**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

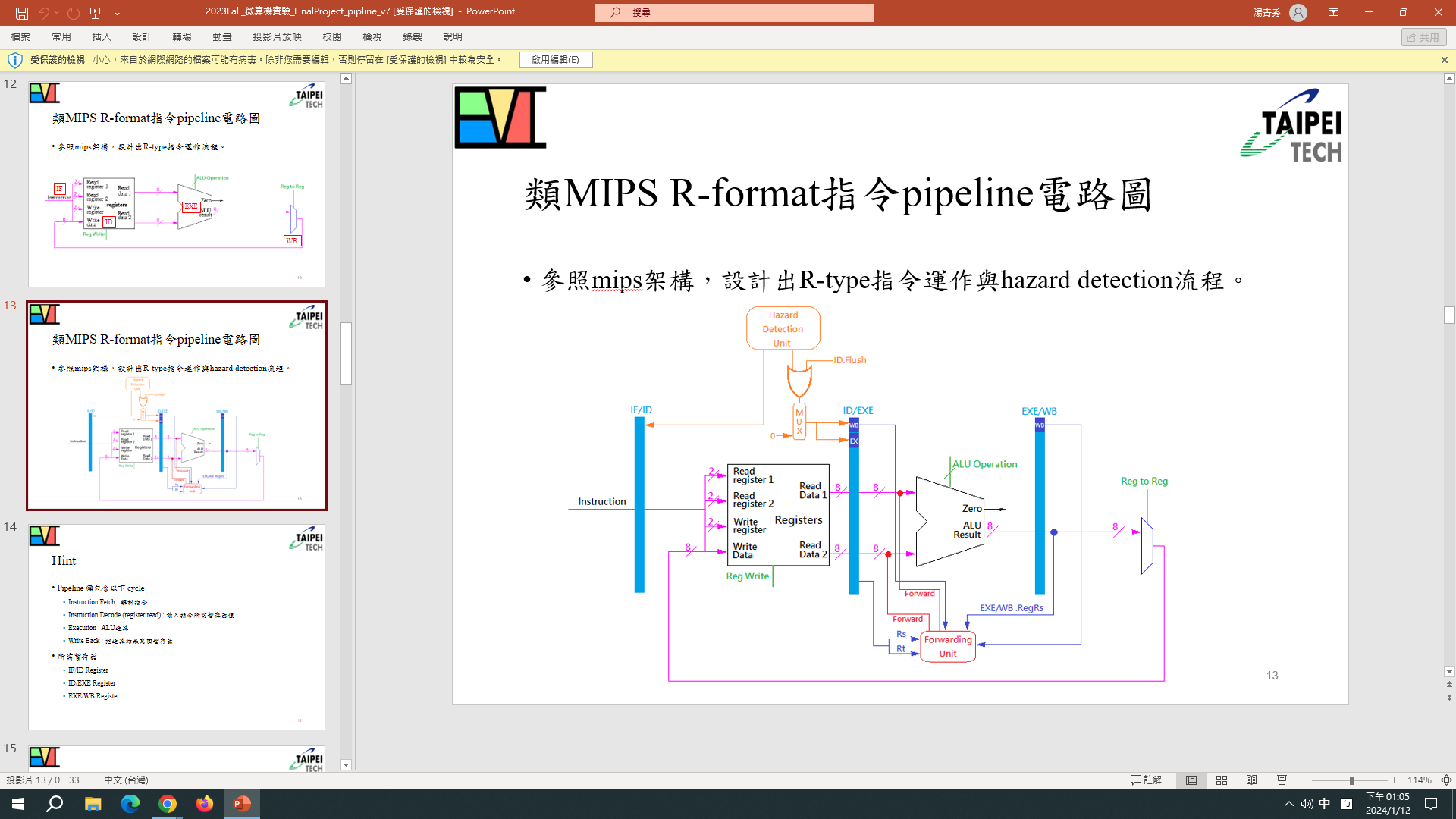
一、實驗內容：

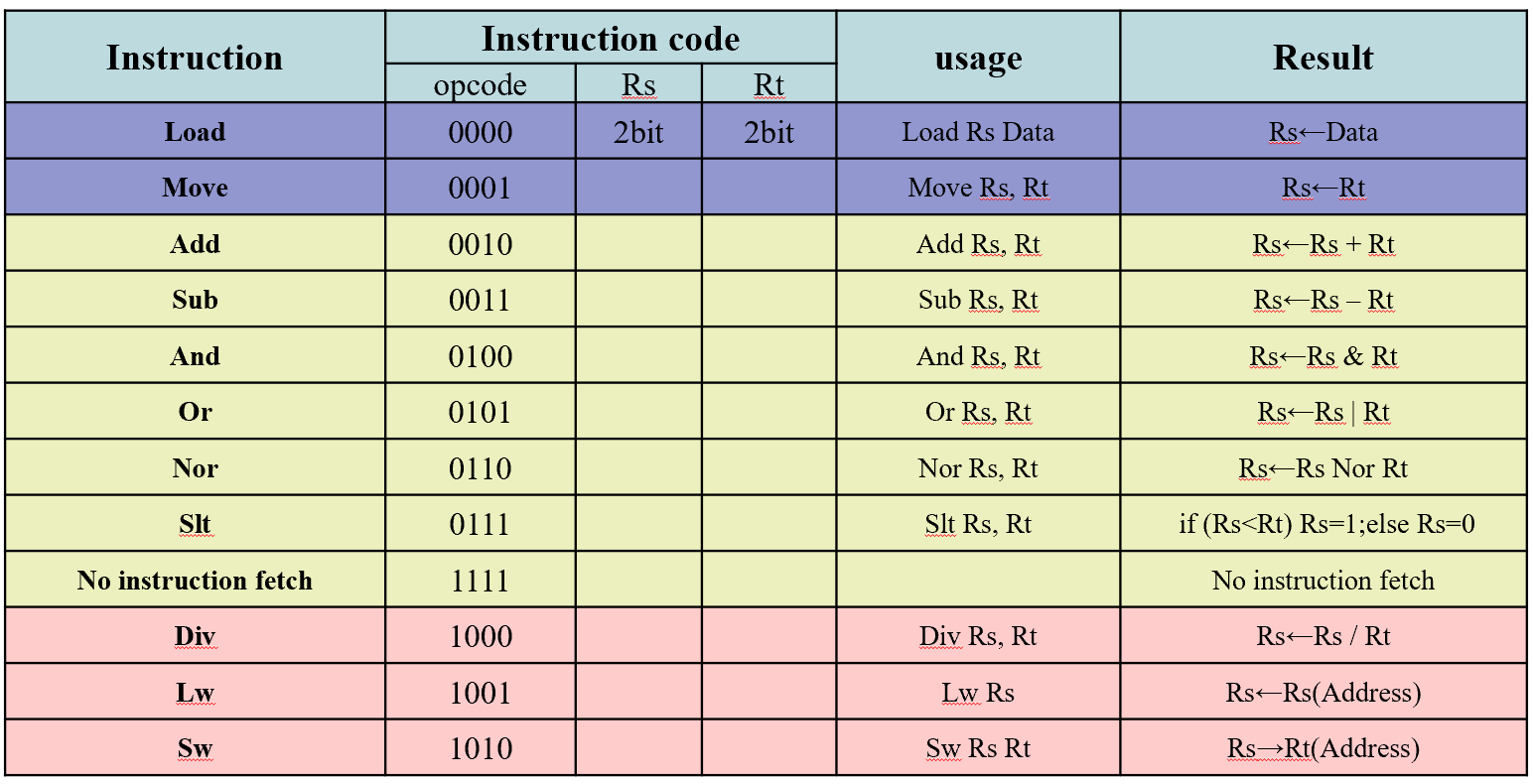
1. 所有功能皆須以pipeline實現才有分數
2. 總共有四個暫存器(R0,R1,R2,R3)，使用RS和RT分別指定要使用的暫存器(七段顯示器須同步更新)
3. 使用七段顯示器顯示 Rs、Rt 以及 Data 的值
4. Load/Move 是必要完成的功能，需實現所有功能本次實驗才有分數
5. 七段顯示器顯示為十六進制，兩顆七段顯示器為一組，範圍為1~FF (超過FF顯示末兩位)
6. 左邊兩組顯示 Rs 及 Rt 兩個暫存器(EX stage階段的Rs & Rt值 )
7. 執行指令時，若暫存器的內容有被改變，則上述兩組七段顯示器須同步更新
8. 七段顯示器的顯示範圍為 0-FF
9.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





