City College of New York

Take Home Test # 3

Zi Xuan Li

CSC 211 Fall 2022

Professor Izidor Gertner

November 20th, 2022

Objective:

This lab introduces the idea of storing multiple bits of data by using multiple flip flops. A number N amount of Flip Flops are connected to store N bits of data. This amalgamation is known as a register, which is used to store such information. The objective of this lab report is to build, simulate, and verify the correctness of a Serial-In Parallel-Out shift register (SIPO), a Parallel-In Serial-Out shift register (PISO), and a Linear-Feedback shift register (LFSR). This lab will not include the Serial-In Serial-Out shift register as well as the Parallel-In Parallel-Out shift register. Although the directional movement of the data through a shift register can be either to the left or right, this lab will focus on data shifts to the right (right shifting). Finally, this lab will incorporate a 16-bit variant in addition to the 4-bit variant of the SIPO, PISO, and LFSR.

Functionality, Specifications, and Simulation:

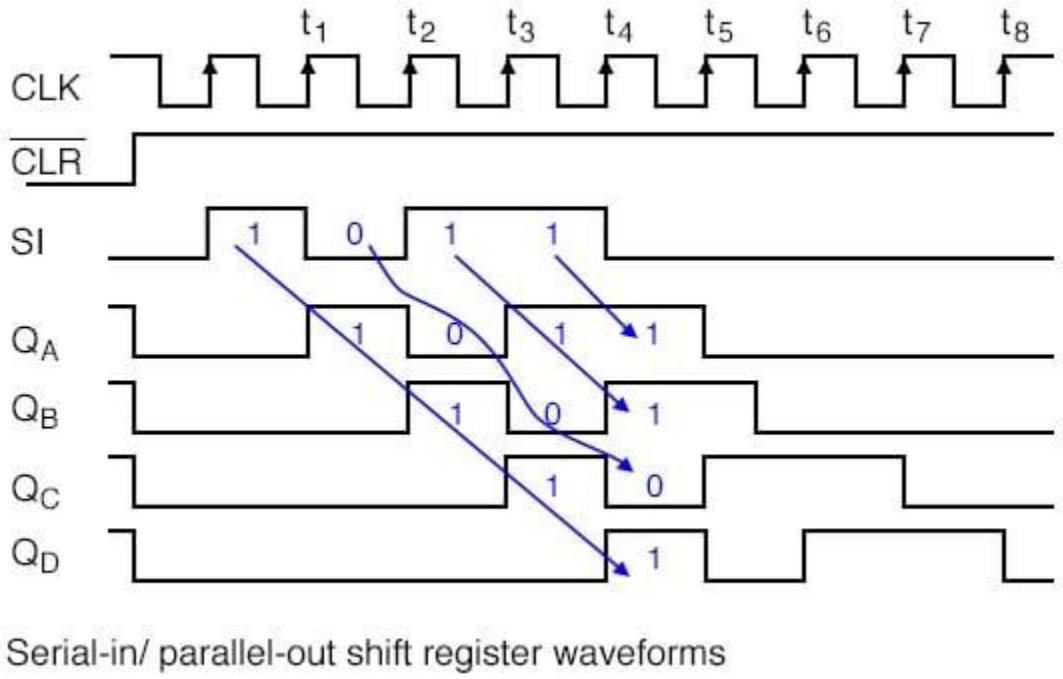
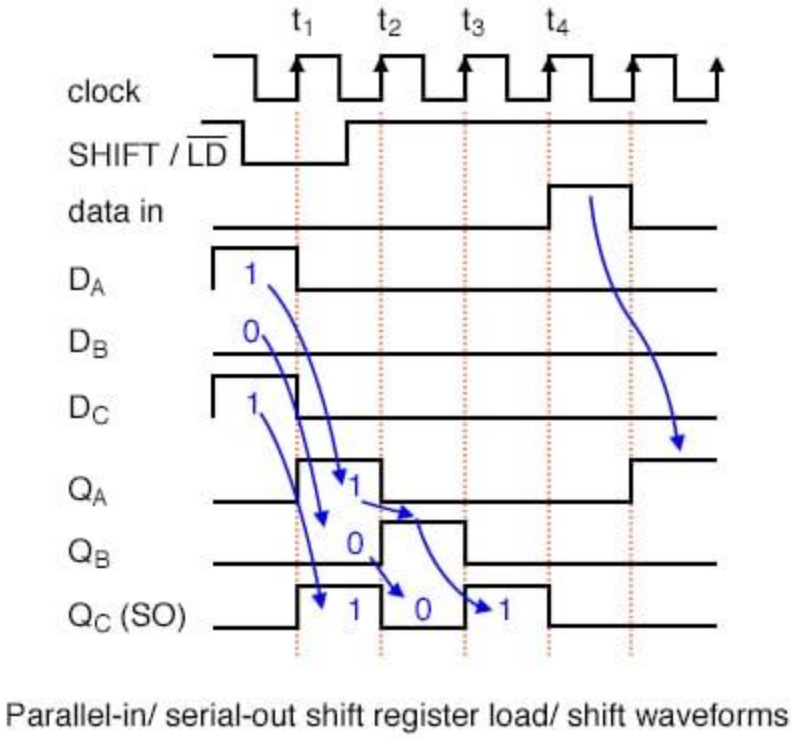


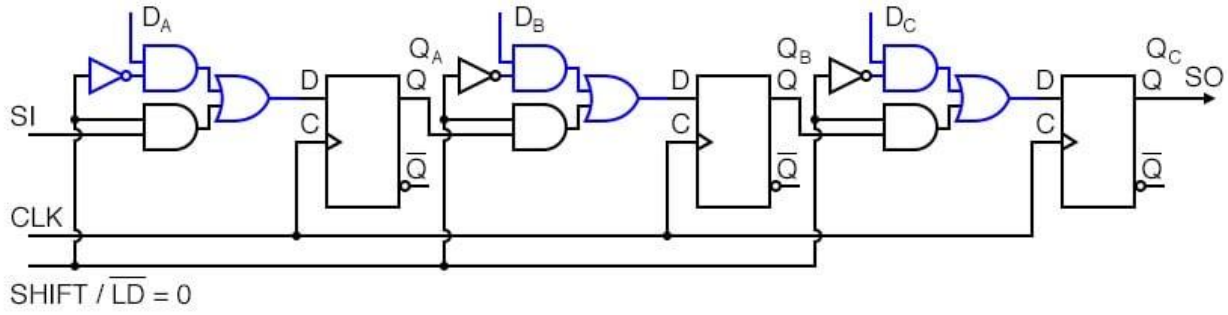
Figure 1: SIPO Shift Register Timing Diagram (Tony R. Kuphaldt 2007)

The timing diagram above is a correct example of the "Simple Shift Register" circuit from the lab instructions. We can prove that this timing diagram is correct because by comparing the given sample sequence to the timing diagram, we can see that SI from the timing diagram matches up with In in the sample sequence, Qa matches up with Q1, Qb matches up with Q2, and so on so forth. The only exception of this is t4 where t4 in the timing diagram is 0 but t4 in the sample sequence is 1, but this result is negligible because the pattern is already recognized by the earlier data shifts. Thus, by comparing our SIPO shift register's waveform outputs to the timing diagram above we can verify the correctness of our own circuit.



Parallel-in/ serial-out shift register load/ shift waveforms

Figure 2: PISO Shift Register Timing Diagram (Tony R. Kuphaldt 2007)



Parallel-in/ serial-out shift register showing parallel load path

Figure 3: PISO Shift Register Circuit Diagram (Tony R. Kuphaldt 2007)

The timing diagram above is a correct example of the "Parallel Shift Register" circuit from the lab instructions. However, we cannot prove this like previously with the SIPO shift register. Instead, we can confirm by using the fact that the timing diagram in figure 2 is a result of the circuit diagram in figure 3, and that the circuit diagram for the Parallel Shift Register in the lab instructions is the same as the circuit diagram in figure 3 to conclude that the timing diagram is correct. Thus, by comparing our PISO shift register's waveform outputs to the timing diagram above we can verify the correctness of our own circuit.

Part 1: (Design and build in Quartus, simulate, and verify correctness of the following serial shift register)

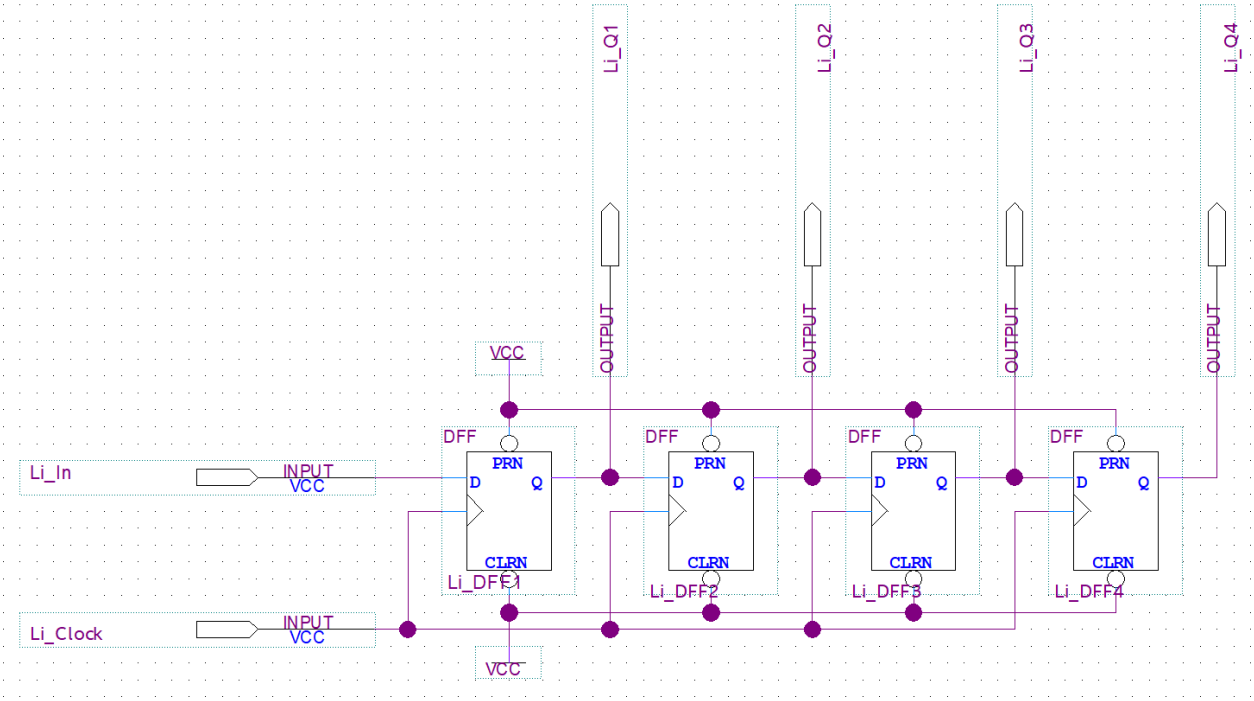


Figure 4: 4-bit Shift Register (SIPO) with VCC (1) connected to all PRN and CLRN as to toggle their functionality off.

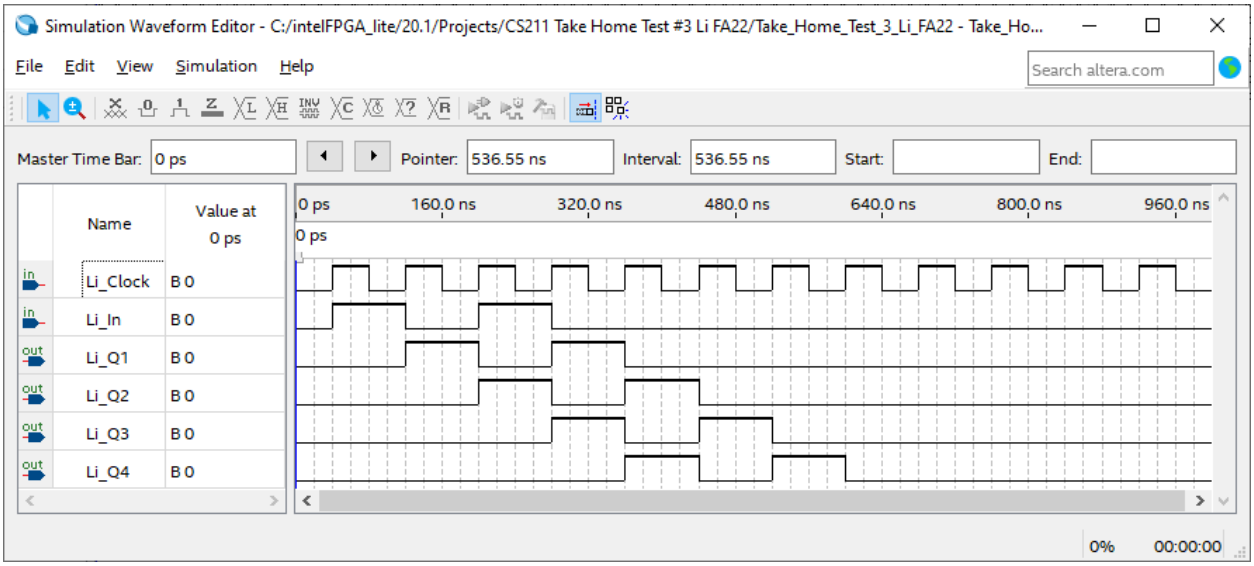


Figure 5: University VWF output for the 4-bit Shift Register in figure 4 with input signals Li_Clock, Li_In and output signals Li_Q1, Li_Q2, Li_Q3, Li_Q4.

```vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.all;

LIBRARY work;

ENTITY Simple4ShiftRegister IS
    PORT
    (
        Li_In :  IN  STD_LOGIC;
        Li_Clock :  IN  STD_LOGIC;
        Li_Q1 :  OUT  STD_LOGIC;
        Li_Q2 :  OUT  STD_LOGIC;
        Li_Q3 :  OUT  STD_LOGIC;
        Li_Q4 :  OUT  STD_LOGIC
    );
END Simple4ShiftRegister;

ARCHITECTURE bdf_type OF Simple4ShiftRegister IS

SIGNAL    SYNTHESIZED_WIRE_8 :  STD_LOGIC;
SIGNAL    SYNTHESIZED_WIRE_9 :  STD_LOGIC;
SIGNAL    DFF_Li_DFF1 :  STD_LOGIC;
SIGNAL    DFF_Li_DFF2 :  STD_LOGIC;
SIGNAL    DFF_Li_DFF3 :  STD_LOGIC;

BEGIN
Li_Q1 <= DFF_Li_DFF1;
Li_Q2 <= DFF_Li_DFF2;
Li_Q3 <= DFF_Li_DFF3;
SYNTHESIZED_WIRE_8 <= '1';
SYNTHESIZED_WIRE_9 <= '1';

PROCESS(Li_Clock,SYNTHESIZED_WIRE_8,SYNTHESIZED_WIRE_9)
BEGIN
IF (SYNTHESIZED_WIRE_8 = '0') THEN
    DFF_Li_DFF1 <= '0';
ELSIF (SYNTHESIZED_WIRE_9 = '0') THEN
    DFF_Li_DFF1 <= '1';
ELSIF (RISING_EDGE(Li_Clock)) THEN
    DFF_Li_DFF1 <= Li_In;
END IF;
END PROCESS;

PROCESS(Li_Clock,SYNTHESIZED_WIRE_8,SYNTHESIZED_WIRE_9)
BEGIN
IF (SYNTHESIZED_WIRE_8 = '0') THEN
    DFF_Li_DFF2 <= '0';
ELSIF (SYNTHESIZED_WIRE_9 = '0') THEN
    DFF_Li_DFF2 <= '1';
ELSIF (RISING_EDGE(Li_Clock)) THEN
    DFF_Li_DFF2 <= DFF_Li_DFF1;
END IF;
END PROCESS;

PROCESS(Li_Clock,SYNTHESIZED_WIRE_8,SYNTHESIZED_WIRE_9)
BEGIN
IF (SYNTHESIZED_WIRE_8 = '0') THEN
    DFF_Li_DFF3 <= '0';
ELSIF (SYNTHESIZED_WIRE_9 = '0') THEN
    DFF_Li_DFF3 <= '1';
ELSIF (RISING_EDGE(Li_Clock)) THEN
    DFF_Li_DFF3 <= DFF_Li_DFF2;
END IF;
END PROCESS;

PROCESS(Li_Clock,SYNTHESIZED_WIRE_8,SYNTHESIZED_WIRE_9)
BEGIN
IF (SYNTHESIZED_WIRE_8 = '0') THEN
    Li_Q4 <= '0';
ELSIF (SYNTHESIZED_WIRE_9 = '0') THEN
    Li_Q4 <= '1';
ELSIF (RISING_EDGE(Li_Clock)) THEN
    Li_Q4 <= DFF_Li_DFF3;
END IF;
END PROCESS;

END bdf_type;
```

Figure 6: VHDL code for 4-bit Shift Register (SIPO).

