

## FPGA マガジン No.3 サンプルデータ

OpenCores EtherMAC MicroBoard 用設計データの使い方 (2013/10/25)

### ファイル一覧

- ・ mb\_mcs\_sys.v      トップ回路
- ・ iobus2wb.v      WishBone バスブリッジ
- ・ iobus\_reg.v      設定レジスタ回路
- ・ iobus\_bram.v      ブロック RAM インターフェース回路
- ・ user\_module.v      デバック回路
- ・ test\_mcs\_sys.v      テストベンチ
- ・ mb\_mcs\_sys.ucf      ピン配置指定
- ・ ethermac\_microboard\_sw1.c      EtherMAC 制御用 C ソースコード

### 免責事項

本データの使用が原因として発生した損失や損害について、(有) ひまわり および 著作者は一切責任を負いません。著作者：横溝憲治 fpga@hmwr-lsi.co.jp

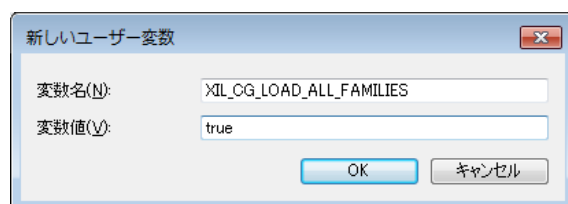
### 手順

- ・ 設計用フォル ethermac/mb\_mcs\_sys を作成する
- ・ 記事のダウンロードデータを解凍して、ether\_mac\_bm\_data の下にある Verilog-HDL ソースと UCF ファイルを ethermac/mb\_mcs\_sys へコピーする。
- ・ EtherMAC のデータを CopenCores のサイト (<http://opencores.org/project,ethmac>) からダウンロード
- ・ ダウンロードした ethmac\_latest.tar.gz を解凍する
- ・ 解凍データの ethermac/trunk/rtl/verilog の下にある Verilog-HDL ソースを ethermac/mb\_mcs\_sys へコピー

- ・ I<sup>2</sup>C のデータを CopenCores のサイト (<http://opencores.org/project,i2c>) からダウンロード
- ・ ダウンロードした i2c\_latest.tar.gz を解凍する
- ・ 解凍データの i2c/trunk/rtl/verilog の下にある Verilog-HDL ソースを ethermac/mb\_mcs\_sys へコピー

- ・ PWM のデータを CopenCores のサイト (<http://opencores.org/project,pwm>) からダウンロード
- ・ ダウンロードした pwm\_latest.tar.gz を解凍する
- ・ 解凍データの pwm/trunk/RTL の下にある Verilog-HDL ソースを ethermac /mb\_mcs\_sys へコピー
- ・ 環境変数 XIL\_CG\_LOAD\_ALL\_FAMILIES が設定されてない場合は環境変数を追加する

コントロールパネル→システムとセキュリティ→システム→システムの詳細設定→環境変数→ユーザー環境変数：新規をクリック



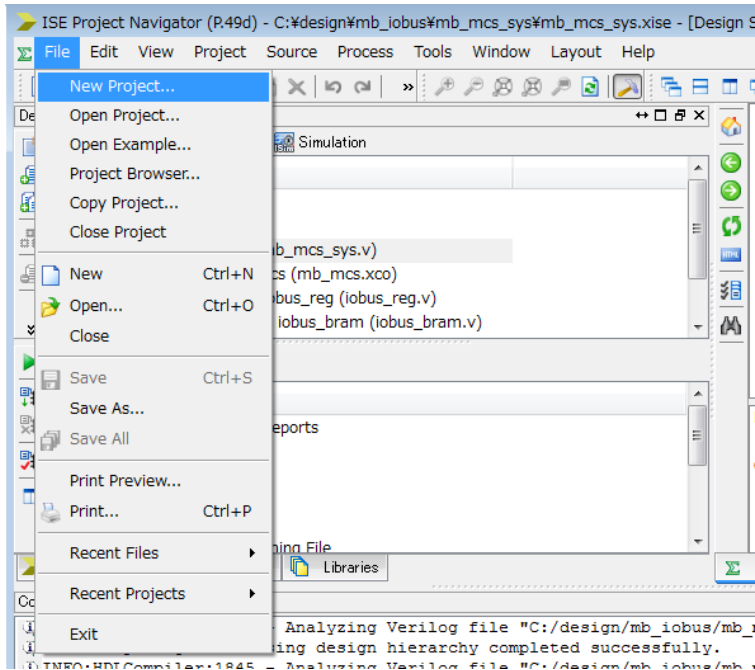
変数名 : XIL\_CG\_LOAD\_ALL\_FAMILIES、値 : true、と入力して OK

・ ProjectNavigator を起動します。

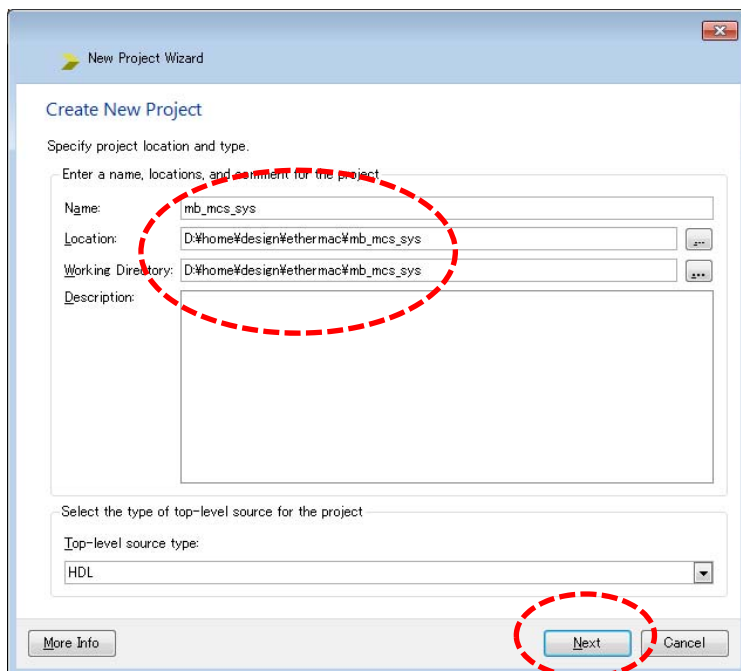
スタートメニューから「Xilinx Design Tools」→「ISE Design Suite 14.6」→「ISE Design Tools」→Project Navigator」を起動してください。



・ 新規設計プロジェクト作成

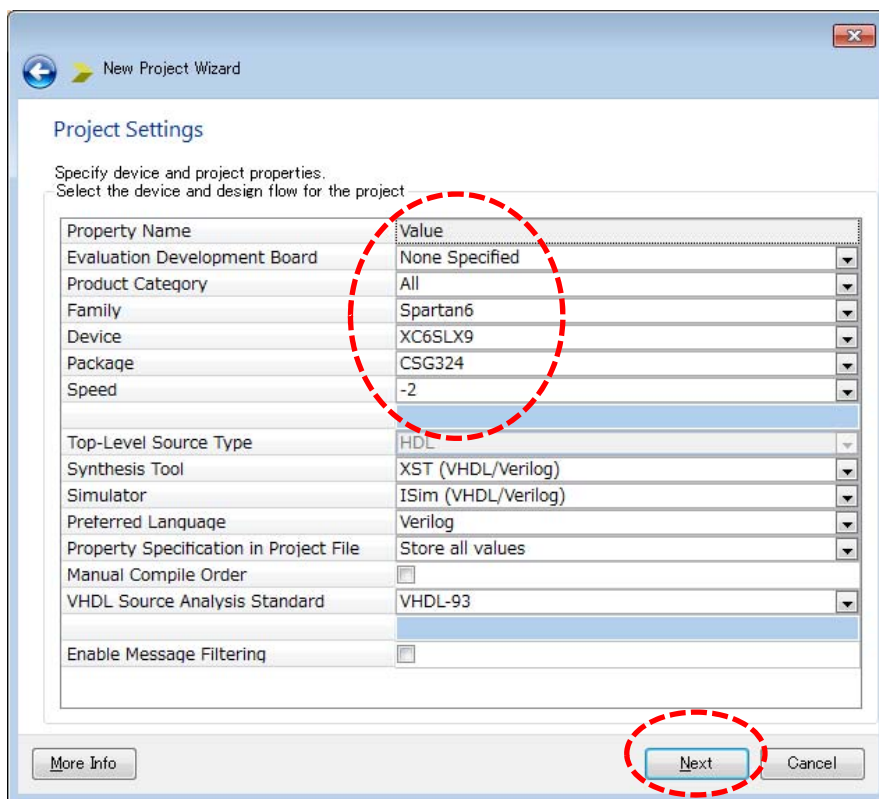


File→New Project 選択



プロジェクト名 : mb\_mcs\_sys、設計フォルダ : 任意フォルダ/ethermac/mb\_mcs\_sys、Next をクリック



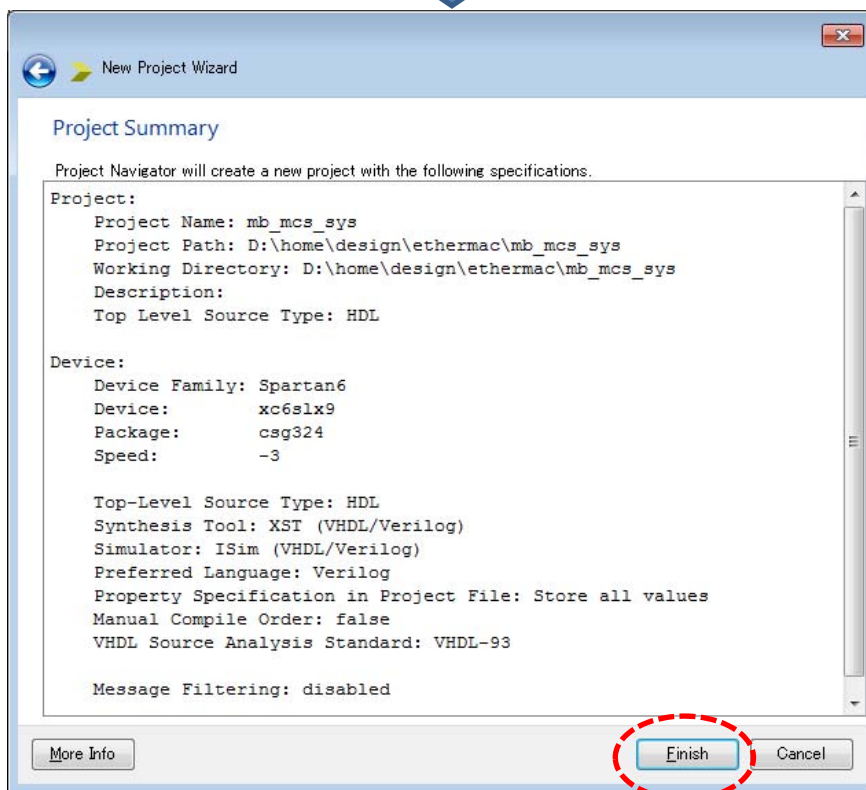


The 'New Project Wizard' dialog box is shown at the 'Project Settings' step. It contains a table of properties and their values. A red dashed circle highlights the 'Value' column of the table. At the bottom right, the 'Next' button is also circled with a red dashed line.

Property Name	Value
Evaluation Development Board	None Specified
Product Category	All
Family	Spartan6
Device	XC6SLX9
Package	CSG324
Speed	-2
<hr/>	
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISim (VHDL/Verilog)
Preferred Language	Verilog
Property Specification in Project File	Store all values
Manual Compile Order	<input type="checkbox"/>
VHDL Source Analysis Standard	VHDL-93
<hr/>	
Enable Message Filtering	<input type="checkbox"/>

Buttons: More Info, Next, Cancel

デバイス指定 LX9 マイクロボードに合わせて、Next をクリック



The 'New Project Wizard' dialog box is shown at the 'Project Summary' step. It displays a text summary of the project specifications. At the bottom right, the 'Finish' button is circled with a red dashed line.

Project Navigator will create a new project with the following specifications.

Project:

- Project Name: mb\_mcs\_sys
- Project Path: D:\home\design\ethermac\mb\_mcs\_sys
- Working Directory: D:\home\design\ethermac\mb\_mcs\_sys
- Description:
- Top Level Source Type: HDL

Device:

- Device Family: Spartan6
- Device: xc6slx9
- Package: csg324
- Speed: -3

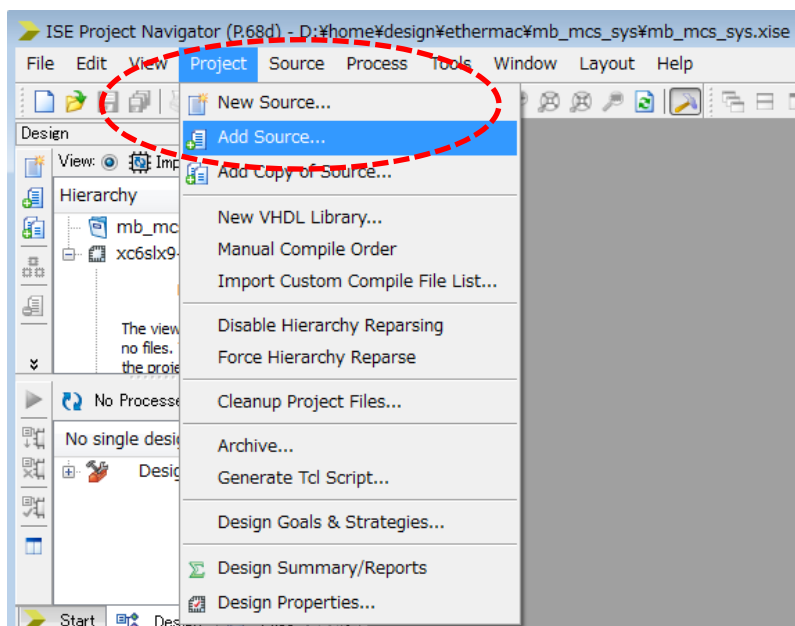
Top-Level Source Type: HDL  
Synthesis Tool: XST (VHDL/Verilog)  
Simulator: ISim (VHDL/Verilog)  
Preferred Language: Verilog  
Property Specification in Project File: Store all values  
Manual Compile Order: false  
VHDL Source Analysis Standard: VHDL-93

Message Filtering: disabled

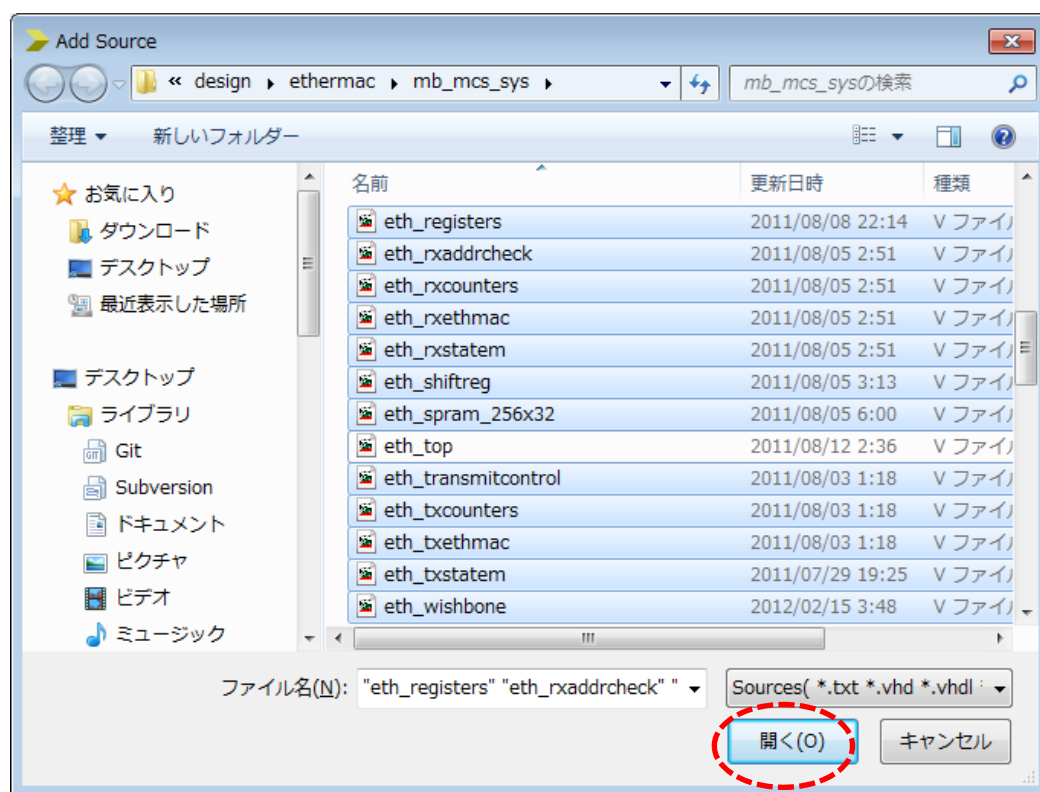
Buttons: More Info, Finish, Cancel

Finish をクリック



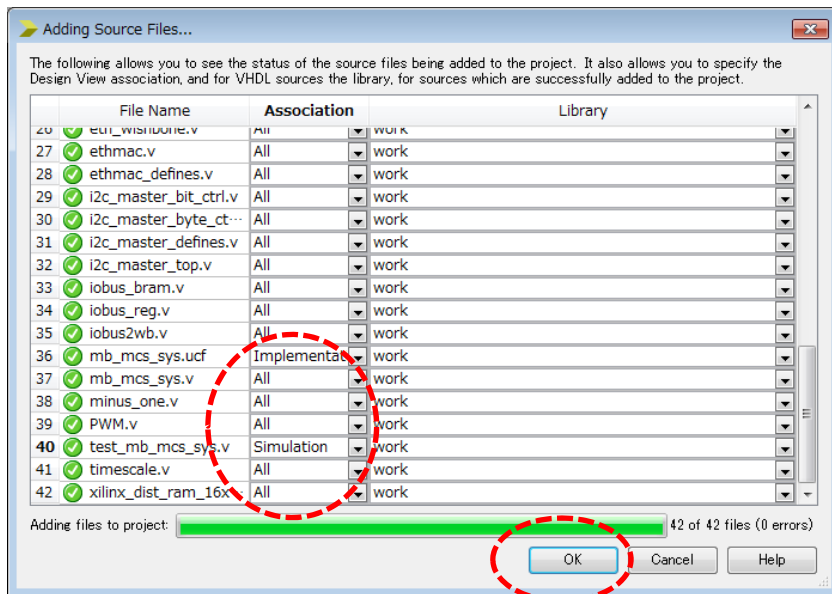


設計データを追加します。Project→Add Source を選択

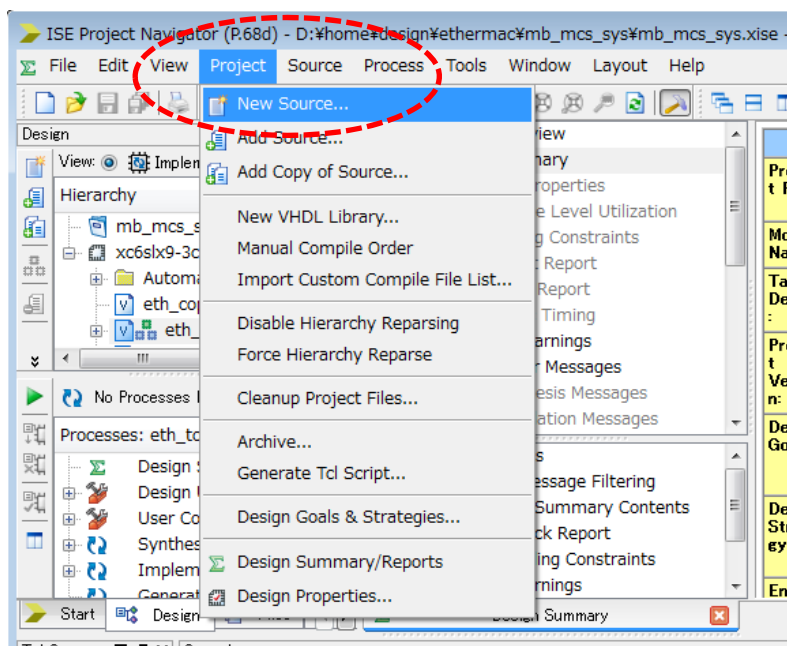


ファイルの指定、mb\_mcs\_sys の下にある Verilog-HDL ファイル (eth\_cop.v, eth\_top.v, xilinx\_dist\_ram\_16x32.v を除く\*.v ファイル) と UCF ファイルを指定



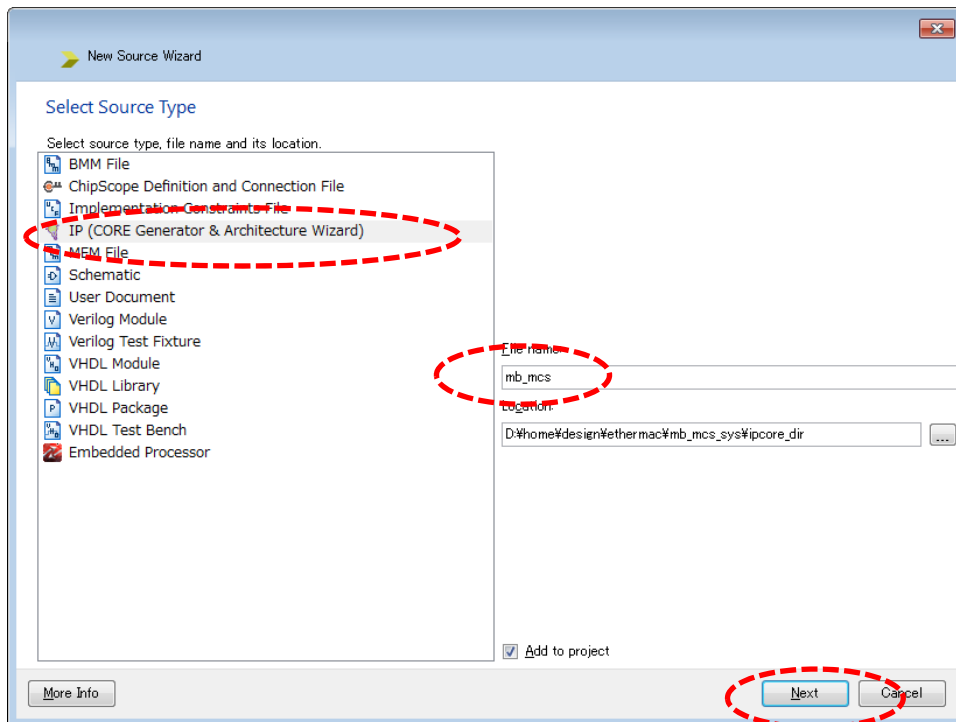


mb\_mcs\_sys.ucf はインプリメンテーションで使用するので Implementation を指定  
 test\_mb\_mcs\_sys.v はテストベンチなので Simulation を指定

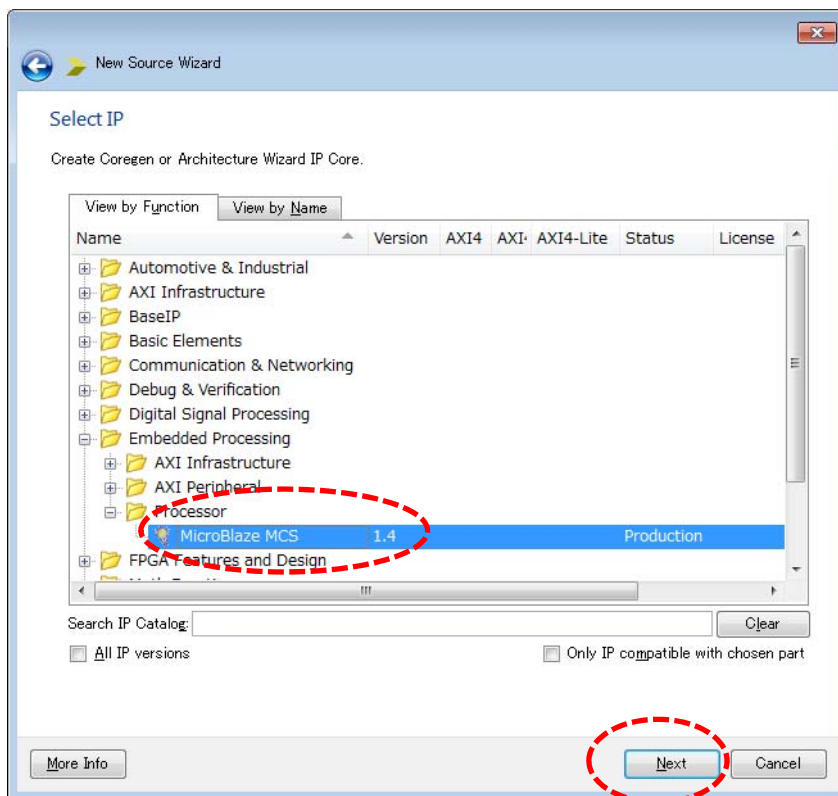


ProjectNavigator で MicroBlaze MCS を追加する。Project→New Source を選択



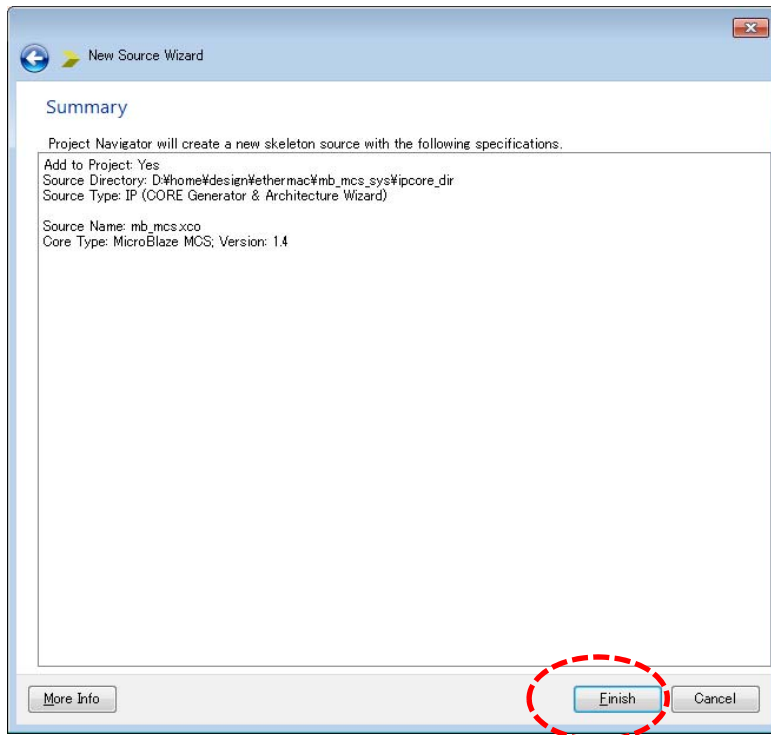


P(CORE Gener…をクリックして選択、ファイル名に mb\_mcs を指定、Next をクリック

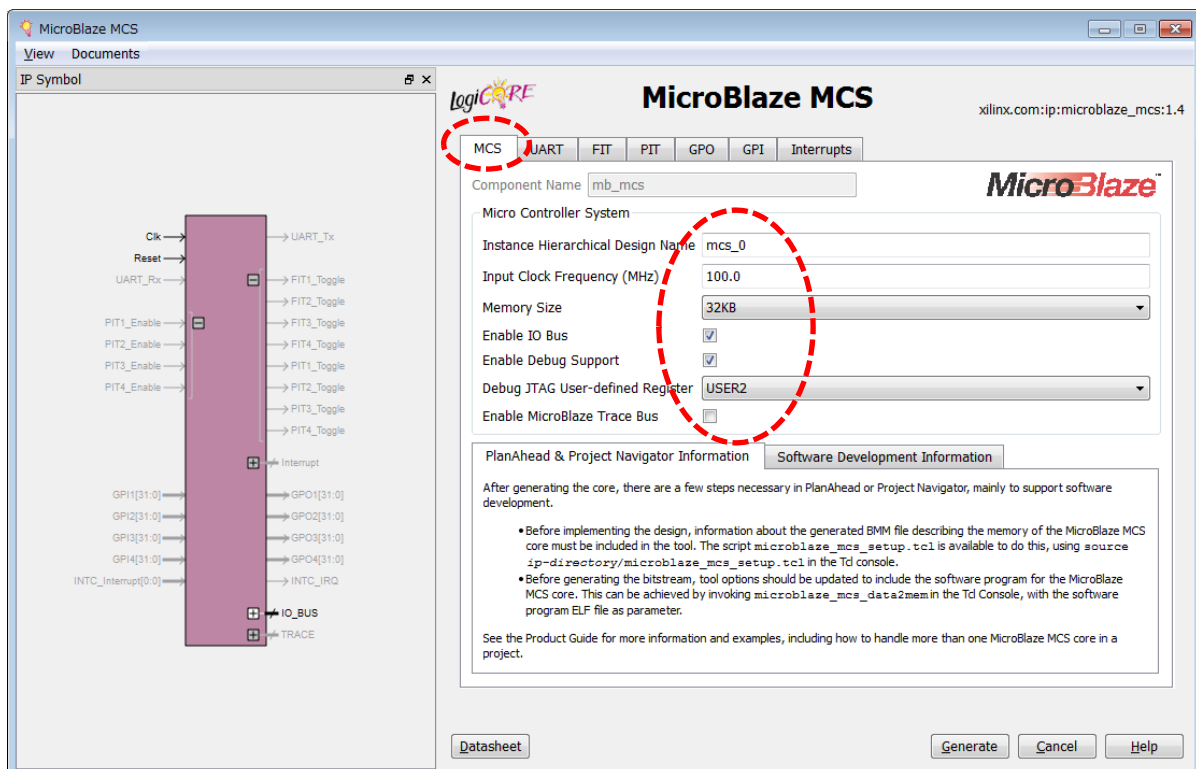


MicroBlaze MCS 選択して Next をクリック



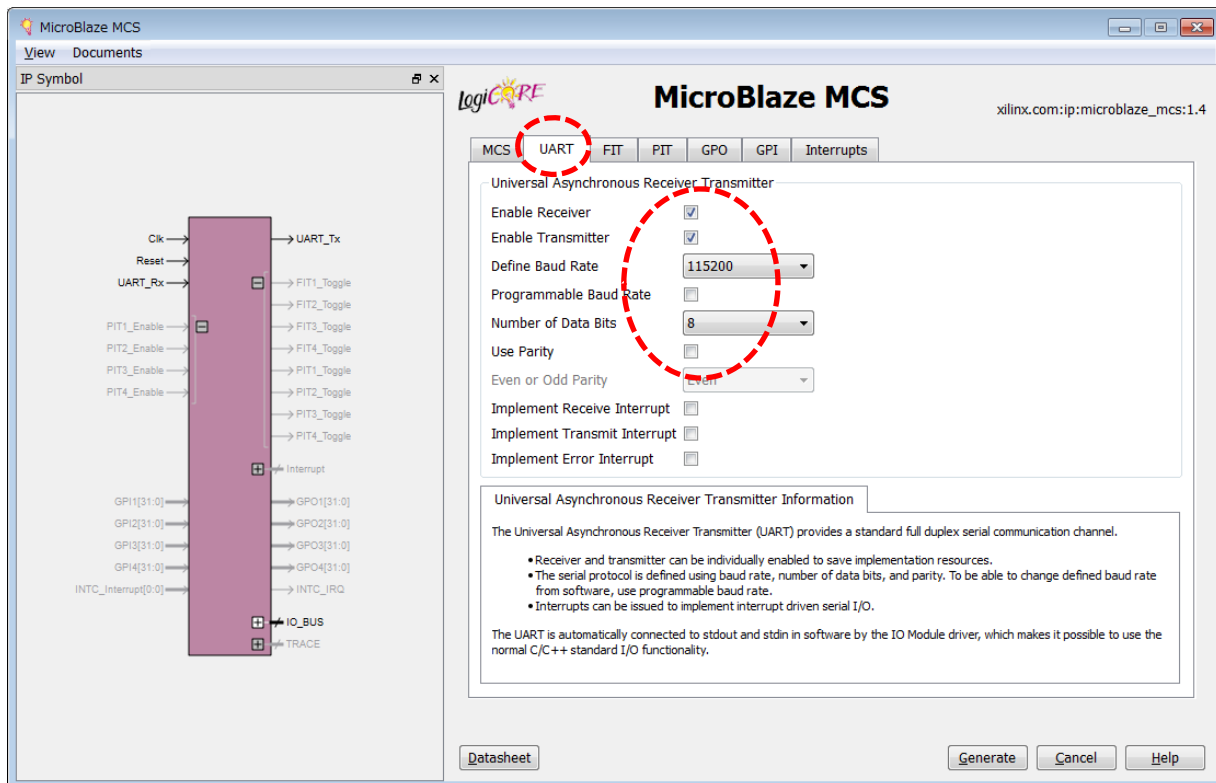


FinishでCORE generator 起動

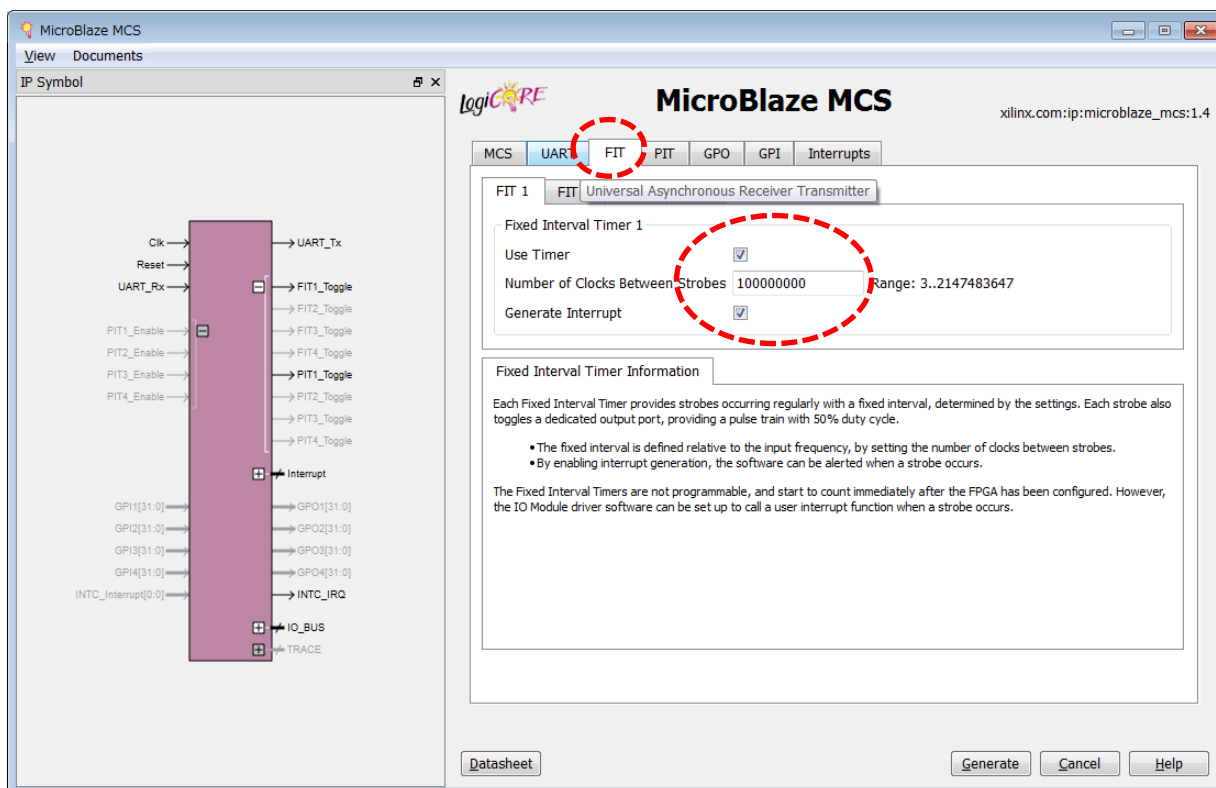


MicroBlaze MCS の基本設定





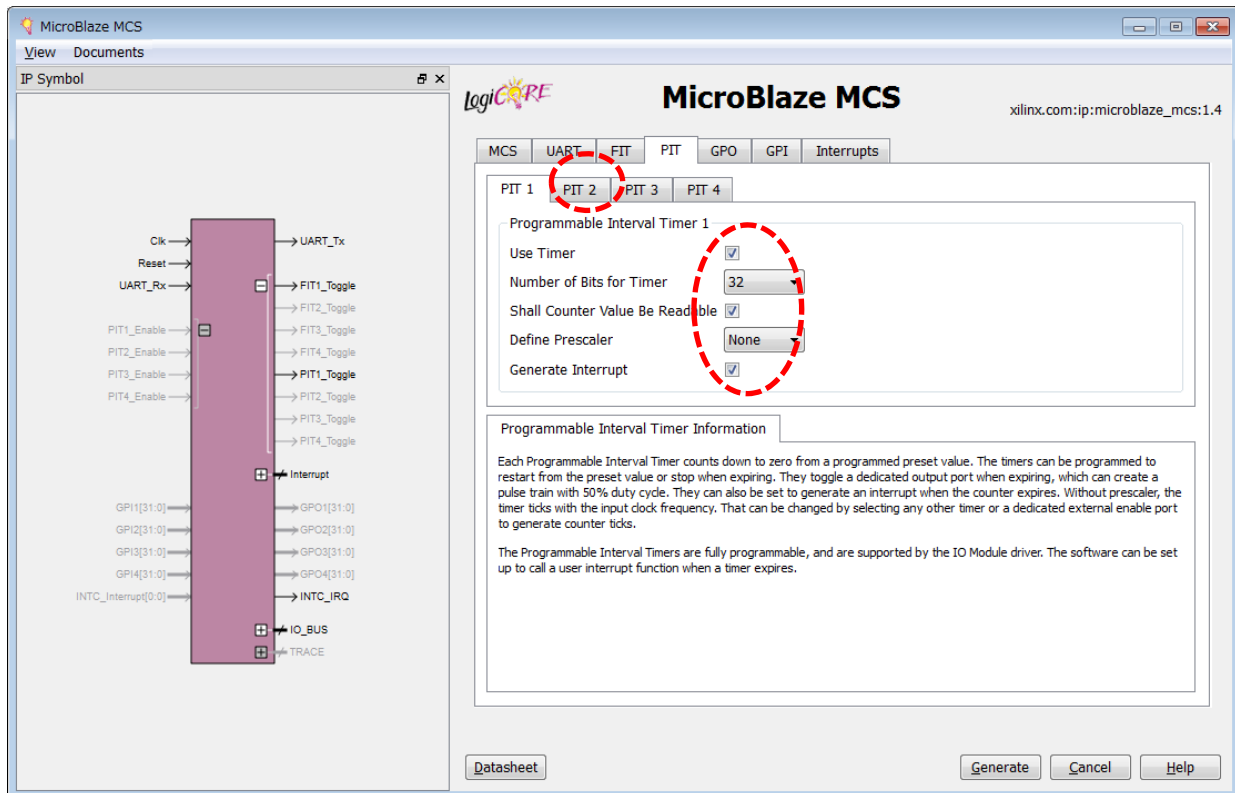
UART の設定



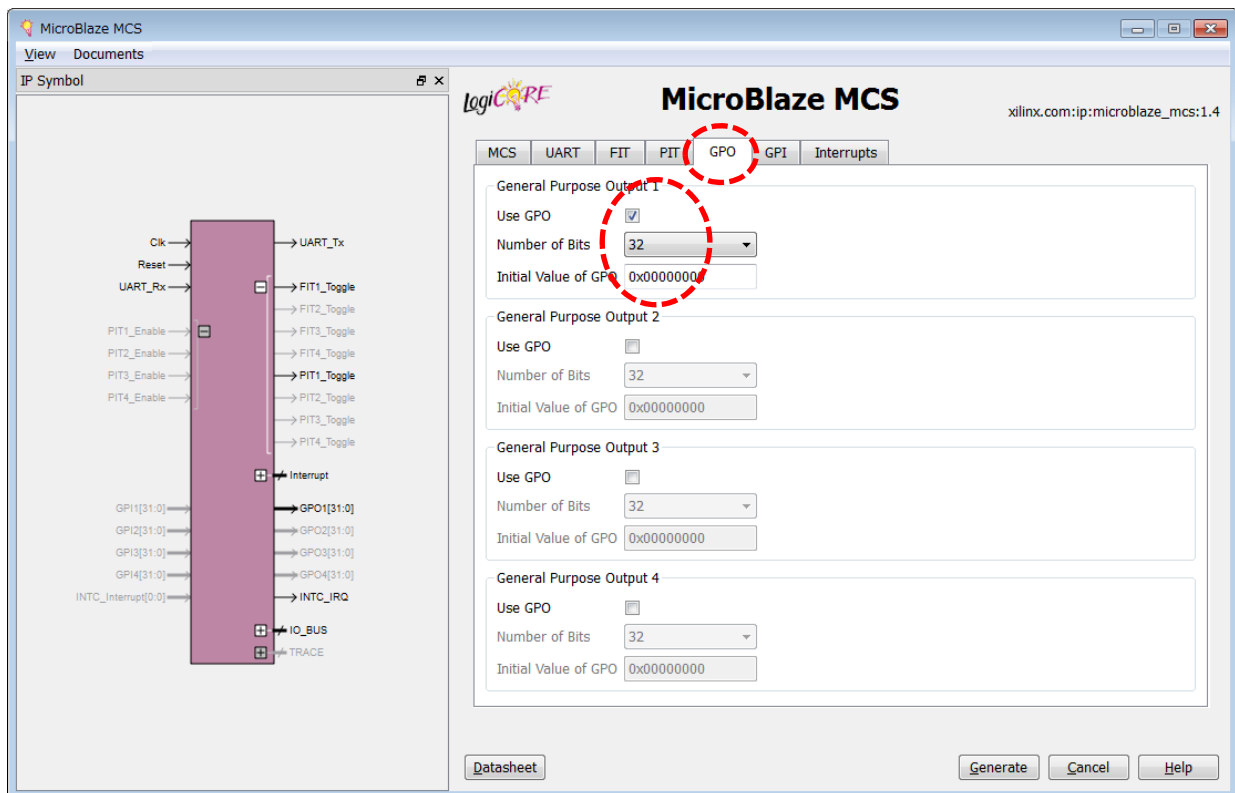
FIT の指定





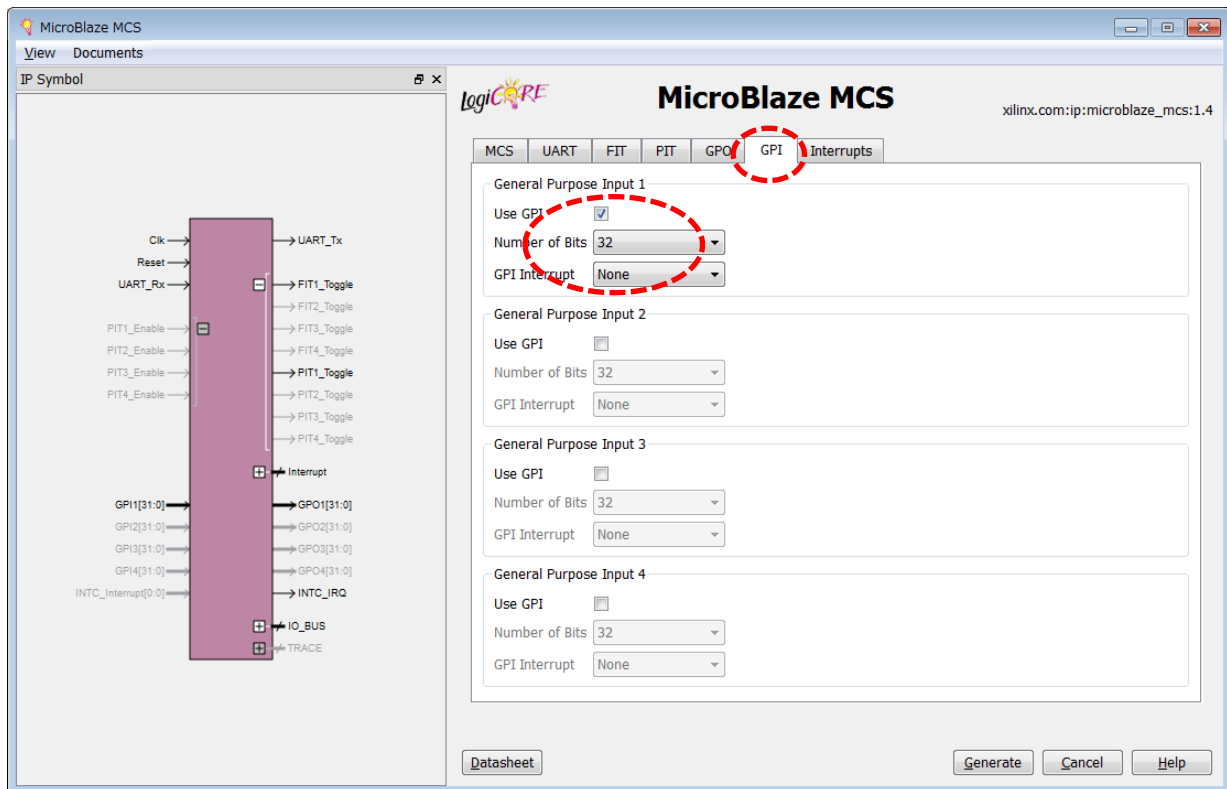


PIT の指定

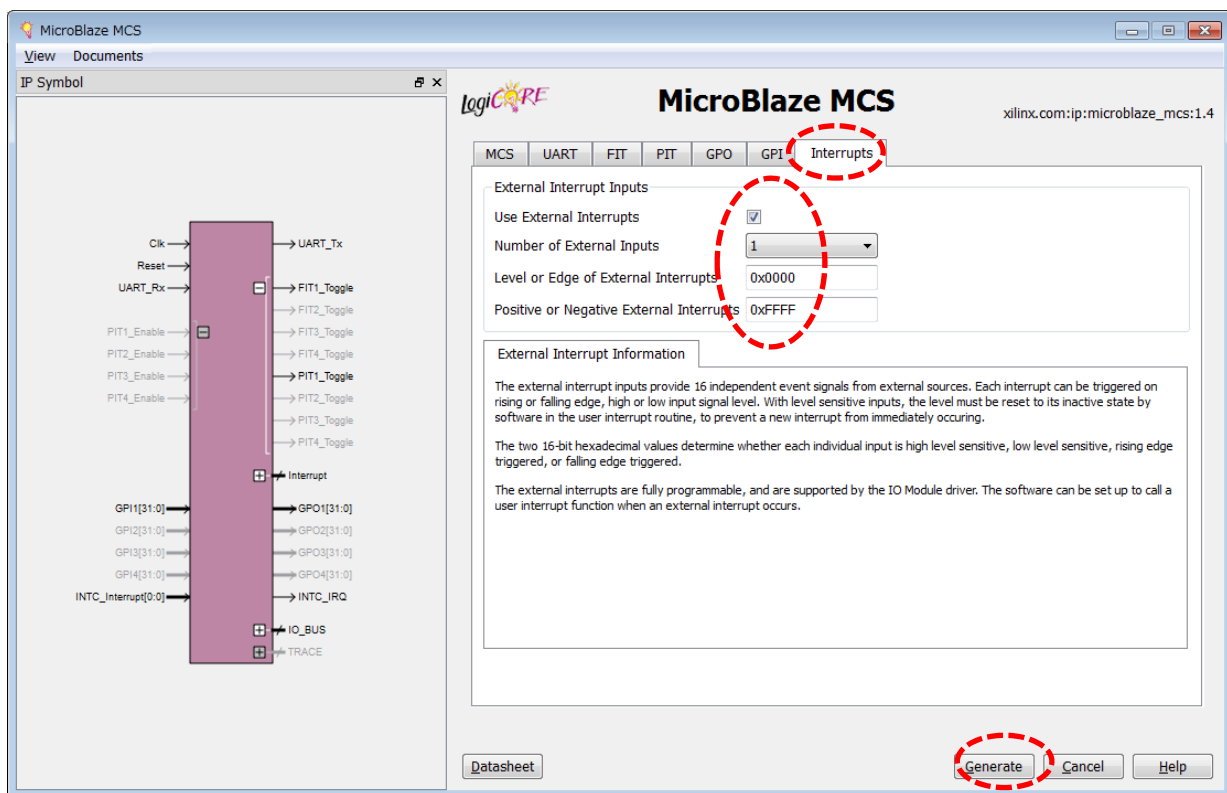


GPO の設定



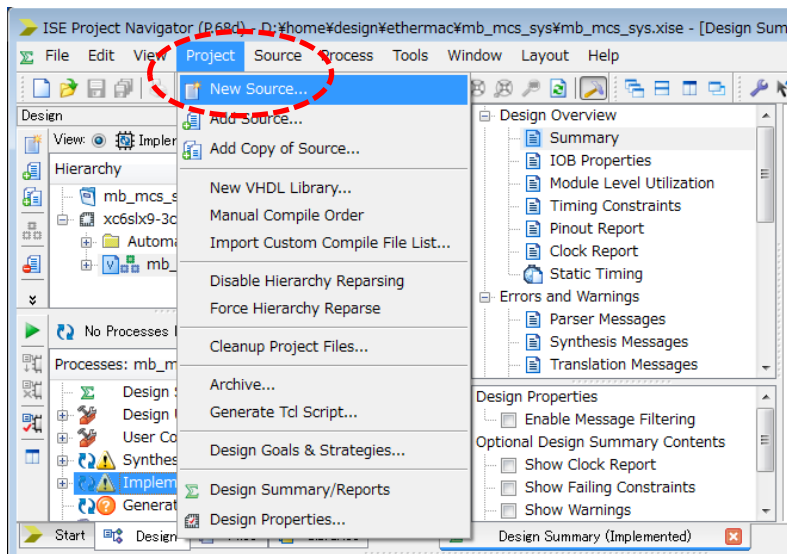


GPI の指定

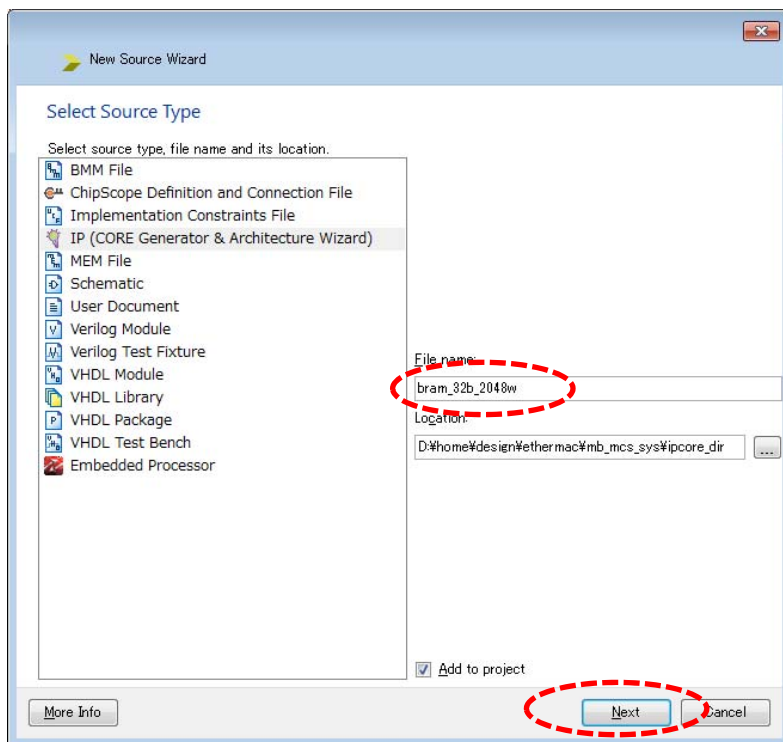


外部割り込みの設定後に Generate をクリックすると MicroBlaze MCS がプロジェクトに追加される



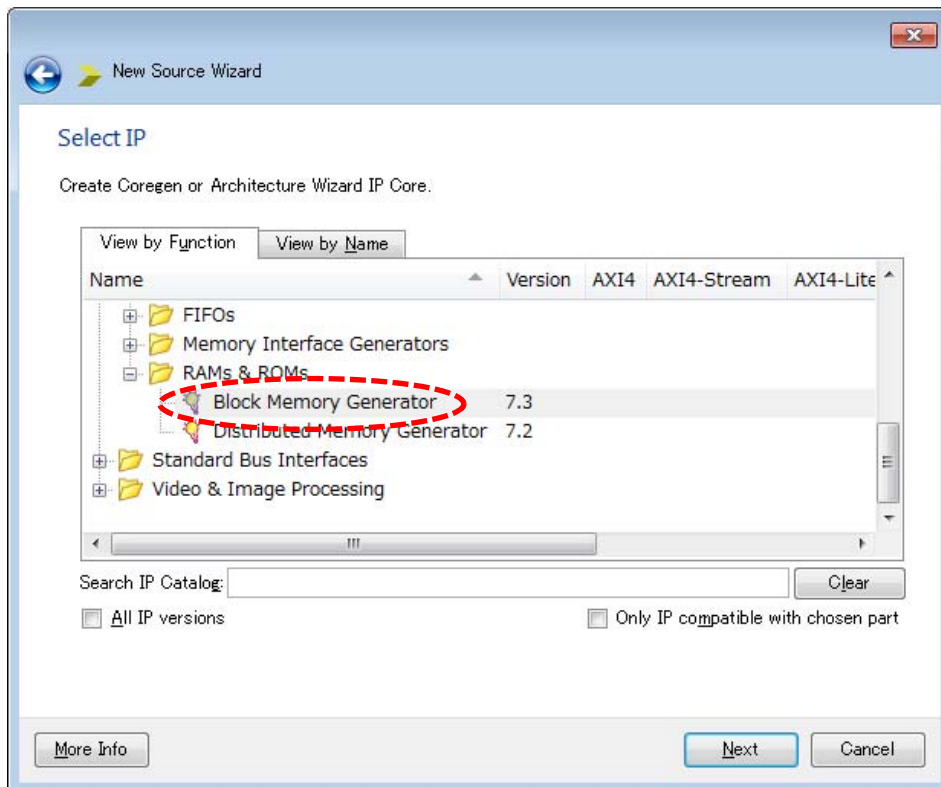


ブロック RAM の作成、Project→New Source を選択

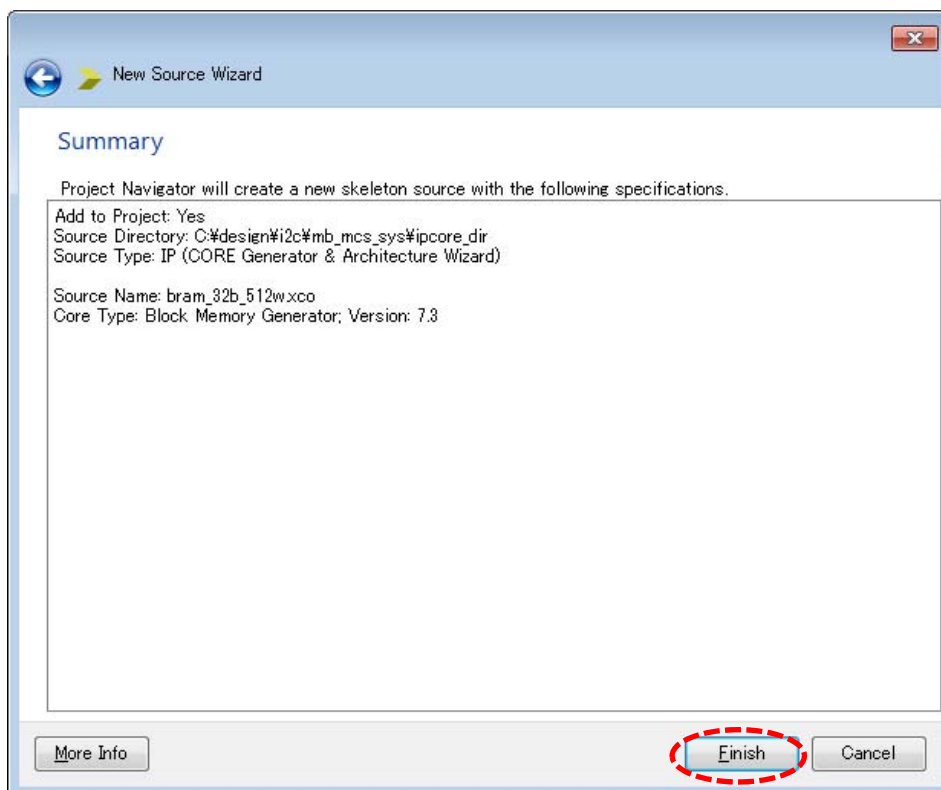


IP (CORE Gener...をクリックして選択、ファイル名に bram\_32b\_2048w を指定、Next をクリック



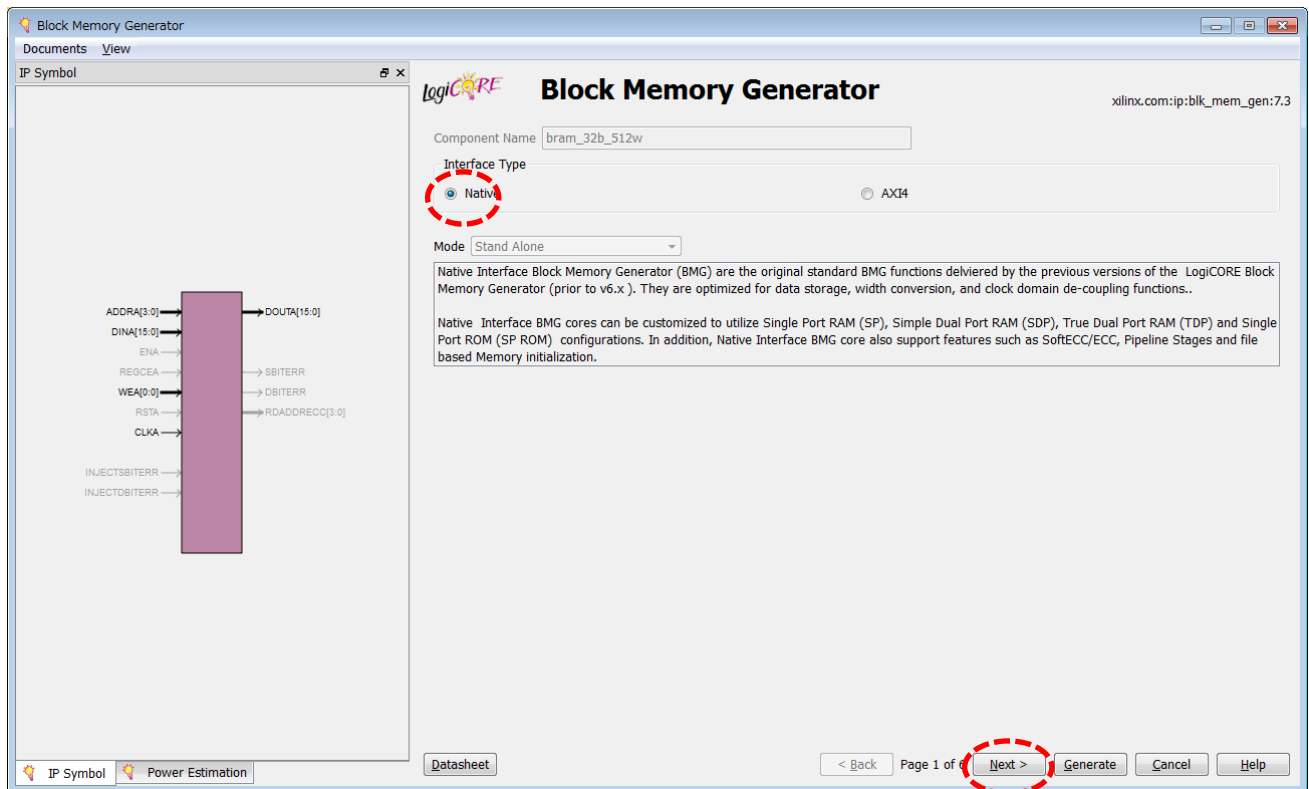


Block Memory Generator を指定

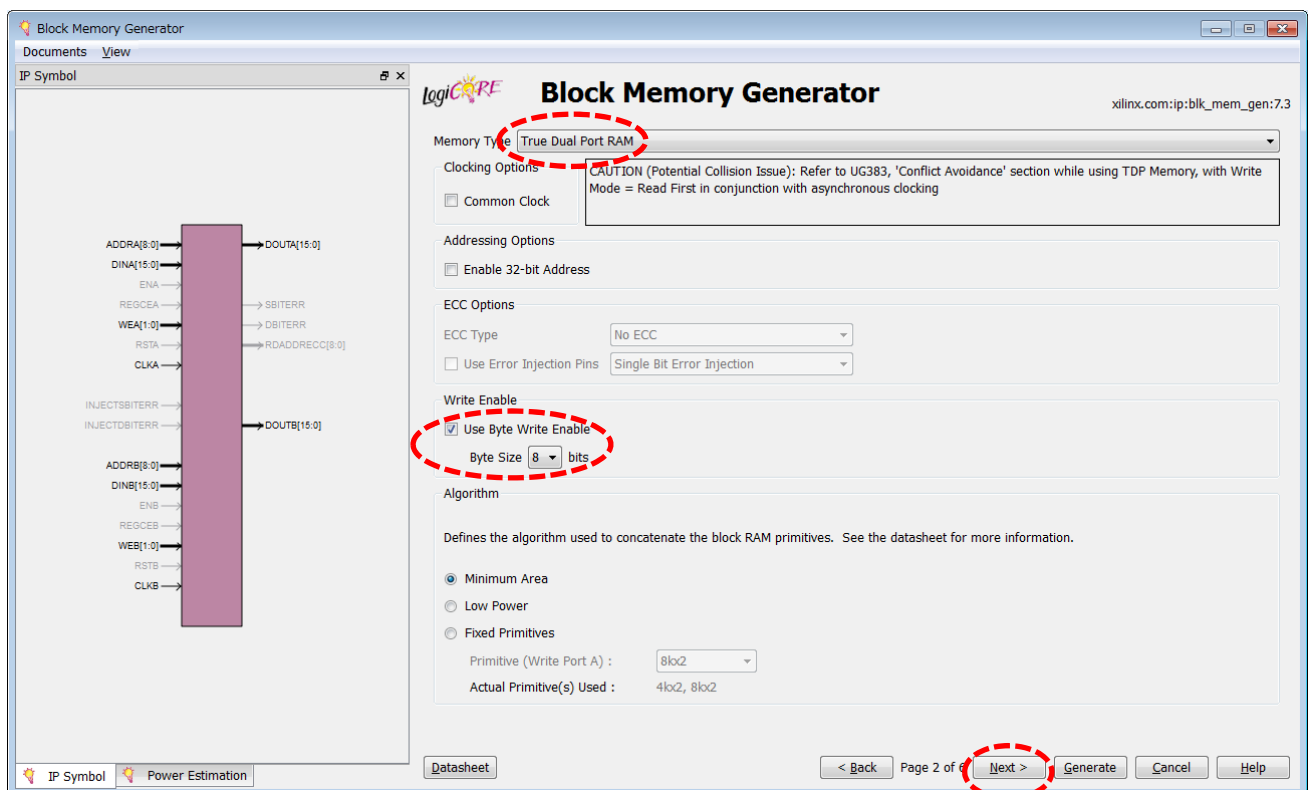


Finish をクリックすると CORE generator が起動



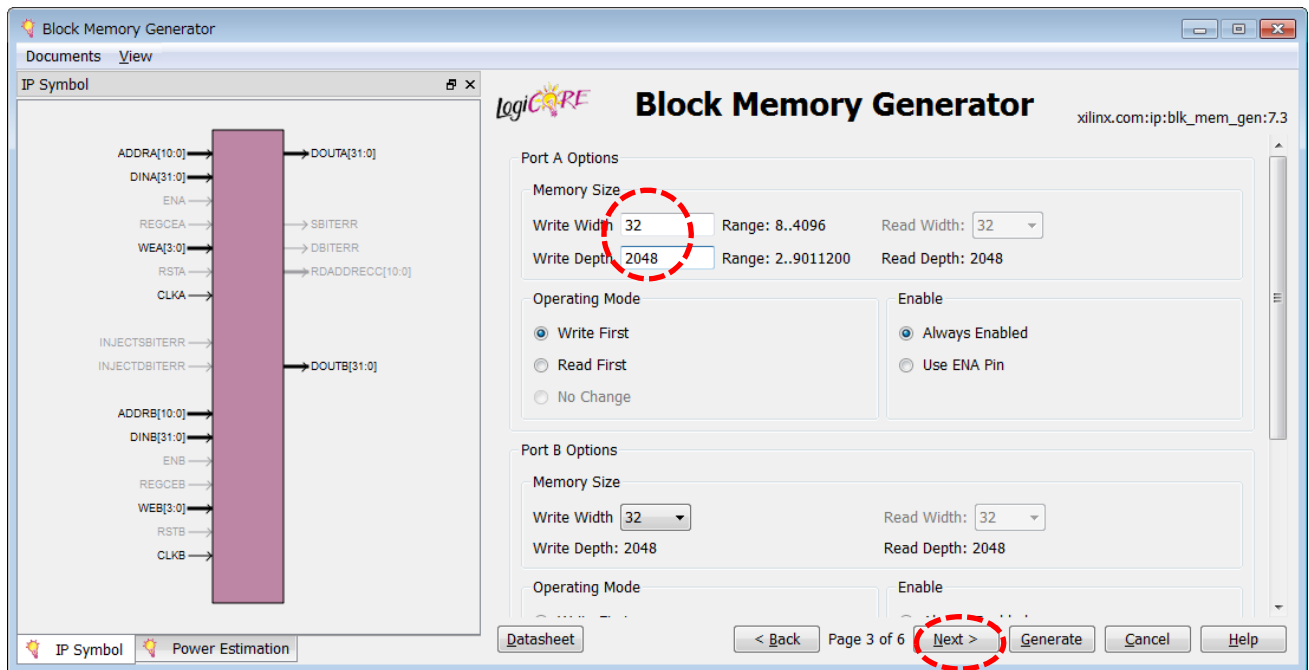


ブロック RAM インターフェースの設定

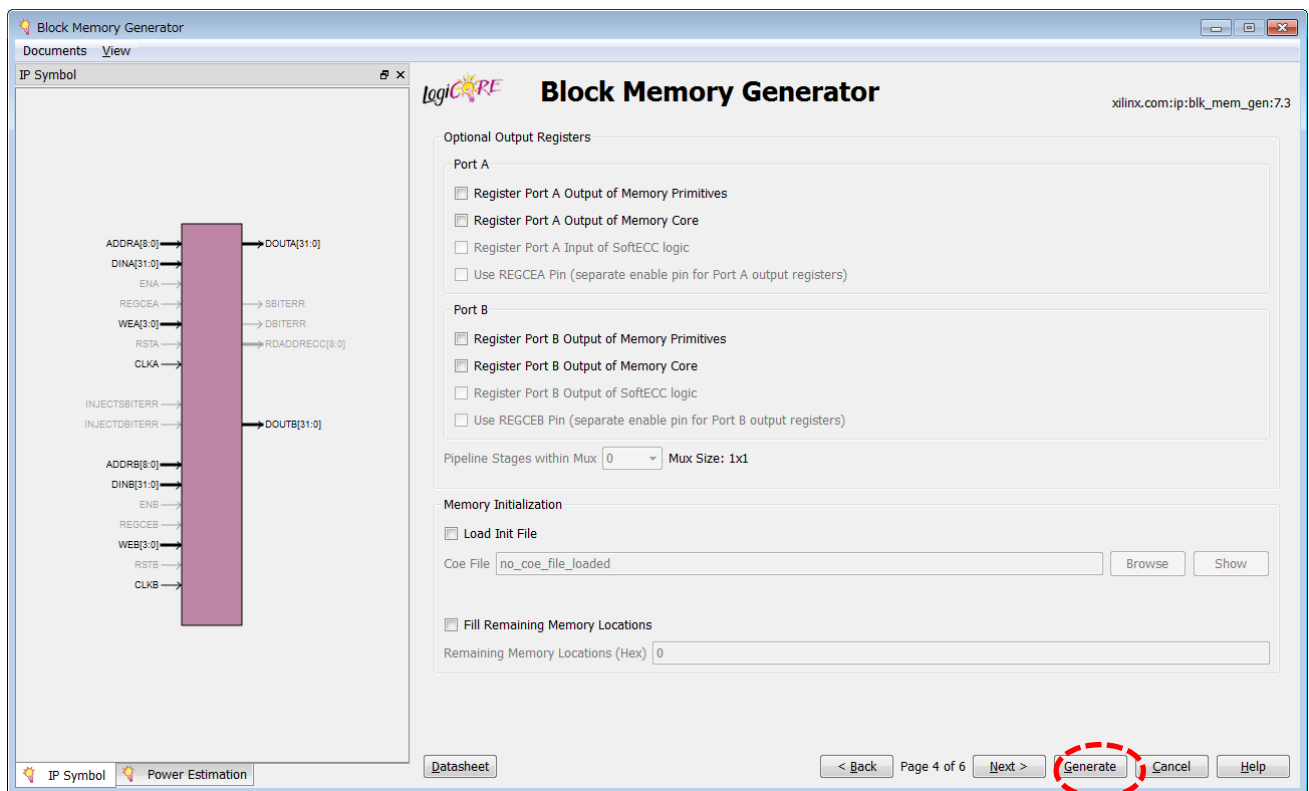


メモリのタイプ指定



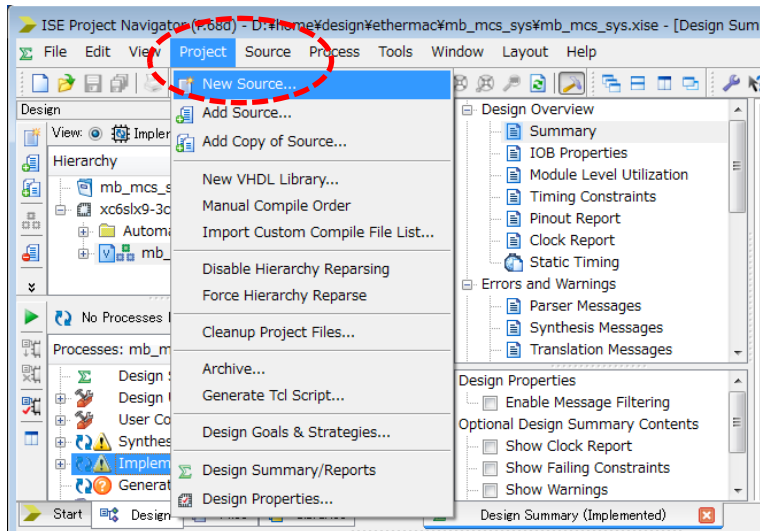


データ幅:32、データ量 2048 の指定

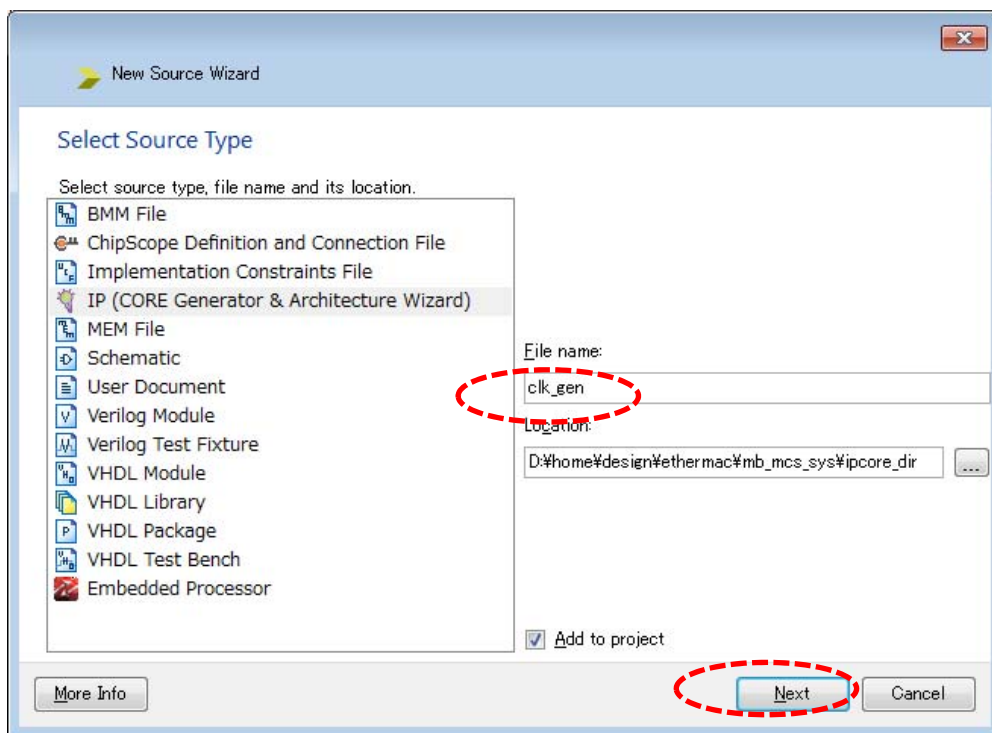


Generate をクリックで作成開始



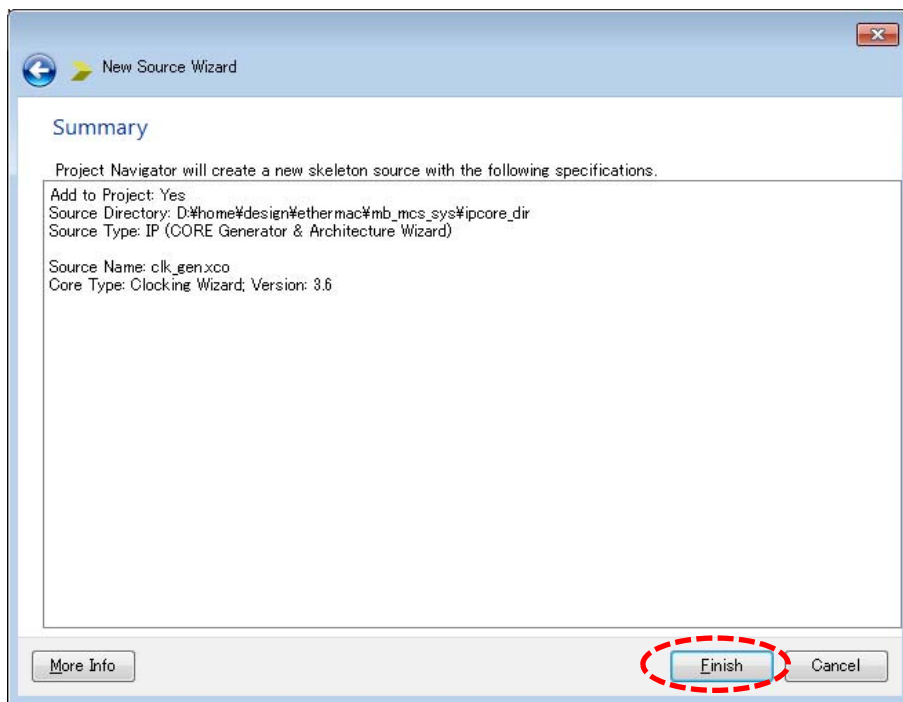


ブロック RAM の作成、Project→New Source を選択

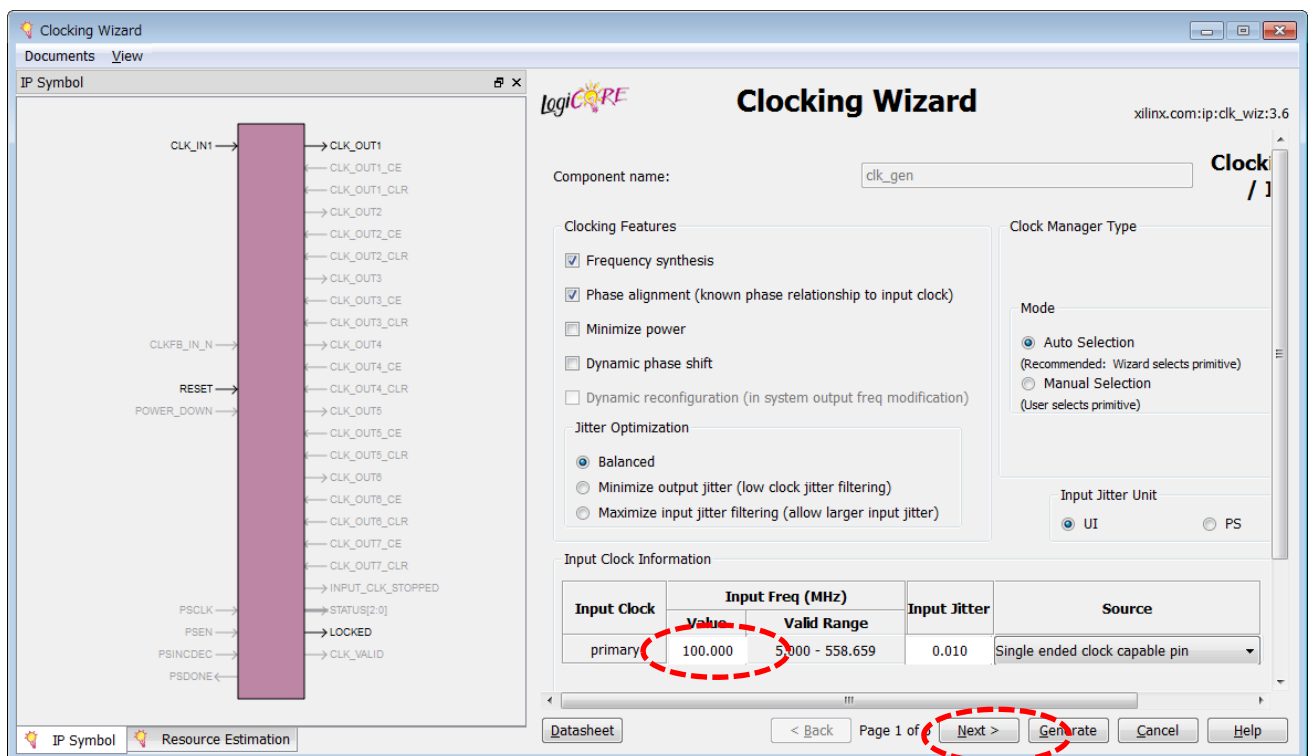


IP (CORE Gener...をクリックして選択、ファイル名に clk\_gen を指定、Next をクリック





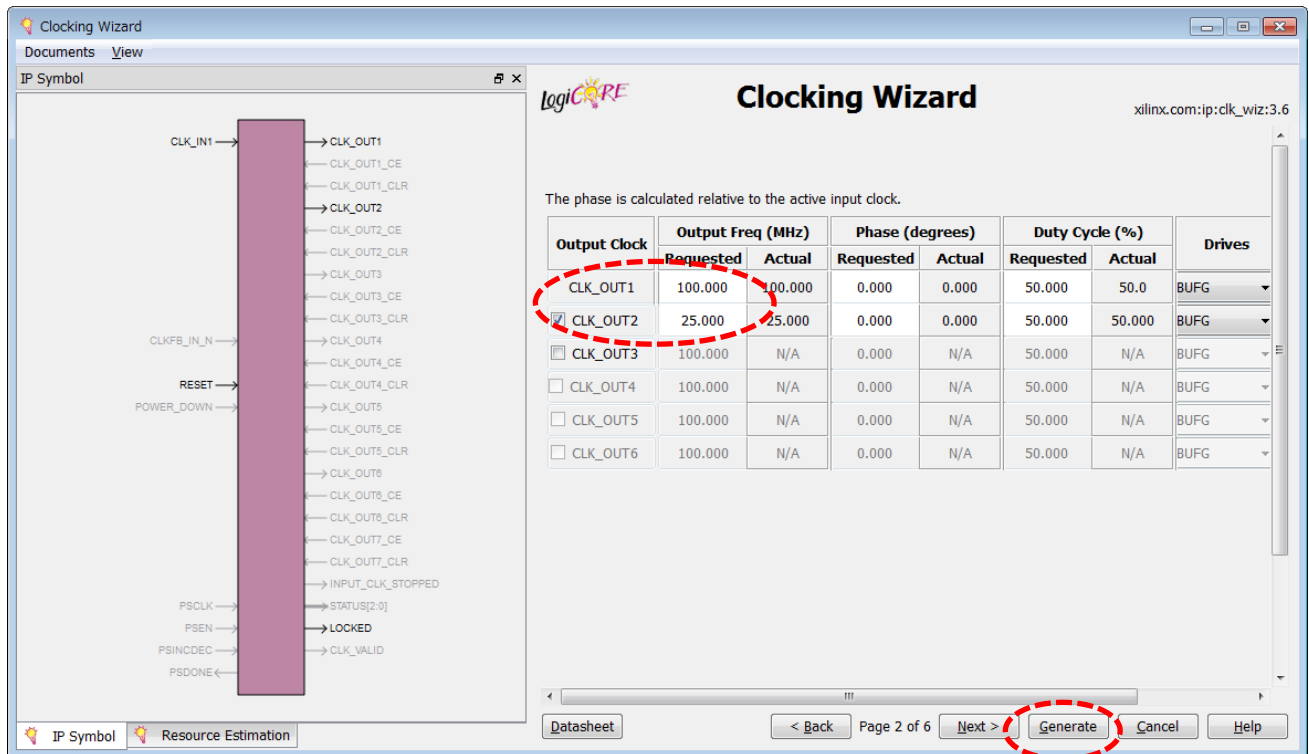
Finish をクリックすると CORE generator が起動



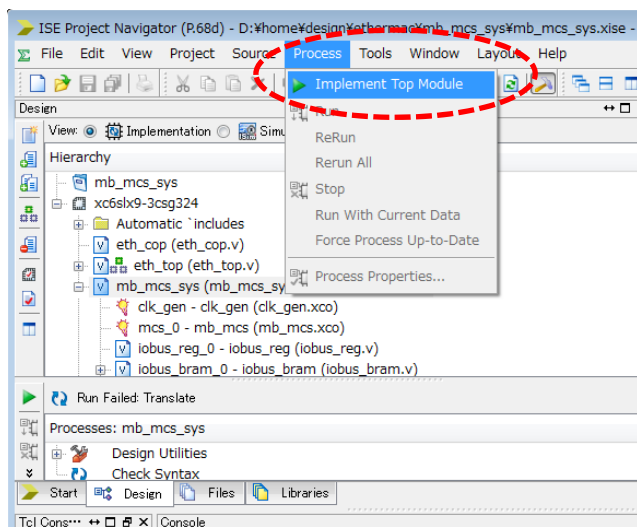
入力クロック周波数 : 100MHz の指定







出力周波数（CLK\_OUT1=100MHz, CLK\_OUT1=25MHz）の設定、Generate をクロックで作成



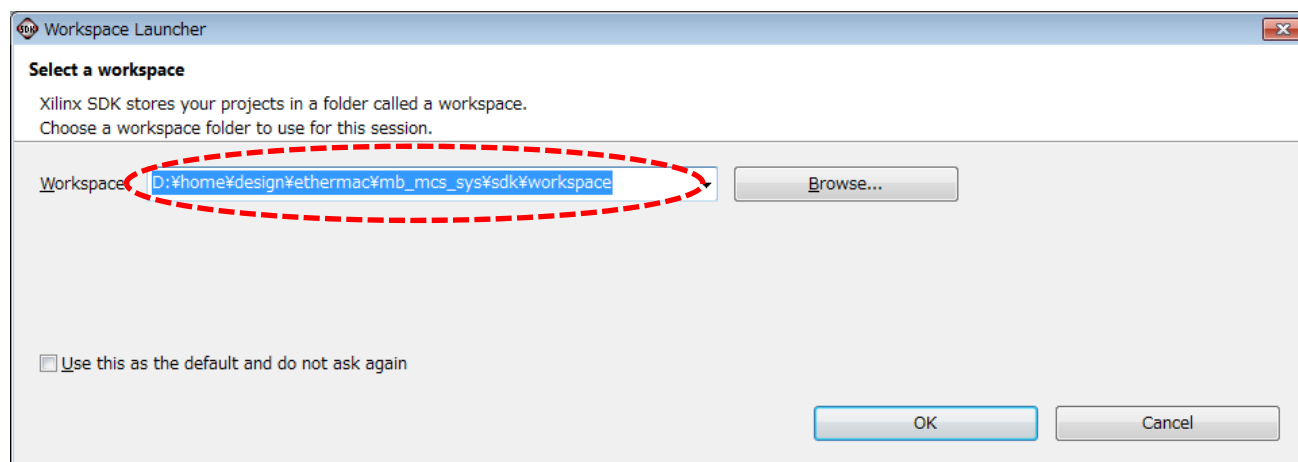
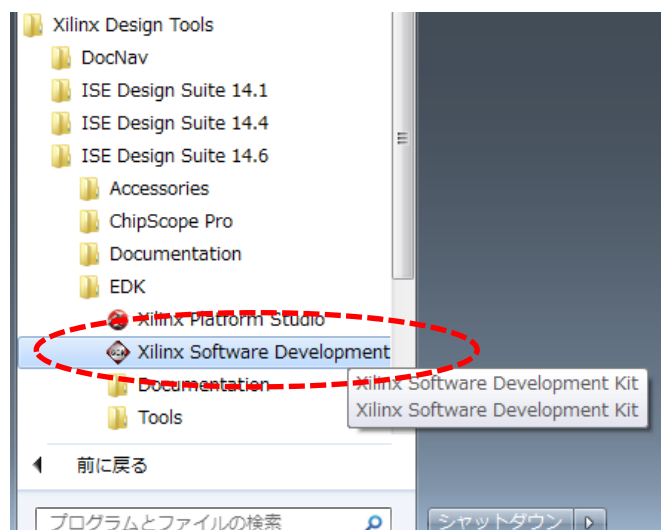
インプリメンテーションの実行、mb\_mcs\_sys を選んで Process→Implement Top Module をクリック



次に Xilinx Software Development Kit(以降 SDK)でソフトウェアを作ります。

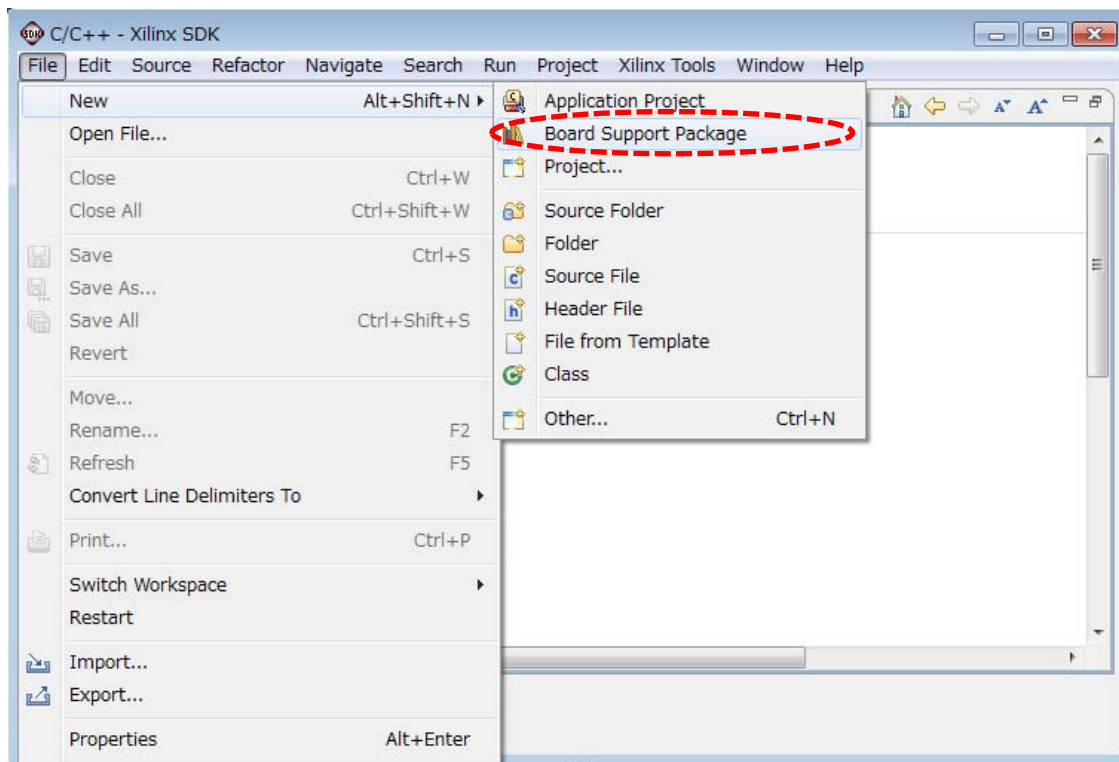
#### ・ SDK の起動

スタートメニューから「Xilinx Design Tools」→「ISE Design Suite 14.6」→「EDK」→「Xilinx Software Development Kit 」を起動してください。



SDK を起動するとワークスペースを指定が要求されます。設計フォルダ/sdk/workspace を設定



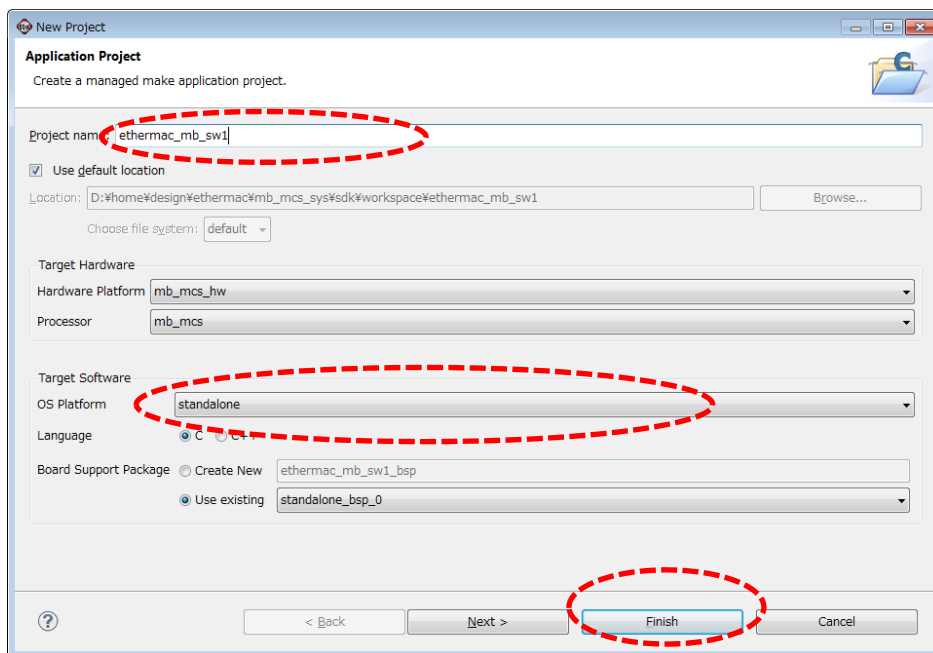


新規のボードサポートパッケージ作成、File→New→Board Support Package



Hardware Platform を定義するか聞かれるので、specify を選択して定義する

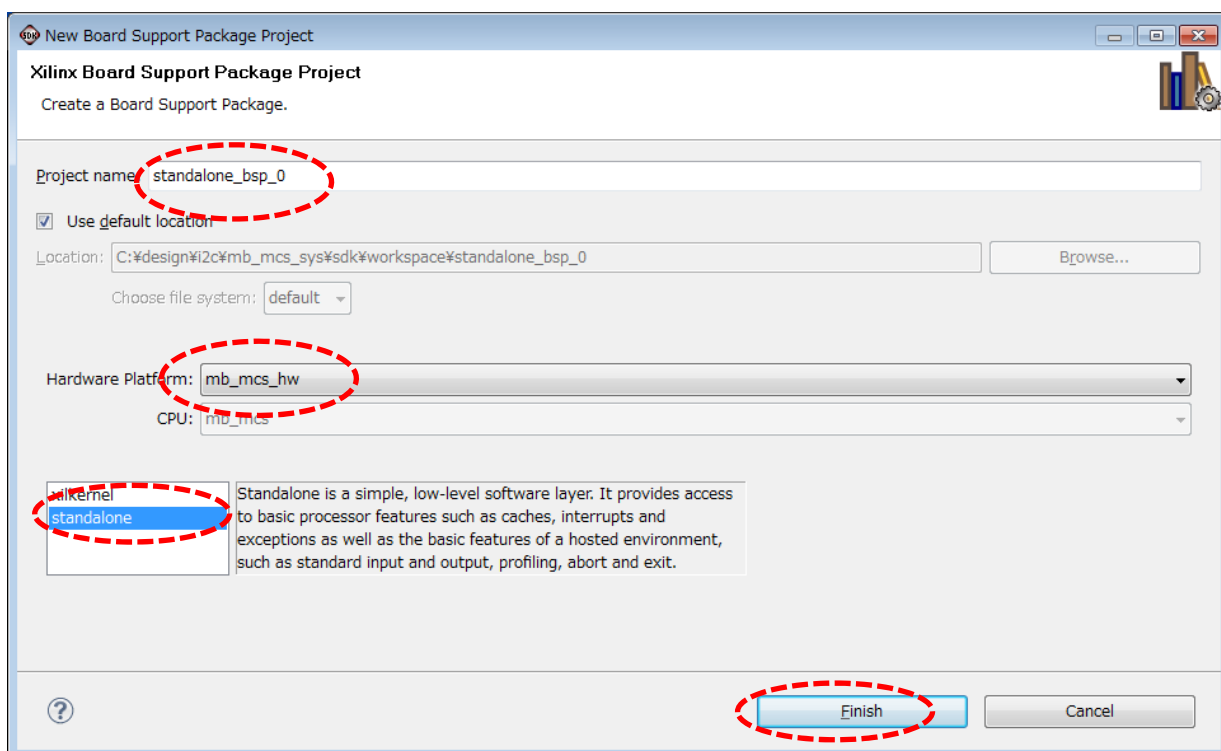




SDK の管理するハードウェアプロジェクト名指定

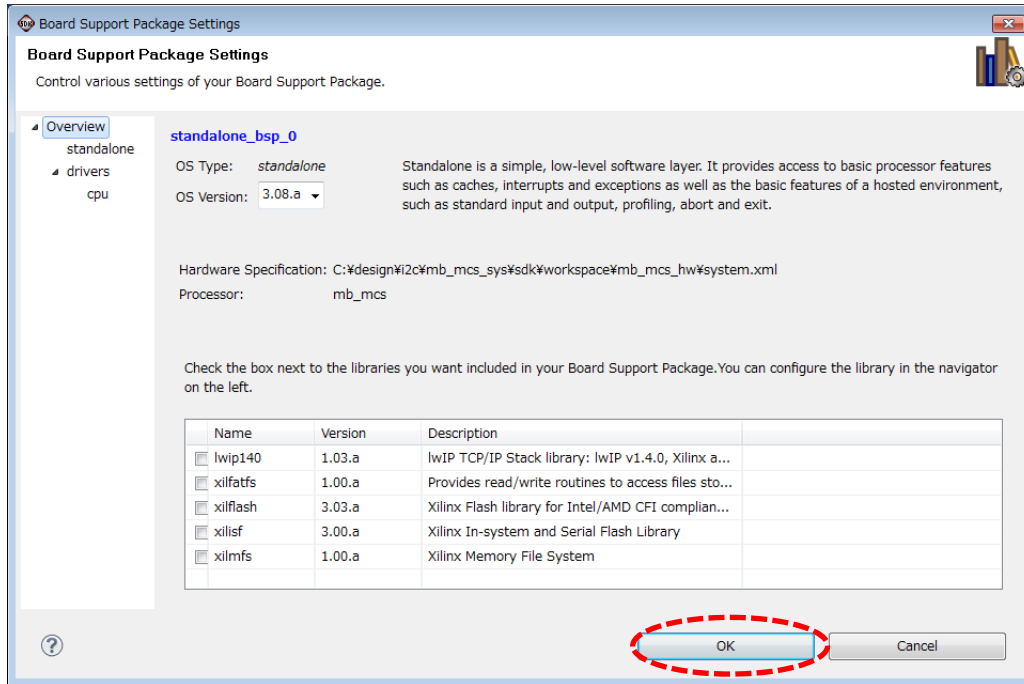
xml ファイル (設計フォルダ/ipcore\_dir/mb\_msc.sdk.xml) 指定

Finish をクリック

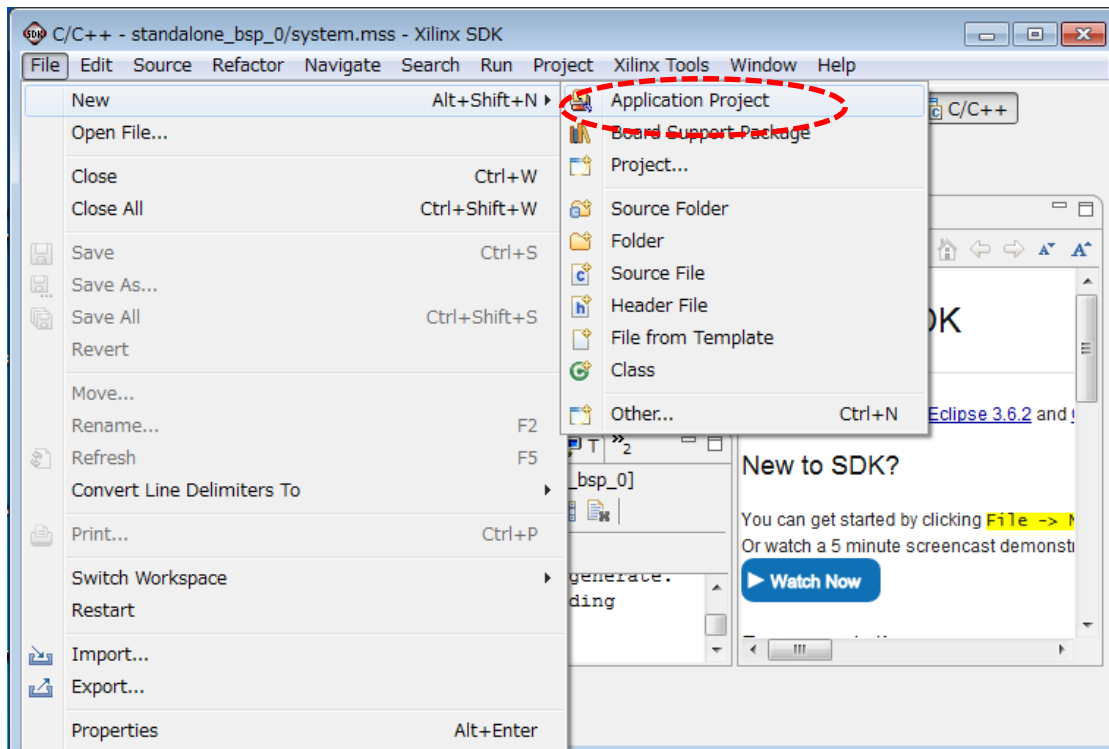


ボードサポートパッケージの定義



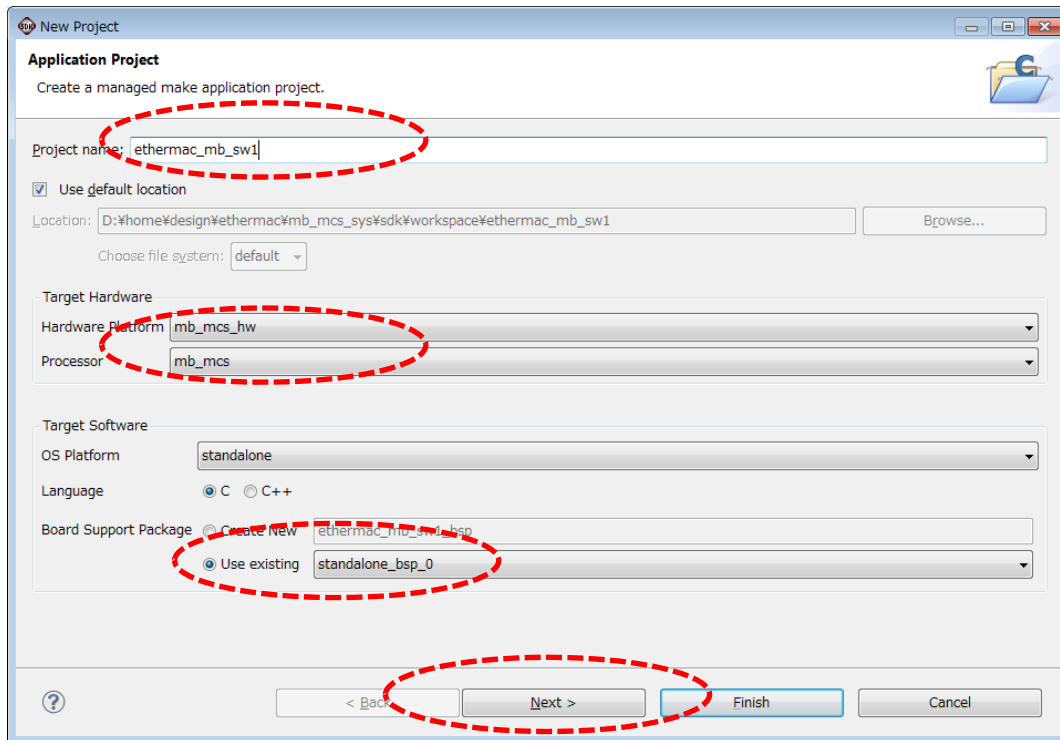


ボードサポートパッケージのオプション定義



新規のソフトウェアプロジェクトの作成





The 'New Project' dialog box is shown with the 'Application Project' tab selected. The 'Project name' field contains 'ethermac\_mb\_sw1'. The 'Use default location' checkbox is checked. The 'Location' field shows a path starting with 'D:\home\design\ethermac\mb\_mcs\_sys\sd\workspace\ethermac\_mb\_sw1'. The 'Choose file system' dropdown is set to 'default'. Under 'Target Hardware', 'Hardware Platform' is 'mb\_mcs\_hw' and 'Processor' is 'mb\_mcs'. Under 'Target Software', 'OS Platform' is 'standalone', 'Language' is 'C', and 'Board Support Package' is 'standalone\_bsp\_0'. The 'Finish' button is highlighted.

**Application Project**  
Create a managed make application project.

Project name: ethermac\_mb\_sw1

☒ Use default location  
Location: D:\home\design\ethermac\mb\_mcs\_sys\sd\workspace\ethermac\_mb\_sw1 Browse...

Choose file system: default

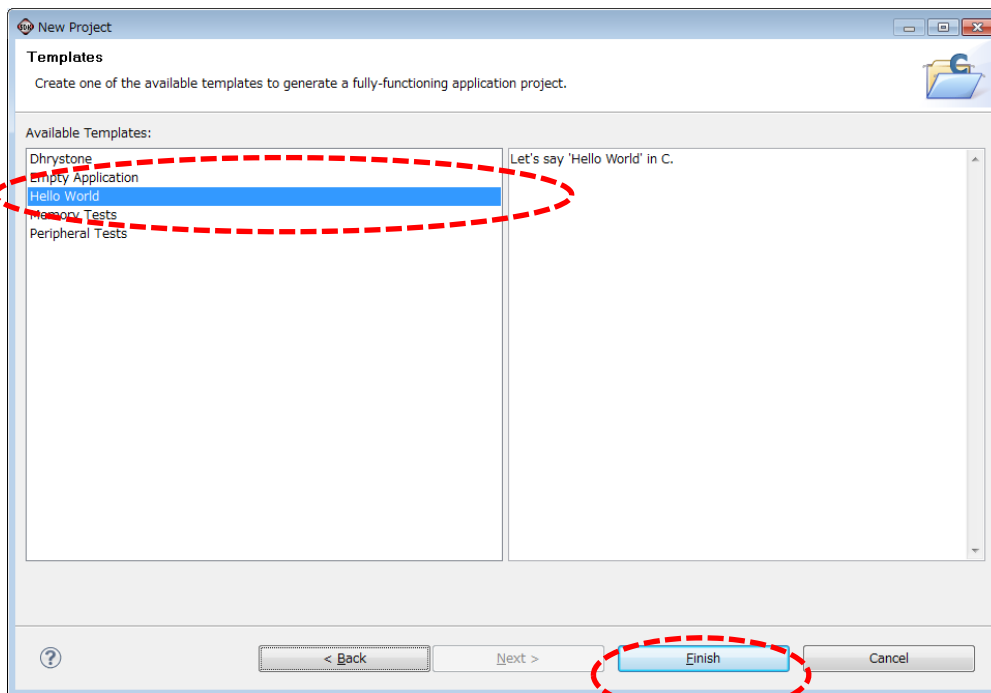
**Target Hardware**  
Hardware Platform: mb\_mcs\_hw  
Processor: mb\_mcs

**Target Software**  
OS Platform: standalone  
Language: ☒ C ☐ C++  
Board Support Package: ☒ Use existing standalone\_bsp\_0

< Back Next > Finish Cancel

ソフトウェアプロジェクト名指定:ethermac\_sw1

Next をクリック



The 'New Project' dialog box is shown with the 'Templates' tab selected. The 'Available Templates' list on the left includes 'Dhrystone', 'Empty Application', 'Hello World', 'Memory Tests', and 'Peripheral Tests'. 'Hello World' is selected and highlighted in blue. The 'Finish' button is highlighted.

**Templates**  
Create one of the available templates to generate a fully-functioning application project.

Available Templates:

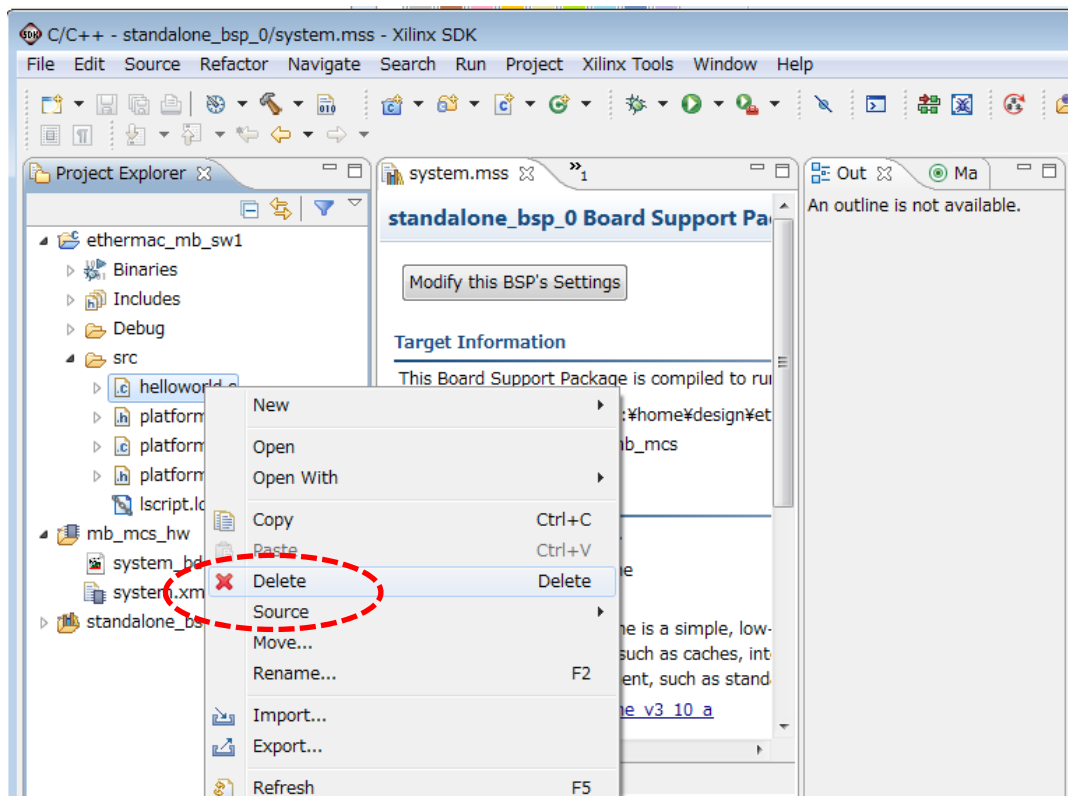
- Dhrystone
- Empty Application
- Hello World
- Memory Tests
- Peripheral Tests

Let's say 'Hello World' in C.

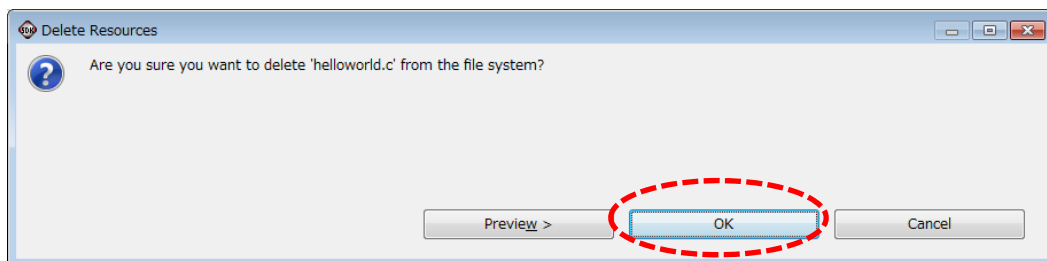
< Back Next > Finish Cancel

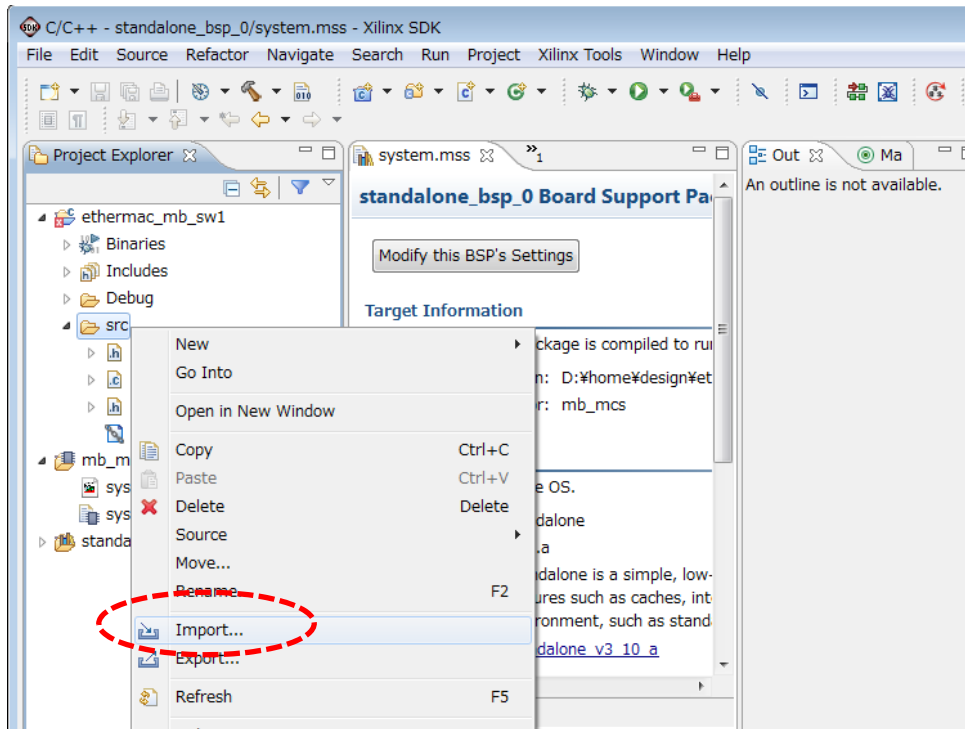
テンプレートに Hello World 選択後、Finish をクリックでソフトウェアプロジェクトが作成される



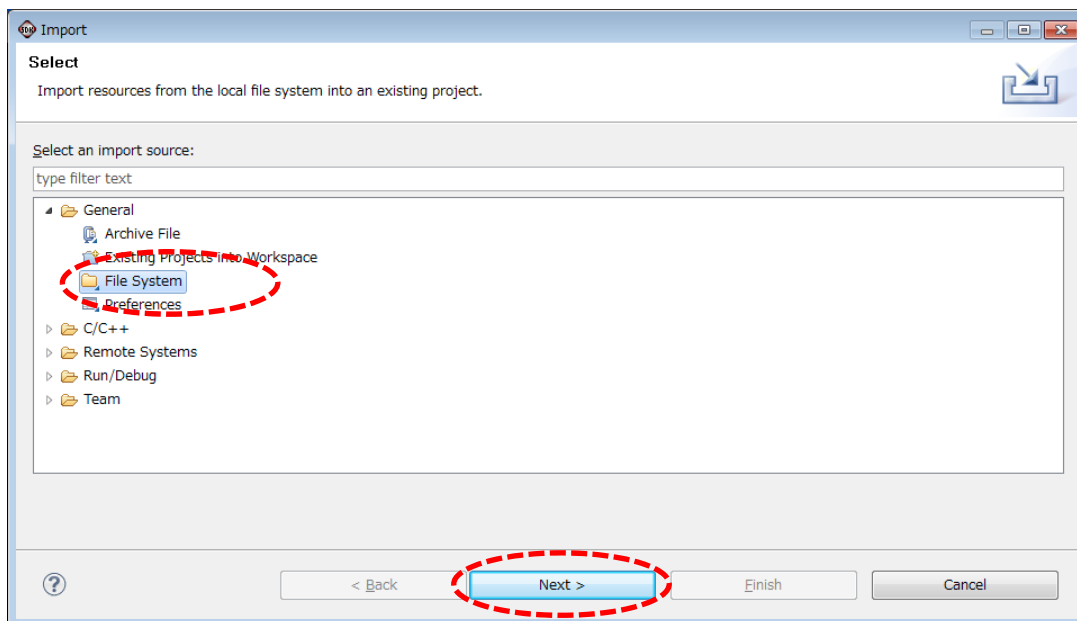


helloworld.c を削除





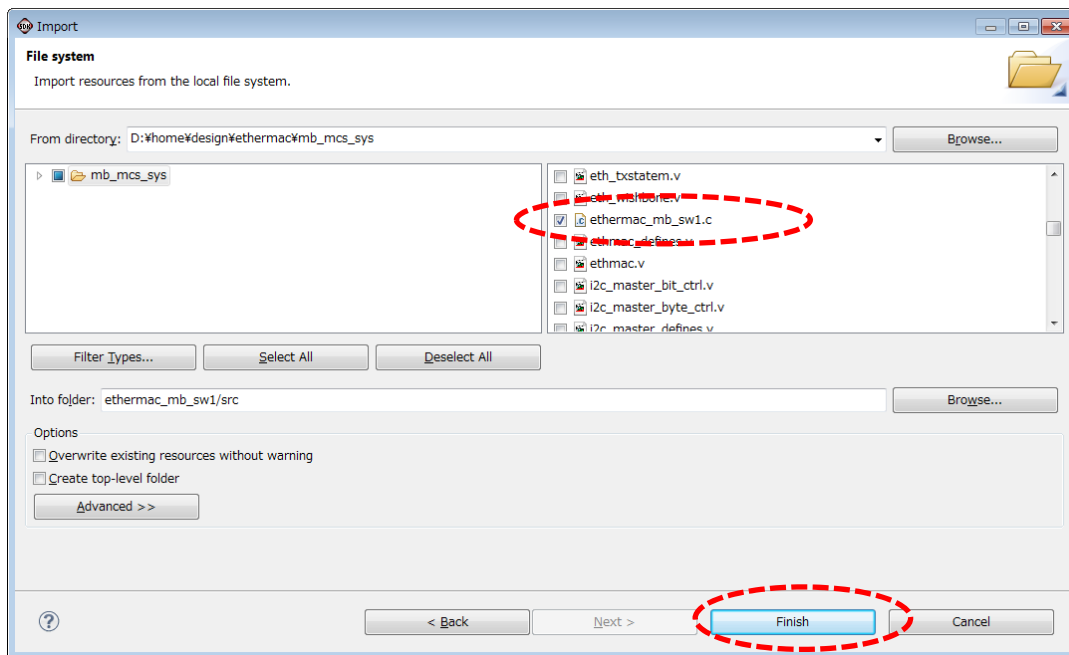
C ソースコードのインポート、ethernac\_mb\_sw1 の src の上で右クリック、import 選択



General→File System 選択





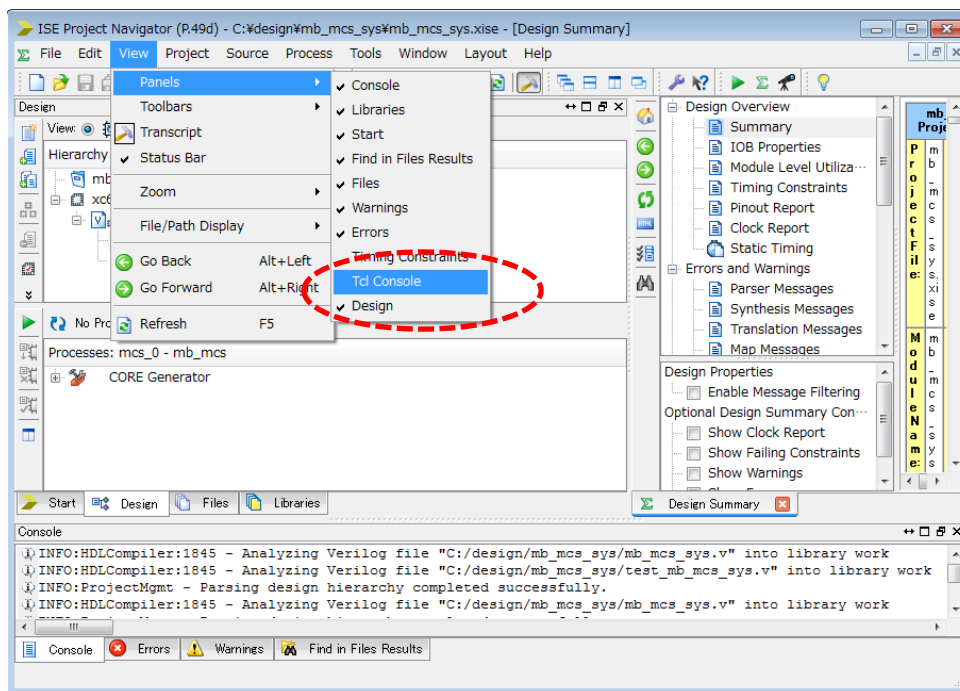


解凍データの ethermac\_mb\_sw1.c を選択

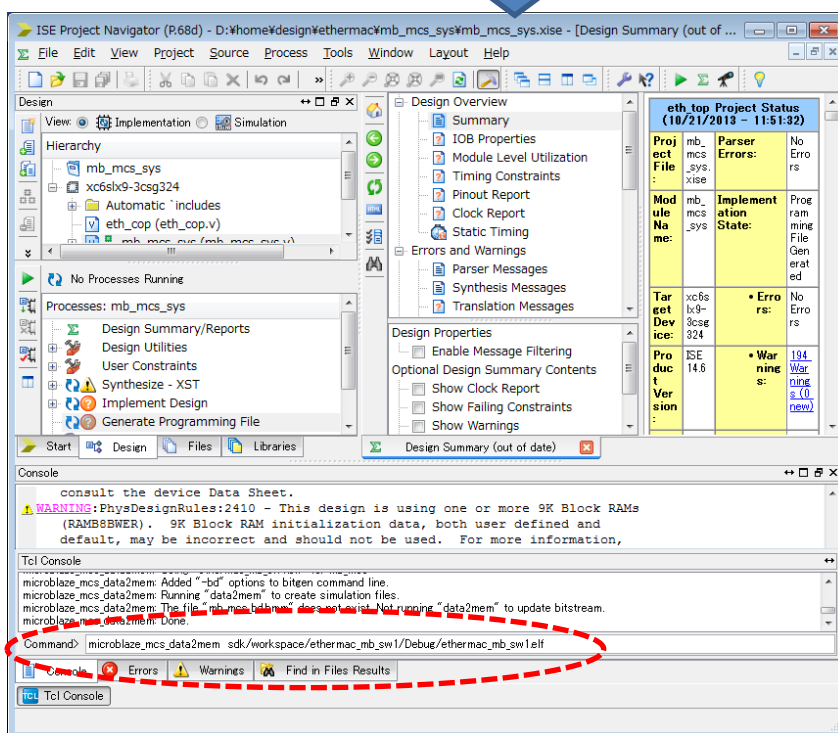
ビルドが実行され、

実行ソフトウェアのファイル（sdk/workspace/ ethermac\_mb\_sw1/Debug/ ethermac\_mb\_sw1.elf）が作られる。





Project Navigator に戻り、Tcl コマンドを使用できるように Tcl Console を表示する  
View→Panels→Tcl Console

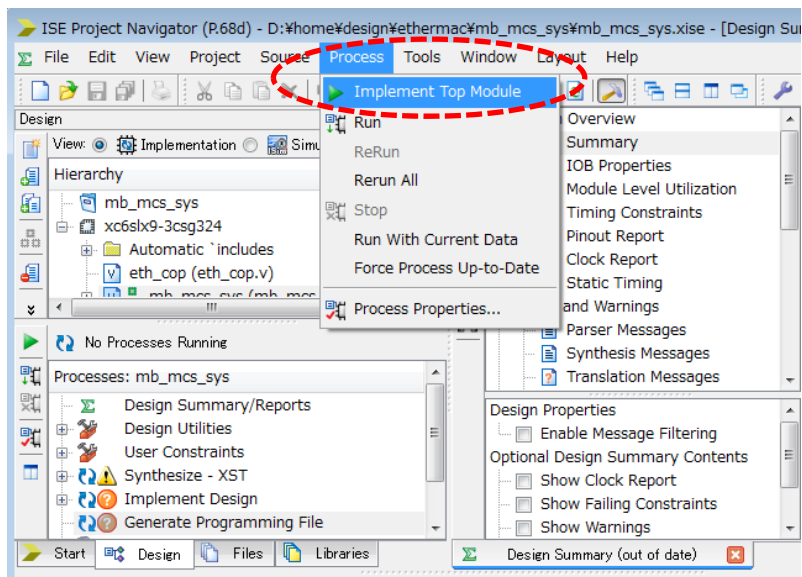


Tcl コマンドを使って実行ソフトウェアのファイルをMicroBlazeMCSのメモリの初期値定義ファイルに変換する。

Tcl コマンド

```
source ipcore_dir/microblaze_mcs_setup.tcl[Enter]
```

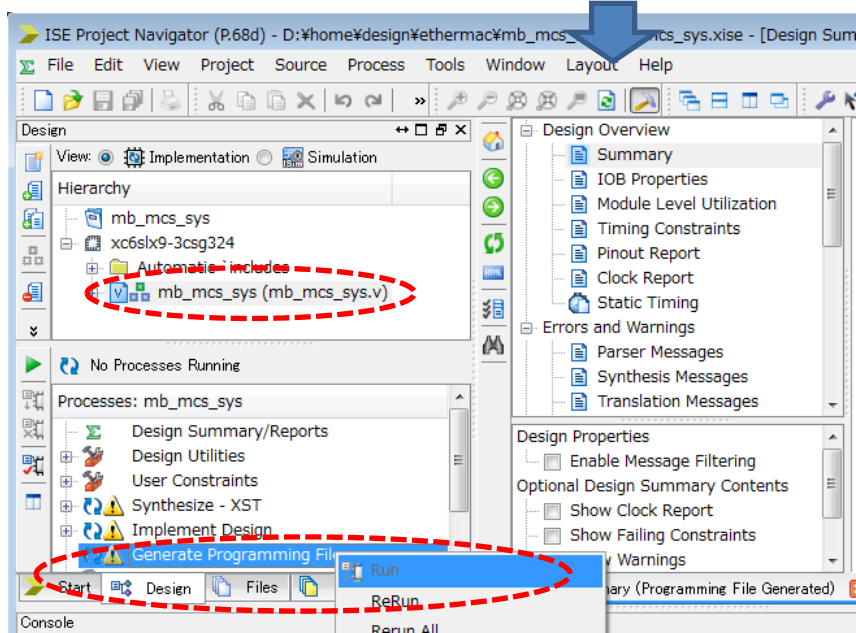
```
microblaze_mcs_data2mem sdk/workspace/ethernac_mb_sw1/Debug/ethernac_mb_sw1.elf[Enter]
```



インプリメンテーションの実施、Process→Implement Top Module



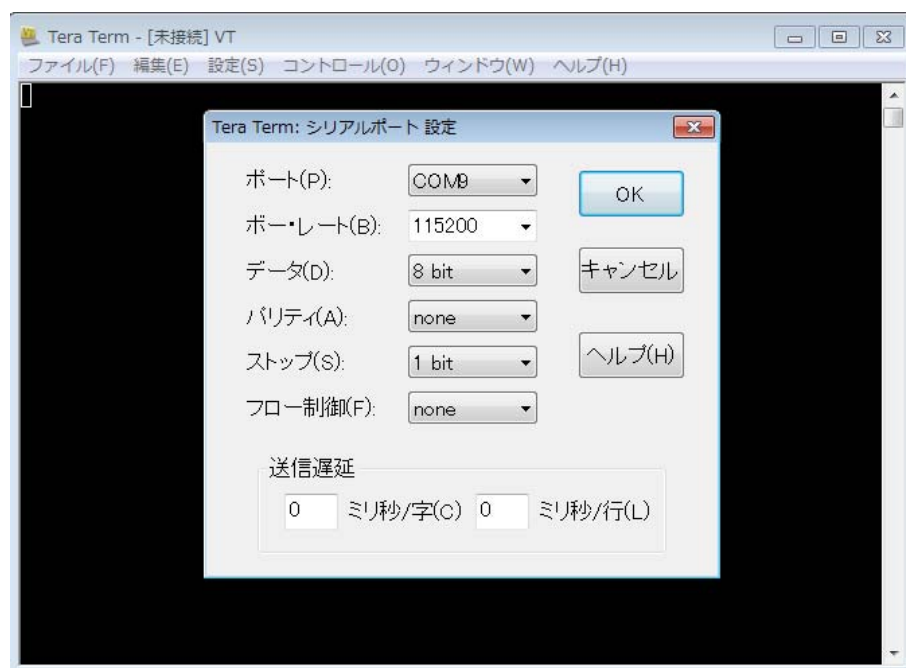
FPGA へ書き込む mb\_mcs\_sys.bit 作成



実機の動作確認をします。

LX9マイクロボードのプログラミング用USBとUART用USBをそれぞれPCのUSBポートに接続で接続します。

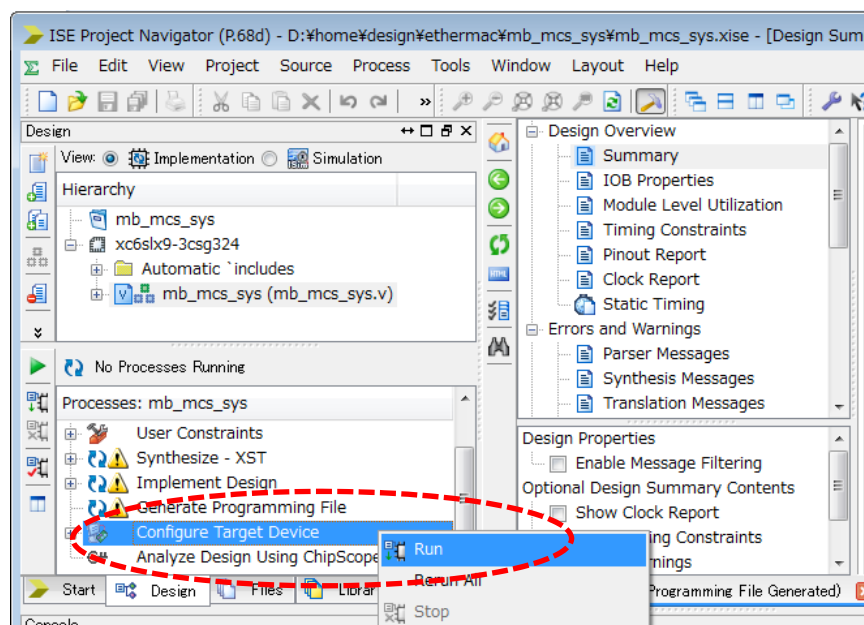
LX9 マイクロボードとディップスイッチ 1-1 (信号名 sw0) を 1 にして Ethernet 通信を折返しモードする。

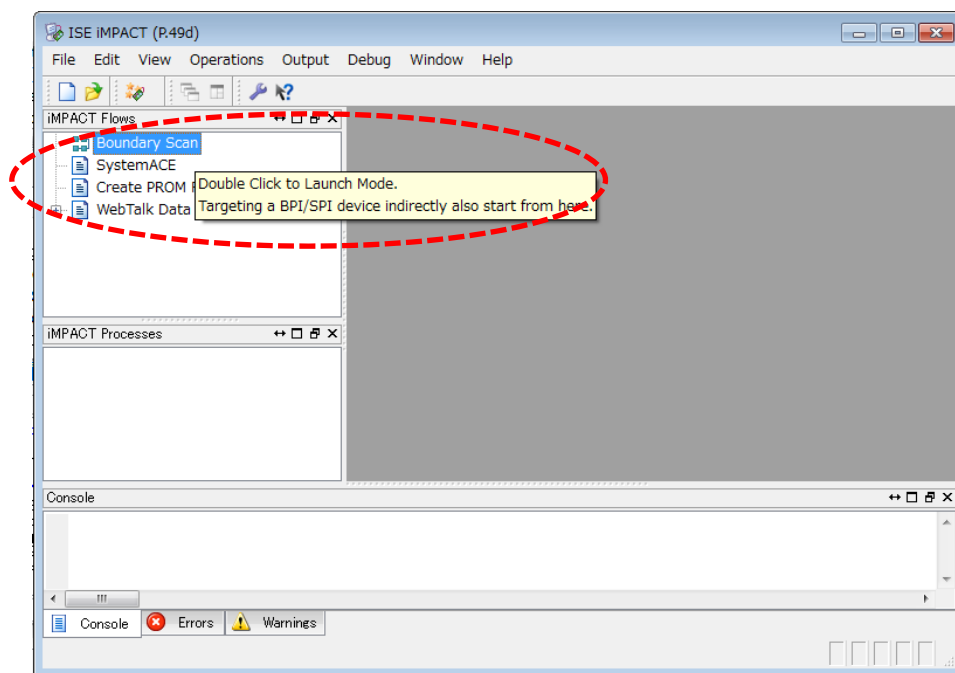


ターミナルソフトを立ち上げる。ポート番号は自分 PC で割り当てられた番号を使う

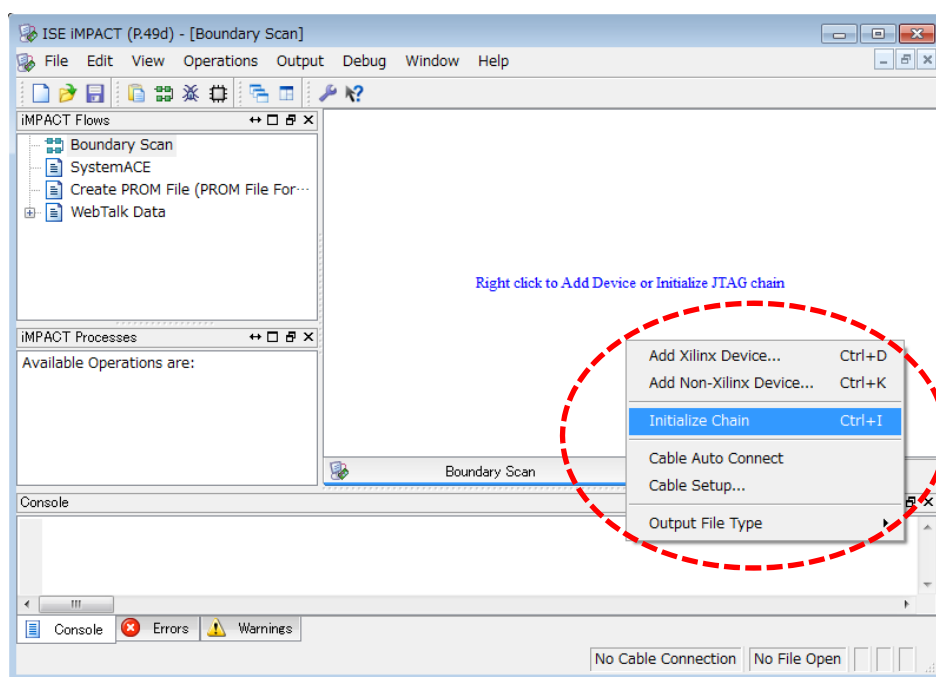


projectNavigator で iMPACT を起動



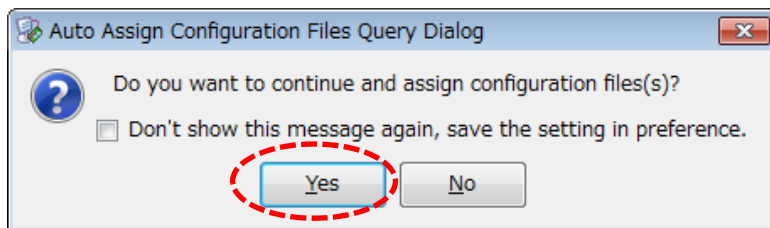


BoundaryScan モードにする

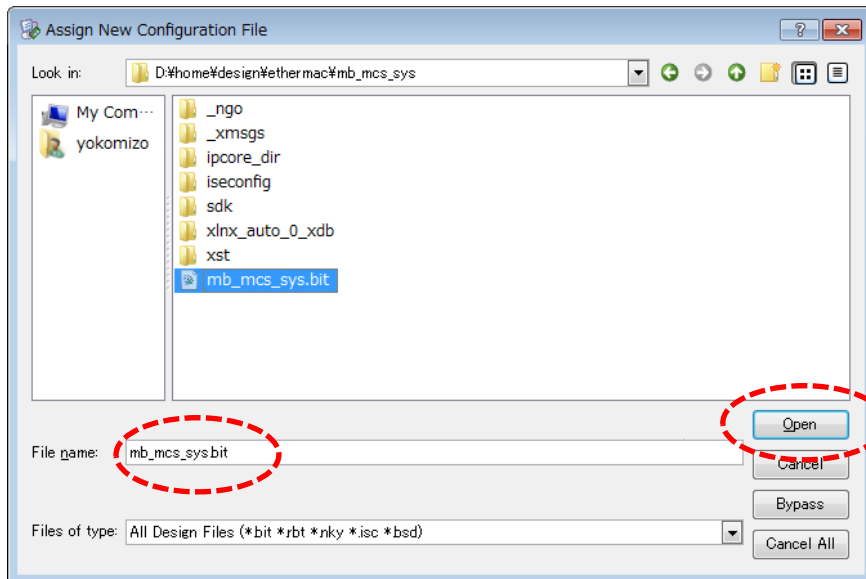


BoundaryScan のウィンドをマウス右ボタン押して、Initialize Chain 選択して FPAG を検出する

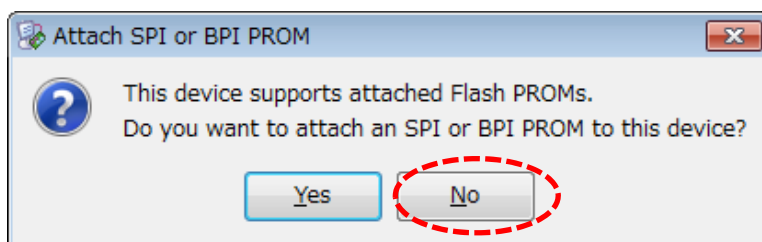




Yes で FPGA に書き込むファイルを指定する

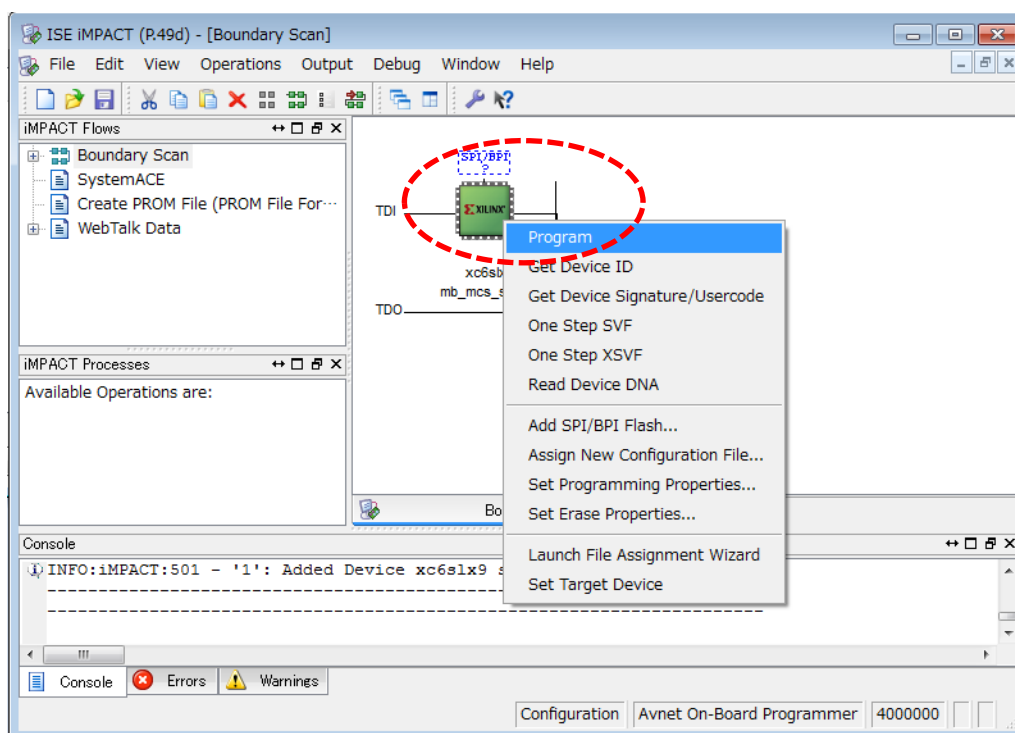
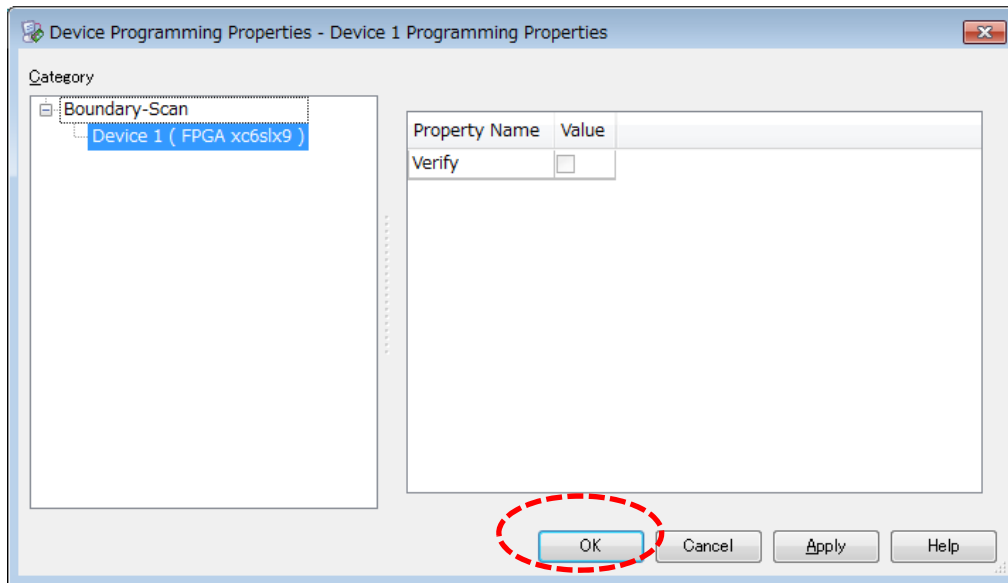


mb\_mcs\_sys.bit を指定



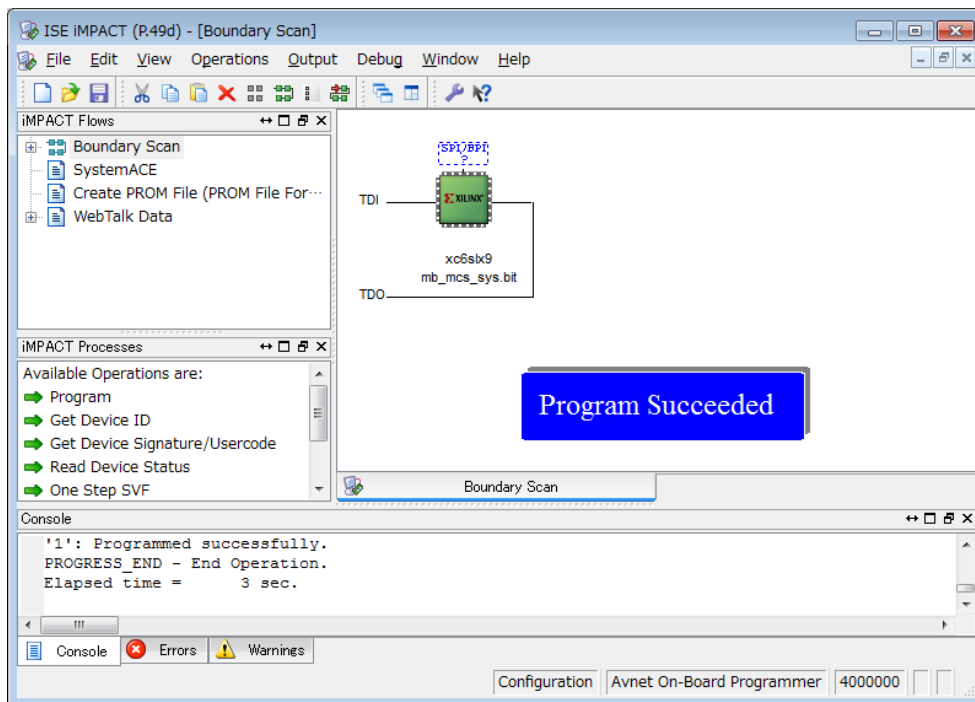
PROM データは使わないので No を選択





FPGA へのプログラミング実行、デバイス上でマウス右ボタンを押して Program 選択

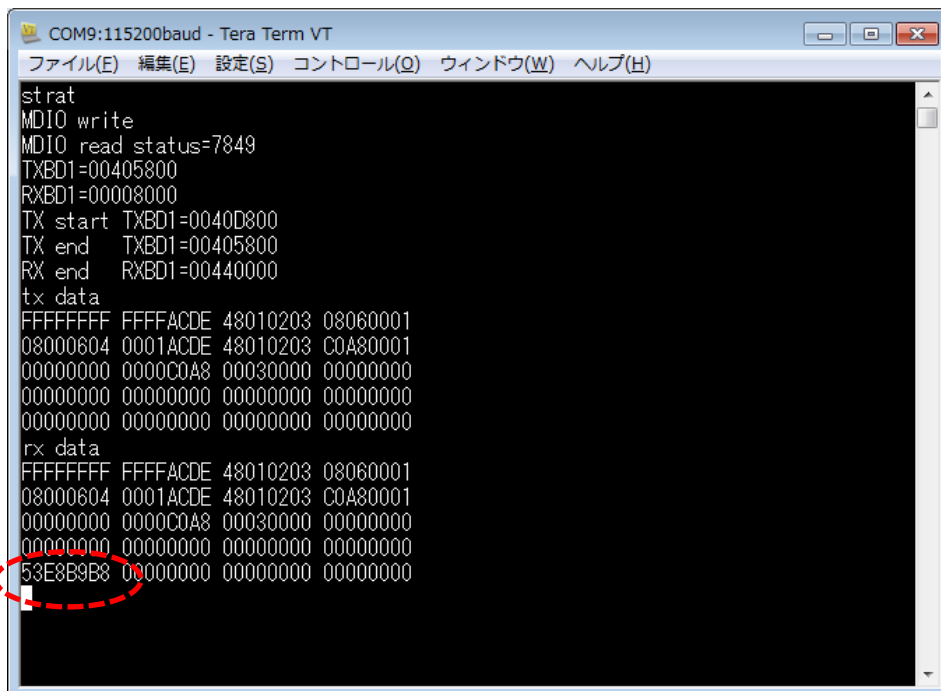




Program Succeeded と表示されればプログラミング完了、LX9 マイクロボードが動作します。



ターミナルソフトに通信結果が表示されます。



折り返しモード[ディップスイッチ 1-1 が1]なので、送信データと受信データが FCS を除いて一致していれば OK。



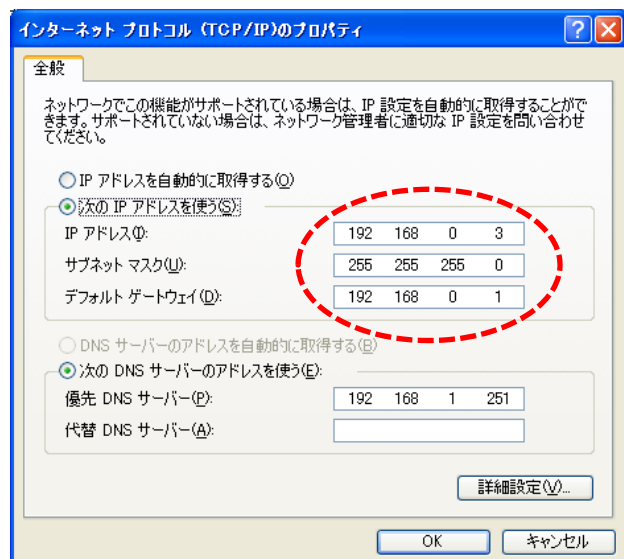


次にEthernetでの動作確認を行います。

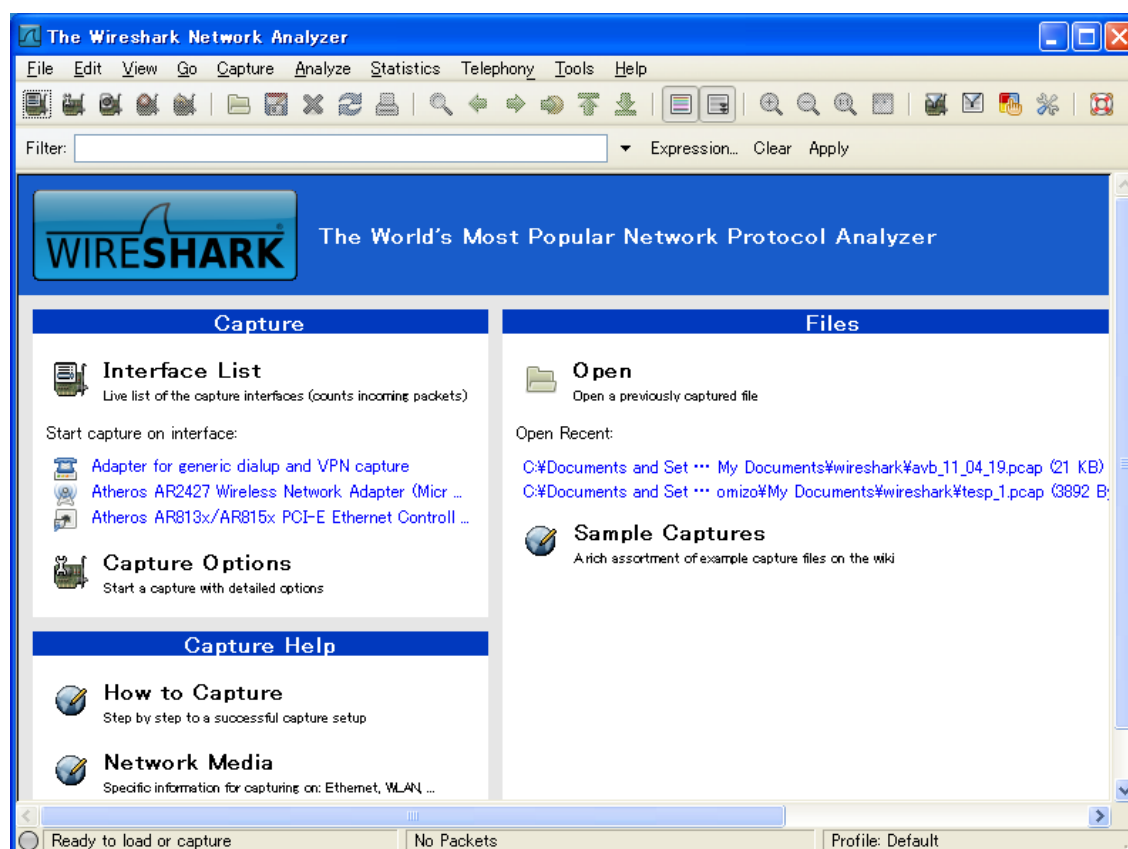
LX9 マイクロボードとディップスイッチ 1-1 (信号名 sw0) を 0 にして通常 Ethernet 通信モードする。

LX9 マイクロボードと PC を LAN ケーブルで接続する。

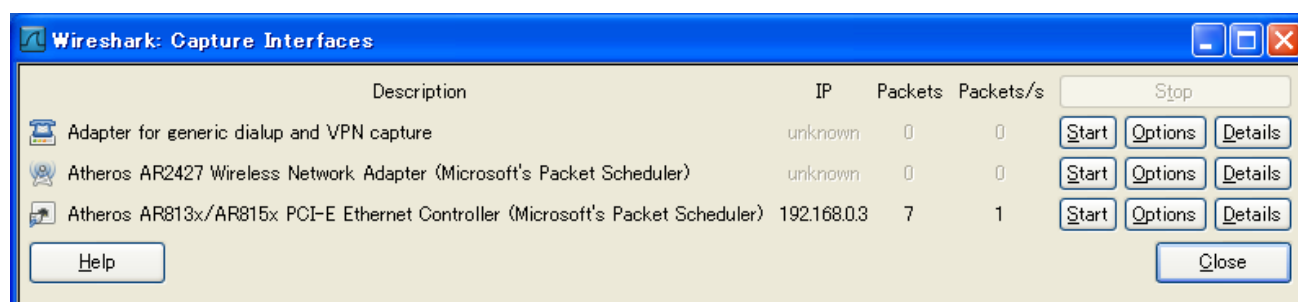
PC のネットワーク設定を固定 IP アドレスに変更



wireshark の起動

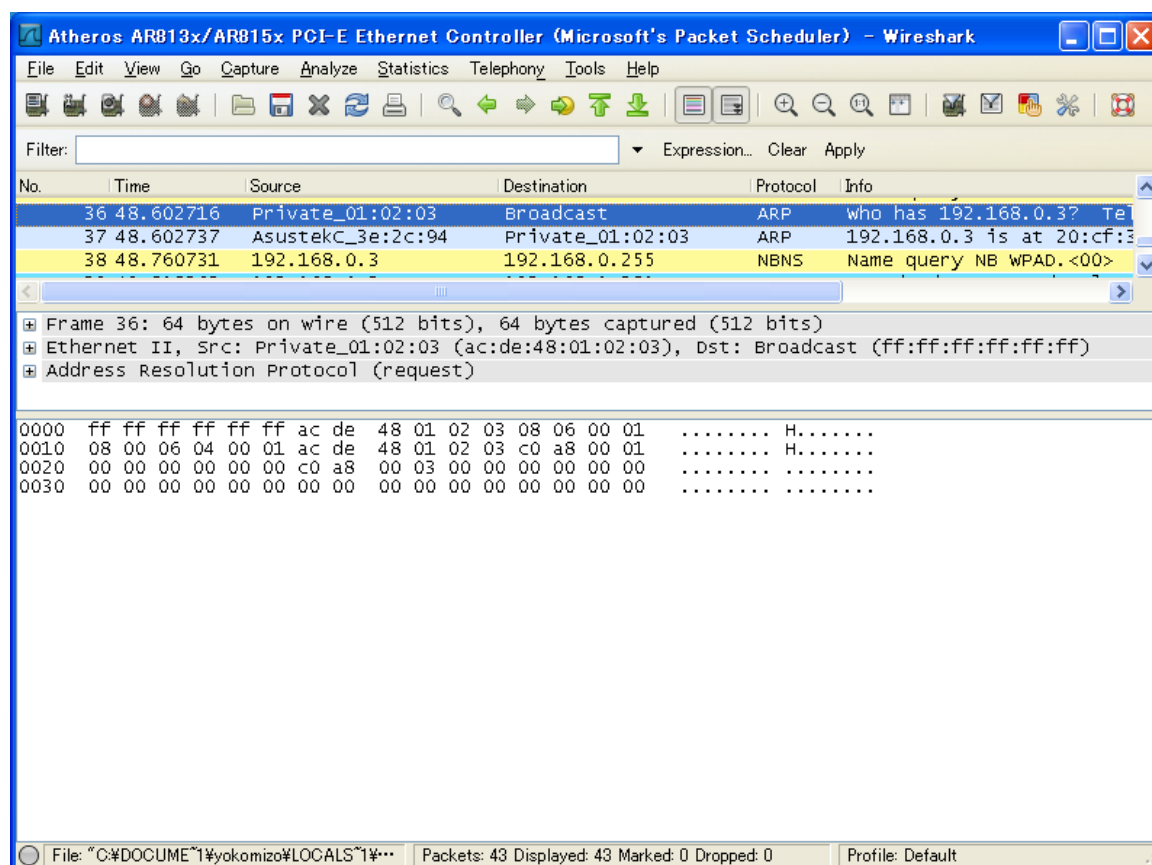


Capture をクリックして、有線 LAN 観測を指定する。



FPGA へのプログラミング実行、iMPACT でデバイス上でマウス右ボタンを押して Program 選択  
FPGA が起動して通信が開始される。

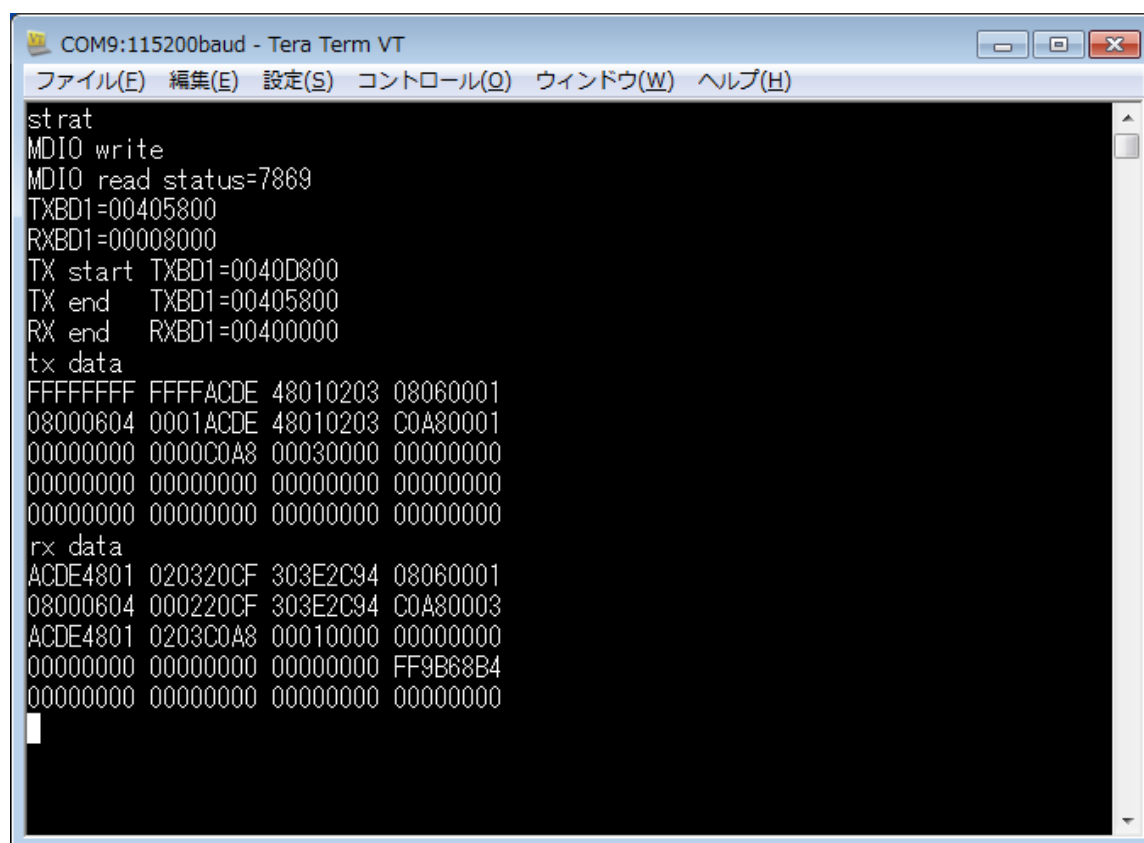
wireshark の表示確認



PC で FPGA から ARP フレームを受信して、応答の ARP フレームを送信していることが分かります。



## ターミナルソフトに通信結果



The screenshot shows a Tera Term VT terminal window titled "COM9:115200baud - Tera Term VT". The menu bar includes "ファイル(E)", "編集(E)", "設定(S)", "コントロール(Q)", "ウィンドウ(W)", and "ヘルプ(H)". The terminal output displays the following text:

```
strat
MDIO write
MDIO read status=7869
TXBD1=00405800
RXBD1=00008000
TX start TXBD1=0040D800
TX end TXBD1=00405800
RX end RXBD1=00400000
tx data
FFFFFFFF FFFFACDE 48010203 08060001
08000604 0001ACDE 48010203 C0A80001
00000000 0000C0A8 00030000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
rx data
ACDE4801 020320CF 303E2C94 08060001
08000604 000220CF 303E2C94 C0A80003
ACDE4801 0203C0A8 00010000 00000000
00000000 00000000 00000000 FF9B68B4
00000000 00000000 00000000 00000000
```

ARP フレームを送信して、PC からの応答の ARP フレームが受信できていれば OK です。

動作確認はここまで

