Premier University, Chittagong

EEE-2201: Digital Electronics and Pulse Technique

Analysis & Synthesis of Digital Logic Circuits

Dr. Sampad Ghosh

Associate Professor

Department of Electrical and Electronic Engineering

Faculty of Electrical and Computer Engineering

Chittagong University of Engineering and Technology (CUET)

December 14, 2024

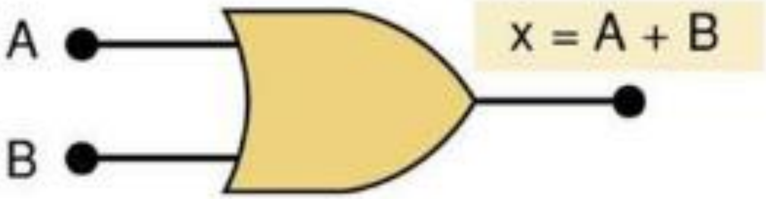
Topics	Reference
Analysis & synthesis of digital logic circuits	
Basic logic functions OR gates, AND gates, NOR gates, Describing logic circuits algebraically, Evaluating logic circuit outputs, Implementing circuits from Boolean expressions, NOR gates & NAND gates	*Tocci*

OR and AND Gates

Truth table and circuit symbol for a 2-input OR and AND gate

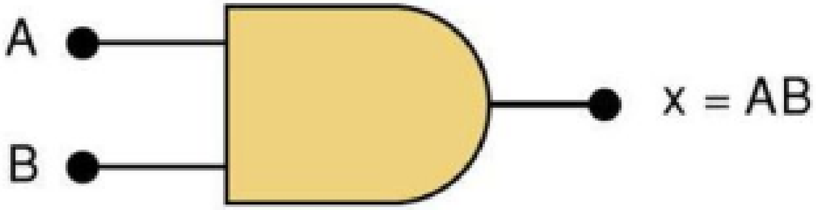
OR

A	B	x = A + B
0	0	0
0	1	1
1	0	1
1	1	1



OR Gate

A	B	x = A · B
0	0	0
0	1	0
1	0	0
1	1	1



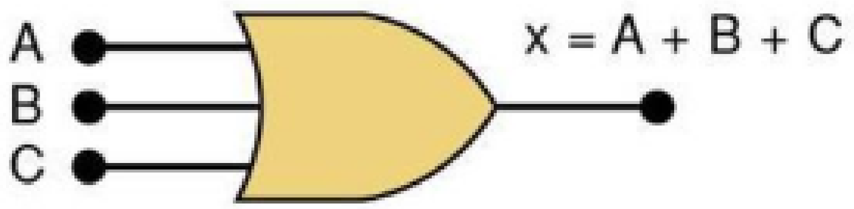
AND gate

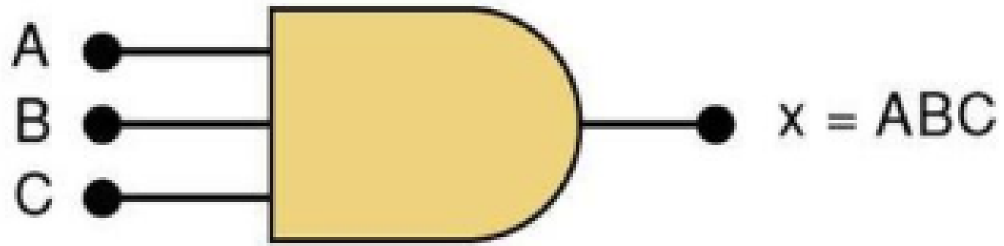
OR and AND Gates...

