



SoC FPGA
パケット生成論理 導入手順書

【ご注意】

- (1) 本書の内容の一部または、全部を無断転載することは禁止されています。
- (2) 本書の内容については、改良のため予告なしに変更することがあります。
- (3) 本書の内容について、ご不明な点やお気付きの点がありましたら、ご連絡ください。
- (4) 本製品を運用した結果の影響については、(3)項にかかわらず責任を負いかねますのでご了承ください。

本マニュアルに記載されている企業名、システム名、製品名は、各社の商標または登録商標です。
なお、本文中では、TM、R マークは明記していません。

©2013 DTS INSIGHT CORPORATION. All rights reserved

Printed in Japan

改訂履歴

版	発行日付	変更内容
第 1 版	2013.06.12	新規発行
第 2 版	2013.12.25	TRQerS 対応

目次

1	はじめに.....	1
2	パケット生成論理組み込み手順.....	2
2.1	既存プロジェクトへのパケット生成論理組み込み準備	2
2.2	Qsys でのパケット生成論理組み込み手順	3
2.3	TOP モジュールの編集.....	18
2.4	Quartus II コンパイル手順.....	21

1 はじめに

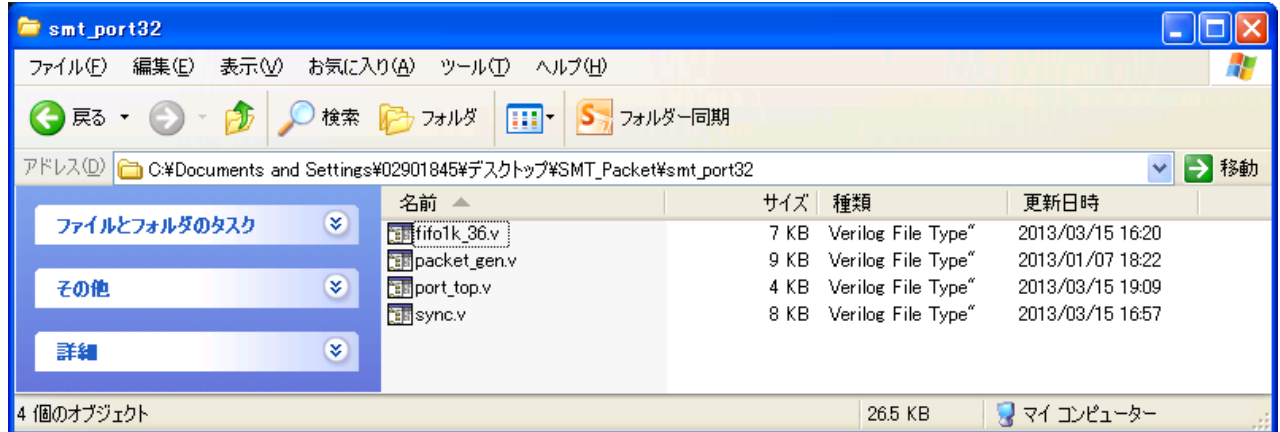
本マニュアルでは、adviceLUNA/TRQerS システムマクロトレース（以下 SMT）のパケットインターフェースに対応するためのパケット生成論理について、SoC FPGA への導入手順を説明します。

また、本マニュアルでは、アルティマ社製の Cyclon V SoC 評価ボード” Helio” を例にしています。

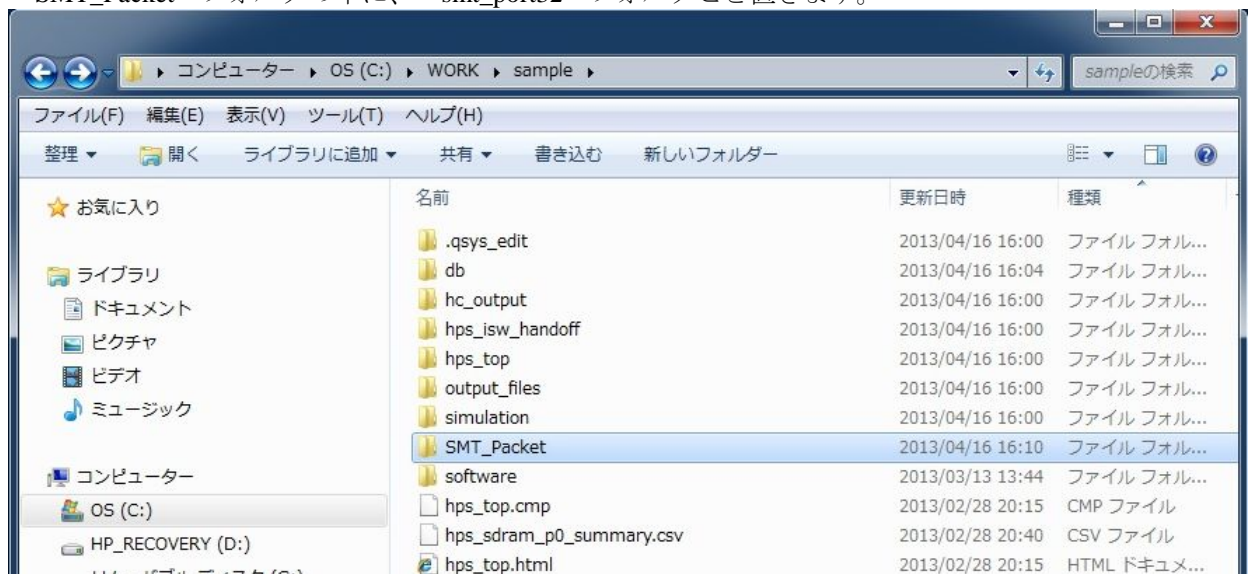
2 パケット生成論理組み込み手順

2.1 既存プロジェクトへのパケット生成論理組み込み準備

- 1) SoC FPGA 用パケット生成論理一式を用意します。

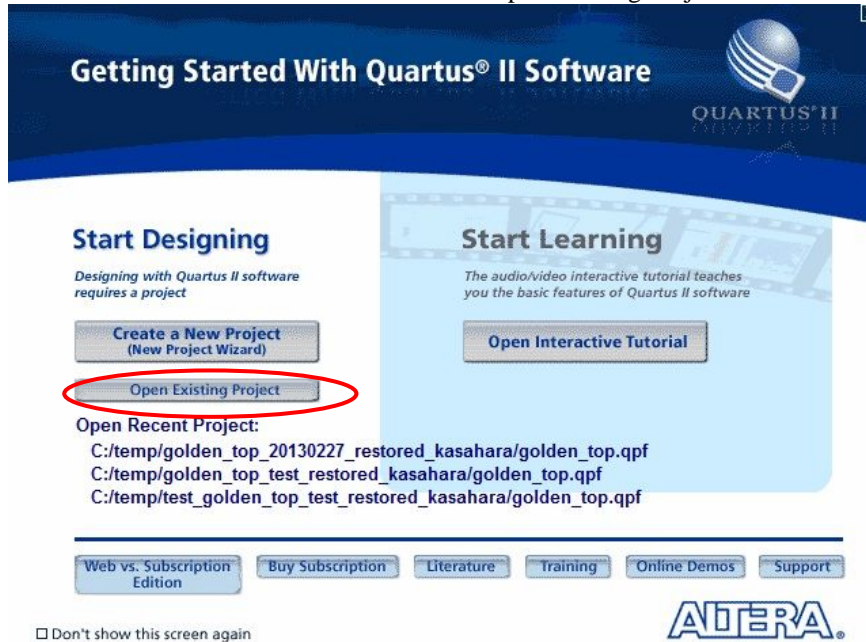


- 2) Quartus II の既存のプロジェクトがあるフォルダに、新規フォルダ “SMT_Packet” を生成し、“SMT_Packet” フォルダの中に、“smt_port32” フォルダごと置きます。

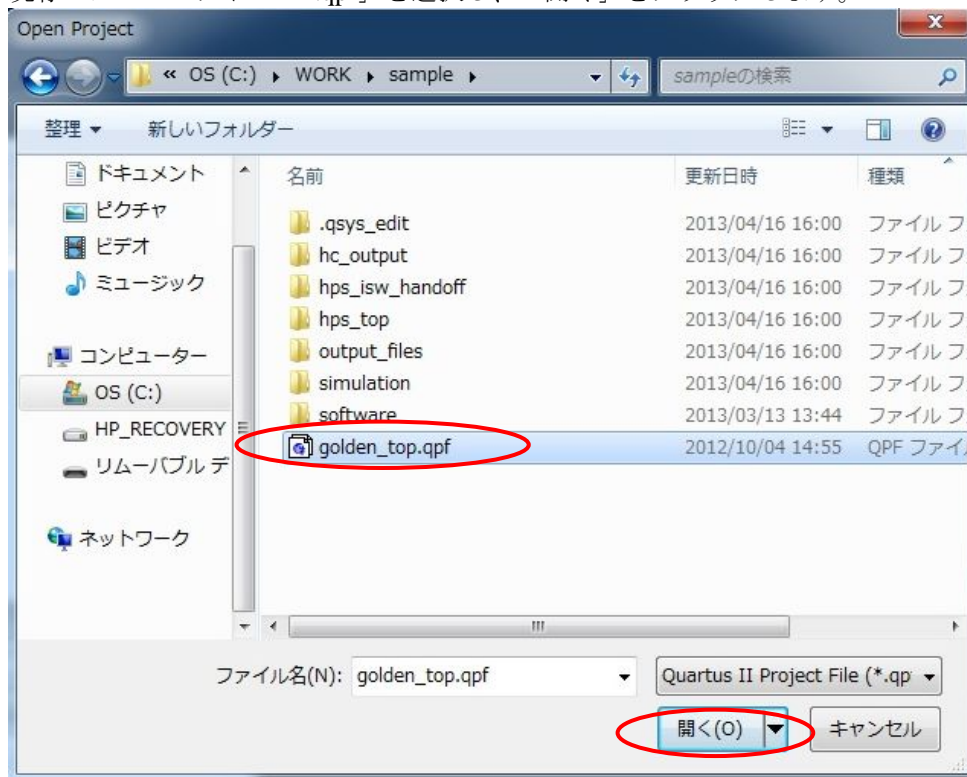


2.2 Qsys でのパケット生成論理組み込み手順

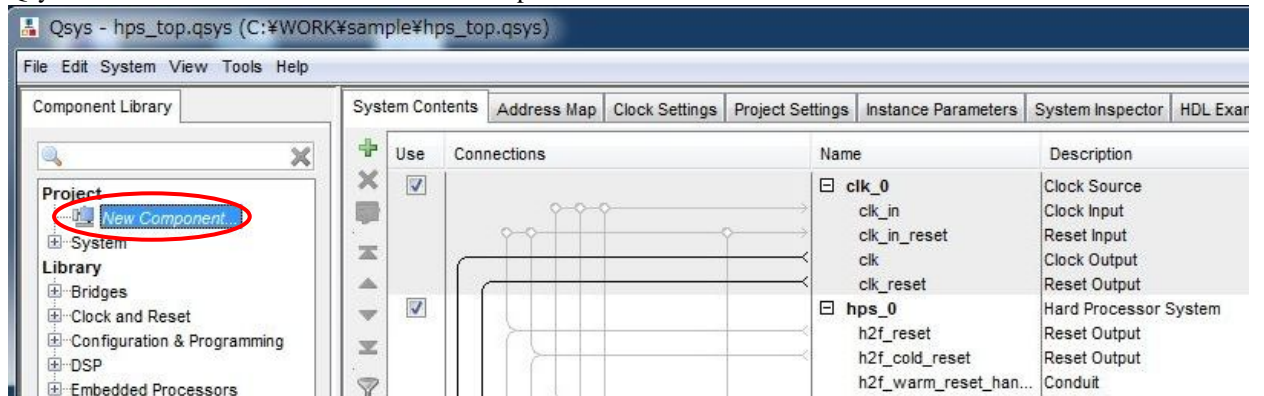
- 1) 既存のプロジェクトを立ち上げます。「Open Existing Project」をクリックします。



