SDU	**Digital Design and Signal Processing - DDS Assignment #3**	IC	26.01.24

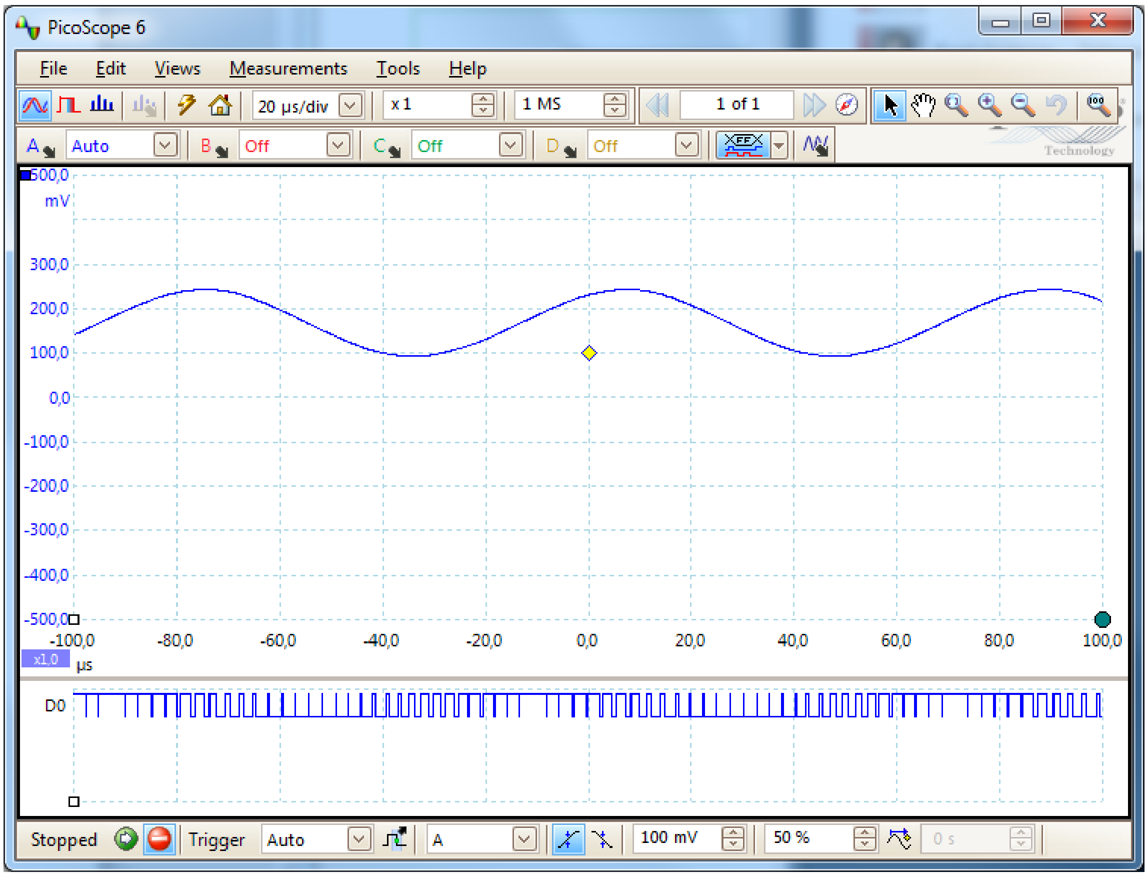
The purpose of this assignment is to work with some basic signal processing elements and implement them in hardware. You will:

- Create a signal generator, – implementing a sinewave in this case, but the same method can be used to generate any digital representation of an analog signal.

- Use the dip-switches on the ZEDBOARD as inputs to select different frequencies for “reading out” the sinewave – and thereby learn how to define the pinout for the target component of the VHDL model.

- Create a PWM generator and use it as a digital to analog converter with the help of a passive low pass filter connected to the ZEDBOARD, - thus being able to monitor the generated signal on an oscilloscope. Se picture below – a sinewave as the output from the low pass filtering of a PWM signal.

All designs will be implemented in the Xilinx Vivado development tools with the ZEDBOARD as the target.



Question 1:
Normally, in signal processing applications, an external signal is sampled by an analog to digital converter, and further processed in the hardware or software. In this case, the sinewave is made by filling the sampled version of the analog signal into a ROM in the FPGA.

- Create a ROM with 32 samples (constant data) – 8 bit resolution – representing one period of a sinewave. Document via relevant simulations that it functions as expected.

Question 2:
A PWM (Pulse Width Modulation) signal is a "pulse pattern", where the pulse width represents an amplitude (or value) of an analog signal. The signal has a basic frequency but the duty cycle changes according to the amplitude it represents.

- Create a PWM generator, with 8 bit resolution, - that is: using an 8-bit counter - and document the functionality with relevant simulations.

Question 3: Making it work!
Now the task is to connect the ROM with the PWM module and thereby create a PWM sequens representing the sinewave. To do this, it is necessary to create a control circuit for controlling the "read-out" from the ROM, "feeding" the PWM module and controlling the frequency of the signal.

The control module can be made as a counter, where the counter output is used as the address input to the ROM plus a clock divider for defining the frequency.

- Calculate the maximum frequency of the sinewave signal, based on the clock frequency of the ZEDBOARD.

- Create a Sine Wave Generator module, based on the ROM from Question 1 and a clock divider and counter for reading out the sampled values. Design the module with an 8 bit input vector for the purpose of choosing different signal frequencies.

- Connect the Sine Wave Generator with the PWM module.

- Verify the design by relevant simulations for all modules.

Question 4: Connecting to the outside world.
There are 8 dip switches on the ZEDBOARD. They will now be used as inputs to the Sine Wave generator for the purpose of choosing different frequencies of the signal. The PWM signal will be connected to an output pin and an external filter, and your design must then be downloaded to the ZEDBOARD. To do this you must first add a constraints file to your design – this defines the outline of pin connections.

- Create the constraints file and synthesize your design. Control that the pinout is correct

- Generate a bit file and download your file to the ZEDBOARD. Make relevant measurements in the lab and show (document) that your design can produce a sinewave with different frequencies.

- Try other waveforms – e.g. a square wave etc.

The constraints file is listed below:

```
set_property PACKAGE_PIN Y9              [get_ports Clk]
set_property IOSTANDARD LVCMOS33    [get_ports Clk]
set_property PACKAGE_PIN Y11            [get_ports PWMout]
set_property IOSTANDARD LVCMOS33    [get_ports PWMout]
set_property PACKAGE_PIN P16            [get_ports Reset]
set_property IOSTANDARD LVCMOS18    [get_ports Reset]
set_property PACKAGE_PIN F22            [get_ports Dip_SW0]
set_property IOSTANDARD LVCMOS18    [get_ports Dip_SW0]
set_property PACKAGE_PIN G22            [get_ports Dip_SW1]
set_property IOSTANDARD LVCMOS18    [get_ports Dip_SW1]
set_property PACKAGE_PIN H22            [get_ports Dip_SW2]
set_property IOSTANDARD LVCMOS18    [get_ports Dip_SW2]
set_property PACKAGE_PIN F21            [get_ports Dip_SW3]
set_property IOSTANDARD LVCMOS18    [get_ports Dip_SW3]
set_property PACKAGE_PIN H19            [get_ports Dip_SW4]
set_property IOSTANDARD LVCMOS18    [get_ports Dip_SW4]
set_property PACKAGE_PIN H18            [get_ports Dip_SW5]
set_property IOSTANDARD LVCMOS18    [get_ports Dip_SW5]
set_property PACKAGE_PIN H17            [get_ports Dip_SW6]
set_property IOSTANDARD LVCMOS18    [get_ports Dip_SW6]
set_property PACKAGE_PIN M15            [get_ports Dip_SW7]
set_property IOSTANDARD LVCMOS18    [get_ports Dip_SW7]
```

The package pins are – as seen – defined and the iostandard is also defined. Se the documentation of the ZEDBOARD and other Xilinx documents for further information. The names of signals (e.g. get_ports Clk) are the names from your design. And:

NB: The signal names are case sensitive in the constraints file – so Clk and CLK are not the same as in VHDL modules.