Zynq-7000EPP
パケット生成論理 導入手順書

株式会社ＤＴＳインサイト

改訂履歴

版	発行日付	変更内容
第 1 版	2013.06.28	新規発行
第 2 版	2013.12.25	TRQerS 対応

目次

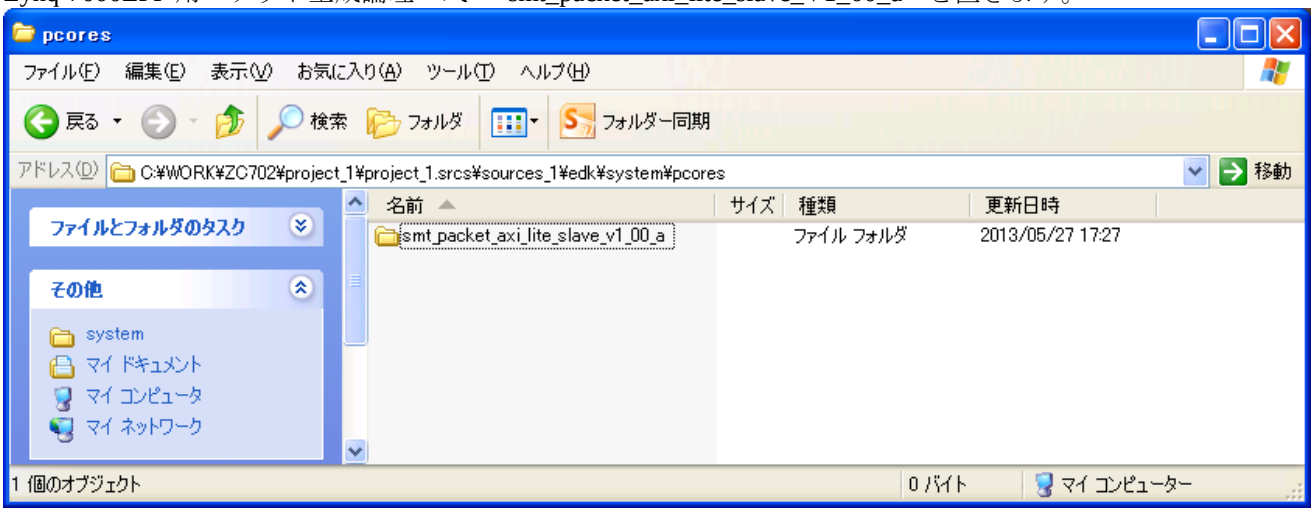
1 はじめに

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、Zynq-7000EPP への導入手順を説明します。
また、本マニュアルでは、Xilinx 社純正の Zynq-7000EPP 評価ボード”ZC702”を例にしています。

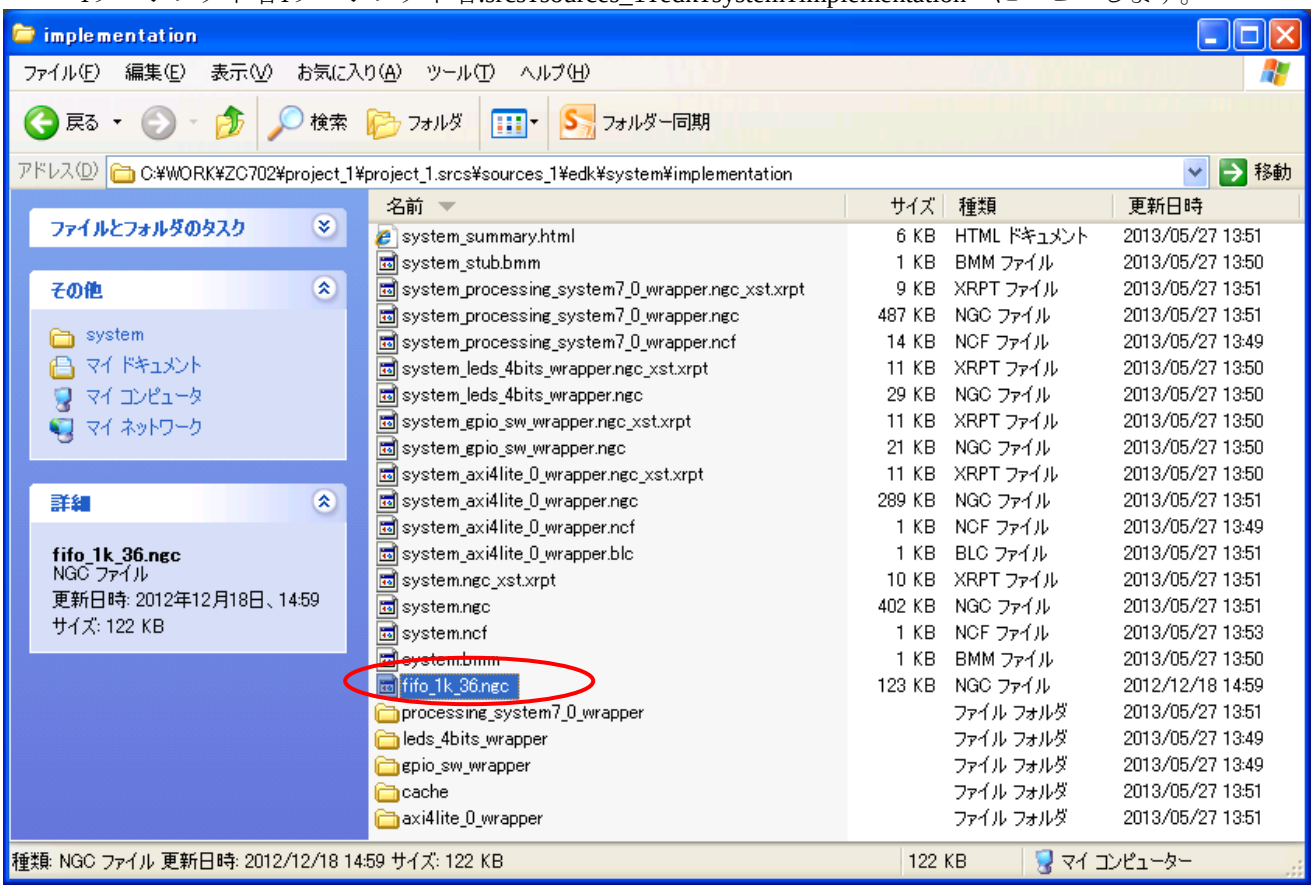
2 パケット生成論理組み込み手順

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

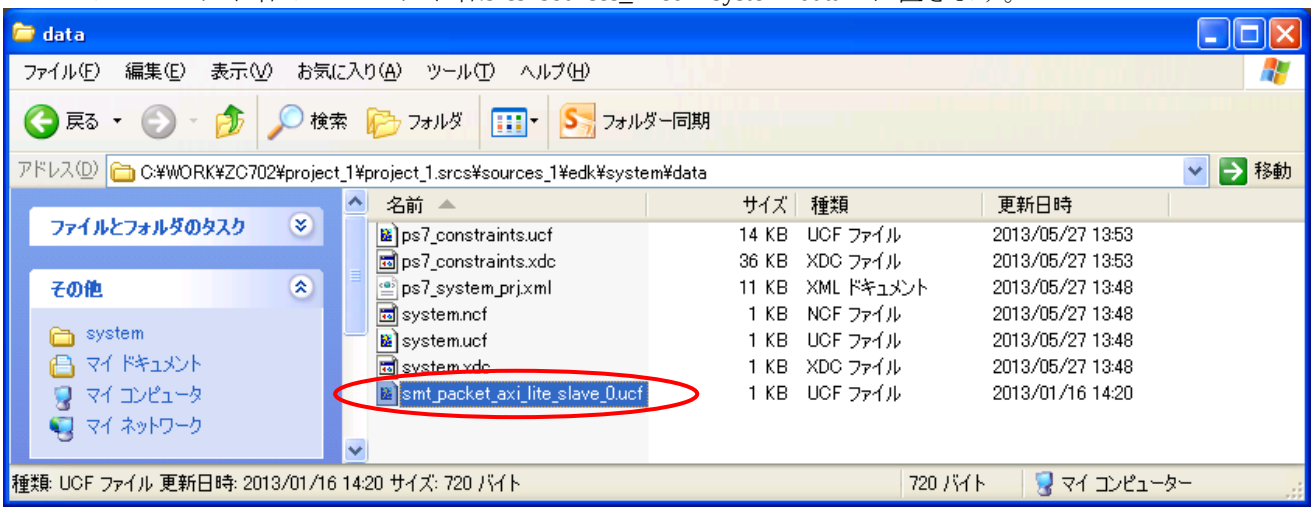
1) 既存プロジェクトの“¥プロジェクト名¥プロジェクト名.srcs¥sources_1¥edk¥system¥pcores”フォルダ下に、Zynq-7000EPP 用パケット生成論理一式 “smt_packet_axi_lite_slave_V1_00_a”を置きます。



2) パケット生成論理 “smt_packet_axi_lite_slave_V1_00_a¥netlist”の FIFO ネットリスト“fifo_1k_36.ngc”を“¥プロジェクト名¥プロジェクト名.srcs¥sources_1¥edk¥system¥implementation”にコピーします。



3) 参考 UCF ファイル“smt_packet_axi_lite_slave_0.ucf”を、お客様の環境に合わせて pin アサイン等を変更し、“¥プロジェクト名¥プロジェクト名.srcs¥sources_1¥edk¥system¥data”に置きます。



4) “¥プロジェクト名¥プロジェクト名.srcs¥sources_1¥edk¥system¥pcores¥smt_packet_axi_lite_slave_v1_00_a¥hdl¥verilog”の“smt_packet_axi_lite_slave.v”71 行目（下記の赤字部分）を編集します。
パケット論理を配置するアドレスが parameter で宣言していますので、実際にパケット論理を配置するアドレスに書き換えます。ここでは、0x40000000～0x40000FFF としています※。

