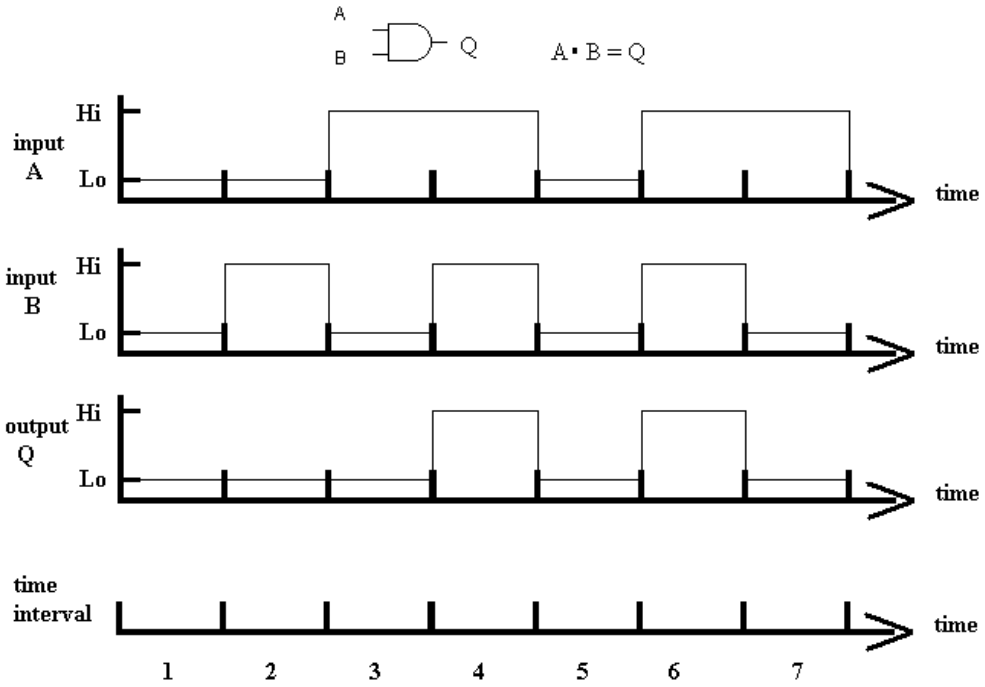
Physics 214 **INTRODUCTION TO LABORATORY ELECTRONICS** **Lecture 16**

Topics: Sequential logic and Flip-flops

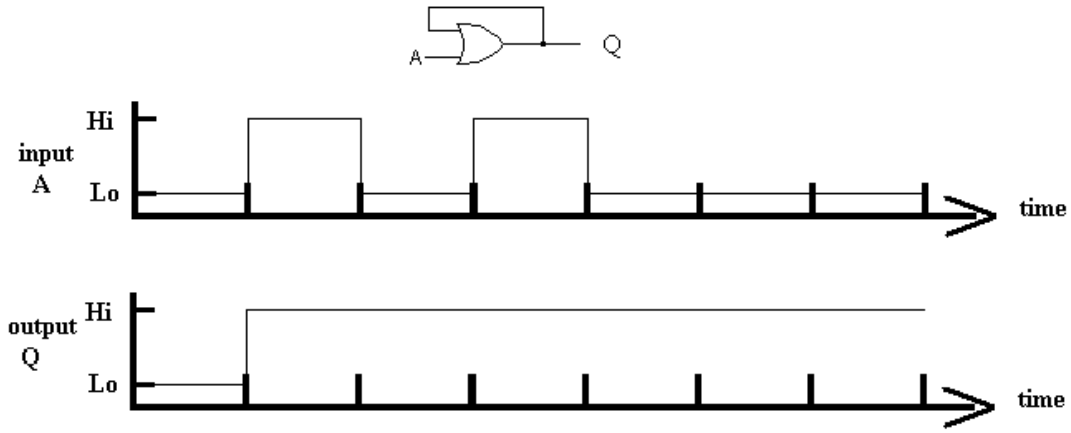
Text: Bugg, 13.1-13.4

In sequential logic, the output depends on the past history of the input. Timing becomes a critical component of the system operation and therefore the sequence of events is important.

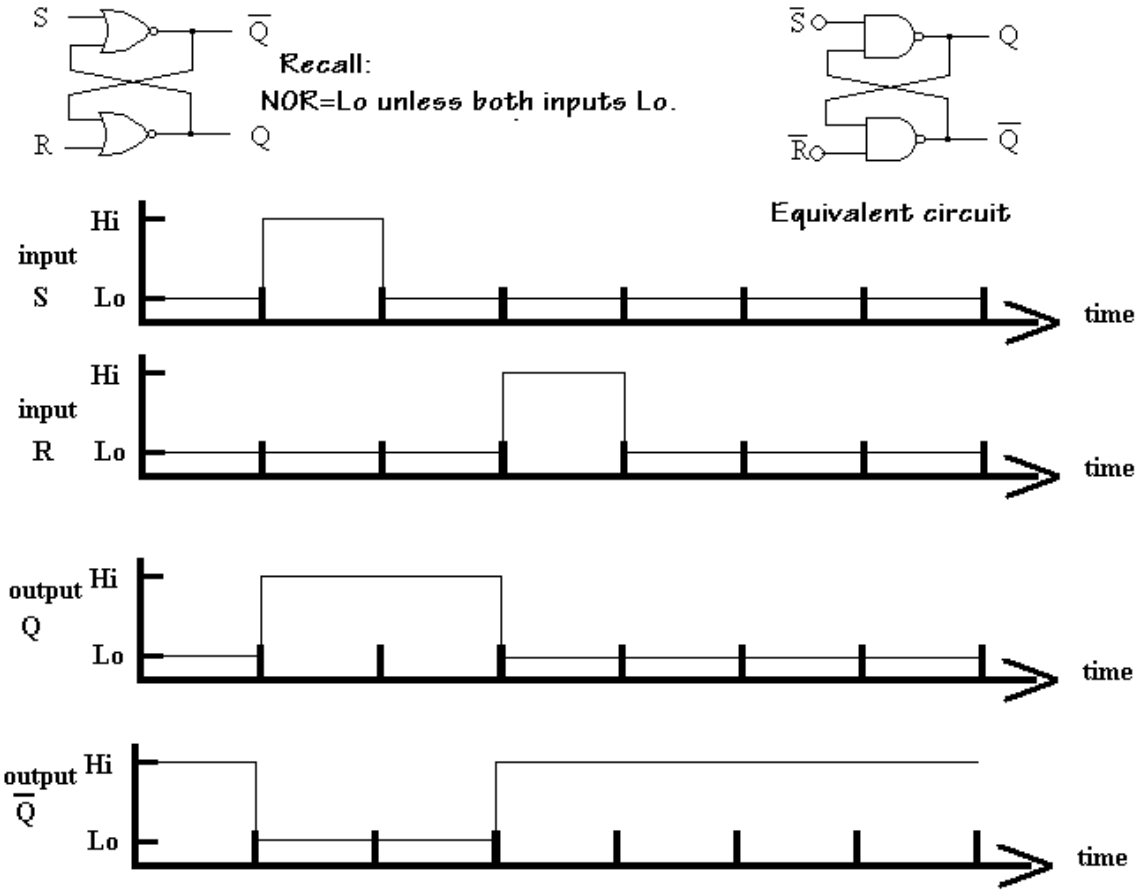
Event sequences are represented using ***timing diagrams*** such as the one presented here, which shows how the output of an AND gate follows the two inputs.



The whole idea of sequential logic implies that we need some way of 'remembering' the passed state. The most trivial is the one-bit register (also called a ***latch***) with no reset, depicted below. This is the simplest memory element.



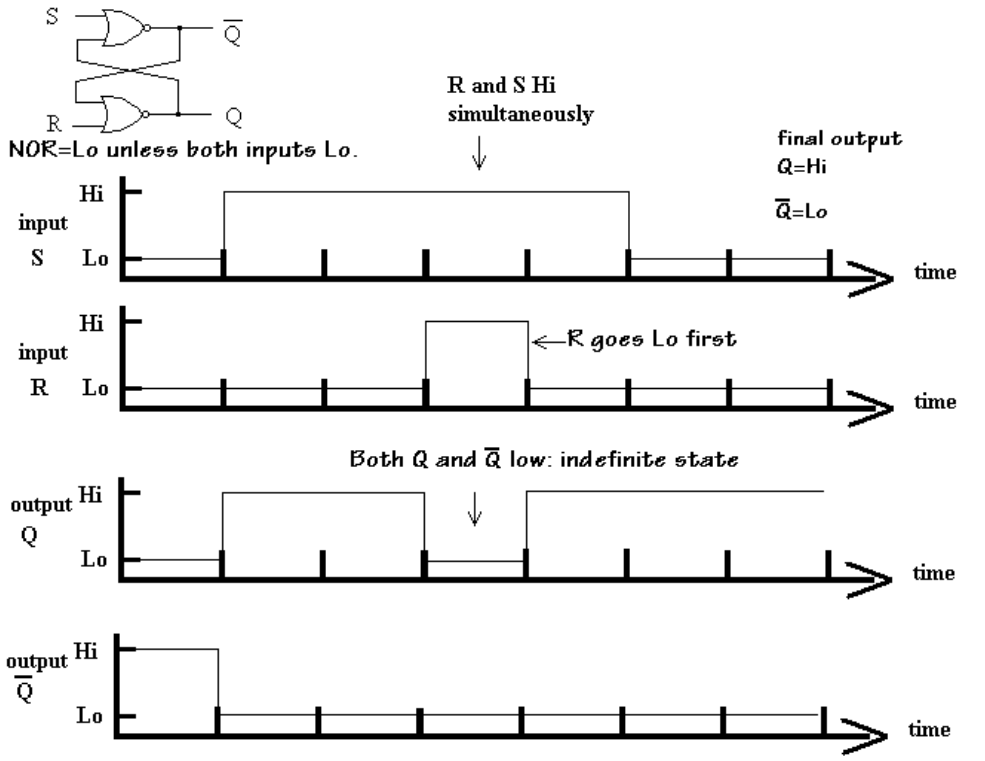
Without the ability to reset it, the latch has limited usefulness. A simple enhancement of the circuit yields the desired result: a resettable register (or latch) looks like:

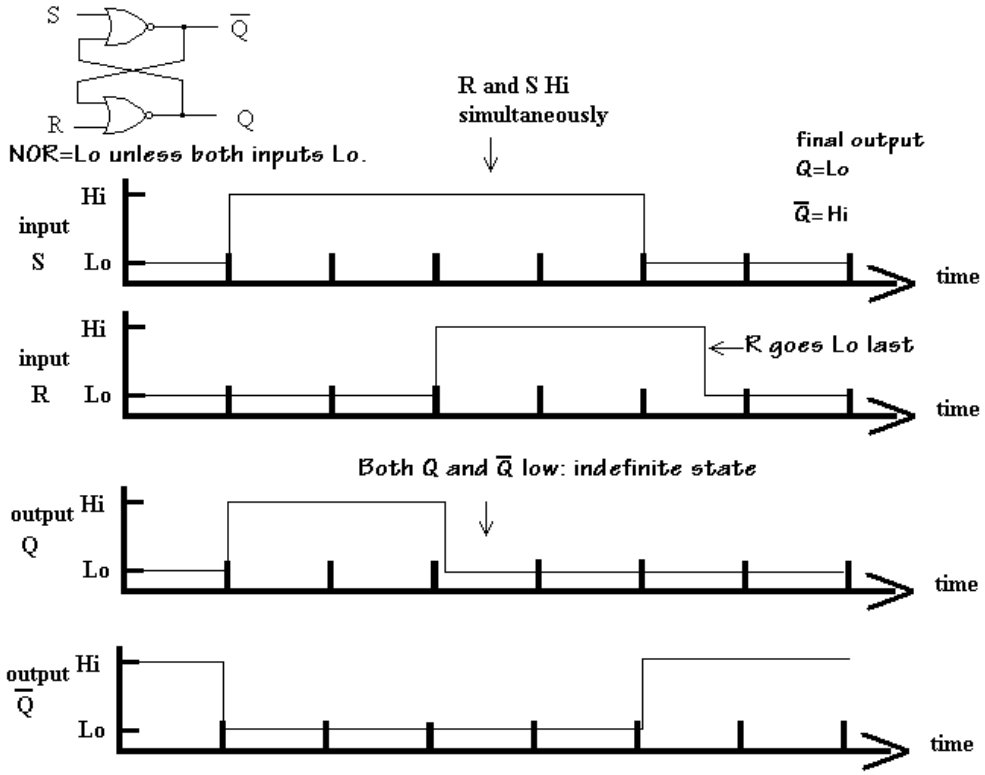


also known as a ***flip-flop*** or bistable multivibrator. More specifically, this is a RS flip-flop. (Reset-Set) The idea is that the latch is set when S is Hi and is reset when R is Hi. The latch is set when Q is Hi and $\overline{Q}$ is Lo. The outputs, Q and $\overline{Q}$ are to be complements of each other. When Q is Hi $\overline{Q}$ is Lo and when Q is Lo, $\overline{Q}$ is Hi. Both are not supposed to be Hi or Lo at the same time. Once set, (i.e. once S goes Hi), the Q stays Hi, even when S goes Lo. The way to reset the latch is to cause R to go Hi (i.e. the Reset is set Hi.)
When that happens, Q goes Lo and $\overline{Q}$ goes Hi.
R can then go Lo again and the latch stays reset.

This is all fine and dandy as long as you never have R and S Hi at the same time. Presumably you don't want to simultaneously Set and Reset the latch at the same time, but this can unintentionally happen. If that is the case, then the output passes through an ***indefinte, undefined,*** or ***indeterminate*** state in which both Q and $\overline{Q}$ are simultaneously Hi or both Lo.
If it happens that the latch passes through an undefined state, then after the Set and Reset both go low, the final state will be defined, but whether Q is Hi or

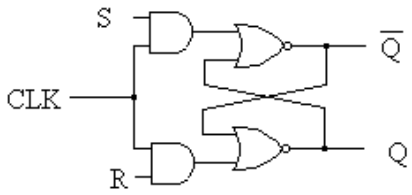
Lo will depend on which input, R or S, went Lo first.

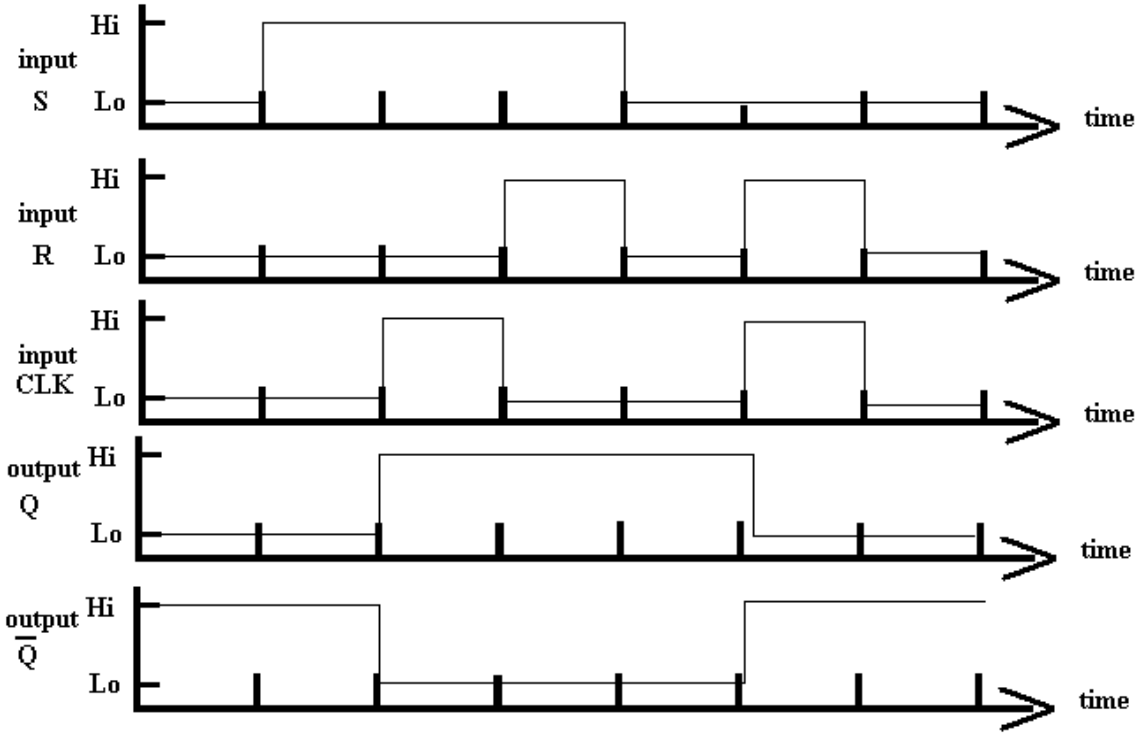




This is clearly not good. There are several ways of dealing with this problem. A simple way is to only allow a Set or Reset when a third signal allows it.

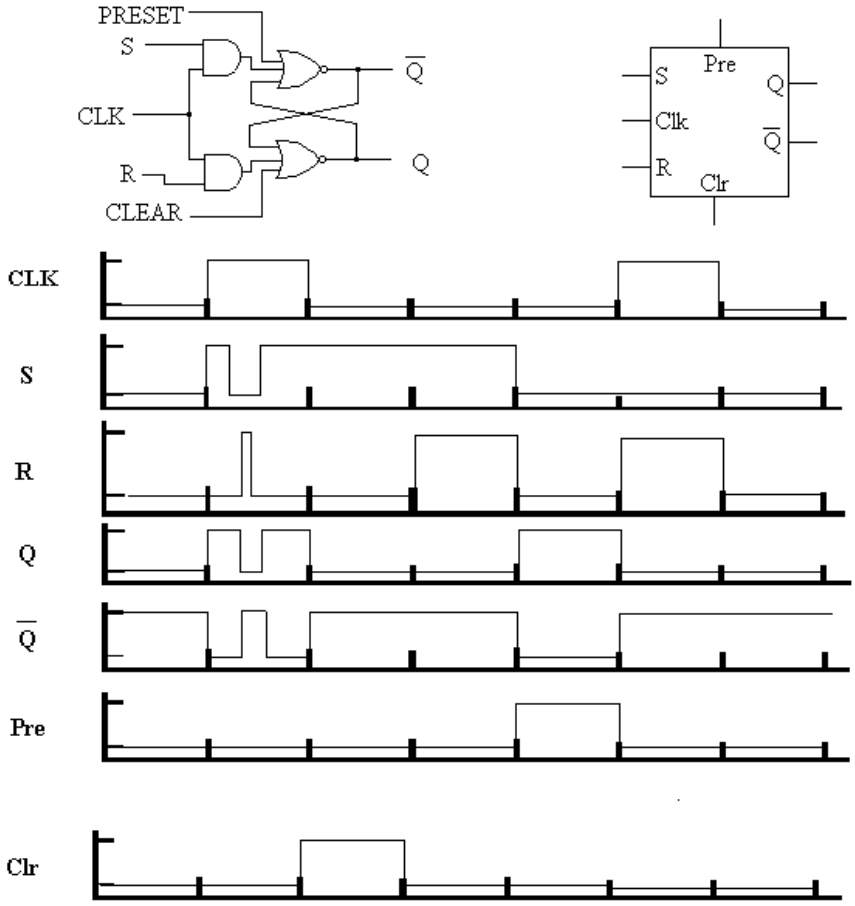
This third control signal is called the ***clock***. State transitions (i.e. Set or Reset) are only permitted when the clock signal is Hi. Such a device is called a ***Clocked Flip-flop*** or ***Clocked Latch***.



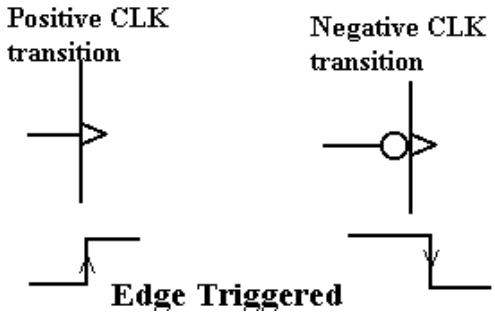


When the CLOCK is Lo, the data is said to be ***latched*** which means the system is disabled for new data. We can only 'write into memory' when the CLOCK is Hi.

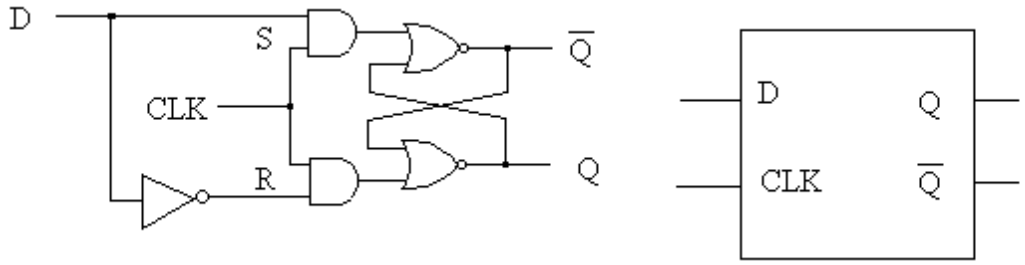
In the above simple clocked RS latch, S and R must be synchronous with the CLK. Sometimes there is a need to Set or Reset independent of the CLK. i.e. **asynchronously**. In such cases, we use an asynchronous set, called a ***PRESET*** and an asynchronous reset called a ***CLEAR***. The circuit may look like:



This circuit still has the undesirable characteristic that the output changes in response to the input when the clock is Hi. Such devices are ***called transparent latches*** or ***see through latches***. This can cause problems with ***logic race conditions*** whereby two states try to change simultaneously. This can be caused by switching delays, for example, which can yield input timing that is not precisely reproducible. The output then is not determined by the value of the input states at a given time as determined by the clock, but by what could be unpredictable and not necessarily reproducible input conditions arising from imperfectly synchronized signals. One partial solution to this is to only look at the inputs when the clock is in a transition, either going from Lo to Hi (positive transition) or from Hi to Lo (negative transition). These are known as ***edge triggered flip-flops*** and provide improved synchronization.

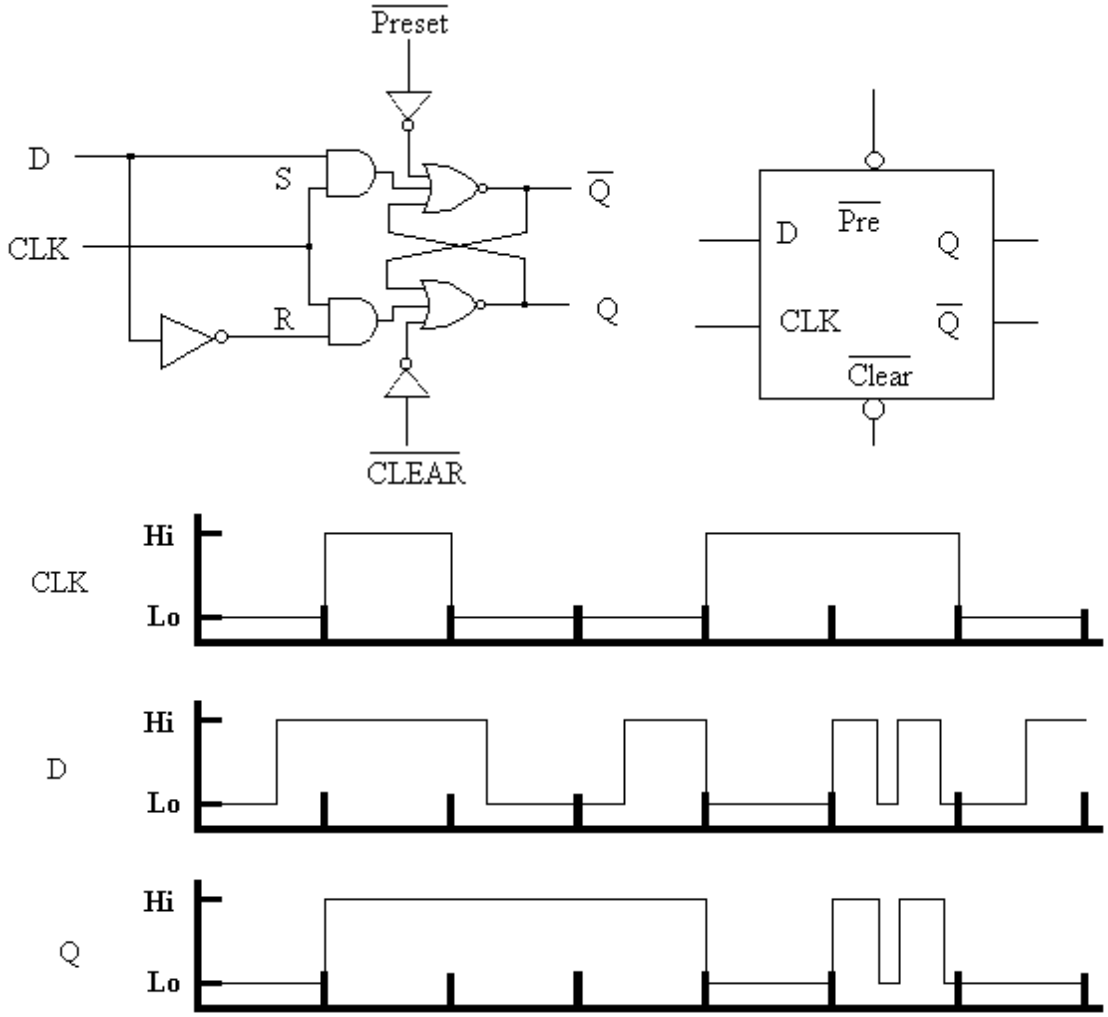


Once we have a clock, then there are many applications that don't need independent SET and RESET inputs. S and R can be set to be the complements of each other:

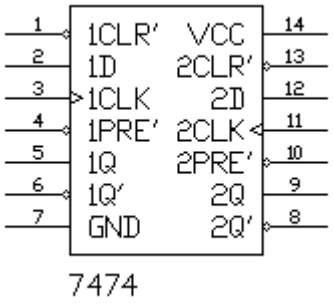


In this way the undefined states can be avoided altogether. Then there is only one input, S, which is renamed D, for data. When D is Hi when the CLK is Hi, then Q goes Hi. When D is Lo when the CLK is Hi, then Q goes Lo.

For many applications of memory, one wants the possibility of asynchronously setting or resetting the state of the device and therefore again introduces PRESET and CLEAR inputs. Often, one uses ***active low*** PRESET and CLEAR. That is, if the line is Hi, then nothing is done. Only when the line goes Lo will the memory be cleared or set.

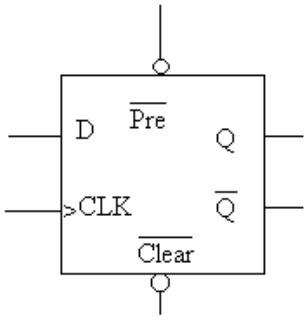


This is still a transparent latch. But, the logic race condition can be avoided again by using edge triggering, either rising or falling edge. In the lab, you'll be using a 7474 Dual D-type Flip-flop.i.e. Two FF in one IC package.

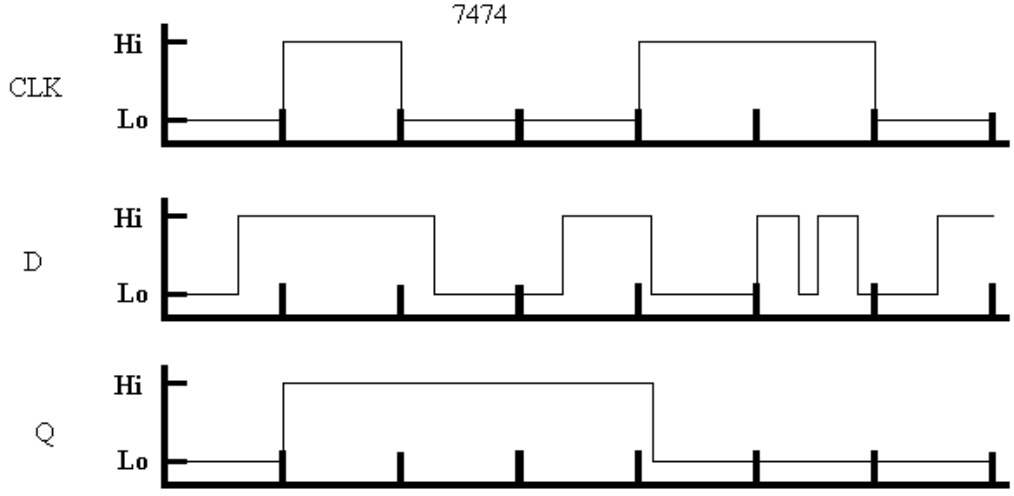


7474

$\overline{PRE}$	$\overline{CLR}$	Clk	D	Q	$\overline{Q}$
0	1	X	X	1	0
1	0	X	X	0	1
0	0	X	X	1	1
1	1	POS	1	1	0
1	1	POS	0	0	1
1	1	0	X	Hold	



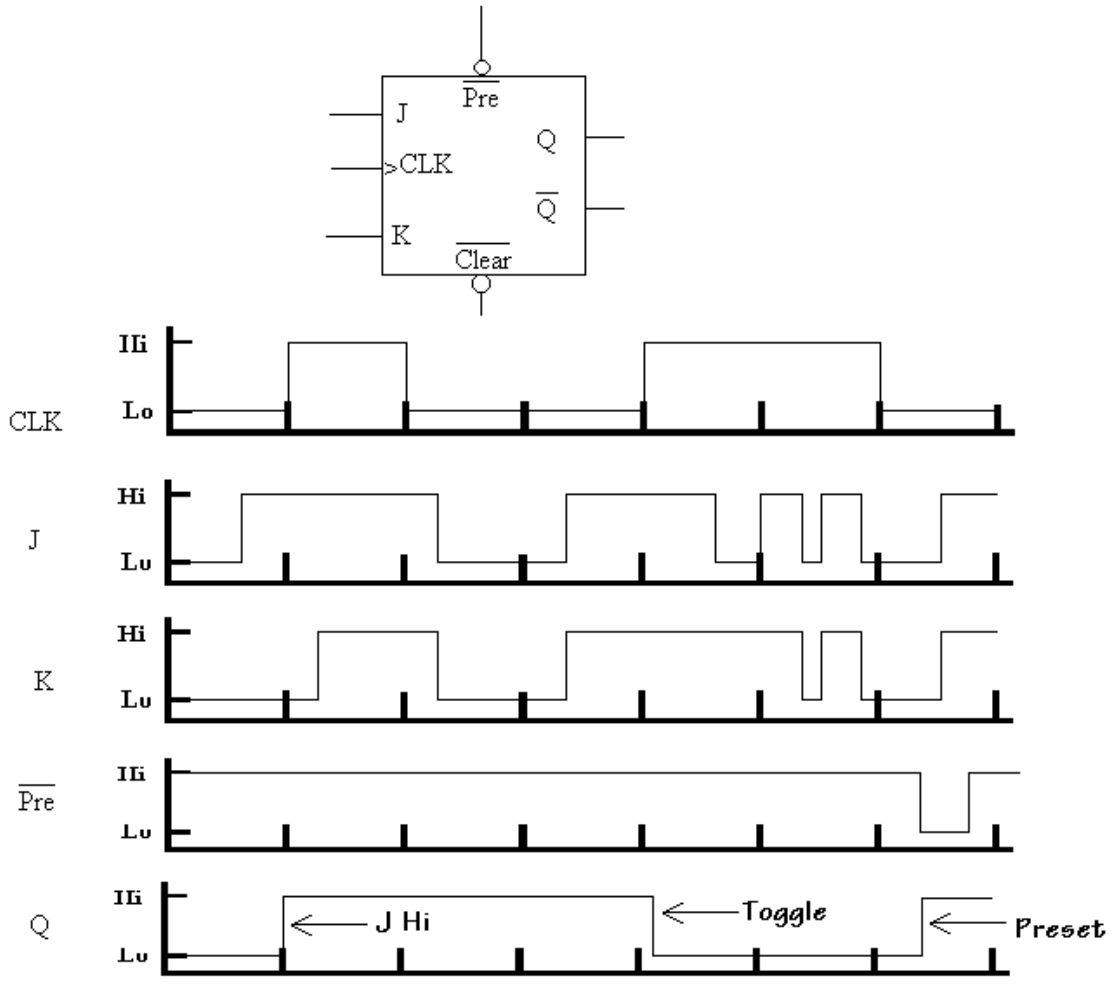
7474



The D-type flip-flop is one specialized configuration of the RS flip-flop designed to remove the undefined status. It is a standard memory element that can be written to synchronous with the clock or cleared/set asynchronously.

JK FLIP-FLOP

Another approach is to realize that the RS undefined state in which both S and R are Hi can be identified as a signal to do something useful. In the case of a JK flip-flop, everything is the same as with an edge-triggered RS flip-flop where J is the S and K is the R except that if both J and K are Hi, then the output Q value toggles from its previous state. i.e. if Q was Lo it goes Hi and if it was Hi it goes Lo. This only happens if both J and K inputs are Hi on the clock transition. For example, for a rising-edge triggered JK flip-flop, if J=K=Hi, and Q=Lo, then on the next Lo to Hi transition of the CLK, then Q=Hi.



In the lab you'll use a falling-edge CLK (7476 JK-FF) so that, unlike in the above example, the inputs are examined only when the CLK transition goes from Hi to Lo. It also has active low PRESET and CLEAR.