EXPANDING PORT COUNT USING A 4-PORT SPACEWIRE ROUTER

Session: SpaceWire Components

Short Paper

Jennifer Larsen

Aeroflex Colorado Springs

4350 Centennial Blvd. Colorado Springs, CO 80907

E-mail: *Jennifer.Larsen@aeroflex.com*

ABSTRACT

The UT200SpW4RTR[2] is a four port SpaceWire router that is capable of operating at data rates from 10 to 200 Mbps, supporting path, logical, and group adaptive routing and offers an effective and simple solution to many networking requirements. This router has a total of 5 ports, 4 are SpaceWire compatible ports and the 5th is a parallel HOST port.

The HOST port allows access to the routers configuration and status registers as well as access to any of the 4 SpaceWire ports. Data may be passed from the HOST port to the SpaceWire ports and vice versa. HOST ports of multiple UT200SpW4RTR devices may be interfaced together using an FPGA which will route data between multiple routers. The interfacing FPGA will need to contain logic that will support reads and writes from the UT200SpW4RTR HOST ports as well as a lookup table block. The look up table block will contain the routing information such that data can be passed to and from the HOST ports of multiple routers.

1 UT200SPW4RTR BASIC FUNCTIONALITY

The Aeroflex 4-port router implements a non-blocking crosspoint switch and a "Round Robin" arbitration scheme allowing all 5 receive ports access to all 5 transmit ports. Path and logical addressing are supported per ECSS-E-ST-50-12C [1], and lookup table storage is replicated five times giving each receive port a dedicated block of memory for logical addressing. Configuration of lookup tables, as well as access to internal registers may occur through any of the 5 ports using a simple configuration protocol. A group adaptive function is also provided for 2 ports when implementing logical addressing.
Each of the four SpaceWire ports is capable of running at an independent speed. This allows for systems to be configured with nodes/instruments running at different speeds.

The HOST port of the 4-port router is composed of both a receive and transmit FIFO. The transmit FIFO (inputs to router) are write capable by the external hardware. Full and Almost Full flags are provided to help the user prevent overwriting the FIFO and should be monitored by external hardware or the interfacing FPGA. Data will be

written into the FIFO on the rising edge of the clock when /TX_PUSH is “Low”. The receive FIFO (outputs from router) receives data from one of the SpaceWire ports and is then read from the receive FIFO on the rising edge of the system clock when /RX_POP is “High”. FIFO status flags Almost Empty and Empty flags are provided for proper data management.

The HOST port transmit interface is connected to a read logic block that controls SpaceWire data flow and determines the addressing scheme being used for the packet received, where as the HOST port receive interface is connected to a write logic block to the receive FIFO interface. Each of the SpaceWire ports on the UT200SpW4RTR contain a read logic block. The basic concept of the read and a write logic blocks should be replicated in the FPGA.

Read logic blocks are connected to each of the SpW ports as well as a the HOST port. The SpaceWire ports read logic block have internal FIFO monitor flags that can not be accessed externally. The read logic block monitors the empty flag on the receive FIFO and reads a byte of data whenever the FIFO is not empty. This block also checks the first byte of data after an EOP to determine the port address or whether a configuration transaction will be initiated. For Path or Logical addressing, the Read Logic Block uses the first byte of data after an EOP/EEP.

The write logic blocks control the data to the transmit FIFOs and the HOST receive port and the SpW transmit ports. A "Round Robin" arbiter manages access and makes sure only one Read Logic Block accesses the Write Logic Block. If more than one receive ports is waiting to send data out of the same output port, the arbiter gives each receive port equal opportunity for access.

The arbiter starts counting whenever a request for that port is received from any of the five receive ports. The count is from Port 1, Port 2, etc, until the count reaches Port 5, looks for configuration commands, and then starts over. The configuration block will be accessing the Write Logic Block when read configuration packets are requested. The HOST port of the UT200SpW4RTR will allow the system designer to interface multiple 4-port routers together with out compromising the count of the SpaceWire capable ports.

2 SYSTEM ARCHITECTURE

Figure 2 shows a notional diagram using three UT200SpW4RTR routers interfaced to an FPGA, a microprocessor could be controlling the system. This example generates a 12-port router using three 4-port routers. These concepts can be applied to generate a router with a larger port count.

The interfacing FPGA should contain a look up table space that is responsible for routing data between the HOST ports of the 4-port routers, as well as, read and write logic blocks as described insection1.0. The look up table space should be configured such that a given logical address will be routed to the HOST port of the destination 4-port router. The look up space can be sized based on the number of 4-port routers being interfaced to the FPGA.

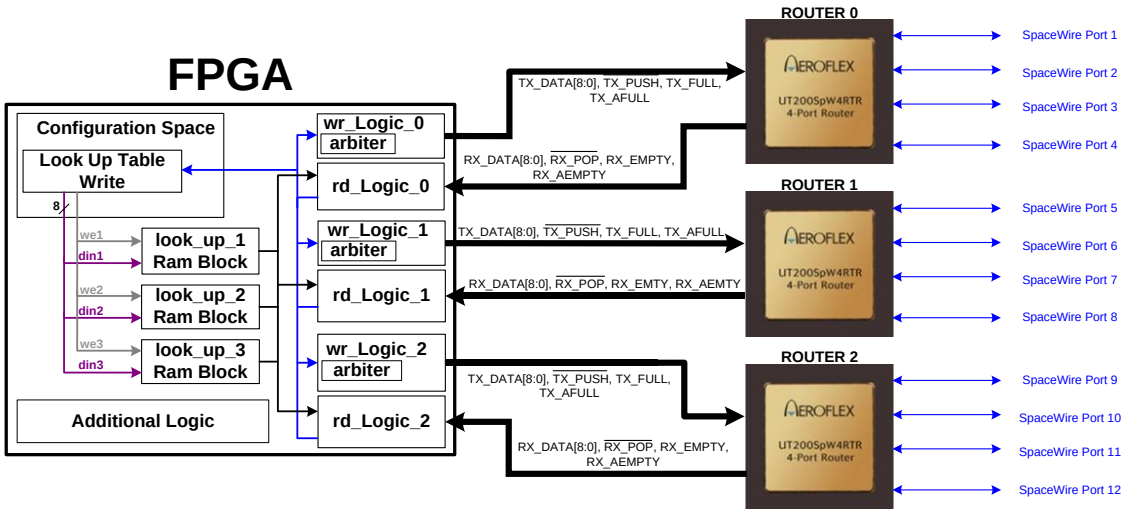

Figure 2. Notional Diagram of FPGA requirements

The HOST port interfaces of the UT200SpW4RTR devices, including TX_DATA[8:0], /TX_PUSH, TX_FULL, TX_AFULL, RX_DATA[8:0], /RX_POP, RX_EMPTY, RX_AEMPTY, should be connected to the FPGA for data transfer and status monitoring. The read and write logic blocks will control the flow of data from one HOST port to the other. Figure 3 shows a block diagram of the main logic block requirements needed to handle arbitration, routing, and data transfer in the FPGA.

Table 1 shows an example of the contents required for the FPGA look up table space. Each of the 12 SpW ports and 3 HOST ports requires a unique look up table location, to ensure proper data routing. Using Table 1, assume SpaceWire port 6 (port 2 on Router 1) has data that needs to be routed to SpaceWire port 11 (port 3 on Router 2). A packet with header 0x30 is sent to Port 6. Router 1 decodes lookup table address 0x30 and sees that data should be sent to the HOST port or local port 5 of Router 1. The FPGA rd_Logic_1 read logic block decodes packet header 0x30 and sees that data should be sent to Router 2 wr_logic_2. Router 2 HOST port decodes lookup table address 0x30 and sees that data should be sent to Port 11 (port 3 on Router 2). The data packet will be routed out on Port 11 of the expanded router.

Table 1. Example Lookup Table Space.

Address	Router 0	Router 1	Router 2	FPGA	SpW Port
0x20	1	HOST	HOST	1	1
0x21	2	HOST	HOST	1	2
0x22	3	HOST	HOST	1	3
0x23	4	HOST	HOST	1	4
0x24	HOST	1	HOST	2	5
0x25	HOST	2	HOST	2	6
0x26	HOST	3	HOST	2	7
0x27	HOST	4	HOST	2	8
0x28	HOST	HOST	1	3	9
0x29	HOST	HOST	2	3	10
0x30	HOST	HOST	3	3	11
0x31	HOST	HOST	4	3	12

3 SYSTEM PERFORMANCE CONSIDERATIONS

The routing of SpW data to and from each of the router devices should follow a flow similar to that shown in Figure 3. The flow in Figure 3 may be modified to optimize data through put and overall system efficiency.

The flow of data to the interfacing FPGA can start with the FPGA in idle state, where the /RX_POP FIFO flag is being monitored for incoming data. Once a /RX_POP FIFO flag is asserted active low, the FPGA will decode which 4-port router has the active /RX_POP flag. After the active router is identified the FPGA starts registering the data present on the RX_DATA[8:0] lines until an End of Packet (EOP) is received.

When an EOP is received the first byte of data needs to be examined. The first byte of data should contain the logical look up information as described in Table 1. Byte 1 will then be compared to the FPGA Logical Look up table address such that the data will be routed to the correct destination 4-port router.

Assuming the first byte of data contains a valid router address the FPGA needs to monitor the corresponding 4-port routers TX_FULL to ensure that there is available space in the routers HOST transmit FIFO. The FPGA then asserts the corresponding 4-port routers /TX_PUSH FIFO flag, this starts the transfer of data from the FPGAs register to the destination 4-port router. Once an EOP has been received by the destination 4-port router the FPGA returns to the idle state and wait for the next data transaction to occur.

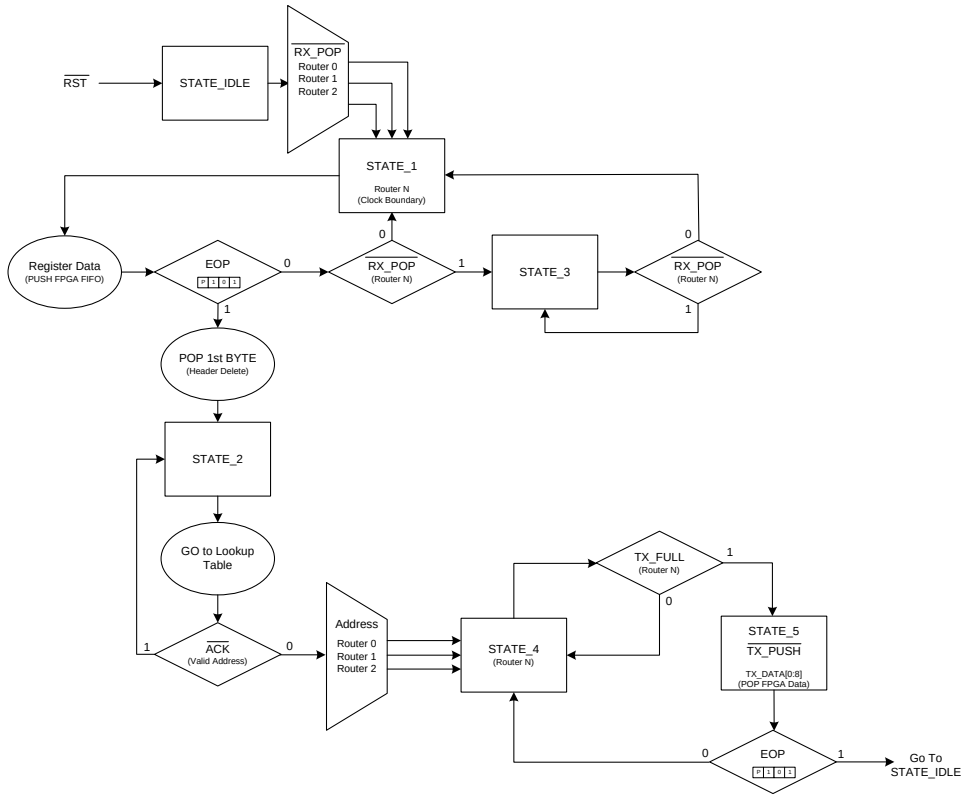

Figure 3. FPGA Data Handling Flow Diagram

4 CONCLUSION

Interfacing multiple UT200SpW4RTRs together using the HOST port offers a simple solution to increase port count. The HOST ports should be interfaced together using a FPGA that will act as an arbiter, equipped with look up tables, between the multiple UT200SpW4RTR devices.

The interfacing FPGA needs to contain logic supporting reads and writes to and from the HOST ports read and write FIFOs. There will be a look up table block which contain the routing information such that data can be passed to and from the HOST ports of multiple routers.

5 REFERENCES

[1] ESA Publications Division, "SpaceWire Standard Document ECSS-E-ST-50-12C", The Netherlands, July 30, 2008.

[2] Aeroflex, "UT200SpW4RTR 4-port SpaceWire Router Datasheet", Colorado Springs, Colorado, February 2010.