**21st CSEC – Past Year Paper Solution (2019 – 2020 Semester 1)**

**CE/CZ 1005 Digital Logic**

1 **(a)** **(i)**

| | | |
|---|---|---|
| 16 \| 1835 | | |
| 16 \| 114 | 11 -> B | |
| 7 | 2 | |

Ans. $1835_{10} = 72B_{16}$

**(ii)** Each digit in ASCII code is 7 bits so 8 digits are 56 bits.

Each digit in BCD is 4 bits so 8 digits are 32 bits.

**(iii)** $ACE6_{16} = (\underline{\mathbf{1}}010\ 1100\ 1101\ 0110)_2$

Receiver checks parity – odd and concludes no error. Therefore, transmitter and receiver agree on odd parity.

**(iv)** 2's complement number: 10010111
Taking 1's complement, we get 01101000
Adding 1, we get 01101001
$(01101001)_2 = 1+8+32+64 = 105_{10}$
Signed decimal value = -105

Alternate method:-
Since $\underline{\mathbf{1}}0010111$ is a 2's complement number, the most significant bit (bold and underlined) determines the negative sign of the number, the decimal conversion can be carried out directly as –
$(10010111)_2 = (-1)x2^7 + 2^4 + 2^2 + 2^1 + 2^0 = -128 + 16 + 4 + 2 + 1 = -105$

**(b)**

F = (w'+x+y+z') (w'y+xz')' (w'y'+xz)'
= (w'+x+y+z') (w'y)' (z')' (w'y')' (xz)' [De Morgan's law]
= (w'+x+y+z') (w+y') (z) (w+y) (x'+z') [De Morgan's law and Involution law]
= (w) (z) (x'+z') (x+y+w'+z') [Distributive law]
= (wzx') (x+y+w'+z') [Distributive law]
= (wzx'y) [Distributive law]

**(c)**

G (a, b, c, d) = (a XOR b) AND (c XNOR d)
= $(a \oplus b) . (c \odot d)$
= (ab'+a'b) (cd+c'd')

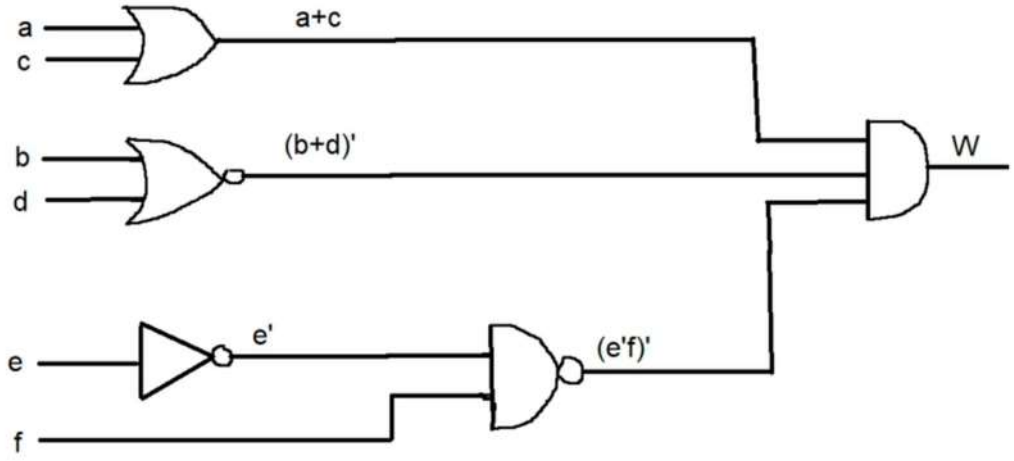
| ab \ cd | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 0 | 0 |
| 01 | 1 | 0 | 1 | 0 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 1 | 0 | 1 | 0 |

Sum of minterms = a'bc'd'+a'bcd+ab'c'd'+ab'cd

| ab \ cd | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 0 | 0 |
| 01 | 1 | 0 | 1 | 0 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 1 | 0 | 1 | 0 |

Product of maxterms = (a+b)(c+d')(c'+d)(a'+b')

