Synthesizable Verilog Structures – A Brief Review

Dr. Paul D. Franzon
ECE
NC State University
paulf@ncsu.edu
Within A Module

module DataPath (input clock;  input reset;
                   input [3:0] data; ...
                   output reg [7:0] ExampleOut);

// local variables
wire foo;

always@(posedge clock)
  begin
    if (!reset) state <= ...
    else state <=
      case (sel1)
        00 : ExampleOut <= ...
        01 : ExampleOut <= ...
      endcase
  end
always@(*)
begin
    casex (sel2)
    0000 : {A,B} :
        default : out = x;
    endcase
end

assign foo = | {out1, out2};

endmodule

Other Important Rules
Each variables can be assigned in only ONE procedural block or continuous assignment statement

Make sure combinational logic does not feedback on itself.

e.g.
assign foo = bar ^ out2;
assign bar = foo | out1;

Ensure every logic path has at least one flip-flop between primary input and output
Putting together your logic

module top (input clock, reset;
    ...
    output wire [7:0] ExampleOut);

    //local variables
    wire control1;

    DataPath u1 (.clock(clock), .reset(reset), ...
        .ExampleOut (ExampleOut);
    )

    Controller u2 (.clock(clock), .reset(reset), ...
        .C1(control1);
    )

endmodule

No logic above leaf cell modules
module test_fixture;

reg clock, reset;
wire [7:0] ExampleOut;

initial
  begin
    clock = 0;

    #100 if (ExampleOut == 8'hFF) $display("error\n");
  end
  always #10 clock =~ clock;

  top ul (.clock(clock), ..., .ExampleOut(ExampleOut));
sram m1 (.DataIn(ExampleOut), ...);
endmodule
Synthesizing The Design

Simplest Approach:

```verilog
read_verilog ... # Make sure to read all the design files
current_design top # Important

# set constraints

# with current_design= top, compile will characterize-compile modules one
# at a time

compile
```