EECS 150 Spring 2004

Lab Lecture 3

Verilog Simulation Mapping

2/6/2004

Michael Liao
Greg Gibeling
Sandro Pintz

Today

- Lab Motivation
- Overall Design
- Practical Verilog
- Think Hardware
- Behavioral vs Structural Verilog
- Lab #2

Motivation

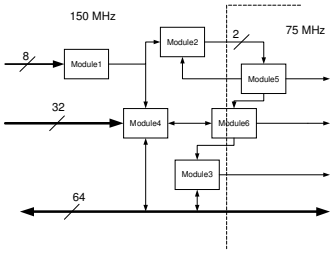
- Basic Introduction to HDL Design Entry (Covered in Main Lecture)
- Architectural Definition and Modularization (Partitioning)
- What are "Good Verilog Techniques"
- Behavioral vs Structural Verilog

Top Down Architecture (1)

- Top Down Refinement Process
- Start Here:

Top Down Architecture (2)

- End Here:



Partitioning

- Define modules that:
  - Have clean, well defined interfaces to other modules
  - Are manageable in size
  - Can be verified independently
  - Might be designed by different people
  - Functionally makes sense

Practical Verilog (1)

- Remember :
  - "Think Hardware" as you write
  - Meaningful Signal Names:
    - clockEnable, busSync (good)
    - A, B&*_ (not good)
  - Choose a naming style and STICK TO IT
  - Match modules and filenames, 1 module per file

Practical Verilog (2)

- Clear Comments:
  - Comment a whole section
  - Comment for special cases (exceptions)
  - Comments for special signals
- Choose a Code Structure and STICK TO IT
  - State machine style
  - Special section for flops
  - Special section for structural stuff (instantiations)

Think Hardware (1)

- You are NOT programming
  - Easiest way to waste time
  - Programming will only confuse you
- You are describing hardware
  - You should have a circuit in mind
  - Know what you want BEFORE YOU WRITE IT
  - Don't try to hack it together
    - You will not be happy with the results

Think Hardware (2)

- Hardware isn't like software
  - Things happen all at once
  - There is no sequential execution model
  - You are not writing a set of instructions
  - You are basically creating a schematic in text form
- ALWAYS REMEMBER THIS!

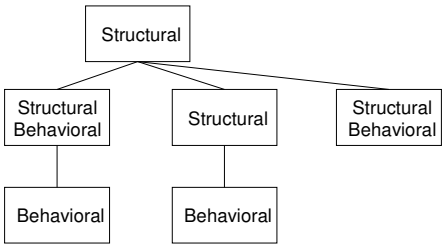
Example

```
////////////////////////////////////
// Module Description, Company Information
// Copyright Notice, Log of Changes
// Cvs version
////////////////////////////////////
module lab2_clr (IN, E, R, CLK, OUT);

////////////////////////////////////
// Inputs
////////////////////////////////////
  input [7:0] IN;          // comment
  input       E, R, CLK;   // comment

////////////////////////////////////
// Outputs
////////////////////////////////////
  output [7:0] OUT;        // comment

////////////////////////////////////
// Declarations
////////////////////////////////////
  reg [7:0]    OUT;        // comment

////////////////////////////////////
// Behavioral Code
////////////////////////////////////

////////////////////////////////////
// Module Instantiations (Structural)
////////////////////////////////////

////////////////////////////////////
// Flip flops
////////////////////////////////////
always @(posedge CLK) begin
  if (R)
          OUT <= 0;
  else if (E)
          OUT <= OUT + IN;
end

endmodule
```

Behavioral vs Structural (1)

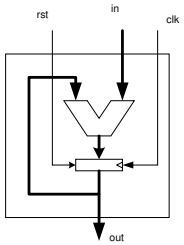
- Rule of thumb:
  - Behavioral doesn't have sub-components
  - Structural has sub-components:
    - Instantiated Modules
    - Instantiated Gates
    - Instantiated Primitives
- Most levels are mixed

Behavioral vs Structural (2)



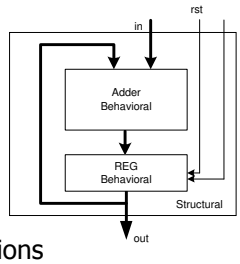
Lab #2 Part I

- Behavioral Only
- No Instantiations

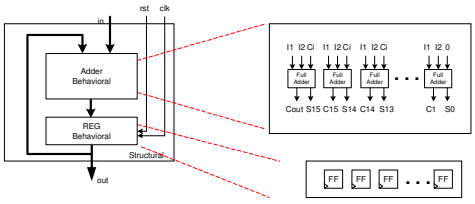


Lab #2 Part II

- Behavioral:
  - Adder
  - Register
- Structural:
  - Top
  - Two Instantiations

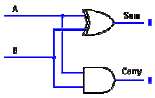


Lab #2 Part III



Lab #2 Part IV

- Half Adder



- Full Adder