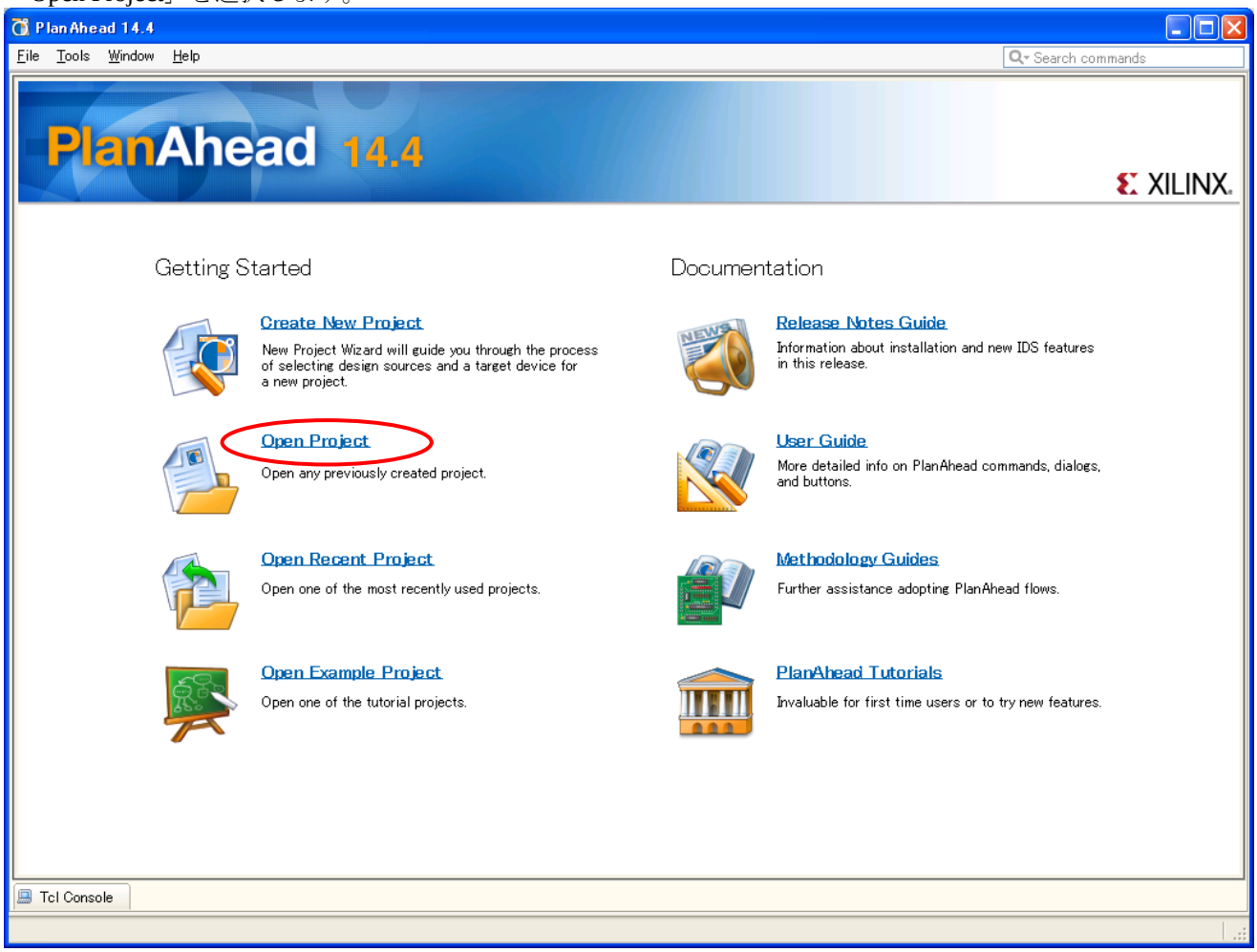
```
/*****************************************************************************/
/* Parameter                                                                 */
/*****************************************************************************/
 parameter      PACKET_IF_ADR =          20'h40000;          //Packet I/F Address = 0x400000000 to 0x40000FFF

 parameter      RESP_OKAY =     2'b00;
 parameter      RESP_EXOKAY =   2'b01;
 parameter      RESP_SLVERR =   2'b10;
 parameter      RESP_DECERR =   2'b11;
```

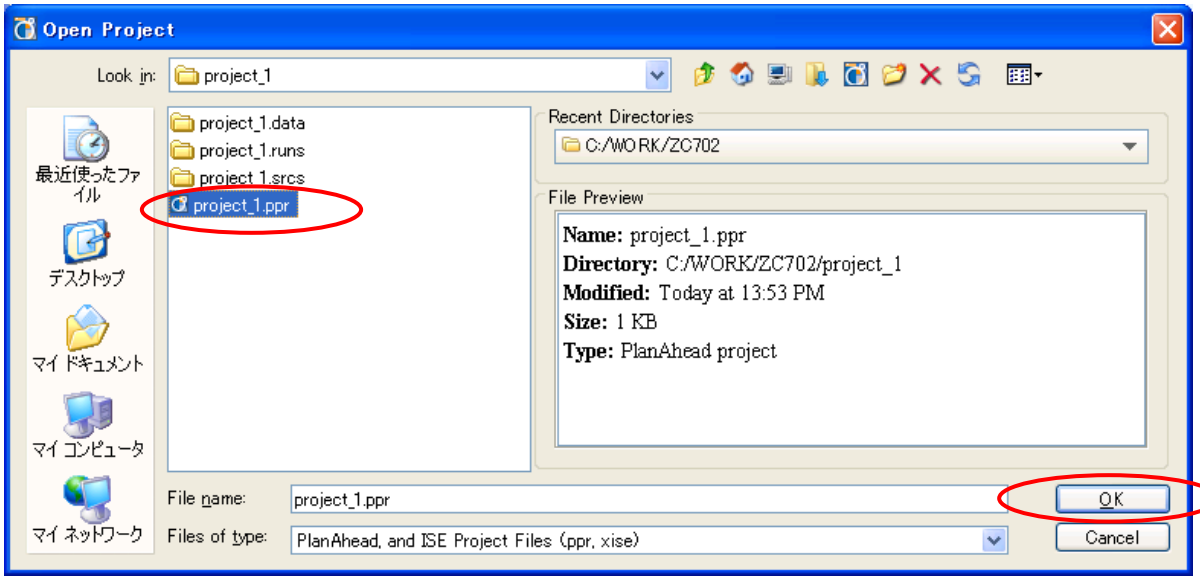
※Zynq-7000EPP の汎用 PL インターフェースとして、M_AXI_GP0，M_AXI_GP1 が存在します。
それぞれのアドレス範囲は、以下となります。
M_AXI_GP0：0x40000000～0x7FFFFFFF, M_AXI_GP1：0x80000000～0xBFFFFFFF
本マニュアルでは、M_AXI_GP0 の空き領域 0x40000000～0x40000FFF を設定しました。

2.2 「Xilinx Platform studio」でのパケット生成論理組み込み手順

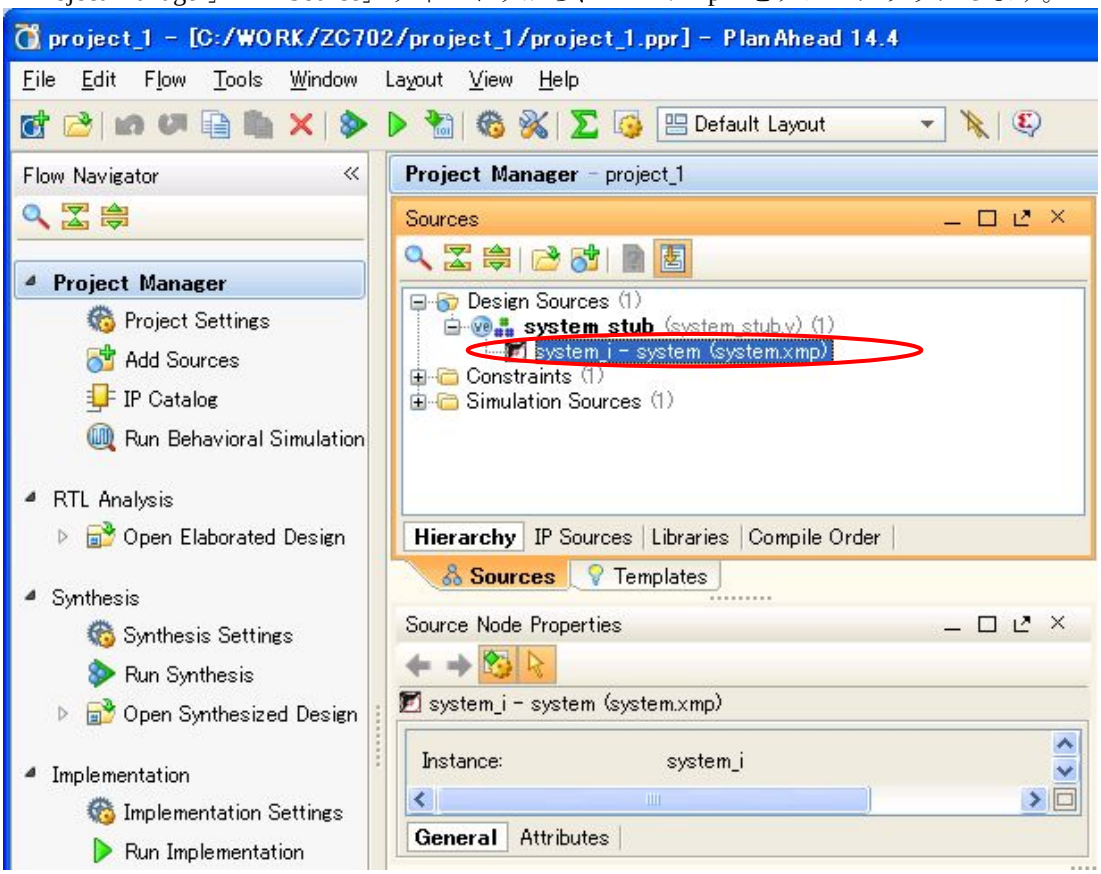
1) 「Plan Ahead」を立ち上げ、既存プロジェクトを開きます。
「Open Project」を選択します。



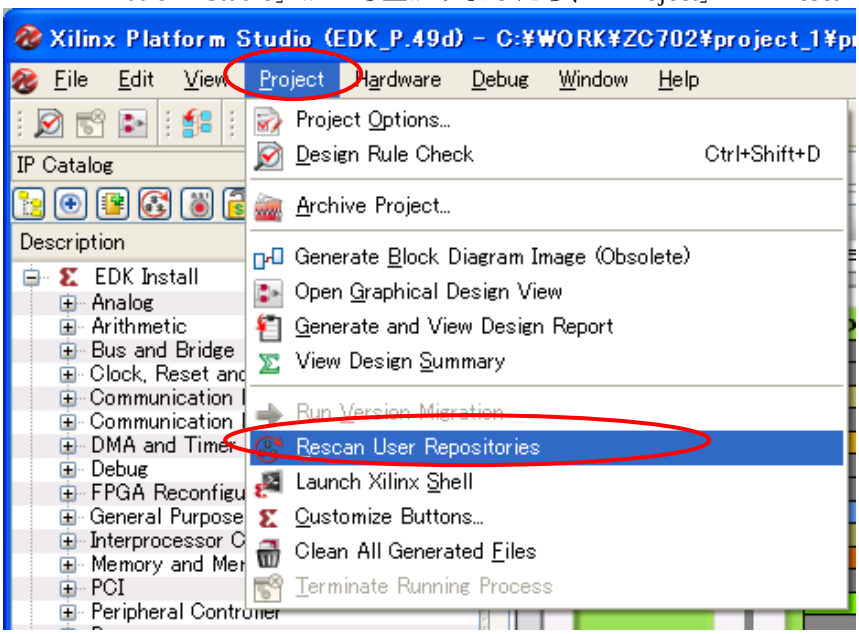
2) パケット生成論理を組み込む対象となる既存プロジェクトを選択し、「OK」をクリックします。



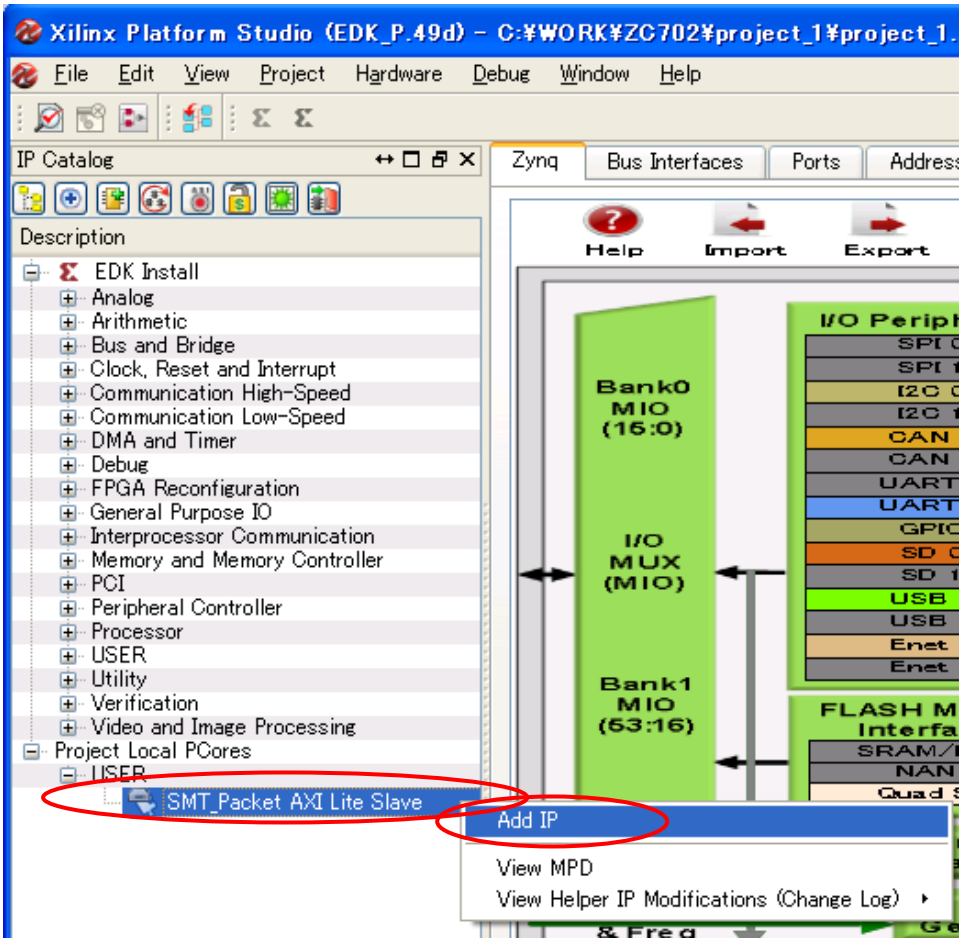
3) Plan Ahead が立ち上がりましたら、「Xilinx Platform Studio」を立ち上げます。
「Project Manager」の「Source」ウィンドウから、“xxx.xmp”をダブルクリックします。



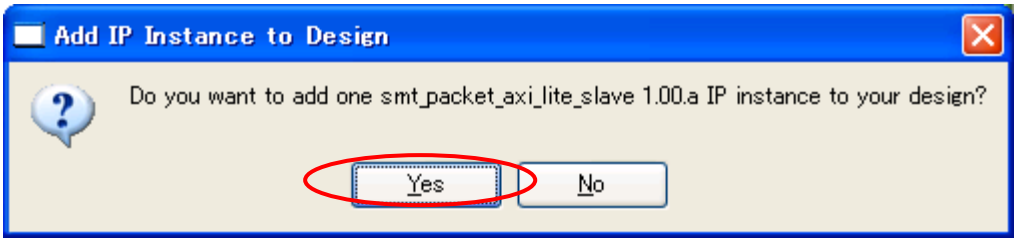
5) 「Xilinx Platform Studio」が立ち上がりましたら、「Project」→「Rescan User Repositories」を選択します。



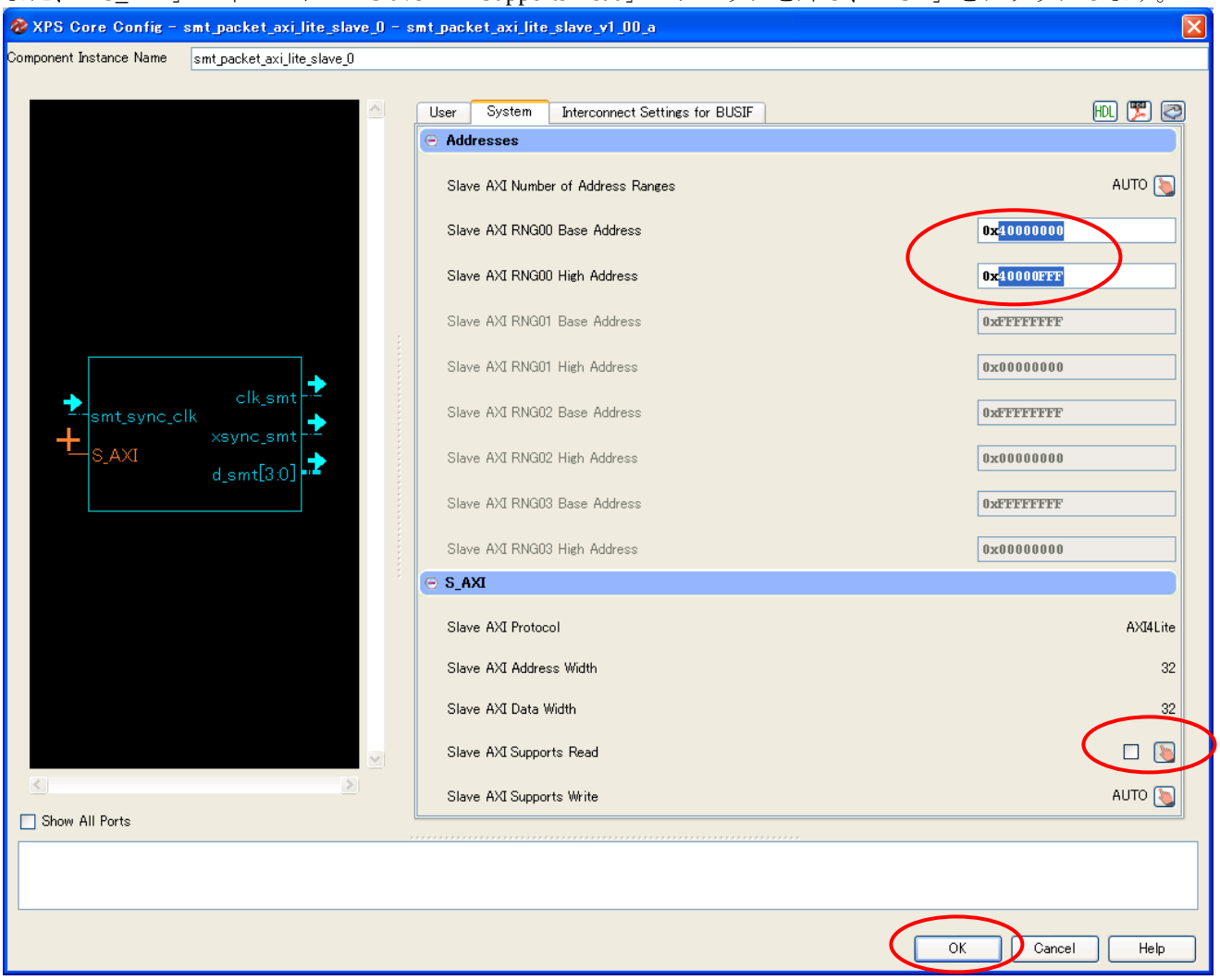
6) 「IP Catalog」の「Project Local PCores」→「USER」に
パケット生成論理の IP「SMT_Packet_AXI_Lite_Slave」が登録されます。
「SMT_Packet_AXI_Lite_Slave」を右クリック→「Add IP」を選択し、Zynq-7000EPP にパケット生成論理を追加します。



