

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. Quartusll 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>



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。

<図 2. GPIF-AVALON ブリッジのブロック図>

- 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 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	0x0000000	GPIF_Masterの ベースアドレスと 同じ値に必ず設定 32bit 幅設定	 未使用
led(LED)	0x0000010	8bit 幅設定	4
swin (スイッチ入力)	0x0000020	8bit 幅設定	8
seg (7 セグ表示)	0x0000030	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>



a Qsys - QSYS_top.qsys* (E:¥Project¥Design_data¥qdesigns¥CXUSB2¥GPIF_QSYS_Verilog¥EP3C25¥QSYS_top.qsys)										
File Edit System View Tools Help)									
Component Library	Syste	em Conte	ents Address I	Map Clock Settings Project !	Settings Instance Parameters System In	nspector HDL Example Gene	ration			
	+	Use	Connections	Name	Description	Export	Clock	Base	End	
Project Project B Smart-USB Plus G GPF - AVALON Bridge. B System Library B Bridges Clock and Reset Configuration & Programming D DSP Embedded Processors B Interface Protocols B Memories and Memory Controller B Microcontroller Peripherals B Debug and Performance B Display B Microcontroller Peripherals B		য় হার্থেরের বির্থের	Address	Name Name Image: Constraints Image: Constraints <	Description Description Clock Source Clock Input Reset Input Clock Output Reset Input Clock Input GPIF - AVALON Bridge. Clock Input Clock Input Reset Input Conduit Avaion Idemory Mapped Master PIO (Parallel VO) PIO (Parallel VO) PIO (Parallel VO) PIO (Parallel VO) PIO (Parallel VO) Clock Input Reset Input Avaion Idemory Mapped Slave Conduit Endpoint PIO (Parallel VO) Clock Input Reset Input Avaion Idemory Mapped Slave Conduit Endpoint PIO (Parallel VO) Clock Input Reset Input Avaion Idemory Mapped Slave Conduit Endpoint Conduit Endpoint	Export Export Export Cilk reset Cilck to export Cilck to expor	Clock Clk_0 [clock_reset] [clock_reset] [clock_reset] [clk_0 clk_0 clk_0 clk_0 clk_0 clk_0 [clk] [clk] [clk] [clk]	Base ■ 0×00000000 ■ 0×0000000 ■ 0×0000000 ■ 0×0000000 ■ 0×0000000 ■ 0×0000000 ■ 0×0000000 ■ 0×00000000 ■ 0×00000000 ■ 0×000000000 ■ 0×00000000 ■ 0×00000000000000 ■ 0×000000000 ■ 0×00000000000000000000000000000000000	End	
⊡-SLS ⊡-Verification										

<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。



				(/			
×	-	Use	Connections	Name	Description	Export	Clock	Base	End
	×			Cik_0	Clock Source				
roject				clk_in	Clock Input	clk			
vew component	1.00			clk_in_reset	Reset Input	reset			
Smart-USB Plus			$ \longrightarrow$	clk	Clock Output	Click to export	clk_0		
GPIF - AVALON Bridge.				clk_reset	Reset Output	Click to export			
····· • ZBT-SSRAM		V		GPIF_Master_0	GPIF - AVALON Bridge.				
System			$ \rightarrow \rightarrow$	clock_reset	Clock Input	Click to export	clk_0		
brary	X.		$ \longrightarrow$	clock reset reset	Reset Input	Click to export	[clock reset]		
Bridges	· -			conduit end	Conduit	qpif master 0 conduit			
Clock and Reset	WY.			avalon master	Avalon Memory Mapped Master	Click to export	[clock reset]		
Configuration & Programming		•	$ \rangle \rightarrow \rangle \rightarrow$	± notuse	PIO (Parallel VO)		clk 0	▲ 0×00000000	0×0000000
DSP		•	$ \rightarrow \rightarrow \rightarrow$	± led	PIO (Parallel VO)		clk 0		0×0000001
Embedded Processors		•	$ \rightarrow \rightarrow \rightarrow$	± swin	PIO (Parallel VO)		clk 0	0x00000020	0×0000002
Interface Protocols		V	$ \rightarrow \rightarrow \rightarrow $	t seg	PIO (Parallel VO)		clk 0	0x0000030	0×0000003
Memories and Memory Controller			$ \rightarrow \rightarrow$	H ssram	ZBT-SSRAM		clk 0	0x00200000	0x003ffff
Microcontroller Peripherals		2			PIO (Parallel I/O)		onit_o	-	
Peripherals				clk	Clock Input	Click to export	clk 0		
Debug and Performance				rapat	Penet Input	Click to export	CIK_0		
+ Display				16561	Avalon Memory Manned Slave	Click to export	[Cik]	- 0×00000040	0×0000004
-Microcontroller Peripherals				external connection	Conduit Endpoint		[Cik]	-	
 Interval Timer 		E.			BIO (Barallal I/O)	pio_o_external			
PIO (Parallel I/O)					Clock Input	Click to export	alk 0		
Vectored Interrupt C					Clock input	Click to export	CIK_U		
PLL				reset	Reset input	Click to export		-0 0×00000050	0×0000005
Processor Subsystems				SI	Avaion memory mapped Slave	Crick to export		0.000000000	0.00000000
Hard Processor System			0	external_connection	Conduit Endpoint	pio_1_external			
Osys Interconnect									
-SI S									
Verification									

