Familiarization with binary addition and subtraction

Objectives:

- To construct and verify binary half adder.
- To construct and verify binary half adder using NAND gate only.
- To construct and verify binary half subtractor.
- To construct and verify binary half subtractor using NAND gate only.
- To construct and verify combinational binary half adder and subtractor.
- To construct and verify binary full adder.
- To construct and verify binary full subtractor.
- To construct and verify combinational binary full adder and subtractor.
- To construct and verify binary full adder using two half adder.
- To construct and verify binary full subtractor using two half subtractor.

Equipments And Components Required:

- PC with proteus.
- Logic circuit friendly bread board.
- Quad two input AND gates (7408).
- Quad two input NOR gates (7432).
- Hex Inverter (7404).
- Quad two input XOR gates (7486/4030)
- Required number of connecting wires.

RELATED THEORY:

Boolean Function:

- Boolean algebra deals with binary variables and logic operation. A Boolean Function is described by an algebraic expression called Boolean expression which consists of binary variables, the constants 0 and 1, and the logic operation symbols. Consider the following example:

F (A, B, C, D) = $A+B\bar{C}+ADC$ Equation No.1

Boolean Function Boolean Expression

- Here the left side of the equation represents the output Y. So we can state equation no. 1:
- Y = $A+B\bar{C}+ADC$

Boolean Function Representation

- By using Boolean laws and theorems, we can simplify the Boolean functions of digital circuits. A brief note of different ways of representing a Boolean function is shown below:
- Sum-of-Products (SOP) Form
- Product-of-sums (POS) form
- Canonical forms

There are two types of canonical forms:

- Sum-of-min terms or Canonical SOP
- Product-of- max terms or Canonical POS

Sum of Product (SOP) Form

- In this SOP form of Boolean function representation, the variables are operated by AND (product) to form a product term and all these product terms are ORed (summed or added) together to get the final function.
- Example: AB + ABC + CDE, (AB)’ + ABC + CDE’
- SOP form can be obtained by:
  - Writing an AND term for each input combination, which produces HIGH output.
  - Writing the input variables if the value is 1, and write the complement of the variable if its value is 0.
  - OR the AND terms to obtain the output function.

Product of Sums (POS) Form

- In this POS form, all the variables are ORed, i.e. written as sums to form sum terms.
- All these sum terms are ANDed (multiplied) together to get the product-of-sum form.
- POS form can be obtained by:

i. Writing an OR term for each input combination, which produces LOW output.

ii. Writing the input variables if the value is 0, and write the complement of the variable if its value is 1.

iii. AND the OR terms to obtain the output function.

Features of Combinational Logical Circuits:

1. Here the output is only depended on Input sequence.

2. The combinatinal circuits do not support bit storage (i.e. no memory storage)

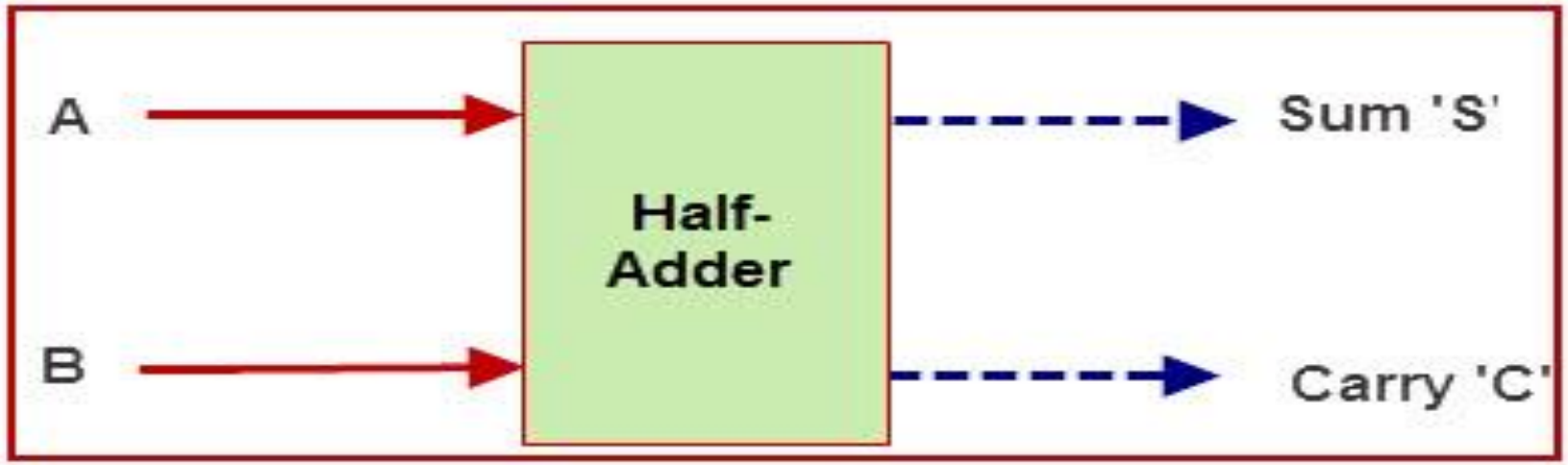
3. Made of Simple Gates like AND, OR ,NOT , NAND ,EXOR etc only.

4. This Systems are Time independent.

Designing Binary Adder

Half Adder

- Half adder is a combinational arithmetic circuit that adds two bits and produces a sum bit (S) and carry bit (C) as the output.
- 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.



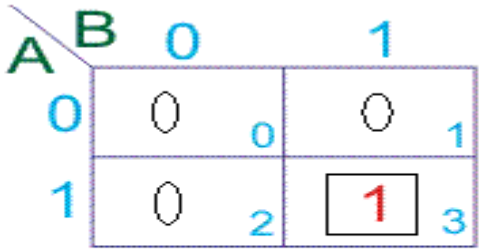
Half Adder Truth Table

- If we assume A and B as the two bits whose addition is to be performed, a truth table for half adder with A, B as inputs and Sum, Carry as outputs can be tabulated as follows:

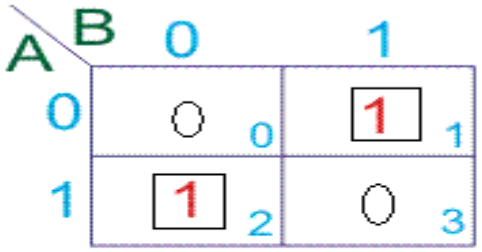
Truth Table			
Input		Output	
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

K-map for Half Adder

- Now from this truth table we can draw K-map for carries and sums separately.



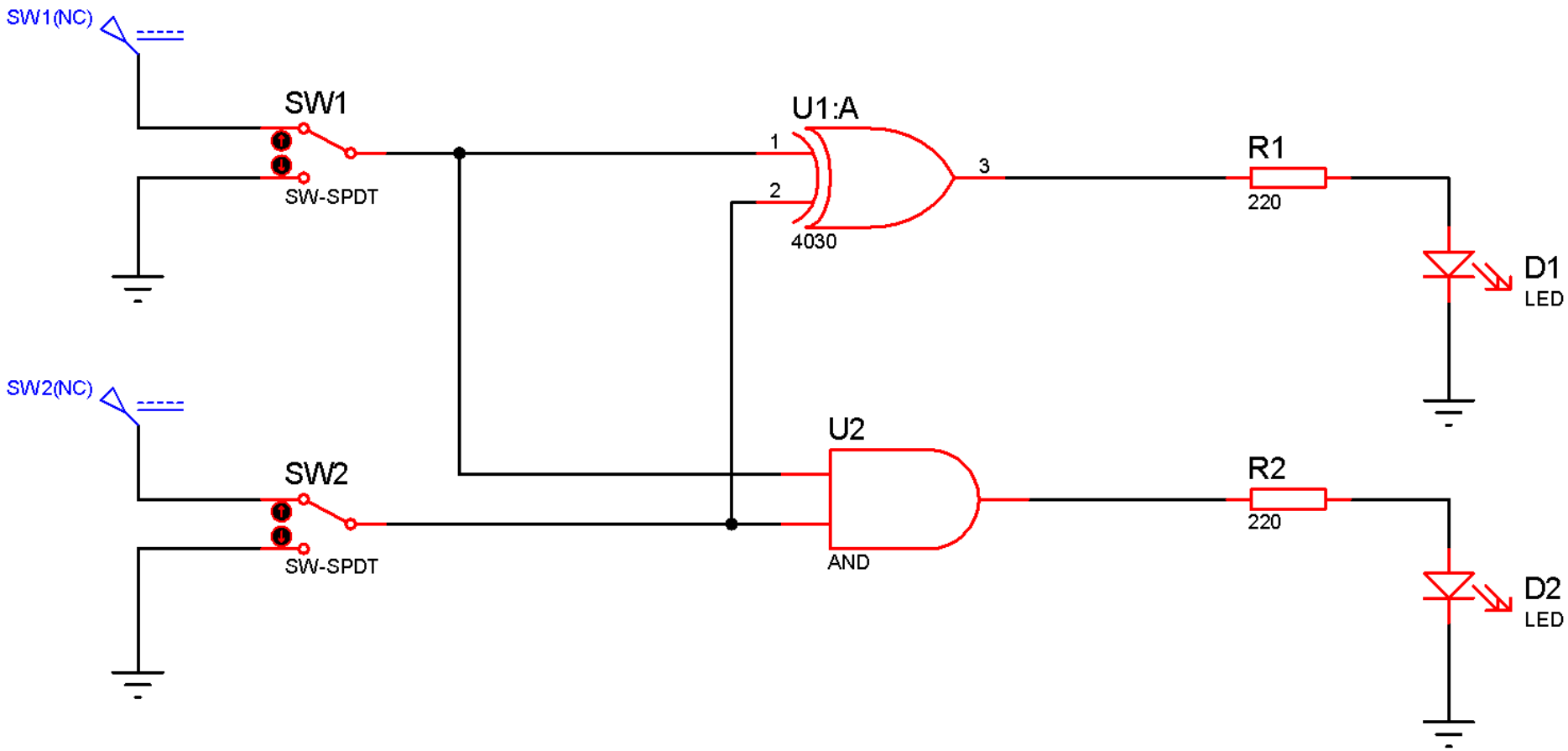
K-map for Carry



K-map for Sum

- From above K-map we get,
- CARRY (C) = AB & SUM (S) = $\bar{A}$B + A$\bar{B}$

Construction Of Binary Half Adder Using XOR & AND Gate



Binary Half Adder using NAND Gate

- SUM = A’B + AB’ CARRY = AB
- = A $\oplus$ B = ((AB)’)’