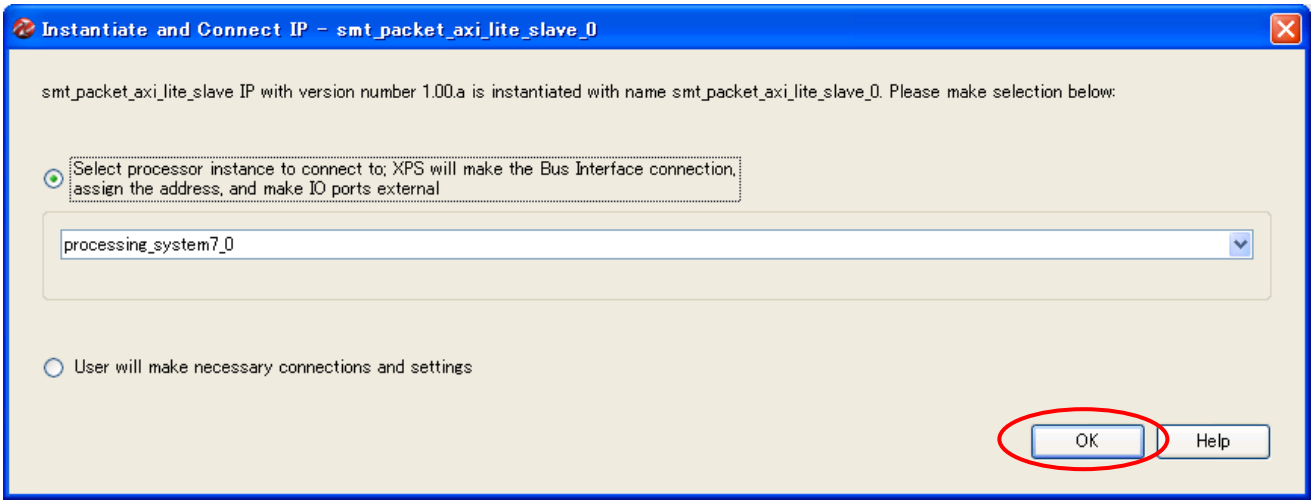
7) 以下のダイアログが出現しますので、「Yes」をクリックします。



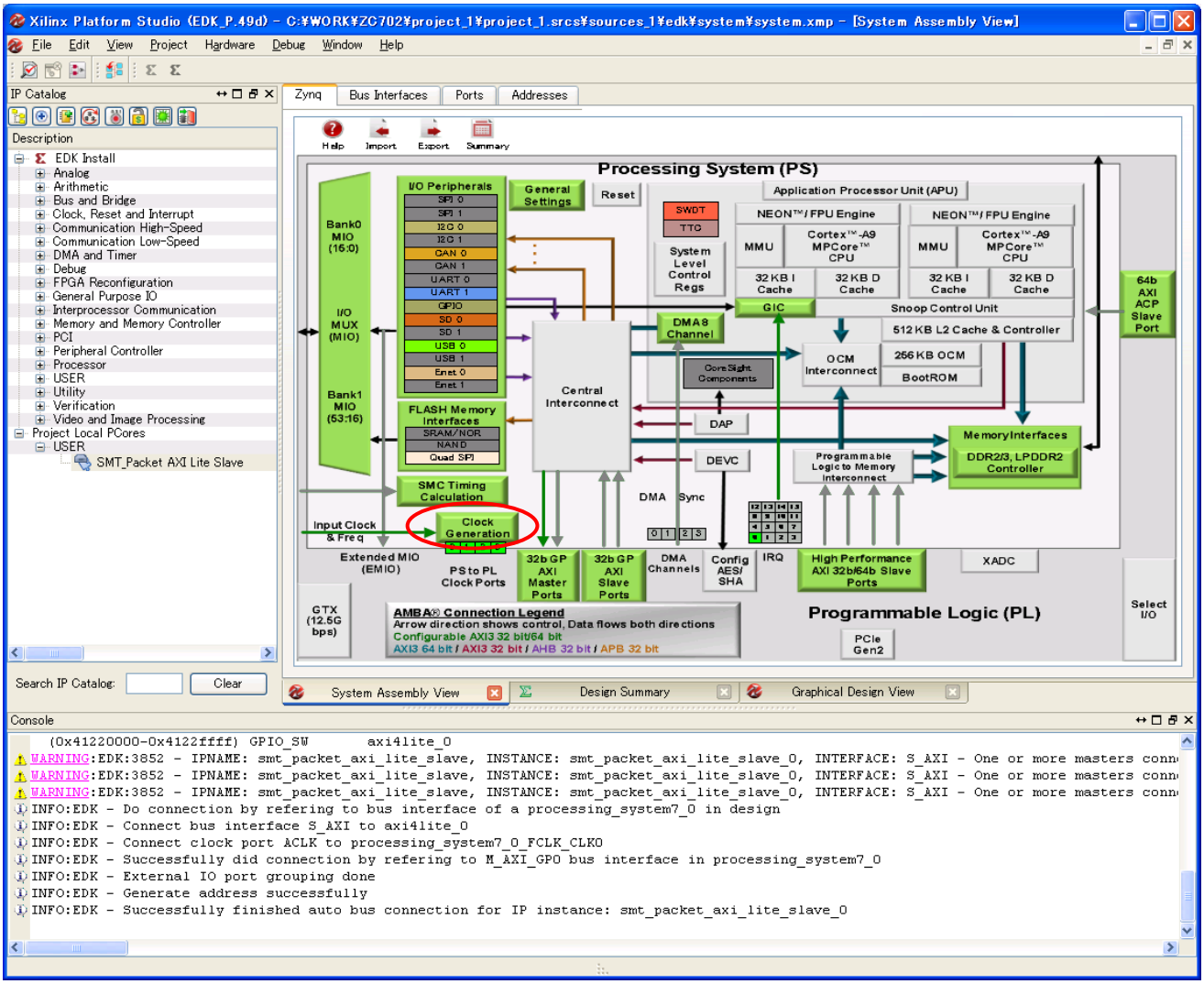
8) 以下のウィンドウが出現しますので、「System」タブを開きます。
「Address」フィールドにパケット生成論理のアドレス範囲を設定します。アドレス範囲は 4kB 単位での設定となります。ここでは、0x40000000～0x40000FFF としています。
「Slave AXI RNG00 Base Address」に先頭アドレス 0x40000000　を、「Slave AXI RNG00 Base Address」に終了アドレス 0x40000FFF を設定します。
また、「S_AXI」フィールドの「Slave AXI Supports Read」のチェックを外し、「OK」をクリックします。



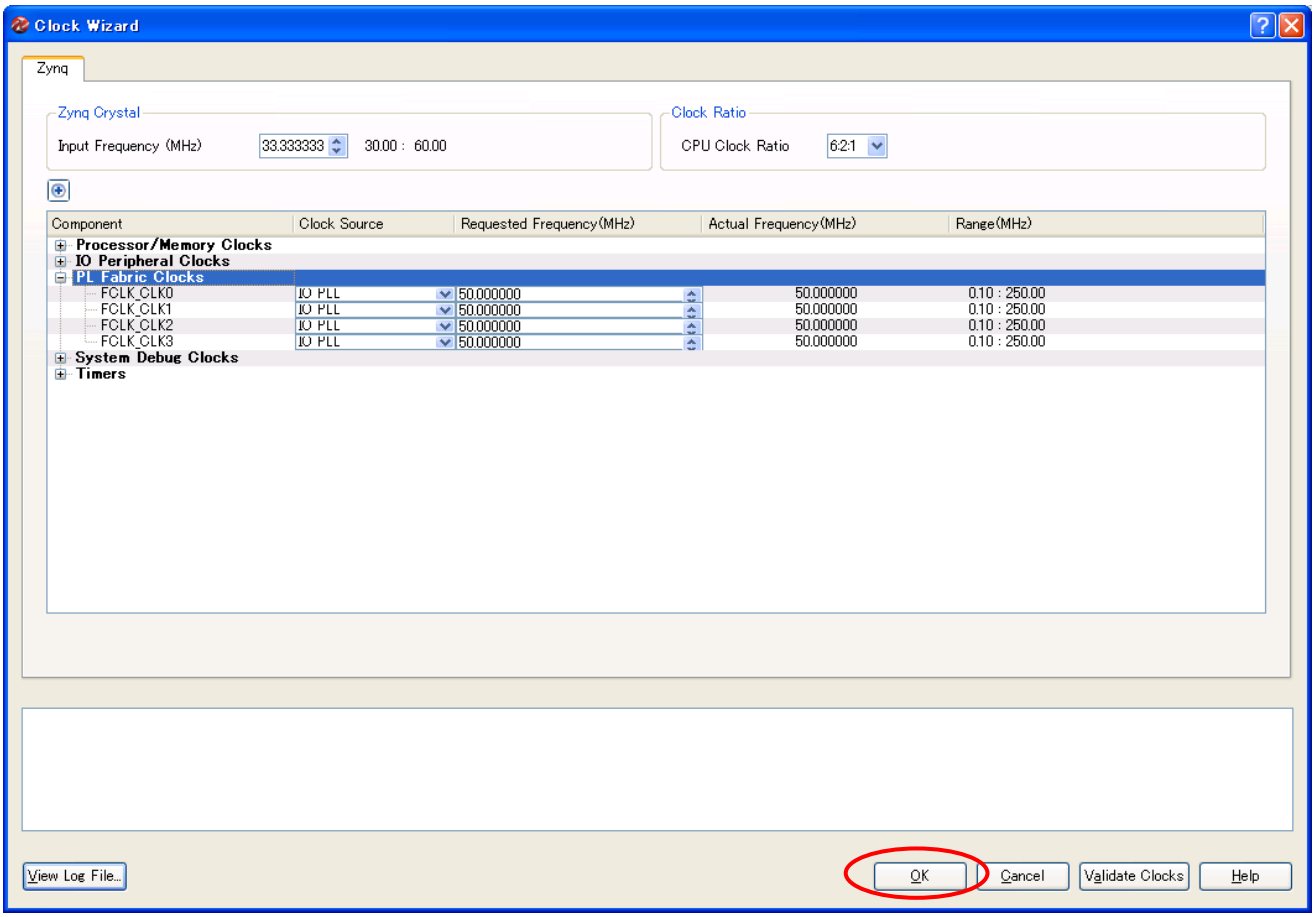
9) 以下のダイアログが出現しますので、「OK」をクリックします。



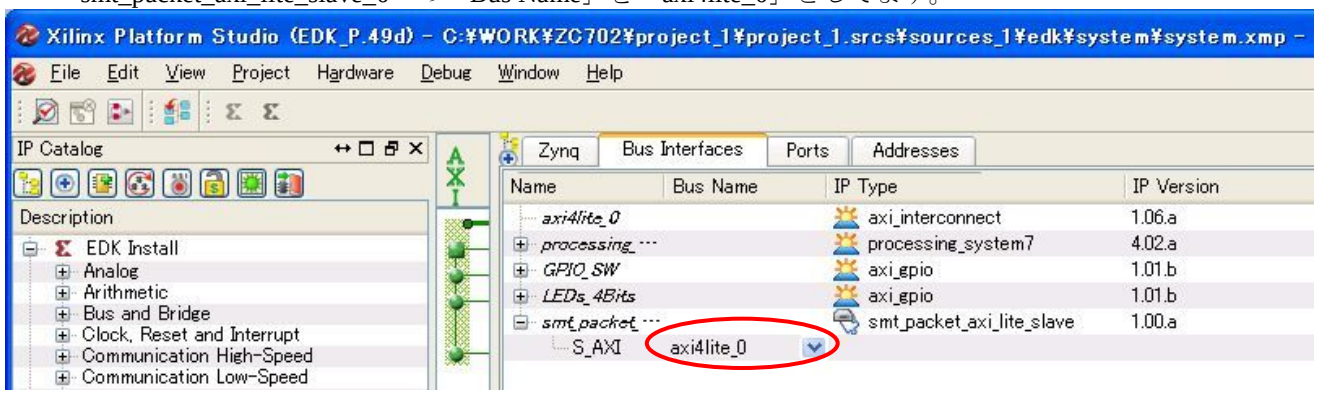
10) 「Xilinx Platform Studio」の画面に戻ります。ここで、「Clock Generation」をクリックします。



11) 以下のウィンドウが出現しますので、「PL Fabric Clocks」でパケット生成論理に接続される CLK の周波数を設定します。今回の例では、デフォルトの「50MHz」で接続します。
※パケット生成論理の出力は fmax=100MHz の使用ですので、100MHz を超える CLK は接続しないでください。
設定が完了しましたら、「OK」をクリックします。



12) 「Xilinx Platform Studio」の「Bus Interface」タブにて、適切な axi_interconnect に接続されていることを確認します。
“smt_packet_axi_lite_slave_0”の「Bus Name」を「axi4lite_0」としてます。



