

COMSM0115 Design Verification:

Are we there yet?

(The back-end of the verification cycle)

Kerstin Eder

(Acknowledgement: Avi Ziv from the IBM Research Labs in Haifa has kindly permitted the re-use of some of his slides.)



Department of
COMPUTER SCIENCE

Last Time

- Assertion-based Verification

Kerstin Eder

COMSM0115 – Design Verification

2

Outline

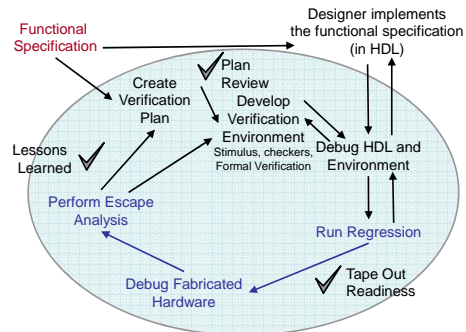
- The verification cycle - revision
- Analysis and adaptation
 - Coverage analysis
 - Failure analysis
- Regression
- Tape-out readiness
- Escape analysis

Kerstin Eder

COMSM0115 – Design Verification

3

The Verification Cycle



Kerstin Eder

COMSM0115 – Design Verification

4

My Environment Is Ready. Now What?

- More functionality is added to the design
 - And therefore, to the verification environment
- Mature enough design is progressed to the next level in the design hierarchy
 - Unit to core to chip to system
- Bugs are being discovered and fixed
 - And bug fixes need to be verified
- The implementation of the verification plan continues
 - Closing holes in coverage
 - Updating the verification plan itself as needed
- Regression is being executed to ensure everything still works

Kerstin Eder

COMSM0115 – Design Verification

5

Analysis and Adaptation

- Building a good verification plan is the first step for successful verification
 - But, it is not enough!
- Need to constantly:
 - Monitor the verification process
 - Analyze the observations
 - Adapt to address issues identified by the analysis
- Three basic levels of adaptation
 - Change the way the verification environment is activated
 - Change the verification environment
 - Change the verification plan

Kerstin Eder

COMSM0115 – Design Verification

6

Two Types of Analysis

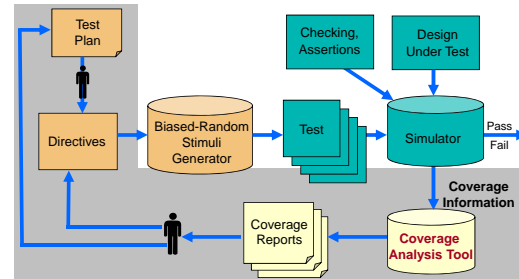
1. Coverage analysis
2. Failure analysis

Kerstin Eder

COMSM0115 – Design Verification

7

Coverage Analysis



Kerstin Eder

COMSM0115 – Design Verification

8

Why Coverage Analysis

- The main goals of the coverage process are
 - Monitor the quality of the verification process
 - Identify unverified and lightly verified areas
 - Help understanding of the verification process
- **Coverage analysis** helps closing the loop from coverage measurement to the verification plan and test generation

Kerstin Eder

COMSM0115 – Design Verification

9

Coverage Analysis Goals

- **Conflicting goals for coverage analysis:**
 - Want to collect as much data as possible
 - Not to miss important events
 - User needs concise and informative reports
 - Not to get drawn into too much detail
- Different types of users require different type of information
- **Goal:** provide concise and informative reports that address the specific needs of the report user

Kerstin Eder

COMSM0115 – Design Verification

10

Types of Coverage Reports

- **Status reports**
 - Coverage status summary
 - Detailed status reports of covered and uncovered tasks
 - Reports can be adapted to specific user needs
 - Allow interactive navigation between reports to explore coverage state
- **Progress reports**
 - Progress of coverage over time

Kerstin Eder

COMSM0115 – Design Verification

11

Coverage Status Summary

- Provides a short summary of the coverage state
- Provides the overall state of the coverage model (or models)
- Useful for
 - Status meetings and status reports
 - A quick glance at the coverage state

Size of coverage space:	1539648
Number of tasks:	4200
Number of tasks covered:	1273
Percent tasks covered:	30.39524
Number of holes:	2927
Number of illegal tasks:	9
Number of traces measured:	16254
Number of cycles measured:	94231273

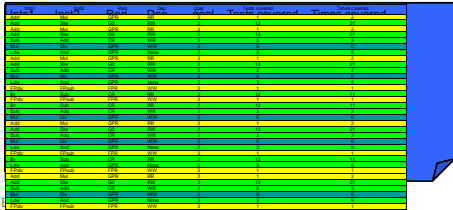
Kerstin Eder

COMSM0115 – Design Verification

12

Detailed Status Report

- Provides details on each task in the coverage model
 - Covered or not
 - How many times covered
 - In how many tests covered
 - First and last time covered
 - Coverage goals
 - ...



