微算機系統

小組專案報告

期末專案：

CPU Pipeline實現

組別： 20

班級、姓名與學號：

醫工三 葉芸茜 B812110004

醫工三 湯青秀 B812110011

日期 :112年1月11日

一、實驗內容：

1. 所有功能皆須以pipeline實現才有分數
2. 總共有四個暫存器(R0,R1,R2,R3)，使用RS和RT分別指定要使用的暫存器(七段顯示器須同步更新)
3. 使用七段顯示器顯示 Rs、Rt 以及 Data 的值
4. Load/Move 是必要完成的功能，需實現所有功能本次實驗才有分數
5. 七段顯示器顯示為十六進制，兩顆七段顯示器為一組，範圍為1~FF（超過FF顯示末兩位）
6. 左邊兩組顯示 Rs 及 Rt 兩個暫存器(EX stage階段的Rs & Rt值)
7. 執行指令時，若暫存器的內容有被改變，則上述兩組七段顯示器須同步更新
8. 七段顯示器的顯示範圍為 0-FF
9. 在執行各 Cycle 時，須以綠色 LED 燈顯示該 Cycle 是否正在執行
10. 偵測 Data Hazard時，須以紅色 LED 燈顯示是否有 Hazard 產生

Variable	Pin Location	Signal Name
clk	PIN_M23	KEY[0]
output[0]	PIN_F19	LED[1]
output[1]	PIN_E19	LED[2]
output[2]	PIN_F21	LED[3]
data(0)	PIN_AB28	SW[0]
data(1)	PIN_AC28	SW[1]
data(2)	PIN_AC27	SW[2]
data(3)	PIN_AD27	SW[3]
data(4)	PIN_AB27	SW[4]
data(5)	PIN_AC26	SW[5]
data(6)	PIN_AD26	SW[6]
data(7)	PIN_AB26	SW[7]

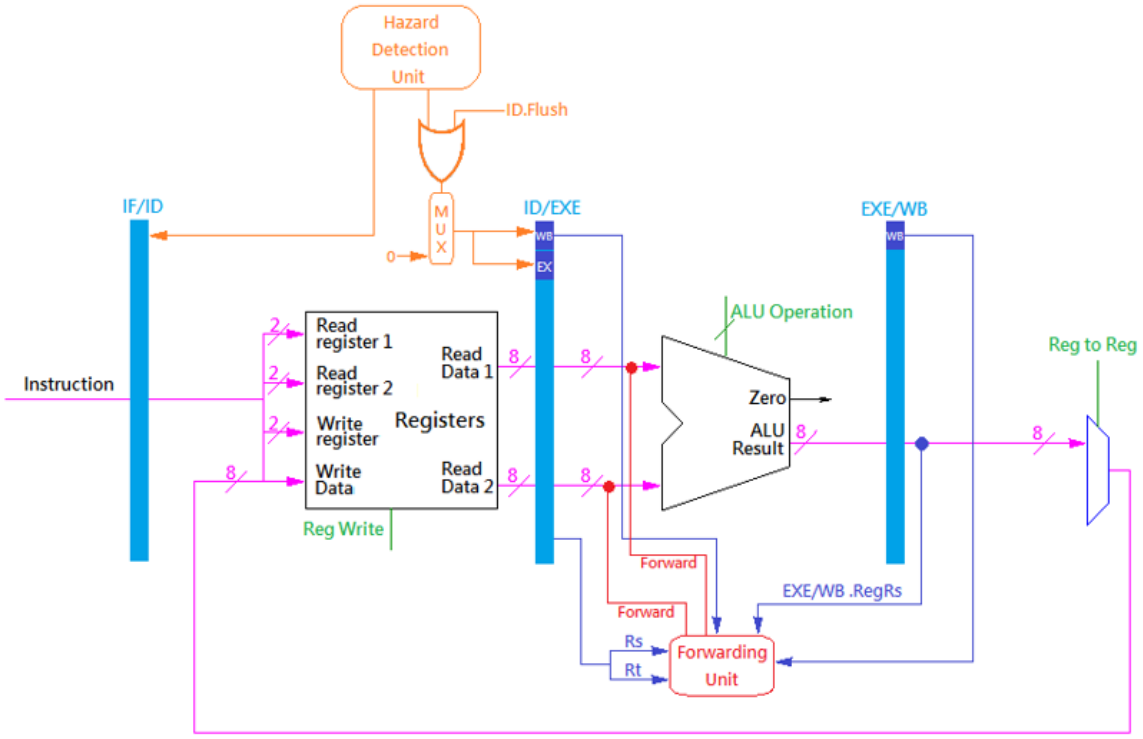
opcode(0)	PIN_AC25	SW[8]
opcode(1)	PIN_AB25	SW[9]
opcode(2)	PIN_AC24	SW[10]
opcode(3)	PIN_AB24	SW[11]
rs(0)	PIN_AB23	SW[12]
rs(1)	PIN_AA24	SW[13]
rt(0)	PIN_AA23	SW[14]
rt(1)	PIN_AA22	SW[15]
a0	PIN_G18	HEX0[0]
b0	PIN_F22	HEX0[1]
c0	PIN_E17	HEX0[2]
d0	PIN_L26	HEX0[3]
e0	PIN_L25	HEX0[4]
f0	PIN_J22	HEX0[5]
g0	PIN_H22	HEX0[6]
a1	PIN_M24	HEX1[0]
b1	PIN_Y22	HEX1[1]
c1	PIN_W21	HEX1[2]
d1	PIN_W22	HEX1[3]
e1	PIN_W25	HEX1[4]
f1	PIN_U23	HEX1[5]
g1	PIN_U24	HEX1[6]
a2	PIN_AA25	HEX2[0]
b2	PIN_AA26	HEX2[1]
c2	PIN_Y25	HEX2[2]
d2	PIN_W26	HEX2[3]

e2	PIN_Y26	HEX2[4]
f2	PIN_W27	HEX2[5]
g2	PIN_W28	HEX2[6]
a3	PIN_V21	HEX3[0]
b3	PIN_U21	HEX3[1]
c3	PIN_AB20	HEX3[2]
d3	PIN_AA21	HEX3[3]
e3	PIN_AD24	HEX3[4]
f3	PIN_AF23	HEX3[5]
g3	PIN_Y19	HEX3[6]
a4	PIN_V21	HEX4[0]
b4	PIN_U21	HEX4[1]
c4	PIN_AB20	HEX4[2]
d4	PIN_AA21	HEX4[3]
e4	PIN_AD24	HEX4[4]
f4	PIN_AF23	HEX4[5]
g4	PIN_Y19	HEX4[6]
a5	PIN_V21	HEX5[0]
b5	PIN_U21	HEX5[1]
c5	PIN_AB20	HEX5[2]
d5	PIN_AA21	HEX5[3]
e5	PIN_AD24	HEX5[4]
f5	PIN_AF23	HEX5[5]
g5	PIN_Y19	HEX5[6]
h1 (hazard LED1)	PIN_H21	LEDG[4]
h2 (hazard LED2)	PIN_G20	LEDG[5]

