# GPIF2 用 AXI バスマスタ IP : GPIF2\_Master

#### 1. 概要

AMD/Xilinx vivado 開発ツールで利用できる AXI バスマスタ IP <GPIF2\_Master> により、USB 制御コアを AXI バスマスタとし、既存の 各種 AXI スレーブ IP を制御できます。また、MicroBlaze や Zynq ARM コア、DMA 等のバスマスタと混在も可能です。 WindowsPC や LinuxPC を AXI バスマスタと見立てて、FPGA を簡単に制御できる IP です。

#### 【適用ボード】

AMD/Xilinx 製 FPGA を搭載する Smart-USB Sigma 製品 AX-Card7 (Artix7), SX-Card7 (Spartan7) KX-Card7 (Kintex7), AX-Card7M(Artix7), ZQ-Card (Zynq7000)

【対応開発ツール】

AMD/Xilinx vivado2022.2 以降

Block Design 環境で、既存の各種 AXI スレーブ IP や他のバスマスタと組み合わせて利用できます。

# 2. USB 制御回路の AXI バスマスタ化

2.1 提供するファイル

GPIF2\_Master.v

|--- GPIF2\_WFIFO.xci |--- GPIF2\_RFIFO.xci |--- GPIF2\_RegMaster.v

vivado ツールの Block Design 環境で、GPIF2\_Master を読み込んで利用することができます。

#### 2.2 使い方 (Block Design で GPIF2\_Master を利用する)

- 1 vivado で新規プロジェクトを作成し、上記の4つのファイルをプロジェクトに追加します。
  - 2 Source ウインドウで GPIF2\_WFIFO\_inst を選択し、「Upgrade IP…」を実行します。



<  $\boxtimes$  1. FIFO  $\mathcal{O}$  Generate Output Products >

3 Upgrade IP 実施後、「Generate Output Products」を実行します。このとき、Synthesis Option は Out of context per IP にしてください。

同じ操作を GPIF2\_RFIFO\_inst にも実施します。

Sources		? _ 🗆 🖒 ×	Proj		
Q   ¥   ♦   +   ?   ● 0		۰	Ov		
✓  ☐ Design Sources (1)					
GPIF2_Master (GPIF2_Master.v) (3)			Se		
👎 GPIF2_WFIF0_inst : GPIF2_WFIF0 (GPIF2_W		Source Node Properties			
GPIF2_RFIF0_inst: GPIF2_RFIF0 (GPIF2_RF		Source Node Fropenies.			
RegMaster_inst: GPIF2_RegMaster (GPIF2_R		Enable Core Container			
> 🗁 Constraints	۶	Re-customize IP			
✓  ☐ Simulation Sources (1)		Generate Output Product	S		
> 🗁 sim_1 (1)		Reset Output Products			
> 🖆 Utility Sources		Upgrade IP			
		0			

# < $\boxtimes$ 1.1. FIFO $\mathcal{O}$ Generate Output Products >

Sources	A Generate Output Products
Q Q C C C C C C C C C C	The following output products will be generated.
GPIF2_RFIF0_inst : GPIF2_RFIF0 (GPIF2_RFIF0.xci) RegMaster_inst : GPIF2_RegMaster (GPIF2_RegMaster)	Preview Q X ↓ ♦
<ul> <li>Constraints</li> <li>Simulation Sources (1)</li> <li>Sim_1 (1)</li> <li>Utility Sources</li> </ul>	<ul> <li>Pi GPIF2_WFIFO.xci (Global)</li> <li>Instantiation Template</li> <li>RTL Sources</li> <li>Behavioral Simulation</li> <li>Change Log</li> </ul>
Hierarchy IP Sources Libraries Compile Order	Synthesis Options
Source File Properties	<ul> <li>○ Global</li> <li>Out of context per IP</li> </ul>
₽ GPIF2_WFIF0.xci	Run Settings
Enabled Location: D:/project/xilinx_work/SXCard7/AXI_doc/project_1/p Type: IP Part: xc7s75fgga676-2 Size: 49.3 KB	<ul> <li>On local host: Number of jobs: 10 </li> <li>Generate scripts only</li> <li>Do not launch</li> <li>Apply</li> <li>Generate</li> <li>Cancel</li> </ul>

< 図1.2. FIFOのGenerate Output Products >

- 4 Flow Navigator の IP INTEGRATOR から「Create Block Design」をクリックしてエディタを起動します。
- 5 画面上で右クリック後「Add Module …」を選択し、GPIF2\_Masterを選択すると以下の画面になります。 これで USB 制御回路を AXI バスマスタとして利用できる準備が整いました。



< 図2. Block Design での GPIF2\_Master 呼び出し >

GPIF2\_Master の他に、AXI Interconnect IP を配置し、必要なスレーブ IP (IIC、GPIO、MIG、QSPI、BRAM 等)を追加して
 回路設計ができます。以下の例は、AX-Card7 システム開発ボードをターゲットに、DDR3 制御(MIG)、LED 制御8個(GPIO)、
 内蔵メモリ BRAM(32KB)、IIC を追加した例です。



<図3. Block Designの完成 >

7 回路が完成したらキーボード「F6」キーを押して「Validate Design」を実行し、エラーのないことを確認後、「Generate Output Products」を実行してください。このとき、図 1.2 で示す画面と同じメニューが表示されるので、「Global」を選択してから Generate ボタン をクリックしてください。 トップ回路を RTL 記述する場合は「Create HDL Wrapper…」を実行し、Block design をインスタンシエイトし て利用します。

### 2.3 PC からの制御方法

従来の RefApp7 制御アプリを利用できません(23.2 月現在)。 Python 等のツールで SUSIv.dll をインポートし制御します。

1 アドレスマップの確認

図3の画面でアドレスマップを確認します。

Diagram ×	Address E	ditor	×	Address M	lap :	×							
Q 🛛 🛨 🛛 🗧	▶   ↓   :	t		Assigned (4	4) (	🕑 Unassigne	ed (0) 🕑	Excluded (0)	Hide All		$\sim$		
Name						^ 1	Interface	Slave Segment	Master Base	Address (	Range		Master High Address
✓  ☐ Network	0										$\smile$		
V 👎 /GPIF	2_Master_0												
∨ 🖽 /G	PIF2_Master	_0/M_A>	<b>(</b> (32	2 address b	oits : 40	3)							
1	/axi_bram_c	trl_0/S_	AXI				S_AXI	Mem0	0×C000_0000	D	32K	*	0×C000_7FFF
1	/axi_gpio_0/	S_AXI					S_AXI	Reg	0×5000_0000	O	4K	Ŧ	0×5000_0FFF
19	/axi_iic_0/S_	AXI					S_AXI	Reg	0×5001_0000	O	4K	•	0x5001_0FFF
1	/mig_7serie	s_0/me	mma	ар			S_AXI	memaddr	0×8000_0000	D	512M	•	0x9FFF_FFF
				Diagram	n x	Address I	Editor ×	Address Map	×				
				Scale	Log		~						
				М	act	orc	netw	OFK U		CL			
				6	ast	ers				Sia	ves		
				/GPIF2	2	ers	0x5(	000_0000 /ax	i_gpio_0/S_	АХІ	ves		4К
				/GPIF2_M.		ers	<b>0x5</b> (	000_0000 /ax	i_gpio_0/S_	AXI	ves		4K 60K
				/GPIF2_Maste	MAX	ers	0x5( 0x5( 0x5)	000_0000 /ax 000_1000 001_0000 /ax	i_gpio_0/S_	AXI	ves		4K 60K 4K
				/GPIF2_Master_0	MAX	ers	0x50 0x50 0x50 0x50	000_000 /ax 000_1000 /ax 001_0000 /ax	ii_gpio_0/S_ ii_iic_0/S_AX	AXI (I	ves		4K 60K 4K 0x2ffe_f0(
				/GPIF2_Master_0	MAXI		0x5( 0x5) 0x5( 0x5) 0x8(	000_000 /ax 000_1000 /ax 001_0000 /ax 000_0000 /m	ii_gpio_0/S_ ii_iic_0/S_AX ig_7series_0	SIa AXI (1 )/S_AXI	ves		4K 60K 4K 0x2ffe_f0( 512M
				/GPIF2_Master_0	MAXI		0x50 0x50 0x50 0x80 0x80	000_0000 /ax 000_1000 /ax 001_0000 /ax 000_0000 /ax	ii_gpio_0/S_ ii_iic_0/S_AX ig_7series_0	51a AXI (I )/S_AXI	ves		4K 60K 4K 0x2ffe_f0( 512M
				/GPIF2_Master_0			0x5( 0x5) 0x5( 0x8( 0x8)	000_0000 /ax 000_1000 /ax 001_0000 /ax 000_0000 /m 000_0000 /ax	ii_gpio_0/S_ ii_iic_0/S_AX ig_7series_0 ii_bram_ctrl_	SIa AXI (I )/S_AXI	ves		4K 60K 4K 0x2ffe_f0( 512M 512M 32K
				/GPIF2_Master_0		ers	0x50 0x50 0x50 0x80 0x80 0x80 0x80	000_0000 /ax 000_1000 /ax 001_0000 /ax 000_0000 /m 000_0000 /ax	ii_gpio_0/S_ ii_iic_0/S_AX ig_7series_0 ii_bram_ctrl_	SIa AXI (I )/S_AXI	ves		4K 60K 4K 0x2ffe_f0( 512M 512M 32K

