ECE 491 – Senior Design 1
Project Specification
Simplified Network Interface – WimpNet 2007

Last Update: November 5, 2007

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 2007 Interface

WimpNet 2007 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 and source address, a *body* that contains that data that is intended for transmission, and a *checksum* field that uses a Cyclic Redundancy Check (CRC) code for error detection.

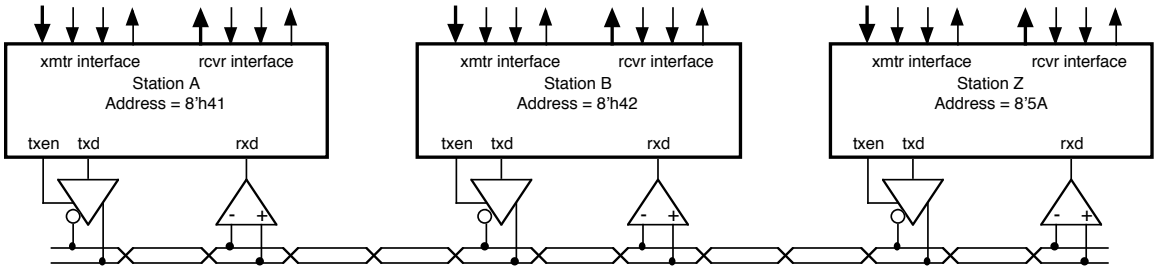

Figure 1 – Network Connections

2.1 Address

Each station connected to the network is assigned a unique 8-bit *MAC (Media Access Control) Address* that is used to identify it to other stations on the network.

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 10,000 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.

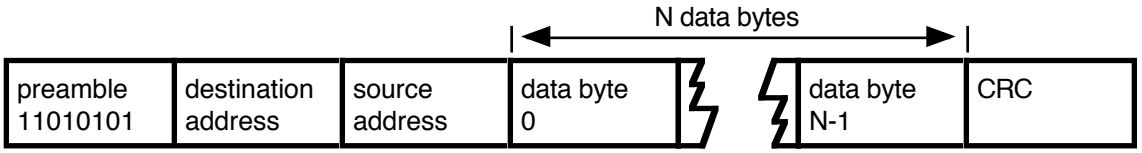

Figure 2 – Frame Format

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):

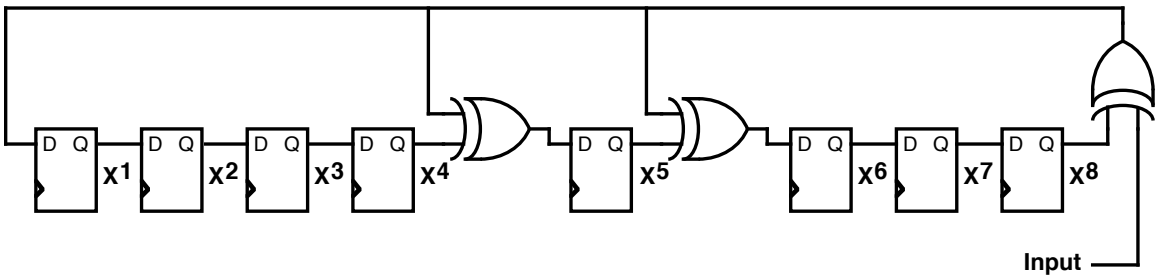

Figure 3 – 8-bit CRC Code

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 will 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 frames 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.Figure 4 shows the external of the transmitter. To set up a frame for transmission, the interface requires that parallel data for the frame be written into a buffer one byte at a time by asserting each byte on the XDATA input while asserting XWR true for one clock cycle. By convention, the first two bytes written are the destination address and source address, respectively. When all data has been loaded into the transmitter, asserting XSND true indicates that the frame is ready for transmission. At this point, the XRDY output is asserted low and remains low until the frame is completely transmitted.

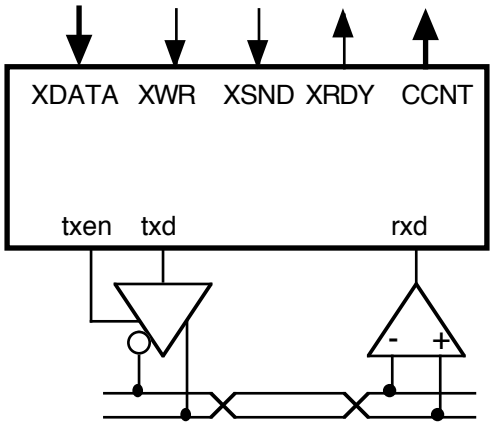

Figure 4 – Transmitter Interface

3.1 Transmitter Operation - Normal

Once XSND has been asserted, the transmitter monitors the shared network "ether" to make sure that no other station is transmitting. It then starts transmitting the frame from the buffer until either all data is transmitted (the normal mode of operation) or a collision has occurred.

3.2 Transmitter Operation – Collision

A collision is detected when a frame is being transmitted and there is a mismatch between the outgoing data and incoming data. 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.

After jamming, *backoff* is performed by waiting a random amount of time. For our purposes, we can approximate a random amount of time using a free-running 8-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 CCNT (collision count) port.

4. Receiver Specification

The receiver is responsible for reading frames of data from the ether and collecting all of the bytes of frames that match its network address. Complete frames are stored in a buffer that is read one byte at a time. Figure 5 shows the interface of the receiver circuit.

When a complete frame has been received, the circuit asserts the RRDY output. After RDONE is asserted, data is read from the receiver one byte at a time from the RDATA port while asserting the RRD output to indicate that the value has been read. The interface then makes the next byte available until all bytes in the frame have been read. At this point, it asserts RDONE to indicate that no more data is available. The CRC field is *not* included in the data read from the receiver.

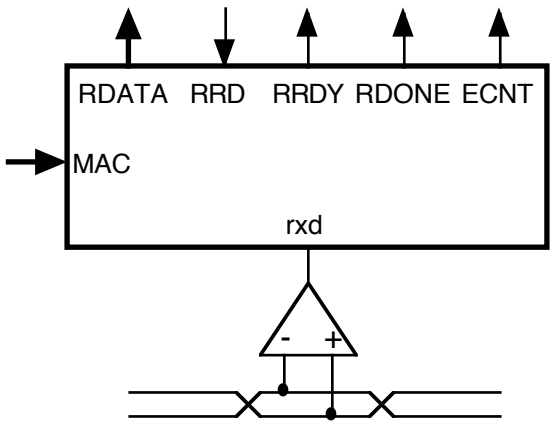

Figure 5 – Receiver Interface

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 MAC (MAC Address) input. If these match, the receiver should continue to accept bytes from the incoming frame and load these bytes into a buffer RAM. If there is no 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 receiver 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 ECNT (error count) output port.

5. Requirements

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

1. Frame transmisstion shall comply with the specification given in Section 2.
2. The transmitter interface shall comply with the specification given in Section 3.
3. The receiver interface shall comply with the specification given in Section 4.
4. Data signaling in the physical link shall comply with the RS-485 standard.
5. The **data rate** in the physical link shall range from 10Kbps and 500Kbps. The design shall be parameterized so that can easily be configured to operate at different bit rates (10Kbits per second default) given a 50Mhz clock input from the Spartan 3 Starter Kit board. Instructions for configuring your circuit at different bit rates must be included in header comment of your top-level Verilog file.
6. The system shall be tolerant to variations in transmission rate. Specifically, the system shall tolerate **data jitter** of ±10% of the transmission rate on each individual bit transmission.

7. **Frame length** shall range from a minimum of 4 bytes up to a maximum of 256 bytes (including address and CRC fields).
8. **Design implementation & cost:** The design will be prototyped using the Spartan 3 Starter Kit's Xilinx XC3S200 FPGA. However, the design must fit in a smaller XC3S50 FPGA for lower cost during production.
9. **MAC Address**: Your design must be assigned a unique MAC Address that is a printable ASCII character. Instructors will maintain a MAC Address "Registry" (i.e., a sign-up sheet in the lab) so that each group can reserve their unique address.
10. **MAC Address reconfiguration**: It must be possible for your design to operate with a different MAC Address when requested by the user. This change must be accomplished without reconfiguring your FPGA.
11. **Interoperability:** To be considered a successfully working design, your system must be able to transmit and receive frames to working systems designed by all other groups in this semester's class.
12. **Intellectual Property:** Your design must not infringe on any active patent which has been brought to your attention (specifically, the Nortel Manchester Decoder patent distributed in class).
13. **Safety**: Your transmitter unit must include a "fail-safe" unit that guarantees that it will not perform any continuous transmission longer than 2 times the maximum frame length. If this occurs, the transmitter must disable any further transmissions and indicate an error condition until system reset or reconfiguration.
14. **Environmental/Sustainability**: It must be possible to manufacture your design in compliance with the EU RoHS (Restriction of Hazardous Substances) Directive.
15. **Design entry** shall be done using Verilog (IEEE Standard 1364-2001). All Verilog code must follow the Coding Guidelines discussed in class.
16. **Verification (Simulation):** Your design must be thoroughly simulated to show that all design requirements are met. This includes (but is not limited to):
    a. Transmitter: Transmission of a normal frame.
    b. Transmitter: Collision detection, backoff, and retransmission of a frame.
    c. Transmitter: Inclusion of the appropriate CRC byte at the end of a frame.
    d. Receiver: Reception of a normal frame.
    e. Receiver: Incoming frame with mismatched address (ignored).
    f. Receiver: Incoming frame with bad CRC (ignored).
    g. Combined system: transmitter successfully sending frame to receiver.
17. **Verification (Hardware)**: Your design must be tested in hardware to show that requirements are met. Figure 6 shows the basic testing configuration, in which the transmitter and receiver are connected via an adapters to your asynchronous serial receiver and transmitter circuits, respectively. The complete circuit is then connected via a Serial cable to a PC running HyperTerminal. The Transmitter

Adapter must be constructed so that it accepts characters in a frame including source, destination, and an arbitrary number of data bits up until a "control-D" character is received. At this point, the transmitter should attempt to send the complete frame. The Receiver Adapter must be constructed so that when a complete frame has been received, it outputs the frame byte by byte on using the Asynchronous Serial Transmitter. You will use this testing configuration to demonstrate that your design can successfully send and receive frames to designs created by other lab groups.

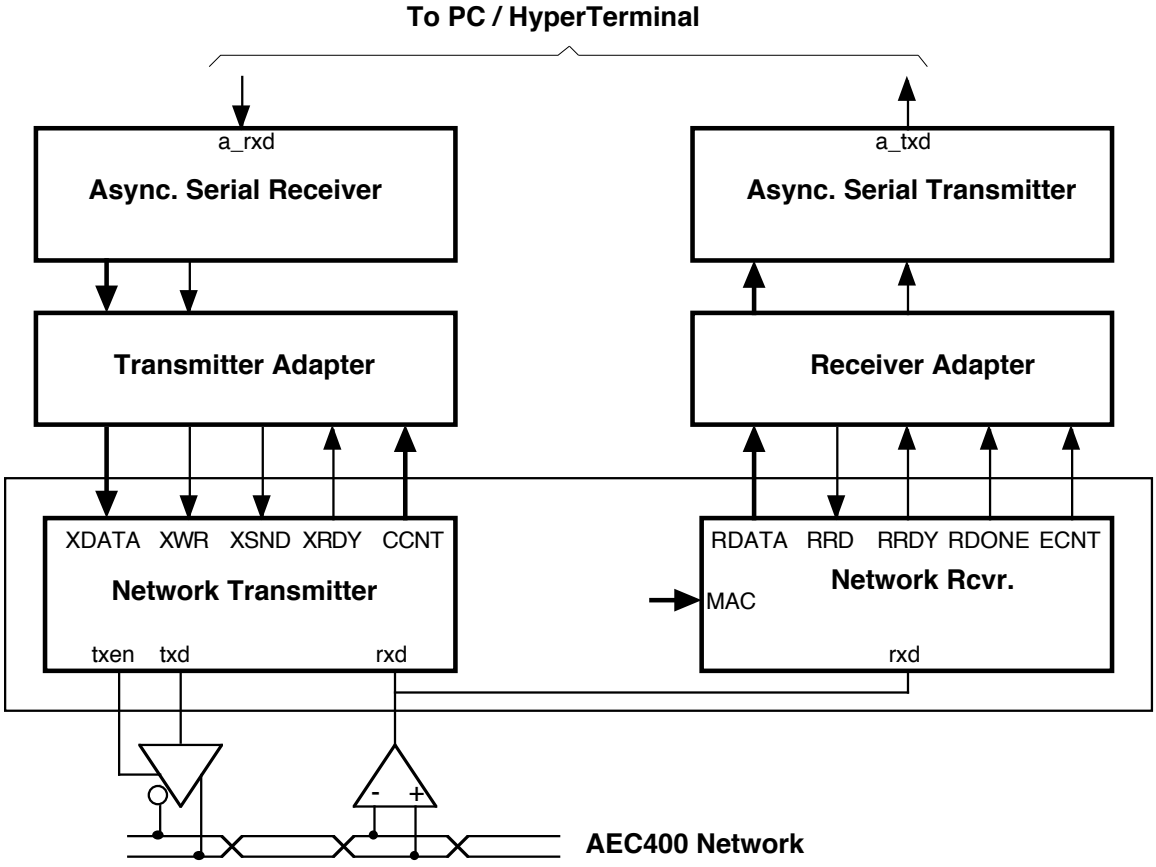

Figure 6 – Hardware Test Configuration

18. **Verification (Report)**: Your Final Report shall include a requirements checklist that summarizes how each requirement was tested and indicates the location of documentation of the test.

6. Deliverables

System Design – Due Friday Nov. 16 (Design Reviews Tuesday, Nov. 20)

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 draft Requirements Checklist that will be completed in the final report.

System Demonstration – Friday Dec. 7 (Rehearsal Wednesday Dec. 5)

Demonstrate that your system can send and receive frames 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. 10

Complete description of the design, including a Block Diagram and a discussion of how it works.

All Verilog source code.

Printouts of all simulations that have been used to verify the circuit, clearly labeled and cross referenced.

Requirements checklist: a summary of how each requirement was met, including references to simulation and hardware test documentation.

"Discussion" section that describes the implications of your design in terms of economic, social, political, safety, environmental, manufacturability, and sustainability constraints (more detail to be provided).

7. Grading Guidelines

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