無償回路ライブラリを利用した組込システム開発

ー USB2.0 と Nios2/e コアを組み合わせた HDL コーディング不要のシステム設計ー

1. 目的

USB2.0 インタフェース組込済みの Smart-USB Plus システム開発ボード向けに、無償で利用できるアルテラ FPGA 向けソフト CPU の Nios2/e（以下、Nios2）を採用した組込システム開発例としてリファレンス回路を提供します。 このリファレンス回路を適用して Smart-USB Plus 製品を USB インタフェース付きマイコンボードとして利用できます。

2. 必要な開発環境とダウンロードケーブル

- ✓ QuratusII ver11.1SP2 WebEdition　：FPGA 回路設計用
- ✓ Nios2 Embedded Design Suite（NIOS II EDS）　：ソフトウエア開発用（重要 Quartus2 と同バージョンを選択してください）

- ✓ アルテラ専用のダウンロードケーブル（USB Blaster）　：ソフトウエアデバッグおよび FPGA コンフィグ ROM 書き込み用
  （デバッグ機能を利用するときに必要です。デバッグ機能を利用しない場合は不要です。）

【ターゲット・ボード】

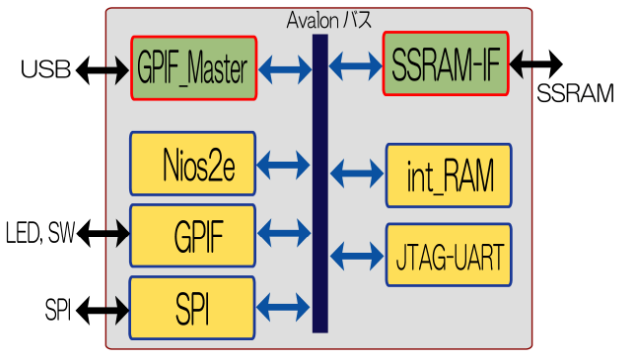
- ✓ CX-USBII システム開発ボード　（Cyclone3 FPGA 搭載：この資料で説明するボードです）
- ✓ CX-CardII システム開発ボード　（Cyclone2 FPGA 搭載）
- ✓ SX-USBII システム開発ボード　（Stratix2 FPGA 搭載、一部のモデルで無償 QuartusII が利用できません）
- ✓ SX-USBIII システム開発ボード　（Stratix3 FPGA 搭載）
- ✓ CX-Card システム開発ボード　（Cyclone FPGA 搭載）
- ✓ System-SXII システム開発ボード　（Startix2 FPGA 搭載、無償 QuartusII が利用できません）

サンプル回路(無償ダウンロード)　：　Qsys_nios2e_CXUSB2_v11.zip

3. 概要

上記ターゲット・ボードで示すアルテラ FPGA を搭載した Smart-USB Plus 製品向けに、「GPIF-Avalon ブリッジ回路」(*1)を提供しています。 この回路では、QuartusII ツール（以下、Q2）内の Qsys を利用し、HDL コーディングを最小限に抑えながら USB 制御回路システムを設計できます。Qsys ツールを利用して、プロセッサバスの一種である「Avalon バス」を利用した組込システムをグラフィカルに設計することができます。「GPIF-Avalon ブリッジ回路」では、Avalon バス・マスタとして「GPIF_Master」ライブラリを提供し、ユーザが Qsys ツール上で各種ライブラリを配線するだけで機能を実現することができます。 この Avalon バスに無償 CPU コアの Nios2 を接続し、USB を利用した組込システムを構築します。 右図に Avalon バスシステムの例を示します。

(*1) SUA006　『GPIF-AVALON ブリッジ回路』



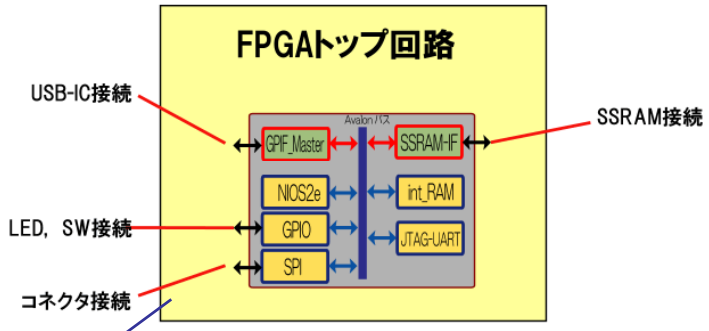
<図 1. GPIF-Master と Nios2e を組み込んだ Avalon バスシステムの例>

【Smart-USB Plus 製品に Nios2e を組み込むメリット】

Qsys ツールで「GPIF_Master」ライブラリを利用することにより、FPGA 設計の手間を省き、ホスト PC 側の USB 制御ソフトウエアと簡単に通信することができます。さらに、Nios2 を組み込むことで CPU 処理したデータを PC に送信したり、ホスト PC からの命令により CPU 処

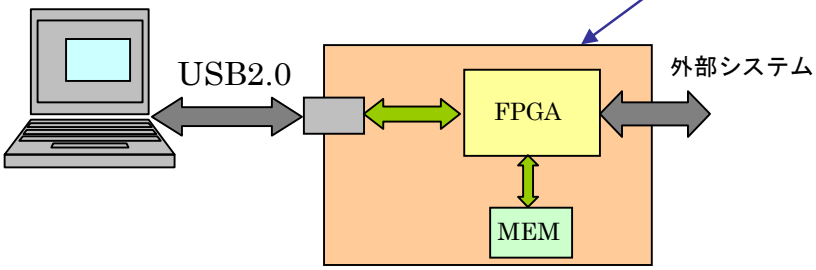
理を制御することができます。
　ボードを USB 接続して運用する場合、Nios2 のインストラクションコードは ROM 化する必要が無く、USB 経由で FPGA をコンフィグする時に FPGA 内蔵 RAM に直接ダウンロードできます。このため、FPGA リソースの低減と効率的なボード運用が可能です。
　また、図 1 に示すように、実現する機能が Qsys ツール内のライブラリだけで完結する場合は、ハードウエアに詳しくないソフトウエア技術者でも FPGA 回路を構築して組込システムの構成ができます。 実際には、Qsys ツールでの作業が完了すると、1 つの Avalon バスシステムブロックができあがります。このブロックと LED や SW 類との接続を行う“FPGA トップ回路の作成”と“FPGA ピンアサイン”作業をするだけでマイコンシステムが開発できます。FPGA ピンアサイン作業はボード購入時に添付する TCL スクリプトを使うと間違いのないピンアサインができます。

図 2 は図 1 で示した Avalon バスシステムのブロックを FPGA トップ回路（黄色背景で示しています）で呼び出し、FPGA のピンアサインを行うイメージです。 Qsys で作成した回路だけなら、FPGA トップ回路では、ピンアサインをするだけです。 この資料で説明するリファレンス回路でも、ユーザが手作業をする部分は、Qsys で作成するブロックとボード上に搭載した 7 セグメント LED 点灯用回路ブロックだけを配線し、FPGA ピンアサインをするだけです。



＜図 2.　FPGA トップ回路のイメージ＞

FPGA に Nios2 と GPIF_Master を組込んだ後は、PC 上の USB 制御ソフトウエアを利用して、FPGA をコンフィグしたり、各種レジスタの設定、メモリ転送ができます。Smart-USB Plus 製品の USB は、ターゲット側なので、ボードから USB を経由してホスト PC に割り込み制御することができません。ボードの状態変化をチェックするには、ホスト PC からポーリングする必要があります。リアルタイム性が必要なシステムなら、Nios2 を利用して FPGA 内で処理し、処理結果を PC に送信するなどの使い方ができ、システム構成の柔軟性が向上します。



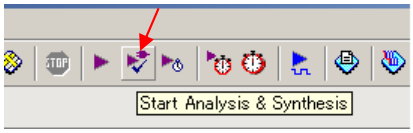
＜図 3. システム構成図＞

4. Nios2 を組み込んだサンプル回路の概要

本資料で説明する FPGA サンプル回路では、USB 通信を行う「GPIF_Master」と「Nios2 コア」を Avalon バス・マスタとし、オンボードの SSRAM アクセス、LED 点灯制御、汎用 IO 制御ができます。 以下、サンプル FPGA プロジェクトを元にして解説します。
サンプル FPGA 回路：　Qsys_nios2e__CXUSB2.zip

4.1　Qsys によるバス設計

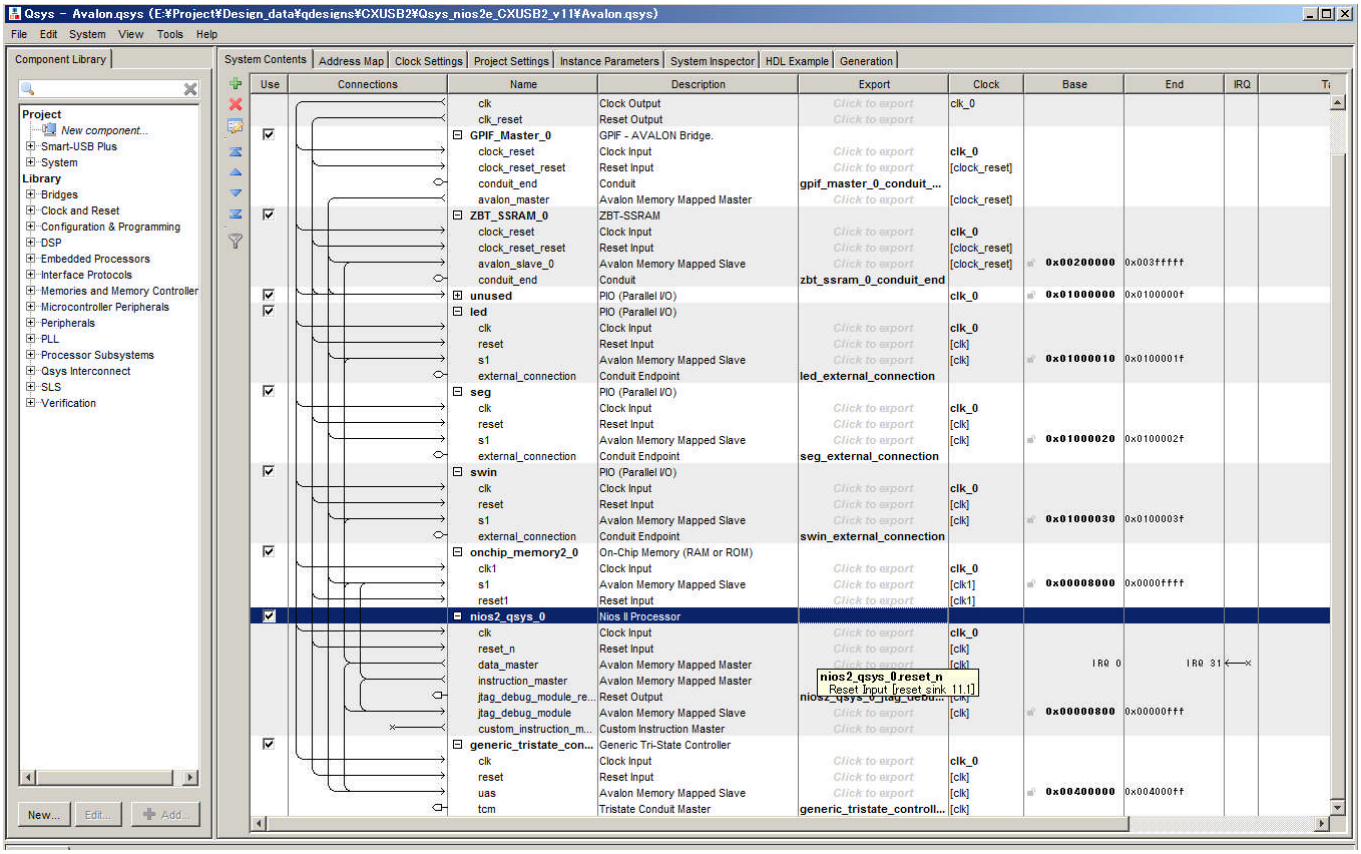
　Qsys_nios2e__CXUSB2.zip ファイルを解凍後プロジェクトを設定し、Q2 ツールバーの「Start Analysis & Synthesis」ボタンをクリックして、Project Navigator 画面にファイル構成を表示します。