< 図4.アドレスマップ >

LED を制御する GPIO 0x5000\_0000、 IIC 制御 0x5001\_0000、 DDR3 制御 0x8000\_0000、 内蔵メモリ制御 0xC000\_0000 です。 アドレス範囲は、Address Editor タブから変更できます。 デフォルトで 64K です。 内蔵メモリの容量はこの画面で Range を選択することで変更できます。

#### 2 レジスタ操作とメモリ操作の識別

従来の RefApp7 では IIC や LED を「レジスタ操作」、DDR3 や内蔵メモリを「メモリ操作」としていました。 レジスタ制御の場合、レジスタベースアドレス 0x1F\_FFFF (d2097151)、 メモリ制御の場合、メモリベースアドレス 0x1F\_FFFE (d2097150)にそれぞれアドレスマップに示す先頭アドレスを書き込みます。

例えば、SUSIv\_Reg\_Write 関数で、レジスタ番号 0x1F\_FFFF にデータ 0x5000\_0000 を書き込んだ後、SUSIv\_Reg\_Write や SUSIv\_Reg\_Read で指定するレジスタ番号"0"で GPIO 制御できます。 IIC ブロックは、オフセット 0x10000 なので、レジスタ番 号"65,536"で制御できます。 IIC 内のサブレジスタは、それぞれ 65,536+各レジスタのオフセット がレジスタ番号となります。同様にメ モリベースアドレス 0x1F\_FFFE に 0x8000\_0000 を WR すると、DDR3 メモリに対して SUSIv\_Data\_Write や SUSIv\_Data\_Read 関数で、データの書き込、読み出しができます。 0xC000\_0000 を設定すれば、内蔵メモリの BRAM(32kB)アクセスができます。

レジスタベースアドレス 0x1F\_FFFF に 0x5001\_0000 を WR すれば、IIC の先頭アドレスは、 レジスタ番号" 0 "でアクセスすることができます。

メモリベースアドレス 0x1F\_FFE に 0xC000\_0000 を設定すれば、内蔵メモリの BRAM (32kB) アクセスができます。

## 3. FPGA サンプルプロジェクト

AX-Card7 システム開発ボード向けに作成していますが、他のボードでもピンアサインの変更とDDR3(MIG)を変更することで対応できます。

<無償ダウンロード> https://www.prime-sys.co.jp/DownLoad/Ref\_Design/GPIF2M.zip

USBのデータ転送レート RefApp7の「転送レート測定」機能で計測した値と比較して、約8%低下します。

PC 仕様{Core I7-12700, 32GB RAM windows10-64bit, Intel H610 chipset} で運用した場合、RD/WR ともに約 340MB/s

の転送レートです。

【備考】 2023.02.20 第1版発行 2023.04.27 第2版発行(Generate Output Productsのパラメータ設定変更)