Methodology for increasing continuous integration throughput

DVT / IC Verification Solutions

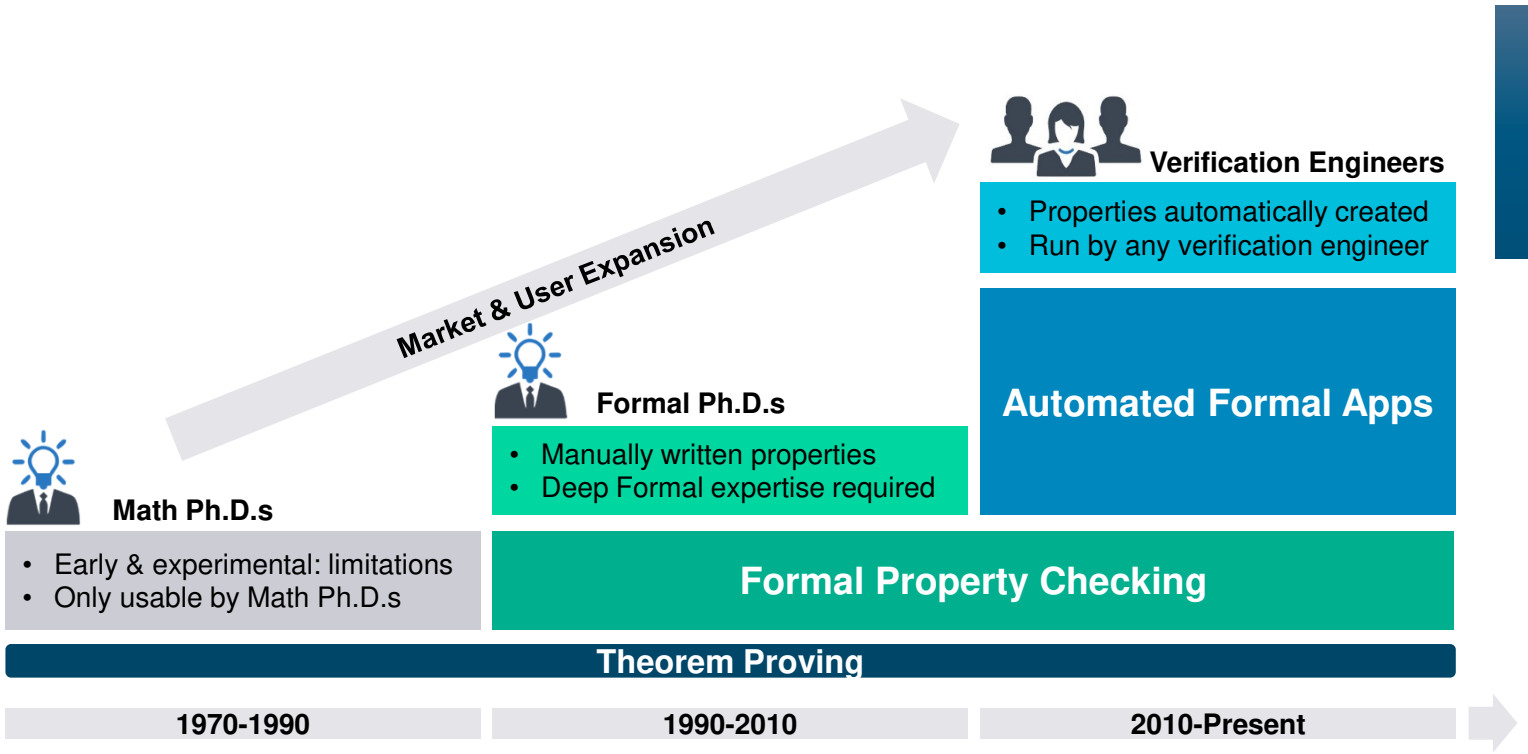
SIEMENS

Agenda

- Introduction:
  - Using formal technology for RTL inspection
  - Automation considerations
- Maximizing continuous integration throughput:
  - Sanity, Deep, Soak/Targeted tests
  - QLNW
- Integrating formal into your CI flow
  - Jenkins – Interface
  - Results
- Summary

Automated apps democratize Formal and drive growth

Apps expand the market from only formal experts to all verification teams



76%

ASICs require 2 or more respins

9,4% CAGR

Automatic formal verification

3,6% CAGR

Formal property checking

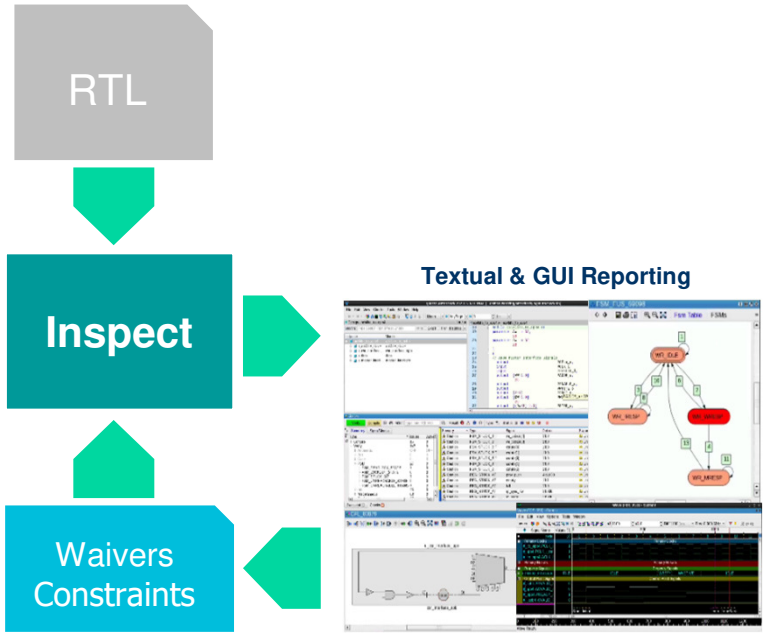
Source: Wilson Research Group and Siemens EDA, 2022 Functional Verification Study

SIEMENS

Advanced Linting / Autochecks
Eliminate common issues without noisy logs

Advantages

Early verification with no test bench or assertions needed
Industry leading performance
Fully automatic, no manual assertions or stimulus



Structure (Easy Lint)	Safety Checks (Assertion Synthesis)			Activation (Coverage)
Mismatch/port /wire	Runtime Errors	Sim-Synth Issue	Safe Function	Reachability
Signal trunc / no sink	Array / Range checks	Full case	Neg / 0-Div exp, rem Arithmetic overflow	Dead code checks
Sensitivity list issues	Function without return	Parallel case	X / Z resolution	Stuck signal (toggle test)
Unused signal / param	Signal domain checks	Write-write race detect	Arithmetic shifts	FSM trans and states

