

## FPGA マガジン No.3 サンプルデータ

### OpenCores EtherMAC DE0 用設計データの使い方 (2013/11/12)

#### ファイル一覧

- ・ nios2e\_sys.v トップ回路
- ・ avalon2wb.v WishBone バスブリッジ回路
- ・ iobus\_reg.v 設定レジスタ回路
- ・ wb\_bram\_if.v RAM インターフェース回路
- ・ user\_module.v デバック回路
- ・ rmii\_if\_tx.v 送信用 MII→RMII 変換回路
- ・ rmii\_if\_rx.v 受信用 RMII→MII 変換回路
- ・ nio2e.qsys Qsys 設定ファイル
- ・ avalon2wb\_hw.tcl avalon2wb.v 組み込み用 TCL コマンド
- ・ nios2e\_sys.qsf ピン配置指定
- ・ i2c\_sw1.c i2c 制御用 C ソース
- ・ ethermac\_de0\_sw1.c EtherMAC 制御用 C ソースコード

#### 免責事項

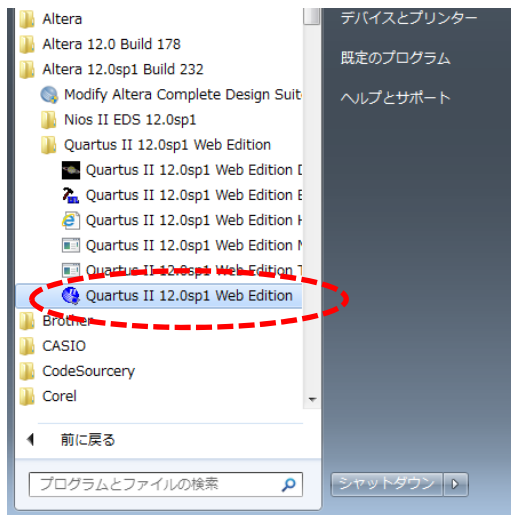
本データの使用が原因として発生した損失や損害について、(有) ひまわり および 著作者は一切責任を負いません。著作者：横溝憲治 fpga@hmwr-lsi.co.jp

#### 設計手順

- ・ 設計用フォルダとして ethermac/nios2e\_sys を作成する
- ・ 記事のダウンロードデータを解凍して、ethermac\_de0\_data の下にある全てのファイルを ethermac/nios2e\_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/nios2e\_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/nios2e\_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/nios2e\_sys へコピー

## Quartus II の起動

スタートメニューから「Altera 12.0sp1 Build 232」→「Quartus II 12.0sp1 Web Edition」→「Quartus II 12.0sp1 Web Edition」を起動する。

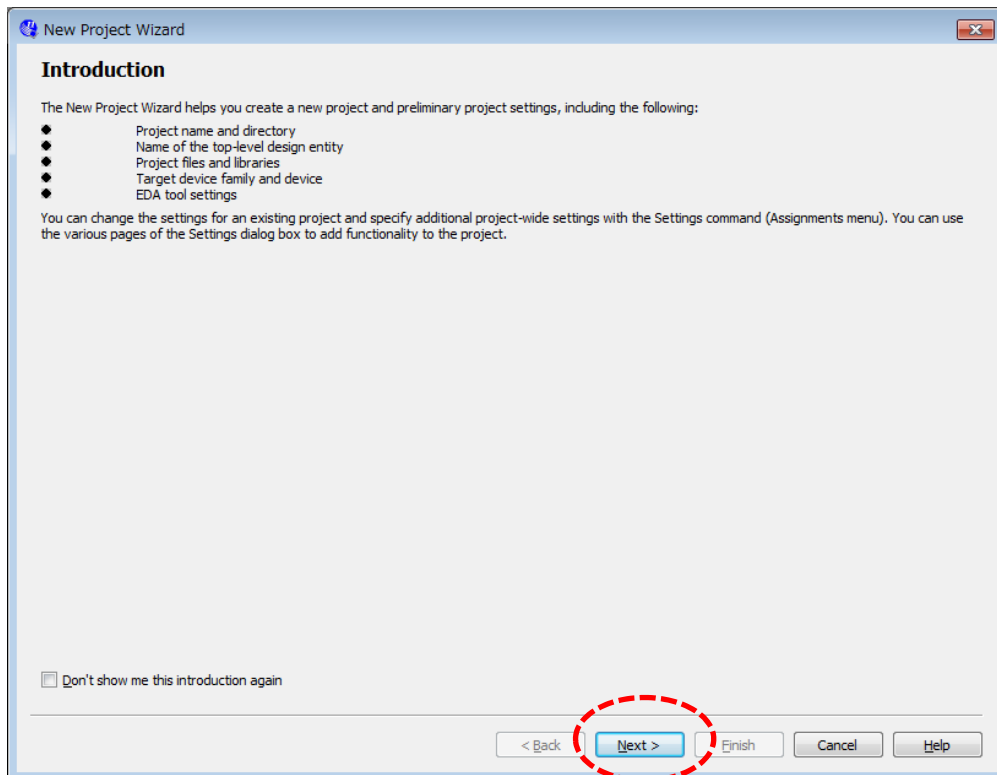


- ・新規設計プロジェクト作成する

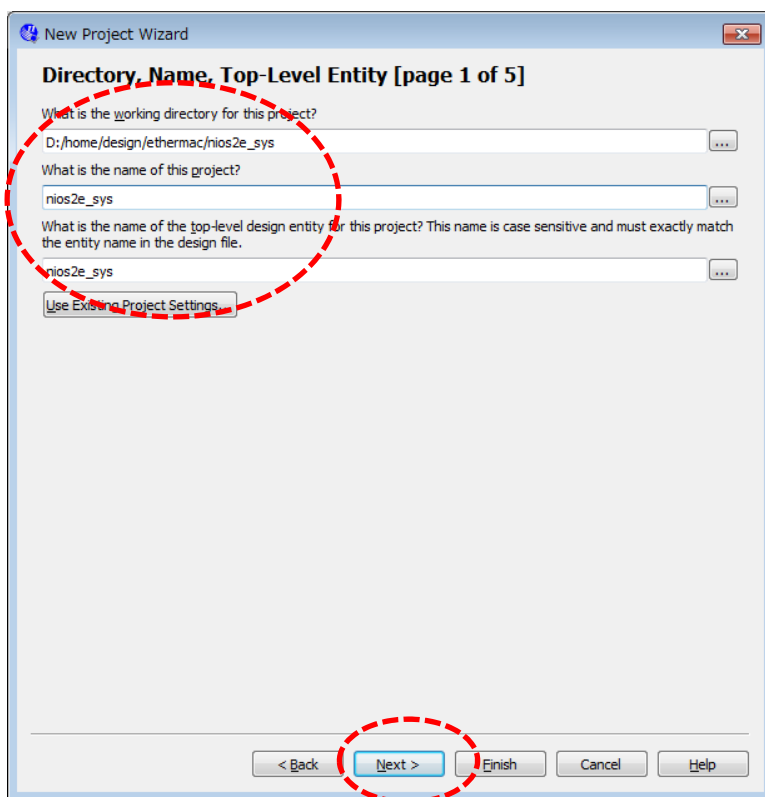


Create a New Project をクリック



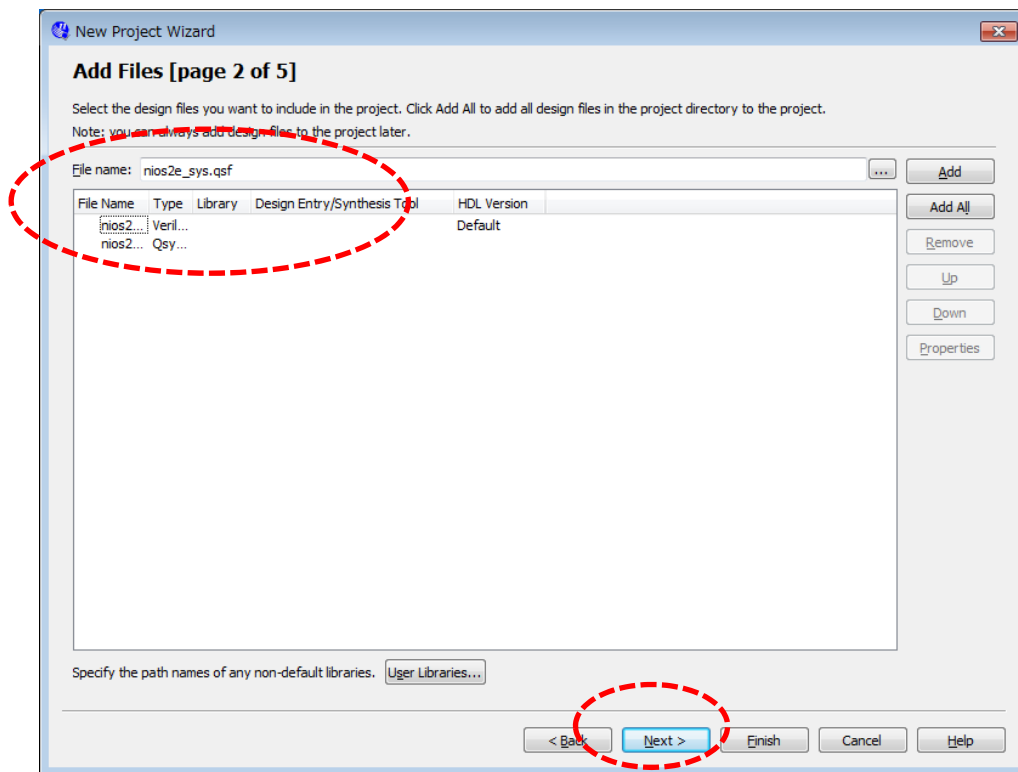


NEXT をクリック

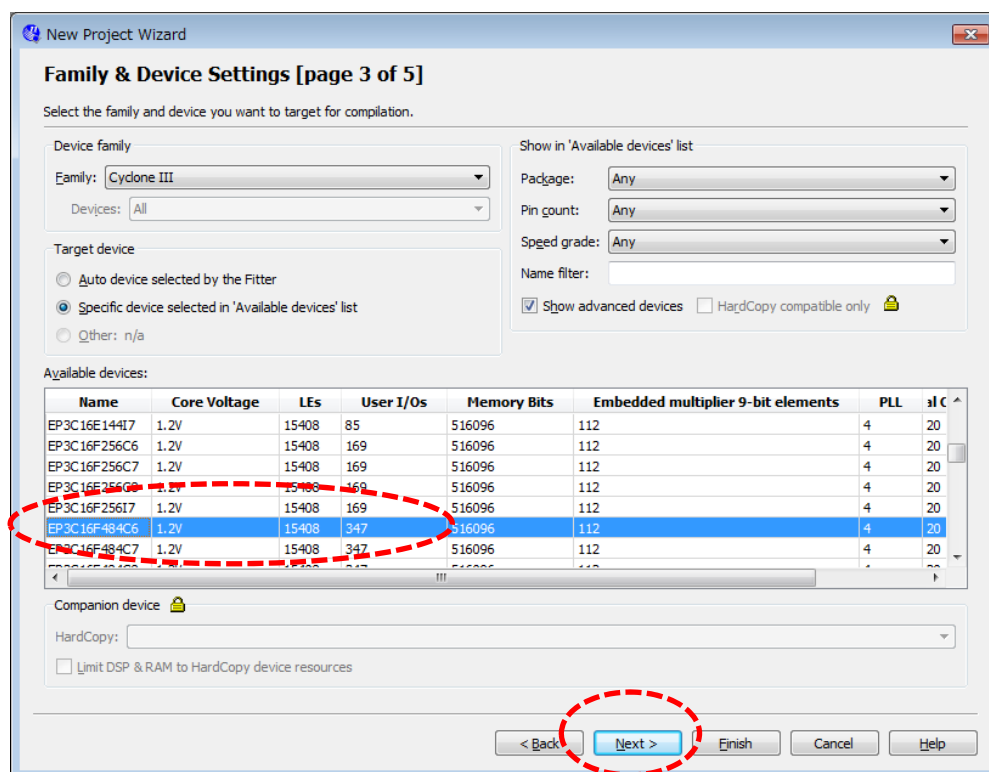


設計フォルダ : [任意]/ethermac/nios2e\_sys、プロジェクト名 : nios2e\_sys を指定





nios2e\_sys.v, nios2e\_sys.qsf, nios2e.qsys を設計データとして追加



デバイスの指定、DE0 に合わせる



New Project Wizard

### EDA Tool Settings [page 4 of 5]

Specify the other EDA tools used with the Quartus II software to develop your project.