（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:**

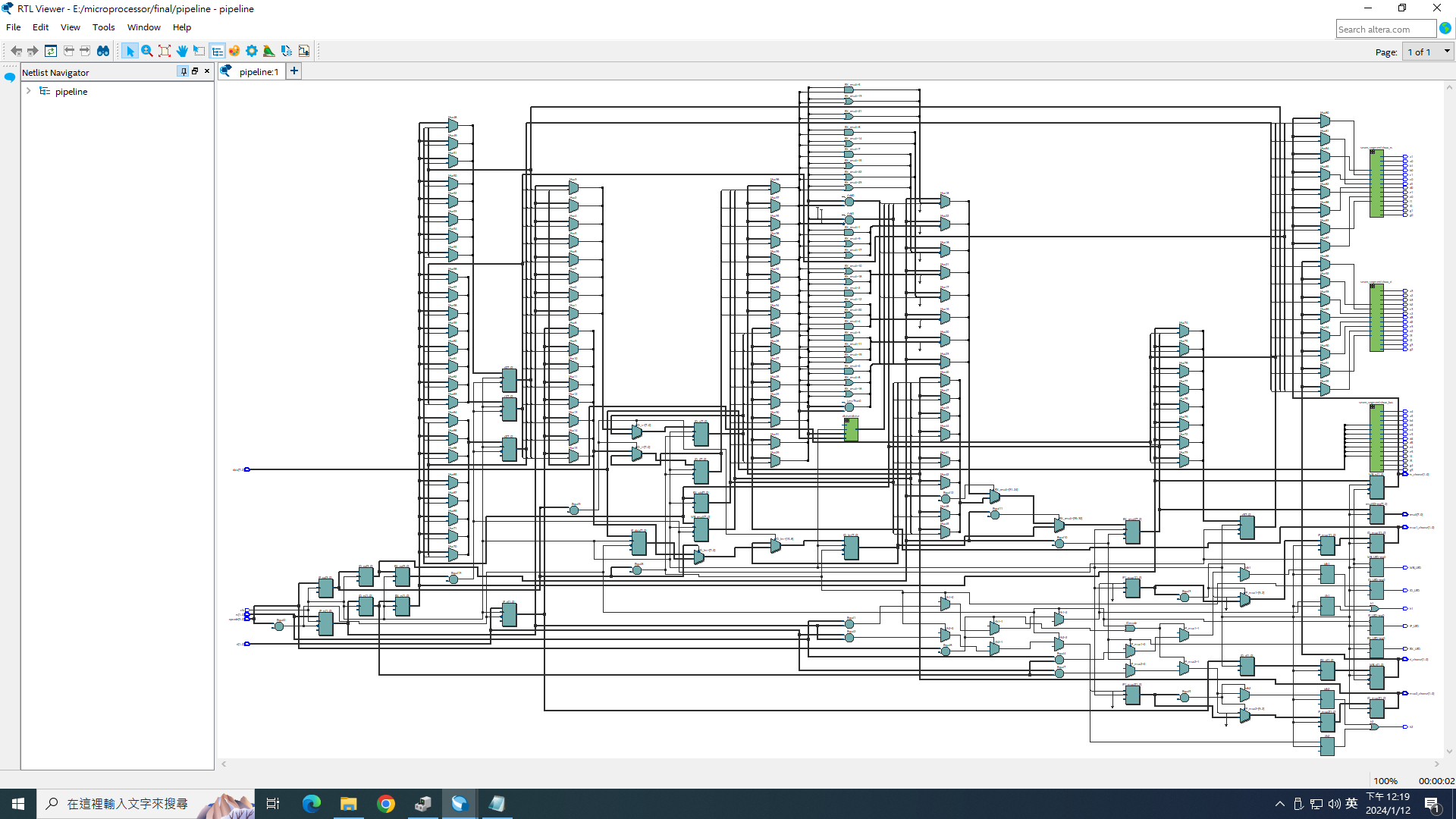
根據偵測得到該條指令mux為00,01或10，對IF ID兩邊偵測得到是否有hazard發生的值(0或1)做OR得到LED控制數。

依mux來選取要傳進EXE做運算的s\_temp, t\_temp和ID\_move。

* 最終結果

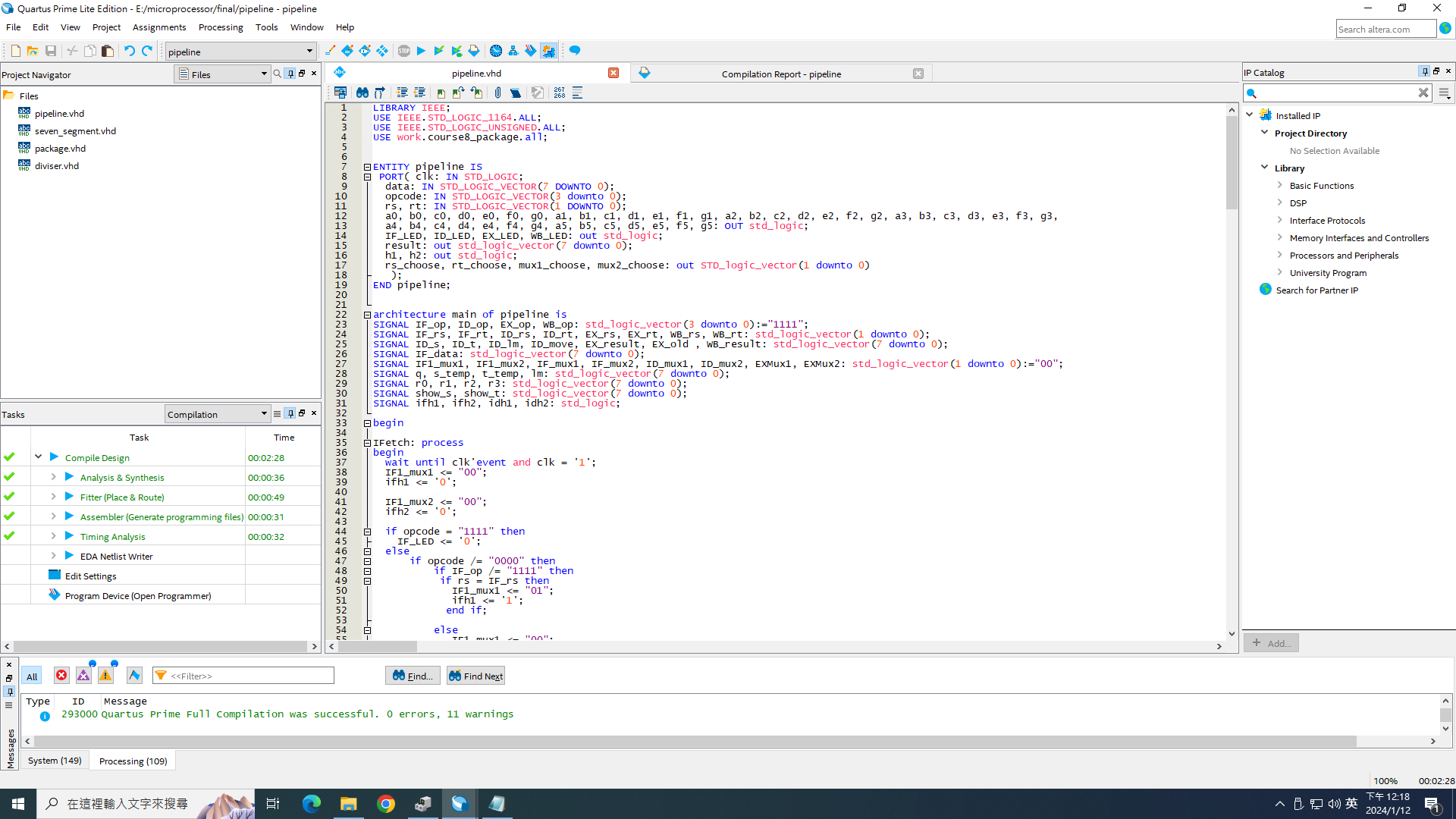
具有基本計算功能(除div)，且具有對write back stage的forwarding(與hazard之程序須間隔 1 nop)。

