WEB SERVER

DESIGN PROPOSAL

CSEE4840 Embedded System Design

Prof. Steven A. Edwards

Franklin Ma	(As/R)
Howard Wang	(Mo/Me)
Victor Wang	(W/Mo)
William Wong	(N/E)

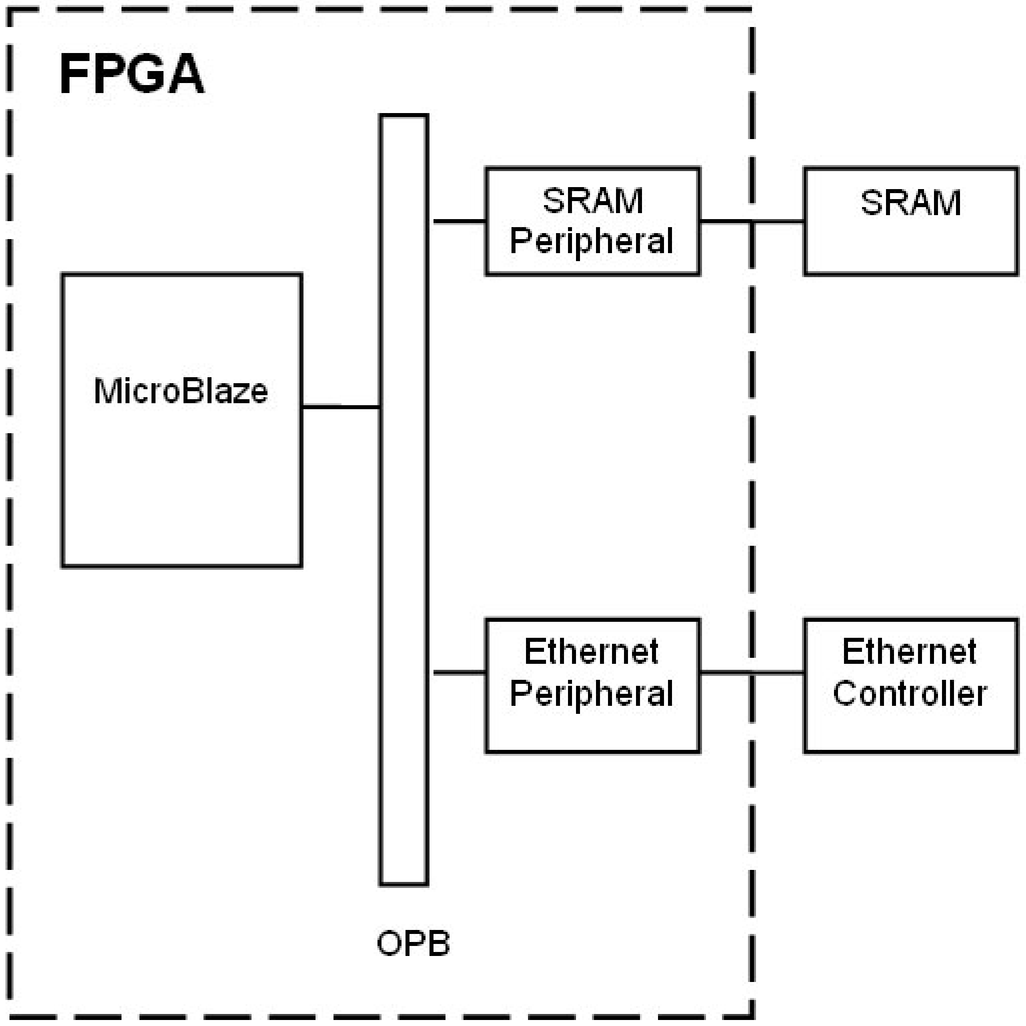
INTRODUCTION

Our aim is to implement a web server on the XESS XSB-300E that will accept requests from clients for data through HTTP over TCP/IP through the onboard Ethernet controller. We will design the VHDL modules that interface with the SRAM, where we will likely store the data that is to be transmitted to the client (unless there is sufficient on-chip memory), and the on-board Ethernet controller. The following protocols will be necessary for our web server:

HTTP	Web page request/response
TCP	reliable communications
IP	low-level data transport
ICMP	diagnostics (Ping)

We will be programming these protocols in low-level C to be carried out by MicroBlaze.

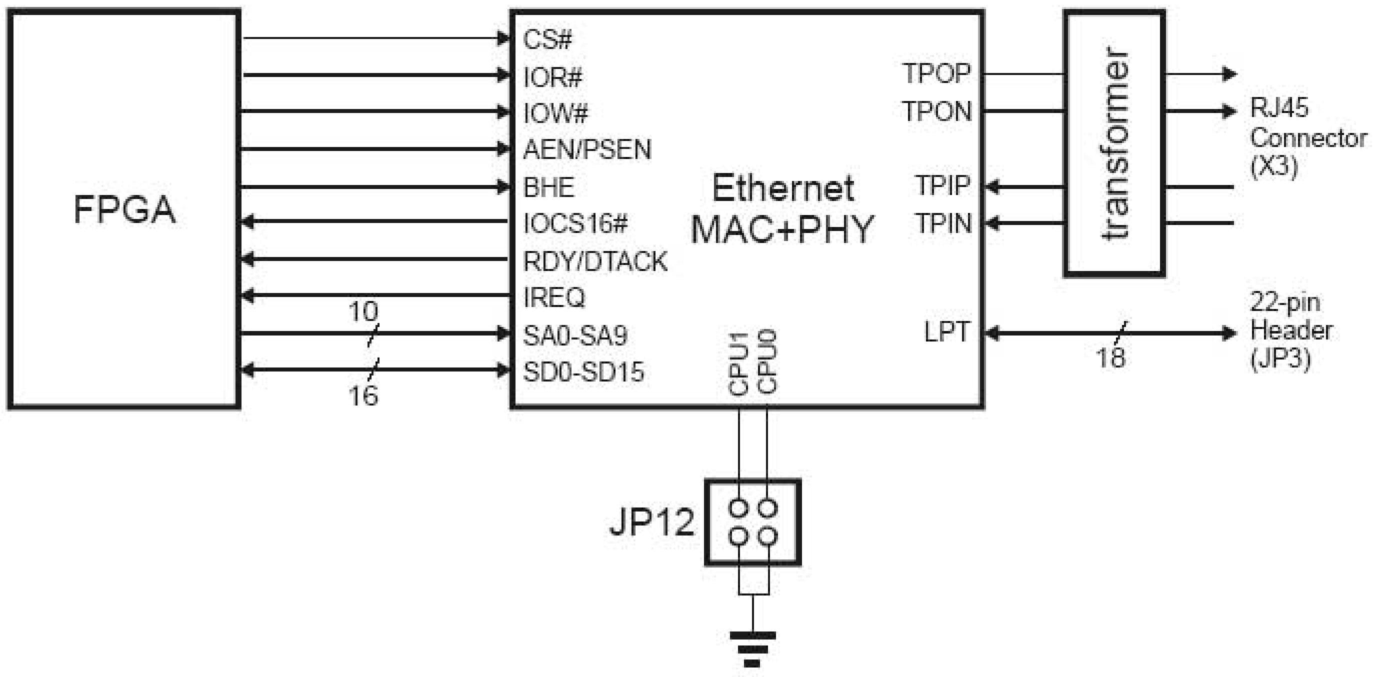
Block Diagram – General Archtecture:



ETHERNET

We will be using the AX88796 Ethernet Controller and communicating with it through the custom peripheral on the OPB. The Ethernet Controller interface is outlined below:

AX88796 Ethernet Controller

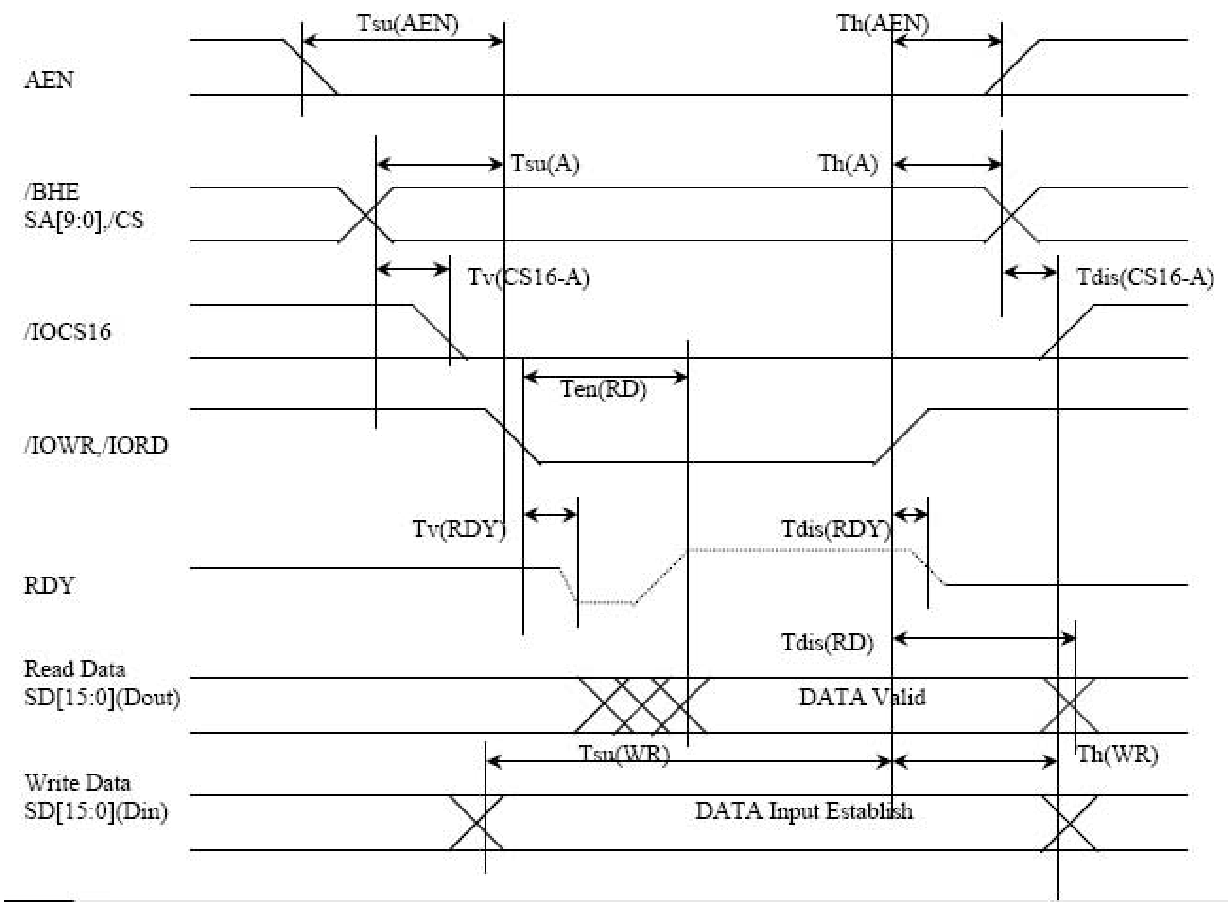


Packets will be constructed and transmitted according to the Ethernet Protocol. We will be initializing and interfacing with the controller through the control registers outlined in the datasheet of the AX88796

Interfacing with the Controller
We will drive the Ethernet Controller by simulating read and write cycles on an ISA bus as follows:

1) Set the address lines.
2) During the read cycle, the data lines from the MicroBlaze will serve as inputs. During a write cycle, they will be outputs and be set with the data to be written.
3) Set the appropriate read or write signals (active low).
4) If it is a read cycle, the Ethernet controller will drive the bus with the appropriate data.
5) Deassert the data if it's a read and latch the data received from the MicroBlaze if it's a write.
6) Disable output drivers in order to free up data bus.

Timing Diagram for the Ethernet Controller



Data Reception & Transmission
It is necessary to fetch and process incoming packets in small portions. The remote DMA controller is used for this process. Incoming packets will be stored in the SRAM before attempting to process them. There are two main processes of data transmission: packet reception and packet transmission.

During packet reception, we will need to complete the following series of tasks:

- Finding the address and length of packet.
- Checking for reception of multiple packets.
- Checking the packet for error and the packet buffer for overflow.

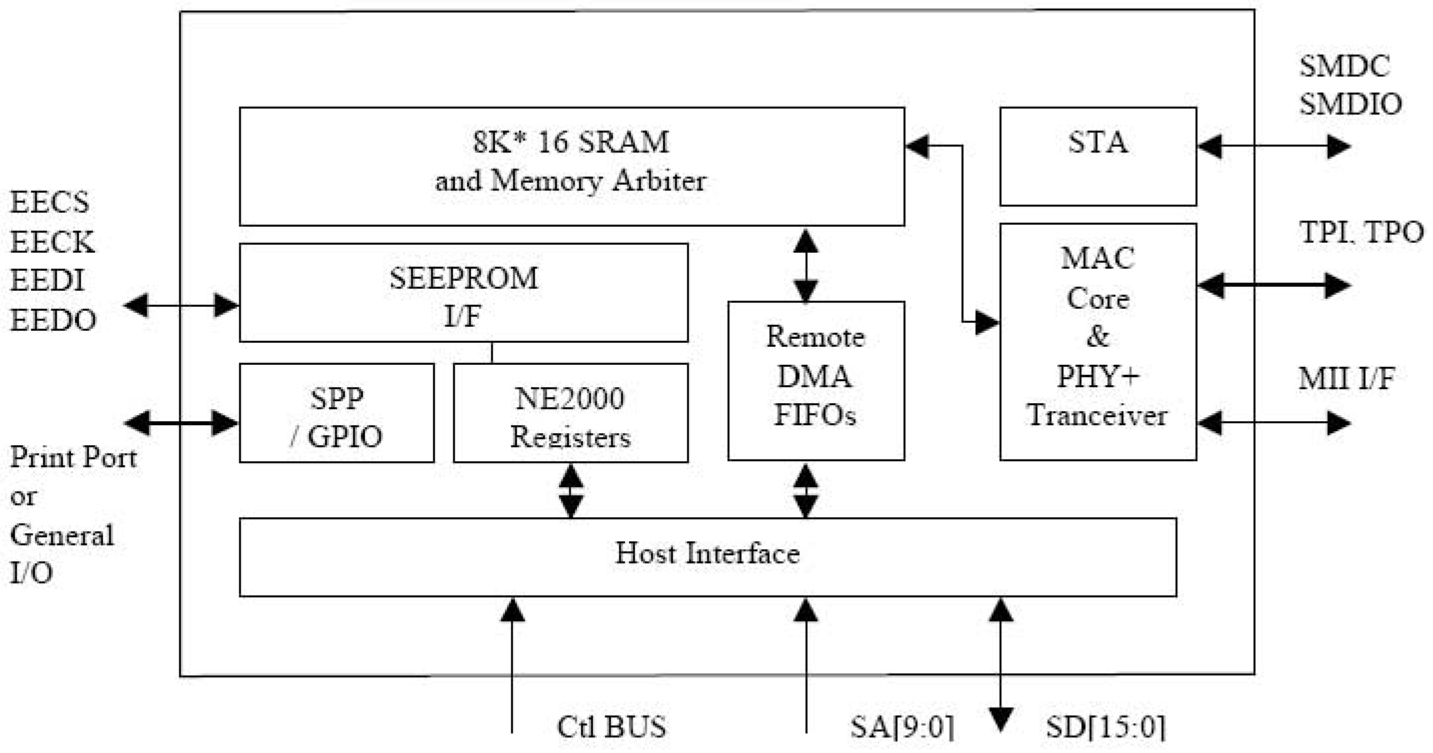
It is important to note that packet reception needs to be vigorously tested to ensure data's quality. Some testing will involve different packet sizes and high rates of transmission. In order for the data to no be corrupted during transmission, we will need to test the packet received at length.

During packet transmission, we will need to complete the following series of tasks:

- Starting the NIC state machine.
- Write Ethernet header and packet data into packet buffer.
- Set length of the packet, making sure that the length would be rounded up if less than 64 bytes.

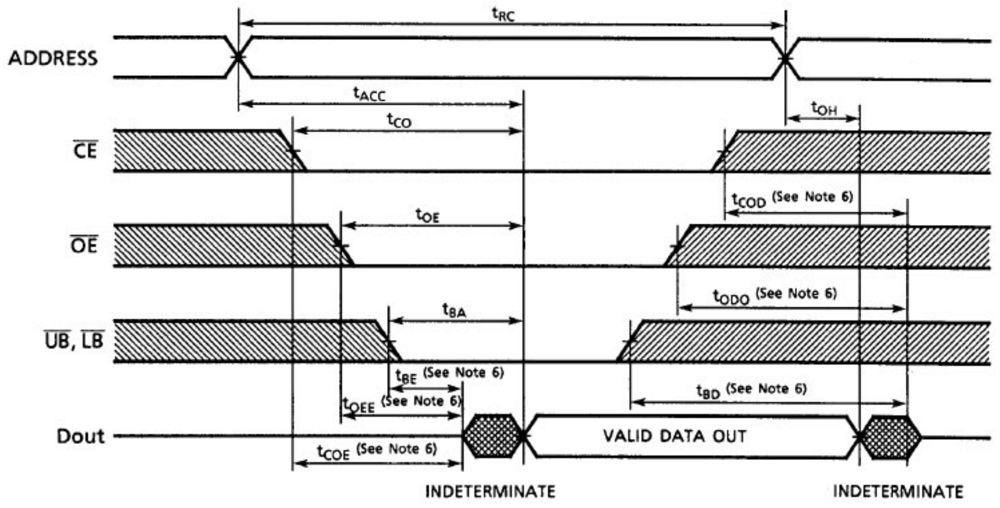
It is also important to note here that if a transmission fails, higher levels such as TCP will initiate a retry, while the low-level drivers do not. Also very important is the problem that there is a possibility that the get_ and put_ calls become mixed in the NIC buffer area. This can be solved by reprogramming the NIC DMA controller to continually read or write the packet buffer.

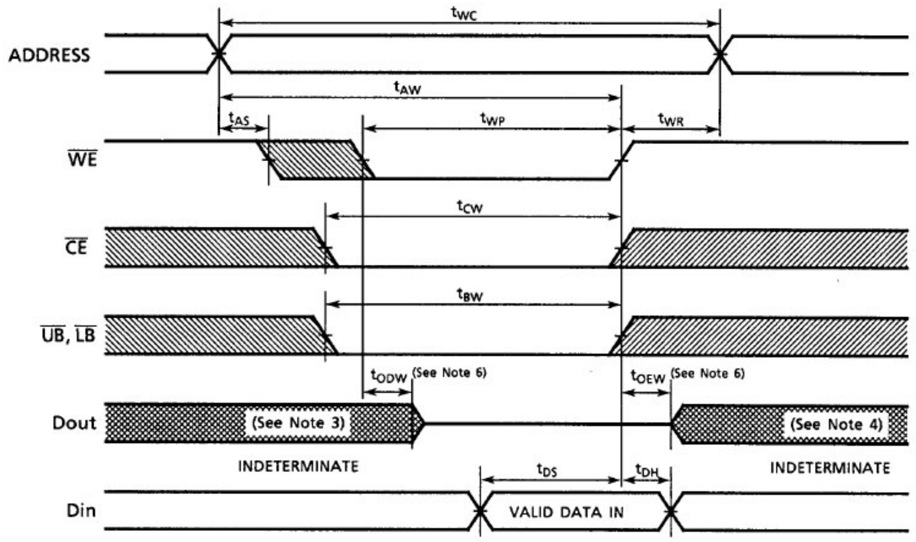
<u>AX88796 Block Diagram</u>

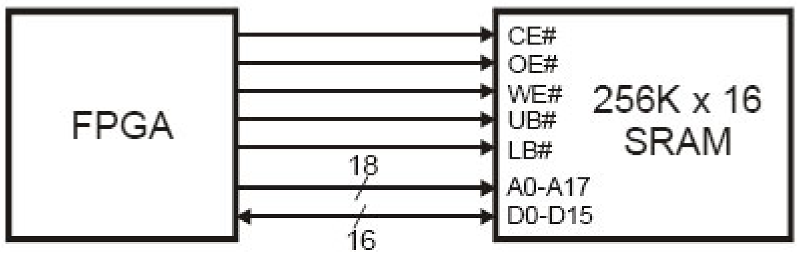


SRAM

We will be storing the data of our web server on the off-chip SRAM, which will be interfaced with the MicroBlaze through a custom peripheral which will handle the initialization write and the subsequent read accesses according to the protocols and interface defined below:



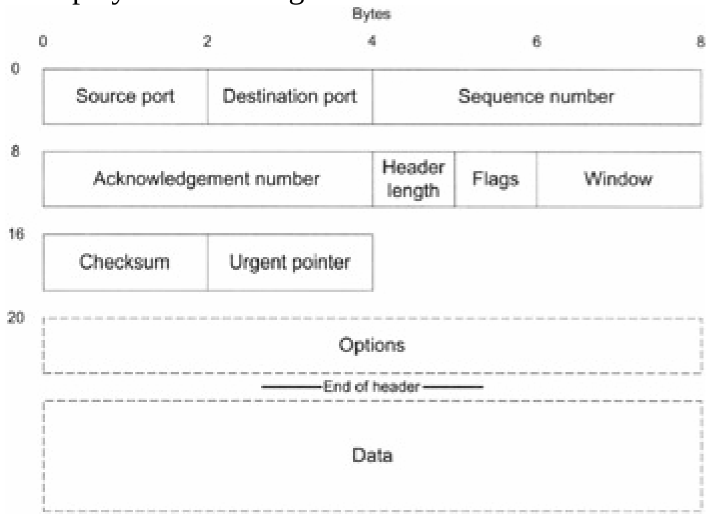




PROTOCOLS

TCP/IP provides a reliable connection between two sockets on a network. Parameters that define a socket are the IP address of the client and server and the port number. A web server will respond only to incoming requests on port 80. As soon as the clients HTTP request has been received, a TCP data segment will be sent out. A TCP segment consists of a header and the data block.

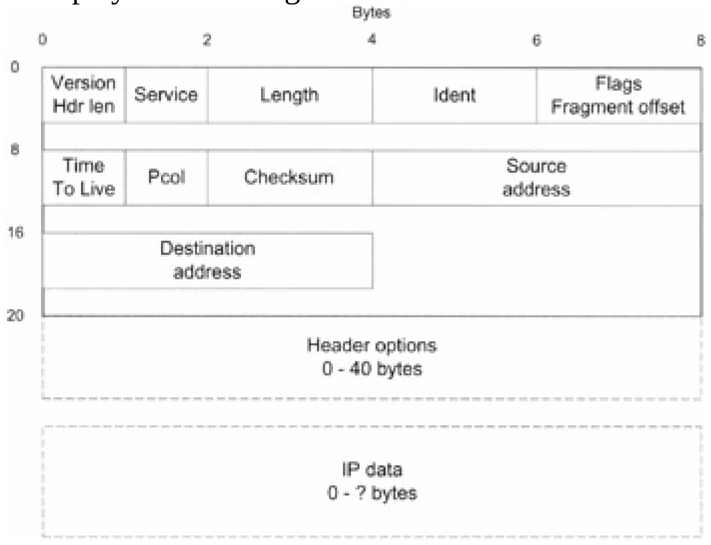
The following diagram displays the TCP segment format:



TCP headers are a minimum of 20 bytes.

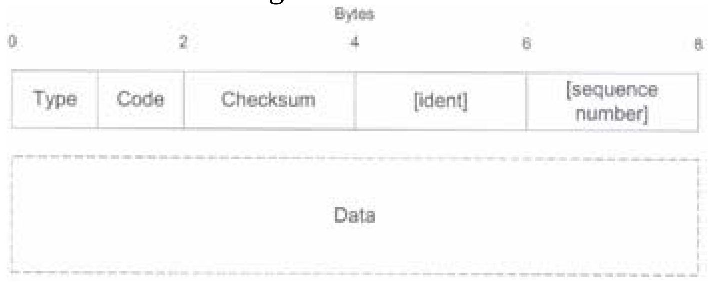
Internet Protocol (IP) is used to convey the TCP segments between hosts. An IP header plus the data block is known as the IP datagram.

The following diagram displays the IP datagram:



ICMP (Internet Control Message Protocol) is used for network diagnostics. An ICMP message is contained in the data field of an IP datagram. We will be using the ICMP Echo Request (Ping) to check the lower protocol layers. This is done even before we've implemented the web server.

The following diagram is an ICMP message:


Bytes
0
2
4
6
8
Type
Code
Checksum
[ident]
[sequence number]
Data