ECE 491 – Senior Design 1
Laboratory 6 – Manchester Code Transmitter

Revised October 12, 2007

Goals

- To develop an FPGA design of a circuit that transmits data using the Manchester Code.
- To gain additional skill designing and debugging complex digital circuits using Verilog and FPGAs.

Background

The Manchester Code transmits data serially while guaranteeing that each transmitted bit contains either a rising or falling transmission in the center of the bit transmission period ("cell"). Figure 1 shows a common variation of the Manchester code, in which a "1" value is transmitted with a logic high in the first half of a cell and a logic low in the second half. A "0" value is transmitted with a logic low in the first half of the cell and a logic high in the second half. A third "idle" value is sometimes used when a transmitter has no data to transmit; this makes it possible for multiple transmitters to use a shared medium like the original Ethernet. In Figure 1, an "idle value" is transmitted as a logic high in both halves of the cell. The fourth possibility (a logic low in both halves of the cell) should not occur and is treated as an error.

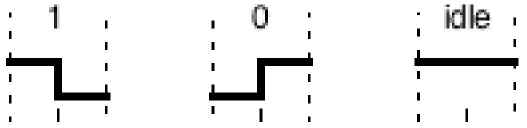

Figure 1 – Manchester Code Bit Values

The presence of a bit transition in each cell means that a receiver circuit can re-synchronize on each successive bit transition. This allows the transmission of a *frame* (i.e., a long string of bits). In contrast, asynchronous serial transmission schemes using a START and STOP bit can only be used to transmit short frames of data (e.g. 8 bits) because they only synchronize on the falling transition of the START bit.

Manchester transmitters usually begin a frame with a *preamble* that allows clock recovery circuits to synchronize to a starting transmission. The preamble is followed by a *Start Frame Delimiter* (SFD) which signals the beginning of actual data. In IEEE 802.3 Ethernet, the preamble is a 56-bit sequence of alternating 1's and 0's followed by an 8-bit SFD that transmits (LSB first) the value 11010101 (i.e., a sequence of alternating 1 and 0 values ending with two adajacent 1 values). Data is then transmitted bit-by-bit until the end of the frame is reached and the transmitter returns to the idle state. Figure 2 illustrates this process.

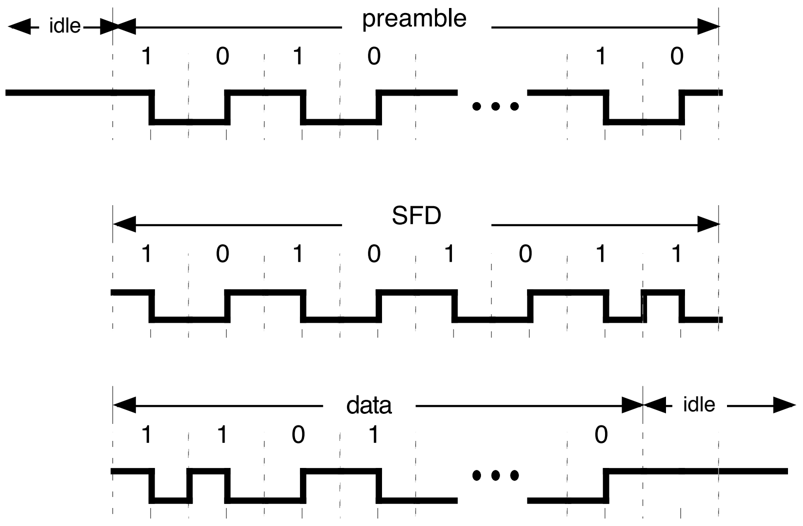

Figure 2 – Ethernet-Style Manchester Transmission

Requirements

1. Your design will implement the interface shown in Figure 3. The parallel data interface accepts data in 8-bit chunks. The transmitter converts these into serial Manchester bits on the "txd" output. The "txen" output is an *enable* output that will be used with a high-impedance buffer that drives a shared transmission line (called the "ether" in the Metcalfe and Boggs paper). Whenever the circuit is actively transmitting bits, the "txen" output should be high. At all other times, it should be low so that other units can transmit.

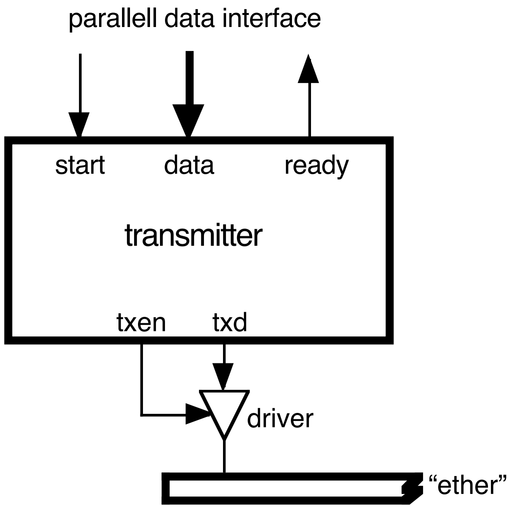

Figure 3 – Manchester Transmitter Interface

Figure 4 shows a timing diagram that illustrates the desired operation of the transmitter. When no data is being transmitted, the "ready" and "txd" outputs are asserted high and the "txen" enable output is asserted low. The user of the transmitter starts transmission by asserting an 8-bit data value on the "data" input and asserting the "start" input true.

After "start" is asserted, the transmitter responds by asserting "ready" low and begins to transmit the data bit-by-bit. When the transmitter *begins* to transmit the 8th bit, it asserts the "ready" output high again to indicate that it is ready to accept additional data for this frame. For transmission to continue, the user must respond by asserting "start" high and a new value on the "data" input before the end of the eighth bit. If this has been done, the transmitter loads the new data and continues transmitting bits as shown in Figure 4. When the transmitter begins transmitting the 8th bit of this second data word, it again asserts the "ready" output. If the user does not assert start at this time, then the frame ends at the end of the bit and the transmitter returns to the idle state and the "txen" enable output is asserted low.

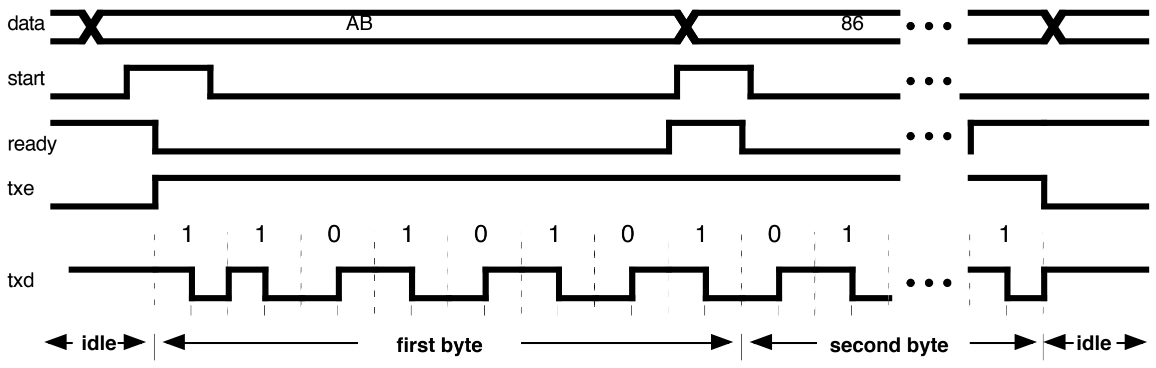

Figure 4 – Transmission of Two Bytes

2. The transmitter design must 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.
3. You must verify your design with one or more self-checking testbenches that simulate (at minimum) the following test cases:
    - Transmission of a single 8-bit value
    - Transmission of at least 3 bytes within the same frame. Verify that there are no “gaps” between the transmission of the 3 bytes.
4. You must verify the operation of the circuit in hardware by observing the “txen”and “txd” waveforms on an oscilloscope. To assist you in testing your transmitter, test circuit “mxtest.v” has been created which will generate inputs to exercise the transmitter. Connect this circuit to your transmitter circuit as shown in Figure 5.
5. All Verilog code must follow the Coding Guidelines discussed in class.

Deliverables

1. Demonstration to Lab Instructor of successfully operating design as it transmits characters from the “mxtest.v” test module and displays them on the oscilloscope.

2. A short technical memorandum which describes (a) what was done, (b) what you learned, and (c) what difficulties you encountered. Include a block diagram of your design showing its major components and state diagrams of any FSMs in your design.
3. Verilog listings of all files, including transmitter design and testbench.
4. Timing diagram printouts showing correct simulation of your transmitter design, including all required test cases.
5. Entries in your Lab Notebook documenting design ideas, the steps taken to create and debug your design, any pitfalls you encountered, and recorded data (if any). Each Lab Notebook entry should be dated and signed by all students in the lab group.

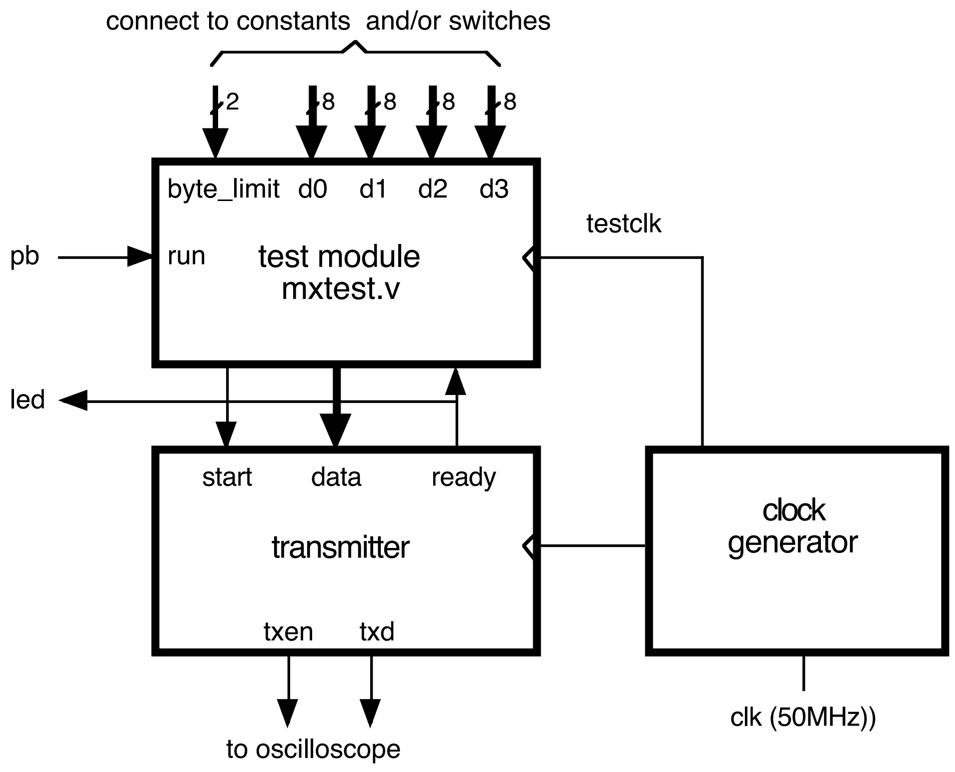

Figure 5 – Hardware Test Configuration