

CSCE 611

Digital Systems Design

2004/2/11

Lecture 15

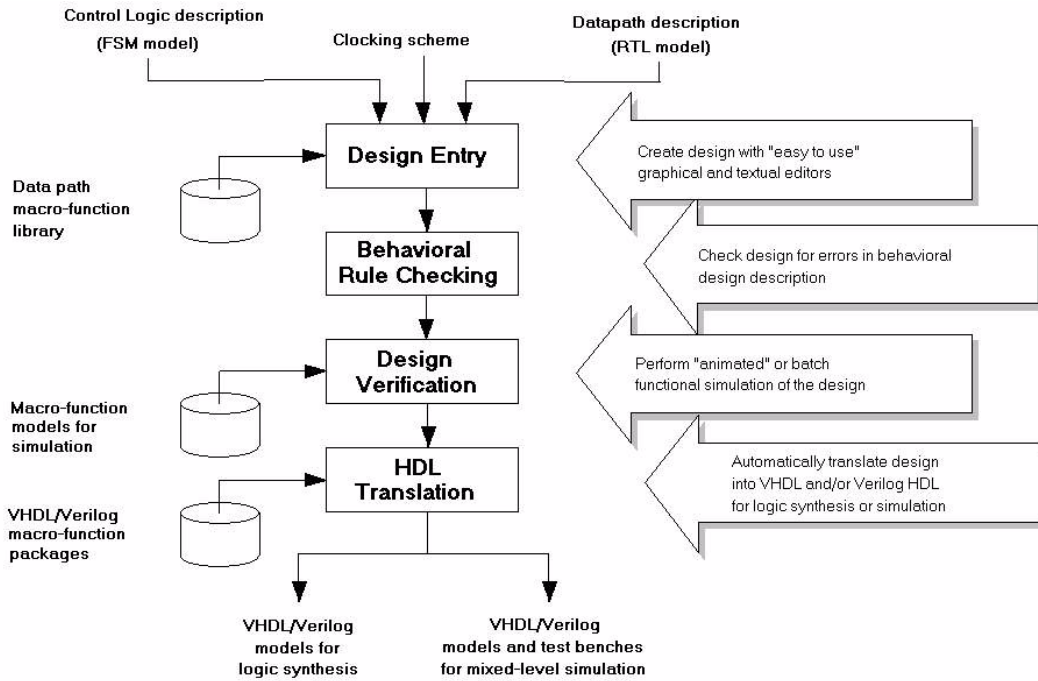
Digital Systems Design: Test Planning and Simulation Verification