EDA tools:

Tool Type	Tool Name	Format(s)	Run Tool Automatically
Design Entry/Synthesis	<None>	<None>	<input type="checkbox"/> Run this tool automatically to synthesize the current design
Simulation	<None>	<None>	<input type="checkbox"/> Run gate-level simulation automatically after compilation
Formal Verification	<None>		
Board-Level	Timing	<None>	
	Symbol	<None>	
	Signal Integrity	<None>	
	Boundary Scan	<None>	

< Back **Next >** Finish Cancel Help



New Project Wizard

### Summary [page 5 of 5]

When you click Finish, the project will be created with the following settings:

Project directory: D:/home/design/ethermac/nios2e\_sys

Project name: nios2e\_sys

Top-level design entity: nios2e\_sys

Number of files added: 2

Number of user libraries added: 0

Device assignments:

Family name: Cydome III

Device: EP3C16F484C7

EDA tools:

Design entry/synthesis: <None> (<None>)

Simulation: <None> (<None>)

Timing analysis: 0

Operating conditions:

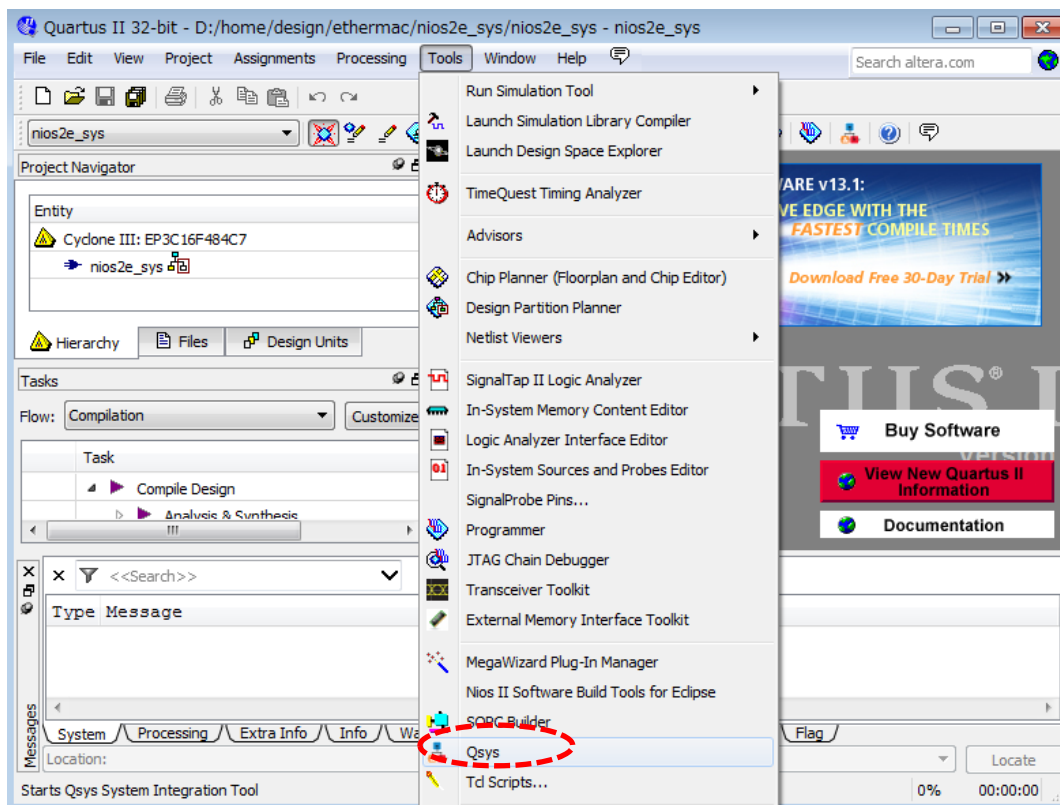
VCCINT voltage: 1.2V

Junction temperature range: 0-85 °C

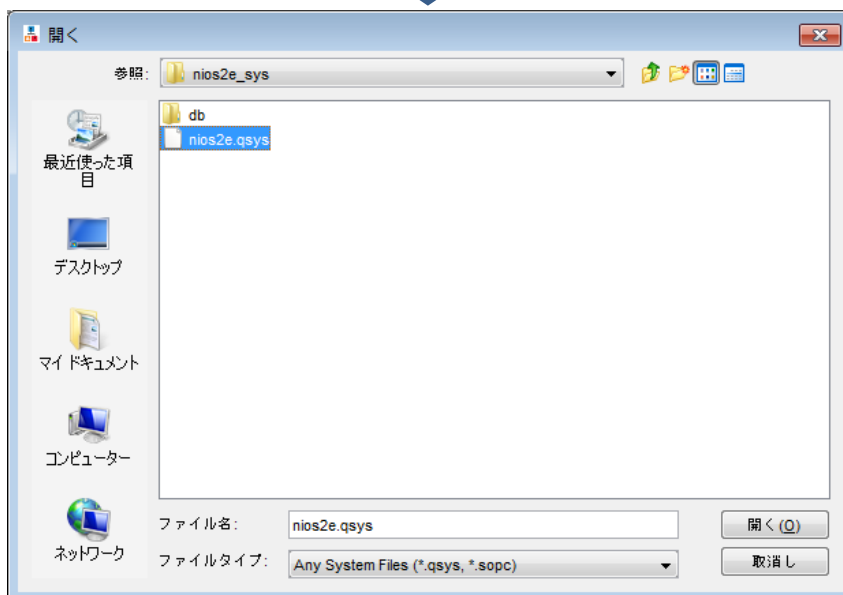
< Back Next > **Finish** Cancel Help

Finish をクリックでプロジェクトが作成される



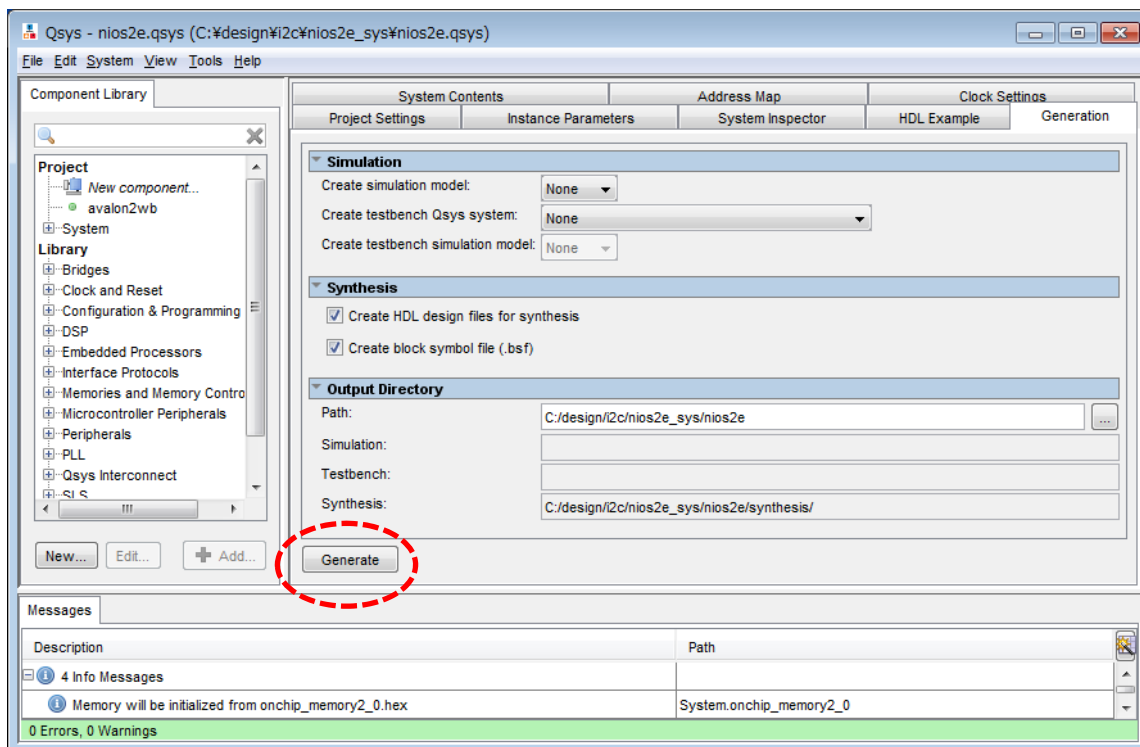


Qsys を起動、Tools→Qsys 選択

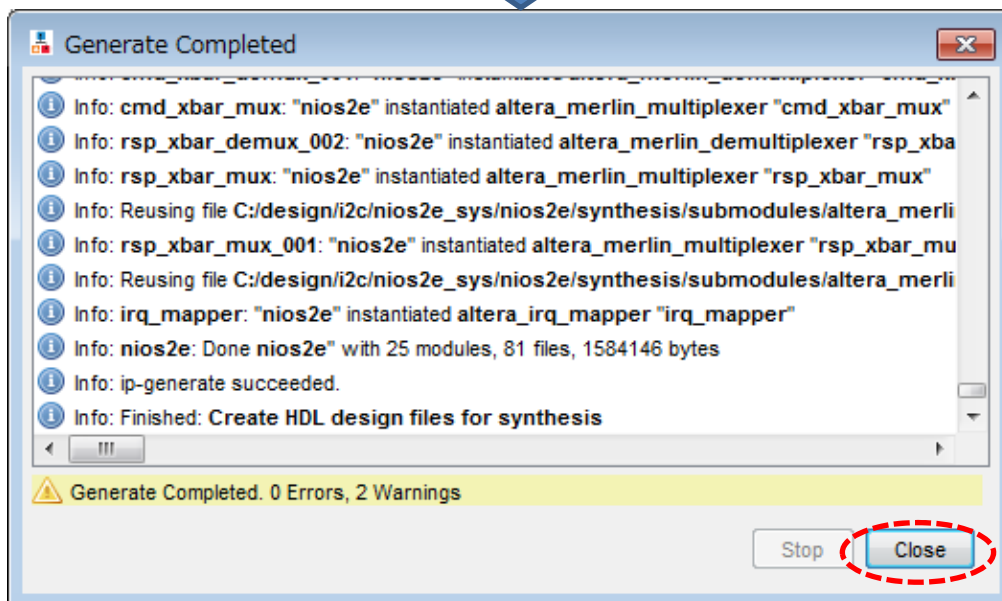


nios2e.qsys を指定



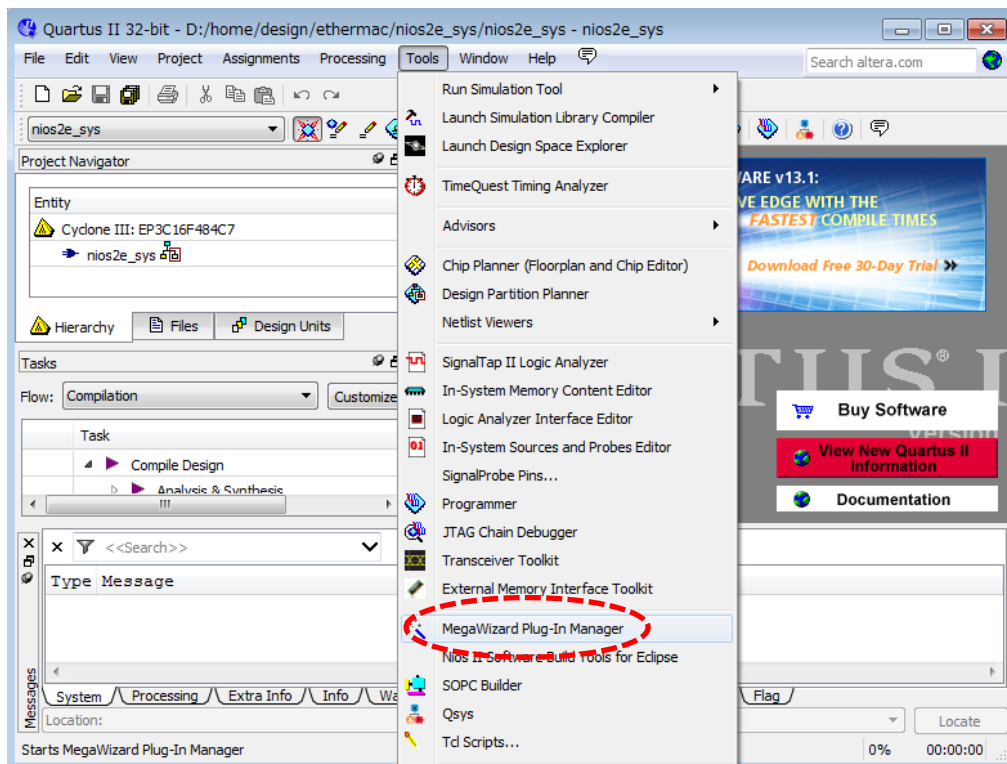


Nios II を作成

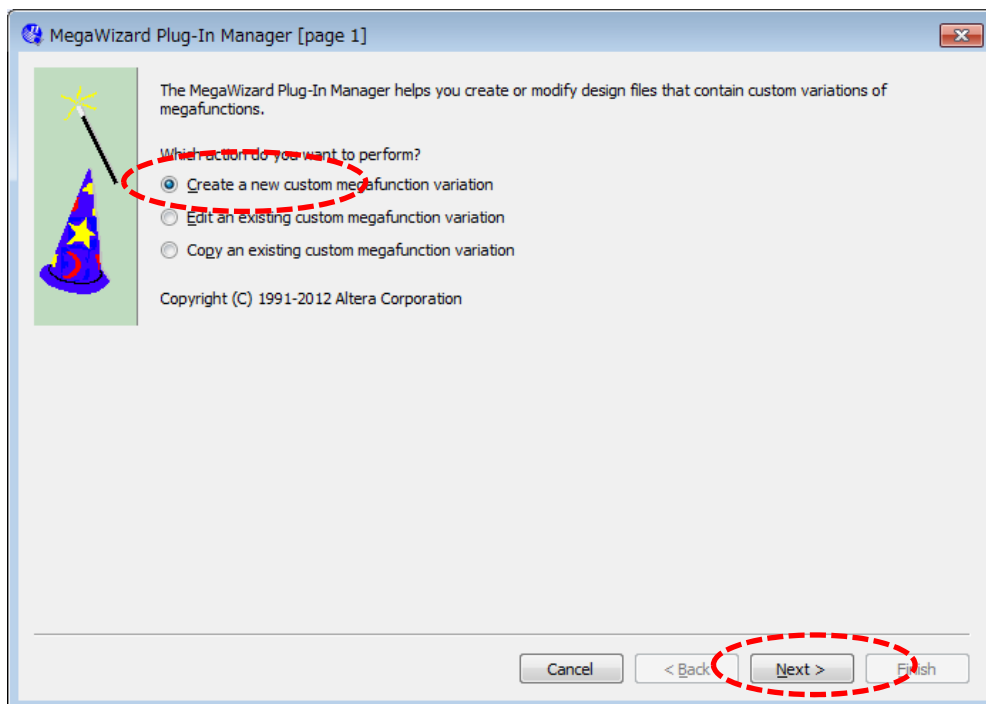


Nios II 作成完了





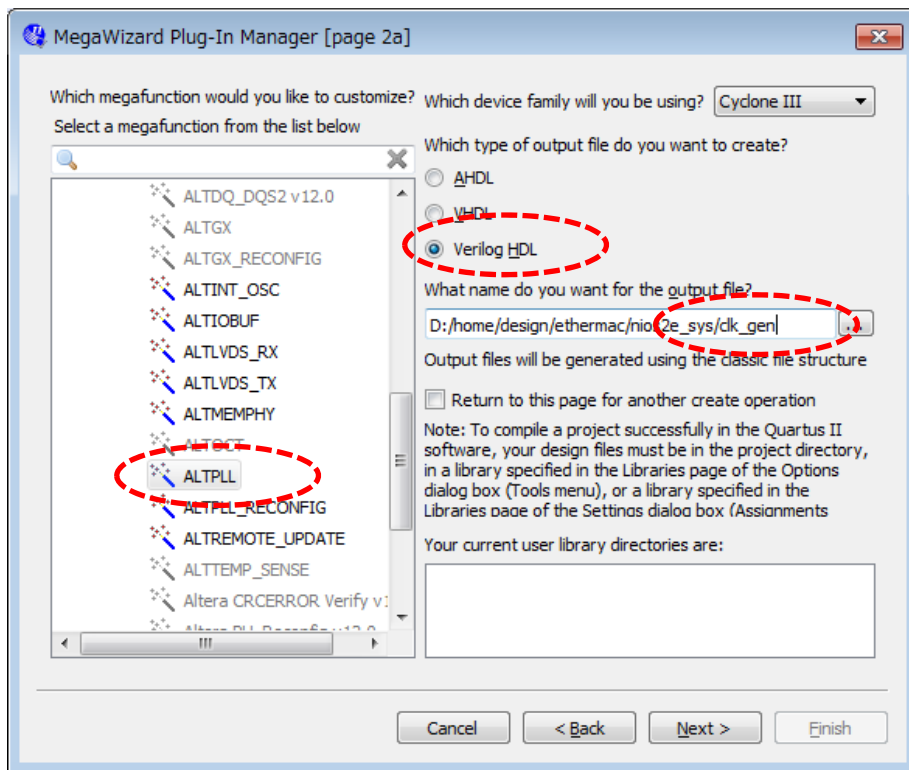
clk\_gen を作成するため、MegaWizard を起動、Tools→MegaWizard 選択



