

Lab Lecture 9
Checkpoint2 Part2
3/19/2004

Greg Gibeling

Today

- Debugging
- Checkpoint2
- Announcements
- Synplify
- Undeclared Wires
- Signal Conditioning
- Timing Analyzer

Debugging (1)

- Debugging Algorithm
 - Hypothesis: What's broken?
 - Control: Give it controlled test inputs
 - Expected Output: What SHOULD it do?
 - Observe: Did it work right?
 - If it broke: THAT'S GREAT!
 - If we can't break anything like this then the project must be working...

Debugging (2)

- Don't debug randomly
 - Just changing things at random often makes things look fixed
 - It won't really help
 - Debug systematically
 - Your first design may be the best
 - "1000 CS150 students at a 1000 typewriters..."
- What can you do?

Debugging (3)

- High Level Debugging
- Localize the problem
 - PGen1 ? FIFO1? SDRAM? FIFO2? PGen2?
 - Test Patterns
 - Lets you easily isolate the broken component
 - If you know exactly what's going in you can check what's coming out

Debugging (4)

- Simulate the broken component(s)
 - Writing test benches takes less time than sitting around wondering why its broken
 - Everyone hates writing testbenches
 - (Even me)
 - Get used to it
- What if the simulation works?
CHIP SCOPE!

Debugging (5)

- Using the logic analyzer / ChipScope
 - The most reliable tool you have
 - When used properly
 - Use the triggers effectively
 - Trigger on recurring sequences
 - Trigger on errors
 - An unstable display is useless
 - Compare synthesis to simulation
 - ChipScope is almost as good as simulation

Debugging (6)

- Your best debugging tool is logic
 - If 3 out of 4 components work, what's broken?
- Question all your assumptions!
 - Just because you think its true doesn't mean it is
 - 90% of debugging time is wasted debugging the wrong problem otherwise
 - Given solutions and modules may not work the way you expect!

Debugging (7)

- Before you change anything
 - Understand exactly what the problem is
 - Find an efficient solution
 - Evaluate alternative solutions
- After the change
 - Fixes may make things worse sometimes
 - May uncover a second bug
 - May be an incorrect fix
 - Repeat the debugging process

Debugging (8)

- Ask around
 - Someone else may have had the same bug
 - They'll probably at least know about where the problem is
 - Different bugs may produce the same results
- TAs
 - The TAs know common problems
 - We've also made a lot of the mistakes

Checkpoint2 (1)

- AC97Codec had a bug!
 - The Clock was at 6MHz, Oops
- The "Decode" blocks on the diagram
 - Just comparators
 - They generate the write/read/control signals from the Bit/Slot Counters
- Analog loopback:
 - You should be able to do this with 3 register writes

Checkpoint2 (2)

- Simulation/Synthesis Mismatch!
 - SDataIn will arrive a cycle early on board
 - SDataIn will arrive "on time" in ModelSim
 - A circuit that works in simulation will be off by a cycle when you put in on the board
 - Because of this you might never get a CodecReady unless you "fix" your verilog

Announcements (1)

- Checkpoint2
 - Part1 due the week after break (its easy)
 - Part2 due the week after that
- Midterm II
 - Tuesday March 30th in Class
 - Review is Sunday March 28th 4-6pm in the lab (it will be webcast/online)
- Homework Solutions will be up soon

Announcements (2)

- All deadlines are posted to the website
 - Please read this
 - The week after break will be a little tough
 - But the project deadlines have to be final, we cannot push them back
- No office hours next week
 - TAs might be around, but don't count on it

Synplify Warnings (1)

- You need to get rid of these!
- Ignore ones in (complete) modules that we've given you
- Warnings in your modules may explain why it doesn't work

Synplify Warnings (2)

- Pruning Sequential Instance
- Combinational Loop
- Latch Generated
 - Too many clocks
- Incomplete Sensitivity List
- Others...

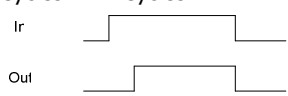
A Dire Warning!

- Wire misspellings can be FATAL
 - ModelSim/Synplify will assume a 1bit wire
 - This is part of the Verilog standard
 - The tools wont even warn you...
- Be very careful with this!
 - Have your partner double check your code
 - Be suspicious of blue lines in ModelSim

Signal Conditioning (1)

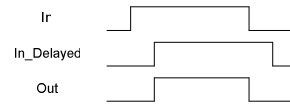
- Off-by-a-cycle Errors
 - Shorten a Pulse
 - Lengthen a Pulse
 - Shift a Pulse
- Remember the Edge Detector?
 - Its something like that...

Signal Conditioning (2)

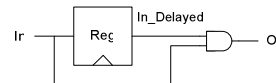
- Shorten a Pulse
 - 5 Cycles -> 4 Cycles
- 
- Any guesses?
 - What if we delayed the input?

Signal Conditioning (3)

- Shorten a Pulse

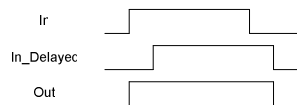


- Out = In & In_Delayed

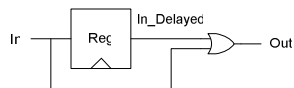


Signal Conditioning (4)

- Lengthen a Pulse



- Out = In | In_Delayed



Timing Analyzer (1)

- Timing Constraints in Synplify
 - Xilinx will attempt to match them
 - Will tell you if it fails
 - They make PAR run slower
 - A constraint will not make your circuit faster
- Better to just let things PAR
- Check the timing when we're done

Timing Analyzer (2)

- "Analyze Post Place and Route Static Timing (Timing Analyzer)"
 - Implement Design -> Place & Route -> Generate Post Place and Route Static Timing
- This will tell you your minimum period
- If its too big then what?
 - Simplify your circuit
 - Constraints probably won't do it