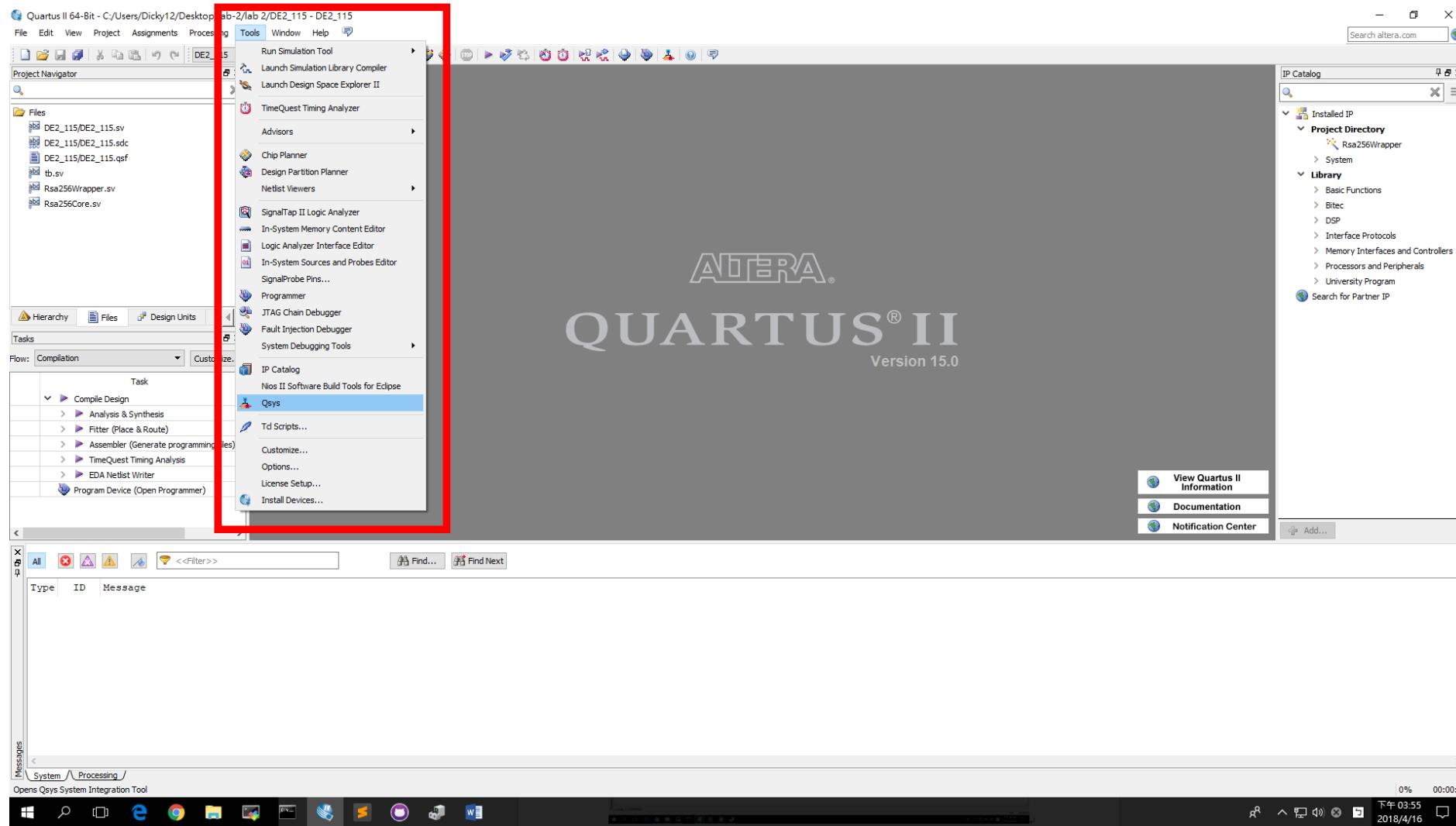


# **Qsys on Quartus 15.0 Tutorial**

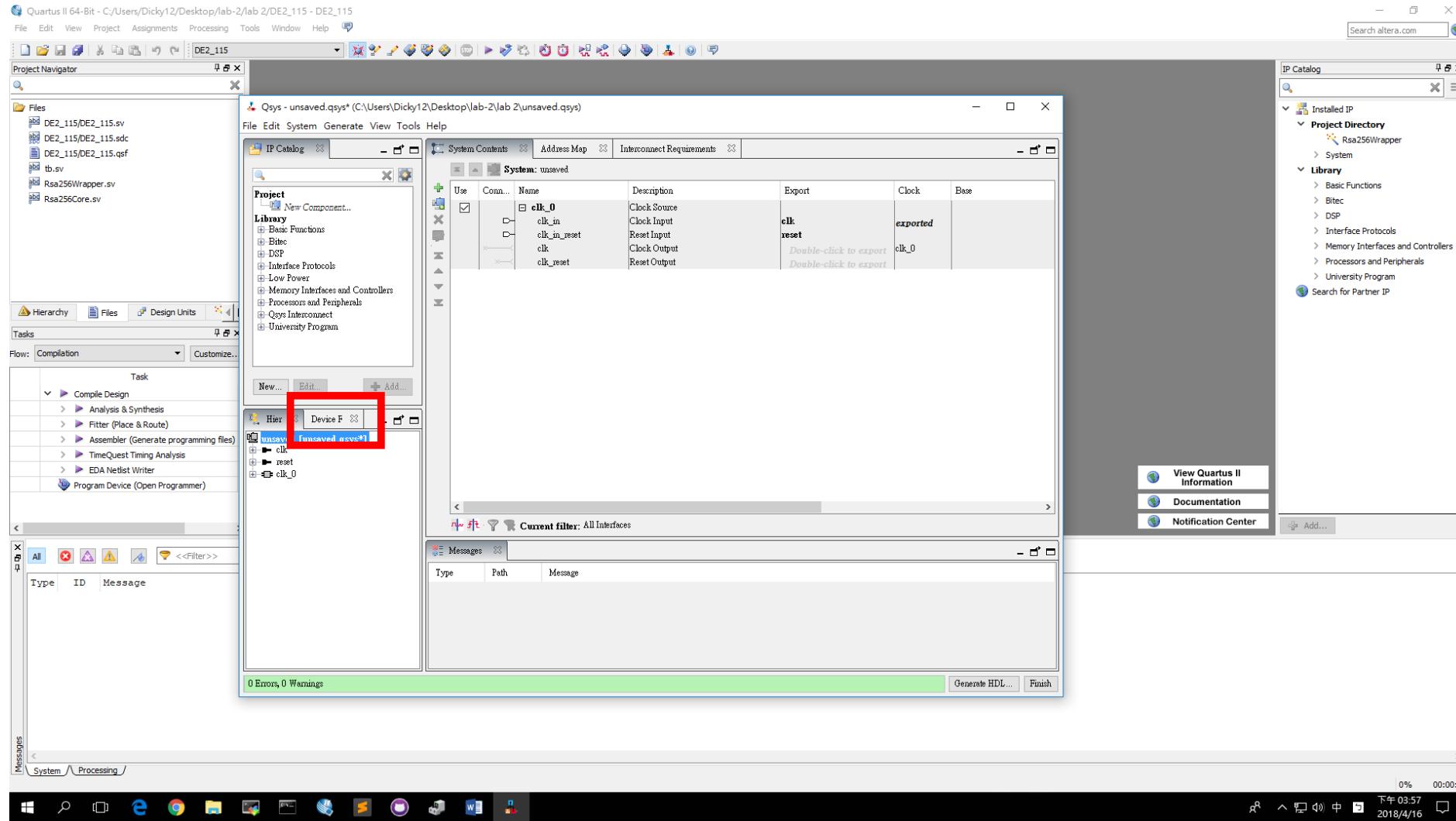
提供：董子維、楊其昇

整理：鍾杰

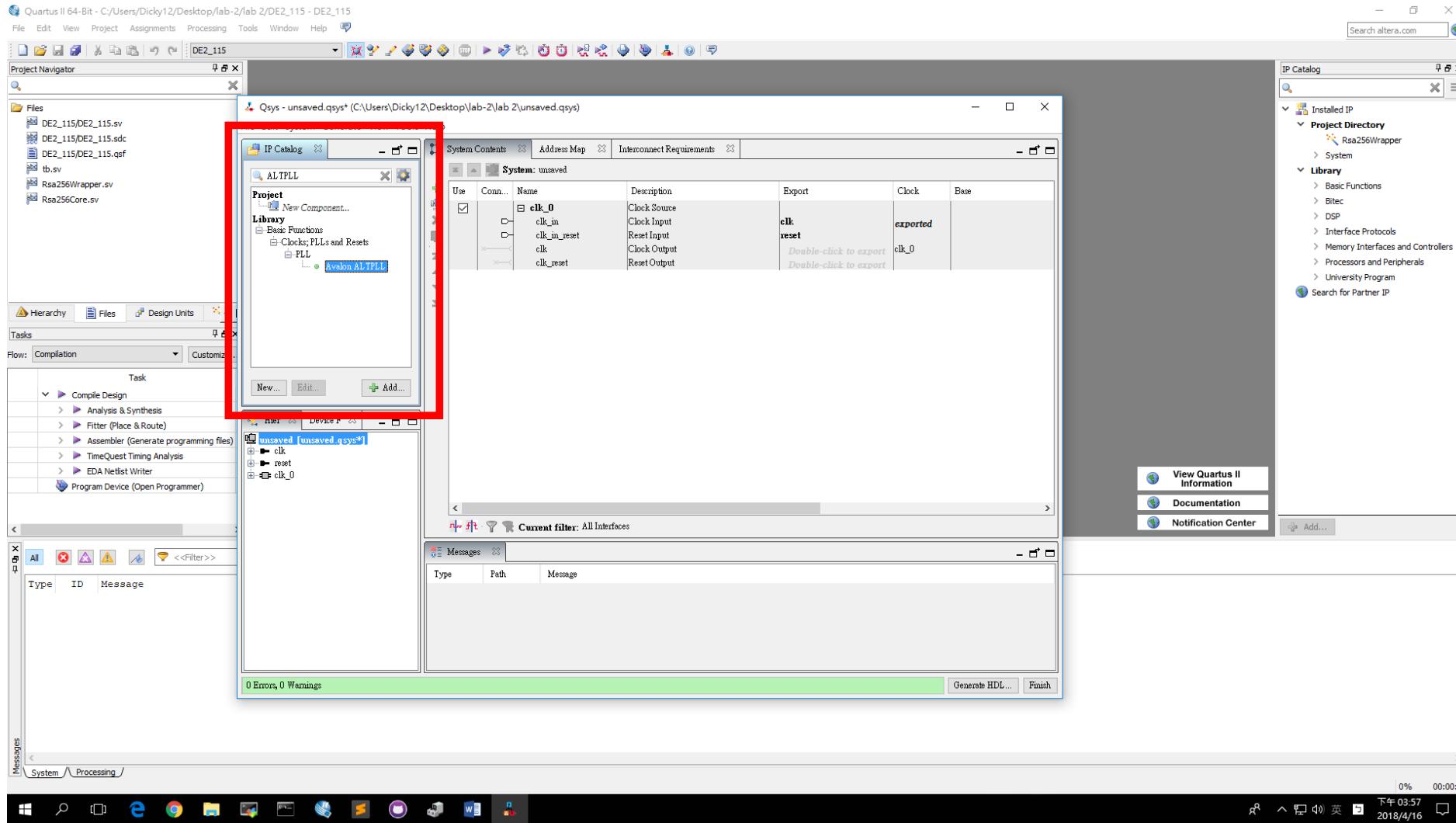
# 打開Qsys (Tool → Qsys)



