OPB PCI Arbiter

DS478 August 5, 2004 **Product Specification**

Introduction

The OPB PCI Arbiter provides arbitration among several PCI Master devices. Parametric selection determines the number of masters competing for PCI bus control. Both fixed and rotating arbitration schemes may be selected by programing a control register. Bus parking occurs in the case that no master requests PCI control. The particular master designated for parking is selected either with a programed register or by parametric selection. Register programming is accomplished through a slave interface to the OPB using the OPB_IPIF.

An alternate use mode provides a PCI arbiter without an OPB interface. Achieved through appropriate parameter settings, this mode completely removes the program registers and the OPB_IPIF leaving open the outputs to the OPB and connecting inputs from the OPB to logic zero. Without an OPB interface the PCI arbiter operates with rotating arbitration, the park master is set directly by parametric selection and there is no facility to modify priorities or arbitration scheme. The arbiter will start with master zero as highest priority then follow the rotating arbitration scheme to determine successive priorities.

The OPB_PCI_Arbiter follows arbitration guidelines defined in the PCI Local Bus Specification. Functional operation is verified to 75 MHz to ensure operation for 66 MHz PCI systems.

Features

- Variable number of PCI masters set by a parameter
- Fixed arbitration
- Rotating arbitration
- Control bit selection of arbitration scheme
- Bus parking
- Park master program register
- Alternative park master may be set by a parameter
- Control bit selection of park master
- Removable processor interface, OPB_IPIF
- Removable pipeline registers for PCI requests
- Removable pipeline registers for PCI grants
- OPB interface for register reads and writes

LogiCORE™ Facts		
Core Specifics		
Supported Device Family	QPro™-R Virtex™-II, QPro Virtex-II, Spartan™-II, Spartan-IIE, Spartan-3, Virtex, Virtex-II, Virtex-II Pro, Virtex-4, Virtex-E	
Version of Core	opb_pci_arbiter	v1.00a
Resources Used		
	Min	Max
I/O	4 + 2 * C_NUM_PCI_MSTRS	114 + 2 * C_NUM_PCI_MSTRS
LUTs	23	205
FFs	19	132
Block RAMs	0	0
Provided with Core		
Documentation	Product Specification	
Design File Formats	VHDL	
Constraints File	N/A	
Verification	N/A	
Instantiation Template	N/A	
Reference Designs & application notes	N/A	
Additional Items		
Design Tool Requirements		
Xilinx Implementation Tools	ISE 6.1.01i or later	
Verification	N/A	
Simulation	ModelSim SE/EE 5.7b or later	
Synthesis	XST	
Support		
Provided by Xilinx, Inc.		

Arbitration Schemes

The OPB_PCI_Arbiter provides fixed and rotating schemes to determine which PCI master is granted bus access in the case that multiple masters simultaneously request control. These schemes are described below.

Fixed Arbitration

The fixed arbitration scheme always places PCI master 0 at the highest priority with successive masters at incrementally lower priorities. This is used in cases that require one PCI master have the majority of bus access. For example, if one master provides initialization for many system components, that master should be connected to PCI_Req_n[0] and PCI_GNT_n[0] to achieve the highest priority in fixed arbitration. Table 1. illustrates the fixed arbitration scheme with four masters. Each column denotes the appropriate priority level for the PCI master in a given row and time is increasing from left to right. Shaded cells indicate the PCI master granted bus access for each transaction. Notice that the priority levels for the PCI masters remain fixed regardless of which master acquires bus control.

Table 1: **Fixed Arbitration Scheme**

PCI Master #	Priority for PCI Transaction n	Priority for PCI Transaction n+1	Priority for PCI Transaction n+2	Priority for PCI Transaction n+3	Priority for PCI Transaction n+4
0	Highest	Highest	Highest	Highest	Highest
1	Next Highest	Next Highest	Next Highest	Next Highest	Next Highest
2	Next Lowest	Next Lowest	Next Lowest	Next Lowest	Next Lowest
3	Lowest	Lowest	Lowest	Lowest	Lowest

Rotating Arbitration

The rotating arbitration scheme that shifts the priority level for each PCI master following bus transactions. This is when equal bus access is desired for all PCI masters. Table 2. illustrates the rotating arbitration scheme with four masters. Each column denotes the appropriate priority level for the PCI master in a given row and time is increasing from left to right. Shaded cells indicate the PCI master granted bus access for each transaction. Notice that the priority levels for the PCI Masters rotate, moving granted masters to the lowest priority for following transaction.

Table 2: **Rotating Arbitration Scheme**

PCI Master #	Priority for PCI Transaction n	Priority for PCI Transaction n+1	Priority for PCI Transaction n+2	Priority for PCI Transaction n+3	Priority for PCI Transaction n+4
0	Highest	Next Highest	Lowest	Highest	Next Lowest
1	Next Highest	Next Lowest	Highest	Next Highest	Lowest
2	Next Lowest	Lowest	Next Highest	Next Lowest	Highest
3	Lowest	Highest	Next Lowest	Lowest	Next Highest

OPB_PCI_Arbiter Design Parameters

The OPB_PCI_Arbiter may be configured for particular systems by appropriately selecting features through parameter values. This allows the user to craft a design with the minimum necessary resources that operates at the best possible performance. The features that are parametrized in the Xilinx OPB_PCI_Arbiter are shown in Table 3.

Table 3: **OPB_PCI_Arbiter Design Parameters**

Grouping / Number		Feature Description	Parameter Name	Allowable Values	Default Value	VHDL Type
Top Level	G1	Device Block ID	C__DEV_BLK_ID	0 - 255	1	integer
	G2	Device family	C_FAMILY	virtex, virtex2, virtex2p, spartan, spartan2 spartan3	virtex2p	string
	G3	Remove processor interface (OPB_IPIF)	C_RMOV_PROC_IF	0, 1 0 = OPB_IPIF included 1 = No OPB_IPIF	0 (include OPB_IPIF and registers)	integer
OPB Interface	G4	Device base address	C_BASEADDR	see note1	0xC00000FF	std logic vector
	G5	Device maximum address	C_HIGHADDR	see note 1	0xC0000000	std logic vector
	G6	Bus address width	C_AWIDTH	32	32	integer
	G7	Bus data width	C_DWIDTH	32	32	integer
PCI Interface	G8	Number of PCI Masters	C_NUM_PCI_MSTRS	2 - 8, see note 2	4	integer
	G9	Park Number of PCI Master designated for bus parking	C_PARK_PCI_MSTR	0 - 7, see note 3	0	integer
	G10	Remove PCI_Req_n Synchronization registers.	C_RMOV_REQ_REG	0, 1, see note 4 0 = Register included 1 = Register removed	0 (include registers)	integer
	G11	Remove PCI_GNT_n Synchronization registers.	C_RMOV_GNT_REG	0, 1, see note 4 0 = Register included 1 = Register removed	0 (include registers)	integer

Notes:

1. If the OPB_IPIF is included by setting G3 to 0, these default values will cause an error. The user must set these parameters to appropriate values as defined in the paragraph "Allowable Parameter Combinations and Considerations."
2. The numbering of PCI Masters, Requests, and Grants range from zero to C_NUM_PCI_MSTRS - 1. Therefore, the default of 4 allows for 4 PCI masters numbered 0 through 3. Setting this to zero is not allowed, while setting to one would be, in effect, a null device. The actual limit on the number of PCI masters is 32 to fit within the Park Register. Adding more masters will increase resource use and slow the response causing a practical limit in the neighborhood of 8 to 15. The IP verifications use 8 masters while implementations with more than 8 have not been tested.
3. This must be within the range 0 to C_NUM_PCI_MSTRS - 1.
4. This reduces resource use and clock latency to grant. However, it also reduces maximum PCI clock frequency and may impact the deterministic behavior of the arbiter operation.

Allowable Parameter Combinations and Considerations

Setting G3, C_RMOV_PROC_IF, to 1 removes the OPB interface and makes the settings of parameters G1, G2 and G4 through G7 irrelevant. If G3 is set 0 then the address range specified by C_BASEADDR and C_HIGHADDR must comprise a complete, contiguous power of two range such that range = 2^n, and the 8 least significant bits of C_BASEADDR must be

zero and the 8 least significant bits of C_HIGHADDR must be 1. The default values accommodate these requirements although C_BASEADDR and C_HIGHADDR are reversed. This forces a user to provide correct values for C_BASEADDR and C_HIGHADDR and think about appropriate memory maps.

The C_NUM_PCI_MSTRS implies inclusion of the number 0. Therefore, the actual number of masters range from zero to C_NUM_PCI_MSTRS minus one. The C_PARK_PCI_MSTR must be within the range of zero to C_NUM_PCI_MSTRS minus one.

OPB_PCI_Arbiter I/O Signals

The I/O signals for the OPB_PCI_Arbiter are listed in Table 4. This lists the signals on the periphery of the IP including IPIF. The interfaces referenced in this table are shown in Figure 1 in the OPB_PCI_Arbiter block diagram.

Table 4: **OPB_PCI_Arbitier I/O Signals**

Grouping		Signal Name	Interface	I/O	Description
PCI Interface	P1	PCI_Clk	PCI	I	PCI Clock
	P2	PCI_Rst_n	PCI	I	PCI Reset, active low
	P3	PCI_Req_n[0: C_NUM_PCI_MSTRS - 1]	PCI	I	PCI Request, active low
	P4	PCI_Gnt_n[0: C_NUM_PCI_MSTRS - 1]	PCI	O	PCI Grant, active low
	P5	PCI_Frame_n	PCI	I	PCI Frame, active low
	P6	PCI_Irdy_n	PCI	I	PCI Ready, active low
System Signals	P7	OPB_Clk	System	I	System clock
	P8	Reset	System	I	System reset
OPB Slave Inputs	P9	OPB_ABus[0:C_OPB_AWIDTH-1]	OPB	I	OPB Address bus
	P10	OPB_Dbus[0:C_OPB_DWIDTH-1]	OPB	I	OPB Data Bus
	P11	OPB_BE[0:(C_OPB_DWIDTH/8)-1]	OPB	I	OPB Byte Enables
	P12	OPB_RNW	OPB	I	OPB Read Not Write
	P13	OPB_select	OPB	I	OPB Select
	P14	OPB_seqAddr	OPB	I	OPB Sequential Address
	P15	OPB_xferAck	OPB	I	OPB Transfer Acknowledge
Outputs	P16	Sln_xferAck	OPB	O	Slave Address Acknowledge
	P17	Sln_errAck	OPB	O	Slave Error Acknowledge
	P18	Sln_retry	OPB	O	Slave Retry
	P19	Sln_toutSup	OPB	O	Slave Time Out Suppress
	P20	Sln_DBus[0:C_OPB_DWIDTH-1]	OPB	O	Slave Data Bus

Parameter/Port Dependencies

The width and sense of two OPB_PCI_Arbiter signals depend upon the selected values of parameters. This is illustrated in Table 5.

Table 5: **Parameter/Port Dependencies**

		Name	Affects	Depends	Relationship Description
Design Parameters	G3	C_RMOV_PROC_IF	P7 - P20		C_RMOV_PROC_IF determines whether or not an OPB_IPIF is used, If there is no OPB interface, then all OPB signals will be disconnected. Disconnected outputs will be open and disconnected inputs will tied to logic zero.
	G8	C_NUM_PCI_MSTRS	P3, P4		C_NUM_PCI_MSTRS determines the number of PCI requests and grants.
I/O Signals	P7 - P20	All OPB and Sln signals plus Reset.		G3	C_RMOV_PROC_IF determines whether or not an OPB_IPIF is used, If there is no OPB interface, then all OPB signals will be disconnected. Disconnected outputs will be open and disconnected inputs will tied to logic zero.
	P3	PCI_Req_n		G8	The number of PCI requests depends on C_NUM_PCI_MSTRS.
	P4	PCI_Gnt_n		G8	The number of PCI grants depends on C_NUM_PCI_MSTRS.

OPB_PCI_Arbiter Register Descriptions

Register Summary

The OPB_PCI_Arbiter contains registers in the IPIF bus interface and the PCI_Arbiter core. Table 6 summarizes the OPB_PCI_Arbiter register set. These registers are removed when C_RMOV_PROC_IF is set to 1, removing the OPB_IPIF.

Table 6: **OPB_PCI_Arbiter Register Summary**

Grouping	Default Address (hex)	Base Address+ Offset (hex)	Register Name	Access Type	Default Value (hex)	Description
Soft Reset and Device MIR	0xC0000040	C_BASEADDR + 0x40	Device S/W Reset	R/W	MIR	S/W Reset port. Write 0x0000000A to activate. Read indicates MIR.
PCI Arbiter Core	0xC0000080	C_BASEADDR + 0x80	MIR	R	0x100001DC	Read Module Identification Register for the IP Block
	0xC0000084	C_BASEADDR + 0x84	CNTRL	R/W	0x00000000	Control Register
	0xC0000088	C_BASEADDR + 0x88	PARK	R/W	0x80000000	Park Master Register

OPB_PCI_Arbiter Module Identification Register

The OPB_PCI_Arbiter Module Identification Register allows for determination and verification of the implemented IP type and version.

Table 7: **OPB_PCI_Arbiter Module Identification Register Bit Definitions**

Bit Label	Name	Core Access	Reset Value	Description
0 - 3	Major Version Number	Read	Version ID 0001	**Module Major Version Number.** Matches the CROM entry for Protocol Block Version for this module.
4 - 10	Minor Version Number	Read	Version ID 0000000	**Module Minor Version Number.** Matches the CROM entry for Protocol Block Version for this module.
11 -15	Rev. Letter	Read	Version ID 00000	**Module Minor Version Letter.** This is a binary encoding of small case letters a through z (00000 - 11001).
16 - 23	Block ID	Read	Assigned by Platform Generator	**Block ID Number.** Distinct number for each OPB_PCI_Arbiter instantiated by Platform Generator.
24 - 31	Block Type	Read	0xDC	**Block Type.** This is an 8 bit identifier unique to each IP type. For OPB_PCI_Arbiter this type is 0xDC.

OPB_PCI_Arbiter Control Register

The OPB_PCI_Arbiter Control Register determines the run time operation of the OPB_PCI_Arbiter controller function. Fields and bits of this register must be programmed to achieve desired functionality. Table 8 defines the bit values for this register.

Table 8: **OPB_PCI_Arbiter Control Register Bit Definitions**

Bit Label	Name	Core Access	Reset Value	Description
0	Fixed_priority	Read / Write	0	**Fixed Priority.** When set the arbitration priority will remain constant with PCI_Request(0) at the highest priority and PCI_Request(n-1) at the lowest priority, where n is the number of PCI Masters defined by the parameter C_NUM_PCI_MSTRS. When cleared the arbitration rotates so that the most recently granted PCI master obtains the lowest priority. 1 = Fixed Priority Arbitration 0 = Rotating Priority Arbitration
1	Use_park_reg	Read / Write	0	**Use Park Register.** When set the PCI Master identified in the Park Register obtains the right to PCI Bus parking. When cleared the PCI Master identified by the parameter C_PARK_PCI_MSTR obtains the right to PCI Bus Parking. 1 = Use the park master defined by the Park Register. 0 = Use the park master defined by C_PARK_PCI_MSTR.
2 - 31	Reserved	Read / Write	0	**Reserved.** Reserved for future expansion

OPB_PCI_Arbiter Park Register

The OPB_PCI_Arbiter Park Register defines the PCI Master that has PCI bus parking privileges when bit one of the control register is set. Table 9 defines the bit values for this register. This is a variable width register and width is determined by the number of PCI Masters indicated by the parameter C_NUM_PCI_MSTRS. The value written must be a one-hot, a single one in a field of zeroes, indication of the PCI Master designated for PCI bus parking privileges. Writing a value that is not one-hot will produce indeterminate PCI Master parking. Application or driver software must ensure that only one-hot values are written to this register.

Table 9: **OPB_PCI_Arbiter Park Register Bit Definitions**

Bit Label	Name	Core Access	Reset Value	Description
0:C_NUM_PCI_MSTRS-1	Park	Read / Write	0x800...	**Park.** Used to define the PCI Master that has PCI bus parking privileges when bit one, Use_park_reg, of the control register is set. Any software, application, or driver for this core must ensure that all writes to this register provide a one-hot value, that is a single one in a field of zeroes. For example: "1000" = Master Zero gets park, there are 4 Masters. "000010" = Master Four gets park, there are 6 Masters
C_NUM_PCI_MSTRS:31	Reserved	Read	0x0	**Reserved.** Reserved for future expansion

OPB_PCI_Arbiter Block Diagram

The top-level block diagram for the OPB_PCI_Arbiter is shown in Figure 1. Note that the OPB_IPIF and Arbiter Registers blocks will be removed when parameter C_RMOV_PROC_IF set to 1. In this case, the appropriate outputs will remain open and associated inputs will be tied to logic zero. The PCI_Sync_Reg and PCI_Grant_Reg blocks may also be removed by the appropriate parameters.

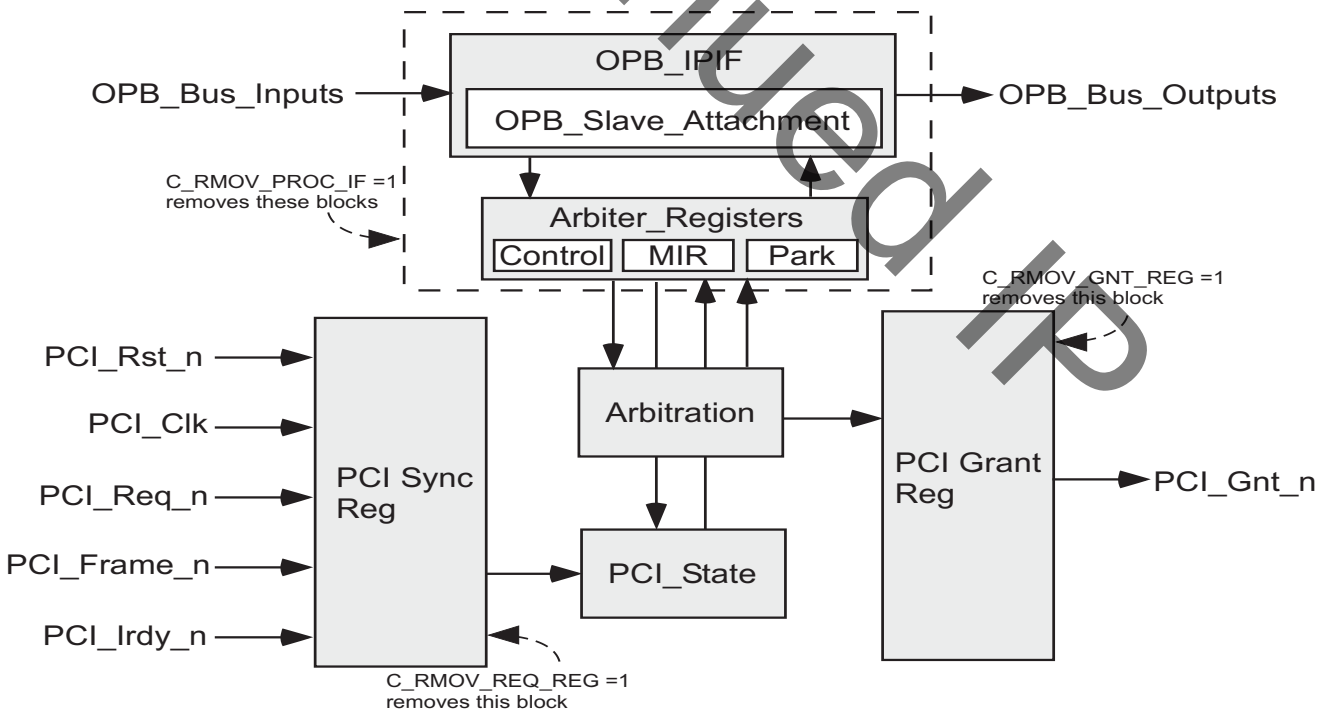

Figure 1: **OPB_PCI_Arbiter Top-level Block Diagram**

Design Implementation

Design Tools

The OPB_PCI_Arbiter was designed and implemented using VHDL. Logic simulations, both functional and post implementation verification, were performed with Model Technology ModelSim.

