

利用 ModelSim 做 Verilog HDL 語法練習及驗證

簡介

本文件主要是講述如何利用 ModelSim Simulator(模擬器)來驗證 Verilog HDL 的語法及行為模式，加深對 Verilog Code 的了解。

環境建立

從雲端硬碟 Day1 的目錄”利用 ModelSim 做 Verilog HDL 語法練習及驗證_範本”中的”verilog_sim_prj_rel”目錄下載。
其該目錄中的內錄如下圖所示

名稱	修改日期	類型	大小
README.md	2022/10/28 下午 02:57	MD 檔案	1 KB
sim_setup.do	2022/10/28 下午 02:35	DO 檔案	1 KB
tb.v	2022/10/28 下午 02:35	V 檔案	1 KB

其中 tb.v 為 verilog code，可將預計要驗證的 verilog code 程式碼放入此檔案中。 sim_setup.do 為 modelsim simulator 的執行 script 檔，用來做為 modelsim simulation 流程控管。

操作流程

假設今天要觀察 continuous assignment **assign** 這個語法的行為模式，整個驗證的流程如下所示：

打開 tb.v

利用文字編輯器來開啟 tb.v 檔。如下圖所示：

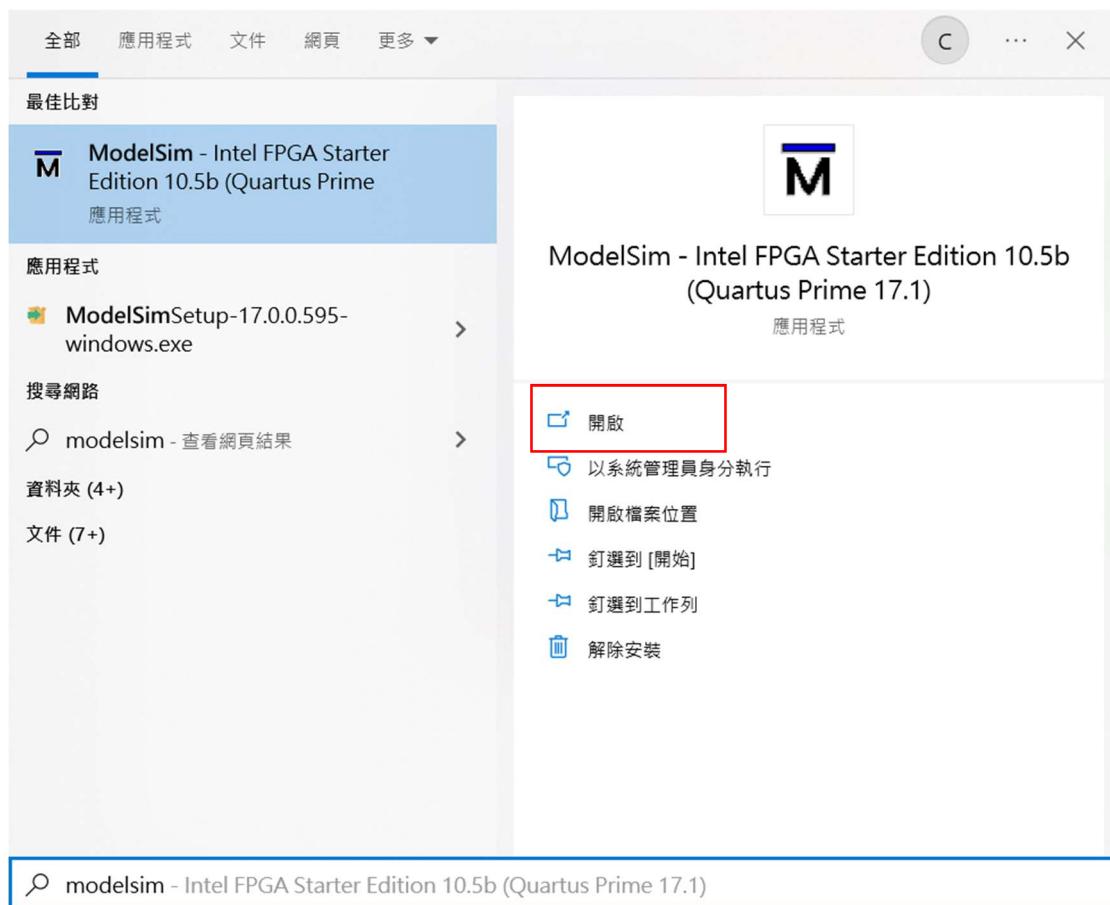
```
1 `timescale 1ns/1ns
2 module tb ();
3
4
5
6
7
8
9
10
11 initial #1000 $stop; // modelsim simulator stop condition
12 // don't delete
13 endmodule
```

於 tb.v 檔中加入預計要驗證的語法

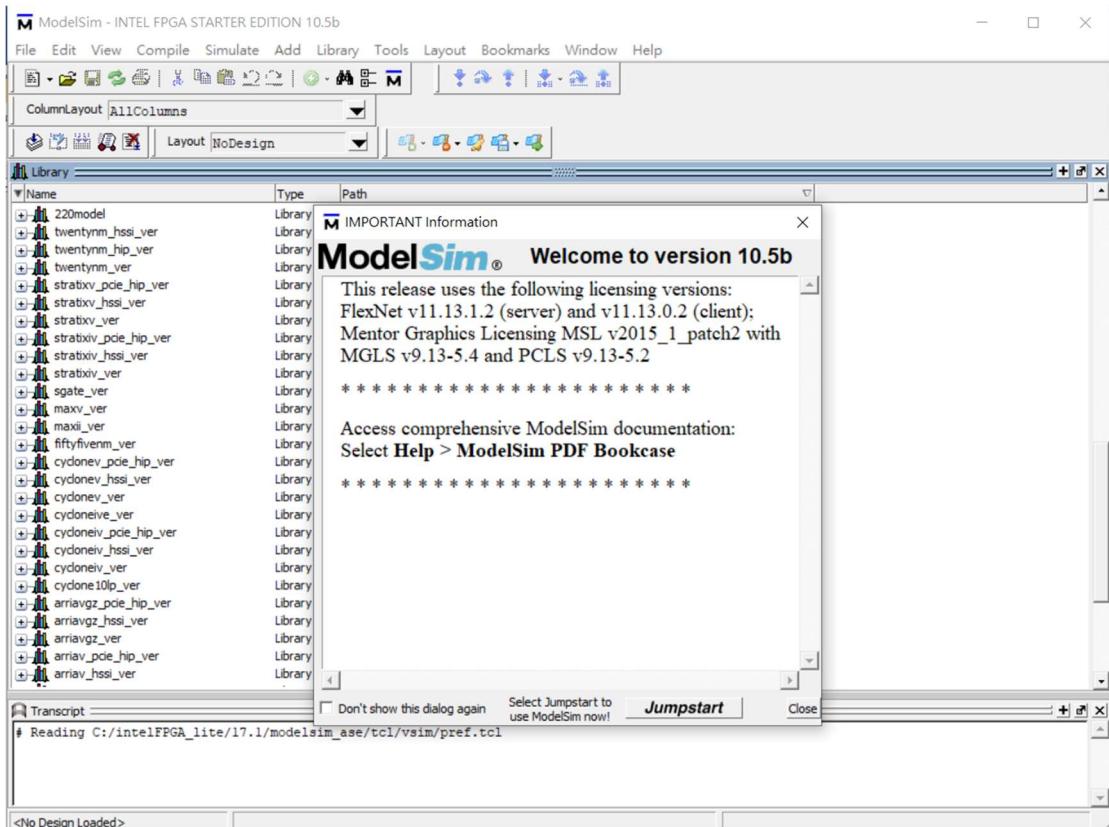
將 continuous assignment 的語法加入 tb.v 中。

```
1 `timescale 1ns/1ns
2 module tb ();
3
4 wire dato; //宣告dato為net type中的wire
5 assign dato = 1'b1; //利用continuous assignment "assign"來對
// dato做assignment。
6
7
8
9
10
11
12 initial #1000 $stop; // modelsim simulator stop condition
13 // don't delete
14 endmodule
```