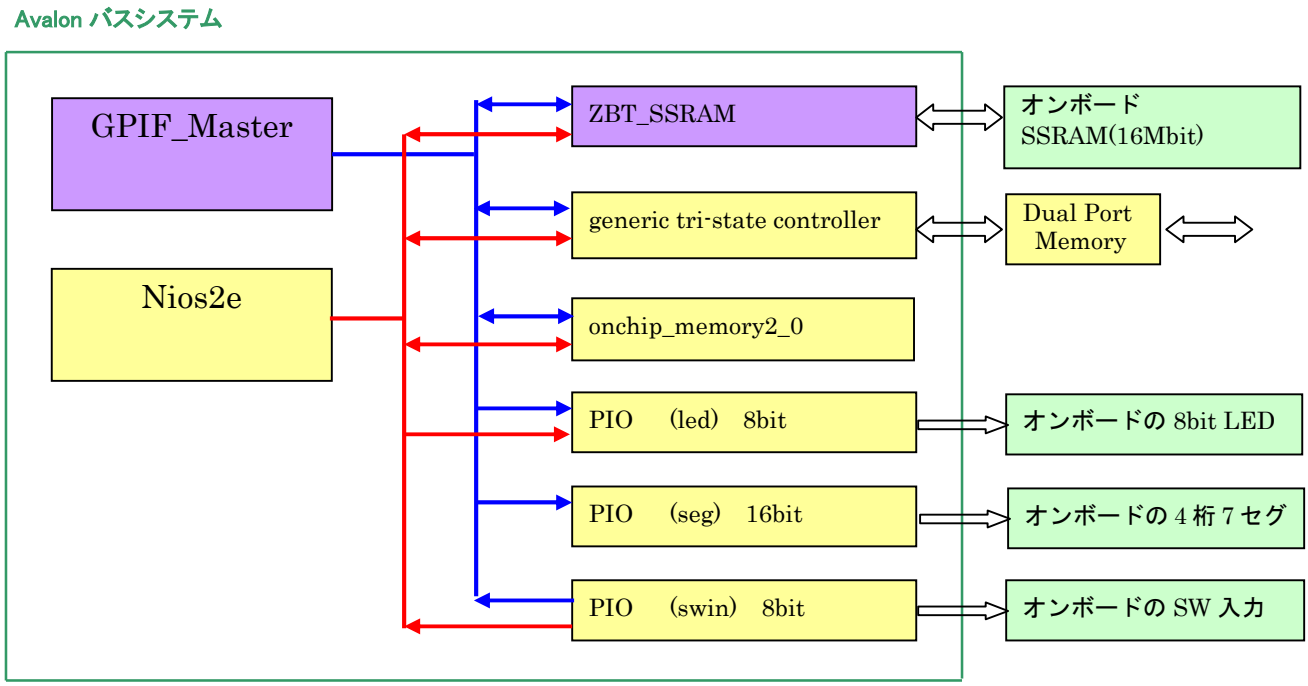
＜図 4. Q2 のツールバー＞

サンプル回路では、Qsys におけるシステム名を「Avalon」と名付けています。Project Navigator 画面の「Avalon」をダブルクリックしても Qsys ツールは起動しません。Q2 ツールバーの「Tools」→「Qsys」を選択し、Qsys プロジェクト Avalon を選択してください。 図 5 で示す画面が得られます。このうち、モジュール名の GPIF_Maste_0 と、ssram1 はそれぞれ当社提供の無償ライブラリです。その他はすべてアルテラ社提供の無償ライブラリです。
この他に追加したいモジュールがあれば、画面左側の“Component Library”から追加します。このとき、Base アドレスが重複しないように注意してください。重複した場合はエラー表示が発生しますので、アドレスが重複したままのシステムができあがることはありません。
ホスト PC から制御したいレジスタを追加する場合は、GPIF_Master のベースアドレス”0x01000000” ＋ “0x00000040”以降を割り当ててください。
サンプル回路で使用しているライブラリの他には、“Interval Timer”や”Vectored Interrupt Controller“、”DMA Controller“、”Avalon-MM Pipeline Bridge”などを利用して、システム構成ができます。

ここで、Nios2 のソフトウエア格納用メモリとして、FPGA の内蔵メモリを使用します。モジュール名”onchip_memory2_0”をダブルクリックすると、メモリ構成 画面が表示されます。 サンプル回路では 32bit 幅、32K バイトのシングルポート RAM を構成しています。



〈図 5. Qsys Avalon バス構成〉



〈図 6.　Avalon バスブロック図〉

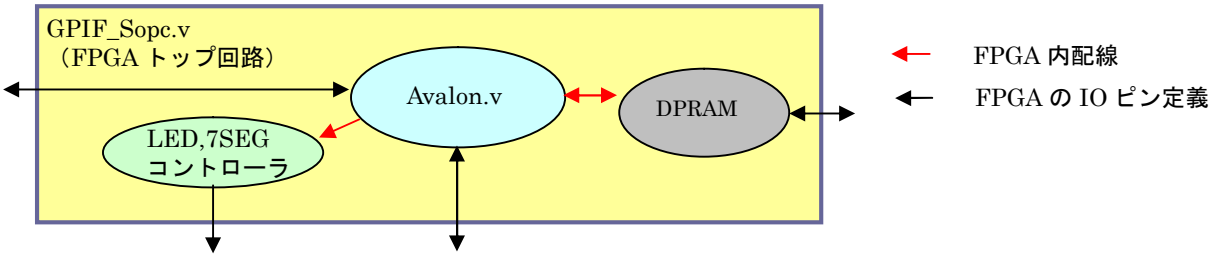
サンプル回路の Avalon バスシステムは図 6 に示すブロック図の通りです。 赤線は Nios2 がバスマスタとなり、Nios2 で制御できるモジュールが接続されています。青線は GPIF_Master がバスマスタとなり、CX-USB2 ボードに接続するホスト PC から制御できるモジュールが接続されています。 緑の線で囲まれている部分が Qsys で作成した Avalon バスシステムです。 Nios2 と GPIF_Master 両方から制御できる「generic tri-state controller」には、Avalon バスシステムの外側に、デュアルポートメモリ（8Kbit）を用意し、Avalon バスシステム外の回路とデータの受け渡しができる様な構成になっています。

サンプル回路の構成に変更がなければ、Qsys「Generation」タブの「Generate」ボタンをクリックします。Generate が正常終了すると、PC の作業フォルダ内に “Avalon.sopcinfo” ファイルが生成されます。 Nios2 ソフトウエア作成時、このファイルを指定して Nios2 プロジェクトを生成します。このため、xxx.sopcinfo ファイルが更新された場合、Nios2 ソフトウエア・プロジェクトも再ビルドする必要があります。

4.2　QuartusII で FPGA コンフィグデータを生成

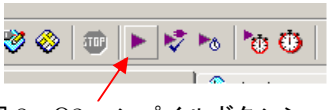
Qsys での作業が完了したら、Avalon バスシステムのモジュールをトップ回路で呼び出し、Avalon バスシステム以外の回路と接続したり、FPGA のピン定義を行います。

サンプル回路は VerilogHDL 記述です。ピン定義済みなので、ピンアサイン作業は不要です。Avalon バスシステムのモジュールは Qsys により“Avalon.v”ファイルとして生成されています。この Avalon.v 回路をトップ回路でインスタンシエイトして最終的な FPGA 回路とします。インスタンシエイトの際には　Qsys 画面の「HDL Example」タブから簡単にコピーできます。このとき、Q2 プロジェクトに Qsys で作成した Avalon.qip を追加してください。このファイルは￥Avalon¥sysnthesis フォルダ内にあります。サンプル回路の構成は以下のようになります。



〈図 7. Q2 でのトップ回路生成〉

FPGA のピン・アサインについては、作業フォルダ内にある“PIN_SET.tcl”を利用することで、ピンアサインの間違いを防ぐことができます。
サンプル FPGA 回路で変更がなければこのままコンパイルを行い、FPGA コンフィグ・ファイルの“GPIF_Sopc.rbf”を生成します。



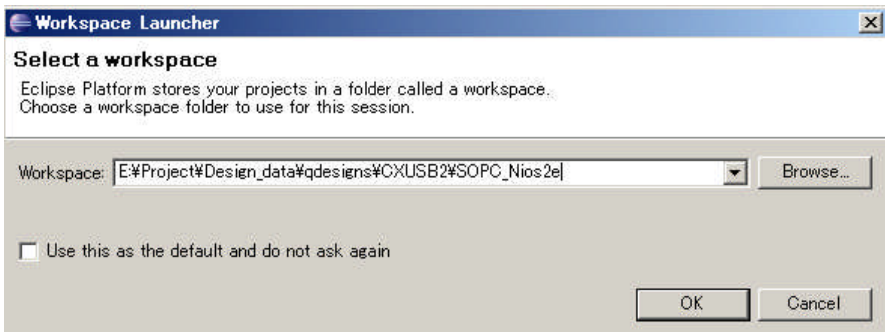
＜図 8.　Q2 コンパイルボタン＞

ここまでの作業で、FPGA 回路は完成し、Nios2 ソフトウエア作成の準備ができました。

5.　Nios2 ソフトウエア・プロジェクトの生成

5.1　Nios-EDS ツールの起動

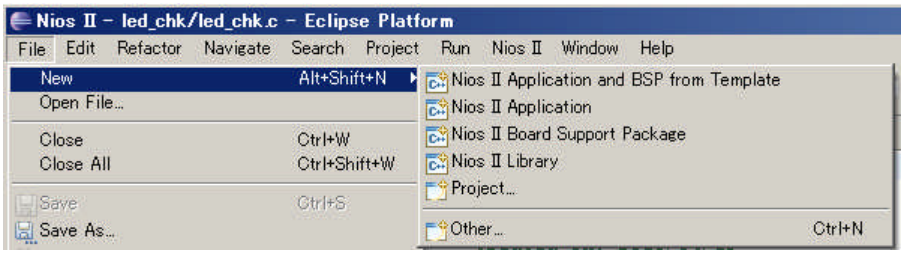
ソフトウエア・プロジェクトの作成には Nios2-EDS を起動します。コマンドラインツールも用意されていますが、ここでは GUI 対応のツールを使用します。ツールを起動するには二通りあります。一つは Windows のスタートメニューから“NiosII 11.1 Software Build Tools for Eclipse”を選択して起動します。もう一つは図 5 で示した Qsys のツールバー「Tools」のプルダウンメニューから「Nios II Software Build Tools for Eclipse」を選択して起動することもできます。どちらでも起動時に“Workspace Launcher”がポップアップし、作業ディレクトリを指定することができます。ここでは、Q2 の作業ディレクトリを指定します。



