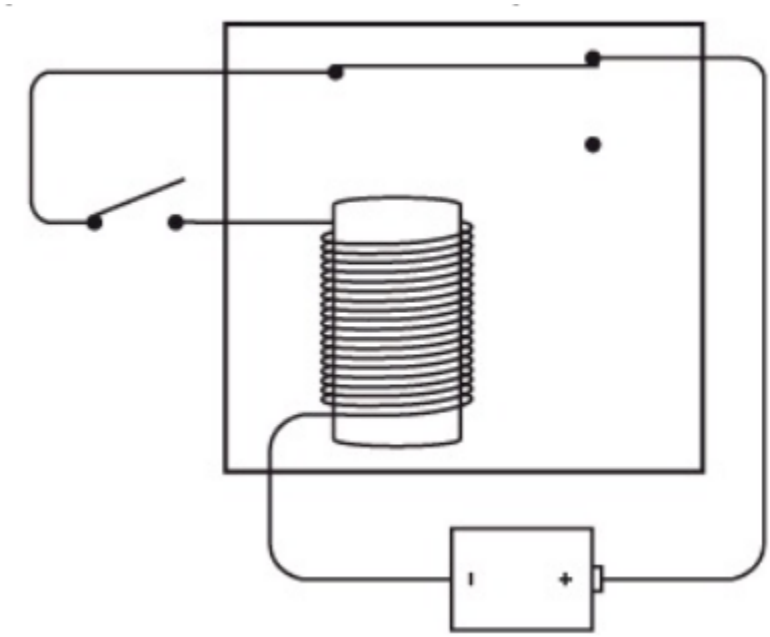
Test

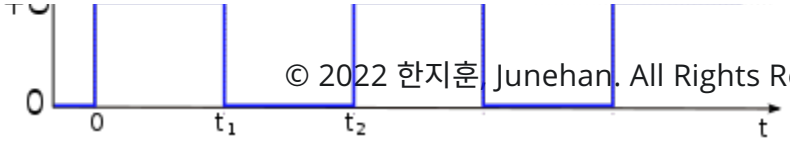
March 29, 2022 in category, subcategory

주기적인 진동신호를 발생시키는 원리



- 입력이 유지되는 상태에서 자성을 가지는 금속이 스위치를 해제하도록 구성합니다.
- 스위치가 해제되면서 전류가 흐르지 않게 되면 금속이 자성을 띄지 않게 되고 다시 위쪽의 스위치는 원래의 형태로 돌아가게 됩니다.
    - 모든 스위치가 연결되고, 물체가 위의 스위치를 끌어당길 수 있을만한 충분한 자성을 가지기전까지 걸리는 시간(1)
    - 전류가 흐르지 않게되면서 물체의 자성이 서서히 떨어져 스위치가 다시 연결되기 까지 걸리는 시간(1)
    - 위 두 시간이 동일하게 1:1을 비율을 가진다면
        - 전류가 흐름(1)
        - 전류가 흐르지 않음(0) 의 상태를 균일한 간격으로 표시하는 것이 가능하다고 합니다.

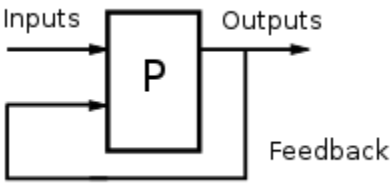
Clock signal



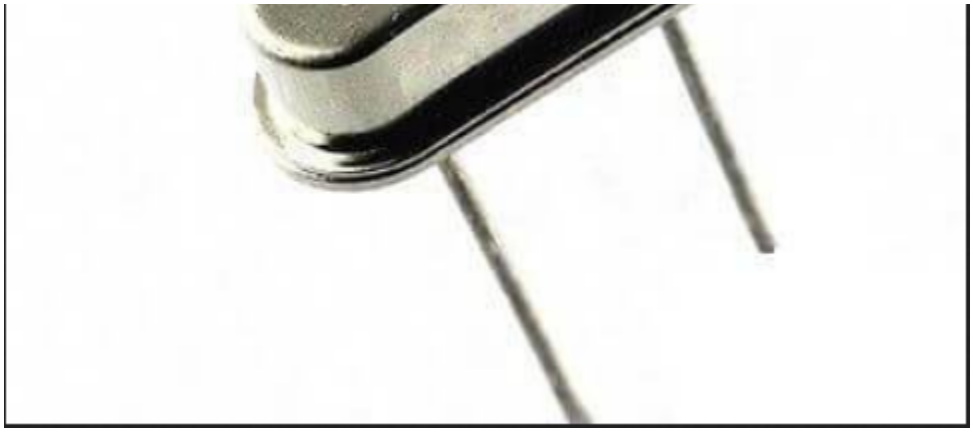
(*위와 같은 원리의 장치에 의해서 동일한 간격으로 0과 1의 신호를 표시하는 장치가 있다고 해봅시다.*)

1. Clock 이란?
    - 1 clock (0 -> 1)로 올라가는 엣지에서부터, 1과 0을 잠시 유지했다가 다시 (0 -> 1)로 전환되는 직전까지를 한 Unit으로 보고 이를 1싸이클, 1클럭이라고 부르고 있습니다.
    - 이 운동을 1 회의 진동으로 간주, 전기적 신호에 한정해 초당 얼마나 많은 싸이클(클럭)을 갖는지에 따라 hz의 단위를 붙여 표기하고 있습니다.

출력을 입력에 연결하는 피드백과 인버터 오실레이터(전기 진동자)



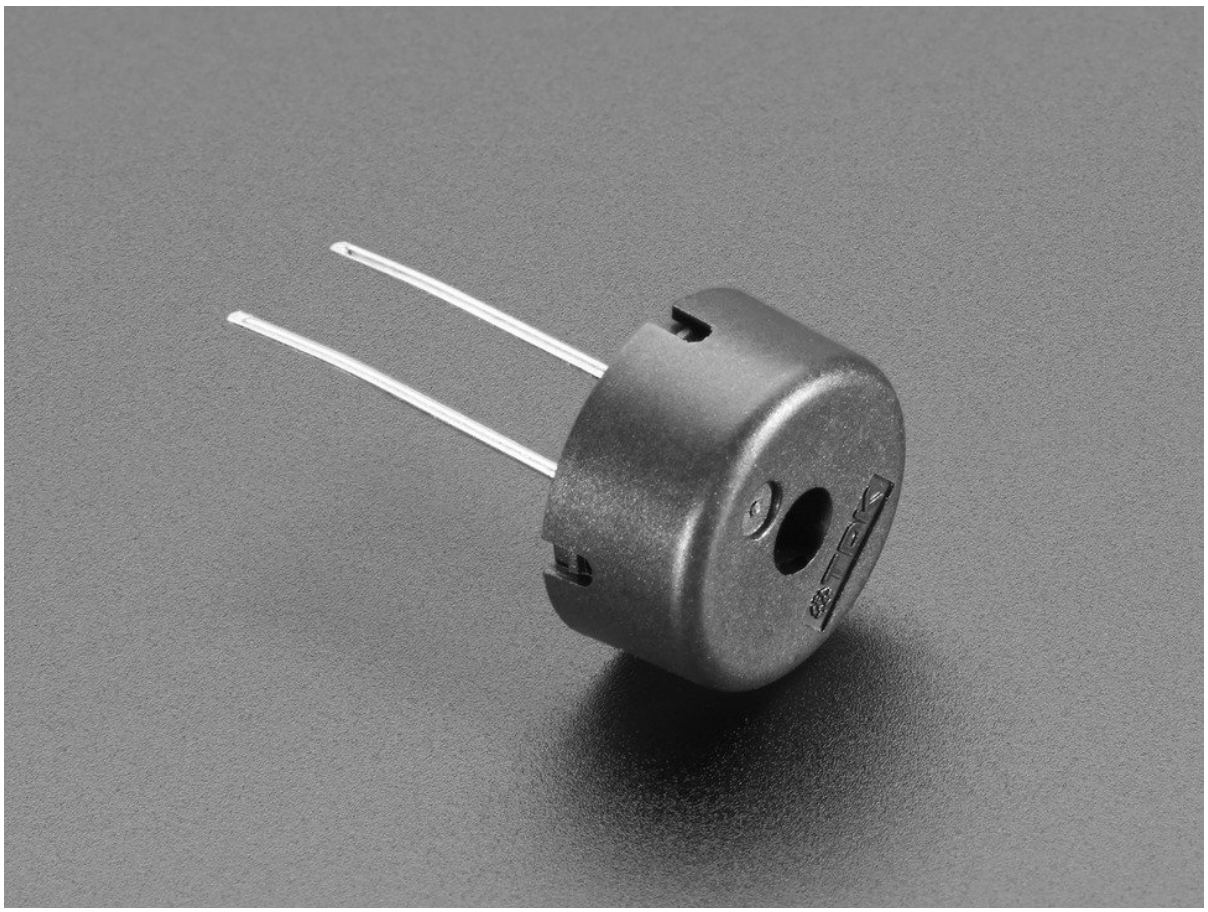
> - *피드백자체는 인버터를 전제하는 개념은 아닌 것으로 보입니다. 위의 릴레이와 동일한 방식으로 입력으로 인해 발생된 출력이 다시 입력으로 이어져 원인-결과의 효과를 연쇄적으로 발생시키는 회로 혹은 루프를 의미합니다.*
> - *마치 마이크를 스피커에 가져대는 것과 같은 개념을 feed-back(다시 먹이다)과 같은 표현으로 적용한 것 같습니다.*



> *Oscilate (동사, 진동하다)*
> *상용으로 사용되는 크리스털 오실레이터(전기 진동자)의 사진입니다. 위에 릴레이에서 표현되는 기능과 같은 기능을 수행하여 현대적인 대체품이라고 보시면 좋을 것 같습니다.*

> *초당 12 억회의 클럭신호를 발생시키는데 컴퓨터가 시간을 재는 단위로 사용할 수 있는 회로입니다. (디바이스별로 전기적 신호로 시간의 간격을 잴 수 있는 가장 작은 단위)*

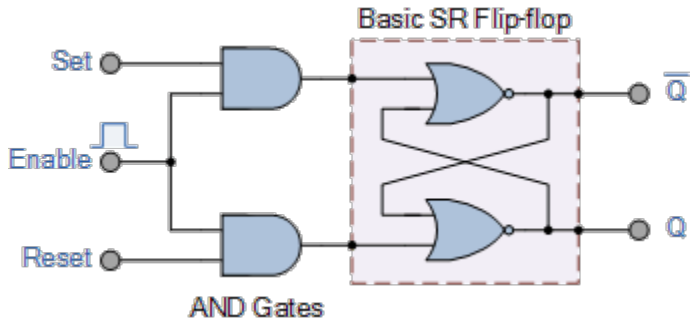
크리스탈에 전기를 가해서 음성진동을 만들어 내는 피에조 버저



> *부저 버저 등으로 부르는데 비프음을 발생시킵니다. 간혹 메모리불량 등 메인보드에서 부팅이 불가능한 상황을 감지하면 메인보드에서 부팅과정에 오류를 출력하면서 이 비프음이 연결된 선으로 전류를 흘리는데 삐- 하는 소리가 계속 나는데, 이어지는 소리가 아니었고 위에 나온 원리처럼 소리가 0과 1을 반복하는 진동을 가지고 나는 소리였다고 합니다.*

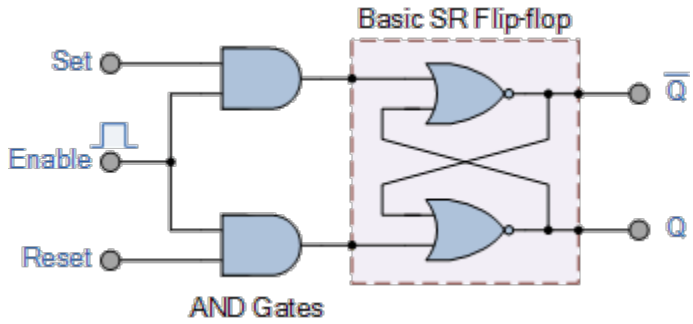
피드백으로 상태를 유지하는 플립플롭

- set-reset flipflop



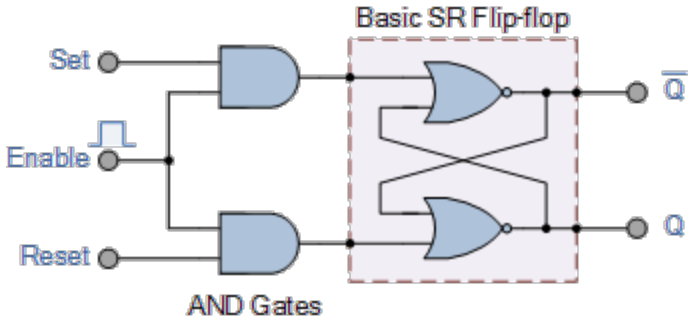
SET	RESET	ENABLE	AND(UP)	NOR(UP)	AND(DOWN)	NOR(DOWN)	Q	!Q
0	0	0	0	1(x)	0	1(x)	0	0
1	0	0	0	1(x)	0	1(x)	0	0

- 최초에는 두 NOR 게이트가 1이라는 출력이 서로의 입력으로 연결되기 때문에 둘 다 0입니다.
- NOR Down: AND DOWN + NOR UP
- NOR UP: AND UP + NOR DOWN



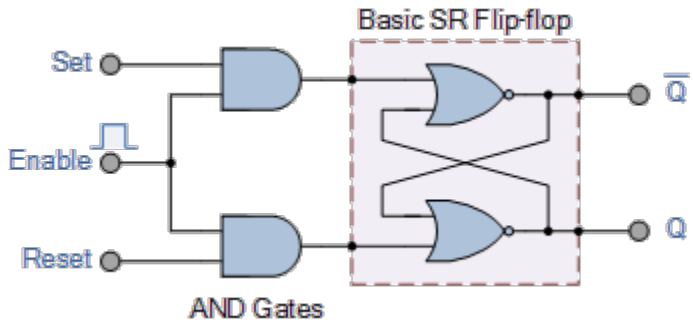
SET	RESET	ENABLE	AND(UP)	NOR(UP)	AND(DOWN)	NOR(DOWN)	Q	!Q
1	0	0	0	0	0	0	0	0
1	0	1	1	0	0	1	1	0

- set + enable을 1로 할 경우(Q: 0 -> 1)
    1. AND UP: 0 -> 1
    2. 따라서 NOR UP은 무조건 0이 됩니다.
    3. AND DOWN은 확실히 0을 유지하고, NOR UP 또한 0으로 전환되었기 때문에
    4. Q == 1 (NOR DOWN(1) = NOR UP 0 + AND DOWN 0)



SET	RESET	ENABLE	AND(UP)	NOR(UP)	AND(DOWN)	NOR(DOWN)	Q	!Q
1	0	1	1	0	0	1	1	0
0	0	1	0	0	0	1	1	0

- set enable을 끌 경우(상태 유지)
    1. AND UP은 0으로 바뀌지만 NOR DOWN이 1이기 때문에, NOR UP은 (1 + 0)으로 계속 0을 유지합니다.
    2. AND DOWN 과 NOR UP이 0이기 때문에 NOR DOWN은 계속 1을 유지하고 Q는 변하지 않습니다.

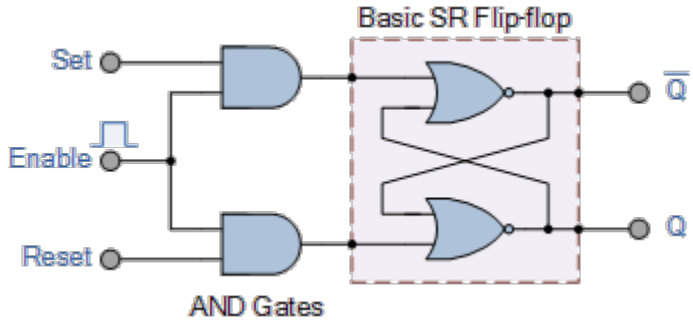


SET	RESET	ENABLE	AND(UP)	NOR(UP)	AND(DOWN)	NOR(DOWN)	Q	!Q
1	0	1	1	0	0	1	1	0
0	0	1	0	0	0	1	1	0

0	1	1	0	0	1	0	0	1

- Q가 1 일때 reset enable을 킬 경우(Q: 1 -> 0, !Q: 0 -> 1)
  1. AND DOWN이 1로 바뀌면서 NOR DOWN또한 무조건 0이 됩니다. 이것으로 Q는 0이 됩니다.
  2. NOR DOWN과 AND UP이 0이기 때문에 NOR UP은 1이 됩니다. 이것으로 !Q가 1 이 됩니다.
- D-flipflop

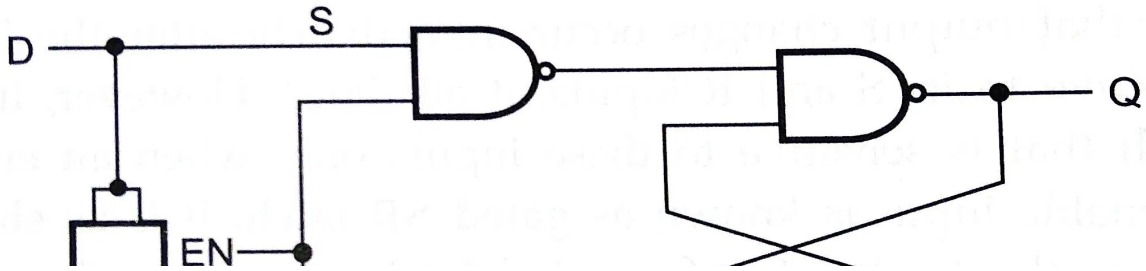
SR flipflop에서 예외적인 상황을 연출해보겠습니다.

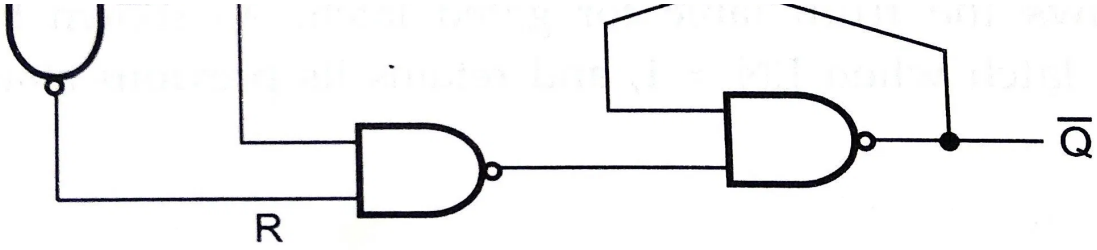


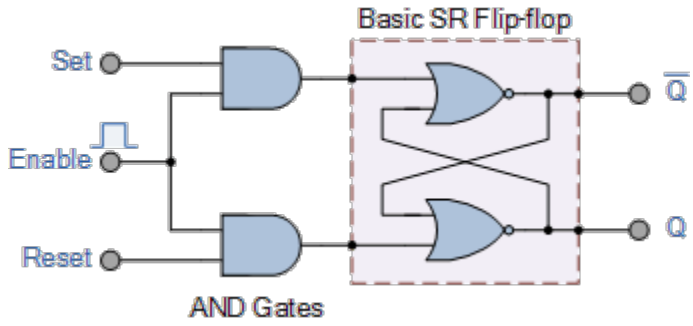
SET	RESET	ENABLE	AND(UP)	NOR(UP)	AND(DOWN)	NOR(DOWN)	Q	!Q
0	1	1	0	0	1	0	0	1
1	1	1	1	0	1	0	0	0

- RESET일때 SET도 킬 경우
  1. AND UP 1로 바뀌는 것으로 모든 AND가 1이 되고 모든 NOR은 0이 됩니다.
  2. SET-RESET을 정확히 조작한게 아닌데 Q, !Q 모두 0이 되어 버렸습니다.

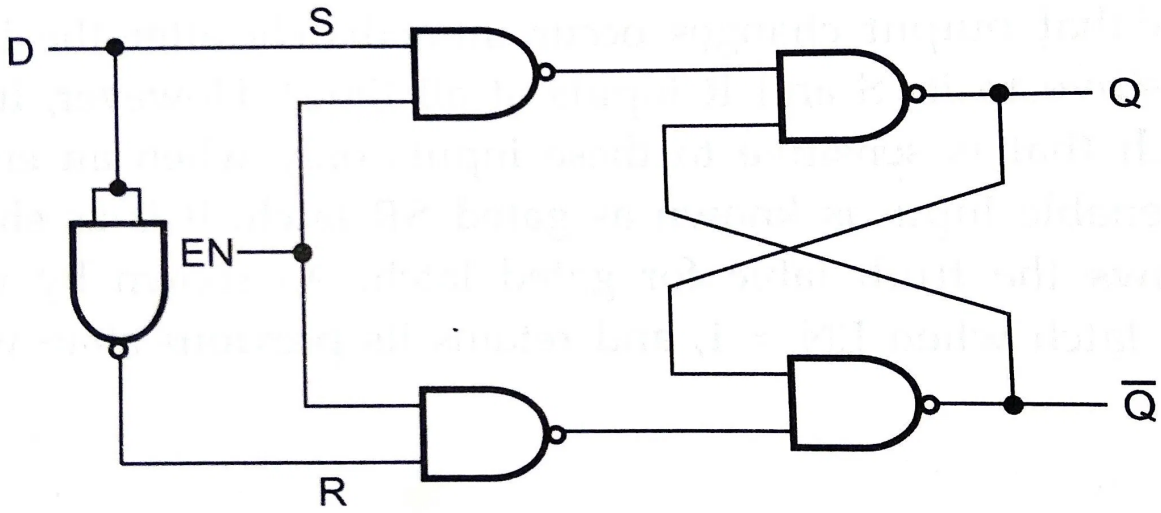
> *SET RESET을 동시에 켜서 저장한 데이터를 삭제하는 것은*
> *동작계획에 포함된 결과가 아니기 때문에 오작동으로 분류한다고 합니다.*
> *SET RESET을 하나의 입력으로 전환하고, Data라는 네이밍을 전환합니다.*





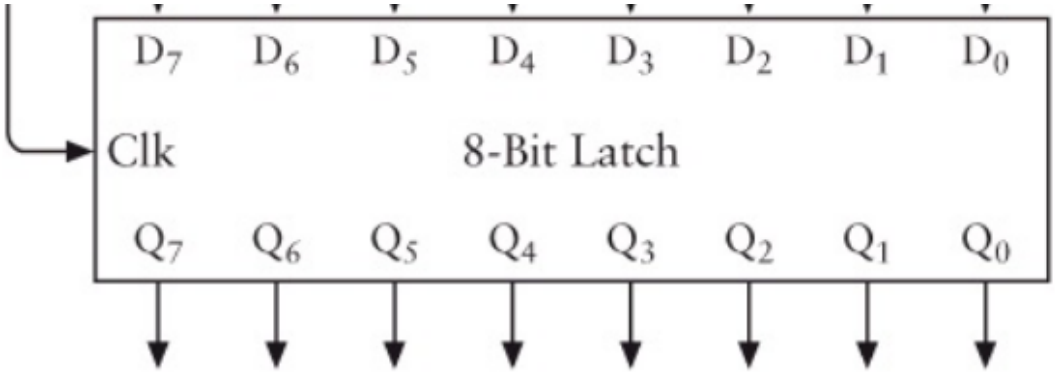


> *그림의 차이는 크지 않습니다.*
> *한번에 하나만 할 수 있도록 입력방식에 인버터를 추가하여 2개로 분리하였습니다.*
> *이것으로 SET RESET중 하나만 할 수 있을 것이며, 한번 입력된 데이터는 잔여전류가 소모되지 않는 이상 삭제되지 않습니다.*



D-flipflop은 동시에 *D-타입 래치*라고 불립니다.
해당 회로가 한 비트의 데이터를 나중에 사용하기 위하여 저장(latch)할 수 있다는 의미라고 합니다.

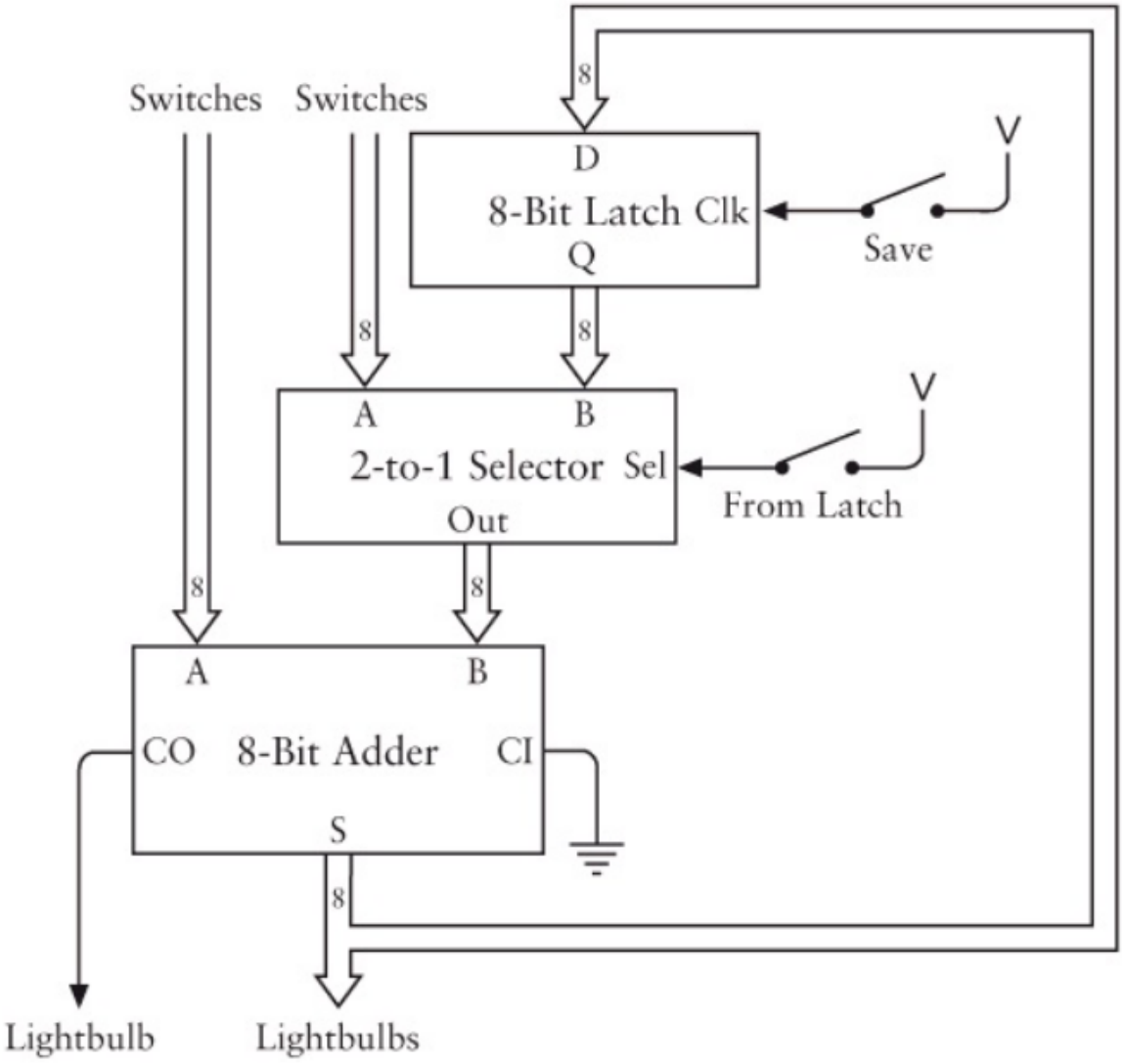
latch + adder



8비트 래치의 구성

- D-타입 래치 8개
- 8개의 D-type latch에 ENABLE을 분산할 수 있는 INPUT을 더 해주면

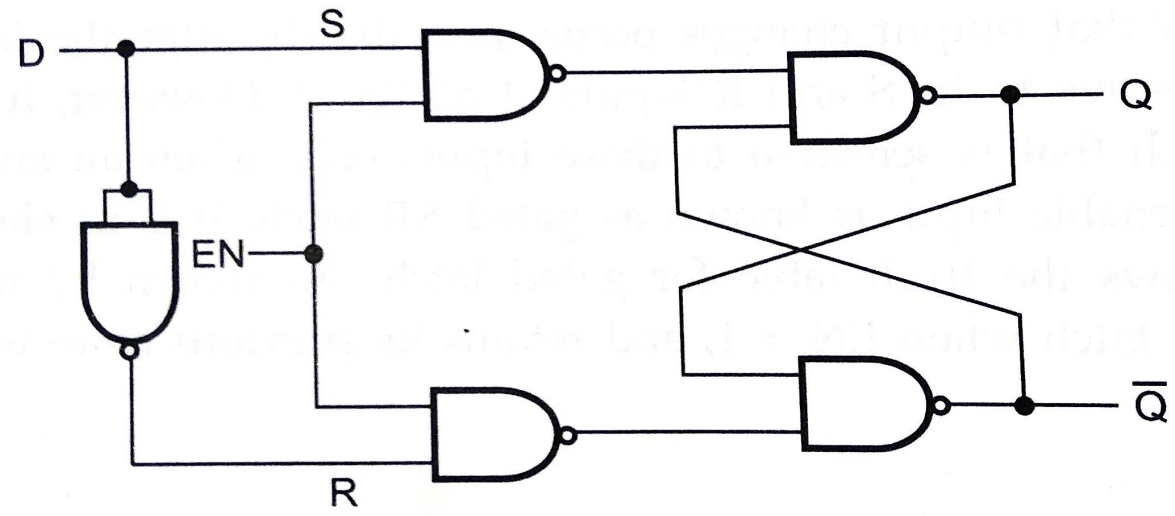
나중을 위해 8비트를 저장 할 수 있습니다.



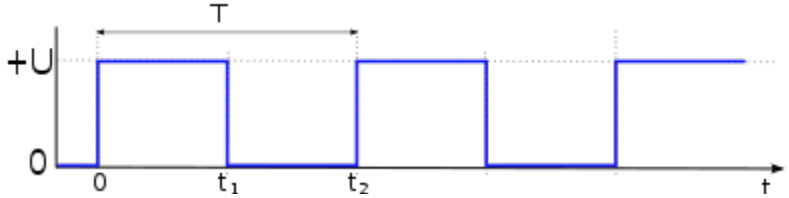
1. 저번시간에 했던 가산기의 출력결과를 DATAPIN으로 8bit latch에 연결해주면, ENABLE(clk)만 들어와 있다면 결과가 잘 저장 될 수 있습니다.
2. Selector 옆에 보이는 스위치를 on을 하면, latch에서 들어오는 8핀의 데이터를 그대로 가산기의 B입력으로 전달해줍니다.

- 이번의 2-1Selector는 양쪽 A, B버스에서 데이터가 들어오는 것을 개별적으로 and스위치에 연결하고, 인버터 스위치를 통해 한 쪽의 입력만 출력으로 연결하도록 보조해주는 단순한 장치라고 합니다.

D-flipflop level trigger 에서 edge trigger로

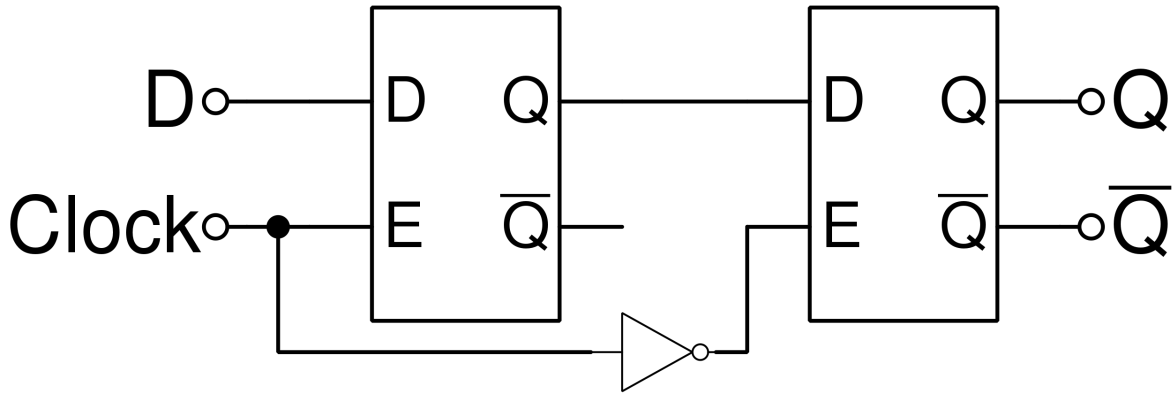


- 기존의 트리거는 ENABLE이 1인 동안만 값을 바꿀 수 있는 상태 입니다.
  - 다르게 말하면 레벨트리거를 사용하게 되면 ENABLE(clock)이 1인 동안 값을 바꾸면 안되기 때문에 홀드타임(유지시간)이 길어집니다.
- 기본적으로 이 상태여도 일부 분야에서는 문제가 없지만, 대부분의 경우에 엣지트리거를 선호 한다고 합니다.
  - 엣지트리거의 경우엔 레벨트리거와 다르게 홀드타임(유지시간)이 자유롭고, data를 미리 설정하는 설정시간(셋업타임)은 차이가 없어서 더 효율적이라고 여겨져서 일단은 더 선호되는 것 같습니다.



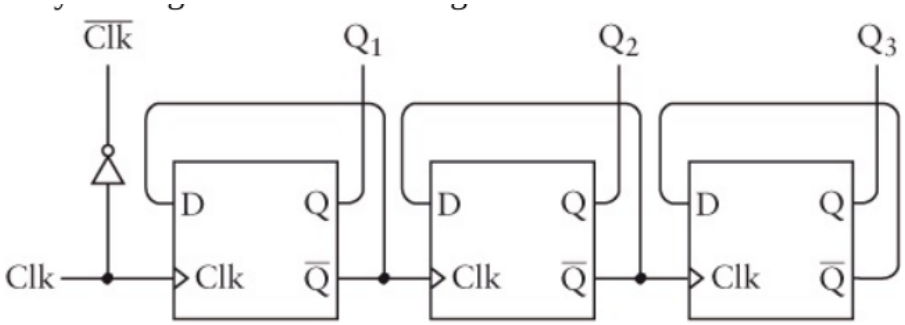
이 말은,

- 데이터를 설정해 놓고, enable을 키는(*클럭이 (0 -> 1)로 전환되는 일순간*) 순간에 데이터를 SYNC하도록 합니다.
- enable이 1의 값을 가지고 있는 상태에서는 데이터를 바꿔도 변화가 메모리에 적용되지 않는다라고 합니다.
- 전류 흐름이 불안정해지는 상황에 값이 변하는 것 또한 막을 수 있을 거라고 생각됩니다.

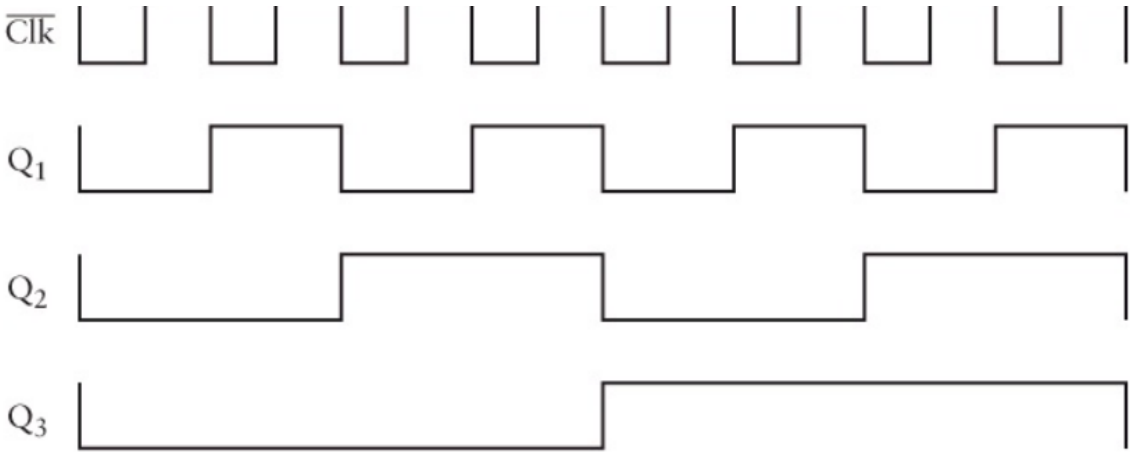


Level trigger D-flipflop을 2개 연결하면, **클럭 상승에 따라** 값을 반영하는 엣지 트리거로 변환이 가능합니다.

timing diagram과 리플카운터



ook at the four signals I've labeled at the top of that diagram:



타이밍 다이어그램

- 그림에 보이는 것은 clock은 반전값이기 때문에 하강 엣지에서 값이 변하게 됩니다. 이렇게 해야 0에서 시작하는 값의 구성할 수 있습니다.
- clock->D1->D2->D3 기준으로 각 D-엣지트리거의 출력을 뒤에 위치한 D-엣지트리거의 clock으로 공급해주면 위 사진과 같은 타이밍으로 값이 변하게 됩니다.
- 이것을 기반으로 카운터를 만들 수 있는데, 각 데이터의 상태변화가 약간의 시차를 두고 바뀌기 때문에 비동기 카운터(asynchronous counter)라고 부르기도 합니다.
    - *(하지만 저항 누적등으로 인해 전파지연이 발생하지만 소프트웨어레벨에서 읽는 시간안에 상승레벨을 감지했다면 1로 처리 하는 등으로 느슨한 허용이 가능합니다. 정밀한 사양이 중요하다면, 동기적 카운터를 설계해서 책에 나오는 리플카운터를 개선해서 쓰기도 한다고 합니다.)*
    - 위에서 아래로 Q1 ~ Q4에 상승레벨에는 1을 넣고, 다운 레벨에는 0을 넣습니다. 그리고 해당 4개의 래치의 데이터를 한번에 읽으면 4비트 숫자가 완성됩니다.
    - (0000, 0001, 0010, 0011, ...);

TAGGED IN

tag1 tag2

Disqus comments not available by default when the website is previewed locally.