SIEMENS

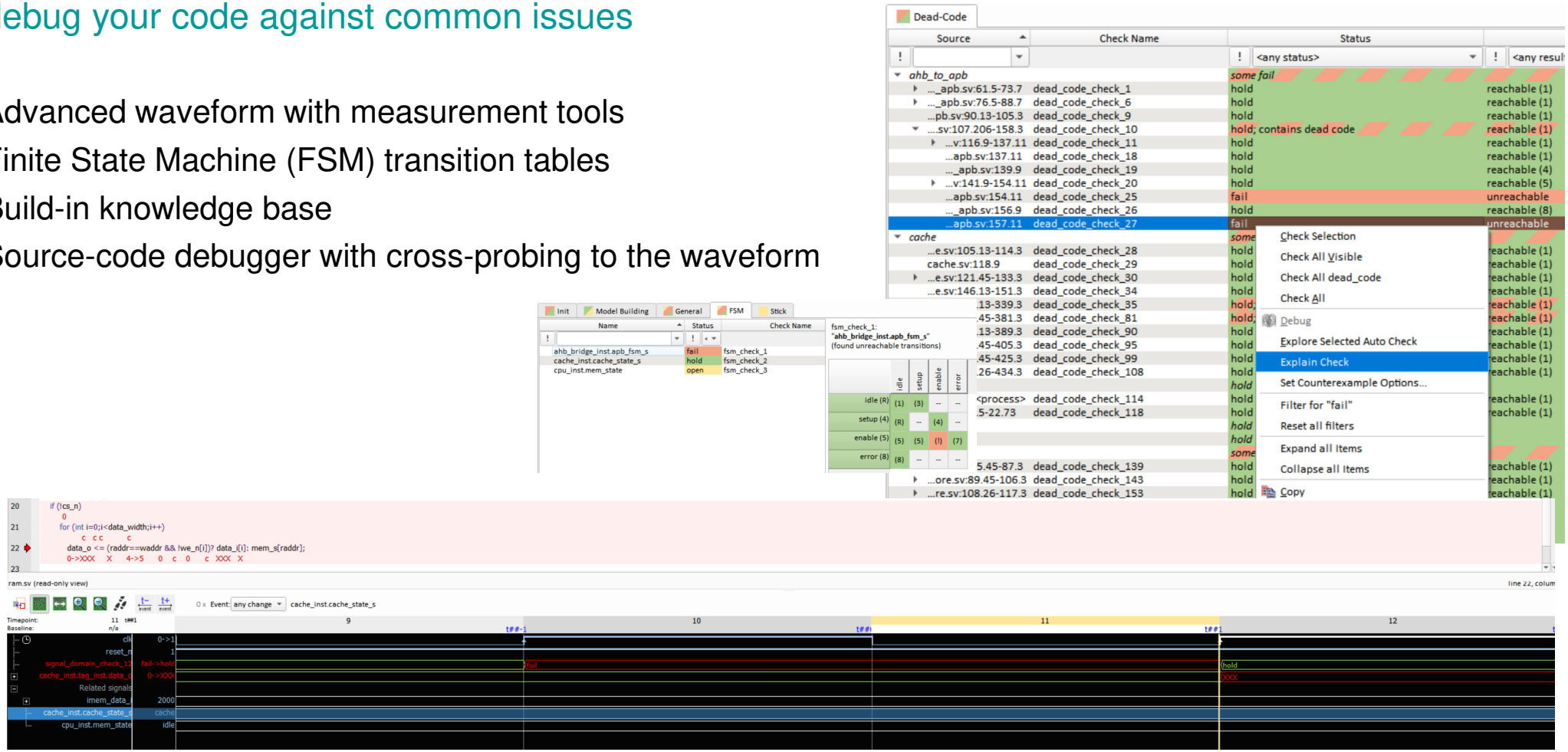
Design inspection debugger
debug your code against common issues

Advanced waveform with measurement tools

Finite State Machine (FSM) transition tables

Build-in knowledge base

Source-code debugger with cross-probing to the waveform



SIEMENS

Maximize Continuous Integration throughput

- Continuous Integration (CI):
  - Part of the D&V strategy of many organisations.
  - Automated building, automated verification and automated status reporting.
  - Highly ideal application for autochecks as no manual intervention is needed.

- Regularly scheduled auto-checks:
  - High Value in detecting RTL issues during code churn
  - Hey - but it‘s formal ?

- Considerations:
  - Runtime: too long? Will it converge?
  - Do I need to modify constraints?

Agenda

- Introduction:
  - Using formal technology for RTL inspection
  - Automation considerations
- Maximizing continuous integration throughput:
  - Sanity, Deep, Soak/Targeted tests
  - QLNW
- Integrating formal into your CI flow
  - Jenkins – Interface
  - Results
- Summary

