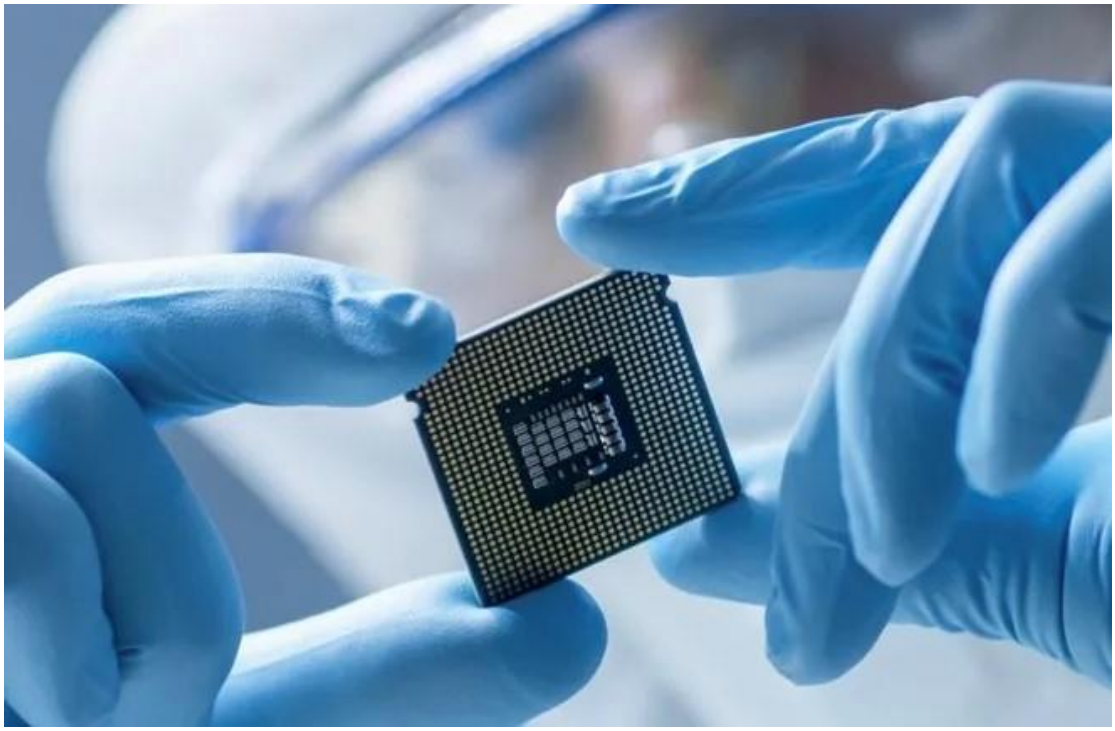


數位系統設計作業

HW6

學號 : 012257027 | 姓名 : 林承羿

第一題

程式碼

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity FA_nbit is
    generic(number:integer range 1 to 32 :=4);              -- 接受輸入 1~32 整數的數值當作 bits 數
    port(
        A:in std_logic_vector(number-1 downto 0);           -- 實現加法，套用輸入 bits 數，range 含 0，故 -1
        B:in std_logic_vector(number-1 downto 0);
        cin:in std_logic;                                   -- 考慮重複利用，接受外部進位
        Q:out std_logic_vector(number-1 downto 0);          -- 加法結果
        cout:out std_logic                                  -- 考慮重複利用，輸出內部進位
    );
end FA_nbit;

