



MAX II/Cyclone II

EPM240/EP2C5/EP2C8 ボード

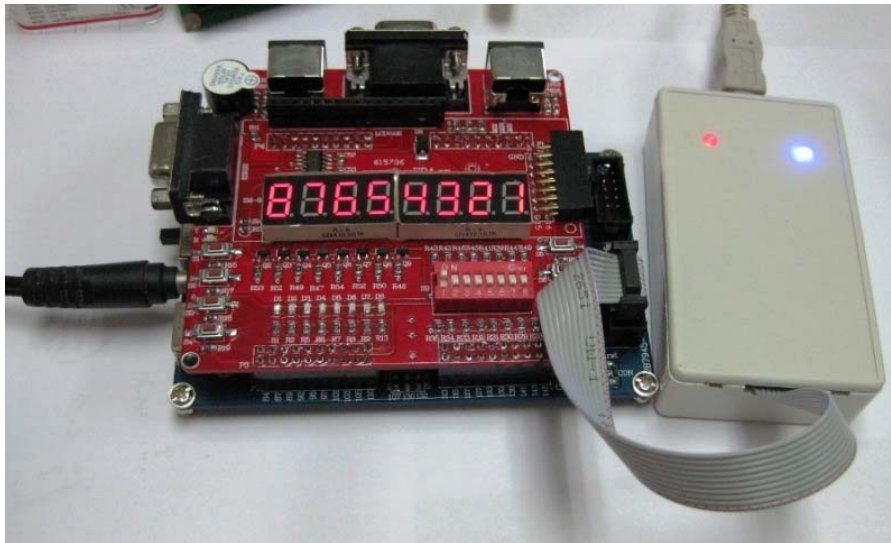
マニュアル

株式会社日昇テクノロジー

<http://www.csun.co.jp>

info@csun.co.jp

2009/11/17



[copyright@2009](#)



| | |
|--|----|
| 第一章 CPLD/FPGAボードの概要..... | 3 |
| 1.1 MAX II/EPM240 ボード..... | 3 |
| 1.2 Cyclone II/ EP2C5-EP2C8 ボード..... | 6 |
| 1.3 CPLD/FPGAの実験用I/Fボード..... | 8 |
| 第二章 開発ツールをインストール..... | 10 |
| 2.1 Quartus II Web Editionをインストールする..... | 10 |
| 2.2 Nios II エンベデッド・デザイン・スイートをインストールする..... | 17 |
| 第三章 MAX II/Cyclone IIの初体験..... | 23 |
| 3.1 Quartus II評価版にソースを読み込む..... | 23 |
| 3.2 USB-Blasterをインストールする..... | 25 |
| 3.3 書き込むソフトウェアを起動する..... | 28 |
| 3.4 FPGAのコンフィギュレーションデバイスに書き込む..... | 30 |
| 3.5 NIOS IIプロセッサの初体験..... | 32 |
| 第四章 CPLD/FPGAの開発入門..... | 40 |
| 4.1 プロジェクトを作成する..... | 40 |
| 4.2 エディタで回路図を描く..... | 47 |
| 4.2.1 トップ・エンティティを作成する..... | 47 |
| 4.2.2 作画手順..... | 50 |
| 4.2 書き込み前の二つの作業..... | 52 |
| 4.2.1 回路図をコンパイルする..... | 52 |
| 4.2.2 回路図の入出力とCPLD/FPGAの端子を関連づける..... | 53 |
| 第五章 NIOS IIシステム・モジュールの設計..... | 54 |
| 第六章 NIOS IIのプログラムの設計..... | 77 |

※ 使用されたソースコードは<http://www.csun.co.jp/>からダウンロードできます。

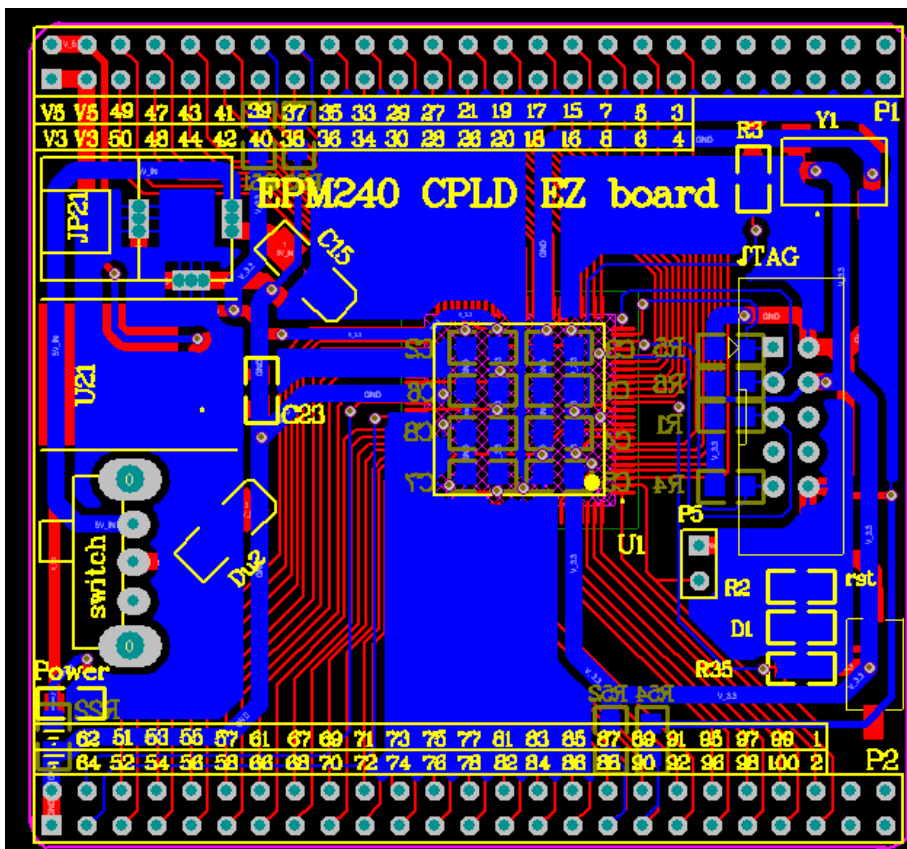
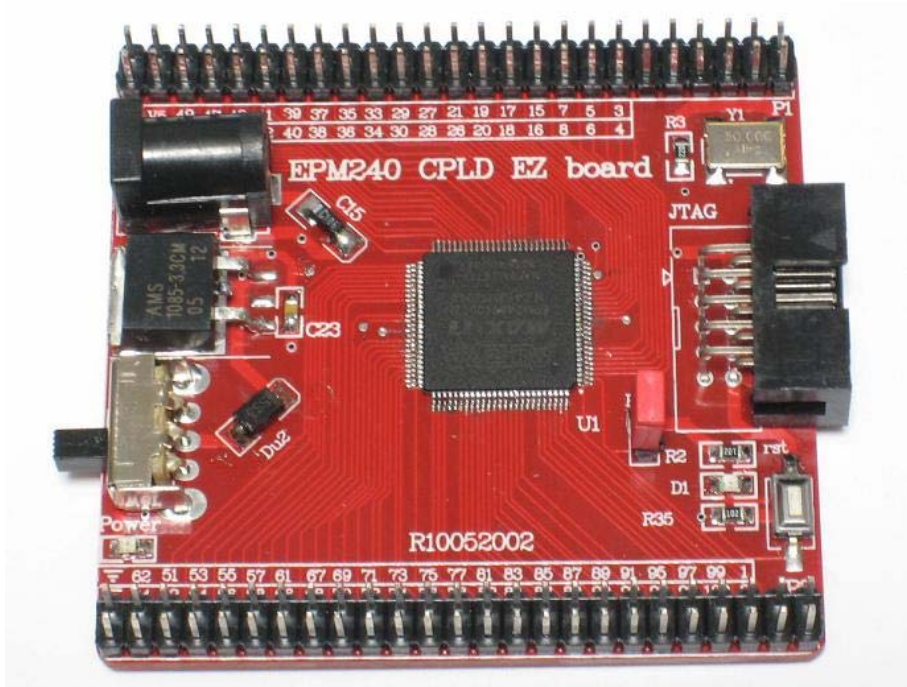
第一章 CPLD/FPGAボードの概要

1.1 MAX II/EPM240 ボード

MAX® II CPLD ファミリは、CPLD ファミリの中で I/O ピンあたり最も低いコストと最少の消費電力を実現し、画期的な新 CPLD アーキテクチャをベースにしている不揮発性、インスタント・オン・プログラマブル・ロジック・ファミリです。この新しいアーキテクチャは、システムの消費電力、スペース、そしてコストの低減を可能にします。

| MAX II デバイス・ファミリの概要 | | | | |
|---------------------|------------|------------|-----------|-----------|
| 特長 | EPM240/G/Z | EPM570/G/Z | EPM1270/G | EPM2210/G |
| ロジック・エレメント(LE)数 | 240 | 570 | 1,270 | 2,210 |
| 標準等価マクロセル数 | 192 | 440 | 980 | 1,700 |
| 最大ユーザ I/O ピン数 | 80 | 160 | 212 | 272 |
| ユーザ・フラッシュ・メモリビット | 8,192 | 8,192 | 8,192 | 8,192 |
| デバイス配給状況 | 出荷中(1) | 出荷中(1) | 出荷中 | 出荷中 |
| tpd1(ns)(コーナ対コーナ性能) | 4.5 | 5.5 | 6 | 6.5 |
| tpd2(ns)(最高速性能) | 3.6 | 3.6 | 3.6 | 3.6 |

MAXII/EPM240 ボードの概要：



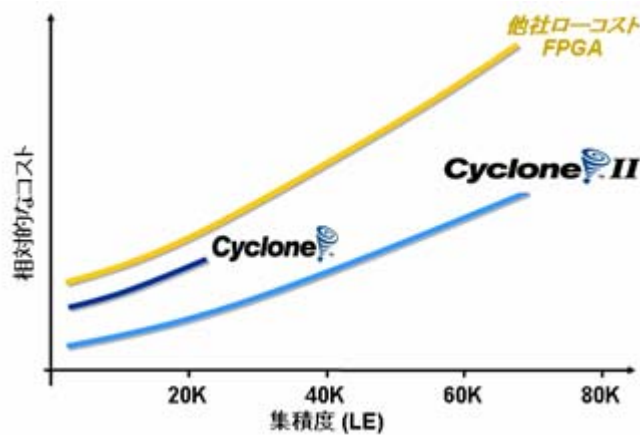


- EPM240 搭載
- 50MHz 水晶発振器搭載
- 5V 電源又は USB ポートで給電、電源スイッチと電源指示 LED 付き
- 3.3V レギュレータ搭載
- JTAG コネクタ
- 回路図を提供しております
- サンプルのソースコードを提供しております。

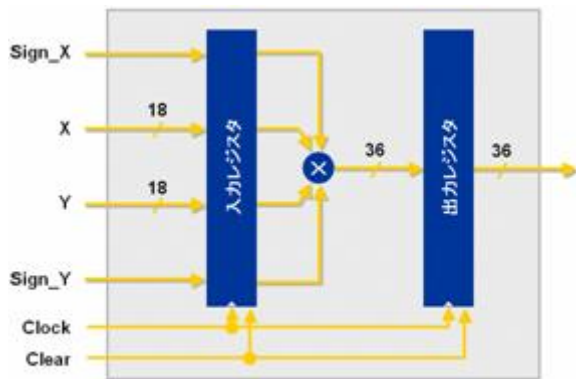
1.2 Cyclone II/ EP2C5-EP2C8 ボード



Cyclone II デバイスは、90-nm テクノロジーの優位性(小型ダイ・サイズ、高集積度、および低コスト)と、低コスト FPGA における最速性能を提供します。すべての Cyclone II デバイスは、TSMC の 90-nm プロセス技術と low-k 低誘電材を使用して 300-mm ウェハ上に製造されています。



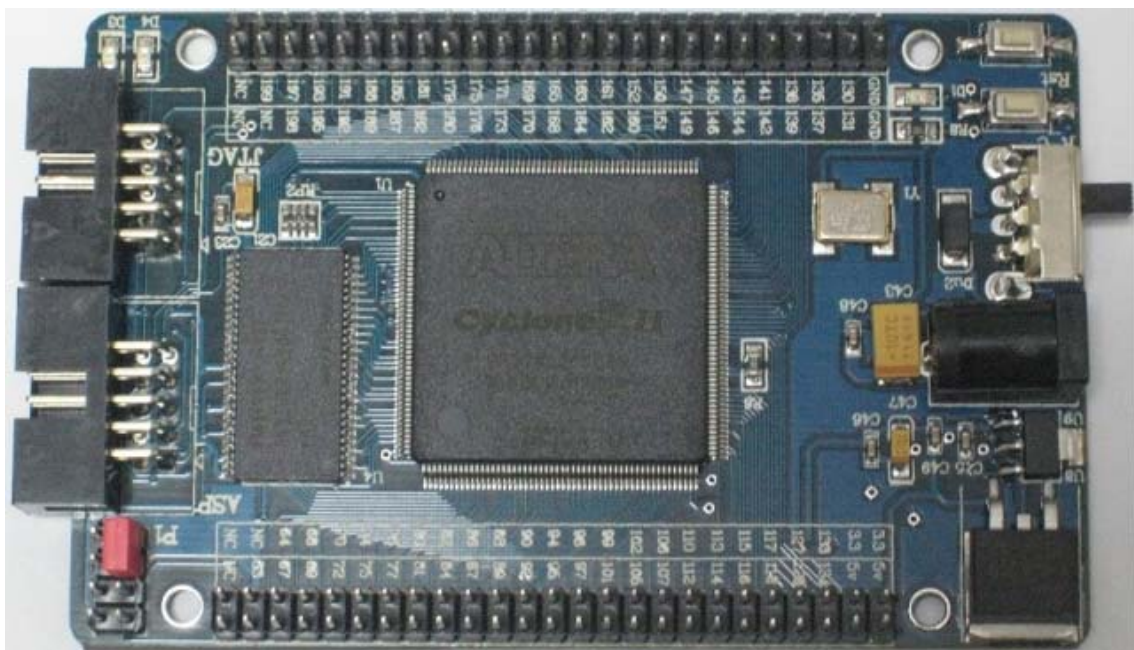
Cyclone II デバイスは、一般的なデジタル信号処理 (DSP) 機能を実装できる、最大 150 個の 18 ビット x 18 ビット・マルチプライヤを備えています。エンベデッド・マルチプライヤは、ロジック・エレメント (LE) ベースのマルチプライヤと比較してより高い性能とロジック効率を提供します。



250 MHz の性能

| 特徴 | デバイス | | | | | |
|---------------|---------|---------|---------|---------|---------|-----------|
| | EP2C5 | EP2C8 | EP2C20 | EP2C35 | EP2C50 | EP2C70 |
| ロジック・エレメント数 | 4,608 | 8,256 | 18,752 | 33,216 | 50,528 | 68,416 |
| M4K RAM ブロック数 | 26 | 36 | 52 | 105 | 129 | 250 |
| RAM 総ビット数 | 119,808 | 165,888 | 239,616 | 483,840 | 594,432 | 1,152,000 |
| エンベデッド乗算器数 | 13 | 18 | 26 | 35 | 86 | 150 |
| PLL 数 | 2 | 2 | 4 | 4 | 4 | 4 |

Cyclone II/EP2C5-EP2C8 ボードの概要 :



- EP2C5Q208 又は EP2C8Q208 搭載
- NIOS II ソフトプロセッサ搭載
- コンフィギュレーションデバイス EPCS4(4Mbit)搭載
- SDRAM(64Mbit)搭載
- 50MHz 水晶発振器搭載
- 5V 電源又は USB ポートで給電、電源スイッチと電源指示 LED 付き
- 3.3V/1.2V レギュレータ搭載
- JTAG と ASP コネクタ
- ユーザ LED とユーザボタン
- 回路図を提供しております
- サンプルのソースコードを提供しております。

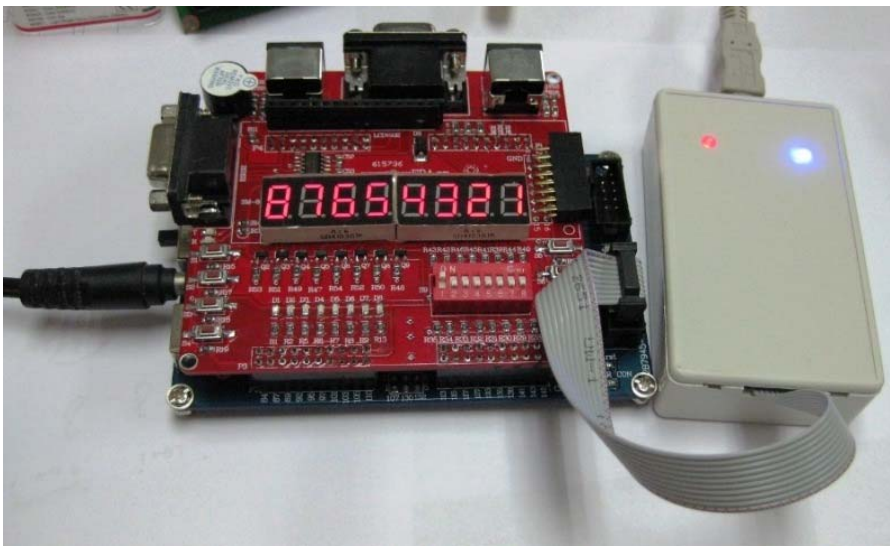
1.3 CPLD/FPGAの実験用I/Fボード



- LED x 8
- 8ビット DIP スイッチ

- ボタン x 8
- PS2 x 2、キーボード又はマウスを接続可
- VGA x 1
- RS232 x 1
- 1602 液晶 I/F x 1
- 7セグメント LED x 8
- Beep x 1
- 回路図を提供しております

実験用 I/F ボードが EPM240 又は EP2C5Q208 又は EP2C8Q208 ボードと一緒に動く様子。



第二章 開発ツールをインストール

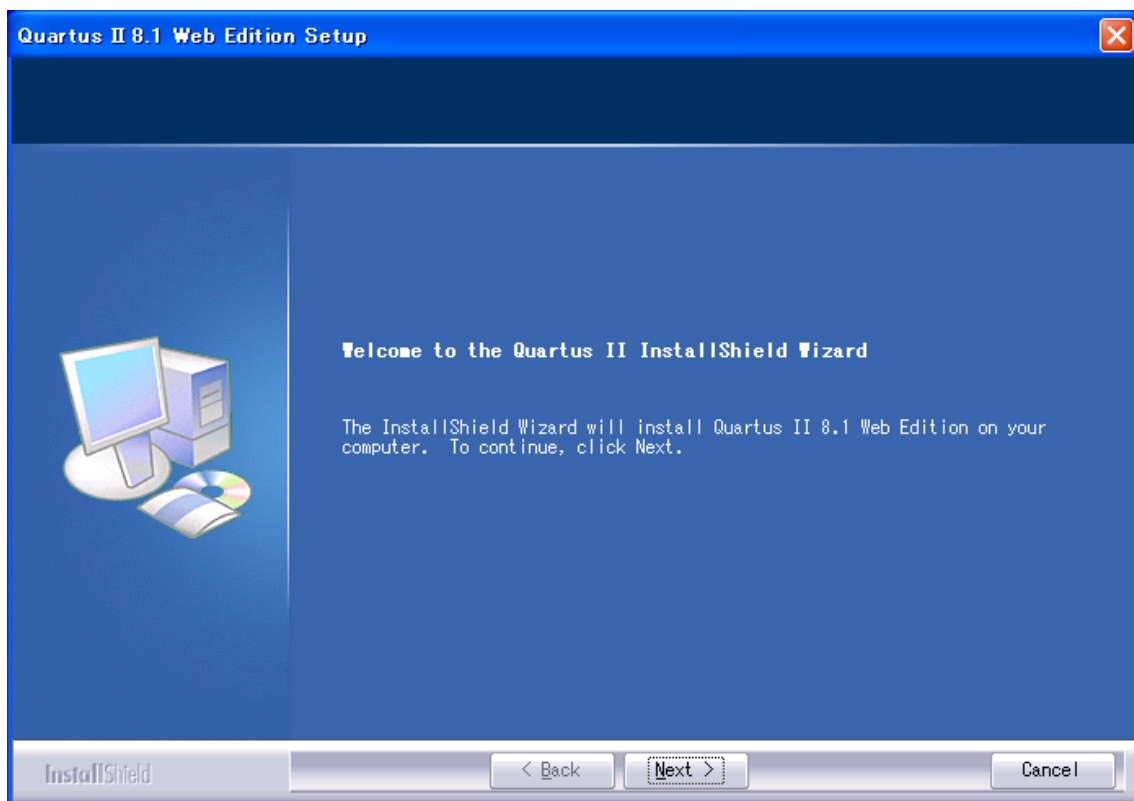
CPLD/FPGA の開発には、ALTERA から Quartus II Web Edition という無償版のツールが公開されているのでこちらを利用します。Quartus II には別に製品版があり、Web Edition は使用できるデバイスなどに制限がありますが、MAX II と Cyclone II に関しては、どのデバイスも使用できるのでまったく問題ありません。Quartus II Web Edition は、総合開発環境になっており、このソフトウェアだけで、ソース・エディタや I/O ピンのアサインメント、論理合成、デバイスの書き込み用のプログラムなど、PLD/FPGA の開発に必要な機能がすべて含まれています。また、Nios II エンベデッド・デザイン・スイートは Nios プロセッサ用の開発ツールです。

Quartus II Web Edition と Nios II エンベデッド・デザイン・スイートのダウンロードは、次の URL から行うことができます。

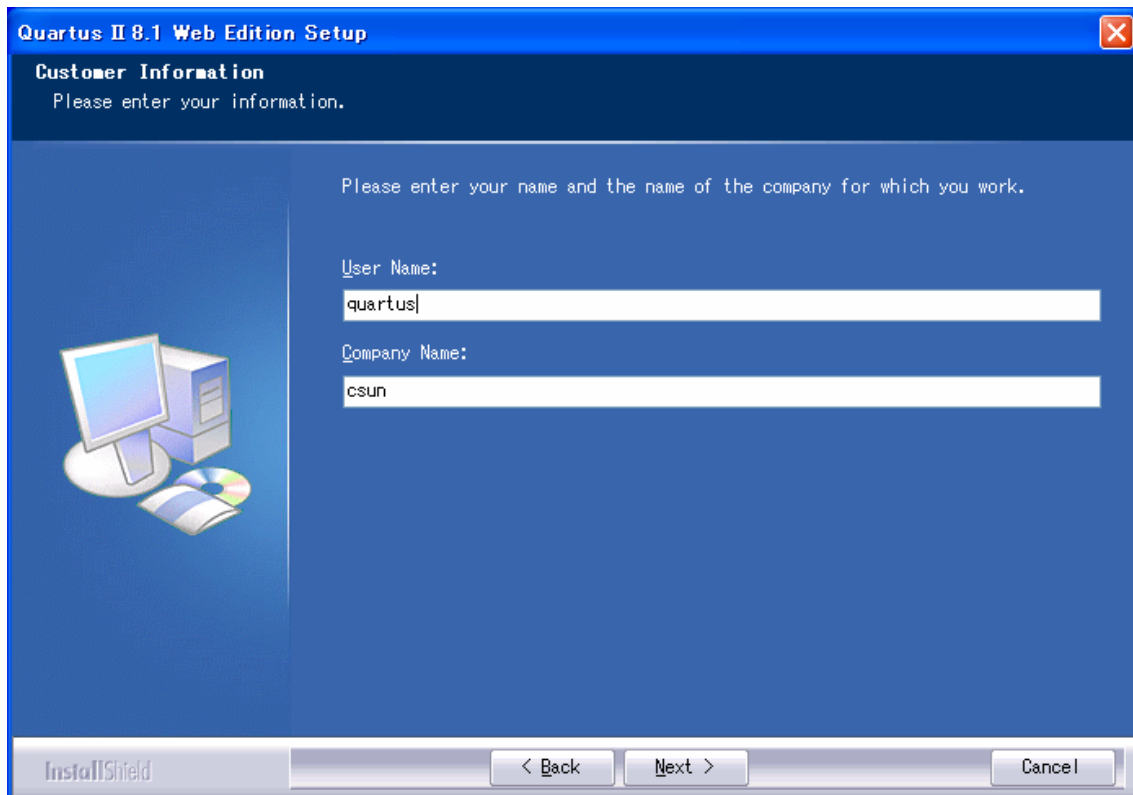
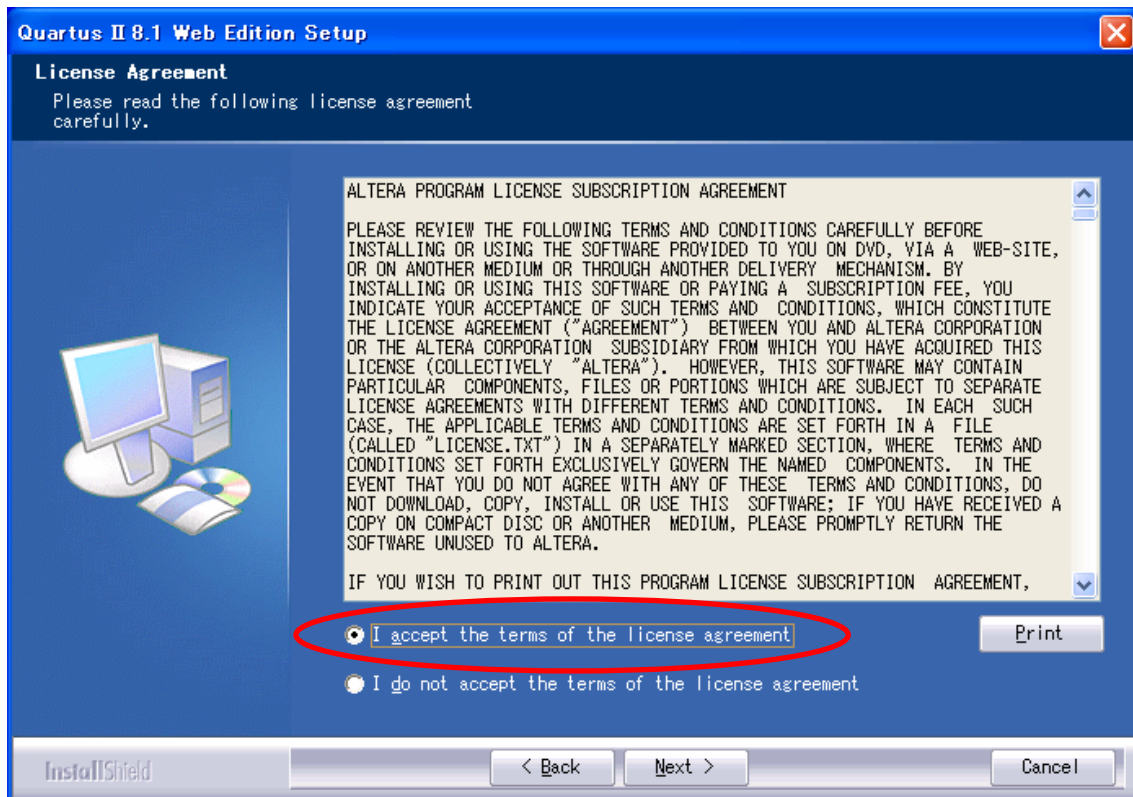
<http://www.altera.co.jp/support/software/download/nios2/dnl-nios2.jsp>

なお、ダウンロードする際は、最初に ALTERA のページにサイン・インを行い、ユーザ情報を登録する必要があります。現時点最新版は v8.1 です。インストールした後、ライセンス・ファイルが不要です。

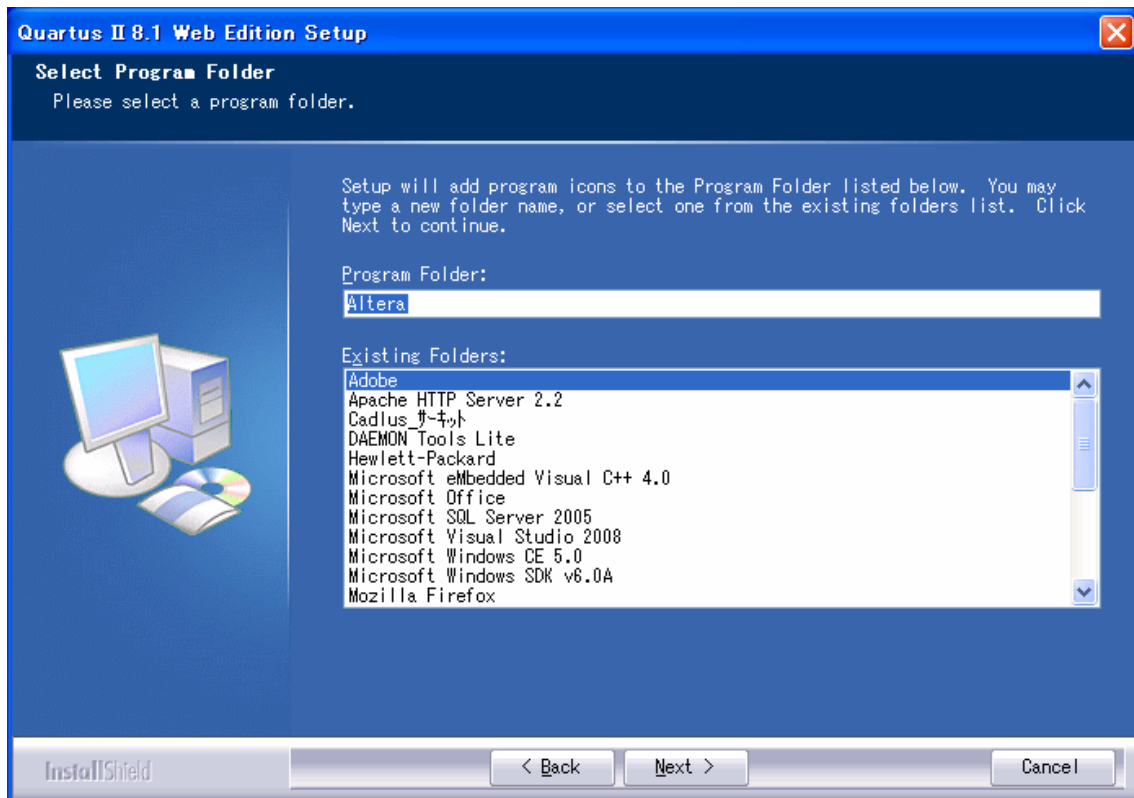
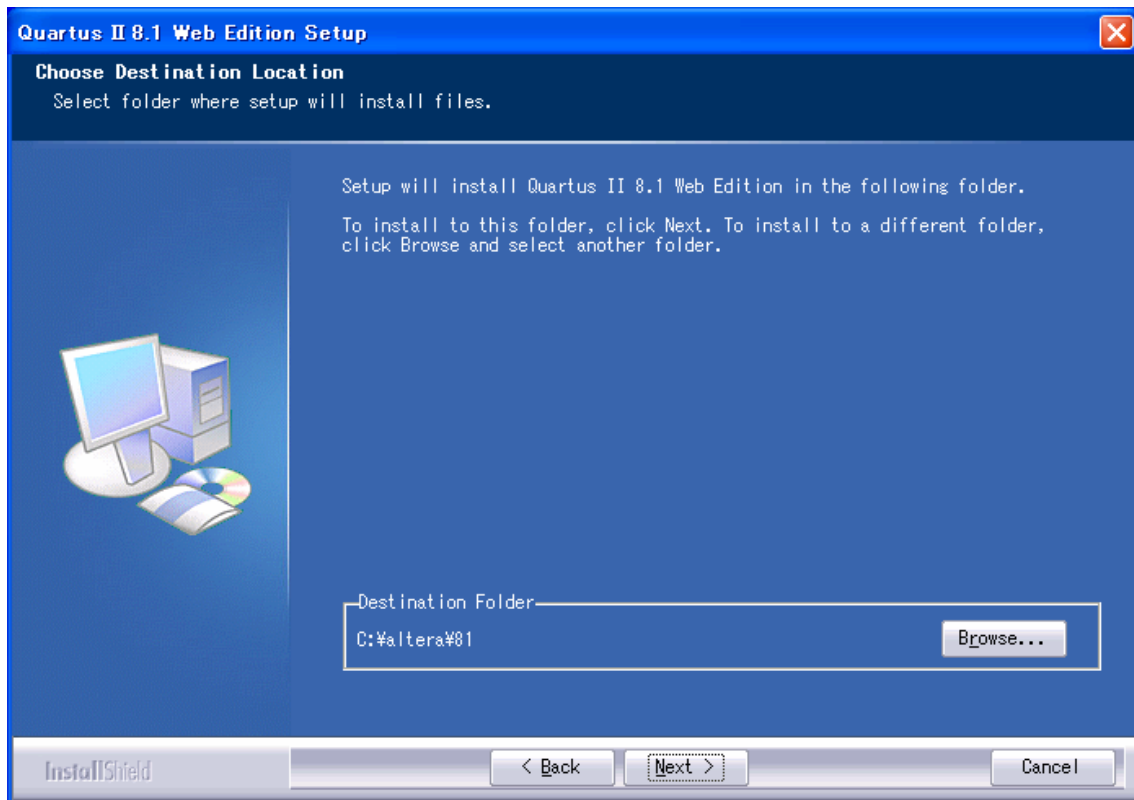
2.1 Quartus II Web Editionをインストールする



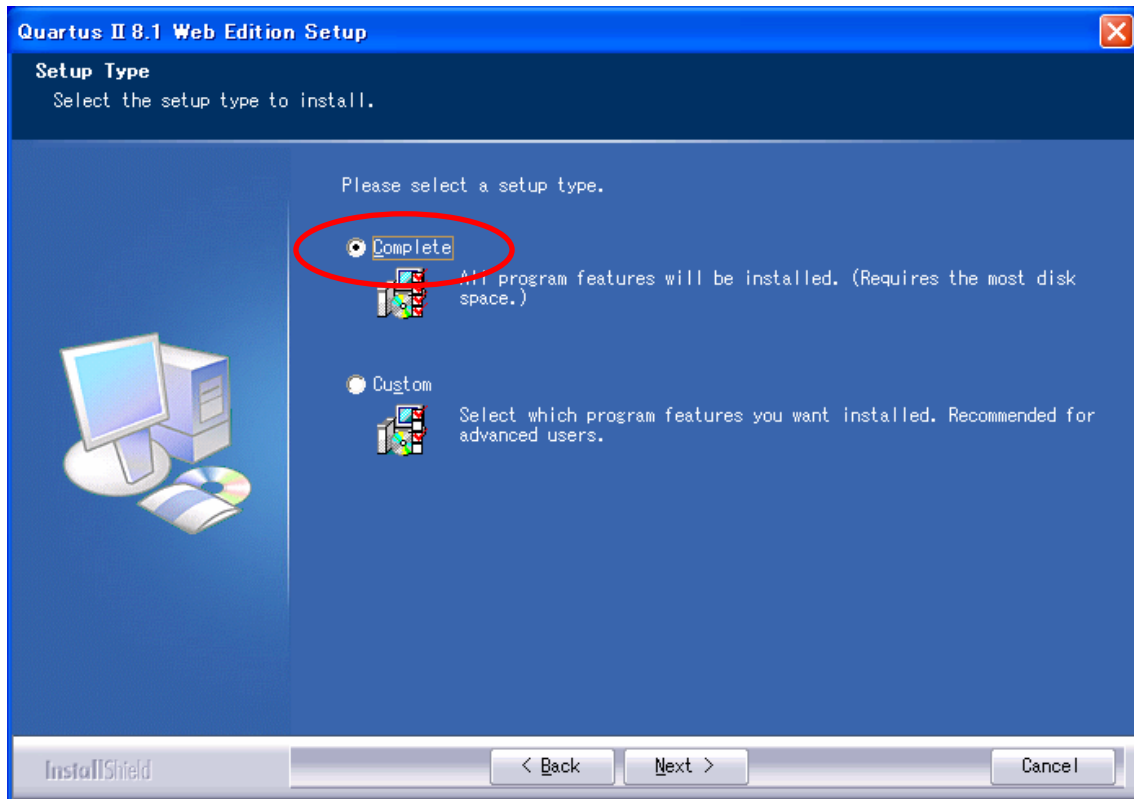
「Next」ボタンを押すと、英文のライセンスが出てきます。同意できる場合は、「I accept the terms of the license agreement」を選択して、「Next」ボタンを押します。



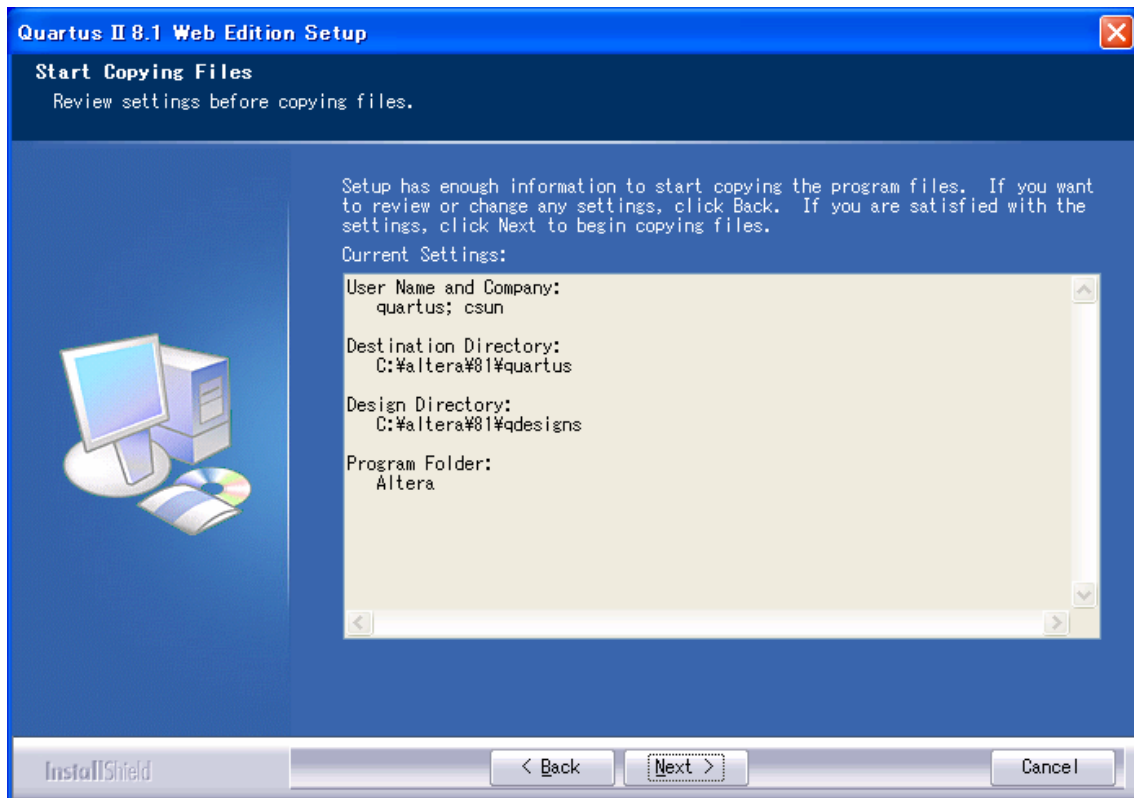
使用者の名前と所属会社名を入力するダイアログが表示されます。名前は半角のアルファベットで入力しましょう。



インストール先フォルダを変更せず、そのまま進んでください。



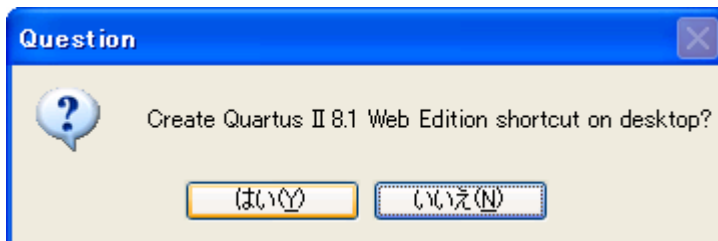
「**Complete**」を選択してください。



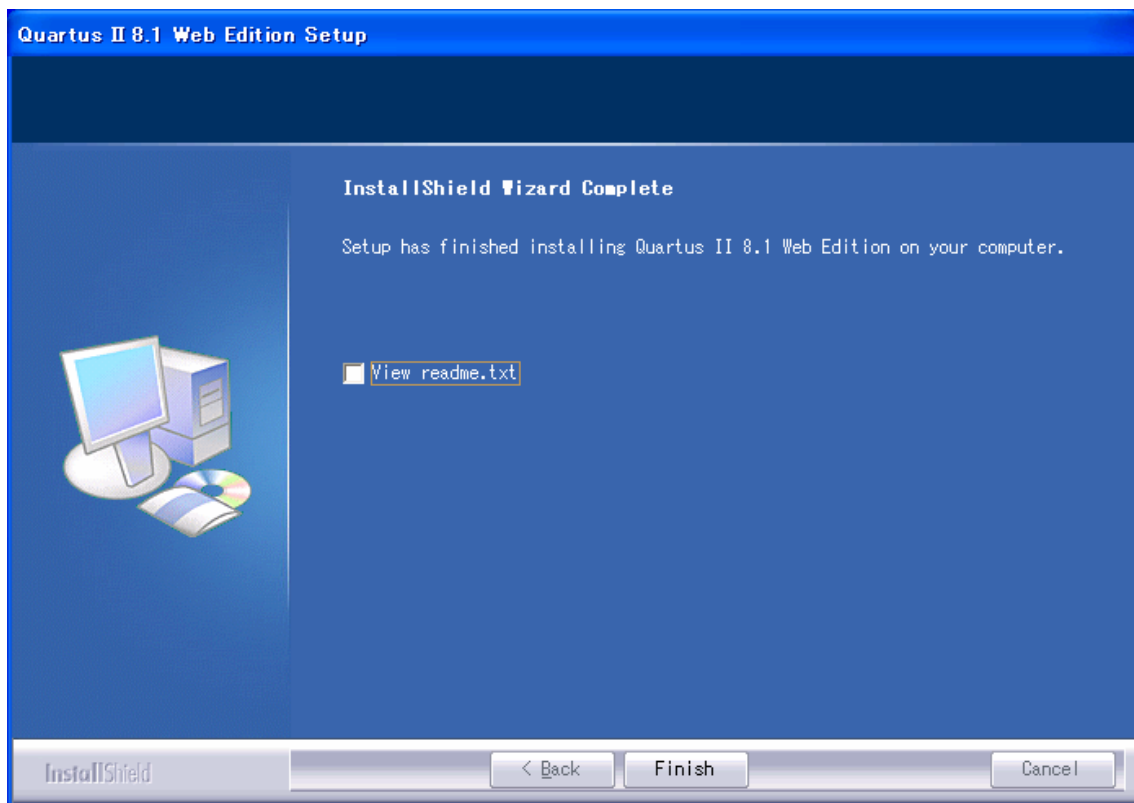
間違いがないかどうか確認し、問題がなければ「Next」を押します。



インストール中の画面です。



インストール完了すると、ショートカットをデスクトップに作るかどうか聞かれます。どちらでも選択できます。

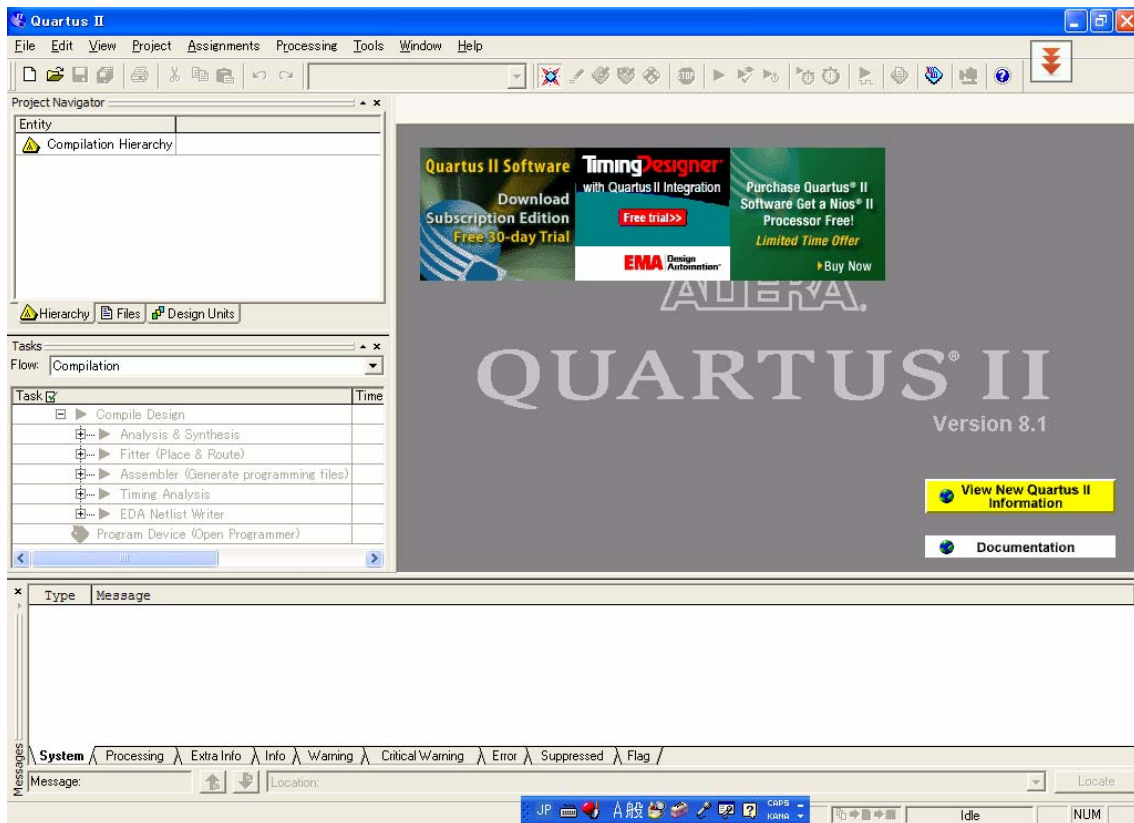


最後に「Finish」をクリックすると、ウィザードが閉じてインストールが終了します。

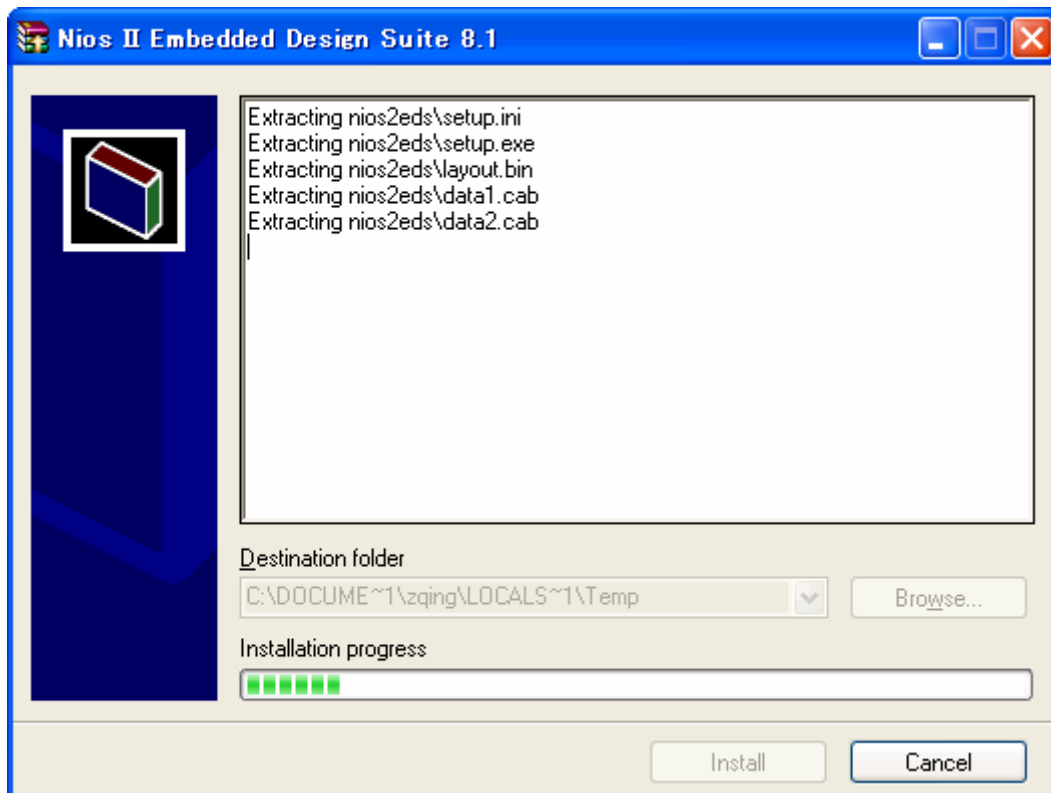
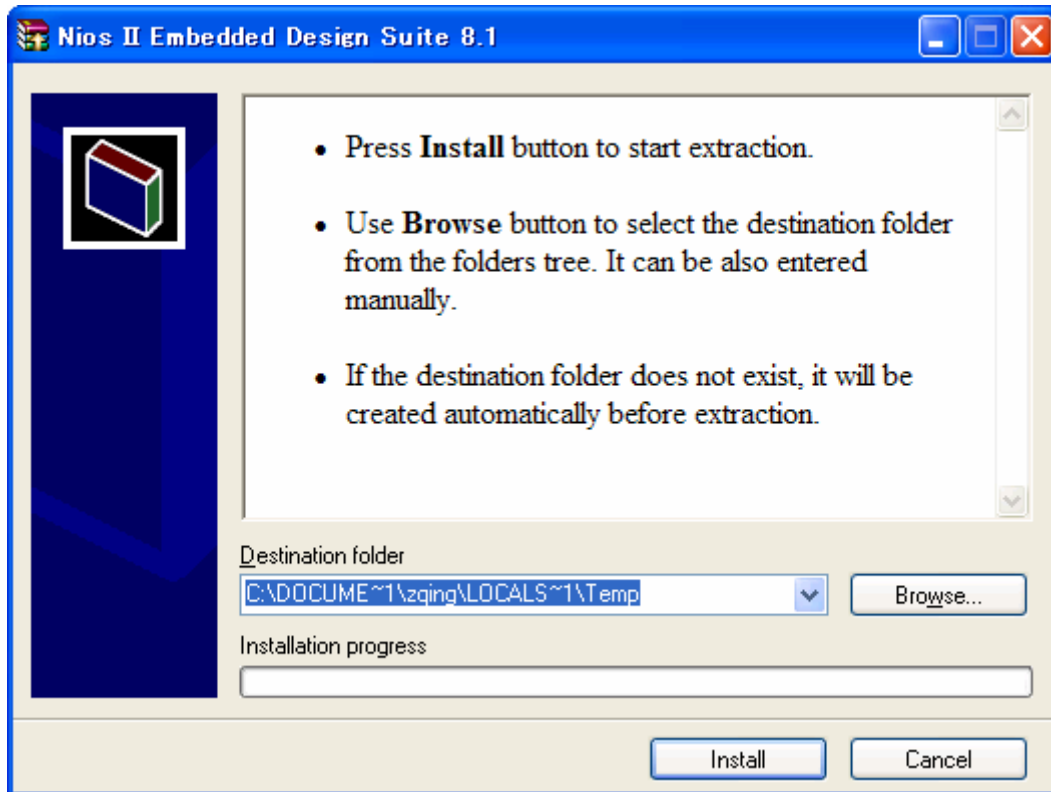
インストールされた Quartus II 評価版をさっそく起動してみます。一番最初に起動したときだけ、次のようなダイアログが現れ、「Run the Quartus II software」を選択してください。「OK」ボタンを押します。



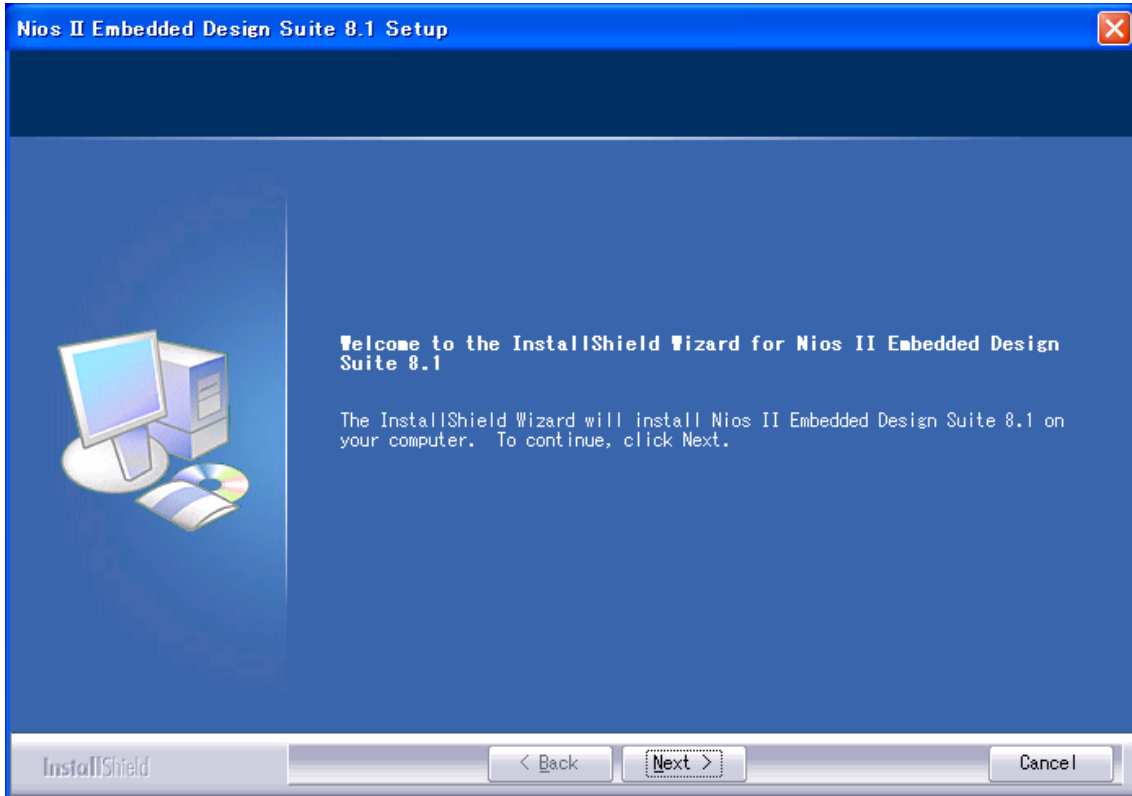
Quartus II の画面出てきます。



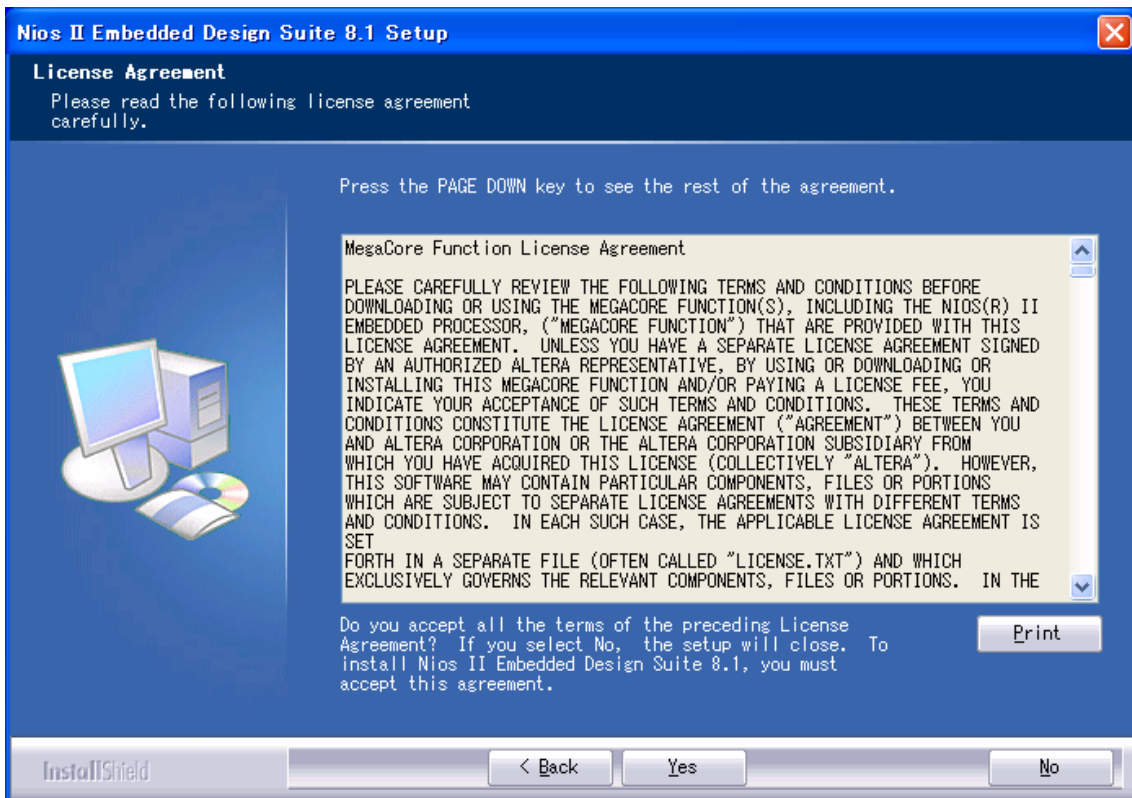
2.2 Nios II エンベデッド・デザイン・スイートをインストールする

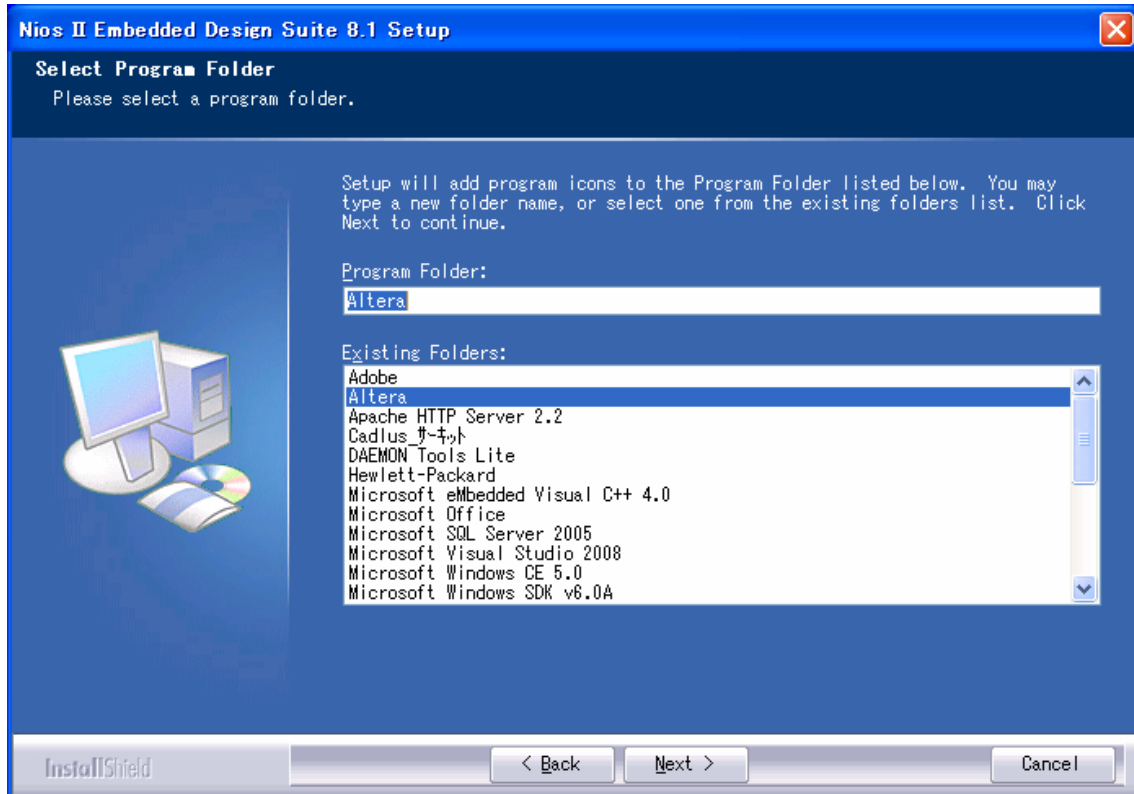
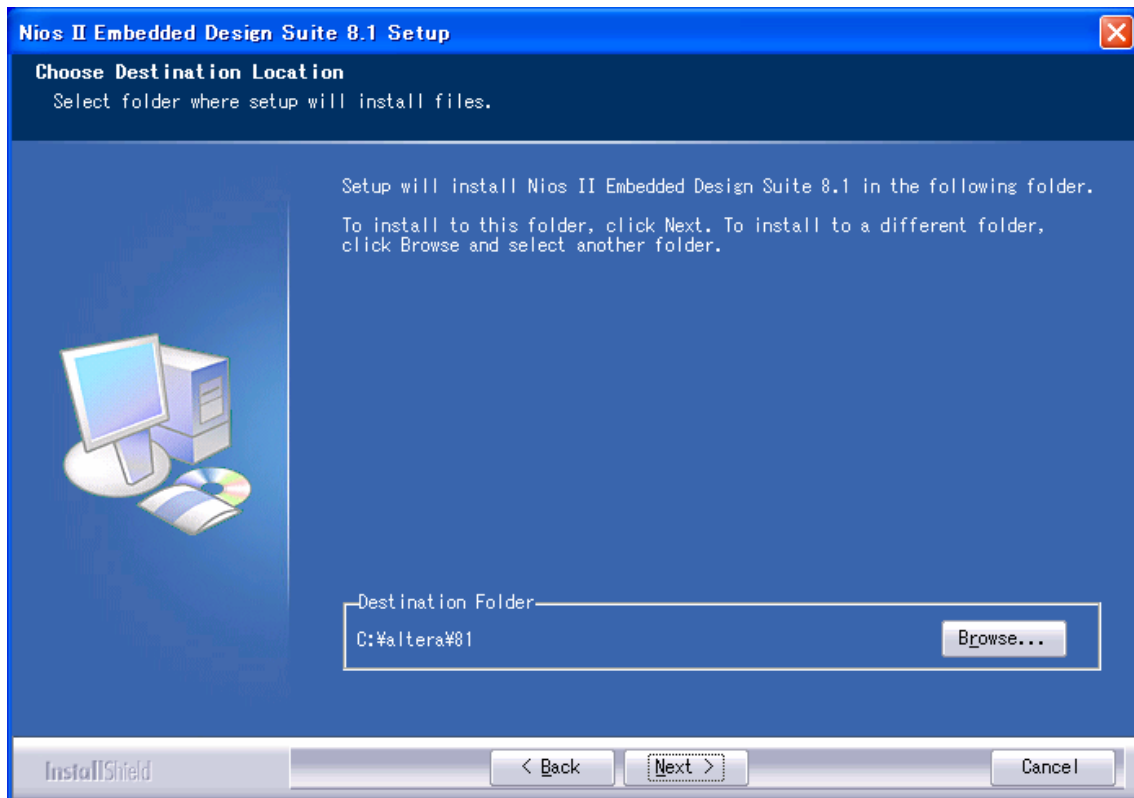


先ず「Install」ボタンを押して解凍します。「Next」ボタンを押します。

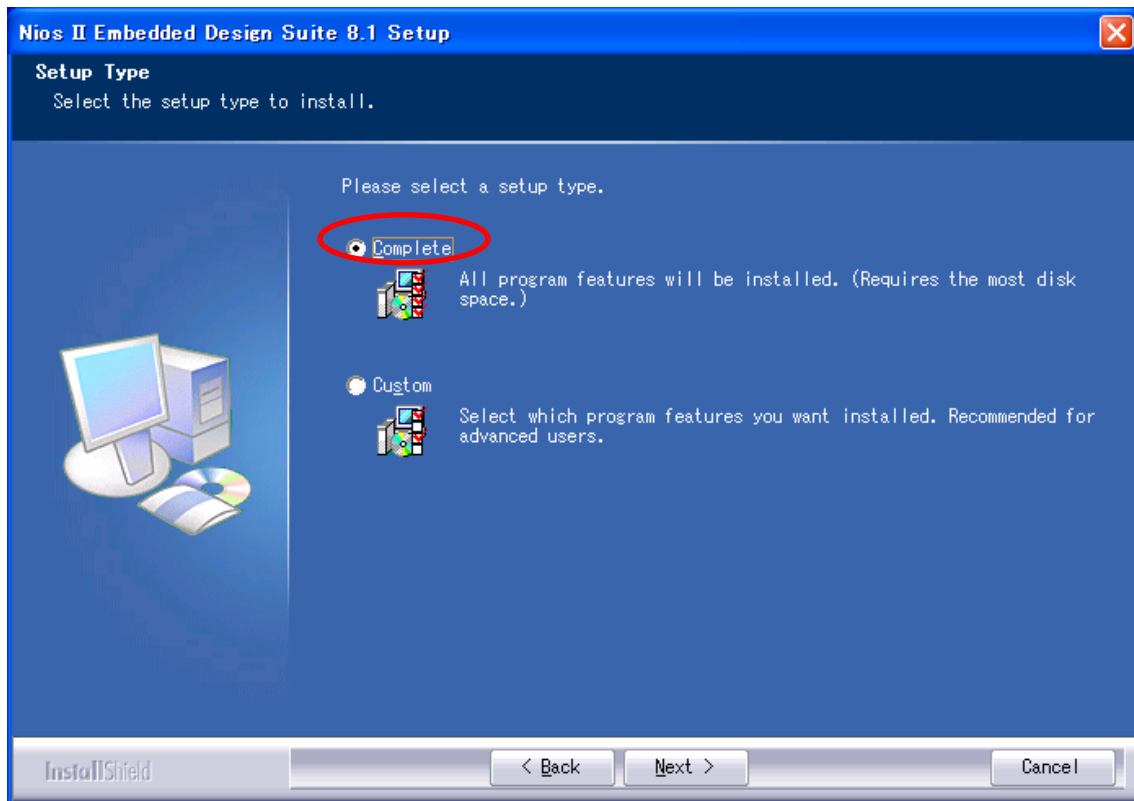


ライセンスを同意すれば、「Yes」ボタンを押します。

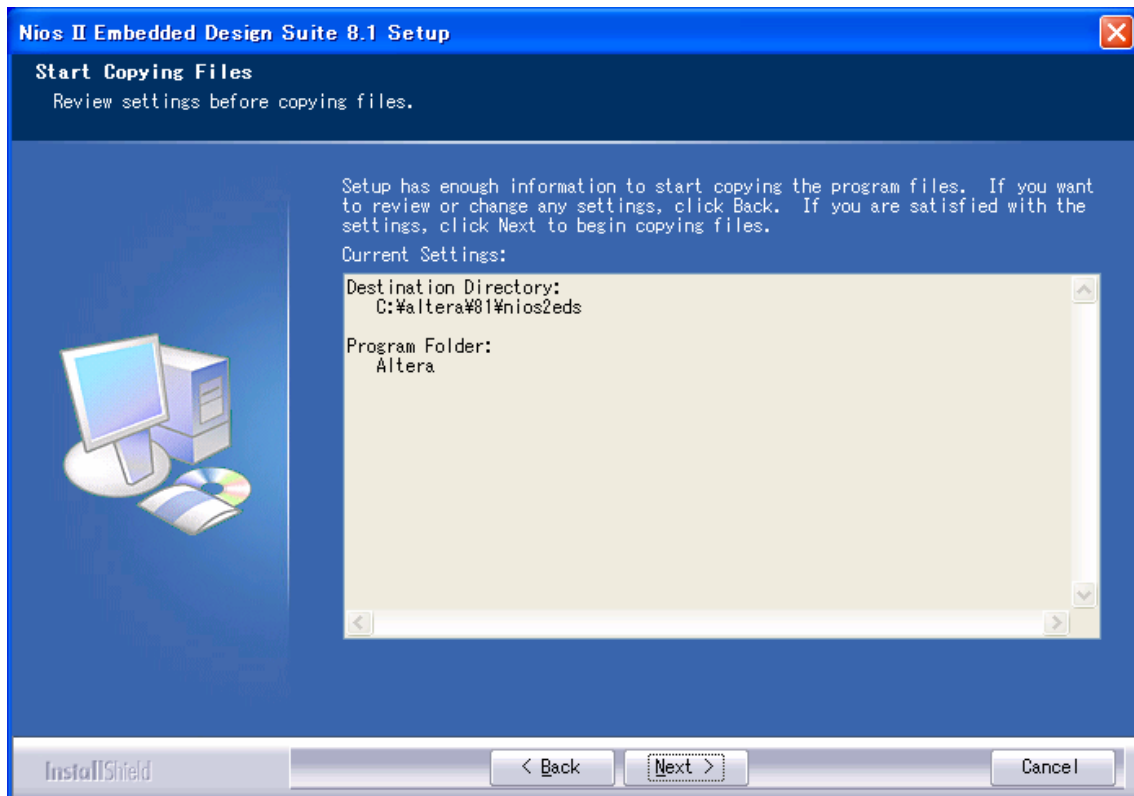




インストール先フォルダを変更せず、そのまま進んでください。



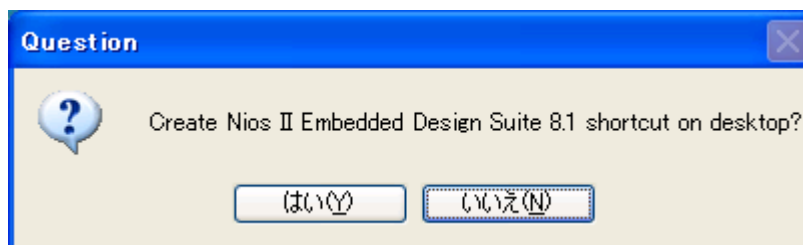
「Complete」を選択してください。



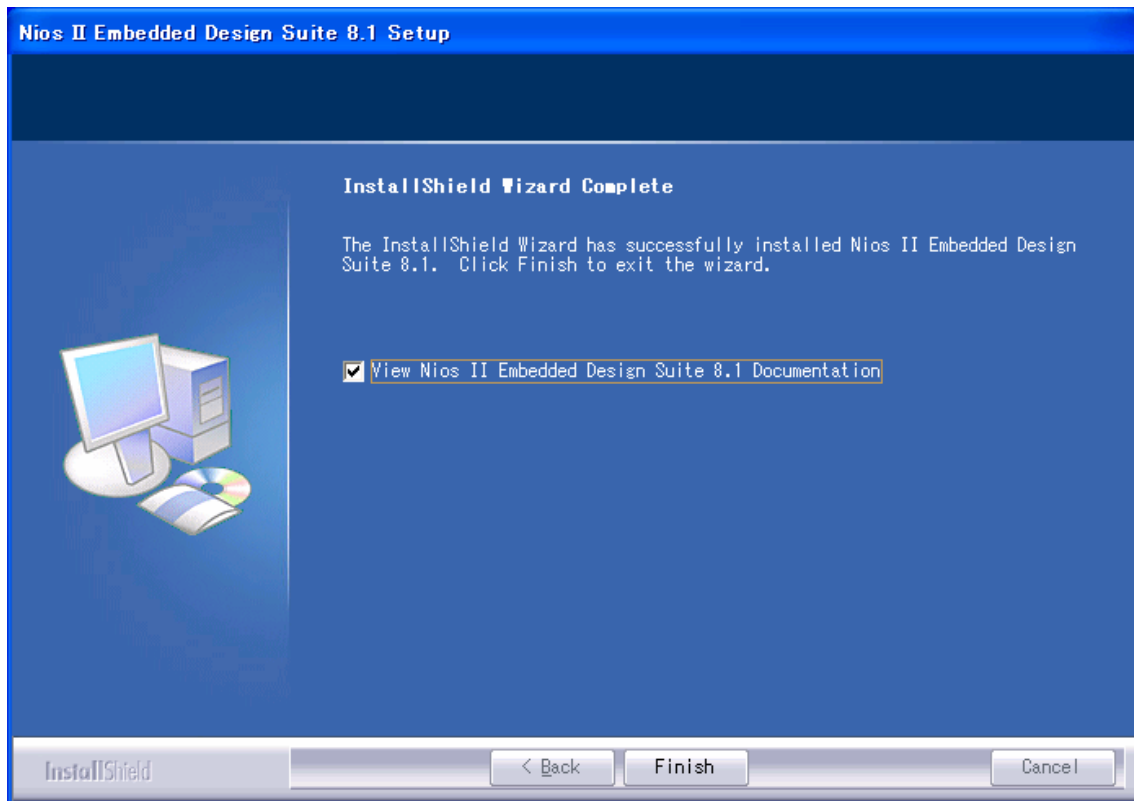
間違いがないかどうか確認し、問題がなければ「Next」を押します。



インストール中。



インストール完了すると、ショートカットをデスクトップに作るかどうか聞かれます。どちらでも選択できます。



最後に「Finish」をクリックすると、ウィザードが閉じてインストールが終了します。

第三章 MAX II/Cyclone IIの初体験

3.1 Quartus II評価版にソースを読み込む

弊社のウェブサイトではMAX II 又は Cyclone II 用のサンプル・ソース・ファイルをダウンロードできます。

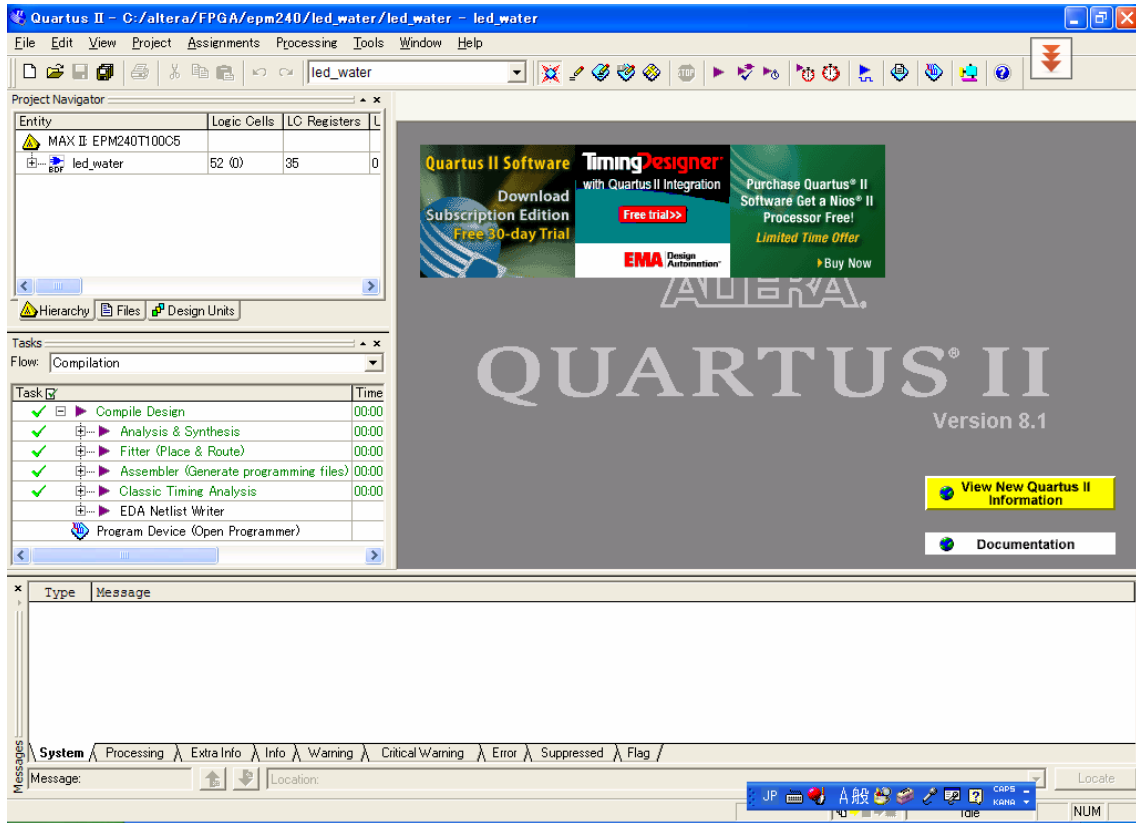
Cyclone II 用ソース・ファイル：EP2C5.zip , EP2C8.zip

MAX II 用ソース・ファイル：epm240.zip

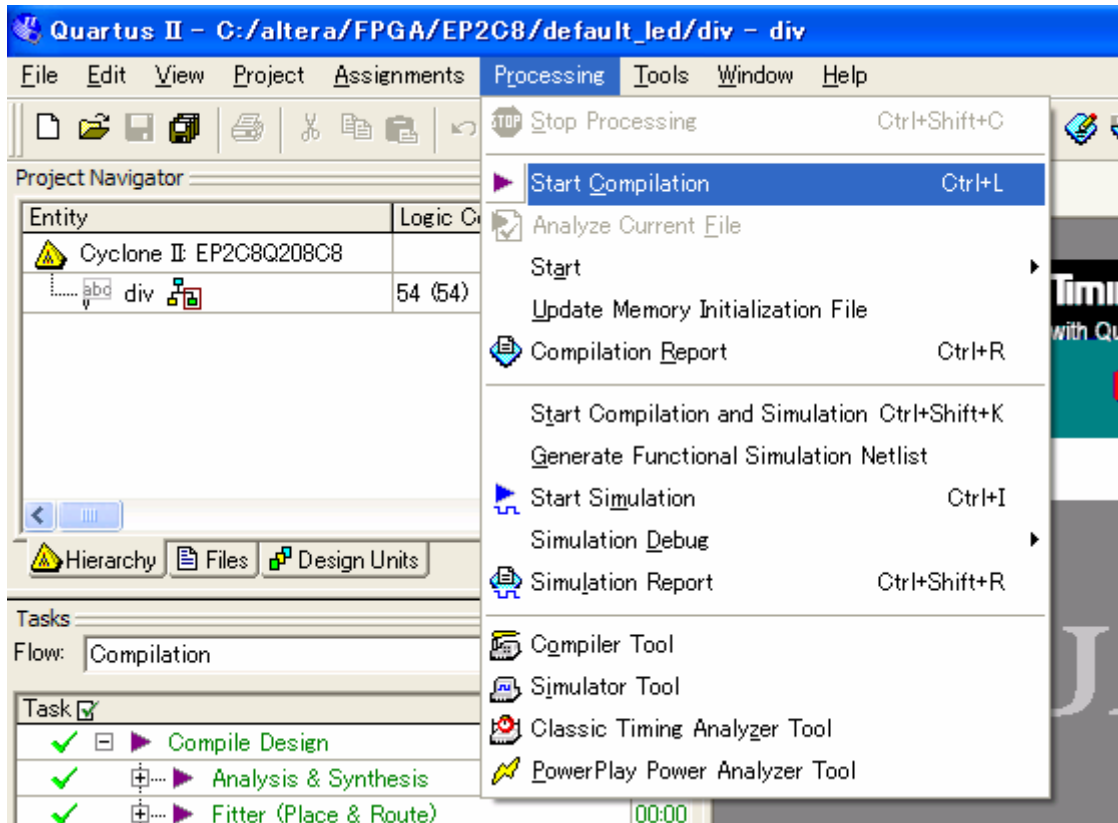
ソース・ファイルを C:\¥altera に展開します。その中に、幾つのサンプルがあります。例えば：Beep、LED、VGA、LCD、UART、Key など。一つのサンプルを紹介いたします。

エクスプローラまたはマイ コンピュータを起動して、
C:\¥altera¥epm240¥led_water
というフォルダを開いてください。

これらの中に、名前が led_water.qpf、Quartus II Project File となっているファイルがあります。これをダブル・クリックすると、Quartus II が起動して、led_water というプロジェクトが開きます。

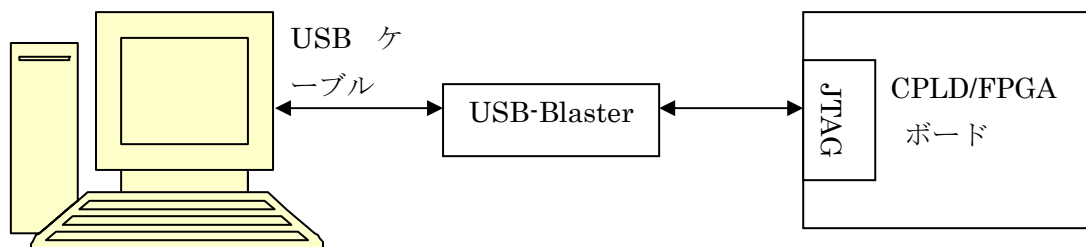


Quartus II の「Processing」メニューから「Start Compilation」を選択します。するとコンパイル処理が始まり、プロGRESS・バーが働き始めます。コンパイルは数十秒で終了します。

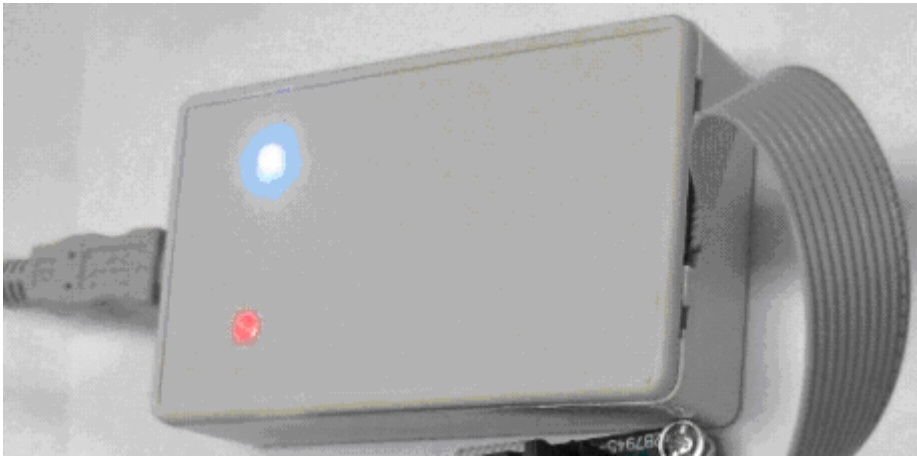


3.2 USB-Blasterをインストールする

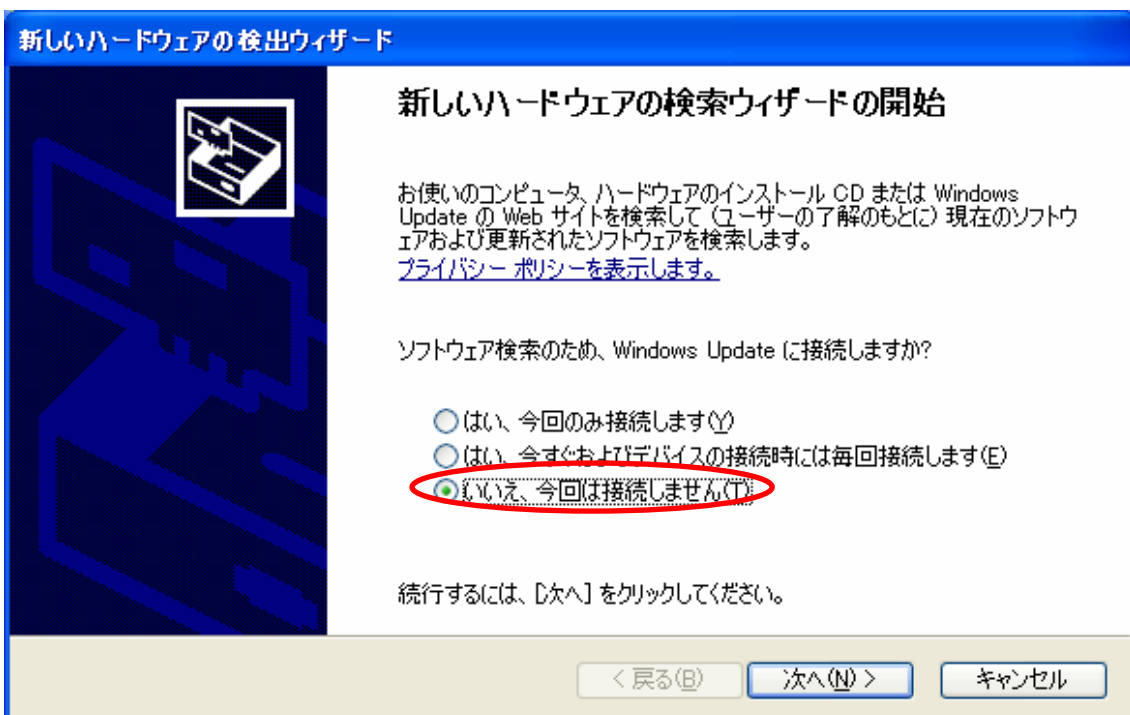
通常、MAX II/Cyclone II にコンフィグレーション・データを書き込むために、アルテラが発売している専用ダウンロード・ケーブル(ByteBlaster MV や ByteBlasterII や USB 接続タイプの USB-Blaster など)を購入しなければなりません。

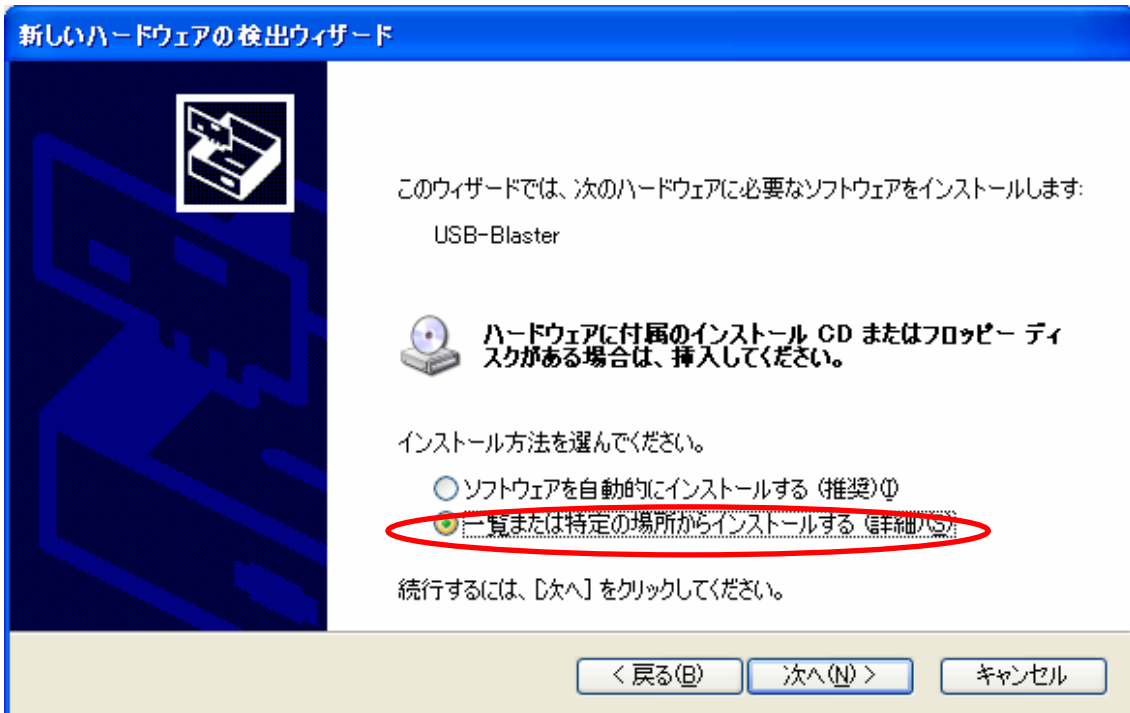


弊社は専用ダウンロード・ケーブル USB-Blaster 同等のデバイスを提供しております。次に示す手順に従って、USB-Blaster のデバイス・ドライバをインストールしてください。

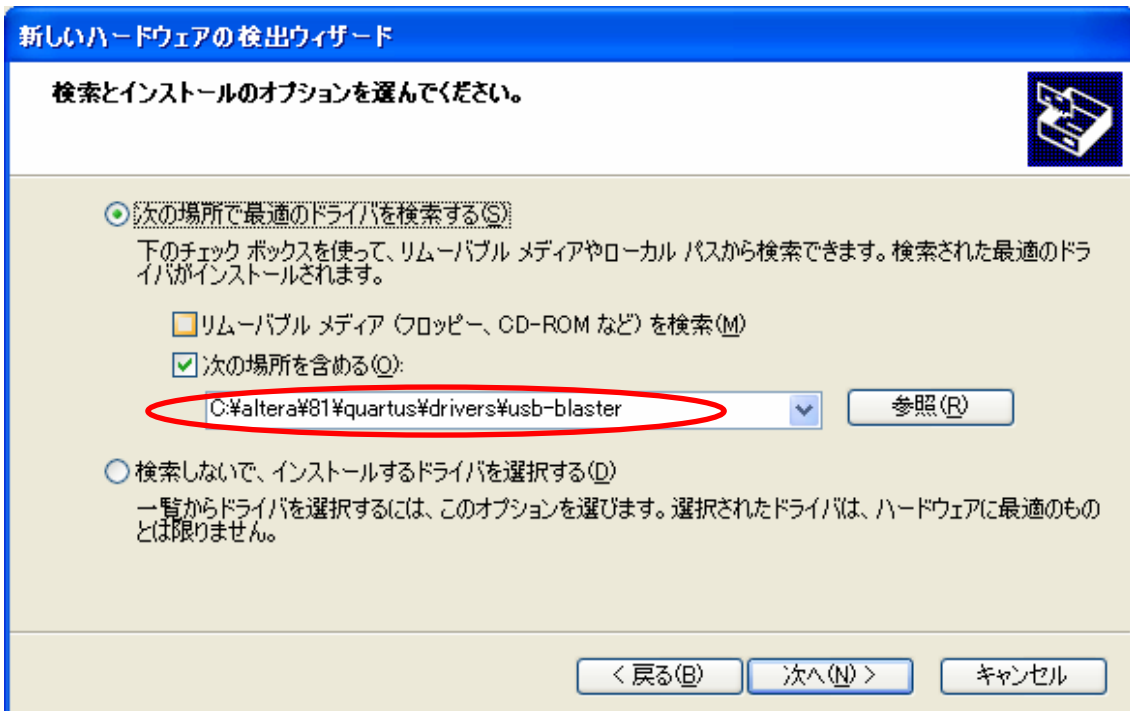


USB-Blaster を USB ケーブルでパソコンと繋ぐと、自動的にこの画面が現れ、「いいえ、今回は接続しません」を選択してください。

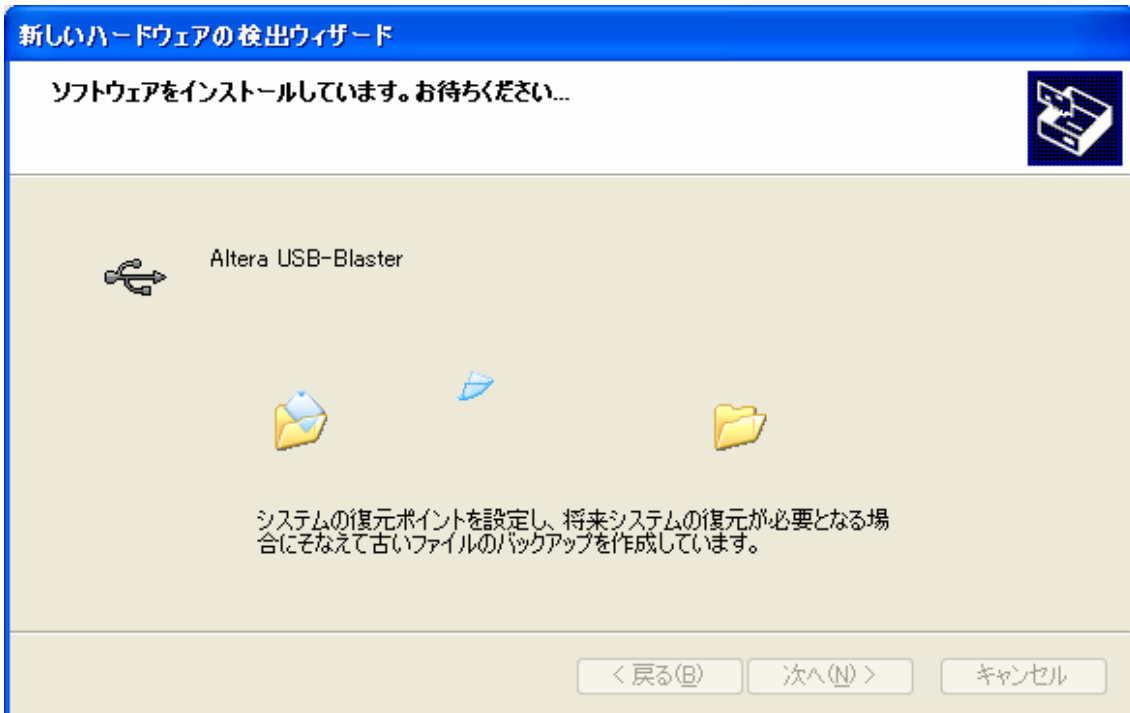




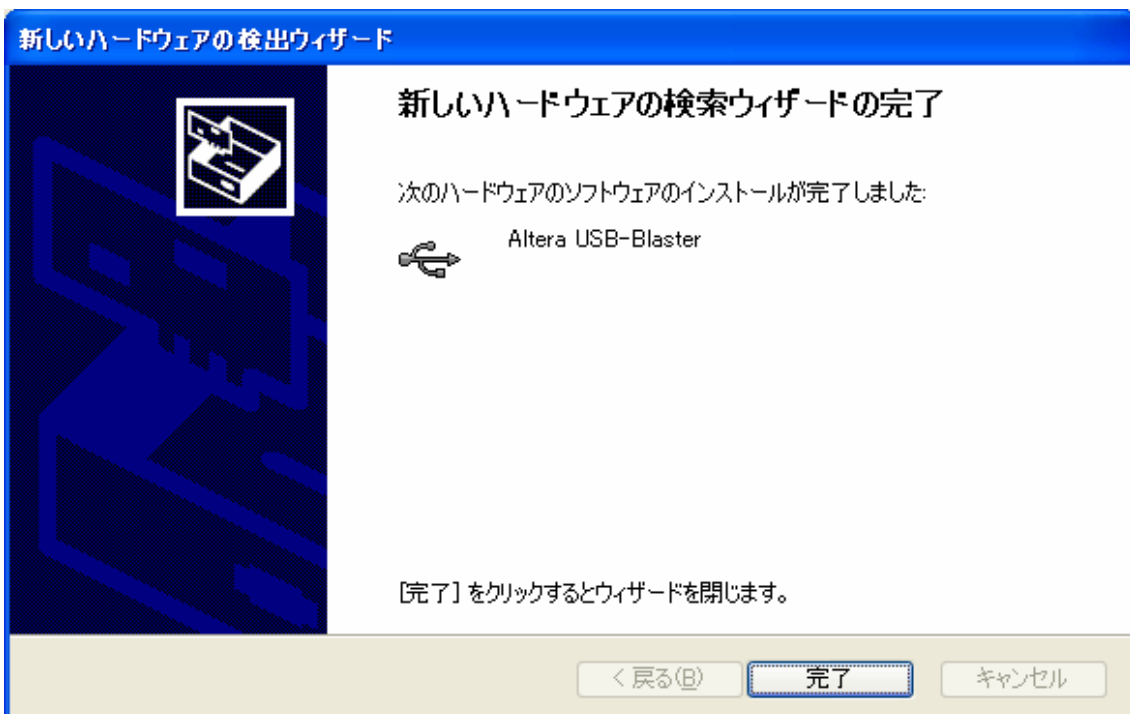
「一覧または特定の場所からインストール」を選択してください。



USB-Blaster のドライバは C:\altera\81\quartus\drivers\usb-blaster にあります。



インストール中。

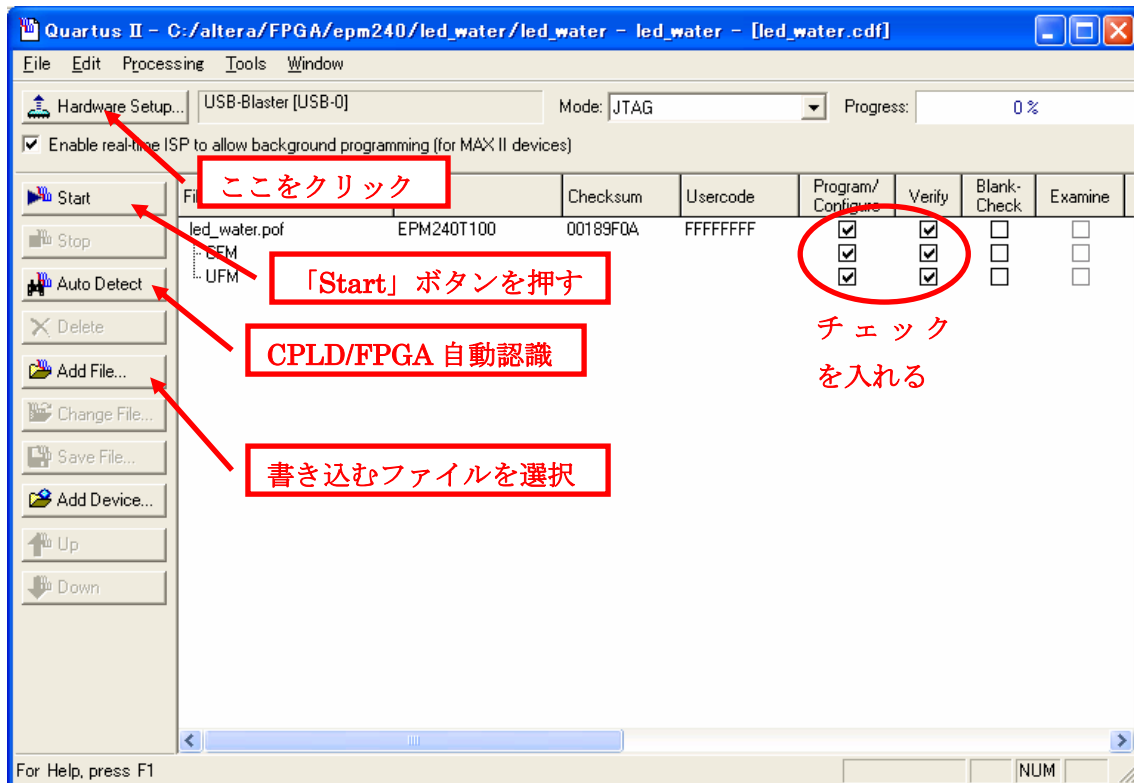
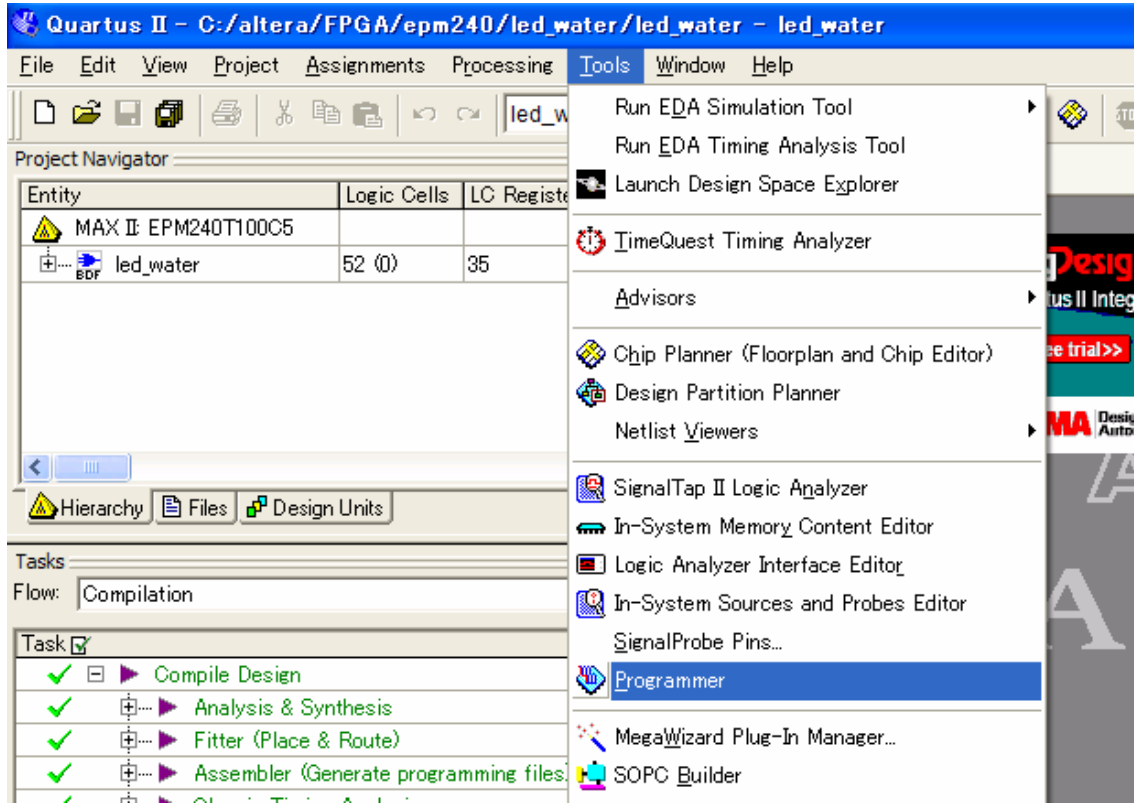


インストール完了します。

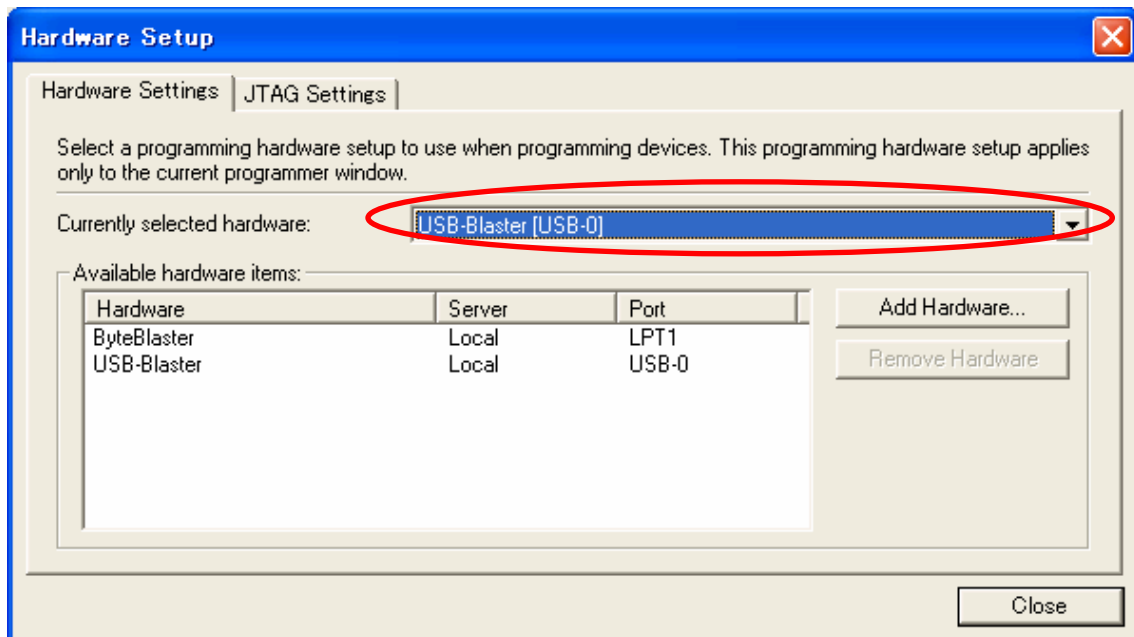
3.3 書き込むソフトウェアを起動する

Quartus II の「Tools」メニューから「Programmer」を選択すると、MAX II/Cyclone II

に回路を書き込むソフトウェア「Programmer ツール」が起動します。



Programmer ツールが起動したら、最初に書き込みケーブルのセットアップを行います。「Hardware Setup」というボタンを押してください。



「USB-Blaster[USB-0]」を選択します。「Close」を押して、Hardware Setup ダイアログを閉じたら、「Auto Detect」というボタンを押してください。これは、ケーブルの先にある CPLD/FPGA を自動認識する操作です。うまく CPLD/FPGA が認識されると、EPM240 又は EP2C5 又は EP2C8 という CPLD/FPGA が発見されるはずですが、発見されない場合は、

- ・ ケーブルが正しく接続されているか、
- ・ FPGA の場合は、ケーブルとボードの JTAG ポートを繋ぎますか
- ・ CPLD/FPGA 基板に電源が入っているか

など、これまでの作業に問題がないか再度チェックをしてください。

CPLD/FPGA の認識に成功すると、「Add File」ボタンを押して、書き込みファイルを添加します。*.pof は CPLD 用書き込みファイル、*.sof は FPGA 用書き込みファイルです。*.pof の右側にある Program/Configure と Verify の欄にチェックを入れて、「Start」ボタンを押します。プログレス・バーが 100%まで達すれば、書き込み成功です。

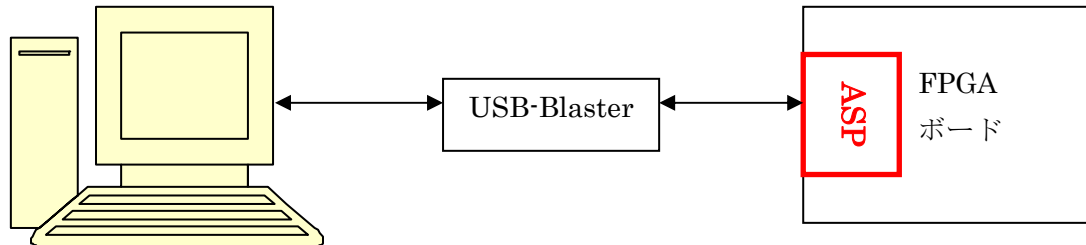
CPLD/FPGA 用 I/F 基板上的 LED が点滅しているのを確認してください。どうでしょうか？うまく点滅したでしょうか。

3.4 FPGAのコンフィギュレーションデバイスに書き込む

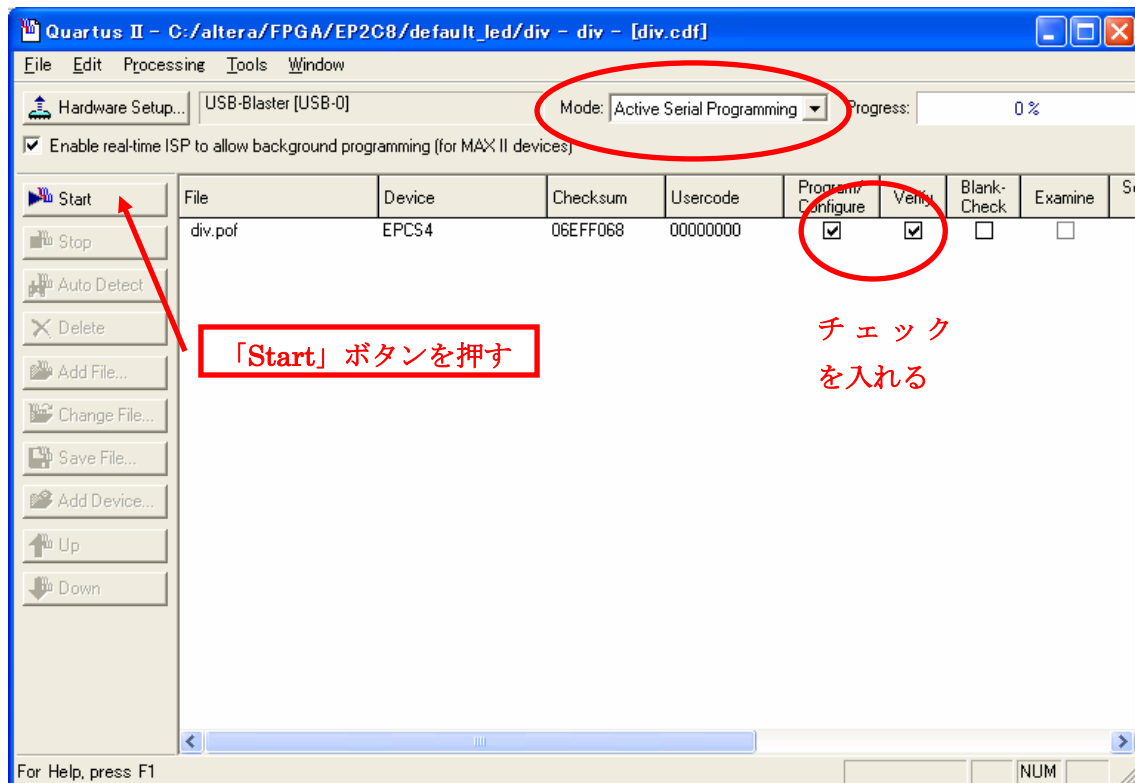
Cyclone II は SRAM ベースの FPGA なので、電源投入直後は中身が空の状態です。製品化

の際や、電源投入後に自動的に動作させる必要がある場合は、専用のコンフィギュレーションデバイス(EPCS4)に回路情報を書き込む必要があります。

専用のコンフィギュレーションデバイスに書き込む手順：



まず、USB-Blaster と FPGA ボードの **ASP** ポートを繋ぎます。
書き込むソフトウェア「Programmer ツール」が起動します。



「Mode」に[Active Serial Programming]を選択します。「Add File」ボタンを押して、書き込みファイル*.pofを添加します。*.pofの右側にある Program/Configure と Verify の欄にチェックを入れて、「Start」ボタンを押します。プログレス・バーが 100%まで達すれば、書き込み成功です。

書き込み成功した後、USB-Blaster を FPGA ボードの ASP ポートから抜いて、FPGA ボードに電源を再投入すると、どうの現象が出てきますか？

3.5 NIOS IIプロセッサの初体験

エクスプローラまたはマイ コンピュータを起動して、

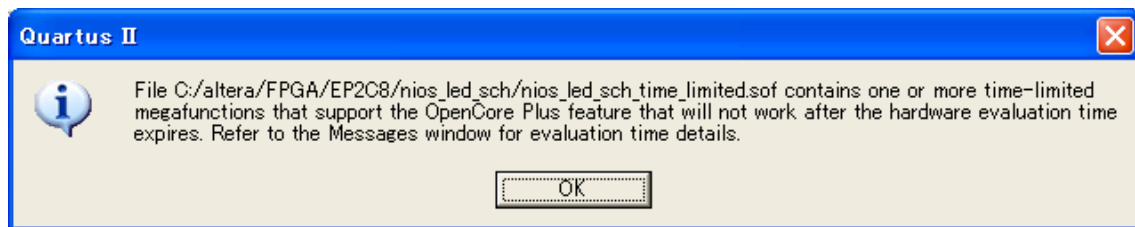
C:\¥altera¥EP2C8¥nios_led_sch

というフォルダを開いてください。

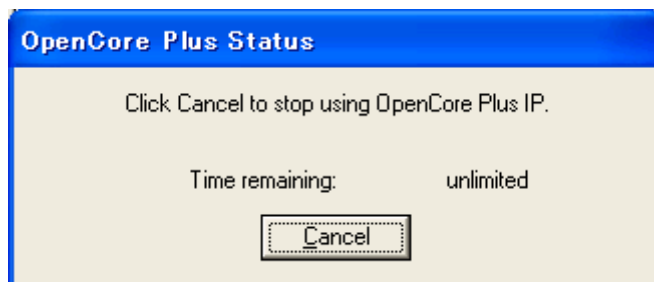
これらの中に、名前が nios_led_sch.qpf、Quartus II Project File となっているファイルがあります。これをダブル・クリックすると、Quartus II が起動して、nios_led_sch.qpf というプロジェクトが開きます。

他のプロジェクトと同じ手順でコンパイルして、Cyclone II ボードに書き込みます。

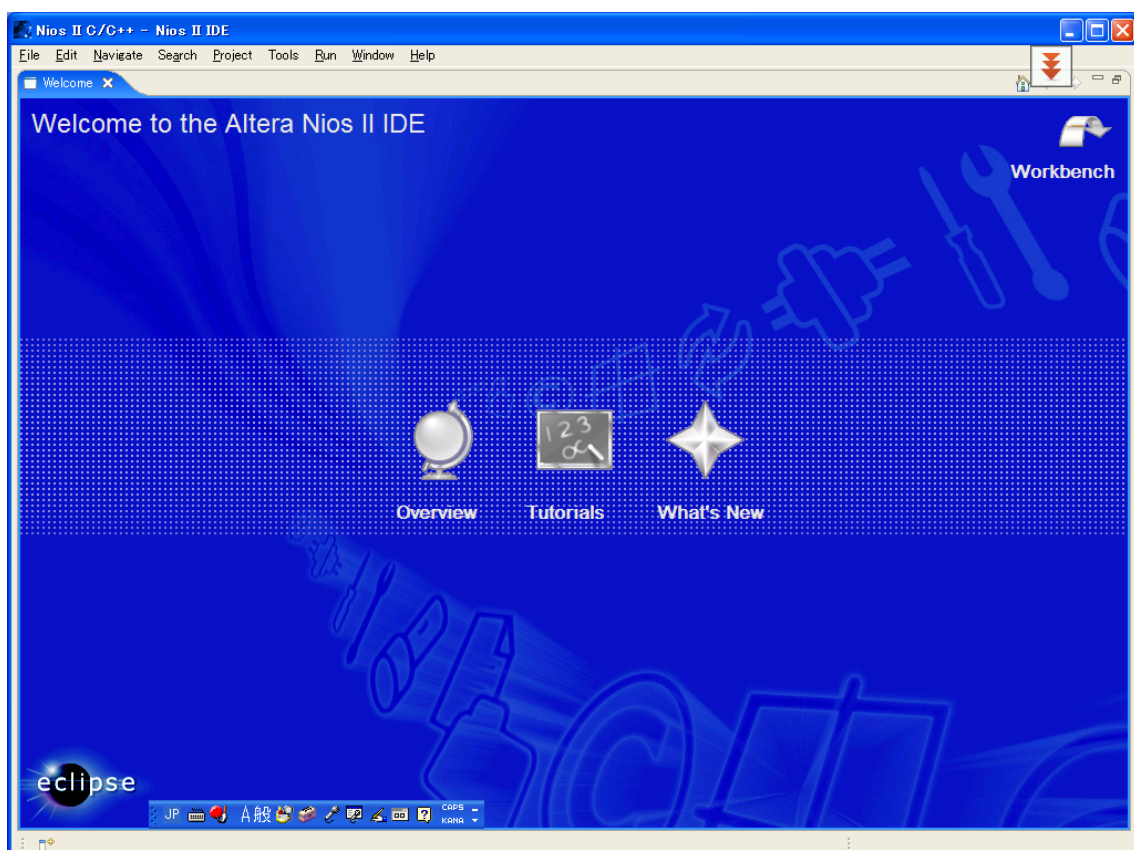
書き込み時、このような情報が出てきます。正式製品なら、アルテラ社からライセンスが必要です。評価の場合は、そのまま「OK」ボタンを押します。



書き込み完了したら、その画面が出てきます。「Cancel」ボタンを押さないでください。その画面をそのまま置いてください。

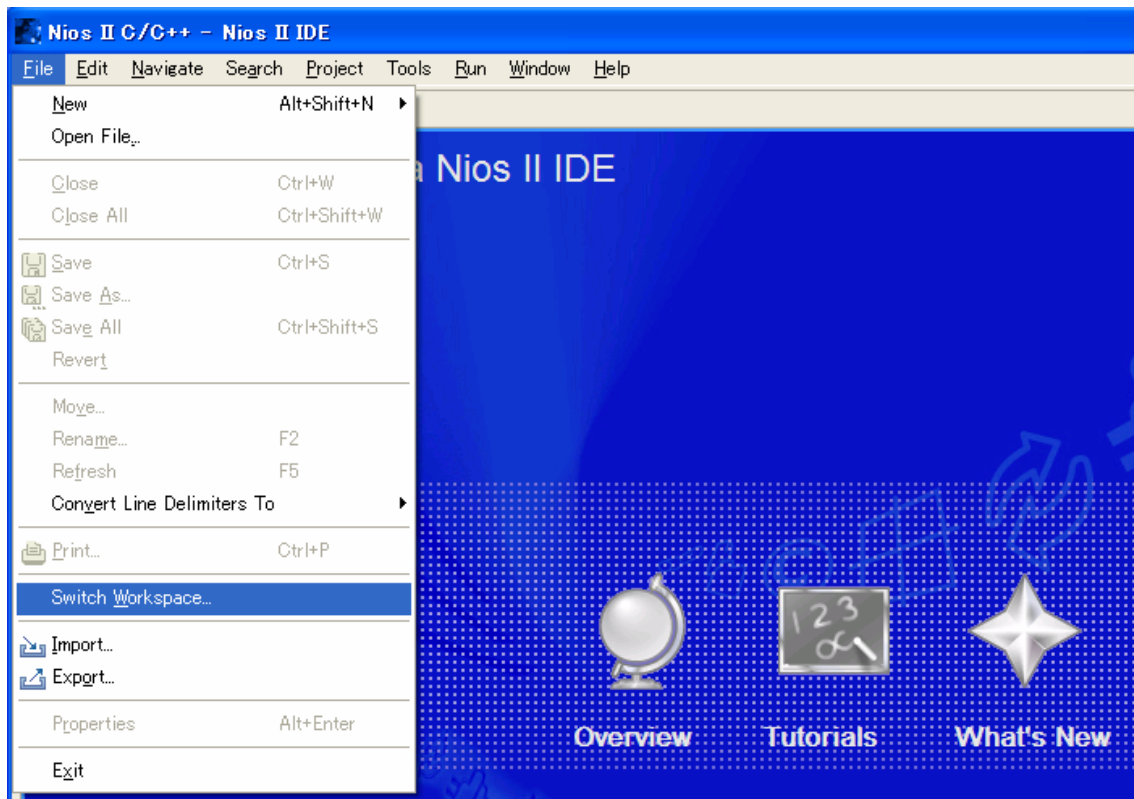


Windows の「スタート」→「すべてのプログラム」→「Altera」→「NIOS II EDS 8.1」から NIOS II 8.1 IDE が起動します。

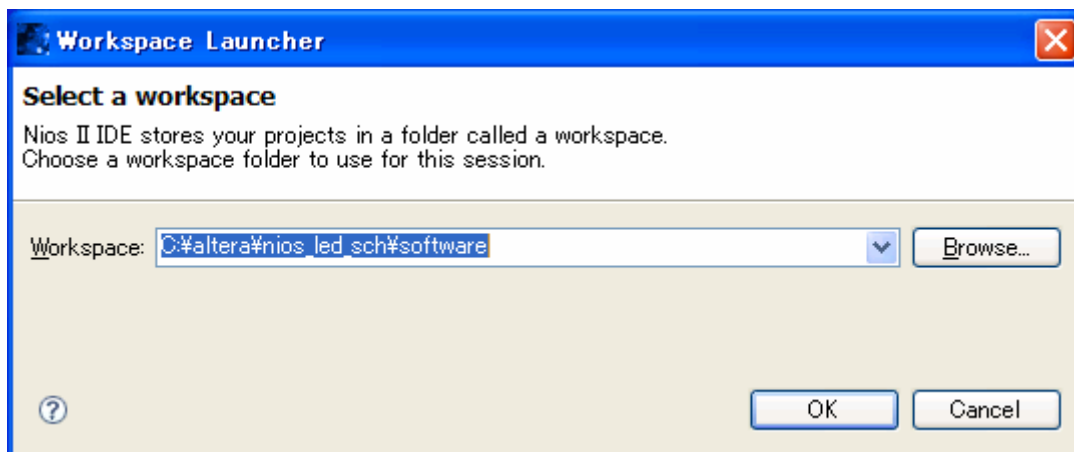


NIOS II IDE の初起動の画面です。

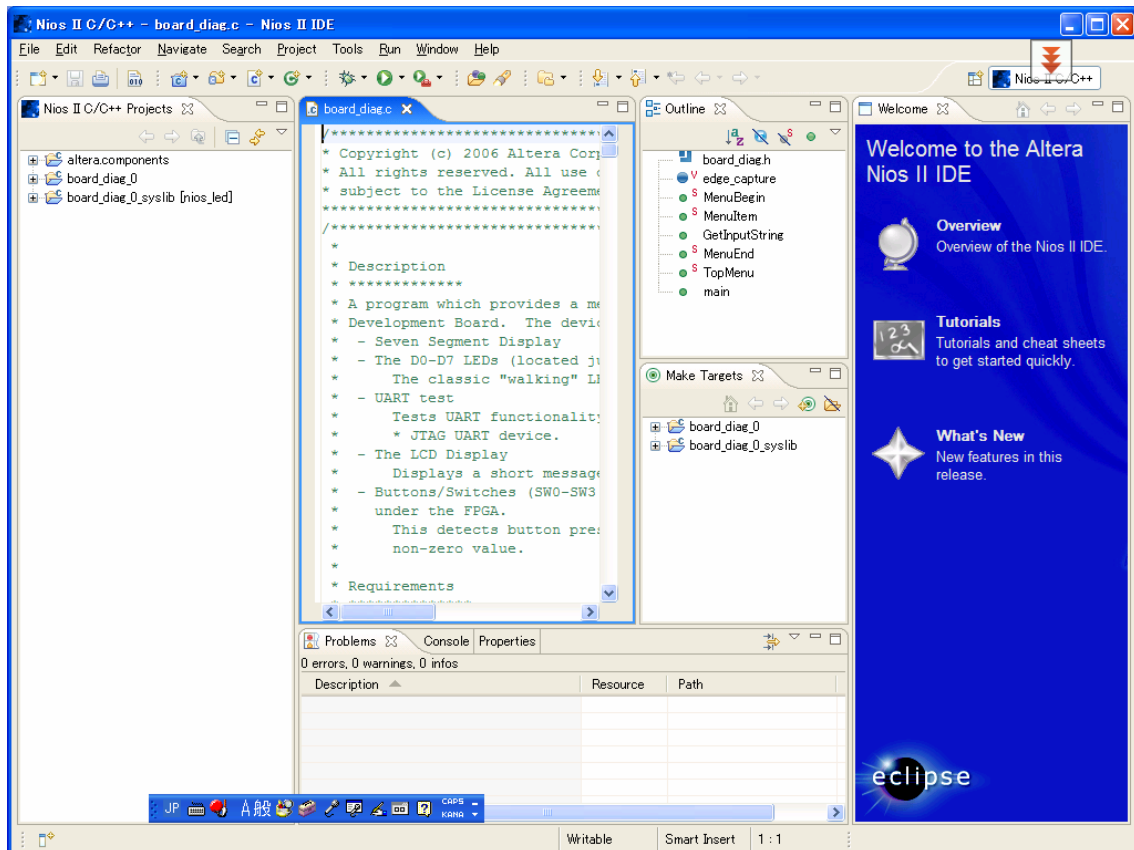
NIOS II IDE のメニュー「File」→「Switch Workspace」を選択します。



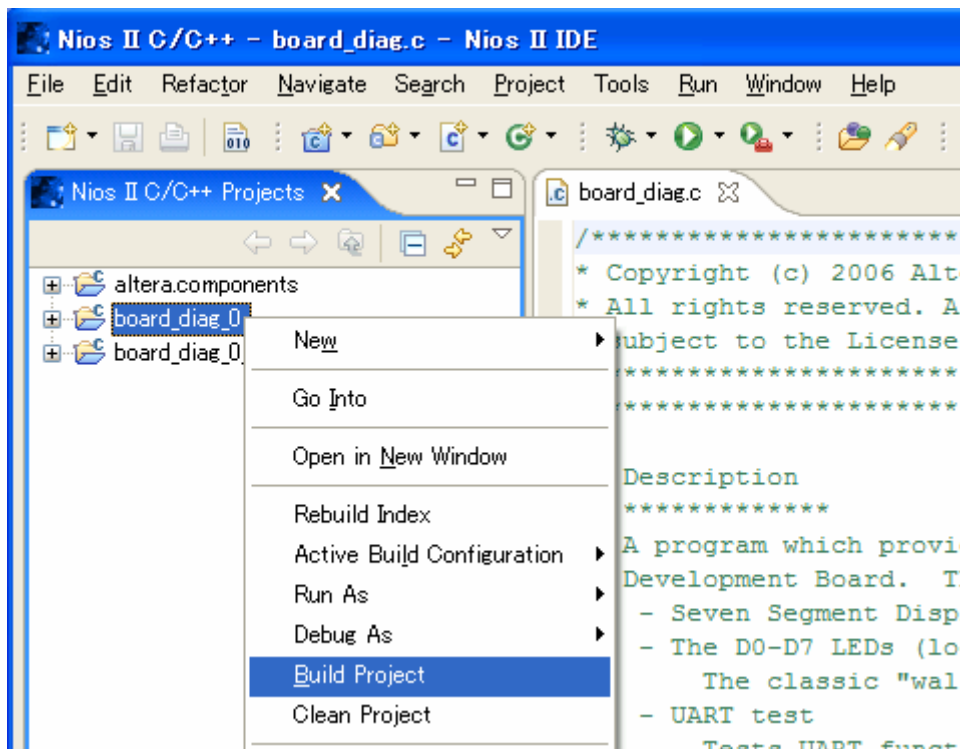
C:\altera\EP2C8\nios_led_sch\software を入力して、「OK」ボタンを押します。

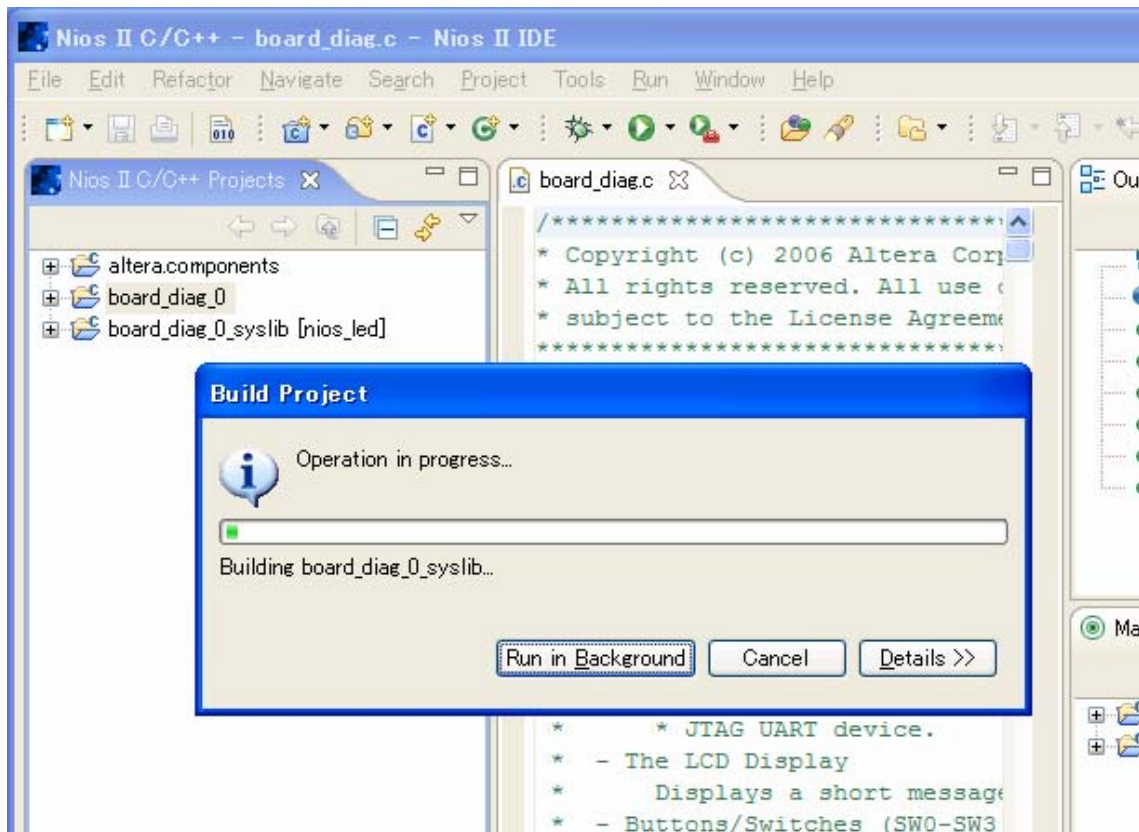


NIOS II IDE が再起動します。

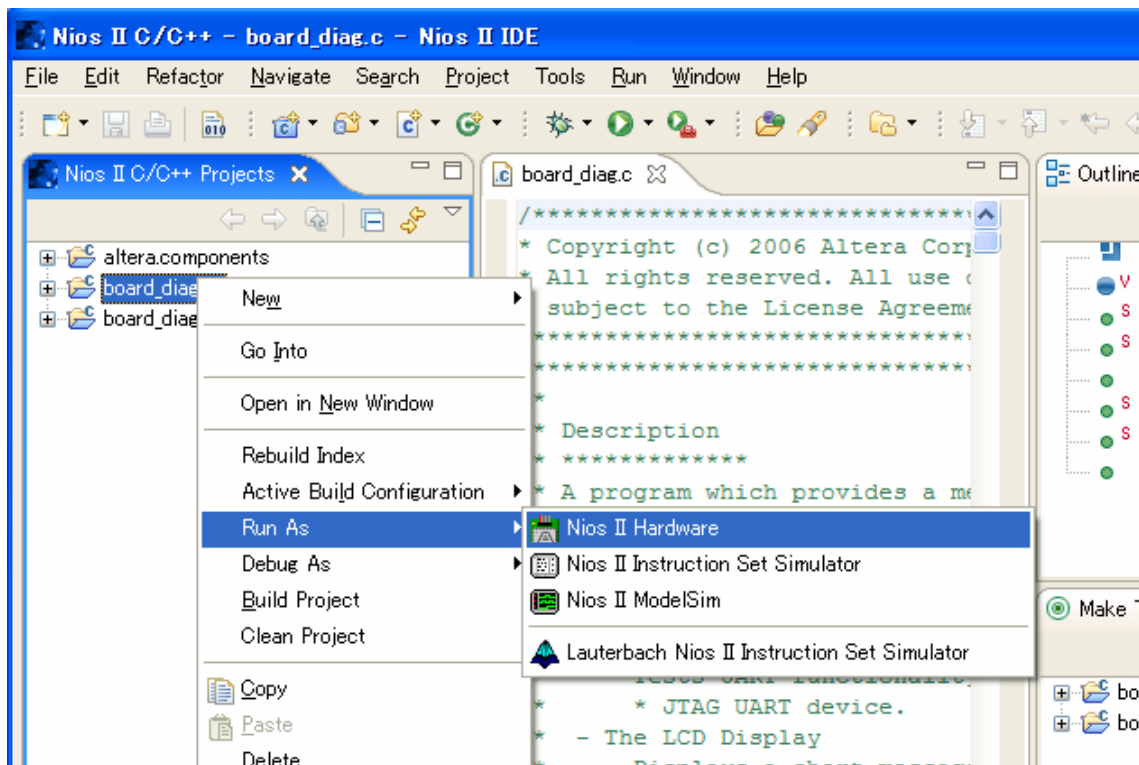


左側の「board_diag_0」でマウスの右ボタンをクリックして、「Build Project」を選択して、ビルドを開始します。





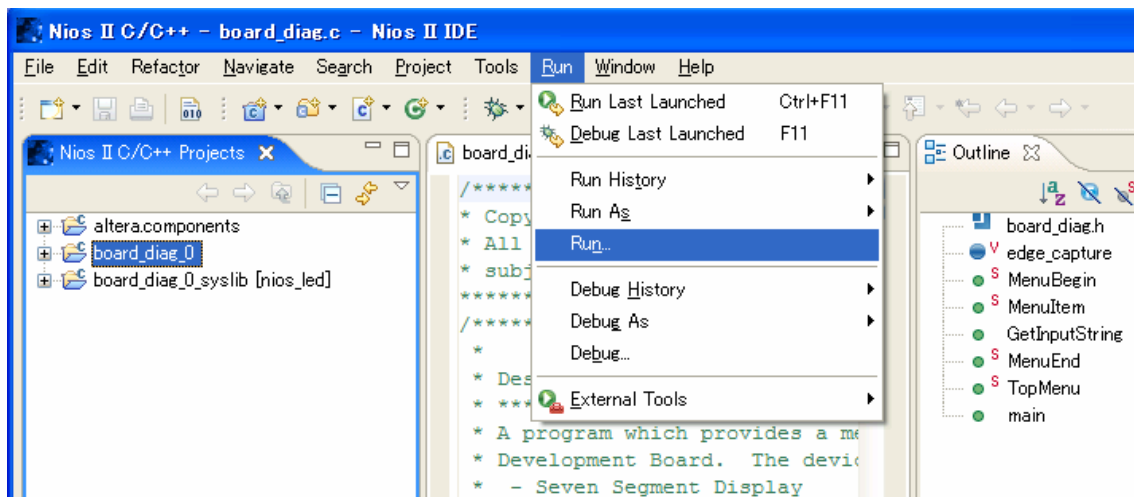
ビルド完了すると、左側の「board_diag_0」でマウスの右ボタンをクリックして、「Run As」→「Nios II Hardware」を選択して、Cyclone ボードにプログラムをダウンロードします。



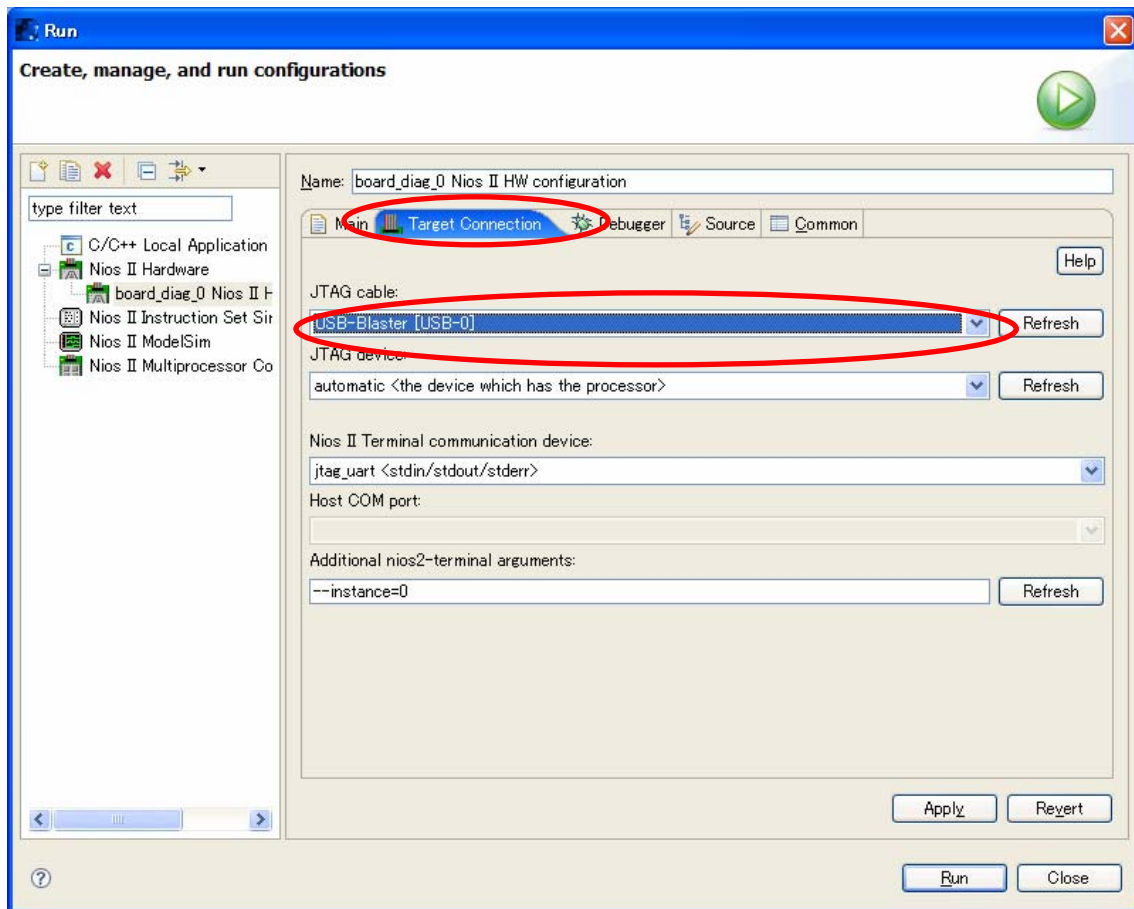
若しこの情報が出てきたら、

You have more than one JTAG cable available so you must use the --cable option to choose between them (or open the "Run/Run" or "Run/Debug" dialog and go to the "Target Connection" tab).
No --cable option was provided (or you selected Automatic)

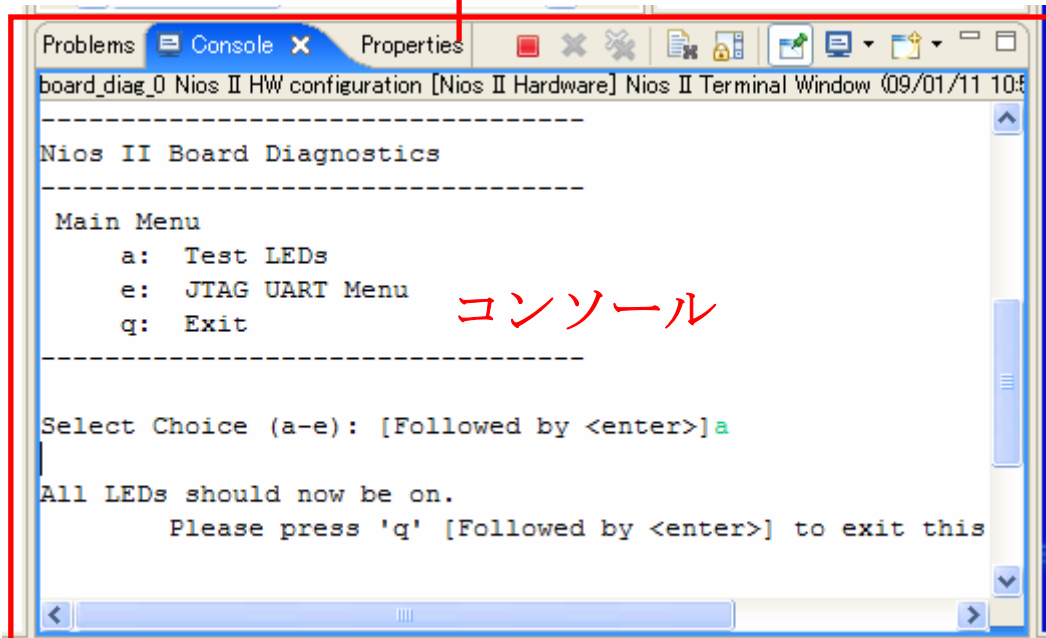
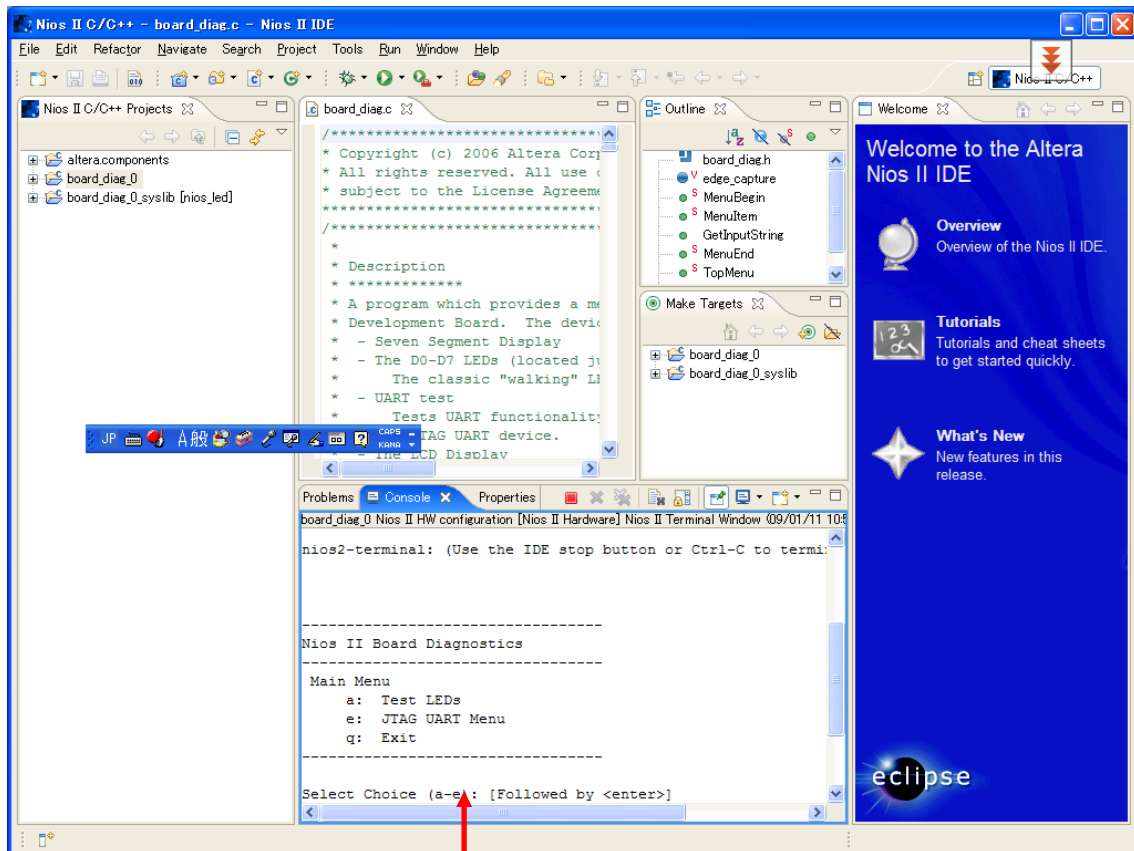
Nios II IDE のメニュー「Run」→「Run」を選択します。



「Target Connection」を押して、JTAG cable で「USB-Blaster [USB-0]」を選択します。
「Run」ボタンを押します。



Cyclone II ボードにプログラムをダウンロード完了すると、自動的にプログラムを実行します。Cyclone II ボードとダイアログするコンソールが出てきます。



コンソールでメニューを選択して、NIO S II システムとダイアログできます。例えば、「a」と「enter」キーを入力すると、Cyclone II ボードのユーザ LED を点灯します。

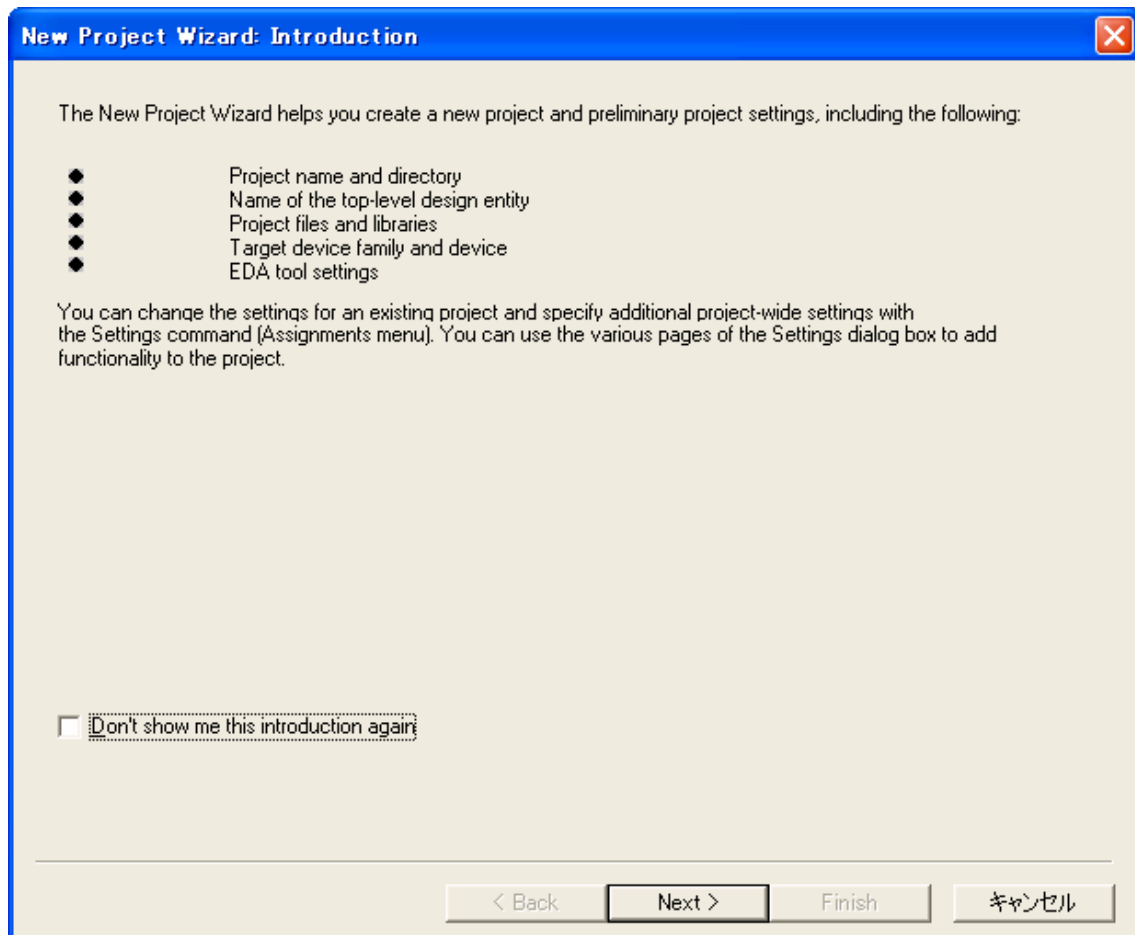
第四章 CPLD/FPGAの開発入門

4.1 プロジェクトを作成する

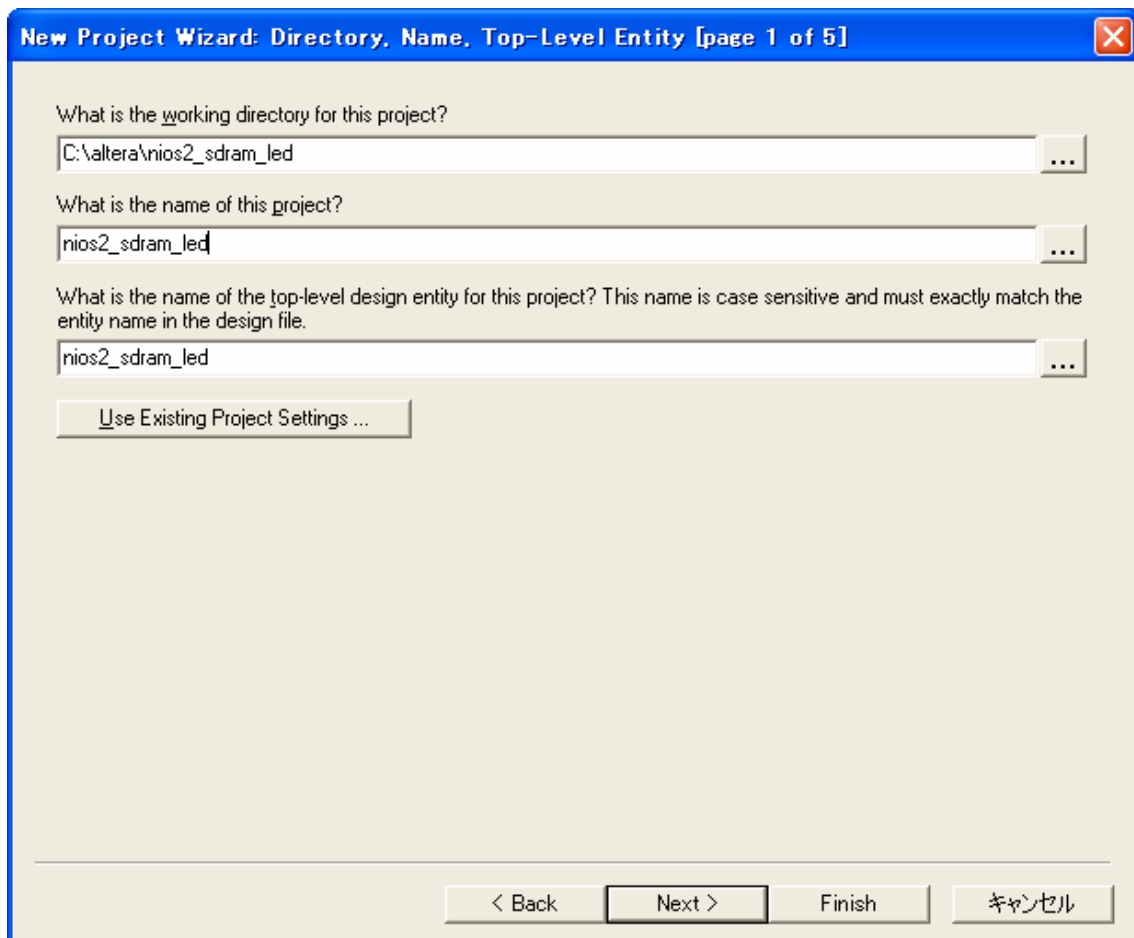
Windows を起動し、「スタート」メニューから Quartus II Web Edition を起動します。

Quartus II 評価版上では、これから作る回路が一つのプロジェクトとして扱われます。まずは、新しいプロジェクトを作成しましょう。

新規にプロジェクトを作成するには、Quartus II 評価版の「File」メニューから「New Project Wizard」を選択し、プロジェクト作成ウィザードを起動します。このウィザードを使えば、ダイアログに表示された質問に答えていくだけで、簡単にプロジェクトを作ることができます。

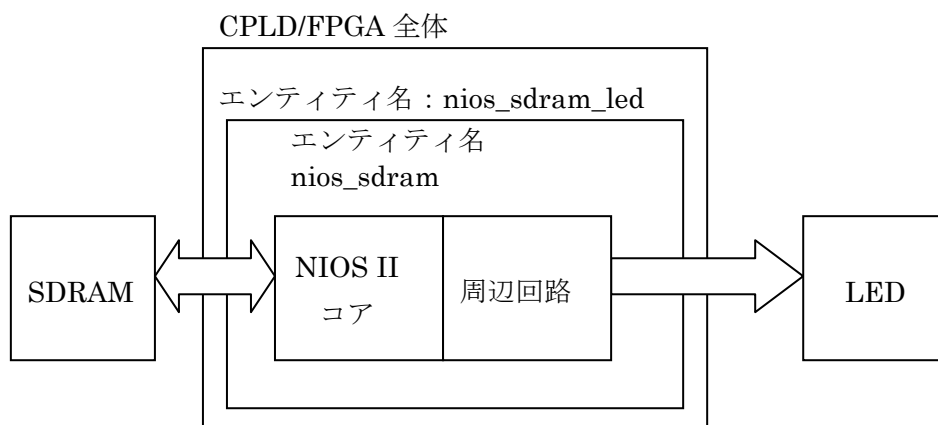


最初に、New Project Wizard に関する説明が表示されるので、そのまま「Next」を押します。すると、プロジェクトの名前や保存場所を聞いてきます。各問いに対して、次のように書き入れて「OK」ボタンを押してください。



パス名やプロジェクト名、モジュール名に漢字や空白などの特殊な記号が含まれていると、ツールによってはうまく動かないものがあるので、半角のアルファベットや数字などを組み合わせただけの単純な名前にしてください。

プロジェクト作成ウィザードで入力した三つめの項目は、回路を階層的に設計する場合に最上位の階層に置くエンティティの名前です。エンティティとは、あるまとまった機能をもった回路のことです。

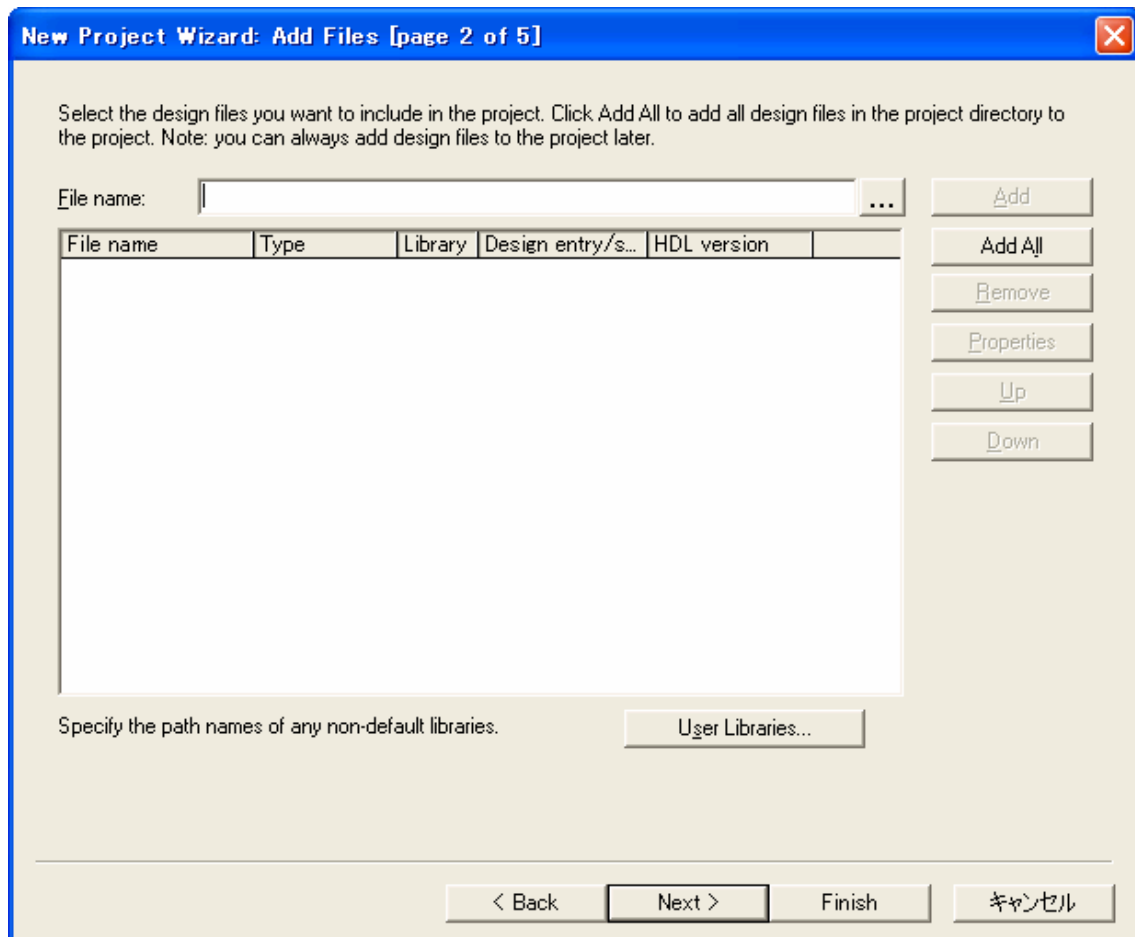


これから作成する回路のエンティティの階層構造です。FPGA 内部は、nios_sdram_led というエンティティで作られています。そして、nios_sdram_led というエンティティの中に、nios_sdram というエンティティがあります。さらに、nios_sdram というエンティティの中に NIOS II コアと周辺回路など幾つのエンティティがあります。

※ MAXII は容量が不足ですので、ソフトプロセッサ NIOS II を搭載できません。

このように、小さなエンティティを組み合わせると大きな回路を作って階層構造にすると、効率良く開発できます。

ダイアログの三つの欄に入力したら「Next」を押します。すると、



この画面が現れます。プロジェクトに追加したいファイルがある場合はここで追加できます。今回は不要なので、なにも選択せずに「Next」を押します。

New Project Wizard: Family & Device Settings [page 3 of 5]

Select the family and device you want to target for compilation.

Device family
Family: Cyclone II
Devices: All

Target device
 Auto device selected by the Fitter
 Specific device selected in 'Available devices' list

Show in 'Available device' list
Package: Any QFP
Pin count: 208
Speed grade: 8
 Show advanced devices
 HardCopy compatible only

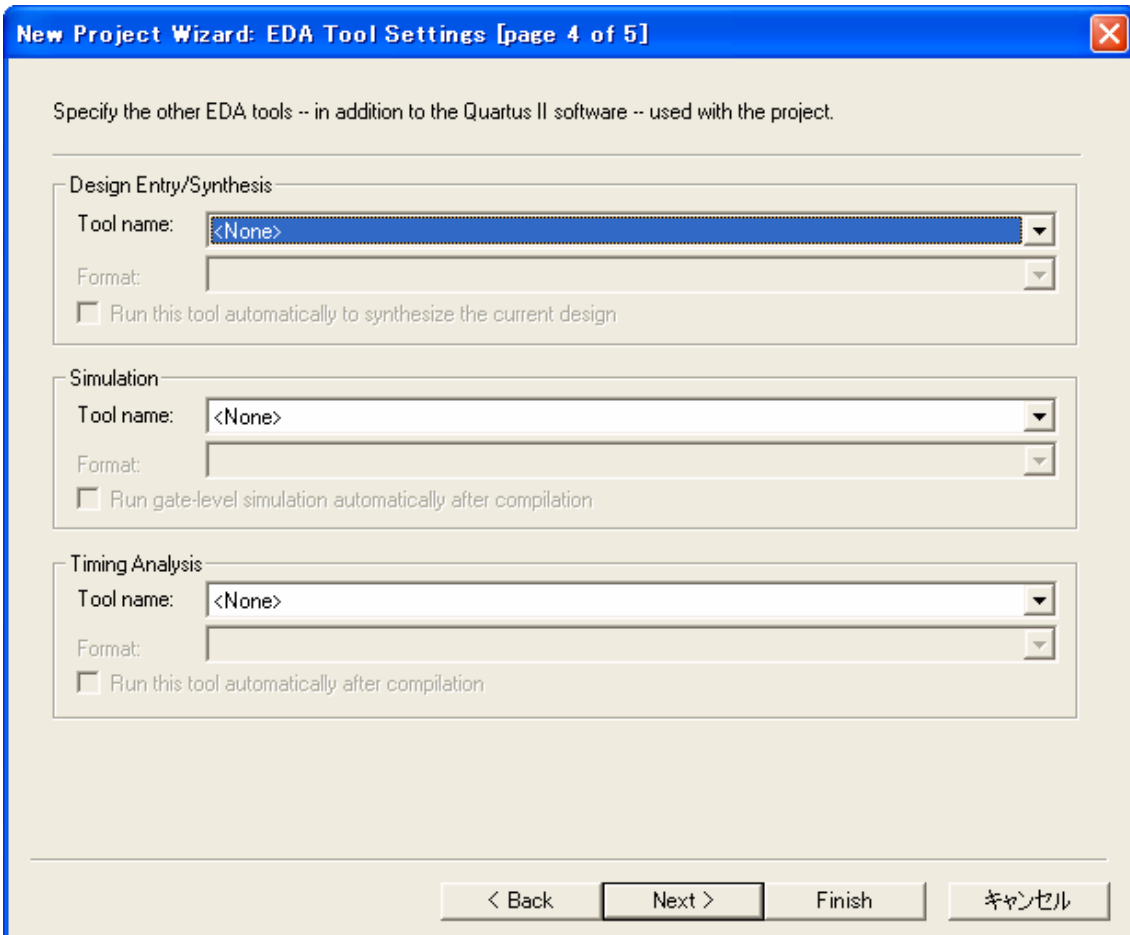
Available devices:

| Name | Core v... | LEs | User I/... | Memor... | Embed... | PLL | Global ... |
|-------------|-----------|------|------------|----------|----------|-----|------------|
| EP2C5Q208C8 | 1.2V | 4608 | 142 | 119808 | 26 | 2 | 8 |
| EP2C5Q208I8 | 1.2V | 4608 | 142 | 119808 | 26 | 2 | 8 |
| EP2C8Q208C8 | 1.2V | 8256 | 138 | 165888 | 36 | 2 | 8 |
| EP2C8Q208I8 | 1.2V | 8256 | 138 | 165888 | 36 | 2 | 8 |

Companion device
HardCopy:
 Limit DSP & RAM to HardCopy device resources

< Back Next > Finish キャンセル

使う FPGA デバイスの選択ダイアログで、まず「Family」というダウン・メニューで「Cyclone II」を選択し、「Available devices」の中から「EP2C8Q208C8」を選びます。選んだら、「Next」を押します。



Specify the other EDA tools -- in addition to the Quartus II software -- used with the project.

Design Entry/Synthesis

Tool name:

Format:

Run this tool automatically to synthesize the current design

Simulation

Tool name:

Format:

Run gate-level simulation automatically after compilation

Timing Analysis

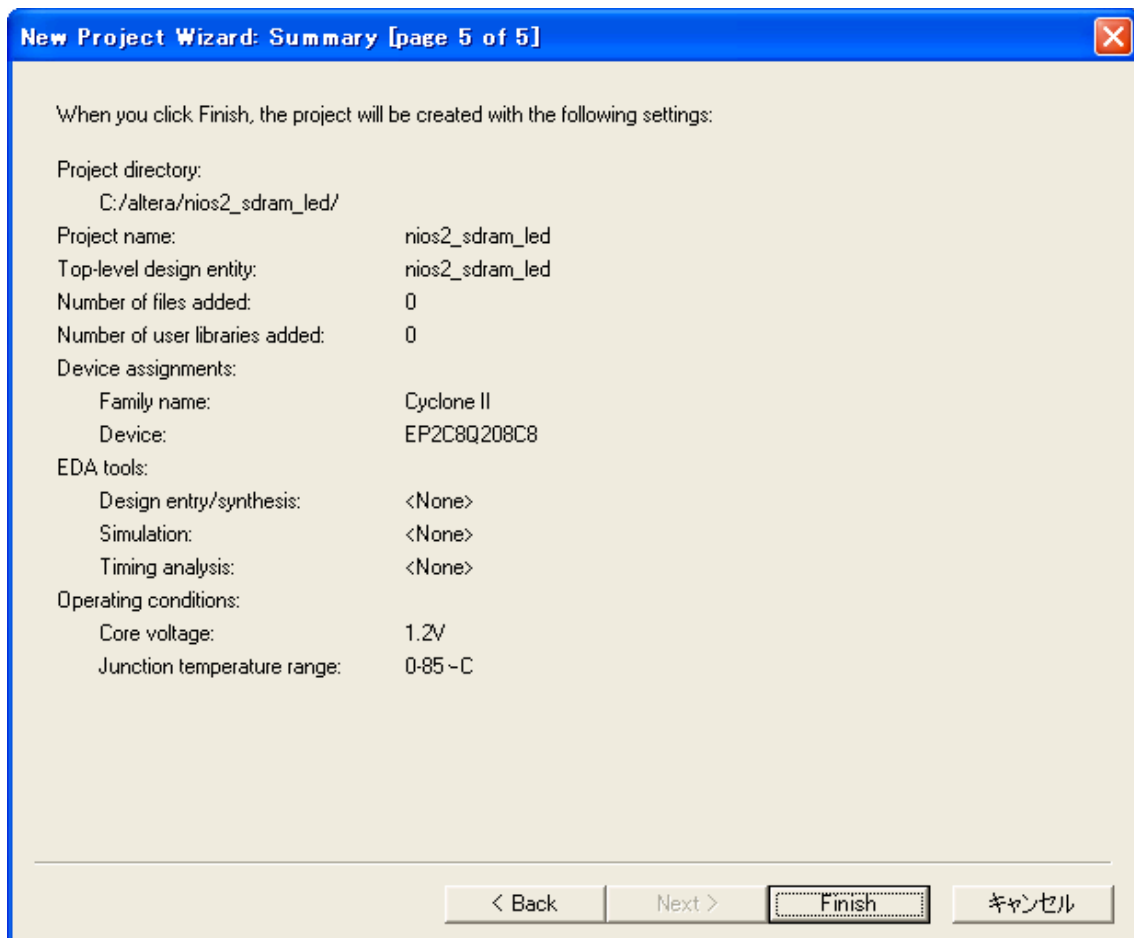
Tool name:

Format:

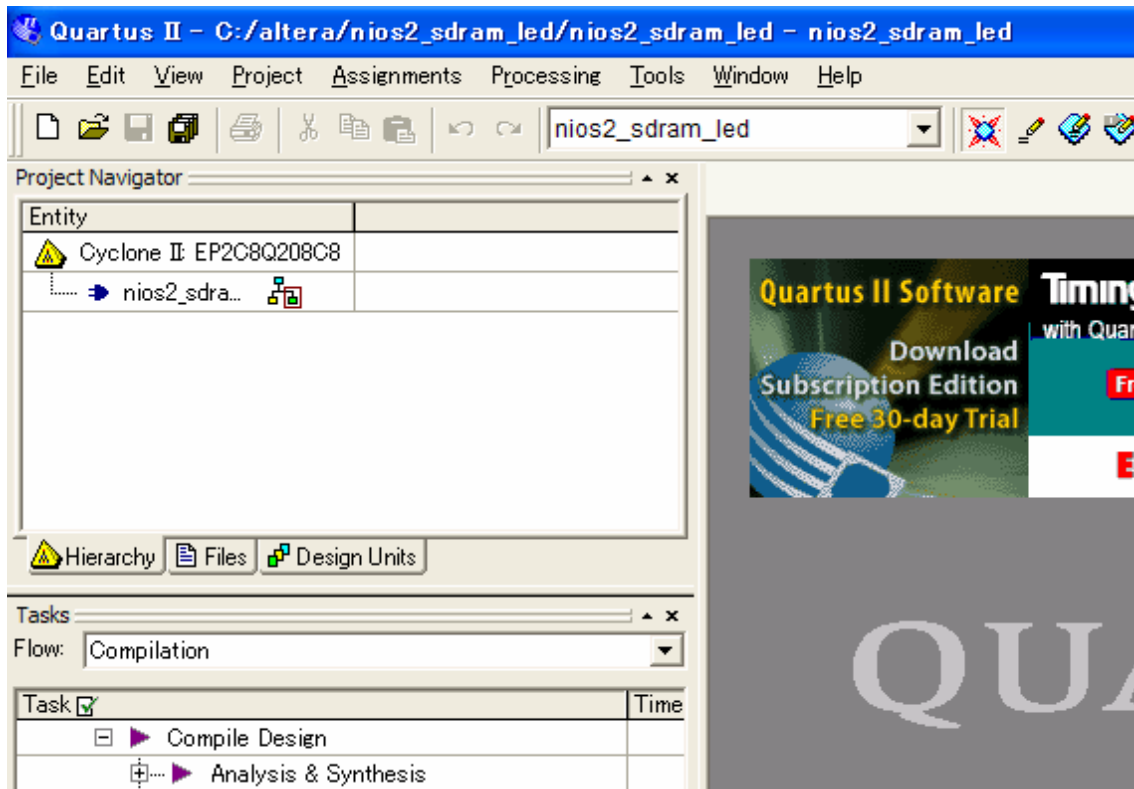
Run this tool automatically after compilation

< Back Next > Finish キャンセル

この画面では、このプロジェクトで使用したい Quartus II 評価版以外の外部ツールを選択できます。今回は Quartus II 評価版だけで最後まで設計を行うので、何も選択せずに「Next」を押します。



内容を確認して「Finish」を押します。



これは nios_sdram_led プロジェクト作成直後の画面です。

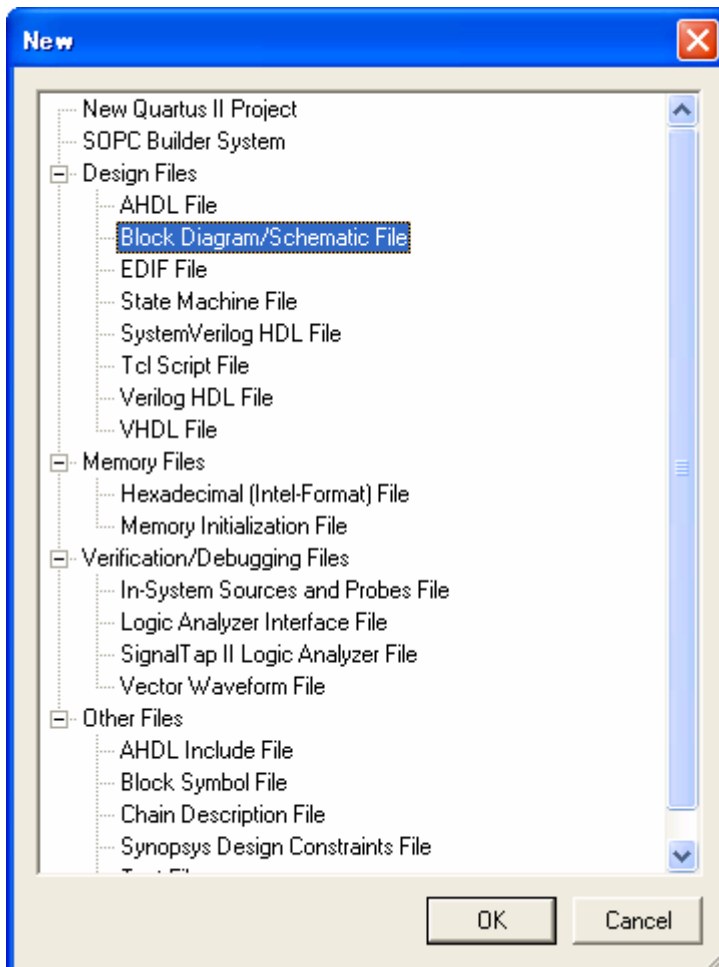
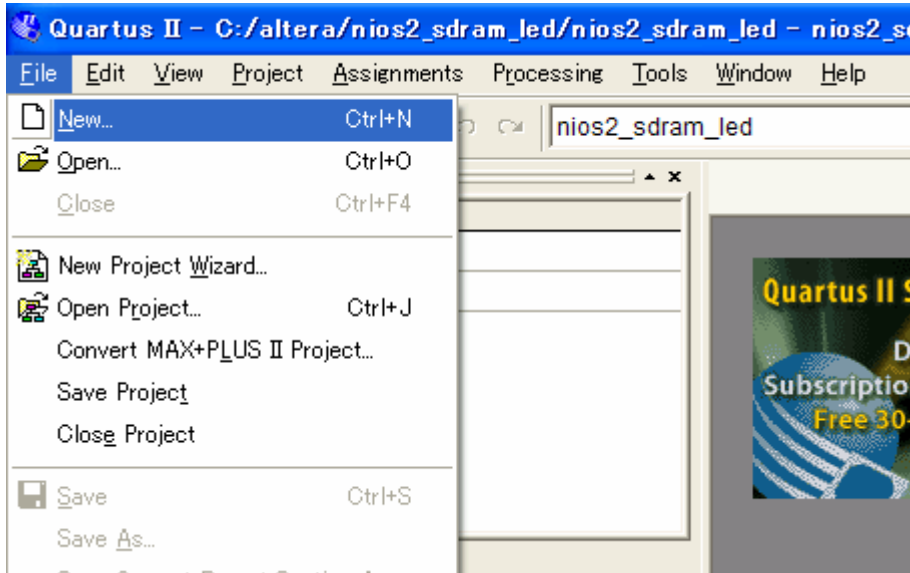
左上の Project Navigator の中に、デバイス名とトップ・レベル・エンティティ名 (nios_sdram_led) が入れ子になって表示されています。これは「EP2C8Q208C8 というデバイスの中に nios_sdram_led というエンティティが入っていますよ」という意味です。

このように、Project Navigator 欄を見れば、今開いているプロジェクトがどのような構造をしているかを確認できます。Project Navigator 欄を閉じてしまった場合は、「View」メニューの中の「Utility Windows」という項目の中に「Project Navigator」があるので、こちらを選択すればもう一度表示できます。

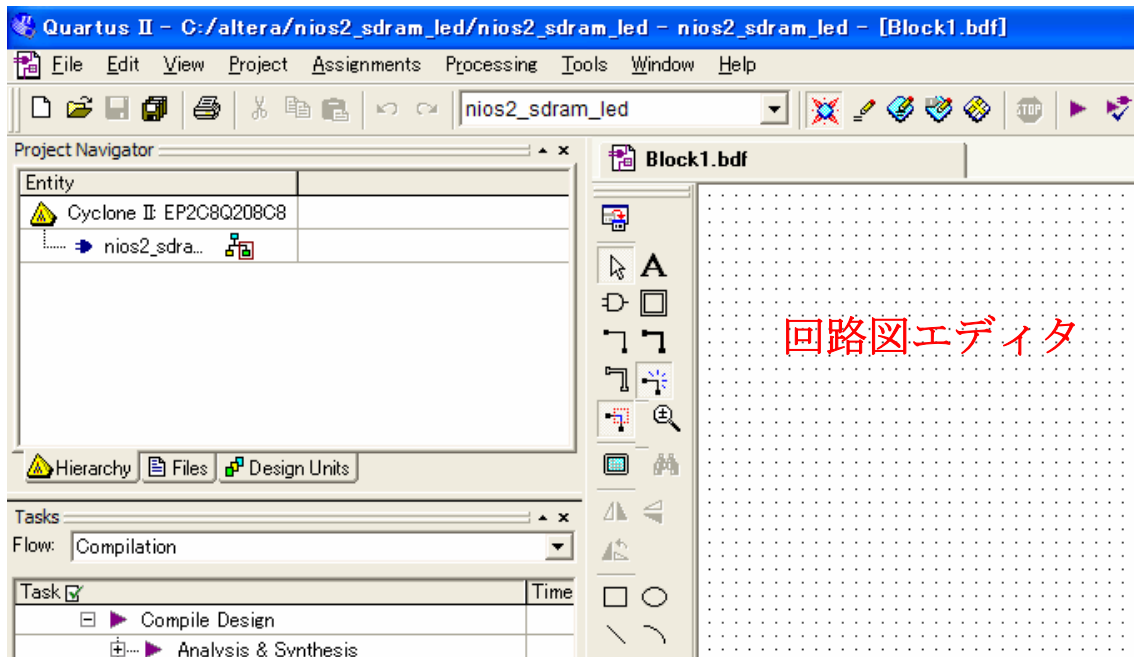
4.2 エディタで回路図を描く

4.2.1 トップ・エンティティを作成する

「File」メニューから「New」を選択します。

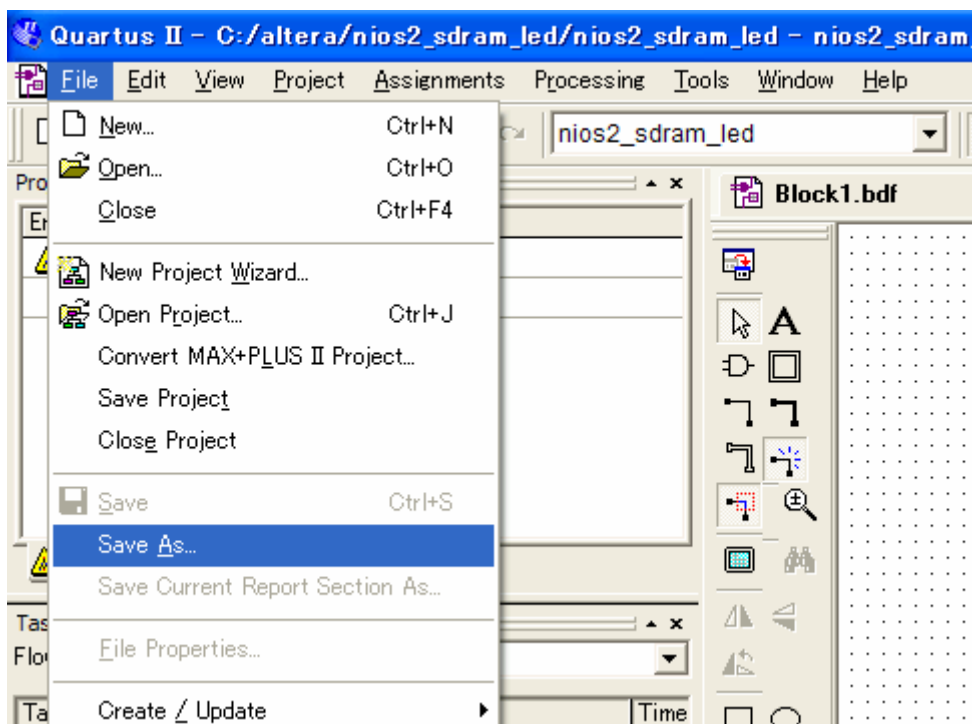


「Block Diagram/Schematic File」を選択します。「OK」を押します。



これが回路図エディタです。ここに回路記号を配置していけば、CPLD/FPGA 内部の回路を設計できます。

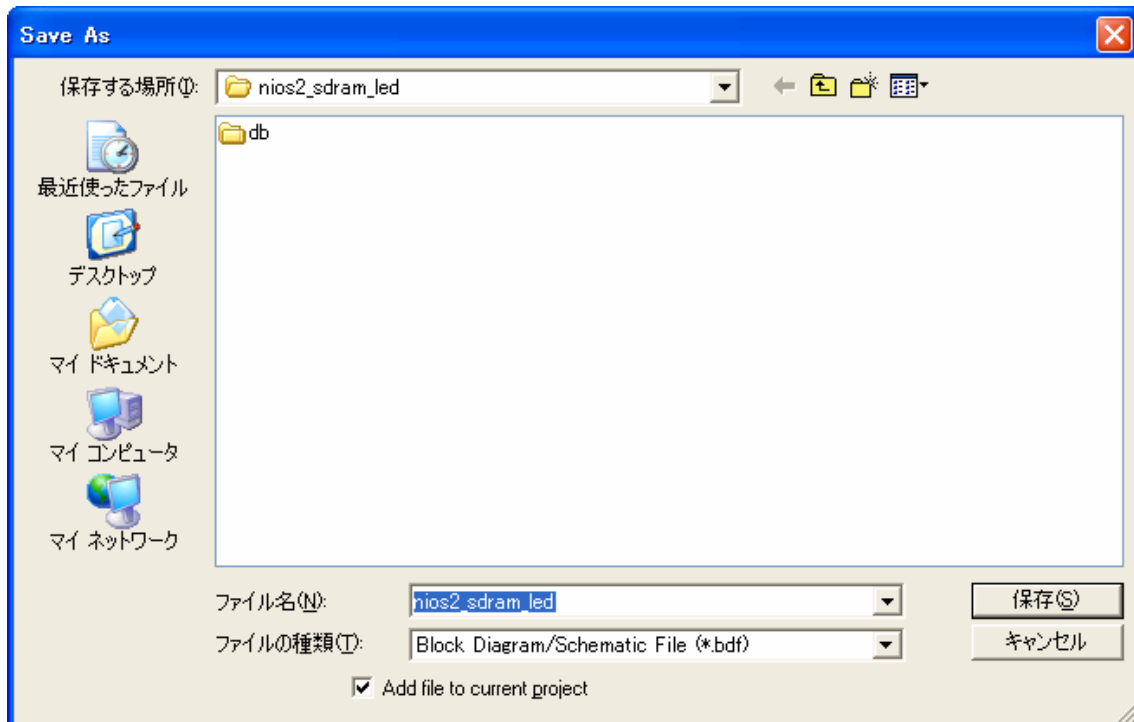
Block1.bdf という名前は、Quartus II 評価版が勝手に付けた名前です。これから作りたのは nios_sDRAM_led というエンティティです。回路図とエンティティを対応させるために、ファイル名を nios_sDRAM_led.bdf として保存します。



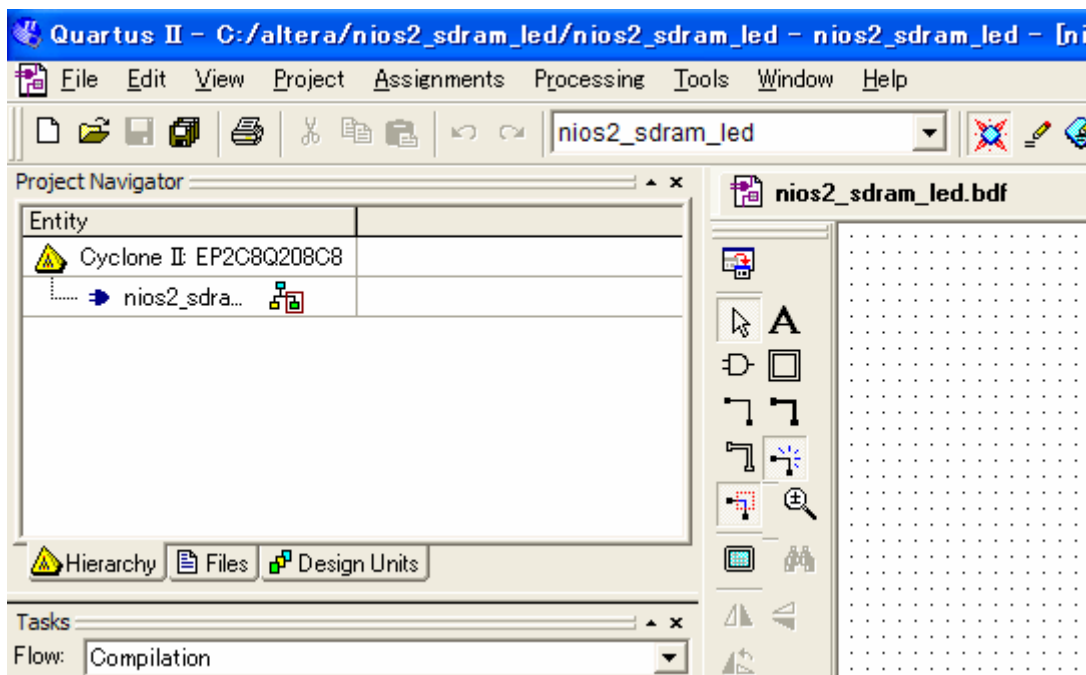
「File」メニューから「Save As...」を選び、ファイル名を nios_sdram_led としてください。保存する場所は先ほど指定したプロジェクトのワーキング・ディレクトリで、

C:\¥altera¥nios_sdram_led

です。「Save As...」を選ぶと自動的に、このフォルダが開かれます。



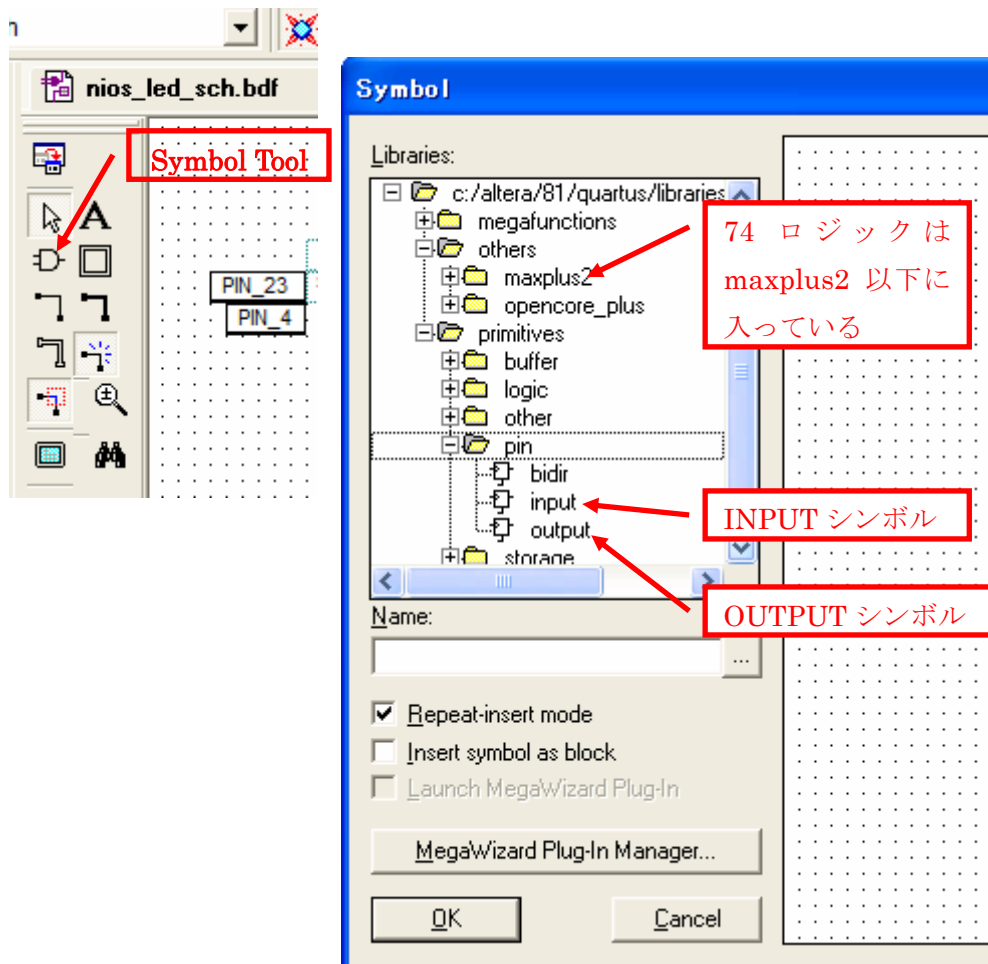
保存が終わると、ウィンドウの名前が nios_sdram_led.bdf に変わります。



4.2.2 作画手順

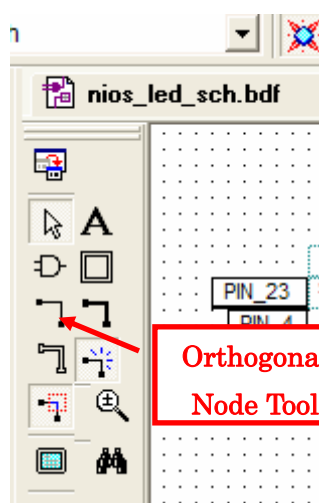
①シンボルを置く

「Symbol Tool」というボタンを押すと、このようなダイアログが開きます。



Libraries 欄に、[+]記号をクリックすると、中身が表示されます。中身のシンボルを選択して「OK」ボタンを押します。マウス・カーソルに選択されたシンボルがくっついた状態になるので、シンボルを配置したい場所へ移動して、マウスの左クリックを押します。配置できた時点で ESC キーを押してカーソルを元に戻してください。

②シンボル同士を接続する



「Symbol Tool」の少し下にある「Orthogonal Node Tool」という鍵状のアイコンを選択します。マウス・カーソルをシンボルのピンに併せて左ボタンを押し、そのまま離さずに他のピンまでドラッグします。すると、ドラッグの開始点と終了点が線で結ばれます。

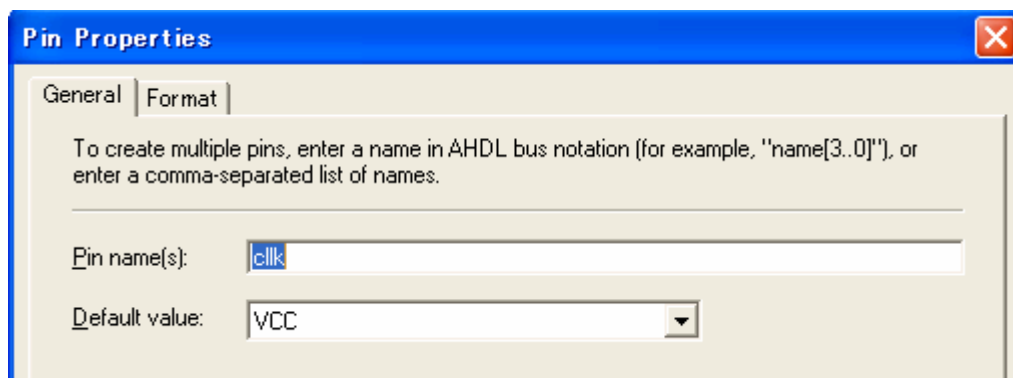
同様にしてほかの線も配置してください。接続が終わったら ESC キーを押し、元のカーソルに戻します。接続を間違えた場合は、一度 ESC キーを押して元のカーソルに戻し、不要な配線をクリックして Delete してください。

③INPUT 端子と OUTPUT 端子の名前を変える

INPUT シンボルと OUTPUT シンボルは、エンティティ外側と接続するための端子です。端子の名前は pin_name という素っ気ないものになっています。

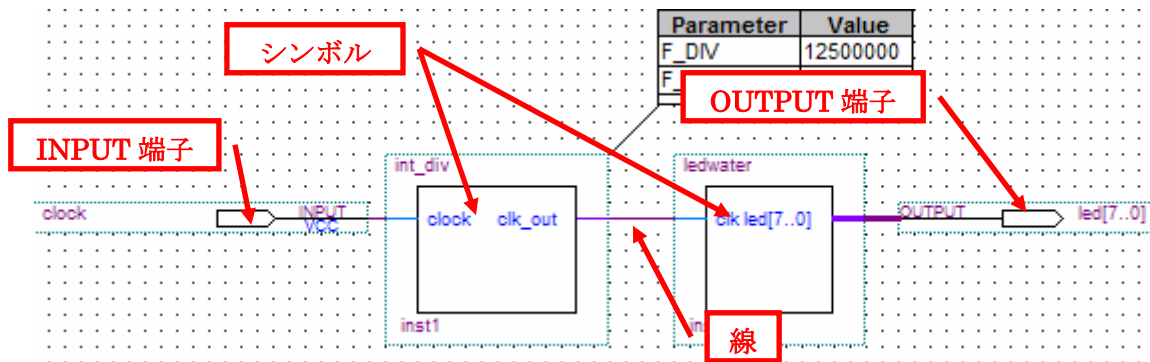
これでは、エンティティを使うときに、いったい何のための端子なのかわからなくなります。そこで、わかりやすい名前に変更しておきます。

端子の名前を変更するためには、ESC キーを押して通常のカーソルに戻した後に、回路図上の INPUT/OUTPUT シンボルをダブル・クリックします。



Pin name 欄にわかりやすい名前を入力します。

エンティティの回路図入力作業は終わったら、「File」メニューから「Save」を実行して回路を保存してください。



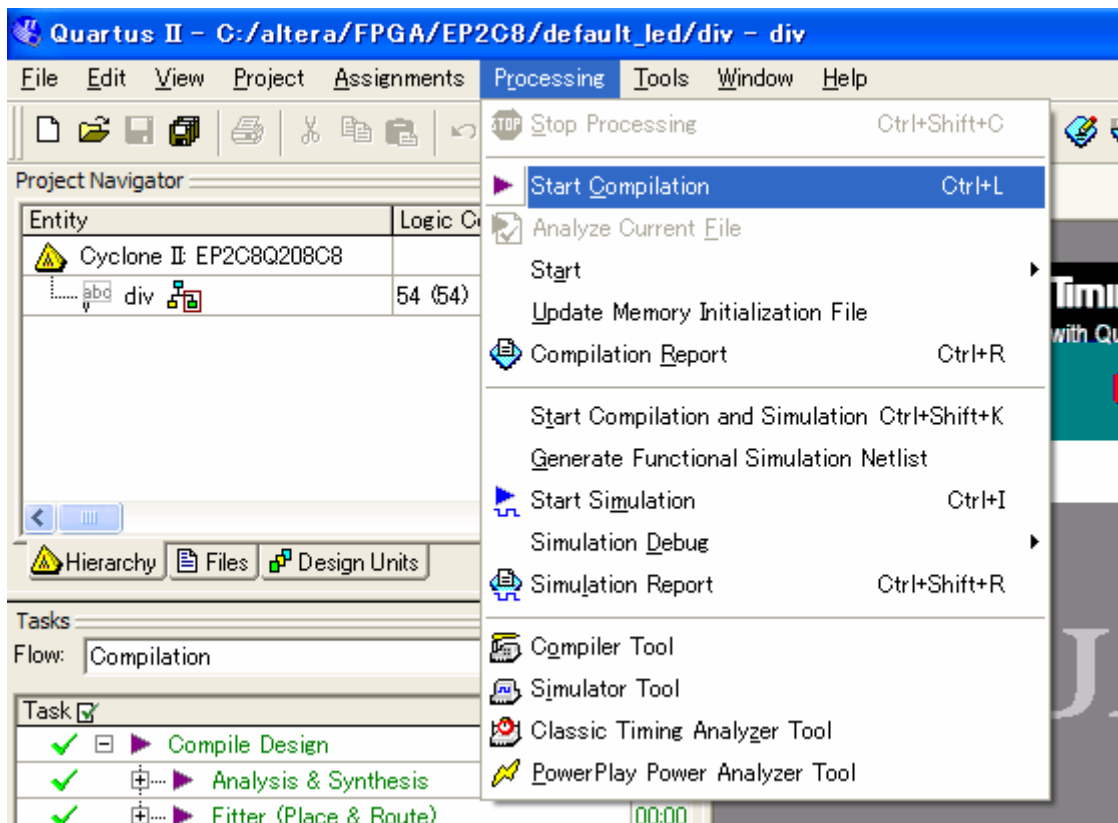
ある回路図の様子。

4.2 書き込み前の二つの作業

4.2.1 回路図をコンパイルする

入力した回路図から CPLD/FPGA に書き込むデータを生成するためには、作成した回路図をコンパイルしなければなりません。

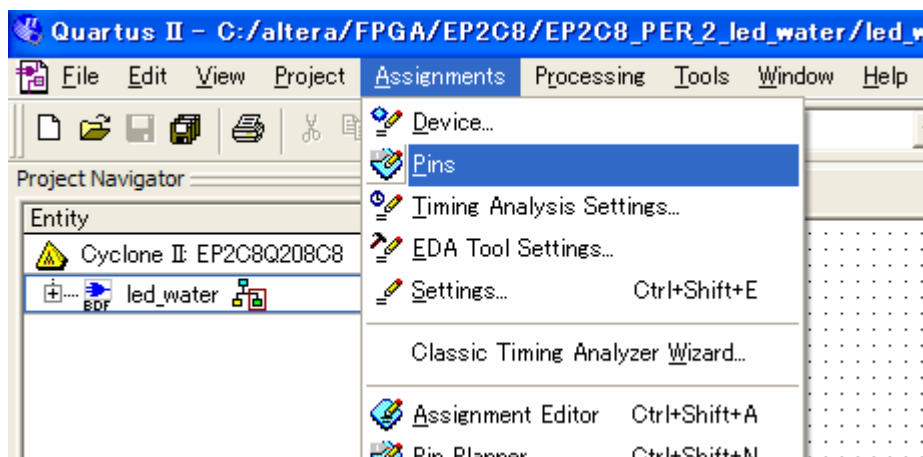
Quartus II の「Processing」メニューから「Start Compilation」を選択します。するとコンパイル処理が始まり、プロGRESS・バーが働き始めます。コンパイルは数十秒で終了します。



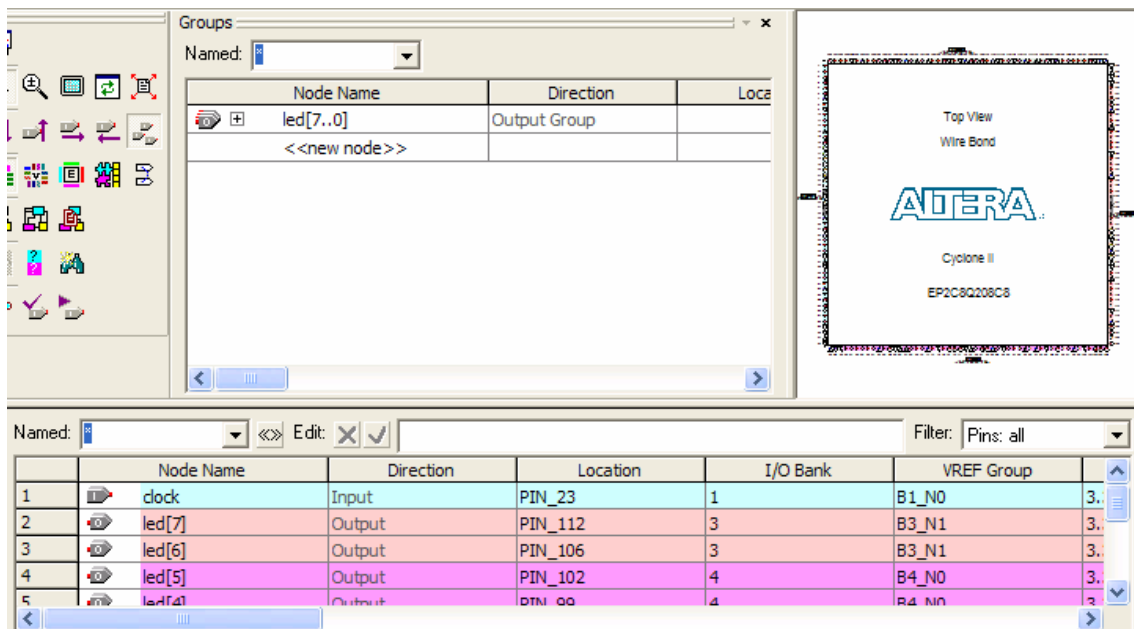
正常にコンパイルが終了しなかった場合は、何か手順を間違えているか、回路図の入力をミスしている可能性があります。画面に表示されたメッセージを読めば、どのようなミスがあるかある程度知ることができます。

4.2.2 回路図の入出力とCPLD/FPGAの端子を関連づける

回路図上の端子名と CPLD/FPGA のピン番号との対応をピン・アサインと言います。ピン・アサインは、「Assignments」メニューの「Pins」を選択します。



Assignments Editor というウィンドウが開きます。



指定したいのはピンの場所、つまり Location です。行の Location の列をダブル・クリックして、ピンの番号を選択します。

| Node Name | Direction | Location | I/O Bank | VREF Group |
|-----------|-----------|----------|----------|--------------------------------|
| clock | Input | PIN_23 | 1 | B1_N0 |
| led[7] | Output | PIN_23 | IOBANK_1 | Dedicated Clock CLK0, LVDSCLK0 |
| led[6] | Output | PIN_24 | IOBANK_1 | Dedicated Clock CLK1, LVDSCLK0 |
| led[5] | Output | PIN_27 | IOBANK_1 | Dedicated Clock CLK2, LVDSCLK1 |
| led[4] | Output | PIN_28 | IOBANK_1 | Dedicated Clock CLK3, LVDSCLK1 |
| | | PIN_30 | IOBANK_1 | Row I/O LVDS7p, DPCLK1 |
| | | PIN_31 | IOBANK_1 | Row I/O LVDS7n |
| | | PIN_33 | IOBANK_1 | Row I/O LVDS6n |
| | | PIN_34 | IOBANK_1 | Row I/O |

ピン・アサインが完成したら、「File」メニューから「Save Project」を選択してプロジェクト全体を保存して、再度コンパイルを実行します。

コンパイル成功すれば、書き込むデータ*`pof`又は*`.sof`を生成します。データをCPLD/FPGAに書き込みましょう。

※ 提供されたサンプルの一部又は全部、回路図で作成したものではありません。VHDL又はVerilogというHDL(Hardware Description Language: ハードウェア記述言語)を使って、作成しました。VHDL/Verilogの使い方はほかの資料を参照してください。

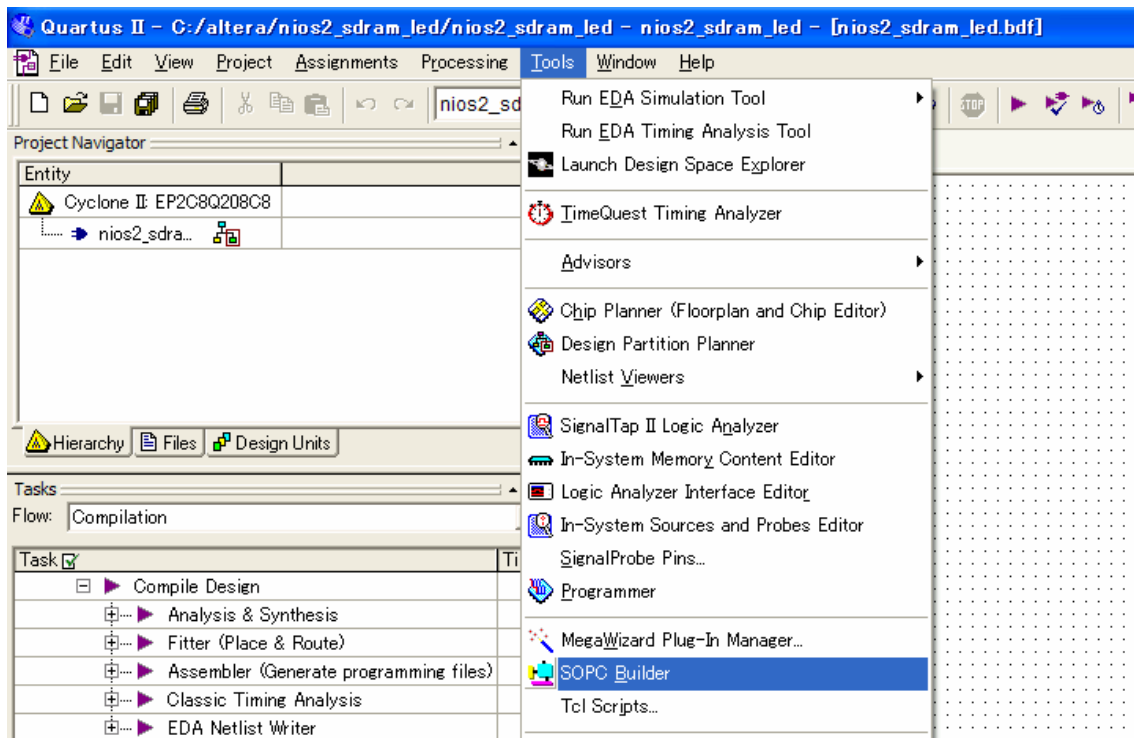
第五章 NIOS IIシステム・モジュールの設計

Cyclone II シリーズ FPGA はソフトプロセッサ NIOS II システムを搭載できます。NIOS II は、32bit CPU、命令・データキャッシュ搭載、最大 250MHz 動作します。とてもハイパフォーマンスなCPUです。開発環境は、ポピュラーなGCCで、無償提供されています。開発環境が信じられないほど簡単です。H8・PICを、お使いの方に使い比べて頂きたいシステムです。

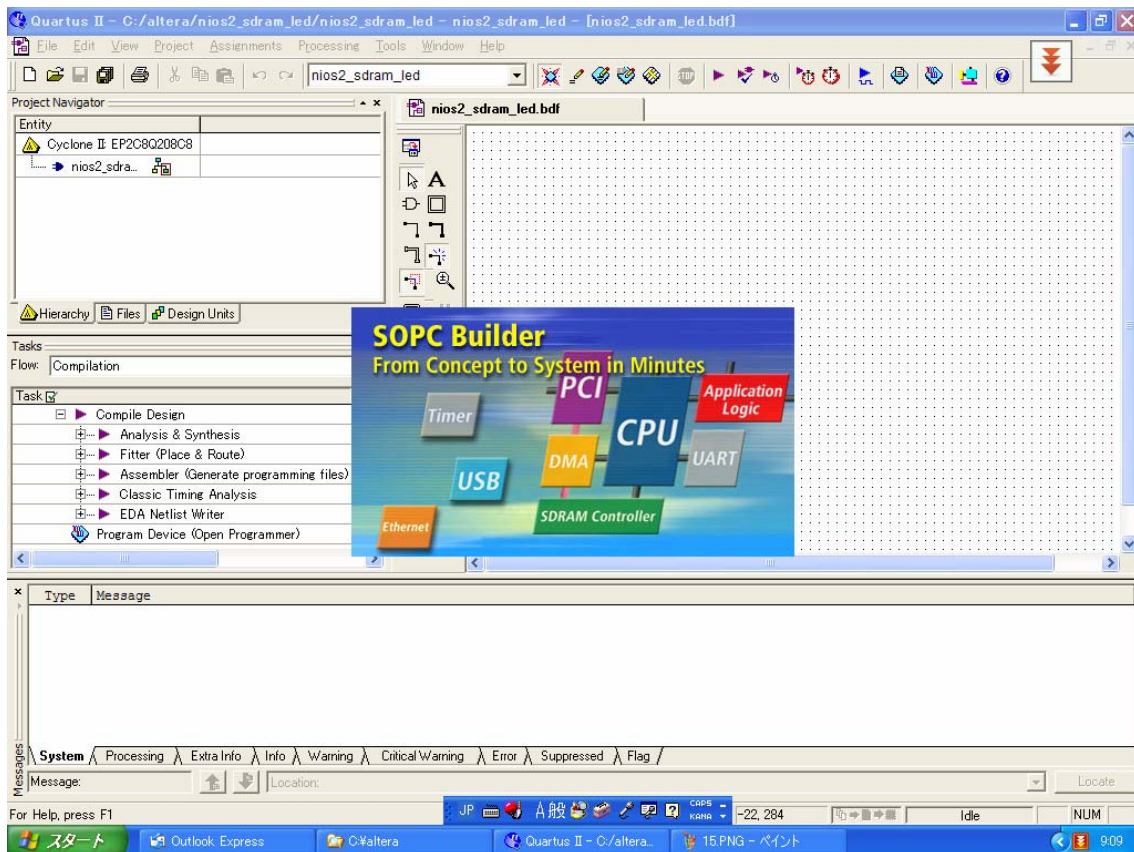
第四章のCPLD/FPGAの開発入門は定番シンボル(例えば74シリーズロジックなど)で回路図を設計します。今回は回路図にCPUを載せます。

まず、Quartus II を起動して、第四章に基づいて、ある空のトップ・エンティティを作ります。

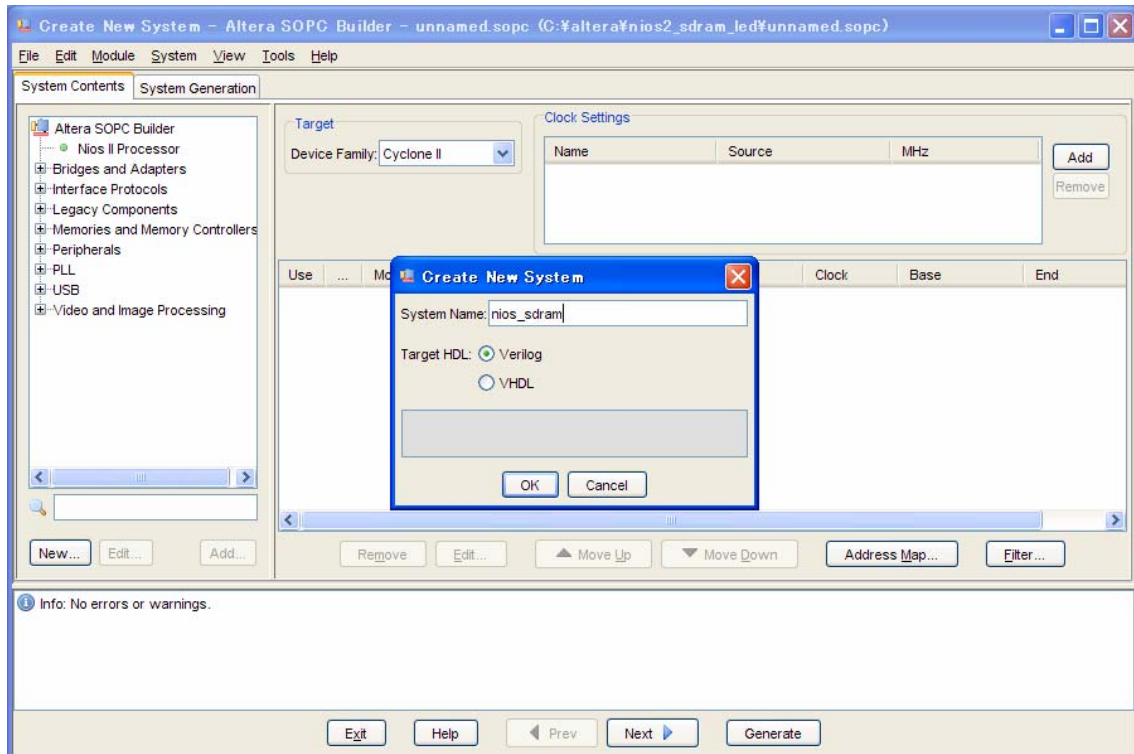
「Tools」メニューから「SOPC Builder」を選択し、SOPC Builder を起動します。



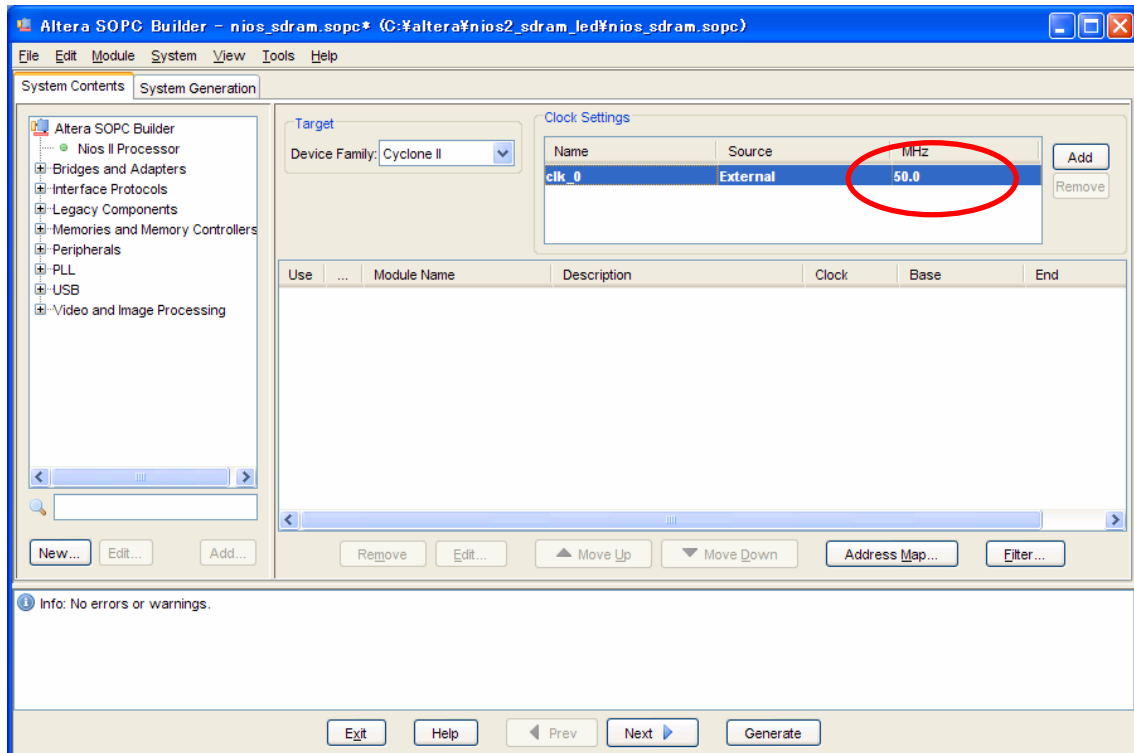
起動の様子です。



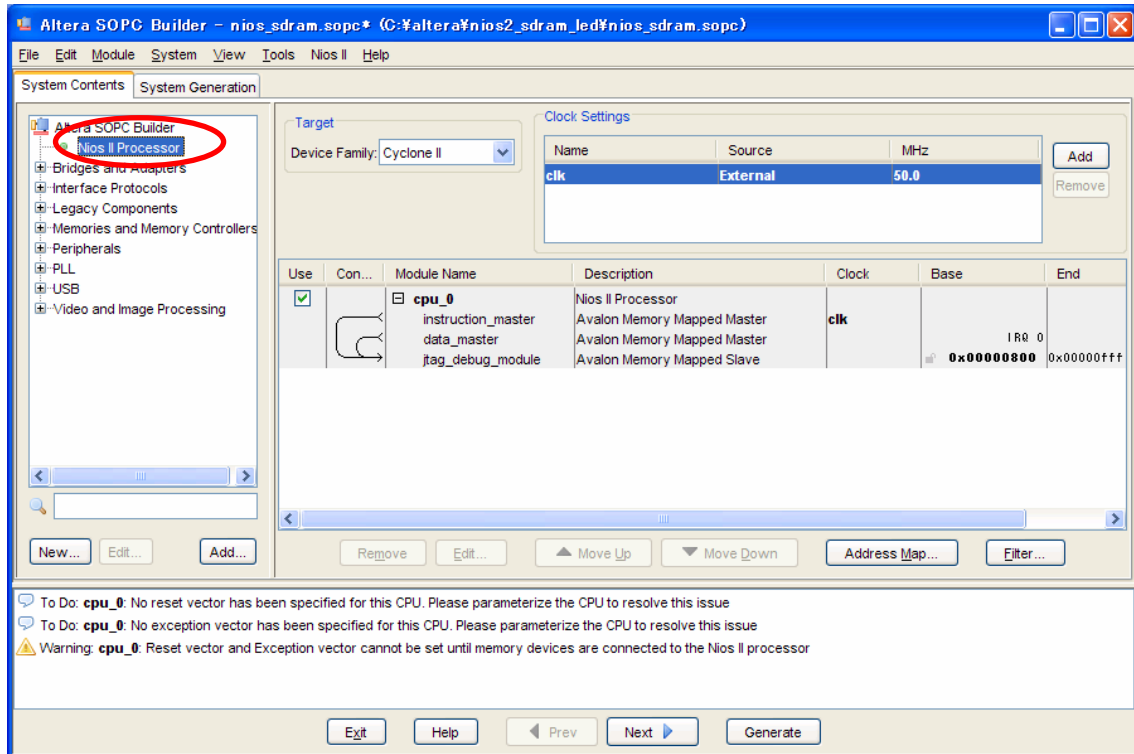
ダイアログに NIOS II システムの名前「nios_sram」を入力してください。名前は必ずトップ・エンティティの名前と異なります。



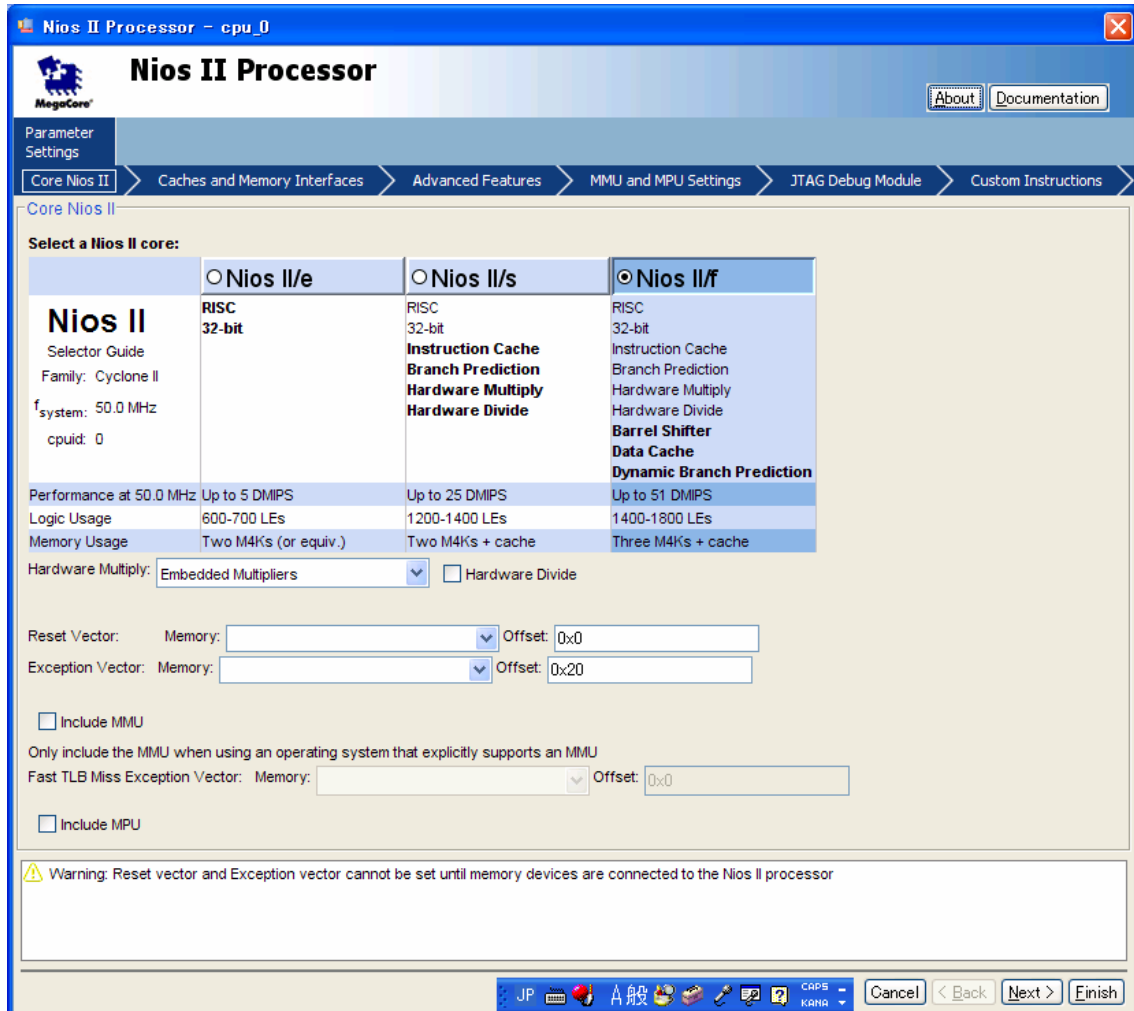
システムの周波数を入力します。今回の例は 50MHz です。周波数の欄でダブル・クリックして、周波数を入力できます。



左側の「Nios II Processor」をダブル・クリックして、Nios II CPU コアを添加します。



Nios II は性能の違う 3 種類の CPU コアが提供されていますが、今回はデフォルトの高性能のグレード(Nios II/f)を使用します。このグレードは、デフォルトでハードウェア乗算器と命令・データキャッシュ、JTAG デバッガが組み込まれています。今回はこれをそのまま使用します。「Finish」ボタンを押してください。



| | <input type="radio"/> Nios II/e | <input type="radio"/> Nios II/s | <input checked="" type="radio"/> Nios II/f |
|--------------------------------|---------------------------------|---------------------------------|--|
| Nios II | RISC 32-bit | RISC 32-bit | RISC 32-bit |
| Selector Guide | | Instruction Cache | Instruction Cache |
| Family: Cyclone II | | Branch Prediction | Branch Prediction |
| f _{system} : 50.0 MHz | | Hardware Multiply | Hardware Multiply |
| cpuId: 0 | | Hardware Divide | Hardware Divide |
| Performance at 50.0 MHz | Up to 5 DMIPS | Up to 25 DMIPS | Up to 51 DMIPS |
| Logic Usage | 600-700 LEs | 1200-1400 LEs | 1400-1800 LEs |
| Memory Usage | Two M4Ks (or equiv.) | Two M4Ks + cache | Three M4Ks + cache |

Hardware Multiply: Hardware Divide

Reset Vector: Memory: Offset:


Exception Vector: Memory: Offset:

Include MMU
Only include the MMU when using an operating system that explicitly supports an MMU

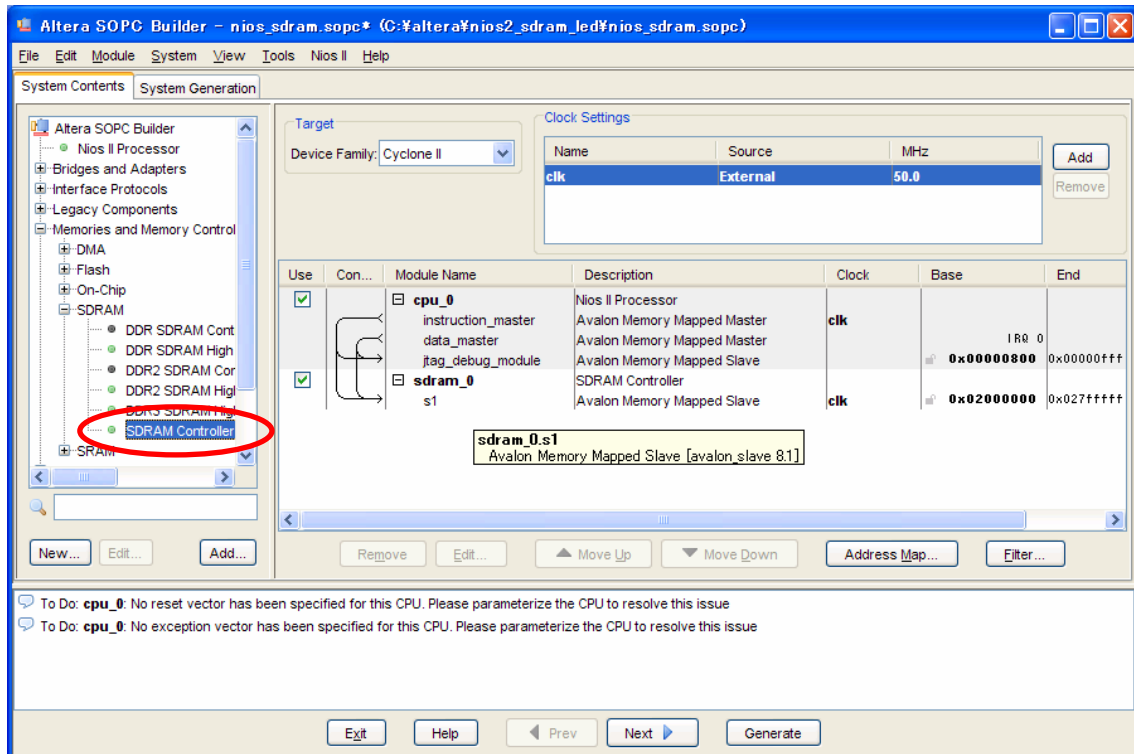
Fast TLB Miss Exception Vector: Memory: Offset:

Include MPU

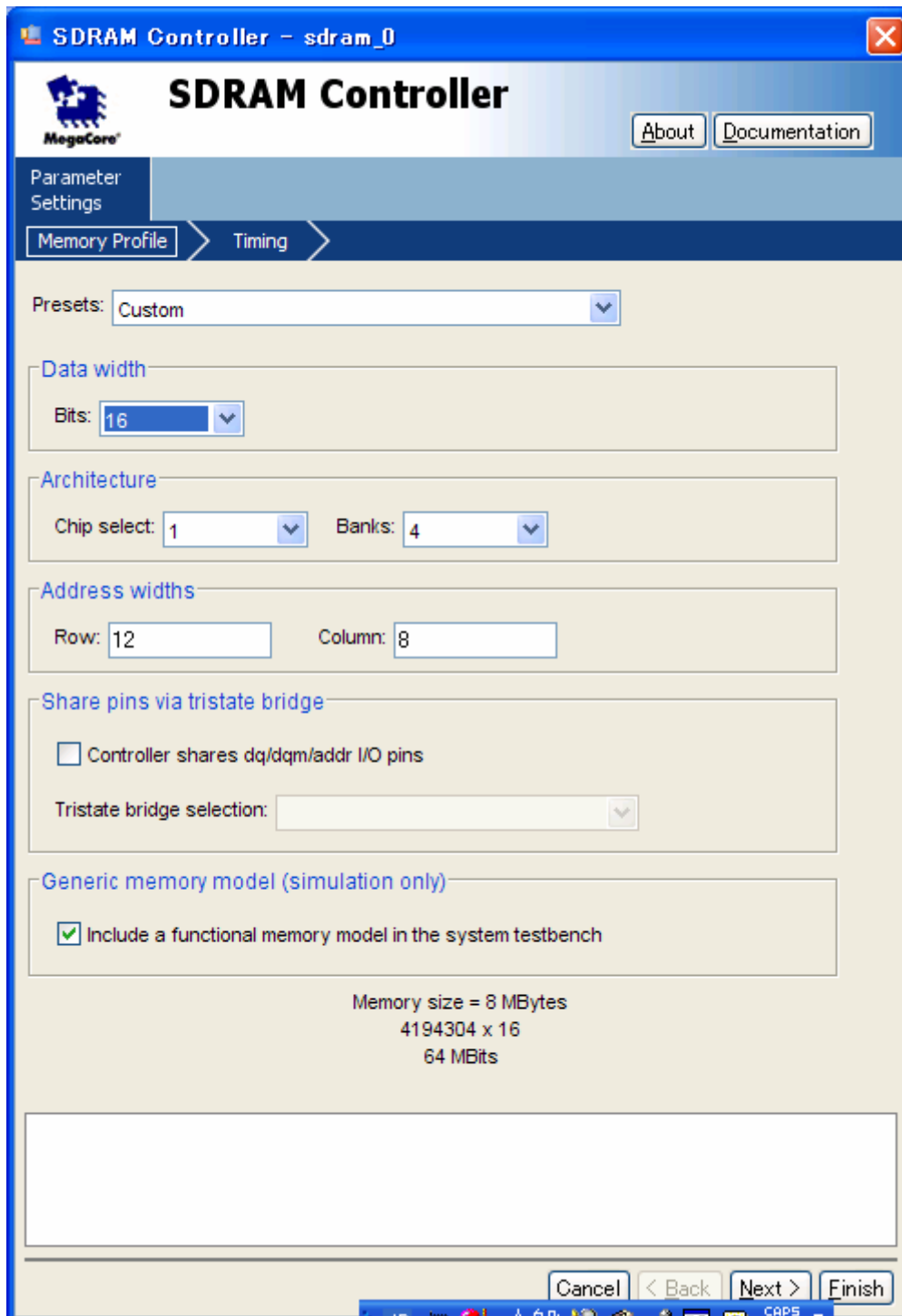
Warning: Reset vector and Exception vector cannot be set until memory devices are connected to the Nios II processor

JP  CAPS KANA

EP2C8 基板の SDRAM を使うため、SDRAM コントローラを Nios II システムに組み込みます。左側の「SDRAM Controller」を選択し、ダブル・クリックして SDRAM コントローラを添加します。

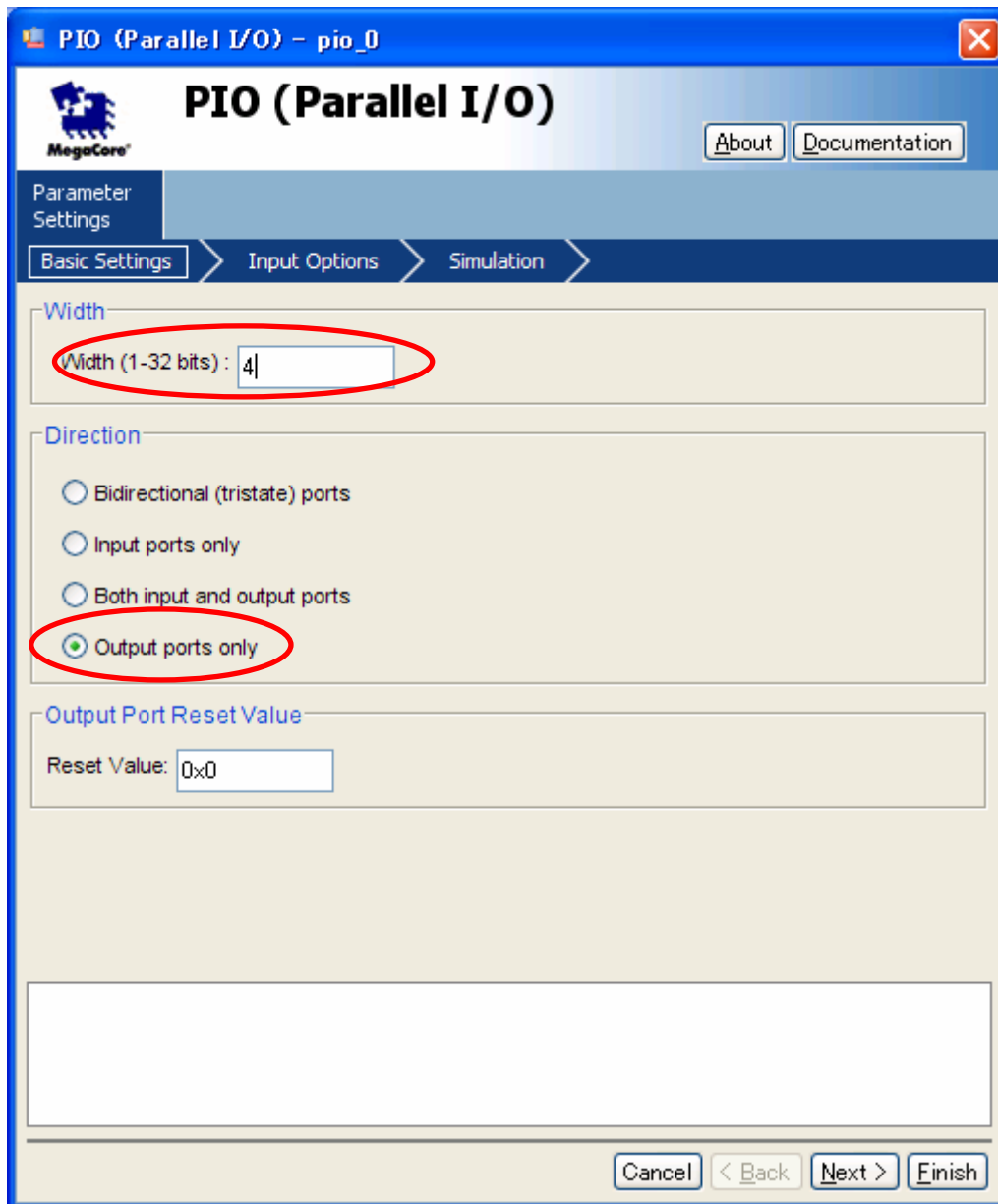


それぞれのメモリごとにデータのビット幅、チップ・セレクトの本数、バンク数、アドレスのロウ・カラムの本数、および AC スパックをデータ・シートから読み取り、間違いのないように入力します。



EP2C8 基板は 4 ビットの LED があります。LED を制御するポートを Nios II システムに

組み込みます。左側の「PIO(Parallel I/O)」を選択し、ダブル・クリックして LED コントローラを添加します。



ビット幅に 4 を入力してください。「Output ports only」を選択します。

ホストと会話するため、JTAG-UART も組み込むことが必要です。左側の「JTAG_UART」を選択し、ダブル・クリックして JTAG-UART を添加します。



The screenshot shows the Altera SOPC Builder interface for a project named 'nios_sdram_socpc'. The 'System Contents' tree on the left has 'JTAG UART' circled in red. The 'System Generation' tab is active, showing the 'Target' set to 'Cyclone II' and 'Clock Settings' with a 'clk' source at 50.0 MHz. A table lists the system components:

| Use | Con... | Module Name | Description | Clock | Base | End |
|-------------------------------------|--------|--------------------|---|-------------------|------------|------------|
| <input checked="" type="checkbox"/> | | cpu_0 | Nios II Processor | | | |
| | | instruction_master | Avalon Memory Mapped Master | clk | 100 0 | |
| | | data_master | Avalon Memory Mapped Master | | 0x00000800 | 0x00000fff |
| | | jtag_debug_module | Avalon Memory Mapped Slave | | | |
| <input checked="" type="checkbox"/> | | sdram_0 | SDRAM Controller | cpu_0.data_master | | |
| | | s1 | Avalon Memory Mapped Slave | | | |
| | | s1 | Avalon Memory Mapped Master [avalon_master 8.1] | | | |
| <input checked="" type="checkbox"/> | | pio_0 | PIO (Parallel I/O) | clk | 0x00000000 | 0x0000000f |
| | | s1 | Avalon Memory Mapped Slave | | | |
| <input checked="" type="checkbox"/> | | jtag_uart_0 | JTAG UART | clk | 0x00000010 | 0x00000017 |
| | | avalon_jtag_slave | Avalon Memory Mapped Slave | | | |

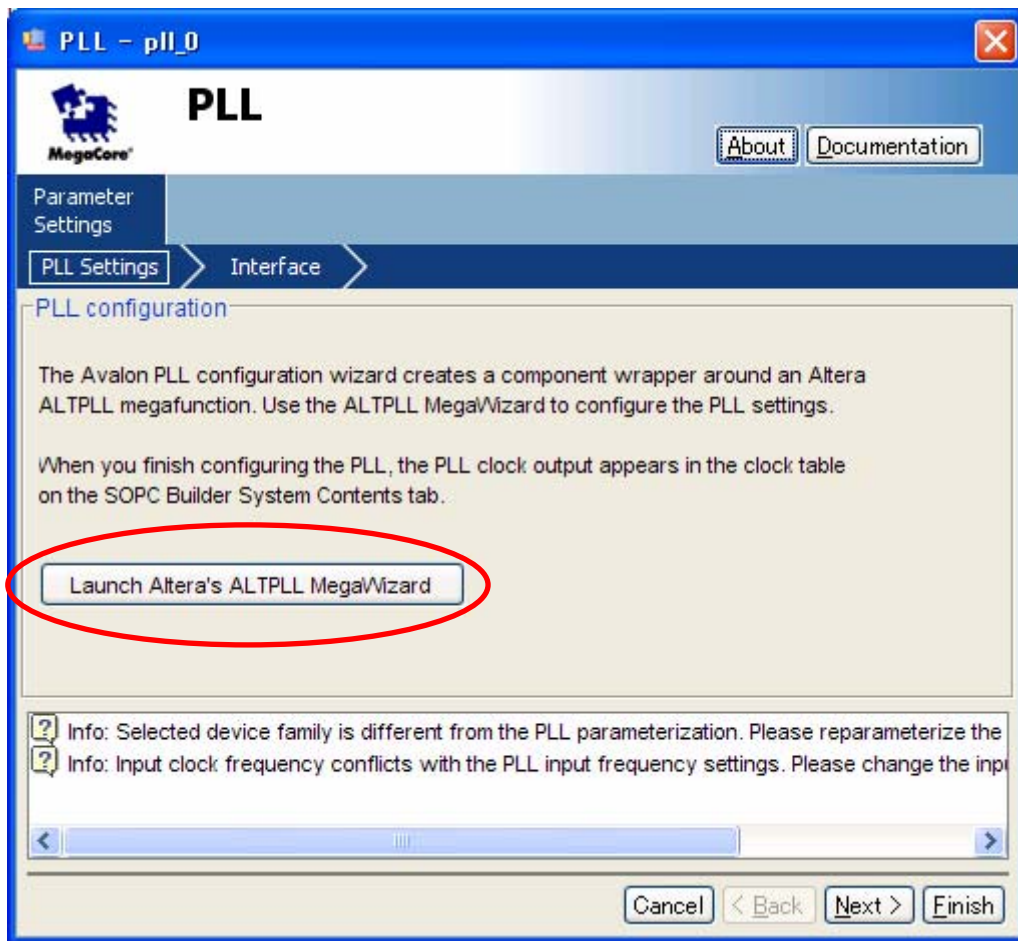
At the bottom, there are two error messages:

- To Do: **cpu_0**: No reset vector has been specified for this CPU. Please parameterize the CPU to resolve this issue
- To Do: **cpu_0**: No exception vector has been specified for this CPU. Please parameterize the CPU to resolve this issue



デフォルトの設定のまま進んでください。

EP2C8 基板は一つの 50MHz の水晶発振器しかありません。ほかの周波数又は位相クロックのため、PLL を Nios II システムに組み込みます。左側の「PLL」を選択し、ダブル・クリックして PLL を添加します。



[Launch Altera's ALTPLL MegaWizard]ボタンを押して、PLL を設定します。

ALTPLL

Currently selected device family: Cyclone II

Match project/default

Able to implement the requested PLL

General

Which device speed grade will you be using? Any

Use military temperature range devices only

What is the frequency of the inclk0 input? 50.00 MHz

Set up PLL in LVDS mode Data rate: 300.000 Mbps

PLL type

Which PLL type will you be using?

Fast PLL

Enhanced PLL

Select the PLL type automatically

Operation mode

How will the PLL outputs be generated?

Use the feedback path inside the PLL

In Normal Mode

In Source-Synchronous Compensation Mode

In Zero Delay Buffer Mode

Connect the fbmimic port (bidirectional)

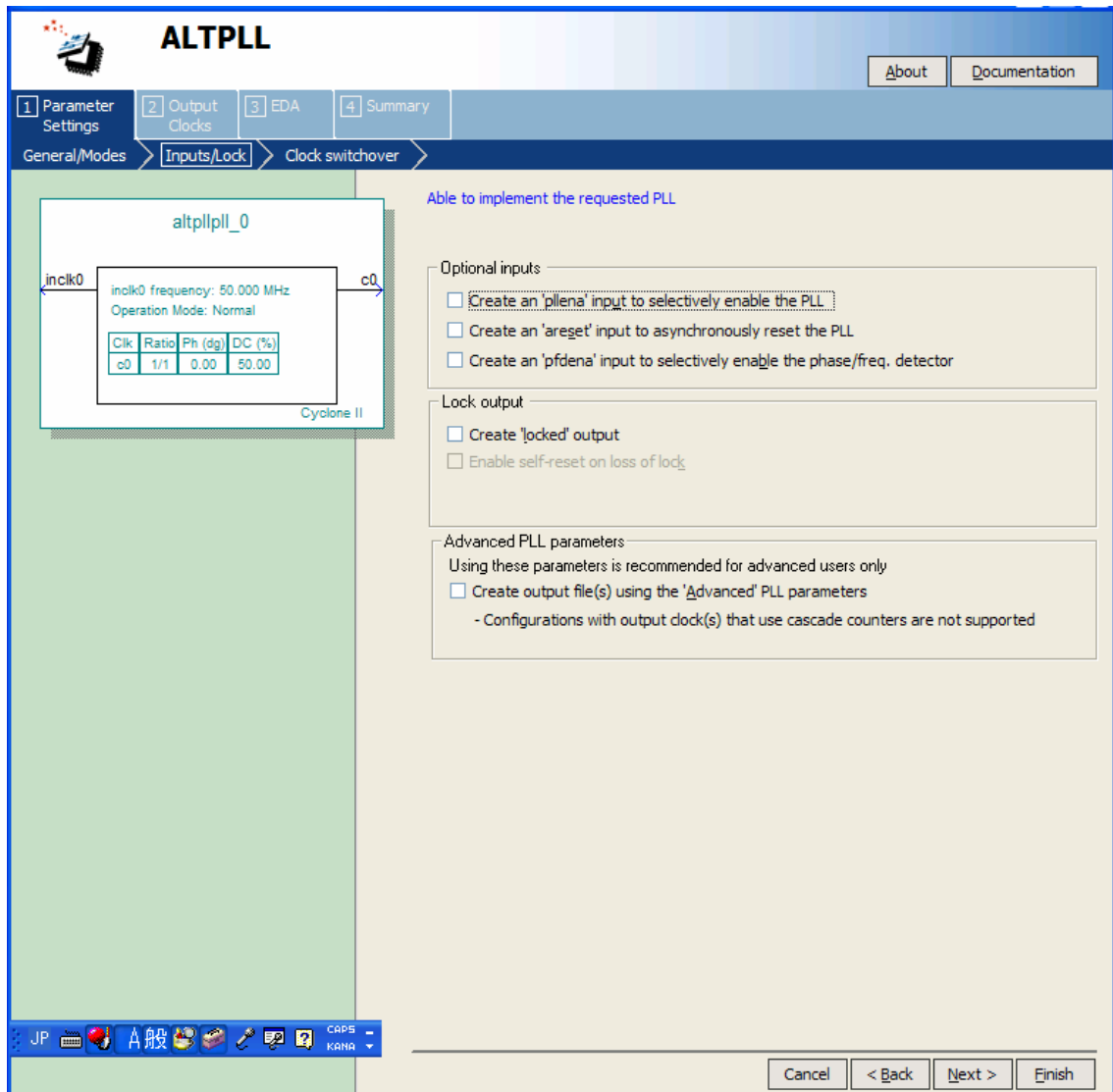
With no compensation

Create an 'fbm' input for an external feedback (External Feedback Mode)

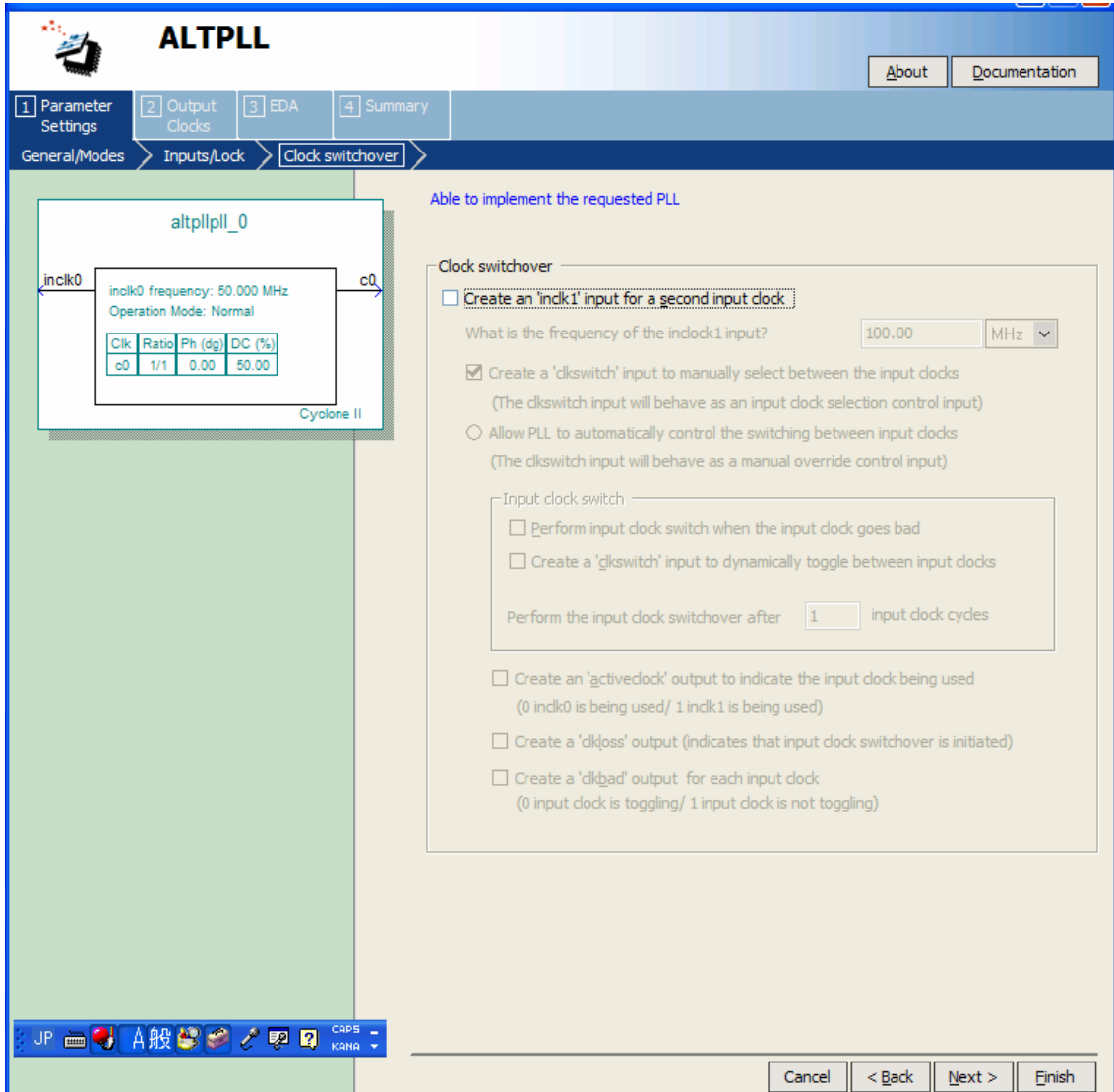
Which output clock will be compensated for? c0

Cancel < Back Next > Finish

入力クロックは 50MHz を入力します。「Next」を押します。



そのまま「Next」を押します。



The screenshot shows the ALTPLL configuration window with the 'Clock switchover' tab selected. The window title is 'ALTPLL' and it has tabs for 'Parameter Settings', 'Output Clocks', 'EDA', and 'Summary'. The 'Clock switchover' tab is active, showing a configuration panel for 'altpll0' and a 'Clock switchover' settings area.

altpll0 Configuration:

- inclk0 frequency: 50.000 MHz
- Operation Mode: Normal

| Clk | Ratio | Ph (dg) | DC (%) |
|-----|-------|---------|--------|
| c0 | 1/1 | 0.00 | 50.00 |

Clock switchover Settings:

- Create an 'inclk1' input for a second input clock
- What is the frequency of the inclk1 input? 100.00 MHz
- Create a 'dkswitch' input to manually select between the input docks (The dkswitch input will behave as an input dock selection control input)
- Allow PLL to automatically control the switching between input docks (The dkswitch input will behave as a manual override control input)
- Input clock switch:**
 - Perform input dock switch when the input dock goes bad
 - Create a 'dkswitch' input to dynamically toggle between input docks
 - Perform the input dock switchover after 1 input dock cycles
- Create an 'activedock' output to indicate the input dock being used (0 inclk0 is being used/ 1 inclk1 is being used)
- Create a 'dkloss' output (indicates that input dock switchover is initiated)
- Create a 'dkbad' output for each input dock (0 input dock is toggling/ 1 input dock is not toggling)

Buttons at the bottom: Cancel, < Back, Next >, Finish.

そのまま「Next」を押します。

そのまま「Next」を押します。

ALTPLL

1 Parameter Settings 2 Output Clocks 3 EDA 4 Summary

dk c0 > dk c1 > dk c2 >

altpll0

inclk0 inclk0 frequency: 50.000 MHz
Operation Mode: Normal

| Clk | Ratio | Ph (dg) | DC (%) |
|-----|-------|---------|--------|
| c0 | 1/1 | 0.00 | 50.00 |
| c1 | 1/1 | -63.00 | 50.00 |

Cyclone II

c1 - Core/External Output Clock
Able to implement the requested PLL

Use this clock

Clock Tap Settings

| | Requested settings | Actual settings |
|--------------------------------|--------------------|-----------------|
| Enter output clock frequency: | 100.00000000 MHz | 50.000000 |
| Enter output clock parameters: | | |
| Clock multiplication factor | 1 | 1 |
| Clock division factor | 1 | 1 |
| Clock phase shift | -63.00 deg | -63.00 |
| Clock duty cycle (%) | 50.00 | 50.00 |

More Details >>

Per Clock Feasibility Indicators

c0 c1 c2

Cancel < Back Next > Finish

新しい出力クロックを添加します。クロックの位相は-63に設定してください。このクロックはSDRAM用の制御クロックです。「Next」を押します。

Turn on the files you wish to generate. A gray checkmark indicates a file that is automatically generated, and a red checkmark indicates an optional file. Click Finish to generate the selected files. The state of each checkbox is maintained in subsequent MegaWizard Plug-In Manager sessions.

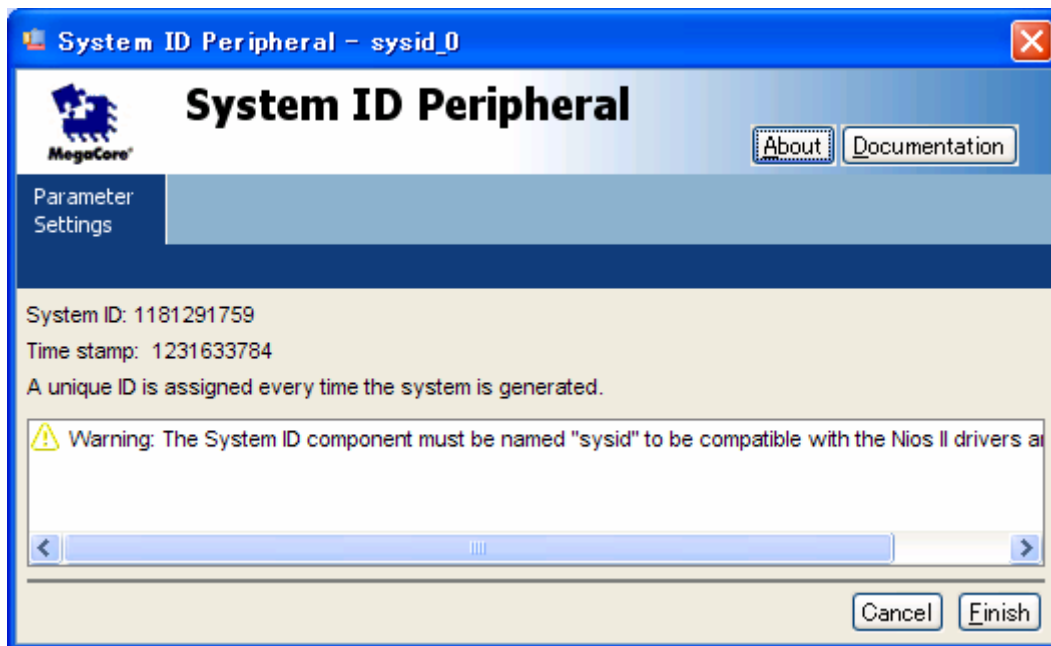
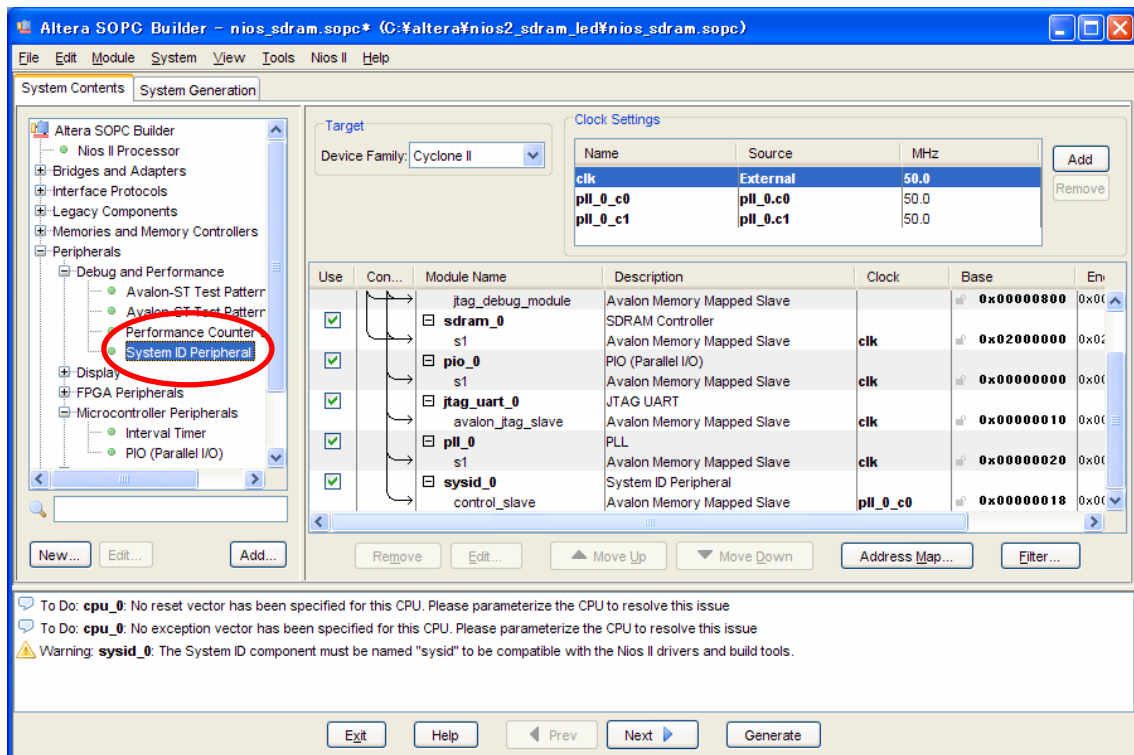
The MegaWizard Plug-In Manager creates the selected files in the following directory:

C:\altera\nios2_sdrum_led\

| File | Description |
|--|---------------------------------|
| <input checked="" type="checkbox"/> altpll0.v | Variation file |
| <input checked="" type="checkbox"/> altpll0.ppf | PinPlanner ports PPF file |
| <input type="checkbox"/> altpll0.inc | AHDL Include file |
| <input type="checkbox"/> altpll0.cmp | VHDL component declaration file |
| <input checked="" type="checkbox"/> altpll0.bsf | Quartus II symbol file |
| <input type="checkbox"/> altpll0_inst.v | Instantiation template file |
| <input checked="" type="checkbox"/> altpll0_bb.v | Verilog HDL black-box file |
| <input checked="" type="checkbox"/> altpll0_waveforms.html | Sample waveforms in summary |
| ... altpll0_wave*.jpg | Sample waveform file(s) |

最後の確認です。「Finish」を押します。

Sysid モジュールは、Altera 社のソフトウェア開発環境 IDE を用いてソフトウェアをダウンロードする際に、ハードウェアとの整合性の確認に利用する ID 情報を格納するモジュールです。ここで読み出せる値は、SOPC Builder でロジックを生成するときと与えられるものです。左側の「System ID Peripheral」を選択し、ダブル・クリックして Sysid を添加します。

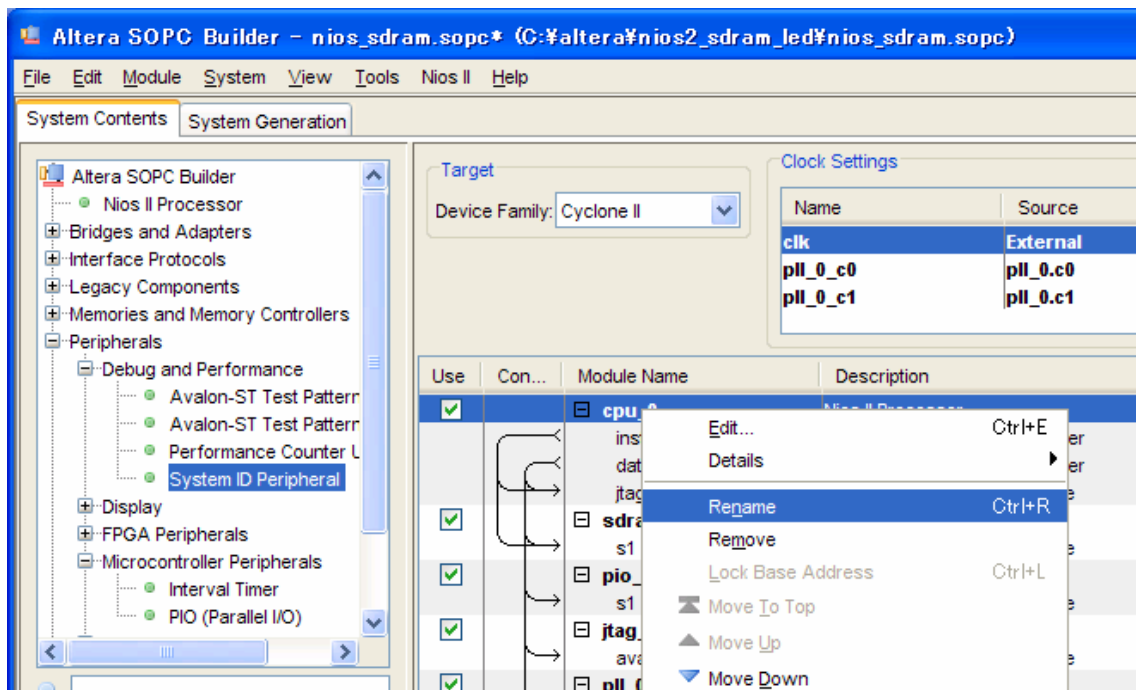


[Finish]を押して、Sysid を生成します。

これまで、必要なモジュールをすべて組み込みました。

| Use | Con... | Module Name | Description | Clock | Base | End | IRQ |
|-------------------------------------|--------|--------------------|-----------------------------|--------|------------|------------|--------|
| <input checked="" type="checkbox"/> | | cpu | Nios II Processor | | | | |
| | | instruction_master | Avalon Memory Mapped Master | pll_c0 | | | |
| | | data_master | Avalon Memory Mapped Master | | | | IRQ 0 |
| | | jtag_debug_module | Avalon Memory Mapped Slave | | 0x00000800 | 0x00000fff | IRQ 31 |
| <input checked="" type="checkbox"/> | | sdram | SDRAM Controller | | | | |
| | | s1 | Avalon Memory Mapped Slave | pll_c0 | 0x02000000 | 0x027fffff | |
| <input checked="" type="checkbox"/> | | led_pio | PIO (Parallel I/O) | | | | |
| | | s1 | Avalon Memory Mapped Slave | pll_c0 | 0x00000000 | 0x0000000f | |
| <input checked="" type="checkbox"/> | | jtag_uart | JTAG UART | | | | |
| | | avalon_jtag_slave | Avalon Memory Mapped Slave | pll_c0 | 0x00000010 | 0x00000017 | |
| <input checked="" type="checkbox"/> | | sysid | System ID Peripheral | | | | |
| | | control_slave | Avalon Memory Mapped Slave | pll_c0 | 0x00000018 | 0x0000001f | |
| <input checked="" type="checkbox"/> | | pll | PLL | | | | |
| | | s1 | Avalon Memory Mapped Slave | clk | 0x00000040 | 0x0000005f | |

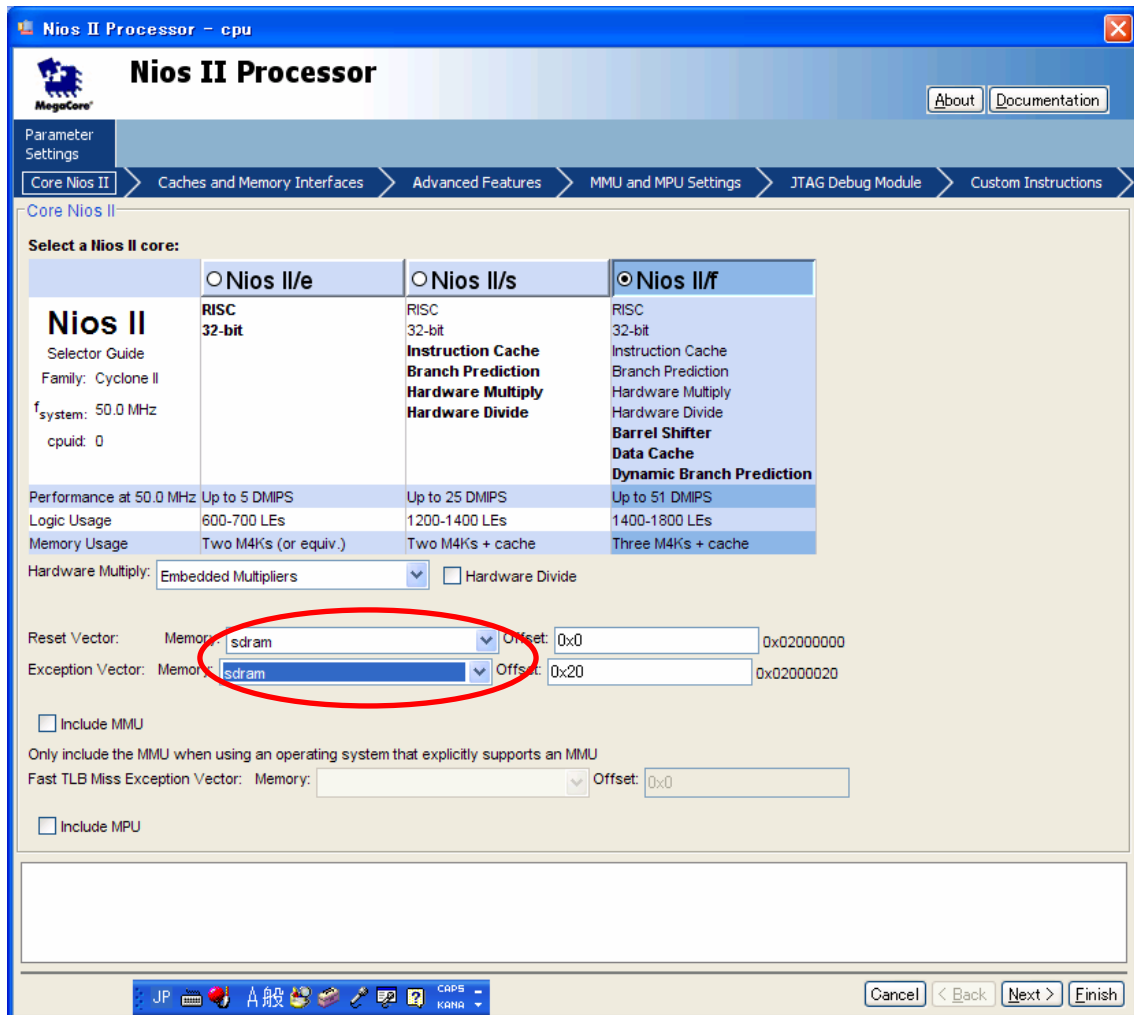
モジュールの名前はわかりやすい名前に変更します。モジュールの名前欄で右クリックして、出てきたメニューから「Rename」を選択します。



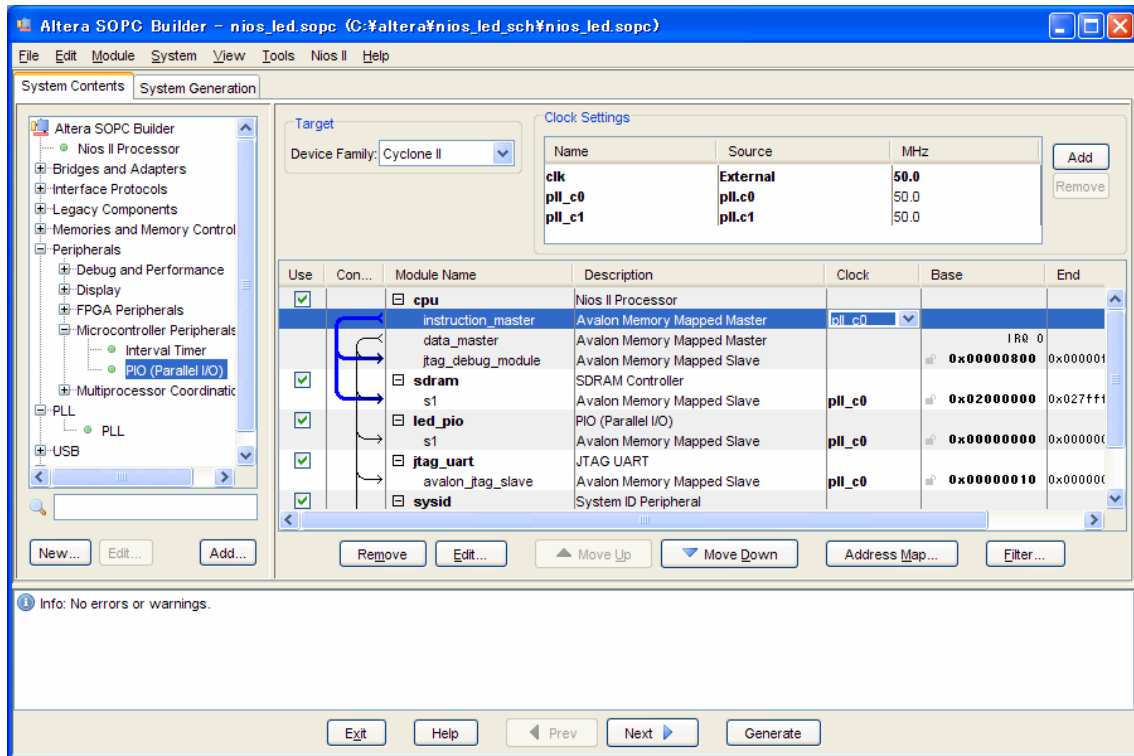
モジュール用のクロックを選択します。[Clock]欄でダブル・クリックして、PLLのクロック pll_c0を選択します。PLLモジュールは外部クロック「clk」を使用します。

| Use | Con... | Module Name | Description | Clock | Base | End | IRQ |
|-------------------------------------|--------|--------------------|-----------------------------|--------|------------|------------|--------|
| <input checked="" type="checkbox"/> | | cpu | Nios II Processor | | | | |
| | | instruction_master | Avalon Memory Mapped Master | pll_c0 | | | |
| | | data_master | Avalon Memory Mapped Master | clk | | | IRQ 0 |
| | | jtag_debug_module | Avalon Memory Mapped Slave | pll_c0 | 0x00000800 | 0x00000fff | IRQ 31 |
| <input checked="" type="checkbox"/> | | sdram | SDRAM Controller | | | | |
| | | s1 | Avalon Memory Mapped Slave | pll_c0 | 0x02000000 | 0x027fffff | |
| <input checked="" type="checkbox"/> | | led_pio | PIO (Parallel I/O) | | | | |
| | | s1 | Avalon Memory Mapped Slave | pll_c0 | 0x00000000 | 0x0000000f | |
| <input checked="" type="checkbox"/> | | jtag_uart | JTAG UART | | | | |
| | | avalon_jtag_slave | Avalon Memory Mapped Slave | pll_c0 | 0x00000010 | 0x00000017 | |
| <input checked="" type="checkbox"/> | | sysid | System ID Peripheral | | | | |
| | | control_slave | Avalon Memory Mapped Slave | pll_c0 | 0x00000018 | 0x0000001f | |
| <input checked="" type="checkbox"/> | | pll | PLL | | | | |
| | | s1 | Avalon Memory Mapped Slave | clk | 0x00000040 | 0x0000005f | |

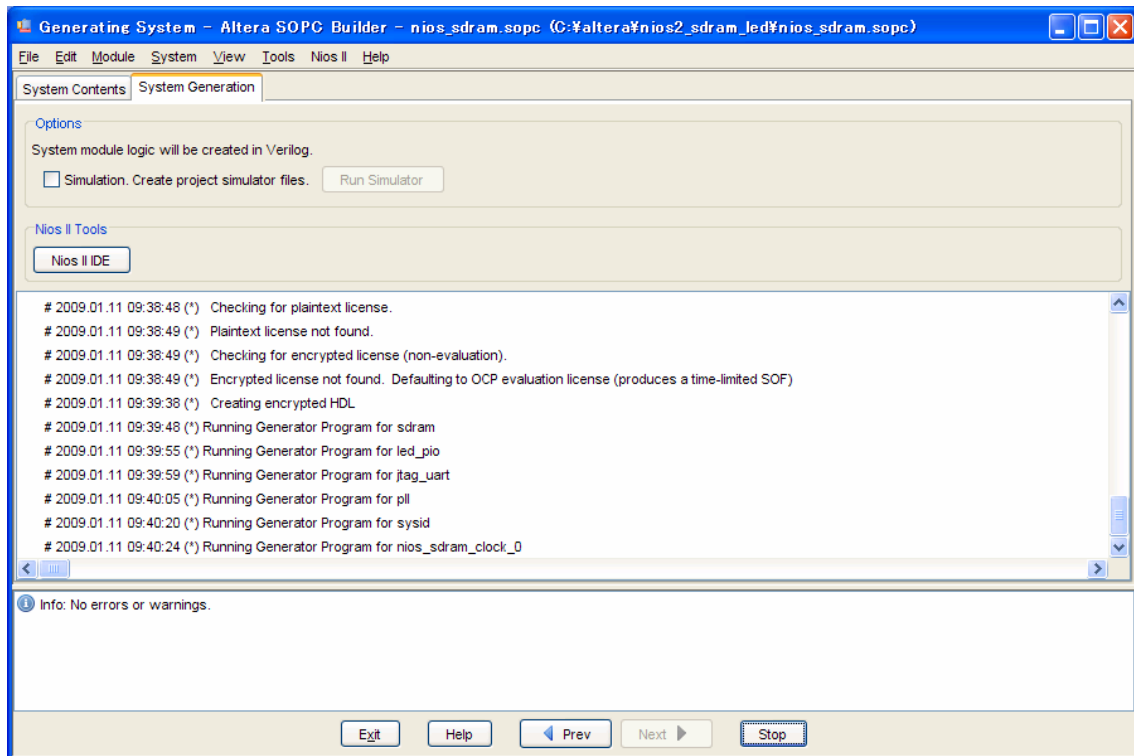
もう一つの設定は CPU のリセットと実行アドレスです。モジュール CPU の名前欄でダブル・クリックして、CPU 設定のダイアログを再び開きます。



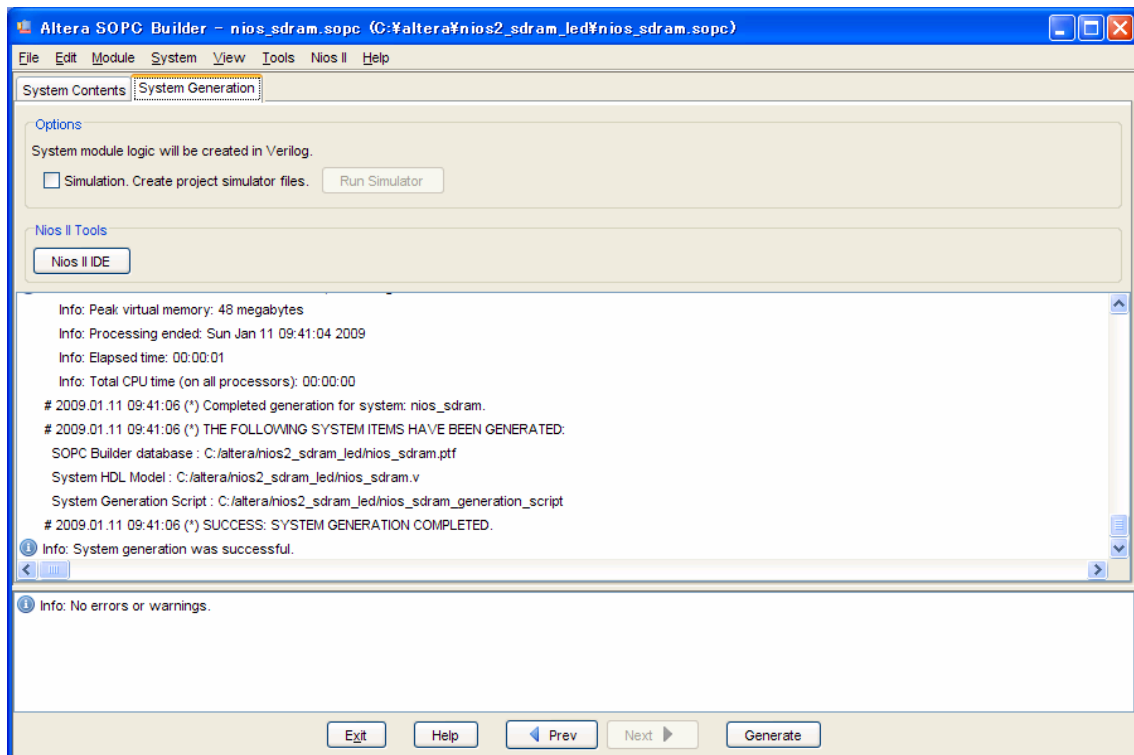
リセットと実行アドレスを SDRAM に設定します。「Finish」を押します。



全部の設定が完了しました。「Generate」ボタンを押して、Nios II システムを生成します。

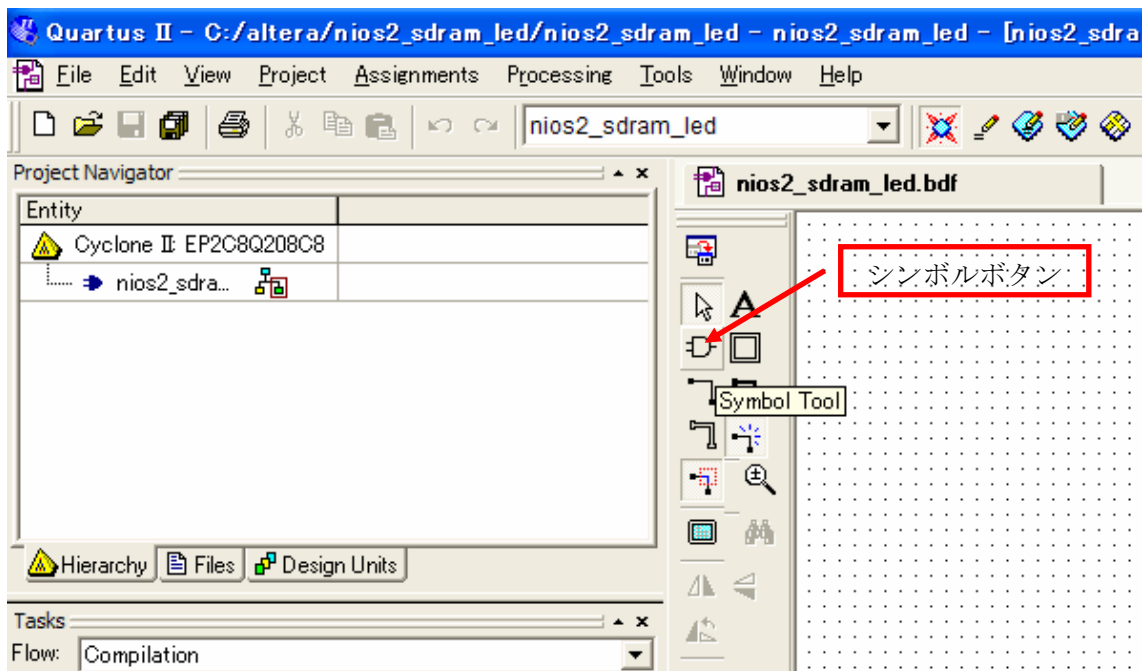


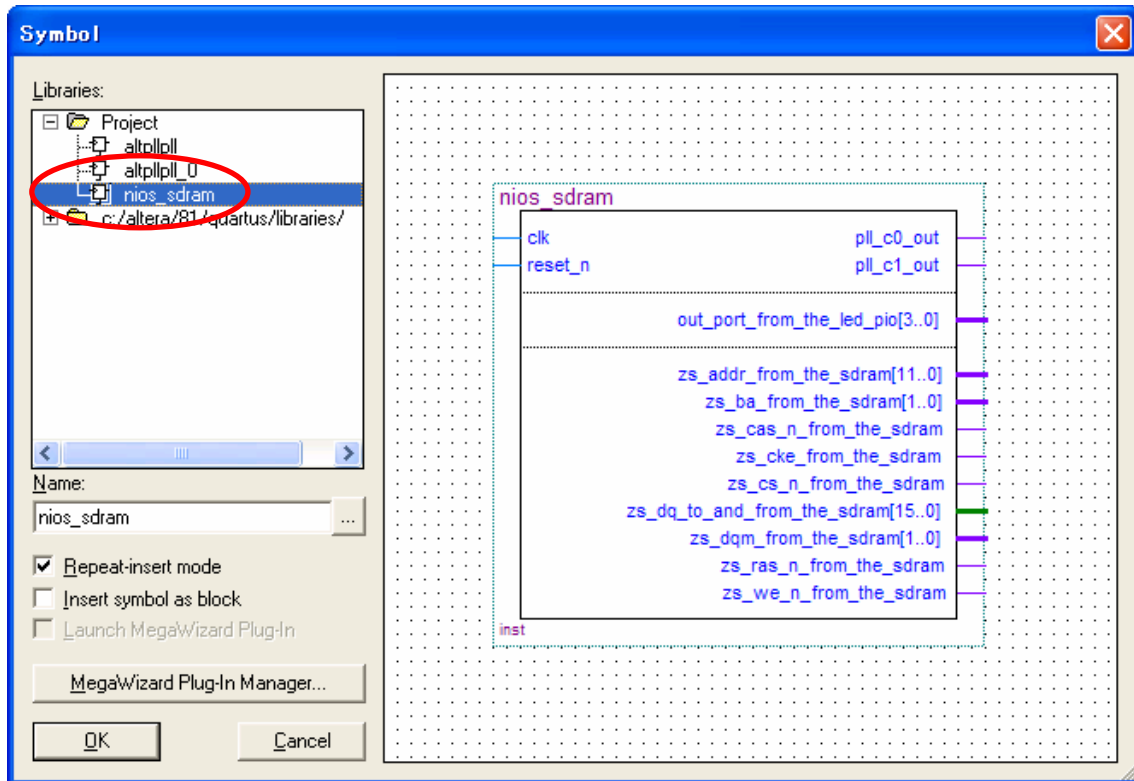
Nios II システムを生成中です。



生成完了すると、「Exit」を押して、Quartus IIに戻します。

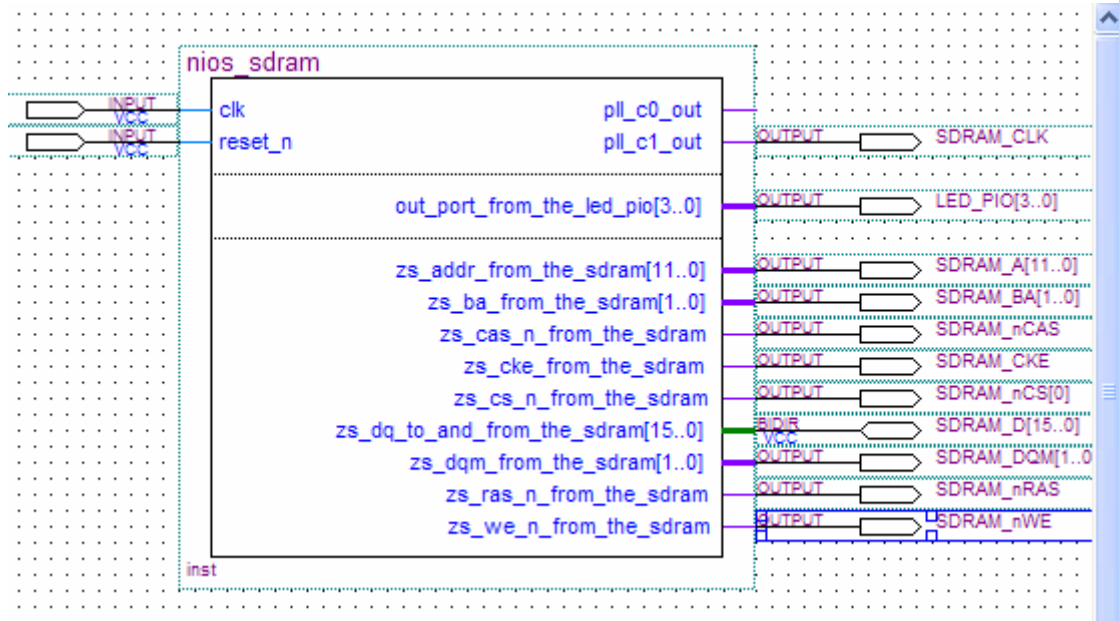
Quartus IIで再びシンボルボタンを押すと、





Libraries 欄に、Project 前の[+]記号をクリックすると、中身が表示されます。生成された Nios II システム「nios_sdram」が見えました！nios_sdram を選択して「OK」ボタンを押します。マウス・カーソルに選択されたシンボルがくっついた状態になるので、シンボルを配置したい場所へ移動して、マウスの左クリックを押します。配置できた時点で ESC キーを押してカーソルを元に戻してください。

次の作業は第四章と同じです。Nios II のシンボルに INPUT/OUTPUT などの端子を配置します。端子の名前は分かりやすい名前に変更してください。



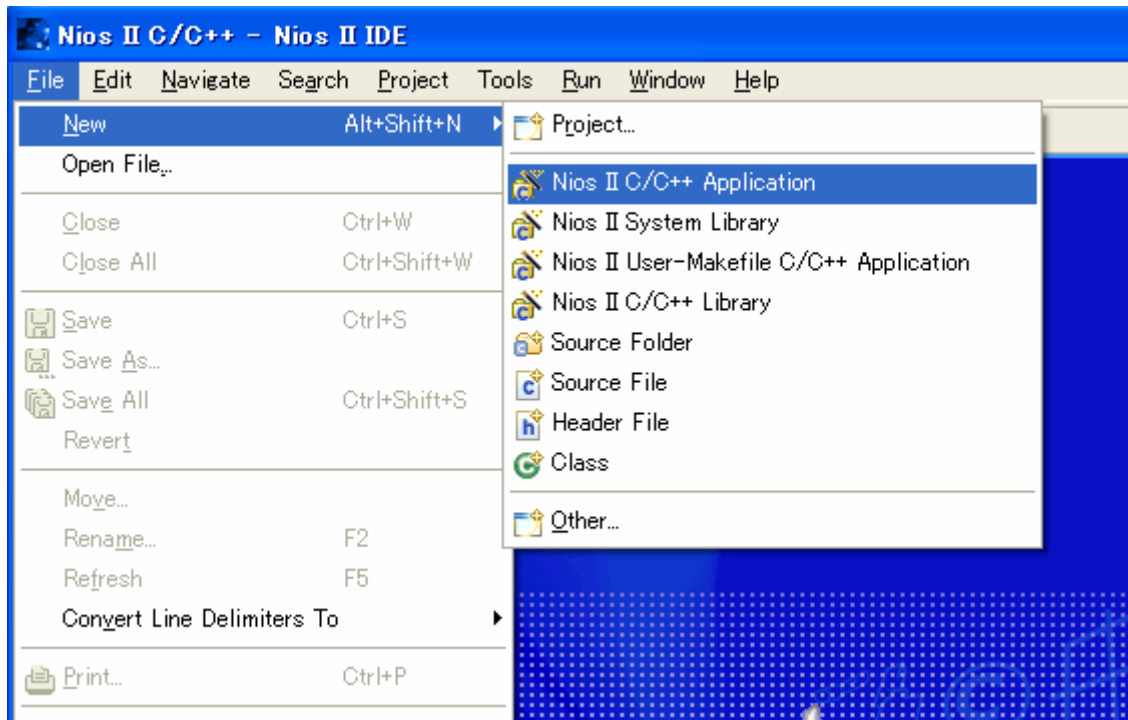
配置が終わったら、コンパイル、ピン・アサイン、再コンパイルを行います。Cyclone II に書き込むデータ*.sof を生成します。このファイルを Cyclone II 書き込みます。

※ EP2C8¥nios_led_sch というフォルダに、すでに完成しているプロジェクト一式があります。

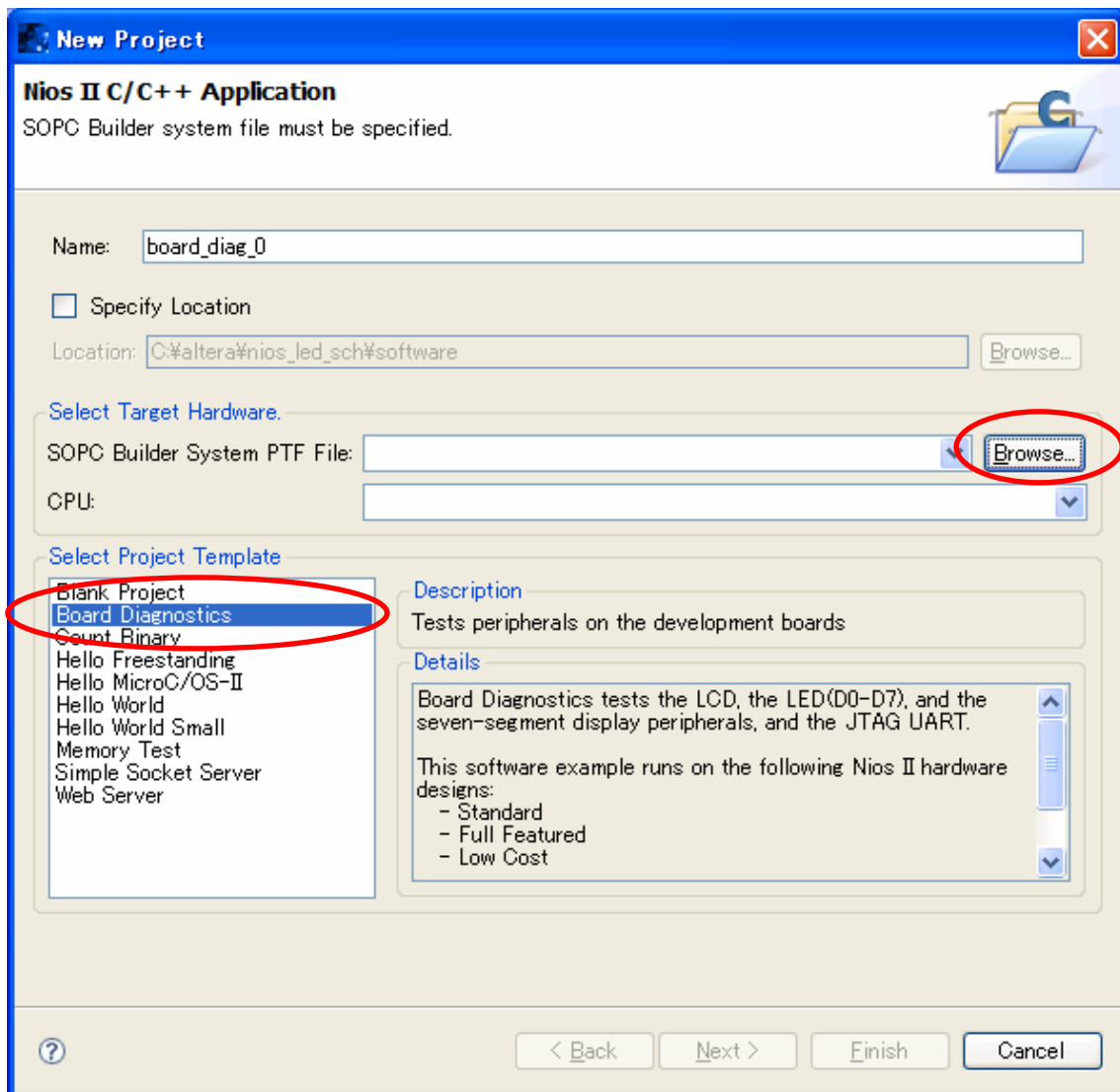
第六章 NIOS II のプログラムの設計

Nios II システムを構築しました。これから、Nios II システムのプログラムを開発します。

Windows の「スタート」→「すべてのプログラム」→「Altera」→「NIOS II EDS 8.1」から NIOS II 8.1 IDE が起動します。

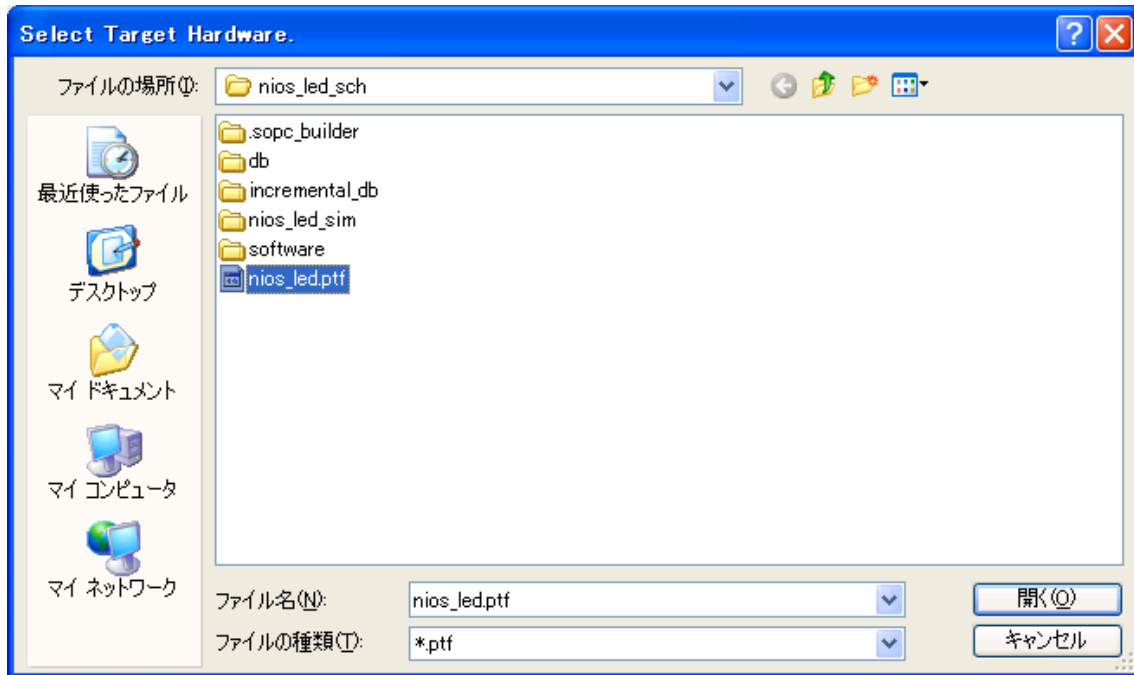


「File」 → 「New」 → 「Nios II C/C++ Application」 を選択します。

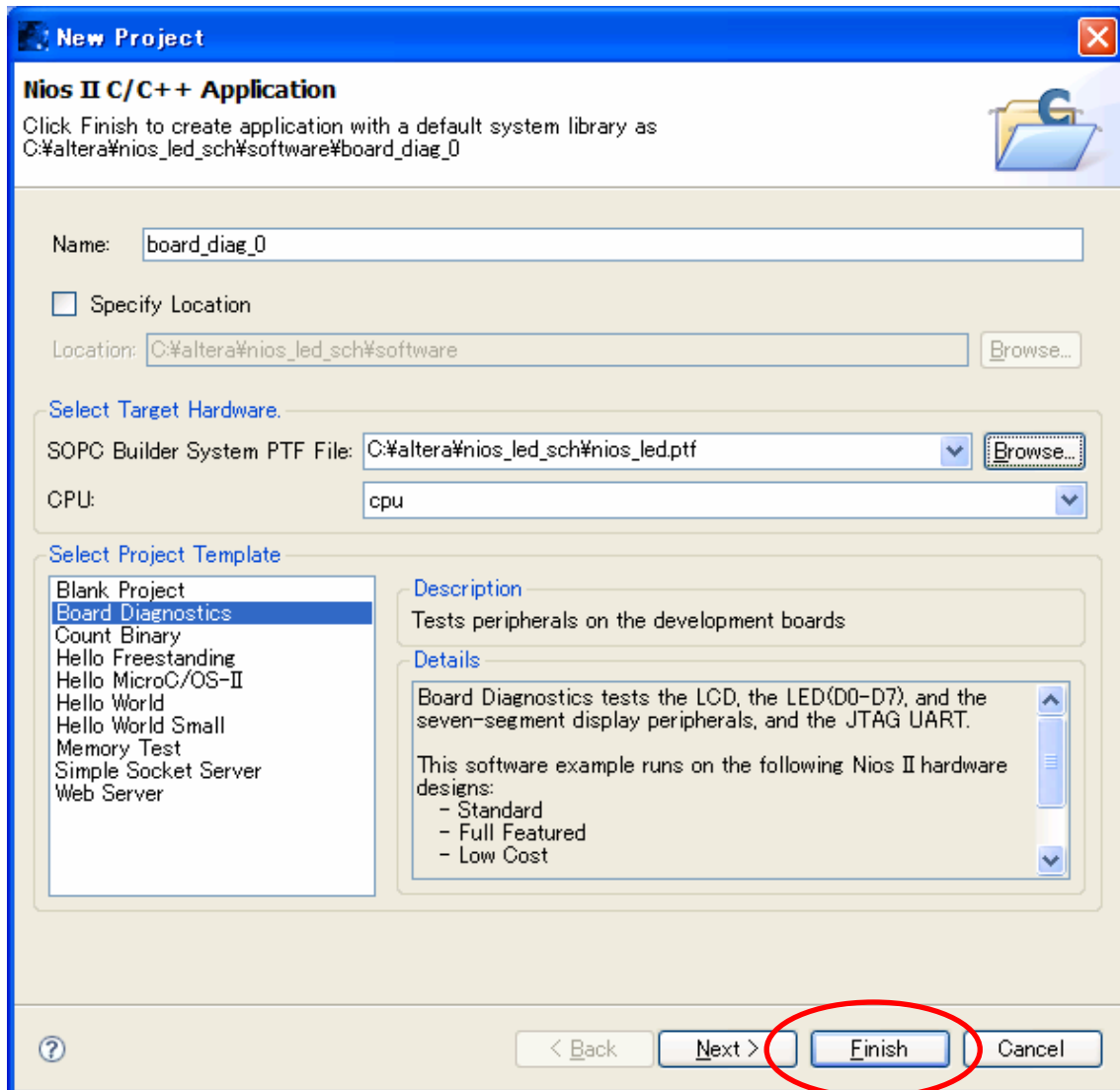


Altera 社は幾つの定番テンプレートを提供しています。これらテンプレートに基づいて、プログラムを開発しやすいです。もちろん、ゼロ「Blank Project」から開発もできます。今回は「Board Diagnostics」を選択します。一行のコードも入力することが必要ないです。

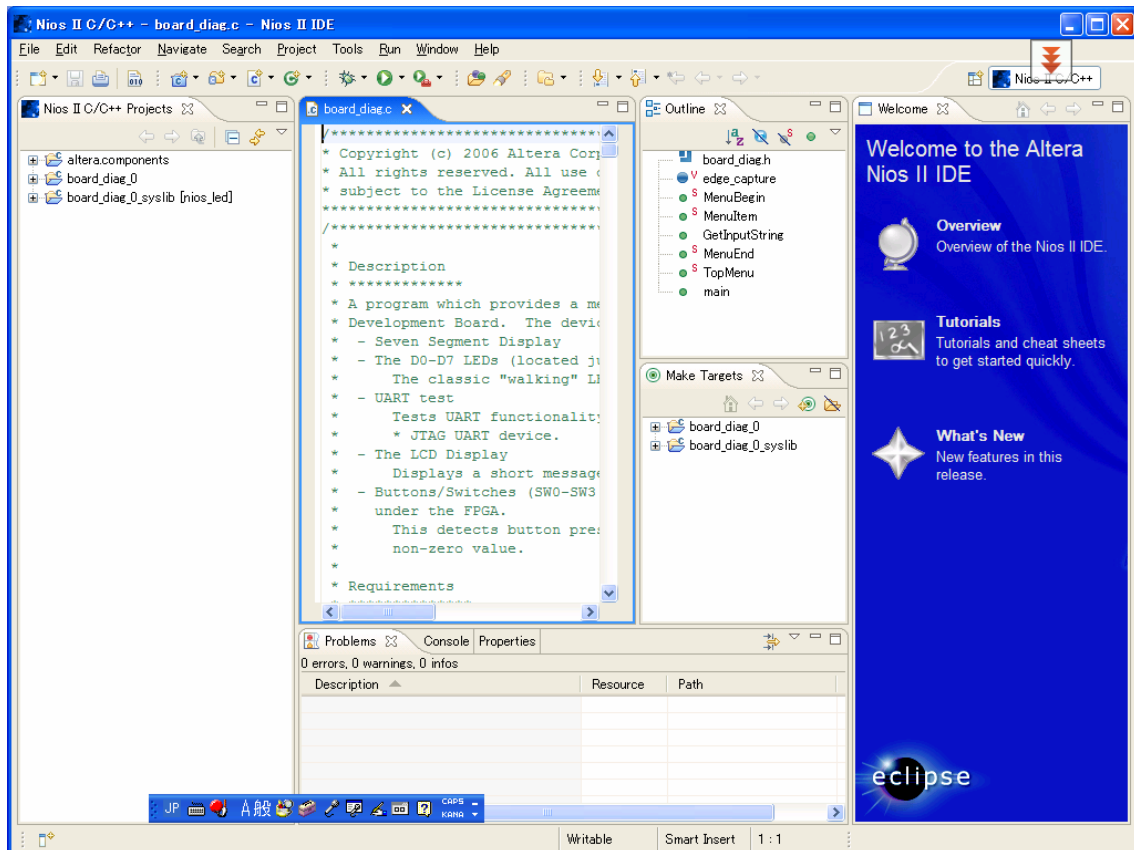
「Browse」ボタンを押して、



Nios II システムの*.ptf ファイルを選択します。



「Finish」ボタンを押すと、プログラムとハードウェアに関連するライブラリを自動的に生成します。



次の手順は「3.5 NIOS II プロセッサの初体験」で紹介いたしました。