Kerstin Eder

13

Detailed Status Reports

- Detailed status reports can provide too much detail even for a moderate coverage model
 - Hard to focus on the areas in the coverage model we are currently interested in
 - Hard to understand the meaning of the coverage information
 - Are we missing something important?
- Solution: **Views into the coverage data**
 - Allow the user to focus on the current area of interest and look at the coverage data with the appropriate level of detail
 - Dynamically define the coverage model

Kerstin Eder

COMSM0115 – Design Verification

14

Types of Coverage Views

- Views based on **coverage data**
 - Counts
 - Dates
- Views based on **coverage definition**
 - Projection
 - Selection
 - Partitioning

- Other filtering mechanisms

All the above options can be combined

Kerstin Eder

COMSM0115 – Design Verification

15

Projection

- Project the n dimensional coverage space onto an m ($< n$) subspace
- Allow users to concentrate on a specific set of attributes
- Help in understanding some of things leading up to the big picture

Instruction	Count	Density
fadd	12321	127/136
fsub	10923	122/136
fmul	4232	94/136
fsqrt	13288	40/56
fabs	9835	38/40

Kerstin Eder

COMSM0115 – Design Verification

16

Selection

- Selects a subset of the values of an attribute
- Allows the report to concentrate on a specific area in the coverage model
- Clears the report from data that is not of interest at the time

Instruction	Count	Density
fmadd	9725	107/136
fmsub	9328	111/136
frsqte	9792	23/36
fsqrt	13288	40/56

Kerstin Eder

COMSM0115 – Design Verification

17

Partitioning

- Provides a more coarse-grained view of the coverage data
- Partitions values of given attributes into non-overlapping sets
 - Example: Operand \rightarrow Pos, Neg, NaNs

4/12	9/12	9/12
5/12	10/12	8/12
7/12	3/12	9/12
8/12	7/12	10/12

Kerstin Eder

COMSM0115 – Design Verification

18

Automatic Coverage Analysis

- Detailed status reports do not always reveal interesting information hidden in the coverage data
 - You need to know where to look
 - You need to know which questions to ask the coverage tool
- Specifically, it is hard to find large areas of uncovered tasks in the coverage model

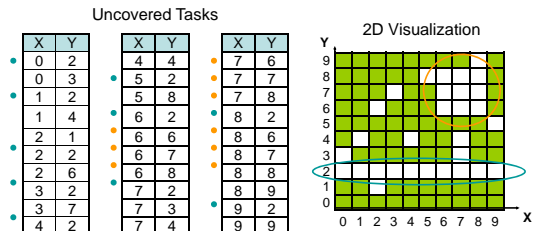
Kerstin Eder

COMSM0115 – Design Verification

19

Large Holes Example

- All combinations of two attributes, X and Y
 - Possible values 0 – 9 for both (100 coverage tasks)
- After a period of testing, 70% coverage is achieved



Kerstin Eder

COMSM0115 – Design Verification

20

Hole Analysis Algorithms

- Try to find large areas in the coverage space that are not covered
- Use basic techniques to combine sets of uncovered events into large meaningful holes
- Two basic algorithms
 - Aggregation
 - Projected holes

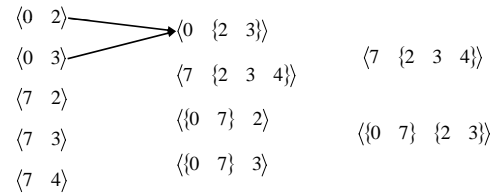
Kerstin Eder

COMSM0115 – Design Verification

21

Aggregated Holes

- Combine uncovered tasks with common values in some attributes
- Similar to Karnaugh maps