architecture Behavioral of FA_nbit is
    signal tmp:std_logic_vector(number downto 0);           -- MSB 為內部進位，故延伸 1 bit
begin
    tmp <= ('0' & A) + B + cin;                             -- 內部兩個來源+外部進位為所有輸出
    Q <= tmp(number-1 downto 0);                            -- Q 不應該包含進位
    cout <= tmp(number);                                    -- 輸出進位
end Behavioral;
```

乘法結果中，兩列加法

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity mul_nn is
    generic(number:integer range 1 to 32 :=4);
    port(
        X:in std_logic_vector(number-1 downto 0);
        Y:in std_logic_vector(number-1 downto 0);
        Ans:out std_logic_vector(2*number-1 downto 0):=(others => '0')
    );
end mul_nn;

architecture Behavioral of mul_nn is
    type MulEthRow is array(0 to number-1) of std_logic_vector(number-1 downto 0);
    type AdTmp is array(0 to number-2) of std_logic_vector(number downto 0);
    signal mulEachRow : MulEthRow := (others => (others => '0'));
    signal addTmp : AdTmp := (others => (others => '0'));
    signal tmp:std_logic_vector(number-1 downto 0);
    component FA_nbit is
        generic(number:integer range 1 to 32);
        port(
            A:in std_logic_vector(number-1 downto 0);
            B:in std_logic_vector(number-1 downto 0);
            cin:in std_logic;
            Q:out std_logic_vector(number-1 downto 0);
            cout:out std_logic
        );
    end component;
begin
    process(X, Y)
    begin
        for i in 0 to (number-1) loop
            for j in 0 to (number-1) loop
                mulEachRow(i)(j) <= X(j) and Y(i);
            end loop;
        end loop;
    end process;

    LOP:for i in 0 to number-2 generate
    FIRST:  if i = 0 generate
                tmp <= ('0' & mulEachRow(i)(number-1 downto 1));
                FA:FA_nbit generic map(number) port map(tmp, mulEachRow(i+1), '0', addTmp(i)(number-1 downto 0), addTmp(i)(number));
            end generate FIRST;
    MID:    if (i > 0) and (i < number-1) generate
                FA0:FA_nbit generic map(number) port map(addTmp(i-1)(number downto 1), mulEachRow(i+1), '0', addTmp(i)(number-1 downto 0), addTmp(i)(number));
            end generate MID;
    LAST:   if i = number-2 generate
                FA1:FA_nbit generic map(number) port map(addTmp(i-1)(number downto 1), mulEachRow(i+1), '0', Ans(2*number-2 downto number-1), Ans(2*number-1));
            end generate LAST;
    end generate LOP;

    Ans(0) <= mulEachRow(0)(0);

    process(addTmp)
    begin
        for i in 1 to (number-2) loop
            Ans(i) <= addTmp(i-1)(0);
        end loop;
    end process;

end Behavioral;
```

可接受輸入 2^(1~32) 的乘法器

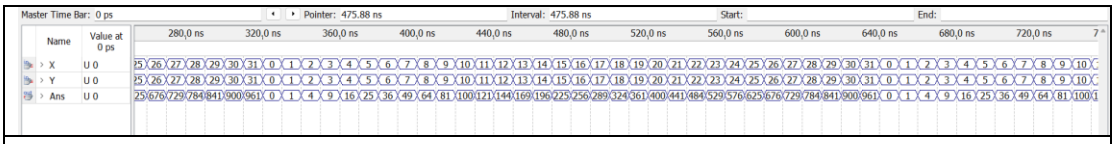
```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity multiple31bits is
    port(
        X:in std_logic_vector(4 downto 0):=(others => '0');          -- 接 component 接口
        Y:in std_logic_vector(4 downto 0):=(others => '0');
        Ans:out std_logic_vector(9 downto 0):=(others => '0')
    );
end multiple31bits;

architecture Behavioral of multiple31bits is
    component mul_nn is
        generic(number:integer range 1 to 32);
        port(
            X:in std_logic_vector(number-1 downto 0);
            Y:in std_logic_vector(number-1 downto 0);
            Ans:out std_logic_vector(2*number-1 downto 0)
        );
    end component;
begin
    fun0:mul_nn generic map(5) port map(X, Y, Ans);                  -- 寫此 code 的目的，令 generic number 為 5，符合題意
end Behavioral;
```

包裝乘法器，並指定 generic number 為 5 ，為 0~31 bits

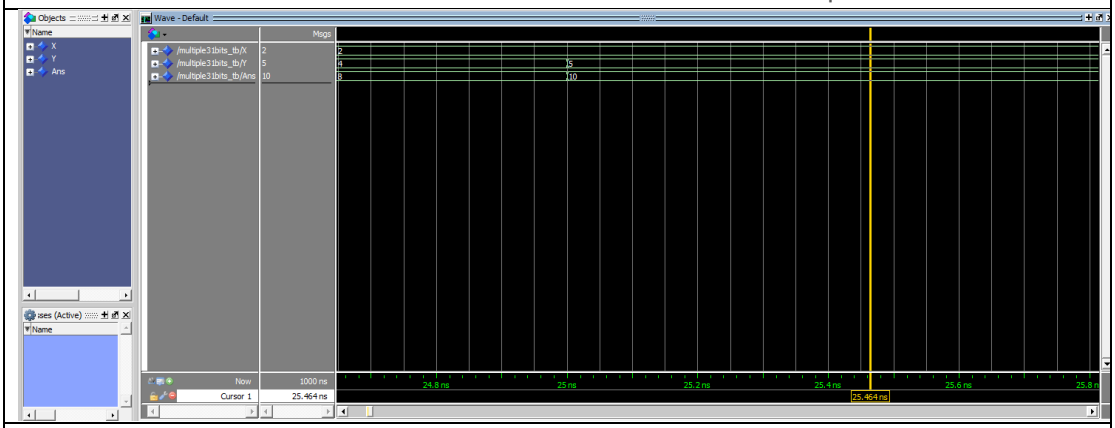
```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity multiple31bits_tb is
end multiple31bits_tb;

architecture Behavioral of multiple31bits_tb is
    component multiple31bits is
        port(
            X:in std_logic_vector(4 downto 0);
            Y:in std_logic_vector(4 downto 0);
            Ans:out std_logic_vector(9 downto 0)
        );
    end component;
    signal X, Y:std_logic_vector(4 downto 0):=(others => '0');
    signal Ans:std_logic_vector(9 downto 0):=(others => '0');
begin
    fun0:multiple31bits port map(X, Y, Ans);
    process
    begin
        for i in 0 to 31 loop
            X <= conv_std_logic_vector(i, 5);                        -- i 為 0~31 整數，轉成 std_logic_vector
            wait for 10 ns;                                          -- 等待 10 ns，轉換下一個狀態
        end loop;
    end process;

    process
    begin
        for i in 0 to 31 loop
            Y <= conv_std_logic_vector(i, 5);
            wait for 5 ns;                                           -- 等待 5 ns，轉換下一個狀態，錯開以看到更多結果
        end loop;
    end process;
end Behavioral;
```

