

CECS 460 System On Chip Design
Spring 2018



Chanartip Soonthornwan

Chip Specification
4/17/2018

Chip Specification

2 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

This document contains information proprietary to the CSULB student that created the file - any reuse without adequate approval and documentation is prohibited.

Class: CECS460 System on Chip Design

Project name: Project3_UART_RX

In submitting this file for class work at CSULB I am confirming that this is my work and the work of no one else.

In the event, other code sources are utilized I will document which portion of code and who is the author.

In submitting this project, I acknowledge that plagiarism in student project work is subject to dismissal from the class.

Chip Specification

3 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

Intentionally
left blank.

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

Table of Content

1	Introduction	8
1.1	Purpose	8
2	External Documents	9
2.1	PicoBlaze.....	9
2.2	Nexy 3	9
2.3	ASCII Table	9
3	Requirements	10
3.1	Interface Requirements	10
3.2	Physical Requirements	10
4	Top Level Design	11
4.1	Description	11
4.2	Block Diagram	12
4.3	Data Flow Description	12
4.4	I/O.....	12
4.4.1	Signal Names	12
4.4.2	Pin Assignments.....	12
4.4.3	Electrical Characteristics	12
4.5	Clocks.....	12
4.6	Resets	12
4.7	Software	12
5	Externally Developed Blocks	13
5.1	TramelBlaze	13
6	Internally Developed Blocks	14
6.1	SOPC Core	14
6.2	Universal Asynchronous Receiver Transmitter(UART).....	16
6.2.1	Transmit Engine.....	18
6.2.2	Receive Engine	27
6.3	Address Decoder	40
6.4	Positive Edge Detector (PED).....	41
6.5	Asynchronous-In-Synchronous-Out Reset (AISO Reset).....	42
7	Chip Level Verification	43

Chip Specification

5 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

7.1	Transmit Engine	43
7.2	Receive Engine.....	45
8	Chip Level Test	47
9	Appendix	48

Table of Appendices

Appendix A	SOPC Code	48
Appendix B	UART Transmit Engine Source Code.....	50
Appendix C	Transmit Engine Source Code.....	53
Appendix D	Baud Decoder Source Code	56
Appendix E	RS Flop Source Code	57
Appendix F	Shift Register Function Source Code	58
Appendix G	Shift Register Source Code	59
Appendix H	UART_RX Code	60
Appendix I	RX_Control Code	62
Appendix J	RX_DATAPATH Code	65
Appendix K	RX_REMAP Code	68
Appendix L	RX_SHIFT_REG Code.....	69
Appendix M	PED Source Code.....	70
Appendix N	Address Decoder Source Code.....	71
Appendix O	AISO Reset Source Code	72
Appendix P	Assembly Code for TX Engine Instruction Memory	Error! Bookmark not defined.

Table of Figures

Figure 1: ASCII Table	9
Figure 2: TramelBlaze Block Diagram	13
Figure 3: SOPC_CORE Diagram	14
Figure 4: SOPC_CORE Detail Diagram.....	15
Figure 5: UART Top Diagram.....	16
Figure 6: UART Top Detail Diagram	17
Figure 7: Transmit Engine Module.....	18
Figure 8: UART TX Detail Block Diagram	19
Figure 9: Transmit Engine Verification	20
Figure 10: Baud Decoder Module.....	21
Figure 11: Baud Decoder Verification.....	21
Figure 12: Bit Time Counter Module	22
Figure 13: Bit Time Counter Verification	22

Chip Specification

6 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

Figure 14: Bit Counter Module.....	23
Figure 15: Bit Counter Verification.....	23
Figure 16: Shift Register Module	24
Figure 17: Shift Register Detail Block Diagram.....	24
Figure 18: Shift Register Verification.....	26
Figure 19: UART_RX block diagram.....	27
Figure 20: UART_RX Detail Block Diagram	27
Figure 21: UART_RX Verification Part 1	28
Figure 22: UART_RX Verification Part 2.....	29
Figure 23: RX_Control_State_Machine Diagram	30
Figure 24: RX_Control Verification part1	31
Figure 25: RX_Control Verification Part2	32
Figure 26: RX_Datapath Block Diagram	33
Figure 27: RX_Datapath Detail Block Diagram.....	34
Figure 28: RX_Datapath o_rxdy Verification	35
Figure 29: RX_Datapath o_perr Verification	36
Figure 30: RX_Datapath o_ferr Verification.....	36
Figure 31: RX_Datapath o_ovf Verification	37
Figure 32: RX_Shift_Register	38
Figure 33: RX_Shift_Register Verification	38
Figure 34: RX_Remap Diagram	39
Figure 35: RX_Remap Verification.....	39
Figure 36: Address Decoder Module.....	40
Figure 37: Address Decoder Verification	40
Figure 38: PED Module	41
Figure 39: PED Verification	41
Figure 40: AISO Reset module.....	42
Figure 41: AISO verification	42
Figure 42: TX Engine with TramelBlaze Verification	43
Figure 43: Transmitted result on RealTerm.....	44
Figure 44: UART_RX receives 8'h3A from the file.....	45
Figure 45: UART_TX transmits 8'h3A to a file.	45
Figure 46: Full UART with TramelBlaze on SerialTerminal	46

Table of Tables

Table 1: Baud Rate Switches	11
Table 2: Bit controls switches.....	11
Table 3: TramelBlaze I/O	13
Table 4: SOPC_CORE I/O	15
Table 5: UART I/O	17
Table 6: Transmit Engine I/O	20
Table 7: Transmit Engine Register Map.....	20

Chip Specification

7 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

Table 8: Baud Decoder Table look up	21
Table 9: Baud Dedecoder I/O	21
Table 10: Bit Time Counter I/O	22
Table 11: Bit Counter I/O	23
Table 12: Shift Register Encoding table	25
Table 13: Shift Register I/O	25
Table 14: UART_RX I/O.....	28
Table 15: RX_Control Block Diagram	30
Table 16: RX_Control I/O	31
Table 17: RX_Datapath I/O	35
Table 18: RX_Shift Register I/O	38
Table 19: RX_REMAP I/O.....	39
Table 20: Address Decoder I/O	40
Table 21: PED I/O	41
Table 22: AISO I/O.....	42

Chip Specification

8 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

1 Introduction

This chip specification document describes details of three major components, Memory Interface Block(MIB), Universal Asynchronous Receiver and Transmitter(UART), and a Microprocessor (TramelBlaze) developed by John Tramel. This System On Programmable Chip (SOPC) provides a capability

1.1 Purpose

Providing UART a transmission protocol with a selectable baud rate, bits transmitted, parity enable/disable, and odd/even parity bit.

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

2 External Documents

2.1 PicoBlaze

2.2 Nexy 3

2.3 ASCII Table

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	Space	64	40	100	0	96	60	140	96	60	140	96
1	1	001	SOH (start of heading)	33	21	041	!	65	41	101	A	97	61	141	97	61	141	a
2	2	002	STX (start of text)	34	22	042	"	66	42	102	B	98	62	142	98	62	142	b
3	3	003	ETX (end of text)	35	23	043	#	67	43	103	C	99	63	143	99	63	143	c
4	4	004	EOT (end of transmission)	36	24	044	\$	68	44	104	D	100	64	144	100	64	144	d
5	5	005	ENQ (enquiry)	37	25	045	%	69	45	105	E	101	65	145	101	65	145	e
6	6	006	ACK (acknowledge)	38	26	046	&	70	46	106	F	102	66	146	102	66	146	f
7	7	007	BEL (bell)	39	27	047	'	71	47	107	G	103	67	147	103	67	147	g
8	8	010	BS (backspace)	40	28	050	(72	48	110	H	104	68	150	104	68	150	h
9	9	011	TAB (horizontal tab)	41	29	051)	73	49	111	I	105	69	151	105	69	151	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	74	4A	112	J	106	6A	152	106	6A	152	j
11	B	013	VT (vertical tab)	43	2B	053	+	75	4B	113	K	107	6B	153	107	6B	153	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	76	4C	114	L	108	6C	154	108	6C	154	l
13	D	015	CR (carriage return)	45	2D	055	-	77	4D	115	M	109	6D	155	109	6D	155	m
14	E	016	SO (shift out)	46	2E	056	.	78	4E	116	N	110	6E	156	110	6E	156	n
15	F	017	SI (shift in)	47	2F	057	/	79	4F	117	O	111	6F	157	111	6F	157	o
16	10	020	DLE (data link escape)	48	30	060	0	80	50	120	P	112	70	160	112	70	160	p
17	11	021	DC1 (device control 1)	49	31	061	1	81	51	121	Q	113	71	161	113	71	161	q
18	12	022	DC2 (device control 2)	50	32	062	2	82	52	122	R	114	72	162	114	72	162	r
19	13	023	DC3 (device control 3)	51	33	063	3	83	53	123	S	115	73	163	115	73	163	s
20	14	024	DC4 (device control 4)	52	34	064	4	84	54	124	T	116	74	164	116	74	164	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	85	55	125	U	117	75	165	117	75	165	u
22	16	026	SYN (synchronous idle)	54	36	066	6	86	56	126	V	118	76	166	118	76	166	v
23	17	027	ETB (end of trans. block)	55	37	067	7	87	57	127	W	119	77	167	119	77	167	w
24	18	030	CAN (cancel)	56	38	070	8	88	58	130	X	120	78	170	120	78	170	x
25	19	031	EM (end of medium)	57	39	071	9	89	59	131	Y	121	79	171	121	79	171	y
26	1A	032	SUB (substitute)	58	3A	072	:	90	5A	132	Z	122	7A	172	122	7A	172	z
27	1B	033	ESC (escape)	59	3B	073	;	91	5B	133	[123	7B	173	123	7B	173	{
28	1C	034	FS (file separator)	60	3C	074	<	92	5C	134	\	124	7C	174	124	7C	174	
29	1D	035	GS (group separator)	61	3D	075	=	93	5D	135]	125	7D	175	125	7D	175	}
30	1E	036	RS (record separator)	62	3E	076	>	94	5E	136	^	126	7E	176	126	7E	176	~
31	1F	037	US (unit separator)	63	3F	077	?	95	5F	137	_	127	7F	177	127	7F	177	DEL

Figure 1: ASCII Table

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

3 Requirements

3.1 Interface Requirements

This design has eleven baud rates to select from four Nexys3 on-board switches, and three bits parity control for eight-bit, parity enable, and odd/even parity selection which are compatible to the industrial standard interface. Also, serially output through TX port and indicating a microprocessor with TXRDY signal.

3.2 Physical Requirements

There are seven switches and one button on Nexy3 being used. The button(up) is used as reset signal, the switches[7:4] are for baud rate selection, and switches[3:1] are parity checking control; switch[3] is for eight-bit data, switch[2] is for parity-bit enable, switch[1] is for odd/even parity bit.

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

4 Top Level Design

4.1 Description

The Top Level demonstrates interconnections between SOPC CORE and TSI where the SOPC CORE consists of the entire digital design of UART and a Microprocessor, while TSI contains references to the Spartan-6 libraries. Any I/O for the SOPC CORE must pass through the TSI before interreacting with the SOPC core. The core would utilize the UART to display banner and prompt stored in the Microprocessor ROM and interacts with a user input then display a response on a serial terminal display with a baud rate and controls set at the on-board switches

Switches[7:4]	Baud Rate	Bit Time	Nexy 3 Count
0000	300	3.3333ms	333333
0001	1200	833.33us	83333
0010	2400	416.66us	41667
0011	4800	208.33us	20833
0100	9600	104.16us	10417
0101	19200	52.083us	5208
0110	38400	26.041us	2604
0111	57600	17.361us	1736
1000	115200	8.6806us	868
1001	230400	4.3403us	434
1010	460800	2.1701us	217
1011	921600	1.0851us	109

Table 1: Baud Rate Switches

Switches[3:1]	Bits	Parity
000	7	Disable
001	7	Disable
010	7	Even
011	7	Odd
100	8	Disable
101	8	Disable
110	8	Even
111	8	Odd

Table 2: Bit controls switches

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

4.2 Block Diagram

4.3 Data Flow Description

4.4 I/O

4.4.1 Signal Names

4.4.2 Pin Assignments

4.4.3 Electrical Characteristics

- Buttons
 - 3.3V is considered as Logical 1
 - 0V is considered as Logical 0
- Switches
 - 1.8V is considered as Logical 1
 - 0V is considered as Logical 0

4.5 Clocks

Nexys 3 utilizes a 100 MHz crystal oscillator.

4.6 Resets

The all registers in SOPC will be synchronizing reset to a known state on a raising edge of a clock when an on-board mechanical reset button is pressed. The Asynchronized-In-Synchronize-Out (AISO_RST) module will generate the button pressed signal to a synchronized reset signal for the entire design.

4.7 Software

Full UART Program p.73

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

5 Externally Developed Blocks

5.1 TramelBlaze

Description

A 16-bit microcontroller that emulates the 8-bit PicoBlaze utilizing 4Kx16 bit ROM as instruction memory where the processor reads and performs the assembly program. In this application, TramelBlaze utilizes UART engine to communicate with a Serial Terminal as to display and receive an ASCII value according to which port_id the processor preferred.

Diagram

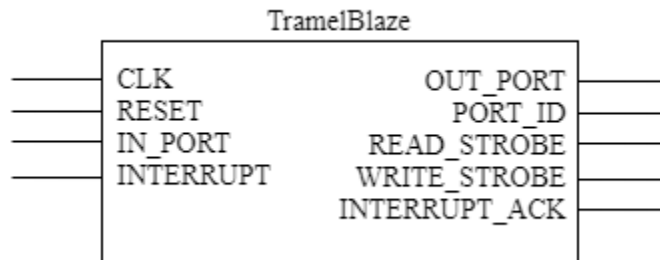


Figure 2: TramelBlaze Block Diagram

I/O

Signal	Size (bit)	I/O	Connected to
CLK	1	I	100MHz Crystal Oscillator
RESET	1	I	AISO_RST
INTERRUPT	1	I	RS_FLOP
IN_PORT	16	I	UART_TOP
OUT_PORT	16	O	UART_TOP
PORT_ID	16	O	Address Decoder
INTERRUPT_ACK	1	O	RS_FLOP
READ_STROBE	1	O	UART_TOP
WRITE_STROBE	1	O	UART_TOP

Table 3: TramelBlaze I/O

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

6 Internally Developed Blocks

6.1 SOPC Core

Description

The Top-level module demonstrates interconnections of controls and data paths through microprocessor (TramelBlaze), Universal Asynchronized Receiver Transmitter (UART), Address Decoder, Asynchronized-In-Synchronized-Out Reset signal module (AISO_RST), and UART interrupt and LED buffers.

AISO_RST is to generate a synchronized signal brings the whole circuit register to a known state since asynchronized reset signal from an on-board switch may cause a metastability for the register inputs and causes the system misbehaves.

UART_TOP is a full duplex serial communication protocol transmits and receives data through TX and RX line, generates interrupts to the microprocessor, and switches a word data or UART status output to the processor as requested.

TramelBlaze, a 16-bit microprocessor emulates 8-bit PicoBlaze. The processor is externally developed by John Tramel to execute assembly programs from a ROM and utilizes the UART to communicate on Serial Terminal.

Address Decoder, a combinational block to set the read or write strobe of an address (port_id) that has been sent out of the TramelBlaze.

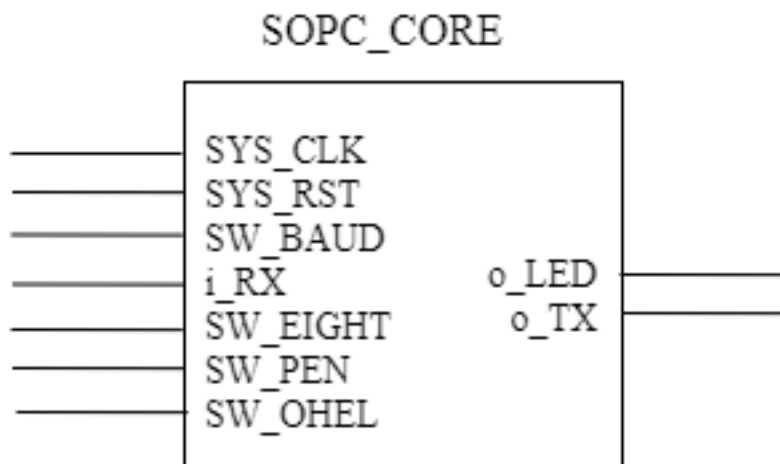


Figure 3: SOPC_CORE Diagram

Chip Specification

15 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

Detail Block Diagram

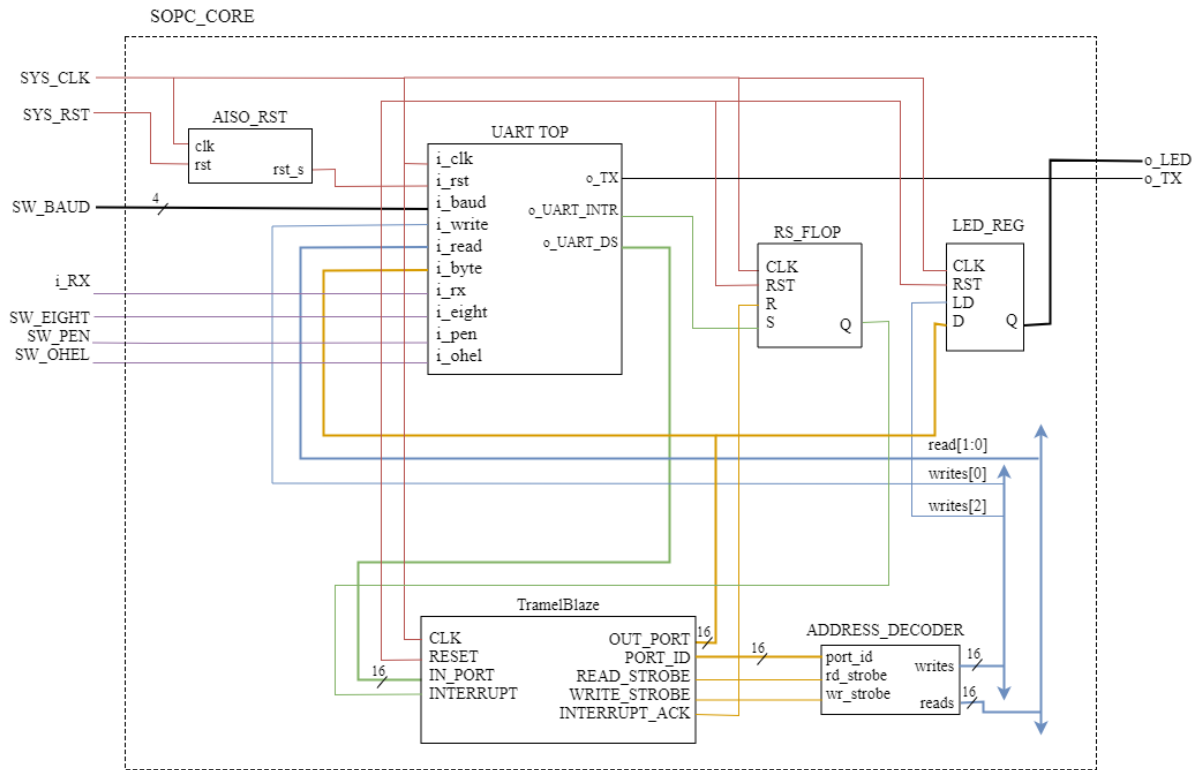


Figure 4: SOPC_CORE Detail Diagram

I/O

Signal	Size (bit)	I/O	Connected to
SYS_CLK	1	I	100 MHz Crystal Oscillator
SYS_RST	1	I	On-board button(up)
SW_BAUD	1	I	On-board switches[7:4]
SW_EIGHT	8	I	On-board switch[3]
SW_PEN	1	I	On-board switch[2]
SW_OHEL	1	I	On-board switch[1]
o_LED	1	O	On-board LED
o_TX	4	O	TX port

Table 4: SOPC_CORE I/O

Source Code: Appendix A

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

6.2 Universal Asynchronous Receiver Transmitter(UART)

Description

UART is a full duplex serial communication protocol device that could transmit a data package by a Transmit Engine through TX line, or receive a data package by Receive Engine through RX line. The UART consists of three major components, Receive Engine(RX), Transmit Engine(TX), and Baud Decoder, which RX responses for receiving data, TX for transmitting data, and Baud Decoder for decode the baud rate of the transmission to a specific bit time value for RX and TX to keep them synchronize to the transition. UART generates interrupts caused by RX or TX to TramelBlaze informing that which engine is ready for the next perform at the time. Moreover, UART dedicates the received data status as it flags all errors checking with RX or TX ready signals on o_UART_DS as it is requested.

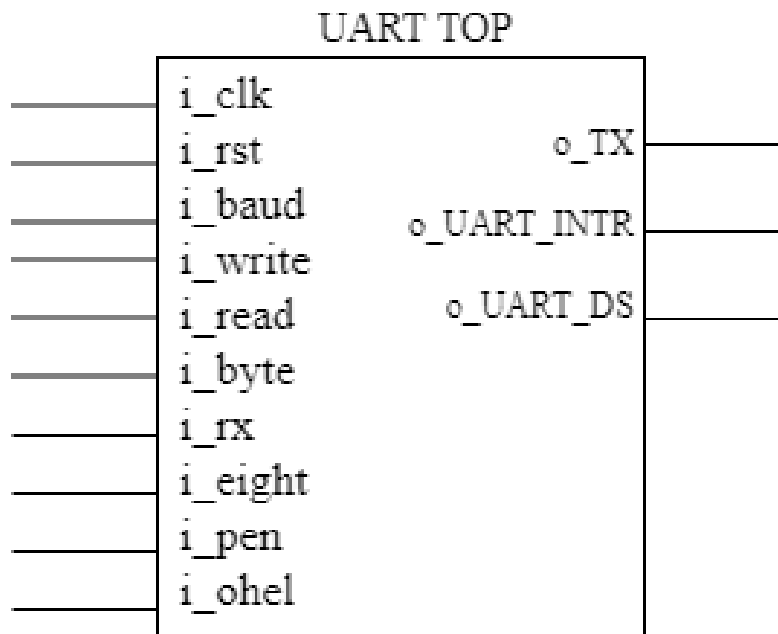


Figure 5: UART Top Diagram

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

Detail Block Diagram

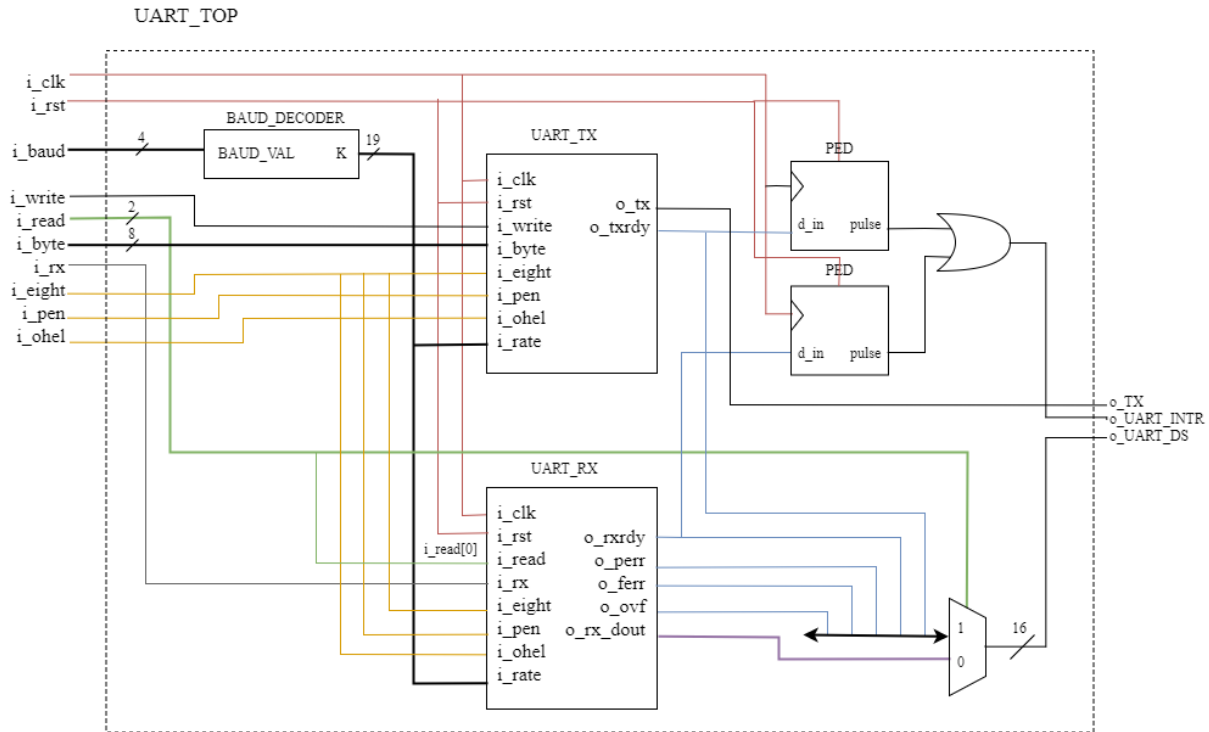


Figure 6: UART Top Detail Diagram

I/O

Signal	Size (bit)	I/O	Connected to
i_clk	1	I	100 MHz Crystal Oscillator
i_rst	1	I	On-board button(up)
i_baud	4	I	On-board switches[7:4]
i_write	1	I	Address Decoder
i_read	2	I	Address Decoder
i_byte	8	I	TramelBlaze
i_rx	1	I	RX port
i_eight	8	I	On-board switch[3]
i_pen	1	I	On-board switch[2]
i_ohel	1	I	On-board switch[1]
o_UART_INTR	1	O	TramelBlaze
o_UART_DS	8	O	TramelBlaze
o_tx	1	O	TX port

Table 5: UART I/O

Source Code: Appendix B

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

6.2.1 Transmit Engine

Description

Transmit Engine, half duplex serial digital communication protocol from a device to another without clock crossing interface. The transmit engine is responsible for shifting a set of data one bit out at a bit time throughout the Transmit line to a receiver device at the other end of an USB cable. The design consists four major components which are Baud Decoder, Bit Time Counter, Bit Counter, and Shift Register. First component, Baud Decoder, determines which frequency for the data transmission by decoding the Baud frequency selection into a constant for Bit Time Counter to generate a pulse signal of the Baud frequency called bit time up (BTU) signal. On each BTU, Bit Counter increase its counter that tracking how many bits of the transmission have been transmitted by one, while the BTU is also inform the Shift Register to shift right by one. The Shift Register concatenates a mark bit, a start bit, a 7 or 8-bit input with a parity bit become an 11-bit data package promptly to be loaded, and LSB-to-MSB shifting.

Block Diagram

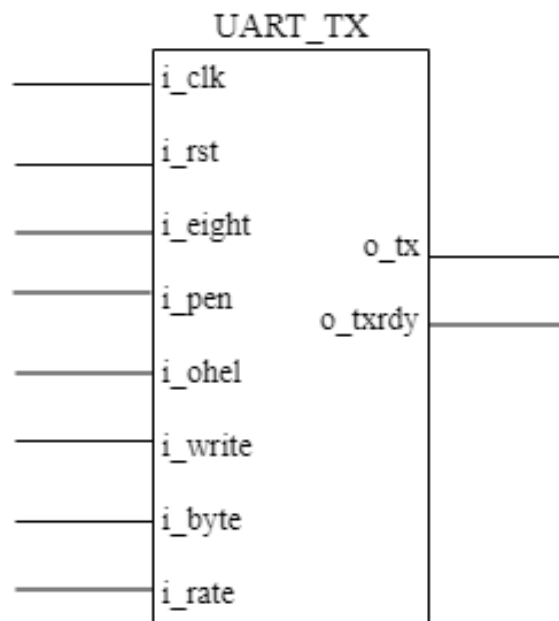


Figure 7: Transmit Engine Module

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

Detail Block Diagram

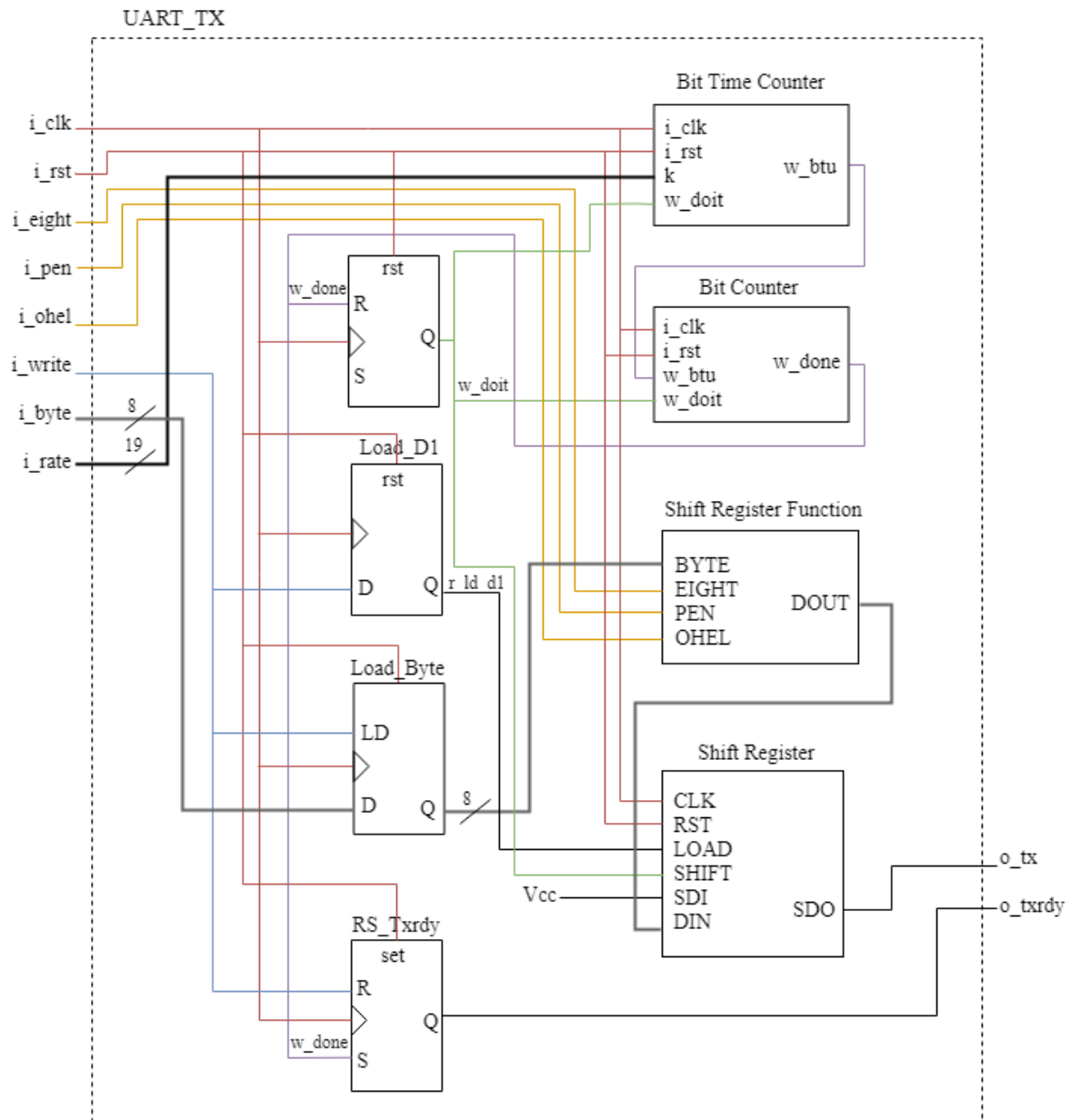


Figure 8: UART TX Detail Block Diagram

Chip Specification

20 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

I/O

Signal	Size (bit)	I/O	Connected to
i_clk	1	I	100 MHz Crystal Oscillator
i_rst	1	I	AISO reset
i_write	1	I	Load signal
i_byte	8	I	TramelBlaze_outport[7:0]
i_eight	1	I	On-board switch
i_pen	1	I	On-board switch
i_ohel	1	I	On-board switch
i_rate	19	I	Baud Rate
o_tx	1	O	USB
o_txrdy	1	O	Positive Edge Detector (PED)

Table 6: Transmit Engine I/O

Register Map

Register	Module	Description
o_txrdy	RS_Flop	Indicate that Transmission is ready
[18:0] count	Bit_Time_Counter	Baud Rate counter
[3:0] bit_count	Bit_Counter	Bit Time counter
r_ld_d1	Load_D1_Flop	Delays package data to reach Shift Register and delays w_doit for a clock period
[7:0] r_byte	Load_Byte_Flop	Holds input data from TramelBlaze

Table 7: Transmit Engine Register Map

Source Code: Appendix C

Verification

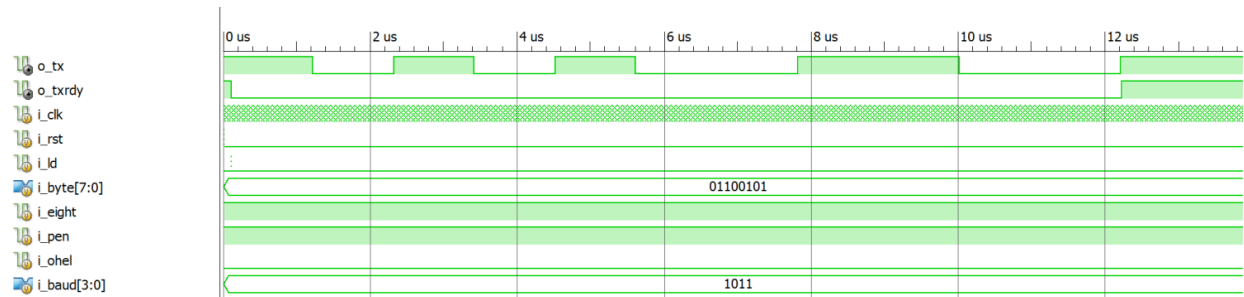


Figure 9: Transmit Engine Verification

Transmit Engine is verified as it is able to shift a data package out according to the parity bit control with a baud rate through the o_Tx line.

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

6.2.1.1 Baud Decoder

Description

A combinational logic block to convert the Baud Rate control from on-board switches to a counter for a pulse maker (Bit Time Counter) to generate a signal according to its requirement.

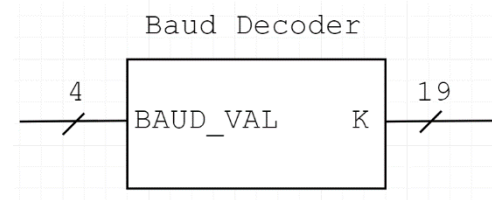


Figure 10: Baud Decoder Module

BAUD_VAL[3:0]	Baud Rate	Eng Not	Bit Time Count
0000	300	3.333 ms	333,333
0001	1200	833.33 us	83,333
0010	2400	416.66 us	41,667
0011	4800	208.33 us	20,833
0100	9600	104.16 us	10,417
0101	19200	52.083 us	5,208
0110	38400	26.041 us	2,604
0111	57600	17.361 us	1,736
1000	115200	8.6806 us	868
1001	230400	4.3403 us	434
1010	460800	2.1701 us	217
1011	921600	1.0851 us	109

Table 8: Baud Decoder Table look up

I/O

Signal	Size (bit)	I/O	Connected to
BAUD_VAL	4	I	On-board switches[7:4]
K	19	O	Bit Time Counter

Table 9: Baud Dedecoder I/O

Source Code: Appendix D

Verification

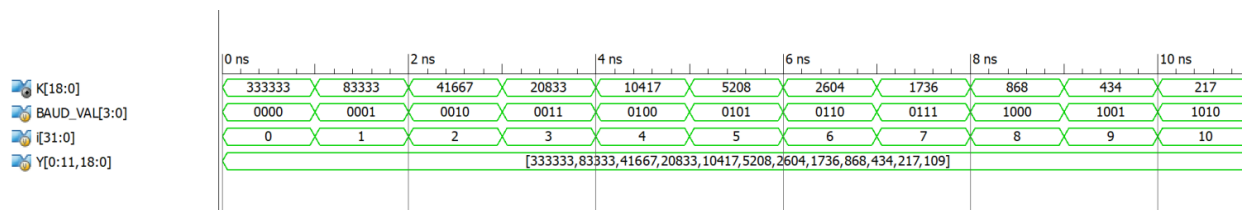


Figure 11: Baud Decoder Verification

Baud Decoder is verified as it provides correct bit-time-count constant(K) according to a table look-up output vector(Y) for each iteration(i).

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

6.2.1.2 Bit Time Counter

Description

Bit Time Counter generates pulse of a desire clock frequency according to Baud Rate count by incrementing a counter as it is allowed, then outputting a Bit Time Up signal and reset the counter.

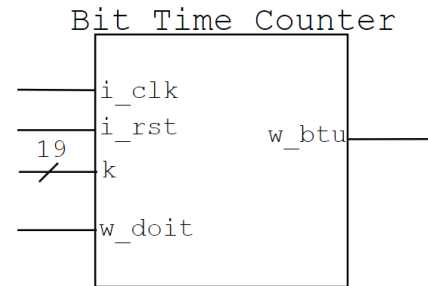


Figure 12: Bit Time Counter Module

I/O

Signal	Size (bit)	I/O	Connected to
i_clk	1	I	100 MHz Crystal Oscillator
i_rst	1	I	AISO reset
k	19	I	Baud Decoder
w_doit	1	I	Bit Counter
w_btu	1	O	Shift Register & Bit Counter

Table 10: Bit Time Counter I/O

Source Code: Appendix C

Verification

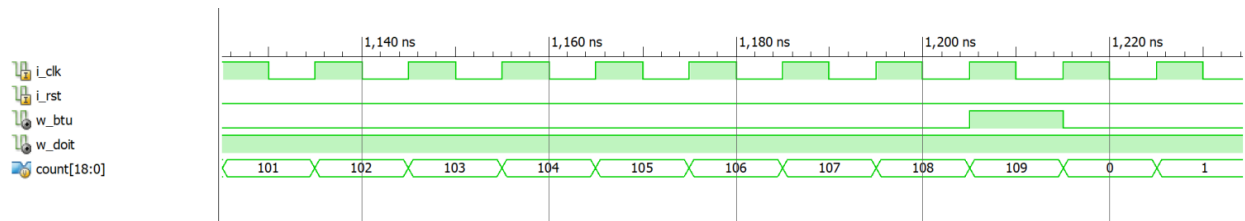


Figure 13: Bit Time Counter Verification

Bit Time Counter is verified as it generates a one-clock period pulse when the counter is counted to a specific baud rate constant.

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

6.2.1.3 Bit Counter

Description

Bit Counter counts number of bits in the transmission, which is 11 bits for this design, utilizing a counter to increment the bit counter at an active edge of Bit Time Up(BTU) signal. The counter will output Done signal to indicate the bit count is reached 11 bits and reset the counter.

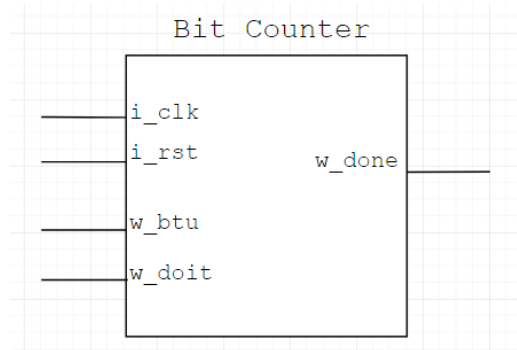


Figure 14: Bit Counter Module

I/O

Signal	Size (bit)	I/O	Connected to
i_clk	1	I	100 MHz Crystal Oscillator
i_rst	1	I	AISO reset
w_btu	1	I	Bit Time Counter
w_doit	1	I	Done RS Flop
w_done	1	O	Bit Time Counter & Txrdy Flop

Table 11: Bit Counter I/O

Source Code: Appendix C

Verification

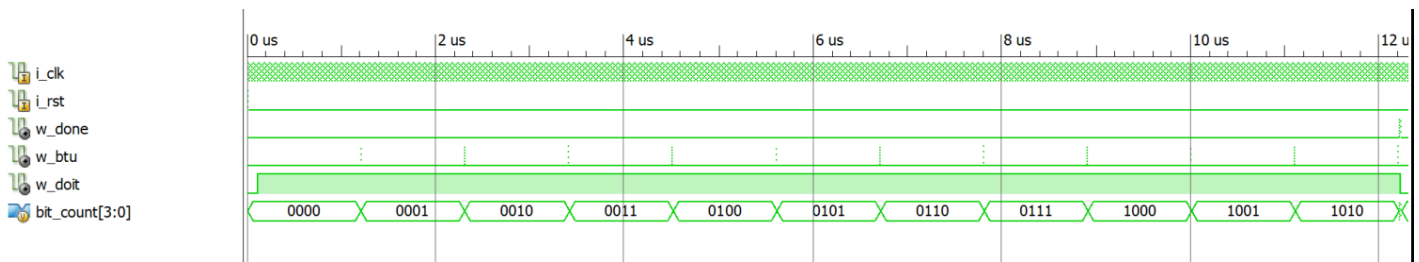


Figure 15: Bit Counter Verification

Bit Counter is verified as it generates w_done signal when bit_count reached 11₁₀.

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

6.2.1.4 Shift Register

Description

Serially shifting 11-bit encoded data according to Parity-bit controls from Shift Register Function through TX line with a frequency of Baud Rate (shifting an active edge of BTU signal). The data transmission transmits LSB to MSB with or without parity bit. At the beginning of a transmission, High represents mark (nothing) and Low represents Start Bit, then the package data will be shifted.

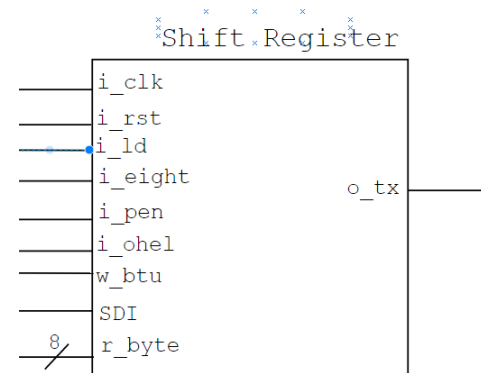


Figure 16: Shift Register Module

Block Diagram

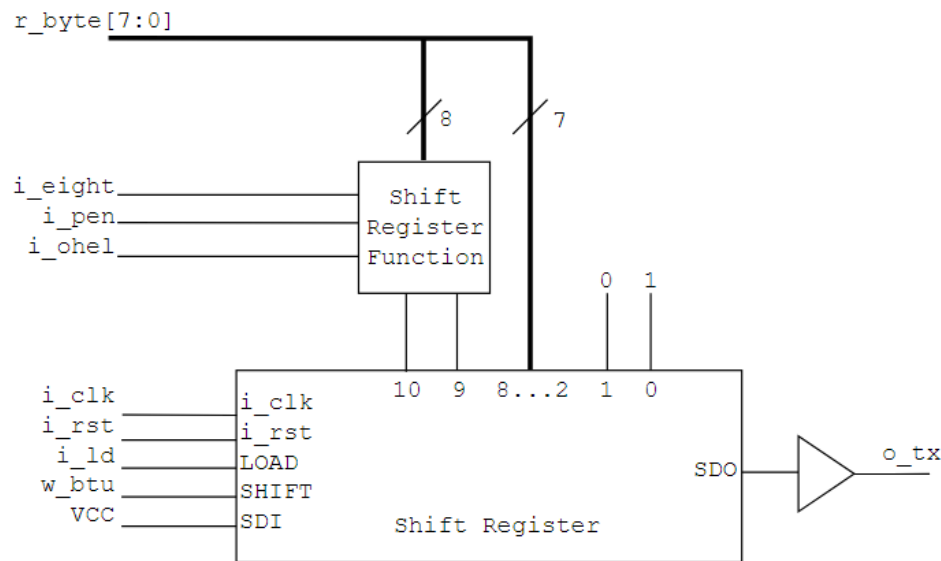


Figure 17: Shift Register Detail Block Diagram

Chip Specification

25 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

Eight	Pen	Ohel	10 th bit	9 th bit
0	0	0	1	1
0	0	1	1	1
0	1	0	1	^BYTE[6:0]
0	1	1	1	~^BYTE[6:0]
1	0	0	1	BYTE[7]
1	0	1	1	BYTE[7]
1	1	0	^BYTE[7:0]	BYTE[7]
1	1	1	~^BYTE[7:0]	BYTE[7]

Table 12: Shift Register Encoding table

I/O

Signal	Size (bit)	I/O	Connected to
i_clk	1	I	100 MHz Crystal Oscillator
i_rst	1	I	AISO reset
r_ld_d1	1	I	Load D1 flop
r_byte	8	I	TramelBlaze OutPort[7:0]
i_eight	1	I	On-board Switch[3]
i_pen	1	I	On-board Switch[2]
i_ohel	1	I	On-board Switch[1]
w_btu	1	I	Bit Time Counter
SDI	1	I	Hardwire high(1'b1)
SDO(o_tx)	1	O	UART TX output

Table 13: Shift Register I/O

Source Code: Appendix F & Appendix G

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

Verification

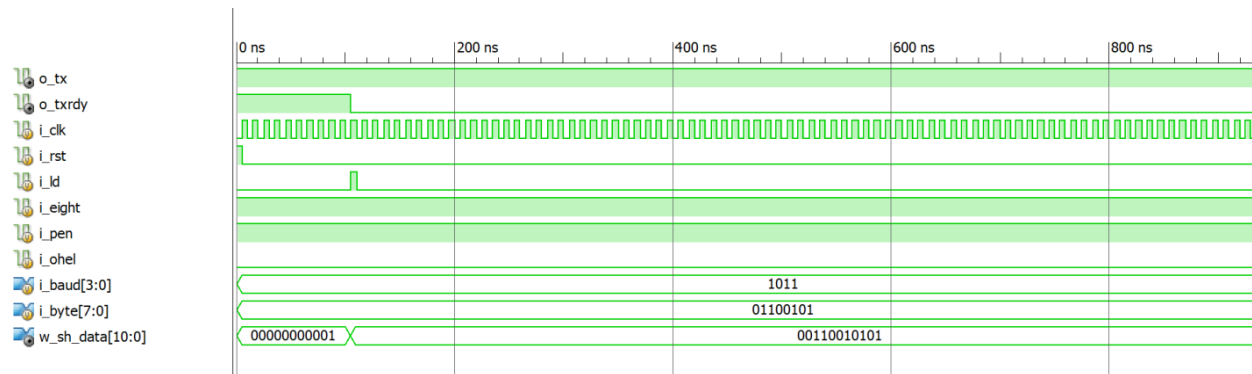


Figure 18: Shift Register Verification

Shift Register is verified since an input data from TramelBlaze(i_byte) has be encoded into 11-bit data according the parity bit control, and then loaded into the Shift Register before shifting LSB to MSB with an even parity bit as a requirement.

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

6.2.2 Receive Engine

Description

UART Receiving Engine module responses for obtaining data from RX_line of UART and output the received data to TramelBlaze while checking the data error. There are major components such as RX_Control Unit and RX_Datapath allow UART_RX synchronously receiving data from another device through the RX_line. RX_Control Unit generates controls for RX_Datapath by utilizing Bit Time Counter, Bit Counter, and a Finite State Machine.

Once, RX_Control Unit detects High-to-low transition for half of a bit time period, it recognizes the start-bit of the data and the Datapath starts receiving data in its shifting register, and then informs the Datapath when the stop bit arrives.

Meanwhile, RX_Datapath receives data, re-mapping the data, and check the frame of data if there is any kind of data transmitting error according to the controls.

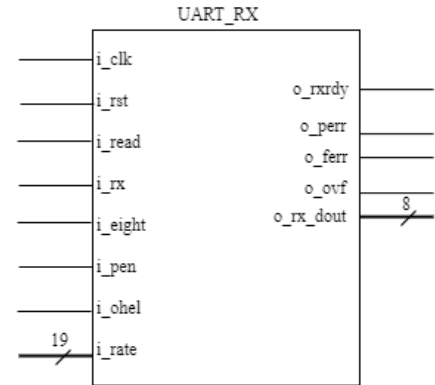


Figure 19: UART_RX block diagram

Detail Block Diagram

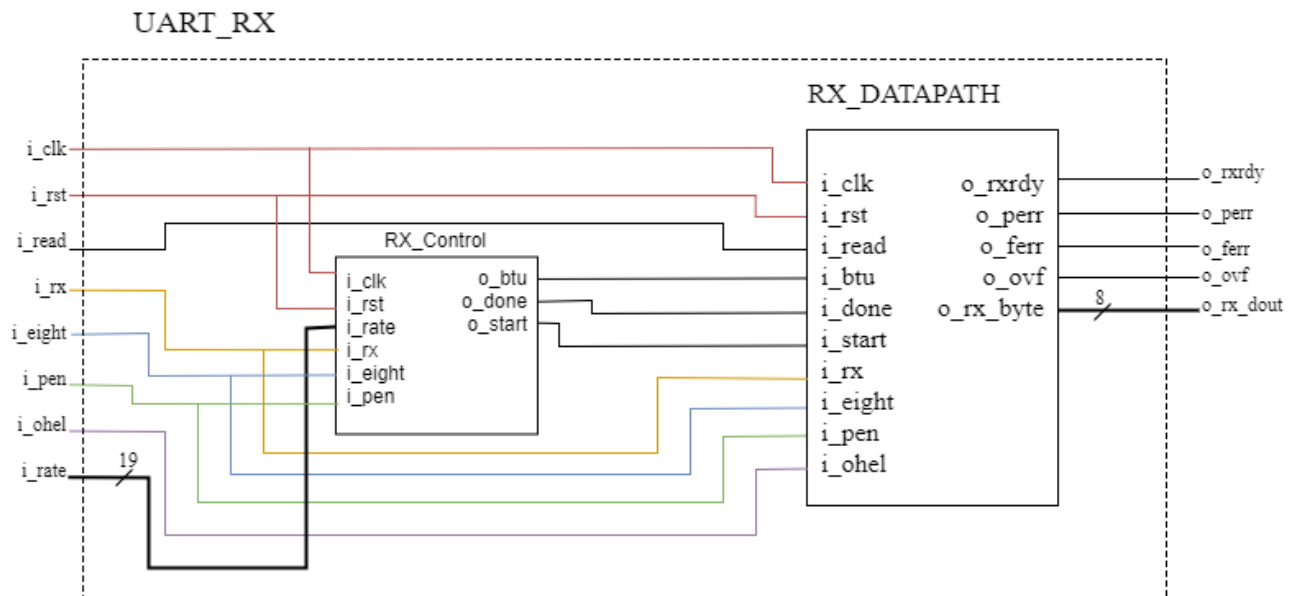


Figure 20: UART_RX Detail Block Diagram

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

I/O

Signal	Size (bit)	I/O	Connected to
i_clk	1	I	100 MHz Crystal Oscillator
i_rst	1	I	AISO reset
i_read	1	I	Address Decoder
i_rate	19	I	Baud Rate Decoder
i_rx	1	I	RX_line
i_eight	1	I	On-board Switch[3]
i_pen	1	I	On-board Switch[2]
i_ohel	1	I	On-board Switch[1]
o_rxdy	1	O	UART
o_perr	1	O	UART
o_ferr	1	O	UART
o_ovf	1	O	UART
o_rx_dout	8	O	UART

Table 14: UART_RX I/O

Source Code: Appendix H

Verification

Utilizing a complete Transmit engine simulation transmits 0hAE as shown in [Figure 22](#) and writes the TX output to a text file, then generating the text file as stimulus vector for UART_RX input (i_rx). With the identical control setup, Eight bit, Parity bit enable, and Even Parity bit, the transmit data are received in RX_Shift_Register and passing forward to RX_Remap module. The Remap has an 8-bit output (0hAE) through o_rx_dout to TramelBlaze as shown below which verifies that the UART_RX has received correct data (Figure 21: UART_RX Verification Part 1) as they were transmitted from UART_TX (Figure 22: UART_RX Verification Part 2).

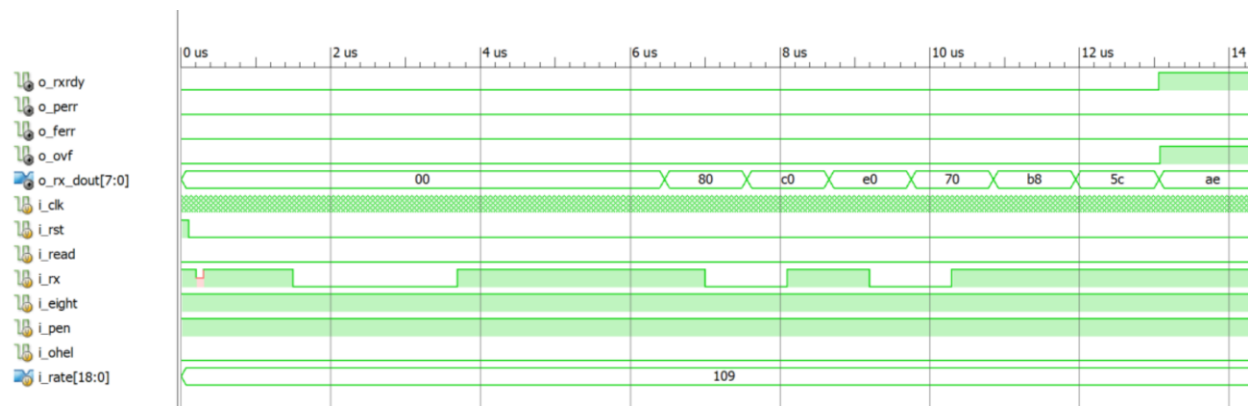


Figure 21: UART_RX Verification Part 1

Chip Specification

29 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

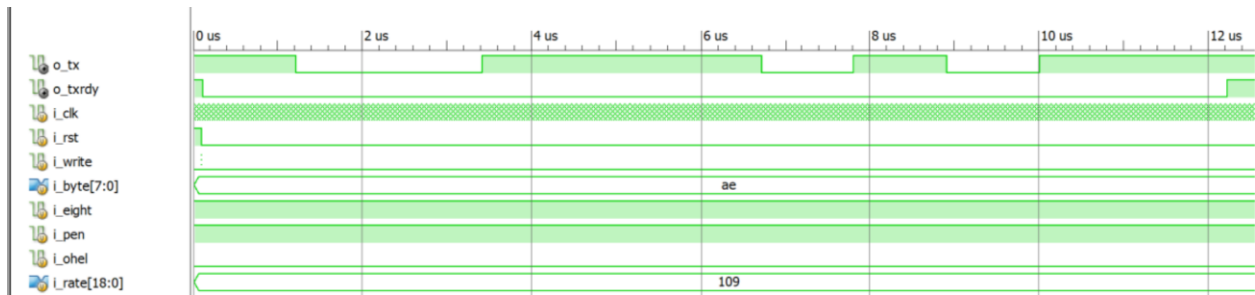


Figure 22: UART_RX Verification Part 2

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

6.2.2.1 RX_Control

Description

Receive Engine Control Unit, a unit generates controls signal to registers and multiplexers in RX_Datapath according to the control inputs. This module consists of three major modules; Bit Time Counter, Bit Counter, and Finite State Machine.

Bit Time Counter is similar to the one in Transmit Engine, while this RX Bit Time Counter selects either half of bit time counting or full bit time counting according to the state of the RX Engine. If the engine is at START state, the bit time is half since the State machine is consistently checking if the start bit holds for half of the bit time, then the bit time is valid. Meanwhile, counting full bit time in other states to synchronize the baud rate of the UART transmission.

Bit Counter counts numbers of arriving frame of data according to Eight and Parity bit controls.

Finite State Machine, a state machine consistently checking the receiving data High-to-Low transition since the start bit is the beginning of the transmitting data and the incoming data would be mark(High) and the start bit(Low). Then, keeps checking if the start bit hold in low active for a half of bit rate time, and then the transition kicks in and the RX engine readings the data until Done flags from Bit Count sets.

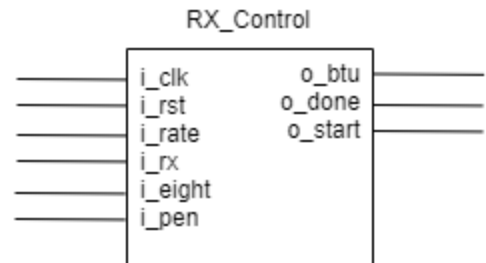


Table 15: RX_Control Block Diagram

Finite State Machine

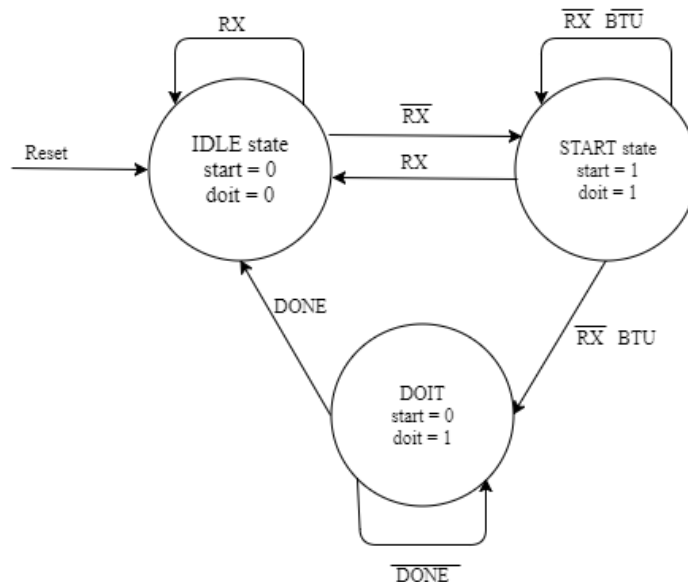


Figure 23: RX_Control_State_Machine Diagram

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

I/O

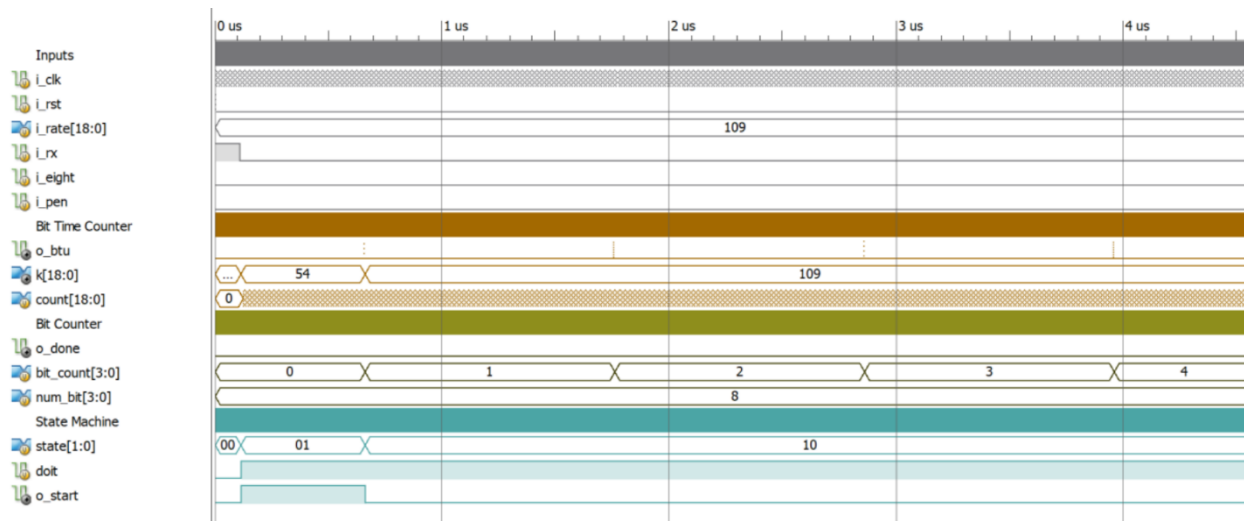
Signal	Size (bit)	I/O	Connected to
i_clk	1	I	100 MHz Crystal Oscillator
i_rst	1	I	AISO reset
i_rate	19	I	Baud Rate Decoder
i_rx	1	I	RX_line
i_eight	1	I	On-board Switch[3]
i_pen	1	I	On-board Switch[2]
i_ohel	1	I	On-board Switch[1]
o_btu	1	O	RX_Datapath
o_done	1	O	RX_Datapath
o_start	1	O	RX_Datapath

Table 16: RX_Control I/O

Source Code: Appendix I

Verification

RX_Control is verified as its Bit Time Counter counts half of baud rate (i_rate) when the State Machine is at START State (2'b01) and then counts full baud rate at other states as shown in part 1, and then generates bit time up(btu) as the counter finishes counting a bit-time period. Moreover, RX_Control is also verified as its Bit Counter generates o_done signal once it finishes counting number of o_btu that Bit Time Counter generated as the Bit Counter is set by Eight and Parity Bit control. Meanwhile, the Finite State Machine cycles through states once it detects the High-to-Low transition of i_rx, generates o_start when it is at START state, and generates o_doit while it is not at IDLE state.



Chip Specification

32 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

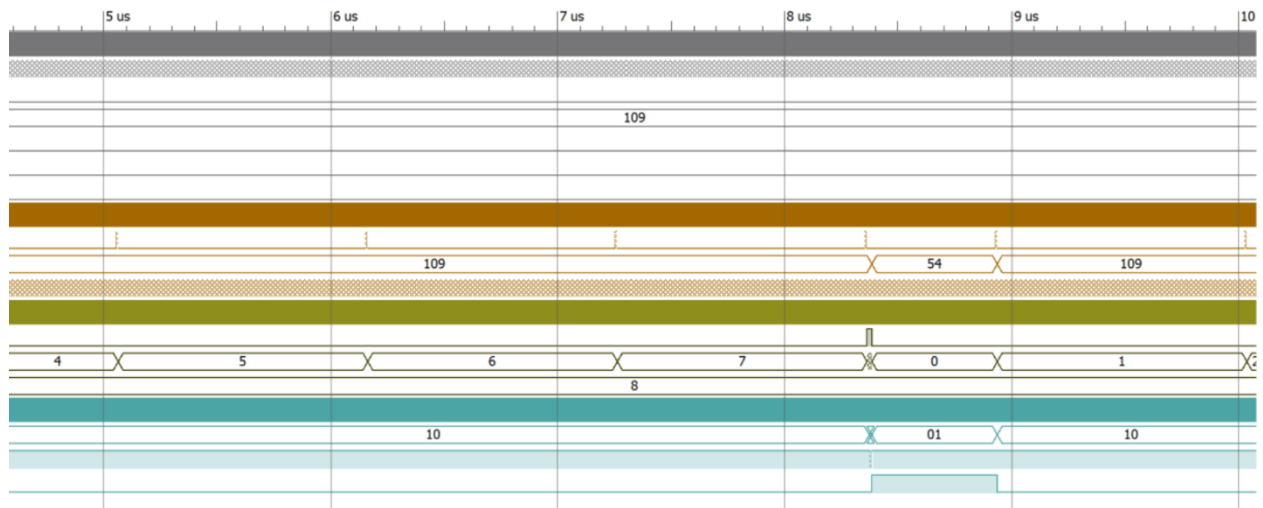


Figure 25: RX_Control Verification Part2

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

6.2.2.2 RX_Datapath

Description

Receive Engine Datapath is a module responsible for receiving a frame of bits input from UART RX line to receive engine shifting register (RX_SHIFT_REG) until the arrival of the stop bit of the data frame, then re-mapping the input data according to the data control before passing the remapped data through a 10-bit bus (w_map_data). Eight lower bit data will be directly sent as data output. While receiving data in through RX_line, Datapath also validifies if the incoming data is legit according data controls. There are three types of error; Parity-bit Error, Framing Error, and Overflow. Moreover, Datapath flags o_rxrdy if the input has been completely received.

Parity-bit Error is flagged when the parity bit in the data frame is mismatched with the parity-bit set by the input control, for example, 8E1 of 0xAA has a high parity bit instead of low bit showing that the parity bit is incorrected.

Framing Error is flagged when the stop bit in the data frame is mismatched with the input control, for example, 7N1 has a stop bit at o_rx_byte[8] where it is supposed to be o_rx_byte[7] showing that the stop bit is mismatched the controls.

Overflow error is flagged when received data is finished while the receive engine is not busy. In other words, there are actually more data coming in when it should have stopped reading data.

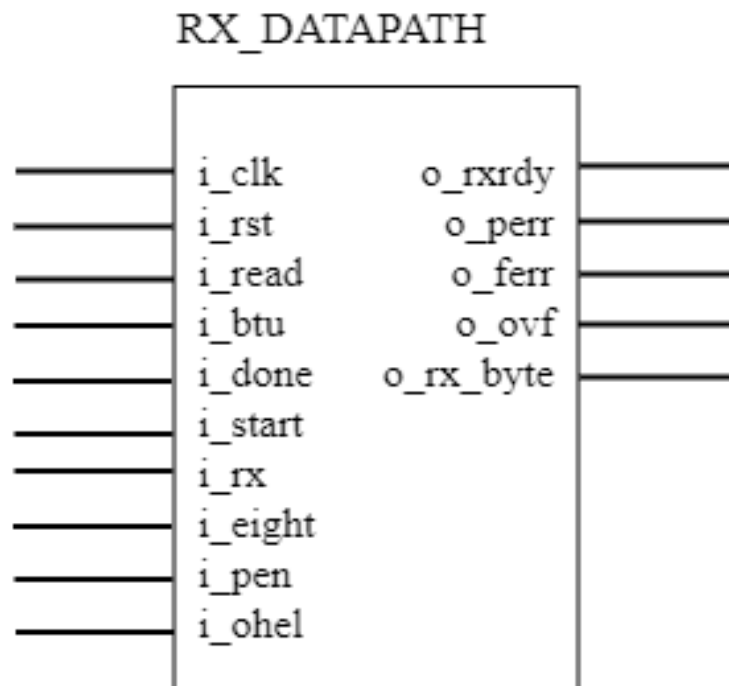


Figure 26: RX_Datapath Block Diagram

Chip Specification

34 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

Detail Block Diagram

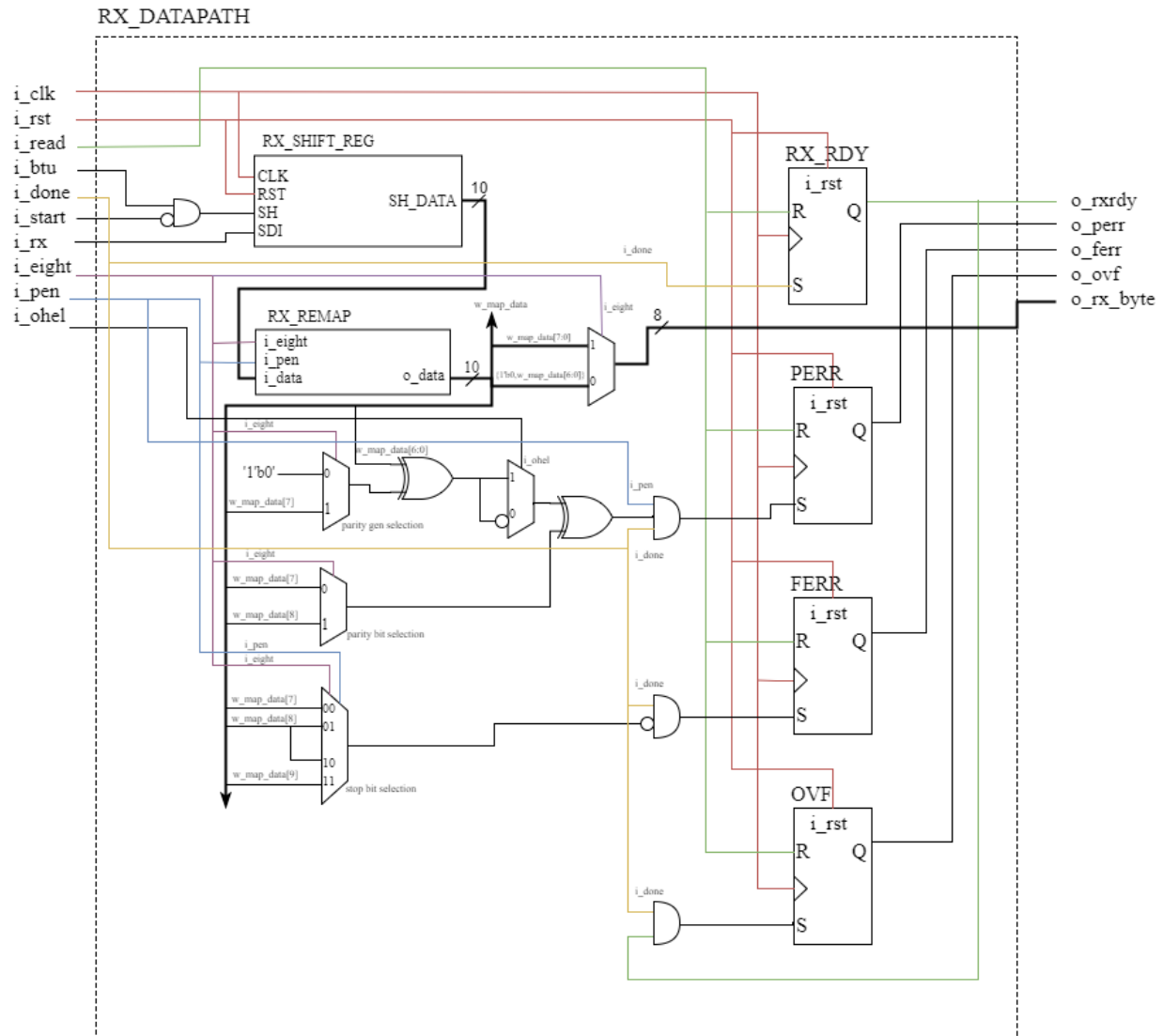


Figure 27: RX_Datapath Detail Block Diagram

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

I/O

Signal	Size (bit)	I/O	Connected to
i_clk	1	I	100 MHz Crystal Oscillator
i_rst	1	I	AISO reset
i_read	1	I	Read[0] from Address Decoder
i_btu	1	I	RX_Control
i_done	1	I	RX_Control
i_start	1	I	RX_Control
i_rx	1	I	RX_line
i_eight	1	I	On-board Switch[3]
i_pen	1	I	On-board Switch[2]
i_ohel	1	I	On-board Switch[1]
o_rxdy	1	O	UART Interrupt
o_perr	1	O	LEDs
o_ferr	1	O	LEDs
o_ovf	1	O	LEDs
o_rx_byte	8	O	TramelBlaze

Table 17: RX_Datapath I/O

Source Code: Appendix J

Verification

• RX Ready signal

The signal is verified as o_rxdy is set once i_done is asserted until RX_Datapath receives an i_read signal, then it clears o_rxdy signal. When i_read is reasserted while i_done is active, o_rxdy is set as shown below.

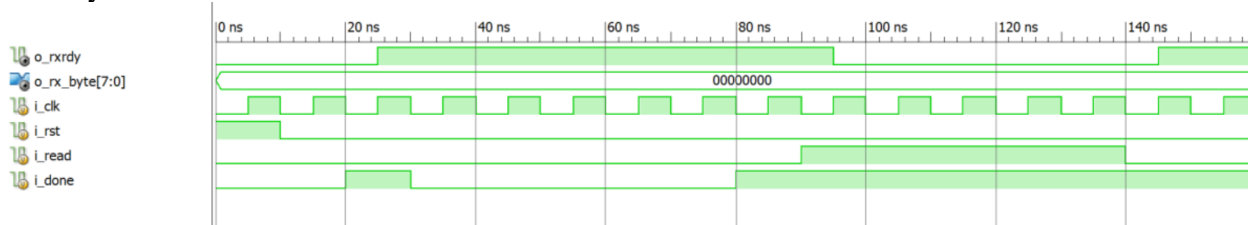


Figure 28: RX_Datapath o_rxdy Verification

• Parity Bit Error signal

Parity Bit Error is verified as it is able to check the parity bit correctly as the stimulus remapped data switched. For this scenario, the remapped data is 0b1[1]_0110_1100 with 8E1 control input. Since 8-bit data is 0b0110_1100, therefor even parity bit should be low, but the remapped data received [1] (high) which is incorrect, thus the o_perr occurs HIGH for some period of time until the perr register is cleared by i_read signal.

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

Then the remapped data is changed to 0b1[0]_0110_1100. As discussed above, since this remapped data has a correct response parity bit, therefore o_perr does not flag.

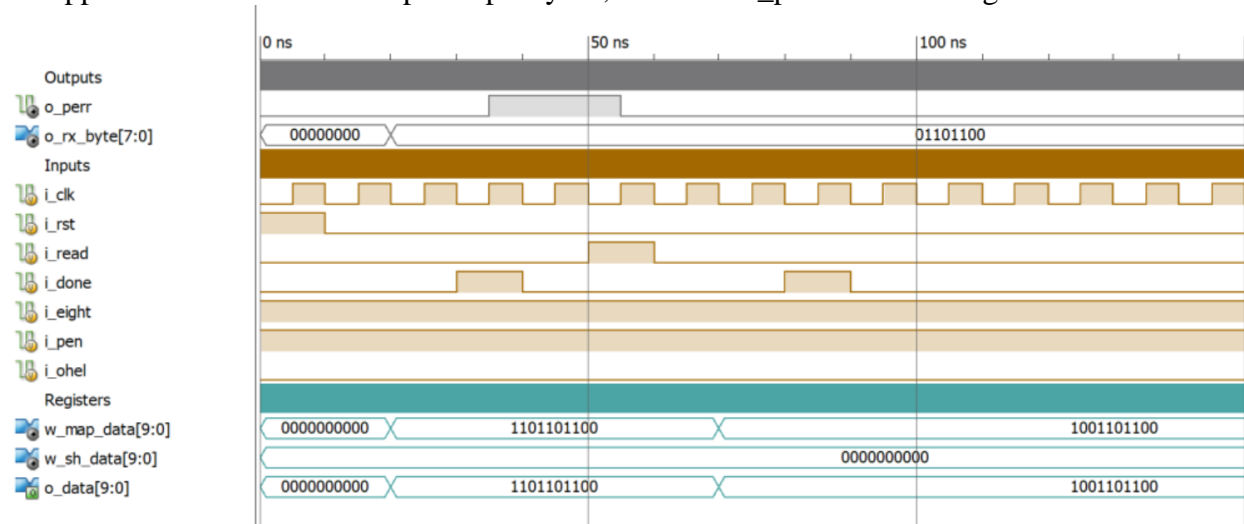


Figure 29: RX_Datapath o_perr Verification

• Framing Error signal

Framing Error detects incoming data if the stop bit of the incoming data frame is corrected. First, the remapped data is 0b[0]001101100 with Eight and Parity bit controls, therefore the stop bit is at the MSB, but the remapped data's stop bit is reasserted, thus o_ferr is set as the remapped data has an incorrected stop bit. After that, clearing the flag with i_read signal and switching the remapped data to 0b[1]001101100 with the same Eight and Parity bit controls which does not set the o_ferr flag as the stop bit is correct.

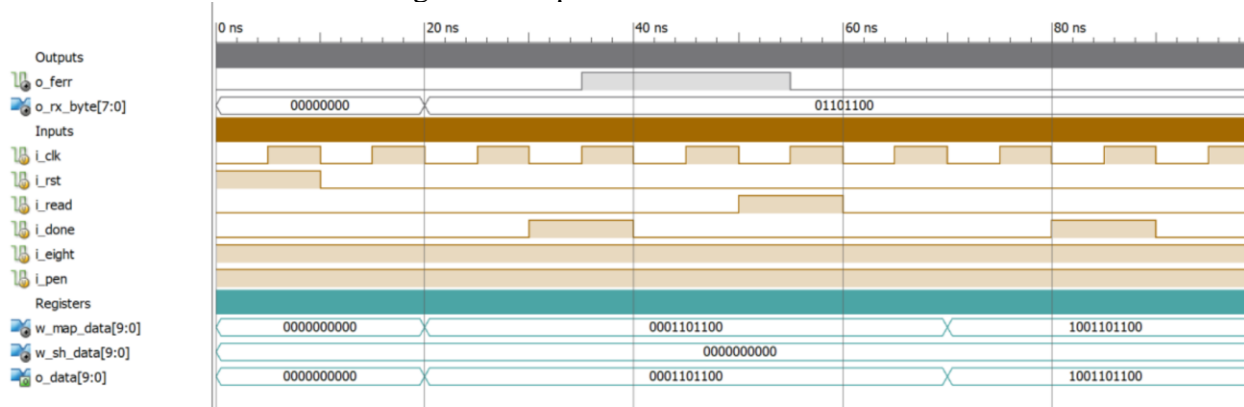


Figure 30: RX_Datapath o_ferr Verification

• Overflowing Error signal

Overflow flag is verified as it is flagged when o_done and o_rxdy are both asserted, and the flag is cleared by i_read signal properly.

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

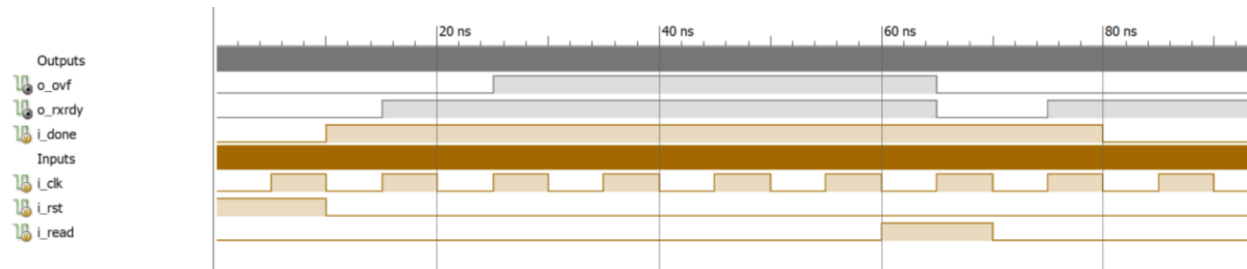


Figure 31: RX_Datapath o_ovf Verification

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

6.2.2.2.1 RX_Shift_Register

Description

A register shifting an incoming data bit from RX_line (i_rx) every Bit Time Up (i_btu) in the DOIT state from RX_Control unit, meaning the shift register shifts in a bit after the start condition bit. And then, stops shifting in data in IDLE and START states. Shift Register is also called FIFO register as it takes incoming input on an active edge of a signal, in this case, from RX_Control's bit time counter.

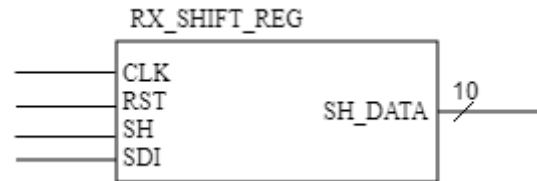


Figure 32: RX_Shift_Register

I/O

Signal	Size (bit)	I/O	Connected to
CLK	1	I	100 MHz Crystal Oscillator
RST	1	I	AISO reset
SH	1	I	RX_Control
SDI	1	I	UART_RX
SH_DATA	10	O	RX_REMAP

Table 18: RX_Shift_Register I/O

Source Code: Appendix L

Verification

RX_Shift_Register is verified as input data have been shifting from MSB to LSB when Shifting signal (SH) is HIGH, allowed the register to shift the input and update the output (SH_DATA) as shown in a figure below.

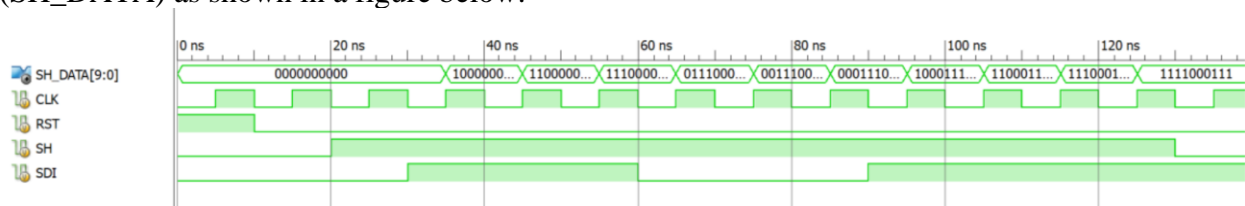


Figure 33: RX_Shift_Register Verification

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

6.2.2.2.2 RX_Remap

Description

A combinational block to re-arrange shifted data from RX_Shift_Register in manageable order or in the correct place according to the data controls, and to ensure the data frame is ready to be ready for errors checking.

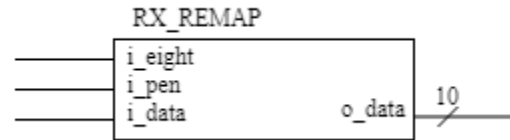


Figure 34: RX_Remap Diagram

I/O

Signal	Size (bit)	I/O	Connected to
i_eight	1	I	On-board Switch[3]
i_pen	1	I	On-board Switch[2]
i_data	10	I	RX_Shift_Register
o_data	10	O	UART_Datapath and TramelBlaze

Table 19: RX_REMAP I/O

Source Code: Appendix K

Verification

RX_Remap waveform below simulates how a 10-bit shifted data from RX_Shift_Register behaves with different data controls (i_eight and i_pen).

When i_eight and i_pen are reasserted, i_data[9:2] are shifted to the right and added two mark bits on the left.

When only i_eight or only i_pen is asserted, i_data[9:1] are shifted to the right and add one mark bit on the left.

And When both i_eight and i_pen is asserted, the i_data is actually place on the correct place.

The waveform below verifies that RX_Remap behaves as expected.

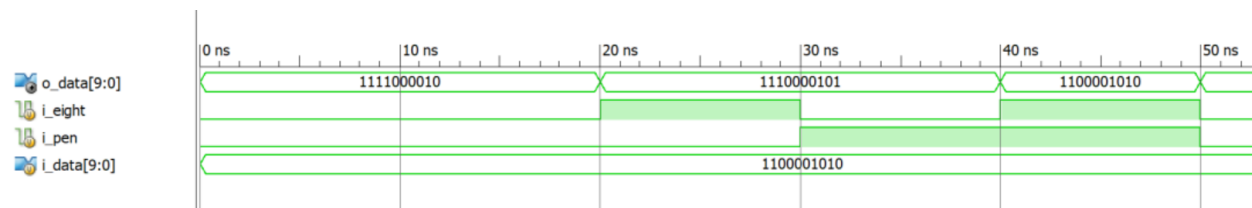


Figure 35: RX_Remap Verification

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

6.3 Address Decoder

Description

A combination block decodes port ID from the TramelBlaze along with its read or write strobe into two 16-bit outputs; reads and writes. When either read or write strobe high, the bit position represented the current port ID would turn high. For instance, if the current port ID is 16'b0001 and write strobe is high, then writes == 16'b0010 would turn high. Likewise, if the current port ID is 16'b0010 and read strobe is high, then reads == 16'b0100.

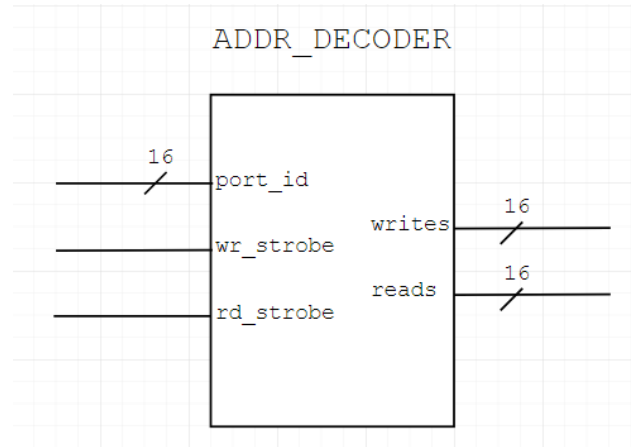


Figure 36: Address Decoder Module

I/O

Signal	Size (bit)	I/O	Connected to
port_id	16	I	TramelBlaze
wr_strobe	1	I	TramelBlaze
rd_strobe	1	I	TramelBlaze
writes	16	O	Load Signal
reads	16	O	**not connected in revision1**

Table 20: Address Decoder I/O

Source Code: [Appendix N](#)

Verification

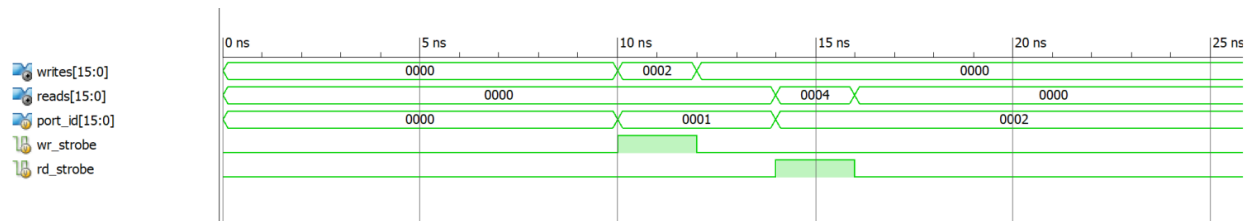


Figure 37: Address Decoder Verification

Address Decoder is verified as it outputs writes 16'h0002 when port_id is 16'h0001 and write strobe is high, and outputs reads 16'h0004 when port_id is 16'h0002 and read strobe is high.

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

6.4 Positive Edge Detector (PED)

Description

A pulse maker detecting a raising edge of a signal and output one-clock-period signal (one shot pulse signal) to a destination device. Typically used for indicating a first sight of a signal change from low to multiple clock high.

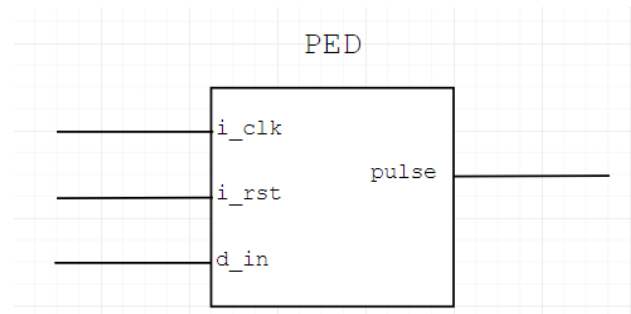


Figure 38: PED Module

I/O

Signal	Size (bit)	I/O	Connected to
i_clk	1	I	100 MHz Crystal Oscillator
i_rst	1	I	AISO reset
d_in	1	I	Txrdy Flop
pulse	1	O	Interrupt Request RS Flop

Table 21: PED I/O

Source Code: [Appendix H](#)

Verification

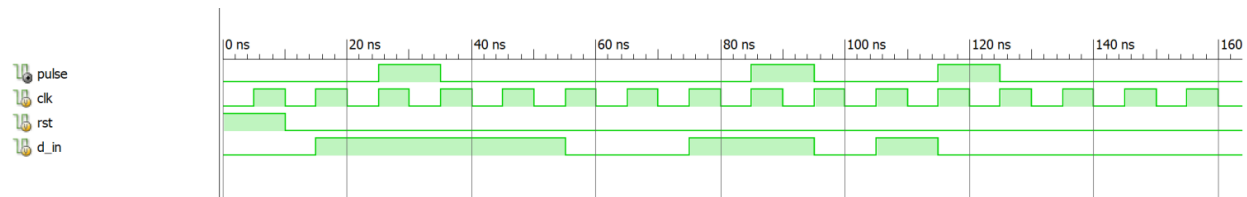


Figure 39: PED Verification

PED is verified as it generates a one-shot pulse for the next raising edge of the system clock after it detected the input signal had changed from low to high.

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

6.5 Asynchronous-In-Synchronous-Out Reset (AISO Reset)

Description

A device to synchronous a digital circuit design's system reset signal to occur on all components at the same edge of the clock instead of asynchronous reset signal that could make the system behave differently than expected.

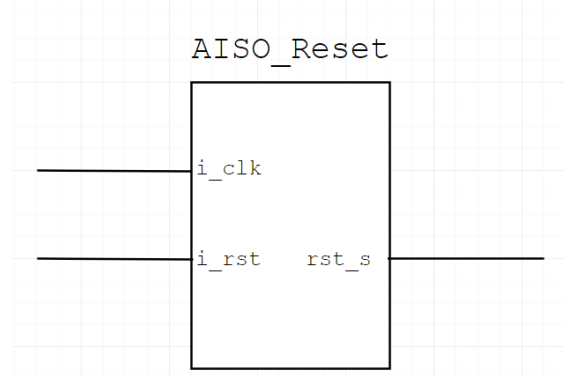


Figure 40: AISO Reset module

I/O

Signal	Size (bit)	I/O	Connected to
i_clk	1	I	100 MHz Crystal Oscillator
i_rst	1	I	AISO reset
rst_s	1	O	All sequential block in the design

Table 22: AISO I/O

Source Code: [Appendix O](#)

Verification

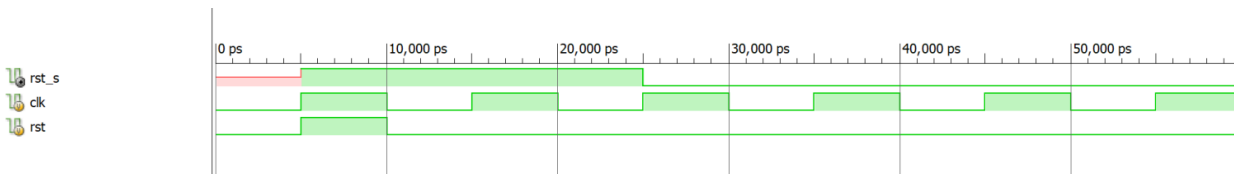


Figure 41: AISO verification

AISO_reset is verified as it generates a synchronous reset output when it receives asynchronous input.

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

7 Chip Level Verification

7.1 Transmit Engine

Transmit Engine is to consistently and serially shift a 16-bit data received from the TramelBlaze after a load signal by encoding the data to a package according to the parity-bit control that decides how is the data package should be. Generally, the data should starts with 01 where 1 is mark (idle state) and 0 is for the start bit to indicate the destination UART device that the transmission is about to begin, followed by 7 or 8 bit data, parity bit or not parity bit, and then filled the rest with 1's for a stop bit and mark bit as shown in Figure 42.

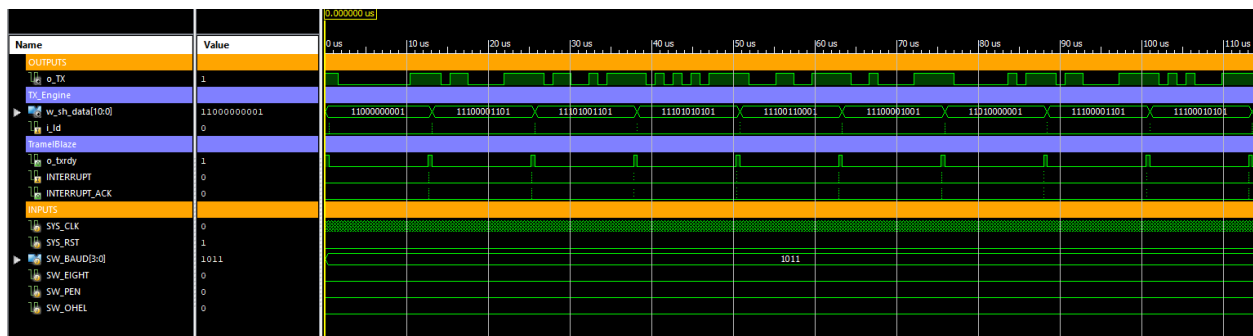


Figure 42: TX Engine with TramelBlaze Verification

As expected, the result from Figure 42 simulates that each data received from TramelBlaze are consistently encoded and serially shifted through TX line in the fashion of parity-bit control. Therefore, the output from TX line to an UART receiver with the similar set of parity-bit, the outputs at the receiver should be exactly as they are shifted from the Transmit Engine.

In order to inspect the result from the Transmit Engine, the Nexys3 board's TX port has to be connected to the RX port of the receiver which, in this case is a micro USB cable to another USB port in a computer with RealTerm program. RealTerm is a terminal could represent as a UART receiver which is suit for displaying a received data from Nexy3 board. After setting the incoming USB port, Baud rate at 330 Hz, 7-bit data, no-parity bit, and one stop bit, then initialize the Nexy3 with .bit file and the result is shown as in Figure 43.

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

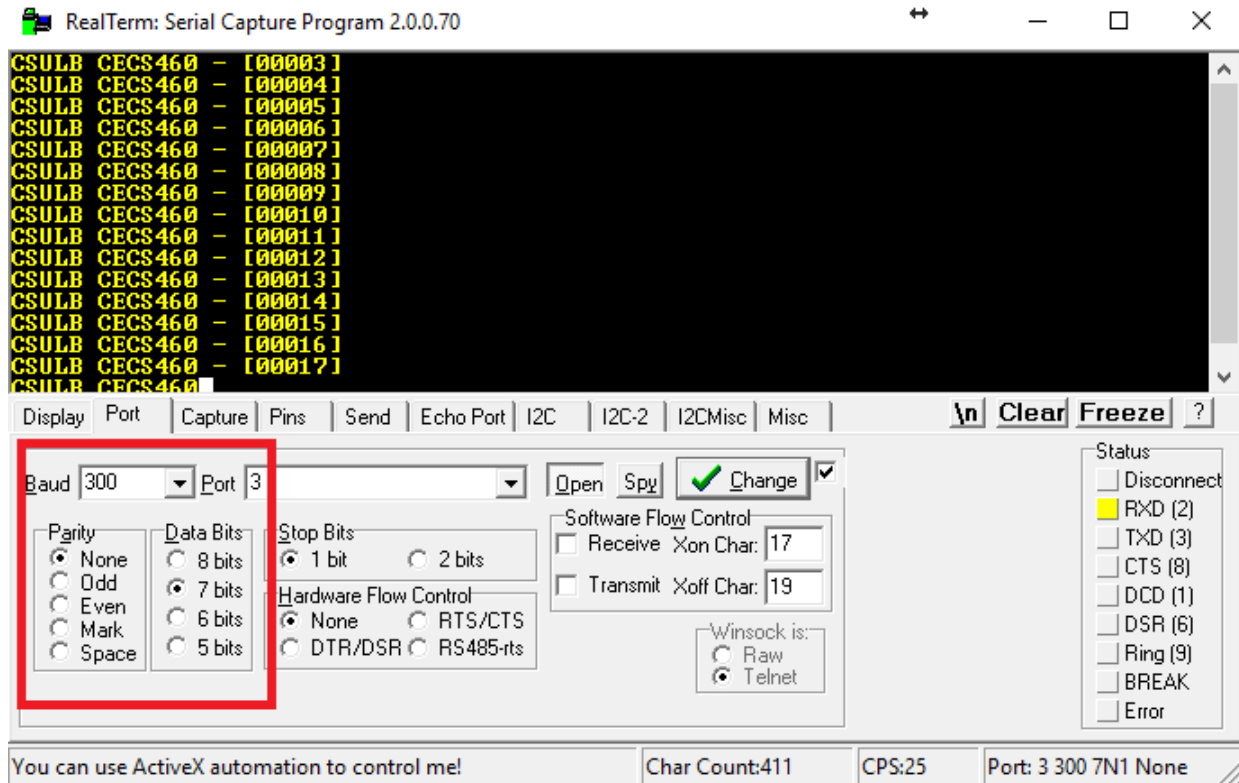


Figure 43: Transmitted result on RealTerm

The result is shown as expected. The string of “CSULB CECS460 – [number]<CR><LF>” has been initialized in the ScratchPad Memory of the TramelBlaze, and it is being fetch one character at the time and transmitted to RealTerm. As long as the agreement of UART protocol for Transmit Engine and the Receiver are match, the transmitted data should be displayed correctly as shown in Figure 43, thus this result verifies this Transmit Engine is functioning properly.

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

7.2 Receive Engine

Receive Engine Verification is utilized a complete transmission of the Transmit Engine to generate its outputs with a certain UART-bit controls (eight-bit, parity enable, and odd/even parity) into a text file, and then the Receive Engine will use the text file as a stimulus vector. In the verification, the bit controls are seven-bit and parity disable (7N1) to transmit 8h'3A into the Receive Engine shown in figures below.

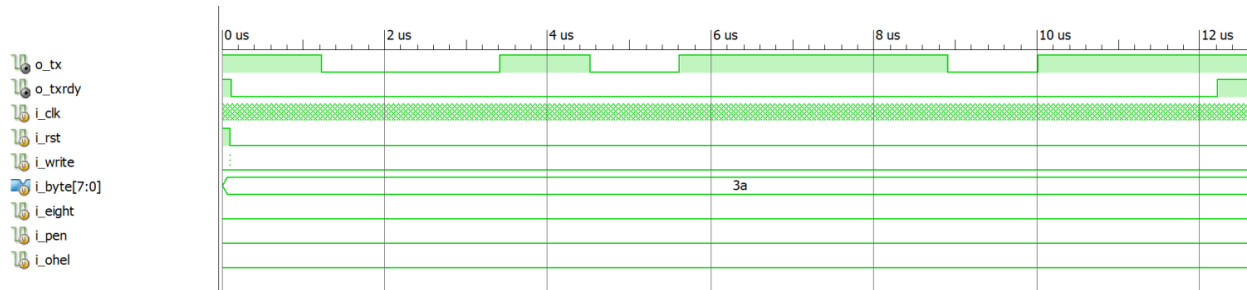


Figure 45: UART_TX transmits 8'h3A to a file.

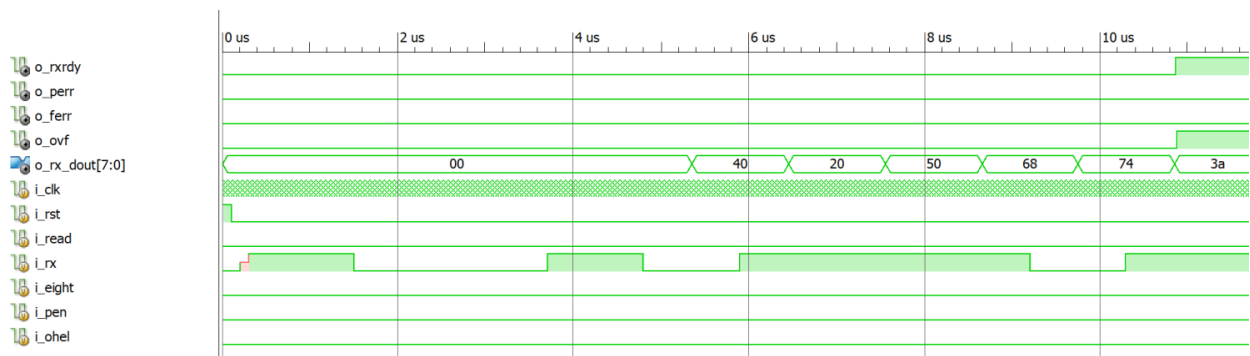


Figure 44: UART_RX receives 8'h3A from the file

After stimulated and verified the RX Engine read the correct data which would be passed to TramelBlaze for further processes, both UART_TX and UART_RX are connected as a unit called UART which will transmit and receive data to and from a USB-microUSB connection to a Serial Terminal to interact with a user.

Therefore, a program has been created and install to the TramelBlaze's ROM to fulfill the project requirement and test if the UART functions properly with various controls and input.

The program is to display a banner and prompt at the start and wait for a user's input. If the input is <cr> or Enter, the cursor in the terminal would be at the start of next line and should refresh the prompt. If the input is <bs>, the character in front of the cursor should be erased, but not to erase the prompt. If the input is '*', displays my hometown and followed by a prompt in a new line. If the input is '@', displays number of character received followed by a prompt in a new line. Otherwise, the terminal should echo input characters up to 40 characters. If 40th character is reached, display a prompt in a new line as shown in the figure below.

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

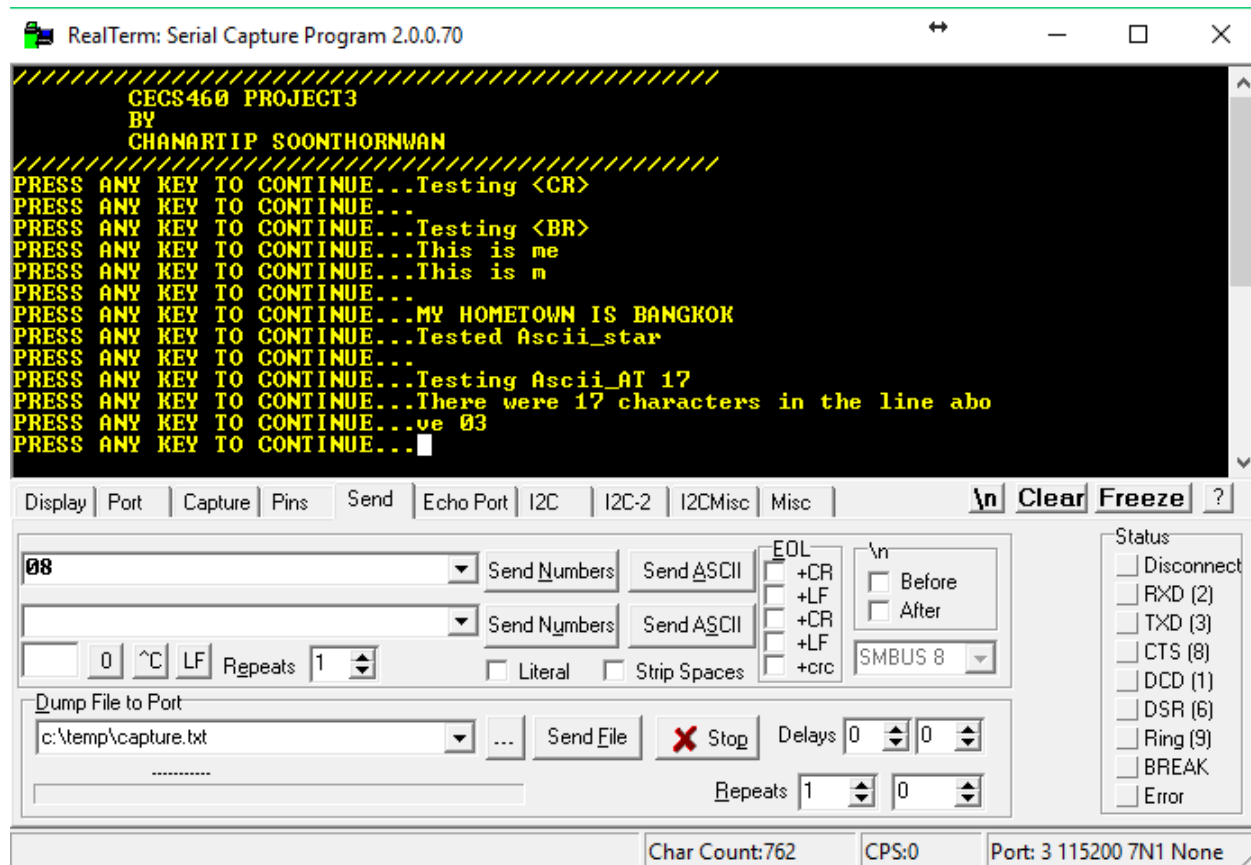


Figure 46: Full UART with TramelBlaze on SerialTerminal

<CR> was demonstrate by pressing an Enter button on a keyboard to get the new line with a prompt. <BS> is a bit difficult to display in one figure, but “This is me” has been erased the last character shown on the next line. Then “*” display my hometown, before testing the echoing character above 40 characters and displaying the number of received character with ‘@’ pressed.

Chip Specification

47 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

8 Chip Level Test

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

9 Appendix

The Appendix includes Verilog codes for every module, except the TramelBlaze, that has been mentioned in this Chip specification.

Appendix A SOPC Code

```

`timescale 1ns / 1ps
//*****
// This document contains information proprietary to the
// CSULB student that created the file - any reuse without
// adequate approval and documentation is prohibited
//
// Class: CECS460 System on Chip Design
// Project name: Project3_UART_RX
// File name: SOPC_Core.v
//
// Created by Chanartip Soonthornwan on March 25, 2018.
// Copyright © 2018 Chanartip Soonthornwan. All rights reserved.
//
// Abstract: Overview of the project shows instantiations of
// UART, TramelBlaze, and other developed modules
//
// Version 1.0: Date: March 25, 2018.
//
//*****
module SOPC_Core(
    input SYS_CLK, // System Clock
    input SYS_RST, // System Reset
    input [3:0] SW_BAUD, // Baud Rate Switches input
    input SW_EIGHT, // Eight Bit Control
    input SW_PEN, // Parity Bit Control
    input SW_OHEL, // Odd/Even Parity Bit Control
    input i_RX, // Receive Line Input
    output reg [7:0] o_LED, // On-Board LEDs
    output o_TX // Transmit Line Output
);

// Interconnection wires
wire w_rst_s;
wire w_ld;
wire w_intr;
wire w_int_ack;
wire [15:0] w_tb_outport;
wire [15:0] w_tb_port_id;
wire w_wr_strobe;
wire w_rd_strobe;
wire [15:0] w_writes;
wire [15:0] w_reads;
wire w_uart_intr;
wire [7:0] w_uart_ds;

// Assigning LEDs on board
always@(posedge SYS_CLK, posedge w_rst_s)
    if(w_rst_s) o_LED <= 8'b0; else
        if(w_writes[2]) o_LED <= w_tb_outport[7:0];
        else o_LED <= o_LED;

// Load Signal for UART_TX
assign w_ld = (w_tb_port_id == 16'b0) & w_writes[0];

```


Chip Specification

49 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

```
// UART Engine
UART_TOP UART_inst(
    .i_clk(SYS_CLK),
    .i_rst(w_rst_s),
    .i_baud(SW_BAUD),
    .i_write(w_ld),
    .i_read(w_reads[1:0]),
    .i_byte(w_tb_outport[7:0]),
    .i_rx(i_RX),
    .i_eight(SW_EIGHT),
    .i_pen(SW_PEN),
    .i_ohel(SW_OHEL),
    .o_TX(o_TX),
    .o_UART_INTR(w_uart_intr),
    .o_UART_DS(w_uart_ds)
);

// RS flop to hold the ped pulse till reset
// by TB's interrupt acknowledge
RS_FLOP rs_intr_inst
(.CLK(SYS_CLK),
 .RST(w_rst_s),
 .R(w_int_ack),
 .S(w_uart_intr),
 .Q(w_intr)
);

// TramelBlaze (TB)
tramelblaze_top tb_top
(.CLK(SYS_CLK),
 .RESET(w_rst_s),
 .IN_PORT({8'b0, w_uart_ds}),
 .INTERRUPT(w_intr),
 .OUT_PORT(w_tb_outport),
 .PORT_ID(w_tb_port_id),
 .READ_STROBE(w_rd_strobe),
 .WRITE_STROBE(w_wr_strobe),
 .INTERRUPT_ACK(w_int_ack)
);

// Address Decoder
ADDR_DECODER addr_dec_inst
(.port_id(w_tb_port_id),
 .wr_strobe(w_wr_strobe),
 .rd_strobe(w_rd_strobe),
 .writes(w_writes),
 .reads(w_reads)
);

// Asynchronized Input, Synchronized Output
AISO_RST aiso_inst
(.clk(SYS_CLK),
 .rst(SYS_RST),
 .rst_s(w_rst_s)
);
```

endmodule

Chip Specification

50 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

Appendix B UART Transmit Engine Source Code

```
`timescale 1ns / 1ps
//*****//
// This document contains information proprietary to the //
// CSULB student that created the file - any reuse without //
// adequate approval and documentation is prohibited //
// //
// Class: CECS460 System on Chip Design //
// Project name: Project3_UART_RX //
// File name: UART_TOP.v //
// //
// Created by Chanartip Soonthornwan on March 1, 2018. //
// Copyright © 2018 Chanartip Soonthornwan. All rights reserved. //
// //
// Abstract: A device Receiving and Transmitting data through //
// UART Protocol, without using a common clock between //
// this and destinated device, using number of bits //
// in a frame of data to confirm the correctness //
// off data collection. //
// //
// Version 1.0: Date: March 1, 2018 //
// Version 1.1: Date: March 18, 2018. //
// - Moved Baud Decoder from UART_TX.v //
// - Added UART_RX.v //
// Version 1.2: Current Date: March 20, 2018. //
// - Added wires and connections //
// - Added PEDs //
// - Added rx Status/Data mutilplexer //
// //
//*****//
/*
UART TOP consists with
UART_TX - Transmitted Engine that shifts 1-bit data out at a Bit Time,
and generates an interrupt signal(txrdy) indicated it is ready
for the next transmission.
PED - Positive Edge Detector confirms the txrdy signal
RS_FLOP - To hold the PED confirmation signal for a clock until being
reset by TramelBlaze interrupt acknowledge
TramelBlaze - Microprocessor processes instructions from 4Kx16 bit ROM
and outputint the result through OUT_PORT to UART, therefore
UART could transmit the data to a receiver.
ADDR_DEC - Address Decoder decodes reading and writing strobe and sends
them to individual I/O.
AISO_RST - Synchronize System Reset Signal to the whole design.
*/
module UART_TOP(
input i_clk, // System Clock
input i_rst, // System Reset
input [3:0] i_baud, // Baud Rate
input i_write, // Write[0] signal from Address Decoder
input [1:0] i_read, // Read[1:0] signal from Address Decoder
input [7:0] i_byte, // Data to Transmit from TramelBlaze
input i_rx, // Received Data from RX_Line
input i_eight, // Eight bit Control
input i_pen, // Parity bit Control
input i_ohel, // Odd/Even Parity bit Control
output o_TX, // Transmission line (TX_line)
output o_UART_INTR, // UART Interrupt flag to TramelBlaze
output [7:0] o_UART_DS // UART DATA Selected output
```

Chip Specification

51 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

```

);

// Interconnection wires
wire      w_txrdy;
wire      w_rxrdy;
wire      w_ped_tx_pulse;
wire      w_ped_rx_pulse;
wire [18:0] w_rate;
wire      w_perr;
wire      w_ferr;
wire      w_ovf;
wire [7:0] w_rx_dout;
wire [7:0] w_RX_STATUS;

// Data Selecting Mux
// multiplexing RX_status or RX_data to Tramelblaze
// with corrected port_id(i_read[1:0]) requested by the Tramelblaze
assign o_UART_DS = (i_read[1])? w_RX_STATUS :
                    (i_read[0])? w_rx_dout   : 8'b0;

// Concatenating Status
assign w_RX_STATUS = {3'b0, w_ovf, w_ferr, w_perr, w_txrdy, w_rxrdy};

// Interrupt Pulse from either TX or RX Engine
assign o_UART_INTR = w_ped_tx_pulse | w_ped_rx_pulse;

// UART Receive Engine
UART_RX uart_RX_inst
(.i_clk(i_clk),
 .i_rst(i_rst),
 .i_read(i_read[0]),
 .i_rx(i_rx),
 .i_eight(i_eight),
 .i_pen(i_pen),
 .i_ohel(i_ohel),
 .i_rate(w_rate),

 .o_rxrdy(w_rxrdy),
 .o_perr(w_perr),
 .o_ferr(w_ferr),
 .o_ovf(w_ovf),
 .o_rx_dout(w_rx_dout)
);

// UART Transmit Engine
UART_TX uart_TX_inst
(.i_clk(i_clk),
 .i_rst(i_rst),
 .i_write(i_write),
 .i_byte(i_byte),
 .i_eight(i_eight),
 .i_pen(i_pen),
 .i_ohel(i_ohel),
 .i_rate(w_rate),

 .o_tx(o_TX),
 .o_txrdy(w_txrdy)
);

```

Chip Specification

52 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

```
// Positive Edge Detector(PED)
//   Generating one-short pulse of
//   UART Interrupt signal
PED
  ped_txrdy_inst
    (.clk(i_clk),
     .rst(i_rst),
     .d_in(w_txrdy),
     .pulse(w_ped_tx_pulse)
    ),
  ped_rxrdy_inst
    (.clk(i_clk),
     .rst(i_rst),
     .d_in(w_rxrdy),
     .pulse(w_ped_rx_pulse)
    );

// Baud Decoder
//   Decode 4-bit input from switches(i_baud)
//   to 19-bit value(k) for BIT_TIME_COUNTER
BAUD_DEC baud_dec_inst
(
  .BAUD_VAL(i_baud),
  .K(w_rate)
);
```

endmodule

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

Appendix C Transmit Engine Source Code

```
`timescale 1ns / 1ps
//*****//
// This document contains information proprietary to the //
// CSULB student that created the file - any reuse without //
// adequate approval and documentation is prohibited //
// //
// Class: CECS460 System on Chip Design //
// Project name: Project3_UART_RX //
// File name: UART_TX.v //
// //
// Created by Chanartip Soonthornwan on March 18, 2018. //
// Copyright © 2018 Chanartip Soonthornwan. All rights reserved. //
// //
// Abstract: Transmitted Engine for UART to send data to a receiver //
// with specific Baud Rate, parity, and number of bit //
// in the data package(7 or 8 bit) //
// //
// Version 1.0: Date: March 1, 2018 //
// Version 1.1: Current Date: March 18, 2018. //
// - Moved Baud Decoder to UART_TOP.v //
// - Changed 4-bit i_baud port to 19-bit i_rate //
// for previous k (bit time count constant) value //
// - Changed port name i_ld to i_write //
// //
//*****//
/*
UART Transmit Engine
    Trasmit data to UART Receiver at a specific Baud Rate (assigned by i_baud)
    through BAUD_DEC coverting a 19-bit counter for BIT_TIME_COUNTER to
    count the period for each bit transmission, while BIT_COUNTER counts bits
    will be sent in a transmission (11-bit at a time.)

    Everytime BIT_TIME_COUNTER finished count a bit time, Bit Time Up(w_btu)
    will revoke the BIT_COUNTER to start counting the next bit time, and
    SHIFT_R to start shifting the next bit.

    Once all 11-bit has been shifted, BIT_COUNTER set w_done to indicate
    the data package's transmission is finised, and then set o_txrdy
    and reset r_doit for BIT_TIM_COUNTER to reset it's counter for the next
    bit time counting.
*/
module UART_TX(
    input i_clk, // System clock
    input i_rst, // System reset
    input i_write, // Load
    input [ 7:0] i_byte, // 8-bit Data_in
    input i_eight, // 7-bit/8-bit selection (Low 7-bit, High 8-bit)
    input i_pen, // Parity Bit Enable
    input i_ohel, // Odd High / Even Low Parity bit
    input [18:0] i_rate, // Baud Value for selecting Baud Rate

    output o_tx, // Transmit wire
    output reg o_txrdy // High when the transmission is finished.
);
    reg [18:0] count;
    reg [ 3:0] bit_count;
    reg r_ld_dl;
    reg [ 7:0] r_byte;
```

Chip Specification

54 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

```
wire          w_doit;
wire          w_btu;
wire          w_done;
wire [10:0]    w_sh_data;

// BIT_TIME_COUNTER
//   a pulse maker creates a pulse(w_btu)
//   when bit time is up by incrementing
//   a 19-bit counter to reach bit_time value(k)
//   then generates a pulse with one clock period.
assign  w_btu = (count == i_rate);

always@(posedge i_clk, posedge i_rst)
  if(i_rst) count <= 19'b0; else
  if(w_btu) count <= 19'b0; else
  if(w_doit) count <= count + 19'b1;
  else      count <= 19'b0;

// BIT_COUNTER
//   a pulse maker creates a pulse(w_done)
//   when the number of TX bits
//   are transmitted(11 bits), and
//   has a counter(bit_count) to keep tracking
//   of number of bits for each bit time up.
assign  w_done = (bit_count == 4'd11);

always@(posedge i_clk, posedge i_rst)
  if(i_rst) bit_count <= 4'b0; else
  if(w_doit) begin
    if(w_btu) bit_count <= bit_count + 4'b1;
    else      bit_count <= bit_count;
  end
  else      bit_count <= 4'b0;

// RS FLOP instances for TXRDY
//   Set output(o_txrdy) at sys_reset
//   holding the output until the 11-bit
//   transmission is done(w_done)
always@(posedge i_clk, posedge i_rst)
  if(i_rst) o_txrdy <= 1'b1;
  else
    case({w_done,i_write})
      2'b00: o_txrdy <= o_txrdy;
      2'b01: o_txrdy <= 1'b0;
      2'b10: o_txrdy <= 1'b1;
      2'b11: o_txrdy <= o_txrdy;
    endcase

// RS FLOP instance for DONE
//   holding w_doit signal until got reset
//   by w_done.
RS_FLOP
  rs_doit_inst
    (.CLK(i_clk),
     .RST(i_rst),
     .R(w_done),
     .S(r_ld_d1),
     .Q(w_doit)
    );
```

Chip Specification

55 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

```
// LOAD_D1 FLOP
// holding load signal for a clock
always@(posedge i_clk, posedge i_rst)
    if(i_rst) r_ld_d1 <= 1'b0;
    else      r_ld_d1 <= i_write;

// LOAD BYTE for SHIFT REGISTER
// loading new 8-bit data when
// got a load signal(i_write)
always@(posedge i_clk, posedge i_rst)
    if(i_rst) r_byte <= 8'b0; else
    if(i_write) r_byte <= i_byte;
    else      r_byte <= r_byte;

// SHIFT REGISTER FUNCTION
// Arrange 8-bit incoming data and
// output an 11-bit data for shift register
// according to 3-bit parity control.
SHIFT_R_FUNC sh_func_inst
    (.BYTE(r_byte),
     .EIGHT(i_eight),
     .PEN(i_pen),
     .OHEN(i_ohel),
     .DOUT(w_sh_data)
    );

// SHIFT REGISTER
// Serially shifts right the 11-bit data
// through TX(o_tx) every bit-time up,
// and receives new 11-bit data at load
// signal(r_ld_d1)
SHIFT_R sh_r_inst
    (.CLK(i_clk),
     .RST(i_rst),
     .LOAD(r_ld_d1),
     .SHIFT(w_btu),
     .SDI(1'b1),
     .DIN(w_sh_data),
     .SDO(o_tx)
    );

endmodule
```

Chip Specification

56 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

Appendix D Baud Decoder Source Code

```
`timescale 1ns / 1ps
//*****//
//      Class:          CECS460 System on Chip Design          //
//      Project name:    Project2_UART_TX                      //
//      File name:       BAUD_DEC.v                            //
//                                                              //
//      Created by Chanartip Soonthornwan on March 1, 2018.    //
//      Copyright @ 2018 Chanartip Soonthornwan. All rights reserved. //
//                                                              //
//      Abstract:        A function to select Baud Rate for the UART protocol //
//*****//
module BAUD_DEC(BAUD_VAL,K);

    input    [ 3:0] BAUD_VAL;
    output [18:0] K;
    reg      [18:0] K;
    // function f() for K
    always@(*)
        case(BAUD_VAL)
            19'b0000: K = 19'd333333;
            19'b0001: K = 19'd 83333;
            19'b0010: K = 19'd 41667;
            19'b0011: K = 19'd 20833;
            19'b0100: K = 19'd 10417;
            19'b0101: K = 19'd 5208;
            19'b0110: K = 19'd 2604;
            19'b0111: K = 19'd 1736;
            19'b1000: K = 19'd 868;
            19'b1001: K = 19'd 434;
            19'b1010: K = 19'd 217;
            19'b1011: K = 19'd 109;
            default: K = 19'd 0;
        endcase
endmodule
```


Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

Appendix E RS Flop Source Code

```

`timescale 1ns / 1ps
//*****
//      Class:          CECS460 System on Chip Design          //
//      Project name:   Project2_UART_TX                      //
//      File name:      RS_FLOP.v                             //
//                                                              //
//      Created by Chanartip Soonthornwan on March 1, 2018.    //
//      Copyright @ 2018 Chanartip Soonthornwan. All rights reserved. //
//                                                              //
//      Abstract:       RS flip flop is another kind of flop   //
//                      outputs LOW at posedge of reset or R,    //
//                      outputs HIGH at posedge of S            //
//*****
/* PORT LISTS
* Inputs: CLK - Clock
*          RST - Reset(System)
*          R   - Reset(Input)
*          S   - Set(Input)
* Output: Q   - Data out
*/
module RS_FLOP(CLK, RST, R, S, Q);
    input      CLK, RST;
    input      R, S;
    output reg  Q;

    always@(posedge CLK, posedge RST)
        if(RST) Q <= 1'b0;
        else
            case({S,R})
                2'b00: Q <= Q;
                2'b01: Q <= 1'b0;
                2'b10: Q <= 1'b1;
                2'b11: Q <= Q;
            endcase
endmodule

```

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

Appendix F Shift Register Function Source Code

```

`timescale 1ns / 1ps
//*****
//      Class:          CECS460 System on Chip Design          //
//      Project name:   Project2_UART_TX                      //
//      File name:      SHIFT_R_FUNC.v                        //
//                                                              //
//      Created by Chanartip Soonthornwan on March 1, 2018.    //
//      Copyright @ 2018 Chanartip Soonthornwan. All rights reserved. //
//                                                              //
//      Abstract:       A function block to setup data package for //
//                      SHIFT_R module                          //
//*****
/*
    Shift Register Function is a combination logic block where it takes
    Parity-bit control (Eight-bit, Parity Enable, and Odd parity High/ Even
    parity Low) and converges in to 11-bit data package sending to the actual
    Shift Register.
    To find that the data if, for example, even parity or not, utilizing an
    Exclusive Or (^) informs that Low if there are even numbers of one's in the
    data, or High if there are odd number of one's in the data.
*/
module SHIFT_R_FUNC(
    // inputs
    BYTE,          // 8-bit data from TramelBlaze out_port
    EIGHT,         // 7-or-8-bit to be transmitted
    PEN,           // Parity bit Enable
    OHEL,          // Odd-High-Even-Low to clarify which parity bit to send
    // outputs
    DOUT           // 11-bit data package to SHIFT_R
);
    input  [ 7:0] BYTE;
    input          EIGHT;
    input          PEN;
    input          OHEL;
    output [10:0] DOUT;
    reg    [10:0] DOUT;
    // Shift Register Data input Logic
always@(*) begin
    DOUT[8:0] = {BYTE[6:0],1'b0,1'b1};
    case({EIGHT, PEN, OHEL})
        3'b000: DOUT[10:9] = 2'b11;
        3'b001: DOUT[10:9] = 2'b11;
        3'b010: DOUT[10:9] = {1'b1, ^BYTE[6:0]}; // Even Parity[7]
        3'b011: DOUT[10:9] = {1'b1, ~^BYTE[6:0]}; // Odd Parity[7]
        3'b100: DOUT[10:9] = {1'b1,BYTE[7]};
        3'b101: DOUT[10:9] = {1'b1,BYTE[7]};
        3'b110: DOUT[10:9] = { ^BYTE[7:0],BYTE[7]}; // Even Parity[8]
        3'b111: DOUT[10:9] = { ~^BYTE[7:0],BYTE[7]}; // Odd Parity[8]
    endcase
end
endmodule

```

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

Appendix G Shift Register Source Code

```
`timescale 1ns / 1ps
//*****
//      Class:          CECS460 System on Chip Design          //
//      Project name:   Project2_UART_TX                      //
//      File name:      SHIFT_R.v                             //
//                                                              //
//      Created by Chanartip Soonthornwan on March 1, 2018.    //
//      Copyright @ 2018 Chanartip Soonthornwan. All rights reserved. //
//                                                              //
//      Abstract:       Serial right shifting input through by Serial Data //
//                      Output(SDO) when received SHIFT signal, and reload //
//                      new Data package(DIN) when LOAD is active. //
//*****
module SHIFT_R(
    // inputs
    CLK,          // System clock
    RST,          // System reset
    LOAD,         // Load new Data Input signal
    SHIFT,        // Shifting signal
    SDI,          // Serial Data input on shifting
    DIN,          // Package Data to shift
    // outputs
    SDO           // a-bit of shifted data
);
    input        CLK;
    input        RST;
    input        LOAD;
    input        SHIFT;
    input        SDI;
    input [10:0] DIN;
    output       SDO;

    parameter ONES = 11'h7FF;

    reg  [10:0] SR;

    // Output is always the LSB of Serial Data
    assign SDO = SR[0];

    /* Load new data package when LOAD is active,
       and shift right the data when SHIFT is active.
       Otherwise, the serial data stay the same. */
    always@(posedge CLK, posedge RST)
        if(RST)  SR <= ONES;
        else
            if(LOAD)  SR <= DIN; else
            if(SHIFT) SR <= {SDI,SR[10:1]};
            else      SR <= SR;

endmodule
```

Chip Specification

60 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

Appendix H UART_RX Code

```
`timescale 1ns / 1ps
//*****
// This document contains information proprietary to the
// CSULB student that created the file - any reuse without
// adequate approval and documentation is prohibited
//
// Class: CECS460 System on Chip Design
// Project name: Project3_UART_RX
// File name: UART_RX.v
//
// Created by Chanartip Soonthornwan on March 18, 2018.
// Copyright @ 2018 Chanartip Soonthornwan. All rights reserved.
//
// Abstract: A top level displaying interconnection wire between
// Datapath and Control Unit of RX Engine.
//
//*****
module UART_RX(
    input i_clk, // System clk
    input i_rst, // System reset
    input i_read, // Read signal from TramelBlaze to clear flag status
    input i_rx, // RX line, 1-bit data input
    input i_eight, // Eight bit Control
    input i_pen, // Parity bit Control
    input i_ohel, // Odd/Even bit Control
    input [18:0] i_rate, // Buad Rate

    output o_rxrdy, // RX enginge ready status
    output o_perr, // Parity bit error status
    output o_ferr, // Framing error status
    output o_ovf, // Overflow error status
    output [ 7:0] o_rx_dout // Received Data output to TramelBlaze
);

// Wires from Controls to Datapath
wire w_btu;
wire w_done;
wire w_start;

// Receive Engine Control Unit
RX_CONTROL rx_ctrl_inst(
    .i_clk(i_clk),
    .i_rst(i_rst),
    .i_rate(i_rate),
    .i_rx(i_rx),
    .i_eight(i_eight),
    .i_pen(i_pen),

    .o_btu(w_btu),
    .o_done(w_done),
    .o_start(w_start)
);

// Receive Engine Datapath
RX_DATAPATH rx_dp_inst(
    .i_clk(i_clk),
    .i_rst(i_rst),
    .i_read(i_read),

```

Chip Specification

61 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

```
.i_btu(w_btu),  
.i_done(w_done),  
.i_start(w_start),  
.i_rx(i_rx),  
.i_eight(i_eight),  
.i_pen(i_pen),  
.i_ohel(i_ohel),  
  
.o_rxrdy(o_rxrdy),  
.o_perr(o_perr),  
.o_ferr(o_ferr),  
.o_ovf(o_ovf),  
.o_rx_byte(o_rx_dout)  
);
```

endmodule

Chip Specification

62 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

Appendix I RX_Control Code

```
`timescale 1ns / 1ps
//*****
// This document contains information proprietary to the
// CSULB student that created the file - any reuse without
// adequate approval and documentation is prohibited
//
// Class: CECS460 System on Chip Design
// Project name: Project3_UART_RX
// File name: RX_Control.v
//
// Created by Chanartip Soonthornwan on March 18, 2018.
// Copyright © 2018 Chanartip Soonthornwan. All rights reserved.
//
// Abstract: Manipulate Datapath control, Generating bit time
// and bit count corresponded to control input,
// utilizing a Finite State Machine detecting a
// High-to-Low Transition(Start bit)
//
//*****
module RX_CONTROL(
    input i_clk, // System clk
    input i_rst, // System reset
    input [18:0] i_rate, // Constant count for bard rate
    input i_rx, // Receiving bit input
    input i_eight, // Eight bit selection
    input i_pen, // Parity bit selection

    output o_btu, // Bit Time Up signal
    output o_done, // Done receiving a data frame signal
    output o_start // Start receiving a new frame signal
);

// Registers and Parameters for Finite State machine
reg doit;
reg n_doit;
reg start;
reg n_start;
reg [1:0] state;
reg [1:0] n_state;

parameter IDLE = 2'b00, START = 2'b01, DOIT = 2'b10;

// Registers for Bit-Time counter and Bit-count coutner
wire [18:0] k;
reg [18:0] count;
reg [3:0] bit_count;
reg [3:0] num_bit;

// BIT_TIME_COUNTER
// a pulse maker creates a pulse(o_btu)
// when bit time is up by incrementing
// a 19-bit counter to reach bit_time value(k)
// then generates a pulse with one clock period.
// k is a time constant holding either
// regular baud rate or half baud rate.
// Half baud rate will be selected on the
// event of starting receiving a new frame of data.
assign o_btu = (count == k);
```

Chip Specification

63 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

```

assign      k = (start)? i_rate>>1 : i_rate;

always@(posedge i_clk, posedge i_rst)
    if(i_rst) count <= 19'b0; else
    if(o_btu) count <= 19'b0; else
    if(doit) count <= count + 19'b1;
    else count <= 19'b0;

// BIT_COUNTER
// a pulse maker creates a pulse(w_done)
// when the number of TX bits are transmitted(num_bit),
// and has a counter(bit_count) to keep tracking
// of number of bits for each bit time up.
// num_bit is the number of bits per frame
// from the transmission (start bit + data bits + stop bit)
assign o_done = (bit_count == num_bit);

always@(posedge i_clk, posedge i_rst)
    if(i_rst) bit_count <= 4'b0; else
    if(doit) begin
        if(o_btu) bit_count <= bit_count + 4'b1;
        else bit_count <= bit_count;
    end
    else bit_count <= 4'b0;

always@(*)
    case({i_eight, i_pen})
        2'b00: num_bit = 4'd9;
        2'b01: num_bit = 4'd10;
        2'b10: num_bit = 4'd10;
        2'b11: num_bit = 4'd11;
    endcase

// Finite State Machine(FSM)
// a state machine responses for polling RX_line(i_rx)
// looking for high to low transition indicating the
// arrival of START bit(IDLE -> START state),
// and ensures that the start bit remains active(low active)
// for half bit-time which is the arrival of collecting
// data time. (START state).
// At the data collecting time, the state machine
// signals 'doit' for datapath (DOIT state) until the
// end of data frame, then signals done.
assign o_start = start;

always@(posedge i_clk, posedge i_rst)
    if(i_rst) {state, start, doit} <= {2'b0,1'b0,1'b0};
    else {state, start, doit} <= {n_state, n_start, n_doit};

always@(*) begin
    case(state)
        IDLE: begin
            {n_state, n_start, n_doit} = (i_rx) ? { IDLE, 2'b00} :
                                                {START, 2'b11} ;
        end
        START: begin
            {n_state, n_start, n_doit} = (i_rx) ? { IDLE, 2'b00} :
                                                (!o_btu) ? {START, 2'b11} :
                                                { DOIT, 2'b01} ;
        end
    end
end

```

Chip Specification

64 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

```
DOIT: begin
    {n_state, n_start, n_doit} = (o_done) ? { IDLE, 2'b00} :
                                     { DOIT,
2'b01} ;
    end
    default: {n_state, n_start, n_doit} = { IDLE, 2'b00} ;
endcase
end
endmodule
```


Chip Specification

65 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

Appendix J RX_DATAPATH Code

```
`timescale 1ns / 1ps
//*****
// This document contains information proprietary to the
// CSULB student that created the file - any reuse without
// adequate approval and documentation is prohibited
//
// Class: CECS460 System on Chip Design
// Project name: Project3_UART_RX
// File name: RX_DATAPATH.v
//
// Created by Chanartip Soonthornwan on April 17, 2018.
// Copyright @ 2018 Chanartip Soonthornwan. All rights reserved.
//
// Abstract: Datapath of Receive Engine where passing transmitted
// data to the TramelBlaze and checking the data
// transmission's error.
//
// Revision 1.1: Current Date April 17, 2018
// - Add a case of 7N1, 7O1, 7E1 outputs to TramelBlaze
//
// Revision 1.0: Date March 18, 2018
//
//*****
/*
Receiving Enging Datapath
Receiving data through RX_line(i_rx) to Shift Register and then REMAPP
the received data before sending to the TramelBlaze while checking
the transmission's error.
*/
module RX_DATAPATH(
    input i_clk,           // System Clock
    input i_rst,           // System Reset
    input i_read,          // interrupt acknowledge from TramelBlaze
    input i_btu,           // Bit Time up signal from RX_Control
    input i_done,          // Done a frame signal from RX_Control
    input i_start,         // Start bit signal from RX_Control
    input i_rx,            // RX_Line serial input from another device
    input i_eight,         // Eight-bit selection
    input i_pen,           // Parity-bit selection
    input i_ohel,          // Odd or Even parity bit selection

    output reg o_rxrdy,    // Receive Engine ready signal
    output reg o_perr,     // Parity Error
    output reg o_ferr,     // Framing Error
    output reg o_ovf,      // Overflow Error
    output [7:0] o_rx_byte // a byte output to TramelBlaze
);

wire [9:0] w_sh_data;      // wiring from RX_SHIFT_REG to RX_REMAP
wire [9:0] w_map_data;    // wiring from RX_REMAP to the rest of DATAPATH

// Receive Engine Shifting Register
// receives serial input through RX_line
// and shift a frame of data to REMAP register.
wire w_shift;
assign w_shift = i_btu & ~i_start;

RX_SHIFT_REG rx_shift_reg_inst(
```

Chip Specification

66 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

```
.CLK(i_clk),
.RST(i_rst),
.SH(w_shift),
.SDI(i_rx),
.SH_DATA(w_sh_data)
);

// Receive Engine Re-mapping Register
// re-arrange shifted data before sending
// out to TramelBlaze and for Errors Checking.
RX_REMAP rx_remap_inst(
.i_eight(i_eight),
.i_pen(i_pen),
.i_data(w_sh_data),
.o_data(w_map_data)
);

// Assigning the re-mapped data to TramelBlaze
assign o_rx_byte = (i_eight)? w_map_data[7:0]: {1'b0, w_map_data[6:0]};

// PARITY GEN SELECT
wire w_par_gen_sel_mux;
assign w_par_gen_sel_mux = (i_eight)? w_map_data[7]: 1'b0;

wire w_even;
assign w_even = ~i_ohel;

wire w_par_even_mux;
assign w_par_even_mux = (w_even)? (w_map_data[6:0] ^ w_par_gen_sel_mux):
~(w_map_data[6:0] ^ w_par_gen_sel_mux);

// PARITY BIT SELECT
wire w_par_bit_sel_mux;
assign w_par_bit_sel_mux = (i_eight)? w_map_data[8]: w_map_data[7];

// STOP BIT SELECT
reg w_stop_bit_sel_mux;
always(*)
case({i_eight,i_pen})
2'b00: w_stop_bit_sel_mux = w_map_data[7];
2'b01: w_stop_bit_sel_mux = w_map_data[8];
2'b10: w_stop_bit_sel_mux = w_map_data[8];
2'b11: w_stop_bit_sel_mux = w_map_data[9];
endcase

// RXRDY RS_FLOP
always@(posedge i_clk, posedge i_rst)
if(i_rst) o_rxdy <= 1'b0; else
if(i_read) o_rxdy <= 1'b0; else
if(i_done) o_rxdy <= 1'b1;
else o_rxdy <= o_rxdy;

// PERR RS_FLOP
wire w_set_perr;
assign w_set_perr = i_pen & i_done & (w_par_even_mux ^ w_par_bit_sel_mux);
always@(posedge i_clk, posedge i_rst)
if(i_rst) o_perr <= 1'b0; else
if(i_read) o_perr <= 1'b0; else
if(w_set_perr) o_perr <= 1'b1;
else o_perr <= o_perr;
```

Chip Specification

67 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

```
// FERR RS_FLOP
wire    w_set_ferr;
assign  w_set_ferr = i_done & ~w_stop_bit_sel_mux;
always@(posedge i_clk, posedge i_rst)
    if(i_rst)      o_ferr <= 1'b0; else
    if(i_read)      o_ferr <= 1'b0; else
    if(w_set_ferr)  o_ferr <= 1'b1;
    else           o_ferr <= o_ferr;

// OVF RS_FLOP
wire    w_set_ovf;
assign  w_set_ovf = i_done & o_rxrdy;
always@(posedge i_clk, posedge i_rst)
    if(i_rst)      o_ovf <= 1'b0; else
    if(i_read)      o_ovf <= 1'b0; else
    if(w_set_ovf)   o_ovf <= 1'b1;
    else           o_ovf <= o_ovf;

endmodule
```

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

Appendix K *RX_REMAP Code*

```

`timescale 1ns / 1ps
//*****
// This document contains information proprietary to the
// CSULB student that created the file - any reuse without
// adequate approval and documentation is prohibited
//
// Class:          CECS460 System on Chip Design
// Project name:   Project3_UART_RX
// File name:      RX_REMAP.v
//
// Created by Chanartip Soonthornwan on March 18, 2018.
// Copyright © 2018 Chanartip Soonthornwan. All rights reserved.
//
// Abstract:       A combinational block to re-arrange incoming data
//                 by shifting the data to its correct place according
//                 the data controls, and ensures that the data frame
//                 is ready to be for error checking.
//
//*****
/*
Receiving Engine Datapath Remapping module
Rearranging Received data from RX Shift Register according to
the control:
by default: there are 7-bit data plus one stop bit (8 bits)
if eight : there are 8-bit data plus one stop bit (9 bits)
if pen   : there are 7-bit data plus one parity bit
           and one stop bit(9 bits)
if pen and eight: there are 10 bits in total.

such that 8 bits will be stopped shifting and be off by 2 bits,
therefore 8 bits data will be right shifted by 2.
likewise, 9 bits will be shifted by 1, but 10 bits will not be shifted.
*/
module RX_REMAP(
    input      i_eight,      // Eight-bit data select
    input      i_pen,        // Parity-bit select
    input [9:0] i_data,       // 10-bit data from RX Shift Register
    output reg [9:0] o_data   // Remapped data
);

    always@(*) begin
        case({i_eight,i_pen})
            2'b00: o_data = {2'b11, i_data[9:2]};
            2'b01: o_data = {1'b1 , i_data[9:1]};
            2'b10: o_data = {1'b1 , i_data[9:1]};
            2'b11: o_data = i_data;
        endcase
    end

endmodule

```

Chip Specification

69 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

Appendix L RX_SHIFT_REG Code

```
`timescale 1ns / 1ps
//*****
// This document contains information proprietary to the
// CSULB student that created the file - any reuse without
// adequate approval and documentation is prohibited
//
// Class: CECS460 System on Chip Design
// Project name: Project3_UART_RX
// File name: RX_SHIFT_REG.v
//
// Created by Chanartip Soonthornwan on March 18, 2018.
// Copyright © 2018 Chanartip Soonthornwan. All rights reserved.
//
// Abstract: Shifting Register to shift data received from
// RX_line into receive engine when High to Low
// Transmission occurred(START bit) and at raising
// edge of Bit Time Up(BTU).
//
//*****
/*
Receiving Engine Shifting Register
Shifting input from RX_line to REMAP Register when START bit
has been acknowledged and shifts data out on raising edge
of bit time up(btu).
Shifting signal(SH) is the combinational logic of BTU & ~START
*/
module RX_SHIFT_REG(CLK, RST, SH, SDI, SH_DATA);
    input CLK; // System clock
    input RST; // System reset
    input SH; // Shifting signal
    input SDI; // Serial Data Input
    output reg [9:0] SH_DATA; // Shifting output

    always@(posedge CLK, posedge RST)
        if(RST) SH_DATA <= 10'b0; else
            if(SH) SH_DATA <= {SDI, SH_DATA[9:1]};
            else SH_DATA <= SH_DATA;

endmodule
```

Chip Specification

70 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

Appendix M PED Source Code

```
`timescale 1ns / 1ps
//*****//
//    Class:          CECS460 System on Chip Design          //
//    Project name:    Project2_UART_TX                      //
//    File name:       PED.v                                  //
//                                                            //
//    Created by Chanartip Soonthornwan on September 17, 2017. //
//    Copyright @ 2017 Chanartip Soonthornwan. All rights reserved. //
//                                                            //
//    Abstract:        A module to detect Positive Edge input then //
//                    returns one-shot pulse output.           //
//                    If the input is HIGH at the first clock and //
//                    second clock period, PED would detect this //
//                    and output HIGH for one clock period.     //
//*****//
module PED(clk, rst, d_in, pulse);

    input      clk, rst;          // on-board clock, and AISO reset signal
    input      d_in;             // input signal
    output wire pulse;           // one-shot pulse

    reg        q1,q2;            // registers

    always@(posedge clk, posedge rst)
        if(rst) {q1, q2} <= 2'b0; // reset
        else    {q1, q2} <= {d_in, q1}; // q2 gets q1, and q1 get new signal

    // output at the moment of input change
    // q1  _____
    // q2  _____
    // pulse ----
    assign pulse = q1 & ~q2;
endmodule
```

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

Appendix N Address Decoder Source Code

```

`timescale 1ns / 1ps
//*****
//    Class:          CECS460 System on Chip Design          //
//    Project name:   Project2_UART_TX                      //
//    File name:      ADDR_DECODER.v                        //
//                                                           //
//    Created by Chanartip Soonthornwan on March 1, 2018.    //
//    Copyright @ 2018 Chanartip Soonthornwan. All rights reserved. //
//                                                           //
//    Abstract:       Decoding Address from TramelBlaze's PORT_ID //
//                   and Read/Write Strobe to 16 addresses inputs/outputs //
//*****
/*
    Assigning Each 16 inputs and 16 outputs either reading or writing
    with the wr_strobe and rd_strobe.
    note: port_id == 16'b0 is when TramelBlaze doesn't read or write.
*/

module ADDR_DECODER(port_id, wr_strobe, rd_strobe, writes, reads);

    input  [15:0] port_id;
    input          wr_strobe;
    input          rd_strobe;
    output [15:0] writes;
    output [15:0] reads;

    reg [15:0] writes;
    reg [15:0] reads;

    always @(*)
        begin
            writes = 16'b0;
            reads  = 16'b0;
            writes[port_id] = wr_strobe;
            reads[port_id]  = rd_strobe;
        end

endmodule

```

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

Appendix O AISO Reset Source Code

```

`timescale 1ns / 1ps
//*****
// Class:          CECS460 System on Chip Design          //
// Project name:    Project2_UART_TX                      //
// File name:       AISO_rst.v                            //
//                                                         //
// Created by Chanartip Soonthornwan on September 17, 2017. //
// Copyright @ 2017 Chanartip Soonthornwan. All rights reserved. //
//                                                         //
// Abstract:        Receives reset signal input from a reset button //
//                  then generates synchronized output at rising edge //
//                  to other module in the design.          //
//*****
module AISO_RST(clk, rst, rst_s);

    input          clk, rst;                // on-board clock, and AISO reset signal
    output wire    rst_s;                  // Synchronized reset signal
    reg            q1, q2;                 // registers

    always@(posedge clk, posedge rst)
        if(rst) {q1,q2} <= 2'b0;           // reset
        else    {q1,q2} <= {1'b1, q1};     // q2 gets q1, and q1 get 1'b1

    /*
    * if reset(rst) is HIGH, the output will be HIGH
    * else output will always be LOW
    */
    assign rst_s = ~q2;

endmodule

```


73 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

Appendix P Full UART Program

[illegible]

Chip Specification

74 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

HOW_MANY EQU RB
RESULT EQU RC

```

;=====
; Port alias
;=====
DATA_PORT EQU 0000
STATUS_PORT EQU 0001
LED_PORT EQU 0002

;=====
; ASCII in uses
;=====
ASC_NULL EQU 0000 ; NULL
ASC_BS EQU 0008 ; Back Space
ASC_TAB EQU 0009 ; Horizontal Tab
ASC_LF EQU 000A ; <LF> Line Feed (new line)
ASC_CR EQU 000D ; <CR> Carriage return
ASC_STAR EQU 002A ; '*' star
ASC_AT EQU 0040 ; '@' at
ASC_DOT EQU 002E ; '.' dot
ASC_SL EQU 002F ; '/' slash
ASC_EQ EQU 003D ; '=' equal sign
ASC_DASH EQU 002D ; '-' dash
ASC_SP EQU 0020 ; ' ' space
ASC_LB EQU 005B ; '[' Left Bracket
ASC_RB EQU 005D ; ']'; Right Bracket
ASC_0 EQU 0030
ASC_1 EQU 0031
ASC_2 EQU 0032
ASC_3 EQU 0033
ASC_4 EQU 0034
ASC_5 EQU 0035
ASC_6 EQU 0036
ASC_7 EQU 0037
ASC_8 EQU 0038
ASC_9 EQU 0039
ASC_A EQU 0041
ASC_B EQU 0042
ASC_C EQU 0043
ASC_D EQU 0044
ASC_E EQU 0045
ASC_F EQU 0046
ASC_G EQU 0047
ASC_H EQU 0048
ASC_I EQU 0049
ASC_J EQU 004A
ASC_K EQU 004B
ASC_L EQU 004C
ASC_M EQU 004D
ASC_N EQU 004E
ASC_O EQU 004F
ASC_P EQU 0050
ASC_Q EQU 0051
ASC_R EQU 0052
ASC_S EQU 0053
ASC_T EQU 0054
ASC_U EQU 0055
ASC_V EQU 0056
ASC_W EQU 0057

```

Chip Specification

75 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

```
ASC_X      EQU    0058
ASC_Y      EQU    0059
ASC_Z      EQU    005A
```

```
;=====
; Startup: Initialize the design at reset
;
; Procedure:
;   - Reset and Set registers
;   - Store Banner, Prompt, HOMETOWN, CRLF and BSSP
;     in the ScratchPad RAM
;   - Reset the index register to be ready for the next use.
;   - Start the Program with Interrupt Enable
;
; Main() {
;   BLINK_LED();
; }
;
;=====
                ADDRESS 0000
INIT
; Initialize registers
                LOAD     TEMP_REG, ZEROS
                LOAD     COUNT, ZEROS
                LOAD     INDEX, ZEROS
                LOAD     LED, ZEROS
                LOAD     PRINT_FLAG, FOUR
                LOAD     UART_STATUS, ZEROS
                LOAD     CHAR_COUNT, ZEROS

                CALL     BANNER_INIT
                CALL     PROMPT_INIT
                CALL     HOMETOWN_INIT

; Initialize CRLF in ScratchPad
                LOAD     TEMP_REG, ASC_CR
                STORE    TEMP_REG, INDEX
                ADD      INDEX, ONE
                LOAD     TEMP_REG, ASC_LF
                STORE    TEMP_REG, INDEX
                ADD      INDEX, ONE

; Initialize BSSP in ScratchPad
                LOAD     TEMP_REG, ASC_BS
                STORE    TEMP_REG, INDEX
                ADD      INDEX, ONE
                LOAD     TEMP_REG, ASC_SP
                STORE    TEMP_REG, INDEX
                ADD      INDEX, ONE
                LOAD     TEMP_REG, ASC_BS
                STORE    TEMP_REG, INDEX

                AND      INDEX, ZEROS ; Reset to ZEROS to be ready to display

START          ENINT

; Main Loop
MAIN
                CALL     BLINK_LED
                JUMP     MAIN
```

Chip Specification

76 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

```

;=====
; Subroutine: BANNER_INIT
;   Displaying a banner with a prompt upon reset
;
;   Store a Banner to ScratchPad Memory
;   //////////////////////////////////////
;   CECS460 PROJECT3
;   By  CHANARTIP SOONTHORNWAN
;   //////////////////////////////////////
;
;   by incrementing the memory pointer each time storing a character
;
;   input registers: none
;   output registers: none
;=====
ADDRESS 0040
BANNER_INIT
SL_50    LOAD    TEMP_REG, ASC_SL
        STORE   TEMP_REG, INDEX
        ADD     INDEX, ONE
        ADD     COUNT, ONE
        COMP    COUNT, 0031          ; if count < 0d50
        JUMPC   SL_50
        LOAD    COUNT, ZEROS

        COMP    INDEX, 0064          ; if index > 0d100
        JUMPNC  DONE_BANNER

        LOAD    TEMP_REG, ASC_CR
        STORE   TEMP_REG, INDEX
        ADD     INDEX, ONE
        LOAD    TEMP_REG, ASC_LF
        STORE   TEMP_REG, INDEX
        ADD     INDEX, ONE

        LOAD    TEMP_REG, ASC_TAB
        STORE   TEMP_REG, INDEX
        ADD     INDEX, ONE
        LOAD    TEMP_REG, ASC_C
        STORE   TEMP_REG, INDEX
        ADD     INDEX, ONE
        LOAD    TEMP_REG, ASC_E
        STORE   TEMP_REG, INDEX
        ADD     INDEX, ONE
        LOAD    TEMP_REG, ASC_C
        STORE   TEMP_REG, INDEX
        ADD     INDEX, ONE
        LOAD    TEMP_REG, ASC_S
        STORE   TEMP_REG, INDEX
        ADD     INDEX, ONE
        LOAD    TEMP_REG, ASC_4
        STORE   TEMP_REG, INDEX
        ADD     INDEX, ONE
        LOAD    TEMP_REG, ASC_6
        STORE   TEMP_REG, INDEX
        ADD     INDEX, ONE
        LOAD    TEMP_REG, ASC_0
        STORE   TEMP_REG, INDEX
        ADD     INDEX, ONE

```

Chip Specification

77 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
-------------------------------------	--------------------------------	---	-----------------------

```

LOAD    TEMP_REG, ASC_SP
STORE   TEMP_REG, INDEX
ADD    INDEX, ONE
LOAD    TEMP_REG, ASC_P
STORE   TEMP_REG, INDEX
ADD    INDEX, ONE
LOAD    TEMP_REG, ASC_R
STORE   TEMP_REG, INDEX
ADD    INDEX, ONE
LOAD    TEMP_REG, ASC_O
STORE   TEMP_REG, INDEX
ADD    INDEX, ONE
LOAD    TEMP_REG, ASC_J
STORE   TEMP_REG, INDEX
ADD    INDEX, ONE
LOAD    TEMP_REG, ASC_E
STORE   TEMP_REG, INDEX
ADD    INDEX, ONE
LOAD    TEMP_REG, ASC_C
STORE   TEMP_REG, INDEX
ADD    INDEX, ONE
LOAD    TEMP_REG, ASC_T
STORE   TEMP_REG, INDEX
ADD    INDEX, ONE
LOAD    TEMP_REG, ASC_3
STORE   TEMP_REG, INDEX
ADD    INDEX, ONE
LOAD    TEMP_REG, ASC_CR
STORE   TEMP_REG, INDEX
ADD    INDEX, ONE
LOAD    TEMP_REG, ASC_LF
STORE   TEMP_REG, INDEX
ADD    INDEX, ONE
LOAD    TEMP_REG, ASC_TAB
STORE   TEMP_REG, INDEX
ADD    INDEX, ONE
LOAD    TEMP_REG, ASC_B
STORE   TEMP_REG, INDEX
ADD    INDEX, ONE
LOAD    TEMP_REG, ASC_Y
STORE   TEMP_REG, INDEX
ADD    INDEX, ONE
LOAD    TEMP_REG, ASC_CR
STORE   TEMP_REG, INDEX
ADD    INDEX, ONE
LOAD    TEMP_REG, ASC_LF
STORE   TEMP_REG, INDEX
ADD    INDEX, ONE
LOAD    TEMP_REG, ASC_TAB
STORE   TEMP_REG, INDEX
ADD    INDEX, ONE
LOAD    TEMP_REG, ASC_C
STORE   TEMP_REG, INDEX
ADD    INDEX, ONE
LOAD    TEMP_REG, ASC_H
STORE   TEMP_REG, INDEX
ADD    INDEX, ONE
LOAD    TEMP_REG, ASC_A
STORE   TEMP_REG, INDEX
ADD    INDEX, ONE

```

Chip Specification

78 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
-------------------------------------	--------------------------------	---	-----------------------

```

LOAD      TEMP_REG, ASC_N
STORE     TEMP_REG, INDEX
ADD      INDEX, ONE
LOAD      TEMP_REG, ASC_A
STORE     TEMP_REG, INDEX
ADD      INDEX, ONE
LOAD      TEMP_REG, ASC_R
STORE     TEMP_REG, INDEX
ADD      INDEX, ONE
LOAD      TEMP_REG, ASC_T
STORE     TEMP_REG, INDEX
ADD      INDEX, ONE
LOAD      TEMP_REG, ASC_I
STORE     TEMP_REG, INDEX
ADD      INDEX, ONE
LOAD      TEMP_REG, ASC_P
STORE     TEMP_REG, INDEX
ADD      INDEX, ONE
LOAD      TEMP_REG, ASC_SP
STORE     TEMP_REG, INDEX
ADD      INDEX, ONE
LOAD      TEMP_REG, ASC_S
STORE     TEMP_REG, INDEX
ADD      INDEX, ONE
LOAD      TEMP_REG, ASC_O
STORE     TEMP_REG, INDEX
ADD      INDEX, ONE
LOAD      TEMP_REG, ASC_O
STORE     TEMP_REG, INDEX
ADD      INDEX, ONE
LOAD      TEMP_REG, ASC_N
STORE     TEMP_REG, INDEX
ADD      INDEX, ONE
LOAD      TEMP_REG, ASC_T
STORE     TEMP_REG, INDEX
ADD      INDEX, ONE
LOAD      TEMP_REG, ASC_H
STORE     TEMP_REG, INDEX
ADD      INDEX, ONE
LOAD      TEMP_REG, ASC_O
STORE     TEMP_REG, INDEX
ADD      INDEX, ONE
LOAD      TEMP_REG, ASC_R
STORE     TEMP_REG, INDEX
ADD      INDEX, ONE
LOAD      TEMP_REG, ASC_N
STORE     TEMP_REG, INDEX
ADD      INDEX, ONE
LOAD      TEMP_REG, ASC_W
STORE     TEMP_REG, INDEX
ADD      INDEX, ONE
LOAD      TEMP_REG, ASC_A
STORE     TEMP_REG, INDEX
ADD      INDEX, ONE
LOAD      TEMP_REG, ASC_N
STORE     TEMP_REG, INDEX
ADD      INDEX, ONE
LOAD      TEMP_REG, ASC_CR
STORE     TEMP_REG, INDEX
ADD      INDEX, ONE

```

Chip Specification

79 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
-------------------------------------	--------------------------------	---	-----------------------

```

        LOAD    TEMP_REG, ASC_LF
        STORE   TEMP_REG, INDEX
        ADD     INDEX, ONE

        LOAD    TEMP_REG, ASC_SL
        JUMP    SL_50

DONE_BANNER
        LOAD    TEMP_REG, ASC_CR
        STORE   TEMP_REG, INDEX
        ADD     INDEX, ONE
        LOAD    TEMP_REG, ASC_LF
        STORE   TEMP_REG, INDEX
        ADD     INDEX, ONE           ; should be index for PROMPT BEGIN

        LOAD    COUNT, ZEROS

        RETURN

;=====
; Subroutine:  PROMPT_INIT
;   Store a Prompt into the ScratchPad Memory
;
;   procedure:
;       - increment the index each time storing a character
;
;       PRESS ANY KEY TO CONTINUE...
;
;   input registers: none
;   output registers: none
;=====
        ADDRESS 0170

PROMPT_INIT
        LOAD    TEMP_REG, ASC_P
        STORE   TEMP_REG, INDEX
        ADD     INDEX, ONE
        LOAD    TEMP_REG, ASC_R
        STORE   TEMP_REG, INDEX
        ADD     INDEX, ONE
        LOAD    TEMP_REG, ASC_E
        STORE   TEMP_REG, INDEX
        ADD     INDEX, ONE
        LOAD    TEMP_REG, ASC_S
        STORE   TEMP_REG, INDEX
        ADD     INDEX, ONE
        LOAD    TEMP_REG, ASC_S
        STORE   TEMP_REG, INDEX
        ADD     INDEX, ONE
        LOAD    TEMP_REG, ASC_SP
        STORE   TEMP_REG, INDEX
        ADD     INDEX, ONE
        LOAD    TEMP_REG, ASC_A
        STORE   TEMP_REG, INDEX
        ADD     INDEX, ONE
        LOAD    TEMP_REG, ASC_N
        STORE   TEMP_REG, INDEX
        ADD     INDEX, ONE
        LOAD    TEMP_REG, ASC_Y
        STORE   TEMP_REG, INDEX

```

Chip Specification

80 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

```

ADD      INDEX, ONE
LOAD     TEMP_REG, ASC_SP
STORE    TEMP_REG, INDEX
ADD      INDEX, ONE
LOAD     TEMP_REG, ASC_K
STORE    TEMP_REG, INDEX
ADD      INDEX, ONE
LOAD     TEMP_REG, ASC_E
STORE    TEMP_REG, INDEX
ADD      INDEX, ONE
LOAD     TEMP_REG, ASC_Y
STORE    TEMP_REG, INDEX
ADD      INDEX, ONE
LOAD     TEMP_REG, ASC_SP
STORE    TEMP_REG, INDEX
ADD      INDEX, ONE
LOAD     TEMP_REG, ASC_T
STORE    TEMP_REG, INDEX
ADD      INDEX, ONE
LOAD     TEMP_REG, ASC_O
STORE    TEMP_REG, INDEX
ADD      INDEX, ONE
LOAD     TEMP_REG, ASC_SP
STORE    TEMP_REG, INDEX
ADD      INDEX, ONE
LOAD     TEMP_REG, ASC_C
STORE    TEMP_REG, INDEX
ADD      INDEX, ONE
LOAD     TEMP_REG, ASC_O
STORE    TEMP_REG, INDEX
ADD      INDEX, ONE
LOAD     TEMP_REG, ASC_N
STORE    TEMP_REG, INDEX
ADD      INDEX, ONE
LOAD     TEMP_REG, ASC_T
STORE    TEMP_REG, INDEX
ADD      INDEX, ONE
LOAD     TEMP_REG, ASC_I
STORE    TEMP_REG, INDEX
ADD      INDEX, ONE
LOAD     TEMP_REG, ASC_N
STORE    TEMP_REG, INDEX
ADD      INDEX, ONE
LOAD     TEMP_REG, ASC_U
STORE    TEMP_REG, INDEX
ADD      INDEX, ONE
LOAD     TEMP_REG, ASC_E
STORE    TEMP_REG, INDEX
ADD      INDEX, ONE
LOAD     TEMP_REG, ASC_DOT
STORE    TEMP_REG, INDEX
ADD      INDEX, ONE
LOAD     TEMP_REG, ASC_DOT
STORE    TEMP_REG, INDEX
ADD      INDEX, ONE
LOAD     TEMP_REG, ASC_DOT
STORE    TEMP_REG, INDEX
ADD      INDEX, ONE      ; This INDEX should be HOMETOWN_BEGIN

RETURN

```


Chip Specification

81 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

```
;=====
; Sub Routine: HOMETOWN_INIT
;   Store "MY HOMETOWN IS BANGKOK"
;
;=====
ADDRESS 0200
```

HOMETOWN_INIT

```
LOAD    TEMP_REG, ASC_M
STORE   TEMP_REG, INDEX
ADD     INDEX, ONE
LOAD    TEMP_REG, ASC_Y
STORE   TEMP_REG, INDEX
ADD     INDEX, ONE
LOAD    TEMP_REG, ASC_SP
STORE   TEMP_REG, INDEX
ADD     INDEX, ONE
LOAD    TEMP_REG, ASC_H
STORE   TEMP_REG, INDEX
ADD     INDEX, ONE
LOAD    TEMP_REG, ASC_O
STORE   TEMP_REG, INDEX
ADD     INDEX, ONE
LOAD    TEMP_REG, ASC_M
STORE   TEMP_REG, INDEX
ADD     INDEX, ONE
LOAD    TEMP_REG, ASC_E
STORE   TEMP_REG, INDEX
ADD     INDEX, ONE
LOAD    TEMP_REG, ASC_T
STORE   TEMP_REG, INDEX
ADD     INDEX, ONE
LOAD    TEMP_REG, ASC_O
STORE   TEMP_REG, INDEX
ADD     INDEX, ONE
LOAD    TEMP_REG, ASC_W
STORE   TEMP_REG, INDEX
ADD     INDEX, ONE
LOAD    TEMP_REG, ASC_N
STORE   TEMP_REG, INDEX
ADD     INDEX, ONE
LOAD    TEMP_REG, ASC_SP
STORE   TEMP_REG, INDEX
ADD     INDEX, ONE
LOAD    TEMP_REG, ASC_I
STORE   TEMP_REG, INDEX
ADD     INDEX, ONE
LOAD    TEMP_REG, ASC_S
STORE   TEMP_REG, INDEX
ADD     INDEX, ONE
LOAD    TEMP_REG, ASC_SP
STORE   TEMP_REG, INDEX
ADD     INDEX, ONE
LOAD    TEMP_REG, ASC_B
STORE   TEMP_REG, INDEX
ADD     INDEX, ONE
LOAD    TEMP_REG, ASC_A
STORE   TEMP_REG, INDEX
ADD     INDEX, ONE
```

Chip Specification

82 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

```

LOAD    TEMP_REG, ASC_N
STORE   TEMP_REG, INDEX
ADD     INDEX, ONE
LOAD    TEMP_REG, ASC_G
STORE   TEMP_REG, INDEX
ADD     INDEX, ONE
LOAD    TEMP_REG, ASC_K
STORE   TEMP_REG, INDEX
ADD     INDEX, ONE
LOAD    TEMP_REG, ASC_O
STORE   TEMP_REG, INDEX
ADD     INDEX, ONE
LOAD    TEMP_REG, ASC_K
STORE   TEMP_REG, INDEX
ADD     INDEX, ONE
LOAD    TEMP_REG, ASC_CR
STORE   TEMP_REG, INDEX
ADD     INDEX, ONE
LOAD    TEMP_REG, ASC_LF
STORE   TEMP_REG, INDEX
ADD     INDEX, ONE      ; This INDEX should be CRLF_BEGIN

```

RETURN

```

;=====
; Subroutine: BLINK_LED
;     Blinking LEDs while idling the main loop
;
; Procedure:
;     - Decrementing counters each time the routine is called.
;     - 0xFFFF * 0x0019 provides 60HZ for LED to visibly blinking.
;     - Once the both counter reach zeros, assigning new LED
;       values and then outputs the value through LED_PORT(0002).
;
; input registers: none
; output registers: none
;=====
ADDRESS 0290
BLINK_LED
LOAD    DELAY, FFFF
LOAD    DELAY2, 0019
TAG
SUB     DELAY, ONE
JUMPNZ TAG
LOAD    DELAY, FFFF
SUB     DELAY2, ONE
JUMPNZ TAG

XOR     LED, 00FF
OUTPUT LED, LED_PORT
RETURN
;=====
; Routine: BIN_TO_ASCII
;
; function: converting a 16-bit number into ASCII.
;     Dividing input CECS_COUNT by TEN_THOUSAND, ONE_THOUSAND,
;     ONE_HUNDREDS, TEN, and ONE then stores the
;     division at scratch-pad address 0011, 0012, 0013,
;     0014, and 0015 consequently.
; pseudo code:

```

Chip Specification

83 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

```
;
;   HOW_MANY = 0
;   RESULT = COUNT;
;   while(RESULT-10 > 0){
;       RESULT -= 10;
;       HOW_MANY++;
;   }
;
;   Input Register:
;       COUNT - number of character received
;               from serial terminal.
;
;   Output Registers:
;       COUNT_TEN - stores ASCII of MSB of COUNT
;       COUNT_ONE - stores ASCII of LSB of COUNT
;=====

; Address for BIN_TO_ASC
;       ADDRESS 0310

BIN_TO_ASCII
;       LOAD     HOW_MANY, ZEROS
;       LOAD     RESULT, COUNT

SUB_FIND
;       SUB      RESULT, TEN
;       JUMPC    DONE_FIND
;       ADD      HOW_MANY, ONE
;       JUMP     SUB_FIND

DONE_FIND
;       ADD      RESULT, TEN
;       ADD      HOW_MANY, ASCII
;       STORE    HOW_MANY, COUNT_TEN

;       ADD      RESULT, ASCII
;       STORE    RESULT, COUNT_ONE

;       RETURN

;=====
;   Subroutine: TX_FUNC
;
;   function: Output a character to TramelBlaze through
;   DATA_PORT(0000) according to the current PRINT_FLAG.
;   If the PRINT_FLAG is zeros, UART is not transmitting
;   at the moment.
;
;   Procedure:
;       if(PRINT_FLAG == 0) return;
;       else{
;           printf(Mem[index]);
;           switch(PRINT_FLAG){
;               case 0006: DISPLAY_COUNT();
;               case 0005: DISPLAY_BSSP();
;               case 0004: DISPLAY_BANNER();
;               case 0003: DISPLAY_PROMPT();
;               case 0002: DISPLAY_HOMETOWN();
;               case 0001: DISPLAY_CRLF();
;           }
;       }
;
;
;
```

Chip Specification

84 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
-------------------------------------	--------------------------------	---	-----------------------

```
;   Input register:
;   PRINT_FLAG - hold current printing mode.
;
;=====
TX_FUNC      ADDRESS 0330

              COMP     PRINT_FLAG, ZEROS
              RETURNZ

              FETCH    TEMP_REG, INDEX
              OUTPUT   TEMP_REG, DATA_PORT
              ADD       INDEX, ONE

              COMP     PRINT_FLAG, SIX
              JUMPZ    DISPLAY_COUNT
              COMP     PRINT_FLAG, FIVE
              JUMPZ    DISPLAY_BSSP
              COMP     PRINT_FLAG, FOUR
              JUMPZ    DISPLAY_BANNER
              COMP     PRINT_FLAG, THREE
              JUMPZ    DISPLAY_PROMPT
              COMP     PRINT_FLAG, TWO
              JUMPZ    DISPLAY_HOMETOWN
              COMP     PRINT_FLAG, ONE
              JUMPZ    DISPLAY_CRLF

              RETURN

DISPLAY_COUNT
              COMP     INDEX, COUNT_END
              RETURNC
              LOAD     INDEX, CRLF_BEGIN
              LOAD     PRINT_FLAG, ONE
              RETURN

DISPLAY_BSSP
              COMP     INDEX, BSSP_END
              RETURNC
              LOAD     PRINT_FLAG, ZEROS
              RETURN

DISPLAY_BANNER
              COMP     INDEX, BANNER_END
              RETURNC
              LOAD     PRINT_FLAG, THREE
              RETURN

DISPLAY_PROMPT
              COMP     INDEX, PROMPT_END
              RETURNC
              LOAD     PRINT_FLAG, ZEROS
              RETURN

DISPLAY_HOMETOWN
              COMP     INDEX, HOMETOWN_END
              RETURNC
              LOAD     INDEX, PROMPT_BEGIN
              LOAD     PRINT_FLAG, THREE
              RETURN
```

Chip Specification

85 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

DISPLAY_CRLF

```
COMP    INDEX, CRLF_END
RETURNC
LOAD    INDEX, PROMPT_BEGIN
LOAD    PRINT_FLAG, THREE
RETURN
```

```
;=====
;  Subroutine: RX_FUNC
;
;  function: Setting the PRINT_FLAG and INDEX to be ready
;             to display on the Serial Terminal on the next
;             TX interrupt according to the received
;             character input.
;
;  Procedure:
;      if (PRINT_FLAG > 0) return;
;      else{
;          TEMP_REG = get_input();
;
;          switch (TEMP_REG){
;              case '*' : SET_HOMETOWN();
;              case '<cr>': SET_CRLF();
;              case '<bs>': SET_BSSP();
;              case '@'  : SET_AT();
;              default   : SET_ECHO();
;          }
;      }
;
;=====
```

```
ADDRESS 0400
RX_FUNC
COMP    PRINT_FLAG, ZEROS
RETURNNZ

INPUT    TEMP_REG, DATA_PORT
COMP    TEMP_REG, ZEROS
RETURNZ

COMP    TEMP_REG, ASC_STAR
JUMPZ   SET_HOMETOWN

COMP    TEMP_REG, ASC_CR
JUMPZ   SET_CRLF

COMP    TEMP_REG, ASC_BS
JUMPZ   SET_BSSP

COMP    TEMP_REG, ASC_AT
JUMPZ   SET_AT

SET_ECHO
COMP    COUNT, MAX_CHAR
JUMPZ   SET_CRLF
ADD     COUNT, ONE
OUTPUT  TEMP_REG, DATA_PORT

RETURN
```

Chip Specification

86 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
-------------------------------------	--------------------------------	---	-----------------------

SET_HOMETOWN

```

LOAD    PRINT_FLAG, TWO
LOAD    INDEX, HOMETOWN_BEGIN
LOAD    TEMP_REG, ASC_NULL
OUTPUT  TEMP_REG, DATA_PORT
LOAD    COUNT, ZEROS
RETURN

```

SET_CRLF

```

LOAD    PRINT_FLAG, ONE
LOAD    INDEX, CRLF_BEGIN
LOAD    TEMP_REG, ASC_NULL
OUTPUT  TEMP_REG, DATA_PORT
LOAD    COUNT, ZEROS
RETURN

```

SET_BSSP

```

COMP    COUNT, ZEROS
RETURNZ
LOAD    PRINT_FLAG, FIVE
LOAD    INDEX, BSSP_BEGIN
LOAD    TEMP_REG, ASC_NULL
OUTPUT  TEMP_REG, DATA_PORT
SUB    COUNT, ONE
RETURN

```

SET_AT

```

CALL   BIN_TO_ASCII
LOAD    PRINT_FLAG, SIX
LOAD    INDEX, COUNT_BEGIN
LOAD    TEMP_REG, ASC_NULL
OUTPUT  TEMP_REG, DATA_PORT
LOAD    COUNT, ZEROS
RETURN

```

```

;=====
; Sub Routine: Interrupt Service Routine(ISR)
;   Checking UART STATUS if the UART is ready to
;   transmit or receive a character
;   If the Transmit Engine is ready, (UART_STATUS & 0x0002) == 1
;   If the Receive Engine is ready, (UART_STATUS & 0x0001) == 1
;   Where UART_STATUS is an 8-bit data from UART data
;   read from STATUS_PORT(0001)
;   UART_STATUS consists
;   {3'b0, w_ovf, w_ferr, w_perr, w_txdy, w_rxdy};
;   which ovf - overflow flag
;           ferr - framing error
;           perr - parity bit error
;   * txdy - transmit ready
;   * rxdy - receive ready
;
; Procedure:
;   UART_STATUS = INPUT[STATUS_PORT];
;   UART_STATUS &= 0x0003;
;
;   if(UART_STATUS = 0x0003) CALL_BOTH(); else
;   if(UART_STATUS = 0x0002) TX_FUNC();   else
;   if(UART_STATUS = 0x0001) RX_FUNC();
;   else

```

Chip Specification

87 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

```
;          return;
;
;  input registers: none
;      UART_STATUS - Reading the current UART status from UART;
;                  then store it for if-statement.
;  output registers: none
;
;=====
                ADDRESS 0E00
ISR
    INPUT      UART_STATUS, STATUS_PORT
    AND        UART_STATUS, THREE          ; check TX_rdy & RX_rdy

    COMP      UART_STATUS, THREE
    JUMPZ     CALL_BOTH

    COMP      UART_STATUS, TWO
    CALLZ     TX_FUNC                      ; TX is ready

    COMP      UART_STATUS, ONE
    CALLZ     RX_FUNC                      ; RX is ready

    RETEN

CALL_BOTH
    CALL      TX_FUNC                      ; prioritize to TX
    CALL      RX_FUNC
    RETEN

;=====
;  ISR vectored through 0FFE
;  Jump to ISR routine
;=====

                ADDRESS 0FFE

ENDIT

    JUMP     ISR

    END
```

Chip Specification

88 of 88

Prepared by: S. Chanartip	Date: APRIL 11, 2018	Document Number and Filename UART Specification	Revision: 2
------------------------------	-------------------------	--	----------------

Intentionally
left blank.