Experiment No. 9
Encryption Using LFSRs
ECE 446

Peter Chinetti

November 4, 2014

Instructor: Professor Shanechi

1 Introduction

Data encryption is a critical field in modern telecommunications, however, it is typically an computationally expensive procedure if done using general purpose procesors. FPGAs can be configured to quickly encode/decode signals ato offload the task of encryption.

LFSRs are a method to create a pseudorandom sequence of bytes. It is created with a combination of XOR gates and D flip flops.

The encryption method used is simple: XOR a secret with a codeword generated by the LFSR, and transmit. Upon recepit: XOR again with the same LFSR codeword to recover the secret.

2 Procedure

a. Write VHDL to implement encoder/decoder logic.

b. Assign pins to ports

c. Simulate

3 Equipment

- PC
- Spartan-3E development board

4 Code

4.1 Top-level Module

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity encryptor is
    Port ( SEED : in  STD_LOGIC_VECTOR (7 downto 0);
           MIN : in  STD_LOGIC_VECTOR (7 downto 0);
           MOUT : out  STD_LOGIC_VECTOR (7 downto 0) := x"00";
        CRYPT : out STD_LOGIC_VECTOR (7 downto 0);
           CLK : in  STD_LOGIC;
           LOAD : in  STD_LOGIC;
        EN : in STD_LOGIC);
end encryptor;

architecture Behavioral of encryptor is

  signal channel : STD_LOGIC_VECTOR(7 downto 0);

  component LFSR
  Port ( mout : out  STD_LOGIC_VECTOR (7 downto 0);
         min, seed: in  STD_LOGIC_VECTOR (7 downto 0);
         clk : in  STD_LOGIC;
         load : in  STD_LOGIC;
         en : in STD_LOGIC);
  end component;
begin
  lfsr_0: LFSR
    port map(
      mout => channel,
      min => MIN,
      seed => SEED,
      clk => CLK,
      load => LOAD,
      en => EN
    );

  lfsr_1: LFSR
    port map(
      mout => MOUT,
      min => channel,
      seed => SEED,
      clk => CLK,
      load => LOAD,
      en => EN
    );
    CRYPT <= channel;
end Behavioral;
```

encryptor.vhd

4.2 LFSR

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

entity LFSR is
    Port ( mout : out  STD_LOGIC_VECTOR (7 downto 0);
           min, seed: in  STD_LOGIC_VECTOR (7 downto 0);
           clk : in  STD_LOGIC;
           load : in  STD_LOGIC;
        en : STD_LOGIC );
end LFSR;

architecture Behavioral of LFSR is
  signal d : STD_LOGIC_VECTOR (7 downto 0);
  signal q : STD_LOGIC_VECTOR (7 downto 0):= x"34";
  signal clk_i : STD_LOGIC;
begin
  clk_i <= en and clk;

  process (clk_i)
  begin
    if rising_edge (clk_i) then
      q <= d;
    end if;
  end process;

  process (load, min)
  begin
    if load='0' then
      for i in 0 to 7 loop
        if i=7 then
          d(7) <= q(0);
        elsif i>2 and i<6 then
          d(i) <= q(0) xnor q(i+1);
        else
          d(i) <= q(i+1);
        end if;
      end loop;
    elsif load = '1' then
      d  <= seed;
    end if;
  end process;

  mout <= q xor min;

end Behavioral;
```

LFSR.vhd

4.3 Test

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;


ENTITY encryptor_test IS
```

```
END encryptor_test;

ARCHITECTURE behavior OF encryptor_test IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT encryptor
    PORT(
         SEED : IN  std_logic_vector(7 downto 0);
         MIN : IN  std_logic_vector(7 downto 0);
         MOUT : OUT  std_logic_vector(7 downto 0);
         CLK : IN  std_logic;
         LOAD : IN  std_logic;
       EN : IN std_logic
        );
    END COMPONENT;


   --Inputs
    signal SEED : std_logic_vector(7 downto 0) := x"34";
    signal MIN : std_logic_vector(7 downto 0) := (others => '0');
    signal CLK : std_logic := '0';
    signal LOAD : std_logic := '1';
   signal EN : std_logic := '0';

   --Outputs
    signal MOUT : std_logic_vector(7 downto 0);

    -- Clock period definitions
    constant CLK_period : time := 10 ns;

BEGIN

   -- Instantiate the Unit Under Test (UUT)
    uut: encryptor PORT MAP (
           SEED => SEED,
           MIN => MIN,
           MOUT => MOUT,
           CLK => CLK,
           LOAD => LOAD,
         EN => EN
          );

   -- Clock process definitions
    CLK_process :process
    begin
     CLK <= '0';
     wait for CLK_period/2;
     CLK <= '1';
     wait for CLK_period/2;
    end process;


   -- Stimulus process
    stim_proc: process
    begin
       -- hold reset state for 100 ns.
```

```
63          wait for 100 ns;

65          wait for CLK_period*10;

67          -- insert stimulus here

69      SEED <= x"FF";

71      wait for 10 ns;

73      EN <= '1';
        SEED <= x"34";
75
        wait for 10 ns;
77
        LOAD <= '0';
79      MIN <= x"67";

81      wait for 10 ns;

83      MIN <= x"6A";

85      wait for 10 ns;

87      MIN <= x"D1";

89      wait for 10 ns;

91      EN <= '0';

93          wait;
    end process;
95
END;
```

encryptor_test.vhd

5 Postlab

The decoded sequence reads "Hello from the ECE department." "Goodbye" encoded becomes 5B 59 4C F5 AA 25 73. Forwarding the LFSR three states forwards gives:

```
d(7) <= q(2);
d(6) <= q(1);
d(5) <= '0';
d(4) <= q(7) xnor q(0) xnor q(1) xnor q(2);
d(3) <= q(6) xnor q(0) xnor q(1) xnor q(2);
d(2) <= q(5) xnor q(0) xnor q(1);
d(1) <= q(3);
```

snip.vhd

6 Conclusions

The purpose of this lab was achieved. A LFSR based FPGA encryptor was built and tested. Operation was verified through simulation.