Synopsys Users Group
Europe

Automation of SoC Verification Environment

Case study

Jukka Heikkilä
Applications Engineer, Verification Group, Synopsys

May 2019

CONFIDENTIAL INFORMATION

The information contained in this presentation is the confidential and proprietary information of Synopsys. You are not permitted to disseminate or use any of the information provided to you in this presentation outside of Synopsys without prior written authorization.

IMPORTANT NOTICE

In the event information in this presentation reflects Synopsys' future plans, such plans are as of the date of this presentation and are subject to change. Synopsys is not obligated to update this presentation or develop the products with the features and functionality discussed in this presentation. Additionally, Synopsys' services and products may only be offered and purchased pursuant to an authorized quote and purchase order or a mutually agreed upon written contract with Synopsys.

© 2019 Synopsys, Inc.

Agenda

- **Introduction**
- SW development practices
- Building continuous regression flow
- Example setups
- Summary

© 2019 Synopsys, Inc.

Introduction

Synopsys Users Group Europe

Verification challenges for today's large IP/SoC teams:

- Planning the verfication efficiently
  - Will all the product requirements get verified?
  - How to get the most important features verified first?
- Long verification times
- Complex verification environments prone to human errors
- Utilizing R&D infra efficiently (grid, multicore support)
- Monitoring progress in multi-site teams
- Keeping SoC design and related SW in sync

Well planned, scalable and continuous regression flow with powerfull engines and automation is needed

© 2019 Synopsys, Inc.

Agenda

- Introduction
- **SW development practices**
- Building continuous regression flow
- Example setups
- Summary

© 2019 Synopsys, Inc.

Differences between HW and SW development

- SW verification is faster and more straightforward comparared to HW verification
  - No need to worry about individual clock cycles
  - No infinite parallelism
  - No synthesis issues, netlist simulations, power simulations (UPF)
  - Test can take only few seconds
- HW processes follows typically tightly specified processes (to avoid expensive re-spins)
  - IP/SoC level functional verification can already take from days to weeks
- SW development processes are advanced and keep evolving all the time
  - Linear waterfall model → Flexible Agile model
  - DevOps (Continuous Integration, Continuous delivery, containers)

SW and HW for the same SoC can be verified totally separately

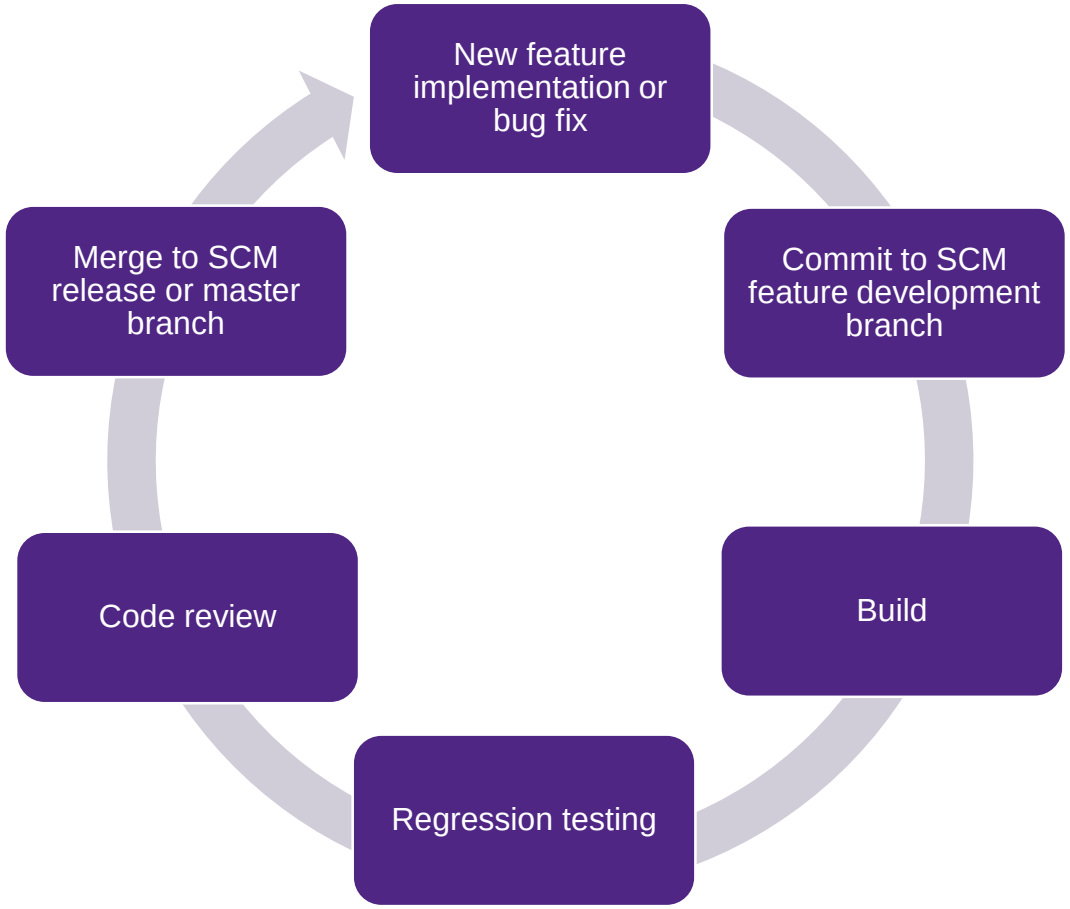
Would it be possible to adopt some best practices from SW world?

Would be even possible to combine the verification flows (SW and HW)?

© 2019 Synopsys, Inc.

What is Continuous Integration (CI)?

- Software development practice requiring developers to check-in source code changes regularly, upto several times a day.
  - Small deviations are easier to debug
  - Each check-in is validated quickly
  - As the code is safe, it can be merged with other check-ins (automatically)
- Jenkins is industry's preferred CI automation tool



© 2019 Synopsys, Inc.

What is Jenkins?

- Automation server used Continuous Integration flow
  - Developed at Sun Microsystems around 2004 (Hudson)
  - Java-based, open source
- Brings automation of CI flow
  - Can trigger builds (and start regressions) automatically if code is checked-in
  - Can also use time-based triggers (cron jobs)
- Runs tools and scripts that are available on the work area
- Usability can be extended by different Plugins
  - Over 1400 plugins available in repository covering many different applications
- Allows different types of job configurations (freestyle, pipeline, multijob)
- Web-based dashboard enabling team and management to get status information
  - Plugins available to represent status of regressions; tables, trend graphs etc.
  - Plugins available for several other collaboration tools (e.g. Jira, Confluence, Git)
- East to set up (run the command java -jar jenkins.war)
- Good for SW testing (integrated built tools, short tests, not much data stored)
- More productized commercial enterprice versions available (Jenkins 2.0 by Cloudbees)

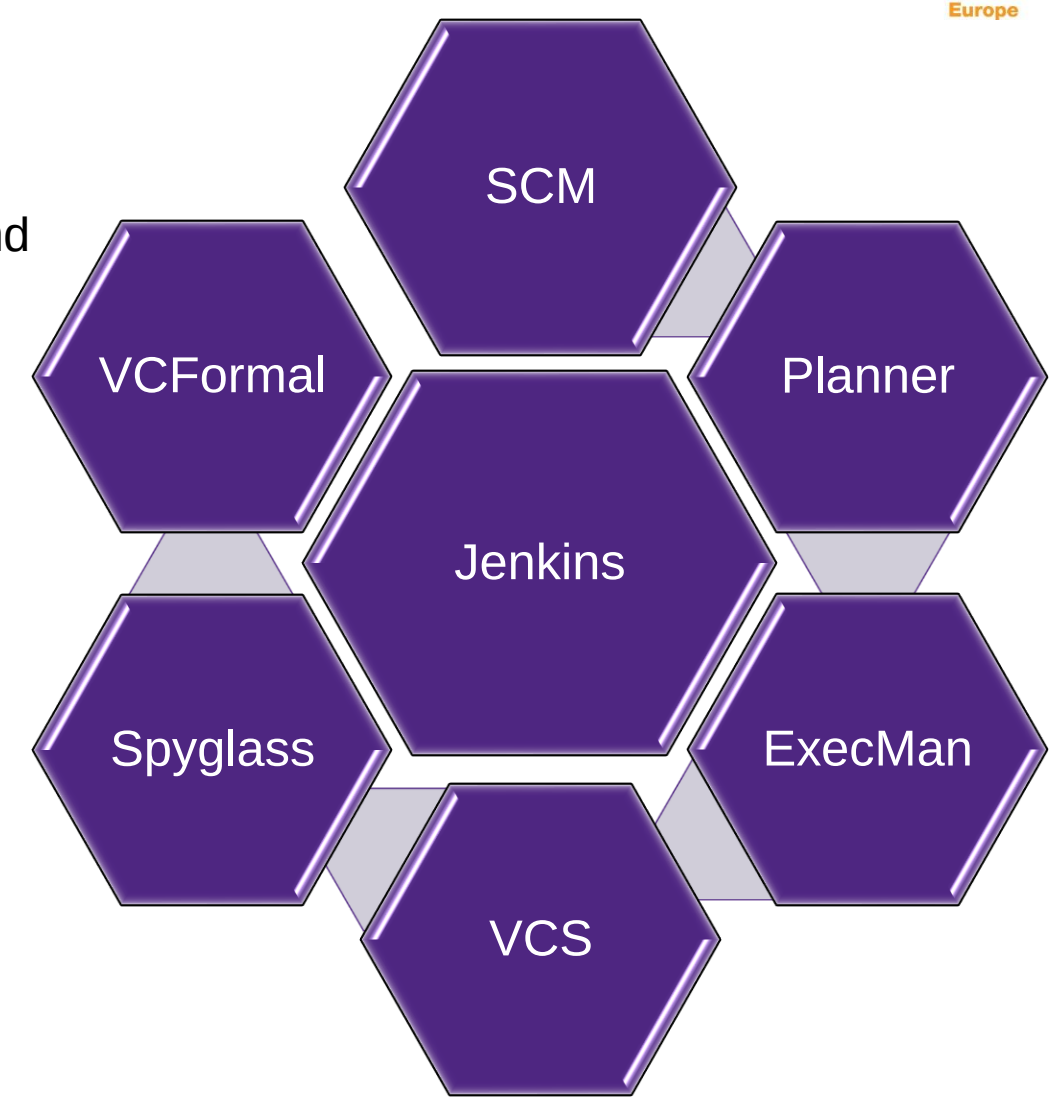
Jenkins

https://jenkins.io

© 2019 Synopsys, Inc.

