HACETTEPE UNIVERSITY
ELECTRICAL AND ELECTRONICS ENGINEERING
ELE227 FUNDAMENTALS OF DIGITAL SYSTEMS LABORATORY
FALL 2019

Experiment 5 – Implementation of Registers, Counters, RAMs and ROMs

Preliminaries:

1) Students who will attend this experiment are assumed to know:
   a. Basic number systems and Boolean algebra
   b. Basic combinational logic operations and gates
   c. Basic VHDL operators and operands
2) Study related pages in the textbook.

Work:

- Basic Logic Functions are "**and, or, not, nand, nor, xor, xnor**"
- **For each question first design your circuit on the paper!!!** (If you start directly with the VHDL code, you will get zero points).
- Then write the VHDL code of each module and testbench of the overall circuit. Include the RTL schematics and testbench graphs.
- Insert comments in your code. Delete all unnecessary comments inserted by the software.
- Avoid too many pages. Present your work in the same sequence in this paper. The number of the answers of each question has to be included.

1) Write a VHDL code to implement:
   a. 4-bit universal shift register by using T-FlipFlops.
   b. 4-bit up/down counter by using D-FlipFlops.
   c. 4-bit Johnson counter by using T-FlipFlops.
   d. 4-bit asynchronous binary ripple counter by using JK-FlipFlops.

2) Design a sequential circuit that performs division by repeated subtractions. At each clock cycle cct performs subtraction until a stop condition occurs. The number of subtractions have to be stored as the quotient. Remainder has to be stored in a register.

Inputs: A (A2A1A0) is dividend, B(B2B1B0) is divisor.

Outputs: Q (Q2Q1Q0) is quotient, R (R2R1R0) is remainder.

The circuit has to perform the following algorithm:

I) Initialize cct: remainder = A and quotient Q=0.
II) Enable cct to perform as a sequential cct.
III) For each clock cycle circuit performs the subtraction $R_{n+1} = R_n - B$.
IV) R_n is stored in a 4-bit register with parallel load. R_{n+1} is the output of the subtractor and input of the 4-bit register.
V) If $R_n >= B$, increment Q and assign $R_n = R_{n+1}$ with the clock cycle, else hold R_n and Q as the remainder and quotient, respectively.

Use a 4-bit subtractor (adder-subtractor of pre#1 can be used), 4-bit register with parallel load, 4-bit comparator and a 4-bit binary counter to store Q.

3) In mathematics, convolution operation on discrete domain is defined as

$$y[n] = \sum_{k=0}^{K-1} g[n-k]x[k]$$

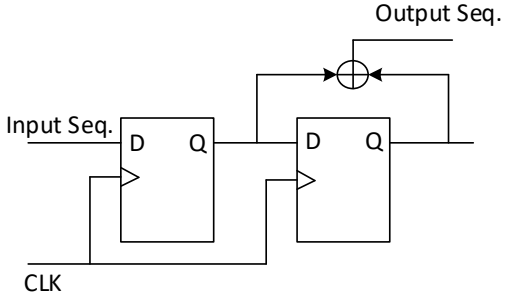
where *K* is equal to the minimum of sizes of sequences x[n] and g[n], i.e. *K* = min{size(*x*[*n*]), size(*g*[*n*])}. Besides, size of *y*[*n*] is determined by the adding size of *x*[*n*] and *g*[*n*] sequences and subtracting 1, i.e. size(*y*[*n*]) = size(*x*[*n*]) + size(*g*[*n*]) - 1.
For example, let *x* = [1 -1 2] and *g* = [1 2]. Then sequence *y* can be calculated by following steps.

Step 1: *n* = 0
y[0] = *g*[0]*x*[0] = 1 (negative indices of sequence *g* are equal to zero!)
Step 2: *n* = 1
y[1] = *g*[1]*x*[0] + *g*[0]*x*[1] = 2 - 1 = 1
Step 3: *n* = 2
y[2] = *g*[1]*x*[1] + *g*[0]*x*[2] = -2 + 2 = 0
Step 4: *n* = 3
y[3] = *g*[1]*x*[2] = 4

Hence, sequence y = [1 1 0 4] is obtained. If you want to do this operation in binary as in the lab, then the only thing you have to do is changing sum operator with modulo-2 adder.
For example, let x = [1 0 1] and g = [1 1] be two binary sequences.

Step 1: *n* = 0
y[0] = *g*[0]*x*[0] = 1
Step 2: *n* = 1
y[1] = *g*[1]*x*[0] ⊕ *g*[0]*x*[1] = 1 ⊕ 0 = 1
Step 3: *n* = 2
y[2] = *g*[1]*x*[1] ⊕ *g*[0]*x*[2] = 0 ⊕ 1 = 1
Step 4: *n* = 3
y[3] = *g*[1]*x*[2] = 1

At the end, *y* = [1 1 1 1] sequence is obtained. By using shift register outputs, this convolution operation can be performed by following design with adjusting initial states of flip-flops to zero.



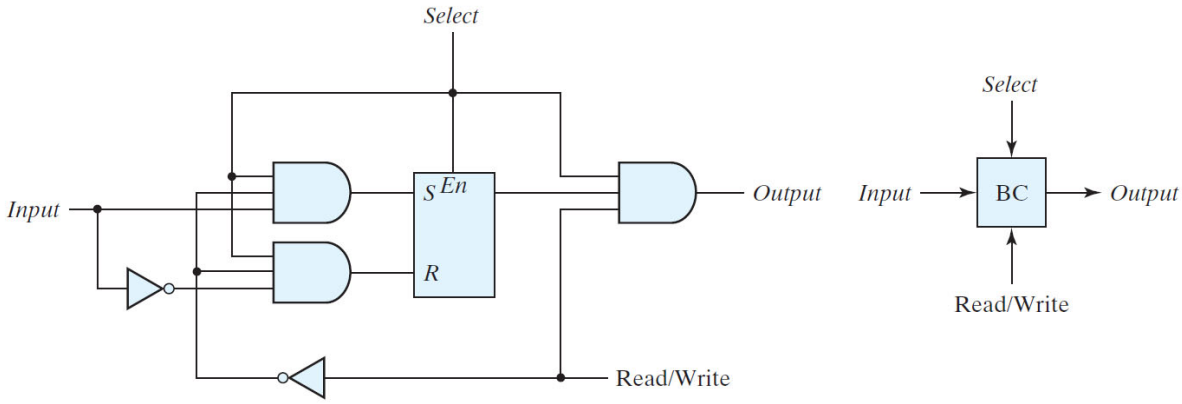
This circuitry is named as “convolutional encoder” and the resulting sequence is “convolutional code” in digital communications literature. Notice that on the above circuit, because of initial state conditions, desired output of this circuit will be lagged by a clock pulse. Means that y = [0 1 1 1 1].

Now, it is asked to you design a circuit that performs convolution with

- *g* = [1 0 0 1] when select bit is ‘1’.
- *g* = [1 1] when select bit is ‘0’.

with input sequence x = [1 0 1 0 1 0 1 1]. Use flip flops as few as you can in your design.

4) The binary storage cell is the basic building block of a memory unit. The equivalent logic of a binary cell that stores one bit of information is shown in the figure below. The storage part of the cell is modeled by an SR latch with control input (see Textbook Chapter 5.3) and associated logic gates.



(a) Logic diagram (b) Block diagram

a) Implement a simple binary cell (BC) that has equivalent circuit in the figure above in VHDL. Test your design with following cases:
   - Input : 0, Select : 1, Read/Write : 0
   - Input : 1, Select : 0, Read/Write : 0
   - Input : 1, Select : 1, Read/Write : 1
   - Input : 1, Select : 1, Read/Write : 0
   - Input : 1, Select : 1, Read/Write : 1
   - Input : 0, Select : 0, Read/Write : 0
   - Input : 1, Select : 1, Read/Write : 1

b) Implement a small RAM which consists of 4 words of 4 bits using required amount of BCs, decoder(s) and other necessary logic gates in VHDL (see Textbook Chapter 7.3).

c) Test your RAM design by writing words '0', '1', '2', '9' (in binary) to consecutive addresses from 0 to 3. After 4 write operations, apply read operations to the memory addresses in following order: 2, 0, 1, 3, 2, 0, 1, 3, 2, 0, 1, 3…

5) Design a combinational circuit using a ROM (see Textbook Chapter 7.5). The circuit accepts a three-bit number and outputs a six-bit binary number equal to the square of the input number. Implement this ROM only using a 3-to-8 decoder and minimum number of basic logic gates. Add your truth table and design to your report. Test your design for all possible inputs.