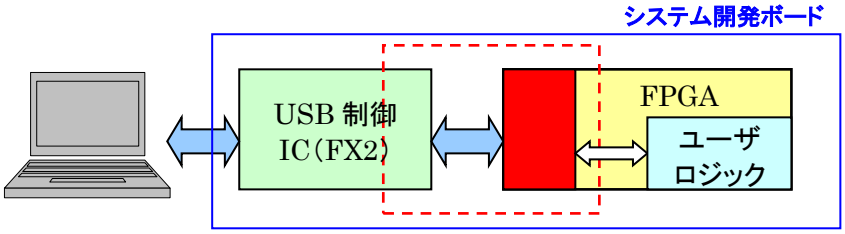
アルテラ社ツール"Qsys"を利用した Smart-USB Plus 製品用リファレンス回路

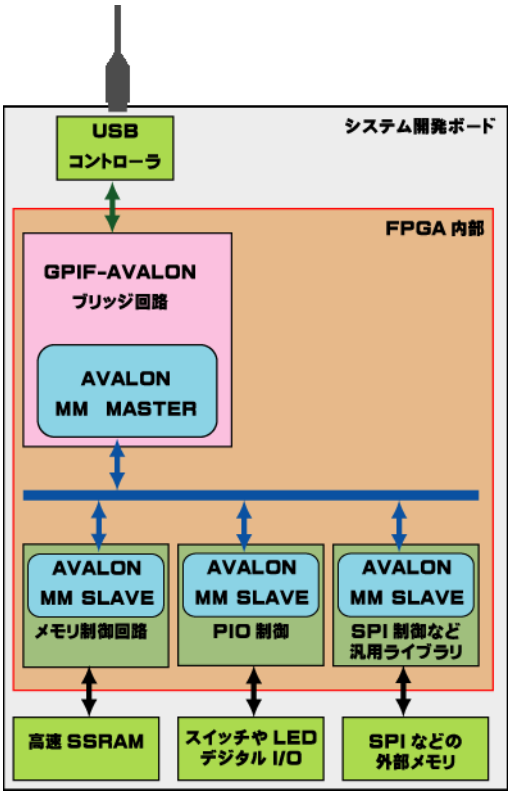
GPIF-AVALON ブリッジ回路

1. QuartusII の Qsys システム統合ツールで利用できる GPIF-AVALON ブリッジとは？

GPIF-AVALON ブリッジとは、当社製 USB2.0 システムコアである「Smart-USB Plus 製品」の外部インターフェース「GPIF」と、ALTERA 社製 FPGA 専用内部接続バス「AVALON」を相互に接続する為のバスブリッジです。（ここで示す GPIF とは、USB 制御 IC（以下、FX2）と FPGA 間の接続のこと（図 1 で赤表示した部分）です。）



<図 1. システムブロック図>



<図 2. GPIF-AVALON ブリッジのブロック図>

従来、Qsys では、AVALON バス・マスタとして NiosII プロセッサが必須でしたが、GPIF-AVALON ブリッジを使用することで、バスマスタの NiosII に代わり、AVALON バス・ペリフェラルを利用することができます。 例えば、SPI 通信など NiosII を実装するまでもなく、ホスト PC から USB2.0 インターフェースを経由して AVALON バスに接続した SPI ライブラリを直接アクセスすることができます。この様に、製品添付の制御ソフトウエア RefApp7 又はお客様が開発した制御アプリケーションを使用し、Qsys で用意される無償の回路ライブラリに、ホスト PC から USB 経由でダイレクトにアクセスすることが可能です。

また、GPIF-AVALON ブリッジは NiosII とも共存できるので、PC と Nios2 間でデータのやりとりが可能です。このため、Smart-USB Plus 製品のアルテラ FPGA 搭載ボードを利用して、USB インタフェース付きのマイコンボードとして運用ができます。

例えば、画像処理システムへの適用の場合、PC 側アプリケーションで各フォーマットの画像データをバイナリデータ化し、ボード上のバッファ・メモリへ転送後、Nios2 プロセッサが各種画像処理を実施して、フラットパネルへ表示すること等が可能になります。

GPIF_AVALON バス・ブリッジ回路は、GPIF_Master という名称でモジュール化されています。

2. GPIF-AVALON ブリッジの目的

- **Smart-USB Plus 製品ファミリの使いやすさを向上**
- **各種 Smart-USB Plus 製品内でのマイグレーション性を向上**
- **GPIF インターフェース回路の移植性の向上**
- **豊富な Qsys 用コンポーネントの有効利用**
- **Nios2 と組み合わせることで、USB 付きのマイコンボード化を促進**

3. 使用環境

GPIF-AVALON ブリッジ（GPIF_Master）製作時点では以下の環境での動作を確認しています。 ただし、Qsys 用コンポーネント全ての動作を保証する訳ではありません。 動作確認をしているコンポーネントは限定的ですので注意して下さい。

対応ツールバージョン：

- QuartusII 11.0 以降 （WebEdition でも動作します）
- MegaCore IP ライブラリ

※NiosII を使用しない限り「Nios II Embedded Design Suite & Service Pack 1 」をインストールする必要はありません。

ボードサンプル回路（Qsys_CXUSB2_v11.zip）は、以下 URL から無償ダウンロードできます。

http://www.prime-sys.co.jp/DownLoad/GPIF_Avalon

上記 URL には、SX-USB3、CX-USB2、CX-Card4、CX-Card2 ボード用のサンプル・プロジェクトを用意しています。これらのボード以外でも Smart-USB Plus 製品ファミリでアルテラ FPGA 搭載製品なら、すべてのボードに適用することができます。
GPIF_Master の最新版は Ver1.7、ZBT_SSRAM の最新版は Ver1.5 です。

4. GPIF-AVALON ブリッジの利用方法

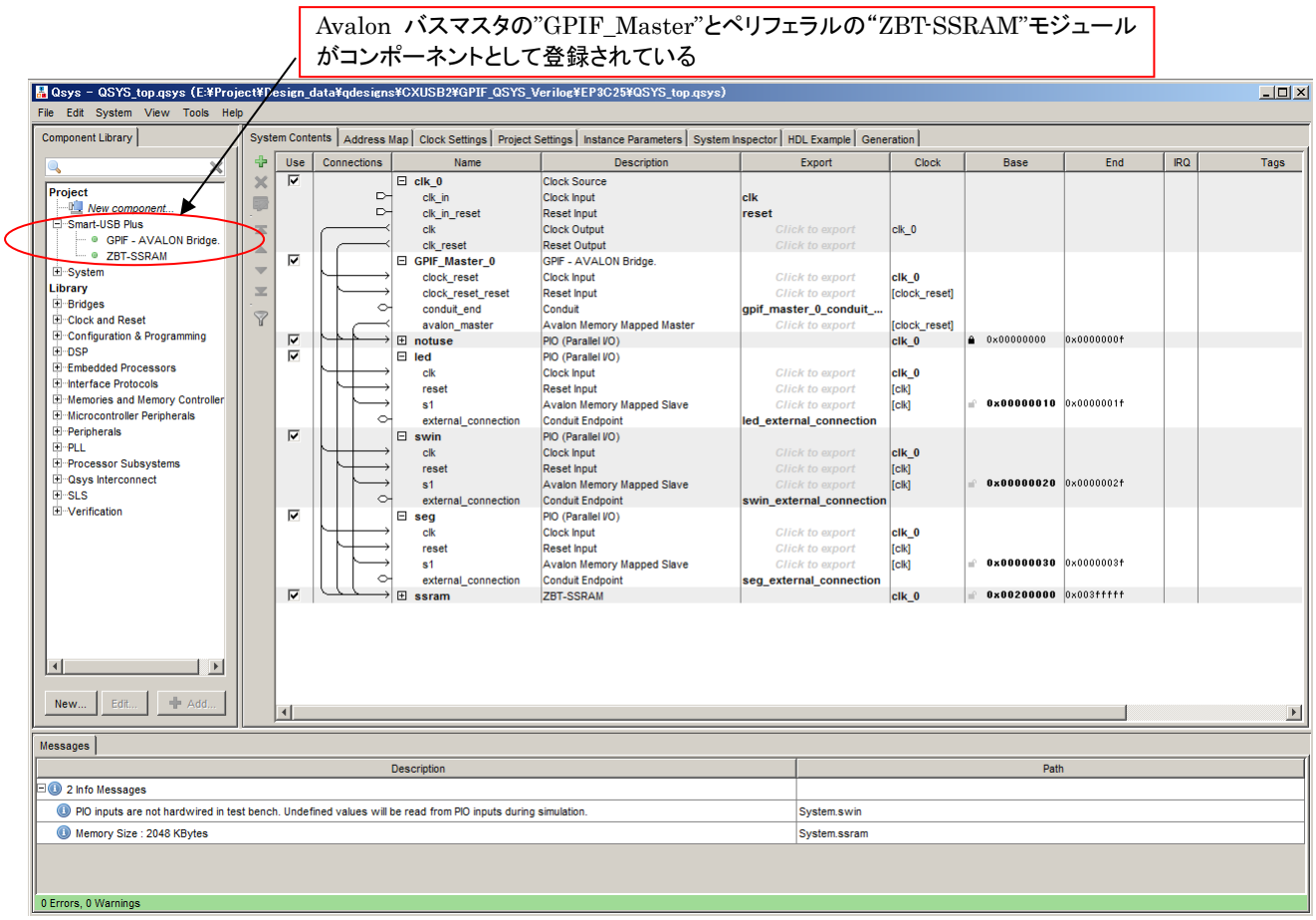
4.1 CX-USB2 用サンプル回路の場合

サンプル回路： Qsys_CXUSB2_v11.zip
サンプル回路は、QuartusII（以下、Q2）プロジェクトを圧縮(zip)しています。

ダウンロードしたファイルを解凍すると、Qsys で使用するコンポーネント・データと、CX-USB2 システム開発ボード用の FPGA サンプル回路ができます。 プロジェクト名は GPIF_Qsys です。

Q2 でプロジェクトをオープンし、“Start Analysis & Synthesis”を実行すると、Project Navigator 欄にデザインファイルの階層構成が表示されます。 Qsys の起動は、Q2 のツールバー”Tools”から”Qsys”を選択します。次に Qsys プロジェクトを選択する画面になるので、ここで”QSYS_top.qsys”を選択し、Qsys を起動します。

図 3 に示す System Contents 画面では、”Component Library”欄に示されるコンポーネントを追加することで、Avalon バスペリフェラルを追加できます。



<図 3. Qsys オープニング画面>

【Avalon バスペリフェラルの追加】

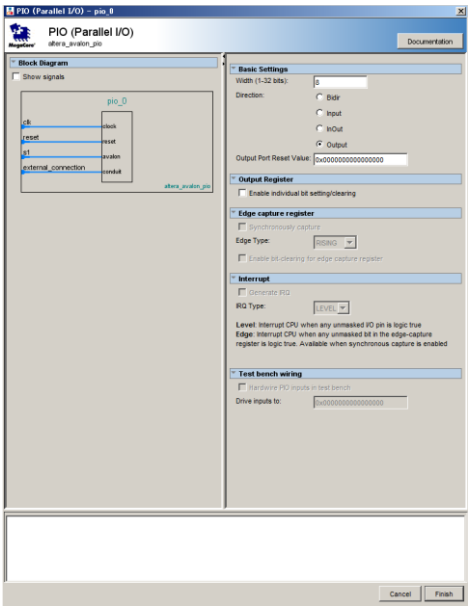
図 3 に示すシステムに、PIO コンポーネントを利用し、ハードウエア・レジスタを追加する場合の例を示します。

サンプル回路では、Avalon バスマスタである”GPIF_Master_0” コンポーネントのベースアドレスを”0x00000000” に設定しています。各 PIO ペリフェラルのベースアドレスは、以下の表 1 の通りです。

コンポーネント名	ベースアドレス (Hex)	備考	RefApp7 のレジスタ操作タブから制御できるレジスタ番号(Dec)
unused	0x00000000	GPIF_Master のベースアドレスと同じ値に必ず設定 32bit 幅設定	ー 未使用
led（LED)	0x00000010	8bit 幅設定	4
swin （スイッチ入力）	0x00000020	8bit 幅設定	8
seg （7 セグ表示）	0x00000030	16bit 幅設定	12
ZBT-SSRAM （メモリアクセス）	0x00200000		

