ADVANCED INFORMATION

CY7C965

Raceway Crossbar

Features

- **160 Mbyte per second per path Block Transfer Rates**
- **6 bidirectional ports**
- **Non-blocking architecture**
- **361-pin CBGA package**
- **Implements Open Bus Standard (VITA 5−1994)**
- **Building Block for Scaleable Networks**
- **Preemptable prioritized transactions**
- **Adaptive Routing support**
- **JTAG Port**

Functional Description

The CY7C965 Raceway Crossbar implements in one device the Raceway open standard for crosspoint interconnect (VITA 5−1994). The Raceway standard allows multiple processor systems to communicate using a crossbar technology that supports very high composite data transfer rates.

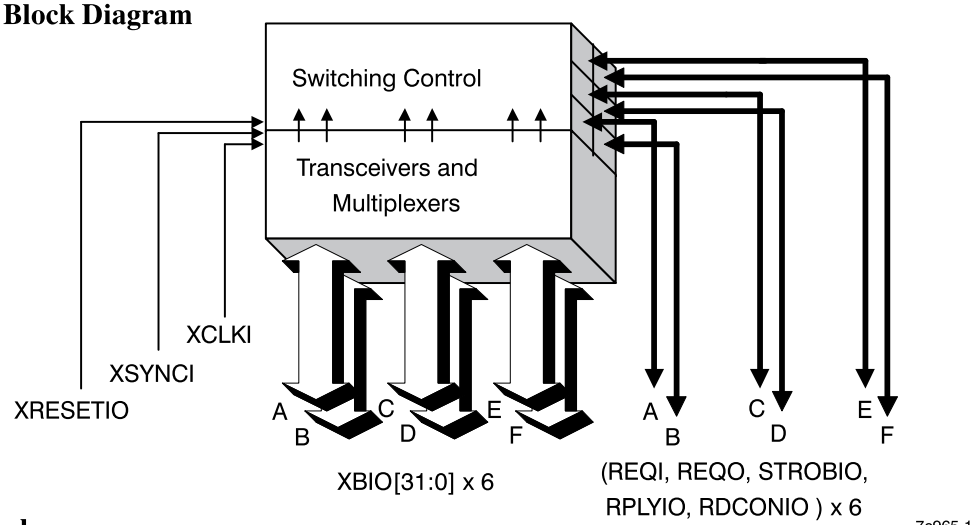
Applications for the Raceway Crossbar include high-performance multiprocessing systems and distributed processing systems. The Raceway Crossbar can be used in backplane-based applications or as switch elements on single boards. The protocol implemented by the Raceway Crossbar is useful in either a message-passing or shared-memory programming model.

The Raceway Crossbar can be connected in many different system configurations. In its simplest configuration, the Crossbar is used to interconnect six standalone boards using a single crossbar. Higher complexity systems may require the implementation of a large fabric of interconnected Crossbars. Fault tolerant systems may require the implementation of multiple redundant paths between nodes in the fabric. Time-critical elements in the fabric may require interrupt support, low latency connections, and adaptive routing support. The Raceway Crossbar has been designed to support all these diverse requirements.

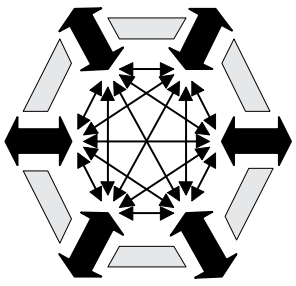
Any system with high data throughput requirements could take advantage of Raceway Crossbar technology to ease the burdens associated with data throughput using conventional methods. Bus and backplane-based systems are not suited to shared-resource, high-throughput applications because of the two major limitations inherent in their design—only one data conversation at a time can occur, and the rate of the data transfer is compromised by the difficulty of providing a large number of bus connections while controlling the impedance of a multi-drop bus or backplane.

The CY7C965 provides an elegant solution to these problems by abandoning the concept of a shared bus in favor of a point-to-point interconnect approach. A single Raceway Crossbar has six independent data ports and is therefore capable of three simultaneous conversations. As the device is cascadeable, several Crossbars can be connected together to support many simultaneous conversations, leading to an aggregate bandwidth which is a product of the number of conversations times the rate of each transfer. Point-to-point interconnect means that each port on each Crossbar is connected to only a single load. Therefore the impedance of the connection is controlled at the point of design and does not change thereafter. This allows for a robust high-performance transfer that is not encumbered by the need to cater for different numbers of connections, variable loads, and other problems associated with multi-drop buses.

