PCI to QuickRing™ Bridge

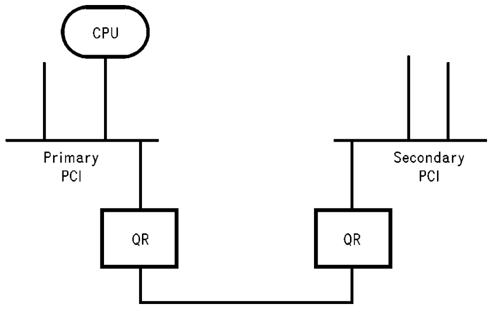
National Semiconductor
Application Note 939
Shilpa Parikh
May 1994

INTRODUCTION

QuickRing can be used to bridge several PCI buses together. The QuickRing to PCI bridging architecture application is where: one PCI primary bus has one or several secondary PCI buses. The QuickRing ring provides a fast communication path to/from the primary PCI bus to the other secondary PCI buses. In this environment, the system should use a Type 1 Configuration Command Support (over a Type 0), as defined in PCI to PCI Bridge Architecture Specification. This will provide a more flexible memory I/O address mapping. However, it should be noted that the address space of the entire system is limited to the address space of the primary PCI bus. (All the secondary buses are mapped into the primary PCI address space.) This environment is a single processor environment, yet, supporting multiple Master capability. (See *Figure 1.*) In this scenario, the primary PCI bus would support PCI register space configuration, but, the PCI secondary buses would not be implementing the PCI configuration register support.

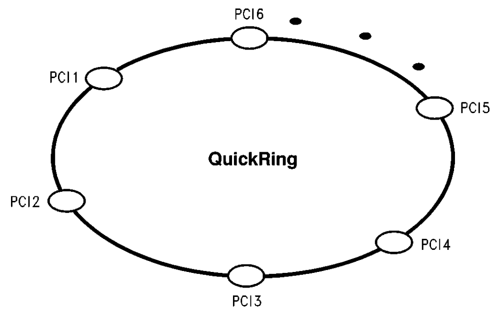
Each PCI bus is a node within one QuickRing ring. (See *Figure 2.*) Methods of translating the PCI bus protocol over QuickRing will be discussed. (See *Figure 3.*) The aim of this application note is to show a few PCI transaction translation approaches, and, raise various topics that need to be resolved during the PCI to QuickRing translation process. The logic to perform the PCI/QR translation is not simple, nor small. This application note is a good starting place when a designer is considering a QuickRing to PCI bridging possibility. The QuickRing protocol has few constraints; making it well suited for bridging applications to various other protocols.

A brief summary of the PCI signals and QR0001 Client Interface signals is first discussed. This application note assumes the reader is familiar with both QuickRing, and PCI. For reference, refer to NSC's QR0001 datasheet, the PCI Specification, rev 2.0, and PCI Local Bus: PCI to PCI Bridge Architecture Specification, Rev. 0.4 Sept. 24,1993.



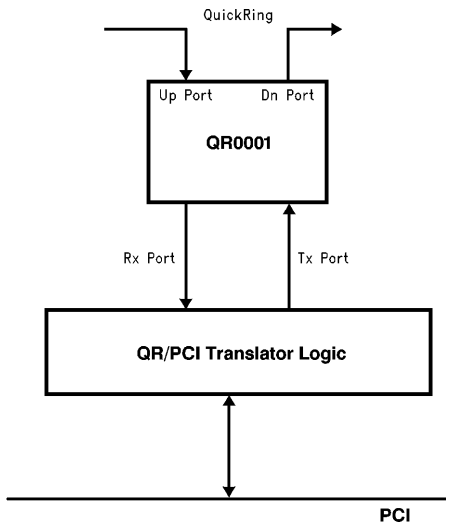
TL/F/12036–1

FIGURE 1. QuickRing Bridging PCI Buses



TL/F/12036–2

FIGURE 2. Each PCI Bus is a Node on the QuickRing



TL/F/12036–3

FIGURE 3. PCI to QR0001 Connection

PCI BUS SIGNALS

CLK	Any frequency up to 33 MHz.
RST#	Reset. This signal can be asynchronous with CLK.
AD[31:00]	Address and Data lines.
C/BE[3:0]#	Bus Command and Byte Enables are multiplexed on the same PCI pins.
PAR	Parity is even parity across AD[31:0] and C/BE[3:0]#. The number of 1s on AD[31:0], C/BE[3:0]#, and PAR equal an even number. PAR is valid one clock after the address, and one clock after data valid on the bus.

PERR#	Parity Error. Must be driven by agent receiving data two clocks following the data when a data parity is detected. Signal is valid for one clock.
FRAME#	Cycle Frame is driven by the current master to indicate the beginning and duration of an access.
IRDY#	Initiator Ready indicates the initiating agent's (bus master's) ability to complete the current data phase of the transaction.
TRDY#	Target Ready indicates the target agent's (selected device's) ability to complete the current data phase of the transaction.
STOP#	Stop indicates the current target is requesting the master to stop the current transaction.
IDSEL	Initialization Device Select is used as a chip select during configuration read/write transfers.
DEVSEL#	Device Select indicates the driving device has decoded its address as the target of the access.
REQ#	Request signal to the arbiter.
GNT#	Grant signal from the arbiter.
SERR#	System Error reports address parity errors, and other system errors.