<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)	0x0000010	8bit 幅設定	4
Swin (スイッチ入力)	0x00000020	8bit 幅設定	8
Seg (7 セグ表示)	0x0000030	16bit 幅設定	12
pio_0	0x0000040	32bit 幅、出力専用	16
pio_1	0x0000050	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(pl	l_clk),			
	.reset_rese	t_n(rstn),			
	//	GPIF I/F			
	.gpif_mast	er_0_conduit_end_fd(fd),			
	.gpif_mast	er_0_conduit_end_ctl({ 2'b11, rgdtn, cmdn, wr	m, rdn}),		
	.gpif_mast	er_0_conduit_end_rdy(rdy_wire),			
	//	notuse			
	//	QSYS Address = 0000_0000h - 0000_000Fh			
	//	GPIF REGISTER Number = 0, 32bit Read O	nly		
	.notuse_ex	ternal_connection_export(),			
	11	led			
	11	$QSYS$ Address = 0000_0010h - 0000_001Fh	7 - 1		
	//	GPIF REGISTER Number = 4, 32bit Read/W	rite		
	.led_exterr	al_connection_export(led_wire),			
		SW1n $OSVS Address = 0.000, 0.020h - 0.000, 0.02Fh$			
	11	QDIE DECISTED Number = 12, 22bit Dood) mlrr		
	awin orto	(1117 REGISTER Number - 12, 5250 Read)	Jilly		
	.swiii_exte	sog			
	11	OSVS Address = 0000 0000b - 0000 000Fb			
	11	GPIF REGISTER Number = 16 32bit Read/	Write		
	.seg exteri	al connection export(seg wire).			
	//	pio 0			
	11	QSYS Address = 0000 0040h - 0000 004Fh			
	//	GPIF REGISTER Number = 12, 32bit Read/	Write		
	.pio_0_exte	ernal_export(pio_wire),			
	//	pio_1			
	//	QSYS Address = 0000_0050h - 0000_005Fh			
	//	GPIF REGISTER Number = 16, 32bit Read			
	.pio_1_exte	ernal_export(pio_wire),			
	//	QSYS Address = 0020_0000h - 002F_FFFFh			
	//	2MiBytes			
	.ssram_cor	nduit_end_ab(ssram_ab),			
	.ssram_cor	iduit_end_db(ssram_db),			
	.ssram_cor	iduit_end_csn(ssram_csn),			
	.ssram_cor	nduit_end_wn(ssram_wen),			
	.ssram_cor	nduit_end_bwan(ssram_bwan),			
	.ssram_cor	nduit_end_bwbn(ssram_bwbn),			
	.ssram_cor	iduit_end_bwcn(ssram_bwcn),			
	.ssram_cor	iduit_end_bwdn(ssram_bwdn),			
	.ssram_cor	duit_end_adv(ssram_adv),			
	.ssram_cor	duit and gn(ssram ocn)			
	.ssialli_001	duit end ftn()	11	notuse	
	ssram cor	uduit end lbon()	//	notuse	
	ssram cor	aduit end zz0		notuse	
);	aatt_ona_bbv		1100000	

