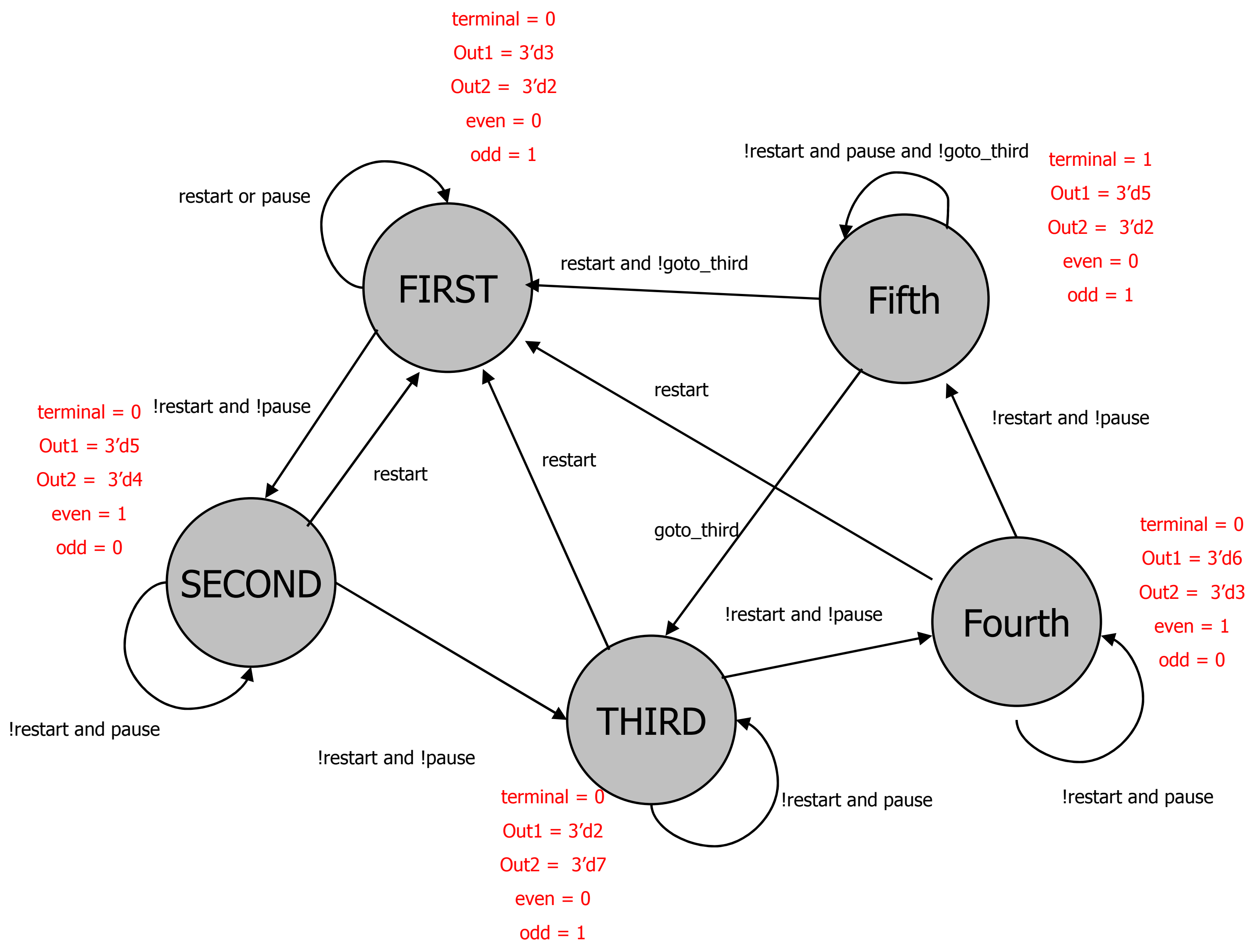
Homework #2 Question 1 (30%)

```
module fsm (state, odd, even, terminal, pause, restart, clk, rst);

input pause, restart, clk, rst;
output [1:0] state;
output odd, even, terminal;

reg [1:0] state;
reg odd, even, terminal;

parameter [1:0] FIRST  = 2'b11;
parameter [1:0] SECOND = 2'b01;
parameter [1:0] THIRD  = 2'b10;

always_ff @(posedge clk or posedge rst) // sequential
begin
  if (rst) state <= FIRST;
  else
  begin
    case(state)
      FIRST:   if (restart | pause) state <= FIRST;
               else state <= SECOND;
      SECOND:  if (restart) state <= FIRST;
               else if (pause) state <= SECOND;
               else state <= THIRD;
      THIRD:   if (!restart & pause) state <= THIRD;
               else state <= FIRST;
      default: state <= FIRST;
    endcase
  end
end

// output logic described using procedural assignment
always_comb
begin
  odd = (state == FIRST) | (state == THIRD);
  even = (state == SECOND);
  terminal = (state == THIRD) & (restart | !pause);
end

endmodule
```

1. Compile this state machine, that we analyzed in class
2. Hand in an annotated simulation of this state machine (use Quartus simulator or Modelsim)
3. Draw by hand and hand in the schematic of the logic defined in the Verilog code

Question 2 (30%)

1. Look at the state diagram on the next slide. Note that the outputs are in red while the inputs are in black. There is one new input (goto_third) and two new 3-bit outputs (Out1 and Out2).
2. Write and hand in Verilog/VHDL code to implement the state machine in a way that **does not have glitches**.
3. Compile in Quartus and hand in an annotated simulation of this state machine (use Quartus simulator or Modelsim)
4. Hand in a hand-drawn schematic of the logic that the Verilog/VHDL code is describing.


terminal = 0
Out1 = 3’d3
Out2 = 3’d2
even = 0
odd = 1
restart or pause
FIRST
restart and !goto_third
!restart and pause and !goto_third
terminal = 1
Out1 = 3’d5
Out2 = 3’d2
even = 0
odd = 1
Fifth
restart
!restart and !pause
terminal = 0
Out1 = 3’d5
Out2 = 3’d4
even = 1
odd = 0
!restart and !pause
restart
restart
goto_third
SECOND
!restart and pause
!restart and !pause
THIRD
!restart and !pause
Fourth
terminal = 0
Out1 = 3’d6
Out2 = 3’d3
even = 1
odd = 0
!restart and pause
!restart and pause
terminal = 0
Out1 = 3’d2
Out2 = 3’d7
even = 0
odd = 1

Question 3 (40%)

- Hand in a hand-drawn schematic of the FSM “viterbi_ctrl” that we saw in class.

Pay attention to the following

- For clarity hand drawn schematics should use higher level primitives such as muxes, adders, and multi-bit registers. Do not decompose these into gates/flip-flops.
- Using an RTL viewer and copying the results by hand will be considered cheating.
- Draw as neatly as possible.
- In the annotated simulations, make sure that your annotations show that you understand what is going on in the simulation.