© 2004 Dr. James P. Davis

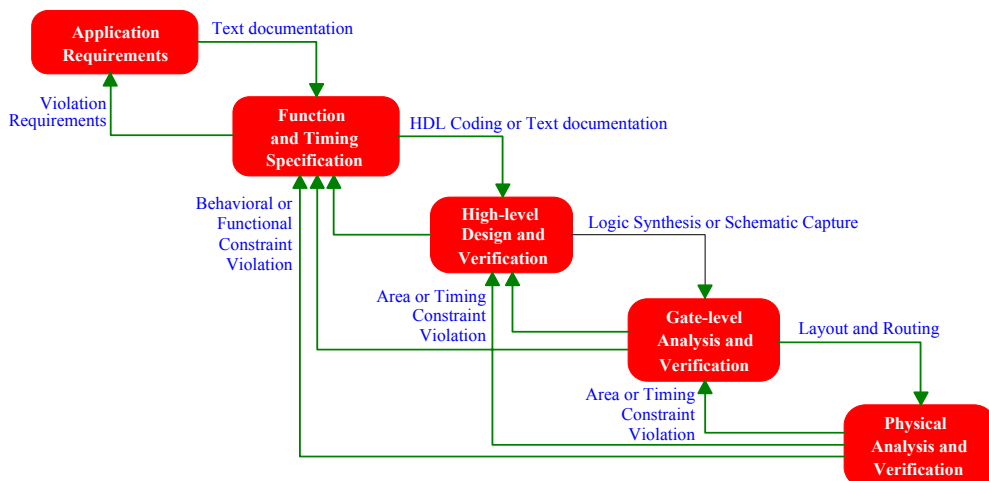
Lecture 15 - Outline

- Objective: To discuss the topic of design verification to give you more insight on:
 - How to test and debug a design model once the model has been created.
 - How to think about verification of digital systems using UML and ASM charts as test planning aids, not only to create the design model, but also to create the model for the “test harness” as well.
 - Fact: the distribution of effort between digital design and test activities is approximately 40% and 60%. So testing methods are important to overall design proficiency.
- Means:
 - ✓ The Up/Down Counter example.
- Result:
 - You should be able to simulate this design in Nimbus, stimulating the design inputs with counter values, and generating/printing a waveform as output.

Executable ASM Design Method Using Nimbus™



Iterative Refinement Design Process

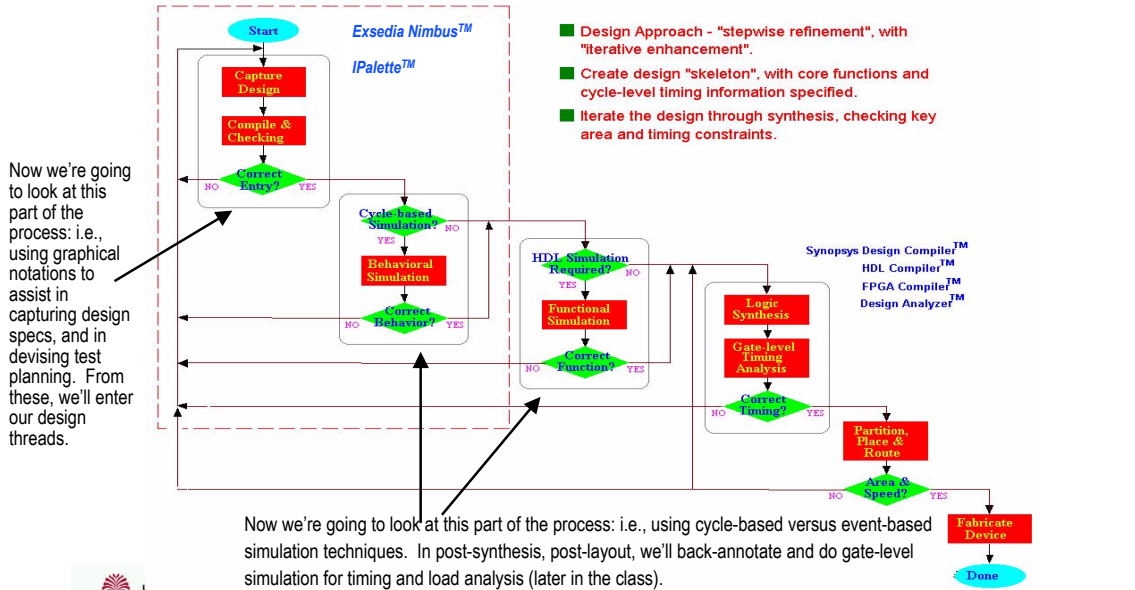


Problem	Goal	Approach
Behavioral or functional constraint violations cause 50-80% of cycling between design steps.	<ul style="list-style-type: none"> Eliminate unnecessary "cycling" through design steps. Improve the turnaround time per cycle. 	<ul style="list-style-type: none"> Support easy exploration of design alternatives. Allow function & behavior changes to be made quickly.



