COLLEGE OF ENGINEERING

Department of Computer Engineering and Computer Science

Final Project

System on Chip Specification

CECS 460: System on Chip Design

Submitted by:

Yamin Shwe Yee

(016993520)

Submitted to:

Mr. John Tramel

CECS Faculty

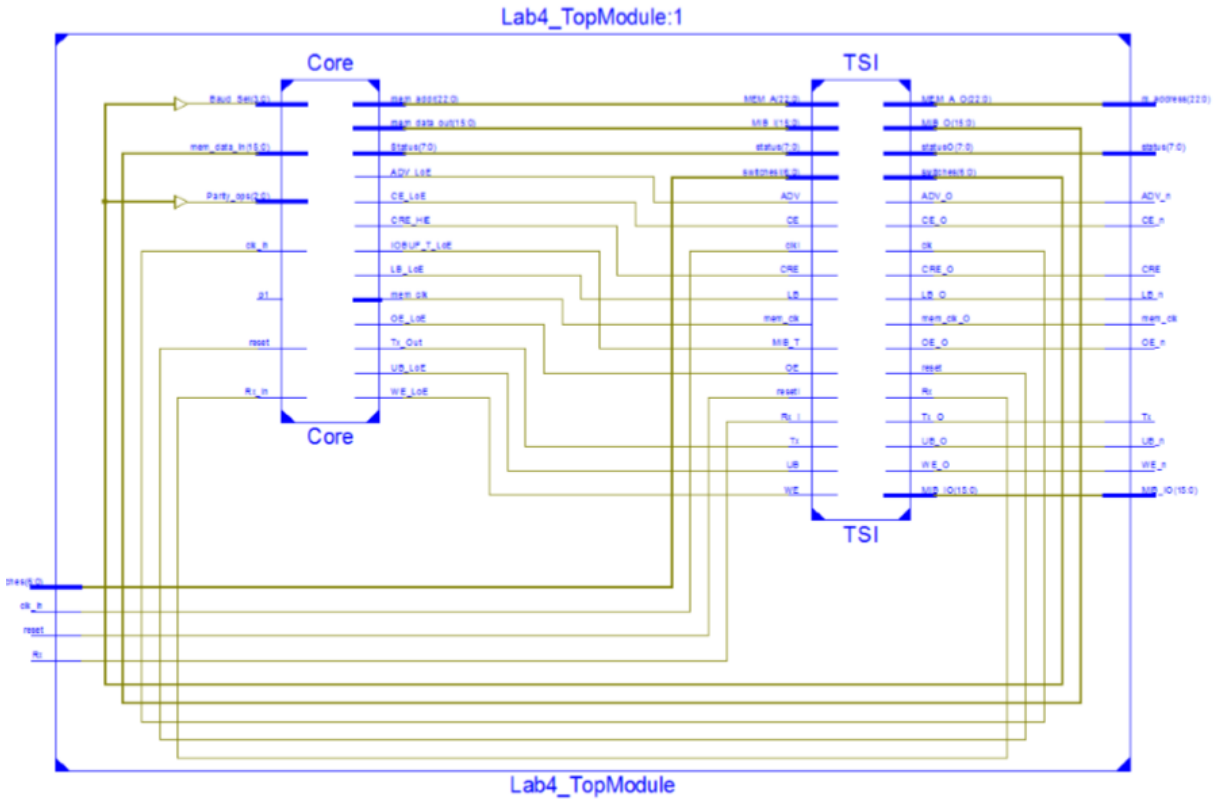
May 14, 2019

Prepared by: Yamin Yee	Date: May 9, 2019	Document Name: Chip Specification

Top Level Design

4.1 Description

The Top Level is the connection between two things, the Core and the TSI. The Core consists of the design itself, while TSI contains all references to the target technology libraries. Any communication from the core's I/O's must pass through the TSI before interacting with the FPGA.



4.4 Data Flow Description

The functionalities of both of Transmit Engine and Receive Engine of the UART. The Transmit Engine was verified when the statement is consistently printed on a Serial Terminal. Next the Receive Engine was verified when it was able to take in a user inputted character and transmit directly to the terminal. The design was also able to detect parity errors, framing errors, and overflow errors in case the transmission of data from one machine to the next was not correct. Additionally, the TramelBlaze was able to service the correct engine based on the Port ID's.

MIB was created to give the TramelBlaze access to the on-board Micron Memory. Alongside with the Full UART, the Tramel Blaze should be able to take in data from Serial Terminal and write those characters into a location in memory. Once an asterisk is received by the Receive Engine, the data in the memory will then be read by the Tramel Blaze and output to the Serial Terminal.

4.5 I/O

4.5.1 Signal Names

Signal	From	TO	Description
c_clk	TSI	Core	100MHz Crystal Oscillator
c_rst	TSI	Core	System Reset
c_switches	TSI	Core	Baud Rate and Parity Options
c_TX	Core	TSI	Transmit Line out to USB
c_Rx	TSI	Core	Receive Line in from USB
c_mem_addr	Core	TSI	Point to a location in Micron Memory
c_status	Core	TSI	Detects data errors, and Engine readiness
c_CE	Core	TSI	Chip Enable for Memory
c_WE	Core	TSI	Write Enable for Memory
c_OE	Core	TSI	Output Enable for Memory
c_ADV	Core	TSI	Address Vaild
c_CRE	Core	TSI	Control Register Enable
c_UB	Core	TSI	Upper 6 bits of Memory Address
c_LB	Core	TSI	Lower 16 bits of Memory
c_mem_clk	TSI	Core	Cell Ram Clock
MIB_I	Core	TSI	Data from MIB to Memory
MIB_O	TSI	Core	Data from Memory to MIB
T	Core	TSI	IOBUF Selector

Pin Assignment

Input Signals	Assignment	Output Signals	Assignment
clk_in	E3	Tx_out	D4
rst	E16	ADV_in	T13
Rx_In	C4	CE_n	L18
Eight	V6	LB_n	J15
PEN	V7	UB_n	J13
OHEL	R5	OE_n	H14
Baud_Sel[3]	R6	WE_n	R11
Baud_Sel[2]	R7	CRE	J14
Baud_Sel[1]	U8	mem_clk	T15
Baud_Sel[0]	U9	Status[7]	U6
		Status[6]	U7
		Status[5]	T4
		Status[4]	T5
		Status[3]	T6
		Status[2]	R8
		Status[1]	V9
		Status[0]	T8

Address Bus	Address Bus	Address Bus	Data Bus	Data Bus
M_address[22]: U13	m_address[13]:U16	m_address[4]:H16	MIB_IO[1]:P17	MIB_IO[6]: T18
M_address[21]: M16	m_address[12]:P14	m_address[3]:J17	MIB_IO[1]:N17	MIB_IO[5]: R17
M_address[20]: T10	m_address[13]:V12	m_address[2]:H15	MIB_IO[1]:P18	MIB_IO[4]: U18
M_address[19]: U13	m_address[13]:V14	m_address[1]:H17	MIB_IO[1]:M17	MIB_IO[3]:R13
M_address[18]: U13	m_address[13]:U16	m_address[0]:J18	MIB_IO[1]:M18	MIB_IO[2]: U12
M_address[17]: U13	m_address[13]:U16		MIB_IO[1]:G17	MIB_IO[1]: T11
M_address[16]: U13	m_address[13]:U16		MIB_IO[9]:G18	MIB_IO[0]: R12
M_address[15]: U13	m_address[13]:U16		MIB_IO[8]:F18	
M_address[14]: U13	m_address[13]:U16		MIB_IO[7]:R18	

Clocks

_____The Nexys 4 is powered by a 100MHz crystal oscillator which is connected to pin E3. Every single flop is the project will utilize this clock.

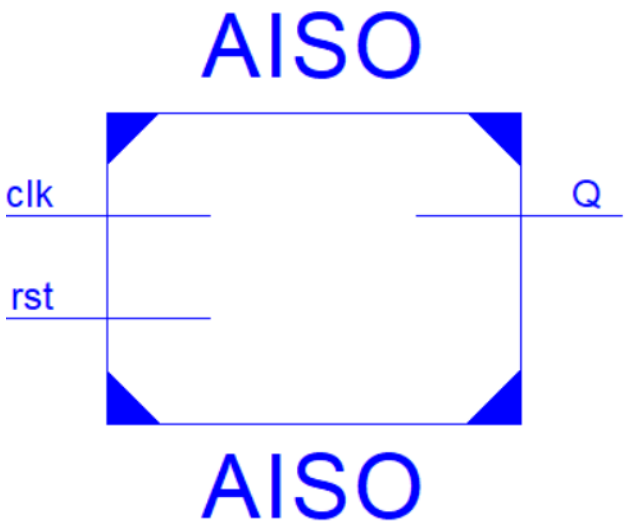
Resets

_____The project will utilize a HIGH active reset, this HIGH active reset will be go into an ASIO module which will make sure every flop in design will not go metastable state when the reset is released.

Externally Developed Blocks

Asynchronous In, Synchronous Out(AISO)

Description



AISO makes sure to prevent every flops in design go metastable stage when the reset is released. AISO will helps every flops go metastable states and the reset of the block is directly connected to a button(BTNC), which will eventually reset every module.

I/O

Signal	I/O	Connected To
clk	Input	100MHz Crystal Oscillator
rst	Input	BTNC
Q	Output	Resets of Every Block

Register Map

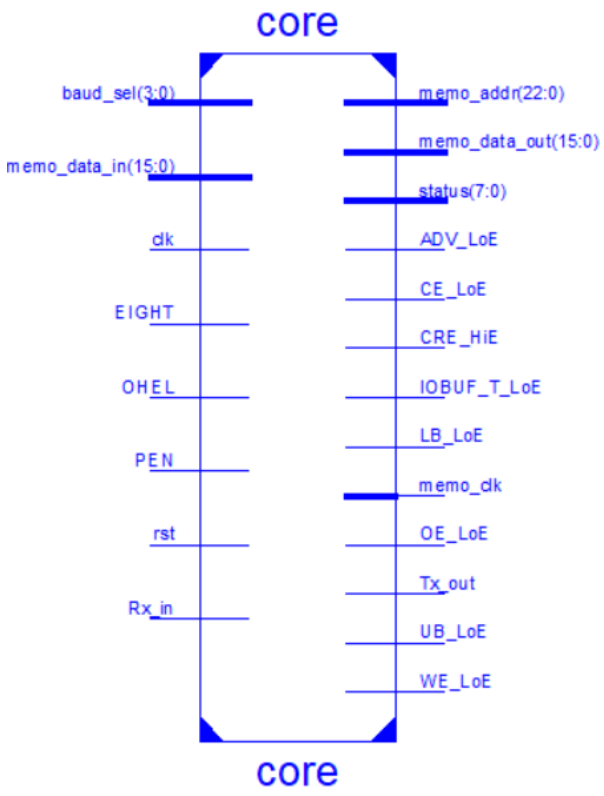
Register	Usage	Description
flop[1:0]	Synchronize	Upon reset, sends a signal pulse to all modules

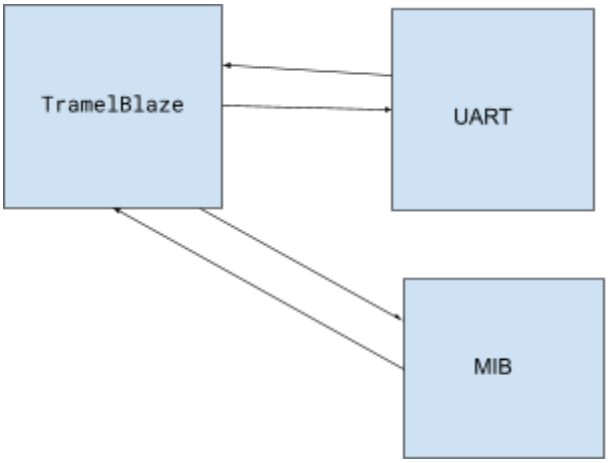
Internally Developed Blocks

Core

Description

The Core complies the processor and its interfaces in one large instantiation. Any signals that interacts with the Core must go through TSI first. In this design, the Core consists of the externally developed TramelBlaze, and internally developed Full UART and MIB. It also instantiates some accessory components as well, such as the AISO to provide a synchronous reset to all modules, and a Pulse Maker and SR flop to handle the TramelBlaze's interrupts.





I/O

All inputs and outputs state in this table are connected to the TSI before interacting with other components of the FPGA.

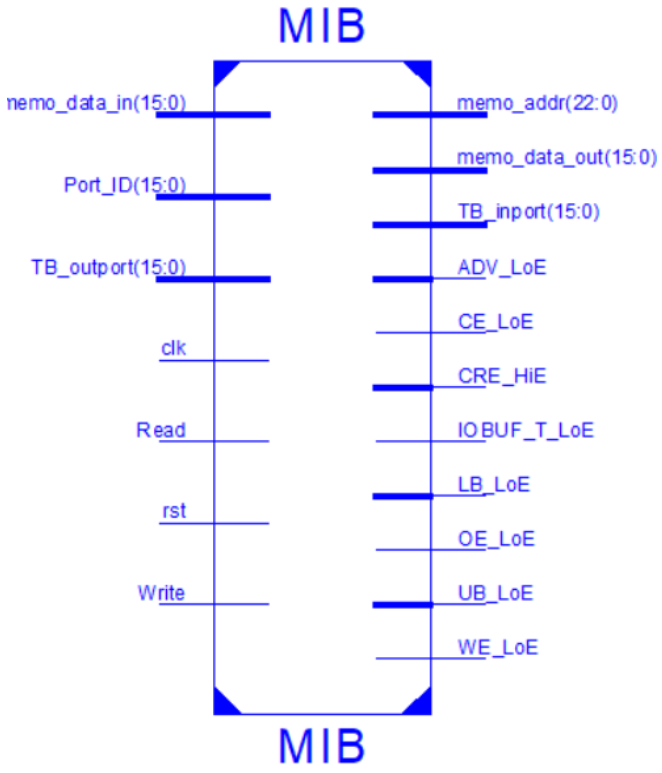
Signal	Size(bits)	I/O	Connected to
clk	1	I	100MHz Crystal Oscillator
rst	1	I	BTNC
Rx_in	1	I	USB
memo_data_in	16	I	Memory
baud_sel	4	I	On-board Switches
EIGHT	1	I	On-board Switches
PEN	1	I	On-board Switches
OHEL	1	I	On-board Switches
memo_addr	23	O	Memory
memo_data_out	16	O	Memory
IOBUF_T_LoE	1	O	TSI
CE_LoE	1	O	Memory
WE_LoE	1	O	Memory
OE_LoE	1	O	Memory

ADV_LoE	**1**	**O**	**Memory**
CRE_HiE	**1**	**O**	**Memory**
UB_LoE	**1**	**O**	**Memory**
LB_LoE	**1**	**O**	**Memory**
Tx_Out	**1**	**O**	**Memory**
status	**8**	**O**	**On-board LEDs**
mem_clk	**1**	**O**	**Memory**

Memory Interface Block(MIB)

Description

The MIB consists of two major blocks, the state machine and the control flops. The state machine sends the signals based on what receives according to Port ID. The state machine allows the signals to be one of the three configurations, IDLE, memory read, memory write, and these states control singlas. With the signals sent out, the control flops of the state machine manage the memory address and the data coming to and from the Memory. From this point, the Tramel Blaze's ROM controls the timing in when to send the memory address location and went to send or receive data.



I/O

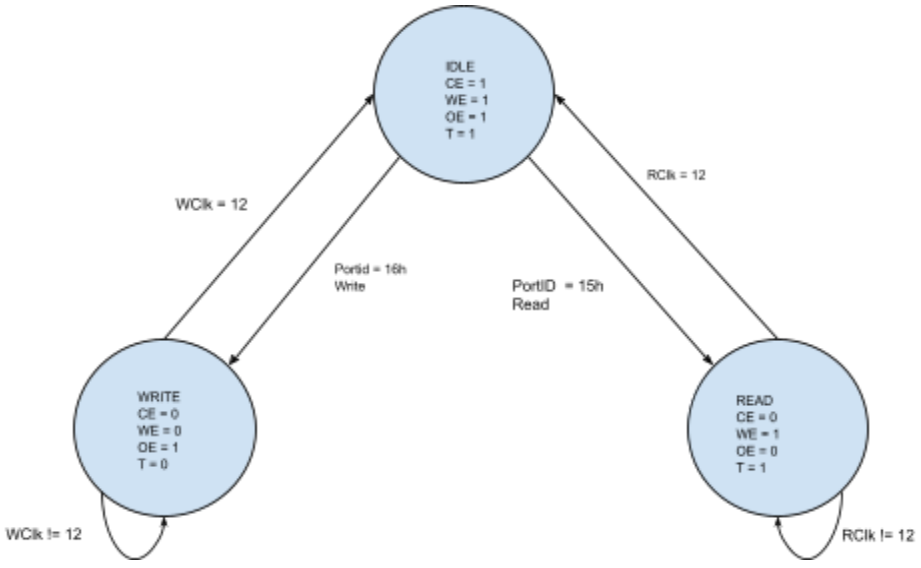
Signal	Size(bits)	I/O	Connected To
clk	**1**	**I**	**100Mhz Crystal Oscillator**
rst	**1**	**I**	**AISO**
Read	**1**	**I**	**TramelBlaze**
Write	**1**	**I**	**TramelBlaze**
Port_ID	**16**	**I**	**TramelBlaze**
memo_data_in	**16**	**I**	**Memory**
TB_outport	**16**	**I**	**TramelBlaze**
CE_LoE	**1**	**O**	**Memory**
WE_LoE	**1**	**O**	**Memory**
OE_LoE	**1**	**O**	**Memory**
IOBUF_T_LoE	**1**	**O**	**TSI**
ADV_LoE	**1**	**O**	**Memory**
CRE_HiE	**1**	**O**	**Memory**
UB_LoE	**1**	**O**	**Memory**
LB_LoE	**1**	**O**	**Memory**
memo_addr	**23**	**O**	**Memory**
memo_data_out	**16**	**O**	**Memory**
TB_inport	**16**	**O**	**TramelBlaze**

Register Map

Register	Module	Description
CE_LoE	**MIB**	**Chip Enable Signal for Memory**
WE_LoE	**MIB**	**Write Enable Signal**
OE_LoE	**MIB**	**Output Enable Signal**
IOBUF_T_LoE	**MIB**	**Controller for IOBUF in TSI**
ADV_LoE	**MIB**	**Address Valid Signal**
CRE_HiE	**MIB**	**Control Byte Signal**
UB_LoE	**MIB**	**Upper Byte Signal**
LB_LoE	**MIB**	**Lower Byte Signal**
memo_addr	**MIB**	**Memory Address Location**
memo_data_out	**MIB**	**Data Sent**
TB_inport	**MIB**	**Data Sent**
state	**MIB**	**State of MIB state machine**
n_state	**MIB**	**Next state of MIB**
WCLK	**MIB**	**Counter to stay in write**
RCLK	**MIB**	**Counter to stay in read**
nCE_LoE	**MIB**	**Next State of Chip Enable**
nWE_LoE	**MIB**	**Next State of Write Enable**

nOE_LoE	**MIB**	**Next State of Output Enable**
nIOBUF_T_LoE	**MIB**	**Next state of T**
nUB_LoE	**MIB**	**Next state of Upper Byte**
nLB_LoE	**MIB**	**Next state of Lower Byte**
WA_R0	**MIB**	**Register to store and send Upper Byte**
WA_R1	**MIB**	**Register to store and send Lower Byte**
WD_R	**MIB**	**Register to store and send character to Memory**
RD_R	**MIB**	**Register to store and receive character from Memory**

State Machine



_____This state machine is representative how the MIB's state machine works. Both WClk and RClk are used to stay in that state for at 120ns.

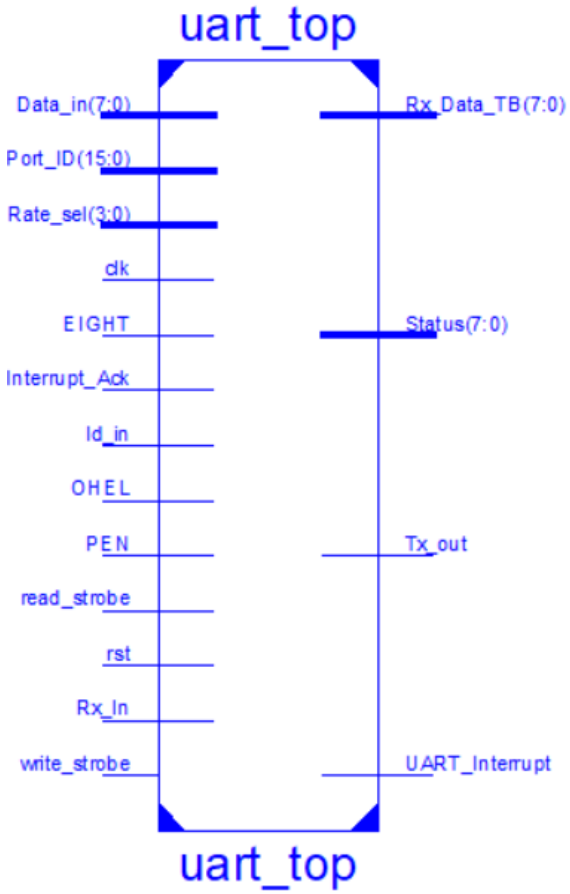
Verification

The verification was created by connecting the project to a module cell ram that simulates the working behavior of Memory. At the bottom, the registers are taking the correct data, especially when ASCII value is received. The value are then seen taken to the Tramel Blaze and outputted onto the Transmit line.

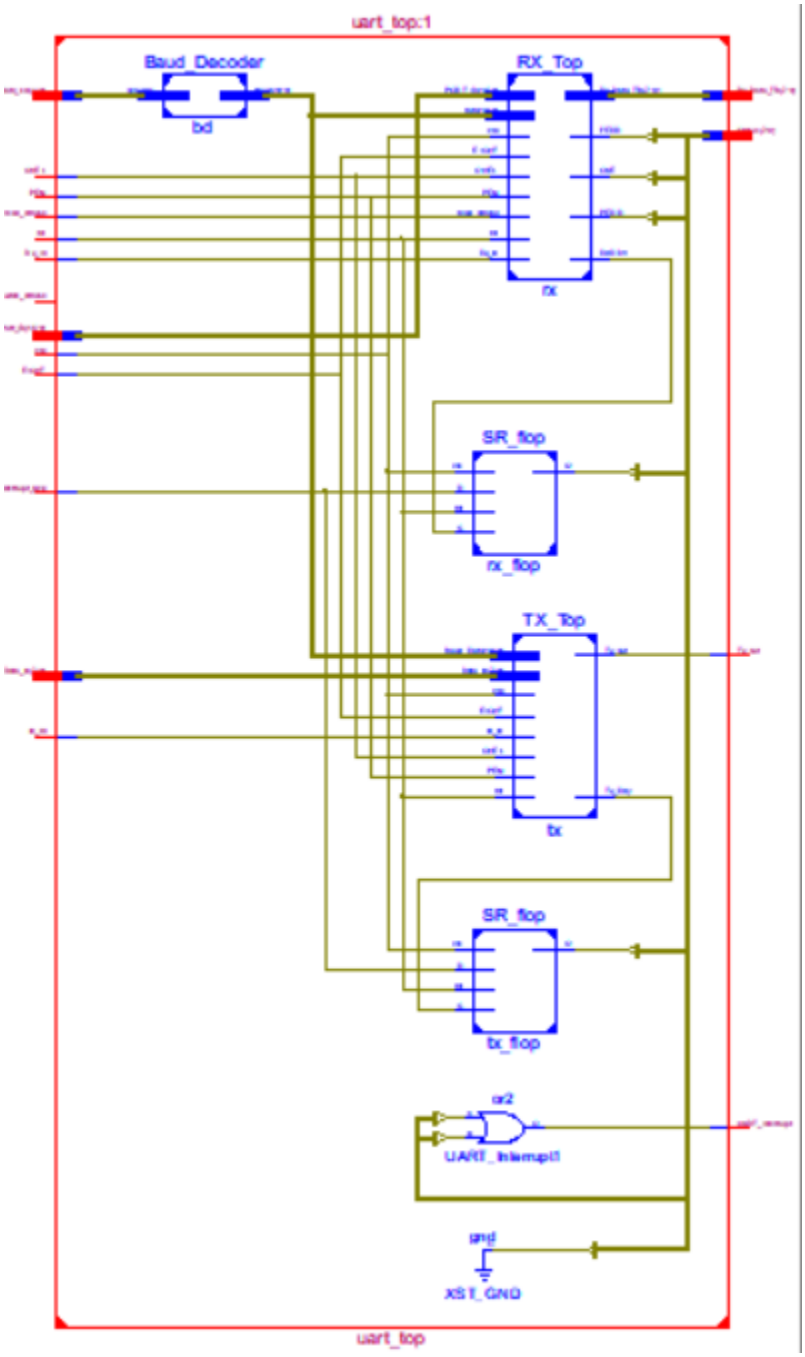
Universal Asynchronous Receiver and Transmitter(UART)

_____UART consists of 3 components, the Transmit Engine(TX) and Baud Rate Decoder, and Receive Engine (RX). TX is used to shift data out to the Transmit line of the USB while RX is responsible for shifting data into its register before outputting them to the TramelBlaze to be processed. Both TX and RX needs Baud Rate which is provided by a user inputted 4-bit value that will translate to appropriate values that dictate the speed of transmitting and receiving data.

An Interrupt signal is also implemented in the UART to cause the TramelBlaze to enter the Interrupt Service Routine. The Interrupt signal named UART_Interrupt is asserted whenever either the TX or RX are ready to work.



Detailed Block Diagram



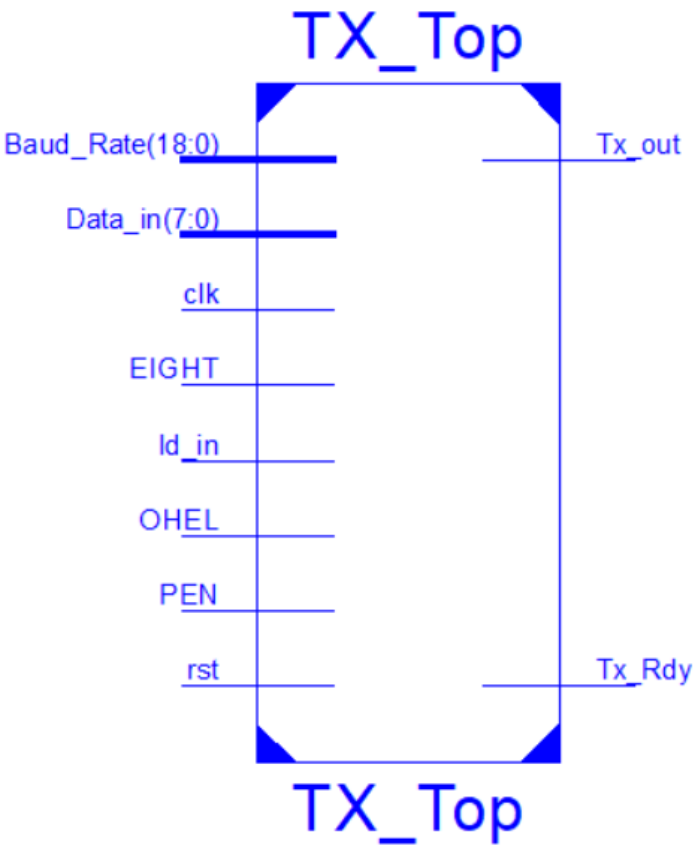
Signal	Size(bits)	I/O	Connected to
clk	1	I	100MHz Crystal Oscillator
rst	1	I	AISO
Port_ID	16	I	TramelBlaze
Data_in	8	I	TramelBlaze
Rate_Sel	4	I	Baud Rate Decoder
PEN	1	I	On-Board Switches
EIGHT	1	I	On-Board Switches
OHEL	1	I	On-Board Switches
write_strobe	1	I	TramelBlaze
read_strobe	1	I	TramelBlaze
Rx_In	1	I	USB
Rx_Out	1	O	TramelBlaze
Status	1	O	TramelBlaze&On-Board LEDs
UART_Interrupt	1	O	Pulse Maker
Tx	1	O	USB

1.1.3 Transmit Engine

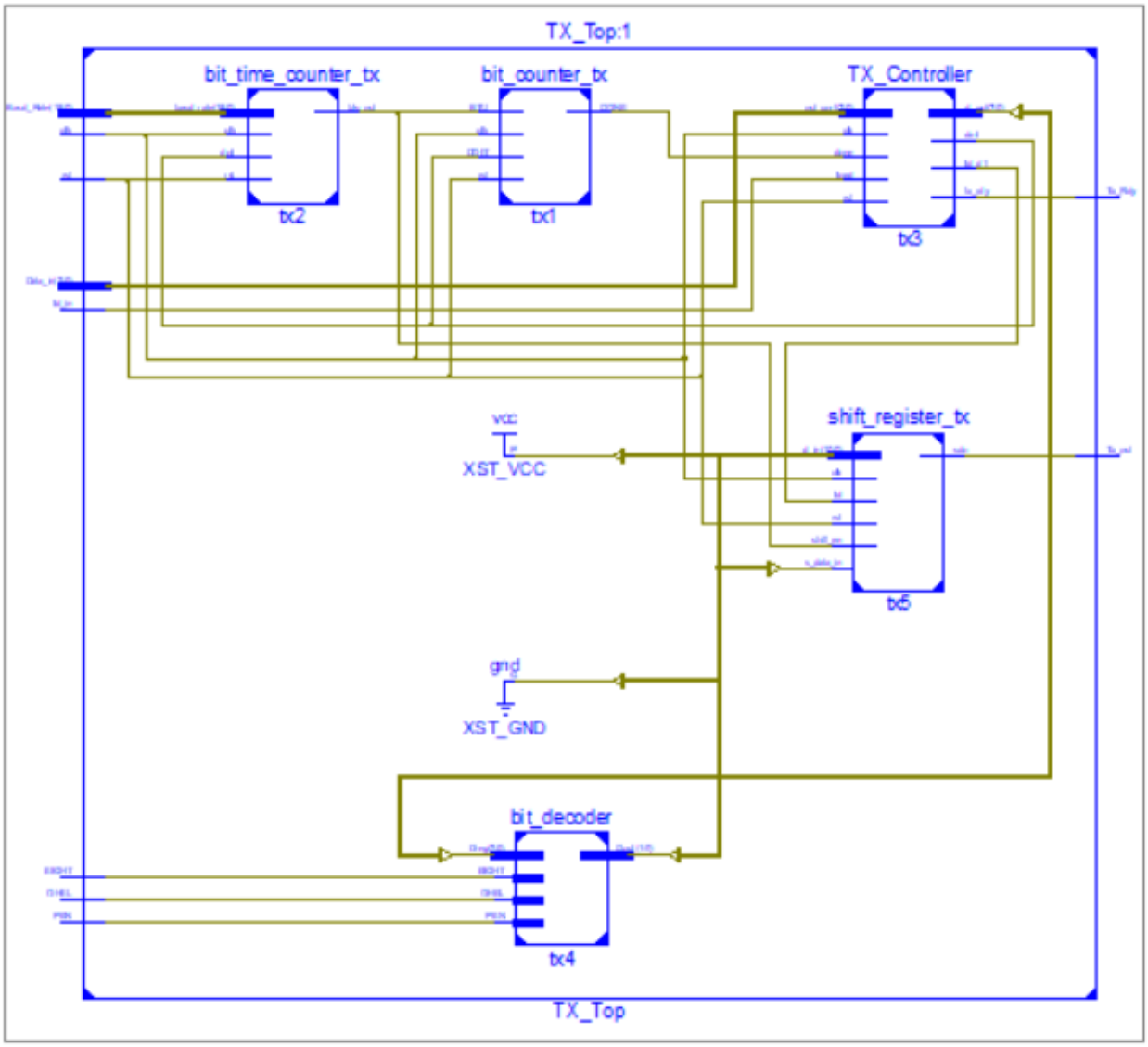
1.1.3.1 Details upon Transmit Engine

Transmit engine gets the parallel data, loads them into the Shift Register, and outputs the LSB. The data are 9, 10 bits and it depends on what are we doing. Start bit is always low and stop bit is always high. The bit count module counts until it hits 11 and all the data will shift out from the Shift Register. Bit Count Timer will wait until it hits the designated Baud rate, and then it will set the bit time to be up.

1.1.3.2 Top Level Diagram for Transmit Engine



1.1.3.3 Detail Diagram for Transmit Engine



I/O

Signal	Size	I/O	Connected to
clk	1	I	100MHz Crystal Oscillator
rst	1	I	AISO
ld_in	1	I	Write Strobe & Port ID
Baud_Rate	19	I	Baud Rate Decoder
Data_in	8	I	TramelBlaze
PEN	1	I	On-Board Switches
EIGHT	1	I	On-Board Switches
OHEL	1	I	On-Board Switches
Tx_Rdy	1	I	UART Interrupt
Tx_Out	1	I	USB

Register Map

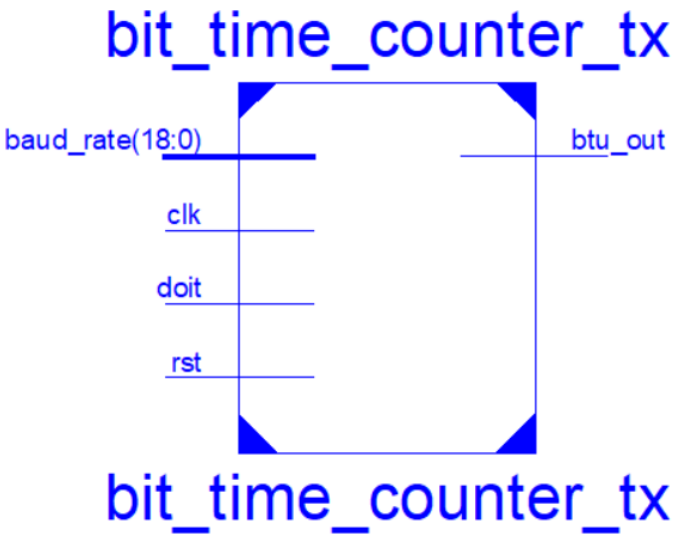
Register	Module	Description
count[18:0]	Time_Counter	Baud Rate Counter
count[3:0]	Bit_Count	Fixed number on how many times shift register can shift
Tx_Rdy	Controller	Acknowledge that Transmission is Ready
Load_D1	Controller	Delays data to reach DoIt and Shift Register for one clock period
DoIt	Controller	Takes in Data From Tramel Blaze
D_out [7:0]	Controller	Takes in Data from Tramel Blaze

bit_10_9	**Bit Decoder**	**Determine what bit 10 and 9 of shift register will do**
register[10:0]	**Shift Register**	**Prepare data to be shift out to a designated pin**
SD0	**Shift Register**	**Takes in Data from Tramel Blaze**

1.1.3.4 Transmit Bit_Time_Counter

Description

Bit Time Counter controls the speed where the system runs at. A Baud Rate value will come in the module and it will increment that values till Controller allows it. The value equals the baud_rate value and it will assert the signal, BTU and rst(reset) of the count back to 0.



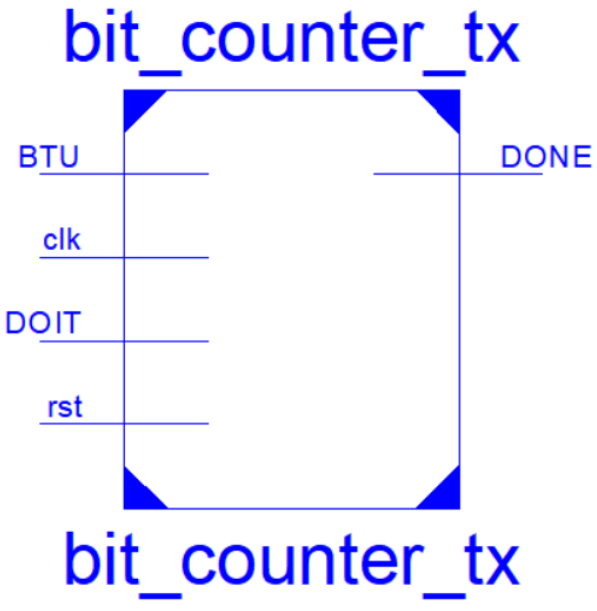
Signal	Size	I/O	Connected To
clk	1	I	100MHz Crystal Oscillator
rst	1	I	AISO
doit	1	I	Tx_Controller
baud_rate	19	I	Global Input
btu_out	1	O	Tx_Shift Register & Bit Count

1.1.3.5 Transmit Bit Counter

Description

The bit counter of Transmit make sure that the shift register shifts out only 11 bits while transmitting. This achieved through by creating a flop will increase every time the Transmit Bit Time Counter and Controller send the signals.

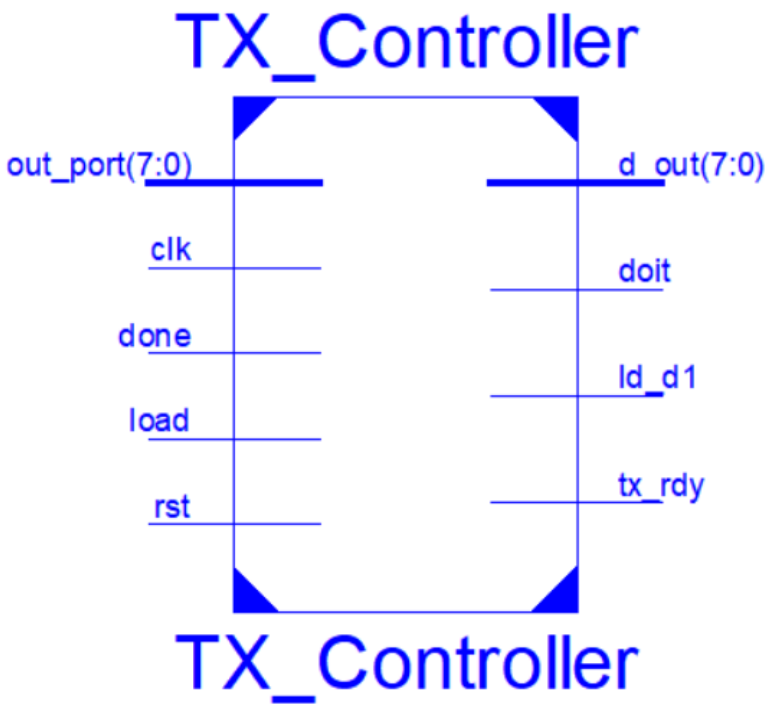
When Eight and PEN are enabled, the bit counter should change to 12 in order to hold for the lack of STOP bit.



1.1.3.6 Transmit Controller

Description

Transmit Controller has two SR flops and two D flip flp. Tx_Rdy tells the processor when the Engine is ready to be utilized. Dealy D flop load to Shift Register. A DoIt SR flop allows the two counters to count. 8-bit load register that take in data from TramelBlaze and transmit them to Bit Decoder and Transmit Shift Register.



6.1.2.2 Receive Engine

Description

Receive Engine has five major components, the Receive Bit Time Counter, State Machine, Receive Bit Counter, Receive Controller, and Receive Shift Register. There are many things that are identical to those of the Transmit Engine, but some has to be modified to allow all the part of the design to receive data rather than to transmit data.

State Machine determines when the Receive Engine will work, and it is also used to detect the Start Bit of the Transmitter to allow the Engine to change the values to run into the Bit Time Counter and Bit Counter.

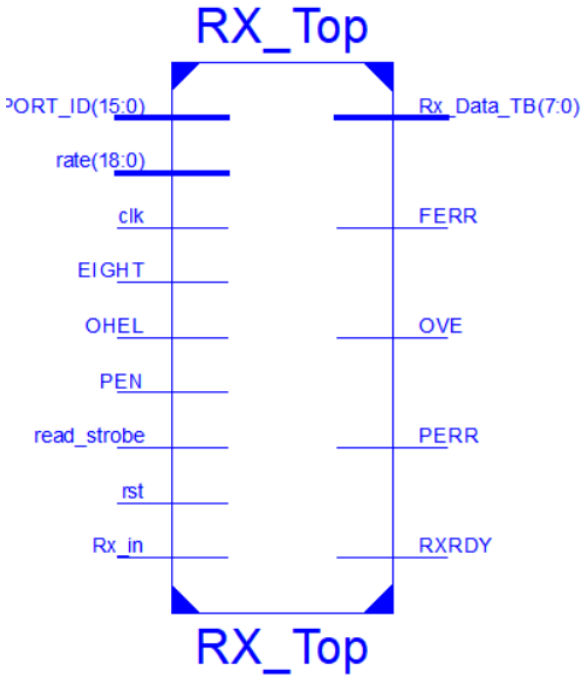
Bit Time Counter Controls the speed a when RX Engine operates at. The speed is determined entirely by the Baud Rate Decoder of UART. Whenever the State Machine Detects a Start Bit, the value of Baud Rate will temporarily be divided by 2 to calibrate the Engine in a way that will allow it to pick up values at the halfway mark of every bit.

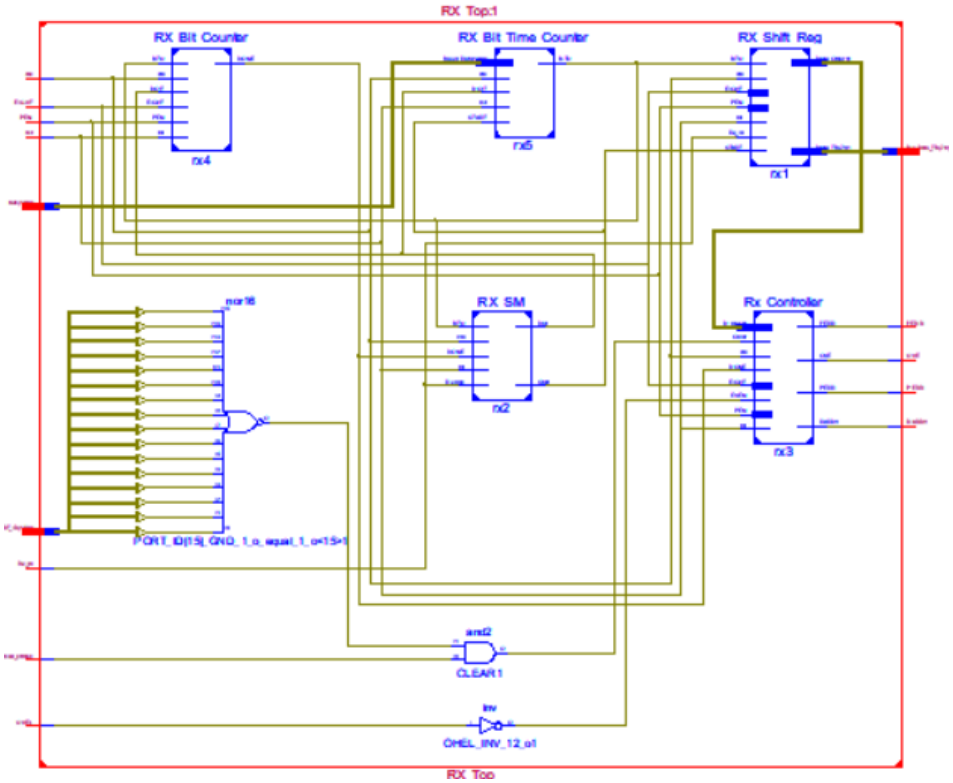
Receive Bit Counter is essentially the same as the Transmit Bit Count but rather than that it will give fixed value of how many bits to count, it entirely depends on whether the Transmitter is sending seven or eight bits and whether there will be a parity sent or not.

Receive Controller takes care of two responsibilities, the first is to send a signal to let the processor know it is ready to receive values and the second is to take the value received and check for errors.

The Receive Shift Register is similar to the Transmit Shift Register, except that will output two : a ten-bit value sent to the controller to check for errors, and an eight-bit value that will be sent straight to the TramelBlaze for processing.

After ensuring both the Transmit Engine and Receive Engine were working properly as individuals, a new ROM was developed in order to utilized both engines. Once ROM was properly developed, the design was connected to Serial Terminal to test its capacities.

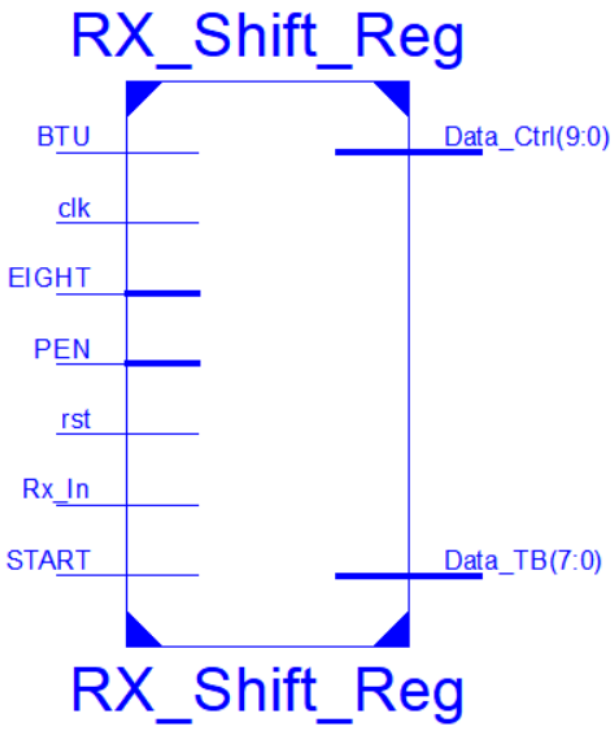

RX_Top
PORT_ID(15:0)
rate(18:0)
clk
EIGHT
OHEL
PEN
read_strobe
rst
Rx_in
Rx_Data_TB(7:0)
FERR
OVE
PERR
RXRDY
RX_Top


RX Top:1
RX Bit Counter
RX Bit Time Counter
RX Shift Reg
RX SM
Rx Controller
CLEAR1
RX Top

Receive Shift Register

Description

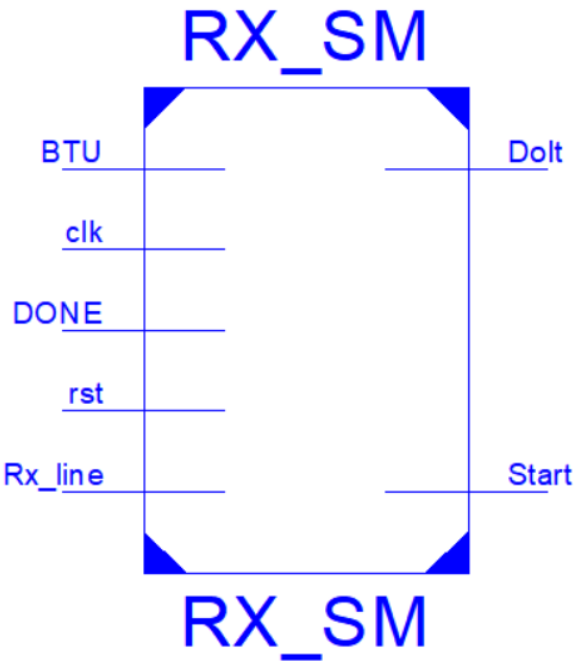
The Receive Shift Register is responsible for taking in the data from the Rx Line and shifting them into a register. Because of Eight, PEN, OHEL switch, the value is copied into two separate registers, Data_Ctril and Data_TB. Data_Ctrl is the data sent to the Controller that consists of every bit of information to calculate any errors. Data_TB is the data sent to the TramelBlaze, and this consists of only the ASCII value that was sent. This data is extracted using a combinational block that will shift the value further and fill the unnecessary information with 0's.



State Machine

Description

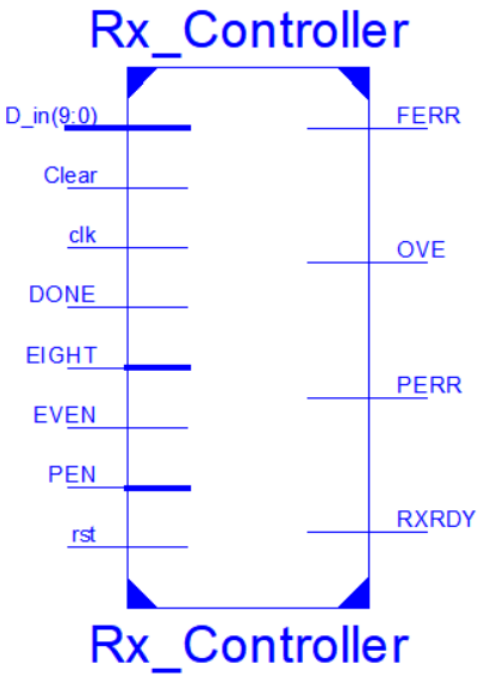
The Receive Engine State Machine determines the state of the signal it is receiving. It is illustrated to show how the data in this machine flows. The Start bit is most significant bit of information the State Machine needs to detect. It allows the Engine to enter the second state and allows itself to prepare for the oncoming data.Once this bit has been detected, the State Machine allows the counters to count and begins collection. Once the Bit Count has received the necessary number of bits, it will return to the IDLE state and it will wait for the next Start bit.



Receive Controller

Description

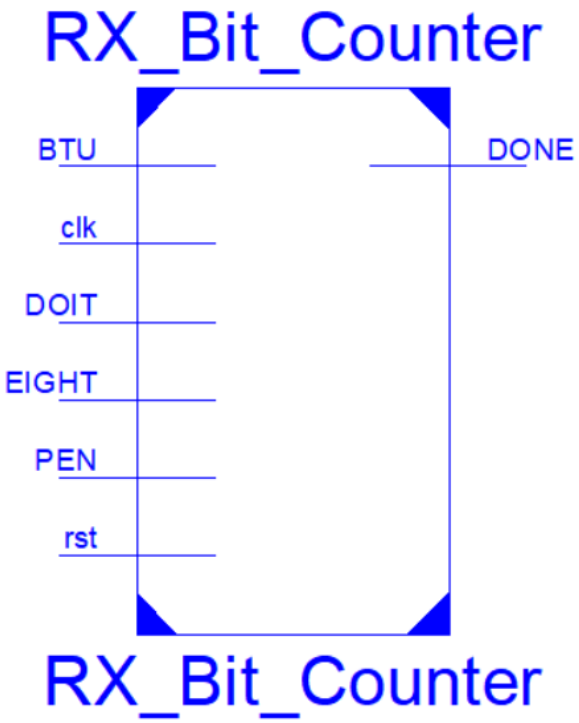
The Receive Controller consists of multiple SR flops, all with a different purpose. The first flop sends out a signal called Rx_Rdy which is asserted whenever the Engine is done receiving data from the Transmitter. When active, this signal asserts the interrupt to the Tramel Blaze. Next, PEN, OHEL, EIGHT are each signal sent to one of the Nexys 4's on-board LEDs. The LED turns on whenever the generated parity does not match the received parity which indicates that there is an error in the transmission. Next the FERR flop sends a signal to LEDs whenever the Stop Bit is not detected after a transmission. Lastly, the OVF flop sends a signal to the LEDs whenever a Stop bit has occurred but there is still more data coming in. These signals allow any user to be able to detect the errors that may occur during the transmission of data to the Receive Engine.



Receive Bit Counter

Description

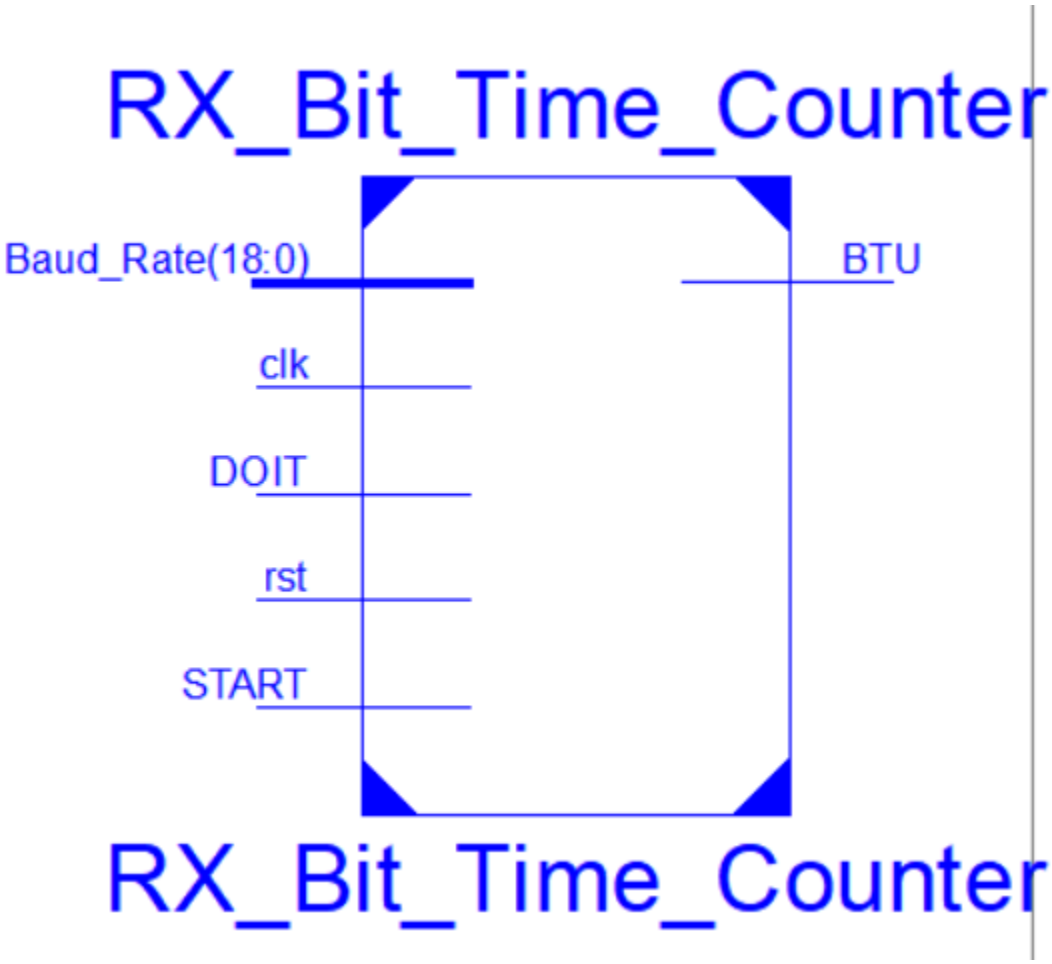
The Receive Bit Count behaves similarly to the Transit Bit Count, where it counts the number of bits to ensure that the Engine is picking up the correctly number of bits. But in addition to this, the Engine is also tasked with the responsibility of changing the compared values due to the difference in how many bit needs to received based on the EIGHT, PEN, OHEL.



Receive Bit Time Counter

Description

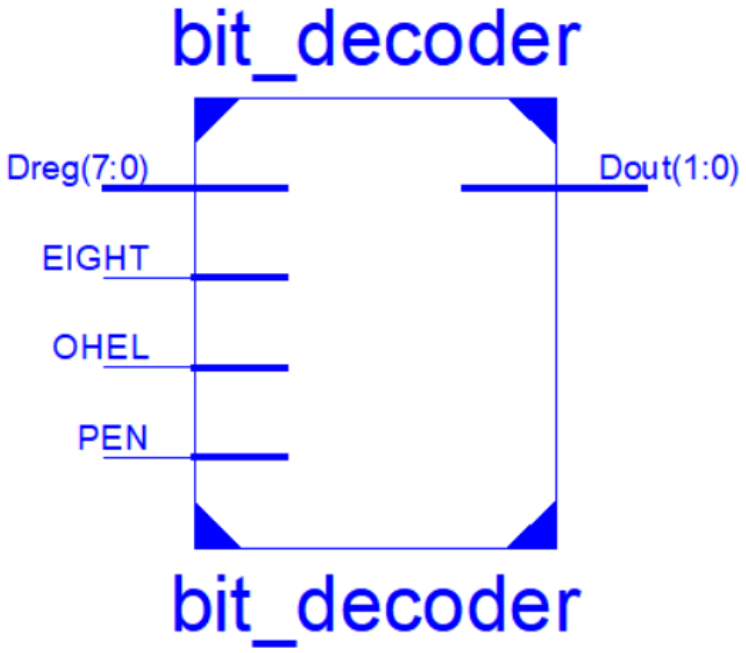
The Receive Bit Time Counter is similar to the Transmit Bit Time Counter, where it determines the speed at which the Engine runs based off the value inputted by the Baud Rate Decoder. The difference between two Bit Time Counter is that Receive engine does not hold the same value whenever it receives a bit. Whenever the Engine detects a Start Bit, the value of the Baud Rate will temporarily be divided by two so that the Receive Engine can sample the transmitted data at the halfway mark of every bit. This allows the Engine to accurately pick up data and allows room for any sort of drift.



1.1.3.7 Bit Decoder

Description

Bit decoder determines the values that will be placed for 10^{th} and 9^{th} bit location of the Transmit Shift Register. It also based on Eight, Parity Enable and OHEL which will come from Nexys 4 on board switches. Above is the logic table.

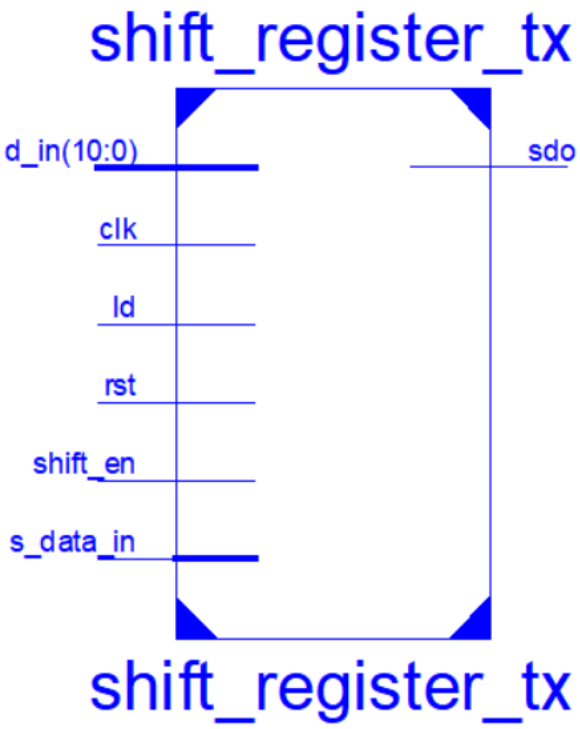


EIGHT	PEN	OHEL	10	9
0	0	0	1	1
0	0	1	1	1
0	1	0	1	^Dreg [6:0]
0	1	1	1	~^Dreg[6: 0]
1	0	0	1	Dreg[7]
1	0	1	1	Dreg[7]
1	1	0	^Dreg[7:0]	Dreg[7]
1	1	1	~^Dreg[7: 0]	Dreg[7]

1.1.3.8 Shift Register

Description

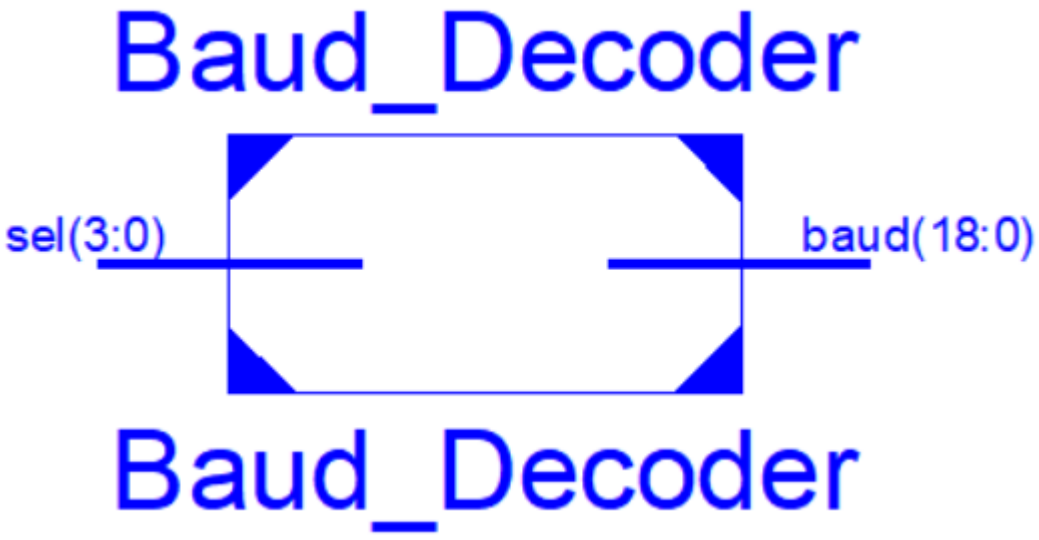
Shift Register for Transmit Engine is responsible for taking in data from Controller and Bit Decoder and it shifts out based on the speed of Baud Rate. When the data is shifted out to USB line, it will shit in 1's to identify the behavior of Engine which is not being transmitted.



1.1.4 Baud Decoder

Description

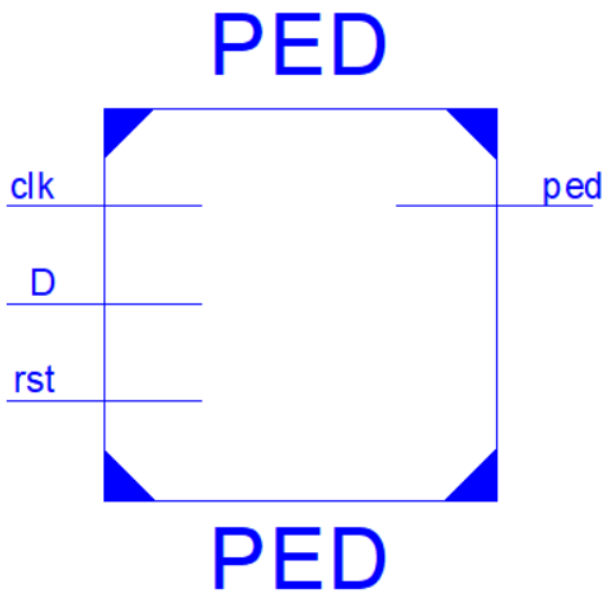
Baud Decoder is a 4-bit Decoder that determines the speed at which the Transmit Engine run at.



1.1.5 Pulse Maker

Description

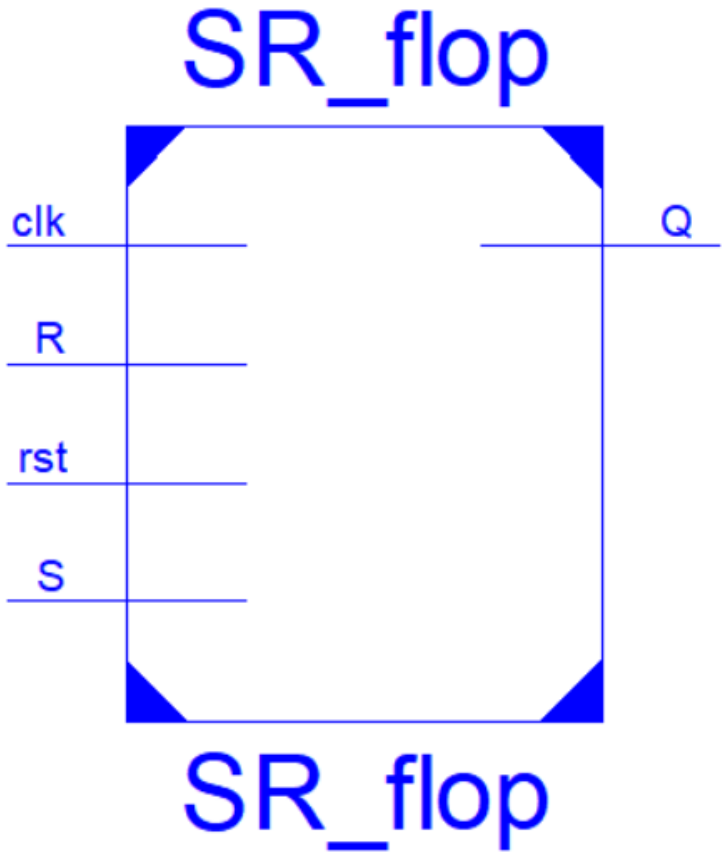
PED is my Pulse Maker and inside that block, there are 2 flip-flops and one AND logic. D_in input takes in an interrupt signal from the UART and that will set the Interrupt through the pulse and sent to an SR flop, which manages the interrupt of Tramel Blaze.



1.1.6 Set Reset Flop

Description

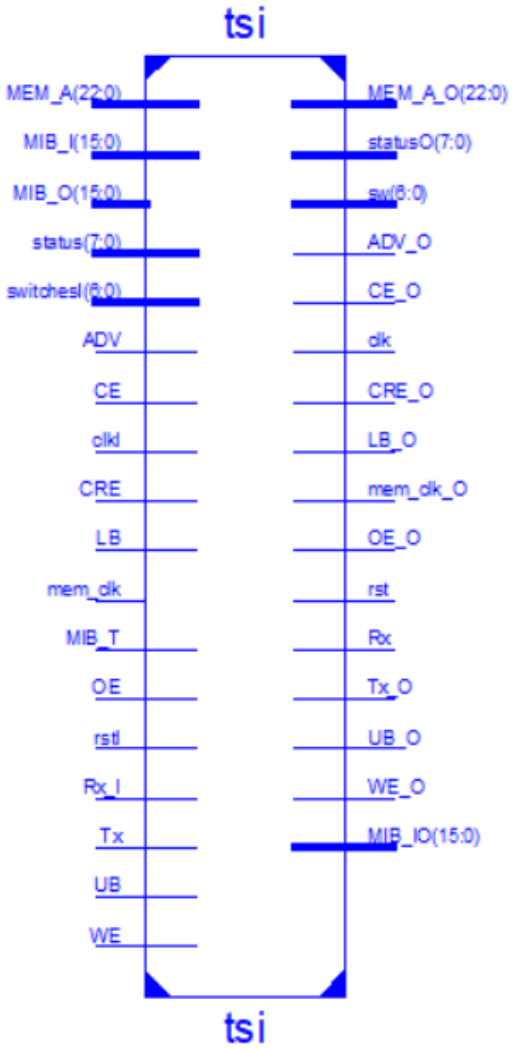
SR flop is connected to the Pulse Maker, which sets output goes high whenever the input is generated. Input is connected to Tramel Blaze's interrupt Acknowledge, which is when it is asserted will clear the value of output and goes low.



Technology Specific Instantiation (TSI)

Description

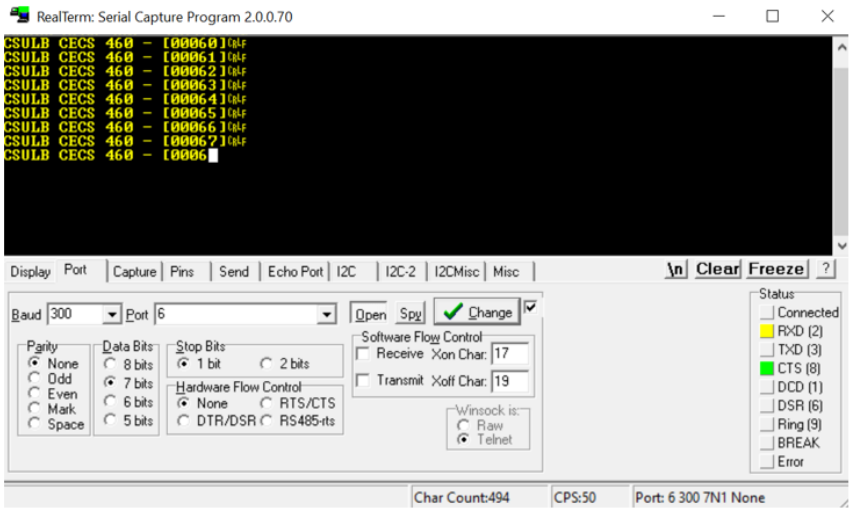
TSI combined with the Core provides a more organized and chip specific design. With TSI, the design is improved by providing easier access to I/O cells which are more specific towards the chip's design. In this design, the usage of an IOBUF was necessary to be able access the memory through a bidirectional wire. Without this type of buffer, the design would not be able to access the memory as it does.



I/O

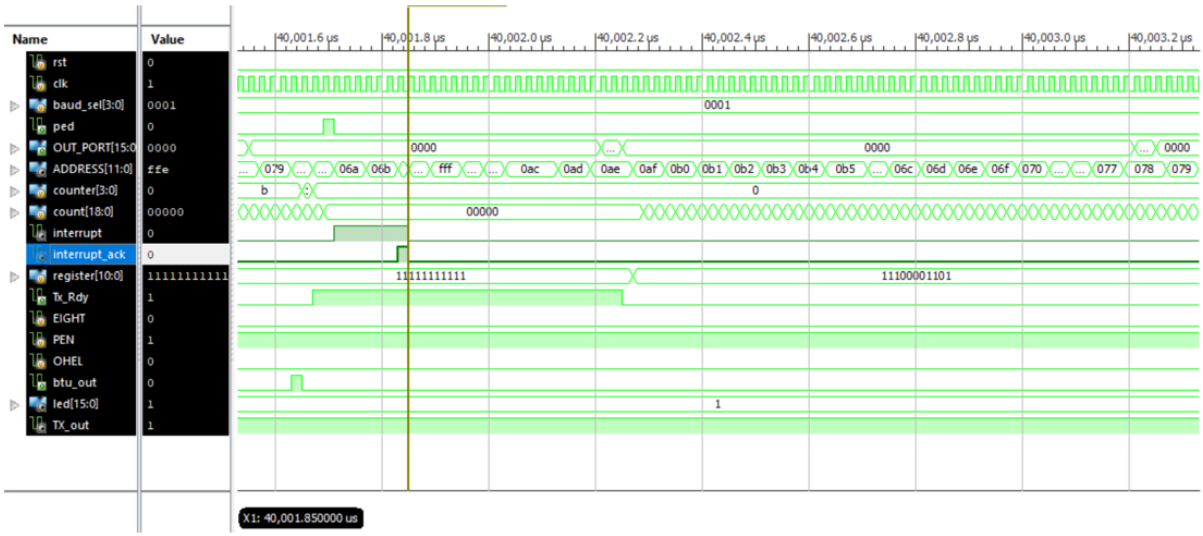
Signal	Size (bits)	I/O	Connected to
clkI	1	I	100MHz Crystal Oscillator
resetI	1	I	On-board Buttons -- BTNC
switchesI	7	I	On-board Switches
clk	1	O	Core
reset	1	O	Core
Switches	7	O	Core
Tx	1	I	UART
Rx_I	1	I	UART
Tx_O	1	O	USB
Rx	1	O	USB
MEM_A	23	I	MIB
status	8	I	UART
mem_clk	1	I	Always Logic '0'
CE	1	I	MIB State Machine
WE	1	I	MIB State Machine
OE	1	I	MIB State Machine
ADV	1	I	Always Logic '0'
CRE	1	I	Always Logic '0'
UB	1	I	Always Logic '0'
LB	1	I	Always Logic '0'
MEM_A_O	23	O	Micron Memory
statusO	8	O	On-board LEDs
mem_clk_O	1	O	Micron Memory
CE_O	1	O	Micron Memory
WE_O	1	O	Micron Memory
OE_O	1	O	Micron Memory
ADV_O	1	O	Micron Memory
CRE_O	1	O	Micron Memory
UB_O	1	O	Micron Memory
LB_O	1	O	Micron Memory
MIB_T	1	I	MIB State Machine
MIB_I	16	I	MIB
MIB_O	16	O	MIB
MIB_IO	16	IO	Micron Memory

Chip Level Verification



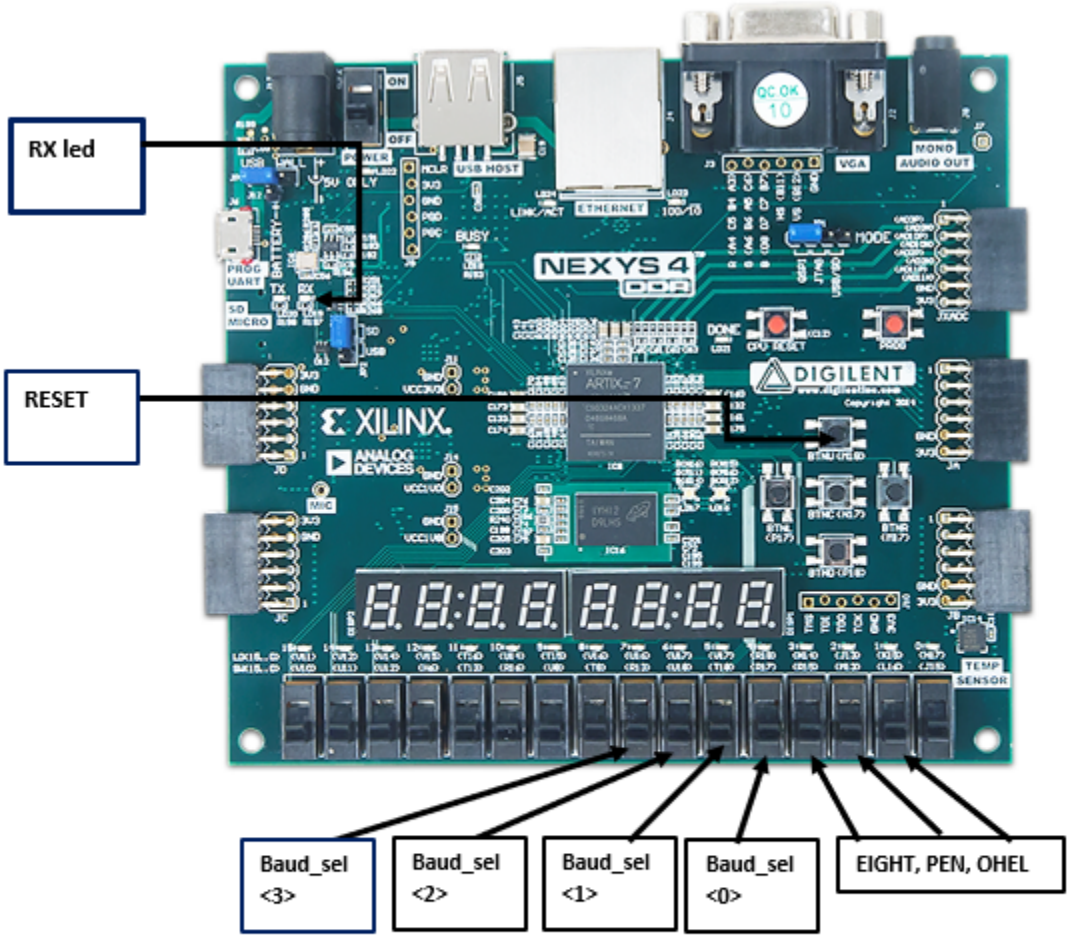
When loaded the bit file to board, it will show as RX led will light up and leds above the switches will travel to right. If the code is done successfully, it will show on RealTerm Serial Capture Program as above.

Wave Form



Waveform tells a simulation of Transmit Engine with Key Signals passing. TX_rdy shows the data entering the TX engine being shifted out serially to a receiver. Then interrupt and interrupt_ack should be active.

The Main Key to Look at On Board and Switches



When we upload our bit file, the main thing we have to check is whether the leds above are switches are travelling left. And at the same tim RX led should light up. Lighting Up RX Led means that the transmit engine works well.