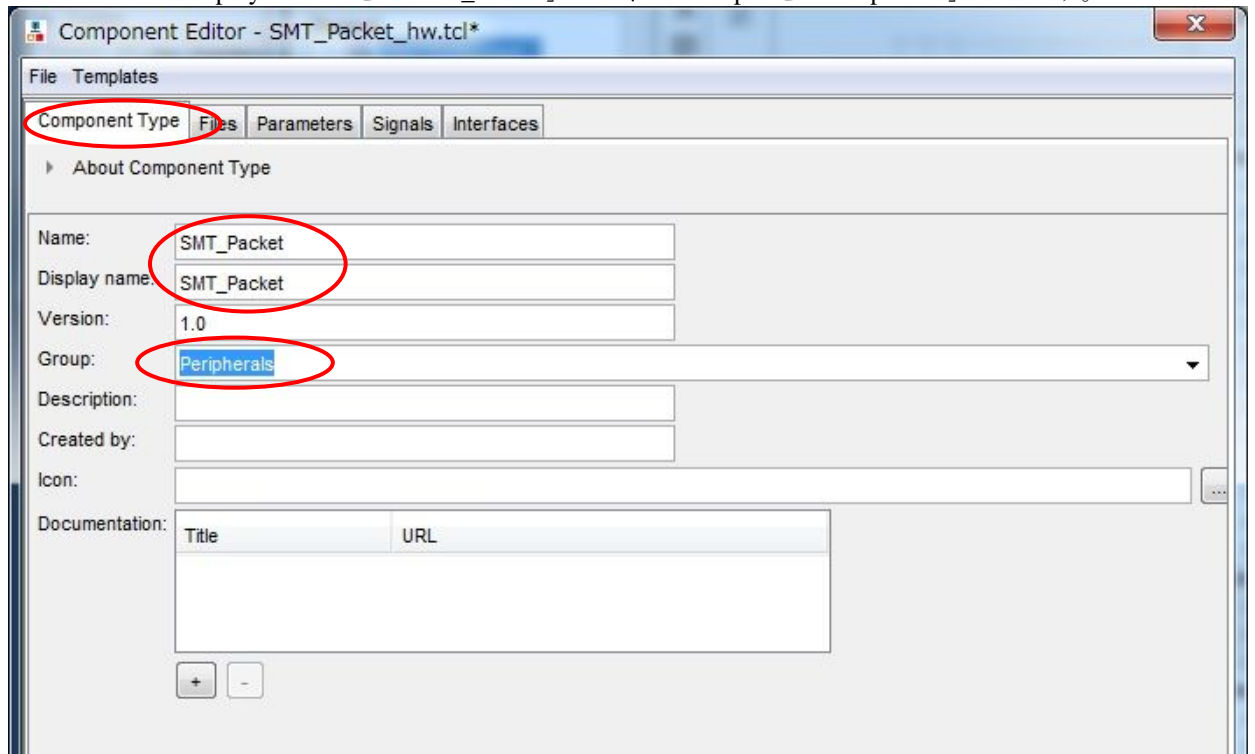
- 2) 既存のプロジェクト「xxx.qpf」を選択し、「開く」をクリックします。



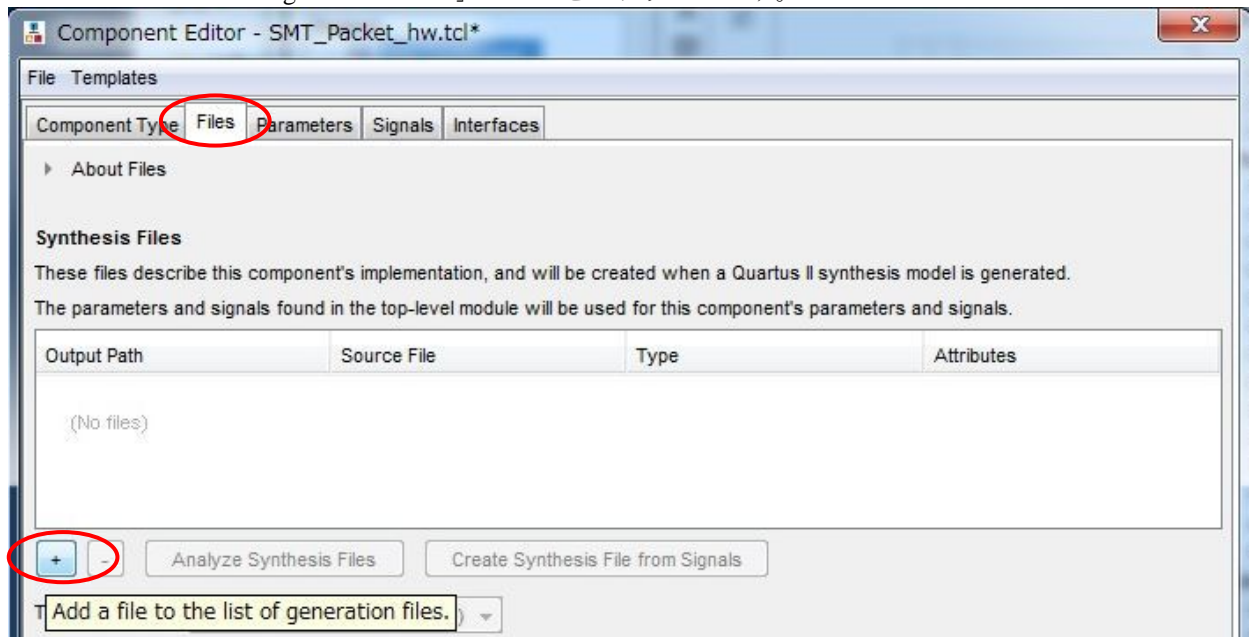
- 3) Qsys が立ち上がりますので、「New Component…」をダブルクリックします。



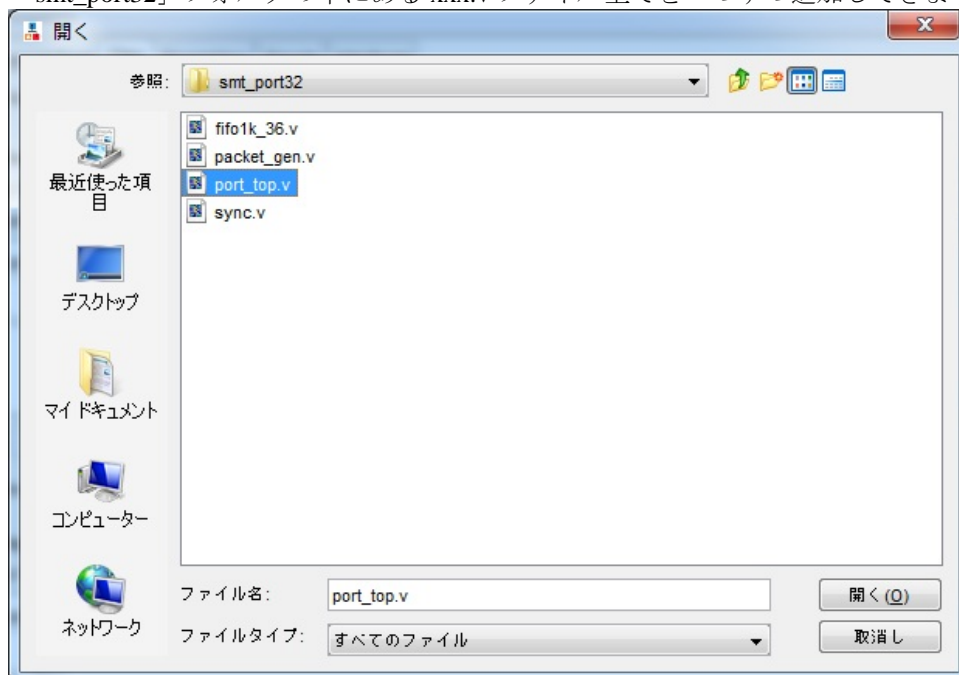
- 4) Component Editor が立ち上がりますので、「Component Type」タブの設定から開始します。
“Name”と“Display name”を「SMT_Packet」とし、“Group”を「Peripherals」とします。



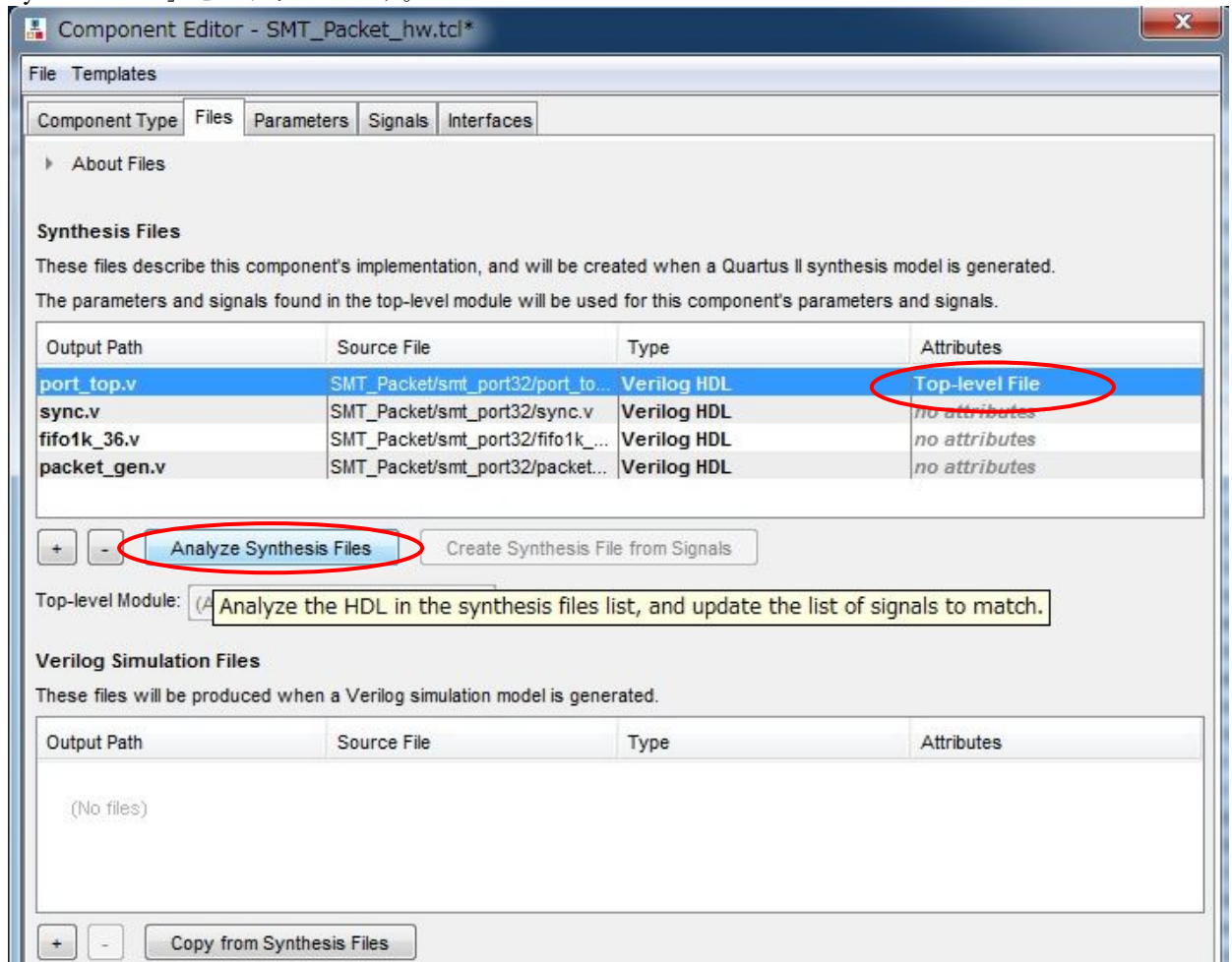
- 5) 「Files」タブでは SMT パケット論理を追加していきます。
「Add a file to the list of generation files.」 ボタンをクリックします。