Target Technology

The intended target technology is Xilinx Virtex, Virtex-II, Virtex-II Pro, Spartan, Spartan-II, and Spartan-3 FPGAs.

Device Utilization and Timing

Since the OPB_PCI_Arbiter is a module that will be used with other design modules in the FPGA, the utilization and timing numbers reported in this section are merely estimates. As the OPB_PCI_Arbiter is combined with other portions of the FPGA design, the utilization of FPGA resources and timing of the OPB_PCI_Arbiter design will vary from the results reported here. The OPB_PCI_Arbiter benchmarks are shown in Table 10 for a Virtex-II -6 FPGA.

Table 10: **OPB_PCI_Arbiter FPGA Performance and Resource Utilization (Virtex-II -6)**

Parameter Values				Device Resources			PCI Clk f_{MAX} (MHz)
C_RMOV_PROC_IF	C_NUM_PCI_MSTRS	C_RMOV_REQ_REG	C_RMOV_GNT_REG	Slices	Slice Flip-Flops	4-input LUTs	f_{MAX}
1	2	0	0	18	25	23	> 100
1	2	0	1	19	23	25	> 100
1	2	1	0	16	23	23	> 100
1	2	1	1	17	19	25	> 100
1	4	0	0	33	33	46	> 100
1	4	0	1	34	29	50	> 100
1	4	1	0	28	27	44	> 100
1	4	1	1	29	23	48	> 100
1	6	0	0	48	41	72	> 100
1	6	0	1	48	35	78	> 100
1	6	1	0	45	33	75	> 100
1	6	1	1	45	27	80	> 100
1	8	0	0	63	49	97	> 100
1	8	0	1	62	41	105	> 100
1	8	1	0	59	39	99	> 100
1	8	1	1	58	31	107	> 100
0	2	0	0	111	91	124	> 100
0	2	0	1	110	90	124	> 100
0	2	1	0	112	89	123	> 100

Table 10: **OPB_PCI_Arbiter FPGA Performance and Resource Utilization (Virtex-II -6)** ***(Continued)***

0	2	1	1	109	85	123	> 100
0	4	0	0	127	104	144	> 100
0	4	0	1	128	100	142	> 100
0	4	1	0	123	94	143	> 100
0	4	1	1	124	94	143	> 100
0	6	0	0	141	112	175	> 100
0	6	0	1	140	108	172	> 100
0	6	1	0	139	108	171	> 100
0	6	1	1	138	104	171	> 100
0	8	0	0	162	132	205	> 100
0	8	0	1	162	124	205	> 100
0	8	1	0	154	122	200	> 100
0	8	1	1	153	14	200	> 100

Reference Documents

The following documents contain reference information important to understanding the OPB_PCI_Arbiter:

- PCI Local Bus Specification, Revision 2.3, March 29, 2002.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
11/05/03	1.0	First released version.
01/23/04	1.1	Updates to TMs and copyright.
02/10/04	1.2	Corrections and added one-hot requirements for Park Register.
04/28/04	1.3	Added PCI specification citation, description of arbitration schemes, and statement regarding the PCI specification guideline.
05/04/04	1.4	Changed Device Block ID default to 1, corrected PCI_arbiter MIR to 0x100001DC, changed synthesis tool to XST, and established 75 MHz verification frequency.
05/05/04	1.5	Added Virtex-4 in family support list and added priority information when used without opb_ipif.
8/5/04	1.6	Updated trademarks and supported device family listing.