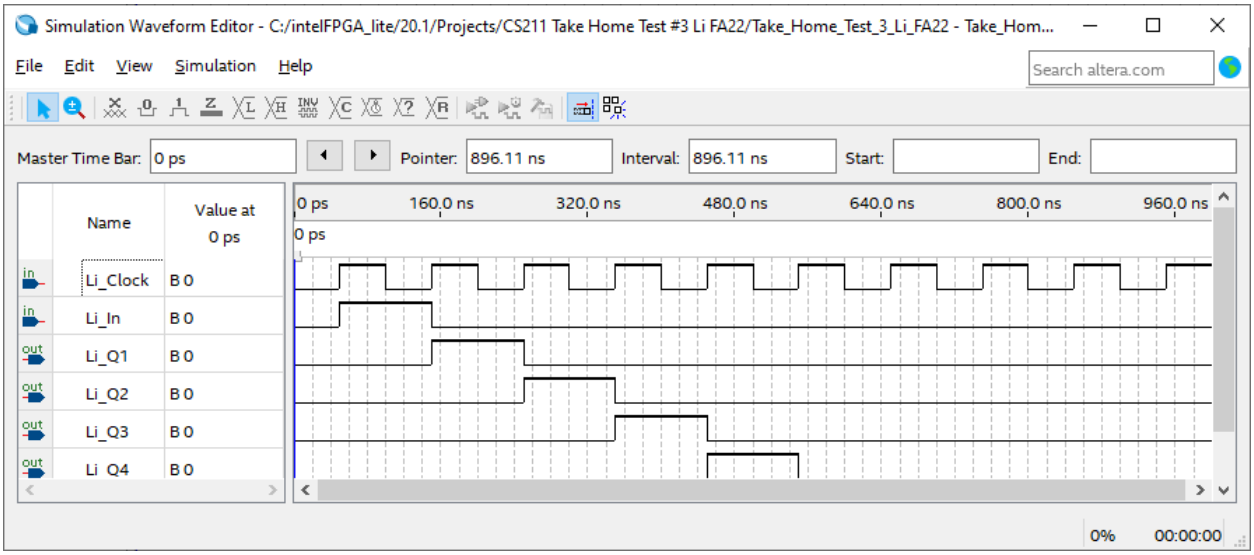


Figure 7: University VWF output for the 4-bit Shift Register built using VHDL code in figure 6 with input signals Li_Clock, Li_In and output signals Li_Q1, Li_Q2, Li_Q3, Li_Q4.

The circuit diagram for the 4-bit shift register built using D-FFs and VHDL code is both correct because we can compare the pattern of their VWF outputs to the pattern of the timing diagram in Figure 1. Inspecting closely, we can see that the inputs of Li_In are being shifted right at every clock pulse which matches with the pattern of timing diagram of the SIPO shift register.

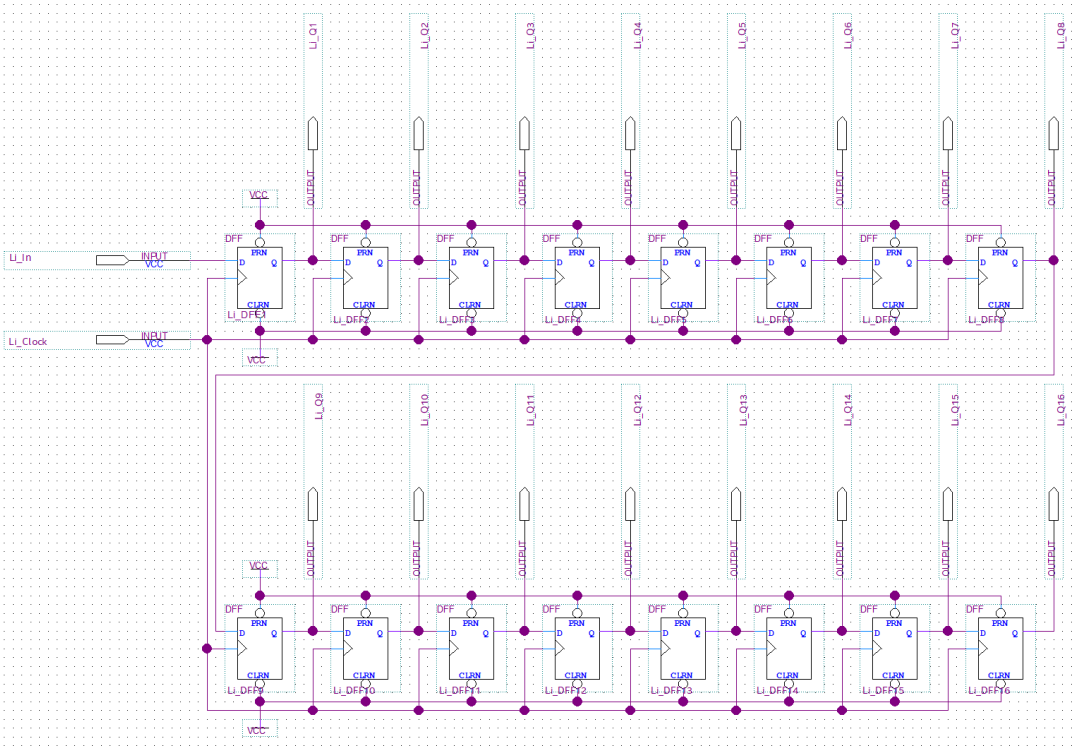


Figure 8: 16-bit Shift Register (SIPO) with VCC (1) connected to all PRN and CLRN as to toggle their functionality off.

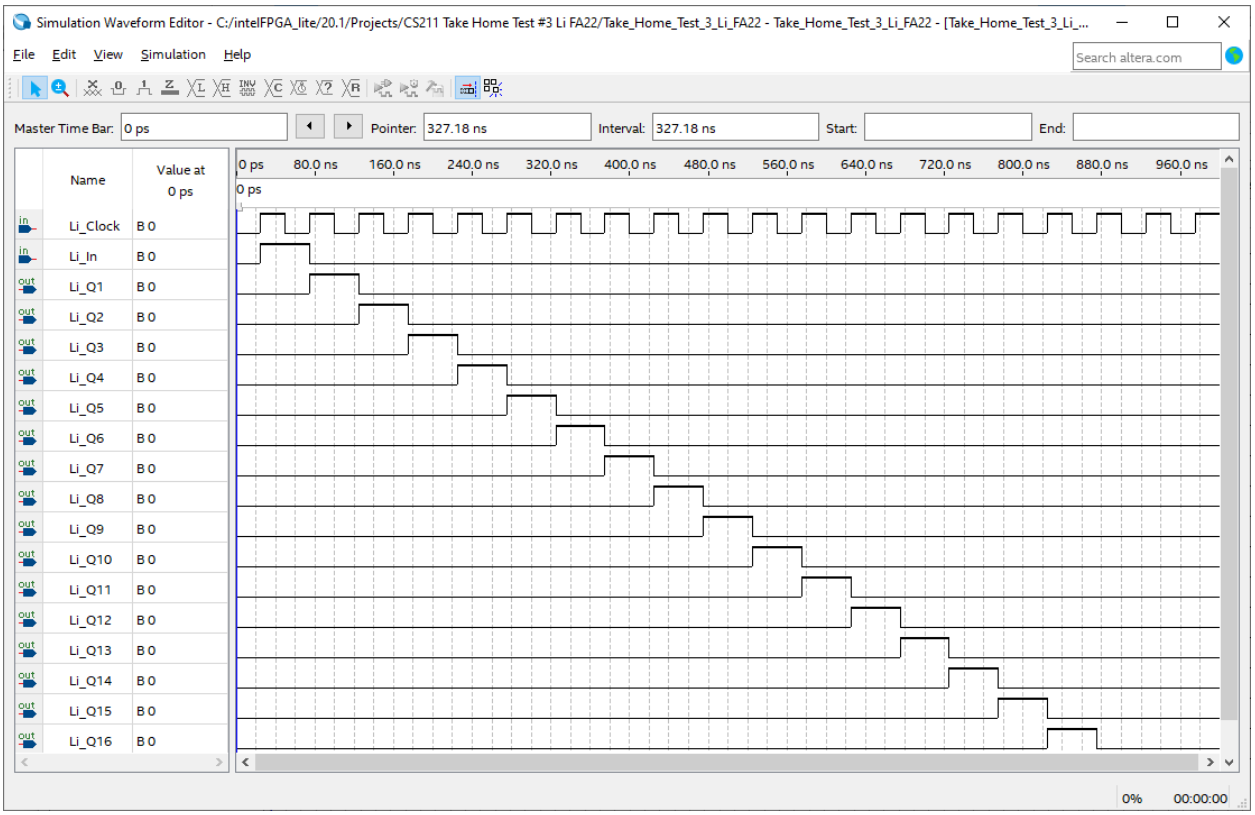


Figure 9: University VWF output for the 16-bit Shift Register in figure 8 with input signals Li_Clock, Li_In and output signals Li_Q1, Li_Q2, Li_Q3, Li_Q4, Li_Q5, Li_Q6, Li_Q7, Li_Q8, Li_Q9, Li_Q10, Li_Q11, Li_Q12, Li_Q13, Li_Q14, Li_Q15, Li_Q16.

Part 2: (Design and build in Quartus, simulate, and verify correctness of the following parallel shift register)

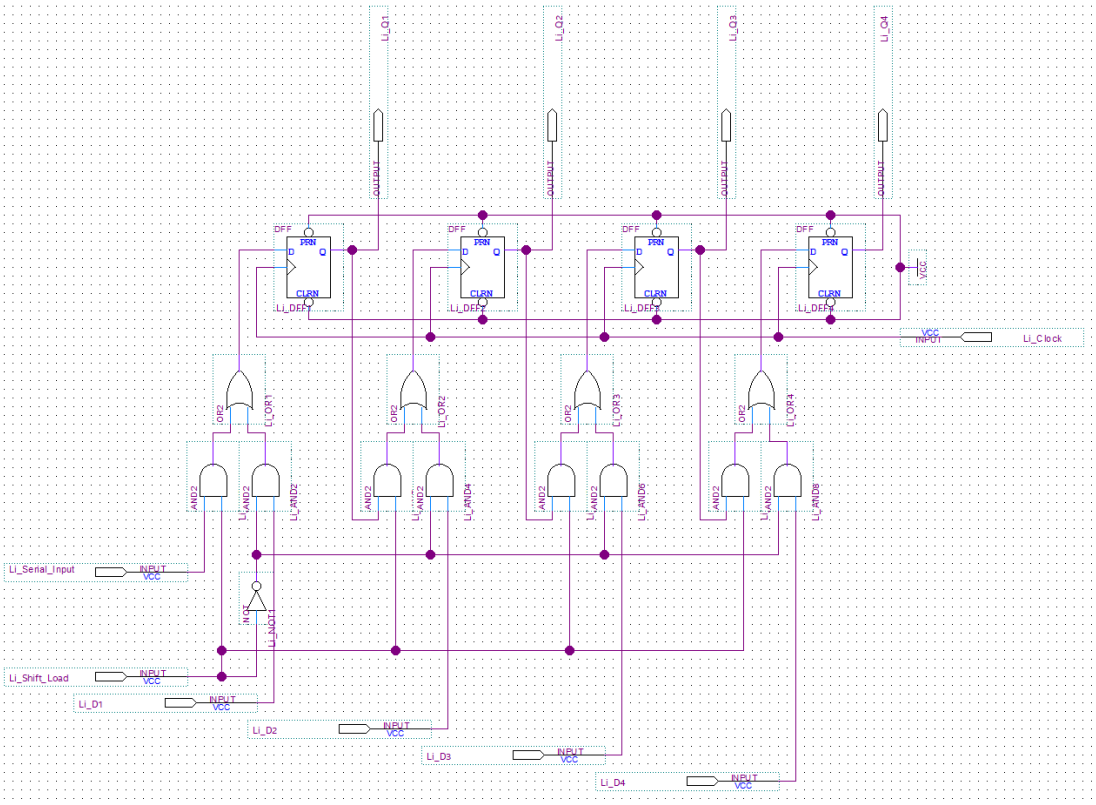


Figure 10: 4-bit Shift Register (PISO) with VCC (1) connected to all PRN and CLRN as to toggle their functionality off.

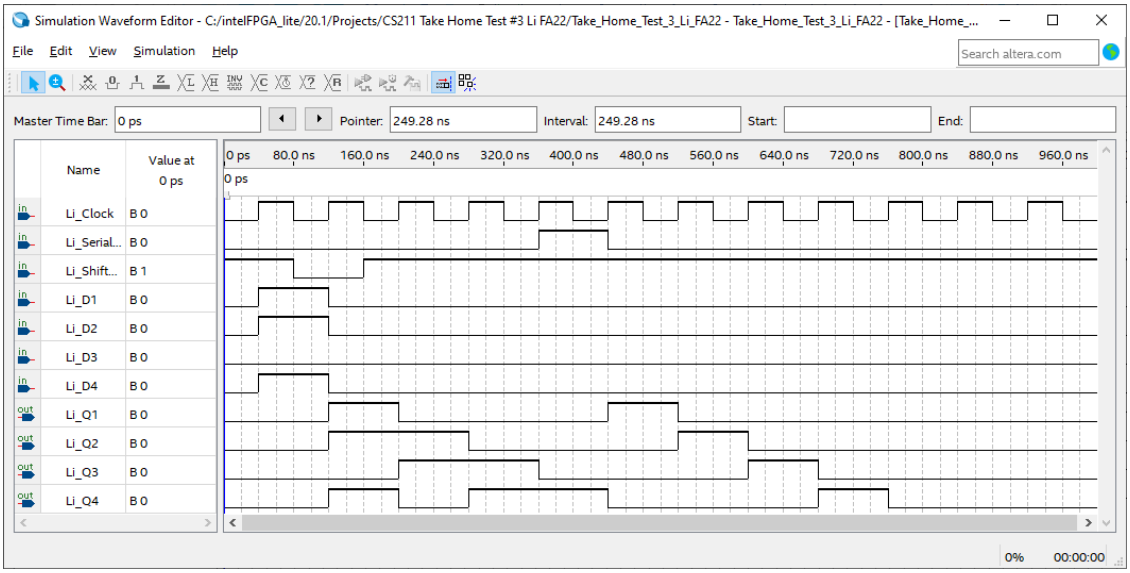


Figure 11: University VWF output for the 4-bit Shift Register in figure 10 with input signals Li_Clock, Li_Serial_Input, Li_Shift_Load, Li_D1, Li_D2, Li_D3, Li_D4and output signals Li_Q1, Li_Q2, Li_Q3, Li_Q4.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

LIBRARY work;

ENTITY Parallel4ShiftRegister IS
   PORT
   (
      Li_Clock :  IN  STD_LOGIC;
      Li_Serial_Input :  IN  STD_LOGIC;
      Li_Shift_Load :  IN  STD_LOGIC;
      Li_D2 :  IN  STD_LOGIC;
      Li_D1 :  IN  STD_LOGIC;
      Li_D3 :  IN  STD_LOGIC;
      Li_D4 :  IN  STD_LOGIC;
      Li_Q1 :  OUT  STD_LOGIC;
      Li_Q2 :  OUT  STD_LOGIC;
      Li_Q3 :  OUT  STD_LOGIC;
      Li_Q4 :  OUT  STD_LOGIC
   );
END Parallel4ShiftRegister;

ARCHITECTURE bdf_type OF Parallel4ShiftRegister IS

SIGNAL   SYNTHESIZED_WIRE_24 :  STD_LOGIC;
SIGNAL   DFF_Li_DFF1 :  STD_LOGIC;
SIGNAL   DFF_Li_DFF2 :  STD_LOGIC;
SIGNAL   DFF_Li_DFF3 :  STD_LOGIC;
SIGNAL   SYNTHESIZED_WIRE_25 :  STD_LOGIC;
SIGNAL   SYNTHESIZED_WIRE_5 :  STD_LOGIC;
SIGNAL   SYNTHESIZED_WIRE_8 :  STD_LOGIC;
SIGNAL   SYNTHESIZED_WIRE_11 :  STD_LOGIC;
SIGNAL   SYNTHESIZED_WIRE_14 :  STD_LOGIC;
SIGNAL   SYNTHESIZED_WIRE_16 :  STD_LOGIC;
SIGNAL   SYNTHESIZED_WIRE_17 :  STD_LOGIC;
SIGNAL   SYNTHESIZED_WIRE_18 :  STD_LOGIC;
SIGNAL   SYNTHESIZED_WIRE_19 :  STD_LOGIC;
SIGNAL   SYNTHESIZED_WIRE_20 :  STD_LOGIC;
SIGNAL   SYNTHESIZED_WIRE_21 :  STD_LOGIC;
SIGNAL   SYNTHESIZED_WIRE_22 :  STD_LOGIC;
SIGNAL   SYNTHESIZED_WIRE_23 :  STD_LOGIC;

BEGIN
Li_Q1 <= DFF_Li_DFF1;
Li_Q2 <= DFF_Li_DFF2;
Li_Q3 <= DFF_Li_DFF3;
SYNTHESIZED_WIRE_25 <= '1';

SYNTHESIZED_WIRE_17 <= Li_Serial_Input AND Li_Shift_Load;


SYNTHESIZED_WIRE_16 <= SYNTHESIZED_WIRE_24 AND Li_D1;


SYNTHESIZED_WIRE_19 <= DFF_Li_DFF1 AND Li_Shift_Load;


SYNTHESIZED_WIRE_18 <= SYNTHESIZED_WIRE_24 AND Li_D2;


SYNTHESIZED_WIRE_21 <= DFF_Li_DFF2 AND Li_Shift_Load;


SYNTHESIZED_WIRE_20 <= SYNTHESIZED_WIRE_24 AND Li_D3;


SYNTHESIZED_WIRE_23 <= DFF_Li_DFF3 AND Li_Shift_Load;


SYNTHESIZED_WIRE_22 <= SYNTHESIZED_WIRE_24 AND Li_D4;


PROCESS(Li_Clock,SYNTHESIZED_WIRE_25,SYNTHESIZED_WIRE_25)
BEGIN
IF (SYNTHESIZED_WIRE_25 = '0') THEN
   DFF_Li_DFF1 <= '0';
ELSIF (SYNTHESIZED_WIRE_25 = '0') THEN
```

Figure 12: VHDL code for 4-bit Shift Register (PISO).

```
78      DFF_Li_DFF1 <= '1';
79    ELSIF (RISING_EDGE(Li_Clock)) THEN
80      DFF_Li_DFF1 <= SYNTHESIZED_WIRE_5;
81    END IF;
82    END PROCESS;
83
84    PROCESS(Li_Clock,SYNTHESIZED_WIRE_25,SYNTHESIZED_WIRE_25)
85    BEGIN
86    IF (SYNTHESIZED_WIRE_25 = '0') THEN
87      DFF_Li_DFF2 <= '0';
88    ELSIF (SYNTHESIZED_WIRE_25 = '0') THEN
89      DFF_Li_DFF2 <= '1';
90    ELSIF (RISING_EDGE(Li_Clock)) THEN
91      DFF_Li_DFF2 <= SYNTHESIZED_WIRE_8;
92    END IF;
93    END PROCESS;
94
95    PROCESS(Li_Clock,SYNTHESIZED_WIRE_25,SYNTHESIZED_WIRE_25)
96    BEGIN
97    IF (SYNTHESIZED_WIRE_25 = '0') THEN
98      DFF_Li_DFF3 <= '0';
99    ELSIF (SYNTHESIZED_WIRE_25 = '0') THEN
100     DFF_Li_DFF3 <= '1';
101   ELSIF (RISING_EDGE(Li_Clock)) THEN
102     DFF_Li_DFF3 <= SYNTHESIZED_WIRE_11;
103   END IF;
104   END PROCESS;
105
106   PROCESS(Li_Clock,SYNTHESIZED_WIRE_25,SYNTHESIZED_WIRE_25)
107   BEGIN
108   IF (SYNTHESIZED_WIRE_25 = '0') THEN
109     Li_Q4 <= '0';
110   ELSIF (SYNTHESIZED_WIRE_25 = '0') THEN
111     Li_Q4 <= '1';
112   ELSIF (RISING_EDGE(Li_Clock)) THEN
113     Li_Q4 <= SYNTHESIZED_WIRE_14;
114   END IF;
115   END PROCESS;
116
117   SYNTHESIZED_WIRE_24 <= NOT(Li_Shift_Load);
118
119   SYNTHESIZED_WIRE_5 <= SYNTHESIZED_WIRE_16 OR SYNTHESIZED_WIRE_17;
120
121   SYNTHESIZED_WIRE_8 <= SYNTHESIZED_WIRE_18 OR SYNTHESIZED_WIRE_19;
122
123   SYNTHESIZED_WIRE_11 <= SYNTHESIZED_WIRE_20 OR SYNTHESIZED_WIRE_21;
124
125   SYNTHESIZED_WIRE_14 <= SYNTHESIZED_WIRE_22 OR SYNTHESIZED_WIRE_23;
126
127   END bdf_type;
```

Figure 13: VHDL code for 4-bit Shift Register (PISO) continued.

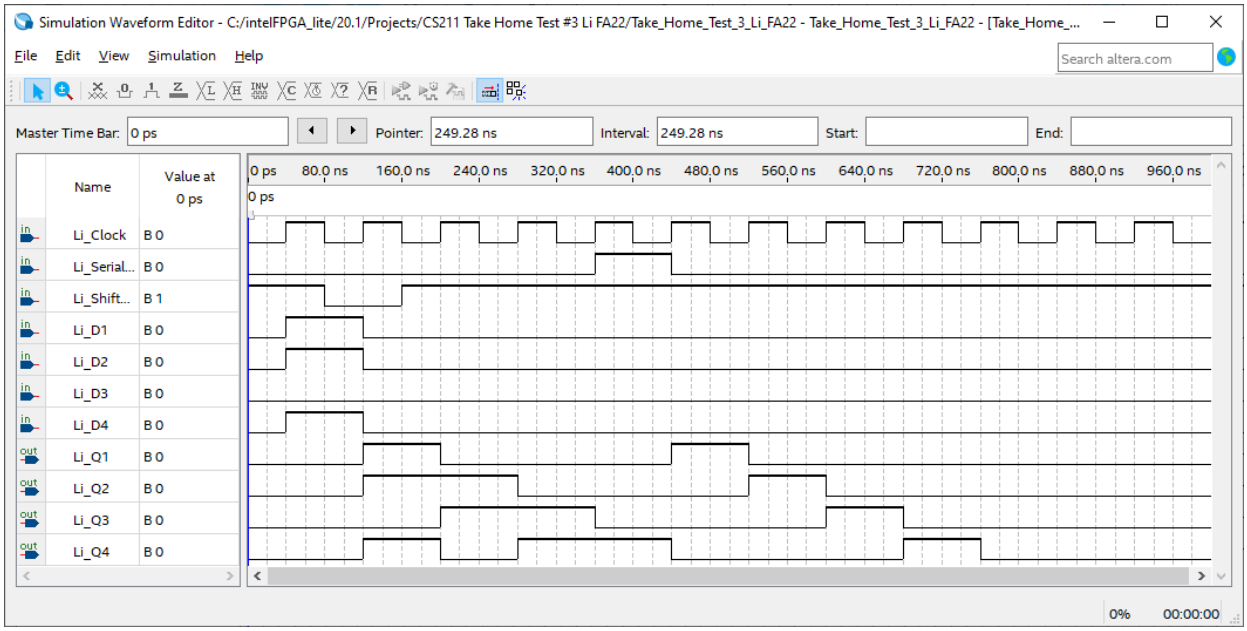


Figure 14: University VWF output for the 4-bit Shift Register built using VHDL code in figure 12 & 13 with input signals Li_Clock, Li_Serial_Input, Li_Shift_Load, Li_D1, Li_D2, Li_D3, Li_D4 and output signals Li_Q1, Li_Q2, Li_Q3, Li_Q4.

The circuit diagram for the 4-bit parallel shift register built using D-FFs and VHDL code is both correct because we can compare the pattern of their VWF outputs to the pattern of the timing diagram in Figure 2. Inspecting closely, we can see that the inputs of Li_In are being shifted right at every clock pulse which matches with the pattern of timing diagram of the SIPO shift register. We can even see that when Li_Serial_Input is 1, that the waveform Q1 becomes 1 during the next clock pulse as well as the cascading pattern afterwards.

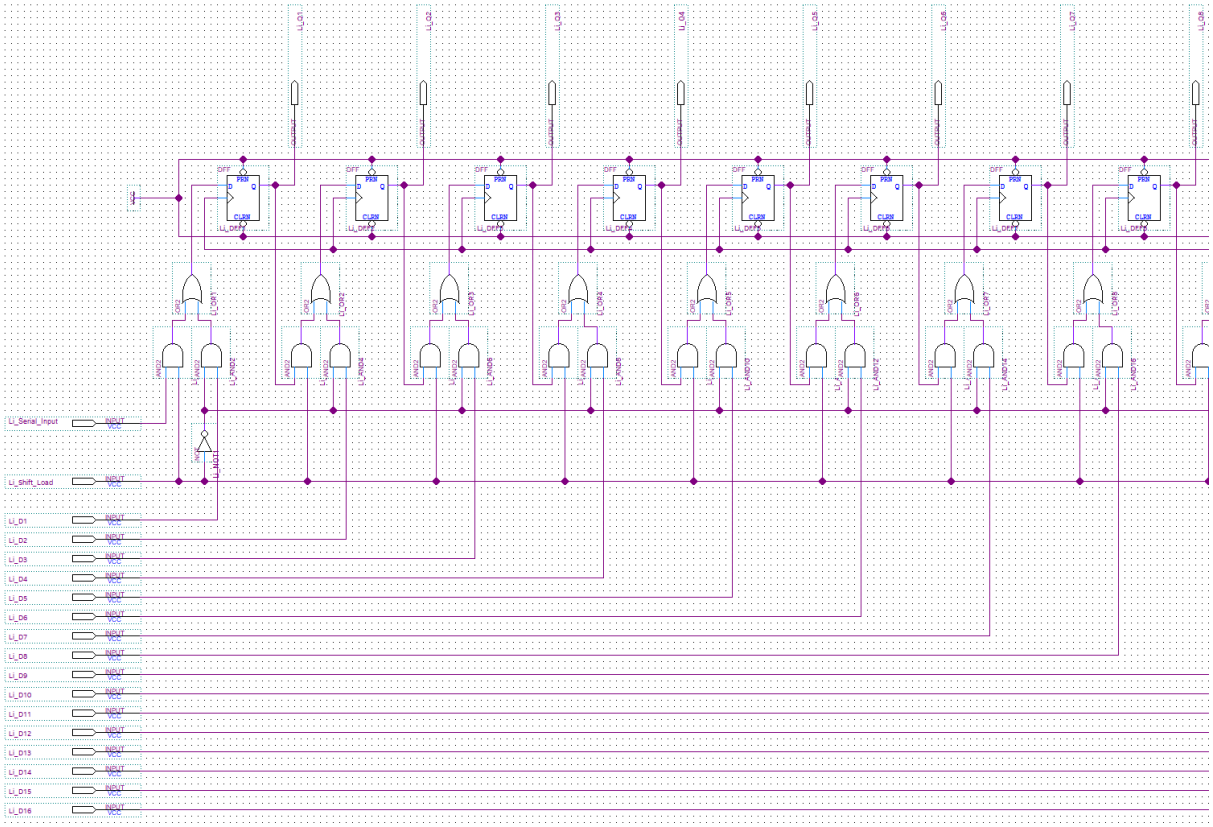


Figure 15: 16-bit Shift Register (PISO) with VCC (1) connected to all PRN and CLRN as to toggle their functionality off.

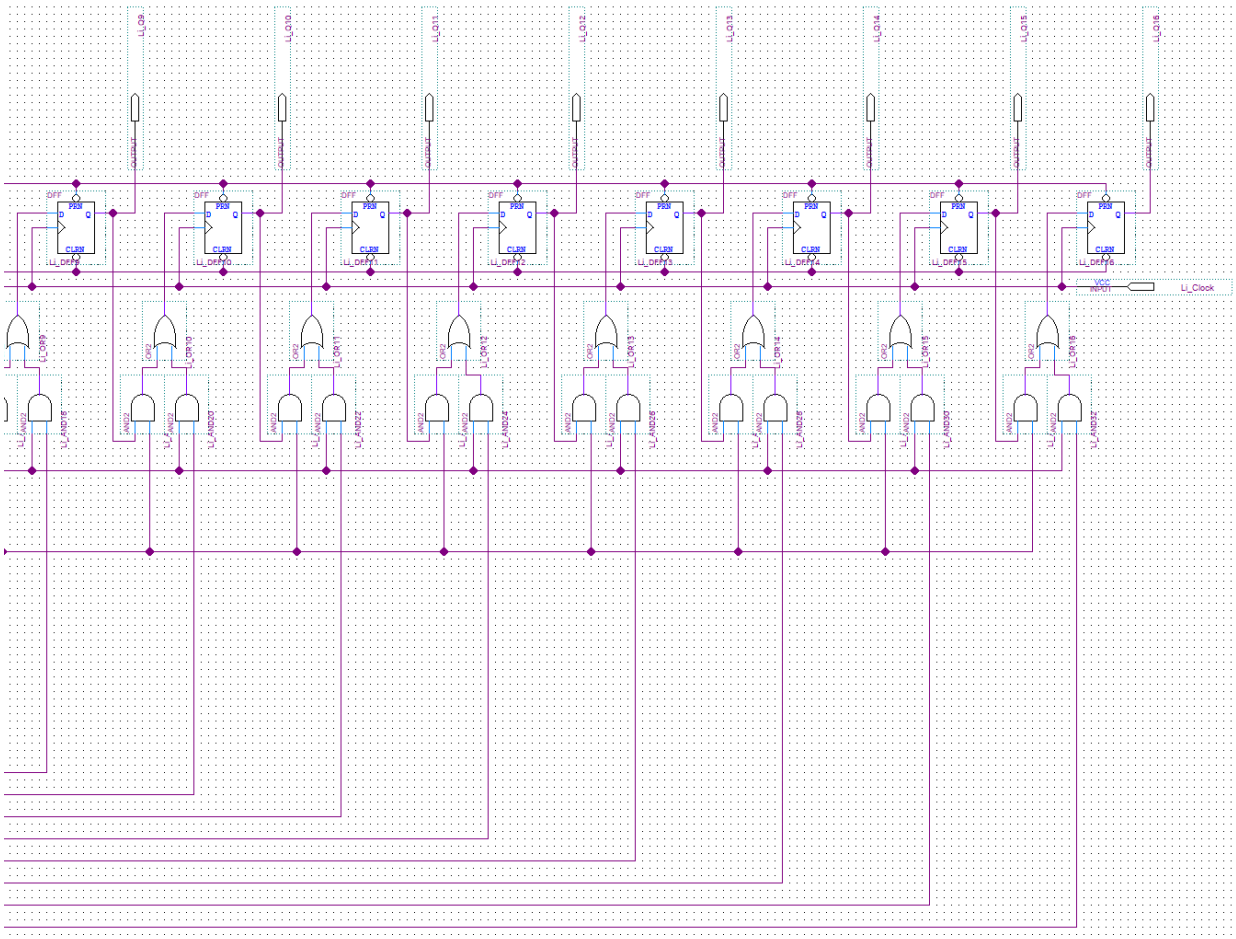


Figure 16: 16-bit Shift Register (PISO) with VCC (1) connected to all PRN and CLRN as to toggle their functionality off continued.

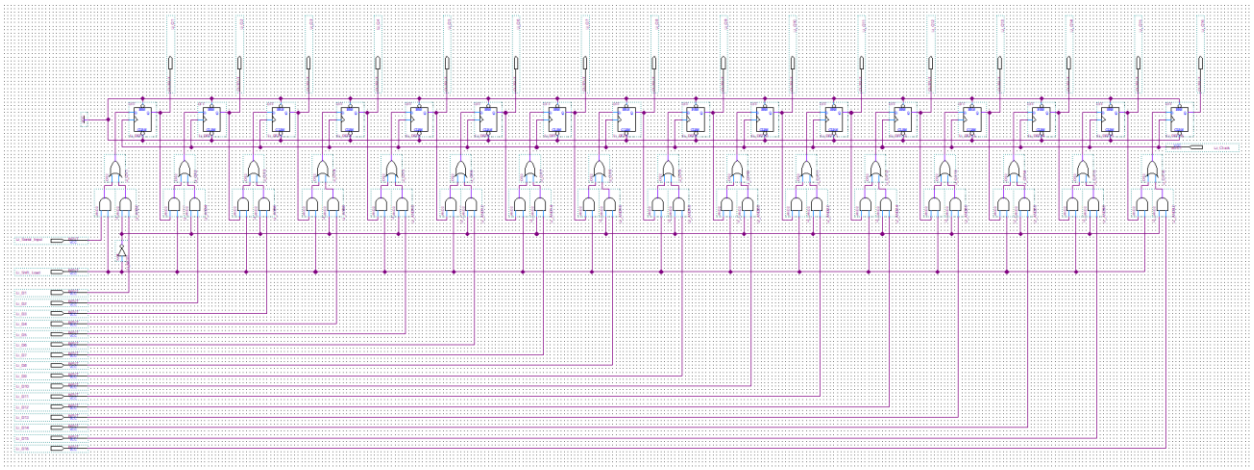


Figure 17: Full Picture of 16-bit Shift Register (PISO) with VCC (1) connected to all PRN and CLRN as to toggle their functionality off.

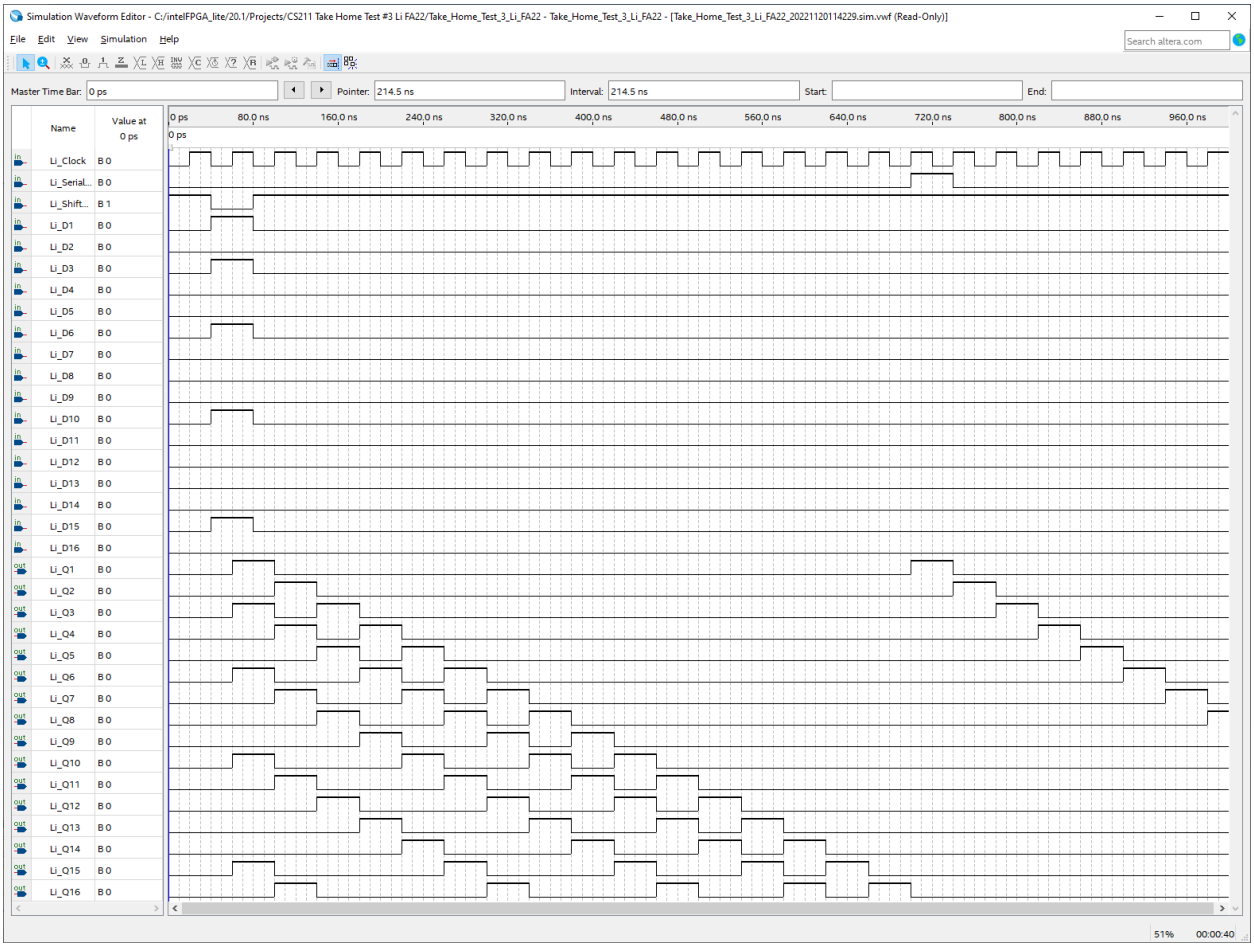


Figure 18: University VWF output for the 16-bit Shift Register in figure 17 with input signals Li_Clock, Li_Serial_Input, Li_Shift_Load, Li_D1, Li_D2, Li_D3, Li_D4, Li_D5, Li_D6, Li_D7, Li_D8, Li_D9, Li_D10, Li_D11, Li_D12, Li_D13, Li_D14, Li_D15, Li_D16 and output Li_Q1, Li_Q2, Li_Q3, Li_Q4, Li_Q5, Li_Q6, Li_Q7, Li_Q8, Li_Q9, Li_Q10, Li_Q11, Li_Q12, Li_Q13, Li_Q14, Li_Q15, Li_Q16.

Part 3: (Design and build in Quartus, simulate, and verify correctness of the linear feedback shift register)

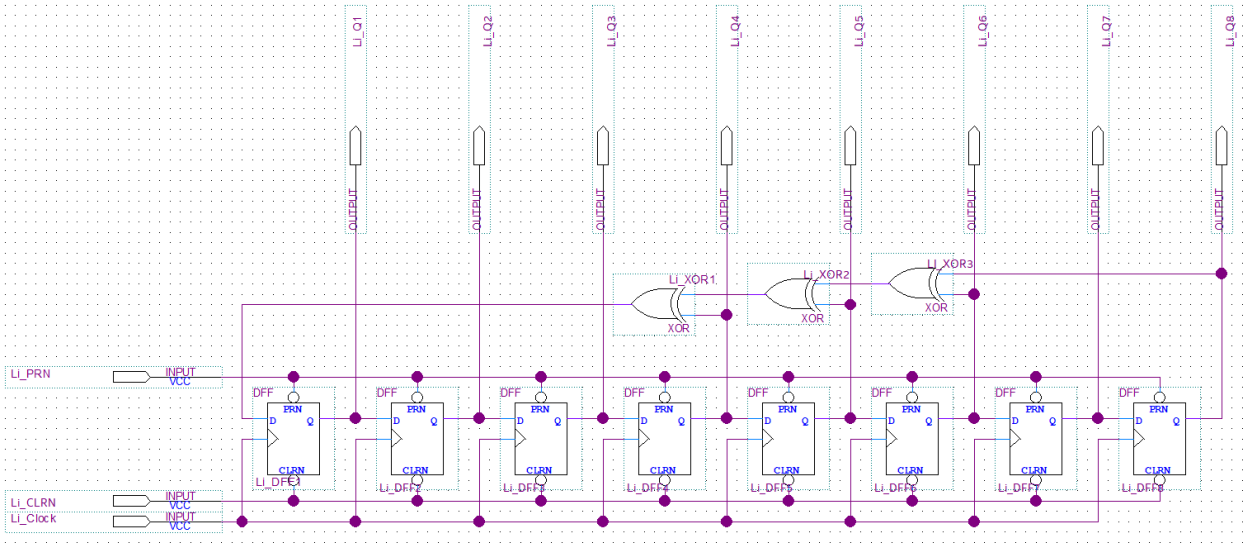


Figure 19: 8-bit Shift Register (LFSR)

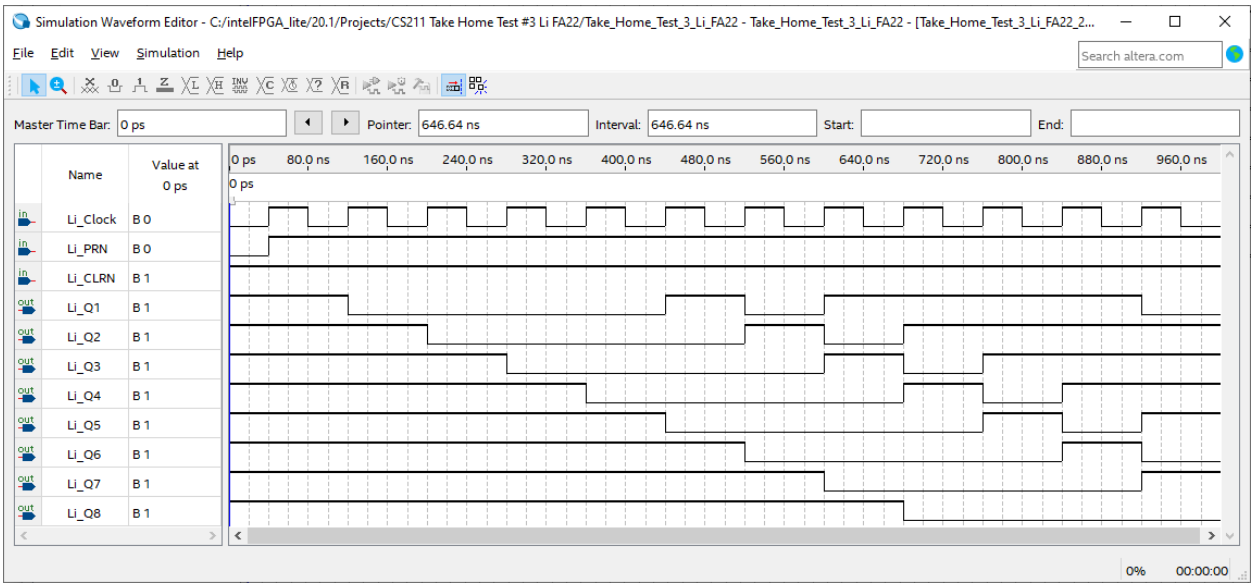


Figure 20: University VWF output for 8-bit Shift Register in figure 19 with input signals Li_Clock, Li_PRN, Li_CLRN and output signals Li_Q1, Li_Q2, Li_Q3, Li_Q4, Li_Q5, Li_Q6, Li_Q7, Li_Q8.

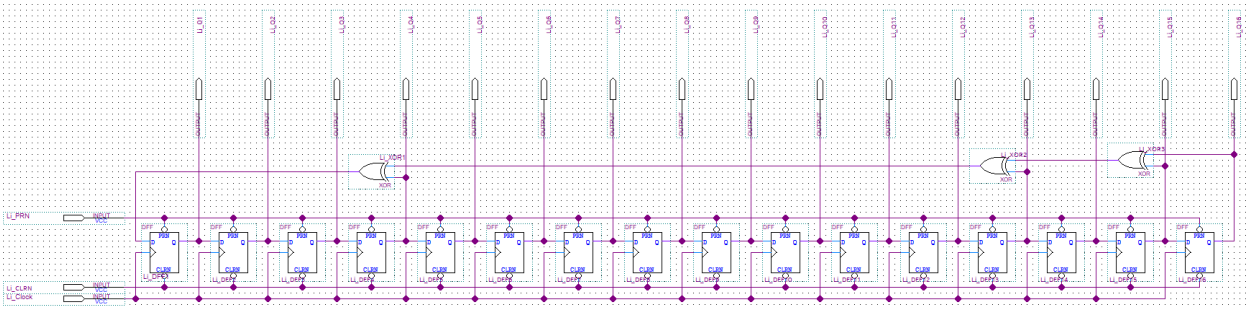


Figure 21: 16-bit Shift Register (LFSR)

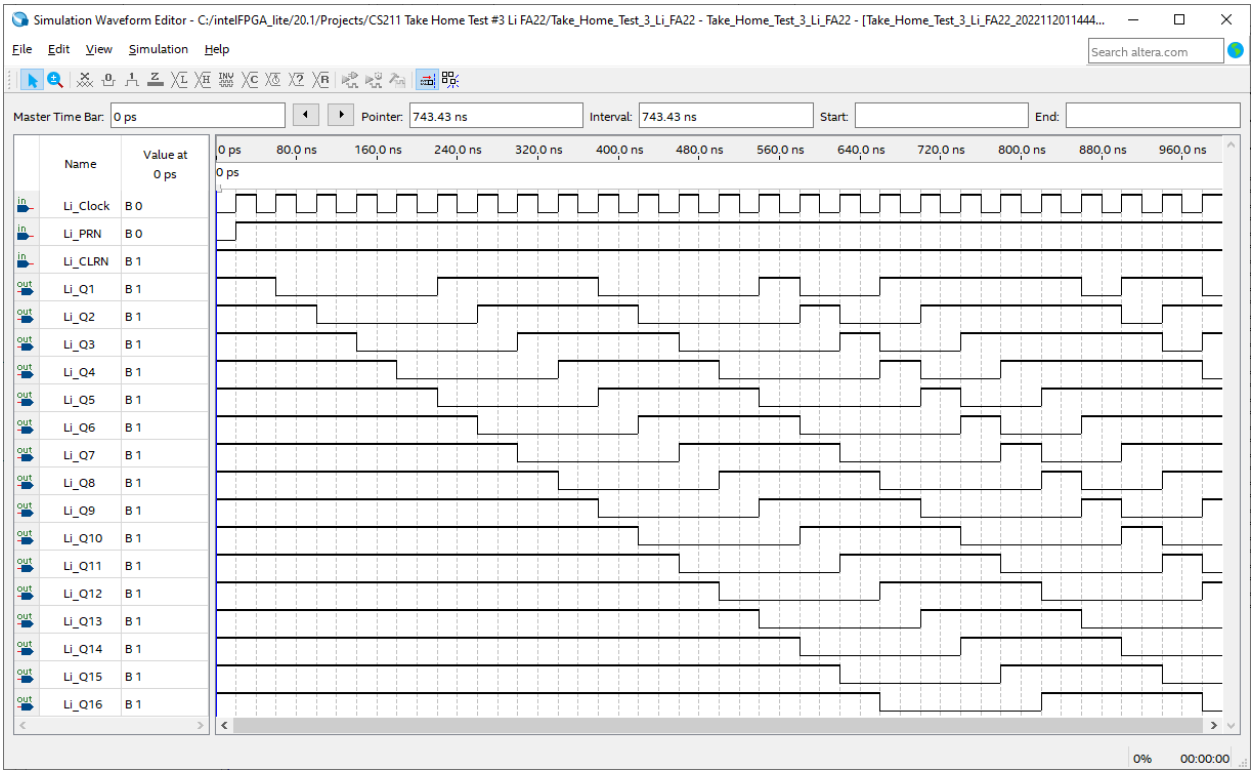


Figure 22: University VWF output for 16-bit Shift Register in figure 21 with input signals Li_Clock, Li_PRN, Li_CLRN and output signals Li_Q1, Li_Q2, Li_Q3, Li_Q4, Li_Q5, Li_Q6, Li_Q7, Li_Q8, Li_Q9, Li_Q10, Li_Q11, Li_Q12, Li_Q13, Li_Q14, Li_Q15, Li_Q16.

These circuits are also correct because we can compare the literal VWF output to the given simulated waveform from the lab. Although the VWF in figure 20 and figure 22 has the outputs listed in ascending order, and the simulated waveform in the lab has outputs listed in descending order. The pattern is still obvious and occurs in both waveforms. This is true for both the 8-bit LFSR as well as the 16-bit LFSR.

Conclusions:

In this completing this lab, I learned about the Shift Register in detail. The Shift Register is a type of sequential logic circuit that can be used to for storage or transfer of binary data. There are generally 4 different modes that shift registers operate in: Serial-in to Serial-out, Serial-in to Parallel-out, Parallel-in to Serial-out, and Parallel-in to Parallel-out. I also learned that the directional movement of the data through a shift register can occur in one of many ways that is, either to the left, to the right, left-in but right-out, or both left and right shifting within the same register also known as bidirectional. Since this lab focuses on SIPO, PISO, and LFSR, I will explain the operation of those specific shift registers. The operation of SIFO shift register is that on each clock pulse, the data contents of each stage are shifted one place to the right. This allows the data output to be read from the outputs of Qa to Qd. The operation of the PISO shit register is the opposite of the SIPO shift register, in that data is loaded into the register in a parallel format and data is read out sequentially right shifted from the register at outputs Qa to Qd. The operation of the LFSR is like a standard shift register except that its output is fed back into the input in such a way to cause an endlessly cycling sequence of patterns.

Citations:

Kuphaldt, Tony R. “Shift Registers: Serial-in, Parallel-out (SIPO) Conversion.” *Lessons in Electric Circuits*, 27 Nov. 2007, https://www.ibiblio.org/kuphaldt/electricCircuits/.

Kuphaldt, Tony R. “Shift Registers: Parallel-in, Serial-out (PISO) Conversion.” *Lessons in Electric Circuits*, 27 Nov. 2007, https://www.ibiblio.org/kuphaldt/electricCircuits/.