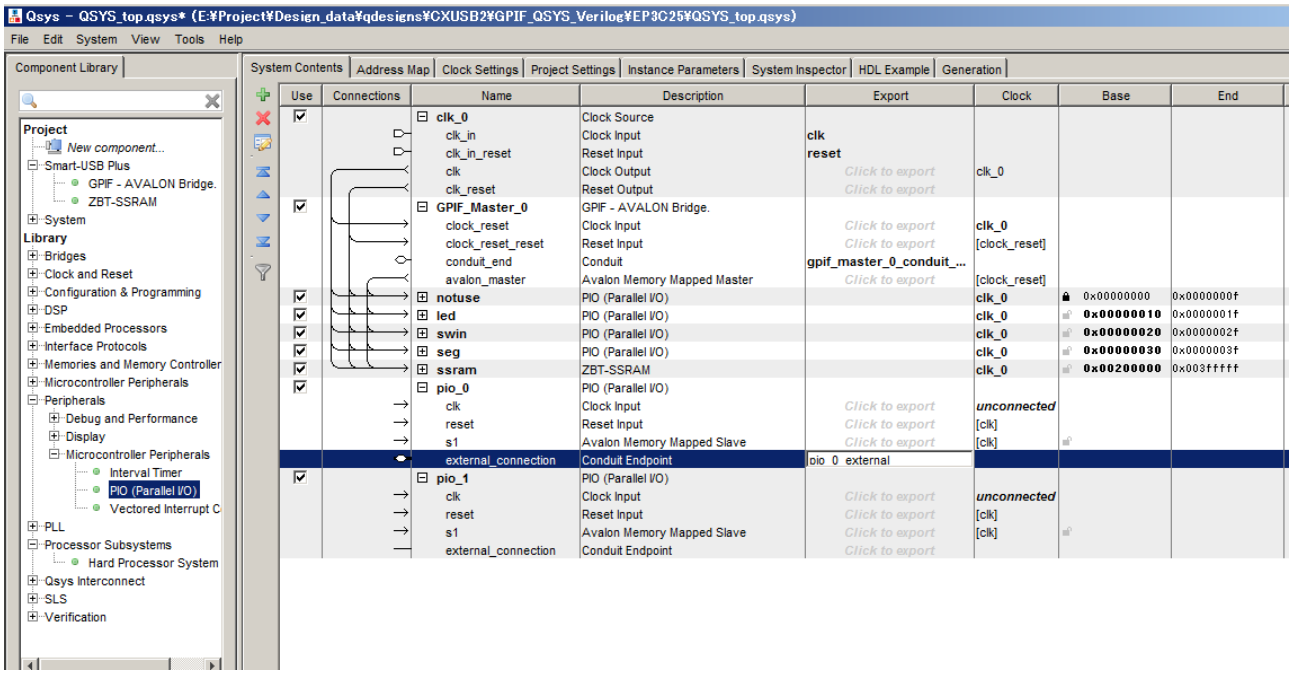
＜表 1. メモリマップ＞

このサンプル回路に 32bit 幅のレジスタを 2 個（入力専用と、出力専用）追加するには、Qsys "Component Library"欄の Library→Peripherals→Microcontroller Peripherals→PIO（Parallel I/O）を選択し Add ボタンを押してください。



PIO コンポーネンツの設定画面は左図の通りです。
Basic Settings 欄で、設定したいレジスタビット幅を指定します。この場合は 32 です。 入力専用にするには、Direction 欄で Input を選択します。 同様に出力専用にするには Output を選択します。

＜図 4. PIO 設定画面＞

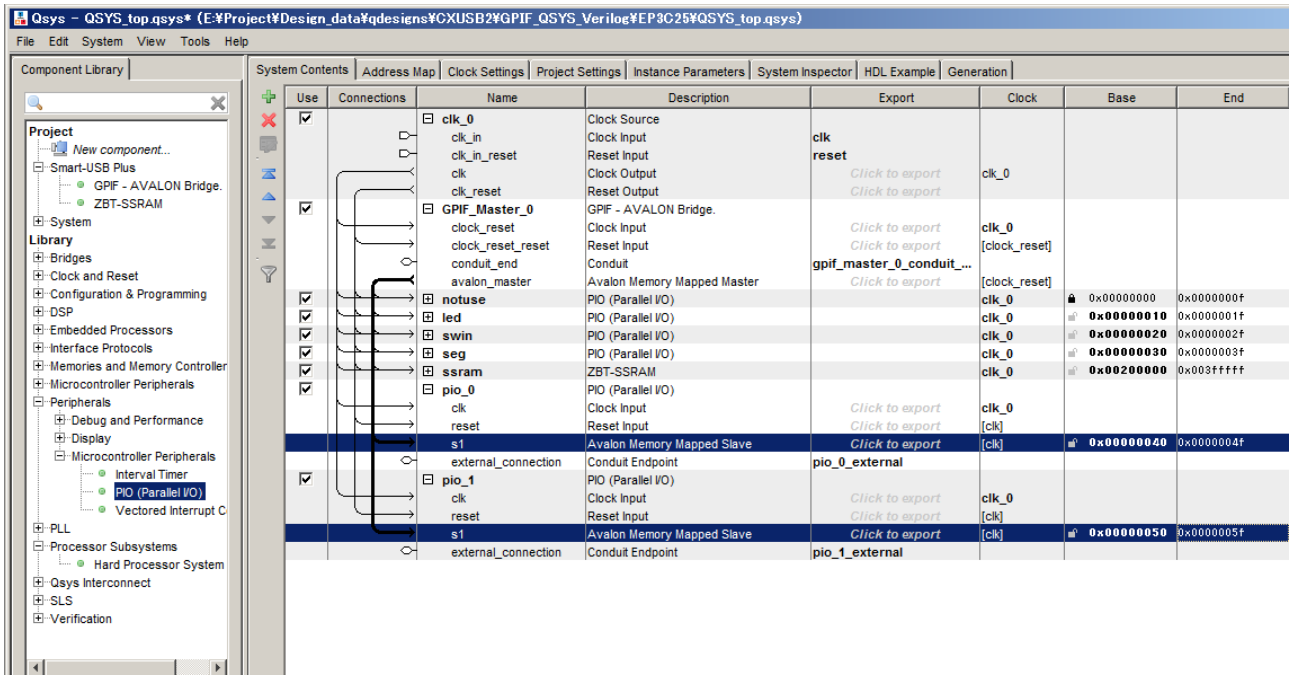


<図 5. PIO を 2 個追加した初期画面>

図 5 では GPIF_Master との接続やクロック、リセット系配線を行っていません。 各 PIO コンポーネントの”Clock Input”, ＂ Reset Input”, “Avalon Memory Mapped Slave” 3 カ所のポートにある、Connection 欄の白丸をクリックして黒丸にします。これで追加した 2 個の PIO が Avalon バスに接続されたことになります。

次に、追加した各 PIO の external_connection ポートの Export 欄をクリックします。ここで Avalon バスシステムと外部回路を接続するための信号を設定します。ここで指定する名称が 設計する Avalon バスシステムのピン名になります。 この例では、pio_0_external、pio_1_external としました。この時点では Avalon バスシステムが完成していないので、Qsys ツール上にはエラー表示が発生しています。

最後に、各 PIO のベースアドレスを ＂0x00000040”（PIO_0 出力専用） と ＂0x00000050”（PIO_1 入力専用） 設定します。



<図 6. Qsys 設定作業の完了>

コンポーネント名	ベースアドレス (Hex)	備考	RefApp7 のレジスタ操作タブから制御できるレジスタ番号(Dec)
unused	0x00000000	GPIF_Master のベースアドレスと同じ値に必ず設定	― 未使用
led（LED)	0x00000010	8bit 幅設定	4
Swin （スイッチ入力）	0x00000020	8bit 幅設定	8
Seg （7 セグ表示）	0x00000030	16bit 幅設定	12
pio_0	0x00000040	32bit 幅、出力専用	16
pio_1	0x00000050	32bit 幅、入力専用	20
ZBT-SSRAM （メモリアクセス）	0x00200000	FlowThrough モード設定	

<表 2. PIO を追加して完成したシステムの最終アドレスマップ>

【Avalon システムの生成】

Qsys の System Contents 画面でシステム構成が完了したら、Generation 画面に移動し、「Generate」ボタンをクリックしてください。エラーがなければ Qsys での作業は完了です。

以上で、Qsys を利用した Avalon システム設計が完了し、Avalon バスモジュールが完成しました。Avalon バスモジュールは、Q2 プロジェクトフォルダの ¥QSYS_top¥sysntehis¥QSYS_top.v です。

【Avalon システムモジュールをインスタンシエイト】

Qsys ツールの “HDL Example”タブをクリックすると、Q2 プロジェクトで Avalon システムをインスタンシエイトするための例が表示されます。Copy ボタンをクリックして Q2 プロジェクトで簡単に貼り付けることができます。 PIO を 2 個追加したので、ここでは、最下部に pio_0_external と pio_1_external が追加されています。

Q2 側では、トップモジュールの QSYS_Verilog.v に作成した Avalon モジュール QSYS_top.v を記述し、追加した 2 つの PIO ポート分を修正します。

```
QSYS_top    qsys1       (
                //              Common signals
                .clk_clk( pll_clk ),
                .reset_reset_n( rstn ),
                //              GPIF I/F
                .gpif_master_0_conduit_end_fd( fd ),
                .gpif_master_0_conduit_end_ctl({ 2'b11, rgdtn, cmdn, wrn, rdn}),
                .gpif_master_0_conduit_end_rdy( rdy_wire ),
                //              notuse
                //              QSYS Address = 0000_0000h - 0000_000Fh
                //              GPIF REGISTER Number = 0, 32bit Read Only
                .notuse_external_connection_export(),
                //              led
                //              QSYS Address = 0000_0010h - 0000_001Fh
                //              GPIF REGISTER Number = 4, 32bit Read/Write
                .led_external_connection_export( led_wire ),
                //              swin
                //              QSYS Address = 0000_0020h - 0000_002Fh
                //              GPIF REGISTER Number = 12, 32bit Read Only
                .swin_external_connection_export( ~{ psw, dipsw } ),
                //              seg
                //              QSYS Address = 0000_0000h - 0000_000Fh
                //              GPIF REGISTER Number = 16, 32bit Read/Write
                .seg_external_connection_export( seg_wire ),
                //              pio_0
                //              QSYS Address = 0000_0040h - 0000_004Fh
                //              GPIF REGISTER Number = 12, 32bit Read/Write
                .pio_0_external_export( pio_wire ),
                //              pio_1
                //              QSYS Address = 0000_0050h - 0000_005Fh
                //              GPIF REGISTER Number = 16, 32bit Read
                .pio_1_external_export( pio_wire ),
                //              QSYS Address = 0020_0000h - 002F_FFFFh
                //              2MiBytes
                .ssram_conduit_end_ab( ssram_ab ),
                .ssram_conduit_end_db( ssram_db ),
                .ssram_conduit_end_csn( ssram_csn ),
                .ssram_conduit_end_wn( ssram_wen ),
                .ssram_conduit_end_bwan( ssram_bwan ),
                .ssram_conduit_end_bwbn( ssram_bwbn ),
                .ssram_conduit_end_bwcn( ssram_bwcn ),
                .ssram_conduit_end_bwdn( ssram_bwdn ),
                .ssram_conduit_end_adv( ssram_adv ),
                .ssram_conduit_end_cken( ssram_cken ),
                .ssram_conduit_end_gn( ssram_oen ),
                .ssram_conduit_end_ftn(),                       //      notuse
                .ssram_conduit_end_lbon(),                      //      notuse
                .ssram_conduit_end_zz()                         //      notuse
                );
```

<Q2 プロジェクト トップモジュール GPIF_qsys.v での追加箇所>

上記ファイルで、赤字で記述した部分が追加した PIO レジスタ部分です。pio_0 に書き込んだデータを pio_1 で読み出せる様に pio_wire で接続しています。

【Q2 コンパイル】

Q2 プロジェクトのトップファイル GPIF_qsys.v で Qsys モジュールを記述後は、Q2 コンパイルを実行してください。サンプル回路では FPGA ピンアサインを実施ずみです。ピンを追加した場合は、適宜ピンアサインを行ってください。

