Department of Computer Science and Information Engineering

National Cheng Kung University

Weekend Practice 11/20

Verilog基本語法

- **wire**
  - input、output預設型別為wire
  - 可使用assign語法賦值
  - 連接module output腳位需使用wire
- **reg**
  - always block
  - 合成後差別：
    - 組合電路：線 (net)
    - 循序電路：Flip-flop (register)

```
1   module circuit(in, out);
2   input [3:0] in;
3   output [3:0] out;
4
5   reg [3:0] a;
6   wire [3:0] b;
7   wire [3:0] tmp;
8
9   Adder u_adder(.in1(a), .in2(b), .out(tmp));
10
11  assign b = in + 4'd3;
12  assign out = tmp;
13
14  always@(in)
15  begin
16      a = in + 4'd1;
17  end
18
19  endmodule
```

input腳位，可連接reg

output腳位，必須連接wire

input腳位，可連接wire

```
24  module Adder(in1, in2, out);
25  input [3:0] in1, in2;
26  output [3:0] out;
27
28  assign out = in1 + in2;
29
30  endmodule
```

Verilog基本語法

- **Module呼叫 (Structural description)** module_name unit_name(.port_name(signal_name));
  - 使用module name進行呼叫，並給予使用者定義的unit name
  - 可依序給予每個腳位對應的訊號(wire/reg)
    - EX: Adder u_adder(a, b, tmp);
  - 可直接明定腳位與訊號(wire/reg)間的對應關係
    - EX: Adder u_adder(.out(tmp), .in1(a), .in2(b));
  - 呼叫後該module便會持續存在於電路中，藉由給予不同的input訊號來得到不同的output結果
  - 不可在always block中呼叫module

```
1   module circuit(in, out);
2   input [3:0] in;
3   output [3:0] out;
4
5   reg [3:0] a;
6   wire [3:0] b;
7   wire [3:0] tmp;
8
9   Adder u_adder(.in1(a), .in2(b), .out(tmp));
```


```
24  module Adder(in1, in2, out);
25  input [3:0] in1, in2;
26  output [3:0] out;
27
28  assign out = in1 + in2;
29
30  endmodule
```

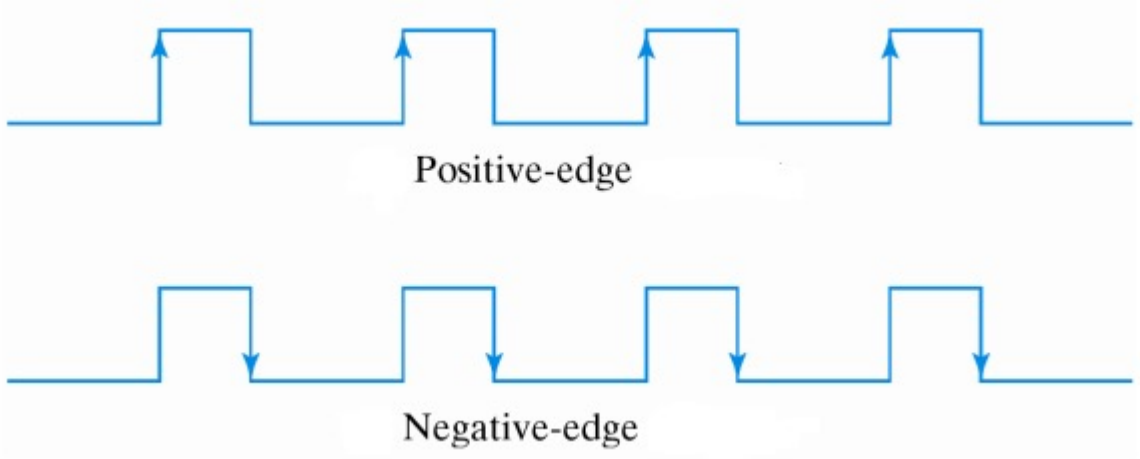
Verilog基本語法

- **wire**
  - input、output預設值
  - assign
  - 連接module output腳位需使用wire
- **reg**
  - always block
  - 合成後差別：
    - 組合電路：線 (net)
    - 循序電路：Flip-flop (register)

```
1   module circuit (clk, reset);
2       input   clk, reset;
3       wire    [7:0] wire_a;
4       reg     [7:0] reg_comb;
5
6       assign  wire_a = 8'd1;
7
8       always @(*) begin
9           reg_comb = 8'd1;
10      end
11
12
13
14      reg     [7:0] reg_seq;
15
16      always @(posedge clk or posedge reset) begin
17          if(reset) begin
18              reg_seq <= 8'd0;
19          end
20          else begin
21              reg_seq <= reg_seq+1'b1;
22          end
23      end
24  endmodule
```

Sequential circuit

- **posedge**
  - 在正緣訊號來時觸發 (0 -> 1)
- **negedge**
  - 在負緣訊號來時觸發 (1 -> 0)




```
29  module posedge_sequential_circuit(clk);
30      input   clk;
31
32      always @(posedge clk) begin
33          // circuit functionality
34      end
35
36  endmodule
37
38
39
40  module negedge_sequential_circuit(clk);
41      input   clk;
42
43      always @(negedge clk) begin
44          // circuit functionality
45      end
46
47  endmodule
```

正緣同步電路

負緣同步電路

Synchronous or Asynchronous

- **Asynchronous reset**
  - 當reset訊號發生改變時(正/負緣)，電路會立刻重置
- **Synchronous reset**
  - 需要等到clk正緣觸發時，電路才會重置

  - 以上以正緣同步電路為例

```
52  module asynchronous_sequential_circuit(clk, reset);
53      input   clk, reset;
54
55      always @(posedge clk or negedge reset) begin
56          if(!reset) begin
57              // initialization
58          end
59          else begin
60              // circuit functionality
61          end
62      end
63
64  endmodule
65
66
67  module synchronous_sequential_circuit(clk, reset);
68      input   clk, reset;
69
70      always @(posedge clk) begin
71          if(!reset) begin
72              // initialization
73          end
74          else begin
75              // circuit functionality
76          end
77      end
78
79  endmodule
```

低位準非同步重置

低位準同步重置

Blocking or Non-blocking

- **Blocking**
  - 由上而下依序執行程式
  - 用於組合電路
- **Non-blocking**
  - 在clock正緣或負緣時，以同步的方式執行程式
  - 在同一瞬間對always block內的reg變數進行操作
  - 用於循序電路

```
238  // blocking
239  always @(posedge clk or negedge rst)
240  begin
241      if(!rst)
242      begin
243          a = a_i;
244          b = b_i;
245      end
246      else
247      begin
248          a = b;
249          b = a;
250          /*
251              result :
252              a = b_i
253              b = b_i
254          */
255      end
256  end
```

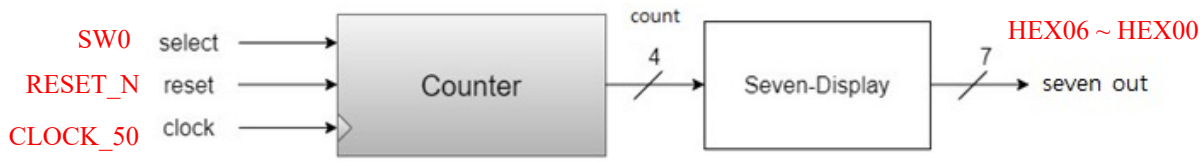
blocking

```
259  // non-blocking
260  always @(posedge clk or negedge rst)
261  begin
262      if(!rst)
263      begin
264          a <= a_i;
265          b <= b_i;
266      end
267      else
268      begin
269          a <= b;
270          b <= a;
271          /*
272              result :
273              a = b_i
274              b = a_i
275          */
276      end
277  end
```

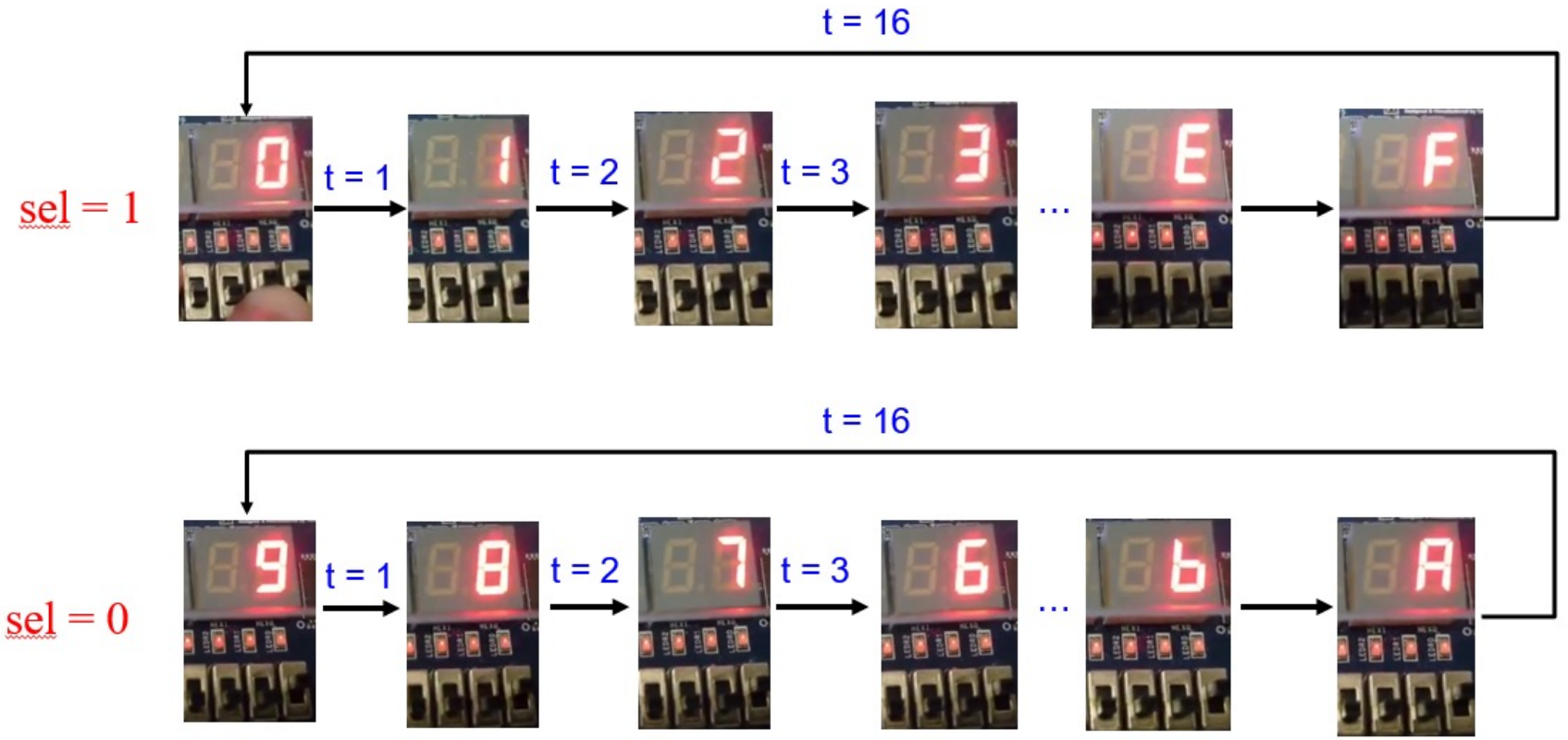
non-blocking

Lab – Counter (1/2)

