Impulse C 用 RS232C ハードウェアラッパーキット

Ver.1.1 2013/04/05a
荒川 尚久、泉 知論（立命館大学）
t-izumi@se.ritsumei.ac.jp

本キットは ICFPT2013 Design Competition やリコンフ研 FPGA 設計コンテスト 2013「Blokus」において「Impulse C/Co Developer」を用いた高位設計を行うための設計資産です。ゲーム対戦アルゴリズムは高位合成を用いて設計しても、ＦＰＧＡと外界をつなぐインターフェースなどは Verilog などのハードウェア記述言語で設計しなければなりません。このキットは最低限必要なインターフェース部分を提供します。これを利用すれば、ゲーム対戦アルゴリズムの設計者は、ハードウェア記述言語によるコーディングをしなくてすみます。

本キットは、Impulse C の通信チャネルの一つである「stream」と「RS232C(UART)」とのプロトコル変換モジュール、そして設定済みのプロジェクトファイルで構成されています。図１に構成を図示します。

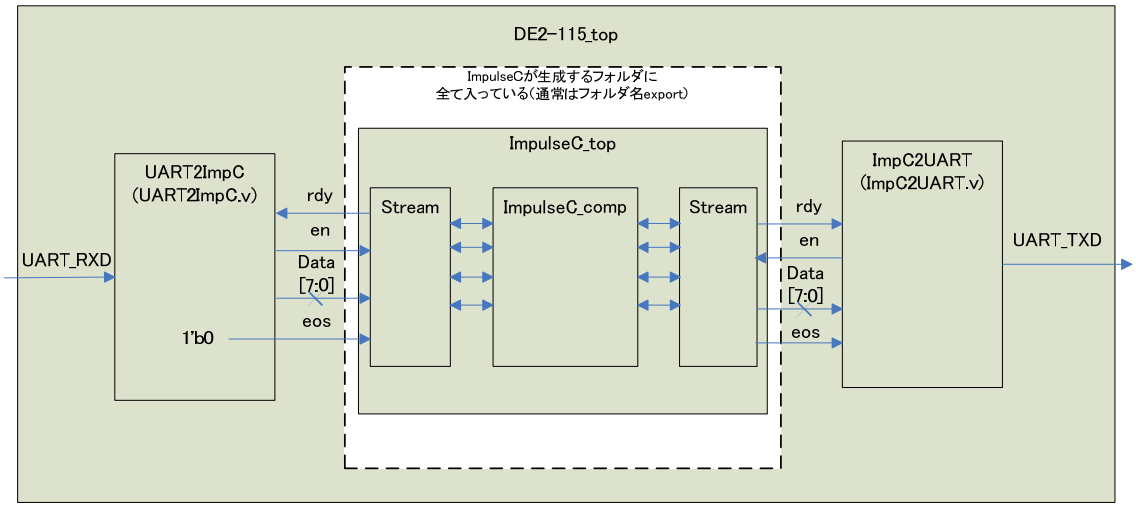


Figure 1　ハードウェアラッパキットの構成

プロトコル変換モジュールは UART 受信(RXD)から stream 入力(data,rdy,en,eos)に変換する UART2ImpC と stream 出力(data,rdy,en,eos)から UART 送信(TXD)に変換する ImpC2UART からなります。これらは、対象ＦＰＧＡやボードを問わず、使用することができます。（クロック速度にあわせて定数を変更する必要があります。）

ＦＰＧＡボード固有のトップ記述とプロジェクトファイル群はリコンフ研から貸し出しされている terasIC 社 DE2-115 ボード（ALTERA Cyclone IV 搭載）向けのものを用意し

ています。それ以外のボードで使用したい場合は、トップ記述およびピン割り当てなどの設定を自身で用意する必要があります。(今後 DE2-115 以外のボードにも対応するかも知れません)

以下では、DE2-115 ボードを対象に ImpulseC プロジェクトの作成方法と「Impulse C 用 RS232C ハードウェアラッパーキット」の使用方法と実装方法を記します。

