Project
Ethernet Switch

Robert Foss (dt08rf1@student.lth.se)
Mikael Sahlström (dt08ms2@student.lth.se)
Mikael Nilsson (dt08mn2@student.lth.se)

Contents

1 Introduction

This project aims to create a layer-3 switch, using a Digilent Nexys-3 board and a PModNIC. The task of such a switch is to avoid flooding big networks with ethernet frames that only belong to a subset of that network, by using a CAM table. The purpose of the CAM table to map MAC addresses to ethernet ports. The project's layer-3 switch will only feature two ethernet ports. Also some kind of firewall is desired in the switch to give the administrator an ability to block certain IP addresses.

Focus will especially be on making the switch able to maintain high troughput.

Embedded systems are desired to consume small amounts of power, hence different strategies to reduce power consumption will be used while keeping the performance at a reasonable level.

2 Project proposal

The project consists of two main parts: ethernet and ip CRC checks implemented in hardware and IP header decoding with switching decisitions implemented in software. Figure 1 shows an rough design description, though in reality packages can flow in eather direction, even out the same NIC they came in.

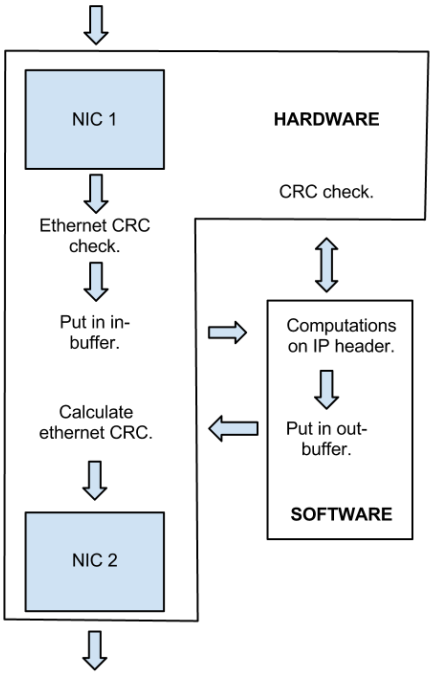

Figure 1: Principle schematic of the projects system design.

2.1 Software

The interrupt handler will be executed when packets have arrived in memory (receive buffer). It will create a new packet job and put it in a worklist that a thread will work on. The thread will unpack the packet for inspection and later pack it and put it in memory (the send buffer). At layer 3, the CAM table will be inspected and the exit ethernet port will be known. If there is nothing in the CAM table, the packet will be sent through all ethernet ports.

2.2 Hardware

There will be a receive controller that will validate incoming packets CRC value. If the CRC is valid, then it will attatch a header to the packet that will contain the length of the ethernet packet and information of which port the packet originated from. The packet will then be put in the memory (a receive ring-buffer) via the memory controller. The receive controller will, when the packet is in memory, generate an interrupt via the interrupt controller to make sure the software will start processing the packet.

The send controller will read packets from memory (a send ring-buffer). Afterwards the controller will strip the packet's header and send it to the correct ethernet port.

2.3 I/O processes

The two NICs LAN8710 and PModNIC will be interfaced via the MII and the SPI interfaces respectively. The MII and the SPI interfaces will be implemented in two controllers which abstract away the specifics of communicating with either NIC.

The receive controller will read all available incoming data on the MII and SPI controller and via the memory controller write it to the cellular ram. When data is read in the receive controller the interrupt controller is notified which will trigger a CPU interrupt in the Microblaze CPU.

In a similiar fashion the send controller will read data from the cellular ram via the memory controller and send it to the correct NIC via the MII or SPI controller.

Communication between differen parts of the system is described in figure 2.

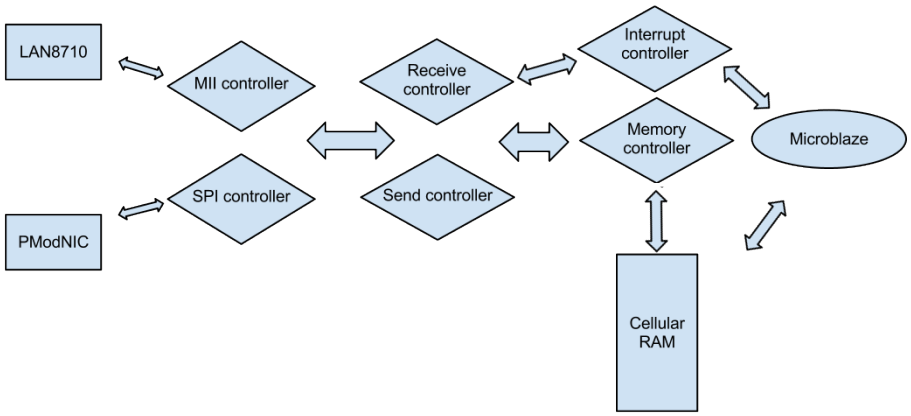

Figure 2: Principle schematic of the communication between different parts.

3 Time plan

1. *Oral presentation* - 1 day
2. *Project proposal* - 2 days
3. *Ethernet on both NIC's* - 7 days
   (a) Set control registers.
   (b) Implement SPI support for PModNIC.
   (c) Implement ring buffer in RAM for ethernet data.
   (d) Implement CRC checks for Ethernet in hardware.
   (e) Transmit and receive Ethernet frames.
4. *Internet protocol* - 2 days
   (a) Implement CRC offload for IP in hardware.
5. *CAM table* - 2 days
   (a) Populate CAM table with NICs and MAC addresses.
6. *Performance optimizations* - 5 days
   (a) Determine average packet sizes.
   (b) Make memory/core-count trade-off.
   (c) Investigate further functions to offload from CPU.
7. *Validation* - 4 days
   (a) Validate unicast, broadcast and multicast.

   (b) Validate high load and out of memory situations.

8. *Project report* - 2 days
9. *Project resentation* - 1 day