DaFPGASwitch

Lauren Chin lmc2265 Fathima Hakeem fh2486 Teng Jiang tj2488
Ilgar Mammadov im2703 Irfan Tamim it2304

27 February 2024

Contents

1 Overview

The predominant aim of the project is to simulate a network switch with an FPGA.
A network switch is a hardware device used to connect multiple computers and other network devices in a computer network. It operates by forwarding data packets to their intended destination ports based on the destination addresses contained within the packets. This enables data exchange within a local area network (LAN) or wide area network (WAN). In this project, we're planning to simulate a network switch with a simplified version of packet definition.
On the software side, a stream of packets will be generated with software and will be passed to hardware (the "Switch") through a software-hardware interface (over Altera's Avalon Communication Fabric). The software is also in charge of receiving the packets - which also includes validation of packets and performance profiling. The reason for the software implementation for packet generation is to avoid the extra complexity generated from a real Ethernet connection and to also be able to easily generate and profile different patterns of workload.
On the hardware side, the MAC address is translated to the port number with the MAC address table. The packet will be added to corresponding buffers connected to a crossbar fabric. The scheduler decides which packets get launched onto the crossbar fabrics. The baseline scheduler will be using a simple FIFO policy, while we are also planning to implement a priority-based scheduling algorithm. In addition, the packet format will include a checksum to check the integrity of the data transfer.

2 Components Breakdown

This is the overall architecture of our system. The software in the processor takes care of packet generation and verification and communicates with the hardware through the Avalon bus. The hardware part is connected with the Checksum module, MAC-to-Port translation module, and the buffer and crossbar fabrics.

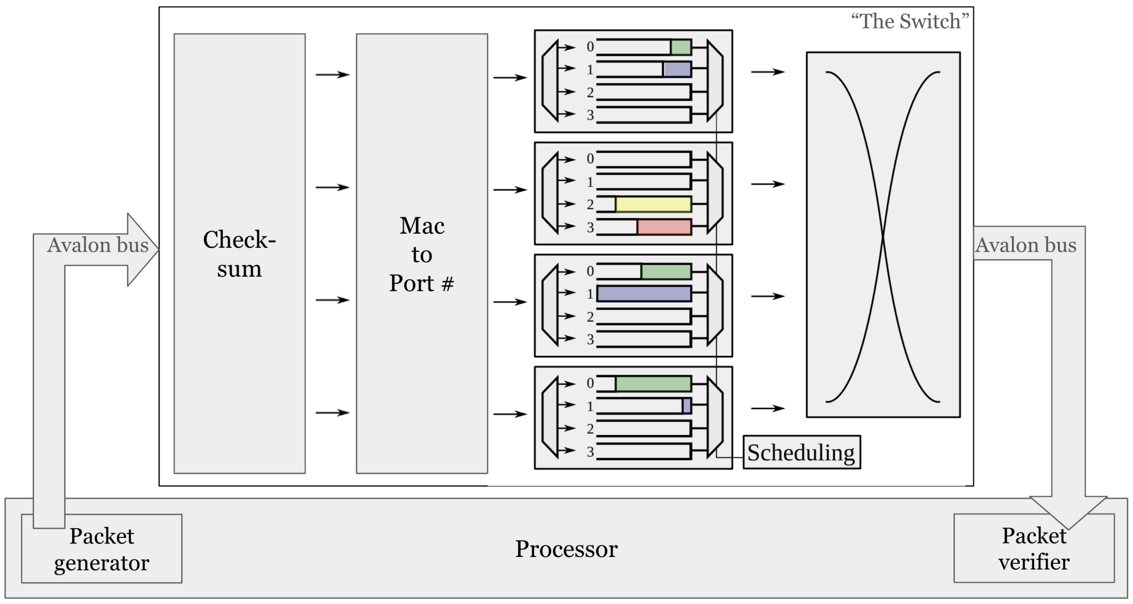

Figure 1: Overall architecture of DaFPGASwitch.

2.1 Packet Definition

For an effective implementation of our FPGA switch, we do not need to use all parts of the real Ethernet data frames. The essential parts that have to be included in the packets are source and destination MAC addresses which will be used for MAC to port translation. We preserve the possibility of variable length specified by the Length portion of the packet, but we might keep the data payload length fixed for simplicity. For checking the data integrity, our packets will include the CRC bits and the priority portion for implementing priority-based scheduling.

Dest MAC(6 Bytes)	Src MAC(6 Bytes)	Length (2 Bytes)	Data	CRC (4 Bytes)	Priority (1 Byte)

Table 1: Packet Definition

2.2 Peripherals: Avalon

Our system will use the Avalon Memory-Mapped (Avalon-MM) interface to send and receive arbitrary packets. Master and slave peripheral modules will be implemented to write data to and read from the address-specified registers. The interaction will be pipe-lined to implement Avalon-MM bursting. If needed, we will use the Intel Quartus Prime IDE to aid with the implementations.

2.3 MAC Address Table

This part converts the destination MAC address to the corresponding port to send packets.
We're planning to use a hash table to implement the mapping. We plan to first have a static MAC-to-port translation table predefined so this standalone part can be bypassed when we test other parts. An example is provided in Table 2.

MAC Address	Port
AA:BB:CC:DD:EE:FF	0
00:1A:2B:3C:4D:5E	1
12:34:56:78:90:AB	2
F1:E2:D3:C4:B5:A6	3

Table 2: An example static MAC Address Table.

Our next-step plan is to have a dynamic MAC Address table that gets updated on-demand and has advanced features like aging. An example is illustrated below.

MAC	Port	TTL

Table 3: We start with an empty MAC address table. TTL here means time to live for this entry (in seconds)

$\longrightarrow$

MAC	Port	TTL
AA	1	60

Table 4: A device AA at port 1 sends to CC. It will first add itself to the table. Since CC is not in the table, AA does a broadcast.

$\longrightarrow$

MAC	Port	TTL
AA	1	59
CC	3	60

Table 5: CC on port 3 receives the packet and also adds itself to the table. AA's TTL decreases over time.

2.4 Networking Fabrics: Crossbar and buffers

A crossbar fabric is a multidimensional array that provides a unique path for packets from each input port to each output port. Since crossbars provide a single unique path from an input to an output, only one packet must be chosen from the buffers at the input. A simple crossbar fabric is illustrated in Figure 2 [1].

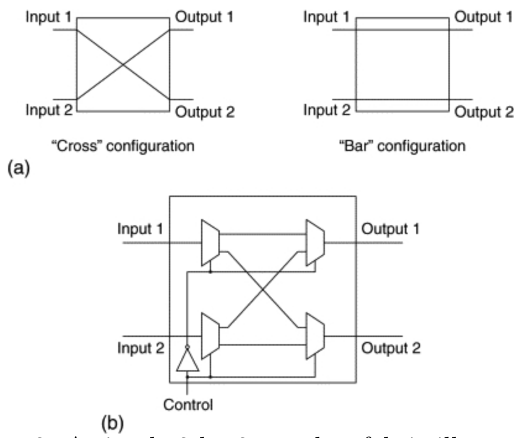

Figure 2: A simple 2 by 2 crossbar fabric illustration.

After the packet goes through the MAC address lookup and knows which port it will go to, the packets will be stored in input buffers known as Virtual Output Queues (VOQs). The VOQs are going to be per egress port. The scheduler decides which packet goes through the Crossbar fabric, which can be a simple FIFO or a priority-based algorithm with the priority portion in our packet definition.

2.5 Testing

We will test the system on two major components: correctness, and performance. The testing will be done in software, in C, and will include different scheduling algorithms and workloads. Correctness will be used to test the effectiveness of the checksum model, as well as whether the packets were sent and received correctly. Performance will be tested in terms of latency and throughput of the entire system.

3 Milestones

1. Write software in C to generate packets with simulated MAC addresses and arbitrary data.
2. Define the Avalon-MM master and slave modules as the hw-sw interface.
3. Implement the checksum module, the MAC address table module, and the network fabrics module (individually, since we've decoupled each component) in SystemVerilog. Create testbenches for each component.
4. Extend the VOQs to priority-based queues in SystemVerilog.

5. Define workload pattern on the software. End-to-end testing of sending and receiving simulated packets.

Bibliography

[1] Dimitrios Serpanos and Tilman Wolf. Chapter 4 - interconnects and switching fabrics. In Dimitrios Serpanos and Tilman Wolf, editors, Architecture of Network Systems, The Morgan Kaufmann Series in Computer Architecture and Design, pages 35–61. Morgan Kaufmann, Boston, 2011.