（三） 設計出的電路

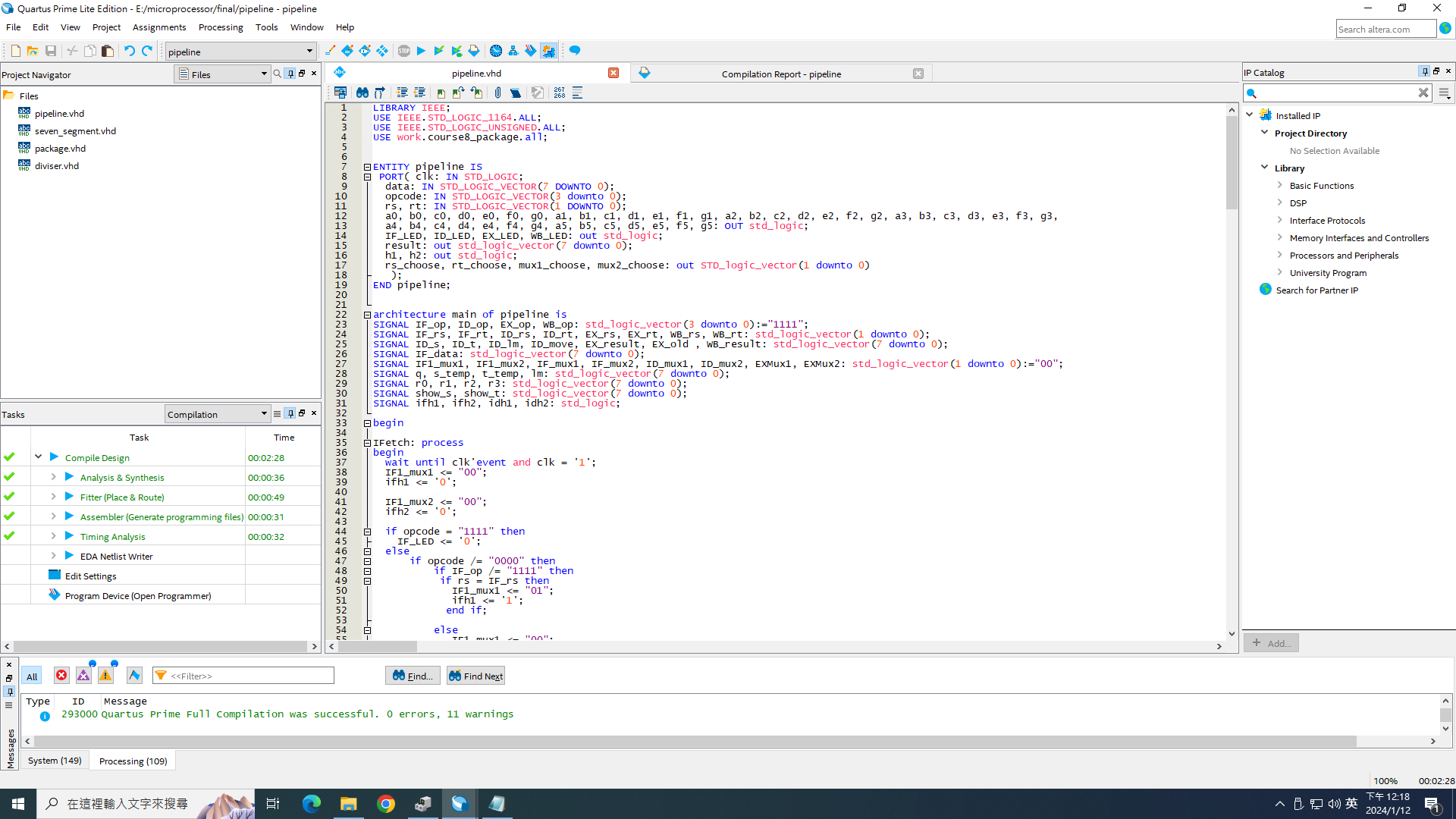


（四） 實驗過程

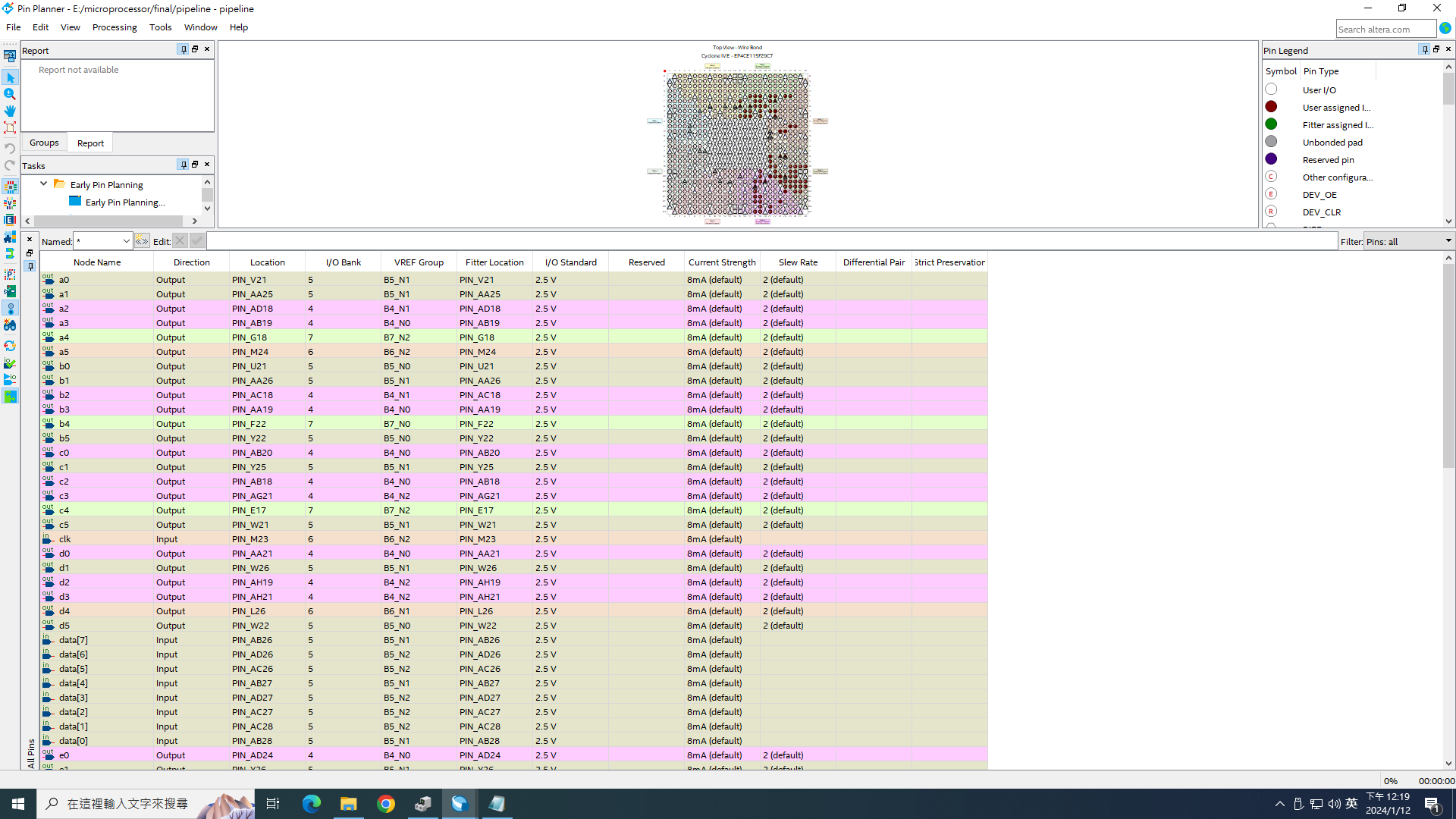
1. 撰寫程式碼

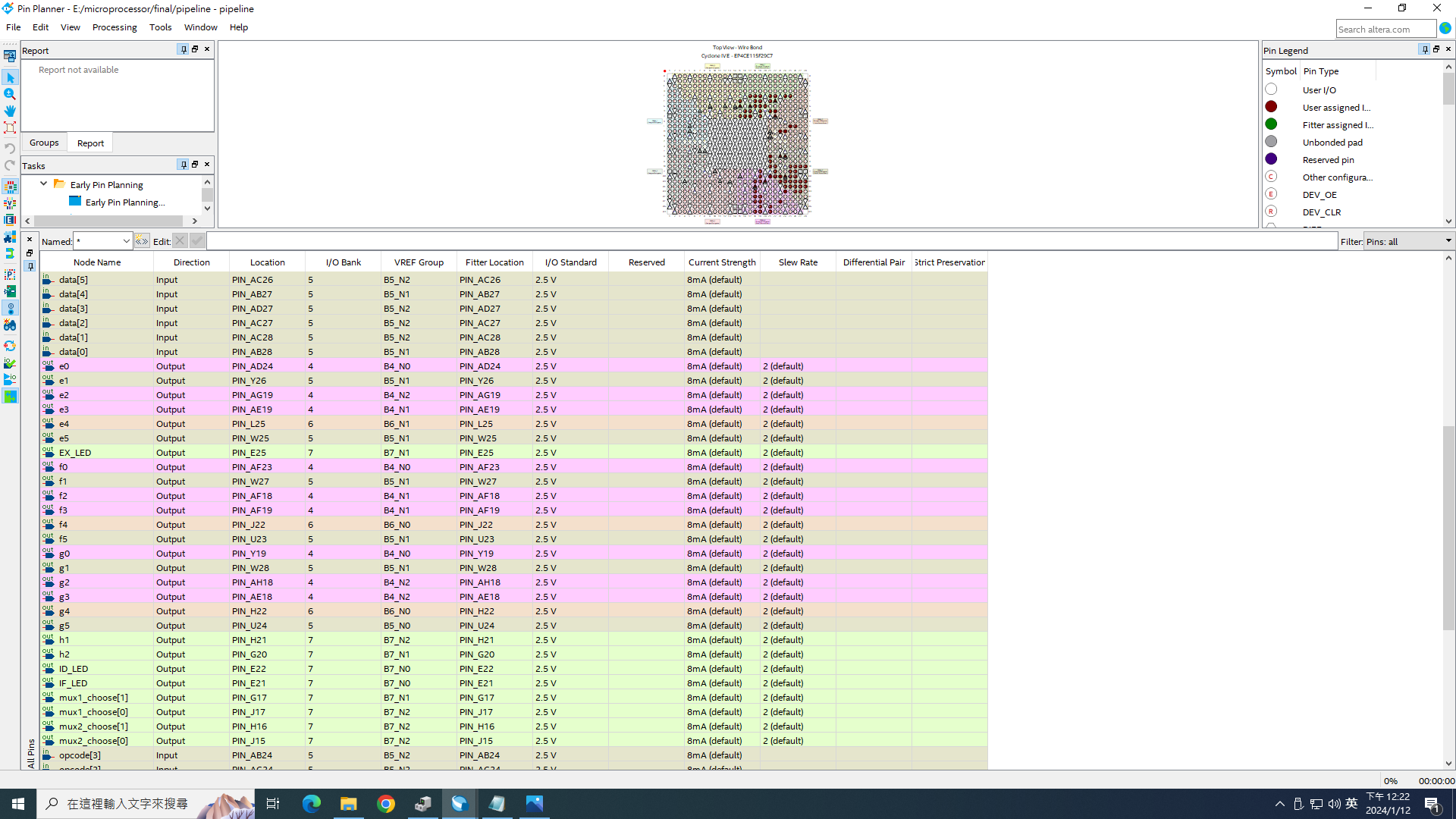


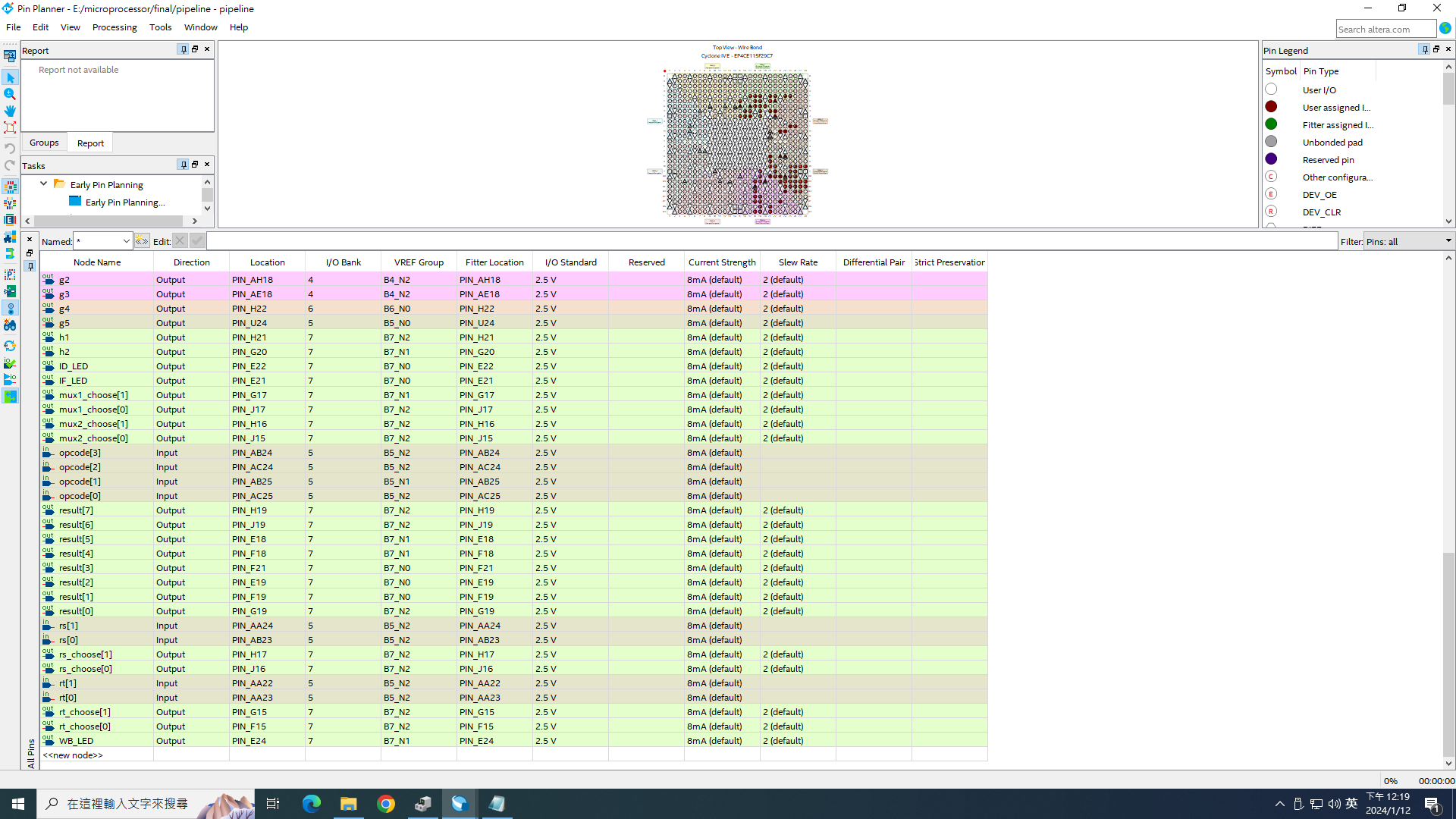
1. 編譯成功



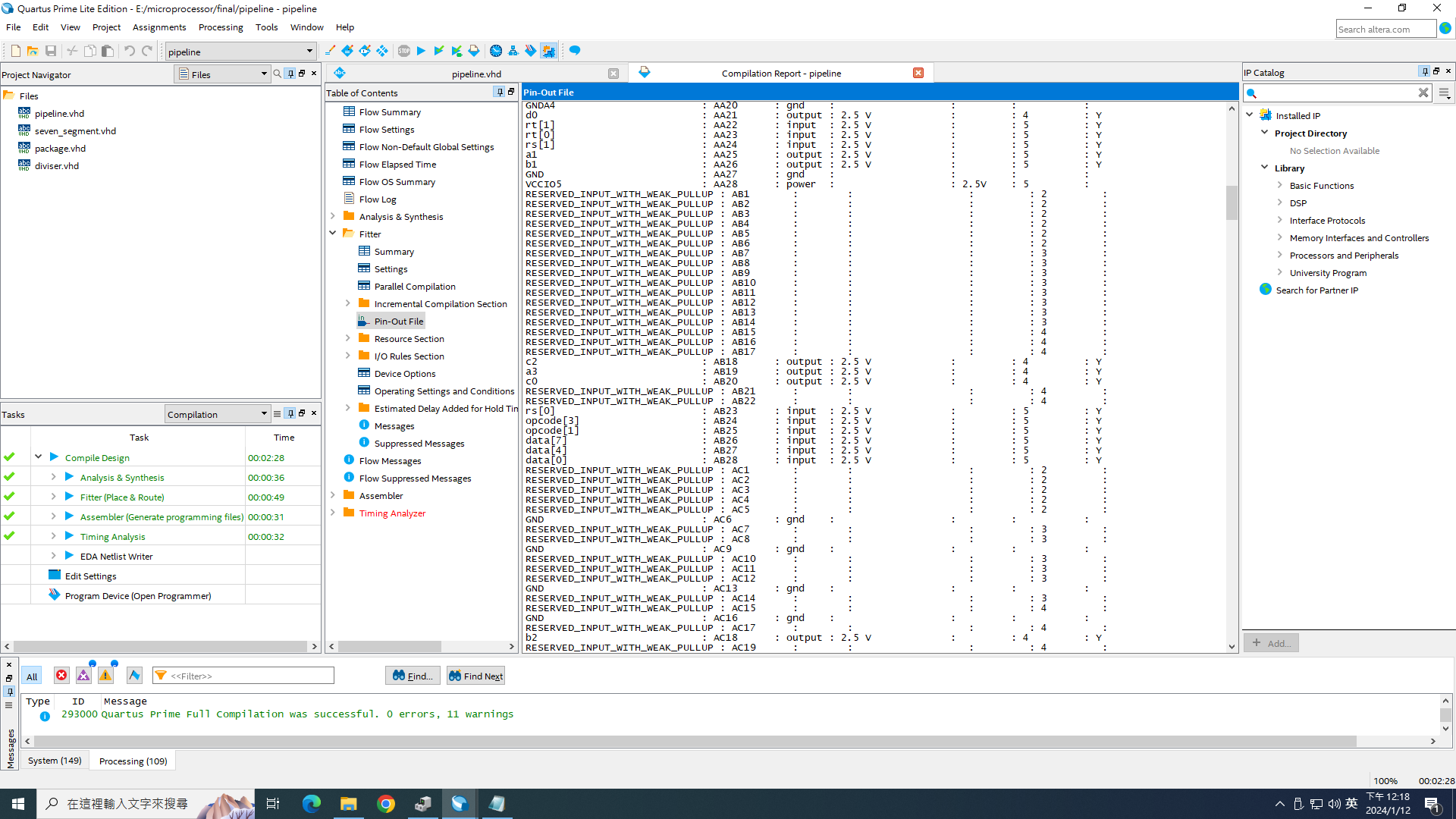
1. 接腳位



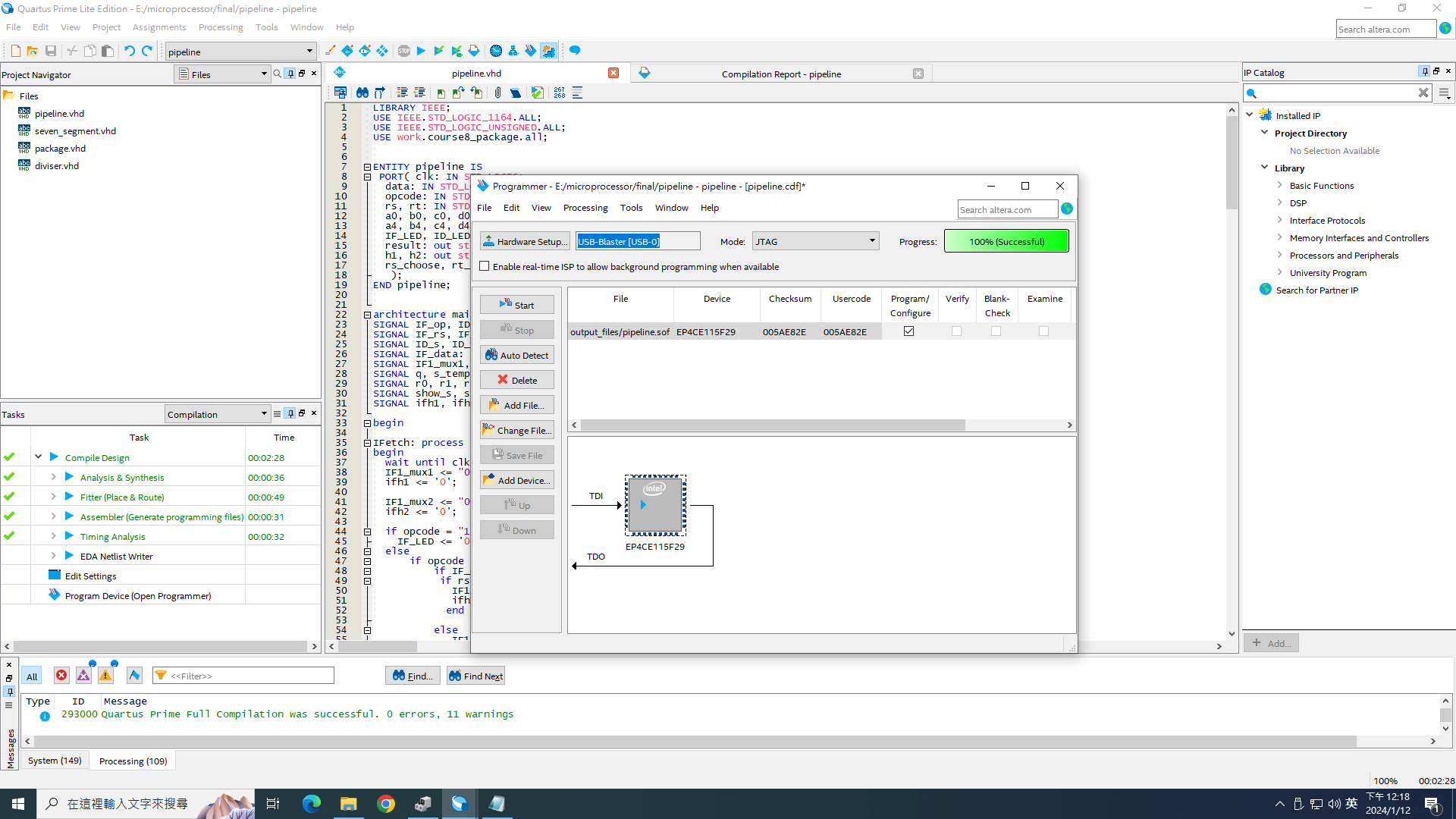




1. 確認接線於正確腳位



1. 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_wXKFlPvW0hx9fp7W5lsUD5ojyO2xB/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, f3, 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 downto 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: std\_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), show\_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), show\_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; |