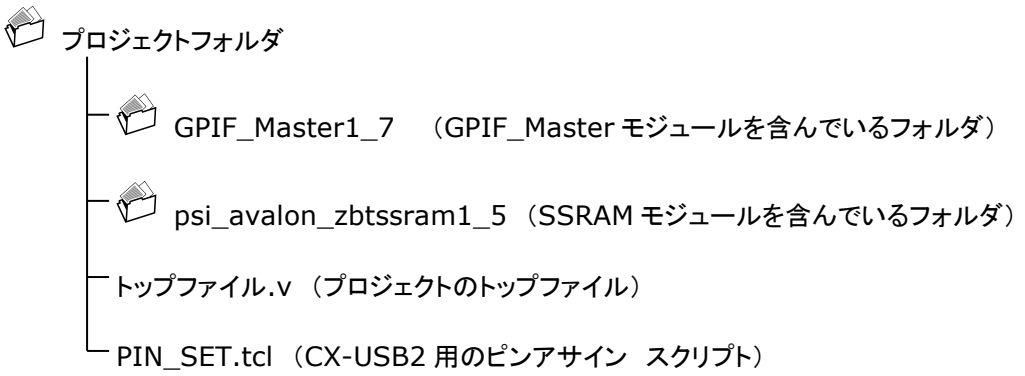
5. GPIF-AVALON ブリッジを利用した Q2 プロジェクトの作成方法

GPIF-AVALON ブリッジを含む Qsys プロジェクトを新規に生成する方法を解説します。

Qsys を起動するには Q2 プロジェクトを Open している必要があります。必要に応じてプロジェクトを作成してから操作を行ってください。

5. 1 GPIF-AVALON ブリッジを登録する

Q2 のプロジェクトを生成したフォルダに、GPIF-AVALON ブリッジや ZBT-SSRAM モジュールを含んだフォルダを配置してください。 生成したプロジェクトのフォルダ内にこれらのモジュールが存在すれば、ライブラリ設定などのパス設定は不要です。

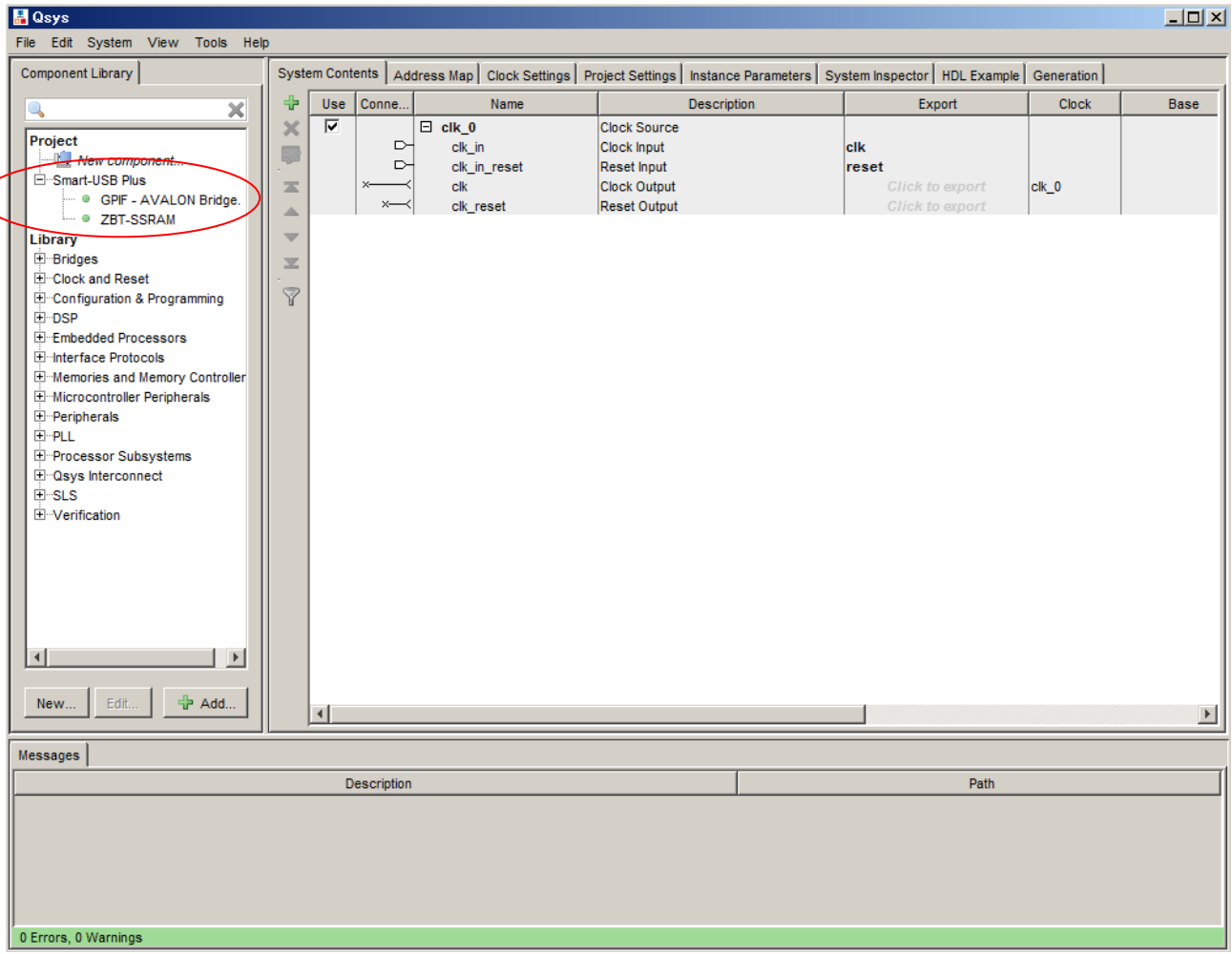


5. 2 Qsys を起動

Q2 プロジェクトを作成したら Qsys を起動します（Tools メニュー→Qsys）。 新規に作成したプロジェクトなので、まだ Qsys プロジェクト名が存在しません。ツールバーの「File」→「Save As…」を選択して Qsys プロジェクト名（ここでは Avalon.qsys）を付与します。 これは Qsys プロジェクト完成後（Generate 前）に行っても構いません。

（注意）Qsys で設計したデータは VeilogHDL コードです。Q2 プロジェクトを VHDL で設計する場合には、Qsys モジュールが VerilogHDL コードになっていることに注意してください。

Q2 プロジェクト内で Qsys を起動すると、下図の様に Peoject 下に使用できるモジュール名が表示されます。



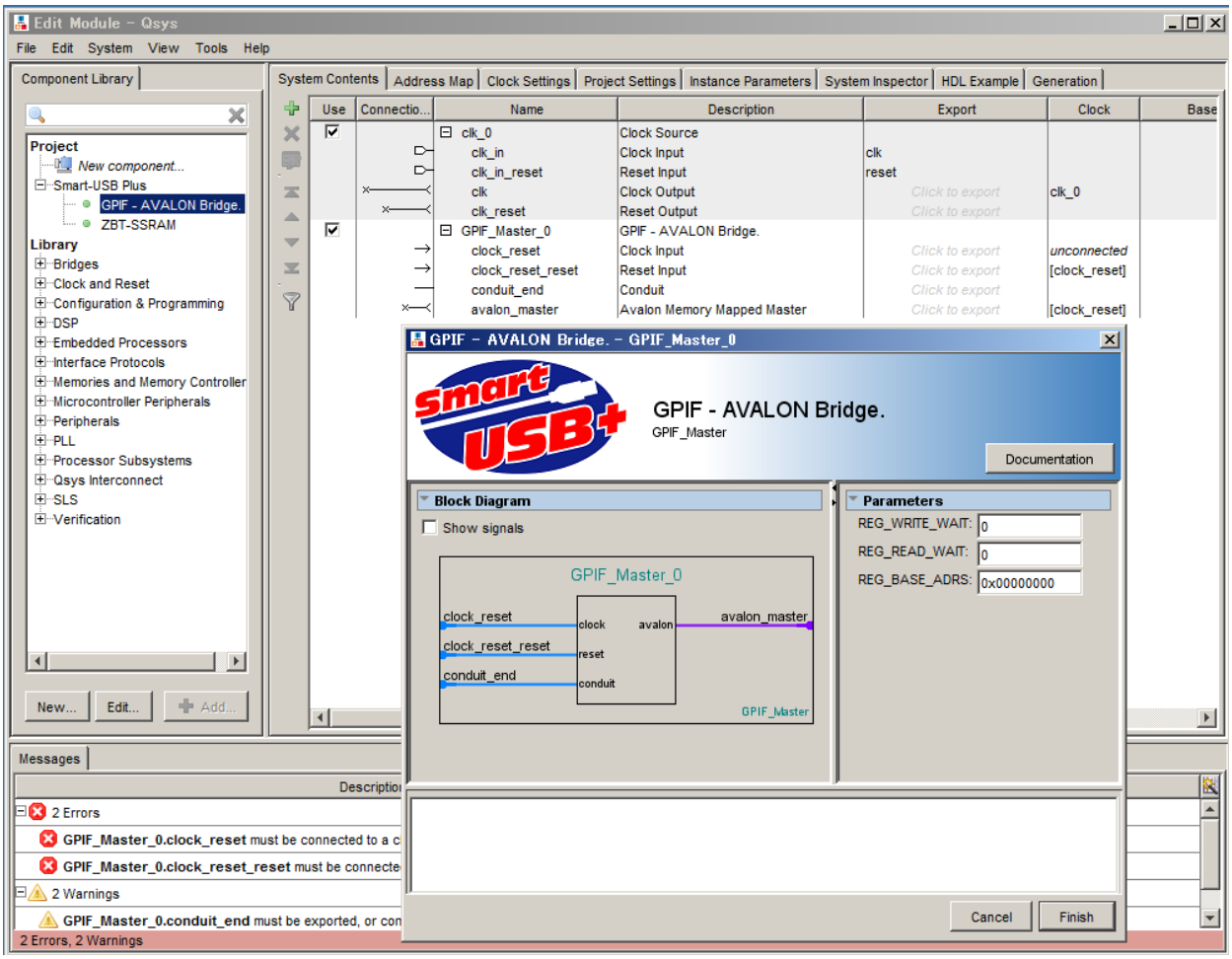
<図 7. Qsys 起動画面>

図 7 の様にモジュールが登録されていない場合は、設定した Q2 プロジェクト下に各モジュールのソースコードが存在していません。

ここまでの作業で、GPIF-AVALON ブリッジ回路を Qsys で利用することができます。

5.1 GPIF_Master を Qsys に登録する

Qsys 左側の「Component Library」タブから「Smart-USB Plus」グループを展開し、「GPIF-AVALON Bridge」をダブルクリックします。
ダイアログが開き、入力を促される「REG_WRITE_WAIT」「REG_READ_WAIT」「REG_BASE_ADRS」のパラメータは全てデフォルトのまま「Finish」をクリックします。 ※エラーが表示されますがここでは無視します。



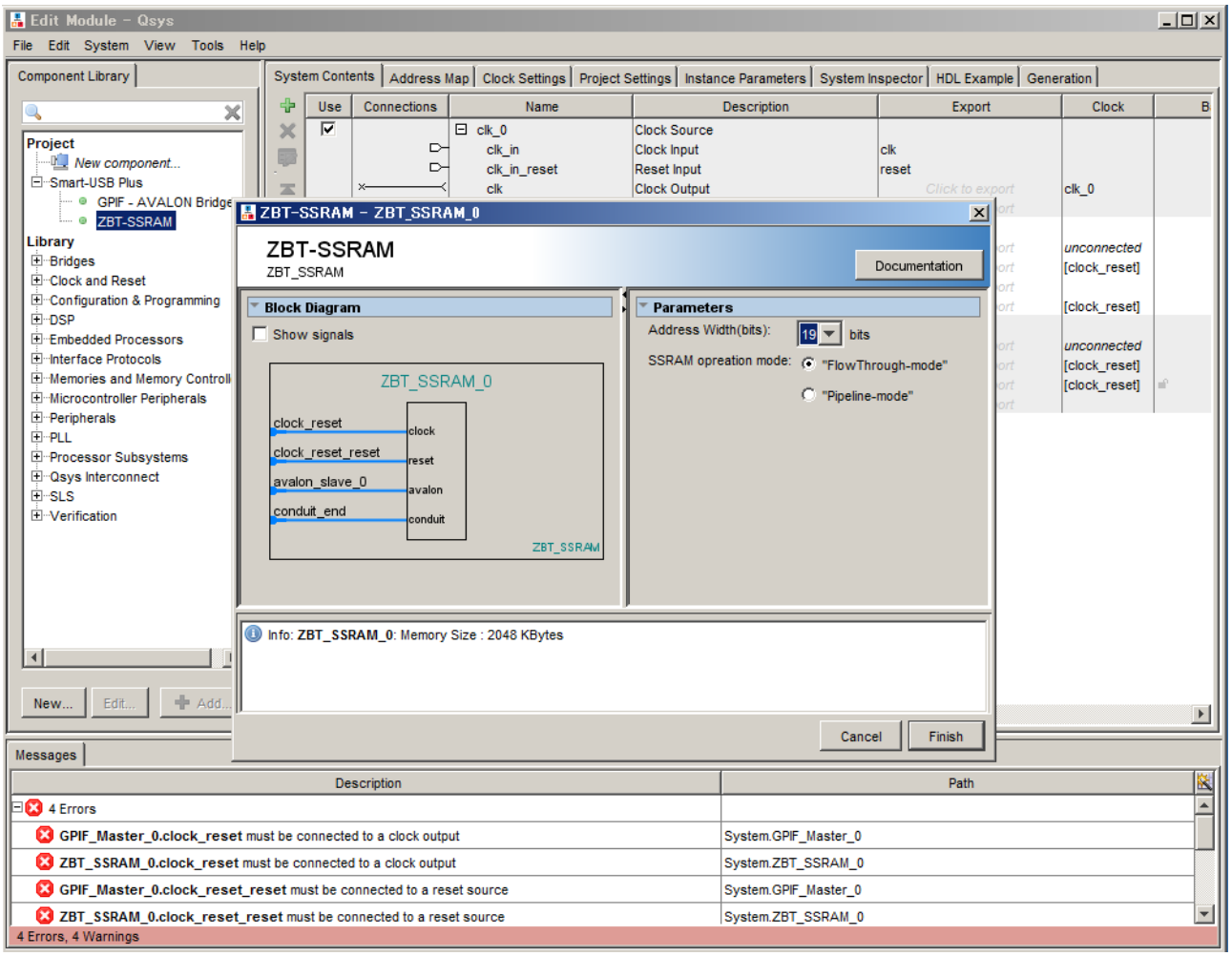
<図 8. GPIF_Master コンポーネントの設定画面>

5.2 SSRAM コンポーネントの登録

この例では CX-USB2 ボードをターゲットボードにしています。ボードには 1 個の 18Mbit SSRAM を搭載してるので、SSRAM のコンポーネントを1個登録します。

