Today: More Verilog and Sequential Logic

- Finite State Machines and Verilog
- Reasoning about Moore and Mealy machines
  - "highlight-the-arrows" method
- Example: Traffic Light Controller in Verilog

Verilog Structural View of a FSM

- General view of a finite state machine in verilog

```verilog
module FSM (CLK, in, out);
    input CLK;
    input in;
    output out;
    reg out;
    // state variable
    reg [1:0] state;
    // local variable
    reg [1:0] next_state;

    always @(posedge CLK) // registers
        state = next_state;

    always @(state or in) // Compute next-state and output logic whenever state or inputs change.
        // (i.e. put equations here for next_state[1:0])
        // Make sure every local variable has an assignment in this block!
    endmodule
```
### Moore Verilog FSM

#### Reduce 1's example

- `define zero 2'b00
- `define one1 2'b01
- `define two1s 2'b10

module reduce (CLK, reset, in, out);
input CLK, reset, in;
output out;
reg out;
reg [1:0] state; // state variables
reg [1:0] next_state;

always @(posedge CLK)
if (reset) state = `zero;
else next_state = state;

always @(in or state)
case (state)
  `zero: // last input was a zero
    begin
      if (in) next_state = `one1;
      else next_state = `zero;
    end
  `one1: // we've seen one 1
    begin
      if (in) next_state = `two1s;
      else next_state = `zero;
    end
  `two1s: // we've seen at least 2 ones
    begin
      if (in) next_state = `two1s;
      else next_state = `zero;
    end
endcase

endmodule
### Mealy Verilog FSM

```verilog
module reduce (CLK, reset, in, out);
  input CLK, reset, in;
  output out;
  reg out;
  reg state; // state variables
  reg next_state;
  always @(posedge CLK)
    if (reset) state = `zero;
    else state = next_state;
  always @(in or state)
    case (state)
      `zero: // last input was a zero
        begin
          out = 0;
          if (in) next_state = `one;
          else next_state = `zero;
        end
      `one: // we've seen one 1
        if (in) begin
          next_state = `one; out = 1;
          end else begin
            next_state = `zero; out = 0;
          end
      endcase
  endmodule
```

### Synchronous Mealy FSM

```verilog
module reduce (clk, reset, in, out);
  input clk, reset, in;
  output out;
  reg out;
  reg state; // state variables
  always @(posedge clk)
    if (reset) state = `zero;
    else
      case (state)
        `zero: // last input was a zero
          begin
            out = 0;
            if (in) state = `one;
            else state = `zero;
          end
        `one: // we've seen one 1
          if (in) begin
            state = `one; out = 1;
            end else begin
              state = `zero; out = 0;
            end
      endcase
  endmodule
```
Example: Traffic Light Controller

Specification of inputs, outputs, and state elements

module FSM(HR, HY, HG, FR, FY, FG, ST, TS, TL, C, reset, Clk);
output HR;
output HY;
output HG;
output FR;
output FY;
output FG;
output ST;
input TS;
input TL;
input C;
input reset;
input Clk;
reg [6:1] state;
reg ST;

`define highwaygreen   6'b001100
`define highwayyellow  6'b010100
`define farmroadgreen  6'b100001
`define farmroadyellow 6'b100010
assign HR = state[6];
assign HY = state[5];
assign HG = state[4];
assign FR = state[3];
assign FY = state[2];
assign FG = state[1];
endmodule

Example: Traffic Light Controller (cont’d)

initial begin state = `highwaygreen; ST = 0; end
always @(posedge Clk)
begin
if (reset)
begine state = `highwaygreen; ST = 1; end
else
begin
ST = 0;
case (state)
`highwaygreen:
if (TL & C) begin state = `highwayyellow; ST = 1; end
`highwayyellow:
if (TS) begin state = `farmroadgreen; ST = 1; end
`farmroadgreen:
if (TL | !C) begin state = `farmroadyellow; ST = 1; end
`farmroadyellow:
if (TS) begin state = `highwaygreen; ST = 1; end
endcase
dend
endmodule
Timer for Traffic Light Controller

Another FSM

module Timer(TS, TL, ST, Clk);
output TS;
output TL;
input ST;
input Clk;
reg[7:0] value;

assign TS = (value >= 4); // 5 cycles after reset
assign TL = (value >= 14); // 15 cycles after reset

always @(posedge ST) value = 0; // async reset
always @(posedge Clk) value = value + 1;
endmodule

Complete Traffic Light Controller

Tying it all together (FSM + timer)

Note, of course, that this structural Verilog does not work in DesignWorks. Use a schematic instead.

module main(HR, HY, HG, FR, FY, FG, reset, C, Clk);
output HR, HY, HG, FR, FY, FG;
input reset, C, Clk;

Timer part1(TS, TL, ST, Clk);
FSM part2(HR, HY, HG, FR, FY, FG, ST, TS, TL, C, reset, Clk);
endmodule