Kerstin Eder

COMSM0115 – Design Verification

22

Projected Holes

- Find holes that are **complete subspaces** of the coverage space
- Holes are in the form $\langle q_1, q_2, \dots, q_n \rangle$
 - q_i is either a single value or a wildcard (*)
 - Hole dimension is the number of wildcards
 - Example: $\langle \text{fadd}, \text{add}, *, \text{WW} \rangle$ has dimension 1
- Hole p is an ancestor of q if all the tasks in q are in p
 - $\langle \text{fadd}, *, *, \text{WW} \rangle$ is ancestor of $\langle \text{fadd}, \text{add}, *, \text{WW} \rangle$
- Holes with higher dimensions usually represent larger subspaces and are more important

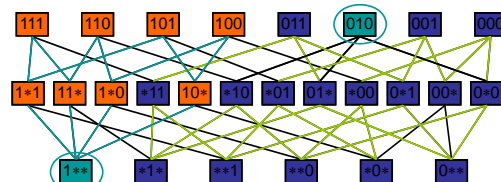
Kerstin Eder

COMSM0115 – Design Verification

23

Projected Holes Algorithm

- Build layered network of all subspaces
- Recursively mark ancestors of covered tasks
- Loop from the bottom
 - Report unmarked nodes as holes
 - Recursively mark descendants



Kerstin Eder

COMSM0115 – Design Verification

24

Coverage Progress

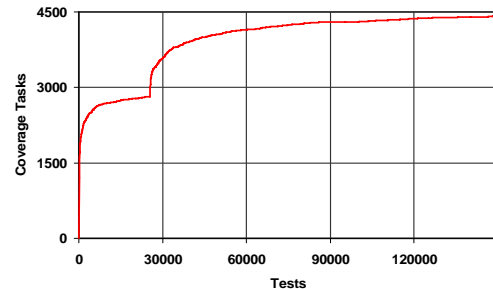
- Shows the progress of coverage over time
- Time can be measured by
 - Wall clock (or calendar) time
 - Number of tests
 - Number of simulation cycles
- Can be used on the entire coverage model or specific views of it

Kerstin Eder

COMSM0115 – Design Verification

25

Coverage Progress Example



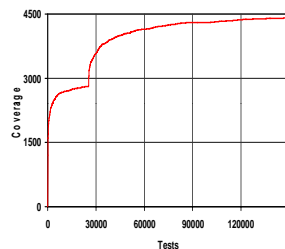
Kerstin Eder

COMSM0115 – Design Verification

26

Progress Report Usage

- Progress report can provide a lot of information
 - How well we are progressing overall
 - What is the current progress rate
 - Are we experiencing changes in the progress rate
 - What is the expected maximal coverage
 - When it would be reached



Kerstin Eder

COMSM0115 – Design Verification

27

Failure Analysis

Failure Analysis

- During execution of the verification plan (many) failures are observed
- This is not a bad phenomena
 - Remember that the goal of the verification process is to identify faults in the design under verification
- The goal of failure analysis is to understand failures, their causes, their relation to one another, and their relation to the verification process

Kerstin Eder

COMSM0115 – Design Verification

29

Failures and Faults

- **Failure** – an observed DUV behavior that violates the specified behavior
- **Fault** – the root cause of a failure
- There can be a **many-to-many relationship** between faults and failures
 - Mishandling of overflow in the input FIFO can cause:
 - Lost commands in the output port
 - Bad data in the output port
 - Bad data in the output port can be caused by:
 - Mishandling of overflow in the input FIFO
 - Bad selection in the output selector

Kerstin Eder

COMSM0115 – Design Verification

30

How Failures Are Detected

- Inspection and code review
- Output of formal verification tools or other static analysis tools, such as lint
- Activation of response checkers during simulation
- Analysis of coverage data
- Visual observation of application misbehavior

Kerstin Eder

COMSM0115 – Design Verification

31

Types of Failure Analysis

- Detailed failure analysis
 - Understand the cause and effects of failures and faults on the design, environment, verification process and more
- Statistical failure analysis
 - Identify trends, provide prediction

Kerstin Eder

COMSM0115 – Design Verification

32

Detailed Failure Analysis

- The outcome of the analysis
 - The failure is understood and recorded
 - The failure is resolved
 - The verification plan and process are adapted
 - Lessons learned for the future
- Note: In most cases failure analysis—and especially the last two items—are simple and the outcome of the analysis is that we found a failure and a fault when and where expected and because we are doing our job the right way.