Jenkins and Hardware Verification

- Jenkins knows nothing about hardware verification
  - What tools to run and under what conditions
  - Need to provide scripts for Jenkins to run compilation and tests
  - Tool can have own plugins to ease the usage and to visualize results
- By default Jenkins can trigger regressions but cannot interpret results
  - Doesn't understand UVM status messages and pass/fail criteria in regression
  - Doesn't know how to execute coverage merging
  - Cannot show status of regression in real time
- Adding parallelism is possible, but not that easy
- There are tools that does these tasks very well and Jenkins can run those, for example Execman



© 2019 Synopsys, Inc.

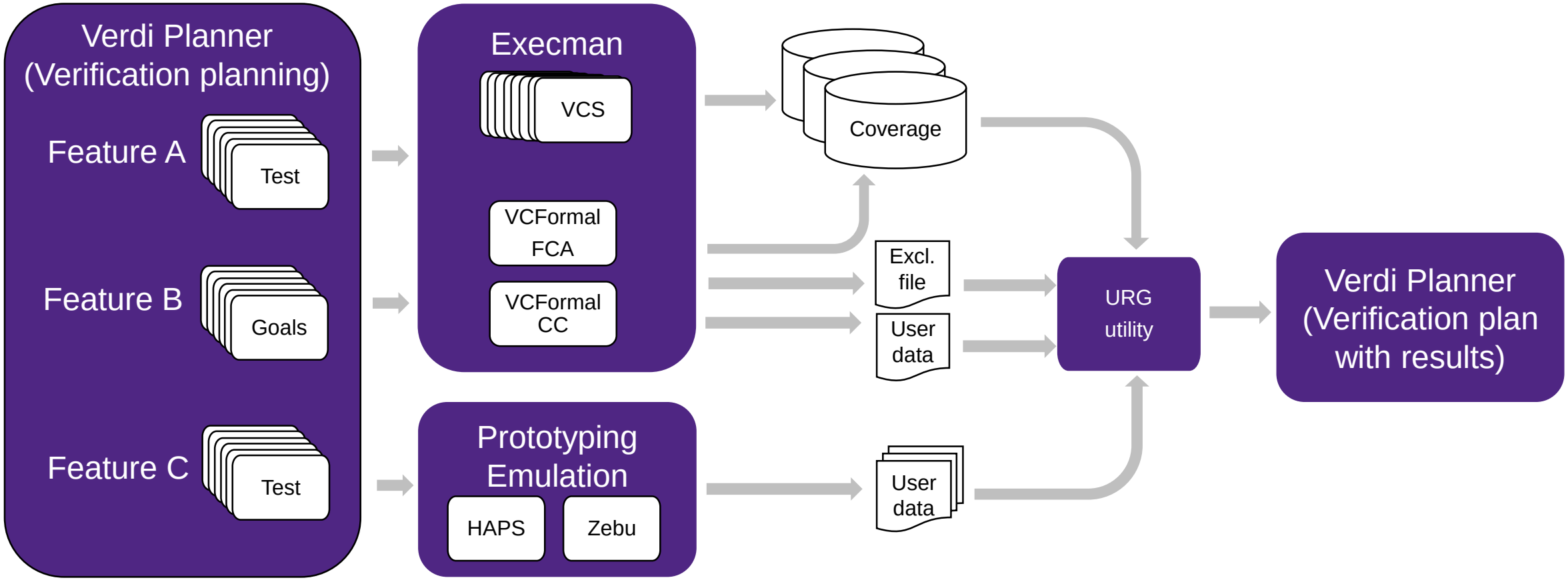
Agenda

- Introduction
- SW development practices
- **Building continuous regression flow**
- Example setups
- Summary

© 2019 Synopsys, Inc.

Feature driven verification

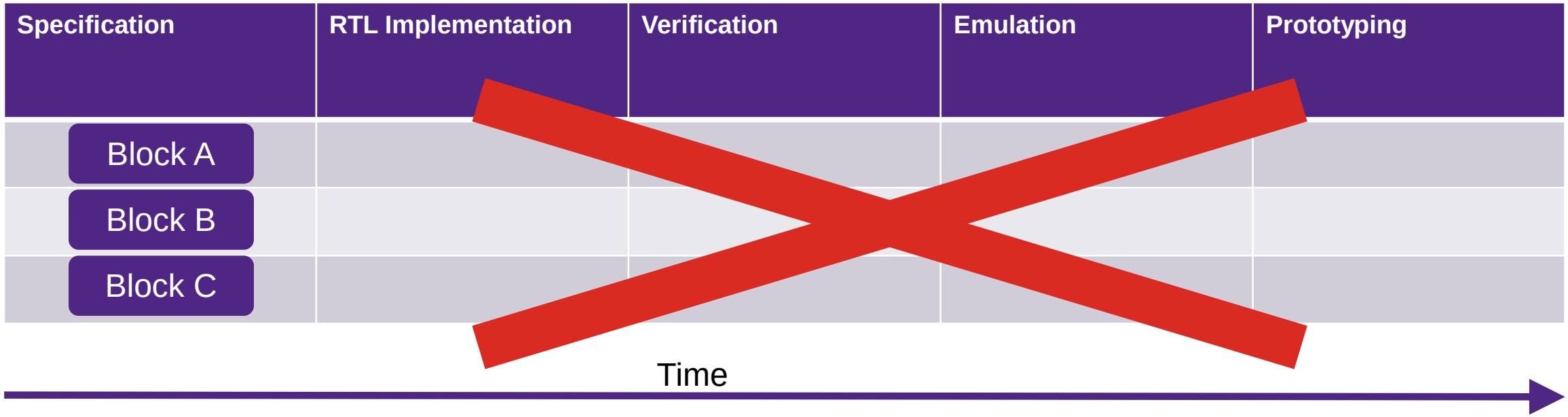
Example SoC verification flow setup



© 2019 Synopsys, Inc.

Traditional verification flow

RTL implementation centric approach



All the parts of the SoC need to be ready before moving to next step in the project

© 2019 Synopsys, Inc.

Feature driven verification flow

Doing the most important things first

Specification	RTL Implementation	Verification	Emulation	Prototyping
Feature A				
Feature B				
Feature C				

Time

<u>Minimum set of design</u> need to be ready before moving to next step in the project

© 2019 Synopsys, Inc.

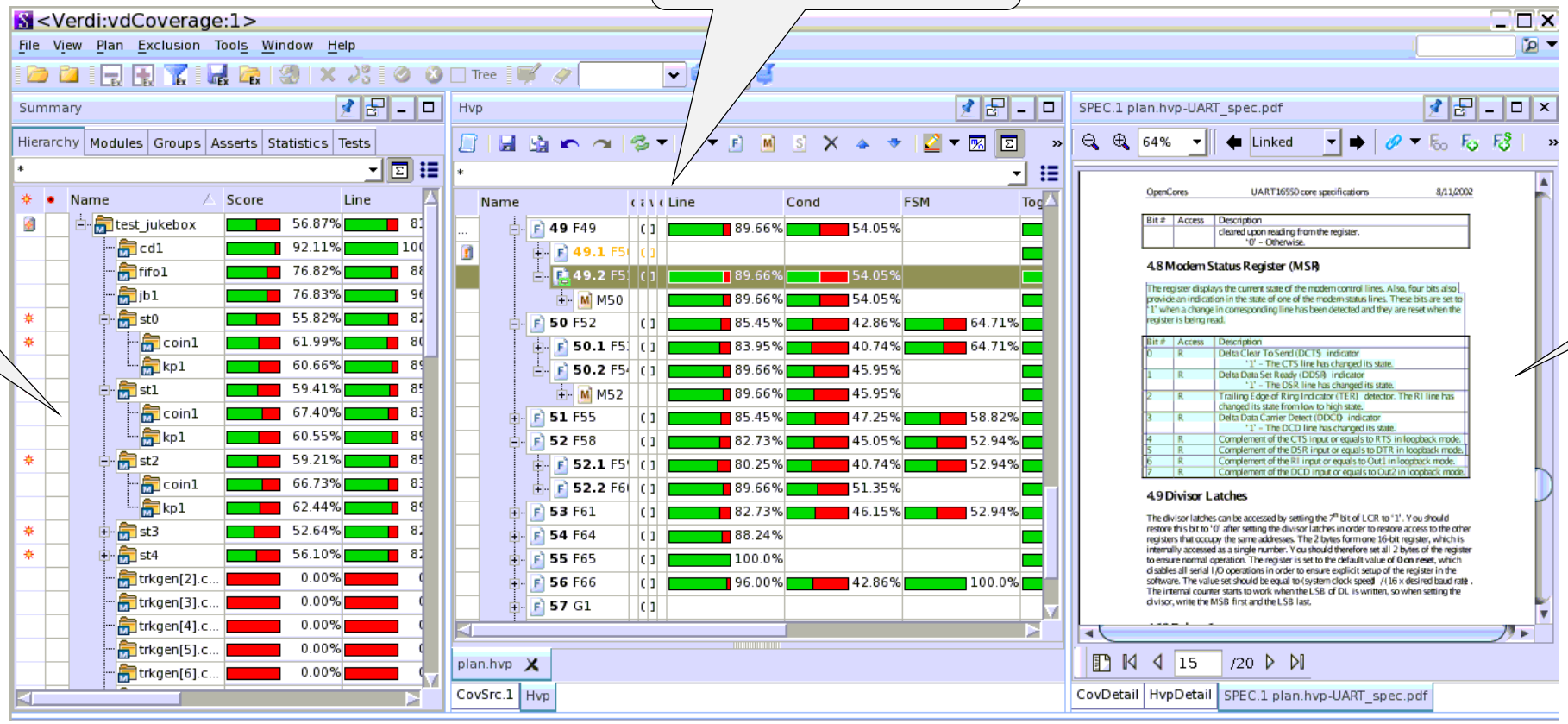
Integrated Coverage Aware Planning with Verdi

