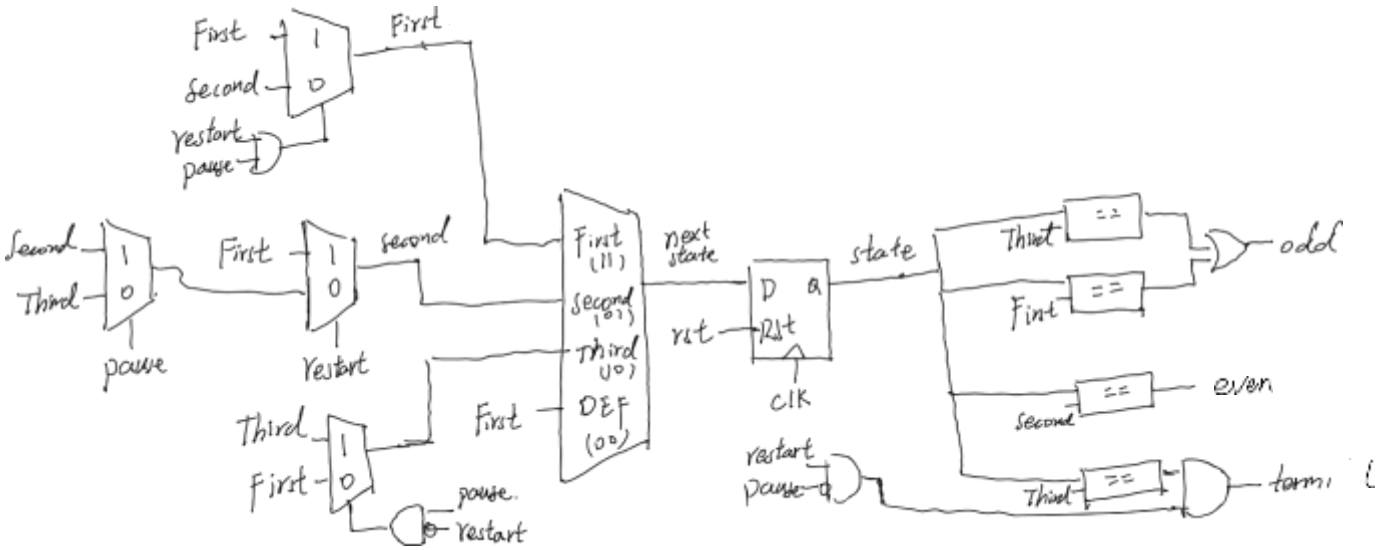
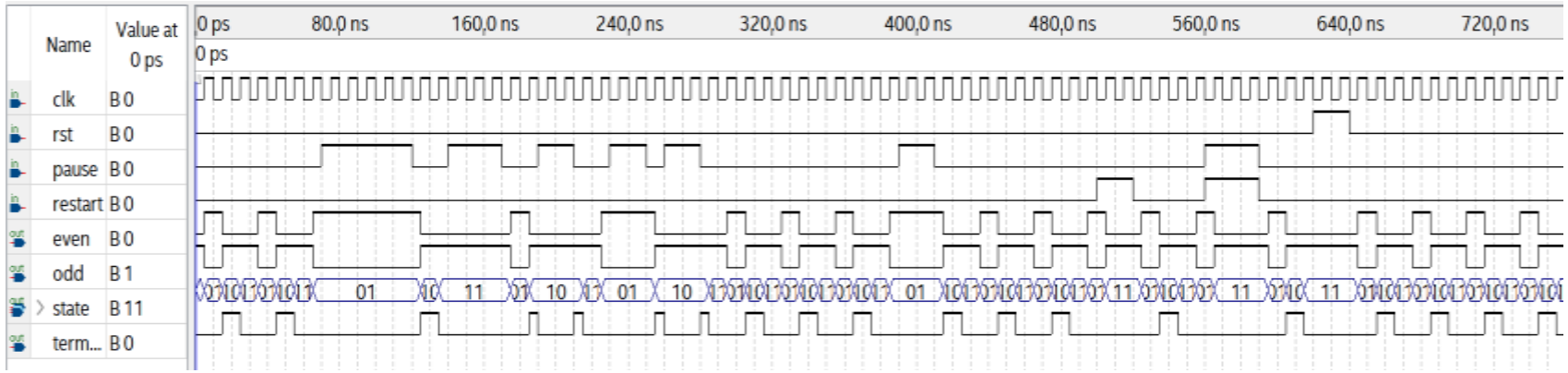
Homework #2
Question 1 (30%)

```
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

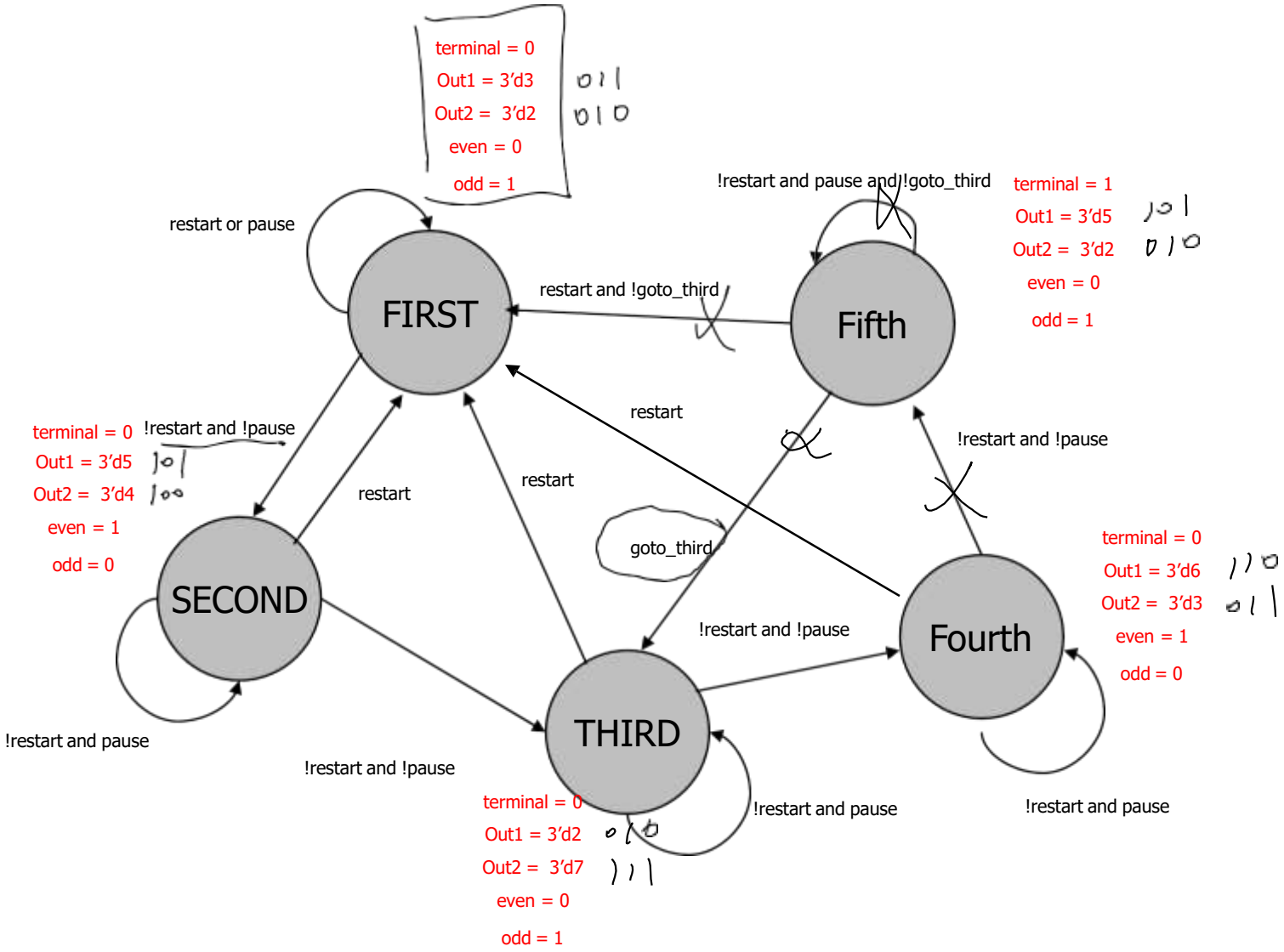

First
Second
restart
pause
First
Second
Third
pause
First
Restart