＜図 9.　作業ディレクトリの指定＞

5.2　Nios2 アプリケーションの作成

起動した Eclipse Platform（以下、IDE）のツールバー　「File」から「NiosII Board Support Package」を選択します。このメニューにより、FPGA に構築した Avalon バス構成を呼び出し、ソフトウエアビルドを実行できるようにします。
次に、IDE ツールバーの「File」から「NiosII　Application」を選択して、C ソースコードを作成します。

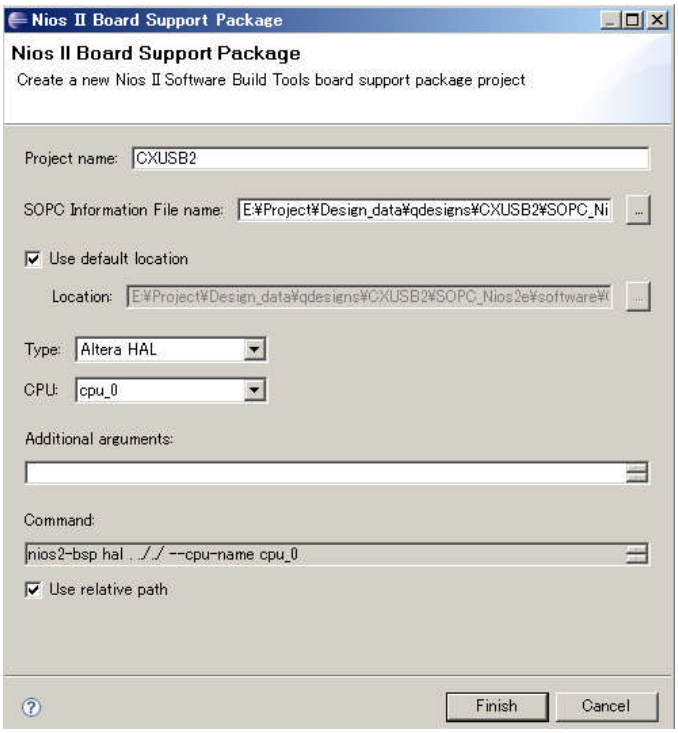


<図 10.　Eclipse メニュー>

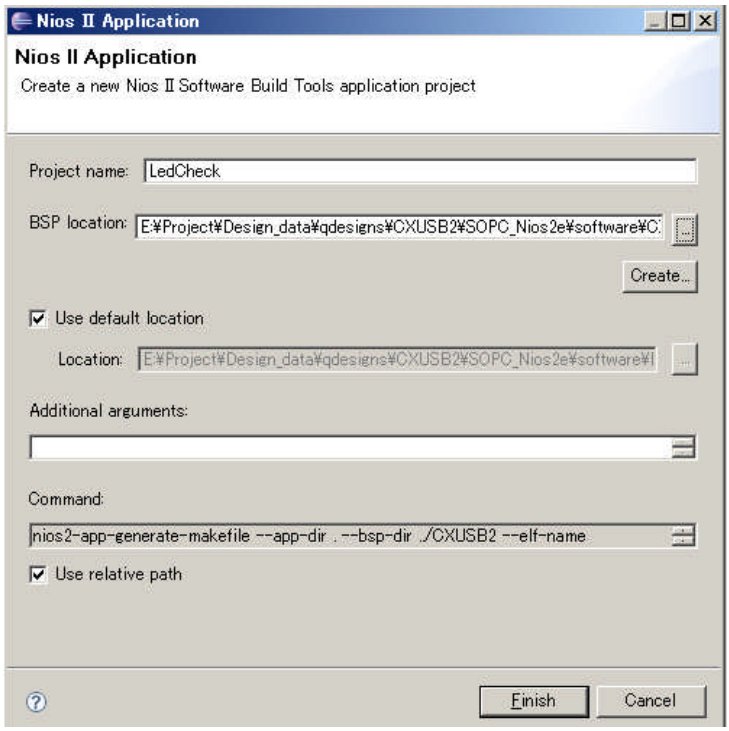
図 10 で示すメニューの中で、「NiosII Application and BSP from Template」を選択すると、「Hello world」などのサンプルコードが利用できます。

5.3　NiosII Board Support Package（BSP）とアプリケーションの作成

図 10 のメニューから「NiosII Board Support Package」選択すると、図 11 で示すメニュー画面が得られます。
“Project name”を　CXUSB2　とし、”SOPC Information File name”には、Qsys で generate して生成した　ファイル“Avalon.sopcinfo”を選択して指定します。最後に「Finish」ボタンをクリックすると、BSP を生成します。このとき、Q2 バージョンと Nios2 ソフト開発ツールのバージョンが一致していないとエラーが発生します。