SIEMENS

Continuous integration pipeline

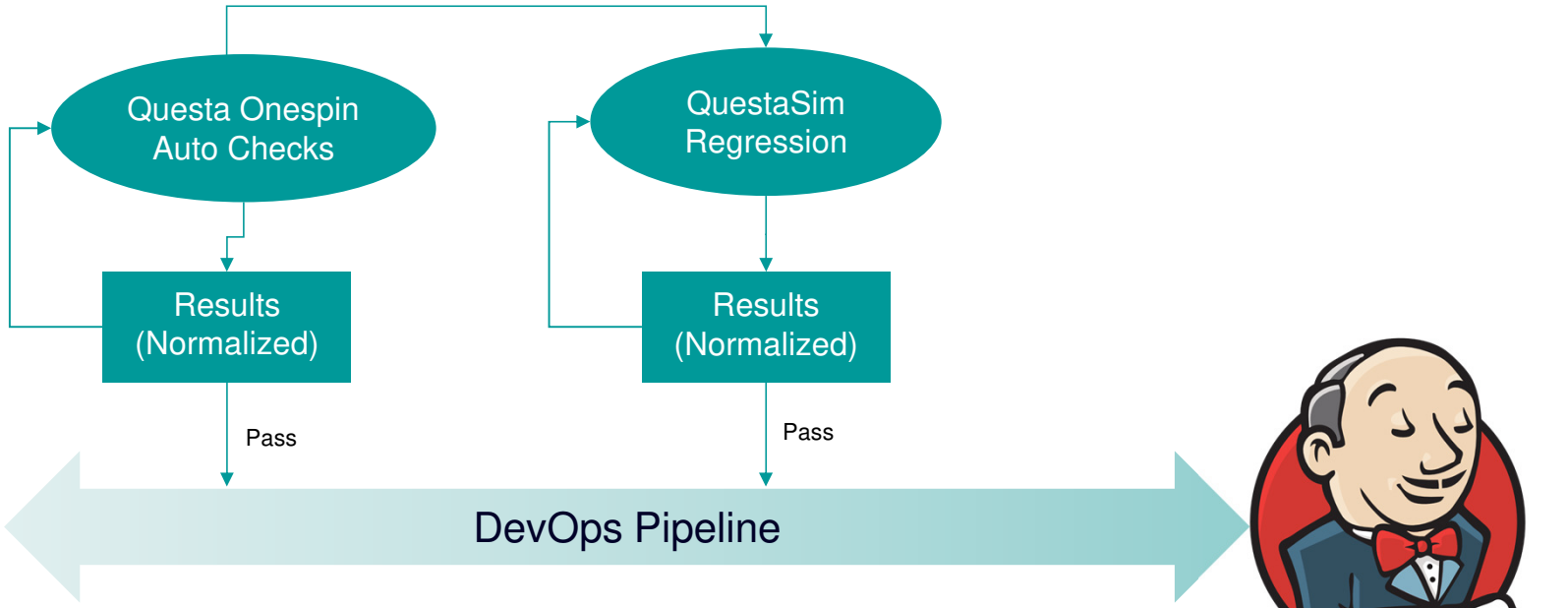


Image from Jenkins Project
https://jenkins.io/

SIEMENS

Autochecks - Prioritization Techniques

Common criterions for choosing scheduled auto-checks:

- Severity of failure:
  - Fatal, sim/synth, etc
  - FSM deadlock analysis
- Runtime:
  - Deadlock analysis
  - Initialisation checks
- Accuracy (ie low chance of false negative)
- End use (ie Designer, Verification Engineer, QA Engineer, Manager)

Autochecks effort based prioritization

Optimal order to perform autochecks:

1. Higher value, Low runtime
2. Higher value, High runtime
3. Lower value, Low runtime
4. Lower value, High runtime

For greater granularity:

1. High value, Low runtime
2. Medium value, Low runtime
3. High value, Medium runtime
4. Medium value, Medium runtime
5. Low value, Low runtime
6. Low value, Medium runtime
7. High value, High runtime
8. Medium value, High runtime
9. Low value, High runtime

End use

Design Engineer

- Logical coding bugs
- Checking at each level of integration (ie module, component, sub-system)
- Signoff before handover to Verification Engineer

Verification Engineer

- Integration bugs
- Checking at each level of integration (ie component, sub-system, system)

QA Engineer

- Functional checking at lower levels of integration (ie module, component)
- Structural checking at higher levels of integration (ie sub-system, system)

Manager

- Planning
- Progress/Trend vs timescales

CI Techniques

Sanity, Deep, Targeted/Soak

- Sanity — High value, variable runtime checks
- Deep — High & medium value, variable runtime checks
- Targeted — High & medium value, low noise, variable runtime checks

QLNW

- Quick — High value, low runtime checks
- Lunch — High value, low & medium runtime checks
- Night — High & medium value, low & medium runtime checks
- Weekend — More checks, extend runtime

These all include time limits so that the maximum runtime is known.