- **請設計一個具備下列功能的計數器：**
  - 正緣同步電路，低位準同步重置(reset)
  - 若reset訊號為0，將計數器歸零
  - 若reset訊號為1，根據sel判斷目前是要向下數或是向上數，若sel＝1則從目前的數字向上數，若sel＝0 則從目前的數字向下數。
    - Ex : sel=1; 0->1->2-> …… ->E->F->0->1
    - Ex : sel=0; 0->F->E->D->……->1->0->F
  - 每0.5秒加1或減1

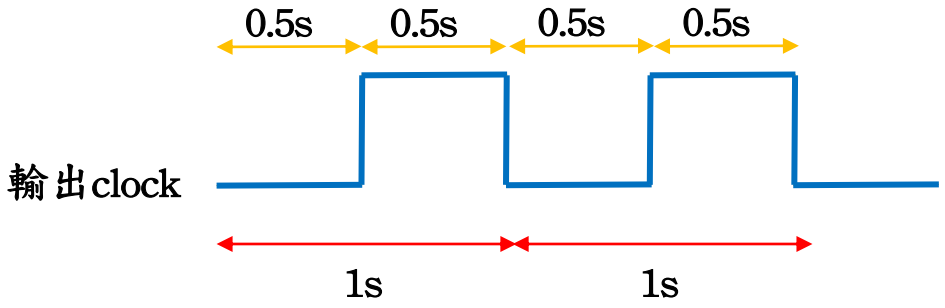


Lab – Counter (2/2)



Frequency Divider

- 利用counter在輸入的clock正緣進行計數，當經過`TimeExpire個正緣後更改輸出clock的正負，即可達到除頻的效果
- FPGA版所使用的clock頻率為50MHz，假設欲產生1Hz的clock，則需每0.5秒更改輸出clock的正負
- 兩個正緣間隔為1/(50*10^6)秒，0.5秒共會經過25*10^{6}個正緣，故`TimeExpire設為25000000




```
1  `define TimeExpire 32'd25000000
2
3  module clk_div(clk,rst,div_clk);
4  input clk,rst;
5  output div_clk;
6
7  reg div_clk;
8  reg [31:0]count;
9
10 always@(posedge clk)
11 begin
12     if(!rst)
13     begin
14         count <= 32'd0;
15         div_clk <= 1'b0;
16     end
17     else
18     begin
19         if(count == `TimeExpire)
20         begin
21             count <= 32'd0;
22             div_clk <= ~div_clk;
23         end
24         else
25         begin
26             count <= count + 32'd1;
27         end
28     end
29 end
30
31 endmodule
```

正緣同步電路

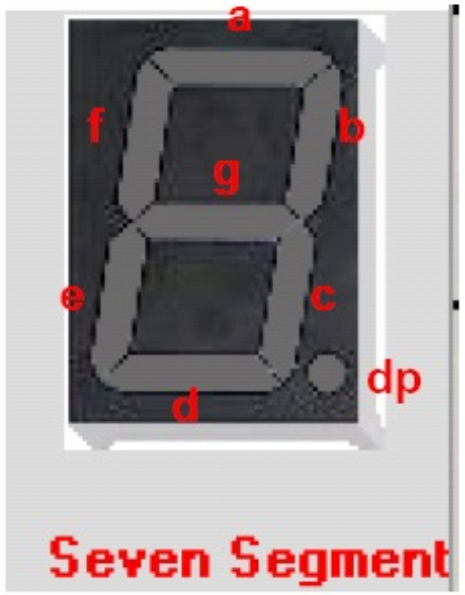
低位準同步reset

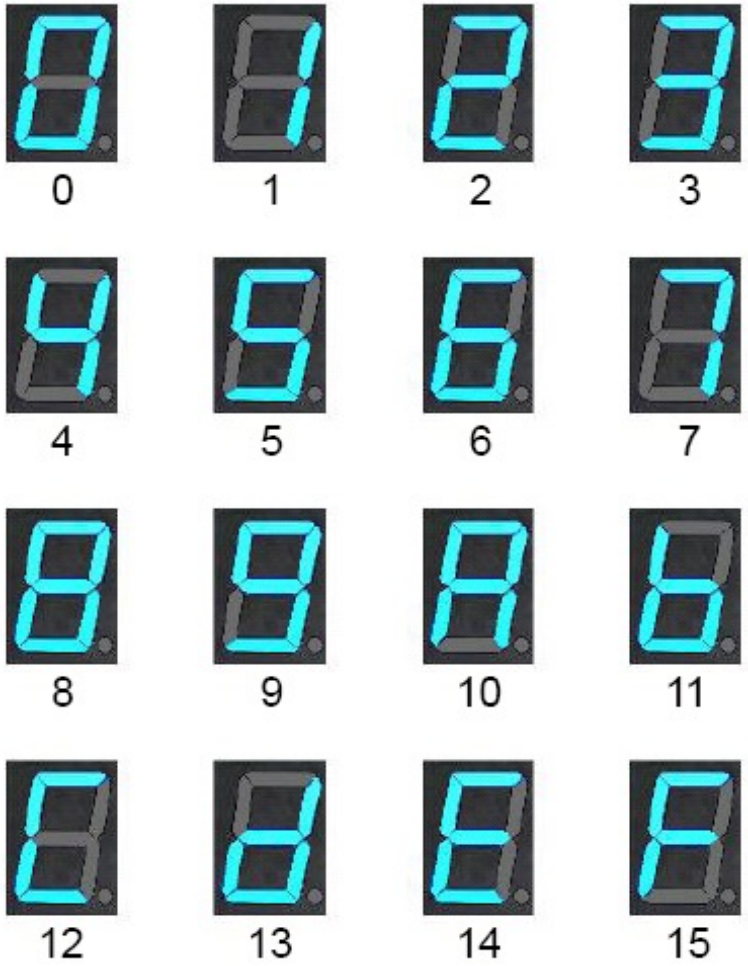
判斷是否經過`TimeExpire個正緣

更改輸出clock的正負並歸零計數器

計數

Seven-segment display (1/2)





Seven-segment display (2/2)

- 0 is on, 1 is off
- dp is useless in DE0-CV board
- Ex: 
0

  out=7'b1000000;

  g=1
- Ex: 
5

  out=7'b0010010;

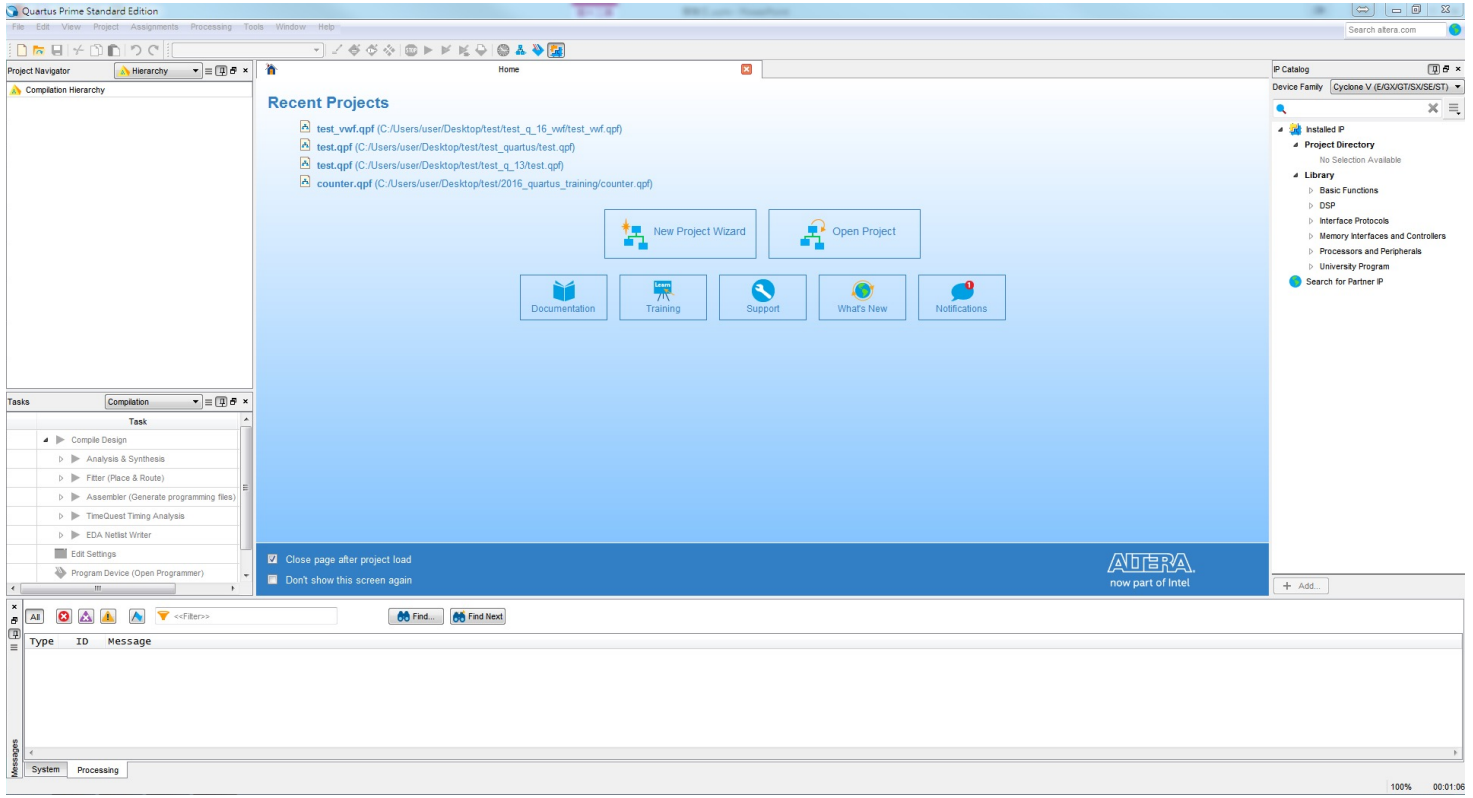
  b=1, e=1

Notice

- 請勿命名中文或數字開頭的資料夾
- Device family 請確認與 FPGA Chip 符合 (5CEFA4F23C7)
- Top module name & Project name 需要一致

