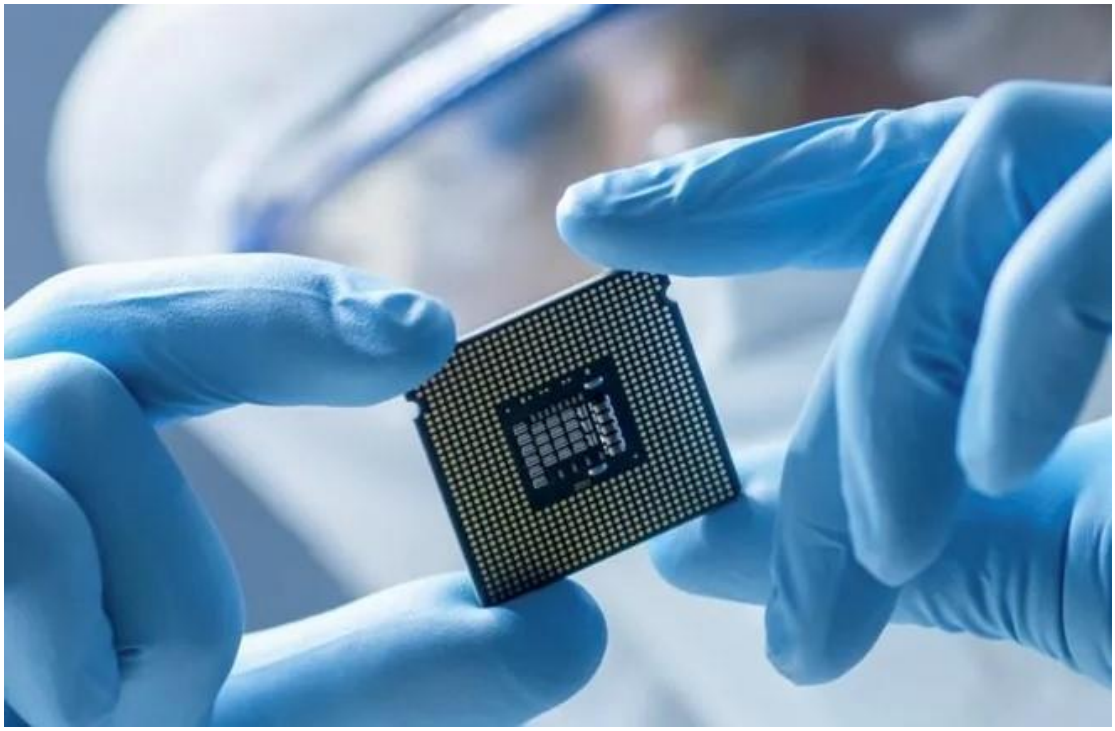


數位系統設計作業

HW7

學號 : 01257027| 姓名 : 林承羿

前置作業：

下列圖片為在執行最後包裝前應該撰寫的各自功能，即指定 sw = 00、01、10、11 該做的事情，以下照片順序也依照功能順序編排。

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_unsigned.all;
4
5  entity add is
6      port(
7          X:in std_logic_vector(3 downto 0);
8          Y:in std_logic_vector(3 downto 0);
9          Q:out std_logic_vector(7 downto 0)
10     );
11 end add;
12
13 architecture add of add is
14 begin
15     Q <= ("0000" & X) + ("0000" & Y);        -- 兩來源相加，補上正確位數的零以符合結果
16 end add;
```

實現當 sw = 00 應該為加法

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity FA_nbit is
    generic(number :integer range 1 to 32 := 3);
    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 FA_nbit;

architecture FA_nbit of FA_nbit is
    signal tmp :std_logic_vector(number downto 0);      -- 比結果多 1bit，存放家完的進位
begin

    tmp <= ('0' & A) + B + cin;                         -- 乘法每列結果相加需考慮上方列補零，對其下方列。(上方列LSB為答案LSB)
    Q <= tmp(number-1 downto 0);
    cout <= tmp(number);                                -- 存入進位


end FA_nbit;
```

實現當 (sw = 01、10、11) or (按下按鈕二) 應該為乘法。

上圖為定義每列結果如何相加。下圖為實現多 bits 乘法電路。

```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 bits 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，故底層用者
    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，第一列的b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
                FAN: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，故外部做位0
            end generate MID;
    LAST:   if i = number-2 generate
                FAI: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                                               -- 相對位置倒下((number-2)~1)次項
            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;
```

```vhdl
library ieee;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_1164.all;


entity digits is
port(
        BIN:in integer range 0 to 9999;
        num3: out integer range 0 to 9;
        num2: out integer range 0 to 9;
        num1: out integer range 0 to 9;
        num0: out integer range 0 to 9
        );

end digits;

architecture digits of digits is
begin

    num3 <= BIN /1000;                      -- 算出千位數
    num2 <= (BIN /100) mod 10;              -- 算出百位數
    num1 <= (BIN /10) mod 10;               -- 算出十位數
    num0 <= BIN mod 10;                     -- 求出個位數


end digits;
```

在做七段顯示器步驟前，需決定每位數應該顯示什麼數字

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity decoder_7seg is
    PORT(
            BCD:in std_logic_vector(3 downto 0);
            HEX:out std_logic_vector(6 downto 0)
            );
end decoder_7seg;

