VHDL Code
Simulation & Implementation
in ISE Xilinx

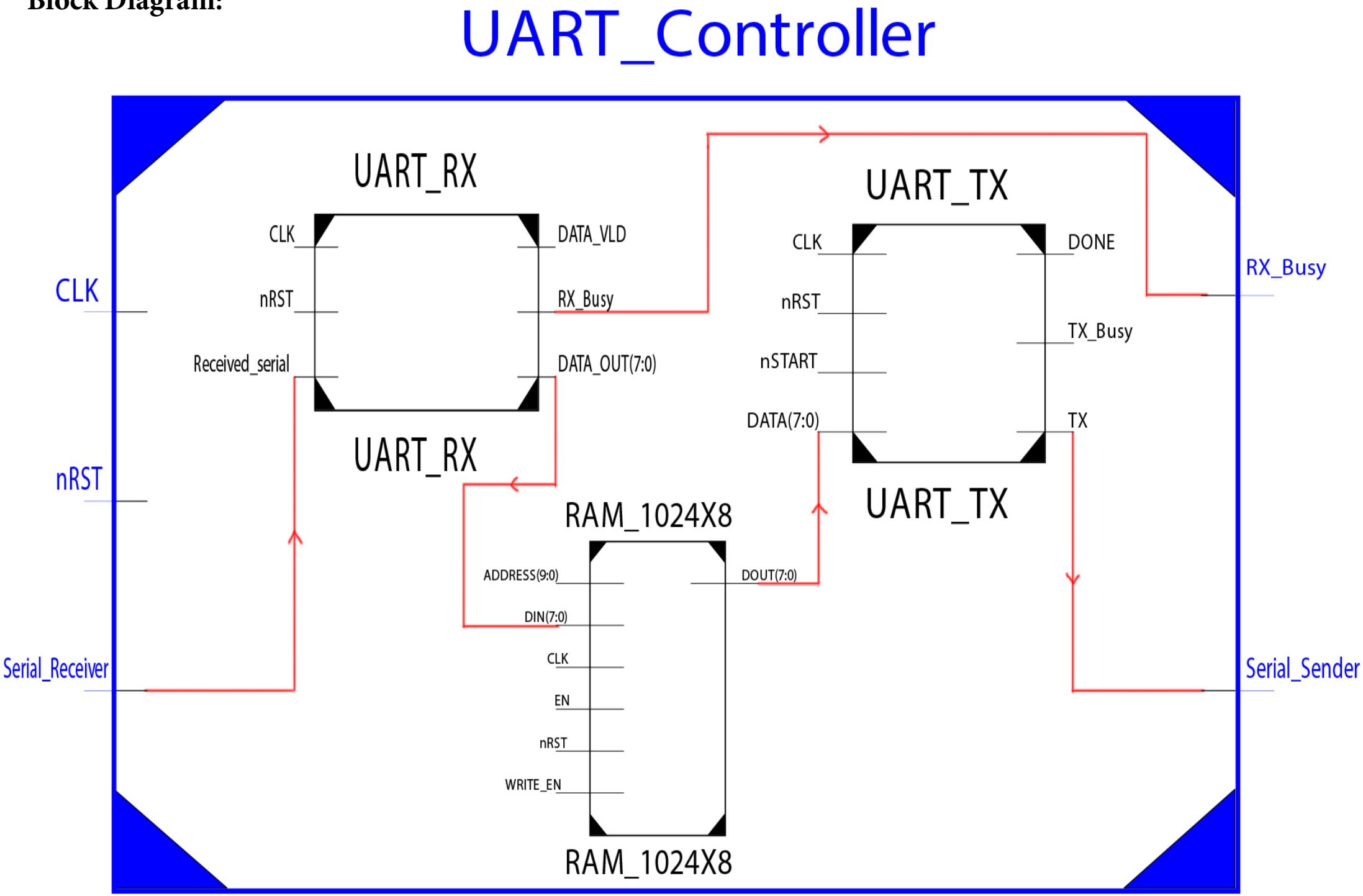
UART Controller:

UART_RX / RAM_1024X8 / UART_TX

UART_RX_TB

Mohammad Niknam

Block Diagram:



ماژولی با عملکرد زیر را طراحی و مبتنی بر VHDL سنتزپذیر توصیف نمایید:

ورودی سریال بر اساس استاندارد UART را دریافت کرده و به ترتیب در یک حافظه با ظرفیت یک کیلو بایت(1024 بایت) ذخیره می کند. پس از تکمیل ظرفیت حافظه (دریافت 1024 بایت ورودی یا همان 1024 بسته داده) این بار مقادیر نوشته شده در حافظه بر اساس استاندارد UART ارسال می شوند. پس از ارسال تمامی داده های حافظه، ماژول دوباره در وضعیت دریافت اطلاعات قرار می گیرد (پیشنهاد میشود، خروجی برای اعلان وضعیت فعلی ماژول در نظر بگیرید).

(راهنمایی: پیش از این هر سه ماژول اصلی شامل کنترلر دریافت و ارسال UART و همچنین حافظه را توصیف کرده ایم. پس کافی است که این ماژول ها به درستی جایگذاری شوند و یک ماشین حالت برای انجام عملکرد فوق طراحی گردد)

در طراحی برای سادگی در ارزیابی ماژول فرکانس و baudrate به نحو زیر در نظر گرفته شد

Freq: 100 MHZ

Baudrate: 115200

Pulse for each bit: 868

Universal Asynchronous Receiver/Transmitter

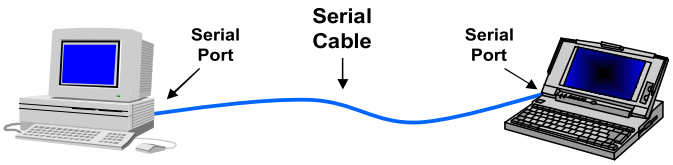
UART

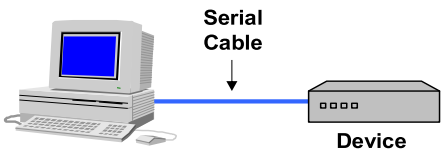
Why use a UART?

- A UART may be used when:
  - High speed is not required
  - A cheap communication line between **two** devices is required
- Asynchronous serial communication is very cheap
  - Requires a transmitter and/or receiver
  - Single wire for each direction (plus ground wire)
  - Relatively simple hardware
  - Asynchronous because the
- PC devices such as mice and modems used to often be asynchronous serial devices

UART Uses

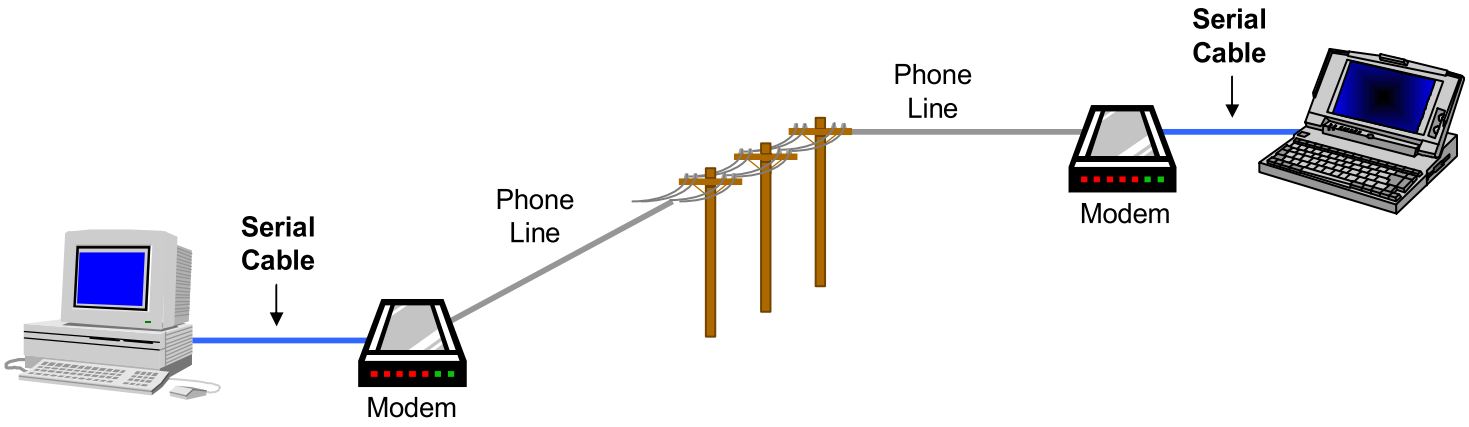
- PC serial port is a UART!
- Serializes data to be sent over serial cable
  - De-serializes received data



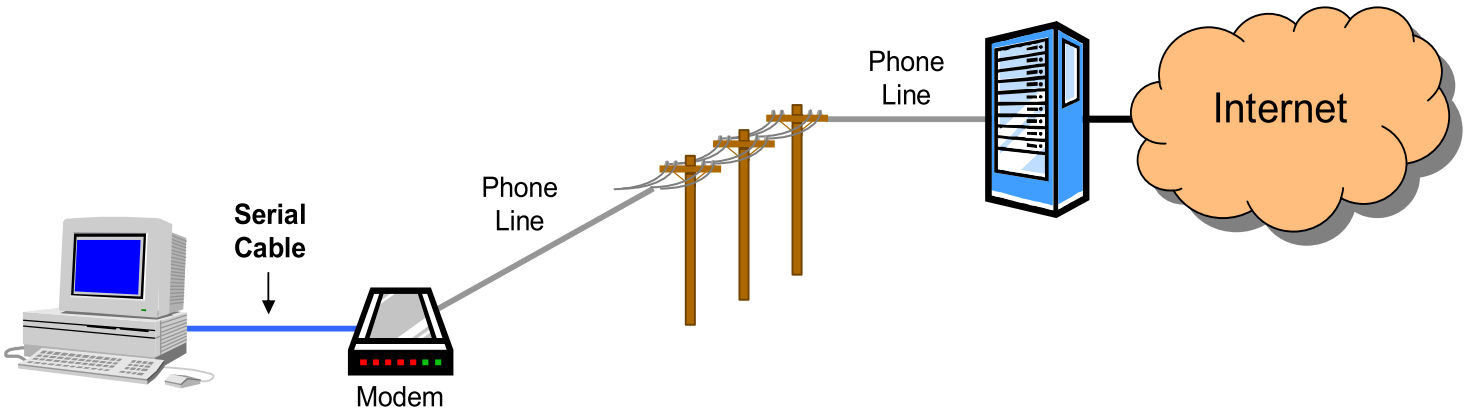


UART Uses

- Communication between distant computers
  - Serializes data to be sent to modem
  - De-serializes data received from modem