IF_LED	PIN_E21	LEDG[0]
ID_LED	PIN_E22	LEDG[1]
EX_LED	PIN_E25	LEDG[2]
WB_LED	PIN_E24	LEDG[3]
mux1_choose(0)	PIN_J17	LED[8]
mux1_choose(1)	PIN_G17	LED[9]
mux2_choose(0)	PIN_J15	LED[10]
mux2_choose(1)	PIN_H16	LED[11]
rs_choose(0)	PIN_J16	LED[12]
rs_choose(1)	PIN_H17	LED[13]
rt_choose(0)	PIN_F15	LED[14]
rt_choose(1)	PIN_G15	LED[15]
result(0)	PIN_G19	LED[0]
result(1)	PIN_F19	LED[1]
result(2)	PIN_E19	LED[2]
result(3)	PIN_F21	LED[3]
result(4)	PIN_F18	LED[4]
result(5)	PIN_E18	LED[5]
result(6)	PIN_J19	LED[6]
result(7)	PIN_H19	LED[7]

二、實驗過程及結果：

（一） 預期實驗結果的流程示意圖



Instruction	Instruction code			usage	Result
	opcode	Rs	Rt		
Load	0000	2bit	2bit	Load Rs Data	Rs←Data
Move	0001			Move Rs, Rt	Rs←Rt
Add	0010			Add Rs, Rt	Rs←Rs + Rt
Sub	0011			Sub Rs, Rt	Rs←Rs – Rt
And	0100			And Rs, Rt	Rs←Rs & Rt
Or	0101			Or Rs, Rt	Rs←Rs | Rt
Nor	0110			Nor Rs, Rt	Rs←Rs Nor Rt
Slt	0111			Slt Rs, Rt	if (Rs<Rt) Rs=1;else Rs=0
No instruction fetch	1111				No instruction fetch
Div	1000			Div Rs, Rt	Rs←Rs / Rt
Lw	1001			Lw Rs	Rs←Rs(Address)
Sw	1010			Sw Rs Rt	Rs→Rt(Address)

（二） 設計電路簡介

- 程式程序

  - IFetch(process):

    當該次輸入為nop則關閉LED，否則偵測前一次rs是否與該次rs rt有相同，再將rs, rt. data傳入IF暫存器，LED燈亮。

    將opcode傳入IF暫存器。

  - IDecode(process):

    先偵測輸入rs rt是否與前兩次的rs相同。

    再看IF暫存器中opcode是否為nop，是則關閉LED，否則LED亮，並依opcode將需要用的暫存器值(在r0, r1, r2, r3中)存入ID暫存器。

    將rs, rt. opcode傳入ID暫存器。

  - EXE(process):

    看ID暫存器中opcode是否為nop，是則關閉LED，否則LED亮，並依opcode將適當的值做為result存入EX暫存器。

    將rs, rt. opcode傳入EX暫存器。

  - WBack(process):

    看EX暫存器中opcode是否為nop，是則關閉LED，否則LED亮，EX中result, rs, rt放入WB，並依EX中的rs選擇將result放回該暫存器。

- **Hazard:**

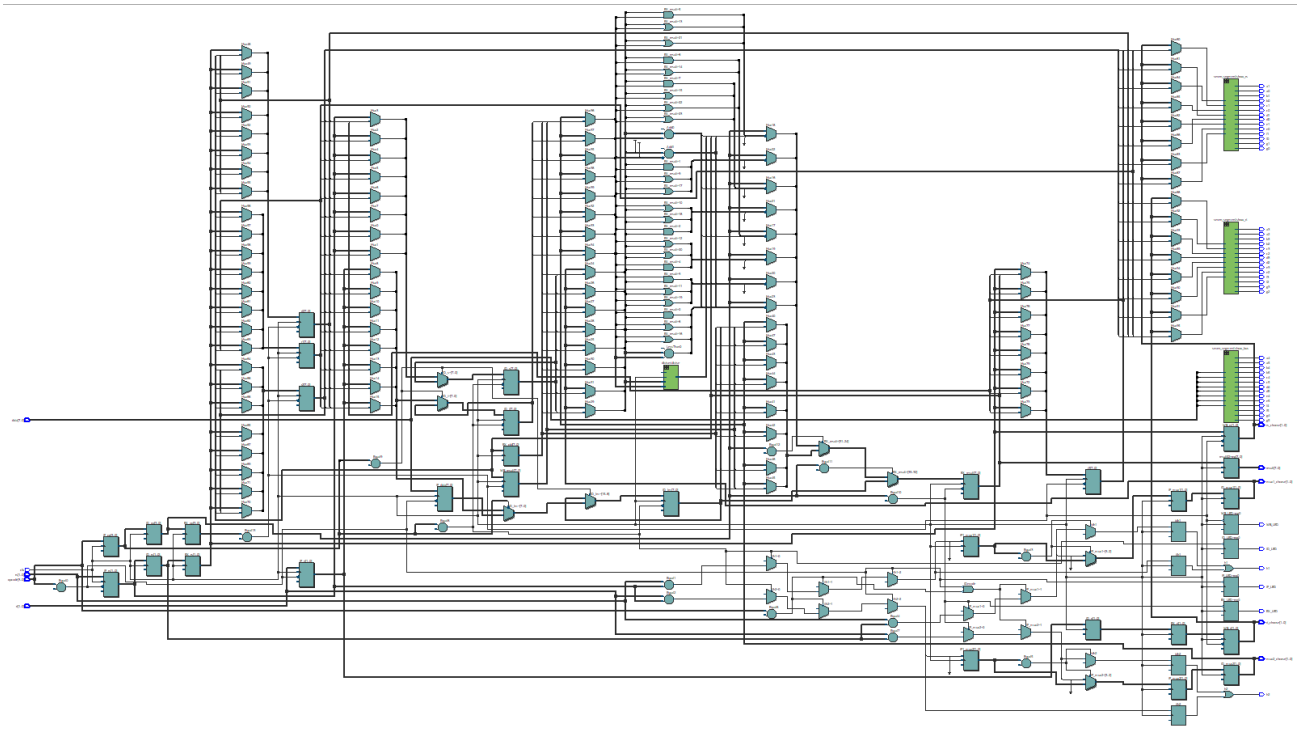
  根據偵測得到該條指令mux為00,01或10，對IF ID兩邊偵測得到是否有hazard發生的值(0或1)做OR得到LED控制數。