(Refer to the PCI Specification Revision 2.0 for more details.)

PCI BUS COMMANDS

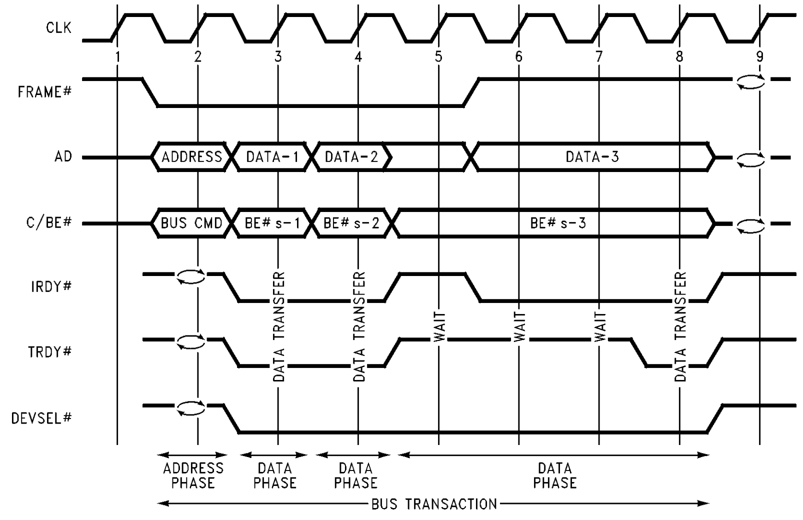
PCI divides the addressing space into three subsections:

1. Configuration
2. Memory and
3. I/O.

PCI defines the Configuration address space to support PCI hardware configuration. The Configuration Read and Write bus commands are the only required PCI bus transactions. All other commands are optional. The I/O Read and Write transfers are not different from Memory cycles, except that I/O cycles are viewed as a single data word transfer; whereas, Memory cycles can be single or multiple data word(s) transfers. Various methods of performing PCI read and write transactions across QuickRing will be shown. The read and write transactions assume blocks of data to be transferred, which takes advantage of the QuickRing bandwidth. The single data word transfer would follow the same protocol, but, transfer only one piece of data. All read and write transfers on PCI follow the same handshaking protocol. The only difference is the address of the transfer designates which region (Configuration, Memory, or I/O) is targeted for the read/write cycle. In essence, the command is providing one level of address decoding.

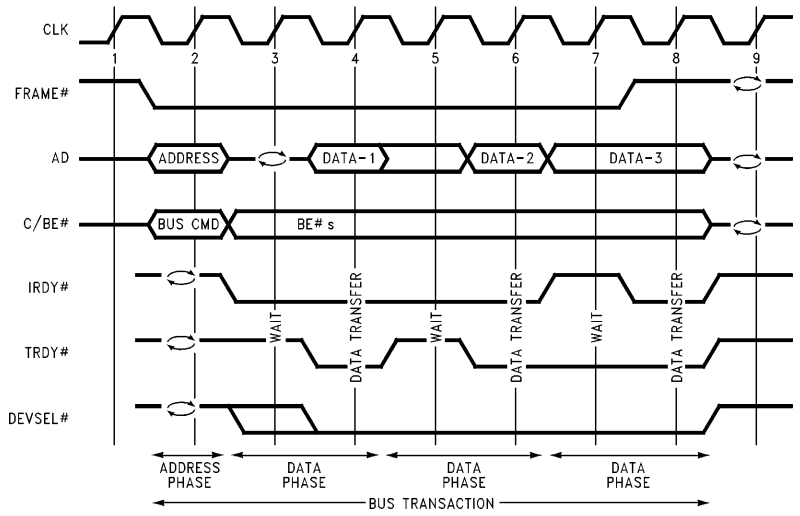
Following are PCI bus commands:

Configuration Read
Configuration Write
Interrupt Acknowledge
Memory Read
Memory Write
Special Cycle
Memory Read Multiple
Memory Read Line
Dual Address Cycle
I/O Read
I/O Write
Memory Write and Invalidate



TL/F/12036–4

FIGURE 4. PCI Basic Write Transfer (PCI Specification Rev 2.0)



TL/F/12036–5

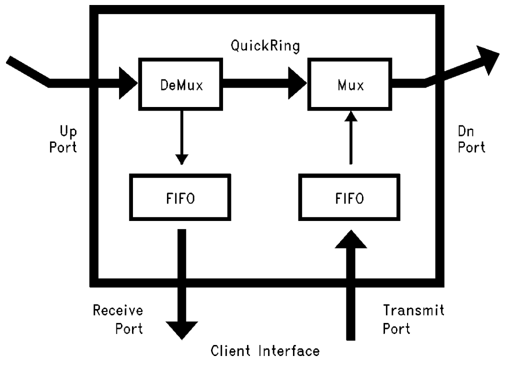
FIGURE 5. PCI Basic Read Transfer (PCI Specification Rev 2.0)

QR0001 QuickRing DATA STREAM CONTROLLER

The QuickRing Controller has two Interfaces: the Ring Interface and the Client Interface. Each Interface has two ports. All ports on the QR0001 are unidirectionai so that incoming and outgoing data can be queued simultaneously.