Truth table and circuit symbol for a 3-input OR and AND gate

A	B	C	x = A + B + C
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

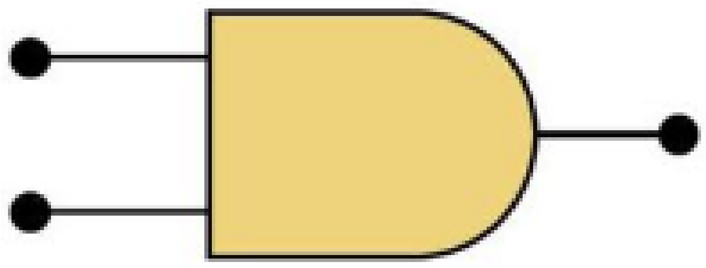
A	B	C	x = ABC
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1





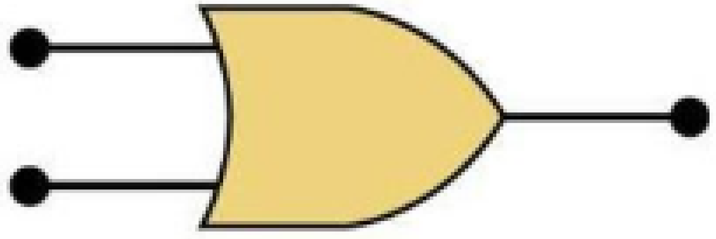
AND / OR

- The AND symbol on a logic-circuit diagram tells you output will go HIGH only when **all inputs** are HIGH.



Input **Output**

- The OR symbol means the output will go HIGH when **any input** is HIGH.

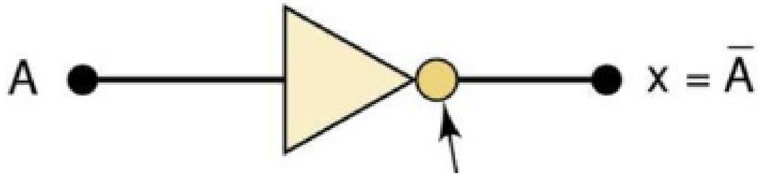


NOT Operation with NOT Gates

- Boolean expression for NOT operation
  - $\mathbf{x = A' = \bar{A}}$ - *read as x equals "NOT A" or "inverse of A" or "complement of A"*
- The NOT gate is an electronic circuit that produces an inverted version of the input at its output.
  - More commonly called an *Inverter*.

Truth table and circuit symbol for NOT gate

A	$x = \bar{A}$
0	1
1	0



Presence of small circle always denotes inversion

Boolean operations

- Summarized rules for OR, AND and NOT.

OR	AND	NOT
$0 + 0 = 0$	$0 \cdot 0 = 0$	$\overline{0} = 1$
$0 + 1 = 1$	$0 \cdot 1 = 0$	$\overline{1} = 0$
$1 + 0 = 1$	$1 \cdot 0 = 0$	
$1 + 1 = 1$	$1 \cdot 1 = 1$	

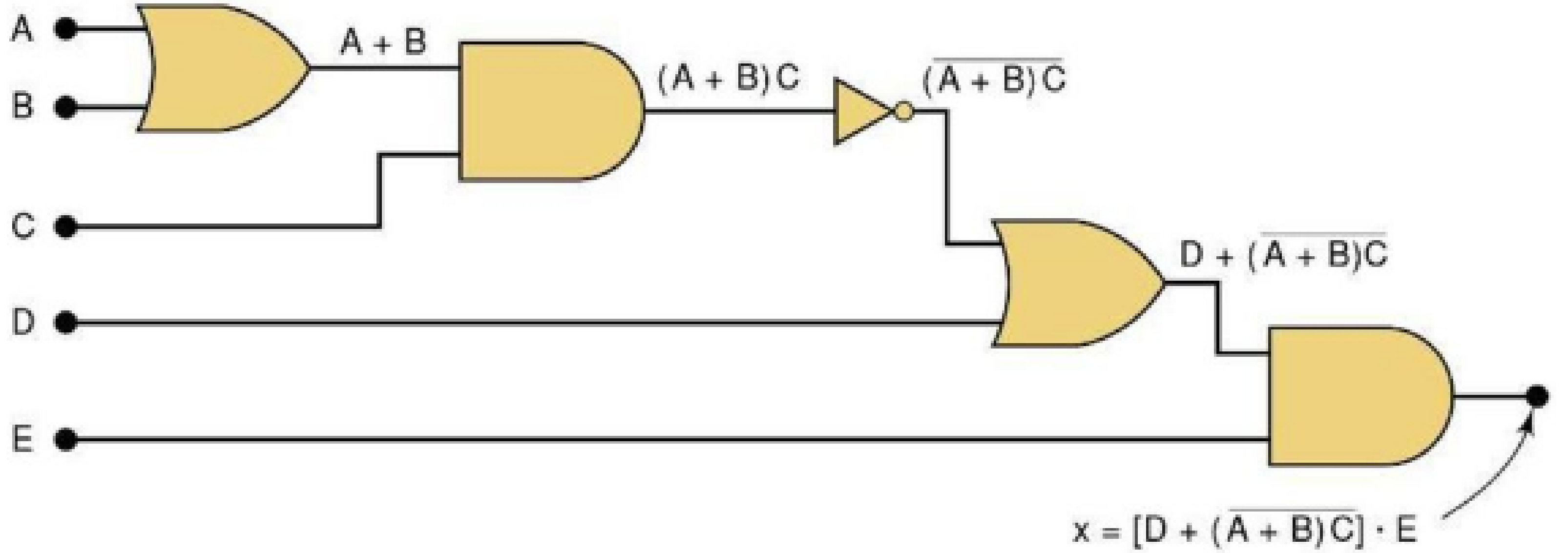
- These three basic Boolean operations can describe any logic circuit.