  依mux來選取要傳進EXE做運算的s_temp, t_temp和ID_move。

- 最終結果

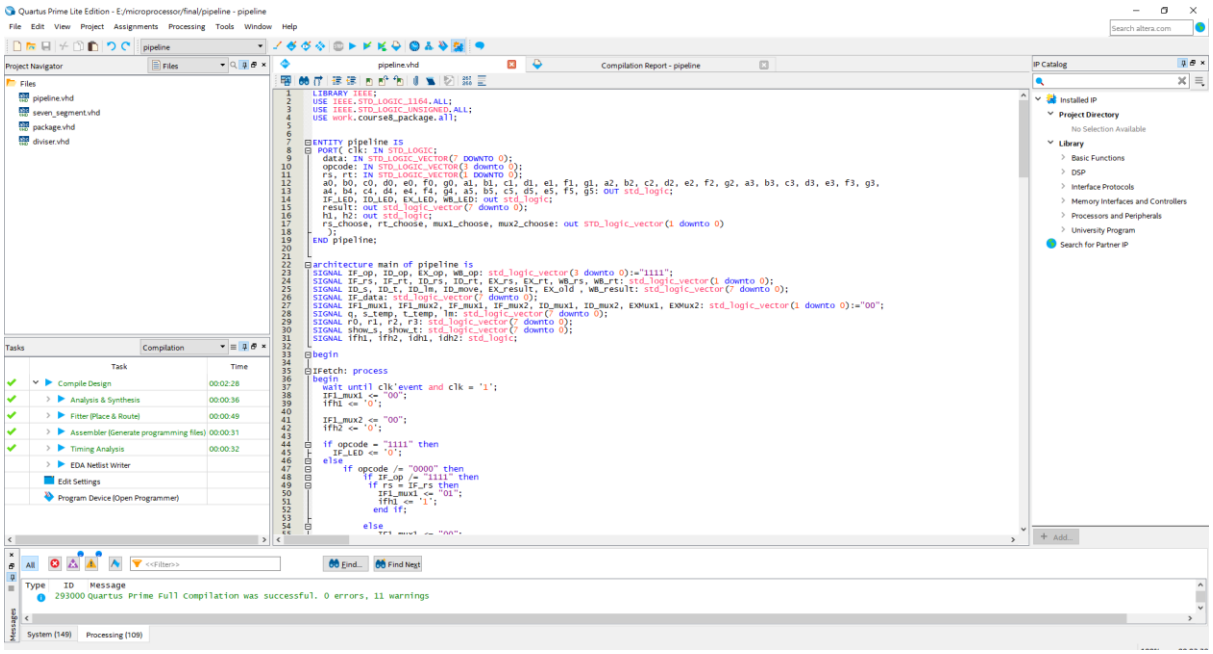
具有基本計算功能(除div)，且具有對write back stage的forwarding(與hazard之程序須間隔 1 nop)。

（三） 設計出的電路

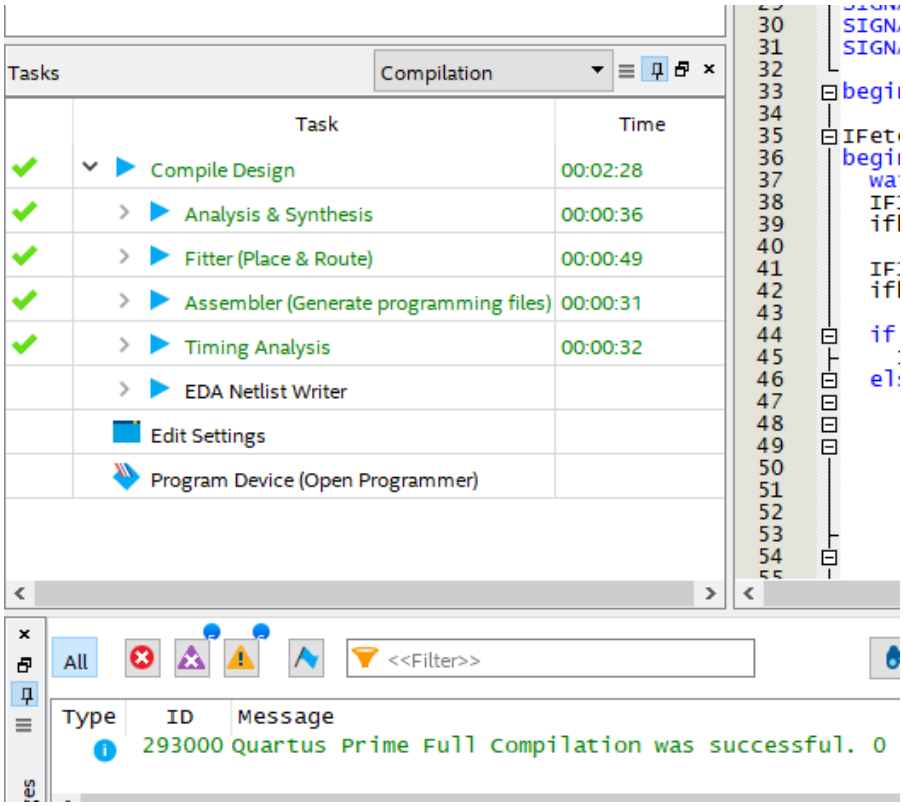


（四） 實驗過程

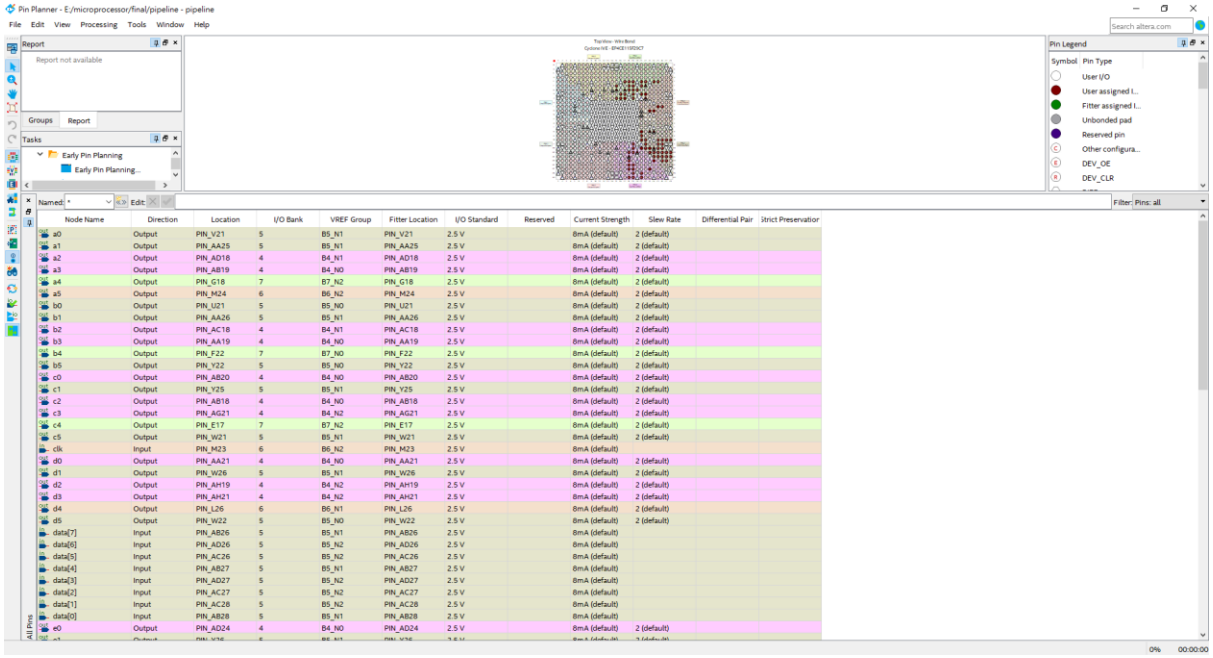
(1) 撰寫程式碼

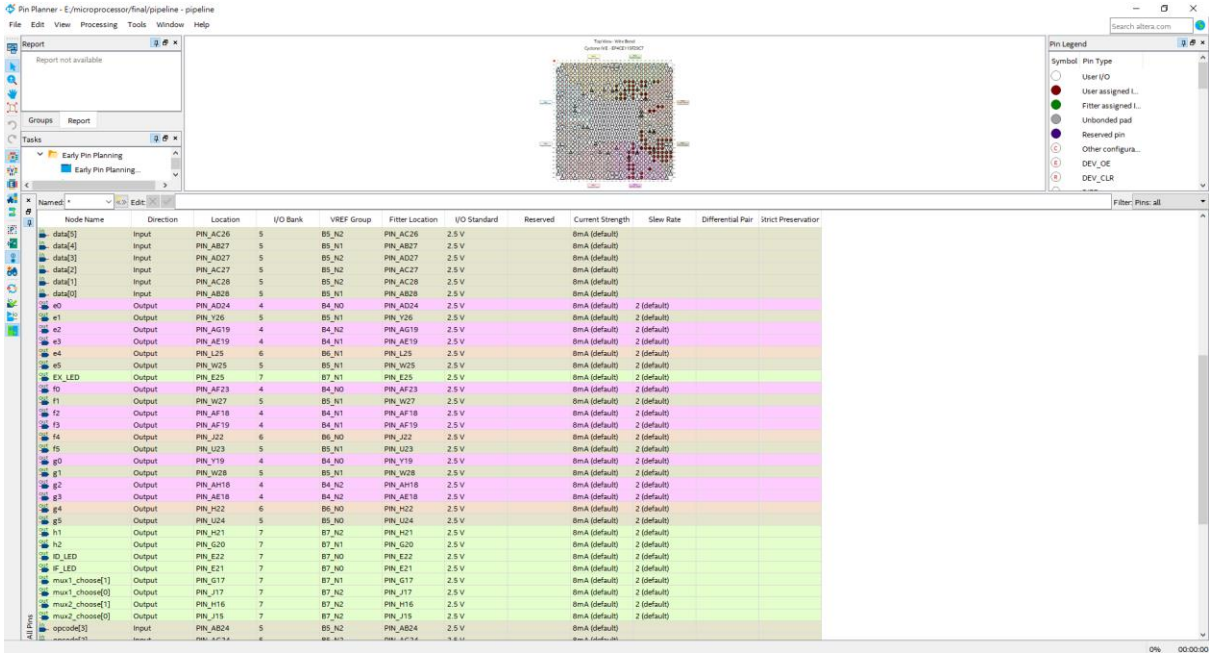


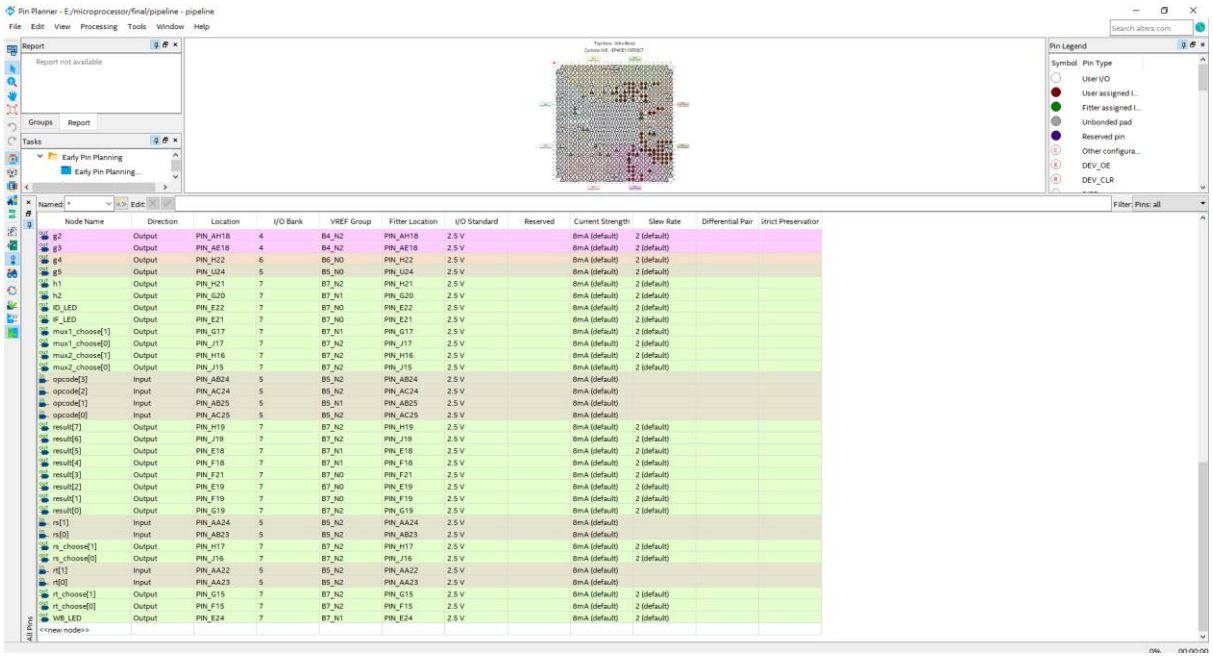
(2) 編譯成功



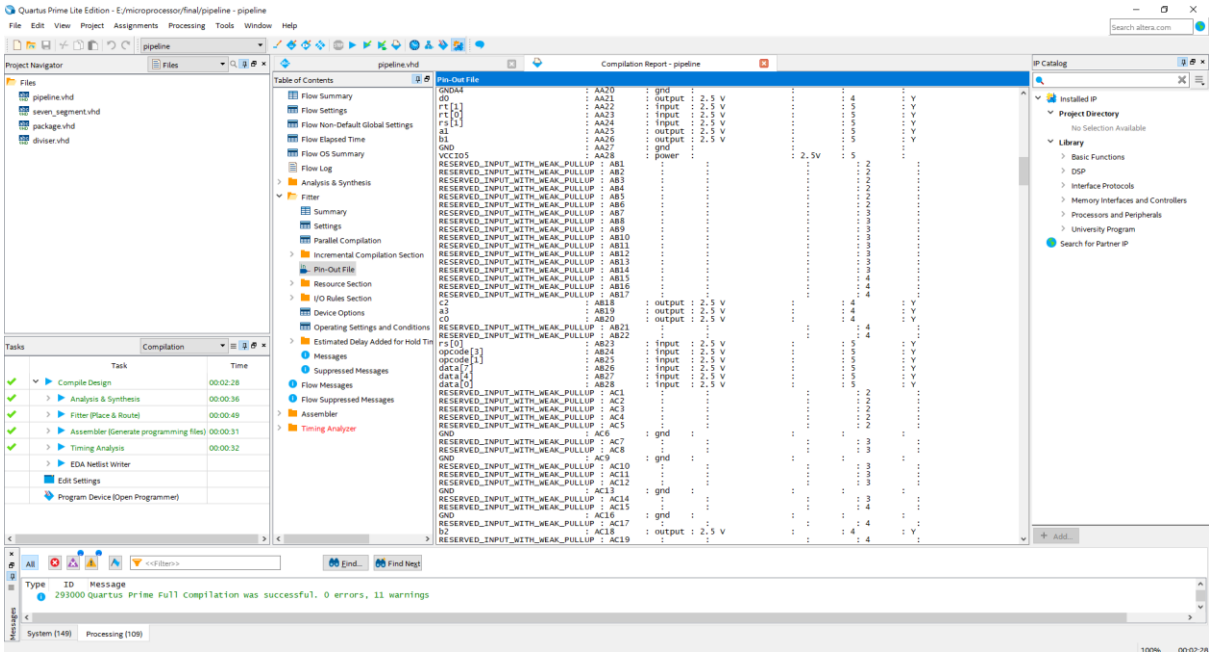
（3） 接脚位


Pin Planner - E:/microprocessor/final/pipeline - pipeline
File Edit View Processing Tools Window Help
Report
Report not available
Groups Report
Tasks
Early Pin Planning
Early Pin Planning...
Pin Legend
Symbol Pin Type
User I/O
User assigned I...
Fitter assigned I...
Unbonded pad
Reserved pin
Other configura...
DEV_OE
DEV_CLR
Named: * Edit:
Filter: Pins: all
Node Name Direction Location I/O Bank VREF Group Fitter Location I/O Standard Reserved Current Strength Slew Rate Differential Pair Strict Preservation
a0 Output PIN_V21 5 B5_N1 PIN_V21 2.5 V 8mA (default) 2 (default)
a1 Output PIN_AA25 5 B5_N1 PIN_AA25 2.5 V 8mA (default) 2 (default)
a2 Output PIN_AD18 4 B4_N1 PIN_AD18 2.5 V 8mA (default) 2 (default)
a3 Output PIN_AB19 4 B4_N0 PIN_AB19 2.5 V 8mA (default) 2 (default)
a4 Output PIN_G18 7 B7_N2 PIN_G18 2.5 V 8mA (default) 2 (default)
a5 Output PIN_M24 6 B6_N2 PIN_M24 2.5 V 8mA (default) 2 (default)
b0 Output PIN_U21 5 B5_N0 PIN_U21 2.5 V 8mA (default) 2 (default)
b1 Output PIN_AA26 5 B5_N1 PIN_AA26 2.5 V 8mA (default) 2 (default)
b2 Output PIN_AC18 4 B4_N1 PIN_AC18 2.5 V 8mA (default) 2 (default)
b3 Output PIN_AA19 4 B4_N0 PIN_AA19 2.5 V 8mA (default) 2 (default)
b4 Output PIN_F22 7 B7_N0 PIN_F22 2.5 V 8mA (default) 2 (default)
