Term Project: UART Design

CEG3155 – Digital System II

Fall 2023 School of Electrical Engineering and Computer Science

University of Ottawa

Wednesday Afternoon Group #32

Qingyun Yang, 300220139

Kuan-Yu Chang, 300201058

Professor Rami Abielmona

CEG3155 Lab

December 6, 2023

Objectives

Throughout this term project, students are implementing a complete UART in VHDL. Students must be able to design, realize and test transmitter, receiver and baud rate generator to build up a UART. Demonstrate a complete understanding for the design of a UART and its interface in a real-time system. UART, stands for *Universal Asynchronous Receiver-Transmitter*, which is used to communicate between two devices. Usually for providing information using minimum number of wires, the data would be transmitted serially bit by bit without a clock signal. UART contains two main component, transmitter and receiver, students are mainly focusing on building the transmitter part for transmitting the previous lab's error signal. This term project is based on previous lab, which is a traffic light controller that controls two street, the main street, and the side street. Using UART would make student understand the debug message which is transmitted by the Altera board into the computer and translated into readable vocabulary.

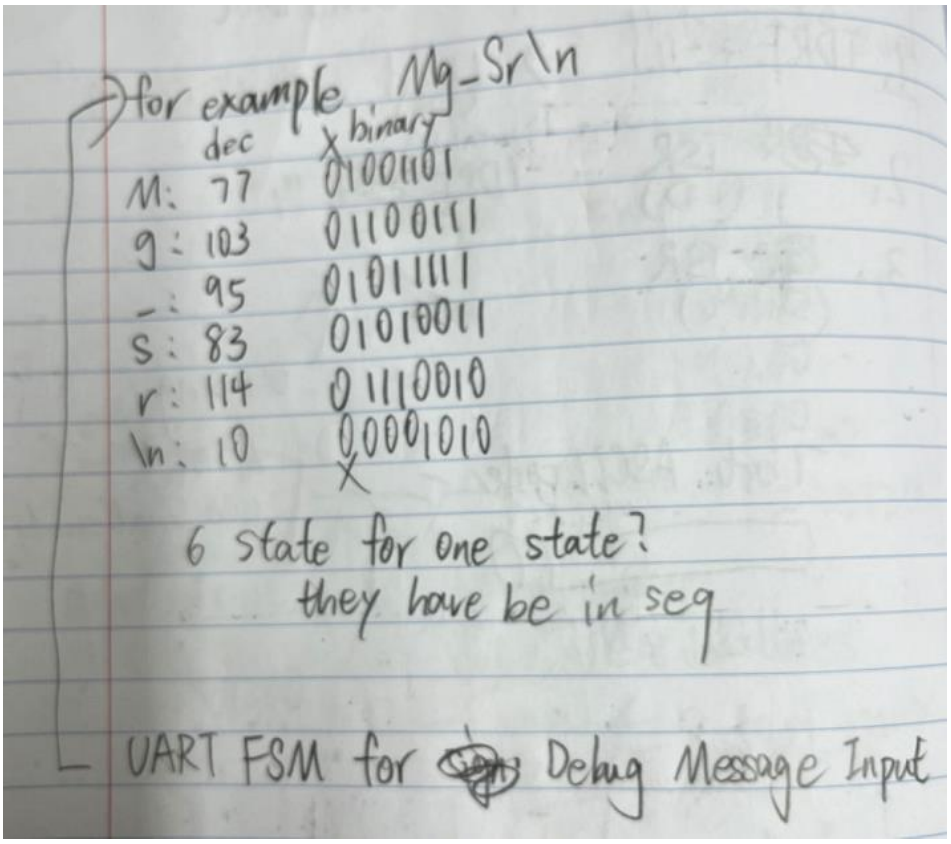
Equipment and components used:

- Quartus II (student edition or web edition)
- Altera DE2-115 board with
  - USB-blaster cable
  - Power supply 12 VDC, 2A

Part I – Design Specifications

Part 1-1 ASCII Code for Data Encoding

In the first part of the lab, students should understand how the debug message is transmitted to the computer screen. Since, the data is transferred by bit serially, this means that we are using ASCII code for data encoding. Data encoding contains nine bits serially transfer by UART one by a time, starting with the start bit which would set the data line to low for a period. One by a time from the last bit 7 bits would be transfer to the computer screen, then the last bit would be transfer as low since for ASCII code students use all have a common 0 at as the first bit. Then set high for stopping the data transfer.



for example Mg-Sr\n

	dec	binary
M:	77	01001101
g:	103	01100111
_:	95	01011111
S:	83	01010011
r:	114	01110010
\n:	10	00001010

6 state for one state?

they have be in seq

UART FSM for ~~[illegible]~~ Debug Message Input

Figure 1.1 Example for ASCII Code Message

Part 1-2 Address Decoder

The second part of the design would be the Address decoder. UART is connected to a computer or microcontroller so that we can read and write characters from/to the outside world. Student would need to design an address decoder to choose which actions should we execute, and which register is transmitting/receiving message. Since students are not doing the part of receiver, the decoder would be simplified into four states only. Address decoder is construct by a 4 to 1 multiplexer and a 2 to 1 multiplexer. The 4 to 1 multiplexer controls if SCCR, SCSR and RDR when should anyone of the data should be transferred onto the data bus. Data bus moving out the 4 to 1 multiplexer connects into the 2 to 1 multiplexer input with another input from outside. This multiplexer controls which data should be write onto the common data bus.

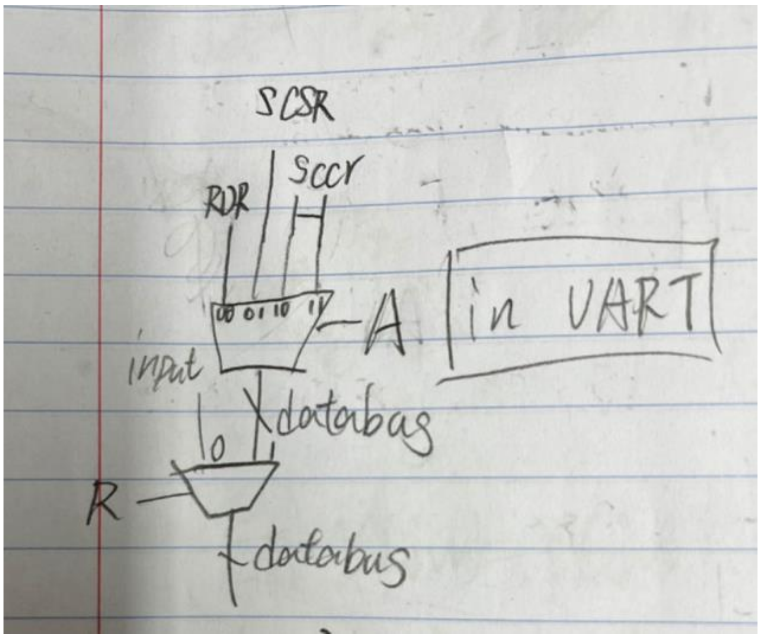


Figure 1.2 Design of Address Decoder

Part 1-3 Transmitter

The third part of the design would be the transmitter, which consists of the TDR, TSR and a TSR control. TDR is a simple 8 bit register which stores an 8-bit data written from data bus into the register. TSR is a 9 bit shift right register which shifts out a bit at a time serially and a 1 would be serial into the register for being the stopping bit every shift. When TSR control receives a 0 from TDRE in SCSR, the TSR control starts to order TSR to transmit the debug message serially and setting the TDRE back to SCSR as a 1.

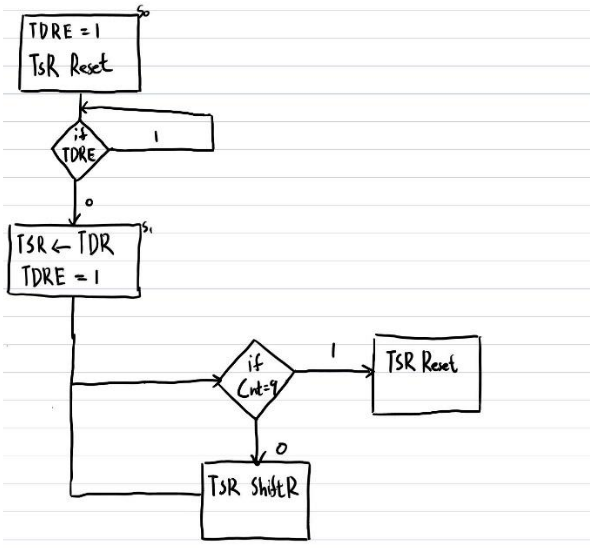


Figure 1.3 ASM Chart for TSR Control

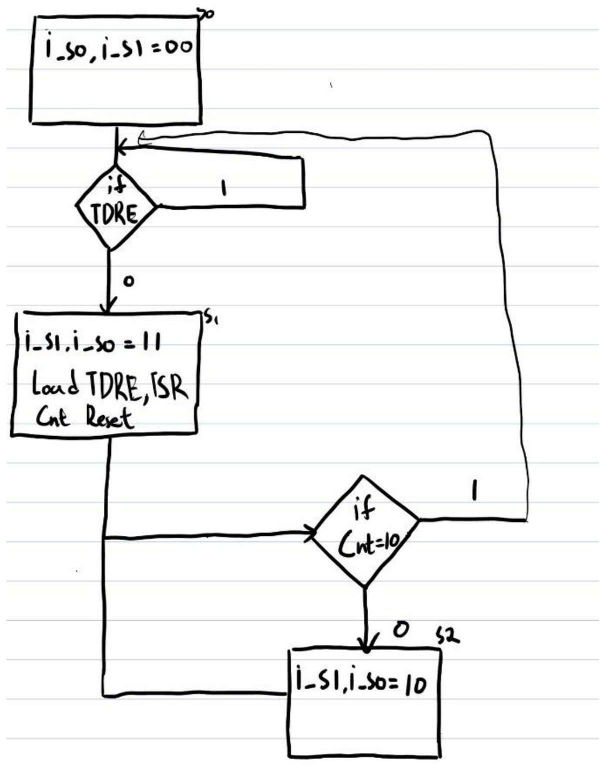


Figure 1.4 Detailed ASM Chart for TSR Control

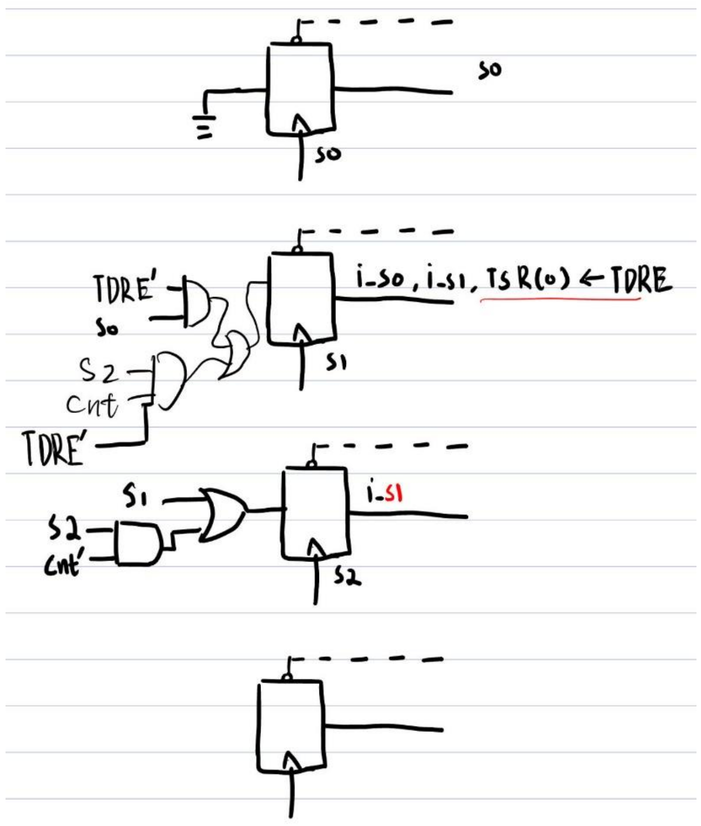


Figure 1.5 Control Path for TSR Control

Part 1-4 Baud Rate Generator

The fourth part of the UART is the Baud Rate Generator. The function of the baud rate generator is to convert the clock frequency into a specific range, allowing common devices to communicate with each other using standard frequencies. It achieves this by outputting the clock frequency every 162 clock cycles, resulting in an output clock rate ranging from 38,400 to 300 by using additional multiplier techniques.

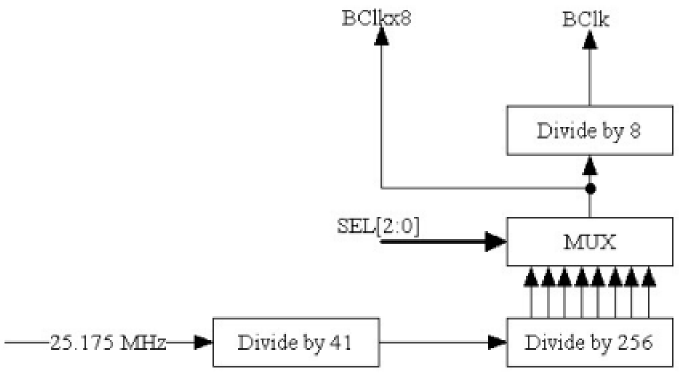


Figure1.6 Baud Rate Generator Component

Part 1-5 UART Data Path

The fifth part of the UART is to connect all the part of registers into a data path, which contains TDR,TSR,SCSR, Baud rate generator in the UART and the traffic light controller is connect with UART and UART FSM.

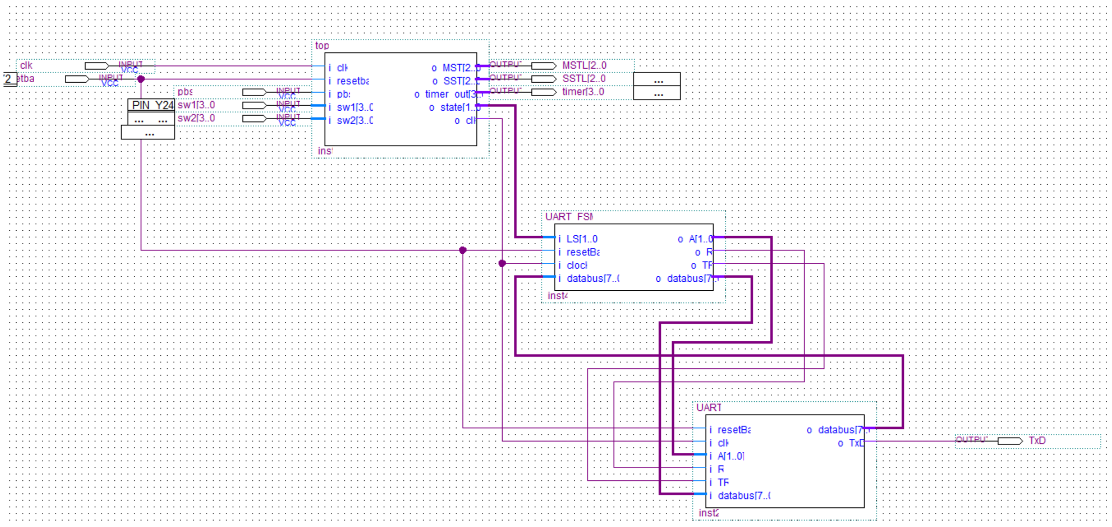


Figure 1.6 UART Data Path with Traffic Light Controller

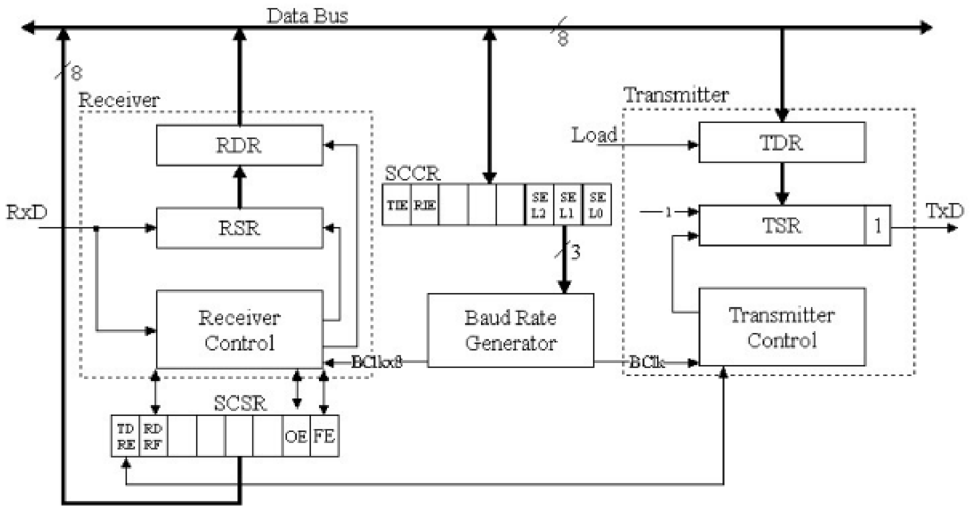


Figure 1.7 UART Detail Components

Part 1-6 UART FSM

The last part of the UART would be the UART FSM. Which is used for receiving message from the UART to decide to invert the debug message or not. There are 48 states in our FSM design. There are 4 different types of debug messages representing the first 4 state, each state has 6 characters(5charactor + return), and in each state it needs to check TDRE to see if the UART is ready or not, so 4X6X2 equals 48 state int the FSM.

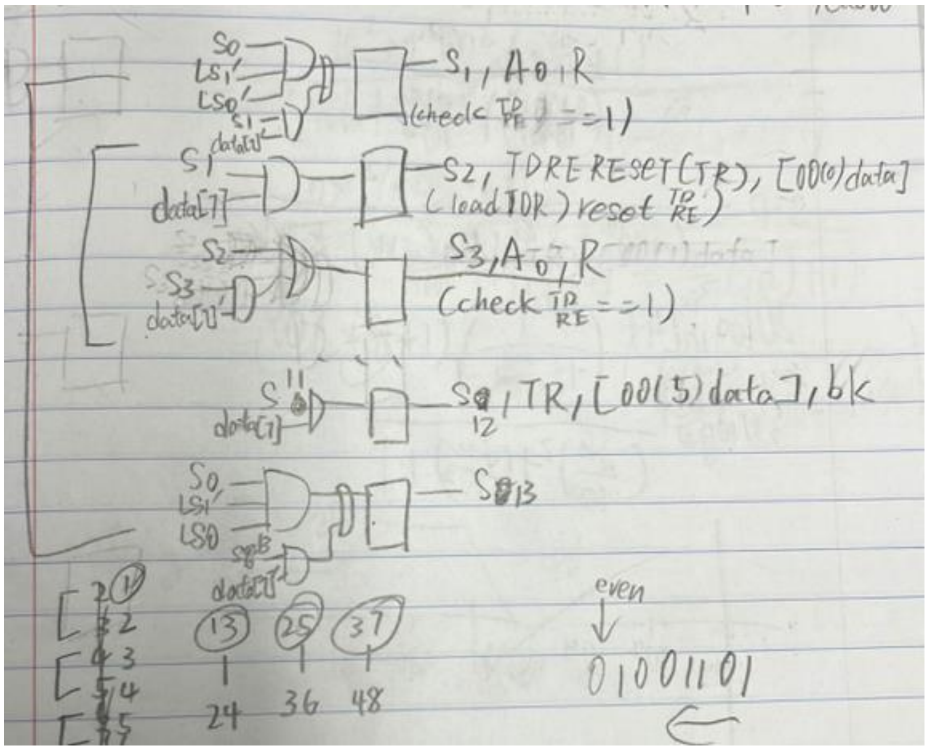


Figure 1.8 UART FSM Design Controller

Part II – Code Implementation

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY UART IS
	PORT(
		i_resetBar, i_clk:IN STD_LOGIC;
		i_A:IN STD_LOGIC_VECTOR(1 downto 0);
		i_R,i_TR,i_TS:IN  STD_LOGIC;
		i_databus:IN  STD_LOGIC_VECTOR(7 downto 0);
		o_databus:OUT STD_LOGIC_VECTOR(7 downto 0);
		o_TxD: OUT STD_LOGIC;
		   o_TSR:OUT STD_LOGIC_VECTOR(8 downto 0);
		   o_TDR:OUT STD_LOGIC_VECTOR(7 downto 0);
		   o_SCSR:OUT STD_LOGIC_VECTOR(7 downto 0);
		   d_s:OUT std_logic_vector(2  downto 0);
		   o_s1,o_s0:OUT STD_LOGIC;
		   di_SCSRLoad:IN STD_LOGIC;
		   d_cnta:OUT std_logic_vector(3  downto 0)
	);
END ENTITY;

ARCHITECTURE rtl of UART IS

	SIGNAL                    int_databus,int_SCSR,int_SCCR,int_mux4,int_TDR:
STD_LOGIC_VECTOR(7 downto 0);
	SIGNAL
int_TS,int_s0,int_s1,int_loadSCCR,int_loadTDR,i_clock,int_SCSRLoad:
STD_LOGIC;

COMPONENT mux4to1_8 IS
	PORT(
		i_S:IN STD_LOGIC_VECTOR(1 downto 0);
		i_I3,i_I2,i_I1,i_I0:IN  STD_LOGIC_VECTOR(7 downto 0);
		o_O:OUT STD_LOGIC_VECTOR(7 downto 0));
END COMPONENT;
```

```
COMPONENT mux2to1_8 IS
      PORT(
            i_S:IN STD_LOGIC;
            i_a,i_b:IN STD_LOGIC_VECTOR(7 downto 0);
            o_O:OUT STD_LOGIC_VECTOR(7 downto 0));
END COMPONENT;

COMPONENT SCSR IS
      PORT(
            i_resetBar, i_load,i_TR,i_TS: IN STD_LOGIC;
            i_clock: IN STD_LOGIC;
            i_Value: IN STD_LOGIC_VECTOR(7 downto 0);
            o_Value: OUT STD_LOGIC_VECTOR(7 downto 0));
END COMPONENT;

COMPONENT SCCR IS
      PORT(
            i_resetBar, i_load: IN STD_LOGIC;
            i_clock: IN STD_LOGIC;
            i_Value: IN STD_LOGIC_VECTOR(7 downto 0);
            o_Sel:      OUT STD_LOGIC_VECTOR(2 downto 0);
            o_Value: OUT STD_LOGIC_VECTOR(7 downto 0));
END COMPONENT;

COMPONENT BaudRateGenerator is
      Port ( i_clk : in STD_LOGIC;
                 i_reset : in STD_LOGIC;
                 i_SEL : in STD_LOGIC_VECTOR (2 downto 0);
                 o_BaudRate : out STD_LOGIC);
end COMPONENT;

COMPONENT TDR IS
      PORT(
            i_resetBar, i_load: IN STD_LOGIC;
            i_clock: IN STD_LOGIC;
            i_Value: IN STD_LOGIC_VECTOR(7 downto 0);
            o_Value: OUT STD_LOGIC_VECTOR(7 downto 0));
END COMPONENT;
```

```
COMPONENT TSR IS
	PORT(
		i_S0, i_S1 : IN STD_LOGIC;
		CLK : IN STD_LOGIC;
		RST : IN STD_LOGIC;
		i_I : IN STD_LOGIC_VECTOR(8 DOWNTO 0);
		o_O : OUT STD_LOGIC;
		   o_PO: OUT STD_LOGIC_VECTOR(8 DOWNTO 0)
	);
END COMPONENT;

COMPONENT tsrControl IS
	PORT(
		clk : IN std_logic;
		reset : IN std_logic;
		TDRE : IN std_logic;
		TDRE_set: OUT std_logic;
		   d_s:OUT std_logic_vector(2  downto 0);
		i_s0,i_s1:  OUT std_logic
	);
END COMPONENT;

BEGIN

int_loadTDR<= not i_A(0) and not i_A(1)and not i_R;

int_SCSRLoad<=di_SCSRLoad;

--start of address decorder
mux_2: mux2to1_8
	PORT MAP(
		i_S => i_R,
		i_a => i_databus,
		i_b => int_mux4,
		o_O => int_databus);

mux_4: mux4to1_8
```

```
    PORT MAP(
        i_S => i_A,
        i_I3 => int_SCCR,
        i_I2 => int_SCCR,
        i_I1 => int_SCSR,
        i_I0 => "00000000",--no RDR, 0 as default
        o_O => int_mux4);
--end of address decorder

c_SCSR: SCSR
    PORT MAP(i_resetBar => i_resetBar,
                i_load => int_TS,
                i_TR => i_TR,
                i_TS => int_TS or i_TS,
                i_clock => i_clk,
                i_Value => int_databus,
                o_Value => int_SCSR);

c_SCCR: SCCR
    PORT MAP(i_resetBar => i_resetBar,
                i_load => int_loadSCCR,
                i_clock => i_clk,
                i_Value => int_databus,
                o_Value => int_SCCR);

c_TDR: TDR
    PORT MAP(i_resetBar => i_resetBar,
                i_load => int_loadTDR,
                i_clock => i_clock,
                i_Value => int_databus,
                o_Value => int_TDR);

c_TSR: TSR
    PORT MAP(
        i_S0 => int_s0,
        i_S1 => int_s1,
        CLK => i_clock,
        RST => i_resetBar,
        i_I => int_TDR & '0',
        o_O => o_TxD,
           o_PO => o_TSR
    );

c_tsrControl: tsrControl
    PORT MAP(
        clk => i_clock,
        reset => i_resetBar,
        TDRE => int_SCSR(7),
        TDRE_set => int_TS,
           d_s=>d_s,
        i_s0 => int_s0,
        i_s1 => int_s1
    );

c_BRG: BaudRateGenerator
    Port Map ( i_clk => i_clk,
                     i_reset => not i_resetBar,
                             o_BaudRate => i_clock,
                     i_SEL => int_SCCR(2 downto 0)
                             );



    --i_clock<=i_clk;

    o_databus <= int_databus;
      o_TDR<= int_TDR;
      o_SCSR<=int_SCSR;
      o_s1<=int_s1;
      o_s0<=int_s0;

END ARCHITECTURE;
```

Figure 2.1 UART Implementation in VHDL

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
```

```
ENTITY UART_FSM IS
    PORT(
        i_LS:IN STD_LOGIC_VECTOR(1 downto 0);
        i_resetBar, i_clock:IN    STD_LOGIC;
        o_A:OUT STD_LOGIC_VECTOR(1 downto 0);
        o_R,o_TR:OUT STD_LOGIC;
        i_databus:IN  STD_LOGIC_VECTOR(7 downto 0);
        o_databus:OUT STD_LOGIC_VECTOR(7 downto 0)
    );
END ENTITY;

ARCHITECTURE rtl of UART_FSM IS
    SIGNAL int_S,int_A,int_B: STD_LOGIC_VECTOR(48 downto 0);
    SIGNAL int_bk:STD_LOGIC;

    COMPONENT enARdFF_2
        PORT(
            i_resetBar      : IN  STD_LOGIC;
            i_d         : IN  STD_LOGIC;
            i_enable    : IN  STD_LOGIC;
            i_clock         : IN  STD_LOGIC;
            o_q, o_qBar     : OUT     STD_LOGIC);
    END COMPONENT;

    COMPONENT enARdFF_2_rr
        PORT(
            i_resetBar      : IN  STD_LOGIC;
            i_d         : IN  STD_LOGIC;
            i_enable    : IN  STD_LOGIC;
            i_clock         : IN  STD_LOGIC;
            o_q, o_qBar     : OUT     STD_LOGIC);
    END COMPONENT;

BEGIN

    int_A(2)<=int_S(1)  and i_databus(7);
    int_A(4)<=int_S(3)  and i_databus(7);
    int_A(6)<=int_S(5)  and i_databus(7);
    int_A(8) <= int_S(7)  and i_databus(7);
    int_A(10) <= int_S(9)  and i_databus(7);
    int_A(12) <= int_S(11)  and i_databus(7);
    int_A(14) <= int_S(13)  and i_databus(7);
    int_A(16) <= int_S(15)  and i_databus(7);
    int_A(18) <= int_S(17)  and i_databus(7);
    int_A(20) <= int_S(19)  and i_databus(7);
    int_A(22) <= int_S(21)  and i_databus(7);
    int_A(24) <= int_S(23)  and i_databus(7);
    int_A(26) <= int_S(25)  and i_databus(7);
    int_A(28) <= int_S(27)  and i_databus(7);
    int_A(30) <= int_S(29)  and i_databus(7);
    int_A(32) <= int_S(31)  and i_databus(7);
    int_A(34) <= int_S(33)  and i_databus(7);
    int_A(36) <= int_S(35)  and i_databus(7);
    int_A(38) <= int_S(37)  and i_databus(7);
    int_A(40) <= int_S(39)  and i_databus(7);
    int_A(42) <= int_S(41)  and i_databus(7);
    int_A(44) <= int_S(43)  and i_databus(7);
    int_A(46) <= int_S(45)  and i_databus(7);
    int_A(48) <= int_S(47)  and i_databus(7);

    int_A(3)<=int_S(2)  or (int_S(3)  and not i_databus(7));
    int_A(5)<=int_S(4)  or (int_S(5)  and not i_databus(7));
    int_A(7)<=int_S(6)  or (int_S(7)  and not i_databus(7));
    int_A(9) <= int_S(8)  or (int_S(9)  and not i_databus(7));
    int_A(11)  <= int_S(10) or (int_S(11)  and not i_databus(7));
    int_A(13) <= int_S(12)  or (int_S(13)  and not i_databus(7));
    int_A(15) <= int_S(14)  or (int_S(15)  and not i_databus(7));
    int_A(17) <= int_S(16)  or (int_S(17)  and not i_databus(7));
    int_A(19) <= int_S(18)  or (int_S(19)  and not i_databus(7));
    int_A(21) <= int_S(20)  or (int_S(21)  and not i_databus(7));
    int_A(23) <= int_S(22)  or (int_S(23)  and not i_databus(7));
    int_A(25) <= int_S(24)  or (int_S(25)  and not i_databus(7));
    int_A(27) <= int_S(26)  or (int_S(27)  and not i_databus(7));
    int_A(29) <= int_S(28)  or (int_S(29)  and not i_databus(7));
    int_A(31) <= int_S(30)  or (int_S(31)  and not i_databus(7));
    int_A(33) <= int_S(32)  or (int_S(33)  and not i_databus(7));
```

```
    int_A(35) <= int_S(34) or (int_S(35) and not i_databus(7));
    int_A(37) <= int_S(36) or (int_S(37) and not i_databus(7));
    int_A(39) <= int_S(38) or (int_S(39) and not i_databus(7));
    int_A(41) <= int_S(40) or (int_S(41) and not i_databus(7));
    int_A(43) <= int_S(42) or (int_S(43) and not i_databus(7));
    int_A(45) <= int_S(44) or (int_S(45) and not i_databus(7));
    int_A(47) <= int_S(46) or (int_S(47) and not i_databus(7));

    int_bk <= int_S(12) or int_S(24) or int_S(36) or int_S(48);

--special states
--state S0
regS0: enARdFF_2_rr
    PORT MAP(i_resetBar => i_resetBar,
             i_d => int_bk,
             i_enable => '1',
             i_clock => i_clock,
             o_q => int_S(0));

--state S1
regS1: enARdFF_2
    PORT MAP(i_resetBar => i_resetBar,
             i_d => (int_S(0) and not i_LS(1) and not i_LS(0)) or (int_S(1) and not
i_databus(7)),
             i_enable => '1',
             i_clock => i_clock,
             o_q => int_S(1));

--state S13
regS13: enARdFF_2
    PORT MAP(i_resetBar => i_resetBar,
             i_d => (int_S(0) and not i_LS(1) and i_LS(0)) or (int_S(13) and not
i_databus(7)),
             i_enable => '1',
             i_clock => i_clock,
             o_q => int_S(13));

--state S25
```

```
regS25: enARdFF_2
        PORT MAP(i_resetBar => i_resetBar,
                  i_d => (int_S(0) and i_LS(1) and not i_LS(0)) or (int_S(25) and not
i_databus(7)),
                  i_enable => '1',
                  i_clock => i_clock,
                  o_q => int_S(25));

--state S37
regS37: enARdFF_2
        PORT MAP(i_resetBar => i_resetBar,
                  i_d => (int_S(0) and i_LS(1) and i_LS(0)) or (int_S(37) and not
i_databus(7)),
                  i_enable => '1',
                  i_clock => i_clock,
                  o_q => int_S(37));



--common states
loop1: FOR i IN 12 downto 2 GENERATE
                reg: enARdFF_2
                PORT MAP(i_resetBar => i_resetBar,
                        i_d => int_A(i),
                        i_enable => '1',
                        i_clock => i_clock,
                        o_q => int_S(i));
END GENERATE;

loop2: FOR i IN 24 downto 14 GENERATE

                reg: enARdFF_2
                PORT MAP(i_resetBar => i_resetBar,
                        i_d => int_A(i),
                        i_enable => '1',
                        i_clock => i_clock,
                        o_q => int_S(i));
```

```
END GENERATE;

loop3: FOR i IN 36 downto 26 GENERATE
            reg: enARdFF_2
            PORT MAP(i_resetBar => i_resetBar,
                         i_d => int_A(i),
                         i_enable => '1',
                         i_clock => i_clock,
                         o_q => int_S(i) );

END GENERATE;

loop4: FOR i IN 48 downto 38 GENERATE

            reg: enARdFF_2
            PORT MAP(i_resetBar => i_resetBar,
                         i_d => int_A(i),
                         i_enable => '1',
                         i_clock => i_clock,
                         o_q => int_S(i));

END GENERATE;

      --common state control
      o_A(0) <= int_S(1) or int_S(3) or int_S(5) or int_S(7) or int_S(9) or
               int_S(11) or int_S(13) or int_S(15) or int_S(17) or int_S(19) or
               int_S(21) or int_S(23) or int_S(25) or int_S(27) or int_S(29) or
               int_S(31) or int_S(33) or int_S(35) or int_S(37) or int_S(39) or
               int_S(41) or int_S(43) or int_S(45) or int_S(47);
      o_A(1) <= '0';
      o_R <= int_S(1) or int_S(3) or int_S(5) or int_S(7) or int_S(9) or
               int_S(11) or int_S(13) or int_S(15) or int_S(17) or int_S(19) or
               int_S(21) or int_S(23) or int_S(25) or int_S(27) or int_S(29) or
               int_S(31) or int_S(33) or int_S(35) or int_S(37) or int_S(39) or
               int_S(41) or int_S(43) or int_S(45) or int_S(47);
      o_TR <= int_S(1) or int_S(3) or int_S(5) or int_S(7) or int_S(9) or
               int_S(11) or int_S(13) or int_S(15) or int_S(17) or int_S(19) or
               int_S(21) or int_S(23) or int_S(25) or int_S(27) or int_S(29) or
            int_S(31) or int_S(33) or int_S(35) or int_S(37) or int_S(39) or
            int_S(41) or int_S(43) or int_S(45) or int_S(47);

    o_databus(7) <= int_S(4) or int_S(16) or int_S(34) or int_S(46);
    o_databus(6) <= int_S(2) or int_S(4) or int_S(6) or int_S(8) or int_S(10) or
                    int_S(14) or int_S(16) or int_S(18) or int_S(20) or int_S(22)
or
                    int_S(26) or int_S(28) or int_S(30) or int_S(32) or int_S(34)
or
                    int_S(38) or int_S(40) or int_S(42) or int_S(44) or int_S(46);
    o_databus(5) <= int_S(4) or int_S(10) or int_S(16) or int_S(22) or int_S(28) or
                    int_S(34) or int_S(40) or int_S(46);
    o_databus(4) <= int_S(6) or int_S(8) or int_S(10) or int_S(16) or int_S(18) or
                    int_S(20) or int_S(22) or int_S(28) or int_S(30) or int_S(32)
or
                    int_S(40) or int_S(42) or int_S(44) or int_S(46);
    o_databus(3) <= int_S(2) or int_S(6) or int_S(12) or int_S(14) or int_S(16) or
                    int_S(18) or int_S(24) or int_S(26) or int_S(30) or int_S(36)
or
                    int_S(38) or int_S(42) or int_S(46) or int_S(48);
    o_databus(2) <= int_S(2) or int_S(4) or int_S(6) or int_S(14) or int_S(18) or
                    int_S(26) or int_S(30) or int_S(34) or int_S(38) or int_S(42);
    o_databus(1) <= int_S(4) or int_S(6) or int_S(8) or int_S(10) or int_S(12) or
                    int_S(18) or int_S(20) or int_S(22) or int_S(24) or int_S(28)
or
                    int_S(30) or int_S(32) or int_S(34) or int_S(36) or int_S(40)
or
                    int_S(42) or int_S(44) or int_S(48);
    o_databus(0) <= int_S(2) or int_S(4) or int_S(6) or int_S(8) or
            int_S(14) or int_S(16) or int_S(18) or
            int_S(20) or int_S(26)or
            int_S(30) or int_S(32) or int_S(34) or int_S(38) or
            int_S(42) or int_S(44) or int_S(46);

END ARCHITECTURE;
```

Figure 2.2 UART FSM Iplementation in VHDL

```
LIBRARY ieee;
```

```
USE ieee.std_logic_1164.ALL;

ENTITY top IS
    PORT(
        i_clk,i_resetbar,i_pbs:  IN STD_LOGIC;
        i_sw1,i_sw2:IN STD_LOGIC_VECTOR(3 downto 0);
        o_MST,o_SST: OUT     STD_LOGIC_VECTOR(2 downto 0);
        o_timer_out:  OUT  STD_LOGIC_VECTOR(3 downto 0);
        o_state              : OUT     STD_LOGIC_VECTOR(1 downto 0);
        o_clk                : OUT     STD_LOGIC
        );
END ENTITY;

ARCHITECTURE rtl OF top IS

    SIGNAL
int_clk,int_MST,int_SST,int_Debouncer,int_SetCounter,int_CounterExpired,int_S:
STD_LOGIC;
    SIGNAL            int_timer_out,             int_count_out,int_mux_out,int_set:
STD_LOGIC_VECTOR(3 downto 0);

COMPONENT fsm IS
    PORT(
        i_resetBar      : IN  STD_LOGIC;
        i_clock              : IN  STD_LOGIC;
      i_SSCS,i_MSC,i_MST,i_SSC,i_SST: IN STD_LOGIC;
        o_MSTL,o_SSTL                : OUT     STD_LOGIC_VECTOR(2   downto
0);
        o_reset,o_sw  : OUT STD_LOGIC;
          o_state              : OUT     STD_LOGIC_VECTOR(1 downto 0)
          );
END COMPONENT;

COMPONENT debouncer_2 IS
    PORT(
        i_resetBar           : IN  STD_LOGIC;
        i_clock              : IN  STD_LOGIC;
        i_raw                : IN  STD_LOGIC;
        o_clean                 : OUT       STD_LOGIC);
END COMPONENT;

COMPONENT clk_div IS
    PORT
    (
        clock_25Mhz                     : IN  STD_LOGIC;
        clock_1MHz                      : OUT       STD_LOGIC;
        clock_100KHz                        : OUT       STD_LOGIC;
        clock_10KHz                     : OUT       STD_LOGIC;
        clock_1KHz                      : OUT       STD_LOGIC;
        clock_100Hz                     : OUT       STD_LOGIC;
        clock_10Hz                      : OUT       STD_LOGIC;
        clock_1Hz                       : OUT       STD_LOGIC);
END COMPONENT;

COMPONENT counter_4bit IS
    PORT(
        clk : IN std_logic;
        rst : IN std_logic;
        count : OUT std_logic_vector(3  DOWNTO 0)
    );
END COMPONENT;

COMPONENT fourBitComparator  IS
    PORT(
        i_A, i_B                : IN        STD_LOGIC_VECTOR(3 DOWNTO 0);
        o_GT, o_LT, o_EQ    : OUT       STD_LOGIC
    );
END COMPONENT;

COMPONENT mux2to1_4 IS
    PORT(
        i_S:IN STD_LOGIC;
        i_a,i_b:IN STD_LOGIC_VECTOR(3 downto 0);
        o_O:OUT STD_LOGIC_VECTOR(3 downto 0));
END COMPONENT;
```

```
COMPONENT timer_o IS
	PORT(
		i_input: IN STD_LOGIC_VECTOR(3 downto 0);
		o_MST,o_SST: OUT	STD_LOGIC);
END COMPONENT;

BEGIN

	--change clock speed here
	int_clk<=i_clk;

clock_divider: clk_div
	PORT MAP(
		clock_25Mhz=> i_clk
		--clock_1Hz=>int_clk
	);

Timer_m: counter_4bit
	PORT MAP(
		clk=> int_clk,
		rst=> i_resetBar,
		count=> int_timer_out
	);

Timer_out: timer_o
	PORT MAP(
		i_input=> int_timer_out,
		o_MST => int_MST,
		o_SST => int_SST
	);

debouncer: debouncer_2
	PORT MAP(
		i_resetBar=>i_resetBar,
		i_clock	=> int_clk,
		i_raw=> i_pbs,
		o_clean=> int_Debouncer
	);
```

```
fsm_controller:  fsm
        PORT MAP(
                i_resetBar     => i_resetBar,
                i_clock        => int_clk,
                i_SSCS     => int_Debouncer,
                i_MSC => int_CounterExpired,
                i_MST => int_MST,
                i_SSC => int_CounterExpired,
                i_SST => int_SST,
                o_sw => int_S,
                o_MSTL => o_MST,
                o_SSTL => o_SST,
                o_reset => int_SetCounter
        );

cmp: fourBitComparator
        PORT MAP(
                i_A => int_count_out,
                i_B => int_mux_out,
                o_EQ => int_CounterExpired
        );

counter: counter_4bit
        PORT MAP(
                clk =>int_clk,
                rst=> i_resetBar,
                count =>int_count_out
        );

mux: mux2to1_4
        PORT MAP(
                i_S=>int_S,
                i_a=>i_sw1,
                i_b=>i_sw2,
                o_O=>int_mux_out
        );
```

o_clk <= int_clk;

END ARCHITECTURE;

Figure 2.3 Traffic Light Controller with Debug Message Implementation with VHDL

Part III – Discussion and Waveform Explanation

In this Project, UART is the main component that students are implementing. There are so problems that we get into which cause a lot of error. The primary problem is that the Baud Rate Generator is not working when we are first testing the code. We found out that there are two way which can deal with this situation. One of the ways is to delete the Baud Rate Generator and connect all the clock on the universal clock. This is not the best way to fix this problem since Baud rate generators are an important part in the UART. The actual way we found to fix the error is that we found out that SCSR and SCCR should not be connected to the Baud rate generator. This solves the problem and preform the actual waveform in the simulation.

The waveform was first preforming well in the traffic light controller, but for the data transfer line TxD, the data is not transferring out to the computer. After changing the connection inside the UART, we can see in the first figure below that we simulate an ASCII code "10010011", and we can see the Transmitted line TxD starts to transfer the data bit serially from the last to the first bit.

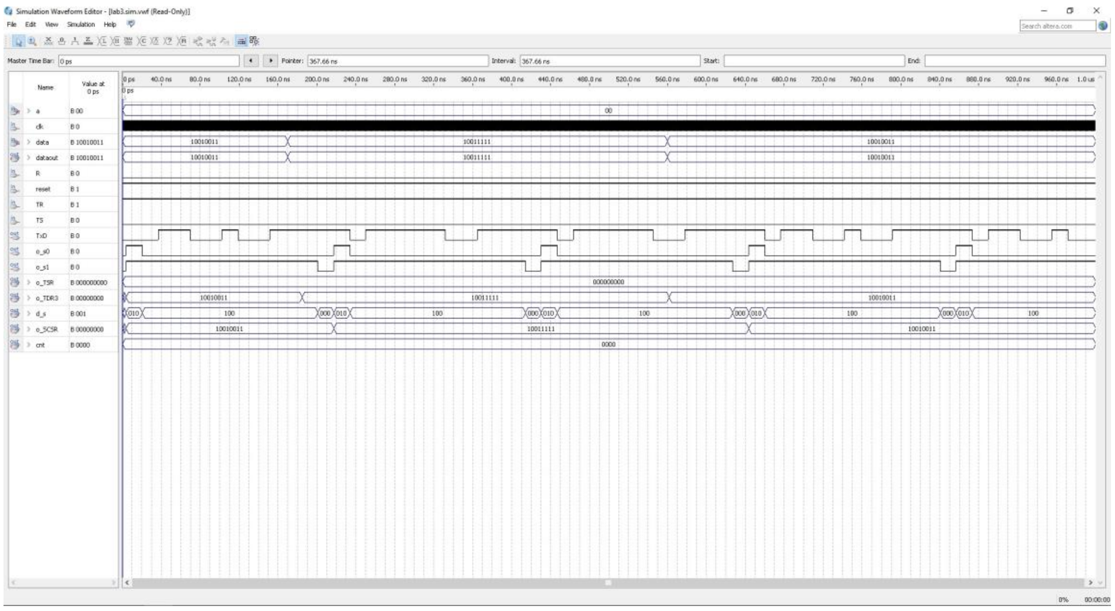


Figure 3.1 Debugging UART with Simulation

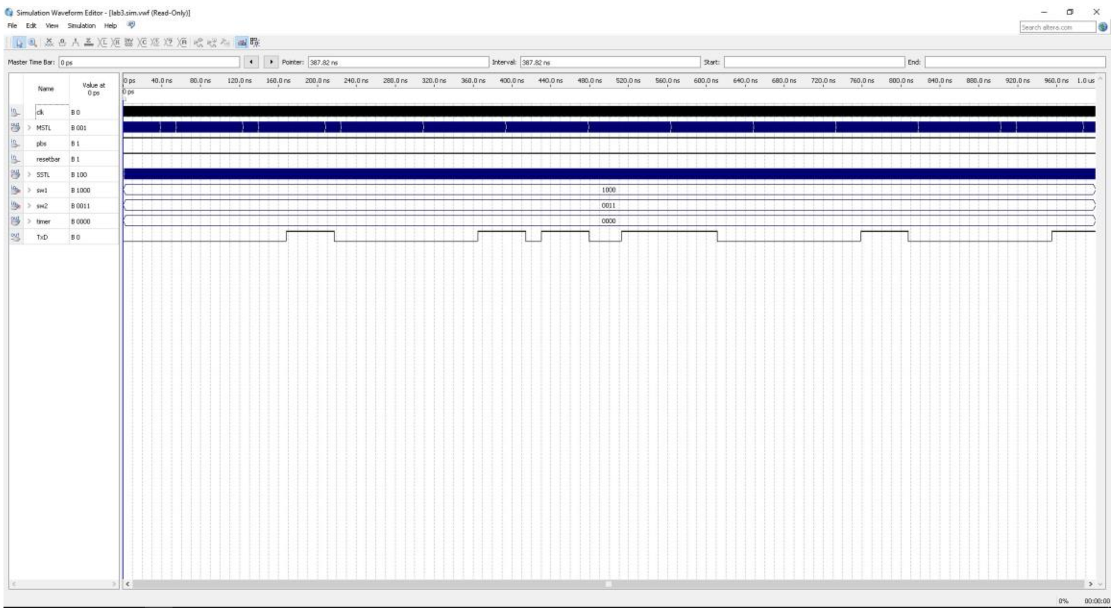


Figure 3.2 Simulation for UART Transmitter Design

Part IV – Conclusion

In conclusion, to perform a well-designed UART transmitter would need to prioritize the part of UART FSM, Baud Rate Generator, address decoder, TSR and TSR controller. These parameters are the most important things that would combine into a transmitter. UART FSM controls the performance of which debug message should be output and when the message should be transferred. Baud rate generator is an important part which controls the time and rate between two devices, facilitating accurate and synchronized serial communication between devices. TSR act as the most important role of transmitting the data out to the other device by using right shift bit serially. TSR control is mainly controlling the timing and how the data is transported out to the other device. Address decoder is used for selecting which part of the UART register should transfer the data into to the data bus for other special registers usage. These important parts in the UART should go through thorough testing, adherence to standards, and comprehensive documentation contribute to the success of the design and its long-term maintenance.

Reference

https://github.com/as38063997/TermProject_CEG3155