Lab Experiment No. 09:

Data Encryption Using LFSRs

By: Alexander Lukens (A20417036)
Instructor: Dr. Jafar Saniie
ECE 441-001
Lab Completion Date: 06/20/2020

Acknowledgement: I acknowledge all of the work (including figures and codes) belongs to me and/or persons who are referenced.

Signature:____Alexander Lukens____

I. Introduction

A. Purpose

The main purpose of this experiment is to introduce students to synchronous logic design in VHDL using the Xilinx Vivado software ecosystem. Students will design a system to encrypt and decrypt bytes of data using a shared key and XOR and linear feedback shift registers (LFSRs). This will allow students to securely encode and transmit any byte size data (such as ascii characters).

B. Background

Xilinx is a technology company based in California, USA that creates field programmable gate array (FPGA), system-on-chips (SoCs), and other semiconductor products. They compliment their hardware designs with a robust software environment for designing digital logic circuits using System Verilog and VHSIC-Hardware Description Language (VHDL). This allows silicon engineers to rapidly develop and test digital logic designs without physically designing them on a breadboard or waiting for dedicated, application specific silicon to be produced.

In this lab, a basic technique for data encryption will be explored. Encryption refers to a method for taking raw data that is generally readable and meaningful to others, and translating the data into a form where the contents are not readable or known by unauthorized parties. This is extremely convenient for transmitting confidential information, or any other information that should not be seen by external parties. A simple form of encryption can be implemented by utilizing a shared key between the communicating devices, and performing the logical operation XOR on the raw data. This will result in the data becoming incomprehensible to individuals who do not know the shared key.

II. Lab Procedure and Equipment List

A. Equipment

Equipment required for this lab includes the following:

- Xilinx Artix-7 based FPGA
- PC with Xilinx Vivado software

B. Procedure

Students will follow the instructions outlined in the ECE 742 - FPGA Lab Manual for experiment 9. This includes designing and implementing a circuit to provide basic encryption and decryption for byte size data . To accomplish this goal, students must first design and

implement a linear feedback shift register to hold and process the encryption key. This register will be synchronous and have parallel load capabilities to allow users to easily load encryption keys. Then, students will design additional logic to implement the LFSR as a VHDL component and use the generated key to encrypt and decrypt bytes of data provided in the laboratory manual.

III. Results and Analysis

A. Discussion

First, in order to create the encryption, a shift register with linear feedback must be implemented. This register will hold the encryption key that is altered with each byte processed. This shift register must be synchronous and have parallel load capabilities in order to load the entire register in one cycle and to remain in sync with the rest of the encryption circuit.

Linear feedback shift register VHDL

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;


entity LFSR is
   Port ( inputByte : in STD_LOGIC_VECTOR (7 downto 0);
          singleShift : in STD_LOGIC :='0';
          loadEnable : in STD_LOGIC :='0';
          OutputByte : out STD_LOGIC_VECTOR (7 downto 0));
end LFSR;

architecture Behavioral of LFSR is
signal CurrentByte,tempByte : STD_LOGIC_VECTOR(7 downto 0);

begin


  process(singleShift,loadEnable)
   begin
      tempByte <= CurrentByte; --store tempByte
      if(rising_edge(singleShift)) then  --when shifting once
      tempByte <= CurrentByte; --store tempByte
      CurrentByte(0) <= tempByte(1);
      CurrentByte(1) <= tempByte(2);
      CurrentByte(2) <= tempByte(3);
      CurrentByte(3) <= tempByte(0) XNOR tempByte(4);
      CurrentByte(4) <= tempByte(0) XNOR tempByte(5);
      CurrentByte(5) <= tempByte(0) XNOR tempByte(6);
      CurrentByte(6) <= tempByte(7);
      CurrentByte(7) <= tempByte(0);
```

```
    elsif (loadEnable = '1') then
    CurrentByte <= inputByte;
    end if;
  end process;


  OutputByte <= CurrentByte;

end Behavioral;
```

The second VHDL file utilizes the LFSR created in the previous VHDL file by incorporating it into a VHDL component. Additional logic in the Encrypt/Decrypt VHDL file utilizes a for-loop to perform bitwise XOR operations on each bit of the Encryption Key and Data String. This for loop will either encrypt or decrypt the data string depending on the current status of the data.

The Digilent Arty A7 FPGA does not have enough switches and LEDs to efficiently utilize the encryption, therefore the VHDL file was only simulated in Vivado and not configured for implementation on an FPGA. To implement this circuit on an FPGA, an additional VHDL file must be created that includes the corresponding I/O pins and variables and a constraints file incorporated. Additionally, the debounce component should be implemented when using an FPGA and physical button in order to ensure a clean transition between states when pressing the button.

EncryptDecrypt8 VHDL file

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity encryptDecrypt8 is
port(
    clk : in std_logic;
    inStringByte : in std_logic_vector(7 downto 0); --data to be encrypted/decrypted
    EncryptEnable : in std_logic;
    outStringByte : out std_logic_vector(7 downto 0)
    );
end encryptDecrypt8;


architecture behav of encryptDecrypt8 is


component LFSR is
  Port ( inputByte : in STD_LOGIC_VECTOR (7 downto 0);
```

```
        singleShift : in STD_LOGIC;
        loadEnable : in STD_LOGIC;
        OutputByte : out STD_LOGIC_VECTOR (7 downto 0));
end component;

-- .. End of Component Specify

-- Specify Signals here..
signal KeyLoad,ActiveKey : std_logic_vector(7 downto 0) :="00000000";
signal loadEnable,shift_enable : std_logic :='0';
-- EndofSignalSpecification

begin
   shift_enable <= EncryptEnable AND clk; --when EncryptEnable is asserted, one byte will be decoded per clk
cycle
   ShiftReg : LFSR port map(inputByte => KeyLoad, singleShift => shift_enable, loadEnable => loadEnable,
OutputByte => ActiveKey);


   process(shift_enable)
   begin
      for i in 0 to 7 loop
         outStringByte(i)<= ActiveKey(i) XOR inStringByte(i); --encrypt/decrypt mechanism does bitwise XOR
operation on data byte
      end loop;
   end process;

end behav;
```

Decryption sequence init. key = C7

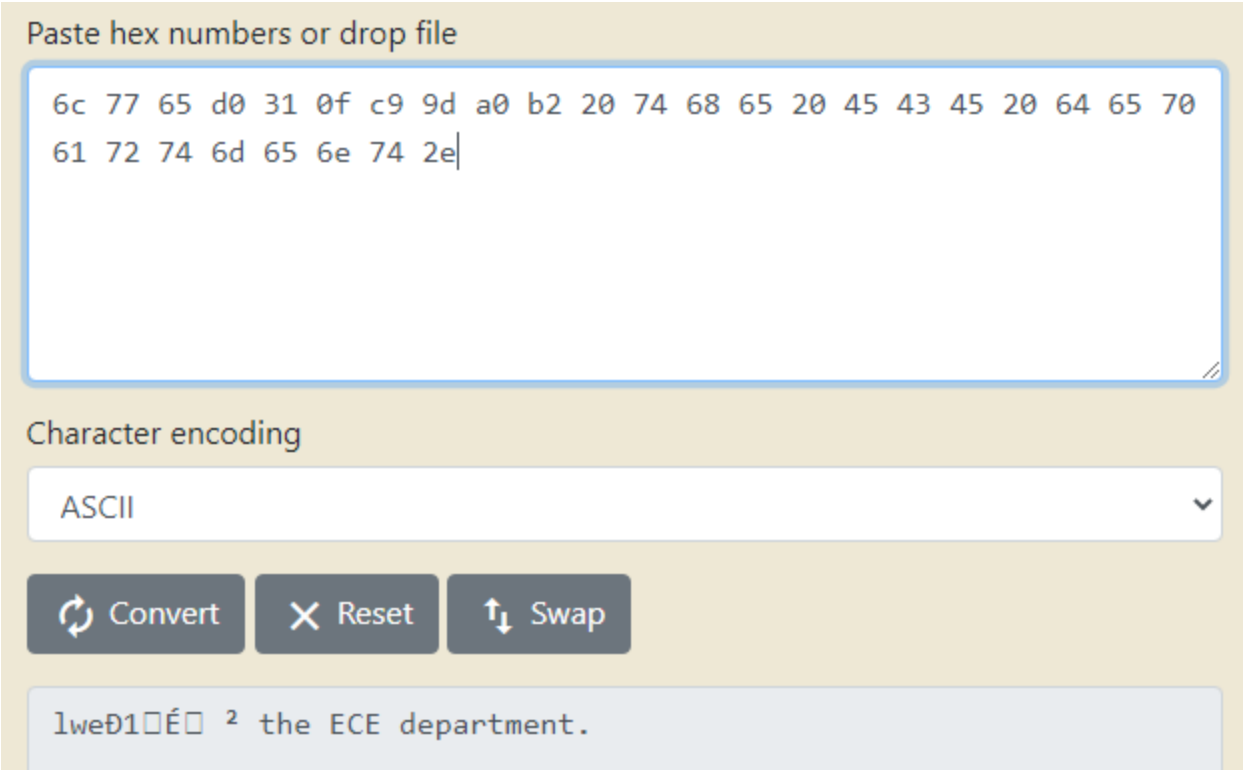
original	AB	94	94	28	75	15	FC	07	D5	08
decoded	6C	77	65	d0	31	0f	C9	9d	a0	b2
original	92	15	D8	05	28	79	65	6E	B5	AE
decoded	20	74	68	65	20	45	43	45	20	64
original	38	DE	OE	C5	AF	80	93	2D	D5	FE
decoded	65	70	61	72	74	6d	65	6e	74	2e

Encryption "Goodbye" 47 6F 6F 64 62 79 65

init. key = 72

encrypted string 35 6e ef 1c 66 43 40

Shown above is the decryption and encryption practice given on the lab manual worksheet. It is concerning that in the decryption example, the first half of the ascii string does not appear to be correct, but the second half does. I do not believe that there was a functional problem with my implementation of the encryption circuit, as this would have led to a completely incorrect string decoding. This leads me to believe that there is an issue with the encoded string provided to students. The decrypted string, converted to ASCII is shown below.



IV. Conclusion

This lab should be considered a complete success. Students were successfully able to implement a circuit to encode and decode bytes of data using VHDL. Furthermore, students were successful in simulating the circuit using Xilinx Vivado, however the results of the encrypted example message were inconclusive. In the future, students can expand on this VHDL circuit design by implementing it on a Xilinx FPGA with an adequate amount of I/O. Additionally, students could potentially utilize pre-built IP blocks to interface with the FPGA and perform encryption and decryption using a serial terminal on a computer. The skills learned in this laboratory will allow students to design and implement more complex circuits in the future.

References

[1] Wikipedia contributors. (2020, June 14). Xilinx. In *Wikipedia, The Free Encyclopedia*. Retrieved 02:24, June 16, 2020, from https://en.wikipedia.org/w/index.php?title=Xilinx&oldid=962581047
[2] FPGA Lab Manuals - ECE 742