# Lab 5 README.pdf

## Lab 5 Partners

Mark Lee, 36183168

Seyed-Ebrahim Mir-Mohammadsadeghi, 84477686

## Break Down

**1. Directory path of SOF and solution:**

rtl/dds_and_nios_lab_time_limited.sof
rtl/Chain1.cdf

**2. Status**

Everything in the lab is working, and according to the provided instructions. There are some caveats, however, as we will discuss below.

Task 7:

- LFSR with correct sequence – Check

Task 8:

- Instantiate the DDS – Check

Task 9:

- Modulate DDS with LFSR – Check
- ASK – Check
- BPSK – Check

Task 10:

- Connect signals to VGA – Check
- ASK – Check
- BPSK – Check
- LFSR – Check
- Use cross clocking logic – Check

Task 11:

- Create PIOs – Check
- Write ISR – Check
- Write FSK to VGA screen – Check
-

Latches:

There are latches in the provided code, and NOT from our code. As per

https://piazza.com/class/kjkioroxku03a6?cid=370
these latches will not be considered.

**3. Annotated simulation screenshots as required by the lab**
The screenshots can be found lower in the document.

**4. Information on how to run the simulations**
For “waveform_gen_TB.sv” and “sincos_lut_TB.sv” an installation of Quatus/ModelSim Altera 16.1 is required, as it is a system verilog testbench that calls a VHDL module.

**5. Any additional information**
Modules were created, and each has their own respective file. These files were called upon by the main “dds_and_nios_lab.v” file.

- “fast_to_slow.sv” for task 10,
- “slow_to_fast.sv” for task 10,
- “Nbit_register_async_reset.sv” for flip flops
- “Nbit_register_async_reset_in.sv” as a special flip flop used in “fast_to_slow.sv”
- “LFSR.sv” the used LFSR module
- “lfsr_ebi.sv” a generic LFSR module that can handle more than 5 bits
- “clock_divider.sv”
- “mux4to1.sv”

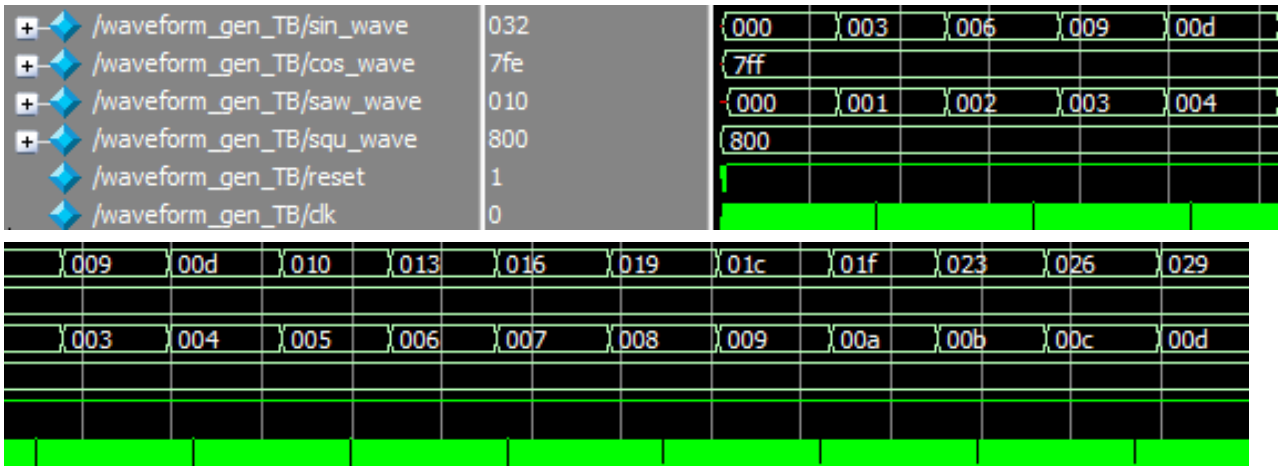
Youtube video of the VGA screen:
https://youtu.be/fl_4Vw-mmPA

There are rigorous comments in each code module that may be helpful for understanding the “what” & “why”s of the code.

Please note that comments containing “//&?&” is just a shorthand that I use to CTRL-F to areas in the professor’s code that requires adjustment.

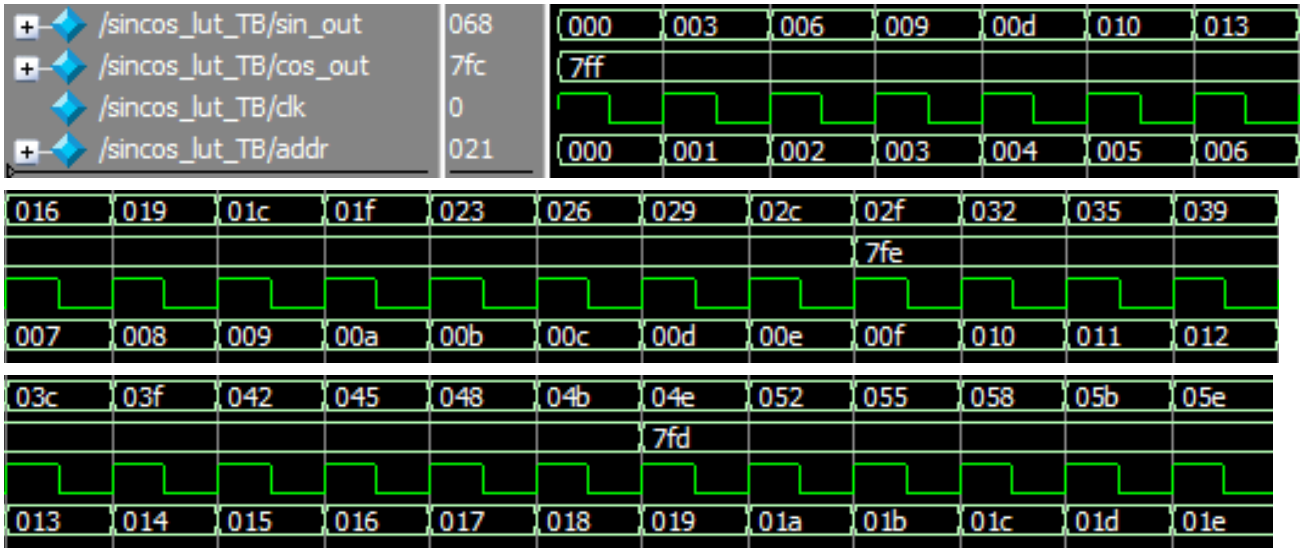
# Annotated Screenshots

**Waveform_gen_TB**
the waveform testbench shows the different waves that are produced with a given phase incrementer.

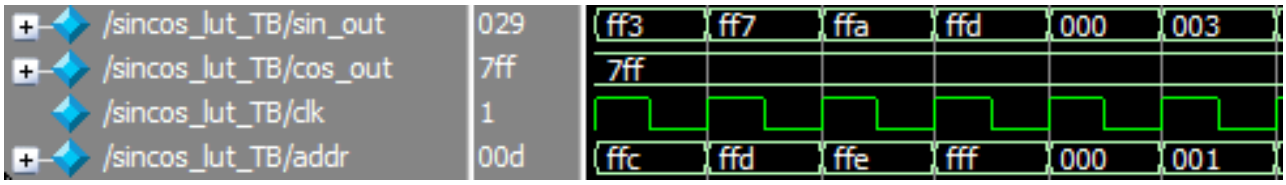



**sincos_lut_TB, sin cos lookup table**

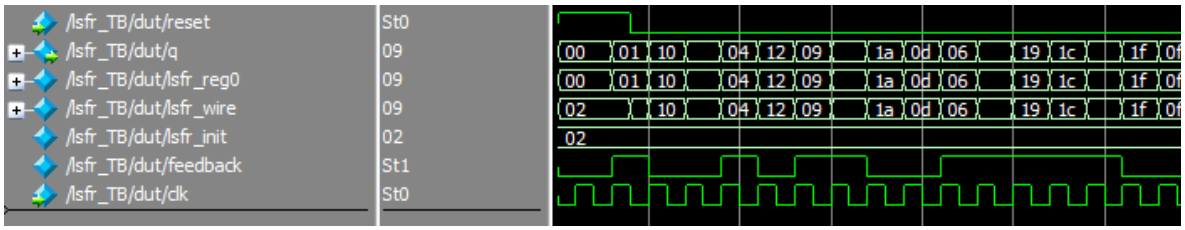
In this testbench I increment addr forever, and observe the values coming through.

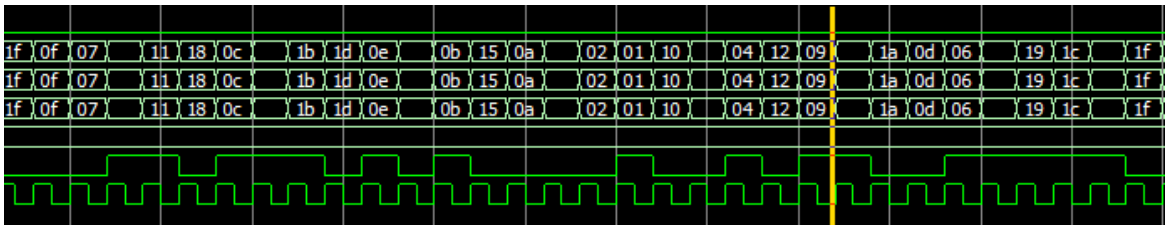



The forever loop in the testbench will overflow and the test loops.



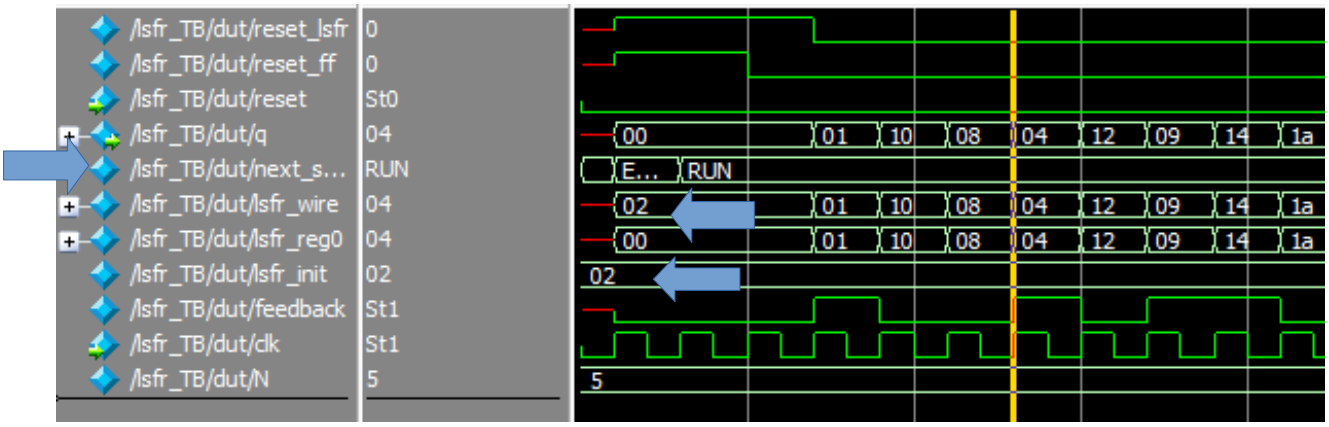

**LFSR – simple**

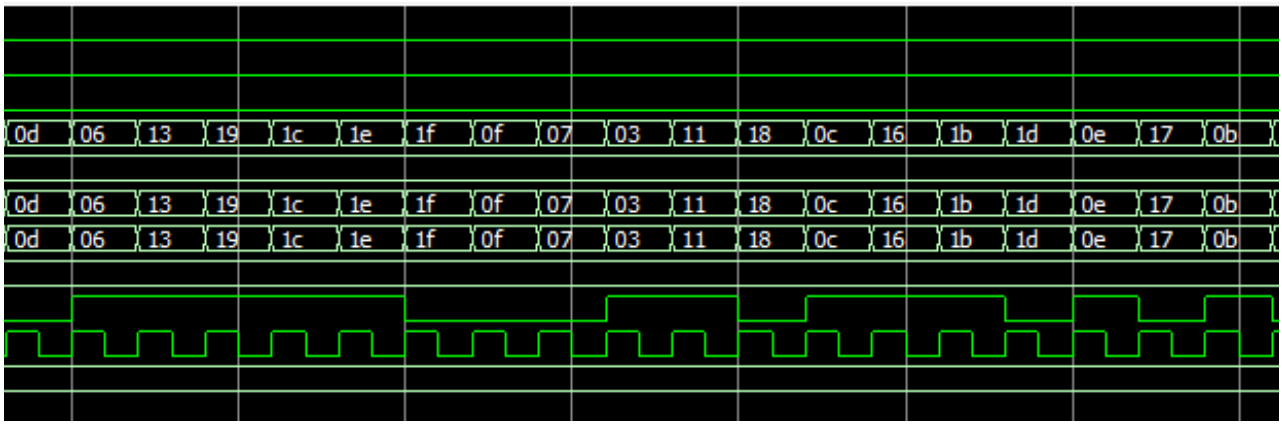


**LFSR – with self-initializing & reset handling FSM**

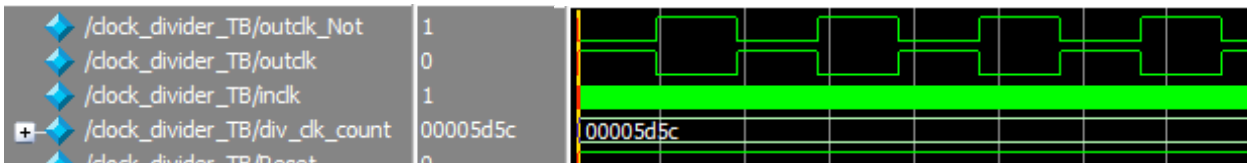
Here we see the LFSR reset its flip flops when first starting, and setting the initial value.



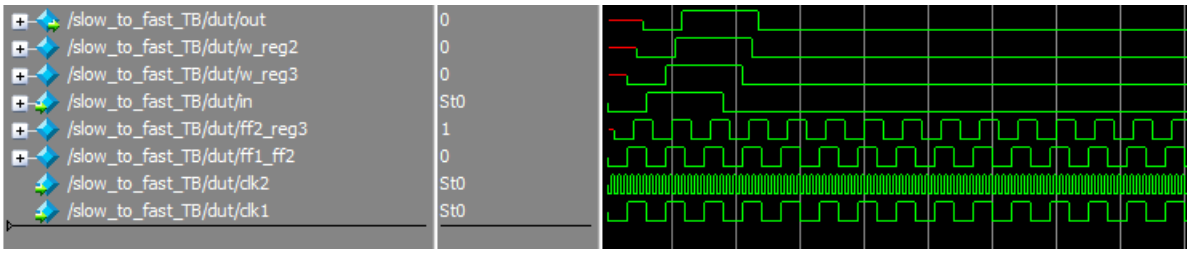



**Clock Divider**

This is fairly self explanatory. The clock divider will wait until the provided number of clock cycles before swapping the outclk.




**Slow to fast, cross clocking logic**




**Fast to slow, cross clocking logic**