ANSWERS FOR TMA 01

EEX5351

Digital Electronic System

By

S.A. Pushpitha Kavinda

Reg No: 617143597

Submitted to,

Department of Electrical and Computer Engineering

Faculty of Engineering Technology

The Open University of Sri Lanka

At

Colombo Regional Center

Due Date

15/07/2022

Q1)

For F1(w,x,y,z)=∑(0,1,5)

y,z \ w,x	00	01	11	10
00	0 1	4	12	8
01	1 1	5 1	13	9
11	3	7	15	11
10	2	6	14	10

F1 = w' x' y' + y' z w'

For F2(w,x,y,z) = ∏(3,5,9,7)

y,z \ w,x	00	01	11	10
00	0 1	4 1	12 1	8 1
01	1 1	5 0	13 1	9 0
11	3 0	7 0	15 1	11 1
10	2 1	6 1	14 1	10 1

F2 =(y'+z'+w)(w+x'+z')(w'+x+y+z') -POS form

F2= z' + wx + wy +y'w'x'-SOP form

For F3(w,x,y,z)=∑(2,6,10,12,14,15)

y,z \ w,x	00	01	11	10
00	0	4	12 1	8
01	1	5	13	9
11	3	7	15 1	11
10	2 1	6 1	14 1	10 1

F3 =(yz' + wxy + wxz')

For F4(w,x,y,z)=∑(3,5,9,7)

y,z \ w,x	00	01	11	10
00	0	4	12	8
01	1	5 1	13	9 1
11	3 1	7 1	15	11
10	2	6	14	10

F4 =(yzw' + w'xz + wx'y'z)

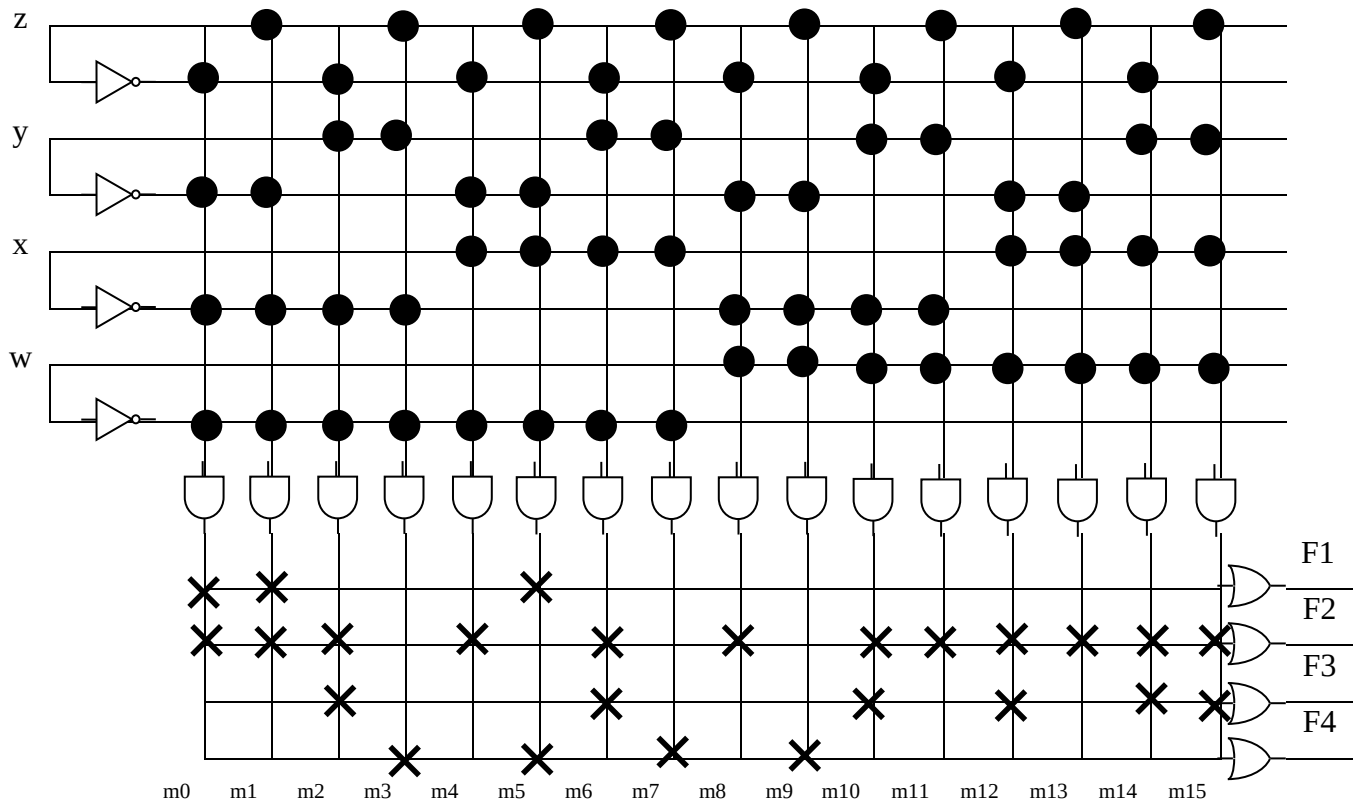
F4' = (y'+z'+w)(w+x'+z')(w'+x+y+z') = F2