Synopsys Users Group Europe

Verification Plan

Coverage

Req. Spec

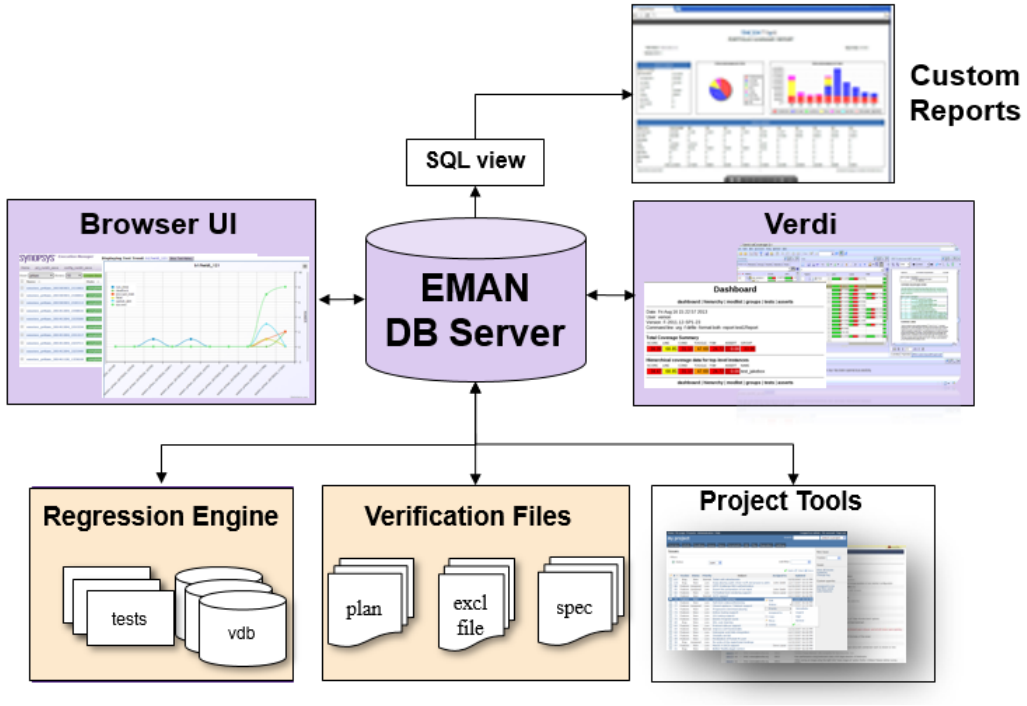


Traceable Verification Across Flow and Tools for Faster Verification Closure

© 2019 Synopsys, Inc.

What is Execman?

- Server-client solution providing execution management for SoC verification
- Can be executed from terminal shell or Verdi Planner
  - Launches and monitors jobs in server farm
- Deploys SQL database
  - Efficient for collecting lot of data
  - Provides APIs for custom solutions
- Real time monitoring through Verdi or Web server
- Supports verification plan with custom metrics
- Random test order option
- Support for running long test set in batches
- Automatic debug runs
- Multiple builds and build specific test sets
- Configurable log file parser to detect all the bugs
- ...

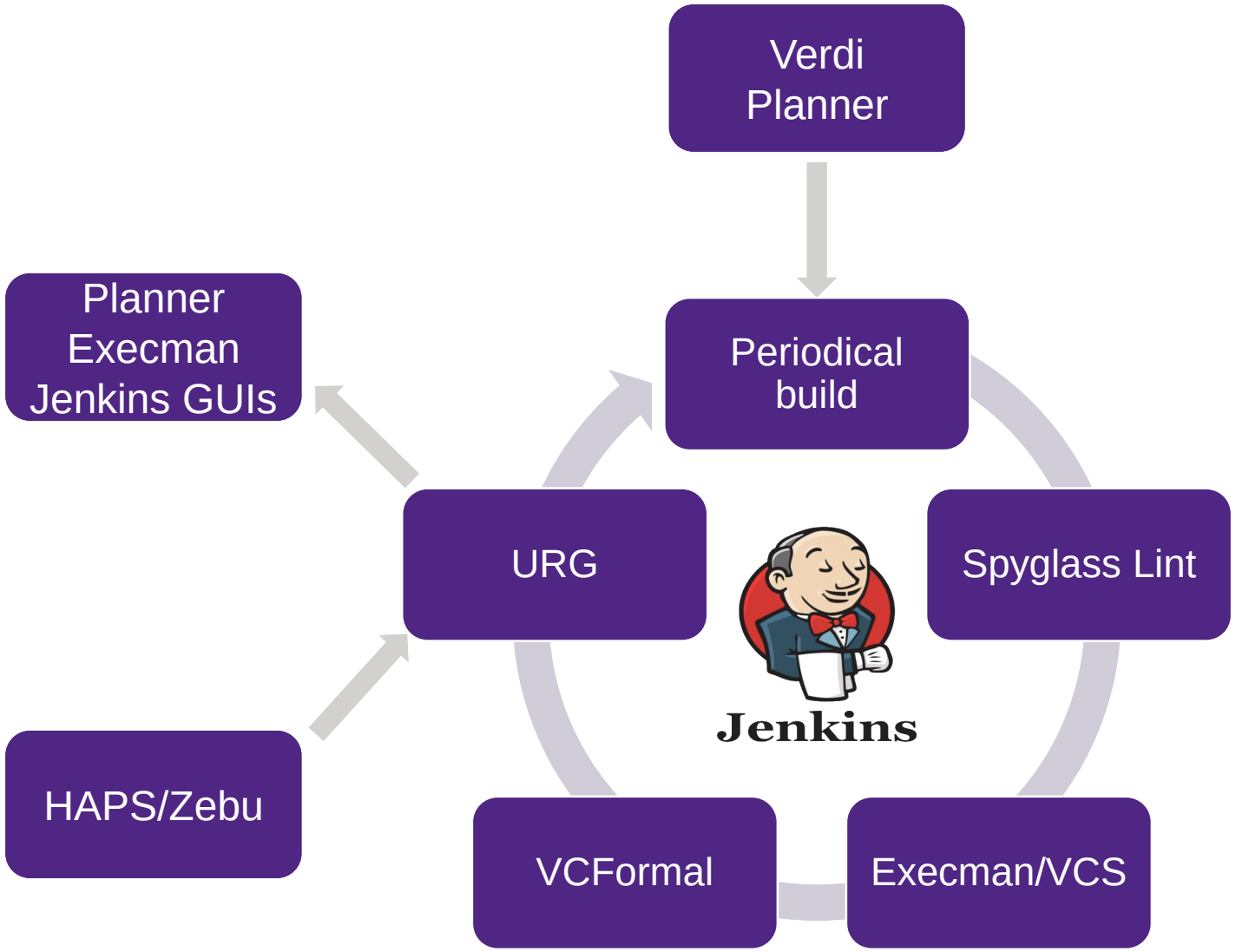


© 2019 Synopsys, Inc.

Continuous regression flow

SoC level example

- Jenkins pipeline
- Time triggered Jenkins project (“nightly build”)
- Possible to run full test set in smaller batches

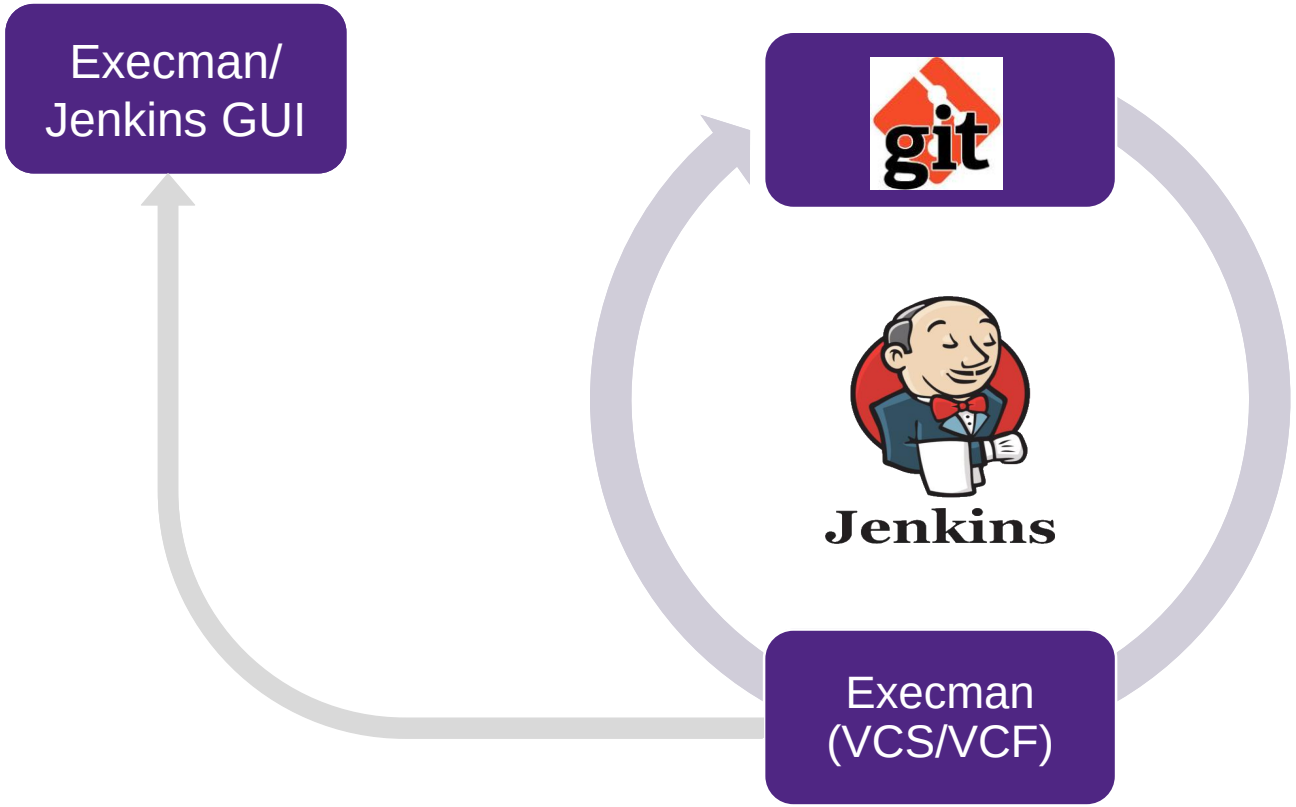


© 2019 Synopsys, Inc.

