アルテラ FPGA 向け、PLL リコンフィグの応用回路

1. PLL リコンフィグとは

アルテラ FPGA は PLL 機能を内蔵しています。PLL を利用して基本周波数を逓倍、分周したクロックを利用することができます。　通常、FPGA 開発ツール Quartus2（以下、Q2）の MegaWizard プラグインマネージャを利用して PLL を設定し、希望のクロック周波数を得ることができます。　PLL の設定を変更する場合は、プラグインマネージャを起動・設定後に再コンパイルすることにより希望の周波数を得ることができます。

PLL リコンフィグでは Q2 でのコンパイルの手間を省き、任意のタイミングで PLL 機能を再設定できます。Smart-USB Plus 製品では、USB を利用して PLL 設定データを FPGA にダウンロードし、任意のタイミングで PLL 設定ができるような仕組みを簡単に構築できます。

【適用製品】

- ✓ CX-USB2 システム開発ボード　　（Cyclone3-FPGA　搭載）
- ✓ CX-Card4 システム開発ボード　　（Cyclone4-FPGA　搭載）

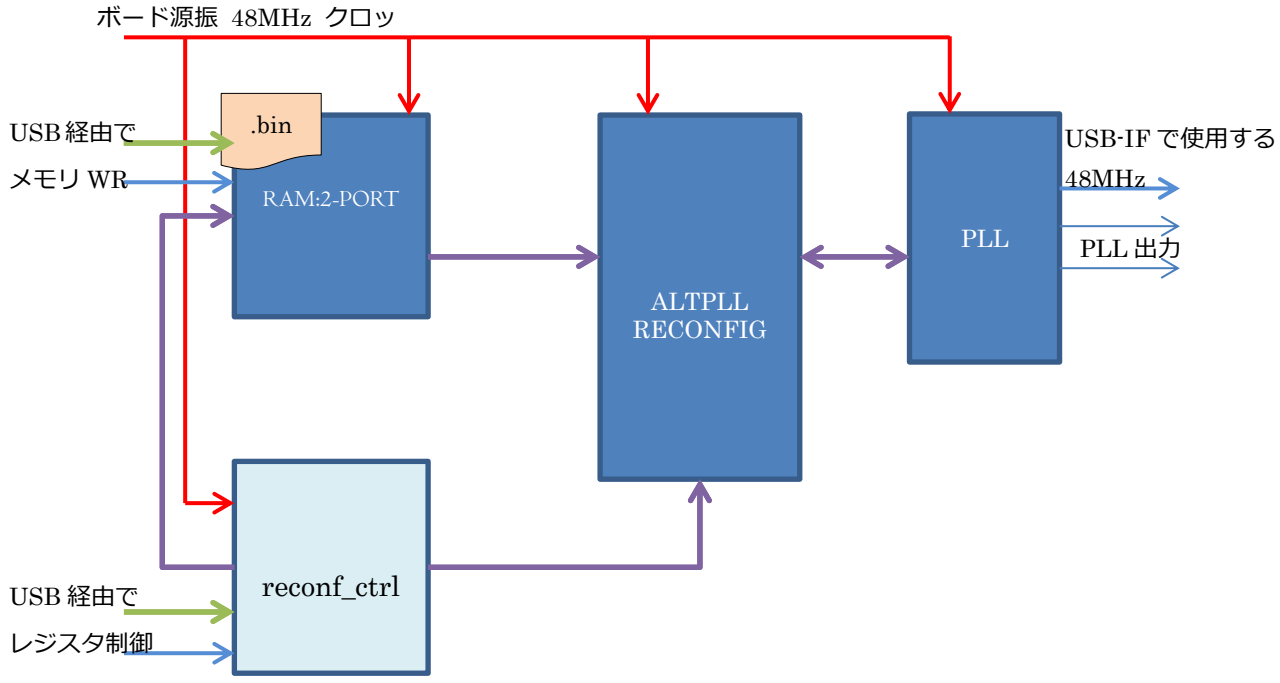
【参考資料】

- ✓ アルテラ社 AN507　http://www.altera.co.jp/literature/an/an507_j.pdf
- ✓ ALTPLL_RECONFIG　ユーザガイド　http://www.altera.com/literature/ug/ug_altpll_reconfig.pdf

2. PLL リコンフィグの制御概要

【PLL リコンフィグ機能を実現する FPGA 回路】

PLL をリコンフィグするには、PLL、　ALTPLL_RECONFIG、　RAM2-PORT メガファンクションを利用し、これらを制御する当社製サンプル回路「reconf_ctrl」と、PLL メガファンクションで生成した設定ファイル(.bin)が必要です。