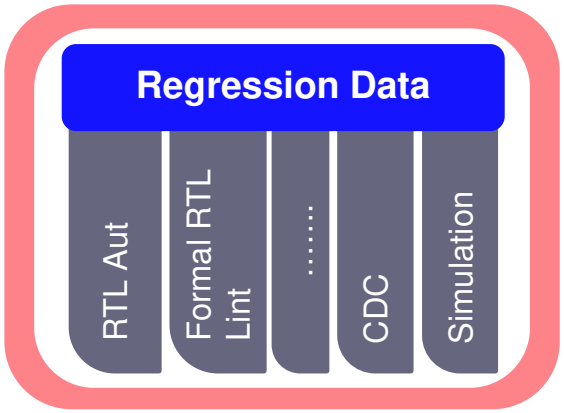
Agenda

- Introduction:
  - Using formal technology for RTL inspection
  - Automation considerations
- Maximizing continuous integration throughput:
  - Sanity, Deep, Soak/Targeted tests
  - QLNW
- Integrating formal into your CI flow
  - Jenkins – Interface
  - Results
- Summary

SIEMENS

Jenkins

- Popular open source Java CI tool
- Server based system running in a Java servlet container such as Apache Tomcat
- User defines tasks to be launched by CI when code is committed

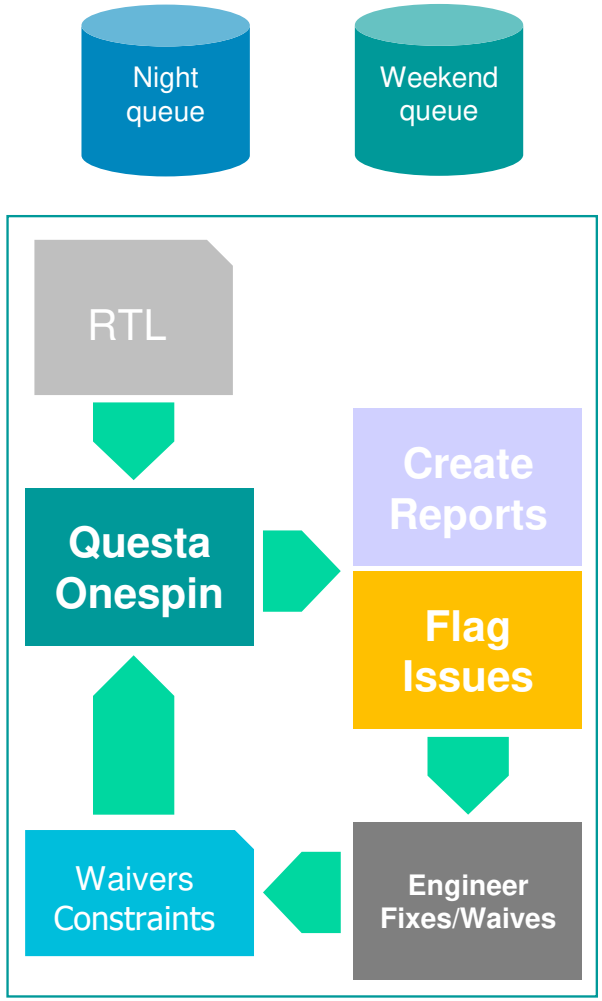


Jenkins flow

- Schedule batch jobs – night/weekend (different checks/runtimes)
- Create plugin to:
  - Collect results
  - Create reports
  - Flag issues/Generate debug traces:
    - create waivers
    - save databases for debug

