What did we forget? - Always Block

- Use System Verilog "always_comb"
- Use blocking assignment only
- System Verilog always_comb infers sensitivity list

```
always_comb
   if(a)
      z = x;
   else
      z = y;
```

What did we forget? - Always Block

- Use System Verilog "always_ff"
- Use non-blocking assignment only

```
always_ff @ (posedge clk, negedge reset_n)
   if(!reset_n) q <= '0;
   else         q <=  d;
```

- Any assignment under the always_ff clause infers flip-flops
- Assignment under "always @ (posedge clk...)" also!

What did we forget? - Counter and Shift Regs

- Counters and shift regs are coded differently from state machines
  - Mixed combo logic and flip-flops under always_ff clause

```
//simple counter
always_ff @ (posedge clk, negedge reset_n)
  if(~reset_n) q_out <= 0;
  else if(en)  q_out <= q_out + 1;
```

What did we forget? - State Machines

- Do it like this!

```
module arbiter0(
  output reg gnt,
  input      clk,    //clock input
  input      rst_n,  //asynchronous active low reset
  input      dly,    //delay input
  input      done,   //done input
  input      req     //request input
  );

//define enumerated types and vectors for ps, ns
  enum reg [1:0]{
    IDLE  = 2'b00,
    BBUSY = 2'b01,
    BWAIT = 2'b10,
    BFREE = 2'b11,
       XX =    'x } arbiter_ps, arbiter_ns;

//infer the present state vector flip flops
  always_ff @(posedge clk, negedge rst_n)
    if (!rst_n) arbiter_ps <= IDLE;
    else        arbiter_ps <= arbiter_ns;
```

```
always_comb begin
  arbiter_ns = XX;   //default, ns vector
  gnt        = 1'b0; //default, output signal
  case (arbiter_ps)
    IDLE :
      if (req)       arbiter_ns = BBUSY;
      else           arbiter_ns = IDLE;
    BBUSY: begin
      gnt = 1'b1;    //assert gnt
      if (!done)     arbiter_ns = BBUSY;
      else if ( dly) arbiter_ns = BWAIT;
      else           arbiter_ns = BFREE;
    end
    BWAIT: begin
      gnt = 1'b1;    //assert gnt
      if (!dly)      arbiter_ns = BFREE;
      else           arbiter_ns = BWAIT;
    end
    BFREE:
      if (req)       arbiter_ns = BBUSY;
      else           arbiter_ns = IDLE;
    endcase
  end //always
endmodule
```

What did we forget? - Coding

- Keep Intent Clear
  - Clear intent implies transparency
  - Keep it simple as possible
  - Robustness is the child of transparency and simplicity
    - Eric Raymond

What did we forget? - Coding

- Keep Intent Clear
  - Clear intent implies transparency
  - Keep it simple as possible
  - Robustness is the child of transparency and simplicity
    - Eric Raymond
- If your code is good, someone else will read it.
  - Code accordingly

What did we forget? - Coding

- Keep Intent Clear
  - Clear intent implies transparency
  - Keep it simple as possible
  - Robustness is the child of transparency and simplicity
    - Eric Raymond
- If your code is good, someone else will read it.
  - Code accordingly
- Use sensible names. It really does matter

What did we forget? - Coding

- Keep Intent Clear
  - Clear intent implies transparency
  - Keep it simple as possible
  - Robustness is the child of transparency and simplicity
    - Eric Raymond
- If your code is good, someone else will read it.
  - Code accordingly
- Use sensible names. It really does matter
- Keep comments tracking with code changes

What did we forget? - Coding

- Code doesn't work right?
  - Go back to your block diagrams

What did we forget? - Coding

- Code doesn't work right?
  - Go back to your block diagrams
  - Check your state machines

What did we forget? - Coding

- Code doesn't work right?
  - Go back to your block diagrams
  - Check your state machines
  - Check your timing diagram