Evaluating Logic Circuit Outputs

- Rules for evaluating a Boolean expression:
  - Perform all inversions of single terms.
  - Perform all operations within parenthesis.
  - Perform AND operation before an OR operation unless parenthesis indicate otherwise.
  - If an expression has a bar over it, perform operations inside the expression, and then invert the result.

A=B=0, C=D=E=1,

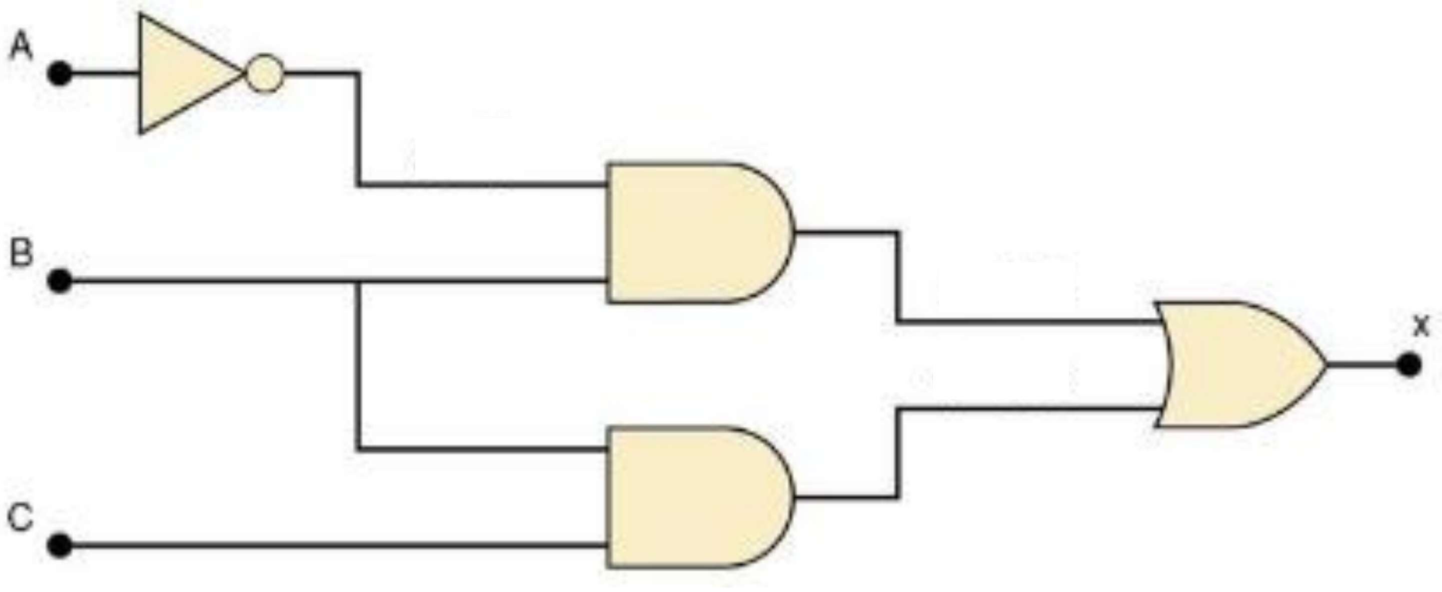
$$
\begin{aligned}
x &= [D + \overline{(A + B)C}] \cdot E \\
&= [1 + \overline{(0 + 0) \cdot 1}] \cdot 1 \\
&= [1 + \overline{0 \cdot 1}] \cdot 1 \\
&= [1 + \overline{0}] \cdot 1 \\
&= [1 + 1] \cdot 1 \\
&= 1 \cdot 1 \\
&= 1
\end{aligned}
$$