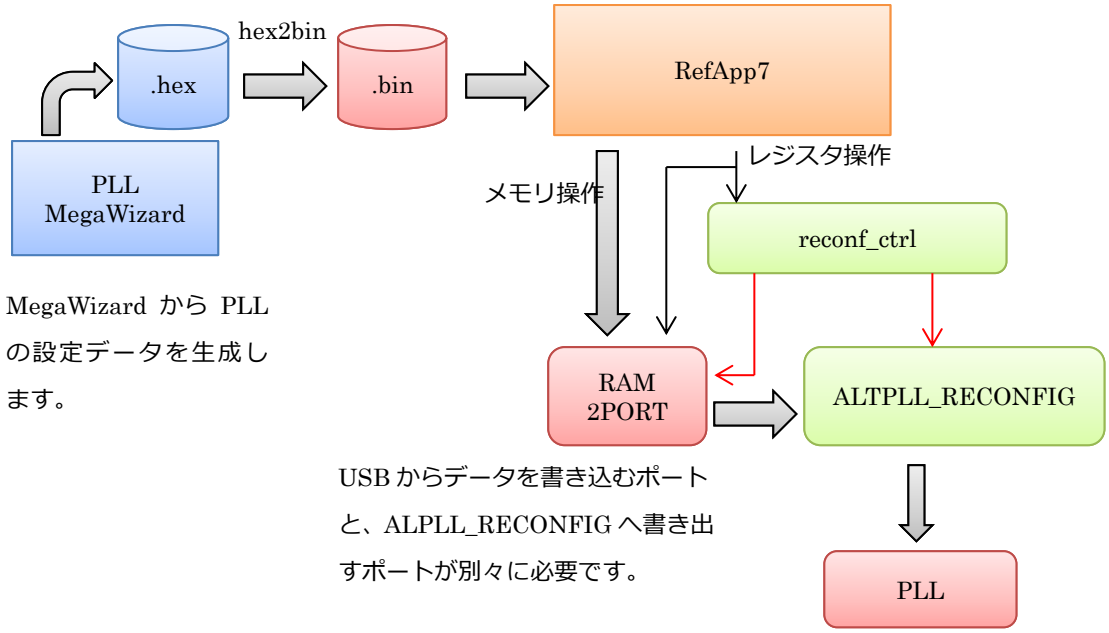
<図１．PLL リコンフィグ・ブロック図>

【PLL リコンフィグ制御の概要】

Smart-USB Plus 製品には、USB インタフェースのために使用する 48MHz の水晶発振器を FPGA 外部に搭載しています。この 48MHz クロックから FPGA の PLL を利用して必要なクロック周波数を生成することができます。　ここでは、ボードの制御アプリケーションに「RefApp7.exe」を利用します。このアプリのメモリ操作、レジスタ操作を利用して、PLL のリコンフィグを行います。

※RefApp7 では、hex ファイルを直接ボードに転送できません。バイナリデータ(.bin)に変換してからメモリ操作を行います。

PLL メガファンクションで生成した設定ファイル（.hex）をバイナリファイルに変換後、USB 経由で RAM に転送し、USB レジスタアクセスによる任意のタイミングで、RAM-2PORT から ALTPLL_RECONFIG ブロックへ設定ファイルを移動（書き込み）します。　さらに ALTPLL_RECONFIG 内に移動した設定ファイルを、任意のタイミングで PLL にダウンロードし、PLL をリコンフィグすることができます。



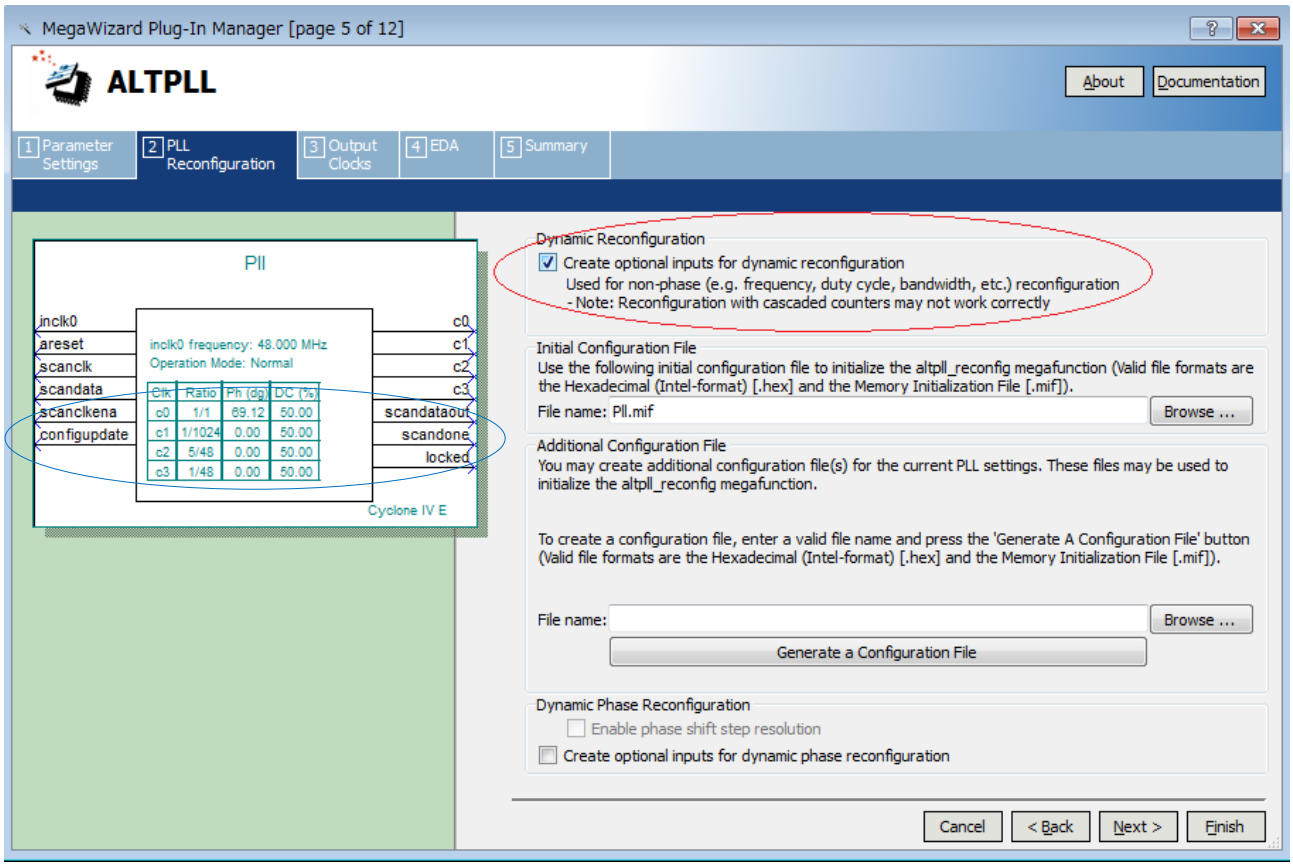
<図 2. PLL リコンフィグ制御の流れ>

3. 各ブロックの設定

Q2 開発ツールの MegaWizard Plug-in Manager から、ALTPLL、ALTPLL_RECONFIG、RAM-2PORT の 3 ブロックを生成します。

【ALTPLL　ブロックの生成】

ALTPLL ブロックで、48MHz クロックから様々なクロックを設定・出力ができます。

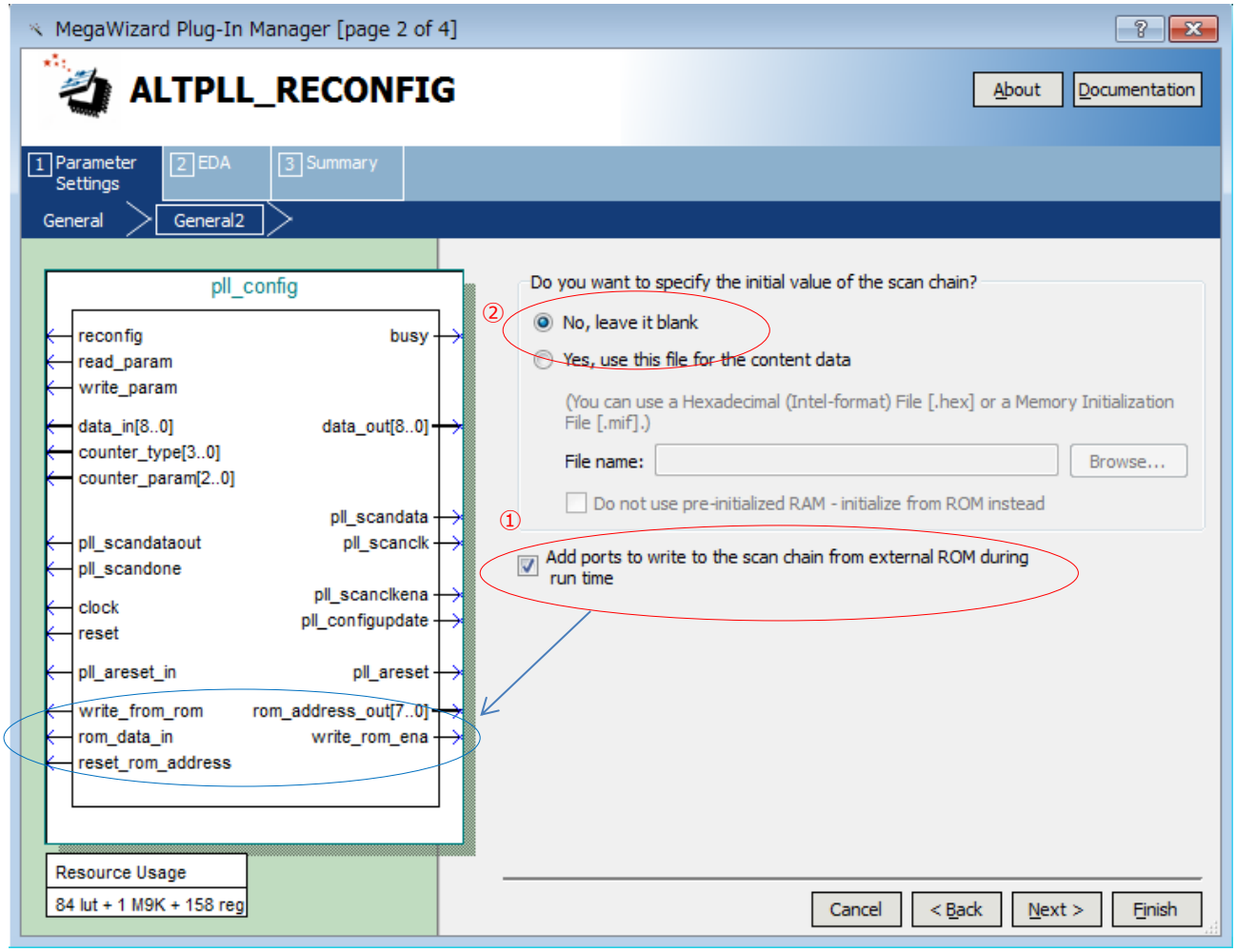


<図 3. ALTPLL 設定画面(5/12)>

図 3 に示すように、Dynamic Reconfiguration のチェックボックスにチェックを入れてください。青枠で囲んだ信号線が追加されます。これらの信号線を ALTPLL_RECONFIG ブロックと接続します。

【ALTPLL__RECONFIG　ブロックの生成】

PLL のリコンフィグを制御するブロックです。



<図 4．ALTPLL_RECONFIG 設定画面（２／４）>

ALTPLL_RECONFIG ブロックは、1 個の M9K メモリブロックを含んでいます。　一般的な手法では、このメモリ内に設定ファイルを設定し、Q2 でコンパイル後、任意のタイミングで PLL のリコンフィグができます。この方法では 1 種類の PLL 設定だけしか処理できません。

USB 経由で任意の PLL 設定ファイルをダウンロードするには、①を選択し、②をチェックして ALTPLL_RECONFIG 外部のメモリから設定データを取り込むようにします。