Logic Block Diagram

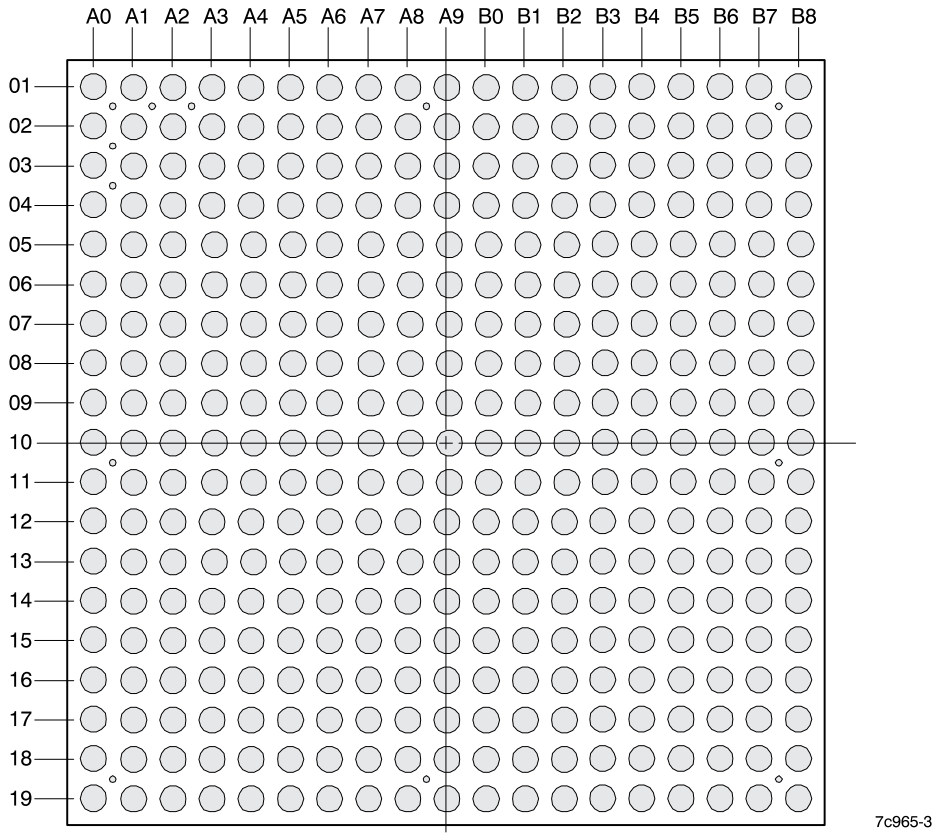


Symbol



Pin Configuration

361-Pin Ceramic Ball Grid Array



Pin Assignments

	A0	A1	A2	A3	A4
01	DXBIO7	FXBIO7	BXBIO8	DXBIO8	CXBIO9
02	BXBIO7	AXBIO4	CXBIO10	BXBIO9	EXBIO8
03	DXBIO6	EXBIO4	AXBIO8	FXBIO6	GND
04	BXBIO5	EXBIO5	AXBIO6	AXBIO7	N/C
05	CXBIO4	CXBIO5	GND	CXBIO7	GND
06	FXBIO3	FXBIO4	BXBIO6	VCC	EXBIO6
07	AXBIO3	BXBIO4	GND	FXBIO5	GND
08	TRESET	CXBIO3	DXBIO4	VCC	DXBIO5
09	TDO	TDI	GND	BXBIO3	GND
10	GND*	TCLK	GND*	GND	XCLKI
11	GND*	GND*	GND	GND*	GND
12	GND*	GND*	GND*	VCC	GND*
13	GND*	ESTROBIO	GND	ASTROBIO	GND
14	N/C*	BSTROBIO	FSTROBIO	VCC	GND*
15	DSTROBIO	GND*	GND	FRDCONIO	GND
16	ERDCONIO	ARDCONIO	CSTROBIO	FRPLYIO	BRDCONIO
17	CRDCONIO	N/C*	DRPLYIO	BRPLYIO	GND
18	ERPLYIO	CREQO	BREQI	AREQI	EXBIO2
19	CRPLYIO	ARPLYIO	FREQO	EREQO	DREQI

VCC*, GND* – a connection to the designated supply is required for CMOS inputs to the crossbar.
N/C* – this pin must not be connected to a low impedance supply as it is a CMOS output.

Pin Assignments (continued)

	A5	A6	A7	A8	A9
01	DXBIO9	BXBIO10	FXBIO10	BXBIO11	CXBIO11
02	AXBIO9	FXBIO9	N/C*	EXBIO11	N/C*
03	EXBIO7	GND	EXBIO10	DXBIO11	BXBIO12
04	VCC	FXBIO8	VCC	FXBIO11	GND
05	CXBIO6	GND	DXBIO10	GND	N/C*
06	VCC	AXBIO10	VCC	GND*	GND
07	CXBIO8	GND	EXBIO9	AXBIO11	GND*
08	VCC	EXBIO3	VCC	AXBIO5	VCC
09	DXBIO3	GND	N/C*	VCC	N/C
10	VCC	XSYNCI/TMS	VCC	VCC	N/C
11	N/C*	GND	N/C	VCC	N/C
12	VCC	XRESETIO	VCC	VCC*	VCC
13	EREQI	GND	BXBIO2	ENFWE	ENKILL
14	VCC	DRDCONIO	VCC	EXBIO1	VCC
15	VCC*	GND	FXBIO2	GND	DXBIO0
16	VCC	DREQO	VCC	CXBIO1	GND
17	FREQI	GND	DXBIO2	DXBIO1	GND*
18	FXBIO1	CREQI	CXBIO2	BXBIO1	N/C*
19	BREQO	AREQO	AXBIO2	AXBIO1	FAIR

	B0	B1	B2	B3	B4
01	GND*	DXBIO13	FXBIO13	FXBIO14	CXBIO15
02	CXBIO12	DXBIO14	AXBIO14	EXBIO15	CXBIO14
03	EXBIO12	BXBIO14	GND	BXBIO15	GND
04	FXBIO12	VCC	DXBIO15	VCC	EXBIO16
05	GND	BXBIO13	GND	CXBIO16	GND
06	DXBIO12	VCC	AXBIO15	VCC	CXBIO18
07	CXBIO13	FXBIO17	GND	DXBIO19	GND
08	AXBIO12	VCC	BXBIO19	VCC	AXBIO20
09	GND	EXBIO13	GND	DXBIO20	GND
10	GND	VCC	BXBIO22	VCC	AXBIO21
11	GND	BXBIO23	GND	BXBIO24	GND
12	FXBIO0	VCC	DXBIO24	VCC	FXBIO22
13	BXBIO27	DXBIO25	GND	FXBIO25	GND
14	CXBIO31	VCC	FXBIO28	VCC	CXBIO23
15	GND	FXBIO31	GND	CXBIO29	GND
16	BXBIO0	VCC	EXBIO29	VCC	FXBIO26
17	CXBIO0	AXBIO31	GND	CXBIO28	GND
18	AXBIO0	BXBIO31	FXBIO30	FXBIO29	BXBIO30
19	EXBIO0	EXBIO31	DXBIO31	EXBIO30	CXBIO30

Pin Assignments (continued)

	B5	B6	B7	B8
01	FXBIO15	BXBIO16	DXBIO16	FXBIO16
02	AXBIO13	EXBIO18	N/C	AXBIO17
03	EXBIO14	FXBIO19	BXBIO20	EXBIO17
04	AXBIO16	DXBIO17	FXBIO21	AXBIO18
05	BXBIO17	GND	CXBIO17	AXBIO19
06	VCC	BXBIO18	CXBIO19	EXBIO19
07	DXBIO18	GND	FXBIO18	BXBIO21
08	VCC	CXBIO20	EXBIO20	FXBIO20
09	CXBIO21	GND	DXBIO21	EXBIO21
10	DXBIO22	CXBIO22	EXBIO22	AXBIO22
11	EXBIO23	GND	CXBIO24	AXBIO23
12	VCC	BXBIO25	FXBIO24	EXBIO24
13	CXBIO26	GND	BXBIO26	DXBIO23
14	VCC	DXBIO26	FXBIO23	CXBIO25
15	FXBIO27	GND	AXBIO24	EXBIO25
16	DXBIO27	AXBIO26	AXBIO25	EXBIO26
17	DXBIO28	BXBIO28	DXBIO30	AXBIO27
18	BXBIO29	EXBIO28	AXBIO28	CXBIO27
19	AXBIO30	DXBIO29	AXBIO29	EXBIO27

Pin Description

Raceway Port Signals (p = A or B or C or D or E or F)

*p*XBIO[31:0] Data Port Bidirectional 4 mA

Six 32-bit signal sets labeled AXBIO through FXBIO are the bi-directional route/address/data pins of the six crossbar ports. They carry route header, address and data in accordance with the time multiplexed Raceway protocol. They are connected to only one other port in the fabric.

*p*REQI Request In Input

Each Raceway port, A through F, uses its Request In signal to detect two types of request. If the port is currently a master of a transaction, assertion of this signal is interpreted as a transaction kill request, causing that port to cleanly terminate its transaction and subsequently deassert its Request Out signal, freeing the port for higher priority traffic. If the port is idle, pREQI is interpreted as a normal connection request. This signal is driven by the *p*REQO of the connected port.

*p*REQO Request Out Output 4 mA

Each Raceway port, A through F, drives its Request Out signal when it is a master to start a Raceway transaction from that port. The Request Out signal remains asserted until the port's master tenure is over. If the port is a slave to a transaction, then the *p*REQO is driven to indicate that a kill is being requested. This signal is intended for the *p*REQI of the connected port.

*p*RDCONIO Read Condition Bidirectional 4 mA

Each Raceway port, A through F, monitors its Read Condition signal while master of a Raceway transaction. Depending on assertion timing, this signal causes the port to three-state its XBIO bus, or indicates that a read error occurred in the transaction just completed. If the port is slave to a Raceway transaction, Read Condition is asserted to signal the transaction master to three-state XBIO, or asserted at the end of a read of the slave port to signal a read error. *p*RDCONIO is driven by the slave port of a connected pair.

