Design Using Structural and Functional Verilog

Instructor: Dr. Vinicius Prado da Fonseca (vpradodafons@online.mun.ca)

Structural vs Functional Verilog

- Structural
  - customarily, refers to describing a design using module instances
  - lower-level building blocks such as AND gates and flip-flops
- Functional or behavioural
  - refers to describing a design using always comb blocks
  - clock gates and synchronizer cells
- We will compare the many functional representations using the YOSYS tool

Modulus Detector Version 1

- Several ways to describe a combinational logic module
- Consider the following circuit that determines if a 4-bit unsigned integer has a reminder of 1 or 2 after being divided by 3.

$$v \bmod 3 = 1 \text{ or } 2$$

- The four inputs, v[3], v[2], v[1], and v[0] for a unsigned integer

$$8*v[3] + 4*v[2] + 2*v[1] + 1*v[0]$$

- always_comb
  - Sensitivity list includes all signals in that function, different from always @ (next classes)

Modulus Detector Version 1 (case)

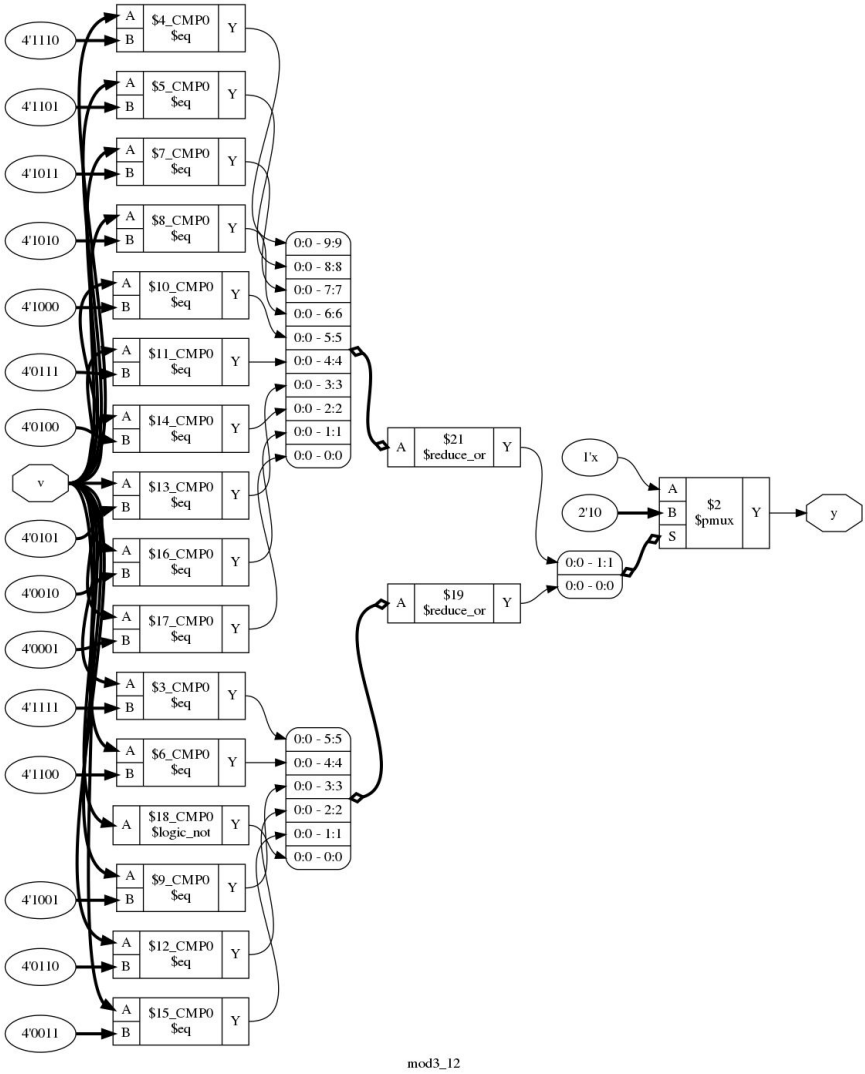
mod3_12_v1.v, tb_mod3_12.v

- Truth tables can be represented almost directly with a case statement

```
case ( v )
      4'b0000: y = 0;
      4'b0001: y = 1;
      4'b0010: y = 1;
            ...
      4'b1110: y = 1;
      4'b1111: y = 0;
endcase
```