b)

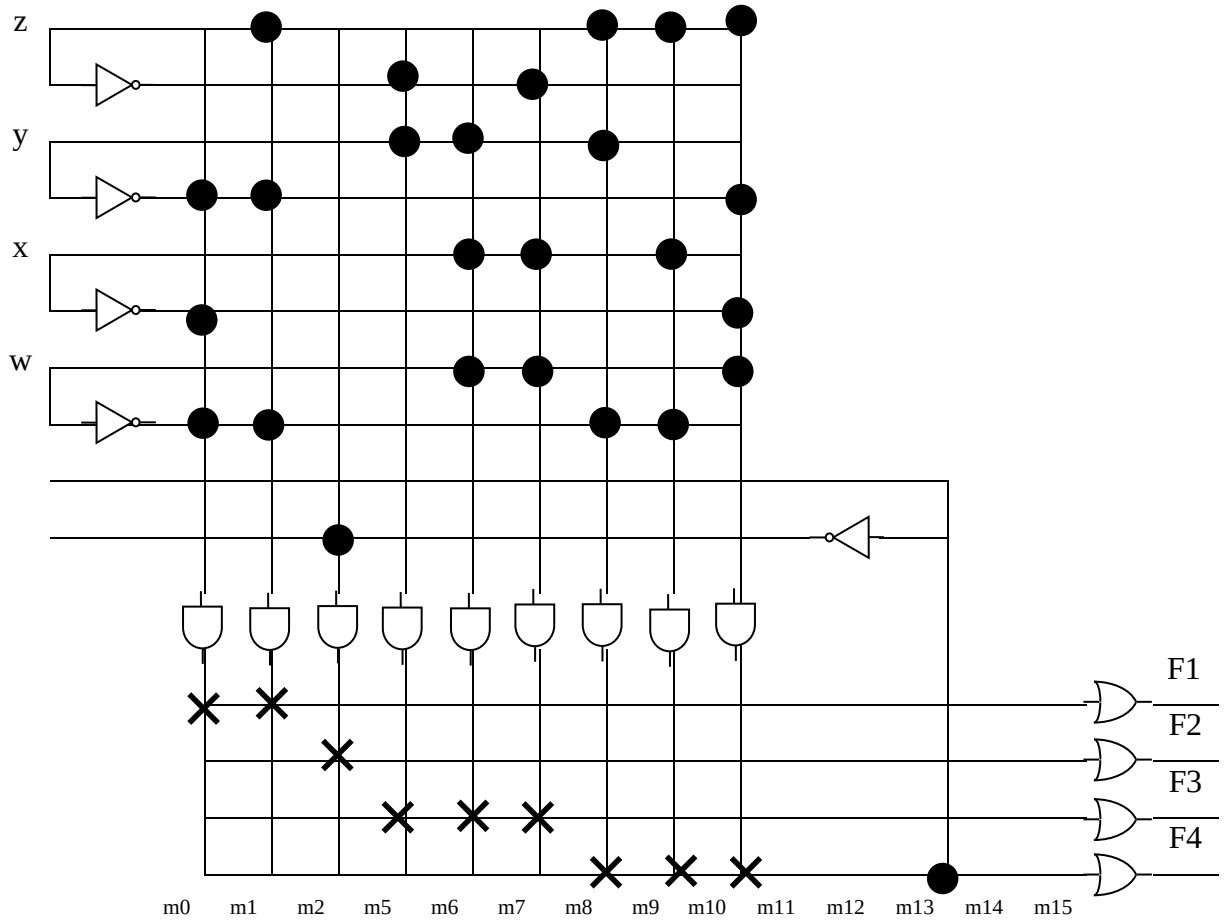
Row #	w	x	y	z	F1	F2	F3	F4
0	0	0	0	0	1	1	0	0
1	0	0	0	1	1	1	0	0
2	0	0	1	0	0	1	1	0
3	0	0	1	1	0	0	0	1
4	0	1	0	0	0	1	0	0
5	0	1	0	1	1	0	0	1
6	0	1	1	0	0	1	1	0
7	0	1	1	1	0	0	0	1
8	1	0	0	0	0	1	0	0
9	1	0	0	1	0	0	0	1
10	1	0	1	0	0	1	1	0
11	1	0	1	1	0	1	0	0
12	1	1	0	0	0	1	1	0
13	1	1	0	1	0	1	0	0
14	1	1	1	0	0	1	1	0
15	1	1	1	1	0	1	1	0

C)

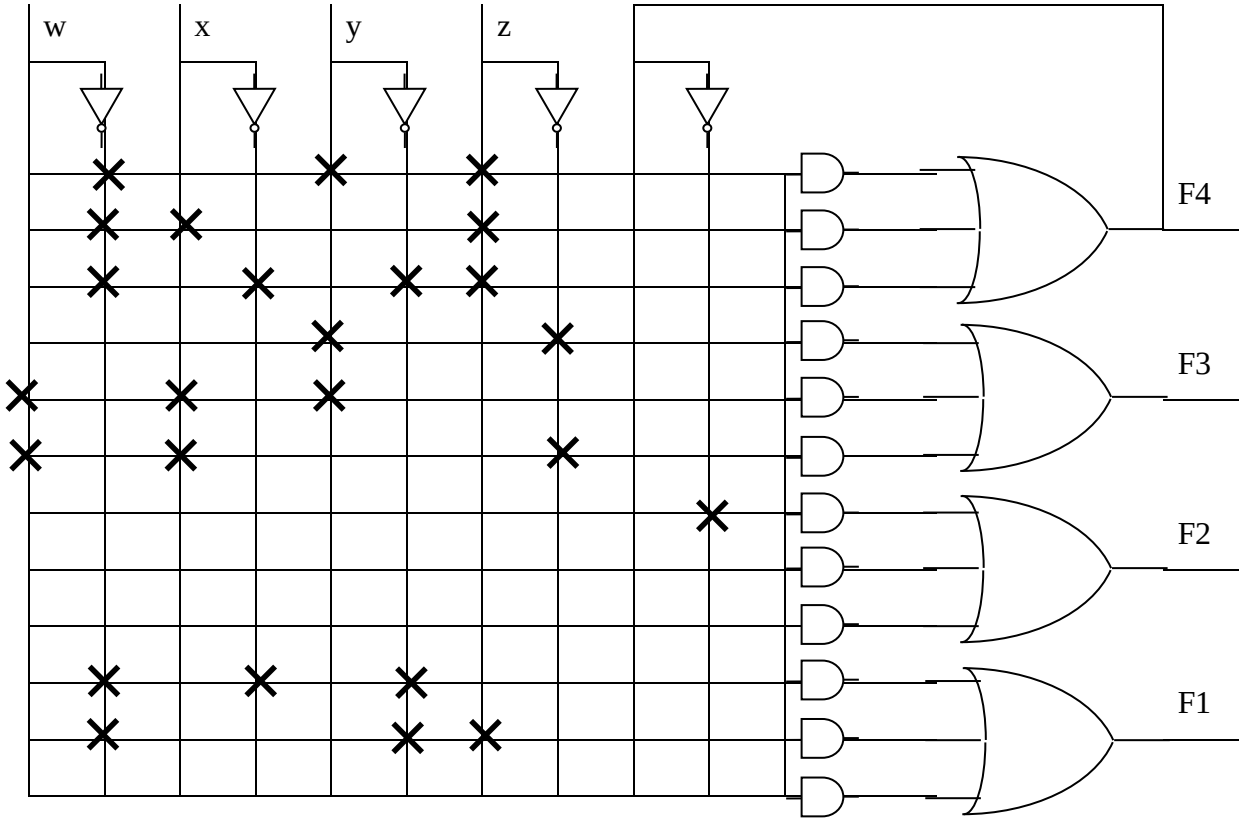
i) ROM (Size : $2^N \; x \; M = 2^4 \; x \; 4 = 64 \; bit = 8 \; byte$)



