ECE 491 – Senior Design 1
Project Specification
Simplified Network Interface – WimpNet 2006

Last Update: November 1, 2006

1. About this Document

This document will serve as the working specification for the ECE 491project. As the project proceeds there will be a need to clarify and correct the specification as a normal part of the design process. The most up-to-date version of this document will be posted on the ECE 491 Website; major specification changes will be announced in lab.

2. General Information About the WimpNet 2006 Interface

WimpNet 2006 is a simplified version of the popular Ethernet network interface. It provides a protocol through which a number of *stations* can communicate over a shared medium, as shown in Figure 1. Data is transmitted from one station to another in *frames* that consist of a header that specifies the frame's destination address, source address, a body that contains that data that is intended for transmission, and a CRC error detection field.

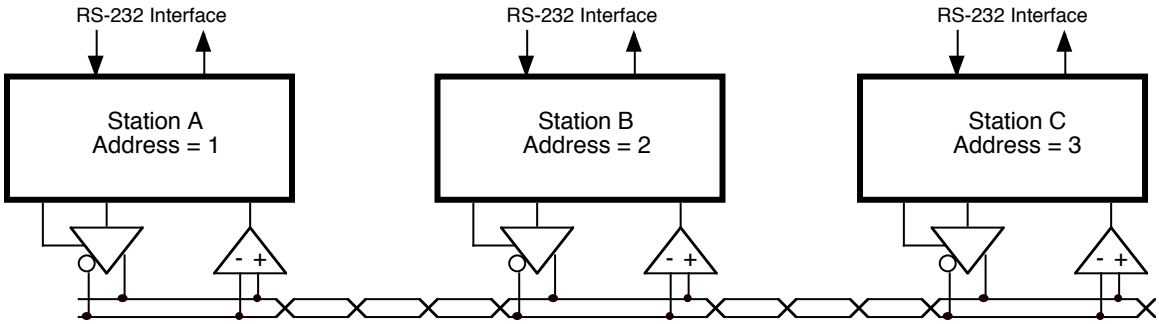

Figure 1 – Network Connections

2.1 Address

Each station connected to the network has a unique 8-bit **address** that is used to identify it to other stations on the network. This can be compiled into your FPGA design as an 80bit constant.

2.2 The Physical Interface

The physical medium for transmitting data will be a twisted pair of wires. Differential signaling over the twisted pair will be done using the RS-485 standard, which allows for "multi-drop" connections. Each station in the network is connected to the twisted pair using an RS-485 transceiver, which consists of a transmitter and receiver circuit. The transmitter has an enable input, a data input, and a differential output. When the enable input is

asserted, the transmitter translates the data input into a corresponding differential output that is driven on the twisted pair wires. When enable is low, the transmitter output is placed in a high impedance state so that other stations can drive the twisted pair. The receiver circuit translates the differential input into a digital output that can be read by the network receiver.

RS-485 transceiver chips will be provided in the lab along with documentation. You will need to connect these chips on the breadboard attached to the Spartan-3 board via the "A2" edge connector. So that the transceivers may be used interchangeably by more than one group, we will use the following input/output conventions:

TxD: FPGA Pin E6 / Connector Pin 4

TxE: FPGA Pin C5 / Connector Pin 6

RxD: FPGA Pin C6 / Connector Pin 8

2.3 Serial Transmission Protocol

Data frames will be transmitted serially using Manchester encoding. Your design should allow the data transmission rate to be configurable to any value between 9600 bits/second and 1 Million bits/second. You will use the Manchester transmitter and receiver circuits designed in Labs 5 and 6 as building blocks in your design.

2.4 Frame Format

Data is transmitted on the network in the form of a *frame* i.e., a package of serially transmitted bits containing a preamble / start frame delimiter (SFD), the address of the destination computer, the address of the sending computer, a sequence of data bytes, and finally a byte containing a Cyclic Redundancy Check (CRC) for error detection. Figure 2 shows the frame format. The destination address and source address fields are each one byte (eight bits) in length (real Ethernet uses 48-bit addresses). The CRC field is also one byte (real Ethernet uses 16 bits). Note that each byte is transmitted starting with the least significant bit.

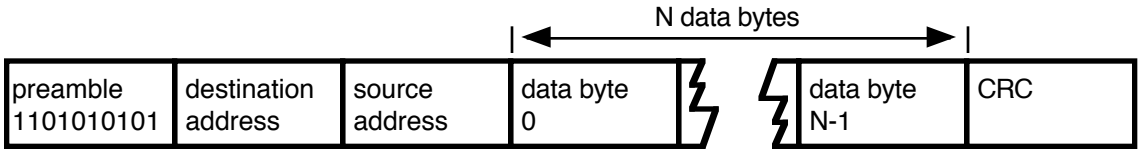

Figure 2 – Frame Format

2.5 Cyclic Redundancy Check (CRC) Code

The Cyclic Redundancy Check (CRC) code is used for error detection. It is calculated using an LFSR structure as the data bits are transmitted. After the last byte of the data for a frame is sent, the CRC byte is sent one byte at a time.

On the receiver side, the CRC is re-calculated using the data as it is received, *including* the CRC sent by the transmitter. A zero value in the final CRC indicates that no errors have been detected

We will use an 8-bit CRC that can be calculated using the circuit shown below (note that real Ethernet uses a 16-bit CRC):

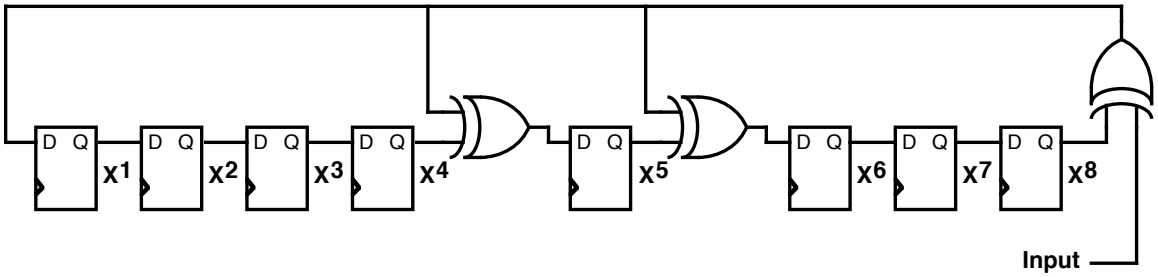

Figure 3 – 8-bit CRC Code

In normal operation, the CRC calculator is initialized to contain all 0's when the preamble is received. Next, on each successive clock cycle one bit of the frame is applied to the input as the flip-flops shift as shown in Figure 3. The CRC calculation is applied to the header and data fields of the frame. Note that the preamble and the CRC field itself are *not* included in the calculation

When transmitting the CRC field in the final byte of the frame, the CRC bits are transmitted starting with the rightmost bit shown in Figure 1 (i.e., bit X^8).

Your design should be configurable so that CRC checking can be turned on and off. When turned off, the transmitter should not send a CRC byte at the end of a frame, and the receiver should accept packets without checking for errors. When turned on, the transmitter should send a CRC byte at the end of a frame, and the receiver should check the CRC of the incoming frame.

3. Transmitter Specification

The transmitter is responsible for encoding and sending frames to another station. In our design, the frame contents (including the preamble, destination address, source address, length, and data) are downloaded from a PC using the RS-232 interface. All bytes in the frame must be downloaded before transmission begins. Sending the "control-d" signal indicates that all of the bytes in the frame have been sent, and transmission should now begin.

Figure 4 shows a block diagram of the transmitter, which includes your Manchester Transmitter, RS-232 receiver, a Block RAM that will act as a buffer to store the frame, and a control unit that you must design to implement the desired function of the transmitter.

To transmit, the transmitter first waits until the "cardet " input (from the Manchester Receiver) is not asserted, indicating that the ether is free. It should then start transmitting the frame from the buffer RAM, starting with address 0 and continuing until all bytes are transmitted unless a collision is detected.

3.1 Transmitter Operation - Normal

While not transmitting, the circuit should accept bytes from the RS-232 receiver and place them in the buffer RAM, starting at address 0. As mentioned earlier, the ^D (control-D)

character is used as an "end-of-frame" indicator. When ^D is received, the circuit should next prepare to transmit.

3.2 Transmitter Operation – Collision

A collision is detected when a is being transmitted and there is a mismatch between the outgoing data and the data that is received by the receiver. When this occurs, the circuit must immediately stop transmitting the frame.

Next, the circuit must *jam* the ether by asserting a logic low on txd for 8 bit periods. The purpose of jamming is to insure that all stations that are transmitting during the collision also detect a collision.

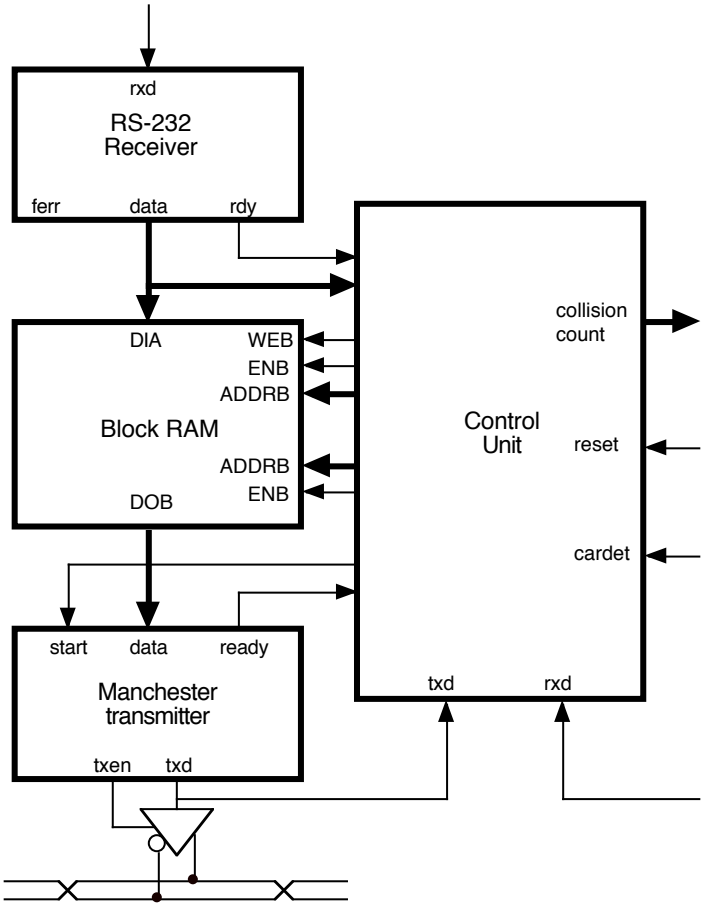

Figure 4 – Transmitter Organization

Backoff will be performed by waiting a random amount of time. For our purposes, we can approximate a random amount of time using a free-running 4-bit counter and waiting until that counter “rolls over”. Since two transmitters are unlikely to have free-running counters in the same state, two transmitters involved in a collision should delay by different amounts of time.

The "real" ethernet specification requires *exponential backoff*, so that if repeated collisions occur, the waiting time increases. In the interest of simplicity, we will not implement this feature in our design.

After backoff, the transmitter will attempt to retransmit the frame as described in the preceding section, starting by testing that the cardet input is false.

Your circuit should keep track of the number of collisions encountered since system reset and output this on the "collision count" port.

4. Receiver Specification

The receiver is responsible for reading frames from the ether and collecting all of the bytes of frames that match its network address. Once a complete frame has been received, it must then be transmitted to a PC one byte at a time using an RS-232 transmitter. Figure 5 shows a block diagram of the receiver circuit.

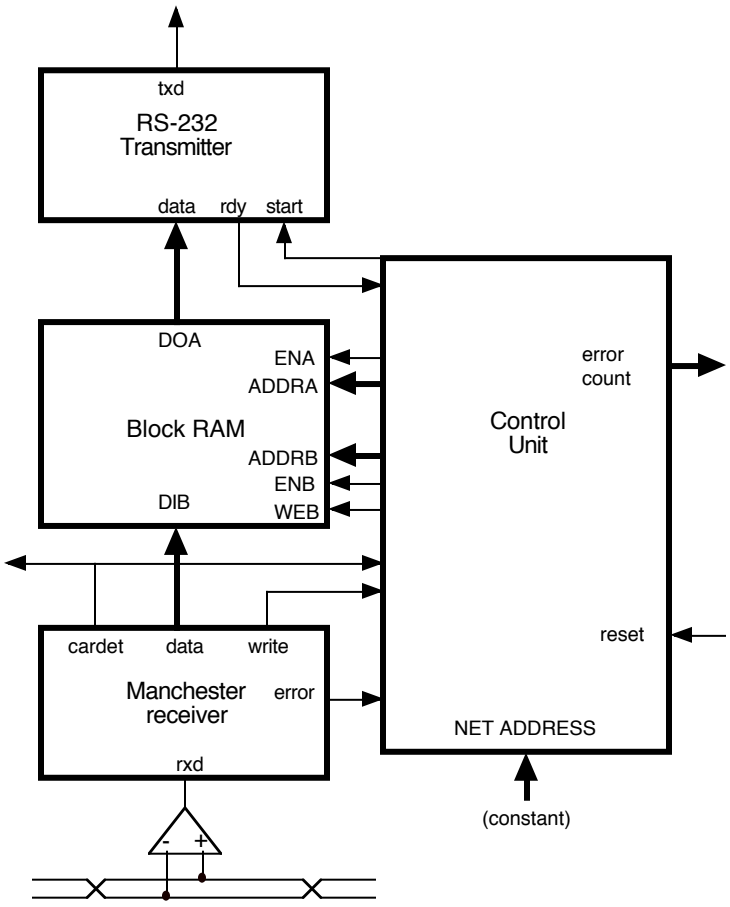

Figure 5 – Receiver Organization

During normal operation the receiver waits for an incoming frame, which will be converted from serial to parallel by the Manchester Receiver. When the first byte of a frame is available, the receiver must compare this value to the "Net Address" input. If these match,

the receiver should continue to accept bytes from the incoming frame and load these bytes into the buffer RAM. If there is not match, the rest of the frame should be ignored.

Errors may occur during transmission for a number of reasons, including noise, collisions, transmitter malfunctions, and (if implemented) CRC errors. When an error occurs, the transmitter should discard any portion of a frame it has received and wait for the network to go idle again before accepting another frame. The control circuit should keep track of the total number of errors encountered since system reset using the "error count" output port.

5. Design Constraints

Your project design will be subject to the following constraints. You must demonstrate that that these constraints are met in your final, working design.

Data Rate:	9.6kbps – 1Mbps
Data Jitter:	10% of the bit transmission period
Packet Length:	4 bytes – 256 bytes
Design Complexity:	Complete design must fit in a Xilinx XC3S200 FPGA.

6. Design Suggestions

- Use the simulator extensively and write lots of testbenches – debugging is easier with the simulator than with the actual hardware!
- Design the receiver module first & simulate/test with your Manchester transmitter and the "mxtest" module.
- Focus on getting the basic function (frame transmit and receive) working first before attempting to implement advanced features such as collision detection and CRC.
- Build your RS-485 interface last and test using the "telephone poles" only after you are certain that your design is working properly.

5. Deliverables

System Design – Due Monday Nov. 13

Block diagram of your overall system design.

Description of the function of each block in your diagram.

System Test Plan – a description of how you will verify that your system works properly and that all constraints are met. Should include a checklist that will be complted in the final report.

System Demonstration – Friday Dec. 8 (Rehearsal Thursday Dec. 7)

Demonstrate that your system can send and receive packets to other systems using the twisted pair network in the DSP lab.

Be able to explain the operation of your circuit to ECE faculty members.

Final Report – Due Monday, Dec. 11

Complete description of the design, including a Block Diagram.

"Theory of Operation" document that describes how the system works.

All Verilog source code. Include a header block in each Verilog file that describes the function of the code in that file.

Printouts of all simulations that have been used to verify the circuit.

A description of how you tested your circuit and verified that all constraints were met.

6. Grading Guidelines

System Design / Design Review	10%
Project functions as specified:	
Basic function: can send and receive packets	50%
Collision detected working and demonstrated	10%
CRC checking working and demonstrated	10%
Final Report	20%