**(d)** W = (a+c) (b+d)' (e'f)'




W is low when –

i) Both a and c are low

OR

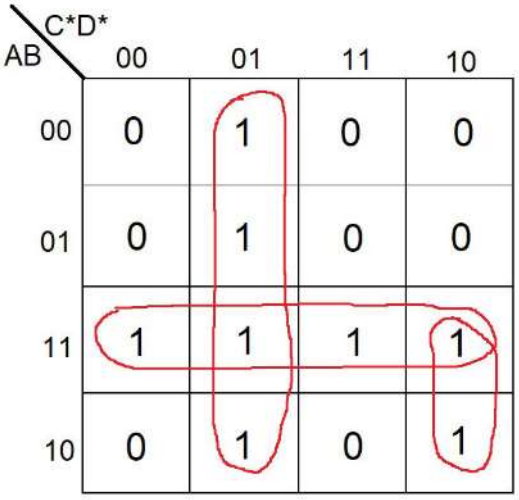
ii)

| b | d |
|---|---|
| 0 | 1 |
| 1 | 0 |
| 1 | 1 |

OR

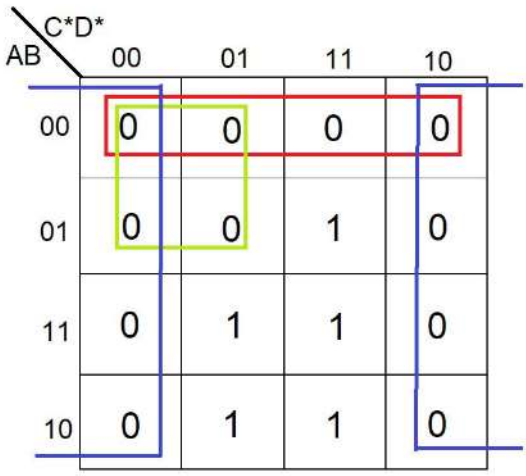
ii) e is low and f is high

**2** **(a)** F = AB + (C*)' (D*) + (C*)' (D*) + A (C*) (B)' (D*)'



Minimum cost SOP expression for F = AC*D + AB + CD*

G* = AD* + (A'BC*D*)




Minimum cost POS expression for G* = (A+C*) (A+B) (D*)

**(b)** **(i)** Propagation delay is the average transition delay time for signal to propagate from input to output.

**(ii)** Carry propagation is the propagation of the carry's addition. When two numbers are summed, if they overflow their place, we carry a one to the next place.

**(iii)** Carry propagation can be reduced (hence increasing the speed of addition) by a special- purposed logic circuit called carry look-ahead circuit. A carry look-ahead adder improves speed by reducing the amount of time required to determine carry bits

**(c)** **(i)** <u>Device X</u>

High state DC noise margin = $V_{OH} - V_{IH}$ = 4.2 – 3.5 = 0.7 volts

Low state DC noise margin = $V_{IL} - V_{OL}$ = 1.5 – 0.7 = 0.8 volts

Fan-out = min( $\lfloor I_{OL} / I_{IL} \rfloor$ , $\lfloor I_{OH} / I_{IH} \rfloor$ )

= min (18, 15)

= 15

<u>Device Y</u>

High state DC noise margin = 4.5 – 3.6 = 0.9 volts

Low state DC noise margin = 1.8 – 0.5 = 1.3 volts

Fan-out = min( $\lfloor I_{OL} / I_{IL} \rfloor$ , $\lfloor I_{OH} / I_{IH} \rfloor$ )

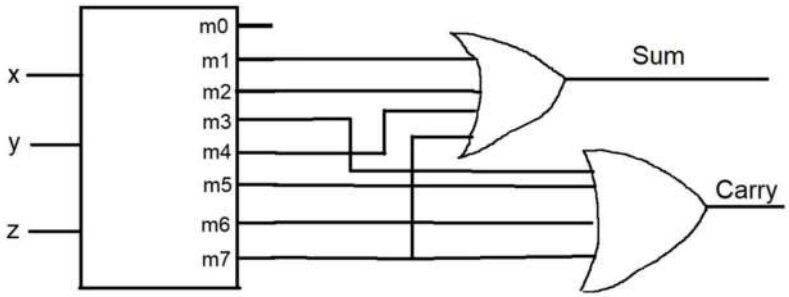
= min(13, 15) = 13

**(ii)** Yes device X will be able to drive device Y. Fan-out of device X is 15. Fan-out of device Y is 13. Fan-out specifies the number of standard loads that the output gate can drive. Since fan-out of X is greater than fan-out of Y, device X can drive device Y.

**3** **(a)**

x = 7'b1101101
x = 7'b0011011
x = 7'b1101111
x = 7'b0000001

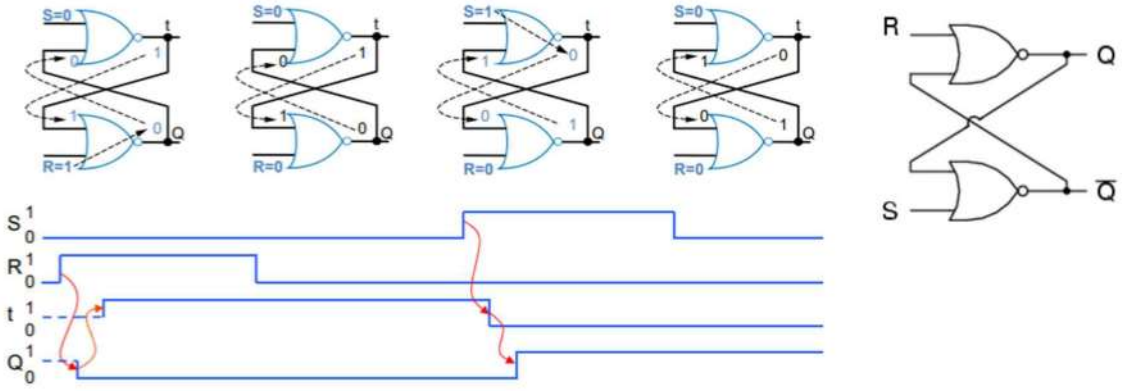
**(b)**




**(c)** Lets assume from the perspective of each ant, a movement to the Left is a '0' and the Right is a '1'. Then, for NO collision, they must all go Left or all go Right. So, we need to detect 000 or 111 = no_col = 0.

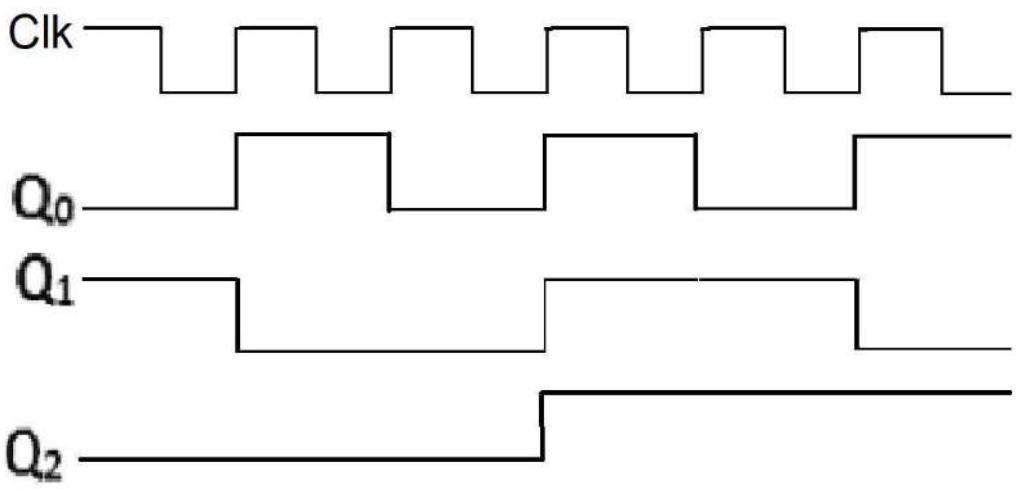
```
module col_detect (input d1, d2, d3, output col);
   assign col = ({d1,d2,d3} == 3'b000) ? 0 : (({d1,d2,d3} == 3'b111) ? 0:1);
endmodule
```

**4** **(a)** SR latch is the most basic circuit used for storing a bit.



An SR has two stable states, i.e., 0 and 1. The Q and not-Q outputs are supposed to be in opposite states. making both the S and R inputs equal to 1 result in both Q and not-Q being 0. For this reason, having both S and R equal to 1 is called an invalid or illegal state for the SR latch.

**(b)**




**(c)** **(i)** It is a Mealy FSM because the output can change in mid-state.

**(ii)**

```
module FSM (Input RESET, CLK, T, output reg Y);
parameter sta = 2'b00, stb = 2'b01, stc = 2'b10, std = 2'b11;
reg [1:0] n, s;
always @* begin
        if (RESET)
            s<=sta;
        else  begin
                n = s; Y = 1'b0;
                 case (s)
                     sta: if(T) n = stb;
                     stb: if(T) n = stc; else n = sta;
                     stc: begin
                             if(T) n = std;
                             else n = stc;
                          end
                     std: if(!T) n = stc;
                  endcase
               end
            end
endmodule
```

Solver: Parashar Kshitij
email ID: KSHITIJ003@e.ntu.edu.sg