Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

PI7C8140A

2-Port PCI-to-PCI Bridge

REVISION 1.01

3545 North First Street, San Jose, CA 95134
Telephone: 1-877-PERICOM, (1-877-737-4266)
Fax: 408-435-1100
Internet: http://www.pericom.com

LIFE SUPPORT POLICY

Pericom Semiconductor Corporation's products are not authorized for use as critical components in life support devices or systems unless a specific written agreement pertaining to such intended use is executed between the manufacturer and an officer of PSC.

1) Life support devices or system are devices or systems which:
   a) Are intended for surgical implant into the body or
   b) Support or sustain life and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2) A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness. Pericom Semiconductor Corporation reserves the right to make changes to its products or specifications at any time, without notice, in order to improve design or performance and to supply the best possible product. Pericom Semiconductor does not assume any responsibility for use of any circuitry described other than the circuitry embodied in a Pericom Semiconductor product. The Company makes no representations that circuitry described herein is free from patent infringement or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent, patent rights or other rights, of Pericom Semiconductor Corporation.

REVISION HISTORY

DATE	REVISION NUMBER	DESCRIPTION
11-13-2003	0.01	First Draft of Datasheet
03-04-2004	0.02	First release of datasheet
03-24-2004	0.03	Corrected reference to the retry counter register in section 2.5.3 from offset 78h to offset 88h. Corrected reference to the chip control register in section 2.5.4 from offset 40h to offset 44h. Changed/revised pin descriptions for P_CLKRUN# and S_CLKRUN# in section 1.2.3. Changed pin descriptions for SCAN_EN and SCAN_TM# in section 1.2.4. Revised pin description for LOO pin in section
05-07-2004	1.00	Added Power consumption and T_{SKEW} data Initial release of the datasheet to the web
03-20-2007	1.01	Removed solutions@pericom.com contact information Removed “Advance Information” from headers

PREFACE

The PI7C8140A datasheet will be enhanced periodically when updated information is available. The technical information in this datasheet is subject to change without notice. This document describes the functionalities of PI7C8140A and provides technical information for designers to design their hardware using PI7C8140A.

This page intentionally left blank.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

INTRODUCTION

Product Description

The PI7C8140A is Pericom Semiconductor's PCI-to-PCI Bridge, designed to be fully compliant with the 32-bit, 66MHz implementation of the *PCI Local Bus Specification, Revision 2.2*. The PI7C8140A supports synchronous bus transactions between devices on the Primary Bus and the Secondary Buses operating up to 66MHz. Both primary and secondary buses must operate at the same frequency. The primary and secondary buses can also operate in concurrent mode, resulting in added increase in system performance.

Product Features

- 32-bit Primary and Secondary Ports run up to 66MHz
- Compliant with the *PCI Local Bus Specification, Revision 2.2*
- Compliant with *PCI-to-PCI Bridge Architecture Specification*, Revision 1.1.
  - All I/O and memory commands
  - Type 1 to Type 0 configuration conversion
  - Type 1 to Type 1 configuration forwarding
  - Type 1 configuration write to special cycle conversion
- Compliant with the *Advanced Configuration Power Interface* (ACPI)
- Compliant with the *PCI Power Management Specification, Revision 1.1.*
- Provides internal arbitration for four secondary bus masters
  - Programmable 2-level priority arbiter
- PCI Clockrun support
- Supports posted write buffers in all directions
- Four 128 byte FIFO's for delay transactions
- Two 128 byte FIFO's for posted memory transactions
- Enhanced address decoding
- 32-bit I/O address range
- 32-bit memory-mapped I/O address range
- 64-bit prefetchable address range
- Extended commercial temperature range 0°C to 85°C
- 3.3V and 5V signaling
- 128-pin QFP package

This page intentionally left blank.

1 SIGNAL DEFINITIONS

1.1 SIGNAL TYPES

SIGNAL TYPE	DESCRIPTION
I	Input only
O	Output only
P	Power
TS	Tri-state bi-directional
STS	Sustained tri-state. Active LOW signal must be pulled HIGH for 1 cycle when deasserting.
OD	Open Drain

1.2 SIGNALS

Signals that end with "#" are active LOW.

1.2.1 PRIMARY BUS INTERFACE SIGNALS

Name	Pin Number	Type	Description
P_AD[31:0]	121, 122, 123, 124, 125, 126, 127, 2, 5, 6, 7, 8, 9, 10, 12, 13, 25, 26, 27, 28, 30, 31, 32, 33, 35, 36, 37, 40, 41, 42, 43, 44	TS	**Primary Address / Data:** Multiplexed address and data bus. Address is indicated by P_FRAME# assertion. Write data is stable and valid when P_IRDY# is asserted and read data is stable and valid when P_TRDY# is asserted. Data is transferred on rising clock edges when both P_IRDY# and P_TRDY# are asserted. During bus idle, PI7C8140A drives P_AD to a valid logic level when P_GNT# is asserted.
P_CBE#[3:0]	3, 14, 24, 34	TS	**Primary Command/Byte Enables:** Multiplexed command field and byte enable field. During address phase, the initiator drives the transaction type on these pins. After that, the initiator drives the byte enables during data phases. During bus idle, PI7C8140A drives P_CBE#[3:0] to a valid logic level when P_GNT# is asserted.
P_PAR	23	TS	**Primary Parity.** Parity is even across P_AD[31:0], P_CBE#[3:0], and P_PAR (i.e. an even number of 1's). P_PAR is an input and is valid and stable one cycle after the address phase (indicated by assertion of P_FRAME#) for address parity. For write data phases, P_PAR is an input and is valid one clock after P_IRDY# is asserted. For read data phase, P_PAR is an output and is valid one clock after P_TRDY# is asserted. Signal P_PAR is tri-stated one cycle after the P_AD lines are tri-stated. During bus idle, PI7C8140A drives P_PAR to a valid logic level when P_GNT# is asserted.
P_FRAME#	15	STS	**Primary FRAME (Active LOW).** Driven by the initiator of a transaction to indicate the beginning and duration of an access. The de-assertion of P_FRAME# indicates the final data phase requested by the initiator. Before being tri-stated, it is driven to a de-asserted state for one cycle.
P_IRDY#	16	STS	**Primary IRDY (Active LOW).** Driven by the initiator of a transaction to indicate its ability to complete current data phase on the primary side. Once asserted in a data phase, it is not de-asserted until the end of the data phase. Before tri-stated, it is driven to a de-asserted state for one cycle.

