# Experiment 3: **Multiplier**

**Prajwal Kishor Kalpande**, Roll Number:**200070028**

**EE214, WEL, IIT BOMBAY**

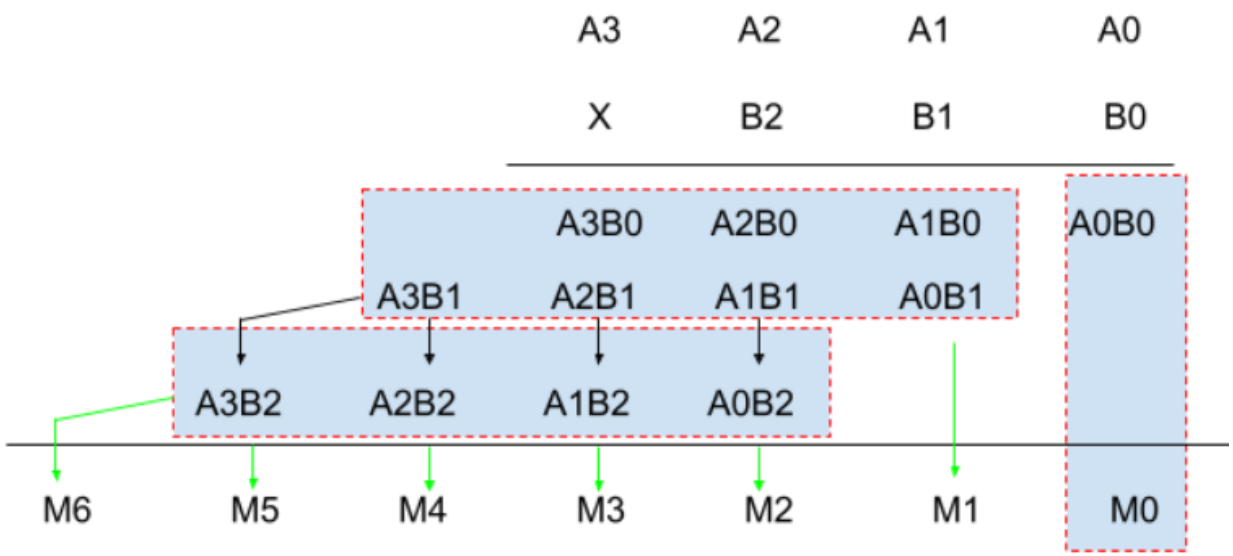
September 7, 2021

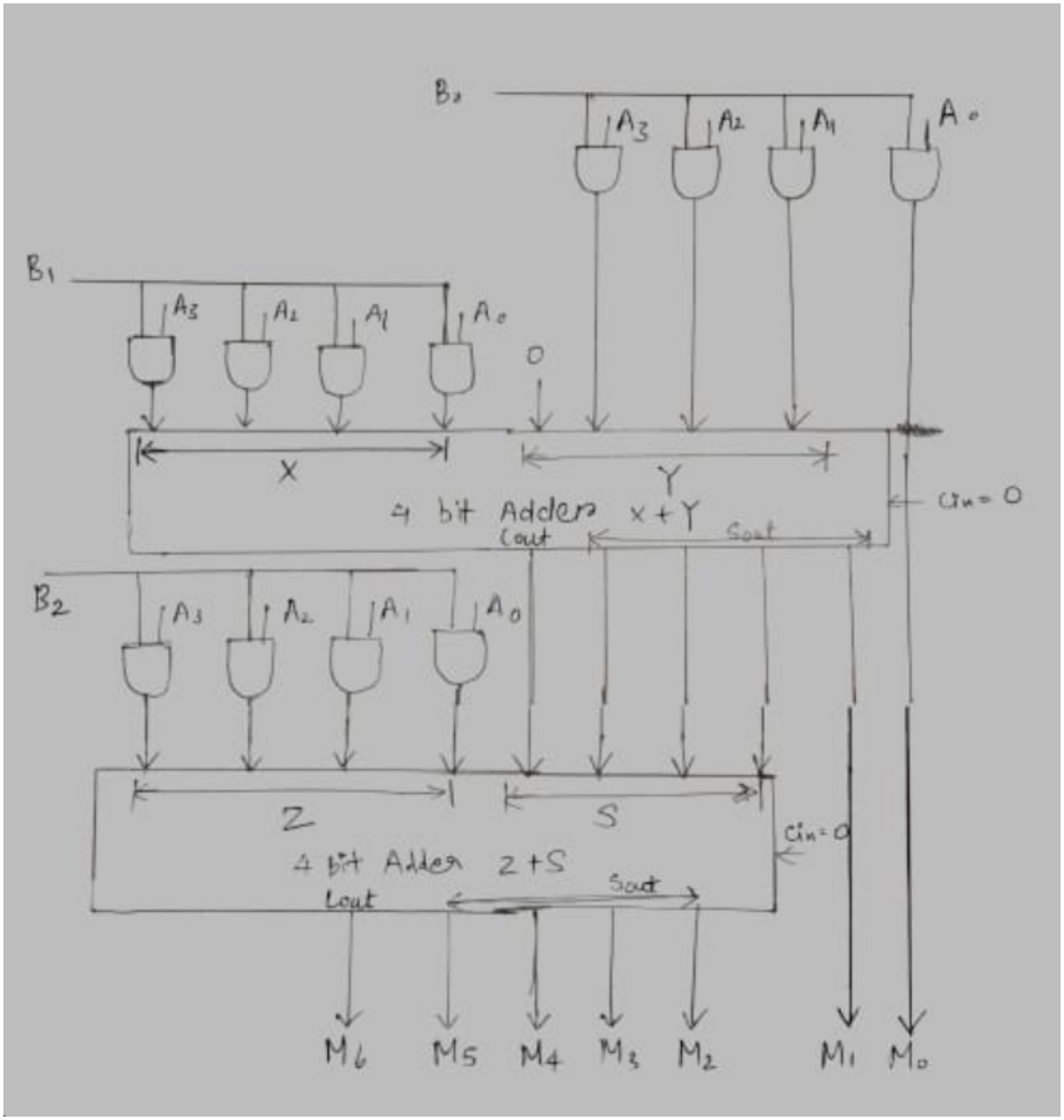
## Overview of the Experiment :

In this experiment we have to design a circuit to multiply two binary numbers i.e. one 4-bit input and one 3-bit input and the output is a 7-bit binary number.

## Approach to the Experiment :

The multiplier has one 4-bit input A3,A2,A1,A0 and one 3-bit input B2, B1, B0. The multiplication done is similar to the one in the decimal system and we need adders to add the respective one bit numbers all having the same place value.
On multiplication we get three rows, out of which we add the first two to get an intermediate sum which is then added to the last row to get the output. The addition operation is performed using the adders noting the carries and which bits of sum correspond to which bit of output.




B0
A3
A2
A1
A0
B1
A3
A2
A1
A0
0
X
Y
4 bit Adder x+Y
Cout
Sout
Cin=0
B2
A3
A2
A1
A0
Z
S
Cin=0
4 bit Adder Z+S
Cout
Sout
M6
M5
M4
M3
M2
M1
M0

# Design document and VHDL code :

The multiplier is made using twelve AND gates and two 4-bit adder gates. We have one 4-bit input A3,A2,A1,A0 and one 3-bit input B2, B1, B0. The output of the system is a 7-bit number M6 being MSB and M0 as LSB. The VHDL code is as follows:

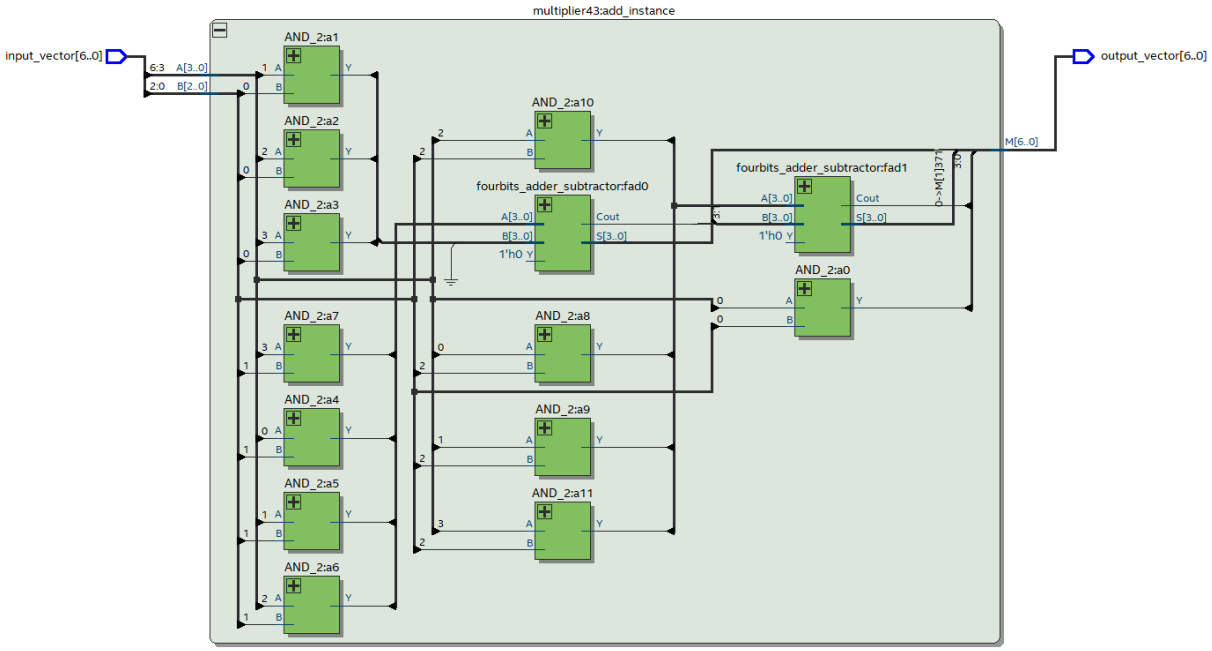
```
library ieee;
use ieee.std_logic_1164.all;
library work;
use work.Gates.all;

entity multiplier43 is
port( A :in std_logic_vector(3 downto 0);B :in std_logic_vector(2 downto 0);
M :out std_logic_vector(6 downto 0));
end entity multiplier43;

architecture structure of multiplier43 is
signal S,X,Z : std_logic_vector(3 downto 0);
 signal Y : std_logic_vector(2 downto 0);
begin
a0: AND_2
port map(A=>A(0),B=>B(0),Y=>M(0));
a1: AND_2
port map(A=>A(1),B=>B(0),Y=>Y(0));
a2: AND_2
port map(A=>A(2),B=>B(0),Y=>Y(1));
a3: AND_2
port map(A=>A(3),B=>B(0),Y=>Y(2));
a4: AND_2
port map(A=>A(0),B=>B(1),Y=>X(0));
a5: AND_2
port map(A=>A(1),B=>B(1),Y=>X(1));
a6: AND_2
port map(A=>A(2),B=>B(1),Y=>X(2));
a7: AND_2
port map(A=>A(3),B=>B(1),Y=>X(3));
fad0: fourbits_adder_subtractor
port map(A(3 downto 0)=>X( 3 downto 0),B(3)=>'0',B(2 downto 0)=>Y(2 downto
0),Y=>'0',
       Cout=>S(3),S( 3 downto 1)=>S(2 downto 0),S(0)=>M(1));
```

```
a8: AND_2
port map(A=>A(0),B=>B(2),Y=>Z(0));
a9: AND_2
port map(A=>A(1),B=>B(2),Y=>Z(1));
a10: AND_2
port map(A=>A(2),B=>B(2),Y=>Z(2));
a11: AND_2
port map(A=>A(3),B=>B(2),Y=>Z(3));
fad1: fourbits_adder_subtractor
port map(A(3 downto 0)=>Z(3 downto 0),B(3 downto 0)=>S(3 downto 0),Y=>'0',
      Cout=>M(6),S(3 downto 0)=>M(5 downto 2));
                  end structure;
```

# RTL View :



## DUT Input/Output Format :

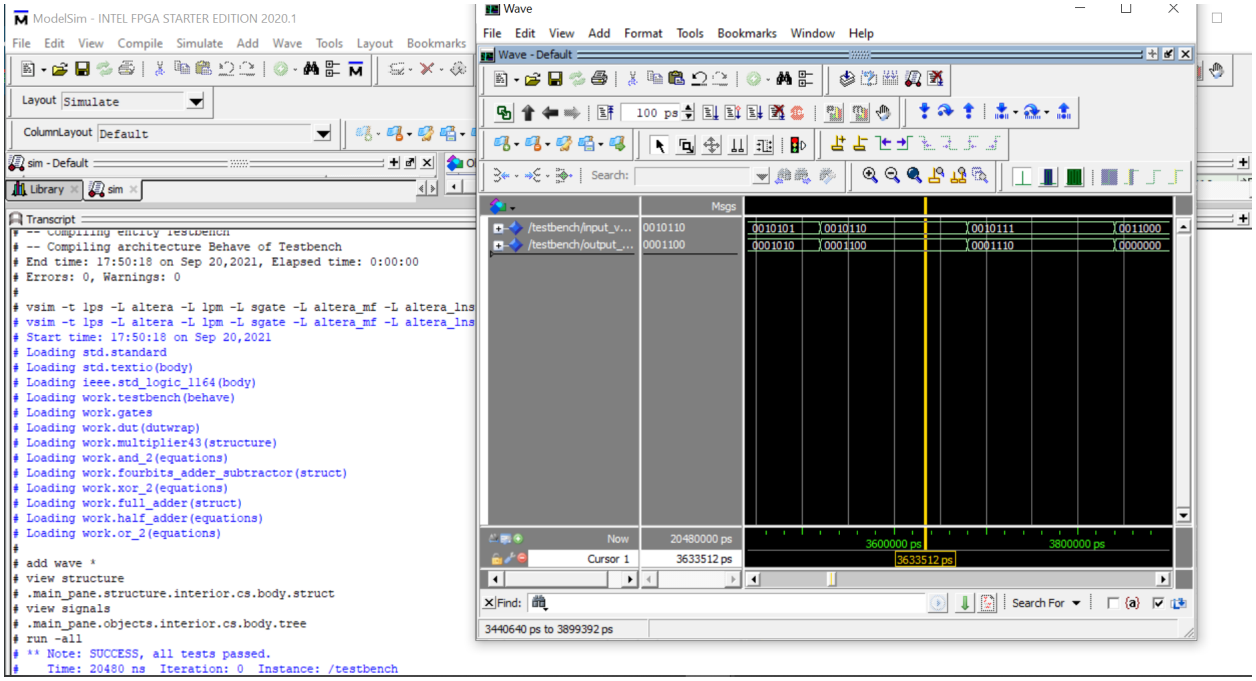
Input :

A(3)=> input_vector(6),
A(2)=> input_vector(5),
A(1) => input_vector(4),
A(0)=> input_vector(3),
B(2)=> input_vector(2),
B(1)=> input_vector(1),
B(0)=> input_vector(0).

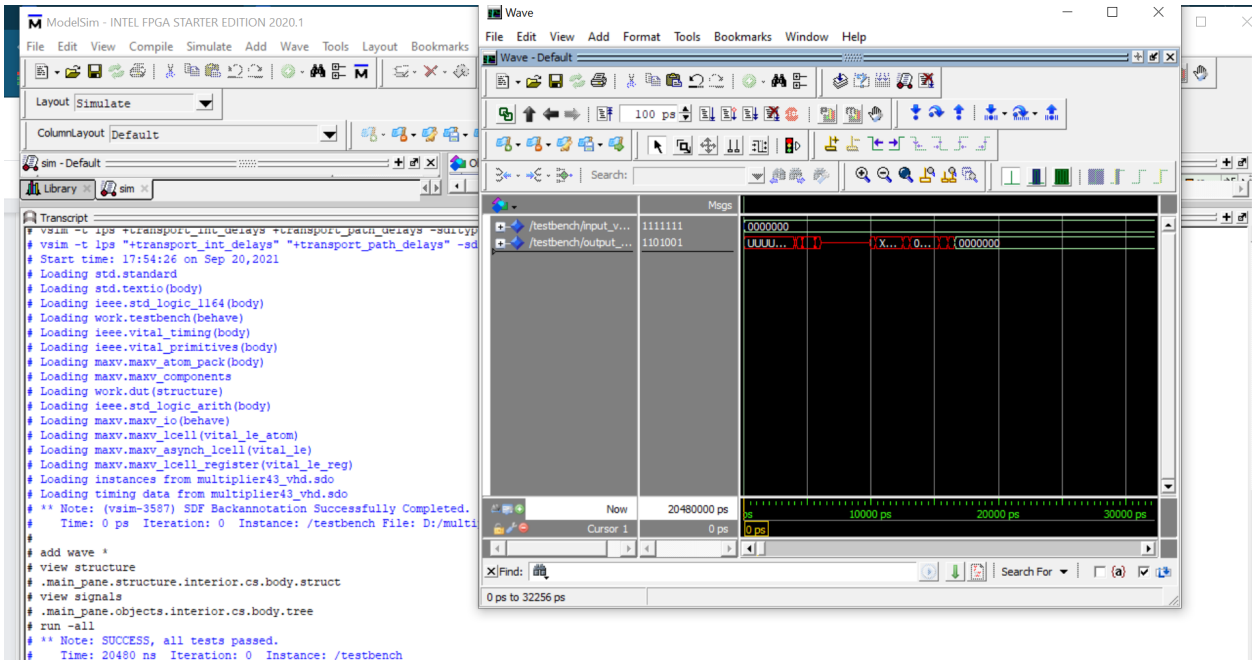
Output :

M(6)=>output_vector(6),
M(5)=>output_vector(5),
M(4)=>output_vector(4),
M(3)=>output_vector(3),
M(2)=>output_vector(2),
M(1)=>output_vector(1),
M(0)=>output_vector(0)).

# RTL Simulation :



# Gate-Level Simulation :



## Krypton Board :

After doing Scan Chain we get the following outputs in the out.txt file :

```
0000000 0000000 Success
0000001 0000000 Success
0000010 0000000 Success
0000011 0000000 Success
0000100 0000000 Success
0000101 0000000 Success
0000110 0000000 Success
0000111 0000000 Success
0001000 0000000 Success
0001001 0000001 Success
0001010 0000010 Success
0001011 0000011 Success
0001100 0000100 Success
0001101 0000101 Success
0001110 0000110 Success
0001111 0000111 Success
0010000 0000000 Success
0010001 0000010 Success
0010010 0000100 Success
0010011 0000110 Success
0010100 0001000 Success
0010101 0001010 Success
0010110 0001100 Success
0010111 0001110 Success
0011000 0000000 Success
0011001 0000011 Success
0011010 0000110 Success
0011011 0001001 Success
0011100 0001100 Success
0011101 0001111 Success
0011110 0010010 Success
0011111 0010101 Success
0100000 0000000 Success
0100001 0000100 Success
0100010 0001000 Success
0100011 0001100 Success
0100100 0010000 Success
0100101 0010100 Success
0100110 0011000 Success
0100111 0011100 Success
0101000 0000000 Success
0101001 0000101 Success
0101010 0001010 Success
0101011 0001111 Success
0101100 0010100 Success
0101101 0011001 Success
0101110 0011110 Success
0101111 0100011 Success
0110000 0000000 Success
0110001 0000110 Success
0110010 0001100 Success
0110011 0010010 Success
0110100 0011000 Success
0110101 0011110 Success
0110110 0100100 Success
0110111 0101010 Success
0111000 0000000 Success
0111001 0000111 Success
0111010 0001110 Success
0111011 0010101 Success
0111100 0011100 Success
0111101 0100011 Success
0111110 0101010 Success
0111111 0110001 Success
1000000 0000000 Success
1000001 0001000 Success
1000010 0010000 Success
1000011 0011000 Success
1000100 0100000 Success
1000101 0101000 Success
1000110 0110000 Success
1000111 0111000 Success
1001000 0000000 Success
```

1001001 0001001 Success
1001010 0010010 Success
1001011 0011011 Success
1001100 0100100 Success
1001101 0101101 Success
1001110 0110110 Success
1001111 0111111 Success
1010000 0000000 Success
1010001 0001010 Success
1010010 0010100 Success
1010011 0011110 Success
1010100 0101000 Success
1010101 0110010 Success
1010110 0111100 Success
1010111 1000110 Success
1011000 0000000 Success
1011001 0001011 Success
1011010 0010110 Success
1011011 0100001 Success
1011100 0101100 Success
1011101 0110111 Success
1011110 1000010 Success
1011111 1001101 Success
1100000 0000000 Success
1100001 0001100 Success
1100010 0011000 Success
1100011 0100100 Success
1100100 0110000 Success
1100101 0111100 Success
1100110 1001000 Success
1100111 1010100 Success
1101000 0000000 Success
1101001 0001101 Success
1101010 0011010 Success
1101011 0100111 Success
1101100 0110100 Success
1101101 1000001 Success
1101110 1001110 Success
1101111 1011011 Success
1110000 0000000 Success
1110001 0001110 Success
1110010 0011100 Success
1110011 0101010 Success
1110100 0111000 Success
1110101 1000110 Success
1110110 1010100 Success
1110111 1100010 Success
1111000 0000000 Success
1111001 0001111 Success
1111010 0011110 Success
1111011 0101101 Success
1111100 0111100 Success
1111101 1001011 Success
1111110 1011010 Success
1111111 1101001 Success