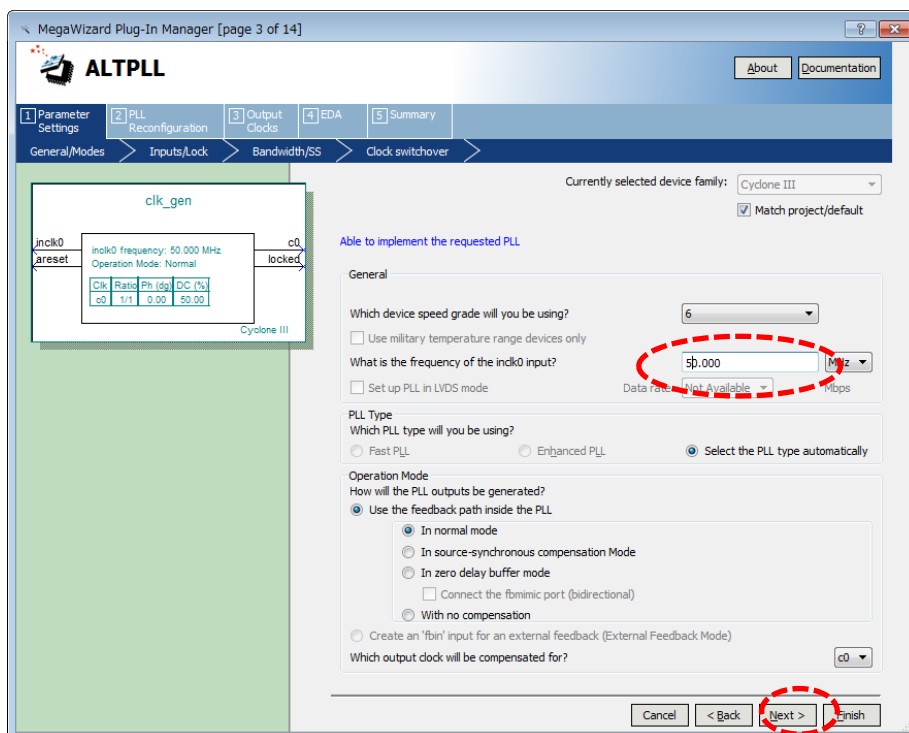
「Create a new...」を選択、Next をクリック





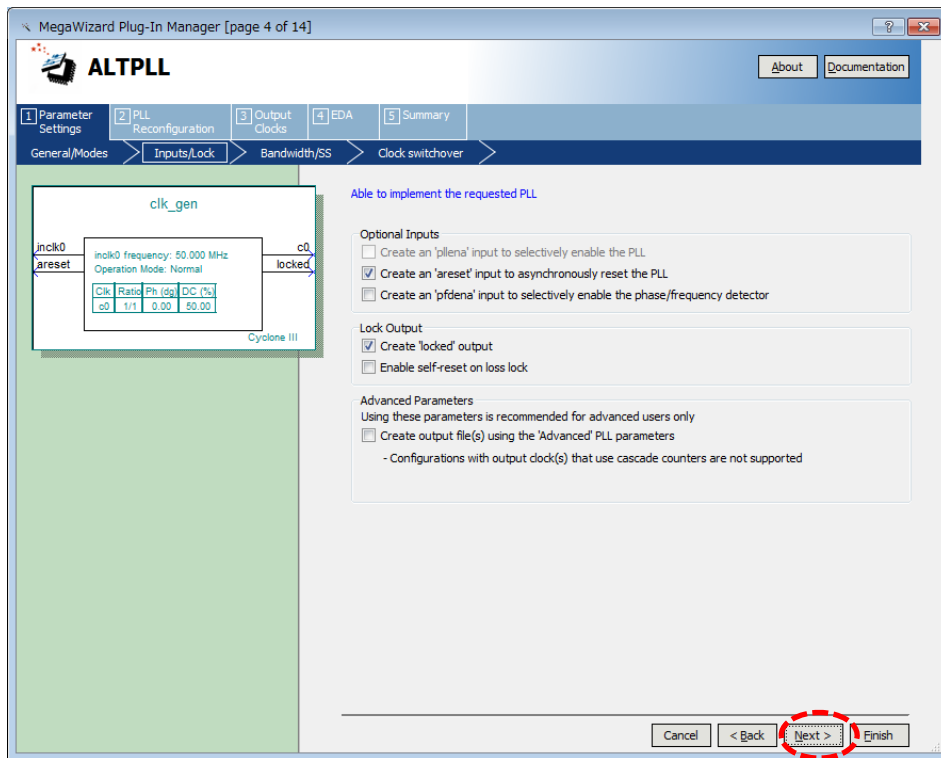


I/O→ALTPLL をクリック、VerilogHDL にチェック、output file に clk\_gen 指定

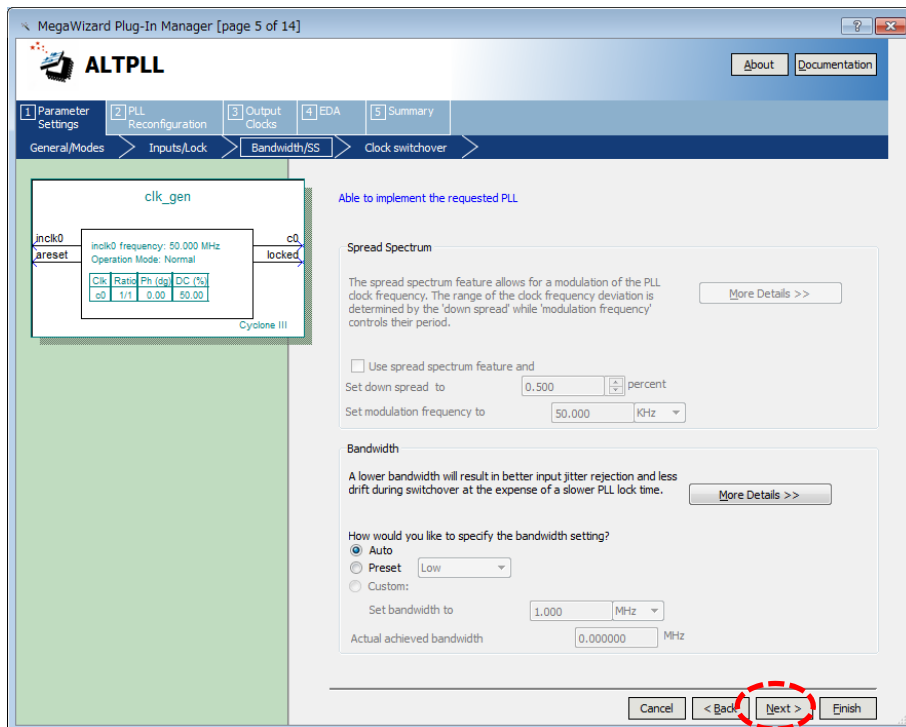


入力クロック周波数を 50MHz に設定、Next をクリック



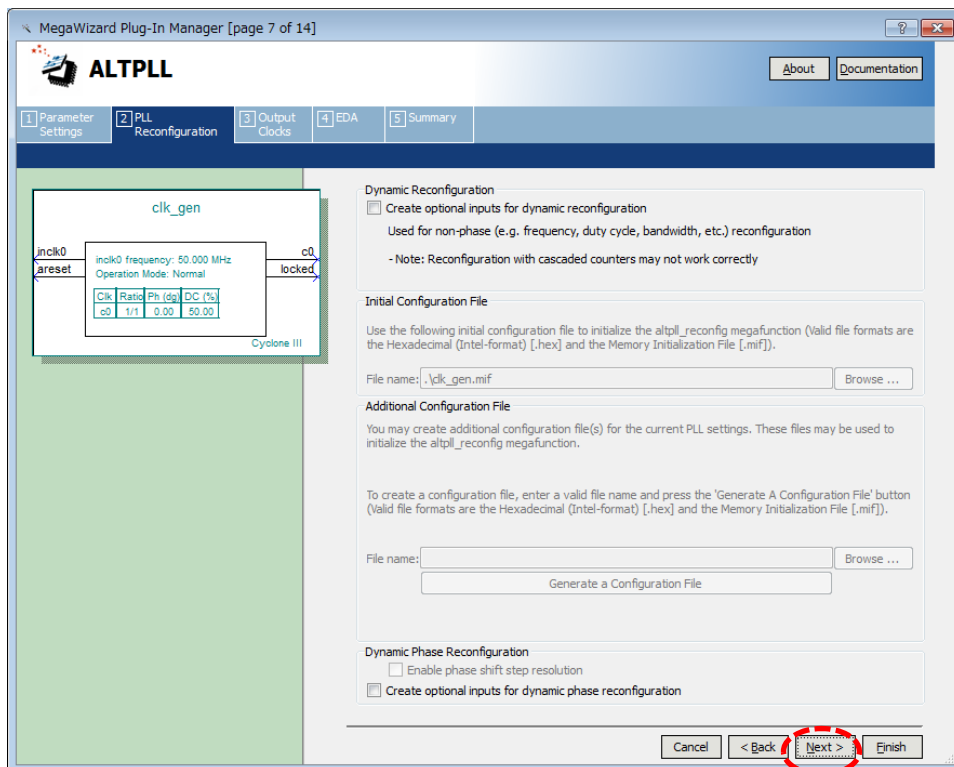


Next をクリック

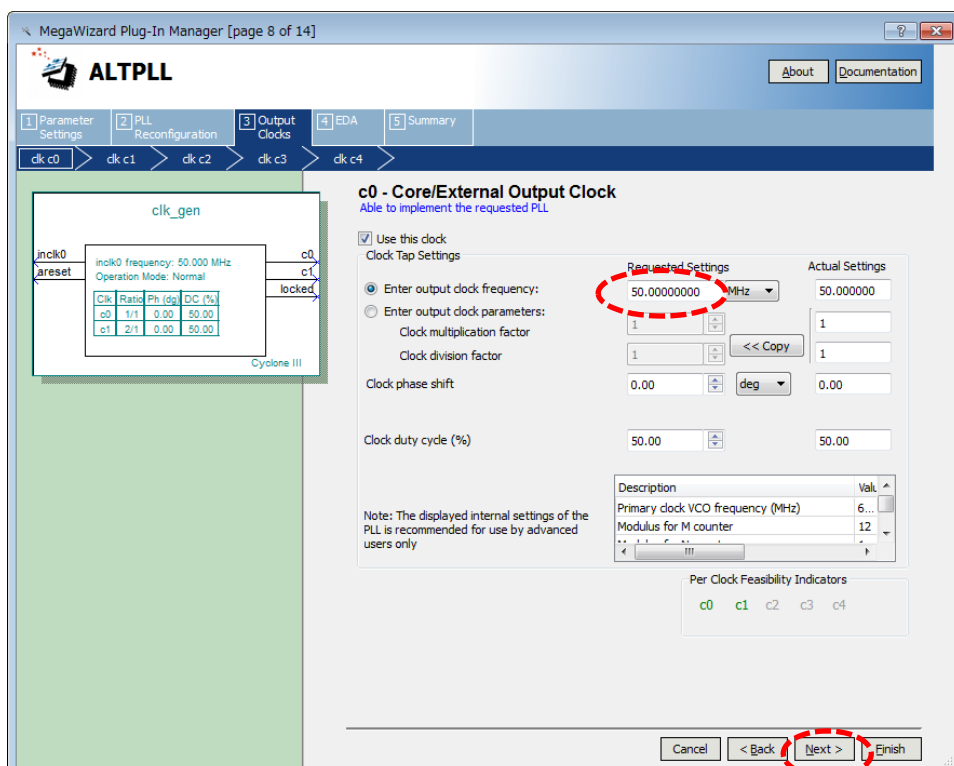


Next をクリック



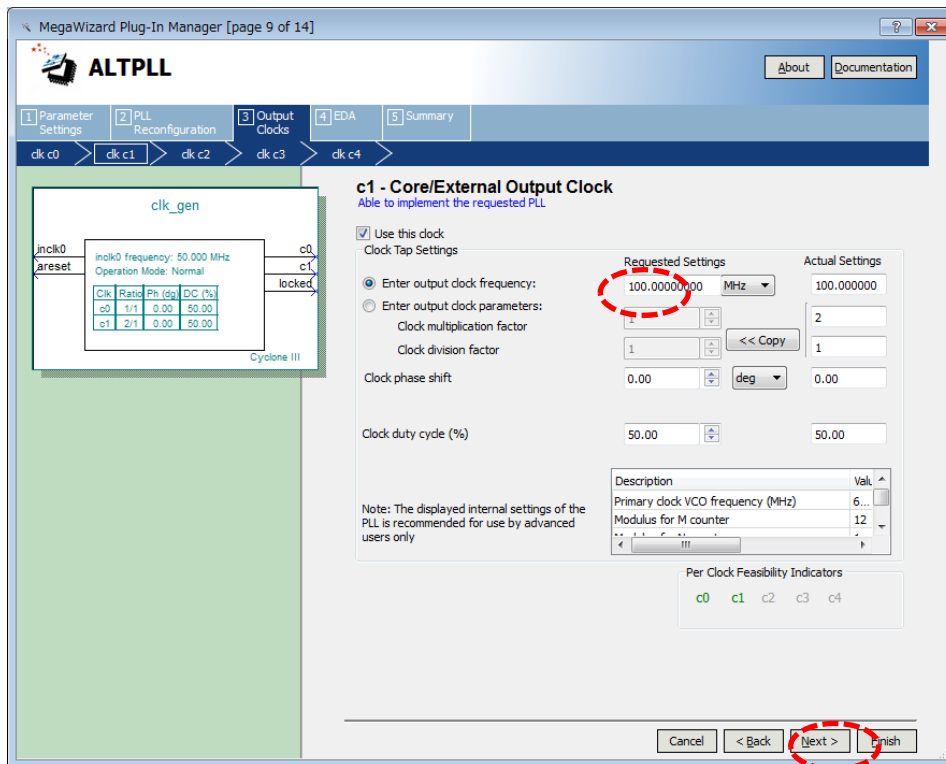


Next をクリック

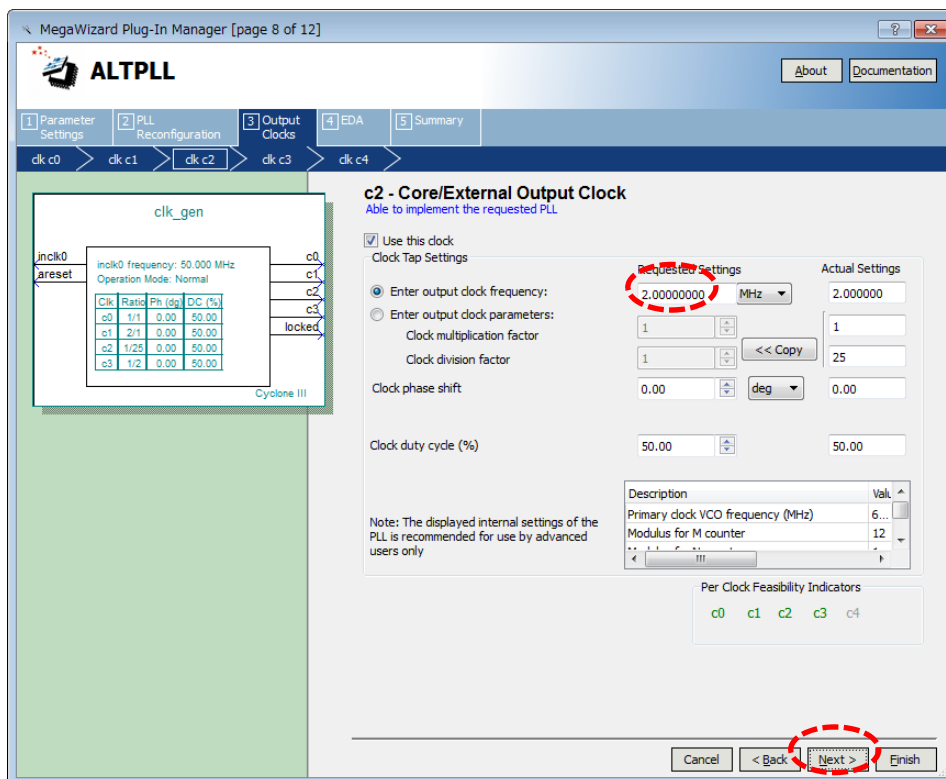


c0 の出力周波数を 50MHz に設定、Next をクリック



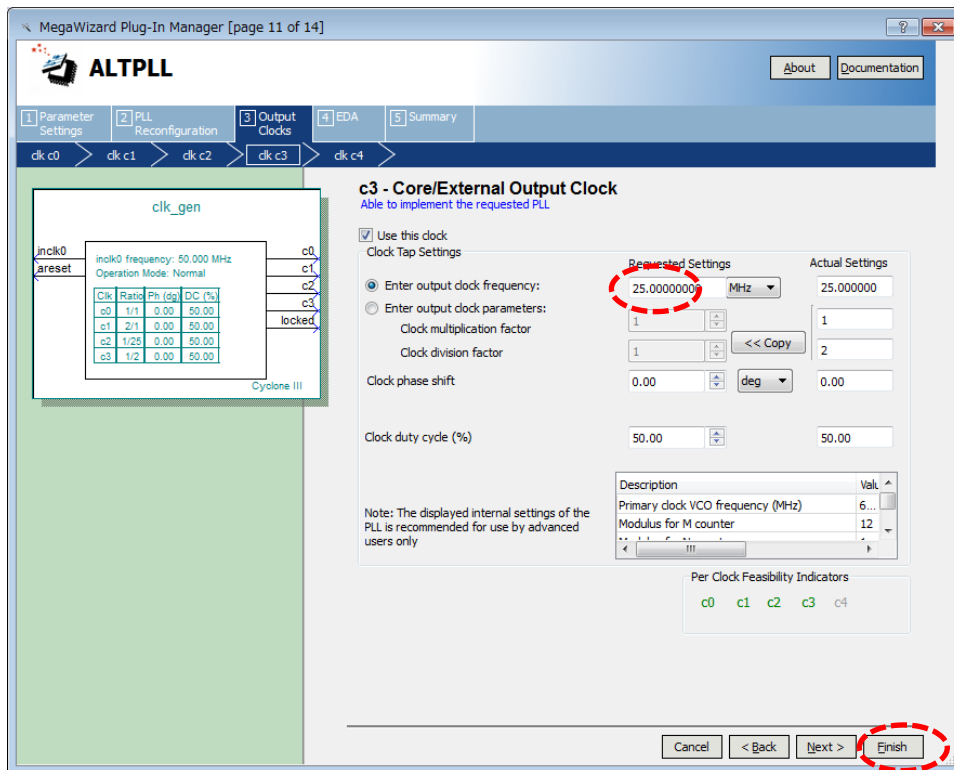


c1 の出力周波数を 100MHz に設定、Next をクリック

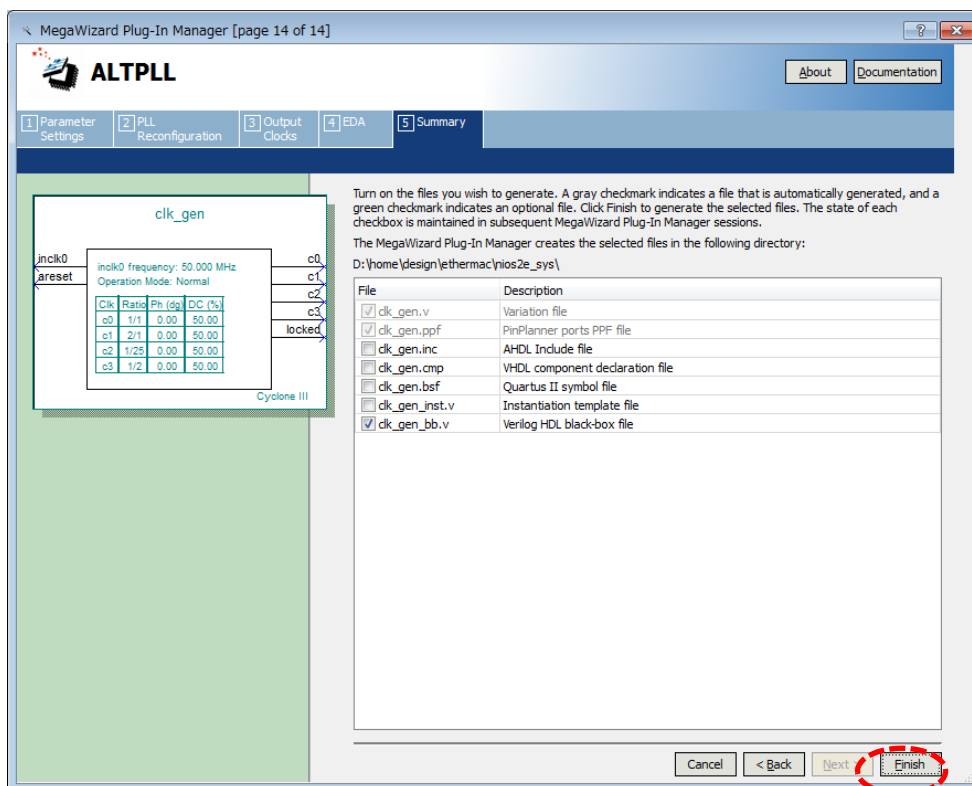


c2 の出力周波数を 2MHz に設定、Next をクリック





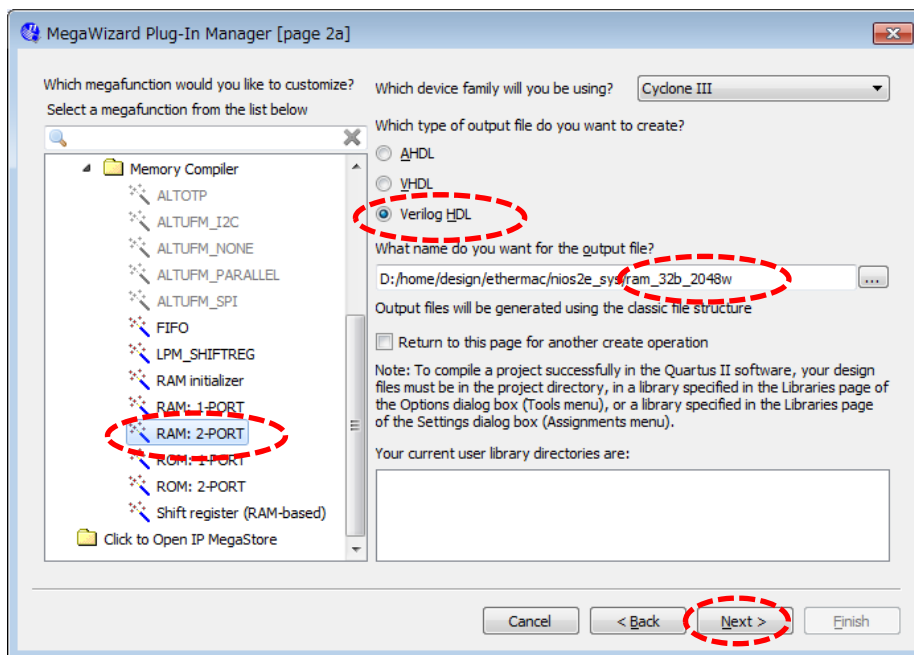
c3 の出力周波数を 25MHz に設定、Finish をクリック



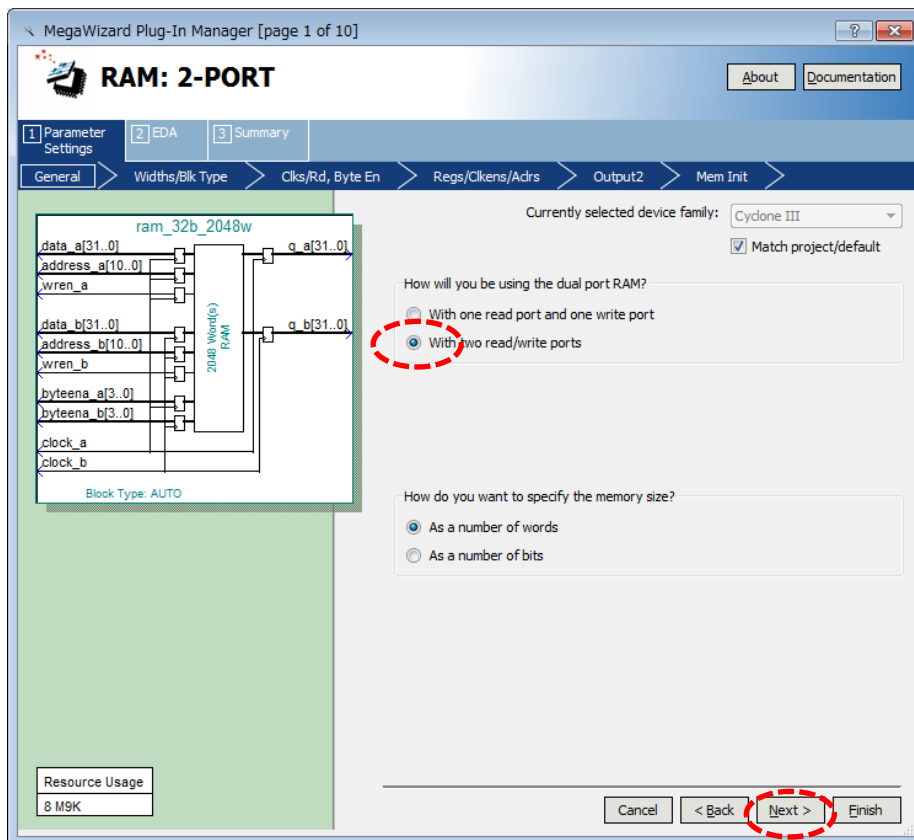
Finish をクリックすると作成を開始する



MegaWizard を起動して ram\_32b\_2048w を作成



Memory Compiler→RAM:2-PORT をクリック、VerilogHDL にチェック、output file に ram\_32b\_2048w 指定



「With two read/write ports」 をチェック、Next をクリック



MegaWizard Plug-In Manager [page 2 of 10]

## RAM: 2-PORT

About Documentation

1 Parameter Settings 2 EDA 3 Summary

General Widths/Blk Type Clks/Rd, Byte En Regs/Clkens/Adrs Output2 Mem Init

ram\_32b\_2048w

data\_a[31..0] address\_a[10..0] wren\_a

data\_b[31..0] address\_b[10..0] wren\_b

byteena\_a[3..0] byteena\_b[3..0]

clock\_a clock\_b

Block Type: AUTO

Resource Usage  
8 M9K

How many 32-bit words of memory? 2048

☐ Use different data widths on different ports

Read/Write Ports

How wide should the 'q\_a' output bus be? 32

How wide should the 'data\_a' input bus be? 32

How wide should the 'q\_b' output bus be? 32

Note: You could enter arbitrary values for width and depth

What should the memory block type be?

☒ Auto ☐ MLAB ☐ M9K

☐ M144K ☐ LCs Options...

Set the maximum block depth to Auto words

Cancel < Back Next > Finish

ワード数を 2048、データ幅を 32 に指定、Next をクリック



MegaWizard Plug-In Manager [page 3 of 10]

## RAM: 2-PORT

About Documentation

1 Parameter Settings 2 EDA 3 Summary

General Widths/Blk Type Clks/Rd, Byte En Regs/Clkens/Adrs Output2 Mem Init

ram\_32b\_2048w

data\_a[31..0] address\_a[10..0] wren\_a

data\_b[31..0] address\_b[10..0] wren\_b

byteena\_a[3..0] byteena\_b[3..0]

clock\_a clock\_b

Block Type: AUTO

Resource Usage  
8 M9K

What clocking method do you want to use?

☐ Single clock

☐ Dual clock: use separate 'read' and 'write' clocks

☐ Dual clock: use separate 'input' and 'output' clocks

☐ No clock (fully asynchronous)

☒ Dual clock: use separate clocks for A and B ports

☐ Create 'rden\_a' and 'rden\_b' read enable signals

Byte Enable Ports

☒ Create byte enable for port A

☒ Create byte enable for port B

What is the width of a byte for byte enables? 8 bits

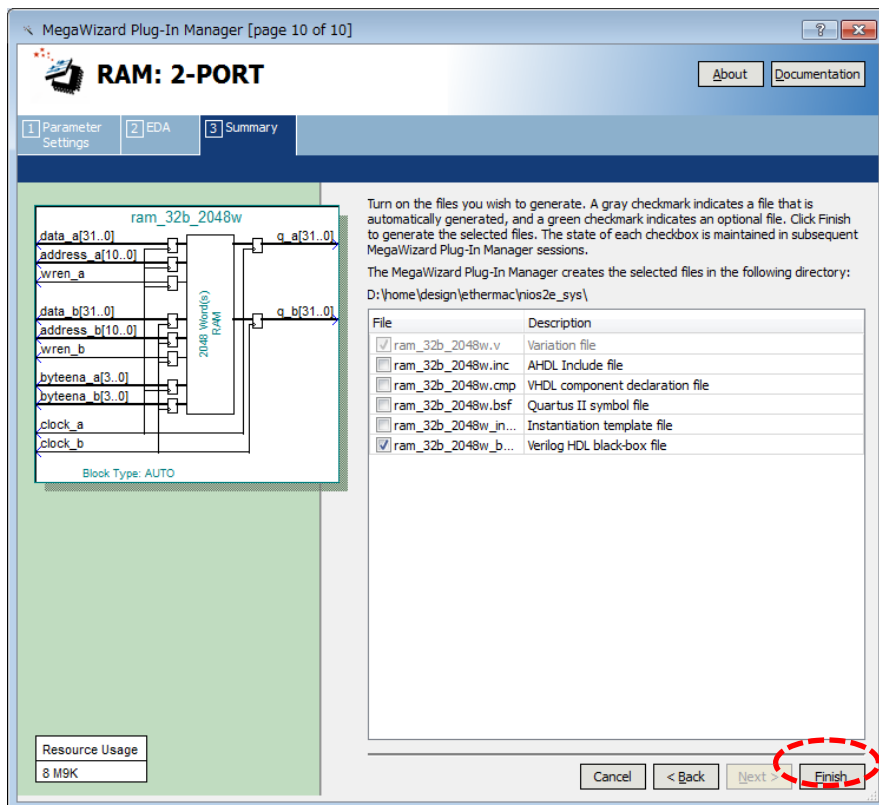
☐ Enable error checking and correcting (ECC) to check and correct single bit errors and detect double errors

☐ Enable ECC pipeline registers before the output decoder to achieve the same performance as non-ECC mode at the expense of one cycle of latency

Cancel < Back Next > Finish

A, B ポートの byte enable をチェック、Finish をクリック

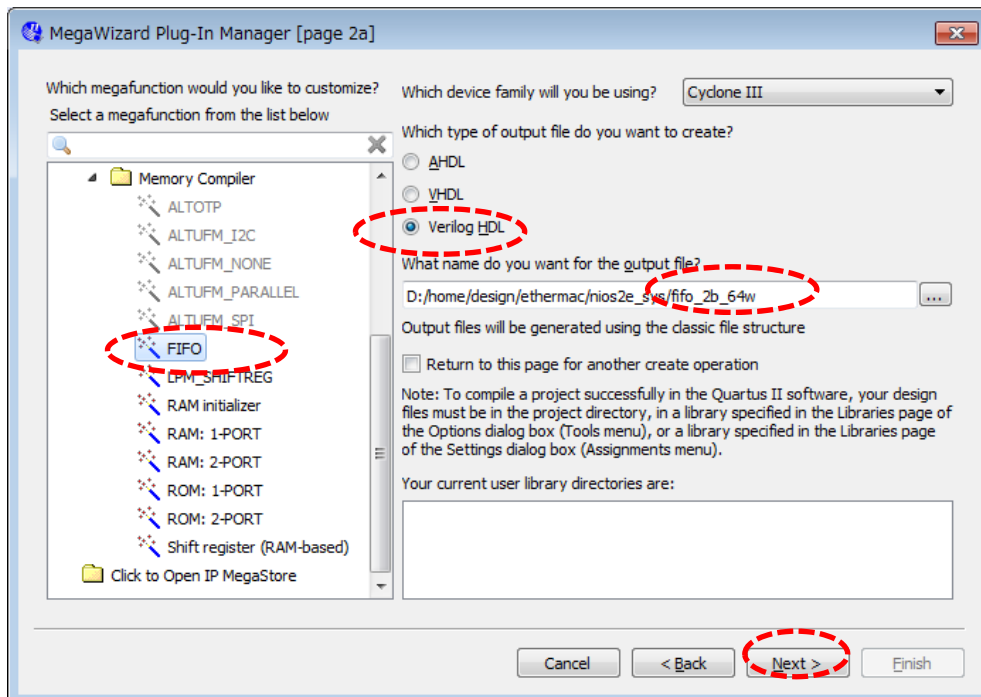




Finish をクリック



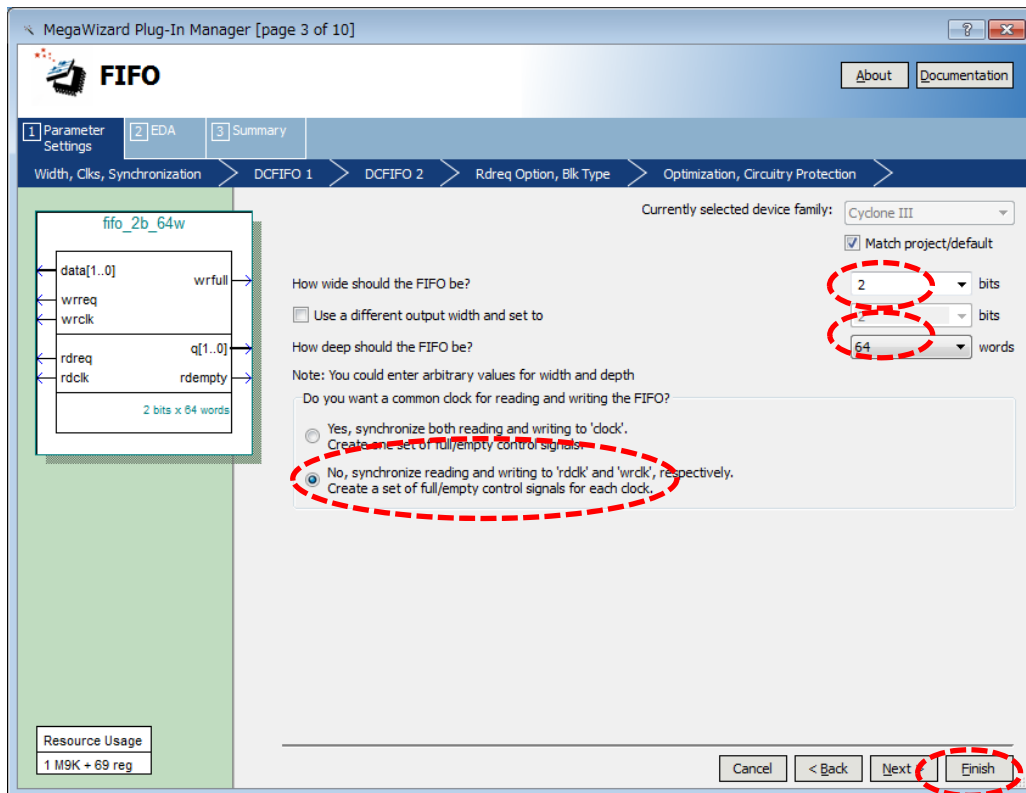
MegaWizard を起動して fifo\_2b\_64w を作成



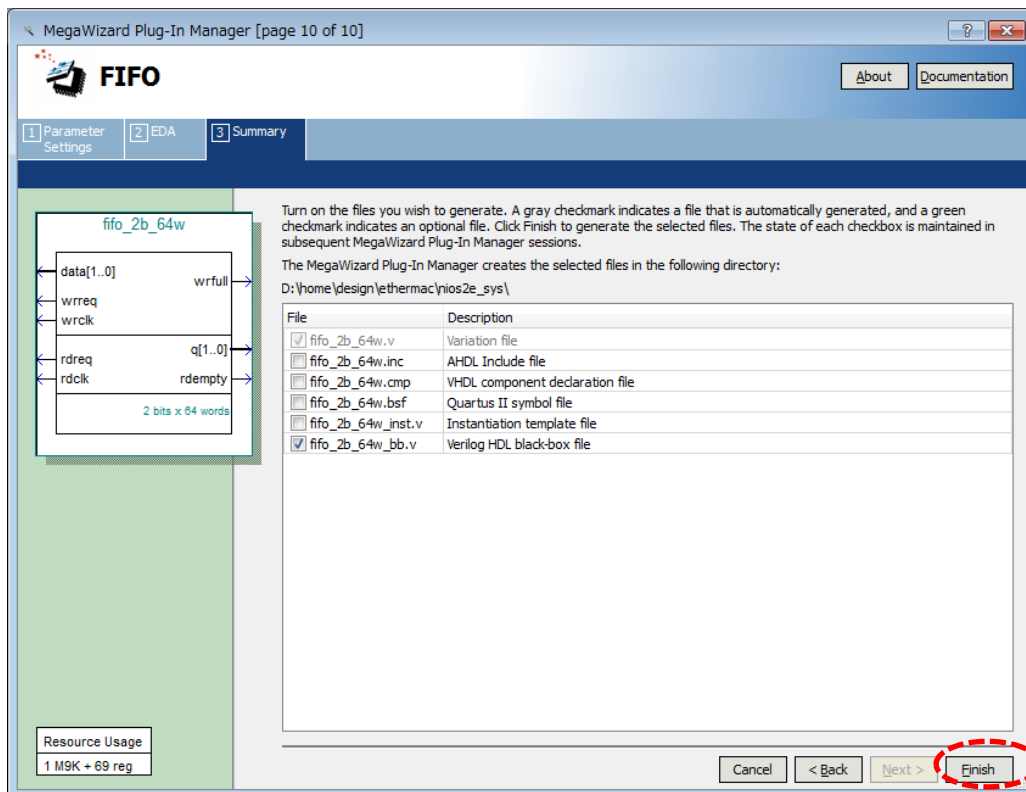
Memory Compiler→FIFO をクリック、VerilogHDL にチェック、output file に fifo\_2b\_64w 指定





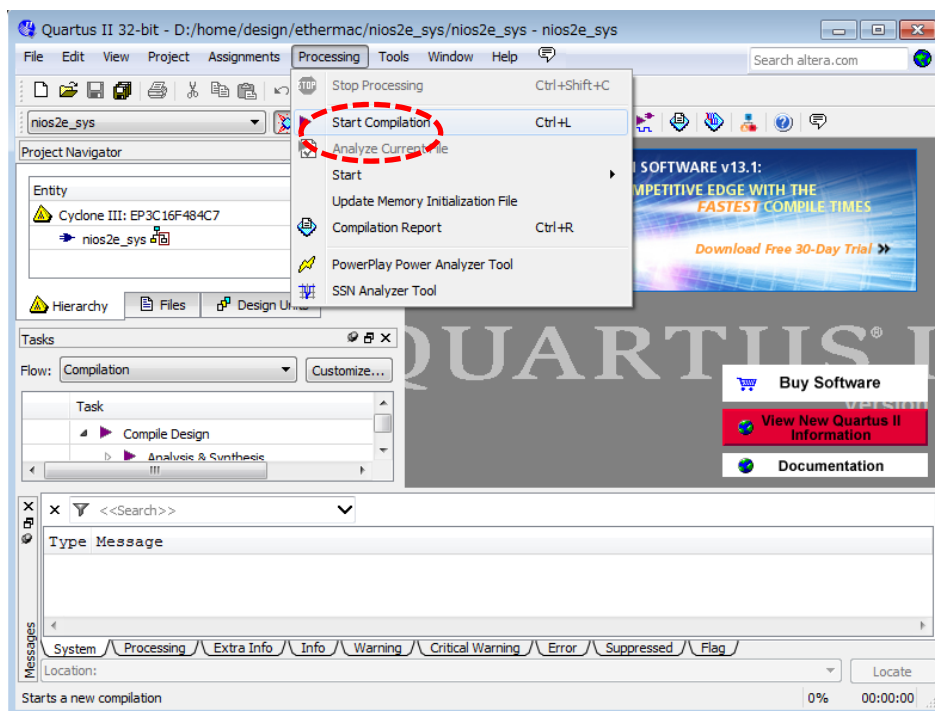


データ幅:2, データ数:64 に設定、「No, synchronize reading and writing...」をチェック  
Finish をクリック



Finish をクリック

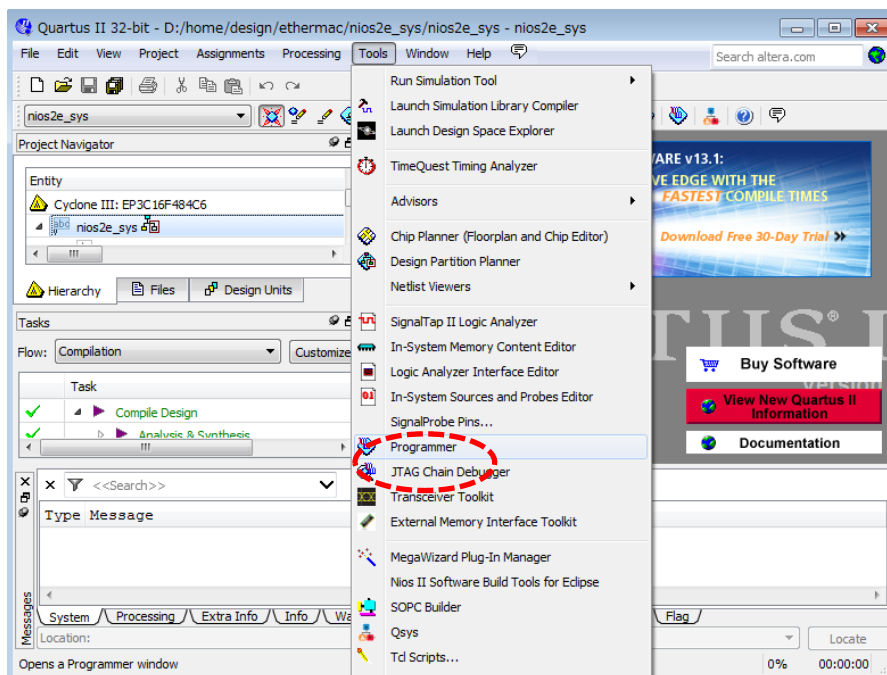




コンパイルを実施、Processing→Start Compilation

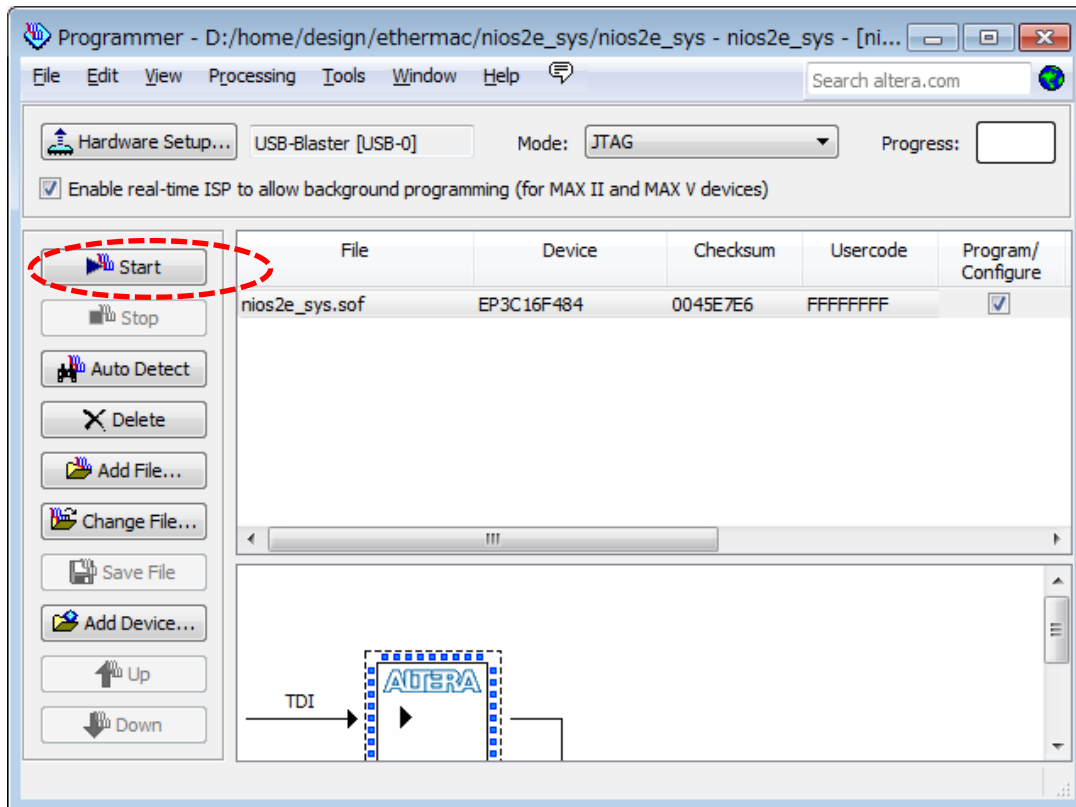


DE0 と PC を UDB で接続、DE0 の UART と PC の USB を UART—USB 変換器で接続、PC でターミナルソフトを立ち上げ、UART—USB 変換器に割り当てられたポート番号へ 115200bps で接続します。  
DE0 とプスィッチ 0 (信号名 sw0) を 1 にして Ethernet 通信を折返しモードする。



プログラムの起動、Tools→Programmer 選択





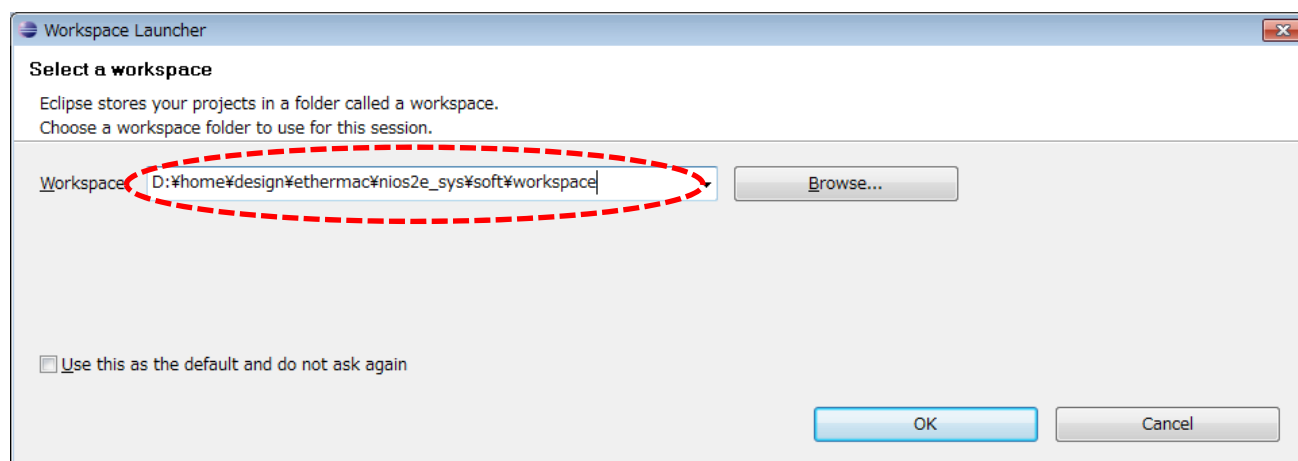
FPGA に回路データ書き込み



次に Nios II EDS (Nios II Software Build Tools) でソフトウェアを作ります。

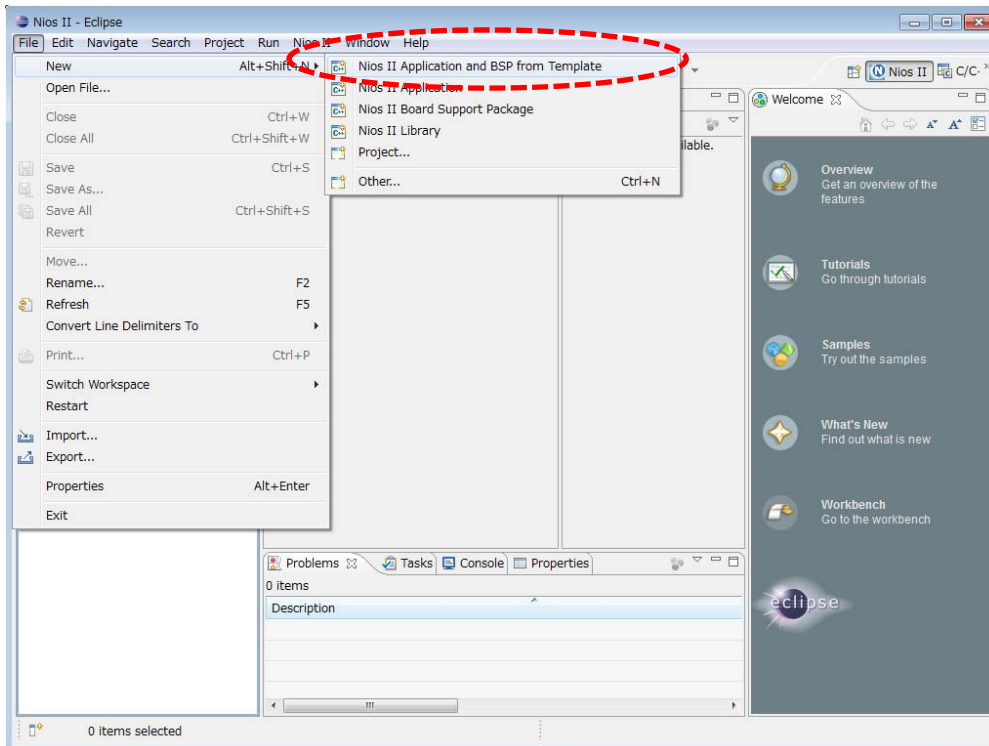
スタートメニューから「Altera 12.0sp1 Build 232」→「Nios II EDS 12.0sp1」→「Nios II 12.0sp1 Software Build Tools for Eclipse」を管理者として実行してください。

管理者として実行しないとソフトウェアプロジェクトを作成でエラーになる場合があります。

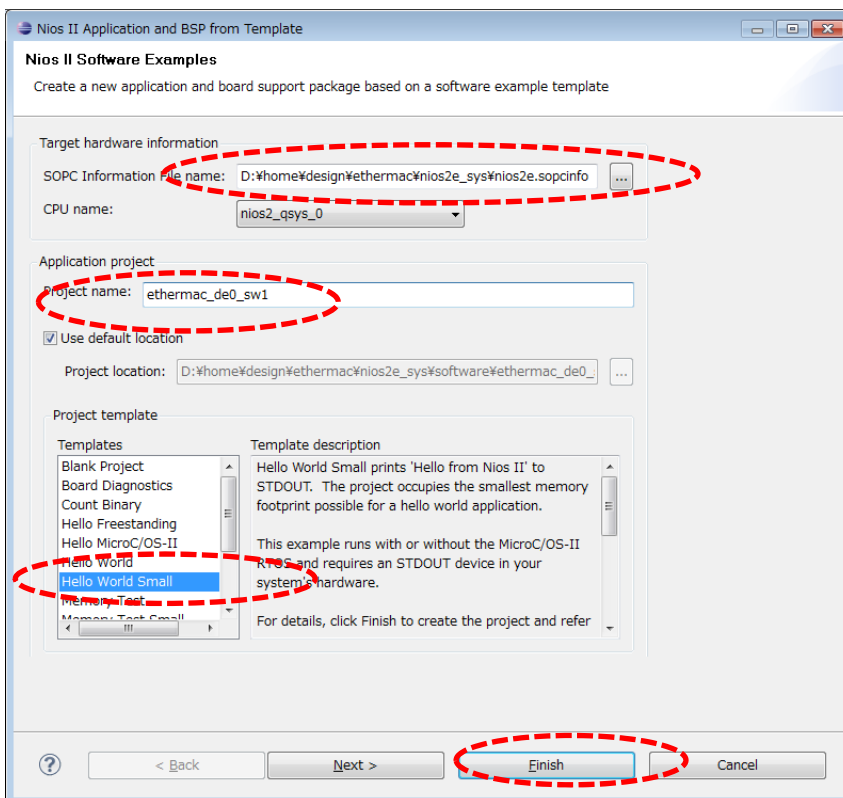


EDS が起動するとワークスペースを指定が要求されます。設計フォルダ/soft/workspace を設定





新規のソフトウェアプロジェクト作成、File→New→NiosII Application and BSP from Template

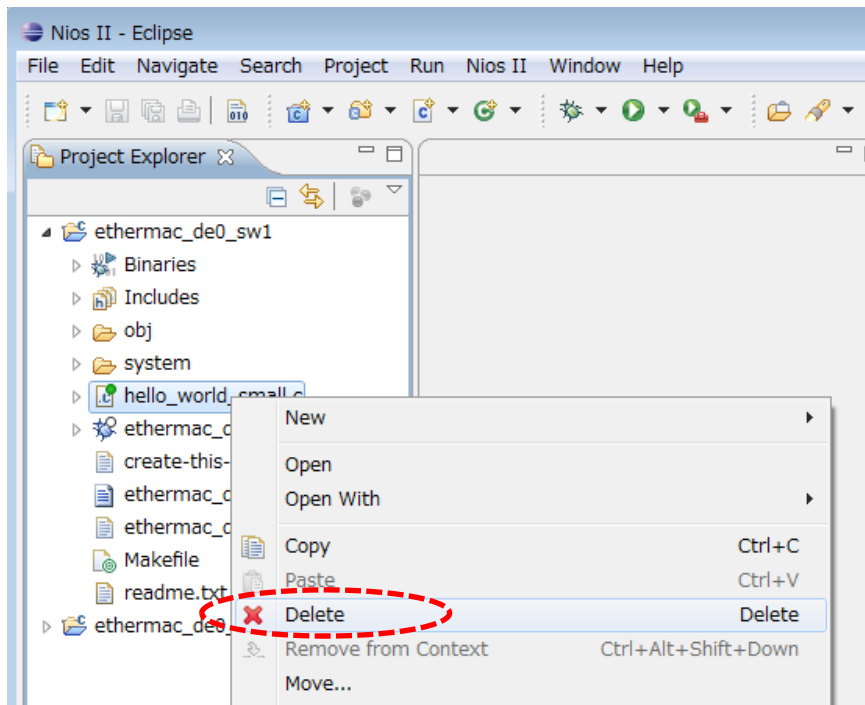


sopc ファイル: 設計フォルダ/ni0s2e.sopcinfo

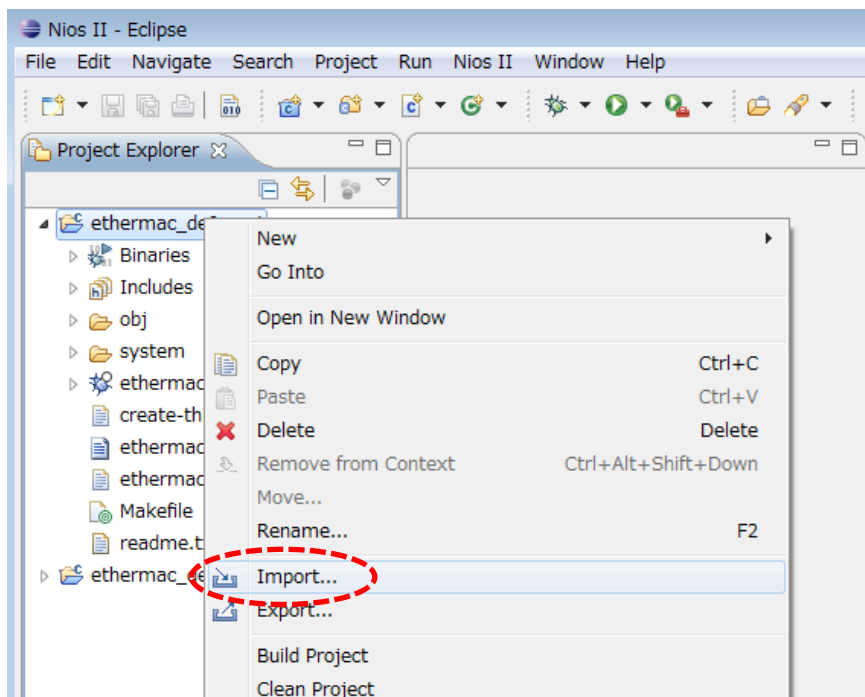
プロジェクト名指定: ethermac\_de0\_sw1

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

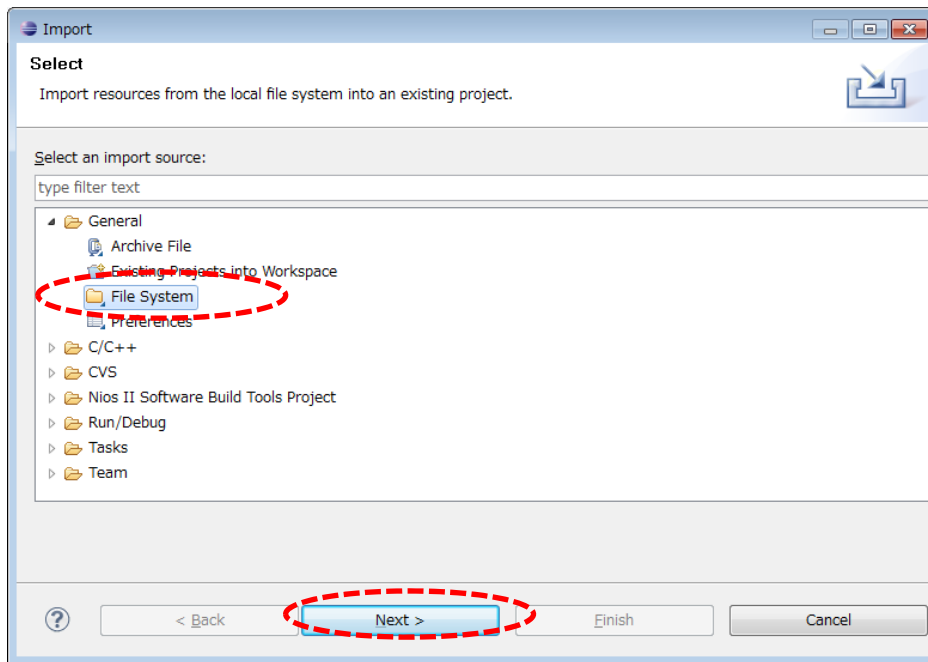




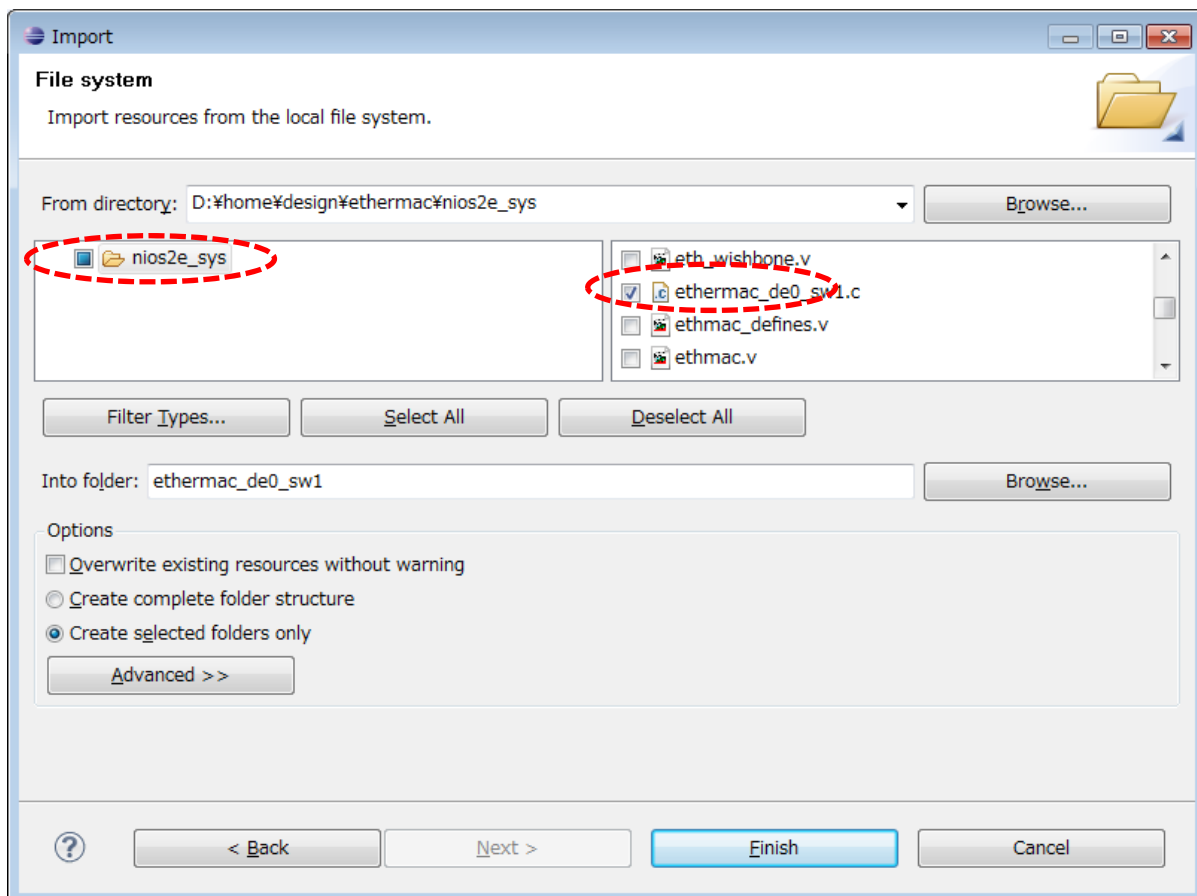
ethermac\_de0\_sw1 にある Hello\_world\_small.c の上でマウスの右ボタンを押して Delete を選択して、削除



ethermac\_de0\_sw1 の上でマウスの右ボタンを押して Import を選択

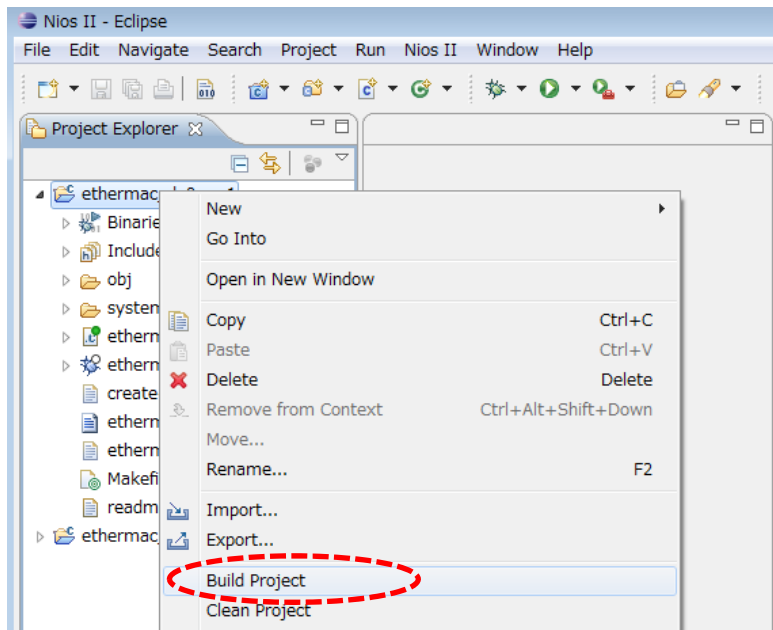


General→File System 選択, Next をクリック

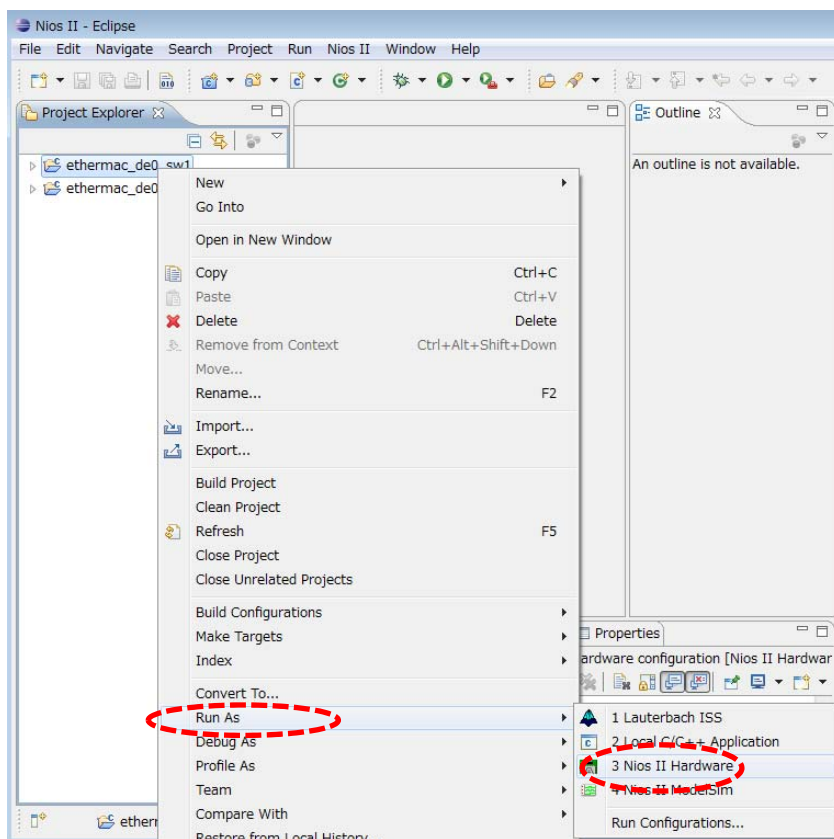


解凍データの ethernac\_de0\_sw1.c を選択





ethermac\_de0\_sw1 でビルドを実行

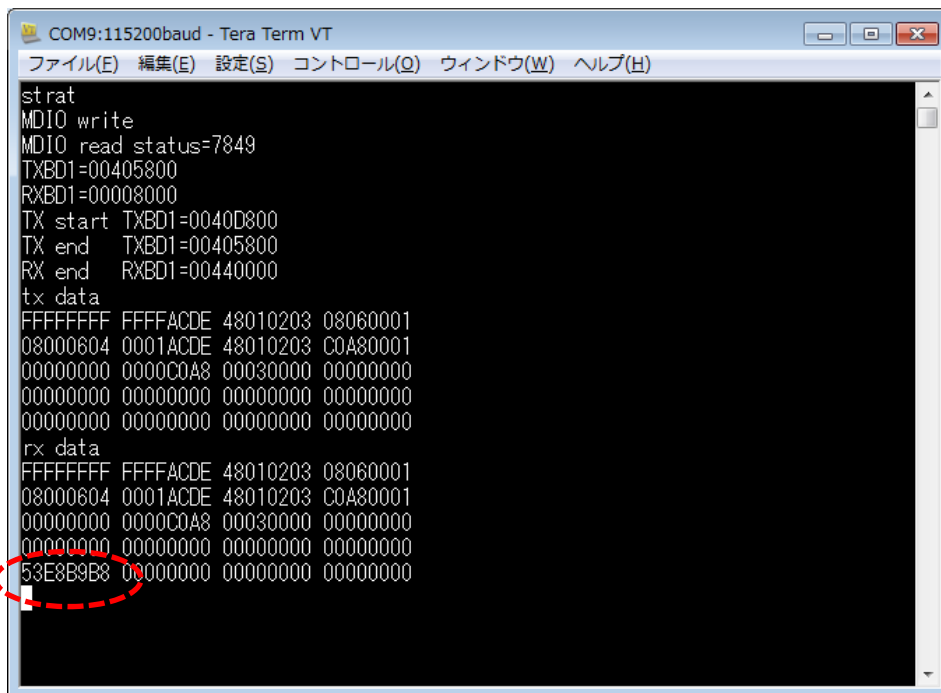


DE0 でソフトウェア実行、ethermac\_de0\_sw1 上でマウス右ボタンを押す→Run As → NiosII Hardware  
ソフトウェアの実行ファイルが書き込まれると DE0 が起動する。





ターミナルソフトに通信結果が表示されます。



```
strat
MDIO write
MDIO read status=7849
TXBD1=00405800
RXBD1=00008000
TX start TXBD1=0040D800
TX end TXBD1=00405800
RX end RXBD1=00440000
tx data
FFFFFFFF FFFACDE 48010203 08060001
08000604 0001ACDE 48010203 C0A80001
00000000 0000C0A8 00030000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
rx data
FFFFFFFF FFFACDE 48010203 08060001
08000604 0001ACDE 48010203 C0A80001
00000000 0000C0A8 00030000 00000000
00000000 00000000 00000000 00000000
53E8B9B8 00000000 00000000 00000000
```

折り返しモード[SW0 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。

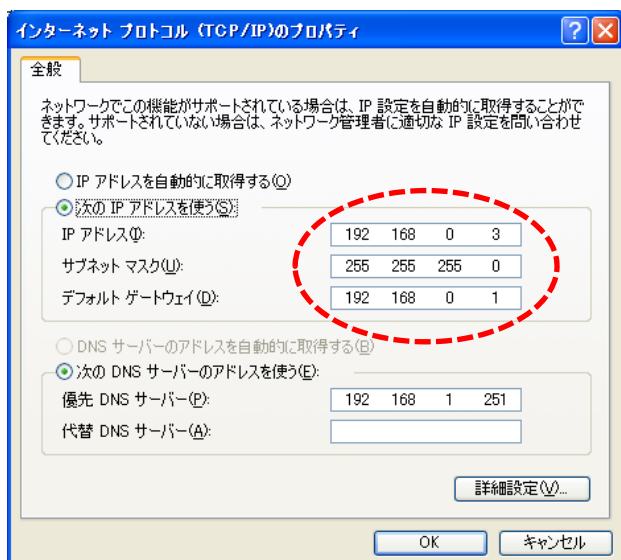
次にEthernetでの動作確認を行います。

DE0 の電源を切り、DE0 とディップスイッチ 0 (信号名 SW0) を 0 にして通常 Ethernet 通信モードする。

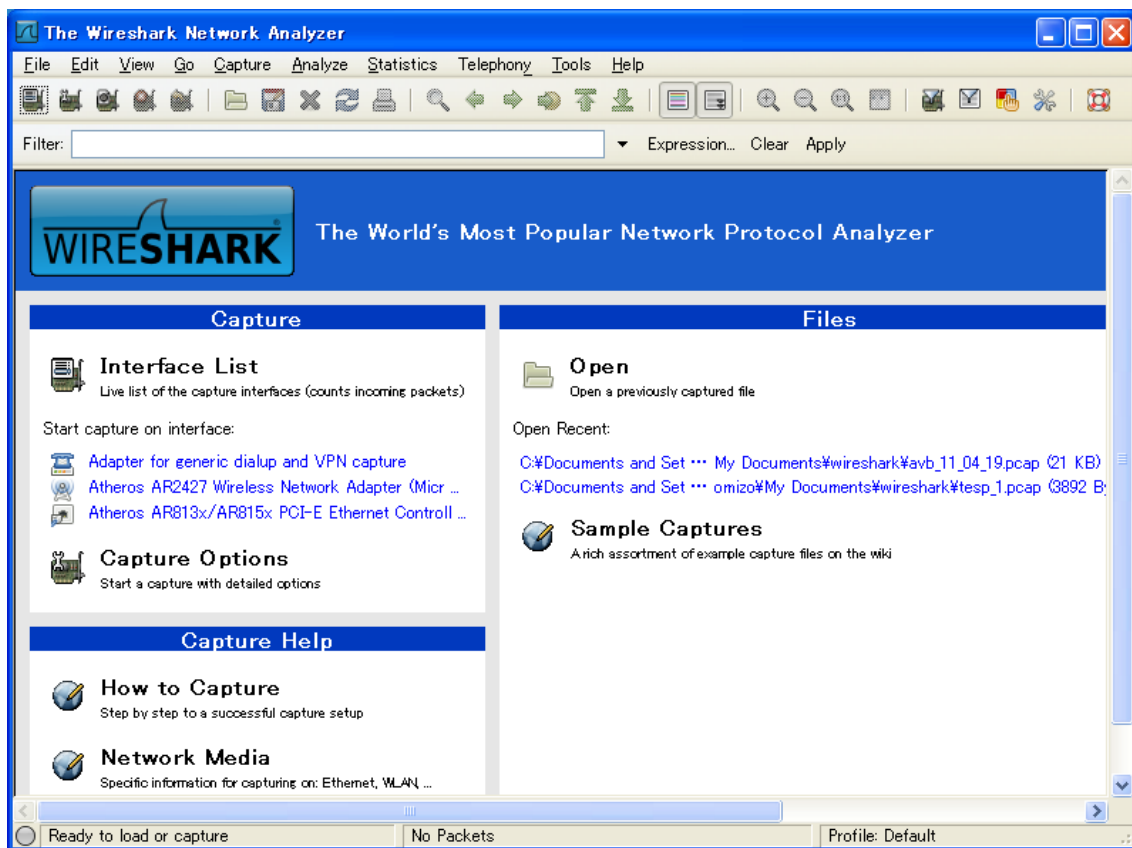
DE0 の GPIO 1 に拡張ボードを接続して、拡張ボードと PC を LAN ケーブルで接続する。

DE0 の電源を入れる。

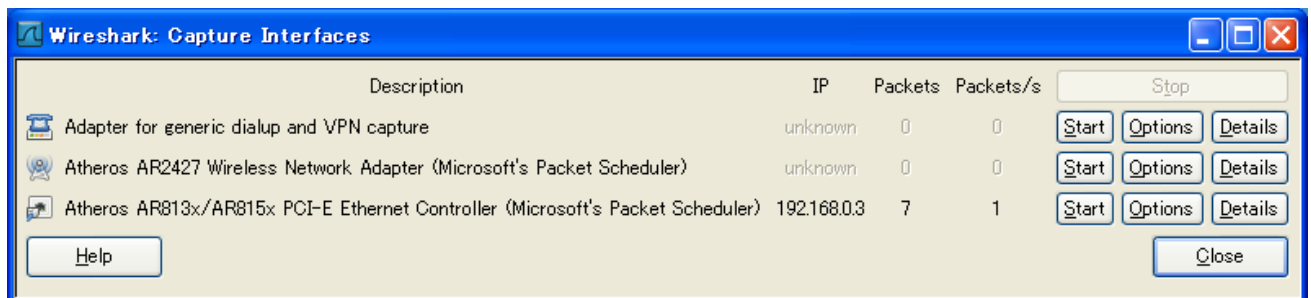
PC のネットワーク設定を固定 IP アドレスに変更



## PC で wireshark の起動



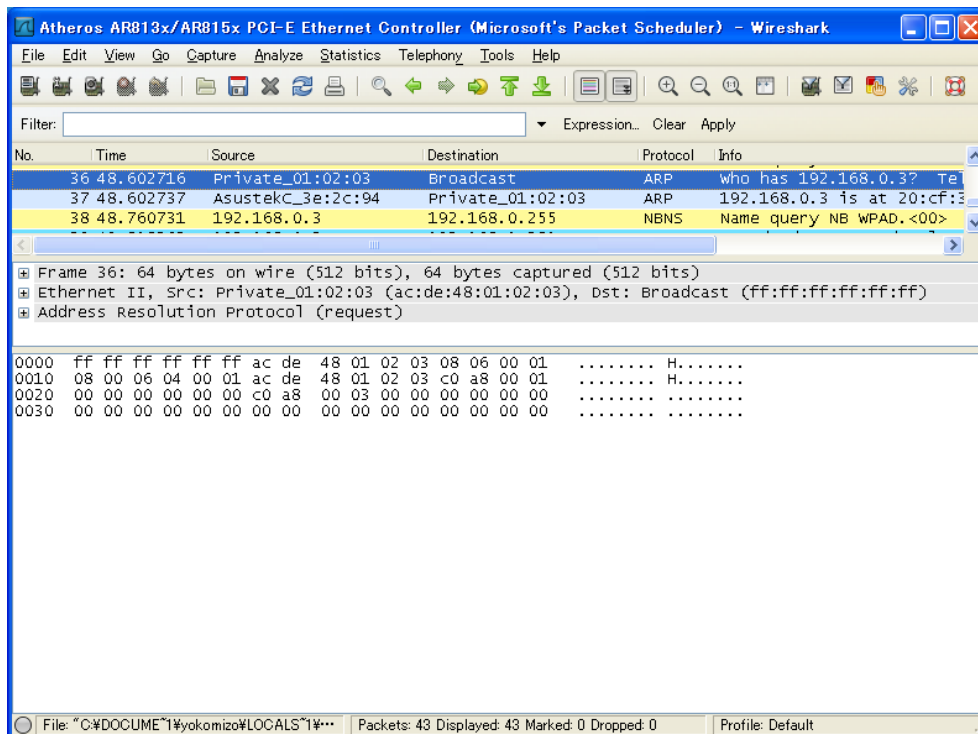
Capture をクリックして、有線 LAN 観測を指定する。



FPGA へのプログラミング実行、Quartus II で FPGA ヘデータ書き込み後、EDS から ethermac\_de0\_sw1 を実行する。

FPGA が起動してプログラムが実行される。

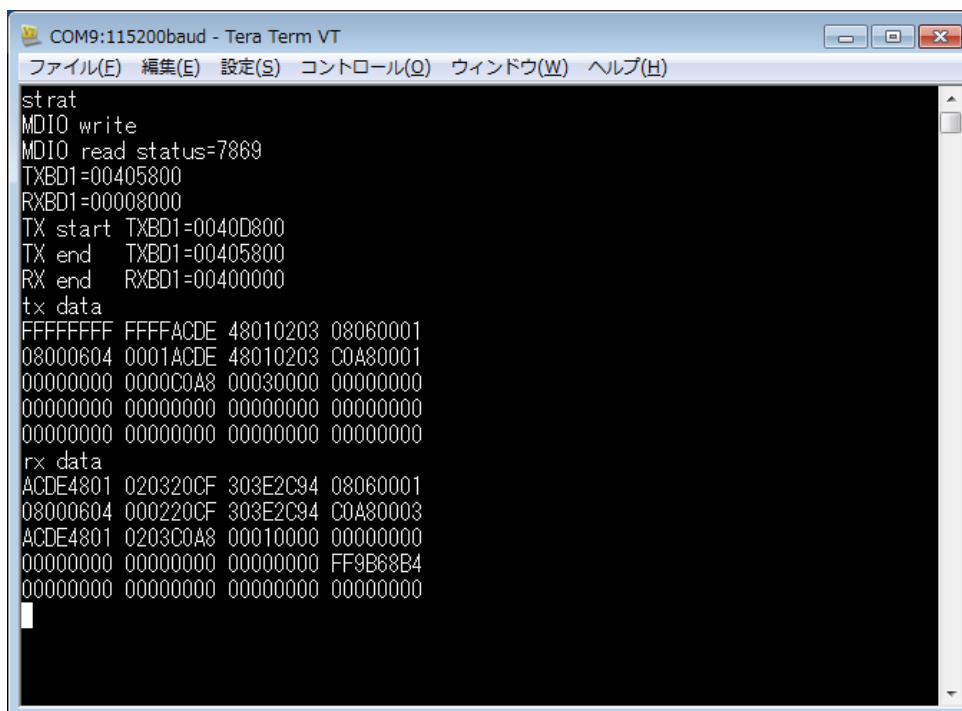
wireshark の表示確認



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



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です。