Evaluating Logic Circuit Outputs …

- The best way to analyze a circuit made up of multiple logic gates is to use a **truth table**:
  - It allows you to analyze one gate or logic combination at a time.
  - It allows you to easily double-check your work.
  - When you are done, you have a table of tremendous benefit in troubleshooting the logic circuit.

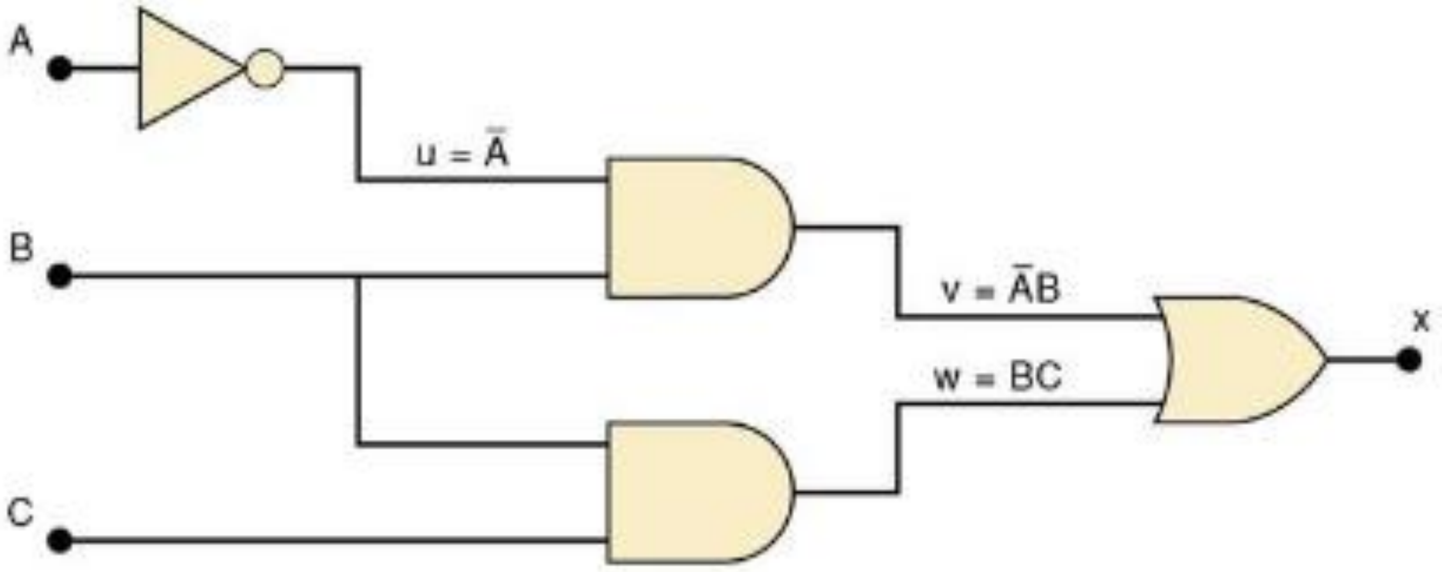
Analysis of a Logic Circuit using Truth Tables



A	B	C	u= $\overline{A}$	v= $\overline{A}B$	w= BC	x= v+w
0	0	0	1			
0	0	1	1			
0	1	0	1			
0	1	1	1			
1	0	0	0			
1	0	1	0			
1	1	0	0			
1	1	1	0			

1. The first step after listing all input combinations is to create a column in the truth table for each intermediate signal (node).

Node *u* has been filled as the complement of *A*

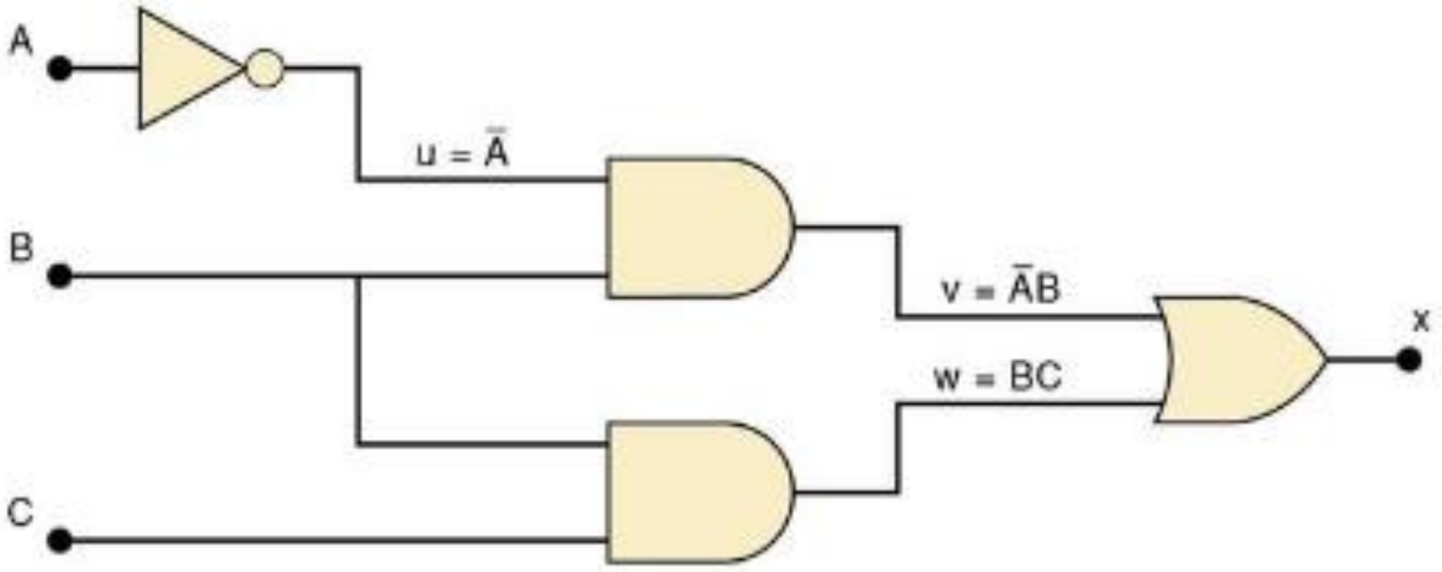


A	B	C	u= $\bar{A}$	v= $\bar{A}B$	w= BC	x= v+w
0	0	0	1	0		
0	0	1	1	0		
0	1	0	1	1		
0	1	1	1	1		
1	0	0	0	0		
1	0	1	0	0		
1	1	0	0	0		
1	1	1	0	0		

2. The next step is to fill in the values for column *v*.

Node *v* should be HIGH when *A′* (node *u*) and *B* are HIGH

Analysis of a Logic Circuit using Truth Tables...

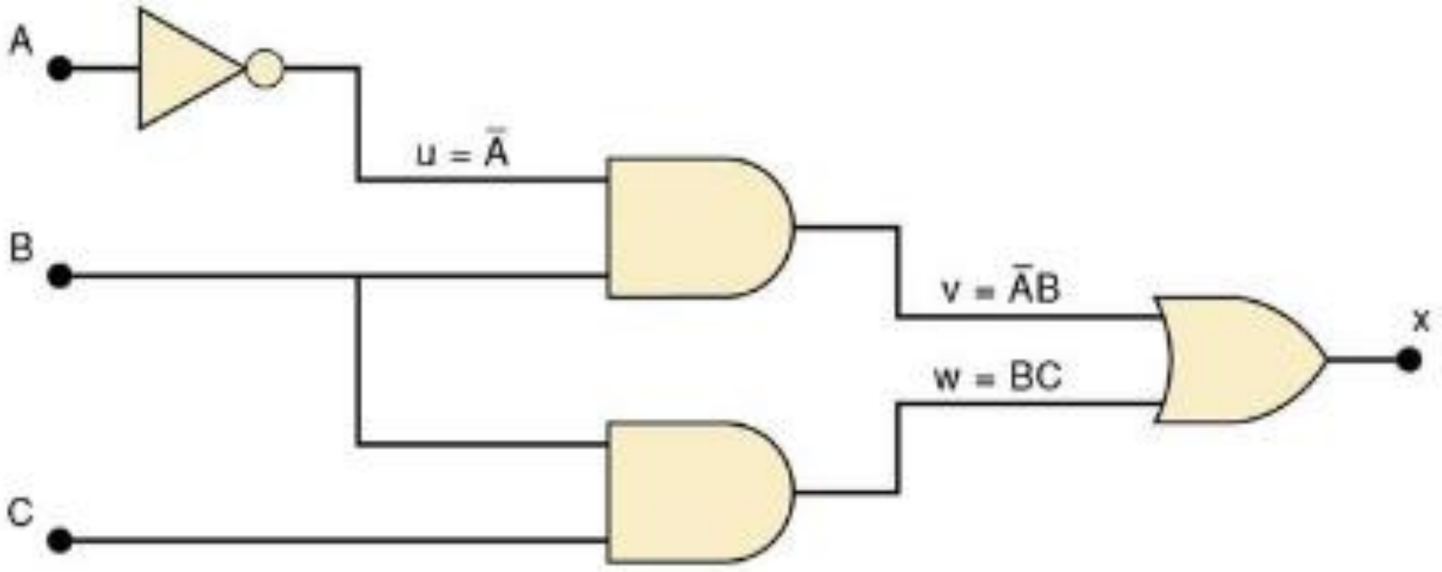


A	B	C	u= $\bar{A}$	v= $\bar{A}B$	w= BC	x= v+w
0	0	0	1	0	0	
0	0	1	1	0	0	
0	1	0	1	1	0	
0	1	1	1	1	1	
1	0	0	0	0	0	
1	0	1	0	0	0	
1	1	0	0	0	0	
1	1	1	0	0	1	

3. The third step is to predict the values at node *w* which is the logical product of *BC*.

This column is HIGH whenever B and C are HIGH

Analysis of a Logic Circuit using Truth Tables...



A	B	C	u= Ā	v= ĀB	w= BC	x= v+w
0	0	0	1	0	0	0
0	0	1	1	0	0	0
0	1	0	1	1	0	1
0	1	1	1	1	1	1
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	0	0	0	0
1	1	1	0	0	1	1

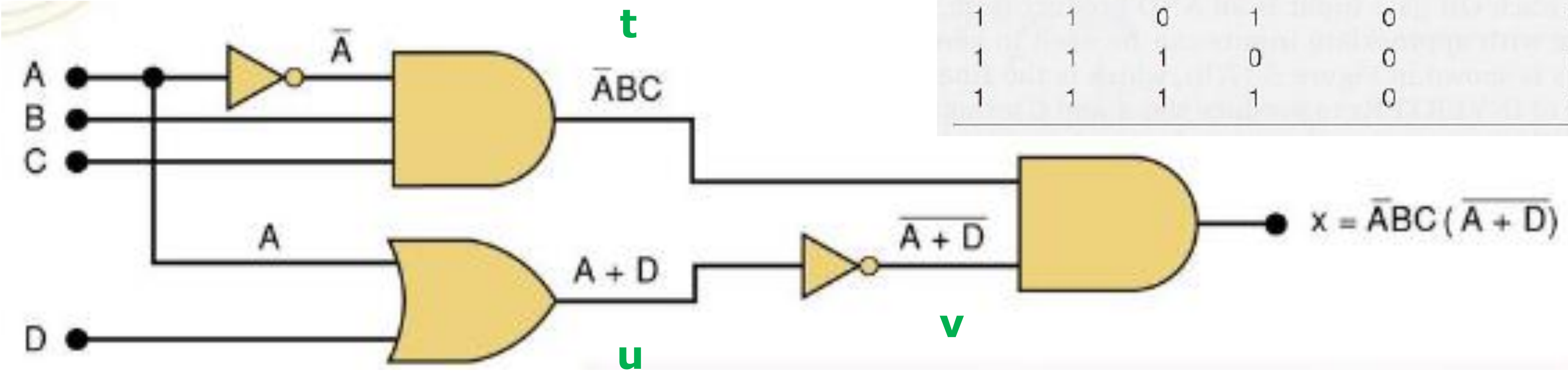
4. The final step is to logically combine columns *v* and *w* to predict the output *x*.

The *x* output will be HIGH when *v* or *w* is HIGH

Evaluating Logic Circuit Outputs

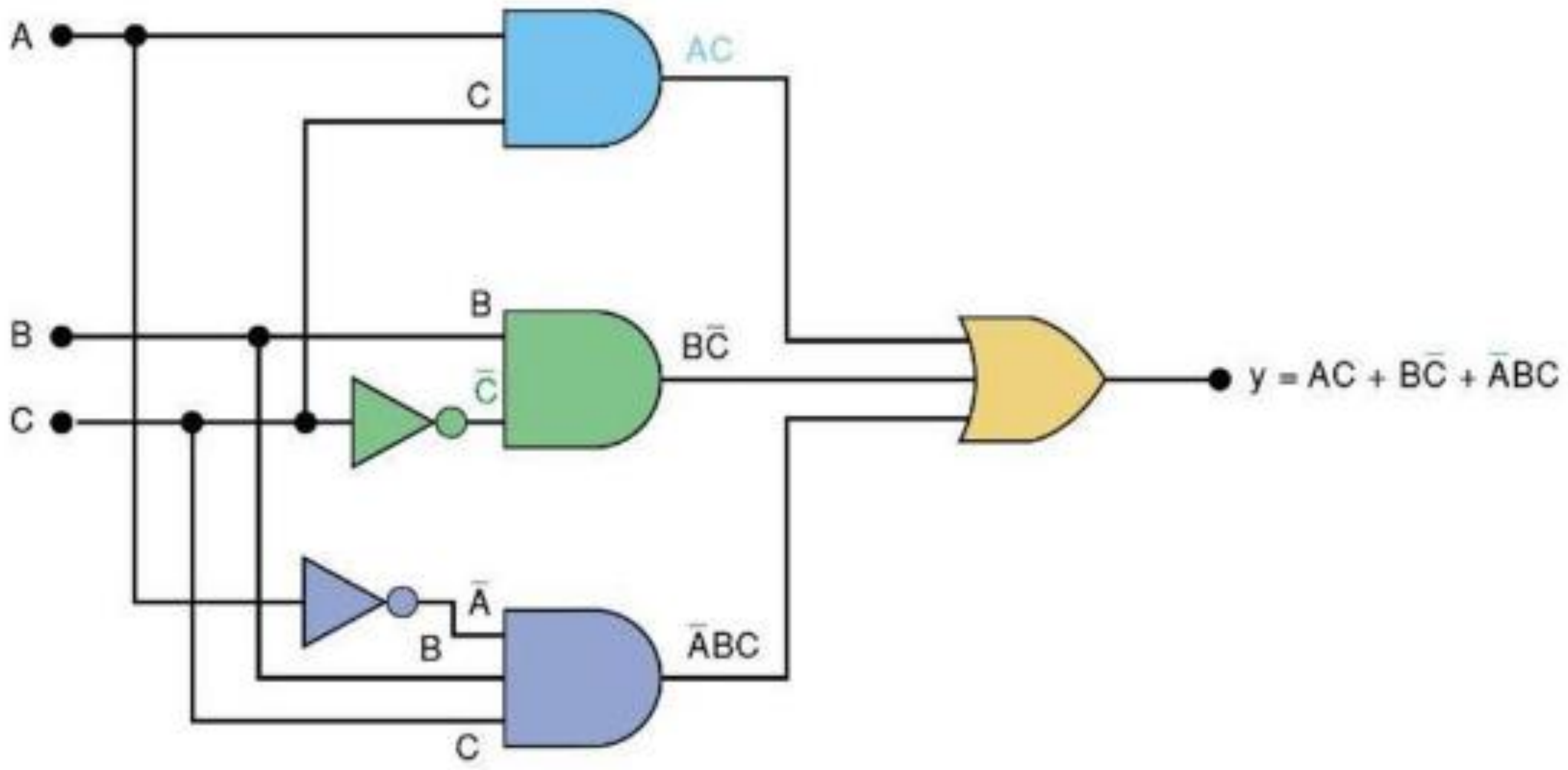
- *t*=1, when A=0, and B=C=1
- *u*=1, when A=1 or D=1
- *v*=u’
- *x*=1, when t=1 and v=1

A	B	C	D	$t = \overline{A}BC$	$u = A + D$	$v = \overline{A + D}$	$x = tv$
0	0	0	0	0	0	1	0
0	0	0	1	0	1	0	0
0	0	1	0	0	0	1	0
0	0	1	1	0	1	0	0
0	1	0	0	0	0	1	0
0	1	0	1	0	1	0	0
0	1	1	0	1	0	1	1
0	1	1	1	1	1	0	0
1	0	0	0	0	1	0	0
1	0	0	1	0	1	0	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	0	0
1	1	0	0	0	1	0	0
1	1	0	1	0	1	0	0
1	1	1	0	0	1	0	0
1	1	1	1	0	1	0	0



Implementing Circuits from Boolean Expressions

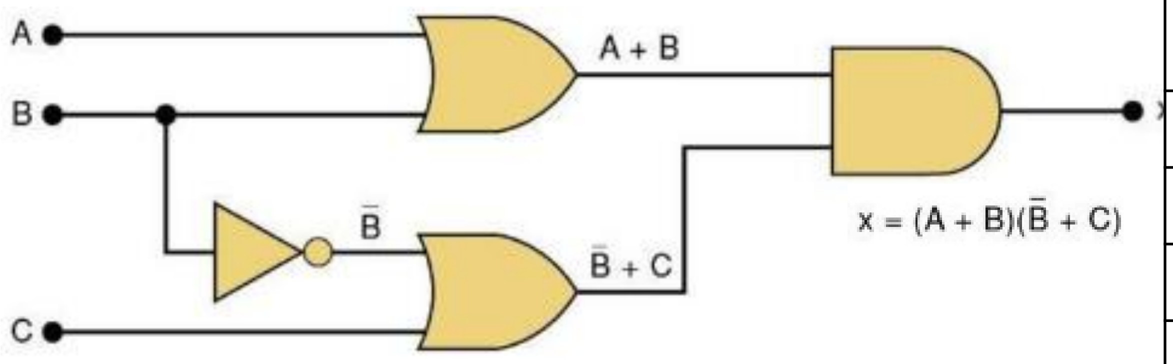
- Example: Construct a circuit whose output is **y=AC+BC′+A′BC**.
  - 3-input OR required with inputs equal AC, BC′, A′BC
  - Each OR gate input is an AND product term
  - and so on…



Implementing Circuits from Boolean Expressions …

- Test: Circuit diagram to implement x=(A+B)(B′+C). Also construct the truth table.

2 marks



A	B	C	u=B′	v=A+B	w=u+C	x=v.w
0	0	0	1	0	1	0
0	0	1	1	0	1	0
0	1	0	0	1	0	0
0	1	1	0	1	1	1
1	0	0	1	1	1	1
1	0	1	1	1	1	1
1	1	0	0	1	0	0
1	1	1	0	1	1	1