<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。



File Edit System View Tools Help Component Library	Syste	m Conte	ents Ad	dress Map Clock Settings F	roject Settings Insta	Ince Parameters Sy	stem Inspector	HDL Example	Generation	
	- + [Use	Conne	Name	Desc	cription	Ext	oort	Clock	Base
Project 			Ϋ́ Ϋ́ Ϋ́	Cik_in cik_in_reset cik cik_reset	Clock Source Clock Input Reset Input Clock Output Reset Output		clk reset Click to Click to	export export	cik_0	
New Edit Add		•								Þ
Messages										
		De	scription					Patn		

<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。



🛔 Edit Module – Qsys				
File Edit System View Tools	Help	Γ	<u>Γ</u> Γ.Γ.	
Component Library	System Contents Address Map Clock Settings Project	Settings Instance Parameters System I	nspector HDL Example Gen	eration
	Use Connections Name	Description	Export	Clock B
Project	Chin Chin Chin Chin Chin Chin Chin Chin	Clock Source Clock Input Reset Input Clock Output	clk reset Click to export	clk_0
Library D-Bridges	ZBT-SSRAM ZBT_SSRAM		Documentation ort	unconnected [clock_reset]
E-Configuration & Programming	🔻 Block Diagram	🔻 Parameters	ort	[clock_reset]
PoSP Embedded Processors Interface Protocols Memories and Memory Controll Microcontroller Peripherals Peripherals Peripherals Processor Subsystems Osys Interconnect SLS Verification Interface Interface	Show signals ZBT_SSRAM_0 clock_reset_clock clock_reset_reset avalon_slave_0 avalon conduit_end conduit ZBT_SSRAM_0: Memory Size : 2048 KBytes	Address Width(bits): 19 v bits SSRAM opreation mode: © "FlowTh © "Pipeline	rough-mode" ort ort ort ort	unconnected [clock_reset] [clock_reset] ⊫ੇ
		Canc	el Finish	
Messages				
	Description		Path	<u>×</u>
4 Errors				
GPIF_Master_0.clock_rese	t must be connected to a clock output	System.GPIF_Master_0		
ZBT_SSRAM_0.clock_reset	must be connected to a clock output	System.ZBT_SSRAM_0		
GPIF_Master_0.clock_rese	t_reset must be connected to a reset source	System.GPIF_Master_0		
ZBT_SSRAM_0.clock_reset	_reset must be connected to a reset source	System.ZBT_SSRAM_0		

<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。



🖁 PIO (Parallel I/O) - pio_0				×
PIO (Parallel I/O) attera_avalon_pio			Docu	imentation
Block Diagram				
Show signals	Basic Settings			
	Width (1-32 bits):	32		
pio_0	Direction:	C Bidir		
		O Input		
clock		🔿 InOut		
reset		Output		
s1avalon	Output Port Reset Value:	0x000000000000000	0	
external_connection conduit		,		
altera_avalon_pio	Output Register			
		setting/cleaning		
	👅 Edge capture registe	r		
	Synchronously capt	Ire		
	Edge Type:	RISING 🔽		
	Enable bit-clearing fo	r edge capture registe	r	
	Tinterrupt			
	Generate IRQ			
	IRQ Type:			
	Edge: Interrupt CPU whe	en any unmasked VO p n any unmasked bit in t	in is logic true the edge-capt	ure
	register is logic true. Ava	ilable when synchrono	ous capture is	enabled
	Test bench wiring			
	Hardwire PIO inputs	in test bench		
	Drive inputs to:	0x000000000000000	0	
	,			
			Canada	Finish
			Cancer	FIIIISN

<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	0x0000030	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	



🛃 Qsys										
File Edit System View Tools He	lp									
Component Library	Syste	em Conte	Address N	Ap Clock Settings Project	Settings Instance Parameters Sys	stem Inspector HDL Example Ge	neration			
	+	Use	Connections	Name	Description	Export	Clock	Base	End	IRQ
	×	2		Cik_0	Clock Source					
Project			₽-	clk_in	Clock Input	clk				
New component			⊳	clk_in_reset	Reset Input	reset				
Smart-USB Plus				clk	Clock Output	Click to export	clk_0			
OFIF - AVALON Bridge. O ZBT_SSRAM		-		clk_reset	Reset Output	Click to export				
Library	-	M		GPIF_Master_0	GPIF - AVALON Bridge.					
Bridges				clock_reset	Clock input	Click to export	CIK_U			
E Clock and Reset				clock_reset_reset	Conduit	Click to export	[Clock_reset]			
E-Configuration & Programming	8			avalon master	Avalon Memory Manned Master	Click to export	[clock_reset]			
		V		E ZBT SSRAM 0	ZBT-SSRAM	onon to export	[clock_redet]			
Embedded Processors			$ \longrightarrow $	clock reset	Clock Input	Click to export	clk 0			
Interface Protocols			$ \rightarrow \rightarrow$	clock_reset_reset	Reset Input	Click to export	[clock_reset]			
Memories and Memory Controller				avalon_slave_0	Avaion Memory Mapped Slave	Click to export	[clock_reset]	💣 0x00100000	0x002fffff	
Hicrocontroller Peripherals				conduit_end	Conduit	Click to export				
E-Peripherals				🗆 dummy	PIO (Parallel I/O)					
Debug and Performance Deplay			\rightarrow	clk	Clock Input	Click to export	clk_0			
Microcontroller Peripherals				reset	Reset Input	Click to export	[clk]	dumm	y Denetlet (O) feite	
Interval Timer				S1	Avaion Memory Mapped Slave	Click to export	[CIK]	S OXOOO PIO	rarallel 1/0) (alte	ra_avaion_pic
PIO (Parallel VO)				E lod	Conduit Endpoint	Crick to export				
Vectored Interrupt C		<u>.</u>	$ \longrightarrow $	clk	Clock Input	Glick to export	clk 0			
			$ \rightarrow \rightarrow$	reset	Reset Input	Glick to export	[clk]			
Processor Subsystems			$ \longrightarrow$	s1	Avaion Memory Mapped Slave	Click to export	[clk]	0x00000010	0x0000001f	
Qsys Interconnect			-	external_connection	Conduit Endpoint	Click to export				
H SLS		V		🗄 seg	PIO (Parallel I/O)					
±Verification			$\vdash \longrightarrow$	clk	Clock Input	Click to export	clk_0			
			$ \rightarrow$	reset	Reset Input	Click to export	[clk]			
			$ \rightarrow$	s1	Avaion Memory Mapped Slave	Click to export	[clk]	e 0x00000020	0×0000002f	
New Edit 🕆 Add		4		external connection	Conduit Endpoint	Click to export	1	1		
Messages										
Description Path K										
A led.external_connection must	be expo	rted, or	connected to a n	natching conduit.		System.led				
seg.external_connection must	be expo	orted, or	connected to a	matching conduit.		System.seg				
swin.external_connection mu	st be exp	ported, o	or connected to a	a matching conduit.		System.swin				
A regout.external_connection n	nust be e	xported	, or connected to	a matching conduit.		System.regout				
A regin.external_connection mu	st be ex	ported, o	or connected to a	a matching conduit.		System.regin				-
1 Error, 8 Warnings										

<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ェイトする場合に、「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=0x0000010、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 0x0000010 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。





レジスラ0のリード概念図

<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。
- 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)。
- 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のリードバ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、開始コマント、を受け取ると、 FPGAにメモリリート、開始コマント、を発行する。



(3)FX2は、FPGAから2048バイト(512×4面)のデータをリードして、 内蔵バッファをメモリデータで埋める。





(4)メモリリードコマンドから、数ms後にホストPCがデータリードを行う。 512バイトのリードが行なわれ、バッファが1面空になる



(5)空になったバッファを埋める為、FX2はFPGAから512バイトのリードを行う。

ホストPCは必要なデータの回数(4)~(5)の処理を繰返し行う



(6)ホストPCは必要なデータをリードし終えると、メモリリード終了コマンドを発行し 処理を完了する。FX2のバッファはすべてクリアされる。

<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、<u>SRAM-FIFO 回路</u>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 <u>SUA007</u>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。

🔨 MegaWizard Plug-In Manag	er [page 1 of 8]	? 💌
🎒 FIFO		About Documentation
Parameter 2 EDA 3		tion
Width, Clks, Synchronization	DCFIFO 1 DCFIFO 2 Rdreq Option, Blk Type Optimization, Circuitry Protect Currently selected device family: How wide should the FIFO be? Image: Use a different output width and set to How deep should the FIFO be? Note: You could enter arbitrary values for width Do you want a common clock for reading and writing the FIFO? Yes, synchronize both reading and writing to 'clock'. Create one set of full/empty control signals. No, synchronize reading and writing to 'clock'. Create a set of full/empty control signals for each clock.	tion Cyclone IV E Match project/default 8 bits 32 bits 16384 words
Resource Usage 52 lut + 16 M9K + 163 reg	Cancel < Ba	ick Next > Finish

<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。



🔨 MegaWizard Plug-In Manager [page	2 4 of 8]		? 💌
🤄 FIFO			About Documentation
Parameter Settings 2 EDA 3 Summary Width, Clks, Synchronization DCFIF	0 1 DCFIFO 2 Rdreg Option, E	3lk Type > Optimization, Circuitry Prot	ection >
dpfifo data[70] wrfull wrreq wrclk wrusedw[140] rdreq q[310] rdreq rdclk rdempty aclr 8 bits x 16384 words	Which optional output control signals do you Read-side full wenty usedw[] Note: These signals are synchronous to 'rdclk' Add an extra MSB to usedw port(s) Add circuit to synchronize 'aclr' input Add circuit to synchronize 'aclr' input	want? Write-side I full empty U usedw] Note: These signals are synchronous to 'wrclk' with 'wrclk' with 'rddk'	usedw] is the number of words in the FIFO. Note: You can use the MSB to generate a half-full flag.
Resource Usage		Cancel	Back Next > Finish

<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 Late

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。

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



👃 Qsys - QSYS_top.qsys (E:¥Project¥Design_data¥qdesigns¥CXCard4¥CX-Card4_GPIF_QSYS_Verilog_ext_mem¥EP4CE40¥QSYS_top.qsys) 🕞 💷 🔤								
File Edit System View Tools Help								
Component Library	System C	ontents Address Ma	ap Clock Settings Project Set	ettings Instance Parameters System	Inspector HDL Example General	tion		
X	🕈 Use	Connections	Name	Description	Export	Clock	Base	End
Project New Component Smart-USB Plus			 clock_reset clock_reset_reset conduit_end avalon_master 	Clock Input Reset Input Conduit Avalon Memory Mapped Master	Double-click to export Double-click to export gpif_master_0_conduit Double-click to export	clk_0 [clock_reset]		
		⊨∣ ∳ ∳ ∲ ────ः	• ⊞ notuse	PIO (Parallel VO)		clk_0		0x0000_000f
Bridges Gock Bridge Gock Bridge	► N (☐ led clk reset s1	PIO (Parallel I/O) Clock Input Reset Input Avalon Memory Mapped Slave	Double-click to export Double-click to export Double-click to export	clk_0 [clk] [clk]	■ 0×0000_0010	0×0000_001f
DMA OMA Controller Scatter-Gather DI OMercontroller Scatter-Gather DI Omemory-Mapped O Avalon-MM Clock O Avalon-MM DDP 1	V		swin swin clk reset s1 external_connection	PIO (Parallel VO) Clock Input Reset Input Avalon Memory Mapped Slave Conduit	Double-click to export Double-click to export Double-click to export Swin_external_connection	cik_0 [cik] [cik]	▲ 0×0000_0020	0×0000_002f
O Avalon-MM Pipelir O Avalon-MM Trista O JTAG to Avalon k O ST Slave to Avale				PIO (Parallel VO) Clock Input Reset Input Avalon Memory Mapped Slave Conduit	Double-click to export Double-click to export Double-click to export pio_0_external_connection	clk_0 [clk] [clk]	▲ 0×0000_0030	0×0000_003f
Avalon Packets tr Avalon -ST Adapt Avalon-ST Bytes Avalon-ST Channel			onchip_memory2_0 clk1 s1 reset1	On-Chip Memory (RAM or ROM) Clock Input Avalon Memory Mapped Slave Reset Input	Double-click to export Double-click to export Double-click to export	clk_0 [clk1] [clk1]	a 0x0001_0000	0×0001_1fff
 Avaion-ST Data F Avaion-ST Delay Avaion-ST Demut Avaion-ST Error J Avaion-ST Multipi 			L sram clock reset avalon_slave_0 conduit_end	Async-SRAM Controller. Clock Input Reset Input Avalon Memory Mapped Slave Conduit	Double-click to export Double-click to export Double-click to export sram_conduit_end	clk_0 [clock] [clock]	₽ 0×0010_0000	0×001f_ffff
Avalon-ST Packe Australia ST Shitta M M M Edit Edit			clk clk reset uas tcm	Clock Input Reset Input Avalon Memory Mapped Slave Tristate Conduit Master	Double-click to export Double-click to export Double-click to export generic_tristate_controll	cik_0 [cik] [cik] [cik]	■ 0x4000_0000	0x7fff_ffff

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

👃 Generic Tri-State Controller - generic_tristate_controller_0						
Generic Tri-State Controller attera_generic_tristate_controller						
Block Diagram Show signals generic_tristate_controller_0 clk elook tristate_conduit tcm reset reset avalon altera_generic_tristate_controller	Signal Selection Signal Timing Signal Polarities Address width: 30 32 Data width: 32 32 Byteenable width: 4 4 Bytes per word: 4 4 Enable the following signals: Refer to the Avalon Interface Specifications for definitions or Image: read and the polymer is a specification of the polymer					

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



Generic Tri-State Controller - generic_tristate_controller_0 Generic Tri-State Controller altera_generic_tristate_controller						
Block Diagram Show signals generic_tristate_controller_0 clk clock reset reset avalon altera_generic_tristate_controller	Signal Selection Signal Timing Signal Polarities Read wait time: 0 0 Write wait time: 0 0 Setup time: 0 0 Data hold time: 0 0 Maximum pending read transactions: 1 1 Turnaround time: 2 1 Read latency: 1 1 Chipselect through read latency 1					

<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。

😂 USB Control Panel Reference Application 7.0 for x86							
ファイル(X) オプション(Y) ヘ	ファイル(X) オプション(Y) ヘルプ(Z)						
TCLスクリフト 転送レート測定 SPI操作 SPIPath SUSPEND操作 温度モニタ GPIO							
FlashPath2	GPIF設定	拡張レジ	*スタ	レシ [、] スタ操作 <mark>(512)</mark>			
レジスタ操作	メモリ操作	USBコンフィク	ホートが情報	员 <u>I2C</u> 操作			
ーメモリアドレスレジスタの設	定	\sim]			
■ 転送開始アドレス(レシ	[*] スタ0) C000000	0 (C0000000	(HEX)	Read Write			
ターケットメモリ(レシ`スタ1) → SSRAM(REG1=00h) ▼ 転送レンクス → 00800000 (HEX)							
-ファイル設定							
③ E:¥my_doc¥Compare_files¥data¥8MB_data.bin ファイル選択							
メモリビット幅 8bit - MemRead MemWrite 回分割以刊転送 開く							
ロかView再表示				終了			
転送レングス → 8.00MiBytes[8388608Bytes]							

<図 23. RefApp7 での設定>

- 【設定手順】
- 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