Methods to improve verification quality on module level

Markus Groß
Siemens AG Automation and Drives
Erlangen

SIEMENS

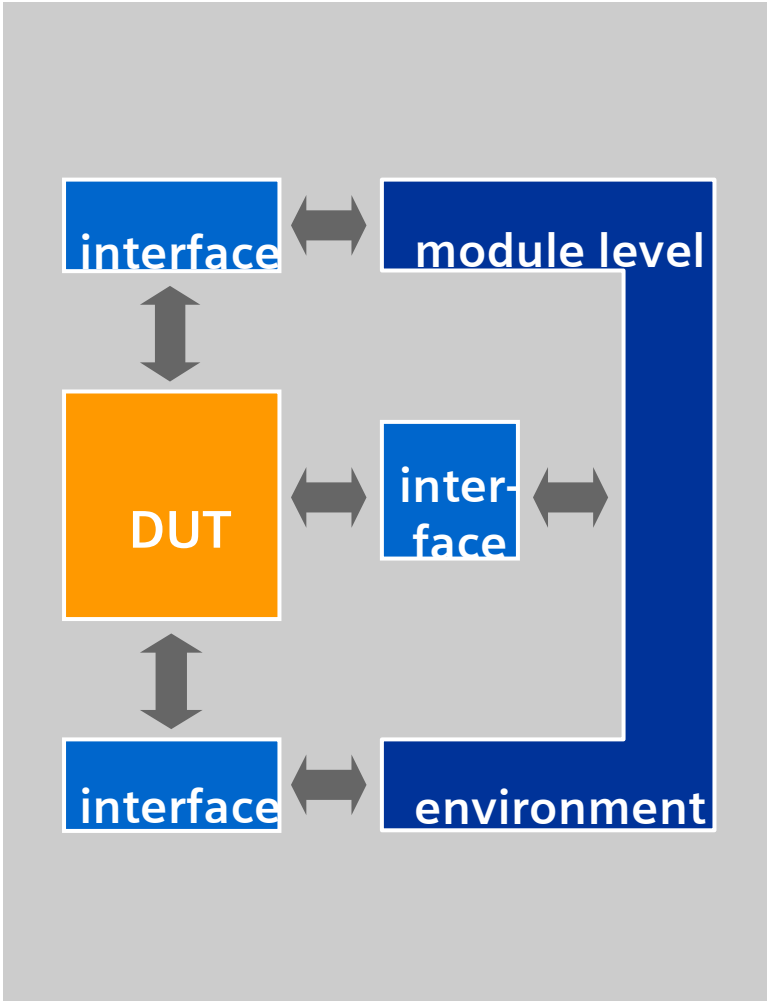
ASIC Design Center Erlangen

ASIC Design Center Erlangen
- 14 team members
- Designing SoC ASICs for tough real-time systems
- application: industrial automation

Why module level?
- faster runs
- easier to stimulate
- easier to reuse

Module level as quality key
- Bug free systems always consist of bug free modules
- Quality is a bottom-up task

Our Challenge
- Up to 50 modules per chip
- No planned redesign



Methods to improve verification quality on module level

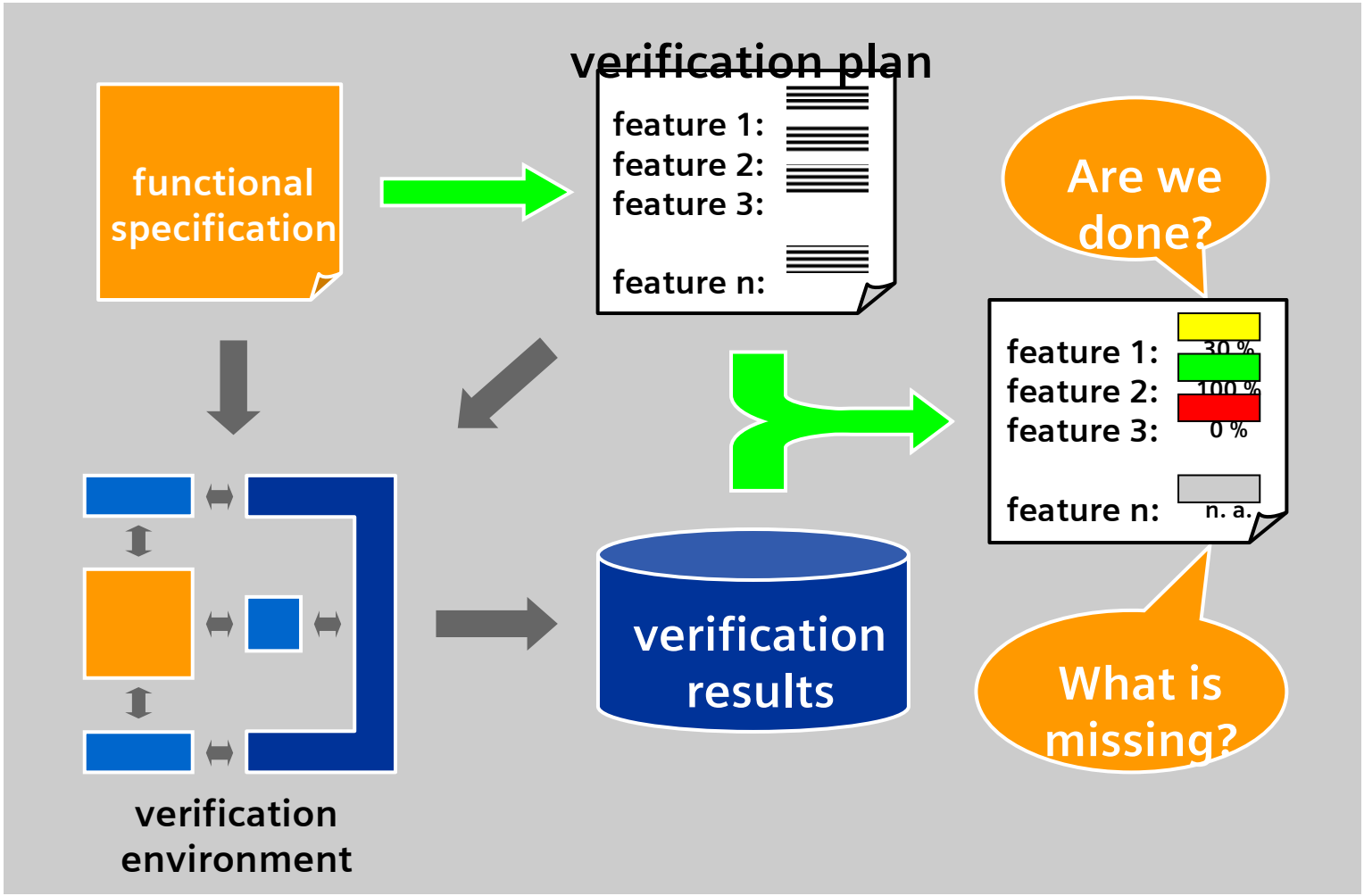
Verification Planning

Make Closure Measurable

Methods to improve verification quality on module level

SIEMENS

Closure Measurement



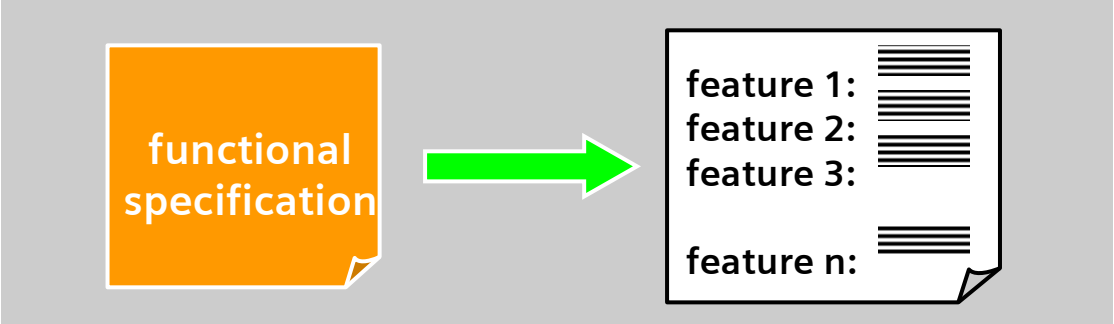
Verification Planning Session: Goal

Goal

- Create verification plan from functional specification
- Transfer knowledge about DUT to verification team
- Have a very effective specification review

Participants

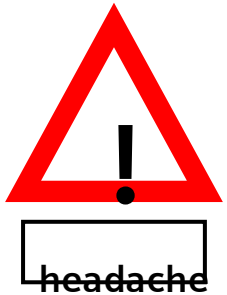
- Moderator
- Author of specification
- Designer of DUT
- Verification engineer
- Other experts for functionality of DUT, e. g. software developers



Verification Planning Session: Schedule

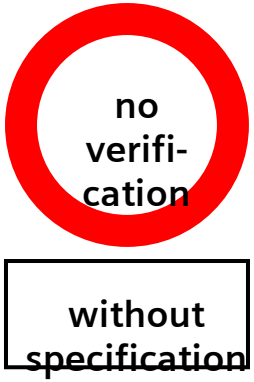
Take your time

- At least 1 day
- Up to 5 days on complex modules
- Very exhausting



Schedule

- After functional specification is stable
- Before start of verification
- Rather independent of state of design



Never start verification without stable specification

- Great impact on quality (many dangerous late changes)
- Great impact on schedule (up to 3 times of planned effort)

SIEMENS

Methodology Mix

Best of all Worlds

Methods to improve verification quality on module level

SIEMENS

3 Principles of Successful Verification

Must-Run-In-Batch Principle

- Every test or run must result in a simple pass or fail
  - self checking mechanism required
  - transcripts and wave only for debugging
- Every test must ensure that functionality is really covered
  - By design of the test itself (test based verification)
  - By method itself (formal property checking)
  - By filled coverage (coverage driven verification)

Two-Head Principle: Designers never verify their own modules

- Double approach: Two engineers make different mistakes
- No double knowledge: Design and verification need different competences

Use-Best-Methodology Principle

- Every single feature is verified with the methodology suited best to this feature

Methodology Overview

Test case based verification

+ Easy to learn, fast results
+ Easy acceleration, can run on prototype
– Very poor corner case coverage (even on very long runs such as on a prototype)
■ Application: simple modules, fixed scenarios (e. g. booting), very long runs (on prototype)

Formal Property Checking

+ Exhaustive though running fast
– Limited complexity of design
– Limited abstraction level
■ Application: interfaces, regular structures

Coverage Driven Verification

+ Good corner case coverage
+ high abstraction level
– Time consuming, needs expert knowledge
■ main workhorse, especially complex modules

Methods to improve verification quality on module level

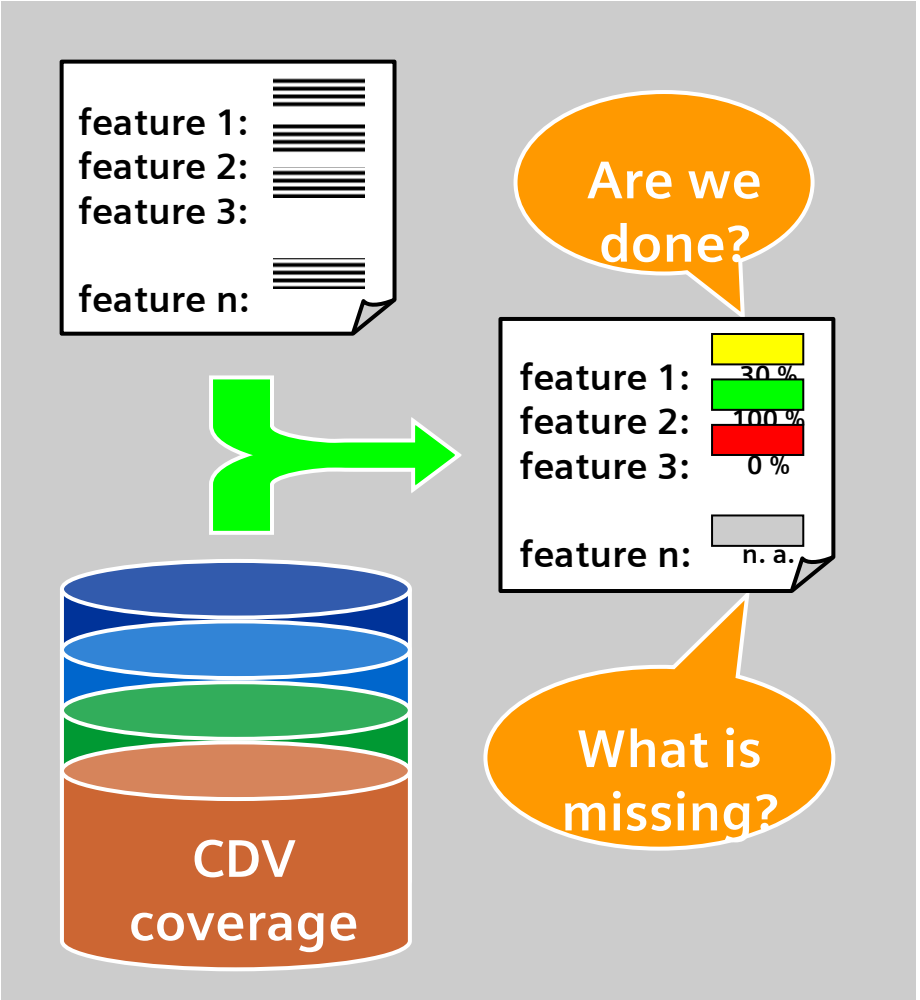
Total Coverage

Every verification item (test, property, coverage) mapped to features of the verification plan

All verification results brought together in one chart

Free mixture of verification methodologies

Control over results of a whole team



Measureable Coverage Driven Verification

Successful regressions need reliable checks

Methods to improve verification quality on module level

SIEMENS

Reliable Checker Requirements

Reliable Checkers must fire on DUT errors

- ... or you might miss them
- Check your checkers

Reliable Checkers must not fire when there is no DUT error

- ... or you will spend most of your time analysing failed runs
- ... or you might miss DUT errors because you cannot analyse every failed run

Reliable Checkers must be linked with coverage

- A check without coverage might never check at all
- A coverage without check might cover an unchecked feature

Reliable Checkers allow massive parallel regression

- ... because you can trust the results

First Collect Then Check

Expected behaviour approach: Verification Environment ...

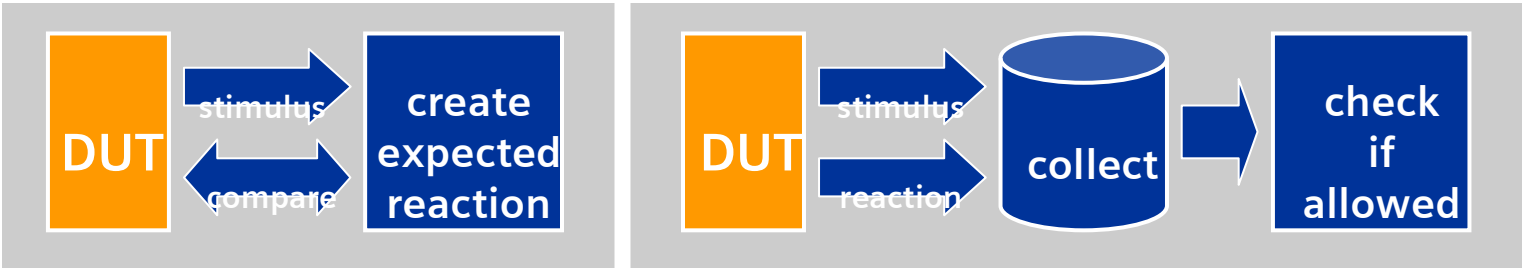
- ... observes (or creates) stimulus
- ... calculates expected reaction of DUT
- ... compares expected reaction with actual reaction of DUT

Permitted behaviour approach: Verification Environment ...