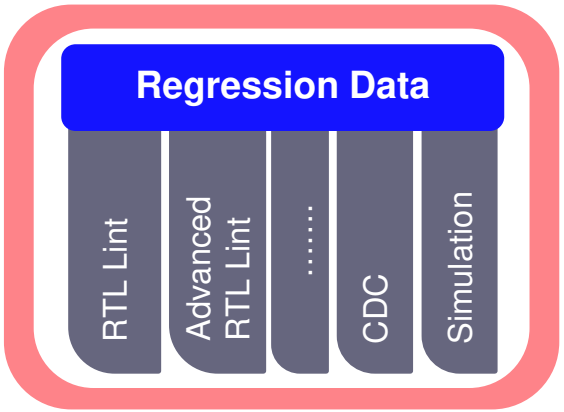
- Questa plugin template can be customized

- Questa Onespin APIs available to:
  - Customise reports
  - Run a choice of QLNW checks
  - Customise runtime / granularity



SIEMENS

What Kind of Data is Needed For Continuous Integration Tools



- **For Jenkins**
  - CSV files containing the information to be analyzed
    - Pass/Fail information
    - Application specific results (attributes & values)
  - UCBD files containing the information to be analyzed
    - Need the Questa VRM Jenkins plug-in
      - Pass/Fail informaiton is stored in UCDB Test status attribute
      - Trendable data should be stored in the UCDB

SIEMENS

Report customisation commands

report_result [options] -signoff {-subchecks subchecks} {consistency-checks}}

Description

Reports the current verification status in a human readable format. If no checks/objects are given explicitly, then information about all respective checks/objects is reported.

get_consistency_info [options] consistency_check_identifier

Description

Returns attribute values of the specified consistency check(s). If no filtering option is specified, then the values for all check attributes are returned.

Options

-branch_path Yield hierarchy if check refers to a specific module

-instance Yield instance path

-module Yield module

-number Yield check number

-result Yield check result

Case study

Onespin customer develops a sanitization Jenkins flow.
Results were logged daily and so differences can be analysed.
Many design engineers now run Initialisation and FSM checks before they check any modifications in.

Decided to build 3 Jenkin jobs to check:

- Design clocks are recognised
- Reset sequence is recognised
- Registers are initialized
- dead_code
- model_building: ready for formal property checking

Case Study – Sanity Checks enabled

Type	Description	Verilog	VHDL
array_index	checks for out-of-bounds array accesses	X	X
no_return	checks for functions without return	X	X
shift	checks if shifting can yield zeros	X	X
Signal_domain	signal domain checks	X	X
write_write	checks for write-write races	X	X
full_case	Synthesis full-case pragma check	X	
parallel_case	Synthesis parallel-case pragma checks	X	
div_zero	checks for division-by-zero		X
neg_div	checks for negative divisors		X
neg_exp	checks for negative exponents		X
range	range checks for subtypes		X
read_write	checks for read-write races		X

Agenda

- Introduction:
  - Using formal technology for RTL inspection
  - Automation considerations
- Maximizing continuous integration throughput:
  - Sanity, Deep, Soak/Targeted tests
  - QLNW
- Integrating formal into your CI flow
  - Jenkins – Interface
  - Results
- Summary

SIEMENS

Advanced Linting / Autochecks for Continuous Integration

- Formal analysis verification techniques add value to Continuous Integration flows

- Questa Onespin Solutions provide automated checks that easily fit into CI:
  - No verification input required – no user written assertions
  - Tool interface allows you to optimise for highest return on your checks:
    - Runtime / Convergence
    - Type of checks and issues you can intercept
    - API to extract relevant information to ease root cause analysis

- Jenkins plugin templates provide a good starting point for your own flow

Disclaimer

Subject to changes and errors. The information given in this document only contains general descriptions and/or performance features which may not always specifically reflect those described, or which may undergo modification in the course of further development of the products. The requested performance features are binding only when they are expressly agreed upon in the concluded contract.