Cardinal Bidirectional Ring NoC Router Architecture Specification

This document specifies the structure and operation of the Cardinal bidirectional ring network-on-chip (NoC) router, to be used as the building block of a NoC for a multi-core processor for the course project for the EE577B Fall 2021 class. The router is a simple router that uses source routing, fixed-size 64-bit packets and a virtual cut-through style of switching. Additionally, it uses two virtual channels per physical channel to provide for deadlock-free routing in the ring. Since it uses virtual cut-through and packets are only 64 bits, all input and output buffers are 64 bits wide. The following sections detail the relevant aspects of the router architecture.

External Interface (Signal) Descriptions

Refer to Figure 1 and Table 1, which shows all external signals for the router and describes the signals, respectively. Each unidirectional channel contains a 64-bit data portion and two control signals, send (s) and ready (r), for handshaking. There are three input channels: 1 for the processing element (pe), 1 for the clockwise (cw) direction, and 1 for the counter-clockwise (ccw) direction. Similarly there are three output channels with corresponding designations. The router is a clocked (synchronous) device, so there is also a clock input. The reset is synchronous and asserted high. When asserted, the reset signal should initialize all state machines to their idle states and buffer statuses to empty. There is also a polarity signal output which simply indicates if the current clock cycle of the router is odd or even and is used to indicate which virtual channel is being forwarded internally for the current cycle, with the opposite virtual channel being forwarded externally for any clk cycle.

