Name: ***Roll Number:***

Laboratory 2 – Combinational circuits → Sequential circuits

Learning Objectives:

1) You will first get a brief tutorial on how to use a simple logic display and pulse generator. You will also learn how to correctly draw a timing diagram for digital circuits.
2) The main aim is to get a thorough understanding of latches, registers and flip-flops: these are the basic building blocks of sequential logic circuits.
3) You will solve moderately difficult problems using the concepts learned.
4) It is assumed that you are familiar with basic digital gates: NAND, NOT etc

Introduction

Combinational logic circuits are 'memory-less': they simply apply the implemented logic on the inputs and pass the result to the output(s). The output of a combinational logic circuit quickly attains the value of the implemented function after the switching transitions have settled.

On the other hand it is often necessary for a circuit to 'remember' the result of a computation – this leads to the concept of sequential logic. One part of a circuit must be able to remember the result of an operation that will be used later.

In Boolean logic, memory elements are built of three basic logical building blocks. All these operate with a synchronized clock (CLK) input. The three basic memory blocks are:

- LATCH: Level sensitive – passes its input to its output on high level of CLK pulse, & stores the value when CLK is low. This is a positive latch; negative latch does the same on the opposite CLK phase.

- REGISTER: Edge-triggered – a positive register samples the input on the rising CLK edge, a negative register samples on the falling CLK edge.

- FLIP-FLOP: An element that has two inputs and two stable output states.

(Note that occasionally some books may use the terms register and flip-flop interchangeably – this can be slightly confusing. As a guide, remember that for practical use flip-flops generally have two inputs and two stable states, while latches and registers have a single input and a single state)

Fig 2.2 on the last page gives the block diagrams and state tables of these elements as a reference.

Procedure:

This assignment has three exercises. The concepts and circuits you use build up as you solve each, so its advisable to do them in sequence. Please demonstrate your circuits to the TA's and submit this assignment sheet filled out with answers where marked at the end of the session. For the circuit demonstrations, it is useful to connect red LED's to your circuit inputs and outputs. *The LED must be connected in series with a 220Ω current limiting series resistor to ground.*

/1+1+3	/3	/3+1+3	/15
Ex. 1	Ex . 2	Ex. 3 Design+timing++ckt	

Name:_____ Roll Number:_____

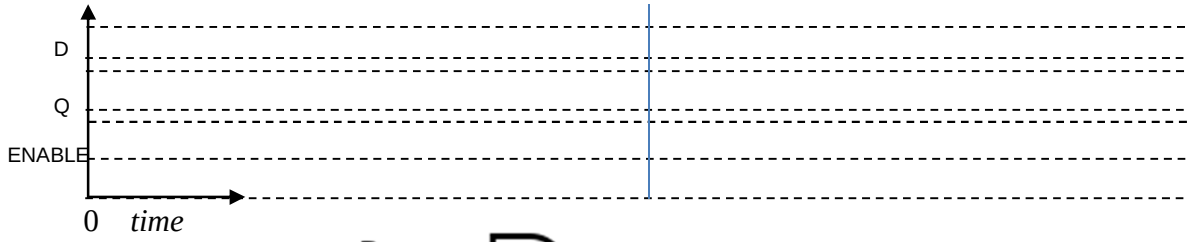
Exercise 1: A very slow memory element: the D-latch…

Wire up a D-latch shown in Fig 1.1 on your breadboard using a 74LS00 NAND IC. You may use LED's connected to the NAND gate inputs and output to make it easier to debug and demonstrate your circuit. *The ideal D latch stores the value of the input D when ENABLE is high, and presents the stored value at it's output Q. Look at Fig 1.1 and use* `(A NAND B) = 1 only if A=0 and B=0` *convince yourself that this is an ideal latch.*

1. Set the ENABLE input to 1. Send in D pulses using the supplied logic pulse generator. In this case, the interval between the D pulses is random since it depends on when you toggle the pulse switch on the pulse generator. Observe the output of the latch. What is it's starting value, and how does it change?_____
   _____

2. Now use the pulse generator to send a 1 Hz square wave into the D input of the latch. Send pulses manually into the ENABLE input of the latch using the logic pulse generator by toggling its pulse switch.
   What do you see on the output Q?_____
   _____
   _____
   _____

3. Explain your observations using a timing diagram:



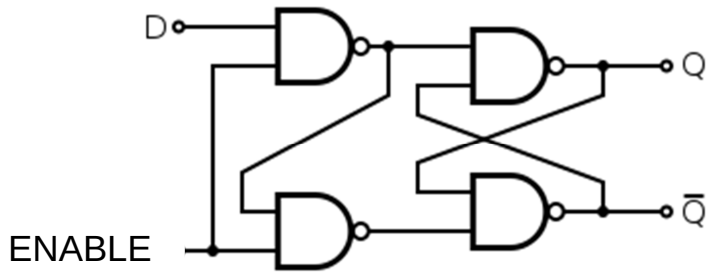


Fig 1.1: Ideal D Latch with NAND gates

Exercise 2: Cheap memory (Design exercise – no need to make circuit)

Due to cost and power restrictions, suppose you are told that using four NAND gates to store a single data bit as in the D-Latch of Fig 1.1 is too much. Design a 1-bit clocked D-latch using just two NOT gates and a couple of switches or transistors. It should obey the same state table.
Hint: Recall your work with the ring oscillator in Assignment #1.
Draw your design here:

Exercise 3: Putting memory to work

Name: Roll Number:

So far, we have worked with latches which are sensitive to CLK *levels*. As you have seen, this can be problematic. In a complex circuit with many delays and glitches, logic levels may change while the CLK pulse is high and circuit operation may become faulty. Registers and Flip-flops avoid this problem by being sensitive to inputs only at the rising (or falling) edge of the CLK input. A D *register* is the edge sensitive equivalent of a D *latch* used in Exercise 1. See page 4 for an explanation of the difference between a latch, register and flip-flop

Experiment:

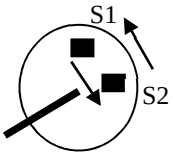
Suppose you have two digital signals with an arbitrary phase difference between them. You don't care about the magnitude of the phase difference, just the sign. A typical example in a real experiment would be a motor driving a disc. There are two sensors attached to the disc's surface. As the motor rotates, a fixed detector gets a signal from each sensor with a phase difference. You need to determine whether the motor is spinning clockwise or counter-clockwise. The objective of this exercise is to implement such a phase detector using the D-register.

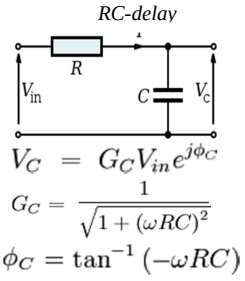
Procedure:

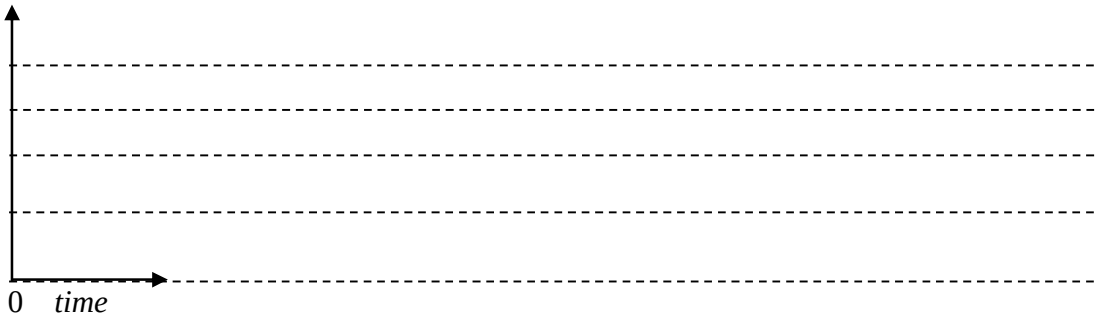
Use the logic pulse generator to generate a square wave. Make a copy of the wave and add a phase delay to it using an RC delay filter. Use sensible values for the frequency and pick appropriate R & C values to get a reasonable time constant. (Remember! You are constrained by TTL Logic – the delayed signal must still follow TTL voltage constraints). You now have two signals S1 and S2 with a phase lag between them. Use the D-register (IC 74LS175) to design a phase detector that can determine which wave has a more advanced phase than the other. The output Q of the register must encode the answer – i.e. it must be 1 if $\Delta\varphi$(S1- S2) is positive, and vice versa. You can demonstrate this by exchanging the input wires in your circuit.

Draw diagram of your phase detector design and explain its timing diagram below:

↓

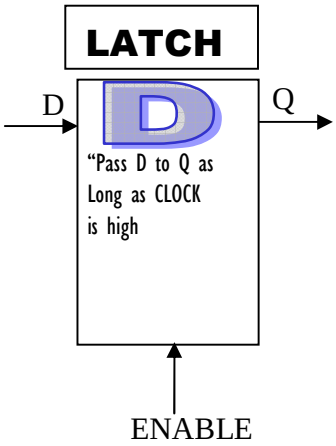






Name: ***Roll Number:***

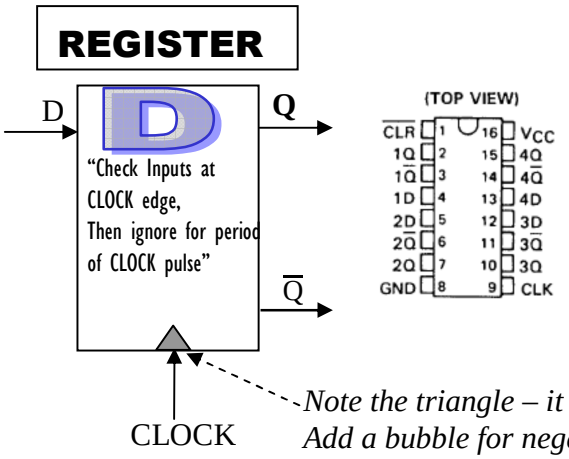
Reference -- Fig 2.2: Reference - State Tables in "plain English"



D Latch		
D	**ENABLE**	**What happens to Q**
0	LOW	Q remains unchanged
1	LOW	Q remains unchanged
0 *	HIGH	0 *
1	HIGH	1 *

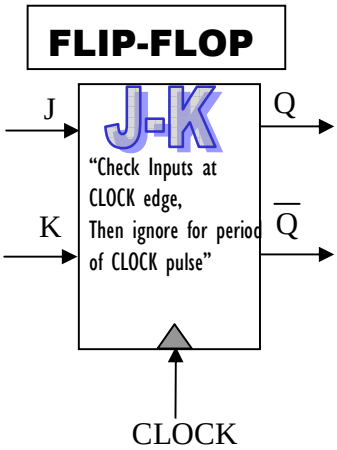
-

Note that this is a 'transparent' latch: if the input switches state while the ENABLE is high, the output will change state too.



D Register (74LS175)		
D	**CLK**	**What happens to Q**
0	0	Q remains unchanged
1	1	Q remains unchanged
0	edge	0
1	edge	1

Note the triangle – it denotes positive edge sensitive clock. Add a bubble for negative edge sensitive.



J-K flip-flop		
J	**K**	**What happens to Q and $\overline{Q}$**
0	0	Q , $\overline{Q}$ next CLK edge leaves output unchanged
0	1	0. 1 next CLK edge resets Q to 0
1	0	1. 0 next CLK edge sets Q to 1
1	1	$\overline{Q}$,Q , next CLK edge toggles output