Name	Pin Number	Type	Description
P_TRDY#	17	STS	**Primary TRDY (Active LOW).** Driven by the target of a transaction to indicate its ability to complete current data phase on the primary side. Once asserted in a data phase, it is not de-asserted until the end of the data phase. Before tri-stated, it is driven to a de-asserted state for one cycle.
P_DEVSEL#	18	STS	**Primary Device Select (Active LOW).** Asserted by the target indicating that the device is accepting the transaction. As a master, PI7C8140A waits for the assertion of this signal within 5 cycles of P_FRAME# assertion; otherwise, terminate with master abort. Before tri-stated, it is driven to a de-asserted state for one cycle.
P_STOP#	19	I	**Primary STOP (Active LOW).** Asserted by the target indicating that the target is requesting the initiator to stop the current transaction. Before tri-stated, it is driven to a de-asserted state for one cycle.
P_IDSEL	4	I	**Primary ID Select.** Used as a chip select line for Type 0 configuration access to PI7C8140A configuration space.
P_PERR#	21	STS	**Primary Parity Error (Active LOW).** Asserted when a data parity error is detected for data received on the primary interface. Before being tri-stated, it is driven to a de-asserted state for one cycle.
P_SERR#	22	OD	**Primary System Error (Active LOW).** Can be driven LOW by any device to indicate a system error condition. PI7C8140A drives this pin on: ▪ Address parity error ▪ Posted write data parity error on target bus ▪ Secondary S_SERR# asserted ▪ Master abort during posted write transaction ▪ Target abort during posted write transaction ▪ Posted write transaction discarded ▪ Delayed write request discarded ▪ Delayed read request discarded ▪ Delayed transaction master timeout This signal requires an external pull-up resistor for proper operation.
P_REQ#	119	TS	**Primary Request (Active LOW):** This is asserted by PI7C8140A to indicate that it wants to start a transaction on the primary bus. PI7C8140A de-asserts this pin for at least 2 PCI clock cycles before asserting it again.
P_GNT#	118	I	**Primary Grant (Active LOW):** When asserted, PI7C8140A can access the primary bus. During idle and P_GNT# asserted, PI7C8140A will drive P_AD, P_CBE, and P_PAR to valid logic levels.
P_RST#	116	I	**Primary RESET (Active LOW):** When P_RST# is active, all PCI signals should be asynchronously tri-stated.

1.2.2 SECONDARY BUS INTERFACE SIGNALS

Name	Pin Number	Type	Description
S_AD[31:0]	95, 94, 92, 91, 90, 89, 88, 87, 85, 83, 82, 81, 80, 79, 78, 77, 63, 62, 61, 60, 59, 57, 56, 55, 53, 52, 51, 50, 48, 47, 46, 45	TS	**Secondary Address/Data:** Multiplexed address and data bus. Address is indicated by S_FRAME# assertion. Write data is stable and valid when S_IRDY# is asserted and read data is stable and valid when S_TRDY# is asserted. Data is transferred on rising clock edges when both S_IRDY# and S_TRDY# are asserted. During bus idle, PI7C8140A drives S_AD to a valid logic level when S_GNT# is asserted respectively.
S_CBE#[3:0]	86, 76, 66, 54	TS	**Secondary Command/Byte Enables:** Multiplexed command field and byte enable field. During address phase, the initiator drives the transaction type on these pins. The initiator then drives the byte enables during data phases. During bus idle, PI7C8140A drives S_CBE#[3:0] to a valid logic level when the internal grant is asserted.

Name	Pin Number	Type	Description
S_PAR	67	TS	**Secondary Parity:** Parity is even across S_AD[31:0], S_CBE#[3:0], and S_PAR (i.e. an even number of 1's). S_PAR is an input and is valid and stable one cycle after the address phase (indicated by assertion of S_FRAME#) for address parity. For write data phases, S_PAR is an input and is valid one clock after S_IRDY# is asserted. For read data phase, S_PAR is an output and is valid one clock after S_TRDY# is asserted. Signal S_PAR is tri-stated one cycle after the S_AD lines are tri-stated. During bus idle, PI7C8140A drives S_PAR to a valid logic level when the internal grant is asserted.
S_FRAME#	74	STS	**Secondary FRAME (Active LOW):** Driven by the initiator of a transaction to indicate the beginning and duration of an access. The de-assertion of S_FRAME# indicates the final data phase requested by the initiator. Before being tri-stated, it is driven to a de-asserted state for one cycle.
S_IRDY#	73	STS	**Secondary IRDY (Active LOW):** Driven by the initiator of a transaction to indicate its ability to complete current data phase on the secondary side. Once asserted in a data phase, it is not de-asserted until the end of the data phase. Before tri-stated, it is driven to a de-asserted state for one cycle.
S_TRDY#	72	STS	**Secondary TRDY (Active LOW):** Driven by the target of a transaction to indicate its ability to complete current data phase on the secondary side. Once asserted in a data phase, it is not de-asserted until the end of the data phase. Before tri-stated, it is driven to a de-asserted state for one cycle.
S_DEVSEL#	71	STS	**Secondary Device Select (Active LOW):** Asserted by the target indicating that the device is accepting the transaction. As a master, PI7C8140A waits for the assertion of this signal within 5 cycles of S_FRAME# assertion; otherwise, terminate with master abort. Before tri-stated, it is driven to a de-asserted state for one cycle.
S_STOP#	70	STS	**Secondary STOP (Active LOW):** Asserted by the target indicating that the target is requesting the initiator to stop the current transaction. Before tri-stated, it is driven to a de-asserted state for one cycle.
S_PERR#	69	STS	**Secondary Parity Error (Active LOW):** Asserted when a data parity error is detected for data received on the secondary interface. Before being tri-stated, it is driven to a de-asserted state for one cycle.
S_SERR#	68	I	**Secondary System Error (Active LOW):** Can be driven LOW by any device to indicate a system error condition.
S_REQ#[3:0]	99, 98, 97, 96	I	**Secondary Request (Active LOW):** This is asserted by an external device to indicate that it wants to start a transaction on the secondary bus. The input is externally pulled up through a resistor to VDD.
S_GNT#[3:0]	104, 103, 101, 100	TS	**Secondary Grant (Active LOW):** PI7C8140A asserts these pins to allow external masters to access the secondary bus. PI7C8140A de-asserts these pins for at least 2 PCI clock cycles before asserting it again. During idle and S_GNT# deasserted, PI7C8140A will drive S_AD, S_CBE, and S_PAR.
S_RST#	105	O	**Secondary RESET (Active LOW):** Asserted when any of the following conditions are met: 1. Signal P_RESET# is asserted. 2. Secondary reset bit in bridge control register in configuration space is set. When asserted, all control signals are tri-stated and zeroes are driven on S_AD, S_CBE, and S_PAR.

1.2.3 CLOCK SIGNALS

Name	Pin Number	Type	Description
P_CLK	117	I	**Primary Clock Input:** Provides timing for all transactions on the primary interface.
S_CLKOUT[3:0]	110, 109, 108, 107	O	**Secondary Clock Output:** Provides secondary clocks phase synchronous with the P_CLK.
P_CLKRUN#	115	TS	**Primary Clock Run:** Allows main system to stop the primary clock based on the specifications in the *PCI Mobile Design Guide*, Revision 1.0. If unused, this pin should be tied to ground to signify that P_CLK is always running.
S_CLKRUN#	112	TS	**Secondary Clock Run:** Allows main system to slow down or stop the secondary clock and is controlled by the primary or bit[4] offset 6Fh. If the secondary devices do not support CLKRUN, this pin should be pulled LOW by a 300 ohm resistor.

1.2.4 MISCELLANEOUS SIGNALS

Name	Pin Number	Type	Description
ENUM#	113	O	**Hot Swap Status Indicator:** The output of ENUM# indicates to the system that an insertion has occurred or that an extraction is about to occur.
LOO	114	I/O	**Hot Swap LED:** The output of this pin lights an LED to indicate insertion or removal ready status. This pin may also be used as a input or detect pin. Every 500us, the pin tri-states for 8 primary PCI clock cycles to sample the status.
SCAN_TM#	65	I	**Full-Scan Test Mode Enable:** For normal operation, pull SCAN_TM# to HIGH. Manufacturing test pin.
SCAN_EN	106	I/O	**Full-Scan Enable Control:** For normal operation, SCAN_TM# should be pulled HIGH and SCAN_EN becomes an output with logic 0. Manufacturing test pin.

1.2.5 POWER AND GROUND

Name	Pin Number	Type	Description
VDD	1, 20, 39, 58, 84, 102, 120	P	**Power:** 3.3V power
VSS	11, 29, 38, 49, 64, 75, 93, 111, 128	P	**Ground**

1.3 PIN LIST – 128-PIN QFP

Pin Number	Name	Type	Pin Number	Name	Type
1	VDD	P	2	P_AD[24]	TS
3	P_CBE#[3]	TS	4	P_IDSEL	I
5	P_AD[23]	TS	6	P_AD[22]	TS
7	P_AD[21]	TS	8	P_AD[20]	TS
9	P_AD[19]	TS	10	P_AD[18]	TS
11	VSS	P	12	P_AD[17]	TS
13	P_AD[16]	TS	14	P_CBE#[2]	TS
15	P_FRAME#	STS	16	P_IRDY#	STS
17	T_RDY#	STS	18	P_DEVSEL#	STS
19	P_STOP#	I	20	VDD	P
21	P_PERR#	STS	22	P_SERR#	OD
23	P_PAR	TS	24	P_CBE#[1]	TS
25	P_AD[15]	TS	26	P_AD[14]	TS
27	P_AD[13]	TS	28	P_AD[12]	TS
29	VSS	P	30	P_AD[11]	TS
31	P_AD[10]	TS	32	P_AD[9]	TS
33	P_AD[8]	TS	34	P_CBE#[0]	TS
35	P_AD[7]	TS	36	P_AD[6]	TS
37	P_AD[5]	TS	38	VSS	P
39	VDD	P	40	P_AD[4]	TS
41	P_AD[3]	TS	42	P_AD[2]	TS
43	P_AD[1]	TS	44	P_AD[0]	TS
45	S_AD[0]	TS	46	S_AD[1]	TS
47	S_AD[2]	TS	48	S_AD[3]	TS
49	VSS	P	50	S_AD[4]	TS
51	S_AD[5]	TS	52	S_AD[6]	TS
53	S_AD[7]	TS	54	S_CBE#[0]	TS
55	S_AD[8]	TS	56	S_AD[9]	TS
57	S_AD[10]	TS	58	VDD	P
59	S_AD[11]	TS	60	S_AD[12]	TS
61	S_AD[13]	TS	62	S_AD[14]	TS
63	S_AD[15]	TS	64	VSS	P
65	SCAN_TM#	I	66	S_CBE#[1]	TS
67	S_PAR	TS	68	S_SERR#	I
69	S_PERR#	STS	70	S_STOP#	STS
71	S_DEVSEL#	STS	72	S_TRDY#	STS
73	S_IRDY#	STS	74	S_FRAME#	STS
75	VSS	P	76	S_CBE#[2]	TS
77	S_AD[16]	TS	78	S_AD[17]	TS
79	S_AD[18]	TS	80	S_AD[19]	TS
81	S_AD[20]	TS	82	S_AD[21]	TS
83	S_AD[22]	TS	84	VDD	P
85	S_AD[23]	TS	86	S_CBE#[3]	TS
87	S_AD[24]	TS	88	S_AD[25]	TS
89	S_AD[26]	TS	90	S_AD[27]	TS
91	S_AD[28]	TS	92	S_AD[29]	TS
93	VSS	P	94	S_AD[30]	TS
95	S_AD[31]	TS	96	S_REQ#[0]	I
97	S_REQ#[1]	I	98	S_REQ#[2]	I
99	S_REQ#[3]	I	100	S_GNT#[0]	TS
101	S_GNT#[1]	TS	102	VDD	P
103	S_GNT#[2]	TS	104	S_GNT#[3]	TS
105	S_RST#	O	106	SCAN_EN	I/O
107	S_CLKOUT[0]	O	108	S_CLKOUT[1]	O
109	S_CLKOUT[2]	O	110	S_CLKOUT[3]	O
111	VSS	P	112	S_CLKRUN#	TS
113	ENUM#	O	114	LOO	I/O

Pin Number	Name	Type	Pin Number	Name	Type
115	P_CLKRUN#	TS	116	P_RST#	I
117	P_CLK	I	118	P_GNT#	I
119	P_REQ#	TS	120	VDD	P
121	P_AD[31]	TS	122	P_AD[30]	TS
123	P_AD[29]	TS	124	P_AD[28]	TS
125	P_AD[27]	TS	126	P_AD[26]	TS
127	P_AD[25]	TS	128	VSS	P

2 PCI BUS OPERATION

This Chapter offers information about PCI transactions, transaction forwarding across the bridge, and transaction termination. The bridge has two 128-byte FIFO's for buffering of upstream and downstream transactions. These hold addresses, data, commands, and byte enables that are used for write transactions. The bridge also has an additional four 128-byte FIFO's that hold addresses, data, commands, and byte enables for read transactions.

2.1 TYPES OF TRANSACTIONS

This section provides a summary of PCI transactions performed by the bridge. Table 2-1 lists the command code and name of each PCI transaction. The Master and Target columns indicate support for each transaction when the bridge initiates transactions as a master, on the primary (P) and secondary (S) buses, and when the bridge responds to transactions as a target, on the primary (P) and secondary (S) buses.

Table 2-1. PCI Transactions

Types of Transactions		Initiates as Master		Responds as Target	
		Primary	Secondary	Primary	Secondary
0000	Interrupt Acknowledge	N	N	N	N
0001	Special Cycle	Y	Y	N	N
0010	I/O Read	Y	Y	Y	Y
0011	I/O Write	Y	Y	Y	Y
0100	Reserved	N	N	N	N
0101	Reserved	N	N	N	N
0110	Memory Read	Y	Y	Y	Y
0111	Memory Write	Y	Y	Y	Y
1000	Reserved	N	N	N	N
1001	Reserved	N	N	N	N
1010	Configuration Read	N	Y	Y	N
1011	Configuration Write	Y (Type 1 only)	Y	Y	Y (Type 1 only)
1100	Memory Read Multiple	Y	Y	Y	Y
1101	Dual Address Cycle	Y	Y	Y	Y
1110	Memory Read Line	Y	Y	Y	Y
1111	Memory Write and Invalidate	Y	Y	Y	Y

As indicated in Table 2-1, the following PCI commands are not supported by the bridge:

- The bridge never initiates a PCI transaction with a reserved command code and, as a target, the bridge ignores reserved command codes.
- The bridge does not generate interrupt acknowledge transactions. The bridge ignores interrupt acknowledge transactions as a target.

- The bridge does not respond to special cycle transactions. The bridge cannot guarantee delivery of a special cycle transaction to downstream buses because of the broadcast nature of the special cycle command and the inability to control the transaction as a target. To generate special cycle transactions on other PCI buses, either upstream or downstream, Type 1 configuration write must be used.
- The bridge neither generates Type 0 configuration transactions on the primary PCI bus nor responds to Type 0 configuration transactions on the secondary PCI buses.

2.2 SINGLE ADDRESS PHASE

A 32-bit address uses a single address phase. This address is driven on P_AD[31:0], and the bus command is driven on P_CBE[3:0]. The bridge supports the linear increment address mode only, which is indicated when the lowest two address bits are equal to zero. If either of the lowest two address bits is nonzero, the bridge automatically disconnects the transaction after the first data transfer.

2.3 DEVICE SELECT (DEVSEL#) GENERATION

The bridge always performs positive address decoding (medium decode) when accepting transactions on either the primary or secondary buses. The bridge never does subtractive decode.

2.4 DATA PHASE

The address phase of a PCI transaction is followed by one or more data phases. A data phase is completed when IRDY# and either TRDY# or STOP# are asserted. A transfer of data occurs only when both IRDY# and TRDY# are asserted during the same PCI clock cycle. The last data phase of a transaction is indicated when FRAME# is de-asserted and both TRDY# and IRDY# are asserted, or when IRDY# and STOP# are asserted. See Section 2.8 for further discussion of transaction termination.

Depending on the command type, the bridge can support multiple data phase PCI transactions. For detailed descriptions of how the bridge imposes disconnect boundaries, see Section 2.5.4 for write address boundaries and Section 2.6.4 read address boundaries.

2.5 WRITE TRANSACTIONS

Write transactions are treated as either posted write or delayed write transactions. Table 2-2 shows the method of forwarding used for each type of write operation.

Table 2-2. Write Transaction Forwarding

Type of Transaction	Type of Forwarding
Memory Write	Posted (except VGA memory)
Memory Write and Invalidate	Posted
Memory Write to VGA memory	Delayed
I/O Write	Delayed
Type 1 Configuration Write	Delayed

2.5.1 MEMORY WRITE TRANSACTIONS

Posted write forwarding is used for "Memory Write" and "Memory Write and Invalidate" transactions.

When the bridge determines that a memory write transaction is to be forwarded across the bridge, the bridge asserts DEVSEL# with medium timing and TRDY# in the next cycle, provided that enough buffer space is available in the posted memory write queue for the address and at least one DWORD of data. Under this condition, the bridge accepts write data without obtaining access to the target bus. The bridge can accept one DWORD of write data every PCI clock cycle. That is, no target wait state is inserted. The write data is stored in an internal posted write buffers and is subsequently delivered to the target. The bridge continues to accept write data until one of the following events occurs:

- The initiator terminates the transaction by de-asserting FRAME# and IRDY#.
- An internal write address boundary is reached, such as a cache line boundary or an aligned 4KB boundary, depending on the transaction type.
- The posted write data buffer fills up.

When one of the last two events occurs, the bridge returns a target disconnect to the requesting initiator on this data phase to terminate the transaction.

Once the posted write data moves to the head of the posted data queue, the bridge asserts its request on the target bus. This can occur while the bridge is still receiving data on the initiator bus. When the grant for the target bus is received and the target bus is detected in the idle condition, the bridge asserts FRAME# and drives the stored write address out on the target bus. On the following cycle, the bridge drives the first DWORD of write data and continues to transfer write data until all write data corresponding to that transaction is delivered, or until a target termination is received. As long as write data exists in the queue, the bridge can drive one DWORD of write data each PCI clock cycle; that is, no master wait states are inserted. If write data is flowing through the bridge and the initiator stalls, the bridge will signal the last data phase for the current transaction at the target bus if the queue empties. The bridge will restart the follow-on transactions if the queue has new data.

The bridge ends the transaction on the target bus when one of the following conditions is met:

- All posted write data has been delivered to the target.
- The target returns a target disconnect or target retry (the bridge starts another transaction to deliver the rest of the write data).
- The target returns a target abort (the bridge discards remaining write data).
- The master latency timer expires, and the bridge no longer has the target bus grant (the bridge starts another transaction to deliver remaining write data).

Section 2.8.3.2 provides detailed information about how the bridge responds to target termination during posted write transactions.

2.5.2 MEMORY WRITE AND INVALIDATE

Posted write forwarding is used for Memory Write and Invalidate transactions.

If offset 74h bits [8:7] = 11, the bridge disconnects Memory Write and Invalidate commands at aligned cache line boundaries. The cache line size value in the cache line size register gives the number of DWORD in a cache line.

If offset 74h bits [8:7] = 00, the bridge converts Memory Write and Invalidate transactions to Memory Write transactions at the destination.

If the value in the cache line size register does meet the memory write and invalidate conditions, the bridge returns a target disconnect to the initiator on a cache line boundary.

2.5.3 DELAYED WRITE TRANSACTIONS

Delayed write forwarding is used for I/O write transactions and Type 1 configuration write transactions.

A delayed write transaction guarantees that the actual target response is returned back to the initiator without holding the initiating bus in wait states. A delayed write transaction is limited to a single DWORD data transfer.

When a write transaction is first detected on the initiator bus, and the bridge forwards it as a delayed transaction, the bridge claims the access by asserting DEVSEL# and returns a target retry to the initiator. During the address phase, the bridge samples the bus command, address, and address parity one cycle later. After IRDY# is asserted, the bridge also samples the first data DWORD, byte enable bits, and data parity. This information is placed into the delayed transaction queue. The transaction is queued only if no other existing delayed transactions have the same address and command, and if the delayed transaction queue is not full. When the delayed write transaction moves to the head of the delayed transaction queue and all ordering constraints with posted data are satisfied. The bridge initiates the transaction on the target bus. The bridge transfers the write data to the target. If the bridge receives a target retry in response to the write transaction on the target bus, it continues to repeat the write transaction until the data transfer is completed, or until an error condition is encountered.

If the bridge is unable to deliver write data after 2^{24} (default) or 2^{32} (maximum) attempts, the bridge will report a system error. The bridge also asserts P_SERR# if the primary SERR# enable bit is set in the command register. See Section 5.4 for information on the assertion of P_SERR#. When the initiator repeats the same write transaction (same command, address, byte enable bits, and data), and the completed delayed transaction is at the head of the queue, the bridge claims the access by asserting DEVSEL# and returns TRDY# to the initiator, to indicate that the write data was transferred. If the initiator requests multiple DWORD, the bridge also asserts STOP# in conjunction with TRDY# to signal a target disconnect. Note that only those bytes of write data with valid byte enable bits are compared. If any of the byte enable bits are turned off (driven HIGH), the corresponding byte of write data is not compared.

If the initiator repeats the write transaction before the data has been transferred to the target, the bridge returns a target retry to the initiator. The bridge continues to return a target retry to the initiator until write data is delivered to the target, or until an error condition is encountered. When the write transaction is repeated, the bridge does not make a new entry into the delayed transaction queue. Section 2.8.3.1 provides detailed information about how the bridge responds to target termination during delayed write transactions.

The bridge implements a discard timer that starts counting when the delayed write completion is at the head of the delayed transaction completion queue. The initial value of this timer can be set to the retry counter register offset 88h.

If the initiator does not repeat the delayed write transaction before the discard timer expires, the bridge

discards the delayed write completion from the delayed transaction completion queue. The bridge also conditionally asserts P_SERR# (see Section 5.4).

2.5.4 WRITE TRANSACTION BOUNDARIES

The bridge imposes internal address boundaries when accepting write data. The aligned address boundaries are used to prevent the bridge from continuing a transaction over a device address boundary and to provide an upper limit on maximum latency. The bridge returns a target disconnect to the initiator when it reaches the aligned address boundaries under conditions shown in Table 2-3.

Table 2-3. Write Transaction Disconnect Address Boundaries

Type of Transaction	Condition	Aligned Address Boundary
Delayed Write	All	Disconnects after one data transfer
Posted Memory Write	Memory write disconnect control bit = 0(1)	4KB aligned address boundary
Posted Memory Write	Memory write disconnect control bit = 1(1)	Disconnects at cache line boundary
Posted Memory Write and Invalidate	Cache line size ≠ 1, 2, 4, 8, 16	4KB aligned address boundary
Posted Memory Write and Invalidate	Cache line size = 1, 2, 4, 8, 16	Cache line boundary if posted memory write data FIFO does not have enough space for the cache line

Note 1. Memory write disconnect control bit is bit 1 of the chip control register at offset 44h in the configuration space.

2.5.5 BUFFERING MULTIPLE WRITE TRANSACTIONS

The bridge continues to accept posted memory write transactions as long as space for at least one DWORD of data in the posted write data buffer remains. If the posted write data buffer fills before the initiator terminates the write transaction, the bridge returns a target disconnect to the initiator.

Delayed write transactions are posted as long as at least one open entry in the delayed transaction queue exists. Therefore, several posted and delayed write transactions can exist in data buffers at the same time. See Chapter 5 for information about how multiple posted and delayed write transactions are ordered.

2.5.6 FAST BACK-TO-BACK TRANSACTIONS

The bridge can recognize and post fast back-to-back write transactions. When the bridge cannot accept the second transaction because of buffer space limitations, it returns a target retry to the initiator. The fast back-to-back enable bit must be set in the command register for upstream write transactions, and in the bridge control register for downstream write transactions.

2.6 READ TRANSACTIONS

Delayed read forwarding is used for all read transactions crossing the bridge. Delayed read transactions are treated as either prefetchable or non-prefetchable. Table 2-5 shows the read behavior, prefetchable or non-prefetchable, for each type of read operation.

2.6.1 PREFETCHABLE READ TRANSACTIONS

A prefetchable read transaction is a read transaction where the bridge performs speculative DWORD reads, transferring data from the target before it is requested from the initiator. This behavior allows a prefetchable read transaction to consist of multiple data transfers. However, byte enable bits cannot be forwarded for all data phases as is done for the single data phase of the non-prefetchable read transaction. For prefetchable read transactions, the bridge forces all byte enable bits to be turned on for all data phases.

Prefetchable behavior is used for memory read line and memory read multiple transactions, as well as for memory read transactions that fall into prefetchable memory space.

The amount of data that is pre-fetched depends on the type of transaction. The amount of pre-fetching may also be affected by the amount of free buffer space available in the bridge, and by any read address boundaries encountered.

Pre-fetching should not be used for those read transactions that have side effects in the target device, that is, control and status registers, FIFO's, and so on. The target device's base address register or registers indicate if a memory address region is prefetchable.

2.6.2 DYNAMIC PREFETCHING CONTROL

For prefetchable reads described in the previous section, the prefetching length is normally predefined and cannot be changed once it is set. This may cause some inefficiency as the prefetching length determined could be larger or smaller than the actual data being prefetched. To make prefetching more efficient, PI7C8140A incorporates dynamic prefetching control logic. This logic regulates the different PCI memory read commands (MR – memory read, MRL – memory read line, and MRM – memory read multiple) to improve memory read burst performance. The bridge tracks every memory read burst transaction and tallies the status. By using the status information, the bridge can determine to increase, reduce, or keep the same cache line length to be prefetched. Over time, the bridge can better match the correct cache line setting to the length of data being requested. The dynamic prefetching control logic is set with bits[3:2] offset 48h.

2.6.3 NON-PREFETCHABLE READ TRANSACTIONS

A non-prefetchable read transaction is a read transaction where the bridge requests one and only one DWORD from the target and disconnects the initiator after delivery of the first DWORD of read data. Unlike prefetchable read transactions, the bridge forwards the read byte enable information for the data phase.

Non-prefetchable behavior is used for I/O and configuration read transactions, as well as for memory read transactions that fall into non-prefetchable memory space.

If extra read transactions could have side effects, for example, when accessing a FIFO, use non-prefetchable read transactions to those locations. Accordingly, if it is important to retain the value of the byte enable bits during the data phase, use non-prefetchable read transactions. If these locations are mapped in memory space, use the memory read command and map the target into non-prefetchable (memory-mapped I/O) memory space to use non-prefetching behavior.

2.6.4 READ PREFETCH ADDRESS BOUNDARIES

The bridge imposes internal read address boundaries on read pre-fetched data. When a read transaction reaches one of these aligned address boundaries, the bridge stops pre-fetched data, unless the target signals a target disconnect before the read pre-fetched boundary is reached. When the bridge finishes transferring this read data to the initiator, it returns a target disconnect with the last data transfer, unless the initiator completes the transaction before all pre-fetched read data is delivered. Any leftover pre-fetched data is discarded.

Prefetchable read transactions in flow-through mode pre-fetch to the nearest aligned 4KB address boundary, or until the initiator de-asserts FRAME_L. Section 2.6.7 describes flow-through mode during read operations.

Table 2-4 shows the read prefetch address boundaries for read transactions during non-flow-through mode.

Table 2-4. Read Prefetch Address Boundaries

Type of Transaction	Address Space	Cache Line Size (CLS)	Prefetch Aligned Address Boundary
Configuration Read	-	*	One DWORD (no prefetch)
I/O Read	-	*	One DWORD (no prefetch)
Memory Read	Non-Prefetchable	*	One DWORD (no prefetch)
Memory Read	Prefetchable	CLS = 0 or 16	16-DWORD aligned address boundary
Memory Read	Prefetchable	CLS = 1, 2, 4, 8, 16	Cache line address boundary
Memory Read Line	-	CLS = 0 or 16	16-DWORD aligned address boundary
Memory Read Line	-	CLS = 1, 2, 4, 8, 16	Cache line boundary
Memory Read Multiple	-	CLS = 0 or 16	32-DWORD aligned address boundary
Memory Read Multiple	-	CLS = 1, 2, 4, 8, 16	2X of cache line boundary

- does not matter if it is prefetchable or non-prefetchable
* don't care

Table 2-5. Read Transaction Prefetching

Type of Transaction	Read Behavior
I/O Read	Prefetching never allowed
Configuration Read	Prefetching never allowed
Memory Read	Downstream: Prefetching used if address is prefetchable space
	Upstream: Prefetching used or programmable
Memory Read Line	Prefetching always used
Memory Read Multiple	Prefetching always used

See Section 3.3 for detailed information about prefetchable and non-prefetchable address spaces.

2.6.5 DELAYED READ REQUESTS

The bridge treats all read transactions as delayed read transactions, which means that the read request from the initiator is posted into a delayed transaction queue. Read data from the target is placed in the read data queue directed toward the initiator bus interface and is transferred to the initiator when the initiator repeats the read transaction.

When the bridge accepts a delayed read request, it first samples the read address, read bus command, and address parity. When IRDY# is asserted, the bridge then samples the byte enable bits for the first data phase. This information is entered into the delayed transaction queue. The bridge terminates the transaction by signaling a target retry to the initiator. Upon reception of the target retry, the initiator is

required to continue to repeat the same read transaction until at least one data transfer is completed, or until a target response (target abort or master abort) other than a target retry is received.

2.6.6 DELAYED READ COMPLETION WITH TARGET

When delayed read request reaches the head of the delayed transaction queue, the bridge arbitrates for the target bus and initiates the read transaction only if all previously queued posted write transactions have been delivered. The bridge uses the exact read address and read command captured from the initiator during the initial delayed read request to initiate the read transaction. If the read transaction is a non-prefetchable read, the bridge drives the captured byte enable bits during the next cycle. If the transaction is a prefetchable read transaction, it drives all byte enable bits to zero for all data phases. If the bridge receives a target retry in response to the read transaction on the target bus, it continues to repeat the read transaction until at least one data transfer is completed, or until an error condition is encountered. If the transaction is terminated via normal master termination or target disconnect after at least one data transfer has been completed, the bridge does not initiate any further attempts to read more data.

If the bridge is unable to obtain read data from the target after 2^{24} (default) or 2^{32} (maximum) attempts, the bridge will report system error. The number of attempts is programmable. The bridge also asserts P_SERR# if the primary SERR# enable bit is set in the command register. See Section 5.4 for information on the assertion of P_SERR#.

Once the bridge receives DEVSEL# and TRDY# from the target, it transfers the data read to the opposite direction read data queue, pointing toward the opposite inter-face, before terminating the transaction. For example, read data in response to a downstream read transaction initiated on the primary bus is placed in the upstream read data queue. The bridge can accept one DWORD of read data each PCI clock cycle; that is, no master wait states are inserted. The number of DWORD's transferred during a delayed read transaction depends on the conditions given in Table 2-4 (assuming no disconnect is received from the target).

2.6.7 DELAYED READ COMPLETION ON INITIATOR BUS

When the transaction has been completed on the target bus, and the delayed read data is at the head of the read data queue, and all ordering constraints with posted write transactions have been satisfied, the bridge transfers the data to the initiator when the initiator repeats the transaction. For memory read transactions, the bridge aliases the memory read, memory read line, and memory read multiple bus commands when matching the bus command of the transaction to the bus command in the delayed transaction queue. The bridge returns a target disconnect along with the transfer of the last DWORD of read data to the initiator. If the bridge initiator terminates the transaction before all read data has been transferred, the remaining read data left in data buffers is discarded.

When the master repeats the transaction and starts transferring prefetchable read data from data buffers while the read transaction on the target bus is still in progress and before a read boundary is reached on the target bus, the read transaction starts operating in flow-through mode. Because data is flowing through the data buffers from the target to the initiator, long read bursts can then be sustained. In this case, the read transaction is allowed to continue until the initiator terminates the transaction, or until an aligned 4KB address boundary is reached, or until the buffer fills, whichever comes first. When the buffer empties, the bridge reflects the stalled condition to the initiator by disconnecting the initiator with data. The initiator may retry the transaction later if data are needed. If the initiator does not need any more data, the initiator will not continue the disconnected transaction. In this case, the bridge will

start the master timeout timer. The remaining read data will be discarded after the master timeout timer expires. To provide better latency, if there are any other pending data for other transactions in the RDB (Read Data Buffer), the remaining read data will be discarded even though the master timeout timer has not expired.

The bridge implements a master timeout timer that starts counting when the delayed read completion is at the head of the delayed transaction queue, and the read data is at the head of the read data queue. The initial value of this timer is programmable through configuration register. If the initiator does not repeat the read transaction and before the master timeout timer expires (2^{15} default), the bridge discards the read transaction and read data from its queues. The bridge also conditionally asserts P_SERR# (see Section 5.4).

The bridge has the capability to post multiple delayed read requests, up to a maximum of four in each direction. If an initiator starts a read transaction that matches the address and read command of a read transaction that is already queued, the current read command is not posted as it is already contained in the delayed transaction queue.

See Section 4 for a discussion of how delayed read transactions are ordered when crossing the bridge.

2.6.8 FAST BACK-TO-BACK READ TRANSACTIONS

The bridge can recognize fast back-to-back read transactions.

2.7 CONFIGURATION TRANSACTIONS

Configuration transactions are used to initialize a PCI system. Every PCI device has a configuration space that is accessed by configuration commands. All registers are accessible in configuration space only.

In addition to accepting configuration transactions for initialization of its own configuration space, the bridge also forwards configuration transactions for device initialization in hierarchical PCI systems, as well as for special cycle generation.

To support hierarchical PCI bus systems, two types of configuration transactions are specified: Type 0 and Type 1.

Type 0 configuration transactions are issued when the intended target resides on the same PCI bus as the initiator. A Type 0 configuration transaction is identified by the configuration command and the lowest two bits of the address set to 00b.

Type 1 configuration transactions are issued when the intended target resides on another PCI bus, or when a special cycle is to be generated on another PCI bus. A Type 1 configuration command is identified by the configuration command and the lowest two address bits set to 01b.

The register number is found in both Type 0 and Type 1 formats and gives the DWORD address of the configuration register to be accessed. The function number is also included in both Type 0 and Type 1 formats and indicates which function of a multifunction device is to be accessed. For single-function devices, this value is not decoded. The addresses of Type 1 configuration transaction include a 5-bit field designating the device number that identifies the device on the target PCI bus that is to be

accessed. In addition, the bus number in Type 1 transactions specifies the PCI bus to which the transaction is targeted.

2.7.1 TYPE 0 ACCESS TO PI7C8140A

The configuration space is accessed by a Type 0 configuration transaction on the primary interface. The configuration space cannot be accessed from the secondary bus. The bridge responds to a Type 0 configuration transaction by asserting P_DEVSEL# when the following conditions are met during the address phase:

- The bus command is a configuration read or configuration write transaction.
- Lowest two address bits P_AD[1:0] must be 00b.
- Signal P_IDSEL must be asserted.

The bridge limits all configuration access to a single DWORD data transfer and returns target-disconnect with the first data transfer if additional data phases are requested. Because read transactions to configuration space do not have side effects, all bytes in the requested DWORD are returned, regardless of the value of the byte enable bits.

Type 0 configuration write and read transactions do not use data buffers; that is, these transactions are completed immediately, regardless of the state of the data buffers. The bridge ignores all Type 0 transactions initiated on the secondary interface.

2.7.2 TYPE 1 TO TYPE 0 CONVERSION

Type 1 configuration transactions are used specifically for device configuration in a hierarchical PCI bus system. A PCI-to-PCI bridge is the only type of device that should respond to a Type 1 configuration command. Type 1 configuration commands are used when the configuration access is intended for a PCI device that resides on a PCI bus other than the one where the Type 1 transaction is generated.

The bridge performs a Type 1 to Type 0 translation when the Type 1 transaction is generated on the primary bus and is intended for a device attached directly to the secondary bus. The bridge must convert the configuration command to a Type 0 format so that the secondary bus device can respond to it. Type 1 to Type 0 translations are performed only in the downstream direction; that is, the bridge generates a Type 0 transaction only on the secondary bus, and never on the primary bus.

The bridge responds to a Type 1 configuration transaction and translates it into a Type 0 transaction on the secondary bus when the following conditions are met during the address phase:

- The lowest two address bits on P_AD[1:0] are 01b.
- The bus number in address field P_AD[23:16] is equal to the value in the secondary bus number register in configuration space.
- The bus command on P_CBE[3:0] is a configuration read or configuration write transaction. When the bridge translates the Type 1 transaction to a Type 0 transaction on the secondary interface, it performs the following translations to the address:
- Sets the lowest two address bits on S_AD[1:0].
- Decodes the device number and drives the bit pattern specified in Table 2-6 on S_AD[31:16] for the purpose of asserting the device's IDSEL signal.
- Sets S_AD[15:11] to 0.