Continuous regression flow

Block level example

- Jenkins triggered from source code management
- ”Smoke set” simulation
- Execman Jenkins Plugin



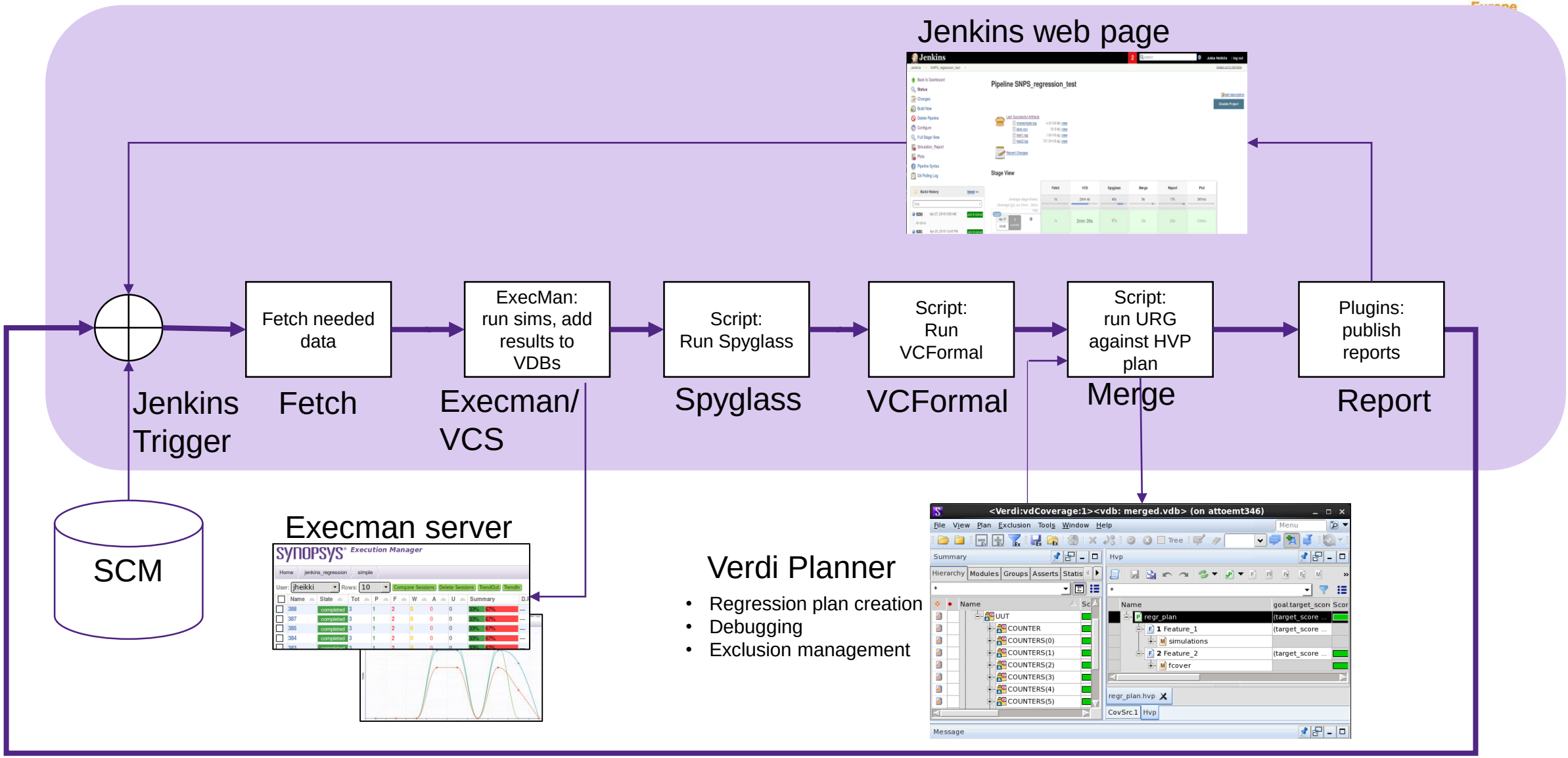
© 2019 Synopsys, Inc.

Agenda

- Introduction
- SW development practices
- Building continuous regression flow
- **Example setups**
- Summary

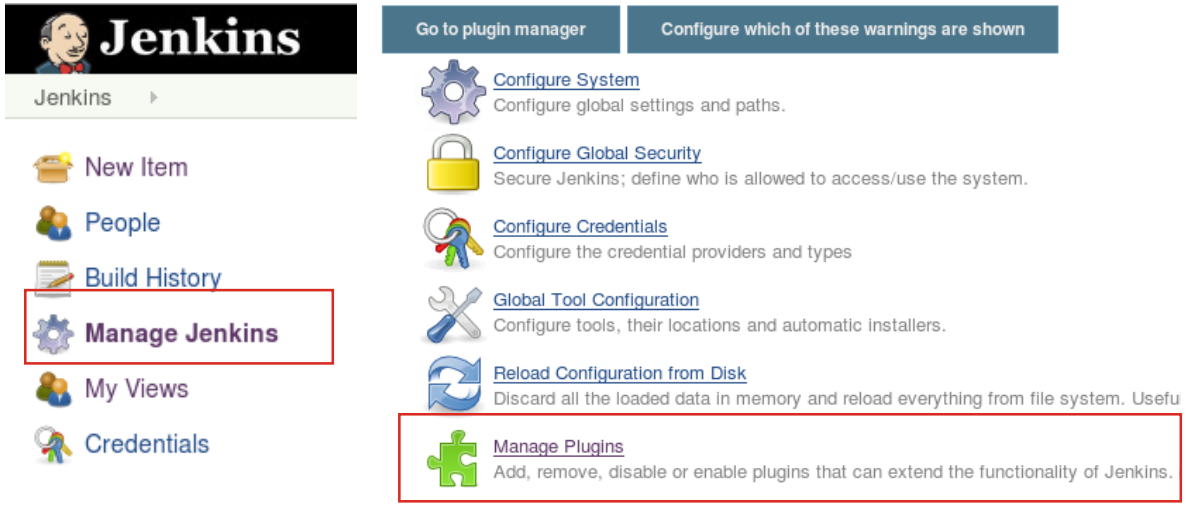
© 2019 Synopsys, Inc.

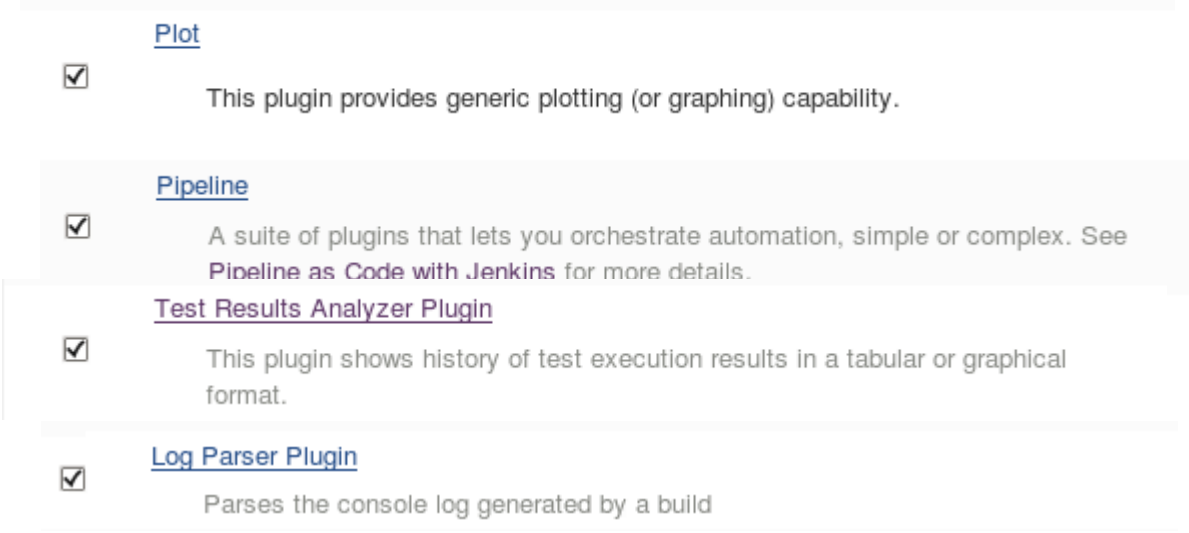
Jenkins pipeline Example



How to install required plugins

1. Manage Jenkins -> Manage Plugins
2. Search required plugins
3. Select and install



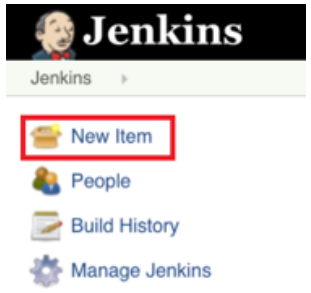


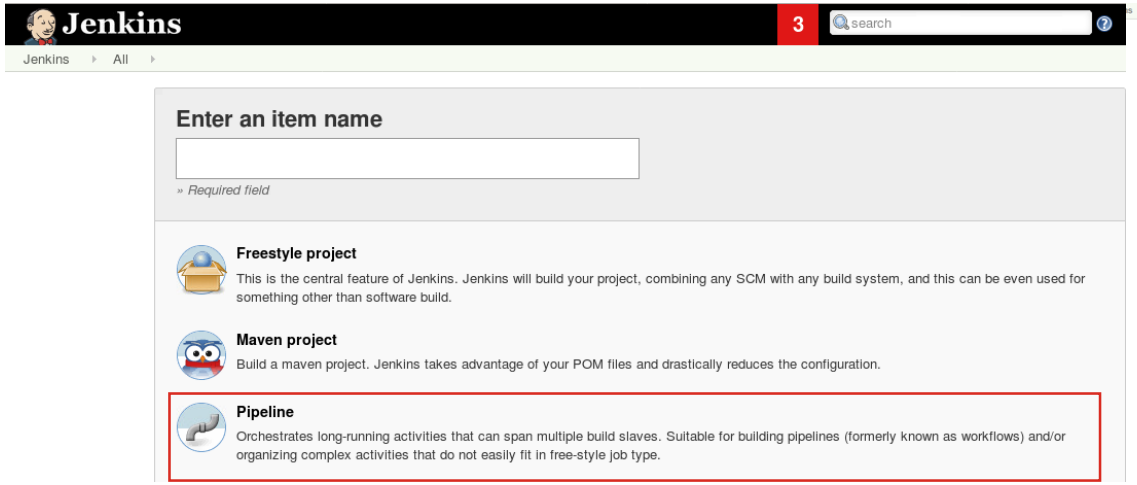
© 2019 Synopsys, Inc.

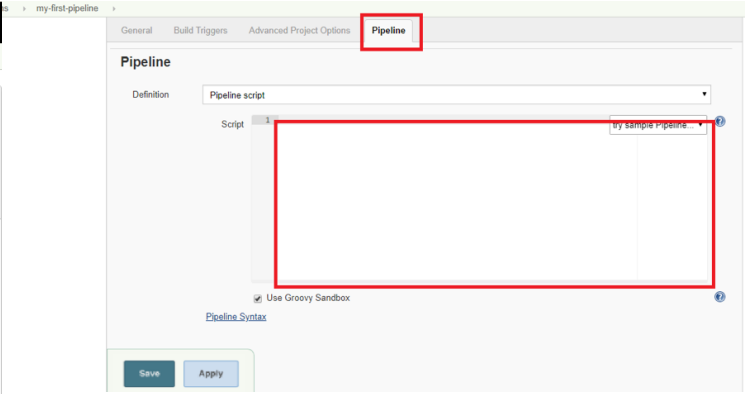
How to implement Jenkins Pipeline

Getting started

1. First, log on to your Jenkins server and select “New Item” from the left panel:
2. Next, enter a name for your pipeline and select “Pipeline” from the options. Click “Ok” to proceed to the next step
3. You can now start working your Pipeline script





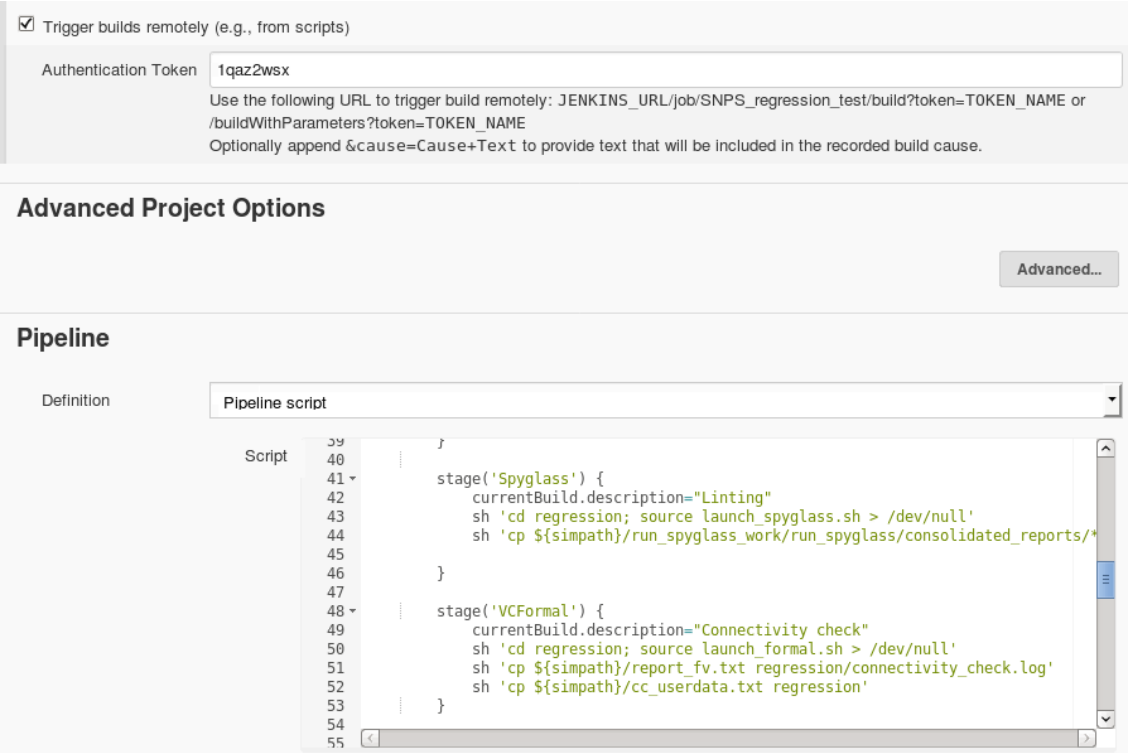


© 2019 Synopsys, Inc.

Run tools in pipeline

Configuration

- Pipeline is described in Jenkins using Groovy syntax
  - Different tools are being executed in different pipeline stages
  - Actual plugins are being called at the end of pipeline to collect results and draw graphs
- Select trigger according to you needs
  - In case of remote trigger with token additional configuration may be needed in version control (e.g. Git hooks) to get commits to launch builds



© 2019 Synopsys, Inc.

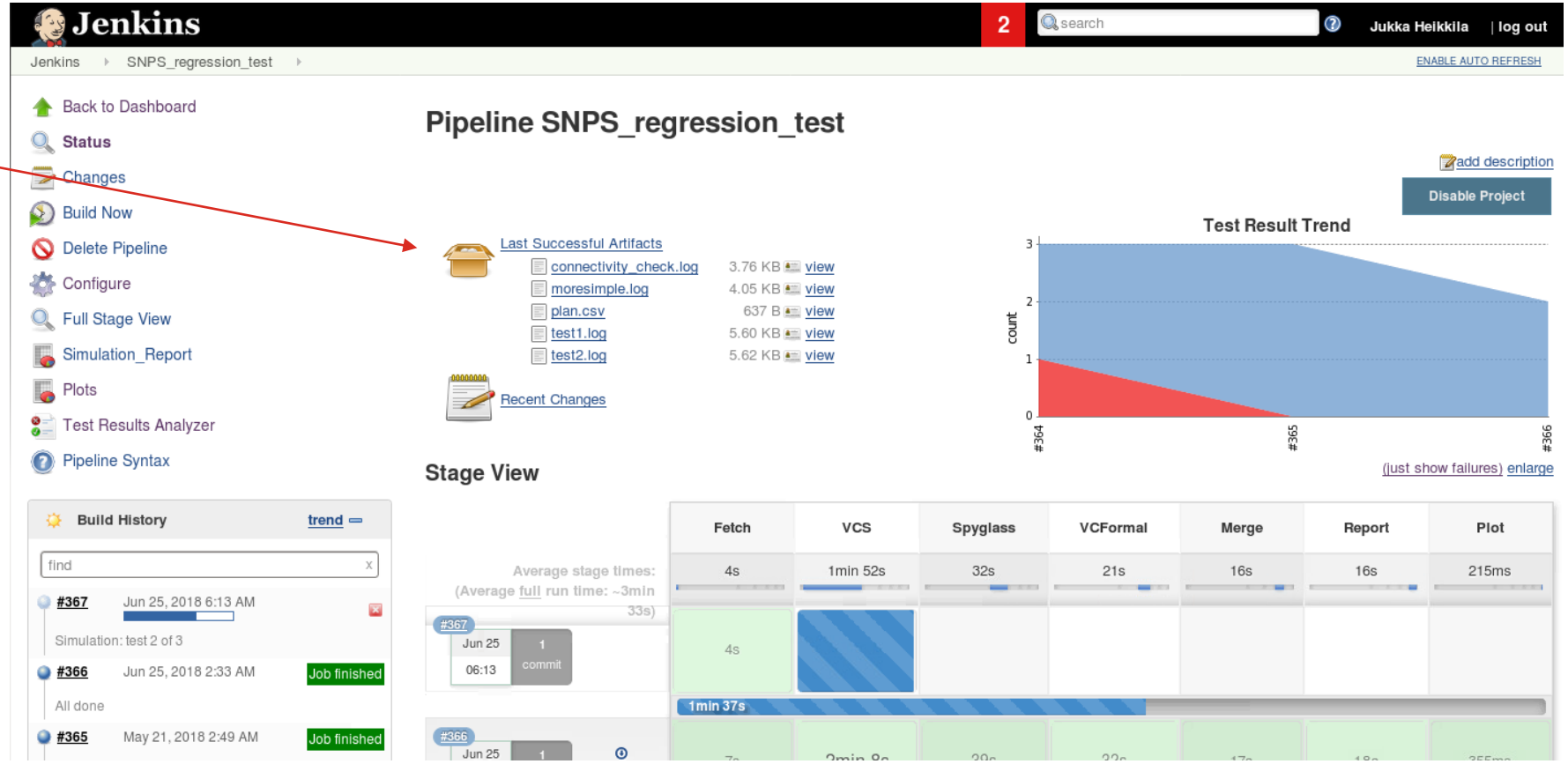
Visualizing the job results (Pipeline project)

Jenkins project page

Collected artifacts

More graphs from plugins

The results of last runs



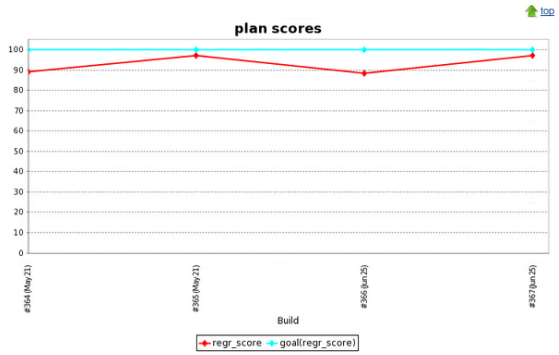
Provided by common plugins

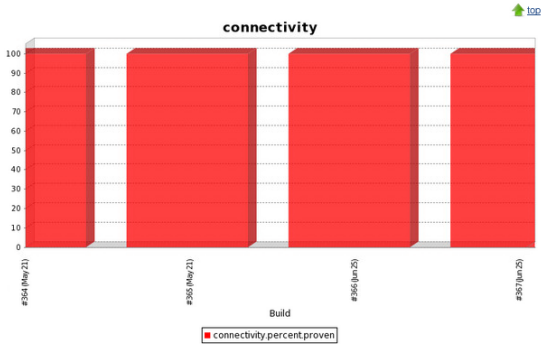
Pipeline stages

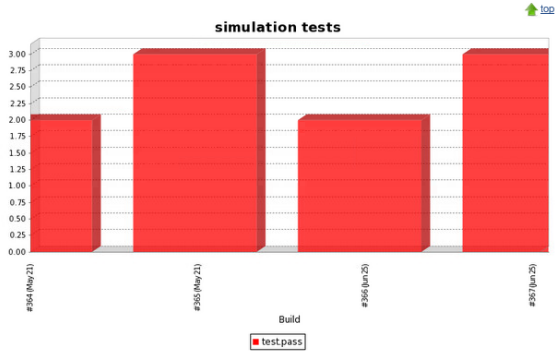
© 2019 Synopsys, Inc.

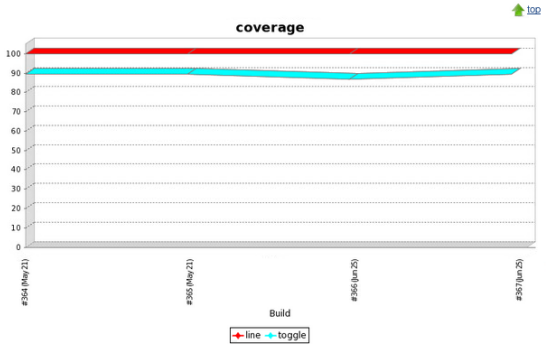
Visualizing the job results (Pipeline project)

Graphs from ”Plot” and ”Test result analyzer” plugins

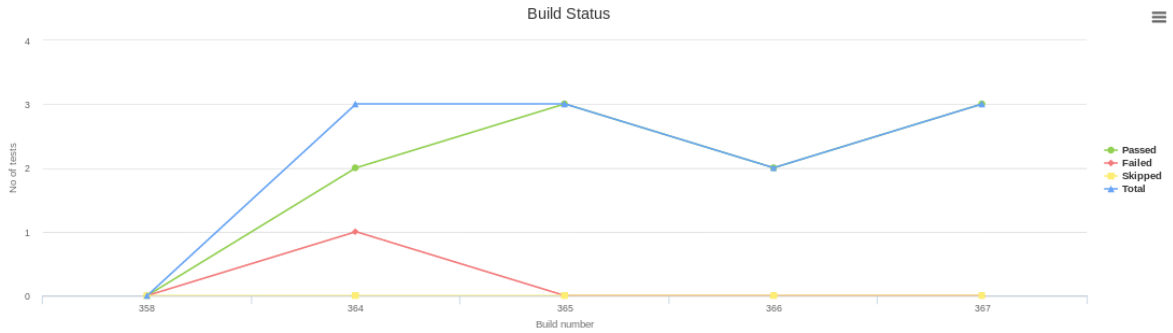








New Failures	Chart	See children	Build Number ⇒ Package-Class-Testmethod names ⇓	367	366	365	364	358
	☑	⊖	(root)	PASSED	PASSED	PASSED	FAILED	N/A
	☑	⊕	test1	PASSED	PASSED	PASSED	PASSED	N/A
	☑	⊕	test2	PASSED	PASSED	PASSED	PASSED	N/A
	☑	⊕	test3	PASSED	N/A	PASSED	FAILED	N/A

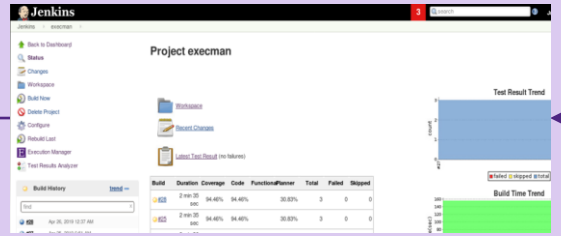


© 2019 Synopsys, Inc.

Jenkins Plugin Example

Synopsys Users Group

Jenkins web page



Execman: Fetch needed data

ExecMan: Unified compilation

Execman: Run simulation tests

Execman: Run Formal test

Execman: merge results to HVP plan

Jenkins Trigger

Execman Init

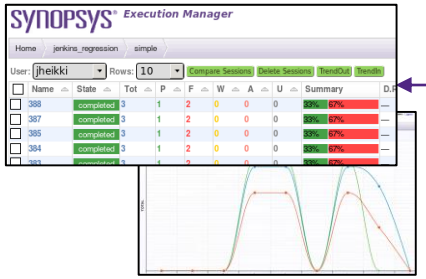
Execman build

Execman tests

Execman end

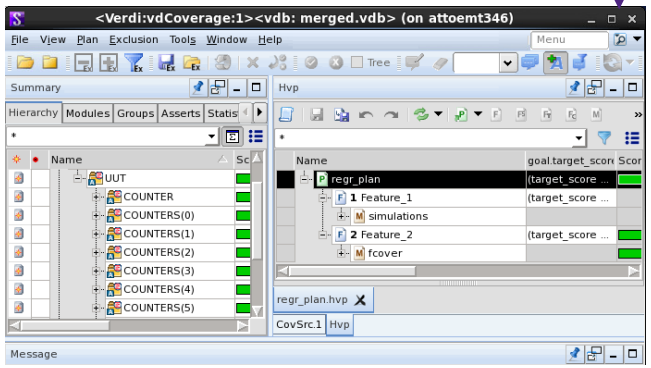
SCM

Execman server



Verdi Planner

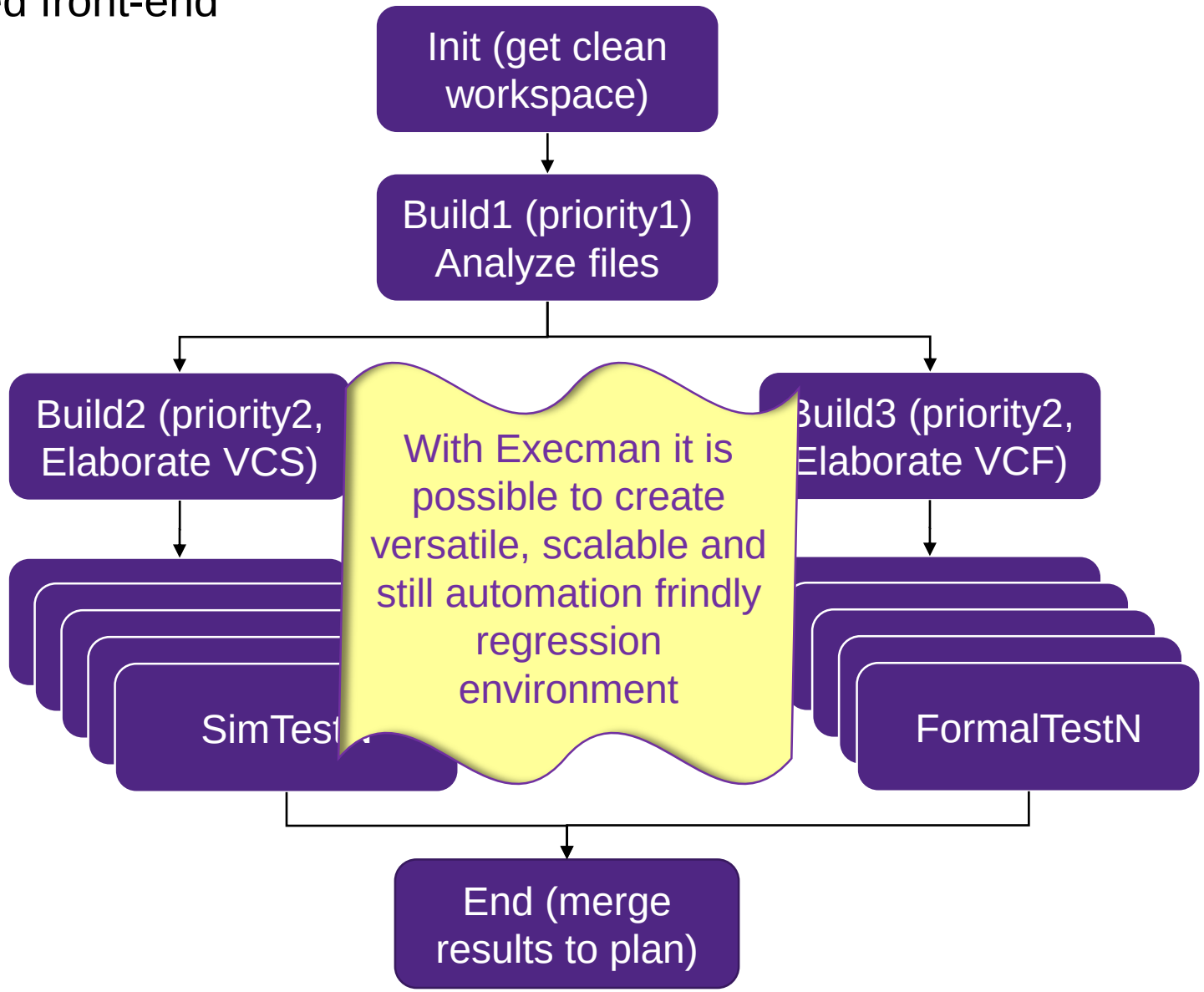
- Regression plan creation
- Debugging
- Exclusion management



© 2019 Synopsys, Inc.

Running multiple tools in parallel with Execman

With unified front-end

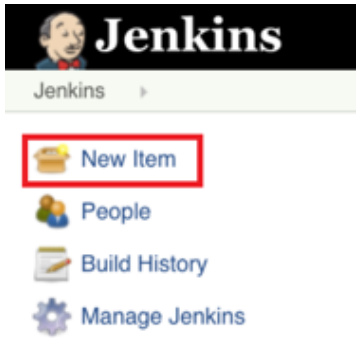


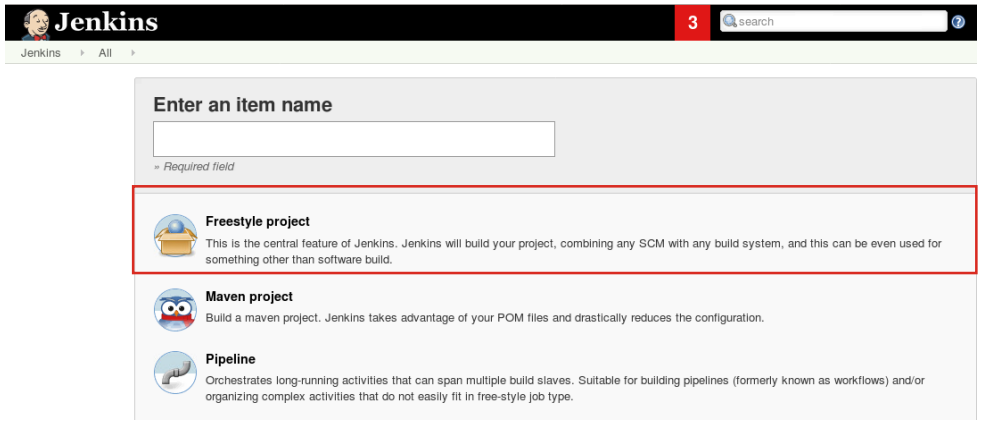

```
init:
  - run_cmd: cd $regr_dir; git clone …

builds:
  - name: Build1
    run_dir: $regr_dir/build_common
    run_cmd: make clean analysis
    priority: 1

  - name: Build2
    run_dir: $ regr_dir/build_vcs
    run_cmd: make elab_vcs
    priority: 2

  - name: Build3
    run_dir: $regr_dir/build_vcf
    run_cmd: make elab_vcf
    priority: 2

tests:
  - name: SimTest1
    run_cmd: make test_sim UVM_TEST= …
    build: Build2

  - name: FormalTest1
    run_cmd: make test_vcf MODE=CC
    build: Build3
…
end:
  - run_cmd: cd $regr_dir/merge; urg ….
```


© 2019 Synopsys, Inc.

