Control Link Serial Interface

November 2010 Reference Design RD1051

Introduction

In today's highly-integrated systems, noise reduction is a high priority for circuit board designers. Serially transmitted data with an embedded clock allows a significant reduction in data traces and eliminates the need to run a clock trace on the circuit board. This minimizes the potential crosstalk on the board and improves signal integrity for the entire system.

Manchester encoding is a method used to combine data and a clock to form a single self-synchronizing data stream, while Manchester decoding is to retrieve the clock and data from the serially received data stream. In order to receive the data correctly without using an accompanying clock signal, the clock and data recovery (CDR) function must be performed on the received serial data. The Phase Locked Loop (PLL) together with oversampling is a technique often used to perform the clock data recovery.

The Differential Manchester code is an alternative to the standard Manchester code. Both have their advantages and are being used in different application areas. One of the common benefits is that the DC component level of the encoded data stream is zero. Compared to standard Manchester code, Differential Manchester code will operate in the same manner if the signal is inverted.

This reference design provides an example of how to implement a low-speed serial control link using Differential Manchester code. It takes advantage of the no-chip PLL to oversample of the incoming serial data stream. The oversampling technique is used for this application since control links usually run at a lower speed than the data path. This, together with the characteristics of the Differential Manchester code, makes the extraction of the data and the clock information from the serial data possible.

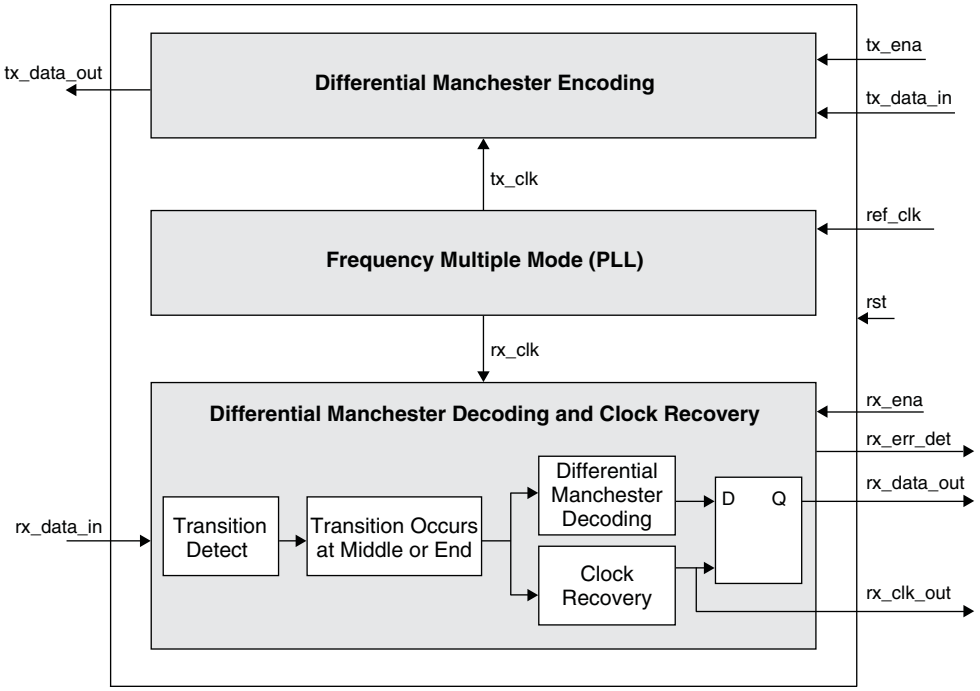
Features

- Differential Manchester encoding/decoding
- Serial data in (receive channel)/serial data out (transmit channel)
- No need for parity insertion and checking
- Clock recovery based on the oversampling technique
- No need for preamble due to the differential characteristic of the encoded signal

Functional Description

Figure 1 shows a block diagram of the different functions implemented in this reference design along with the input/output signals.

Figure 1. Block Diagram



Signal Descriptions

Table 1. Signal Descriptions

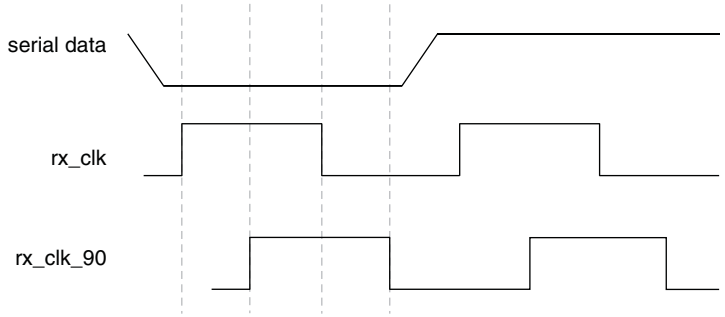
Signal Name	Signal Direction	Active State	Definition
ref_clk	Input	N/A	The reference clock input to frequency multiple module, used to generate rx_clk and tx_clk.
rst	Input	High	The active high reset for all designs.
Differential Manchester Encoding			
tx_ena	Input	High	When active and the lock signal of PLL is high, the encoder starts to encode.
tx_data_in	Input	N/A	The serial data input which will be encoded.
tx_data_out	Output	N/A	The Differential Manchester code.
Differential Manchester Decoding			
rx_ena	Input	High	When active and the lock signal of PLL is high, the decoder starts to decode.
rx_data_in	Input	N/A	The serial Differential Manchester code input.
rx_err_det	Output	High	When active, this signal indicates the recovered data is incorrect.
rx_clk_out	Output	N/A	Clock recovered from Differential Manchester code input.
rx_data_out	Output	N/A	Data recovered from Differential Manchester code input.

Frequency Multiplication Module

The Differential Manchester encoded data requires two logic levels to represent each binary bit, so the data rate after encoding is two times the incoming data rate. Based on this, the Differential Manchester encoding requires a clock with a frequency twice of the input serial data rate.

In order to oversample the serial data stream on the receiving side, each bit of the incoming data is sampled four times using the two clocks that are 90 degrees out of phase. Figure 2 shows the four-time oversampling concept used in this design.

Figure 2. Oversampling Concept



The frequency multiplication module makes use of the on-chip PLL to generate clocks rx_clk and rx_clk_90 based on a local reference clock. The frequency of clock rx_clk and rx_clk_90 is equivalent to the incoming serial data rate. Each code bit sees four clock edges which in turn provide four sampled data points. This is equivalent to being sampled by a clock four times faster than the incoming data rate.

The PLL module can be generated through the ispLEVER® development tool.

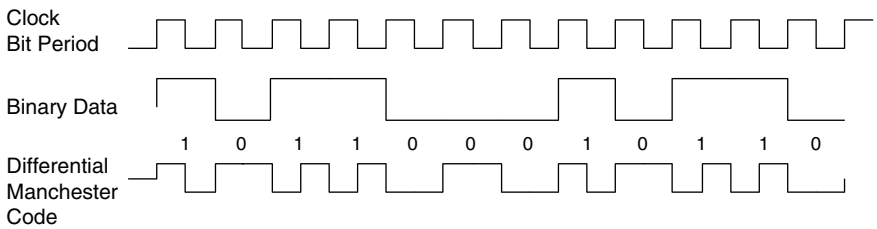
Differential Manchester Encoding

The Differential Manchester encoding requires a single clock with a frequency twice the input serial data rate. In this reference design, Differential Manchester encoding is defined as follows:

- Each bit is transmitted in a fixed time (the “period” of the reference clock).
- In the beginning of the bit-time there is always a transition, whether from high to low, or low to high.
- A '0' bit is indicated by making the first half of the signal equal to the second half of the signal; that is, no transition at the middle of the bit-time.
- A '1' bit is indicated by making the first half of the signal the opposite of the second half of the signal; that is, a transition occurs at the middle of the bit-time.

Figure 3 shows how Differential Manchester code in this reference design represents binary values.

Figure 3. Example of Differential Manchester Encoding



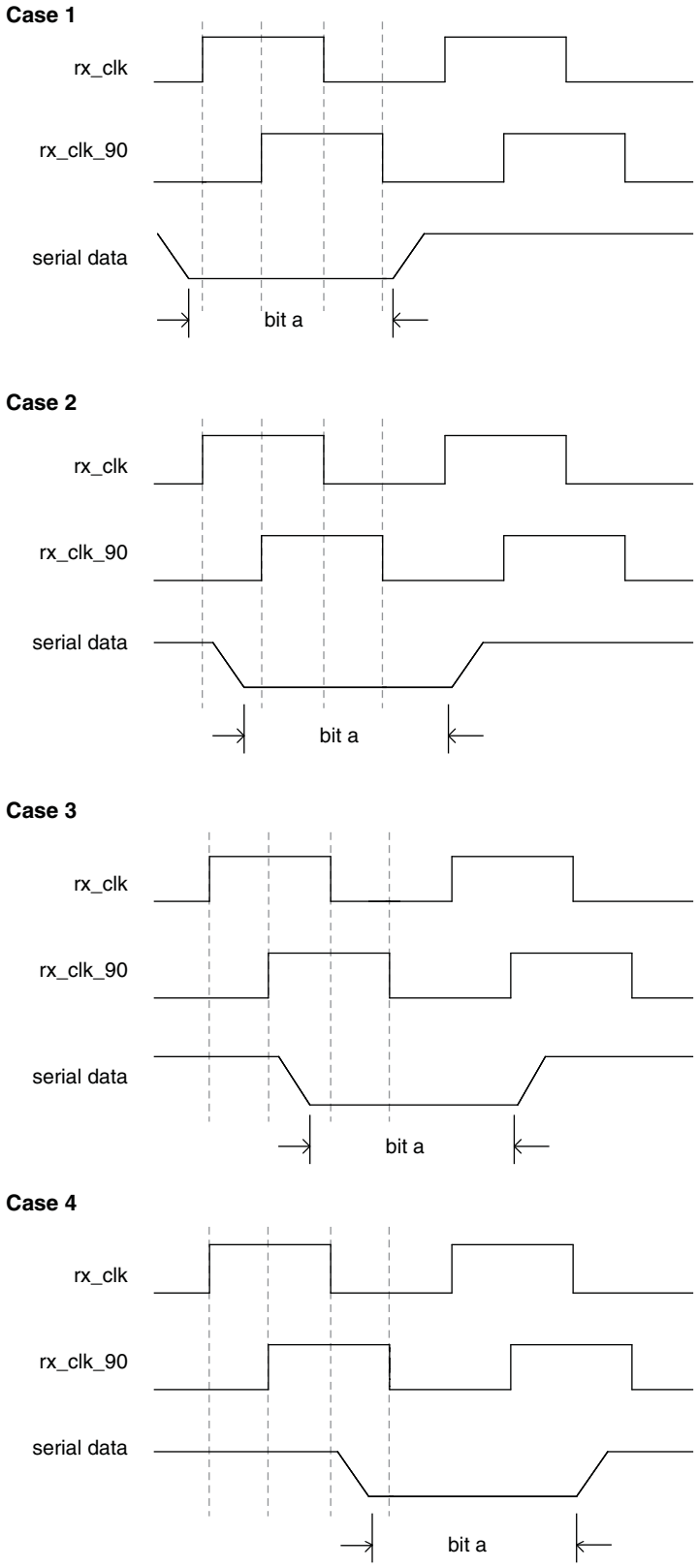
Differential Manchester Decoding and Clock Recovery

This module requires two clocks with a 90-degree phase difference running at the same frequency as the input Differential Manchester code rate. A good sample of the incoming data is when both the setup and hold times of the sampling clock can be met. This usually happens when the middle of the bit is sampled. Depending on the arrival of the data bit and the clock phase, the edges that sample the middle of the bit may vary. Figure 4 summarizes the four scenarios that use different clock edges to get the best sample. Using case 2 as an example, the rising edge of rx_clk_90 sees the data bit “a” first. As a result, the data point sampled by the falling edge of rx_clk or the falling edge of rx_clk_90 could be considered as the best representation of the data bit “a”. This theory, together with the

characteristics of Differential Manchester code, provides accurate sampling and recovery of each data bit. With differential Manchester code, bit “1” can be sampled and recovered when transition occurs at the middle of the code, and bit “0” can be sampled and recovered when the transition occurs at the end of the code.

Since the frequency of rx_clk is equal to the input data code rate, the recovered clock frequency is one-half of the rx_clk frequency. The phase information of the clock is embedded in the transitions of the input data stream. As a result, the phase of the recovered clock is detected when the transition occurs.

Figure 4. Oversampling Data Using Different Clock Edges



Test Bench Description

The test bench for this design includes a random binary signal generator (prbs_gen.v) which generates the random binary signal used for encoding and decoding. During the simulation, the bench creates two files: original_data.txt and recovery_data.txt in the testbench subdirectory. The first file is to record the original data which are used for encoding and the second file is to record the recovered data clocked by the recovered clock.

Timing Diagrams

The following timing diagrams show the major timing milestones in the simulation.

Figure 5. Differential Manchester Encoding

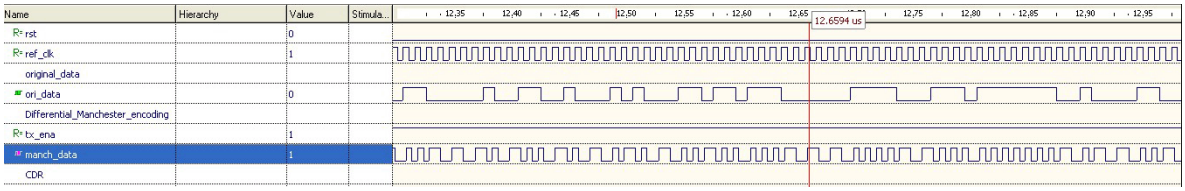
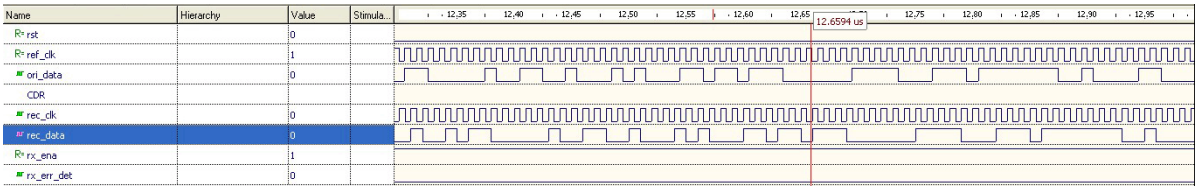


Figure 6. Differential Manchester Decoding and Clock Recovery



Implementation

Table 2. Performance and Resource Utilization

Device Family	Language	Speed Grade	Utilization (LUTs)	f_{MAX} (MHz)	I/Os (LVDS)	Architecture Resources
MachXO2™ [1]	Verilog	-6	26	rx_clk > 200	10+2	1 PLL
	VHDL	-6	26	rx_clk > 200	10+2	1 PLL
MachXO™ [2]	Verilog	-3	25	rx_clk > 200	10+2	1 PLL
	VHDL	-3	25	rx_clk > 200	10+2	1 PLL
LatticeXP2™ [3]	Verilog	-5	26	rx_clk > 200	10+2	1 PLL
	VHDL	-5	26	rx_clk > 200	10+2	1 PLL

1. Performance and utilization characteristics are generated using LCMXO2-1200HC-6TG100CES, with Lattice Diamond 1.1 and ispLEVER 8.1 SP1 software. When using this design in a different device, density, speed or grade, performance and utilization may vary.
2. Performance and utilization characteristics are generated using LCMXO1200E-3T100C, with Lattice Diamond™ 1.1 and ispLEVER® 8.1 SP1 software. When using this design in a different device, density, speed or grade, performance and utilization may vary.
3. Performance and utilization characteristics are generated using LFXP2-5E-5FT256C, with Lattice Diamond 1.1 and ispLEVER 8.1 SP1 software. When using this design in a different device, density, speed or grade, performance and utilization may vary.

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)
+1-503-268-8001 (Outside North America)
e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com

Revision History

Date	Version	Change Summary
May 2009	01.0	Initial release.
August 2009	01.1	Document title changed from “Control Plane Serial Interface” to “Control Link Serial Interface.”
December 2009	01.2	Added VHDL support.
January 2010	01.3	Added support for LatticeXP2 device family.
November 2010	01.4	Added support for MachXO2 device family and Lattice Diamond design software.