1. 準備

本キットを使用する際には、次の機材やソフトウェアツールを準備してください。

- 機材
  - terasIC DE2-115 (コンテスト用ボードを貸し出ししています)
  - USB-RS232C 変換ケーブル(サンワサプライ USB-CRSV9 等、必要に応じてボード側のネジを)
- ソフトウェアツール
  - Impulse CoDeveloper(コンテスト用ライセンスを貸し出ししています)
  - ALTERA QuartusII(フリーの Web 版で実行可能です)
  - ターミナルソフト(TeraTerm 等のフリーのものが使えます)

また、Impulse と Quartus の作業フォルダを適当な場所に用意してください。キットの中の Impulse_project フォルダと Quartus_project フォルダをコピーしておきます。以下では、次のフォルダを想定しています。

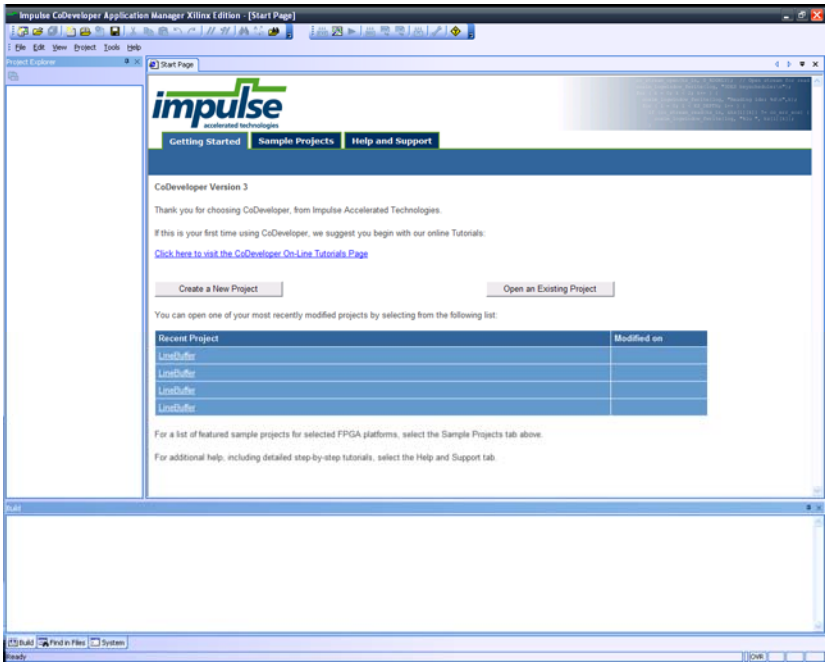
Impulse 作業フォルダ

C:¥Users¥izumi¥Desktop¥Blokus¥Impulse_project¥

Quartus 作業フォルダ

C:¥Users¥izumi¥Desktop¥Blokus¥Quartus_project¥

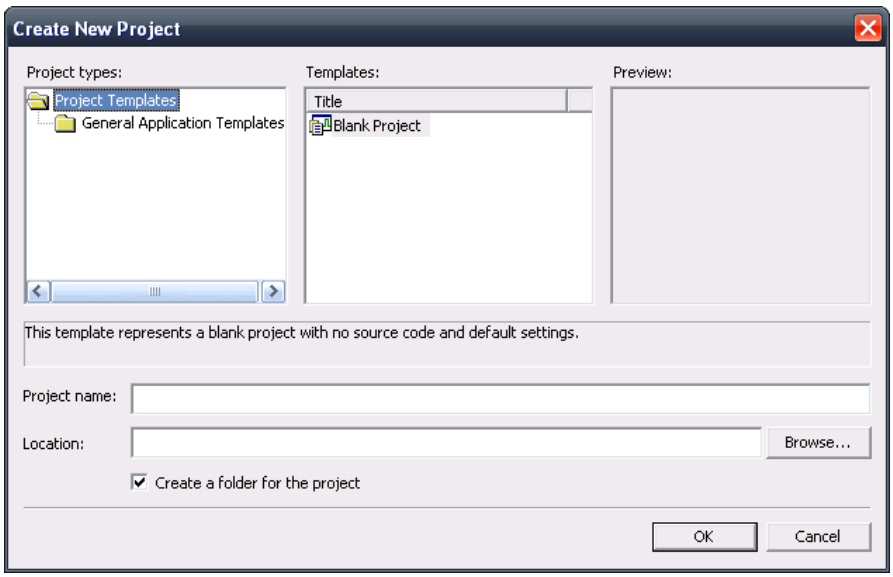
2. Impulse C/Co Developer を起動



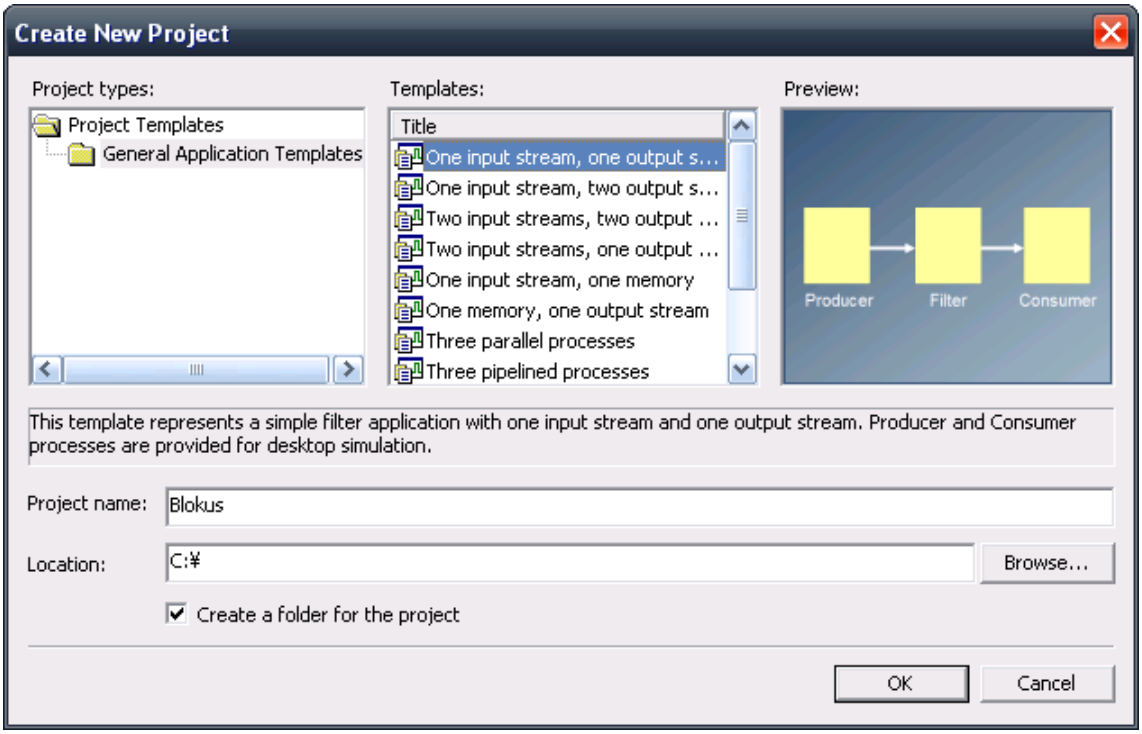
3. 新規プロジェクトの作成

- [File]タブ->[NewProjectWizard]をクリック。

「Create New Project」ウィザードが立ち上がる。

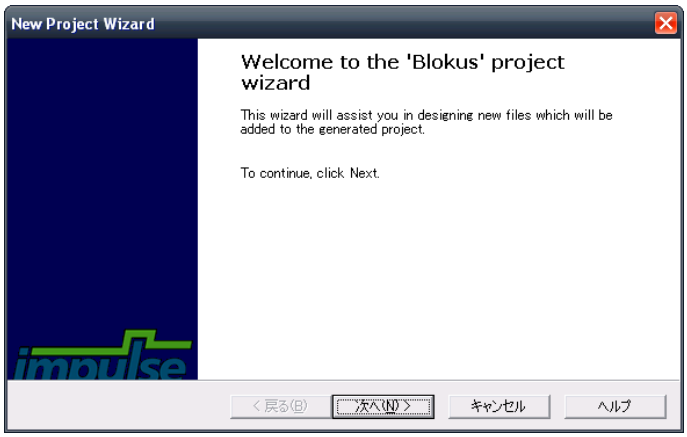


- 「Project types」の[Project Templates]->[General Application Templates]を選択。
- 「Templates」の[One input stream, one output stream]を選択。
- 「Project name」は「Blokus」に。
- 「Location」を選択。例：C:¥Users¥izumi¥Desktop¥Blokus¥Impulse_project
- [OK]をクリック。

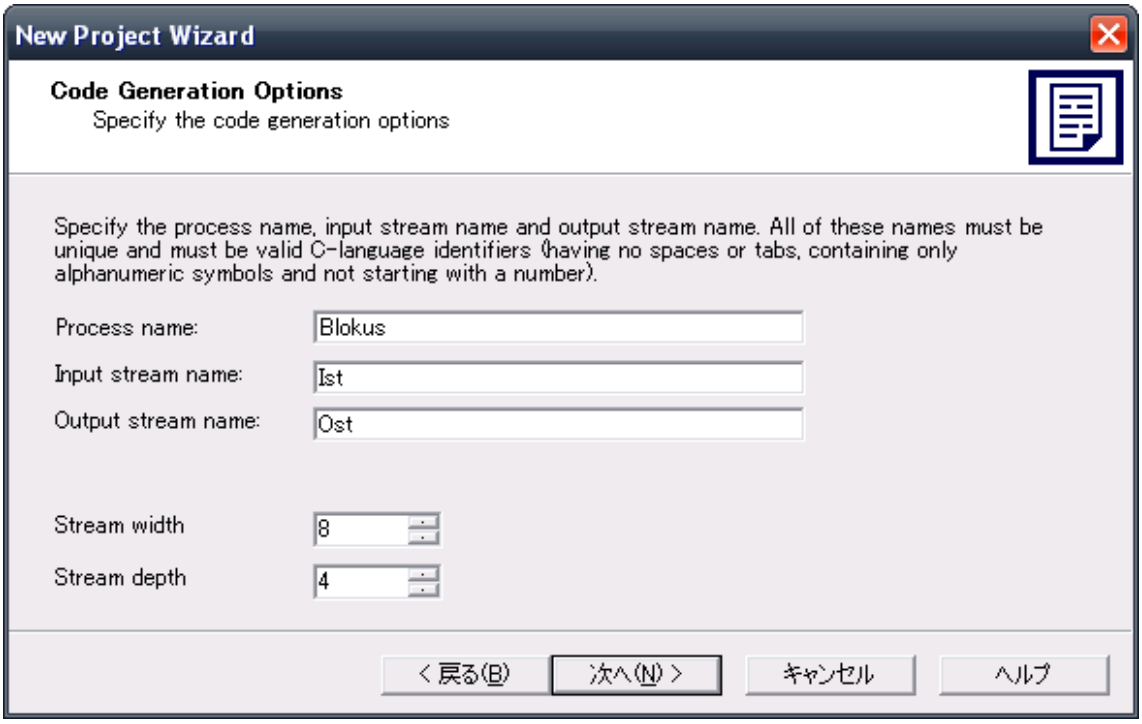


※プロジェクト名について…プロジェクト名は、Impulse C が生成する「HDL ファイル名」と「HDL に記述される top モジュール名」に対応しています。提供する QuartusII または ISE のプロジェクトファイルとの互換性がなくなるので、プロジェクト名を変更する際は適宜対応する箇所の変更が必要です。