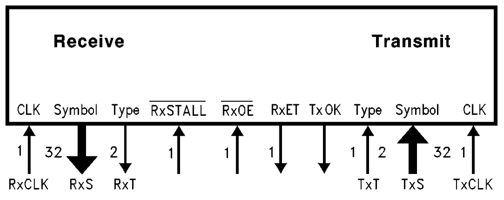
The two **Ring Interface** ports are:

1. upstream port for arriving traffic,
2. downstream port for departing traffic.



TL/F/12036–6

FIGURE 6. The QuickRing Controiler Has Four Ports



TL/F/12036–7

FIGURE 7. Client Ports of a QuickRing Controller

The Ring Interface forms the link to other nodes on the point-to-point QuickRing architecture. QuickRing connects multiple nodes by attaching the upstream port of each node to the downstream port of another node. The ring ports, upstream and downstream, are 6 bits wide plus a clock. The ring interface is implemented using LVDS drivers and receivers. The Ring Interface signals are not accessible from the board except through the controller. The on board logic connects to the QR0001 controller via the Client interface.

The two **Client Interface** ports are:

1. the transmit port for locally generated symbol streams, and
2. the receive port for locally-absorbed symbol streams.

The transmit and receive ports have a 32-bit data path which use TTL compatible I/Os. The Transmit (Tx) and Receive (Rx) ports each have a separate clock plus control signals for information flow. Also, some QR0001 internal status bits can be read through the receive interface. All on board circuitry interfaces to the Client transmit and receive ports, never to the Ring ports.

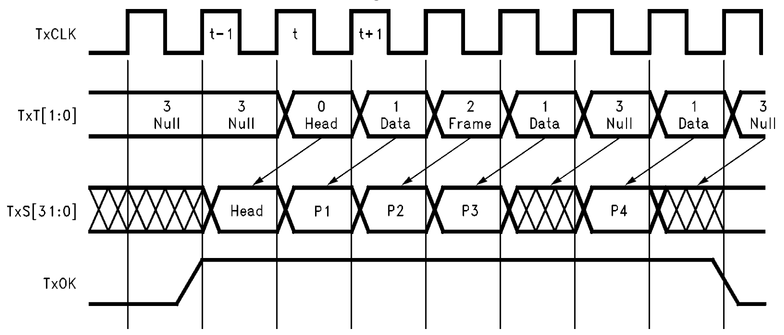
The QuickRing client can multiplex multiple independent data streams onto and from the transmit (Tx) and receive (Rx) ports of the controller. The type fields (TxT[1:0], RxT[1:0]) distinguishes the contents of the symbol (main data) fields (TxS[31:0], RxS[31:0]). The type field identifies the nature of the symbol field information at the 32-bit ports as: head, data, frame or null.

The transmit port can be thought of as the input to a bank of fast, deep FIFOs, connected to other nodes on the ring. The receive port can be treated as the output of the bank of FIFOs connected to other nodes on the ring.

THE CLIENT INTERFACE SIGNALS

TxCLK	Transmit Clock. All transmit port signals are synchronous to the rising edge of this clock. Maximum clock frequency of 50 MHz.
TxS[31:0]	Transmit Symbol. These signals form the data path of the transmit port.
TxT[1:0]	Transmit Type, this field defines the contents of TxS as head, data, frame or null.
TxOK	Transmit OKAY is the transmit port status signal. Loading of non-null symbols must cease within 20 symbols of the negation of TxOK. Transmission may not resume until TxOK is re-asserted.
RxCLK	Receive Clock. All receive port signals are synchronous to the rising edge of this clock; except $\overline{\text{RxSTALL}}$, which is sampled on the following edge of RxCLK. Maximum clock frequency of 50 MHz.
RxS[31:0]	Receive Symbol. These signals form the data path of the receive port.
RxT[1:0]	Receive Type, this field defines the contents of RxS as head, data, frame or null.
RxET[1:0]	Receive Early Type, it identifies in advance whether the information entering the Rx Port block is a head, data, frame or null.
$\overline{\text{RxOE}}$	When Receive Output Enable is asserted, it enables outputs RxS[31:0].