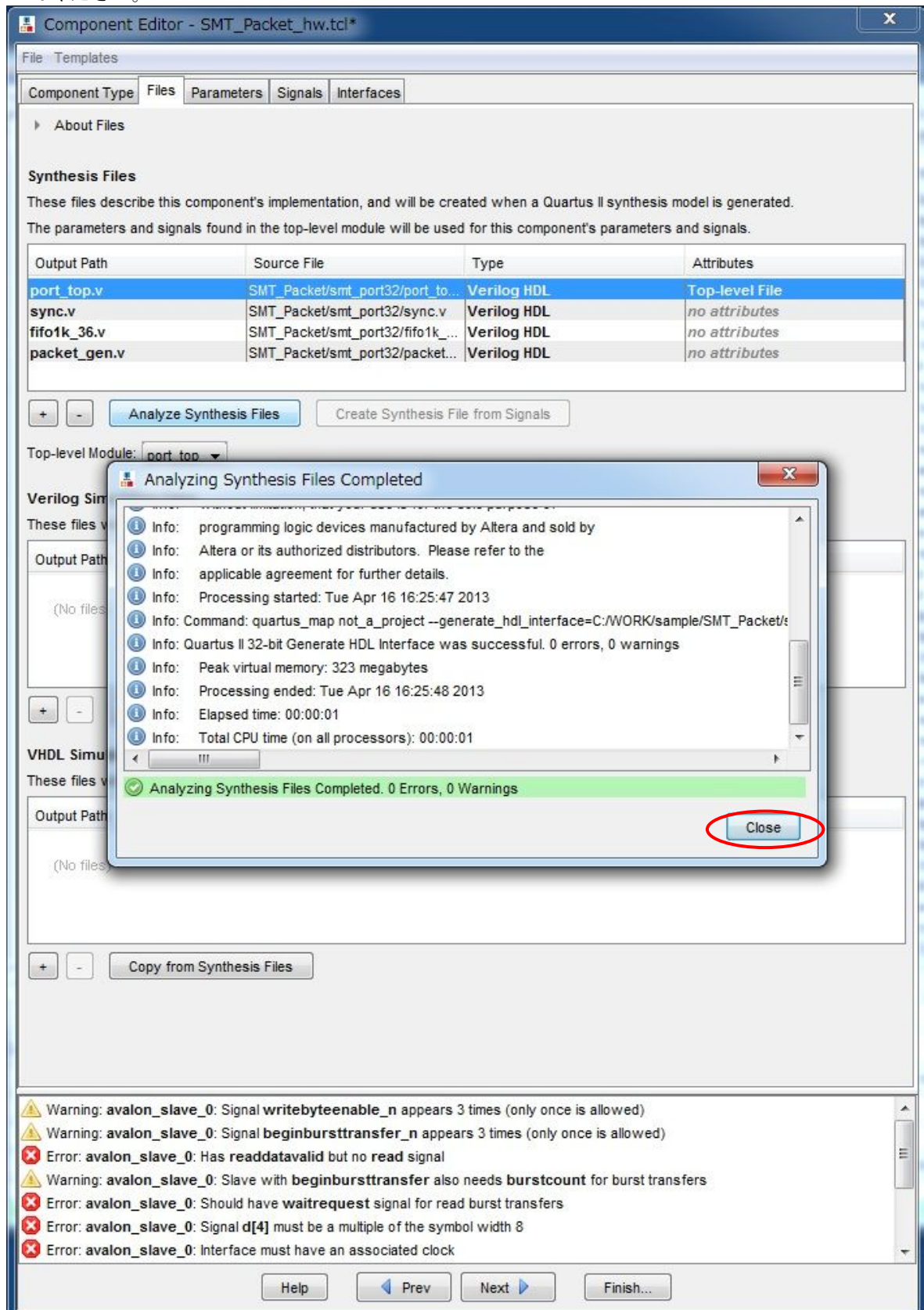
- 6) 「smt_port32」フォルダの中にある xxx.v ファイル全てを一つずつ追加していきます。



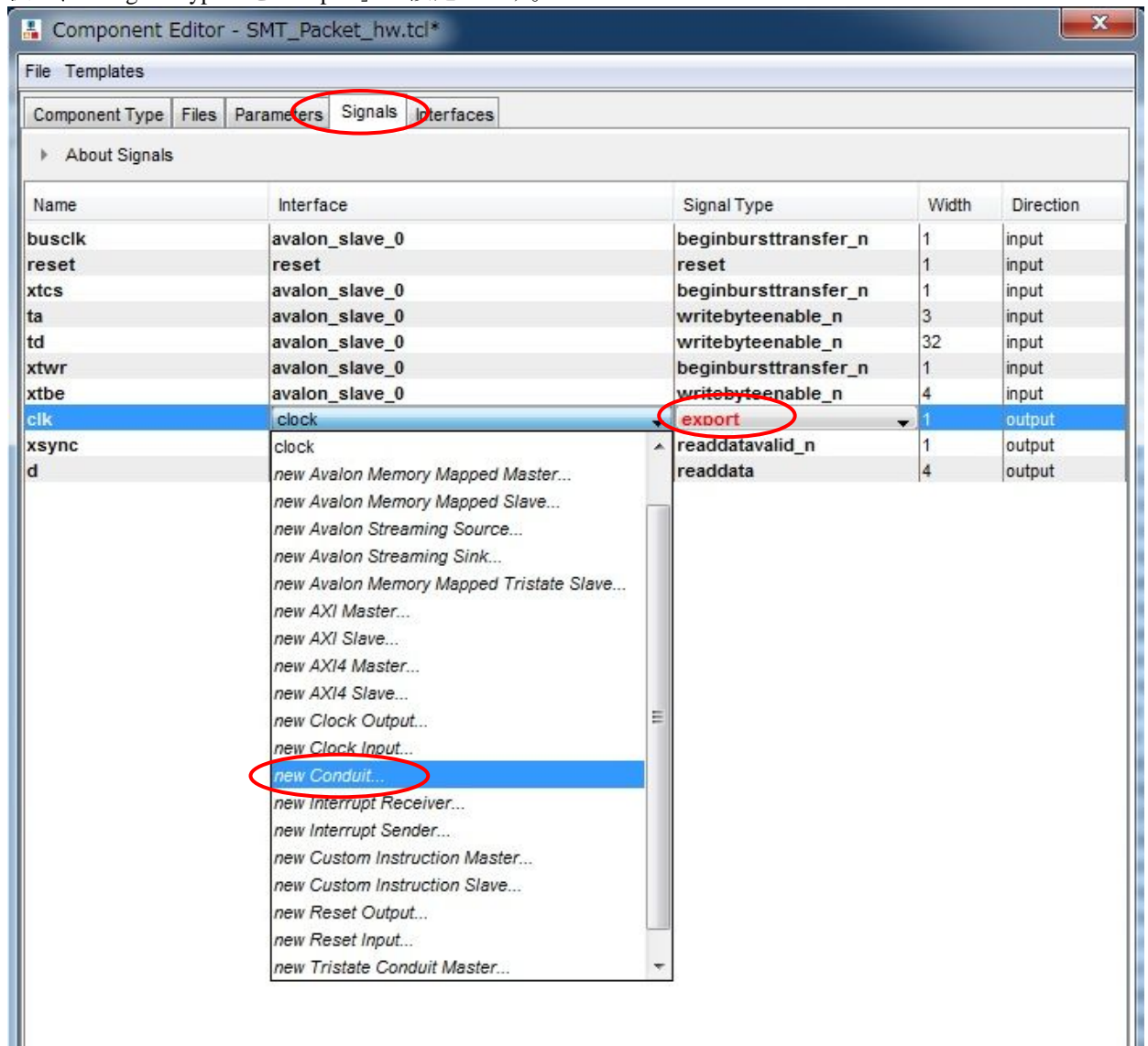
- 7) 全て登録が完了しましたら「port_top.v」が「Top-level Files」になっていることを確認し、「Analyze Synthesis Files」をクリックします。



- 8) 「Analyze Synthesis Files」が完了し、以下のダイアログが出てくるので「Close」をクリックします。
ここで、Error が多数出ていますが、後の設定でこの Error は消えていきますので、ここでは気にしないでください。



- 9) 「Signals」タブにて、各信号の定義を行います。まずは、「clk」の”Interface”で「new Conduit」を選択し、”Signal Type”を「export」に設定します。



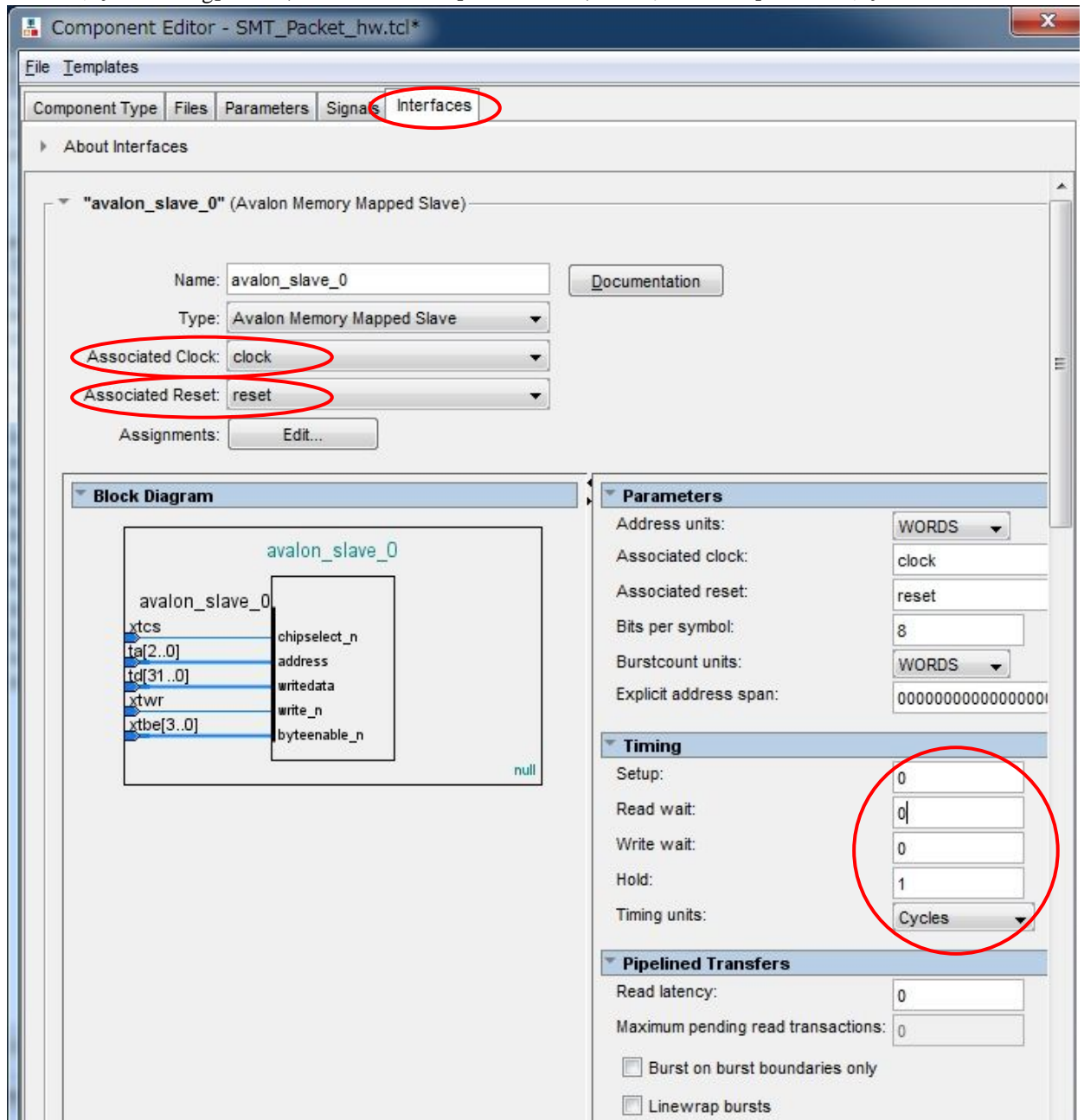
- 10) SMT の全信号に対し、以下の様に設定してください。

About Signals				
Name	Interface	Signal Type	Width	Direction
busclk	clock	clk	1	input
reset	reset	reset	1	input
xtcs	avalon_slave_0	chipselect_n	1	input
ta	avalon_slave_0	address	3	input
td	avalon_slave_0	writedata	32	input
xtwr	avalon_slave_0	write_n	1	input
xtbe	avalon_slave_0	byteenable_n	4	input
clk	conduit_end	export	1	output
xsync	conduit_end	export	1	output
d	conduit_end	export	4	output