ii) PLA(5 inputs, 9 product terms, 4 outputs)



iii)PAL (Size: 5 input, 4 output, 12 product terms – 3 for each output)

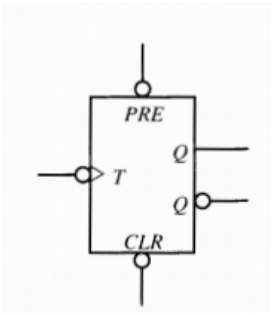


d)

	ROM	PLA	PAL
# Of inputs	4	5	5
# Of Feedback inputs	Not typically included	1	1
# Of product terms	16	9	12
# Of outputs	4	4	4
# Of bidirectional pins	0	0	0
Programmable Array	OR Array	Both Or and AND Array	AND Array
Fixed Array	AND Array	-	OR Array
Design Technique	canonical SOP form must be use. No advantage to minimizing the function	Concentrate on reducing number of products	Uses Minimum SOP terms
Design Complexity	Medium	High	Low

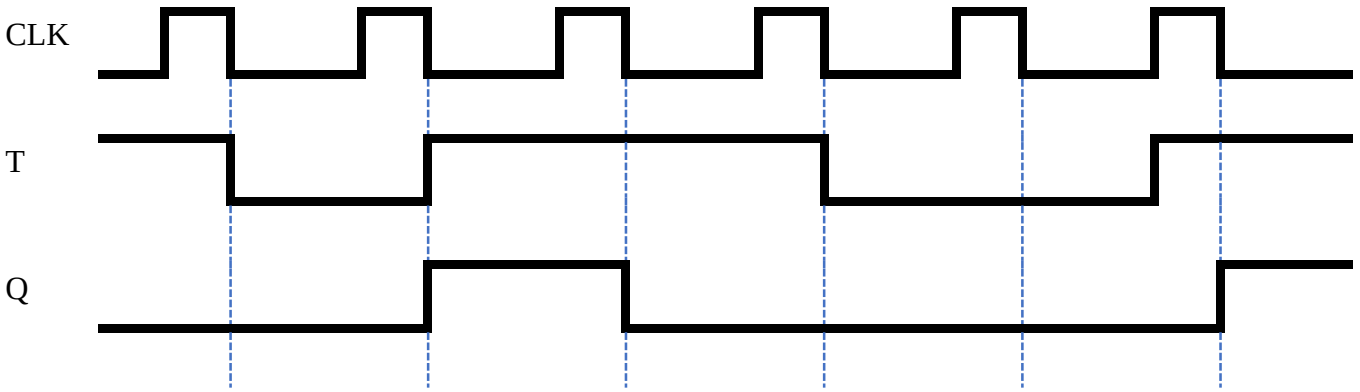
Q2)

a)



