CE/CZ1005 Digital Logic
Tutorial 8

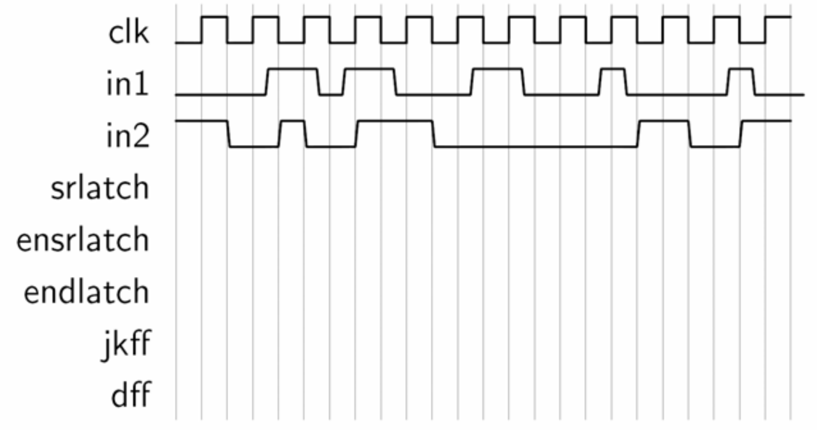
Q1. Using a behavioural description, write a Verilog module that has three 8-bit inputs, a, b, and c, and outputs the largest (max) and smallest (min) of them. Add a third output (diff) that outputs the difference between the two. You should use if statement(s). You can ignore equal inputs.

Q2. The *clk* input in the timing diagram is connected to the control/enable/clk input of the following circuits:

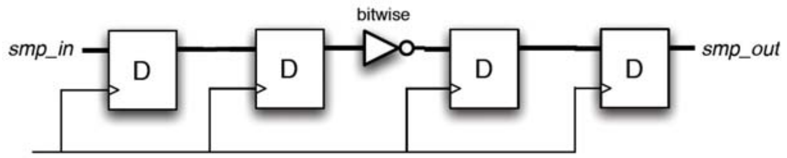
- SR-latch (no enable/control)
- Enabled SR-latch
- Enabled D-type latch
- JK flip-flop
- D flip-flop

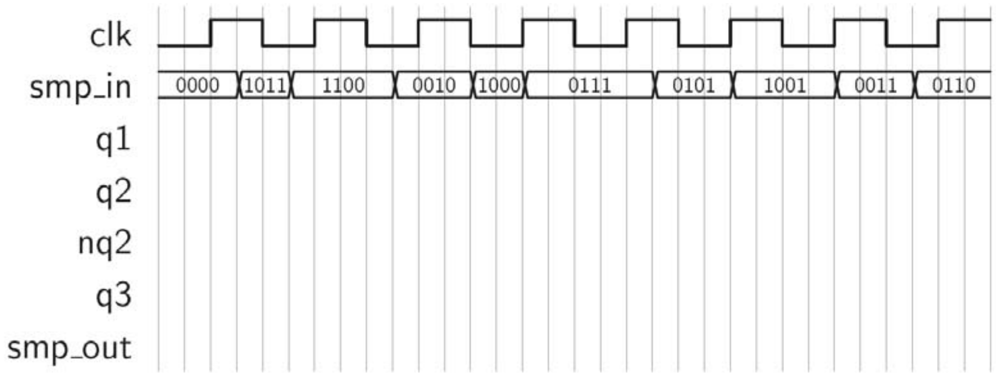
For the D-type latch and D flip-flop, the *in1* input is connected to the D input, and *in2* is left disconnected. For the SR-latches, *in1* is connected to the S input and *in2* is connected to the R input. For the JK flip-flop, *in1* is connected to the J input and *in2* is connected to the K input.

Complete the timing diagram showing the outputs of the four sequential circuits. Indicate any undefined output by shading the relevant area.



Q3. A bank of 4-bit registers is arranged as per the diagram below, to form a FIFO, with a bitwise inversion in the middle. The input shown in the diagram is applied to the leftmost register. All registers share the same clock. Show a timing diagram for the output at the rightmost register. A bitwise inversion simply inverts each bit of the signal individually. Write down the resulting hexadecimal number sequence at the output.





Q1.

```
module compmod (input [7:0] a, b, c,
                output reg [7:0] max, min,
                output [7:0] diff);

    assign diff = max – min;

    always @ * begin
    if (a>b) begin
        if (c>a) begin
            min = b;
            max = c;
        end
        else begin
            max = a;
            if (c>b) min = b;
            else min = c;
        end
    end
    else begin
        if (c>b) begin
            min = a;
            max = c;
        end
        else begin
            max = b;
            if (c>a) min = a;
            else min = c;
        end
    end

endmodule
```

Note: it is very easy to make syntax errors when using nested conditional if statements.

Q2 The rules:

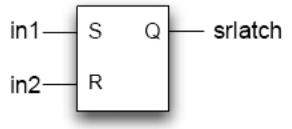
SR-Latch: ignore clk, when S (in1) is high, set to 1, when R (in2)is high, set to 0, when both are high, undefined output, that may go metastable on de-assertion.

Enabled SR-latch: As above, but only consider inputs when clk is high

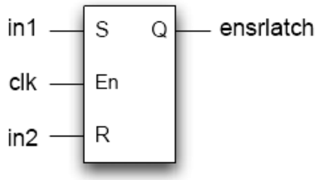
Enabled D-latch: When clk is high, in1 should pass to the output

JK-flip-flop: output only changes at edges of clk. J (in1)=1 sets output high, K(in2)=1 sets output low, Both J and K 1 toggles output. But all only considered at clock rising **edge**.

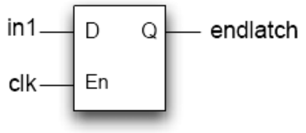
D-flip-flop: at rising edge, output will change to follow D (in1) input at that instance.



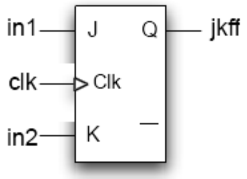
S	R	Q+
0	0	Q
0	1	0
1	0	1
1	1	undef



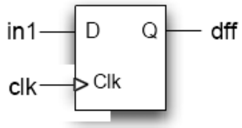
E	S	R	Q+
0	X	X	Q
1	0	0	Q
1	0	1	0
1	1	0	1
1	1	1	undef