*p*STROBIO Data Strobe Bidirectional 4 mA

Each Raceway port, A through F, drives its Data Strobe signal during master transaction tenure to initiate data transfer on the XBIO bus. A port engaged as slave to a Raceway transaction receives Data Strobe as an input. If the transaction is a read, data will be sourced by the port, if the transaction is a write, data will be captured by the port. *p*STROBIO is driven by the master of a connected pair.

*p*RPLYIO Reply Bidirectional 4 mA

Each Raceway port, A through F, receives Reply during master transaction tenure. Reply has several meanings depending on the transaction type and timing of Reply assertion. Its first assertion causes the port to switch from driving route information to driving address onto XBIO. This is the "change to address" function. The master port interprets subsequent assertions of Reply based on transaction data direction and phasing. "split," "ready read," and "data strobe enable" are the possible functions indicated by the Reply signal. If the port is slave to a Raceway transaction, Reply is driven from the port with "change to address," "data strobe enable," "ready read," and "split" as possible output encodings. *p*RPLYIO is driven by the slave of a connected pair.

Crossbar JTAG Port Signals

TDI Test Data In Input

Test Data In is the serial data input to the I/O boundary scan function built into the crossbar device. The crossbar receives test codes and bit patterns through this input when the TMS and TCLK inputs are appropriately manipulated.

TDO Test Data Out Output 4 mA

Test Data Out is the serial data output of the I/O boundary scan function built into the crossbar device. This output is provided to

allow chaining of the crossbar JTAG facility to other JTAG equipped components.

XSYNCI/TMS Test Mode Select Input

Test Mode Select enables the JTAG boundary scan facility, reconfiguring crossbar I/O functionality in conformance with ANSI/IEE 1149.1−1990. This pin is also the crossbar control signal XSYNCI.

TCLK Test Clock Input

Test Clock is the JTAG serial port clock. TDI set-up and hold requirements are referenced to the rising edge of this signal. Each rising edge advances the boundary scan serial bit stream one stage.

TRESET Test Reset Input

Test Reset is the JTAG serial port reset. The JTAG port can be asynchronously reset at any time to clear and initialize the test port.

Crossbar Control Signals

XCLKI Crossbar Clock In Input

Clock In Reference is the crossbar clock. The crossbar is designed to accept a 40-MHz 50% duty cycle CMOS logic level input. All Raceway port inputs are sampled and outputs driven from the rising edge of this clock signal.

XRESETIO Crossbar Reset Input

This signal resets and initializes the crossbar.

XSYNCI/TMS Crossbar Sync Input

Internal to the crossbar a divide by two circuit generates phase markers (phase 0 and phase 1) for driving and sampling Raceway signals. Raceway ports rely on phase definition between master and slave to communicate. The Crossbar Synchronization signal defines the phase of a crossbar in relation to the ports to which it is connected. Phase 0 is defined for a crossbar as the clock period in which XSYNCI is sampled LOW on the rising edge of XCLKI. XSYNCI can be one-shot or repetitive. This pin is also the JTAG Test Mode Select signal TMS.

Crossbar Arbitration Signals

FAIR Fair Input

The FAIR signal is normally set true on all crossbars of a "fat tree" topology. This signal affects the prioritization scheme used for requests into "child" ports A, B, C, & D. FAIR, if true, allows an ongoing transfer on a "child" port to continue (no kills) unless a higher priority software request comes along on another "child" port. "Parent" ports E and F are not affected by this signal and hence requests in on these ports will always kill transfers of the same software priority from "child" ports A through D. FAIR, if set LOW (disabled), always lets the request with the highest alphabetical priority kill any connection of the same software priority but lower alphabetical port priority.

ENKILL Enable Peer Kills Input

The Enable Peer Kills signal modifies the arbitration on (requests into) ports A through D when the FAIR signal is true. Enable Peer Kills allows peer kills to take place, but only when no data has yet been sent through the connection (no Data Strobe yet). The purpose of Enable Peer Kills is to allow killing of stalled requests. This is the case where a request makes it through some but not all the crossbars. In this case Enable Peer Kills allows stalled connections to be killed to let other requests get through.

ENFWE Enable F Wins Over E Input

The Enable Request F Wins Over E signal introduces asymmetry in tree topologies to prevent a thrashing condition where two nodes are simultaneously requesting each other over a pyramid. This setting is normally true for all crossbars that are not at the very top of a fat tree topology. When set true the alphabetical prioritization of ports is modified so that arbitration occurs as follows: Any request into port F always kills any blocking connection; next, any request in on ports A through D going to port F kill any blocking connection; next, requests in on ports E − A are prioritized in decreasing order (E wins over D, etc.)

Crossbar System Overview

The CY7C965 implements the fabric of the interconnect topology. The source of the data transfer must provide the data in a format that is acceptable to the fabric before the data can be transmitted. This function can be provided by a simple PLD-controlled interface. This interface device, sometimes called an "on-ramp," is likely to be located on a board that plugs into the Raceway fabric.

The functions performed by a Raceway "on-ramp" include transmission of the routing information header, and providing the control signals needed by the first stage of the Crossbar fabric. Additional functions that could be provided by the "on-ramp" device include local DMA control, local bus arbitration, and Raceway management functions like pre-emption, broadcast, and adaptive routing control. The interface provides the controls needed to commence the establishment of the route through the fabric. Once the reply has been received from the destination the interface provides the timing signals for the block of data to be transferred.

Figure 1 shows the use of a single crossbar to interconnect four separate nodes. Two spare ports, E and F, are shown. This topology provides up to 320 Mbytes/second data transfer supporting two simultaneous conversations.

Figure 2 shows the use of three crossbars to provide interconnection of eight nodes. Note that this topology supports adaptive routing, hence the use of the third crossbar. If adaptive routing is not required, then two crossbars could be used. This topology gives an aggregate bandwidth of 640 Mbytes/second supporting four simultaneous conversations. Again, two spare ports, E and F are available for expansion of the fabric.