Qsys 左側の「Component Library」タブから「Smart-USB Plus」グループを展開し、「ZBT-SSRAM」をダブルクリックします。 下記のダイアログが表示されますので「Address Width」のドロップ・ダウンリストから「19」を選択、「SSRAM operation mode」はボード側の設定と一致する様に「Flow Through」または「Pipeline」を選択して「Finish」をクリックします。 （CX-USB2 製品出荷時、SSRAM 動作モードは「Flow Through」です。）

32bit(データバス)×512K Word(アドレス 19 本)=16Mbit です。



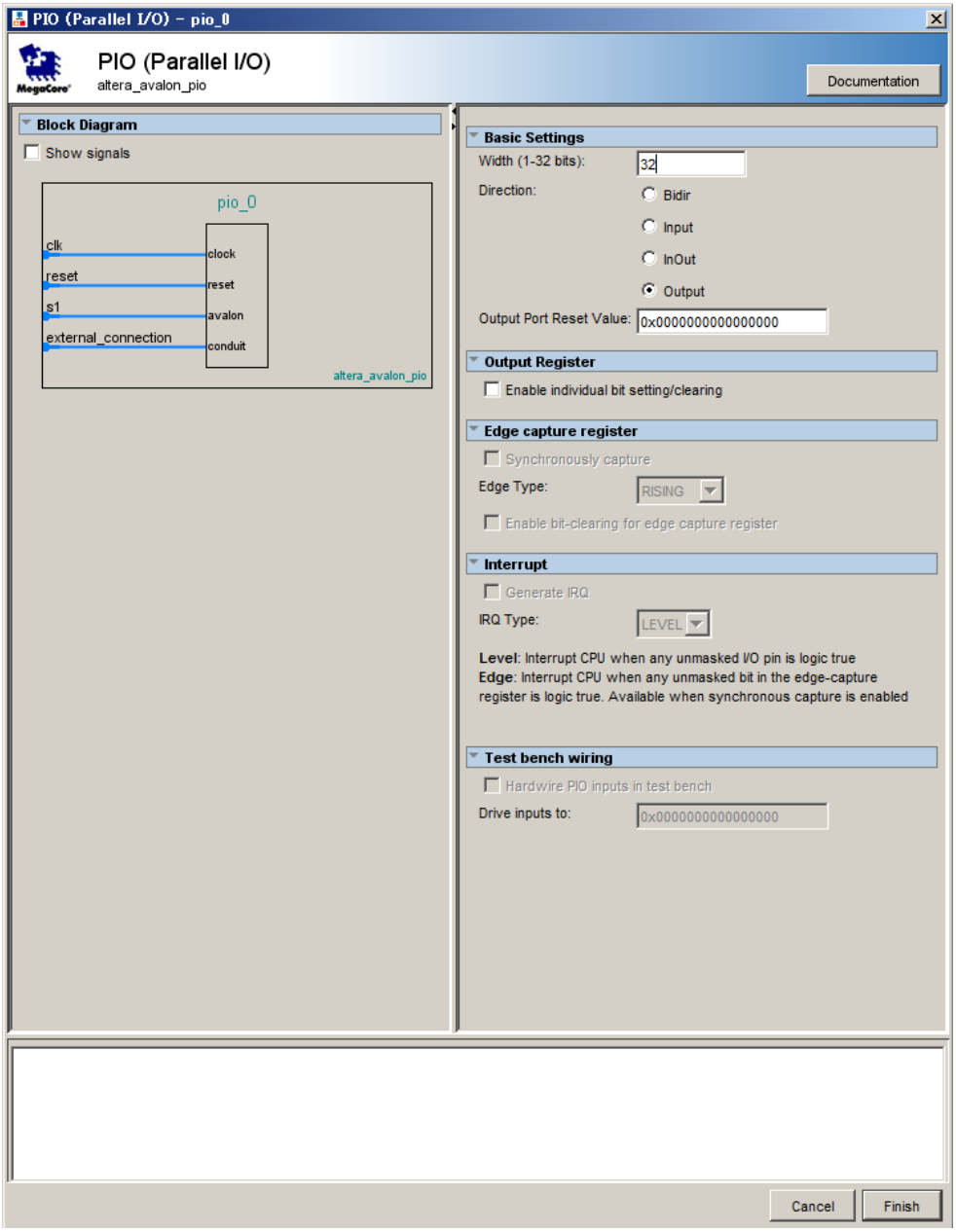
<図 9. SSRAM コンポーネントの設定画面>

SSRAM を 2 個搭載している様な SX-USB3 ボードを使用する場合は 2 個登録してください。ただし、GPIF_Master は 1 個しか登録できません。

5.3 ダミーレジスタの登録

REG_BASE+0 のアドレスは、ホスト PC から USB 経由でメモリデータを読み書きする時のベースアドレスとして「GPIF_Master」で予約しています。 このため、プロジェクトでユーザが使用しないレジスタを「ダミー」として配置することで、使用しないことを明示します。
(注)レジスタ"0"はメモリアクセスする際の先頭アドレス設定用のレジスタとしてシステム予約しています。

ダミーレジスタは ALTERA 製の標準 PIO である「PIO(Parallel I/O)」を使用します。
Qsys 画面左側の「Component Library」タブから「Library」→「Peripherals」→「Microcontroller Peripherals」グループを展開、「PIO(Parallel I/O)」をダブルクリックします。
図 11 で示すダイアログが表示されるので、「Width」を「32」bit、「Direction」を「Output ports only」に設定して「Finish」をクリックします。 Qsys 画面(System Contents タブ)の「Name」列に表示されるコンポーネントの名称は「Dummy」に変更すると、見やすい回路になります。



<図 10. ダミーレジスタの設定画面>

5.4 PIO(LED)の登録

8ビット汎用 LED 用の出力 PIO を登録します。 標準 PIO の「PIO(Parallel I/O)」を使用します。「ダミーレジスタの登録」と同じ手順です。
「Width」を「8」bit、「Direction」を「Output ports only」に設定して「Finish」をクリックしてください。 コンポーネントの名称は「led」に変更すると、見やすい回路になります。

5.5 7 セグメントの登録

7 セグメント LED コンポーネントを登録します。標準 PIO の「PIO(Parallel I/O)」を使用します。
「Width」を「16」bit、「Direction」を「Output ports only」に設定して「Finish」をクリックしてください。
コンポーネントの名称は「seg」に変更します。

5.6 スイッチ入力コンポーネントの登録

標準 PIO の「PIO(Parallel I/O)」を使用します。
「Width」を「8」bit、「Direction」を「Input ports only」に設定して「Finish」をクリックします。
コンポーネントの名称は「swin」に変更します。

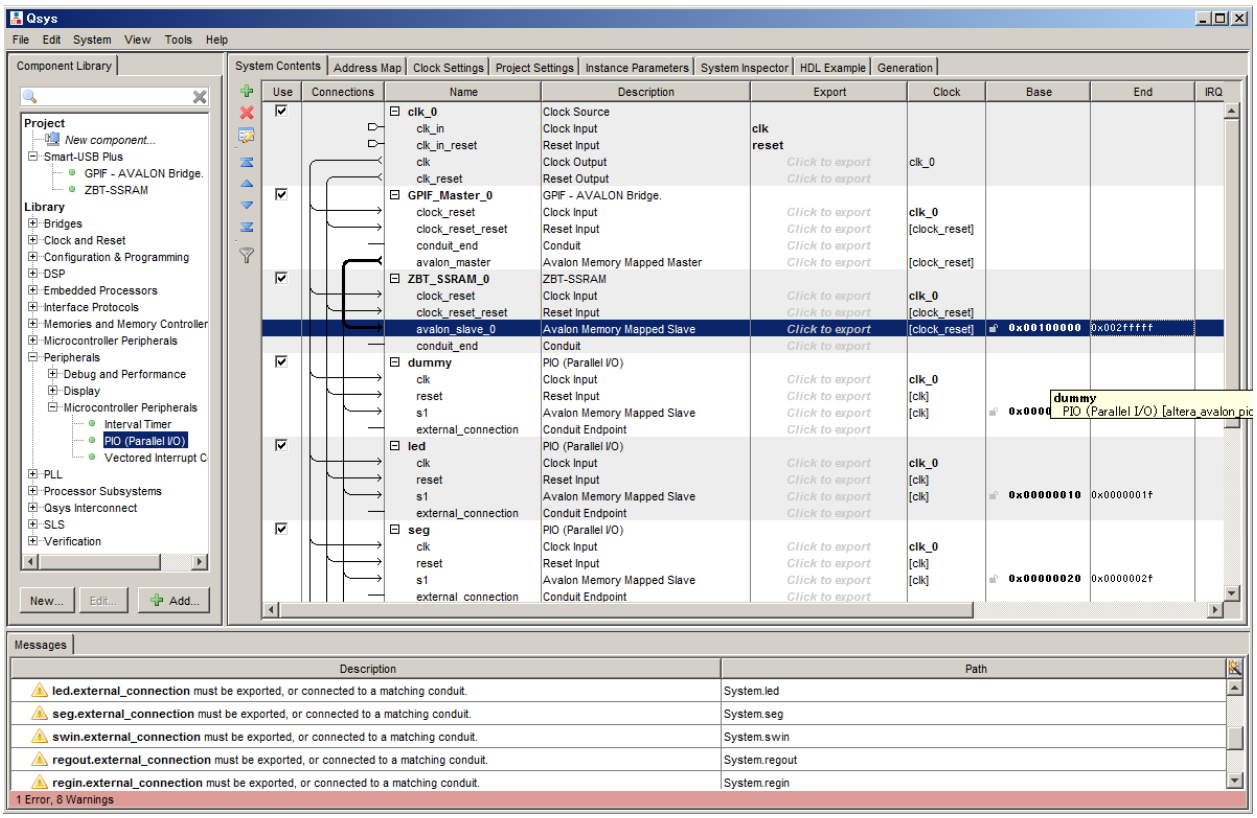
5.7 コンポーネント間の接続とベースアドレスの設定

各コンポーネントを登録したら、それぞれ必要に応じて GPIF_Master に接続します。ここで GPIF_Master に接続しなければ、ホスト PC からボード制御したときに、該当するコンポーネントへのアクセスができません。

また、各モジュールはクロックとリセット配線が必要です。Qsys 起動時にデフォルトで表示されている「Clock Source」コンポーネントの「Clock Output」、「Reset Output」に各モジュールの「Clock Input」と「Reset Input」を接続してください。Qsys ツール上では、白丸をクリックすることで黒丸に変化させ、各モジュールが接続したことになります。