Testbench，並等待不同時間期待看到更多結果

波形圖



從圖中可看到不管是在頭、尾銜接皆符合期待，即 31 回到 0。運作過程中也各做了 n^2，數字以確認每項正確，可參照本人寫 C++檔案結果 output.txt



以 (2*4 = 8) 及 (2*5 = 10) 做舉例，表示出正確運作。

加分題

程式碼

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_ARITH.ALL;
4  use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6  entity FA_nbit is
7      generic(number:integer range 1 to 32 :=4);              -- 接受輸入 1~32 整數的數值當作 bits 數
8      port(
9          A:in std_logic_vector(number-1 downto 0);           -- 實現加法，套用輸入 bits 數，range 含 0，故 -1
10         B:in std_logic_vector(number-1 downto 0);
11         cin:in std_logic;                                   -- 考慮重複利用，接受外部進位
12         Q:out std_logic_vector(number-1 downto 0);          -- 加法結果
13         cout:out std_logic                                  -- 考慮重複利用，輸出內部進位
14     );
15 end FA_nbit;
16
17 architecture Behavioral of FA_nbit is
18     signal tmp:std_logic_vector(number downto 0);           -- MSB 為內部進位，故延伸 1 bit
19 begin
20     tmp <= ('0' & A) + B + cin;                             -- 內部兩個來源+外部進位為所有輸出
21     Q <= tmp(number-1 downto 0);                            -- Q 不應該包含進位
22     cout <= tmp(number);                                    -- 輸出進位
23 end Behavioral;
```

乘法結果中，兩列加法

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity mul_nn is
    generic(number:integer range 1 to 32 :=4);
    port(
        X:in std_logic_vector(number-1 downto 0);                                   -- n bis 數字 range = ((n-1)~0)
        Y:in std_logic_vector(number-1 downto 0);
        Ans:out std_logic_vector(2*number-1 downto 0):=(others => '0')              -- n bits 乘法結果最多 2n bits, range = ((2n-1)~0)
    );
end mul_nn;

architecture Behavioral of mul_nn is
    type MulEhRow is array(0 to number-1) of std_logic_vector(number-1 downto 0);  -- 假設 X 在上 Y 在下，Y 每個 bit 與 X 相乘的結果存放於此，n bits * n bits 最多 n 列
    type AdTmp is array(0 to number-2) of std_logic_vector(number downto 0);       -- 將乘完結果相加，需做 (n-1) 次
    signal mulEachRow : MulEhRow := (others => (others => '0'));
    signal addTmp : AdTmp := (others => (others => '0'));
    signal tmp:std_logic_vector(number-1 downto 0);                                 -- 為了 modelsim 正常運作，因為不接受參數進行額外運算，故使用暫存方式將結果當作固定值丟入function
    component FA_nbit is
        generic(number:integer range 1 to 32);
        port(
            A:in std_logic_vector(number-1 downto 0);
            B:in std_logic_vector(number-1 downto 0);
            cin:in std_logic;
            Q:out std_logic_vector(number-1 downto 0);
            cout:out std_logic
        );
    end component;
begin
    process(X, Y)                                                                   -- X 與 Y 相乘，故偵測兩者
    begin
        for i in 0 to (number-1) loop
            for j in 0 to (number-1) loop
                mulEachRow(i)(j) <= X(j) and Y(i);                                  -- 固定 i 取 j 每項乘法結果 (假設 X 在上 Y 在下)
            end loop;
        end loop;
    end process;

    LOP:for i in 0 to number-2 generate
    FIRST:  if i = 0 generate
                tmp <= ('0' & mulEachRow(i)(number-1 downto 1));
                FA:FA_nbit generic map(number) port map(tmp, mulEachRow(i+1), '0', addTmp(i)(number-1 downto 0), addTmp(i)(number));  -- 當第一次相加，根據乘法，第一列MSB比第二列少 1 bit，考慮對齊，補0，並取相同 bits，LSB 應直接為答案 LSB
            end generate FIRST;
    MID:    if (i > 0) and (i < number-1) generate
                FA0:FA_nbit generic map(number) port map(addTmp(i-1)(number downto 1), mulEachRow(i+1), '0', addTmp(i)(number-1 downto 0), addTmp(i)(number));  -- 第一個來源已考慮進位，故外部進位0
            end generate MID;
    LAST:   if i = number-2 generate
                FA1:FA_nbit generic map(number) port map(addTmp(i-1)(number downto 1), mulEachRow(i+1), '0', Ans(2*number-2 downto number-1), Ans(2*number-1));  -- 最後一次相加結果應為答案相對應位置的 bits
            end generate LAST;
    end generate LOP;

    Ans(0) <= mulEachRow(0)(0);                                                     -- 第一次相乘結果，LSB 即為答案 LSB

    process(addTmp)
    begin
        for i in 1 to (number-2) loop                                               -- 相對位置剩下((number-2)~1)沒填
            Ans(i) <= addTmp(i-1)(0);                                               -- 將每列相乘結果 LSB 填入相對位置
        end loop;
    end process;

end Behavioral;
```

可接受輸入 2^(1~32) 的乘法器

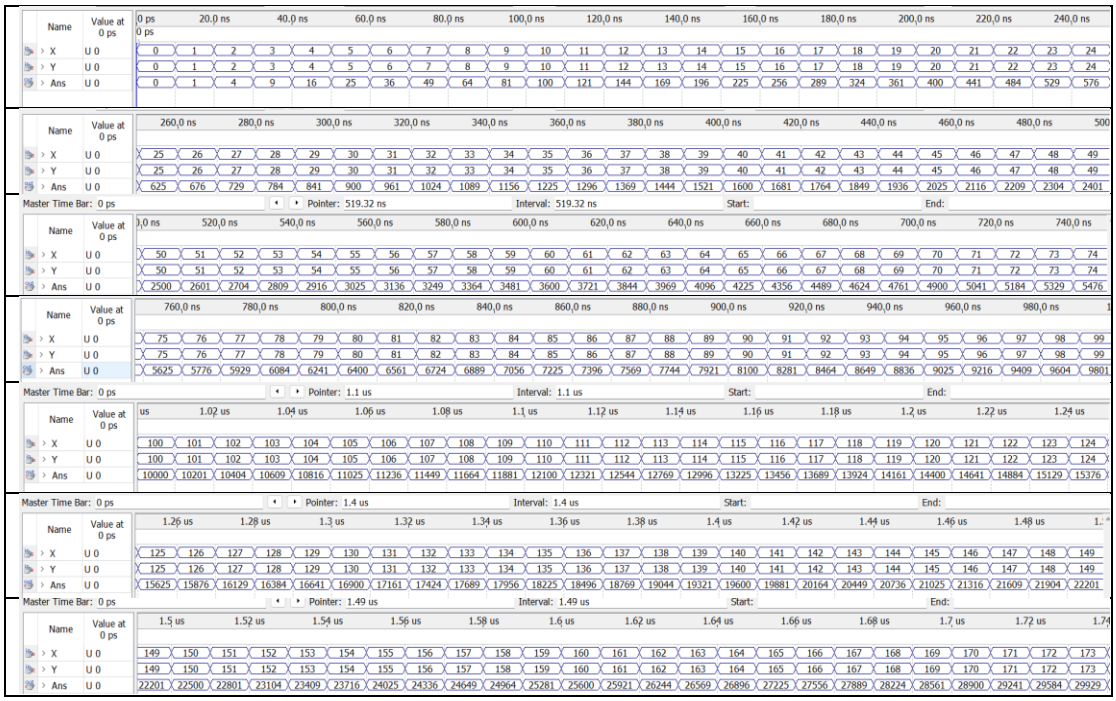
```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity multiple255bits is
    port(
        X:in std_logic_vector(7 downto 0);          -- 接 component 接口
        Y:in std_logic_vector(7 downto 0);
        Ans:out std_logic_vector(15 downto 0)
    );
end multiple255bits;

architecture Behavioral of multiple255bits is
    component mul_nn is
        generic(number:integer range 1 to 32);
        port(
            X:in std_logic_vector(number-1 downto 0);
            Y:in std_logic_vector(number-1 downto 0);
            Ans:out std_logic_vector(2*number-1 downto 0)
        );
    end component;
begin
    fun0:mul_nn generic map(8) port map(X, Y, Ans);     -- 寫此 code 的目的，令 generic number 為 8，符合題意(2^8 = 256, 為0~255)
end Behavioral;
```

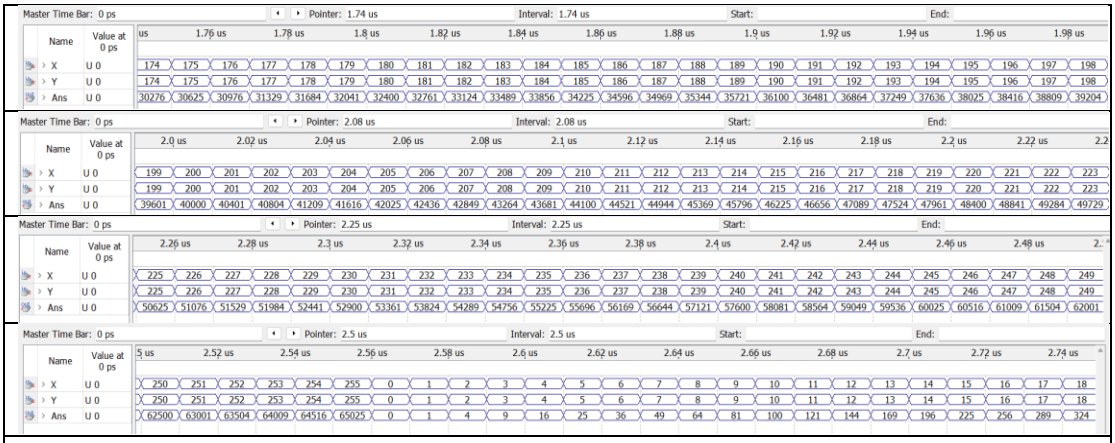
包裝乘法器，並指定 generic number 為 8 ，為 0~255 bits

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity multiple255bits_tb is
end multiple255bits_tb;

architecture Behavioral of multiple255bits_tb is
    component multiple255bits is
        port(
            X:in std_logic_vector(7 downto 0);
            Y:in std_logic_vector(7 downto 0);
            Ans:out std_logic_vector(15 downto 0)
        );
    end component;
    signal X, Y:std_logic_vector(7 downto 0):=(others => '0');
    signal Ans:std_logic_vector(15 downto 0):=(others => '0');
begin
    fun0:multiple255bits port map(X, Y, Ans);
    process
    begin
        for i in 0 to 255 loop                          -- i 為 0~255 整數，轉成 std_logic_vector
            X <= conv_std_logic_vector(i, 8);
            wait for 10 ns;                             -- 等待 10 ns，轉換下一個狀態
        end loop;
    end process;

    process
    begin
        for i in 0 to 255 loop
            Y <= conv_std_logic_vector(i, 8);
            wait for 5 ns;                              -- 等待 5 ns，轉換下一個狀態
        end loop;
    end process;
end Behavioral;
```

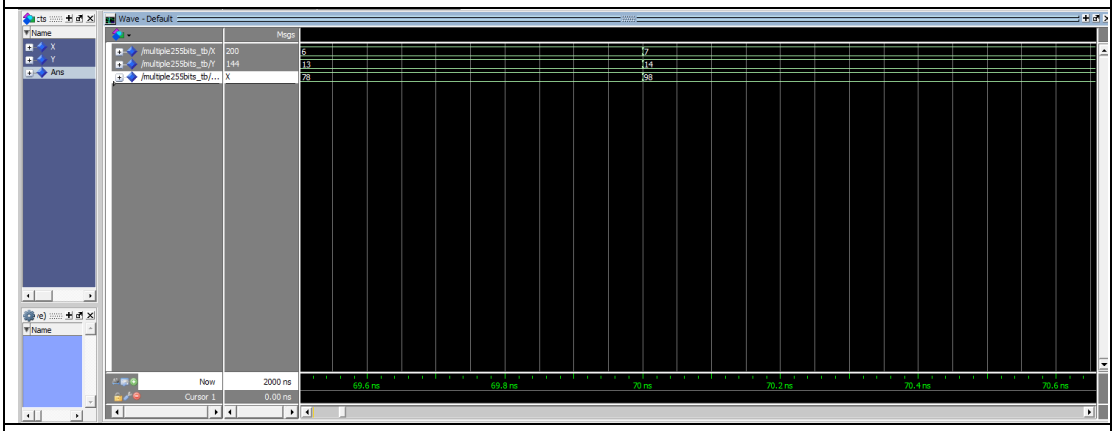
Testbench，並等待不同時間期待看到更多結果

波形圖





以上為 0~255 做平方的結果，各自結果可參考 output.txt。且從最後一張圖可清楚表示出 255 下一個因溢位到 0 的正確循環。

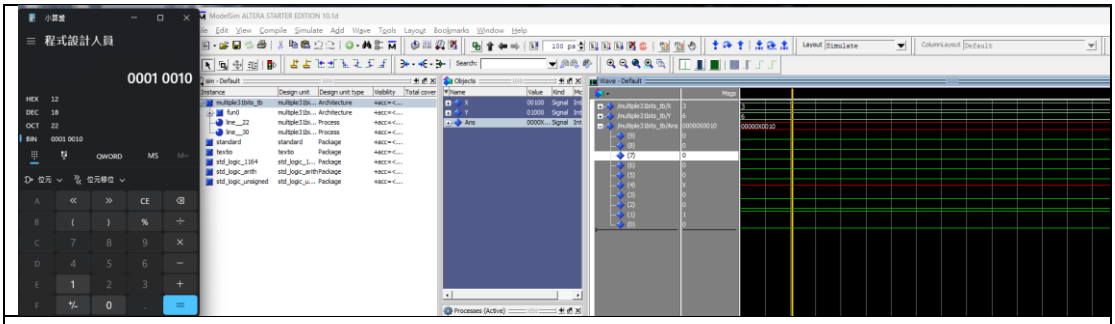


以 (6*13 = 78) 及 (7*14 = 98) 做舉例，表示出正確運作。

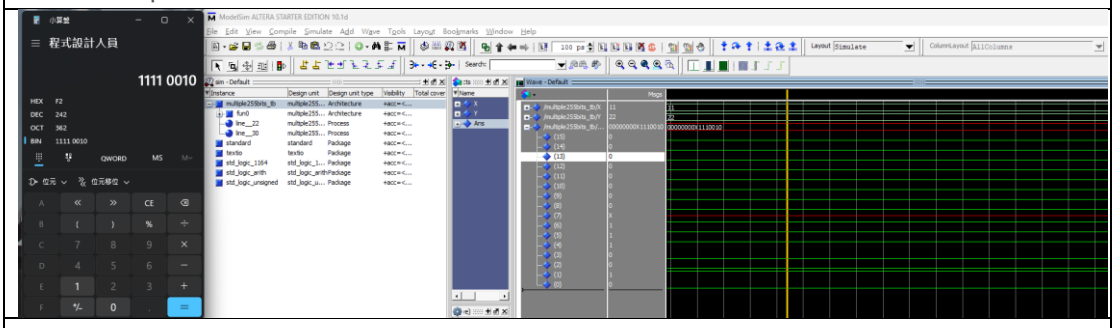
心得

從此次實驗我清楚知道乘法器該如何運作。細節包括如何做每列加法，並非單純加法，需要考慮移位。並在每列加完後如何決定答案，除了最後一列直接決定答案的相對位置，其他則靠著 LSB 不會有人與其相加的特性直接對到達案的相對位置。

此次實驗也是第一次使用 vhdl 的 modelsim，也是撞到了坑，包括 entity mul_nn.vhd 中 FIRST 的 port map 為何參數需要做暫存而非 ('0' & 數字)，我了解到因為 port map 本身不引許，經由觀察 Error 發現要求參數只能以 constant 方式傳入。



此為說明實做發生的例外，看了 code，我嘗試尋找發生衝突，即兩個信號同時輸入造成不知接受哪個而發生信號為 X 的情況，但沒發現。且考慮過是否因為沒初始化而造成信號 X，因此 code 都改成了有初始化的，但依然沒有改變。最後以計算機圖式表示信號 X 的正確答案皆為 1，經過好幾組測試，我發現皆是如此，以 (3*6 = 18) 舉例在 multiple31bits_tb 發生的例外。



此為說明實做發生的例外，看了 code，我嘗試尋找發生衝突，即兩個信號同時輸入造成不知接受哪個而發生信號為 X 的情況，但沒發現。且考慮過是否因為沒初始化而造成信號 X，因此 code 都改成了有初始化的，但依然沒有改變。最後以計算機圖式表示信號 X 的正確答案皆為 1，經過好幾組測試，我發現皆是如此，以 (11*22 = 242) 舉例在 multiple255bits_tb 發生的例外。