Lab Lecture 3
*Verilog Simulation Mapping*
2/6/2003

Sandro Pintz

---

**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 ARCH (1)**
- Top Down Refinement Process
- Start Here:

```
  Inputs  Project  Outputs
```

---

**TOP DOWN ARCH (2)**
- End Here:

```
  Inputs  Project  Outputs
```

---

**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)

Example

```
module adder (i1, i2, o);
  input i1, i2;
  output o;
  reg [1:0] reg1, reg2;
  reg [1:0] reg3;
  always @ (posedge i1 or posedge i2)
    begin
      reg1 = i1;
      reg2 = i2;
      reg3 = reg1 + reg2;
      o = reg3;
    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)

```
module behavioral
  input in;
  output out;
  reg [4:0] reg1, reg2;
  reg [4:0] reg3;
  always @ (posedge in)
    begin
      reg1 = in;
      reg2 = in;
      reg3 = reg1 + reg2;
      out = reg3;
    end
endmodule
```

Part I

- Behavioral Only
- No Instantiations

Part II

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

\[
\begin{array}{c}
\text{Adder} \\
\text{Behavioral} \\
\text{Structural} \\
I_1 \ I_2 \ I_3 \ C_i \ S_0 \\
. . . \\
\text{FF} \\
. . . \\
\text{FF} \\
\end{array}
\]

Part IV

- Half Adder

- Full Adder