- [次へ]をクリック。



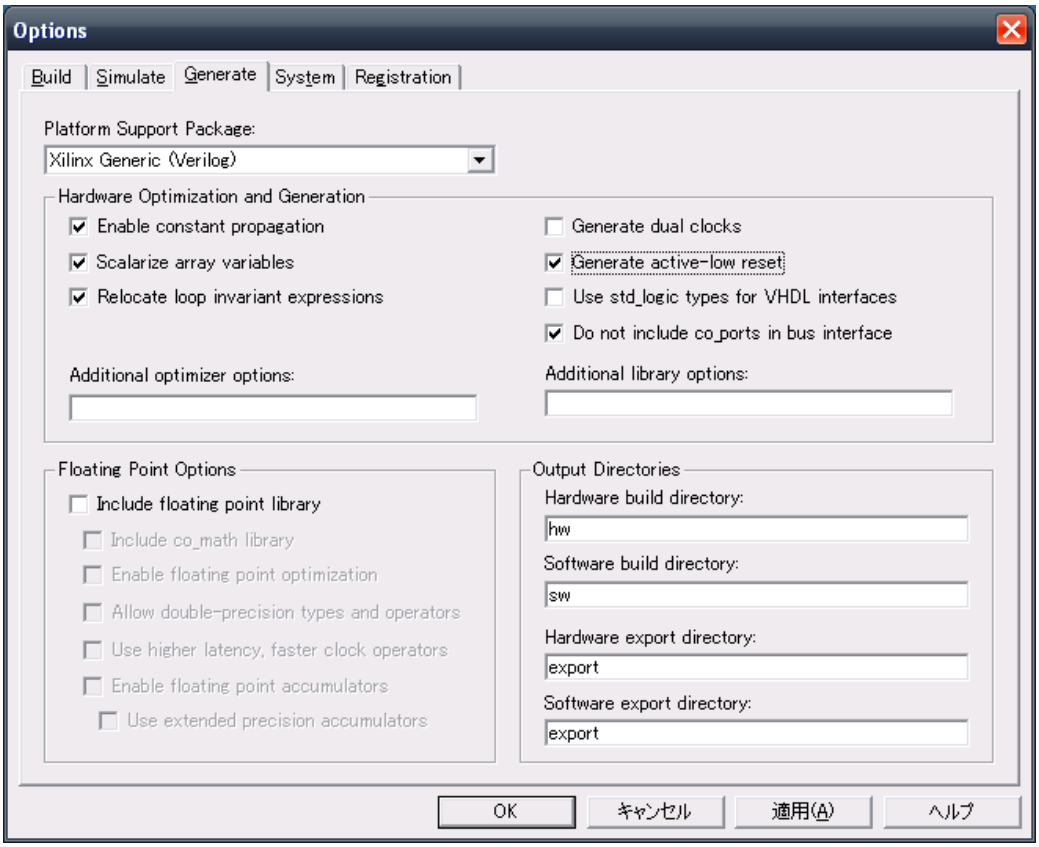
- 「CodeGeneration Options」で以下のように入力。
  - Process name　　Blokus
  - Input stream name　　Ist
  - Output stream name　　Ost
  - Stream width　　8
  - Stream depth　　4
- [次へ]をクリック。
- [完了]をクリック。



※ここでの「Process name」は「Blokus」以外でも問題はありません。

4. **プロジェクトオプションの変更**

- [Project]タブ->[Options]をクリックして「Options」ウィンドウを開く。
- [Generate]タブをクリックし、以下のように設定する。
  - Platform Support Package　　ALTERA(または Xilinx) Generic (Verilog)
  - 〔✔〕Generate active-low reset



5. プログラムを記述する

テンプレートを選択してありますので、プログラムのテンプレートが生成されています。「Blokus_hw.c」をクリックしてみましょう。ここの Blokus 関数がハードウェア化される部分で実際に処理を記述する場所です。

今回は分かりやすいように「入力に 1 加算して出力する」記述を追加しておきます。またデフォルトでは「#pragma CO PIPELINE」とパイプライン化指定のプラグマが記述されていますが、分かりやすさのために今回はコメントアウトしておきます。

```
23  void Blokus(co_stream Ist, co_stream Ost)
24  {
25      co_int8 nSample;
26      IF_SIM(int samplesread; int sampleswritten;)
27
28      IF_SIM(cosim_logwindow log;)
29      IF_SIM(log = cosim_logwindow_create("Blokus");)
30
31      do {    // Hardware processes run forever
32          IF_SIM(samplesread=0; sampleswritten=0;)
33
34          co_stream_open(Ist, O_RDONLY, INT_TYPE(STREAMWIDTH));
35          co_stream_open(Ost, O_WRONLY, INT_TYPE(STREAMWIDTH));
36
37          // Read values from the stream
38          while ( co_stream_read(Ist, &nSample, sizeof(co_int8)) == c
39          //#pragma CO PIPELINE
40              IF_SIM(samplesread++;)
41
42              // Sample is now in variable nSample.
43              // Add your processing code here.
44              nSample++;
45
46              co_stream_write(Ost, &nSample, sizeof(co_int8));
47              IF_SIM(sampleswritten++;)
48          }
49          co_stream_close(Ist);
50          co_stream_close(Ost);
51          IF_SIM(cosim_logwindow_fwrite(log,
52              "Closing filter process, samples read: %d, samples written: %d\n",
53              samplesread, sampleswritten);)
54
55          IF_SIM(break;)  // Only run once for desktop simulation
56      } while(1);
57  }
```

