CSE 140L Lab 5

Aryan Desai, A16718270;

Pranav Vijay, A16743255;

Academic Integrity

Your work will not be graded unless the signatures of all members of the group are present beneath the honor code.

To uphold academic integrity, students shall:

- Complete and submit academic work that is their own and that is an honest and fair representation of their knowledge and abilities at the time of submission.
- Know and follow the standards of CSE 140L and UCSD.

Please sign (type) your name(s) below the following statement:

I pledge to be fair to my classmates and instructors by completing all of my academic work with integrity. This means that I will respect the standards set by the instructor and institution, be responsible for the consequences of my choices, honestly represent my knowledge and abilities, and be a community member that others can trust to do the right thing even when no one is watching. I will always put learning before grades, and integrity before performance. I pledge to excel with integrity.

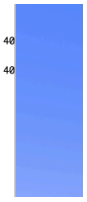
Aryan Desai
Pranav Vijay

Transcript

Both Part 1 and Part 2 Transcripts Are Included Here

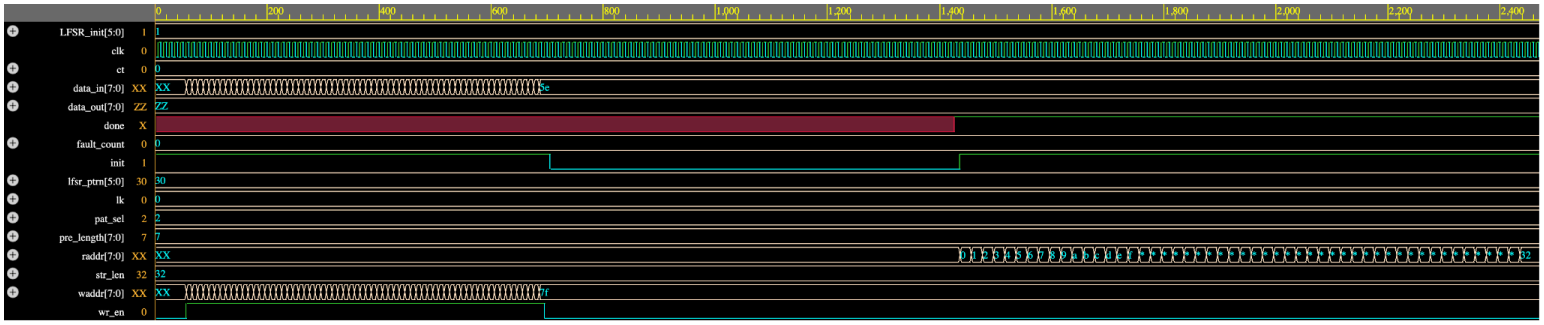
```
# run -all
# preamble length =   7
# tap pattern           2 selected
# LFSR starting pattern = 000001
# original message string length =          50
# embedded leading underscore count =           0
# run encryption of this original message:
# ~~~~~~~~~~`````````~~~~~~~~~~`````````~~~~~~~~~~
#
# LFSR_ptrn = 30, LFSR_init = 01 01
# here is the original message with _ preamble padding
# _______~~~~~~~~~~`````````~~~~~~~~~~`````````~~~~~~~~~~_______
#
# here is the padded and encrypted pattern in ASCII
# ^][WO~\FLXqbEJTiSGo~]ZTHqCgN\yrdIReKVM{WN}[VLySfMZujUkWo_`acgo^
#
# here is the padded pattern in hex
#  5f 5f 5f 5f 5f 5f 5f 40 40 40 40 40 40 40 40 40 40 60 60 60 60 60 60 60 60 60 60 40 40 40 40 40 40 40 40 40 40 60 60 60 60 60 60 60 60 60 60 40 40 40 40 40 40 40 40 40 40 5f 5f 5f 5f 5f 5f 5f
#
# preamble length =   7
# tap pattern           2 selected
# LFSR starting pattern = 000001
# original message string length =          50
# embedded leading underscore count =           0
# run encryption of this original message:
# ~~~~~~~~~~`````````~~~~~~~~~~`````````~~~~~~~~~~
#
# LFSR_ptrn = 30, LFSR_init = 01 01
# here is the original message with _ preamble padding
# _______~~~~~~~~~~`````````~~~~~~~~~~`````````~~~~~~~~~~_______
#
# here is the padded and encrypted pattern in ASCII
# ^][WO~\FLXqbEJTiSGo~]ZTHqCgN\yrdIReKVM{WN}[VLySfMZujUkWo_`acgo^
#
# here is the padded pattern in hex
#  5f 5f 5f 5f 5f 5f 5f 40 40 40 40 40 40 40 40 40 40 60 60 60 60 60 60 60 60 60 60 40 40 40 40 40 40 40 40 40 40 60 60 60 60 60 60 60 60 60 60 40 40 40 40 40 40 40 40 40 40 5f 5f 5f 5f 5f 5f 5f
#
# done at time                1435
# match = 000100  foundit = 2
# dut decryption =
#   40  40  40  40  40  40  40  40  40  60  60  60  60  60  60  60  60  60  60  40  40  40  40  40  40  40  40  40  60  60  60  60  60  60  60  60  60  40  40  40  40  40  40  40  40  40  5f
# run decryption:
# ct =           0
# done at time                1435
# match = 000100  foundit = 2
# dut decryption =
#   40  40  40  40  40  40  40  40  40  60  60  60  60  60  60  60  60  60  60  40  40  40  40  40  40  40  40  40  60  60  60  60  60  60  60  60  60  40  40  40  40  40  40  40  40  40  5f
```

```
# run decryption:
# ct =           0
# 40 40   40 40   40 40   40 40   40 40   40 40   40 40   40 40   60 60   60 60   60 60   60 60   60 60   60 60   60 60   60 60   60 60   40 40   40 40   40 40   40 40   40 40   40 40   40 40
# fault_count =           0
# 40 40   40 40   40 40   40 40   40 40   40 40   40 40   40 40   60 60   60 60   60 60   60 60   60 60   60 60   60 60   60 60   60 60   40 40   40 40   40 40   40 40   40 40   40 40   40 40
# fault_count =           0
# ** Note: $stop    : Lab_5b_tb.sv(185)
#    Time: 2475 ns  Iteration: 0  Instance: /Lab_5b_tb
# Break in Module Lab_5b_tb at Lab_5b_tb.sv line 185
# Stopped at Lab_5b_tb.sv line 185
# exit
# End time: 15:44:55 on Jun 13,2024, Elapsed time: 0:00:01
# Errors: 0, Warnings: 0
```

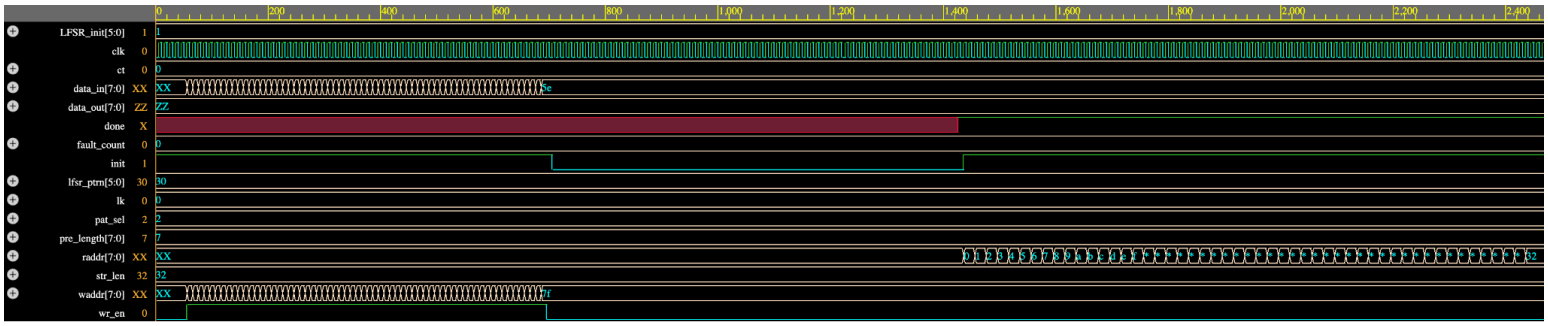


Waveforms

5b_tb



5c_tb:



Summary and Proof of Synthesizability (Both Parts)

For this lab, everything worked as expected and there were no significant walls that were hit that caused major issues. The main problems came with figuring out how the preamble worked for part 2, but we just used the waveforms to debug our issues. For proof of synthesizability, since access to Quartus was limited, EDA Playground was used to produce a netlist for proof of synthesizability. This netlist can be seen below. Was having trouble with running the run.do file. We used Siemens Precision, and this was all that was outputted.

```
//Precision RTL Synthesis 64-bit 2023.1.0.14 (Production Release) Fri Feb 17 02:47:53 PST 2023
//
// This material contains trade secrets or otherwise confidential
// information owned by Siemens Industry Software Inc. or its
// affiliates (collectively, "Siemens"), or its licensors. Access to
// and use of this information is strictly limited as set forth
// in the Customer's applicable agreements with Siemens.
//
// Unpublished work. Copyright 2023 Siemens
//
//Running on Linux runner@cf2d42bf6e6e #169-Ubuntu SMP Tue Jun 6 22:23:09 UTC 2023 5.4.0-152-generic x86_64
//
//
// Verilog description for cell top_level_5b,
//

module top_level_5b ( clk, init, done ) ;

input clk ;
input init ;
output done ;

wire nx26479z1;



OBUF done_obuf (.O (done), .I (nx26479z1)) ;
VCC ps_vcc (.P (nx26479z1)) ;
endmodule
```