CSE1003 Digital Logic and Design
Module 3
Combinational Circuits I
L1

Dr. S.Hemamalini

Professor

School of Electrical Engineering

VIT Chennai

Contents

4 hrs

- Adder
- Subtractor
- Code converter
- Analyzing a combinational circuit

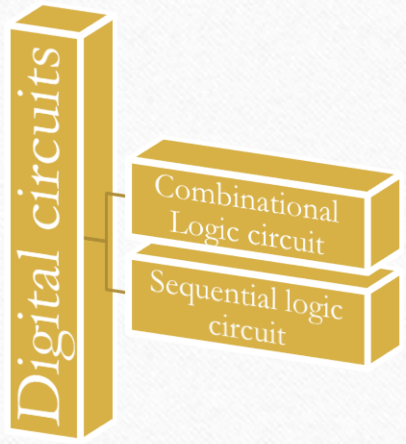
Digital Circuits

Combinational Logic Circuits

- Output at any instant of time depends only on the inputs present at that instant of time.
- The logic gate is the most basic building block of combinational logic.
- Combinational logic circuits do not have memory elements (storage device).
- It can be designed using gates or available ICs.
- Adder, subtractor, ALU comparators, parity generator and checker, multiplexer, demultiplexer, encoder, and code converters are the examples of combinational logic circuits.

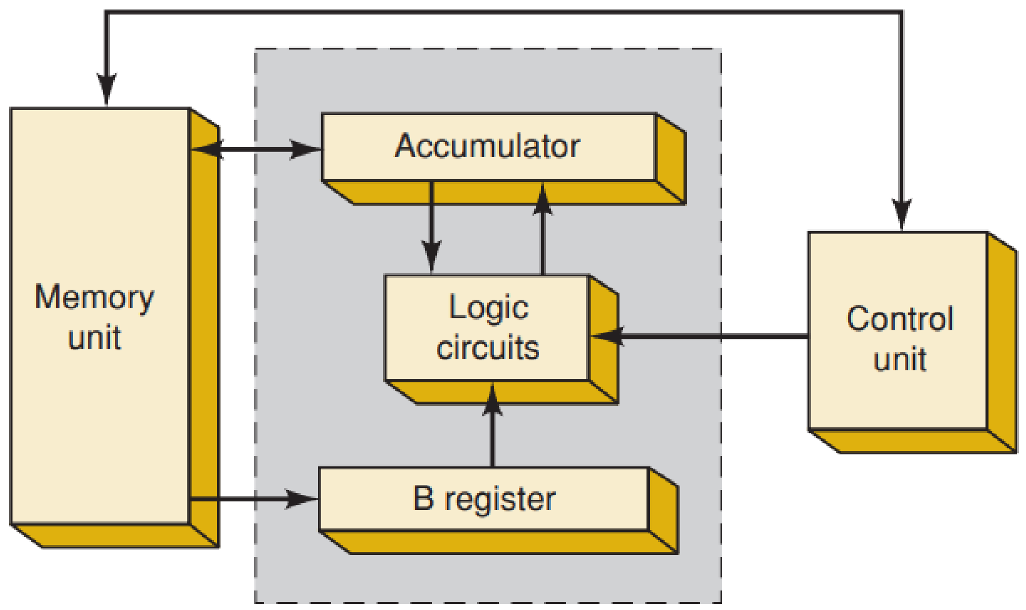
Sequential Logic Circuits

- The output depends upon not only the present but also the past state of inputs.
- Sequential circuits comprise both logic gates and memory elements such as flip-flops.



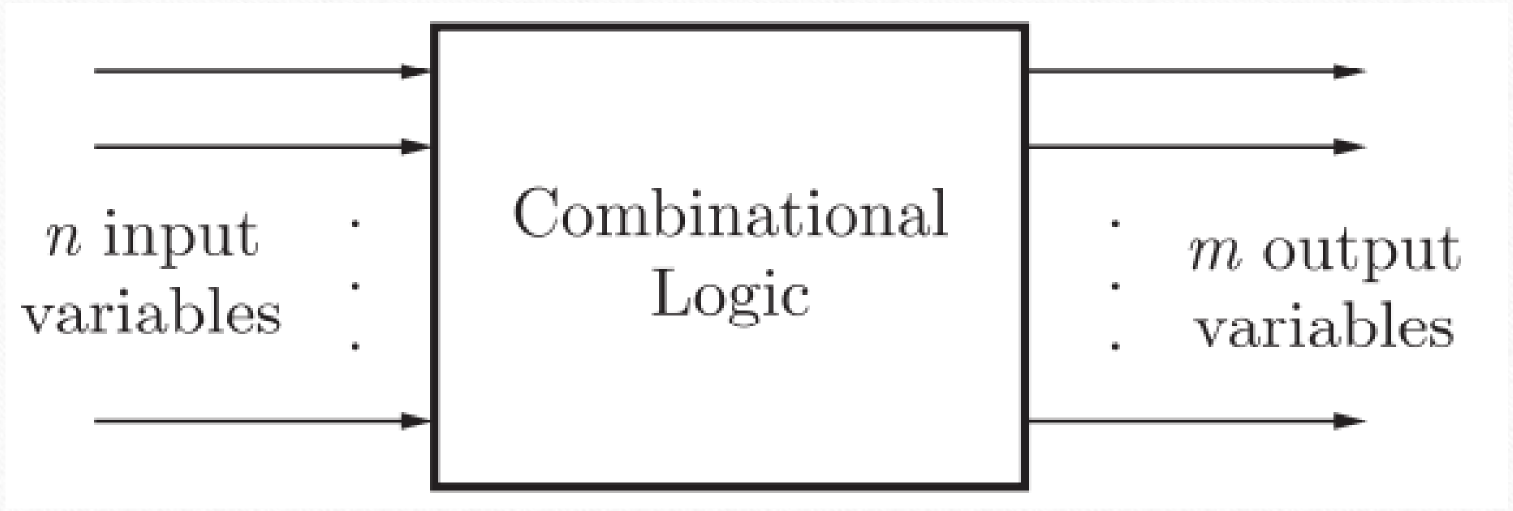
ARITHMETIC CIRCUITS
Arithmetic/Logic Unit

- All arithmetic operations take place in the arithmetic/logic unit (ALU) of a computer.
- The arithmetic/logic unit contains at least two flip-flop registers: the B register and the accumulator register.
- It also contains combinational logic, which performs the arithmetic and logic operations on the binary numbers that are stored in the B register and the accumulator.



Arithmetic/logic unit

Block diagram of a combinational circuit



DESIGN PROCEDURE

Any combinational circuit can be designed by the following steps of design procedure.

1. The problem is stated.
2. Identify the input variables and output functions.
3. The input and output variables are assigned letter symbols.
4. The truth table is prepared that completely defines the relationship between the input variables and output functions.
5. The simplified Boolean expression is obtained by any method of minimization—algebraic method, Karnaugh map method, or tabulation method.
6. A logic diagram is realized from the simplified expression using logic gates.

Design Constraints

(1) minimum number of gates

(2) Minimum number of outputs

(3) minimum propagation time of the signal through a circuit

(4) minimum number of interconnections

(5) limitations of the driving capabilities of each logic gate

ADDERS

- The most basic arithmetic operation is the addition of two binary digits.

$$0 + 0 = 0 \qquad 0 + 1 = 1 \qquad 1 + 0 = 0 \qquad 1 + 1 = \underset{\text{Carry}}{1}\,\underset{\text{Sum}}{0}$$

- A combinational circuit that performs the addition of two 1-bit numbers is called as **half-adder**.
- The logic circuit that adds three 1-bit numbers is called as **full-adder**.

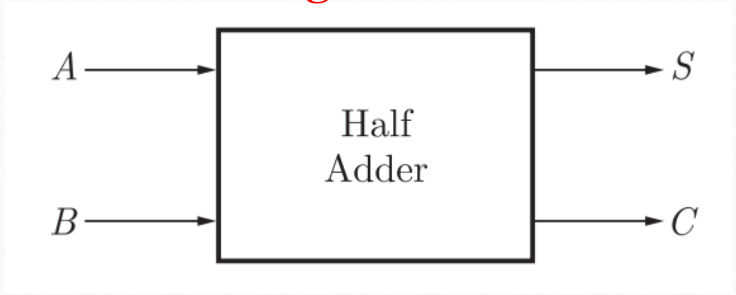
Half-Adder

- The logic circuit that performs the addition of two 1-bit numbers is called as half-adder.
- It is the basic building block for addition of two single bit numbers.
- This circuit has two outputs namely Carry (C) and Sum (S).

Truth table for half-adder

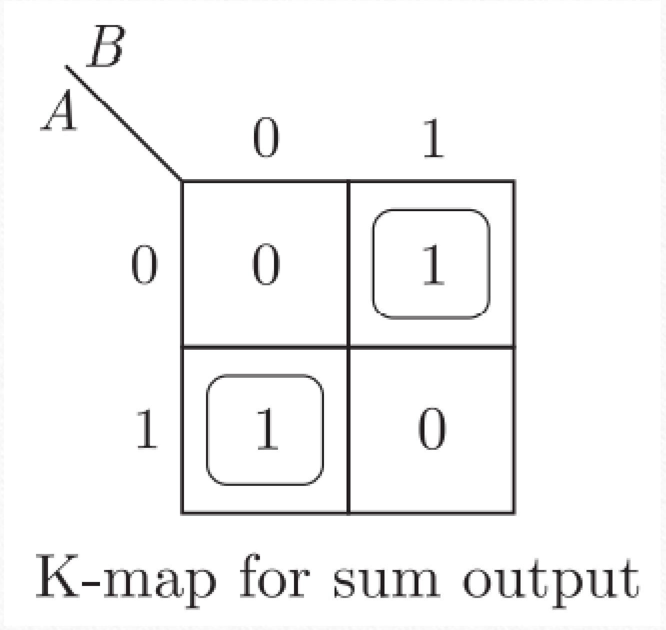
Inputs		Outputs	
A	B	Sum (S)	Carry (C)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Block diagram of a half-adder



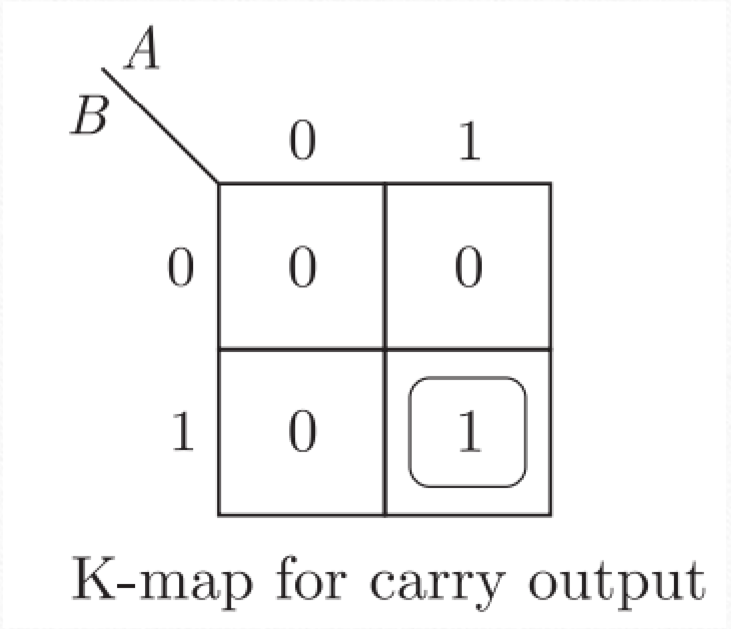
Half-Adder

- **K-map Simplification for Carry and Sum**



K-map for sum output

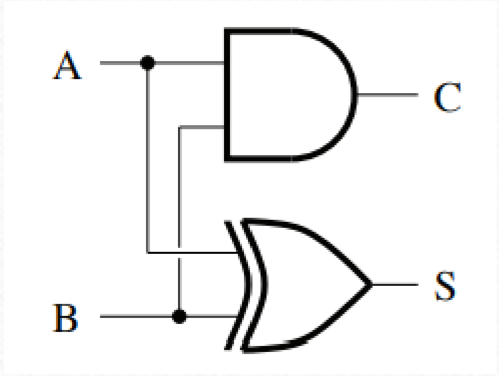
$$S = \overline{A}B + A\overline{B} = (A \oplus B)$$



K-map for carry output

$C = A.B$

Logic circuit



Logic diagram of half-adder using only 2-input NOR gates

Sum, $S = A\overline{B} + \overline{A}B = A\overline{B} + A\overline{A} + \overline{A}B + B\overline{B}$

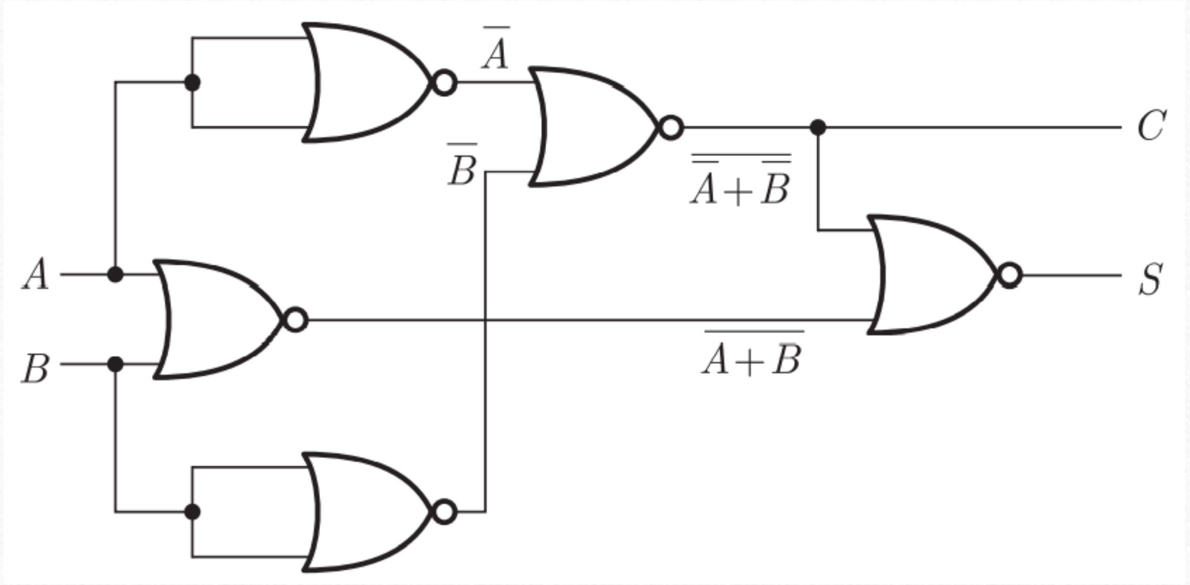
$$= A(\overline{A} + \overline{B}) + B(\overline{A} + \overline{B})$$

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

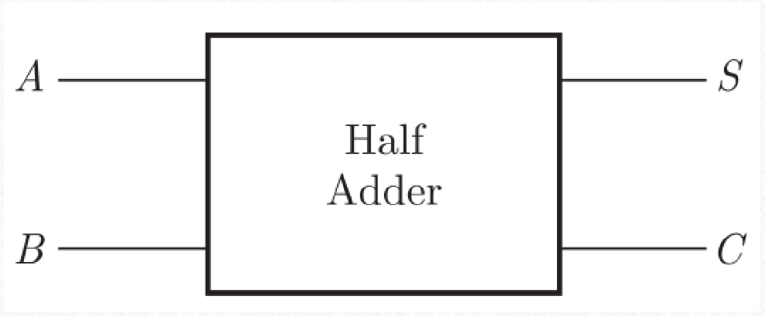
$$= \overline{\overline{(A + B)(\overline{A} + \overline{B})}}$$

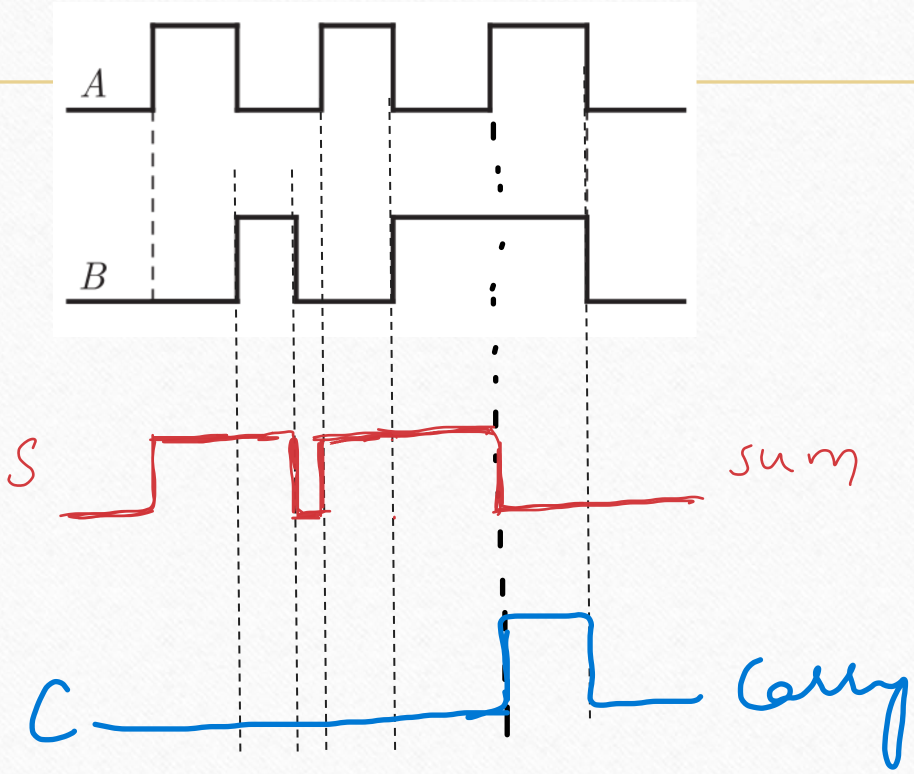
$$= \overline{\overline{A + B} + \overline{\overline{A} + \overline{B}}}$$

Carry, $C = AB = \overline{\overline{AB}} = \overline{\overline{A} + \overline{B}}$



For the half-adder circuit of Figure (a), the inputs applied at A and B are as shown in Figure (b). Plot the corresponding SUM and CARRY outputs for the half-adder.

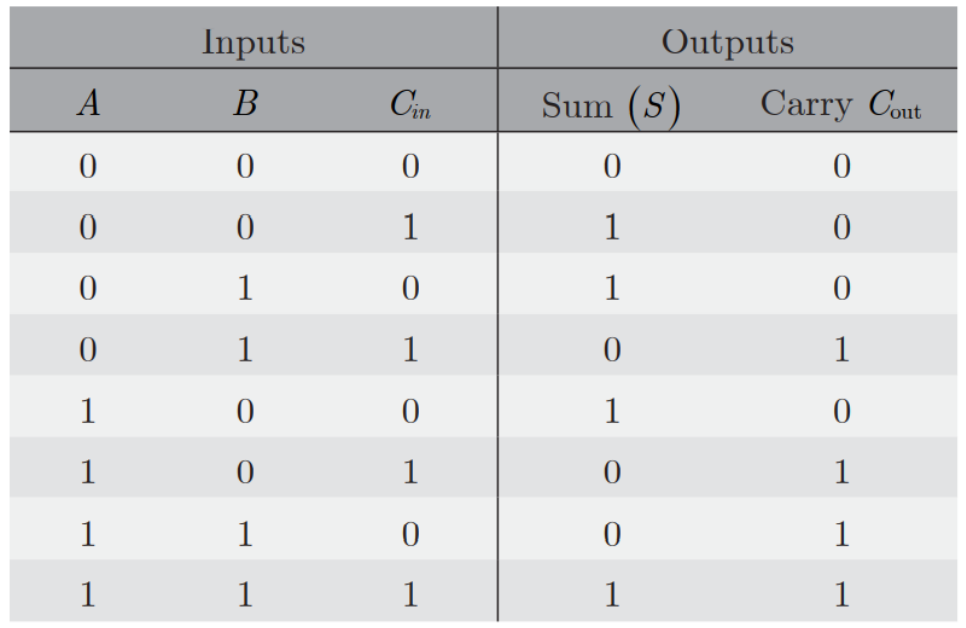




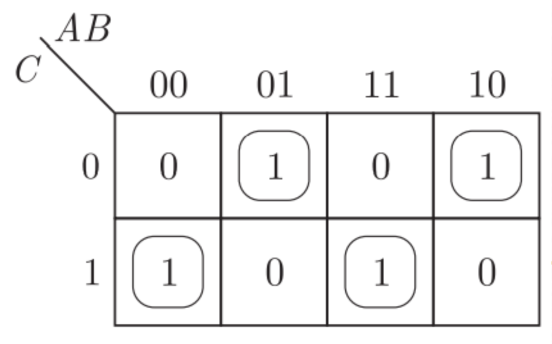
Full-Adder

- A half-adder has two 1-bit inputs and there is no provision to add a carry which could have been generated from lower bit order additions. This limitation of half-adder is overcome in full-adder.
- A full adder circuit is an arithmetic circuit block that can be used to add three bits to produce a sum and a carry output.

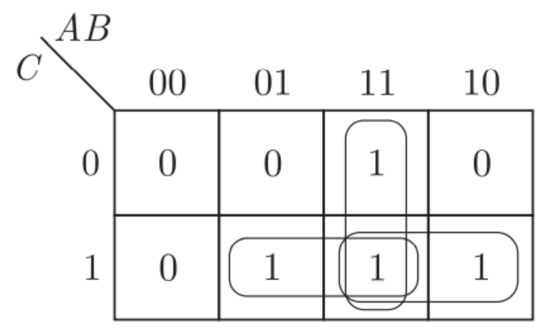
Full-Adder



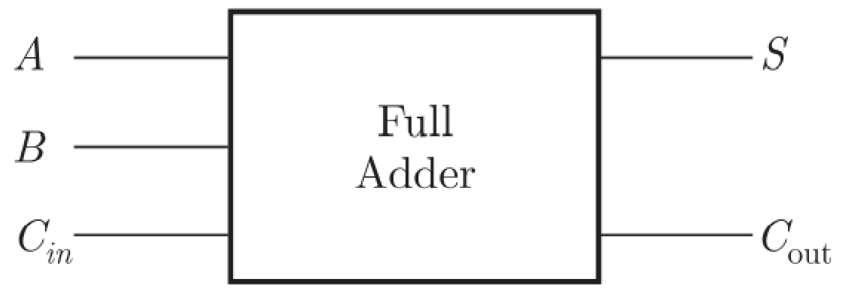
Inputs			Outputs	
A	B	C_{in}	Sum (S)	Carry C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



K-map for sum S



K-map for C_{out}

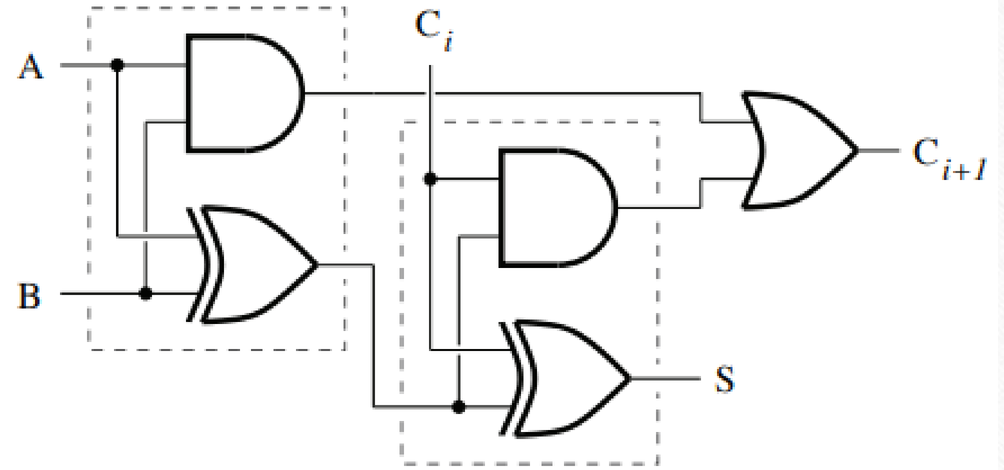


Sum,
$$\begin{aligned} S &= \overline{A}\,\overline{B}C_{in} + \overline{A}B\overline{C}_{in} + ABC_{in} + A\overline{B}\,\overline{C}_{in} \\ &= C_{in}(\overline{A}\,\overline{B} + AB) + \overline{C}_{in}(\overline{A}B + A\overline{B}) \\ &= C_{in}(A \odot B) + \overline{C}_{in}(A \oplus B) \\ &= C_{in}(\overline{A \oplus B}) + \overline{C}_{in}(A \oplus B) \\ S &= C_{in} \oplus A \oplus B \end{aligned}$$

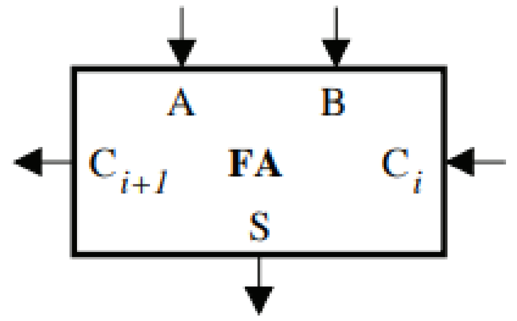
Carry, $C_{out} = AB + AC_{in} + BC_{in}$

Full-Adder

The full-adder circuit can be formed with two half-adders and one OR gate.

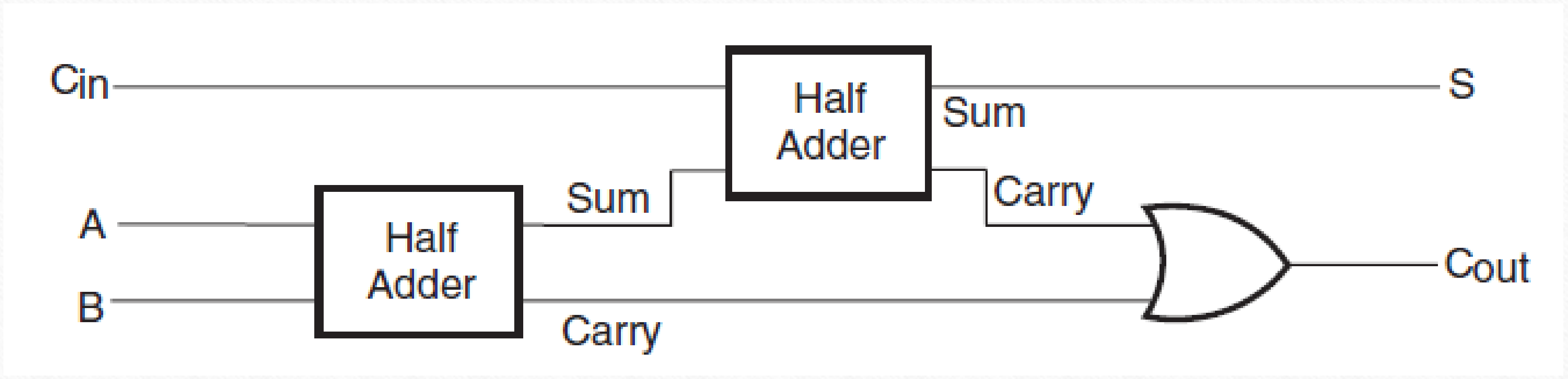


Logic circuit

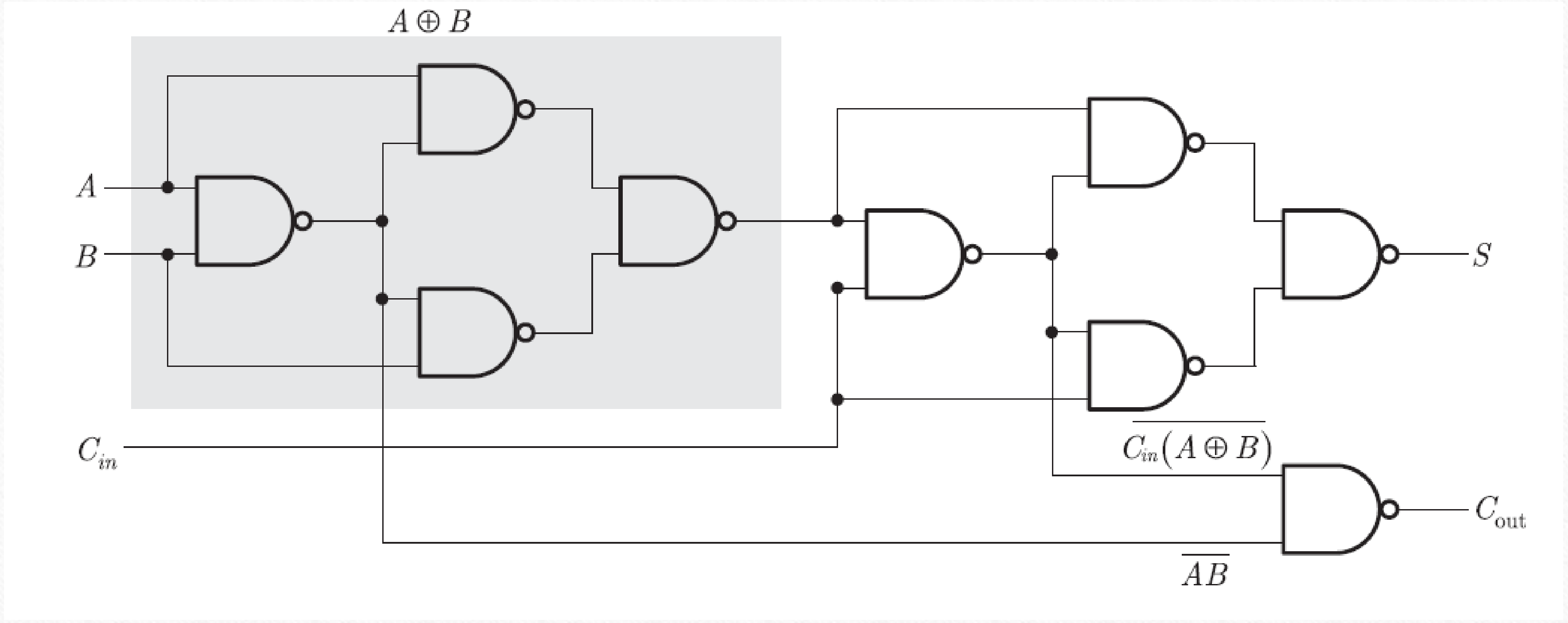


Symbol

Full-Adder



Full-Adder using NAND Gates



Logic diagram of a full-adder using only 2-input NAND gates

SUBTRACTORS

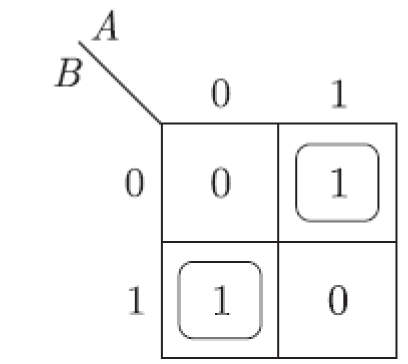
- The logic circuit of subtraction of two 1-bit numbers is called as **half-subtractor.**
- The logic circuit, which performs the subtraction of two 1-bit numbers, taking into account the borrow of the pervious stage, is called as **full-subtractor.**

$$0 - 0 = 0$$
$$1 - 0 = 1$$
$$1 - 1 = 0$$
$$0 - 1 = 1 \text{ (with borrow 1)}$$

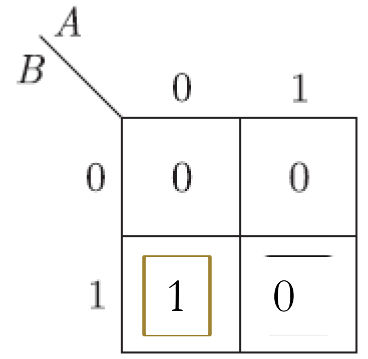
Half-Subtractor

Truth Table of Half-subtractor

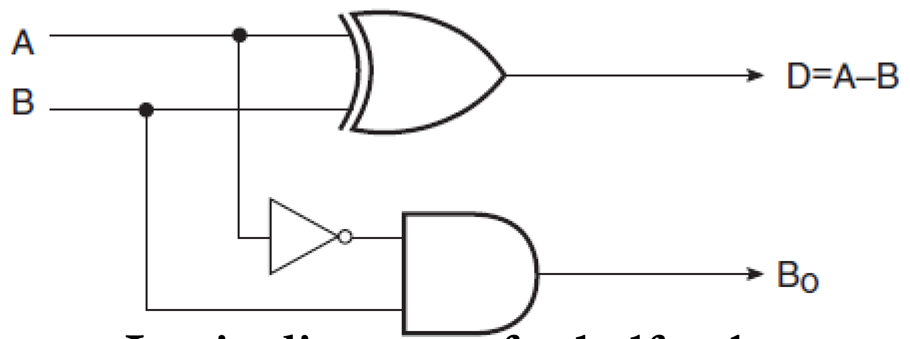
Inputs		Outputs	
A	B	D	B_{out}
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0



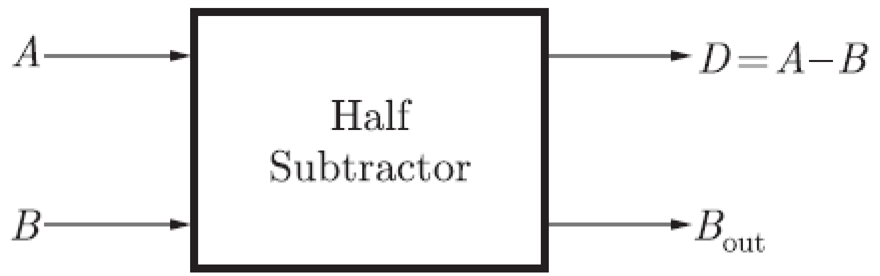
K-map for difference output



K-map for borrow output



Logic diagram of a half-subtractor

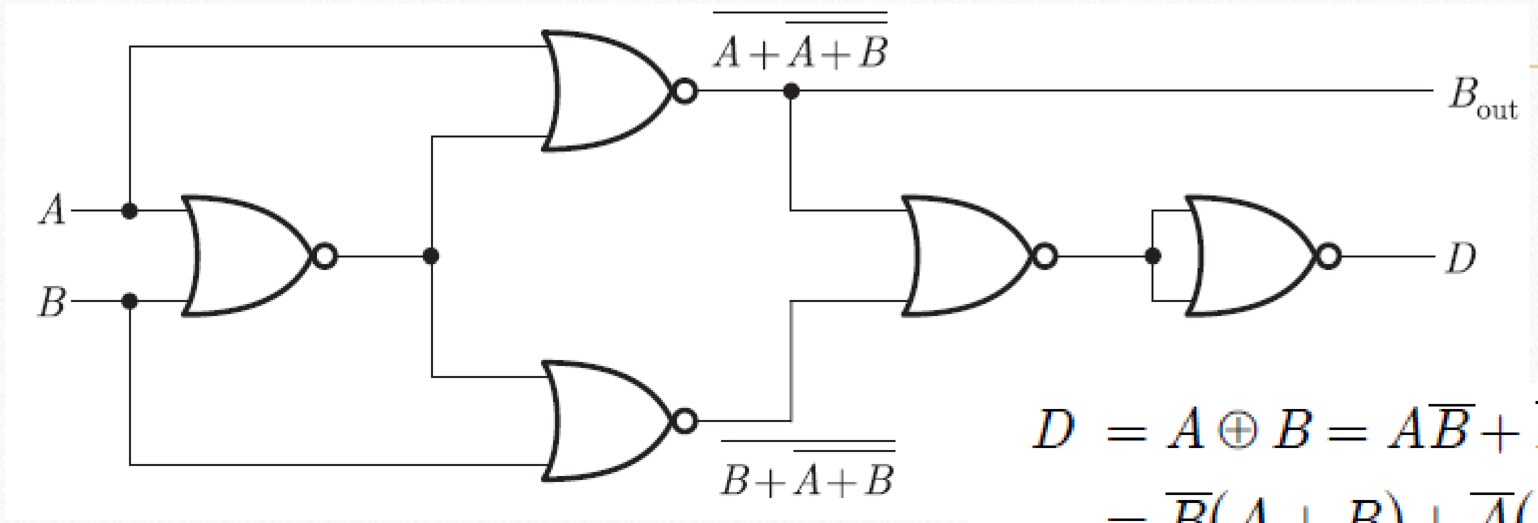


Block diagram of a half-subtractor

$$D = \overline{A}.B + A.\overline{B}$$

$$B_o = \overline{A}.B$$

Half-Subtractor using NOR Gates



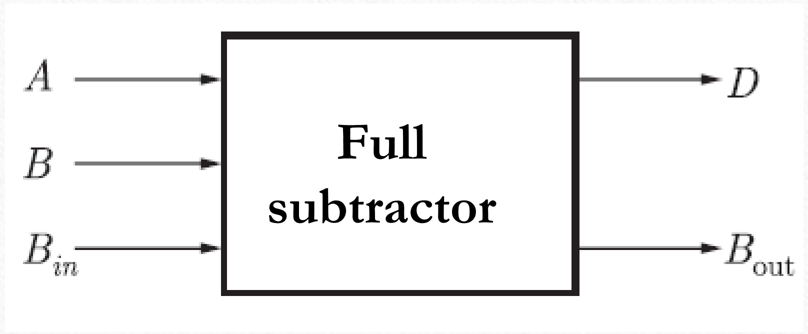
Logic diagram of a half-subtractor using only 2-input NOR gates

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

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

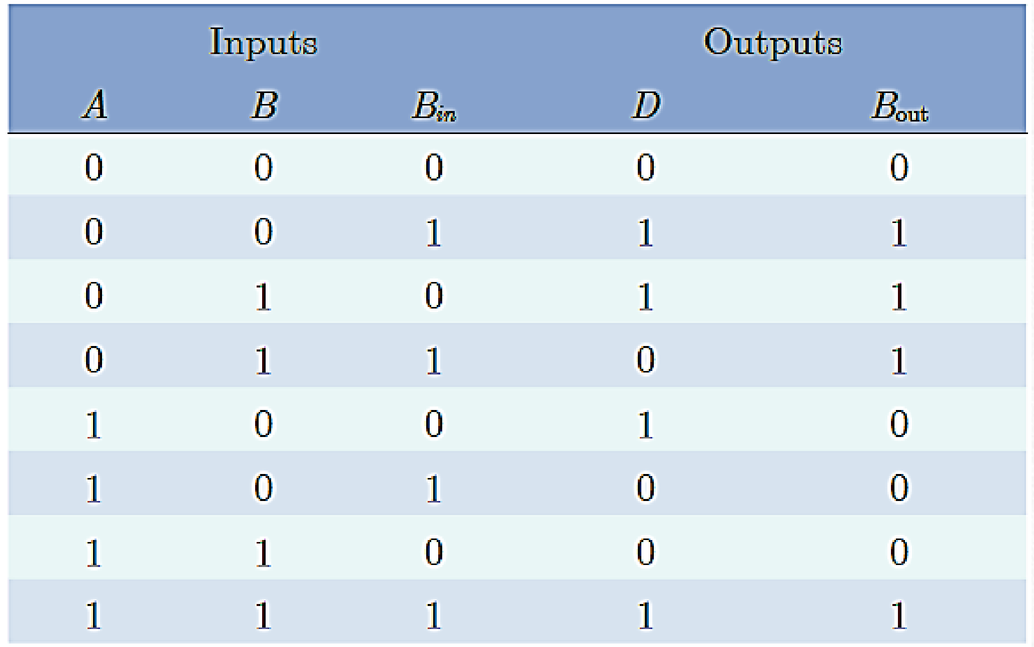
Full-Subtractor

- A full subtractor performs subtraction operation on two bits, a minuend and a subtrahend.
- It also takes into consideration whether a ‘1’ has already been borrowed by the previous adjacent lower minuend bit or not.
- Therefore, there are three input bits, namely the two bits to be subtracted and a borrow bit designated as *Bin* .
- There are two outputs, namely the difference output *D* and the borrow output *B*out.
- The BORROW output bit tells whether the minuend bit needs to borrow a ‘1’ from the next possible higher minuend bit.

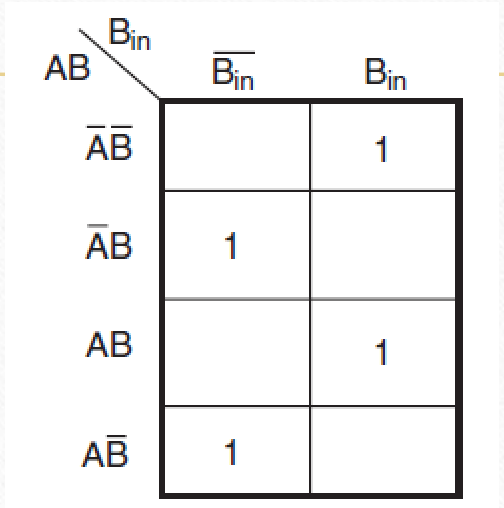


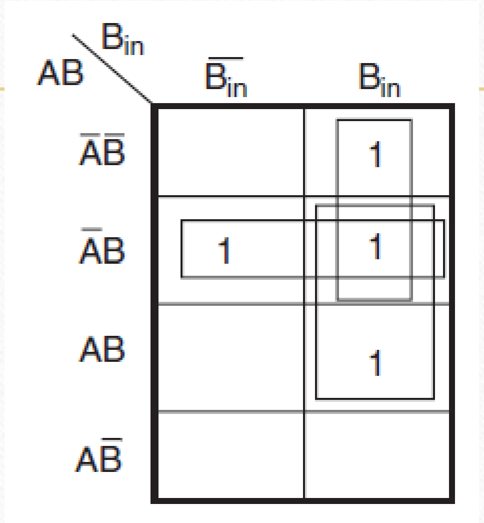
Full-Subtractor

Truth table of Full-Subtractor



Inputs			Outputs	
A	B	B_{in}	D	B_{out}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1





Karnaugh maps for difference and borrow outputs

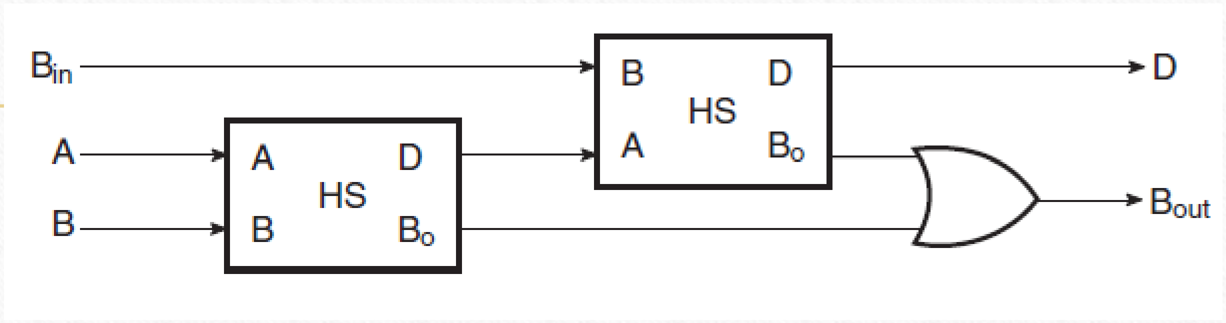
$$D = \overline{A}.\overline{B}.B_{\text{in}} + \overline{A}.B.\overline{B}_{\text{in}} + A.\overline{B}.\overline{B}_{\text{in}} + A.B.B_{\text{in}}$$

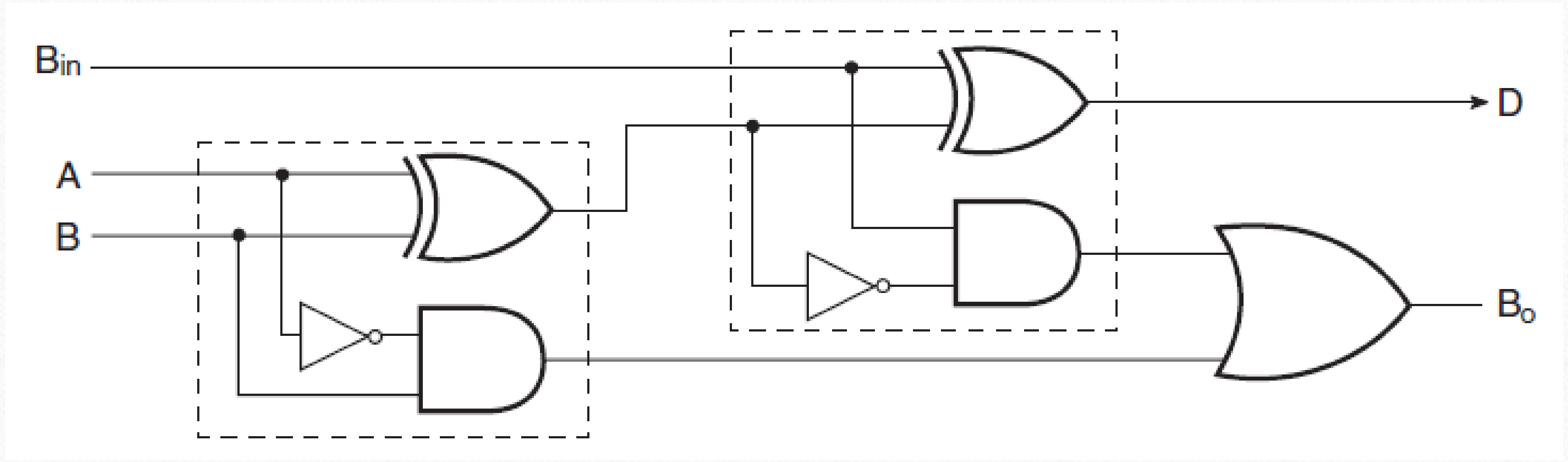
$$B_{\text{o}} = \overline{A}.\overline{B}.B_{\text{in}} + \overline{A}.B.\overline{B}_{\text{in}} + \overline{A}.B.B_{\text{in}} + A.B.B_{\text{in}}$$

$$B_{\text{o}} = \overline{A}.B + \overline{A}.B_{\text{in}} + B.B_{\text{in}}$$

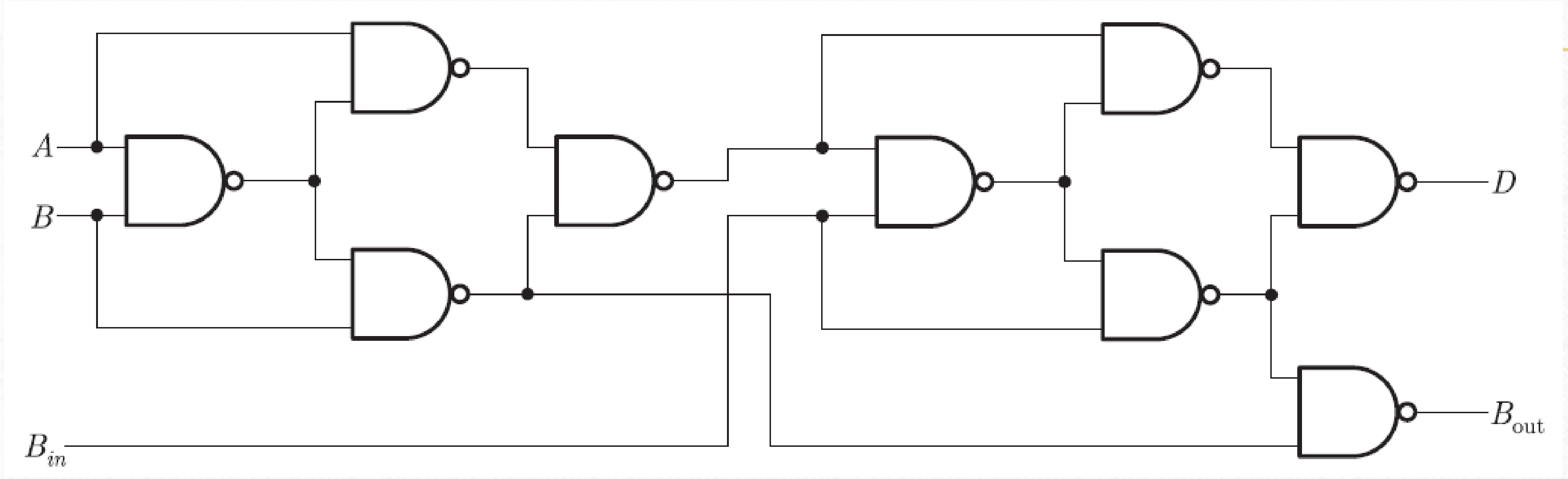
Full-Subtractor

A full subtractor using two half-subtractors





Full Subtractor Using NAND Gates



Difference,

$$D = A \oplus B \oplus B_{in} = \overline{\overline{(A \oplus B) \oplus B_{in}}}$$

$$= \overline{\overline{(A \oplus B)\overline{(A \oplus B) B_{in}}} \cdot \overline{B_{in}\overline{(A \oplus B) B_{in}}}}$$

where,

$$A \oplus B = \overline{\overline{A \cdot \overline{AB}} \cdot \overline{B \cdot \overline{AB}}}$$

Borrow,

$$B_{\text{out}} = \overline{A}B + B_{in}\overline{(A \oplus B)} = \overline{\overline{\overline{A}B + B_{in}\overline{(A \oplus B)}}}$$

$$= \overline{\overline{\overline{A}B} \cdot \overline{B_{in}\overline{(A \oplus B)}}}$$

$$= \overline{\overline{B(\overline{A} + \overline{B})} \cdot \overline{B_{in}(\overline{B}_{in} + (\overline{A \oplus B}))}}$$

$$B_{\text{out}} = \overline{\left[\overline{B \cdot \overline{AB}}\right] \cdot \left[\overline{B_{in}\left\{\overline{B_{in} \cdot (A \oplus B)}\right\}}\right]}$$