Kerstin Eder

COMSM0115 – Design Verification

33

Understanding the Failure

- The goal is to **understand the scope and severity** of the failure and how the failure can be recreated
- Provides useful information for debugging and other parts of the failure analysis
 - Simplify and generalize the failure conditions
 - Find simpler settings / stimuli that recreates the failure
 - Find necessary and sufficient conditions for the failure
 - Localize the fault in terms of place and time
 - **Research:** Generate easy-to-debug tests

Kerstin Eder

COMSM0115 – Design Verification

34

What to Look For

- **In simulation**
 - Determinism
 - Does the failure always occur in the same settings?
 - With the same seed?
 - With different seeds (or random seed)?
 - Parameters that are correlated with the failure
 - Parameters that cause the failure to disappear
 - Parameters that cause the failure to change
 - Specific parts in the stimuli that are correlated to the failure
- **In formal verification**
 - Constraints that affect the failure
 - Time bounds that affect the failure

Kerstin Eder

COMSM0115 – Design Verification

35

Resolving the Failure

- **This does not always mean fix the fault**
 - Defer to future tape outs / releases
 - Bypass by software or surrounding modules
 - Record in errata sheets
- **Need to ensure that the resolution is complete**
 - The fix / bypass is correct
 - All cases are covered
 - No new faults introduced in the process
 - (Similar cases are also handled)
- **Mini-verification plan is needed**
 - Coverage models
 - Stimuli generation strategy
 - New result checkers

Kerstin Eder

COMSM0115 – Design Verification

36

Adapting the Verification Plan and Process

- Need to minimize faults found by chance or found too late
 - These faults can easily be missed if we are less lucky
- Indicators that faults are found by chance
 - Faults are not found at the right time
 - Fault is found at the wrong level of the hierarchy
 - Faults are found not at the area we concentrate on
 - Need to understand why faults are not found at the right time
 - And, change the plan and process accordingly
 - Faults are not found by the right checker
 - Only a side effect of the fault is detected
 - May indicate missing checker or problems in existing checker
 - Simulation with failure is not flagged by coverage
 - Does not activate uncovered or rarely covered coverage point
 - Indicates missing coverage models

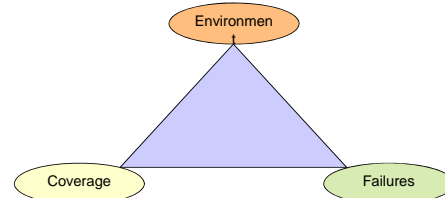
Kerstin Eder

COMSM0115 – Design Verification

37

Correlating Coverage and Failures

- There is a direct correlation between
 - Changes in the verification environment and the DUV
 - Progress in coverage
 - Detection of new failures

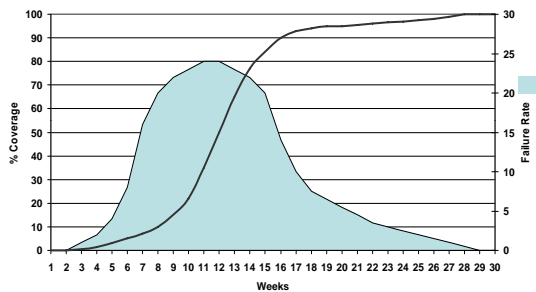


Kerstin Eder

COMSM0115 – Design Verification

38

Correlating Failure Rate and Coverage Progress



Kerstin Eder

COMSM0115 – Design Verification

39

Individual Coverage and Failure Correlation

- Correlating a failure to specific coverage can be helpful in the failure analysis and debugging processes
- Rare coverage points exercised by a simulation that fails can hint at the location of the fault that caused the failure
 - Rare coverage points are coverage points rarely, if ever, exercised by passing simulations
 - These coverage points record what happened in the DUV prior to the failure
 - They are very useful if the failure is distant (in logic or time) from the fault or the fault is complex
- If no such rare coverage points are recorded, then it is likely that the failure is found by chance
 - The verification plan needs to be refined to catch these failures

Kerstin Eder

COMSM0115 – Design Verification

40

Regression

Regression Suites

- A **regression suite** is a set of tests that are run on the verified design on a regular basis
 - After major changes
 - Periodically: Every night or every weekend
- Regression goals
 - Assuring that things that worked did not stop working
 - This is vital because every bug fix, on average, introduces one fifth of a bug
 - Detecting “unexpected” bugs

Kerstin Eder

COMSM0115 – Design Verification

42

Types of Regression

- **Static regression**
 - The regression suite is comprised of a set of “interesting” test patterns
 - Tests that found bugs in the past
 - Tests that reach corner cases
- **Random regression**
 - A.k.a. dynamic regression, probabilistic regression
 - The regression suite is comprised of a set of test specifications and an execution policy
 - For example, execute 100 tests of specification A, 35 tests of specification B, and 20 tests of specification C

Kerstin Eder

COMSM0115 – Design Verification

43

Static Vs. Random Regression

- **Static regression**
 - ☑ **Known, guaranteed quality**
 - ☒ **Sensitive to changes**
 - ☒ **Hard to maintain**
- **Random regression**
 - ☒ **Unknown quality**
 - ☑ **Less sensitive to changes**
 - ☑ **Easy to maintain**
 - ☑ **Easy to adapt to simulation resources**

Kerstin Eder

COMSM0115 – Design Verification

44

The Preferred Solution

- Combination of static and random suites
- **Small static suite** with hard to recreate cases
 - Hard to reach corner cases
 - Tests that discovered hard to find bugs
- **Random suites** for everything else

Kerstin Eder

COMSM0115 – Design Verification

45

Regression Suites Requirements

- A regression suite must be:
 - **Comprehensive** so that it is likely to catch all the bugs introduced
 - **Small** so that it can economically be executed many times
- **How can we make our regression suite small and comprehensive?**
- **Solution:** use coverage information
 - Select a set of tests that achieve 100% coverage (of the coverage achieved so far)
 - Select the smallest possible such set

Kerstin Eder

COMSM0115 – Design Verification

46

The Set Cover Problem

- Let $S = \{C_1, \dots, C_n\}$ be the **set of coverage tasks**
- Let $T = \{T_1, \dots, T_n\}$ be a **set of tests**
 - Each test T_i covers subset $\{C_{i1}, C_{i2}, \dots\}$ of the coverage tasks
- The **set cover problem**: Find the smallest subset of T that covers S
- The set cover problem is a known NP-Complete problem
 - However, there are a number of good algorithms for it

Kerstin Eder

COMSM0115 – Design Verification

47

Online Algorithm

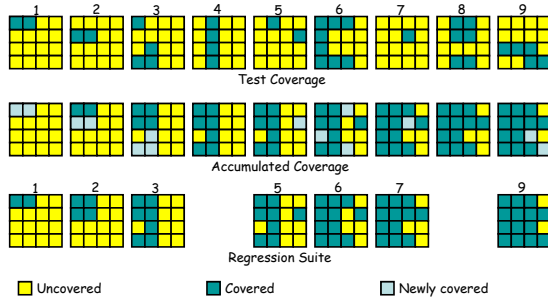
- For each new test T
 - If T covers an uncovered coverage task
 - Add T to the regression suite
- **Advantages**
 - Very simple
 - Low memory requirements

Kerstin Eder

COMSM0115 – Design Verification

48

Online Algorithm Example



Kerstin Eder

COMSM0115 – Design Verification

49

Greedy Algorithm

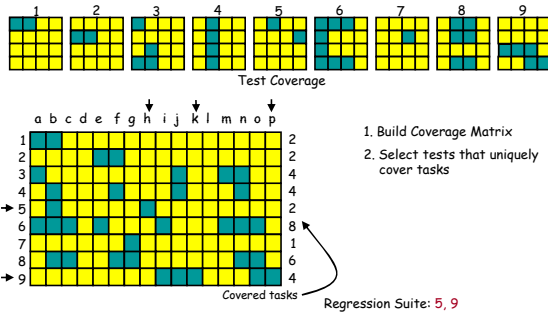
- Initialization
 - Build coverage matrix tests vs. tasks
 - Select tests that uniquely cover tasks
- Loop until all covered tasks are removed
 - Remove all the tasks covered by selected tests
 - Choose the test that covers most remaining tasks
- Advantages
 - Complexity is polynomial in the number of tests and coverage tasks
 - Quality solution
- Disadvantage
 - Requires to keep the entire coverage matrix in memory