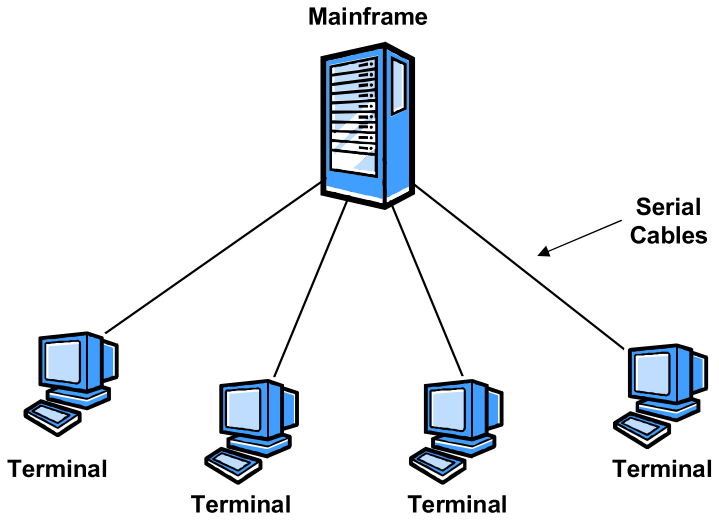


- Used to be commonly used for internet access



UART Uses

- Used to be used for mainframe access
  - A mainframe could have dozens of serial ports



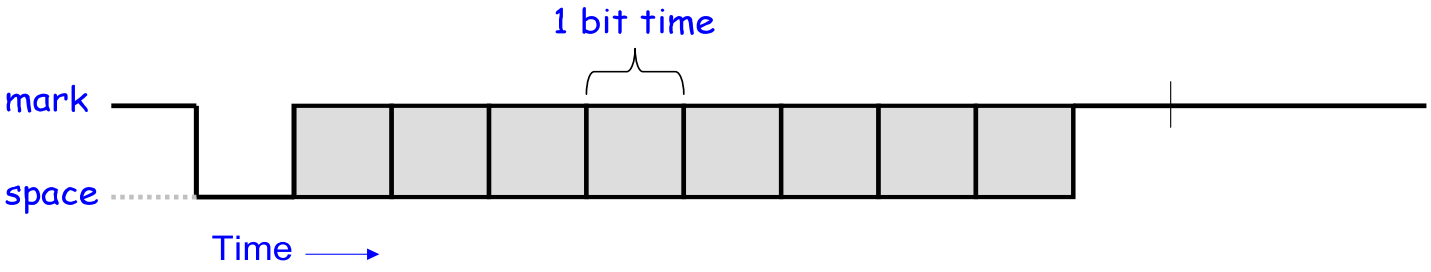
- Becoming much less common
- Largely been replaced by faster, more sophisticated interfaces
  - PCs: USB (peripherals), Ethernet (networking)
  - Chip to chip: I2C, SPI
- Still used today when simple low speed communication is needed

UART Functions

- Outbound data
  - Convert from parallel to serial
  - Add start and stop delineators (bits)
  - Add parity bit
- Inbound data
  - Convert from serial to parallel
  - Remove start and stop delineators (bits)
  - Check and remove parity bit

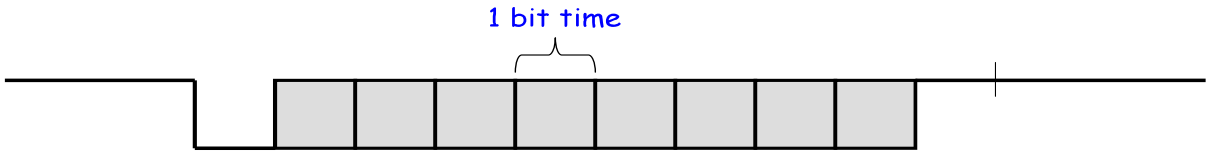
UART Character Transmission

- Below is a timing diagram for the transmission of a single byte
- Uses a single wire for transmission
- Each block represents a bit that can be a **mark** (logic '1', high) or **space** (logic '0', low)



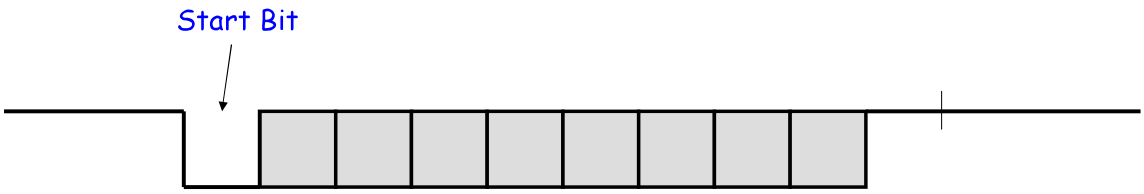
UART Character Transmission

- Each bit has a fixed time duration determined by the transmission rate
- Example: a 1200 bps (bits per second) UART will have a 1/1200 s or about 833.3 μs bit width



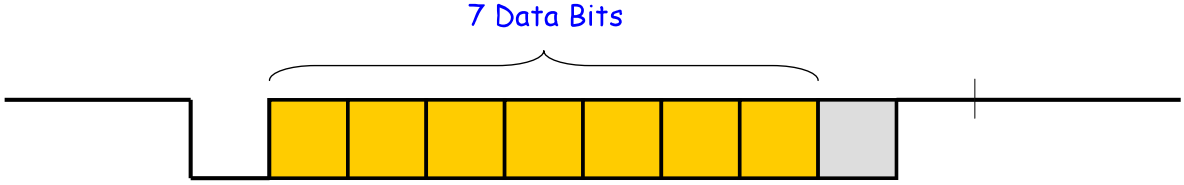
UART Character Transmission

- The **start bit** marks the beginning of a new word
- When detected, the receiver synchronizes with the new data stream



UART Character Transmission

- Next follows the **data bits** (7 or 8)
- The least significant bit is sent first

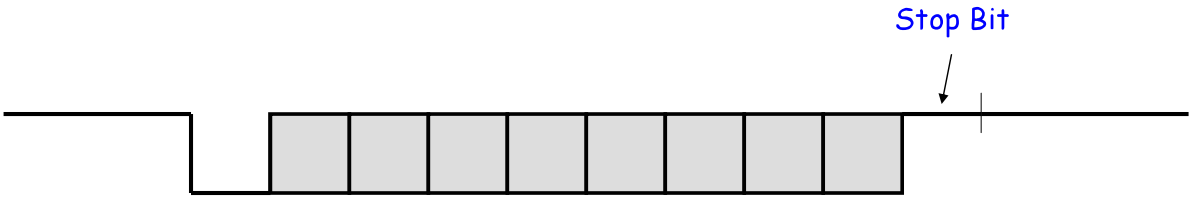


UART Character Transmission

- The **parity bit** is added to make the number of 1's even (even parity) or odd (odd parity)
- This bit can be used by the receiver to check for transmission errors

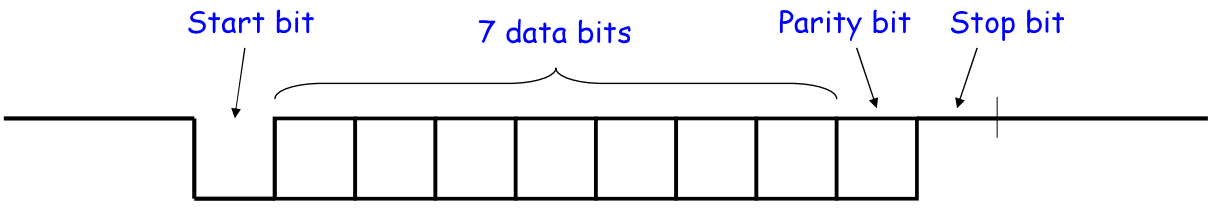
UART Character Transmission

- The **stop bit** marks the end of transmission
- Receiver checks to make sure it is '1'
- Separates one word from the start bit of the next word



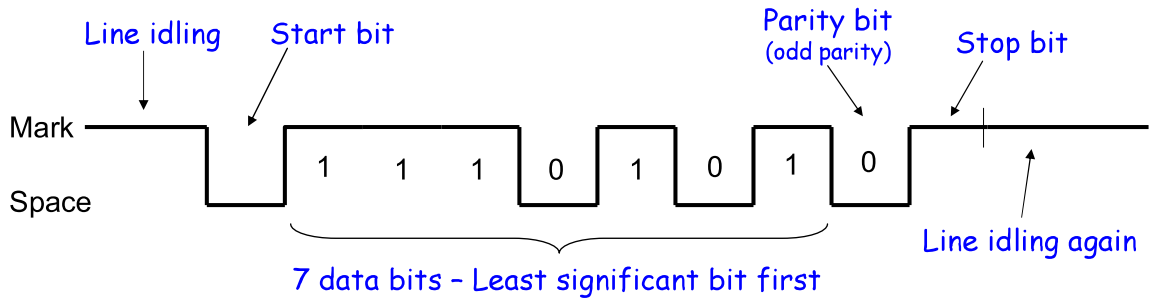
UART Character Transmission

- In the configuration shown, it takes 10 bits to send 7 bits of data



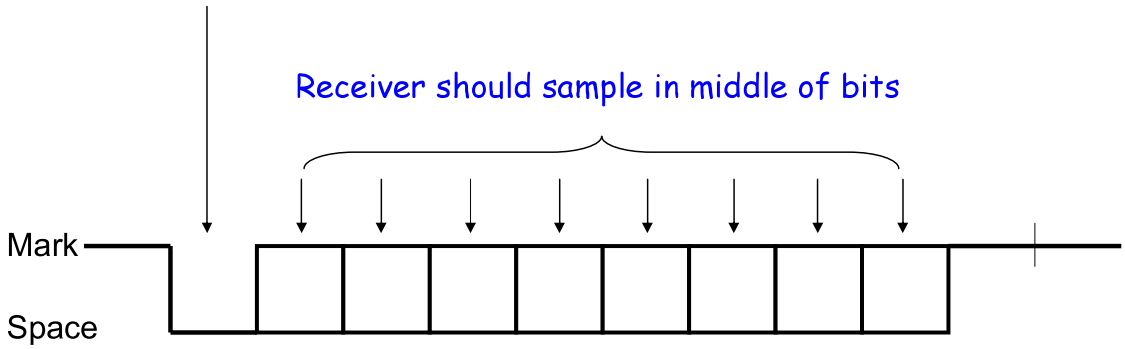
UART Transmission Example

- Send the ASCII letter 'W' (1010111)



UART Character Reception

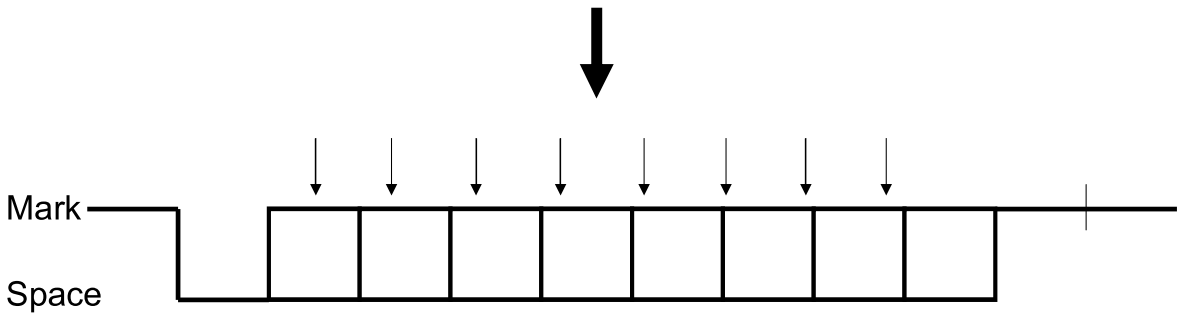
Start bit says a character is coming,
receiver resets its timers



Receiver uses a timer (counter) to time when it samples.
Transmission rate (i.e., bit width) must be known!

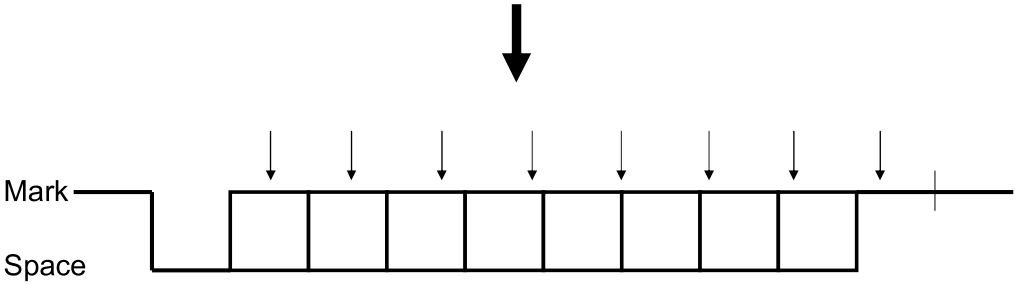
UART Character Reception

If receiver samples too quickly, see what happens…



UART Character Reception

If receiver samples too slowly, see what happens…



Receiver resynchronizes on every start bit.
Only has to be accurate enough to read 9 bits.

UART Character Reception

- Receiver also verifies that stop bit is '1'
  - If not, reports "framing error" to host system
- New start bit can appear immediately after stop bit
  - Receiver will resynchronize on each start bit

UART Options

- UARTs usually have programmable options:
  - **Data:** 7 or 8 bits
  - **Parity:** even, odd, none, mark, space
  - **Stop bits:** 1, 1.5, 2
  - **Baud rate:** 300, 1200, 2400, 4800, 9600, 19.2K, 38.4k, 57.6k, 115.2k…

UART_RX:

VHDL CODE

```vhdl
-- Engineer: Mohammad Niknam
-- Project Name:  UART_Controller
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity UART_RX is
    Generic(CLK_PULSE_NUM : integer := 868); --868 for freq=100Mhz / BUADRATE 115200
    port (  CLK : in std_logic;
            nRST :in std_logic;
            DATA_OUT : out std_logic_vector(7 downto 0);
            DATA_VLD : out std_logic; -- when DATA_VLD = 1, data reception is completed data on DATA_OUT are valid
            RX_Busy : out std_logic;  --data reception in progress
            Received_serial : in std_logic);
end UART_RX;

architecture Behavioral of UART_RX is
type FSMTYPE is (INIT_STATE, START_BIT_Receive, BIT_0, BIT_1, BIT_2, BIT_3, BIT_4, BIT_5, BIT_6, BIT_7, STOP_BIT_Receive);

signal CSTATE, NSTATE : FSMTYPE;
signal CLK_CNT : unsigned(11 downto 0) ;
signal CLK_CNT_RST : std_logic;
signal DATA_REG : std_logic_vector(7 downto 0);
signal TMP : std_logic_vector(7 downto 0);
signal DATA_VALID : std_logic;

begin

    data_registration : process( CLK )
        variable VTMP : std_logic_vector(7 downto 0);
    begin
        if (CLK'event and CLK = '1') then
            TMP <= DATA_REG;
            if (DATA_VALID = '1') then
                DATA_OUT <= DATA_REG;
                DATA_VLD <= '1';
                VTMP := DATA_REG;
            else
                DATA_VLD <= '0';
                DATA_OUT <= VTMP;
            end if ;
        end if ;
    end process ; --data_registration


clk_counter : process( CLK )
begin
    if (CLK'event and CLK = '1') then
        if (CLK_CNT_RST = '1') then
            CLK_CNT <= (others => '0');
        else
            CLK_CNT <= CLK_CNT + 1;
        end if ;
    end if ;
end process ; -- clk_counter


state_registration : process( CLK )
begin
    if (CLK'event and CLK = '1') then
        if (nRST = '0') then
            CSTATE <= INIT_STATE;
        else
            CSTATE <= NSTATE;
        end if ;
    end if ;
end process ; -- state_registration
```

UART_RX:

VHDL CODE

```
74  next_state : process( CSTATE, CLK_CNT, DATA_REG, Received_serial, TMP)
75  begin
76      NSTATE <= CSTATE;
77      CLK_CNT_RST <= '0';
78      DATA_VALID <= '0';
79      DATA_REG <= TMP;
80
81      case( CSTATE ) is
82          when INIT_STATE =>
83              DATA_REG <= (others => '0');
84              RX_Busy <= '0';
85              CLK_CNT_RST <= '1';
86              if (Received_serial = '0') then
87                  NSTATE <= START_BIT_Receive;
88              end if ;
89
90          when START_BIT_Receive =>
91              RX_Busy <= '1';
92              if (TO_integer(CLK_CNT) = (CLK_PULSE_NUM - 1)) then
93                  CLK_CNT_RST <= '1';
94                  NSTATE <= BIT_0 ;
95              end if ;
96
97          when BIT_0 =>
98              DATA_REG(0) <= Received_serial;
99              RX_Busy <= '1';
100             if (to_integer(CLK_CNT) = (CLK_PULSE_NUM - 1)) then
101                 CLK_CNT_RST <= '1';
102                 NSTATE <= BIT_1;
103             end if;
104
105         when BIT_1 =>
106             DATA_REG(1) <= Received_serial;
107             RX_Busy <= '1';
108             if (to_integer(CLK_CNT) = (CLK_PULSE_NUM - 1)) then
109                 CLK_CNT_RST <= '1';
110                 NSTATE <= BIT_2;
111             end if;
112
113         when BIT_2 =>
114             DATA_REG(2) <= Received_serial;
115             RX_Busy <= '1';
116             if (to_integer(CLK_CNT) = (CLK_PULSE_NUM - 1)) then
117                 CLK_CNT_RST <= '1';
118                 NSTATE <= BIT_3;
119             end if;
120
121         when BIT_3 =>
122             DATA_REG(3) <= Received_serial;
123             RX_Busy <= '1';
124             if (to_integer(CLK_CNT) = (CLK_PULSE_NUM - 1)) then
125                 CLK_CNT_RST <= '1';
126                 NSTATE <= BIT_4;
127             end if;
128
129         when BIT_4 =>
130             DATA_REG(4) <= Received_serial;
131             RX_Busy <= '1';
132             if (to_integer(CLK_CNT) = (CLK_PULSE_NUM - 1)) then
133                 CLK_CNT_RST <= '1';
134                 NSTATE <= BIT_5;
135             end if;
136
137         when BIT_5 =>
138             DATA_REG(5) <= Received_serial;
139             RX_Busy <= '1';
140             if (to_integer(CLK_CNT) = (CLK_PULSE_NUM - 1)) then
141                 CLK_CNT_RST <= '1';
142                 NSTATE <= BIT_6;
143             end if;
144
```

UART_RX:

VHDL CODE

```
151             when BIT_6 =>
152                 DATA_REG(6) <= Received_serial;
153                 RX_Busy <= '1';
154                 if (to_integer(CLK_CNT) = (CLK_PULSE_NUM - 1)) then
155                     CLK_CNT_RST <= '1';
156                     NSTATE <= BIT_7;
157                 end if;
158 
159             when BIT_7 =>
160                 DATA_REG(7) <= Received_serial;
161                 RX_Busy <= '1';
162                 if (to_integer(CLK_CNT) = (CLK_PULSE_NUM - 1)) then
163                     CLK_CNT_RST <= '1';
164                     NSTATE <= STOP_BIT_Receive;
165                 end if;
166 
167             when STOP_BIT_Receive =>
168                 RX_Busy <= '1';
169                 if (to_integer(CLK_CNT) = (CLK_PULSE_NUM - 1)) then
170                     CLK_CNT_RST <= '1';
171                     NSTATE <= INIT_STATE;
172                     RX_Busy <= '0';
173                     if (Received_serial = '1') then
174                         DATA_VALID <= '1';
175                     end if ;
176                 end if;
177 
178             when others =>
179         end case ;
180     end process ; -- next_state
181 
182     end Behavioral;
183 
184 
```

UART_RX_TB:

VHDL CODE

```
1   -- Engineer: Mohammad Niknam
2   -- Project Name:  UART_Controller
3   -- VHDL Test Bench Created by ISE for module: UART_RX
4   -- CLK_PULSE_NUM : 868 for freq=100Mhz / BUADRATE 115200
5   LIBRARY ieee;
6   USE ieee.std_logic_1164.ALL;
7   use IEEE.NUMERIC_STD.ALL;
8
9   ENTITY UART_RX_TB IS
10  END UART_RX_TB;
11
12  ARCHITECTURE behavior OF UART_RX_TB IS
13
14      -- Component Declaration for the Unit Under Test (UUT)
15
16      COMPONENT UART_RX
17      PORT(
18           CLK : IN  std_logic;
19           nRST : IN  std_logic;
20           DATA_OUT : OUT  std_logic_vector(7 downto 0);
21           DATA_VLD : OUT  std_logic;
22           RX_Busy : OUT  std_logic;
23           Received_serial : IN  std_logic
24          );
25      END COMPONENT;
26
27
28     --Inputs
29     signal CLK : std_logic := '0';
30     signal nRST : std_logic := '0';
31     signal Received_serial : std_logic := '0';
32
33      --Outputs
34     signal DATA_OUT : std_logic_vector(7 downto 0);
35     signal DATA_VLD : std_logic;
36     signal RX_Busy : std_logic;
37
38     -- Clock period definitions
39     constant CLK_TIME : time := 5 ns;       --DATA_Receive_period : ((868*2)+1) *
       CLK_TIME ns;
40
41     BEGIN
42
43      -- Instantiate the Unit Under Test (UUT)
44     uut: UART_RX PORT MAP (
45            CLK => CLK,
46            nRST => nRST,
47            DATA_OUT => DATA_OUT,
48            DATA_VLD => DATA_VLD,
49            RX_Busy => RX_Busy,
50            Received_serial => Received_serial
51          );
52
53
54
55
56
```

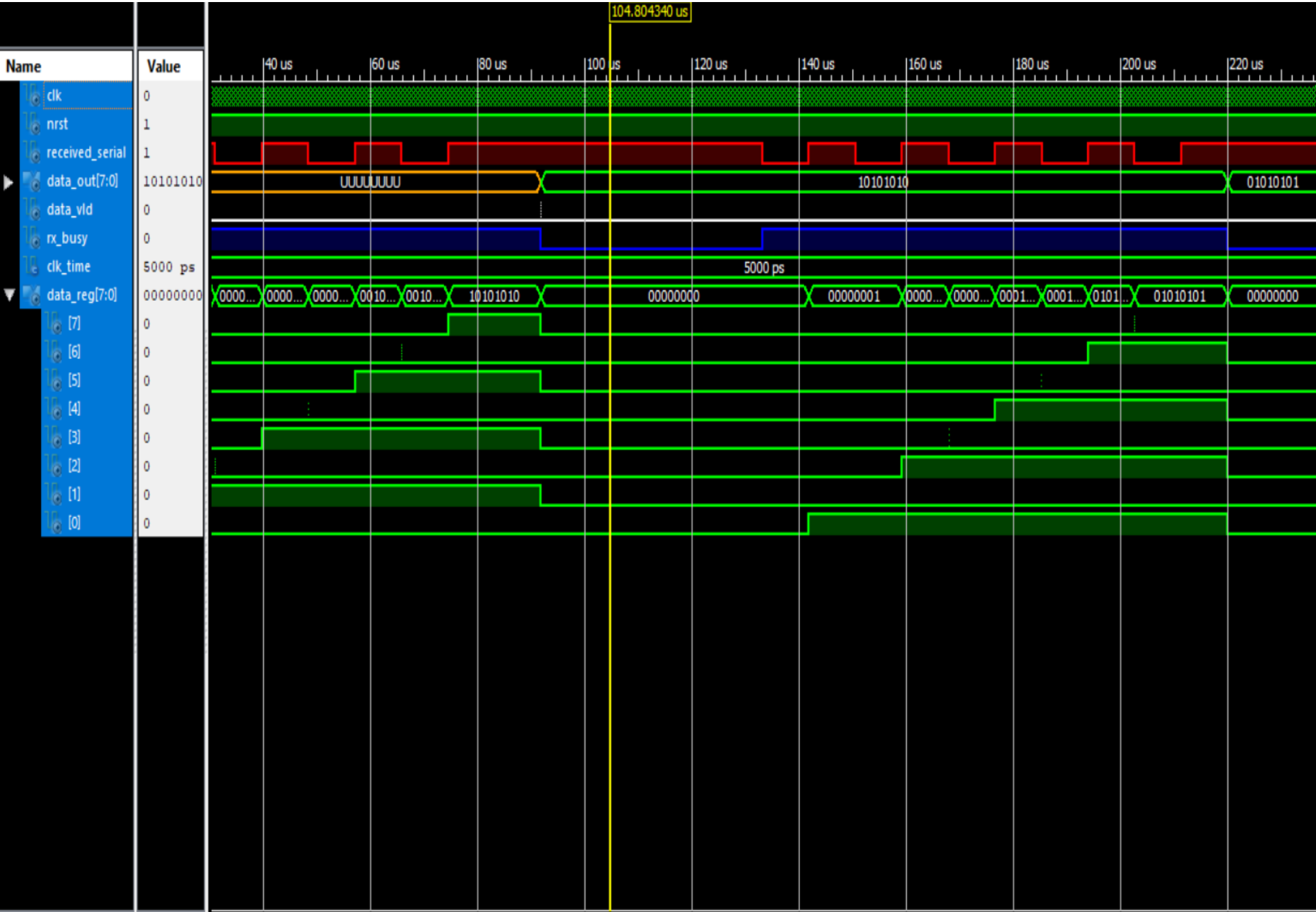
UART_RX_TB::

VHDL CODE

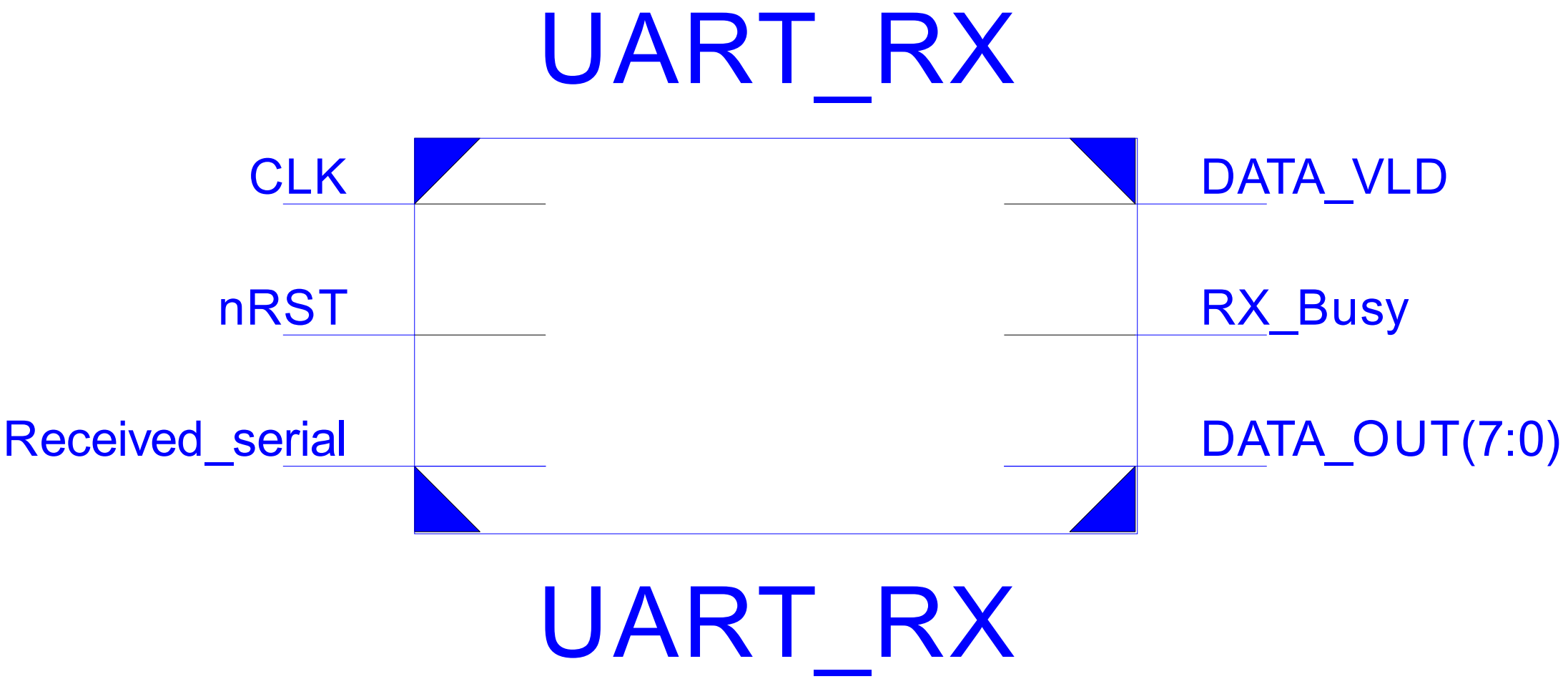
```
74      CLK <= not(CLK) after CLK_TIME;
75      nRST <= '0', '1' after 400 ns;  -- hold reset state for 400 ns.
76      pro: process
77      begin
78         Received_serial <= '1';
79         wait for 5000 ns;
80         Received_serial <= '0'; --START_BIT
81         wait for 1737 * CLK_TIME;
82         Received_serial <= '0'; --BIT_0
83         wait for 1737 * CLK_TIME;
84         Received_serial <= '1'; --BIT_1
85         wait for 1737 * CLK_TIME;
86         Received_serial <= '0'; --BIT_2
87         wait for 1737 * CLK_TIME;
88         Received_serial <= '1'; --BIT_3
89         wait for 1737 * CLK_TIME;
90         Received_serial <= '0'; --BIT_4
91         wait for 1737 * CLK_TIME;
92         Received_serial <= '1'; --BIT_5
93         wait for 1737 * CLK_TIME;
94         Received_serial <= '0'; --BIT_6
95         wait for 1737 * CLK_TIME;
96         Received_serial <= '1'; --BIT_7
97         wait for 1737 * CLK_TIME;
98         Received_serial <= '1'; --STOP_BIT
99         wait for 50000 ns;
100        Received_serial <= '0'; --START_BIT
101        wait for 1737 * CLK_TIME;
102        Received_serial <= '1'; --BIT_0
103        wait for 1737 * CLK_TIME;
104        Received_serial <= '0'; --BIT_1
105        wait for 1737 * CLK_TIME;
106        Received_serial <= '1'; --BIT_2
107        wait for 1737 * CLK_TIME;
108        Received_serial <= '0'; --BIT_3
109        wait for 1737 * CLK_TIME;
110        Received_serial <= '1'; --BIT_4
111        wait for 1737 * CLK_TIME;
112        Received_serial <= '0'; --BIT_5
113        wait for 1737 * CLK_TIME;
114        Received_serial <= '1'; --BIT_6
115        wait for 1737 * CLK_TIME;
116        Received_serial <= '0'; --BIT_7
117        wait for 1737 * CLK_TIME;
118        Received_serial <= '1'; --STOP_BIT
119        wait;
120     end process;
121  END;
122
```

UART_RX Simulation:

