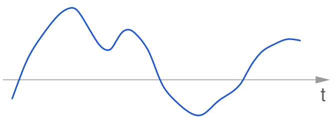
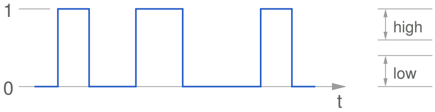
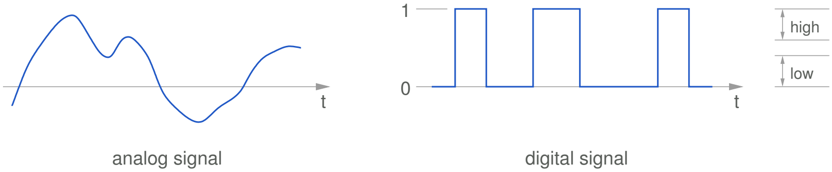
Digital circuits



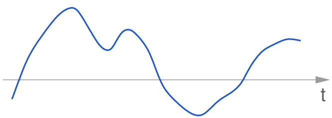
analog signal

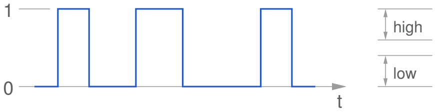


digital signal

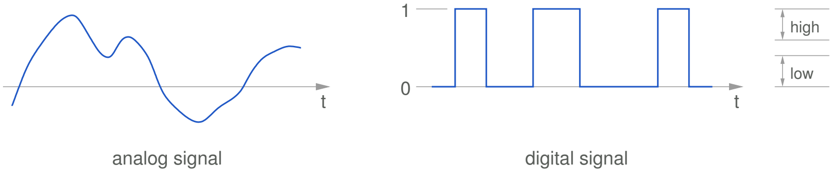


* An analog signal $x(t)$ is represented by a real number at a given time point.

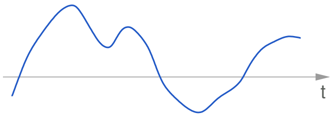




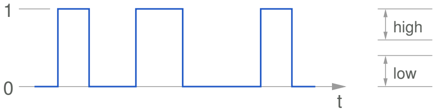
* An analog signal $x(t)$ is represented by a real number at a given time point.
* A digital signal is "binary" in nature, i.e., it takes on only two values: low (0) or high (1).



* An analog signal $x(t)$ is represented by a real number at a given time point.
* A digital signal is "binary" in nature, i.e., it takes on only two values: low (0) or high (1).
* Although we have shown 0 and 1 as constant levels, in reality, that is not required. Any value in the low (high) band will be interpreted as 0 (1) by digital circuits.



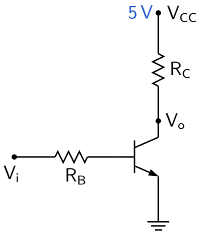
analog signal

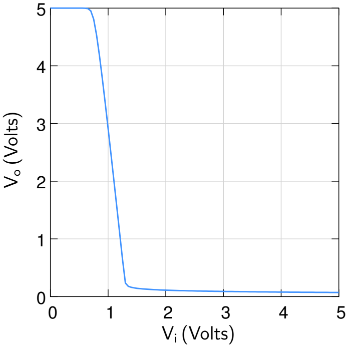


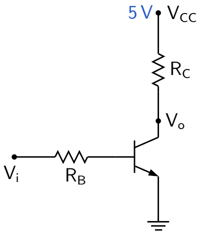
digital signal

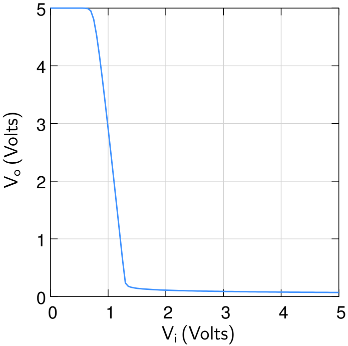
* An analog signal $x(t)$ is represented by a real number at a given time point.
* A digital signal is "binary" in nature, i.e., it takes on only two values: low (0) or high (1).
* Although we have shown 0 and 1 as constant levels, in reality, that is not required. Any value in the low (high) band will be interpreted as 0 (1) by digital circuits.
* The definition of low and high bands depends on the technology used, e.g.,
  TTL (Transitor-Transitor Logic)
  CMOS (Complementary MOS)
  ECL (Emitter-Coupled Logic)

A simple digital circuit

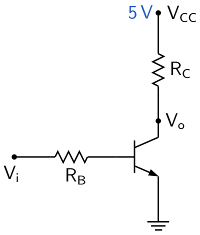


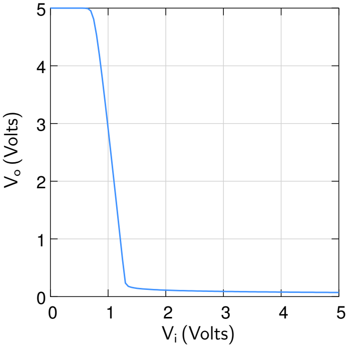






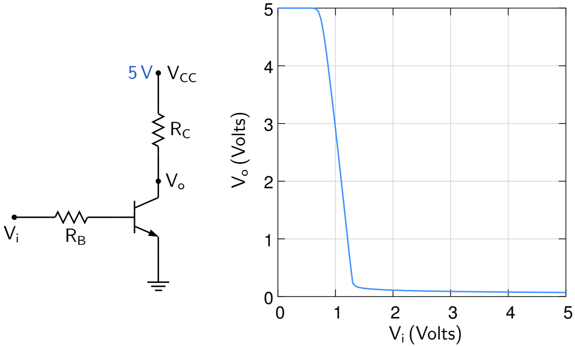
* If V_i is low ("0"), V_o is high ("1").
  If V_i is high ("1"), V_o is low ("0").





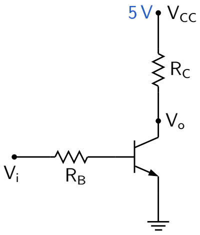
* If V_i is low ("0"), V_o is high ("1").
  If V_i is high ("1"), V_o is low ("0").
* The circuit is called an "inverter" because it inverts the logic level of the input. If the input is 0, it makes the output 1, and vice versa.