b5 Output PIN_Y22 5 B5_N0 PIN_Y22 2.5 V 8mA (default) 2 (default)
c0 Output PIN_AB20 4 B4_N0 PIN_AB20 2.5 V 8mA (default) 2 (default)
c1 Output PIN_Y25 5 B5_N1 PIN_Y25 2.5 V 8mA (default) 2 (default)
c2 Output PIN_AB18 4 B4_N0 PIN_AB18 2.5 V 8mA (default) 2 (default)
c3 Output PIN_AG21 4 B4_N2 PIN_AG21 2.5 V 8mA (default) 2 (default)
c4 Output PIN_E17 7 B7_N2 PIN_E17 2.5 V 8mA (default) 2 (default)
c5 Output PIN_W21 5 B5_N1 PIN_W21 2.5 V 8mA (default) 2 (default)
clk Input PIN_M23 6 B6_N2 PIN_M23 2.5 V 8mA (default)
d0 Output PIN_AA21 4 B4_N0 PIN_AA21 2.5 V 8mA (default) 2 (default)
d1 Output PIN_W26 5 B5_N1 PIN_W26 2.5 V 8mA (default) 2 (default)
d2 Output PIN_AH19 4 B4_N2 PIN_AH19 2.5 V 8mA (default) 2 (default)
d3 Output PIN_AH21 4 B4_N2 PIN_AH21 2.5 V 8mA (default) 2 (default)
d4 Output PIN_L26 6 B6_N1 PIN_L26 2.5 V 8mA (default) 2 (default)
d5 Output PIN_W22 5 B5_N0 PIN_W22 2.5 V 8mA (default) 2 (default)
data[7] Input PIN_AB26 5 B5_N1 PIN_AB26 2.5 V 8mA (default)
data[6] Input PIN_AD26 5 B5_N2 PIN_AD26 2.5 V 8mA (default)
data[5] Input PIN_AC26 5 B5_N2 PIN_AC26 2.5 V 8mA (default)
data[4] Input PIN_AB27 5 B5_N1 PIN_AB27 2.5 V 8mA (default)
data[3] Input PIN_AD27 5 B5_N2 PIN_AD27 2.5 V 8mA (default)
data[2] Input PIN_AC27 5 B5_N2 PIN_AC27 2.5 V 8mA (default)
data[1] Input PIN_AC28 5 B5_N2 PIN_AC28 2.5 V 8mA (default)
data[0] Input PIN_AB28 5 B5_N1 PIN_AB28 2.5 V 8mA (default)
e0 Output PIN_AD24 4 B4_N0 PIN_AD24 2.5 V 8mA (default) 2 (default)
All Pins
0% 00:00:00

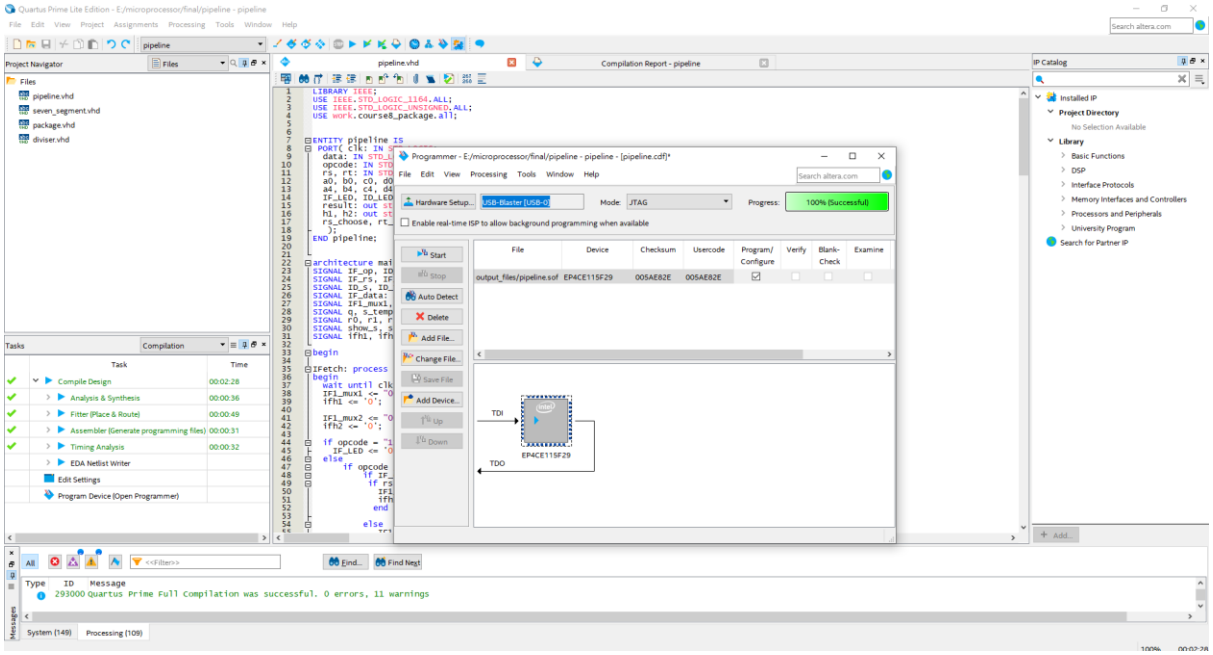

Pin Planner - E:/microprocessor/final/pipeline - pipeline
File Edit View Processing Tools Window Help
Report
Report not available
Groups Report
Tasks
Early Pin Planning
Early Pin Planning...
Pin Legend
Symbol Pin Type
User I/O
User assigned I...
Fitter assigned I...
Unbonded pad
Reserved pin
Other configura...
DEV_OE
DEV_CLR
Named: * Edit:
Filter: Pins: all
Node Name Direction Location I/O Bank VREF Group Fitter Location I/O Standard Reserved Current Strength Slew Rate Differential Pair Strict Preservation
data[5] Input PIN_AC26 5 B5_N2 PIN_AC26 2.5 V 8mA (default)
data[4] Input PIN_AB27 5 B5_N1 PIN_AB27 2.5 V 8mA (default)
data[3] Input PIN_AD27 5 B5_N2 PIN_AD27 2.5 V 8mA (default)
data[2] Input PIN_AC27 5 B5_N2 PIN_AC27 2.5 V 8mA (default)
data[1] Input PIN_AC28 5 B5_N2 PIN_AC28 2.5 V 8mA (default)
data[0] Input PIN_AB28 5 B5_N1 PIN_AB28 2.5 V 8mA (default)
e0 Output PIN_AD24 4 B4_N0 PIN_AD24 2.5 V 8mA (default) 2 (default)
e1 Output PIN_Y26 5 B5_N1 PIN_Y26 2.5 V 8mA (default) 2 (default)
e2 Output PIN_AG19 4 B4_N2 PIN_AG19 2.5 V 8mA (default) 2 (default)
e3 Output PIN_AE19 4 B4_N1 PIN_AE19 2.5 V 8mA (default) 2 (default)
e4 Output PIN_L25 6 B6_N1 PIN_L25 2.5 V 8mA (default) 2 (default)
e5 Output PIN_W25 5 B5_N1 PIN_W25 2.5 V 8mA (default) 2 (default)
EX_LED Output PIN_E25 7 B7_N1 PIN_E25 2.5 V 8mA (default) 2 (default)
f0 Output PIN_AF23 4 B4_N0 PIN_AF23 2.5 V 8mA (default) 2 (default)
f1 Output PIN_W27 5 B5_N1 PIN_W27 2.5 V 8mA (default) 2 (default)
f2 Output PIN_AF18 4 B4_N1 PIN_AF18 2.5 V 8mA (default) 2 (default)
f3 Output PIN_AF19 4 B4_N1 PIN_AF19 2.5 V 8mA (default) 2 (default)
f4 Output PIN_J22 6 B6_N0 PIN_J22 2.5 V 8mA (default) 2 (default)
f5 Output PIN_U23 5 B5_N1 PIN_U23 2.5 V 8mA (default) 2 (default)
g0 Output PIN_Y19 4 B4_N0 PIN_Y19 2.5 V 8mA (default) 2 (default)
g1 Output PIN_W28 5 B5_N1 PIN_W28 2.5 V 8mA (default) 2 (default)
g2 Output PIN_AH18 4 B4_N2 PIN_AH18 2.5 V 8mA (default) 2 (default)
g3 Output PIN_AE18 4 B4_N2 PIN_AE18 2.5 V 8mA (default) 2 (default)
g4 Output PIN_H22 6 B6_N0 PIN_H22 2.5 V 8mA (default) 2 (default)
g5 Output PIN_U24 5 B5_N0 PIN_U24 2.5 V 8mA (default) 2 (default)
h1 Output PIN_H21 7 B7_N2 PIN_H21 2.5 V 8mA (default) 2 (default)
h2 Output PIN_G20 7 B7_N1 PIN_G20 2.5 V 8mA (default) 2 (default)
ID_LED Output PIN_E22 7 B7_N0 PIN_E22 2.5 V 8mA (default) 2 (default)
IF_LED Output PIN_E21 7 B7_N0 PIN_E21 2.5 V 8mA (default) 2 (default)
mux1_choose[1] Output PIN_G17 7 B7_N1 PIN_G17 2.5 V 8mA (default) 2 (default)
mux1_choose[0] Output PIN_J17 7 B7_N2 PIN_J17 2.5 V 8mA (default) 2 (default)
mux2_choose[1] Output PIN_H16 7 B7_N2 PIN_H16 2.5 V 8mA (default) 2 (default)
mux2_choose[0] Output PIN_J15 7 B7_N2 PIN_J15 2.5 V 8mA (default) 2 (default)
opcode[3] Input PIN_AB24 5 B5_N2 PIN_AB24 2.5 V 8mA (default)
All Pins
0% 00:00:00



(4)　確認接線於正確腳位



(5)　燒錄視窗設定及燒錄成功畫面



（五） 實驗結果

● 實際操作影片連結:

測資1:

https://drive.google.com/file/d/1k-cTR-kuigpimAvi5fCPxrvUd_UYQfzn/view?usp=sharing

測資2:

https://drive.google.com/file/d/19S_wXKF1PvW0hx9fp7W51sUD5ojyO2xB/view?usp=sharing

測資3:

https://drive.google.com/file/d/10me9L5hqGVL7PSYj_FkU85_dH3ZxKeDa/view?usp=sharing

三、程式碼

pipeline.vhd

```
LIBRARY IEEE;

USE IEEE.STD_LOGIC_1164.ALL;

USE IEEE.STD_LOGIC_UNSIGNED.ALL;

USE work.course8_package.all;




ENTITY pipeline IS

 PORT( clk: IN STD_LOGIC;

  data: IN STD_LOGIC_VECTOR(7 DOWNTO 0);

  opcode: IN STD_LOGIC_VECTOR(3 downto 0);

  rs, rt: IN STD_LOGIC_VECTOR(1 DOWNTO 0);

  a0, b0, c0, d0, e0, f0, g0, a1, b1, c1, d1, e1, f1, g1, a2, b2, c2, d2, e2, f2, g2, a3, b3, c3, d3, e3, f
3, g3,

  a4, b4, c4, d4, e4, f4, g4, a5, b5, c5, d5, e5, f5, g5: OUT std_logic;

  IF_LED, ID_LED, EX_LED, WB_LED: out std_logic;

  result: out std_logic_vector(7 downto 0);

  h1, h2: out std_logic;
```

```
  rs_choose, rt_choose, mux1_choose, mux2_choose: out STD_logic_vector(1 downto 0)

   );

END pipeline;



architecture main of pipeline is

SIGNAL IF_op, ID_op, EX_op, WB_op: std_logic_vector(3 downto 0):="1111";

SIGNAL IF_rs, IF_rt, ID_rs, ID_rt, EX_rs, EX_rt, WB_rs, WB_rt: std_logic_vector(1 downto 0);

SIGNAL ID_s, ID_t, ID_lm, ID_move, EX_result, EX_old , WB_result: std_logic_vector(7 downt
o 0);

SIGNAL IF_data: std_logic_vector(7 downto 0);

SIGNAL IF1_mux1, IF1_mux2, IF_mux1, IF_mux2, ID_mux1, ID_mux2, EXMux1, EXMux2: st
d_logic_vector(1 downto 0):="00";

SIGNAL q, s_temp, t_temp, lm: std_logic_vector(7 downto 0);

SIGNAL r0, r1, r2, r3: std_logic_vector(7 downto 0);

SIGNAL show_s, show_t: std_logic_vector(7 downto 0);

SIGNAL ifh1, ifh2, idh1, idh2: std_logic;



begin
```

```
IFetch: process
begin
  wait until clk'event and clk = '1';
  IF1_mux1 <= "00";
  ifh1 <= '0';

  IF1_mux2 <= "00";
  ifh2 <= '0';

  if opcode = "1111" then
    IF_LED <= '0';
  else
                    if opcode /= "0000" then
                              if IF_op /= "1111" then
                               if rs = IF_rs then
                                        IF1_mux1 <= "01";
                                        ifh1 <= '1';
                                       end if;
```

```
                else
                        IF1_mux1 <= "00";
                        ifh1 <= '0';
                end if;

                if IF_op /= "1111" then
                        if rt = IF_rs then
                         IF1_mux2 <= "01";
                         ifh2 <= '1';
                        end if;

                        else
                                IF1_mux2 <= "00";
                                ifh2 <= '0';
        end if;
end if;
IF_rs <= rs;
IF_rt <= rt;
IF_data <= data;
```

```
    IF_LED <= '1';
  end if;
  IF_op <= opcode;
end process;



IDecode: process
begin
  wait until clk'event and clk = '1';
  ID_mux1 <= "00";
  idh1 <= '0';


  ID_mux2 <= "00";
  idh2 <= '0';
  ID_mux1 <= IF_mux1;
          ID_mux2 <= IF_mux2;
          IF_mux1 <= "00";
          IF_mux2 <= "00";
```

```
If IF1_mux1 = "01" then
IF_mux1 <= IF1_mux1;
else
if opcode /= "0000" and opcode /= "1111" then
      if ID_op /= "1111" then
        if rs = ID_rs then
         IF_mux1 <= "10";
         idh1 <= '1';
        end if;

      else
        IF_mux1 <= "00";
        idh1 <= '0';
      end if;
          end if;
End if;

If IF1_mux2 = "01" then
IF_mux2 <= IF1_mux2;
```

```
else
if opcode /= "0000" and opcode /= "1111" then
        if ID_op /= "1111" then
          if rt = ID_rs then
           IF_mux2 <= "10";
           idh2 <= '1';
          end if;
        else
          IF_mux2 <= "00";
          idh2 <= '0';
    end if;
End if;
    end if;


  if IF_op = "1111" then
    ID_LED <= '0';


  else
    ID_LED <= '1';
```

```
if IF_op(3 downto 1) = "000" then
  case IF_op(0) is
    when '0' =>
      ID_lm <= IF_data;
    when '1' =>
      case IF_rt is
        when "00" =>
          ID_lm <= r0;
        when "01" =>
          ID_lm <= r1;
        when "10" =>
          ID_lm <= r2;
        when "11" =>
          ID_lm <= r3;
      end case;
  end case;
else
```

```
case IF_rs is
  when "00" =>
    ID_s <= r0;
  when "01" =>
    ID_s <= r1;
  when "10" =>
    ID_s <= r2;
  when "11" =>
    ID_s <= r3;
end case;

case IF_rt is
  when "00" =>
    ID_t <= r0;
  when "01" =>
    ID_t <= r1;
  when "10" =>
    ID_t <= r2;
  when "11" =>
```

```
          ID_t <= r3;
      end case;
    end if;
  end if;
    ID_op <= IF_op;
    ID_rs <= IF_rs;
    ID_rt <= IF_rt;
end process;

h1 <= ifh1 or idh1;
h2 <= ifh2 or idh2;

divise: diviser PORT MAP(clk,'0', t_temp, s_temp, q);

EXE:process
begin
      wait until clk'event and clk = '1';

      if ID_op = "1111" then
```

```
        EX_LED <= '0';


    else
        EX_LED <= '1';



        if ID_op= "0000" then
           EX_result <= ID_lm;
        elsif ID_op = "0001" then
                                        EX_result <= ID_move;


        else
             case ID_op is
                    when "0010" =>
                                                                    EX_result
<= s_temp + t_temp;
                    when "0011" =>
                                                                    EX_result
<= s_temp - t_temp;
                    when "0100" =>
```

```
                                                                    EX_result
<= s_temp and t_temp;
                    when "0101" =>
                                                                    EX_result
<= s_temp or t_temp;
                    when "0110" =>
                                                                    EX_result
<= s_temp nor t_temp;
                    when "0111" =>
                        if   s_temp < t_temp then
                            EX_result <= "00000001";
                        else
                            EX_result <= "00000000";
                        end if;
                    when "1000" =>
                                                                    EX_result
<= q;
                    when others =>


                end case;
            end if;
```

```
    end if;
    result <= EX_result;
                EX_old <= EX_result;
    EX_op <= ID_op;
    EX_rs <= ID_rs;
    EX_rt <= ID_rt;
end process;

EXMux1 <= ID_mux1;
EXMux2 <= ID_mux2;

with EXMux1 select
  s_temp <= ID_s when "00",
  EX_old when "01",
  WB_result when "10",
  ID_s when others;

with EXMux2 select
  t_temp <= ID_t when "00",
```

```
  EX_old when "01",
  WB_result when "10",
  ID_t when others;

with EXMux2 select
  ID_move <= ID_lm when "00",
  EX_old when "01",
  WB_result when "10",
  ID_lm when others;

WBack:process
begin
      wait until clk'event and clk = '1';
      if EX_op = "1111" then
        WB_LED <= '0';
      else
                  WB_result <= EX_result;
      WB_rs <= EX_rs;
      WB_rt <= EX_rt;
```

```
        WB_LED <= '1';
        case EX_rs is
            when "00" =>
                r0 <= EX_result;
            when "01" =>
                r1 <= EX_result;
            when "10" =>
                r2 <= EX_result;
            when "11" =>
                r3 <= EX_result;
        end case;
    end if;
end process;


show_select: with WB_rs select
                                        show_s <= r0 when "00",
                                                  r1 when "01",
                                                  r2 when "10",
                                                  r3 when "11";
```

```
                    with WB_rt select
                    show_t <= r0 when "00",
                                r1 when "01",
                                r2 when "10",
                                r3 when "11";
rs_choose <= WB_rs;
rt_choose <= WB_rt;
mux1_choose <= EXMux1;
mux2_choose <= ExMux2;

show_rs: seven_segment PORT MAP(show_s(3), show_s(2), show_s(1), show_s(0), show_s(7), s
how_s(6), show_s(5), show_s(4), a1, b1, c1, d1, e1, f1, g1, a0, b0, c0, d0, e0, f0, g0);
show_rt: seven_segment PORT MAP(show_t(3), show_t(2), show_t(1), show_t(0), show_t(7), sho
w_t(6), show_t(5), show_t(4), a3, b3, c3, d3, e3, f3, g3, a2, b2, c2, d2, e2, f2, g2);
show_bus: seven_segment PORT MAP(data(3), data(2), data(1), data(0), data(7), data(6), data(5),
data(4), a4, b4, c4, d4, e4, f4, g4, a5, b5, c5, d5, e5, f5, g5);
end main;
```

package. vhd

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

PACKAGE course8_package IS

        component seven_segment
                PORT( W0, X0, Y0, Z0,
                                W1, X1, Y1, Z1: IN STD_LOGIC;
                                a0, b0, c0, d0, e0, f0, g0,
                                a1, b1, c1, d1, e1, f1, g1: OUT STD_LOGIC);
        END component seven_segment;

        component diviser
 PORT( clk, clear: IN STD_LOGIC;
   divisor, dividend: IN STD_LOGIC_VECTOR(7 downto 0);
   q : out STD_LOGIC_VECTOR(7 downto 0)
   );
END component diviser;
```

```
END course8_package;
```

seven_segment.vhd

```
Library ieee;
USE ieee.std_logic_1164.all;

ENTITY seven_segment IS
        PORT( W0, X0, Y0, Z0,
                W1, X1, Y1, Z1      : IN STD_LOGIC;
            a0, b0, c0, d0, e0, f0, g0,
                a1, b1, c1, d1, e1, f1, g1: OUT STD_LOGIC);
END seven_segment;

ARCHITECTURE LogicFunc OF seven_segment IS
BEGIN
        a0 <= (NOT W0 AND NOT X0 AND NOT Y0 AND Z0)
                OR (NOT W0 AND X0 AND NOT Y0 AND NOT Z0)
                OR (W0 AND NOT X0 AND Y0 AND Z0)
                OR (W0 AND X0 AND NOT Y0);
        b0 <= (NOT W0 AND X0 AND NOT Y0 AND Z0)
                OR (NOT W0 AND X0 AND Y0 AND NOT Z0)
                OR (W0 AND NOT X0 AND Y0 AND Z0)
```

```
                OR (W0 AND X0 AND NOT Z0)

                OR (W0 AND X0 AND Y0);

c0 <= (NOT W0 AND NOT X0 AND Y0 AND NOT Z0)

                OR (W0 AND X0 AND NOT Z0)

                OR (W0 AND X0 AND Y0);

d0 <= (NOT X0 AND NOT Y0 AND Z0)

                OR (NOT W0 AND X0 AND NOT Y0 AND NOT Z0)

                OR (X0 AND Y0 AND Z0)

                OR (W0 AND NOT X0 AND Y0 AND NOT Z0);

e0 <= (NOT W0 AND Z0)

                OR (NOT W0 AND X0 AND NOT Y0)

                OR (NOT X0 AND NOT Y0 AND Z0);

f0 <= (NOT W0 AND NOT X0 AND Z0)

                OR (NOT W0 AND NOT X0 AND Y0)

                OR (NOT W0 AND Y0 AND Z0)

                OR (W0 AND X0 AND NOT Y0);

g0 <= (NOT W0 AND NOT X0 AND NOT Y0)

                OR (NOT W0 AND X0 AND Y0 AND Z0);
```

```
a1 <= (NOT W1 AND NOT X1 AND NOT Y1 AND Z1)
        OR (NOT W1 AND X1 AND NOT Y1 AND NOT Z1)
        OR (W1 AND NOT X1 AND Y1 AND Z1)
        OR (W1 AND X1 AND NOT Y1);
b1 <= (NOT W1 AND X1 AND NOT Y1 AND Z1)
        OR (NOT W1 AND X1 AND Y1 AND NOT Z1)
        OR (W1 AND NOT X1 AND Y1 AND Z1)
        OR (W1 AND X1 AND NOT Z1)
        OR (W1 AND X1 AND Y1);
c1 <= (NOT W1 AND NOT X1 AND Y1 AND NOT Z1)
        OR (W1 AND X1 AND NOT Z1)
        OR (W1 AND X1 AND Y1);
d1 <= (NOT X1 AND NOT Y1 AND Z1)
        OR (NOT W1 AND X1 AND NOT Y1 AND NOT Z1)
        OR (X1 AND Y1 AND Z1)
        OR (W1 AND NOT X1 AND Y1 AND NOT Z1);
e1 <= (NOT W1 AND Z1)
        OR (NOT W1 AND X1 AND NOT Y1)
        OR (NOT X1 AND NOT Y1 AND Z1);
```

```
        f1 <= (NOT W1 AND NOT X1 AND Z1)
                        OR (NOT W1 AND NOT X1 AND Y1)
                        OR (NOT W1 AND Y1 AND Z1)
                        OR (W1 AND X1 AND NOT Y1);
        g1 <= (NOT W1 AND NOT X1 AND NOT Y1)
                        OR (NOT W1 AND X1 AND Y1 AND Z1);
END LogicFunc;
```

diviser.vhd

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;



ENTITY diviser IS
 PORT( clk,clear: IN STD_LOGIC;
   divisor, dividend: IN STD_LOGIC_VECTOR(7 downto 0);
   q : out STD_LOGIC_VECTOR(7 downto 0)
   );
END diviser;



architecture main of diviser is
 signal w : STD_LOGIC :='0';
 signal re : INTEGER:= 0;
 signal run : std_logic;
 signal temp_s, temp_p: STD_LOGIC_VECTOR(15 downto 0);
```

```
signal output : STD_LOGIC_VEcTOR(2 downto 0);

signal remainder: STD_LOGIC_VECTOR(15 DOWNTO 0);



begin




 PROCESS

 begin

  WAIT UNTIL Clk'EVENT AND clk = '1';

  if clear = '1' then

   output <= "100";

  else

   Case output is

    when "000" =>

     output <= "111";



    when "111" =>

     output <= "001";
```

```
when "001" =>
 if w = '1' then
output <= "011";
 else
output <= "010";
 end if;

when "011" =>
 if w = '1' then
output <= "101";
 else
output <= "001";
 end if;

when "010" =>
 if w = '1' then
output <= "101";
 else
output <= "001";
```

```
    end if;


   when "100"=>

   output <= "000";


   when others =>

   output <= "110";


   end Case;

 end if;

end PROCESS;
```

```
case3: temp_s <= (( remainder(15 downto 8) - divisor) & remainder(7 downto 0));


case4: temp_p <= (( remainder(15 downto 8) + divisor) & remainder(7 downto 0));
```

```
output_select: process
begin
 WAIT UNTIL Clk'EVENT AND clk = '1';
 case output is
  when "000" =>
  remainder <= ("00000000" & dividend);

  when "111" =>
   for_1: FOR i IN 14 downto 0 loop
       remainder(i + 1) <= remainder(i);
    END loop;
    remainder(0) <= '0';

  when "001" =>
   remainder <= (( remainder(15 downto 8) - divisor) & remainder(7 downto 0));

  when "011" =>
   for_2: FOR i IN 14 downto 0 loop
      remainder(i + 1) <= remainder(i);
```

```
  END loop;
  remainder(0) <= '1';

when "010" =>
 for_3: FOR i IN 14 downto 0 loop
    remainder(i + 1) <= temp_p(i);
  END loop;
  remainder(0) <= '0';

when "100"=>
remainder <= "0000000000000000";

when "101" =>
for_r: FOR i IN 8 to 14 loop
    remainder(i) <= remainder(i + 1);
  END loop;
  remainder(15) <= '0';
remainder(7 downto 0) <= remainder(7 downto 0);
```

```
   when others =>
   remainder <= remainder;


 end Case;
end process;


count:process
begin
 WAIT UNTIL Clk'EVENT AND clk = '1';
 if clear = '1' then
  re <= 0;
 elsif output = "001" and run = '1' then
  re <= re + 1;
 else
  re <= re;
 end if;
end process;
run <= '0' when re = 8 else '1';
```

```
 with output select
   w <= not temp_s(15) when "001",
    not run when "011",
    not run when "010",
    '0' when others;

q <= remainder(7 downto 0);
end main;
```