Second
Third
First
pause
restart
First (11)
Second (01)
Third (10)
DEF (00)
next state
rst
D
Rst
CLK
state
Third
First
Second
Third
restart
pause
odd
even
term



In the 1 pause, the state is stay in state of second, the out put is 1_0_0
In the 2 pause, the state is stay in state of first, the out put is 0_1_0
In the 3 pause, the state is stay in state of third, the out put is 0_1_0
In the restart, the state transfer to the first

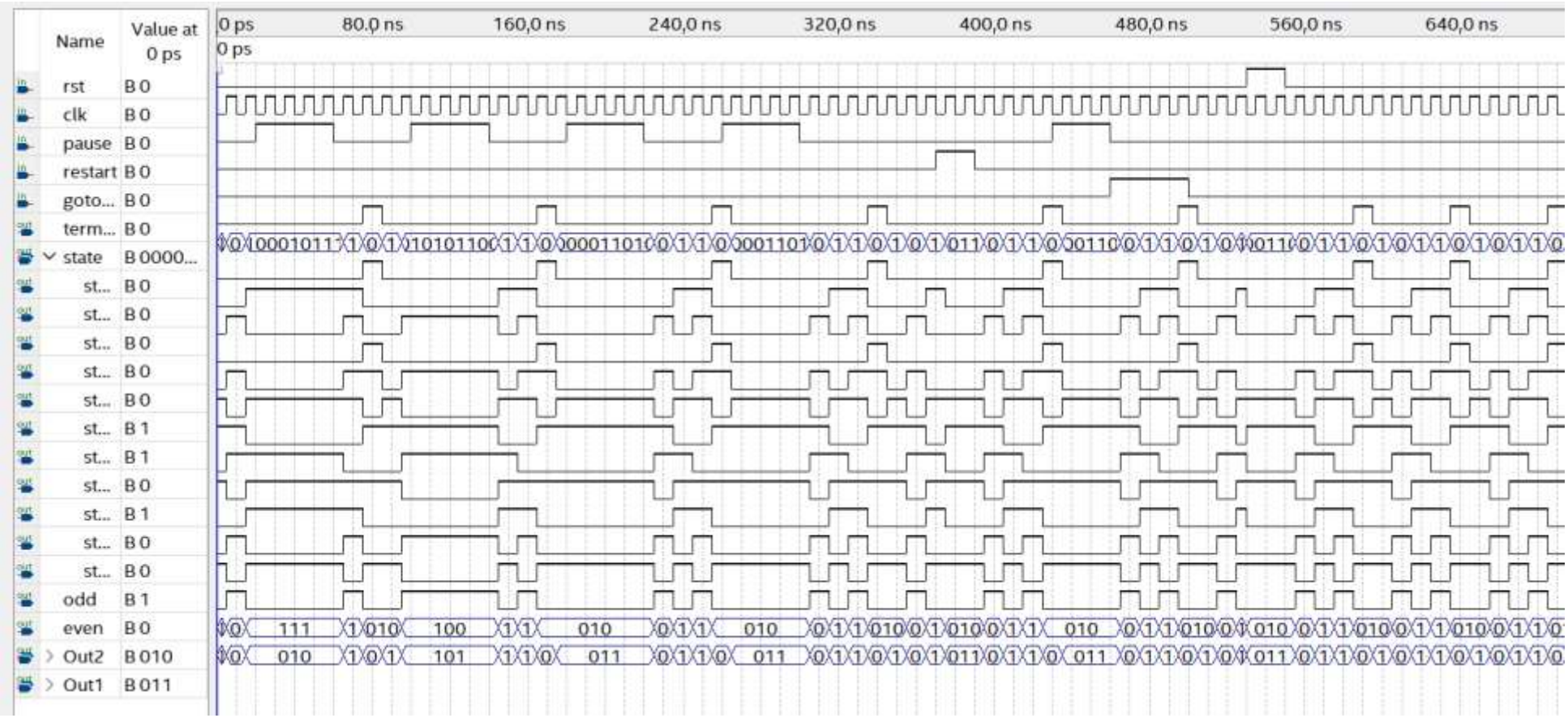
Question 2 (30%)

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.

