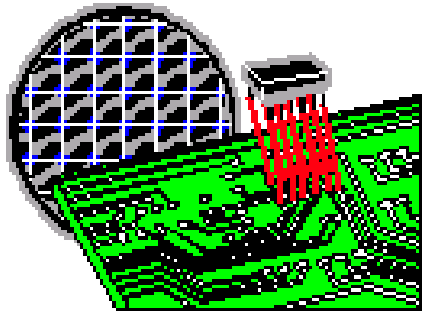
Code Coverage Tutorial



Semicon Confidential

Semicon IC Design Training Center

Questions

- Ask at any time
  - Just speak out
  - Raise your hand only if “Bus Contention”
- No such thing as a “dumb” question
- Audience Assumptions
  - Some knowledge about HDLs (Hardware Design Languages)
  - Some familiarity with RTL (Register Transfer Level)

What are Coverage Tools?

- Debugging aids which tells users how much of a RTL (Register Transfer Level) design has been exercised by a suite of verification tests
- Available in standard simulators (Modelsim and Active-HDL)
- Only work on RTL, not on primitives

Types of Coverage Tools

- Code Coverage (also called Statement Coverage)
- Branch Coverage
- Toggle Coverage

Code Coverage (Also called Statement Coverage)

- Code Coverage examines each executable statement and checks to see if the statement was executed during the simulations.
- If a line of code wasn't executed, its's a safe bet that you didn't catch any bugs in it

$$\text{Code Coverage} = \frac{\text{\# of statements executed}}{\text{\# of executable statements}}$$

Branch Coverage
(sometimes called Decision Coverage)

- Branch Coverage examines each branch of IF and CASE statements and checks to see if the branch was taken
- If a branch wasn't taken, its's a safe bet that you didn't catch any bugs in it

$$\text{Branch Coverage} = \frac{\text{\# of branches taken}}{\text{\# of branches}}$$

Toggle Coverage

- Toggle Coverage examines each signal to see if it was both ‘0’ and ‘1’
- Different modes of toggle coverage - check your simulator’s documentation for details

Using Coverage Results

- Results for several tests can be merged
- Results can be viewed with a graphical tool which shows which lines of code have or have not been exercised

Why Use Coverage Tools?

- Let you know it your tests have covered all of the design
- Adds very little, if any, overhead to your run time

Important Things to Remember

- Coverage tools do not grade your tests.
- High coverage percentages only tell you that exercised most of your code
- “Necessary, but not Sufficient”
- Need to sample outputs such that your tests actually detect changes in all of the exercised code

Testbench and Models

- Perform coverage on your testbench and models as well as UUT (unit under test)
- Unexercised paths in your testbench may tip you off to something that you didn’t test

Summary

- Coverage Tools are available in standard simulators (ModelSim and Active-HDL)
  - Code Coverage
  - Branch Coverage
  - Toggle coverage
- Very little overhead
- Need to make sure changes are propagated to outputs and sampled

Demo

- See the example result