- ... observes (or creates) and collects stimulus
- ... observes and collects reaction of DUT
- ... checks whether collected dataset is within specification

Advantages of permitted behaviour approach

- Can cope with multiple allowed reaction
- Can easier be adapted to unexpected corner cases



Weak Checks

Functional Specification allows many different implementations

- Current DUT is only one of them
- Implementation of DUT may change
- Reliable checkers should cover all possible implementations

Typical uncertainties

- When does a register write take effect?
- How fast should the DUT react on external signals?
- In which sequence does the DUT react on concurrent events?

Weak checks

- ... only check what the functional specification requires
- ... allow as many different behaviours as possible
- ... consider target application needs

Resynchronisation to DUT

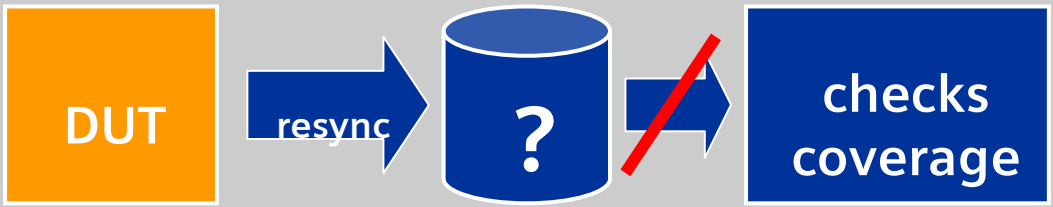
Problem: DUT behaviour becomes unpredictable

- Concurrent events or unexpected corner case
- Multiple different reactions possible
- Multiple possible states of DUT afterwards

- No need to check actual reaction of DUT
- But need to know state of DUT for further checking

Solution: Resynchronisation to DUT

- Switch off checks and coverage
- Wait until DUT is stable again
- Copy state of DUT into environment
- Resume checks and coverage



Summary

Verification Planning Session

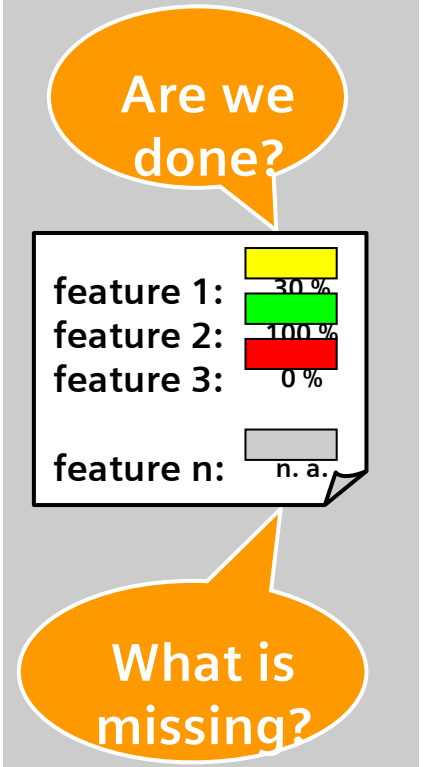
- Knowing when you are done needs a definition what to do
- Effective process to create verification plan
- Total Coverage: Map all verification results to verification plan

Best of all worlds

- Several verification methodologies available
- Use them all!
- Create best mix for your module

Reliable Checkers

- Checkers must be able to cope with random tests
- First collect, then check; create “weak checks”
- Reliable checkers need long experience
- Reliable checkers find more bugs



Motion Control Systems

Thank you very much

SIEMENS