13) 次に smt_packet_axi_lite_slave_0”の信号定義を行います。「Ports」タブを選択し、以下の様に設定を行います。

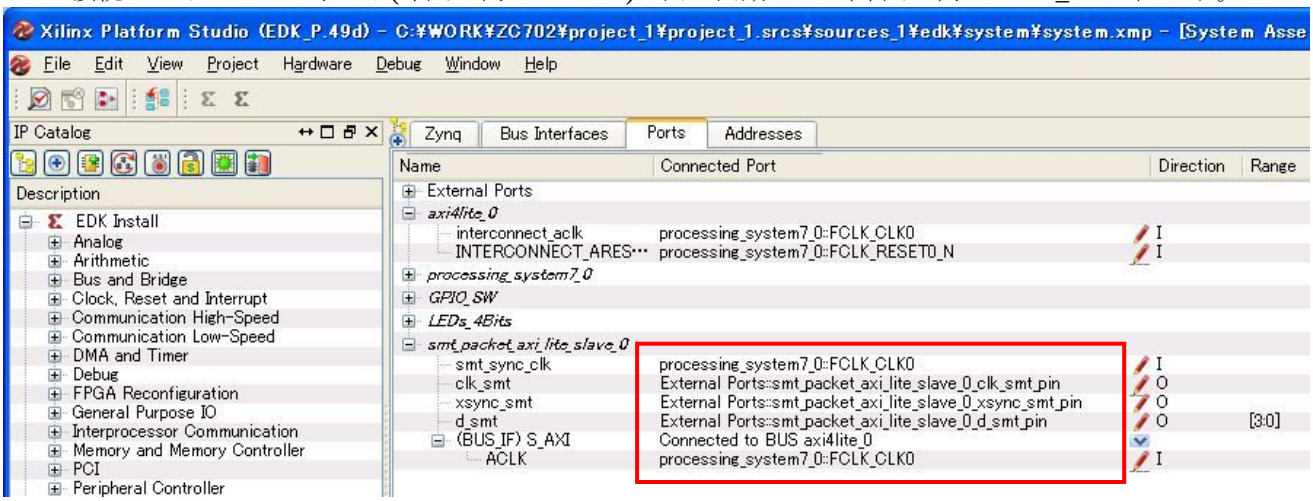
smt_sync_clk：processing_sysytem7_0::FCLK_CLKx　※1

clk_smt，xsync_smt，d_smt：External Ports::デフォルトの信号名

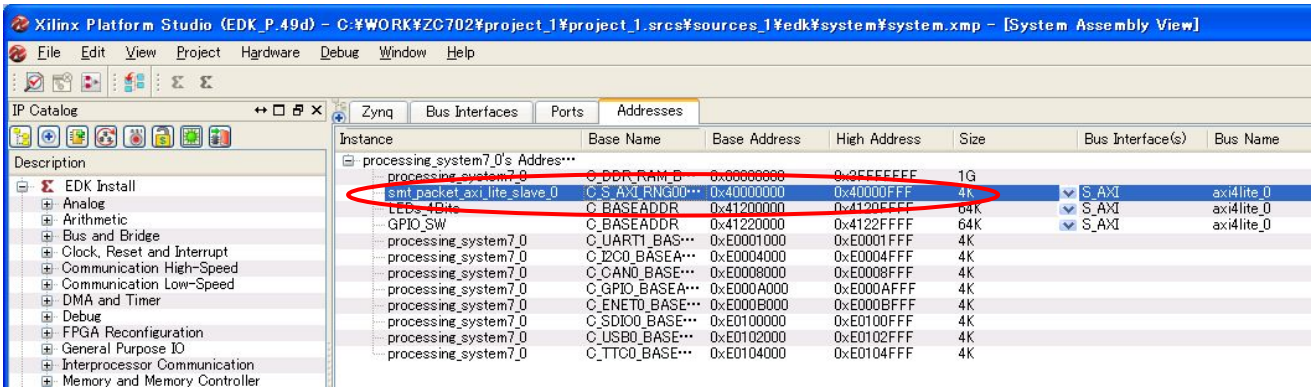
ACLK：processing_sysytem7_0::FCLK_CLKx　※2

※1　パケット論理出力側の同期 CLK です。100MHz 以下の CLK を接続してください。ACLK と同じ CLK にする必要はありません。

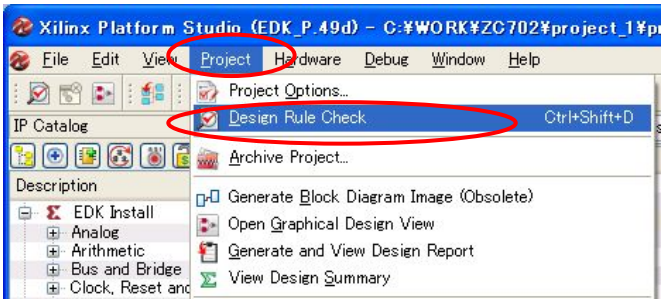
※2　接続したインターコネクト(今回の例：axi4lite0)と同じ同期 CLK（今回の例：FCLK_CLK0）です。



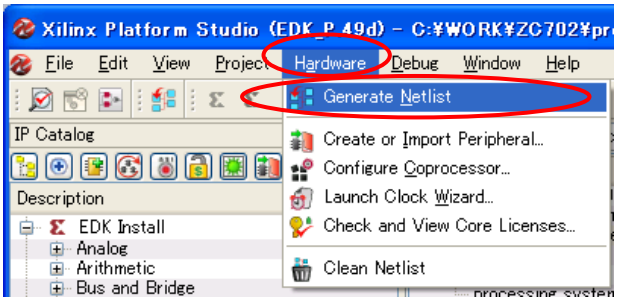
14) 「Address」タブを選択し、念のためパケット生成論理“smt_packet_axi_lite_slave_0”に設定したアドレスを確認します。



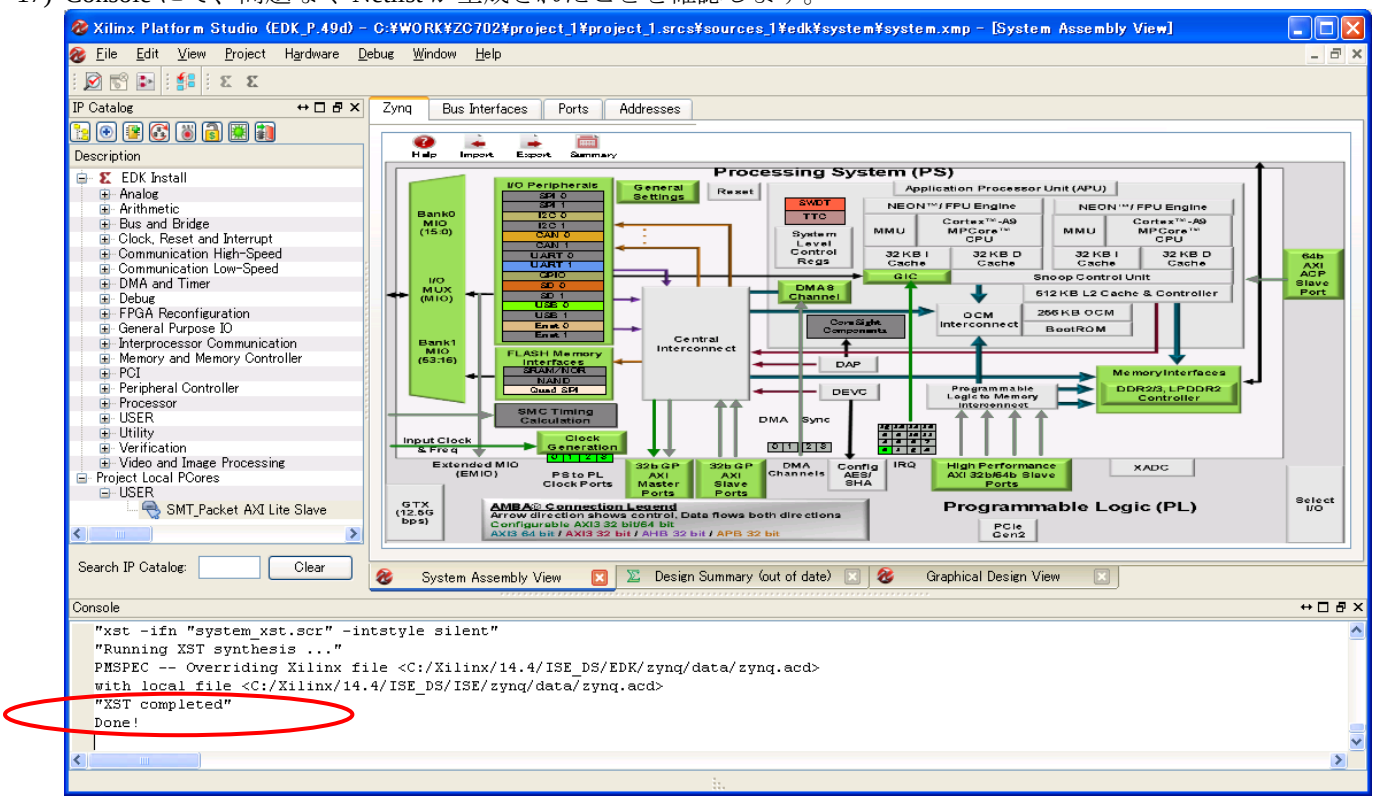
15) 以上の手順で設定が完了しましたら、「Project」→「Design Rule Check」を実行し、問題なく「Design Rule Check」が通ることを確認します。



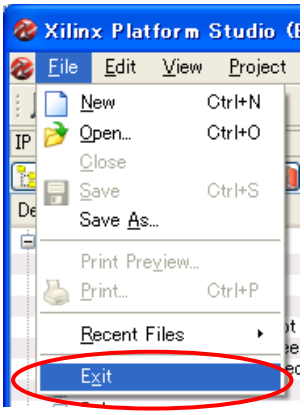
16) 次に「Hardware」→「Generate Netlist」を実行します。