= (A $\odot$ B)’

= (A’B’ + AB)’

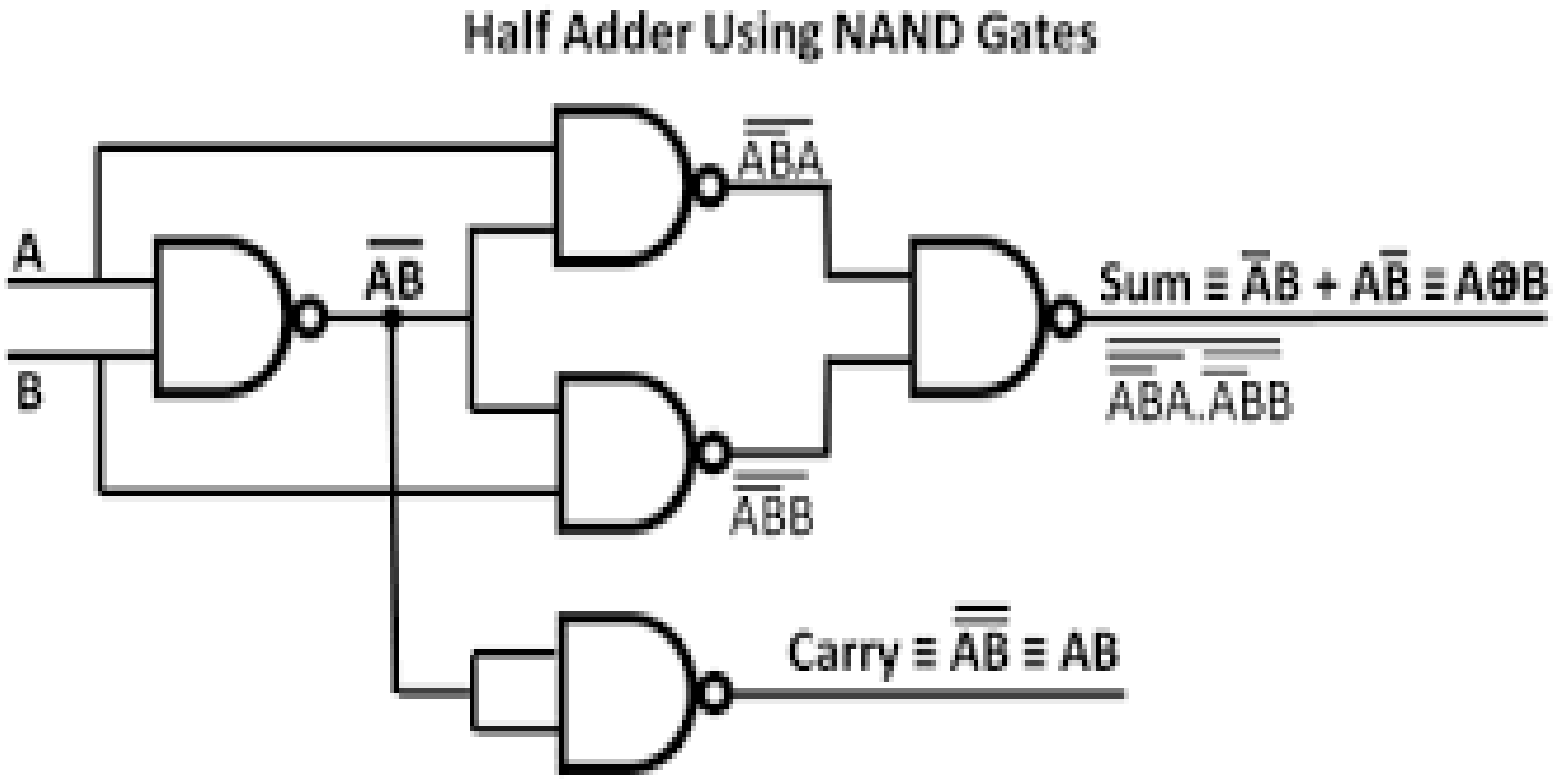
= (A’B’)’ . (AB)’

= (A + B) . (AB)’

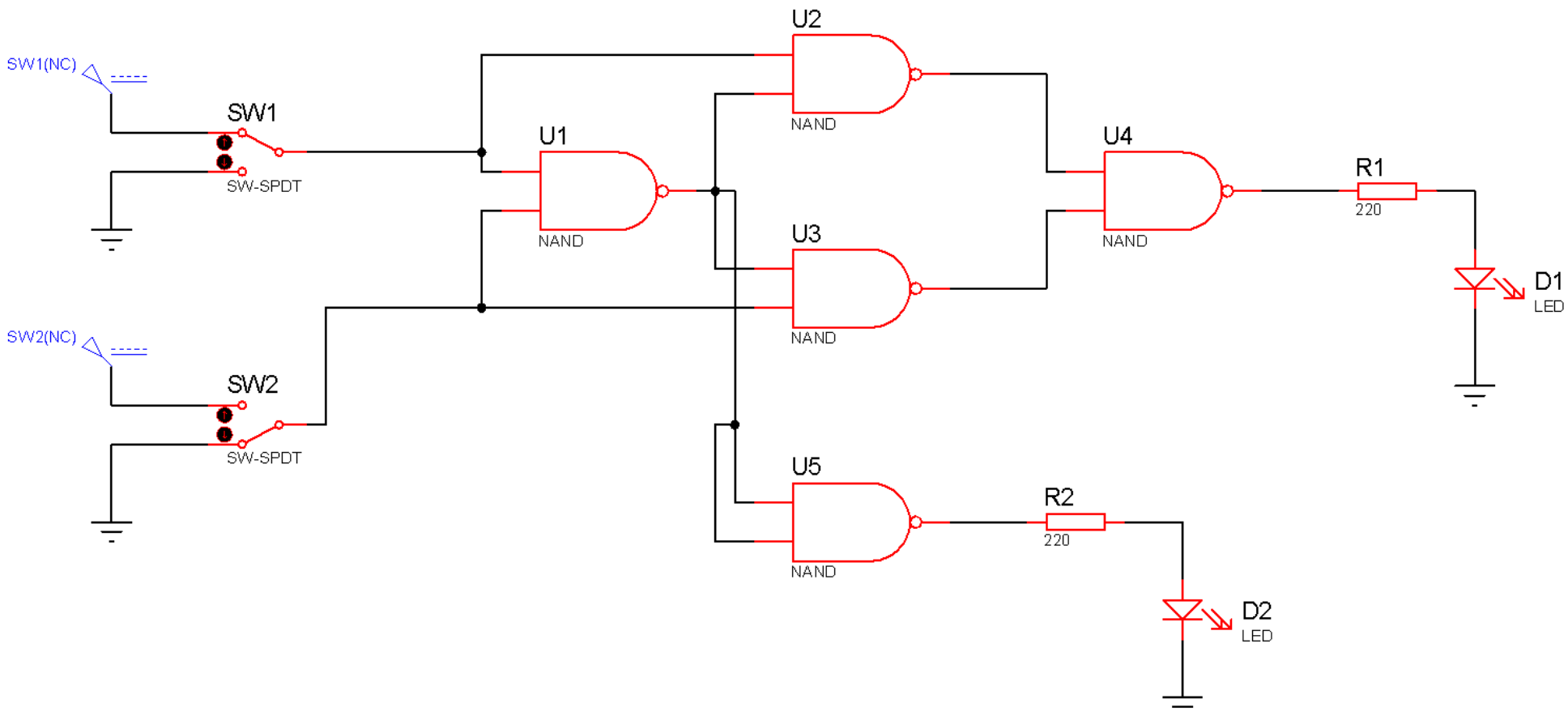
= A (AB)’ + B (AB)’

= [(A(AB)’)’ . (B(AB)’)’]

Concept


Half Adder Using NAND Gates
A
B
$\overline{AB}$
$\overline{\overline{AB}A}$
$\overline{\overline{AB}B}$
Sum ≡ $\overline{A}B + A\overline{B} \equiv A \oplus B$
$\overline{\overline{AB}A}.\overline{\overline{AB}B}$
Carry ≡ $\overline{\overline{AB}} \equiv AB$

Practical diagram for HA using NAND



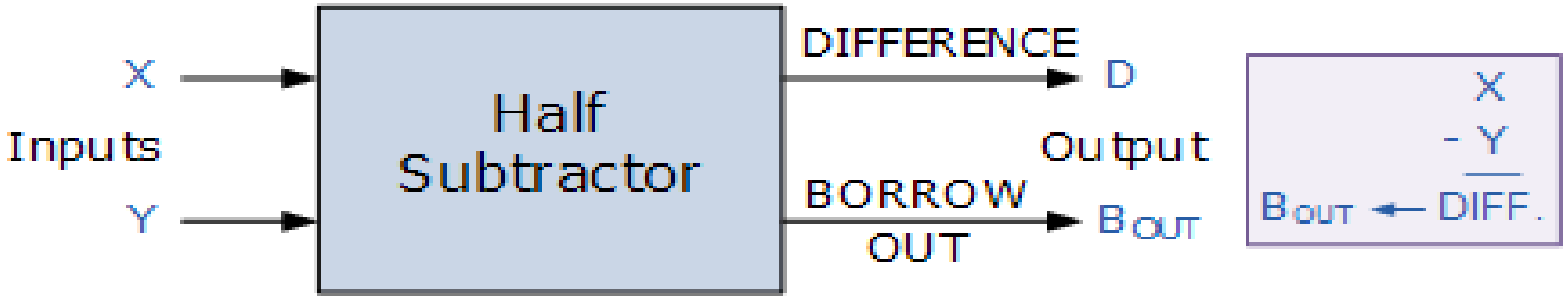
Binary Half Adder using NOR Gate:

- SUM = A’B’ + AB
- = A⊙B
- = (A ⊕ B)’
- = (A’B’ + AB)’
- = (A’B’)’ . (A.B)’
- = (A + B) . (A’ + B’)
- = A.A’ + A.B’ + A’.B +B.B’
- = A.A’ + A’.B + A.B’ + B.B’
- = [(A’ (A+B))’]’ + [(B’ (A+B))’]’
- = [A + (A+B)’]’ + [B + (A+B)’]’

CARRY = (A+B) (A+B’) (A’+B)

