Name : irfan ullah

ID : 15431

Examination : Lab

Subject : Digital Logic Design

Course code(CS): CSC-201

Program : BC (CS)

(a) Half adder using logic gates

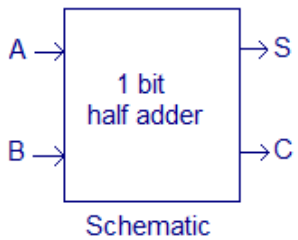
Half adder circuit.

To understand what is a half adder you need to know what is an adder first. Adder circuit is a combinational digital circuit that is used for adding two numbers. A typical adder circuit produces a sum bit (denoted by S) and a carry bit (denoted by C) as the output. Typically adders are realized for adding binary numbers but they can be also realized for adding other formats like BCD (binary coded decimal, XS-3 etc. Besides addition, adder circuits can be used for a lot of other applications in digital electronics like address decoding, table index calculation etc. Adder circuits are of two types: Half adder ad Full adder. Full adders have been already explained in a previous article and in this topic I am giving stress to half adders.

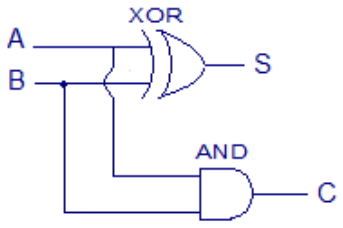
Half adder is a combinational arithmetic circuit that adds two numbers and produces a sum bit (S) and carry bit (C) as the output. If A and B are the input bits, then sum bit (S) is the X-OR of A and B and the carry bit (C) will be the AND of A and B. From this it is clear that a half adder circuit can be easily constructed using one X-OR gate and one AND gate. Half adder is the simplest of all adder circuit, but it has a major disadvantage. The half adder can add only two input bits (A and B) and has nothing to do with the carry if there is any in the input. So if the input to a half adder have a carry, then it will be neglected it and adds only the A and B bits. That means the binary addition process is not complete and that's why it is called a half adder. The truth table, schematic representation and XOR//AND realization of a half adder are shown in the figure below.

Inputs		Outputs	
A	B	S	C
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1

Truth table



Schematic



Realization

(b) Half-subtractor using logic gate

Half Subtractor-

Half Subtractor

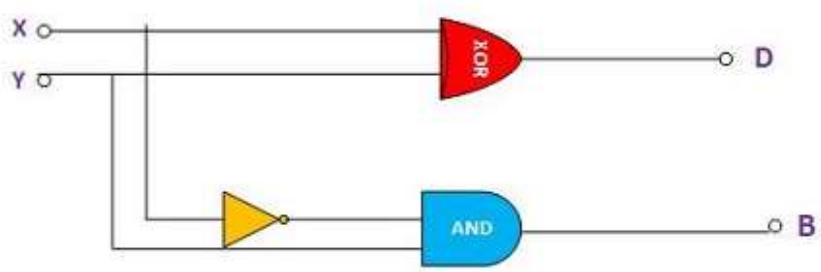
Definition:

The Half Subtractor is a digital circuit which processes the subtraction of two 1-bit numbers. In this, the two numbers involved are termed as subtrahend and minuend**.** In the subtraction procedure, the subtrahend will be subtracted from minuend. The circuit of Half subtractor consists of two inputs and two outputs.

The inputs of the half subtractor circuit will be subtrahend and minuend. On the other hand, the output will be the difference and the borrow. The word “HALF” before the subtractor signifies that it deals with only two 1-bit numbers**,** it has nothing to do with the borrow from the previous stage. The logic symbol of half subtractor is represented in the diagram below.

Circuit of Half Subtractor :

The logic circuit of Half subtractor involves usage of logic gates. In order to design logic circuit, we should understand two concepts. First is the difference operation of half subtractor resembles operation of which logic circuit. Secondly, the borrow generated by half subtractor will also be in accordance with the particular operation which will resemble operation of any logic gate.