開啟 Modelsim



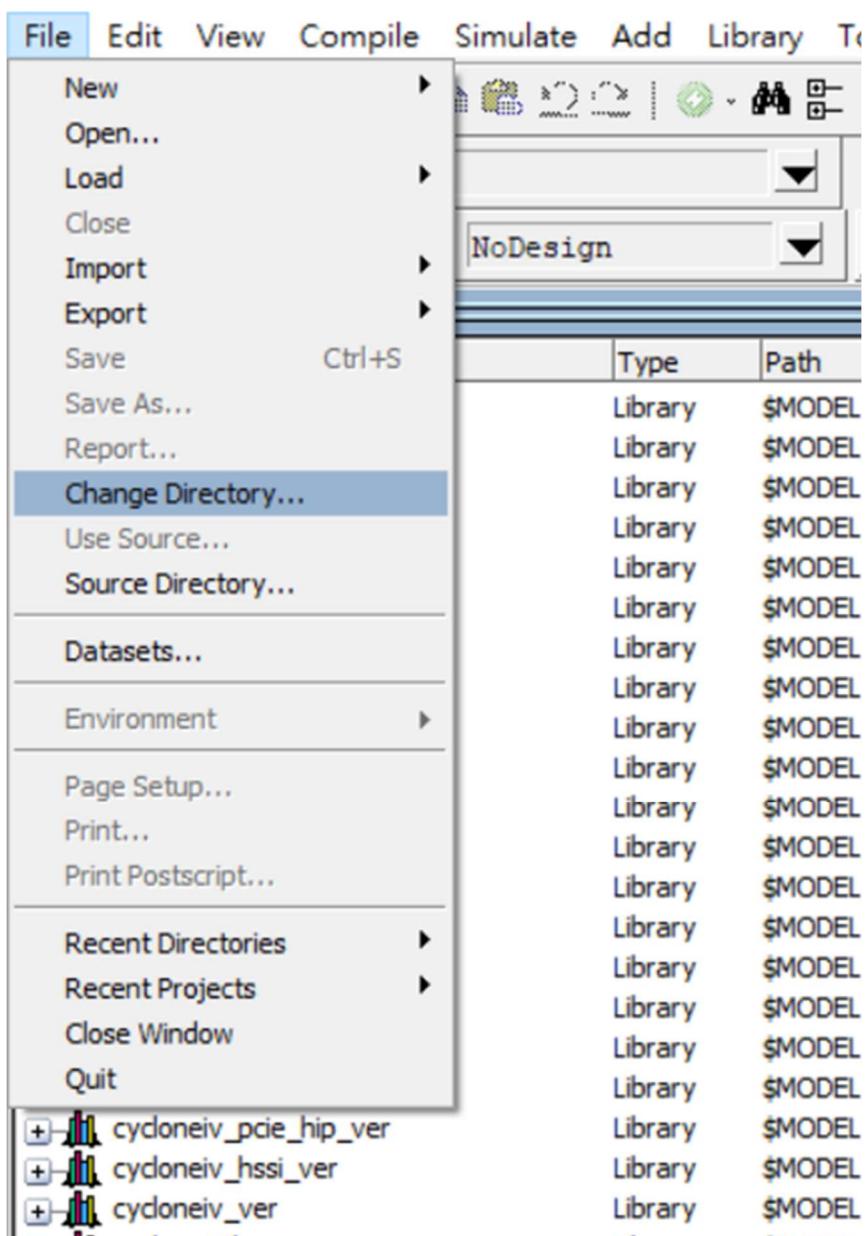
點選”開啟”如上圖紅框所示。點選後即會出現如下圖的畫面



更換 Modelsim 的工作目錄

於 Modelsim 軟體上點選 File -> Chang Directory 來改變工作目錄，如下圖所示：

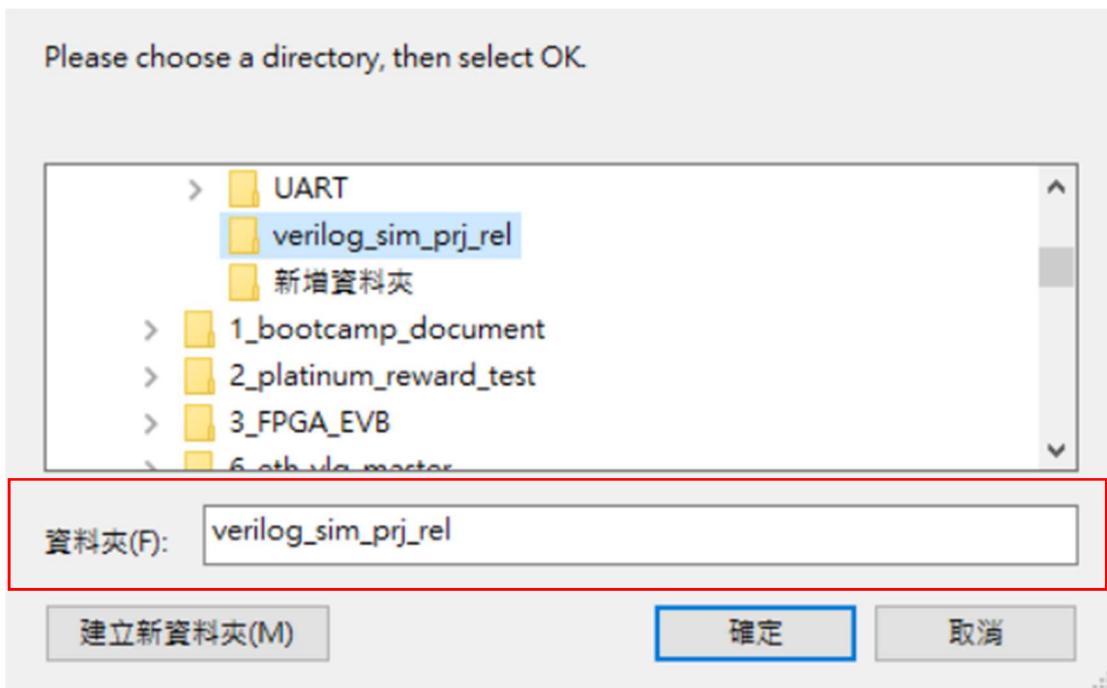
ModelSim - INTEL FPGA STARTER EDITION 10.5b



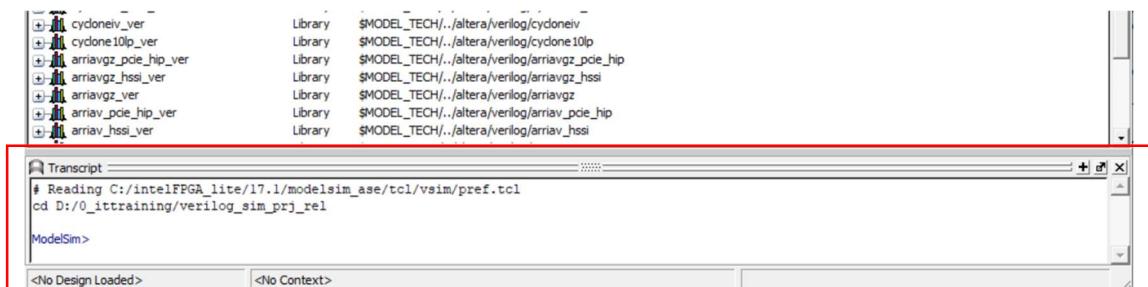
點選完後，即會出現如下的視窗

瀏覽資料夾

X



於“資料夾”的欄位中選取由雲端硬碟下載下來的 verilog_sim_prj_rel 這個目錄的路徑，如上圖紅框所示。選定後，再按“確定”，於 Transcript 的視窗(如下圖中紅框所示)，即會出現“cd D:/_PATH_/verilog_sim_prj_rel”的字樣(其中 _PATH_ 為自己電腦的路徑)，即代表切換成功。

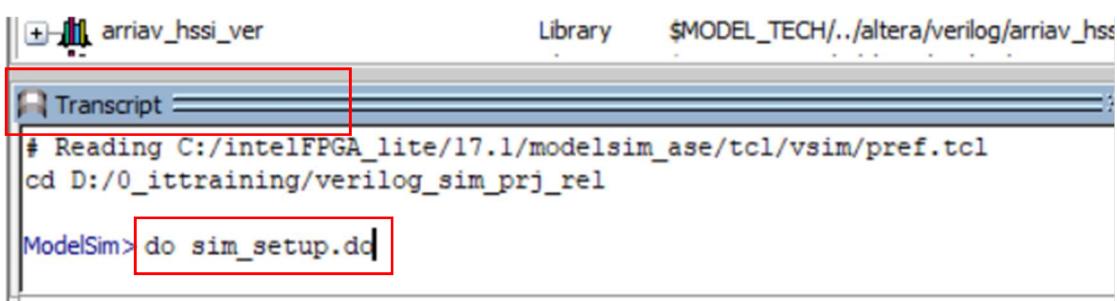


執行 modelsim script 檔

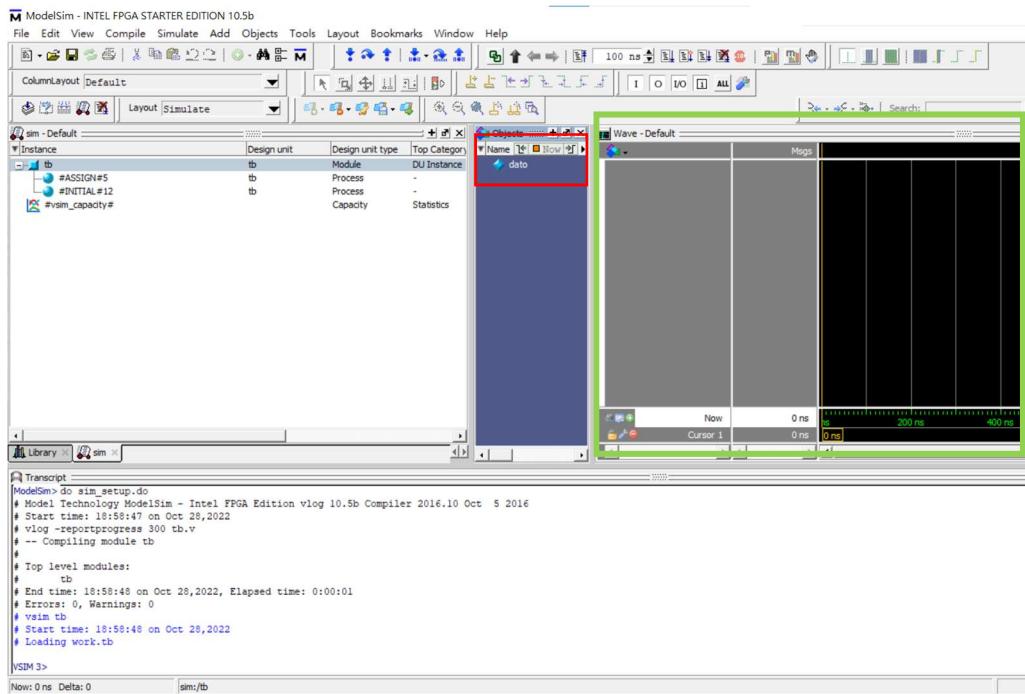
於 Transcript 視窗中鍵入下列的指令

do sim_setup.do

如下圖所示



若是成功，即會出現如下的視窗。

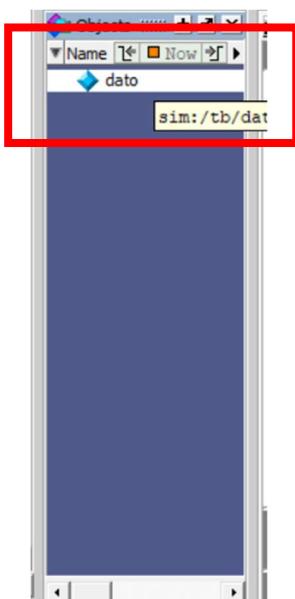


其中紅框所示為”可觀察的訊號列表”，其中為 dato。因為我們在 tb.v 中有宣告一個 net type 為 wire 的變數 dato。

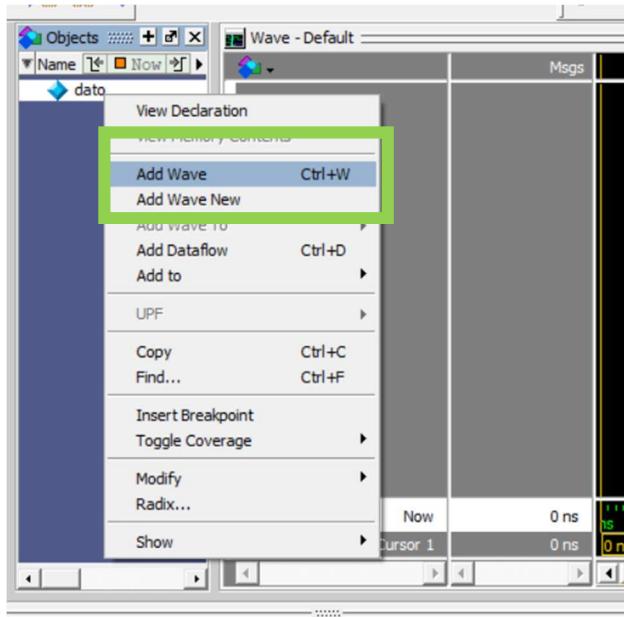
4 **wire** dato; //宣告dato為net type中的wire

綠框為變數 waveform 的觀察視窗，所以可觀察的變數 waveform 皆會在這個視窗中呈現。

加訊號至 Wave 視窗



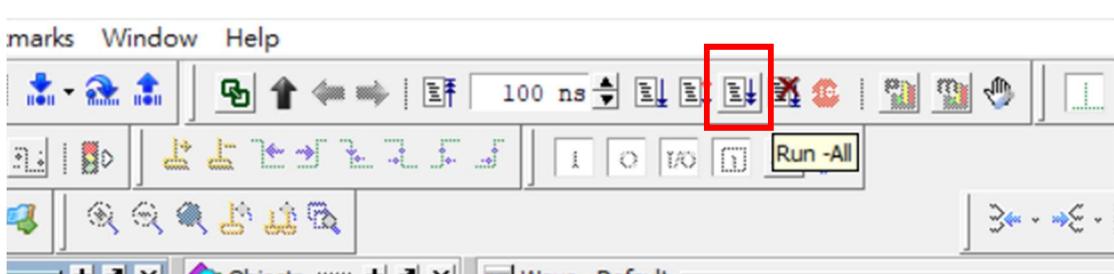
將滑鼠指標移至預計要觀察的變數如上圖紅框上,按下滑鼠右鍵,即會出現新的視窗如下圖綠框所示,



再利用滑鼠左鍵點選"Add Wave"，即會將 data 訊號加入 Wave 視窗中(如下圖綠框所示)。



開始執行模擬



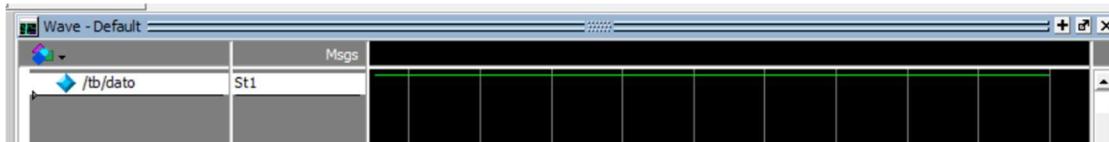
利用滑鼠左鍵點擊 Modelsim 的 icron¹，其位置如上圖紅框所示。則會出現如下視窗。

```

1 `timescale 1ns/1ns
2 module tb ();
3
4 wire dato;           // 定義資料網線型別為 wire
5 assign dato = 1'b1;  // 利用連續賦值語句 assign 對 dato 做 assignment
6                         // dato 的值為 1'b1
7
8
9
10
11
12 → initial #1000 $stop; // modelsim 模擬器停止條件
13                                // 不要刪除
14 endmodule
15
16