Designing Binary Subtractor

- **Half Subtractor**
- The half-subtractor is a combinational circuit which is used to perform subtraction of two bits. It has two inputs, X (minuend) and Y (subtrahend) and two outputs D (difference) and B (borrow). Thus, Half Subtractor is used for subtracting one single bit binary digit from another single bit binary digit. The block diagram of half subtractor is:



Half Subtractor Truth Table:

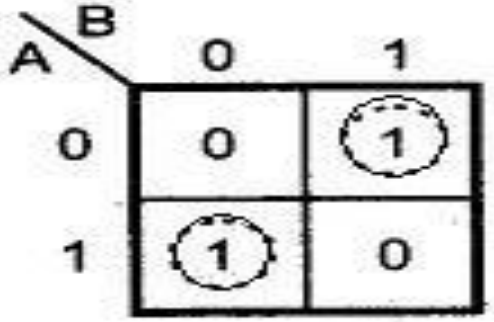
Minuend (A)	Subtrahend (B)	Difference (D)	Borrow (b)
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

K-Map For Half Subtractor:

D = A'B+AB' = $A \oplus B$

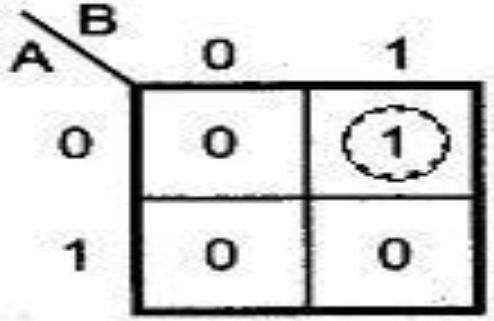
Bo = A'B

For Difference



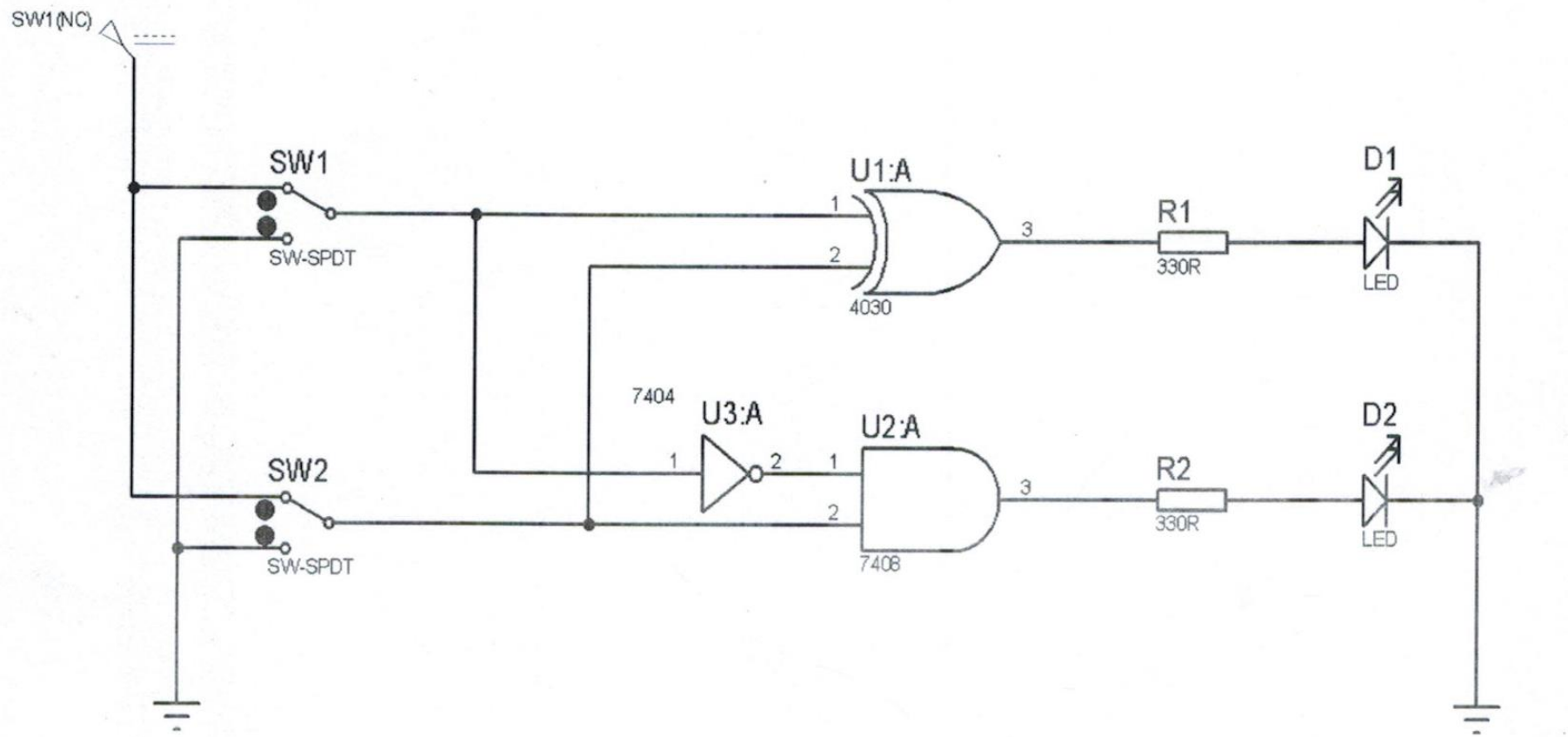
Difference = $A\overline{B} + \overline{A}B$
= $A \oplus B$

For Borrow

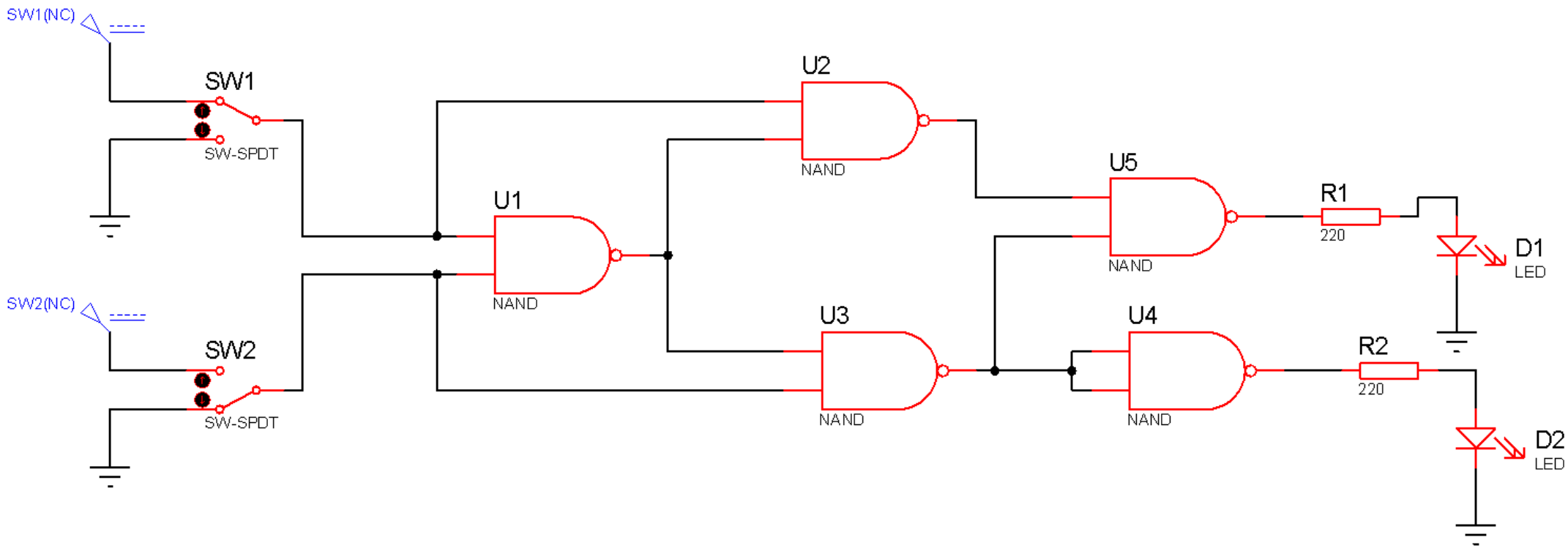


Borrow = $\overline{A}B$

Practical diagram for Half Subtractor



Construction Of Binary Half Subtractor Using NAND Gate



Binary Adder and Subtractor in a circuit

Half Adder and Half Subtractor in a circuit

It is a circuit that that accepts three inputs to produce two outputs. The first two inputs are A and B, which are the significant bits, on which binary addition and subtraction is applied according to the third input bit, D, which is represents the mod of the operation. The circuit performs addition on A and B if D=0 and subtraction if D=1. The two-output generated are the sum or difference, represented by S and carry or borrow represented by C.

Truth Table for HA and HS in a circuit

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

K Map for HA and HS in a circuit

Simplifying S using k-map,

	A'B'	A'B	AB	AB'
D'	0	1	0	1
D	0	1	0	1

→S = A'B+ AB'

