

EE-789: Mini Project

4x4 Switch

Prajwal D Kamble, 17D070024

November 29, 2020

1 Problem Statement

Design and Test an 4x4 switch, which can handle packets with a maximum size of 256 bytes (ie. 64 words). Validate and characterize above switch for the following kinds of traffic

- Data comes in from a single port and goes out to a single port
- Data comes in from both ports and goes out to the same destination port
- Data comes in from both ports and is distributed across the destination ports (incoming packet is equally likely to go to any of the destination ports).

2 Introduction

An NxM switch transfers the data from one of the N input ports to one of the M outputs. Input packets arrives at the switch. The packet length (in bytes) is a multiple of 4-byte quantities (words). The first word in the packet is called the header, and its has basic information about packet (destination - 8bit, data length - 16bit, sequence id - 8bit).

2.1 Block diagram

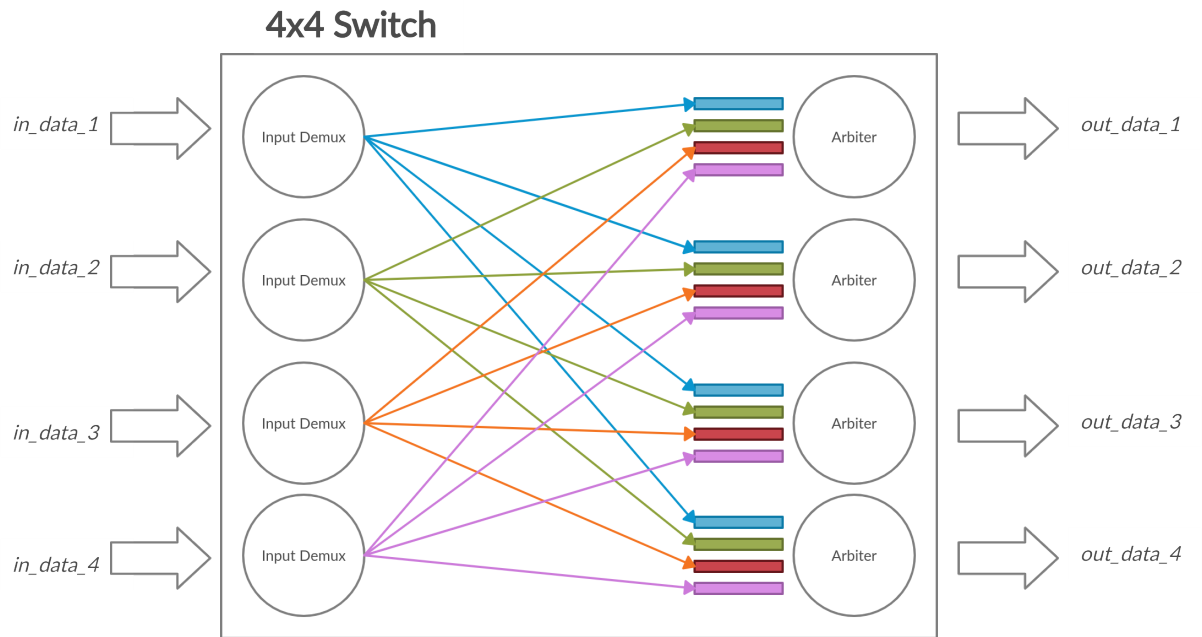


Figure 1: 4x4 Switch

3 Design decisions

Instead of monolithic design we will be using modular design strategy. We need to implement four identical input demuxes and four identical output arbiter. We would be designing one input demux and one output arbiter and copy that module to generate other input and output ports.

4 Implementation

4.1 Input demux

Input demux has pipeline running continuously and accepting input at each loopback. if *count_down* is '0' then new packet is accepted and recognised as a header. Header is decoded and splits into destination, packet length in words (4-byte) and sequence id with bit lengths of 8, 16 and 8 respectively. '1' is appended to the input to identify it as a valid packet. Then it is sent to the corresponding queue according to destination.

4.2 Output arbiter

Output arbiter accepts 33 bits from all queues corresponding to 4 input demuxes if corresponding *read_from_* is true. Active packet refers to the current queue from which packet is being sent out currently. *priority_queue* is a 3-bit variable which stores the priority for next active packet.

We check the valid packets from four queues and whether there is any valid packet to send or not. We then apply priority select logic which is explained in next section. We get the next active packet and next priority packet as a output from priority select logic. Next active packet decides which packet needs to be sent out.

4.3 Priority Select

Priority select implements **Round-robin scheduling** method to decide priority. Priority changes from 0—1, 1—2, 2—3, 3—4, 4—1. *priority_queue* is 3-bit variable and stores the queue index which is at current priority.

5 Design verification

Design is verified by the testbench in C. Report statements were added to debug the code and see intermediate states.

```
[5177]inputPort_2_Daemon> count_down := 0x0002
[5178]inputPort_2_Daemon> received_input_word
[5178]inputPort_2_Daemon> input_word := 0x0000003f
[5179]inputPort_2_Daemon> new_packet_summary
[5179]inputPort_2_Daemon> dest_id := 0x00
[5179]inputPort_2_Daemon> pkt_length := 0x0000
[5179]inputPort_2_Daemon> seq_id := 0x3f
[5180]inputPort_2_Daemon> write_status
[5180]inputPort_2_Daemon> send_to_1 := 0x0
[5180]inputPort_2_Daemon> send_to_2 := 0x1
[5180]inputPort_2_Daemon> send_to_3 := 0x0
[5180]inputPort_2_Daemon> send_to_4 := 0x0
[5180]inputPort_2_Daemon> data_to_outport := 0x10000003f
[5180]inputPort_2_Daemon> new_packet := 0x0
[5180]inputPort_2_Daemon> last_dest_id := 0x02
[5180]inputPort_2_Daemon> next_last_dest_id := 0x02
[5180]inputPort_2_Daemon> count_down := 0x0001
[5181]inputPort_2_Daemon> received_input_word
[5181]inputPort_2_Daemon> input_word := 0x0100400c
[5182]inputPort_2_Daemon> new_packet_summary
[5182]inputPort_2_Daemon> dest_id := 0x01
[5182]inputPort_2_Daemon> pkt_length := 0x0040
[5182]inputPort_2_Daemon> seq_id := 0x0c
[5183]inputPort_2_Daemon> write_status
[5183]inputPort_2_Daemon> send_to_1 := 0x1
[5183]inputPort_2_Daemon> send_to_2 := 0x0
[5183]inputPort_2_Daemon> send_to_3 := 0x0
[5183]inputPort_2_Daemon> send_to_4 := 0x0
[5183]inputPort_2_Daemon> data_to_outport := 0x10100400c
[5183]inputPort_2_Daemon> new_packet := 0x1
[5183]inputPort_2_Daemon> last_dest_id := 0x02
[5183]inputPort_2_Daemon> next_last_dest_id := 0x01
[5183]inputPort_2_Daemon> count_down := 0x0000
[5184]inputPort_2_Daemon> received_input_word
[5184]inputPort_2_Daemon> input_word := 0x00000001
[5185]inputPort_2_Daemon> new_packet_summary
[5185]inputPort_2_Daemon> dest_id := 0x00
[5185]inputPort_2_Daemon> pkt_length := 0x0000
[5185]inputPort_2_Daemon> seq_id := 0x01
```

Figure 2: Terminal screenshot

```

[5405] outputPort_2_Daemon>         p4_valid          := 0x0
[5406] outputPort_2_Daemon> state_summary
[5406] outputPort_2_Daemon>         active_packet      := 0x0
[5406] outputPort_2_Daemon>         down_counter       := 0xffff
[5406] outputPort_2_Daemon>         priority_queue     := 0x3
[5406] outputPort_2_Daemon>         read_from_1        := 0x0
[5406] outputPort_2_Daemon>         pkt_1_e_word       := 0x102004001
[5406] outputPort_2_Daemon>         read_from_2        := 0x0
[5406] outputPort_2_Daemon>         pkt_2_e_word       := 0x102004004
[5406] outputPort_2_Daemon>         read_from_3        := 0x1
[5406] outputPort_2_Daemon>         pkt_3_e_word       := 0x000000000
[5406] outputPort_2_Daemon>         read_from_4        := 0x1
[5406] outputPort_2_Daemon>         pkt_4_e_word       := 0x000000000
[5408] outputPort_2_Daemon> send_information
[5408] outputPort_2_Daemon>         started_new_packet  := 0x0
[5408] outputPort_2_Daemon>         next_active_packet  := 0x0
[5408] outputPort_2_Daemon>         send_flag          := 0x0
[5408] outputPort_2_Daemon>         p1_valid           := 0x1
[5408] outputPort_2_Daemon>         p2_valid           := 0x1
[5408] outputPort_2_Daemon>         p3_valid           := 0x0
[5408] outputPort_2_Daemon>         p4_valid           := 0x0
[5409] outputPort_2_Daemon> state_summary
[5409] outputPort_2_Daemon>         active_packet      := 0x0
[5409] outputPort_2_Daemon>         down_counter       := 0xffff
[5409] outputPort_2_Daemon>         priority_queue     := 0x3
[5409] outputPort_2_Daemon>         read_from_1        := 0x0
[5409] outputPort_2_Daemon>         pkt_1_e_word       := 0x102004001
[5409] outputPort_2_Daemon>         read_from_2        := 0x0
[5409] outputPort_2_Daemon>         pkt_2_e_word       := 0x102004004
[5409] outputPort_2_Daemon>         read_from_3        := 0x1
[5409] outputPort_2_Daemon>         pkt_3_e_word       := 0x000000000
[5409] outputPort_2_Daemon>         read_from_4        := 0x1
[5409] outputPort_2_Daemon>         pkt_4_e_word       := 0x000000000
[5411] outputPort_2_Daemon> send_information
[5411] outputPort_2_Daemon>         started_new_packet  := 0x0

```

Figure 3: Terminal screenshot

6 Results

Current implementation till now works for all combinations of input ports and output port 1, 2 and 4.

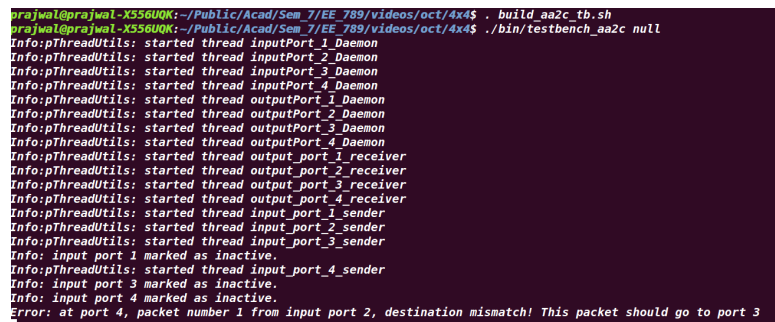
```
Rx[1] at output port 2 from input port 4
Rx[2] at output port 2 from input port 3
Rx[3] at output port 1 from input port 4
Rx[3] at output port 2 from input port 4
Rx[4] at output port 1 from input port 1
Rx[4] at output port 2 from input port 1
Rx[5] at output port 1 from input port 2
Rx[5] at output port 2 from input port 2
Rx[6] at output port 2 from input port 3
Rx[6] at output port 1 from input port 3
Rx[7] at output port 2 from input port 4
Rx[7] at output port 1 from input port 4
Rx[8] at output port 2 from input port 1
Rx[8] at output port 1 from input port 1
Rx[9] at output port 2 from input port 2
Rx[9] at output port 1 from input port 2
Rx[10] at output port 1 from input port 3
Rx[10] at output port 2 from input port 3
Rx[11] at output port 2 from input port 4
Rx[12] at output port 2 from input port 1
Rx[13] at output port 2 from input port 1
```

Figure 4: Terminal screenshot

Any packet with destination as output port 3 gives error

*Error: at port 4, packet number 1 from input port 2, destination mismatch!
This packet should go to port 3*

This means packet is arriving at output port 4 instead of port 3. I checked corresponding report statements in run.trace file to look for intermediate states. I checked that all intermediate states were giving correct result. Relevant screenshots are attached below.



```
prajwal@prajwal-X556UQK:~/Public/Acad/Sem_7/EE_789/videos/oct/4x4$ . build_aazc_tb.sh
prajwal@prajwal-X556UQK:~/Public/Acad/Sem_7/EE_789/videos/oct/4x4$ ./bin/testbench_aazc null
Info:pThreadUtils: started thread inputPort_1_Daemon
Info:pThreadUtils: started thread inputPort_2_Daemon
Info:pThreadUtils: started thread inputPort_3_Daemon
Info:pThreadUtils: started thread inputPort_4_Daemon
Info:pThreadUtils: started thread outputPort_1_Daemon
Info:pThreadUtils: started thread outputPort_2_Daemon
Info:pThreadUtils: started thread outputPort_3_Daemon
Info:pThreadUtils: started thread outputPort_4_Daemon
Info:pThreadUtils: started thread output_port_1_receiver
Info:pThreadUtils: started thread output_port_2_receiver
Info:pThreadUtils: started thread output_port_3_receiver
Info:pThreadUtils: started thread output_port_4_receiver
Info:pThreadUtils: started thread input_port_1_sender
Info:pThreadUtils: started thread input_port_2_sender
Info:pThreadUtils: started thread input_port_3_sender
Info: input port 1 marked as inactive.
Info:pThreadUtils: started thread input_port_4_sender
Info: input port 3 marked as inactive.
Info: input port 4 marked as inactive.
Error: at port 4, packet number 1 from input port 2, destination mismatch! This packet should go to port 3
```

Figure 5: Terminal screenshot

Input ports 1, 3 and 4 were made inactive and destination was set to 3 for above test case.

My first guess is that testbench must have some error as I have checked report statements and didn't find any discrepancy.

7 Conclusion

4x4 Switch is created successfully with a little testing bug to resolve.