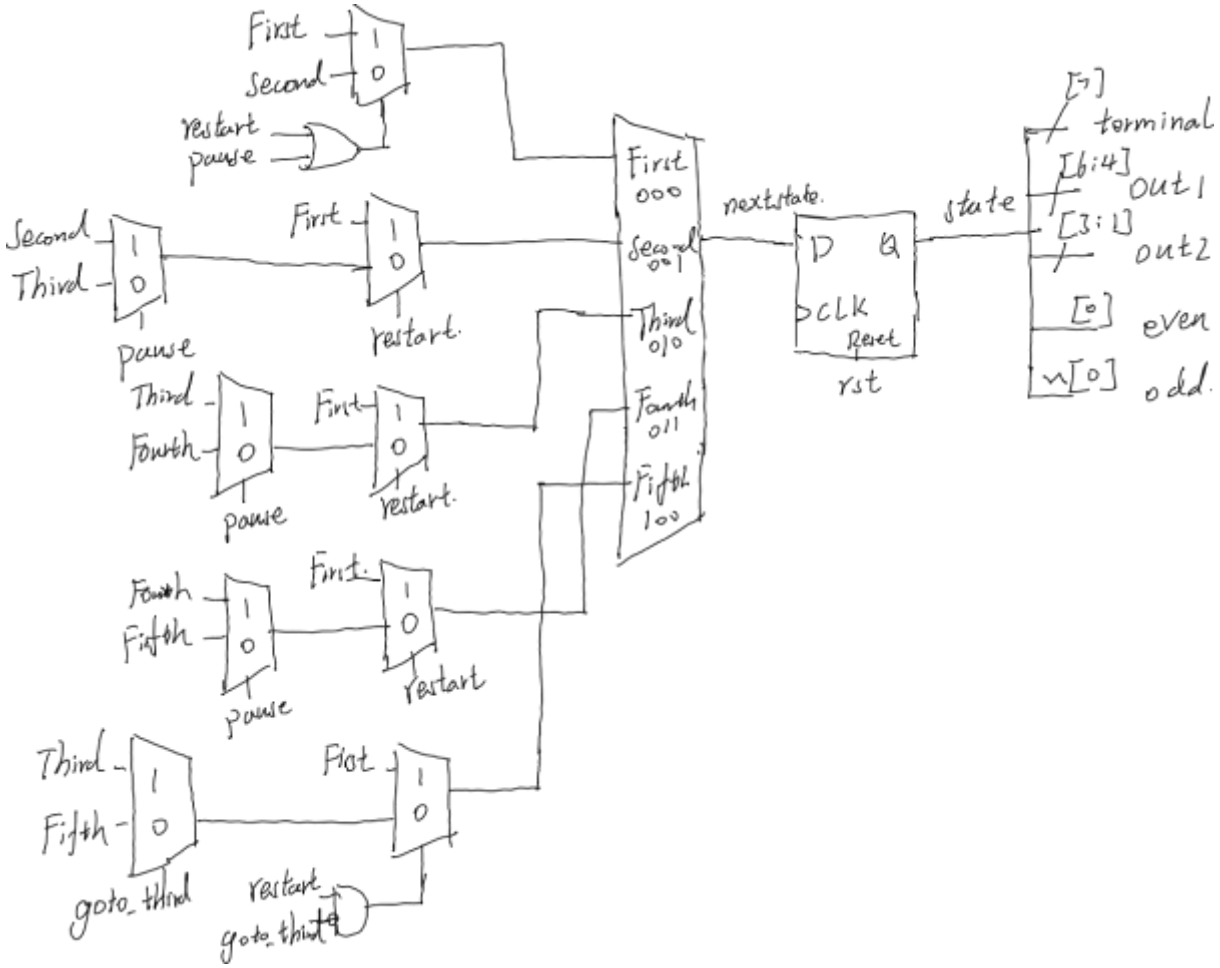

terminal = 0
Out1 = 3'd3
Out2 = 3'd2
even = 0
odd = 1
restart or pause
FIRST
!restart and pause and!goto_third
terminal = 1
Out1 = 3'd5
Out2 = 3'd2
even = 0
odd = 1
Fifth
restart and !goto_third
terminal = 0
Out1 = 3'd5
Out2 = 3'd4
even = 1
odd = 0
!restart and !pause
restart
restart
restart
!restart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

```
module qs2 (state, odd, even, terminal, pause, restart, clk, rst, goto_third, Out1, Out2);
                input logic pause, restart, clk, rst, goto_third;
                output logic [10:0] state;
                output logic odd, even, terminal;
                output logic [2:0] Out1;
                output logic [2:0] Out2;
                                // 1098 7 654 321 0
                parameter [10:0] FIRST  =       12'b000_0_011_010_0;
                parameter [10:0] SECOND =  12'b001_0_101_100_1;
                parameter [10:0] THIRD  =       12'b010_0_010_111_0;
                parameter [10:0] FOURTH =       12'b011_0_110_011_1;
                parameter [10:0] FIFTH  =       12'b100_1_101_010_0;

                assign even = state[0];
                assign odd  = ~state[0];
                assign Out2 = state[3:1];
                assign Out1 = state[6:4];
                assign terminal = state[7];

                always_ff @(posedge clk or posedge rst) 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) state <= FIRST;

                                                                                else if (pause) state <= THIRD;
                                                                                else state <= FOURTH;
                                                                FOURTH: if (restart) state <= FIRST;