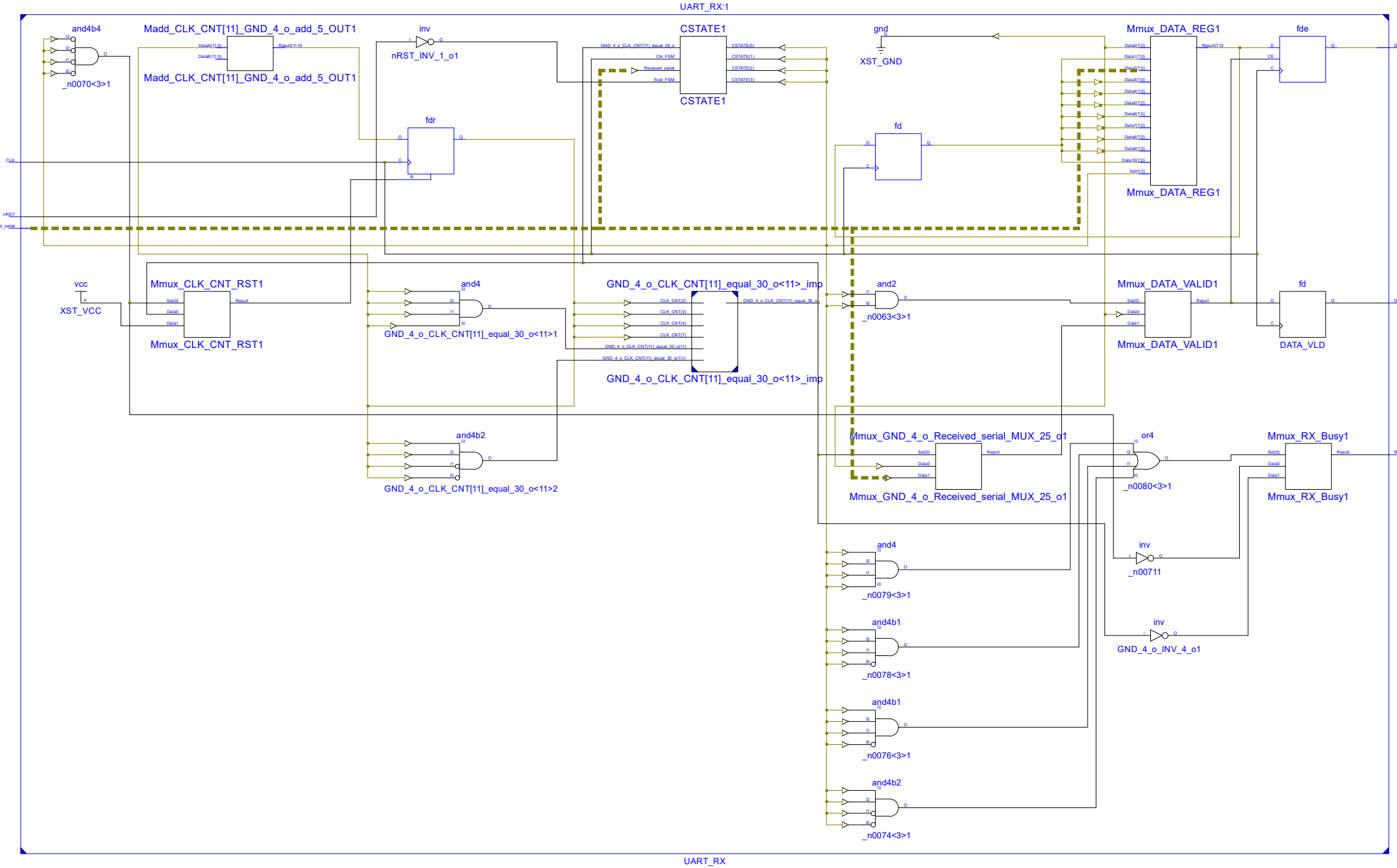
UART_RX Simulation:



UART_RX Synthesize & RTL Schematic:

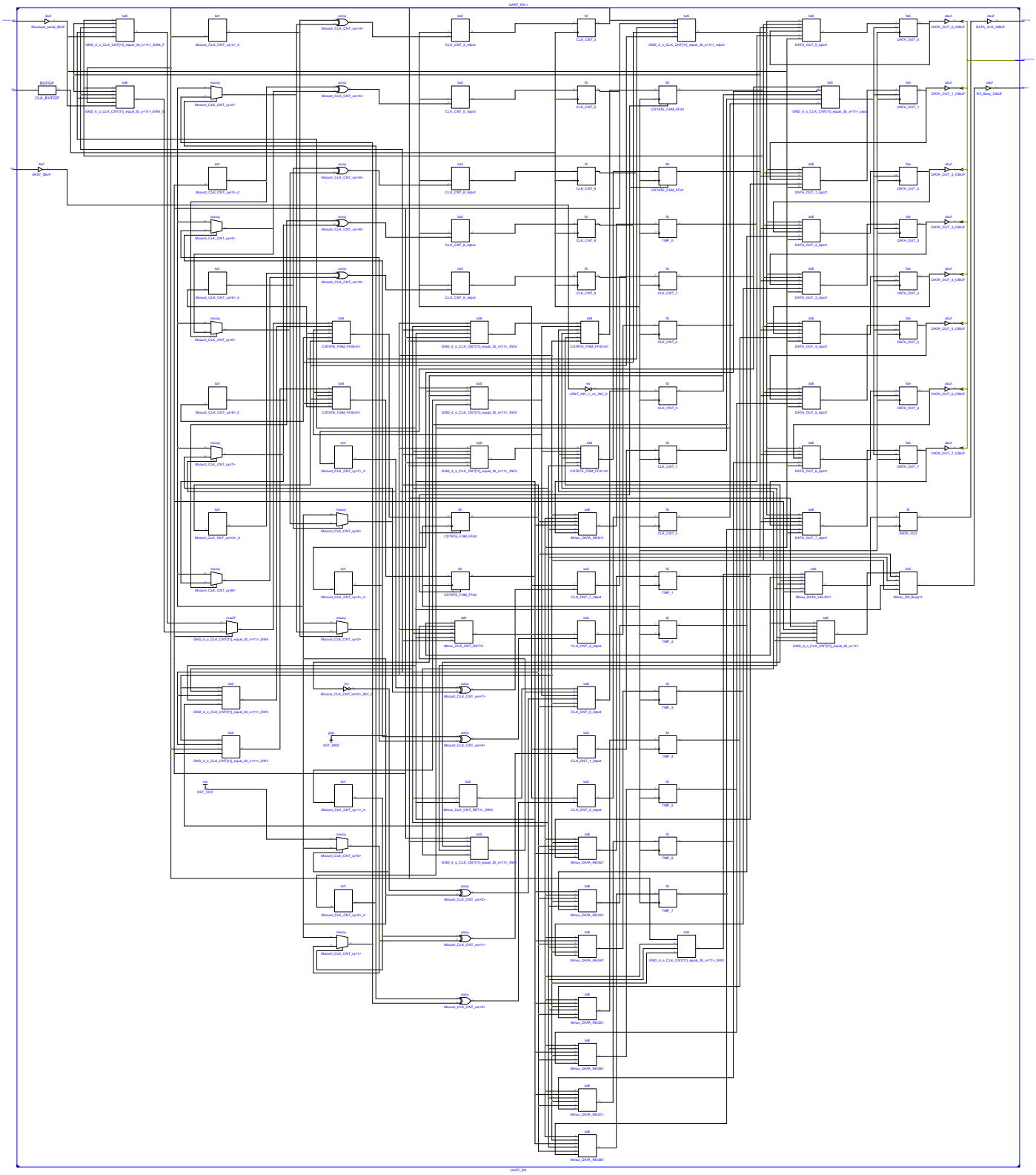


UART_RX Synthesize & RTL Schematic:



UART_RX Post-Route Simulation:

UART_RX Technology Schematic:



UART_TX:

VHDL CODE

```vhdl
-- Engineer: Mohammad Niknam
-- Project Name:  UART_Controller
-- Module Name:   UART_TX - Behavioral
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity UART_TX is
    Generic(CLK_PULSE_NUM : integer := 868); --868 for freq=100Mhz / BUADRATE 115200
    port (  CLK : in std_logic;
            nRST :in std_logic;
            nSTART : in std_logic;
            DATA : in std_logic_vector(7 downto 0);
            DONE : out std_logic;
            TX_Busy : out std_logic;  --data sending in progress
            TX : out std_logic);
end UART_TX;

architecture Behavioral of UART_TX is
type FSMTYPE is (INIT_STATE, START_BIT_SEND, BIT_0, BIT_1, BIT_2, BIT_3, BIT_4,
BIT_5, BIT_6, BIT_7, STOP_BIT_SEND);

signal CSTATE, NSTATE : FSMTYPE;
signal CLK_CNT : unsigned(11 downto 0) ;
signal CLK_CNT_RST : std_logic;
signal DATA_REG : std_logic_vector(7 downto 0);

begin

data_registration : process( CLK )
begin
    if (CLK'event and CLK = '1') then
        if (nSTART = '0') then
        DATA_REG <= DATA;
        end if ;
    end if ;
end process ; --data_registration


clk_counter : process( CLK )
begin
    if (CLK'event and CLK = '1') then
        if (CLK_CNT_RST = '1') then
            CLK_CNT <= (others => '0');
        else
            CLK_CNT <= CLK_CNT + 1;
        end if ;
    end if ;
end process ; -- clk_counter


state_registration : process( CLK )
begin
    if (CLK'event and CLK = '1') then
        if (nRST = '0') then
            CSTATE <= INIT_STATE;
        else
            CSTATE <= NSTATE;
        end if ;
    end if ;
end process ; -- state_registration
```

UART_TX:

VHDL CODE

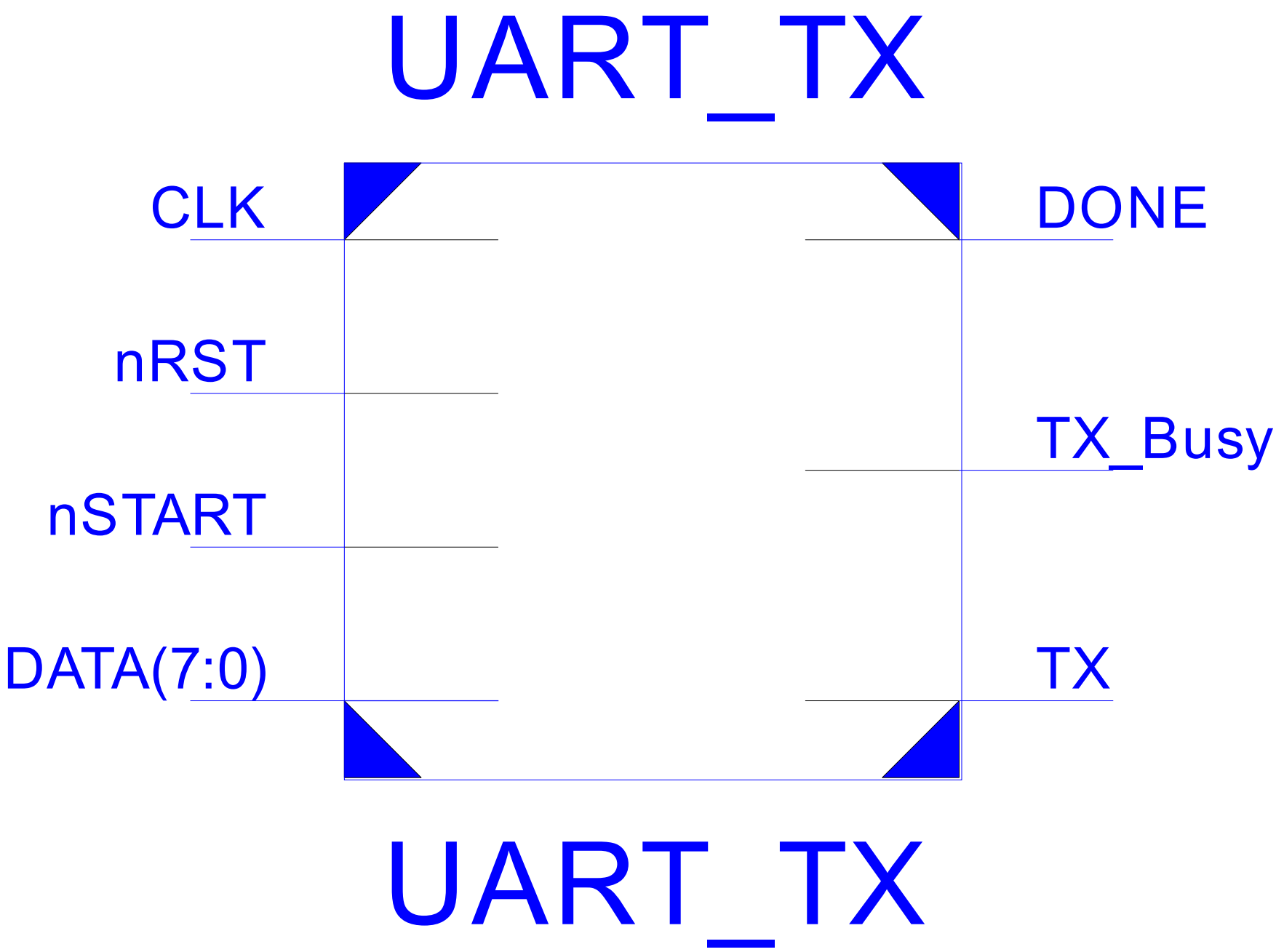
```
74   next_state : process( CSTATE, CLK_CNT, DATA_REG, nSTART )
75   begin
76       NSTATE <= CSTATE;
77       CLK_CNT_RST <= '0';
78       TX <= '1';
79       DONE <= '0';
80
81       case( CSTATE ) is
82           when INIT_STATE =>
83               TX <= '1';
84               CLK_CNT_RST <= '1';
85               TX_Busy <= '0';
86               if (nSTART = '0') then
87                   NSTATE <= START_BIT_SEND;
88               end if ;
89
90           when START_BIT_SEND =>
91               TX <= '0';
92               TX_Busy <= '1';
93               if (TO_integer(CLK_CNT) = (CLK_PULSE_NUM - 1)) then
94                   CLK_CNT_RST <= '1';
95                   NSTATE <= BIT_0 ;
96               end if ;
97
98           when BIT_0 =>
99               TX <= DATA_REG(0);
100              TX_Busy <= '1';
101              if (to_integer(CLK_CNT) = (CLK_PULSE_NUM - 1)) then
102                  CLK_CNT_RST <= '1';
103                  NSTATE <= BIT_1;
104              end if;
105
106          when BIT_1 =>
107              TX <= DATA_REG(1);
108              TX_Busy <= '1';
109              if (to_integer(CLK_CNT) = (CLK_PULSE_NUM - 1)) then
110                  CLK_CNT_RST <= '1';
111                  NSTATE <= BIT_2;
112              end if;
113
114          when BIT_2 =>
115              TX <= DATA_REG(2);
116              TX_Busy <= '1';
117              if (to_integer(CLK_CNT) = (CLK_PULSE_NUM - 1)) then
118                  CLK_CNT_RST <= '1';
119                  NSTATE <= BIT_3;
120              end if;
121
122          when BIT_3 =>
123              TX <= DATA_REG(3);
124              TX_Busy <= '1';
125              if (to_integer(CLK_CNT) = (CLK_PULSE_NUM - 1)) then
126                  CLK_CNT_RST <= '1';
127                  NSTATE <= BIT_4;
128              end if;
129
130          when BIT_4 =>
131              TX <= DATA_REG(4);
132              TX_Busy <= '1';
133              if (to_integer(CLK_CNT) = (CLK_PULSE_NUM - 1)) then
134                  CLK_CNT_RST <= '1';
135                  NSTATE <= BIT_5;
136              end if;
137
```

UART_TX:

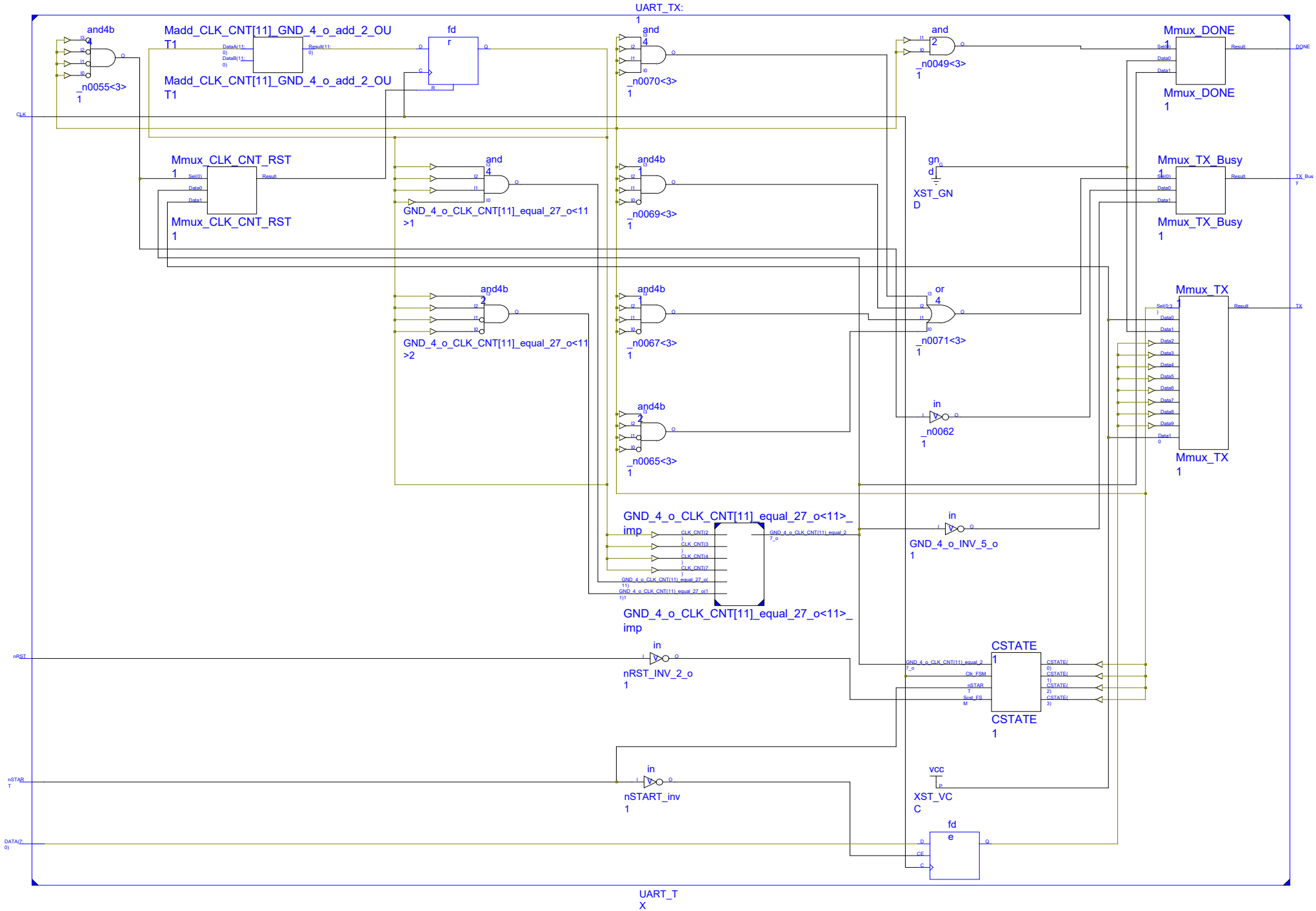
VHDL CODE

```
148          when BIT_5 =>
149              TX <= DATA_REG(5);
150              TX_Busy <= '1';
151              if (to_integer(CLK_CNT) = (CLK_PULSE_NUM - 1)) then
152                  CLK_CNT_RST <= '1';
153                  NSTATE <= BIT_6;
154              end if;
155
156          when BIT_6 =>
157              TX <= DATA_REG(6);
158              TX_Busy <= '1';
159              if (to_integer(CLK_CNT) = (CLK_PULSE_NUM - 1)) then
160                  CLK_CNT_RST <= '1';
161                  NSTATE <= BIT_7;
162              end if;
163
164          when BIT_7 =>
165              TX <= DATA_REG(7);
166              TX_Busy <= '1';
167              if (to_integer(CLK_CNT) = (CLK_PULSE_NUM - 1)) then
168                  CLK_CNT_RST <= '1';
169                  NSTATE <= STOP_BIT_SEND;
170              end if;
171
172          when STOP_BIT_SEND =>
173              TX <= '1';
174              TX_Busy <= '1';
175              if (to_integer(CLK_CNT) = (CLK_PULSE_NUM - 1)) then
176                  CLK_CNT_RST <= '1';
177                  NSTATE <= INIT_STATE;
178                  DONE <= '1';
179                  TX_Busy <= '0';
180              end if;
181
182          when others =>
183      end case ;
184  end process ; -- next_state
185
186  end Behavioral;
187
188
```

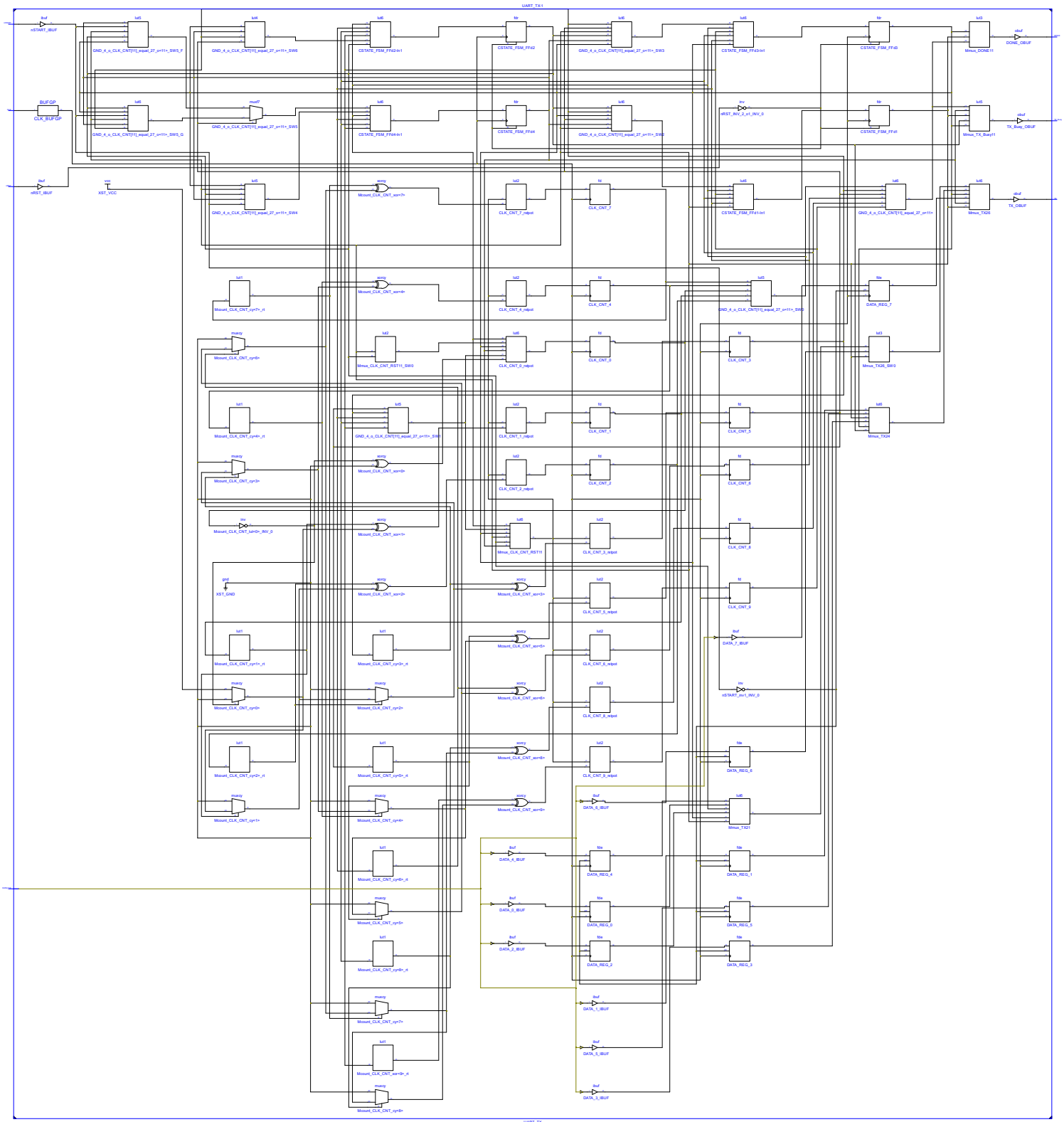
UART_TX Synthesize & RTL Schematic:



UART_TX Synthesize & RTL Schematic:



UART_TX Technology Schematic:



RAM_1024X8:

VHDL CODE

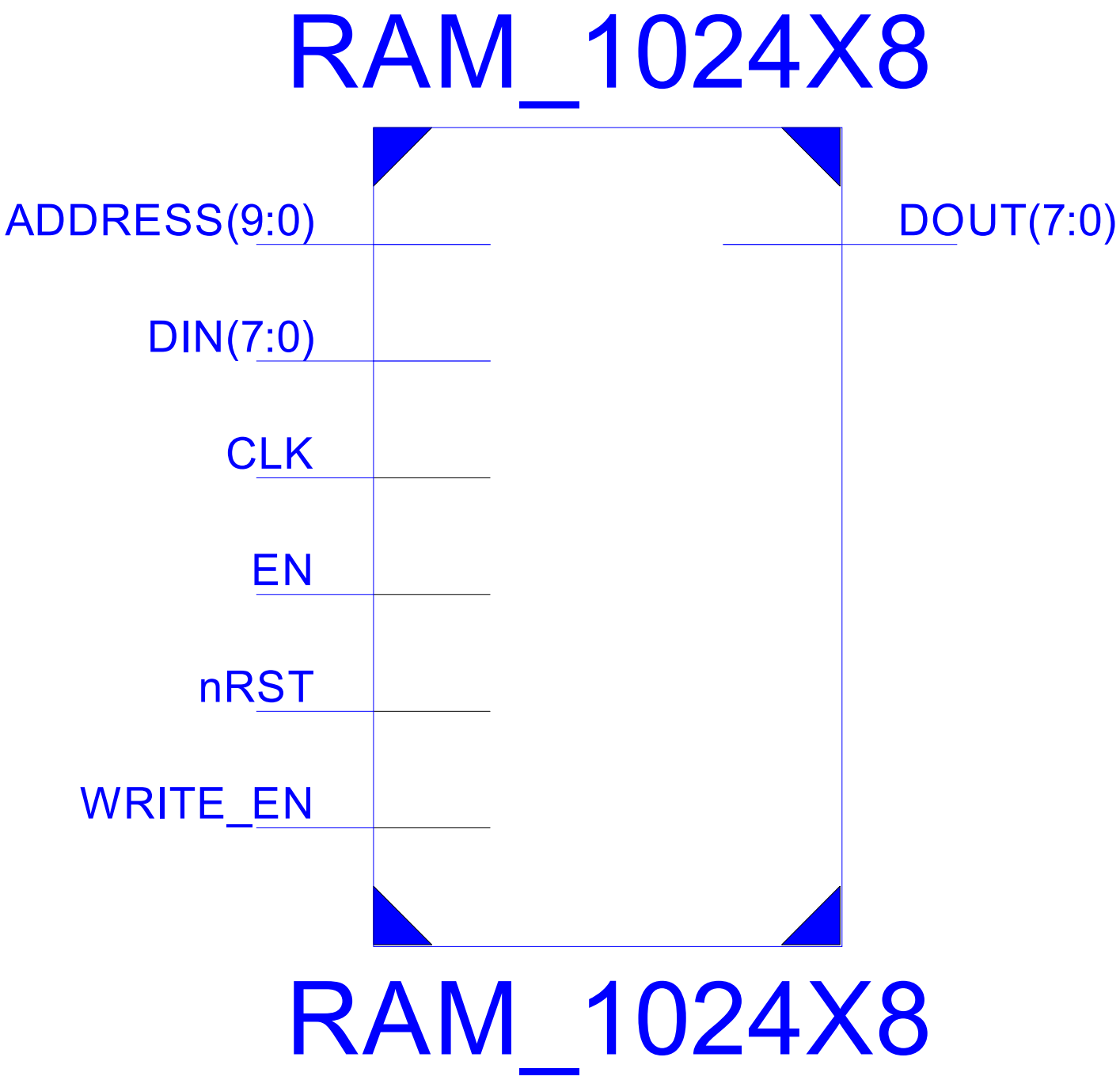
```vhdl
-- Engineer: Mohammad Niknam
-- Project Name:  UART_Controller
-- Module Name:   RAM_1024X8 - Behavioral
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity RAM_1024X8 is
    port (
        CLK : IN std_logic; --Clock Pulse
        nRST :in std_logic;
        EN : IN std_logic; --RAM Enable
        WRITE_EN : IN std_logic; --Write Enable
        ADDRESS : IN std_logic_vector(9 downto 0); --Address BUS for 1024 byte
        DIN : IN std_logic_vector(7 downto 0); --Input Data
        DOUT : OUT std_logic_vector(7 downto 0));
end RAM_1024X8;

architecture Behavioral of RAM_1024X8 is
    type MEMORY_TYP is array (integer range<>) of std_logic_vector(7 downto 0);
    signal MEM : MEMORY_TYP(0 to 1023);
begin

read_or_write: process(clk)
begin
    if (CLK'event and CLK = '1') then
        if (nRST='0') then
            if (EN='1') then
                if (WRITE_EN='1') then
                    MEM(to_integer(unsigned(ADDRESS))) <= DIN;
                else
                    DOUT <= MEM(to_integer(unsigned(ADDRESS)));
                end if;
            end if;
        else
        MEM <= (others => "00000000");
        end if;
    end if;
end process read_or_write;
end Behavioral;
```

RAM_1024X8 Synthesize & RTL Schematic:



UART_Controller:

VHDL CODE

```
1   -- Engineer: Mohammad Niknam
2   -- Project Name:  UART_Controller
3   -- Module Name:    UART_Controller - Behavioral
4
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.NUMERIC_STD.ALL;
8
9   entity UART_Controller is
10      port (  CLK : in std_logic;
11              nRST :in std_logic;
12              Serial_Receiver : in std_logic;
13              RX_Busy : out std_logic;  --data reception in progress
14              Serial_Sender : out std_logic
15              );
16  end UART_Controller;
17
18  architecture Behavioral of UART_Controller is
19
20  component UART_RX
21      Generic(CLK_PULSE_NUM : integer := 868); --868 for freq=100Mhz / BUADRATE 115200
22      port (  CLK : in std_logic;
23              nRST :in std_logic;
24              DATA_OUT : out std_logic_vector(7 downto 0);
25              DATA_VLD : out std_logic; -- when DATA_VLD = 1, data reception is
                completed data on DATA_OUT are valid
26              RX_Busy : out std_logic;  --data reception in progress
27              Received_serial : in std_logic);
28  end component;
29
30  component UART_TX is
31      Generic(CLK_PULSE_NUM : integer := 868); --868 for freq=100Mhz / BUADRATE 115200
32      port (  CLK : in std_logic;
33              nRST :in std_logic;
34              nSTART : in std_logic;
35              DATA : in std_logic_vector(7 downto 0);
36              DONE : out std_logic;
37              TX_Busy : out std_logic;  --data sending in progress
38              TX : out std_logic);
39  end component;
40
41  component RAM_1024X8 is
42      port (
43          CLK : IN std_logic; --Clock Pulse
44          nRST :in std_logic;
45          EN : IN std_logic; --RAM Enable
46          WRITE_EN : IN std_logic; --Write Enable
47          ADDRESS : IN std_logic_vector(9 downto 0); --Address BUS for 1024 byte
48          DIN : IN std_logic_vector(7 downto 0); --Input Data
49          DOUT : OUT std_logic_vector(7 downto 0));
50  end component;
51
52  type FSMTYPE is (INIT_STATE, DATA_Reception, DATA_send);
53
54  signal CSTATE, NSTATE : FSMTYPE;
55  signal nRST_signal : std_logic;
56  signal DATA_VLD, DATA_VLD_CNT_RST: std_logic;
57  signal DATA_VLD_CNT : unsigned(10 downto 0);
58  signal RAM_ADDRESS, RAM_ADDRESS_WRITE, RAM_ADDRESS_READ : std_logic_vector(9 downto
    0);
59  signal SEND_DONE, SEND_DONE_CNT_RST : std_logic;
60  signal SEND_DONE_CNT : unsigned(10 downto 0);
61  signal RX_DATA_OUT, RAM_DOUT: std_logic_vector(7 downto 0);
62  signal WRITE_EN, RAM_EN : std_logic;
63  signal TX_Busy, nSTART_signal : std_logic;
64
```

UART_Controller:

VHDL CODE

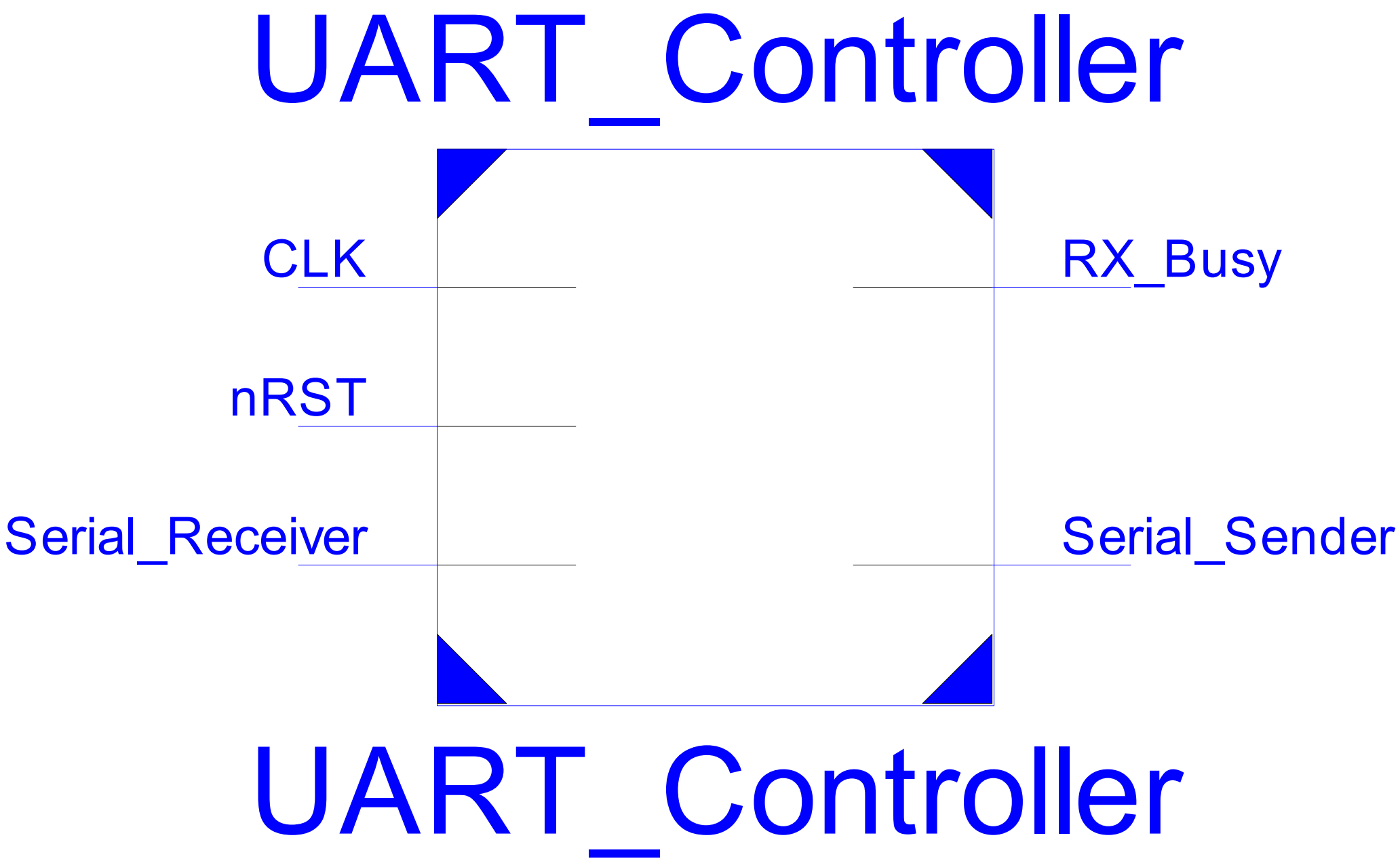
```
73  begin
74      DATA_VLD_counter_AND_SEND_DONE_counter : process( CLK )
75      begin
76          if (CLK'event and CLK = '1') then
77              if (DATA_VLD_CNT_RST = '1') then
78                  DATA_VLD_CNT <= (others => '0');
79                  RAM_EN <= '0';
80              else
81                  if (DATA_VLD = '1') then
82                      DATA_VLD_CNT <= DATA_VLD_CNT + 1;
83                      RAM_EN <= '1';
84                  else
85                      RAM_EN <= '0';
86                  end if;
87              end if ;
88
89              if (SEND_DONE_CNT_RST = '1') then
90                  SEND_DONE_CNT <= (others => '0');
91              else
92                  if (SEND_DONE = '1') then
93                      SEND_DONE_CNT <= SEND_DONE_CNT + 1;
94                      RAM_EN <= '1';
95                  else
96                      RAM_EN <= '0';
97                  end if;
98              end if ;
99          end if ;
100     end process ; -- DATA_VLD_counter_AND_SEND_DONE_counter
101
102
103     RAM_ADDRESS_WRITE_PRO: process(CLK, DATA_VLD_CNT)
104         variable DATA_VLD_CNT_var : unsigned(10 downto 0);
105     begin
106         DATA_VLD_CNT_var := DATA_VLD_CNT + "1111111111";   --DATA_VLD_CNT Minus One
        to address
107         RAM_ADDRESS_WRITE <= std_logic_vector(DATA_VLD_CNT_var(9 downto 0));
108     end process RAM_ADDRESS_WRITE_PRO; --RAM_ADDRESS_WRITE_PRO
109
110
111     RAM_ADDRESS_READ <= std_logic_vector(SEND_DONE_CNT(9 downto 0));
112
113
114     state_registration : process( CLK )
115     begin
116         if (CLK'event and CLK = '1') then
117             if (nRST = '0') then
118                 CSTATE <= INIT_STATE;
119             else
120                 CSTATE <= NSTATE;
121             end if ;
122         end if ;
123     end process ; -- state_registration
124
125
126     next_state : process( CSTATE, RAM_ADDRESS_WRITE, RAM_ADDRESS_READ, DATA_VLD,
        DATA_VLD_CNT, SEND_DONE_CNT, TX_Busy)
127     begin
128         NSTATE <= CSTATE;
129         nRST_signal <= '1';
130         DATA_VLD_CNT_RST <= '0';
131         SEND_DONE_CNT_RST <= '0';
132         WRITE_EN <= '1';
133         RAM_ADDRESS <= (others => '0');
134         nSTART_signal <= '1';
135
```

UART_Controller:

VHDL CODE

```
143         case( CSTATE ) is
144             when INIT_STATE =>
145                 nRST_signal <= '0';
146                 DATA_VLD_CNT_RST <= '1';
147                 SEND_DONE_CNT_RST <= '1';
148                 if (DATA_VLD = '1') then
149                     NSTATE <= DATA_Reception;
150                 end if ;
151 
152             when DATA_Reception =>
153                 WRITE_EN <= '1';
154                 RAM_ADDRESS <= RAM_ADDRESS_WRITE;
155                 SEND_DONE_CNT_RST <= '1';
156                 if (DATA_VLD_CNT = "10000000000") then  --10000000000=1024
157                     DATA_VLD_CNT_RST <= '1';
158                     NSTATE <= DATA_send ;
159                 end if ;
160 
161             when DATA_send =>
162                 WRITE_EN <= '0';
163                 RAM_ADDRESS <= RAM_ADDRESS_READ;
164                 DATA_VLD_CNT_RST <= '1';
165                 if (SEND_DONE_CNT = "10000000000") then  --10000000000=1024
166                     SEND_DONE_CNT_RST <= '1';
167                     NSTATE <= INIT_STATE ;
168                 else
169                     if (TX_Busy = '0') then
170                         nSTART_signal <= '0';
171                     else
172                         nSTART_signal <= '1';
173                     end if;
174                 end if ;
175 
176             when others =>
177         end case ;
178     end process ; -- next_state
179 
180 
181     UART_RX_Instantiation : UART_RX
182     Generic map(CLK_PULSE_NUM => 868)
183     port map(
184         CLK => CLK,
185         nRST => nRST_signal,
186         DATA_OUT => RX_DATA_OUT,
187         DATA_VLD => DATA_VLD,
188         RX_Busy => RX_Busy,
189         Received_serial => Serial_Receiver
190     );
191 
192     UART_TX_Instantiation : UART_TX
193     Generic map(CLK_PULSE_NUM => 868)
194     port map(
195         CLK => CLK,
196         nRST => nRST_signal,
197         nSTART => nSTART_signal,
198         DATA => RAM_DOUT,
199         DONE => SEND_DONE,
200         TX_Busy => TX_Busy,
201         TX => Serial_Sender
202     );
203 
204     RAM_1024X8_Instantiation : RAM_1024X8
205     port map(
206         CLK => CLK,
207         nRST => nRST_signal,
208         EN => RAM_EN,
209         WRITE_EN => WRITE_EN,
210         ADDRESS => RAM_ADDRESS,
211         DIN => RX_DATA_OUT,
212         DOUT => RAM_DOUT
213     );
214 end Behavioral;
```

UART_Controller Synthesize & RTL Schematic:



UART_Controller Synthesize & RTL Schematic: