Midterm Exam

ECE 448

Spring 2014

Thursday, March 6

(15 points)

Instructions:

Zip all your deliverables into an archive <last_name>.zip and submit it through Blackboard no later than Thursday, March 6, 10:15 PM EST.

Lab Midterm Exam

Introduction:

The described below circuit performs encryption of an 8-bit sequence of bits, using a simple algorithm based on four Linear Feedback Shift Registers (LFSRs) and an adder. At the beginning of operation, four Initialization Vectors (IVs) are loaded into four LFSRs one by one using enable signals generated by a 2-to-4 decoder. Once the LFSRs are initialized, message bits are entered into the circuit bit by bit, and the corresponding ciphertext bits are generated using an XOR of a message bit with the corresponding keystream bit.

The result of encryption depends on the IVs (which can be chosen at random on the sender's side and transmitted in clear to the receiver's side), as well as the secret key that determines the coefficients of all LFSRs.

Interface:

Assume the following interface to your circuit.

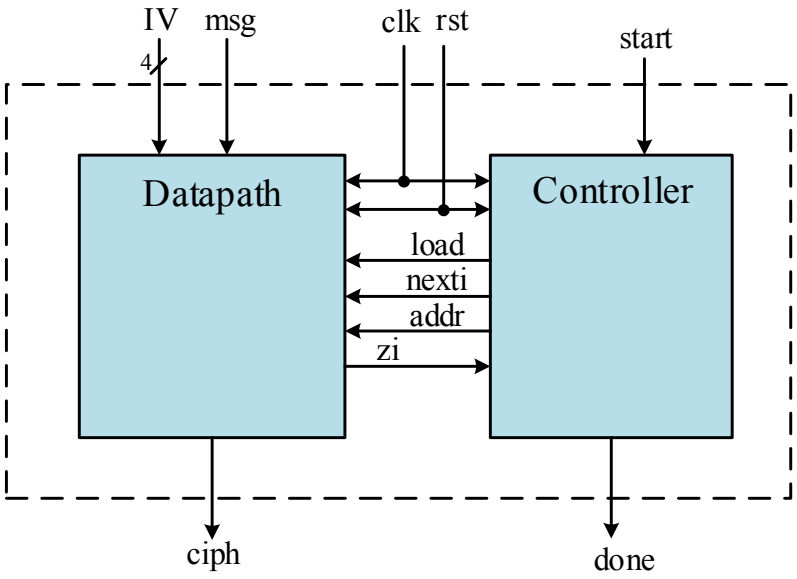


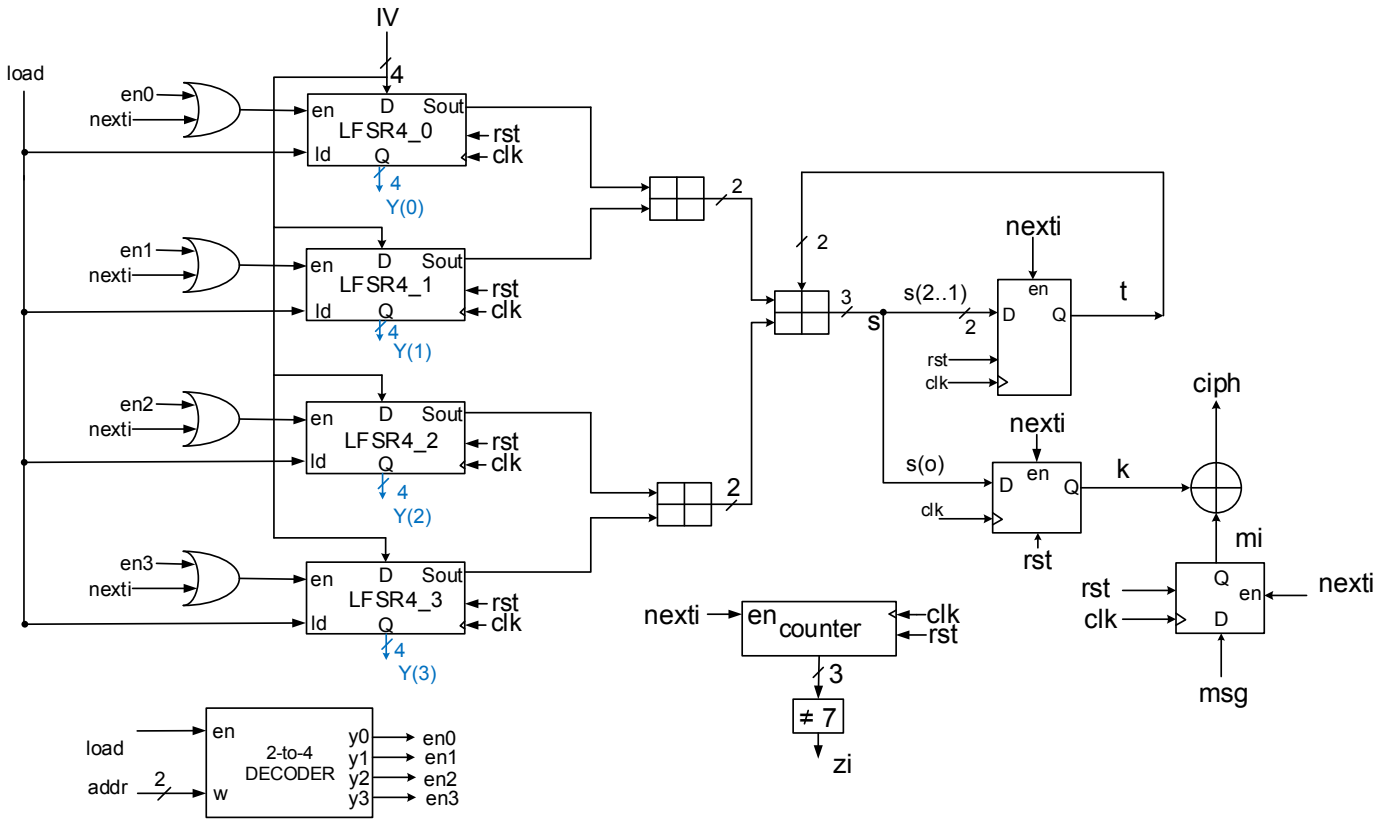
Table of input/ output ports:

Port	Mode	Width	Function
clk	IN	1	System clock
rst	IN	1	Asynchronous system reset
IV	IN	4	Initialization Vector
msg	IN	1	Message bit
start	IN	1	Start encryption
ciph	OUT	1	Ciphertext bit
done	OUT	1	Encryption done

Table of control/status signals between datapath and controller:

Port	Direction	Width	Function
load	To datapath	1	Loads the LFSRs with IVs
nexti	To datapath	1	Controls the LFSRs, the counter, and the message input
addr	To datapath	1	Index of an initialized LFSR
zi	To controller	1	Indicates that the counter has reached its maximum value

Block Diagram:



Linear Feedback Shift Register (LFSR):

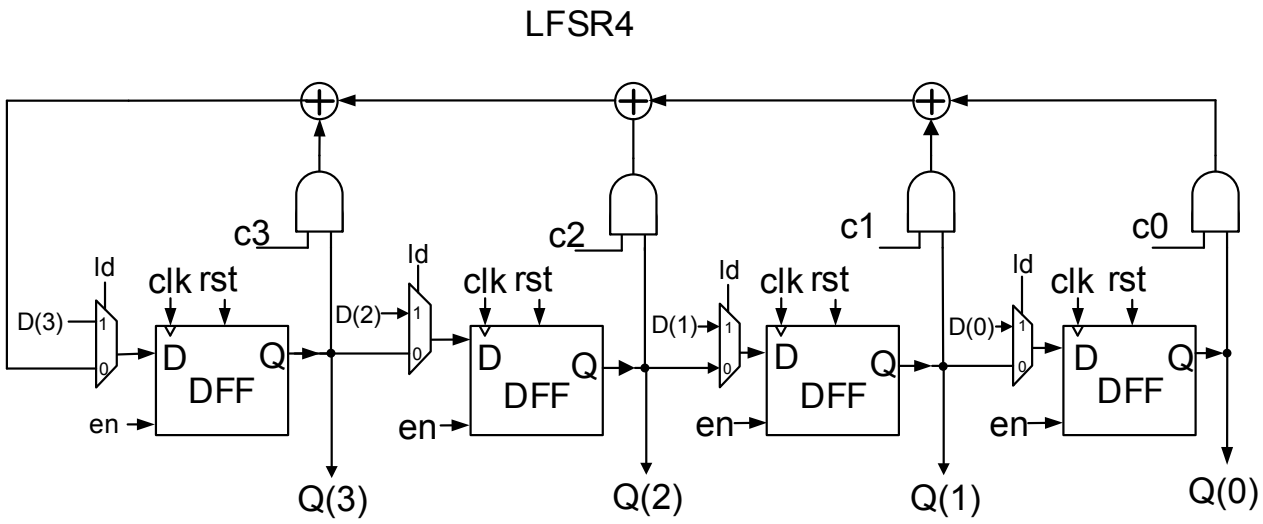
Linear feedback shift register is a sequential circuit shown in the diagram below.

Coefficients c3, c2, c1, c0 form a 4-bit vector c, which is different for each of the instantiated LFSRs.

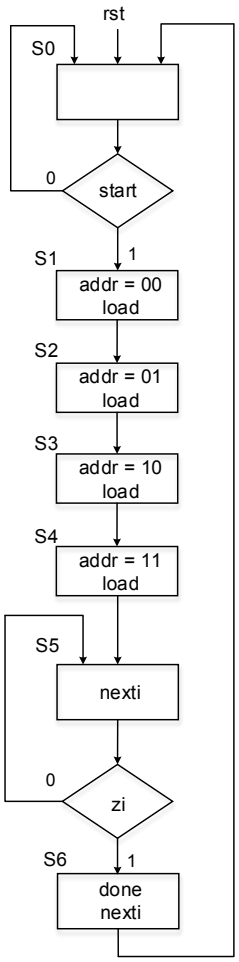
LFSR4_0: c="0011" = X"3"
LFSR4_1: c="1001" = X"9"
LFSR4_2: c="1011" = X"B"
LFSR4_3: c="1101" = X"D".

In your VHDL code, a particular value of the coefficient vector c should be provided using a generic c.

Block Diagram of LFSR:



ASM Chart:


rst
S0
0
start
1
S1
addr = 00
load
S2
addr = 01
load
S3
addr = 10
load
S4
addr = 11
load
S5
nexti
0
zi
1
S6
done
nexti

Test vectors:

Inputs:

IV for LFSR4_0 = "0011"
IV for LFSR4_1 = "0101"
IV for LFSR4_2 = "0111"
IV for LFSR4_3 = "1001"

msg = 0 1 1 0 1 0 1 1

Output:

ciph = 0 1 1 1 1 0 1 1

Table with expected values of intermediate signals after initialization:

Count Value	Y0 (4-bit)	Y1 (4-bit)	Y2 (4-bit)	Y3 (4-bit)	s (3-bit)	k (1-bit)	t (2-bit)	mi (1-bit)	ciph (1-bit)
0	0011	0101	0111	1001	100	0	00	-	-
1	0001	1010	0011	0100	100	0	10	0	0
2	1000	1101	0001	1010	100	0	10	1	1
3	0100	0110	1000	1101	011	0	10	1	1
4	0010	0011	1100	1110	010	1	01	0	1
5	1001	1001	1110	0111	100	0	01	1	1
6	1100	0100	0111	0011	100	0	10	0	0
7	0110	0010	0011	1001	100	0	10	1	1
0	1011	0001	0001	0100	101	0	10	1	1

Design Requirements:

The combinational portion of the circuit should be described using the dataflow VHDL code, and the sequential portion of the circuit should be described using the synthesizable behavioral code. Your code should infer a circuit that requires a minimum amount of FPGA resources. The target clock frequency should be **100 MHz**.

Tasks:
Perform the following tasks:

1. Write a synthesizable VHDL code representing the encryption circuit
2. Write a testbench verifying the operation of your circuit.

3. Perform functional simulation of your circuit and use it to debug your VHDL code. Take a print out of the waveform showing the entire operation using default PDF conversion tool installed in the lab (Use multiple page option in order to display necessary information on multiple pages, if required).
4. Synthesize your circuit.
5. Implement your circuit using
    a. FPGA family: Spartan 6
    b. Device: xc6slx16-3csg324
    c. Speed Grade: -3
6. Run the static timing analysis of your circuit.
7. Based on the circuit block diagram and the report from the static timing analysis, determine the most critical path in your circuit and the circuit maximum clock frequency.
8. Based on the implementation reports, determine the number of CLB slices, LUTs, flip-flops, and pins used by the circuit.
9. Perform the timing simulation of your circuit at the maximum clock frequency returned by the static timing analysis. Take a printout of the waveform showing the entire operation using default PDF conversion tool installed in the lab (Use multiple page option in order to display necessary information on multiple pages, if required).

Deliverables:

1. VHDL code of your entire circuit fulfilling the requirements specified in the *Design Requirements* section above.
2. VHDL code of your testbench.
3. Timing waveforms from the functional and timing simulations demonstrating the correct operation of your circuit.
4. Description of the critical path in your circuit
5. FPGA resource utilization (as defined in Task 8 above).
6. Minimum clock period and maximum clock frequency of your circuit.