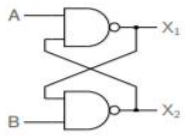
Date:	**29/5/2020**	**Name:**	**SAFIYA BANU**
Course:	**LOGIC DESIGN**	**USN:**	**4AL16EC061**
Topic:	Analysis of clocked sequential circuits Digital clock design	**Semester & Section:**	**8th , B**
Github Repository:	**Safiya-Courses**		

REPORT

NAND latch (RS latch)

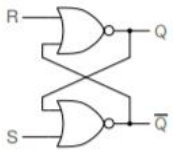


A	B	X_1	X_2
1	0	0	1
0	1	1	0
1	1	previous	
0	0	1	1

* A, B: inputs, X_1, X_2: outputs
* Consider $A=1$, $B=0$.
  $B=0 \Rightarrow X_2=1 \Rightarrow X_1=\overline{A\,X_2}=\overline{1\cdot 1}=0$.
  Overall, we have $X_1=0$, $X_2=1$.
* Consider $A=0$, $B=1$.
  $\rightarrow X_1=1$, $X_2=0$.
* Consider $A=B=1$.
  $X_1=\overline{A\,X_2}=\overline{X_2}$, $X_2=\overline{B\,X_1}=\overline{X_1} \Rightarrow \boxed{X_1=\overline{X_2}}$
  If $X_1=1$, $X_2=0$ previously, the circuit continues to "hold" that state.
  Similarly, if $X_1=0$, $X_2=1$ previously, the circuit continues to "hold" that state.
  The circuit has "latched in" the previous state.
* For $A=B=0$, X_1 and X_2 are both 1. This combination of A and B is *not* allowed for reasons that will become clear later.

M. B. Patil, IIT I

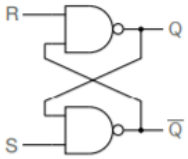
NOR latch (RS latch)



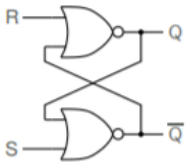
R	S	Q	$\overline{Q}$
1	0	0	1
0	1	1	0
0	0	previous	
1	1	invalid	

- The NOR latch is similar to the NAND latch:
  When $R=1$, $S=0$, the latch gets *reset* to $Q=0$.
  When $R=0$, $S=1$, the latch gets *set* to $Q=1$.
- For $R=S=0$, the latch retains its previous state (i.e., the previous values of Q and $\overline{Q}$).
- $R=S=1$ is not allowed for reasons similar to those discussed in the context of the NAND latch.

Comparison of NAND and NOR latches

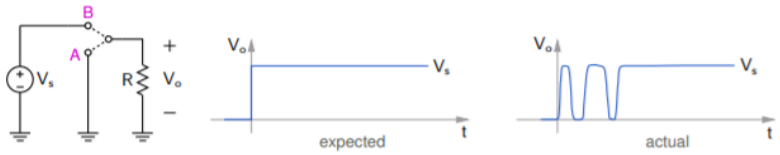


R	S	Q	$\overline{Q}$
1	0	0	1
0	1	1	0
1	1	previous	
0	0	invalid	



R	S	Q	$\overline{Q}$
1	0	0	1
0	1	1	0
0	0	previous	
1	1	invalid	

Chatter (bouncing) due to a mechanical switch



- When the switch is thrown from A to B, V_o is expected to go from 0 V to V_s (say, 5 V).
- However, mechanical switches suffer from "chatter" or "bouncing," i.e., the transition from A to B is not a single, clean one. As a result, V_o oscillates between 0 V and 5 V before settling to its final value (5 V).
- In some applications, this chatter can cause malfunction → need a way to remove the chatter.

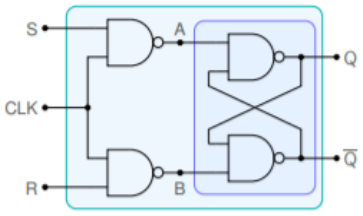
CLOCK

Complex digital circuits are generally designed for synchronous operation, i.e., transitions in the various signals are synchronised with the clock.

Synchronous circuits are easier to design and troubleshoot because the voltages at the nodes (both output nodes and internal nodes) can change only at specific times.

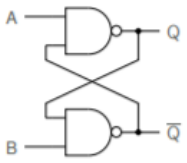
The clock frequency determines the overall speed of the circuit.
For example, a processor that operates with a 1 GHz clock is 10 times faster than one that operates with a 100 MHz clock. Intel 80286 (IBM PC-AT): 6 MHz Modern CPU chips: 2 to 3 GHz.

Clocked RS latch



Clocked RS latch

CLK	R	S	Q	$\overline{Q}$
0	X	X	previous	
1	1	0	0	1
1	0	1	1	0
1	0	0	previous	
1	1	1	invalid	

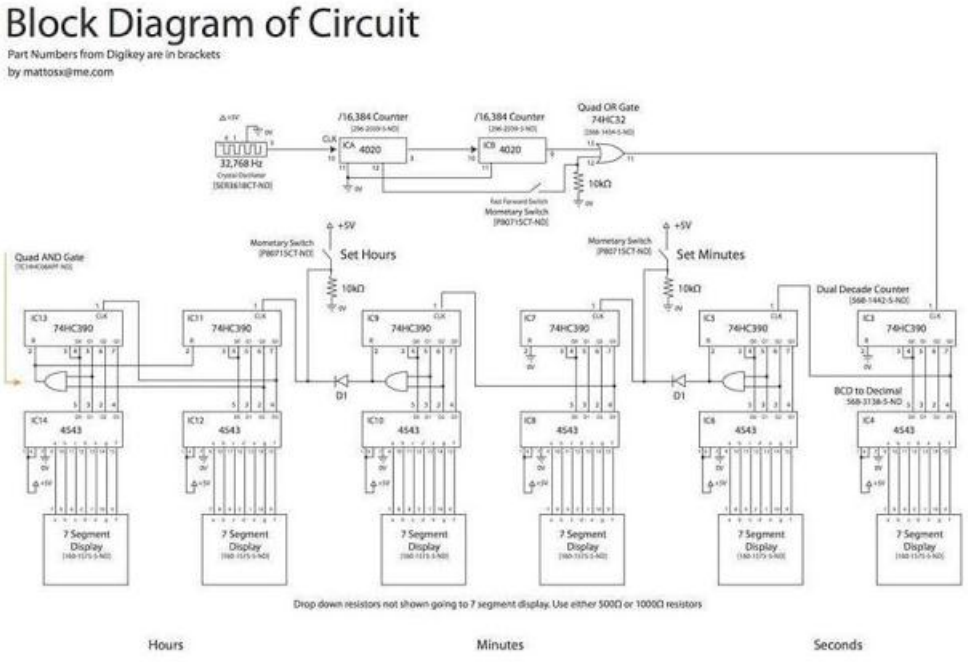


NAND RS latch

A	B	Q	$\overline{Q}$
1	0	0	1
0	1	1	0
1	1	previous	
0	0	invalid	

* When clock is inactive (0), $A = B = 1$, and the latch holds the previous state.
* When clock is active (1), $A = \overline{S}$, $B = \overline{R}$. Using the truth table for the NAND RS latch (right), we can construct the truth table for the clocked RS latch.
* Note that the above table is sensitive to the *level* of the clock (i.e., whether CLK is 0 or 1).

Digital clock design



The main parts of the circuit are as follows:

1- **Timer 555**: Responsible for generating the clock pulses for the counters, the frequency of the output shoul be 1 hz which means 1 second for each pulse.

2- **Counters**: Responsible for generating the time in BCD (Binary Coded decimal).

3- **Decoders** : Takes the BCD of the counter as input and produces 7 segment output .

4- 7 **segments** : Displays the time, of course!

* Seconds have 2 displays , 2 decoders and 2 counters. The same for minutes and hours.

The circuit works as follows :

555 timer produces 1 second pulses to the clock input of the first counter which is responsible the first column of seconds, so its output will change every second.

The counter produces numbers from 0 to 9 in BCD form and automatically resets to 0 after that. So the output of the first counter will count from 0 to 9 every second.

How can this be done

From the BCD code above (6: 0110) when the output is 6 the two middle bits are 1 (Q1,Q2),
So By ANDing these two bits the output will be 1.

This output will be connected to the reset pin of the same counter (2nd one) and the clock input of the next one(3rd).

When the output is 6 the AND gate output (1) will reset the same counter and its outputs goes 0000 so the output of the and gate again goes to 0 (1---->0), that will clock the next counter.