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

 $C: \label{eq:c:stop} Users \ensuremath{\texttt{Yizumi}} Desktop \ensuremath{\texttt{Blokus}} Blokus \ensuremath{\texttt{Quartus}} project \ensuremath{\texttt{Y}}$

2. Impulse C/Co Developer を起動

Getting Started Sample Projects Help and Support CoDeveloper Version 3	
Thank you for choosing CoDeveloper, from Impulse Accelerated Technologies. If this is your first time using CoDeveloper, we suggest you begin with our online Tutorials: Click here to usit the CoDeveloper Co-Line Tutorials Page	
Create a New Project Open an Existing Project or open one of your most recently modified projects by selecting from the following list:	roject
Recent Project Unablef	Modified on
For a 5st of featured sample projects for selected FPGA platforms, select the Sample Projects tab above. For additional help, including detailed step-by-step tutonials, select the Help and Support tab	
n	

- 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。

Create New Project		
Project types: Project Templates General Application Templates	Templates: Title Blank Project	Preview:
This template represents a blank projec	t with no source code and default settings.	
Project name:		
Location:		Browse
Create a folder for t	he project	
		OK Cancel

- 「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。

Create New P	Project			×
Project types:	mplates al Application Templates	Templates: Title Title Two input stream, one output s Two input stream, two output s Two input streams, two output Two input streams, one output One input stream, one memory One memory, one output stream Three parallel processes Three pipelined processes	Preview:	Consumer
This template r processes are Project name:	epresents a simple filter a provided for desktop simu Blokus	pplication with one input stream and one lation.	output stream. Producer and	d Consumer
Location:	C:¥	ne project		Browse
			OK	Cancel

※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
 - \succ Stream depth 4
- [次へ]をクリック。
- [完了]をクリック。

Code Generation Options Specify the code generation options				
Specify the process na unique and must be va alphanumeric symbols	ame, input stream name and output stream name. All of these names must be Ilid C-language identifiers (having no spaces or tabs, containing only and not starting with a number).			
Process name:	Blokus			
Input stream name:	Ist			
Output stream name:	Öst			
Stream width	8			
Stream depth	4			

※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)
 - > $[\checkmark]$ Generate active-low reset

Build Simulate Generate System Registration Platform Support Package:	Options	
Platform Support Package: Xilinx Generic (Verilog) Hardware Optimization and Generation Enable constant propagation Scalarize array variables Relocate loop invariant expressions Use std_logic types for VHDL interfaces Do not include co_ports in bus interface Additional optimizer options: Floating Point Options Include floating point library Include co_math library Enable floating point optimization Allow double-precision types and operators Use higher latency, faster clock operators	Build Simulate Generate System Registration	
Floating Point Options Output Directories Include floating point library Include co_math library Enable floating point optimization Allow double-precision types and operators Use higher latency, faster clock operators Hardware export directory:	Platform Support Package: Xilinx Generic (Verilog) Hardware Optimization and Generation F Enable constant propagation Scalarize array variables Relocate loop invariant expressions Additional optimizer options:	 Generate dual clocks ✓ Generate active-low reset Use std_logic types for VHDL interfaces ✓ Do not include co_ports in bus interface Additional library options:
Floating Point Options Output Directories Include floating point library Hardware build directory: Include co_math library hw Enable floating point optimization Software build directory: Allow double-precision types and operators Hardware export directory:	Additional optimizer options:	Additional library options:
Enable floating point accumulators Image: second secon	Floating Point Options Floating Point Options Flocude floating point library Enable floating point optimization Allow double-precision types and operators Use higher latency, faster clock operators Enable floating point accumulators Use extended precision accumulators	Output Directories Hardware build directory: hw Software build directory: sw Hardware export directory: export Software export directory: export

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 0st)
     - {
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
                  // Hardware processes run forever
31
           do {
32
              IF_SIM(samplesread=0; sampleswritten=0;)
33
34
               co_stream_open(Ist, 0_RDONLY, INT_TYPE(STREAMWIDTH));
               co_stream_open(0st, 0_WRONLY, INT_TYPE(STREAMWIDTH));
35
36
               // Read values from the stream
37
                                                                            コメントアウト
38
               while ( co_stream_read(Ist, &nSample, sizeof(co_int8)) ==
               //#pragma CO PIPELINE
39
40
                   IF SIM(samplesread++;)
41
                   // Sample is now in variable nSample.
42
                                                                            ココを追加する
43
                   // Add your processing code here.
44
                  nSample++;
45
46
                   co_stream_write(Ost, &nSample, sizeof(co_int8));
47
                   IF_SIM(sampleswritten++;)
48
               3
               co_stream_close(Ist);
49
               co stream close(Ost);
50
               IF_SIM(cosim_logwindow_fwrite(log,
51
                   "Closing filter process, samples read: %d, samples written: %d\n",
52
53
                   samplesread, sampleswritten);)
54
               IF_SIM(break;) // Only run once for desktop simulation
55
           } while(1);
56
57
       3
```

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 エ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 Quartusl 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**ノ**する。
- 「Programmer」ウィンドウの「Start」をクリックし FPGA ヘダウンロードする。

🖥 Programmer - C:/cygwin/home/ri000087/ImpulseC_LibHwWrapper/HwUartWrapper/QuartusII_project/Blokus - Blokus - [🔳 🗖 🔀									
<u>Eile E</u> dit <u>V</u> iew	[,] P <u>r</u> ocessing <u>T</u> ools <u>W</u> indow <u>H</u> elp 록				Searc	Search altera.com			
Hardware Setup, USB-Blaster [USB-0] Mode: JTAG Progress: 100% (Successful)									
Enable real-tir	me ISP to allow backg	round programmi	ng (for MAX	II and MAX \	/ devices)				
Start	File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank- Check	Examine	Security Bit
🔎 Stop	Blokus.sof	EP4CE115F29	005876D7	FFFFFFF	~				
🎢 Auto Detec									
🗙 Delete									
🎽 Add File									
🔓 Change File.	4		111						
🔛 Save File									
👌 Add Device.									
1 [™] Up ↓ [™] Down									

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。