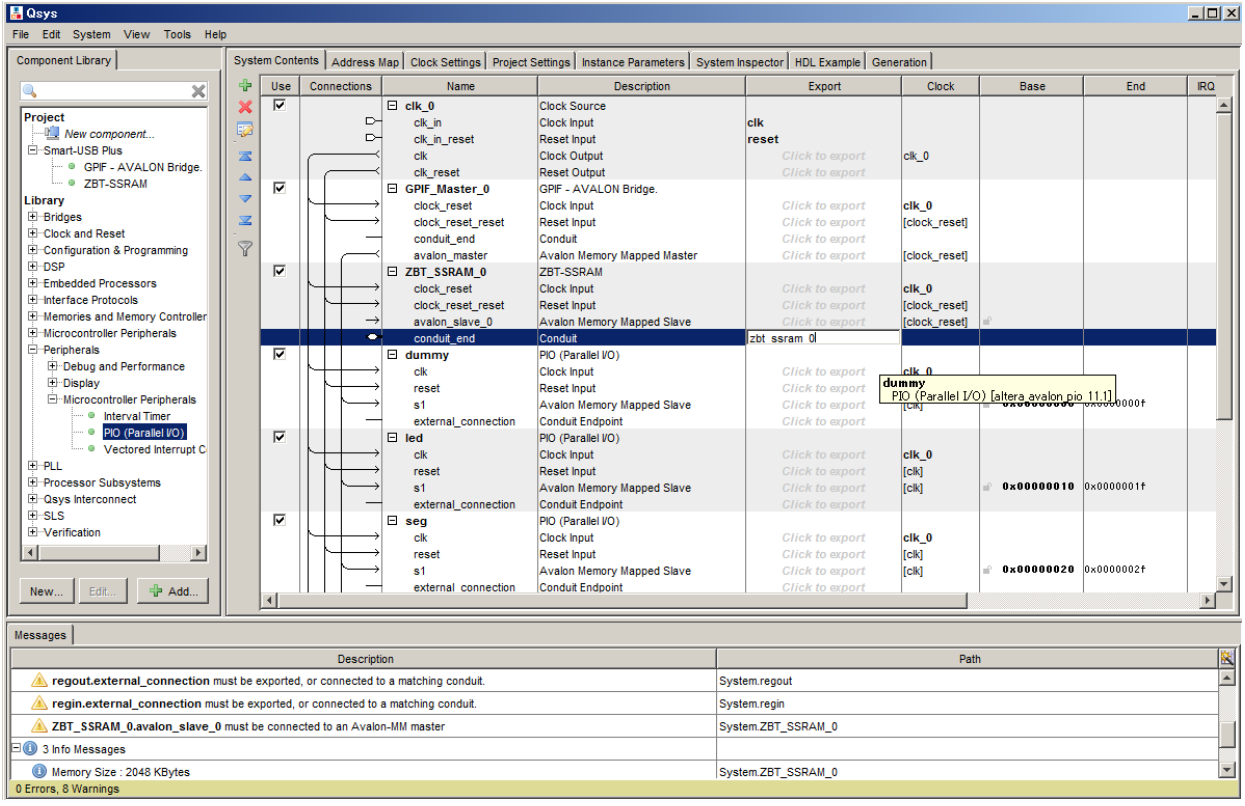
各モジュールのベースアドレス設定は、「System Contents」タブの「Description」列 「Avalon Memory Mapped Slave」行の Base 欄に記述します。

コンポーネント名	ビット幅	ベースアドレス (hex)	レジスタ番号 (Dec)
dummy	32	0x00000000	0
led（LED）	8	0x00000010	4
seg（7 セグ）	16	0x00000020	8
swin（スイッチ入力）	8	0x00000030	12
regout（レジスタ出力）	32	0x00000040	16
regin（入力専用レジスタ）	32	0x00000050	20
ZBT_SSRAM	32	0x00200000	



<図 11. 各モジュールのベースアドレス設定>

「System Contents」タブの Export 列には、Avalon バスシステムと外部システムとを接続するための信号ポートを表示できます。「*Click to export*」欄をクリックすると、信号ポートを表示できます。
「Description」列で 「Conduit」の行をクリックしてください。 名称は自由に変えられます。



<図 12. Avalon バスシステムと外部回路の接続ポート設定>

5.8 Avalon バスシステムの生成

「Clock Settings」タブを選択し、設計する Avalon バスシステムが動作するクロックを設定します。 ホスト PC と USB インタフェースするために、48MHz を指定してください。

次に「HDL Example」をクリックし、設計した Avalon バスシステムのインスタンス・テンプレートを表示してください。画面の「Copy」ボタンをクリックすれば、Q2 プロジェクトのトップ回路に簡単に貼り付けることができます。

合成可能な Avalon バスシステムは「Generation」タブをクリックして「Generate」ボタンをクリックしてください。エラー無く処理が完了すれば、Qsys での作業は完了です。

Avalon バスシステムは VerilogHDL で記述されています。VHDL で Q2 プロジェクトを記述していても、実際に合成可能な Avalon バスシステムは VerilogHDL になります。 VHDL 記述している場合でも、Avalon バスシステムをインスタンシエイトする場合に、「HDL Example」画面で HDL language を VHDL に設定することにより VHDL 記述のインスタンスを得ることができます。 また、VHDL でシミュレーションする場合には、「Generate」画面で VHDL のシミュレーションモデルを作成できます。

この例で Generate 後に生成された Qsys プロジェクトは、Avalon.v として Generate 画面で設定されたパスに保存されています。同じフォルダには Avalon.qip ファイルがあります。 Q2 プロジェクトでコンパイルする前に、この qip ファイルをプロジェクトに追加してください。 プロジェクトへの追加は、Q2 ツールバー「Project」→「Add/Remode Files in Project…」を選択し、作成した qip ファイルを選択してください。

5.9 Q2 コンパイル

Q2 プロジェクトのトップファイルに作成した Avalon バスモジュールをインスタンシエイトして、ピンアサイン等の設定を行った後、コンパイルして xxx.rbf ファイルを生成してください。

6. ホスト PC からの制御方法

ホスト PC のボード制御アプリケーション(例えば RefApp7.exe)から、作成した Avalon バスシステムにアクセスすることができます。 ただし、Qsys ツール上で設定したベースアドレスが間違っていると、ボード制御アプリケーションからレジスタアクセスしてもボードは動作しません。

LED のコンポーネントを、スタート・アドレス＝0x00000010、 エンド・アドレス＝0x0000001F に設定した場合、このアドレスにボード制御アプリケーションからアクセスするためのレジスタ番号は、"4"～"7"になります。 LED コンポーネントは 8bit なので、実際に使用するアドレスは 0x00000010 だけです。レジスタ4を 32bit でアクセスすれば、アドレス空間"0x00000010～0x00000013"をアクセスしていることになります。

0x00000014～0x00000017 のアドレス空間に 8bit アクセスする場合は、レジスタ番号"5"になりますが、サンプル回路では何もアサインされていないので、動作しません。

以下、同様に、32bit のレジスタとして登録した regin レジスタのベースアドレスは 0x00000050 なので、レジスタ番号は "20" になります。 ホスト PC の制御アプリケーションから 32bit 幅でアクセスすると、アドレス空間"0x00000050～0x00000053"のデータを読み出すことができます。

(注)ここで、アドレス表示は 16 進表示、レジスタ番号は 10 進表示です。

Qsys ツールを利用して、バスマスタに GPIF-AVALON ブリッジを使用した場合、PC 上の制御アプリケーションでは、次のように設定して制御できます。

6.1 レジスタアクセス

レジスタ番号と AVALON バス上のアドレスは、下表の通り「レジスタベースアドレス(hex)」＋「レジスタ番号（hex）*0x04」で示します。

レジスタ No. (dec)	AVALON バス上のアドレス (hex)
レジスタ 0	レジスタベースアドレス＋0x00
レジスタ 1	レジスタベースアドレス＋0x04
レジスタ 2	レジスタベースアドレス＋0x08
レジスタ 3	レジスタベースアドレス＋0x0C
レジスタ 4	**レジスタベースアドレス＋0x10**
.....	.....
レジスタ 8	**レジスタベースアドレス＋0x20**
.....	.....
レジスタ 40	**レジスタベースアドレス＋0xA0**
.....	.....

レジスタベースアドレスは、コンポーネントの登録で設定した値です。

6.2 レジスタ長

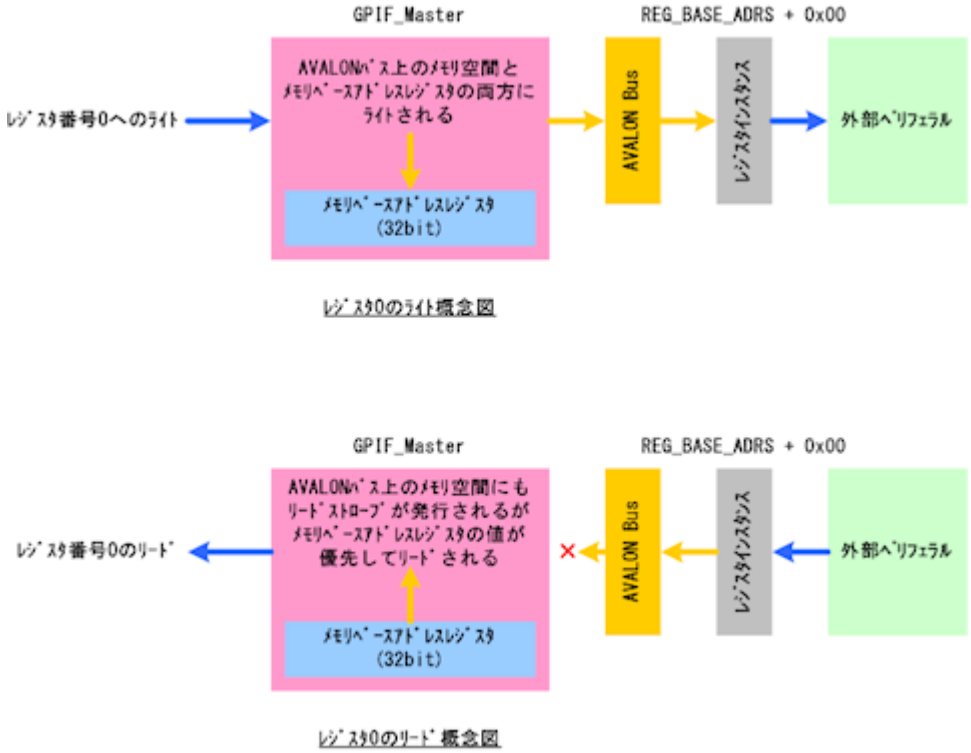
8/16/32 ビット・アクセスのみ可能です。 64 ビットアクセスは 32 ビットアクセスとして扱われます。
GPIF_Master は 32 ビットのデータ幅固定ですので常に 32 ビットアクセスを行ってください。
ただし、ペリフェラル側で 8/16bit のデータ幅しか無い場合には 32 ビットアクセスではなく、ペリフェラル側のデータ幅に合わせても問題ありません
各ビット幅でアクセスした場合の「byteenable」信号は下表の通り出力します。

レジスタ長	byteenable[3]	byteenable[2]	byteenable[1]	byteenable[0]
8bit アクセス	0	0	0	1
16bit アクセス	0	0	1	1
32bit アクセス	1	1	1	1

6.3 メモリ・アクセス

メモリ・アクセスは高速で大容量のデータ転送を行う時に使用する転送方法です。 GPIF-AVALON ブリッジ回路により AVALON バスのプロトコルに変換された後では、メモリ・アクセスとレジスタ・アクセスの本質的な違いはありません。
（※レジスタベースアドレスと物理メモリのアドレスを重複させることにより、低速になりますが、レジスタアクセスで物理メモリにアクセスする事も可能です。）

メモリ・アクセスは、GPIF-AVALON ブリッジ・マスタ回路内にあるメモリベースアドレスレジスタを転送開始時のスタート・アドレスとして使用します。



＜図 13. 制御アプリケーションからのレジスタ“0”アクセス概要＞

注意点は以下 3 点です。

1. レジスタ番号"0" へのライト時は「GPIF-AVALON ブリッジ・マスタ回路内のメモリベースアドレスレジスタ」と「AVALON バスにあるレジスタベースアドレス+0x00」の両ペリフェラルにライトを行います。

2. レジスタ番号"0" のリード時は「GPIF-AVALON ブリッジ・マスタ回路内のメモリベースアドレスレジスタ」が優先してリードされます。この際「AVALON バスにあるレジスタベースアドレス+0x00」にもリードストローブを発行しますが、リードデータは破棄します。

3. メモリベースアドレスレジスタの下位 2 ビットは「0」に固定になります。

（注意）レジスタ No.0 は、メモリ・アクセスのための先頭アドレスを指定するレジスタなので、ユーザが汎用的に利用できません。このために、Avalon バスにダミーレジスタとして レジスタ”0”を接続しています。

6.4 メモリ操作の手順

ホスト PC の制御アプリケーションからメモリ転送を行う手順を説明します。
CX-USB2 システム開発ボードのサンプル回路と同じマップ(下表)でメモリが配置されている場合です。

