve checking that even though the hard CDC rules cannot/are not followed, the clock is glitch-free
- Additional clock rules
  - Clock-enabling/disabling

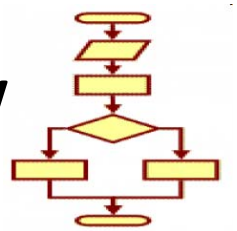
Power-Up/Down Sequence

- Ensure proper power-down sequence
  - For each power domain, follow proper sequence – isolation, turn off clock, turn off power
  - Check for hazards – turn off power before isolation, etc.
- Ensure proper power-on sequence
  - Similarly, follow proper sequence – turn on power, turn on clock, remove isolation
  - Do not start normal operation until full power-on sequence
- Other aspects of power up/down logic
  - Performance expectation – how long it takes to power on/off
  - Wake-up interrupting power-down and vice-versa

Overcoming Roadblocks

- Capacity
  - Full-chip verification (connectivity, CSR) structurally involves most of the chip
  - Functionally, the task can effectively be divide-and-conquer but achieve full verification by post-processing the data
  - Each application needs something more specific to address capacity
- Complexity and sequential depth
  - Path analysis (MCP, security path) and lab debug are not only complex but usually takes many cycles
  - Using intermediate information (intermediate points for path, potential intermediate events) to optimize formal algorithm and under-the-hood abstractions
- Setup and applicability
  - Automation allows reuse on subsequent SoC, or ongoing regression
  - Replacing items on testplan
  - “Hide” formal so that users do not need to be experts in formal

Formal as Part of SoC Verification Flow



- Targeting specific areas
  - Identify challenging areas even though it is not a traditional formal application
  - Mission-critical functions that are worth the time and effort
- Potential quality and productivity gain
  - Areas with no good solution
  - Areas where solution can be better and faster – automation, exhaustive nature of formal, saving human time investment
- Challenge us to come up with the technology and methodology
  - Good applications stir good technological innovations
  - Combination of methodology and technology
  - Often targeted technology enhancement is possible for specific application