v3	v2	v1	v0	y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

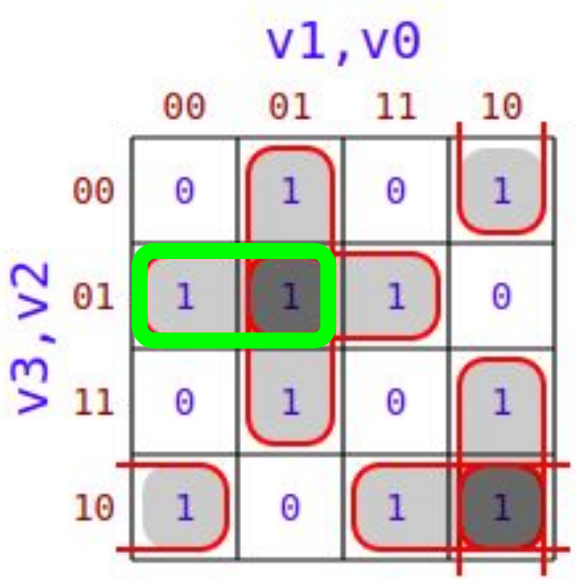
Modulus Detector Version 1



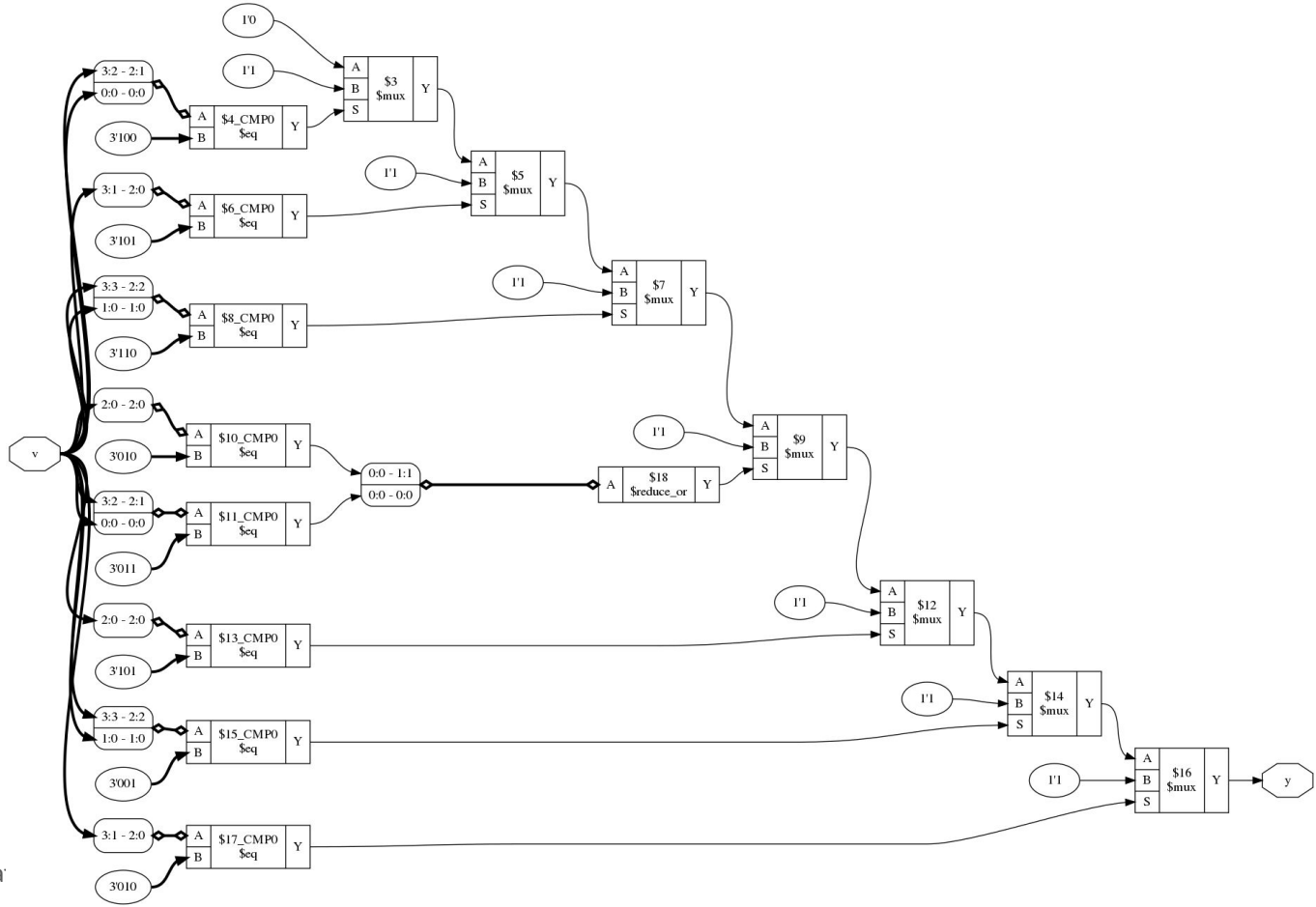
Modulus Detector Version 2 (casez)

