Martin Skriver
maskr@mmmi.sdu.dk
RMEMB1 fall 2018

Lab assignment part 1/2

In this exercise it is only allowed to have the pl_top.vhd module instantiated in the block design.

Journal for this exercise is combined with the one from part 2/2 so please take notes, screenshots and measurements for that.

IMPORTANT:

- DO ONLY USE 3.3 V WHEN WORKING WITH THE SoC.
- Only connect RX(SoC output), TX(SoC input) and GND from the FTDI cable to the SoC. DO NOT connect the VCC wire to the FPGA.

Materials:

- TE0722 propeller SoC board
- TE0790 XMOD programmer
- Breadboard
- FTDI cable (TTL-232R-3V3-WE)

Exercise 1: UNITY link

- Create a new project.
- Add the processor IP and follow the guide for setting up the peripherals. Set the PL clk frequency to 200 MHz.
- Add the source code from the code.zip file to the project and add the pl_top.vhd module to the block design. Connect the 200 MHz clock input to the module and make the others external.
- Set the rx, tx and ground wires of the FTDI to pins next to each other and set the R, G, and B signals to the RGB LED connected pins.
- Follow the guide on how to export the project to SDK and program the flash memory and verify that the LED is blinking white.
- Setup a serial connection and check if you can read and write to the registers (e.g. read register 02 “#R:02” or write to register 04 “#W:0400000001”). The baud rate should be set to 3 MHz. The serial_test.py can be used for testing.

Exercise 2: Instantiation of components in modules

We can only control the RGB LED colour in a 3 bit resolution and not control the brightness with this setup. To change that we can add a PWM signal to each of the R, G and B colours that can be controlled by the terminal on the computer.

- Change the unity_ctrl.vhd file to output 3 times 8-bit PWM duty cycles and connect them to a read register (e.g. 4, 5 and 6).
- Connect the duty cycle signals from unity to PWM generators (see blackboard) and connect the 1 Hz signal to the PWM generators enable input. Instantiate the PWM generator to have a low PWM frequency but still high enough so the naked eye cannot see it is flashing.
- It is possible to make the resolution even better by increasing the PWM resolution to e.g. 12 bit. Since it is not always desired to have an 8 bit PWM generator we would like to make the PWM generator even more generic. Add a generic parameter that decide the resolution of the PWM generator. Chose a different resolution and change the std_logic_vectors that are connected to the module.

Using generics: https://www.ics.uci.edu/~jmoorkan/vhdlref/generics.html

Exercise 3: Sigma delta and simulation

Other ways to create analog signals can be by using sigma delta. This have the advantage of being much faster than the PWM generator that are limited by the clock frequency and the resolution. The sigma delta have some disadvantage about varying frequency that potentially can disturb other electronic component.

- Create a module that contains a low pass averaging filter.
- Create a simulation file in Vivado and simulate the module to verify it works. Use a varying input signal to verify that it works under all conditions.
- Create a decimator that reads the output data rate from an input port and input and a set value to determining if the output should be high.

(This will be used in the next lab exercise when we add ADC’s to give a reference input)