architecture decoder_7seg of decoder_7seg is
begin

    HEX <=  "1000000" when BCD = 0 else             -- 根據七段顯示器的排列(0為暗，1為亮)，顯示出人眼方便識別的數字
            "1111001" when BCD = 1 else
            "0100100" when BCD = 2 else
            "0110000" when BCD = 3 else
            "0011001" when BCD = 4 else
            "0010010" when BCD = 5 else
            "0000010" when BCD = 6 else
            "1111000" when BCD = 7 else
            "0000000" when BCD = 8 else
            "0010000" when BCD = 9 else
            "1111111";                              -- 不會用到，但符合語法要求
end decoder_7seg;
```

接收到數值，轉成規定格式以符合七段顯示器表示的樣子

第一題 (不含 BUTTON2)

程式碼

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity mul_nbit_hex is
    PORT(
            X:in std_logic_vector(3 downto 0);                      -- 資料來源
            Y:in std_logic_vector(3 downto 0);                      -- 資料來源
            sw:in std_logic_vector(1 downto 0);
            bt, bt2:in std_logic;                                   -- bt 為基本題按鈕，bt2 為加法題按鈕
            HEX0,HEX1,HEX2,HEX3:out std_logic_vector(6 downto 0)    -- 七段顯示器結果
            );
end mul_nbit_hex;

architecture mul_nbit_hex of mul_nbit_hex 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;

        component digits is
        port(
                BIN:in integer range 0 to 9999;
                num3: out integer range 0 to 9;
                num2: out integer range 0 to 9;
                num1: out integer range 0 to 9;
                num0: out integer range 0 to 9
                );

        end component;

        component add is
            port(
                X:in std_logic_vector(3 downto 0);
                Y:in std_logic_vector(3 downto 0);
                Q:out std_logic_vector(7 downto 0)
            );
        end component;

        component decoder_7seg is
            PORT(
                    BCD:in std_logic_vector(3 downto 0);
                    HEX:out std_logic_vector(6 downto 0)
                    );
        end component;


        type INT_array is Array (0 to 3) of integer range 0 to 9;
        type LOGIC_array is Array (0 to 3) of std_logic_vector(3 downto 0);
        signal num,num1,num2: INT_array;                            -- 存放乘法每位數轉換的整數
        signal P, f0, f1, f2, f3:std_logic_vector(7 downto 0);      -- 存放運算結果
        signal f4:std_logic_vector(15 downto 0);                    -- 8 bits * 8 bits = 16 bits
        signal show:LOGIC_array;                                    -- 存放待轉換七段顯示器的數值

```

```
begin

    FUN0: add port map(X, Y, f0);                                   -- 00 為加法
    FUN1: mul_nn generic map(4) port map(X, Y, f1);                 -- 01 為乘法
    FUN2: mul_nn generic map(4) port map(X, X, f2);                 -- 10 為X的平方
    FUN3: mul_nn generic map(4) port map(Y, Y, f3);                 -- 11 為Y的平方
    FUN4: mul_nn generic map(8) port map(f2, f3, f4);               -- 當 bt2 按下(0)時，為 ((X*X) * (Y*Y))

    process(sw, bt2)                                                -- 偵測bt2, sw，因 bt 只決定顯示結果 or 撥到的數字，但這裡只做P的運算結果，而非顯示結果
    begin
        if (bt2 = '0') then
            p <= f4(7 downto 0);                                    -- 乘法結果溢位，取末 8 bits
        else
            if (sw = "00") then
                p <= f0;
            elsif (sw = "01") then
                p <= f1;
            elsif (sw = "10") then
                p <= f2;
            elsif (sw = "11") then
                p <= f3;
            end if;
        end if;
    end process;

    U2:digits port map(Conv_integer(P),num(3),num(2),num(1),num(0));            -- num 為 運算結果(每位數)轉換的整數(按下按鈕的時候)

    U3:digits port map(Conv_integer(X),num1(3),num1(2),num1(1),num1(0));        -- num1 為 本身來源(每位數)的整數(沒按按鈕的時候)
    U4:digits port map(Conv_integer(Y),num2(3),num2(2),num2(1),num2(0));        -- num2 為 本身來源(每位數)的整數(沒按按鈕的時候)

    show(0) <= conv_std_logic_vector(num(0),4) when (bt = '0')or(bt2 = '0') else    -- 按下按鈕(0)顯示運算結果，不然(1)顯示撥到的數字
            conv_std_logic_vector(num1(0),4);

    show(1) <= conv_std_logic_vector(num(1),4) when (bt = '0')or(bt2 = '0') else
            conv_std_logic_vector(num1(1),4);

    show(2) <= conv_std_logic_vector(num(2),4) when (bt = '0')or(bt2 = '0') else
            conv_std_logic_vector(num2(0),4);

    show(3) <= conv_std_logic_vector(num(3),4) when (bt = '0')or(bt2 = '0') else
            conv_std_logic_vector(num2(1),4);

    HEX0_part:decoder_7seg port map(show(0),HEX0);                              -- 轉換數值成七段顯示器
    HEX1_part:decoder_7seg port map(show(1),HEX1);
    HEX2_part:decoder_7seg port map(show(2),HEX2);
    HEX3_part:decoder_7seg port map(show(3),HEX3);


end mul_nbit_hex;
```

以上為包裝所有時做功能的主程式碼，包括加分題也在其中

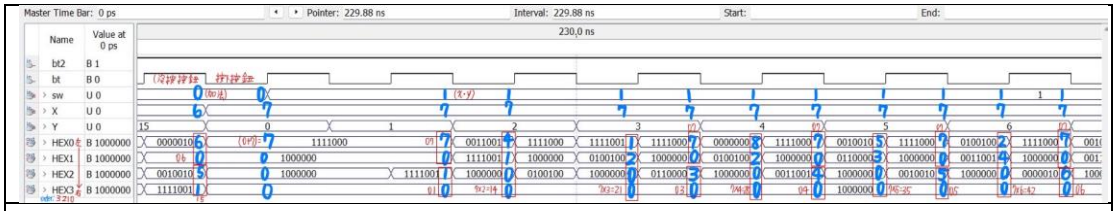
波形圖

本人在此處只顯示出部分結果，由於波行模擬在停止線大於 1us 會產生 ERROR，除非最小的週期壓縮至 0.48828125 ns，必超出限制範圍，故以部分結果表示此 code 可以正常運作。

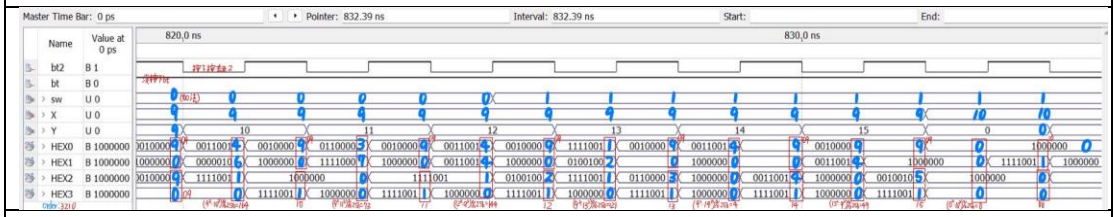
分析為：考慮一開始是否 bt 作用分別結果是啥，Y 為 2ns，各 1ns 為區間。

再考慮區間為(0~15)，故 2 * 16 = 32ns 為 X，再考慮 sw 區間也是(0~15)，即 32 * 16 = 1536ns，而 bt 要包含 sw 四個模式，為 1536 * 4 = 6144ns，最後考慮 bt2 是否作用，故 6144 * 2 = 12888ns。

而在心得後面本人補上全部結果，當然因為時間的限制，我只能找出限制時間的問題，因此全部的結果無法顯示七段顯示器結果。



以上圖片附上本人計算結果，確認即使只有部分結果，依然是正確的顯示。(沒按 bt2)



以上圖片附上本人計算結果，確認即使只有部分結果，依然是正確的顯示。(按下 bt2)

加分題:按下 BUTTON2 後顯示$(X \cdot Y)^2$

程式碼（由於基本提即包含加分題，佳芬提部分為複製貼上，說明已在基本題）

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity mul_nbit_hex is
    PORT(
            X:in std_logic_vector(3 downto 0);                      -- 資料來源
            Y:in std_logic_vector(3 downto 0);                      -- 資料來源
            sw:in std_logic_vector(1 downto 0);
            bt, bt2:in std_logic;                                   -- bt 為基本題按鈕，bt2 為加法題按鈕
            HEX0,HEX1,HEX2,HEX3:out std_logic_vector(6 downto 0)    -- 七段顯示器結果
            );
end mul_nbit_hex;

architecture mul_nbit_hex of mul_nbit_hex 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;

        component digits is
        port(
                BIN:in integer range 0 to 9999;
                num3: out integer range 0 to 9;
                num2: out integer range 0 to 9;
                num1: out integer range 0 to 9;
                num0: out integer range 0 to 9
                );

        end component;

```

```vhdl
        component add is
            port(
                X:in std_logic_vector(3 downto 0);
                Y:in std_logic_vector(3 downto 0);
                Q:out std_logic_vector(7 downto 0)
            );
        end component;

        component decoder_7seg is
            PORT(
                    BCD:in std_logic_vector(3 downto 0);
                    HEX:out std_logic_vector(6 downto 0)
                    );
        end component;


        type INT_array is Array (0 to 3) of integer range 0 to 9;
        type LOGIC_array is Array (0 to 3) of std_logic_vector(3 downto 0);
        signal num,num1,num2: INT_array;                                    -- 存放乘法每位數轉換的整數
        signal P, f0, f1, f2, f3:std_logic_vector(7 downto 0);              -- 存放運算結果
        signal f4:std_logic_vector(15 downto 0);                            -- 8 bits * 8 bits = 16 bits
        signal show:LOGIC_array;                                            -- 存放待轉換七段顯示器的數值

begin

    FUN0: add port map(X, Y, f0);                                           -- 00 為加法
    FUN1: mul_nn generic map(4) port map(X, Y, f1);                         -- 01 為乘法
    FUN2: mul_nn generic map(4) port map(X, X, f2);                         -- 10 為X的平方
    FUN3: mul_nn generic map(4) port map(Y, Y, f3);                         -- 11 為Y的平方
    FUN4: mul_nn generic map(8) port map(f2, f3, f4);                       -- 當 bt2 按下(0)時，為 ((X*X) * (Y*Y))

    process(sw, bt2)                                                        -- 偵測bt2, sw，因 bt 只決定顯示結果 or 撥到的數字，但這裡只做P的運算結果，而非顯示結果
    begin
        if (bt2 = '0') then
            p <= f4(7 downto 0);                                            -- 乘法結果溢位，取末 8 bits
        else
            if (sw = "00") then
                p <= f0;
            elsif (sw = "01") then
                p <= f1;
            elsif (sw = "10") then
                p <= f2;
            elsif (sw = "11") then
                p <= f3;
            end if;
        end if;
    end process;

    U2:digits port map(Conv_integer(P),num(3),num(2),num(1),num(0));        -- num 為 運算結果(每位數)轉換的整數(按下按鈕的時候)

    U3:digits port map(Conv_integer(X),num1(3),num1(2),num1(1),num1(0));    -- num1 為 本身來源(每位數)的整數(沒按按鈕的時候)
    U4:digits port map(Conv_integer(Y),num2(3),num2(2),num2(1),num2(0));    -- num2 為 本身來源(每位數)的整數(沒按按鈕的時候)

    show(0) <= conv_std_logic_vector(num(0),4) when (bt = '0')or(bt2 = '0') else    -- 按下按鈕(0)顯示運算結果，不然(1)顯示撥到的數字
            conv_std_logic_vector(num1(0),4);

    show(1) <= conv_std_logic_vector(num(1),4) when (bt = '0')or(bt2 = '0') else
            conv_std_logic_vector(num1(1),4);

    show(2) <= conv_std_logic_vector(num(2),4) when (bt = '0')or(bt2 = '0') else
            conv_std_logic_vector(num2(0),4);

    show(3) <= conv_std_logic_vector(num(3),4) when (bt = '0')or(bt2 = '0') else
            conv_std_logic_vector(num2(1),4);

    HEX0_part:decoder_7seg port map(show(0),HEX0);                          -- 轉換數值成七段顯示器
    HEX1_part:decoder_7seg port map(show(1),HEX1);
    HEX2_part:decoder_7seg port map(show(2),HEX2);
    HEX3_part:decoder_7seg port map(show(3),HEX3);


end mul_nbit_hex;
```

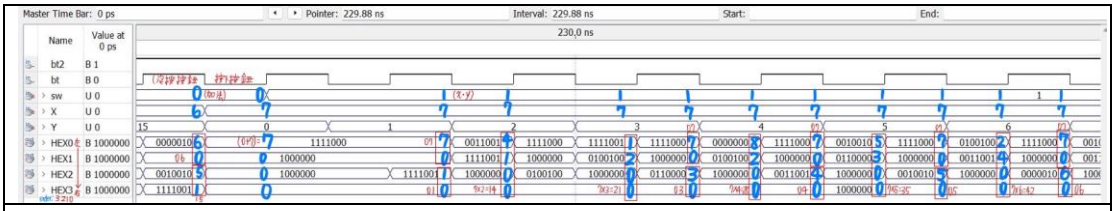
以上為包裝所有時做功能的主程式碼，包括加分題也在其中

波形圖

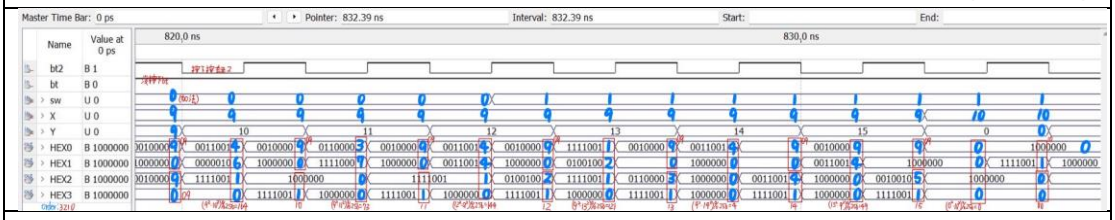
本人在此處只顯示出部分結果，由於波行模擬在停止線大於 1us 會產生 ERROR，除非最小的週期壓縮至 0.48828125 ns，必超出限制範圍，故以部分結果表示此 code 可以正常運作。

分析為：考慮一開始是否 bt 作用分別結果是啥，Y 為 2ns，各 1ns 為區間。再考慮區間為(0~15)，故 2 * 16 = 32ns 為 X，再考慮 sw 區間也是(0~15)，即 32 * 16 = 1536ns，而 bt 要包含 sw 四個模式，為 1536 * 4 = 6144ns，最後考慮 bt2 是否作用，故 6144 * 2 = 12888ns。

而在心得後面本人補上全部結果，當然因為時間的限制，我只能找出限制時間的問題，因此全部的結果無法顯示七段顯示器結果。



以上圖片附上本人計算結果，確認即使只有部分結果，依然是正確的顯示。(沒按 bt2)



以上圖片附上本人計算結果，確認即使只有部分結果，依然是正確的顯示。(按下 bt2)

心得

經由此次實驗，我更了解如何運用開學至今所學的部分，從一開始把全部代碼寫在同一份檔案，而現在的我學會了 port map，學會包裝程式碼，增加可讀性，也提高重複利用率，更增加了可驗證性，即更方便的為一小單元做個別的測試，而不是全部做完才只能祈禱一次撞過。

在此次實習學到最重要的即轉換為數之餘時間的問題，由於想要驗證全部的結果是否正確，又因為想要取整故最小的週期抓了 6ns，而結果就是最大的週期需要 12,288ns，因此需要分開看結果是否正確。

確認 show

經由排除法，本人發現問題點出再 digit 的 entity，但因為邏輯上我需要除法、取餘數的操作，因為不可省略，似乎也沒有改善的餘地，因此以下結果為在 1us 內能做到最大限度確認 show 即在進入七段顯示器前的各位數應為什麼。

發生了點小意外，本人發現等待五分鐘以上是有機會跑出來的，因此即使出現 ERROR，可發現全是接近 1us 附近會發生的，但似乎還是可以的，但很顯然這屬於一種危險操作。

```
# ERROR! Vector Mismatch for output port ans[0] :: @time = 999424.000 ps
#      Expected value = 00000000
#      Real value = 11000001
# ERROR! Vector Mismatch for output port ans[6] :: @time = 999424.000 ps
#      Expected value = 00000000
#      Real value = 11000001
# ERROR! Vector Mismatch for output port ans[7] :: @time = 999424.000 ps
#      Expected value = 00000000
#      Real value = 11000001
# ERROR! Vector Mismatch for output port ans[5] :: @time = 999936.000 ps
#      Expected value = 00000000
#      Real value = 11100001
#          4 mismatched vectors : Simulation failed !
# ** Note: $finish    : mul_nbit_nohex.vwf.vt(585)
#    Time: 1 us  Iteration: 0  Instance: /mul_nbit_nohex_vlg_vec_tst/tb_out
```

程式碼

可以明顯看出本人修改了 port 使輸出部分增加了 ans(讓 P 接出結果)，與 show_ans(讓接入七段顯示器前的數直接出結果)，並於程式最後一列使用&串接 4 個 4 bits(0~9 需要 4 bits)

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity mul_nbit_nohex is
    PORT(
            X:in std_logic_vector(3 downto 0);
            Y:in std_logic_vector(3 downto 0);
            sw:in std_logic_vector(1 downto 0);
            bt, bt2:in std_logic;
            show_ans:out std_logic_vector(15 downto 0);
            ans:out std_logic_vector(7 downto 0)
            );
end mul_nbit_nohex;

architecture mul_nbit_hex of mul_nbit_nohex 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;

        component digits is
        port(
                BIN:in integer range 0 to 9999;
                num3: out integer range 0 to 9;
                num2: out integer range 0 to 9;
                num1: out integer range 0 to 9;
                num0: out integer range 0 to 9
                );

        end component;
```

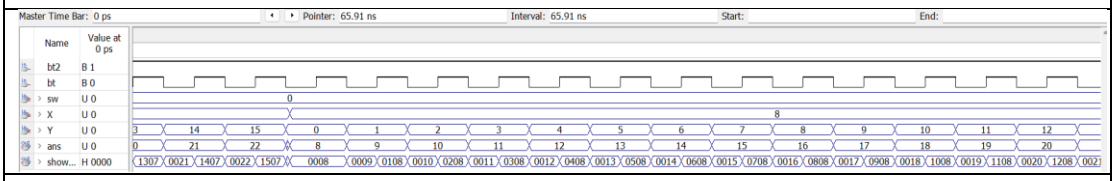
```
38
39          component add is
40              port(
41                  X:in std_logic_vector(3 downto 0);
42                  Y:in std_logic_vector(3 downto 0);
43                  Q:out std_logic_vector(7 downto 0)
44              );
45          end component;
46
47          component decoder_7seg is
48              PORT(
49                      BCD:in std_logic_vector(3 downto 0);
50                      HEX:out std_logic_vector(6 downto 0)
51                      );
52          end component;
53
54
55          type INT_array is Array (0 to 3) of integer range 0 to 9;
56          type LOGIC_array is Array (0 to 3) of std_logic_vector(3 downto 0);
57          signal num,num1,num2: INT_array;
58          signal P, f0, f1, f2, f3:std_logic_vector(7 downto 0);
59          signal f4:std_logic_vector(15 downto 0);
60          signal show:LOGIC_array;
```

```
begin

    FUN0: add port map(X, Y, f0);
    FUN1: mul_nn generic map(4) port map(X, Y, f1);
    FUN2: mul_nn generic map(4) port map(X, X, f2);
    FUN3: mul_nn generic map(4) port map(Y, Y, f3);
    FUN4: mul_nn generic map(8) port map(f2, f3, f4);

    process(sw, bt2)
    begin
        if (bt2 = '0') then
            p <= f4(7 downto 0);
        else
            if (sw = "00") then
                p <= f0;
            elsif (sw = "01") then
                p <= f1;
            elsif (sw = "10") then
                p <= f2;
            elsif (sw = "11") then
                p <= f3;
            end if;
        end if;
    end process;
    ans <= p;
```

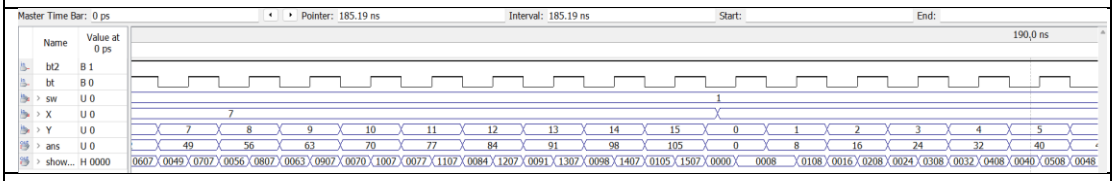
```

    U2:digits port map(Conv_integer(P),num(3),num(2),num(1),num(0));

    U3:digits port map(Conv_integer(X),num1(3),num1(2),num1(1),num1(0));
    U4:digits port map(Conv_integer(Y),num2(3),num2(2),num2(1),num2(0));

    show(0) <= conv_std_logic_vector(num(0),4) when (bt = '0')or(bt2 = '0') else
            conv_std_logic_vector(num1(0),4);

    show(1) <= conv_std_logic_vector(num(1),4) when (bt = '0')or(bt2 = '0') else
            conv_std_logic_vector(num1(1),4);

    show(2) <= conv_std_logic_vector(num(2),4) when (bt = '0')or(bt2 = '0') else
            conv_std_logic_vector(num2(0),4);

    show(3) <= conv_std_logic_vector(num(3),4) when (bt = '0')or(bt2 = '0') else
            conv_std_logic_vector(num2(1),4);
    show_ans <= show(3) & show(2) & show(1) & show(0);                    -- 為了可以顯示，將.vwf 時間壓在 1us，無法顯示出全部結果(最小的週期要取道0.步)故顯示部分以表正常運作
end mul_nbit_hex;
```

波形圖

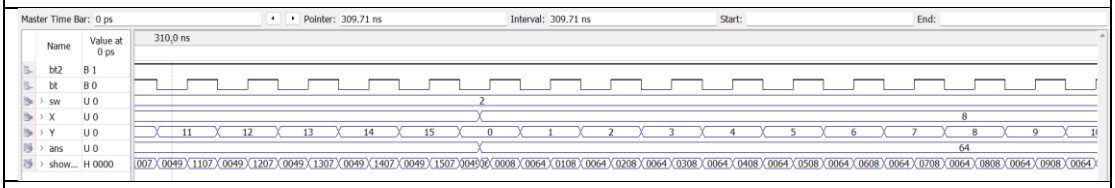
參考檔案：mul_nbit_nohex.vwf



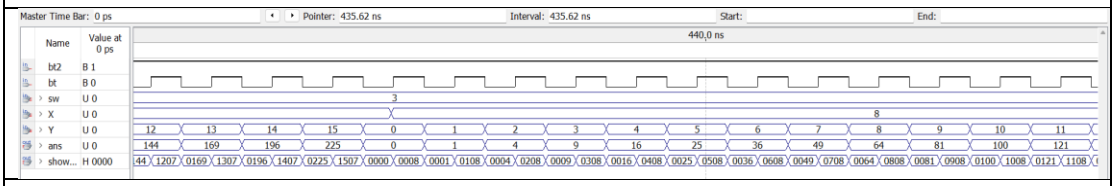
部分加法，考慮為了讓您看到 sw = 0，停在此區間



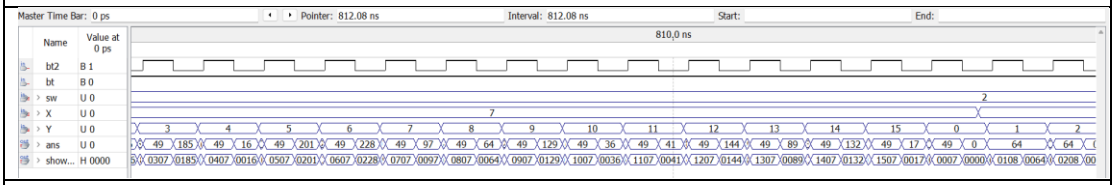
部分乘法，考慮為了讓您看到 sw = 1，停在此區間



部分平方，考慮為了讓您看到 sw = 2，停在此區間



部分平方，考慮為了讓您看到 sw = 3，停在此區間



部分 bt2 運作結果，旨在表示運作符合預期

確認全部輸出結果：

藉由禮拜五當場坐牢搞出來燒完板子觀察發現基本題可以正常運作，保證結果 P 每位數轉換成七段顯示器是正確結果，因此只確認 P 結果是否符合加分題。且因為 P 為運算結果，而非顯示結果，在未按下任何按鈕時目前強迫每 bits 為 1，即 255。

程式碼

可以明顯看出本人修改了 port 使 P 可以串接 ans 查看是否經由 sw 選到合適的運算，最後也從波型圖得知結果符合自身預期。

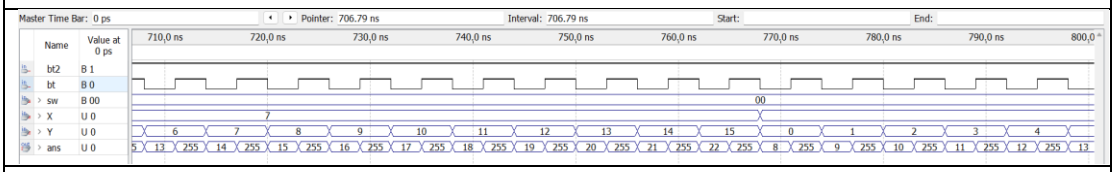
```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_unsigned.all;
4  use ieee.std_logic_arith.all;
5
6  entity mul_nbit_all is
7      PORT(
8              X:in std_logic_vector(3 downto 0);
9              Y:in std_logic_vector(3 downto 0);
10             sw:in std_logic_vector(1 downto 0);
11             bt, bt2:in std_logic;
12             ans:out std_logic_vector(7 downto 0)
13             );
14 end mul_nbit_all;
15
16 architecture mul_nbit_hex of mul_nbit_all is
17
18         component mul_nn is
19             generic(number :integer range 1 to 32);
20             PORT(
21                     X:in std_logic_vector(number-1 downto 0);
22                     Y:in std_logic_vector(number-1 downto 0);
23                     Ans:out std_logic_vector(2*number-1 downto 0)
24                     );
25         end component;
26
27         component digits is
28         port(
29                 BIN:in integer range 0 to 9999;
30                 num3: out integer range 0 to 9;
31                 num2: out integer range 0 to 9;
32                 num1: out integer range 0 to 9;
33                 num0: out integer range 0 to 9
34                 );
35
36         end component;
37
```

```vhdl
        component add is
            port(
                X:in std_logic_vector(3 downto 0);
                Y:in std_logic_vector(3 downto 0);
                Q:out std_logic_vector(7 downto 0)
            );
        end component;

        component decoder_7seg is
            PORT(
                    BCD:in std_logic_vector(3 downto 0);
                    HEX:out std_logic_vector(6 downto 0)
                    );
        end component;


        type INT_array is Array (0 to 3) of integer range 0 to 9;
        type LOGIC_array is Array (0 to 3) of std_logic_vector(3 downto 0);
        signal num,num1,num2: INT_array;
        signal P, f0, f1, f2, f3:std_logic_vector(7 downto 0);
        signal f4:std_logic_vector(15 downto 0);
        signal show:LOGIC_array;

```

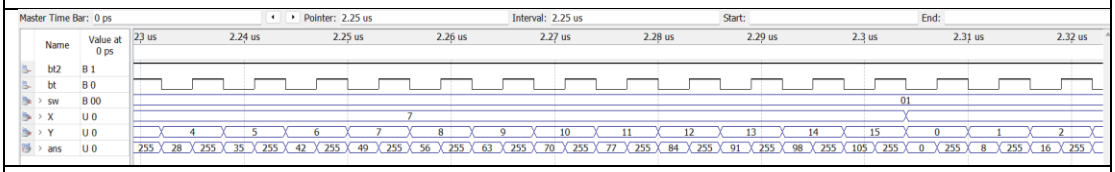
```vhdl
begin

    FUN0: add port map(X, Y, f0);
    FUN1: mul_nn generic map(4) port map(X, Y, f1);
    FUN2: mul_nn generic map(4) port map(X, X, f2);
    FUN3: mul_nn generic map(4) port map(Y, Y, f3);
    FUN4: mul_nn generic map(8) port map(f2, f3, f4);

    process(sw, bt2, bt)
    begin
        if (bt2 = '0') then
            p <= f4(7 downto 0);
        else
            if (bt = '0') then
                if (sw = "00") then
                    p <= f0;
                elsif (sw = "01") then
                    p <= f1;
                elsif (sw = "10") then
                    p <= f2;
                elsif (sw = "11") then
                    p <= f3;
                end if;
            else
                p <= (others => '1');                    -- 如果 不按任何按鈕，結果為 255 ，因為沒有決定P應該顯示什麼，故顯示運算結果(波型圖卡到QQ)
            end if;
        end if;
    end process;
    ans <= p;                                            -- 顯示運算結果，確認在七段顯示前是正確的
end mul_nbit_hex;
```

波形圖

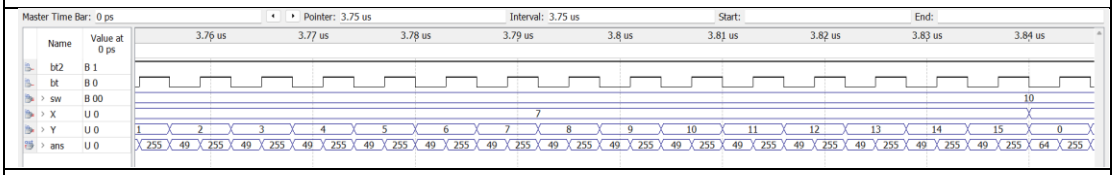
參考檔案：mul_nbit_all.vwf



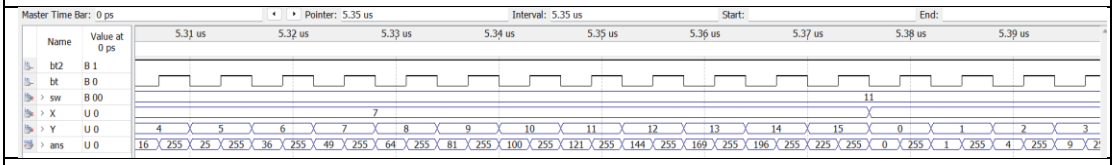
部分加法，考慮為了讓您看到 sw = 0，停在此區間



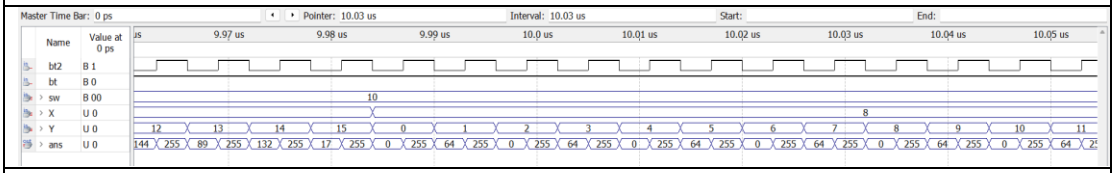
部分乘法，考慮為了讓您看到 sw = 1，停在此區間



部分平方，考慮為了讓您看到 sw = 2，停在此區間



部分平方，考慮為了讓您看到 sw = 3，停在此區間



部分 bt2 運作結果，旨在表示運作符合預期

最後的嘗試：

這是一場對於 cpu 的折磨，經過前面幾個 ERROR 多到爆炸但只要等得夠久，好像都有正確的波型圖跑出來，我漸漸意識到似乎不是 ERROR 跑不出來，是因為某個沒$的人到現在還在用四代 i5，所以理論上 code 沒問題，只要拿著這代碼跟十幾年前的時脈比誰有耐心應該~就會過了？（我很希望是這樣）

程式碼

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity mul_nbit_hex is
    PORT(
            X:in std_logic_vector(3 downto 0);                      -- 資料來源
            Y:in std_logic_vector(3 downto 0);                      -- 資料來源
            sw:in std_logic_vector(1 downto 0);
            bt, bt2:in std_logic;                                   -- bt 為基本題按鈕，bt2 為加法題按鈕
            HEX0,HEX1,HEX2,HEX3:out std_logic_vector(6 downto 0)    -- 七段顯示器結果
            );
end mul_nbit_hex;

architecture mul_nbit_hex of mul_nbit_hex 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;

        component digits is
        port(
                BIN:in integer range 0 to 9999;
                num3: out integer range 0 to 9;
                num2: out integer range 0 to 9;
                num1: out integer range 0 to 9;
                num0: out integer range 0 to 9
                );

        end component;

       component add is
           port(
              X:in std_logic_vector(3 downto 0);
              Y:in std_logic_vector(3 downto 0);
              Q:out std_logic_vector(7 downto 0)
           );
       end component;

       component decoder_7seg is
           PORT(
                   BCD:in std_logic_vector(3 downto 0);
                   HEX:out std_logic_vector(6 downto 0)
                   );
       end component;


       type INT_array is Array (0 to 3) of integer range 0 to 9;
       type LOGIC_array is Array (0 to 3) of std_logic_vector(3 downto 0);
       signal num,num1,num2: INT_array;                             -- 存放乘法每位數轉換的整數
       signal P, f0, f1, f2, f3:std_logic_vector(7 downto 0);       -- 存放運算結果
       signal f4:std_logic_vector(15 downto 0);                     -- 8 bits * 8 bits = 16 bits
       signal show:LOGIC_array;                                     -- 存放待轉換七段顯示器的數值

```

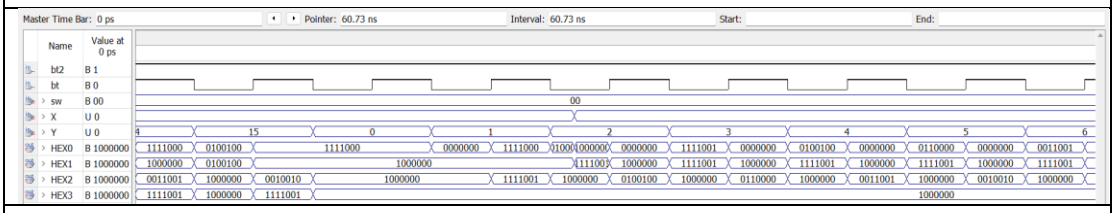
```
begin

    FUN0: add port map(X, Y, f0);                                   -- 00 為加法
    FUN1: mul_nn generic map(4) port map(X, Y, f1);                 -- 01 為乘法
    FUN2: mul_nn generic map(4) port map(X, X, f2);                 -- 10 為X的平方
    FUN3: mul_nn generic map(4) port map(Y, Y, f3);                 -- 11 為Y的平方
    FUN4: mul_nn generic map(8) port map(f2, f3, f4);               -- 當 bt2 按下(0)時，為 ((X*X) * (Y*Y))

    process(sw, bt2)                                                -- 偵測bt2, sw，因 bt 只決定顯示結果 or 撥到的數字，但這裡只做P的運算結果，而非顯示結果
    begin
        if (bt2 = '0') then
            p <= f4(7 downto 0);                                    -- 乘法結果溢位，取末 8 bits
        else
            if (sw = "00") then
                p <= f0;
            elsif (sw = "01") then
                p <= f1;
            elsif (sw = "10") then
                p <= f2;
            elsif (sw = "11") then
                p <= f3;
            end if;
        end if;
    end process;

    U2:digits port map(Conv_integer(P),num(3),num(2),num(1),num(0));            -- num 為 運算結果(每位數)轉換的整數(按下按鈕的時候)

    U3:digits port map(Conv_integer(X),num1(3),num1(2),num1(1),num1(0));        -- num1 為 本身來源(每位數)的整數(沒按按鈕的時候)
    U4:digits port map(Conv_integer(Y),num2(3),num2(2),num2(1),num2(0));        -- num2 為 本身來源(每位數)的整數(沒按按鈕的時候)

    show(0) <= conv_std_logic_vector(num(0),4) when (bt = '0')or(bt2 = '0') else    -- 按下按鈕(0)顯示運算結果，不然(1)顯示撥到的數字
            conv_std_logic_vector(num1(0),4);

    show(1) <= conv_std_logic_vector(num(1),4) when (bt = '0')or(bt2 = '0') else
            conv_std_logic_vector(num1(1),4);

    show(2) <= conv_std_logic_vector(num(2),4) when (bt = '0')or(bt2 = '0') else
            conv_std_logic_vector(num2(0),4);

    show(3) <= conv_std_logic_vector(num(3),4) when (bt = '0')or(bt2 = '0') else
            conv_std_logic_vector(num2(1),4);

    HEX0_part:decoder_7seg port map(show(0),HEX0);                              -- 轉換數值成七段顯示器
    HEX1_part:decoder_7seg port map(show(1),HEX1);
    HEX2_part:decoder_7seg port map(show(2),HEX2);
    HEX3_part:decoder_7seg port map(show(3),HEX3);


end mul_nbit_hex;
```

波形圖

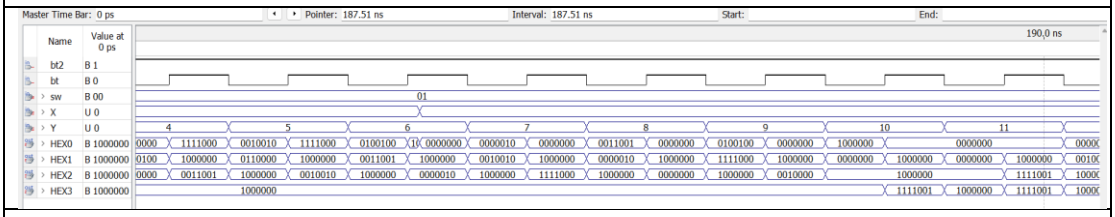
是的，這是一個等等黨的勝利。在經過煎熬的十分鐘，他終究把結果吐出來了，不是我寫得不好，是我等的不夠久： P

但這一切的前提建立在用最多 1us 的時間做完所有事情，因此最小週期甚至只有 0.24ns，如果你跑了這檔案，你可以很明顯發現因為小數點後其實有更多位數，只是他最多做到小數點後兩位故出現許多不同大小的區間。

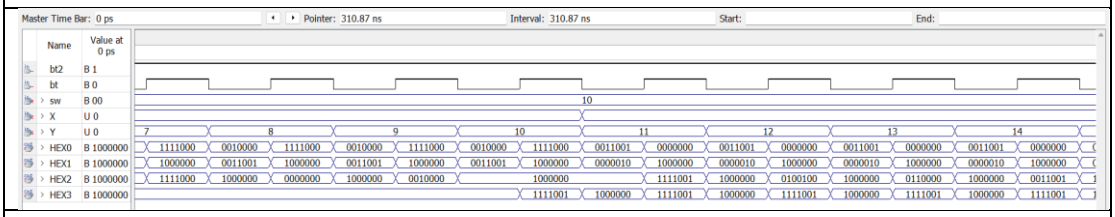
參考檔案：mul_nbit_last.vwf



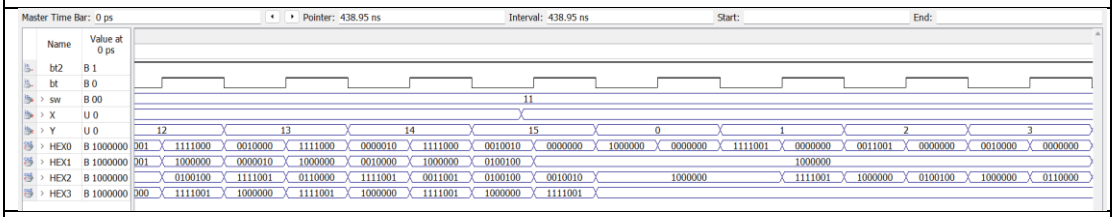
部分加法，考慮為了讓您看到 sw = 0，停在此區間



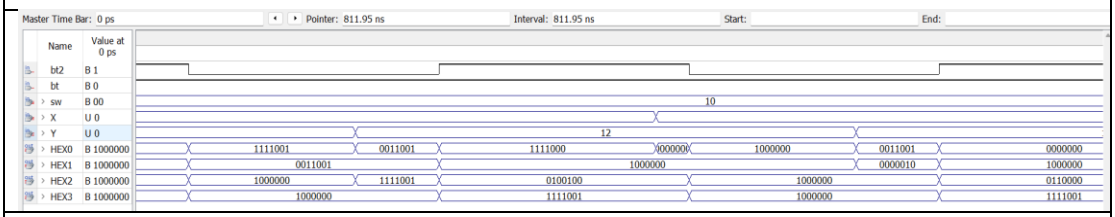
部分乘法，考慮為了讓您看到 sw = 1，停在此區間



部分平方，考慮為了讓您看到 sw = 2，停在此區間



部分平方，考慮為了讓您看到 sw = 3，停在此區間



部分 bt2 運作結果，旨在表示運作符合預期