國立彰化師範大學 資訊工程學系 硬體描述語言 期末考

2025/01/09

1. Answer the following questions. **[40%:** (4*10)%]

(a) The use of a "reg" variable in Verilog does not imply that the associated behavior is sequential, why?

(b) What is the difference between a D-type "latch" and a D-type "flip-flop"?

(c) What is the difference between a signal that is the output of a combinational logic and a signal that is a registered output?

(d) What is the difference between a Moore machine and a Mealy machine?

(e) For the one-stage sequential circuit in Fig. 1, what is the constraint of the clock period when data is to be sent correctly from the left flip-flop to the right one?

(f) For a D-FF, explain the meanings its "setup time" and "hold time".

(g) For a Verilog-based design, what is the difference between "function" and "task"?

(h) For a combinational sorting circuit, how to redesign it to be an 8-stage pipeline?

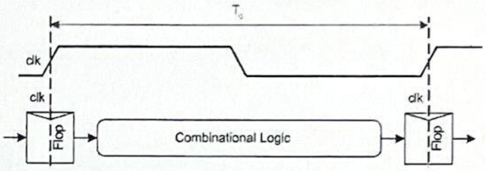


Fig. 1

(i) Explain the meaning of a "ripple counter".

(j) For an incompletely specified "case" statement, a "default" statement, e.g., "**default: alu_reg=4'b0;**" or "**default: alu_reg=4'bx;**", can both be used to make it completely specified, what is the difference between these two statements after the circuit has been synthesized?

2. For the circuit in Fig. 2, use "if…else" statement to complete the Verilog code in Fig. 3. **[10%]**

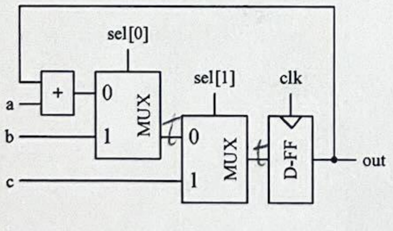
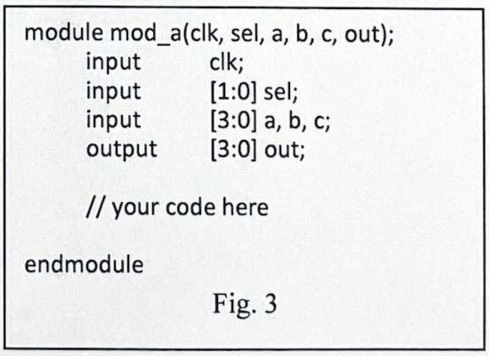


Fig. 2



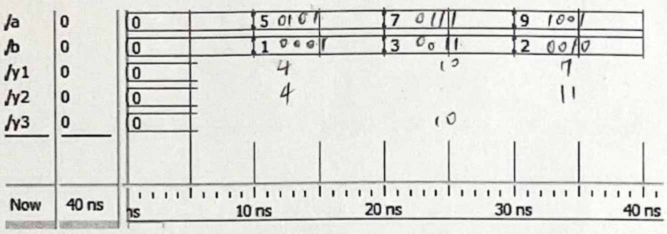
```
module mod_a(clk, sel, a, b, c, out);
        input          clk;
        input          [1:0] sel;
        input          [3:0] a, b, c;
        output         [3:0] out;

        // your code here

endmodule
```

Fig. 3

if (sel[0]==0){
out+a
else b

@always (

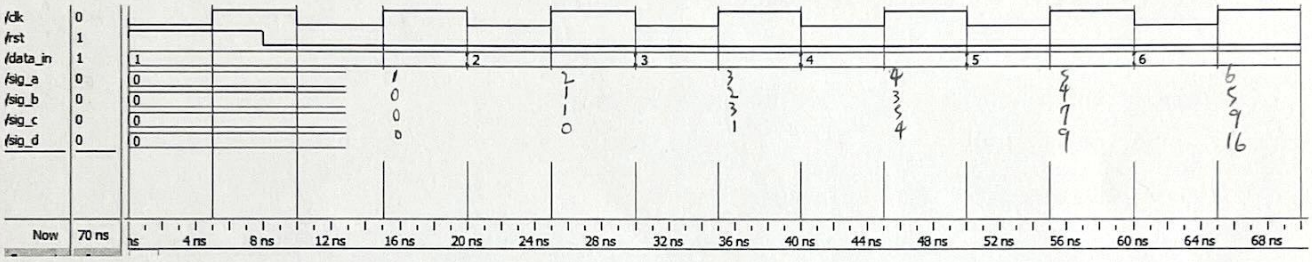
3. For the Verilog description in Fig. 4, sketch the simulation waveforms of signals y1, y2, and y3 (in decimal). **[10%]**



```
module mod_b (a, b, y1, y2, y3);
input        [3:0] a, b;
output reg       [7:0] y1, y2, y3;

always @(a or b)
      y1=(a[1]==1) ? (a+b):(a-b);

always @(a[0] or b[0])
      y2=(b[1]==1) ? (a+b):(a-b);

always @(b[2:1])
      y3=(b[0]==1) ? (a+b):(a-b);
endmodule
```

Fig. 4

4. For the Verilog description in Fig. 5. (a) Sketch the simulation waveforms of signals sig_a, sig_b, sig_c, and sig_d (in decimal). (b) Draw the schematic design for these four signals. [**20%:** (12+8)%]



```
module shifter_3 (clk, rst, data_in, sig_a, sig_b, sig_c, sig_d);
input clk, rst;
input [7:0] data_in;
output [7:0] sig_a, sig_b, sig_c, sig_d;

reg [7:0] sig_a, sig_b, sig_c, sig_d;

always @(posedge clk or posedge rst) begin
        if (rst==1) begin
          sig_a=0; sig_b=0; sig_c=0; sig_d=0;
        end
        else begin
                  sig_a <= data_in;
                  sig_b <= sig_a;
                  sig_c <= sig_b+sig_a;
                  sig_d <= sig_c+sig_d;
        end
end
endmodule
```

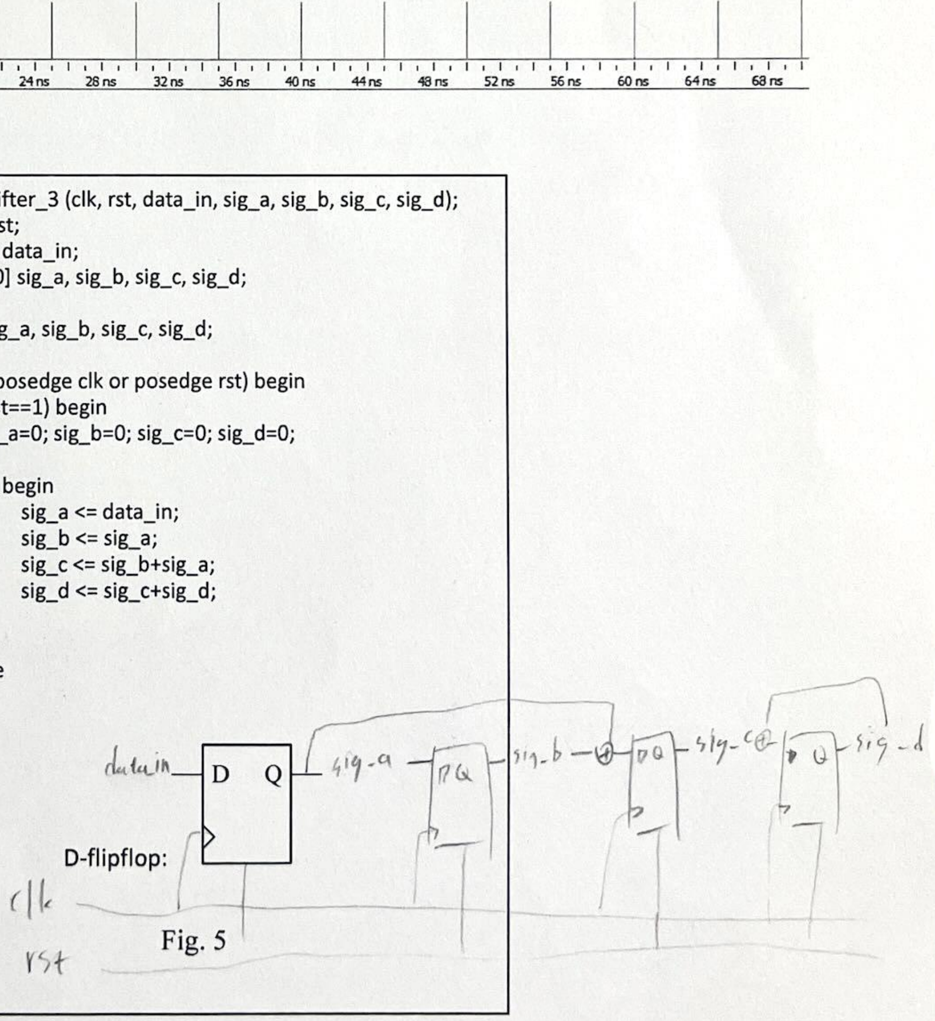
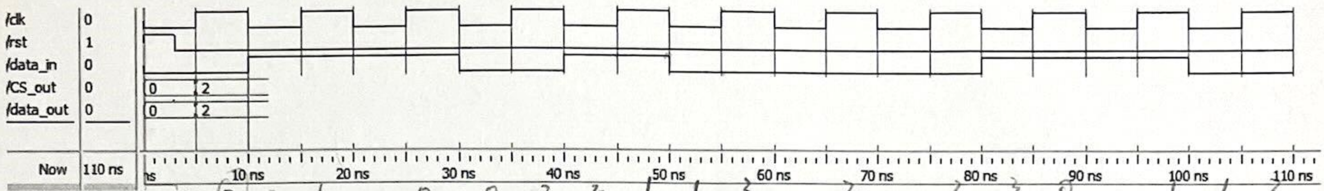


Fig. 5

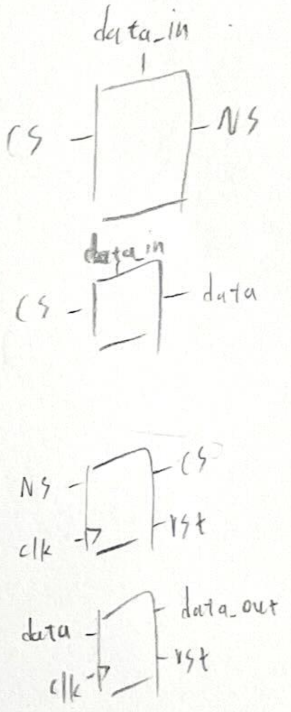
5. The code segment in Fig. 6 is the design of a Mealy machine with registered output. (a) Draw its corresponding state diagram. (b) Sketch the simulation waveforms of signals *CS_out* and *data_out*. [**20%:** (10+10)%]



CS	0	2	2	1	0	0	2	2	1	1	3	2	3	3	0	1	1	3
NS	2	3	1	0	1	2	3	1	0	3	2	3	2	0	1	0	3	2
data	2	6	5	3	1	2	6	5	3	4	8	6	8	7	1	3	4	8

```
module fsm(clk, rst, data_in, CS_out, data_out);
        input    clk, rst, data_in;
        output [1:0] CS_out;
        output [3:0] data_out;
        reg [3:0] data_out;

        reg [1:0] CS, NS; // current state, next state
        assign CS_out=CS;

        reg [3:0] data;

        // state encoding
        parameter s0=2'd0, s1=2'd1, s2=2'd2, s3=2'd3;

        always@(posedge clk or posedge rst) begin
                if (rst==1) begin
                        CS=s0; data_out=0;
                end
                else begin
                        CS=NS; data_out=data;
                end
        end

        // determine NS
        always@(CS or data_in) begin
                case(CS)
                        s0: NS=(data_in==1)?s1:s2;
                        s1: NS=(data_in==1)?s0:s3;
                        s2: NS=(data_in==1)?s1:s3;
                        s3: NS=(data_in==1)?s0:s2;
                        default: NS=s0;
                endcase
        end

        // determine data_out
        always@(CS or data_in) begin
                case(CS)
                        s0: data=(data_in==1)?1:2;
                        s1: data=(data_in==1)?3:4;
                        s2: data=(data_in==1)?5:6;
                        s3: data=(data_in==1)?7:8;
                        default: data_out=0;
                endcase
        end
endmodule
```

Fig. 6


data_in
CS
NS
data_in
CS
data
NS
clk
CS
rst
data
clk
data_out
rst