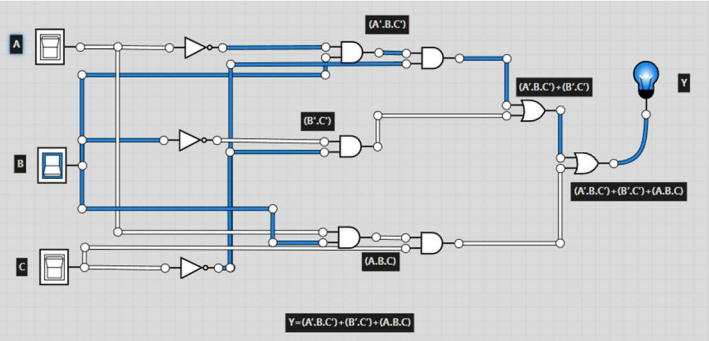
Dylan Liesenfelt
CDA3203
19 June 2024

Class Activity 2

Y=(A'.B.C')+(B'.C')+(A.B.C)



A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

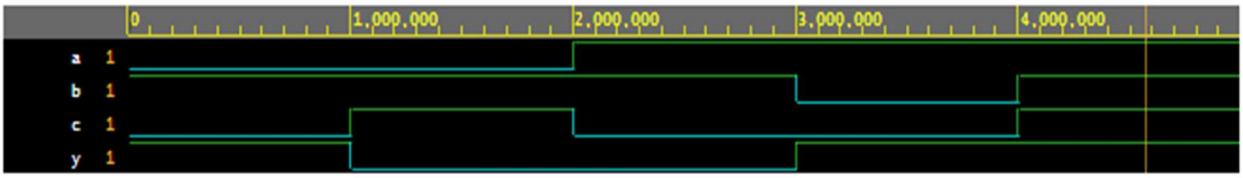
Design:

```
library IEEE;
use IEEE.std_logic_1164.all;

-- ENTITY
entity Circut_Y is
port(
    A, B, C: in std_logic;
    Y: out std_logic);
end Circut_Y;

-- ARCHITECTURE
architecture dataflow of Circut_Y is
signal and1, and2, and3, or1: std_logic;
begin
    and1 <= not(A) and B and not(C);
    and2 <= not(B) and not(C);
    and3 <= A and B and C;
    or1 <= and1 or and2 or and3;
    y <= or1;
end dataflow;
```

Testbench:

```
library IEEE;
use IEEE.std_logic_1164.all;

-- TESTBENCH ENTITY
entity testbench is
--empty
end testbench;

architecture tb of testbench is
-- DUT COMPONENT
component Circut_Y is
port(
    A, B, C: in std_logic;
    Y: out std_logic);
end component;

signal A_IN, B_IN, C_IN, Y_OUT : std_logic;

begin

-- CONNECT DUT
DUT: Circut_Y port map(A_IN, B_IN, C_IN, Y_OUT);

    process
    begin
        A_IN <='0';
        B_IN <='1';
        C_IN <='0';
        wait for 1 ns;
        A_IN <='0';
        B_IN <='1';
        C_IN <='1';
        wait for 1 ns;

        A_IN <='1';
        B_IN <='1';
        C_IN <='0';
        wait for 1 ns;

        A_IN <='1';
        B_IN <='0';
        C_IN<='0';
        wait for 1 ns;

        A_IN <='1';
        B_IN <='1';
        C_IN<='1';
        wait for 1 ns;

        -- CLEAR INPUTS
        A_IN <='0';
        B_IN <='0';
        C_IN <='0';
        wait;
    end process;
end tb;
```

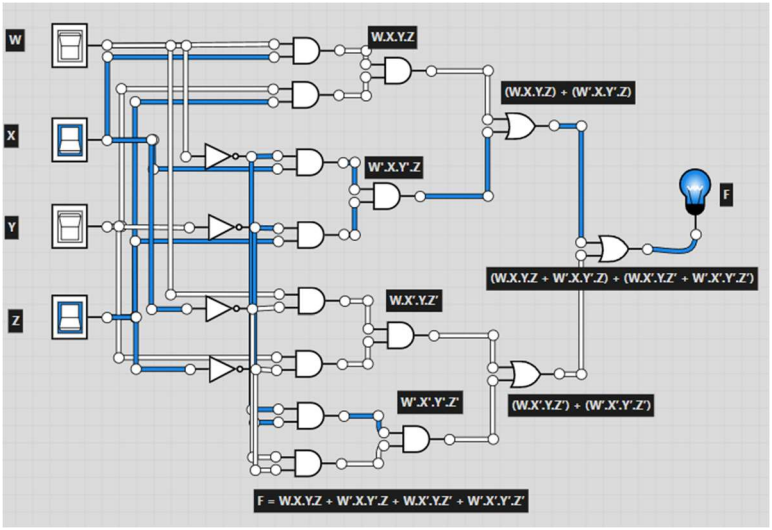
Wave:



Observation:

Wave and Truth Table are the same for the given test cases (010, 011, 110, 100, 111), Implementation successful. The trickiest part was organizing the circuit, logicly.ly really helped with and helped me plan out how my circuit was going to be written in VHDL

F(W,X,Y,Z) = WXYZ + W'XY'Z + WX'YZ' + W'X'Y'Z'



W	X	Y	Z	F
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

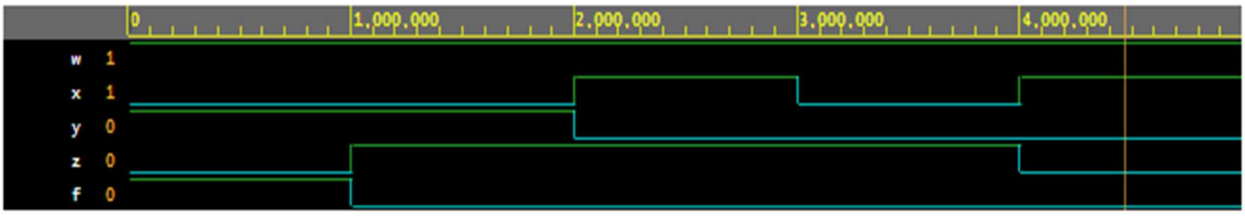
Design:

```
library IEEE;
use IEEE.std_logic_1164.all;

-- ENTITY
entity Circut_F is
port(
    W, X, Y, Z: in std_logic;
    F: out std_logic);
end Circut_F;

-- ARCHITECTURE
architecture dataflow of Circut_F is
signal and1, and2, and3, and4, or1: std_logic;
begin
    and1 <= W and X and Y and Z;
    and2 <= not(W) and X and not(Y) and Z;
    and3 <= W and not(X) and Y and not(Z);
    and4 <= not(W) and not(X) and not(Y) and not(Z);
    or1 <= and1 or and2 or and3 or and4;
    F <= or1;
end dataflow;
```

Testbench:

```
library IEEE;
use IEEE.std_logic_1164.all;

-- TESTBENCH ENTITY
entity testbench is
--empty
end testbench;

architecture tb of testbench is
-- DUT COMPONENT
component Circut_F is
port(
    W, X, Y, Z: in std_logic;
    F: out std_logic);
end component;

signal W_IN, X_IN, Y_IN, Z_IN, F_OUT : std_logic;

begin
-- CONNECT DUT
DUT: Circut_F port map(W_IN, X_IN, Y_IN, Z_IN, F_OUT);

    process
    begin
        W_IN <='1';
        X_IN <='0';
        Y_IN <='1';
        Z_IN <='0';
        wait for 1 ns;
        W_IN <='1';
        X_IN <='0';
        Y_IN <='1';
        Z_IN <='1';
        wait for 1 ns;

        W_IN <='1';
        X_IN <='1';
        Y_IN <='0';
        Z_IN <='1';
        wait for 1 ns;

        W_IN <='1';
        X_IN <='1';
        Y_IN <='0';
        Z_IN <='0';
        wait for 1 ns;

        W_IN <='1';
        X_IN <='0';
        Y_IN <='0';
        Z_IN <='1';
        wait for 1 ns;

        W_IN <='1';
        X_IN <='1';
        Y_IN <='0';
        Z_IN <='0';
        wait for 1 ns;

        -- CLEAR INPUTS
        W_IN <='0';
        X_IN <='0';
        Y_IN <='0';
        Z_IN <='0';
        wait;
    end process;
end tb;
```

Wave:



Observation:

Wave and Truth Table are the same for the given test cases (1010, 1011, 1101, 1001, 1100), Implementation successful. Starting to get a bit messy now with four inputs but once again logicly.ly really helped me organize and map the circuit.