2019

UART

Chip Specification

Kevin D Lopez

Table of contents

1. Introduction

UART protocol is a form of data transfer, is an old data transfer that is still used till today. Is a simple way for us to transfer data from one device to another with the need of just one cable which transfers data in serial. There can be three different ways of UART protocols, wd. Our UART has programmable speeds that need to be asserted in both ends of the data transfer, the speeds vary from 300 to 19 thousand bits per seconds. There are two main parts of the UART module, there are a TX engine and an RX engine, one for transfer, and one for received perceptibly. The transmit and receive engine of the UART have been created and fully tested and proven that both work under any circumstances.

2. Documents

2.1. Applicable External Documents

2.1.1 PicoBlaze

8-bit Processor that is being emulated. This processor is made to work with spartan 3 but not with the Artix7 and that's why we need to use an emulator. The Pico-blaze manual has the instructions set and proper timing diagrams that our processor works on.

2.1.2 Universal Asynchronous Receiver/Transmitter (UART) Keystone Architecture.

This user guide provides the architecture that the UART works on. It provides the data format and format if we want to change the word length being transmitted.

2.1.3 Nexys 4™ FPGA Board Reference Manual.

FPGA that we program our chip to work on. The IO and Electrical characteristics are documented in this manual.

2.1.4 Xilinx 7 Series FPGA Libraries Guide.

Library for the electrical characteristics of the Nexys4 FPGA. This library includes the input and output buffers, we can also modify for performance or duration and change the slew rate.

2.2. Applicable Internal Documents

2.2.1 Transmit Engine

The first part of the UART where we create the transmit engine of the UART. We can only output to the terminal using the internal processor.

2.2.2 Full UART

Full UART Document that has all the functionality and test performed. This SOC design has the ability to receive data check what it is and send corresponding functionality.

2.2.3 UART and SRAM

Full UART with the added ability to store data being received and echoed once user requests. This document has the tests performed for the memory and the specifications of such SOC.

2.2.4 UART and TSI

Full UART SOC design with input and output buffers for the Nexys4 FPGA board. The TSI adds the functionality to change the electric characteristics of the inputs and outputs which affects the performance of the design.

3. Requirements

Generate a Universal form of data communication. This data communication is on serial and its extremally configurable. Send data at different speeds, size and a way of error checking(parity). Use this same universal form of communication to communicate between an embedded MCU, terminal and keyboard of a computer. Have terminal in the computer output all the characters entered by the keyboard and create an algorithm to display backspace, returns and a count for keypresses.

3.1 Transmit engine Requirements

For this part of the SOC design, we're creating the Transmit side of the UART and there is an Embedded processor to communicate in one way to our serial terminal on our computers. For this part of requirements, we are transmitting "CSULB CECS460 [" Followed up by a line counter and a new line.

3.2 Full UART

The full UART is the main core of this Project and it's the only way of communication between the Terminal and our processor. For this project, we created the receiving engine and added extra functionality to our keyboard. When a character is pressed on the keyboard is echoed back into the Terminal and keys @*Backspace and enter have extra functionality.

KEY PRESS	ACTION
@	Character Count
*	Displays user Hometown
Backspace	Space, Back, Space
Enter	Character Return, Line Feed

3.3 SRAM Integration

Integrating a Memory into the Project adds a lot of extra functionality and it adds the topic of memory testing on our project. For this section of the project, we add and do a Memory test on a 32k x 16 bits SRAM memory. Every time a key is pressed, we need to echo such key and store it into the SRAM and when the Key % is pressed we can see every key that has been pressed by outputting all the values inside of it.

3.4 TSI

Creating an application specific module is necessary to specify what is the targets outputs and inputs for a Chip design. In our case, we're working on a Nexys4 FPGA Board. For this section of the project, there would be BUFG buffers for the clock, Single-ended input buffers for the inputs and Single-ended Output buffers for the output.

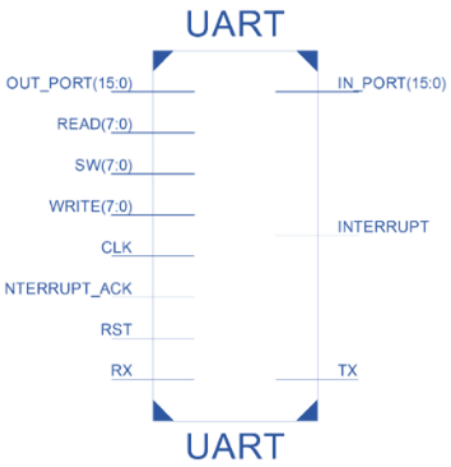
4. Top Level Design

4.1 Description

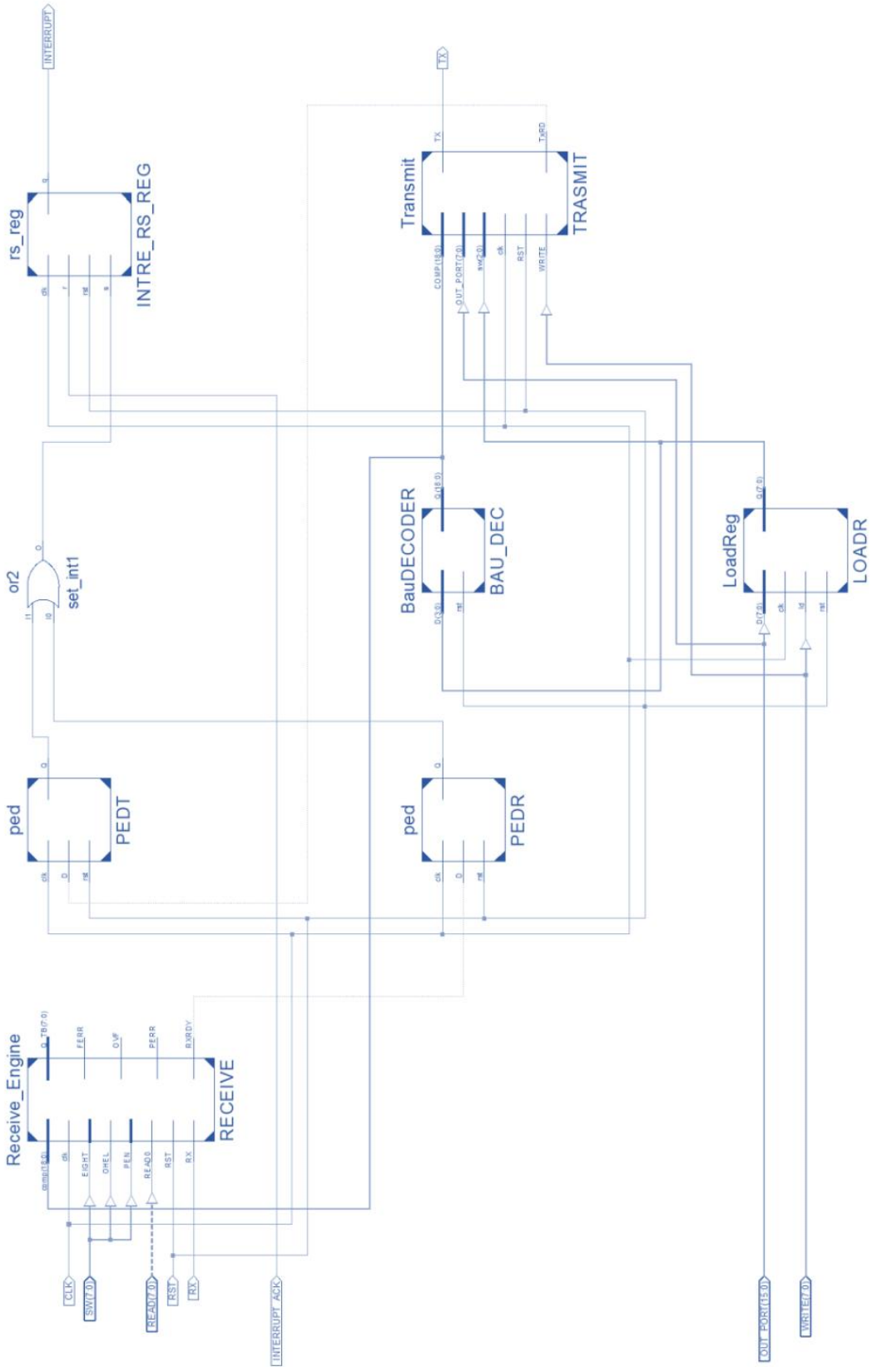
The UART consist of two main blocks a receive engine and a transmit engine. Both get data and transmit data in a way that would be expandable between other UART modules. Our receive engine receives data in serial and transmits it to the processor in parallel. The transmit engine receives data in parallel and transfers it in series, we can see how UARTS are connected RX to the TX and TX to the RX to other UARTs respectably. The UART we have implemented is connected between a processor, the terminal and keyboard in our computer. We send data from our keyboard and we can view it on the terminal, once it has been received and registered by the processor.

4.2 Block Diagram

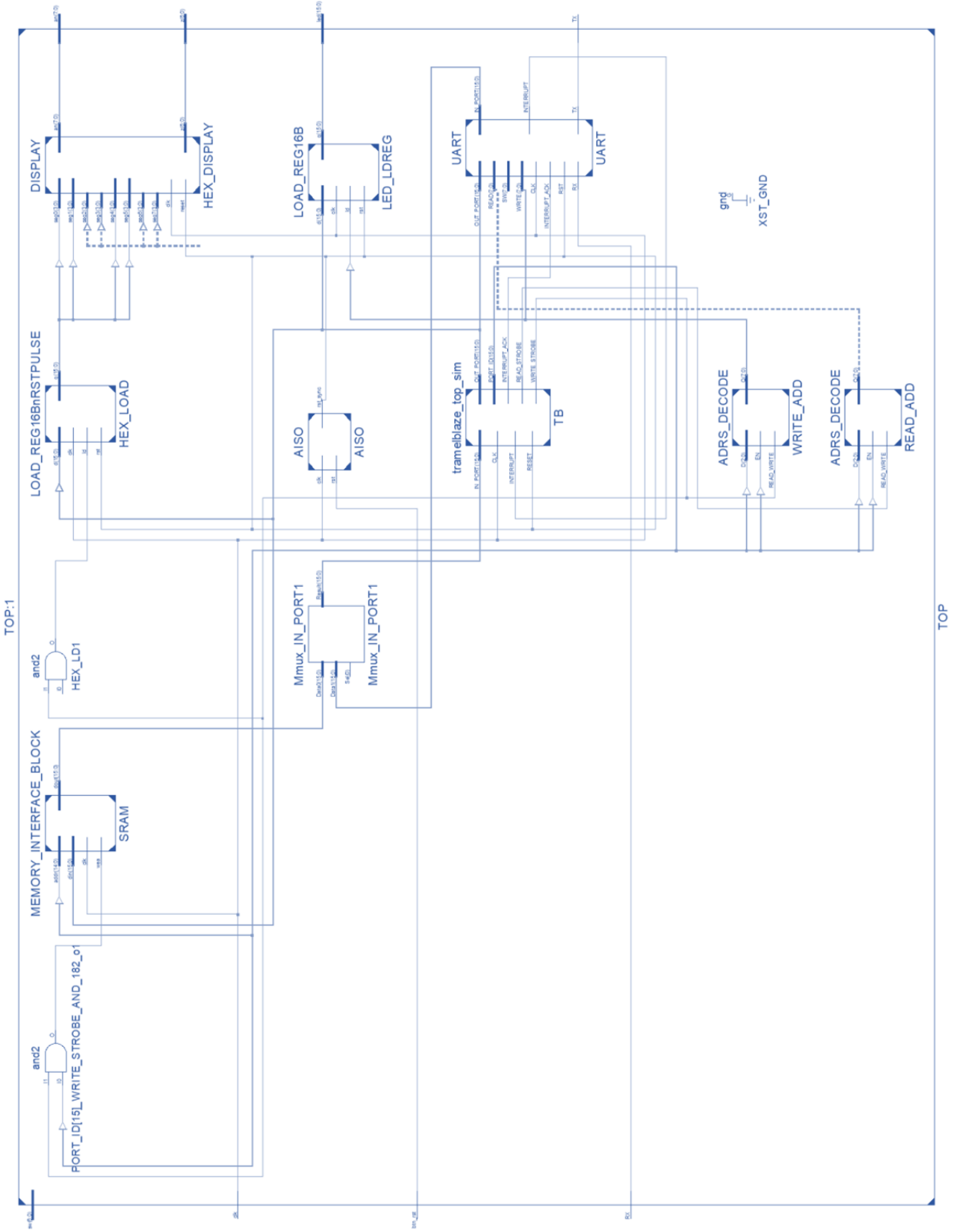
4.2.1 UART Top Level Diagram



4.2.2 UART Detail Diagram



4.2.3 System Core Detail Block Diagram



4.3 Data flow description

The data in the transmit engine is received in parallel and its transmitted in parallel. It gets data from the Trailblaze processor and loaded in a load register which we decode if we have 8 bits or parity enable and we send it to a ten-bit shift register. The shift registers firsts transmit the start bit then followed by the data and parity if it was enabled.

The data in the received engine is received in series and it's transformed in parallel data. Once there is a start bin received, we start to shift data into a 10-bit shift register. Once all the data has been received, we shift the data with a remap register. This remap register shifts the data depending on the configuration we have, for example, 8-bits or parity enable. There is extra hardware to check if the data that has been received has no errors, there are three flags to do so. There is a parity error flag that checks for the parity correctness of the parity bit. There is also a Framing error flag that checks if our baud rate has been shifted causing an error with the stop bit. For the last flag, we have overflow error that checks for data lost because data continues to being received without being done transmitting it to our processor.

4.4 I/O

4.4.1 Signal Names

Inputs:

- **RX**
- **SW[B.R, EIGHT, PEN, OHEL]**
- **CLK**
- **RST**

Outputs:

- **Leds:** 15 to 0
- **TX:**

4.4.2 Pin Assignments

Inputs:

- **RX**: Receiving serial data
- **SW** 6 to 1: 6 to 4 control baud rates (Baud rate values in the table below), 3 controls 8-bit mode, 2 and 1 controls parity enable and odd or even parity, respectably.
- **RST:** Reset four our design
- **Clk:** clock source for out design

Outputs:

- **LEDs:** 15 to 0
- **TX:** Sending serial data out
- **Hex to Seven:** For outputting UART Flags

SW[6:4]	Baud Rate
0000	300
0001	1,200
0010	2,400
0011	4,800
0100	9,600
0101	19,200
0110	38,400
0111	57,600
1000	115,200
1001	230,400
1010	460,800
1011	921,600

4.4.3 Electricals Characteristics

Inputs:

- **SW**: connected to 3.3 volts for a logic high and ground for zero
- **RST:** Push button acting as a momentary switch

Outputs:

- **LEDs:** 3.3v and connected to ground. Only on when being sourced with 3.3 volts
- **Hex to Seven:** Connected to 3.3v and transistor to ground to indicated what anode is on.

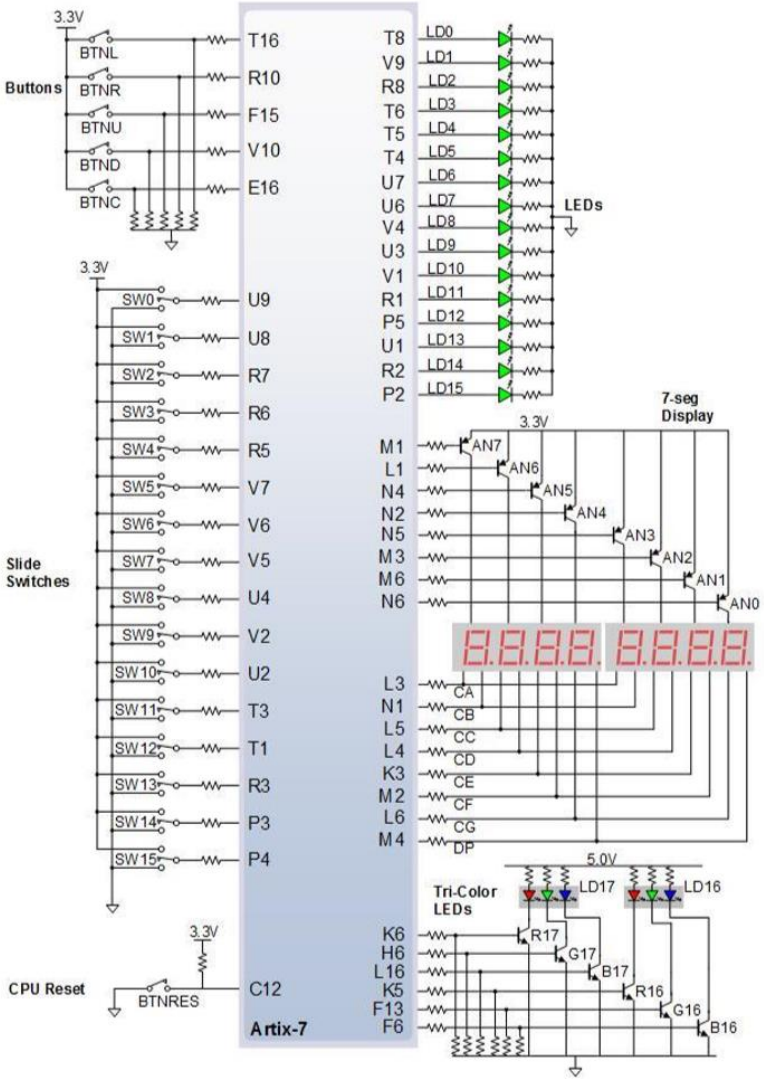


Figure 16. General Purpose I/O devices on the Nexys 4

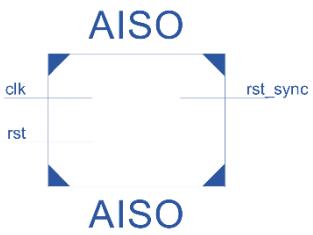
4.5 Clocks

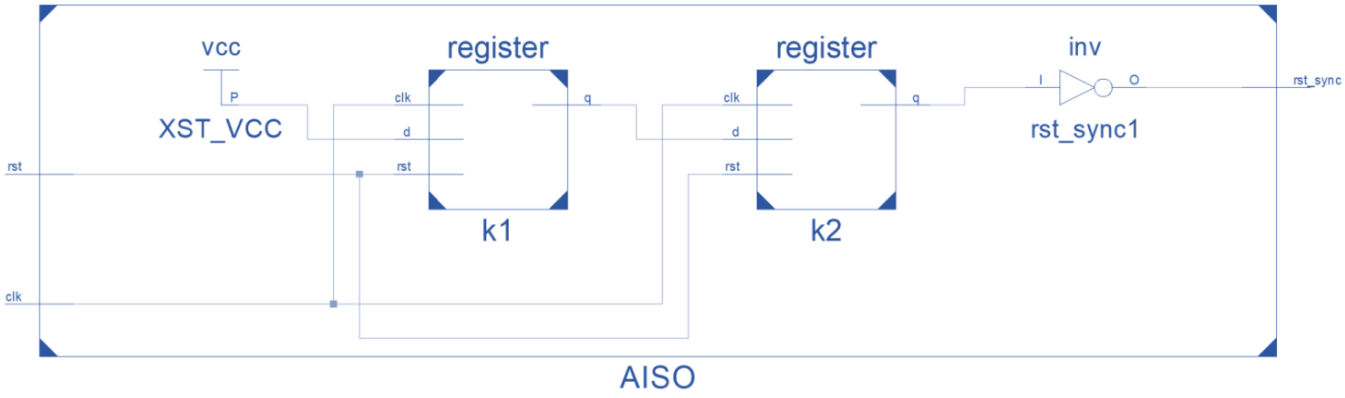
The Nexys 4 board includes a single 100MHz crystal oscillator connected to pin E3 (E3 is an MRCC input on bank 35). The input clock can drive MMCMs or PLLs to generate clocks of various frequencies and with known phase relationships that may be needed throughout a design. Some rules restrict which MMCMs and PLLs may be driven by the 100MHz input clock.

4.6 Resets

4.6.1 AISO

Asynchronous in Synchronous out, the use of this reset is so that we could synchronize the release of reset. By Synchronizing the release of reset we get rid of the change that our registers would be in a metastable state for a period.





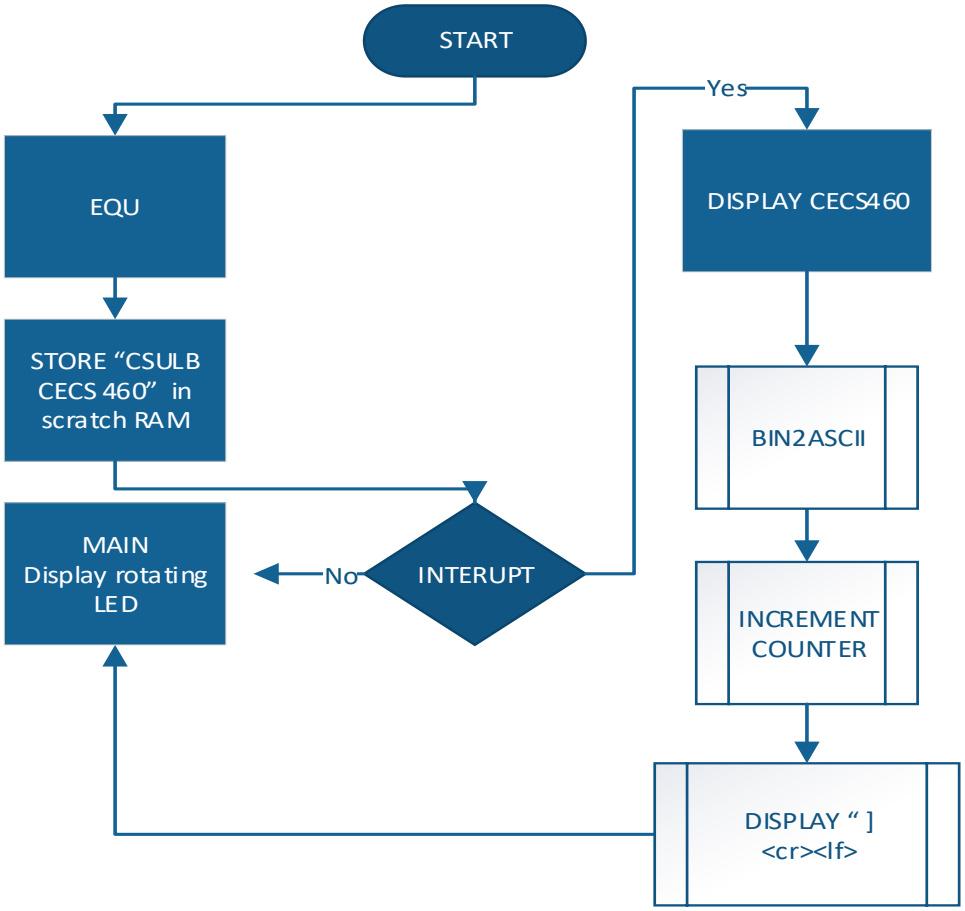
4.7 Software

4.7.2 Transmit engine

4.7.2.1 Description

The software to test the transmit engine is loaded into the Processor. Such software outputs CSULB CECS 460 [, followed up by a line counter, and "] <cr><lf>. We created an algorithm so that our counter would output the line counter in ASCII characters. There is a software flowchart to display the way the software is formatted.

4.7.2.2 Detailed Software Planning Document

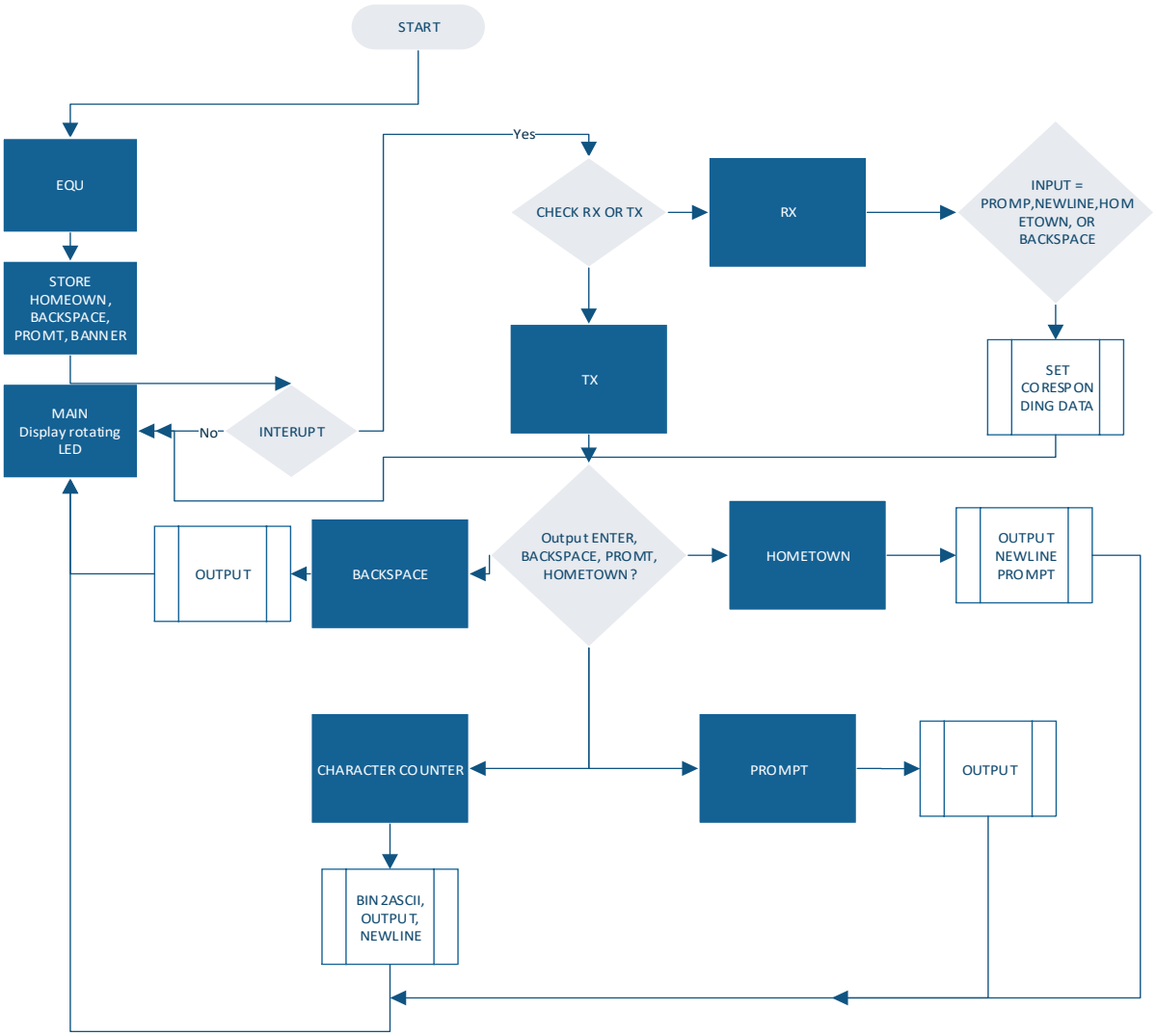


4.7.2 UART Engine Test

4.7.2.1 Description

The software for the full UART receives and sends data between the keyboard presses and the terminal. We can click any character on our keyboard, and we should see it appear in the terminal line. We have special characters that do another type of functions like display our hometown how many characters have been received and backspace and character return. We check for these characters in our receive engine and we output the corresponding command depending on the ASCII value.

4.7.2.2 Detail Software planning document



4.7.2.3 Source Code on appendix

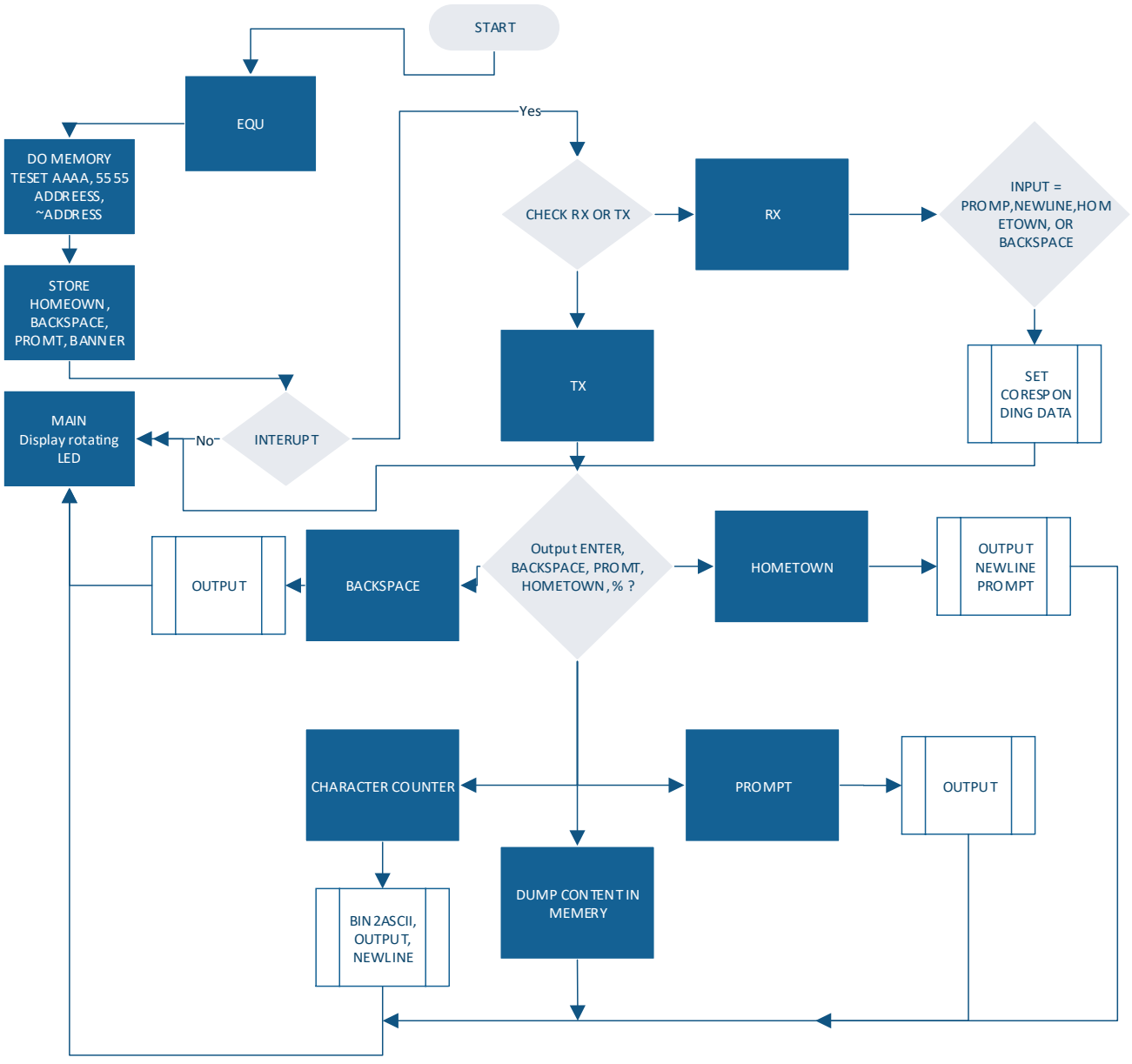
4.7.3 Memory Software

4.7.3.1 Description

The addition of the memory added the function to store everything that was stored, and such functionality needed to be programed in the software of our Processor. At the moment that the chip is startup we need to perform a memory test so we can assure that there are no fault registers on our memory, so we write and output 5555 then we write AAAA. To add the functionality of dumping all characters that the user had entered we store every character in the RAM as the same time that we output it to the terminal. To store information onto our memory and not store when were outputting for

other addresses I used PORT_ID 15. Our address decoder would only work when PORT_ID 15 is not enabled and the its used, so we don't write to memory and to the LEDS at the same time.

4.7.3.2 Detail Software Planning document



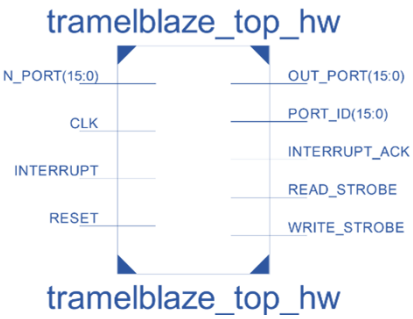
5. External Acquired Blocks

5.1 TramelBlaze MCU

5.1.1 Description

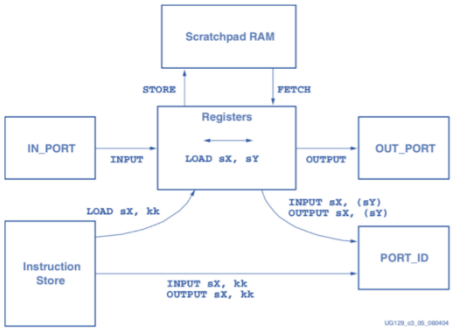
The tramelblaze is a 16-bit emulator microcontroller. This MCU emulates the 8-bit Pico-blaze that were able to instantiate in Xilinx ISE. Because of its simplicity, the tramelblaze doesn't support high-level programming languages, So the code needs to be generated in assembly language. This MCU is synthesized along other digital logic to make SOC designs. Since is a one to one assembly instruction to machine instruction.

5.1.2 Block Diagram



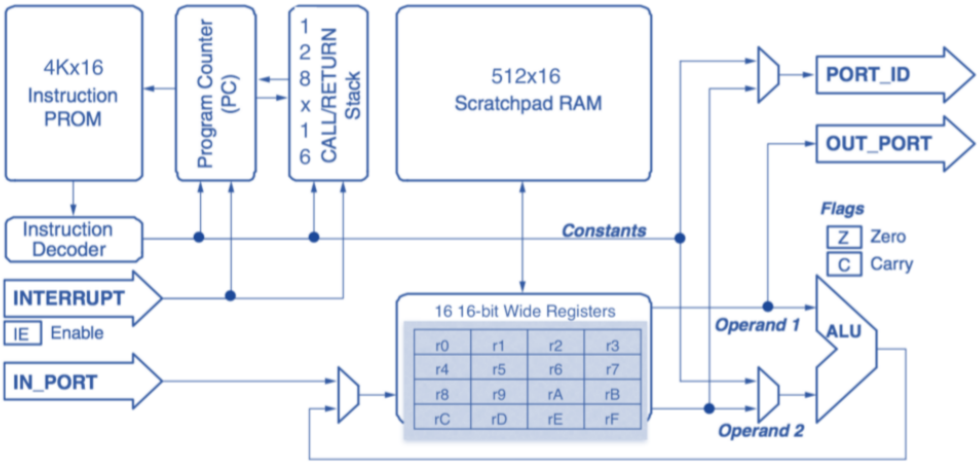
This embedded MCU consists of 3 memories ROM which is 4096 x 16, a scratch RAM which is 512 x 16 bits, a stack RAM which is 128 x 16 bits.

The ROM is where we store our instructions and its read address by address suing the addr and instruction ports. We store information from our ROM using the INPUT instruction and we specify what port data will be coming from. Outputting uses the OUT_PORT to specify what PORT_ID were outputting form. As we can see store writes into the scratchpad and fetch gets data from the scratchpad into the registers.



5.1.3 Register Map

TramelBlaze is a 16-bit microcontroller with the following characteristics



- 16-bit data width
- 16-bit ALU with carry and zero flags

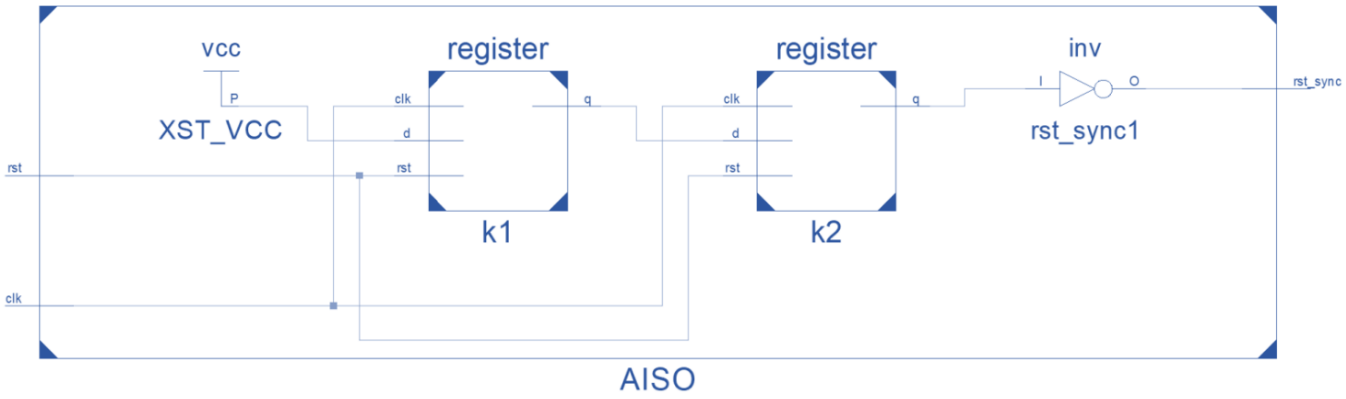
- 16 16-bit general purpose registers
- 2 clock cycles per instruction
- 5 clock cycles for interrupt handling instructions.

5.2 AISO

5.2.1 Description

When the release of reset is meet right before a Posedge side of our clock an Asynchronous release of reset can create metastability in the design. The purpose of this block is to synchronize the release of reset as the name says Asynchronous In synchronous Out. This block contains the input reset button and it provides a reset synchronous Reset for our whole design.

5.2.2 Block Diagram



5.2.3 I/0

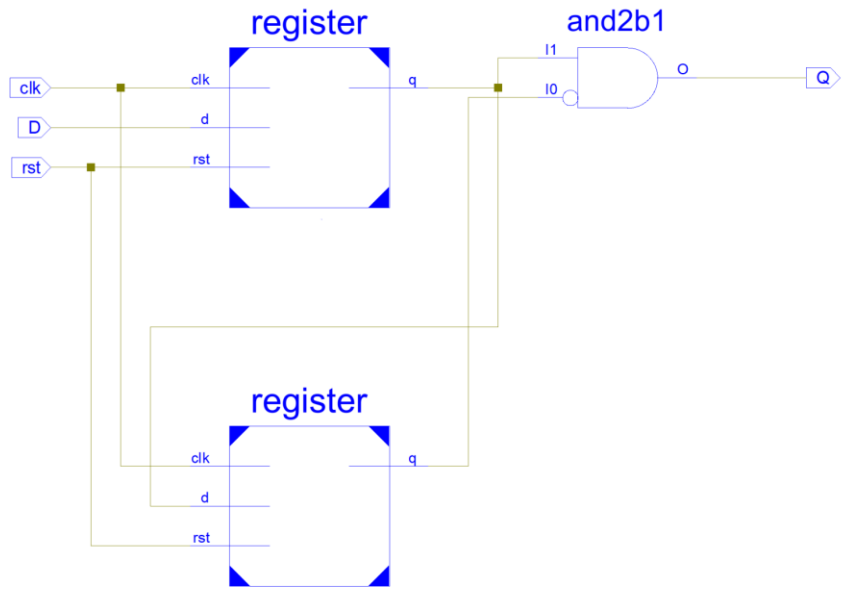
The input for this block is a button reset input for our design and a clock signal. The Output is Synchronized reset pin.

5.3 Posedge Edge Detect

5.3.1 Description

When a signal goes to high state there is a signal that goes up for an unpredictable time. The purpose of this block is to generate a small pulse that would only trigger an interrupt for only one time and one time only.

5.3.2 Block Diagram



5.3.2 I/0

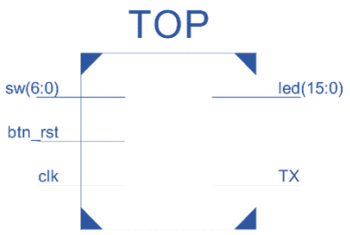
Inputs: clock signal, Input signal to shorten, asynchronous Reset.

Output: Signal pulse

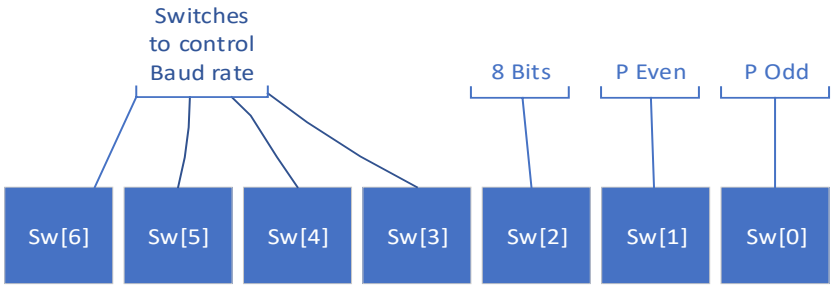
6. Internal block Design

6.1 Transmit Engine

6.1.1 Description

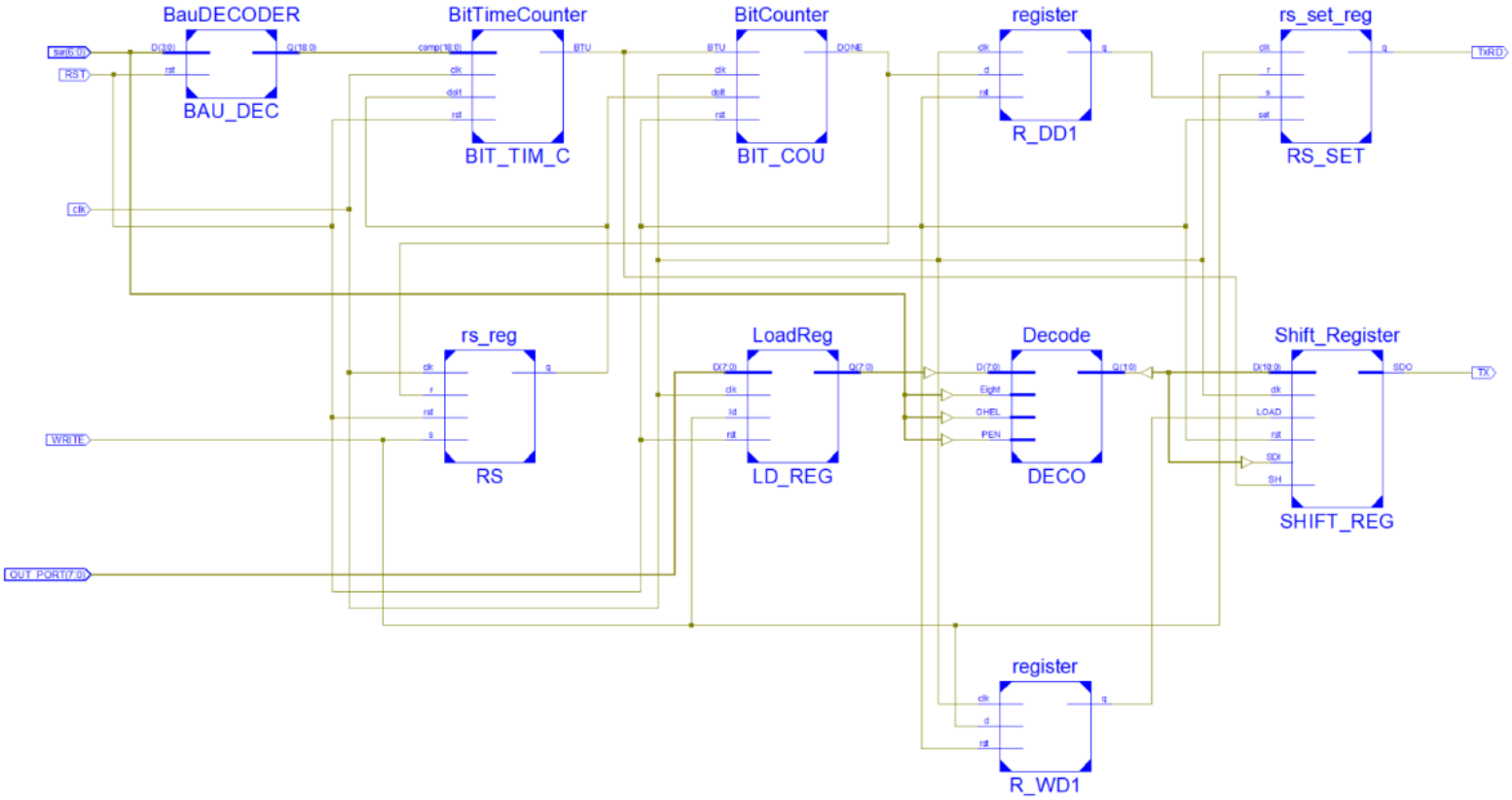


The Transmit engine is what sends the bits at a certain and specified speed. The TX engine has the control of sending seven to eight bit and sending parity bits for examination the accuracy of data. When our TX engine is not sending any data is constantly outputting a one to signify that there is no new signal being sent. When we start the process of sending a bit first, we output a start bit (zero) and we send our data. At the end of our data, we have the option of sending a parity bit, and then a stop bit to signal that the byte has been sent. We have six switches in this design that we use to program how our data is going to be sent, we have a way to send seven bits with even or odd party, and same for eight bits.



Switches three to six are used to control baud rate and switches zero to two are used for parity.

6.1.2 Block diagram.



We can see we have eight different purpose modules. Two RS registers to set a command, two decoders to select baud rate or parity, and shift register for transferring the data serially out of our TX engine. We used six switches to control such operations form the decoders and the input (out_port) is being written on the negedge side of the write signal.

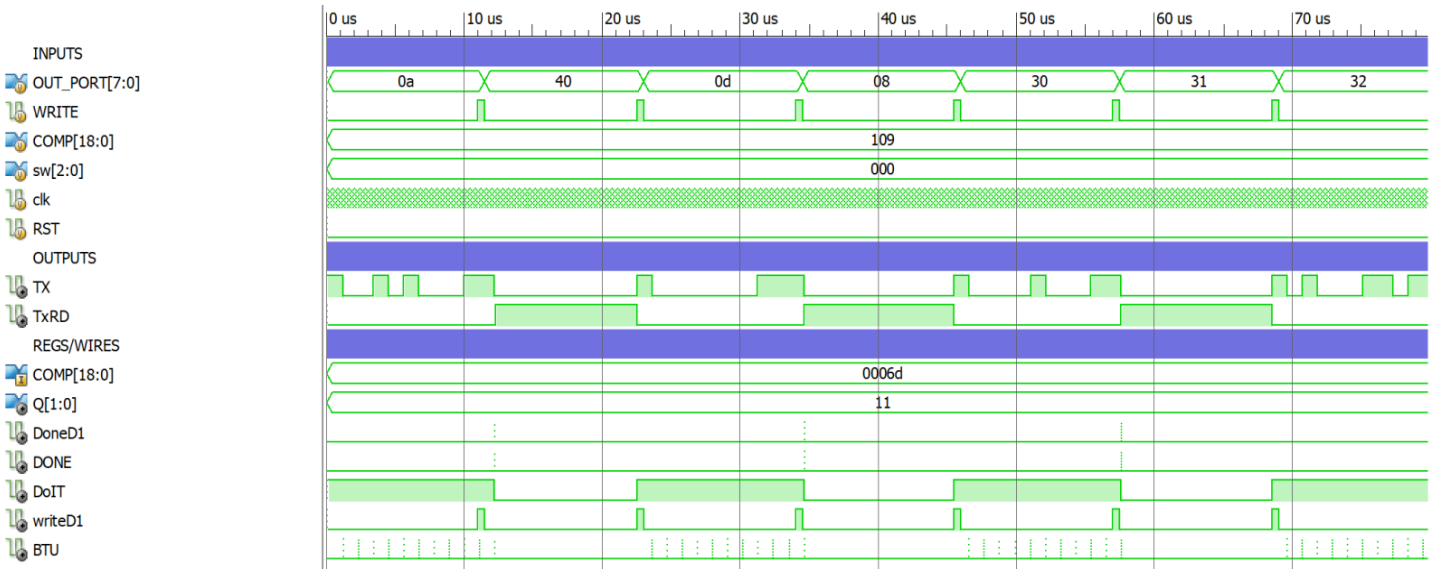
6.1.3 I/0

Inputs	Purpose
7 x Switches	Generate Desired Baud Rate
Reset	Reset for design
Clock	Clock for design
WRITE	Load Signal for data
OUTPORT	Parallel Data coming in

Outputs	Purpose
TxRD	Signals when a data is ready to send out
TX	Serial Data

6.3.4 Verification

In this verification were sending. At the same time that the input arrives the Write signal needs to be set high and the Data start to go through the transmit engine and starts be transmitted serial out of the TX output signal. First it sends the start bit followed by MSB to LSB then stop bit.

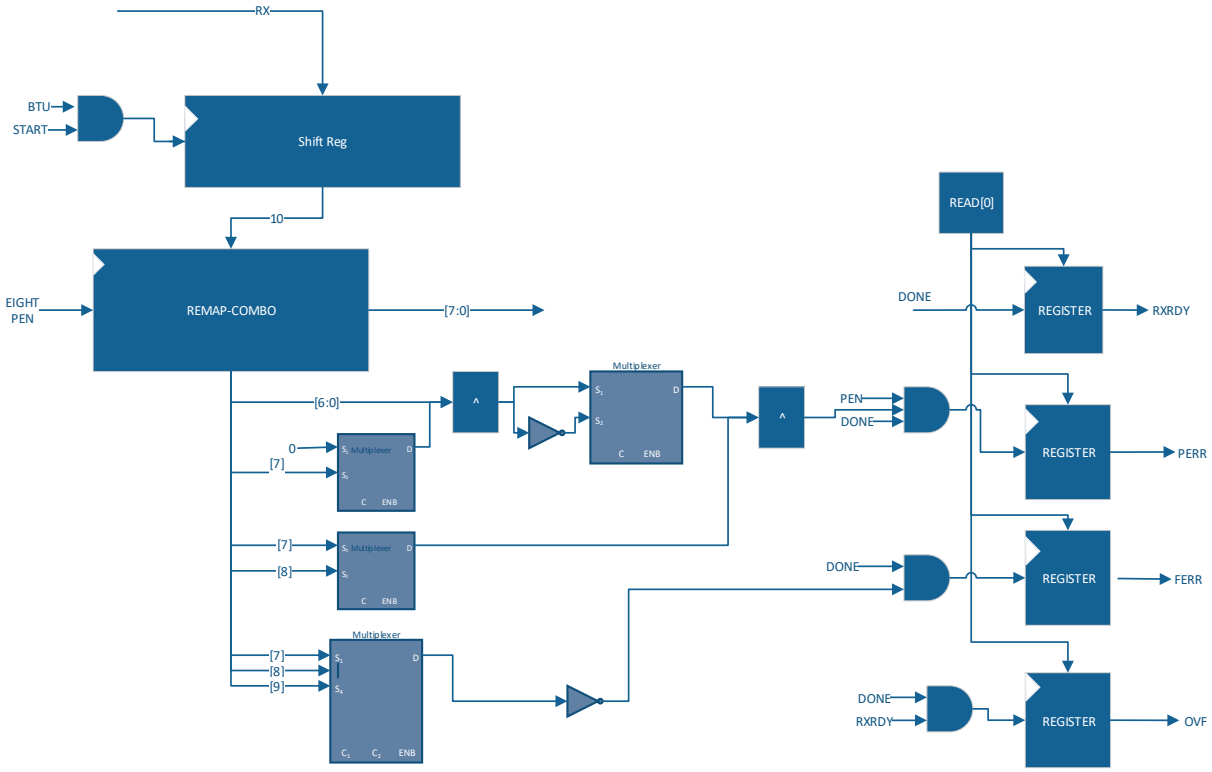


6.2 Receive Engine

6.2.1 Description

To make a full UART work there needs to be a sending and receiving of data. This receive engine give the UART such functionality. In this module Data is been received serial and its transformed into a parallel data that can be represented in a hex number.

6.2.2 Receive Engine Block Diagram

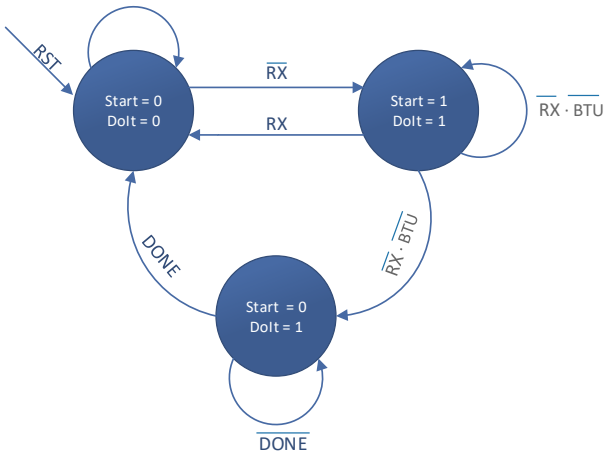


6.2.3 I/O

Inputs	Purpose
BTU	Bit time up
Reset	Reset for design
Clock	Clock for design
EIGHT	Eight Data bit mode
PEN	Parity enable
START	Start signal

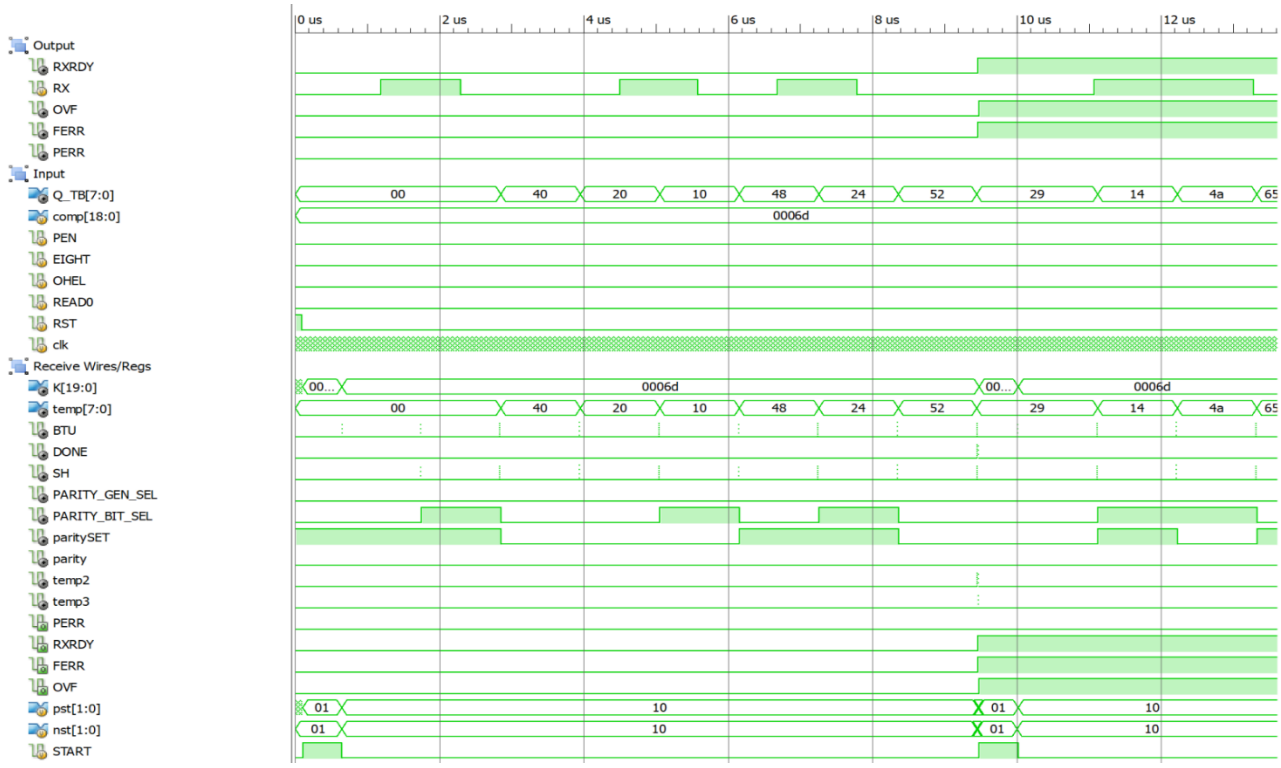
Outputs	Purpose
RXRDY	RX interrupt signal
RX	Serial Data In

6.2.4 State Machine



6.2.5 Verification

To verify this engine, I used the Transmit engine to write a file of ones and zeros of its own test. The receive engine was testing the sending of 0x0a, 0x4, 0x08 and other hex numbers. Using such output file of ones and zero was the source of the input for the receive engine since it can only receive data serial.

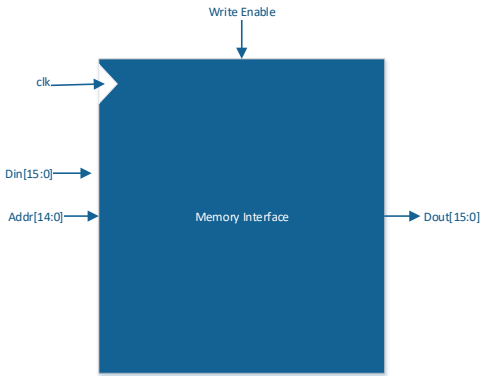


6.3 SRAM

6.3.1 Description

The integration of having a memory added extra functionality to our project, for this project we added the functionality of echoing all the character the user has pressed. To add this functionality we had to wire the MSB of our address into the write enable of our memory so that we only write to our memory and no to any of the other functions of the UART since we specify on our address decoder not to use address the MSB of the address.

6.3.2 Block Diagram



6.3.3 I/0

Inputs	Purpose
Reset	Reset for design
Clock	Clock for design
ADDR	where to store data
Write Enable	Write enable for memory
Din	Data Coming in

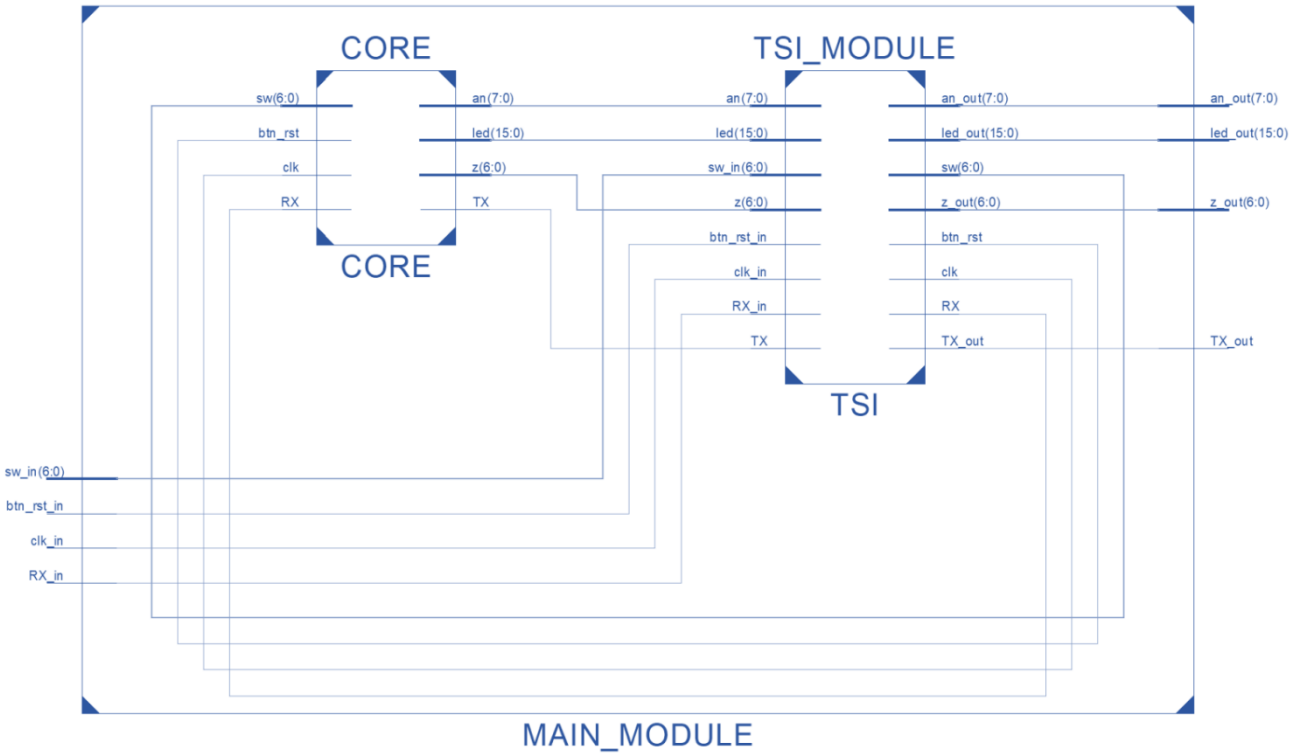
Outputs	Purpose
Dout	Output for reading data

6.4 Technology Specific instantiation

6.4.1 Description

The TSI adds the functionality to change the electric characteristics of the inputs and outputs which affects the performance of the design. This module exclusively is to make the core design work with the Nexys4 FPGA board. Such module is composed of buffers that are found in the Xilinx 7 series FPGA. For this design there was used IBUF (Input Buffer), BUFG (Global clock Buffer), and OBUF(Output clock buffer).

6.4.2 Block Diagram



6.4.2 I/O

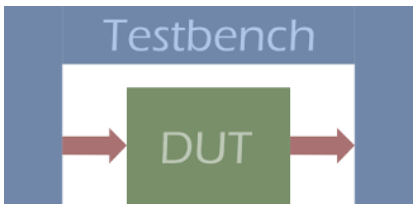
Inputs	Purpose
sw_in	Switch input buffer
btn_rst_in	reset input buffer
clk_in	clock input buffer
RX_in	RX input buffer
an	anode input buffer
led	LED input buffer
z	Segment input buffer
TX	TX input buffer

Outputs	Porpuse
an_out	Anothe output
led_out	Led ouput
sw	Sw output buffer
z_out	Segment outptu buffer
bnt_rst	reset outptu buffer
clk	clk output buffer
RX	RX output buffer
TX_out	TX outptu buffer

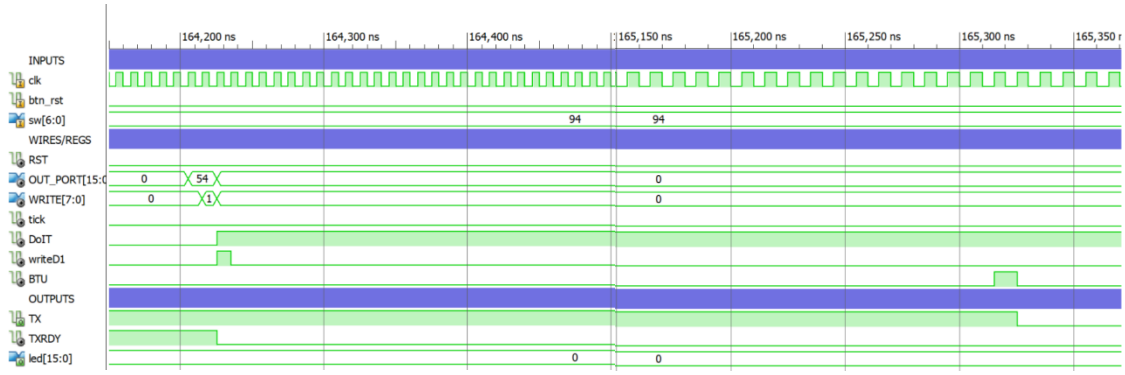
7. Chip Level Verification

7.1 Transmit Engine

We test our transmit engine with our ISE tools before the synthesis process. A TX engine test bench provided the stimulus so it would pretend that I was being simulated in hardware. The stimulus was provided to the switches, writes signals, and data input. We



observed our data being shift out by the shift register at every Bit Time up which was generated by our Bit timer and baud rate decoder.

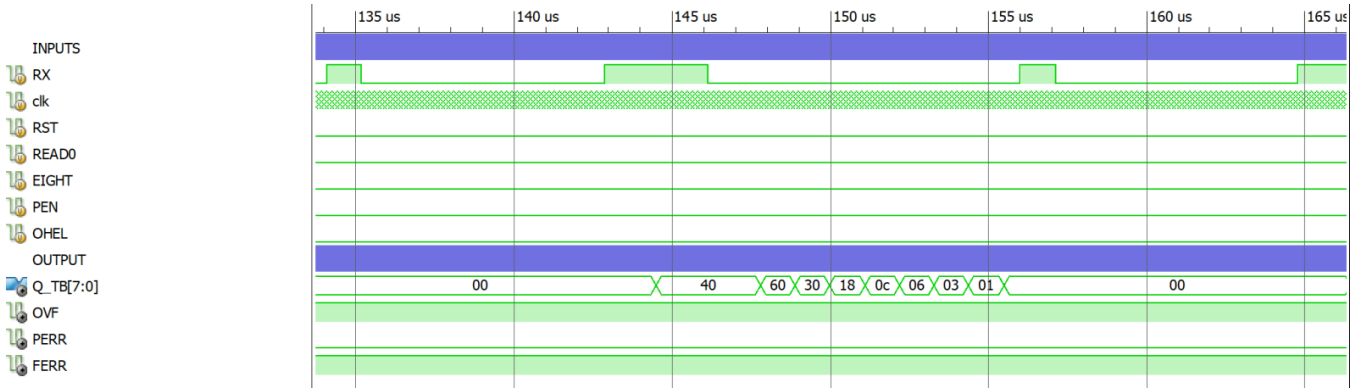


We can see in our waveform that once doIT goes high the counter for the BTU signal starts counting and it stays counting up until our count for the Bit Time is complemented and then we see our TX signal go low for the Start bit. Such action is repeated until all bits to write 54 is completed or 110110 in binary.

7.2 Receive Engine

Testing the receive engine was more complicated since the data that we were receiving needed to formatted in serial form, it needed to start with the start bit then data, parity bit followed by stop bit and every single data bit needed to meet the proper timing depending on the baud rate.

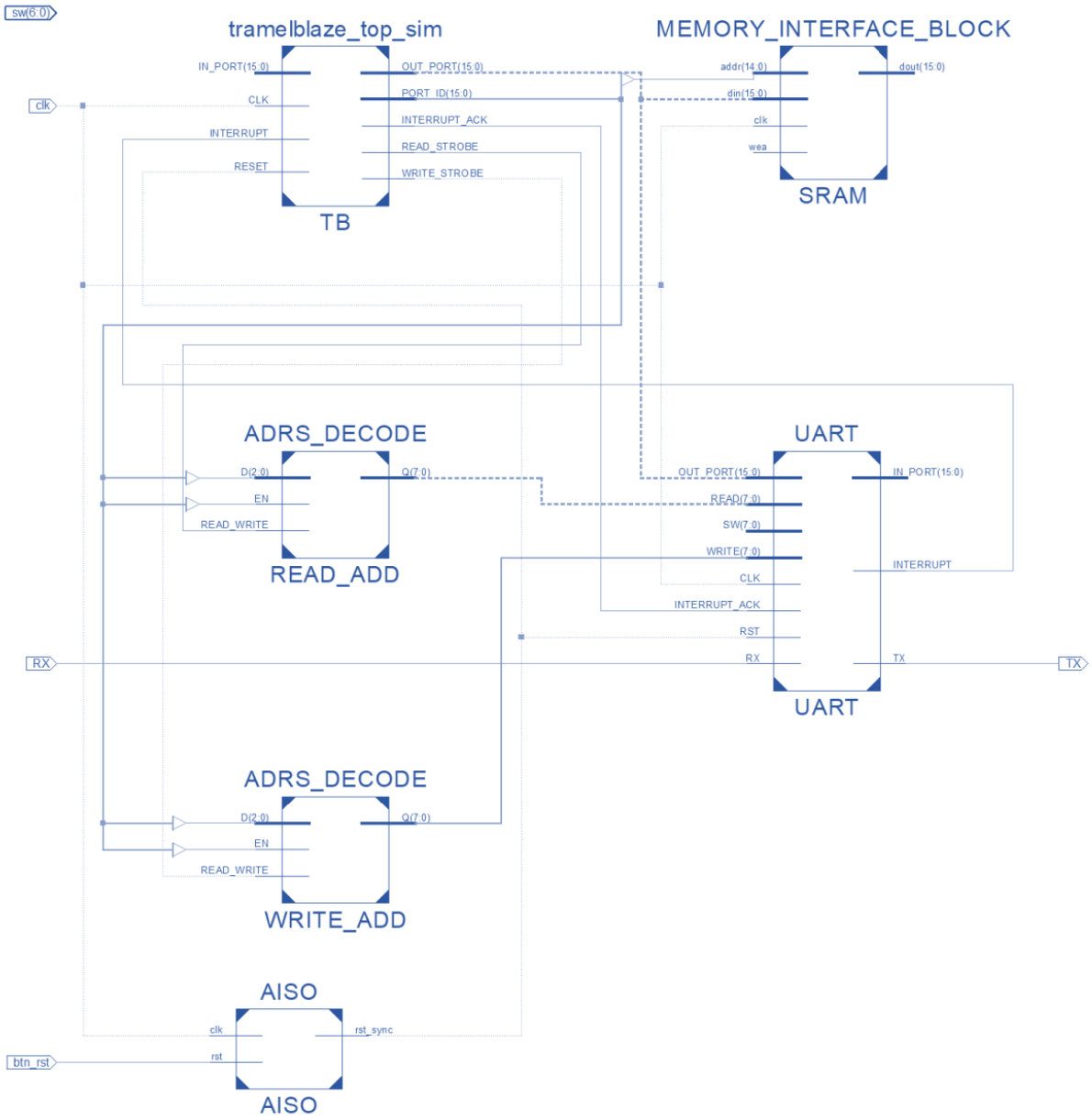
For this testing, I used the old and tested transmit engine to write an output file with ASCII to see the data that would come out of the receive engine. Looking at the data that would be inputted into t tramelblaze would determinate if our design was correct or not.



7.3 Memory Test

In this RAM integration, there are two memory test algorithms to verify that is fully working. For the first algorithm, there is a check for address decoding fault, in where I write to every address in the memory, the second test I check at stuck at fault in where I write 5555 in every memory address and then write AAAA and check that everything changed correctly. We can see such tests in the pictures below.

7.3.1 SRAM on chip Level test Schematic



7.3.1 Address test

Address: | Columns: 16 | Address Radix: Hexadecimal | Value Radix: Hexadecimal

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x0	8000	8001	8002	8003	8004	8005	8006	8007	8008	8009	800A	800B	800C	800D	800E	800F
0x10	8010	8011	8012	8013	8014	8015	8016	8017	8018	8019	801A	801B	801C	801D	801E	801F
0x20	8020	8021	8022	8023	8024	8025	8026	8027	8028	8029	802A	802B	802C	802D	802E	802F
0x30	8030	8031	8032	8033	8034	8035	8036	8037	8038	8039	803A	803B	803C	803D	803E	803F
0x40	8040	8041	8042	8043	8044	8045	8046	8047	8048	8049	804A	804B	804C	804D	804E	804F
0x50	8050	8051	8052	8053	8054	8055	8056	8057	8058	8059	805A	805B	805C	805D	805E	805F
0x60	8060	8061	8062	8063	8064	8065	8066	8067	8068	8069	806A	806B	806C	806D	806E	806F
0x70	8070	8071	8072	8073	8074	8075	8076	8077	8078	8079	807A	807B	807C	807D	807E	807F
0x80	8080	8081	8082	8083	8084	8085	8086	8087	8088	8089	808A	808B	808C	808D	808E	808F
0x90	8090	8091	8092	8093	8094	8095	8096	8097	8098	8099	809A	809B	809C	809D	809E	809F
0xA0	80A0	80A1	80A2	80A3	80A4	80A5	80A6	80A7	80A8	80A9	80AA	80AB	80AC	80AD	80AE	80AF
0xB0	80B0	80B1	80B2	80B3	80B4	80B5	80B6	80B7	80B8	80B9	80BA	80BB	80BC	80BD	80BE	80BF
0xC0	80C0	80C1	80C2	80C3	80C4	80C5	80C6	80C7	80C8	80C9	80CA	80CB	80CC	80CD	80CE	80CF
0xD0	80D0	80D1	80D2	80D3	80D4	80D5	80D6	80D7	80D8	80D9	80DA	80DB	80DC	80DD	80DE	80DF
0xE0	80E0	80E1	80E2	80E3	80E4	80E5	80E6	80E7	80E8	80E9	80EA	80EB	80EC	80ED	80EE	80EF
0xF0	80F0	80F1	80F2	80F3	80F4	80F5	80F6	80F7	80F8	80F9	80FA	80FB	80FC	80FD	80FE	80FF
0x100	8100	8101	8102	8103	8104	8105	8106	8107	8108	8109	810A	810B	810C	810D	810E	810F
0x110	8110	8111	8112	8113	8114	8115	8116	8117	8118	8119	811A	811B	811C	811D	811E	811F
0x120	8120	8121	8122	8123	8124	8125	8126	8127	8128	8129	812A	812B	812C	812D	812E	812F
0x130	8130	8131	8132	8133	8134	8135	8136	8137	8138	8139	813A	813B	813C	813D	813E	813F
0x140	8140	8141	8142	8143	8144	8145	8146	8147	8148	8149	814A	814B	814C	814D	814E	814F
0x150	8150	8151	8152	8153	8154	8155	8156	8157	8158	8159	815A	815B	815C	815D	815E	815F
0x160	8160	8161	8162	8163	8164	8165	8166	8167	8168	8169	816A	816B	816C	816D	816E	816F
0x170	8170	8171	8172	8173	8174	8175	8176	8177	8178	8179	817A	817B	817C	817D	817E	817F
0x180	8180	8181	8182	8183	8184	8185	8186	8187	8188	8189	818A	818B	818C	818D	818E	818F
0x190	8190	8191	8192	8193	8194	8195	8196	8197	8198	8199	819A	819B	819C	819D	819E	819F
0x1A0	81A0	81A1	81A2	81A3	81A4	81A5	81A6	81A7	81A8	81A9	81AA	81AB	81AC	81AD	81AE	81AF
0x1B0	81B0	81B1	81B2	81B3	81B4	81B5	81B6	81B7	81B8	81B9	81BA	81BB	81BC	81BD	81BE	81BF
0x1C0	81C0	81C1	81C2	81C3	81C4	81C5	81C6	81C7	81C8	81C9	81CA	81CB	81CC	81CD	81CE	81CF
0x1D0	81D0	81D1	81D2	81D3	81D4	81D5	81D6	81D7	81D8	81D9	81DA	81DB	81DC	81DD	81DE	81DF
0x1E0	81E0	81E1	81E2	81E3	81E4	81E5	81E6	81E7	81E8	81E9	81EA	81EB	81EC	81ED	81EE	81EF
0x1F0	81F0	81F1	81F2	81F3	81F4	81F5	81F6	81F7	81F8	81F9	81FA	81FB	81FC	81FD	81FE	81FF
0x200	8200	8201	8202	8203	8204	8205	8206	8207	8208	8209	820A	820B	820C	820D	820E	820F
0x210	8210	8211	8212	8213	8214	8215	8216	8217	8218	8219	821A	821B	821C	821D	821E	821F
0x220	8220	8221	8222	8223	8224	8225	8226	8227	8228	8229	822A	822B	822C	822D	822E	822F
0x230	8230	8231	8232	8233	8234	8235	8236	8237	8238	8239	823A	823B	823C	823D	823E	823F
0x240	8240	8241	8242	8243	8244	8245	8246	8247	8248	8249	824A	824B	824C	824D	824E	824F
0x250	8250	8251	8252	8253	8254	8255	8256	8257	8258	8259	825A	825B	825C	825D	825E	825F
0x260	8260	8261	8262	8263	8264	8265	8266	8267	8268	8269	826A	826B	826C	826D	826E	826F
0x270	8270	8271	8272	8273	8274	8275	8276	8277	8278	8279	827A	827B	827C	827D	827E	827F
0x280	8280	8281	8282	8283	8284	8285	8286	8287	8288	8289	828A	828B	828C	828D	828E	828F
0x290	8290	8291	8292	8293	8294	8295	8296	8297	8298	8299	829A	829B	829C	829D	829E	829F
0x2A0	82A0	82A1	82A2	82A3	82A4	82A5	82A6	82A7	82A8	82A9	82AA	82AB	82AC	82AD	82AE	82AF
0x2B0	82B0	82B1	82B2	82B3	82B4	82B5	82B6	82B7	82B8	82B9	82BA	82BB	82BC	82BD	82BE	82BF
0x2C0	82C0	82C1	82C2	82C3	82C4	82C5	82C6	82C7	82C8	82C9	82CA	82CB	82CC	82CD	82CE	82CF
0x2D0	82D0	82D1	82D2	82D3	82D4	82D5	82D6	82D7	82D8	82D9	82DA	82DB	82DC	82DD	82DE	82DF
0x2E0	82E0	82E1	82E2	82E3	82E4	82E5	82E6	82E7	82E8	82E9	82EA	82EB	82EC	82ED	82EE	82EF
0x2F0	82F0	82F1	82F2	82F3	82F4	82F5	82F6	82F7	82F8	82F9	82FA	82FB	82FC	82FD	82FE	82FF
0x300	8300	8301	8302	8303	8304	8305	8306	8307	8308	8309	830A	830B	830C	830D	830E	830F
0x310	8310	8311	8312	8313	8314	8315	8316	8317	8318	8319	831A	831B	831C	831D	831E	831F
0x320	8320	8321	8322	8323	8324	8325	8326	8327	8328	8329	832A	832B	832C	832D	832E	832F
0x330	8330	8331	8332	8333	8334	8335	8336	8337	8338	8339	833A	833B	833C	833D	833E	833F

7.3.2 AAAA

Address: | Columns: 16 | Address Radix: Hexadecimal | Value Radix: Hexadecimal

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x0	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x10	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x20	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x30	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x40	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x50	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x60	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x70	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x80	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x90	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0xA0	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0xB0	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0xC0	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0xD0	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0xE0	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0xF0	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x100	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x110	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x120	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x130	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x140	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x150	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x160	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x170	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x180	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x190	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x1A0	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x1B0	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x1C0	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x1D0	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x1E0	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x1F0	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x200	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x210	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x220	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x230	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x240	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x250	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x260	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x270	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x280	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x290	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x2A0	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x2B0	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x2C0	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x2D0	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x2E0	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x2F0	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x300	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x310	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x320	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
0x330	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA

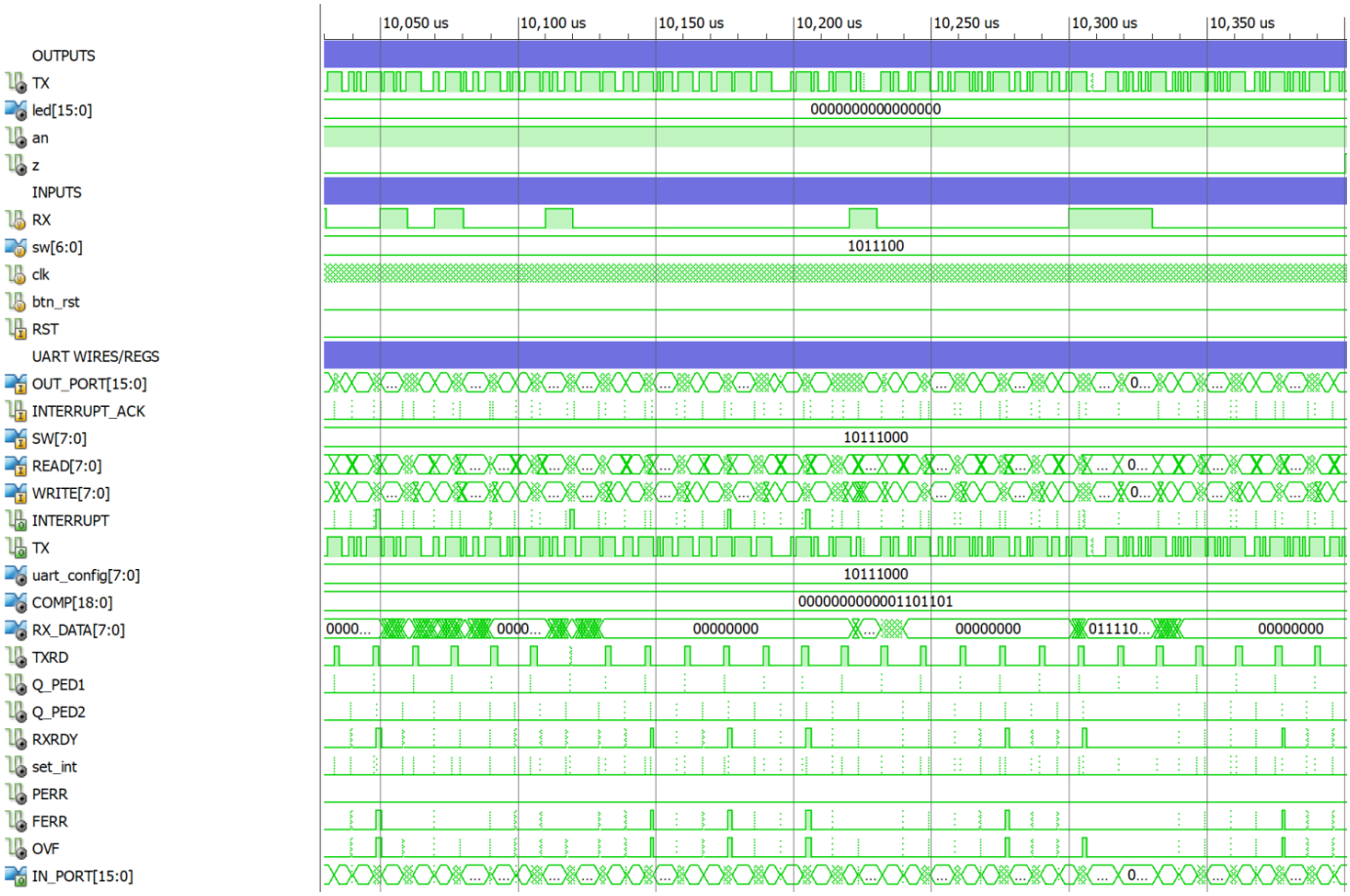
7.3.2 5555

Address: | Columns: 16 | Address Radix: Hexadecimal | Value Radix: Hexadecimal

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x0	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x10	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x20	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x30	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x40	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x50	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x60	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x70	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x80	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x90	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0xA0	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0xB0	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0xC0	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0xD0	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0xE0	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0xF0	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x100	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x110	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x120	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x130	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x140	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x150	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x160	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x170	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x180	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x190	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x1A0	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x1B0	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x1C0	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x1D0	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x1E0	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x1F0	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x200	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x210	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x220	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x230	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x240	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x250	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x260	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x270	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x280	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x290	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x2A0	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x2B0	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x2C0	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x2D0	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x2E0	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x2F0	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x300	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x310	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x320	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
0x330	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555

7.4 TSI Chip Level Test

Integration the Technology Specific Module means that we have to make sure all the inputs and outputs are on working conditions and that they supposed to go down or up at the time that they're commanded.



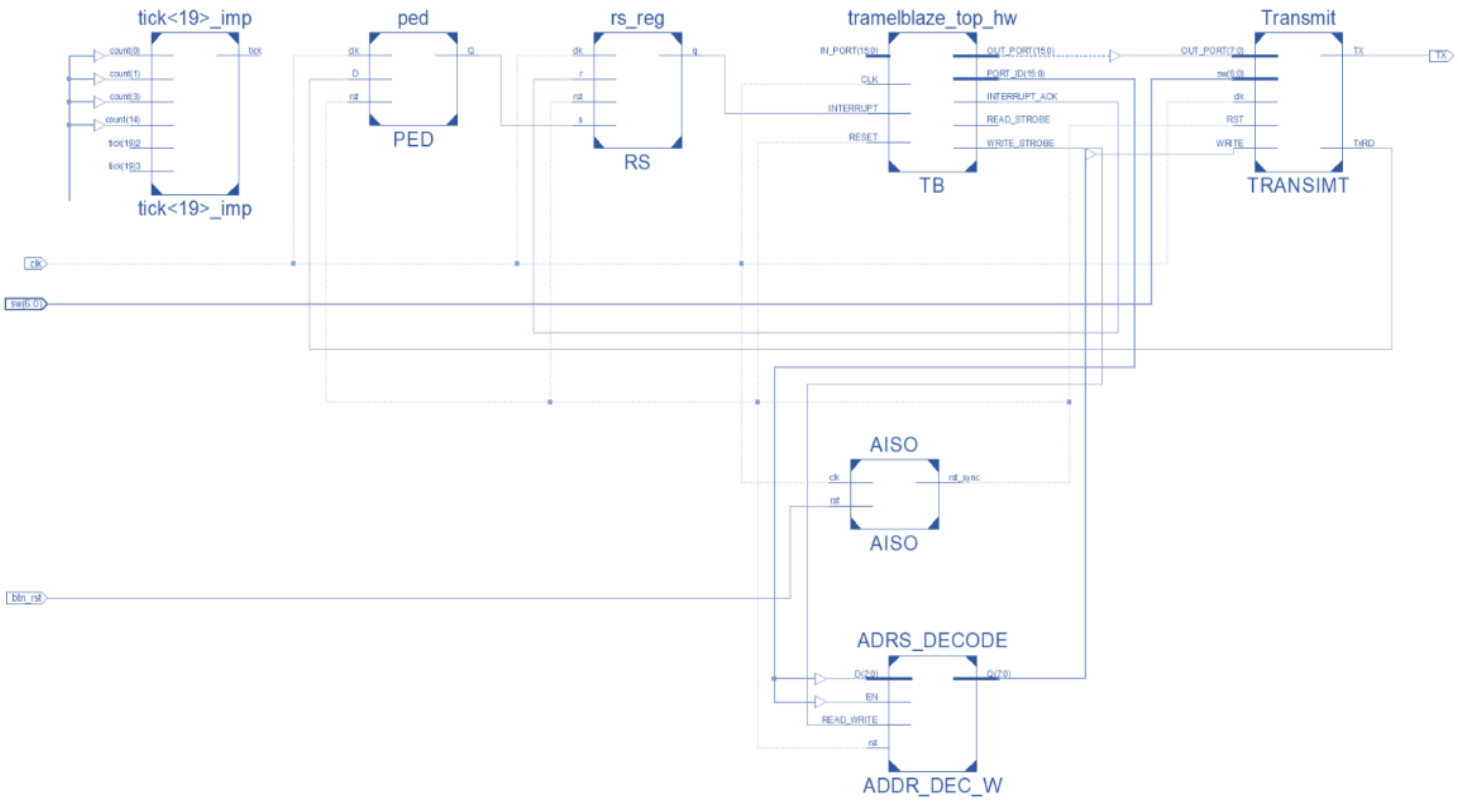
8. Chip Level Test

8.1. Test for Transmit Engine Chip Level

We test our Transmit engine by using a processor and using the TX engine itself to sends bits from the FPGA to a terminal window. We programmed our processor to convert bits into ASCII characters and to double check our correctness we made a counter that increases every new line four us to display this counter we used a binary to ASCII algorism that converts a binary counter register into an ASCII character.

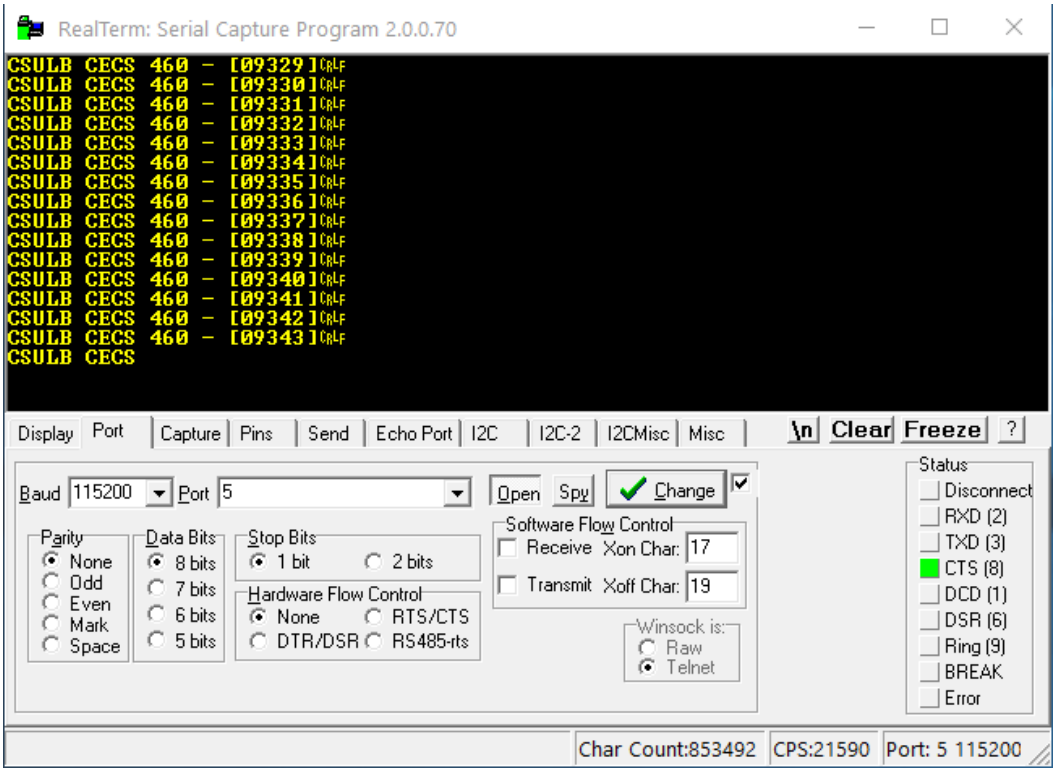
8.1.1. Test detail block diagram

We tested our design by using a 16-bit processor (tramelblaze) that sends data into the transmit engine and our engine sends the data out on that way that we specify using the switches in the FPGA. In this test we constantly send CSULB CECS 460 [i] <cr> <lf> where i is the line counter. We programmed our processor in assembly, the code is presented in the appendix.



8.1.2 Transmit on-chip test

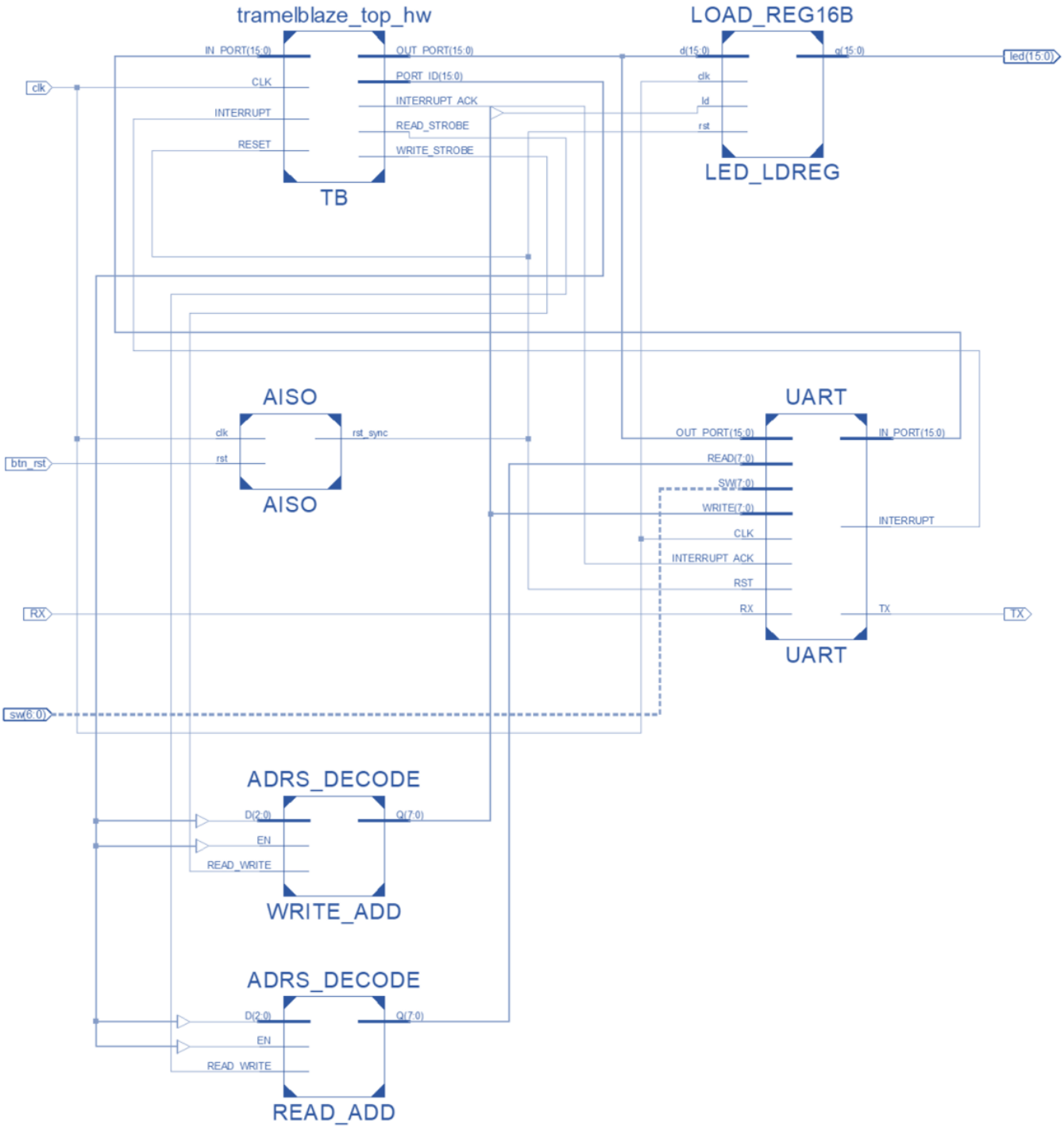
To test the design on the actual board we connect the board into a serial Terminal window, in this case, we used RealTerm because of the added functionality of changed data bits and baud rate speed while running. We can see that the asked functionality is meet and also, we can see when a new line is done (CrLf).



8.2 Test for Receive Engine Chip Level

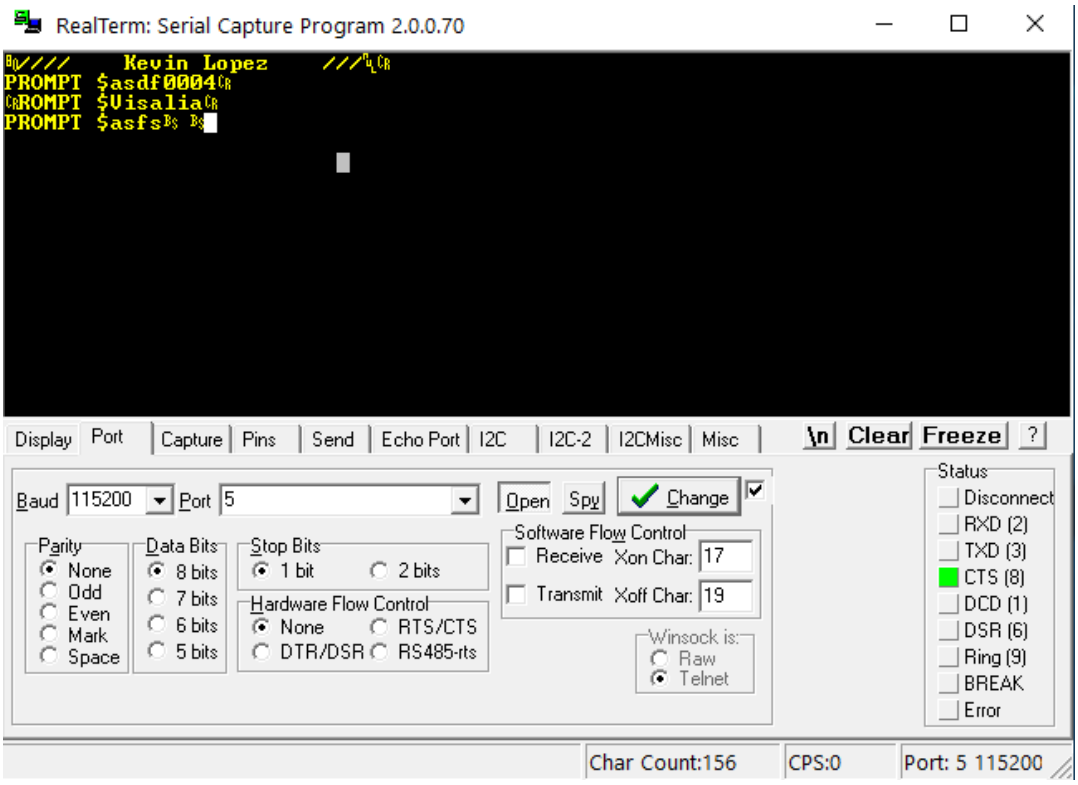
For the system on cheap test, we program the tramelblaze to get ASCII values and output them back on to the terminal window. We added a couple of more functions to keys * and @ and set the proper functionality for backspace and enter. When we press @, we receive the number of keys we have pressed and when we press *, we receive the user's hometown.

8.2.1. Test detail block diagram



8.2.2 UART on-chip Test

The terminal below shows how the specific functions of the UART work. Every character is being echoed, once AT character is pressed, we get the number of characters entered, once asterisks is pressed, we get the programmers hometown, an enter is performed and a backspace is also performed.

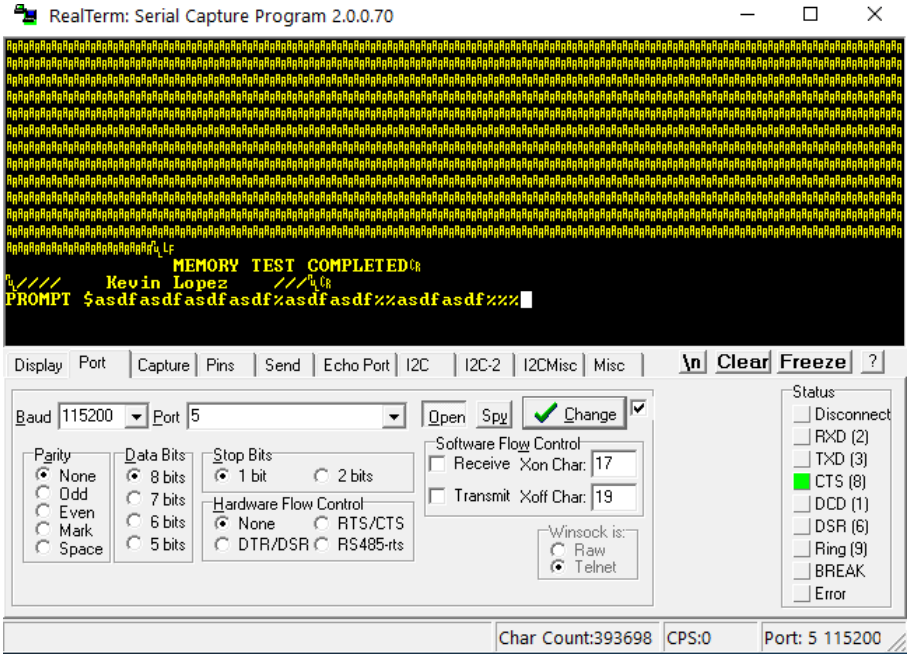


8.3 SRAM Memory Chip test

At the moment of disserting reset, there are about 5 seconds where there is a memory test performed. The first test is the address test where we can see the ASCII characters from 0 to FF. After that, there is a test for checking for empty register where we see UU's (5555) and then we see another ASCII character for the output of (AAAA).

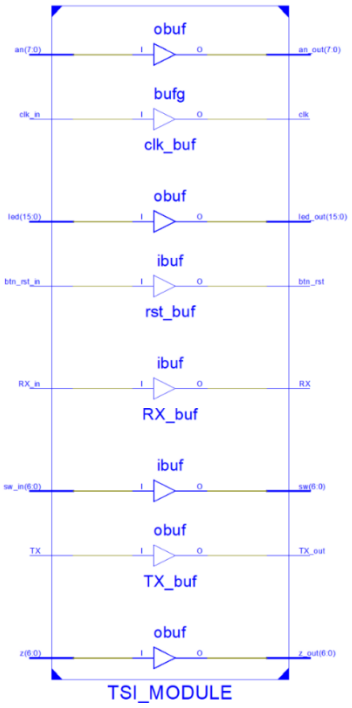
8.3.2 SRAM on-chip test

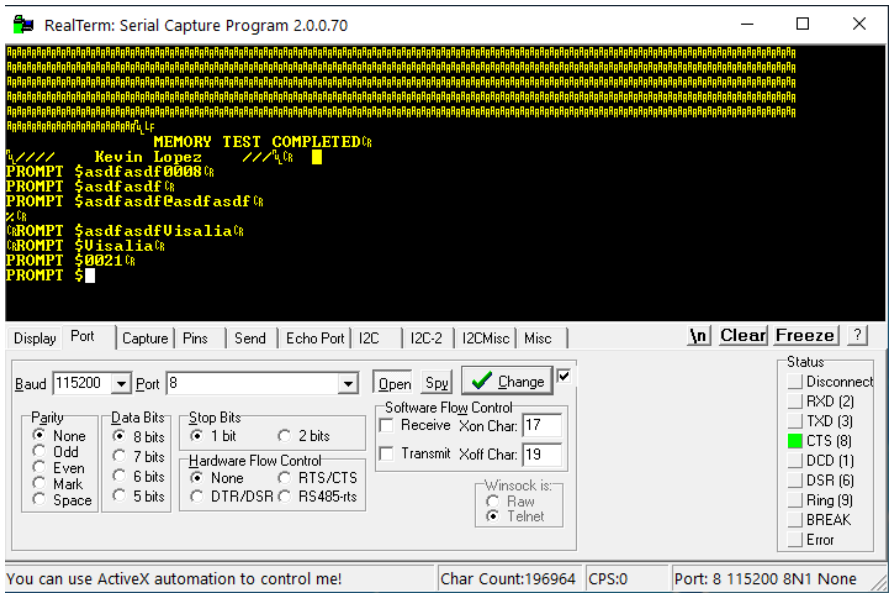
Just like for every chip test on this design we used RealTerm to look at the data being transmitted out of the design. We can see on the terminal below that once we press the percent key everything is pressed is being output it back, which is the data being dumped out of memory. We can also see the end of the last memory test performed (Writing AAAA on memory).



8.4 Technology Specific Instantiation Chip-Level Test

Adding a Buffer to our whole core means that we have to make sure the core still doing what it supposes to do. On this portion we verify that the system is still fully working. After synthesizing our code, we can test it again and in the terminal window below we can see that every single function is still working. The AT character still outputs the number of characters that have been entered, the asterisk character outputs the user hometown, the enter does a Character return line feed, and the percent character dumps all the content of the memory.




RealTerm: Serial Capture Program 2.0.0.70
MEMORY TEST COMPLETED
Kevin Lopez
PROMPT $asdfasdf0008
PROMPT $asdfasdf
PROMPT $asdfasdf@asdfasdf
PROMPT $asdfasdfVisalia
PROMPT $Visalia
PROMPT $0021
PROMPT $
Display
Port
Capture
Pins
Send
Echo Port
I2C
I2C-2
I2CMisc
Misc
\n
Clear
Freeze
Baud 115200
Port 8
Open
Spy
Change
Parity
None
Odd
Even
Mark
Space
Data Bits
8 bits
7 bits
6 bits
5 bits
Stop Bits
1 bit
2 bits
Hardware Flow Control
None
RTS/CTS
DTR/DSR
RS485-rts
Software Flow Control
Receive
Xon Char: 17
Transmit
Xoff Char: 19
Winsock is:
Raw
Telnet
Status
Disconnect
RXD (2)
TXD (3)
CTS (8)
DCD (1)
DSR (6)
Ring (9)
BREAK
Error
You can use ActiveX automation to control me!
Char Count:196964
CPS:0
Port: 8 115200 8N1 None

10. Appendix

10.1. Top

TOP.v Mon Mar 11 16:22:54 2019

```
1   `timescale 1ns / 1ps
2   //////////////////////////////////////////////////////////////////////////////////
3   //  File name: <TOP>                                                            //
4   //                                                                              //
5   //  Created by       <Kevin Lopez> on <March 12, 2019>.                         //
6   //  Copyright © 2018 <Kevin Lopez>. All rights reserved.                        //
7   //                                                                              //
8   //                                                                              //
9   //  In submitting this file for class work at CSULB                             //
10  //  I am confirming that this is my work and the work                           //
11  //  of no one else. In submitting this code I acknowledge that                  //
12  //  plagiarism in student project work is subject to dismissal.                 //
13  //  from the class                                                              //
14   //////////////////////////////////////////////////////////////////////////////////
15  module TOP(clk, btn_rst, sw, TX, led);
16     ////////////    INPUTS   //////////
17     input          clk, btn_rst;
18     input    [6:0] sw;
19
20     ////////////    OUTTTTPUTS  //////////
21     output   TX;
22     output   [15:0] led;
23
24     ////////////    WIRES/REGS  //////////
25     wire  [15:0]   OUT_PORT;
26     wire  [15:0]   PORTID;
27     wire  [7:0]    WRITE;
28     reg      [19:0] count, d;
29
30     assign led = (WRITE[1] && tick ) ? OUT_PORT : 16'b0;
31     assign load = (PORTID == 16'h0000) ? OUT_PORT: 15'b0;
32
33
34     AISO                AISO( .clk(clk), .rst(btn_rst), .rst_sync(RST) );
35
36
37     Transmit            TRANSIMT ( .OUT_PORT(OUT_PORT[7:0]), .WRITE( WRITE[0] ),
38                            .sw(sw), .clk(clk), .RST(RST), .TX(TX), .TxRD(TXRDY));
39
40
41     ped                 PED (.clk(clk), .rst(RST), .Q(PED_OU), .D(TXRDY) );
42
43
44     rs_reg              RS(.s(PED_OU), .r(INT_ACK) ,.clk(clk), .rst(RST),  .q(w3));
45
46
47     tramelblaze_top_hw  TB (.CLK(clk), .RESET(RST), .IN_PORT(), .INTERRUPT(w3),
48                         .OUT_PORT(OUT_PORT), .PORT_ID(PORTID), .READ_STROBE(READ),
49                         .WRITE_STROBE(WRITE_STO), .INTERRUPT_ACK(INT_ACK));
50
51
52     ADRS_DECODE         ADDR_DEC_R( .D(PORTID[2:0]), .EN(PORTID[15]),
53                                     .READ_WRITE(READ), .rst(RST), .Q());
54
55     ADRS_DECODE         ADDR_DEC_W( .D(PORTID[2:0]), .EN(PORTID[15]),
56                               .READ_WRITE(WRITE_STO), .rst(RST), .Q(WRITE)  );
57
```

Page 1

TOP.v Mon Mar 11 16:22:54 2019

```
58      assign tick = (count == 20'd_16372);
59
60      always @(*)
61         if(tick)      d = 20'b0;
62         else          d = count + 20'b1;
63
64      always @(posedge clk, posedge RST)
65         if(RST)     count <= 20'b0;
66         else          count <= d;
67
68
69
70   endmodule
71
72
```

10.2. Transmit Engine

Transmit.v Mon Mar 11 16:23:08 2019

```
1   `timescale 1ns / 1ps
2   //////////////////////////////////////////////////////////////////
3   //  File name: <Transmit>                                       //
4   //                                                              //
5   //  Created by       <Kevin Lopez> on <March 12, 2019>.         //
6   //  Copyright © 2018 <Kevin Lopez>. All rights reserved.        //
7   //                                                              //
8   //                                                              //
9   //  In submitting this file for class work at CSULB             //
10  //  I am confirming that this is my work and the work           //
11  //  of no one else. In submitting this code I acknowledge that  //
12  //  plagiarism in student project work is subject to dismissal. //
13  //  from the class                                              //
14   //////////////////////////////////////////////////////////////////
15
16  module Transmit(OUT_PORT, WRITE, sw, clk, RST, TX, TxRD );
17
18     ////////////    INPUTS   //////////
19     input [7:0] OUT_PORT;
20     input RST, clk, WRITE;
21     input [6:0] sw; //controls baut Rate
22
23     ////////////    OUTPUTS  //////////
24     output TX, TxRD;
25
26      ////////////   WIRES/REGS  //////////
27     wire [7:0] L_DATA;
28     wire [1:0] Q;
29     wire [18:0] COMP;
30
31
32     rs_set_reg     RS_SET( .s(DoneD1), .r(WRITE), .set(RST), .clk(clk), .q(TxRD) );
33
34     rs_reg         RS(.s(WRITE), .r(DONE) ,.clk(clk), .rst(RST),  .q(DoIT) );
35
36     LoadReg        LD_REG(.D(OUT_PORT ), .ld(WRITE), .Q(L_DATA), .clk(clk), .rst(RST));
37
38     register       R_WD1( .d(WRITE), .clk(clk), .rst(RST), .q(writeD1) );
39
40     Decode         DECO(.D(L_DATA), .Eight(sw[2]), .PEN(sw[1]), .OHEL(sw[0]), .Q(Q));
41
42     Shift_Register SHIFT_REG(.D({ Q, L_DATA[6:0],1'b0,1'b1}), .clk(clk),
43                    .LOAD(writeD1), .SH(BTU), .SDI(1'b1), .rst(RST), .SDO(TX));
44
45     BitCounter     BIT_COU( .doIt(DoIT), .BTU(BTU), .clk(clk), .rst(RST), .DONE(DONE));
46
47     register       R_DD1( .d(DONE), .clk(clk), .rst(RST), .q(DoneD1) ); //done d1
48
49     BitTimeCounter BIT_TIM_C( .doIt(DoIT), .comp(COMP), .clk(clk),
50                    .rst(RST), .BTU(BTU));
51
52     BauDECODER     BAU_DEC(.D(sw[6:3]), .rst(RST), .Q(COMP) );
53
54  endmodule
55
```

Page 1

10.3. RS_floop

rs_reg.v Mon Mar 11 16:23:33 2019

```
1   `timescale 1ns / 1ps
2   //////////////////////////////////////////////////////////////////////////////////
3   //  File name: <rs_reg>                                                     //
4   //                                                                          //
5   //  Created by       <Kevin Lopez> on <Feb 7, 2019>.                       //
6   //  Copyright © 2018 <Kevin Lopez>. All rights reserved.                   //
7   //                                                                          //
8   //                                                                          //
9   //  In submitting this file for class work at CSULB                         //
10  //  I am confirming that this is my work and the work                       //
11  //  of no one else. In submitting this code I acknowledge that              //
12  //  plagiarism in student project work is subject to dismissal.             //
13  //  from the class                                                          //
14   //////////////////////////////////////////////////////////////////////////////////
15  module rs_reg(s, r ,clk, rst,  q);
16     input clk, s, r, rst;
17     output reg q;
18
19     always @(posedge clk, posedge rst) begin
20        if (rst) q <= 1'b0;
21        else begin
22           if (s) q <= 1'b1;
23           else if (r) q <= 1'b0;
24           else if (r&s) q<=1'b0;
25           else q <= q;
26        end
27     end
28
29  endmodule
30
```

Page 1

10.4. Address Decoder

ADRS_DECODE.v Mon Mar 11 16:23:49 2019

```
1   `timescale 1ns / 1ps
2   //////////////////////////////////////////////////////////////////
3   //  File name: <ADRS_DECODE>                                    //
4   //                                                              //
5   //  Created by       <Kevin Lopez> on <March 12, 2019>.         //
6   //  Copyright © 2018 <Kevin Lopez>. All rights reserved.        //
7   //                                                              //
8   //                                                              //
9   //  In submitting this file for class work at CSULB             //
10  //  I am confirming that this is my work and the work           //
11  //  of no one else. In submitting this code I acknowledge that  //
12  //  plagiarism in student project work is subject to dismissal. //
13  //  from the class                                              //
14   //////////////////////////////////////////////////////////////////
15
16  module ADRS_DECODE( D, EN, READ_WRITE, rst, Q);
17
18     ////////////    INPUTS   //////////
19     input [2:0] D;
20     input EN, READ_WRITE, rst;
21
22     ////////////    OUTPUTS  ////////
23     output reg [7:0] Q;
24
25     always @(*) begin
26        Q = 8'b0;
27        case (~EN)
28           1'b1 : Q[D] = READ_WRITE;
29           default: Q = Q;
30        endcase
31     end
32
33
34  endmodule
35
36
```

Page 1

10.5. Loadable register

LoadReg.v Mon Mar 11 16:24:24 2019

```
1   `timescale 1ns / 1ps
2   //////////////////////////////////////////////////////////////////
3   //  File name: <LoadReg>                                        //
4   //                                                              //
5   //  Created by       <Kevin Lopez> on <March 12, 2019>.           //
6   //  Copyright © 2018 <Kevin Lopez>. All rights reserved.        //
7   //                                                              //
8   //                                                              //
9   //  In submitting this file for class work at CSULB             //
10  //  I am confirming that this is my work and the work           //
11  //  of no one else. In submitting this code I acknowledge that  //
12  //  plagiarism in student project work is subject to dismissal. //
13  //  from the class                                              //
14   //////////////////////////////////////////////////////////////////
15   module LoadReg(D, ld, Q, clk, rst );
16     input clk, rst, ld;
17     input [7:0] D;
18     output reg [7:0] Q;
19
20     always @(posedge clk, posedge rst)begin
21        if (rst)
22           Q <= 8'b0;
23        else begin
24           if (ld)
25              Q <= D;
26           else
27              Q <= Q;
28        end
29     end
30
31  endmodule
32
```

Page 1

10.6. Decoder

Decode.v Mon Mar 11 16:24:38 2019

```
1   `timescale 1ns / 1ps
2   //////////////////////////////////////////////////////////////////
3   //  File name: <Decode>                                         //
4   //                                                              //
5   //  Created by       <Kevin Lopez> on <March 12, 2019>.           //
6   //  Copyright © 2018 <Kevin Lopez>. All rights reserved.        //
7   //                                                              //
8   //                                                              //
9   //  In submitting this file for class work at CSULB             //
10  //  I am confirming that this is my work and the work           //
11  //  of no one else. In submitting this code I acknowledge that  //
12  //  plagiarism in student project work is subject to dismissal.  //
13  //  from the class                                              //
14   //////////////////////////////////////////////////////////////////
15
16
17  module Decode(D, Eight, PEN, OHEL, Q);
18     input [7:0] D;
19     input Eight, PEN, OHEL;
20     output reg [1:0] Q;
21
22     reg setEven, setOdd;
23     assign oddCheck =  ^D;
24
25     always@(*)begin
26        if ( oddCheck == 1'b1)begin //its odd
27           setEven = 1'b1;
28           setOdd = 1'b0; end
29        else begin
30           setOdd = 1'b1;
31           setEven = 1'b0; end
32     end
33
34
35     always@(*)
36        case({Eight,PEN,OHEL})
37           3'b000: Q =  2'b11;
38           3'b001: Q =  2'b11;
39           3'b010: Q = {1'b1, setEven } ; //even parity
40           3'b011: Q = {1'b1, setOdd  } ; //odd parity
41           3'b100: Q = {1'b1, D[7] };
42           3'b101: Q = {1'b1,  D[7] };
43           3'b110: Q = { setEven , D[7] }; //even parity
44           3'b111: Q = { setOdd  , D[7] }; //odd parity
45           default: Q = 2'b11;
46     endcase
47
48
49
50  endmodule
51
```

Page 1

10.7. Shift Register

Shift_Register.v Mon Mar 11 16:24:51 2019

```
1   `timescale 1ns / 1ps
2   //////////////////////////////////////////////////////////////////
3   //  File name: <Shift_Register>                                   //
4   //                                                                //
5   //  Created by       <Kevin Lopez> on <March 12, 2019>.           //
6   //  Copyright © 2018 <Kevin Lopez>. All rights reserved.          //
7   //                                                                //
8   //                                                                //
9   //  In submitting this file for class work at CSULB               //
10  //  I am confirming that this is my work and the work             //
11  //  of no one else. In submitting this code I acknowledge that    //
12  //  plagiarism in student project work is subject to dismissal.   //
13  //  from the class                                                //
14   //////////////////////////////////////////////////////////////////
15
16
17  module Shift_Register(D, LOAD, SH, SDI, clk, rst, SDO);
18      input [10:0] D;
19      input LOAD, SH, SDI, rst, clk;
20
21      output SDO;
22
23      reg [10:0] shiftReg;
24
25     always @(posedge clk, posedge rst) begin
26        if (rst)    shiftReg <= 11'H_7FF; else
27        if (LOAD)   shiftReg <= D; else
28        if (SH)     shiftReg <= { SDI ,shiftReg[10:1] };
29        else shiftReg <= shiftReg;
30     end
31
32     assign SDO = shiftReg[0];
33
34
35
36
37  endmodule
38
```

Page 1

10.8. Bit Counter

BitCounter.v Mon Mar 11 16:25:08 2019

```
1
2   `timescale 1ns / 1ps
3   //////////////////////////////////////////////////////////////////
4   //  File name: <BitCounter>                                     //
5   //                                                                //
6   //  Created by       <Kevin Lopez> on <March 12, 2019>.            //
7   //  Copyright © 2018 <Kevin Lopez>. All rights reserved.          //
8   //                                                                //
9   //                                                                //
10  //  In submitting this file for class work at CSULB               //
11  //  I am confirming that this is my work and the work             //
12  //  of no one else. In submitting this code I acknowledge that    //
13  //  plagiarism in student project work is subject to dismissal.   //
14  //  from the class                                                //
15   //////////////////////////////////////////////////////////////////
16
17
18
19  module BitCounter( doIt, BTU, clk, rst, DONE );
20
21      input doIt, BTU, clk, rst;
22      output DONE;
23
24      reg [3:0] d, count;
25
26
27      assign DONE = (count == 4'd_11 );
28
29      always@(*)
30          case({doIt, BTU})
31              2'b00: d = 4'b0;
32              2'b01: d = 4'b0;
33              2'b10: d = count;
34              2'b11: d = count + 4'b1;
35              default  d = count;
36           endcase
37
38
39     always @(posedge clk, posedge rst)
40           if(rst) count <= 4'b0; else
41           count <= d;
42
43
44  endmodule
45
46
47
48
```

Page 1

10.9. Bit Time Counter

BIT_TIME_COUNTER.v Mon Mar 11 16:25:28 2019

```
1   `timescale 1ns / 1ps
2   //////////////////////////////////////////////////////////////////
3   //  File name: <BitTimeCounter>                                   //
4   //                                                               //
5   //  Created by       <Kevin Lopez> on <March 12, 2019>.          //
6   //  Copyright © 2018 <Kevin Lopez>. All rights reserved.         //
7   //                                                               //
8   //                                                               //
9   //  In submitting this file for class work at CSULB              //
10  //  I am confirming that this is my work and the work            //
11  //  of no one else. In submitting this code I acknowledge that   //
12  //  plagiarism in student project work is subject to dismissal.  //
13  //  from the class                                               //
14   //////////////////////////////////////////////////////////////////
15
16
17
18  module BitTimeCounter( doIt, comp, clk, rst, BTU );
19
20      input doIt, clk, rst;
21      input [18:0] comp;
22      output BTU ;
23
24      reg [18:0] d, count;
25
26
27      assign BTU  = (count == comp );
28
29      always @(posedge clk, posedge rst)
30          if (rst) count <= 19'b0;  else
31           if (doIt) begin
32              if ( BTU) count <= 19'b0;
33              else      count <= count + 1;
34           end //if doIT
35           else count <= 19'b0;
36
37
38  endmodule
39
40
41
42
```

Page 1

10.10. Baud Rate decoder

BauDECODER.v Mon Mar 11 16:25:41 2019

```
1   `timescale 1ns / 1ps
2   //////////////////////////////////////////////////////////////////
3   //  File name: <BitTimeCounter>                                 //
4   //                                                              //
5   //  Created by       <Kevin Lopez> on <March 12, 2019>.         //
6   //  Copyright © 2018 <Kevin Lopez>. All rights reserved.        //
7   //                                                              //
8   //                                                              //
9   //  In submitting this file for class work at CSULB             //
10  //  I am confirming that this is my work and the work           //
11  //  of no one else. In submitting this code I acknowledge that  //
12  //  plagiarism in student project work is subject to dismissal. //
13  //  from the class                                              //
14   //////////////////////////////////////////////////////////////////
15
16
17
18
19  module BauDECODER(D, rst, Q );
20
21     input [3:0] D;
22     input rst;
23     output reg [18:0] Q;
24
25     always@(*)
26        casez({rst, D})
27           5'b1_???? : Q = 19'd_333_333;
28           5'b0_0000 : Q = 19'd_333_333; //300
29           5'b0_0001 : Q = 19'd_83_333;  //1200
30           5'b0_0010 : Q = 19'd_41667;   //2400
31           5'b0_0011 : Q = 19'd_208333;  //4800
32           5'b0_0100 : Q = 19'd_10417;   //9600
33           5'b0_0101 : Q = 19'd_5208;    //19200
34           5'b0_0110 : Q = 19'd_2604;    //38400
35           5'b0_0111 : Q = 19'd_1736;    //57600
36           5'b0_1000 : Q = 19'd_868;     //115200
37           5'b0_1001 : Q = 19'd_434;     //230400
38           5'b0_1010 : Q = 19'd_217;     //460800
39           5'b0_1011 : Q = 19'd_109;     //921600
40           default :  Q = 19'd_333_333;
41     endcase
42
43  endmodule
44
```

Page 1

11.12 Posedge Edge Detect

ped.v **Sun May 12 14:04:40 2019**

```
1   `timescale 1ns / 1ps
2   //////////////////////////////////////////////////////////////////////////////////
3   //  File name: <ped>                                                            //
4   //                                                                              //
5   //  Created by       <Kevin Lopez> on <March 12, 2019>.                         //
6   //  Copyright © 2018 <Kevin Lopez>. All rights reserved.                        //
7   //                                                                              //
8   //                                                                              //
9   //  In submitting this file for class work at CSULB                             //
10  //  I am confirming that this is my work and the work                           //
11  //  of no one else. In submitting this code I acknowledge that                  //
12  //  plagiarism in student project work is subject to dismissal.                 //
13  //  from the class                                                              //
14  //////////////////////////////////////////////////////////////////////////////////
15
16
17
18  module ped( // it will return 1 when 1,0 (posedge edge dectect)
19      input clk,
20      input rst,
21      output Q,
22      input D
23      );
24
25      register n1 (  .clk(clk), .rst(rst), .d(D), .q(q1) );
26      register n2 (  .clk(clk), .rst(rst), .d(q1), .q(q2) );
27
28      assign Q = q1& ~q2;
29
30  endmodule
31
```

Page 1

11.13 AISO

AISO.v Sun May 12 14:07:29 2019

```
1   `timescale 1ns / 1ps
2   //////////////////////////////////////////////////////////////////////////////////
3   //  File name: <AISO>                                                           //
4   //                                                                              //
5   //  Created by       <Kevin Lopez> on <March 12, 2019>.                           //
6   //  Copyright © 2018 <Kevin Lopez>. All rights reserved.                        //
7   //                                                                              //
8   //                                                                              //
9   //  In submitting this file for class work at CSULB                             //
10  //  I am confirming that this is my work and the work                           //
11  //  of no one else. In submitting this code I acknowledge that                  //
12  //  plagiarism in student project work is subject to dismissal.                 //
13  //  from the class                                                              //
14   //////////////////////////////////////////////////////////////////////////////////
15
16  module AISO( clk, rst, rst_sync );
17      input clk, rst;
18      output rst_sync;
19
20      register k1 (  .clk(clk), .rst(rst), .d(1'b1), .q(w1) );
21      register k2 (  .clk(clk), .rst(rst), .d(w1), .q(q) );
22
23      assign rst_sync = ~q;
24
25
26  endmodule
27
```

10.11. Transmit engine Software

```
1.  ; Kevin Lopez
2.  ; CECS  460
3.  ; BIN TO ASCII
4.  DEBUGG                  EQU         R6
5.  COUNT                   EQU         R2          ;REGISTER THAT KEEPS COUNT OF THE VALUES
6.  ZEROS                   EQU         0000                ;ZEROS
7.  ONE                                 EQU         0001         ;ONE
8.  ISR_ADR                 EQU         0300        ;ADDRESS FOR ISR
9.  TEMP_REG      EQU       R4                      ;FOR STORING VALUES
10. WALKING_LED   EQU                   R3          ;FOR LEDS IN MAIN LOOP
11. PTR                                 EQU         R8          ;POINTER FOR OUTPUTING AND STORING
12. TEMP_OUT      EQU                   R9          ;TEMP REGISTER FOR OUTPUTING
13.
14.
15. START
16.               ENINT
17.               LOAD      COUNT, 0000
18.               LOAD      R3, 0001                                    ; WALKING ONES
19.               LOAD      TEMP_REG, 0000
20.               LOAD      PTR, ZEROS
21.               LOAD      TEMP_OUT, ZEROS
22.               LOAD      DEBUGG, ZEROS
23.
24. ;CSULB CECS 460 - [COUNT] <CR><LF>
25. ;43 53 55 4c 42  43 45 43 53 20 34 36 30 20 2d 20 5b   BIN2ASCII     5D 0D     0A
26.
27.               LOAD      TEMP_REG, 0043
28.               STORE     TEMP_REG, (PTR) ;0
29.               ADD       PTR, ONE
30.
31.               LOAD      TEMP_REG, 0053
32.               STORE     TEMP_REG, (PTR) ; 1
33.               ADD       PTR, ONE
34.
35.               LOAD      TEMP_REG, 0055
36.               STORE     TEMP_REG, (PTR) ; 2
37.               ADD       PTR, ONE
38.
39.               LOAD      TEMP_REG, 004C
40.               STORE     TEMP_REG, (PTR) ;3
41.               ADD       PTR, ONE
42.
43.               LOAD      TEMP_REG, 0042
44.               STORE     TEMP_REG, (PTR)  ;4
45.               ADD       PTR, ONE
46.
47.               LOAD      TEMP_REG, 0020
48.               STORE     TEMP_REG, (PTR) ;5
49.               ADD       PTR, ONE
50.
51.               LOAD      TEMP_REG, 0043
52.               STORE     TEMP_REG, (PTR) ;6
53.               ADD       PTR, ONE
54.
55.               LOAD      TEMP_REG, 0045
56.               STORE     TEMP_REG, (PTR) ;7
57.               ADD       PTR, ONE
58.
59.               LOAD      TEMP_REG, 0043
60.               STORE     TEMP_REG, (PTR) ;8
61.               ADD       PTR, ONE
62.
63.               LOAD      TEMP_REG, 0053
64.               STORE     TEMP_REG, (PTR) ;9
65.               ADD       PTR, ONE
```

```
66.
67.             LOAD        TEMP_REG, 0020
68.             STORE       TEMP_REG, (PTR) ;a
69.             ADD         PTR, ONE
70.
71.             LOAD        TEMP_REG, 0034
72.             STORE       TEMP_REG, (PTR) ;b
73.             ADD         PTR, ONE
74.
75.             LOAD        TEMP_REG, 0036
76.             STORE       TEMP_REG, (PTR) ;c
77.             ADD         PTR, ONE
78.
79.             LOAD        TEMP_REG, 0030
80.             STORE       TEMP_REG, (PTR) ;d
81.             ADD         PTR, ONE
82.
83.             LOAD        TEMP_REG, 0020
84.             STORE       TEMP_REG, (PTR) ;e
85.             ADD         PTR, ONE
86.
87.             LOAD        TEMP_REG, 002D
88.             STORE       TEMP_REG, (PTR) ;f
89.             ADD         PTR, ONE
90.
91.             LOAD        TEMP_REG, 0020
92.             STORE       TEMP_REG, (PTR) ;10
93.             ADD         PTR, ONE
94.
95.             LOAD        TEMP_REG, 005B
96.             STORE       TEMP_REG, (PTR) ;11
97.             ADD         PTR, ONE
98.
99. ;           CALL BIN2ASCII ;STORE [16-12]
100.
101.            LOAD        PTR, 0017
102.            LOAD        TEMP_REG, 005D
103.            STORE       TEMP_REG, (PTR) ;17
104.            ADD         PTR, ONE
105.
106.            LOAD        TEMP_REG, 000D
107.            STORE       TEMP_REG, (PTR) ;18
108.            ADD         PTR, ONE
109.
110.            LOAD        TEMP_REG, 000A
111.            STORE       TEMP_REG, (PTR) ;19
112.            LOAD        PTR, ZEROS
113.
114.
115.
116.;////////////////////////////////////////////////////////
117.;                                       MAIN LOOP (walking ones)
118.;////////////////////////////////////////////////////////
119.MAIN
120.            OUTPUT      WALKING_LED, 0001
121.            RL                  WALKING_LED                     ; ROTATE LEFT
122.
123.            ADD         R7, ONE             ; FOR DELAY
124.            ADD         R7, ONE
125.            SUB         R7, ONE
126.            SUB         R7, ONE
127.            ADD         R7, ONE
128.            SUB         R7, ONE
129.
130.            JUMP        MAIN
131.;******************************************************
132.
133.
134.
135.
136.;////////////////////////////////////////////////////////
137.;                                                   BIN2ASCII
138.;////////////////////////////////////////////////////////
139.
140.BIN2ASCII
141.            LOAD        PTR, 0012  ;SET PTR IN BETWEN           460[PTR] <><>
142.            LOAD        RE, COUNT  ; R2 IS COUNTER
143.
144.            LOAD        RD, 2710   ;10,000
145.            CALL        FIND                ; FIND HOW MANY RETURNS
146.            ADD                 RB, 0030    ; CONVERT TO ASCII
```

```
147.              STORE       RB, (PTR)   ;12
148.              ADD         PTR, ONE
149.
150.              LOAD        RD, 03B8   ; 1,000
151.              CALL        FIND                ; FIND HOW MANY RETURNS
152.              ADD                     RB, 0030   ; CONVERT TO ASCII
153.              STORE       RB, (PTR) ;13
154.              ADD         PTR, ONE
155.
156.              LOAD        RD, 0064   ; 100
157.              CALL        FIND                ; FIND HOW MANY RETURNS
158.              ADD                     RB, 0030   ; CONVERT TO ASCII
159.              STORE       RB, (PTR)  ;14
160.              ADD         PTR, ONE
161.
162.              LOAD        RD, 000A   ; 10
163.              CALL        FIND                ; FIND HOW MANY RETURNS
164.              ADD                     RB, 0030   ; CONVERT TO ASCII
165.              STORE       RB, (PTR)  ;15
166.              ADD         PTR, ONE
167.
168.              ADD                     RE, 0030   ; 1
169.              STORE       RE, (PTR)  ;16
170.              ADD         PTR, ONE
171.              RETURN
172.                          ;   FIND
173. FIND
174.              LOAD        RB, ZEROS
175. AGN
176.              SUB         RE, RD
177.              COMP        RE, 8000            ; CHECK MSB FOR NEGATIVE NUMBER
178.              JUMPC       POSITIVE            ; IF NOT NEGATIVE JUMP
179.              ADD                     RE, RD                          ; IF NEGATIVE R3<- R3 + RD
180.              RETURN
181.
182. POSITIVE
183.              ADD         RB, 0001                        ;INCREMENT RB
184.              JUMP        AGN                             ;REPET
185. ;*************************************************************
186.
187. CALL_BIN2ASCII
188.              CALL        BIN2ASCII
189.              ADD         COUNT, ONE
190.              LOAD        PTR, 0012 ; SET POINTER BACK TO INTITAL POSITION TO DISPLAY THE NUMBER
191.              RETURN
192.
193.
194. ;////////////////////////////////////////////////////////
195. ;                                                       ISR
196. ;////////////////////////////////////////////////////////
197.                          ADDRESS 0300
198. ISR
199.              ADD         DEBUGG, 0001          ; JUST TO MAKE SURE WE ENTERING ISR
200.              COMP        PTR, 0012                       ;CHECK IF IS TIME TO DISPLAY COUNT
201.              CALLZ       CALL_BIN2ASCII
202.
203.              FETCH       TEMP_OUT, (PTR)
204.              OUTPUT      TEMP_OUT, 0000
205.              ADD         PTR, 0001  ; INCREMENT PTR
206.              COMP        PTR, 001A  ; CHEKC IF COUNT HAS MAX VAL
207.              JUMPZ       RESET_PTR             ; IF ZERO RESET PTR TO O
208.              RETEN
209.
210. RESET_PTR
211.              LOAD        PTR, ZEROS
212.              RETEN
213.
214.
215.
216.              ADDRESS 0FFE
217. ENDIT
218.              JUMP ISR
219. END
```

TOP LEVEL RECEIVE

TOP.v Tue Apr 16 17:01:30 2019

```
1   `timescale 1ns / 1ps
2
3   //////////////////////////////////////////////////////////////////
4   //  File name: <TOP>                                            //
5   //                                                              //
6   //  Created by       <Kevin Lopez> on <March 12, 2019>.         //
7   //  Copyright Â© 2018 <Kevin Lopez>. All rights reserved.        //
8   //                                                              //
9   //                                                              //
10  //  In submitting this file for class work at CSULB             //
11  //  I am confirming that this is my work and the work           //
12  //  of no one else. In submitting this code I acknowledge that  //
13  //  plagiarism in student project work is subject to dismissal. //
14  //  from the class                                              //
15   //////////////////////////////////////////////////////////////////
16
17  module TOP(clk, btn_rst, sw, led, TX, RX  );
18     ////////////    INPUTS   //////////
19     input       clk, btn_rst;
20     input [6:0] sw;
21     input       RX;
22
23     ////////////    OUTPUTS  //////////
24     output [15:0]  led;
25     output         TX;
26
27     ////////////    WIRES/REGS  //////////
28     wire [7:0]  WRITE, READ;
29     wire [15:0] PORT_ID, IN_PORT, OUT_PORT;
30
31     AISO                AISO( .clk(clk), .rst(btn_rst), .rst_sync(RST) );
32
33     UART                UART( .RX(RX), .CLK(clk), .RST(RST), .SW({sw, 1'b0}),
34                               .INTERRUPT_ACK(INTERRUPT_ACK), .WRITE(WRITE[7:0]), .READ
   (READ[7:0]),
35                               .OUT_PORT(OUT_PORT),
36                               .INTERRUPT(INTERRUPT), .TX(TX), .IN_PORT( IN_PORT ) );
37
38     tramelblaze_top_hw  TB( .CLK(clk), .RESET(RST), .IN_PORT(IN_PORT),
39                               .INTERRUPT(INTERRUPT), .OUT_PORT(OUT_PORT[15:0]),
40                               .PORT_ID(PORT_ID[15:0]), .READ_STROBE(READ_STROBE),
41                               .WRITE_STROBE(WRITE_STROBE),
42                               .INTERRUPT_ACK(INTERRUPT_ACK)  );
43
44     ADRS_DECODE         READ_ADD( .D(PORT_ID[2:0]), .EN(PORT_ID[15]),
45                                   .READ_WRITE(READ_STROBE), .Q(READ));
46
47     ADRS_DECODE         WRITE_ADD(  .D(PORT_ID[2:0]), .EN(PORT_ID[15]),
48                                     .READ_WRITE(WRITE_STROBE),
49                                      .Q(WRITE) );
50
51     LOAD_REG16B         LED_REG( .clk(clk), .rst(rst), .ld(WRITE[1]) ,.d(OUT_PORT) ,
52                                  .q(led) );
53  endmodule
54
```

UART

UART.v Tue Apr 16 17:10:11 2019

```
1   `timescale 1ns / 1ps
2   //////////////////////////////////////////////////////////////////////////////////
3   //  File name: <UART>                                                           //
4   //                                                                              //
5   //  Created by       <Kevin Lopez> on <March 12, 2019>.                          //
6   //  Copyright © 2018 <Kevin Lopez>. All rights reserved.                        //
7   //                                                                              //
8   //                                                                              //
9   //  In submitting this file for class work at CSULB                             //
10  //  I am confirming that this is my work and the work                           //
11  //  of no one else. In submitting this code I acknowledge that                  //
12  //  plagiarism in student project work is subject to dismissal.                 //
13  //  from the class                                                              //
14   //////////////////////////////////////////////////////////////////////////////////
15  module UART(  RX, CLK, RST, SW, INTERRUPT_ACK, WRITE, READ, OUT_PORT, INTERRUPT, TX,
    IN_PORT );
16     ////////////////////////////// INPUTS ////////////////
17     input CLK, RST, RX, INTERRUPT_ACK;
18     input [15:0] OUT_PORT;
19     input [7:0] SW;
20     input [7:0] READ, WRITE;
21     ////////////////////////////// OUTPUTS ////////////////
22     output INTERRUPT, TX;
23     output reg [15:0]  IN_PORT;
24
25     wire [7:0] uart_config;
26     wire [18:0] COMP;
27     wire [7:0] RX_DATA;
28
29      LoadReg       LOADR(.D(OUT_PORT[7:0]), .ld(WRITE[2]), .Q(uart_config),
30                          .clk(CLK), .rst(RST) );
31
32      BauDECODER    BAU_DEC(.D(uart_config[7:4] ), .rst(RST), .Q(COMP) );
33
34      Transmit      TRASMIT(.OUT_PORT(OUT_PORT[7:0]), .WRITE(WRITE[0]), .COMP(COMP),
35                    .sw(uart_config[3:1]), .clk(CLK), .RST(RST), .TX(TX), .TxRD(TXRD) );
36
37      ped           PEDT( .clk(CLK), .rst(RST), .Q(Q_PED1), .D(TXRD)  );
38      ped           PEDR( .clk(CLK), .rst(RST), .Q(Q_PED2), .D(RXRDY)  );
39
40     assign set_int = Q_PED2 | Q_PED1;
41
42     rs_reg         INTRE_RS_REG(.s(set_int), .r(INTERRUPT_ACK), .clk(CLK),
43                                 .rst(RST),  .q(INTERRUPT));
44
45     Receive_Engine RECEIVE( .RX(RX), .clk(CLK), .comp(COMP), .RST(RST), .EIGHT(SW[2]),
46                        .PEN(SW[1]), .OHEL(SW[0]), .RXRDY(RXRDY), .PERR(PERR),
47                        .FERR(FERR), .OVF(OVF),.Q_TB(RX_DATA), .READ0( READ[0]) );
48
49     always@(*)
50        case(READ[2:0])
51           2: IN_PORT = { 11'b0 , OVF, FERR, PERR, TXRD, RXRDY};
52           1: IN_PORT = { 8'b0 , RX_DATA };
53           4: IN_PORT = { 8'b0 ,SW };
54           default: IN_PORT = 16'b0;
55        endcase
56
```

Page 1

Receive Engine

Receive_Engine.v **Tue Apr 16 17:14:07 2019**

```
1   `timescale 1ns / 1ps
2   //////////////////////////////////////////////////////////////////////
3   //  File name: <Receive_Engine>                                     //
4   //                                                                  //
5   //  Created by       <Kevin Lopez> on <April 16, 2019>.             //
6   //  Copyright © 2018 <Kevin Lopez>. All rights reserved.            //
7   //                                                                  //
8   //                                                                  //
9   //  In submitting this file for class work at CSULB                 //
10  //  I am confirming that this is my work and the work               //
11  //  of no one else. In submitting this code I acknowledge that      //
12  //  plagiarism in student project work is subject to dismissal.     //
13  //  from the class                                                  //
14   //////////////////////////////////////////////////////////////////////
15  module Receive_Engine( RX, clk, comp, RST, EIGHT, PEN, OHEL, RXRDY, PERR, FERR, OVF,
    Q_TB, READ0
16  );
17
18     input RX, clk, RST, EIGHT, PEN, OHEL, READ0;
19     input [18:0] comp;
20
21     output reg PERR, RXRDY, FERR, OVF;
22     output [7:0] Q_TB; //input into the tramelBlaze
23
24     wire [19:0] K;
25     wire [7:0]  temp;
26
27     reg [1:0] pst, nst;
28     reg       START, nstart, DOIT, ndoIt;
29     reg [3:0] numberBits;
30     reg [18:0] count;
31     reg [3:0] dB, countB;
32     reg [9:0] sift_reg, remap_reg;
33     reg STOP_BIT_SELECT;
34
35
36     ////////////////////////////////////////////////////////////////////////////
37     ////////////////////////////// STATE MATCHINE //////////////////////////////
38     ////////////////////////////////////////////////////////////////////////////
39     always @(*)
40        casez( {RX, BTU, DONE, pst} )
41           5'b1??_00   : {nst, nstart, ndoIt} = {4'b00_00};
42           5'b0??_00   : {nst, nstart, ndoIt} = {4'b01_11};
43           5'b1??_01   : {nst, nstart, ndoIt} = {4'b00_00};
44           5'b00?_01   : {nst, nstart, ndoIt} = {4'b01_11};
45           5'b01?_01   : {nst, nstart, ndoIt} = {4'b10_01};
46           5'b??0_10   : {nst, nstart, ndoIt} = {4'b10_01};
47           5'b??1_10   : {nst, nstart, ndoIt} = {4'b00_00};
48           default     : {nst, nstart, ndoIt} =  4'b0;
49        endcase
50
51
52        always@(posedge clk, posedge RST)begin
53           if (RST) begin
54               pst <= 2'b00;
55               START <= 1'b0;
56               DOIT <= 1'b0;
```

```
171      assign parity = (paritySET ^ PARITY_BIT_SEL) & PEN & DONE;
172
173      always@(posedge clk, posedge RST)
174         if (RST) PERR <= 1'b0; else
175         if ( parity) PERR <= 1'b1; else
176         if ( READ0 ) PERR <= 1'b0;
177         else PERR <= PERR;
178
179      ////////////////////////////////////////////////////////////////////////////
180      ////////////////////////////////////       FERR  REG ////////////////////
181      ////////////////////////////////////////////////////////////////////////////
182      assign temp2 = DONE & STOP_BIT_SELECT;
183
184      always@(posedge clk, posedge RST)
185            if (RST) FERR <= 1'b0; else
186            if ( temp2) FERR <= 1'b1; else
187            if ( READ0 ) FERR <= 1'b0;
188            else FERR <= FERR;
189
190      ////////////////////////////////////////////////////////////////////////////
191      ////////////////////////////////   ONF   REG   ////////////////////////////
192      ////////////////////////////////////////////////////////////////////////////
193      assign temp3 = DONE & RXRDY;
194
195      always@(posedge clk, posedge RST)
196            if (RST) OVF <= 1'b0; else
197            if ( temp3) OVF <= 1'b1; else
198            if ( READ0 ) OVF <= 1'b0;
199            else OVF <= OVF;
200
201
202
203
204
205
206   endmodule
207
```

Load Register 16bits

LOAD_REG16B.v Tue Apr 16 17:19:49 2019

```
1   `timescale 1ns / 1ps
2   //////////////////////////////////////////////////////////////////////////////////
3   //  File name: <LOAD_REG16B>                                                    //
4   //                                                                              //
5   //  Created by       <Kevin Lopez> on <April 16, 2019>.                           //
6   //  Copyright © 2018 <Kevin Lopez>. All rights reserved.                        //
7   //                                                                              //
8   //                                                                              //
9   //  In submitting this file for class work at CSULB                             //
10  //  I am confirming that this is my work and the work                           //
11  //  of no one else. In submitting this code I acknowledge that                  //
12  //  plagiarism in student project work is subject to dismissal.                 //
13  //  from the class                                                              //
14
15  module LOAD_REG16B ( clk, rst, ld ,d , q );
16
17      input ld, clk, rst;
18      input [15:0] d;
19      output reg [15:0] q;
20
21     always@(posedge clk, posedge rst) begin
22           if (rst) q <= 1'b0;
23           else begin
24              if (ld) q <= d;
25              else q <= q;
26           end
27     end
28
29
30  endmodule
31
```

Received Test Bench

Receive_tf2.v **Mon May 13 13:05:38 2019**

```
1   `timescale 1ns / 1ps
2
3   //////////////////////////////////////////////////////////////////////////////////
4   //  File name: <Receive_tf2>                                                    //
5   //                                                                              //
6   //  Created by       <Kevin Lopez> on <March 12, 2019>.                           //
7   //  Copyright © 2018 <Kevin Lopez>. All rights reserved.                        //
8   //                                                                              //
9   //                                                                              //
10  //  In submitting this file for class work at CSULB                             //
11  //  I am confirming that this is my work and the work                           //
12  //  of no one else. In submitting this code I acknowledge that                  //
13  //  plagiarism in student project work is subject to dismissal.                 //
14  //  from the class                                                              //
15   //////////////////////////////////////////////////////////////////////////////////
16
17  module Receive_tf2;
18
19     // Inputs
20     reg RX;
21     reg clk;
22     reg [18:0] comp;
23     reg RST;
24     reg EIGHT;
25     reg PEN;
26     reg OHEL;
27     reg READ0;
28
29     // Outputs
30     wire RXRDY;
31     wire PERR;
32     wire FERR;
33     wire OVF;
34     wire [6:0] Q_TB;
35
36     integer i;
37
38     // Instantiate the Unit Under Test (UUT)
39     Receive_Engine uut (
40        .RX(RX),
41        .clk(clk),
42        .comp(comp),
43        .RST(RST),
44        .EIGHT(EIGHT),
45        .PEN(PEN),
46        .OHEL(OHEL),
47        .RXRDY(RXRDY),
48        .PERR(PERR),
49        .FERR(FERR),
50        .OVF(OVF),
51        .Q_TB(Q_TB),
52        .READ0(READ0)
53     );
54
55     always #5 clk = ~ clk;
56
57     initial begin
```

Page 1

```
58          // Initialize Inputs
59          RX = 1;
60          clk = 0;
61          comp = 109;
62          RST = 1;
63          EIGHT = 0;
64          PEN = 0;
65          OHEL = 0;
66          READ0 = 0;
67
68          // Wait 100 ns for global reset to finish
69          #100; RST = 0;
70
71          for (i = 0; i < 2; i = i + i)begin
72             RX = 0;
73             #1098;
74          end
75
76          for ( i = 0 ; i < 8; i = + 1) begin
77          RX = ~ RX;
78          #1098;
79          end
80
81          $finish;
82
83          // Add stimulus here
84
85       end
86
87    endmodule
88
89
```

Transmit engine Test Bench

TRANSMIT_TF2.v Mon May 13 13:08:25 2019

```
1   `timescale 1ns / 1ps
2   //////////////////////////////////////////////////////////////////
3   //  File name: <TRANSMIT_TF2>                                   //
4   //                                                              //
5   //  Created by       <Kevin Lopez> on <March 12, 2019>.           //
6   //  Copyright © 2018 <Kevin Lopez>. All rights reserved.        //
7   //                                                              //
8   //                                                              //
9   //  In submitting this file for class work at CSULB             //
10  //  I am confirming that this is my work and the work           //
11  //  of no one else. In submitting this code I acknowledge that  //
12  //  plagiarism in student project work is subject to dismissal. //
13  //  from the class                                              //
14   //////////////////////////////////////////////////////////////////
15  module TRANSMIT_TF2;
16
17     // Inputs
18     reg [7:0] OUT_PORT;
19     reg WRITE;
20     reg [18:0] COMP;
21     reg [2:0] sw;
22     reg clk;
23     reg RST;
24
25     // Outputs
26     wire TX;
27     wire TxRD;
28
29     time     t ;       //variable to control write period
30     integer  f;        //file reference
31     reg      flip;     //flag to design\
32     integer  ChangeDataIn;
33
34
35     // Instantiate the Unit Under Test (UUT)
36     Transmit uut (
37        .OUT_PORT(OUT_PORT),
38        .WRITE(WRITE),
39        .COMP(COMP),
40        .sw(sw),
41        .clk(clk),
42        .RST(RST),
43        .TX(TX),
44        .TxRD(TxRD)
45     );
46
47        always #5 clk = ~clk;
48
49        always #t begin
50           flip = !flip;
51           $fwrite(f, TX, " ");
52        end
53
54
55        always #11_000 begin
56           WRITE = 1;
57           #500 WRITE = 0;
```

Page 1

TRANSMIT_TF2.v Mon May 13 13:09:41 2019

```
58            ChangeDataIn = ChangeDataIn + 1;
59          end
60
61        always@(*)
62              case(ChangeDataIn)
63                0: OUT_PORT = 8'h_2A; //Output for *
64                1: OUT_PORT = 8'h_40; //Output for @
65                2: OUT_PORT = 8'h_0D; //Output for CR
66                3: OUT_PORT = 8'h_08; //Output for Backspace
67                4: OUT_PORT = 8'h_30; //Output for Character
68                5: OUT_PORT = 8'h_31; //Output for Character
69                6: OUT_PORT = 8'h_32; //Output for Character
70                default : OUT_PORT = 8'h_40; //Output for Character
71          endcase
72
73        initial begin
74            // Initialize Inputs
75            OUT_PORT = 0;
76            sw = 0;
77            clk = 0;
78            WRITE = 0;
79            RST = 1;
80            COMP = 109;
81            ChangeDataIn = 0;
82             f = $fopen("output.txt");
83             t = 1098;
84
85            // Wait 100 ns for global reset to finish
86            #100; RST = 0;
87            OUT_PORT = 8'H_0A;
88            WRITE = 1;
89
90            #10;
91            WRITE = 0;
92
93            // Add stimulus here
94
95            @(posedge uut.BTU) begin
96                #6 $write( "%b", TX);
97            end
98          #50_000_000
99          $fclose(f);
100         $finish;
101
102       end
103
104
105
106   endmodule
107
108
```

SRAM TOP MODULE

TOP.v Tue Apr 30 12:37:53 2019

```
1   `timescale 1ns / 1ps
2
3   //////////////////////////////////////////////////////////////////
4   //  File name: <TOP>                                            //
5   //                                                              //
6   //  Created by       <Kevin Lopez> on <April 30, 2019>.         //
7   //  Copyright Â© 2018 <Kevin Lopez>. All rights reserved.        //
8   //                                                              //
9   //                                                              //
10  //  In submitting this file for class work at CSULB             //
11  //  I am confirming that this is my work and the work           //
12  //  of no one else. In submitting this code I acknowledge that  //
13  //  plagiarism in student project work is subject to dismissal. //
14  //  from the class                                              //
15   //////////////////////////////////////////////////////////////////
16
17  module TOP(clk, btn_rst, sw, led, TX, RX, an, z );
18     ////////////    INPUTS   //////////
19     input       clk, btn_rst;
20     input [6:0] sw;
21     input       RX;
22
23     ////////////    OUTPUTS  //////////
24     output [15:0]  led;
25     output         TX;
26     output [7:0]   an; // 8 anodes
27     output [6:0]   z;  // 7 pixels
28
29
30     ////////////    WIRES/REGS  //////////
31     wire [7:0]  WRITE, READ, FLAGS;
32     wire [15:0] PORT_ID, IN_PORT, IN_PORT1, IN_PORT2 , OUT_PORT, SRAM_OUT ;
33
34     ////////////////TO SELECT WHAT DATA TO GO INTO THE PROCESSOR //////////
35     assign   IN_PORT = ( (READ == 1)||(READ == 2)||(READ == 4))? IN_PORT1: IN_PORT2;
36
37     AISO                AISO( .clk(clk), .rst(btn_rst), .rst_sync(RST) );
38
39     UART                UART( .RX(RX), .CLK(clk), .RST(RST), .SW({sw, 1'b0}),
40                               .INTERRUPT_ACK(INTERRUPT_ACK), .WRITE(WRITE[7:0]),
41                               .READ(READ[7:0]),  .OUT_PORT(OUT_PORT),
42                               .INTERRUPT(INTERRUPT), .TX(TX), .IN_PORT( IN_PORT1 )
43                                );
44
45     tramelblaze_top_sim  TB( .CLK(clk), .RESET(RST), .IN_PORT(IN_PORT),
46                               .INTERRUPT(INTERRUPT), .OUT_PORT(OUT_PORT[15:0]),
47                               .PORT_ID(PORT_ID[15:0]), .READ_STROBE(READ_STROBE),
48                               .WRITE_STROBE(WRITE_STROBE),
49                               .INTERRUPT_ACK(INTERRUPT_ACK)  );
50
51     ADRS_DECODE          READ_ADD( .D(PORT_ID[2:0]), .EN(PORT_ID[15]),
52                                    .READ_WRITE(READ_STROBE), .Q(READ));
53
54     ADRS_DECODE          WRITE_ADD(  .D(PORT_ID[2:0]), .EN(PORT_ID[15]),
55                                      .READ_WRITE(WRITE_STROBE),
56                                       .Q(WRITE) ) ;
57
```

Page 1

Memory Interface

MEMORY_INTERFACE_BLOCK.v Tue Apr 30 12:40:31 2019

```
1   `timescale 1ns / 1ps
2   //*************************************************************************//
3   //  File name: <MEMORY_INTERFACE_BLOCK>                                    //
4   //                                                                         //
5   //  Created by       <Kevin Lopez> on <April 30, 2019>.                    //
6   //  Copyright Â© 2018 <Kevin Lopez>. All rights reserved.                   //
7   //                                                                         //
8   //                                                                         //
9   //  In submitting this file for class work at CSULB                        //
10  //  I am confirming that this is my work and the work                      //
11  //  of no one else. In submitting this code I acknowledge that             //
12  //  plagiarism in student project work is subject to dismissal.            //
13  //  from the class                                                         //
14   //*************************************************************************//
15
16  module MEMORY_INTERFACE_BLOCK(clk, wea, addr, din, dout );
17     input             clk, wea;
18     input     [14:0]  addr;
19     input     [15:0]  din;
20     output    [15:0]  dout;
21
22
23     SRAM              SRAM (.clka(clk), // input clka
24                         .wea( wea) , // input [0 : 0] wea  write enable  WHENEVER
    PORTID15 IS ACTIVE
25                         .addra(addr), // input [14 : 0] addra  portID 14:0
26                         .dina(din), // input [15 : 0] dina
27                         .douta(dout) // output [15 : 0] douta
28                          );
29
30
31
32
33  endmodule
34
```

Full UART and Memory Software

```
1.  ; Kevin Lopez
2.  ; CECS  460
3.  ; PROJECT 4
4.
5.  ;In beginning display Memory Test, banner and give a prompt
6.
7.  ;CR         sends cursor to the start of the next line
8.  ;BS         erase the character in front of the cursor - care must be taken to not erase the prompt
9.  ;*          hometown name
10. ;@                  will output number of characters received since reset, then reset
11. ;           Other characters should be echoed. At the 40th character we would make a new line
12. %           Will output everything user has pressed.
13.
14. DATA_REG    EQU       R0        ;Data form RX
15. STATUS_REG  EQU       R1        ; STATUS FOR OVF, FERR, PERR, TXRD, RXRDY
16.
17. COUNT                 EQU       R2        ;REGISTER THAT KEEPS COUNT OF characters
18. WALKING_LED EQU                 R3        ;FOR LEDS IN MAIN LOOP
19. TEMP_REG    EQU       R4        ;FOR STORING ACII VALUES
20. SWITCH_REG  EQU                 R5        ;SWITCHES
21. OUT_SET               EQU       R6  ;FOR COMUNICATION INSIDE OF UART
22. DELAY_REG   EQU       R7        ;REGISTER FOR DELAY
23. PTR                             EQU       R8        ;POINTER FOR OUTPUTING
24. SET_DUMP    EQU                 R9        ;TEMP REGISTER FOR OUTPUTING
25. SP                              EQU       RA  ;STACK  POINTER FOR SRAM
26. ERROR_REG   EQU       RF        ;FOR DEGUGGING PORPOSES.
27.
28.
29. ZERO                  EQU       0000 ;ZERO
30. ONE                             EQU       0001 ;ONE
31.
32. ASTERISK    EQU                 002A ; *
33. AT                              EQU       0040 ; @
34. CR                    EQU       000D ; <CR>
35. BACKSPACE   EQU       007F ; <-
36. PERCENT               EQU                 0025 ; %
37.
38.
39.
40.
41. START
42.             LOAD      COUNT,            0000
43.             LOAD      ERROR_REG,                    ZERO
44.             LOAD      WALKING_LED,      ZERO
45.             LOAD      SP,                           8000
46.             LOAD      SWITCH_REG,       ZERO
47.             LOAD      WALKING_LED,      0001                                        ; WALKING ONES
48.             LOAD      TEMP_REG,         0000
49.             LOAD      PTR, ZERO
50.             LOAD      SET_DUMP,         ZERO
51.
52.
53. ;////////////////////////////////////////////////////
54. ;                                                   HOMETOWN            0 - 9
55. ;////////////////////////////////////////////////////
56. ;Visalia  0-9
57. ;56 69 73 61 6c 69 61 0D 0A
58.
59.             LOAD      TEMP_REG, 0056
60.             STORE     TEMP_REG, 0000 ;0
61.
62.             LOAD      TEMP_REG, 0069
63.             STORE     TEMP_REG, 0001 ; 1
64.
65.             LOAD      TEMP_REG, 0073
66.             STORE     TEMP_REG, 0002 ; 2
67.
```

```
68.             LOAD      TEMP_REG, 0061
69.             STORE     TEMP_REG, 0003 ;4
70.
71.             LOAD      TEMP_REG, 006C
72.             STORE     TEMP_REG, 0004 ;5
73.
74.             LOAD      TEMP_REG, 0069
75.             STORE     TEMP_REG, 0005 ;5
76.
77.             LOAD      TEMP_REG, 0061
78.             STORE     TEMP_REG, 0006 ;5
79.
80.             LOAD      TEMP_REG, 000D
81.             STORE     TEMP_REG, 0007 ;7
82.
83.             LOAD      TEMP_REG, 000A
84.             STORE     TEMP_REG, 0008 ;8
85.
86.
87.
88. ;////////////////////////////////////////////////////
89. ;                   BANNER                      0A - 24
90. ;////////////////////////////////////////////////////
91.
92. ; ///////////    Kevin Lopez    ////////////  [0A-25]
93. ; //2f 2f 2f 2f 2f 2f 2f 2f 2f 2f 20 20 20 20 4b
94. ;// 65 76 69 6e 20 4c 6f 70 65 7a 20 20 20 20 2f 2f 2f
95. ;//2f 2f 2f 2f 2f 2f 2f 2f
96.             LOAD      TEMP_REG, 002F ;/
97.             STORE     TEMP_REG, 000A
98.
99.             LOAD      TEMP_REG, 002F ;/
100.                       STORE     TEMP_REG, 000B
101.
102.                      LOAD      TEMP_REG, 002F ;/
103.                      STORE     TEMP_REG, 000C
104.
105.                      LOAD      TEMP_REG, 002F ;/
106.                      STORE     TEMP_REG, 000D
107.
108.                      LOAD      TEMP_REG, 0020 ; " "
109.                      STORE     TEMP_REG, 000E
110.
111.                      LOAD      TEMP_REG, 0020 ; " "
112.                      STORE     TEMP_REG, 000F
113.
114.                      LOAD      TEMP_REG, 0020 ; " "
115.                      STORE     TEMP_REG, 0010
116.
117.                      LOAD      TEMP_REG, 0020 ; " "
118.                      STORE     TEMP_REG, 0011
119.
120.                      LOAD      TEMP_REG, 004B ;K
121.                      STORE     TEMP_REG, 0012
122.
123.                      LOAD      TEMP_REG, 0065 ;e
124.                      STORE     TEMP_REG, 0013
125.
126.                      LOAD      TEMP_REG, 0076 ;v
127.                      STORE     TEMP_REG, 0014
128.
129.                      LOAD      TEMP_REG, 0069 ;i
130.                      STORE     TEMP_REG, 0015
131.
132.                      LOAD      TEMP_REG, 006E ;n
133.                      STORE     TEMP_REG, 0016
134.
135.                      LOAD      TEMP_REG, 0020 ; " "
136.                      STORE     TEMP_REG, 0017
137.
138.                      LOAD      TEMP_REG, 004C ;L
```

```
139.                    STORE     TEMP_REG, 0018
140.
141.                    LOAD      TEMP_REG, 006F ;o
142.                    STORE     TEMP_REG, 0019
143.
144.                    LOAD      TEMP_REG, 0070 ;p
145.                    STORE     TEMP_REG, 001A
146.
147.                    LOAD      TEMP_REG, 0065 ;e
148.                    STORE     TEMP_REG, 001B
149.
150.                    LOAD      TEMP_REG, 007A ;z
151.                    STORE     TEMP_REG, 001C
152.
153.                    LOAD      TEMP_REG, 0020 ; " "
154.                    STORE     TEMP_REG, 001D
155.
156.                    LOAD      TEMP_REG, 0020 ; " "
157.                    STORE     TEMP_REG, 001E
158.
159.                    LOAD      TEMP_REG, 0020 ; " "
160.                    STORE     TEMP_REG, 001F
161.
162.                    LOAD      TEMP_REG, 0020 ; " "
163.                    STORE     TEMP_REG, 0020
164.
165.                    LOAD      TEMP_REG, 002F ;/
166.                    STORE     TEMP_REG, 0021
167.
168.                    LOAD      TEMP_REG, 002F ;/
169.                    STORE     TEMP_REG, 0022
170.
171.                    LOAD      TEMP_REG, 002F ;/
172.                    STORE     TEMP_REG, 0023
173.
174.                    LOAD      TEMP_REG, 002F ;/
175.                    STORE     TEMP_REG, 0024
176.
177.  ;////////////////////////////////////
178.  ;         BACKSPACE                    26 - 28
179.  ;////////////////////////////////////
180.                    LOAD      TEMP_REG, 0008 ; BACK
181.                    STORE     TEMP_REG, 0026
182.
183.                    LOAD      TEMP_REG, 0020 ; " "
184.                    STORE     TEMP_REG, 0027
185.
186.                    LOAD      TEMP_REG, 0008 ; BACK
187.                    STORE     TEMP_REG, 0028
188.
189.  ;////////////////////////////////////
190.  ;                           NEW LINE                                29 - 2A
191.  ;////////////////////////////////////
192.                    LOAD      TEMP_REG, 000D ; CR
193.                    STORE     TEMP_REG, 0029
194.
195.                    LOAD      TEMP_REG, 000A ; LF
196.                    STORE     TEMP_REG, 002A
197.
198.
199.  ;////////////////////////////////////
200.  ;         PROMPT                       2B - 33
201.  ;////////////////////////////////////
202.  ; PROMPT $     50 52 4f 4d 50 54 20 24
203.
204.                    LOAD      TEMP_REG, 0050 ;P
205.                    STORE     TEMP_REG, 002B
206.
207.                    LOAD      TEMP_REG, 0052 ;R
208.                    STORE     TEMP_REG, 002C
209.
```

```
210.                    LOAD     TEMP_REG, 004f ;O
211.                    STORE    TEMP_REG, 002D
212.
213.                    LOAD     TEMP_REG, 004d ;M
214.                    STORE    TEMP_REG, 002E
215.
216.                    LOAD     TEMP_REG, 0050 ;P
217.                    STORE    TEMP_REG, 002F
218.
219.                    LOAD     TEMP_REG, 0054 ;T
220.                    STORE    TEMP_REG, 0030
221.
222.                    LOAD     TEMP_REG, 0020 ;" "
223.                    STORE    TEMP_REG, 0031
224.
225.                    LOAD     TEMP_REG, 0024 ;$
226.                    STORE    TEMP_REG, 0032
227.
228.
229.
230.  ;/////////////////////////////////////
231.  ;         MEMORY TEST DONE   33 - 49
232.  ;/////////////////////////////////////
233.  ;// 4d 45 4d 4f 52 59 20 54 ;;;;;45 53
234.  ;//54 20 43 4f 4d 50 4c 45 54 45 44 20 0a
235.
236.                    LOAD     TEMP_REG, 000A ; LF
237.                    STORE    TEMP_REG, 0033
238.
239.
240.                    LOAD     TEMP_REG, 004d
241.                    STORE    TEMP_REG, 0034
242.
243.                    LOAD     TEMP_REG, 0045
244.                    STORE    TEMP_REG, 0035
245.
246.                    LOAD     TEMP_REG, 004d
247.                    STORE    TEMP_REG, 0036
248.
249.                    LOAD     TEMP_REG, 004f
250.                    STORE    TEMP_REG, 0037
251.
252.                    LOAD     TEMP_REG, 0052
253.                    STORE    TEMP_REG, 0038
254.
255.                    LOAD     TEMP_REG, 0059
256.                    STORE    TEMP_REG, 0039
257.
258.                    LOAD     TEMP_REG, 0020
259.                    STORE    TEMP_REG, 003A
260.
261.                    LOAD     TEMP_REG, 0054
262.                    STORE    TEMP_REG, 003B
263.
264.                    LOAD     TEMP_REG, 0045
265.                    STORE    TEMP_REG, 003C
266.
267.                    LOAD     TEMP_REG, 0053
268.                    STORE    TEMP_REG, 003D
269.
270.                    LOAD     TEMP_REG, 0054
271.                    STORE    TEMP_REG, 003E
272.
273.                    LOAD     TEMP_REG, 0020
274.                    STORE    TEMP_REG, 003F
275.
276.                    LOAD     TEMP_REG, 0043
277.                    STORE    TEMP_REG, 0040
278.
279.                    LOAD     TEMP_REG, 004F
280.                    STORE    TEMP_REG, 0041
```

```
281.
282.                     LOAD      TEMP_REG, 004D
283.                     STORE     TEMP_REG, 0042
284.
285.                     LOAD      TEMP_REG, 0050
286.                     STORE     TEMP_REG, 0043
287.
288.                     LOAD      TEMP_REG, 004C
289.                     STORE     TEMP_REG, 0044
290.
291.                     LOAD      TEMP_REG, 0045
292.                     STORE     TEMP_REG, 0045
293.
294.                     LOAD      TEMP_REG, 0054
295.                     STORE     TEMP_REG, 0046
296.
297.                     LOAD      TEMP_REG, 0045
298.                     STORE     TEMP_REG, 0047
299.
300.                     LOAD      TEMP_REG, 0044
301.                     STORE     TEMP_REG, 0048
302.
303.                     LOAD      TEMP_REG, 000D ; CR
304.                     STORE     TEMP_REG, 0049
305.
306.                     LOAD      TEMP_REG, 000A ; LF
307.                     STORE     TEMP_REG, 004A
308.
309.
310.
311.
312.                     ENINT      ; ENABLE INTERUPTS
313.
314.
315.  ;////////////////////////////////// ///////////////////
316.  ;                          MEMORY TEST
317.  ;////////////////////////////////////////////////////////
318.
319.
320.
321.  ;/////////////               1st MEMORY TEST                        ///////////
322.  ;/////////////      WRITE ADDRESS ON ADDRESS     ///////////
323.  STORE_ADD
324.                     OUTPUT    SP,       (SP)
325.                     ADD       SP,       ONE
326.                     COMP      SP,       FFFF
327.                     JUMPC     STORE_ADD ;IF FFFE IS GREATER THAN SP REPEAT AGAIN
328.
329.                     LOAD      PTR,               8000
330.                     LOAD      SP,                8000
331.                     LOAD      SET_DUMP, 000A ;OUTPUT ADDRESS
332.                     LOAD      TEMP_REG, ZERO
333.                     OUTPUT    TEMP_REG, ZERO ;SET FLAG
334.
335.
336.  DELAYT
337.                     COMP      TEMP_REG,          FFFF
338.                     JUMPNZ    DELAYT             ;WAIT TILL SP IS DONE OUTPUTING
339.
340.                     LOAD      PTR,               8000
341.                     LOAD      SP,                8000
342.
343.
344.
345.  ;/////////////               2nd MEMORY TEST                                    ///////////
346.  ;/////////////      WRITE AAAA n 5555 ON ADDRESS          ///////////
347.  STORE_FIVES
348.                     LOAD      TEMP_REG, 5555
349.                     OUTPUT    TEMP_REG,          (SP)  ;WRITE DATA IN MEMROY
350.                     ADD       SP,                ONE
351.                     COMP      SP,                          FFFF
```

```
352.                    JUMPC     STORE_FIVES
353.
354.
355.                    LOAD      SP,               8000
356.                    LOAD      PTR,              8000
357.                    LOAD      SET_DUMP,         0008 ;    OUTPUT THE 5555
358.                    LOAD      TEMP_REG, ZERO
359.                    OUTPUT    TEMP_REG, ZERO    ;SET FLAG
360.
361.  DELAYZ
362.                    COMP      TEMP_REG,         FFFF
363.                    JUMPNZ    DELAYZ            ;WAIT TILL SP IS DONE OUTPUTING
364.
365.                    LOAD      PTR,              8000
366.                    LOAD      SP,               8000
367.
368.
369.  ;//////////////    LOAD AAAA //////////
370.  STORE_As
371.                    LOAD      TEMP_REG, AAAA
372.                    OUTPUT    TEMP_REG,         (SP)
373.                    ADD       SP,       ONE
374.                    COMP      SP,                         FFFF
375.                    JUMPC     STORE_As
376.
377.                    LOAD      SP,               8000
378.                    LOAD      PTR,              8000
379.                    LOAD      SET_DUMP,         0009 ; OUTPUT  AAAA
380.                    LOAD      TEMP_REG, ZERO
381.                    OUTPUT    TEMP_REG, ZERO ;SET FLAG
382.
383.  DELAYO
384.                    COMP      TEMP_REG,         FFFF
385.                    JUMPNZ    DELAYO            ;WAIT TILL SP IS DONE OUTPUTING
386.
387.
388.                    LOAD      SP,               8000
389.
390.  ;DISPLAY_MEM_TEST_DONE
391.                    LOAD      PTR,              0033
392.                    LOAD      OUT_SET,  0007
393.                    LOAD      TEMP_REG,         ZERO
394.                    OUTPUT    TEMP_REG, ZERO
395.
396.  DELAYF
397.                    COMP      TEMP_REG,         FFFF
398.                    JUMPNZ    DELAYF            ;WAIT TILL SP IS DONE OUTPUTING
399.
400.
401.  ;////////////////////////////////////////////MEMORY TEST DONE
402.
403.
404.
405.
406.                    CALL      DIS_BANNER
407.
408.
409.  ;////////////////////////////////////////////////////////
410.  ;                                          MAIN LOOP (walking ones)
411.  ;////////////////////////////////////////////////////////
412.                    ; DISPLAY BANNER IN BEGINNING  CR LF AND PROMPT
413.
414.  MAIN
415.                    ADD       DELAY_REG,        ONE       ; FOR DELAY
416.                    COMP      DELAY_REG,        FFFE      ;CHECK FOR OVERLOAD
417.                    JUMPNZ    MAIN                        ;JUMPS IF VALUE IS LESS THANFD
418.
419.                    OUTPUT    WALKING_LED,      0001
420.                    RL                WALKING_LED         ; ROTATE LEFT
421.                    LOAD      DELAY_REG,        ZERO
422.
```

```
423.
424.                     JUMP      MAIN
425.
426.
427.
428.
429.  ;////////////////////////////////////////////////////
430.  ;                                                    BIN2ASCII
      34-38
431.  ;////////////////////////////////////////////////////
432.
433.  BIN2ASCII
434.                     LOAD      RE,       COUNT     ; R2 IS COUNTER
435.
436.                     ;LOAD     RD,       2710      ;10,000
437.                     ;CALL     FIND                ; FIND HOW MANY RETURNS
438.                     ;ADD                RB,       0030      ; CONVERT TO ASCII
439.                     ;STORE    RB,       0034       ;34
440.
441.                     LOAD      RD,       03B8      ; 1,000
442.                     CALL      FIND                ; FIND HOW MANY RETURNS
443.                     ADD                 RB,       0030      ; CONVERT TO ASCII
444.                     STORE     RB,       0034      ; 35
445.
446.                     LOAD      RD,       0064      ; 100
447.                     CALL      FIND                ; FIND HOW MANY RETURNS
448.                     ADD                 RB,       0030      ; CONVERT TO ASCII
449.                     STORE     RB,       0035      ;36
450.
451.                     LOAD      RD,       000A      ; 10
452.                     CALL      FIND                ; FIND HOW MANY RETURNS
453.                     ADD                 RB,       0030      ; CONVERT TO ASCII
454.                     STORE     RB,       0036      ;37
455.
456.                     ADD                 RE,       0030      ; 1
457.                     STORE     RE,       0037      ;38
458.                     RETURN
459.                              ;   FIND
460.  FIND
461.                     LOAD      RB,       ZERO
462.  AGN
463.                     SUB       RE,       RD
464.                     COMP      RE,                 8000      ; CHECK MSB FOR NEGATIVE NUMBER
465.                     JUMPC     POSITIVE                      ; IF NOT NEGATIVE JUMP
466.                     ADD                 RE,       RD        ; IF NEGATIVE R3<- R3 + RD
467.                     RETURN
468.
469.  POSITIVE
470.                     ADD       RB, 0001                      ;INCREMENT RB
471.                     JUMP      AGN                           ;REPET
472.  ;************************************************************
473.
474.
475.
476.
477.
478.
479.  ;////////////////////////////////////////////////////
480.  ;                                                    ISR
481.  ;////////////////////////////////////////////////////
482.                              ADDRESS 0300
483.  ISR       ; INTERUPT HAS BEEN SET CHECK WHERE IT CAME FROM
484.
485.                     ; IF DATA FROM INPORT1 = 1  then  STATUS        0001
486.                     ; IF DATA FROM INPORT0 = 1  then DATA           0000
487.                     ; IF DATA FROM INPORT2 = 1  then SWITHCES       0002
488.
489.                           ;///////////////CHEKING FOR SWITCHES ////////////
490.           INPUT   SWITCH_REG ,          0002 ;INPUT DATA FROM SWITCHES
491.           OUTPUT  SWITCH_REG ,          0002 ; OUTPUT DATA FROM SWITCHES
492.
```

```
493.
494.                     ;STATUS FOR OVF, FERR, PERR, TXRD, RXRDY
495.                     INPUT     STATUS_REG,        0001
496.
497.                     LOAD      TEMP_REG, STATUS_REG
498.                     AND       TEMP_REG, 0003      ; CHECK FOR TX, RX
499.
500.
501.
502.                     COMP      TEMP_REG, 0001     ;RX
503.                     CALLZ     CALL_RX
504.
505.                     COMP      TEMP_REG, 0002     ;TX
506.                     CALLZ     CALL_TX
507.
508.                     COMP      TEMP_REG, 0003     ;BOTH
509.                     JUMPNZ    SKIP
510.                     CALL      CALL_RX
511.                     CALL      CALL_TX
512.
513.  SKIP
514.
515.                     RETEN
516.
517.                     ; IF DATA FROM INPORT0 = 1  then DATA                              0000
518.  ;////////////////////////////////  RX  ////////////////////
519.
520.  CALL_RX ;
521.
522.                     INPUT     DATA_REG,          0000 ; DATA COMES FORM PORT0
523.                     COMP      DATA_REG,          ZERO
524.                     RETURNZ  ; RETURN IF DATA IS EMPTY
525.
526.                     LOAD      ERROR_REG, STATUS_REG
527.                     AND       ERROR_REG, 001C    ;OVF, FERR, PERR, 0 , 0
528.                     COMP      ERROR_REG, 0000    ;CHEKC IF ANY FLAGS ARE RAISED
529.                     CALLNZ    OUT_ERROR          ; IF ITS NOT ZERO THEN ERROR
530.
531.                     ; CHECK DATA THAT IS COMING INPUT
532.
533.                     COMP      DATA_REG,  ASTERISK
534.                     CALLZ     DIS_HOMETOWN
535.
536.                     COMP      DATA_REG, AT
537.                     CALLZ     DIS_CHAR_COUNT
538.
539.                     COMP      DATA_REG, CR
540.                     CALLZ     DIS_CR
541.
542.                     COMP      DATA_REG, BACKSPACE
543.                     CALLZ     DIS_BACKSPACE
544.
545.                     COMP      DATA_REG, PERCENT
546.                     CALLZ     DIS_MEM
547.
548.
549.                     COMP      DATA_REG, BACKSPACE
550.                     CALLNZ    OUTPUT_CHARACTER            ;SO IT WONT DISPLAY 0x7F
551.
552.
553.
554.                     COMP      COUNT,    0028      ; COMPARE TO 40th CHARACTER
555.                     CALLZ     DIS_CR ;
556.
557.                     RETURN
558.
559.  OUTPUT_CHARACTER
560.                     OUTPUT    DATA_REG, ZERO               ; OUTPUT CHARACTER
561.                     OUTPUT    DATA_REG, (SP)               ; STORE CHARACTER IN SRAM
562.          LOAD       DATA_REG,           ZERO      ; RESET DATA REG ATER WE OUTPUT
563.                     ADD       COUNT,    ONE
```

```
564.                    ADD       SP,       ONE
565.                    RETURN
566.
567.  DIS_PROMPT    ; 2B- 33
568.                    LOAD      OUT_SET,  0005 ; OUTPUT PROMPT 5
569.                    LOAD      PTR,                002B
570.                    LOAD      TEMP_REG,           ZERO
571.                    OUTPUT    TEMP_REG,           ZERO ;SET TX INTERUPT
572.                    RETURN
573.
574.  DIS_HOMETOWN                          ;0-9
575.                    LOAD      OUT_SET,  0001 ; OUTPUT HOMETOWN 1
576.                    LOAD      PTR,                ZERO
577.                    LOAD      TEMP_REG, ZERO
578.                    OUTPUT    TEMP_REG,           ZERO ;SET TX INTERUPT
579.                    RETURN
580.
581.  DIS_CHAR_COUNT              ; 34 - 38
582.                    CALL      BIN2ASCII
583.                    LOAD      OUT_SET,  0002 ; OUTPUT NUBMER OF CHARACTERS ENTERED 2
584.                    LOAD      PTR,                0034 ; 0034 - 38
585.                    LOAD      TEMP_REG,           ZERO
586.                    OUTPUT    TEMP_REG,           ZERO ;SET TX INTERUPT
587.                    RETURN
588.
589.  DIS_CR            ;29 - 2A
590.                    LOAD      OUT_SET,  0003 ; OUTPUT CR 3
591.                    LOAD      PTR,                0029  ;
592.                    LOAD      TEMP_REG,           ZERO
593.                    OUTPUT    TEMP_REG,           ZERO ;SET TX INTERUPT
594.                    RETURN
595.
596.  DIS_BACKSPACE ;26 - 28
597.                    COMP      COUNT,              0000
598.                    RETURNZ                                ;IF COUNT = 1 GOES BACK
599.                    LOAD      OUT_SET,  0004 ; OUTPUT BACKSPACE 4
600.                    LOAD      PTR,                0026  ;
601.                    SUB       COUNT,              0001 ; "1 "FOR THE OFFSET OF ADDIDNG THE CHAR
602.                    SUB       SP,                        ONE       ;
603.                    LOAD      TEMP_REG,           ZERO
604.                    OUTPUT    TEMP_REG,           ZERO ;SET TX INTERUPT
605.                    RETURN
606.
607.  DIS_BANNER
608.                    LOAD      OUT_SET,  0006 ; BANNER
609.                    LOAD      PTR,                000A
610.                    LOAD      TEMP_REG,           ZERO
611.                    OUTPUT    TEMP_REG,           ZERO ;SET TX INTERUPT
612.                    RETURN
613.
614.  OUT_ERROR
615.                    OUTPUT    ERROR_REG,          0005 ;
616.                    LOAD      STATUS_REG, ZERO
617.                    LOAD      ERROR_REG,          ZERO
618.                    RETURN  ;RETURN OUT OF ISR
619.
620.  DIS_MEM
621.                    LOAD      SET_DUMP,           0007
622.                    LOAD      PTR,                8000
623.                    LOAD      TEMP_REG,           ZERO
624.                    OUTPUT    TEMP_REG,           ZERO
625.                    RETURN
626.
627.
628.  ;////////////////////////////  TX  ///////////////////////
629.  CALL_TX
630.
631.
632.
633.                    COMP      OUT_SET,  0001 ; OUTPUTING HOMETOWN
634.                    CALLZ     OUT_HOMETOWN ;
```

```
635.
636.                     COMP      OUT_SET,  0002 ; OUTPUTING DIS_CHAR_COUNT
637.                     CALLZ     OUT_CHARCOUNT
638.
639.                     COMP      OUT_SET,  0003 ;
640.                     CALLZ     OUT_CR
641.
642.                     COMP      OUT_SET,  0004 ;
643.                     CALLZ     OUT_BACKSPACE
644.
645.                     COMP      OUT_SET,  0005 ;
646.                     CALLZ     OUT_PROMPT
647.
648.                     COMP      OUT_SET,  0006 ;
649.                     CALLZ     OUT_BANNER
650.
651.                     COMP      OUT_SET,  0007 ; FOR OUTPUTING MEM_TEST_DONE0
652.                     CALLZ     MEM_TEST_DONE
653.
654.                     COMP      SET_DUMP, 0007 ; FOR CHARACTERS
655.                     CALLZ     DUMP_MEMORY
656.
657.                     COMP      SET_DUMP, 0008
658.                     CALLZ     OUTPUT_FIVES
659.
660.                     COMP      SET_DUMP,           0009
661.                     CALLZ     OUTPUT_As
662.
663.                     COMP      SET_DUMP,           000A
664.                     CALLZ     OUTPUT_ADD
665.
666.                     RETURN
667.
668.
669.  OUT_HOMETOWN       ;0-8
670.                     FETCH     TEMP_REG,           (PTR)
671.                     OUTPUT    TEMP_REG,           ZERO
672.                     ADD       PTR,                ONE
673.                     COMP      PTR,                0009
674.                     CALLZ     DIS_CR
675.                     RETURN
676.
677.  OUT_CHARCOUNT      ; 34 - 38
678.                     FETCH     TEMP_REG,           (PTR)
679.                     OUTPUT    TEMP_REG,           ZERO
680.                     ADD       PTR,                ONE
681.                     COMP      PTR,                0039
682.                     CALLZ     RESET_CHARCOUNTER
683.                     RETURN
684.
685.  OUT_BACKSPACE      ;26 - 28
686.                     FETCH     TEMP_REG,           (PTR)
687.                     OUTPUT    TEMP_REG,           ZERO
688.                     ADD       PTR,                ONE
689.                     COMP      PTR,                0029
690.                     CALLZ     RESET_TX
691.                     RETURN
692.
693.
694.  OUT_CR                     ; 29 - 2A
695.                     FETCH     TEMP_REG,           (PTR)
696.                     OUTPUT    TEMP_REG,           ZERO
697.                     ADD       PTR,                ONE
698.                     COMP      PTR,                002B
699.                     CALLZ     DIS_PROMPT
700.                     RETURN
701.
702.
703.  OUT_BANNER     ;   0A - 24
704.                     FETCH     TEMP_REG,           (PTR)
705.                     OUTPUT    TEMP_REG,           ZERO
```

```
706.                    ADD       PTR,                ONE
707.                    COMP      PTR,                0025
708.                    CALLZ     DIS_CR
709.                    RETURN
710.
711.  OUT_PROMPT                   ; 2B - 32
712.                    FETCH     TEMP_REG,           (PTR)
713.                    OUTPUT    TEMP_REG,           ZERO
714.                    ADD       PTR,                ONE
715.                    COMP      PTR,                0033
716.                    CALLZ     RESET_TX
717.                    RETURN
718.
719.
720.  MEM_TEST_DONE
721.                    FETCH     TEMP_REG,           (PTR)
722.                    OUTPUT    TEMP_REG,           ZERO
723.                    ADD       PTR,                ONE
724.                    COMP      PTR,                004B
725.                    CALLZ     RESET_MEM_DUMP
726.                    RETURN
727.
728.
729.  OUTPUT_FIVES
730.                    INPUT     TEMP_REG, (PTR)
731.                    OUTPUT    TEMP_REG, ZERO  ;DISPLAY THE CONTENT OF THE MEM
732.                    ADD       PTR,                ONE
733.                    COMP      PTR,                FFFE
734.                    CALLNC    RESET_MEM_DUMP
735.                    RETURN
736.
737.  OUTPUT_As
738.                    INPUT     TEMP_REG, (PTR)
739.                    OUTPUT    TEMP_REG, ZERO  ;DISPLAY THE CONTENT OF THE MEM
740.                    ADD       PTR,                ONE
741.                    COMP      PTR,                FFFE
742.                    CALLNC    RESET_MEM_DUMP
743.                    RETURN
744.
745.  OUTPUT_ADD
746.                    INPUT     TEMP_REG, (PTR)
747.                    OUTPUT    TEMP_REG, ZERO  ;DISPLAY THE CONTENT OF THE MEM
748.                    ADD       PTR,                ONE
749.                    COMP      PTR,                FFFE
750.                    CALLNC    RESET_MEM_DUMP
751.                    RETURN
752.
753.
754.  DUMP_MEMORY
755.                    INPUT     TEMP_REG,           (PTR) ; FETCH DATA FORM SRAM
756.                    OUTPUT    TEMP_REG, ZERO ; OUTPUT DATA FORM SRAM
757.                    ADD       PTR,                ONE
758.                    COMP      PTR,                SP
759.                    CALLNC    RESET_MEM_DUMP
760.                    RETURN
761.
762.  RESET_MEM_DUMP
763.                    LOAD      SET_DUMP,           ZERO
764.                    LOAD      TEMP_REG,           FFFF
765.                    CALL      RESET_TX
766.                    RETURN
767.
768.
769.
770.
771.  RESET_TX
772.                    LOAD      PTR,                ZERO
773.                    LOAD      OUT_SET,  ZERO ; SET TO RX
774.                    RETURN
775.
776.
```

```
777.  RESET_CHARCOUNTER
778.                      LOAD      COUNT,              ZERO ;
779.                      CALL      DIS_CR
780.                      RETEN
781.
782.
783.
784.
785.
786.
787.                      ADDRESS 0FFE
788.  ENDIT
789.                      JUMP ISR
790.                      END
```

Technologic Specific Instantiation

TSI_MODULE.v Thu May 02 17:49:43 2019

```
1   `timescale 1ns / 1ps
2   //////////////////////////////////////////////////////////////////
3   //  File name: <TSI_MODULE>                                     //
4   //                                                                //
5   //  Created by       <Kevin Lopez> on <March 12, 2019>.            //
6   //  Copyright © 2018 <Kevin Lopez>. All rights reserved.          //
7   //                                                                //
8   //                                                                //
9   //  In submitting this file for class work at CSULB               //
10  //  I am confirming that this is my work and the work             //
11  //  of no one else. In submitting this code I acknowledge that    //
12  //  plagiarism in student project work is subject to dismissal.   //
13  //  from the class                                                //
14   //////////////////////////////////////////////////////////////////
15
16
17  module TSI_MODULE(  clk, btn_rst, sw, led, TX, RX, an, z ,
18                  clk_in, btn_rst_in, sw_in, led_out, TX_out, RX_in, an_out, z_out );
19  //https://www.xilinx.com/support/documentation/sw_manuals/xilinx2012_2/ug953-vivad
    series-libraries.pdf
20
21     ///////////    INPUTS   //////////
22     input          clk_in, btn_rst_in;
23     input [6:0]    sw_in;
24     input          RX_in ;
25
26     input [15:0]   led;
27     input          TX;
28     input [7:0]    an; // 8 anodes
29     input [6:0]    z;  // 7 pixels
30
31
32     ///////////    OUTPUTS  //////////
33     output      clk, btn_rst;
34     output [6:0] sw;
35     output      RX;
36
37     output [15:0]  led_out;
38     output         TX_out;
39     output [7:0]   an_out; // 8 anodes
40     output [6:0]   z_out;  // 7 pixels
41
42
43
44  ////////////////////////////////////////////////////////////////////////////////
45  //                            INPUTS
46  ////////////////////////////////////////////////////////////////////////////////
47  // BUFG: Global Clock Simple Buffer
48  // 7 Series
49  // Xilinx HDL Libraries Guide, version 2012.2
50  BUFG clk_buf (
51  .O(clk), // 1-bit output: Clock output
52  .I(clk_in) // 1-bit input: Clock input
53  );
54  // End of BUFG_inst instantiation
55
56
```

Page 1

TSI_MODULE.v Thu May 02 17:50:20 2019

```
  57   // IBUF: Single-ended Input Buffer
  58   // 7 Series
  59   // Xilinx HDL Libraries Guide, version 2012.2
  60   IBUF #(
  61   .IBUF_LOW_PWR("TRUE"), // Low power (TRUE) vs. performance (FALSE) setting for
  62                          // referenced I/O standards
  63   .IOSTANDARD("DEFAULT") // Specify the input I/O standard
  64   ) rst_buf (
  65   .O(btn_rst), // Buffer output
  66   .I(btn_rst_in) // Buffer input (connect directly to top-level port)
  67   ); // End of IBUF_inst instantiation
  68
  69
  70   // IBUF: Single-ended Input Buffer
  71   // 7 Series
  72   // Xilinx HDL Libraries Guide, version 2012.2
  73   IBUF #(
  74   .IBUF_LOW_PWR("TRUE"), // Low power (TRUE) vs. performance (FALSE) setting for
  75                          //referenced I/O standards
  76   .IOSTANDARD("DEFAULT") // Specify the input I/O standard
  77   ) sws_buf[6:0] (
  78   .O(sw[6:0]), // Buffer output
  79   .I(sw_in[6:0]) // Buffer input (connect directly to top-level port)
  80   ); // End of IBUF_inst instantiation
  81
  82   // IBUF: Single-ended Input Buffer
  83   // 7 Series
  84   // Xilinx HDL Libraries Guide, version 2012.2
  85   IBUF #(
  86   .IBUF_LOW_PWR("TRUE"), // Low power (TRUE) vs. performance (FALSE) setting for
  87                          //referenced I/O standards
  88   .IOSTANDARD("DEFAULT") // Specify the input I/O standard
  89   ) RX_buf (
  90   .O(RX), // Buffer output
  91   .I(RX_in) // Buffer input (connect directly to top-level port)
  92   ); // End of IBUF_inst instantiation
  93
  94
  95
  96
  97
  98   //////////////////////////////////////////////////////////////////////////////////
  99   //                              OUTPUTS
 100   //////////////////////////////////////////////////////////////////////////////////
 101
 102   //OBUF: Single-ended Output Buffer
 103   // 7 Series
 104   // Xilinx HDL Libraries Guide, version 2012.2
 105   OBUF #(
 106   .DRIVE(12), // Specify the output drive strength
 107   .IOSTANDARD("DEFAULT"), // Specify the output I/O standard
 108   .SLEW("SLOW") // Specify the output slew rate
 109   ) leds_buf[15:0] (
 110   .O(led_out[15:0]), // Buffer output (connect directly to top-level port)
 111   .I(led[15:0]) // Buffer input
 112   ); // End of OBUF_inst instantiation
 113
```

TSI_MODULE.v Thu May 02 17:50:48 2019

```
114
115   //OBUF: Single-ended Output Buffer
116   // 7 Series
117   // Xilinx HDL Libraries Guide, version 2012.2
118   OBUF #(
119   .DRIVE(12), // Specify the output drive strength
120   .IOSTANDARD("DEFAULT"), // Specify the output I/O standard
121   .SLEW("SLOW") // Specify the output slew rate
122   ) TX_buf (
123   .O(TX_out), // Buffer output (connect directly to top-level port)
124   .I(TX) // Buffer input
125   ); // End of OBUF_inst instantiation
126
127   //OBUF: Single-ended Output Buffer
128   // 7 Series
129   // Xilinx HDL Libraries Guide, version 2012.2
130   OBUF #(
131   .DRIVE(12), // Specify the output drive strength
132   .IOSTANDARD("DEFAULT"), // Specify the output I/O standard
133   .SLEW("SLOW") // Specify the output slew rate
134   ) anothes_buf[7:0] (
135   .O(an_out[7:0]), // Buffer output (connect directly to top-level port)
136   .I(an[7:0]) // Buffer input
137   ); // End of OBUF_inst instantiation
138
139
140
141   //OBUF: Single-ended Output Buffer
142   // 7 Series
143   // Xilinx HDL Libraries Guide, version 2012.2
144   OBUF #(
145   .DRIVE(12), // Specify the output drive strength
146   .IOSTANDARD("DEFAULT"), // Specify the output I/O standard
147   .SLEW("SLOW") // Specify the output slew rate
148   ) Z_buf[6:0] (
149   .O(z_out[6:0]), // Buffer output (connect directly to top-level port)
150   .I(z[6:0]) // Buffer input
151   ); // End of OBUF_inst instantiation
152
153
154
155
156   endmodule
157
```

Main Module

MAIN_MODULE.v Thu May 02 17:53:21 2019

```
1   `timescale 1ns / 1ps
2   //////////////////////////////////////////////////////////////////
3   //  File name: <MAIN_MODULE>                                    //
4   //                                                              //
5   //  Created by       <Kevin Lopez> on <May 2, 2019>.            //
6   //  Copyright Ã‚Â© 2018 <Kevin Lopez>. All rights reserved.      //
7   //                                                              //
8   //                                                              //
9   //  In submitting this file for class work at CSULB             //
10  //  I am confirming that this is my work and the work           //
11  //  of no one else. In submitting this code I acknowledge that  //
12  //  plagiarism in student project work is subject to dismissal. //
13  //  from the class                                              //
14   //////////////////////////////////////////////////////////////////
15  module MAIN_MODULE(clk_in, btn_rst_in, sw_in, RX_in,
16     led_out, TX_out, an_out, z_out );
17
18      ////////////      INPUTS   //////////
19     input       clk_in, btn_rst_in;
20     input [6:0] sw_in;
21     input       RX_in;
22
23     ////////////    OUTPUTS  //////////
24     output [15:0]  led_out;
25     output         TX_out;
26     output [7:0]   an_out; // 8 anodes
27     output [6:0]   z_out;  // 7 pixels
28
29     ////////////    WIRES    //////////
30     wire [6:0] sw;
31     wire [15:0] led;
32     wire [7:0] an;
33     wire [6:0] z;
34
35
36     /*************************************************
37     DESING IS INSIDE THE CORE AND THE TSI BUFFER PROVIDES THE
38     ELECTRICAL CHARACTERISTICS TO WORK ON NEXYS4 ARTIX 7 FPGA
39     *************************************************/
40     CORE        CORE(.clk(clk), .btn_rst(btn_rst), .sw(sw),
41                    .led(led), .TX(TX), .RX(RX), .an(an), .z(z) );
42
43     TSI_MODULE  TSI( .clk(clk), .btn_rst(btn_rst), .sw(sw), .led(led),
44                   .TX(TX), .RX(RX), .an(an), .z(z) , .clk_in(clk_in),
45                   .btn_rst_in(btn_rst_in), .sw_in(sw_in), .led_out(led_out),
46                  .TX_out(TX_out), .RX_in(RX_in), .an_out(an_out),
47                  .z_out(z_out));
48
49
50  endmodule
51
```

Constrains File

Constrains.ucf **Sun May 12 14:09:19 2019**

```
## This file is a general .ucf for the Nexys4 DDR Rev C board
## To use it in a project:
## - uncomment the lines corresponding to used pins
## - rename the used signals according to the project

## Clock signal
NET "clk_in"   LOC = "E3"  | IOSTANDARD = "LVCMOS33";
#Bank = 35, Pin name = #IO_L12P_T1_MRCC_35,              Sch name = clk100mhz
NET "clk_in" TNM_NET = sys_clk_pin;
TIMESPEC TS_sys_clk_pin = PERIOD sys_clk_pin 100 MHz HIGH 50%;


## Switches
#NET "sw<0>"          LOC=J15 | IOSTANDARD=LVCMOS33; #IO_L24N_T3_RS0_15
NET "sw_in<0>"          LOC=L16 | IOSTANDARD=LVCMOS33; #IO_L3N_T0_DQS_EMCCLK_14
NET "sw_in<1>"          LOC=M13 | IOSTANDARD=LVCMOS33; #IO_L6N_T0_D08_VREF_14
NET "sw_in<2>"          LOC=R15 | IOSTANDARD=LVCMOS33; #IO_L13N_T2_MRCC_14
NET "sw_in<3>"          LOC=R17 | IOSTANDARD=LVCMOS33; #IO_L12N_T1_MRCC_14
NET "sw_in<4>"          LOC=T18 | IOSTANDARD=LVCMOS33; #IO_L7N_T1_D10_14
NET "sw_in<5>"          LOC=U18 | IOSTANDARD=LVCMOS33; #IO_L17N_T2_A13_D29_14
NET "sw_in<6>"          LOC=R13 | IOSTANDARD=LVCMOS33; #IO_L5N_T0_D07_14
#NET "sw<8>"          LOC=T8 | IOSTANDARD=LVCMOS18; #IO_L24N_T3_34
#NET "sw<9>"          LOC=U8 | IOSTANDARD=LVCMOS18; #IO_25_34
#NET "sw<10>"         LOC=R16 | IOSTANDARD=LVCMOS33; #IO_L15P_T2_DQS_RDWR_B_14
#NET "sw<11>"         LOC=T13 | IOSTANDARD=LVCMOS33; #IO_L23P_T3_A03_D19_14
#NET "sw<12>"         LOC=H6 | IOSTANDARD=LVCMOS33; #IO_L24P_T3_35
#NET "sw<13>"         LOC=U12 | IOSTANDARD=LVCMOS33; #IO_L20P_T3_A08_D24_14
#NET "sw<14>"         LOC=U11 | IOSTANDARD=LVCMOS33; #IO_L19N_T3_A09_D25_VREF_14
#NET "sw<15>"         LOC=V10 | IOSTANDARD=LVCMOS33; #IO_L21P_T3_DQS_14


## Buttons
#NET "cpu_resetn"     LOC=C12 | IOSTANDARD=LVCMOS33; #IO_L3P_T0_DQS_AD1P_15

#NET "btn"           LOC=N17 | IOSTANDARD=LVCMOS33; #IO_L9P_T1_DQS_14
NET "btn_rst_in"           LOC=P18 | IOSTANDARD=LVCMOS33; #IO_L9N_T1_DQS_D13_14
#NET "btnl"           LOC=P17 | IOSTANDARD=LVCMOS33; #IO_L12P_T1_MRCC_14
#NET "btnr"           LOC=M17 | IOSTANDARD=LVCMOS33; #IO_L10N_T1_D15_14
#NET "btnu"           LOC=M18 | IOSTANDARD=LVCMOS33; #IO_L4N_T0_D05_14


## LEDs
NET "led_out<0>"         LOC=H17 | IOSTANDARD=LVCMOS33; #IO_L18P_T2_A24_15
NET "led_out<1>"         LOC=K15 | IOSTANDARD=LVCMOS33; #IO_L24P_T3_RS1_15
NET "led_out<2>"         LOC=J13 | IOSTANDARD=LVCMOS33; #IO_L17N_T2_A25_15
NET "led_out<3>"         LOC=N14 | IOSTANDARD=LVCMOS33; #IO_L8P_T1_D11_14
NET "led_out<4>"         LOC=R18 | IOSTANDARD=LVCMOS33; #IO_L7P_T1_D09_14
NET "led_out<5>"         LOC=V17 | IOSTANDARD=LVCMOS33; #IO_L18N_T2_A11_D27_14
NET "led_out<6>"         LOC=U17 | IOSTANDARD=LVCMOS33; #IO_L17P_T2_A14_D30_14
NET "led_out<7>"         LOC=U16 | IOSTANDARD=LVCMOS33; #IO_L18P_T2_A12_D28_14
NET "led_out<8>"         LOC=V16 | IOSTANDARD=LVCMOS33; #IO_L16N_T2_A15_D31_14
NET "led_out<9>"         LOC=T15 | IOSTANDARD=LVCMOS33; #IO_L14N_T2_SRCC_14
NET "led_out<10>"        LOC=U14 | IOSTANDARD=LVCMOS33; #IO_L22P_T3_A05_D21_14
NET "led_out<11>"        LOC=T16 | IOSTANDARD=LVCMOS33; #IO_L15N_T2_DQS_DOUT_CSO_B_14
NET "led_out<12>"        LOC=V15 | IOSTANDARD=LVCMOS33; #IO_L16P_T2_CSI_B_14
NET "led_out<13>"        LOC=V14 | IOSTANDARD=LVCMOS33; #IO_L22N_T3_A04_D20_14
NET "led_out<14>"        LOC=V12 | IOSTANDARD=LVCMOS33; #IO_L20N_T3_A07_D23_14
```

Page 1

Constrains.ucf Sun May 12 14:10:09 2019

```
58   NET "led_out<15>"        LOC=V11 | IOSTANDARD=LVCMOS33; #IO_L21N_T3_DQS_A06_D22_14
59
60
61   ##LEDs_RGB
62   #NET "led16_b"         LOC=R12 | IOSTANDARD=LVCMOS33; #IO_L5P_T0_D06_14
63   #NET "led16_g"         LOC=M16 | IOSTANDARD=LVCMOS33; #IO_L10P_T1_D14_14
64   #NET "led16_r"         LOC=N15 | IOSTANDARD=LVCMOS33; #IO_L11P_T1_SRCC_14
65   #NET "led17_b"         LOC=G14 | IOSTANDARD=LVCMOS33; #IO_L15N_T2_DQS_ADV_B_15
66   #NET "led17_g"         LOC=R11 | IOSTANDARD=LVCMOS33; #IO_0_14
67   #NET "led17_r"         LOC=N16 | IOSTANDARD=LVCMOS33; #IO_L11N_T1_SRCC_14
68
69
70   ## 7 segment display
71   NET "z_out<0>"             LOC=T10 | IOSTANDARD=LVCMOS33; #IO_L24N_T3_A00_D16_14
72   NET "z_out<1>"             LOC=R10 | IOSTANDARD=LVCMOS33; #IO_25_14
73   NET "z_out<2>"             LOC=K16 | IOSTANDARD=LVCMOS33; #IO_25_15
74   NET "z_out<3>"             LOC=K13 | IOSTANDARD=LVCMOS33; #IO_L17P_T2_A26_15
75   NET "z_out<4>"             LOC=P15 | IOSTANDARD=LVCMOS33; #IO_L13P_T2_MRCC_14
76   NET "z_out<5>"             LOC=T11 | IOSTANDARD=LVCMOS33; #IO_L19P_T3_A10_D26_14
77   NET "z_out<6>"             LOC=L18 | IOSTANDARD=LVCMOS33; #IO_L4P_T0_D04_14
78   #NET "z<7>"              LOC=H15 | IOSTANDARD=LVCMOS33; #IO_L19N_T3_A21_VREF_15
79
80   NET "an_out<0>"           LOC=J17 | IOSTANDARD=LVCMOS33; #IO_L23P_T3_FOE_B_15
81   NET "an_out<1>"           LOC=J18 | IOSTANDARD=LVCMOS33; #IO_L23N_T3_FWE_B_15
82   NET "an_out<2>"           LOC=T9 | IOSTANDARD=LVCMOS33; #IO_L24P_T3_A01_D17_14
83   NET "an_out<3>"           LOC=J14 | IOSTANDARD=LVCMOS33; #IO_L19P_T3_A22_15
84   NET "an_out<4>"           LOC=P14 | IOSTANDARD=LVCMOS33; #IO_L8N_T1_D12_14
85   NET "an_out<5>"           LOC=T14 | IOSTANDARD=LVCMOS33; #IO_L14P_T2_SRCC_14
86   NET "an_out<6>"           LOC=K2 | IOSTANDARD=LVCMOS33; #IO_L23P_T3_35
87   NET "an_out<7>"           LOC=U13 | IOSTANDARD=LVCMOS33; #IO_L23N_T3_A02_D18_14
88
89
90
91   ##Temperature Sensor
92   #NET "tmp_ct"          LOC=B14 | IOSTANDARD=LVCMOS33; #IO_L2N_T0_AD8N_15
93   #NET "tmp_int"         LOC=D13 | IOSTANDARD=LVCMOS33; #IO_L6N_T0_VREF_15
94   #NET "tmp_scl"         LOC=C14 | IOSTANDARD=LVCMOS33; #IO_L1N_T0_AD0N_15
95   #NET "tmp_sda"         LOC=C15 | IOSTANDARD=LVCMOS33; #IO_L12N_T1_MRCC_15
96
97
98   ##USB-RS232 Interface
99   #NET "uart_cts"        LOC=D3 | IOSTANDARD=LVCMOS33; #IO_L12N_T1_MRCC_35
100  #NET "uart_rts"        LOC=E5 | IOSTANDARD=LVCMOS33; #IO_L5N_T0_AD13N_35
101  NET "TX_out"    LOC=D4 | IOSTANDARD=LVCMOS33; #IO_L11N_T1_SRCC_35
102  NET "RX_in"     LOC=C4 | IOSTANDARD=LVCMOS33; #IO_L7P_T1_AD6P_35
103
104
105
```