

**College of Engineering**

2018

**Department of Computer Engineering and Computer Science**

UART Full Duplex Protocol and Technology Specific Instantiation Chip Specification

**PREPARED FOR**

Mr. John Tramel

CECS 460 System-On-Chip Design

**PREPARED BY**

Jose Sotelo

**Table of Contents**

**1 Introduction5**

1.1 Purpose5

2 Applicable Documents 6

2.1 External Documents6

2.2 PicoBlaze 8-bit Microcontroller User’s Guide……………………………………6

2.3 TramelBlaze…………………………………………………………………….6-7

2.4 TramelBlaze ISA………………………………………………………………….8

3 Requirements9

3.1.1 Performance Requirements9

3.1.2 Interface Requirements9

3.1.3 Physical Requirements9

3.1.4 Power Requirements…………………………………………………………...9-10

4 Implementation11

4.1.1 Design Description………………………………………………………...…11-12

4.1.2 Top Level Description12

4.1.3 Block Diagram13

4.1.4 Input/Output Interface Description………………………………………………14

4.1.5 Clocks14

4.1.6 Resets14

**5 Internally Acquired Blocks15**

5.1 Core\_Logic………………………………………………………………………15

5.2 TSI……………………………………………………………………………….16

5.3 UART\_Top…………………………………………………………………...17-19

5.4 Tx Engine…………………………………………………………………….20-21

5.5 Rx Engie……………………………………………………………………...22-25

5.6 Rx FSM………………………………………………………………………….26

5.7 Rx Remap………………………………………………………………………..27

5.8 Baud Decoder…………………………………………………………………….28

5.9 Tx Shift Register…………………………………………………………………29

5.10 Tx Bit Time Counter……………………………………………………………..30

5.11 Tx Bit Counter…………………………………………………………………...31

5.12 Tx Parity Gen…………………………………………………………………….32

5.13 Rx Bit Time Counter……..………………………………………………………33

5.14 Rx Bit Counter…………………………………………………………………...34

5.15 SR Flop…………………………………………………………………………..35

5.16 PED………………………………………………………………………………36

5.17 AISO……………………………………………………………………………..37

**6 Externally Acquired Blocks38**

6.1 TramelBlaze Embedded Processor38

6.2 Description38

6.3 Block Diagram38

6.4 I/O Definition38-39

6.5 I/O Timing……………………………………………………………………….39

**7 Chip Verification40**

7.1 Testbench WaveForms40-43

**8 Software Code44**

Top\_Level.v…………………………………………………………………..44-45

Core\_Logic…………………………………………………………………...46-48

TSI.v………………………………………………………………………….49-51

AISO.v…………………………………………………………………………...52

UART\_Top.v…………………………………………………………………53-55

SR\_Flop.v………………………………………………………………………..56

Baud\_Dec.v………………………………………………………………………57

Tx\_Engine.v………………………………………………………………….58-60

Rx\_Engine.v………………………………………………………………….61-62

PED………………………………………………………………………………63

SR\_Flop\_Txrdy.v………………………………………………………………..64

Tx\_Load\_Reg.v………………………………………………………………….65

Tx\_Parity\_Gen\_Dec……………………………………………………………..66

Tx\_shift\_register.v……………………………………………………………….67

Tx\_bit\_time\_counter.v………………………………….………………………..68

Tx\_bit\_counter.v…………………………………………………………………69

Rx\_Engine\_Control.v………………………………………………………...70-71

Rx\_FSM.v……………………………………………………………………72-73

Rx\_bit\_time\_counter.v……………………………………………………..……74

Rx\_bit\_counter.v…………………………………………………………..…74-76

Rx\_DataPath.v………………………………………………………………..77-79

Rx\_Shift\_Reg.v…………………………………………………………………..80

Rx\_Remap.v……………………………………………………………………...81

Assembly Code……………………………………………………………….82-93

**Appendix A: Key Terms 94**

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 that is been 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.

In Revision 2, we interface the full duplex UART circuit with the technology specific instantiation (TSI) circuit. The TSI contains all references to the target technology library. All the communications between the inputs and outputs of the device pass through the TSI. The I/O cells for a design are one of the primary inclusions of the TIS block. Each I/O of the chip must have a particularly selected device to meet electrical and timing requirements of the external interface. We will be using Nexys Artix-7 Library Guide which defines the cells and their capabilities.

1.1 Purpose

For this project, we will be designing an SOPC (System on Programmable Chip) that contains the Transmit Engine, Receive Engine and technology specific instantiation (TSI). The two engines created in this chip specification will make the full duplex UART protocol. The user will be able to configure the transfer of 8 or 7 bits, a parity bit, odd or even bit and the baud rate. We will interface the Transmit and Receive Engine with the 16-bit TramelBlaze and the TSI to send and receive data via a serial terminal RealTerm. The serial terminal will output a Banner, prompt the user, display the hometown of the designer, delete and input characters, and print the number of characters in the current line.

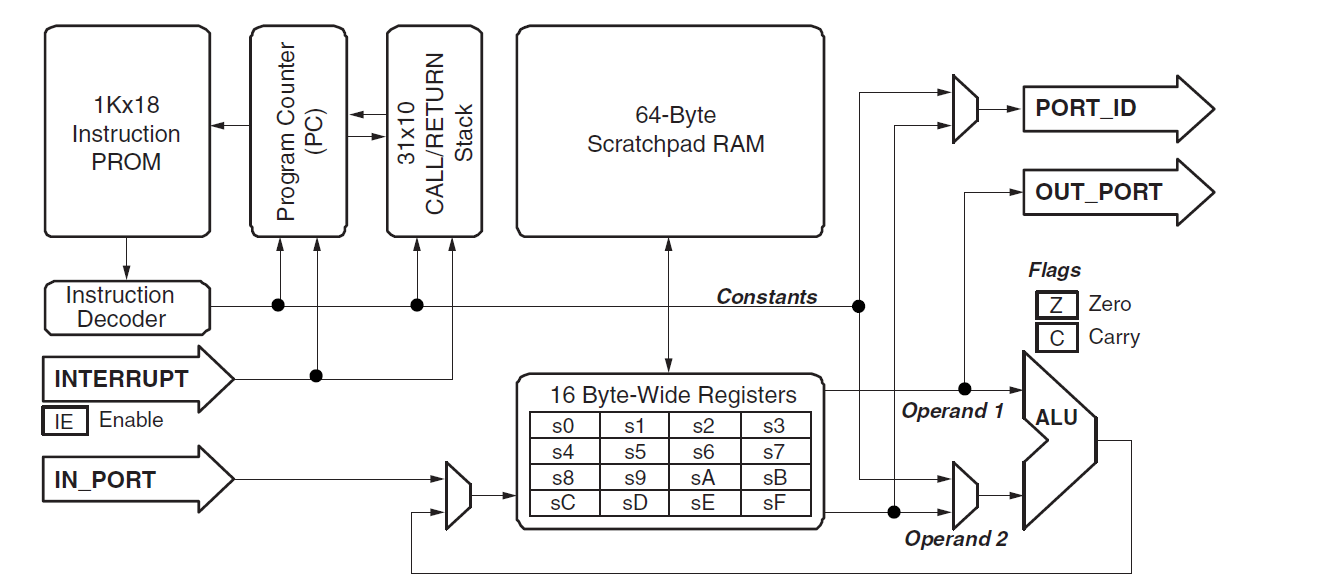
**2. Applicable Documents:**

2.1 External Documents

These documents were used in the development of the full duplex UART protocol. There main purpose was to assist the designer in the understanding of the embedded microcontroller the TramelBlaze and how to interface it with other FPGA logic.

2.2 PicoBlaze 8-bit Microcontroller User’s Guide

The PicoBlazeTM embedded microcontroller is an efficient, cost-effective embedded processor core for Spartan -3, Virtex -II, and Virtex-II Pro PFGAs. It is natively hosted on the Nexys 3/6 families. The 8-bit PicoBlaze microcontroller is specially designed and optimized for the Spartan -3, Virtex -II, and Virtex-II Pro architectures thus allowing for other FPGA logic to be connect to an embedded microcontroller’s input and output ports.

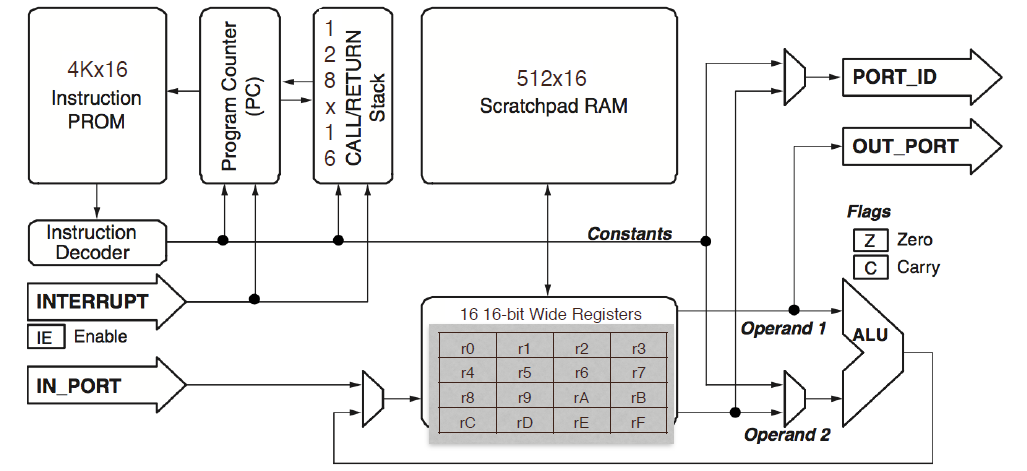


2.3 TramelBlaze

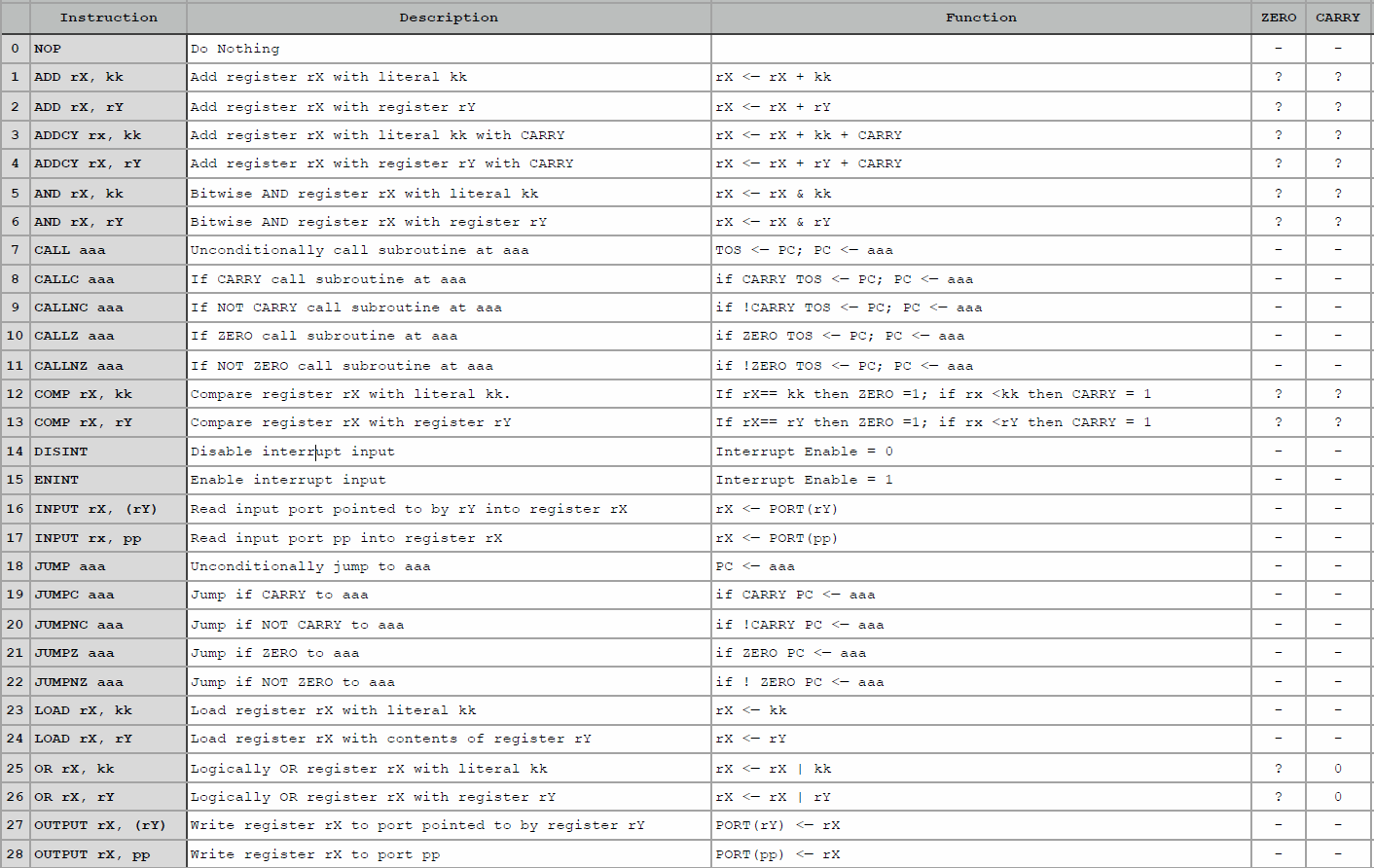
The TrambelBlaze is a 16-bit processor core designed to emulate the Xilinx PicoBlaze. We will be using the TramelBlaze to interface it with develop FPGA logic. Since we switch on to the Nexys 4 FPGA, the PicoBlaze is no longer supported. Thus, the TramelBlaze was developed by John Tramel to allow the student to interface the embedded microcontroller with their own developed FPGA logic. The instruction set for the TramelBlaze is the PicoBlaze instruction set.

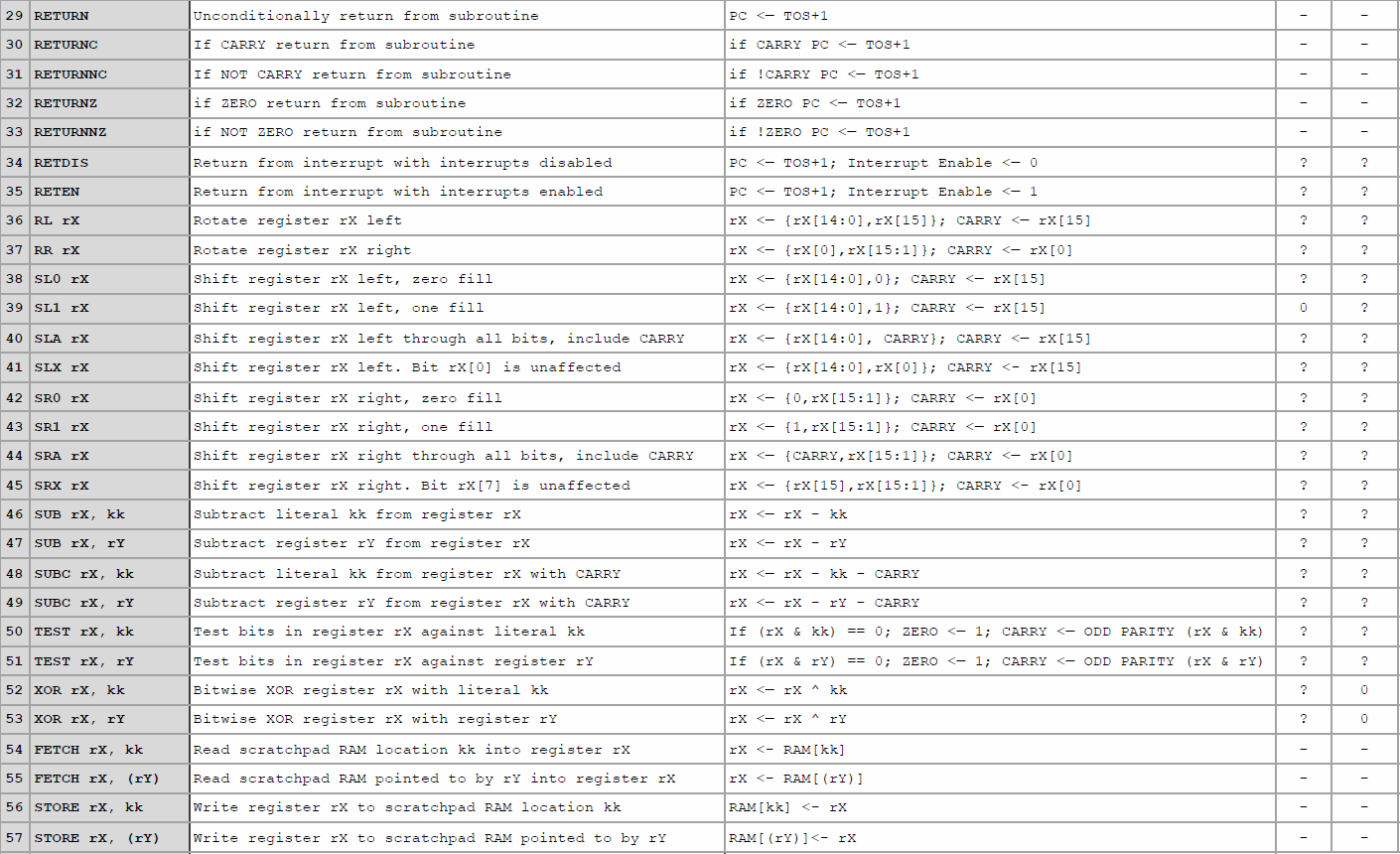
The TramelBlaze has the following:

* 16 16-bit wide general-purpose registers (r0-rF)
* 4096 Instruction Program Store - The code ROM provides storage for a suitable number of instructions. The instructions are either one or two words each.
* Arithmetic Logic Unit (ALU) – 16-bit wide ALU performs all microcontroller operations such as basic arithmetic, bitwise logic operations, arithmetic compare, comprehensive shift and rotate operations, program control operation such as jump or call.
* Flags – The Zero flag indicates the result of the previous operation was zero and the CARRY flag indicates various conditions. The INTERRUPT\_ENABLE flag indicates interrupt operations.
* 512-word scratchpad – A RAM within the processor with access provided by STORE and FETCH instructions. STORE copies the contents of a register to the RAM and FETCH copies the contents of the RAM to a register.
* I/O – the Input/Output ports extends the TramelBlaze’s capabilities to allow interfacing to other PFGA logic. The addressing capabilities of the I/O instructions is 0 to 65535 (FFFF) which will require an external address decoder. INPUT operations move data from surrounding FPGA logic into a register and the OUTPUT operations move data from a register to the surrounding FPGA logic.
* Program Counter – The PC points to the next instruction to be executed. JUMP, CALL, RETURN, RETURNI instructions and interrupt or rest events will modify the contents of the PC.



2.4 Programming TramelBlaze





**3. Requirements:**

3.1.1 Performance Requirements

The full duplex Universal Asynchronous Receiver will be designed in the Nexy4 DDR FPGA board. It 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, Receive Engine, TramelBlaze, TSI, and the serial terminal RealTerm will be used to transmit and receive data.

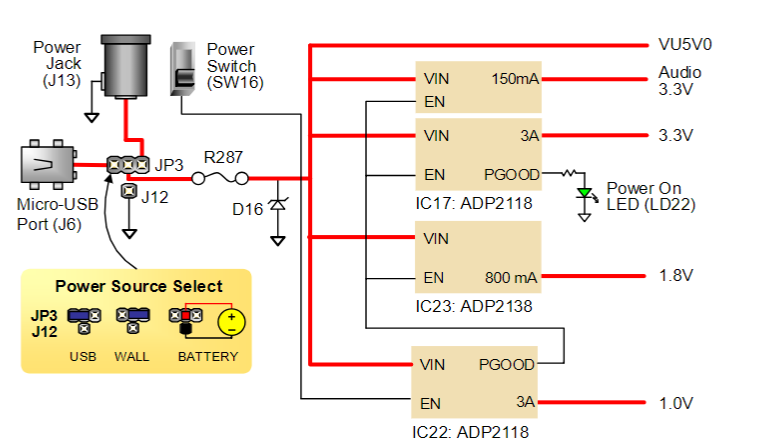
3.1.3 Physical Requirements

The highest baud rate the full duplex UART engine is capable to communicate via the serial terminal without breaking the data being sent is 921600.

3.1.4 Power Requirements

The Nexys4 DDR board can receive power from the Digilent USB-JTAG port(J6) or from an external power supply. Jumper JP3 (near the power jack) determines which source is used. All Nexys4 DDR power supplies can be turned on and off by a single logic-level power switch (SW16).

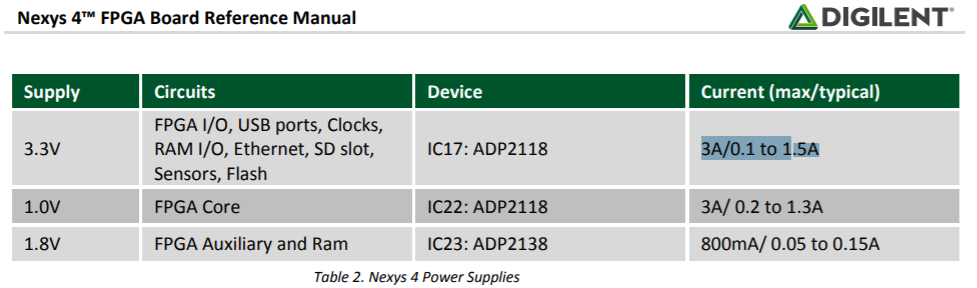
A power-good LED (LD22), driven by the “power good” output of the ADP2118 supply, indicates that the supplies are turned on and operating normally.



The USB port can deliver enough power for the vast majority of designs. The out-of-the-box demo draws ~400mA of current from the 5V input rail. Depending on your application, more power might be required to power your design. Some applications can be run without being connected to a PC’s USB port. In these instances, an external power supply or battery pack can be used.

External power supply can be used by plugging into the power jack (JP3) and setting jumper J13 to “wall”. The supply must use a coax, center-positive 2.1mm internal-diameter plug and deliver 4.5VDC to 5.5VDC and at least 1A of current.

External battery pack can be used by connecting the battery’s positive terminal to the center pin of JP3 and the negative terminal to the pin labeled J12, directly below JP3. Since the main regulator on the Nexys4 DDR cannot accommodate input voltages over 5.5VDC, an external battery pack must be limited to 5.5VDC. If the USB Host function (J5) is used, at least 4.6V needs to be provided. In other cases, the minimum voltage is 3.6V.



**4. Implementation**

4.1.1 Design Description

This project implements the Transmit and the Receive Engine with selectable baud rates, 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. For the Receive Engine, the designer must implement receive engine control and data path. The goal of the receive engine is to synchronize the data collection with the Tx communication by always polling the Rx line looking for a high to low transition indicating the arrival of the Start bit. A simple finite state machine ensures that the Start bit remains active until the mid-bit at which time the data collection will proceed at a bit time until all bits are received. The FSM will output Start and DoIt indicating that the receive engine is looking for a Start bit and that the receive engine is currently processing data.

The receive engine data path is used to read and write data. There are two addresses dedicated to the UART: 0000 (UART DATA) and (UART Status). When the TramelBlaze writes address 0000, the transmit engine will receive the data byte. When the TramelBlaze reads address 0000 the receive engine will respond with the received data. When the TramelBlaze reads address 0001 it will receive the status collected by the receive engine along with two ready bits (tx ready, rx ready). The other three status flags PERR (Parity Error) indicates that a Parity Error has been detected, FERR (Framing Error) n indicates that the synchronization is lost when no Stop bit is found, and lastly OVF (Overflow) indicates that another byte is being received before the TramelBlaze has read the previous byte.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 |

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.

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 registers outputs 11-bits 1’s. The output of the shift register gets the LSB 0 of the data being shifted.

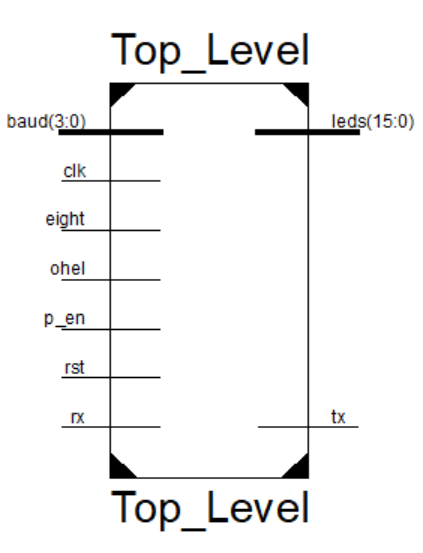
The design must also include two counters: 1) Count clocks to determine the bit time and 2) Count the bits to know when we are done. These two counters will let us know when we have waited a bit time and to know when all the bits have been transmitted. Lastly, a Baud Decoder circuit is created to generate the number of clocks that are necessary to fill one-bit time. Combining all the modules describe above, produces the Transmit Engine. Once the Transmit and Receive Engine are completed, you can interface the UART with the 16-bit TramelBlaze microcontroller. For the user to select a different baud speed, the following table is used to select the baud by using the on-board switches.

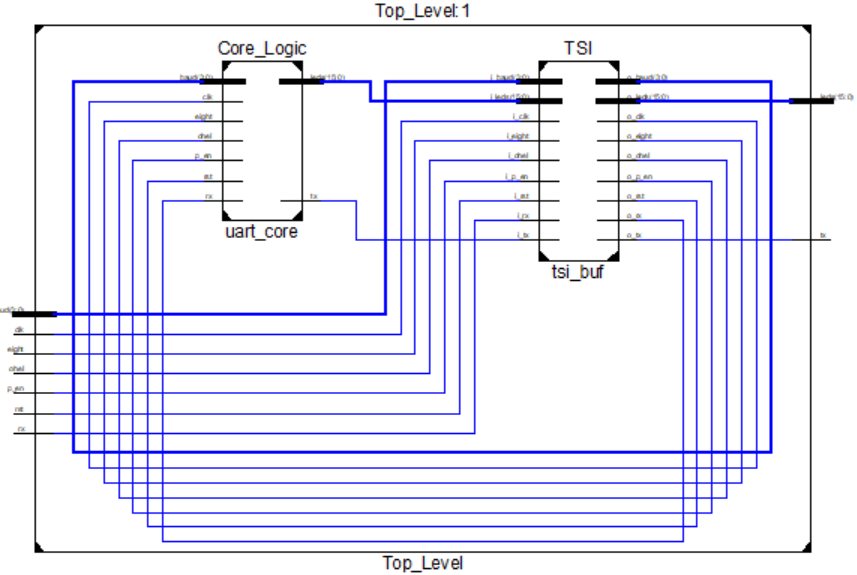
|  |  |  |  |
| --- | --- | --- | --- |
| **Baud Switches** | **Baud Rate** | **Bit Time** | **Nexys4 Count** |
| 0000 | 300 | 0.003333 | 333,333 |
| 0001 | 1200 | 0.0008333 | 83,333 |
| 0010 | 2400 | 0.000416667 | 41,667 |
| 0011 | 4800 | 0.000208333 | 20,833 |
| 0100 | 9600 | 0.000104167 | 10,417 |
| 0101 | 19200 | 5.20833E-05 | 5,208 |
| 0110 | 38400 | 2.60417E-05 | 2,604 |
| 0111 | 57600 | 1.73611E-05 | 1,736 |
| 1000 | 115200 | 8.68056E-05 | 868 |
| 1001 | 230400 | 4.34028E-06 | 434 |
| 1010 | 460800 | 2.17014E-06 | 217 |
| 1011 | 921600 | 1.08507E-06 | 109 |

4.1.2 Top Level Description

The top-level block diagram contains the Core design and the TSI. The Core design consist of the full UART and TSI contains all the references to the target technology libraries. For the Core design to communicate with the outside world, it’s I/O’s must pass through the TSI before interacting with the FPGA. The assembly code written for the TramelBlaze will be used to display text to the serial terminal. The requirements are to display a banner, hometown when entering an asterisk, line count, and backspace to delete previously entered characters.

4.1.3 Top Level Block Diagram





|  |  |  |  |
| --- | --- | --- | --- |
| **I/O** | | **Signal Description** | **FPGA board Switches & Buttons** |
| Input | clk | 100MHz FPGA board crystal oscillator | None |
| Input | rst | Synchronous Rest |  |
| Input | baud | 4-bit input used to select baud rate | LOC Pin Assignment: R13(SW8), U18(SW7), T18(SW6), R17(SW5) |
| Input | eight | 1-bit input to select [7:0] ot [6:0] bits | LOC Pin Assignment: R15(SW4) |
| Input | p en | Parity enable – 1: odd parity, 0: no parity | LOC Pin Assignment: M13(SW3) |
| Input | ohel | odd high even low- 1: odd parity, 0: even parity | LOC Pin Assignment: L16(SW2) |
| Input | rx | Receive signal which indicates a high to low transition | LOC Pin Assignment: C4 |
| Output | tx | Transmission signal which indicates that data is being shifted out from Transmit Engine | LOC Pin Assignment: D4 |

4.1.5 Clocks

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

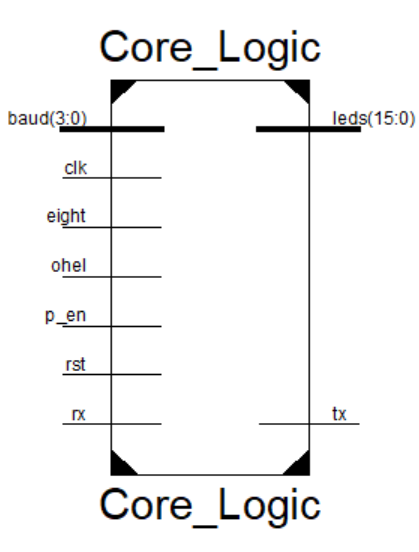
4.1.6 Resets

All the designs used a synchronous reset (AISO) to prevent metastability. The purpose for the AISO circuit in our design is to prevent any of the flops in our design from entering a metastable state when reset is released. Synchronously releasing reset ensures all-of the flops are reset at the same time and that they are stable when reset is released.

**5. Internally Acquired Blocks**

5.1 Core Logic

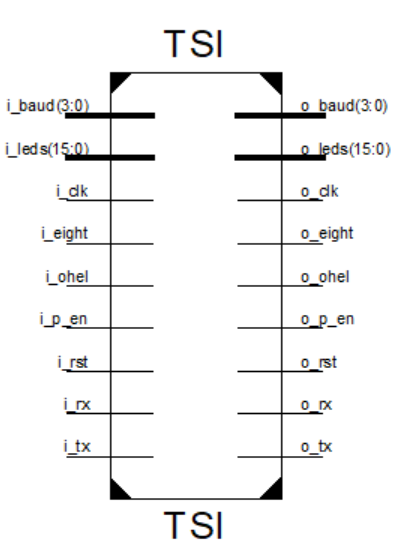
The Core Logic Circuit is used to combine the Address Decoder, UART, and TramelBlaze so it could interface with the technology specific instantiation (TSI) circuit. The core logic also contains and AISO, PED, SR flop, counters, load register, and shift register that are essential to the design.



|  |  |  |
| --- | --- | --- |
| **Name** | **I/O** | **Description** |
| clk | I | Synchronous clock |
| rst | I | Synchronous reset |
| baud[3:0] | I | 4-bit input to select baud rate |
| eight | I | 1-bit input to select 8-bit data transmission |
| ohel | I | 1-bit input to select if we want even or odd |
| p\_en | I | 1-bit input to select parity bit |
| rx | I | Receive signal looks for a high to low transition indicating the arrival of the start bit |
| leds | O | LEDs are used to demonstrate the TramelBlaze is reading data |
| tx | O | Data being shifted out of the transmit engine |

5.2 Technology Specific Instantiation (TSI)

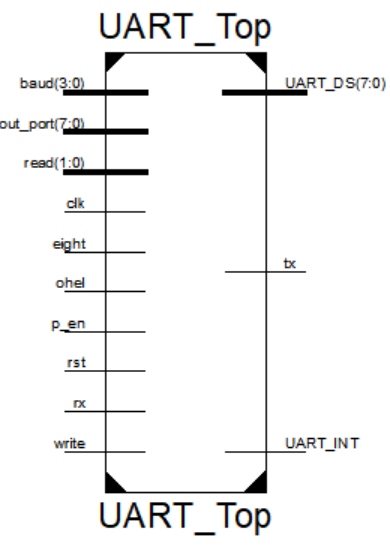
The TSI contains all references to the target technology library. All communications to or from the I/O of the device pass through the TSI. Each I/O of the chip must have a particular selected device to meet electrical timing requirements of the external interface.

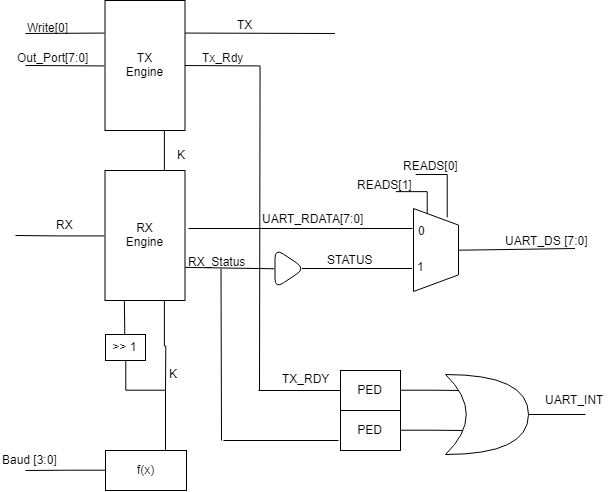


|  |  |  |
| --- | --- | --- |
| **Name** | **I/O** | **Description** |
| i\_baud[3:0] | I | Baud input buffer |
| i\_leds | I | LEDs input buffer |
| i\_clk | I | Clock input buffer |
| i\_eight | I | Eight input buffer |
| i\_ohel | I | Odd high even low input buffer |
| i\_p\_en | I | Parity enable input buffer |
| i\_rst | I | Reset input buffer |
| i\_rx | I | Receive input buffer |
| i\_tx | I | Transmit input buffer |
| o\_baud[3:0] | O | Baud output buffer |
| o\_leds | O | LEDS output buffer |
| o\_clk | O | Clock output buffer |
| o\_eight | O | Eight output buffer |
| o\_ohel | O | Odd high even low output buffer |
| o\_p\_en | O | Parity enable buffer |
| o\_rst | O | Rest output buffer |
| o\_rx | O | Receive output buffer |
| o\_tx | O | Transmit output buffer |

5.3 UART Top

The UART Top in composed of the Transmit Engine, Receive Engine, Baud Decoder, UART status flags and pulse edge detectors. Both engines in this design require a specific baud rate selected by the user. An interrupt signal is also implemented to cause the TramelBlaze to enter an Interrupt Service Routine. The UART top also contains status flag such as parity error, over flow, and framing error. The tx signal is used to transmit data over USB.

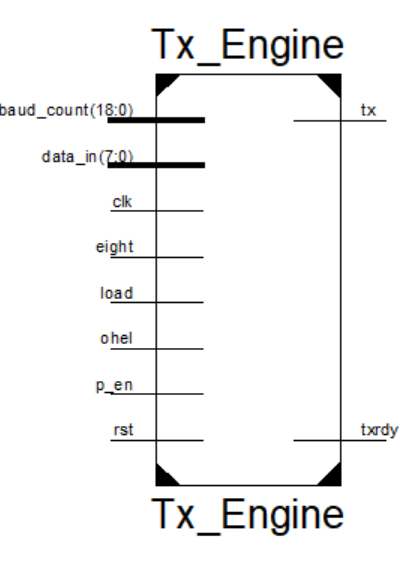


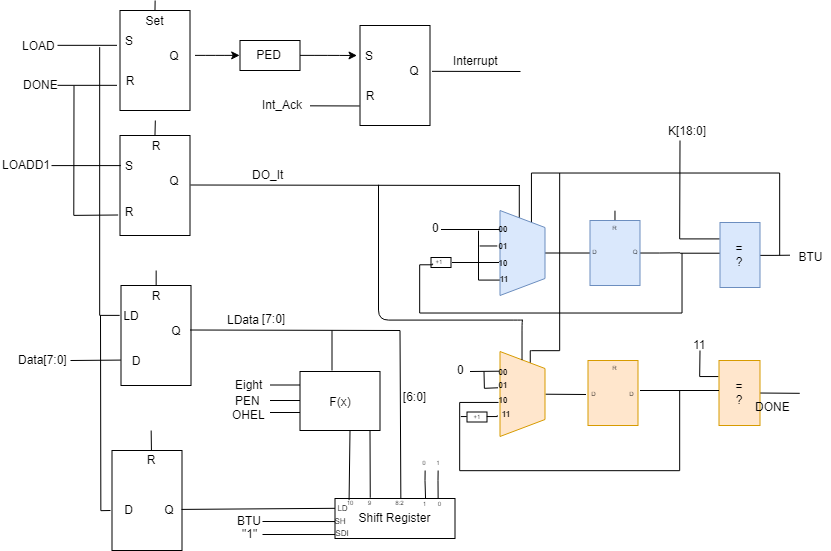


|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Signal Description** | **I/O** | **FPGA board Switches & Buttons** |
| clk | 100MHz FPGA board crystal oscillator | I | None |
| rst | Synchronous Rest | I | LOC Pin Assignment: M18 |
| write | Write address 0000 | I | None |
| read | Read address 0001 | I | None |
| out\_port | 8-bit data coming from TramelBlaze | I | None |
| baud | 4-bit input used to select baud rate | I | LOC Pin Assignment: R13(SW8), U18(SW7), T18(SW6), R17(SW5) |
| eight | 1-bit input to select [7:0] ot [6:0] bits | I | LOC Pin Assignment: R15(SW4) |
| p\_en | Parity enable – 1: odd parity, 0: no parity | I | LOC Pin Assignment: M13(SW3) |
| ohel | odd high even low- 1: odd parity, 0: even parity | I | LOC Pin Assignment: L16(SW2) |
| rx | Receive signal which indicates a high to low transition | I | LOC Pin Assignment: C4 |
| tx | Transmission signal which indicates that data is being shifted out from Transmit Engine | O | LOC Pin Assignment: D4 |
| UART\_DS | Status flags: parity error, framing error, and over flow error | O | None |
| UART\_Int | Set an interrupt signal that gets feed into the TramelBlaze | O | None |

5.4 Transmit Engine

The Transmit Engine is composed of an SR flop, loadable register, parity decoder, a register, shift register, bit time counter, and a bit counter.

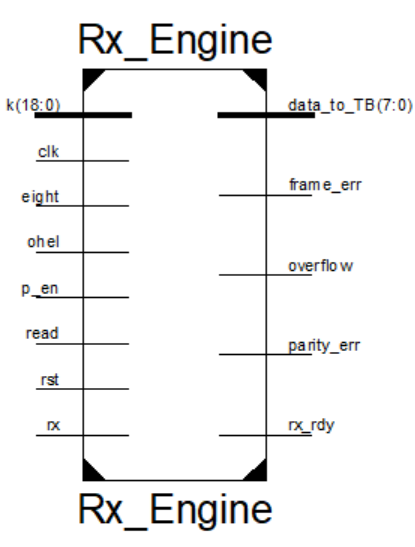




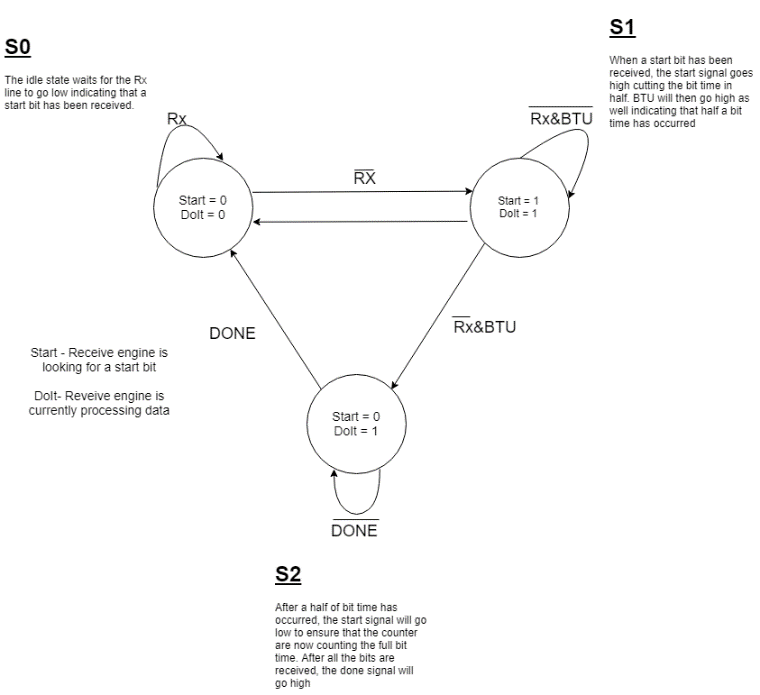
|  |  |  |
| --- | --- | --- |
| **Name** | **I/O** | **Description** |
| baud\_count[18:0] | I | Baud Rate Decoder |
| data\_in[7:0] | I | Data from TramelBlaze |
| clk | I | 100MHz crystal oscillator |
| eight | I | 8 bits |
| ohel | I | Odd high even low |
| p\_en | I | Parity enable |
| load | I | From Address Decoder |
| rst | I | AISO |
| tx | O | USB |
| tx rdy | O | UART Interrupt |

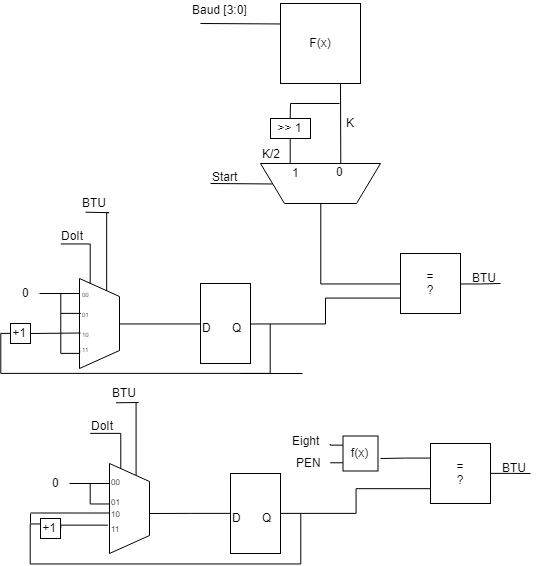
5.5 Receive Engine

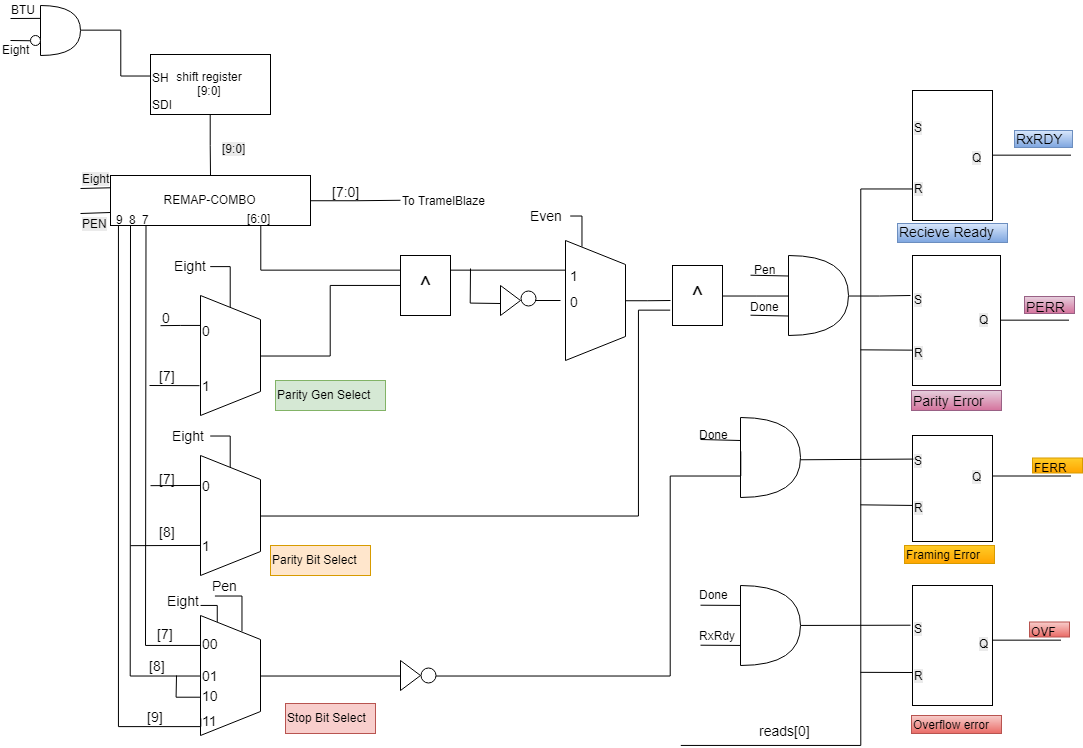
The Receive Engine is composed of control and data path. The control path contains a finite state machine, bit time counter and bit counter to sync the signals coming into the UART. The data path checks for the correct transmission, if the transmission is incorrect the status flags are set to indicate which type of error it was. Otherwise, the data will be outputted to the TramelBlaze.



|  |  |  |
| --- | --- | --- |
| **Name** | **I/O** | **Description** |
| k[18:0] | I | Baud Rate Decoder |
| clk | I | 100MHz crystal oscillator |
| eight | I | On board switch |
| ohel | I | On board switch |
| p en | I | On board switch |
| read | I | TramelBlaze |
| rst | I | AISO |
| rx | I | UART Interrupt |
| data to TB[7:0] | O | TramelBlaze |
| frame err | O | Framing error |
| over flow | O | Overflow error |
| parity err | O | Parity error |
| rx rdy | O | TramelBlaze |

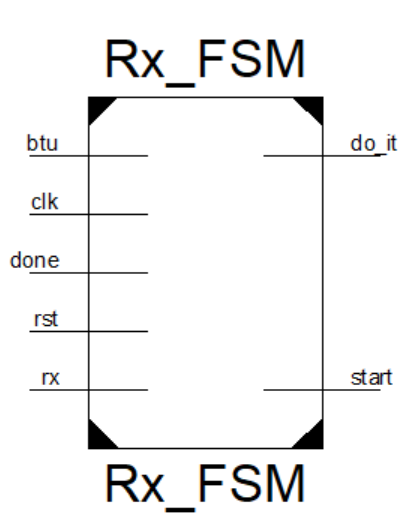






5.6 Rx FSM

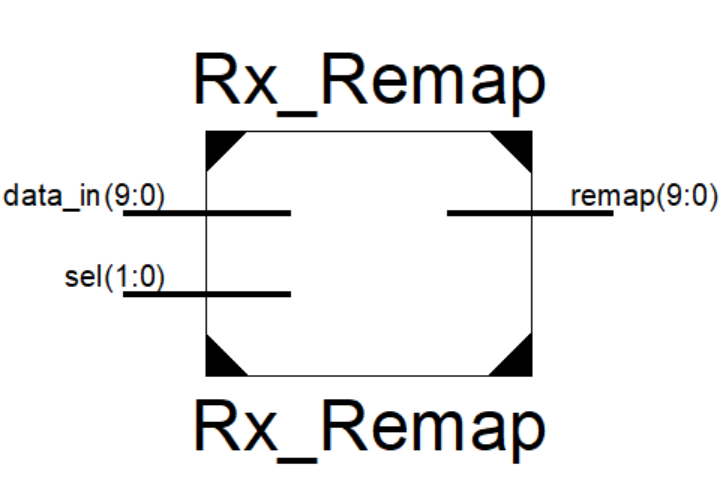
The Receive Engine contains a finite state machine that determines if its receiving a signal or not.



|  |  |  |
| --- | --- | --- |
| **Name** | **I/O** | **Description** |
| btu | I | Rx bit time counter |
| clk | I | 100MHz crystal oscillator |
| done | I | Rx bit counter |
| rst | I | AISO |
| rx | I | USB |
| do it | I | Rx bit time counter and bit counter |
| start | I | Rx bit time counter |

5.7 Rx\_Remap

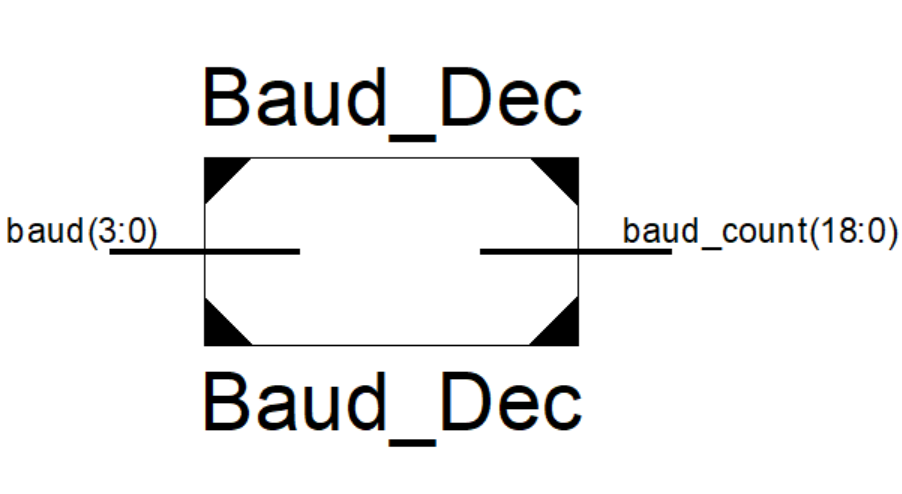
The purpose for the Rx ramap circuit is to justify the received data. Depending on the transmission data, the remap will adjust the data according to the eight and parity enable select switches. 8-bits from the remap circuit will go to the TramelBlaze, while the remaining bits will be distributed accordingly.



|  |  |  |
| --- | --- | --- |
| Name | I/O | Description |
| data in[9:0] | I | Data coming in from shift register |
| sel[1:0] | I | Eight and Parity enable inputs |
| remap[9:0] | O | Data out |

5.8 Baud Decoder

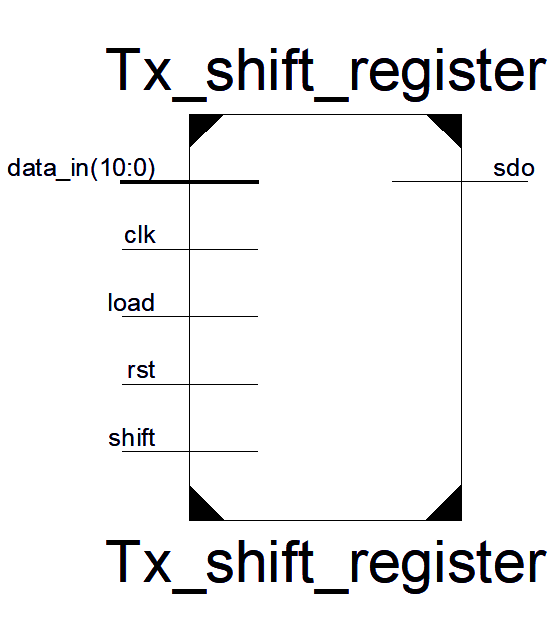
The Baud decoder is a mux that selects how many bits per second to run the design.



|  |  |  |
| --- | --- | --- |
| **Name** | **I/O** | **Description** |
| baud[3:0] | I | On board switches |
| baud count | O | Transmit Engine |

5.9 Tx Shift Register

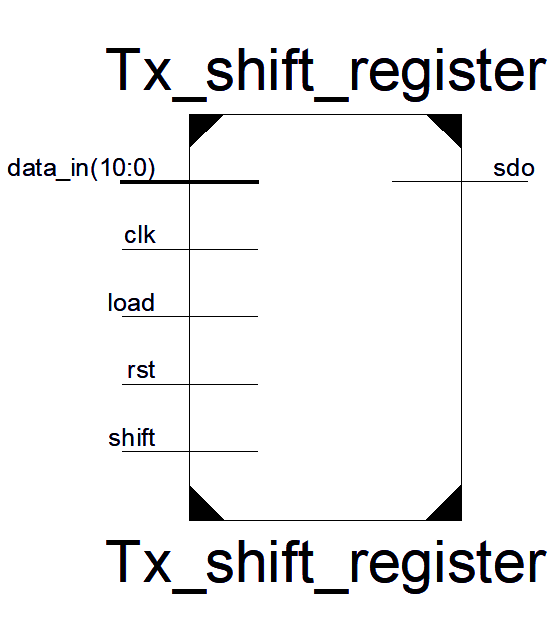
The shift register is responsible for taking in the data and shifting them into a register. If the switches eight and parity enable are selected, the data must be remapped accordingly.



|  |  |  |
| --- | --- | --- |
| **Name** | **I/O** | **Description** |
| clk | I | 100MHz crystal oscillator |
| rst | I | AISO |
| load | I | Load |
| shift | I | Shift Data |
| data in[10:0] | I | Data coming in |
| sdo | O | Data shifted out |

5.10 Tx Bit Time Counter

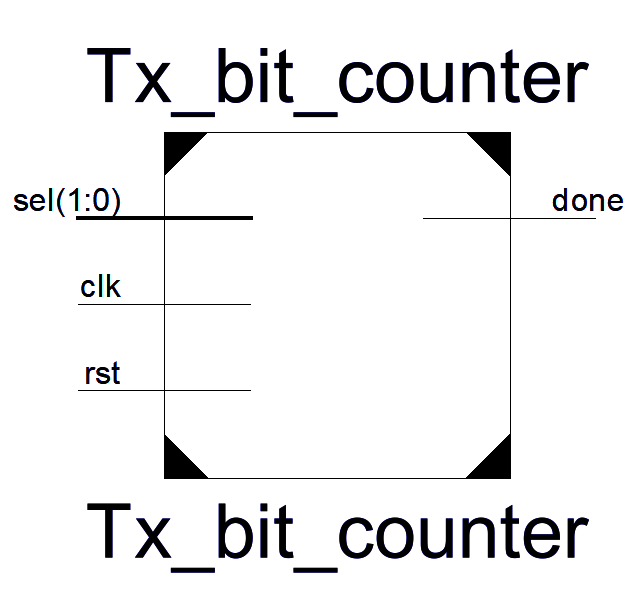
The Bit Time Counter is used to determine the speed at which the engine runs caused by the value selected in the baud decoder.



|  |  |  |
| --- | --- | --- |
| **Name** | **I/O** | **Description** |
| baud[3:0] | I | On board switches |
| sel[1:0] | I | DoIt and BTU |
| clk | I | 100MHz crystal oscillator |
| btu | O | Tx shift register and bit count |

5.11 Tx Bit Time Counter

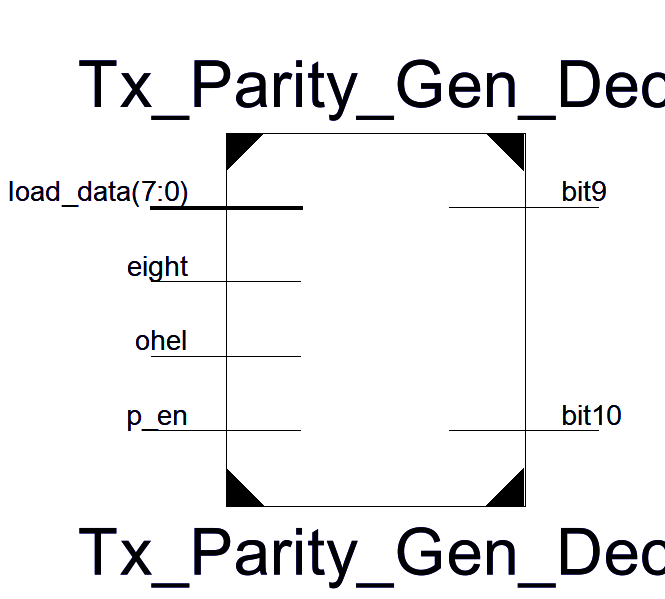
The Bit Counter is used to count the number of bits to ensure that the engine is picking up the correct number of bits. It also checks the values to see if there is any discrepancies by seen how many bits are need to receive based on the eight and parity enable switches.



|  |  |  |
| --- | --- | --- |
| **Name** | **I/O** | **Description** |
| sel[1:0] | I | DoIt and BTU |
| clk | I | 100MHz crystal oscillator |
| rst | I | AISO |

5.12 Tx Parity Generator Decoder

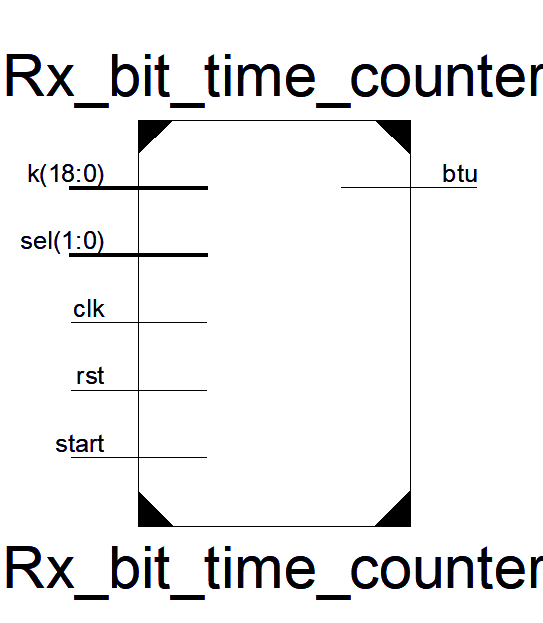
The transmit parity generator decoder is responsible for generating the even or odd parity bits depending on the configuration. If eight is selected, the loadable register will load 8 bits, else 7-bits. Depending on the data transmitted, if the data transmitted requires a parity bit the data will be account for and an even or odd no parity bit will be added.



|  |  |  |
| --- | --- | --- |
| **Name** | **I/O** | **Description** |
| load data[7:0] | I | 8 bit data coming loadable register |
| eight | I | On board switch |
| ohel | I | On board switch |
| p\_en | I | On board switch |
| bit9 | O | 9-bit data |
| bit10 | O | 10-bit data |

5.13 Rx Bit Time Counter

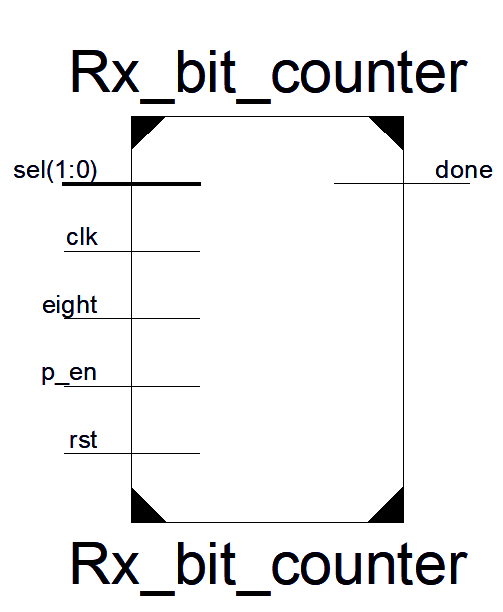
The bit timer counter is responsible for counting the number of clocks depending on the baud rate that is selected. The baud rate is configurable with the on-board switches. The baud rate determines the bit time which is the length of time that a transmitted bit is held on the wire.



|  |  |  |
| --- | --- | --- |
| Name | I/O | Description |
| k | I | Baud rate decoder |
| sel[1:0] | I | DoIt and Btu |
| clk | I | 100MHz clock oscillator |
| rst | I | AISO |
| start | I | FSM |
| btu | O | register |

5.14 Rx Bit Counter

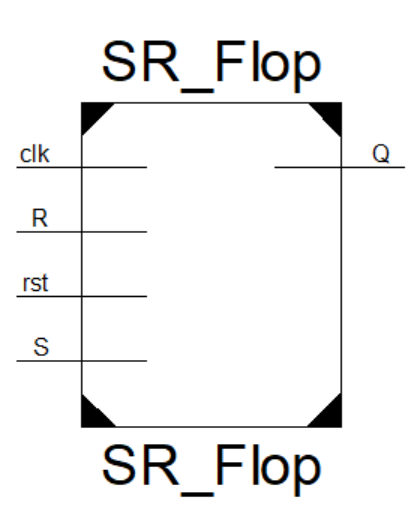
The bit counter in the transmit engine is responsible for counting the number of bits that have been transmitted. The purpose of this block is to count the number of bits that have been transmitted. The number of bits transmitted is always 11 and once all 11-bits have been transmitted, the done signal goes high which set the tx\_rdy signal high which indicates that the transmit engine is ready to transmit another byte of data.



|  |  |  |
| --- | --- | --- |
| Name | I/O | Description |
| sel[1:0] | I | DoIt and Btu |
| clk | I | 100MHz clock oscillator |
| eight | I | On board switches |
| p en | I | On board switches |
| rst | I | AISO |
| done | O | FSM and Rx Controller |

5.15 SR Flop

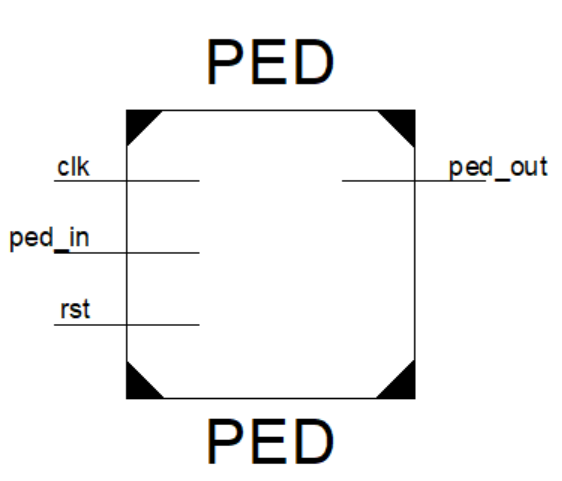
The SR flop are used to store information after the inputs are turned off. The set input causes the output of 1 and the R input cause the output of 0. For the transmit engine, when we reset the design, we need to set the reset value high.



|  |  |  |
| --- | --- | --- |
| **Name** | **I/O** | **Description** |
| clk | I | 100MHz crystal oscillator |
| R | I | TramelBlaze |
| rst | I | AISO |
| S | I | PED |
| Q | O | TramelBlaze |

5.13 Pulse Edge Detector (PED)

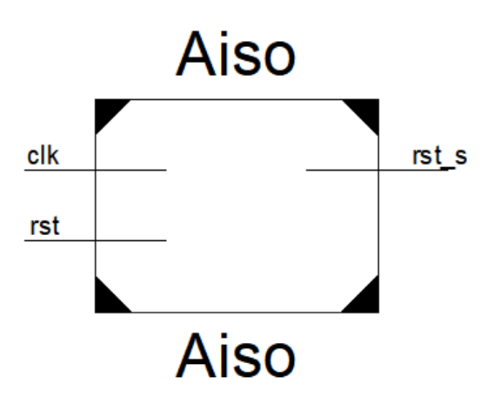
The PED is used to generate a one clock period pulse whenever the signal is asserted.



|  |  |  |
| --- | --- | --- |
| **Name** | **I/O** | **Description** |
| clk | I | 100MHz crystal oscillator |
| ped in | I | UART |
| rst | I | AISO |
| ped out | O | SR flop |

5.14 Asynchronous-In-Synchronous-Out (AISO)

The Asynchronous-In-Synchronous-Out circuit is used to synchronize the reset of digital flip flops.



|  |  |  |
| --- | --- | --- |
| **Name** | **I/O** | **Description** |
| clk | I | 100MHz crystal oscillator |
| rst | I | Button |
| rst s | O | Output to every block |

**6. Externally Acquired Blocks**

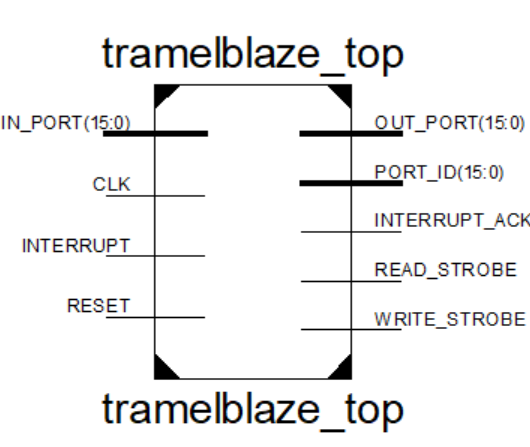
6.1 TramelBlaze

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

6.2 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.

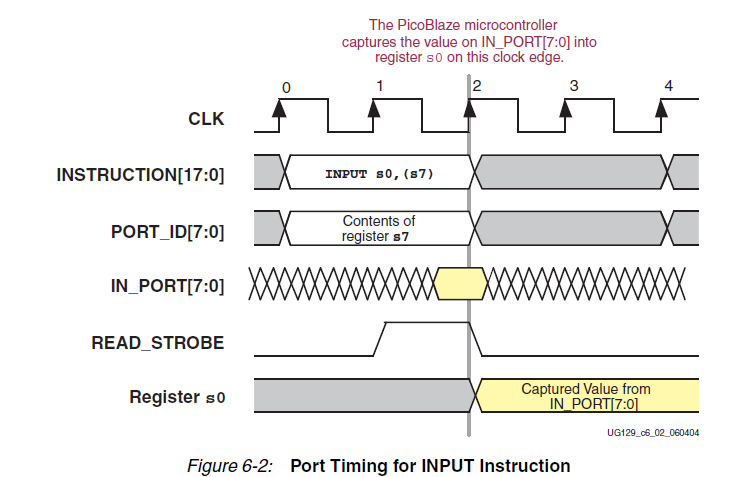
6.3 TramelBlaze Block Diagram

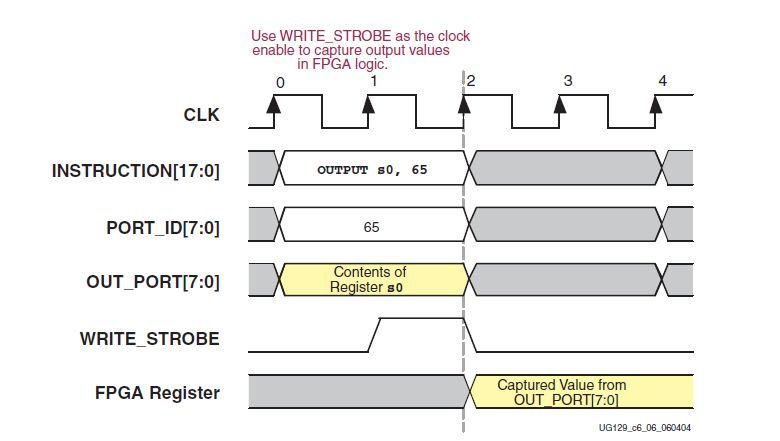


6.4 I/O Definitions

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.

6.5 I/O Timing



\

**7. Chip Verification**

7.1 TestbenchWaveforms

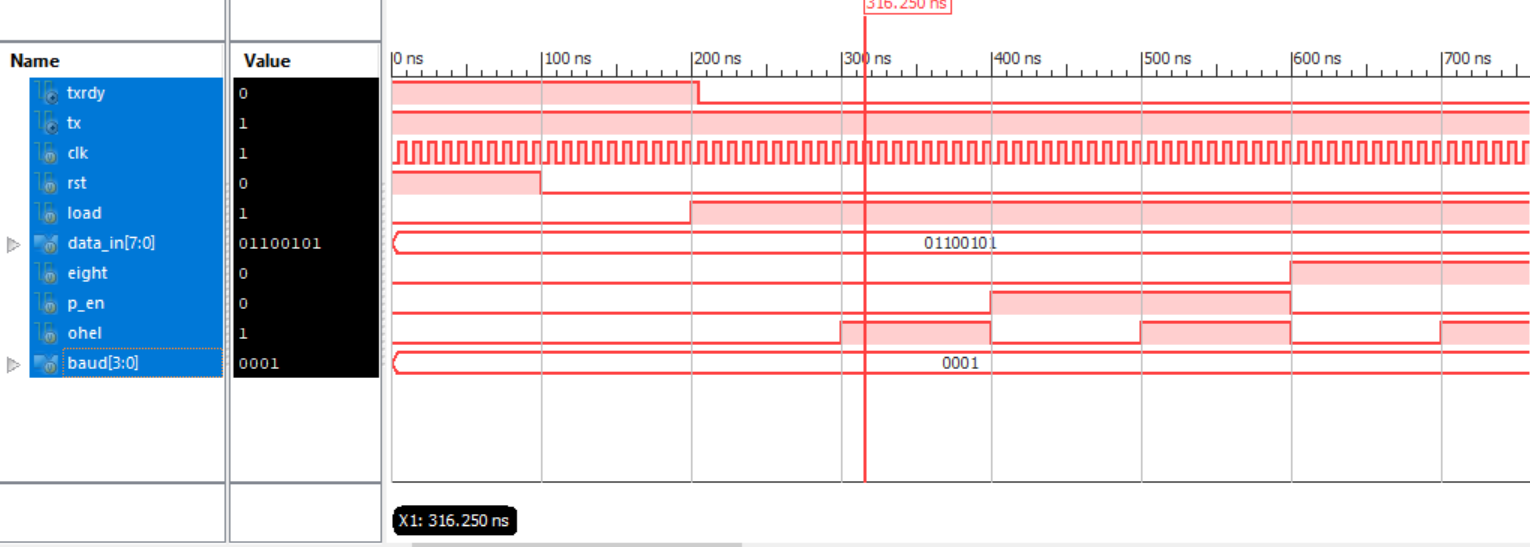


Figure 27: Transmit Engine TestFixture

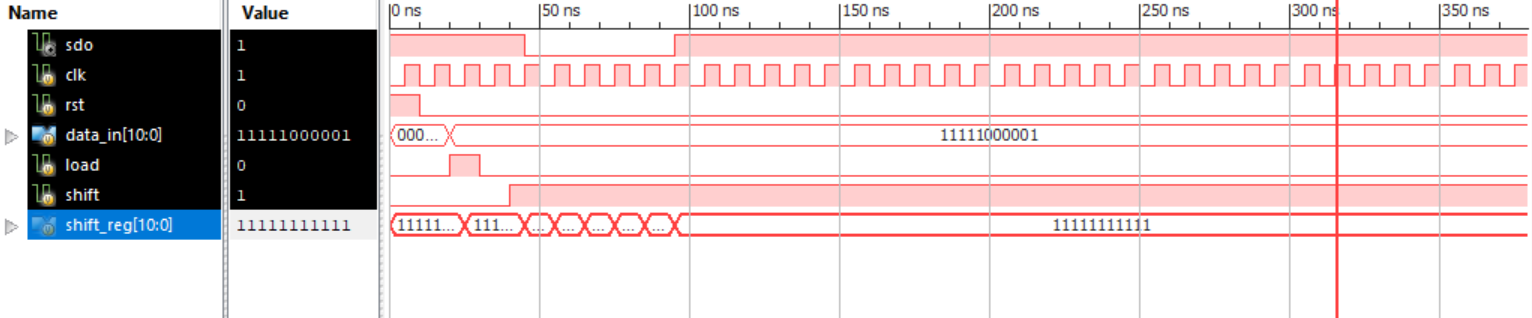


Figure 28: Shift Register TestFixture

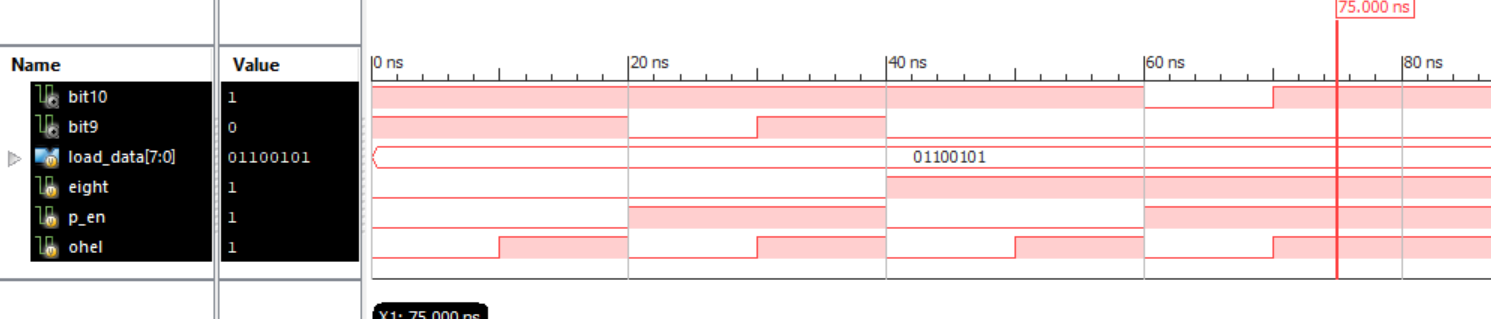
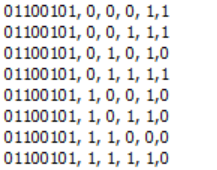


Figure 29: Parity Decoder TestFixture



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

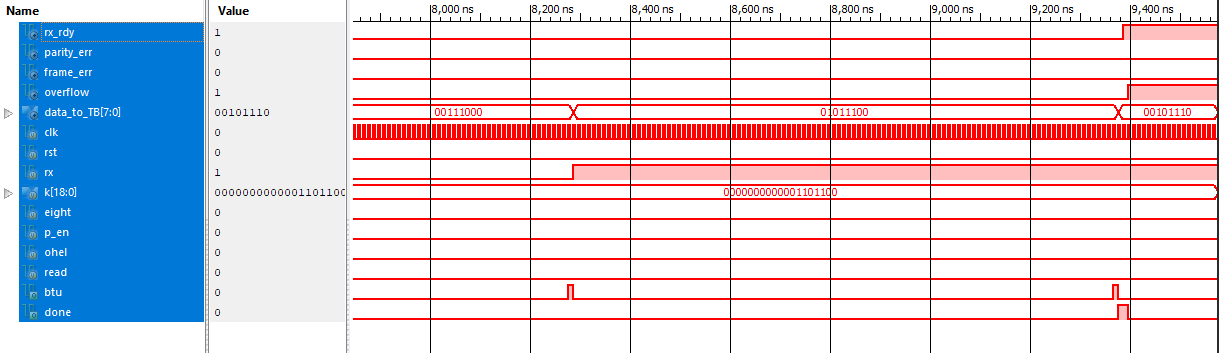


Figure 30: Receive Engine TestFixture

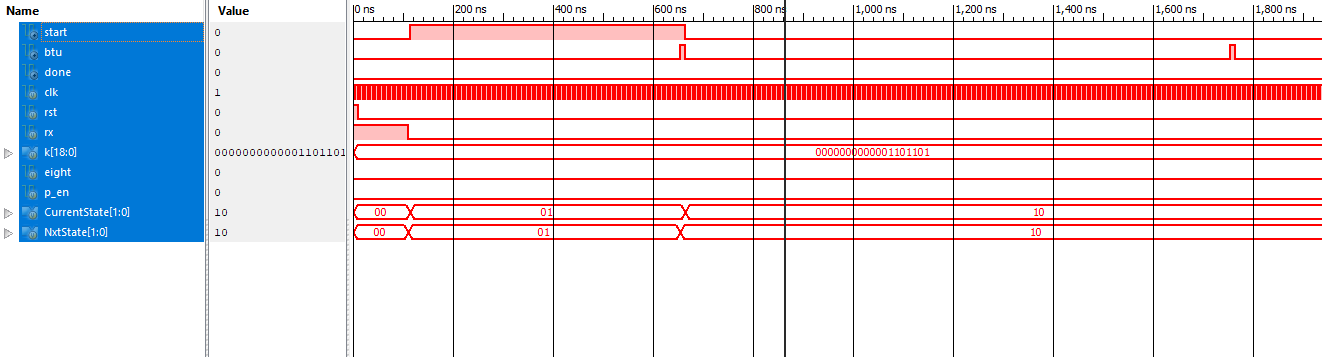
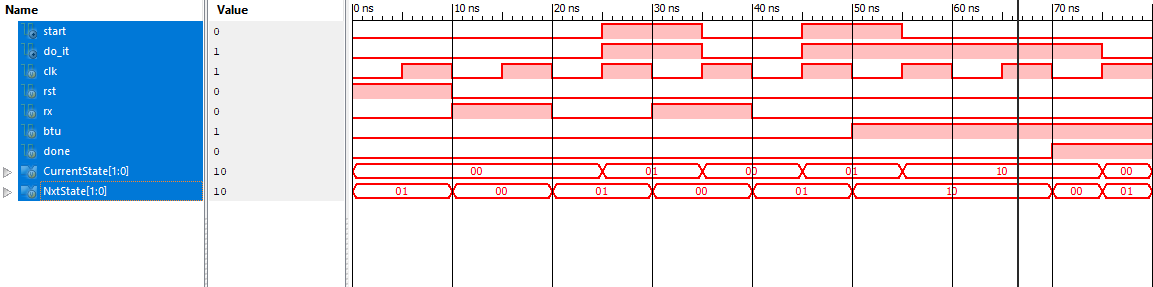


Figure 31: Receive Engine Control Path TestFixture



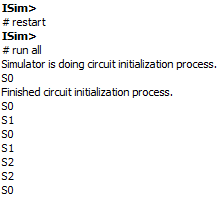


Figure 32: Receive Engine Finite State Machine TestFixture

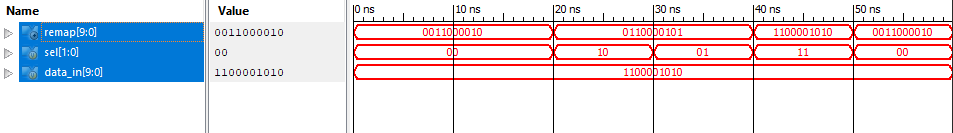


Figure 33: Receive Engine Remap TestFixture

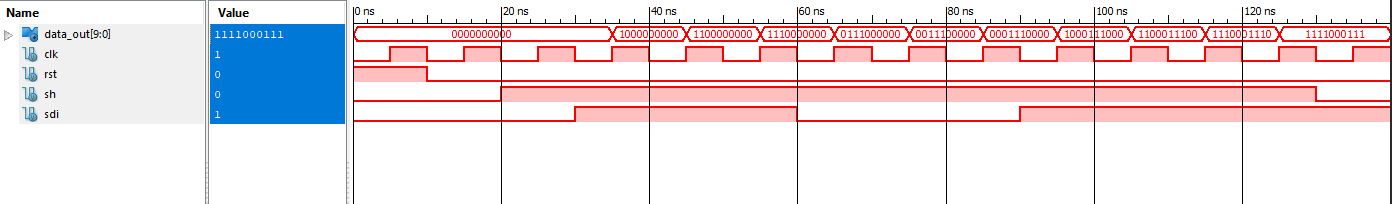


Figure 34: Receive Engine Shift Register TestFixture

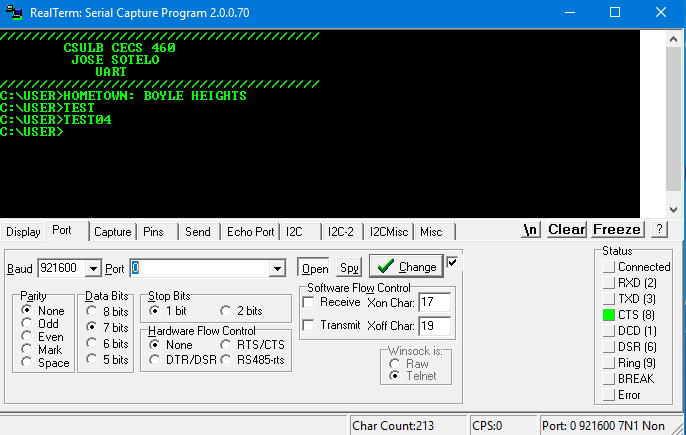


Figure 35: Full UART and TSI RealTerm Output

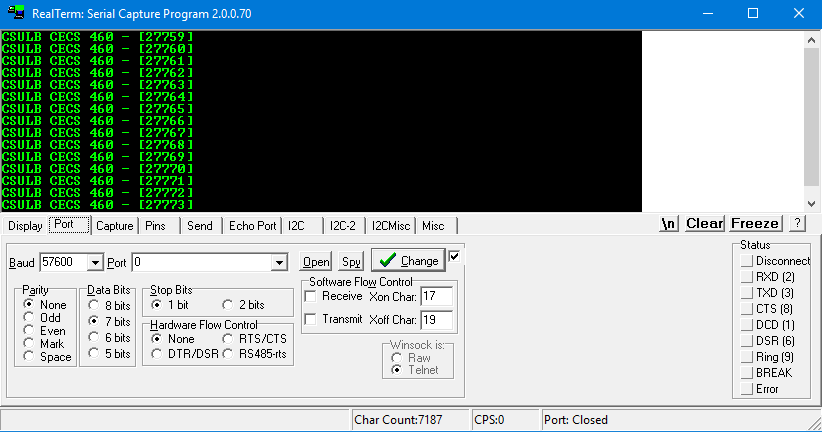


Figure 36: Transmit Engine RealTerm Output

**8. Software Code**

`timescale 1ns **/** 1ps

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

// This document contains information proprietary //

// to the CSULB student that created the //

// file - any reuse without adequate approval and //

// documentation is prohibited //

// //

// Class: <CECS 460 SOC> //

// Project: <Final Project> //

// File name: Top\_Level.v //

// //

// Created by <Jose Sotelo> on <April 16, 2018> //

// //

// In submitting this file for class work at CSULB //

// I am confirming that this is my work and the work //

// of no one else. //

// //

// In the event other code source are utilized I will //

// document which portion of code and who is the author //

// //

// In submitting this code I acknowledge that plagiarism //

// in student project work is subject to dismissal from //

// the class //

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

**module** Top\_Level**(**clk**,** rst**,** baud**,** eight**,** p\_en**,** ohel**,** rx**,** tx**,** leds**);**

**input** clk**,** rst**;**

**input** **[**3**:**0**]** baud**;**

**input** eight**;**

**input** p\_en**;**

**input** ohel**;**

**input** rx**;**

**output** tx**;**

**output** **[**15**:**0**]** leds**;**

**wire** w\_clk**,** w\_rst**;**

**wire** **[**3**:**0**]** w\_baud**;**

**wire** w\_eight**;**

**wire** w\_p\_en**;**

**wire** w\_ohel**;**

**wire** w\_rx**;**

**wire** **[**15**:**0**]** w\_leds**;**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Core Logic

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Core\_Logic

uart\_core**(**

**.**clk**(**w\_clk**),**

**.**rst**(**w\_rst**),**

**.**baud**(**w\_baud**),**

**.**eight**(**w\_eight**),**

**.**p\_en**(**w\_p\_en**),**

**.**ohel**(**w\_ohel**),**

**.**rx**(**w\_rx**),**

**.**tx**(**w\_tx**),**

**.**leds**(**w\_leds**)**

**);**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// TSI

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TSI

tsi\_buf**(**

**.**i\_clk**(**clk**),**

**.**i\_rst**(**rst**),**

**.**i\_baud**(**baud**),**

**.**i\_eight**(**eight**),**

**.**i\_p\_en**(**p\_en**),**

**.**i\_ohel**(**ohel**),**

**.**i\_rx**(**rx**),**

**.**i\_tx**(**w\_tx**),**

**.**i\_leds**(**w\_leds**),**

**.**o\_clk**(**w\_clk**),**

**.**o\_rst**(**w\_rst**),**

**.**o\_baud**(**w\_baud**),**

**.**o\_eight**(**w\_eight**),**

**.**o\_p\_en**(**w\_p\_en**),**

**.**o\_ohel**(**w\_ohel**),**

**.**o\_rx**(**w\_rx**),**

**.**o\_tx**(**tx**),**

**.**o\_leds**(**leds**)**

**);**

**endmodule**

`timescale 1ns **/** 1ps

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

// This document contains information proprietary //

// to the CSULB student that created the //

// file - any reuse without adequate approval and //

// documentation is prohibited //

// //

// Class: <CECS 460 SOC> //

// Project: <Final Project> //

// File name: Core\_Logic.v //

// //

// Created by <Jose Sotelo> on <April 16, 2018> //

// //

// In submitting this file for class work at CSULB //

// I am confirming that this is my work and the work //

// of no one else. //

// //

// In the event other code source are utilized I will //

// document which portion of code and who is the author //

// //

// In submitting this code I acknowledge that plagiarism //

// in student project work is subject to dismissal from //

// the class //

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

**module** Core\_Logic**(**clk**,** rst**,** baud**,** eight**,** p\_en**,** ohel**,** rx**,** tx**,** leds**);**

**input** clk**,** rst**;**

**input** **[**3**:**0**]** baud**;**

**input** eight**,** p\_en**,** ohel**;**

**input** rx**;**

**output** tx**;**

**output** **[**15**:**0**]** leds**;**

**reg** **[**15**:**0**]** leds**;**

**wire** w\_rst\_s**;**

**wire** w\_write\_strobe**;**

**wire** w\_read\_strobe**;**

**wire** w\_int\_ack**;**

**wire** **[**7**:**0**]** w\_uart\_ds**;**

**wire** w\_uart\_int**;**

**wire** w\_interrupt**;**

**wire** **[**15**:**0**]** w\_port\_id**;**

**wire** **[**15**:**0**]** w\_out\_port**;**

**reg** **[**15**:**0**]** w\_write**;**

**reg** **[**15**:**0**]** w\_read**;**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// AISO

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Aiso

aiso**(**

**.**clk**(**clk**),**

**.**rst**(**rst**),**

**.**rst\_s**(**w\_rst\_s**)**

**);**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Address Decode

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**always** **@(\*)**

**begin**

w\_write **=** 16'b0**;**

w\_read **=** 16'b0**;**

w\_write**[**w\_port\_id**]** **=** w\_write\_strobe**;**

w\_read**[**w\_port\_id**]** **=** w\_read\_strobe**;**

**end**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// UART

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

UART\_Top

uart**(**

**.**clk**(**clk**),**

**.**rst**(**w\_rst\_s**),**

**.**write**(**w\_write**[**0**]),**

**.**read**(**w\_read**[**1**:**0**]),**

**.**out\_port**(**w\_out\_port**[**7**:**0**]),**

**.**rx**(**rx**),**

**.**baud**(**baud**),**

**.**eight**(**eight**),**

**.**p\_en**(**p\_en**),**

**.**ohel**(**ohel**),**

**.**tx**(**tx**),**

**.**UART\_DS**(**w\_uart\_ds**),**

**.**UART\_INT**(**w\_uart\_int**)**

**);**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// SR Flop Interrupt

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SR\_Flop

sr\_interrupt**(**

**.**clk**(**clk**),**

**.**rst**(**w\_rst\_s**),**

**.**S**(**w\_uart\_int**),**

**.**R**(**w\_int\_ack**),**

**.**Q**(**w\_interrupt**)**

**);**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// TramelBlaze

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

tramelblaze\_top

tb\_top**(**

**.**CLK**(**clk**),**

**.**RESET**(**w\_rst\_s**),**

**.**IN\_PORT**({**8'b0**,** w\_uart\_ds**}),**

**.**INTERRUPT**(**w\_interrupt**),**

**.**OUT\_PORT**(**w\_out\_port**),**

**.**PORT\_ID**(**w\_port\_id**),**

**.**READ\_STROBE**(**w\_read\_strobe**),**

**.**WRITE\_STROBE**(**w\_write\_strobe**),**

**.**INTERRUPT\_ACK**(**w\_int\_ack**)**

**);**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Walking LEDs

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**always** **@(posedge** clk**,** **posedge** w\_rst\_s**)**

**begin**

**if(**w\_rst\_s**)**

leds **<=** 16'b0**;**

**else** **if(**w\_write**[**2**])**

leds **<=** w\_out\_port**;**

**else**

leds **<=** leds**;**

**end**

**endmodule**

`timescale 1ns **/** 1ps

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

// This document contains information proprietary //

// to the CSULB student that created the //

// file - any reuse without adequate approval and //

// documentation is prohibited //

// //

// Class: <CECS 460 SOC> //

// Project: <Final Project> //

// File name: TSI.v //

// //

// Created by <Jose Sotelo> on <April 16, 2018> //

// //

// In submitting this file for class work at CSULB //

// I am confirming that this is my work and the work //

// of no one else. //

// //

// In the event other code source are utilized I will //

// document which portion of code and who is the author //

// //

// In submitting this code I acknowledge that plagiarism //

// in student project work is subject to dismissal from //

// the class //

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

**module** TSI**(**i\_clk**,** i\_rst**,** i\_baud**,** i\_eight**,** i\_p\_en**,** i\_ohel**,** i\_rx**,** i\_tx**,** i\_leds**,**

o\_clk**,** o\_rst**,** o\_baud**,** o\_eight**,** o\_p\_en**,** o\_ohel**,** o\_rx**,** o\_tx**,** o\_leds**);**

**input** i\_clk**;**

**input** i\_rst**;**

**input** **[**3**:**0**]** i\_baud**;**

**input** i\_eight**;**

**input** i\_p\_en**;**

**input** i\_ohel**;**

**input** i\_rx**;**

**input** i\_tx**;**

**input** **[**15**:**0**]** i\_leds**;**

**output** o\_clk**;**

**output** o\_rst**;**

**output** **[**3**:**0**]** o\_baud**;**

**output** o\_eight**;**

**output** o\_p\_en**;**

**output** o\_ohel**;**

**output** o\_rx**;**

**output** o\_tx**;**

**output** **[**15**:**0**]** o\_leds**;**

// BUFG: Global Clock Simple Buffer

// 7 Series

// Xilinx HDL Libraries Guide, version 2012.2

BUFG

BUFG\_inst **(**

**.**O**(**o\_clk**),** // 1-bit output: Clock output

**.**I**(**i\_clk**)** // 1-bit input: Clock input

**);**

// IOBUF: Single-ended Bi-directional Buffer

// All devices

// Xilinx HDL Libraries Guide, version 2012.2

IBUF **#(**

**.**IBUF\_LOW\_PWR**(**"TRUE"**),** // Low Power - "TRUE", High Perfor = "FALSE"

**.**IOSTANDARD**(**"DEFAULT"**)** // Specify the I/O standard

**)** rst**(**

**.**O**(**o\_rst**),** // Buffer output

**.**I**(**i\_rst**)** // Buffer input

**);**

IBUF **#(**

**.**IBUF\_LOW\_PWR**(**"TRUE"**),** // Low Power - "TRUE", High Perfor = "FALSE"

**.**IOSTANDARD**(**"DEFAULT"**)** // Specify the I/O standard

**)** baud**[**3**:**0**](**

**.**O**(**o\_baud**),** // Buffer output

**.**I**(**i\_baud**)** // Buffer input

**);**

IBUF **#(**

**.**IBUF\_LOW\_PWR**(**"TRUE"**),** // Low Power - "TRUE", High Perfor = "FALSE"

**.**IOSTANDARD**(**"DEFAULT"**)** // Specify the I/O standard

**)** eight**(**

**.**O**(**o\_eight**),** // Buffer output

**.**I**(**i\_eight**)** // Buffer input

**);**

IBUF **#(**

**.**IBUF\_LOW\_PWR**(**"TRUE"**),** // Low Power - "TRUE", High Perfor = "FALSE"

**.**IOSTANDARD**(**"DEFAULT"**)** // Specify the I/O standard

**)** p\_en**(**

**.**O**(**o\_p\_en**),** // Buffer output

**.**I**(**i\_p\_en**)** // Buffer input

**);**

IBUF **#(**

**.**IBUF\_LOW\_PWR**(**"TRUE"**),** // Low Power - "TRUE", High Perfor = "FALSE"

**.**IOSTANDARD**(**"DEFAULT"**)** // Specify the I/O standard

**)** ohel**(**

**.**O**(**o\_ohel**),** // Buffer output

**.**I**(**i\_ohel**)** // Buffer input

**);**

IBUF **#(**

**.**IBUF\_LOW\_PWR**(**"TRUE"**),** // Low Power - "TRUE", High Perfor = "FALSE"

**.**IOSTANDARD**(**"DEFAULT"**)** // Specify the I/O standard

**)** rx**(**

**.**O**(**o\_rx**),** // Buffer output

**.**I**(**i\_rx**)** // Buffer input

**);**

// OBUF: Single-ended Output Buffer

// 7 Series

// Xilinx HDL Libraries Guide, version 2012.2

OBUF **#(**

**.**DRIVE**(**12**),** // Specify the output drive strength

**.**IOSTANDARD**(**"DEFAULT"**),** // Specify the output I/O standard

**.**SLEW**(**"SLOW"**)** // Specify the output slew rate

**)**tx **(**

**.**O**(**o\_tx**),** // Buffer output (connect directly to top-level port)

**.**I**(**i\_tx**)** // Buffer input

**);**

OBUF **#(**

**.**DRIVE**(**12**),** // Specify the output drive strength

**.**IOSTANDARD**(**"DEFAULT"**),** // Specify the output I/O standard

**.**SLEW**(**"SLOW"**)** // Specify the output slew rate

**)**leds**[**15**:**0**]** **(**

**.**O**(**o\_leds**),** // Buffer output (connect directly to top-level port)

**.**I**(**i\_leds**)** // Buffer input

**);**

**endmodule**

`timescale 1ns **/** 1ps

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

// This document contains information proprietary //

// to the CSULB student that created the //

// file - any reuse without adequate approval and //

// documentation is prohibited //

// //

// Class: <CECS 460 SOC> //

// Project: <Project 2> //

// File name: Aiso.v //

// //

// Created by <Jose Sotelo> on <> //

// //

// In submitting this file for class work at CSULB //

// I am confirming that this is my work and the work //

// of no one else. //

// //

// In the event other code source are utilized I will //

// document which portion of code and who is the author //

// //

// In submitting this code I acknowledge that plagiarism //

// in student project work is subject to dismissal from //

// the class //

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

**module** Aiso**(**clk**,** rst**,** rst\_s**);**

**input** clk**;**

**input** rst**;**

**output** rst\_s**;**

**reg** q1**,** q2**;**

**always** **@(posedge** clk**,** **posedge** rst**)**

**if(**rst**)**

**{**q1**,**q2**}** **<=** 2'b0**;**

**else**

**{**q1**,**q2**}** **<=** **{**1'b1**,** q1**};**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Assign Statement

// Modeling combinational logic

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**assign** rst\_s **=** **~**q2**;**

**endmodule**

`timescale 1ns **/** 1ps

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

// This document contains information proprietary //

// to the CSULB student that created the //

// file - any reuse without adequate approval and //

// documentation is prohibited //

// //

// Class: <CECS 460 SOC> //

// Project: <Project 3> //

// File name: UART\_Top.v //

// //

// Created by <Jose Sotelo> on <> //

// //

// In submitting this file for class work at CSULB //

// I am confirming that this is my work and the work //

// of no one else. //

// //

// In the event other code source are utilized I will //

// document which portion of code and who is the author //

// //

// In submitting this code I acknowledge that plagiarism //

// in student project work is subject to dismissal from //

// the class //

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

**module** UART\_Top**(**clk**,** rst**,** write**,** read**,** out\_port**,** rx**,** baud**,**

eight**,** p\_en**,** ohel**,** tx**,** UART\_DS**,** UART\_INT**);**

**input** clk**,** rst**;**

**input** write**;**

**input** **[**1**:**0**]** read**;**

**input** **[**7**:**0**]** out\_port**;**

**input** **[**3**:**0**]** baud**;**

**input** eight**,** p\_en**,** ohel**;**

**input** rx**;**

**output** tx**;**

**output** **[**7**:**0**]** UART\_DS**;**

**output** UART\_INT**;**

**wire** w\_tx\_rdy**;**

**wire** w\_rx\_rdy**;**

**wire** w\_ped\_tx**;**

**wire** w\_ped\_rx**;**

**wire** w\_perr**;**

**wire** w\_ferr**;**

**wire** w\_ovf**;**

**wire** **[**18**:**0**]** k**;**

**wire** **[**7**:**0**]** w\_data\_to\_TB**;**

**wire** **[**7**:**0**]** w\_status**;**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Baud

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Baud\_Dec

baudDec**(**

**.**baud**(**baud**),**

**.**baud\_count**(**k**)**

**);**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Tx Engine

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Tx\_Engine

transmit\_engine**(**

**.**clk**(**clk**),**

**.**rst**(**rst**),**

**.**load**(**write**),**

**.**baud\_count**(**k**),**

**.**data\_in**(**out\_port**),**

**.**eight**(**eight**),**

**.**p\_en**(**p\_en**),**

**.**ohel**(**ohel**),**

**.**txrdy**(**w\_tx\_rdy**),**

**.**tx**(**tx**)**

**);**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Rx Engine

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Rx\_Engine

receive\_engine**(**

**.**clk**(**clk**),**

**.**rst**(**rst**),**

**.**rx**(**rx**),**

**.**k**(**k**),**

**.**eight**(**eight**),**

**.**p\_en**(**p\_en**),**

**.**ohel**(**ohel**),**

**.**read**(**read**[**0**]),**

**.**rx\_rdy**(**w\_rx\_rdy**),**

**.**parity\_err**(**w\_perr**),**

**.**frame\_err**(**w\_ferr**),**

**.**overflow**(**w\_ovf**),**

**.**data\_to\_TB**(**w\_data\_to\_TB**)**

**);**

**assign** w\_status **=** **{**3'b0**,** w\_ovf**,** w\_ferr**,** w\_perr**,** w\_tx\_rdy**,** w\_rx\_rdy**};**

**assign** UART\_DS **=** **(**read**[**1**])** **?** w\_status **:**

**(**read**[**0**])** **?** w\_data\_to\_TB **:**

8'b0**;**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// PED Tx

// PED Rx

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PED

tx\_ped**(**

**.**clk**(**clk**),**

**.**rst**(**rst**),**

**.**ped\_in**(**w\_tx\_rdy**),**

**.**ped\_out**(**w\_ped\_tx**)**

**),**

rx\_ped**(**

**.**clk**(**clk**),**

**.**rst**(**rst**),**

**.**ped\_in**(**w\_rx\_rdy**),**

**.**ped\_out**(**w\_ped\_rx**)**

**);**

**assign** UART\_INT **=** w\_ped\_tx **|** w\_ped\_rx**;** // wire to SR Flop

**endmodule**

`timescale 1ns **/** 1ps

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

// This document contains information proprietary //

// to the CSULB student that created the //

// file - any reuse without adequate approval and //

// documentation is prohibited //

// //

// Class: <CECS 460 SOC> //

// Project: <Project 2> //

// File name: SR\_Flop.v //

// //

// Created by <Jose Sotelo> on <> //

// //

// In submitting this file for class work at CSULB //

// I am confirming that this is my work and the work //

// of no one else. //

// //

// In the event other code source are utilized I will //

// document which portion of code and who is the author //

// //

// In submitting this code I acknowledge that plagiarism //

// in student project work is subject to dismissal from //

// the class //

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

**module** SR\_Flop**(**clk**,** rst**,** S**,** R**,** Q**);**

**input** clk**,** rst**;**

**input** S**,** R**;**

**output** Q**;**

**reg** Q**;**

**always** **@(posedge** clk**,** **posedge** rst**)**

**if(**rst**)**

Q **<=** 1'b0**;**

**else** **if(**S**)**

Q **<=** 1'b1**;**

**else** **if(**R**)**

Q **<=** 1'b0**;**

**else**

Q **<=** Q**;**

**endmodule**

`timescale 1ns **/** 1ps

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

// This document contains information proprietary //

// to the CSULB student that created the //

// file - any reuse without adequate approval and //

// documentation is prohibited //

// //

// Class: <CECS 460 SOC> //

// Project: <Project 2> //

// File name: Baud\_Dec.v //

// //

// Created by <Jose Sotelo> on <> //

// //

// In submitting this file for class work at CSULB //

// I am confirming that this is my work and the work //

// of no one else. //

// //

// In the event other code source are utilized I will //

// document which portion of code and who is the author //

// //

// In submitting this code I acknowledge that plagiarism //

// in student project work is subject to dismissal from //

// the class //

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

**module** Baud\_Dec**(**baud**,** baud\_count**);**

**input** **[**3**:**0**]** baud**;**

**output** **[**18**:**0**]** baud\_count**;**

**reg** **[**18**:**0**]** baud\_count**;**

// Baud Rate

// Calculation (1/baud rate) / (1/100MHz) - 1

**always** **@(\*)**

**case(**baud**)**

4'b0000 **:** baud\_count **<=** 333333 **-** 1**;** // Baud rate 300

4'b0001 **:** baud\_count **<=** 83333 **-** 1**;** // Baud rate 1200

4'b0010 **:** baud\_count **<=** 41667 **-** 1**;** // Baud rate 2400

4'b0011 **:** baud\_count **<=** 20833 **-** 1**;** // Baud rate 4800

4'b0100 **:** baud\_count **<=** 10417 **-** 1**;** // Baud rate 9600

4'b0101 **:** baud\_count **<=** 5208 **-** 1**;** // Baud rate 19200

4'b0110 **:** baud\_count **<=** 2604 **-** 1**;** // Baud rate 38400

4'b0111 **:** baud\_count **<=** 1736 **-** 1**;** // Baud rate 57600

4'b1000 **:** baud\_count **<=** 868 **-** 1**;** // Baud rate 115200

4'b1001 **:** baud\_count **<=** 434 **-** 1**;** // Baud rate 230400

4'b1010 **:** baud\_count **<=** 217 **-** 1**;** // Baud rate 460800

4'b1011 **:** baud\_count **<=** 109 **-** 1**;** // Baud rate 921600

**default** **:** baud\_count **<=** 333333 **-** 1**;** // Default rate 300

**endcase**

**endmodule**

`timescale 1ns **/** 1ps

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

// This document contains information proprietary //

// to the CSULB student that created the //

// file - any reuse without adequate approval and //

// documentation is prohibited //

// //

// Class: <CECS 460 SOC> //

// Project: <Project 3> //

// File name: Tx\_Engine.v //

// //

// Created by <Jose Sotelo> on <> //

// //

// In submitting this file for class work at CSULB //

// I am confirming that this is my work and the work //

// of no one else. //

// //

// In the event other code source are utilized I will //

// document which portion of code and who is the author //

// //

// In submitting this code I acknowledge that plagiarism //

// in student project work is subject to dismissal from //

// the class //

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

**module** Tx\_Engine**(**clk**,** rst**,** load**,** baud\_count**,** data\_in**,**

eight**,** p\_en**,** ohel**,** txrdy**,** tx**);**

**input** clk**,** rst**,** load**;**

**input** **[**18**:**0**]** baud\_count**;**

**input** **[**7**:**0**]** data\_in**;**

**input** eight**,** p\_en**,** ohel**;**

**output** txrdy**;**

**output** tx**;**

**reg** w\_load\_d1**;**

**wire** w\_done**;**

**wire** w\_do\_it**;**

**wire** w\_btu**;**

**wire** **[**7**:**0**]** w\_load\_data**;**

**wire** w\_bit10**,** w\_bit9**;**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// SR Flip Flop TxRdy

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SR\_Flop\_Txrdy

txrdy\_sr\_flop**(**

**.**clk**(**clk**),**

**.**rst**(**rst**),**

**.**S**(**w\_done**),**

**.**R**(**load**),**

**.**Q**(**txrdy**)**

**);**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// SR Flip Flop DoIt

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Tx\_SR\_Flop

doit\_sr\_flop**(**

**.**clk**(**clk**),**

**.**rst**(**rst**),**

**.**S**(**w\_load\_d1**),**

**.**R**(**w\_done**),**

**.**Q**(**w\_do\_it**)**

**);**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// 8-bit Loadable Register

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Tx\_Load\_Reg

tx\_ld\_reg**(**

**.**clk**(**clk**),**

**.**rst**(**rst**),**

**.**ld**(**load**),**

**.**D**(**data\_in**),**

**.**Q**(**w\_load\_data**)**

**);**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Parity Decoder

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Tx\_Parity\_Gen\_Dec

tx\_p\_gen**(**

**.**load\_data**(**w\_load\_data**),**

**.**eight**(**eight**),**

**.**p\_en**(**p\_en**),**

**.**ohel**(**ohel**),**

**.**bit10**(**w\_bit10**),**

**.**bit9**(**w\_bit9**)**

**);**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Register

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**always** **@(posedge** clk**,** **posedge** rst**)**

**if(**rst**)**

w\_load\_d1 **<=** 1'b0**;**

**else**

w\_load\_d1 **<=** load**;**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Shift Register

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Tx\_shift\_register

tx\_shift\_reg**(**

**.**clk**(**clk**),**

**.**rst**(**rst**),**

**.**data\_in**({**w\_bit10**,** w\_bit9**,** w\_load\_data**[**6**:**0**],** 1'b0**,** 1'b1**}),**

**.**load**(**w\_load\_d1**),**

**.**shift**(**w\_btu**),**

**.**sdo**(**tx**)**

**);**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Bit Time Counter

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Tx\_bit\_time\_counter

tx\_btc**(**

**.**clk**(**clk**),**

**.**rst**(**rst**),**

**.**baud\_count**(**baud\_count**),**

**.**sel**({**w\_do\_it**,** w\_btu**}),**

**.**btu**(**w\_btu**)**

**);**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Bit Counter

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Tx\_bit\_counter

tx\_bc**(**

**.**clk**(**clk**),**

**.**rst**(**rst**),**

**.**sel**({**w\_do\_it**,** w\_btu**}),**

**.**done**(**w\_done**)**

**);**

**endmodule**

`timescale 1ns **/** 1ps

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

// This document contains information proprietary //

// to the CSULB student that created the //

// file - any reuse without adequate approval and //

// documentation is prohibited //

// //

// Class: <CECS 460 SOC> //

// Project: <Project 3> //

// File name: Rx\_Engine.v //

// //

// Created by <Jose Sotelo> on <> //

// //

// In submitting this file for class work at CSULB //

// I am confirming that this is my work and the work //

// of no one else. //

// //

// In the event other code source are utilized I will //

// document which portion of code and who is the author //

// //

// In submitting this code I acknowledge that plagiarism //

// in student project work is subject to dismissal from //

// the class //

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

**module** Rx\_Engine**(**clk**,** rst**,** rx**,** k**,** eight**,** p\_en**,** ohel**,** read**,**

rx\_rdy**,** parity\_err**,** frame\_err**,** overflow**,** data\_to\_TB**);**

**input** clk**,** rst**;**

**input** rx**;**

**input** **[**18**:**0**]** k**;**

**input** eight**,** p\_en**,** ohel**;**

**input** read**;**

**output** rx\_rdy**;**

**output** parity\_err**;**

**output** frame\_err**;**

**output** overflow**;**

**output** **[**7**:**0**]** data\_to\_TB**;**

**wire** w\_start**;**

**wire** w\_btu**;**

**wire** w\_done**;**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Rx Engine Control

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Rx\_Engine\_Control

control\_rx**(**

**.**clk**(**clk**),**

**.**rst**(**rst**),**

**.**rx**(**rx**),**

**.**k**(**k**),**

**.**eight**(**eight**),**

**.**p\_en**(**p\_en**),**

**.**start**(**w\_start**),**

**.**btu**(**w\_btu**),**

**.**done**(**w\_done**)**

**);**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Rx Engine Datapath

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Rx\_DataPath

datapath\_rx**(**

**.**clk**(**clk**),**

**.**rst**(**rst**),**

**.**btu**(**w\_btu**),**

**.**start**(**w\_start**),**

**.**rx**(**rx**),**

**.**eight**(**eight**),**

**.**p\_en**(**p\_en**),**

**.**even**(~**ohel**),**

**.**read**(**read**),**

**.**done**(**w\_done**),**

**.**rx\_rdy**(**rx\_rdy**),**

**.**parity\_err**(**parity\_err**),**

**.**frame\_err**(**frame\_err**),**

**.**overflow**(**overflow**),**

**.**data\_to\_TB**(**data\_to\_TB**)**

**);**

**endmodule**

`timescale 1ns **/** 1ps

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

// This document contains information proprietary //

// to the CSULB student that created the //

// file - any reuse without adequate approval and //

// documentation is prohibited //

// //

// Class: <CECS 460 SOC> //

// Project: <Project 2> //

// File name: PED.v //

// //

// Created by <Jose Sotelo> on <> //

// //

// In submitting this file for class work at CSULB //

// I am confirming that this is my work and the work //

// of no one else. //

// //

// In the event other code source are utilized I will //

// document which portion of code and who is the author //

// //

// In submitting this code I acknowledge that plagiarism //

// in student project work is subject to dismissal from //

// the class //

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

**module** PED**(**clk**,** rst**,** ped\_in**,** ped\_out**);**

**input** clk**,** rst**;**

**input** ped\_in**;**

**output** ped\_out**;**

**reg** q1**,** q2**;**

**always** **@(posedge** clk**,** **posedge** rst**)**

**if(**rst**)**

**{**q1**,** q2**}** **<=** 2'b0**;**

**else**

**{**q1**,**q2**}** **<=** **{**ped\_in**,** q1**};**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Assign Statement

// Modeling combinational logic

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**assign** ped\_out **=** **~**q2 **&** q1**;**

**endmodule**

`timescale 1ns **/** 1ps

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

// This document contains information proprietary //

// to the CSULB student that created the //

// file - any reuse without adequate approval and //

// documentation is prohibited //

// //

// Class: <CECS 460 SOC> //

// Project: <Project 2> //

// File name: SR\_Flop\_Txrdy.v //

// //

// Created by <Jose Sotelo> on <> //

// //

// In submitting this file for class work at CSULB //

// I am confirming that this is my work and the work //

// of no one else. //

// //

// In the event other code source are utilized I will //

// document which portion of code and who is the author //

// //

// In submitting this code I acknowledge that plagiarism //

// in student project work is subject to dismissal from //

// the class //

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

**module** SR\_Flop\_Txrdy**(**clk**,** rst**,** S**,** R**,** Q**);**

**input** clk**,** rst**;**

**input** S**,** R**;**

**output** Q**;**

**reg** Q**;**

**always** **@(posedge** clk**,** **posedge** rst**)**

**if(**rst**)**

Q **<=** 1'b1**;**

**else** **if(**S**)**

Q **<=** 1'b1**;**

**else** **if(**R**)**

Q **<=** 1'b0**;**

**else**

Q **<=** Q**;**

**endmodule**

`timescale 1ns **/** 1ps

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

// This document contains information proprietary //

// to the CSULB student that created the //

// file - any reuse without adequate approval and //

// documentation is prohibited //

// //

// Class: <CECS 460 SOC> //

// Project: <Project 2> //

// File name: Tx\_Load\_Reg.v //

// //

// Created by <Jose Sotelo> on <> //

// //

// In submitting this file for class work at CSULB //

// I am confirming that this is my work and the work //

// of no one else. //

// //

// In the event other code source are utilized I will //

// document which portion of code and who is the author //

// //

// In submitting this code I acknowledge that plagiarism //

// in student project work is subject to dismissal from //

// the class //

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

**module** Tx\_Load\_Reg**(**clk**,** rst**,** ld**,** D**,** Q**);**

**input** clk**,** rst**,** ld**;**

**input** **[**7**:**0**]** D**;**

**output** **[**7**:**0**]** Q**;**

**reg** **[**7**:**0**]** Q**;**

**always** **@(posedge** clk**,** **posedge** rst**)**

**if(**rst**)**

**begin**

Q **<=** 8'b0**;**

**end**

**else** **if(**ld**)**

**begin**

Q **<=** D**;**

**end**

**else**

**begin**

Q **<=** Q**;**

**end**

**endmodule**

`timescale 1ns **/** 1ps

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

// This document contains information proprietary //

// to the CSULB student that created the //

// file - any reuse without adequate approval and //

// documentation is prohibited //

// //

// Class: <CECS 460 SOC> //

// Project: <Project 2> //

// File name: Tx\_Parity\_Gen\_Dec.v //

// //

// Created by <Jose Sotelo> on <> //

// //

// In submitting this file for class work at CSULB //

// I am confirming that this is my work and the work //

// of no one else. //

// //

// In the event other code source are utilized I will //

// document which portion of code and who is the author //

// //

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

**module** Tx\_Parity\_Gen\_Dec**(**load\_data**,** eight**,** p\_en**,** ohel**,** bit10**,** bit9**);**

**input** **[**7**:**0**]** load\_data**;**

**input** eight**,** p\_en**,** ohel**;**

**output** bit10**,** bit9**;**

**reg** EP**,** OP**;**

**reg** bit10**,** bit9**;**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// y = f(x)

// Function is used to determine

// Odd/Even parity bits

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**always** **@** **(\*)**

**begin**

EP **<=** **(**eight**)** **?** **^**load\_data**[**7**:**0**]** **:** **^**load\_data**[**6**:**0**];**

OP **<=** **(**eight**)** **?** **~(^**load\_data**[**7**:**0**])** **:** **~(^**load\_data**[**6**:**0**]);**

**end**

**always** **@(\*)**

**case({**eight**,** p\_en**,** ohel**})**

3'b000 **:** **{**bit10**,** bit9**}** **<=** 2'b11**;** // 7N1

3'b001 **:** **{**bit10**,** bit9**}** **<=** 2'b11**;** // 7N1

3'b010 **:** **{**bit10**,** bit9**}** **<=** **{**1'b1**,** EP**};** // 7E1

3'b011 **:** **{**bit10**,** bit9**}** **<=** **{**1'b1**,** OP**};** // 7O1

3'b100 **:** **{**bit10**,** bit9**}** **<=** **{**1'b1**,** load\_data**[**7**]};** // 8N1

3'b101 **:** **{**bit10**,** bit9**}** **<=** **{**1'b1**,** load\_data**[**7**]};** // 8N1

3'b110 **:** **{**bit10**,** bit9**}** **<=** **{**EP**,** load\_data**[**7**]};** // 8E1

3'b111 **:** **{**bit10**,** bit9**}** **<=** **{**OP**,** load\_data**[**7**]};** // 8O1

**endcase**

**endmodule**

`timescale 1ns **/** 1ps

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

// This document contains information proprietary //

// to the CSULB student that created the //

// file - any reuse without adequate approval and //

// documentation is prohibited //

// //

// Class: <CECS 460 SOC> //

// Project: <Project 2> //

// File name: Tx\_shift\_register .v //

// //

// Created by <Jose Sotelo> on <> //

// //

// In submitting this file for class work at CSULB //

// I am confirming that this is my work and the work //

// of no one else. //

// //

// In the event other code source are utilized I will //

// document which portion of code and who is the author //

// //

// In submitting this code I acknowledge that plagiarism //

// in student project work is subject to dismissal from //

// the class //

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

**module** Tx\_shift\_register**(**clk**,** rst**,** data\_in**,** load**,** shift**,** sdo**);**

**input** clk**,** rst**;**

**input** **[**10**:**0**]** data\_in**;**

**input** load**;**

**input** shift**;**

**output** sdo**;**

**reg** **[**10**:**0**]** shift\_reg**;**

**always** **@(posedge** clk**,** **posedge** rst**)**

**if(**rst**)**

shift\_reg **<=** 11'b11111\_111111**;**

**else** **if(**load**)**

shift\_reg **<=** data\_in**;**

**else** **if(**shift**)**

shift\_reg **<=** **{**1'b1**,** shift\_reg**[**10**:**1**]};**

**else**

shift\_reg **<=** shift\_reg**;**

**assign** sdo **=** shift\_reg**[**0**];**

**endmodule**

`timescale 1ns **/** 1ps

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

// This document contains information proprietary //

// to the CSULB student that created the //

// file - any reuse without adequate approval and //

// documentation is prohibited //

// //

// Class: <CECS 460 SOC> //

// Project: <Project 2> //

// File name: Tx\_bit\_time\_counter.v //

// //

// Created by <Jose Sotelo> on <> //

// //

// In submitting this file for class work at CSULB //

// I am confirming that this is my work and the work //

// of no one else. //

// //

// In the event other code source are utilized I will //

// document which portion of code and who is the author //

// //

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

**module** Tx\_bit\_time\_counter**(**clk**,** rst**,** baud\_count**,** sel**,** btu**);**

**input** clk**,** rst**;**

**input** **[**18**:**0**]** baud\_count**;**

**input** **[**1**:**0**]** sel**;** // concat 1-bit do\_it & btu wires

**output** btu**;**

**reg** **[**18**:**0**]** mux\_out**;**

**reg** **[**18**:**0**]** bit\_time\_count**;**

// MUX Selector

**always** **@(\*)**

**case(**sel**)**

2'b00 **:** mux\_out **<=** 19'b0**;**

2'b01 **:** mux\_out **<=** 19'b0**;**

2'b10 **:** mux\_out **<=** bit\_time\_count **+** 19'b1**;**

2'b11 **:** mux\_out **<=** 19'b0**;**

**default** **:** mux\_out **<=** 19'b0**;**

**endcase**

// 19 bit Register

**always** **@(posedge** clk**,** **posedge** rst**)**

**if(**rst**)**

bit\_time\_count **<=** 19'b0**;**

**else**

bit\_time\_count **<=** mux\_out**;**

// Comparator

**assign** btu **=** **(**bit\_time\_count **==** baud\_count**)** **?** 1'b1 **:** 1'b0**;**

**endmodule**

`timescale 1ns **/** 1ps

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

// This document contains information proprietary //

// to the CSULB student that created the //

// file - any reuse without adequate approval and //

// documentation is prohibited //

// //

// Class: <CECS 460 SOC> //

// Project: <Project 2> //

// File name: Tx\_bit\_counter.v //

// //

// Created by <Jose Sotelo> on <> //

// //

// In submitting this file for class work at CSULB //

// I am confirming that this is my work and the work //

// of no one else. //

// //

// In the event other code source are utilized I will //

// document which portion of code and who is the author //

// //

// In submitting this code I acknowledge that plagiarism //

// in student project work is subject to dismissal from //

// the class //

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

**module** Tx\_bit\_counter**(**clk**,** rst**,** sel**,** done**);**

**input** clk**,** rst**;**

**input** **[**1**:**0**]** sel**;**

**output** done**;**

**reg** **[**3**:**0**]** mux\_out**;**

**reg** **[**3**:**0**]** bit\_count**;**

// MUX Select

**always** **@(\*)**

**case(**sel**)**

2'b00 **:** mux\_out **<=** 4'b0**;**

2'b01 **:** mux\_out **<=** 4'b0**;**

2'b10 **:** mux\_out **<=** bit\_count**;**

2'b11 **:** mux\_out **<=** bit\_count **+** 4'b1**;**

**endcase**

// 4-bit Register

**always** **@(posedge** clk**,** **posedge** rst**)**

**if(**rst**)**

bit\_count **<=** 4'b0**;**

**else**

bit\_count **<=** mux\_out**;**

**assign** done **=** **(**bit\_count **==** 11**)** **?** 1'b1 **:** 1'b0**;**

**endmodule**

`timescale 1ns **/** 1ps

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

// This document contains information proprietary //

// to the CSULB student that created the //

// file - any reuse without adequate approval and //

// documentation is prohibited //

// //

// Class: <CECS 460 SOC> //

// Project: <Project 3> //

// File name: Rx\_Engine\_Control.v //

// //

// Created by <Jose Sotelo> on <> //

// //

// In submitting this file for class work at CSULB //

// I am confirming that this is my work and the work //

// of no one else. //

// //

// In the event other code source are utilized I will //

// document which portion of code and who is the author //

// //

// In submitting this code I acknowledge that plagiarism //

// in student project work is subject to dismissal from //

// the class //

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

**module** Rx\_Engine\_Control**(**clk**,** rst**,** rx**,** k**,** eight**,** p\_en**,**

start**,** btu**,** done**);**

**input** clk**,** rst**;**

**input** rx**;**

**input** **[**18**:**0**]** k**;**

**input** eight**;**

**input** p\_en**;**

**output** start**;**

**output** btu**;**

**output** done**;**

**wire** w\_do\_it**;**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Rx FSM

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Rx\_FSM

rx\_fsm**(**

**.**clk**(**clk**),**

**.**rst**(**rst**),**

**.**rx**(**rx**),**

**.**btu**(**btu**),**

**.**done**(**done**),**

**.**start**(**start**),**

**.**do\_it**(**w\_do\_it**)**

**);**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Rx Bit Time Counter

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Rx\_bit\_time\_counter

rx\_btc**(**

**.**clk**(**clk**),**

**.**rst**(**rst**),**

**.**start**(**start**),**

**.**k**(**k**),**

**.**sel**({**w\_do\_it**,** btu**}),**

**.**btu**(**btu**)**

**);**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Rx Bit Counter

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Rx\_bit\_counter

rx\_bc**(**

**.**clk**(**clk**),**

**.**rst**(**rst**),**

**.**sel**({**w\_do\_it**,** btu**}),**

**.**eight**(**eight**),**

**.**p\_en**(**p\_en**),**

**.**done**(**done**)**

**);**

**endmodule**

`timescale 1ns **/** 1ps

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

// This document contains information proprietary //

// to the CSULB student that created the //

// file - any reuse without adequate approval and //

// documentation is prohibited //

// //

// Class: <CECS 460 SOC> //

// Project: <Project 3> //

// File name: Rx\_FSM.v //

// //

// Created by <Jose Sotelo> on <> //

// //

// In submitting this file for class work at CSULB //

// I am confirming that this is my work and the work //

// of no one else. //

// //

// In the event other code source are utilized I will //

// document which portion of code and who is the author //

// //

// In submitting this code I acknowledge that plagiarism //

// in student project work is subject to dismissal from //

// the class //

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

**module** Rx\_FSM**(**clk**,** rst**,** rx**,** btu**,** done**,** start**,** do\_it**);**

**input** clk**,** rst**;**

**input** rx**,** btu**,** done**;**

**output** start**,** do\_it**;**

**reg** start**,** do\_it**;**

**reg** **[**1**:**0**]** CurrentState**,** NxtState**;**

**reg** NxtStart**,** NxtDo\_It**;**

**parameter** s0 **=** 2'b00**,**

s1 **=** 2'b01**,**

s2 **=** 2'b10**;**

**always** **@(posedge** clk**,** **posedge** rst**)**

**begin**

**if(**rst**)**

**{**CurrentState**,** start**,** do\_it**}** **<=** **{**s0**,** 2'b00**};**

**else**

**{**CurrentState**,** start**,** do\_it**}** **<=** **{**NxtState**,** NxtStart**,** NxtDo\_It**};**

**end**

**always** **@(\*)**

**begin**

NxtStart **=** 0**;**

NxtDo\_It **=** 0**;**

NxtState **=** CurrentState**;**

**case(**CurrentState**)**

s0**:**

**begin**

**{**NxtState**,** NxtStart**,** NxtDo\_It**}** **=** **(**rx**)** **?** **{**s0**,** 2'b00**}** **:**

**{**s1**,** 2'b11**};**

**end**

s1**:**

**begin**

**{**NxtState**,** NxtStart**,** NxtDo\_It**}** **=**

**(**rx**)** **?** **{**s0**,** 2'b00**}** **:**

**(~**rx **&&** **~**btu**)** **?** **{**s1**,** 2'b11**}** **:**

**{**s2**,** 2'b01**};**

**end**

s2**:**

**begin**

**{**NxtState**,** NxtStart**,** NxtDo\_It**}** **=** **(~**done**)** **?** **{**s2**,** 2'b01**}** **:**

**{**s0**,** 2'b00**};**

**end**

**default** **:** **{**NxtState**,** NxtStart**,** NxtDo\_It**}** **=** **{**s0**,** 2'b00**};**

**endcase**

**end**

**endmodule**

`timescale 1ns **/** 1ps

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

// This document contains information proprietary //

// to the CSULB student that created the //

// file - any reuse without adequate approval and //

// documentation is prohibited //

// //

// Class: <CECS 460 SOC> //

// Project: <Project 3> //

// File name: Rx\_bit\_time\_counter.v //

// //

// Created by <Jose Sotelo> on <> //

// //

// In submitting this file for class work at CSULB //

// I am confirming that this is my work and the work //

// of no one else. //

// In the event other code source are utilized I will //

// document which portion of code and who is the author //

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

**module** Rx\_bit\_time\_counter**(**clk**,** rst**,** start**,** k**,** sel**,** btu**);**

**input** clk**,** rst**;**

**input** start**;**

**input** **[**18**:**0**]** k**;**

**input** **[**1**:**0**]** sel**;** // concat 1-bit do\_it & btu wires

**output** btu**;**

**reg** **[**18**:**0**]** mux\_out**;**

**reg** **[**18**:**0**]** bit\_time\_count**;**

**wire** **[**18**:**0**]** mux\_baud\_count\_out**;**

// MUX Selector

**always** **@(\*)**

**case(**sel**)**

2'b00 **:** mux\_out **<=** 19'b0**;**

2'b01 **:** mux\_out **<=** 19'b0**;**

2'b10 **:** mux\_out **<=** bit\_time\_count **+** 19'b1**;**

2'b11 **:** mux\_out **<=** 19'b0**;**

**default** **:** mux\_out **<=** 19'b0**;**

**endcase**

// 19 bit Register

**always** **@(posedge** clk**,** **posedge** rst**)**

**if(**rst**)**

bit\_time\_count **<=** 19'b0**;**

**else**

bit\_time\_count **<=** mux\_out**;**

// Comparator

**assign** btu **=** **(**bit\_time\_count **==** mux\_baud\_count\_out**)** **?** 1'b1 **:** 1'b0**;**

// Buad Mux Select

**assign** mux\_baud\_count\_out **=** **(**start**)** **?** **(**k **>>** 1**)** **:** **(**k**);**

**endmodule**

`timescale 1ns **/** 1ps

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

// This document contains information proprietary //

// to the CSULB student that created the //

// file - any reuse without adequate approval and //

// documentation is prohibited //

// //

// Class: <CECS 460 SOC> //

// Project: <Project 3> //

// File name: Rx\_bit\_counter.v //

// //

// Created by <Jose Sotelo> on <> //

// //

// In submitting this file for class work at CSULB //

// I am confirming that this is my work and the work //

// of no one else. //

// //

// In the event other code source are utilized I will //

// document which portion of code and who is the author //

// //

// In submitting this code I acknowledge that plagiarism //

// in student project work is subject to dismissal from //

// the class //

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

**module** Rx\_bit\_counter**(**clk**,** rst**,** sel**,** eight**,** p\_en**,** done**);**

**input** clk**,** rst**;**

**input** **[**1**:**0**]** sel**;**

**input** eight**,** p\_en**;**

**output** done**;**

**reg** **[**3**:**0**]** mux\_out**;**

**reg** **[**3**:**0**]** bit\_count**;**

**reg** **[**3**:**0**]** num\_bits**;**

// MUX Select

**always** **@(\*)**

**begin**

**case(**sel**)**

2'b00 **:** mux\_out **<=** 4'b0**;**

2'b01 **:** mux\_out **<=** 4'b0**;**

2'b10 **:** mux\_out **<=** bit\_count**;**

2'b11 **:** mux\_out **<=** bit\_count **+** 4'b1**;**

**default** **:** mux\_out **<=** 4'b0**;**

**endcase**

**end**

// 4-bit Register

**always** **@(posedge** clk**,** **posedge** rst**)**

**if(**rst**)**

bit\_count **<=** 4'b0**;**

**else**

bit\_count **<=** mux\_out**;**

// f(x) used to determine the bits

**always** **@(\*)**

**begin**

**case({**eight**,** p\_en**})**

2'b00 **:** num\_bits **=** 4'h9**;** // 7N1

2'b01 **:** num\_bits **=** 4'hA**;** // 7E1/7O1

2'b10 **:** num\_bits **=** 4'hA**;** // 8N1

2'b11 **:** num\_bits **=** 4'hB**;** // 8O1/8E1

**default** **:** num\_bits **=** 4'hA**;** // 7N1

**endcase**

**end**

// Comparator

**assign** done **=** **(**bit\_count **==** num\_bits**)** **?** 1'b1 **:** 1'b0**;**

**endmodule**

`timescale 1ns **/** 1ps

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

// This document contains information proprietary //

// to the CSULB student that created the //

// file - any reuse without adequate approval and //

// documentation is prohibited //

// //

// Class: <CECS 460 SOC> //

// Project: <Project 2> //

// File name: Rx\_DataPath.v //

// //

// Created by <Jose Sotelo> on <> //

// //

// In submitting this file for class work at CSULB //

// I am confirming that this is my work and the work //

// of no one else. //

// //

// In the event other code source are utilized I will //

// document which portion of code and who is the author //

// //

// In submitting this code I acknowledge that plagiarism //

// in student project work is subject to dismissal from //

// the class //

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

**module** Rx\_DataPath**(**clk**,** rst**,** btu**,** start**,** rx**,**

eight**,** p\_en**,** even**,** done**,** read**,**

rx\_rdy**,** parity\_err**,** frame\_err**,** overflow**,** data\_to\_TB**);**

**input** clk**,** rst**;**

**input** btu**,** start**,** rx**;**

**input** eight**,** p\_en**;**

**input** even**,** done**;**

**input** read**;**

**output** rx\_rdy**;**

**output** parity\_err**;**

**output** frame\_err**;**

**output** overflow**;**

**output** **[**7**:**0**]** data\_to\_TB**;**

**reg** stop\_bit\_mux\_out**;**

**wire** **[**9**:**0**]** w\_shift\_reg\_out**;**

**wire** **[**9**:**0**]** w\_remap**;**

**wire** w\_p\_gen\_mux**;**

**wire** w\_p\_gen\_even\_mux**;**

**wire** w\_p\_bit\_mux**;**

**wire** w\_perr**;**

**wire** w\_ferr**;**

**wire** w\_ovf**;**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Rx Shift Register

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Rx\_Shift\_Reg

rx\_shift\_reg**(**

**.**clk**(**clk**),**

**.**rst**(**rst**),**

**.**sh**(**btu **&** **~**start**),**

**.**sdi**(**rx**),**

**.**data\_out**(**w\_shift\_reg\_out**)**

**);**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Rx ReMap

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Rx\_Remap

rx\_remap**(**

**.**sel**({**eight**,** p\_en**}),**

**.**data\_in**(**w\_shift\_reg\_out**),**

**.**remap**(**w\_remap**)**

**);**

**assign** w\_p\_gen\_mux **=** **(**eight**)** **?** w\_remap**[**7**]** **:** 1'b0**;**

**assign** w\_p\_gen\_even\_mux **=** **(**even**)** **?** **(^{**w\_p\_gen\_mux**,** w\_remap**[**6**:**0**]})** **:**

**~(^{**w\_p\_gen\_mux**,** w\_remap**[**6**:**0**]});**

**assign** w\_p\_bit\_mux **=** **(**eight**)** **?** w\_remap**[**8**]** **:** w\_remap**[**7**];**

**assign** w\_perr **=** p\_en **&** **(^{**w\_p\_gen\_even\_mux**,** w\_p\_bit\_mux**})** **&** done**;**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Stop Bit Select MUX

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**always** **@(\*)**

**case({**eight**,** p\_en**})**

2'b00 **:** stop\_bit\_mux\_out **<=** w\_remap**[**7**];**

2'b01 **:** stop\_bit\_mux\_out **<=** w\_remap**[**8**];**

2'b10 **:** stop\_bit\_mux\_out **<=** w\_remap**[**8**];**

2'b11 **:** stop\_bit\_mux\_out **<=** w\_remap**[**9**];**

**default** **:** stop\_bit\_mux\_out **<=** w\_remap**[**7**];**

**endcase**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// SR Rx Ready Flop

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Rx\_SR\_Flop

rx\_ready**(**

**.**clk**(**clk**),**

**.**rst**(**rst**),**

**.**S**(**done**),**

**.**R**(**read**),**

**.**Q**(**rx\_rdy**)**

**);**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// SR Rx Parity Error

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Rx\_SR\_Flop

p\_err**(**

**.**clk**(**clk**),**

**.**rst**(**rst**),**

**.**S**(**w\_perr**),**

**.**R**(**read**),**

**.**Q**(**parity\_err**)**

**);**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// SR Rx Framing Error

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Rx\_SR\_Flop

framing\_err**(**

**.**clk**(**clk**),**

**.**rst**(**rst**),**

**.**S**(**done **&** **~**stop\_bit\_mux\_out**),**

**.**R**(**read**),**

**.**Q**(**frame\_err**)**

**);**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// SR Rx Overflow Error

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Rx\_SR\_Flop

ovf\_err**(**

**.**clk**(**clk**),**

**.**rst**(**rst**),**

**.**S**(**done **&** rx\_rdy**),**

**.**R**(**read**),**

**.**Q**(**overflow**)**

**);**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Data sent to TramelBlaze

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**assign** data\_to\_TB **=** **(**eight**)** **?** w\_remap**[**7**:**0**]** **:** **{**1'b0**,** w\_remap**[**6**:**0**]};**

**endmodule**

`timescale 1ns **/** 1ps

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

// This document contains information proprietary //

// to the CSULB student that created the //

// file - any reuse without adequate approval and //

// documentation is prohibited //

// //

// Class: <CECS 460 SOC> //

// Project: <Project 3> //

// File name: Rx\_Shift\_Reg.v //

// //

// Created by <Jose Sotelo> on <> //

// //

// In submitting this file for class work at CSULB //

// I am confirming that this is my work and the work //

// of no one else. //

// //

// In the event other code source are utilized I will //

// document which portion of code and who is the author //

// //

// In submitting this code I acknowledge that plagiarism //

// in student project work is subject to dismissal from //

// the class //

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

**module** Rx\_Shift\_Reg**(**clk**,** rst**,** sh**,** sdi**,** data\_out**);**

**input** clk**,** rst**;**

**input** sh**;**

**input** sdi**;** // Rx input

**output** **[**9**:**0**]** data\_out**;**

**reg** **[**9**:**0**]** data\_out**;**

**always** **@(posedge** clk**,** **posedge** rst**)**

**if(**rst**)**

data\_out **<=** 10'b0**;**

**else** **if(**sh**)**

data\_out **<=** **{**sdi**,** data\_out**[**9**:**1**]};**

**else**

data\_out **<=** data\_out**;**

**endmodule**

`timescale 1ns **/** 1ps

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

// This document contains information proprietary //

// to the CSULB student that created the //

// file - any reuse without adequate approval and //

// documentation is prohibited //

// //

// Class: <CECS 460 SOC> //

// Project: <Project 3> //

// File name: Rx\_Remap.v //

// //

// Created by <Jose Sotelo> on <> //

// //

// In submitting this file for class work at CSULB //

// I am confirming that this is my work and the work //

// of no one else. //

// //

// In the event other code source are utilized I will //

// document which portion of code and who is the author //

// //

// In submitting this code I acknowledge that plagiarism //

// in student project work is subject to dismissal from //

// the class //

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

**module** Rx\_Remap**(**sel**,** data\_in**,** remap**);**

**input** **[**1**:**0**]** sel**;** // eight and p\_en

**input** **[**9**:**0**]** data\_in**;** // shift reg data

**output** **[**9**:**0**]** remap**;** // remap combo

**reg** **[**9**:**0**]** remap**;**

**always** **@(\*)**

**case(**sel**)**

2'b00 **:** remap **<=** **{**2'b00**,** data\_in**[**9**:**2**]};**

2'b01 **:** remap **<=** **{**1'b0**,** data\_in**[**9**:**1**]};**

2'b10 **:** remap **<=** **{**1'b0**,** data\_in**[**9**:**1**]};**

2'b11 **:** remap **<=** data\_in**;**

**default** **:** remap **<=** **{**2'b00**,** data\_in**[**9**:**2**]};**

**endcase**

**endmodule**

; Jose Sotelo

; CECS 460

; Full duplex UART and TramelBlaze Assembly

;=======================================================

; Define Variables Alias

;=======================================================

ZERO EQU 0000

ONE EQU 0001

PORT EQU 0001

LED\_PORT EQU 0005

RX\_RDY EQU 0001

TX\_RDY EQU 0002

ASCII\_BS EQU 0008

ASCII\_TAB EQU 0009

ASCII\_LF EQU 000A

ASCII\_CR EQU 000D

ASCII\_SLASH EQU 002F

ASCII\_EQUALS EQU 003D

ASCII\_ASTERISK EQU 002A

ASCII\_AT EQU 0040

ASCII\_DOT EQU 002E

ASCII\_SPACE EQU 0020

ASCII\_GREATER EQU 003E

ASCII\_COLON EQU 003A

ASCII\_B\_SLASH EQU 005C

ASCII\_ZERO EQU 0030

ASCII\_ONE EQU 0031

ASCII\_TWO EQU 0032

ASCII\_THREE EQU 0033

ASCII\_FOUR EQU 0034

ASCII\_FIVE EQU 0035

ASCII\_SIX EQU 0036

ASCII\_SEVEN EQU 0037

ASCII\_EIGHT EQU 0038

ASCII\_NINE EQU 0039

ASCII\_A EQU 0041

ASCII\_B EQU 0042

ASCII\_C EQU 0043

ASCII\_D EQU 0044

ASCII\_E EQU 0045

ASCII\_F EQU 0046

ASCII\_G EQU 0047

ASCII\_H EQU 0048

ASCII\_I EQU 0049

ASCII\_J EQU 004A

ASCII\_K EQU 004B

ASCII\_L EQU 004C

ASCII\_M EQU 004D

ASCII\_N EQU 004E

ASCII\_O EQU 004F

ASCII\_P EQU 0050

ASCII\_Q EQU 0051

ASCII\_R EQU 0052

ASCII\_S EQU 0053

ASCII\_T EQU 0054

ASCII\_U EQU 0055

ASCII\_V EQU 0056

ASCII\_W EQU 0057

ASCII\_X EQU 0058

ASCII\_Y EQU 0059

ASCII\_Z EQU 005A

;=======================================================

; Register Alias

;=======================================================

TEMP\_REG EQU R0

POINTER EQU R1

COUNTER EQU R2

COUNT EQU R3

UART\_STATUS EQU R4

CURRENT\_CASE EQU R5

CHAR\_COUNT EQU R6

STORE\_DATA EQU R7

;=======================================================

; Start

; Startup Code Initialization

; only executed at startup (reset)

;=======================================================

START

LOAD TEMP\_REG**,** ZERO

LOAD POINTER**,** ZERO

LOAD COUNTER**,** ZERO

LOAD COUNT**,** ZERO

LOAD UART\_STATUS**,** ZERO

LOAD CURRENT\_CASE**,** ONE

**CALL** BANNER

**CALL** PROMPT\_USER

**CALL** TOWN

**CALL** BACKSPACE

**CALL** CRLF

ENINT

JUMP MAIN

;=======================================================

; Banner Subroutine

; Banner is executed on startup

; /////////////////////////////////

; CSULB CECS 460

; Jose Sotelo

; /////////////////////////////////

;=======================================================

BANNER

LOAD TEMP\_REG**,** ASCII\_SLASH

ADD\_SLASH STORE TEMP\_REG**,** COUNTER

**ADD** COUNTER**,** 0001

COMP COUNTER**,** 0028

JUMPC ADD\_SLASH

LOAD TEMP\_REG**,** ASCII\_CR

STORE TEMP\_REG**,** 0029

LOAD TEMP\_REG**,** ASCII\_LF

STORE TEMP\_REG**,** 002A

LOAD TEMP\_REG**,** ASCII\_TAB

STORE TEMP\_REG**,** 002B

LOAD TEMP\_REG**,** ASCII\_C

STORE TEMP\_REG**,** 002C

LOAD TEMP\_REG**,** ASCII\_S

STORE TEMP\_REG**,** 002D

LOAD TEMP\_REG**,** ASCII\_U

STORE TEMP\_REG**,** 002E

LOAD TEMP\_REG**,** ASCII\_L

STORE TEMP\_REG**,** 002F

LOAD TEMP\_REG**,** ASCII\_B

STORE TEMP\_REG**,** 0030

LOAD TEMP\_REG**,** ASCII\_SPACE

STORE TEMP\_REG**,** 0031

LOAD TEMP\_REG**,** ASCII\_C

STORE TEMP\_REG**,** 0032

LOAD TEMP\_REG**,** ASCII\_E

STORE TEMP\_REG**,** 0033

LOAD TEMP\_REG**,** ASCII\_C

STORE TEMP\_REG**,** 0034

LOAD TEMP\_REG**,** ASCII\_S

STORE TEMP\_REG**,** 0035

LOAD TEMP\_REG**,** ASCII\_SPACE

STORE TEMP\_REG**,** 0036

LOAD TEMP\_REG**,** ASCII\_FOUR

STORE TEMP\_REG**,** 0037

LOAD TEMP\_REG**,** ASCII\_SIX

STORE TEMP\_REG**,** 0038

LOAD TEMP\_REG**,** ASCII\_ZERO

STORE TEMP\_REG**,** 0039

LOAD TEMP\_REG**,** ASCII\_CR

STORE TEMP\_REG**,** 003A

LOAD TEMP\_REG**,** ASCII\_LF

STORE TEMP\_REG**,** 003B

LOAD TEMP\_REG**,** ASCII\_TAB

STORE TEMP\_REG**,** 003C

LOAD TEMP\_REG**,** ASCII\_SPACE

STORE TEMP\_REG**,** 003D

LOAD TEMP\_REG**,** ASCII\_J

STORE TEMP\_REG**,** 003E

LOAD TEMP\_REG**,** ASCII\_O

STORE TEMP\_REG**,** 003F

LOAD TEMP\_REG**,** ASCII\_S

STORE TEMP\_REG**,** 0040

LOAD TEMP\_REG**,** ASCII\_E

STORE TEMP\_REG**,** 0041

LOAD TEMP\_REG**,** ASCII\_SPACE

STORE TEMP\_REG**,** 0042

LOAD TEMP\_REG**,** ASCII\_S

STORE TEMP\_REG**,** 0043

LOAD TEMP\_REG**,** ASCII\_O

STORE TEMP\_REG**,** 0044

LOAD TEMP\_REG**,** ASCII\_T

STORE TEMP\_REG**,** 0045

LOAD TEMP\_REG**,** ASCII\_E

STORE TEMP\_REG**,** 0046

LOAD TEMP\_REG**,** ASCII\_L

STORE TEMP\_REG**,** 0047

LOAD TEMP\_REG**,** ASCII\_O

STORE TEMP\_REG**,** 0048

LOAD TEMP\_REG**,** ASCII\_CR

STORE TEMP\_REG**,** 0049

LOAD TEMP\_REG**,** ASCII\_LF

STORE TEMP\_REG**,** 004A

LOAD TEMP\_REG**,** ASCII\_TAB

STORE TEMP\_REG**,** 004B

LOAD TEMP\_REG**,** ASCII\_SPACE

STORE TEMP\_REG**,** 004C

LOAD TEMP\_REG**,** ASCII\_SPACE

STORE TEMP\_REG**,** 004D

LOAD TEMP\_REG**,** ASCII\_SPACE

STORE TEMP\_REG**,** 004E

LOAD TEMP\_REG**,** ASCII\_SPACE

STORE TEMP\_REG**,** 004F

LOAD TEMP\_REG**,** ASCII\_U

STORE TEMP\_REG**,** 0050

LOAD TEMP\_REG**,** ASCII\_A

STORE TEMP\_REG**,** 0051

LOAD TEMP\_REG**,** ASCII\_R

STORE TEMP\_REG**,** 0052

LOAD TEMP\_REG**,** ASCII\_T

STORE TEMP\_REG**,** 0053

LOAD TEMP\_REG**,** ASCII\_CR

STORE TEMP\_REG**,** 0054

LOAD TEMP\_REG**,** ASCII\_LF

STORE TEMP\_REG**,** 0055

LOAD TEMP\_REG**,** ASCII\_SLASH

LOAD COUNT**,** 0056

LAST STORE TEMP\_REG**,** COUNT

**ADD** COUNT**,** 0001

COMP COUNT**,** 007E

JUMPC LAST

LOAD TEMP\_REG**,** ASCII\_CR

STORE TEMP\_REG**,** 007F

LOAD TEMP\_REG**,** ASCII\_LF

STORE TEMP\_REG**,** 0080

;=======================================================

; Prompt User Subroutine

; Allows the user to type

; C:\User>

;=======================================================

PROMPT\_USER

LOAD TEMP\_REG**,** ASCII\_C

STORE TEMP\_REG**,** 0081

LOAD TEMP\_REG**,** ASCII\_COLON

STORE TEMP\_REG**,** 0082

LOAD TEMP\_REG**,** ASCII\_B\_SLASH

STORE TEMP\_REG**,** 0083

LOAD TEMP\_REG**,** ASCII\_U

STORE TEMP\_REG**,** 0084

LOAD TEMP\_REG**,** ASCII\_S

STORE TEMP\_REG**,** 0085

LOAD TEMP\_REG**,** ASCII\_E

STORE TEMP\_REG**,** 0086

LOAD TEMP\_REG**,** ASCII\_R

STORE TEMP\_REG**,** 0087

LOAD TEMP\_REG**,** ASCII\_GREATER

STORE TEMP\_REG**,** 0088

;=======================================================

; TOWN Subroutine

; Prints the designer's hometown

; Hometown: Boyle Heights

;=======================================================

TOWN

LOAD TEMP\_REG**,** ASCII\_H

STORE TEMP\_REG**,** 0089

LOAD TEMP\_REG**,** ASCII\_O

STORE TEMP\_REG**,** 008A

LOAD TEMP\_REG**,** ASCII\_M

STORE TEMP\_REG**,** 008B

LOAD TEMP\_REG**,** ASCII\_E

STORE TEMP\_REG**,** 008C

LOAD TEMP\_REG**,** ASCII\_T

STORE TEMP\_REG**,** 008D

LOAD TEMP\_REG**,** ASCII\_O

STORE TEMP\_REG**,** 008E

LOAD TEMP\_REG**,** ASCII\_W

STORE TEMP\_REG**,** 008F

LOAD TEMP\_REG**,** ASCII\_N

STORE TEMP\_REG**,** 0090

LOAD TEMP\_REG**,** ASCII\_COLON

STORE TEMP\_REG**,** 0091

LOAD TEMP\_REG**,** ASCII\_SPACE

STORE TEMP\_REG**,** 0092

LOAD TEMP\_REG**,** ASCII\_B

STORE TEMP\_REG**,** 0093

LOAD TEMP\_REG**,** ASCII\_O

STORE TEMP\_REG**,** 0094

LOAD TEMP\_REG**,** ASCII\_Y

STORE TEMP\_REG**,** 0095

LOAD TEMP\_REG**,** ASCII\_L

STORE TEMP\_REG**,** 0096

LOAD TEMP\_REG**,** ASCII\_E

STORE TEMP\_REG**,** 0097

LOAD TEMP\_REG**,** ASCII\_SPACE

STORE TEMP\_REG**,** 0098

LOAD TEMP\_REG**,** ASCII\_H

STORE TEMP\_REG**,** 0099

LOAD TEMP\_REG**,** ASCII\_E

STORE TEMP\_REG**,** 009A

LOAD TEMP\_REG**,** ASCII\_I

STORE TEMP\_REG**,** 009B

LOAD TEMP\_REG**,** ASCII\_G

STORE TEMP\_REG**,** 009C

LOAD TEMP\_REG**,** ASCII\_H

STORE TEMP\_REG**,** 009D

LOAD TEMP\_REG**,** ASCII\_T

STORE TEMP\_REG**,** 009E

LOAD TEMP\_REG**,** ASCII\_S

STORE TEMP\_REG**,** 009F

LOAD TEMP\_REG**,** ASCII\_CR

STORE TEMP\_REG**,** 00A0

LOAD TEMP\_REG**,** ASCII\_LF

STORE TEMP\_REG**,** 00A1

;=======================================================

; Backspace

; Allows the user to move the display cursor one

; position backwards and deletes the character at

; that position

;=======================================================

BACKSPACE

LOAD TEMP\_REG**,** ASCII\_BS

STORE TEMP\_REG**,** 00A2

LOAD TEMP\_REG**,** ASCII\_SPACE

STORE TEMP\_REG**,** 00A3

LOAD TEMP\_REG**,** ASCII\_BS

STORE TEMP\_REG**,** 00A4

RETURN

;=======================================================

; Carriage Return and line feed

; They're used to note the termination of a line

;=======================================================

CRLF

LOAD TEMP\_REG**,** ASCII\_CR

STORE TEMP\_REG**,** 00A5

LOAD TEMP\_REG**,** ASCII\_LF

STORE TEMP\_REG**,** 00A6

RETURN

;=======================================================

; TX Subroutine

;=======================================================

TX

COMP CURRENT\_CASE**,** ZERO

RETURNZ

FETCH TEMP\_REG**,** POINTER

OUTPUT TEMP\_REG**,** ZERO

**ADD** POINTER**,** ONE

COMP CURRENT\_CASE**,** 0001

JUMPZ SHOW\_BANNER

COMP CURRENT\_CASE**,** 0002

JUMPZ SHOW\_PROMPT\_USER

COMP CURRENT\_CASE**,** 0003

JUMPZ SHOW\_TOWN

COMP CURRENT\_CASE**,** 0004

JUMPZ SHOW\_BACKSPACE

COMP CURRENT\_CASE**,** 0005

JUMPZ SHOW\_CRLF

COMP CURRENT\_CASE**,** 0006

JUMPZ SHOW\_COUNT

RETURN

SHOW\_BANNER

COMP POINTER**,** 0081

RETURNC

LOAD CURRENT\_CASE**,** 0002

RETURN

SHOW\_PROMPT\_USER

COMP POINTER**,** 0089

RETURNC

LOAD CURRENT\_CASE**,** 0000

RETURN

SHOW\_TOWN

COMP POINTER**,** 00A2

RETURNC

LOAD POINTER**,** 0081

LOAD CURRENT\_CASE**,** 0002

RETURN

SHOW\_BACKSPACE

COMP POINTER**,** 00A5

RETURNC

LOAD CURRENT\_CASE**,** 0000

RETURN

SHOW\_CRLF

COMP POINTER**,** 00A7

RETURNC

LOAD POINTER**,** 0081

LOAD CURRENT\_CASE**,** 0002

RETURN

SHOW\_COUNT

COMP POINTER**,** 00A9

RETURNC

LOAD POINTER**,** 00A5

LOAD CURRENT\_CASE**,** 0005

RETURN

;=======================================================

; RX Subroutine

;=======================================================

RX

COMP CURRENT\_CASE**,** ZERO

RETURNNZ

INPUT STORE\_DATA**,** ZERO

COMP STORE\_DATA**,** ZERO

RETURNZ

COMP STORE\_DATA**,** ASCII\_ASTERISK

JUMPZ RECEIVE\_TOWN

COMP STORE\_DATA**,** ASCII\_BS

COMP STORE\_DATA**,** ASCII\_CR

JUMPZ RECEIVE\_CRLF

COMP STORE\_DATA**,** ASCII\_AT

JUMPZ RECIEVE\_AT

;used to display char count

**ADD** CHAR\_COUNT**,** 0001

OUTPUT STORE\_DATA**,** ZERO

COMP CHAR\_COUNT**,** 0029

JUMPZ RECEIVE\_CRLF

RETURN

RECEIVE\_TOWN

LOAD CURRENT\_CASE**,** 0003

LOAD POINTER**,** 0089

LOAD TEMP\_REG**,** 0000

OUTPUT TEMP\_REG**,** 0000

LOAD CHAR\_COUNT**,** 0000

RETURN

RECEIVE\_BACKSPACE

COMP CHAR\_COUNT**,** 0000

RETURNZ

LOAD CURRENT\_CASE**,** 0004

LOAD POINTER**,** 00A2

LOAD TEMP\_REG**,** 0000

OUTPUT TEMP\_REG**,** 0000

**SUB** CHAR\_COUNT**,** 0001

RETURN

RECEIVE\_CRLF

LOAD CURRENT\_CASE**,** 0005

LOAD POINTER**,** 00A5

LOAD TEMP\_REG**,** 0000

OUTPUT TEMP\_REG**,** 0000

LOAD CHAR\_COUNT**,** 0000

RETURN

RECIEVE\_AT

**CALL** BinToASC

LOAD CURRENT\_CASE**,** 0006

LOAD Pointer**,** 00A7

LOAD TEMP\_REG**,** 0000

OUTPUT TEMP\_REG**,** 0000

LOAD CHAR\_COUNT**,** 0000

RETURN

;=======================================================

; BIN\_TO\_ASCII

;=======================================================

BinToASC

LOAD RE**,** CHAR\_COUNT

LOAD RD**,** 000A

**CALL** FINDIT

**ADD** RB**,** 0030

STORE RB**,** 00A7

**ADD** RE**,** 0030

STORE RE**,** 00A8

RETURN

;=======================================================

; FInd It Routine

;=======================================================

FINDIT

LOAD RB**,** ZERO

REPEAT **SUB** RE**,** RD

JUMPC FOUNDIT

**ADD** RB**,** 0001

JUMP REPEAT

FOUNDIT

**ADD** RE**,** RD

RETURN

;=======================================================

; ISR

;=======================================================

ADDRESS 0300

ISR

INPUT UART\_STATUS**,** PORT

**AND** UART\_STATUS**,** 0003

COMP UART\_STATUS**,** 0003

JUMPZ BOTH\_RX\_TX

COMP UART\_STATUS**,** TX\_RDY

CALLZ TX

COMP UART\_STATUS**,** RX\_RDY

CALLZ RX

RETEN

BOTH\_RX\_TX

**CALL** TX

**CALL** RX

RETEN

;=======================================================

; Main Loop

; main loop is where processor spends most of its time

;=======================================================

MAIN

; update LEDs

JUMP MAIN

;=======================================================

; ISR vectored through 0FFE

;=======================================================

ADDRESS 0FFE

ENDIT

JUMP ISR

END

**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.

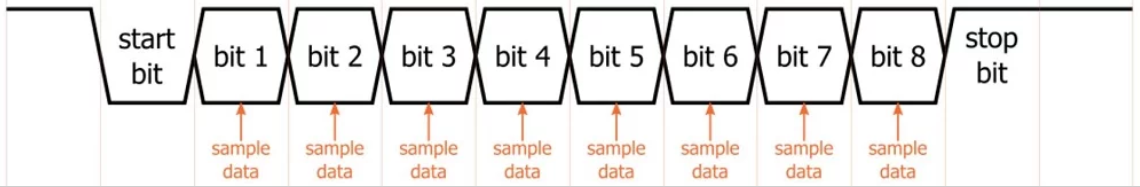


Figure : Illustration of Data Transmission