                                                                                else if (pause) state <= FOURTH;
                                                                                else state <= FIFTH;
                                                                FIFTH: if (restart|!goto_third) state <= FIRST;

                                                                                else if (goto_third) state <= THIRD;
                                                                                else state <= FIFTH;

                                                                default: state <= FIRST;
                                                endcase
                                end
                end
endmodule
```

```
1  //Question 2-code, Ruiqi Zang 24075277
2  module qs2 (state, odd, even, terminal, pause, restart, clk, rst, goto_third, Out1, Out2);
3     input logic pause, restart, clk, rst, goto_third;
4     output logic [10:0] state;
5     output logic odd, even, terminal;
6     output logic [2:0] Out1, Out2;
7                                 // 1098 7 654 321 0
8     parameter [10:0] FIRST  =  12'b000_0_011_010_0;
9     parameter [10:0] SECOND =  12'b001_0_101_100_1;
10    parameter [10:0] THIRD  =  12'b010_0_010_111_0;
11    parameter [10:0] FOURTH =  12'b011_0_110_011_1;
12    parameter [10:0] FIFTH  =  12'b100_1_101_010_0;
13
14    assign even = state[0];
15    assign odd  = ~state[0];
16    assign Out2 = state[3:1];
17    assign Out1 = state[6:4];
18    assign terminal = state[7];
19
20    always_ff @(posedge clk or posedge rst) begin
21       if (rst) state <= FIRST;
22       else
23       begin
24          case(state)
25             FIRST: if (restart | pause) state <= FIRST;