How to use Jenkins Execman Plugin

1. First, log on to your Jenkins server and select “New Item” from the left panel:
2. Next, enter a name for your pipeline and select “Freestyle project” from the options. Click “Ok” to proceed to the next step
3. Install the Plugin like described earlier
4. You can now start working your build configuration

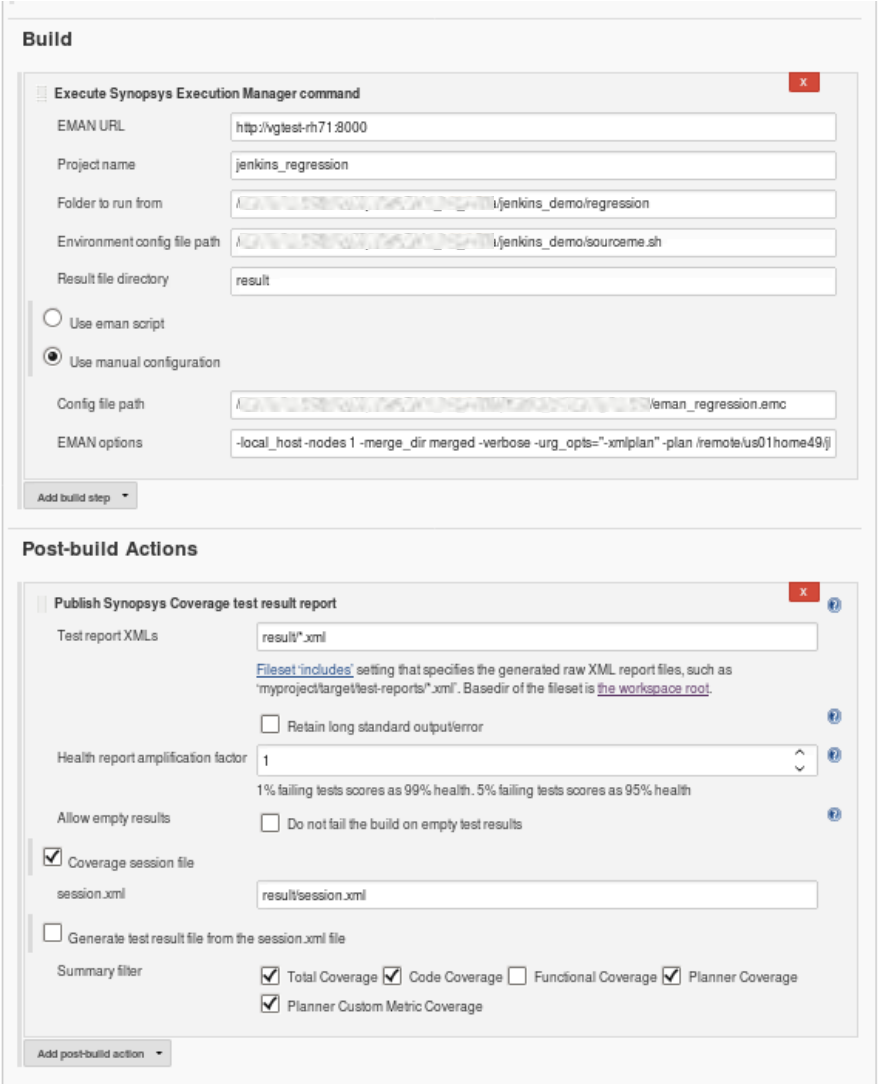




© 2019 Synopsys, Inc.

Configure the Plugin

- Add build step for Synopsys Execution Manager
  - Configure server and project details
- Add Post-build Action for publishing test results
  - Select metrics you want to see in Jenkins



© 2019 Synopsys, Inc.

Visualizing the job results (Execman plugin project)

Jenkins project page

More graphs from Execman and plugins

Previous Builds

Test results

Jenkins | 3 | search | Jukka Heikkila | log out

Jenkins › execman › ENABLE AUTO REFRESH

- Back to Dashboard
- Status
- Changes
- Workspace
- Build Now
- Delete Project
- Configure
- Rebuild Last
- Execution Manager
- Test Results Analyzer

Build History trend

find

- #28 Apr 26, 2019 12:37 AM
- #27 Apr 25, 2019 6:51 AM
- #25 Apr 25, 2019 3:28 AM
- #24 Apr 25, 2019 12:38 AM
- #23 Apr 25, 2019 12:09 AM

RSS for all | RSS for failures

Project execman

add description

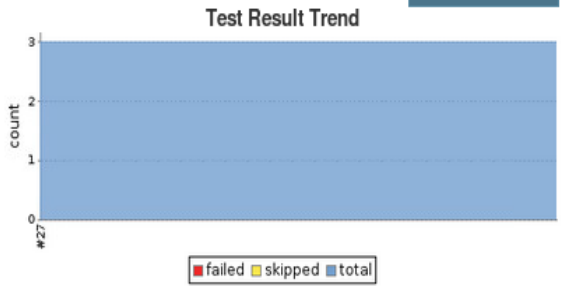
Disable Project

Workspace

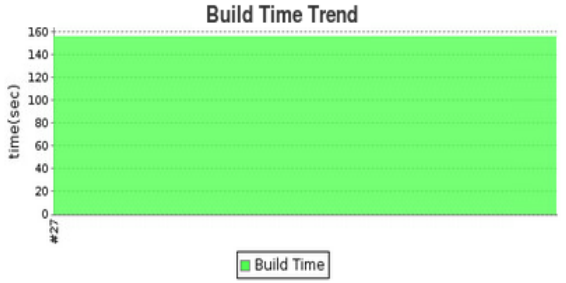
Recent Changes

Latest Test Result (no failures)

Build	Duration	Coverage	Code	Functiona	Planner	Total	Failed	Skipped
#28	2 min 35 sec	94.46%	94.46%		30.83%	3	0	0
#25	2 min 35 sec	94.46%	94.46%		30.83%	3	0	0
#27	2 min 36 sec	94.46%	94.46%		30.83%	3	0	0
#24	2 min 37 sec	94.46%	94.46%		30.83%	3	0	0
#23	2 min 4 sec	94.46%			90.51%	3	0	0



(just show failures) enlarge



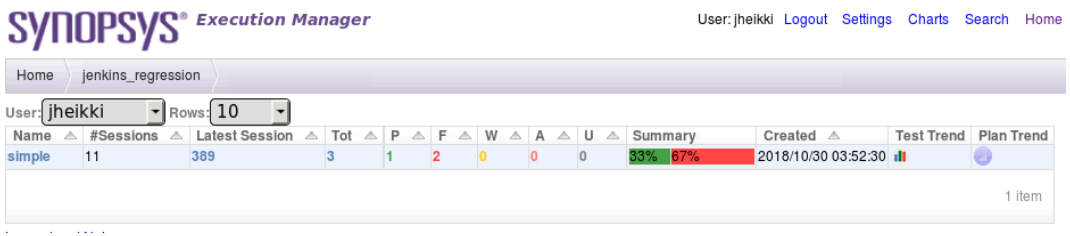
enlarge

© 2019 Synopsys, Inc.

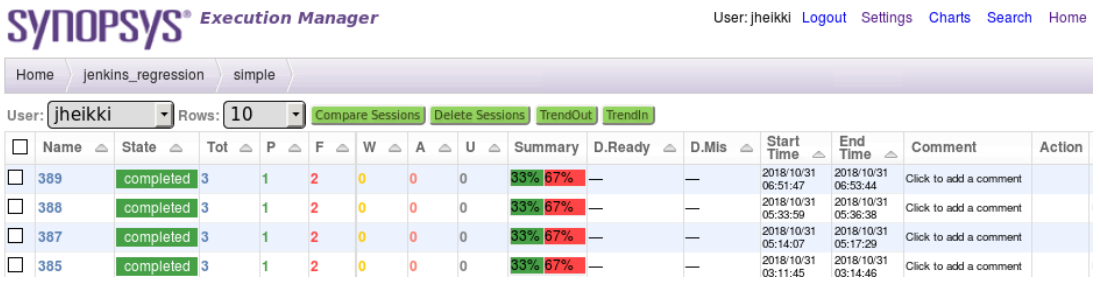
Visualizing the job results (both examples)