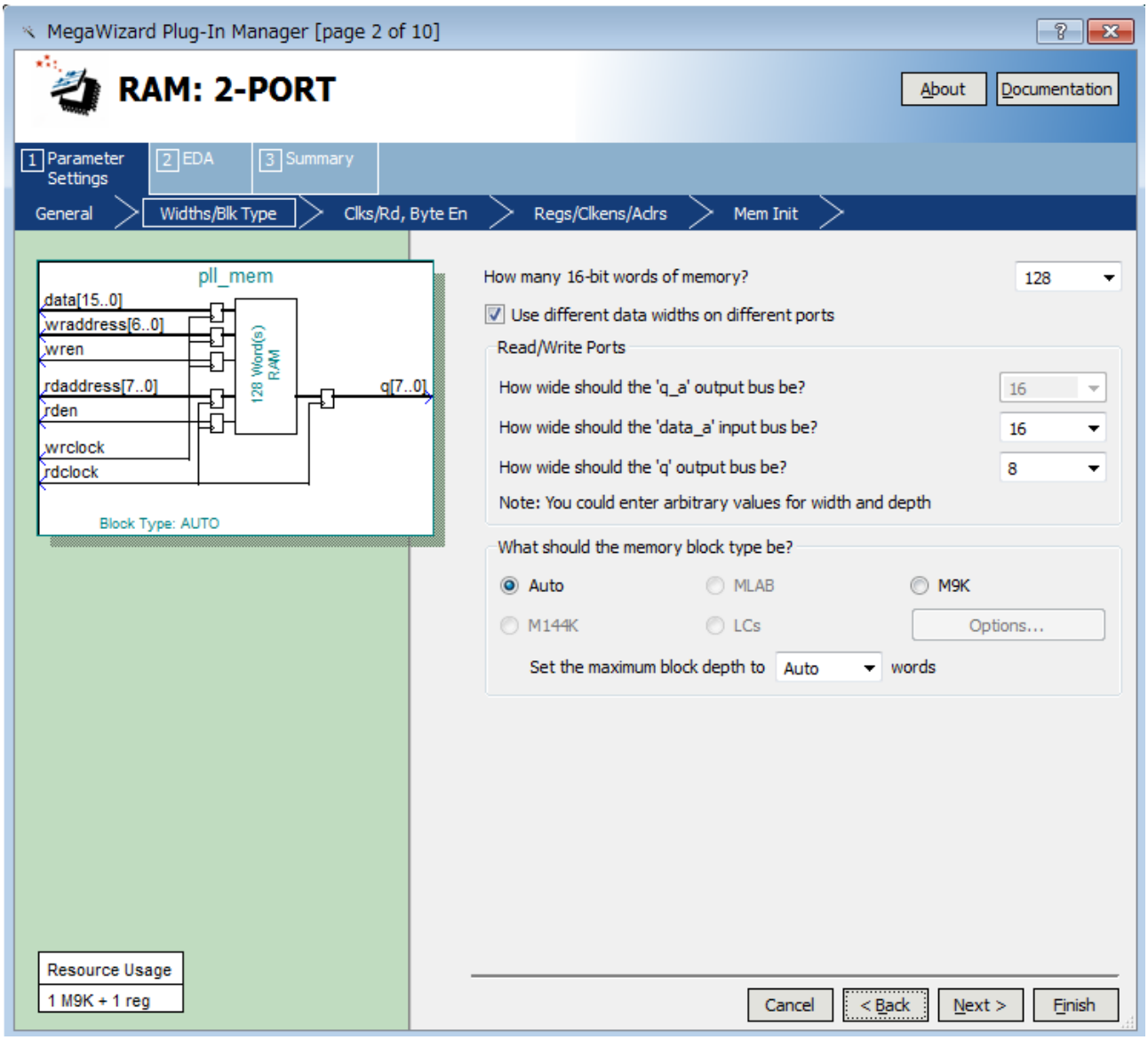
【RAM-2PORT ブロックの生成】

PLL 設定用データを一時保管させるためのデュアルポートメモリを作成します。

RefApp7.exe アプリのメモリ操作画面から、このメモリに対して PLL 設定ファイルを書き込むためのポートと、ALTPLL_RECONFIG へデータを書き出すポートの２つを生成します。

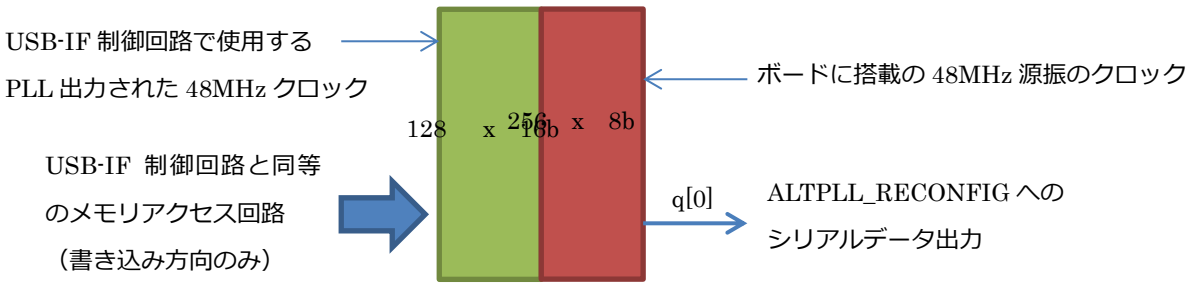
(各パラメータの設定)

USB 経由でデータを書き込むポートは、128 ワード x 16bit、ALTPLL_RECONFIG へ書き出すポートは 256 ワード x 8bit に設定します。



<図 5．デュアルポート RAM の設定>

ここで設定したメモリの出力は 8bit 幅(q[7:0])ですが、ALTPLL_RECONFIG の rom_data_in ポートは 1bit なので、q[0]ビットと rom_data_in ポートを接続してください。　PLL の設定ファイル（.hex）では下位 1bit だけが有効データのためです。