```

再點選上圖中 Wave 的 icon(如上圖紅框所示)，即會出現 Wave 視窗。



由上圖可以看到 dato 的訊號為 1，符合 wire 屬性的變數 dato,利用 continuous assignment assign 做值 1'b1 的 assignment。

```

5 assign dato = 1'b1; // 利用continuous assignment "assign"來對
6                         // dato做assignment。

```

重新執行模擬

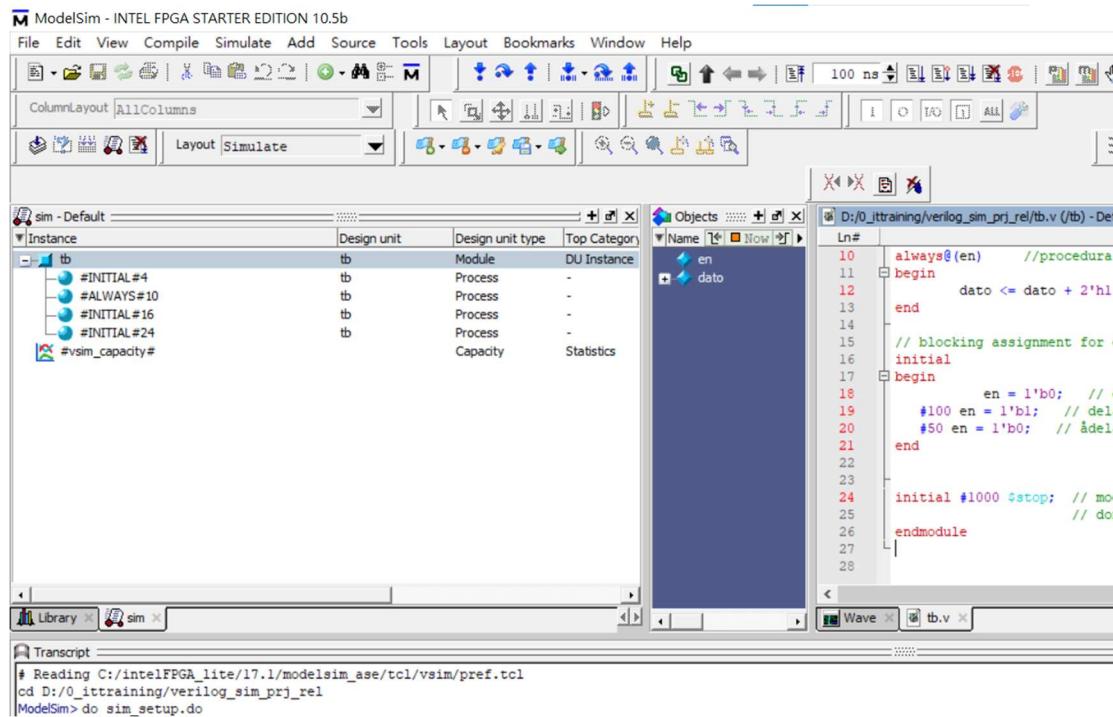
若對 tb.v 有修改時(如下圖所示)，

```

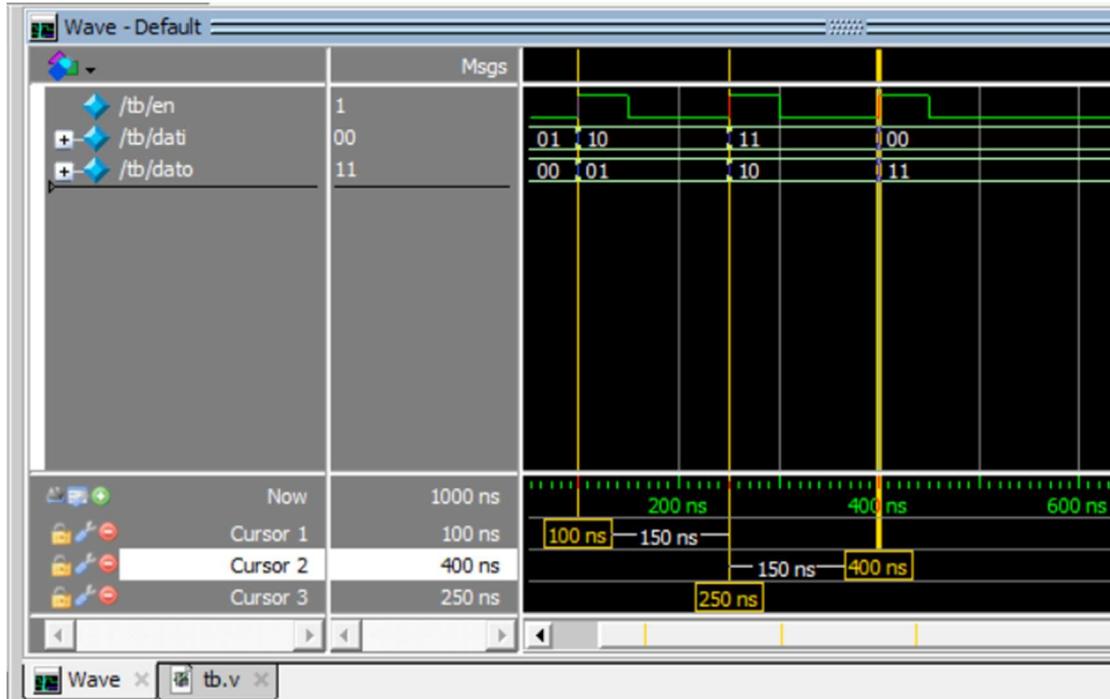
1 `timescale 1ns/1ns
2 module tb ();
3
4 reg [1:0] dato = 2'b00; //宣告 dato為varilabe type的reg,長度為2.初始值為2'b00
5 reg en; //宣告 en為varilabe type的reg,長度為1.
6 wire [1:0] dati; //宣告 dati為net type的Wire,長度為2.
7 // 當en由0到1時(posedge) ,會將RHS的運算結果
8 // 透過nonblocking procedural assignment ( <=)
9 // update到LHS的變數dato.always@(posedge en)
10 always@(posedge en) //procedural
11 begin
12     dato <= dati; // nonblocking assignment (<=).
13 end // when en 0 -> 1, dati update to dato
14 assign dati = dato + 2'h1;
15 // blocking assignment for en assignment .
16 initial
17 begin
18     en = 1'b0; // en初始值設定
19     #100 en = 1'b1; // delay 100 unit(ns)後,將en assign為1'b1 (@100ns)
20     #50 en = 1'b0; // 再delay 50 unit(ns)後,將en assign為1'b0
21     #100 en = 1'b1; // delay 100 unit(ns)後,將en assign為1'b1 (@250ns)
22     #50 en = 1'b0; // 再delay 50 unit(ns)後,將en assign為1'b0
23     #100 en = 1'b1; // delay 100 unit(ns)後,將en assign為1'b1 (@400ns)
24     #50 en = 1'b0; // 再delay 50 unit(ns)後,將en assign為1'b0
25 end
26
27
28 initial #1000 $stop; // modelsim simulator stop condition
29 // don't delete
30 endmodule

```

待修改完畢後，重新從“執行 modelsim script 檔”步驟開始做至“開始執行模擬”，結果如下圖所示：



觀察 Wave 視窗，即可以發現 dato[1:0]變數，在 en 訊號由 0 變為 1 的 edge 變化時，其值即會加 1，而其他的時間，維持原值不變。而此就是 variable type 為 reg 會儲存數值的特性，換句話說，dato[1:0]會保持數值，只到 en 產生 positive edge 變化時，才會更新內容值。



由上圖可以看到

- 0ns ~ 100ns : dato[1:0] 為 2'b00, 且 dati 值為 2'b01.
- @100ns : en 由 0 到 1, dati 值 2'b01 更新至 dato 且 dati 值為 2'b10.
- 100ns ~ 250ns : dato[1:0] 維持 2'b01
- @250ns : en 由 0 到 1, dati 值 2'b10 更新至 dato 且 dati 值為 2'b11.
- 250ns ~ 400ns : dato[1:0] 維持 2'b10
- @400ns : en 由 0 到 1, dati 值 2'b11 更新至 dato, 且 dati 值為 2'b00.

將 Verilog code 由 Quartus 軟體產生 FPGA 的燒錄檔(Verilog Code 要使用可 合成的語法)

利用 Quartus 產生教學開發 FPGA 板(Terasic DE10 Nano Development Kit)的燒錄檔首先要確認 Verilog Code 是為可合成的語法。D10 Nano Development Kit 的介面有振盪器(osc), 指撥開關(Switch), 按鈕(button), 8 顆 LED。其中 OSC 可以當成觸發源，指撥開關和按鈕可以當輸入源，8 顆 LED 可以當輸出訊號顯示裝置。

將 Verilog Code 由 Quartus 軟體轉成 FPGA 的燒錄檔(sof)的流程如下

下載範本

從雲端硬碟 Day1 的目錄”利用 ModelSim 做 Verilog HDL 語法練習及驗證_範本”中的”Golden_top”目錄下載。下載下來後會有一個 Golden 的目錄，其中的內容如下所示：

名稱	修改日期	類型	大小
DE10_Nano_golden_top.qpf	2022/10/28 上午 01:28	QPF 檔案	1 KB
DE10_Nano_golden_top.qsf	2022/10/28 上午 01:28	QSF 檔案	4 KB
DE10_Nano_golden_top.sdc	2022/10/28 上午 01:28	SDC 檔案	3 KB
DE10_Nano_golden_top.v	2022/10/28 上午 01:28	V 檔案	1 KB

DE10_Nano_golden_top.qpf : Quatus Project file

DE10_Nano_golden_top.qsf : Quatus Project setting file

DE10_Nano_golden_top.sdc : Quatus Project timing constraint file

DE10_Nano_golden_top.v : Quatus Project Verilog Code

加入 Verilog Code 於範本中

例如我們想要驗證 continuous assignment assign 的語法來點亮 FPGA 板上的 8 顆 LED 燈。所以先利用文件編輯器開啟檔案” DE10_Nano_golden_top.v”,其內容如下圖所示：

```

1  module DE10_Nano_golden_top(
2      ////////////// FPGA ///////////
3      input          FPGA_CLK1_50,    //振盪器
4      input          FPGA_CLK2_50,    //振盪器
5      input          FPGA_CLK3_50,    //振盪器
6      ////////////// KEY ///////////
7      input [1:0]    KEY,           // 按鈕
8
9      ////////////// LED ///////////
10     output [7:0]   LED,           // 8顆LED
11
12     ////////////// SW ///////////
13     input [3:0]    SW,            // 4個指撥開關
14 );
15
16
17
18
19
20
21
22 endmodule