コメントアウト

ココを追加する

6. シミュレーション

ImpulseC のコードは、ソフトウェアとしてコンパイルしてシミュレーションすることができます。入力を供給する Producer()と出力を受け取る Consumer()は Blokus_sw.c に記述されていますので、確認してください。また、プロジェクトのフォルダに filter_in.dat というテキストファイルが作成されています。これが Producer()から与える入力データ列です。内容を確認し、必要に応じて与えたいデータ列を入力してください。

- [Project]タブ→[Build Software Simulation Executable]をクリックして、ソフトウェアとしてコンパイル。ログに「Build of target "Build_exe" complete」と表示されれば成功。
- [Project]タブ→[Launch Software Simulation Executable]をクリックして、ソフトウェアシミュレーションを実行。
- 実行画面が現れる→Enter キーを押して終了

出力はプロジェクトのフォルダの filter_out.dat というテキストファイルに格納されます。ここでは filter_in.dat の値に対して 1 を加えた値となっていることを確認します。

7. コンパイル

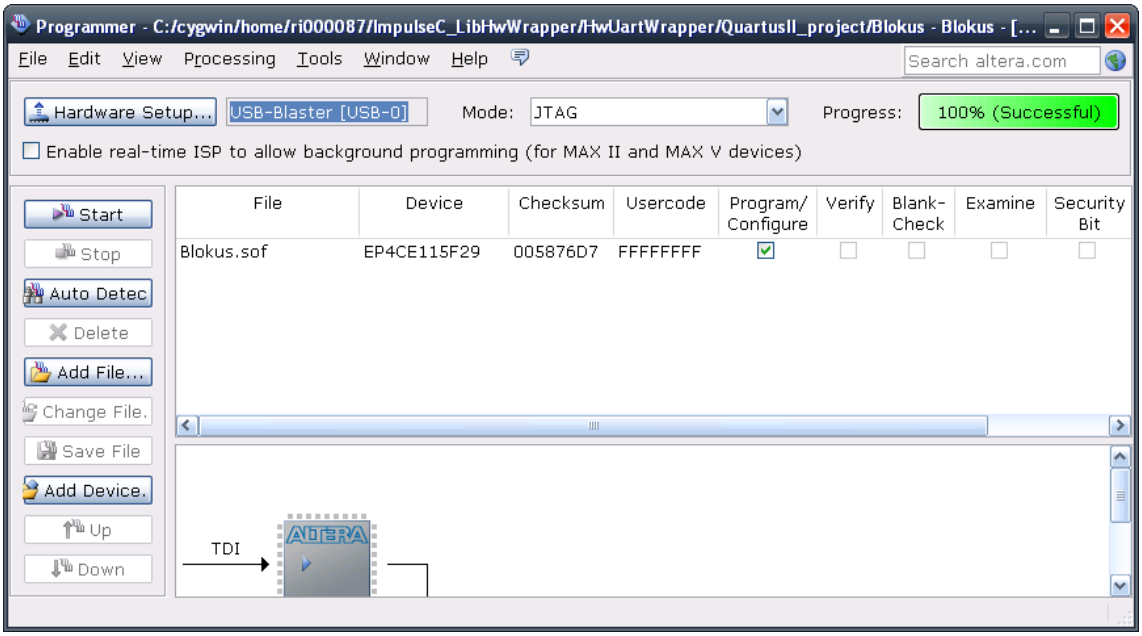
- [Project]タブ->[Export Generated Hardware (HDL)]をクリック。「Build ログ」に「Build of target 'export_hardware' complete」と表示されれば成功。

プロジェクトフォルダに「export」フォルダが作成され、その下に生成されたハードウェアのソースコードが保存されています。

8. FPGA への実装（ALTERA　QuartusII　DE2-115 の場合）

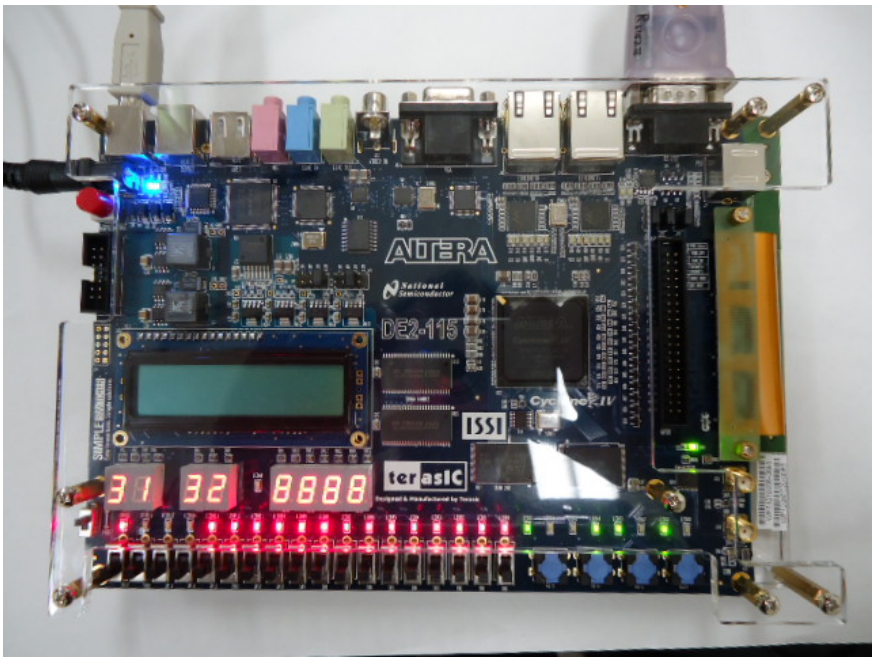
- Impulse C のプロジェクトフォルダに生成された「export」フォルダの中身を、キットの「Quartus_project」フォルダにコピー（上書き）する。
- 「Blokus.qpf」ファイルをダブルクリックし、QuartusII を起動。
- [Processing]タブ->[Start Compilation]をクリックし、コンパイルを行う。
- 「Full Compilation was successful」が出れば成功、OK を押す。
- DE2-115 を PC と USB で接続。（BLASTER のコネクタ）
- DE2-115 の SW19 が RUN 側になっていることを確認。
- [Tools]タブ->[Programmer]をクリック。Programmer のウインドウが開く。

- 「Programmer」ウィンドウの[Hardware Setup]をクリックする。
- 「Hardware Setup」ウィンドウの「Currently selected hardware」で
- 「USB-Blaster [USB-0]」を選択し、「Hardware Setup」ウィンドウを閉じる。
- 「Programmer」ウィンドウの「Add File」（あるいは「Change File」）をクリックし、Quartus プロジェクトフォルダの Blokus.sof を選択する。
- Blokus.sof の「Program/Configure」をチェック✔する。
- 「Programmer」ウィンドウの「Start」をクリックし FPGA へダウンロードする。



9. 実動作の確認

- DE2-115 と PC を RS232C のケーブルで接続する。
- TeraTerm などのターミナルソフトを起動する。シリアルの設定は「115200bps, no parity, 1 stopbit, 8bit, no flow control とする。
- ボードの KEY0 でリセットする。
- ターミナルソフトから 01234567 を入力する。+1 した 12345678 が返ってくることを確認する。（ボード上の 7SEG LED にも入力値と出力値が１６進数で表示される）



999. 関連情報

1) CoDeveloper トラブル情報 (2013.04.04)

症状:Windows7 で CoDeveloper の画面をリサイズすると CoDeveloper が反応しなくなる。Imuplse 社でも把握しており、対応中とのこと。

対処：プロジェクトファイル、*.icproj ファイルをダブルクリックして起動する。プロジェクト作成から最初の保存まではフルスクリーンで作業する。この方法で起動した場合、リサイズの際、反応しなくなるという現象が発生しなくなるようだ。