FPGA Asynchronous Communication

1. Summary

This document outlines the sheme that will be used to transfer data between the FPGAs in the MCE. To implement the physical layer of communications we plan to use a serial, asynchronous scheme that will allow receivers to self-align to a transmitted bit stream by 8X over-sampling of the data rate. A receiver will align itself with the transmitter at the beginning of every byte, using a start bit. The proposed scheme has been implemented, but has yet to be integrated into existing FPGA firmware. The scheme supports a data rate of 2.5 MB/s (at a clock rate of 25MHz), and is robust in noisy environments.

2. Introduction

The SCUBA-2 multi-channel electronics consists of a number of different cards, each fitted with an Altera Stratix FPGA (except for the power card). The FPGAs communicate with each other using multi-dropped and point-to-point LVDS data lines across the Bus Backplane. Originally, a synchronous method was to be used, but because of synchronization issues, an asynchronous method was chosen instead. The asynchronous transmission method allows all of the FPGA clocks to skew from each other because the receiver can recover the clock from the data stream. Also, the asynchronous method embeds synchronization signals into the bit stream, at the penalty of using 2 extra bits for every transmission word.

The following sections outline the asynchronous communication methods used and the reasons why it was chosen.

3. Requirements

The requirements of the FPGA serial link are:

1) It must be possible to sample the serial data based on the transmission clock.
2) The receiver must be able to synchronize itself to the transmitter.
3) It should be capable of at least 2.5MB/s sustained transmission data rate.
4) It receivers should filter out noisy signals during transmission.

4. Previous Communication Methods

Several communications methods were considered before the serial asynchronous method was chosen. A shared, parallel bus was considered and rejected because of the complex arbitration needed to split usage between 10 FPGAs. Individual parallel buses from the Clock Card to each FPGA were considered and rejected because of the number of I/O required. Thus, serial point-to-point links between the Clock Card and each card were chosen.

Also at issue was a choice between synchronous and asynchronous protocols. Initially, a synchronous protocol was considered. To implement it, the transmitter was to provide a serial clock and data to the receiver for each serial link. That way the receiver would know exactly when to sample the data. The issues with this method were:

1) The receiver could not distinguish which bit was the start of a word without an additional framing signal from the transmitter.
2) Altera's SERDES hardware in the FPGAs also wasn't appropriate for this application.

However, this method was capable of operating at high frequencies.

5. Current Communication Scheme

Currently, the Clock Card distributes Clock, Command and Synchronization signals to every other card using multi-tapped LVDS lines. Each card initiates return communication with the Clock Card using two unshared point-to-point LVDS lines. Both the multi-tapped and point-to-point lines will use the asynchronous serial communication scheme.

The asynchronous serial communication scheme proposed is based on RS232 and had been adapted for LVDS signals. Figure 1 shows an example waveform (Bit 1 is the MSB).

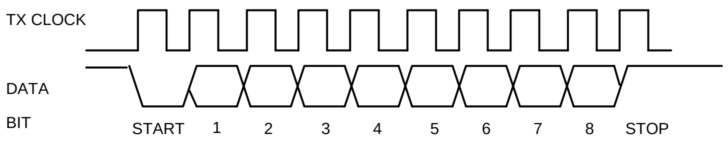

Figure 1: Serial Communication Waveform

The idle state of the transmitter is high. A low transition signals a start bit to the receiver and allows the receiver to synchronize its sample clock to the transmitter. The receiver then calculates when to sample the transmitted data (ideally on the falling edge of the transmitter clock). After 8 bits are transmitted, the transmitter brings the line high to signal that the transmission is complete.

The receiver works by sampling the transmitted data at 8 times the transmit clock rate. This is necessary because the transmit clock isn't transmitted to the receiver. On the first falling edge that the receiver sees, it synchronizes itself to the next closest sample clock. It then counts 3 clocks to sample the start bit. All of the other bits are sampled every 8 clocks thereafter.

While the receiver is receiving, its clock will deviate from the transmit clock. However, because the length of the transmitted word is short, the clock doesn't deviate far enough to get out of sync with the transmit clock. The error is nullified on each new byte received because the sample clock gets resynchronized.

The main issue with this asynchronous method is that it requires more logic to receive signals than a synchronous method, limiting the upper frequency. However, it is felt that it is possible to achieve the required data speed with enough margin.