<図 11.　BSP 生成メニュー>

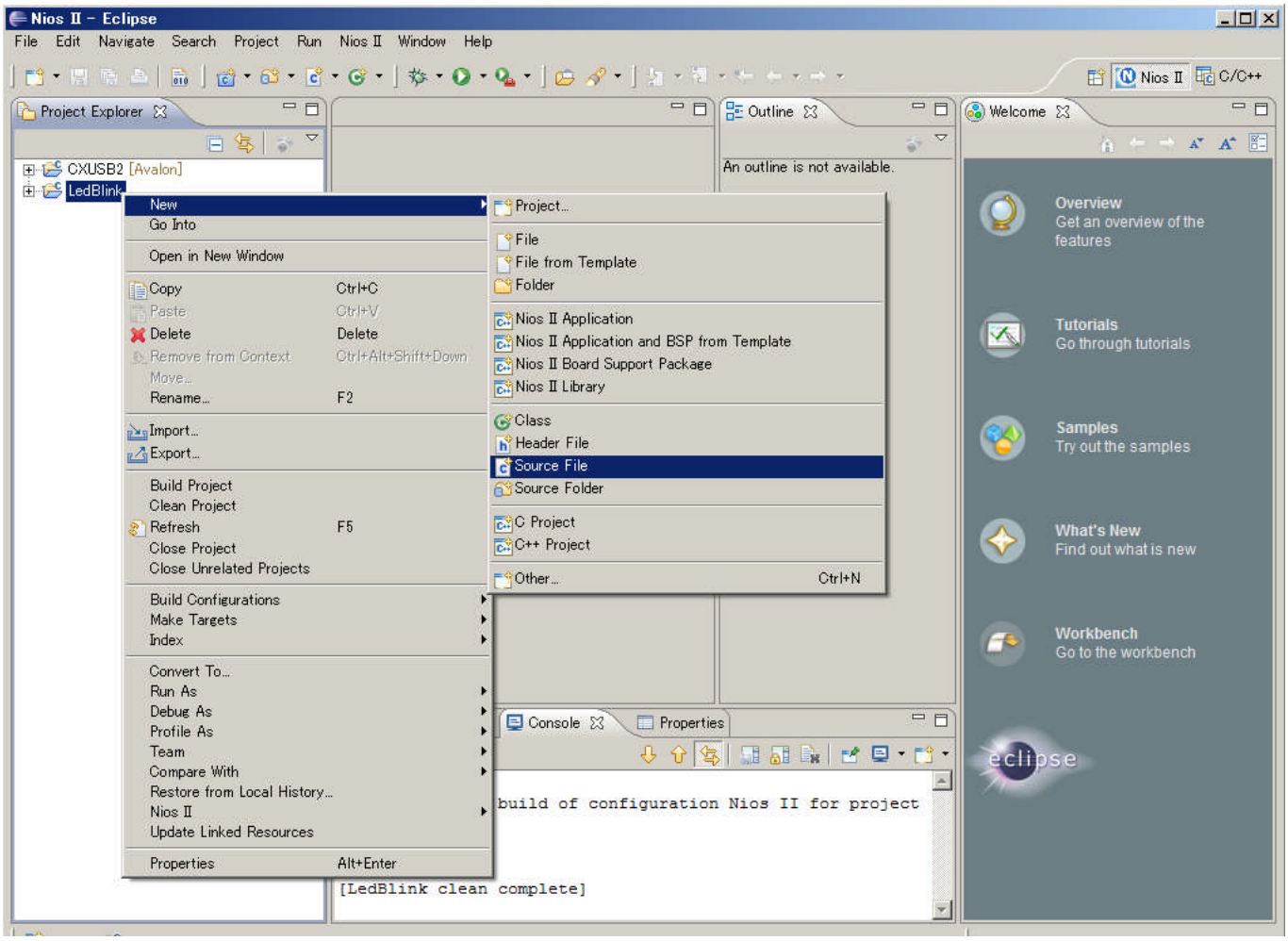


<図 12.　アプリケーション生成メニュー>

次に、図 10 で示すメニューから “NiosII Application”を選択し　図 12 のアプリケーション生成メニューを得ます。
ここで、”Project name”は　BSP のプロジェクト名と異なる必要があります。ここでは“LedBlink”としました。”BSP Location”欄には、先ほど生成した”CXUSB2”を指定してください。ここまでの設定で「Finish」ボタンをクリックし、Nios2 アプリケーションの作成準備が整いました。

5.4　ソースファイルの作成

Eclipse Platform の “Project Explorer” に表示されている Nios2 アプリケーション・プロジェクトの “LedBlink”を選択し、マウスの右ボタンをクリックして、”New” → ”Source File” を選択します。
“Source File”を指定する欄に、”LedBlink.c”　というファイル名を付け、「Finish」ボタンをクリックすると、ソースコードを記述できるエディタが起動して、ソースコードの記述ができるようになります。拡張子を忘れずに記述してください。



＜図 13.　LedBlink.c を作成する画面>

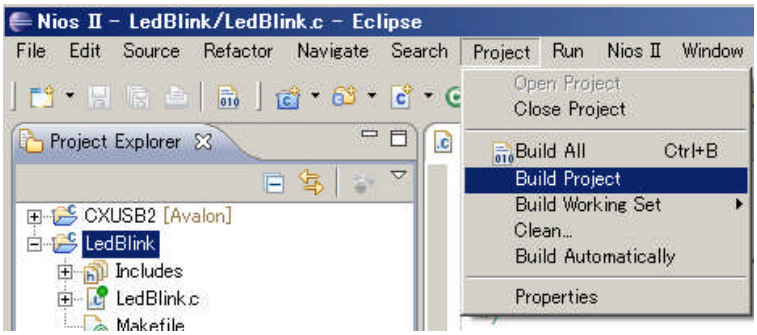
5.5　ソースコード

以下に示すソースコードは、CX-USB2 ボードの 8bit　LED を制御します。1bit ずつ左から右へ、右から左へ行ったり来たりする内容です。このサンプルコードは、Nios2 ver7.2 で提供する“hello_led.c”を元にしています。　このサンプルコードに、CX-USB2 ボードの汎用タクトスイッチ（SW5）を押したときに、点滅動作が停止する様に変更しています。
このソースコードを先ほど作成した“LedBlink.c”に記述してソースファイルを作成します。

```
#include "system.h"
#include "io.h"
#include "altera_avalon_pio_regs.h"
#include "alt_types.h"
#define led_stop 0x80

int alt_main (void)
        {
          alt_u8 led = 0x2;
          alt_u8 dir = 0;
          volatile int i;

          /*
           * Infinitely shift a variable with one bit set back and forth, and write
           * it to the LED PIO.   Software loop provides delay element.
           */
          while (! (IORD(SWIN_BASE,0) & led_stop))
          {
            if (led & 0x81)
            {
              dir = (dir ^ 0x1);
            }

            if (dir)
            {
              led = led >> 1;
            }
            else
            {
              led = led << 1;
            }
            IOWR(LED_BASE,0x0, led);

            /*
            * The delay element in this design has been written as a while loop
            * to avoid confusing the software debugger.   A tight, one line software
            * delay loop such as:
            *     for(i=0; i<200000; i++);
            * can cause problems when it is stepped through using a software debugger.
            * The while loop below produces the same behavior as the for loop shown
            * above, but without causing potential debugger problems.
            */

            i = 0;
            while (i<200000)
              i++;
          }
}
```

5.6 ビルド

ソースコード作成が完了したら、プロジェクトをビルドします。ビルドするプロジェクトを選択してから、ツールバーの「Project」→「Build Project」を選択します。　ソースコードの内容にエラーがなければ、Eclipse Platform 画面下部の「Console」タブに、「LedBlink build complete」と表示されます。　エラーがあれば、「Problems」タブにエラー表示が発生します。
この時点で Nios2 が動作するための実行形式ファイル（LedBlink.elf）が生成されています。



<図 14.　ビルドメニュー>

6.　ボード（CX-USB2）の運用

Nios2 ソフトは、FPGA をコンフィグ後、USB Blaster ケーブル（JTAG 経由）を利用してダウンロードするか、RefApp7 のメモリ操作機能を利用して、FPGA 内のオンチップメモリに USB 経由でダウンロードできます。USB Blaster ケーブルを利用すれば Nios2 のデバッグ機能を利用することができます。　デバッグ機能を利用しなければ、USB Blaster ケーブルを利用するより、RefApp7 から USB 経由でダウンロードする方が高速で便利です。

6.1　FPGA のコンフィグ

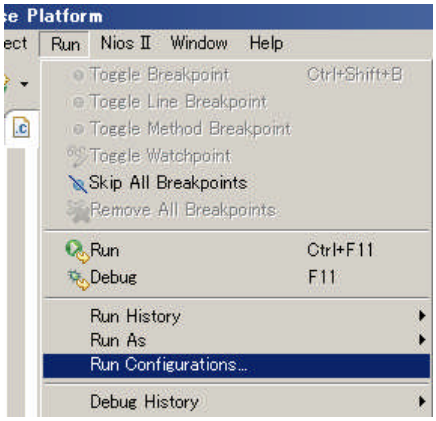
ターゲットボードの CX-USB2 ボードに添付する制御アプリケーション“RefApp7.exe”を起動し、「USB コンフィグ」タブから、Q2 でコンパイルして生成した“GPIF_Sopc.rbf”を選択して「Configure」ボタンをクリックし、FPGA コンフィグしてください。Smart-USB Plus 製品ならすべて共通の作業です。

コンフィグの正常終了後、FPGA には　図 5 で示す Avalon バスシステムが構築されていますが、Nios2 プログラムが無いので、Nios2 は動作していません。このサンプル回路では、制御アプリの“RefApp7.exe”から、SSRAM のメモリ操作や 7 セグ表示を行うレジスタ制御などができる状態です。