The K-map for v mod 3 = 1 or 2 is:

```
casez ( v )  // don't cares
      4'b010?: y = 1;   // (~v3v2~v1) prime implicants
      4'b0?01: y = 1;
      4'b?101: y = 1;
      4'b01?1: y = 1;
      4'b?010: y = 1;
      4'b1?10: y = 1;
      4'b101?: y = 1;
      4'b10?0: y = 1;
      default: y = 0;
endcase
```



Modulus Detector Version 2



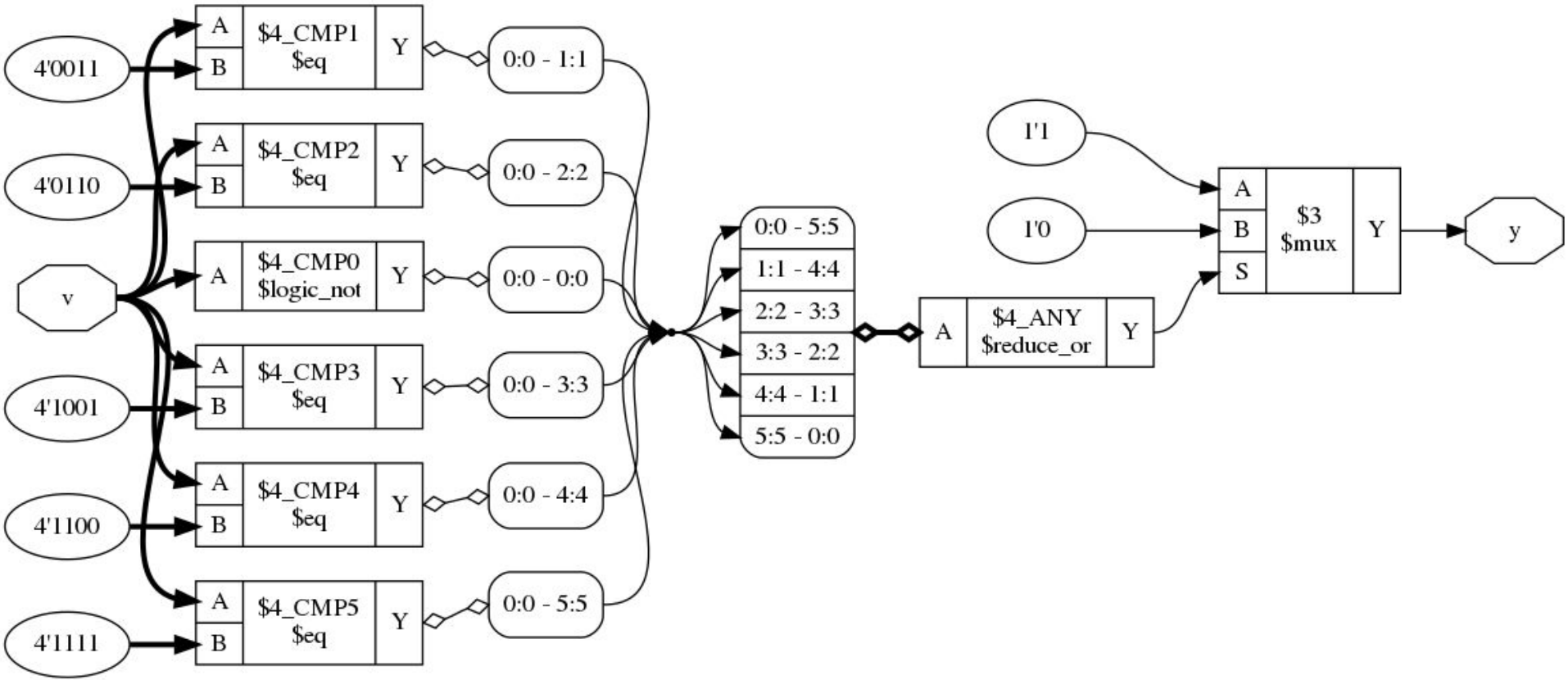
Modulus Detector Version 3 (case, decimals)

- Decimal integers, that represent the 4 bit value, can be used directly

```
case ( v )
     0,3,6,9,12,15: y = 0;
     default: y = 1;
endcase
```

v3	v2	v1	v0	y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Modulus Detector Version 3

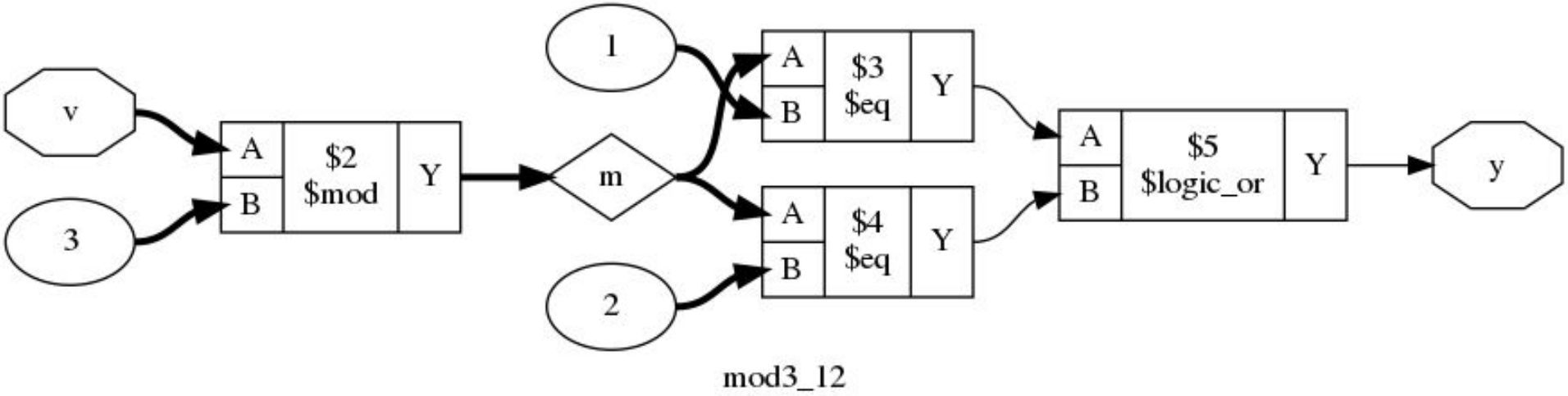


Modulus Detector Version 4 (mod operator)

- Verilog also allows mathematical expressions to be used

```
integer m;
always_comb begin
     m = v % 3;
     y = m == 1 || m == 2;
end
```

Modulus Detector Version 4



Modulus Detector Version 5 (mod operator 2)

