2018

UART Transmit Engine Chip Specification

Jose Sotelo
CECS 460
March 20, 2018

Prepared by:	Date:	Revision:
Jose Sotelo	March 19, 2018	1

Table of Contents

1. Introduction

The Universal Asynchronous Receiver Transmitter (UART) is a very common useful full duplex serial communication protocol. It serves as a basis for many ubiquitous protocols such as RS-232 (COM ports in computers). Its basis is a two-wire communication on which one port transmits and the other receives. The basic data units transferred is 1-byte. Data is transferred from the data bus to the transmitting UART in parallel from. Data the is transmitted is organized into packets. Each packet contains 1 start bit, 5-9 data bits, an optional parity bit and 1 or 2 stop bits. The transferring speed of the data depends on the baud rate. The baud rate is a unit of measurement of bits per second (bps). Once the baud rate is selected, the data packet will be transferred to the receiving pin.

During the transmitting stage, the UART gets the parallel data from the data bus, it adds a start bit, a parity bit, and a stop bit. The data put together is output serially, bit by bit at the Tx pin. The receiving UART read the data bit by bit at its Rx pin. Once the data has been received it's then converted back into parallel form and removes the start bit, parity but and stop bits. Lastly, the receiving UART transfers the data back in parallel to the data bus on the receiving end.

1.1 Purpose

For this project, we will be designing an SOPC (System on Programmable Chip) the Transmit Engine of the UART were the user can configure to transfer 8 or 7 bits, a parity bit, odd or even bit and the baud rate. We will interface the Transmit Engine with the 16-bit TramelBlaze to send data to transmit characters to a serial terminal RealTerm. The serial terminal will receive the data and output onto the screen CSULB CECS 460 [line count] <CR><LF>.

2. Applicable Documents:

2.1 External Documents

PicoBlaze 8-bit Microcontroller User's Guide

TramelBlaze

Programming TramelBlaze

3. Requirements:

3.1 Performance Requirements

The Universal Asynchronous Receiver will be able to transmit and receive data via the following baud rates: 300, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400, 460800 and 921600.

3.1.2 Interface Requirements

The Transmit Engine, TramelBlaze and the serial terminal RealTerm will be used to transmit and receive data.

3.1.4 Physical Requirements

The highest baud rate the Transmit Engine and the serial terminal can communicate with each other without breaking the data being sent is 460800.

3.1.5 Power Requirements

Supply	Circuits	Device	Current (max/typical)
3.3V	FPGA I/O, USB ports, Clocks, RAM I/O, Ethernet, SD slot, Sensors, Flash	IC17: ADP2118	3A/0.1 to 1.5A
1.0V	FPGA Core	IC22: ADP2118	3A/ 0.2 to 1.3A
1.8V	FPGA Auxiliary and Ram	IC23: ADP2138	800mA/ 0.05 to 0.15A

Table 2. Nexys 4 Power Supplies

3.1.6 Environmental Requirements

4. Implementation

4.1.1 Design Description

This project first implements the Transmit Engine for the UART that has a selectable baud rate, the number of bits transmitted, parity enable/disable and odd/even parity bits. For the Transmit Engine to function, the designer needs to implement SR flip flops, an 8-bit loadable register, one-bit register, the parity bit decoder, bit time counter, bit counter and shift register. Once the Transmit Engine is completed, we interface the Tx Engine with the 16-bit TramelBlaze microcontroller.

The first block inside the Tx Engine, is a SR flip flop that is used to set the TxRdy signal high at reset. The second SR flop goes low at reset and both SR flops are used to maintain stable outputs after the inputs are turned off. The 8-bit loadable register that is used to load the data in when the load input pin goes high and outputs 8 bits of data. The 8 bits of data are then used to determine how many bits are needed to produce a transmission of data. The user determines what type of protocol to send out. The are 7 options to transmit data from the inputs eight, pen (parity enable), and ohel (odd and even bits): 7N1, 7E1, 7O1, 8N1, 8E1, and 8O1. The truth table is used to produce the combo logic circuit that produces the output bits 10 and 9.

Eight	Parity Enable	Odd and Even	Bit 10	Bit 9
0	0	0	1	1
0	0	1	1	1
0	1	0	1	EP
0	1	1	1	OP
1	0	0	1	D7
1	0	1	1	D7
1	1	0	EP	D7
1	1	1	OP	D7

At the heart of the Transmit Engine, is a 11-bit shift register that is used to shift the transmitted data one bit of a time. Once the parity decoder determines bit 10 and bit 9, a one-bit register goes high, it tells the shift register to load the data to bit-10, bit-9, bits 8-2 and bit-1 goes high and bit-0 goes low. If the reset is pressed, the shift register outputs 11-bits 1's. The output of the shift register gets the LSB 0 of the data being shifted.

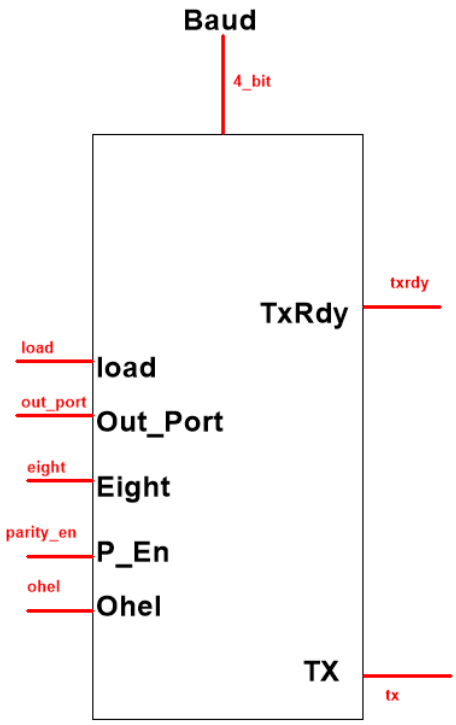
The design must also include two counters: 1) Count clocks to determine the bit time and 2) Count the bits to know when we are done. These two counters will let us know when we have waited a bit time and to know when all the bits have been transmitted. Lastly, a Baud Decoder circuit is created to generate the number of clocks that are necessary to fill one-bit time. Combining all the modules describe above, produces the Transmit Engine.

4.1.2 Top Level Description

The Transmit Engine block is used to interface with the TramelBlaze. The TramelBlaze contains the data that will be used by the Transmit Engine to transmit the data to the serial terminal.

4.1.3 Block Diagram

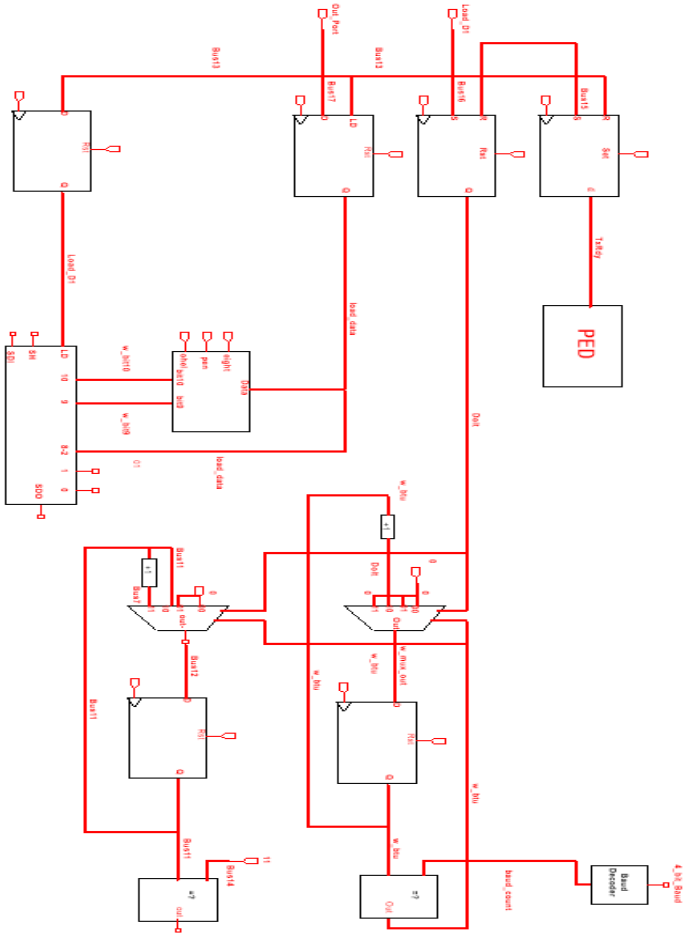
Top level diagram of the Transmit Engine.



Block diagram of the Transmit Engine

On Next Page

Prepared by: Jose Sotelo	Date: March 19, 2018	Revision: 1



4.1.4 Input/Output Interface Description

Inputs

- Baud – 4-bit input used to select the baud rate
  Pin Assignment: R13(SW8), U18(SW7), T18(SW6), R17(SW5)
- Load – 1-bit input used to load data
- Data_In – 8-bit input that contains the data to be transmitted
- Eight – 1-bit input to select [7:0] bits or [6:0] bits
  Pin Assignment: R15(SW4)
- Parity_En – 1: odd parity, 0: no parity
  Pin Assignment: M13(SW3)
- OHEL – 1: odd parity, 0: even parity
  Pin Assignment: L16(SW2)

Output

- TrxRdy – High Active signal that gets feed into an SR flop telling the TramelBlaze that its transmitting data.
- Tx- Data that's being shifted out from Transmit Engine

4.1.5 Clocks

All blocks in the design used one clock that runs at a frequency of 100MHz. This mean that the clock has a period of 10 nanoseconds.

4.1.6 Resets

All the designs used a synchronous reset to prevent metastability.

5. Externally Acquired Blocks

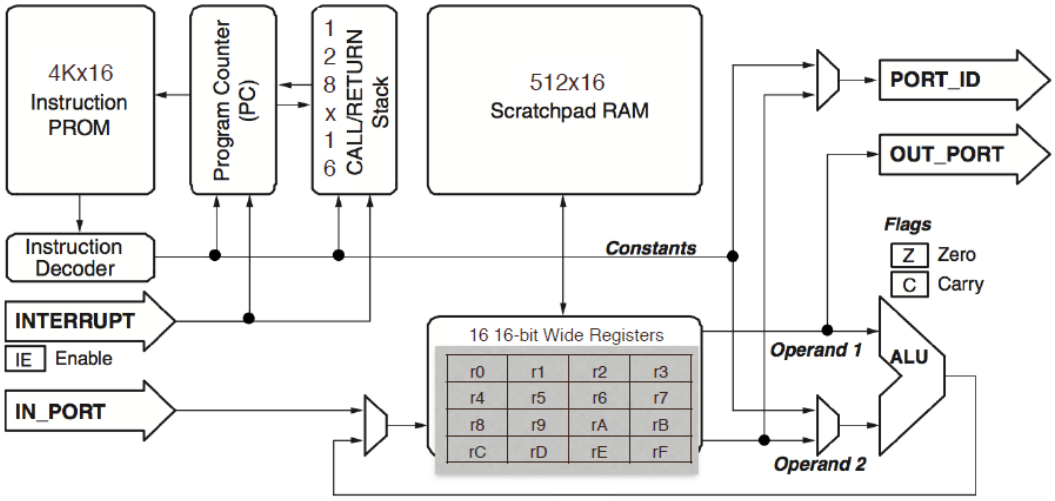
5.1 Block Name

The TramelBlaze is a 16-bit embedded microcontroller designed to emulate the 8-bit PicoBlaze.

5.1.1 Description

The TramelBlaze is a 16-bit processor core designed to emulate the Xilinx PicoBlaze. The instruction set is the PicoBlaze instruction set. It contains three memories Instructions Rom, Call/Return Stack and Scratchpad RAM. The TramelBlaze's I/O are a 16-bit In_Port, Interrupt, Reset, Clk, a 16-bit Out_Port, 16-bit Port_Id output, Read_Strobe output, Write_Strobe output and interrupt acknowledge output. When implementing the TramelBlaze into your design, you must generate the tb_rom and load the coe file into RAM. A Python assembler is used to generate assembly code. We will be using assembly code to output to the serial terminal and keep track of the line count.

5.1.2 Block Diagram



The TramelBlaze MCU block diagram

5.1.3 I/O Definitions

The I/O ports extend the TramelBlaze MCU's capabilities and allow the MCU to connect custom peripherals or to FPGA. It contains a 16-bit In_Port input that is used to write to the scratchpad memory. It allows to use 16 16-byte-wide registers. It also handles the interrupt input used to let the MCU that an event occurred and executes interrupt service routine corresponding to the received interrupt. Once the interrupt is complete, it returns to the main program. The TramelBlaze shares the 100MHz clock from the FGPA. The 16-bit output Port_ID produces the address/instruction. The 16-bit Out_Port produces data contained within the TramelBlaze.

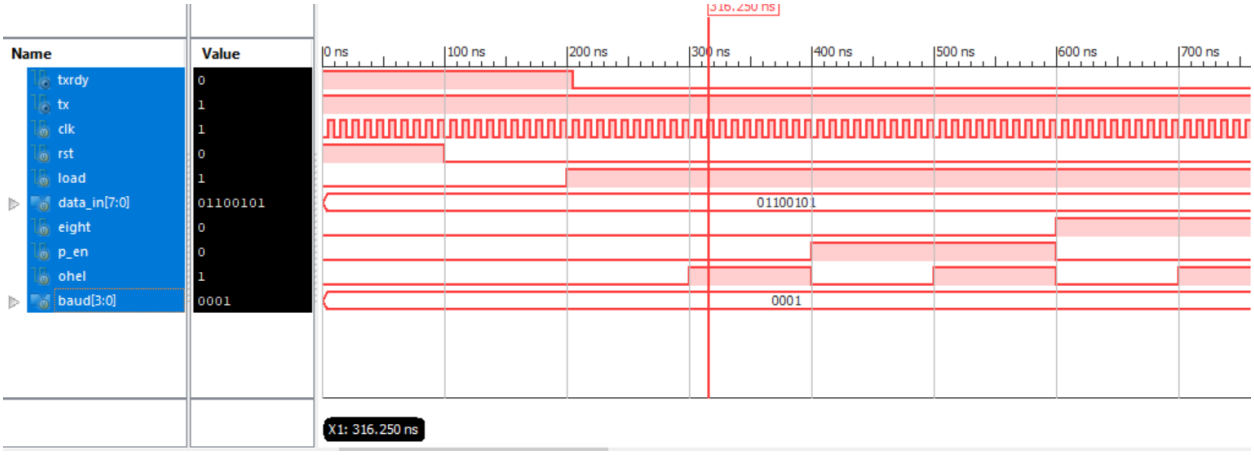
Read_Strobe output is asserted high indicating that the input data on the In_Port was captured to the specified data register during an input instruction. The Write_Strobe is asserted high validating the output signal data of the Out_Port. Lastly, the Int_Ack is asserted high to acknowledge that an interrupt event occurred.

6. Chip Verification

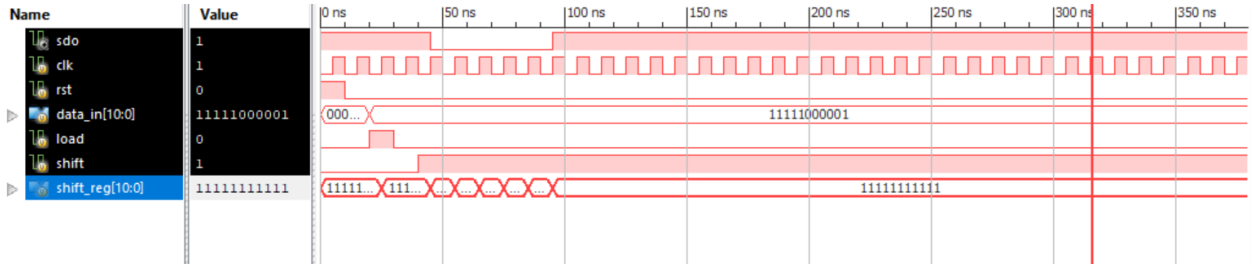
6.1 Block Names

- Transmit Engine TestFixture
- Shift Register TestFixture
- Parity Decoder TestFixtue

6.1 Testbench Waveforms

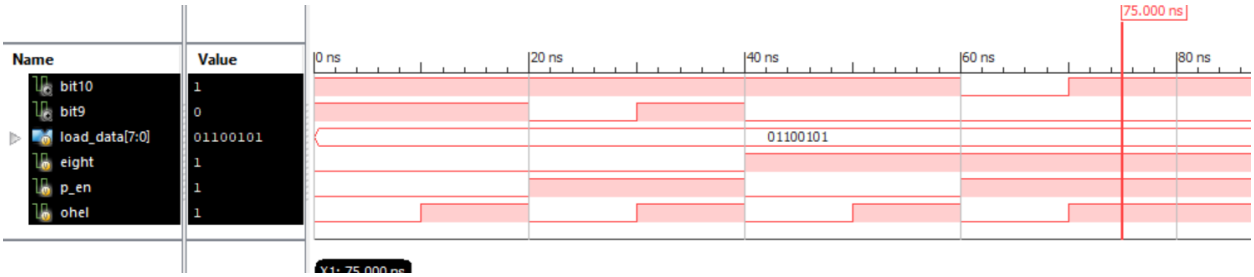


Transmit Engine



Shift Register

Prepared by: Jose Sotelo	Date: March 19, 2018	Revision: 1



Parity Decoder

01100101, 0, 0, 0, 1,1
01100101, 0, 0, 1, 1,1
01100101, 0, 1, 0, 1,0
01100101, 0, 1, 1, 1,1
01100101, 1, 0, 0, 1,0
01100101, 1, 0, 1, 1,0
01100101, 1, 1, 0, 0,0
01100101, 1, 1, 1, 1,0

To the left truth table Inputs: Data_In, Eight, Parity Enable and Ohel. Outputs: bit10 and bit 9 on the right.

7. Software Code

Appendix A: Key Terms

Start Bit

The UART data transmission line is normally held at a high voltage level when it's not transmitting data. To begin the transfer of data, the Tx pin pulls the transmission line from high too low for one clock cycle. When the receiving UART detects the high to low voltage transition, it begins reading the bits in the data frame at the frequency of the baud rate selected.

Data Frame/Packet

The data frame contains the actual data being transferred. It can be 5 to 8 bits long if a parity bit is used. If no parity bit is used, the data frame can be 9 bits long. In certain cases, the data sent can begin with the least significant bit first.

Parity

A parity bit is form of error checking that describes the evenness or oddness of a number. The parity bit is a way for the receiving UART to tell if any data has changed during transmission. If the parity bit is a 1 (odd parity), then there are an odd number of 1 bits in the data frame. If the parity bit is a 0 (even parity), then there are an even number of 1 bits in the data frame.

Stop Bit

To signal the end of the data transmission, the sending UART drives the data transmission line from a low voltage to a high voltage.

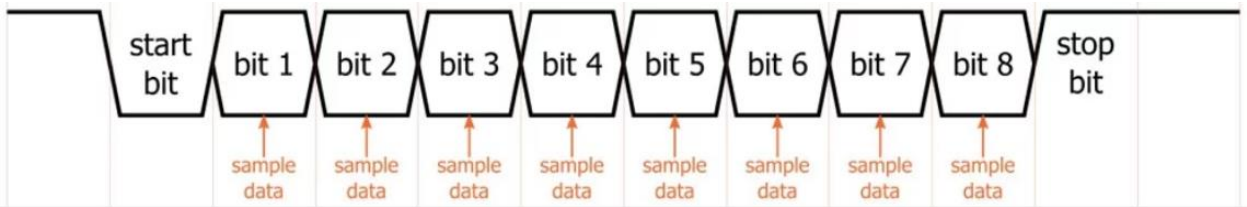
Baud Rate

The baud rate identifies the frequency the data is being transmitted at. (Bits per second)

Bit time

Bit time is the amount of time data bits are held on the wire.

Illustration of Data Transmission



Prepared by:	Date:	Revision:
Jose Sotelo	March 19, 2018	1