Simulation

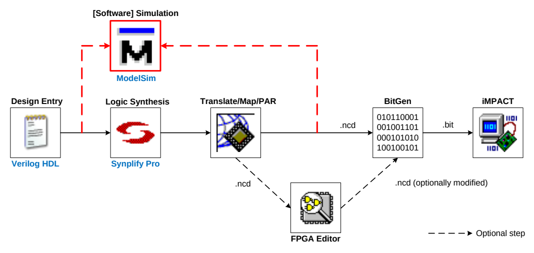
UCB EECS150 Spring 2010

Lab Lecture #4

Agenda

- Simulation
  - Software simulation vs. Hardware testing
  - Administrative Info
  - General testing methodologies
    - Bottom-up, top-down
    - {Exhaustive, random} testing
    - FSM testing

Where are we?



Test Methodologies (Macro) (1)

- Why simulate in *software*?
  - Design feedback
    - Waveforms
    - Text messages
  - Design visibility
    - Examine every signal!
  - Fast startup time
    - Functional simulation comes before you run the tools
  - Slow running time
    - It takes 1 s to simulate 1 ms

Test Methodologies (Macro) (2)

- Why test in *hardware*?
  - Fast running time
  - Slow startup time
    - Running the tools takes from minutes to hours
  - [Lack of] design visibility
    - Can only probe several signals at a time (more on this in Lab 5)
  - Subject to “hard” errors (broken FPGA)

Test Methodologies (Macro) (3)

- When to use software/hardware for testing
  - Software simulation: Functional testing
    - Very that the circuits behaves correctly
      - Outputs correct waveforms
      - Interacts with neighboring blocks correctly
      - Fully verify to the extent that you can!
  - Hardware testing: Live debugging
    - When it works in simulation but not in hardware…
    - Use as a last resort – it is time consuming
    - Is all hope lost? *No.* This will be the subject of lab 5.

Administrative Info

- Newsgroup
  - Post questions!
  - We don't bite! [?]
- SVN is functional
  - Email Ilia with any problems
  - Homework is only accepted through SVN
- Coming up…
  - Mini-project (week after next)
  - Actual project (week after that)

Test Methodologies (Micro) (1)

- Bottom-up testing
  - Design is a tree of modules
  - Test and verify the "leaves" first, and move up
    Example: Broken accumulator
    1. Test each function in an ALUBitSlice
    2. Verify the entire ALUBitSlice
    3. Make sure the registers are instantiated correctly
    4. Test the entire accumulator (again)

Circuits seldom "just work" unless you test individual pieces!

Test Methodologies (Micro) (2)

- Top-down testing
  - Recap: don't do this initially
  - Some bugs don't occur when you test pieces apart
  - Thus, test procedure really is:
    1. Bottom-up (Simulation)
    2. Top-down* (Simulation)
    3. Top (Hardware): See if it works
    4. Top-down (Hardware): If it doesn't work…

*Take the time to build a simulation model of your FPGA_TOP file.
This is the *crucial (and most overlooked)* step.

Test Methodologies (Micro) (3)

- Exhaustive testing
  - Find out input combinations for a circuit
  - Drive circuit with every combination of inputs
    - "for" loops can handle input generation
  - Ideal case: test every input (100% coverage)
    - Not often tractable (circuits with many inputs)
    - Do the best you can through *targeted test vectors*
  - The Good
    - Ideal testing – covers all cases
  - The Bad
    - Very slow – complexity does not scale with design size

Test Methodologies (Micro) (4)

- Random testing
  - After using targeted test vectors…
  - "Lazy man's method"
    - Feed random inputs into design for X days
    - Does it still work? Great! It must be bug free.
    - After targeted test vectors, this works surprisingly well
  - The Good
    - Doesn't require much thought to set up
    - Is decently effective
  - The Bad
    - Isn't necessarily going to test frequently occurring problems

Test Methodologies (Micro) (5)

- FSM testing
  - The method from lab 3
    - Check every state transition arc
    - Check every output
  - Leverage what you have
    - Make this about iterating through input domain

Acknowledgements & Contributors

Slides developed by Chris Fletcher & John Wawrzynek (2/2010).

This work is based in part on slides by:
 Chen Sun (2008-2009)
 Sarah Swisher (2008)
 Greg Gibeling (2003-2005)

This work has been used by the following courses:
- UC Berkeley CS150 (Spring 2010): Components and Design Techniques for Digital Systems