Transmit Port Timing when $\overline{\text{PIPE}}$ is Asserted



TL/F/12036–8

FIGURE 8. When $\overline{\text{PIPE}}$ is Asserted, the Type Field Lags the Symbol Field by One Clock Cycle at the Transmit Port

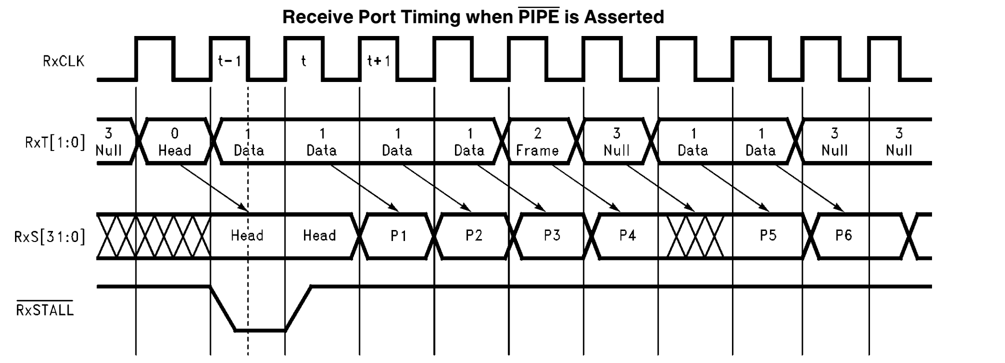


FIGURE 9. When $\overline{\text{PIPE}}$ is asserted, the Type Field Leads the Symbol Field by One Clock Cycle at the Receive Port

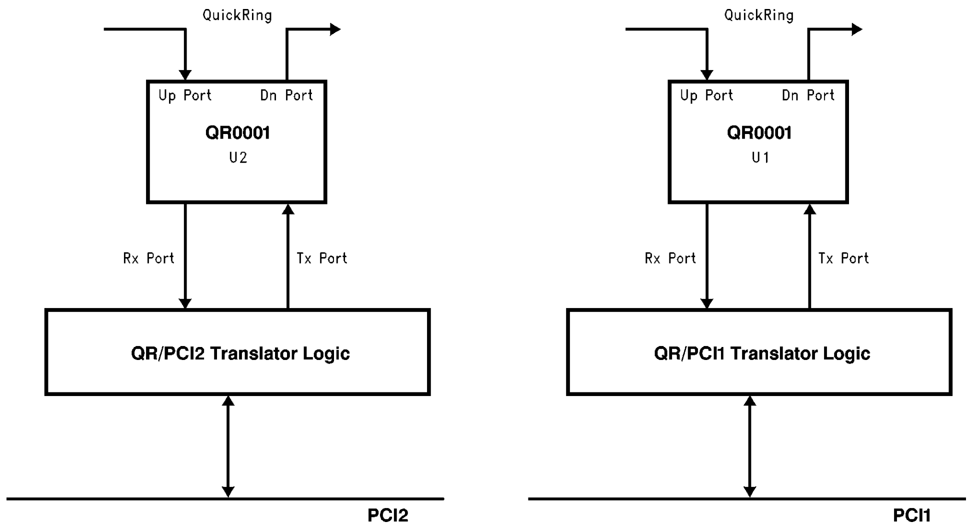
$\overline{\text{RxSTALL}}$	Receive Stall, when asserted: When $\overline{\text{PIPE}}$ is asserted, pipelined timing: RxS shall remain for the next clock cycle. When $\overline{\text{PIPE}}$ is negated, non-pipelined timing: RxT will indicate a null for the next clock cycle and RxS shall remain.
$\overline{\text{PIPE}}$	When $\overline{\text{PIPE}}$ is negated (non-pipelined timing), at the Client ports, both the symbol and type fields correspond to each other during the same clock cycle. When $\overline{\text{PIPE}}$ is asserted (pipelined timing), the timing of the Type field leads by one clock at the receive port and trails by one clock at the transmit port. (The type and symbol fields are pipelined.)
RxNBL[3:0]	Receive Nibble: it contains one of the 16 selectable fields of two readable internal areas.
RxSEL[3:0]	Receive Select: selects one of the 16 fields appearing on the RxNBL.

(For more information, refer to the QR0001 datasheet.)

PCI COMMAND TRANSLATION OVER QuickRing

In all the PCI to QuickRing translation approaches that will be discussed in the subsequent sections, the following assumptions are made:

1. The QR0001 Client Interface will use the same clock frequency as the PCI Bus (33 MHz).
2. The QR0001 Client Interface will use the pipelined timing mode, where the $\overline{\text{PIPE}}$ signal is asserted. This will give the PCI/QR translator logic a clock to encode/decode the type fields.
3. The HOP fields in the QR0001 can be used for further addressing information and/or identifying various data streams.
4. Following is a discussion of two methods for PCI command encoding:
   a. Method A:
      — Map the Configuration access as a low bandwidth connection in the Head field to the QR0001 Tx port. This is a higher priority access with a signal payload (symbol) packet on the QuickRing.
      — Map the I/O or Memory accesses as normal connections in the Head field to the QR0001 Tx port. The subsequent symbol type, TxT can be Data for I/O access or Frame for Memory access or vice versa.
   b. Method B:
      — Map the entire PCI Command into the symbol immediately following the Head. The type encoding for this symbol can be Frame, and, for the following symbols the type encoding can be Data (for the actual data). (This is a better recommended approach, compared to point a above.)