= A ⊕ B

Simplifying C using k-map,

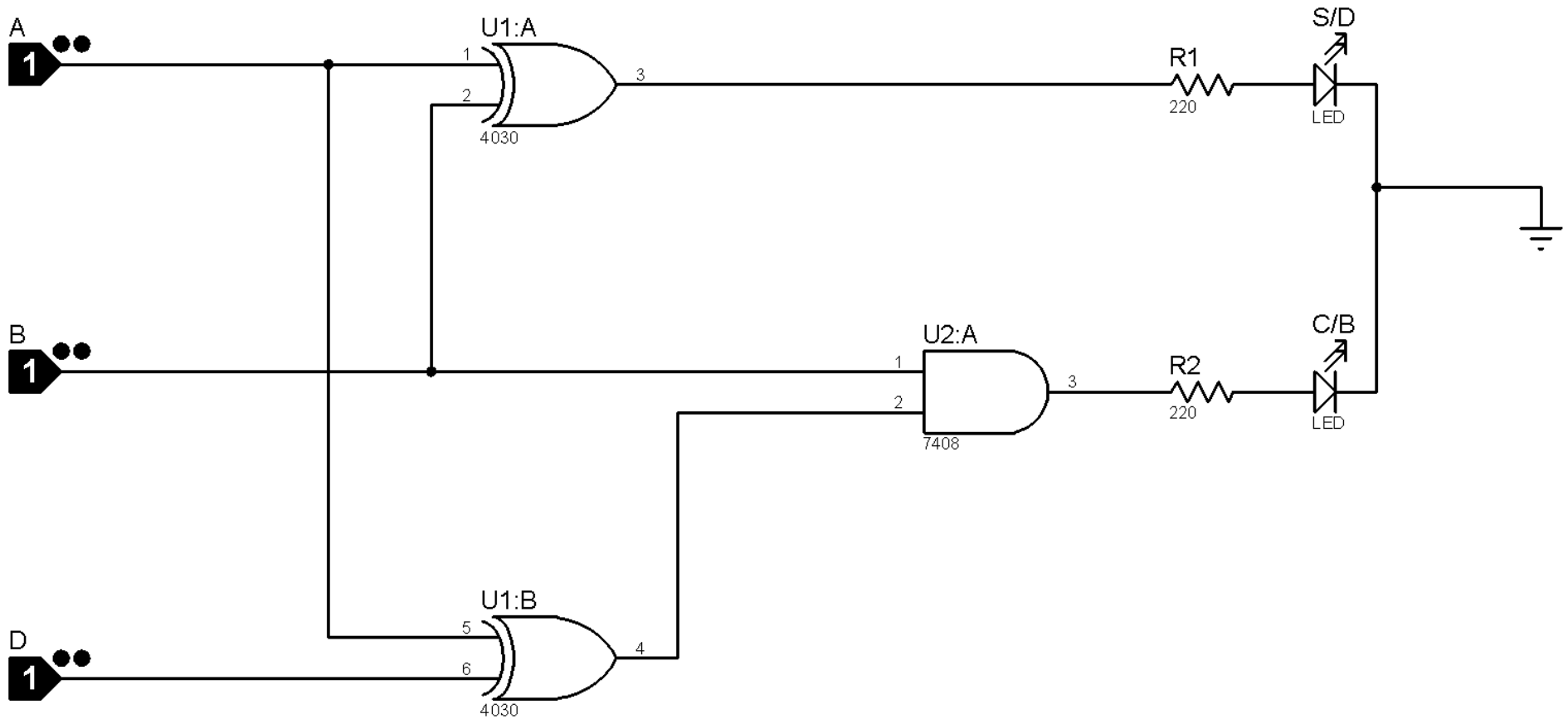
	A'B'	A'B	AB	AB'
D'	0	0	1	0
D	0	1	0	0

→C = D'AB + DA'B

= B (D'A + DA')

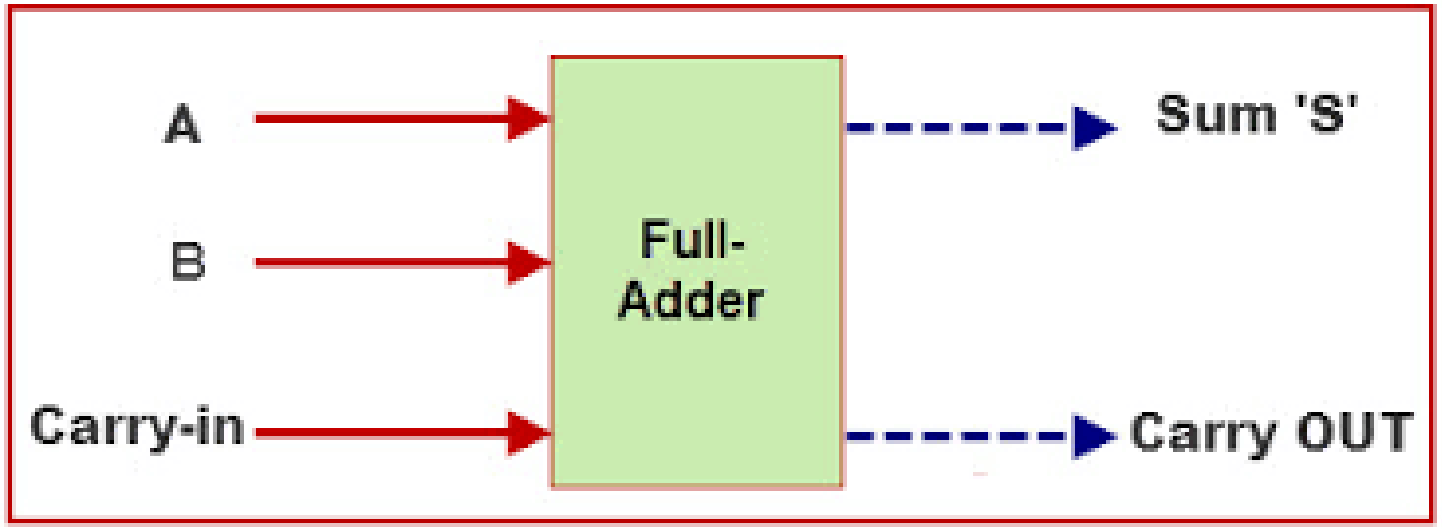
= B (D ⊕ A)

Practical Diagram For HA and HS in a circuit



FULL ADDER

- Full adder is a conditional circuit which performs full binary addition that means it adds two bits and a carry and outputs a sum bit and a carry bit. The first two inputs are A and B and the third input is an input carry as C-IN. The block diagram of a full adder with A, B and CIN as inputs and S, CoUT as outputs is shown below:

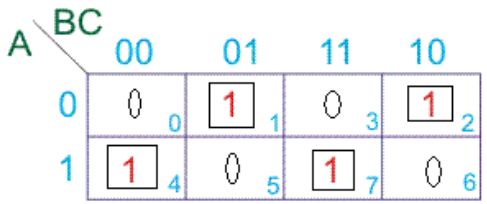


Full Adder Truth Table

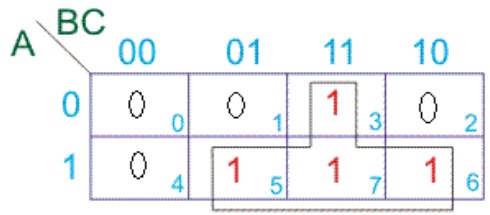
The truth table for full adder is shown below:

INPUTS			OUTPUTS	
A	**B**	**C-IN**	**SUM**	**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

Full adder K Map and Boolean Equation



K-map for Sum (S)

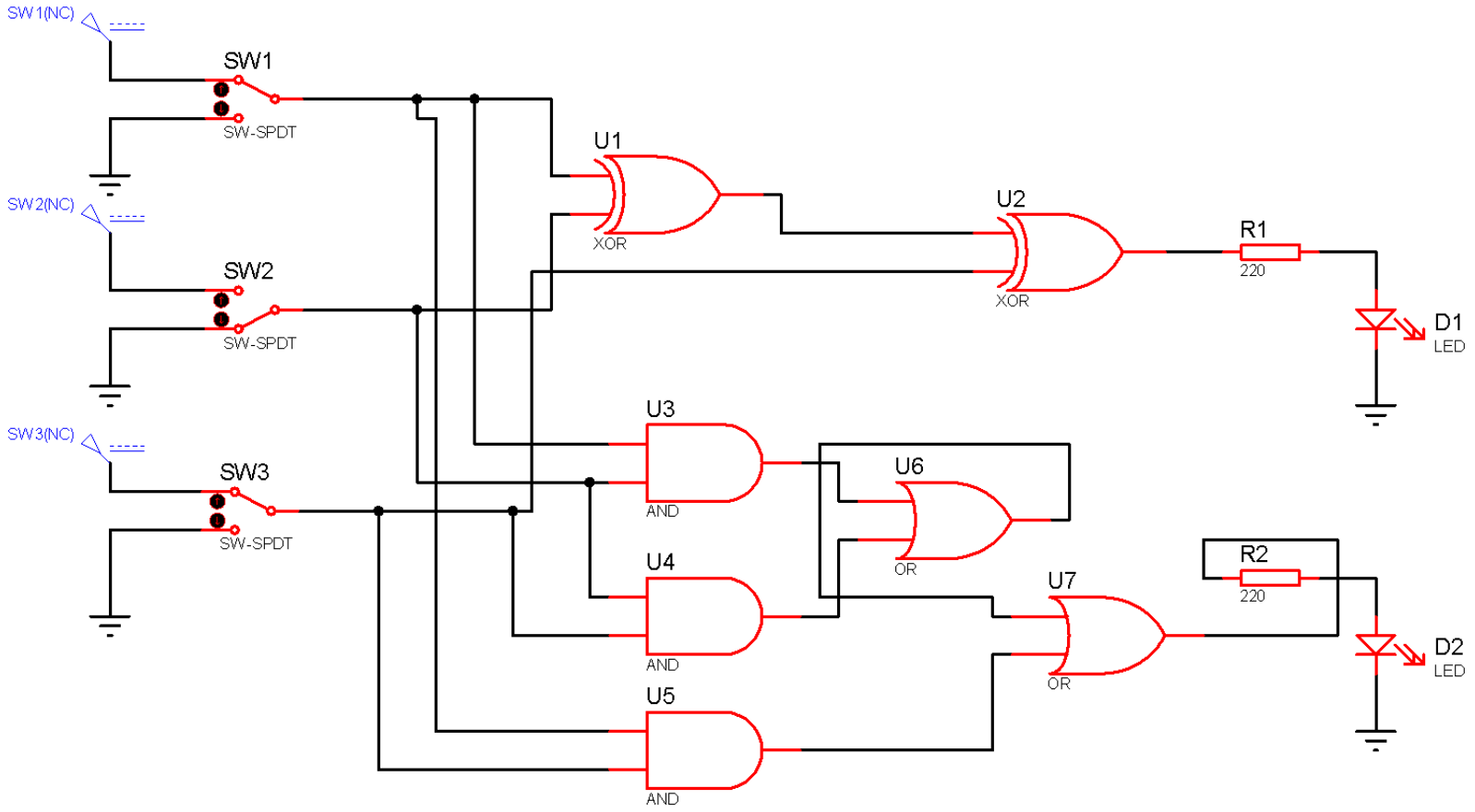


K-map for Carry (C_{out})

$$S = A\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + ABC + \overline{A}B\overline{C}$$
$$= C\,(AB + \overline{A}\,\overline{B}) + \overline{C}\,(\overline{A}B + A\,\overline{B})$$
$$= C\,\overline{(\overline{A}B + A\,\overline{B})} + \overline{C}\,(\overline{A}B + A\,\overline{B})$$
$$= C\,\overline{(A \oplus B)} + \overline{C}\,(A \oplus B) \;= A \oplus B \oplus C.$$

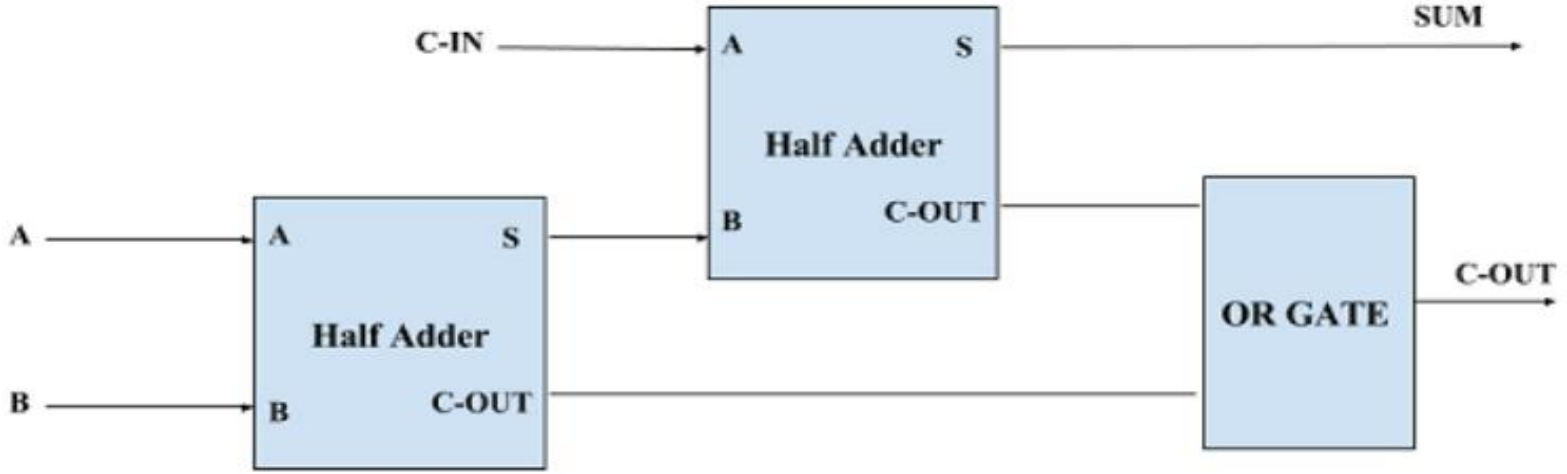
$$C_{out} = BC_{in} + AB + C_{in}A$$
$$= AB + BC_{in} + C_{in}A$$

Construction Of Binary Full Adder Using XOR, AND & OR Gate From Proteus Software:

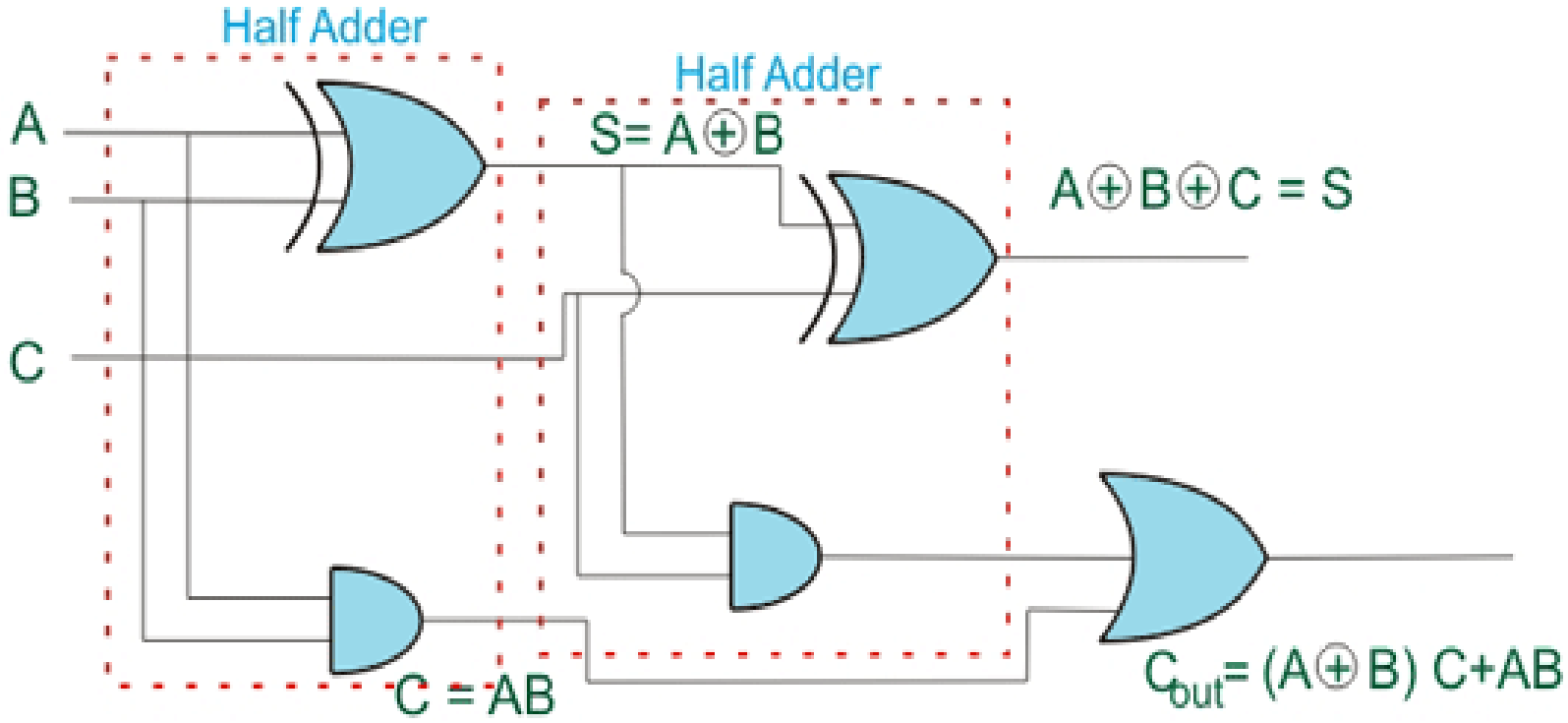


Implementation Of Binary Full Adder Using Two Half Adders:

- A full adder can be formed by logically connecting two half adders. The block diagram that shows the implementation of a full adder using two half adders is shown below:



Full adder using two Half Adders



Full adder using two HalfAdders Derivaions

- We know the equations for S and COUT from earlier calculations as

S = A'B' Cin + A' BC'in + ABCin

Cout = AB + ACin + Bcin

We can rewrite the equation for sum as follows.

S = A'B' Cin + A'BC' in + ABCin

= Cin (A' B' + AB) + C' in (A' B + A B')

Therefore S = CIN XOR (A XOR B)

= Cin (A X-NOR B) + C'in (A X-OR B)

= Cin XOR (A XOR B)

Cout is simplified as:

COUT = A B + A CIN + B CIN.

COUT = AB + A CIN + B CIN (A + A')

= ABCIN + AB + ACIN + A'B CIN

= AB (1 +CIN) + ACIN + A' B CIN

= A B + ACIN + A' B CIN

= AB + ACIN (B + B') + A' B CIN

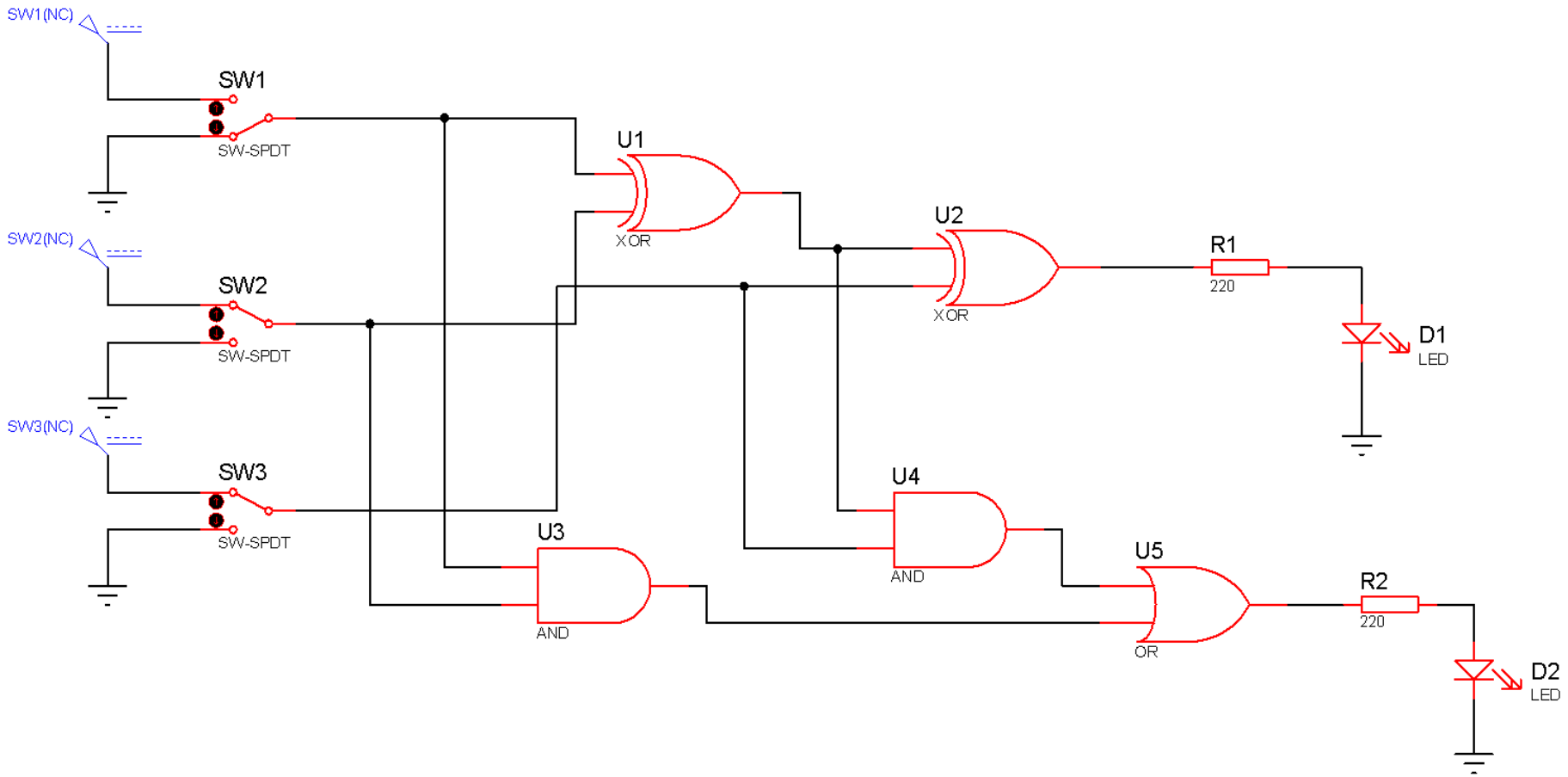
= ABCIN + AB + A'B CIN + A' B CIN

= AB (CIN + 1) + A'B CIN + A' B CIN

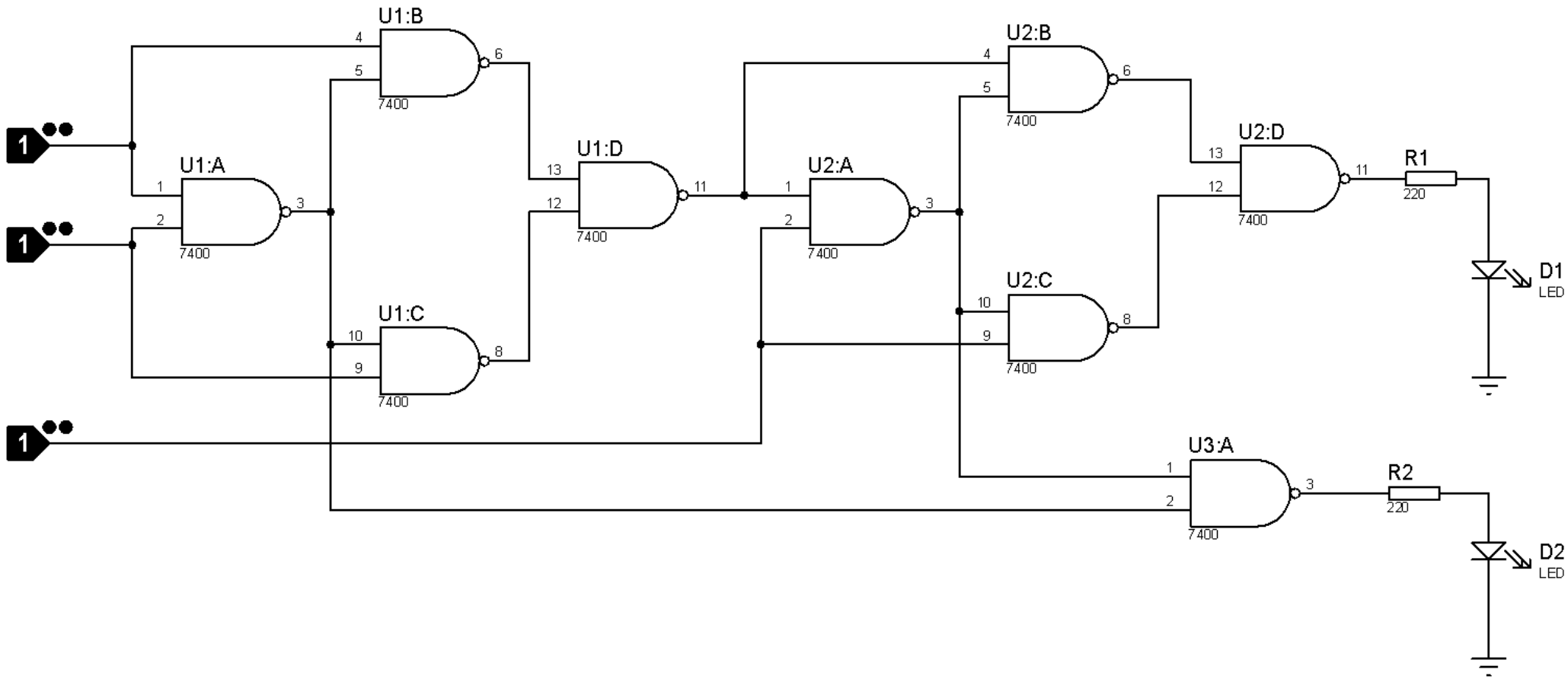
= AB + A'B CIN + A' B CIN

= AB + CIN (A'B + A'B)

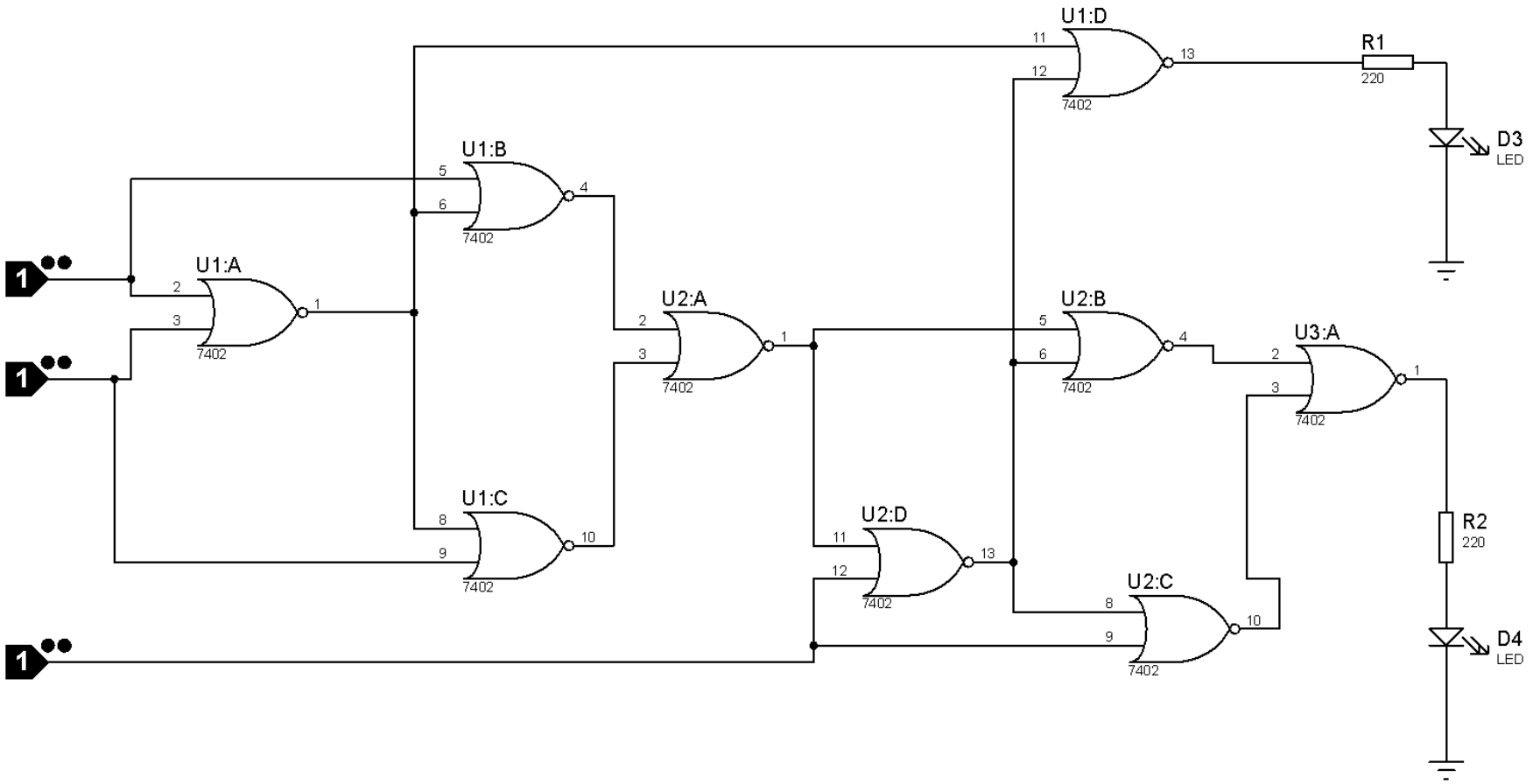
Construction Of Binary Full Adder Using Two Half Adders



Implementation Of Full Adder Using NAND Gates:



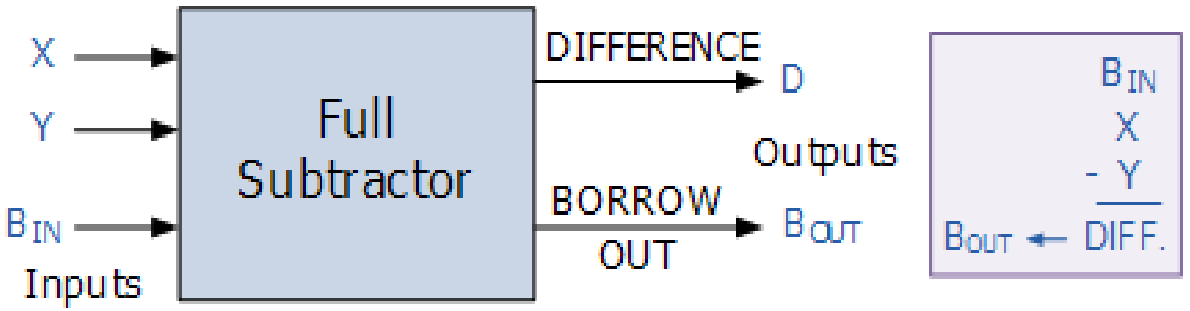
Implementation of Full Adder using NOR gates:



Full Subtractor

- A full subtractor is a combinational circuit that performs subtraction of two bits, one is minuend and other is subtrahend, taking into account borrow of the previous adjacent lower minuend bit. This circuit **has three inputs and two outputs**. The three inputs A, B and Bin, denote the minuend, subtrahend, and previous borrow, respectively. The two outputs, D and Bout represent the difference and output borrow, respectively.

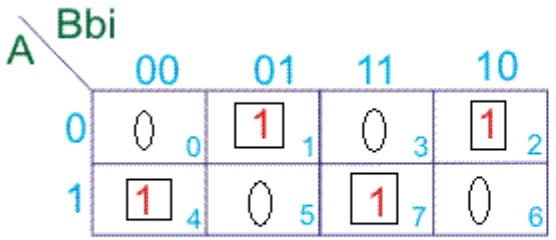
- The block diagram of half subtractor is shown below:



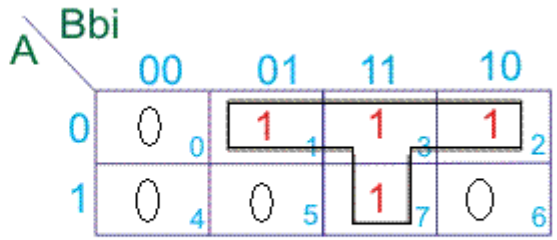
Full Subtractor Table

Sl No.	Minuend bit (A)	Subtrahend bit (B)	Input borrow (bi)	Difference (D)	Borrow (b)
0	0	0	0	0	0
1	0	0	1	1	1
2	0	1	0	1	1
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	0
6	1	1	0	0	0
7	1	1	1	1	1

K Map for full subtractor



K-map for D



K-map for b

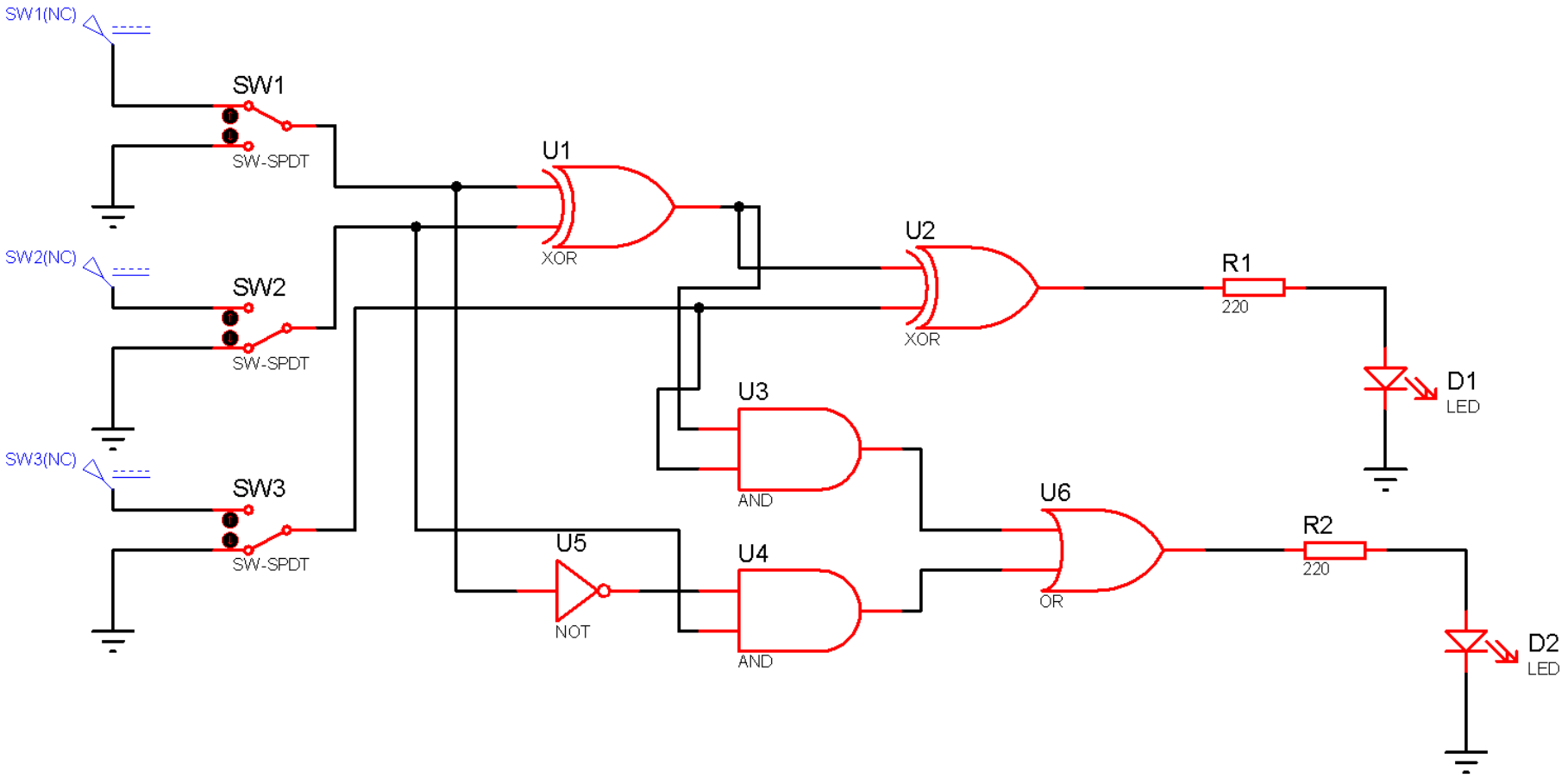
$$D = A\overline{B}\,\overline{bi} + ABbi + \overline{A}B\overline{bi} = \overline{bi}(A\overline{B} + \overline{A}B) + bi(\overline{A}\,\overline{B} + AB)$$

$$= \overline{bi}(A\overline{B} + \overline{A}B) + bi\overline{(A\overline{B} + \overline{A}B)} = bi \oplus A \oplus B = A \oplus B \oplus bi$$

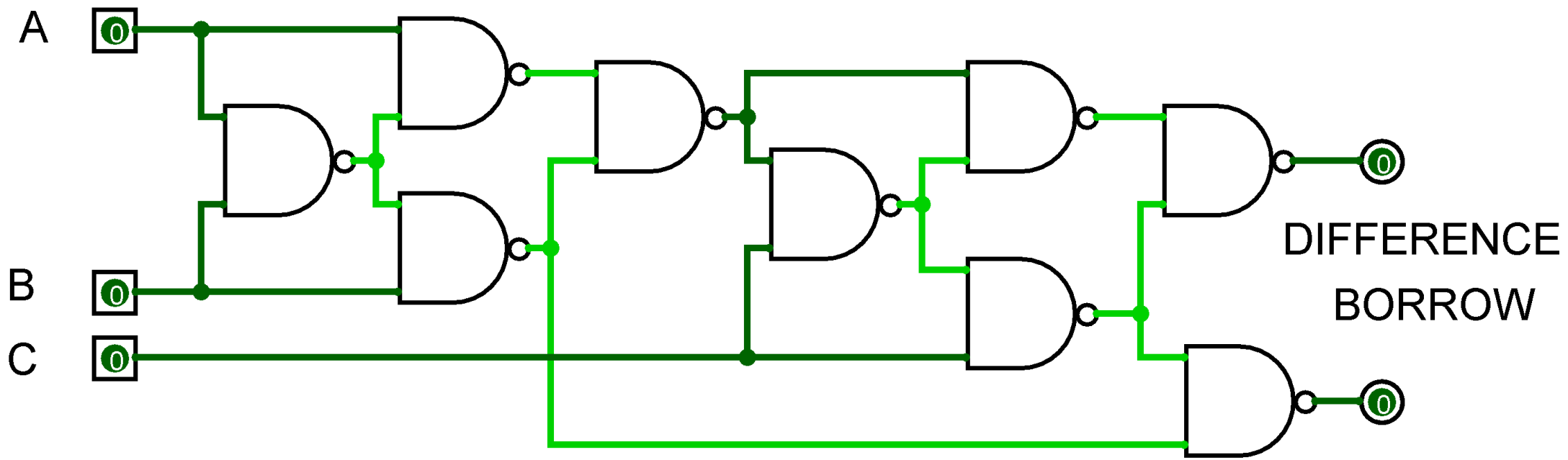
$$b = \overline{A}\,\overline{B}bi + \overline{A}Bbi + \overline{A}B\overline{bi} + ABbi = \overline{A}B\,(bi + \overline{bi}) + (AB + \overline{A}\,\overline{B})\,bi$$

$$= \overline{A}B + \overline{(A \oplus B)}\,bi$$

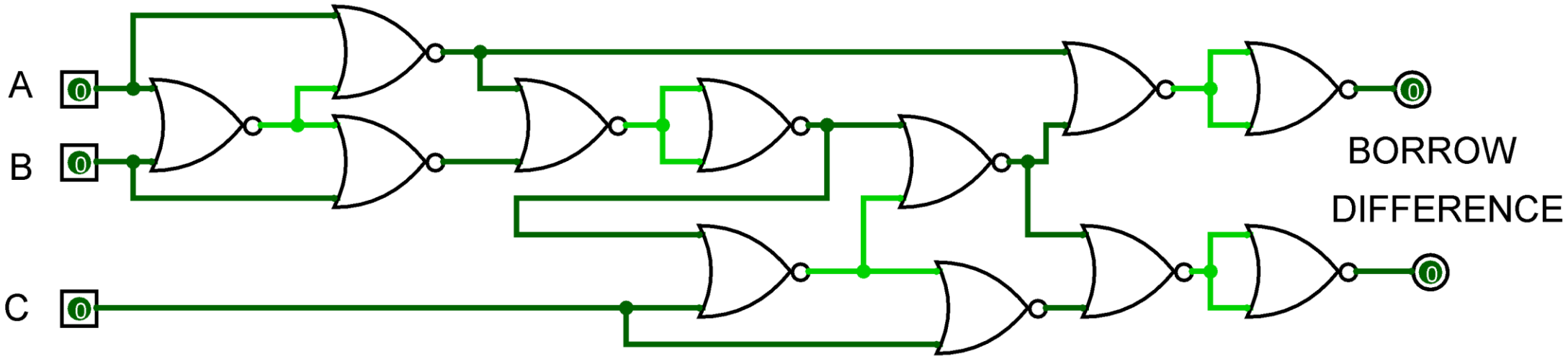
Construction Of Binary Full Subtractor Using XOR, AND, OR & NOT Gate



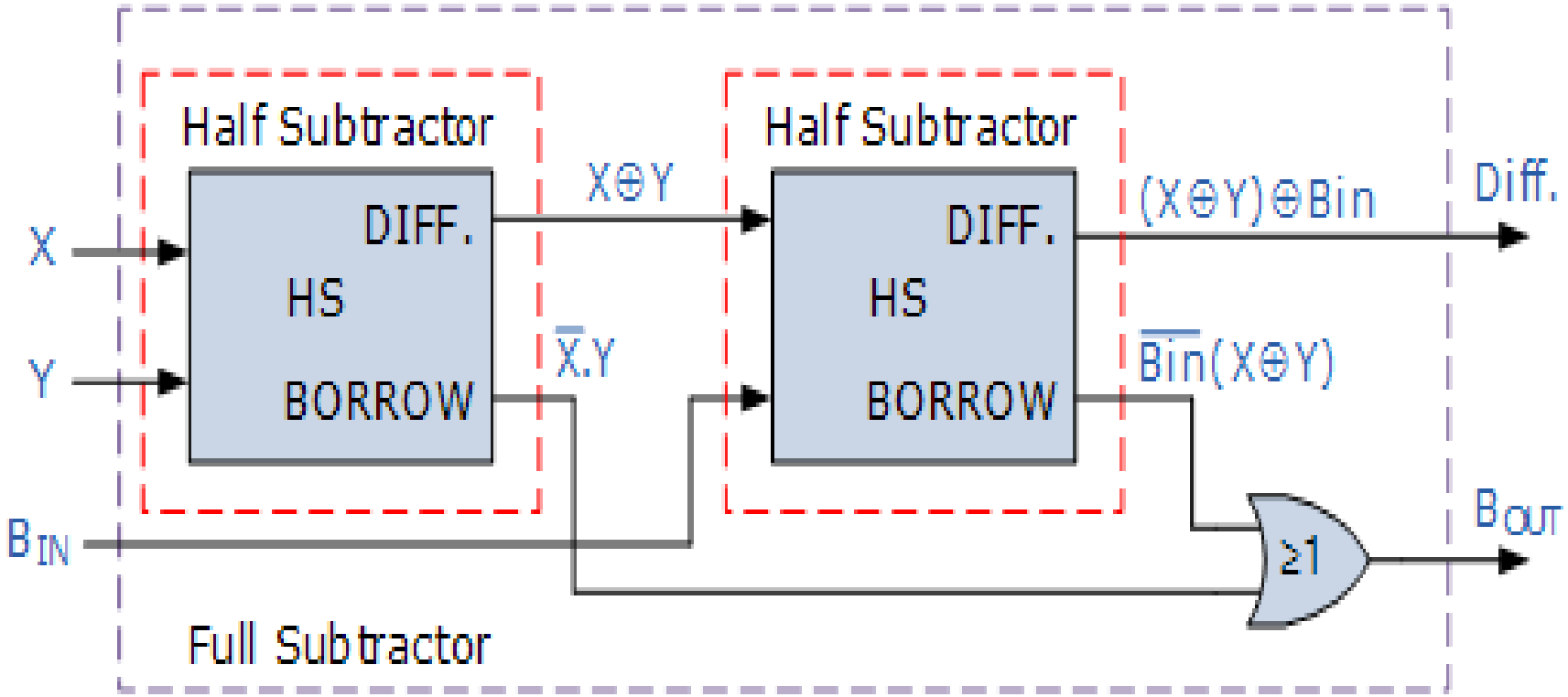
Implementation Of Full Subtractor Using NAND Gates:



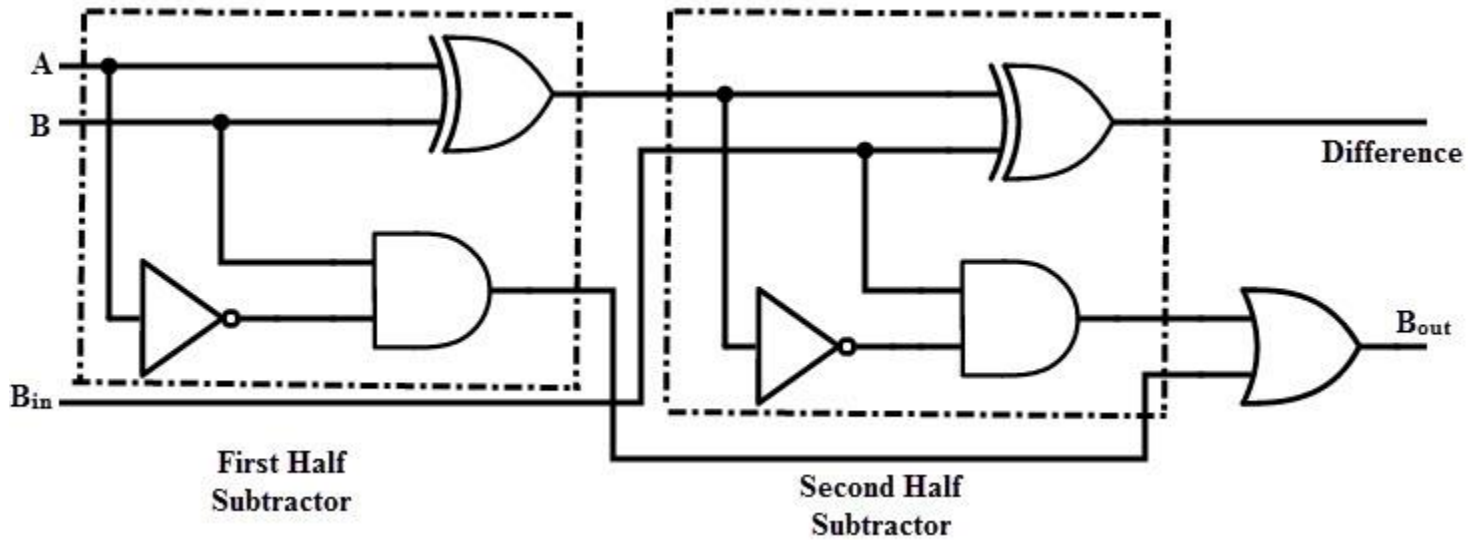
Implementation of Full Subtractor using NOR gates:



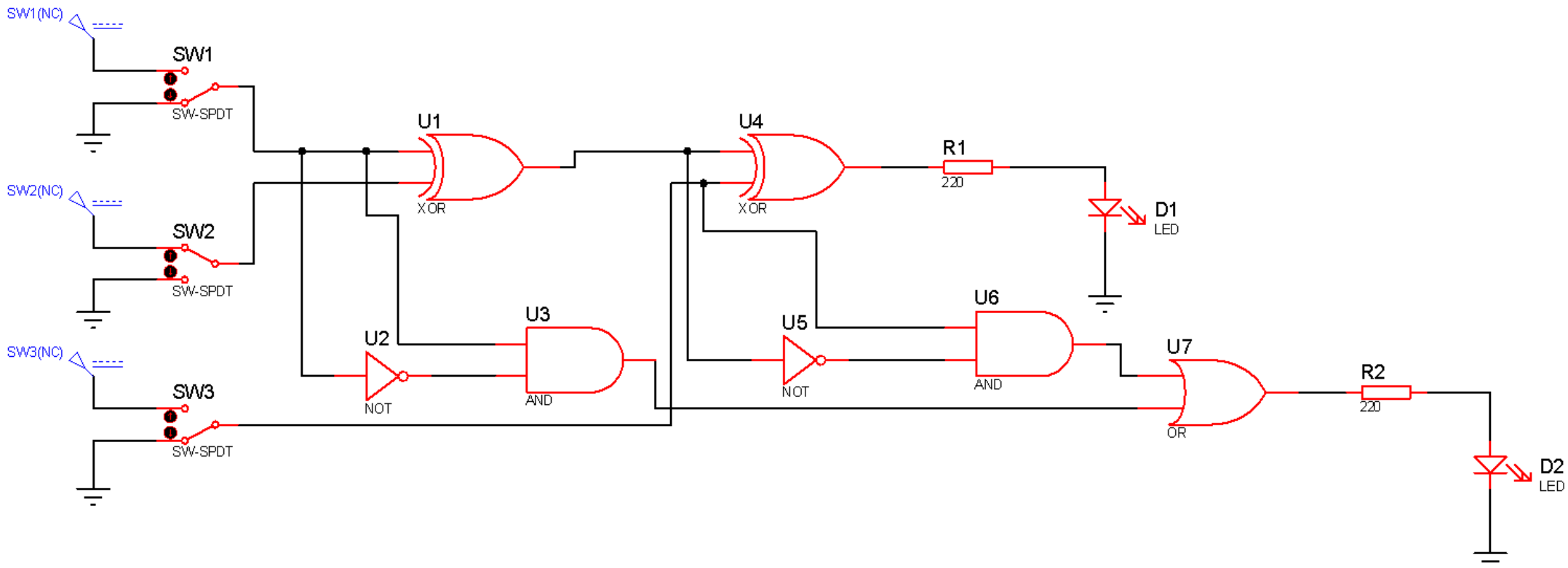
Implementation Of Binary Full Subtractor Using Two Half Subtractors:



Full subtractor using two H.S.



Practical diagram of Full Subtractor using two H.S.



Implementation Of Full Adder And Full Subtractor In One Circuit:

Dm	A	B	C	Y(s/d)	Y(c/b)
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	1	0
0	0	1	1	0	1
0	1	0	0	1	0
0	1	0	1	0	1
0	1	1	0	0	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	0	1	1	1
1	0	1	0	1	1
1	0	1	1	0	1
1	1	0	0	1	0
1	1	0	1	0	0
1	1	1	0	0	0
1	1	1	1	1	1

K Map for FA and FS in a circuit

DmA \ AB	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	1	0	1	0
10	0	1	0	1

K-MAP FOR Y(s/d)

DmA \ AB	00	01	11	10
00	0	0	1	0
01	0	1	1	1
11	0	0	1	0
10	0	1	1	1

K-MAP FOR Y(c/b)

From above k-maps, we get that:

- $Y(s/d) = AB'C' + A'B'C + ABC + A'BC'$

  $= AB'C' + ABC + A'B'C + A'BC$

  $= A (B'C' + BC) + A' (B'C + BC')$

  $= A (B \oplus C)' + A' (B \oplus C)$

  $= A \oplus B \oplus C$

$$Y(c/b) = BC + Dm'AC + Dm'AB + DmA'C + DmA'B$$

$$= BC + (Dm'AC + DmA'C + Dm'AB + DmA'B)$$

$$= BC + [C (Dm'A + DmA') + B (Dm'A + DmA')]$$

$$= BC + [C (Dm \oplus A) + B (Dm \oplus A)]$$

$$= BC + [(Dm \oplus A) (B+C)]$$

Construction Of Full Adder & Full Subtractor In One Circuit Using Proteus Software: