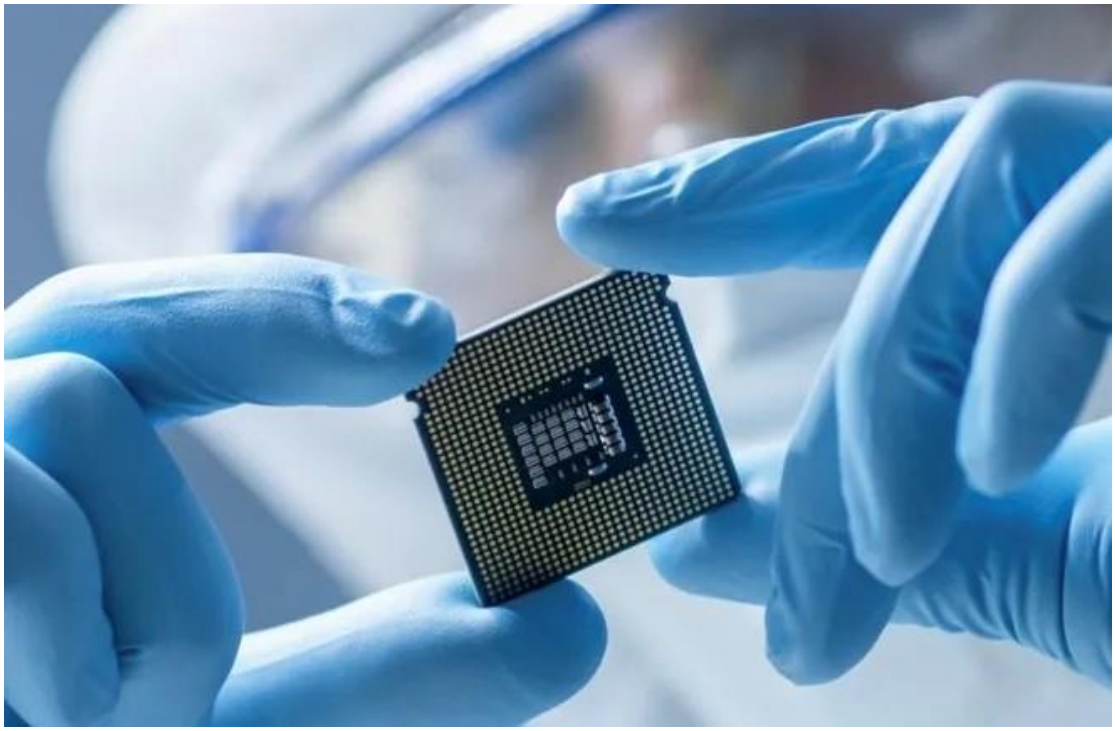


# 數位系統設計作業

# HW8

學號 : 01257027 | 姓名 : 林承羿

# 第一題 : X/Y

前置流程 :

```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;
```

七段顯示器顯示輸入(X, Y, 餘數, 商) 皆是以此轉換

```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;
```

將對應數字轉成方便觀看的七段顯示器

## 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 div is
    port(
            clk:in std_logic;                                                   -- 計算除法需要的clk，此題最多10次
            XY:in std_logic_vector(9 downto 0);                                 -- 選擇輸入
            sw0:in std_logic;                                                   -- 輸入被除數
            sw1:in std_logic;                                                   -- 輸入除數
            start:in std_logic;                                                 -- 開始除法
            HEX0,HEX1,HEX2,HEX3:out std_logic_vector(6 downto 0)                -- 顯示 被除數、除數、(餘數、商)
            );
end div;

architecture MUL_ver1 of div is
        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 decoder_7seg is
            PORT(
                    BCD:in std_logic_vector(3 downto 0);
                    HEX:out std_logic_vector(6 downto 0)
                    );
        end component;
```

```
    signal cnt:std_logic_vector(3 downto 0);                                    -- 指撥開關最多10個，且除法做bits次，最多10次
    signal cnt1:std_logic_vector(25 downto 0);                                  -- 除頻器(延長顯示時間)
    signal divisor, dividen:std_logic_vector(9 downto 0);                       -- 除數、被除數
    signal remainder:std_logic_vector(19 downto 0);                             -- 左(餘)右(商)


    type INT_array is Array (0 to 3) of integer range 0 to 9;
    type INT_array_large is Array (0 to 7) of integer range 0 to 9;
    type LOGIC_array is Array (0 to 3) of std_logic_vector(3 downto 0);
    signal divisor_num, dividen_num: INT_array;                                 -- 除數、被除數(顯示數字)
    signal remainder_num: INT_array_large;                                      -- 餘數、商(顯示數字)
    signal show:LOGIC_array;                                                    -- 顯示數字(轉成七段顯示器)
    signal flag:std_logic;                                                      -- 接受start，確定開始除法
    signal dp:std_logic_vector(3 downto 0);                                     -- 小數點輪播(表示餘數)
begin
    process(clk)                                                                -- 除頻器
    begin
        if clk'event and clk = '1' then
            cnt1 <= cnt1 + 1;
        end if;
    end process;


    process(sw0)                                                                -- 確定被除數
    begin
        if sw0 = '0' then
            dividen <= XY;
        end if;
    end process;
```

```
    process(sw0,start)                                                          -- 開始除法
    begin
        if sw0 = '0' and start = '1' then
            flag <= '0';
        elsif sw0 = '1' and start = '0' then
            flag <= '1';
        end if;
    end process;

    process(sw1)                                                                -- 確定除數
    begin
        if sw1 = '0' then
            divisor <= XY;
        end if;
    end process;
```

```

    process(clk,start)                                                          -- 除法開始(以 10 clk 為一次週期)，於確認start信號
    variable remainder_reg:std_logic_vector(19 downto 0);
    variable divisor_reg, tmp:std_logic_vector(9 downto 0);                     -- 以 tmp 存放相減結果，避免小於零復原動作
    begin
        if start = '0' then                                                     -- 按下按鈕，先做初始化動作
            tmp := (others => '0');
            divisor_reg := divisor;
            remainder_reg := ("000000000" & dividen & '0');                     -- 補零、左移 1 bit
            cnt <= "0000";
        elsif (rising_edge(clk)) then
            if cnt < 10 then
                cnt <= cnt + 1;
                tmp := remainder_reg(19 downto 10) - divisor_reg;               -- 相減
                if(tmp(9) = '0') then                                           -- 最高位數判斷是否大於零
                    remainder_reg := tmp(8 downto 0) & remainder_reg(9 downto 0) & '1';     -- 大於零，左移併補一
                else
                    remainder_reg := remainder_reg(18 downto 0) & '0';          -- 以 tmp 避免還原，左移補零
                end if;
            elsif (cnt = 10) then
                cnt <= cnt +1;                                                  -- 確保 cnt = 10 只會執行一次(餘數只會右移一次)
                remainder_reg := '0' & remainder_reg(19 downto 11) & remainder_reg(9 downto 0);     -- 左半右移
            end if;
        end if;
        remainder <= remainder_reg;                                             -- 於 process 結束前回填 variable 至 signal
    end process;
```

```
108
109     U0:digits port map(Conv_integer(remainder(19 downto 10)),remainder_num(7),remainder_num(6),remainder_num(5),remainder_num(4));  -- 將餘數轉成十進位
110     U1:digits port map(Conv_integer(remainder(9 downto 0)),remainder_num(3),remainder_num(2),remainder_num(1),remainder_num(0));    -- 將商數轉成十進位
111     U2:digits port map(Conv_integer(dividen),dividen_num(3),dividen_num(2),dividen_num(1),dividen_num(0));                        -- 將被除數轉成十進位
112     U3:digits port map(Conv_integer(divisor),divisor_num(3),divisor_num(2),divisor_num(1),divisor_num(0));                        -- 將除數轉成十進位
113
114     show(0) <= conv_std_logic_vector(dividen_num(0),4) when sw0 = '0' else                                                        -- 按下 sw0 顯示被除數十進位
115                conv_std_logic_vector(divisor_num(0),4) when sw1 = '0' else                                                        -- 按下 sw1 顯示除數十進位
116               conv_std_logic_vector(remainder_num(0),4)  when cnt1(25) = '0' and flag = '1' else                                  -- 餘數十進位(配合除頻器延長顯示)
117                conv_std_logic_vector(remainder_num(4),4)  when cnt1(25) = '1' and flag = '1' else                                 -- 商數十進位(配合除頻器延長顯示)
118                "1111";
119
120     show(1) <= conv_std_logic_vector(dividen_num(1),4) when sw0 = '0' else
121                conv_std_logic_vector(divisor_num(1),4) when sw1 = '0' else
122               conv_std_logic_vector(remainder_num(1),4)  when cnt1(25) = '0' and flag = '1' else
123                conv_std_logic_vector(remainder_num(5),4)  when cnt1(25) = '1' and flag = '1' else
124                "1111";
125
126     show(2) <= conv_std_logic_vector(dividen_num(2),4) when sw0 = '0' else
127                conv_std_logic_vector(divisor_num(2),4) when sw1 = '0' else
128               conv_std_logic_vector(remainder_num(2),4)  when cnt1(25) = '0' and flag = '1' else
129                conv_std_logic_vector(remainder_num(6),4)  when cnt1(25) = '1' and flag = '1' else
130
131                "1111";
132
133     show(3) <= conv_std_logic_vector(dividen_num(3),4) when sw0 = '0' else
134                conv_std_logic_vector(divisor_num(3),4) when sw1 = '0' else
135               conv_std_logic_vector(remainder_num(3),4)  when cnt1(25) = '0' and flag = '1' else
136                conv_std_logic_vector(remainder_num(7),4)  when cnt1(25) = '1' and flag = '1' else
137                "1111";
138
139     dp <= (others => '0') when (flag = '1' and cnt1(25) = '1') else (others => '1');                                              -- 餘數小數點(配合除頻器延長顯示)
140
141      HEX0_part:decoder_7seg port map(show(0),HEX0);                                                                               -- 將十進位數字轉成七段顯示
142      HEX1_part:decoder_7seg port map(show(1),HEX1);
143      HEX2_part:decoder_7seg port map(show(2),HEX2);
144      HEX3_part:decoder_7seg port map(show(3),HEX3);
145  end MUL_ver1;
```

以上為主要程式碼，實現除法電路。

## 為更快看到結果，去除除頻器 (波型圖也以此為主)：

```
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   entity div_quick is
6       port(
7               clk:in std_logic;
8               XY:in std_logic_vector(9 downto 0);
9               sw0:in std_logic;
10              sw1:in std_logic;
11              start:in std_logic;
12              dp:out std_logic_vector(3 downto 0);                    -- 接出來看看是否在餘數時顯示
13              HEX0,HEX1,HEX2,HEX3:out std_logic_vector(6 downto 0)
14              );
15  end div_quick;
16  architecture MUL_ver1 of div_quick is
17          component digits is
18          port(
19                  BIN:in integer range 0 to 9999;
20                  num3: out integer range 0 to 9;
21                  num2: out integer range 0 to 9;
22                  num1: out integer range 0 to 9;
23                  num0: out integer range 0 to 9
24                  );
25          end component;
26          component decoder_7seg is
27              PORT(
28                      BCD:in std_logic_vector(3 downto 0);
29                      HEX:out std_logic_vector(6 downto 0)
30                      );
31          end component;
```

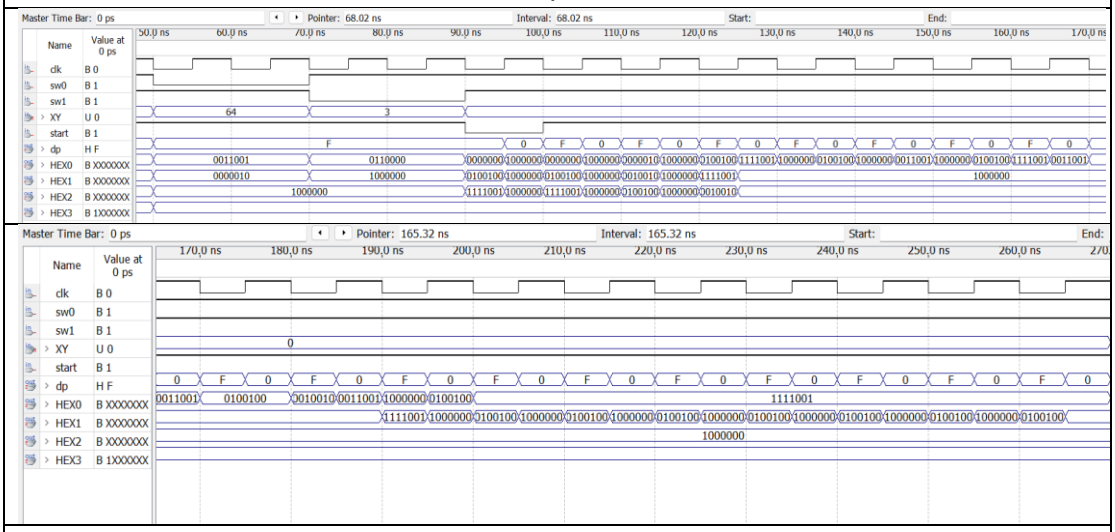
```
    signal cnt:std_logic_vector(3 downto 0);                    -- 指撥開關最多10個，且除法做bits次，最多10次
    signal cnt1:std_logic_vector(25 downto 0);                  -- 除頻器
    signal divisor, dividen:std_logic_vector(9 downto 0);       -- 除數、被除數
    signal remainder:std_logic_vector(19 downto 0);             -- 左(餘)右(商)
    type INT_array is Array (0 to 3) of integer range 0 to 9;
    type INT_array_large is Array (0 to 7) of integer range 0 to 9;
    type LOGIC_array is Array (0 to 3) of std_logic_vector(3 downto 0);
    signal divisor_num, dividen_num: INT_array;                 -- 除數、被除數
    signal remainder_num: INT_array_large;                      -- 餘數、商
    signal show:LOGIC_array;
    signal flag:std_logic;
begin
    process(clk)
    begin
        if clk'event and clk = '1' then
            cnt1 <= cnt1 + 1;
        end if;
    end process;
    process(sw0)                                                -- 被除數
    begin
        if sw0 = '0' then
            dividen <= XY;
        end if;
    end process;
    process(sw0,start)
    begin
        if sw0 = '0' and start = '1' then
            flag <= '0';
        elsif sw0 = '1' and start = '0' then
            flag <= '1';
        end if;
    end process;
    process(sw1)                                                -- 除數
    begin
        if sw1 = '0' then
            divisor <= XY;
        end if;
    end process;
    process(clk,start)
    variable remainder_reg:std_logic_vector(19 downto 0);
    variable divisor_reg, tmp:std_logic_vector(9 downto 0);
    begin
        if start = '0' then
            tmp := (others => '0');
            divisor_reg := divisor;
            remainder_reg := ("000000000" & dividen & '0');
            cnt <= "0000";
        elsif (rising_edge(clk)) then
            if cnt < 10 then
                cnt <= cnt + 1;
                tmp := remainder_reg(19 downto 10) - divisor_reg;
                if(tmp(9) = '0') then
                    remainder_reg := tmp(8 downto 0) & remainder_reg(9 downto 0) & '1';
                else
                    remainder_reg := remainder_reg(18 downto 0) & '0';
                end if;
            elsif (cnt = 10) then
                cnt <= cnt +1;
                remainder_reg := '0' & remainder_reg(19 downto 11) & remainder_reg(9 downto 0);
            end if;
        end if;
        remainder <= remainder_reg;
    end process;
```

```
95      U0:digits port map(Conv_integer(remainder(19 downto 10)),remainder_num(7),remainder_num(6),remainder_num(5),remainder_num(4));
96      U1:digits port map(Conv_integer(remainder(9 downto 0)),remainder_num(3),remainder_num(2),remainder_num(1),remainder_num(0));
97      U2:digits port map(Conv_integer(dividen),dividen_num(3),dividen_num(2),dividen_num(1),dividen_num(0));
98      U3:digits port map(Conv_integer(divisor),divisor_num(3),divisor_num(2),divisor_num(1),divisor_num(0));
99      show(0) <= conv_std_logic_vector(dividen_num(0),4) when sw0 = '0' else
100                conv_std_logic_vector(divisor_num(0),4) when sw1 = '0' else
101             conv_std_logic_vector(remainder_num(0),4)  when clk = '0' and flag = '1' else                    -- 改掉除頻器
102                conv_std_logic_vector(remainder_num(4),4) when clk = '1' and flag = '1' else                  -- 改掉除頻器
103                "1111";
104     show(1) <= conv_std_logic_vector(dividen_num(1),4) when sw0 = '0' else
105                conv_std_logic_vector(divisor_num(1),4) when sw1 = '0' else
106             conv_std_logic_vector(remainder_num(1),4)  when clk = '0' and flag = '1' else
107                conv_std_logic_vector(remainder_num(5),4)  when clk = '1' and flag = '1' else
108                "1111";
109     show(2) <= conv_std_logic_vector(dividen_num(2),4) when sw0 = '0' else
110                conv_std_logic_vector(divisor_num(2),4) when sw1 = '0' else
111             conv_std_logic_vector(remainder_num(2),4)  when clk = '0' and flag = '1' else
112                conv_std_logic_vector(remainder_num(6),4)  when clk = '1' and flag = '1' else
113                "1111";
114     show(3) <= conv_std_logic_vector(dividen_num(3),4) when sw0 = '0' else
115                conv_std_logic_vector(divisor_num(3),4) when sw1 = '0' else
116             conv_std_logic_vector(remainder_num(3),4)  when clk = '0' and flag = '1' else
117                conv_std_logic_vector(remainder_num(7),4)  when clk = '1' and flag = '1' else
118                "1111";
119     dp <= (others => '0') when (flag = '1' and clk = '1') else (others => '1');                              -- 改掉除頻器
120     HEX0_part:decoder_7seg port map(show(0),HEX0);
121     HEX1_part:decoder_7seg port map(show(1),HEX1);
122     HEX2_part:decoder_7seg port map(show(2),HEX2);
123     HEX3_part:decoder_7seg port map(show(3),HEX3);
124 end MUL_ver1;
```

可以看到主要就是修改除頻器，並將顯示的細節拉出至顯示(dp)

## 波形圖

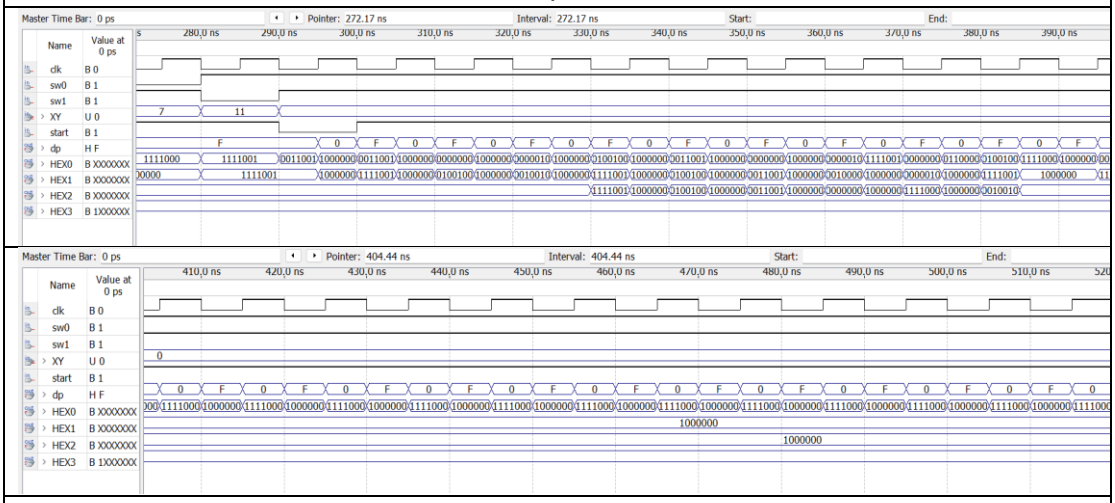
參考檔案：div_quick.vwf (大除小)




以上第一張主要表示本人拿被除數 64，除數 3 做運算，確認 dp 為正常運作，於 0 結果為餘數，1 為商數。

第二章主要展示結果，餘數結果由下而上為 0001，商數結果為 0021，符合運算結果。

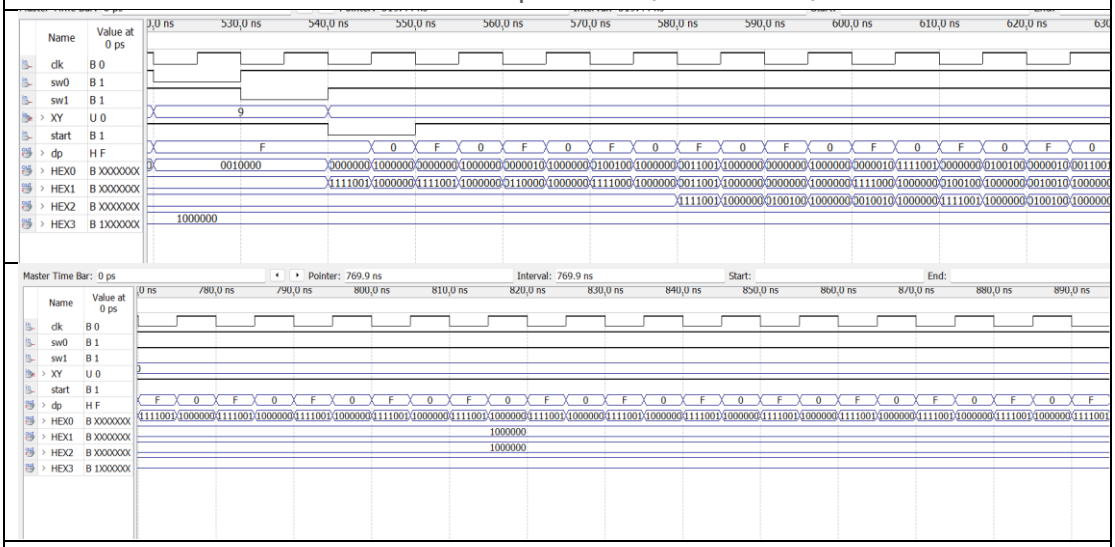
參考檔案：div_quick.vwf (小除大)




以上第一張主要表示本人拿被除數 7，除數 11 做運算，確認 dp 為正常運作，於 0 結果為餘數，1 為商數。

第二章主要展示結果，餘數結果由下而上為 0007，商數結果為 0000，符合運算結果。

參考檔案：div_quick.vwf (同數箱相除)




以上第一張主要表示本人拿被除數 9，除數 9 做運算，確認 dp 為正常運作，於 0 結果為餘數，1 為商數。

第二章主要展示結果，餘數結果由下而上為 0001，商數結果為 0000，符合運算結果。

補充：

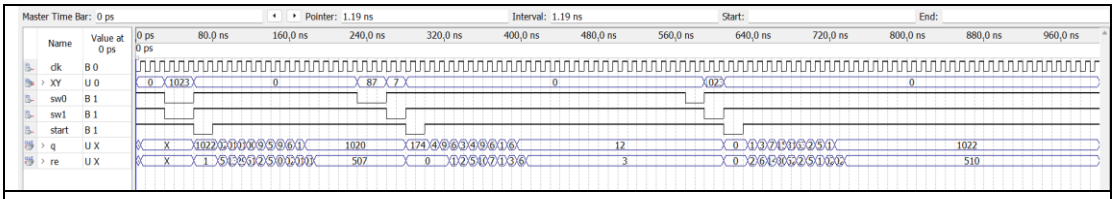
程式碼

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

entity div_watch is
    port(
            clk:in std_logic;
            XY:in std_logic_vector(9 downto 0);
            sw0:in std_logic;
            sw1:in std_logic;
            start:in std_logic;
            re, q:out std_logic_vector(9 downto 0)                  -- 餘數、商(拉出來看結果)
            );
end div_watch;

architecture MUL_ver1 of div_watch is
    signal cnt:std_logic_vector(3 downto 0);                        -- 指撥開關最多10個，且除法做bits次，最多10次
    signal divisor, dividen:std_logic_vector(9 downto 0);           -- 除數、被除數
    signal remainder:std_logic_vector(19 downto 0);                 -- 左(餘)右(商)
    signal flag:std_logic;
begin
    process(sw0)                                                        -- 被除數
    begin
        if sw0 = '0' then
            dividen <= XY;
        end if;
    end process;

    process(sw0,start)
    begin
        if sw0 = '0' and start = '1' then
            flag <= '0';
        elsif sw0 = '1' and start = '0' then
            flag <= '1';
        end if;
    end process;
```

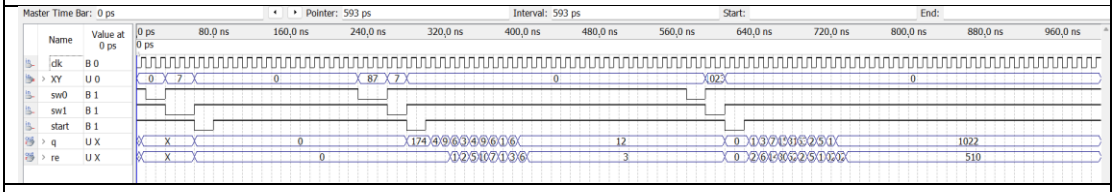
```vhdl
    process(sw1)                                                    -- 除數
    begin
        if sw1 = '0' then
            divisor <= XY;
        end if;
    end process;

    process(clk,start)
    variable remainder_reg:std_logic_vector(19 downto 0);
    variable divisor_reg, tmp:std_logic_vector(9 downto 0);
    begin
        if start = '0' then
            tmp := (others => '0');
            divisor_reg := divisor;
            remainder_reg := ("000000000" & dividen & '0');
            cnt <= "0000";
        elsif (rising_edge(clk)) then
            if cnt < 10 then
                cnt <= cnt + 1;
                tmp := remainder_reg(19 downto 10) - divisor_reg;
                if(tmp(9) = '0') then
                    remainder_reg := tmp(8 downto 0) & remainder_reg(9 downto 0) & '1';
                else
                    remainder_reg := remainder_reg(18 downto 0) & '0';
                end if;
            elsif (cnt = 10) then
                cnt <= cnt +1;
                remainder_reg := '0' & remainder_reg(19 downto 11) & remainder_reg(9 downto 0);
            end if;
        end if;
        remainder <= remainder_reg;
    end process;
    re <= remainder(19 downto 10);                                  -- 回填餘數
    q <= remainder(9 downto 0);                                     -- 回填商
end MUL_ver1;
```

此程式碼主要目的為確認除法的正確性，確認結果進而驗證七段顯示器結果

## 波形圖



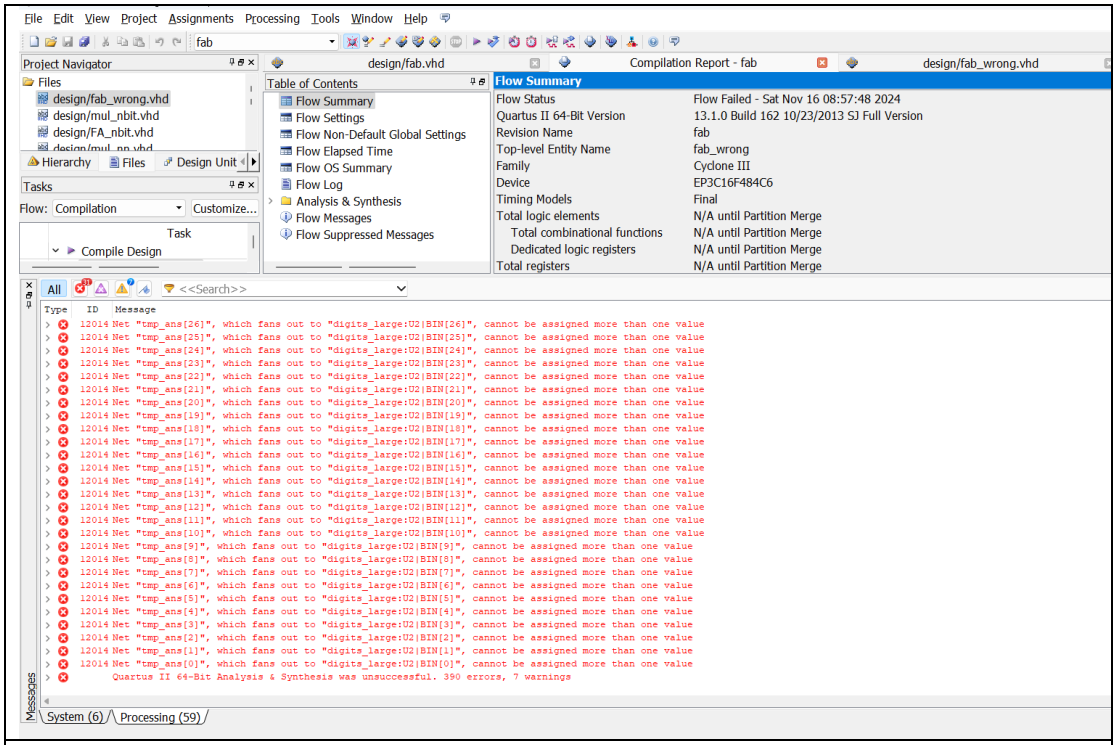
從以上圖的結果可以看出此除法需要在一定範圍內，受到 MSB 無條件左移 1 bit 影響，1023 輸入是一個不合法的。



從以上圖的第一個結果可以看出此除法的問題，遇到零時，因為被除數減去除數必小於零，故每次左移補零，最終餘數、商皆為零。

# 加分題: X! (請用乘法做)

## 第一次嘗試：




謝謝~390 個 ERROR

先左轉適合笨蛋的基本題：P

## 前置作業

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

entity div_1s is
    port(
        clk:in std_logic;
        clk_out:out std_logic:='0'
    );
end div_1s;

architecture bhv of div_1s is
    signal cnt:unsigned(31 downto 0):=(others => '0');       -- 絕對夠用的計算存放空間，並歸零
    signal clk_tmp:std_logic:='0';
begin
    process(clk)
    begin
        if(rising_edge(clk)) then                            -- 累加時脈
            if(cnt = 50000000) then                          -- 50MHZ = MSB 為 1s
                cnt <= (others => '0');                      -- 歸零
                clk_tmp <= not clk_tmp;                      -- 製造除頻的clk
            else
                cnt <= cnt + 1;                              -- 小於目標就繼續累加時脈
            end if;
        end if;
    end process;
    clk_out <= clk_tmp;                                      -- 輸出
end bhv;
```

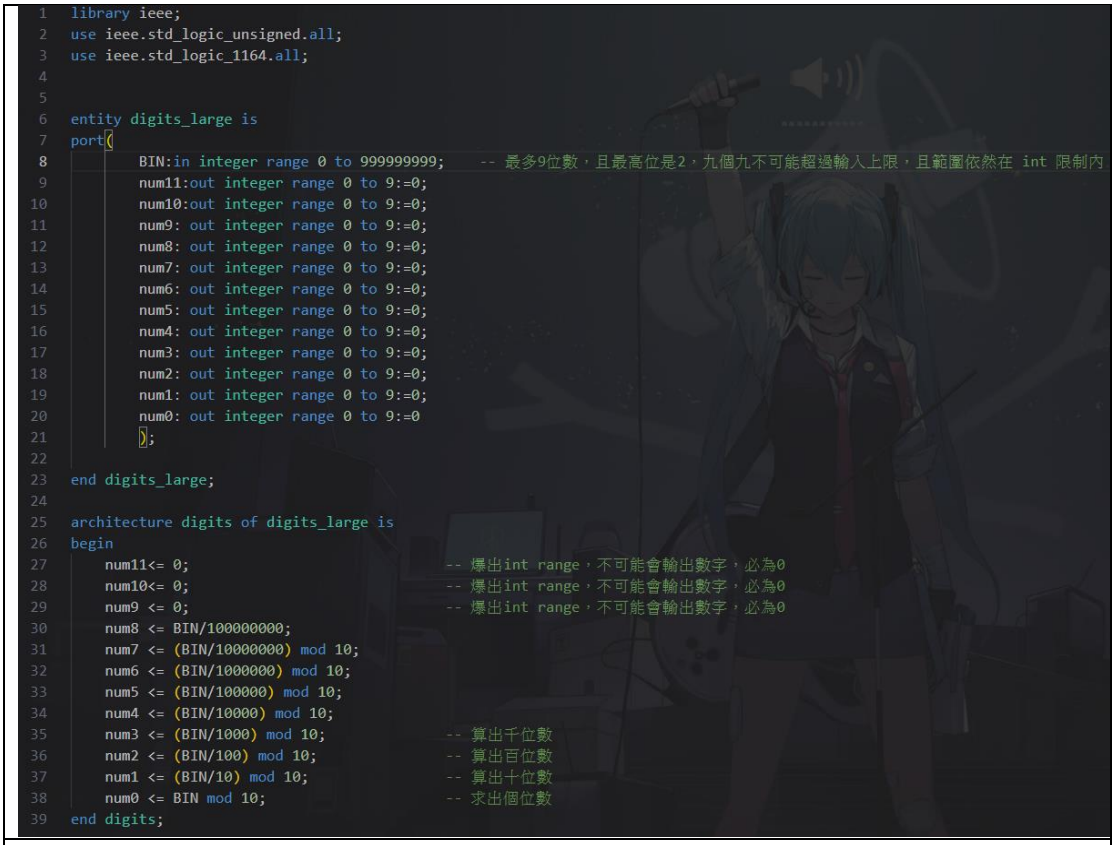
時脈除頻為 1s

```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 MulEhRow 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 : MulEhRow := (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;
```

```vhdl
    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;
```

以上兩張為乘法電路，非快速乘法，就是前兩次的作業

( 選擇此的好處是計算非跟著時脈走，一次算完 )



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


entity digits_large is
port(
        BIN:in integer range 0 to 999999999;    -- 最多9位數，且最高位是2，九個九不可能超過輸入上限，且範圍依然在 int 限制內
        num11:out integer range 0 to 9:=0;
        num10:out integer range 0 to 9:=0;
        num9: out integer range 0 to 9:=0;
        num8: out integer range 0 to 9:=0;
        num7: out integer range 0 to 9:=0;
        num6: out integer range 0 to 9:=0;
        num5: out integer range 0 to 9:=0;
        num4: out integer range 0 to 9:=0;
        num3: out integer range 0 to 9:=0;
        num2: out integer range 0 to 9:=0;
        num1: out integer range 0 to 9:=0;
        num0: out integer range 0 to 9:=0
        );

end digits_large;

architecture digits of digits_large is
begin
    num11<= 0;                          -- 爆出int range，不可能會輸出數字，必為0
    num10<= 0;                          -- 爆出int range，不可能會輸出數字，必為0
    num9 <= 0;                          -- 爆出int range，不可能會輸出數字，必為0
    num8 <= BIN/100000000;
    num7 <= (BIN/10000000) mod 10;
    num6 <= (BIN/1000000) mod 10;
    num5 <= (BIN/100000) mod 10;
    num4 <= (BIN/10000) mod 10;
    num3 <= (BIN/1000) mod 10;          -- 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;
```

上圖是針對答案作的處理，根據資料的預測判斷上限並給出設定極限值

```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;
```

針對來源的轉換，最多接受 12。

```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;
```

轉換每個位數至七段顯示器

預測資料大小：

```python
if __name__  == "__main__":
    ans = 1
    for i in range(1, 16):
        ans = ans*i
        YN = "Y" if(ans < int(2147483647)) else"N"
        print(f"{i:<2} : {ans:<13}, len : {len(str(ans)):<2},  bits len : {ans.bit_length():<2}, display : {YN}")
```

```
> poetry shell
Spawning shell within C:\Users\Magician\Desktop\PyClass\.venv
PowerShell 7.4.6
Loading personal and system profiles took 1883ms.
(pyclass-py3.12) > python .\limit.py
1  : 1           , len : 1 ,  bits len : 1 , display : Y
2  : 2           , len : 1 ,  bits len : 2 , display : Y
3  : 6           , len : 1 ,  bits len : 3 , display : Y
4  : 24          , len : 2 ,  bits len : 5 , display : Y
5  : 120         , len : 3 ,  bits len : 7 , display : Y
6  : 720         , len : 3 ,  bits len : 10, display : Y
7  : 5040        , len : 4 ,  bits len : 13, display : Y
8  : 40320       , len : 5 ,  bits len : 16, display : Y
9  : 362880      , len : 6 ,  bits len : 19, display : Y
10 : 3628800     , len : 7 ,  bits len : 22, display : Y
11 : 39916800    , len : 8 ,  bits len : 26, display : Y
12 : 479001600   , len : 9 ,  bits len : 29, display : Y
13 : 6227020800  , len : 10,  bits len : 33, display : N
14 : 87178291200 , len : 11,  bits len : 37, display : N
15 : 1307674368000, len : 13,  bits len : 41, display : N
(pyclass-py3.12)  03:17:05  CPU: 61% | RAM: 9/15GB  87ms
 home\Desktop\PyClass
```

從以上結果可以看出，最多 12 階，驗證上方 vhdl 為何 10~12 位數直接輸出 0。且可以看出就算套 unsigned 也沒救，12 與 13 階之間差距過大。

## 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 fab is
    port(
        X:in std_logic_vector(9 downto 0);
        start:in std_logic;
        clk:in std_logic;
        HEX0, HEX1, HEX2, HEX3:out std_logic_vector(6 downto 0);
        dp_ans:out std_logic_vector(3 downto 0)
    );
end fab;
architecture behavior of fab is
    component mul_nn is                                    -- 上次的乘法電路，懶惰蟲想吃老本：P
        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 div_1s is                                    --  除頻器
        port(
            clk:in std_logic;
            clk_out:out std_logic:='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;

    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 digits_large is                              -- 位數大的恐怖，經過計算，最多12位顯示，13會爆出int range
        port(
                BIN:in integer range 0 to 999999999;
                num11:out integer range 0 to 9;
                num10:out integer range 0 to 9;
                num9: out integer range 0 to 9;
                num8: out integer range 0 to 9;
                num7: out integer range 0 to 9;
                num6: out integer range 0 to 9;
                num5: out integer range 0 to 9;
                num4: out integer range 0 to 9;
                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;
```

```
67
68      signal clk_out:std_logic;                                       -- 除頻器輸出
69      signal ans:std_logic_vector(29 downto 0):=(others => '0');      -- 最多30 bits 可以容納12的階層顯示
70      signal sor:std_logic_vector(29 downto 0):=(others => '0');
71
72
73      type INT_array is Array (0 to 3) of integer range 0 to 9;
74      type INT_array_ans is Array (0 to 11) of integer range 0 to 9;
75      type ans_array is Array (0 to 12) of std_logic_vector(59 downto 0);
76      type LOGIC_array is Array (0 to 3) of std_logic_vector(3 downto 0);
77      signal num1: INT_array;                                         -- 來源每位數轉換的整數
78      signal show:LOGIC_array;                                        -- 顯示結果
79      signal num:INT_array_ans;
80      signal dp:std_logic_vector(3 downto 0);                         -- 小數點輪播
81      signal flag:std_logic:='0';                                     -- 剛開機，啥都不顯示
82      signal tmp_ans:ans_array;
83  begin
84      tmp_ans(0) <= "000000000000000000000000000000000000000000000000000000000000001";                   -- 初始狀態
85      div: div_1s port map(clk,clk_out);
86      sor <= conv_std_logic_vector(conv_integer(X),30);                                                   -- 因為不知道X幾bit，先轉整數，再轉固定 30 bits
87      multiple:for i in 1 to 12 generate                                                                  -- 想偷懶~但問題卡在這裡，死掉惹：(
88          mul: mul_nn generic map(30) port map(tmp_ans(i-1)(29 downto 0), conv_std_logic_vector(i, 30), tmp_ans(i));
89      end generate multiple;
90      ans <= tmp_ans(conv_integer(sor))(29 downto 0);                                                     -- 將最後的答案輸出
91
92      U1:digits port map(conv_integer(sor), num1(3), num1(2), num1(1), num1(0));                          -- 來源每位數
93      U2:digits_large port map(conv_integer(ans), num(11), num(10), num(9), num(8), num(7), num(6), num(5), num(4), num(3), num(2), num(1), num(0));-- 結果每位數
94
95      begin
96          if (start = '0') then                                       -- 輸入來源
97              show(0) <= conv_std_logic_vector(num1(0),4);
98              show(1) <= conv_std_logic_vector(num1(1),4);
99              show(2) <= conv_std_logic_vector(num1(2),4);
100             show(3) <= conv_std_logic_vector(num1(3),4);
101             dp <= "1000";
102         elsif(start = '1') then                                     -- 開始輪播答案
103             if (rising_edge(clk_out)) then
104                 if (dp = "1000") then                               -- 共陽接角，亮一顆小點(最低四位結果)
105                     show(0) <= conv_std_logic_vector(num(0),4);
106                     show(1) <= conv_std_logic_vector(num(1),4);
107                     show(2) <= conv_std_logic_vector(num(2),4);
108                     show(3) <= conv_std_logic_vector(num(3),4);
109                     dp <= "1110";                                   -- 迭代下一個輸出結果
110                 elsif(dp = "1110") then                             -- 共陽接角，亮兩顆小點(中間四位結果)
111                     show(0) <= conv_std_logic_vector(num(4),4);
112                     show(1) <= conv_std_logic_vector(num(5),4);
113                     show(2) <= conv_std_logic_vector(num(6),4);
114                     show(3) <= conv_std_logic_vector(num(7),4);
115                     dp <= "1100";
116                 elsif(dp = "1100") then                             -- 共陽接角，亮三顆小點(最高四位結果)
117                     show(0) <= conv_std_logic_vector(num(8),4);
118                     show(1) <= conv_std_logic_vector(num(9),4);
119                     show(2) <= conv_std_logic_vector(num(10),4);
120                     show(3) <= conv_std_logic_vector(num(11),4);
121                     dp <= "1000";
122                 end if;
123             end if;
124         end if;
125     end process;
126
127     HEX0_part:decoder_7seg port map(show(0),HEX0);                  -- 轉換數值成七段顯示器
128     HEX1_part:decoder_7seg port map(show(1),HEX1);
129     HEX2_part:decoder_7seg port map(show(2),HEX2);
130     HEX3_part:decoder_7seg port map(show(3),HEX3);
131     dp_ans <= dp;                                                   -- 輸出小數點輪播
132 end behavior;
```

以上為此加分題主要的 code，藉由前置作業提供的 code，最終完善此項課題。值得注意的是狀態的輪播本人是以放開 start 當作開始的信號，即按下 start 只是鎖定輸入，並決定輸出哪個結果當作答案。

但是，並非放開才開始計算，之後本人會補充為何不使用此方法而使用全部都計算完，看您輸入的數字決定輸出哪項結果。

細部說明：

理想上，一開始在已知最多 30bits 是輸入極限，乘法結果自然是 60bits，因此只要一個結果：signal ans : std_logic_vector(59 downto 0):="59 個零+'1'"。最後一個 1 是為了符合 0! = 1，依照高階程式語言如 C++、python，很自然的會拿這個去迭帶 for 迴圈，mul:mul_nn generic map(30) port map(ans(29 downto 0), conv_std_logic_vector(I, 30), ans);。但是結果就是 390 個 ERROR。首先確定的是，目前的作業我都很盡責的沒有找槍手代打，所以我 100000000%肯定乘法電路絕對沒問題，他那報錯幾乎肯定是循環輸入問題。

確認完問題，接下來考慮解決途徑，看上去，那報錯應該是我 ans 接出去又拿回來重新當輸入迭帶出現的問題，因此我需要的是多組答案而非個答案一直循環(寫了三小時看到 390 ERROR 真的會 S...，當然我也修過了，但沒救，直接找別的方法)。

在基本題寫完，我思考了一下，一開始的限制是 for 一定要 generate(因為我需要用到自己寫的乘法器)，可是我要的是開關撥到多少就吐出幾階，因此他不是一個 static 輸入，一定要用 generic(使用這個語法與 signal 不同的是他是一個 static input，因此只要 port map 12 次就可以列舉所有範圍內結果)，這會導致我需要在寫一個 vhdl 去暴力列舉每階層的結果以確保運行的正確。拋開這問題不談，我實在修不好這循環輸入問題...那~何不暴力一點每階都做？

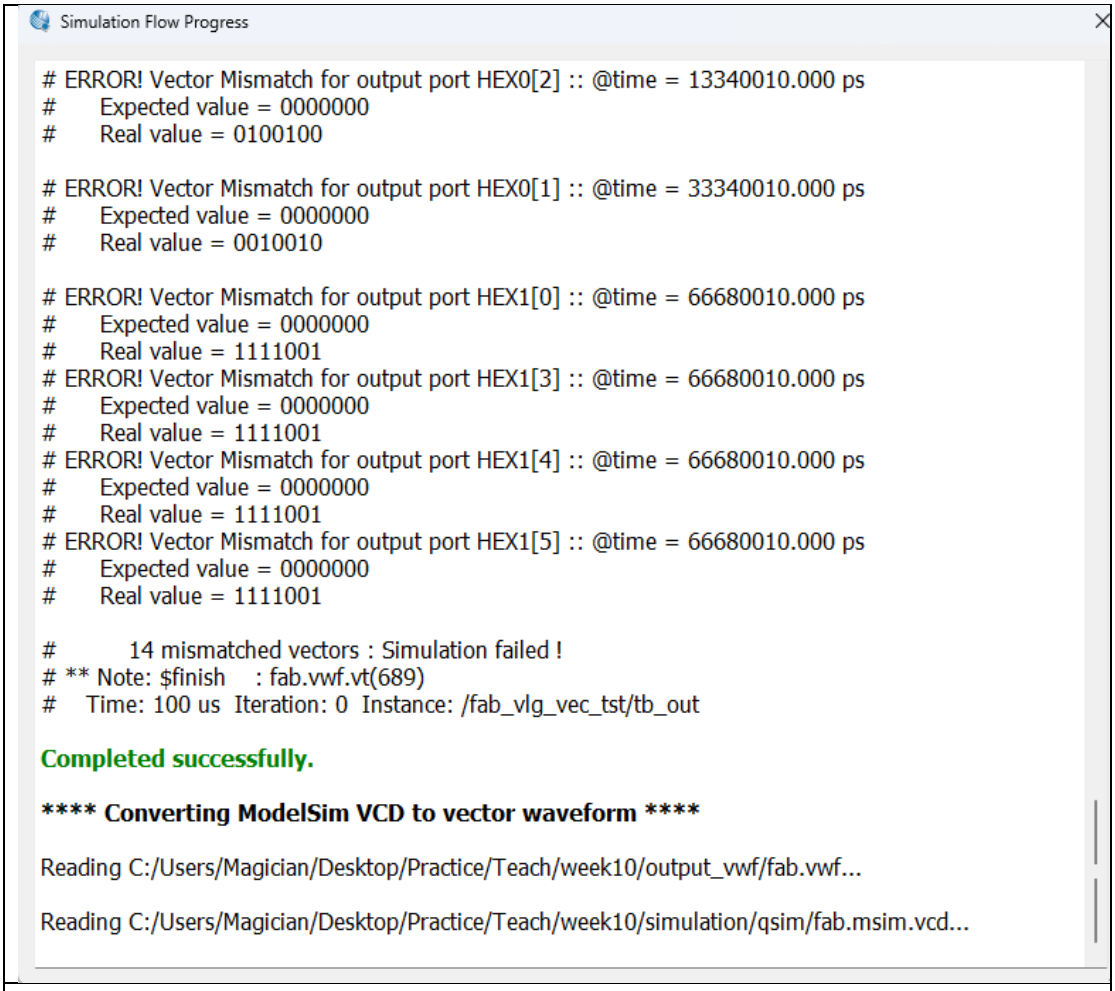
每階都做的好處有，解決了 static 輸入問題(因為已知所有都要做，直接跑到最大值 12)，為了方便搜尋，第一次迭帶我使用 ans(0) = "59 個零+1 個一"當作迭帶條件，0!=1 且第一次是(1! = 1)，這代表要拿出一階我只要 ans(1)，二階就是 ans(2)，以此類推，最大到 ans(12)，這就非常合理： )

結果就是，沒有了循環輸入問題，可以理解的是每次拿前一次的結果與 for 迭帶數字相乘，拿到新的階層當作下一次迭帶的資本，如此一來就結案了。

波形圖

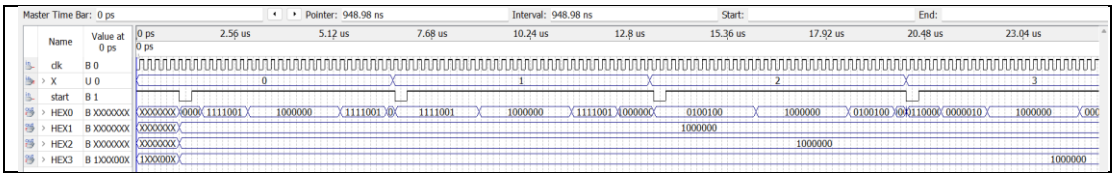
```
# Loading altera_ver.PRIM_GDFF_LOW
# ** Warning: Design size of 4 instances exceeds ModelSim ALTERA recommended capacity.
# This may because you are loading cell libraries which are not recommended with
# the ModelSim Altera version. Expect performance to be adversely affected.

# ERROR! Vector Mismatch for output port HEX0[6] :: @time = 999010.000 ps
#      Expected value = 0000000
#      Real value = 1000000
# ERROR! Vector Mismatch for output port HEX1[6] :: @time = 999010.000 ps
#      Expected value = 0000000
#      Real value = 1000000
# ERROR! Vector Mismatch for output port HEX2[6] :: @time = 999010.000 ps
#      Expected value = 0000000
#      Real value = 1000000
# ERROR! Vector Mismatch for output port HEX3[6] :: @time = 999010.000 ps
#      Expected value = 0000000
#      Real value = 1000000

# ERROR! Vector Mismatch for output port HEX0[0] :: @time = 6670010.000 ps
#      Expected value = 0000000
#      Real value = 1111001
# ERROR! Vector Mismatch for output port HEX0[3] :: @time = 6670010.000 ps
#      Expected value = 0000000
#      Real value = 1111001
# ERROR! Vector Mismatch for output port HEX0[4] :: @time = 6670010.000 ps
#      Expected value = 0000000
#      Real value = 1111001
# ERROR! Vector Mismatch for output port HEX0[5] :: @time = 6670010.000 ps
#      Expected value = 0000000
#      Real value = 1111001

# ERROR! Vector Mismatch for output port HEX0[2] :: @time = 13340010.000 ps
#      Expected value = 0000000
#      Real value = 0100100
```

可以看到為了跑全部的結果，必定超過 1us，且因為設定 1s 切換狀態，clk 必定也是縮到最小，且因為這東西最高只接受 100us，熟悉的東西又跑出來了，我猜猜...大概一小時過後就會跑出來了，反正程式碼是不可能再改的： )




是的，雖然上面看起來好恐怖，但只要綠綠的就好了，這傲嬌的東西一定會吐出對的結果給我的....一定...吧。反正他說在轉波形就代表我成功了，一定是這樣。


雖然 30 分鐘就這麼過了...依然還是沒有結果吐出來，但是看在他這隻怪獸一人吃了四分之一的資源...他應該還在努力中，讓子彈再飛一下。




以上為此加分題的所有波形圖，我相信你也跟我沒什麼耐心一個一個對，於是我寫了下面的東西，將 dp 與結果十進制拉了出來，一切都值觀了許多。

## 細部說明：

可以看到在按下 start 決定完輸入後，並非馬上進入循環，因為需要等待除頻器。且除頻器正緣偵測條件有用必須建立在放開 start 的前提下，意思是如果一直按著不放，會進不了循環，無法顯示答案。

## 程式碼

參考檔案：fab_quick.vhd

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

entity fab_quick is
    port(
        X:in std_logic_vector(9 downto 0);
        start:in std_logic;
        clk:in std_logic;
        dp_ans:out std_logic_vector(3 downto 0);                    -- 配合HEX顯示，查看小數點是否依照順序輪播
        ans_result:out std_logic_vector(29 downto 0);               -- 接出結果，直接看數字
        HEX0, HEX1, HEX2, HEX3:out std_logic_vector(6 downto 0)
    );
end fab_quick;

architecture behavior of fab_quick is
    component mul_nn is                                             -- 上次的乘法電路，懶惰蟲想吃老本：P
        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 div_1s is                                             --  除頻器
        port(
            clk:in std_logic;
            clk_out:out std_logic:='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;

  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 digits_large is                                         -- 位數大的恐怖，經過計算，最多12位顯示，13會爆出int range
     port(
            BIN:in integer range 0 to 999999999;
            num11:out integer range 0 to 9;
            num10:out integer range 0 to 9;
            num9: out integer range 0 to 9;
            num8: out integer range 0 to 9;
            num7: out integer range 0 to 9;
            num6: out integer range 0 to 9;
            num5: out integer range 0 to 9;
            num4: out integer range 0 to 9;
            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;

  signal clk_out:std_logic;                                                 -- 除頻器輸出
  -- signal ans:std_logic_vector(29 downto 0):=(others => '0');             -- 最多30 bits 可以容納12的階層顯示
  signal sor:std_logic_vector(29 downto 0):=(others => '0');
```

```
    type INT_array is Array (0 to 3) of integer range 0 to 9;
    type INT_array_ans is Array (0 to 11) of integer range 0 to 9;
    type ans_array is Array (0 to 12) of std_logic_vector(59 downto 0);
    type LOGIC_array is Array (0 to 3) of std_logic_vector(3 downto 0);
    signal num1: INT_array;                                          -- 來源每位數轉換的整數
    signal show:LOGIC_array;                                         -- 顯示結果
    signal num:INT_array_ans;
    signal dp:std_logic_vector(3 downto 0):=(others => '1');         -- 小數點輪播
    signal flag:std_logic:='0';                                      -- 剛開機，啥都不顯示
    signal tmp_ans:ans_array;
begin
    tmp_ans(0) <= "000000000000000000000000000000000000000000000000000000000001";  -- 初始狀態
    div: div_1s port map(clk,clk_out);
    sor <= conv_std_logic_vector(conv_integer(X),30);                -- 因為不知道X幾bit，先轉整數，再轉固定 30 bits
    multiple:for i in 1 to 12 generate                               -- 想偷懶~但開題卡在這裡，死掉哥：(
        mul: mul_nn generic map(30) port map(tmp_ans(i-1)(29 downto 0), conv_std_logic_vector(i, 30), tmp_ans(i));
    end generate multiple;
    ans_result <= tmp_ans(conv_integer(sor))(29 downto 0);                                  -- 將最後的答案輸出

    U1:digits port map(conv_integer(sor), num1(3), num1(2), num1(1), num1(0));              -- 來源每位數
    U2:digits_large port map(conv_integer(tmp_ans(conv_integer(sor))(29 downto 0)), num(11), num(10), num(9), num(8), num(7), num(6), num(5), num(4), num(3), num(2), num(1), num(0));-- 結果每位數
```

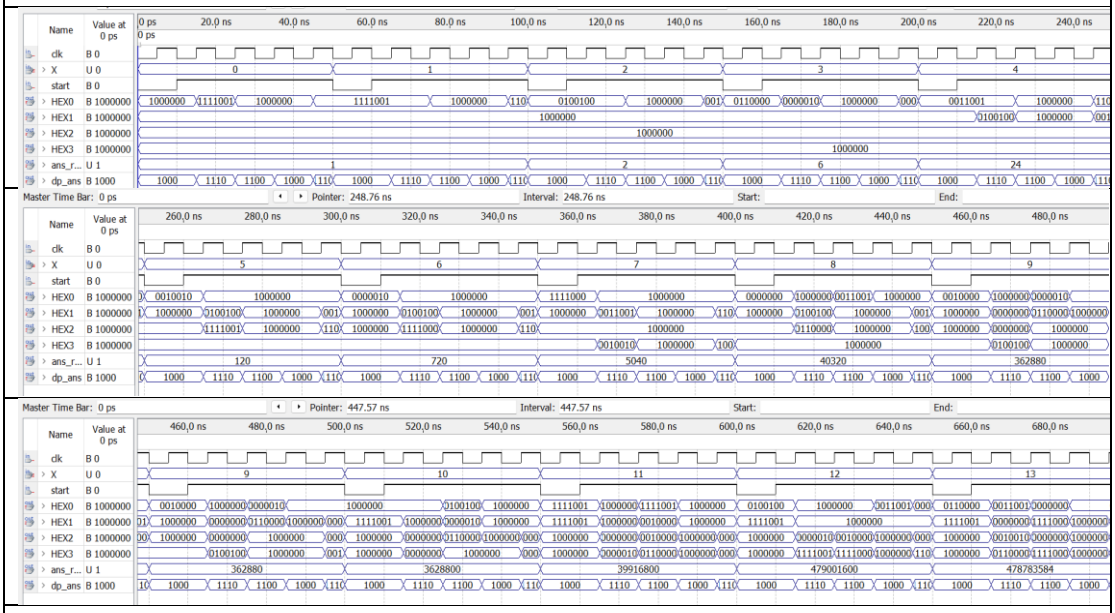
```
    -- process(start, clk_out):                                      -- 最後測試完再調時脈
    process(start, clk)
    begin
        if (start = '0') then                                        -- 輸入來源
            flag <= '1';
            dp <= "1000";
            show(0) <= conv_std_logic_vector(num1(0),4);
            show(1) <= conv_std_logic_vector(num1(1),4);
            show(2) <= conv_std_logic_vector(num1(2),4);
            show(3) <= conv_std_logic_vector(num1(3),4);
        elsif(start = '1' and flag = '1') then                       -- 開始輪播答案
            -- if (rising_edge(clk_out)) then                        -- 測試後調時脈
            if (rising_edge(clk)) then
                if (dp = "1000") then                                -- 共陽接角，亮一顆小點(最低四位結果)
                    show(0) <= conv_std_logic_vector(num(0),4);
                    show(1) <= conv_std_logic_vector(num(1),4);
                    show(2) <= conv_std_logic_vector(num(2),4);
                    show(3) <= conv_std_logic_vector(num(3),4);
                    dp <= "1110";                                    -- 迭代下一個輸出結果
                elsif(dp = "1110") then                              -- 共陽接角，亮兩顆小點(中間四位結果)
                    show(0) <= conv_std_logic_vector(num(4),4);
                    show(1) <= conv_std_logic_vector(num(5),4);
                    show(2) <= conv_std_logic_vector(num(6),4);
                    show(3) <= conv_std_logic_vector(num(7),4);
                    dp <= "1100";
                elsif(dp = "1100") then                              -- 共陽接角，亮三顆小點(最高四位結果)
                    show(0) <= conv_std_logic_vector(num(8),4);
                    show(1) <= conv_std_logic_vector(num(9),4);
                    show(2) <= conv_std_logic_vector(num(10),4);
                    show(3) <= conv_std_logic_vector(num(11),4);
                    dp <= "1000";
                end if;
            end if;
        end if;
    end process;
    dp_ans <= dp;
```

```

    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 behavior;
```

以上 code 為排除除頻器的操作，並將所有需要知道的結果拉出，以確保正確。

## 波形圖

參考檔案：fab_quick.vwf




從以上可以看出，除了數值的正確外，dp 也如本人期望的顯示。

## 心得

透過此次的實驗，我更清楚的了解到 vhdl 這語言同時進行的特性，在寫 code 的過程中撞到了許多次的 multiple assign error，雖然迄今為止，我還是沒有搞得很懂他到底幾個意思，反正我選擇繞路，如果我不能聰明的解決問題，那就用最噁心暴力的方式，如果解決不了問題，肯定是我還不夠暴力。