UNIVERSITY of GUELPH

School of Computer Science
Faculty of Engineering and Computer Science

Student Information

First Name

Last Name

Student ID Number

Lab Cover Page

Please complete all (empty) fields:

Course Name: DIGITAL SYSTEMS 1

Course Number: CIS3120 Section Number: ___________________

Teaching Assistant to whom you are submitting

First Name: ______________________ Last Name: ______________________

Mark Assigned: __________

Notes

This (complete) lab exercise should be printed and stapled before coming to lab. Any questions requiring a written answer should be answered by writing in the spaces provided in this lab document, and the document should be submitted for marking. Remember to keep this document for study purposes. Also, you will need the signed version of this document if a re-grade or grade correction is requested.

Laboratory Exercise 11

In this final laboratory exercise you will design a *code converter*. In doing so, you will obtain additional experience designing sequential circuits, with a special emphasis on state graph construction and state reduction.

Your task is design a sequential circuit to convert Binary Coded Decimal (BCD) to Excess-3 code. As discussed several times in class, BCD is a straight assignment of the ten decimal digits 0 through 9 to binary. Excess-3 is another code used in some older computers to represent decimal digits, and can be found by taking the BCD code for a decimal digit and adding 3. Your task, therefore, is to design a sequential circuit that adds three to a BCD digit in the range of 0 to 9. The input and output of your circuit will be serial with the least significant bit appearing first. A list of the allowed input and output sequences is shown in Table 1 below.

Table 1: BCD to Excess-3

X Input (BCD)	Z Output (excess-3)
t_3 t_2 t_1 t_0	t_3 t_2 t_1 t_0
0 0 0 0	0 0 1 1
0 0 0 1	0 1 0 0
0 0 1 0	0 1 0 1
0 0 1 1	0 1 1 0
0 1 0 0	0 1 1 1
0 1 0 1	1 0 0 0
0 1 1 0	1 0 0 1
0 1 1 1	1 0 1 0
1 0 0 0	1 0 1 1
1 0 0 1	1 1 0 0

The previous table lists the desired input and outputs at times t_0, t_1, t_2, and t_3. After receiving four inputs, the circuit should reset to the initial state, ready to receive another group of four inputs – one bit at a time.

It is important to note that a cursory glance of Table 1 may leave you feeling somewhat perplexed, as it is not readily obvious that a sequential circuit can be actually be realized to produce the output sequences specified in the table on a clock cycle by clock cycle basis.

For example, if at time t_0 some sequences required an output $Z = 0$ for $X = 0$, while other sequences required $Z = 1$ for $X = 0$, it would be impossible to design the circuit to output the correct value. However, for this particular application, this problem does not exist. From Table 1 we can see that at t_0 if the input is 0 the output is always 1 and vice-versa. Therefore, there is no conflict at time t_0. Similarly, there are no conflicts at times t_1, t_2, and t_3.

Following the procedure given in class, start by creating a correct state diagram. Your state graph should be based on a Mealy machine, where the output is associated with the arcs in the graph (i.e., combinational logic) rather than the vertices (flip-flops). Ultimately, your state graph will have the form of a tree, with each path starting at the reset state representing one of the 10 possible input sequences. Hint: First construct the paths for each input sequence, then work backwards from the leaf vertices to the root vertex to fill in the output on each arc.

Having completed the state graph, convert the graph into a state-transition table. Observe that you will have some don't'-care conditions in your state table, because only 10 of the possible 16 (4-bit) sequences can occur as inputs to the code converter.

Next, you will reduce the number of states in the state table using the row-matching technique given in class. When matching rows which contain don’t-cares, a don’t-care will match with any state or with any output value.

How many states, if any, were you able to remove?

How many flip-flops do you require to implement the state machine?

Redraw the final state table. Note: You will need to encode each state in binary.

Complete your design, by implementing the circuit using D flip-flops.