<図 6．デュアルポート RAM の制御概要>

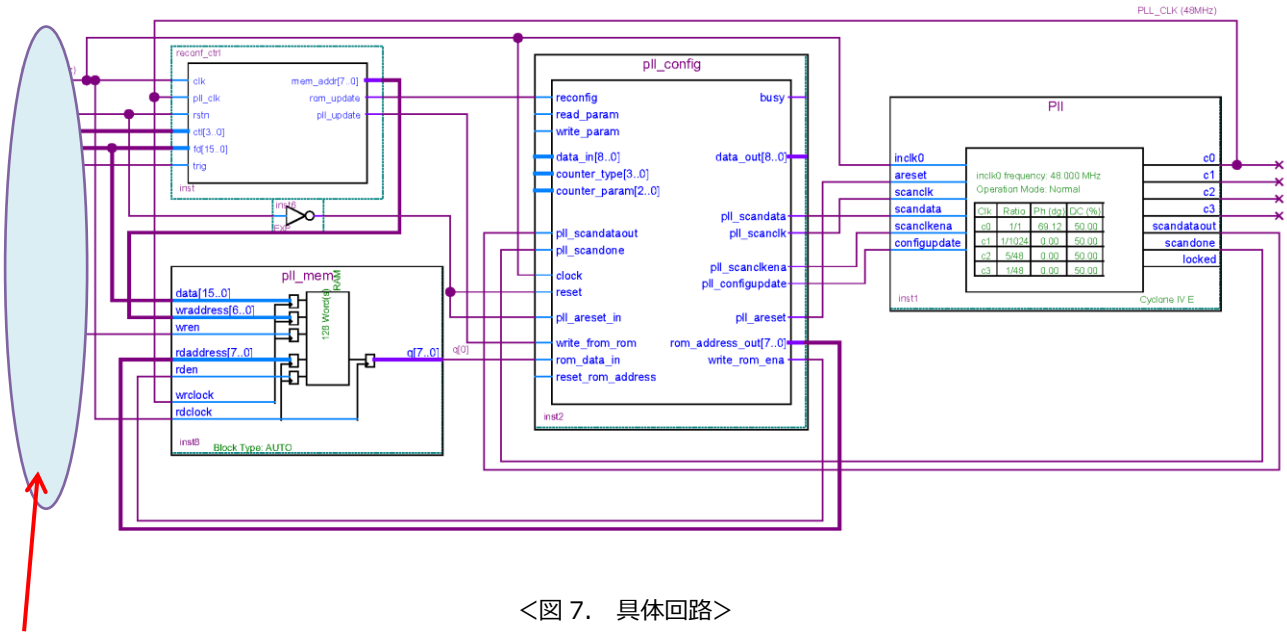
【reconf_ctrl ブロックの概要】

reconf_ctrl.v ファイルは、弊社オリジナルのサンプル回路です。このブロックでは、USB 経由でメモリに設定ファイルを書き込む際のアドレスと書き込みタイミングを生成します。

また、ALTPLL_RECONFIG ブロックへのデータ書き出しと、PLL リコンフィグをイニシエイトする update 信号を生成します。

ポート名	属性	機能
clk	入力	ボード搭載の 48MHz 水晶発振器出力信号
pll_clk	入力	FPGA の PLL を通過した 48MHz クロック : 通常、USB-IF 制御回路で使用するメインクロック
rstn	入力	リセット信号（low active）
ctl[3:0]	入力	USB-IF 制御回路で使用する制御線 ctl[3]=rgdtn, ctl[2]=cndn, ctl[1]=wrn, ctl[0]=rdn
fd[15:0]	入力	USB-IF 制御回路で使用する USB データ線(16bit)
trig	入力	信号の立ち上がりエッジを検出し、rom_update 信号を出力する
mem_addr[7:0]	出力	trig 信号アクティブ後、ALTPLL_RECONFIG へデータを書き出すための、デュアルポート RAM アドレスを生成
rom_update	出力	trig 信号アクティブ後、デュアルポート RAM から ALTPLL_RECONFIG へデータを読み出すタイミングを生成
pll_update	出力	ALTPLL_RECONFIG から PLL へデータをダウンロードさせるタイミングを生成

4. 各ブロック接続回路図



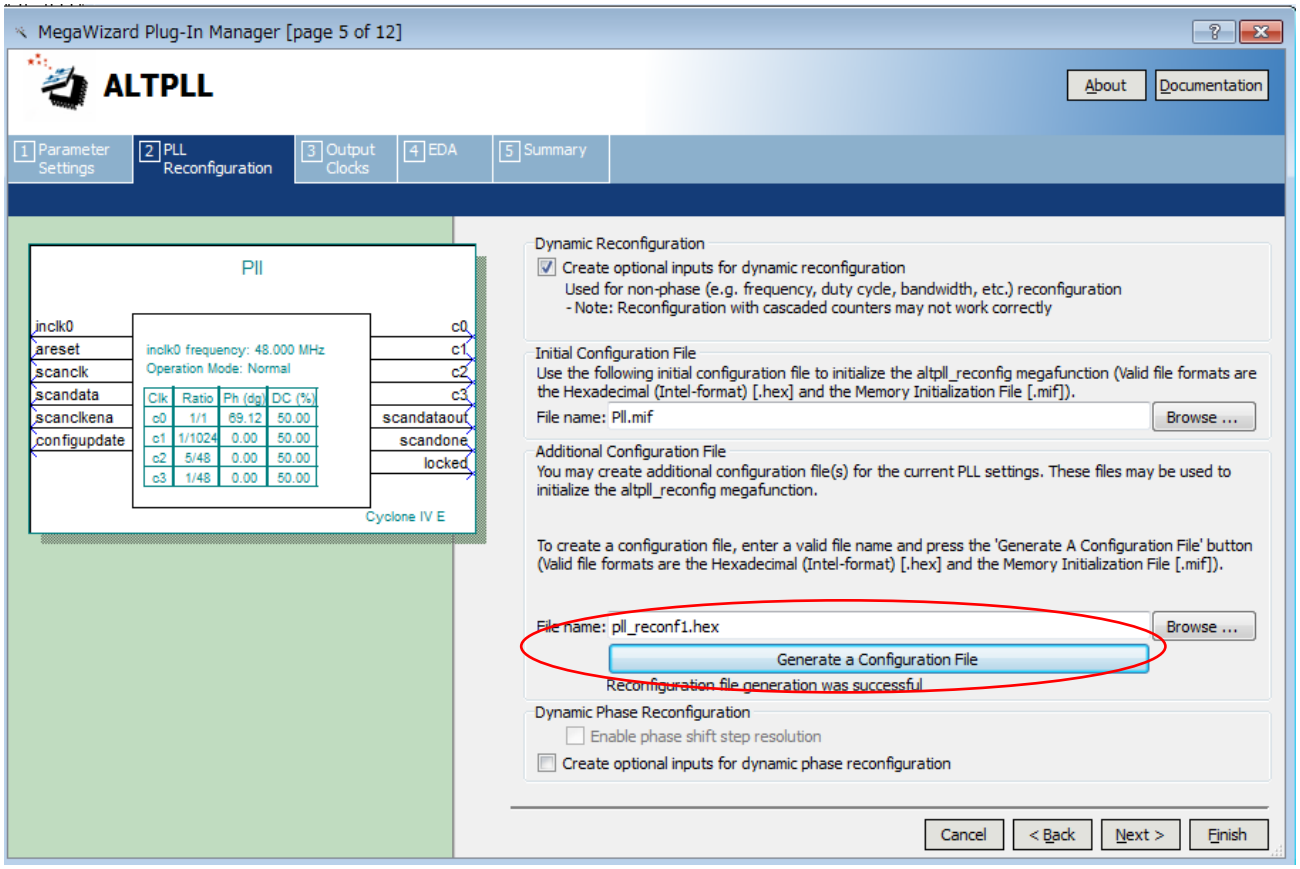
<図 7.　具体回路>

ここに示す各ポートは、USB-IF 制御回路ブロックで利用します。　例えば、USB 経由で設定ファイルをメモリに書き込む場合、レジスタアクセスにより特定ビットをメモリイネーブル信号として、pll_mem ブロックの wren ポートに接続したり、reconf_ctrl.v ブロックの trig ポートに、特定のレジスタの 1bit をアサインすることにより、RefApp7.exe アプリのレジスタ操作により、PLL リコンフィグを簡単に操作できます。具体的な操作方法は次章を参照してください。

5. PLL リコンフィグ操作方法

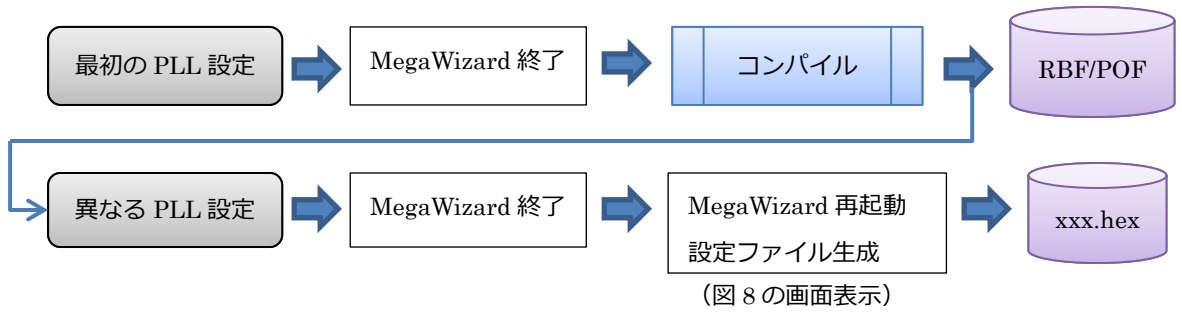
FPGA 回路の準備が整ったので、RefApp7.exe 制御アプリから PLL 設定ファイルのダウンロードと、制御回路の操作を行います。

【PLL 設定ファイルを用意する】



<図 8．PLL 設定画面（2）>

デフォルト時の PLL 設定を行い、FPGA プロジェクトをコンパイルした後、PLL 設定ファイルだけを生成するために ALTPLL の MegaWizard を再度起動します。デフォルト時の PLL 設定と同じ手順で、異なるクロック周波数を設定し、「Finish」ボタンをクリックして MegaWizard を終了します。　その後、再度 MegaWizard を起動し、図 8 の画面でファイル名 xxx.hex を指定して「Generate a Configuration File」ボタンをクリックし、設定ファイル（xxx.hex）を生成できます。



<図 9．PLL 設定ファイル生成手順>

【PLL 設定ファイルを変換する】

生成した xxx.hex ファイルは、RefApp7.exe 制御アプリでそのままデータ転送ができません。このため、バイナリデータ(xxx.bin)に変換が必要です。

今回は sourceforge.jp からダウンロードできる hex ファイルを bin ファイルに変換するユーティリティアプリ(hex2bin)を使用します。このユーティリティは、Windows のコマンドプロンプトから実行できます。この hex2bin.exe により、xxx.bin ファイルを生成できます。

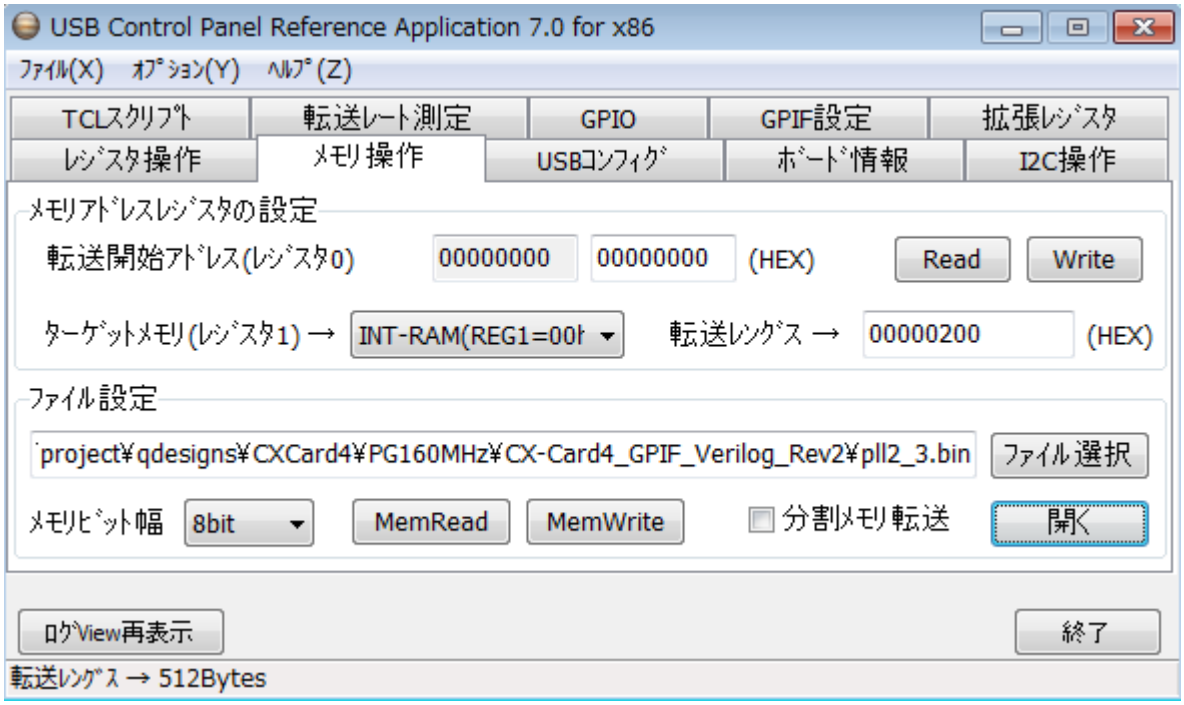
生成したバイナリファイルのファイルサイズは 144 バイトです。最下位ビットのみ使用するので、実際に PLL 設定ファイルとしては 144bit のデータになります。

参考サイト：　http://sourceforge.jp/projects/sfnet_hex2bin/

【RefApp7 制御アプリからの操作】

RefApp7.exe を起動し、FPGA をコンフィグした後、「メモリ操作」タブから生成した bin ファイルをボードにダウンロードします。

RefApp7.exe のメモリ操作では最小転送ファイル容量を 512 バイトとしています。511 バイト以下のファイルは転送できません。このため、RefApp7 では 512 バイト転送（200h）を設定します。実際のバイナリファイルが 144 バイトなので、RefApp7 が自動的に残りの 368 バイト分に対して“００”を追加して転送します。　FPGA 回路の reconf_ctrl.v では、144 バイトを受信したら余計なデータを受信しないような回路構成になっています。



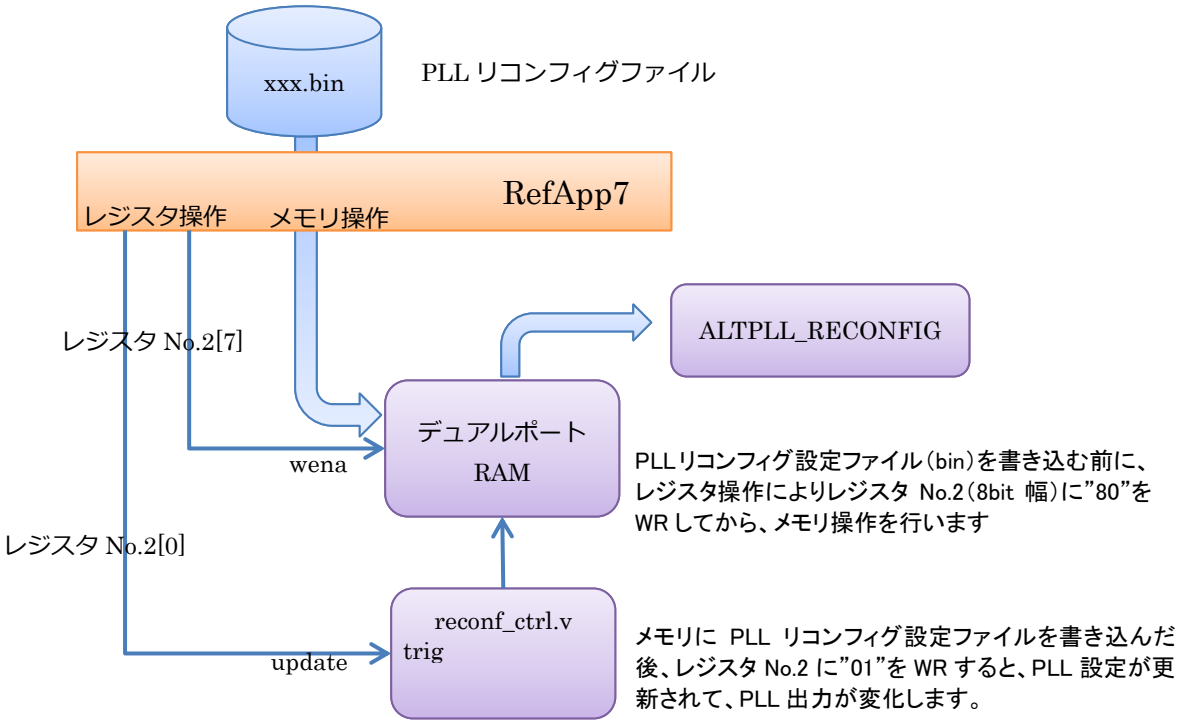
<図 10．RefApp7 アプリでの PLL 設定データ転送>

【サンプル FPGA プロジェクト】

サンプル FPGA プロジェクトでは、レジスタ操作により、PLL 設定用データを格納するデュアルポート RAM のイネーブルや、PLL リコンフィグのタイミング制御を行っています。

レジスタ No.2（8bit 幅）の最上位ビットに“１”を書き込むと、PLL 設定用データメモリのデュアルポート RAM・wren ポートが有効になり、引き続き図 10 で示すメモリ操作を行うことで、正しくデュアルポート RAM に設定ファイルを書き込めます。

また、レジスタ No.2 の最下位ビットを「reconf_ctrl.v」の trig ポートに接続しているので、ここに”1“を書き込むことで、PLL のリコンフィグができます。



<図 11. サンプル FPGA プロジェクトでの動作手順>

【サンプル FPGA プロジェクト】

以下 URL からダウンロードできます。

CX-Card4 用：http://www.prime-sys.co.jp/DownLoad/sample/CX-Card4_GPIF_Verilog_Rev2_PLLRECONF.zip

CX-USB2 用：http://www.prime-sys.co.jp/DownLoad/sample/GPIF_Verilog_RegEx_PLL.zip

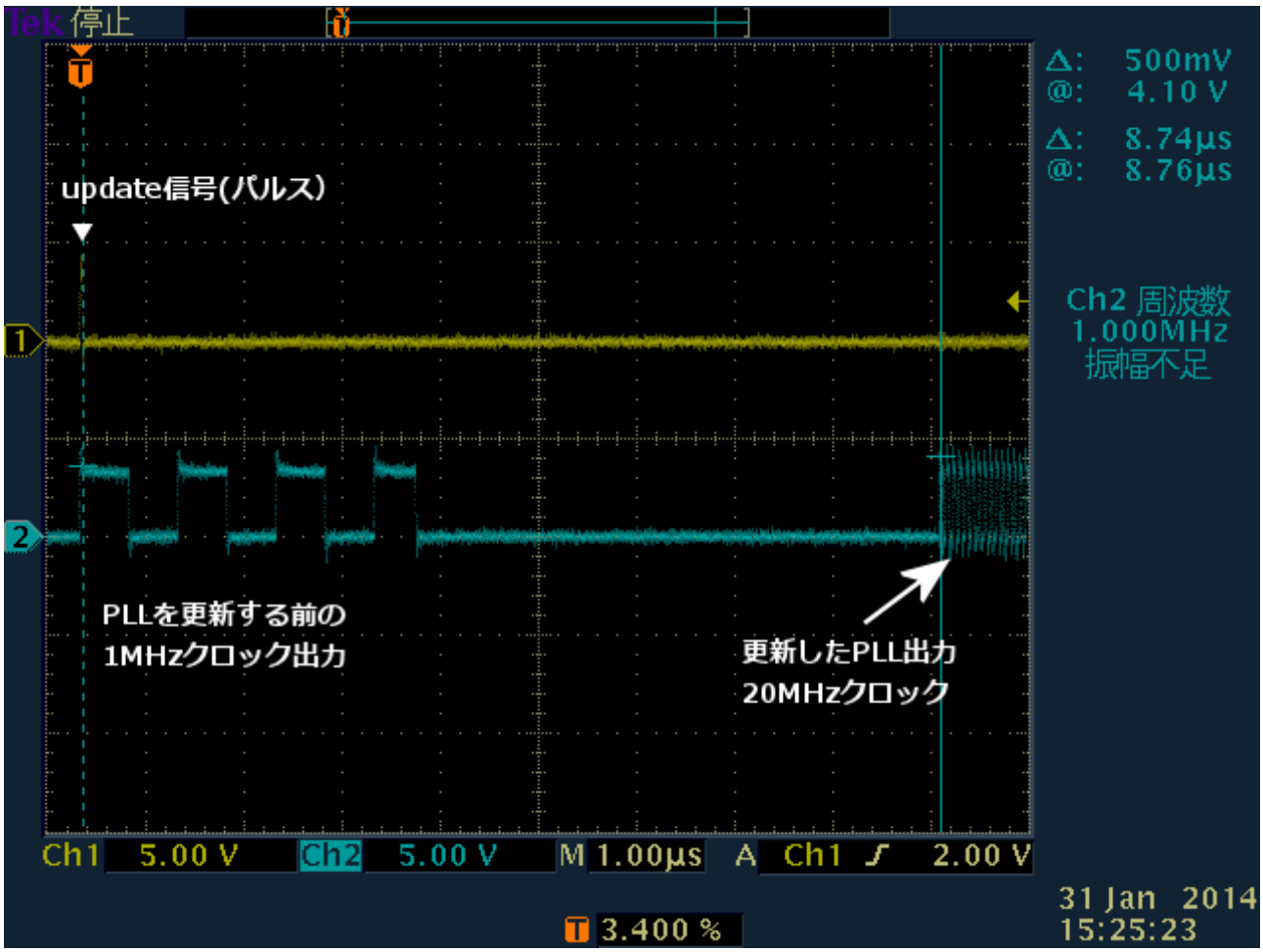
Cyclone3/4 用 PLL リコンフィグコントローラ回路（VerilogHDL）：

http://www.prime-sys.co.jp/DownLoad/sample/reconf_ctrl_verilog.zip

【PLL リコンフィグ時の PLL ロック時間】

FPGA データシートでは、PLL のロック時間は最大 1ms に規定されています。図 12 に示すオシロスコープ観測画面では、PLL 設定を更新するタイミング（update：サンプル FPGA プロジェクトでは、レジスタ No.2 に”01”を WR したタイミング）から、実際にクロック出力されるまでの時間を観測したものです。

この例では、PLL 更新までの所要時間が、約 8.7us になっています。



<図 12. PLL 更新時間>