Lab Lecture 5
Debugging
9/26/2003

Simulation vs. Hardware Debug
- Debugging in Simulation:
  - Slow run time
  - Fast debug time (waveform, text)
  - Very high coverage (90-100%)
  - Easy to fix!
- Debugging in Hardware:
  - Fast run time
  - Slow debug time
  - No or little internal visibility
  - It might be too late to fix (ASICs)

Test Case Generation
- Complete Coverage: Exhaustive Testing
- FSM: Test all Transitions and States
- In practice: Goal is to get 95%+ Coverage
- Correct by Design: Only the Coverage Specified by Design

Part I Bottom Level Testing (1)

```
one_bit_comp

\[ \text{g0 (a > b)} \]
\[ \text{e0 (a = b)} \]
```

What does it mean if (g0 == 0) and (e0 == 0) ?

Part I Bottom Level Testing (2)

- Ideal Testing: Exhaustive Testing
  - Every possible input combination has been tested. Can we do it here?
- Determine the truth table first
- Write a testbench that test every combination of the circuit
- This circuit is 100% tested!
Part I Intermediate Level Testing (1)

four_bits_comp

a →

one_bit_comp

b →

one_bit_comp

greater

too

Part I Intermediate Level Testing (2)

- Can we do Exhaustive Testing Here?
- Two sets of four bit inputs:
  - 16 * 16 = 256 possible inputs/outputs
- Use a for or a while loop to go through all 256 combinations

Part I System Level Testing (1)

```plaintext
module main;
    input [3:0] in1; input [1:16];
    initial
        begin
            // read the file specified and put the values in 'in1'
            // readmemh("data_vector", in1);
            for(j=1; i<16; i++)
                begin
                    // Remember to advance the time forward
                    n = n+4;
                    in = in+1;
                    //display("in = %d, peak = %d", in, peak);
                end
            end
        end
endmodule
```

Part I System Level Testing (2)

- Read a test vector from a file
- Make sure you get a good coverage with your test vectors

Part II Test an Adder in FPGA

Part II continued

- Design a test harness (checker) that instantiates the adder module
- Try every combination of inputs to determine whether the adder works correctly in each failure mode
- Checker stops when an error is detected
Part III Test FSM

- Exercise every transition
- Verify states and outputs
- Don’t have to correct errors

Part IV Probe Internal Signals

- Simulate the circuit to make sure it works properly
- Set probe points on internal signals
- Download to the board and probe internal signals using logic analyzer