Design and Test of a Universal Asynchronous Receiver and Transmitter Module

Brandon Boesch, Josh Randall, James Schulman
Professor Jacob Abraham
EE 382M.7/460R - VLSI1
14 October 2015

***Abstract*—This report deals with a small team of students' design and test of a Universal Asynchronous Receiver and Transmitter (UART) module. The purpose of this project is to utilize skills acquired in the classroom in order to design a module that is personally interesting to the team. UART was chosen because it is a common method of communication between electronics that do not share a clock. UART communication is achieved by breaking down parallel data from a transmitting device, and sending the data serially to a receiving device. The receiving device collects the serial data, and in turn, converts it back into parallel data. This project was a rewarding experience because it provided the students with an opportunity to design and test within a team environment, which is a common practice in industry.**

***Index Terms*—Asynchronous, baud rate, communication protocols, receivers, transmitters, UART**

I. SPECIFICATIONS

When designing this project, our first step was to determine our operating specifications. Below we highlight all relevant specifications used in our project:

- Provide asynchronous communications, both read and transmit, between two systems.
- Has the ability to convert data from serial to parallel, and from parallel to serial, using shift registers.
- An on-chip baud rate generator is included to control transmit and receive data rate.
- Two on-chip 4 byte FIFO buffers will be used for incoming and outgoing data; this gives the host system more time to respond to an interrupt generated by the UART, without loss of data.
- Two interrupt signals are provided to the host system. One of these signals lets the system know when the transmit FIFO is full, to prevent loss of data. The other interrupt lets the system know when the Receive FIFO is empty, so that only valid data is fetched.
- The operating voltage will be 5V, and will have a max baud rate of 1.5Mbps. The area of the design should fit within a 40PDIP package.

II. DESIGN

A. *Signal Descriptions*

The signals received and generated by the module will be those included in the figure 1 below:

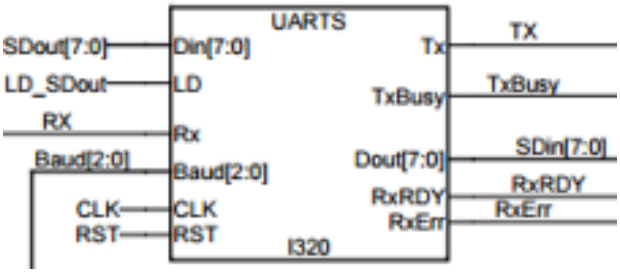

Fig. 1. UART Module

Our signal definition is as follows:

- Host Interface:
  - D_out[7:0] - The parallel transmit data lines
  - D_in[7:0] - The parallel receive data lines
  - LD_D_out - Load the next byte into the transmit FIFO
  - LD_D_in - Load the next byte from the receive FIFO
  - CLK - Host clock to be divided
  - RST - Chip reset (Active High)
  - Baud[2:0] - 3 bits from processor to select baud rate
  - RX_empty - No rx data in FIFO
  - TX_full - No more room for tx data
- Serial Interface:
  - TX - Serial transmit data line
  - RX - Serial receive data line

B. *Baud Rate Selector*

One of the inputs to our UART device is a three bit baud rate selection input. This input determines the rate of communication, and the internal clock rate that our device uses. The baud rate generation module of our device takes the 3 bit baud rate input, and the value of this signal is latched into a

register before communication begins. Then, this module divides the input clock and uses the selection register to select the appropriate clk divider out of the 8 possible dividers to drive the rest of our device. **We have not yet decided which possible clock divisors we will use.**

C. Transmitting Modules

The transmit FIFO is an 8-bit wide, 4-location deep, first-in-first-out memory buffer. Data written to the UART module will be stored in the buffer until it is read out by the transmit logic. If this FIFO is full, it will assert the TX_full interrupt signal high, and it should not accept any additional data while it is full. When the FIFO is no longer full, it should lower the TX_full signal. The transmit logic successively reads words from the transmit FIFO when appropriate and performs parallel to serial conversion on the word; sending the serial data stream synchronized to TX_clk via the TX pin.

D. Receiving modules

The receive FIFO is an 8-bit wide, 16-location deep, first-in-first-out memory buffer. Receive data from the serial interface is stored in the buffer until read out by the processor. If this FIFO is full, it should generate the RX_full signal, and refuse to accept any additional data. The RX_clk signal is provided by the processor and used to synchronize its reception sequences. Receive logic performs serial to parallel conversion on the incoming synchronous RX data stream, extracting and storing values into the receive FIFO, to be read subsequently by the processor.

III. USER DOCUMENT

To be filled in.

IV. TESTING

Our design was verified with Verilog testbench simulations. We verified the timing of the baud rates, the integrity of both our serial and parallel data streams and the functionality of both the architecture as a whole and each individual module. We used Verilog delay operators to create appropriate timing for our simulated signals.

V. OPTIMIZATION

One notable optimization we implemented was FIFO bypassing. For transmission, this case occurs when the processor requests a transmit, the TX FIFO is empty and the transmit module is idle. In this state, we send the transmit data directly to the transmit module instead of storing it in the FIFO and then sending from the FIFO to the transmit module. The analog state for the receive hardware is when we receive a byte, the RX FIFO is empty and the processor is idly requesting a read. In this case we bypass the receive FIFO and send the receivre byte directly to the processor.