Name: ***Roll #***

Laboratory 8 – Bebop to the Boolean Boogie*

A: /3 B: /4 C: /8 + 5 bonus

Goal:

This lab is split in three parts of increasing difficulty:

A) (Somewhat difficult)
We will learn a simple technique to store *n* bits of data into an *n* bit memory given a single serial input line to the memory. We will work with $n=4$

B) (More difficult, though not too much if you have understood Part A)
We will use the technique developed in part A to make the numbers 'dance' around (hence the title of the assignment) – this can be used to create blinking lights like those used on festive occasions

C) (Very difficult)
We will enhance the technique of Part B to a non-trivial level – and demonstrate how to make the numbers dance around in a random manner to create a sequence of 4 bit (pseudo)random numbers

Part A: Storage of *n* bits

Recall from earlier exercises that the reliable storage of <u>one</u> data bit requires the use of <u>one</u> D register.

Design a memory circuit that is able to store $n=4$ bits of data $D_3D_2D_1D_0$ with the following constraints:

1. The circuit is driven by a clock signal.
2. The input to your circuit is a *single* serial line. At every clock cycle $i = 0,1,2,3$ the values of the data bits D_0 D_1 D_2 D_3 (in that order) are made available on the input.

Draw your circuit design here:

Build your design with the 74LS175 4-bit D register IC. It is useful to connect an LED+220Ω at the output of each D-register to clearly indicate the status of the bit.

Demonstrate its operation by using the 1 Hz clock and sending in a simple input like 10001 using the push button switch on the logic signal generator. Note that after the 4th clock input, your memory will start getting overwritten!

A: /3

**Bebop to the Boolean Boogie: An unconventional guide to electronics, ebook by Clive Maxfield*

Part B: Make *n* bits perform a rhythmic dance

Recall the use of decorative lights in recent festivities – they typically have a string of lights with the on/off states alternating in time to give an illusion of motion along the length of the lights.
Modify your circuit of Part B, to make such a dancing light pattern that repeats itself long as the clock signal is sent in.

Hints:

1) 'Repetition' usually equates to some type of feedback
2) Each D register in the 4-bit 74LS175 has both a Q and Q-bar output
3) You can make a 4-bit lighting string for Ganesh Chaturthi by re-arranging one of the inputs & outputs in the design of Part A.
4) Only one extra connection is needed over the design of Part A
5) The D-register IC has a global reset pin $\overline{CLR}$. When it is connected to logic 0, all bits are cleared to 0.

Draw the (re)-design of your circuit here. Modify the connections of Part A on the breadboard to demonstrate its operation

Part C: Make *n* bits dance *randomly*

In Part B, the circuit was setup to repeat the same sequence of bits.
Inserting a *linear* operation in the repetitive loop can *change* the sequence of bits that is repeated.
Two such linear operations are A(X-NOR)B or A(XOR)B which are very much like algebraic multiplication: A(X-NOR)B=NOT[A(XOR)B]=1 only for A=B=0 and A=B=1; 0 if A≠B.

1) The objective of Part C is that at every clock cycle the sequence of bits $D_3D_2D_1D_0$ must be random
2) This can be done by using the original design of Part A, with the concept of repetitive feedback used in Part B.
3) The feedback must be modified by inserting a linear operation in the path. Tap two suitable bits D_A and D_B from the 4-bit memory and insert the D_A(X-NOR)D_B operation in the feedback path.

C:	/3 : Basic design
	/5 : Choice of D_A D_B
	/5 : Proof of randomness

Circuit Design for Part C here:

3 marks

Note: Initially you can choose by trial and error which points 'A' and 'B' to tap as D_A and D_B from $D_3D_2D_1D_0$ You must demonstrate that the sequence of numbers $D_3D_2D_1D_0$ on successive clock pulses at least *looks* random i.e. it does not get stuck at some fixed number which repeats forever.

-Demo -

5 marks

For those with a mathematical and/or Boolean inclination for BeBop:
For an arbitrary value of n bits calculate how long a sequence of random numbers $D_nD_{n-1}\ldots D_0$ such a design would go through before repeating itself. (It's easy to work this out explicitly with $n=3$)
The general formula is quite simple, though its derivation may require a little bit of work.
It leads to the fact that, for example, a 32-bit memory register with a single linear operation in the feedback can create a sequence of 4,294,967,295 random combinations before repeating.

5 marks (and probably a couple of extra pages!)