TL/F/12036–10

FIGURE 10. Information Flow from/to PCI1 to/from PCI2

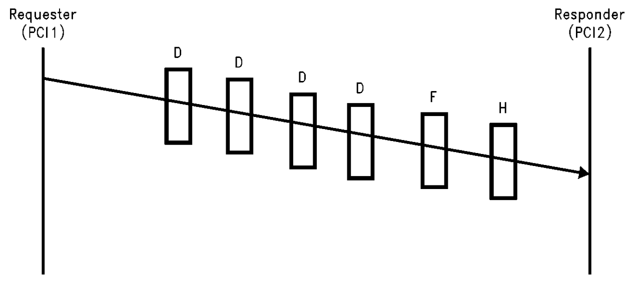
METHOD 1

- Various length streams at QR0001 Client Port
- Multiple streams
- Multiple destinations
- Complex QR/PCI Translator Logic
- Fast data thoughput
- No Link Protocol

The aim of this method is to get the maximum performance from the QuickRing architecture.

Method 1: Write Cycle (PCI1 write to PCI2)

1. PCI1 bus master places address and bus command on the PCI1 bus.
2. QR/PCI1 Translator latches the address and bus command. Decodes (either by positive decode or PCI subtractive address decoding) address and asserts device selected on PCI1, DEVSEL#.
3. QR/PCI1 Translator generates a head symbol to the QR0001Tx Port.
4. QR/PCI1 Translator generates the PCI target ready signal, TRDY# when QR/PCI1 Translator is ready to accept data. The QR/PCI1 Translator continues to pass data to the QR0001 Tx Port until either:
   a) data transfer is complete, or,
   b) QR0001 transmit OK output, TxOK is negated. Then QR/PCI1 Translator negates TRDY#. When TxOK is asserted, the QR/PCI1 Translator re-asserts TRDY#. This scenario repeats until the PCI1 bus master has transferred all the data.
5. QR/PCI2 Translator sees the head symbol and bus command from the QR0001 Rx Port. It asserts $\overline{\text{RxSTALL}}$.
6. When the QR/PCI2 Translator becomes the PCI2 bus master, it negates $\overline{\text{RxSTALL}}$ to the QR0001.
7. QR/PCI2 Translator places address and bus command on the PCI2 bus. Then asserts IRDY#.
8. QR/PCI2 Translator continues to transfer data as long as TRDY# is asserted. If TRDY# is negated, then QR/PCI2 Translator asserts $\overline{\text{RxSTALL}}$ to QR0001, and negates IRDY# to insert wait states on the PCI2 bus. When TRDY# is re-asserted, $\overline{\text{RxSTALL}}$ is negated and IRDY# is asserted. This scenario continues until all the data is transferred.
9. QR/PCI2 Translator removes its request for the PCI2 bus.



TL/F/12036–11

FIGURE 11. Method 1

READ CYCLES

The QuickRing QR0001 architecture is designed to move chunks of data from source A to destination B. Basically, QuickRing performs write transfers. There are various ways of implementing the read transfer.

One method of implementing the PCI bus read command is by having the read requester perform a write and then a read:

1. First, the PCI bus master (requester) performs a write to the QR/PCI Translator to set up the needed information (burst length, packet size, and setting the flag bit for read in progress). If the system is going to perform a set block size or burst length then this information needs to be supplied only once during system configuration. The system designer needs to ensure that only one read request is processed at a time. (Flag bit to indicate read transfer in progress.) This step may be skipped all together.
2. The PCI bus master (requester) performs a read command. The QR/PCI Translator will latch the address and bus command. Then, the QR/PCI Translator performs a PCI Target Initiated Termination, Retry. While the PCI requester will continue to retry, the Head and request is queued into the Tx port of the QR0001. When the read data returns, the QR/PCI Translator will stop retrying the cycle, and supply the data.

Method 1: Read Cycle (PCI1 Read from PCI2)