What did we forget? - Coding

- Code doesn't work right?
  - Go back to your block diagrams
  - Check your state machines
  - Check your timing diagram
  - Then check your code

What did we forget? - Coding

- Code doesn't work right?
  - Go back to your block diagrams
  - Check your state machines
  - Check your timing diagram
  - Then check your code
  - Don't keep on hacking using "design by simulator"

What did we forget? - Coding

- Code doesn't work right?
  - Go back to your block diagrams
  - Check your state machines
  - Check your timing diagram
  - Then check your code
  - Don't keep on hacking using "design by simulator"
- Don't go too low in the abstraction level: no gates

What did we forget? - Coding

- One module per file
- MiXed CasEs caN bE dEAdLy. soMe TOolS are CaSE blInd!
  - Eric's examples were all lower case!
  - Constants should be ALL CAPS
- Don't try out all the features of System Verilog
- If you can't draw it, don't code it
- This ain't just another language, you're makin' hardware
- Bad RTL + synthesis = steaming pile of gates

What did we forget? - Debugging

- Read the cotton pickin' transcripts!

What did we forget? - Debugging

- Read the cotton pickin' transcripts!
- Read what the shell is telling you as well as the tools

What did we forget? - Debugging

- Read the cotton pickin' transcripts!
- Read what the shell is telling you as well as the tools
- Google error messages. That's right! Google'em!

What did we forget? - Debugging

- Read the cotton pickin' transcripts!
- Read what the shell is telling you as well as the tools
- Google error messages. That's right! Google'em!
- Fix the first error you find. Then fix those that follow

What did we forget? - Debugging

- Read the cotton pickin' transcripts!
- Read what the shell is telling you as well as the tools
- Google error messages. That's right! Google'em!
- Fix the first error you find. Then fix those that follow
- At the gate level, you only get to see the top level pins

What did we forget? - Debugging

- Read the cotton pickin' transcripts!
- Read what the shell is telling you as well as the tools
- Google error messages. That's right! Google'em!
- Fix the first error you find. Then fix those that follow
- At the gate level, you only get to see the top level pins
- Modelsim is hierarchical, descend to lower blocks when necessary

What did we forget? - Debugging

- Read the cotton pickin' transcripts!
- Read what the shell is telling you as well as the tools
- Google error messages. That's right! Google'em!
- Fix the first error you find. Then fix those that follow
- At the gate level, you only get to see the top level pins
- Modelsim is hierarchical, descend to lower blocks when necessary
- Save wave setups in wave.do

What did we forget? - Debugging

- Read the cotton pickin' transcripts!
- Read what the shell is telling you as well as the tools
- Google error messages. That's right! Google'em!
- Fix the first error you find. Then fix those that follow
- At the gate level, you only get to see the top level pins
- Modelsim is hierarchical, descend to lower blocks when necessary
- Save wave setups in wave.do
- "Somebody said the other way didn't work, this seems easier..."

What did we forget? - Debugging

- Read the cotton pickin' transcripts!
- Read what the shell is telling you as well as the tools
- Google error messages. That's right! Google'em!
- Fix the first error you find. Then fix those that follow
- At the gate level, you only get to see the top level pins
- Modelsim is hierarchical, descend to lower blocks when necessary
- Save wave setups in wave.do
- "Somebody said the other way didn't work, this seems easier..."
- "Divide and conquer, simplify the case, THINK"

What did we forget? - Debugging

- Linux commands: Google and read
  - i.e., "compare files Linux"

What did we forget? - Debugging

- Linux commands: Google and read
  - i.e., "compare files Linux"
- Don't try "random stuff". Don't confuse action with motion. We're engineers, not the Federal Reserve

What did we forget? - Debugging

- Linux commands: Google and read
  - i.e., "compare files Linux"
- Don't try "random stuff". Don't confuse action with motion. We're engineers, not the Federal Reserve
- Please learn a programming editor. It hurts to watch some of you