ECE 3216: Computer Systems Design Lab

Lab 3: RS232 Signal Generation - Hardware

Anthony Lam, Robert Campbell

28 January 2020

Objective: The goal of this lab is to investigate RS232 signaling. This is accomplished by generating and capturing RS232 signals that are generated with different parameters, such as different baud rates and parity schemes.

Equipment:

-Spartan 3 FPGA Board

-RS232 to USB cable

-Xilinx ISE

-PuTTY terminal on Windows

-BlueFruit Bluetooth Radio

-Oscilloscope

Procedure:

The SiLabs IDE was used to write code to generate and transmit a message over the BlueFruit radio. In addition to that, to allow the board to be compatible with the RS232 signal, the native clock was divided in the program to have 9600 baud given a 100MHz clock. In order to test that the code was functioning properly, an oscilloscope was used to verify the design. As the messages were being sent, PuTTY was used to verify that the correct message was transmitted.

Data and Observations:

main.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity main is
    port ( clock : in STD_LOGIC;
            baud : out std_logic;
           output : out STD_LOGIC);
end main;

architecture Behavioral of main is

-- Array declaration
type myROM is array(natural range <>) of std_logic_vector(7 downto 0);
signal text: myROM(1 to 5) := (x"68", x"65", x"6c", x"6c", x"6f");

-- Message format
signal message: std_logic_vector(9 downto 0);

-- Counter signal declarations
signal counter : integer range 0 to 10420 := 0;
signal bdclk : std_logic := '0';
signal nxtchar: std_logic_vector(7 downto 0);

begin
-- Clock divider to create a baudrate of 9600 given a 100MHz clock
process(clock)
begin
    if rising_edge(clock) then
        counter <= counter + 1;
```

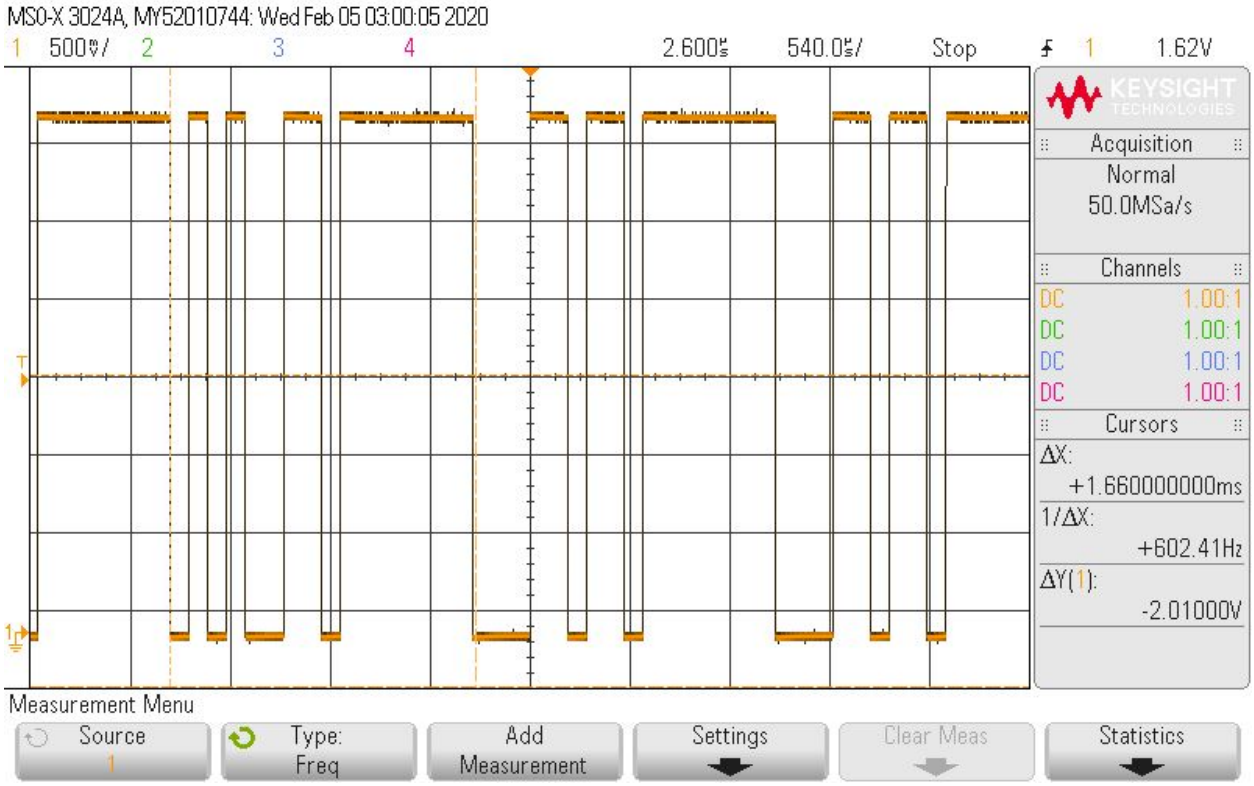
```
        if counter < 5208 then
            bdclk <= '1';
            baud <= '1';
        else
            bdclk <= '0';
            baud <= '0';
        end if;

    end if;
    if (counter > 10415) then
        counter <= 0;
    end if;
end process;

-- Message generation and transmission
process(bdclk)
variable index : integer range 0 to 15 := 0;
variable txtin : integer range 1 to 7 := 1;
begin
    if rising_edge(bdclk) then
        if(index = 0) then
            nxtchar <= text(txtin);
            txtin := txtin + 1;
            index := index + 1;
            output <='1';
        elsif (index = 1) then
            output <= '0';
            index := index + 1;
        elsif(index <10) then
            output <= nxtchar(natural(index - 2));
            index := index + 1;
        else
            output <= '1';
            index := index + 1;
```

```
        end if;
        if(txtin > 5) then
            txtin := 1;
        end if;
    end if;
end process;

end Behavioral;
```



The code uses the FPGA's board clock and divided it down in order to create a 9600 baud clock. The message generation and transmission code is then used to index the message and transmit the message one at a time. The output was verified on the scope and PuTTY.

Analysis and Discussion:

This lab was successful. The generated signals from the FPGA matched expectations, with the message “hello” being correctly generated and observed onto the PuTTY terminal and oscilloscope. Analysis is not applicable in this experiment.

Questions:

How are PC Baud rates generated?

Baud rates were historically generated by an integer clock division of the original IBM PC’s internal clock of 115200 Hz.

How are Baud rates generated in your FPGA?

The baud rates are generated as half the clock from Timer1 divided by 256 (using the high byte of Timer1).

What is the percent difference in the PC baud rate and the baud rate of your designs?

There is a 0.016% difference in the PC baud rate and the UART baud rate.

What is the maximum allowed difference in Baud rates at the rate you choose, demonstrate your answer with a plot, annotation, and text?

If the baud rate if the receiver is faster or slower than the transmitter baud rate, this causes the first data bit to be sampled slightly before or after its center by the baud rate differences. This continues on with each additional bit where the error essentially cumulatively becomes larger until a situation may occur in which the receiver samples a bit twice. A 25% error range on the center is generally safe for communication.

115200 N81

$9T_P + 0.5T_P - 9T_R - 0.5T_R <= 0.25T_P$

$1/T_R <= 112246$ Baud

$1/T_R <= 118314$ Baud

The baud rate range where an error would likely not occur is between 112246 baud and 118314 baud.

Conclusion:

This experiment was to gain familiarity with generating RS232 signals from data that is stored in parallel, similar to how a UART functions. The generated signal met expectations, but there were some difficulties in reading the signal on the PC. The PC had difficulty locking onto the baud rate that the Spartan was generating with the original design only having 2 stop bits. As such the terminal displayed a wide variety of characters that was not the data in our text array.

The solution that we eventually used was to massively increase the mark time, so that the PC could lock onto when the start-bit occurred.