CSEC 20th – Past Year Paper Solution 2017-2018 Sem2

CE/CZ1005 – Digital Logic

Question 1

1)

a) 0x810 = 1000 0001 0000 1100

i) Unsigned = 33036

ii) Sign-magnitude = -268

iii) Two's complement = -32500

b) 128 < 150 < 255 = 8 bits for integer part

0.0625 < 0.1 < 0.125 = 4 bits for fraction part

Minimum number of bits store each value = 12 bits

c) F = [(u + v w') (u' x + w)'+ x (u' v')']'

= [(u + v w') (u' x + w)']' [x (u' v')']'

= [(u + v w')' + (u' x + w)] [x'+ (u' v')]

= [u' + (v w')' + u' x + w] [x' + u' v']

= [u' + (v' + w) + u' x + w] [x' + u' v']

= [u' v' + u' w + u' x + w] [x' + u' v']

= [u' v' x' + u' w x' + 0 + w x'] + [u' v' + u' w v' + u' x v' + w u' v']

= u' v' (x' + 1 + w + x + w) + (u' + 1) w x'

= u' v' + w x'

d)

i) 0x49 = 0100 1001

0xB6 = 1011 0110

```
    0100 1001
+   1011 0110
-------------
    1111 1111        - no overflow
```

0xF3 = 1111 001

0x7A = 011 1010

2's complement of 0x7A = 1000 0110

```
    1111 0011
+   1000 0110
-------------
   10111 1001        - overflow
```

ii) Arithmetic overflow occurs when an arithmetic operation between two N-bit operands produces a result that cannot be sufficiently represented by N bits. Rule to detect overflow in 2's complement addition:

- No overflow occurs if the two operands have opposite signs.
- Overflow is detected when operands have the same signs, but the sum has an opposite sign.

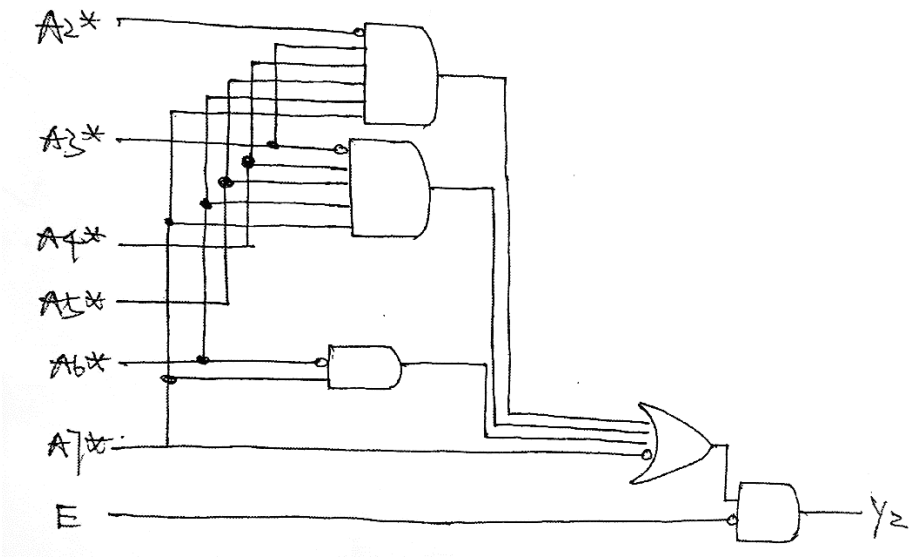
For subtraction, no overflow occurs if the operands have the same signs.

Question 2

2)

a)

i)



ii) The purpose of input E* is to enable or disable the circuit. When the E* is asserted, i.e. E* is active low, the circuit is enabled, and the outputs are allowed to change in response to changes in the inputs. When the E* is negated, i.e. E* is active high, the circuit is disabled and the outputs are not allowed to change.

b)

a,b \ c,d	00	01	11	10
00	1	1	0	1
01	1	0	X	0
11	1	1	1	0
10	X	1	0	X

F(a,b,c,d) = (a +b' + d') (a +c' +d') (b' + c' +d) (a' + b +c')

c)

i) Tristate. Two or more outputs can be connected together. However, at any time only one (or no) output should be enabled. Otherwise, it can lead to bus contention and damage the devices.

ii) When the inputs of all four logic gates are logical high, namely OR fate, NOT gate, NAND gate and XOR gate, the output is logical high. When any inputs of these four logic gates are logical low, the output is logical low.

iii) X = (A + B) (C D)' E' (F G' + F' G)

d) t_r – rise time from 10% of V_{DD} to 90% of V_{DD}
t_f – fall time from 90% of V_{DD} to 10% of V_{DD}
t_{PHL} – propagation delay when output changes from high to low
t_{PLH} – propagation delay when output changes from low to high

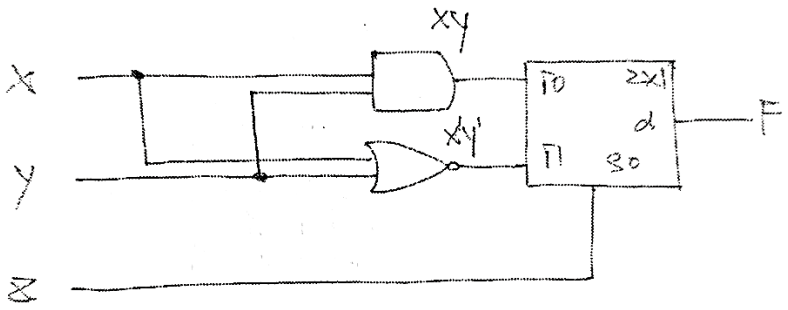
Question 3

3)

a) False – Modules can contain code to describe hardware and instances of other modules
False – Regs can only be assigned to from within always blocks
True – For the statements in an always block, the order matters
True – A n input multiplexer requires a $\log_2(n)$ – bit select
True – A parameter is a constant that is local to a module

b)

i)



ii)
```
assign F = ((~x&~y&z)|(x&y&~z))? 1: 0;
```

c)
```
module ALU ( input [30:0] a, b,
             input [2:0] OP,
             output reg [31:0] ALU_OUT);

        always @ *

             case (OP)

                  3'b000 : ALU_OUT = A + B;

                  3'b001 : ALU_OUT = A - B;

                  3'b010 : ALU_OUT = A & B;

                  3'b011 : ALU_OUT = A | B;

                  3'b100 : ALU_OUT = A ^ B;

                  default : ALU_OUT = 32'b0;

             endcase

endmodule
```

Question 4

4)

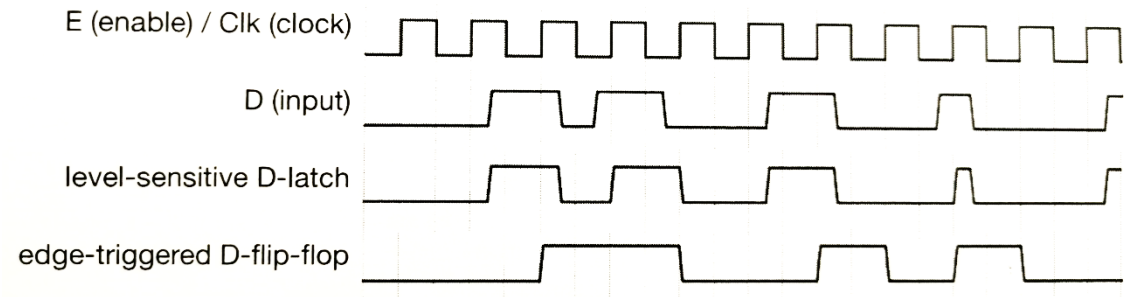
a) Level-sensitive D-latch -if enable is high, the input is passed through to the output, which is called transparent and if enable is low, the output is held

E(Enable)	**D (input)**	**Q (output)**	**Function**
0	X	Q	Store
1	0	0	Transparent
1	1	1	Transparent

Edge-triggered D-flip-flop – at each clock rising edge each flip flop will take its D input and produce that value on its Q output.

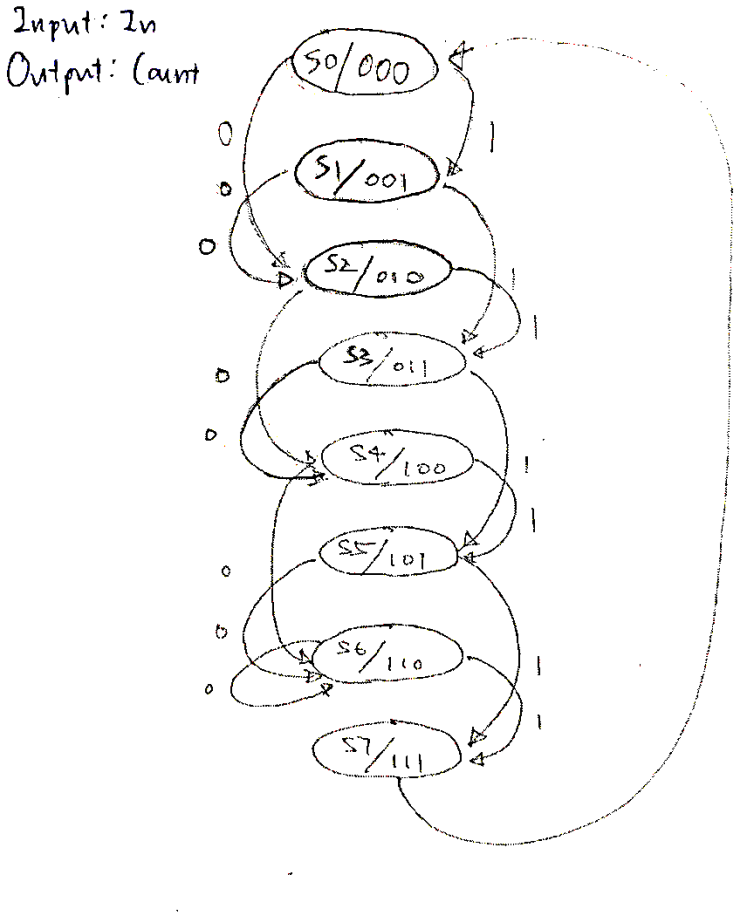
Clk (Clock)	**D (Input)**	**Q (Output)**	**Function**
0	X	Q	Store
0 -> 1	0	0	Take value
0 -> 1	1	1	Take value
1	X	Q	Store

timing diagram



b)

i)



ii) The output Count = 000

iii)

```
module counter (input CLK, RESET,
                input In, output [2:0] Count);
      reg [2:0] nst = st;
      assign Count = st;
      always @ (posedge clk)
      begin
            if (RESET)
                  st <= 3'b000;
            else
                  st <= nst;
      end

      always @ *
      begin
            nst = st;
            case (st)
                  3'b000:
                  begin
                        if (In)
                              nst = 3'b001;
                        else
                              nst = 3'b010;
                  end
                  3'b001:
                  begin
                        if (In)
                              nst = 3'b011;

                        else
                              nst = 3'b010;
                  end
                  3'b010:
                  begin
                        if (In)
                              nst = 3'b011;

                        else
                              nst = 3'b100;
                  end
                  3'b011:
                  begin
                        if (In)
                              nst = 3'b101;

                        else
                              nst = 3'b100;
                  end
```

```
                    3'b100:
                    begin
                        if (In)
                            nst = 3'b101;

                        else
                            nst = 3'b110;
                    end
                    3'b101:
                    begin
                        if (In)
                            nst = 3'b111;

                        else
                            nst = 3'b110;
                    end
                    3'b110:
                    begin
                        if (In)
                            nst = 3'111;

                        else
                            nst = 3'b110;
                    end
                    3'b010:
                        nst = 3'b011;
                endcase
            end

        endmodule
```

==End of Answers==

Solver: Wu Ziang

Email Address ZWU018@e.ntu.edu.sg