Kerstin Eder

COMSM0115 – Design Verification

50

Greedy Algorithm Example

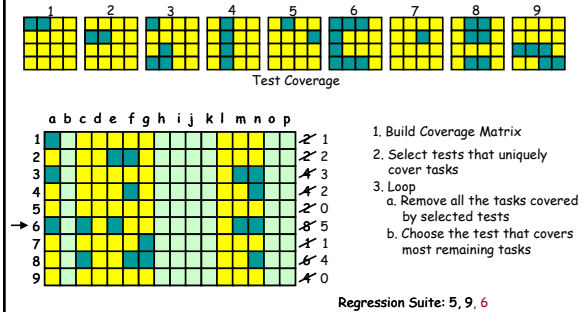


Kerstin Eder

COMSM0115 – Design Verification

51

Greedy Algorithm Example

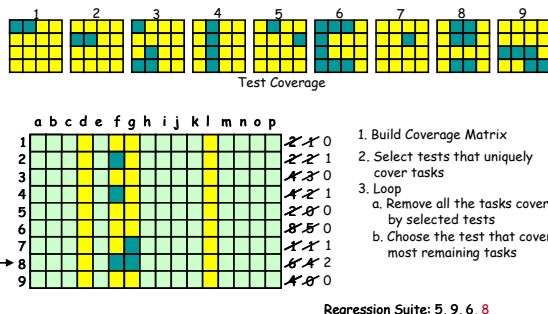


Kerstin Eder

COMSM0115 – Design Verification

52

Greedy Algorithm Example

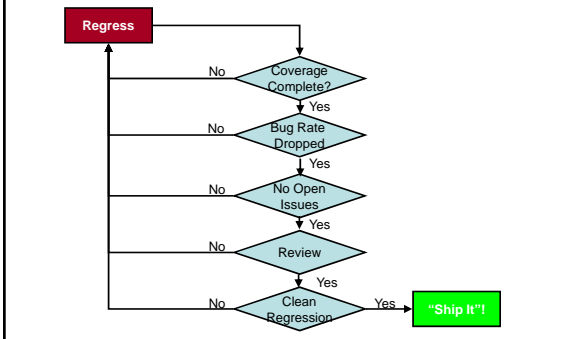


Kerstin Eder

COMSM0115 – Design Verification

53

When Is Verification Done?



Kerstin Eder

COMSM0115 – Design Verification

54

Tape-Out Readiness

- Before sending a design to manufacturing, it must meet established **tape-out criteria**
- The criteria is a series of checklists that indicate completion of planned work
- Verification is just one element in this series of checklists
- **Tape-out readiness is measured by a set of metrics**
- The most relevant metrics for verification are **bug rates and coverage**

Kerstin Eder

COMSM0115 – Design Verification

55

Escape Analysis

Escape Analysis

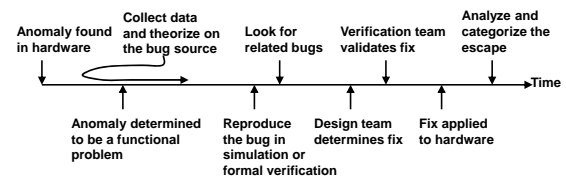
- An **escape** is a bug found later in the verification process than it should have been
 - In other words, it escaped its target place
 - Usually, escapes refer to bugs found in the hardware itself instead of during simulation
- Escape analysis has **two important aspects**
 - Make sure that the bug is fully understood and fixed correctly
 - We do not want another tape-out because of a bad fix
 - Understand why the bug escaped simulation in the first place and try to improve the verification plan and process to avoid such escapes in the future

Kerstin Eder

COMSM0115 – Design Verification

57

Individual Escape Analysis Timeline



Kerstin Eder

COMSM0115 – Design Verification

58

Summary

- Completion of the Verification Cycle includes:
 - Coverage analysis
 - Failure analysis
 - Regression
 - Tape-out readiness
 - Escape analysis

Kerstin Eder

COMSM0115 – Design Verification

59

Are We There Yet?



Kerstin Eder

COMSM0115 – Design Verification

60