XILINX®

XAPP339 (v1.3) October 1, 2002

Manchester Encoder-Decoder for Xilinx CPLDs

Summary

This application note provides a functional description of VHDL and Verilog source code for a Manchester Encoder Decoder. The reasons to use Manchester code are discussed. The code can be compiled into either the Xilinx XC9572, XCR3064XL, or XC2C64 CPLD. To obtain the VHDL (or Verilog) source code described in this document, go to section **VHDL (or Verilog) Code Download**, page 6 for instructions.

Introduction

Manchester code is defined, and the advantages relative to Non-Return to Zero code are given. Target applications of Manchester code are discussed. Verilog and VHDL implementations of the Manchester Encoder-Decoder are available from the Xilinx website. The decoder and encoder are simulated using Verilog and VHDL testbenches. The encoder-decoder function is given in the context of more familiar serial communication circuits as UARTs, with the intent to illustrate some of the issues in designing serial communication functions in CPLDs.

NRZ and Manchester code defined

Non-return to Zero (NRZ) and Manchester codes are used in digital systems to represent the binary values "1" and "0". Figure 1 defines how NRZ and Manchester code represent binary values. NRZ is the code most often used. In NRZ, a logic "1" is represented as a high level throughout a data cell, and a logic "0" is represented by a low level. Manchester code represents binary values by a transition rather than a level. The transition occurs at mid-bit, with a low-to-high transition used to represent a logic "0", and a high-to-low to represent a logic "1". Depending on the data pattern, there may be a transition at the cell boundary (beginning/end). A pattern of consecutive "1s" or "0s" results in a transition on the cell boundary. When the data pattern alternates between "1" and "0", there is no transition on the cell boundary.

In NRZ, only one level/data cell is required, while in Manchester, two levels are required. A DC component exist in NRZ when contiguous "1s" or contiguous "0s" are transmitted. When the data pattern alternates between "1s" and "0s", the frequency response is equal to 1/ 2 the clock rate.The frequency response for NRZ then ranges from DC to clock/2. The frequency response of Manchester code ranges from clock/2, occurring when the data pattern is alternating "1s" and "0s", to clock, which occurs when the data pattern is consecutive "1s" or "0s". The frequency response of Manchester is a single octave vs. 5-10 octaves for NRZ.

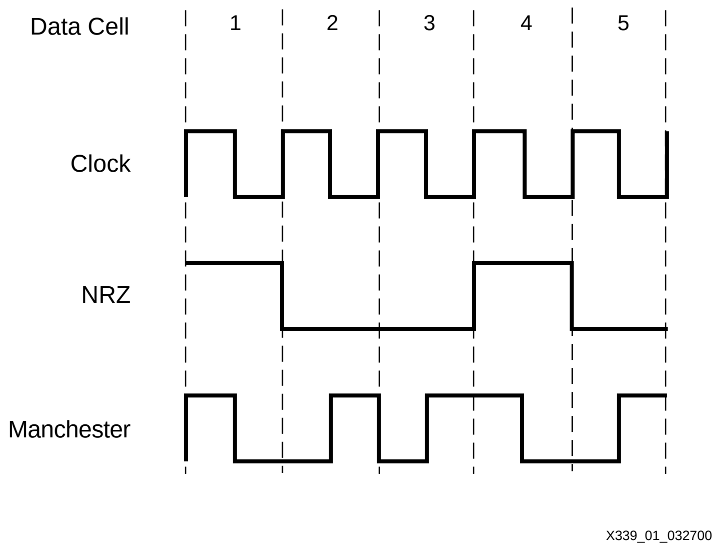

Figure 1: **Binary Values for NRZ and Manchester Codes**

Relative Advantages of NRZ/Manchester Code

Two advantages of NRZ are that it does not require encoding/decoding, and it makes the most efficient use of a communication channels bandwidth. Manchester requires a modulation rate twice that of NRZ to transmit the same amount of information. This can be important in bandwidth limited communication channels. On the other hand, the receiver of NRZ requires a true DC response. Since, Manchester code has no DC component, it can be transformer coupled.

The mid-bit transition in Manchester code provides a self-clocking feature of code. This can be used to improve synchronization over non-self clocking codes as NRZ. The transition also allows additional error detection to be done with relatively little circuitry.

Synchronization

In serial communication, clocks are used to define the size/boundary of a data cell. With a non-self clocking code, since the clock and data are distinct, there can be skew between clock and data. In magnetic media applications, skew may be due to variations in the tape drive speed. In serial communication, skew results from differences in the transit delay between clock and data lines on long serial links.

One design objective in serial communication is to decode data correctly in the presence of noise, or an otherwise degraded signal. Signals have non-zero rise and fall times during which the signal value is indeterminate. In a receiver, sampling, or the decoding of the value of a signal, should occur as far from the signal transition as possible. Sampling at the time furthest from the signal transition is known as center sampling.

A UART is a serial communication circuit which uses NRZ code. To sample at mid-bit of the data cell, the receiver in a UARTs uses a local clock which is 16X the received data rate. The data format of a UART consists of a high idle state, and a character format consisting of a start bit, five to eight data bits, optional parity, and one or more stop bits. After detecting the edge of a start bit, the receiver validates the start bit by counting the 16X clock to 8 and verifying that it is still Low. Subsequent center samples are reached by counting the 16X clock to 16.

In a Manchester decoder, center sampling occurs at points 1/4 and 3/4 through the cell, since transitions occur always at mid-bit and sometimes on the cell boundary. In addition to center sampling, the receiver in a Manchester decoder does clock recovery. Since Manchester has transitions at least once each data cell, the receiver has known references to which it can resynchronize at each bit.

To synchronize to an incoming serial data stream, the receiving ciruitry in a Manchester decoder can use a digital phase lock loop or a counter algorithm. Digital phase lock loops are most often used in networks with a ring topology while counter algorithms are common in point to point links. An example of a counter algorithm which uses a 16x clock is:

1. After receiving the initial transition on manchester data in, count the 16x clock to 4 and sample. The count of 4 is known as end count. At this time, end count is 1/4 through the data cell.
2. Reset the counter to 0. Begin counting the 16x clock with an end count of 8, and sample. If there is a transition on manchester data, reset the counter and go to (1).

When initialized correctly to the manchester data, this algorithm causes the counter to use an end count equal to 4 when consecutive "1s" or "0s" are transmitted, and an end count equal to 8 when alternating "1s" and "0s".

In summary, UARTs synchronize on a character basis while MEDs synchronize on a bit basis. In a UART, the timing jitter for the each bit in the character is cumulative until the end of the character. MEDs resynchronize at each bit, or at least once each bit.

Error Detection

The most commonly used error detection schemes in serial communication are parity and cyclic redundancy check codes. When Manchester code is used, a small amount of additional circuitry can detect bit errors. Figure 2 illustrates how the mid-bit transitions of Manchester code allow error detection. Figure 2 shows four rows of transmitted data, with the first row the valid Manchester representation for a logic "1". The three lower rows are waveforms of a corrupted form of the first row, and are erroneously transmitted data. When Manchester data is shifted in serially into a shift register in the decoder, an exclusive OR can monitor for different values on each side of the data cell (since there is a mid-bit transition) With this error detection, an error is undetected only if each half of a data cell transitions from its original state.

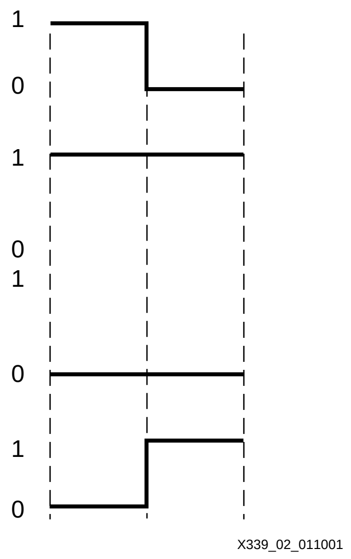

Figure 2: **Manchester Code Mid-bit Transitions**

Manchester Encoder/ Decoder Functionality

The functions of the encoder section of the MED include a microprocessor interface, parallel to serial conversion, frame generation, and NRZ to Manchester encoding. This circuitry can run very fast since it does not require a high-frequency clock. The frame format used is similar to that of a UART.

The Manchester decoder limits the maximum frequency of operation of the MED, since it uses a high-frequency clock. The receiver circuitry is more complex, since clock recovery and center sampling is done. Additional receiver functions are frame detection, decoding of Manchester to NRZ, serial to parallel conversion, and a microprocessor interface.

Manchester Decoder

The **mdi1** and **mdi2** registers (Table 1) have decoding circuitry which detects an incoming edge on **mdi** and activates the **clk1x_enable**. The **clk1x_enable** signal is used as a clock enable for various registers clocked by the **clk1x** clock. This significantly reduces power consumption when data is not being received. The **no_bits_rcvd** keeps track of the number of bits received and sequences the decoder through its operations. To vary the word size, change the value of **no_bits_rcvd**. The decoder does clock recovery using the first variable, **clkdiv** register, and sample signal.The Manchester data is decoded at the sample signal.

Table 1: **Manchester Decoder Pinout Functionality**

Signal	Direction	Function
rst	Input	Resets clk1x, no_bits_rcvd, nrz registers
clk16x	Input	16x clock input used as reference for clock recovery, center sampling
mdi	Input	Serial manchester data input
dout[7:0]	Output	Parallel NRZ data bus out
mdi1, mdi2	Internal	Internal registers used to detect edge on mdi
no_bits_rcvd	Internal	Controls word size and sequences decoder through operations
clk1x_enable	Internal	Enables 1x clock upon receipt of a word
clk1x	Internal	Internal 1x clock
sample	Internal	Determines time at which the receiver is to decode data
first	Internal	Variable used by the counter to determine end count
data_ready	Output	Status signal indicating data is present on the data bus dout.
rdn	Input	Control signal initiates a read operation

The following should be verified from running a simulation.

1. The **clk1x_enable** signal is High upon receipt of data on **mdi**, and Low during the idle state of the line.
2. NRZ is all "0s" for first byte, all "1s" second byte, alternating data third byte (not visible)
3. A read operation on **rdn** resets the data_ready status signal.

The simulation waveform shows the center sampling done by the manchester decoder shows the sample pulse occurring midway between the transitions on the **mdi2** signal.

Manchester Encoder

Table 2 defines the Manchester encoder pinout functionality.

The following should be verified in a simulation.

1. The pattern on manchester data out (**mdo**) when input data is consecutive "1s" (ff), and again when input data is alternating "1s" and "0s" (aa).
2. The ready status signal goes High after word is transmitted, and is reset by a strobe on **wr**.

Table 2: **Manchester Encoder Pinout Functionality**

Pin/signal	Direction	Function
din[7:0]	Input	Input data bus
clk	Input	Clock
wr	Input	Control signal input used to strobe data into the buffer register through **din[7:0]**
mdo	Output	Serial Manchester data out
rst	Input	Resets **mdi1**, **mdi2**, **data[7:0]**, **clk1**, **ready** registers
ready	Output	Status signal indicating the encoder can accept data.

Other Manchester Encoder Decoder Functions

The functions discussed in the remainder of this application note can be used in addition to or in place of those described earlier. They include:

- Sync pulse and manchester error detection
- Adding a buffer register to the decoder

Sync pulse and manchester error detection

The design above uses a frame format similar to that of a UART. Additional noise immunity can be provided by substituting a 3-bit wide sync pulse for the 1-bit start bit. The sync pulse used in this section is 1-1/2 bits at one level followed by 1-1/2 bits at the opposite level.

The reason for allowing either polarity (High/Low or Low/High) is that it provides a very efficient method for the transmitter and receiver to distinguish between a command and data. Since the 3-bit wide pattern is a relatively long pattern without a valid Manchester, and one which is unlikely to occur randomly, the decoder is unlikely to start erroneously.

When this pattern is detected, a 10-bit register md is used. The Manchester data in (**mdi**) input is routed in serially into the 10-bit **md** register, which is clocked by a 2X clock clk2x. There is pattern detection circuitry monitoring **md[9:0]**. Since **md** is clocked by clk2x, the detection circuitry is actually continuously monitoring 5-bit data patterns. In the scheme given below, the pattern detection asserts the **sync_ pulse** signal if the 3-bit wide sync pulse is followed by two valid Manchester bits. The pulse on **sync_pulse** can trigger the decode operation.

VHDL (or Verilog) Code Download

VHDL (or Verilog) source code and test benches are available for this design. THE DESIGN IS PROVIDED TO YOU "AS IS". XILINX MAKES AND YOU RECEIVE NO WARRANTIES OR CONDITIONS, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, AND XILINX SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR A PARTICULAR PURPOSE. This design has not been verified on hardware (as opposed to simulations), and it should be used only as an example design, not as a fully functional core. XILINX does not warrant the performance, functionality, or operation of this Design will meet your requirements, or that the operation of the Design will be uninterrupted or error free, or that defects in the Design will be corrected. Furthermore, XILINX does not warrant or make any representations regarding use or the results of the use of the Design in terms of correctness, accuracy, reliability or otherwise.

XAPP339 - http://www.xilinx.com/products/xaw/coolvhdlq.htm

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
03/27/00	1.0	Initial Xilinx release.
04/17/00	1.1	Added VHDL Code Download link.
01/10/01	1.2	Changed Figure 2 levels 3 and 4.
10/10/02	1.3	Minor revisions