Exercise 11.

Measurement of programmable peripheral units

Required knowledge

- Digital design knowledge.
- Verilog knowledge.
- Xilinx ISE (Exercise 3 & 10).
- SPI & UART communication standards (see Serial Communication Standards documentation).

Goal of the measurement

The goal of this lab is (1) to study and/or implement the UART and SPI communication protocols, (2) to implement a typical peripheral interface (3) get more experience about CAD aided hardware design.

Preparing for the measurement

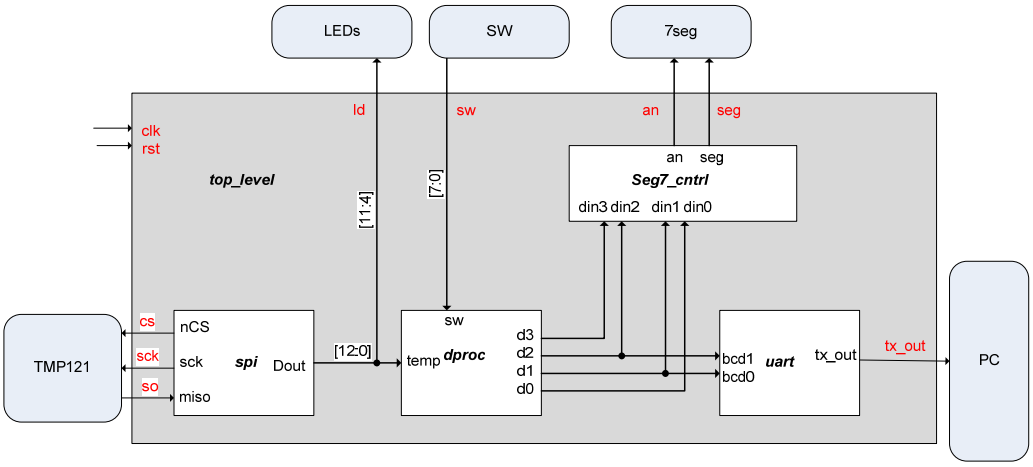
This measurement is based on earlier exercises. Scrutinize what you have learned in “Digital devices basics” and “Implementation and analysis of sequential networks”!

Read the papers available on the homepage: “Serial Communication Standards”, “Design of an SPI receiver”

Check the measurement tasks and the test questions!

Measurement tasks

The block diagram of the complete hardware to implement is depicted below. All of the modules use the system clock and reset input.

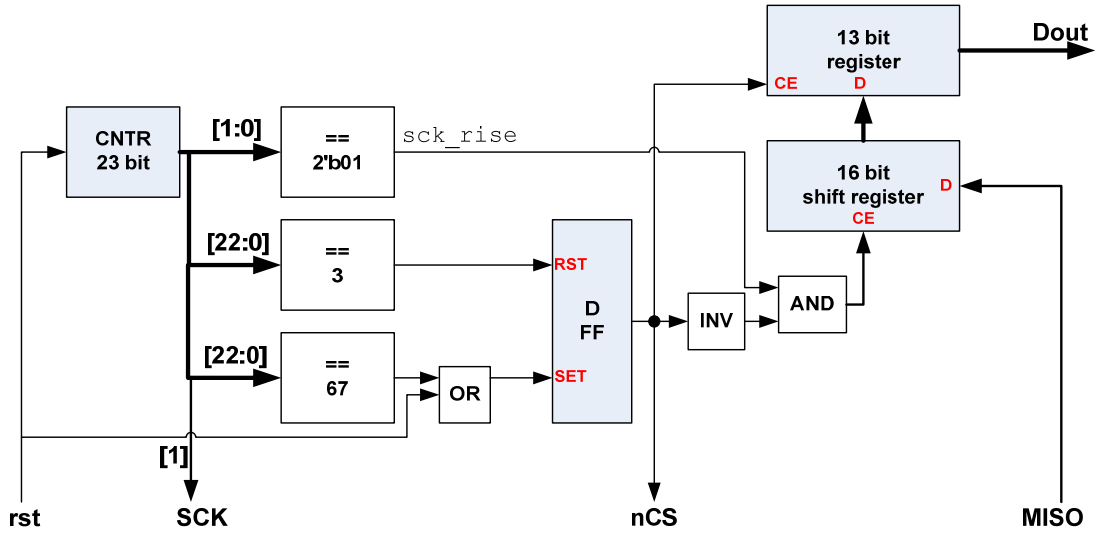


The following skeleton files can be downloaded from the homepage:

- In *top_level* module every sub-module but the UART is instantiated.
- *Seg7_cntrl*: The 7-segment display controller module you have designed earlier. The only improvement is that beside the BCD codes, this controller is able to display the minus sign on a digit (if the input is 13d) or to switch the digit off (for any other non-BCD input). The decimal point for the segment *din1* is always on, for all the other segments, it is always off. The internal enable signal is generated in a more efficient way as before. You have nothing to modify in this module, but examine the its new features and modifications.
- *spi*: the skeleton for the SPI receiver to implement. Initially *nCS* = 1, *sck* = 0, *Dout* = 13'b0. This should be modified, and Dout should contain the 13 bit temperature value. The integer part will be displayed on the LEDs.
- The *tb_spi_temp* is a test bench for the SPI module. During the measurement, you hace to examine and use this. (There is nothing to modify.)
- The input for the *dproc* module is the temperature and the value of the switches. The output is a 4 digit BCD value. Depending on sw[7], the output represents directly sw[6:0] or temperature-sw[6:0]. During the measurement, you have to examine this module and to do a little modification.
- The *pins.ucf* contains the pinout for our hardware. There is nothing to modify.
- The uart is added to the project as a black-box module.
- *cs.zip*: a chip scope configuration file. Initially, do not add this to your project!

1. SPI receiver: implementation, simulation and test

Based on the description made in the paper "Design of an SPI receiver", selecting SCK for 4 MHz-re, and considering that the system clock is 16 MHz, the temperature conversion time is 130 ms, and 13 of 16 transferred data bits are relevant, we get the following block diagram:



1.1. Examine the block diagram! Is the specification of TMP121 fulfilled? How long is one communication cycle? What are the 3 comparators for? Is the Dout output always consistent?

1.2. Implement the SPI module and check its functionality with simulation! You should understand the structure of the given test bench file! How is the temperature sensor simulated? How can you see on the simulation waveforms, wether your module is functioning well?

1.3. Generate and download the bit file, check the functionality without offset temperature (sw[6:0]=0)! Set up 42 deg. offset! What have you experienced? Supplement the dproc module in order to correct this error?

2. Examining the UART protocol

2.1. Instantiate the UART black box in your project, check the communication by means of a PC terminal program!

2.2. Analyze the UART communication with ChipScope. Display the transfer of one 3-byte data packet! The ChipScope configuration is included in *cs.zip* .

Use the time of the ChipScope synthesis for report writing and thinking over the remaining tasks!

In this ChipScope configuration, two trigger ports are used with a basic/w edges and a basic Match unit, respectively.

TRIG0 connections: SPI clock, chip select and data input (*cntr<1>, cs_ff, so_IBUF*), UART data output, and internal bit counter (*tx_shr<0>, tx_cntr*)

To the TRIG1 trigger port, only the UART enable signal is connected. This is used for the appropriate baud-rate generation (*tx_en*). During the analysis, you can use it as a Storage qualification condition.

3. Optional task

3.1. Evaluate the binary-BCD converter implemented in the *dproc* module! What is the used conversion algorithm? How many clock cycles does a conversion take? When is a new conversion started? What is the role of the following registers: *data_old, data_out, data_conv* és *data_high?* What are their values before and after a conversion?

Test questions

1. Which serial protocol needs clock to be transmitted?
2. Why it is hard to transfer some Mbit/s using UART transfer?
3. Which serial protocol doesn't support bus topology?
4. How to control the four digit 7-segment display?
5. What kind of setup parameters the users have, when configuring a serial port on the PC?
6. We are using a 115200 baud UART connection, with 8N1 setup. (8 bit data, no parity, 1 STOP bit). How long does it take to transfer 100 bytes?
7. If 115200 baud, 8N1 UART is used what could be the difference of a single bit time measured on transmitter and receiver, if successful transmission is needed?
8. What is START bit on UART protocol? What is the polarity?
9. What does odd parity means? What would be the parity value transmitted when the data is 0xAB (HEX)?
10. How many wires do we need to interface 4 SPI peripherals?
11. Give -42h in binary two's complement form!