＜図 15.　制御アプリ“RefApp7.exe”による FPGA の USB コンフィグ＞

6.2　Nios2 プログラムのダウンロード

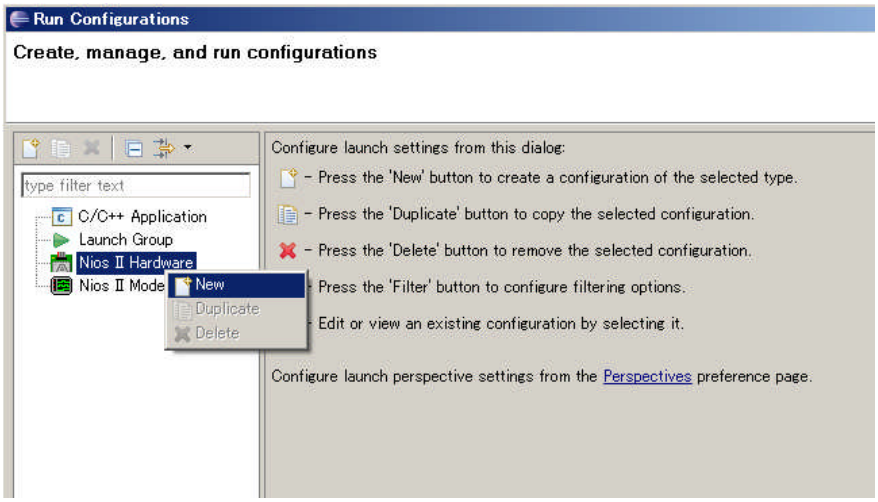
アルテラ社製のダウンロードケーブル　USB Blaster を利用して、図 6 で示した Avalon バスシステムのオンチップメモリ（32KB）に” LedBlink.elf”をダウンロードし、Nios2　CPU を実際に動作させます。　ダウンロードするためには　あらかじめ FPGA をコンフィグし、Eclipse Platform 側で USB Blaster を認識しなければなりません。
ターゲットボードの CX-USB2 ボードには、JTAG コネクタに USB Blaster ダウンロードケーブルを接続し、別の USB ケーブルを CX-USB2 に接続してください。



<図 16.　Run メニューのコンフィギュレーション>

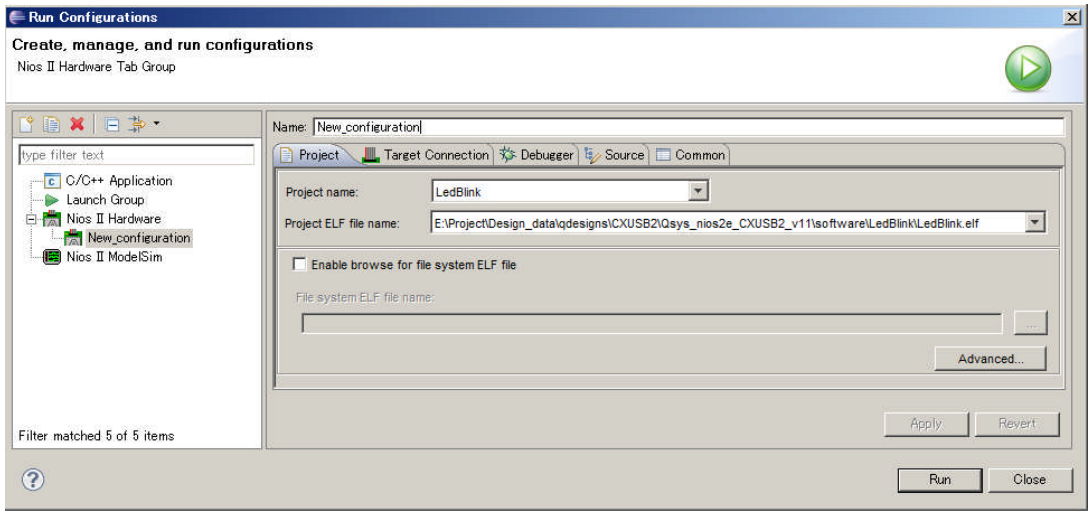
6.2.1　USB Blaster を利用する

FPGA コンフィグが完了したら、Eclipse Platform のツールバー「Run」メニュー→「Run」を選択するのですが、Eclipse Platform に USB Blaster ダウンロードケーブルを認識させなければなりません。このために、「Run」メニューから「Run　Configurations...」を選択して、ターゲットボード（CX-USB2）にダウンロードしたい Nios2 プロジェクトの指定とダウンロードケーブルを認識させます。