名称	先頭アドレス	終了アドレス
ZBT-SSRAM	0x00200000	0x003fffff

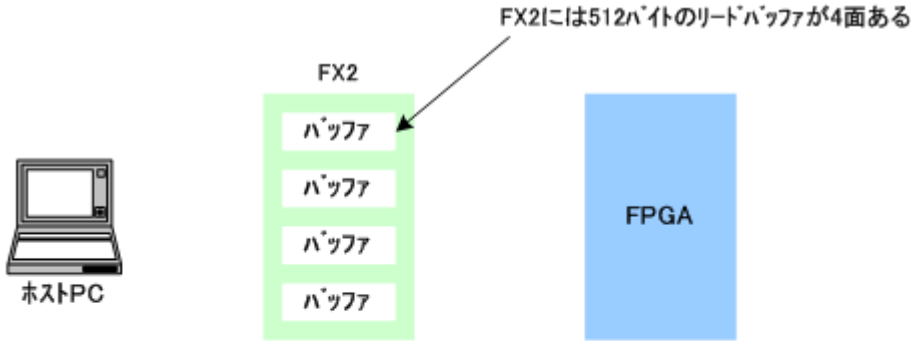
1. 転送レングスを「0x00200000」に設定する（2MB）。

2. SSRAM にアクセスする場合には、レジスタ番号"0"に 32 ビット長で「0x00200000」を書き込んでからメモリ転送を行う。
RefApp7 を利用する場合には、「メモリ操作」タブの 「転送開始アドレス（レジスタ 0）」に Qsys で設定したメモリ・ベースアドレスを設定します。

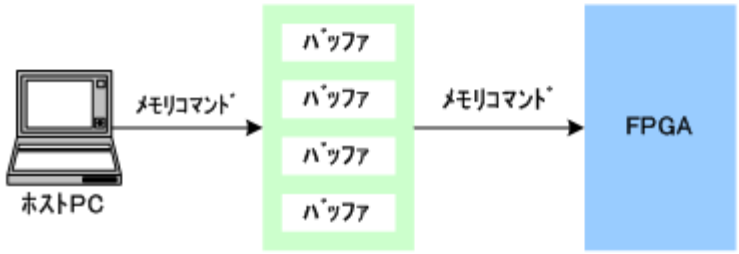
6.5 メモリ転送時の総転送バイト数

メモリリードの場合、ホスト PC 上の制御アプリケーションで指定したメモリリードのデータ量と、実際に GPIF-AVALON ブリッジ・マスタがペリフェラルから**読み出すデータ量が異なります。**
これは、USB コントローラ(FX2)と GPIF-AVALON ブリッジ・マスタ内に各々2,048 バイトの FIFO を持っているためです。

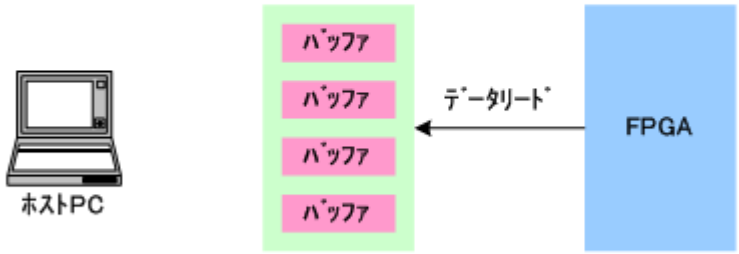
ホスト PC がメモリリードを開始すると FX2 と GPIF-AVALON ブリッジ・マスタ内の FIFO は、ホスト PC から USB のリードパケット(データ転送要求)を受信後、即座にリードデータをホスト PC に出力するため、FIFO を常に Full の状態にする様にペリフェラルからデータをリードしておきます。



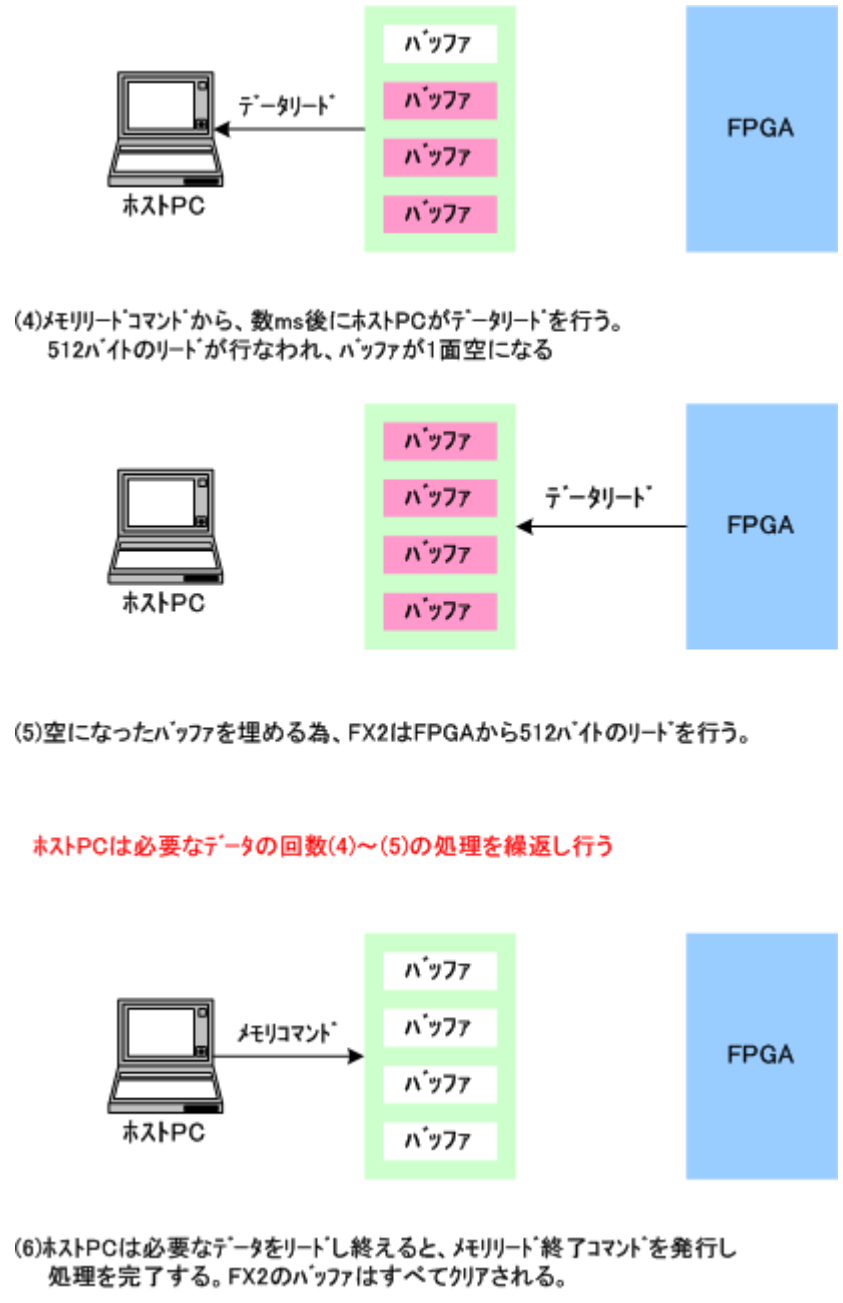
(1)ﾒﾓﾘ転送開始前、FX2のﾘｰﾄﾞﾊﾞｯﾌｧはすべて空の状態である。



(2)FX2は、ﾎｽﾄPCからのﾒﾓﾘﾘｰﾄﾞ開始ｺﾏﾝﾄﾞを受け取ると、
FPGAにﾒﾓﾘﾘｰﾄﾞ開始ｺﾏﾝﾄﾞを発行する。



(3)FX2は、FPGAから2048ﾊﾞｲﾄ(512×4面)のﾃﾞｰﾀをﾘｰﾄﾞして、
内蔵ﾊﾞｯﾌｧをﾒﾓﾘﾃﾞｰﾀで埋める。



<図 14. メモリ RD 時の注意点>

このため、**「ホスト PC が読み出すデータ量 = メモリリードで指定したデータ量 + 4096 バイト」**となります。
また、余分にリードした FX2 内の 4096 バイトは、メモリリードが完了した時点で破棄してしまいますので注意して下さい。

FPGA から読み出すメモリ構成が、リードクリア方式や FIFO などの場合に、不具合が発生する可能性が考えられます。 この場合、FX2 からのメモリ終了コマンドを待たずに、FPGA が FX2 からメモリを読み出させないようにする必要があります。 制御アプリケーションで設定するメモリ転送レングス情報を FPGA 内部のレジスタに格納し、FX2 が出力する RDn 信号と RGDTn 信号を FPGA 内でカウントして、メモリ転送レングス設定値と一致させ、設定したデータ転送量をハードウエアで検出することができます。

FX2 から読み出されるデータ量が、設定したメモリ転送レングス値に一致した時点で、FPGA からのデータ送出を停止し、正しいデータを消失させないようにすることができます。

7. Avalon バスに外部データを取り込む方法

ここでは FPGA に入力されるパラレルデータを Avalon バス経由でホスト PC に収集する方法を示します。
システム開発ボード上に同期 SRAM（SSRAM）を搭載する製品の場合、SRAM-FIFO 回路が利用できますが、SSRAM が無いボードや、数 MB/s 以下の低速なデータ収集の場合には、FPG 内蔵メモリ領域を利用してデータ収集することができます。

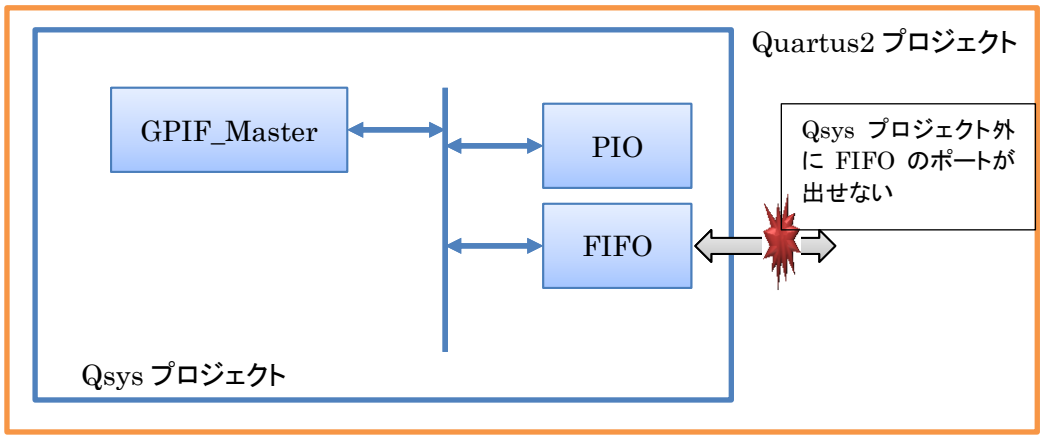
SRAM-FIFO 回路では、FPGA 外部の SSRAM をバッファメモリとして利用できるので、高速なデータを連続して PC に収集したり、PC からデータを書き出すことができます。 アプリケーションノート SUA007 を参照してください。

大規模なバッファメモリを利用できない場合には、FPGA 内蔵メモリをデュアルポート FIFO（DP-FIFO）化して利用します。このために、Avalon バスに“Generic Tri-State Controller”モジュールを追加して対応します。

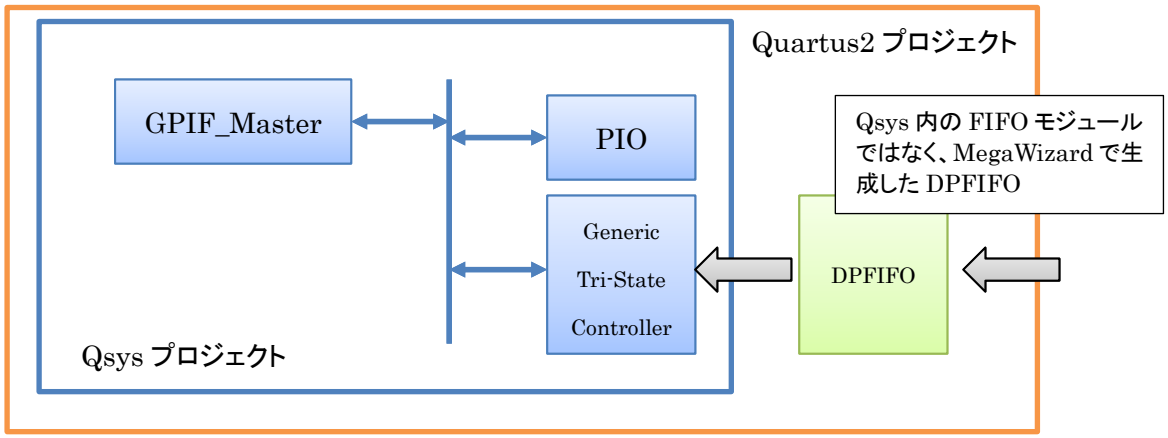
7.1 概要

Quartus2 の MegaWizard Plug-In Manager を起動し、デュアルポート FIFO モジュールを作成します。このモジュールは直接 Avalon バスに接続できません。このため、Avalon バスには Generic Tri-State Controller を追加し、このモジュールに生成した FIFO を接続します。

