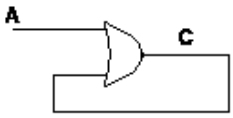
Data Storage in Digital Circuits

The ability to store binary data is critical to the operation of computer systems as well as to other digital circuits. The simple circuits that have been examined so far - NAND, OR, the adder circuits - all are *combinational* logic circuits: their outputs are purely a function of their inputs. For these circuits, if the inputs are known, then the output values can be determined; this is not true for circuits that can store binary information.

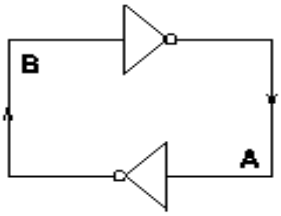
The general structure of a digital circuit is that it has a number of inter-connected functional units, or *gates*, and that the circuit has some inputs and some outputs. If the circuit can be arranged into ordered layers of gates such that the circuit's inputs drive the first layer, the outputs of this layer drive the second layer, and so on, with no feedback of signals from a layer into that layer or a previous one, then the circuit is *combinational* and the circuit's outputs are purely a function of its inputs. The adder circuits of page 12 are of this form and are combinational. If at least one output from a layer is fed back into that layer or a previous layer, then the outputs are no longer just a function of the inputs, but are a function of the inputs and the feedback signals. The feedback signals can be made to reflect earlier (in time) signal inputs, so that the circuit has some sort of `memory' of earlier inputs. In the circuit on the right, once A has been set to 1, C is always 1 after that regardless of A. Thus, this circuit has 2 responses to the input A = 0, depending on whether C = 0 or C= 1. Circuits like this are said to have *internal state*, which modifies their response to inputs.



Circuits with internal state are extremely useful for they enable the current output of a circuit to be determined by previous inputs to the circuit, not purely by the current inputs. For example in a coin-operated ticket machine, the digital control circuit must wait until the correct amount of money has been inserted and only then produce the ticket and the correct change. This circuit has to keep track of the sequence of inputs, the coin values, and activate particular outputs at appropriate times only: this cannot be done with a *combinational* circuit, it needs a circuit with state (a State Machine).
Very complex State Machine circuits can be produced; they are examined in more detail later.

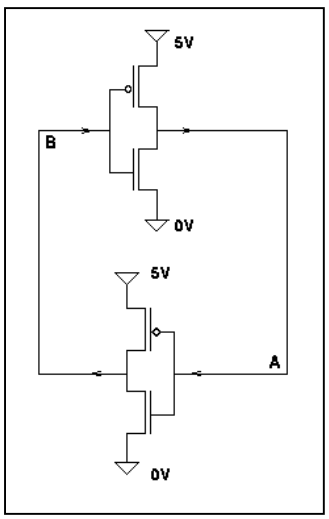
Once a machine has *state*, it has a storage capability: it stores the current state. This is the basis of one major form of storage capability in digital circuits and computer systems.

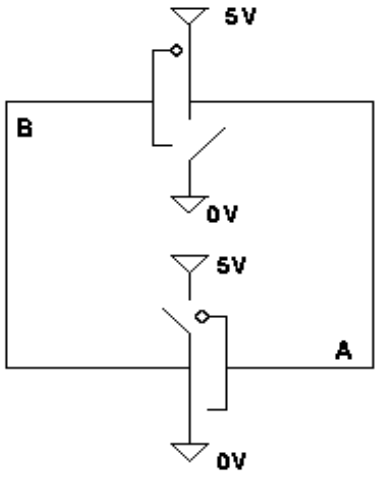
A different form of electronic storage, which will be mentioned later, is used in the *dynamic memory* chips used for main computer memory, while computer disks and tapes use magnetic storage, not electronic.



The feedback circuit with state, which is the basis of many digital storage circuits, is shown below.
This circuit has no inputs: obviously, A, the output value of the upper inverter, and B, the output value of the lower inverter could be used as outputs of the circuit.
There are 2 possible stable *states* of this circuit:-

A = 0, B = 1
A = 1, B = 0



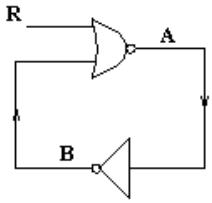


Because there are no inputs to the circuit, there is no means to change its state. When power is applied to the circuit, the circuit enters one of the 2 states, which one cannot be determined in advance.

Although there seems to be a flow around the circuit, there isn't any. This can be seen in the figures to the side. On the left is the circuit redrawn at transistor level in CMOS; on the right the switch

representation of the CMOS circuit for A = 1 (5V) and B = 0 (0V). There is no current flow in the circuit. The output voltage of one inverter maintains the switch settings of the other. While power is applied to the circuit, this configuration will remain. Only if the power is turned off and then turned on again is there any chance of the other circuit state being obtained.
To turn this inverter-based feedback circuit into a memory requires input signals to alter the state. There are several ways of doing this, one of which is demonstrated here.

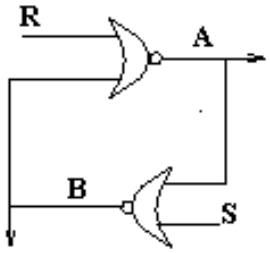
The circuit on the right has one of the inverters replaced by a NOR gate and there is now an input R. The circuit can by setting R = 1 be put into state (A=0,B=1), if it isn't already in that state. However, it is not possible to change the circuit from the state (A=0,B=1) once entered. State (A=1,B=0) can only occur by chance on power up.



Truth Table

R	B	A
0	0	1
0	1	0
1	0	0
1	1	0

The figure below has both inverters replaced by NOR gates and 2 inputs R and S (short-hand for Reset and Set). It is now possible to switch the circuit between its 2 states. The truth table shows the effect of the 4 combinations of the 2 inputs:



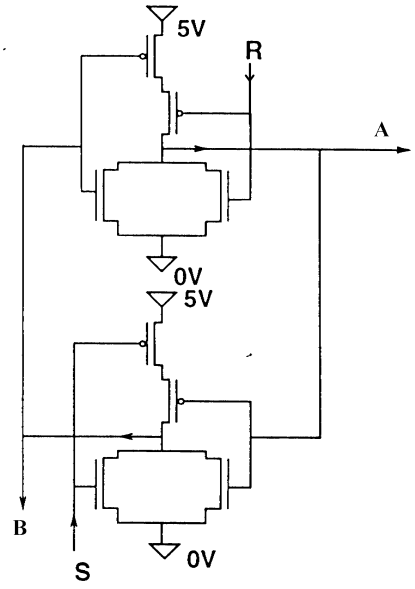
Truth Table:

R	S	A	B	Comment
0	0	A	B	storage mode: A and B stay as set by previous input
0	1	1	0	sets to state A=1,B=0
1	0	0	1	resets to state A=0,B=1
1	1	0	0	this is not a useful input arrangement

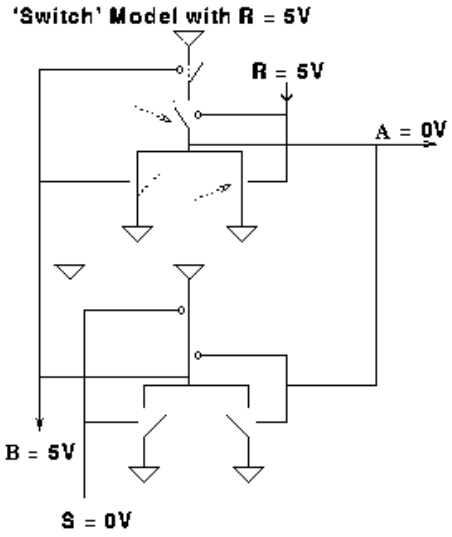
For inputs *R=1,S=0,* the '*1'* on *R* makes *A=0* so that *B =1*.
Changing *R* to '*0'*, so that *R=0 & S=0, B* at '*1'* keeps *A=0*.
Now changing *S* to '*1'* makes *B=0,* so that *A=1,* since *R=0 & B=0*.
Setting *S* back to '*0'*, so that *R=0 & S=0, A* at '*1'* keeps *B=0*.
Thus for *R=0,S=0* the circuit outputs reflects the previous operation, i.e. it has a memory.

The input combination *R=1, S=1* is not useful for the circuit's operation as a memory, because

- on changing to the 'storage' combination of inputs (*R=0,S=*0), it cannot be determined which of the 2 possible states the circuit will enter: *A=0,B=1* or *A=1,B=0*.
- changing one input at a time, the circuit state entered will depend on the order of change.

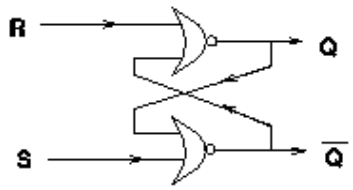


The circuit to the left shows the CMOS logic implementation of the circuit, while the figure to the right shows the switch representation of the CMOS circuit for R at 5V. If R is set to 0V, only the switches marked by an arrow change, and the values on A or B are unchanged.



This circuit is usually called an ***RS flip-flop***: *flip-flop* is one name for an electronic storage device another name is *latch*.

The circuit below is the traditional layout of the RS-flipflop.
Since for all the useful input combinations the outputs are opposite binary values, the outputs are marked as Q and its inverse. It can be seen that R resets Q, while S sets Q. The signal feedback in the circuit is clearly shown in this layout.



If the circuit's states are labelled, *State 0* and *State 1* as Q is 0 or 1, then Q outputs the stored state of the circuit.