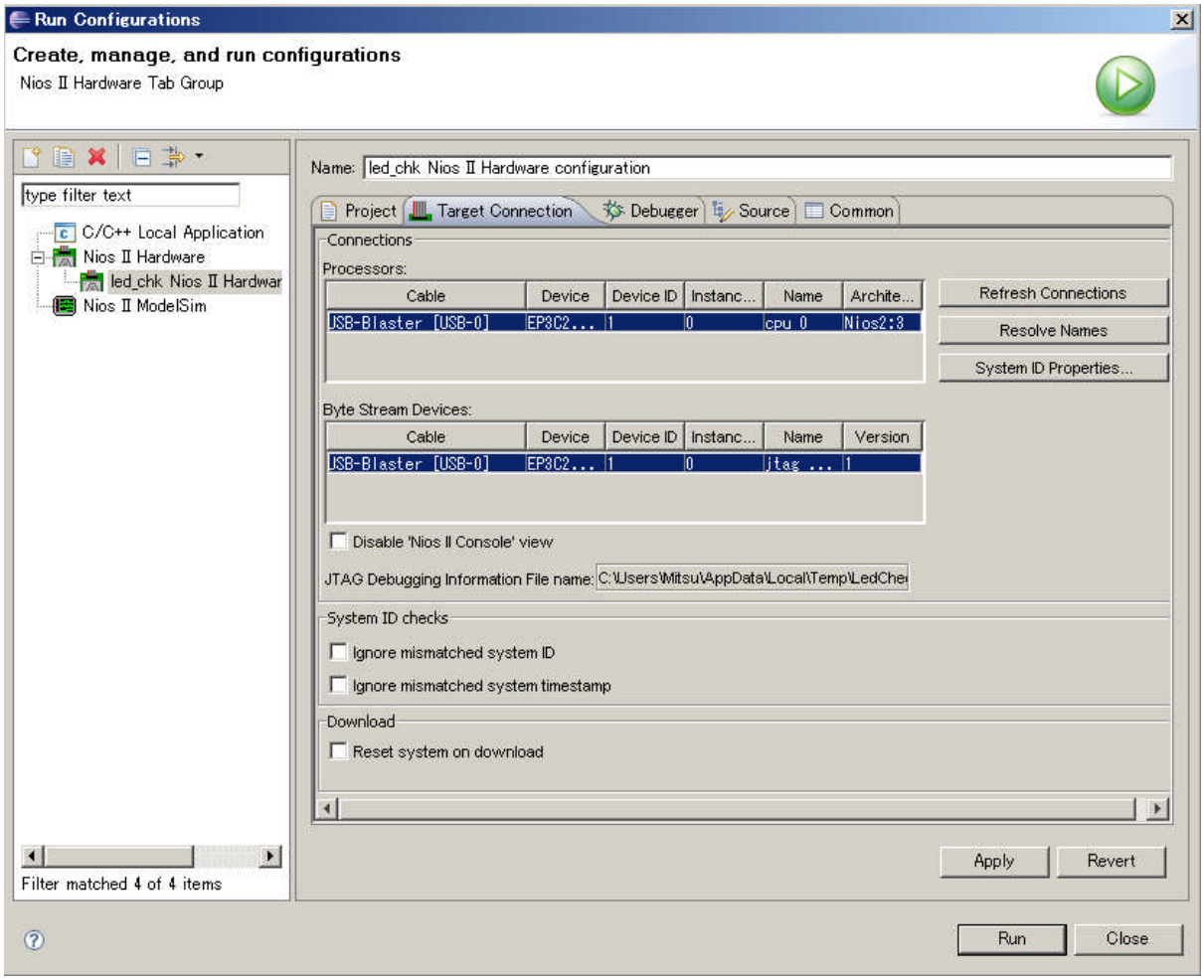


<図 17.　USB Blaster を認識させる準備>

図 18 では、”Project”タブで FPGA にダウンロードしたい Nios2 プロジェクトを指定します。



<図 18.　ダウンロードする Nios2 プロジェクトの設定>



<図 19.　ダウンロードケーブルの選択>

図 19 で示すメニュー画面では、「Refresh Connections」ボタンをクリックし、ボードに接続しているダウンロードケーブル（ここでは、USB Blaster）を認識します。　System ID やタイムスタンプを利用していなければ、「System ID Check」欄にチェックを入れて無視してください。
ここまで完了したら、最後に画面右下の「Run」ボタンをクリックします。これで Nios2 のプログラムコードが FPGA 内のオンチップメモリ（onchip_memory2_0 モジュール）にダウンロードされ、Nios2 が動作するので、ターゲットボード上の LED が点灯します。

6.2.2　RefApp7 のメモリ操作を利用する

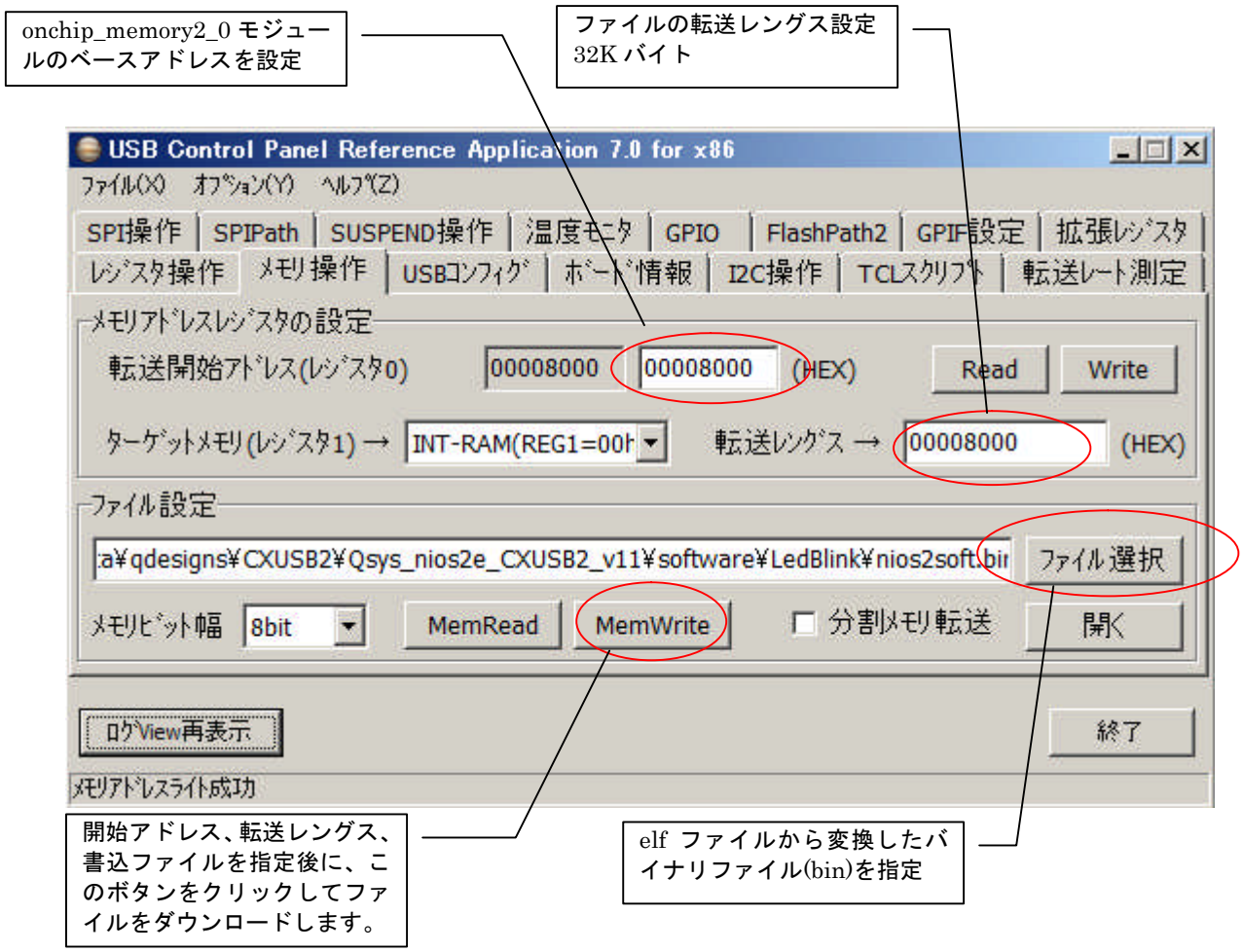
USB Blaster ケーブルを所有していない場合や、JTAG デバッグが完了している場合など、RefApp7 リファレンス制御アプリケーションの「メモリ操作」画面から、Nios2 ソフト格納メモリの”onchip_memory2_0”モジュールへ直接ダウンロードできます。この場合、P.9 で示したビルド後に得られる elf ファイル（LedBlink.elf）をバイナリファイル(.bin)に変換してから操作してください。
elf ファイルから bin ファイルへの変換は、NiosII コマンドシェル*を起動して、以下に示すコマンドを利用します。

nios2-elf-objcopy␣-O␣binary␣LedBlink.elf␣nios2soft.bin

この例では、LedBlink.elf ファイルを FPGA のオンチップメモリに直接書き込める nios2soft.bin ファイルに変換しています。

*）Nios2 コマンドシェルの起動は、Eclipsae Platform の“Project Explorer”画面に表示するプロジェクトを選択し、右クリック後“NiosII”→“NiosII Command Shell…”　を選択してください。

生成したバイナリファイルは、RefApp7 の「USB コンフィグ」タブで FPGA をコンフィグ後、「メモリ操作」タブからオンチップメモリにダウンロードできます。

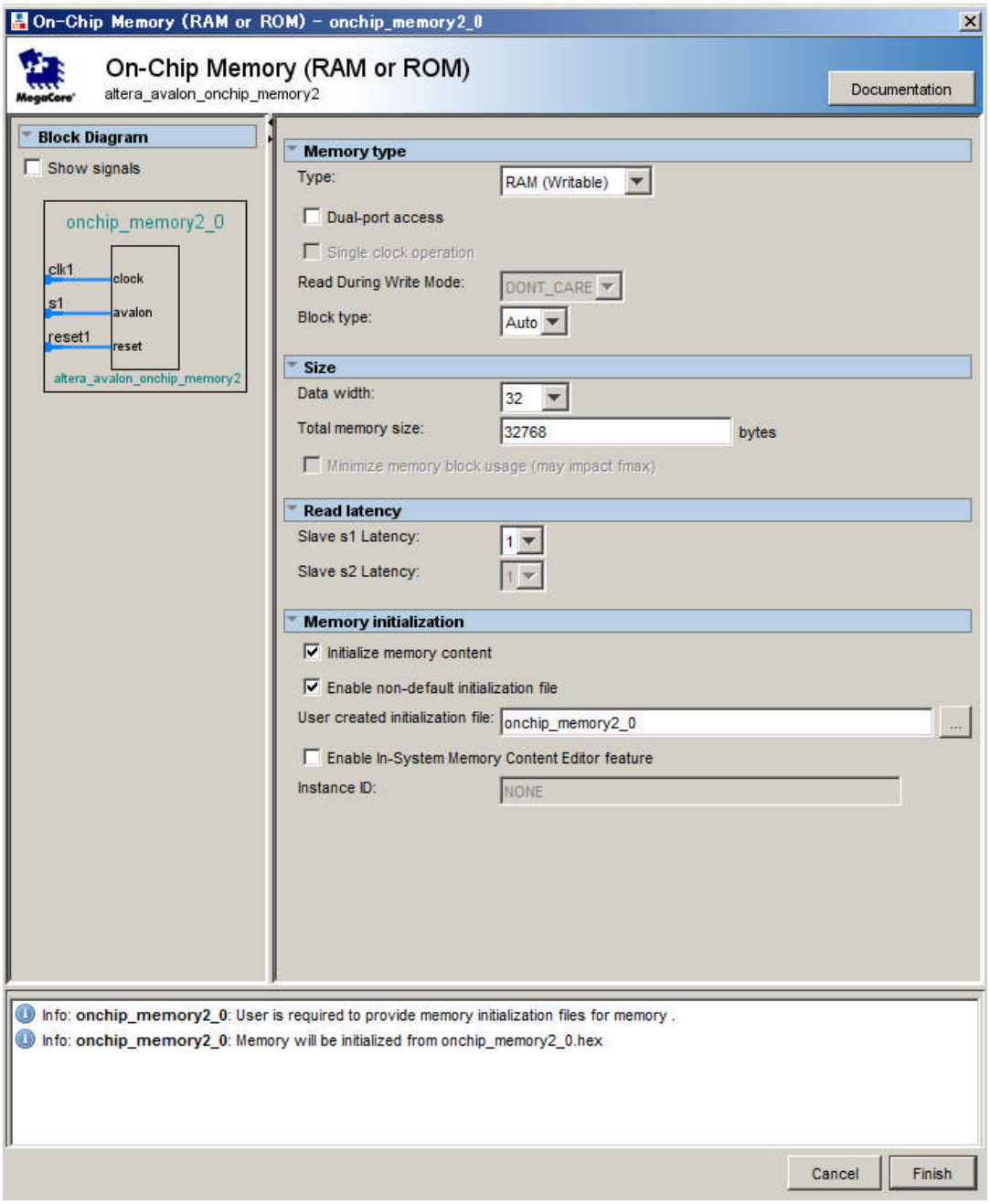


<図 20.　RefApp7 のメモリ操作タブ>

作成した Nios2 ソフトを FPGA 内の onchip_memory2_0 にダウンロードするには、P.4　図 6　Avalon バスブロック図で示した様に、GPIF_Master モジュールと、Nios2e モジュールの両方からアクセスできる様に設定してください。この機能を利用しない場合は、GPIF_Master モジュールから Nios2 ソフトを格納する onchip_memory2_0 モジュールにアクセスしない様に設定してください。

6.3　Nios2 プログラムの ROM 化

Nios2 プログラムの開発が完了したら、一般的にはプログラムコードを ROM 化します。　この Q2 プロジェクトでは図 15 で示したように FPGA のコンフィグを USB 経由で行うので、ボード上のコンフィグ ROM には書き込みません。その代わり、FPGA 内蔵メモリの onchip_memory2_0 モジュールに、Nios2 プログラムをメモリ初期化ファイルとして設定し、FPGA コンフィグファイルの“GPIF_Sopc.rbf”に埋め込みます。これにより、USB コンフィグ後に LED 点灯が始まります。



<図 20.　オンチップメモリ構成画面>

上図は、Qsys で、”onchip_memory2_0”モジュールをダブルクリックして得られる設定画面です。メモリ容量の設定とメモリの初期設定ファイルを設定します。
この例では、“onchip_memory2_0.hex”というファイルを指定しています。図.20 で示すファイル名入力欄には拡張子の”.hex”を削除してください。ここで拡張子の”.hex”まで記入すると、Q2 で正しくファイルのパスが設定されません。
このファイルを Q2 プロジェクトファイル（.qpf）のあるフォルダに配置し、Q2 コンパイルを行います。　サンプル FPGA プ

ロジェクトでは、すでに左図のようにメモリ初期化ファイルを設定していますので、“onchip_memory2_0.hex” を生成した後、Q2 コンパイルを実行してください。
hex ファイルの生成は、次項を参照してください。

6.3.1　HEX ファイルの生成

Nios2 コマンドシェルを起動して、HEX ファイルを生成します。
elf2hex コマンドを使用し、ビルドして生成した.elf ファイル（サンプルでは、LedBlink.elf）から.hex ファイル（onchip_memory2_0.hex）に変換します。

elf2hex␣--base=0x8000␣--end=0xffff␣--width=32␣--input=LedBlink.elf␣--output=xxx.hex

(注意：␣はスペースです)

この例では、xxx.hex　に　onchip_memory2_0.hex が入ります。

6.4 ボード制御

表 1 には、USB 制御アプリ “RefApp7.exe”から制御できるレジスター覧を示します。このうち、Nios2 からも制御できるレジスタには◎印をつけています。P.3　図 6　Avalon バスブロック図も参照してください。

レジスタ No.	bit 幅	モジュール名	内容	備考
4　◎	8	led	LED 点灯レジスタ	
8	16	seg	7 セグ表示レジスタ	
12　◎	8	swin	タクトスイッチ入力レジスタ	

RefApp7.exe の「メモリ操作」タブでは、CX-USB2 ボード上の SSRAM と FPGA 内に用意するデュアルポート RAM にアクセスできます。

ベースアドレス	bit 幅	モジュール名	内容
0x00008000 ～ 0x0000FFFF	32	onchip_memory2_0	Nios2 ソフト格納用メモリ（RAM）
0x00200000 ～ 0x003FFFF	32	ZBT_SSRAM_0	FPGA に接続する SSRAM の制御
0x00400000 ～ 0x004000FF	32	Generic_tristate_controller_0	FPGA 内に用意したデュアルポート RAM の制御

ZBT_SSRAM への RD/WR は、RefApp7 の転送開始アドレスに　0x200000　を設定し、転送レングスには　0x200000（2M バイト）を設定します。同様に　デュアルポートへのアクセスには、転送開始アドレス= 0x00400000、転送レングス=0x400（1K バイト）を設定してください。

関連資料

SUA006　『GPIF-AVALON ブリッジ回路』

Nios2 を利用せず、Avalon バスマスタの GPIF_Master ライブラリを利用するアプリケーションノートです。
Qsys の使い方や GPIF_Master の使用方法については、この資料を参照してください。

【資料製作、お問い合わせ】
有限会社プライムシステムズ
オフィシャルサイト　：　http://www.prime-sys.co.jp/
技術サポートサイト　：　http://www.smartusb.info/
E-mail：　info@prime-sys.co.jp

ver1.0　初版
ver2.0　SOPCBuilder から Qsys へ移行