EE345

Verilog: Procedural Programming

If

- Can make a decision with “if statement”
- Controls a single item
  - But that item can be a block of items too

Syntax: condition in parens

```
if(x == y)     // If X and Y have same value…
  z = 3'b000; //     Then z gets 000
```

Note begin/end block

```
if(x == y) begin
  z = 3'b000;
  alpha = gamma >> 1;
end
```

Verilog ugliness: no semicolon here!

If/Else

- You can tack on an “else” part
  - What to do if condition is false

```
if(x == y)
  z = 3'b000;
else
  z = 3'b111;   // If x != y
```

Example: If/Else

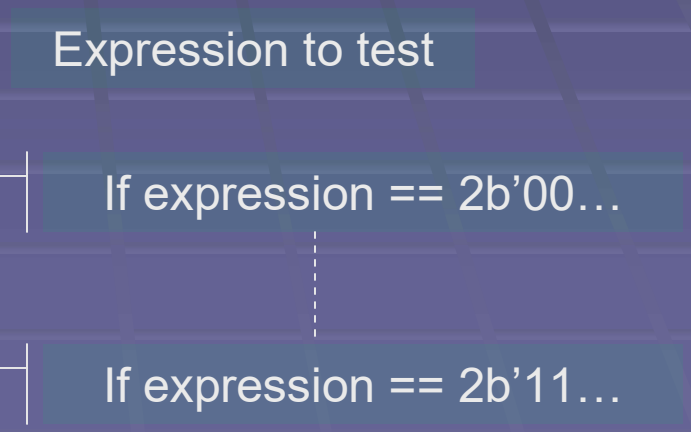
- May be used in combinatorial logic

```
module Adder (
                    input logic         reset,
                    input logic [31:0] aInput,
                    input logic [31:0] bInput,
                    output logic[31:0] sum
                )

  always_comb begin
    if(reset)
      sum = 32'b0;              // if resetting, result is 0
    else
      sum = aInput + bInput; // else, result is add
  end

endmodule
```

Case Statement

- A multi-way conditional expression

```
logic [1:0] selector;
logic in0, in1, in2, in3, out;

  // A 4-way mux
case(selector)
  2'b00: out = in0;
  2'b01: out = in1;
  2'b10: out = in2;
  2'b11: out = in3;
endcase;
```



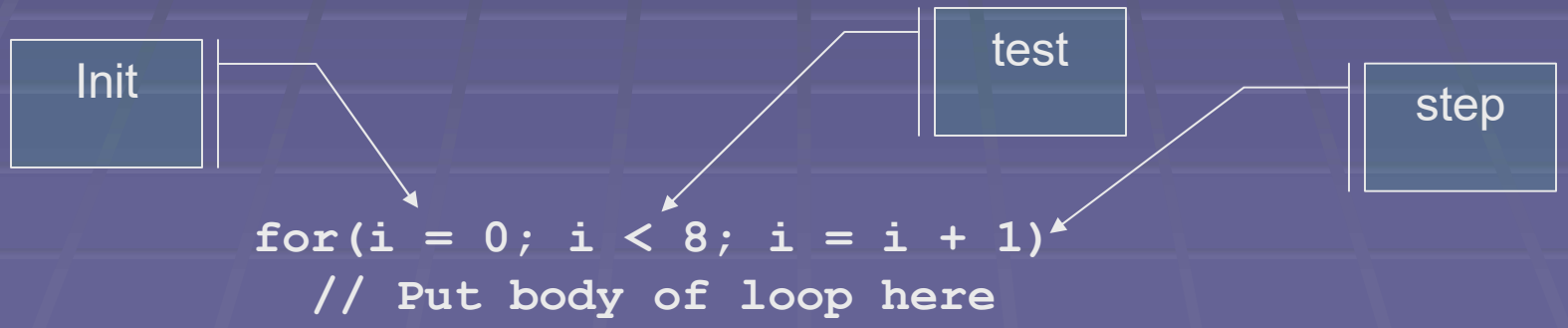
Case Statement

- "default": if nothing else matches, do this

```
logic [6:0] segments;
                                // 7-segment decoder
 case (value)
        0: segments = 7'b0111111;
        1: segments = 7'b0000110;
        2: segments = 7'b1011011;
        3: segments = 7'b1001111;
        4: segments = 7'b1100110;
        5: segments = 7'b1101101;
        6: segments = 7'b1111101;
        7: segments = 7'b0000111;
        8: segments = 7'b1111111;
        9: segments = 7'b1101111;
      default: segments = 7'b0000000;
endcase
```

For Loop

- Repeats code segment multiple times
- Syntax: *for(<init> ; <test> ; <step>)*
- *Where*
  - <init> : initial value of counter variable
  - <test> : end repetition when this is true
  - <step> : action to prep variable for next iteration



- This loop runs for 8 iterations
  - I = 0, 1, …, 6, 7

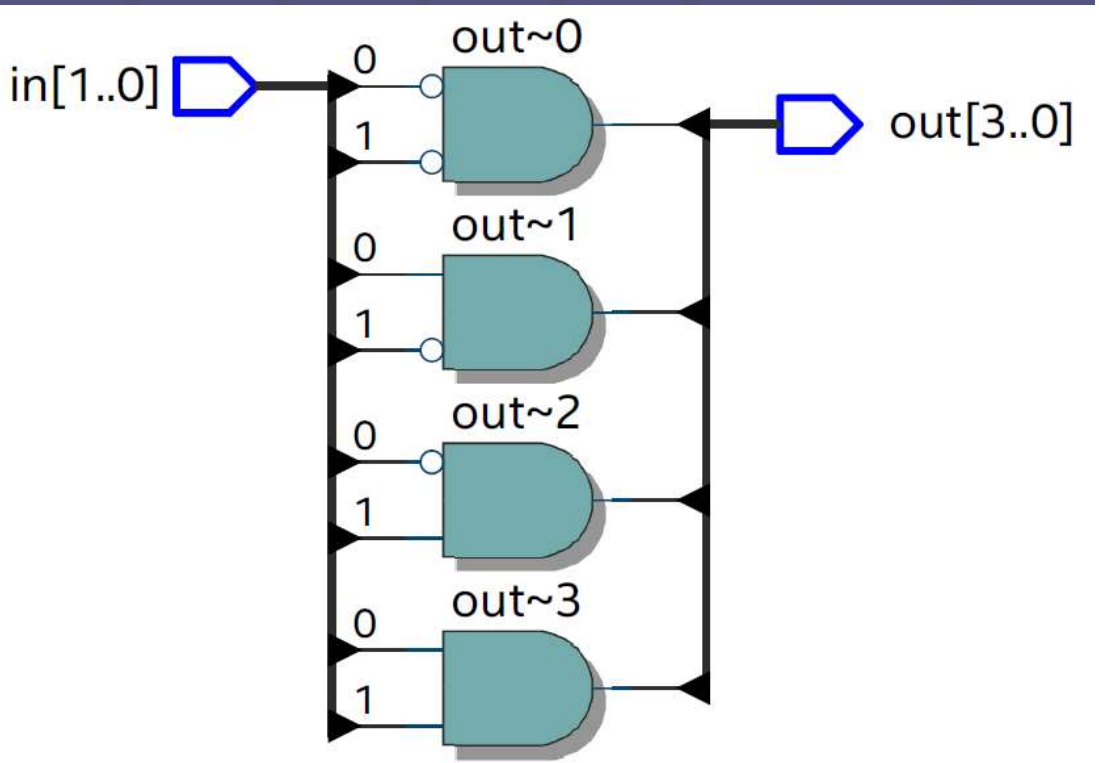
Example For Loop

```
parameter InBits = 2;                    // Named constant
parameter OutBits = 1 << InBits;

module top
  (
   input logic [InBits-1 : 0]   in,
   output logic [OutBits-1 : 0] out
   );

   int i; // Value, indefinite width

   // 2:4 decoder
   always_comb
     for(i = 0; i < OutBits; i = i + 1)
       out[i] = (in == i);

endmodule
```

What will this produce?

- Synthesized into FPGA
  - Tools figure out this is just a handful of gates

- No actual looping
- Eliminated “i”
- Easy to extend
  - Increase InBits to handle wider input
  - Re-use is important



Variations you might see

- Can declare loop counter in loop

```
for(int count = 0; count != 8; count = count + 1)
```

- Step specifier may be any assignment

```
for(int count = 7; count >= 0; count = count - 1)
```

- Common to use increment operator
  - We haven't see this until now!

```
for(int xyz; xyz < 16; xyz++ )
```