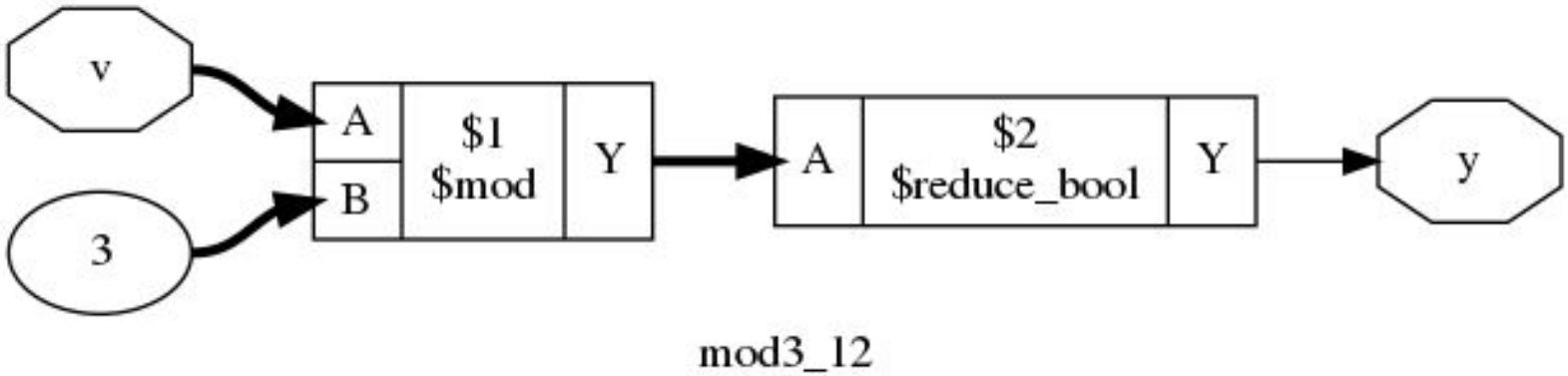
- Really compact version

```
module mod3_12( output logic y, input logic [3:0] v );

     assign y = v % 3 != 0;

endmodule
```

Modulus Detector Version 5

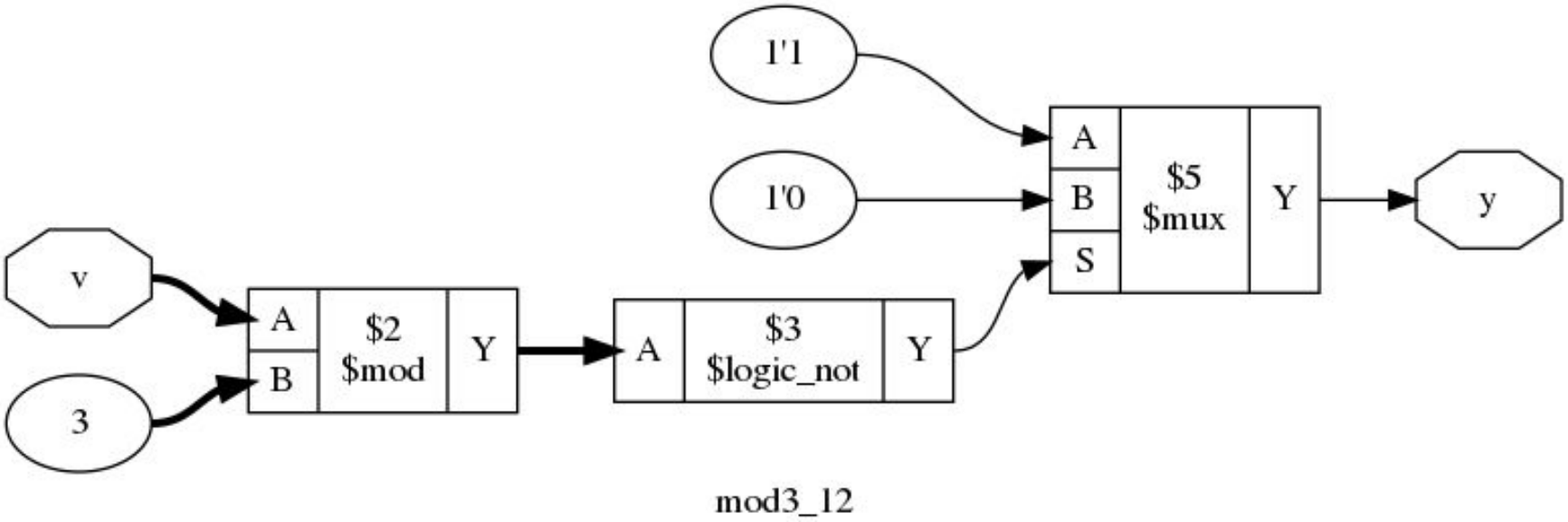


Modulus Detector Version 6 (if else)

- Verilog also provides conditional if statements

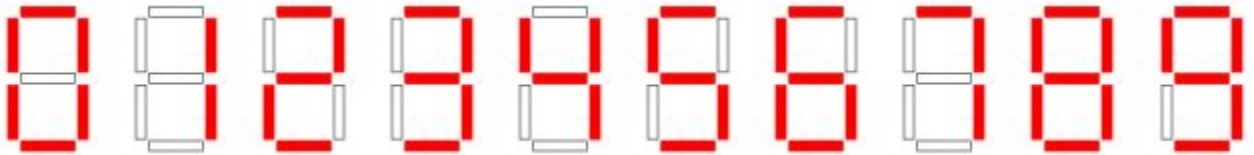
```
always_comb begin
     if ( v % 3 == 0 ) begin
          y = 0;
     end
     else begin
          y = 1;
     end
end
```

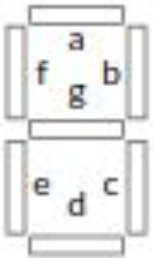
Modulus Detector Version 6



Seven Segment Display (case, all segments)

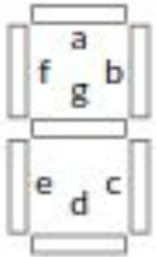
- The digits 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9 can be displayed with seven segments
- It also can be implemented using case.





Seven Segment Display

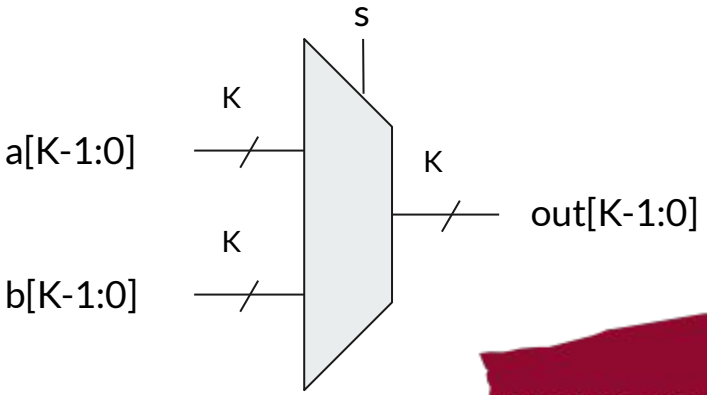
```
// {g,f,e,d,c,b,a} == seg
always_comb begin
    case( digit )
        0: seg = 7'b0111111;
        1: seg = 7'b0000110;
        // digits 2,3,4,5,6,7,8,9 are missing
        default: seg = 7'bxxxxxxx;
    endcase
end
```



Parameterized Modules

- Verilog provides parameters to customize a module
- A K-bit 2-to-1 multiplexor can be implemented with:
- out, b and a are buses of K bits
- s selects all bits from a or b at the same time.

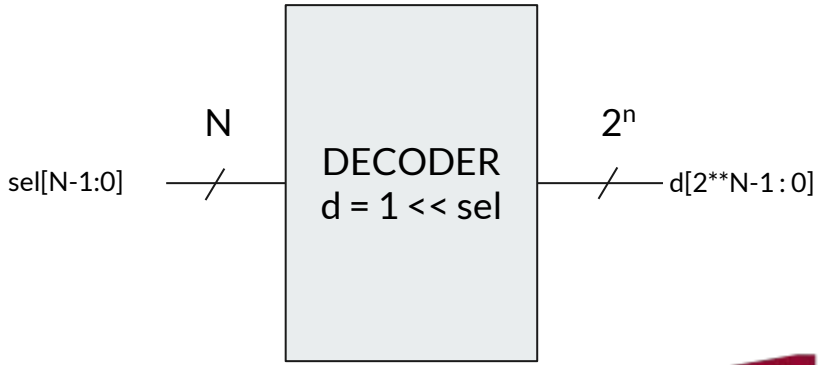
```
module mux #(parameter K=4) (
                    output logic [K-1:0] out,
                    input logic [K-1:0] a, b,
                    input logic s);
             assign out = s ? b : a;
endmodule
```



Parameterized Modules

- N to 2^N decoder
- N bits selector
- 2N bits output, only one bit activated per selector.

```
module decoder #(parameter N=2) (
                  output logic [2**N-1 : 0] d,
                  input logic [N-1:0] sel );
      assign d = 1 << sel;
endmodule
```



Questions?

Next: 15 - Modeling Synchronous Systems With Verilog