17) Console にて、問題なく Netlist が生成されたことを確認します。

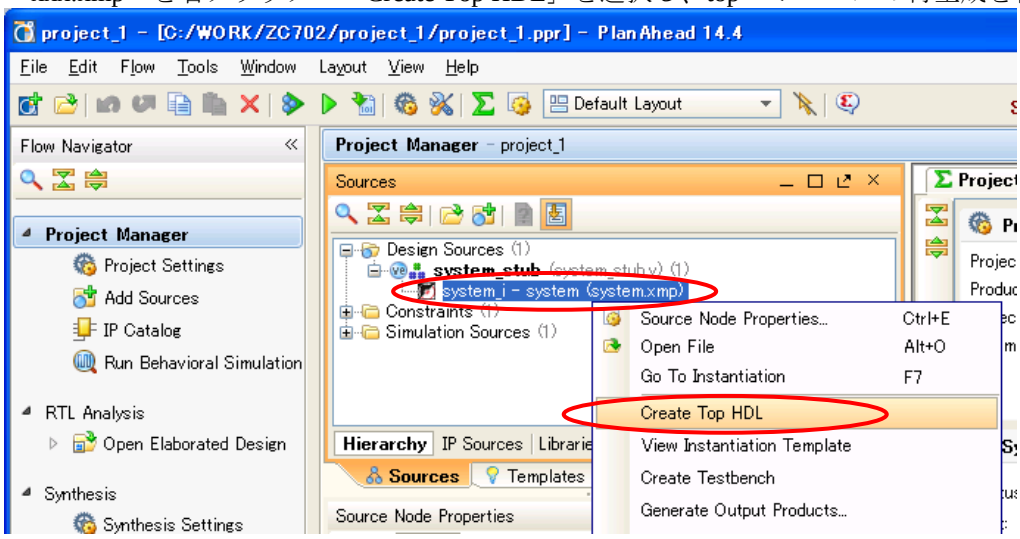


18) 「File」→「Exit」にて、「Xilinx Platform Studio」を終了します。

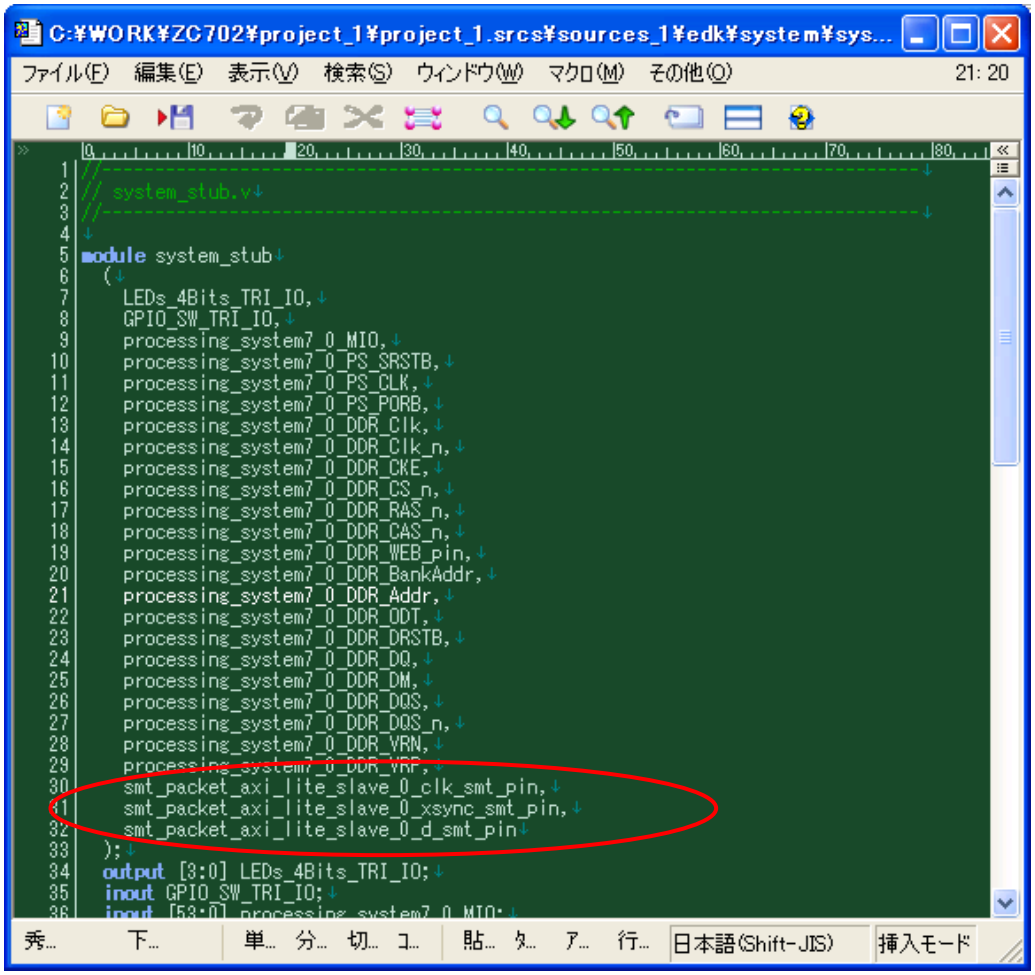


2.3 「Plan Ahead」でのコンパイル作業手順

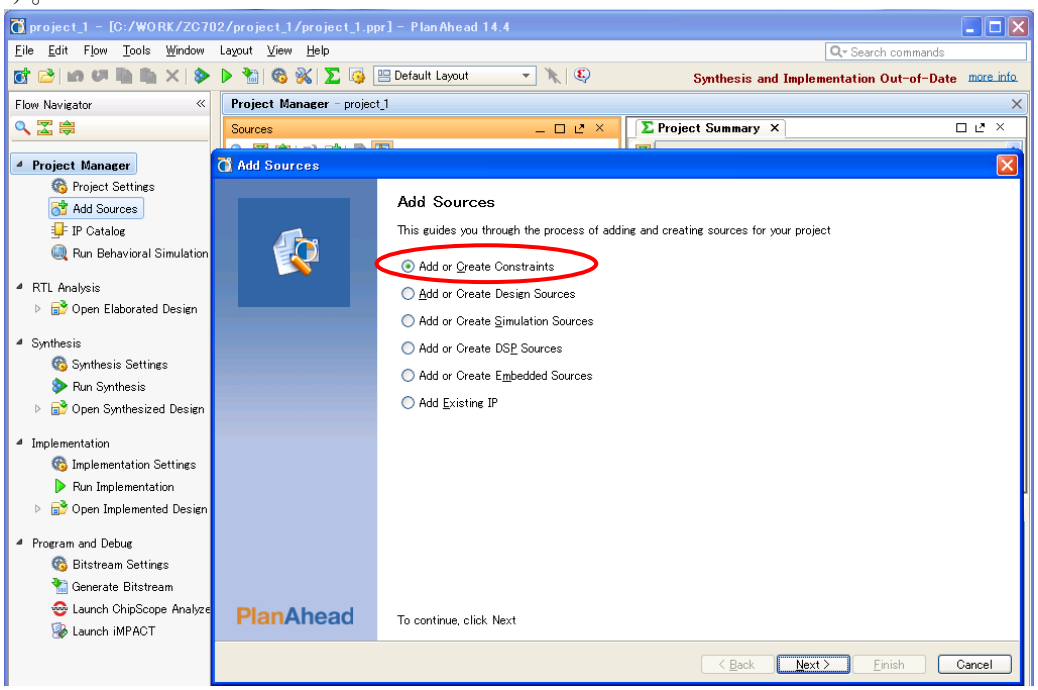
1) 「Plan Ahead」の画面に戻りましたら、「Project Manager」の「Source」ウィンドウにある“xxx.xmp”を右クリック→「Create Top HDL」を選択し、top モジュールの再生成を行います。



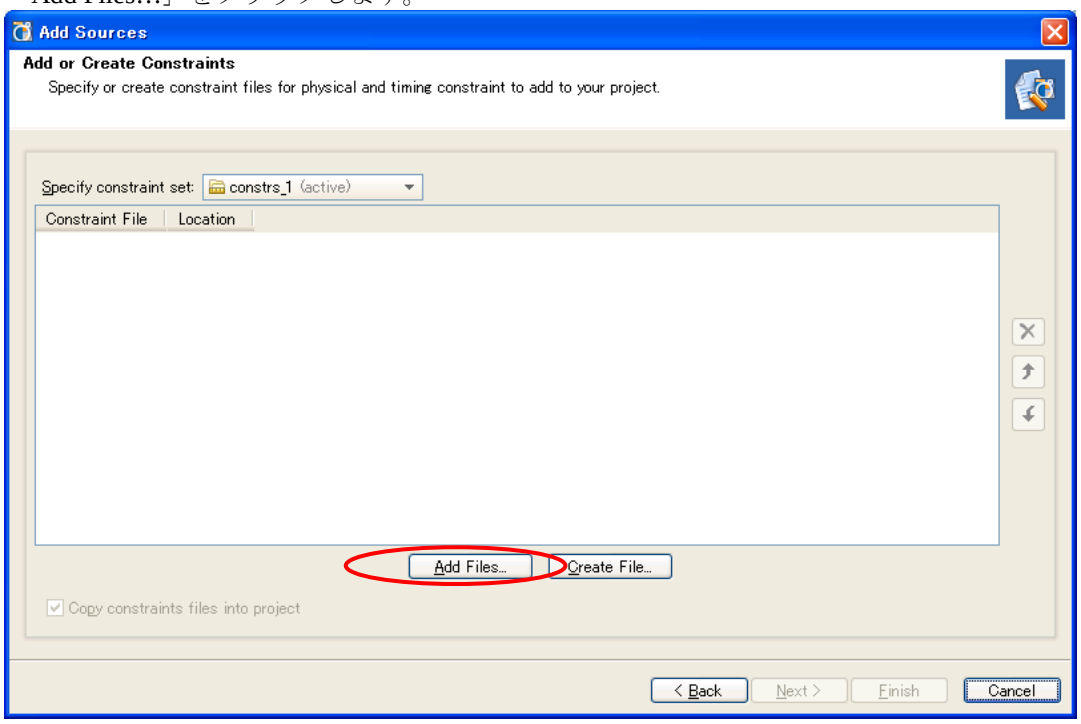
2) Top モジュールの VerilogHDL(或いは VHDL)ソースファイルを確認し、パケット生成論理の信号が反映されていることを確認します。



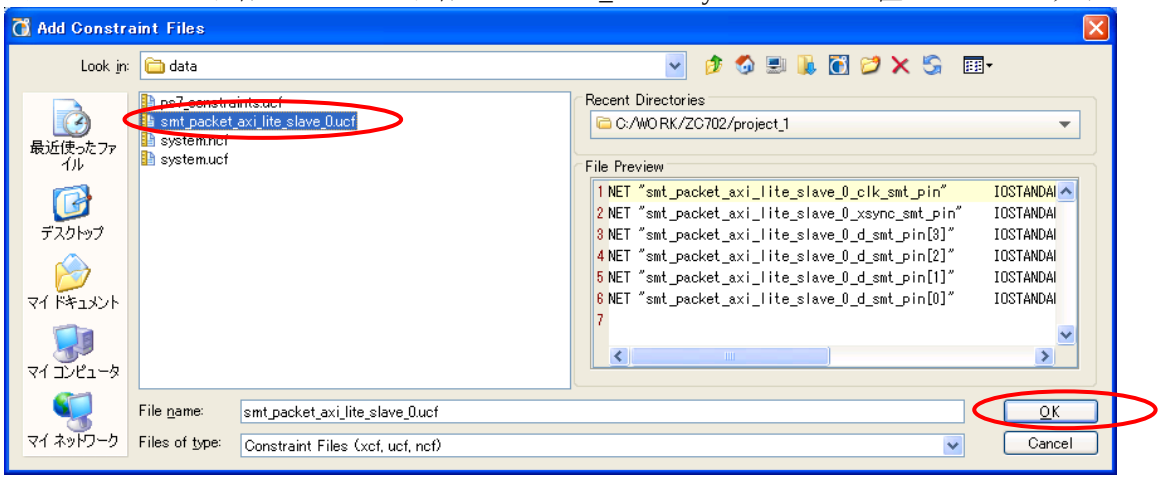
3) パケット生成論理の UCF ファイルを追加します。
「Project Manager」の「Add Sources」から、「Add or Create Constraints」を選択し、「Next」を押します。