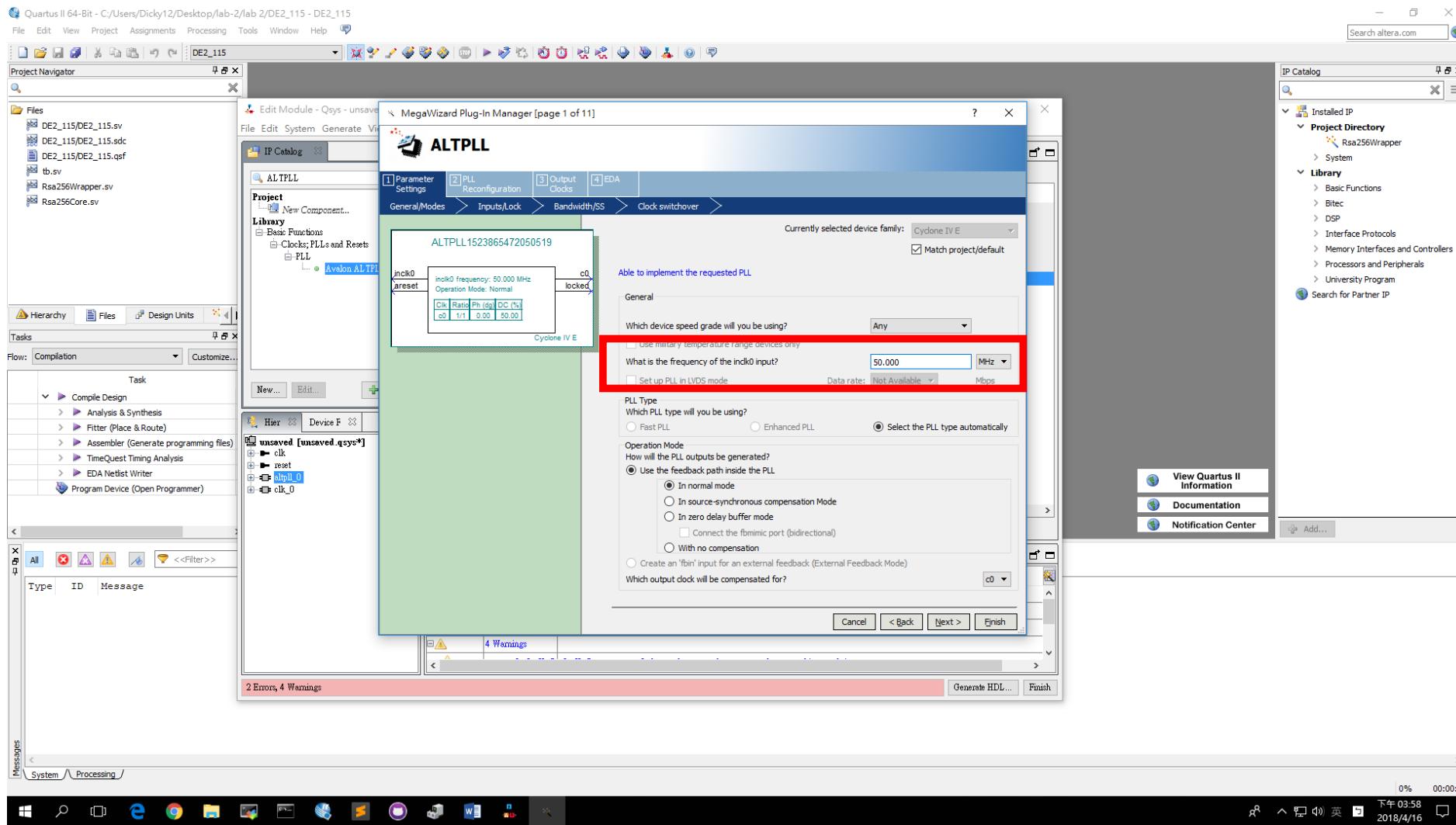
# 到Device Family選擇Cyclone IV E



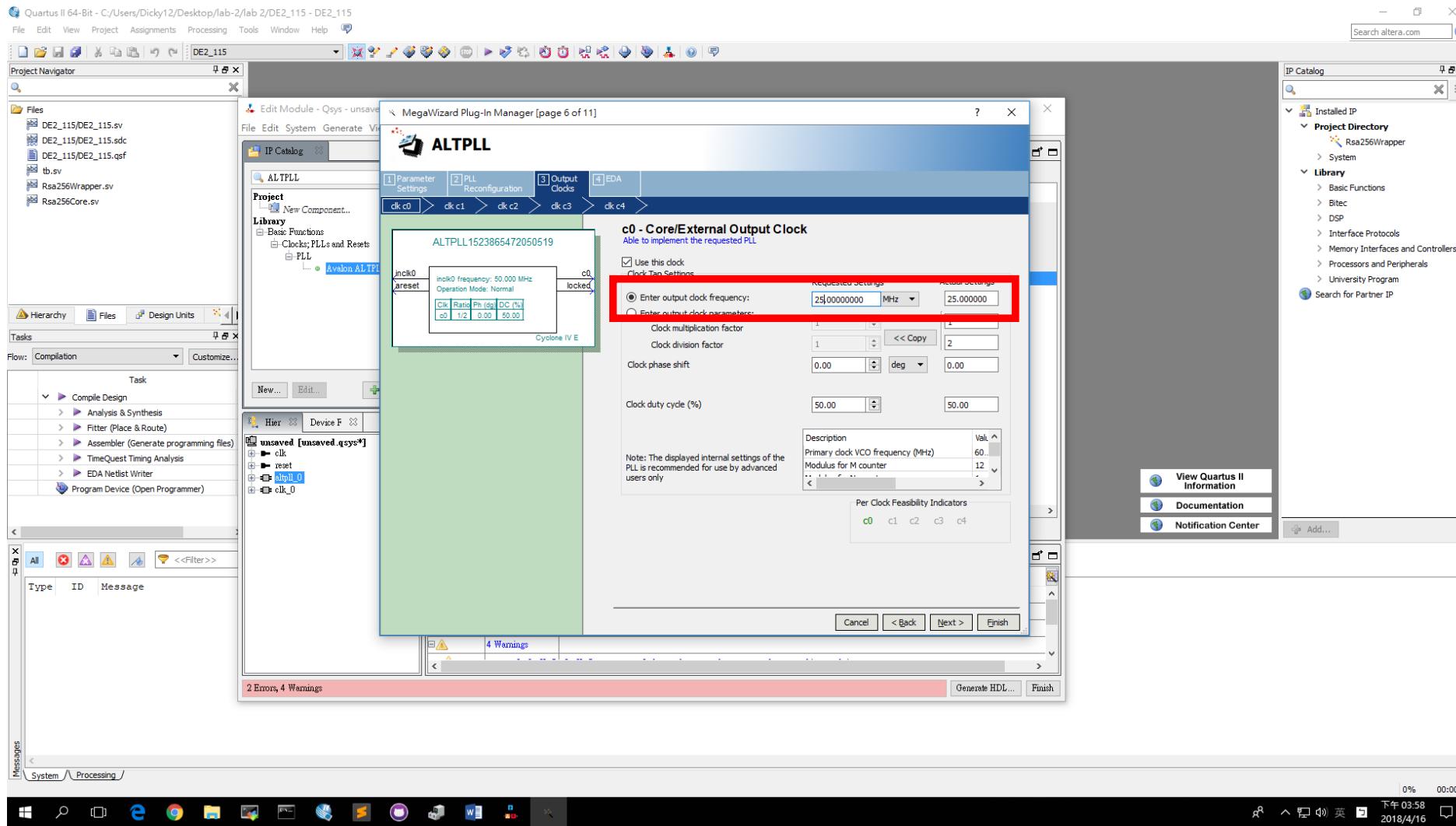
# 搜尋Avalon ALTPLL, 雙擊新增



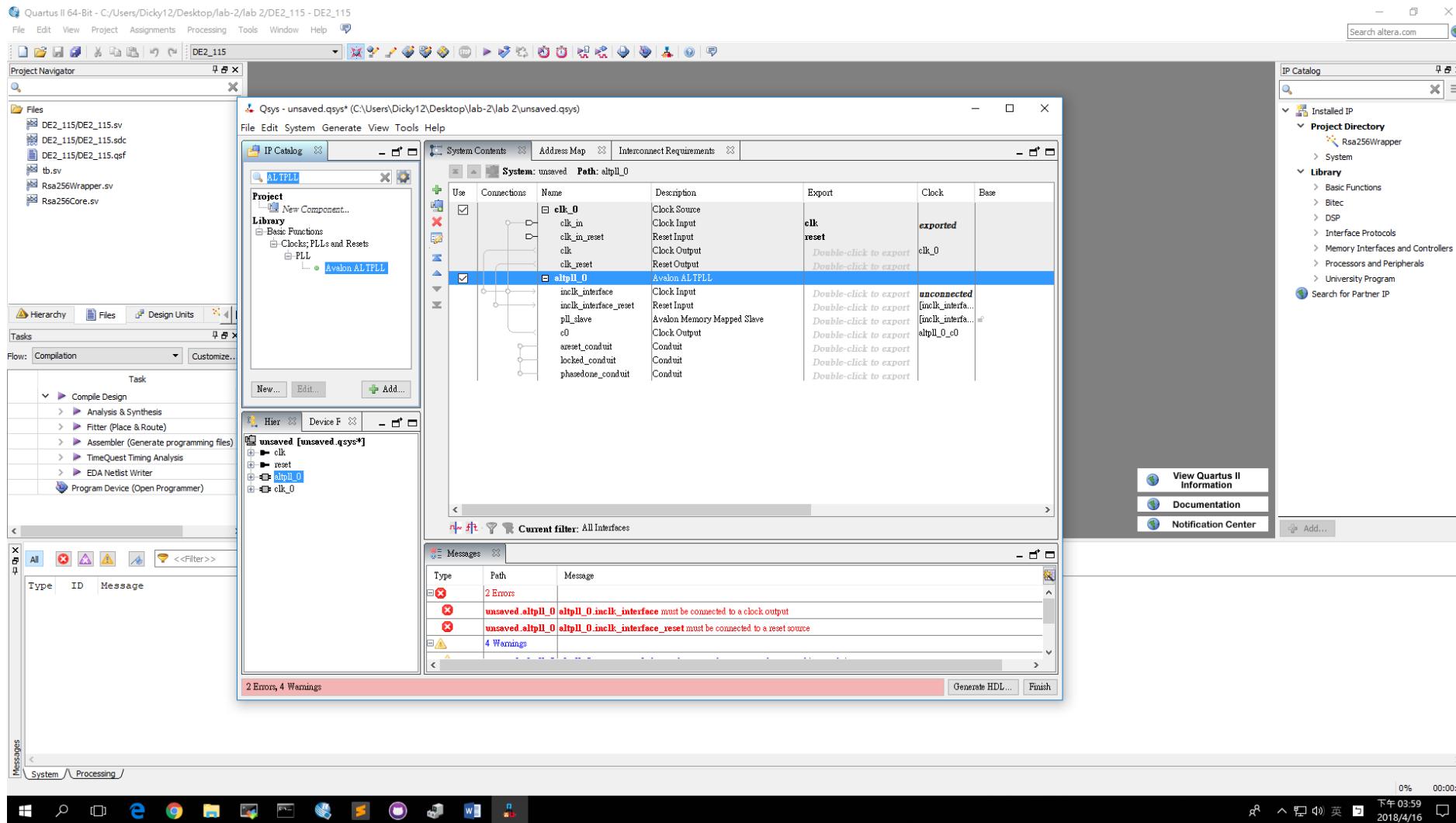
# Frequency of inclk0 input = 50MHz



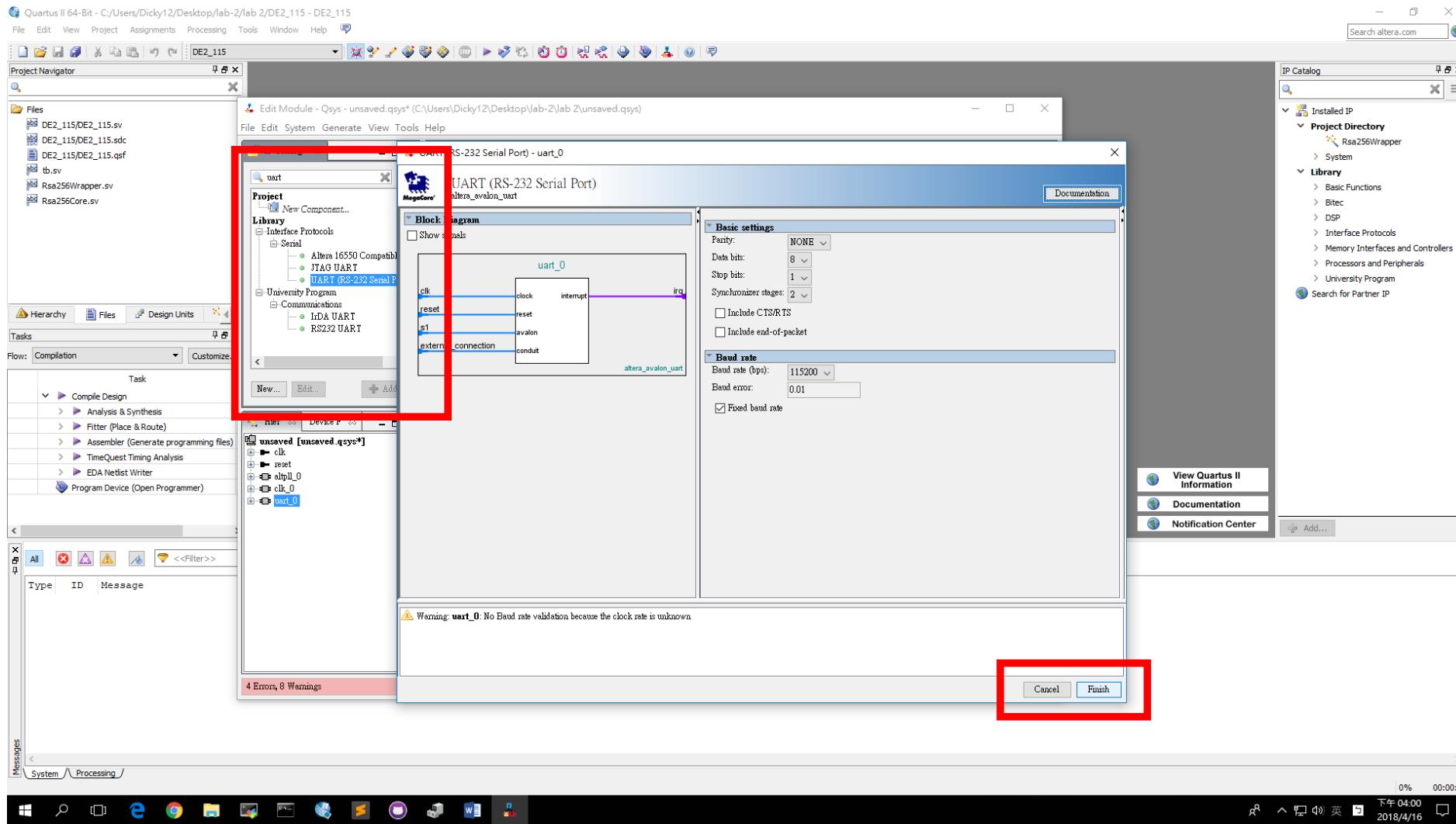
# Output clock frequency = 25MHz



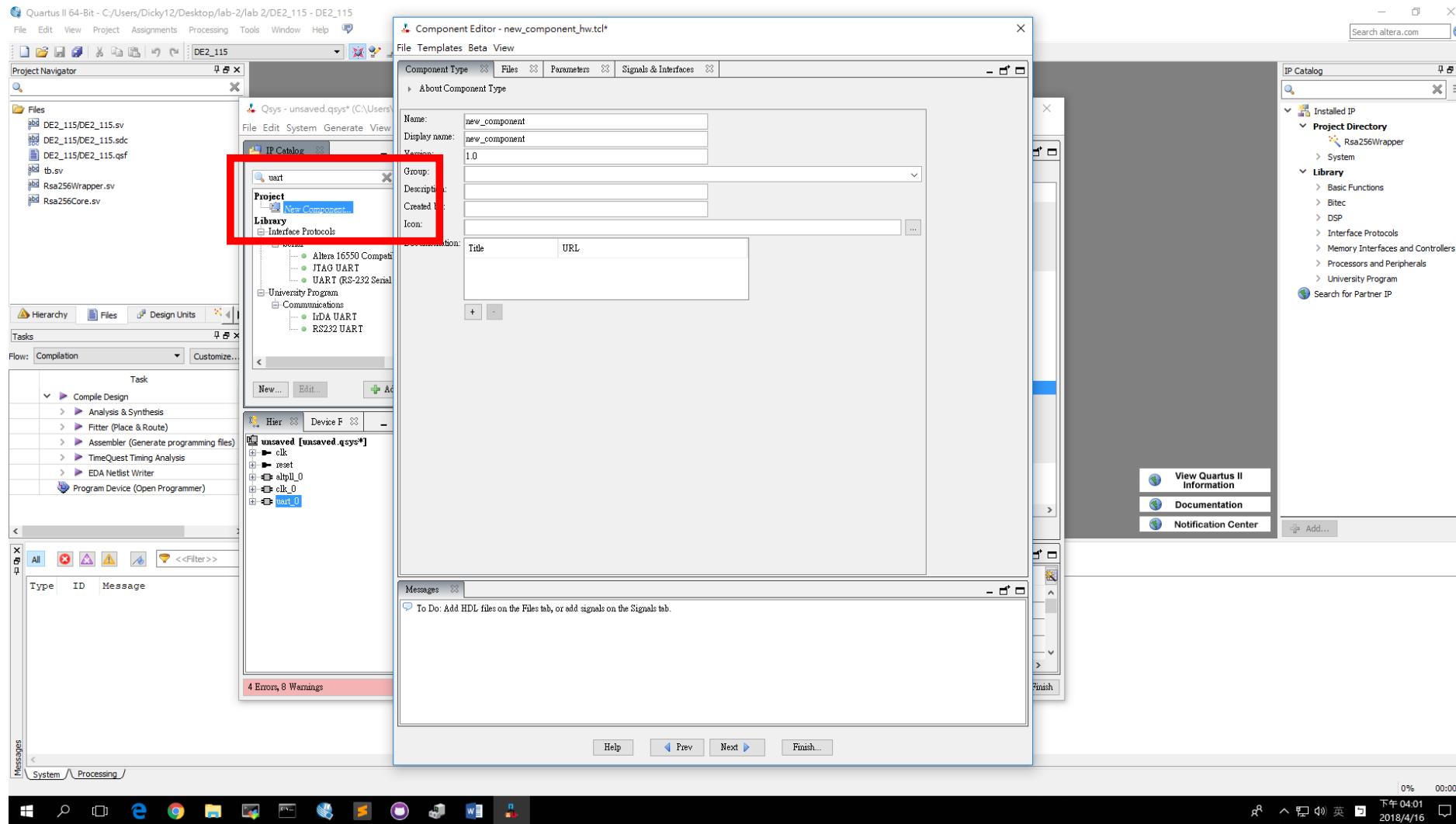