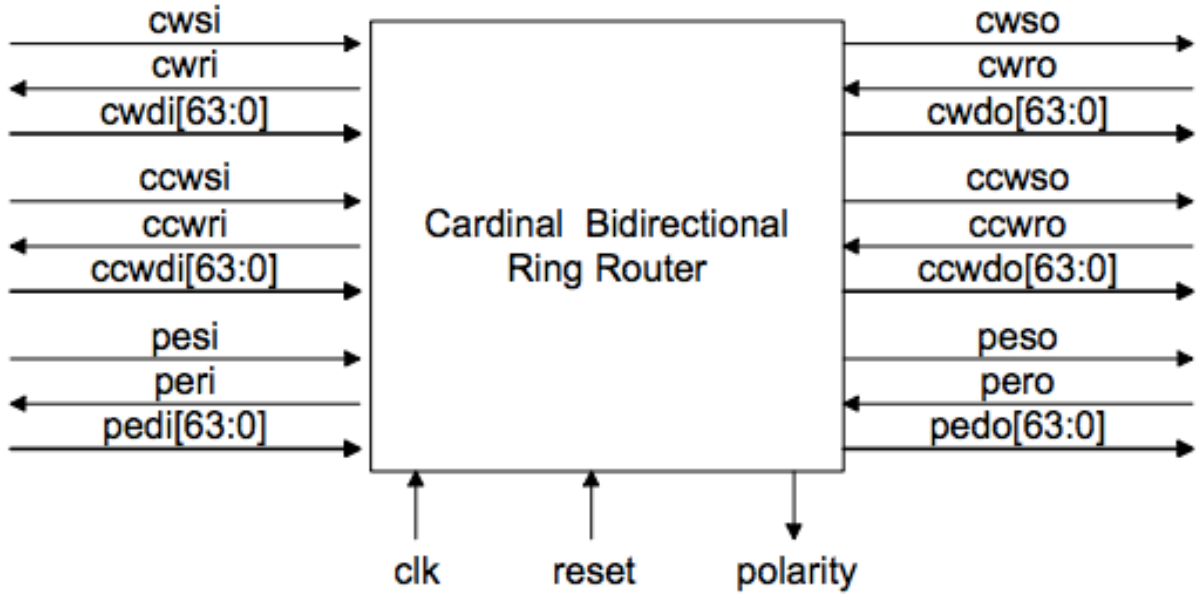

Figure 1: Cardinal Router External Interface

Table 1: Signal Description for Cardinal Router I/O

Signal Name	Signal Type	Bit Width	Description
cwsi	input	1	Send handshaking signal for the clockwise input channel. When asserted, indicates channel data is a valid packet that should be latched at next rising clk edge into corresponding input channel buffer.
cwri	output	1	Ready handshaking signal for the clockwise input channel. When asserted, indicates corresponding input channel buffer is empty and can accept a new packet.
cwdi	input	64	Packet data for the clockwise input channel.
ccwsi	input	1	Send handshaking signal for the counter-clockwise input channel. When asserted, indicates channel data is a valid packet that should be latched at next rising clk edge into corresponding input channel buffer.
ccwri	output	1	Ready handshaking signal for the counter-clockwise input channel. When asserted, indicates corresponding input channel buffer is empty and can accept a new packet.
ccwdi	input	64	Packet data for the counter-clockwise input channel.
pesi	input	1	Send handshaking signal for the processing element input channel. When asserted, indicates channel data is a valid packet that should be latched at next rising clk edge into corresponding input channel buffer.
peri	output	1	Ready handshaking signal for the processing element input channel. When asserted, indicates corresponding input channel buffer is empty and can accept a new packet.
pedi	input	64	Packet data for the processing element input channel.
cwso	output	1	Send handshaking signal for the clockwise output channel. Asserted when associated output channel has packet data to send and cwro signal is asserted.
cwro	input	1	Ready handshaking signal for the clockwise output channel. When asserted, indicates neighboring router has space and can accept a new packet.
cwdo	output	64	Packet data for the clockwise output channel.
ccwso	output	1	Send handshaking signal for the counter- clockwise output channel. Asserted when associated output channel has packet data to send and ccwro signal is asserted.
ccwro	input	1	Ready handshaking signal for the counter-clockwise output channel. When asserted, indicates neighboring router has space and can accept a new packet.
ccwdo	output	64	Packet data for the counter-clockwise output channel.
peso	output	1	Send handshaking signal for the processing element output channel. Asserted when associated output channel has packet data to send and pero signal is asserted.
pero	input	1	Ready handshaking signal for the processing element output channel. When asserted, indicates processing element has space and can accept a new packet.
pedo	output	64	Packet data for the processing element output channel.

clk	input	1	Clock
reset	input	1	Active-high synchronous reset
polarity	output	1	Indicates if current clk cycle is even (0) or odd (1); defined as even during reset, toggles to odd at first rising clk edge after reset is negated, and toggles every cycle thereafter while reset is negated

Packet Format/Header Processing

Each packet to be routed through a ring of Cardinal routers is of fixed length of 64 bits. While this is not a realistic packet size (real packets are much larger), most of the principles of NoCs will still be able to be demonstrated. Thus, the packet size equals the flit size equals the phit size equals the channel width. Given such a small packet size, the network can forward the entire packet from the output buffer of one router channel to the input buffer of the next router channel in one cycle. Similarly, assuming no contention, an entire packet can be forwarded from an input channel buffer to an output channel buffer within a router in one cycle. Refer to Figure 2 for a description of the packet
format. The most significant 32 bits of the packet are used as header information. The most significant sixteen bits of this header information are used for routing purposes, with the most significant 2 bits used for the virtual channel polarity and direction, respectively, while the least significant 8-bits of the
16-bit routing field represent a unary-encoded hop value (the other 6 bits of the routing info are reserved for potential future use and can always be set to 0). The values for the routing header are set by the source node that injects the packet into the network. However, the hop value field will get updated at each hop of a packet's traversal until it reaches its destination, specifically when the packet is transferred internally in a router from an input channel buffer to an output channel buffer. For simplicity, the hop value is unary encoded so that the header processing at each hop of a packet's traversal is simply a right shift of the hop value. For example, if a packet is to traverse 4 hops around the ring, the hop value field should contain 8'h0F when it is first injected. As it traverses through the network, the hop value will be updated at each router in the following sequence: 8'h07, 8'h03, 8'h01, and finally 8'h00. This encoding also makes it very simple to determine which output channel should be requested when the header flit of a packet arrives at a cw or ccw input channel simply by examining the least significant bit of the hop value. More info on routing is given in the next section. The direction bit indicates whether the packet should travel in the clockwise direction (value of 0) or the counter-clockwise direction (value of 1). It is only needed when a packet is first injected into the pe input of a router. The vc field indicates which virtual channel polarity this packet should use: 0 for
even polarity and 1 for odd polarity. The 16-bit source field represents the identification number of a packet's source node, that is, the node which injected the packet into the network. Instructions for setting the value of this field will be given in future assignments.

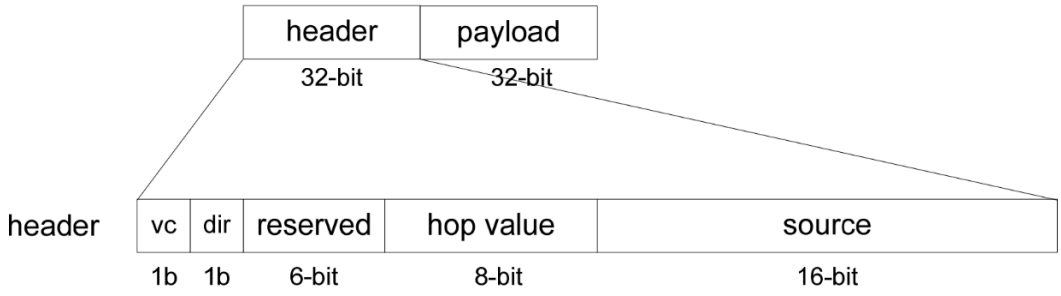

Figure 2: Cardinal Packet and Header Format

Routing/Switching

Recall that the Cardinal Router is a ring router using two virtual channels per physical channel. More info on how the virtual channels are multiplexed is given later, but for all practical purposes the virtual channels can be thought of simply as two sets (even and odd) of buffers sharing physical channels and control logic externally, and can even share some internal logic with the operation defined in this specification. Basically, one set of virtual channels is operational externally on even clock cycles, and the other set of virtual channels is operational externally on odd clock cycles, and vice versa for internal forwarding from input buffers to output buffers. Regardless of virtual channel, when a packet first enters the network on a pe input channel of some router, the routing logic (or address decoder) will first inspect the direction bit to determine if the packet should be routed in the clockwise direction (direction bit equal to 0) or the counter-clockwise direction (direction bit equal to 1). By definition of this router, the hop value field must be non-zero at the time of injection on a pe input channel, i.e., a processing element cannot inject a zero-hop packet through the router to itself. For packets arriving at cw or ccw input channels, the least significant bit of the hop value field is inspected to determine whether to continue the packet in the network or not. If this bit is 0, the packet has arrived at its destination and should be routed to the pe output channel at the proper time. Given this routing paradigm and the virtual channel multiplexing scheme, a representation of the internal components and the input channel to output channel switching can be readily determined and is depicted in Figure 3.

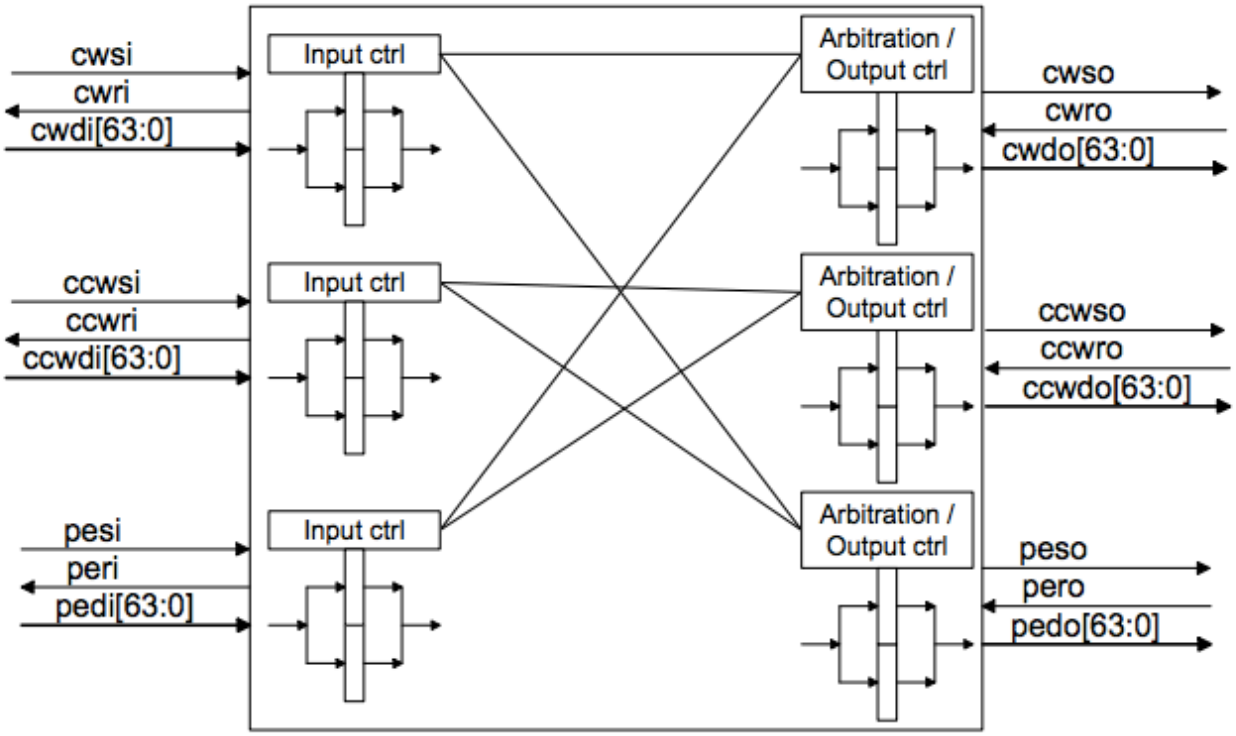

Figure 3: Cardinal Router Internal Components and Switching

Flow Control/Handshaking

Since the network and routers use fixed-size single-phit packets and each input buffer contains enough space to hold an entire packet, flow control is fairly simple. The only complexity arises from the virtual channel multiplexing. The virtual channel polarity is defined very simply based on whether the current clk cycle is even or odd. So the implication is that a router must contain some ability to track

this information, e.g., a toggle function that runs continuously. By definition of this router, this cycle polarity is reset to 0 when the reset input is asserted, indicating an even cycle during reset, and after the first rising clk edge after reset is negated will toggle so that the first full clk cycle after reset is negated will be an odd cycle, with polarity equal to 1. To make the virtual channel implementation as simple as possible, the router supports both input and output buffering where every input and output virtual channel contains exactly one 64-bit packet buffer, as indicated in Figure 3. With this scheme, the following convention is assumed. On even clk cycles, packets in even input virtual channel buffers are forwarded to even output virtual channel buffers assuming they are granted, and any packet in an odd output virtual channel buffer is forwarded to the corresponding odd input virtual channel buffer of the next router, assuming the next router indicates it has space. Conversely, on odd clk cycles, packets in odd input virtual channel buffers are forwarded to odd output virtual channel buffers assuming they are granted, and any packet in a even output virtual channel buffer is forwarded to the corresponding even input virtual channel buffer of the next router, assuming the next router indicates it has space. With that definition, the virtual channels can simply be regarded as buffers that are sharing all control logic and physical wires, with even channels acted upon by the internal logic on even clk cycles and external logic on odd clk cycles, and the converse for odd channels. Furthermore, if a packet is injected on an even virtual channel through a pe port, it traverses even virtual channels for its entire traversal from source to destination, similarly for odd virtual channels.

The channel synchronization signals used for handshaking are send (s) and ready (r). At system reset, all *so* signals should be negated (reset to 0) and *ri* signals should be asserted (set to 1). An input channel controller asserts that it has available buffer space by asserting its associated *ri* signal which is connected to the corresponding *ro* signal of an adjacent router. The *ri* signal can then simply be regarded as an indication of whether the corresponding input buffer is occupied or not.

When an output channel has data that it wishes to forward and if its *ro* input signal is asserted, it asserts its *so* signal along with placing the packet on the data channel. At an input channel, when a *si* input is asserted, on the next rising clock edge the corresponding data on the channel should be clocked into the appropriate input virtual channel buffer, depending on the polarity of the clock, as described above (note that even/odd virtual channel determination is guided purely by the state of the polarity indicator and the vc field in the routing header is ignored by the router; this field will be used in later stages of the project for other components of our design). During the clock cycle following the latching clock edge of the packet into a virtual channel, the input channel routing logic will decode the routing header and request the appropriate output channel. If the output channel is able to grant access to this requesting input channel (e.g., the requested output channel is not occupied and there is no contention or the requesting input channel has arbitration priority), the data is forwarded from the input virtual channel buffer to the appropriate output virtual channel buffer. Then at the next rising clock edge, the *ri* of the input virtual channel that was granted access to its requested output virtual channel will reassert to indicate it is ready to accept new data. An example handshaking timing diagram depicting the handshaking and timing of a packet traversal through two routers is shown in Figure 4 for this non-blocking case. A shorthand notation is used where _1 signals are associated with one router and _2 signals correspond to an adjacent router. In our case these signals could be associated with either a cw or ccw type channel or even the pe channel for the _1 input channel. (Each signal transition in Figure 4 is shown slightly delayed from a clock edge just to depict that the clock edge is a triggering event.) The diagram shows the timing for both a case when a packet is traversing even virtual channels and another case where a packet is traversing odd virtual channels.

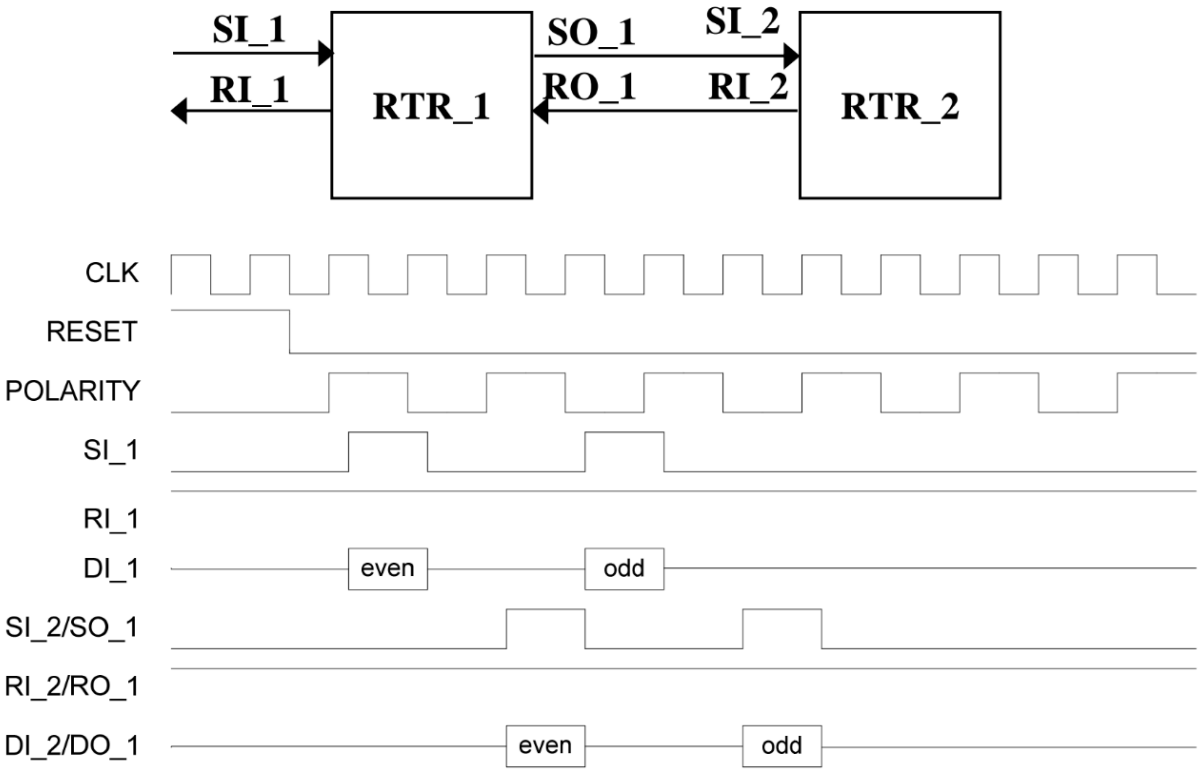

Figure 4: Sample Handshaking Timing Diagram

Note that the output controller / arbitration logic is based on a Mealy state machine where outputs, such as *so* and internal grant signals to input controllers, are functions not only of the current state but also of signals, such as *ro* and internal request signals from input controllers. This is necessary to achieve the contention-free latency goal of a packet phit traversing one router in two clock cycles (one cycle for internal forwarding from an input channel buffer to an output channel buffer and one cycle for external forwarding from the output channel buffer to the corresponding input channel buffer of the next router).

In blocking situations, the *ro* signal input of an output controller will be negated. In such a case, if the output controller has data in an output buffer to forward, it must wait until the *ro* signal asserts before it can forward the data. Figure 5 shows the behavior of handshaking signals for a case where an output channel of router_1 is temporarily blocked. Similarly, blocking situations can occur internally in the router when the output channel buffer requested by an input channel buffer is currently occupied.

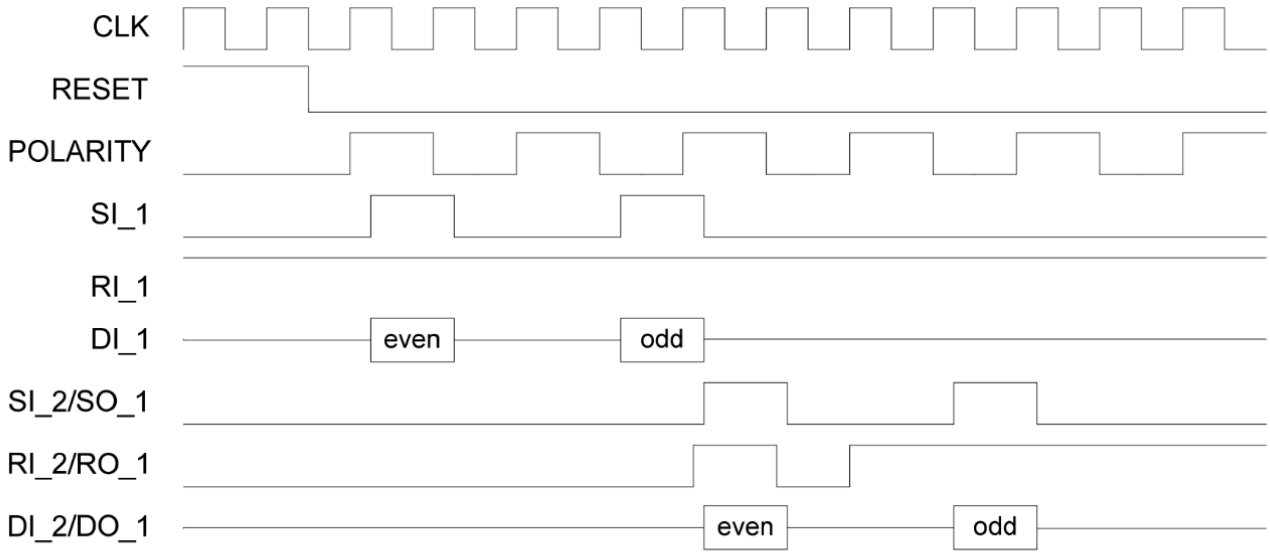

Figure 5: Handshaking in the Case of Blocking

Arbitration/Output Control

A rotating arbitration scheme should be used for arbitrating among multiple requestors for each set of output virtual channels. For example, for the pe output channel of a router, there are two potential input channel buffer requestors: cw, ccw. If at the first time the pe output channel is requested after system reset there are multiple requestors, the channel should be granted to the requestor with highest priority according to the ranking cw, ccw. Once that request is granted, the output controller should log which requestor was granted and reverse the priority scheme so that the other requestor is given priority the next time the same conflict occurs. For example, if cw was the highest priority requestor the prior time the channel was granted, the new ranking for the next request becomes ccw, cw. Since virtual channels are independent, a copy of this priority tracking logic will need to be maintained for each output virtual channel. Similarly, for the cw output, the initial priority order is cw, pe; and for the ccw output, the initial priority order is ccw, pe. Note that as long as there are not multiple (or contending) requests for

an output channel, the priority ranking is irrelevant and doesn't change. The priority ranking changes only when multiple requests conflict and the highest-priority request is granted.

Note also that regardless of how many input channel buffers are requesting a particular output channel buffer, the output channel controller can only grant access if its corresponding buffer is not occupied. Also, when a packet in any input channel buffer is being forwarded to a cw or ccw output channel buffer, note that the header portion of the packet must be updated (shifting of hop value field), while all other packet info passes through unmodified.

Channel Buffer Design

Any channel buffer can simply be implemented as a 64-bit register with a synchronous write port and a read port. Assume that writes occur on the rising edge of the clock. For input channel buffers, when the corresponding *si* signal is asserted, the corresponding data should be latched directly into the appropriate input channel buffer at the rising clk edge, with no computation being performed. Address decoding and output channel buffer requesting will occur on the next cycle when router internal forwarding occurs. There is no requirement for any buffer output to go into a tri-state mode. In fact, the buffer can be designed without a read enable signal such that in every clock cycle the buffer contents are available at the buffer output. Note that much of the control logic of the router depends on the status of either input or output buffers, so the implication is that every buffer has some corresponding full/empty bit or similar status tracking information.

References

It may be helpful to read papers about similar routers. While the references below describe such routers, it should be noted that there are differences between the Cardinal Router and the routers referenced below. In all such cases, this Cardinal Router architecture specification supersedes any other such info.

http://www.isi.edu/~draper/papers/vlsi04.pdf
http://www.isi.edu/~draper/papers/mwscas2000.pdf