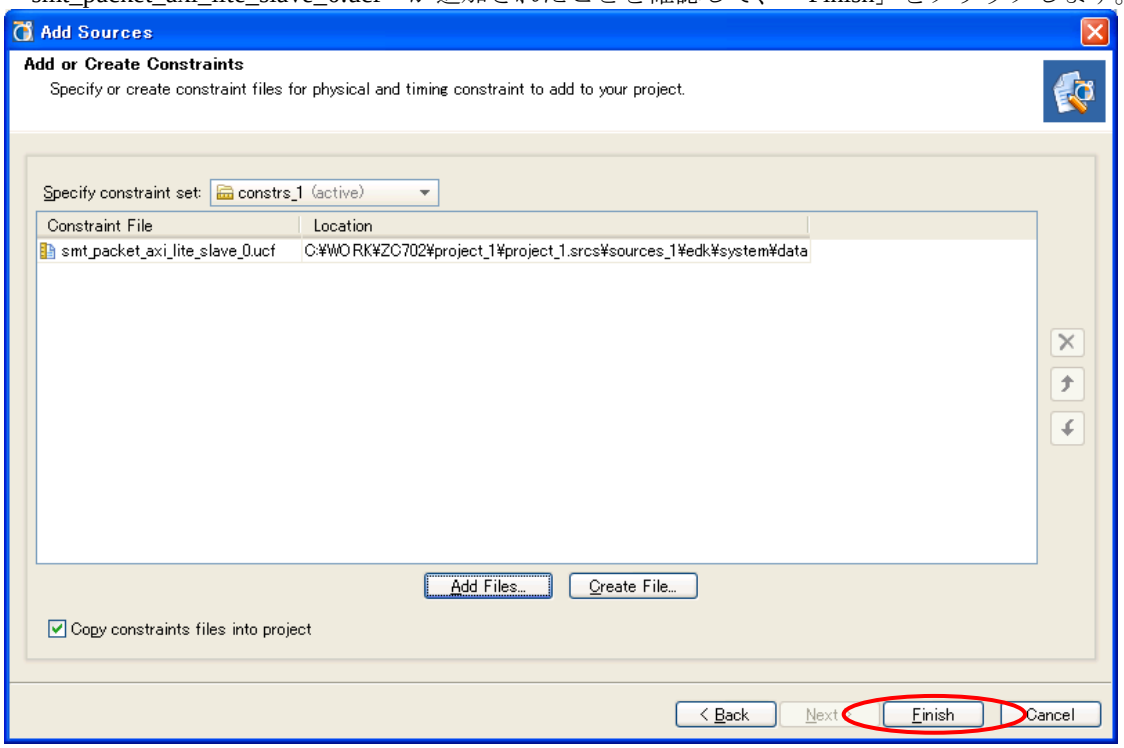
4) 「Add Files…」をクリックします。



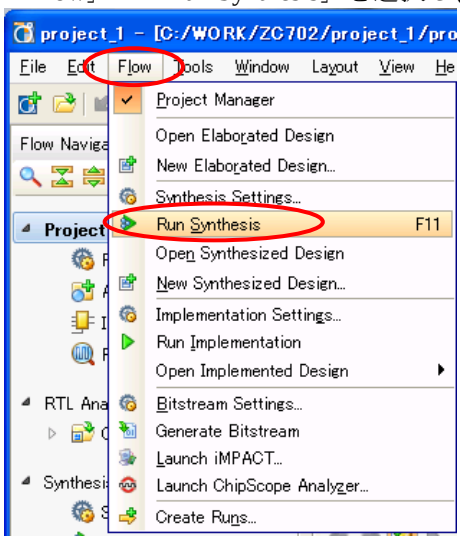
5) 以下のダイアログが出現しますので、「2.1 既存プロジェクトへのパケット生成論理組み込み準備」の3)で用意した“smt_packet_axi_lite_slave_0.ucf”を選択し、「OK」をクリックします。
※“¥プロジェクト名¥プロジェクト名.srcs¥sources_1¥edk¥system¥data”に置いた UCF ファイル



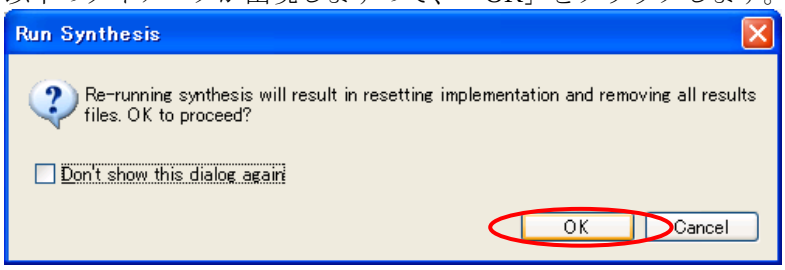
6) “smt_packet_axi_lite_slave_0.ucf”が追加されたことを確認して、「Finish」をクリックします。



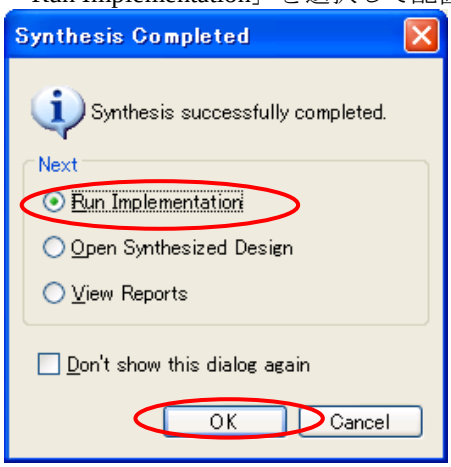
7) 「Flow」→「Run Synthesis」を選択し、論理合成を開始します。



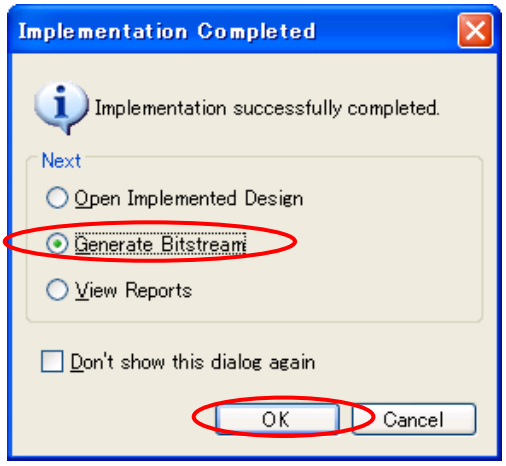
8) 以下のダイアログが出現しますので、「OK」をクリックします。



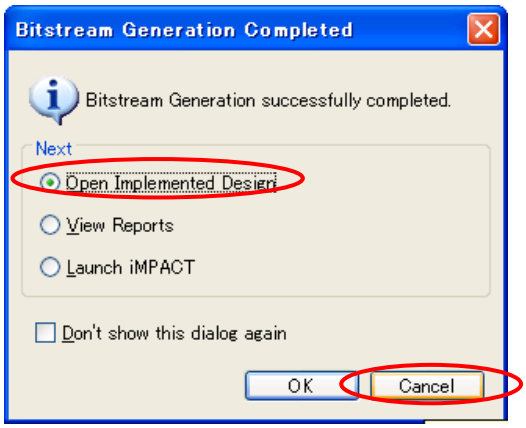
9) 論理合成が完了すると、以下のダイアログが出現します
「Run Implementation」を選択して配置配線を実施します。



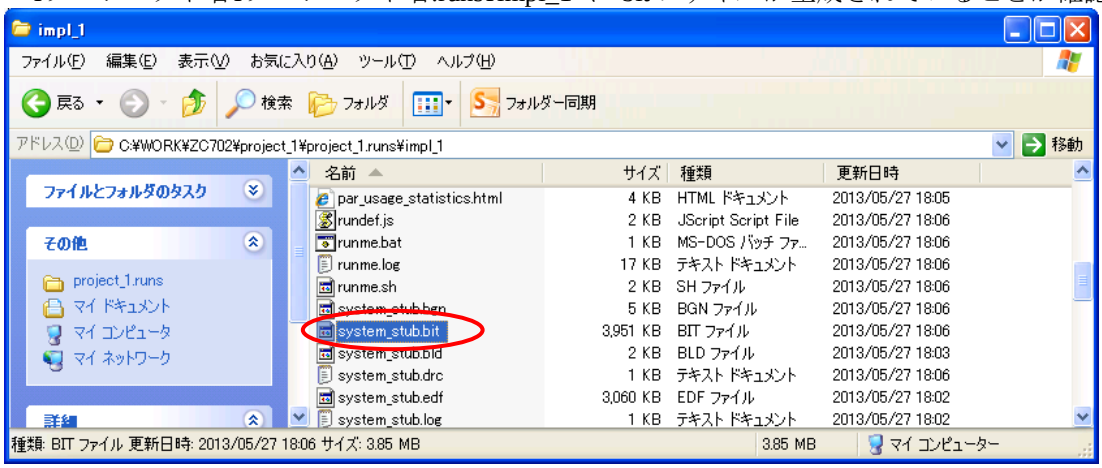
10) 配置配線が完了すると、以下のダイアログが出現します。
「Generate Bitstream」を選択し、「OK」をクリックします。



11) Bit ファイルが無事に生成されると、以下のダイアログが出現します。
ここでは「Cancel」をクリックしてください。



12) Bit ファイルが生成されたことを確認します。
“¥プロジェクト名¥プロジェクト名.runs¥impl_1”に bit ファイルが生成されていることが確認できます。

