

# 數位系統設計作業

## HW6

學號: 012257027 | 姓名: 林承羿

# 第一題

## 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);
8     port(
9         A:in std_logic_vector(number-1 downto 0);
10        B:in std_logic_vector(number-1 downto 0);
11        cin:in std_logic;
12        Q:out std_logic_vector(number-1 downto 0);
13        cout:out std_logic
14    );
15 end FA_nbit;
16
17 architecture Behavioral of FA_nbit is
18     signal tmp:std_logic_vector(number downto 0);
19 begin
20     tmp <= ('0' & A) + B + cin;
21     Q <= tmp(number-1 downto 0);
22     cout <= tmp(number);
23 end Behavioral;
```

-- 接受輸入 1~32 整數的數值當作 bits 數

-- 實現加法，套用輸入 bits 數，range 含 0，故 -1

-- 考慮重複利用，接受外部進位

-- 加法結果

-- 考慮重複利用，輸出內部進位

-- MSB 為內部進位，故延伸 1 bit

-- 內部兩個來源+外部進位為所有輸出

-- Q 不應該包含進位

-- 輸出進位

## 乘法結果中，兩列加法

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

entity mul_n 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);
    );
end mul_n;

architecture Behavioral of mul_n is
    type mulRow is array(0 to number-1) of std_logic_vector(number-1 downto 0);
    type addTmp is array(0 to number-2) of std_logic_vector(number-1 downto 0);
    signal mulEachRow : mulRow := (others => (others => '0'));
    signal addTmp : addTmp := (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;

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

    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;
```

-- n bits 數字 range = ((n-1)-0)

-- n bits 乘法結果最多 2n bits, range = ((2n-1)-0)

-- 假設 X 在上 Y 在下，Y 每個 bit 與 X 相乘的結果存於此，n bits \* n bits 最多 n 列

-- 將每列結果相加，需加 (n-1) 次

-- 為了 modelsim 正常運作，因為不滿足參數後行額外確認，故使用暫存方式將結果當作迴圈值去 function

-- X 與 Y 相乘，形成兩列

-- 假設 i 取 j 每列乘法結果 (假設 X 在上 Y 在下)

-- 當第一次相加，根據乘法，第一列的比第二列少 1 bit，考慮對齊，tmp 與 addTmp 應為相同 bits，LSB 應置為最高，LSB

-- 第一個來源已考慮進位，與外部進位 0

-- 最後一次相加結果應為最高相對應位置的 bits

-- 第一次相加結果，LSB 應為最高，LSB

-- 相對位置剩下 ((number-2)-1) 沒填

-- 將每列相加結果，LSB 填入相對位置

## 可接受輸入 2^(1~32) 的乘法器

```

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 multiple31bits is
7      port(
8          X:in std_logic_vector(4 downto 0):=(others => '0'); -- 接 component 接口
9          Y:in std_logic_vector(4 downto 0):=(others => '0');
10         Ans:out std_logic_vector(9 downto 0):=(others => '0')
11     );
12 end multiple31bits;
13
14 architecture Behavioral of multiple31bits is
15     component mul_nn is
16         generic(number:integer range 1 to 32);
17         port(
18             X:in std_logic_vector(number-1 downto 0);
19             Y:in std_logic_vector(number-1 downto 0);
20             Ans:out std_logic_vector(2*number-1 downto 0)
21         );
22     end component;
23 begin
24     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
25 end Behavioral;

```

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

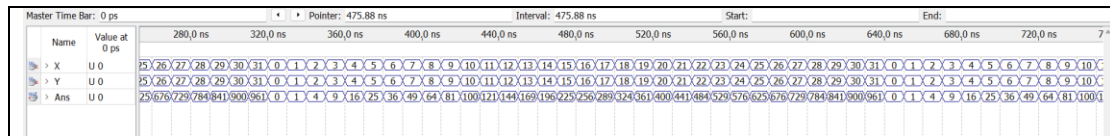
```

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 multiple31bits_tb is
7      end multiple31bits_tb;
8
9  architecture Behavioral of multiple31bits_tb is
10     component multiple31bits is
11         port(
12             X:in std_logic_vector(4 downto 0);
13             Y:in std_logic_vector(4 downto 0);
14             Ans:out std_logic_vector(9 downto 0)
15         );
16     end component;
17     signal X, Y:std_logic_vector(4 downto 0):=(others => '0');
18     signal Ans:std_logic_vector(9 downto 0):=(others => '0');
19 begin
20     fun0:multiple31bits port map(X, Y, Ans);
21     process
22     begin
23         for i in 0 to 31 loop
24             X <= conv_std_logic_vector(i, 5); -- i 為 0~31 整數，轉成 std_logic_vector
25             wait for 10 ns; -- 等待 10 ns，轉換下一個狀態
26         end loop;
27     end process;
28
29     process
30     begin
31         for i in 0 to 31 loop
32             Y <= conv_std_logic_vector(i, 5);
33             wait for 5 ns; -- 等待 5 ns，轉換下一個狀態，錯開以看到更多結果
34         end loop;
35     end process;
36 end Behavioral;

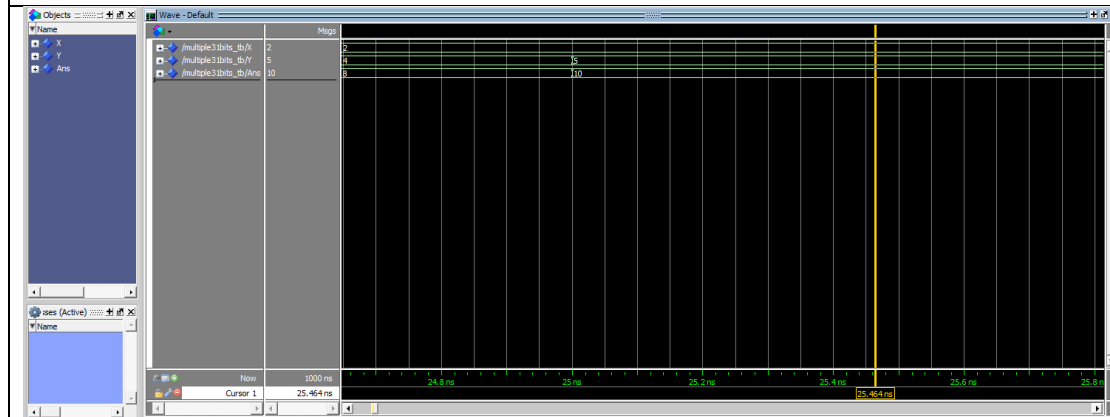
```

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);
8      port(
9          A:in std_logic_vector(number-1 downto 0);
10         B:in std_logic_vector(number-1 downto 0);
11         cin:in std_logic;
12         Q:out std_logic_vector(number-1 downto 0);
13         cout:out std_logic
14     );
15 end FA_nbit;
16
17 architecture Behavioral of FA_nbit is
18     signal tmp:std_logic_vector(number downto 0);
19 begin
20     tmp <= ('0' & A) + B + cin;
21     Q <= tmp(number-1 downto 0);
22     cout <= tmp(number);
23 end Behavioral;
```

乘法結果中，兩列加法

```

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

entity mul_n 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));
end mul_n;

architecture Behavioral of mul_n is
    type MultRow is array(0 to number-1) of std_logic_vector(number downto 0);
    type AddTmp is array(0 to number-2) of std_logic_vector(number downto 0);
    signal multRow: MultRow := (others => (others => '0'));
    signal addTmp: AddTmp := (others => (others => '0'));
    signal tmp:std_logic_vector(number-1 downto 0);
    component FA_full 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
                multRow(i)(j) <= X(j) and Y(i);
            end loop;
        end loop;
    end process;

    L0:for i in 0 to number-2 generate
        FIRST: if i = 0 generate
            tmp <= ('0' & multRow(0)(number-1 downto 1));
            FA0:full generic map(number) port map(tmp, multRow(0)(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:full generic map(number) port map(addTmp(i-1)(number downto 1), multRow(i)(i+1), '0', addTmp(i)(number-1 downto 0), addTmp(i)(number));
            end generate MID;
        LAST: if i = number-2 generate
            FA0:full generic map(number) port map(addTmp(i-1)(number downto 1), multRow(i)(i+1), '0', Ans(2*number-2 downto number-1), Ans(2*number-1));
            end generate LAST;
    end generate L0;

    Ans(0) <= multRow(0)(0);
end Behavioral;

```

-- n bits 數字 range = ((n-1)-0)  
-- n bits 乘法結果最多 2n bits, range = ((2n-1)-0)  
-- 假設 X 在 i 下, Y 在 j 下, 每個 bit 與 X 相乘的結果存於此, n bits \* n bits 最多 n 列  
-- 將乘法結果相加, 需要 (n-1) 次  
-- 為了 modelsim 正確運作, 因為下標要從最後行開始儲存, 故使用變數方式的乘法結果會作固定值送入 function  
-- X 與 Y 相乘, 故減則兩者  
-- 需要 i 取 j 每條乘法結果 (假設 X 在 i 下, Y 在 j 下)  
-- 當第一次相加, 根據乘法, 第一列的比第二列少 1 bit, 考慮對齊, i 和 0, 並取相同 bits, L0 最高位為答案 L0  
-- 第一個來源已考慮進位, 故外部進位 0  
-- 最後一次相加結果是為答案相對應位置的 bits  
-- 第一次相加結果, L0 最高答案 L0  
-- 相對位置剩下 ((number-2)-1) 沒填  
-- 將每列相加結果 L0 填入相對位置

可接受輸入  $2^{(1\sim32)}$  的乘法器

```

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 multiple255bits is
7     port(
8         X:in std_logic_vector(7 downto 0);
9         Y:in std_logic_vector(7 downto 0);
10        Ans:out std_logic_vector(15 downto 0)
11    );
12 end multiple255bits;
13
14 architecture Behavioral of multiple255bits is
15     component mul_nn is
16         generic(number:integer range 1 to 32);
17         port(
18             X:in std_logic_vector(number-1 downto 0);
19             Y:in std_logic_vector(number-1 downto 0);
20             Ans:out std_logic_vector(2*number-1 downto 0)
21         );
22     end component;
23 begin
24     fun0:mul_nn generic map(8) port map(X, Y, Ans);
25 end Behavioral;

```

-- 接 component 接口  
-- 寫此 code 的目的, 令 generic number 為 8, 符合題意 ( $2^8 = 256$ , 為 0~255)

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



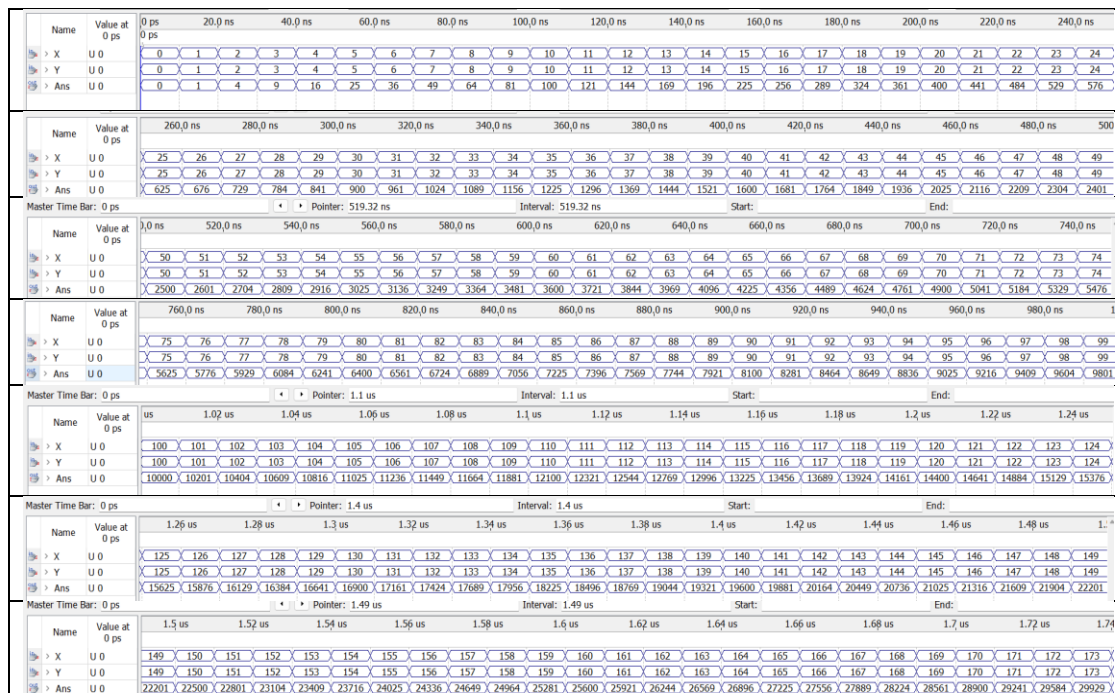
```

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 multiple255bits_tb is
7  end multiple255bits_tb;
8
9  architecture Behavioral of multiple255bits_tb is
10     component multiple255bits is
11     port(
12         X:in std_logic_vector(7 downto 0);
13         Y:in std_logic_vector(7 downto 0);
14         Ans:out std_logic_vector(15 downto 0)
15     );
16     end component;
17     signal X, Y:std_logic_vector(7 downto 0):=(others => '0');
18     signal Ans:std_logic_vector(15 downto 0):=(others => '0');
19 begin
20     fun0:multiple255bits port map(X, Y, Ans);
21     process
22     begin
23         for i in 0 to 255 loop
24             X <= conv_std_logic_vector(i, 8);
25             wait for 10 ns;
26         end loop;
27     end process;
28
29     process
30     begin
31         for i in 0 to 255 loop
32             Y <= conv_std_logic_vector(i, 8);
33             wait for 5 ns;
34         end loop;
35     end process;
36 end Behavioral;

```

Testbench，並等待不同時間期待看到更多結果

## 波形圖



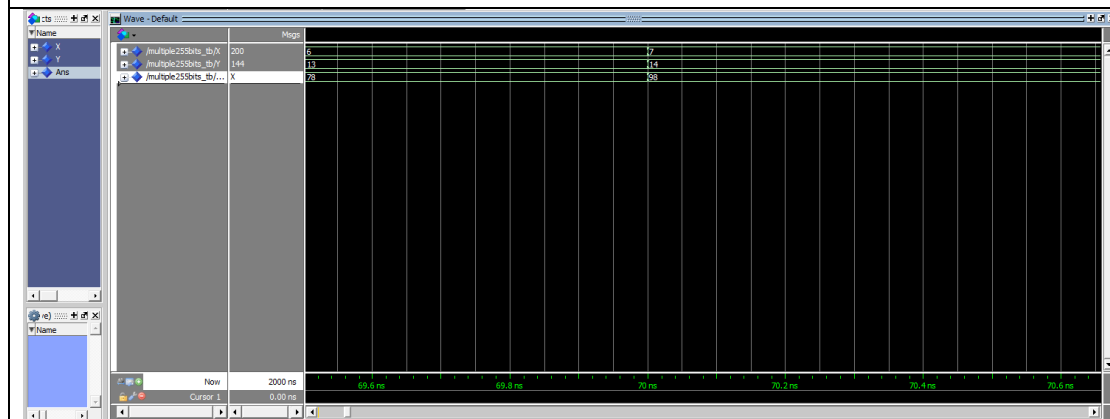
Master Time Bar: 0 ps		• • Pointer: 1.74 us										Interval: 1.74 us										Start:										End:																																																																																																																																																																																																																											
Name	Value at 0 ps	us										1.75 us										1.82 us										1.86 us										1.88 us										1.9 us										1.92 us										1.94 us										1.96 us										1.98 us																																																																																																																																																															
X	U 0	174										175										176										177										178										179										180										181										182										183										184										185										186										187										188										189										190										191										192										193										194										195										196										197										198									
Y	U 0	174										175										176										177										178										179										180										181										182										183										184										185										186										187										188										189										190										191										192										193										194										195										196										197										198									
Ans	U 0	30276										30625										30976										31329										31684										32041										32400										32761										33124										33489										33856										34225										34596										34969										35344										35721										36100										36481										36864										37249										37636										38025										38416										38809										39204									

Master Time Bar: 0 ps		• • Pointer: 2.08 us										Interval: 2.08 us										Start:										End:																																																																																																																																																																																																																											
Name	Value at 0 ps	2.0 us										2.02 us										2.04 us										2.06 us										2.08 us										2.1 us										2.12 us										2.14 us										2.16 us										2.18 us										2.2 us										2.22 us										2.																																																																																																																																	
X	U 0	199										200										201										202										203										204										205										206										207										208										209										210										211										212										213										214										215										216										217										218										219										220										221										222										223									
Y	U 0	199										200										201										202										203										204										205										206										207										208										209										210										211										212										213										214										215										216										217										218										219										220										221										222										223									
Ans	U 0	39601										40000										40401										40804										41209										41616										42025										42436										42849										43264										43681										44100										44521										44944										45369										45796										46225										46656										47089										47524										47961										48400										48841										49284										49729									

Master Time Bar: 0 ps		• • Pointer: 2.25 us										Interval: 2.25 us										Start:										End:																																																																																																																																																																																																																											
Name	Value at 0 ps	2.26 us										2.28 us										2.3 us										2.32 us										2.34 us										2.36 us										2.38 us										2.4 us										2.42 us										2.44 us										2.46 us										2.48 us										2.																																																																																																																																	
X	U 0	225										226										227										228										229										230										231										232										233										234										235										236										237										238										239										240										241										242										243										244										245										246										247										248										249									
Y	U 0	225										226										227										228										229										230										231										232										233										234										235										236										237										238										239										240										241										242										243										244										245										246										247										248										249									
Ans	U 0	50625										51076										51529										51984										52441										52900										53361										53824										54289										54756										55225										55696										56169										56644										57121										57600										58081										58564										59049										59536										60025										60516										61009										61504										62001									

Master Time Bar: 0 ps		• • Pointer: 2.5 us										Interval: 2.5 us										Start:										End:																																																																																																																																																																																																																											
Name	Value at 0 ps	2.5 us										2.52 us										2.54 us										2.56 us										2.58 us										2.6 us										2.62 us										2.64 us										2.66 us										2.68 us										2.7 us										2.72 us										2.74 us										*																																																																																																																							
X	U 0	250										251										252										253										254										255										0										1										2										3										4										5										6										7										8										9										10										11										12										13										14										15										16										17										18									
Y	U 0	250										251										252										253										254										255										0										1										2										3										4										5										6										7										8										9										10										11										12										13										14										15										16										17										18									
Ans	U 0	62500										63001										63504										64009										64516										65025										0										1										4										9										16										25										36										49										64										81										100										121										144										169										196										225										256										289										324									

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。

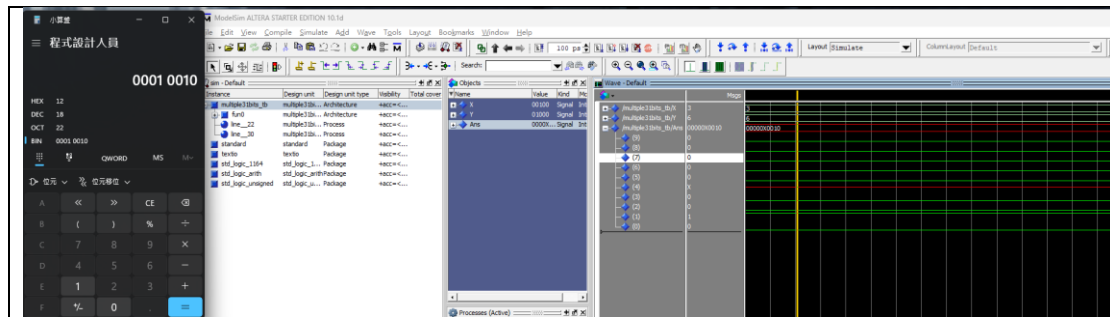


以  $(6 \times 13 = 78)$  及  $(7 \times 14 = 98)$  做舉例，表示出正確運作。

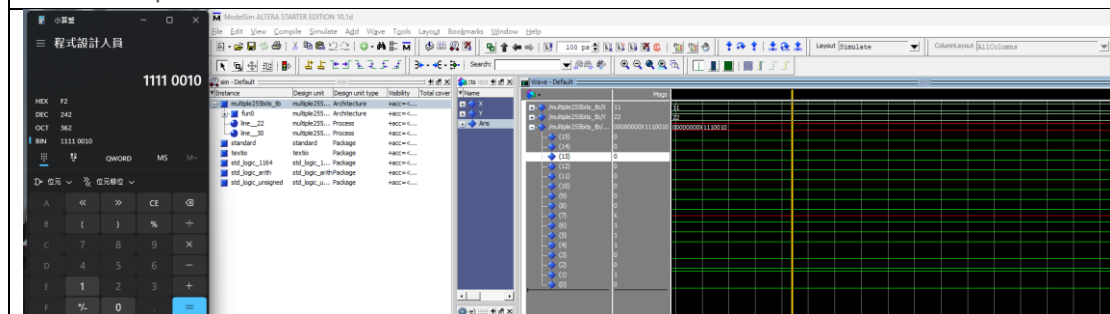
## 心得

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。