# 按Finish可在右側看到新增的ALTPLL



# 搜尋UART → UART (RS-232 Serial Port) → Finish

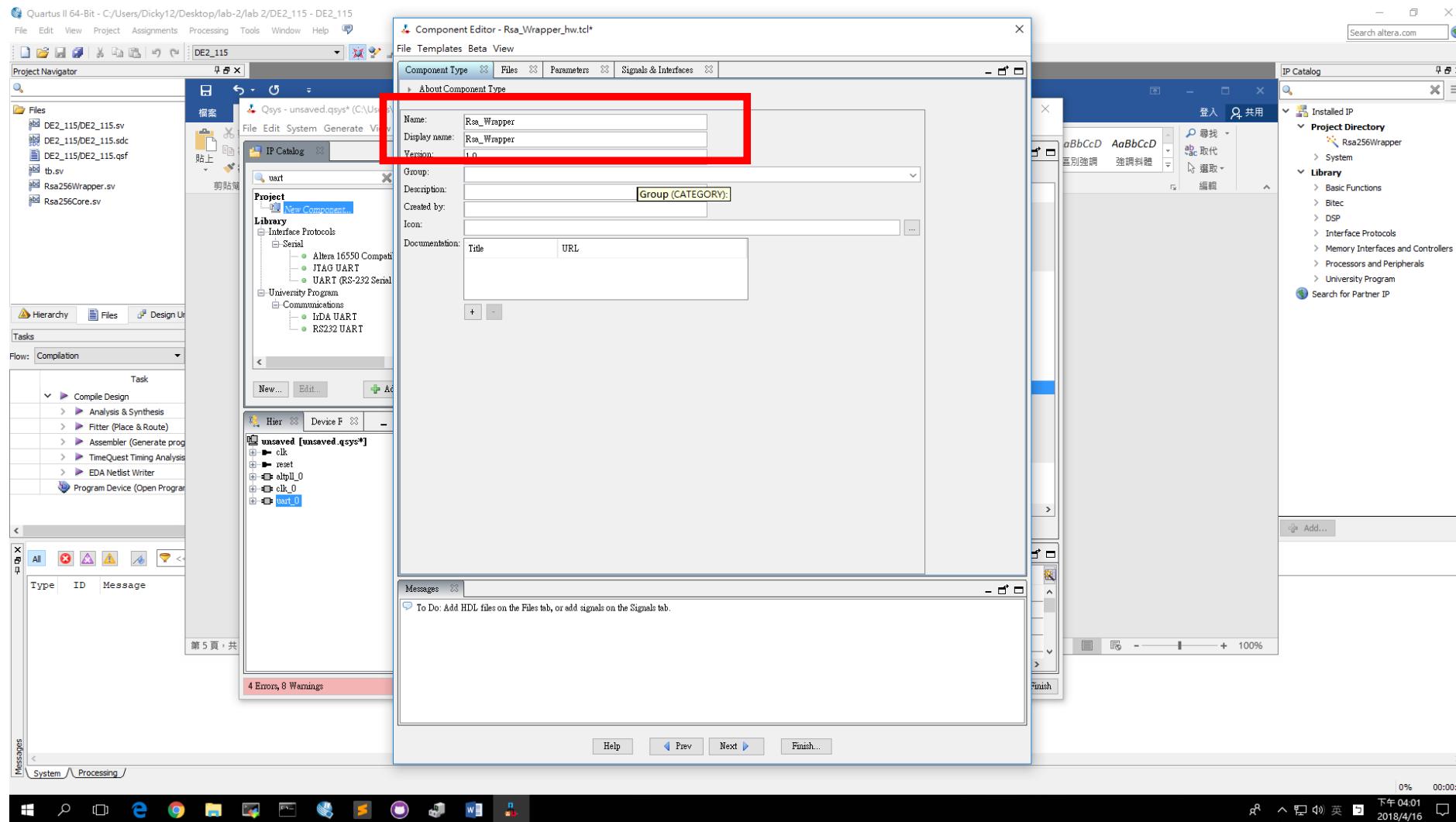


# 雙擊 New Component

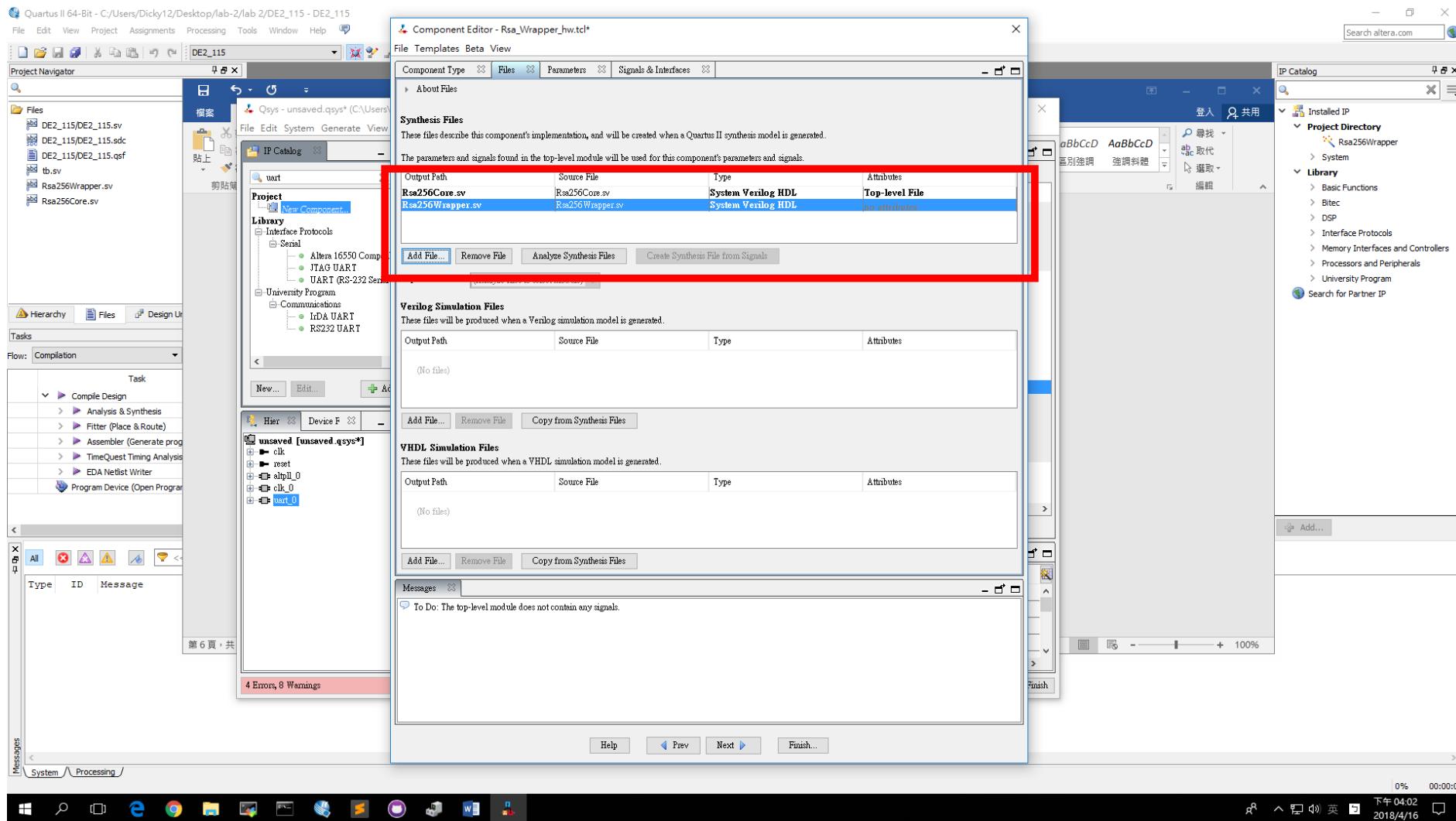


# Name = Rsa\_Wrapper

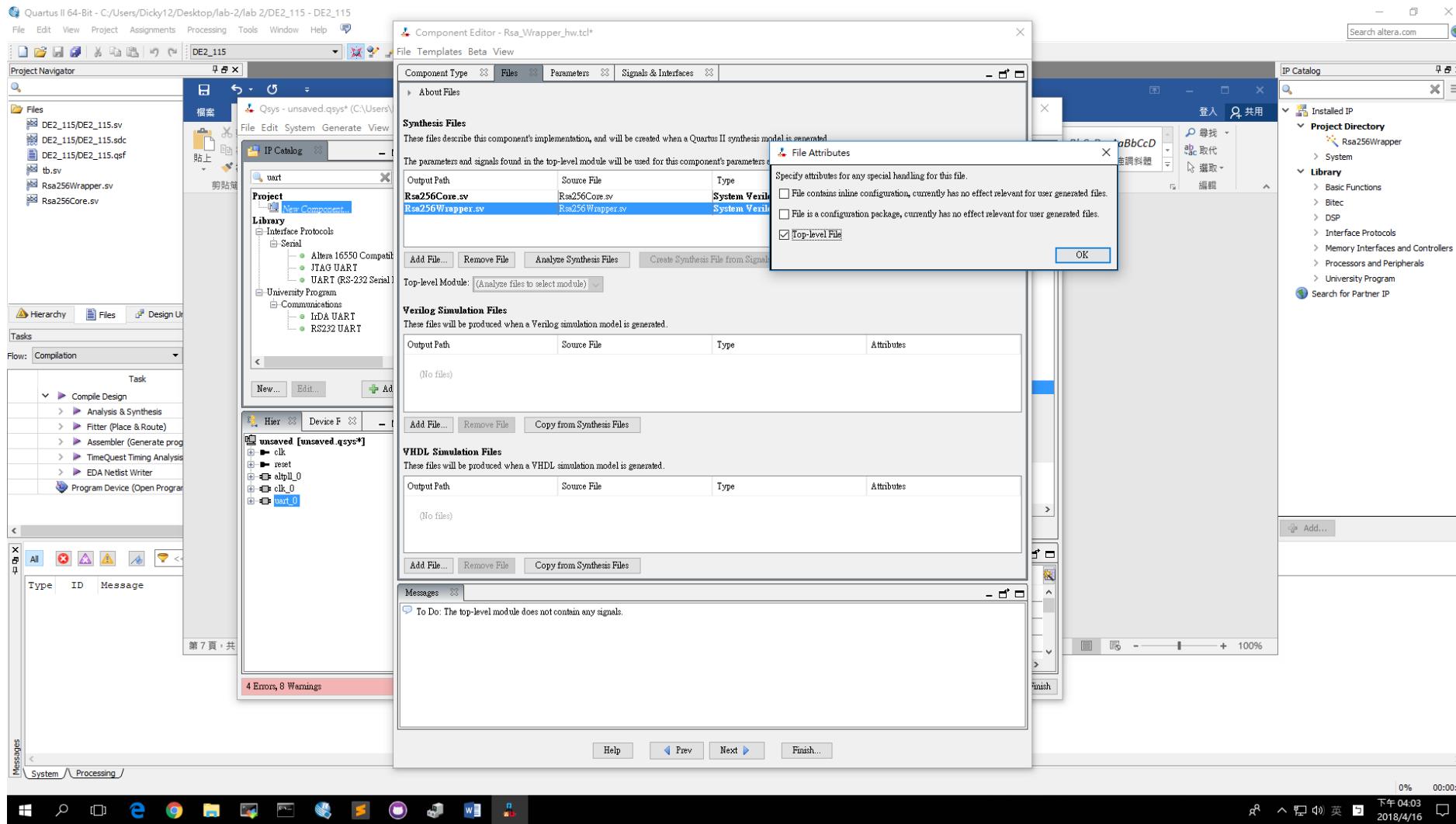
# Display Name = Rsa\_Wrapper



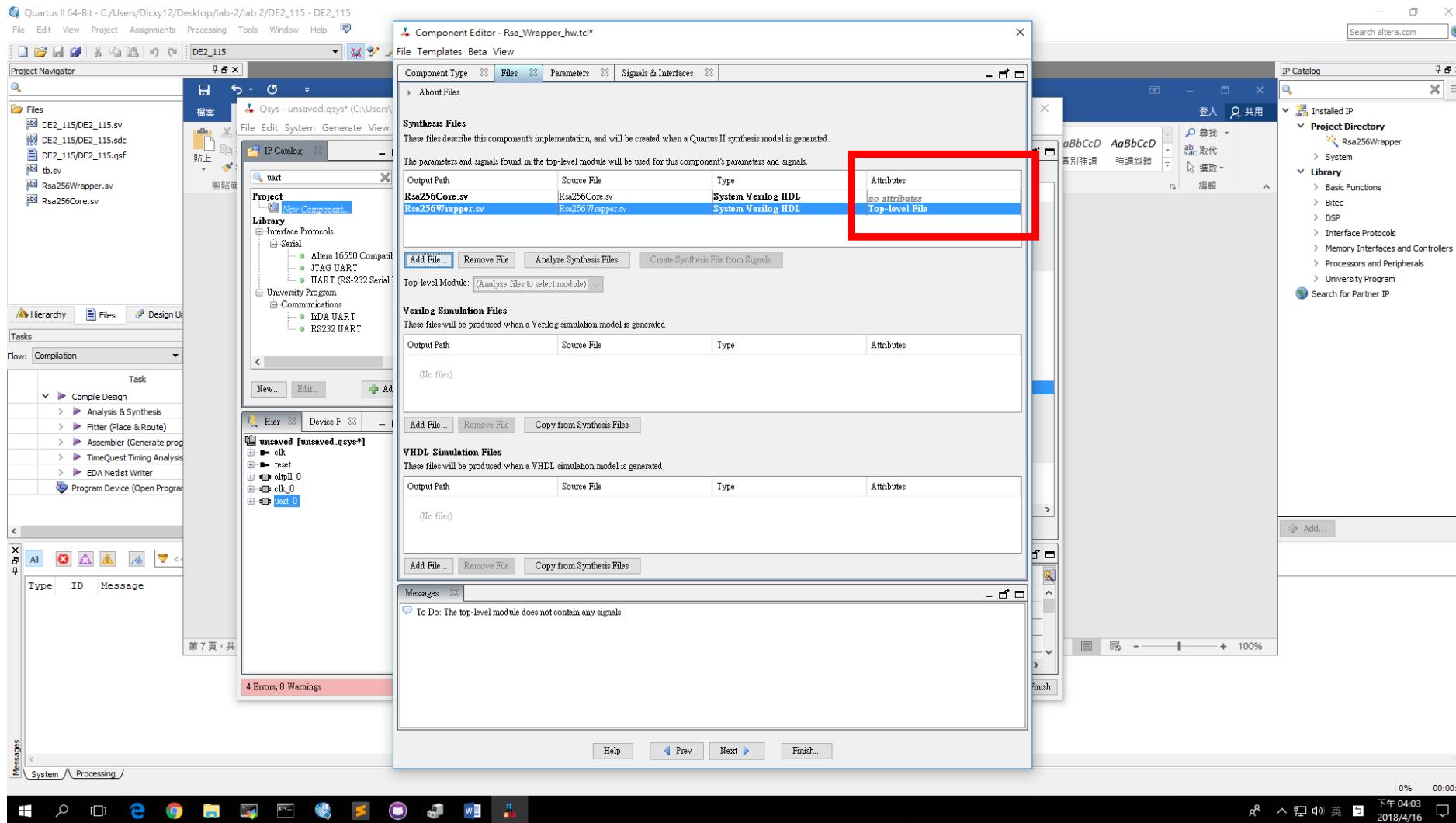
# File → Add File 新增Rsa256Core.sv和Rsa256Wrapper.sv



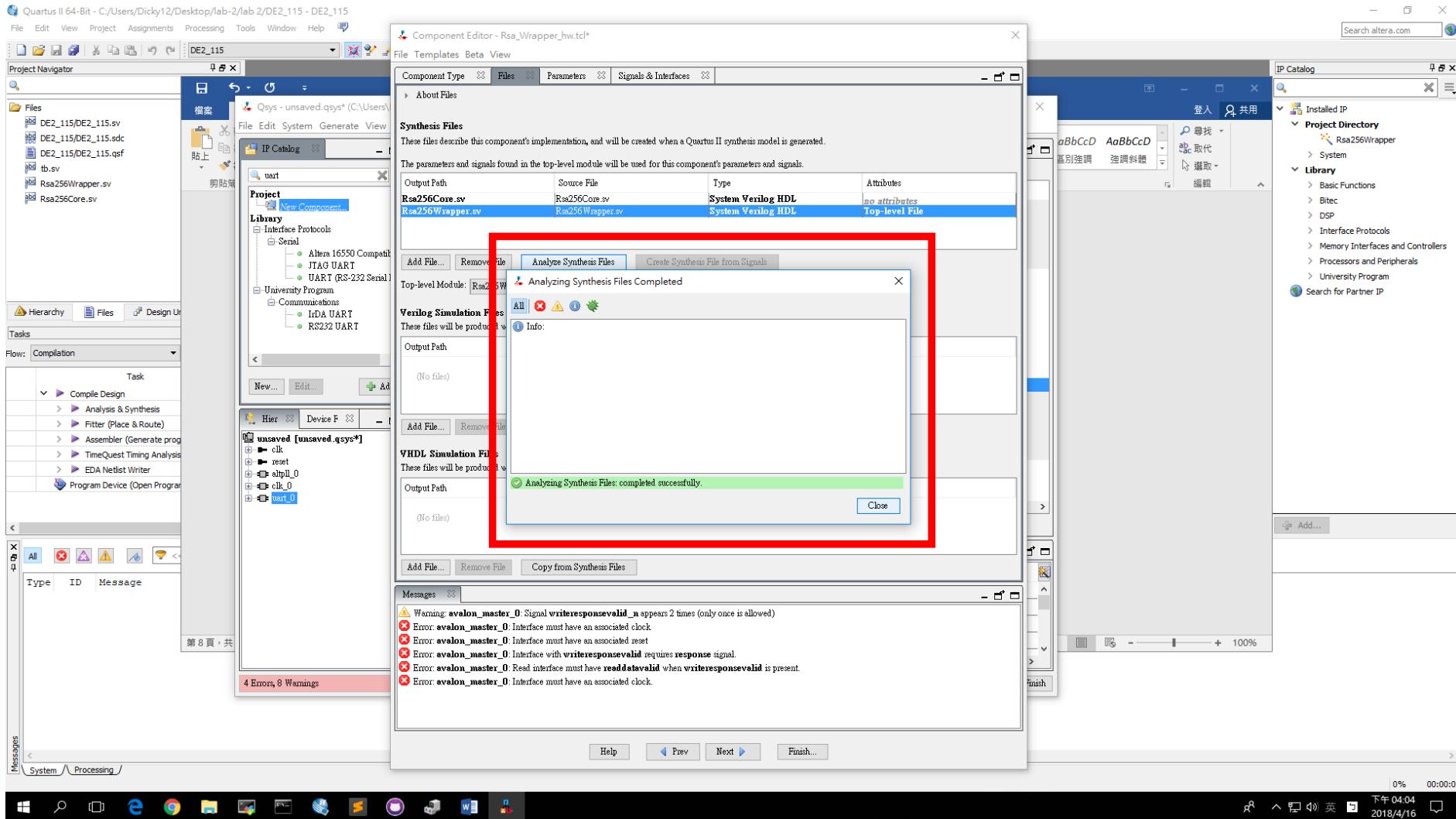
# 將Rsa256Wrapper改成Top level file



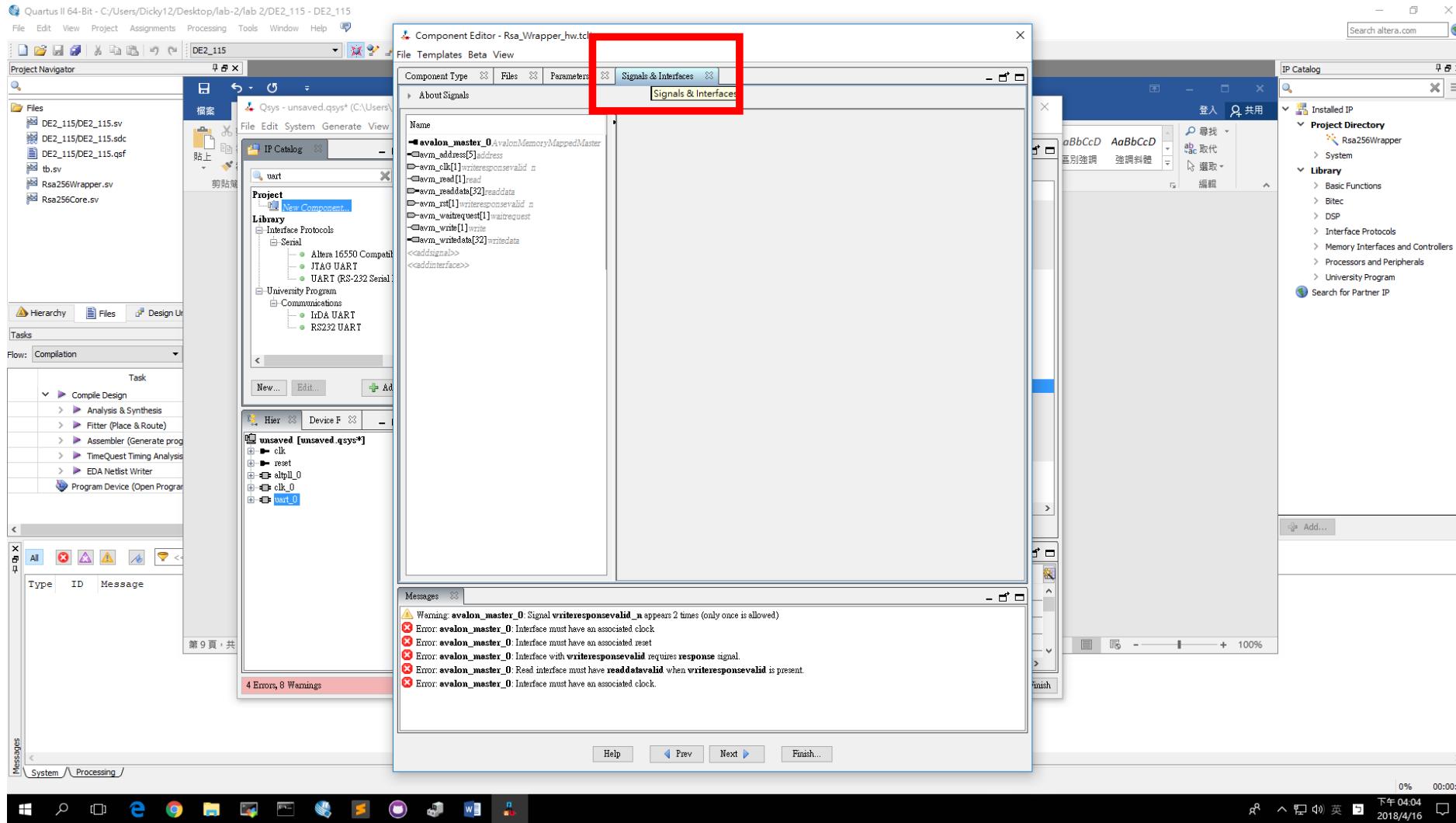
# 將Rsa256Wrapper改成Top level file



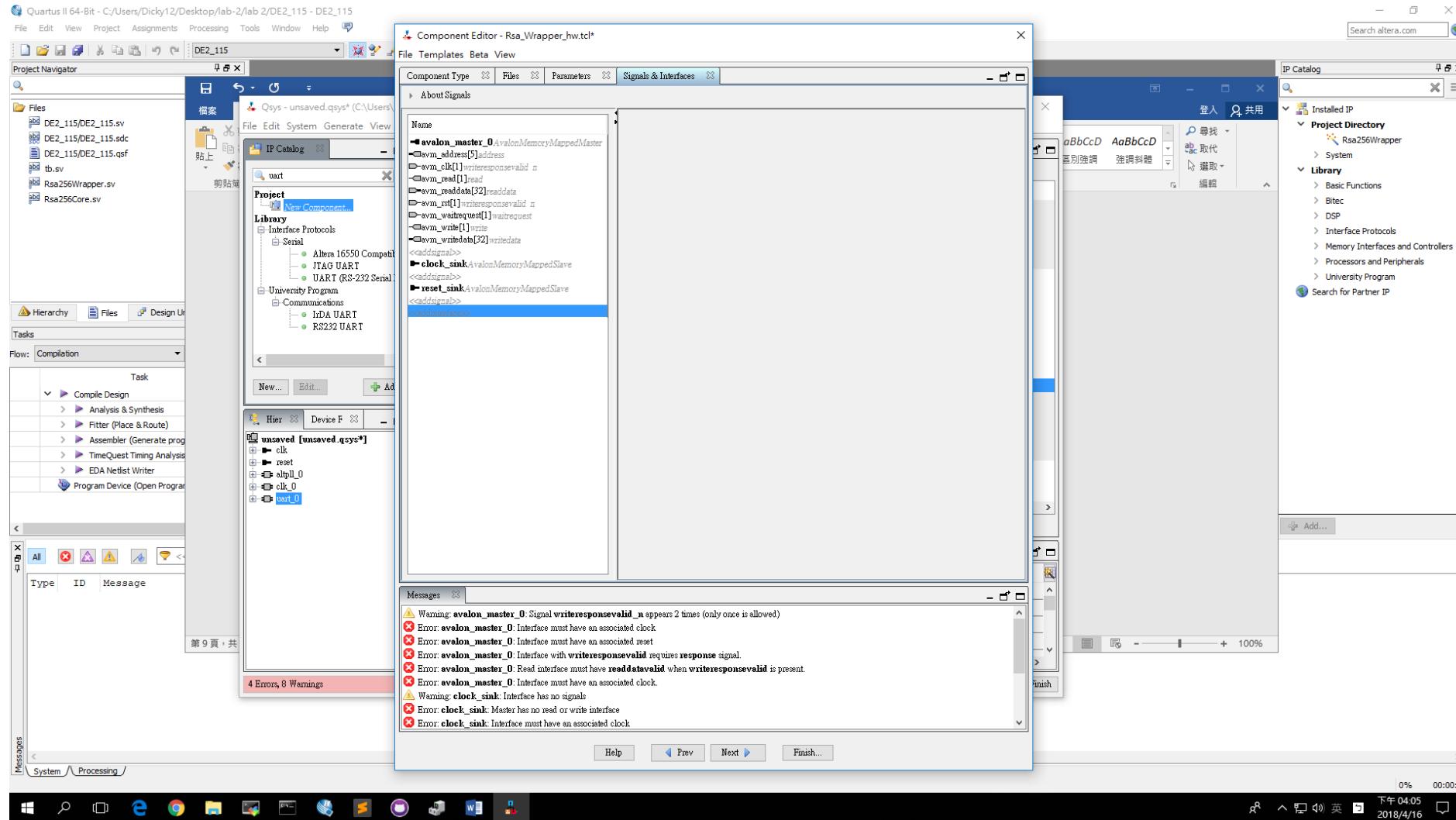
# Analyze Synthesis File → Close



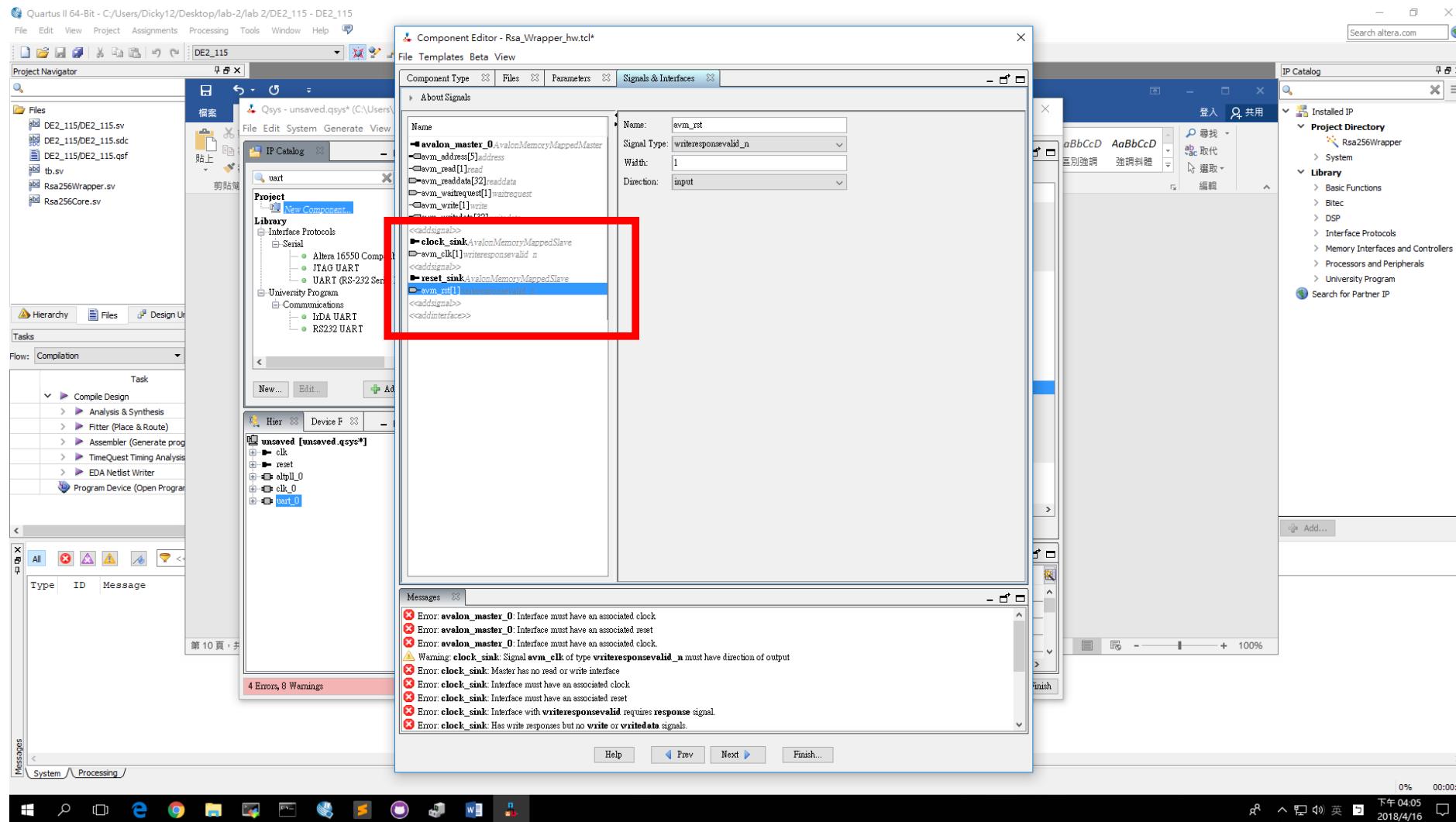
# 選 Signal Interface



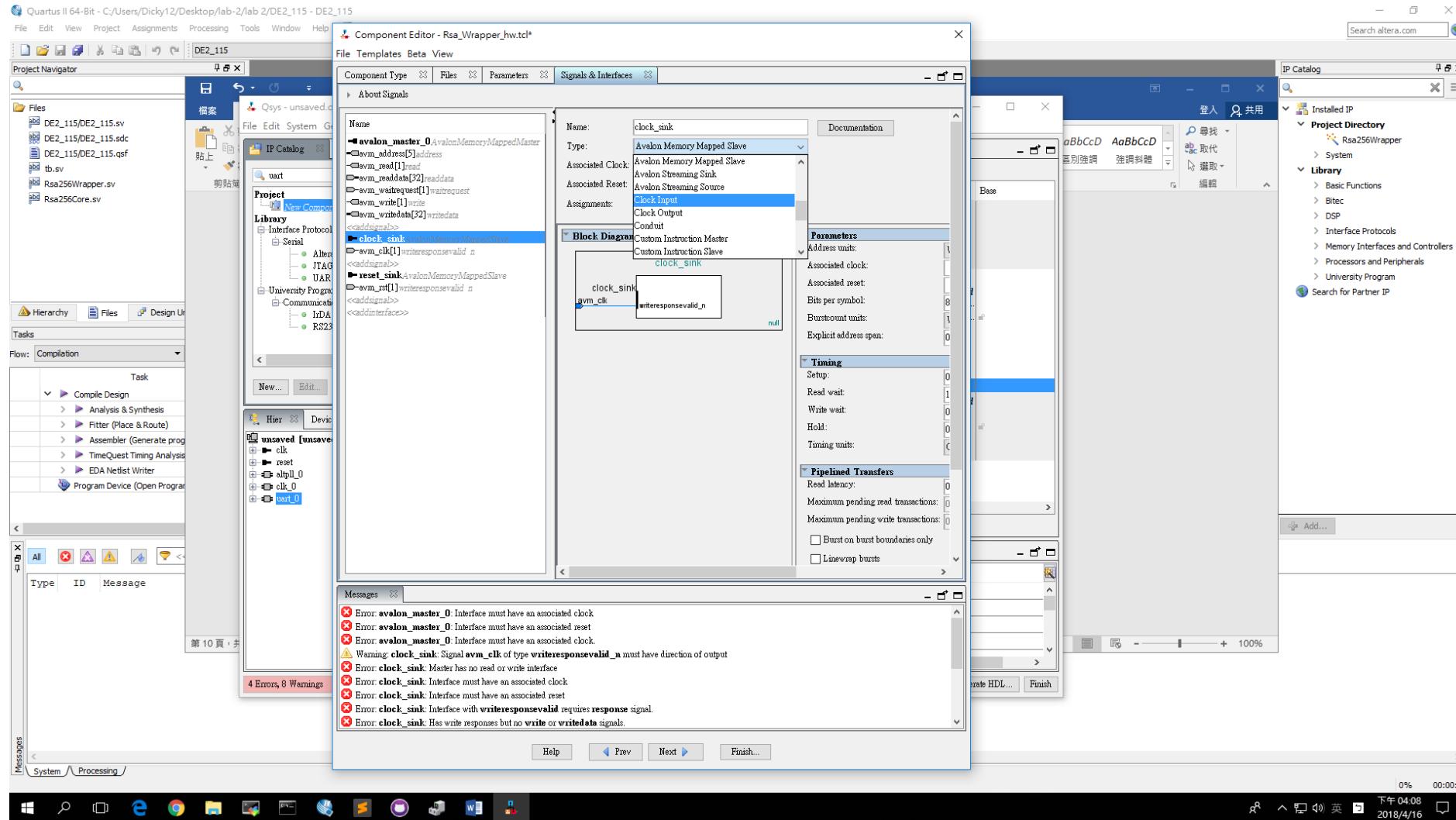
雙擊 Add interface → clock\_sink  
雙擊 Add interface → reset\_sink



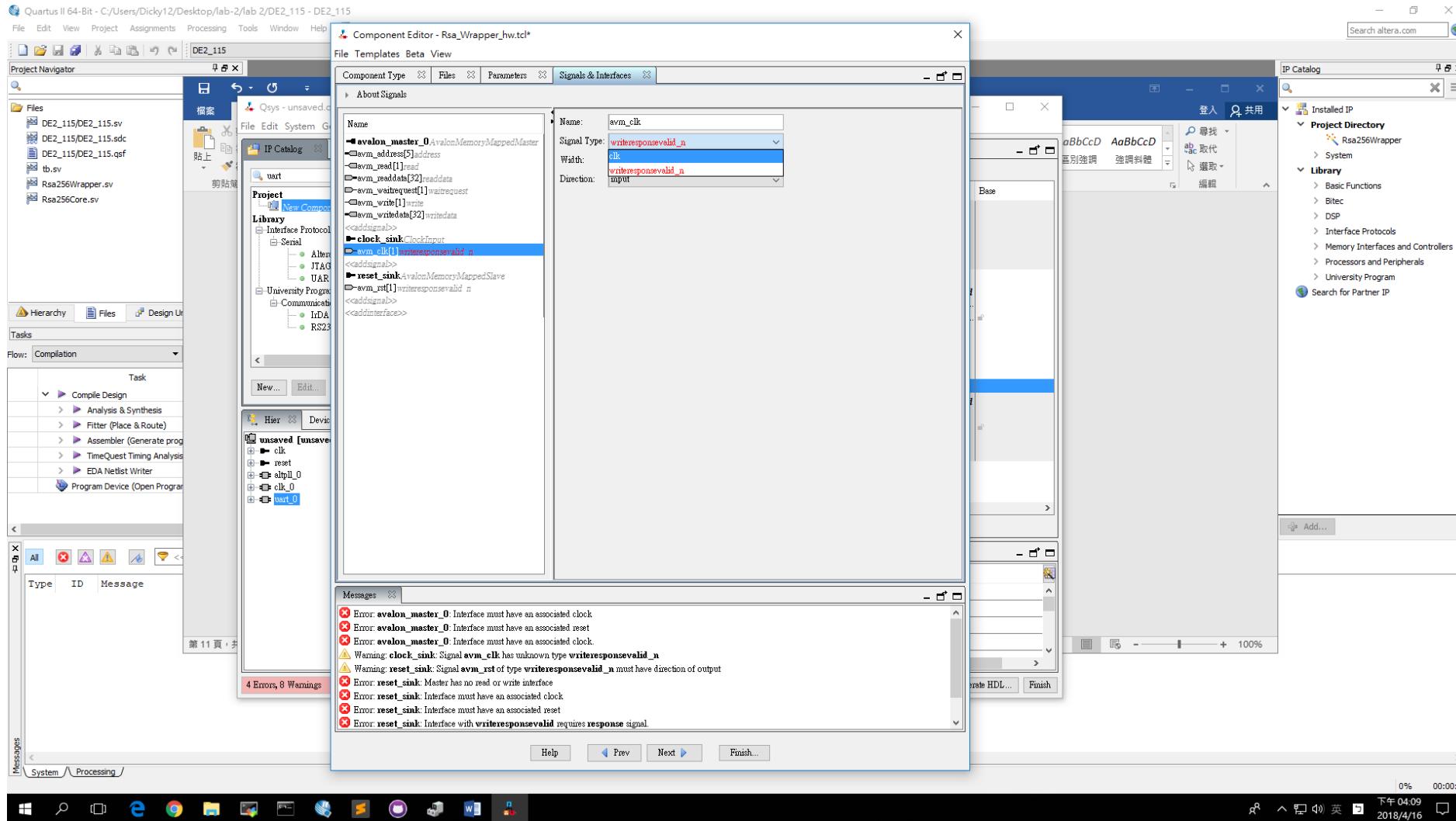
avm\_clk拖曳至clock\_sink下方  
avm\_rst拖曳至reset\_sink下方



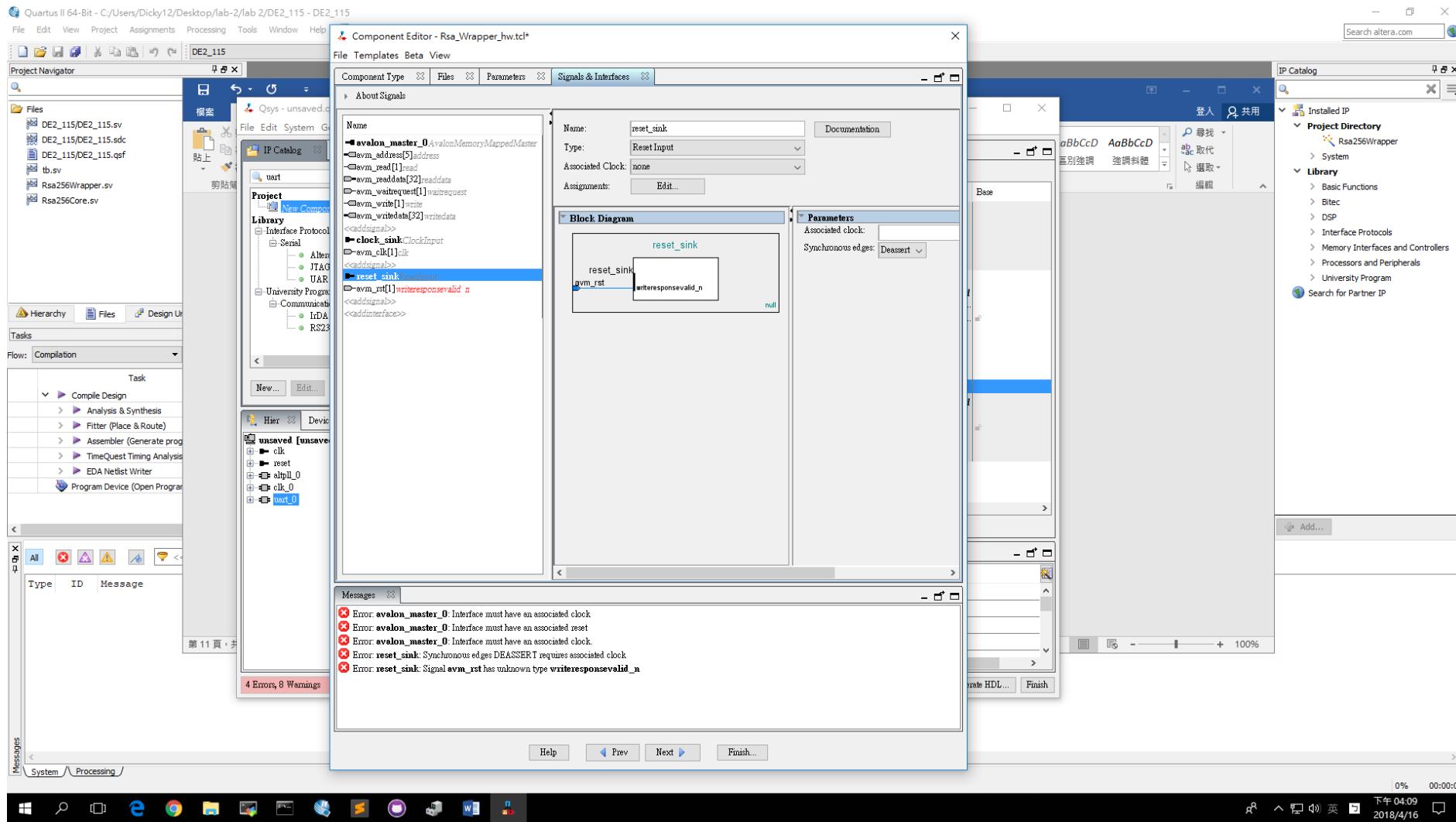
# 將clock\_sink的Type改成Clock Input



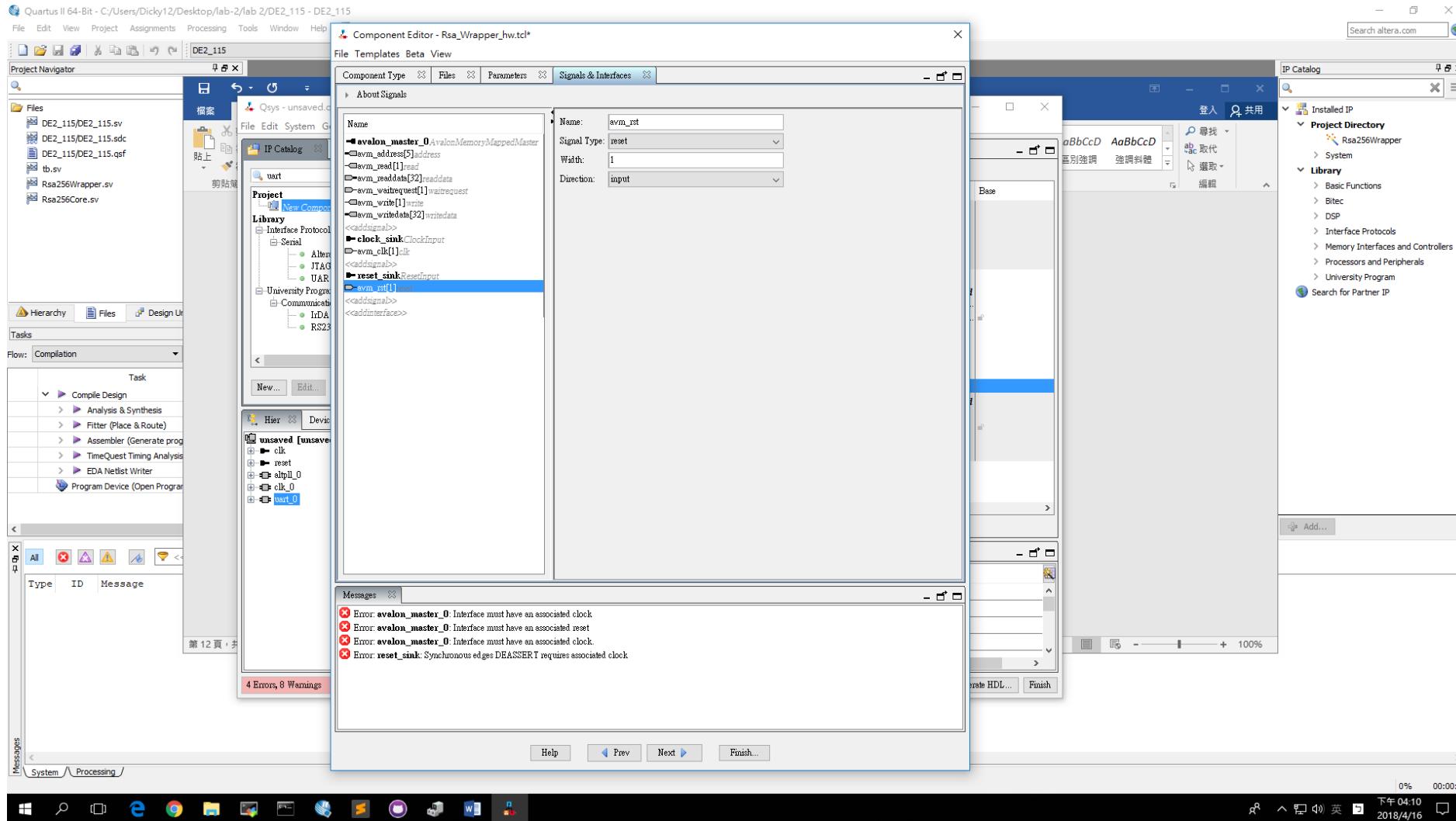
# 將avm\_clk的Signal Type改成clk



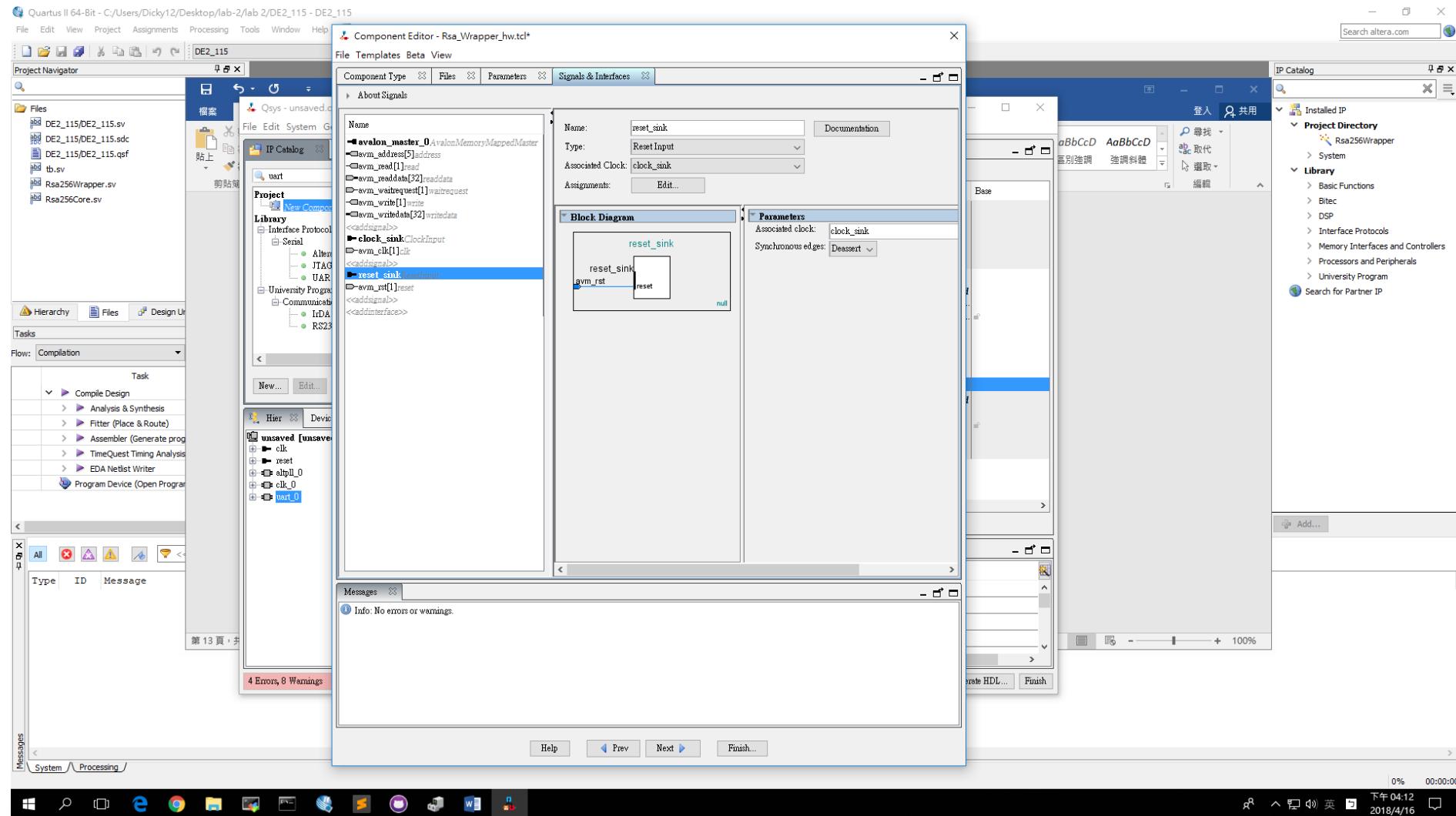
# 將reset\_sink的Type改成Reset Input



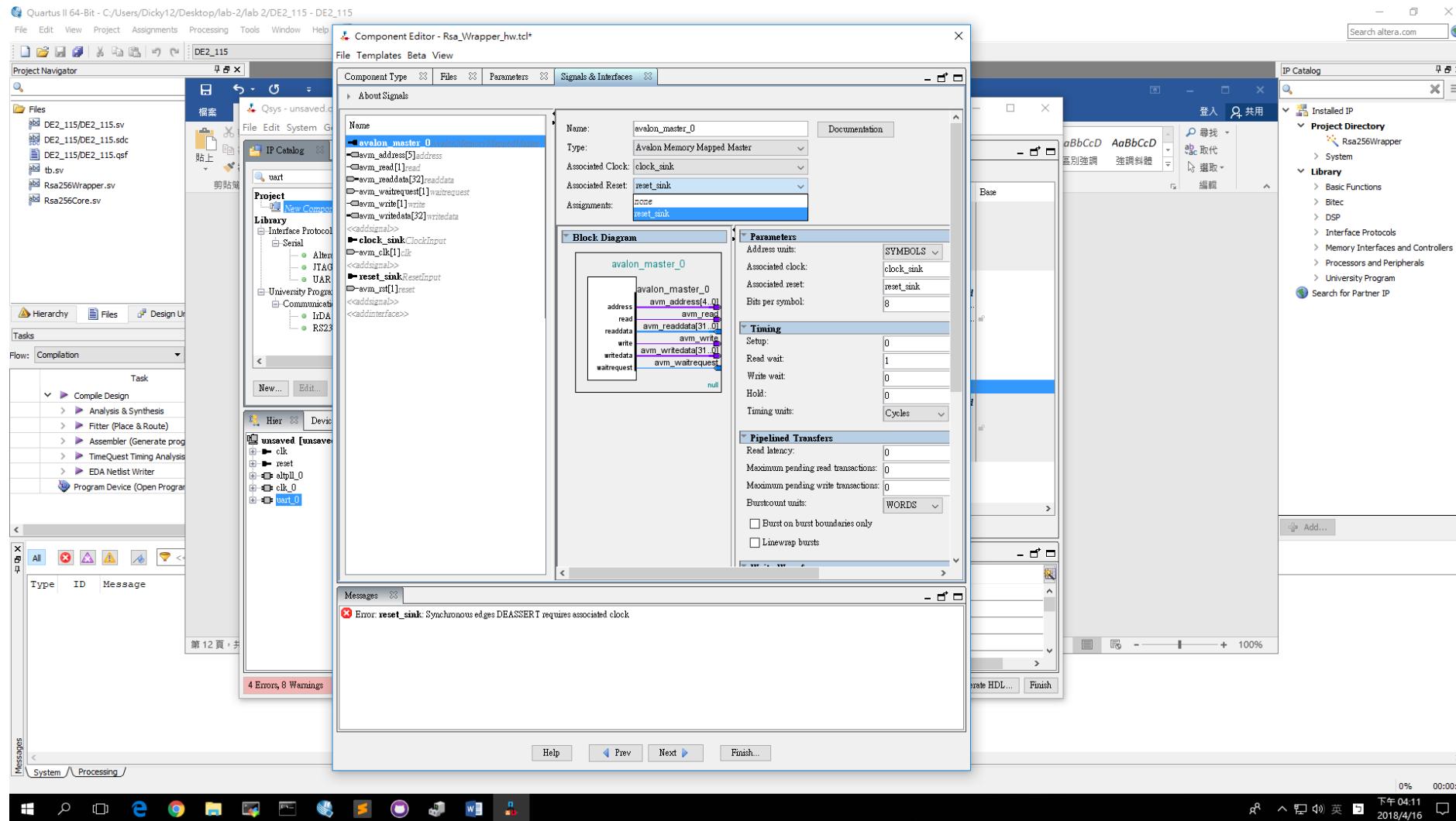
# 將avm\_rst的Signal Type改成reset



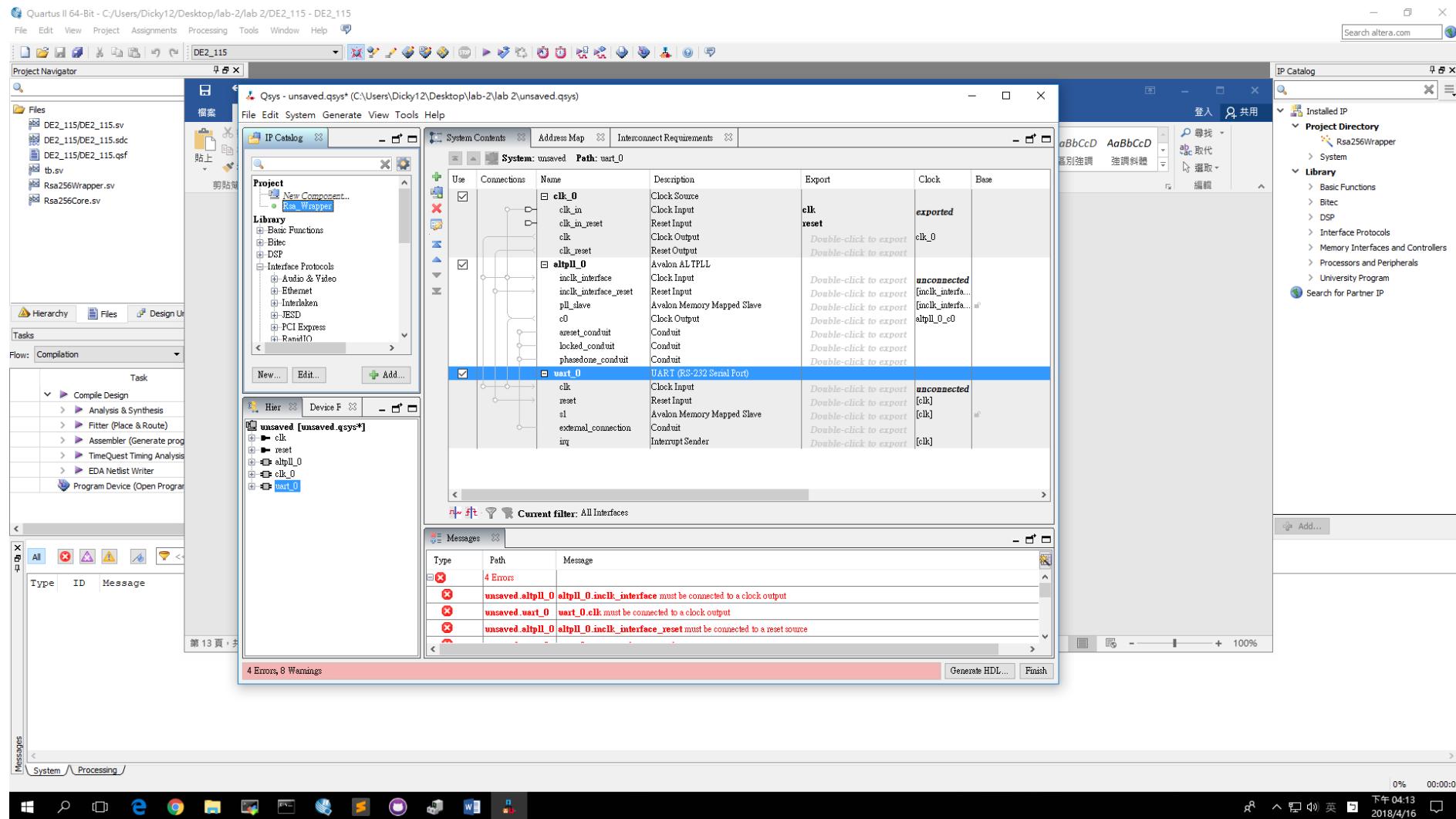
# 將reset\_sink的associate clock改成clock\_sink



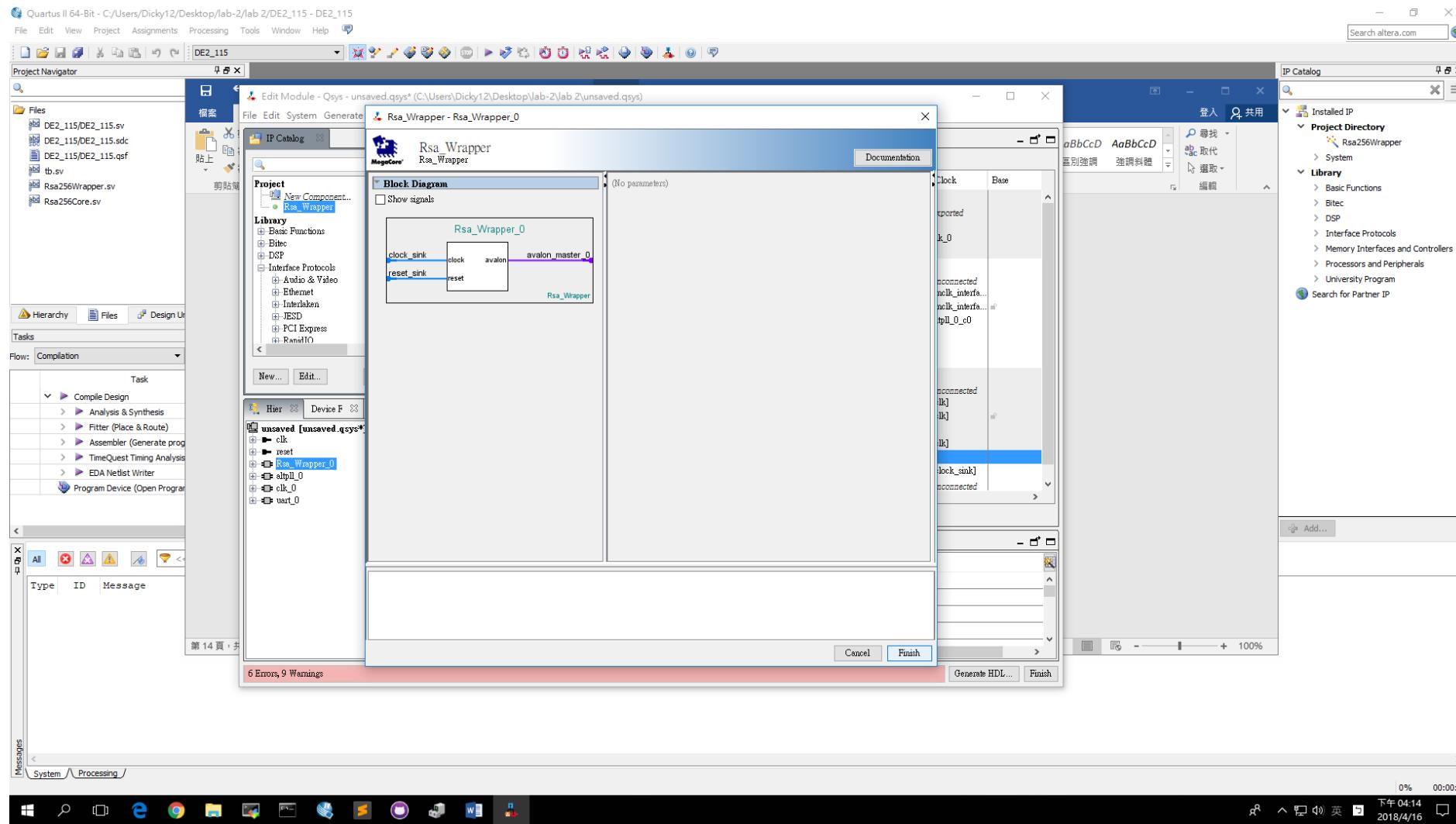
# 將avalon\_master\_0的associate clock改成clock\_sink 將avalon\_master\_0的associate reset改成reset\_sink



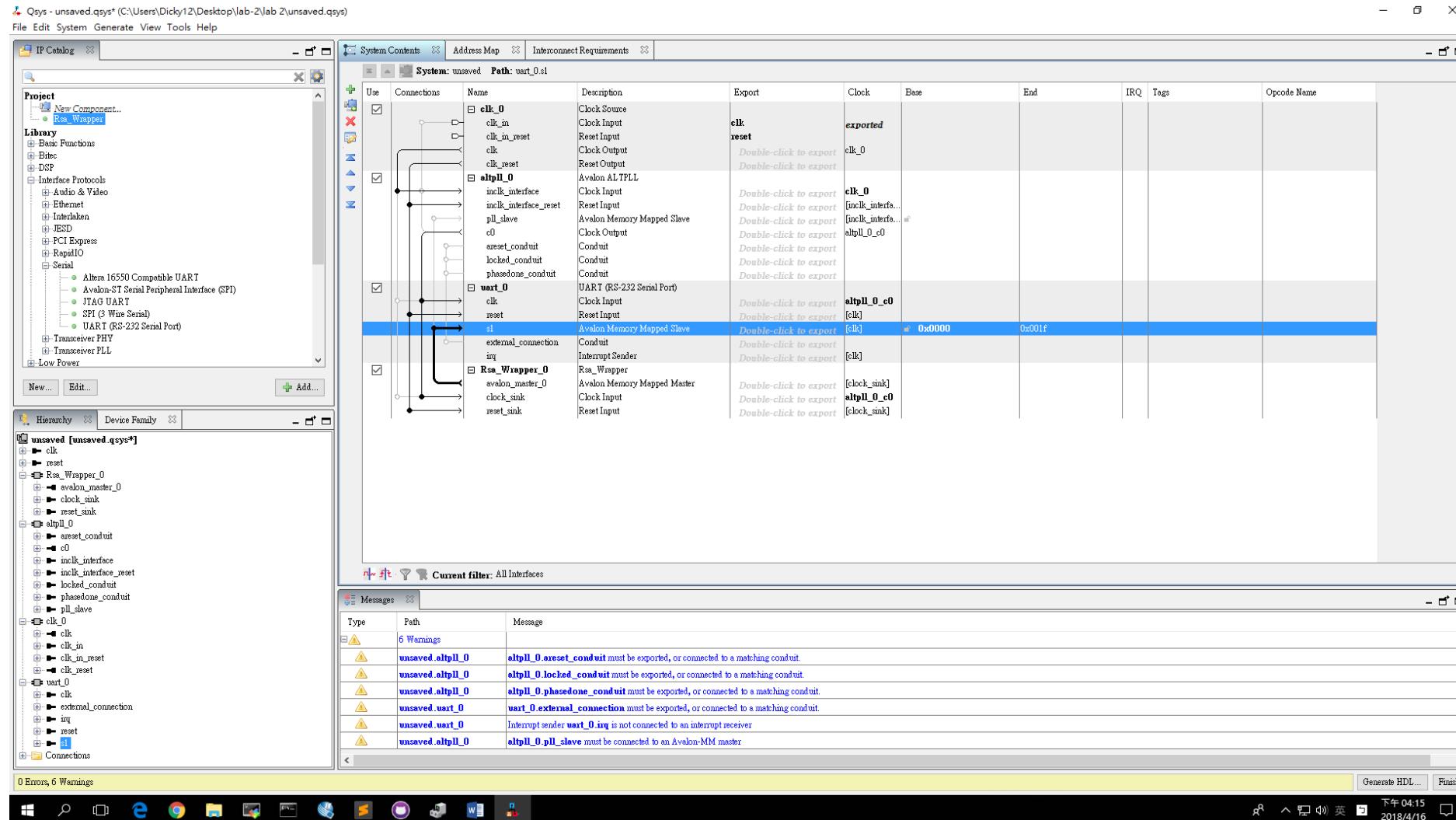
Finish → Yes, save  
將搜尋列字串清空，可見RsaWrapper在New Component底下



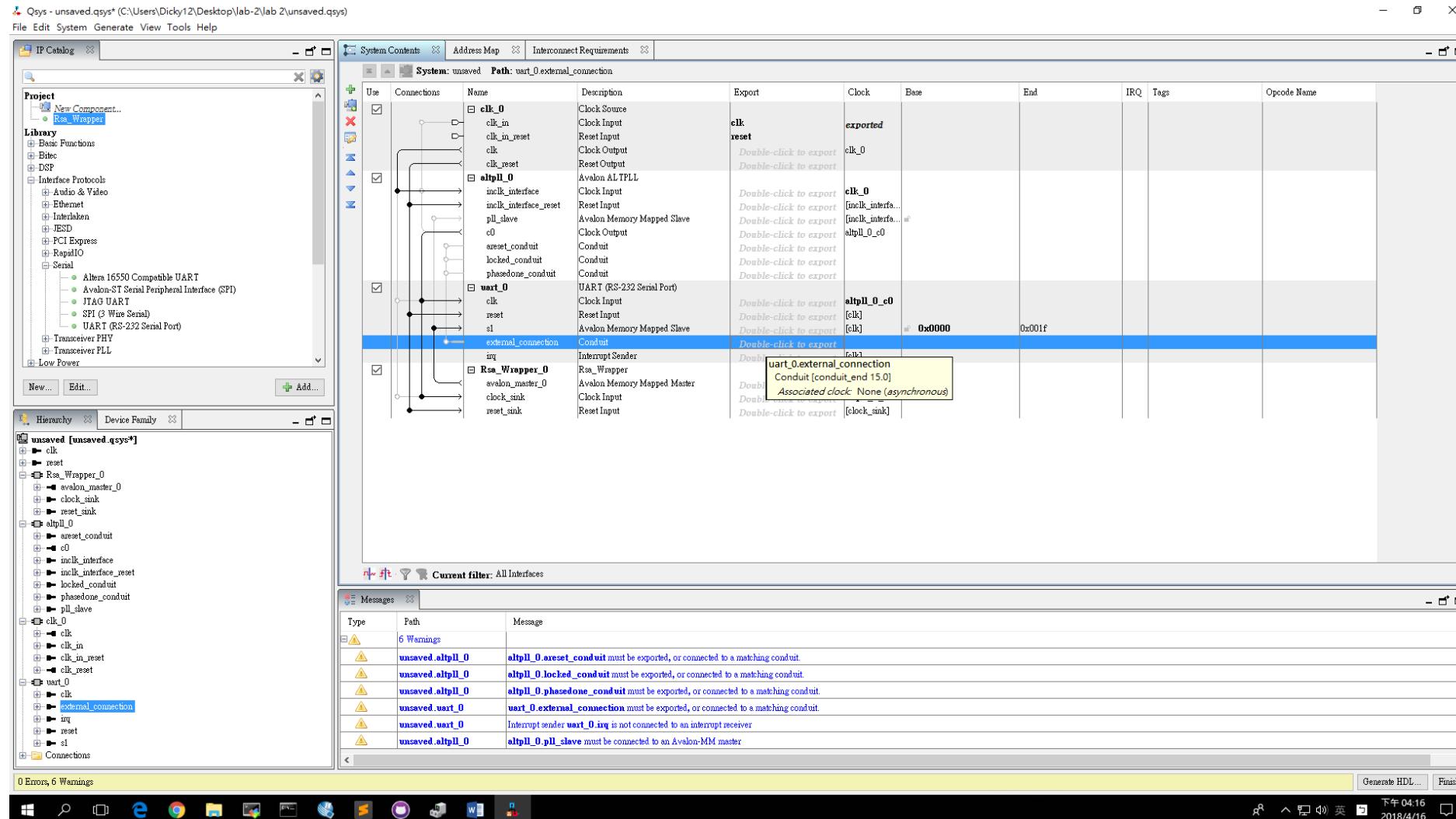
# 雙擊RsaWrapper → Finish



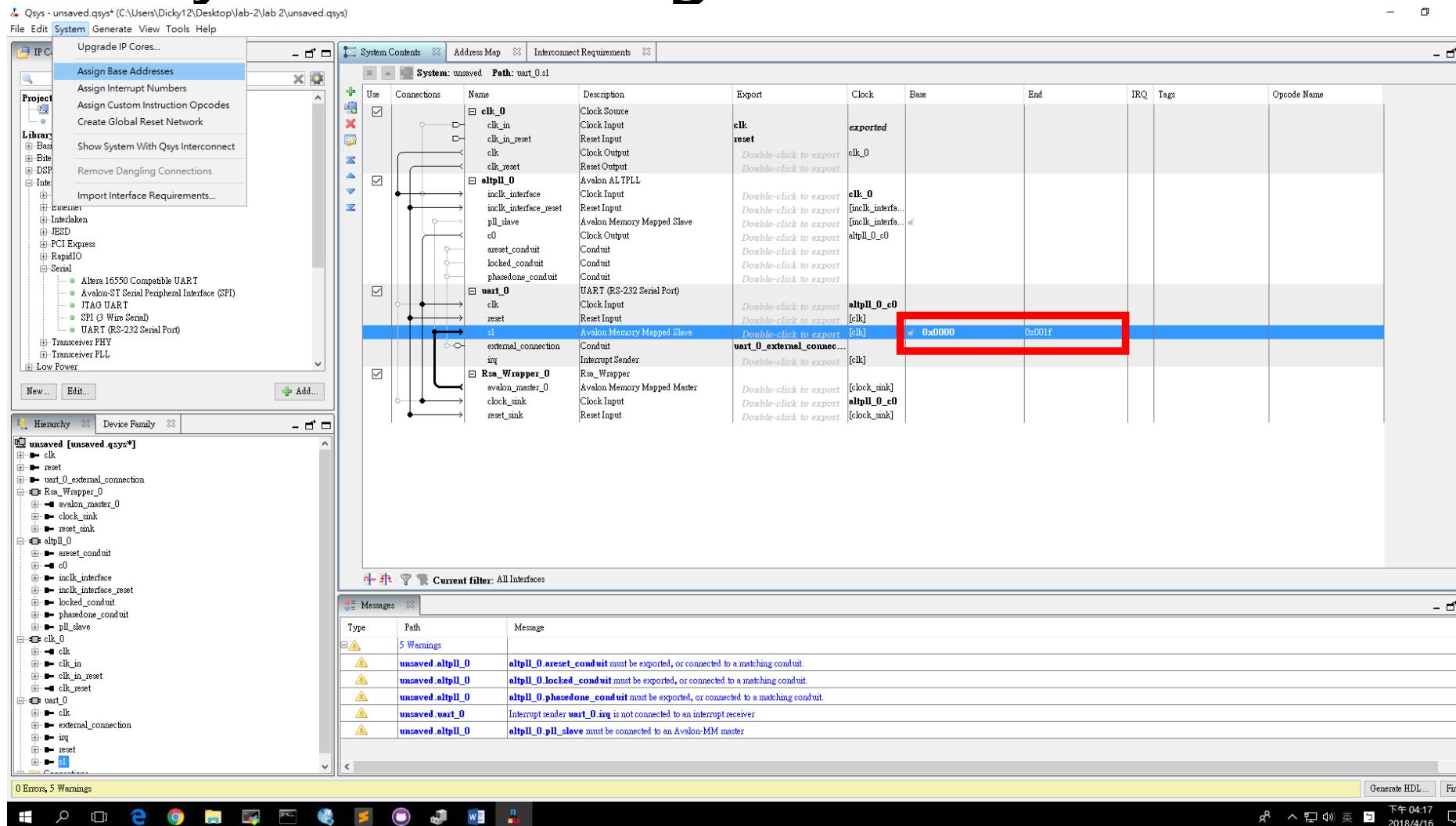
# 請照圖例接線 (下面Message內紅字會不見)



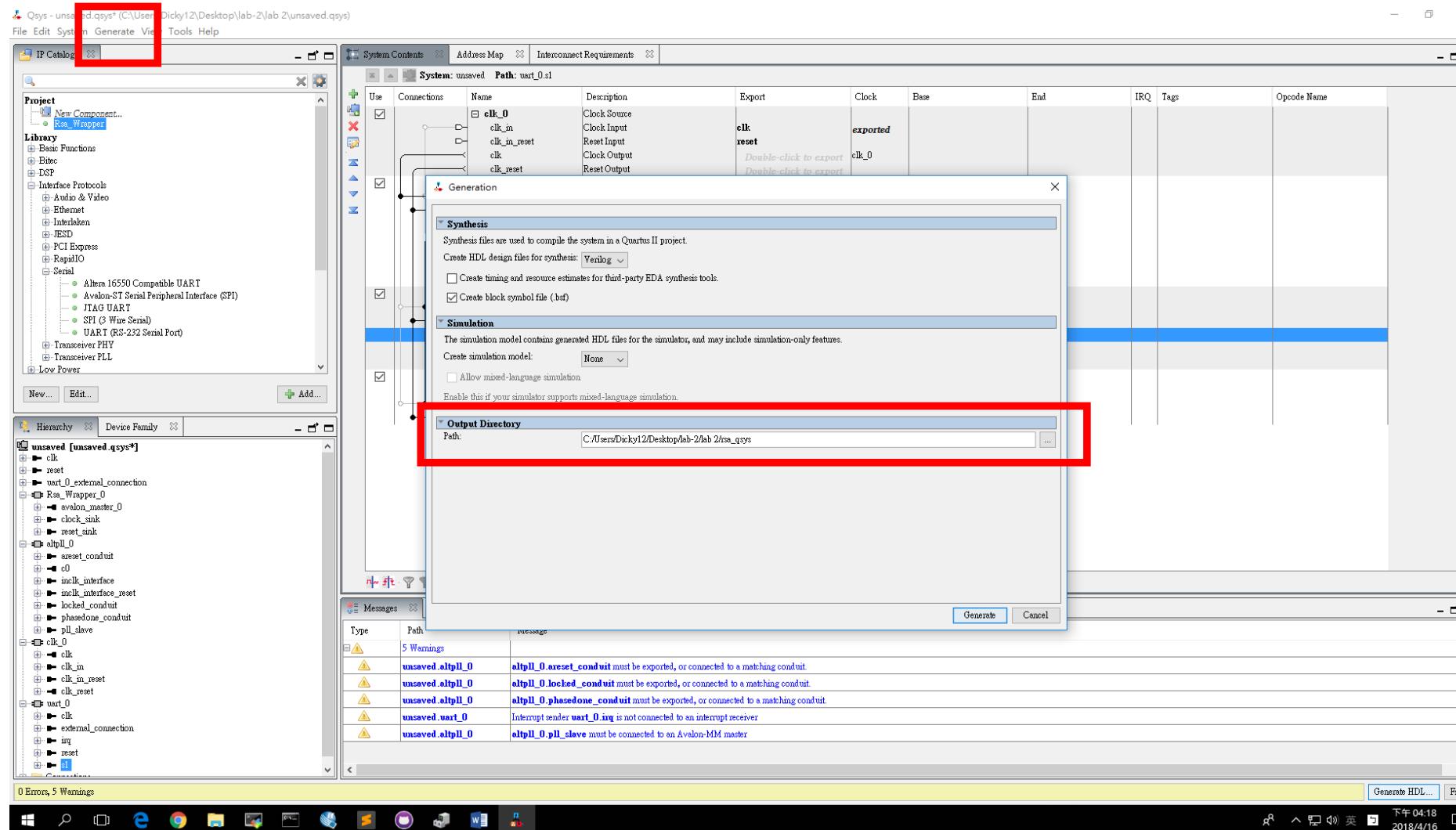
# 雙擊external\_connection的Export欄位，Enter



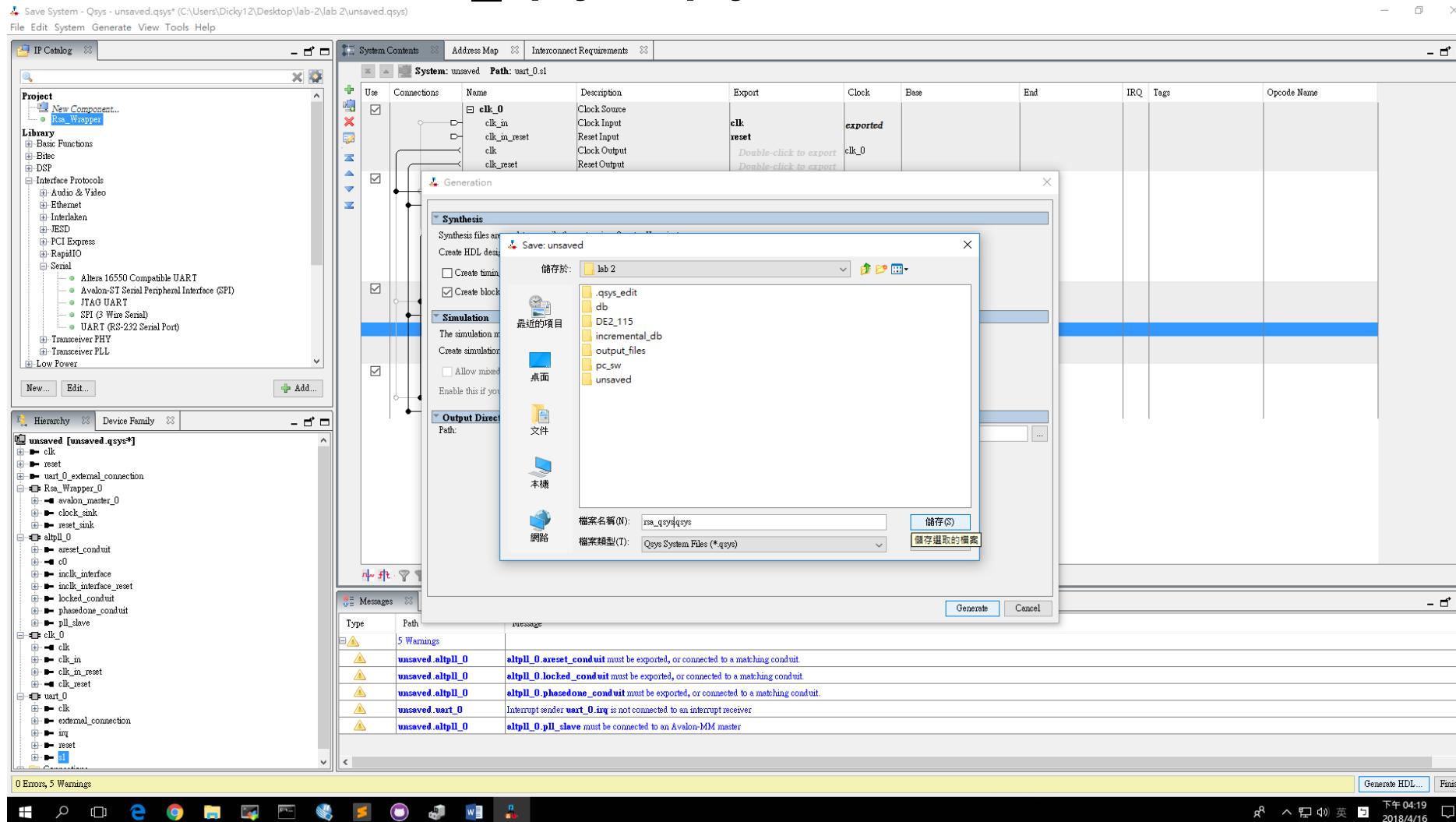
# 點s1的base欄位 → s1被反白選取 System → Assign Base Addresses



# Generate → Generate HDL ... → 將Output Directory的最後一層改成rsa\_qsys



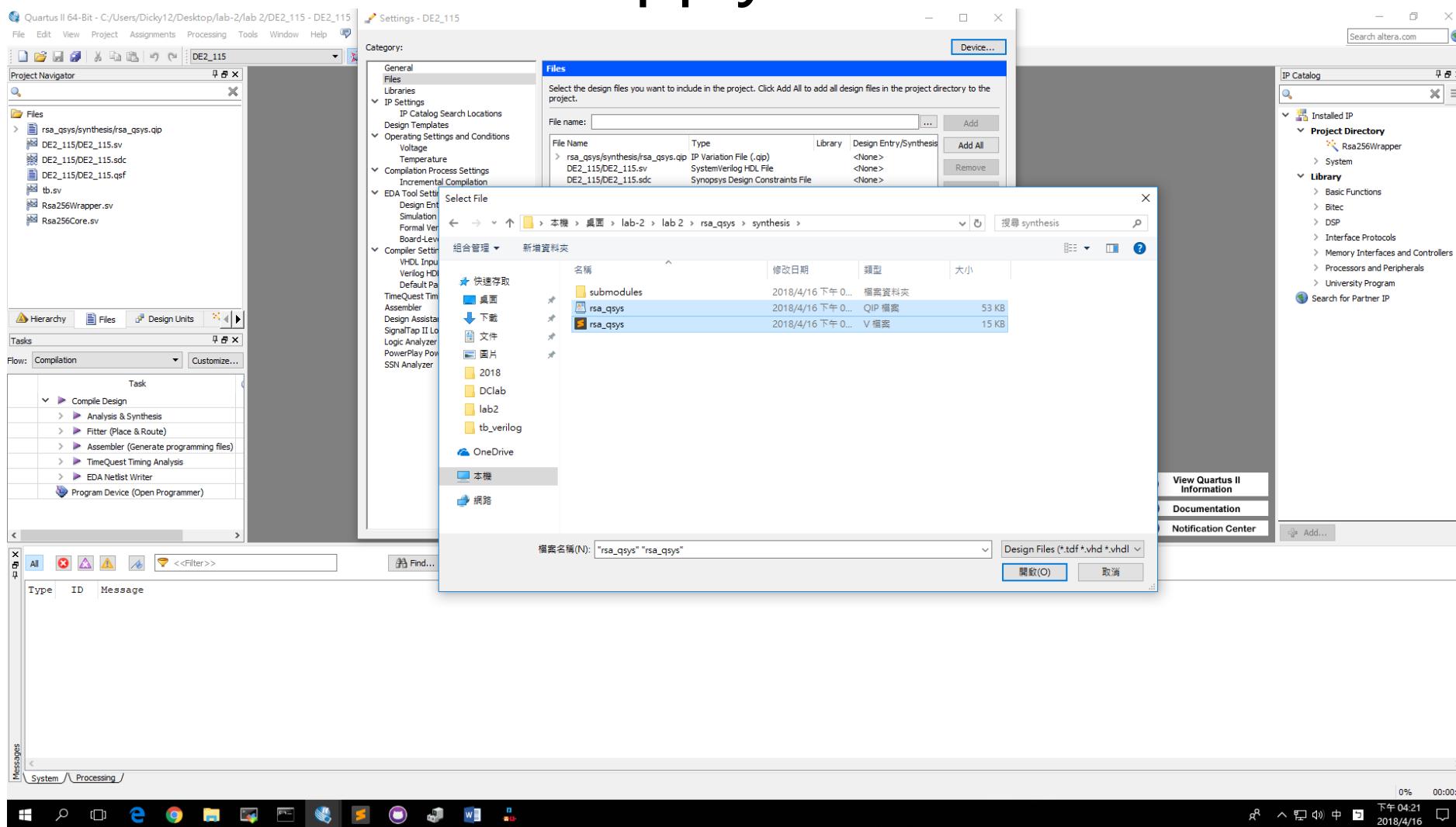
# Generate → Save 檔案名稱改成rsa\_qsys.qsys, 儲存, 完畢後按Close



# 回到Quartus視窗→Assignment → Settings



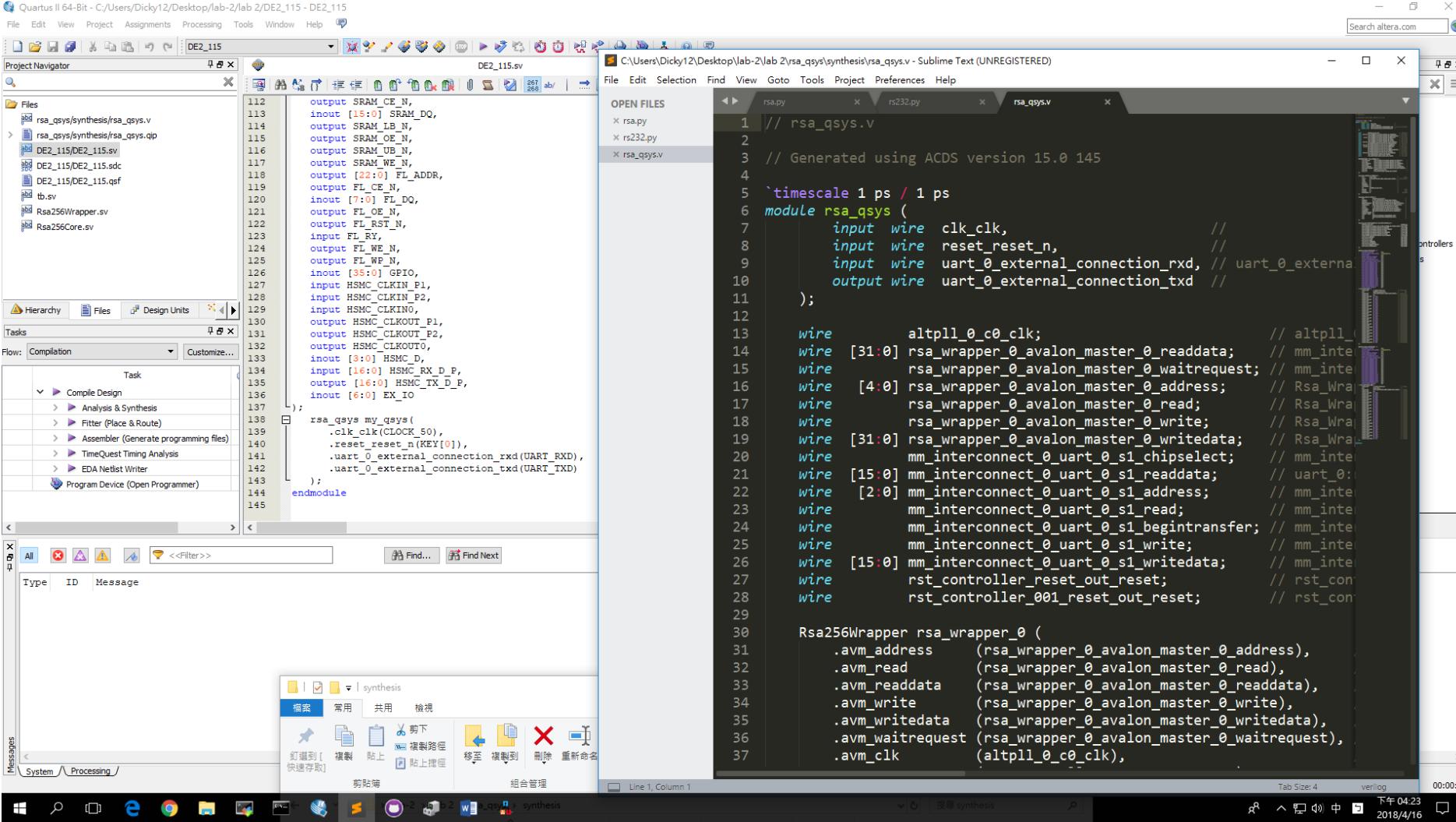
新增rsa\_qsys/synthesis内的rsa\_qsys.qip和rsa\_qsys.v  
→ Apply → OK



# 完成後可以見到Files內多出rsa\_qsys.v和rsa\_qsys.qip



# 檢查DE2\_115.sv內使用rsa\_qsys和rsa\_qsys.v的port要相同



The screenshot shows the Quartus II 64-Bit interface. On the left, the Project Navigator displays files like rsa\_qsys/synthesis/rsa\_qsys.v, DE2\_115.sv, and Rsa256Wrapper.sv. The main window shows the Verilog code for DE2\_115.sv:

```
112     output SRAM_CE_N,
113     inout [15:0] SRAM_DQ,
114     output SRAM_LB_N,
115     output SRAM_OE_N,
116     output SRAM_UB_N,
117     output SRAM_WE_N,
118     output [22:0] FL_ADDR,
119     output FL_CE_N,
120     inout [7:0] FL_DQ,
121     output FL_OE_N,
122     output FL_RST_N,
123     input FL_RY,
124     output FL_WE_N,
125     output FL_WP_N,
126     inout [35:0] GPIO,
127     input HSMC_CLKIN_P1,
128     input HSMC_CLKIN_P2,
129     input HSMC_CLKIN_O,
130     output HSMC_CLKOUT_P1,
131     output HSMC_CLKOUT_P2,
132     output HSMC_CLKOUT_O,
133     inout [3:0] HSMC_D,
134     input [16:0] HSMC_RX_D_P,
135     output [16:0] HSMC_TX_D_P,
136     inout [6:0] EX_IO
137   );
138   rsa_qsys my_qsys(
139     .clk_clk(CLOCK_50),
140     .reset_reset_n(KEY[0]),
141     .uart_0_external_connection_rxd(UART_RXD),
142     .uart_0_external_connection_txd(UART_TXD)
143   );
144 endmodule
```

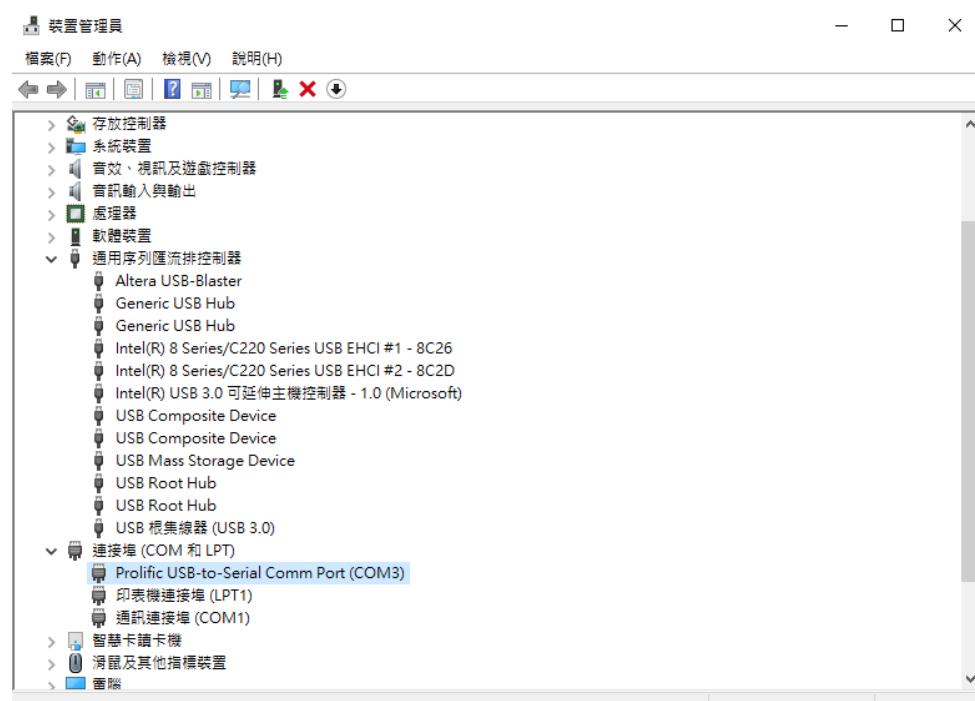
On the right, a Sublime Text window shows the Verilog code for rsa\_qsys.v:

```
// Generated using ACDS version 15.0 145
`timescale 1 ps / 1 ps
module rsa_qsys (
    input wire clk_clk,                                // altpll_0_c0_clk;
    input wire reset_reset_n,                           // mm_interconnect_0_mmresetn;
    input wire uart_0_external_connection_rxd,        // uart_0_external_connection_rxd;
    output wire uart_0_external_connection_txd // uart_0_external_connection_txd
);
wire altpll_0_c0_clk;                                // altpll_0_c0_clk;
wire [31:0] rsa_wrapper_0_avalon_master_0_readdata; // mm_interconnect_0_mmreaddata;
wire rsa_wrapper_0_avalon_master_0_waitrequest; // mm_interconnect_0_mmwaitrequest;
wire [4:0] rsa_wrapper_0_avalon_master_0_address; // Rsa_Wrapper_0_avalon_master_0_address;
wire rsa_wrapper_0_avalon_master_0_read; // Rsa_Wrapper_0_avalon_master_0_read;
wire rsa_wrapper_0_avalon_master_0_write; // Rsa_Wrapper_0_avalon_master_0_write;
wire [31:0] rsa_wrapper_0_avalon_master_0_writedata; // Rsa_Wrapper_0_avalon_master_0_writedata;
wire mm_interconnect_0_uart_0_s1_chipselect; // mm_interconnect_0_uart_0_s1_chipselect;
wire [15:0] mm_interconnect_0_uart_0_s1_readdata; // uart_0_mmreaddata;
wire [2:0] mm_interconnect_0_uart_0_s1_address; // mm_interconnect_0_uart_0_s1_address;
wire mm_interconnect_0_uart_0_s1_read; // mm_interconnect_0_uart_0_s1_read;
wire mm_interconnect_0_uart_0_s1_begintransfer; // mm_interconnect_0_uart_0_s1_begintransfer;
wire mm_interconnect_0_uart_0_s1_write; // mm_interconnect_0_uart_0_s1_write;
wire [15:0] mm_interconnect_0_uart_0_s1_writedata; // mm_interconnect_0_uart_0_s1_writedata;
wire rst_controller_reset_out_reset; // rst_controller_reset_out_reset;
wire rst_controller_001_reset_out_reset; // rst_controller_001_reset_out_reset;

Rsa256Wrapper rsa_wrapper_0 (
    .avm_address (rsa_wrapper_0_avalon_master_0_address),
    .avm_read (rsa_wrapper_0_avalon_master_0_read),
    .avm_readdata (rsa_wrapper_0_avalon_master_0_readdata),
    .avm_write (rsa_wrapper_0_avalon_master_0_write),
    .avm_writedata (rsa_wrapper_0_avalon_master_0_writedata),
    .avm_waitrequest (rsa_wrapper_0_avalon_master_0_waitrequest),
    .avm_clk (altpll_0_c0_clk),
);
```

# Appendix: Compile (Windows)

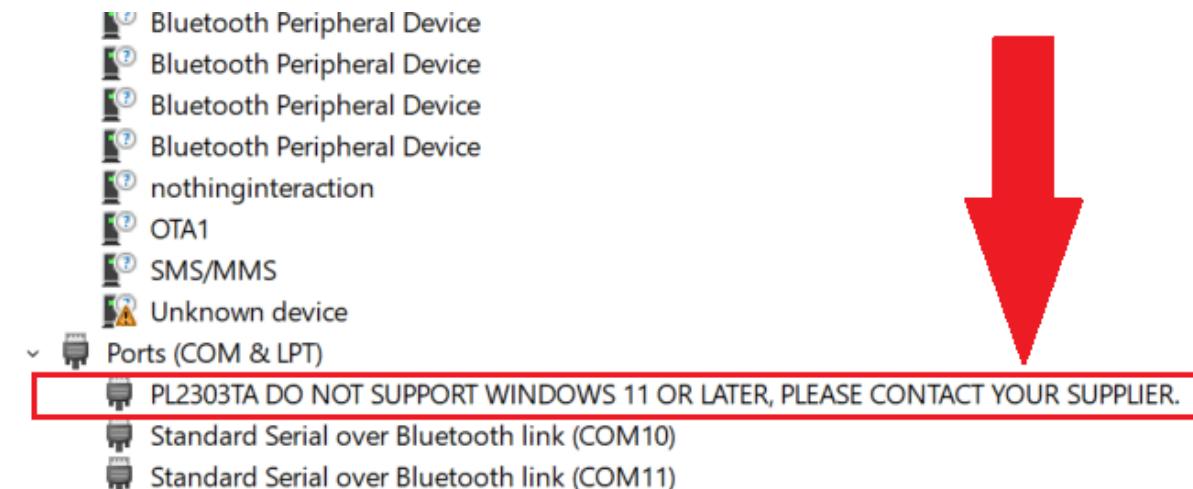
1. 裝置管理員 → 連接埠 (COM和LPT) → Prolific USB-to-Serial Common Port (COM?)
  - 1) 若沒有發現類似的裝置，則需要安裝對應型號的驅動程式
  - 2) COM?即為OS指派的port代號



# Appendix: Compile (Windows)

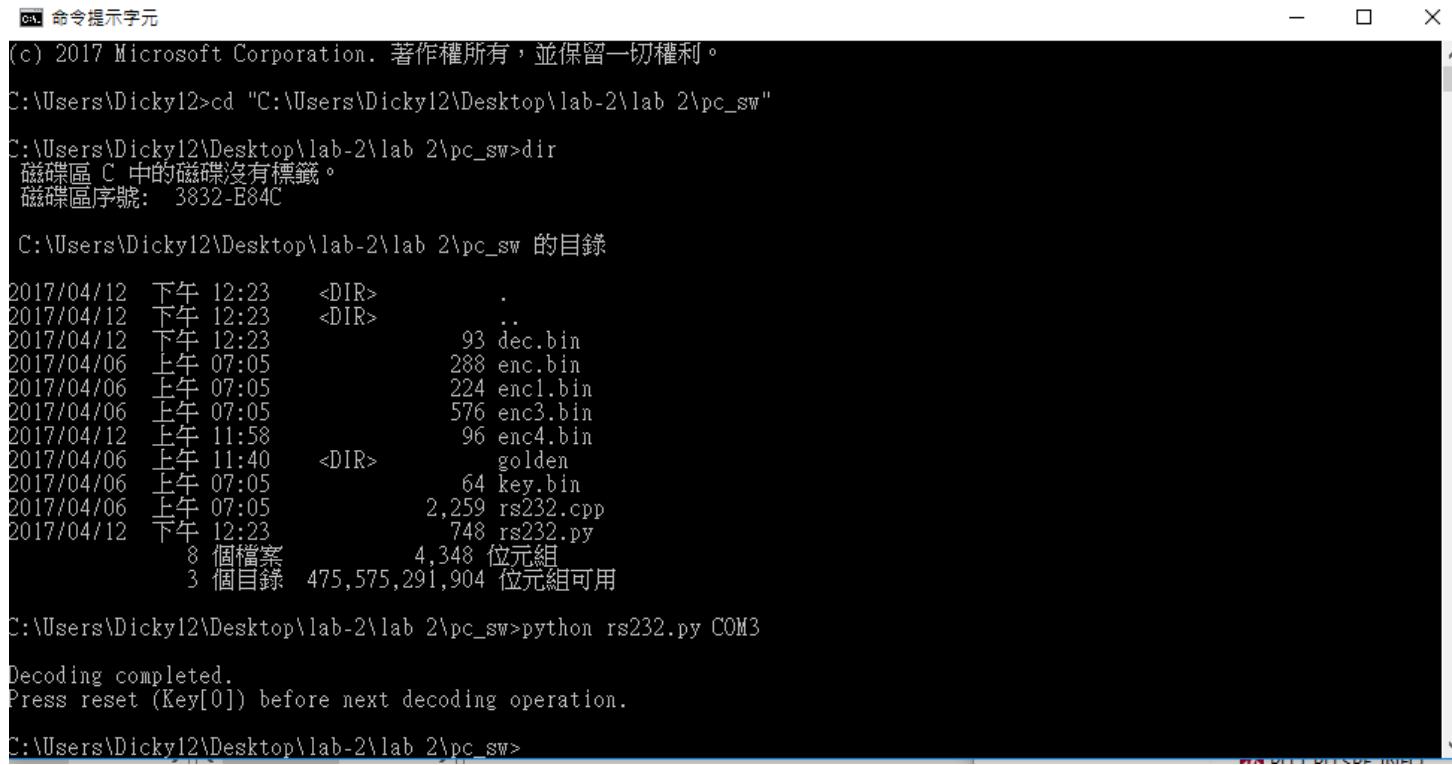
1. 裝置管理員→連接埠 (COM和LPT)→Prolific USB-to-Serial Common Port (COM?)
  - 3) 如果遇到DO NOT SUPPORT WINDOWS 11可以參考

<https://embetronicx.com/uncategorized/fixed-prolific-pl2303ta-usb-to-serial-and-windows-11/>



# Appendix: Compile (Windows)

2. 按下FPGA上的KEY[0] (做reset)
3. \$> python rs232.py COM?, ?從步驟1.可以得知



```
命令提示字元
(c) 2017 Microsoft Corporation. 著作權所有，並保留一切權利。
C:\Users\Dickey12>cd "C:\Users\Dickey12\Desktop\lab-2\lab 2\pc_sw"
C:\Users\Dickey12\Desktop\lab-2\lab 2\pc_sw>dir
磁碟區 C 中的磁碟沒有標籤。
磁碟區序號: 3832-E84C

C:\Users\Dickey12\Desktop\lab-2\lab 2\pc_sw 的目錄

2017/04/12 下午 12:23 <DIR> .
2017/04/12 下午 12:23 <DIR> ..
2017/04/12 下午 12:23 93 dec.bin
2017/04/06 上午 07:05 288 enc.bin
2017/04/06 上午 07:05 224 enc1.bin
2017/04/06 上午 07:05 576 enc3.bin
2017/04/12 上午 11:58 96 enc4.bin
2017/04/06 上午 11:40 <DIR> golden
2017/04/06 上午 07:05 64 key.bin
2017/04/06 上午 07:05 2,259 rs232.cpp
2017/04/12 下午 12:23 748 rs232.py
               8 個檔案   4,348 位元組
               3 個目錄 475,575,291,904 位元組可用

C:\Users\Dickey12\Desktop\lab-2\lab 2\pc_sw>python rs232.py COM3
Decoding completed.
Press reset (Key[0]) before next decoding operation.
C:\Users\Dickey12\Desktop\lab-2\lab 2\pc_sw>
```

# Appendix: Compile (Windows)

4. 若要測試不同的enc檔，則將rs232.py內的fp\_enc = open()的檔案路徑改成golden/enc?.bin
5. 完成後，請以sublime等編輯器打開dec.bin(或對應檔名)
6. 經驗法則
  - 1) Compile時間過短或使用的logic elements數量過少，就算compile successful也可能出錯
  - 2) 重新qsys的話，可以將qsys這個資料夾砍掉直接重來
  - 3) Critical warning: Timing requirement not met代表qsys無法滿足timing的要求。有時候可能導致每一次compile的結果不盡相同