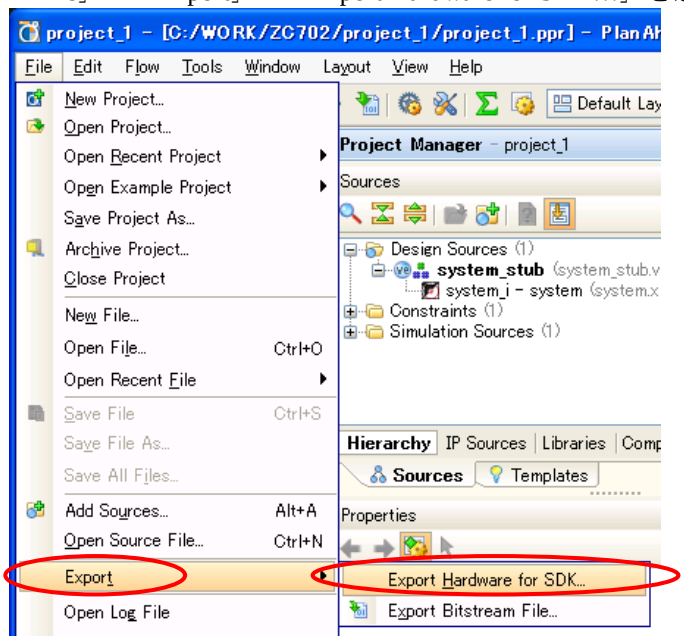


以上で、パケット生成論理の組み込みが完了しました。

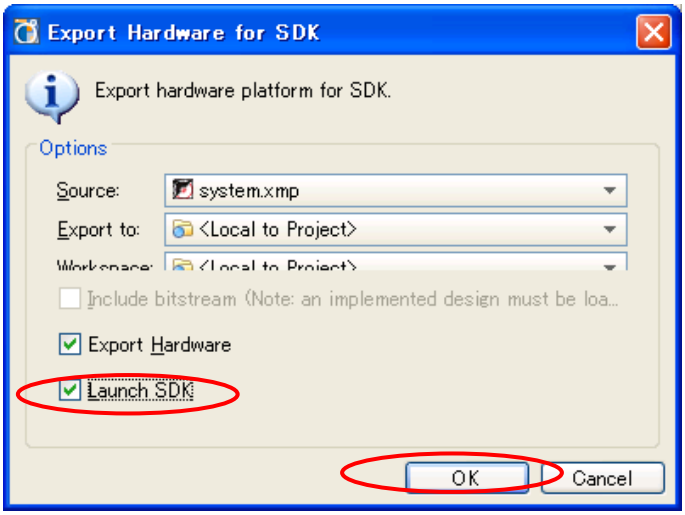
3 SD BOOT イメージ生成

ここでは、Linux の SD BOOT イメージの生成手順を例に記載します。

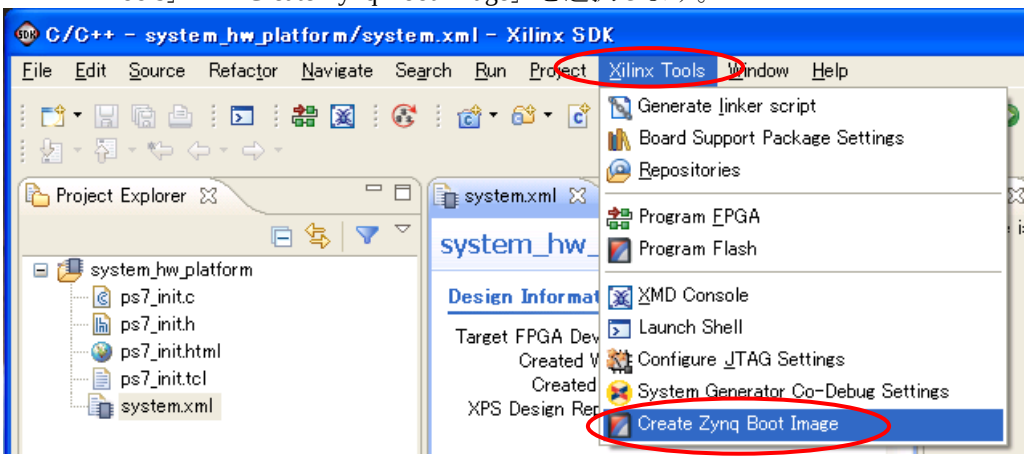
1) Plan Ahead から Xilinx Software Development Kit(SDK)を立ち上げます。
「File」→「Export」→「Export Hardware for SDK…」を選択します。



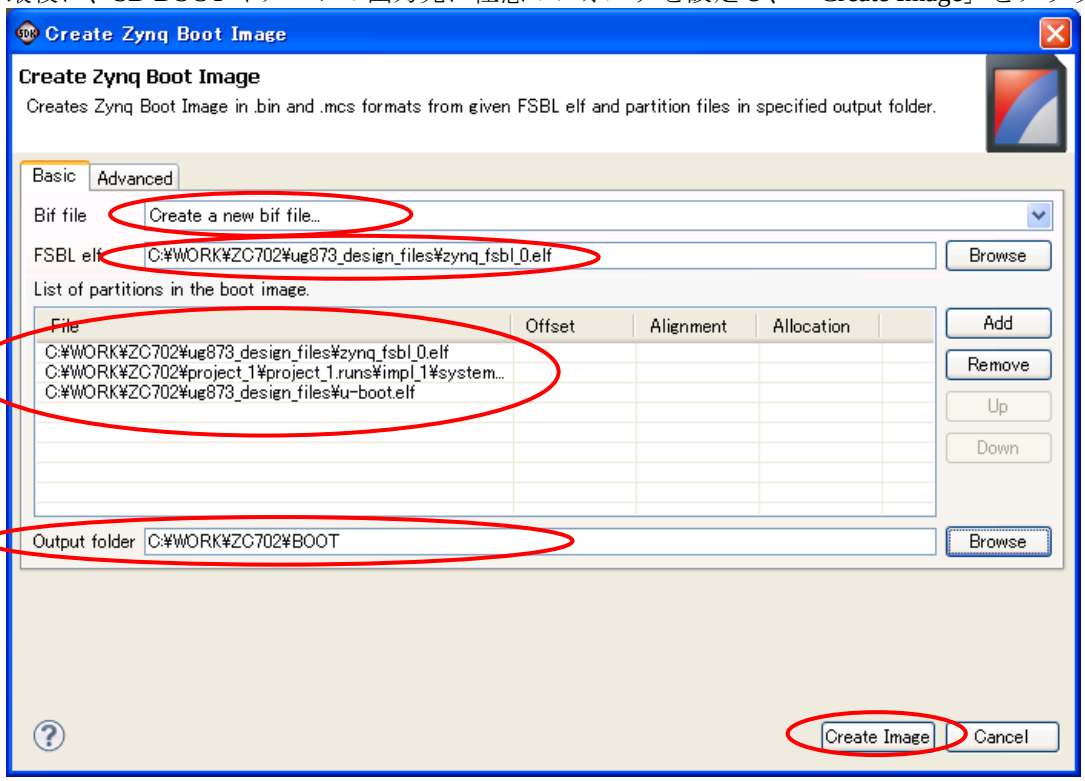
2) 以下のダイアログが出現しますので、「Launch SDK」にチェックを付けて、「OK」をクリックします。



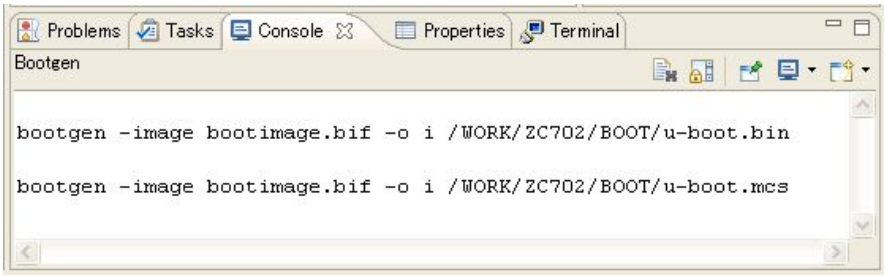
3) Xilinx Software Development Kit(SDK)が立ち上がります。
「Xilinx Tools」→「Create Zynq Boot Image」を選択します。



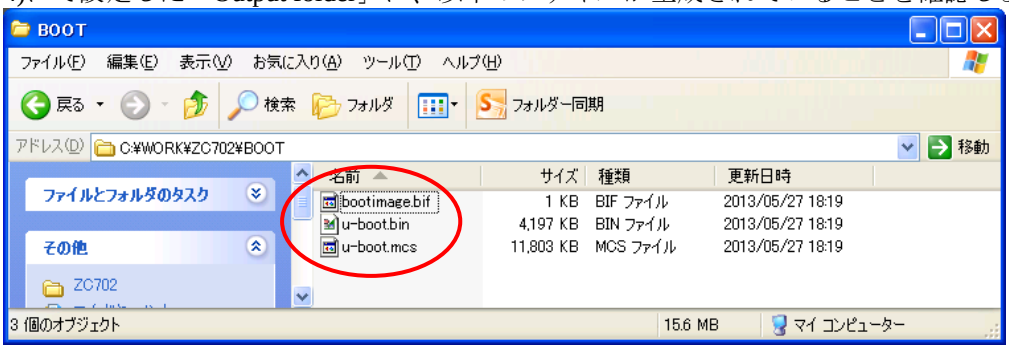
4) 以下のダイアログが出現しますので、
「Bit file」を「Create a new bit file…」を選択し、「FSBL elf」に fsbl.elf ファイルを設定します。
次に、「List of partitions in the boot image」に、前章で生成された bit ファイルと、u-boot.elf ファイルを「Add」ボタンで追加していきます。
最後に、SD BOOT イメージの出力先に任意のフォルダを設定し、「Create Image」をクリックします。



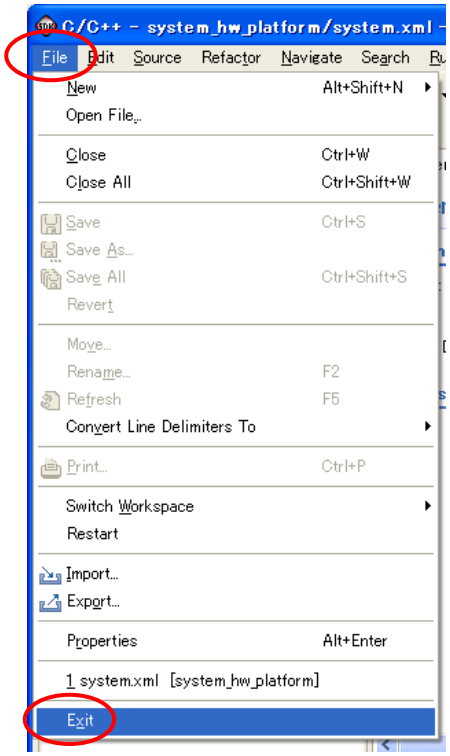
5) SD BOOT イメージの生成が完了すると、Console タブにて以下のメッセージが確認できます。



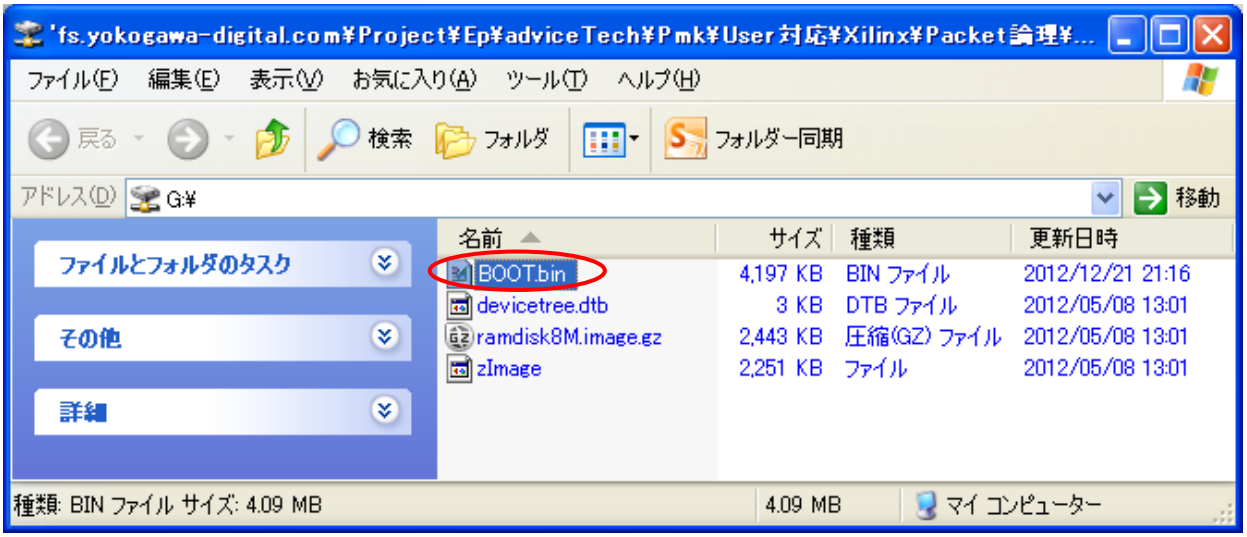
6) 4)にて設定した「Output folder」に、以下のファイルが生成されていることを確認します。



7) SDK の「File」→「Exit」を選択し、SDK を終了します。



8) “u-boot.bin”を SD カードにコピーし、“u-boot.bin”を“BOOT.bin”に rename すれば完了です。



以上。