Qsys ツール上でも FIFO モジュールを接続できますが、RD ポート／WR ポートともに Avalon バスに接続するので、FPGA 外部からのデータを FIFO に入力できません。



<図 15. FIFO モジュールの接続方法（Avalon バス内）>

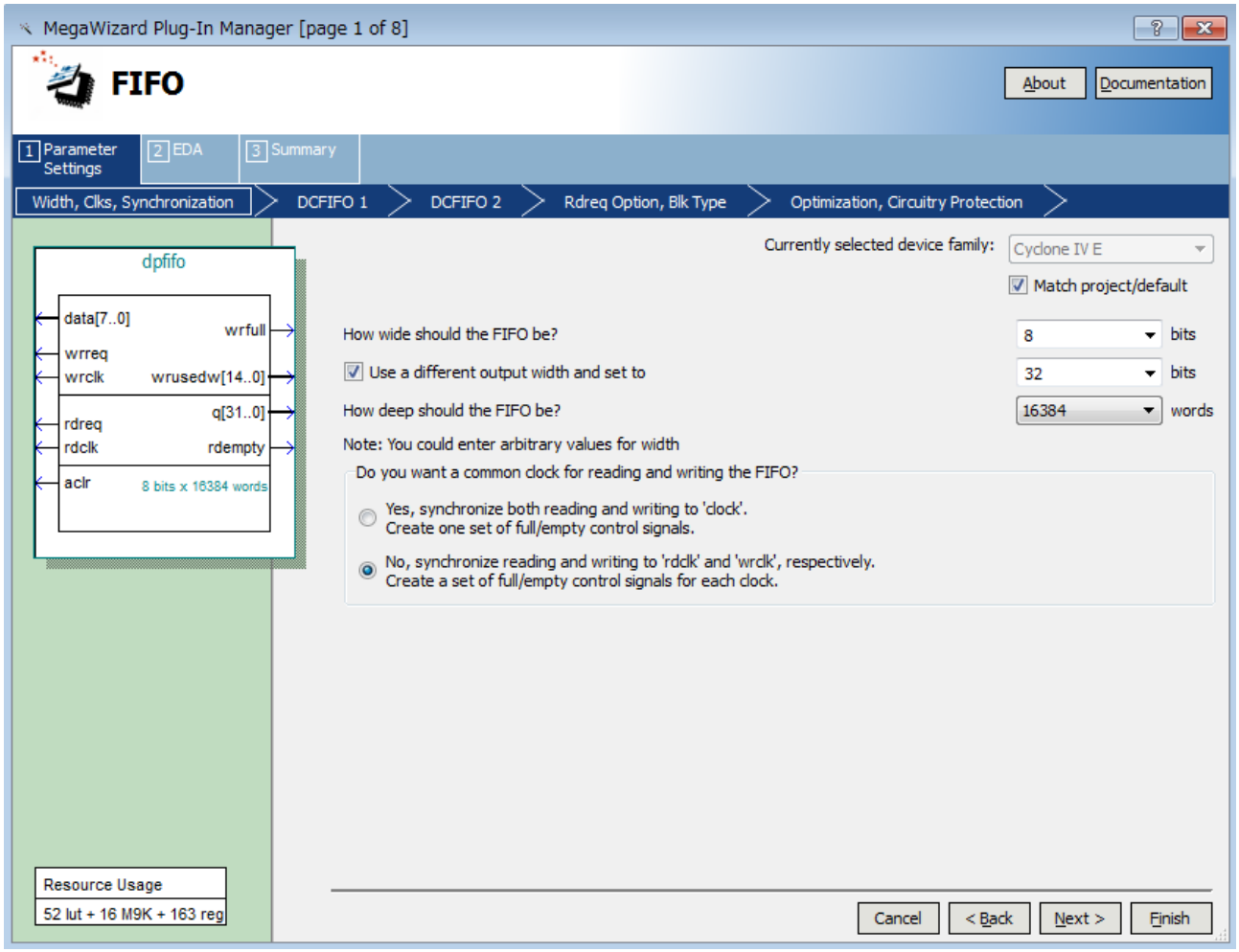


<図 16. FIFO モジュールの接続方法（Avalon バス外）>

7.2 FIFO モジュールの生成

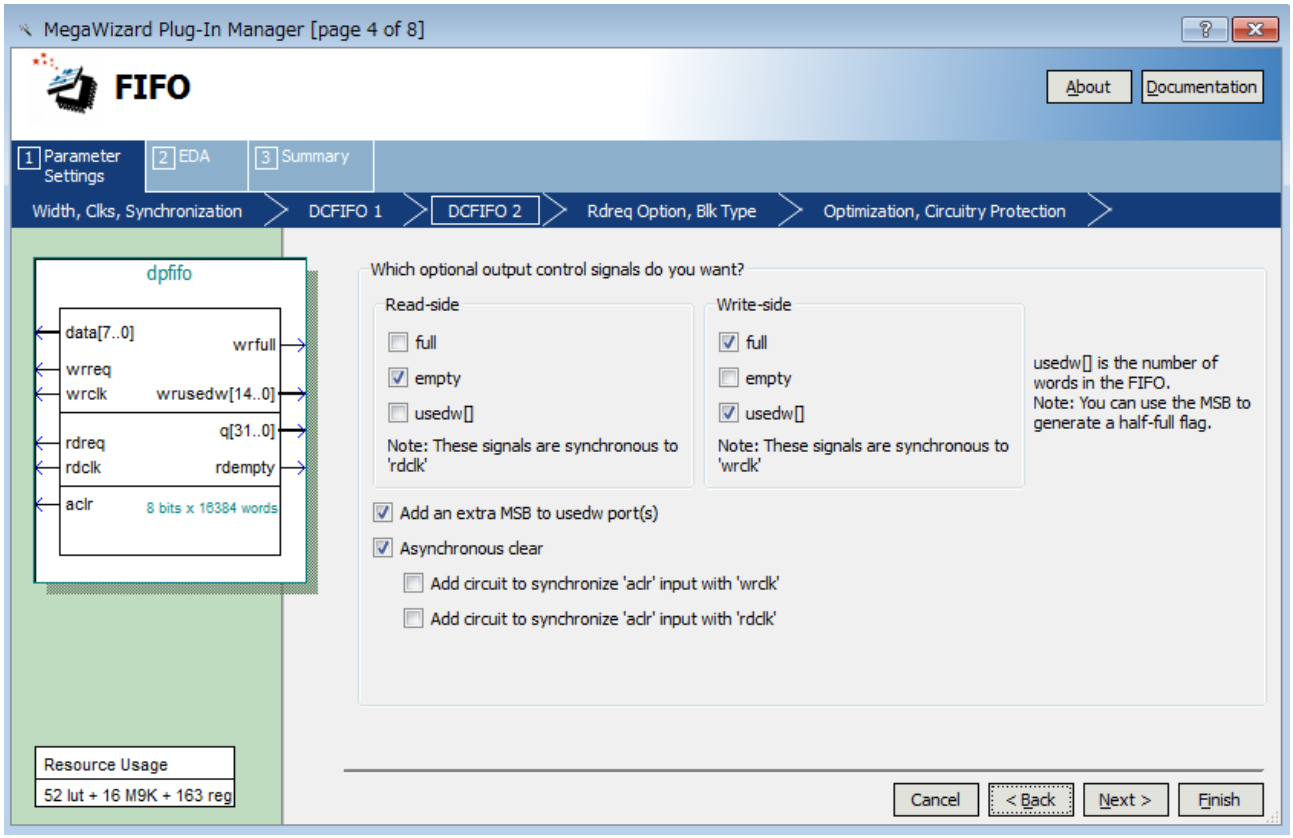
Quartus2 の MegaWizard Plug-In Manager から“FIFO”を選択し、デュアルポート FIFO になるように各種パラメータを設定します。

データを収集するので、Avalon バスと接続する読出しポートは 32bit データ幅になります。 FIFO へのデータ書込み側は 1bit～256bit まで任意に設定できます。 図 17 の例では書込み側データ幅は 8bit です。



<図 17. FIFO モジュール設定画面 （MegaWizard）>

図 17 では、FIFO の深さを 16K バイトとしました。「6.5 メモリ転送時の総転送バイト数」で示したように、FPGA とインタフェースする USB 制御 IC（FX2）には 2K バイトのバッファがあるので、最低 2K バイト以上の FIFO が必要です。この FIFO 容量が大きいほど、実際に収集するデータの転送レートが向上します。



<図 18. FIFO ステータス信号の生成>

図 18 では、FIFO ステータス信号を選択します。Read-side では、必ず empty 信号を生成し、Write-side では full 信号を生成してください。残りのステータス信号は任意です。

図 17、図 18 で示したパラメータ以外は、ツールデフォルト設定です。

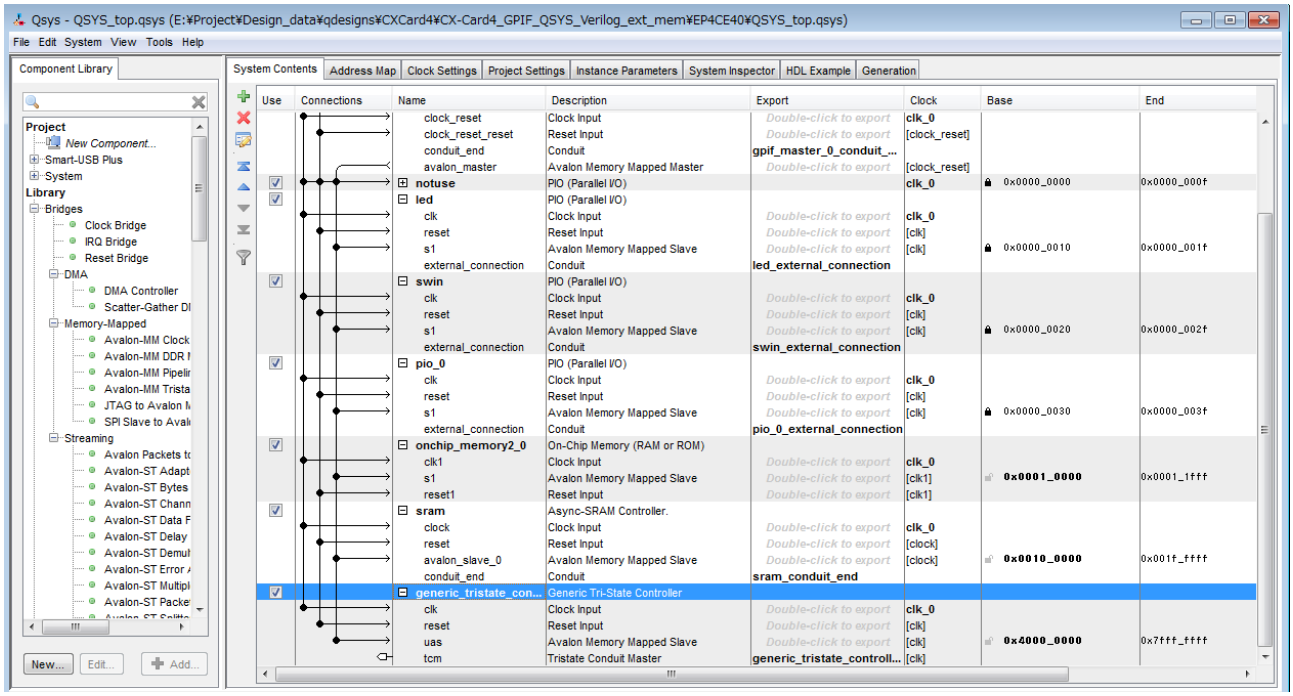
7.3 Generic Tri-State Controller の生成

Qsys ツールの Component Library から、“Qsys Interconnect”→“Tri-State Components”→“Generic Tri-State Controller”を選択し、“System Contents”画面で接続してください。