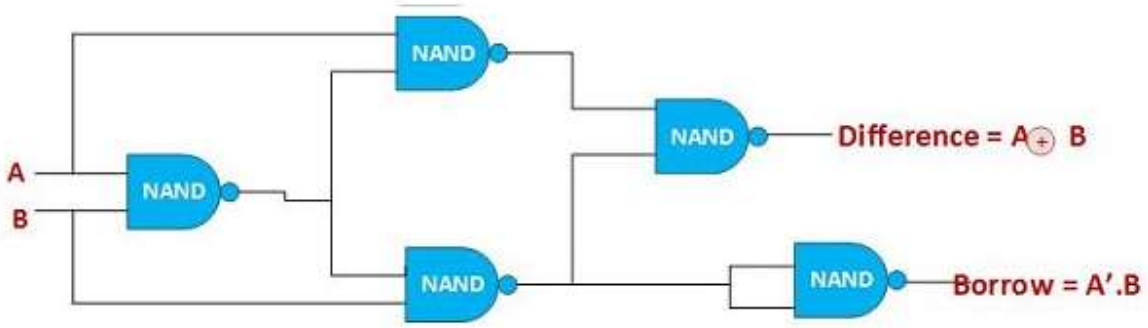


Half Subtractor

Truth Table of Half Subtractor :

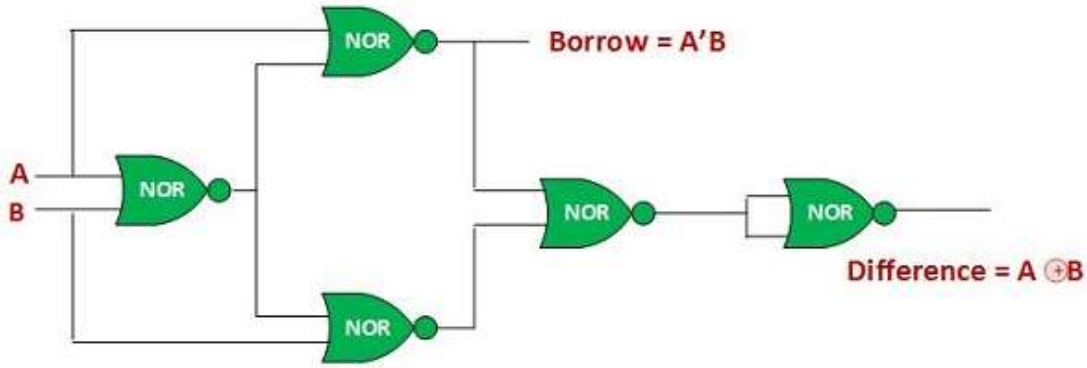
Inputs		Outputs	
X	Y	D	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Half Subtractor using NAND gates:



Half Subtractor using NAND Gate

Half Subtractor using NOR gates:

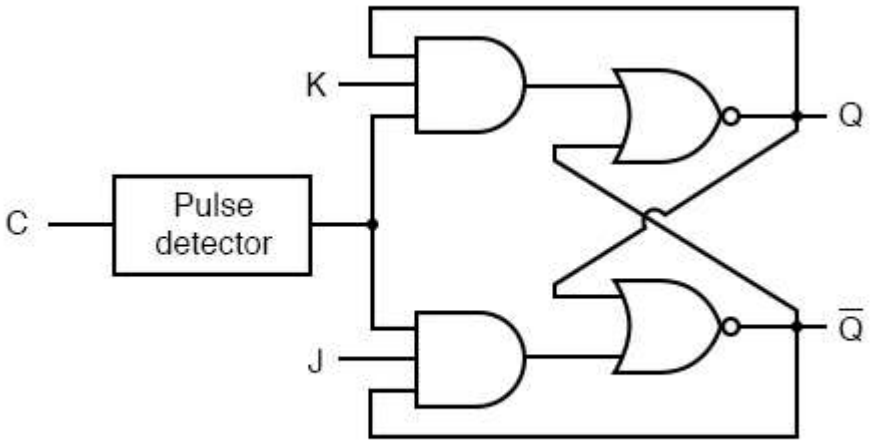


Half Subtractor using NOR Gate

(c) J K Flip flop

J K Flip flop:

Another variation on a theme of bistable multivibrators is the J-K flip-flop. Essentially, this is a modified version of an S-R flip-flop with no “invalid” or “illegal” output state. Look closely at the following diagram to see how this is accomplished:



C	J	K	Q	$\overline{Q}$
⌐¬	0	0	latch	latch
⌐¬	0	1	0	1
⌐¬	1	0	1	0
⌐¬	1	1	toggle	toggle
x	0	0	latch	latch
x	0	1	latch	latch
x	1	0	latch	latch
x	1	1	latch	latch

The J and K Inputs

What used to be the S and R inputs are now called the J and K inputs, respectively. The old two-input AND gates have been replaced with 3-input AND gates, and the third input of each gate receives feedback from the Q and not-Q outputs.

What this does for us is permit the J input to have effect only when the circuit is reset, and permit the K input to have effect only when the circuit is set.

In other words, the two inputs are *interlocked*, to use a relay logic term, so that they cannot both be activated simultaneously.

If the circuit is "set," the J input is inhibited by the 0 status of not-Q through the lower AND gate; if the circuit is "reset," the K input is inhibited by the 0 status of Q through the upper AND gate.

When both J and K inputs are 1, however, something unique happens. Because of the selective inhibiting action of those 3-input AND gates, a "set"

state inhibits input J so that the flip-flop acts as if J=0 while K=1 when in fact both are 1.

On the next clock pulse, the outputs will switch ("toggle") from set (Q=1 and not-Q=0) to reset (Q=0 and not-Q=1). Conversely, a "reset" state inhibits input K so that the flip-flop acts as if J=1 and K=0 when in fact both are 1. The next clock pulse toggles the circuit again from reset to set.

(d) Serial in-serial Out shift register

Serial in Serial Out (SISO) Shift Register:

Serial In Serial Out (SISO) shift registers are a kind of shift registers where both data loading as well as data retrieval to/from the shift register occurs in serial-mode. Figure 1 shows a n-bit synchronous SISO shift register sensitive to positive edge of the clock pulse. Here the data word which is to be stored is fed bit-by-bit at the input of the first flip-flop. Further it is seen that the inputs of all other flip-flops (except the first flip-flop FF_1) are driven by the outputs of the preceding ones say for example, the input of FF_2 is driven by the output of FF_1. At last the data stored within the register is obtained at the output pin of the n^{th} flip-flop in serial-fashion.

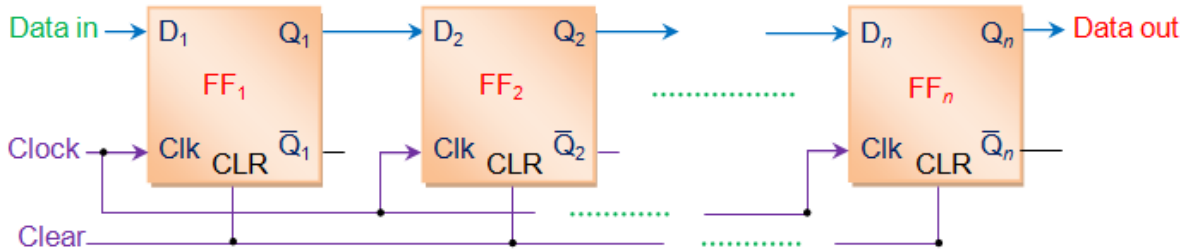


Figure 1 *n*-bit Right-Shift Serial-in Serial-Out Shift Register

Initially all the flip-flops in the register are cleared by applying high on their clear pins. Next the input data word is fed serially to FF_1.

This causes the bit appearing at the D_1 pin (B_1) to be stored into FF_1 as soon as the first leading edge of the clock appears. Further at the second clock tick, B_1 gets stored into FF_2 while a new bit enters into FF_1 (B_2).

This kind of shift in data bits continues for every rising edge of the clock pulse. This indicates that for every single clock pulse the data within the register moves towards right by a single bit. Thus the design shown in Figure 1 is regarded as a right-shift SISO shift **register**. Following the data transmission as explained, one can note that the first bit of an input word appears at the output of n^{th} flip-flop for the n^{th} clock tick. On applying further clock cycles, one gets the next successive bits of the input data word as the serial output (Table I). The waveforms pertaining to the same are shown by

Clock Cycle	Data in	Q_1	Q_2	...	Q_n = Data out
1	B_1	B_1	0	...	0
2	B_2	B_2	B_1	...	0
3	B_3	B_3	B_2	...	0
4	B_4	B_4	B_3	...	0
5	B_5	B_5	B_4	...	0
6	B_6	B_6	B_5	...	0
.	.	.	.	...	.
.	.	.	.		.
.	.	.	.		.
n	B_n	B_n	B_{n-1}	...	B_1
n+1	B_{n+1}	B_{n+1}	B_n	...	B_2
.	.	.	.	...	.
.	.	.	.		.
.	.	.	.		.

Serial Output Bits of SISO (Right-Shift) Shift Register

Table I **Data Movement in Right-Shift SISO Shift**

(e) Synchronous BCD Counter

BCD-synchronous-counter: