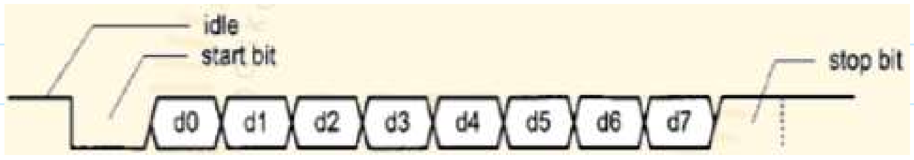
W03 - UART, the stop state

torsdag 7. mai 2020 08:13

1. In order to describe the behaviour of the modified circuit (that is "stop" state deleted altogether and "rx_done_tick<='1'" asserted immediately before leaving "data" state), I would first like to describe the operational workflow depicted in the ASMD chart for UART (RS-232). The ASDM chart can be found on p. 2 of this document



(an overview of start bit (logic 0), 8 data bits and stop bit (logic 1).)

STATE "IDLE"

The receiving subsystem is itself a finite state machine with datapath, so its behaviour can be represented by an ASMD chart as shown in the figure. The ASMD chart comprises only 4 states which are closely related to the RS-232 frame format (we're not using a parity bit in this case). First state IDLE is where the FSM remains after the stop bit while it is waiting for the next start bit('0') to occur. When that happens, the samling ticks counter is reset to 0 (S<-0) and moves onto the next state; start.

STATE "START"

In the start state, the sampling tick is incremented until the sampling ticks counter reaches 7; middle point of the start bit. Then it resets the sampling ticks counter and the data bits counter and transitions to next state; data.

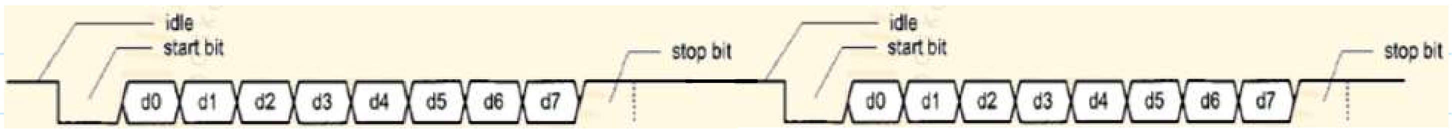
STATE "DATA"

Data state is remained until all data bits have been read. In the data state, we again increment the sampling ticks counter for every tick received from the baud generator, but now we wait until this counter reaches 15 instead of 7, since we want to go from the middle point of one data bit period to the middle point of the next data bit period. When the sampling ticks counter reaches 15, we know that it needs to be reset, and deserialize another bit.

Deserializing means to read the value that is present at the rX dataline and insert it into the deserializer register and check whether that is the last databit to be received. If not, then we must increment the databit counter and reenter the datastate until all databits have been read. In the middle of the bit period of last databit, the system-of-interest transitions to stop state.

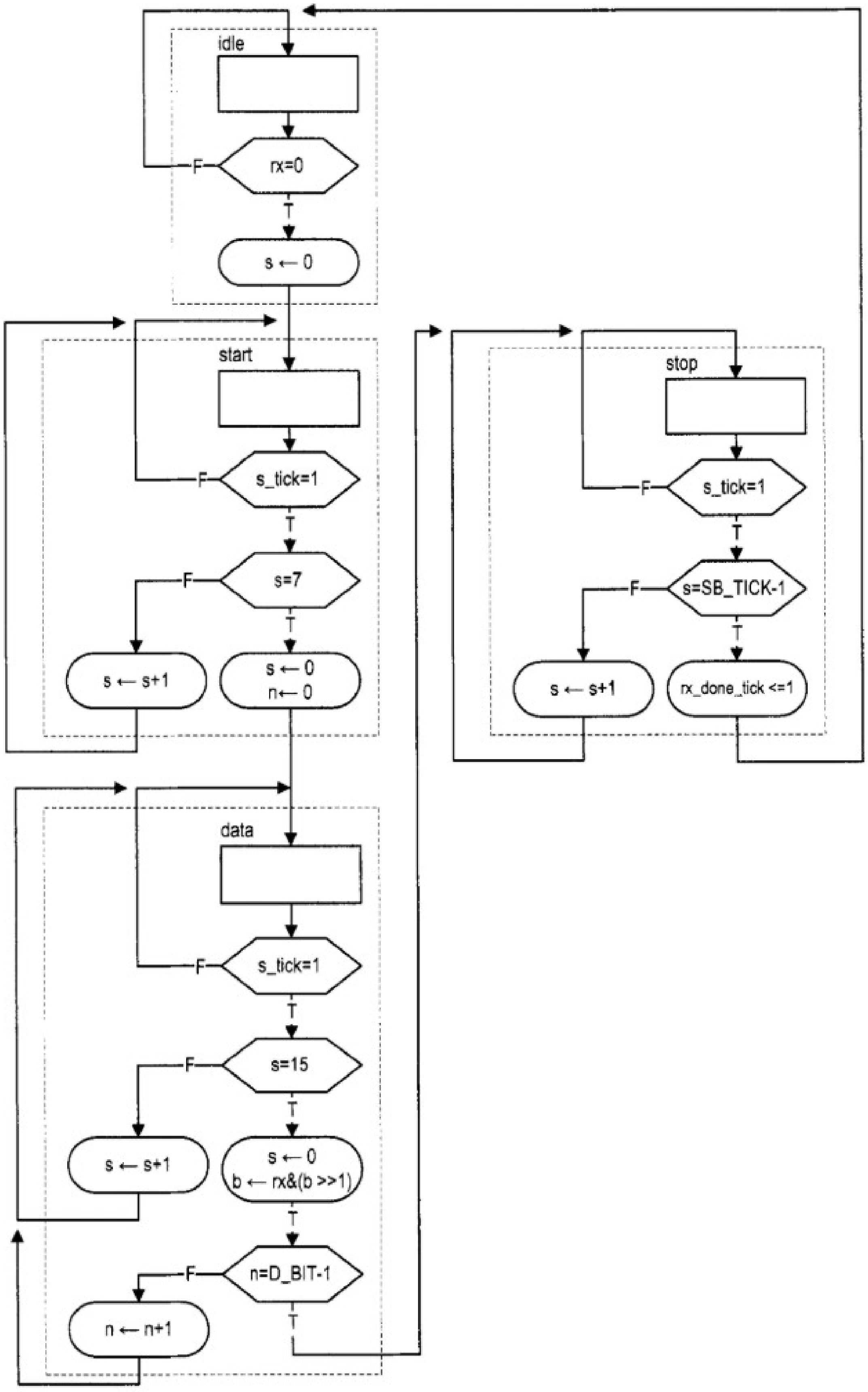
STATE "STOP"

In the stop state we again count the sampling ticks counter until reaching the middle point of the stopbit period. When that happens, we will assert the rx_done_tick to be equal to 1 for one clock cycle and restart the whole process. In other words the stop state provides a half bit period for the last databit to pass and another half bit period to reach to the middle-point of the stop bit. Then a "rx_done_tick" flag is set (logic 1). This means that the reception rightfully stops a half bit period after the last databit, which is enough time before a potential, next startbit can be received.



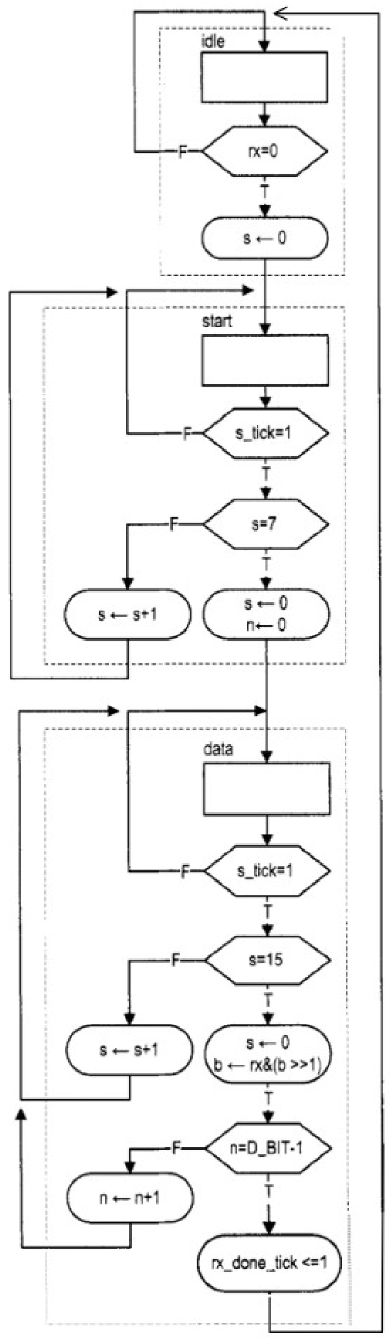
UART - ASMD chart

Friday, May 8, 2020 1:24 AM



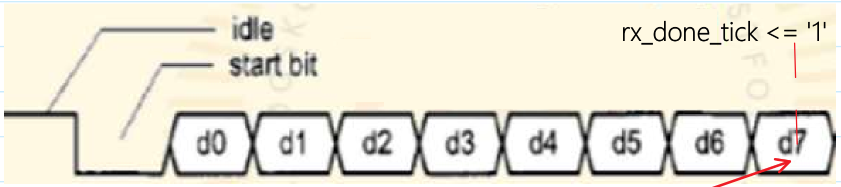
Q1: Changing the ASMD chart

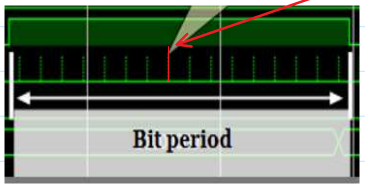
Friday, May 8, 2020 1:43 AM



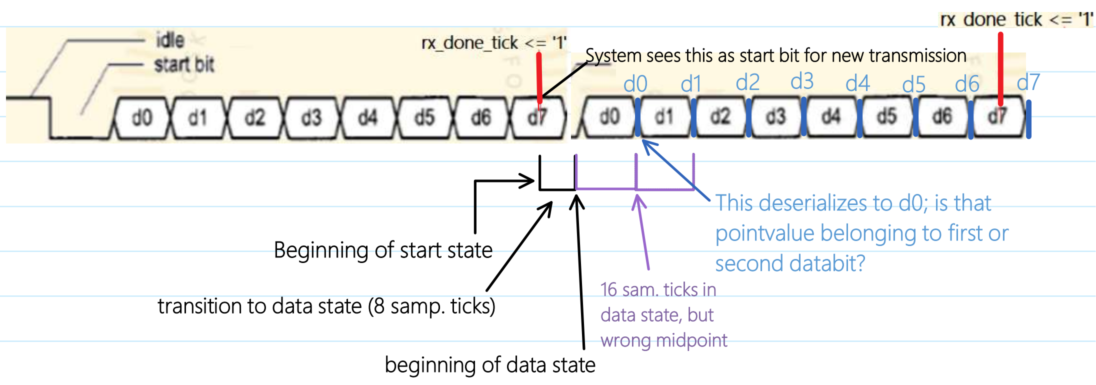
As can be seen from this figure, the "stop" state has been removed entirely, and the "rx_done_tick" flag has been set immediately before leaving "data" state.

This new operational workflow creates new problems;
- The "rx_done_tick" flag will be set while the reception is half bit period into the last databit. This is contradictive as the "rx_done_tick" flag is supposed to be set after reception is done.





Furthermore, if the case of the last databit being a logical 0 and the "rx_done_tick" flag set as depicted in the ASMD chart, the "idle" state will see the last databit ('0') as a startbit for new reception (rx=0). This creates a cascading effect; the real d0 of next reception will be misinterpreted as a start bit and the deserialization of bitstream will be phased by 0.5 bit period, leading to problematic deserializing of datastream.


rx_done_tick <= '1'
idle
start bit
rx_done_tick <= '1'
System sees this as start bit for new transmission
d0 d1 d2 d3 d4 d5 d6 d7
d0 d1 d2 d3 d4 d5 d6 d7
d0 d1 d2 d3 d4 d5 d6 d7
This deserializes to d0; is that pointvalue belonging to first or second databit?
Beginning of start state
transition to data state (8 samp. ticks)
16 sam. ticks in data state, but wrong midpoint
beginning of data state

Q2 - Modified VHDL code

Friday, May 8, 2020 6:33 AM

UART TOP MODULE (BAUDRATE GENERATOR+UART RX)

```
-- Listing 7.4
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity uart is
   generic(
     -- Default setting:
     -- 19,200 baud, 8 data bis, 1 stop its, 2^2 FIFO
     DBIT: integer:=8;      -- # data bits
      SB_TICK: integer:=16; -- # ticks for stop bits, 16/24/32
                          --   for 1/1.5/2 stop bits
      DVSR: integer:= 163;  -- baud rate divisor
                          -- DVSR = 50M/(16*baud rate)
      DVSR_BIT: integer:=8; -- # bits of DVSR
      FIFO_W: integer:=2    -- # addr bits of FIFO
                          -- # words in FIFO=2^FIFO_W
   );
   port(
      clk, reset: in std_logic;
  --    rd_uart, wr_uart: in std_logic;
      rx: in std_logic;
  --    w_data: in std_logic_vector(7 downto 0);
 --     tx_full, rx_empty: out std_logic;
      r_data: out std_logic_vector(7 downto 0)
--      tx: out std_logic
   );
end uart;

architecture str_arch of uart is
   signal tick: std_logic;
   signal rx_done_tick: std_logic;
  -- signal tx_fifo_out: std_logic_vector(7 downto 0);
   signal rx_data_out: std_logic_vector(7 downto 0);
 --  signal tx_empty, tx_fifo_not_empty: std_logic;
--   signal tx_done_tick: std_logic;
begin

   baud_gen_unit: entity work.mod_m_counter
      generic map(M=>DVSR, N=>DVSR_BIT)
      port map(clk=>clk, reset=>reset,
               q=>open, max_tick=>tick);

   uart_rx_unit: entity work.uart_rx(arch)
      generic map(DBIT=>DBIT, SB_TICK=>SB_TICK)
      port map(clk=>clk, reset=>reset, rx=>rx,
               s_tick=>tick, rx_done_tick=>rx_done_tick,
               dout=>rx_data_out);
```

```
--   uart_tx_unit: entity work.uart_tx(arch)
--       generic map(DBIT=>DBIT, SB_TICK=>SB_TICK)
--       port map(clk=>clk, reset=>reset,
--                tx_start=>tx_fifo_not_empty,
--                s_tick=>tick, din=>tx_fifo_out,
--                tx_done_tick=> tx_done_tick, tx=>tx);
--  tx_fifo_not_empty <= not tx_empty;

r_data <= rx_data_out;

end str_arch;
```

UART RX MODULE

```
-- Listing 7.1
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity uart_rx is
   generic(
      DBIT: integer:=8;     -- # data bits
      SB_TICK: integer:=16  -- # ticks for stop bits
   );
   port(
      clk, reset: in std_logic;
      rx: in std_logic;
      s_tick: in std_logic;
      rx_done_tick: out std_logic;
      dout: out std_logic_vector(7 downto 0)
   );
end uart_rx ;

architecture arch of uart_rx is
   type state_type is (idle, start, data);
   signal state_reg, state_next: state_type;
   signal s_reg, s_next: unsigned(3 downto 0);
   signal n_reg, n_next: unsigned(2 downto 0);
   signal b_reg, b_next: std_logic_vector(7 downto 0);

begin

   -- FSMD state & data registers
   process(clk,reset)
   begin
      if reset='1' then
         state_reg <= idle;
         s_reg <= (others=>'0');
         n_reg <= (others=>'0');
         b_reg <= (others=>'0');
      elsif (clk'event and clk='1') then
         state_reg <= state_next;
         s_reg <= s_next;
         n_reg <= n_next;
         b_reg <= b_next;
      end if;
   end process;

   -- next-state logic & data path functional units/routing
   process(state_reg,s_reg,n_reg,b_reg,s_tick,rx)
   begin
      state_next <= state_reg;
      s_next <= s_reg;
      n_next <= n_reg;
      b_next <= b_reg;
      rx_done_tick <='0';

      case state_reg is
         when idle =>
            if rx='0' then
               state_next <= start;
               s_next <= (others=>'0');
            end if;

         when start =>
            if (s_tick = '1') then
               if s_reg=7 then
                  state_next <= data;
                  s_next <= (others=>'0');
                  n_next <= (others=>'0');
               else
                  s_next <= s_reg + 1;
               end if;
            end if;

         when data =>
            if (s_tick = '1') then
               if s_reg=15 then
                  s_next <= (others=>'0');
                  b_next <= rx & b_reg(7 downto 1) ;

                  if n_reg=(DBIT-1) then
                     rx_done_tick <= '1';
                     state_next <= idle ;
                  else
                     n_next <= n_reg + 1;
                  end if;
               else
                  s_next <= s_reg + 1;
               end if;
            end if;
```

```
--         when stop =>
--           if (s_tick = '1') then
--              if s_reg=(SB_TICK-1) then
--                 state_next <= idle;
--                 rx_done_tick <='1';
--              else
--                 s_next <= s_reg + 1;
--              end if;
--           end if;
      end case;
   end process;
   dout <= b_reg;
end arch;
```

BAUDRATE GENERATOR

```
-- Listing 4.11
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity mod_m_counter is
   generic(
      N: integer := 8;     -- number of bits
      M: integer := 163     -- mod-M
  );
   port(
      clk, reset: in std_logic;
      max_tick: out std_logic;
      q: out std_logic_vector(N-1 downto 0)
   );
end mod_m_counter;

architecture arch of mod_m_counter is
   signal r_reg: unsigned(N-1 downto 0);
   signal r_next: unsigned(N-1 downto 0);
begin
   -- register
   process(clk,reset)
   begin
      if (reset='1') then
         r_reg <= (others=>'0');
      elsif (clk'event and clk='1') then
         r_reg <= r_next;
      end if;
   end process;
   -- next-state logic
   r_next <= (others=>'0') when r_reg=(M-1) else
             r_reg + 1;
   -- output logic
   q <= std_logic_vector(r_reg);
   max_tick <= '1' when r_reg=(M-1) else '0';

end arch;
```

UART TOP MODULE TESTBENCH

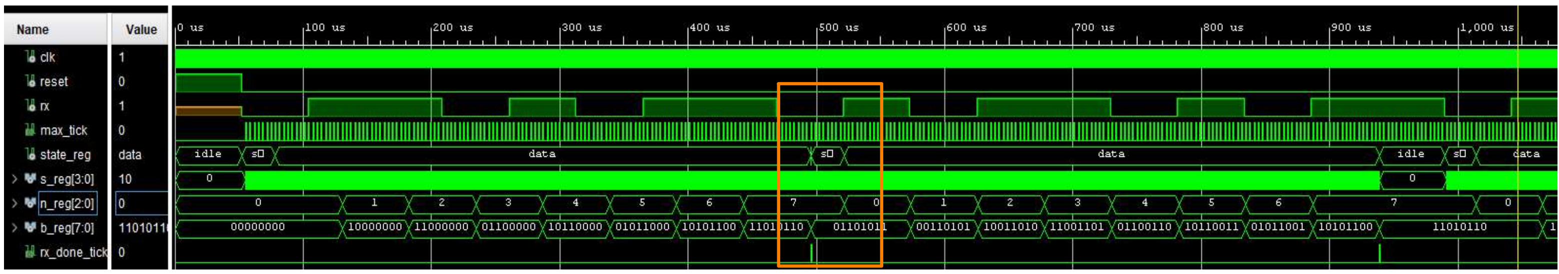
```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 05/08/2020 03:44:45 AM
-- Design Name:
-- Module Name: uart_tb - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity uart_tb is
--  Port ( );
end uart_tb;

architecture Behavioral of uart_tb is

-- Signals
signal clk, reset, rx : std_logic;
signal r_data : std_logic_vector( 7 downto 0);

constant clk_period : time := 20ns; -- 50MHz <--> 20ns period
constant bit_period : time := 52083.3333ns; -- 1 / baud rate (baud rate is actually a frequency (bits/second)), baudrate= 19200 b/s
constant clks_per_bit: integer := 2604; -- clk freq/baudrate = how many clocks per bit
constant max_tick_period : time := 3255.20833ns; -- 307200 ticks/s <--> period = 1/307200

begin

-- Unit under test:
    uut: entity work.uart
       port map(
            clk => clk,
            reset =>  reset,
            rx => rx,
          --  w_data => w_data,
           -- tx_full => tx_full,
           -- rx_empty => rx_empty,
            r_data => r_data
           -- tx => tx,
           -- rd_uart => rd_uart,
            --wr_uart => wr_uart
        );

-- Clock process
clk_process: process
begin
    clk <= '0';
    wait for clk_period/2;
    clk<= '1';
    wait for clk_period/2;
end process;




-- Stimulus process
stim_process : process
begin
    -- Resetting for two clock periods; 20ns.
    reset <= '1';
    wait for bit_period;
    reset <= '0';

    -- We want to send in a 10-bit stream to rx, in which first bit is startbit ('0'),
    -- following 8 bits are databits (last databit is a 0), and 10th bit is stop bit.
    -- I choose 0-11010110-1.

    rx <= '0';          -- STARTBIT
    wait for bit_period;
```

```
rx <= '1';          -- D0
wait for bit_period;

rx <= '1';          -- D1
wait for bit_period;

rx <= '0';          -- D2
wait for bit_period;

rx <= '1';          -- D3
wait for bit_period;

rx <= '0';          -- D4
wait for bit_period;

rx <= '1';          -- D5
wait for bit_period;

rx <= '1';          -- D6
wait for bit_period;

rx <= '0';          -- D7
wait for bit_period;

rx <= '1';          -- STOPBIT
wait for bit_period;
-------------------------------------------FIRST RECEPTION DONE, SECOND RECEPTION STARTS---------------------

rx <= '0';          -- STARTBIT
wait for bit_period;

rx <= '1';          -- D0
wait for bit_period;

rx <= '1';          -- D1
wait for bit_period;

rx <= '0';          -- D2
wait for bit_period;

rx <= '1';          -- D3
wait for bit_period;

rx <= '0';          -- D4
wait for bit_period;

rx <= '1';          -- D5
wait for bit_period;

rx <= '1';          -- D6
wait for bit_period;
```

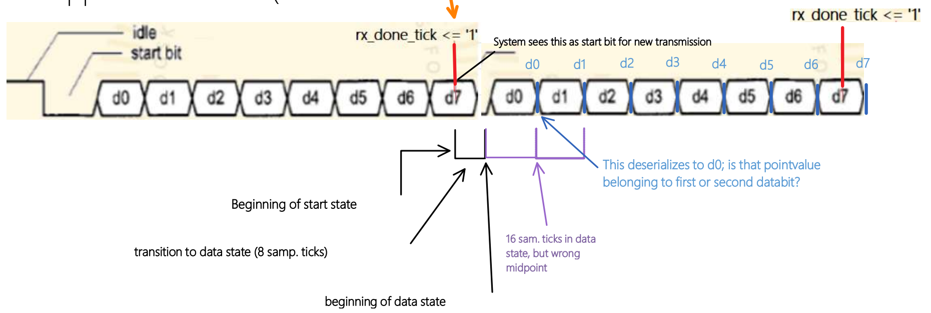
```
    rx <= '0';          -- D7
    wait for bit_period;

    rx <= '1';          -- STOPBIT
    wait for bit_period;

end process;

end Behavioral;
```

Q2: Screenshot of testbench

Friday, May 8, 2020 3:20 AM



As predicted earlier, the system interprets the last databit as a beginning of the start state which results in wrongful deserialization of the input datastream. Note how a zero is cascaded to b_reg (digits shifted to the right) at 520µs, when it is supposed to be a 1 (as seen during the correct first reception).



idle
start bit
d0
d1
d2
d3
d4
d5
d6
d7
rx_done_tick <= '1'
System sees this as start bit for new transmission
rx_done_tick <= '1'
This deserializes to d0; is that pointvalue belonging to first or second databit?
Beginning of start state
transition to data state (8 samp. ticks)
16 sam. ticks in data state, but wrong midpoint
beginning of data state