Revisiting the Full-Adder

A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1 ×	0
0	1	0	1 ×	0
0	1	1	0	1 ×
1	0	0	1 ×	0
1	0	1	0	1 ×
1	1	0	0	1 ×
1	1	1	1 ×	1 ×

$$S = \bar{A}\bar{B}C_{in} + \bar{A}B\bar{C}_{in} + A\bar{B}\bar{C}_{in} + ABC_{in}$$

$$= \bar{C}_{in}(\bar{A}B + A\bar{B}) + C_{in}(\bar{A}\bar{B} + AB)$$

$$= \bar{C}_{in} \cdot A \oplus B + C_{in} \cdot \overline{(A \oplus B)}$$

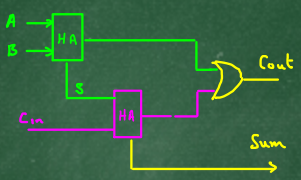
$$= C_{in} \oplus (A \oplus B)$$

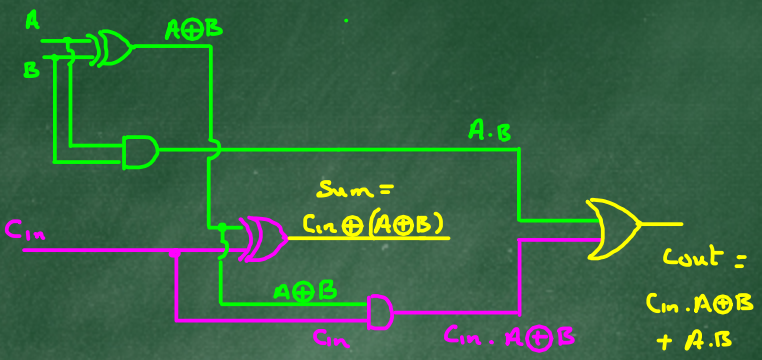
$$C_{out} = \bar{A}BC_{in} + A\bar{B}C_{in} + AB\bar{C}_{in} + ABC_{in}$$

$$= C_{in}(\bar{A}B + A\bar{B}) + AB(\bar{C}_{in} + C_{in})$$

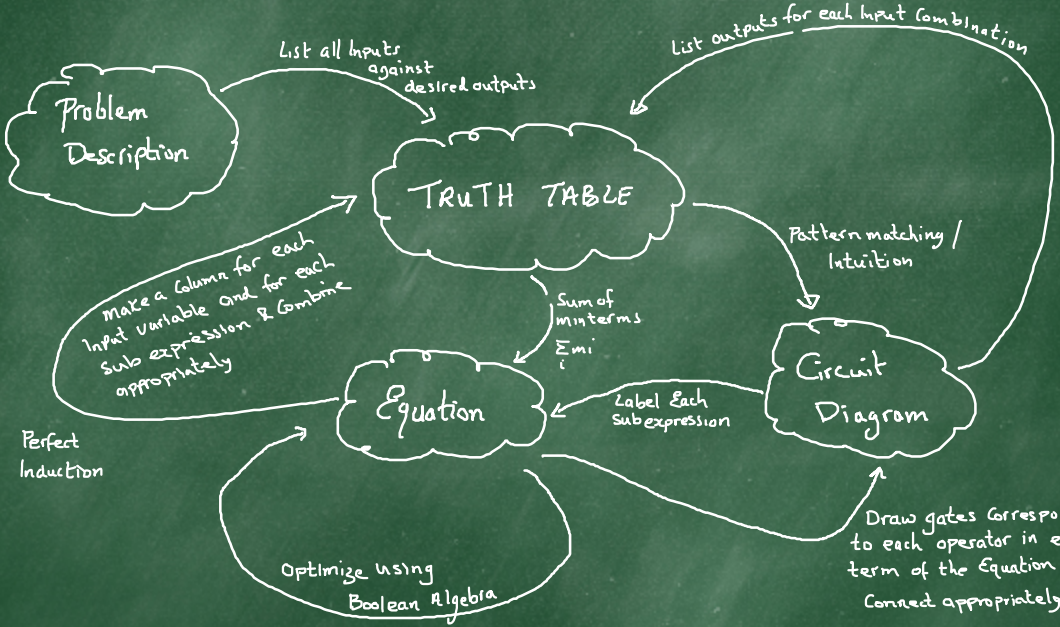
$$= C_{in} \cdot A \oplus B + AB$$

These equations are the same as joining together 2 Half-adders:





The Big Picture

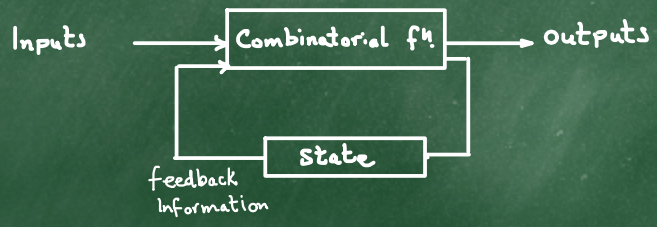


Introducing Memory and Computing with State

Up to this point, we have been only concerned with Combinatorial Circuits – Circuits whose outputs depend only on (are determined by) their Inputs.

We used these Circuits for implementing arithmetic operations, for data transmission using mux, demux, for Serial to Parallel Conversion, for data encoding/decoding (keyboards, 7-Segment displays), and for building Controllers.

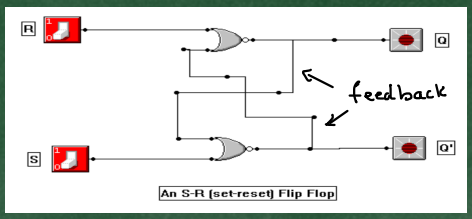
By introducing memory, we can realize the Concept of **State** – allowing us to incorporate past information into the current functioning of our Circuit. In this way, what we do next and be made dependent on what happened in the past.



A Circuit of this kind is called a **Sequential Circuit**

Example: Vending Machine.

Consider the following:

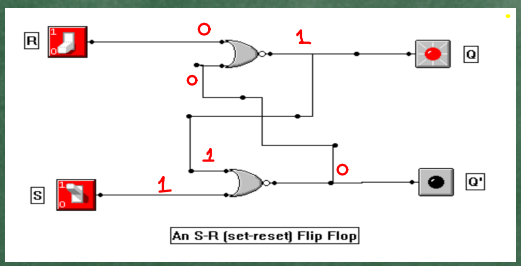


This circuit is called an S-R Flip-Flop. It represents a very simple 1-bit memory element.

The output of each NOR gate is "fed-back" into the input of the other.

The R and S inputs stand for Reset and Set, respectively.

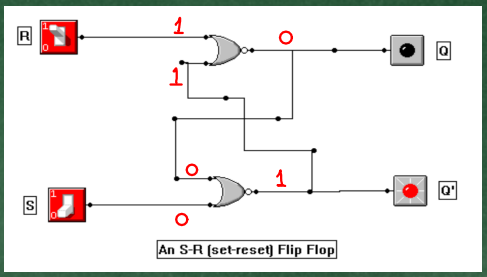
Now, Consider when S=1; R=0:



R S	$\overline{R+S}$
0 0	1
0 1	0
1 0	0
1 1	0

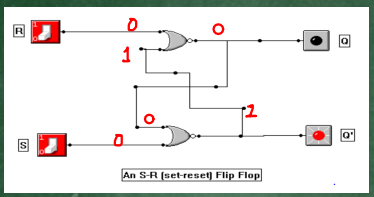
We say the flip-flop is Set. (Q=1)

Consider S = 0 ; R = 1

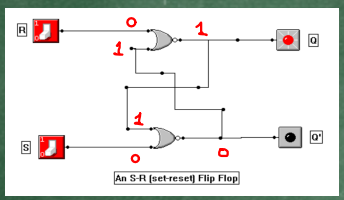


We say the flip flop is reset. (Q = 0)

Consider S = 0 ; R = 0

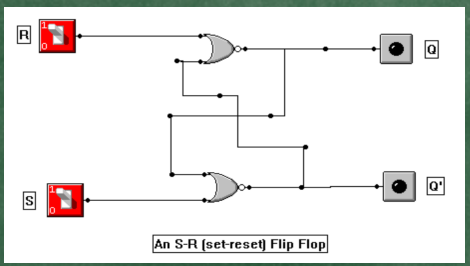


If $\bar{Q}$ was = 1, the ff will remain reset



If Q was = 1, the ff will remain set.

Consider S = 1 ; R = 1



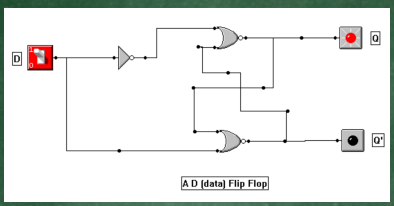
These Inputs will result in both Q and its Complement ($\bar{Q}$) = 0. Therefore, the ff will neither be set nor reset.

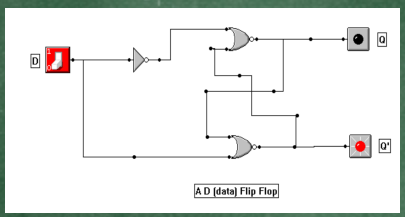
This Combination should be avoided.

To Summarize

R	S	Q	$\bar{Q}$	
0	0	Q	$\bar{Q}$	← state unchanged
0	1	1	0	← set
1	0	0	1	← reset
1	1	0	0	← avoid

To avoid the S=R combinations, consider:





This is called a D-flip flop (D, for data)

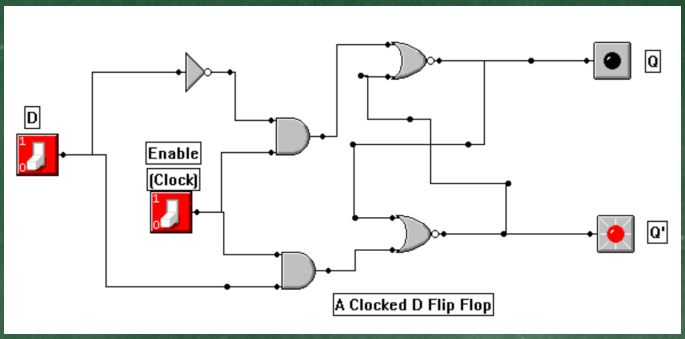
If D=0, Q=0
D=1, D=1

So Q 'follows' D.

At first sight this seems daft. This is the same effect as Connecting the Switch, D, directly to the Led, Q.!

But ....

if we add some more circuitry we can create our first recognisable 1-bit memory element:



When enable = 1 ; Q = D

When enable = 0 ; (i) Q will remain unchanged since R, S are both 0

(ii) D can be changed without affecting the value of Q.

Thus, the value of D is remembered when enable = 1 and can be changed without affecting the "remembered" (stored) value when enable = 0.

The clocked D-flip flop is therefore a single bit memory device we address the memory location using the enable line and we store the value of D in Q as a result.

Multiple D-flip-flops can be used to construct registers — hardware devices capable of storing a number of bits.