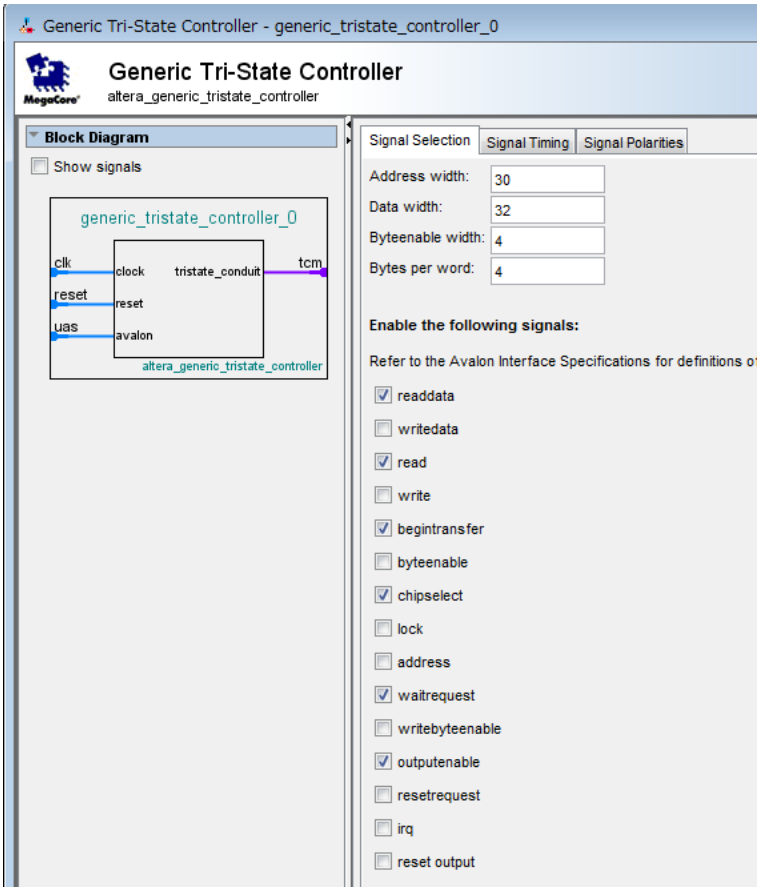
図 19 の例では、Avalon バススレーブモジュールとして“Generic Tri-State Controller”を Avalon バスシステムに追加しています。ベースアドレスは 0x4000_0000 です。
また、Avalon バスシステム外の DP-FIFO と接続するために、Tri-State Conduit Master 信号を有効にして Export します。

図 19 で示す Qsys 画面で、Generic Tri-State Controller 部分をダブルクリックすると、各パラメータ設定ができます。 生成した DP-FIFO と接続するために必要な信号を選択します（図 20）。 ここでは読み出し専用なので、read に関連する信号線のみアクティブにしています。

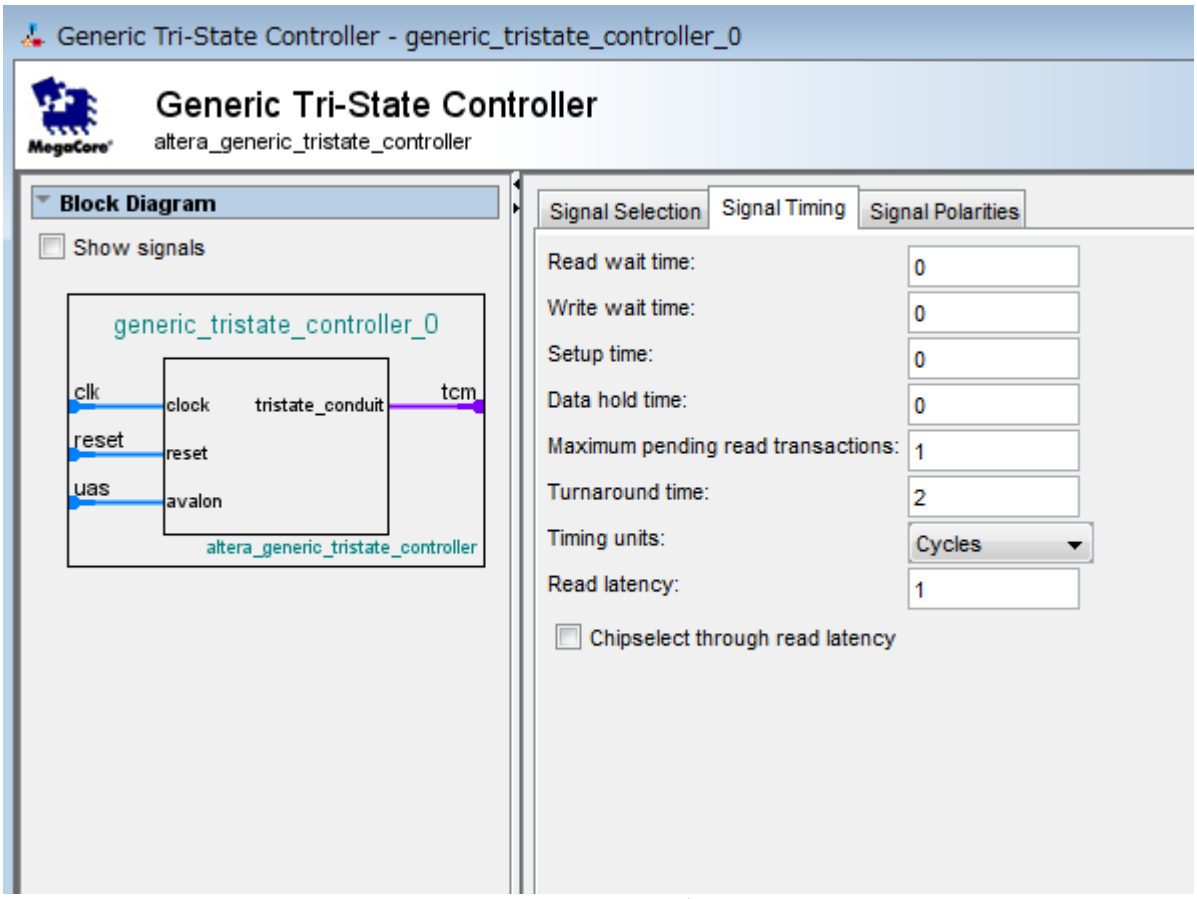
readdata	DP-FIFO の Rd-saide と接続する 32bit データ
read	読み出しイネーブル、DP-FIFO の rdreq 信号と接続
chipselect	Generic Tri-State Controller を選択したときにアクティブ
waitrequest	FIFO の empty 信号と接続



<図 19. Qsys トップ画面での Generic Tri-State Controller モジュールの追加>



<図 20. Generic Tri-State Controller のパラメータ設定>

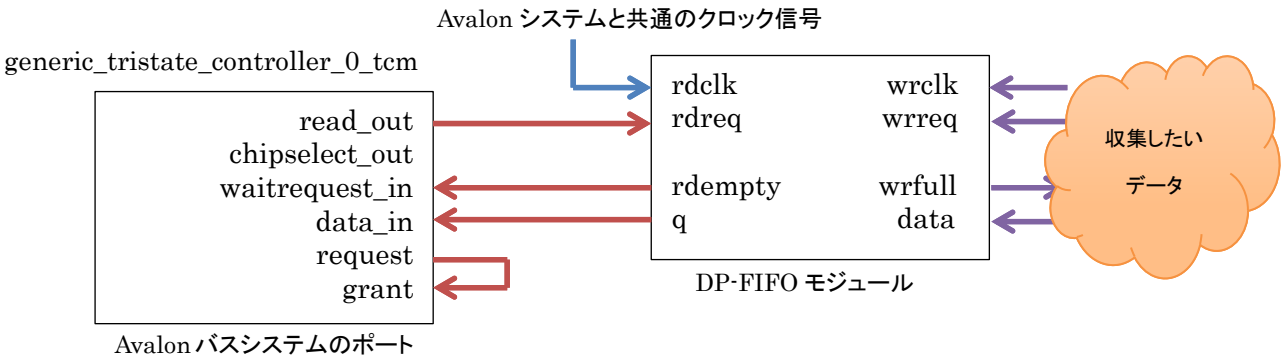


<図 21. 信号タイミング設定>

図 21 で示すように信号タイミングを調整して Generic Tri-State Controller と DP-FIFO を接続する準備が整いました。

7.4 Generic Tri-State Controller と DP-FIFO の接続

生成した Qsys プロジェクトと DP-FIFO を Quartus2 プロジェクトのトップ階層で接続します。

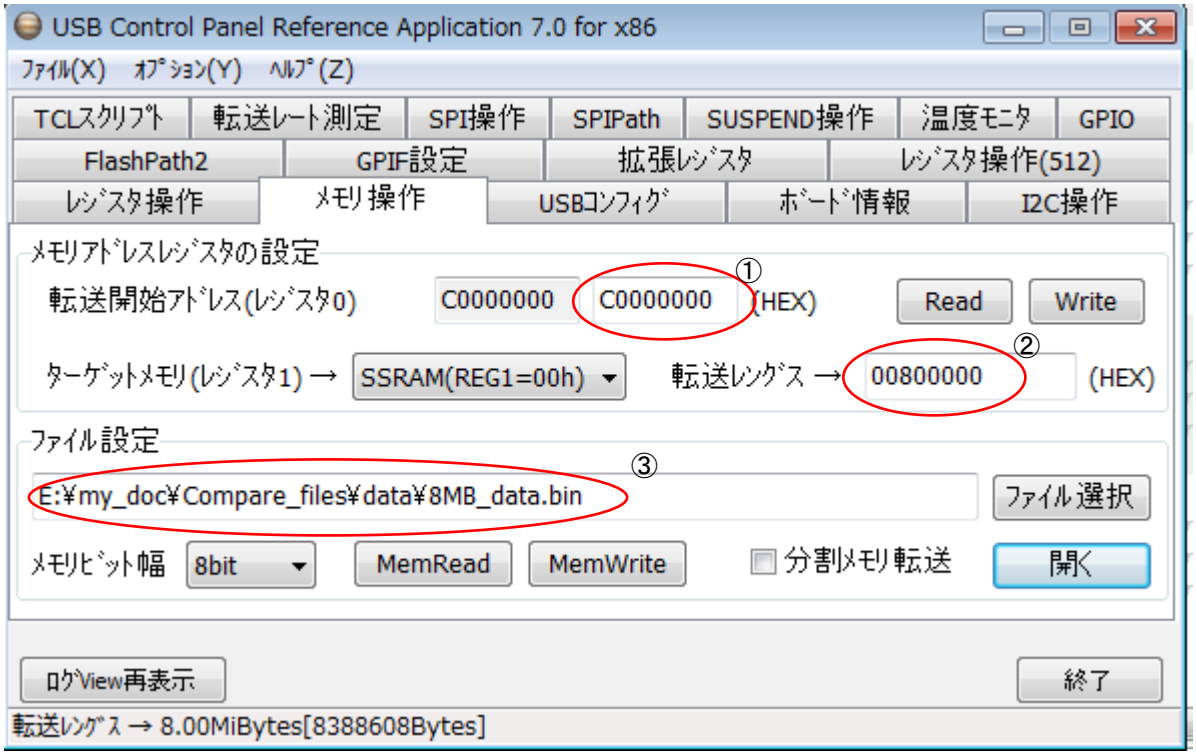


<図 22. Generic Tri-State Controller と DP-FIFO の接続>

DP-FIFO が空の場合は Avalon バスシステムに wait 信号を出力することで、FIFO の内容を読み出すことを停止できます。 また、wrfull 信号がアクティブの場合は FIFO への書き込みを停止できます。wrfull、rdempty 信号によりデータの書き込み、データの読み出しを制御しながらデータを PC に収集することができます。

7.5 RefApp7 からのデータ収集設定方法

ボード製品に標準添付しているリファレンス制御アプリケーション“RefApp7”を使用して、データを読み出し、PC にファイル化します。
RefApp7 のメモリ操作画面では、次のように設定します。



<図 23. RefApp7 での設定>

【設定手順】
① DP-FIFO を接続した Generic Tri-State Controller のベースアドレスを設定します。この例では 0x4000_000 がベースアドレスですが、最上位ビットを“1”にセットすると、GPIF_Master はアドレスをインクリメントしなくなるので、0xC000_0000 を設定します。
② PC に収集したいデータ容量(バイト設定)を 16 進数で設定します。この例では 8MB です。
ボードに大きなメモリを搭載していなくても、FIFO をバッファとしながらデータを収集することができます。
設定できる最大容量は 4GB です。
③ 読み出したデータをファイル化して保存するために、フルパス指定してください。保存形式はバイナリ(.bin)です。

7.6 応用・適用分野

今回の例ではデータ収集を目的に示しましたが、データ流の方向を変えればボードへの書き込みも可能になります。低速な AD コンバータや、DA コンバータなど連続したデータを転送したい場合などに便利です。

また、FIFO の代わりにデュアルポート RAM を Generic Tri-State Controller に接続することもできます。

【関連資料】 SUA007.pdf 「SRAM-FIFO モジュール」のリファレンス回路解説