1. The PCI1 bus master places address and read bus command on the PCI1 bus.
2. QR/PCI1 Translator latches the address and bus command. Decodes (either by positive decode or PCI subtractive address decoding) address and asserts device selected on PCI1, DEVSEL#.
3. QR/PCI1 Translator generates a head symbol to the QR0001 Tx Port.
4. The QR/PCI1 Translator asserts STOP# and performs the "Target Initiated Termination" retry cycle to the PCI1 bus master. The PCI1 read requester will continue to retry until QR/PCI1 is ready to continue with the cycle. At that time, STOP# will not be asserted.
5. QR/PCI2 Translator sees the head symbol and bus command from the QR0001 Rx Port. It asserts $\overline{\text{RxSTALL}}$.
6. When the QR/PCI2 Translator becomes the PCI2 bus master, it negates $\overline{\text{RxSTALL}}$ to the QR0001.
7. QR/PCI2 Translator places address and bus command on the PCI2 bus. Then asserts IRDY#.
8. As the PCI2 bus target supplies the data, QR/PCI2 Translator generates a Head and loads the data into the Tx Port. QR/PCI2 Translator maintains IRDY# asserted as long as:
   a) data transfer is not completed (QR/PCI2 Translator needs to keep track of number of words transferred), or,
   b) if TxOK is negated, then QR/PCI2 Translator negates IRDY# until TxOK asserts. At that time, IRDY is reasserted again to continue the data transfer.
9. When QR/PCI1 Translator receives the Head and Data, the next time the PCI1 read requester attempts the cycle, the QR/PCI1 Translator will not assert STOP#; instead, it will assert TRDY#.
10. QR/PCI1 Translator transfers data as long as IRDY# is asserted by the PCI1 bus master. If IRDY# is negated, then QR/PCI1 Translator negates TRDY# and asserts $\overline{\text{RxSTALL}}$ to the QR0001. When IRDY# is asserted again, then $\overline{\text{RxSTALL}}$ is negated and TRDY# is asserted.
11. When all the data is transferred, the PCI1 bus master ends the PCI bus cycle.

METHOD 2

- Various length streams at QR0001 Client Port
- Single stream
- Simpler QR/PCI Translator Logic
- Link Protocol

The aim of this method is to simplify the QR/PCI Translator Logic. In order to get a simpler hardware design, a higher link protocol is needed. The designer can use a unique link protocol that best suits his system environment, or use a National Semiconductor defined Link protocol, or a combination of various other protocols. Contact National Semiconductor's QuickRing Applications for the Link protocol (Level 0 (Basic protocol) + Higher Level Link) information.

The QR/PCI Translator Logic gets simpler when the receiving part of the QR/PCI Translator logic needs to process only a single stream from a given source at a time.

A request/acknowledge for the transfer is embedded in the link protocol before the write or read data can be suppiled. In this method, an arbitrary link protocol is shown to give an indication of how the PCI transfer can be processed, in this type of an environment. This case is to be used only as an example. This link protocol has not been used in a system, and this is not the National Semiconductor recommended Link Protocol.

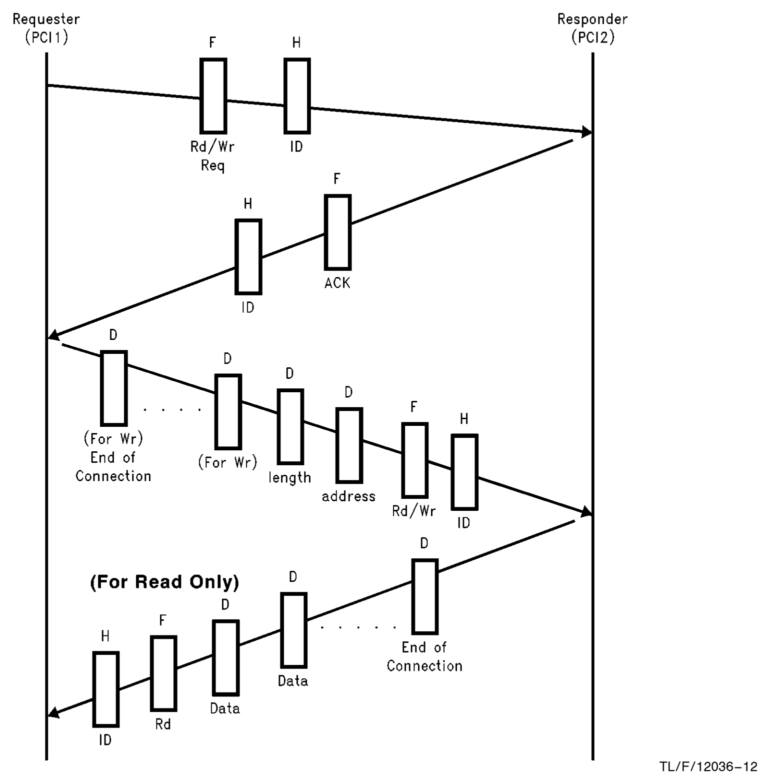


FIGURE 12. Method 2

Method 2: Write Cycle (PCI1 write to PCI2)

1. PCI1 bus master places address and bus command on the PCI1 bus.
2. QR/PCI1 Translator latches the address and bus command.
3. QR/PCI1 Translator generates a Head and request symbols to the QR0001 Tx Port.
4. QR/PCI1 Translator performs the "Target Initiated Termination" retry cycle to the PCI1 bus master.
5. When QR/PCI2 translator sees the request for a transfer:
   a) generates an ACK (positive acknowledge) if available then arbitrates for the PCI2 bus until granted.
   b) generates a NACK (negative acknowledge) if currently handling an earlier request.
6. if QR/PCI1 receives a
   a) ACK, when the PCI1 bus master is retrying the transfer, QR/PCI1 Translator continues to the data phase, this time.
   b) NACK, the QR/PCI1 Translator generates another request to the QR0001 Tx Port and continues to have the PCI1 bus master Retry until an ACK is received.
7. QR/PCI1 Translator generates the PCI target ready signal, TRDY# when QR/PCI1 Translator is ready to accept data. The QR/PCI1 Translator continues to pass data to the QR0001 Tx Port until either:
   a) data transfer is complete, or,
   b) QR0001 transmit OK output, TxOK is negated. Then QR/PCI1 Translator negates TRDY#. When TxOK is asserted, the QR/PCI1 Translator re-asserts TRDY#. This scenario repeats until the PCI1 bus master has transferred all the data.
8. When QR/PCI2 Translator receives the transfer, it initiates the write cycle then asserts IRDY#.
9. QR/PCI2 Translator continues to transfer data as long as TRDY# is asserted. If TRDY# is negated, then QR/PCI2 Translator asserts $\overline{\text{RxSTALL}}$ to QR0001, and negates IRDY# to insert wait states on the PCI2 bus. When TRDY# is re-asserted, $\overline{\text{RxSTALL}}$ is negated and IRDY# is asserted. This scenario continues until all the data is transferred.
10. QR/PCI2 Translator removes its request for the PCI2 bus.

Method 2: Read Cycle (PCI1 read from PCI2)

1. PCI1 bus master places address and bus command on the PCI1 bus.
2. QR/PCI1 Translator latches the address and bus command.
3. QR/PCI1 Translator generates a Head and request symbols to the QR0001 Tx Port.
4. QR/PCI1 Translator performs the "Target Initiated Termination" retry cycle to the PCI1 bus master.
5. When QR/PCI2 translator sees the request for a transfer:
   a) generates an ACK (positive acknowledge) if available then arbitrates for the PCI2 bus until granted.

   b) generates a NACK (negative acknowledge) if currently handling an earlier request.

6. if QR/PCI1 receives a
   a) ACK, then the QR/PCI1 Translator sends a Head, read command and length of the transfer to the Tx Port. The PCI1 bus master is still retrying the transfer.
   b) NACK, the QR/PCI1 Translator generates another request to the QR0001 Tx Port and continues to have the PCI1 bus master Retry until an ACK is received.
7. QR/PCI2 Translator places address and bus command on the PCI2 bus. Then asserts IRDY#.
8. As the PCI2 bus target supplies the data, QR/PCI2 Translator generates a Head and loads the data into the TxPort. QR/PCI2 Translator maintains IRDY# asserted as long as:
   a) data transfer is not completed (QR/PCI2 Translator needs to keep track of number of words transferred), or,
   b) if TxOK is negated, then QR/PCI2 Translator negates IRDY# until TxOK asserts. At that time, IRDY is reasserted again to continue the data transfer.
9. When QR/PCI1 Translator receives the Head and Data, the next time the PCI1 read requester attempts the cycle, the QR/PCI1 Translator will not assert STOP#; instead, it will assert TRDY#.
10. QR/PCI1 Translator transfers data as long as IRDY# is asserted by the PCI1 bus master. If IRDY# is negated, then QR/PCI1 Translator negates TRDY# and asserts $\overline{\text{RxSTALL}}$ to the QR0001. When IRDY# is asserted again, then $\overline{\text{RxSTALL}}$ is negated and TRDY# is asserted.
11. When all the data is transferred, the PCI1 bus master ends the PCI bus cycle.

METHOD 3 (Variation of Method 2 with Fixed Packet Size)

- Fixed packet size
- Single stream
- Link Protocol

Method 3 is embedded within Method 2 (along with the request/acknowledge protocol). The aim of this method is to simplify the QR/PCI Translator Logic where an I/O device only accepts a given burst length block of data. Fixing the packet length simplifies the Rx Port interface of the QR0001. This allows the Type field to be ignored during the data part of the packet unloading. This example shows how to generate a particular given block size (block size less then 20 so the entire packet makes it into the Tx Port) from the PCI bus.

To get fixed packet size on QuickRing, each burst should be assigned a new Head. A Head is assigned to the entire data transfer. Within the Head, a few bits from the unused HOP field can be used to distinguish the various bursts. For instance, two bits from the HOP field can be used as a counter. The QR/PCI logic can attach these two bits to the Head for each burst. The two bit counter bits can just be incrementing with each new burst. As long as each subsequent Head is different from the previous Head in the queue, the QR0001 will not internally lump the data together.

The block size of the data burst should be less then 20 words so all the data can get loaded into the QR0001 at one time. The negation of the TxOK signal would not further break up the block. (When TxOK negates, up to 20 more non-null symbols can be loaded into the Tx Port before having to stop transmission.) The QR0001 is geared towards sending out as many words as it can, as fast as it can. The QR0001 has the ability of internally generating new heads for data loaded from the Tx Port associated with the same Head. Thus, it is critical to load all the data within a block into the Tx Client port at regular intervals. With the current QR0001, the following table shows the maximum packet size with a given number of clock wait states for a given ring configuration. This table was put together from simulation data. The QR/PCI Translator may or may not require an external FIFO for a given configuration. Refer to following sections for more details.

TABLE I. Maximum Fixed Packet Size (Estimates from Simulations)

CLK/Data at Tx Port	Number of Nodes in QuickRing										
	2	3	4	5	6	7	8	9	10	11	12
1	20	20	20	20	20	20	20	20	20	20	20
2	16	18	20	20	20	20	20	20	20	20	20
3	8	9	10	11	14	15	16	18	19	20	20
4	6	7	8	9	10	12	13	14	15	17	18
5	4	4	5	6	7	8	8	9	11	12	13

To get a fixed burst size on the PCI bus:

1. For a PCI1 write to PCI2: the QR/PCI1 can initiate a “Disconnect” cycle at each burst boundary to the PCI1 bus master writing the data. This will cause the PCI1 bus master to generate a new address and try to send the remaining data. Again, the QR/PCI1 Translator will disconnect at the burst boundary, and the PCI1 bus master will generate a new address for the remaining words. This scenario is repeated until all the data is broken into the desired block size.
2. For a PCI1 read from PCI2: The QR/PCI2 Translator is the PCI2 bus master. The QR/PCI2 Translator needs to end the PCI2 read cycle at each burst boundary, and begin a new cycle to get the remaining data. Thus, the QR/PCI Translator will need to have a counter for the block size, and be able to generate a new address to begin another PCI2 bus cycle to get subsequent data.

PCI Parity and Byte Enable Information

The PCI bus generates and checks parity for AD[31:00], and C/BE[3:0]#. The QR/PCI Translator needs to check parity for information queued for the Tx Port, and needs to generate a parity bit for information from the Rx Port. The parity bit is a single bit, PAR. Even parity is implemented on PCI. For error handling, PERR# and/or SERR# need to be asserted on the PCI bus. The parity information is not actually passed over QuickRing. However, it must be checked and supplied for the PCI bus.

1. From the PCI bus, if data has correct parity, then the QR/PCI Translator passes the data to QR0001 Tx Port. If data has incorrect parity, then error is signaled on the PCI bus and data is not forwarded over QuickRing.
2. For data from the Rx port, if the QR0001 EDC code indicates no error, then a parity bit is generated, and data and parity is passed to the PCI bus. If the QR0001 EDC code indicates an error, then appropriate error handling mechanism needs to be put into the design.

Byte Enable information needs to be passed for each data word. This byte enable information may be included in with the QR0001 symbol containing the command information. Whenever, the byte enable information changes, a Frame symbol can be inserted before the data to indicate valid lanes.

QR/PCI Translator Logic

There are various approaches a designer needs to consider for the QR/PCI Translator Logic to the QR0001 Client interface. Here is a list of some options:

1. μP or μC interface
2. Hardware DMA Controller (single or multiple channel)
3. Local FIFO
4. Direct I/O Bus

Depending on the QR/PCI protocol translation method selected, one of the above methods may make more sense then others. In each interface choice, there is a price/performance tradeoff. The various Client Interface approaches are discussed in another application note. Please review "QuickRing Client Interfaces" for more details.

A few example methods have been presented in this application note. For a specific PCI to QuickRing design, the system designer will be aware of relevant issues to be able to determine which methodology to use in his design.

National Semiconductor Corporation
2900 Semiconductor Drive
P.O. Box 58090
Santa Clara, CA 95052-8090
Tel: 1(800) 272-9959
TWX: (910) 339-9240

National Semiconductor GmbH
Livry-Gargan-Str. 10
D-82256 Fürstenfeldbruck
Germany
Tel: (81-41) 35-0
Telex: 527649
Fax: (81-41) 35-1

National Semiconductor Japan Ltd.
Sumitomo Chemical
Engineering Center
Bldg. 7F
1-7-1, Nakase, Mihama-Ku
Chiba-City,
Ciba Prefecture 261
Tel: (043) 299-2300
Fax: (043) 299-2500

National Semiconductor Hong Kong Ltd.
13th Floor, Straight Block,
Ocean Centre, 5 Canton Rd.
Tsimshatsui, Kowloon
Hong Kong
Tel: (852) 2737-1600
Fax: (852) 2736-9960

National Semiconductores Do Brazil Ltda.
Rue Deputado Lacorda Franco
120-3A
Sao Paulo-SP
Brazil 05418-000
Tel: (55-11) 212-5066
Telex: 391-1131931 NSBR BR
Fax: (55-11) 212-1181

National Semiconductor (Australia) Pty, Ltd.
Building 16
Business Park Drive
Monash Business Park
Nottinghill, Melbourne
Victoria 3168 Australia
Tel: (3) 558-9999
Fax: (3) 558-9998