```

我們加入 continuous assignment assign 的語法來讓 8 顆 LED 燈全亮。

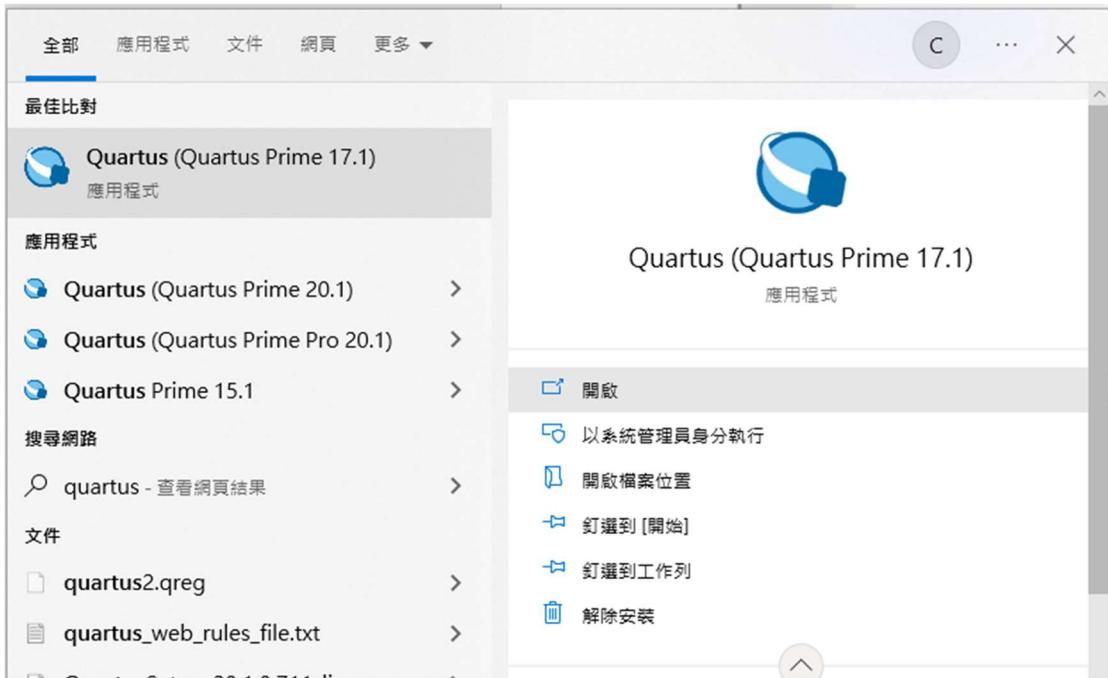
整個 Verilog Code 如下圖所示:

```

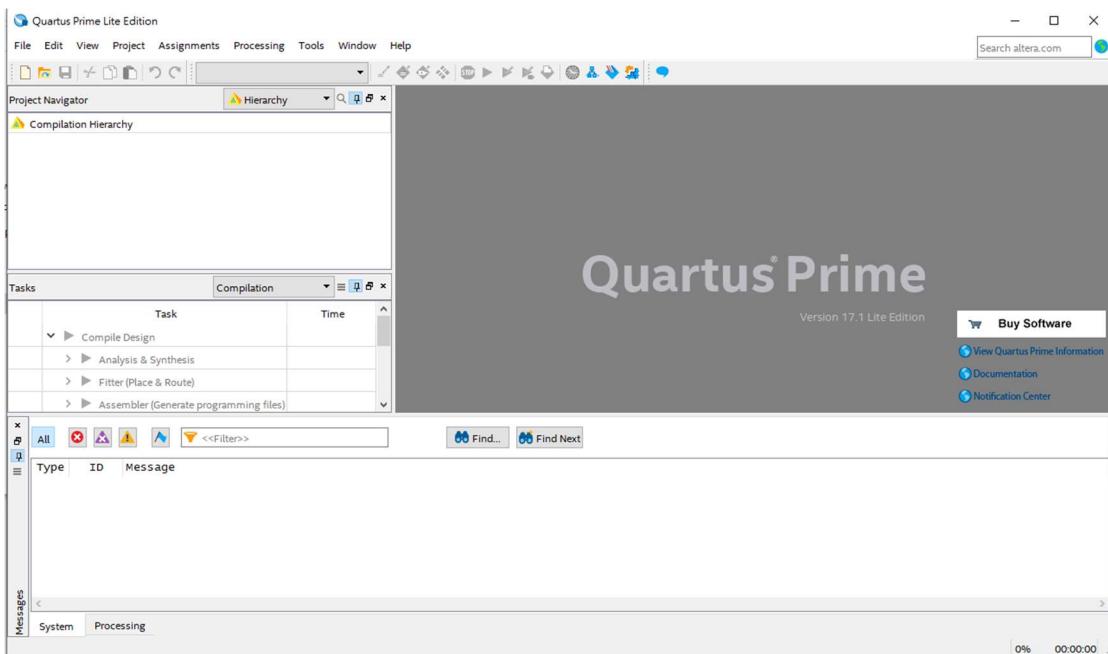
1  module DE10_Nano_golden_top(
2      ////////////// FPGA ///////////
3      input          FPGA_CLK1_50,    //振盪器
4      input          FPGA_CLK2_50,    //振盪器
5      input          FPGA_CLK3_50,    //振盪器
6      ////////////// KEY ///////////
7      input [1:0]    KEY,           // 按鈕
8
9      ////////////// LED ///////////
10     output [7:0]   LED,           // 8顆LED
11
12     ////////////// SW ///////////
13     input [3:0]    SW,            // 4個指撥開關
14 );
15
16
17
18
19     assign LED = 8'b11111111;    // 讓8顆LED全亮
20
21
22 endmodule

```

開啟 Quartus

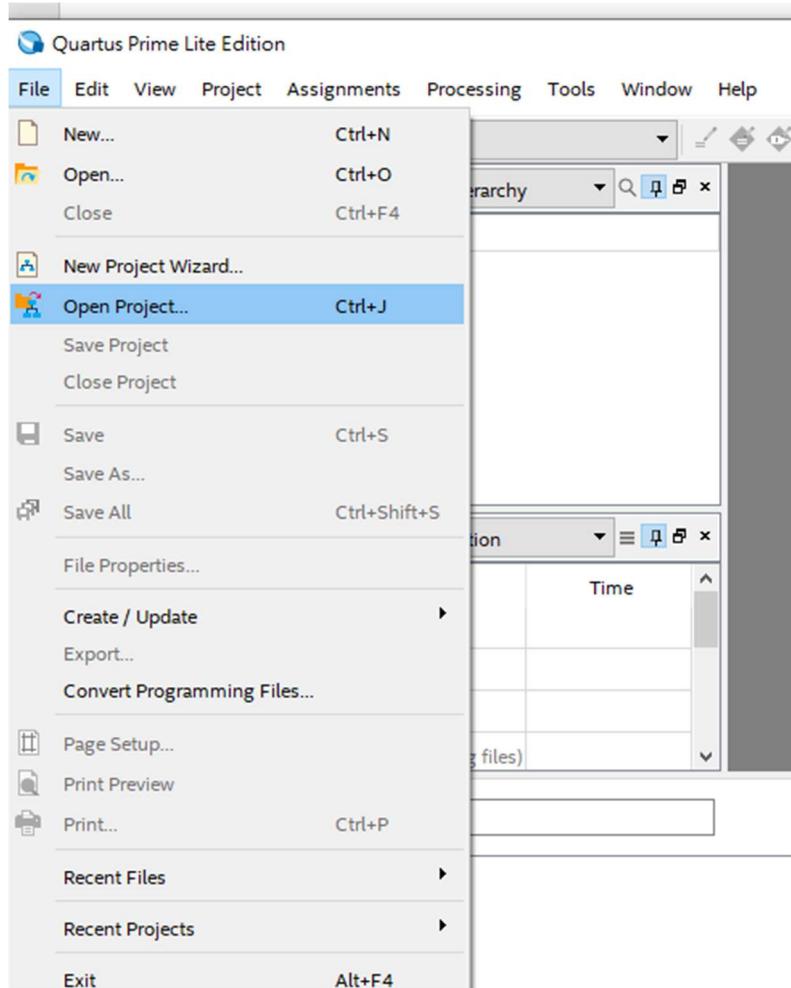


點選上圖中所示的”開啟”來開始 Quartus(Quartus Prime 17.1)
其圖如下所示：

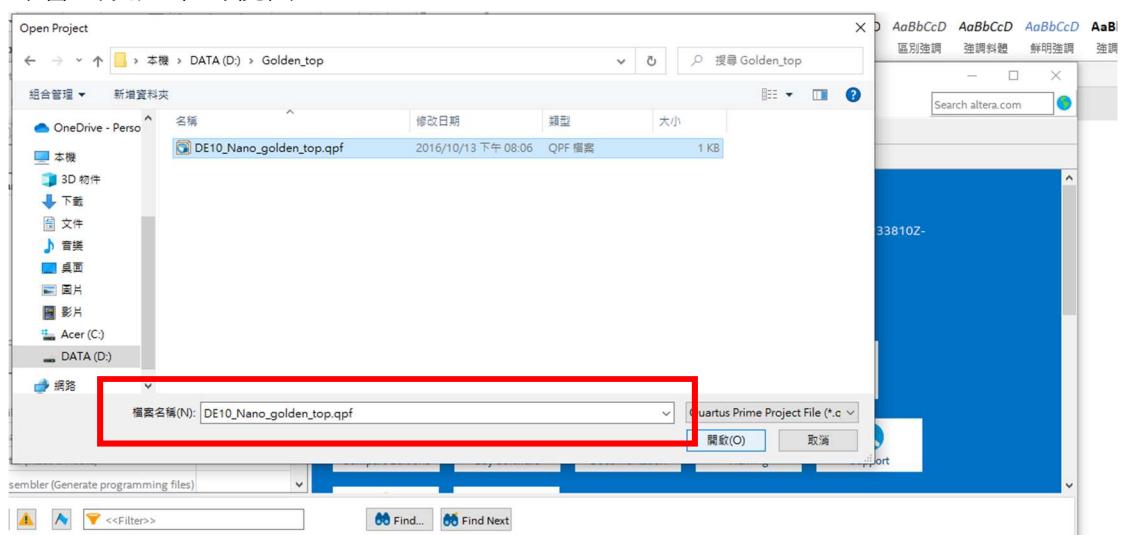


開啟 Project

點選 File -> Open Project(如下圖所示)

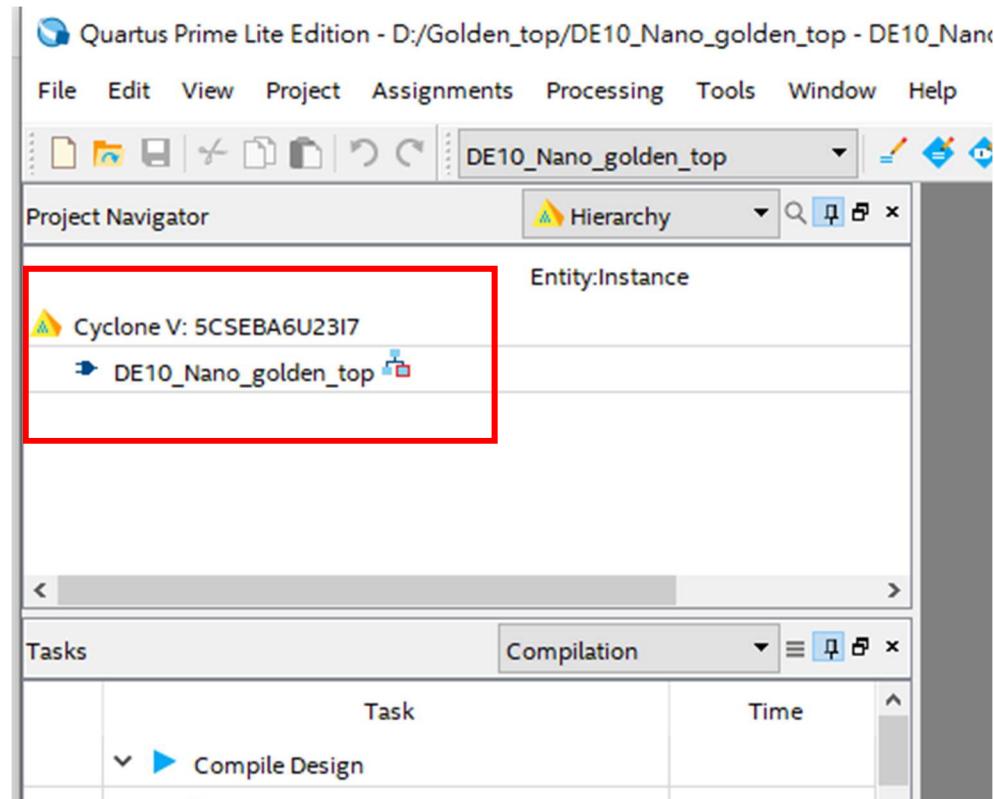


即會出現如下的視窗



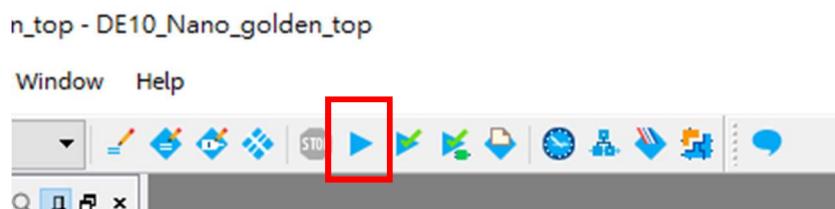
在檔案名稱(如上圖紅框所示)點選由雲端硬碟下載下來的 Golden_top 目錄中

的"DE10_Nano_golden_top.qsf".待選擇完畢後,再點選"開啟"，接著應會在 Project Navigator 的視窗中看到 FPGA Device 為 Cyclone V : 5CSEBA6U23I7 及 Top module Name : DE10_Nano_Golden_top，如下圖紅框標示所示。

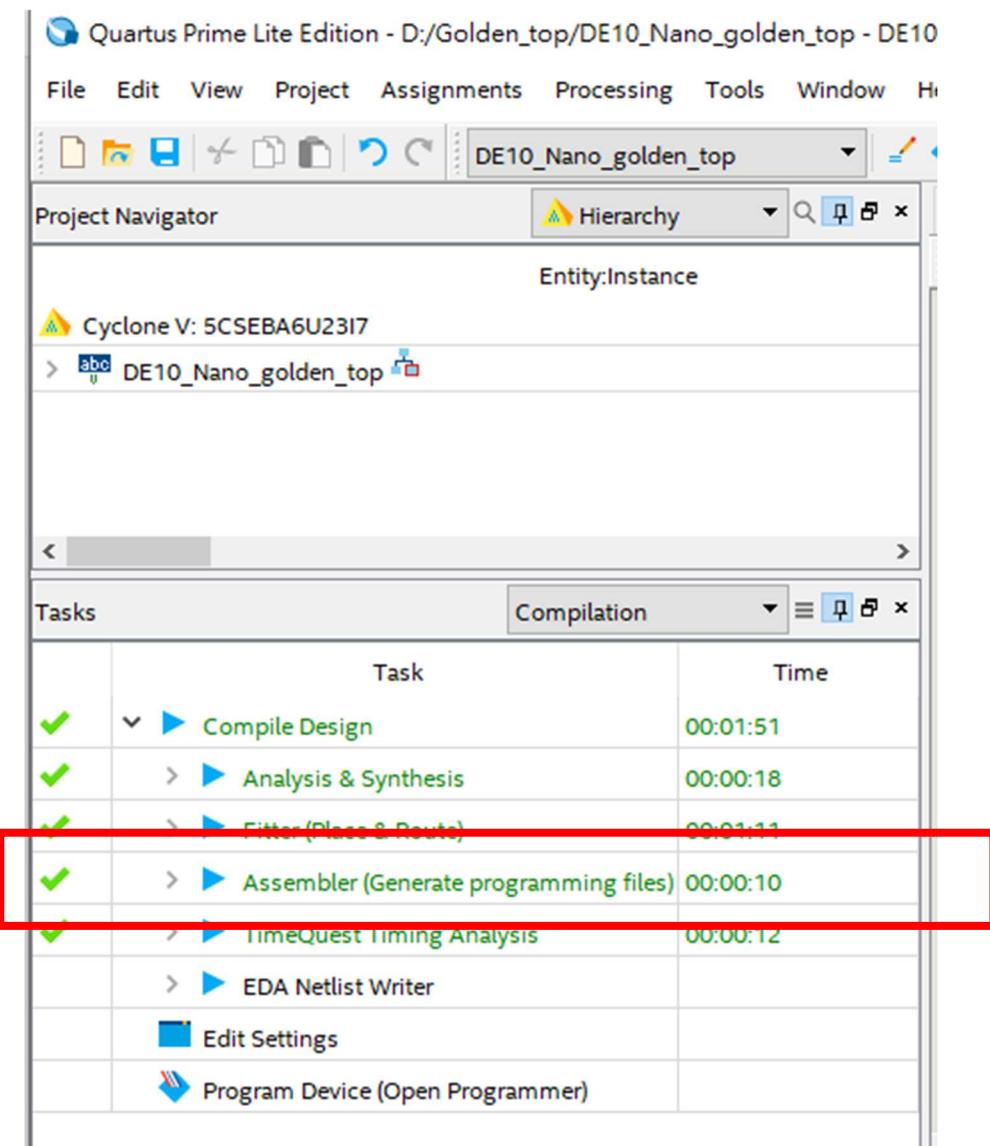


產生 FPGA 的燒錄檔(SOF)

利用滑鼠左鍵點選 Quartus 軟體上的 cron，其位置如下圖紅框標示。



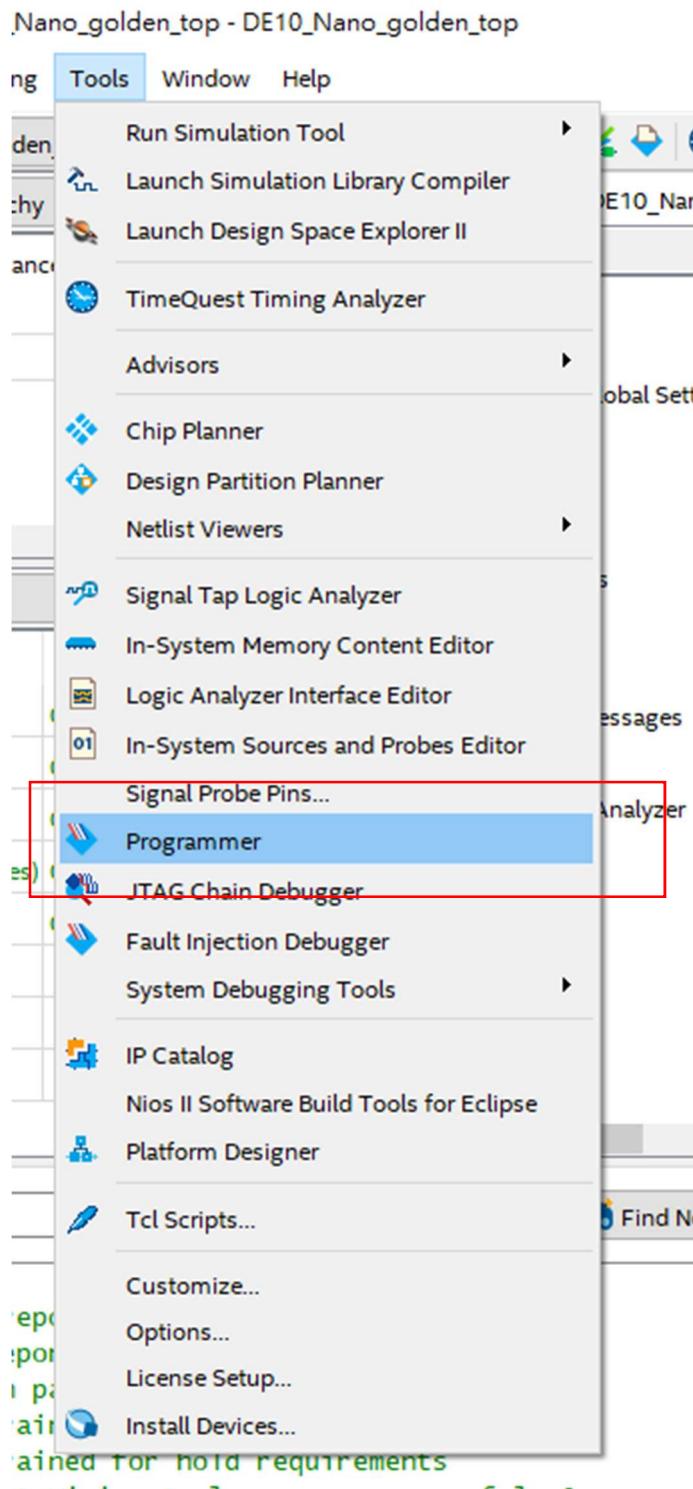
等待數分鐘後，當 Assembler(Generate Programming files)呈現 100%時，即代表 Quartus 成功產生 FPGA 的燒錄檔。(如下圖紅框標示所示)



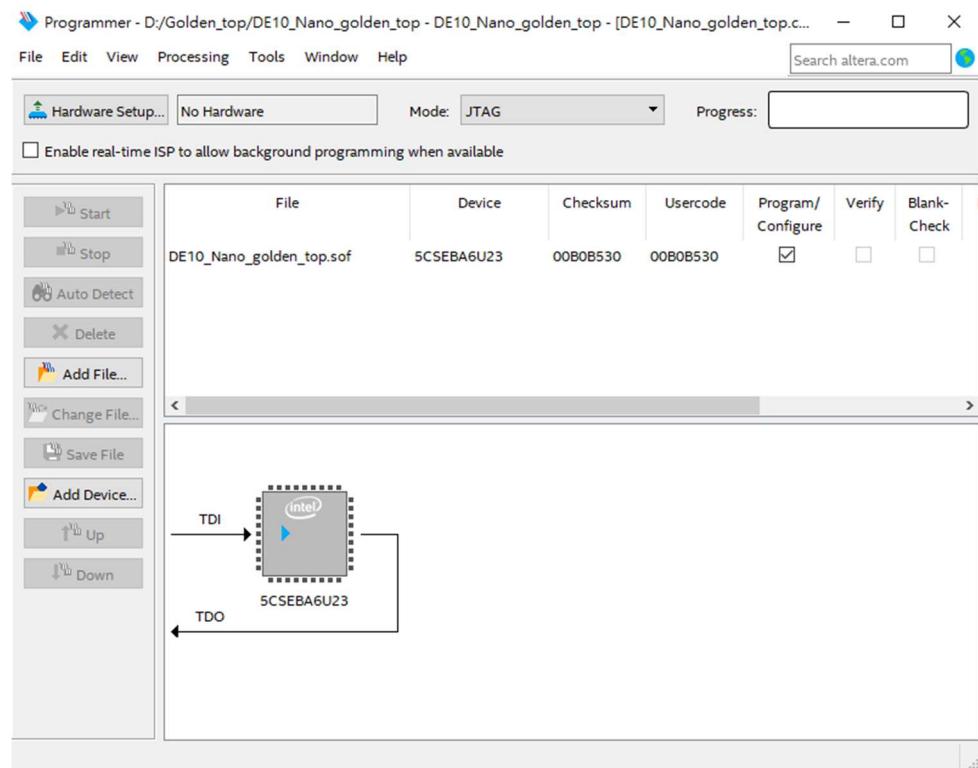
下載 FPGA 燒錄檔至 DE10 Nano

Development Kit

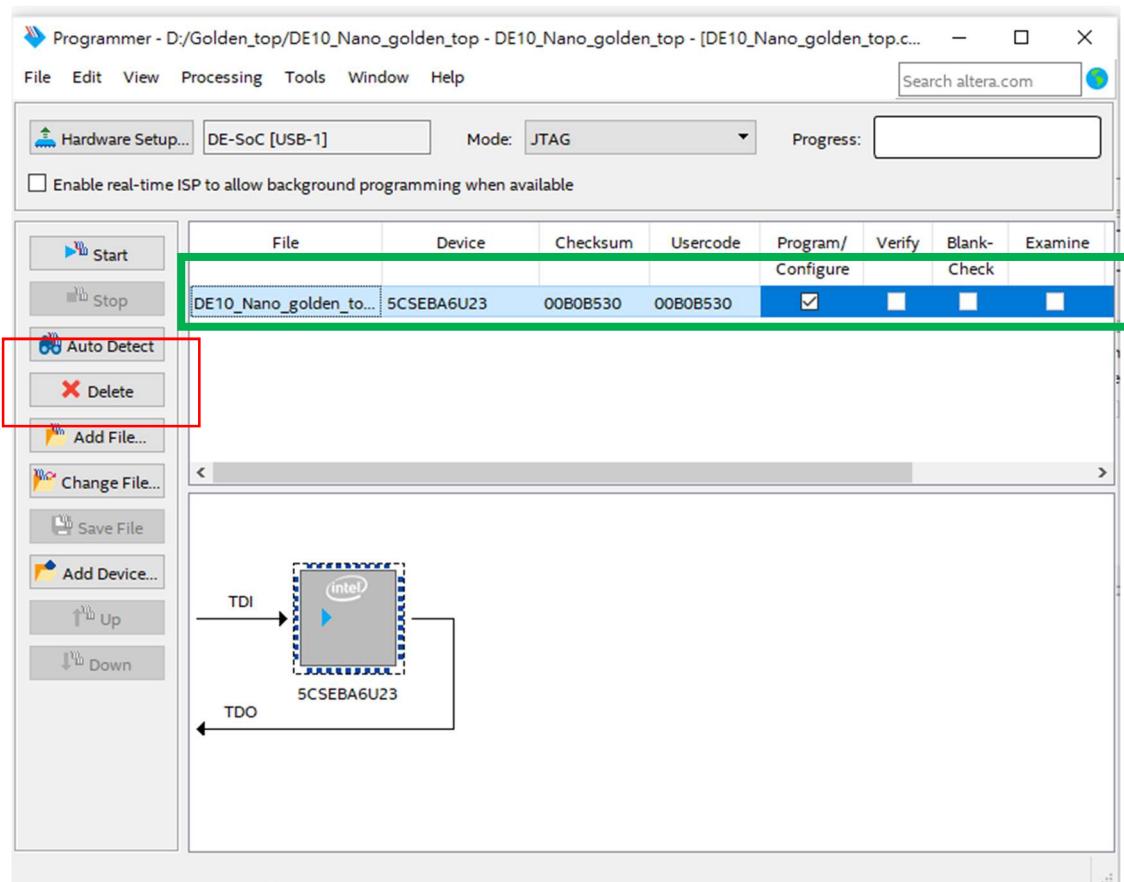
點選 Tools → Programmer



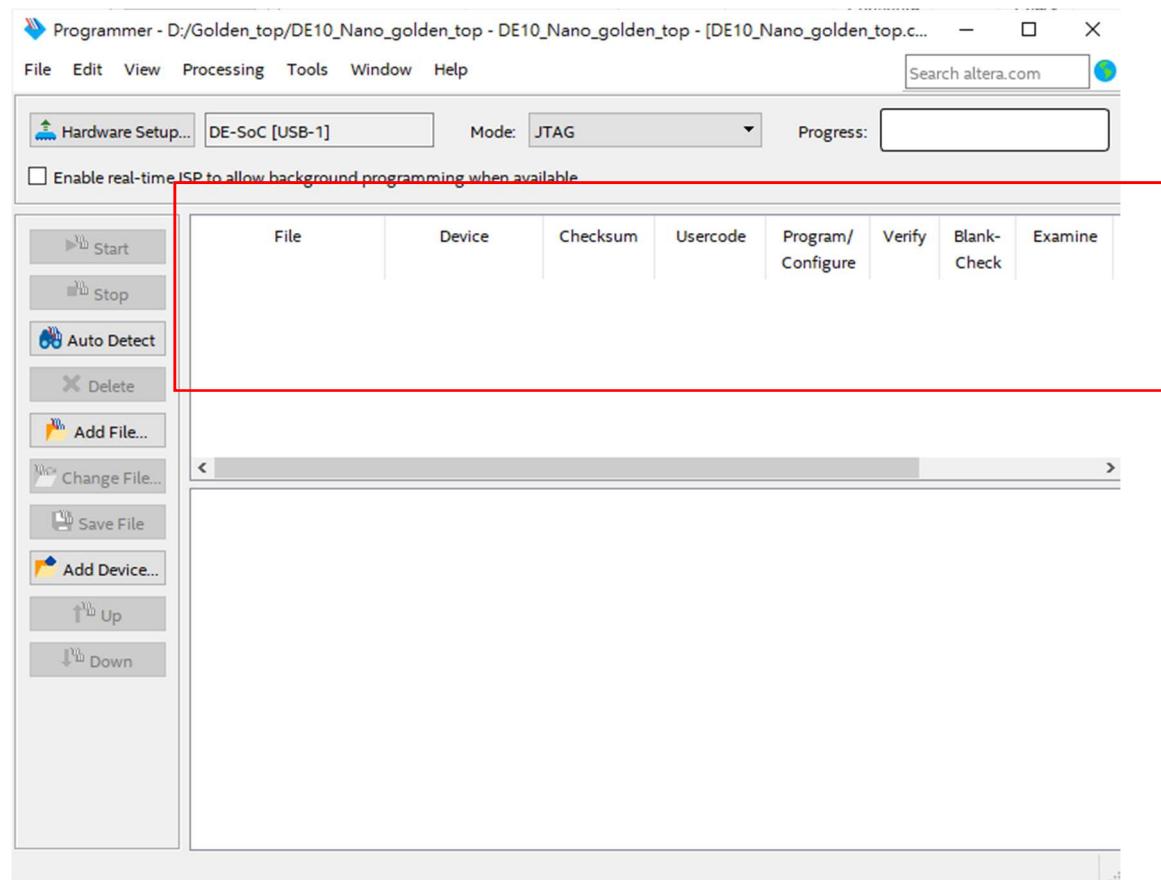
畫面如下圖所示：



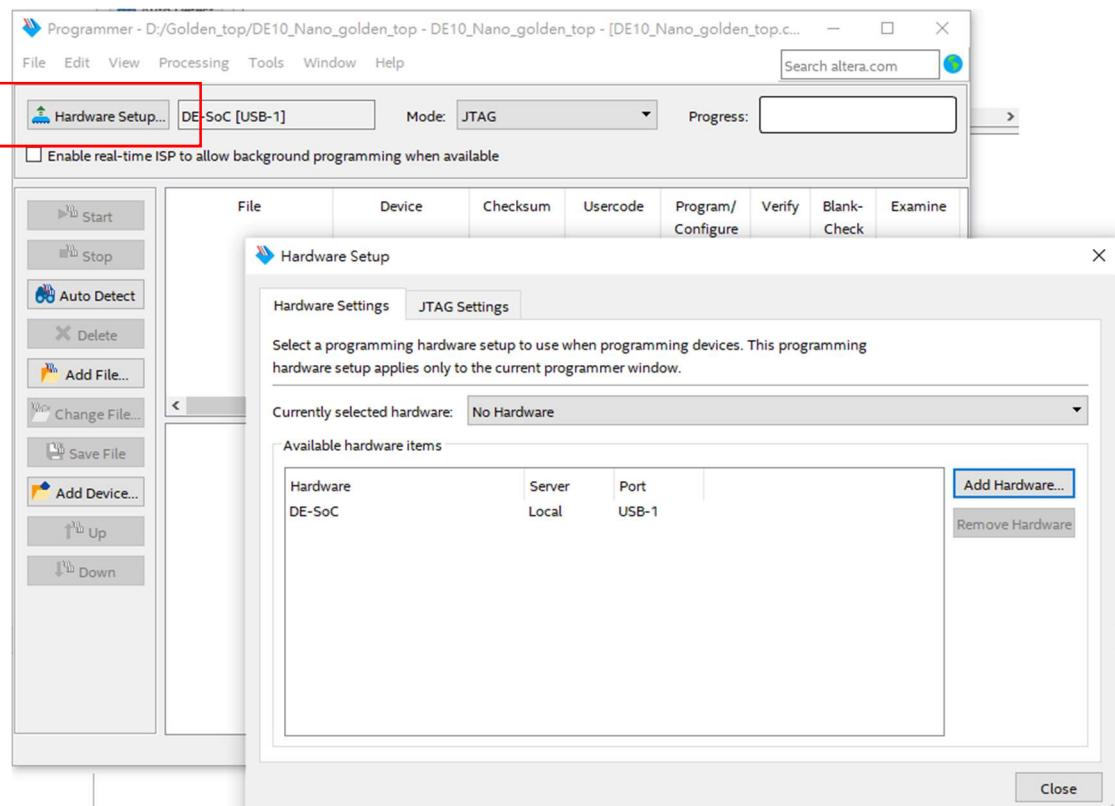
先用滑鼠左鍵點選如下圖綠色方框所示的位置，再用滑鼠左鍵點上"Delete" 選紅框所示



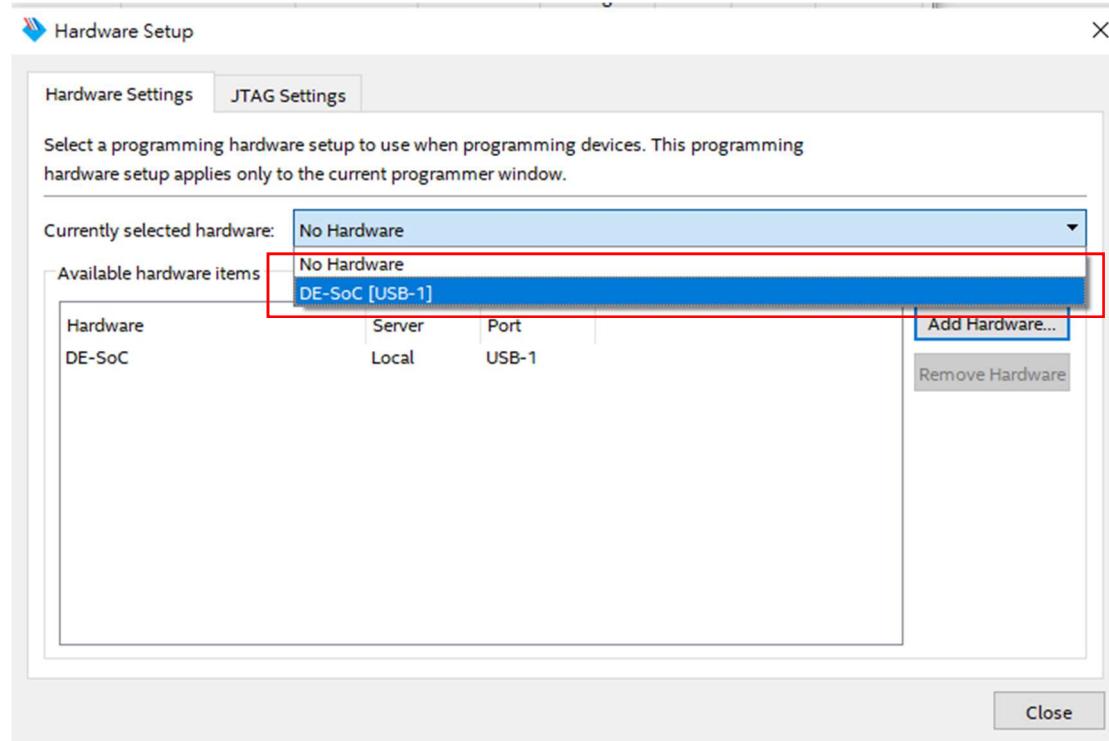
其結果應如下圖所示，於紅框所標示處的位置應沒有任何東西。



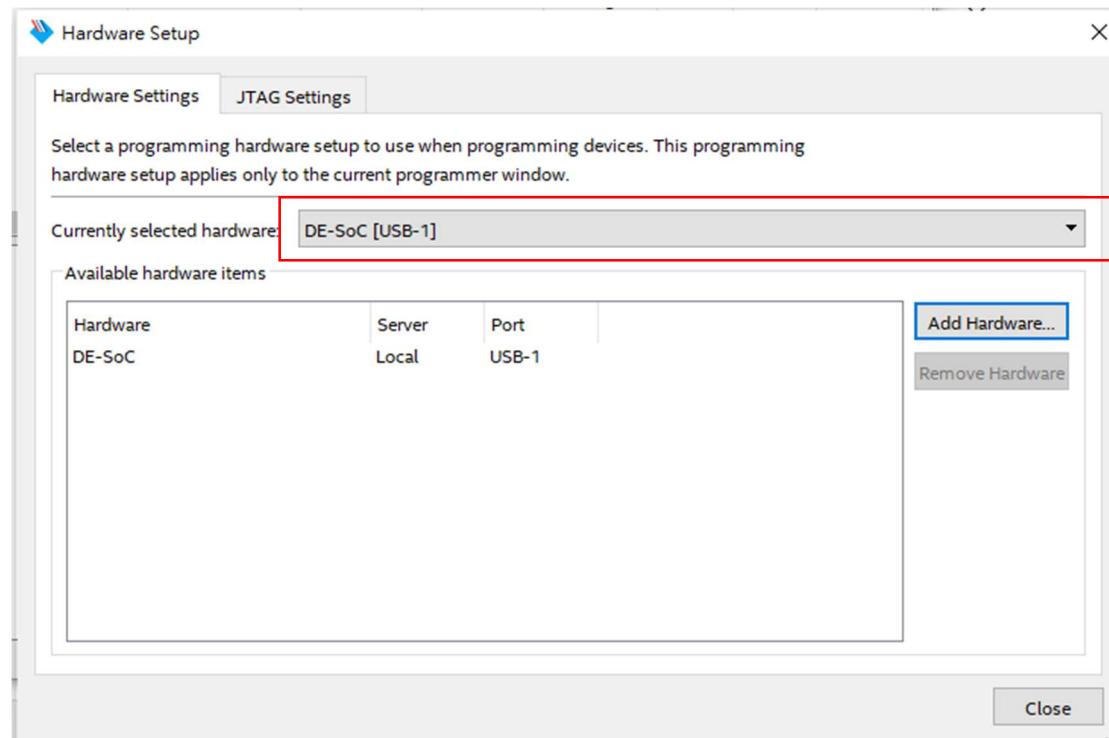
用滑鼠左鍵點選“Hardware Setup”如下圖紅框所示：



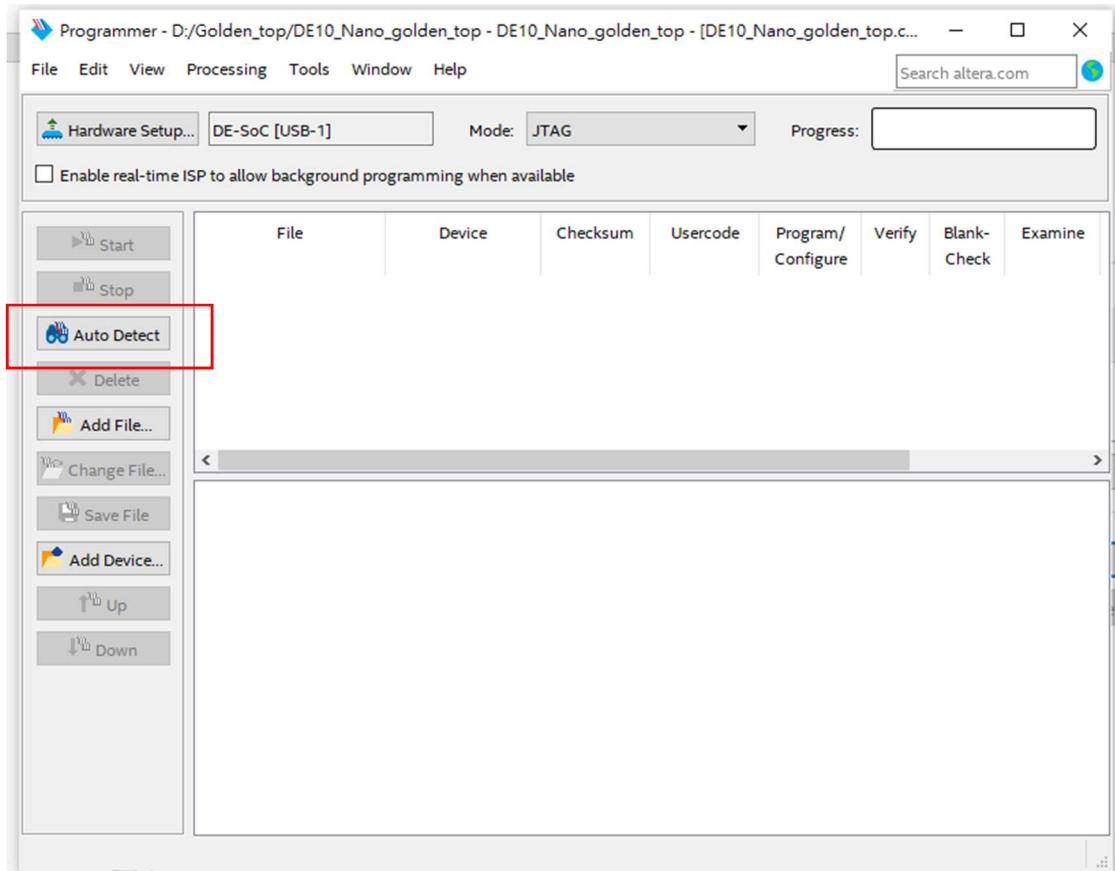
於 Hardware Setup 視窗處，在 Available hardware items 欄位，選取”DE-SoC[USB-1]”如下圖紅框所示：



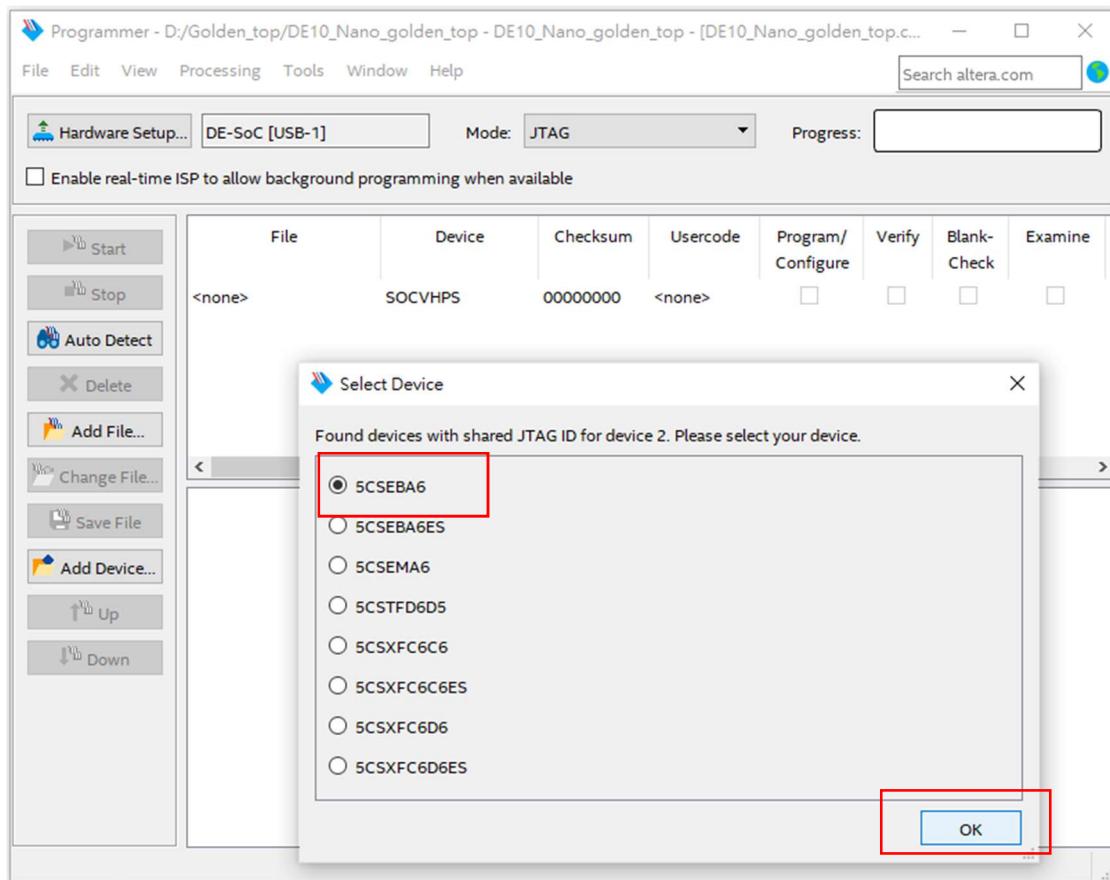
其結果如下圖所示，再點選”Close”，完成 download cable 的設定。



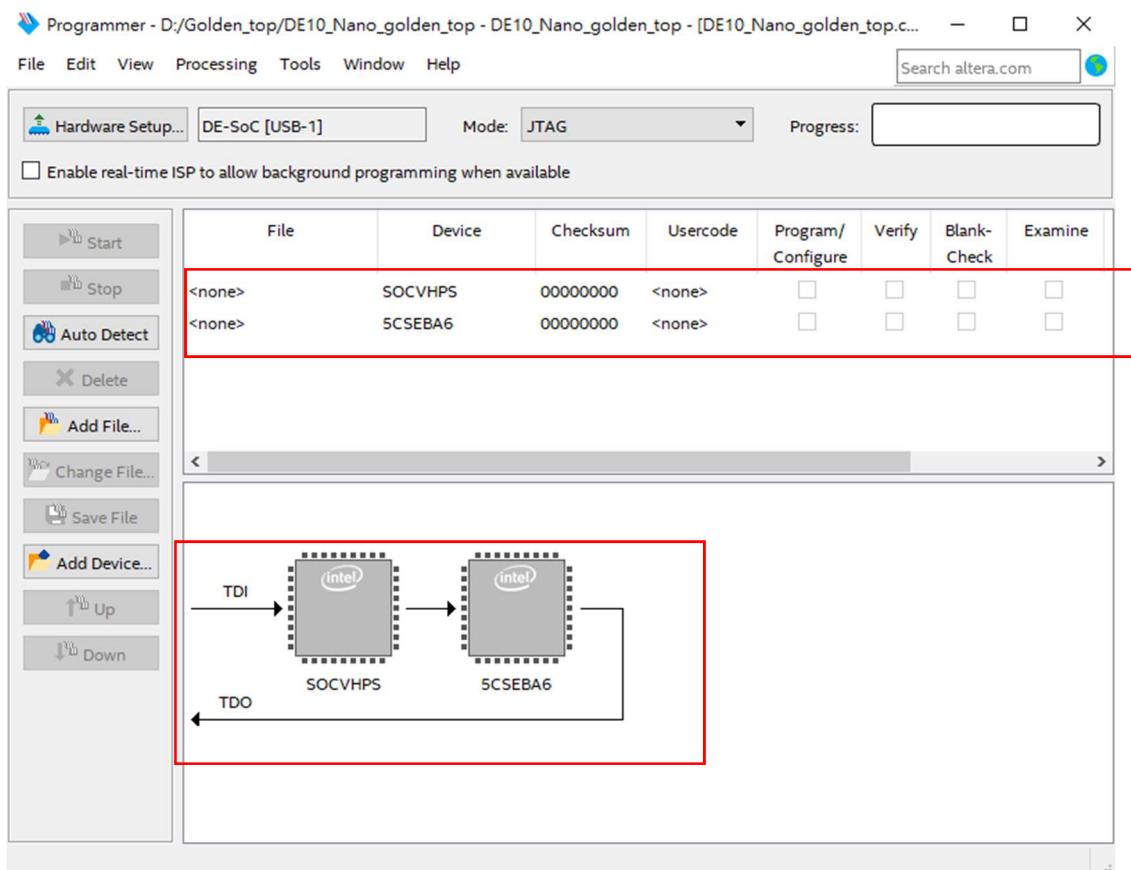
點選“Auto Detect”如下圖紅框所示：



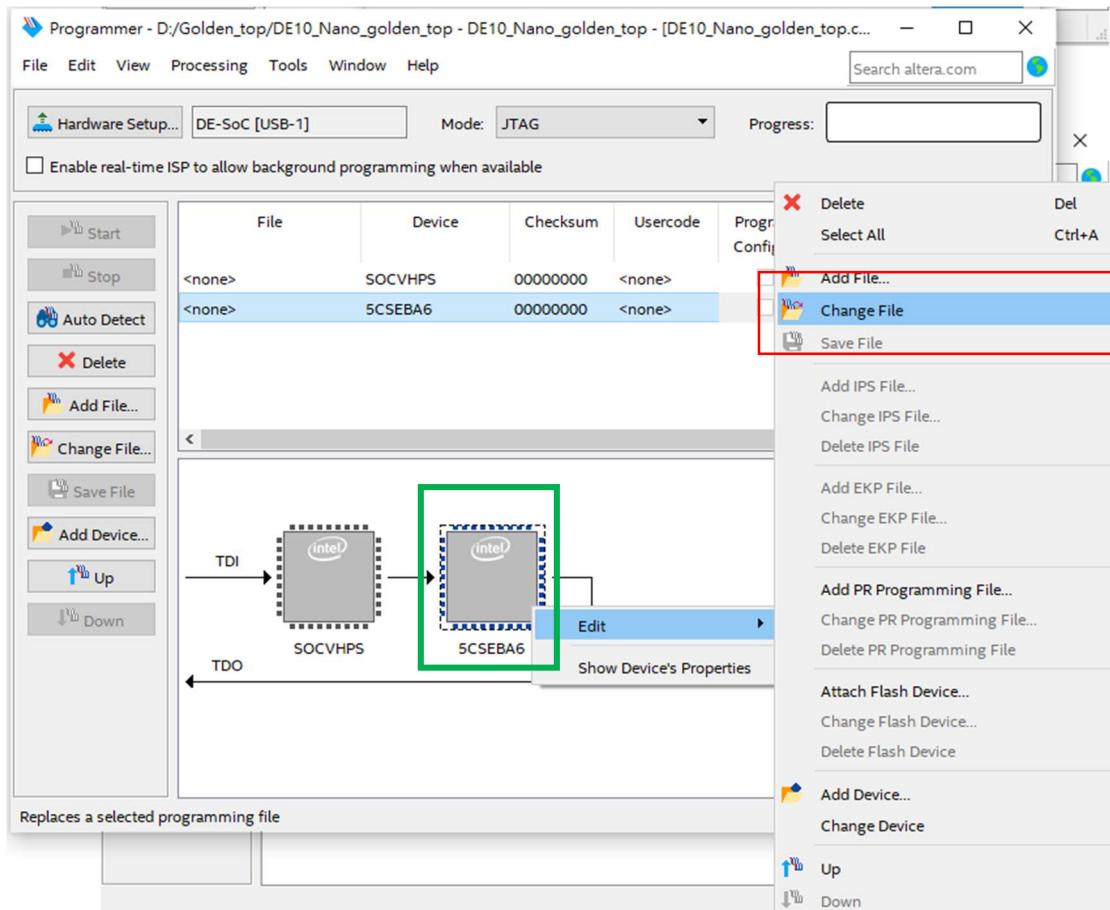
其結果會如下圖所示。選擇 FPGA Device 型號為 5CSEBA6，再點選 OK。



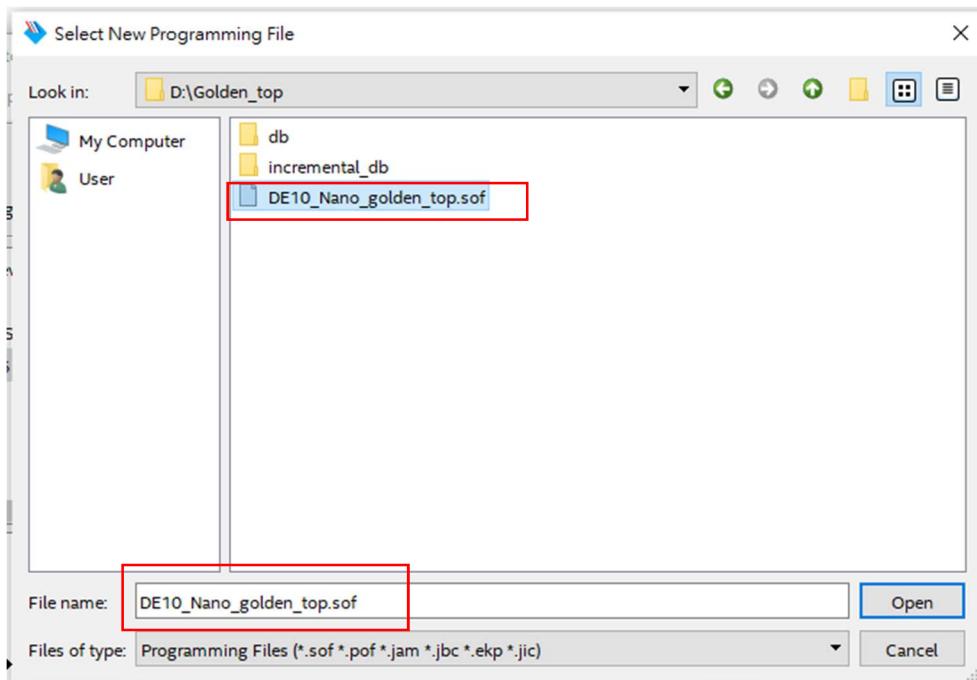
其結果如下圖所示。



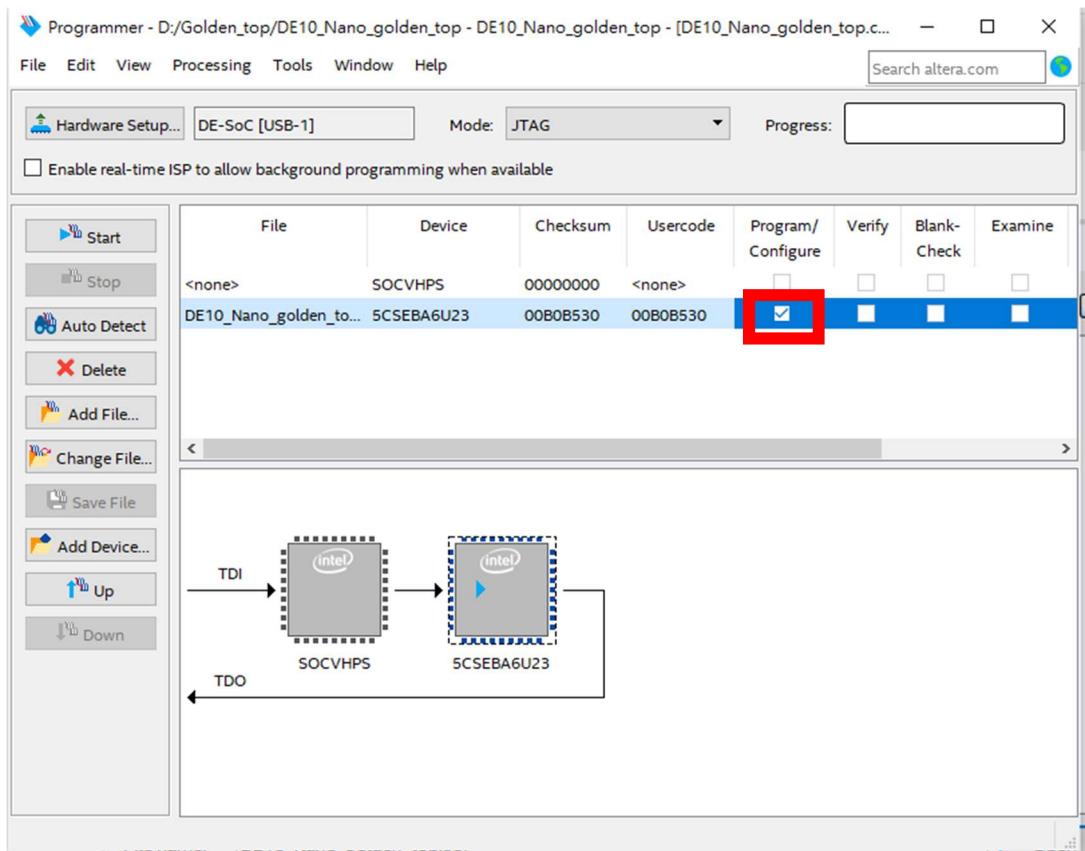
將滑鼠右鍵點選 5CSEBA6 的 icon，如下圖綠框所示，再點選”Edit”的選項，即會出現另一個視窗，再選擇”Chang File”，其位置如下圖紅框所示。



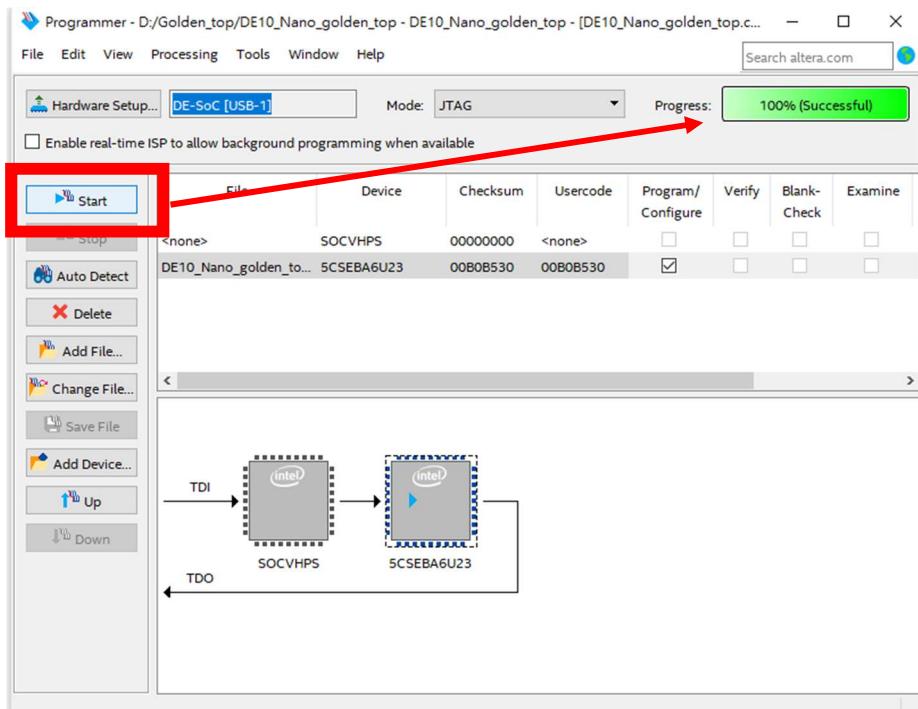
即會出現”Select New Programming File”的視窗，於 File name:的欄位選取 FPGA 的燒錄檔“DE10_Nano_golden_top.sof”，如下圖紅框所示，再點選”Open”



其結果會如下圖所示。然後將下圖紅框所示的方格打勾，如下圖紅框所示：



點選 Start(如下圖紅框所示)的 icon，即會開始將 SOF 檔燒至 FPGA 中，待 Progress 呈現 100% 時，即代表燒錄完成。



即可以觀察 FPGA 上的 8 顆 LED 全亮。