26                    else state <= SECOND;
27             SECOND: if (restart) state <= FIRST;
28                    else if (pause) state <= SECOND;
29                    else state <= THIRD;
30             THIRD: if (restart) state <= FIRST;
31                    else if (pause) state <= THIRD;
32                    else state <= FOURTH;
33             FOURTH: if (restart) state <= FIRST;
34                    else if (pause) state <= FOURTH;
35                    else state <= FIFTH;
36             FIFTH: if (restart|!goto_third) state <= FIRST;
37                    else if (goto_third) state <= THIRD;
38                    else state <= FIFTH;
39             default: state <= FIRST;
40          endcase
41       end
42    end
43 endmodule
44
45
```

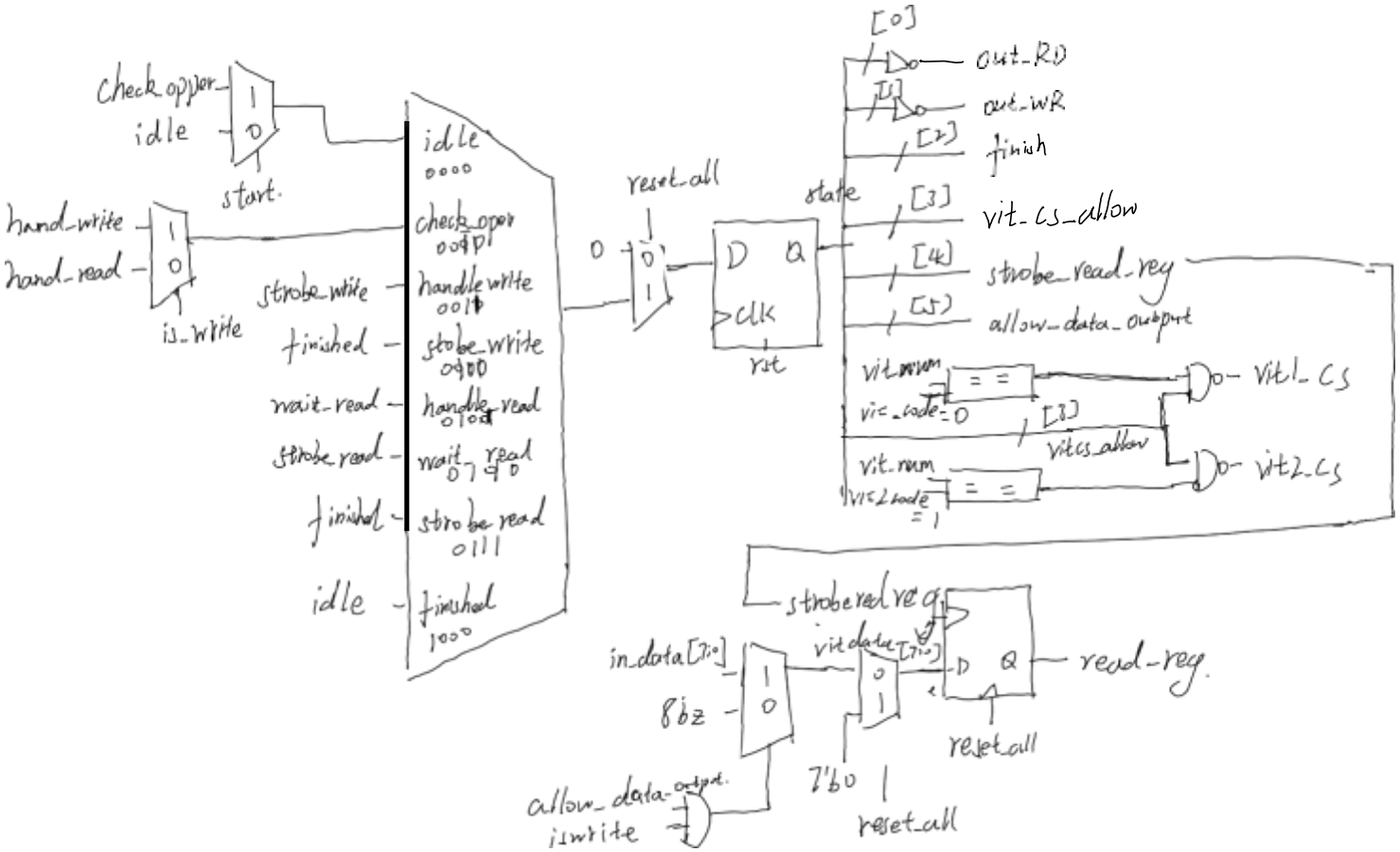


In the 1pause, the state is stay in state of Third, the out put is 0_010_111_0_1
In the 2 pause, the state is stay in state of Second, the out put is 0_101_100_1_0
In the 3 pause, the state is stay in state of First, the out put is 0_011_010_0_1
In the goto_third, the state transfer to state Fifth.
In the restart, the state transfer to state First.
In the restart, the state transfer to state First.
In th reset, the state transfer to state First.


First
Second
restart
pause
Second
Third
pause
First
restart
Third
Fourth
pause
First
restart
Fourth
Fifth
pause
First
restart
Third
Fifth
goto_third
Fiot
restart
goto_third
First
000
Second
001
Third
010
Fourth
011
Fifth
100
nextstate
D
Q
CLK
Reset
rst
state
[7]
terminal
[6:4]
out1
[3:1]
out2
[0]
even
~[0]
odd

Question 3 (40%)

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



out_addr----------in_addr;

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.