Shift registers

This is designed to be a hardware assignment which you should be able to complete providing you are current with the module material. This is an opportunity for you to demonstrate your understanding of the theory discussed in the modules and gain more experience of seeing theory work in simulation.

Prerequisites

You need a basic understanding of the operation of a shift register, Module 10.

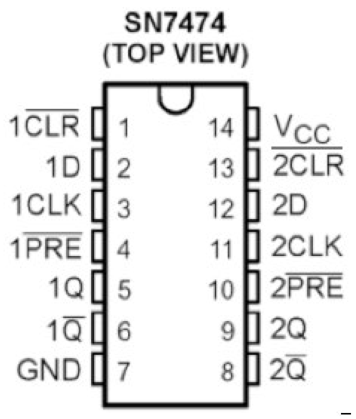
Objectives

Today's exercise is to use four D flip flops to build a register to show that serial and parallel input and output can be achieved through the D preset and clear inputs. You will also learn about the construction of debounce circuits.

Theory

This lab is full of important information that you will need to learn now and use for the remainder of the semester. Work slowly and make sure you have a firm grasp of the differences between parallel and serial input and output. Also, you should understand that the shift register you will be building will be able to do **ALL** of these functions. If you purchase an MSI shift register, you will need to select one based on the specific mode you will need.

In general, a shift register may be composed of two or more D-Flip-Flops (D-FFs). These devices are assembled with a common clock so that the clock pulse causes each bit of data in the D-FFs to move at once through each D-FF. The D-FFs you will be using are the 74x74 which allow data to move on the rising edge of a clock pulse. This means data can only move during the transition period from 0 to 1. It does not matter if the pulse remains at 1 or drops back to 0 - no data will move until the next rising edge. This is called positive edge triggering – remember? Other methods of passing data through a shift register are available but for this lab, we will *only* be working with these components. It will also be assumed that all of our theoretical shift registers are 4 D-FFs because that is what you will be constructing today.



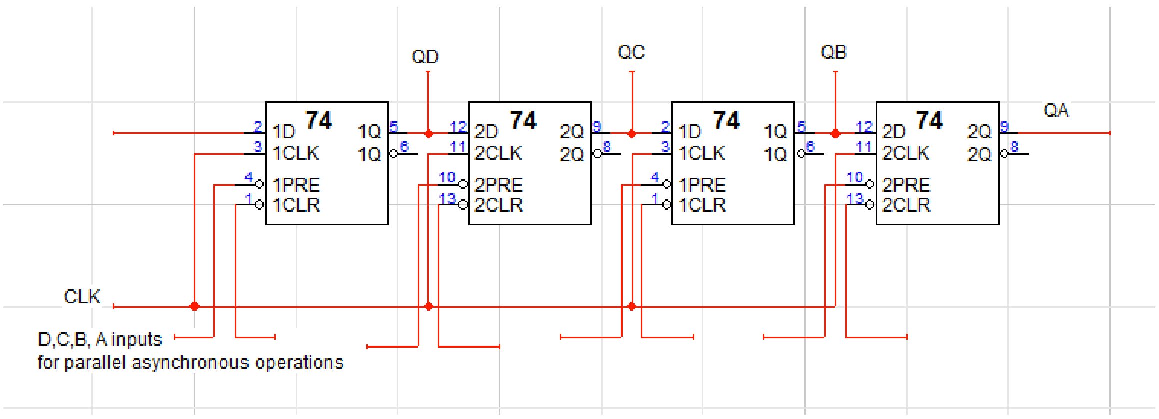
FUNCTION TABLE

INPUTS				OUTPUTS	
$\overline{PRE}$	$\overline{CLR}$	CLK	D	Q	$\overline{Q}$
L	H	X	X	H	L
H	L	X	X	L	H
L	L	X	X	H†	H†
H	H	↑	H	H	L
H	H	↑	L	L	H
H	H	L	X	Q_0	$\overline{Q}_0$

There are four ways you can use your shift register:
Serial in - serial out; serial in-parallel out; parallel in- serial out; parallel in-parallel out

Procedure

To build the 4-bit Register connect four D-Flip-Flops together as shown below. Use a common clock input. This provides the synchronous operation.



The normal value on the transfer/clock pulse line is low (ground). The transfer pulse can be obtained by connecting this line to Vcc.
D, C, B, and A are your parallel input values that are allowed to set the respective Q outputs asynchronously through the PRE and CLR independent inputs. QD, QC, QB, and QA are your parallel outputs. Connect the Q outputs to four LEDs to be able to read the contents of the register all the time..

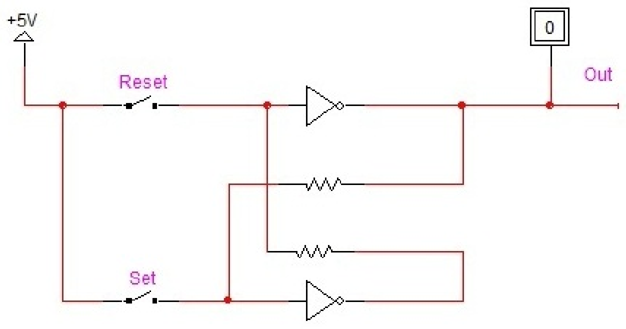
Debounce circuit

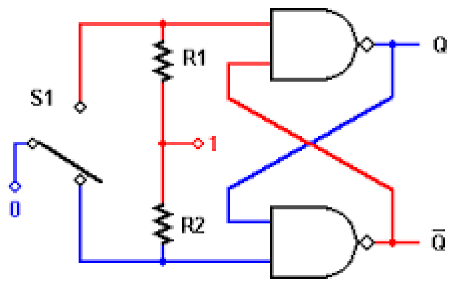
Debouncing is a way to prevent spikes in the output of electronic/electrical devices having switches. When we press any switch manually and release it, it bounces due to inherent elasticity. Due to this there is a multiple make and break of electrical contact. A large time response does not create much of a problem but the small ones can get multiple responses for a single key press.

A generic circuit used for the debouncer circuit is the following on the left. The normal value on the transfer/clock pulse line is low (ground). The transfer pulse can be obtained by connecting this line to Vcc. The circuit acts like a latch and when both the switches are open, it retains the last value. Once you close a switch, it SETS or RESETS depending on the switch. You can then open the switch again.

The debouncer circuit operates by touching the input of the upper Reset inverter to Vcc to make the output (Out) equal to 0 and touching the input of the lower Set inverter to Vcc to make the output (Out) equal to 1. When resetting the circuit's output to 0, the input of the Set inverter is left open (i.e., not connected), and when setting the circuit's output to 1, the input of the Reset inverter is left open (i.e., not connected)."

The point of having a debouncer circuit is to maintain a solid 1 or 0, thereby countering any noise.





Similar debounce circuits exist; with NANDs for example (diagram above on right). The resistor values are not critical but each should have the same resistance value. You will have to build the debouncer for your circuit. We recommend using the NAND implementation on the right.

Testing

Having set up your circuit as shown, you should perform the following operations:

1. Serially input 1, 0, 1, o (result is DCBA = 0101).
2. Clear everything, then input DCBA = 1011 using SET operations.
3. Clear the registers buy inputting sequential zeroes.
4. Using serial inputs, make DCBA = 0001, then use a SET operation to change D to 1.

Deliverables

Complete the required operations and show them to your TA for assessment. You are required to complete a report of this assignment as performed.

Imagine an 8-bit register. Take the last two digits of your PeopleSoft number and convert them both to Binary Coded Decimal (BCD). For example, my last two digits are 39, so in BCD form these are 0011 and 1001. Concatenate these two values, then input them to the imaginary shift register from LSB to MSB (recall that inputs start at the MSB and are shifted right with each clock in this lab's implementation).

My 8-bit number is 00111001, so I would start by inputting 1, then 0, then 0, and 1, and so on. Assuming that the shift register starts out cleared, create a table that indicates the parallel output at each step, and find the decimal representation of that 8-bit sequence. Example below…

STEP	INPUT	REGISTERS	DECIMAL
0	N/A	00000000	0
1	1	10000000	128
2	0	01000000	64
3	0	00100000	32
4	1	10010000	144
ETC.			

Questions

What are the main functional purposes of a register?
The chip photo looks like a SN5474. Is this the same as a SN7474?

Follow-up

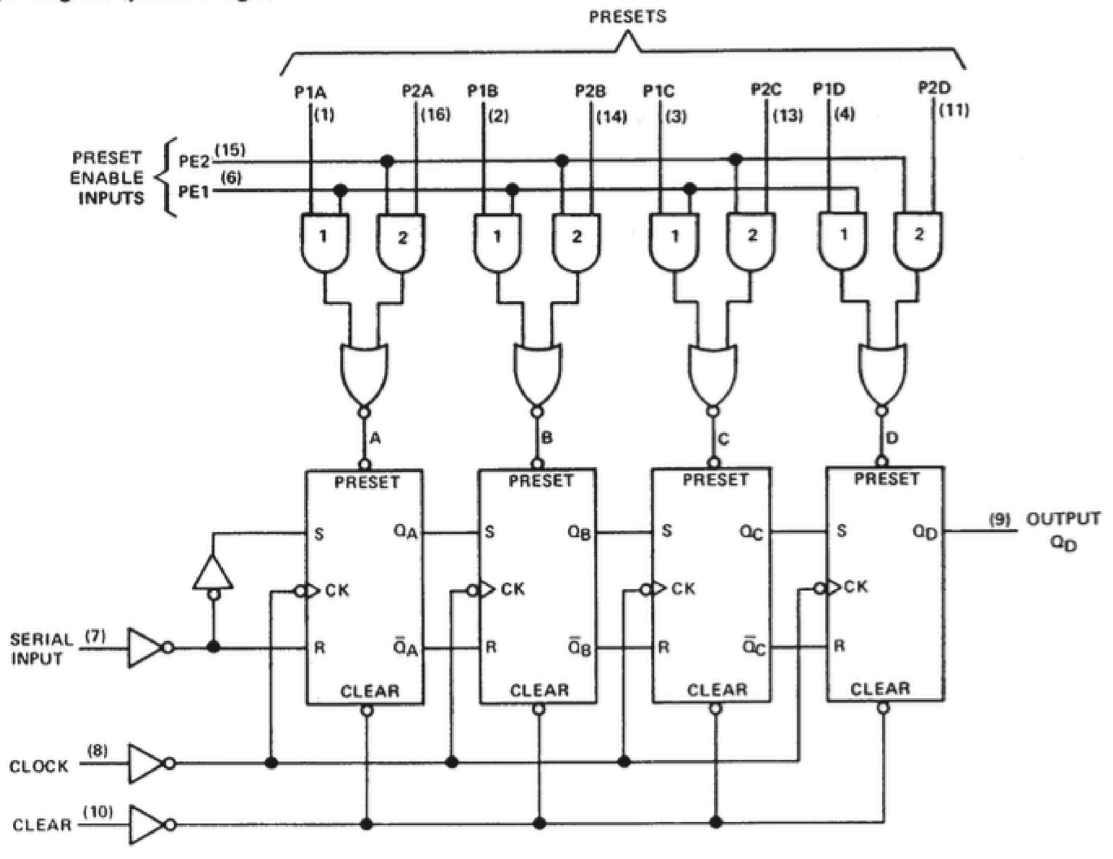
The details below are for a SN7494 chip. To fully understand what you have done in the lab, please read through the description below and study the diagram on the following page. You should be able to see how versatile these types of devices are and how they play such a large part in the arithmetic and logical operations of a computer. You have enough digital logic background to be able to analyze the circuit.

Description of the SN7494

The '94 is a 4-bit register with serial and parallel (ones transfer) data entry. To facilitate parallel ones transfer from two sources, two parallel Load inputs ($PL_0$and PL_1) with associated Parallel Data inputs (D_{0a}-D_{0d} and D_{1a}-D_{1d} are provided. To accommodate these extra inputs only the output of the last stage is available. The asynchronous Master Reset (MR) is active HIGH. When MR is HIGH, it overrides the clock and clears the register, forcing Q_d LOW.

Four flip-flops are connected so that shifting is synchronous: they change state when the clock goes from LOW-to-HIGH. Data is accepted at the serial D_S input prior to this clock transition. Two Parallel Load inputs and Parallel Data inputs allow an asynchronous ones transfer from two sources. The flip-flops can be set independently to the HIGH state when the appropriate Parallel input is activated. Parallel inputs D_{0a} through D_{0d} are activated during the time the PL_0 is HIGH and parallel inputs D_{1a} through D_{1d} are activated when PL_1 is HIGH. If both sets of inputs are activated, a HIGH on either input will set the flip-flops to a HIGH. The register should not be clocked while the Parallel Load inputs are activated. The Parallel Load and Parallel Date inputs will override the MR if both are activated simultaneously. However, for predicable operation, both signals should not be activated simultaneously.

logic diagram (positive logic)


PRESETS
P1A (1)
P2A (16)
P1B (2)
P2B (14)
P1C (3)
P2C (13)
P1D (4)
P2D (11)
PRESET ENABLE INPUTS
PE2 (15)
PE1 (6)
1
2
A
B
C
D
PRESET
S
CK
R
CLEAR
Q_A
$\bar{Q}_A$
Q_B
$\bar{Q}_B$
Q_C
$\bar{Q}_C$
Q_D
(9) OUTPUT Q_D
SERIAL INPUT (7)
CLOCK (8)
CLEAR (10)