Let's consider the negative edge triggered T Flip Flop

When Idle case,



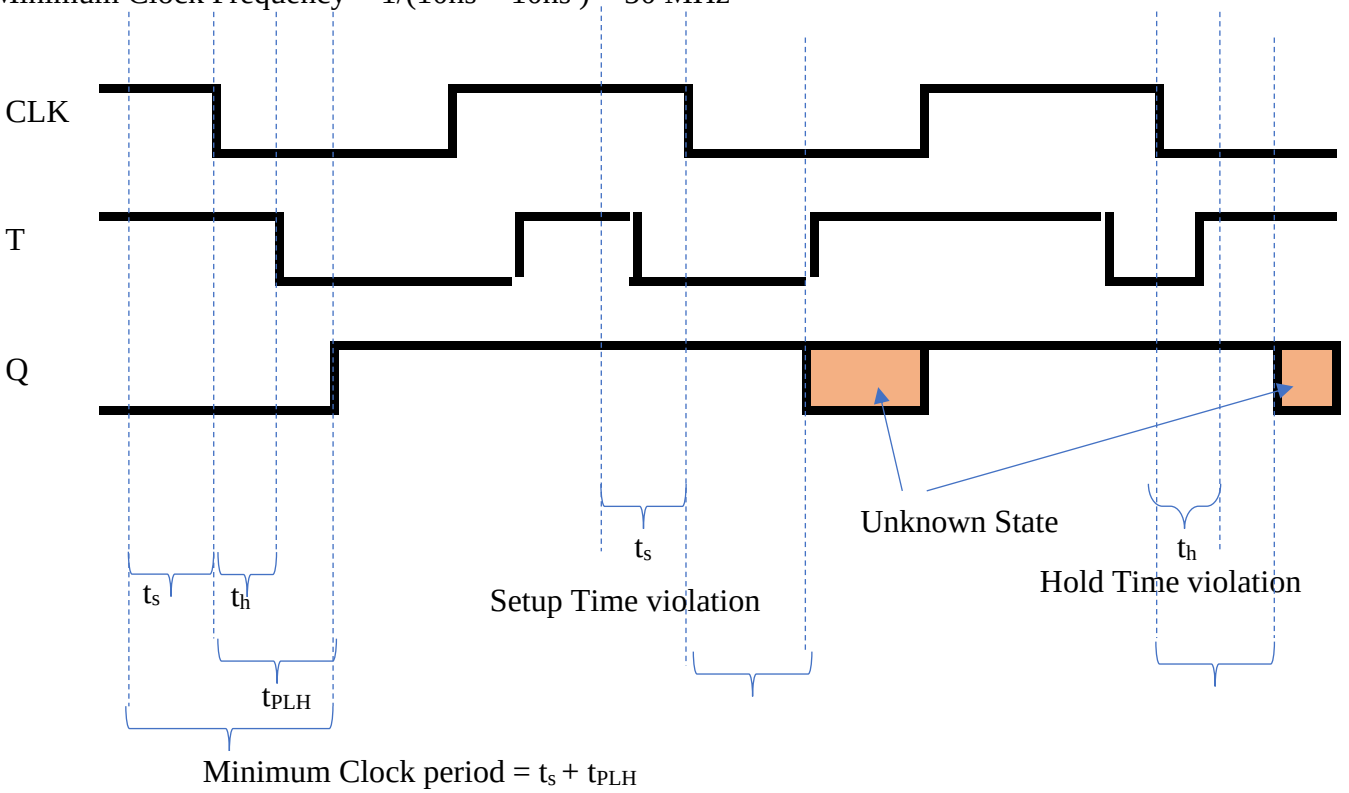
When Practical case,

Assume at Vcc = +5V test conditions,

t_{PLH} (Max) = 10 ns, t_{PHL} (Max) = 10ns

Setup Time t_s = 10ns, Hold Time t_h = 5ns

Minimum Clock Frequency = 1/(10ns + 10ns) = 50 MHz



b) Clock Period = $t_s + t_{PD}$

Where t_s = setup time

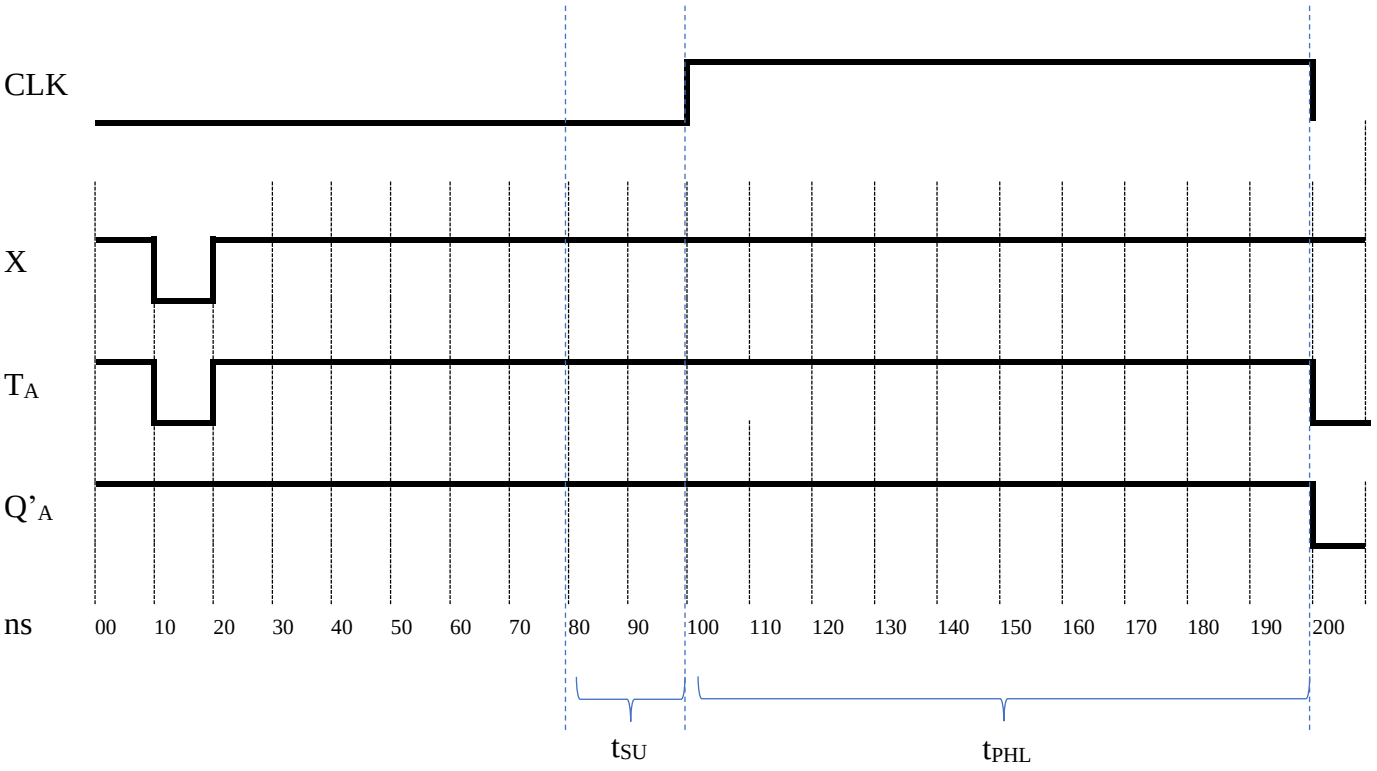
t_{PD} = Propagational Delay

c)

Assume the average propagational delay of AND Gate is 0ns

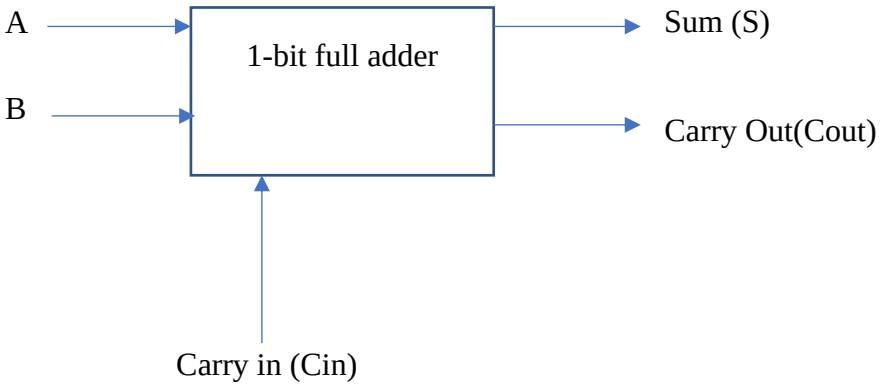
Assume Initially Q_A' = 1

minimum Clock period = 1/10^(-6) = 100 ns



Q3)

a)



b)

```
architecture adder_dataflow of full_adder is
begin
    sum <= (x xor y) xor cin;
    cout <= (x and y) or (x and cin) or (y and cin);
end adder_dataflow;
```

c)

```
LIBrary IEEE;
use IEEE.std_logic_1164.all;

entity full_adder_test is
end full_adder_test;

architecture my_test of full_adder_test is
component full_adder
    port(x,y,cin:in std_logic;
    sum,cout :out std_logic);
end component;

for U1: full_adder use entity work.full_adder(adder_dataflow);
    signal X_s,Y_s  : std_logic;
    signal cin_s    :std_logic;
    signal sum_s    :std_logic;
    signal cout_s   :std_logic;

begin

U1: full_adder port map (X_s, Y_s, CIN_s, SUM_s, COUT_s);

process
begin
```

```
--case 0  :0+0 with carry 0
    X_s <= '0';
    Y_s <= '0';
    Cin_s <= '0';
    wait for 10ns;
    assert(sum_s = '0') report "Faild case 0-SUM" severity error;
    assert(COUT_s = '0') report "Faild case 0-COUT" severity error;
    wait for 40ns;

--case 1  :0+0 with carry 1
    X_s <= '0';
    Y_s <= '0';
    Cin_s <= '0';
    wait for 10ns;
    assert(sum_s = '1') report "Faild case 1-SUM" severity error;
    assert(COUT_s = '0') report "Faild case 1-COUT" severity error;
    wait for 40ns;

--case 2  :0+1 with carry 0
    X_s <= '0';
    Y_s <= '1';
    Cin_s <= '0';
    wait for 10ns;
    assert(sum_s = '1') report "Faild case 2-SUM" severity error;
    assert(COUT_s = '0') report "Faild case 2-COUT" severity error;
    wait for 40ns;
```

```
--case 3  :0+1 with carry 1
    X_s <= '0';
    Y_s <= '1';
    Cin_s <= '1';
    wait for 10ns;
    assert(sum_s = '0') report "Faild case 3-SUM" severity error;
    assert(COUT_s = '1') report "Faild case 3-COUT" severity error;
    wait for 40ns;

--case 4  :1+0 with carry 0
    X_s <= '1';
    Y_s <= '0';
    Cin_s <= '0';
    wait for 10ns;
    assert(sum_s = '1') report "Faild case 4-SUM" severity error;
    assert(COUT_s = '0') report "Faild case 4-COUT" severity error;
    wait for 40ns;

--case 5  :1+0 with carry 1
    X_s <= '1';
    Y_s <= '0';
    Cin_s <= '1';
    wait for 10ns;
    assert(sum_s = '0') report "Faild case 5-SUM" severity error;
    assert(COUT_s = '1') report "Faild case 5-COUT" severity error;
    wait for 40ns;
```

```
--case 6  :1+1 with carry 0
    X_s <= '1';
    Y_s <= '1';
    Cin_s <= '0';
    wait for 10ns;
    assert(sum_s = '0') report "Faild case 6-SUM" severity error;
    assert(COUT_s = '1') report "Faild case 6-COUT" severity error;
    wait for 40ns;

--case 7  :1+1 with carry 1
    X_s <= '1';
    Y_s <= '1';
    Cin_s <= '1';
    wait for 10ns;
    assert(sum_s = '1') report "Faild case 7-SUM" severity error;
    assert(COUT_s = '1') report "Faild case 7-COUT" severity error;
    wait for 40ns;

end process;
end my_test;
```