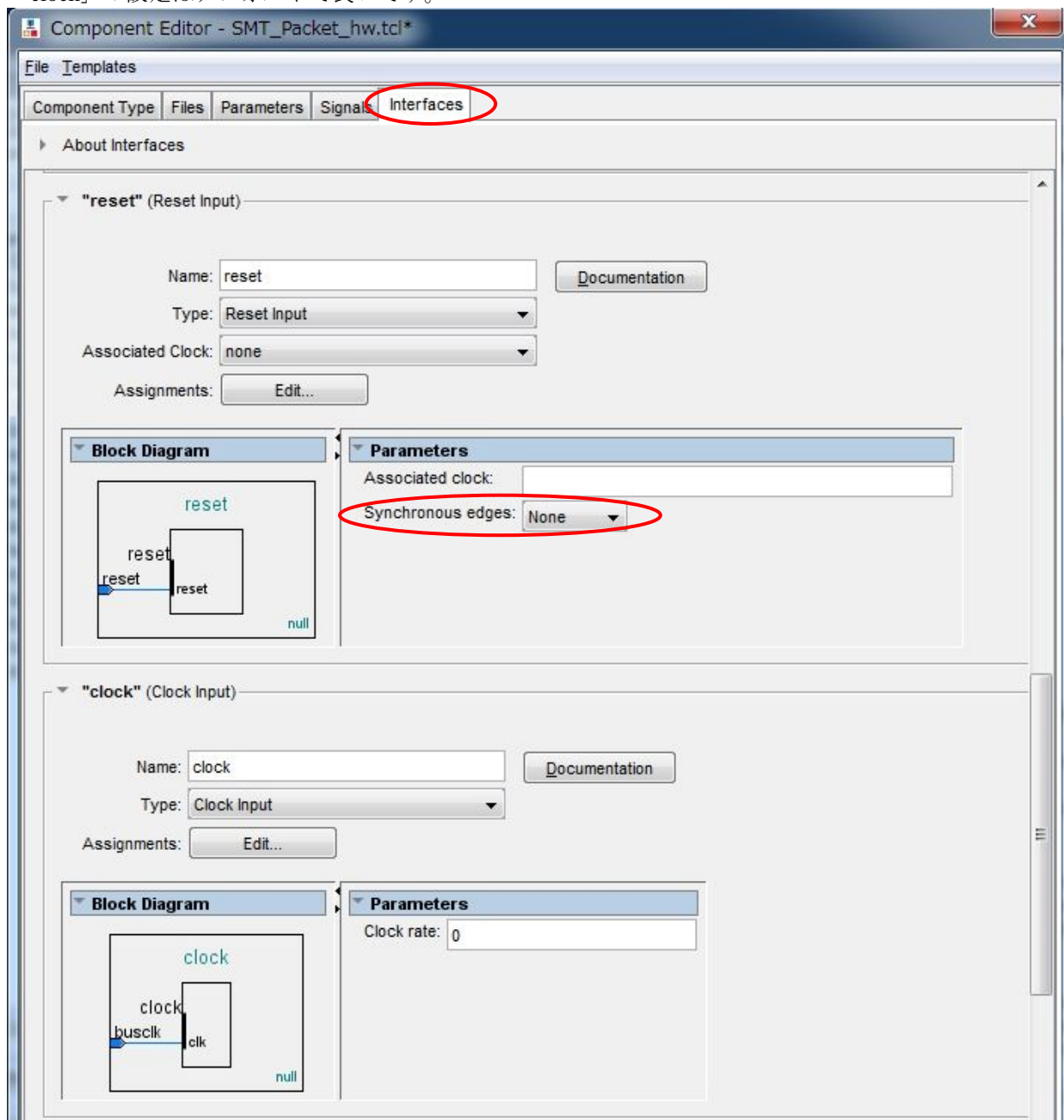
【注意】 xtcs, xtwr, xtbe は、Low アクティブ信号であるため、Signal Type は「_n」付きの信号を選択してください。

- 11) 「Interface」タブでは、以下の設定を行います。

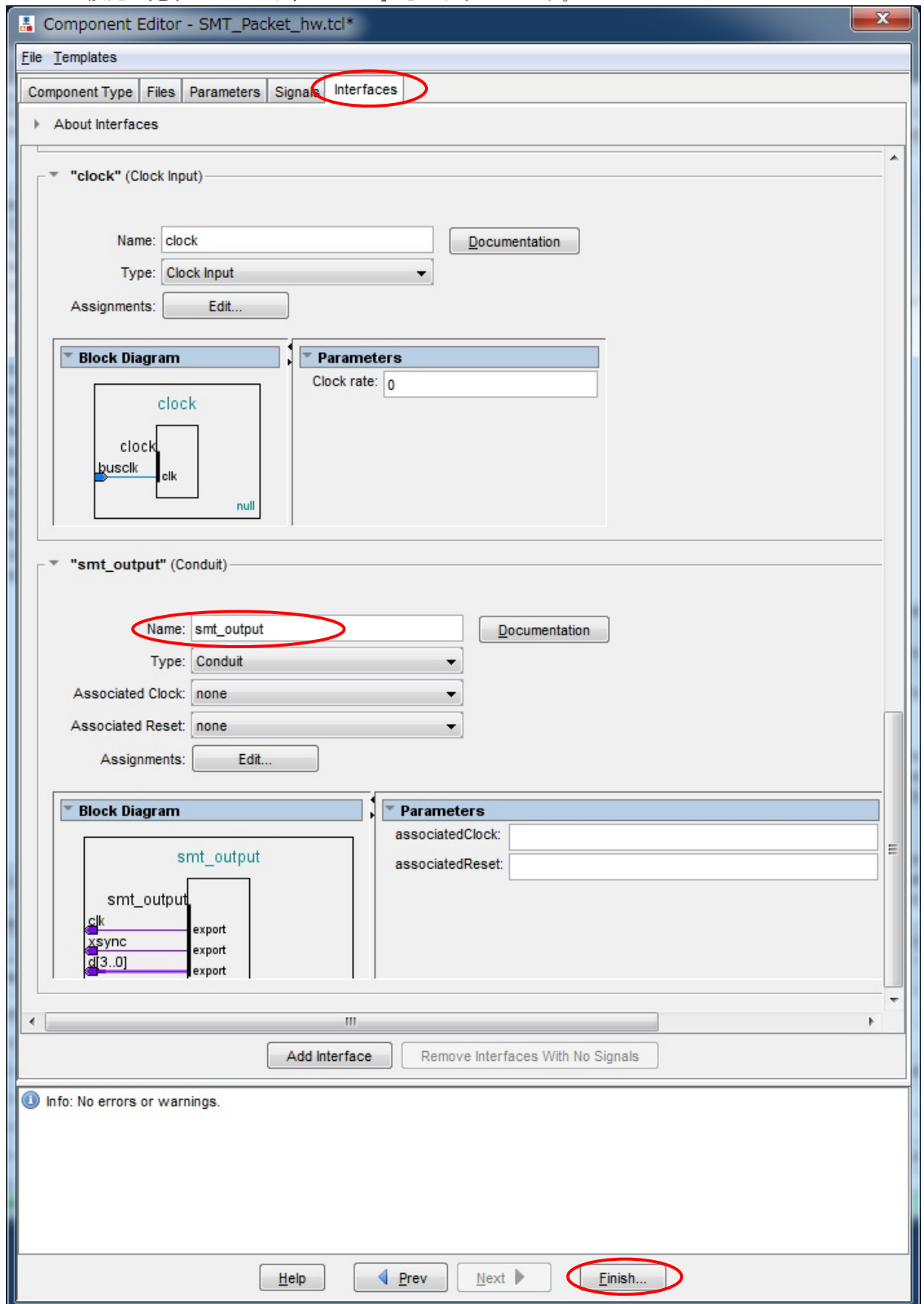
「avalon_slave_0」のバス設定にて、“Associated Clock”を「clock」とし、“Associated Reset”は「reset」とします。「Timing」では、“Hold”を「1」clockとし、それ以外は「0」とします。



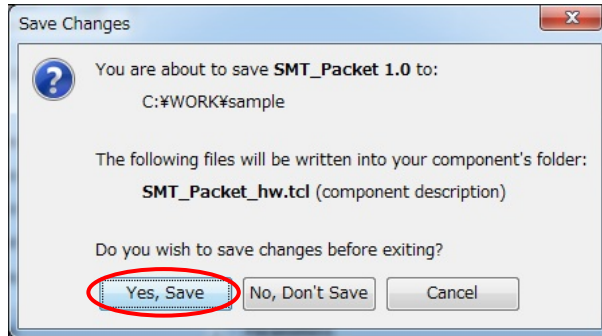
- 12) 「Interface」タブの「reset」の設定にて、“Synchronous edges”を「None」にします。
「clock」の設定はデフォルトで良いです。



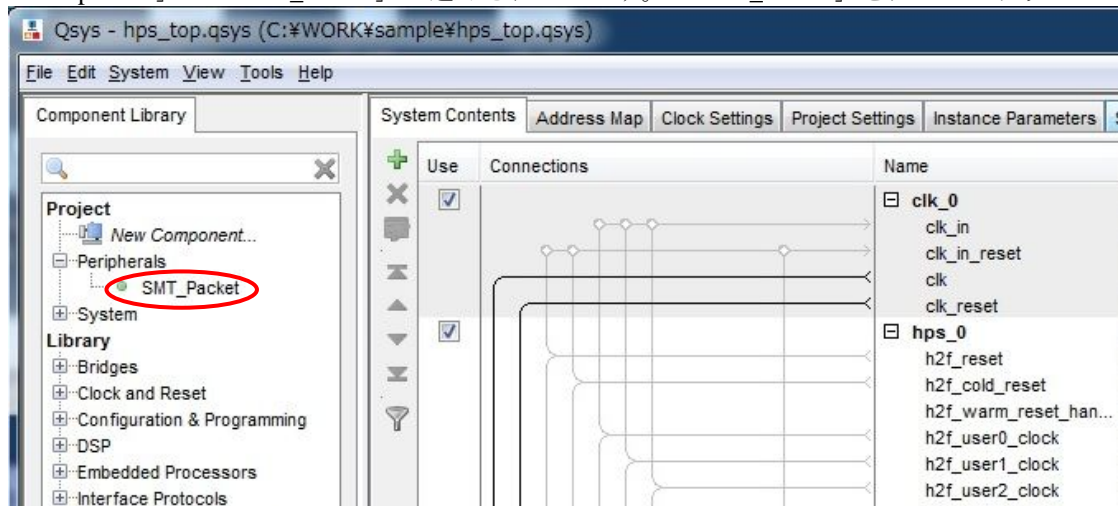
- 13) 「Interface」タブの「smt_output」の設定にて、“Name”を「smt_output」にします。
ここまで設定が完了しましたら、「Finish」をクリックします。



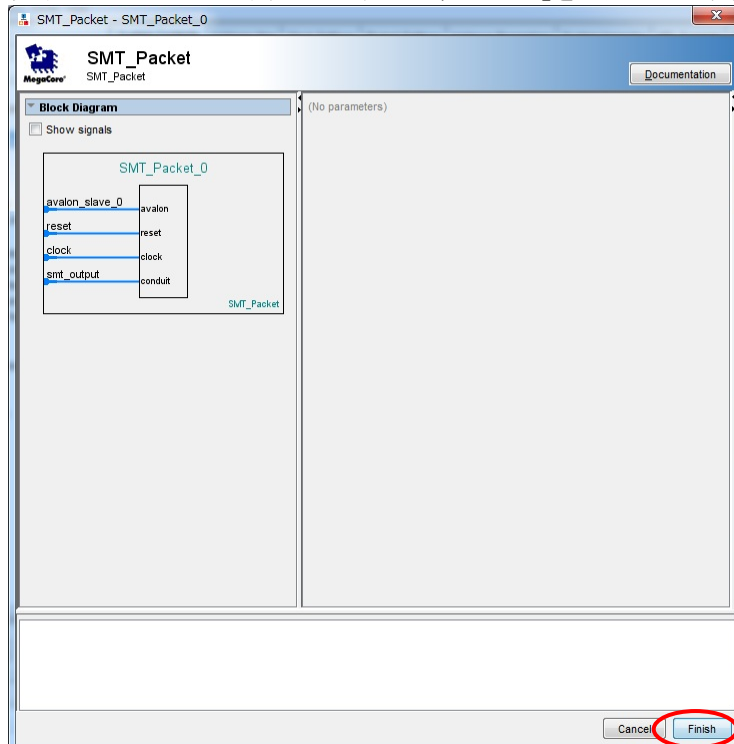
- 14) 以下のダイアログが出てくるので、「Yes, Save」をクリックします。
これでパケット論理が組み込める状態になりました。