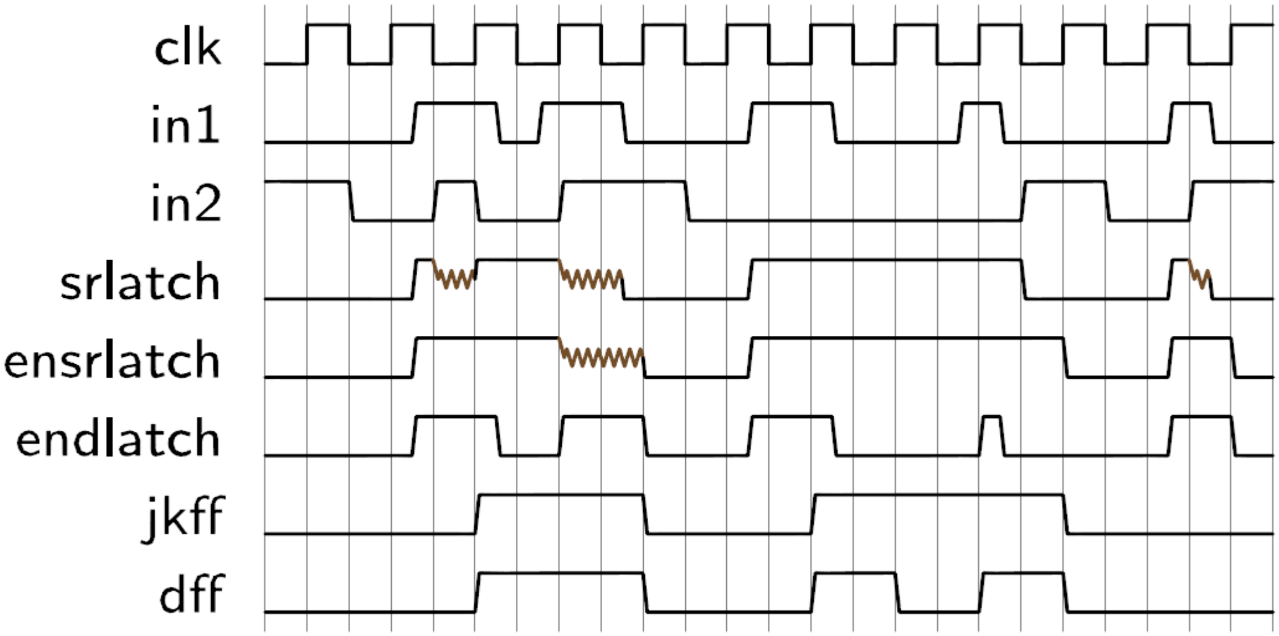
E	D	Q+
0	X	Q
1	0	0
1	1	1



clk	J	K	Q+
↑	0	0	Q
↑	0	1	0
↑	1	0	1
↑	1	1	Q’
	X	X	Q

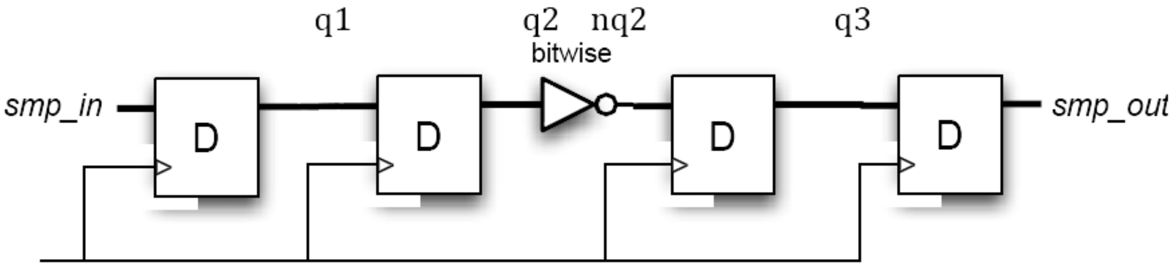


clk	D	Q+
↑	0	0
↑	1	1
	X	Q

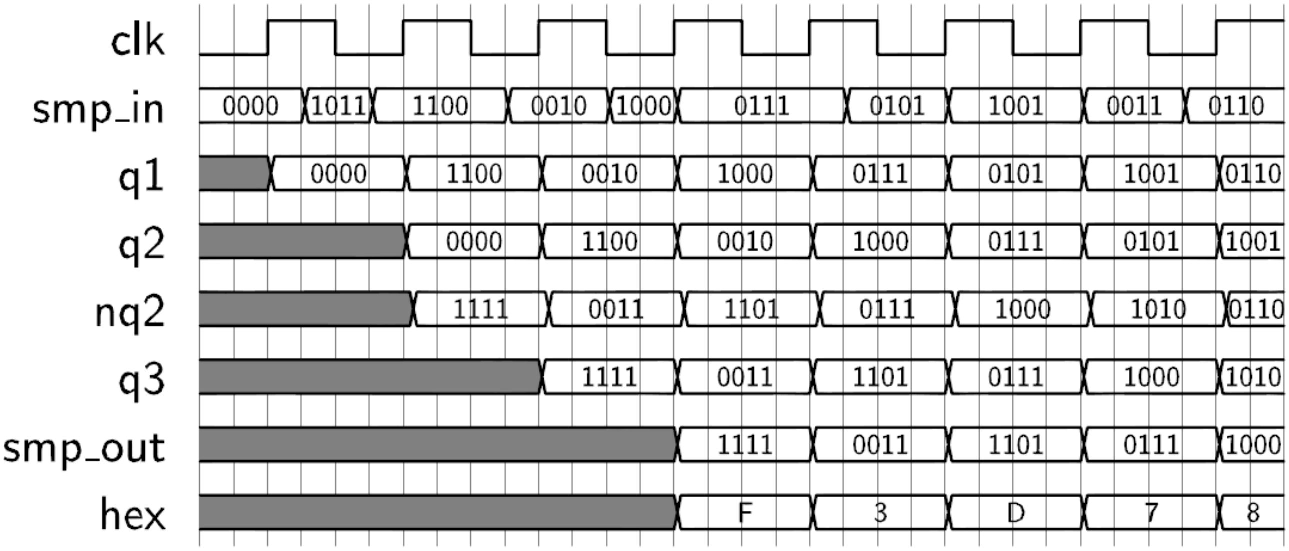


Note: the output of an SR latch is undefined if the S and R inputs are both 1.

Q3. The extra signals in the timing diagram should give a hint to how to do this, i.e. one step at a time. Annotate the diagram:



Remember each register's output only changes at the rising edge. Hence, some inputs are missed. Important to remember that the input value passed to the output is the one **just** before the rising edge.


clk
smp_in 0000 1011 1100 0010 1000 0111 0101 1001 0011 0110
q1 0000 1100 0010 1000 0111 0101 1001 0110
q2 0000 1100 0010 1000 0111 0101 1001
nq2 1111 0011 1101 0111 1000 1010 0110
q3 1111 0011 1101 0111 1000 1010
smp_out 1111 0011 1101 0111 1000
hex F 3 D 7 8