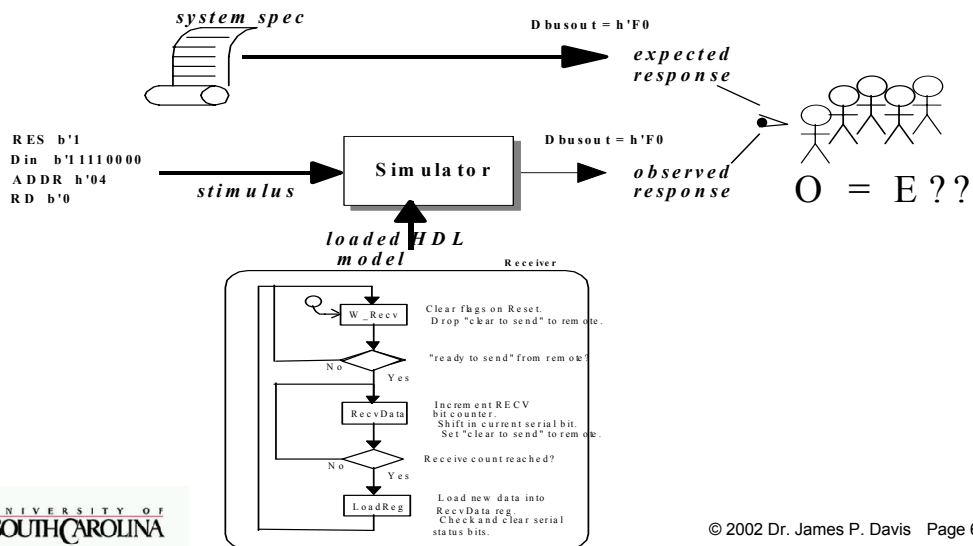
Systems Design Method – Design Process

- Conceptual Approach to Architecture and Design.
 - ✓ Create a design model, and iterate on its definition, incorporating debug and verification through simulation runs.

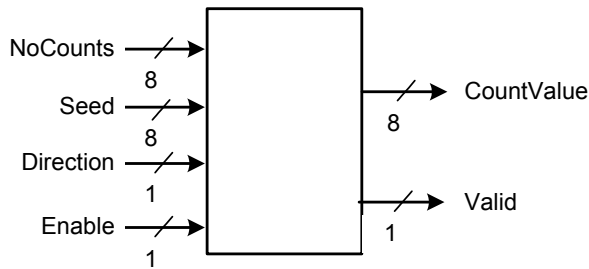


Systems Design Method – Test Process

- Approach to Verification
 - ✓ Create a set of stimuli that can be used to verify observed behavior against expected behavior.
 - ✓ We'll use different levels of design and test specification, to take advantage of fast turnaround in gaining functional and cycle-level timing verification, then obtaining gate-level timing and load analysis, post-layout. The same test harness will be used at both levels.



Test Example - Binary Up/Down Counter

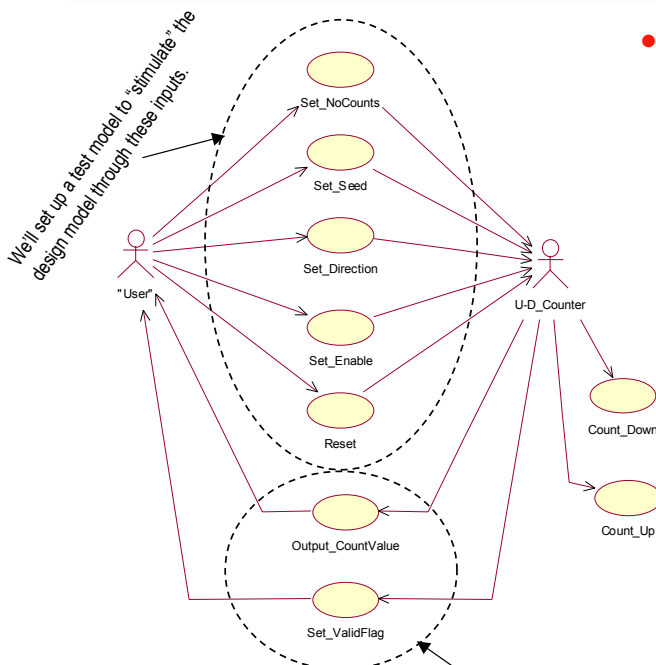


• The Binary Up/Down Counter

- ✓ We have discussed the basic function and operation of the counter.
- ✓ We have created a design model for its behavior.
- ✓ Now, we want to devise a means to test its correctness and accuracy:
- ✓ *Question 1:* does it perform the function we have intended for it to do?
- ✓ *Question 2:* how “robust” is the design in the face of different patterns of inputs?
- ✓ *Question 3:* how do we evaluate the design to insure that it does what it should do?



Creating a Test Plan – Use Case Diagram



We'll set up a test model to "stimulate" the design model through these inputs.

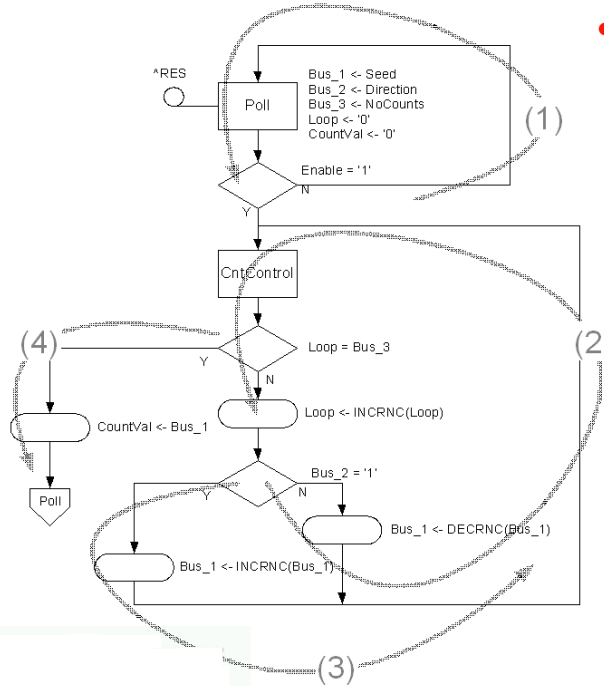
• Use cases and actors:

- ✓ Use Cases have been used in the analysis part of our process.
- ✓ We now want to employ them in helping us to identify and articulate specific test scenarios, and localize test “points” in the design model.
- ✓ In the analysis model, we identified the “user” as an entity external to the Counter; we’ll create a test model that provides some of its functionality for purposes of feeding test data to our design model.
- ✓ The Use Case model gives us the set of actions and interactions we’ll need to be concerned with.



Creating a Test Plan – ASM Design Model

- In examining an ASM “thread”, we want to evaluate which logic paths we can test by defining a set of input stimuli to excite the design, so that each run allows different logic to be tested.



- **Test points:**
 - ✓ We start by looking at specific points allowing us to check different logic paths through the design.
 - ✓ We look at the signals and the value ranges that will cause our design to choose a different logic path.
 - ✓ We then create a set of stimulus “steps” that will allow us to trace the design in simulation.

© 2002 Dr. James P. Davis Page 11

The ASM Test Case Planning Worksheet

Instructions: For each test case scenario, fill in one row in the table. Number each test case. Give each one a meaningful test name. For each case, indicate as precisely as you can, (1) the input bus set-up to trigger your test, and (2) the expected results of your test run, in terms of bus values you should see, and on what simulation clock cycles you expect to see these values. The simulation clock cycle should be indicated for intermediate values, or when you are watching an output bus change value multiple times during a single test run. These fields get filled out **before** you execute the simulation run. The field for observed description of the simulation run results is filled in **after** you simulate. This should describe the overall results. This would provide a narrative that goes along with the simulation waveform output.

Test No.	Test Name	Simulation Input			Expected Results			Observed Outcome (in text)
		Elapsed Cycle #	Input Bus Name	Value	Elapsed Cycle #	Bus Name	Value	