- 15) 「Peripherals」に「SMT_Packet」が追加されています。「SMT_Packet」をダブルクリックします。



- 16) 以下のダイアログが出てきますので、「Finish」をクリックします。



- 17) 「SMT_Packet_0」が組み込まれました。ここで Error が出ていますが、接続が完了すればこの Error は消えますので、ここでは気にしないでください。

ここからバスの接続を行います。まずは見やすい様に IP の順番を入れ替えます。

「SMT_Packet_0」を選択し、上矢印ボタンを押して「hps」の近くに持っていきます。

Qsys - hps_top.qsys* (C:\WORK\sample\hps_top.qsys)

File Edit System View Tools Help

Component Library

Project

- New Component...
- Peripherals
 - SMT_Packet
- System

Library

- Bridges
- Clock and Reset
- Configuration & Programming
- DSP
- Embedded Processors
- Interface Protocols
- Memories and Memory Controllers
- Merlin Components
- Microcontroller Peripherals
- Peripherals
- PLL
- Qsys Interconnect
- SLS
- Verification
- Window Bridge

System Contents

Use	Connections	Name	Description	Exp
<input checked="" type="checkbox"/>		led_pio	PIO (Parallel I/O)	
<input checked="" type="checkbox"/>		dipsw_pio	PIO (Parallel I/O)	
<input checked="" type="checkbox"/>		button_pio	PIO (Parallel I/O)	
<input checked="" type="checkbox"/>		jtag_uart_0	JTAG UART	
<input checked="" type="checkbox"/>		nios2_qsys_0	Nios II Processor	
<input checked="" type="checkbox"/>		ram	On-Chip Memory (RAM or ROM)	
<input checked="" type="checkbox"/>		SMT_Packet_0	SMT_Packet	

Messages

Description	Path
2 Errors	
SMT_Packet_0.clock must be connected to a clock output	System.SMT_Packet_0
SMT_Packet_0.reset must be connected to a reset source	System.SMT_Packet_0
2 Warnings	
SMT_Packet_0.avalon_slave_0 must be connected to an Avalon-MM master	System.SMT_Packet_0
SMT_Packet_0.smt_output must be exported, or connected to a matching conduit.	System.SMT_Packet_0
2 Errors, 2 Warnings	

- 18) 「hps_0」の「h2f_lw_axi_master」バスに、「SMT_Packet_0」の「avalon_alave_0」を接続します。
この接続により、ツールの方でlw_axi-avalonのブリッジが自動生成されます。

clk,reset も以下の様に接続を行います。

smt_output は「Export」欄をダブルクリックし、外部信号に設定します。

Use	Connections	Name	Description	Export
<input checked="" type="checkbox"/>		clk_0	Clock Source	clk
<input checked="" type="checkbox"/>		clk_in	Clock Input	clk_0_clk_in_reset
<input checked="" type="checkbox"/>		clk_in_reset	Reset Input	Double-click to export
<input checked="" type="checkbox"/>		clk	Clock Output	Double-click to export
<input checked="" type="checkbox"/>		clk_reset	Reset Output	Double-click to export
<input checked="" type="checkbox"/>		hps_0	Hard Processor System	
<input checked="" type="checkbox"/>		h2f_reset	Reset Output	hps_0_h2f_reset
<input checked="" type="checkbox"/>		h2f_cold_reset	Reset Output	hps_0_h2f_cold_reset
<input checked="" type="checkbox"/>		h2f_warm_reset_han...	Conduit	hps_0_h2f_warm_reset_h...
<input checked="" type="checkbox"/>		h2f_user0_clock	Clock Output	hps_0_h2f_user0_clock
<input checked="" type="checkbox"/>		h2f_user1_clock	Clock Output	hps_0_h2f_user1_clock
<input checked="" type="checkbox"/>		h2f_user2_clock	Clock Output	hps_0_h2f_user2_clock
<input checked="" type="checkbox"/>		h2f_axi_clock	Clock Input	Double-click to export
<input checked="" type="checkbox"/>		h2f_axi_master	AXI Master	Double-click to export
<input checked="" type="checkbox"/>		h2f_lw_axi_clock	Clock Input	Double-click to export
<input checked="" type="checkbox"/>		h2f_lw_axi_master	AXI Master	Double-click to export
<input checked="" type="checkbox"/>		f2h_sdram0_data	AXI Slave	Double-click to export
<input checked="" type="checkbox"/>		f2h_sdram0_clock	Clock Input	Double-click to export
<input checked="" type="checkbox"/>		memory	Conduit	memory
<input checked="" type="checkbox"/>		hps_io	Conduit	hps_io
<input checked="" type="checkbox"/>		SMT_Packet_0	SMT_Packet	
<input checked="" type="checkbox"/>		avalon_slave_0	Avalon Memory Mapped Slave	Double-click to export
<input checked="" type="checkbox"/>		reset	Reset Input	Double-click to export
<input checked="" type="checkbox"/>		clock	Clock Input	Double-click to export
<input checked="" type="checkbox"/>		smt_output	Conduit	Double-click to export
<input checked="" type="checkbox"/>		onchip_memory2_0	On-Chip Memory (RAM or ROM)	
<input checked="" type="checkbox"/>		clk1	Clock Input	SMT_Packet_0.smt_output
<input checked="" type="checkbox"/>		s1	Avalon Memory	Conduit [conduit_end 12.1]
<input checked="" type="checkbox"/>		reset1	Reset Input	Double-click to export

名称は、デフォルト (SMT_Packet_0_smt_output) としました。

Use	Connections	Name	Description	Export
<input checked="" type="checkbox"/>		SMT_Packet_0	SMT_Packet	
<input checked="" type="checkbox"/>		avalon_slave_0	Avalon Memory Mapped Slave	Double-click to export
<input checked="" type="checkbox"/>		reset	Reset Input	Double-click to export
<input checked="" type="checkbox"/>		clock	Clock Input	Double-click to export
<input checked="" type="checkbox"/>		smt_output	Conduit	SMT Packet 0 smt output

- 19) 「Address」タブで、パケット生成論理のアドレスを決めます。ここでは、オフセットアドレスを 0x30000 番地としています。パケット生成論理は lw_axi バスに接続していますので、今回のケースでは、パケット生成論理の物理アドレスは 0xFF230000 となります。つまり、SMT のドライバがアクセスするアドレスが、0xFF230000 番地となります。

System Contents	Address Map	Clock Settings	Project Settings	Instance Parameters	System Inspector	HDL Example	Generation
	hps_0.h2f_axi_master		hps_0.h2f_lw_axi_master				
hps_0.f2h_sdr0_data							
led_pio.s1						0x0001_0040 - 0x0001_005f	
dipsw_pio.s1						0x0001_0080 - 0x0001_009f	
button_pio.s1						0x0001_00c0 - 0x0001_00df	
jtag_uart_0.avalon_jtag_slave							
nios2_qsys_0.jtag_debug_module							
ram.s1							
onchip_memory2_0.s1	0x0000_0000 - 0x0000_ffff						
SMT_Packet_0.avalon_slave_0						0x0003_0000 - 0x0003_001f	

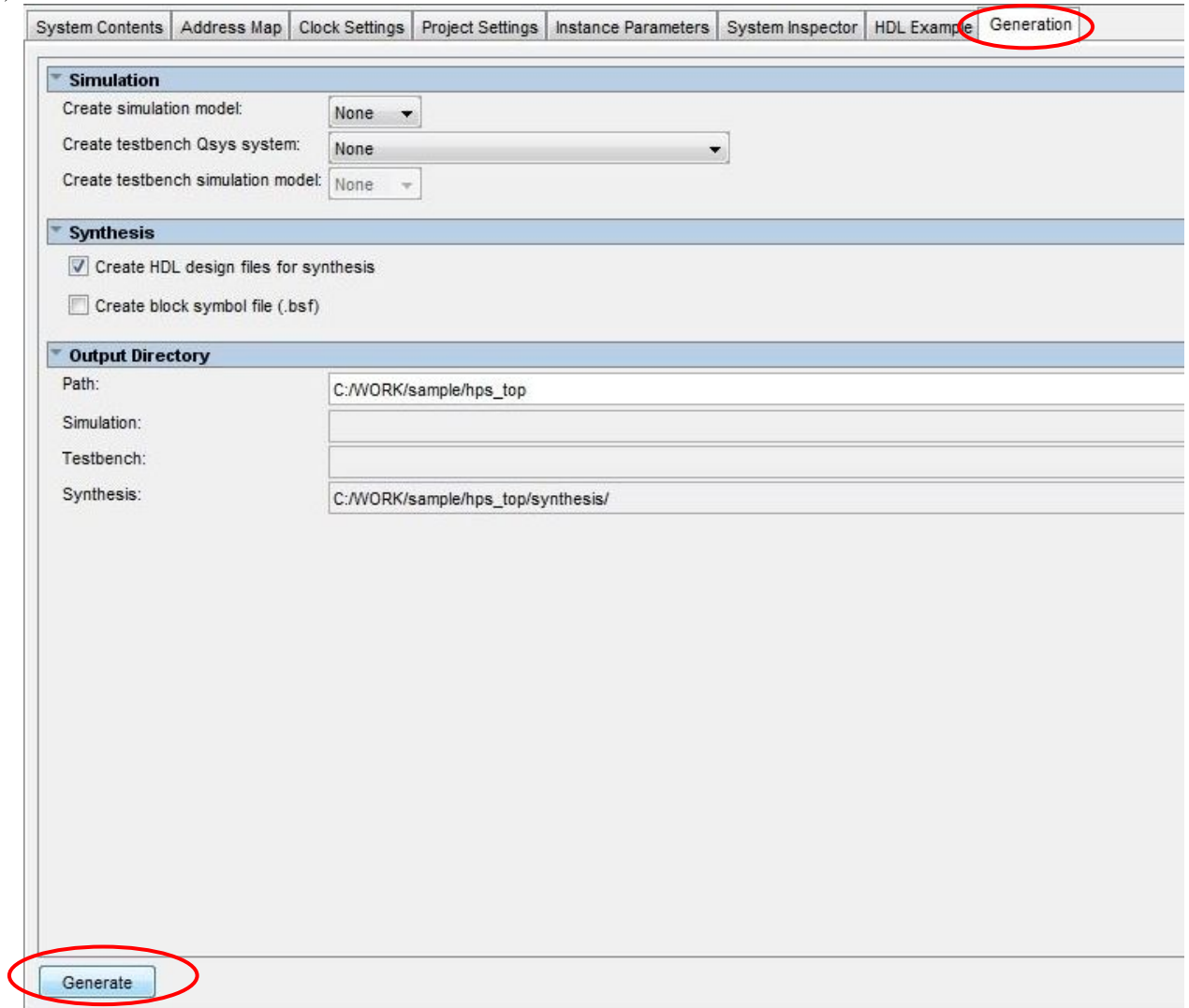
- 20) 「clock settings」タブでは、パケット論理に接続されているクロックが 100MHz を超えていないことを確認します。

System Contents	Address Map	Clock Settings	Project Settings	Instance Parameters	System Inspector	HDL Example	Generation
Clock Settings							
Name	Source	MHz					
clk_0	External	100.0					
hps_0.h2f_user0_clock	hps_0.h2f_user0_clock	100.0					
hps_0.h2f_user1_clock	hps_0.h2f_user1_clock	100.0					
hps_0.h2f_user2_clock	hps_0.h2f_user2_clock	100.0					

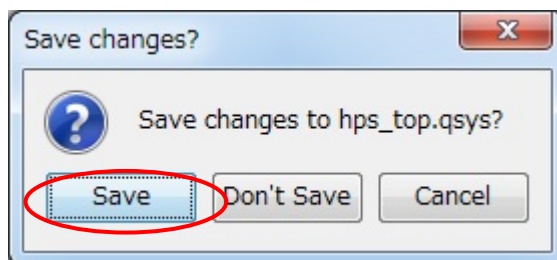
- 21) 「HDL Example」タブでは、パケット論理の出力信号がアサインされているか確認します。

System Contents	Address Map	Clock Settings	Project Settings	Instance Parameters	System Inspector	HDL Example	Generation
You can copy the example HDL below to declare an instance of your Qsys system.							
HDL Language: Verilog							
Example HDL							
.hps_io_hps_io_tzcu_inst_scl	((connected-to-hps_io_hps_io_tzcu_inst_scl)),					//	
.hps_io_hps_io_trace_inst_clk	((connected-to-hps_io_hps_io_trace_inst_clk)),					//	
.hps_io_hps_io_trace_inst_D0	((connected-to-hps_io_hps_io_trace_inst_D0)),					//	
.hps_io_hps_io_trace_inst_D1	((connected-to-hps_io_hps_io_trace_inst_D1)),					//	
.hps_io_hps_io_trace_inst_D2	((connected-to-hps_io_hps_io_trace_inst_D2)),					//	
.hps_io_hps_io_trace_inst_D3	((connected-to-hps_io_hps_io_trace_inst_D3)),					//	
.hps_io_hps_io_trace_inst_D4	((connected-to-hps_io_hps_io_trace_inst_D4)),					//	
.hps_io_hps_io_trace_inst_D5	((connected-to-hps_io_hps_io_trace_inst_D5)),					//	
.hps_io_hps_io_trace_inst_D6	((connected-to-hps_io_hps_io_trace_inst_D6)),					//	
.hps_io_hps_io_trace_inst_D7	((connected-to-hps_io_hps_io_trace_inst_D7)),					//	
.hps_io_hps_io_gpio_inst_GPI000	((connected-to-hps_io_hps_io_gpio_inst_GPI000)),					//	
.hps_io_hps_io_gpio_inst_GPI009	((connected-to-hps_io_hps_io_gpio_inst_GPI009)),					//	
.hps_io_hps_io_gpio_inst_GPI035	((connected-to-hps_io_hps_io_gpio_inst_GPI035)),					//	
.hps_0.h2f_reset_reset_n	((connected-to-hps_0.h2f_reset_reset_n)),					//	hps_0.h2f
.clk_0_clk_in_reset_reset_n	((connected-to-clk_0_clk_in_reset_reset_n)),					//	clk_0_clk_i
.hps_0.h2f_cold_reset_reset_n	((connected-to-hps_0.h2f_cold_reset_reset_n)),					//	hps_0.h2f_col
.hps_0.h2f_warm_reset_handshake_h2f_pending_rst_req_n	((connected-to-hps_0.h2f_warm_reset_handshake_h2f_pending_rst_req_n)),					//	hps_0.h2f_warm_reset_ha
.hps_0.h2f_warm_reset_handshake_f2h_pending_rst_ack_n	((connected-to-hps_0.h2f_warm_reset_handshake_f2h_pending_rst_ack_n)),					//	
.hps_0.h2f_user1_clock_clk	((connected-to-hps_0.h2f_user1_clock_clk)),					//	hps_0.h2f_user
.hps_0.h2f_user2_clock_clk	((connected-to-hps_0.h2f_user2_clock_clk)),					//	hps_0.h2f_user
.hps_0.h2f_user0_clock_clk	((connected-to-hps_0.h2f_user0_clock_clk)),					//	hps_0.h2f_user
.led_pio_external_connection_export	((connected-to-led_pio_external_connection_export)),					//	led_pio_external_con
.dipsw_pio_external_connection_export	((connected-to-dipsw_pio_external_connection_export)),					//	dipsw_pio_external_con
.button_pio_external_connection_export	((connected-to-button_pio_external_connection_export)),					//	button_pio_external_con
.smt_packet_0_smt_output_clk	((connected-to-smt_packet_0_smt_output_clk)),					//	smt_packet_0_smt
.smt_packet_0_smt_output_xsync	((connected-to-smt_packet_0_smt_output_xsync)),					//	
.smt_packet_0_smt_output_d	((connected-to-smt_packet_0_smt_output_d)),					//	
);							

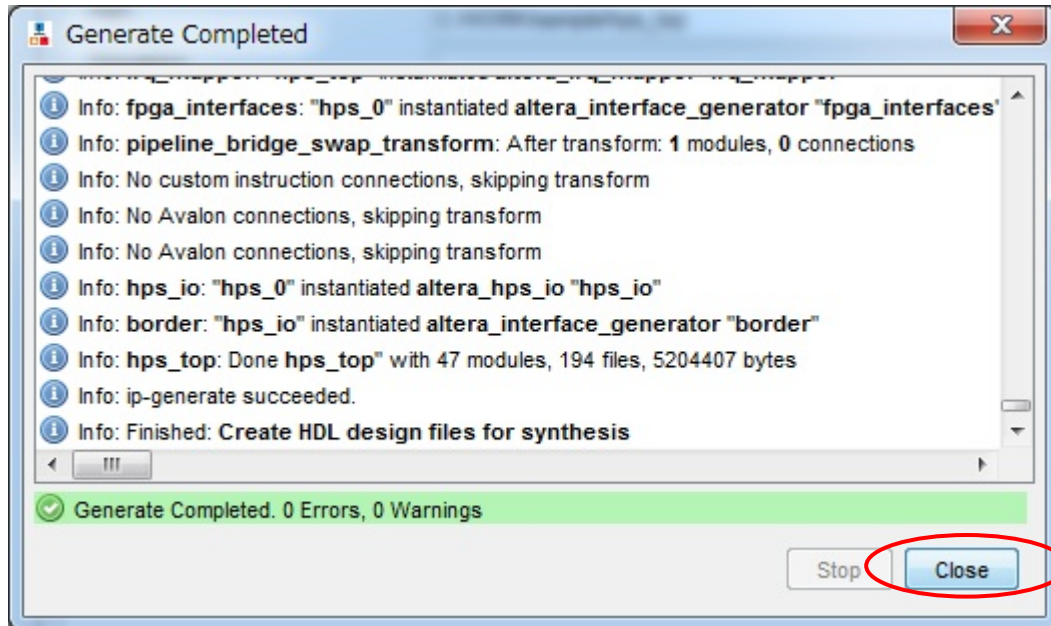
- 22) 「Generation」 タブで、「Generate」をクリックしてコンパイルを開始します。



- 23) 以下のダイアログが出てきますので、「Save」を押します。

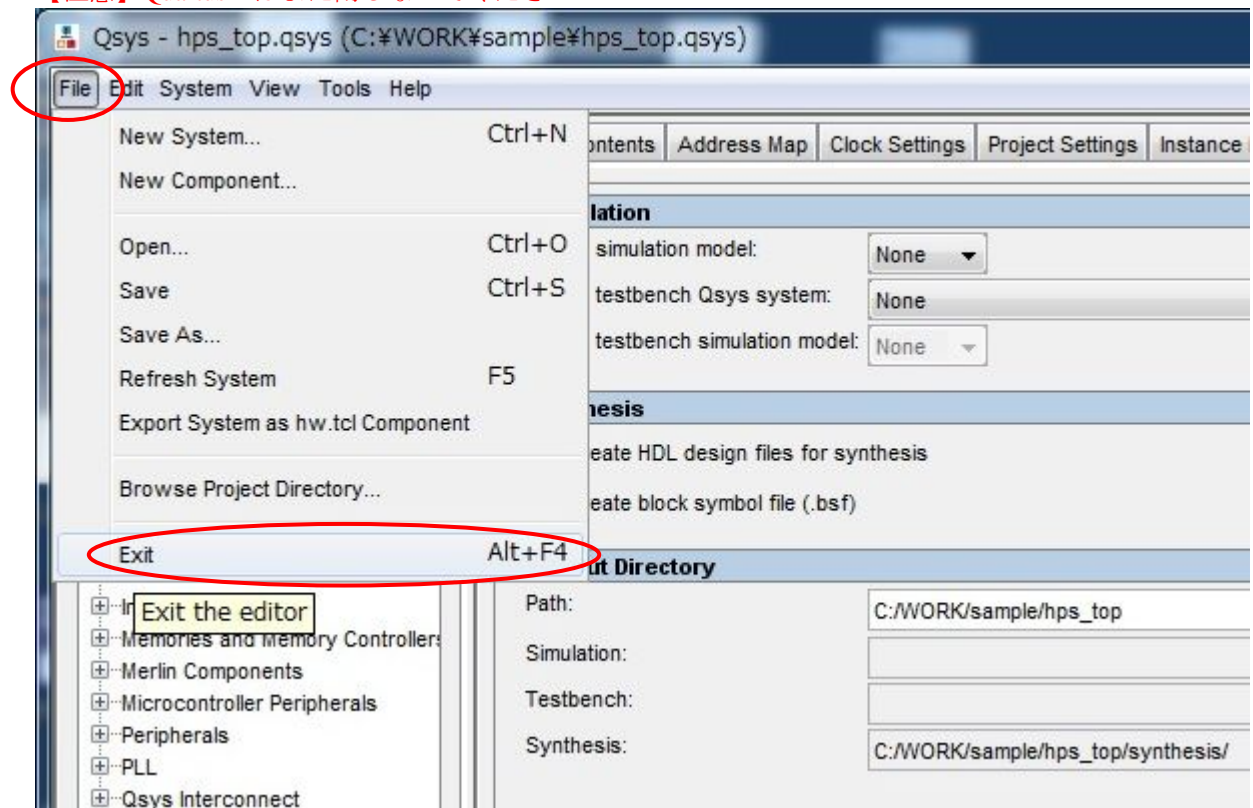


24) コンパイルが完了すると、以下のダイアログの内容となります。「Close」をクリックします。



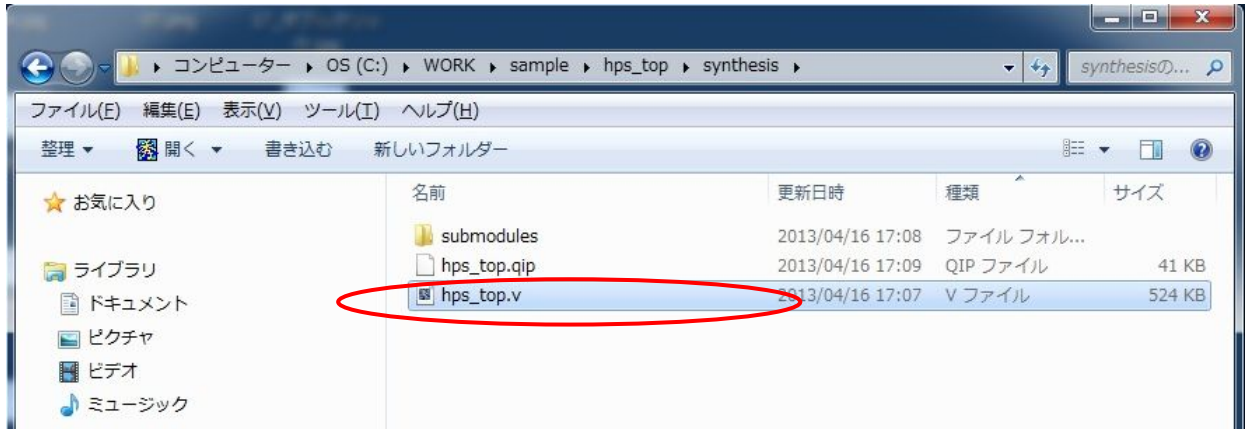
25) これで Qsys 側の作業は終了となります。「File」→「Exit」で、Qsys を閉じます。

【注意】 Quartus II はまだ閉じないでください

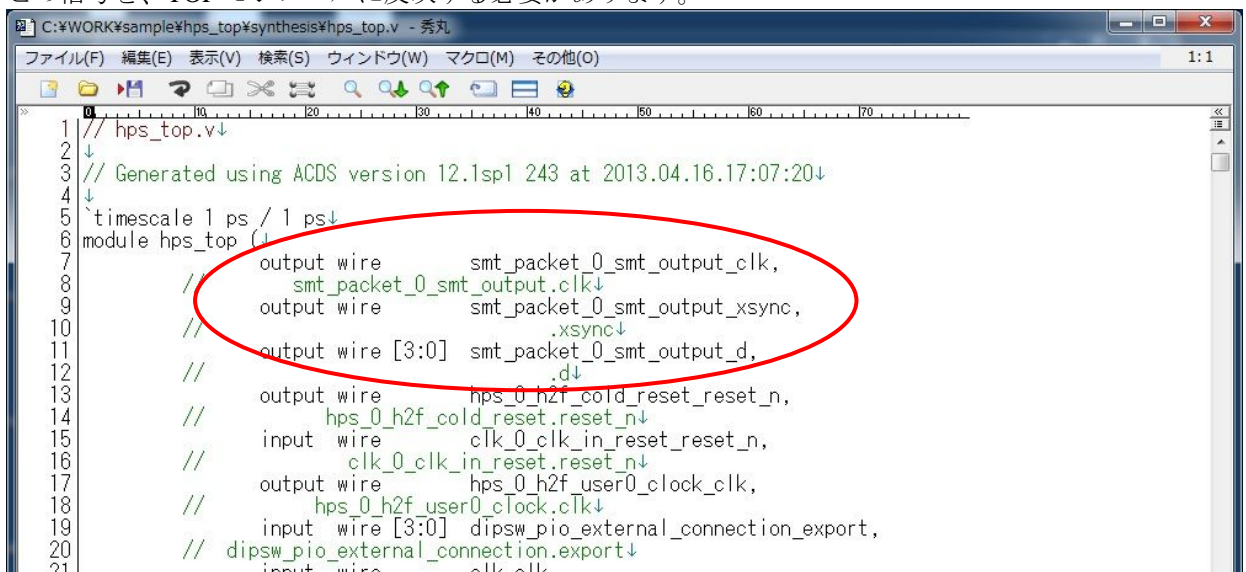


2.3 TOP モジュールの編集

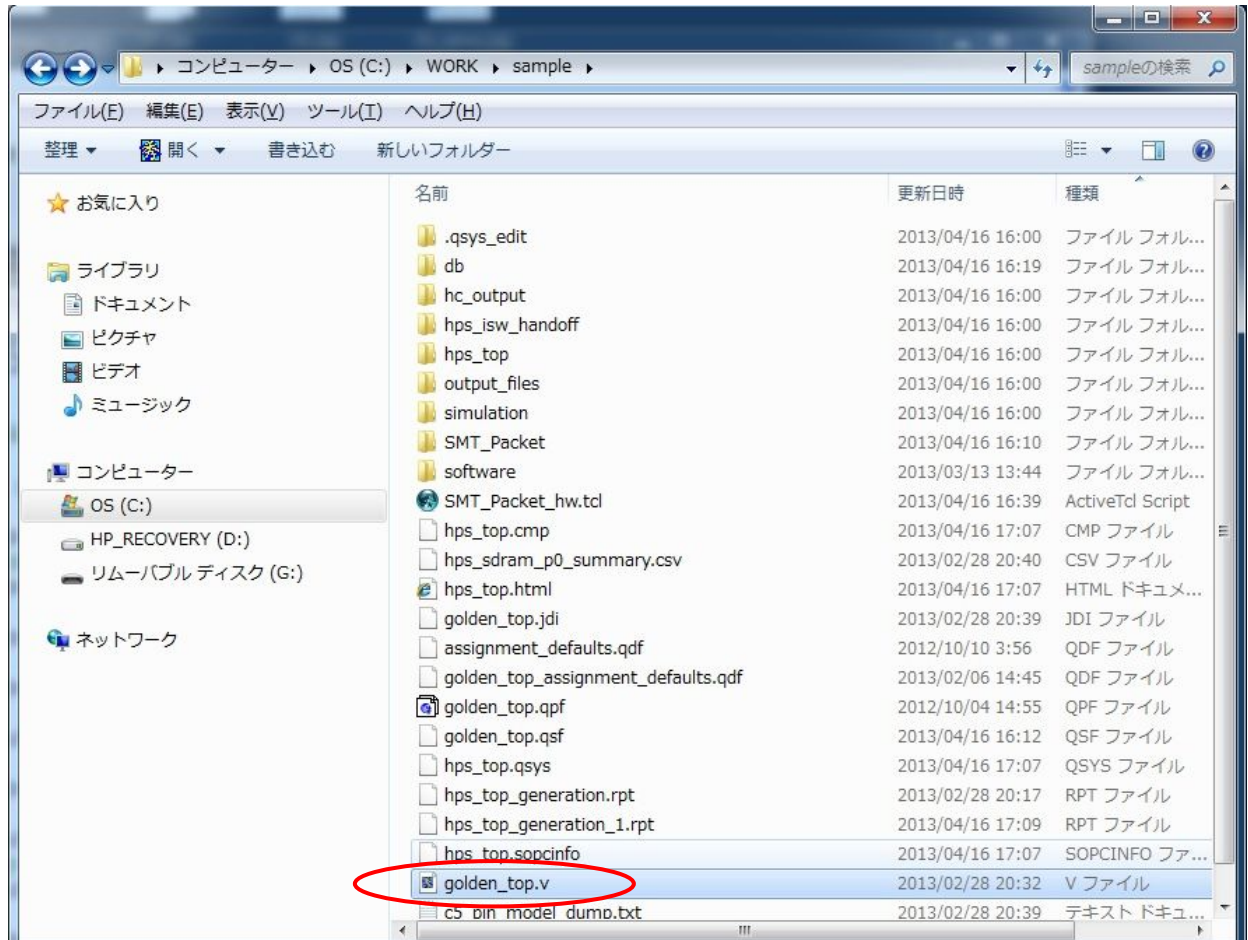
- 1) “ワークフォルダ¥hps_top¥synthesis¥hps_top.v”を開き、SMT パケット生成論理の出力信号を確認します。



- 2) パケット生成論理の出力信号が反映されていることを確認します。
この信号を、TOP モジュールに反映する必要があります。



- 3) 本プロジェクトの TOP モジュールを開きます。



- 4) hps_top.v の SMT パケット論理の出力信号を、TOP モジュールに接続していきます。
下記の様な変更（赤字の追記）を行います。

※入出力定義の箇所（先頭）

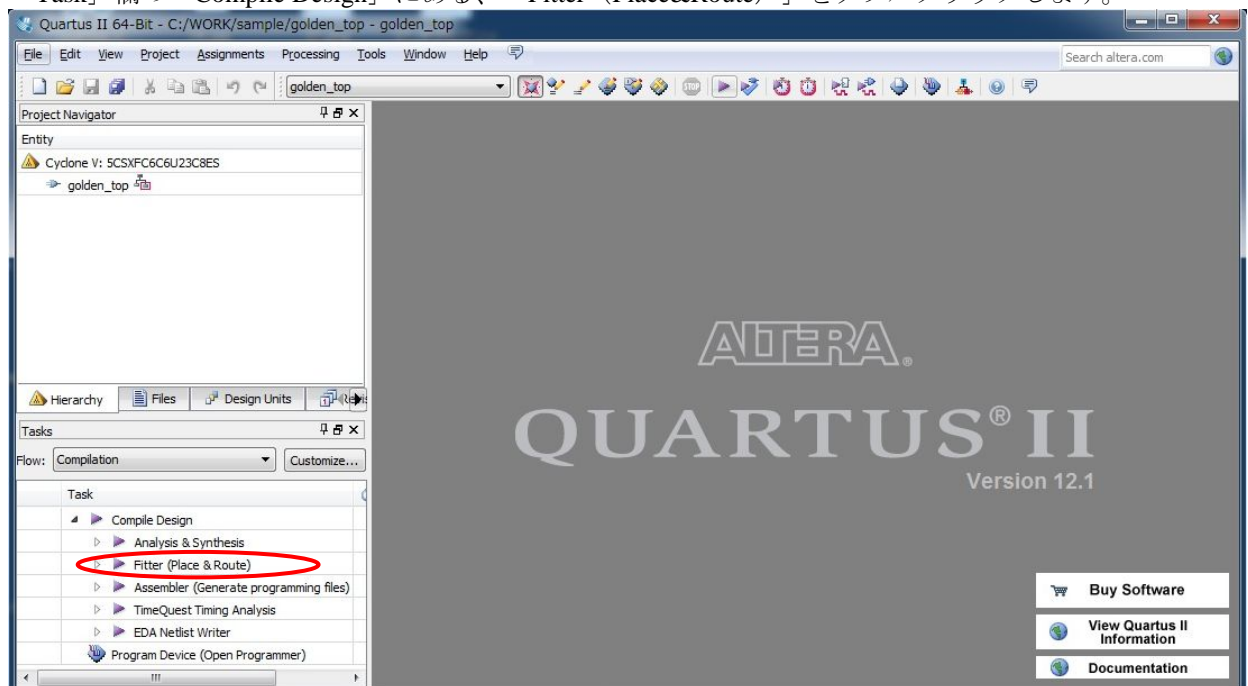
```
module golden_top (
    input      uart_rx,           //3.3V LV //UART Receive
    output     uart_tx,           //3.3V //UART Transmit
    inout      conv_hps_usb_n,    //3.3V //Placed on HPS GPIO
    inout      i2c_scl_hps,       //3.3V //HPS I2C Clock output
    inout      i2c_sda_hps,       //3.3V //HPS I2C Data Input/Output
    :
    // @YDC add begin
    // SMT Signals-----
    output     smt_packet_0_smt_output_clk, // clk
    output     smt_packet_0_smt_output_xsync, // xsync
    output [3:0] smt_packet_0_smt_output_d, // d
    // @YDC add end
    output     overtemp,         //2.5V
    output [2:0] test_pad        //2.5V
);
```

※hps_top モジュールへの接続の箇所

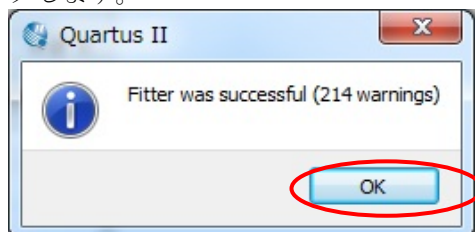
```
hps_top u0 (  
  //@YDC add begin  
  .smt_packet_0_smt_output_clk(smt_packet_0_smt_output_clk),      // clk  
  .smt_packet_0_smt_output_xsync      (smt_packet_0_smt_output_xsync),      //xsync  
  .smt_packet_0_smt_output_d  (smt_packet_0_smt_output_d),// d  
  //@YDC add end  
  .clk_clk                        (clk_50m_fpga),      // clk.clk  
  .clk_0_clk_in_reset_reset_n    (cpu_resetsn),      // clk_0_clk_in_reset.reset_n  
  :  
  :  
  .hps_0_h2f_user0_clock_clk      (h2f_user0_clock),  
  .hps_0_h2f_user1_clock_clk      (h2f_user1_clock),  
  .hps_0_h2f_user2_clock_clk      (h2f_user2_clock)  
);
```


2.4 Quartus II コンパイル手順

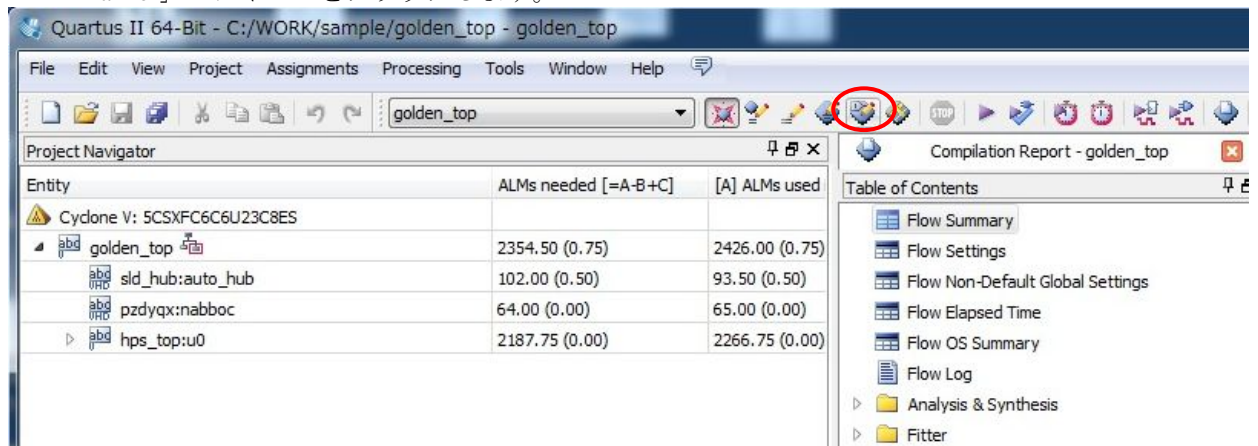
- 1) Quartus IIにて、「Fitter (Place&Route)」まで実行します。
「Task」欄の「Compile Design」にある、「Fitter (Place&Route)」をダブルクリックします。



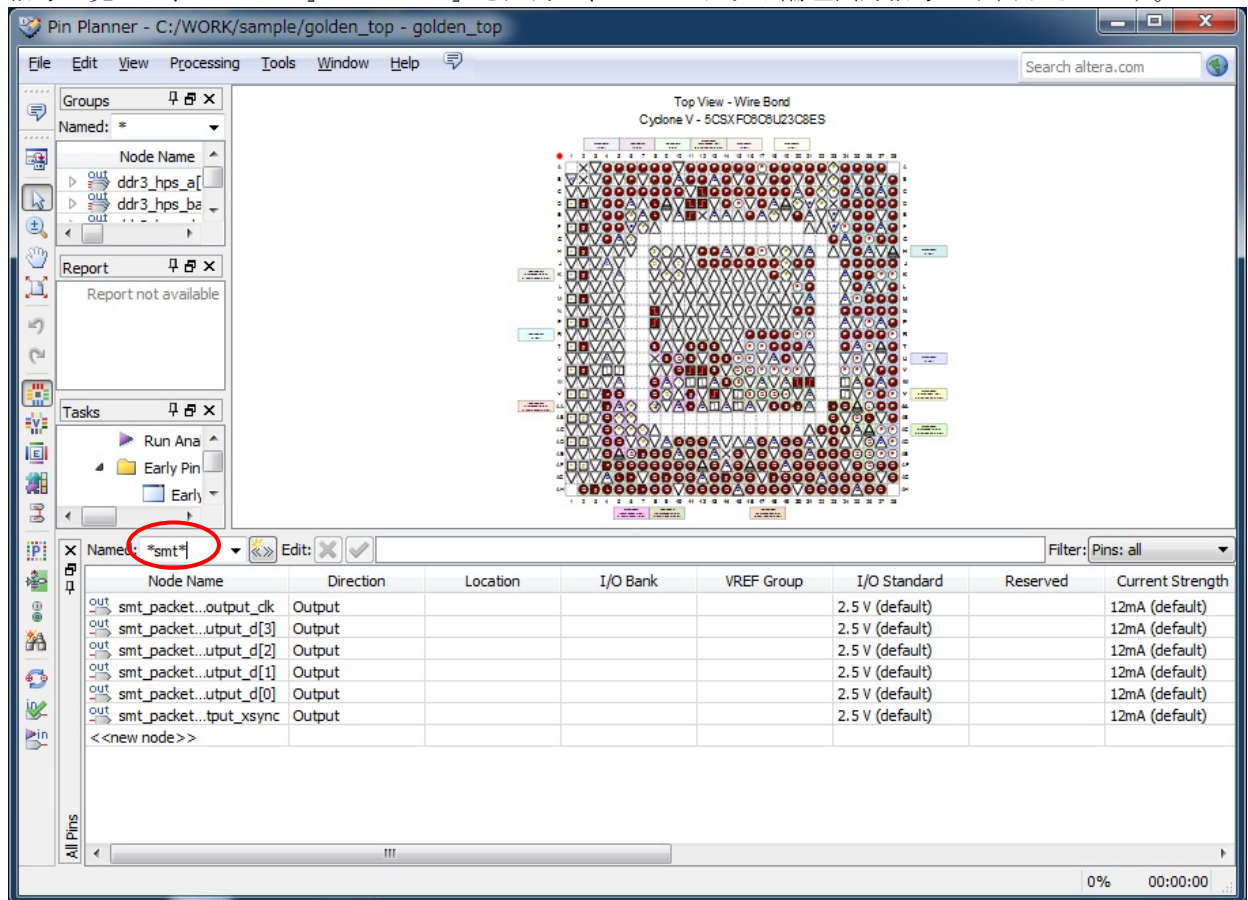
- 2) 「Fitter (Place&Route)」まで無事に完了すると以下のダイアログが出てくるので、「OK」をクリックします。



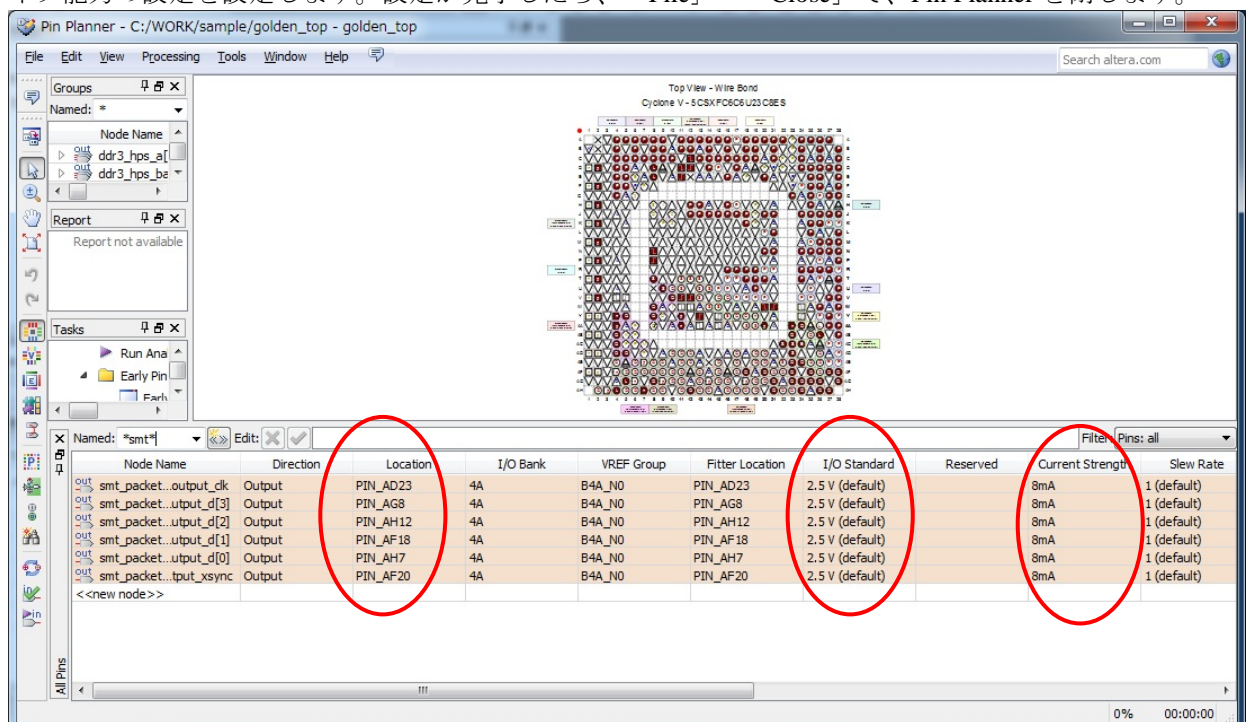
- 3) 「Pin Planner」のアイコンをクリックします。



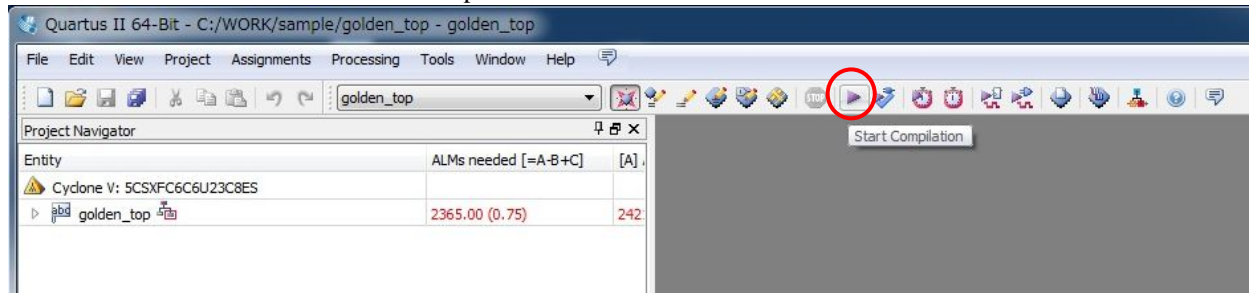
- 4) 信号一覧にて、「Named:」で「*smt*」を入力し、SMT パケット論理出力信号のみ表示させます。



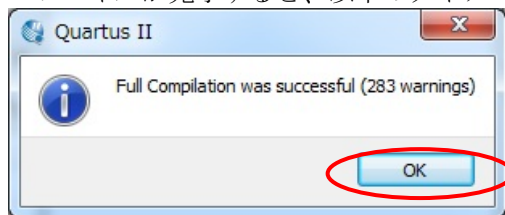
- 5) 「Location」で pin アサイン, 「I/O Standard」で信号の電圧レベル, 「Current Strength」で信号のドライブ能力の設定を設定します。設定が完了したら、「File」→「Close」で、Pin Planner を閉じます。



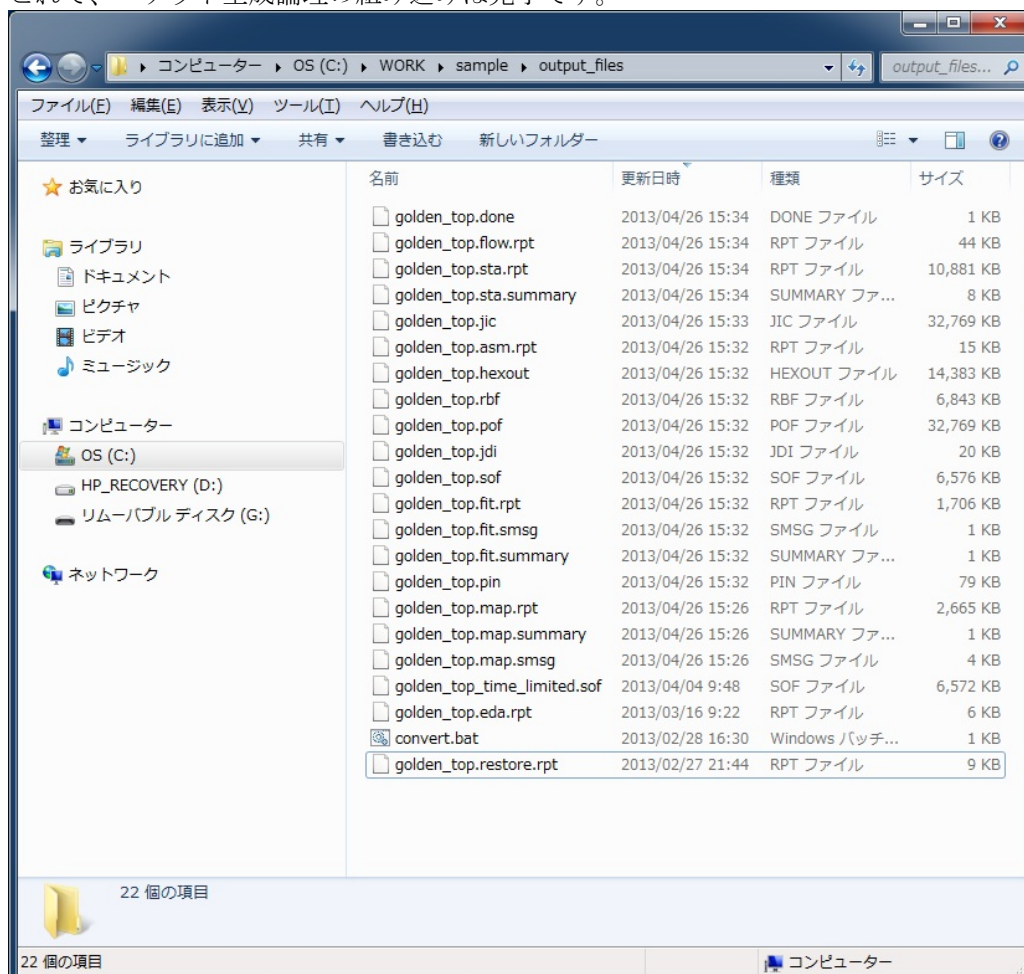
- 6) コンパイルを実施します。「Start Compilation」ボタンをクリックし、コンパイルを開始します。



- 7) コンパイルが完了すると、以下のダイアログが出てくるので「OK」をクリックします。



- 8) “” ワークフォルダ¥hps_top¥synthesis¥hps_top.v
sof ファイル、Pof ファイルが無事に生成されていることを確認します。
これで、パケット生成論理の組み込みは完了です。



以上。