Execman web GUI

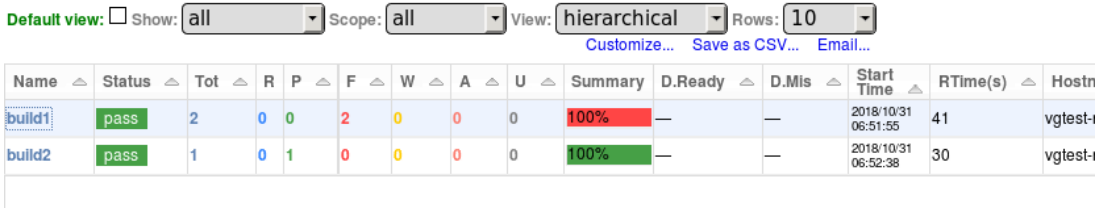
Project page



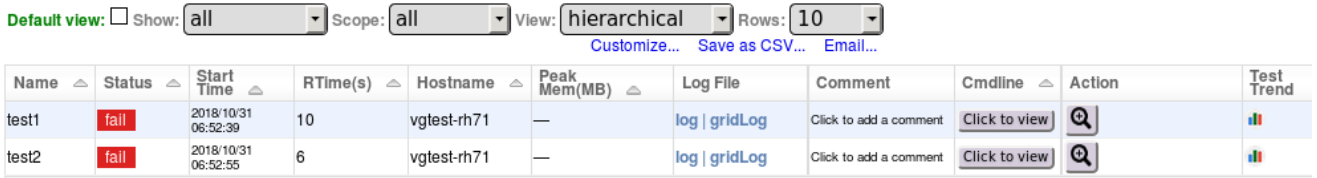
Config page



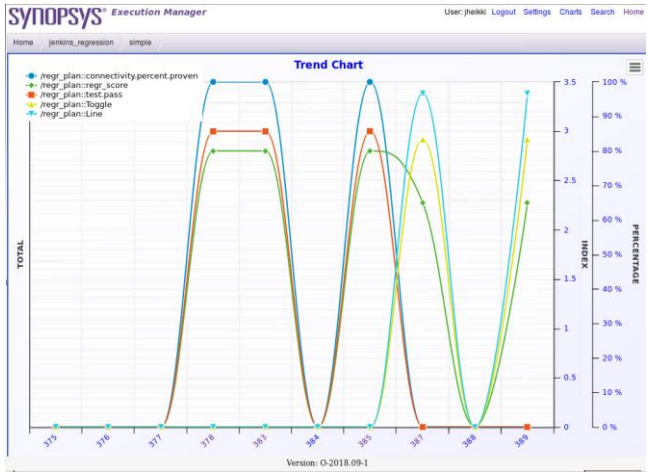
Build page



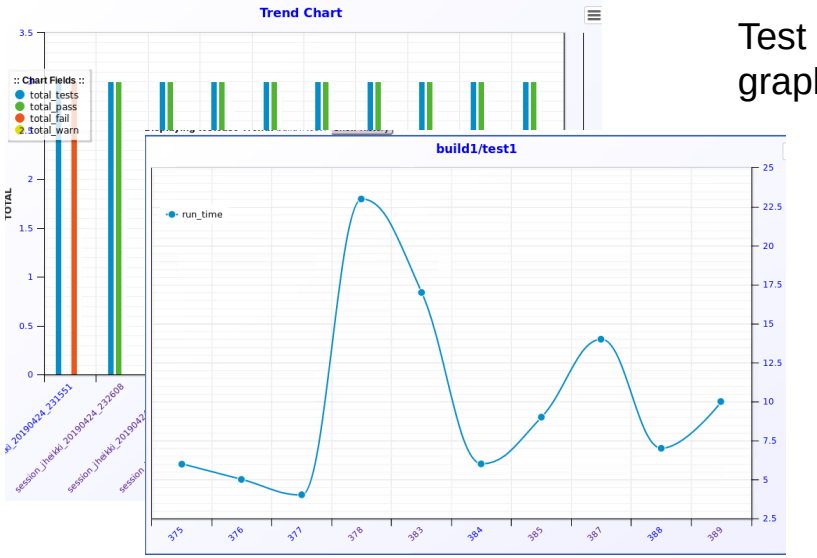
Test page



Plan graphs



Test graphs



© 2019 Synopsys, Inc.

Visualizing the job results (both examples)

urgReport HTML report

- Can be published in Jenkins as such using *HTML publisher* plugin

Back to SNPS_regression_test | dashboard

dashboard | **hierarchy** | **modlist** | groups | **tests** | **asserts** | **userdata** | **hvp** | **trend**

HVP Hierarchy regr_plan Feature_1

Feature : regr_plan.Feature_1

SCORE	LINE	TOGGLE	connectivity	proven	falsified	connectivity.percent.proven	regr_score	regr_score_icm2	regr_score_icm3	test	pass	fail	warn	assert	unknown	test.percent.pass
94.70	100.00	89.39	10	10	0	100.00	97.35	97.35		2	2	0	0	0	0	100.00

Measure Totals:

SCORE	LINE	TOGGLE	connectivity	proven	falsified	connectivity.percent.proven	regr_score	regr_score_icm2	regr_score_icm3	test	pass	fail	warn	assert	unknown	test.percent.pass
94.70	100.00	89.39	10	10	0	100.00	97.35	97.35		2	2	0	0	0	0	100.00

Measure simulations:
Metrics: test, test.percent.pass, regr_score, regr_score_icm2
Goal: (regr_score >= 100) (regr_score_icm2 == 90)

SCORE	LINE	TOGGLE	connectivity	proven	falsified	connectivity.percent.proven	regr_score	regr_score_icm2	regr_score_icm3	test	pass	fail	warn	assert	unknown	test.percent.pass
							100.00	100.00		2	2	0	0	0	0	100.00

Sources:
test2

NAME	SCORE	LINE	TOGGLE	connectivity	proven	falsified	connectivity.percent.proven	regr_score	regr_score_icm2	regr_score_icm3	test	pass	fail	warn	assert	unknown	test.percent.pass
test2								100.00	100.00		1	1	0	0	0	0	100.00

test3

NAME	SCORE	LINE	TOGGLE	connectivity	proven	falsified	connectivity.percent.proven	regr_score	regr_score_icm2	regr_score_icm3	test	pass	fail	warn	assert	unknown	test.percent.pass
test3								100.00	100.00		1	1	0	0	0	0	100.00

Measure connectivity:
Metrics: Line, Toggle, Assert, connectivity, connectivity.percent.proven, regr_score, regr_score_icm2
Goal: (connectivity.proven == 1) (regr_score >= 100) (regr_score_icm2 == 90)

SCORE	LINE	TOGGLE	connectivity	proven	falsified	connectivity.percent.proven	regr_score	regr_score_icm2	regr_score_icm3	test	pass	fail	warn	assert	unknown	test.percent.pass
			10	10	0	100.00	100.00	100.00								

Sources:
CON1

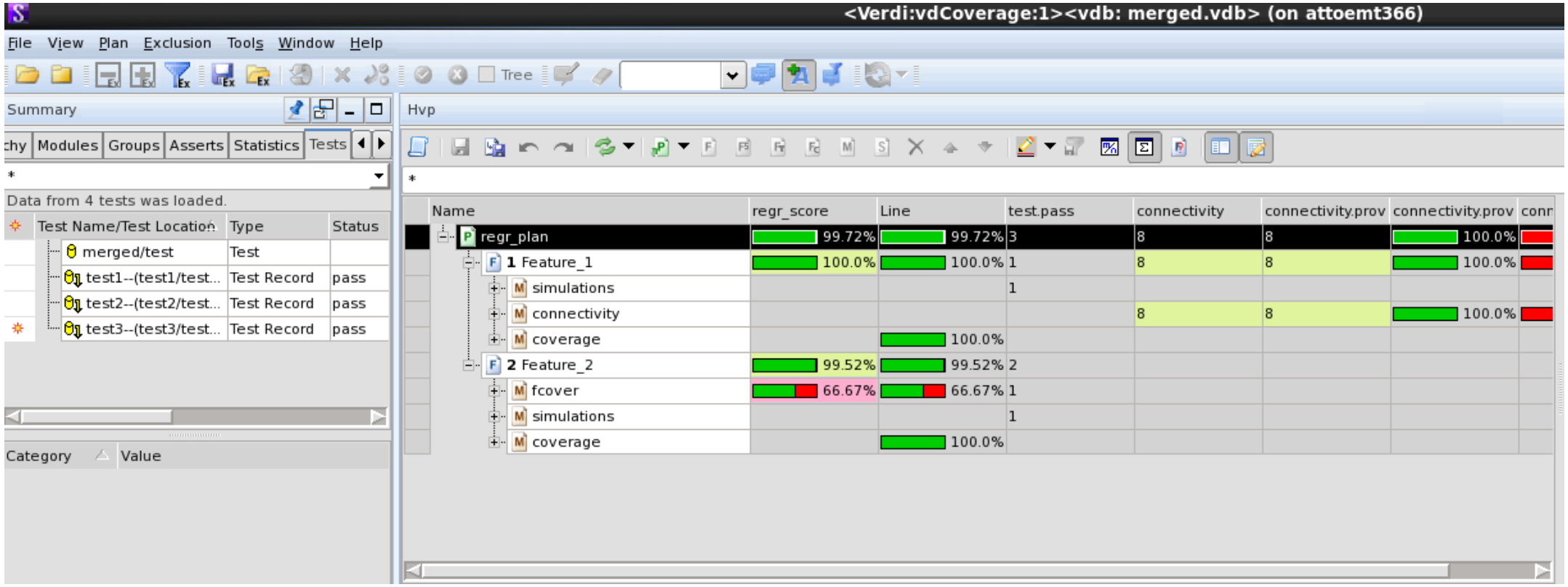
NAME	SCORE	LINE	TOGGLE	connectivity	proven	falsified	connectivity.percent.proven	regr_score	regr_score_icm2	regr_score_icm3	test	pass	fail	warn	assert	unknown	test.percent.pass
CON1				1	1	0	100.00	100.00	100.00								

© 2019 Synopsys, Inc.

Visualizing the job results (both examples)

Verdi Planner

- Detailed results can be viewed in Verdi Planner



© 2019 Synopsys, Inc.

Agenda

- Introduction
- SW development practices
- Building continuous regression flow
- Example setups
- **Summary**

© 2019 Synopsys, Inc.

Facing the challenges

Synopsys Users Group Europe

How to plan the verification efficiently?

Feature driven verification with Planner enables project to reach the goal step by step

What is the overall verification status? Information can come from multiple sources

Using Planner/Execman flow enables progress tracking with data from several sources

How can this information be made visible to users/management?

Execman SQL server with web interface provides scalable and flexible way of distributing information

Verification time gets long as designs grow

Distributed parallel simulations with multicore option is the best solution for this

Utilizing R&D infra efficiently for regression simulations (grid, multicore, configs)

Execman is the tool for this. Excluding this part of the flow to Execman makes the regression flow simpler to maintain

Launching functional and static checks in FE development takes lots of time for small changes

Some CI platform, like Jenkins, can be harnessed here to bring automation

Building and testing regression solutions for Jenkins is difficult

You can build and test environment with Execman and when ready, just add ”eman” command in Jenkins

How to keep SW and HW in sync?

Using Synopsys Virtual prototyping tools it is possible to run co-simulations with real SW

© 2019 Synopsys, Inc.

Summary

- SoC verication is complex subject and automation is needed
- Jenkins is a widely used general purpose automation platform, but it is not capable of running large scale SoC verification flow as such
- It is possible to implement parts of CI flow in SoC development, or start using existing CI infrastructure
- Verdi Planner enables smart verification planning with custom metrics for comprehensive progress monitoring
- VCS/VCFormal/Spyglass provides the leading engines for workstation verification
- ExecMan is a final piece of a puzzle providing computing farm control, efficient SQL based database for data storage and real-time progress monitoring

© 2019 Synopsys, Inc.

Thank you!