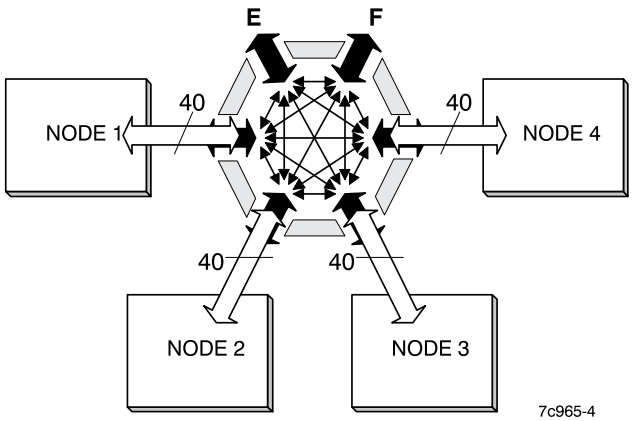


Figure 1. Four Node Fabric

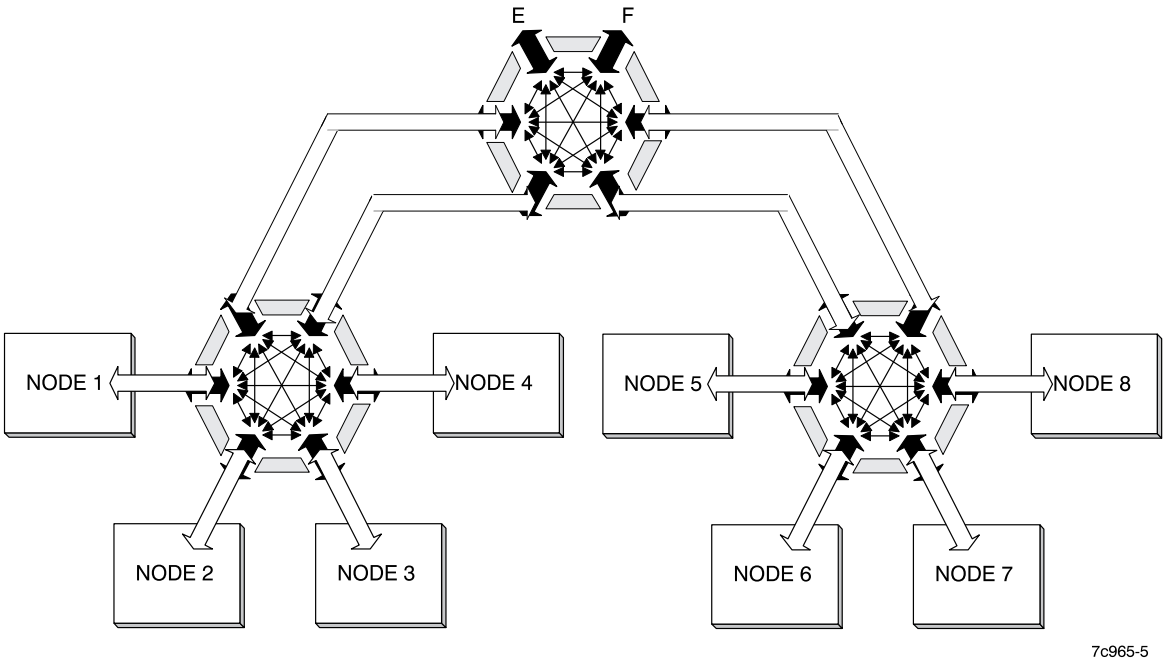


Figure 2. Eight Node Fabric

Crossbar Feature Overview

Six Ports

The CY7C965 Raceway Crossbar provides six independent 32-bit bidirectional data ports. Each port has signals associated with it that control arbitration, direction, pre-emption, handshaking, and all other aspects of the protocol operation.

Bidirectional

Each port is bidirectional. The direction of a block transfer is determined during the arbitration and connection phase and remains fixed for the duration of that block transfer. Following the block transfer the port is available for data transfers in either direction. The exception to this is the LOCK transaction, used for read-modify-write operations: the data direction reverses between the read and the write.

Preemptable

If a data port is engaged in a conversation and a higher priority data transfer is requested which requires the use of the busy port, then a "kill" request is sent by the Crossbar to the master controlling that conversation. Once the master has responded to the kill request by releasing the port associated with the low-priority transfer, the higher priority master acquires the port. The lower priority master can immediately attempt to re-establish the original conversation, and the request will go unanswered until the higher priority master has relinquished the connection. At that time the original connection can be resumed.

Adaptive Routing

The Raceway Crossbar supports adaptive routing, which allows the data connection to be routed automatically around busy ports. There are limitations on this behavior which are dependent upon the interconnection topology chosen by the designer. Basically, the network of Crossbars has to be "balanced," allowing at least two paths between any two nodes and ensuring that the distance between any two node in the fabric includes the same number of Crossbars. Knowing these limitations, the designer can construct a topology which supports adaptive routing.

Scaleable

As additional crossbars are added to a network fabric, the aggregate throughput of data increases proportionally. To a first approximation, doubling the number of crossbars in a fabric doubles the bandwidth of the fabric.

Data Rate

The clock that governs transaction timing is defined to be 40 MHz. The maximum rate at which the data is strobed through the fabric is one 32-bit word per rising edge of the clock signal, implying a maximum data rate of 160 MByte/second (four bytes wide, 40-MHz strobe signal). The master and the slave can each throttle the data transfers by withholding the strobe signals, thus though synchronous, the protocol does not demand maximum performance from all participants.

The Raceway protocol is an efficient mechanism for moving blocks of data. Only eight bytes of header information are required to transfer two Kbytes of data. This efficiency allows sustained transfer rates to be 95% of the burst maximum.

Compelled Transfers

The data transfers are compelled, which means that when a master starts to write a block of data, the slave is compelled to take the entire block. The slave can throttle the rate that it accepts the data, but it must accept every word. (No facility is defined in the VITA 5 −1994 specification for the slave to terminate the block transfer.)

Control Signals

Each Crossbar port is comprised of 32 data lines, XBIO[31:0], controlled by 5 dedicated lines per port. The five control signals are REQO, REQI, RPLYIO, STROBIO, and RDCONIO. Three signals are common to all ports. The three common lines are XRESETIO, XSYNCI, and XCLKI.

Port Contention

The XSYNCI signal provides a reference phase signal that is used in two ways: it provides a multiplexing reference timing signal allowing the control signals to have two functions each; and it provides a mechanism for resolving potential deadlock situations. Each port on a crossbar is connected to only one other port, either a port of another crossbar, or a Raceway interface node. By ensuring that a crossbar is "out of phase" with all the ports connected to it, the protocol can never generate a situation where two ports (crossbar-crossbar or crossbar-interface) contend for the ownership of their connecting path. This imposes the limitation that certain fabric topologies (for example a "triangular" connection) are not allowed.

Split Read

A split read transaction is one where the master wishes to read data from a slave which, either because the data is located in a slow electromechanical device like a disk, or because the peripheral has a local bus contention issue, is not able to provide the data immediately. If the slave is designed to be capable of split reads, the slave will assert a control signal which propagates back to the master. In this case the master writes a return route to the slave. Following the completion of this short write transaction the Raceway connection is released while the slave prepares itself to send the data. When ready the slave will become a master to establish the Raceway connection and write a 32-byte data block back to the location specified earlier. This provides better utilization of Raceway bandwidth.

Locked Operations

Locked operations, sometimes called atomic operations, are supported through the use of a lock bit in the route header. Thus semaphore operations, test and set, and other locked cycles are supported. The method used to implement locking is simple: the master of a multi-crossbar data transaction refuses kill requests until the locked operation is complete, and the slave is prepared for the atomic read/modify by the lock bit in the header.

Broadcast/Multicast

The Raceway Crossbar provides a broadcast/multicast facility whereby a master can send a block of data to a group of slave destinations. This is supported by the use of a broadcast bit in the routing word. When the bit is enabled, participating crossbars send the route information and the subsequent data block out of all data ports. The protocol supports transmission of data to a subset of all nodes in the fabric. A field in the route information may be used to select a group of nodes with matching broadcast accept fields. This is accomplished by the receiving nodes, not by the participating crossbars. The crossbars merely provide the field unmodified to the slaves.

Crossbar Use

Data Width and Alignment

Although the physical port width is 32 bits, the logical data structures that the crossbar handles are 64 bits wide. The crossbar protocol demands that the interface to the Raceway provide two 32-bit data words as the unit of exchange. Thus as shown in *Figure 3* the data block transferred through the fabric is composed of consecutive 64-bit quantities. For this reason the address field defined by the protocol specifies a "double longword" address. Another way to look at this is to say that the data blocks are aligned on 8 byte address boundaries. Unaligned transfers are not supported during block transfers.

For non-block (single cycle) transactions, byte addressing within the eight-byte double long word is communicated using the Width/Alignment field of the header address word (see *Figure 4*). This field contains encoded byte enable information to allow most alignments of one-byte, two-byte, or four-byte transactions to be specified (refer to *Table 2*).

Routing Principles

The purpose of the transaction header is twofold—it provides the desired path through the crossbar fabric, and it provides the des-

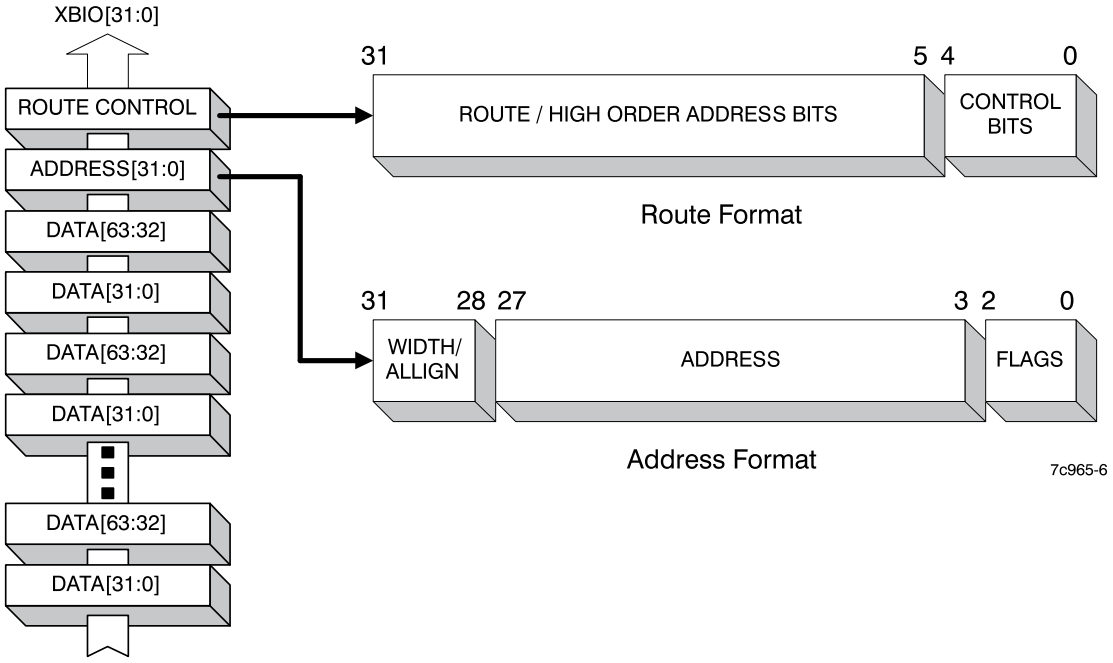


Figure 3. Format of Data Transfer

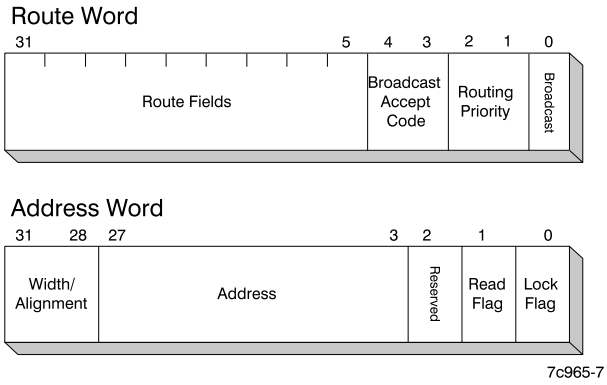


Figure 4. Header Fields

tination with the starting address of the data to be transferred. It also contains control signals for actions such as broadcasts, kill operations, and whether the transaction is a read or a write.

The fields of the route header are shown in *Figure 4*. The initiating master specifies a path through the fabric that consists of the desired output ports at each crossbar in the path. Each port field is three bits wide. This illustrates a protocol-imposed limit on the size of the fabric—the total space for the fields is 27 bits, implying a maximum of nine route fields. Each path can traverse up to nine crossbars. Note that this is actually a substantial fabric as each crossbar can connect to six other crossbars.

After the initiating master has transmitted the route field, the most-significant three bits are used by the first crossbar to pick an output port to complete the connection across that crossbar. The first crossbar shifts the upper 27 bits of the route header left by three bits and passes the result through to the next crossbar, which looks at the most significant port field to determine which output port is desired. This continues until the responding slave has been contacted. *Figure 5* illustrates the progression of the route header through the fabric.

If the fabric does not require all nine fields, the remnant fields may be used at the responding slave for any purpose, such as additional high-order address bits. Note, however, that this remnant field resides in the most significant bit positions, and the number of chips traversed determines how wide the field is. It is the initiating master's responsibility to place the high-order address bits in the appropriate fields, knowing how many shift operations will be performed on the field by the intermediate crossbars.

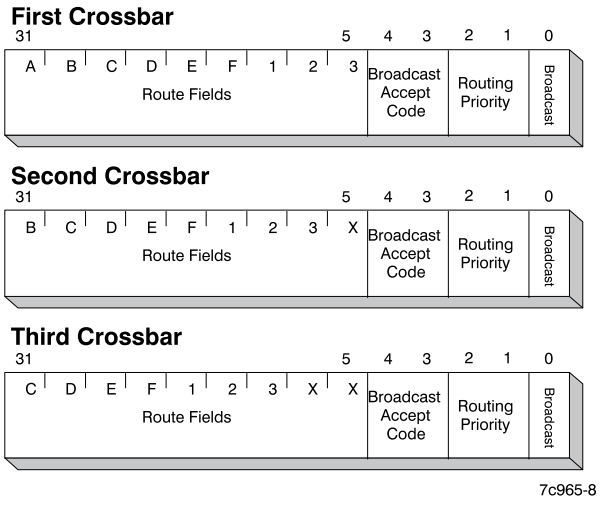


Figure 5. Route Progression

The lower 5 bits of the Route word are not shifted as the fabric is traversed. Instead the bits are examined by each crossbar along the route and passed unchanged.

The Broadcast bit indicates to the crossbar whether the transaction is to be a single mode (0) or broadcast write (1). The Routing Priority field (Bits 2−1) indicates to the crossbar which of three priorities the incoming message is. This determines whether the crossbar will issue a kill request to an ongoing transaction that is using the desired port.

Table 1. Routing Priority Field Assignment

Priority Level	Routing Priority[1:0]
Low	00
Middle	01
High	10
Reserved	11

The Broadcast accept code provides responding slaves with a means to identify with one of four node populations. This allows multicast operations. When the Broadcast bit is 0 these two bits are reserved and must be set to 0.

The address word of the route header is composed of three flag bits, 25 address bits, and 4 byte alignment bits. When reset to 0, bit 0 indicates to the responding slave that the current transaction is to be a locked operation such as a read-modify-write. Bit 1 indicates a Read transaction while set to 1. Bit 2 is reserved and must be set to 0. The address field is passed to the responding slave unchanged. The width/alignment field is defined in *Table 2*, and is used by the responding slave to determine which of the 8 byte locations are to be considered active during the following single 64-bit transaction.

Table 2. Byte Selection

Byte Enables								Width/Align			
63:56							7:0	Bit			
B7	B6	B5	B4	B3	B2	B1	B0	31	30	29	28
1								0	0	0	0
	1							0	0	0	1
		1						0	0	1	0
			1					0	0	1	1
				1				0	1	0	0
					1			0	1	0	1
						1		0	1	1	0
							1	0	1	1	1
1	1							1	0	0	0
		1	1					1	0	1	0
				1	1			1	1	0	0
						1	1	1	1	1	0
1	1	1	1					1	0	0	1
				1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	0	1	1
			Reserved					1	1	1	1

Routing Codes

Table 3 shows the method used by the CY7C965 to pick an output port based upon the contents of the most-significant route field, the broadcast bit, and the entry port used by the message.

Table 3. Routing Code Interpretation

Code	Entry Port	Single Mode Exit Port	Broadcast Mode Exit Ports
7	B,C,D,E,F	A	A
7	A	undefined	B,C,D
6	A,C,D,E,F	B	B
6	B	undefined	A,C,D
5	A,B,D,E,F	C	C
5	C	undefined	A,B,D
4	A,B,C,E,F	D	D
4	D	undefined	A,B,C
3	A,B,C,D,F	E	E
3	E	undefined	undefined
2	A,B,C,D,E	F	F
2	F	undefined	undefined
1	A,B,C,D,E,F	Adaptive E first	A,B,C,D,E (except port entered)
0	A,B,C,D,E,F	Adaptive F first	A,B,C,D,F (except port entered)

This table illustrates that two of the ports, E and F, are considered as parent ports, and the other four are considered child ports. This partitioning makes the broadcast operation and adaptive routing possible. Here are some examples of routing codes which may help clarify the situation:

A broadcast message entering Port A with code 7 outputs on Ports B, C, and D. But a broadcast message entering port F with code 7 exits Port A.

A broadcast message entering Port F with code 0 exits Ports A, B, C, and D.

A broadcast message entering Port F with code 1 exits Ports A, B, C, D, and E.

A broadcast message entering Port A with code 1 exits Ports B, C, D, and E (not F).

A non-broadcast message entering Port A with code 1 first attempts to exit Port E: if Port E is busy it tries Port F. If F is busy it toggles between the two waiting for one to become free. Note that this adaptive routing will only work if the fabric of crossbars is symmetrical. The destination will only be reached correctly if the rest of the route fields are valid no matter which of E or F is taken.

Figure 6 shows the route fields needed to communicate using the three crossbar, eight ported fabric. Node 1 performs a single mode (non-broadcast) transaction with node 8 by requesting adaptive route preference F, A, A (field values 077). As port F is available that is selected. At the central crossbar port A is available, as it is at the final crossbar, and the route is established. Node 7 wishing to talk with node 5 requires only a single route field, value 4. Node 6 wishes to communicate with node 4, and requests adaptive routing preference F. Port F is already occupied as the path for node 1's communication, so the crossbar automatically checks port E, and uses it. Route word 047 provides an unencumbered path to node 4. The three conversations occur simultaneously, with a potential fabric throughput of 480 Mbyte/second.

If the transfer initiated by node 6 had occurred first, the route would have been C−F, A−D, E−A. Then when the transfer from node 1 was initiated, the requested route (D−F, C−A, F−A) would have been blocked at the central crossbar until the first

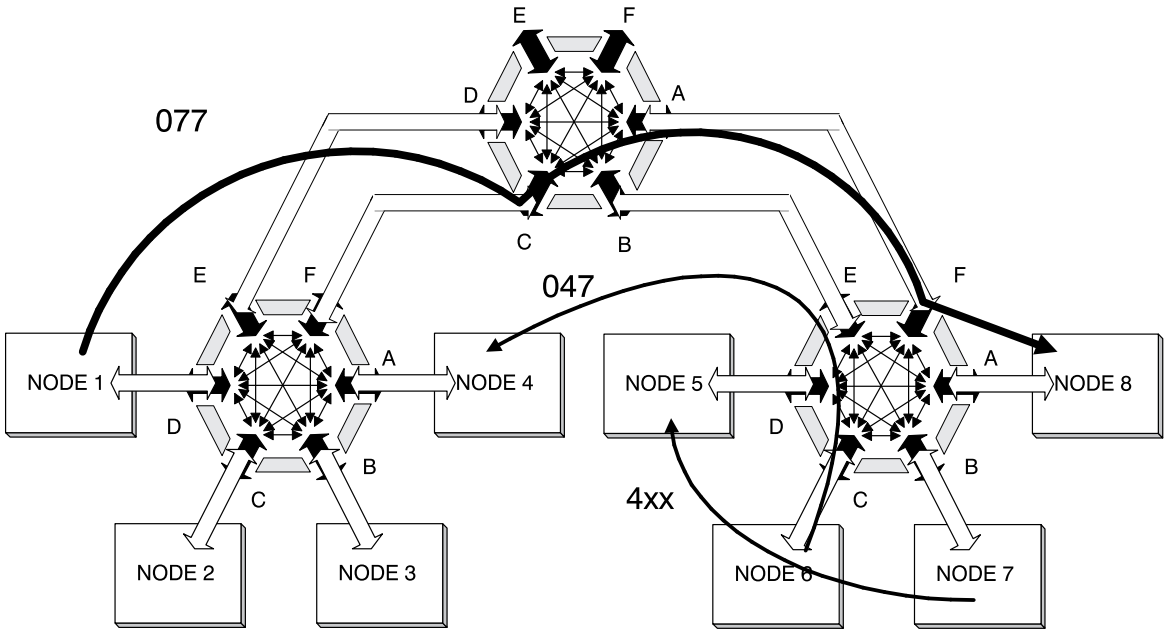


Figure 6. Example Route Programming

transfer was completed or pre−empted. This illustrates the complexity of fabric design, and the care with which route tables should be constructed.

If node 3 wishes to perform a broadcast operation, the route fields it would use are 170, with the broadcast bit set. the first "1" indicates output ports shall be A, C, D, and E. The "7" indicates that port A be used at the middle crossbar. The final "0" causes A, B, C, and D to be outputs.

Latency

The protocol carries an overhead due to the establishment of the route, and the propagation of the slave strobes. Once the connection is established, the data is piped through the fabric, and the length of the pipe is itself an additional overhead. The time from the master first asserting REQO until the slave has acquired the first data word depends upon the number of chips in the fabric between the initiating master and the responding slave. It is also dependent upon whether the requested route is available. These factors are both under the control of the system designer.

Latency could be defined as the time from the point that the initiating master asserts REQO till the data source receives permission to start driving data out (connection latency). Alternatively it may be the time from the assertion of REQO till the data destination has received the first 64-bit data word (data latency). Both calculations are shown below. Also, the data latency on write operations is shorter than reads because write operations take three "trips" through the fabric to establish the connection (one for the route, one strobe from the slave, and one handshake plus data from the master): read operations take four "trips" (one for the route, one strobe from the slave, one handshake from the master, and one for the actual data). Though this may sound excessive, the actual overhead is relatively small compared to the amount of data in each block. Refer to *Figure 7* for an example of routing latency.

For connection latency with no route contention, there are three clock periods per crossbar for the assertion of REQO, followed by 5 periods until the assertion of RPLYIO requesting the write data, followed by one clock per crossbar strobe propagation back to the initiating master, followed by one clock till the assertion of the response strobe by the master—4n+6. This is the same for both reads and writes.

For write data latency, the connection latency is followed by two clocks for the assertion of the first 64-bit word by the master, followed by one clock per crossbar propagation back to the responding slave—5n+8.

For read data latency, following the connection latency the master response strobe propagates back to the slave at one clock per crossbar, then there are two clocks to the assertion of RDCONIO, two clocks to asserting RPLYIO indicating the read data is ready, and two clocks to the assertion of the first 32-bit data word. Then the read data propagates back at one clock per crossbar. One additional cycle is needed for the master to capture the final 32-bit word—6n+13.

Broadcast/Multicast

When an initiating master wishes to broadcast data, or multicast to a subset of the slave population, the following sequence takes place. A write transaction is initiated with the broadcast bit set in the route word. Each crossbar that is presented with this route word waits for all ports to become free, then drives the route from the previous stage through to the next crossbars. When all ports are free, it also drives RPLYIO HIGH back to the previous stage, indicating that the previous stage should change to address. Route shifting occurs just as for the standard write operation described earlier. This operation propagates through the fabric just as a normal write operation, until all receiving slaves have been reached. As each responding slave sees the transaction it drives the RPLYIO line in each phase 0 indicating it is ready to accept the broadcast data. As this is a broadcast operation, the individual crossbars do not generate the RPLYIO signal back towards the initiating master until ALL their 5 responding ports have provided the first RPLYIO. In this way it is ensured that a connection has been established to every node in the fabric, no matter how many crossbars are in the paths. The limitation of using broadcast operations for block transfers is that all nodes must be able to accept data at the full transfer rate. Nodes unable to accept data at this rate must behave as though the broadcast accept code did not match: they should still generate a strobe with every phase 0 on RPLYIO, but should discard the data.

The broadcast accept code provides a means of transmitting data to a subset of the total node population. It is the receiving slave's responsibility to respond to any broadcast with the RPLYIO strobes, but it should discard the data if the code does not match its predefined pattern. Up to 4 groups of nodes can be designated. Nodes that are unimplemented appear to the crossbar to respond because the crossbars have on-chip pull-up resistors which emulate the signals needed for write operations (on RPLYIO and RDCON).

Kill Operations

To enable higher priority transactions to interrupt lower priority ones the kill protocol has been defined. The priority bits in the route header word are compared with the latched priority word associated with a busy port, and if the new request is of higher priority then the crossbar sends a kill request back towards the initiating master of the ongoing transaction. The signal used is REQO, which is connected to the REQI pin of the initiating master. The initiating master may choose to ignore this kill request if the ongoing transaction is a locked transaction. Otherwise the protocol requires that the initiating master terminate the ongoing transaction cleanly, meaning that no further write data will be sent and no further read requests will be sent. After the initiating master drops its REQO, the route is free for use by the higher priority message. Presumably, the lower priority initiating master will re-establish the connection and continue where it left off. This is simply a normal connection request, and must be initiated at least 4 cycles after deasserting REQO.

Transaction Timing

Figure 7 shows a 24-byte write operation taking place across a two-crossbar fabric. The initiating master first samples REQI to determine if it is being accessed as a slave (in phase 1), and if it is not being accessed, it asserts REQO in phase 0, along with the route word. It also drives STROBIO LOW preparatory to asserting it in every phase 1 (as a heartbeat used by the slave).

The slave port connected to the initiating master is on the opposite control phase and therefore receives the REQO signal from the master in phase 1 (on REQI pin). The slave port drives the RPLYIO signal LOW in phase 0. The crossbar must now determine if the requested output port is free, and uses the route word as provided by the master. The route word is valid on XBIO for two clock cycles, then the initiating master provides the route shifted by three bits. The shifted route is driven through the crossbar as the route word for the next stage in the fabric, unless the requested port is not available. If the port is not available the initiating master continues to drive the shifted route on XBIO until the crossbar slave port asserts RPLYIO high (in phase 1) to indicate that the port is available and the connection has been made with it.

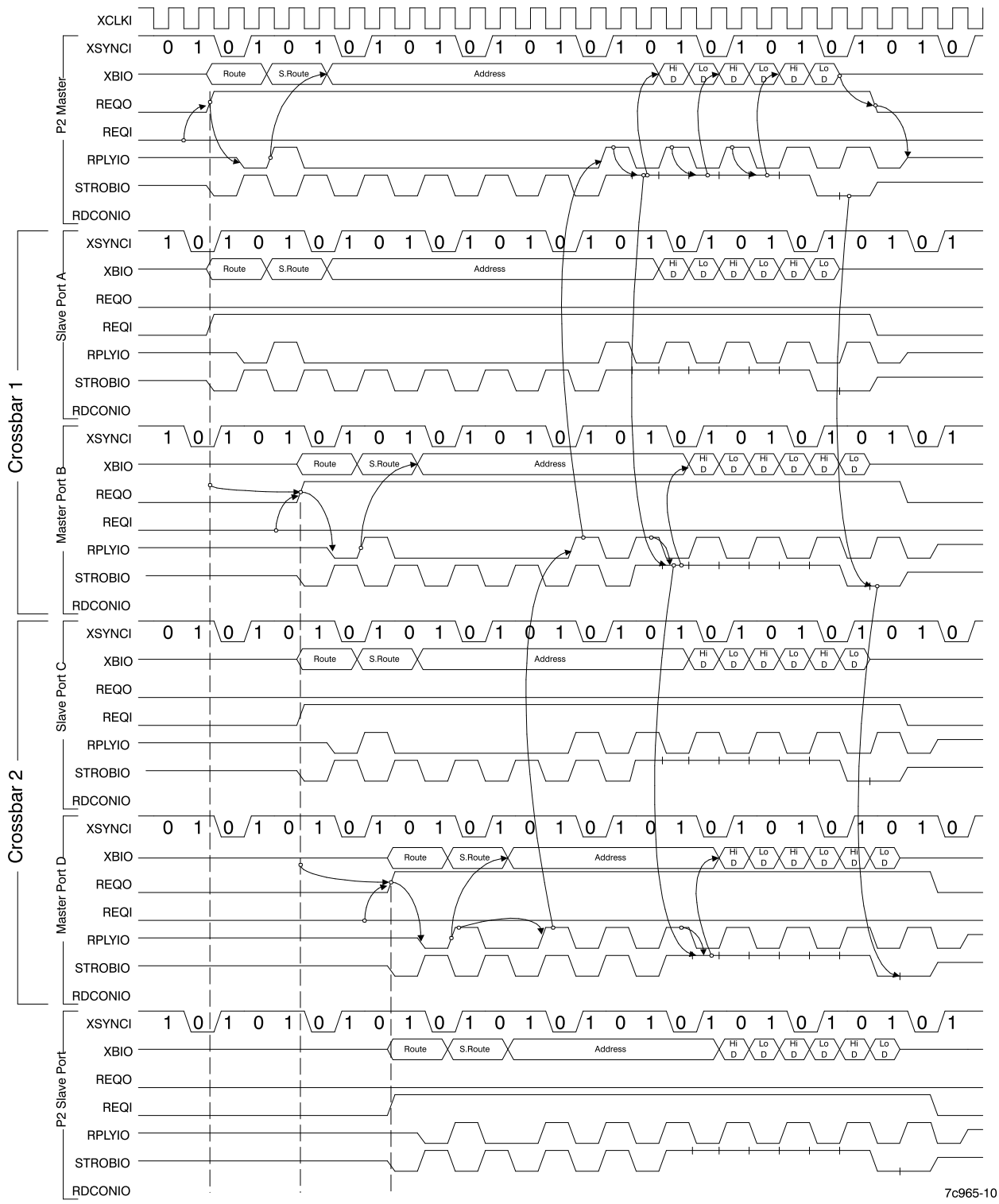


Figure 7. Write Block Transfer Timing

The master interprets RPLYIO going HIGH during its phase 0 as meaning that the slave is ready to receive the address word. The master drives the address word on XBIO in either the next phase 0 if able, or if the master is not high-performance it may drive the address in the second phase 0.

This header information now passes through the fabric of crossbars in an identical manner. The output port of the crossbar becomes a master, and drives REQO along with the route information passed through from the previous stage (in its phase 0) , then

shifts the route and waits for the response before driving the address on XBIO.

When the final crossbar is traversed, the responding slave must latch the "route" word if it wishes to use the high-order address bits, otherwise the final master crossbar will shift them when it provides what it regards as the shifted route. The responding slave drives RPLYIO (in its phase 1) to indicate it is ready to accept the address. Now the route is established through the fabric, and the responding slave has been provided with all the relevant information on the transaction in process (read or write, broadcast or not, starting address). The slave must now indicate to the initiating master that it is ready to accept data (for a write) or source data (for a read).

Figure 7 is a write, and the slave therefore drives a signal indicating that it can accept data. It uses the RPLYIO signal driven HIGH in phase 0. The connected crossbar is not the initiating master, so it does not have any data to present to the slave. The RPLYIO signal is passed through to the other side one clock later. This propagates back through the fabric until the initiating master sees RPLYIO going HIGH in its phase 1. Meanwhile, back at the responding slave, the device is capable of receiving data on each clock edge so it continues to toggle RPLYIO indicating that it can take the data whenever the connected master has some. This string of pulses continues to propagate through the fabric towards the initiating master.

Once the master sees RPLYIO HIGH during phase 1, it drives STROBIO during phase 0. This indicates to the connected slave that the master is about to drive data on XBIO, which it does in the next two phases: in phase 1 it drives D63:32, and in phase 0 it drives D31:0. If the master is capable of providing data on every phase it continues to drive STROBIO during phase 0, followed by the data words. If the connected slave is unable to accept data, it does not drive RPLYIO until ready: this throttles the master. Note that as the protocol is compelled, the slave cannot indefinitely throttle the master, and must provide the strobe signal as soon as it can until the master has ceased driving REQO. In *Figure 7* the master is capable of providing data on every phase, and the slave is capable of receiving it on every phase, so no throttling is employed. The data propagates back towards the responding slave through the fabric in an identical manner at each crossbar.

Meanwhile the responding slave has continued to provide pulses on each phase 0, and these pulses have been piped along the fabric towards the initiating master, many more pulses than the master actually wishes to respond to. This is not significant, and in fact is a feature of the protocol. The responding slave sees STROBIO high during its phase 1, and takes the data on the following two phases.

After three data cycles the initiating master has finished, and ignores further requests on RPLYIO. After the final data word has been transferred during its phase 0, it drives STROBIO LOW during phase 0 indicating that no data will be driven in the next cycle, then continues to drive STROBIO LOW during phase 1 to indicate that the transaction is complete (heartbeat stops). It drives REQO LOW entering phase 0, and three-states STROBIO, thus ending the transfer. The connected slave port responds to the low level on REQI (master's REQO) by three-stating RPLYIO in the next phase (its phase 0). This sequence of controls propagates through the fabric behind the data words, and each crossbar traversed releases the ports as the signals pass through. The responding slave has continued generating strobes on RPLYIO, and upon seeing the connected master drop REQO, it stops generating pulses, and three-states the signal.

Related Documents

CYM9651/2/3/4/5 Raceway Interlink Modules Data Sheet

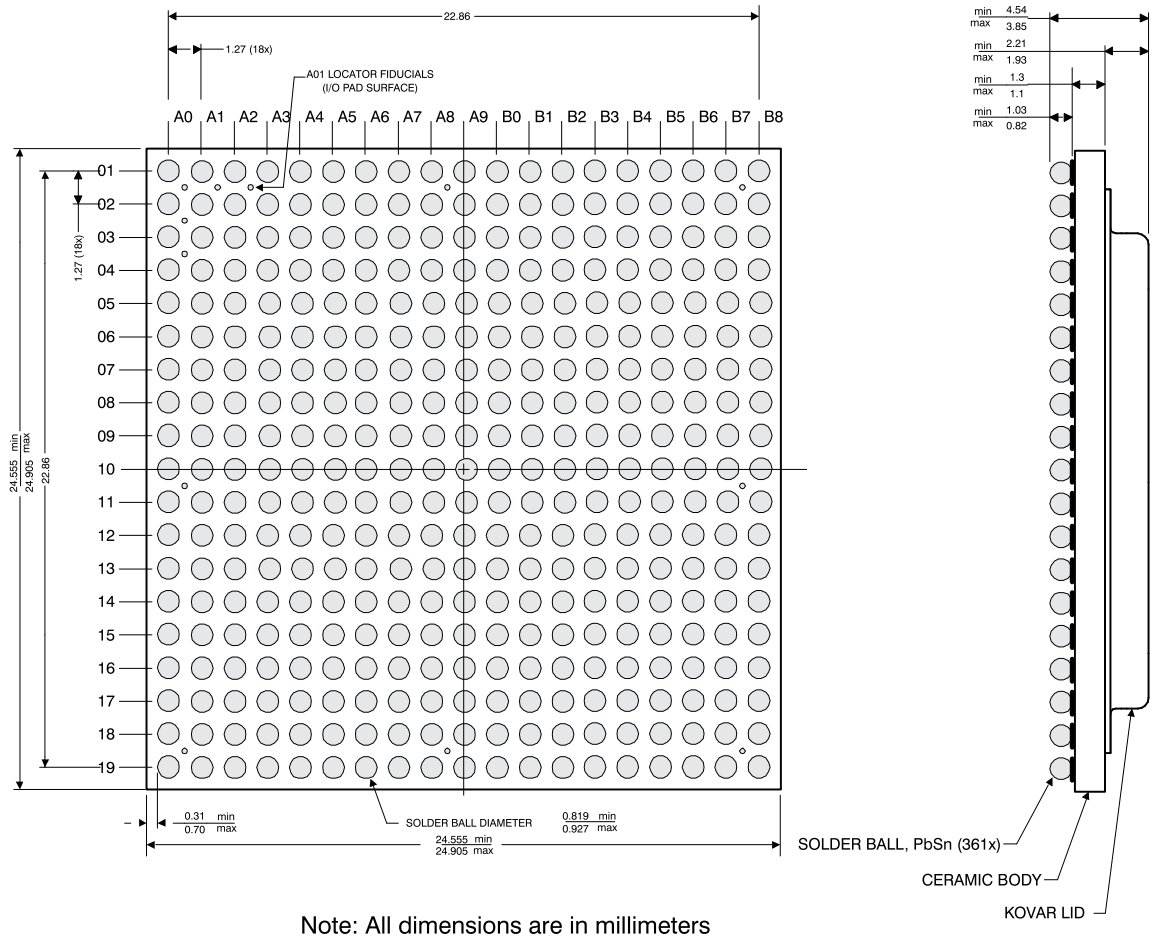
Availability

Contact your local Cypress Sales Office.

Ordering Information

Ordering Code	Package Name	Package Type	Operating Range
CY7C965−GC	G361	361-Pin Ceramic Ball Grid Array	Commercial

Ceramic Ball Grid Array Package



Note: All dimensions are in millimeters

Document #: 38−00461