Introducing the CYPRESS QuadPort™ Switch (CY7C0430BV)

Introduction

The QuadPort™ Backplane Switch is a revolutionary product that allows the user to re-define the design of the data path. It is a true buffered switch fabric that allows data to be read from and written to the data array at different speeds on each of its four separate ports. It can buffer up to 1 Meg of data in its array and it allows the user to randomly access the data in the array. This provides the maximum flexibility in designing a data path to meet system requirements and it also allows for processing data at line speed.

Overview

The QuadPort Backplane switch is a true four-ported data array which allows a designer to move data through the switch at a rate of 10 Gbps. Each of the ports supports an 18 bit x 133-MHz data path interface which is similar to an SRAM's interface for ease of integration (no custom glue logic is required). Each of the ports run in separate time domains, so data can easily cross time domains. The data array provides 1 Mb of buffering capability which can be randomly accessed by any of the four ports.

The QuadPort Backplane Switch also contains two very powerful features that will ease the overall design requirements of the system. The first is a burst counter which allows the user to read or write large blocks of data into the QuadPort quickly. The second is a mailbox feature that allows for port to port communication across the QuadPort and is an easy arbitration method to prevent simultaneous writes and reads.

Burst Counter Operation

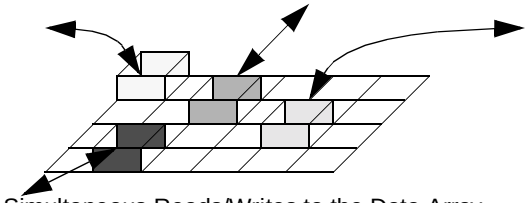
The burst counter works by first loading an address to the counter. This sets the starting location in the array and using the counter increment pin, data can be continuously written to the array without an additional address phase. The end address of the burst can then be read out of the address pins of the backplane switch, using counter read back.

The counter can address the entire data array, when it reaches the end of the array the counter will roll back to zero or to a value determined by the counter mask register. The use of the mask register allows the user to easily segment the QuadPort's data buffer. This allows setting up of data partitions of any size down to a single 18-bit address.

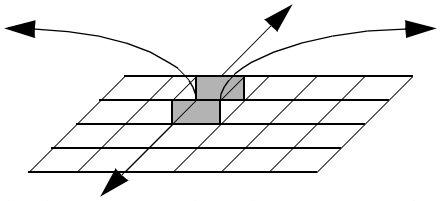
When the counter rolls-over, a counter interrupt occurs for one cycle which can be used to by the network processor to start or stop traffic along the data path.

QuadPort Arbitration

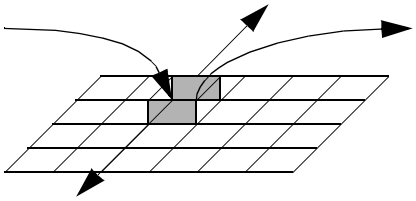
The data array inside of the QuadPort is a shared resource between all four ports. Since reads are nondestructive, all four ports can read the same location in the array at the same time. Writes though are destructive in nature. So when data is written to the array, the system has to ensure that no other port is writing to the same location in the array or reading from it. If two ports write to the same location, or if one port reads the cell while another is writing to it, the data that is read or written is indeterminate (see *Figure 1*).



Simultaneous Reads/Writes to the Data Array



Simultaneous Reads to the Same Location



Simultaneous Writes and Reads are Unpredictable

Figure 1. Basic QuadPort Operation.

To ensure that a conflict does not occur, the designer that is integrating a QuadPort in a system must enforce an arbitration scheme. There are several ways to do this. These include address comparison to determine port access, enforcing a timing circuit that does not allow for simultaneous operations, or by using the QuadPort's built in mailboxes. Since the first two schemes are highly system dependent, the following will describe the best way to employ the mailbox.

Arbitration using the QuadPort's Mailbox

The upper four addesses of the QuadPort can be used to pass messages between ports. Address FFFFh is for port 1, address FFFEh is for port 2, address FFFDh is for port 3, and address FFFCh is for port 4. If port 4 has a message for port

1, it would write to address FFFFh. Once the message is written the INT pin for port 1 will assert. This alerts port 1 that there is a message in its mailbox. Once port 1 reads the message the interrupt is cleared. The other ports can read address FFFFh but only when port 1 reads it does the interrupt get cleared.

This feature can be used to pass tokens. The token can consist of up to 18 bits. If the upper ten bits were the ten most significant bits of the address, the other eight bits can specify the number of 1K blocks that the port wanted to reserve. So if port 4 wanted to reserve an address 4K block starting at 0000h, it would write 00004h to addresses FFFFh to FFFDh. This will cause an interrupt to occur on Ports 1, 2, and 3. External logic attached to the ports will read addresses FFFF to FFFDh and will not write to addresses 0000h to 1000h. Port 4 can proceed to write to that block. When port 4 was done with the address block it could then write the same value to address FFFFh to FFFDh to let the other ports know that it no longer needs that block of memory.

The format of the 18 bits can change based upon system requirements. The mailbox feature though is a quick and convenient way to design an arbitration scheme for the QuadPort.

Why Design with a QuadPort Backplane Switch?

Dual Ported or FIFO memories have been traditionally used in the communications industry as a data speed matching tool. These memories were used as simple data buffers which stored data coming from a high-speed bus until a slower bus was ready for the data. This is shown in *Figure 2*.

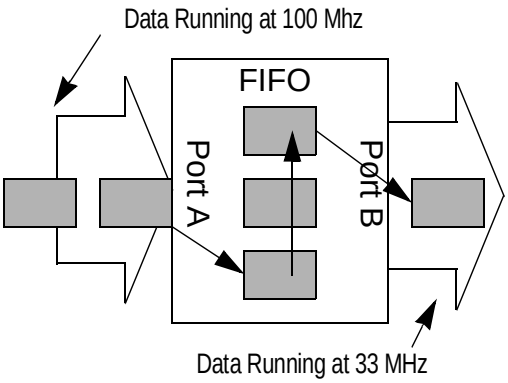


Figure 2. Data Speed Matching.

These devices serve only a limited application because they provide only buffering capability.

The QuadPort Backplane Switch does more than just buffer data though. In essence, the QuadPort provides the user the ability to directly manage the data in the data path on the fly removing the need to have network processors or coprocessors in the data path. This allows for processing at reduced latency and allows for a more efficient and dedicated design for the processor.

Figure 3 shows a QuadPort in a system with a look-aside architecture. The Network Search Engine would pull the addresses of exception packets or those packets that require higher processing from the data path and passes the addresses to the network processor. The processor would then read the packet from the buffer, process it, and then write the packet back to the buffer before it is read out.

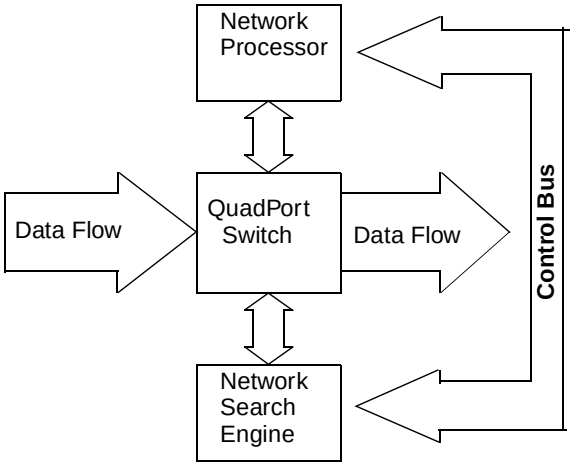


Figure 3. A QuadPort System.

As this example shows, the QuadPort provides a unique challenge to the communications engineer because it gives the engineer so much flexibility that it merits re-evaluating the entire data path design. The effort will be worthwhile because designing with a QuadPort can reduce the complexity and cost of a design. The following sections will describe how to use the QuadPort in some common applications. Each section attempts to explain the problem facing network designers, how the QuadPort can be integrated into the design, and finally the benefits of using the QuadPort.

Problems with Communications Design

The challenge that is being faced by most network designers is how to perform the increasingly complex processing required to support today's network services and still maintain the overall throughput of modern broadband systems.

Traditional network designs placed the network processor directly in the data path. The data was buffered inside the processor while it did the layer 2 table look up to determine the destination of the packet. This was sufficient when the data path speed was a mere 1.5 Mbps and the only processing done was a routing table look-up.

Today's network processor must not only determine the destination address, it must determine the priority of the packet based upon its quality of service (QoS), it may also need to determine what type of protocol the packet is to determine its egress port, and finally it must determine if the packet requires any special processing such as encryption in the case of Virtual LANS or Virtual Private Networks. To process at OC-48 wire speed, the network processor would have to perform all of its functions within a 65 ns time frame.

To handle all the processing, the network processing tasks has been divided between the network processor and one or more network coprocessors. The coprocessors are designed to support specific functions such as classification and forwarding. Although this approach allows for faster processing because tasks are done in parallel, it comes at a penalty since many of these processors and coprocessors have proprietary interfaces which make integration of the processors together

difficult without some amount of glue logic unless you default to the SRAM interface.

Even a coprocessor solution though does not solve the fundamental problem faced by network designers, which is how to process data at line speed? As *Figure 4* shows, the processor becomes the bottleneck in the system unless it either processes the data faster than it is coming in or the processor buffers multiple packets.

.

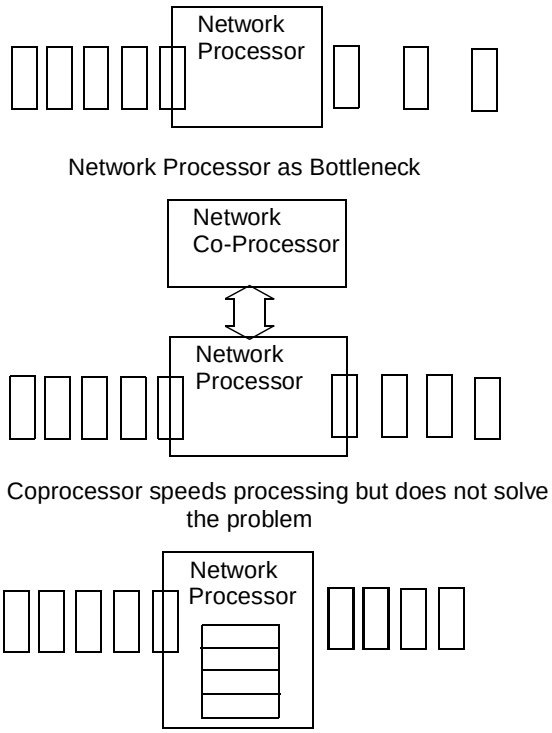


Figure 4. Network Processing in the Data Path.

Placing a buffer inside a network processor though is a significant investment. For example, a network processor that must handle an entire OC-48 frame will have to handle 48 STS-1 frames that consist of 6480 bits per frame. The total memory needed than is 304K bits. If the data is not completely processed before the next frame enters the device, you need to buffer more than one frame. This could change the amount of memory needed to 608K or more. This is a substantial portion of a 1 million gate FPGA just to store the data before you invest any logic in actual processing functions. This does not include the additional logic that is required to manage and maintain the data path. The resource requirement as a percentage of gate count is represented in *Figure 5*.

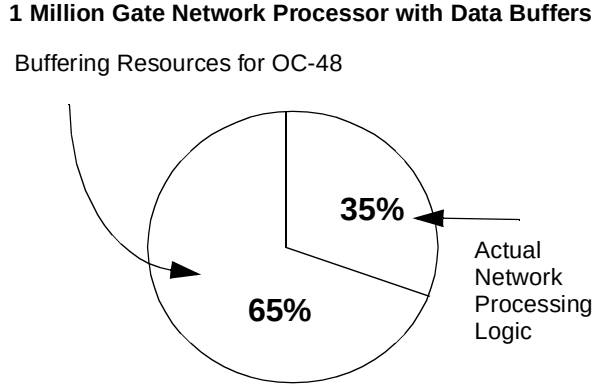


Figure 5. Buffering Resources Required for OC-48.

Even if the device used external SRAMs as buffers, there is a turnaround time impact of writing the entire packet to memory and then reading the packet back after the processing is done on the header.

Not only does adding buffering capability to the network processor take a lot of resources in the processor, buffering is not actually required because the routing decisions that need to be made are done on the header and not on the entire packet. The packet header contains addressing, protocol, class of service, and quality of service information which form the basis for queuing, routing, and scheduling decisions. The processor does not validate the payload, relying instead on higher network layer protocols.

Table 1. Packet Header as Percentage of total size

Protocol	Packet Header%
SONET	3%
Frame Relay	Variable < 5%
ATM	10%

As *Table 1* shows only a small amount of the data packet is actually processed. So there is no need to buffer the packet other than to allow the processor time to process the data.

The QuadPort Solves the Problem

The QuadPort Backplane Switch Fabric ensures that processing can be done at wire speed. It does this by removing the processor from the data path and also by buffering the data inside its memory array. To explain how this works, let's review the QuadPort system original proposed in *Figure 3*. The data path in the system flows through the QuadPort. The Network Search Engine (a Network C-processor), scans the data for exception packets or packets that require additional processing because of service features in the router. The network search engine then sends the address of the packet to the network processor. The network processor will then read

or tap that packet from the data stream and process it. Once the packet has been processed, it will be written into the QuadPort without interrupting the data flow. The processor must insert the data back into the data flow in the same position that it was taken out to maintain packet order integrity.

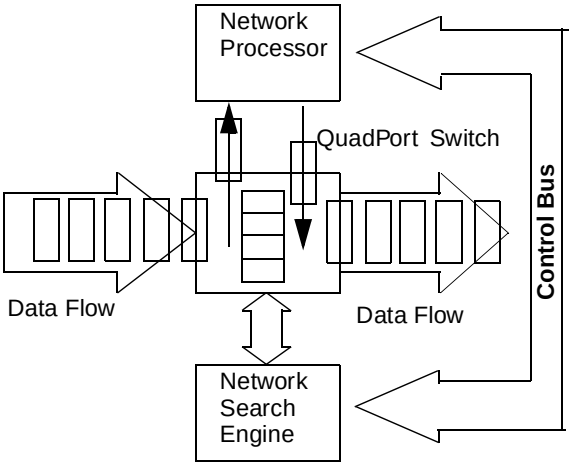


Figure 6. Network Processing with a QuadPort.

Figure 6 is an illustration of the QuadPort system with the data path overlaid on it. The key to this design is that:

1. Data is still being received by the QuadPort while the network processor works on the packet
2. The network processor does not need to run at wire speed because the data will remain buffered in the QuadPort until the processor is ready
3. To process at wire speed, the processor needs to run just fast enough to ensure that the QuadPort's buffer is not exceeded.

Another nice feature of this design is that communication between the network processor and coprocessor do not necessarily have to go across a separate control bus. Either device can use the QuadPort's mailbox feature to send address information back and forth. This simplifies the control bus design because additional glue logic is not required.

Conclusion

The Cypress Semiconductor CY7C0430 QuadPort backplane Switch is a revolutionary product that will redefine network communications. It is ideal for network line cards because it takes the network processor out of the data path, eliminating the bottleneck caused by holding a packet in the device until the processor can complete its tasks. Data can now be transmitted and received independent of the time it takes to process the packet. The QuadPort also provides a easy way to integrate coprocessors in a design without the expense and time of developing glue logic.

For further information, please visit the Cypress web site at www.cypress.com. The web site also provides the latest data sheets, models, and any related documentation.