LABORATION

COMPUTER DESIGN TSEA83

UART

Version: 1.1 English
2013 1.0 (OS)
2016 1.1 (AN)
2019 1.1 English (PK)

Name and personal ID number	Passed

blank page

Contents

blank page

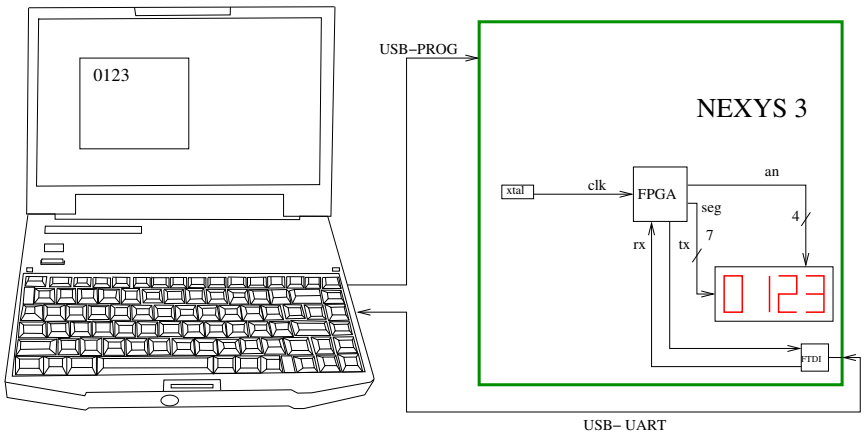


Figure 1: Lab setup. Numbers are sent serially from the computers keyboard to the FPGA board.

1 Introduction

Numbers entered from a computer keyboard shall be sent over an RS232 connection, to be presented on a 7-seg display on the FPGA board Nexys-3, see fig. 1.

1.1 Purpose

In this lab, you will learn to:

- use the FPGA board Nexys 3, that you will use in your project.
- write VHDL code, related to both block diagram and timing diagram.
- simulate your construction with ModelSim.
- synthesize and test your construction.

1.2 Preparations

When you enter the lab, you must be prepared. The tasks you need to demonstrate are marked by an index finger. Some of them should be done in advance, as a home exercises.

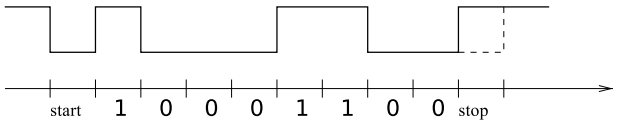


Figure 2: Transmission of the digit 1. ASCII code 0x31 is sent with LSB first.

2 Theory

2.1 RS232

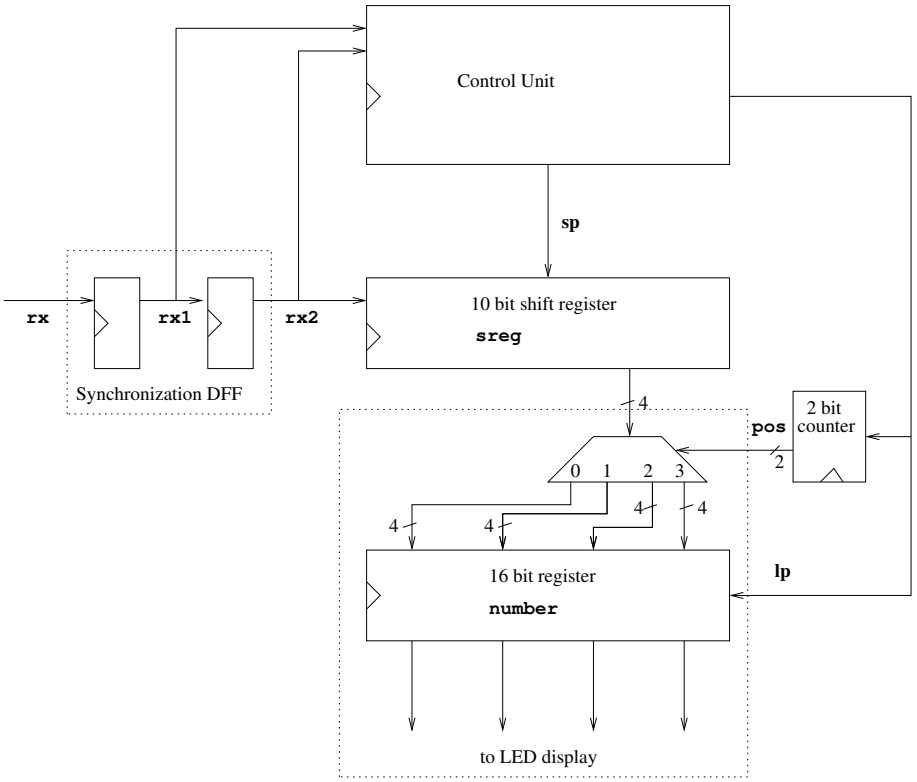
RS232 is an (old) standard for serial communication between a sender and a receiver. In many cases data is sent in both directions in the same time, so called full duplex. A circuit that can handle this is called UART (universal asynchronous receiver transmitter). In this lab only the receiver on the FPGA board will be implemented. Beside this, only digits will be decoded. The FPGA board is equipped with an FTDI chip (google it!), so a USB cable can be used. The serial signals `rx` (receive) and `tx` (transmit) are only available inside the FPGA, see fig. 1.
RS232 is an asynchronous and slow transmission protocol. Each character is sent as ten bits. First there is a start bit (always `0`). Then comes the eight data bits, least significant bit (LSB) first. Finally there is a stop bit (always `1`). See fig. 2. In this lab, the bit rate (baud rate) is 115200 bits/s. This means that each bit is 8.68μs long, or 868 pulses with a 100 MHz clock.

2.2 Partitioning

A block diagram with related timing diagram for our UART is shown in fig. 3. The clock frequency is 100 MHz. The construction is decomposed into the following blocks:

- *Synchronization DFF*. Synchronizes the input to the clock.
- *Control Unit*. This produces two control signals, both one pulsers:
  - The shift pulse `sp`, which comes in the middle of each bit.
  - The load pulse `lp`, which come after the last shift pulse.
- *Shift register*. The 10 bits of each digit is shifted in.
- *Register* for four digits. This is loaded by the load pulse. In the same time, a 2-bit counter is incremented.

a)



b)

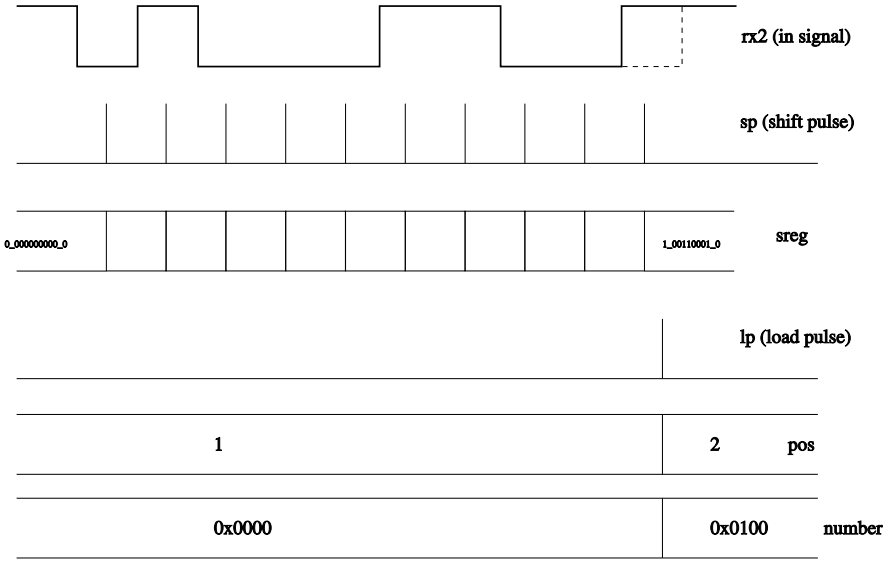


Figure 3: a) Block diagram. b) Timing diagram. The digit 1 is received and written to position 1 in the display.

2.3 Good Advises on VHDL

2.3.1 Clocked Processes

All clocked processes must look like this. We recommend that you simply use this code as a template. Row 1 and 2 must look as below. Row 3 and 4 should be there. The signal `clk` has the frequency 100 MHz. You must not clock on anything else than `clk`.

```
process(clk) begin
  if rising_edge(clk) then
    if rst='1' then
      <init>
    elsif <condition> then
      <do something>
    else
      <do something else>
    end if;
  end if;
end process;
```

3 Tasks

3.1 Home Exercise - Coding

Start with downloading the lab skeleton `lab_uart.tgz` from the home page [1]. Unpack it with `tar xzvf lab_uart.tgz`.

Translate the block diagram in fig. 3 to VHDL. Use the same signal names as in the figure. Before you continue, let the lab assistant pass your code. Code that is not easy enough to read might be rejected.

3.1.1 Easy to Read Code

The block diagram in fig. 3 should be a natural part of the construction. Hence, the diagrams should be translated to VHDL code with the same structure. I.e., translate according to:

- *Synchronization DFF*. 1 process.
- *Control Unit*. 1 or 2 processes.
- *Shift register*. 1 process.
- *16 bit register + demultiplexer*. 1 process.
- *2 bit counter*. 1 process.

3.2 Task - Simulation

Simulate the construction by running the command:

```
make lab.sim
```

Don't forget the `module add TSEA83`. Right click on `lab_tb` up to the left in the ModelSim window and select `Add->To Wave->All Items in design`. A wave window will pop up. Then type `run 500 us` in the Transcript pane in the ModelSim window.
The top test bench is `lab_tb.vhd`. Find out what this does! You should see the same figures as in fig. 3.
Before you continue, let the lab assistant pass your simulation.

3.3 Task - Demonstration

Make a bit file from the construction by running the command:

```
make lab.bitgen
```

Now you can program the FPGA board using:

```
make lab.prog
```

Start the terminal program `gtkterm` and configure according to:

parameter	value
Port	/dev/ttyUSB0
Baud Rate	115200
Parity	None
Bits	8
Stop bits	1
Flow control	none

It should now be possible to type digits to the display.

References

[1] Ragnemalm,Seger: *Digital konstruktion TSEA43*, http://www.da.isy.liu.se/courses/tsea83

A User Constraint File

The content on the file `lab.ucf`:

```
##################################################
# Define Device, Package, And Speed Grade
##################################################
#
CONFIG PART = xc6slx16-3-csg324;

##################################################
# clk, rst
##################################################
##Clock signal
Net "clk" LOC=V10 | IOSTANDARD=LVCMOS33;
Net "clk" TNM_NET = sys_clk_pin;
TIMESPEC TS_sys_clk_pin = PERIOD sys_clk_pin 100000 kHz;

Net "rst" LOC = B8 | IOSTANDARD = LVCMOS33;

#######################################
# Multiplexed display                 #
#######################################
## 7 segment display
Net "seg<7>" LOC = M13 | IOSTANDARD = LVCMOS33;
Net "seg<6>" LOC = T17 | IOSTANDARD = LVCMOS33;
Net "seg<5>" LOC = T18 | IOSTANDARD = LVCMOS33;
Net "seg<4>" LOC = U17 | IOSTANDARD = LVCMOS33;
Net "seg<3>" LOC = U18 | IOSTANDARD = LVCMOS33;
Net "seg<2>" LOC = M14 | IOSTANDARD = LVCMOS33;
Net "seg<1>" LOC = N14 | IOSTANDARD = LVCMOS33;
Net "seg<0>" LOC = L14 | IOSTANDARD = LVCMOS33;


Net "an<0>" LOC = N16 | IOSTANDARD = LVCMOS33;
Net "an<1>" LOC = N15 | IOSTANDARD = LVCMOS33;
Net "an<2>" LOC = P18 | IOSTANDARD = LVCMOS33;
Net "an<3>" LOC = P17 | IOSTANDARD = LVCMOS33;

## Usb-RS232 interface
Net "rx" LOC = N17 | IOSTANDARD=LVCMOS33;
```