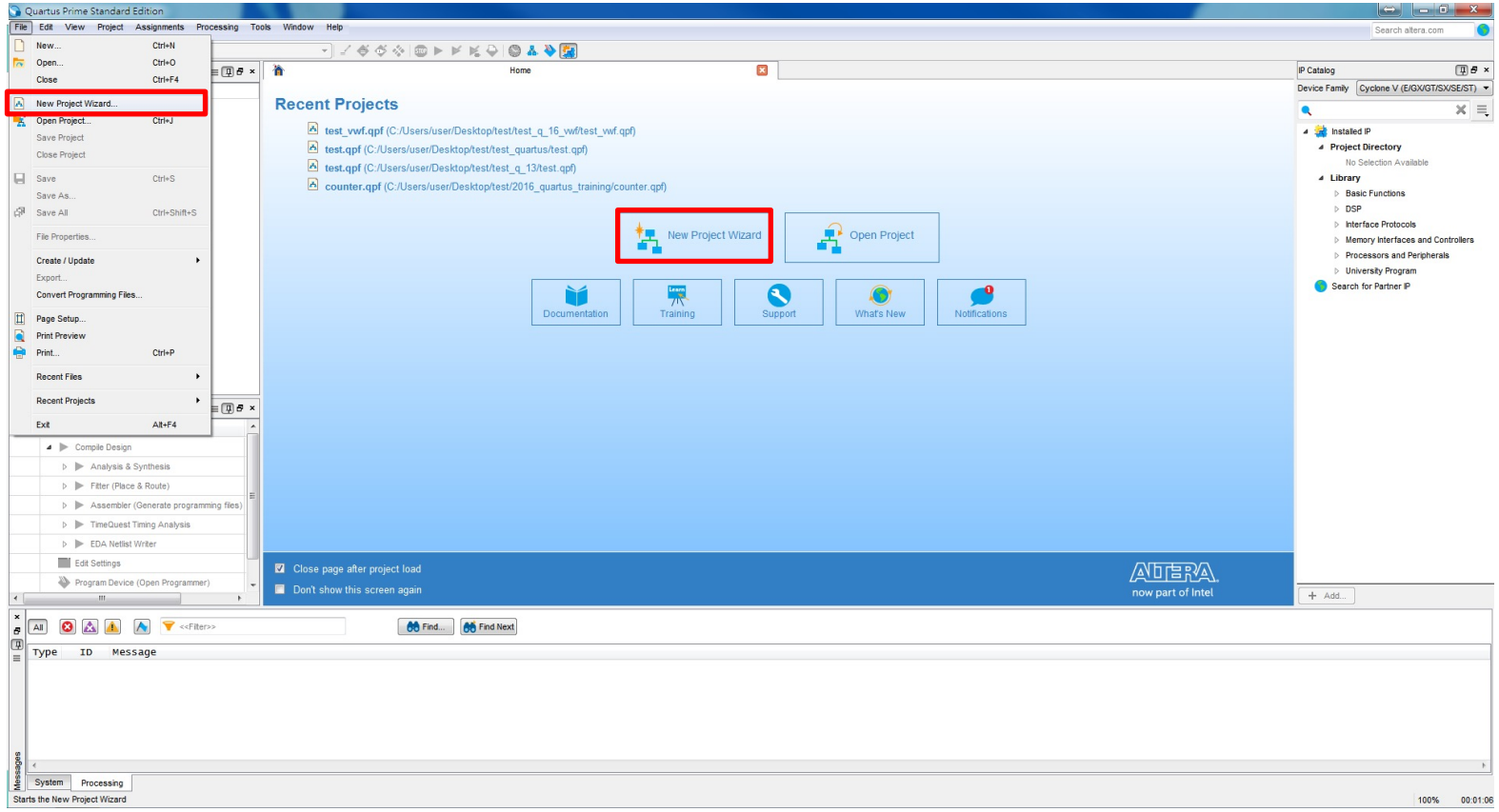
Quartus II Tutorial (1/10)

- **Getting Started –**
  - **Start the Quartus II software**



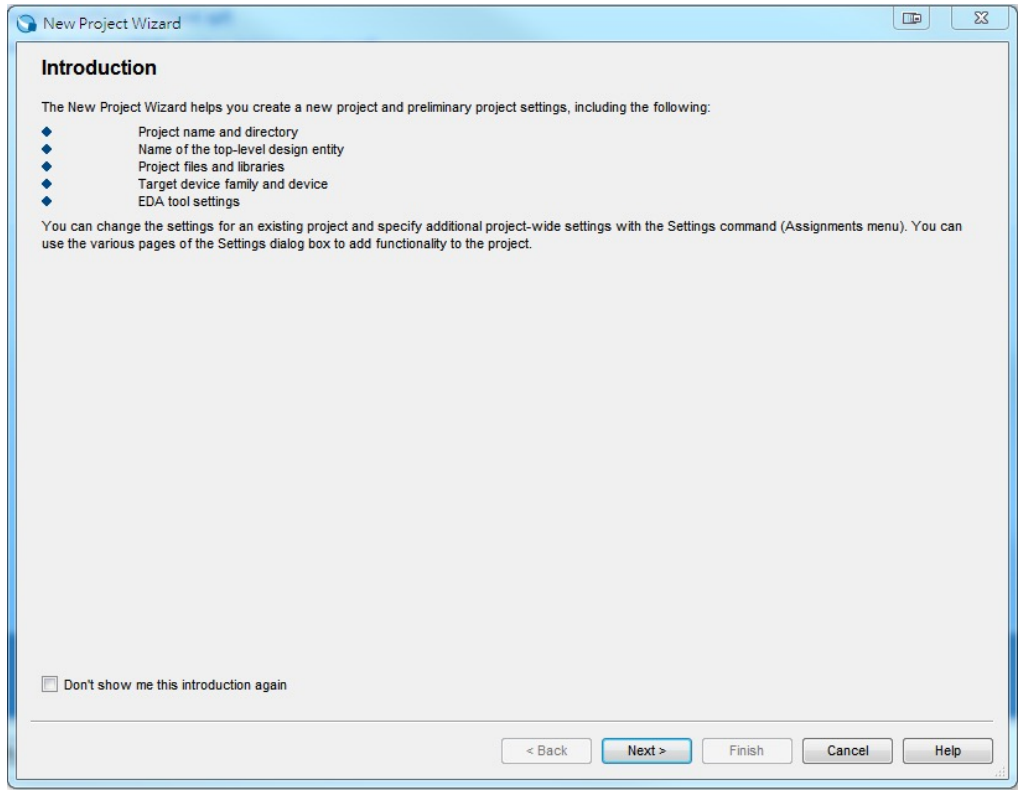
Quartus II Tutorial (2/10)

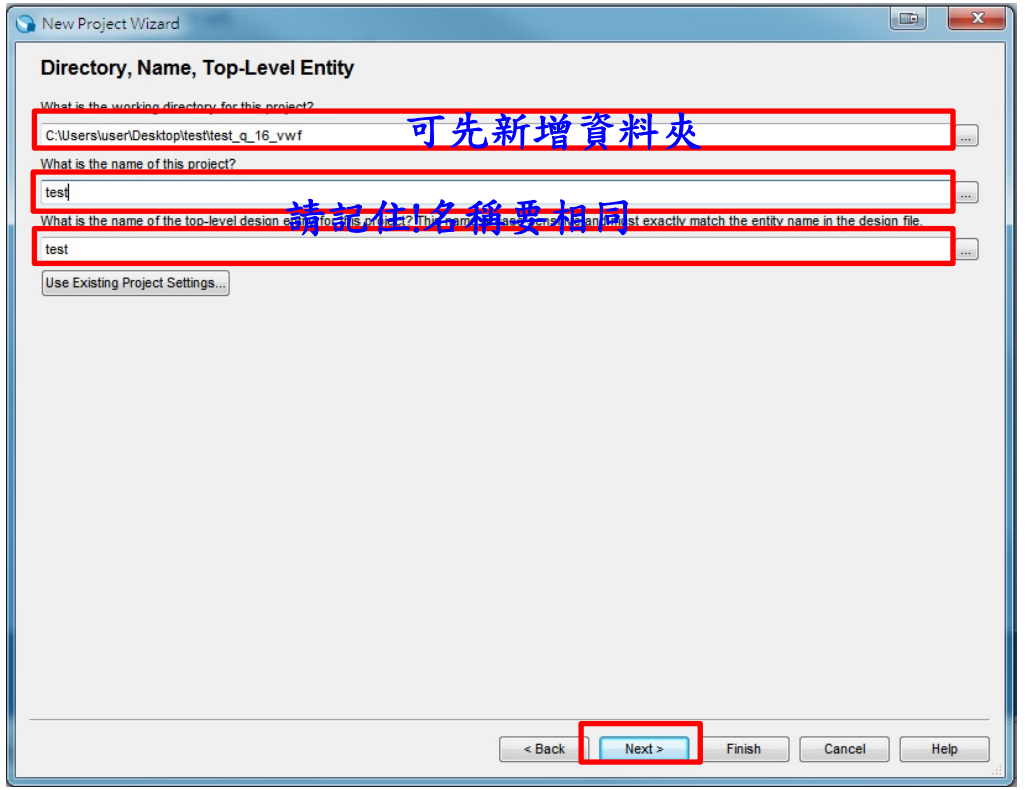
- **Create a New Project –**
  - **Open New Project Wizard (File → New Project Wizard…)**



Quartus II Tutorial (3/10)

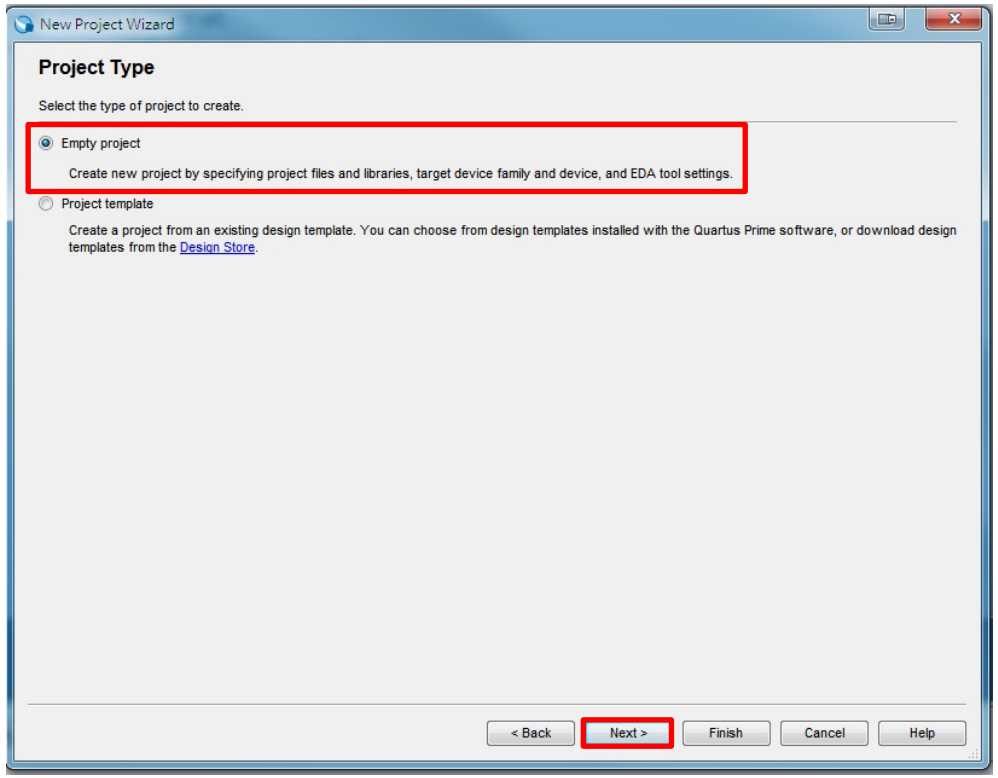
- **Specify the working directory and the name of the project**

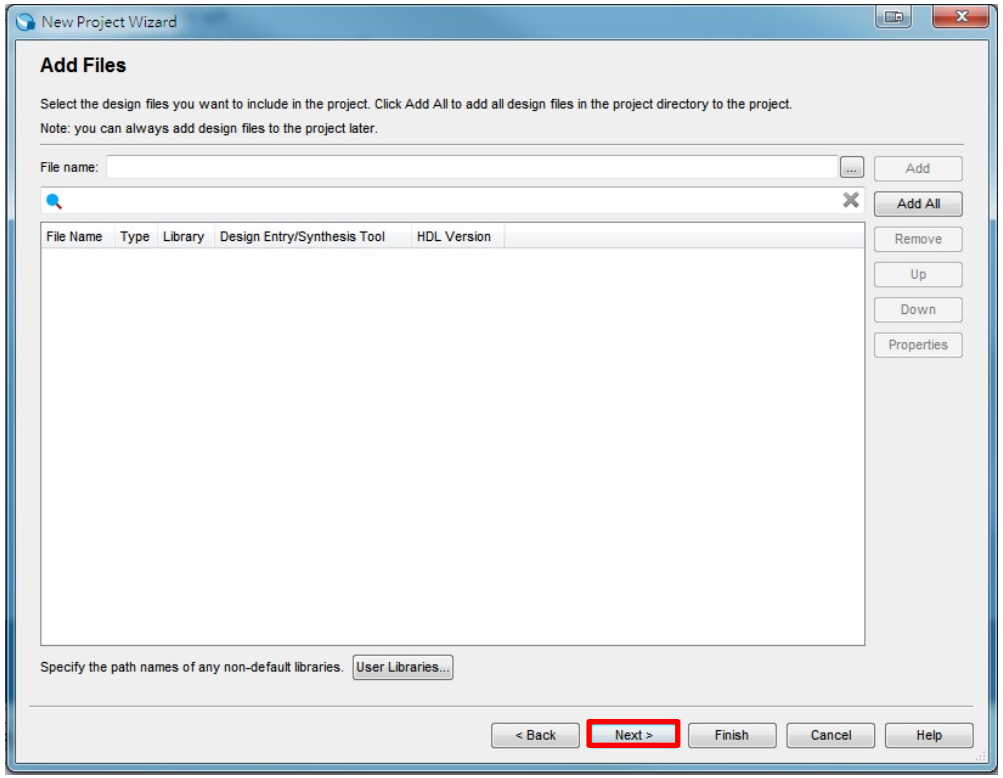




Quartus II Tutorial (4/10)

- Select “Empty project”. Then, click “Next”.
- Select design files. Or click “Next” to skip this step.

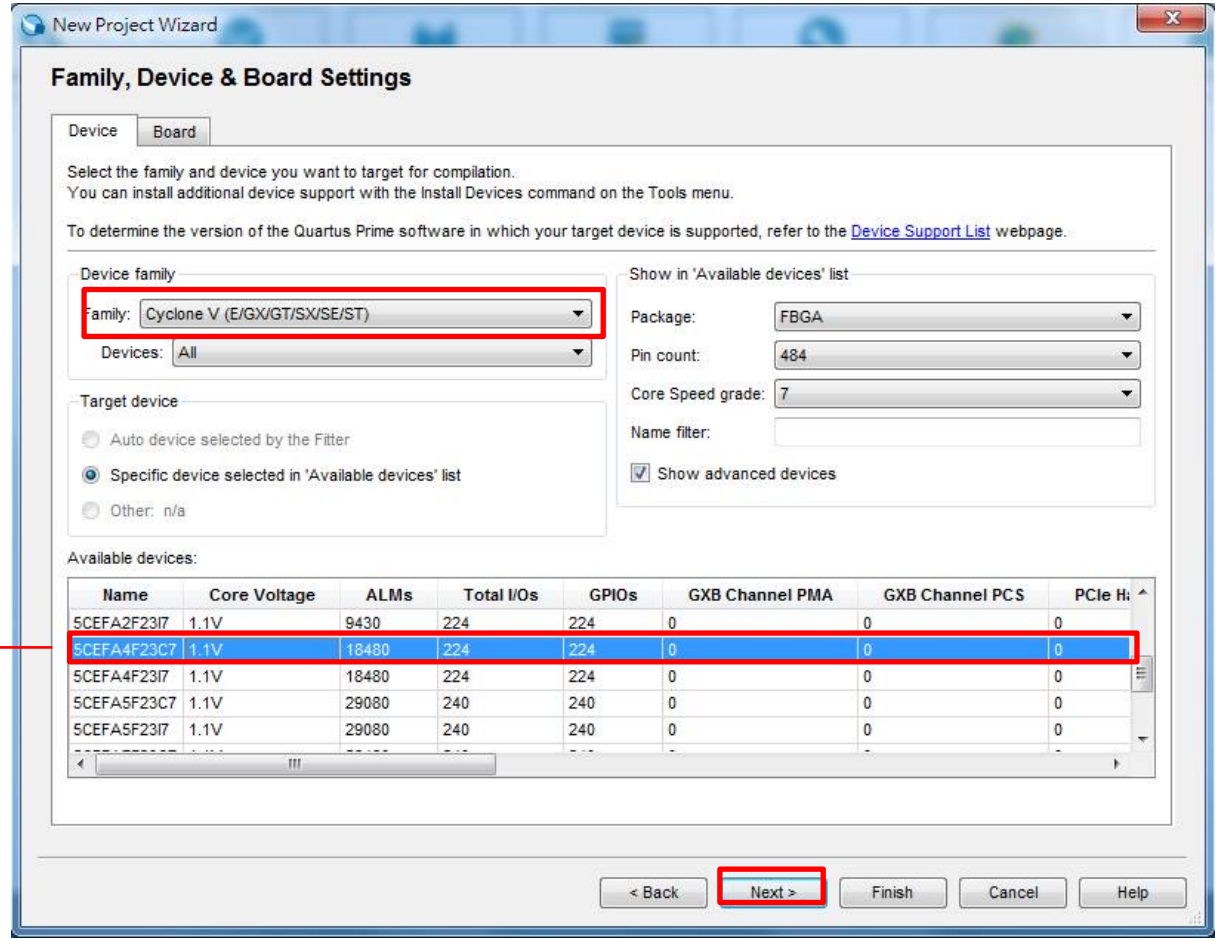




Quartus II Tutorial (5/10)

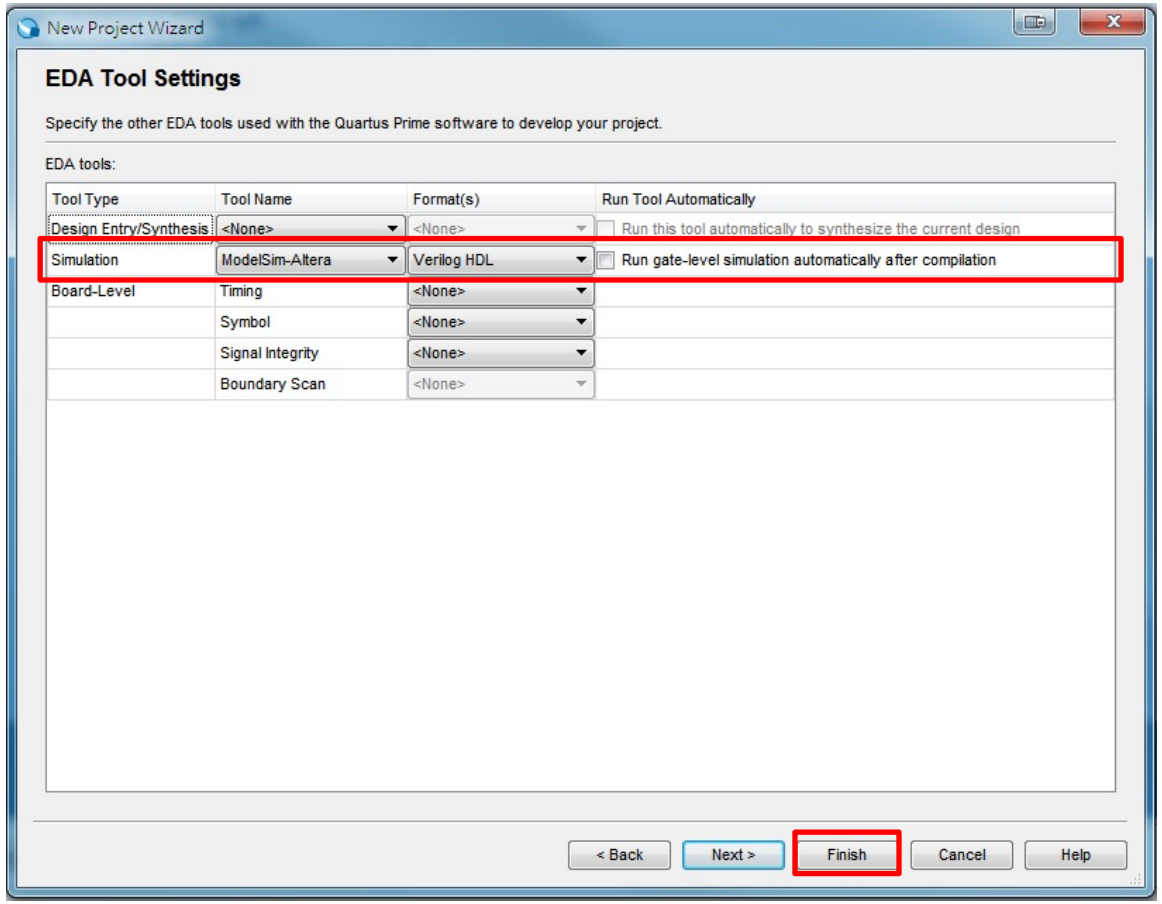
- Specify device settings - (DE0-CV Device family are used). Click "Next."

5CEFA4F23C7



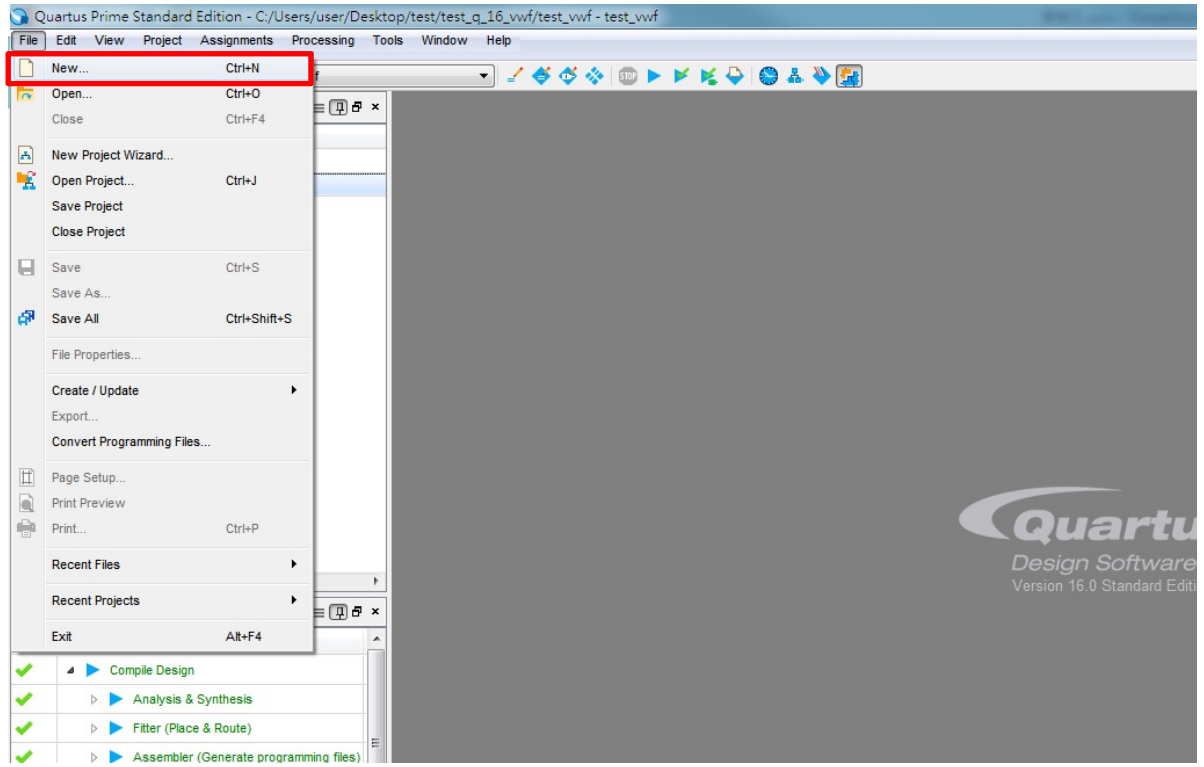
Quartus II Tutorial (6/10)

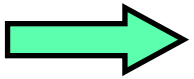
- **Specify EDA Tool – (Modelsim-Altera is selected for simulation). Click "Finish."**

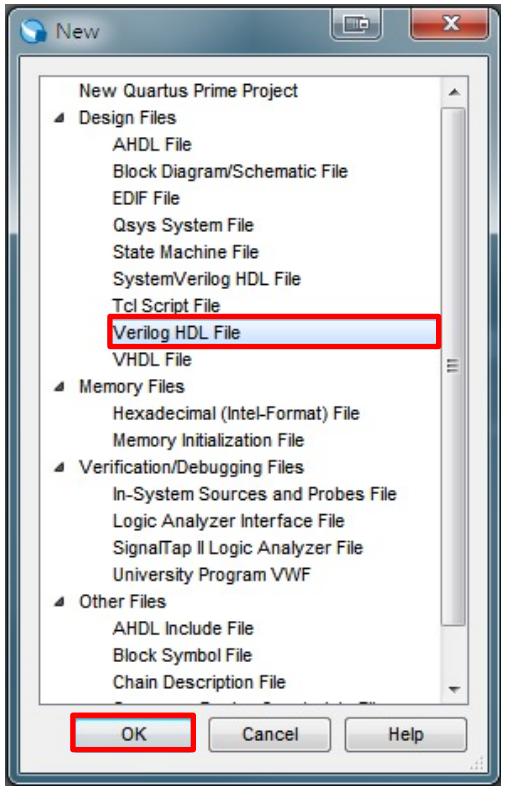


Quartus II Tutorial (7/10)

- **Edit a new file by opening a Verilog HDL file**
  - **(File → New → Verilog HDL File → OK)**







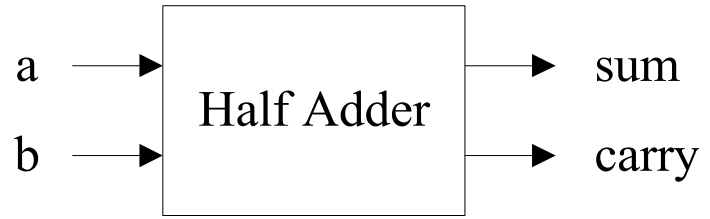
Quartus II Tutorial (8/10)

- **Write Verilog code**

Top module name 一定要跟 Project name 相同 !!

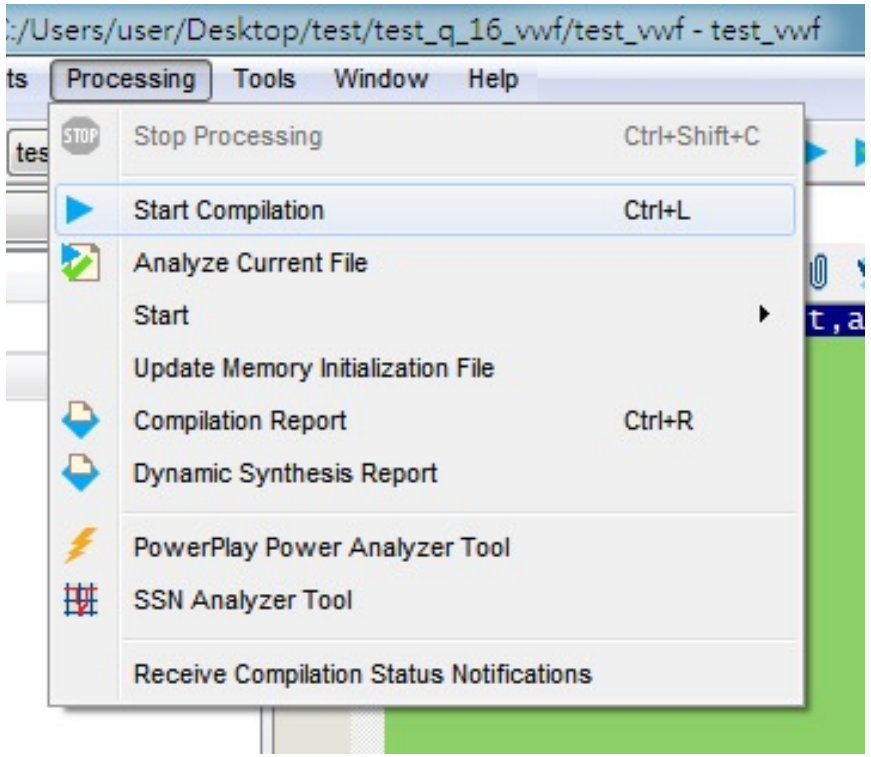
```
1: //File Name：Half_Adder.v
2: module    Half_Adder(a, b, sum, carry);
3: input     a, b;
4: output    sum, carry;
5:
6: assign sum = a ^ b;
7: assign carry = a & b;
8:
9: endmodule
```

輸入(input)		輸出(output)	
被加數(a)	加數(b)	和(sum)	進位(carry)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



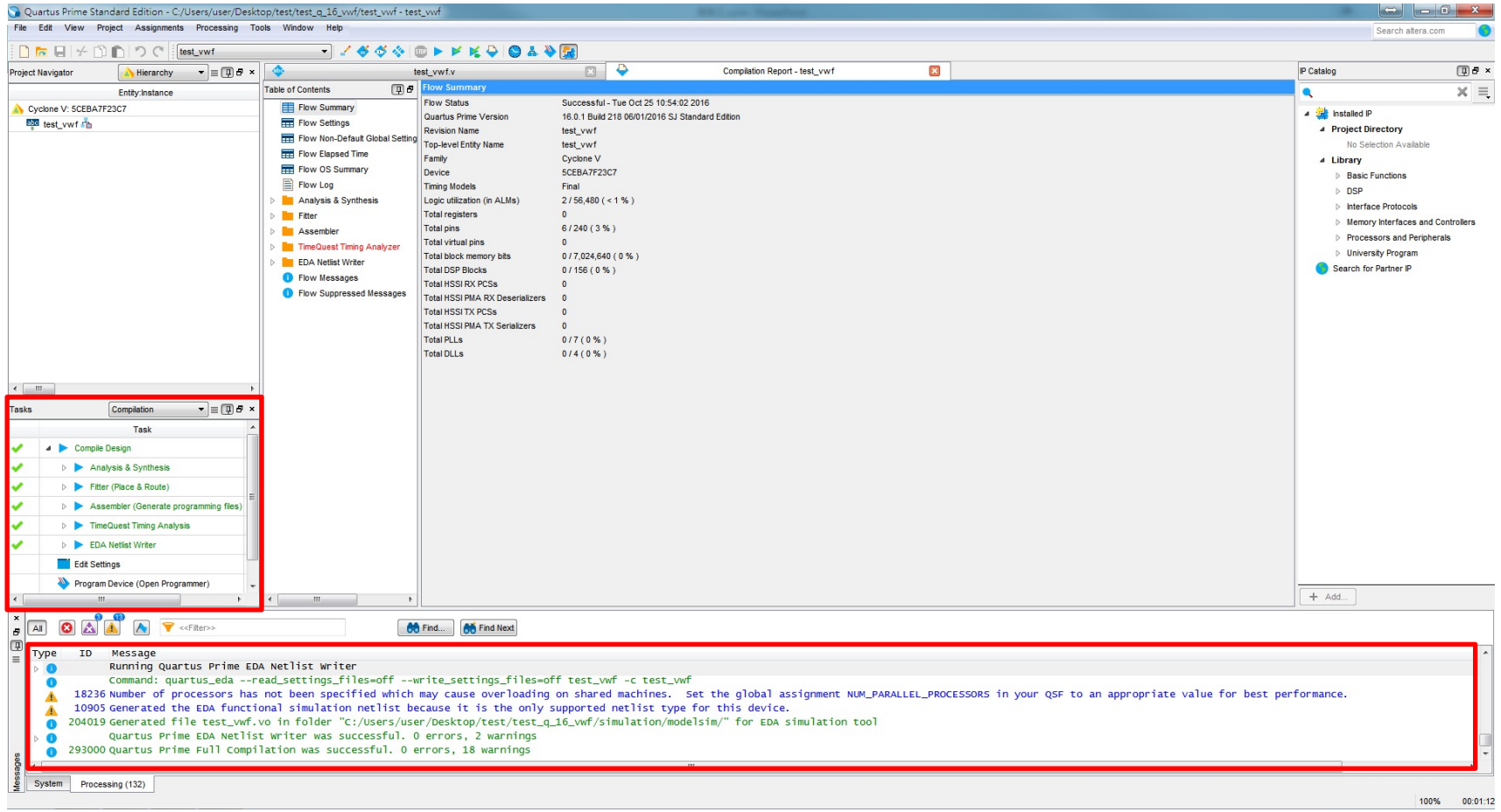
Quartus II Tutorial (9/10)

- **Compiling the Designed Circuit (synthesis 合成)**
  - **(Processing → Start Compilation)**



Quartus II Tutorial (10/10)

- **Successful compilation**

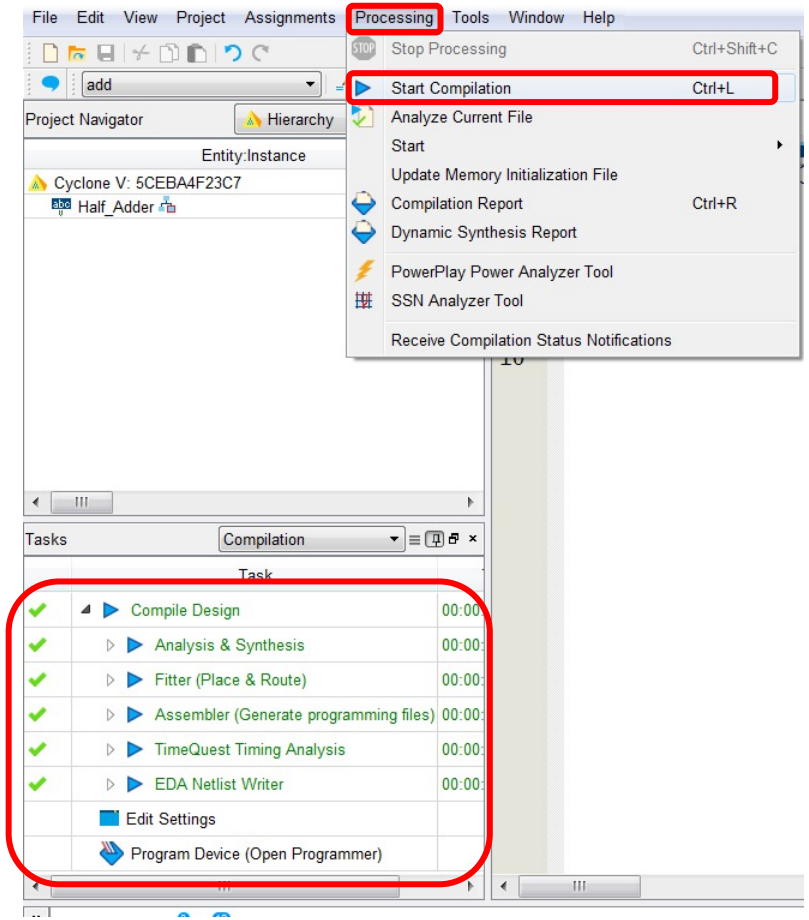


Programming DE0-CV (1/13)

```
 1   module   Half_Adder(a, b, sum, carry);
 2
 3   input a,b;
 4   output   sum, carry;
 5
 6   and(carry,a,b);
 7   xor(sum,a,b);
 8
 9   endmodule
10
```

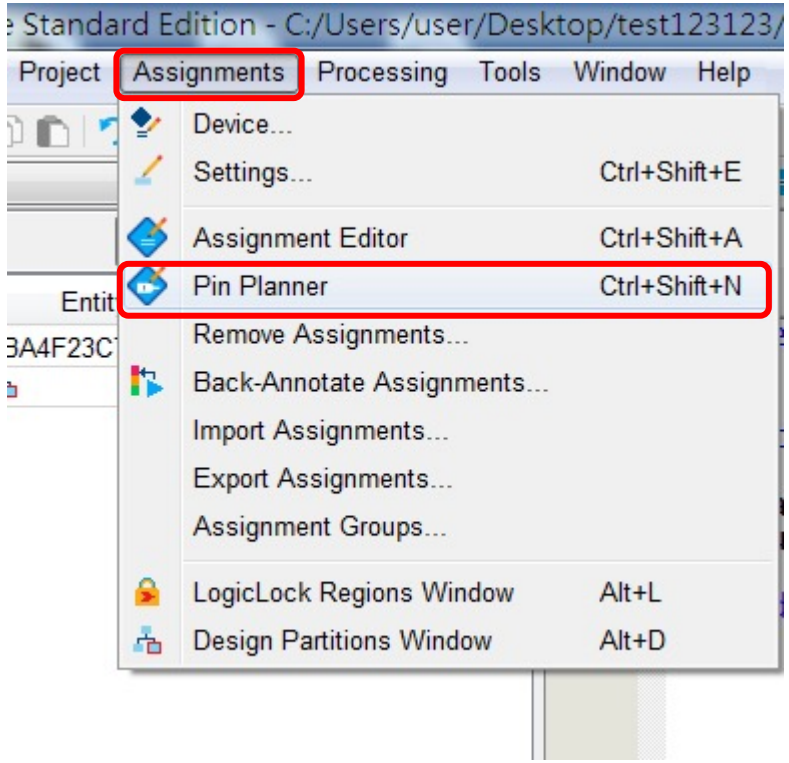
Programming DE0-CV (2/13)

- Start compilation



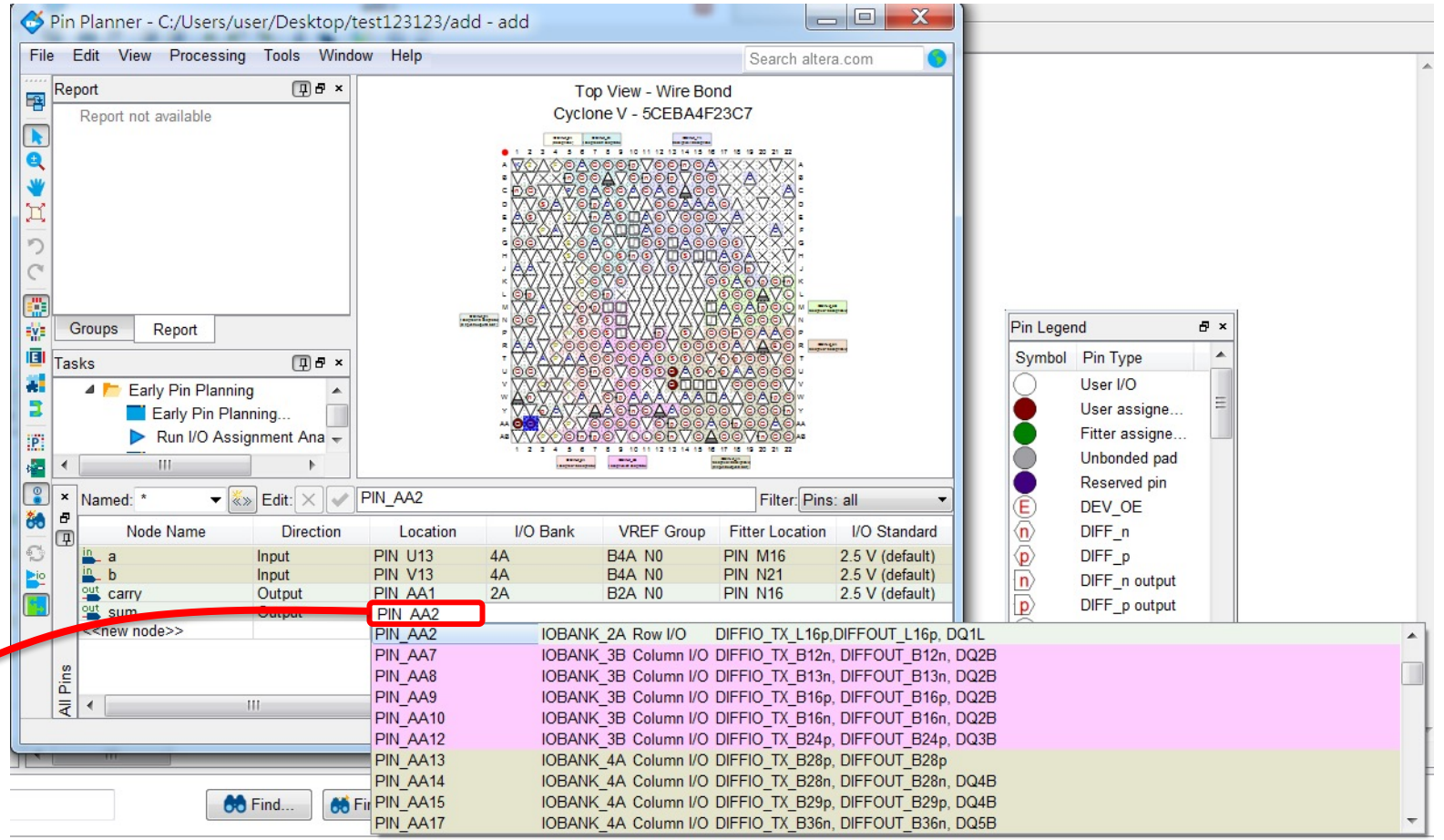
Programming DE0-CV (3/13)

- Open Pin Planner



Programming DE0-CV (4/13)

- Pin assignment



Programming DE0-CV (5/13)

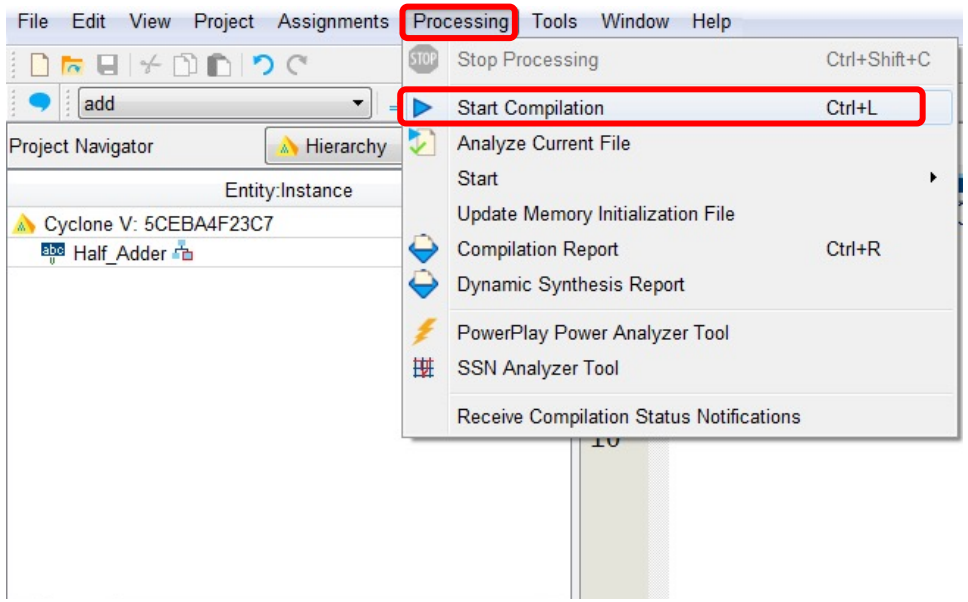
- Assign pin location to all inputs and outputs

Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard
in a	Input	PIN U13 SW0	4A	B4A N0	PIN M16	2.5 V (default)
in b	Input	PIN V13 SW1	4A	B4A N0	PIN N21	2.5 V (default)
out carry	Output	PIN AA1 LED1	2A	B2A N0	PIN N16	2.5 V (default)
out sum	Output	PIN AA2 LED0				

  - Please refer to DE0_pin.xls for pin location assignment

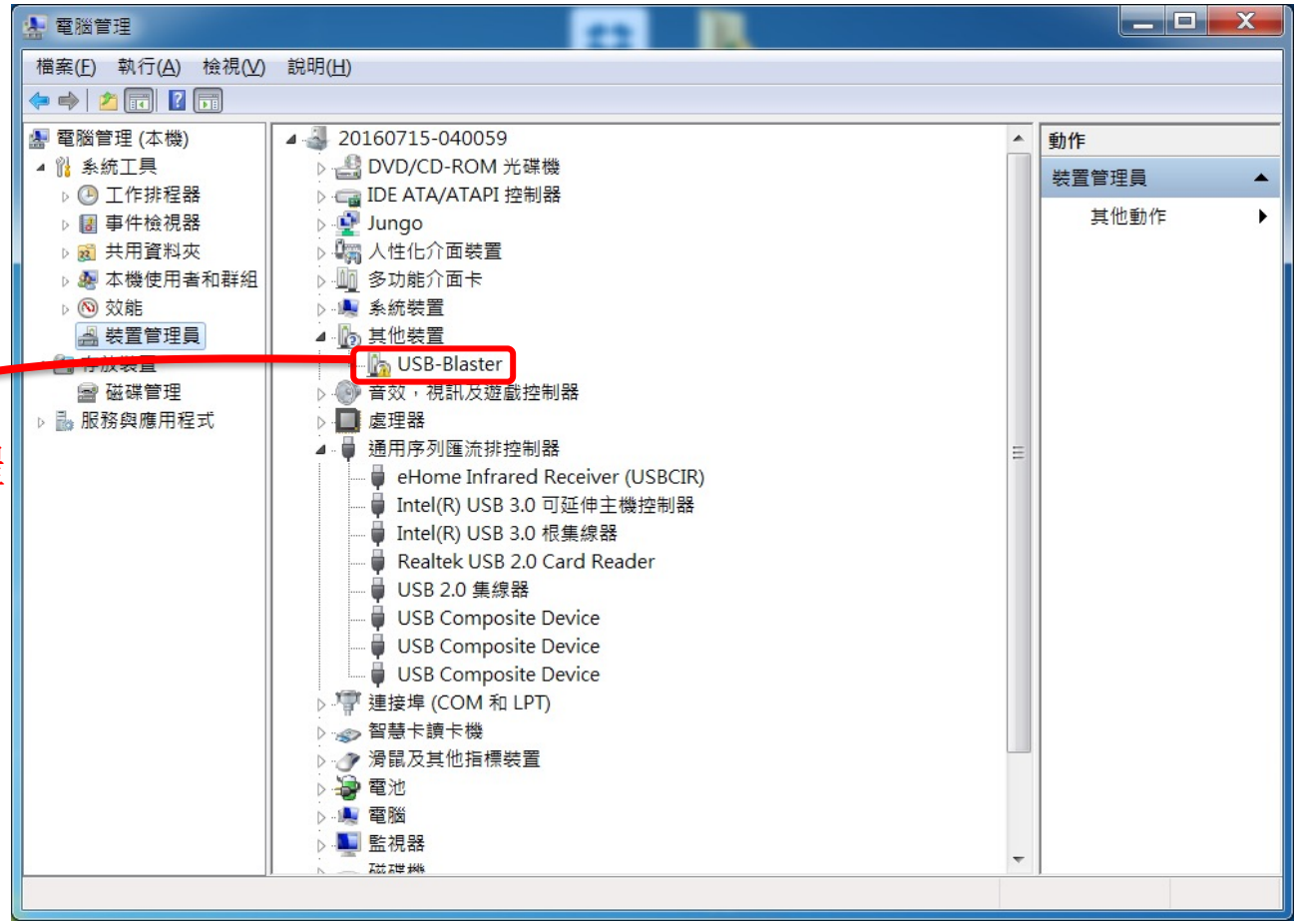
Programming DE0-CV (6/13)

- Start compilation

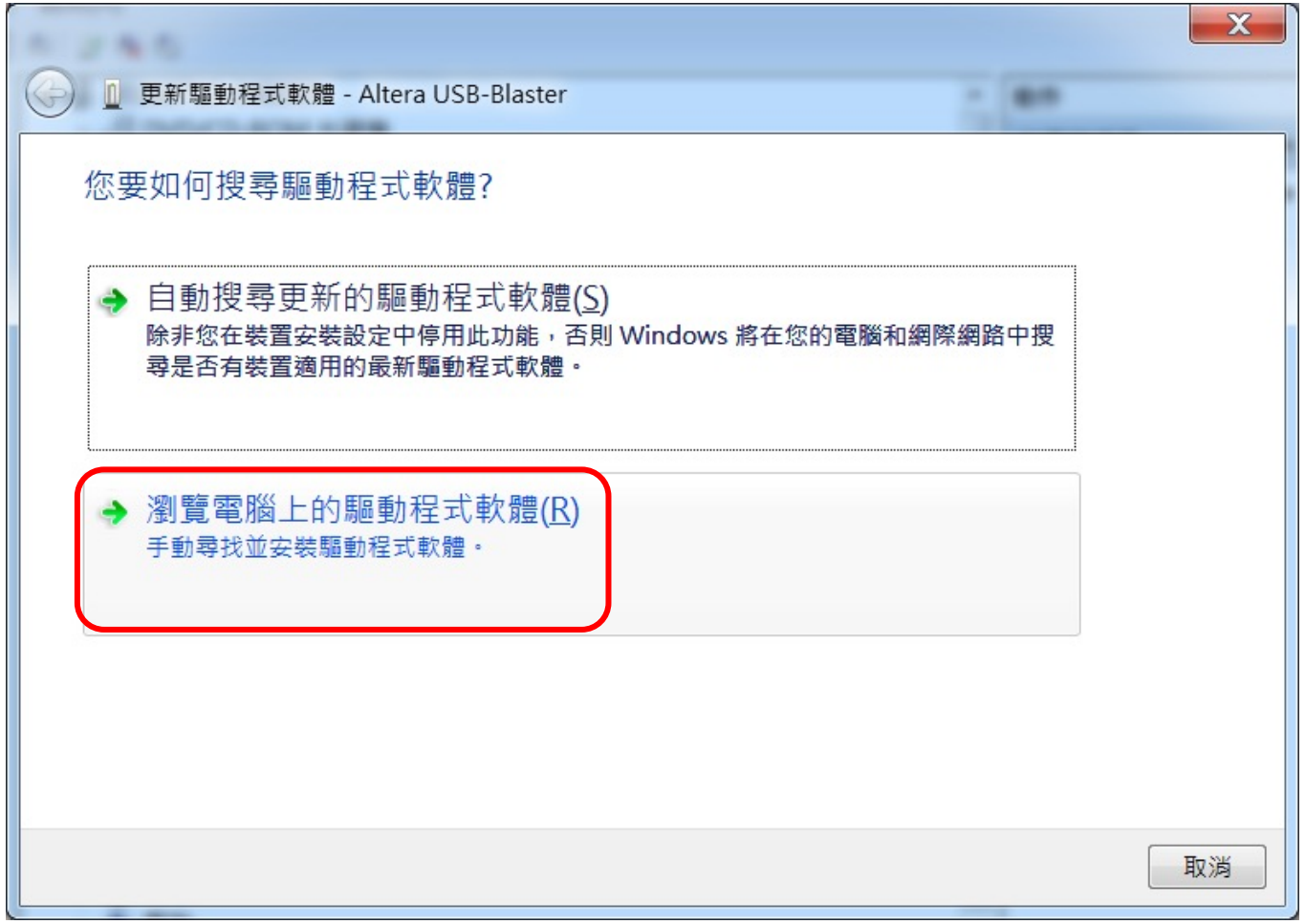


Programming DE0-CV (7/13)

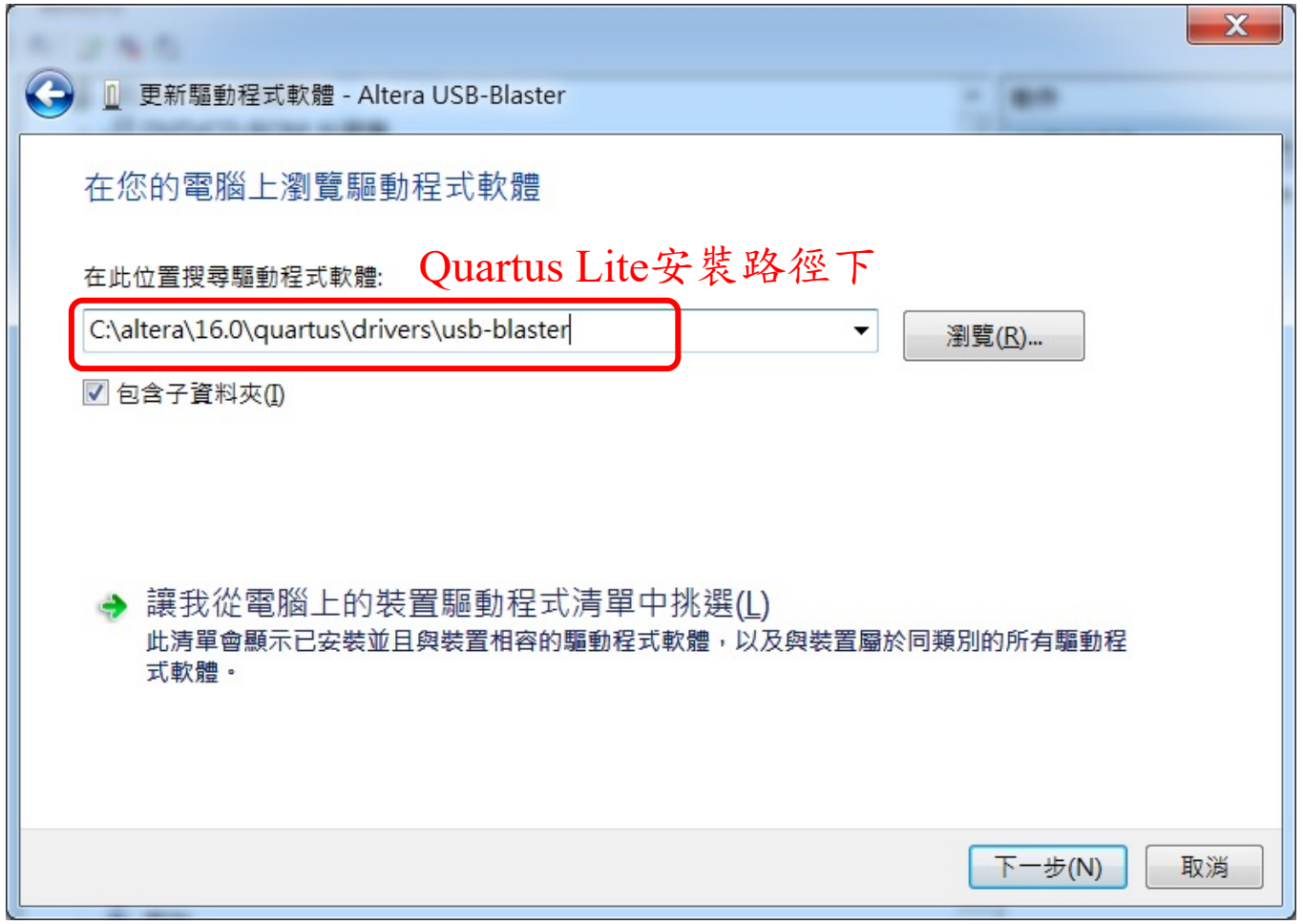
右鍵選更新驅動程式軟體



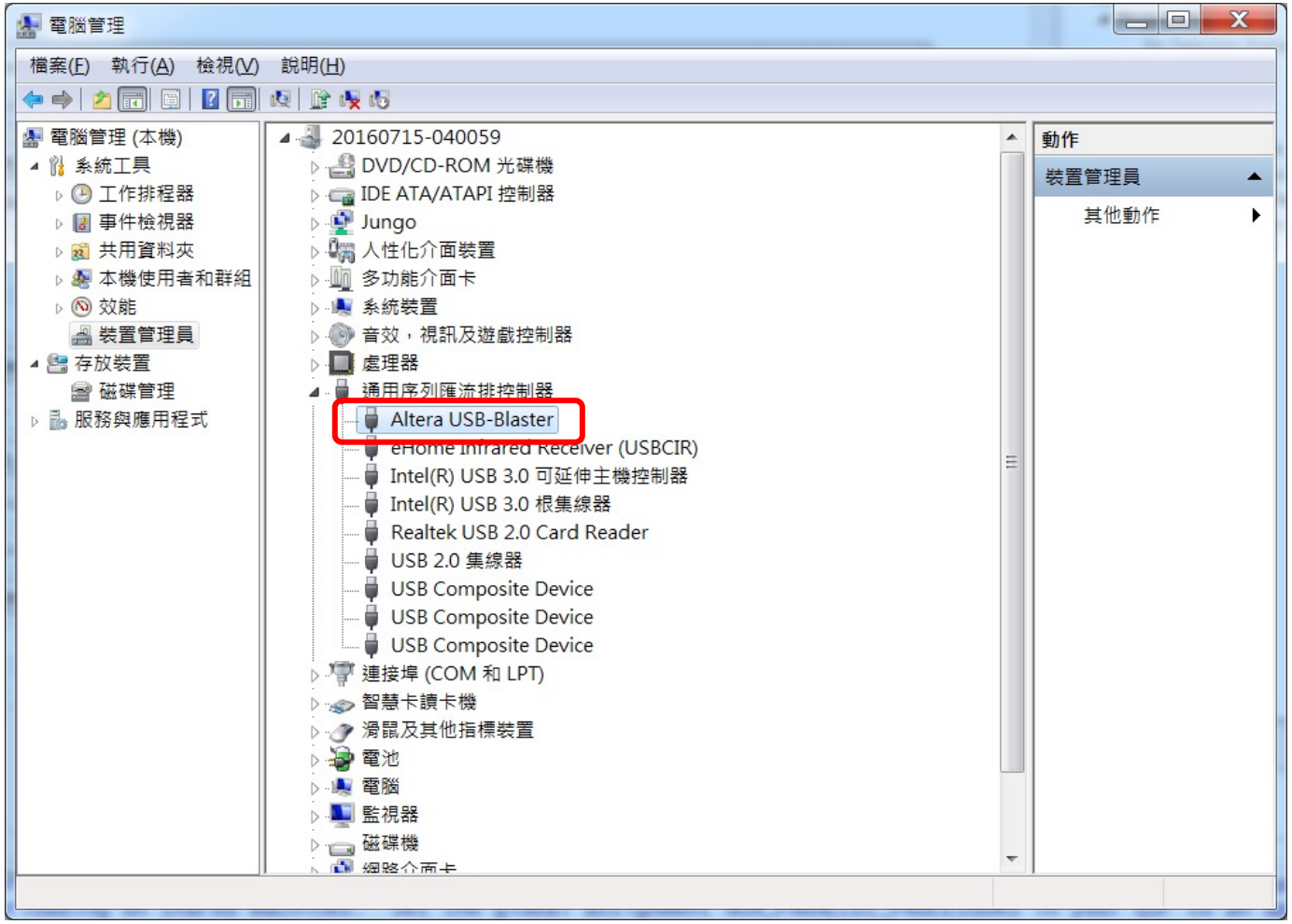
Programming DE0-CV (8/13)



Programming DE0-CV (9/13)

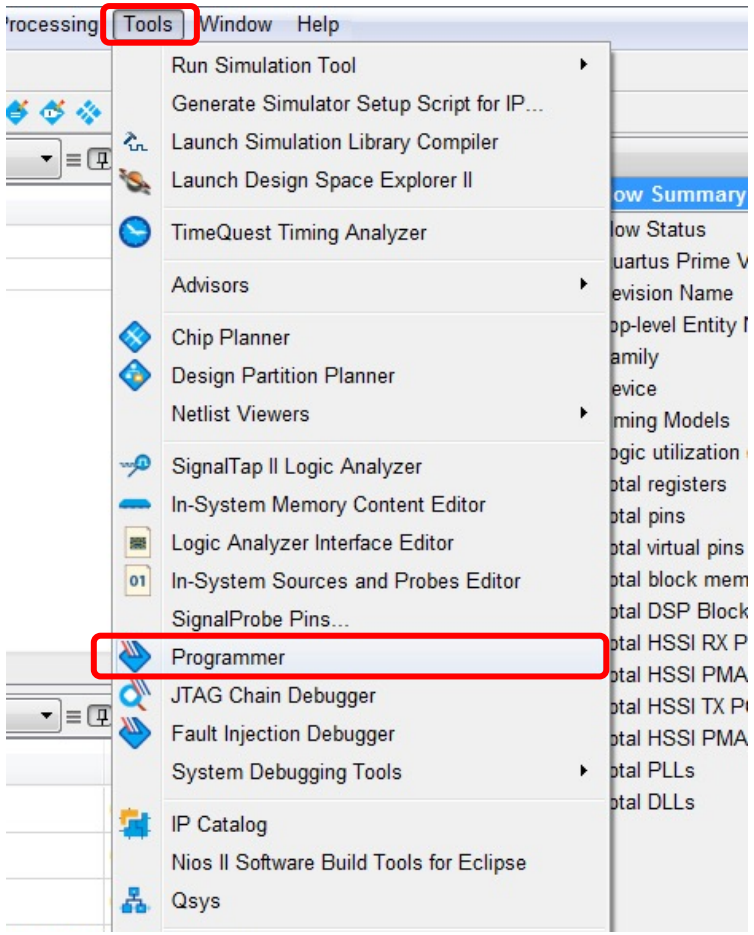


Programming DE0-CV (10/13)



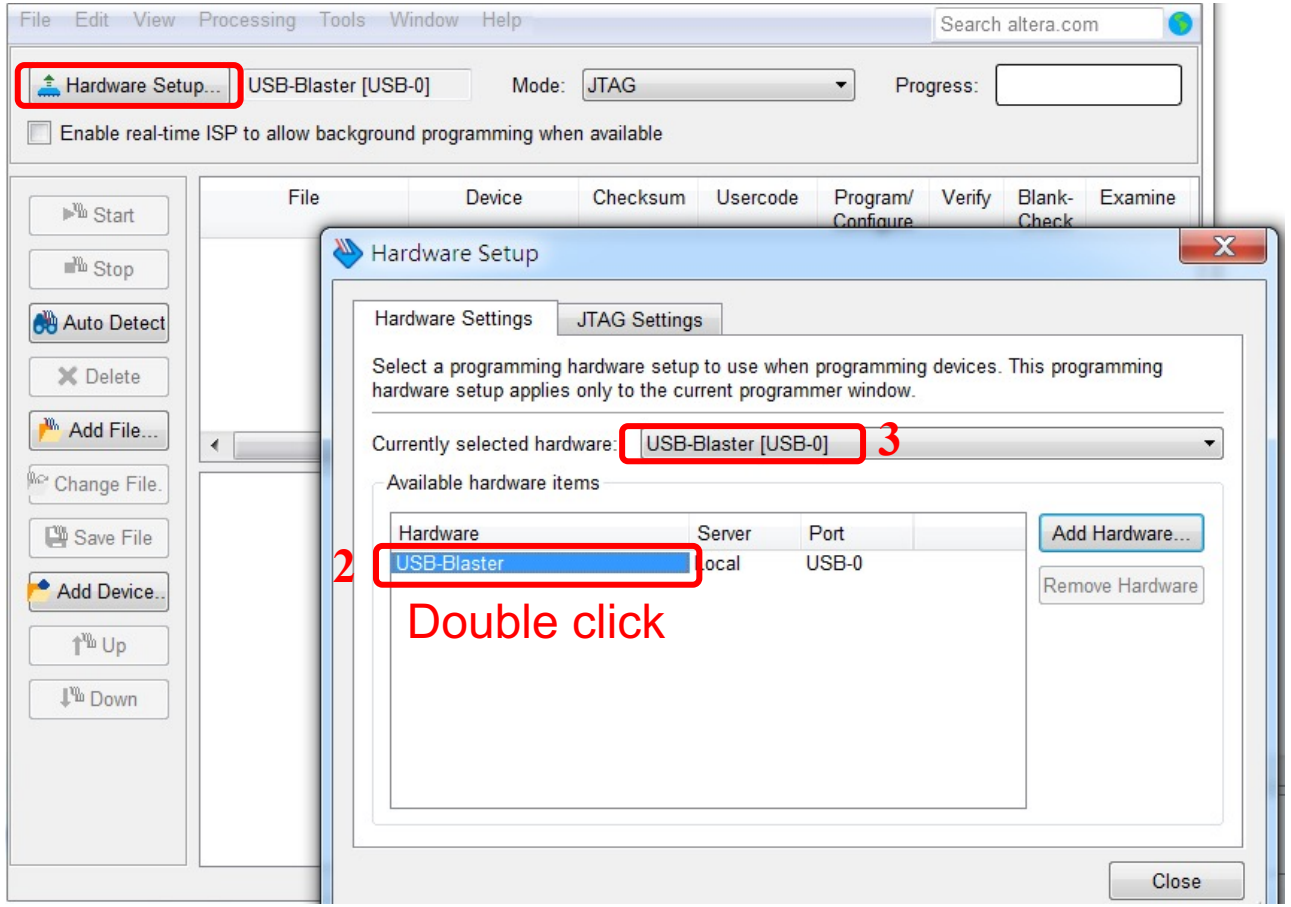
Programming DE0-CV (11/13)

- Programming device



Programming DE0-CV (12/13)

- Hardware setup: add USB-Blaster



Programming DE0-CV (13/13)

- Programming device