Structural Verilog

EECS150 Spring2009 - Lab Lecture #2

Chris Fletcher

Slides designed by Chris Fletcher

Slides: "Verilog (1)", "Verilog (2)" and "CAD + Verilog" by Greg Gibeling

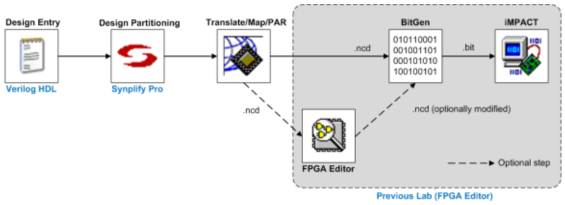
Today

- CAD Flow Extension
- Verilog
- Structural Verilog
- Administrative Info
- Lab #2: The Structural Accumulator
  - Lab2 circuit
  - Lab2 testing
  - Analysis of resource usage and timing

CAD Flow Extension (1)

- This week
  1. FPGA Editor
  2. BitGen
  3. iMPACT
- Next week
  1. Design Entry
  2. PPR
     - Design Partitioning
     - Place and Route
  3. See "This week" section

CAD Flow Extension (2)



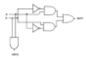
Verilog (1)

- What's an HDL?
  - Textual Description of a Circuit
  - Human and Machine Readable
  - Hierarchical
  - Meaningful Naming
- NOT A PROGRAM
  - Describe what the circuit IS
  - Not what it DOES

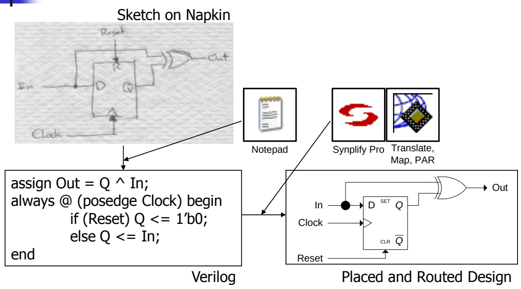
Verilog (2)

Digital Design Productivity, in Gates/Week

Source: DataQuest

Behavioral HDL		2K-10K
RTL HDL		1K-2K
Gates		100-200
Transistors		10-20

CAD + Verilog: The Big Picture



```
assign Out = Q ^ In;
always @ (posedge Clock) begin
        if (Reset) Q <= 1'b0;
        else Q <= In;
end
```

Structural Verilog (1)

- Verilog Subsets
  - Structural: primitive gates + modules
    - Gate level design
    - **You will ONLY use Structural Verilog in this lab**
  - Dataflow: compact boolean expressions
    - More compact expression of structural Verilog
  - Behavioral: abstract syntax
    - Timing nuances
    - You will see this starting next lab

Structural Verilog (2)

- Structural 2:1 Mux example

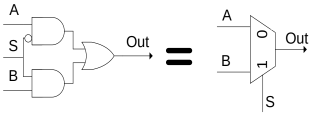
```
module Mux21(A, B, S, Out);
       input wire A, B, S;
       output wire Out;

       wire notS, ATemp, BTemp;

       not invertS(notS, S);
       and and(ATemp, A, notS),
           andB(BTemp, B, S);
       or  result(Out, ATemp, BTemp);
endmodule
```

Key

Module wrapper
Input/Output wire declarations
Wire declarations
Gates



Administrative Info

- Homework submission policy
- Lab lecture conflicts
- Card key access
- Check-off procedure

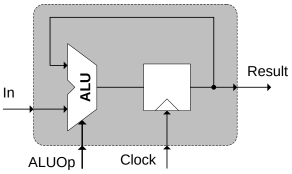
- Questions?

Lab #2 (1)

- Build a structural Accumulator
  - Work with a real design
  - Write parameterized Verilog
- Debugging
  - Synplicity RTL/Technology schematic
- Analysis
  - Resource consumption
  - Timing

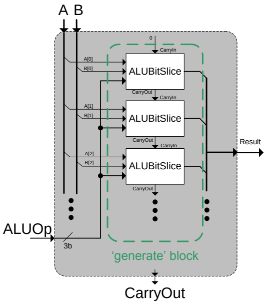
Lab #2 (2)

- Structural Accumulator
  - ALU
  - ‘FDRSE’ Xilinx primitive flip-flop



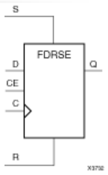
Lab #2 (3)

- ALU
  - We provide the Verilog for N-bit version
  - You must implement the 1-bit ALU model
  - **Must** support our ALUOp
  - Supports: +, -, &, |, ~, pass-through (7 operations)



Lab #2 (4)

- FDRSE
  - Xilinx primitive
  - D-type flip-flop
    - Instantiated like a simple module
    - Specific Set/Reset characteristics
  - "Read all about it!" → virtex5_hdl.pdf
    - It's part of the PreLab!

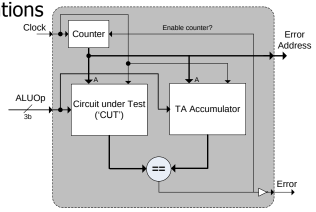


Lab #2 (5)

- Accumulator
  - We give you port specification
  - You will implement the rest of the circuit
- Use code examples
  - Mux21: Structural Verilog (gates, wires)
  - ALU: `generate` statements
- Abide by our interfaces!

Lab #2 (6)

- HW test harness
  - TA Accumulator vs. your Accumulator
  - Check all input combinations
  - Signal error and show address if mismatch



Lab #2 (7)

- Circuit Analysis
  - Resource Usage
    - Accumulator(width) = how many LUTs / SLICEs?
    - `generate` allows you to experiment
  - Timing
    - Locate nets → **"Technology Schematic"**
    - Calculate delay on the nets → **FPGA Editor**

Lab #2 (8)

- PreLab
  - Read specified material
  - **Write all of your Verilog**
- Lab starts at debugging phase
  - Assumption:

    you have written all of your Verilog ahead of time