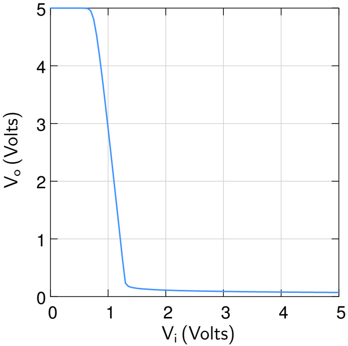
A simple digital circuit



* If V_i is low (“0”), V_o is high (“1”).
  If V_i is high (“1”), V_o is low (“0”).
* The circuit is called an “inverter” because it inverts the logic level of the input. If the input is 0, it makes the output 1, and vice versa.
* Digital circuits are made using a variety of devices. The simple BJT inverter is only an illustration.

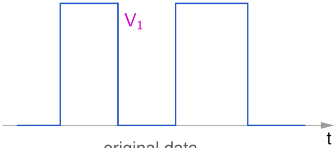
A simple digital circuit





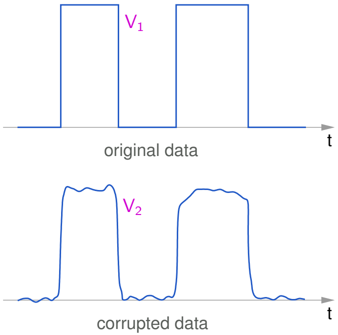
* If V_i is low ("0"), V_o is high ("1").
  If V_i is high ("1"), V_o is low ("0").
* The circuit is called an "inverter" because it inverts the logic level of the input. If the input is 0, it makes the output 1, and vice versa.
* Digital circuits are made using a variety of devices. The simple BJT inverter is only an illustration.
* Most of the VLSI circuits today employ the MOS technology because of the high packing density, high speed, and low power consumption it offers.

Digital circuits

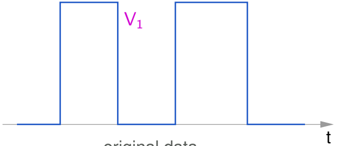


original data

Digital circuits

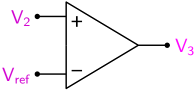


Digital circuits

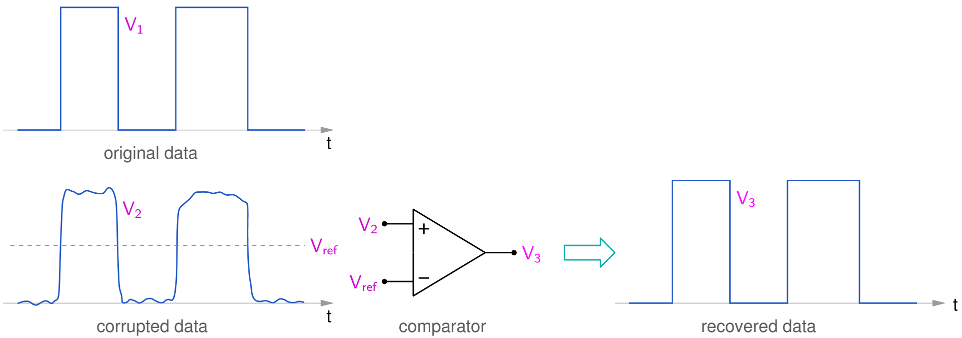


original data

corrupted data

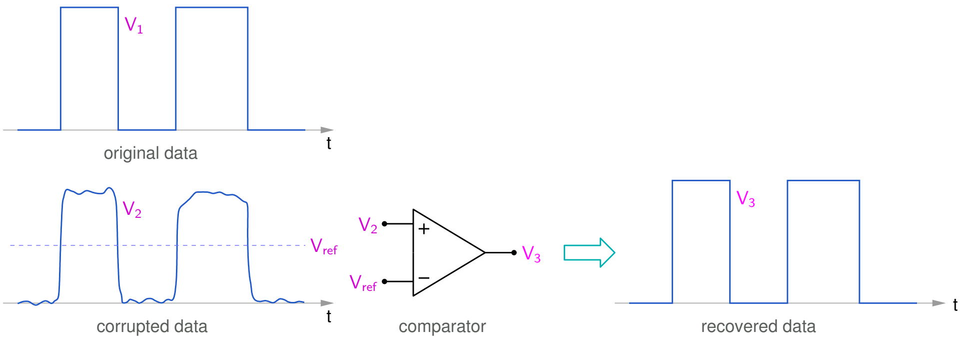


comparator



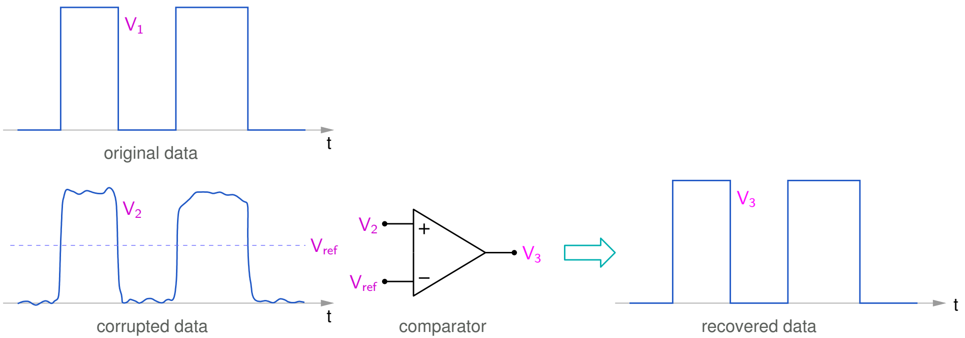
V_1
t
original data
V_2
V_{ref}
t
corrupted data
V_2
+
V_{ref}
−
V_3
comparator
V_3
t
recovered data

Digital circuits

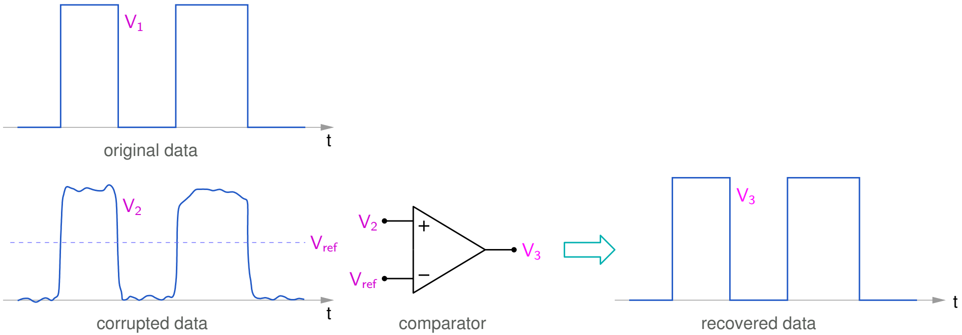


* A major advantage of digital systems is that, even if the original data gets distorted (e.g., in transmitting through optical fibre or storing on a CD) due to noise, attenuation, etc., it can be retrieved easily.

Digital circuits

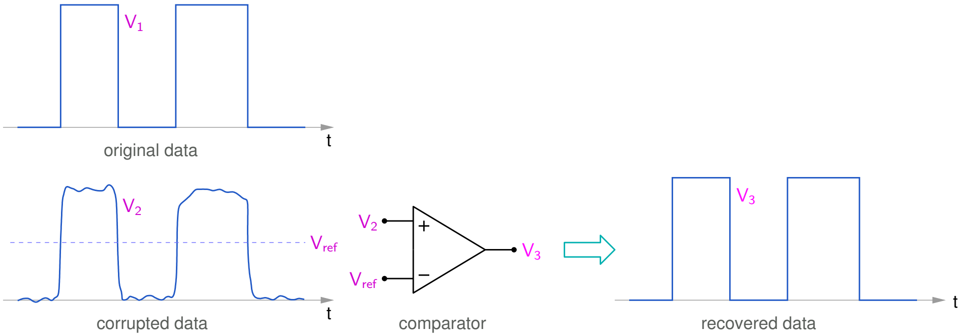


* A major advantage of digital systems is that, even if the original data gets distorted (e.g., in transmitting through optical fibre or storing on a CD) due to noise, attenuation, etc., it can be retrieved easily.
* There are several other benefits of using digital representation:



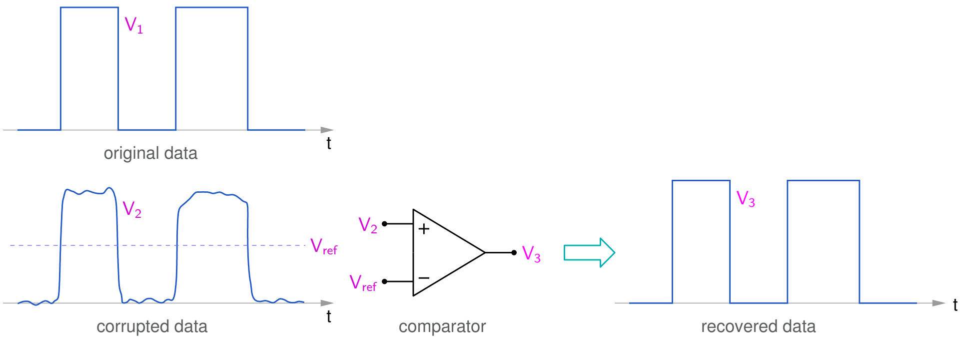
* A major advantage of digital systems is that, even if the original data gets distorted (e.g., in transmitting through optical fibre or storing on a CD) due to noise, attenuation, etc., it can be retrieved easily.

* There are several other benefits of using digital representation:
  - can use computers to process the data.



* A major advantage of digital systems is that, even if the original data gets distorted (e.g., in transmitting through optical fibre or storing on a CD) due to noise, attenuation, etc., it can be retrieved easily.
* There are several other benefits of using digital representation:
  - can use computers to process the data.
  - can store in a variety of storage media.

Digital circuits



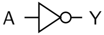
* A major advantage of digital systems is that, even if the original data gets distorted (e.g., in transmitting through optical fibre or storing on a CD) due to noise, attenuation, etc., it can be retrieved easily.
* There are several other benefits of using digital representation:
  - can use computers to process the data.
  - can store in a variety of storage media.
  - can *program* the functionality. For example, the behaviour of a digital filter can be changed simply by changing its coefficients.

Logical operations

Operation	NOT	AND	OR
Gate			
Truth table			
Notation			

Operation NOT AND OR

Gate



Truth table

A	Y
0	1
1	0

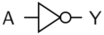
Notation $Y = \overline{A}$

Logical operations

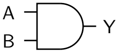
Operation	NOT	AND	OR

Gate

NOT:



AND:



Truth table

NOT:

A	Y
0	1
1	0

AND:

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

Notation

NOT: $Y = \overline{A}$

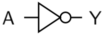
AND:
$$\begin{aligned} Y &= A \cdot B \\ &= A\,B \end{aligned}$$

Logical operations

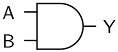
Operation: NOT | AND | OR

Gate

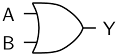
NOT:



AND:



OR:



Truth table

NOT:

A	Y
0	1
1	0

AND:

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

OR:

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Notation

NOT: $Y = \overline{A}$

AND: $Y = A \cdot B$
$= A\,B$

OR: $Y = A + B$

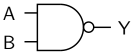
Logical operations

Operation	NAND	NOR	XOR
Gate			
Truth table			
Notation			

Logical operations

Operation	NAND	NOR	XOR

Gate



Truth table

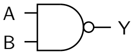
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

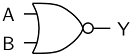
Notation

$$\begin{aligned} Y &= \overline{A \cdot B} \\ &= \overline{A\,B} \end{aligned}$$

Operation	NAND	NOR	XOR

Gate





Truth table

NAND:

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

NOR:

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

Notation

NAND:

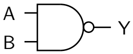
$$Y = \overline{A \cdot B} = \overline{A\,B}$$

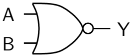
NOR:

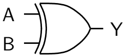
$$Y = \overline{A + B}$$

Operation	NAND	NOR	XOR

Gate







Truth table

NAND:

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

NOR:

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

XOR:

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Notation

NAND: $Y = \overline{A \cdot B} = \overline{A\,B}$

NOR: $Y = \overline{A + B}$

XOR: $Y = A \oplus B = A\,\overline{B} + \overline{A}\,B$

* The AND operation is *commutative*.
  $\rightarrow A \cdot B = B \cdot A$.

* The AND operation is *commutative*.
  $\rightarrow A \cdot B = B \cdot A$.
* The AND operation is *associative*.
  $\rightarrow (A \cdot B) \cdot C = A \cdot (B \cdot C)$.

* The AND operation is *commutative*.
  $\rightarrow A \cdot B = B \cdot A$.
* The AND operation is *associative*.
  $\rightarrow (A \cdot B) \cdot C = A \cdot (B \cdot C)$.
* The OR operation is *commutative*.
  $\rightarrow A + B = B + A$.

Logical operations

* The AND operation is *commutative*.
  $\rightarrow A \cdot B = B \cdot A$.
* The AND operation is *associative*.
  $\rightarrow (A \cdot B) \cdot C = A \cdot (B \cdot C)$.
* The OR operation is *commutative*.
  $\rightarrow A + B = B + A$.
* The OR operation is *associative*.
  $\rightarrow (A + B) + C = A + (B + C)$.

* Theorem: $\overline{\overline{A}} = A$.

* Theorem: $\overline{\overline{A}} = A$.

  The theorem can be proved by constructing a truth table:

A	$\overline{A}$	$\overline{\overline{A}}$
0	1	0
1	0	1

* Theorem: $\overline{\overline{A}} = A$.

  The theorem can be proved by constructing a truth table:

A	$\overline{A}$	$\overline{\overline{A}}$
0	1	0
1	0	1

  Therefore, for all possible values that A can take (i.e., 0 and 1), $\overline{\overline{A}}$ is the same as A.

  $\Rightarrow \overline{\overline{A}} = A$.

* Theorem: $\overline{\overline{A}} = A$.

  The theorem can be proved by constructing a truth table:

A	$\overline{A}$	$\overline{\overline{A}}$
0	1	0
1	0	1

  Therefore, for all possible values that A can take (i.e., 0 and 1), $\overline{\overline{A}}$ is the same as A.

  $\Rightarrow \overline{\overline{A}} = A$.

* Similarly, the following theorems can be proved:

$$A + 0 = A \qquad A \cdot 1 = A$$
$$A + 1 = 1 \qquad A \cdot 0 = 0$$
$$A + A = A \qquad A \cdot A = A$$
$$A + \overline{A} = 1 \qquad A \cdot \overline{A} = 0$$

* Theorem: $\overline{\overline{A}} = A$.

  The theorem can be proved by constructing a truth table:

A	$\overline{A}$	$\overline{\overline{A}}$
0	1	0
1	0	1

  Therefore, for all possible values that A can take (i.e., 0 and 1), $\overline{\overline{A}}$ is the same as A.

  $\Rightarrow \overline{\overline{A}} = A$.

* Similarly, the following theorems can be proved:

$$
\begin{array}{ll}
A + 0 = A & A \cdot 1 = A \\
A + 1 = 1 & A \cdot 0 = 0 \\
A + A = A & A \cdot A = A \\
A + \overline{A} = 1 & A \cdot \overline{A} = 0
\end{array}
$$

  Note the duality: $(+ \longleftrightarrow \cdot)$ and $(1 \longleftrightarrow 0)$.

De Morgan's theorems

A	B	$A+B$	$\overline{A+B}$	$\overline{A}$	$\overline{B}$	$\overline{A}\cdot\overline{B}$	$A\cdot B$	$\overline{A\cdot B}$	$\overline{A}+\overline{B}$
0	0								
0	1								
1	0								
1	1								

De Morgan's theorems

A	B	$A+B$	$\overline{A+B}$	$\overline{A}$	$\overline{B}$	$\overline{A}\cdot\overline{B}$	$A\cdot B$	$\overline{A\cdot B}$	$\overline{A}+\overline{B}$
0	0	0							
0	1	1							
1	0	1							
1	1	1							

A	B	$A+B$	$\overline{A+B}$	$\overline{A}$	$\overline{B}$	$\overline{A}\cdot\overline{B}$	$A\cdot B$	$\overline{A\cdot B}$	$\overline{A}+\overline{B}$
0	0	0	1						
0	1	1	0						
1	0	1	0						
1	1	1	0						

A	B	$A+B$	$\overline{A+B}$	$\overline{A}$	$\overline{B}$	$\overline{A}\cdot\overline{B}$	$A\cdot B$	$\overline{A\cdot B}$	$\overline{A}+\overline{B}$
0	0	0	1	1					
0	1	1	0	1					
1	0	1	0	0					
1	1	1	0	0					

De Morgan's theorems

A	B	$A+B$	$\overline{A+B}$	$\overline{A}$	$\overline{B}$	$\overline{A}\cdot\overline{B}$	$A\cdot B$	$\overline{A\cdot B}$	$\overline{A}+\overline{B}$
0	0	0	1	1	1				
0	1	1	0	1	0				
1	0	1	0	0	1				
1	1	1	0	0	0				

A	B	$A+B$	$\overline{A+B}$	$\overline{A}$	$\overline{B}$	$\overline{A}\cdot\overline{B}$	$A\cdot B$	$\overline{A\cdot B}$	$\overline{A}+\overline{B}$
0	0	0	1	1	1	1			
0	1	1	0	1	0	0			
1	0	1	0	0	1	0			
1	1	1	0	0	0	0			

De Morgan's theorems

A	B	$A+B$	$\overline{A+B}$	$\overline{A}$	$\overline{B}$	$\overline{A}\cdot\overline{B}$	$A\cdot B$	$\overline{A\cdot B}$	$\overline{A}+\overline{B}$
0	0	0	1	1	1	1	0		
0	1	1	0	1	0	0	0		
1	0	1	0	0	1	0	0		
1	1	1	0	0	0	0	1		

A	B	$A+B$	$\overline{A+B}$	$\overline{A}$	$\overline{B}$	$\overline{A} \cdot \overline{B}$	$A \cdot B$	$\overline{A \cdot B}$	$\overline{A}+\overline{B}$
0	0	0	1	1	1	1	0	1	
0	1	1	0	1	0	0	0	1	
1	0	1	0	0	1	0	0	1	
1	1	1	0	0	0	0	1	0	

A	B	$A+B$	$\overline{A+B}$	$\overline{A}$	$\overline{B}$	$\overline{A}\cdot\overline{B}$	$A\cdot B$	$\overline{A\cdot B}$	$\overline{A}+\overline{B}$
0	0	0	1	1	1	1	0	1	1
0	1	1	0	1	0	0	0	1	1
1	0	1	0	0	1	0	0	1	1
1	1	1	0	0	0	0	1	0	0

De Morgan's theorems

A	B	$A+B$	$\overline{A+B}$	$\overline{A}$	$\overline{B}$	$\overline{A}\cdot\overline{B}$	$A\cdot B$	$\overline{A\cdot B}$	$\overline{A}+\overline{B}$
0	0	0	1	1	1	1	0	1	1
0	1	1	0	1	0	0	0	1	1
1	0	1	0	0	1	0	0	1	1
1	1	1	0	0	0	0	1	0	0

* Comparing the truth tables for $\overline{A+B}$ and $\overline{A}\,\overline{B}$, we conclude that $\overline{A+B} = \overline{A}\,\overline{B}$.

A	B	$A+B$	$\overline{A+B}$	$\overline{A}$	$\overline{B}$	$\overline{A}\cdot\overline{B}$	$A\cdot B$	$\overline{A\cdot B}$	$\overline{A}+\overline{B}$
0	0	0	1	1	1	1	0	1	1
0	1	1	0	1	0	0	0	1	1
1	0	1	0	0	1	0	0	1	1
1	1	1	0	0	0	0	1	0	0

* Comparing the truth tables for $\overline{A+B}$ and $\overline{A}\,\overline{B}$, we conclude that $\overline{A+B}=\overline{A}\,\overline{B}$.
* Similiarly, $\overline{A\cdot B}=\overline{A}+\overline{B}$.

A	B	$A+B$	$\overline{A+B}$	$\overline{A}$	$\overline{B}$	$\overline{A}\cdot\overline{B}$	$A\cdot B$	$\overline{A\cdot B}$	$\overline{A}+\overline{B}$
0	0	0	1	1	1	1	0	1	1
0	1	1	0	1	0	0	0	1	1
1	0	1	0	0	1	0	0	1	1
1	1	1	0	0	0	0	1	0	0

* Comparing the truth tables for $\overline{A+B}$ and $\overline{A}\,\overline{B}$, we conclude that $\overline{A+B}=\overline{A}\,\overline{B}$.
* Similiarly, $\overline{A\cdot B}=\overline{A}+\overline{B}$.
* Similar relations hold for more than two variables, e.g.,

De Morgan's theorems

A	B	$A+B$	$\overline{A+B}$	$\overline{A}$	$\overline{B}$	$\overline{A}\cdot\overline{B}$	$A\cdot B$	$\overline{A\cdot B}$	$\overline{A}+\overline{B}$
0	0	0	1	1	1	1	0	1	1
0	1	1	0	1	0	0	0	1	1
1	0	1	0	0	1	0	0	1	1
1	1	1	0	0	0	0	1	0	0

* Comparing the truth tables for $\overline{A+B}$ and $\overline{A}\,\overline{B}$, we conclude that $\overline{A+B}=\overline{A}\,\overline{B}$.
* Similiarly, $\overline{A\cdot B}=\overline{A}+\overline{B}$.
* Similar relations hold for more than two variables, e.g.,

  $\overline{A\cdot B\cdot C}=\overline{A}+\overline{B}+\overline{C}$,

A	B	$A+B$	$\overline{A+B}$	$\overline{A}$	$\overline{B}$	$\overline{A}\cdot\overline{B}$	$A\cdot B$	$\overline{A\cdot B}$	$\overline{A}+\overline{B}$
0	0	0	1	1	1	1	0	1	1
0	1	1	0	1	0	0	0	1	1
1	0	1	0	0	1	0	0	1	1
1	1	1	0	0	0	0	1	0	0

* Comparing the truth tables for $\overline{A+B}$ and $\overline{A}\,\overline{B}$, we conclude that $\overline{A+B}=\overline{A}\,\overline{B}$.
* Similiarly, $\overline{A\cdot B}=\overline{A}+\overline{B}$.
* Similar relations hold for more than two variables, e.g.,

  $\overline{A\cdot B\cdot C}=\overline{A}+\overline{B}+\overline{C}$,

  $\overline{A+B+C+D}=\overline{A}\cdot\overline{B}\cdot\overline{C}\cdot\overline{D}$,

A	B	$A+B$	$\overline{A+B}$	$\overline{A}$	$\overline{B}$	$\overline{A}\cdot\overline{B}$	$A\cdot B$	$\overline{A\cdot B}$	$\overline{A}+\overline{B}$
0	0	0	1	1	1	1	0	1	1
0	1	1	0	1	0	0	0	1	1
1	0	1	0	0	1	0	0	1	1
1	1	1	0	0	0	0	1	0	0

* Comparing the truth tables for $\overline{A+B}$ and $\overline{A}\,\overline{B}$, we conclude that $\overline{A+B}=\overline{A}\,\overline{B}$.
* Similiarly, $\overline{A\cdot B}=\overline{A}+\overline{B}$.
* Similar relations hold for more than two variables, e.g.,

  $\overline{A\cdot B\cdot C}=\overline{A}+\overline{B}+\overline{C}$,

  $\overline{A+B+C+D}=\overline{A}\cdot\overline{B}\cdot\overline{C}\cdot\overline{D}$,

  $\overline{(A+B)\cdot C}=\overline{(A+B)}+\overline{C}=\overline{A}\cdot\overline{B}+\overline{C}$.

1. $A \cdot (B + C) = A\,B + A\,C$.

Distributive laws

1. $A \cdot (B + C) = A\,B + A\,C$.

A	B	C	$B + C$	$A \cdot (B + C)$	$A\,B$	$A\,C$	$A\,B + A\,C$
0	0	0					
0	0	1					
0	1	0					
0	1	1					
1	0	0					
1	0	1					
1	1	0					
1	1	1					

Distributive laws

1. $A \cdot (B + C) = A\,B + A\,C$.

A	B	C	$B + C$	$A \cdot (B + C)$	$A\,B$	$A\,C$	$A\,B + A\,C$
0	0	0	0				
0	0	1	1				
0	1	0	1				
0	1	1	1				
1	0	0	0				
1	0	1	1				
1	1	0	1				
1	1	1	1				

1. $A \cdot (B + C) = A\,B + A\,C$.

A	B	C	$B + C$	$A \cdot (B + C)$	$A\,B$	$A\,C$	$A\,B + A\,C$
0	0	0	0	0			
0	0	1	1	0			
0	1	0	1	0			
0	1	1	1	0			
1	0	0	0	0			
1	0	1	1	1			
1	1	0	1	1			
1	1	1	1	1			

1. $A \cdot (B + C) = A\,B + A\,C$.

A	B	C	$B+C$	$A \cdot (B+C)$	$A\,B$	$A\,C$	$A\,B + A\,C$
0	0	0	0	0	0		
0	0	1	1	0	0		
0	1	0	1	0	0		
0	1	1	1	0	0		
1	0	0	0	0	0		
1	0	1	1	1	0		
1	1	0	1	1	1		
1	1	1	1	1	1		

1. $A \cdot (B + C) = A\,B + A\,C$.

A	B	C	$B + C$	$A \cdot (B + C)$	$A\,B$	$A\,C$	$A\,B + A\,C$
0	0	0	0	0	0	0	
0	0	1	1	0	0	0	
0	1	0	1	0	0	0	
0	1	1	1	0	0	0	
1	0	0	0	0	0	0	
1	0	1	1	1	0	1	
1	1	0	1	1	1	0	
1	1	1	1	1	1	1	

1. $A \cdot (B + C) = A\,B + A\,C$.

A	B	C	$B + C$	$A \cdot (B + C)$	$A\,B$	$A\,C$	$A\,B + A\,C$
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

1. $A \cdot (B + C) = A\,B + A\,C$.

A	B	C	$B + C$	$A \cdot (B + C)$	$A\,B$	$A\,C$	$A\,B + A\,C$
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

Distributive laws

2. $A + B \cdot C = (A + B) \cdot (A + C)$.

2. $A + B \cdot C = (A + B) \cdot (A + C)$.

A	B	C	$B\,C$	$A + B\,C$	$A + B$	$A + C$	$(A + B)\,(A + C)$
0	0	0					
0	0	1					
0	1	0					
0	1	1					
1	0	0					
1	0	1					
1	1	0					
1	1	1					

Distributive laws

2. $A + B \cdot C = (A + B) \cdot (A + C)$.

A	B	C	$B\,C$	$A + B\,C$	$A + B$	$A + C$	$(A + B)\,(A + C)$
0	0	0	0				
0	0	1	0				
0	1	0	0				
0	1	1	1				
1	0	0	0				
1	0	1	0				
1	1	0	0				
1	1	1	1				

2. $A + B \cdot C = (A + B) \cdot (A + C)$.

A	B	C	$B\,C$	$A + B\,C$	$A + B$	$A + C$	$(A + B)\,(A + C)$
0	0	0	0	0			
0	0	1	0	0			
0	1	0	0	0			
0	1	1	1	1			
1	0	0	0	1			
1	0	1	0	1			
1	1	0	0	1			
1	1	1	1	1			

2. $A + B \cdot C = (A + B) \cdot (A + C)$.

A	B	C	$B\,C$	$A + B\,C$	$A + B$	$A + C$	$(A + B)\,(A + C)$
0	0	0	0	0	0		
0	0	1	0	0	0		
0	1	0	0	0	1		
0	1	1	1	1	1		
1	0	0	0	1	1		
1	0	1	0	1	1		
1	1	0	0	1	1		
1	1	1	1	1	1		

2. $A + B \cdot C = (A + B) \cdot (A + C)$.

A	B	C	$B\,C$	$A + B\,C$	$A + B$	$A + C$	$(A + B)\,(A + C)$
0	0	0	0	0	0	0	
0	0	1	0	0	0	1	
0	1	0	0	0	1	0	
0	1	1	1	1	1	1	
1	0	0	0	1	1	1	
1	0	1	0	1	1	1	
1	1	0	0	1	1	1	
1	1	1	1	1	1	1	

2. $A + B \cdot C = (A + B) \cdot (A + C)$.

A	B	C	$B\,C$	$A + B\,C$	$A + B$	$A + C$	$(A + B)\,(A + C)$
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

2. $A + B \cdot C = (A + B) \cdot (A + C)$.

A	B	C	$B\,C$	$A + B\,C$	$A + B$	$A + C$	$(A + B)\,(A + C)$
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

* $A + AB = A$.

  To prove this theorem, we can follow two approaches:

* $A + A\,B = A$.

  To prove this theorem, we can follow two approaches:

  (a) Construct truth tables for LHS and RHS for all possible input combinations, and show that they are the same.

* $A + A\,B = A$.

  To prove this theorem, we can follow two approaches:

  (a) Construct truth tables for LHS and RHS for all possible input combinations, and show that they are the same.

  (b) Use identities and theorems stated earlier to show that LHS=RHS.

* $A + A\,B = A$.

  To prove this theorem, we can follow two approaches:

  (a) Construct truth tables for LHS and RHS for all possible input combinations, and show that they are the same.

  (b) Use identities and theorems stated earlier to show that LHS=RHS.

$$
\begin{aligned}
A + A\,B &= A \cdot 1 + A \cdot B \\
&= A \cdot (1 + B) \\
&= A \cdot (1) \\
&= A
\end{aligned}
$$

Useful theorems

* $A + A\,B = A$.

  To prove this theorem, we can follow two approaches:

  (a) Construct truth tables for LHS and RHS for all possible input combinations, and show that they are the same.

  (b) Use identities and theorems stated earlier to show that LHS=RHS.

$$\begin{aligned} A + A\,B &= A \cdot 1 + A \cdot B \\ &= A \cdot (1 + B) \\ &= A \cdot (1) \\ &= A \end{aligned}$$

* $A \cdot (A + B) = A$.

* $A + A\,B = A$.

  To prove this theorem, we can follow two approaches:

  (a) Construct truth tables for LHS and RHS for all possible input combinations, and show that they are the same.

  (b) Use identities and theorems stated earlier to show that LHS=RHS.

$$\begin{aligned} A + A\,B &= A \cdot 1 + A \cdot B \\ &= A \cdot (1 + B) \\ &= A \cdot (1) \\ &= A \end{aligned}$$

* $A \cdot (A + B) = A$.

  Proof: $$\begin{aligned} A \cdot (A + B) &= A \cdot A + A \cdot B \\ &= A + A\,B \\ &= A \end{aligned}$$

$A + A\,B = A \quad \longleftrightarrow \quad A \cdot (A + B) = A.$

Note the duality between OR and AND.

Duality

$$A + A\,B = A \quad \longleftrightarrow \quad A \cdot (A + B) = A.$$

Note the duality between OR and AND.

Dual of $A + (A\,B)$ (LHS): $A\,B \rightarrow A + B$
$A + A\,B \rightarrow A \cdot (A + B)$.

Duality

$A + A\,B = A \quad \longleftrightarrow \quad A \cdot (A + B) = A.$

Note the duality between OR and AND.

Dual of $A + (A\,B)$ (LHS): $A\,B \rightarrow A + B$

$A + A\,B \rightarrow A \cdot (A + B).$

Dual of A (RHS) $= A$ (since there are no operations ivolved).

Duality

$A + A\,B = A \quad \longleftrightarrow \quad A \cdot (A + B) = A.$

Note the duality between OR and AND.

Dual of $A + (A\,B)$ (LHS): $A\,B \rightarrow A + B$

$A + A\,B \rightarrow A \cdot (A + B).$

Dual of A (RHS) $= A$ (since there are no operations ivolved).

$\Rightarrow A \cdot (A + B) = A.$

Duality

$A + A\,B = A \quad \longleftrightarrow \quad A \cdot (A + B) = A.$

Note the duality between OR and AND.

Dual of $A + (A\,B)$ (LHS): $A\,B \rightarrow A + B$

$A + A\,B \rightarrow A \cdot (A + B).$

Dual of A (RHS) $= A$ (since there are no operations ivolved).

$\Rightarrow A \cdot (A + B) = A.$

Similarly, consider $A + \overline{A} = 1$, with $(+ \longleftrightarrow .)$ and $(1 \longleftrightarrow 0)$.

$A + A\,B = A \quad \longleftrightarrow \quad A \cdot (A + B) = A.$

Note the duality between OR and AND.

Dual of $A + (A\,B)$ (LHS): $A\,B \rightarrow A + B$

$A + A\,B \rightarrow A \cdot (A + B).$

Dual of A (RHS) $= A$ (since there are no operations ivolved).

$\Rightarrow A \cdot (A + B) = A.$

Similarly, consider $A + \overline{A} = 1$, with $(+ \longleftrightarrow .)$ and $(1 \longleftrightarrow 0)$.

Dual of LHS $= A \cdot \overline{A}$.

Duality

$A + A\,B = A \quad \longleftrightarrow \quad A \cdot (A + B) = A.$

Note the duality between OR and AND.

Dual of $A + (A\,B)$ (LHS): $A\,B \rightarrow A + B$

$A + A\,B \rightarrow A \cdot (A + B).$

Dual of A (RHS) $= A$ (since there are no operations ivolved).

$\Rightarrow A \cdot (A + B) = A.$

Similarly, consider $A + \overline{A} = 1$, with $(+ \longleftrightarrow .)$ and $(1 \longleftrightarrow 0)$.

Dual of LHS $= A \cdot \overline{A}$.

Dual of RHS $= 0$.

Duality

$A + A\,B = A \quad \longleftrightarrow \quad A \cdot (A + B) = A.$

Note the duality between OR and AND.

Dual of $A + (A\,B)$ (LHS): $A\,B \rightarrow A + B$

$A + A\,B \rightarrow A \cdot (A + B).$

Dual of A (RHS) $= A$ (since there are no operations ivolved).

$\Rightarrow A \cdot (A + B) = A.$

Similarly, consider $A + \overline{A} = 1$, with $(+ \longleftrightarrow .)$ and $(1 \longleftrightarrow 0)$.

Dual of LHS $= A \cdot \overline{A}$.

Dual of RHS $= 0$.

$\Rightarrow A \cdot \overline{A} = 0.$

Useful theorems

* $A + \overline{A}\,B = A + B$.

Useful theorems

* $A + \overline{A}\,B = A + B.$

  Proof: $$\begin{aligned} A + \overline{A}\,B &= (A + \overline{A}) \cdot (A + B) \quad \text{(by distributive law)} \\ &= 1 \cdot (A + B) \\ &= A + B \end{aligned}$$

  Dual theorem: $A \cdot (\overline{A} + B) = A\,B.$

* $A + \overline{A}\,B = A + B$.

  Proof: $$\begin{aligned} A + \overline{A}\,B &= (A + \overline{A}) \cdot (A + B) \quad \text{(by distributive law)} \\ &= 1 \cdot (A + B) \\ &= A + B \end{aligned}$$

  Dual theorem: $A \cdot (\overline{A} + B) = A\,B$.

* $A\,B + A\,\overline{B} = A$.

Useful theorems

* $A + \overline{A}\,B = A + B$.

  Proof: $$\begin{aligned} A + \overline{A}\,B &= (A + \overline{A}) \cdot (A + B) \quad \text{(by distributive law)} \\ &= 1 \cdot (A + B) \\ &= A + B \end{aligned}$$

  Dual theorem: $A \cdot (\overline{A} + B) = A\,B$.

* $A\,B + A\,\overline{B} = A$.

  Proof: $$\begin{aligned} A\,B + A\,\overline{B} &= A \cdot (B + \overline{B}) \quad \text{(by distributive law)} \\ &= A \cdot 1 \\ &= A \end{aligned}$$

  Dual theorem: $(A + B) \cdot (A + \overline{B}) = A$.

In an India-Australia match, India will win if one or more of the following conditions are met:

In an India-Australia match, India will win if one or more of the following conditions are met:

(a) Tendulkar scores a century.

In an India-Australia match, India will win if one or more of the following conditions are met:

(a) Tendulkar scores a century.

(b) Tedulkar does not score a century AND Warne fails (to get wickets).

In an India-Australia match, India will win if one or more of the following conditions are met:

(a) Tendulkar scores a century.

(b) Tedulkar does not score a century AND Warne fails (to get wickets).

(c) Tedulkar does not score a century AND Sehwag scores a century.

A game of words

In an India-Australia match, India will win if one or more of the following conditions are met:

(a) Tendulkar scores a century.

(b) Tedulkar does not score a century AND Warne fails (to get wickets).

(c) Tedulkar does not score a century AND Sehwag scores a century.

Let T $\equiv$ Tendulkar scores a century.
S $\equiv$ Sehwag scores a century.
W $\equiv$ Warne fails.
I $\equiv$ India wins.

In an India-Australia match, India will win if one or more of the following conditions are met:

(a) Tendulkar scores a century.

(b) Tedulkar does not score a century AND Warne fails (to get wickets).

(c) Tedulkar does not score a century AND Sehwag scores a century.

Let $T \equiv$ Tendulkar scores a century.
$S \equiv$ Sehwag scores a century.
$W \equiv$ Warne fails.
$I \equiv$ India wins.

$I = T + \overline{T}\, W + \overline{T}\, S$

In an India-Australia match, India will win if one or more of the following conditions are met:

(a) Tendulkar scores a century.

(b) Tedulkar does not score a century AND Warne fails (to get wickets).

(c) Tedulkar does not score a century AND Sehwag scores a century.

Let $T \equiv$ Tendulkar scores a century.
$S \equiv$ Sehwag scores a century.
$W \equiv$ Warne fails.
$I \equiv$ India wins.

$$\begin{aligned} I &= T + \overline{T}\,W + \overline{T}\,S \\ &= T + T + \overline{T}\,W + \overline{T}\,S \end{aligned}$$

In an India-Australia match, India will win if one or more of the following conditions are met:

(a) Tendulkar scores a century.

(b) Tedulkar does not score a century AND Warne fails (to get wickets).

(c) Tedulkar does not score a century AND Sehwag scores a century.

Let $T \equiv$ Tendulkar scores a century.
$S \equiv$ Sehwag scores a century.
$W \equiv$ Warne fails.
$I \equiv$ India wins.

$$
\begin{aligned}
I &= T + \overline{T}\,W + \overline{T}\,S \\
&= T + T + \overline{T}\,W + \overline{T}\,S \\
&= (T + \overline{T}\,W) + (T + \overline{T}\,S)
\end{aligned}
$$

In an India-Australia match, India will win if one or more of the following conditions are met:

(a) Tendulkar scores a century.

(b) Tedulkar does not score a century AND Warne fails (to get wickets).

(c) Tedulkar does not score a century AND Sehwag scores a century.

Let $T \equiv$ Tendulkar scores a century.
$S \equiv$ Sehwag scores a century.
$W \equiv$ Warne fails.
$I \equiv$ India wins.

$$\begin{aligned} I &= T + \overline{T}\,W + \overline{T}\,S \\ &= T + T + \overline{T}\,W + \overline{T}\,S \\ &= (T + \overline{T}\,W) + (T + \overline{T}\,S) \\ &= (T + \overline{T}) \cdot (T + W) + (T + \overline{T}) \cdot (T + S) \end{aligned}$$

In an India-Australia match, India will win if one or more of the following conditions are met:

(a) Tendulkar scores a century.

(b) Tedulkar does not score a century AND Warne fails (to get wickets).

(c) Tedulkar does not score a century AND Sehwag scores a century.

Let $T \equiv$ Tendulkar scores a century.
$S \equiv$ Sehwag scores a century.
$W \equiv$ Warne fails.
$I \equiv$ India wins.

$$
\begin{aligned}
I &= T + \overline{T}\,W + \overline{T}\,S \\
&= T + T + \overline{T}\,W + \overline{T}\,S \\
&= (T + \overline{T}\,W) + (T + \overline{T}\,S) \\
&= (T + \overline{T}) \cdot (T + W) + (T + \overline{T}) \cdot (T + S) \\
&= T + W + T + S \\
&= T + W + S
\end{aligned}
$$

In an India-Australia match, India will win if one or more of the following conditions are met:

(a) Tendulkar scores a century.

(b) Tedulkar does not score a century AND Warne fails (to get wickets).

(c) Tedulkar does not score a century AND Sehwag scores a century.

Let $T \equiv$ Tendulkar scores a century.
$S \equiv$ Sehwag scores a century.
$W \equiv$ Warne fails.
$I \equiv$ India wins.

$$
\begin{aligned}
I &= T + \overline{T}\,W + \overline{T}\,S \\
&= T + T + \overline{T}\,W + \overline{T}\,S \\
&= (T + \overline{T}\,W) + (T + \overline{T}\,S) \\
&= (T + \overline{T}) \cdot (T + W) + (T + \overline{T}) \cdot (T + S) \\
&= T + W + T + S \\
&= T + W + S
\end{aligned}
$$

i.e., India will win if one or more of the following hold:

(a) Tendulkar strikes, (b) Warne fails, (c) Sehwag strikes.

Consider a function X of three variables A, B, C:

$$X = \overline{A}\,B\,\overline{C} + \overline{A}\,B\,C + A\,\overline{B}\,\overline{C} + A\,B\,\overline{C}$$
$$\equiv X_1 \qquad + X_2 \qquad + X_3 \qquad + X_4$$

Consider a function X of three variables A, B, C:

$$X = \overline{A}\,B\,\overline{C} + \overline{A}\,B\,C + A\,\overline{B}\,\overline{C} + A\,B\,\overline{C}$$
$$\equiv X_1 \quad + X_2 \quad + X_3 \quad + X_4$$

This form is called the "sum of products" form ("sum" corresponding to OR and "product" corresponding to AND).

Consider a function X of three variables A, B, C:

$$X = \overline{A}\,B\,\overline{C} + \overline{A}\,B\,C + A\,\overline{B}\,\overline{C} + A\,B\,\overline{C}$$
$$\equiv X_1 \quad + X_2 \quad + X_3 \quad + X_4$$

This form is called the "sum of products" form ("sum" corresponding to OR and "product" corresponding to AND).

We can construct the truth table for X in a systematic manner:

Consider a function X of three variables A, B, C:

$$X = \overline{A}\,B\,\overline{C} + \overline{A}\,B\,C + A\,\overline{B}\,\overline{C} + A\,B\,\overline{C}$$
$$\equiv X_1 \quad + X_2 \quad + X_3 \quad + X_4$$

This form is called the "sum of products" form ("sum" corresponding to OR and "product" corresponding to AND).

We can construct the truth table for X in a systematic manner:

(1) Enumerate all possible combinations of A, B, C.
Since each of A, B, C can take two values (0 or 1), we have 2^3 possibilities.

Consider a function X of three variables A, B, C:

$$X = \overline{A}\,B\,\overline{C} + \overline{A}\,B\,C + A\,\overline{B}\,\overline{C} + A\,B\,\overline{C}$$
$$\equiv X_1 \quad + X_2 \quad + X_3 \quad + X_4$$

This form is called the "sum of products" form ("sum" corresponding to OR and "product" corresponding to AND).

We can construct the truth table for X in a systematic manner:

(1) Enumerate all possible combinations of A, B, C.
Since each of A, B, C can take two values (0 or 1), we have 2^3 possibilities.

(2) Tabulate $X_1 = \overline{A}\,B\,\overline{C}$, etc. Note that X_1 is 1 only if $\overline{A} = B = \overline{C} = 1$ (i.e., $A = 0$, $B = 1$, $C = 0$), and 0 otherwise.

Consider a function X of three variables A, B, C:

$$X = \overline{A}\,B\,\overline{C} + \overline{A}\,B\,C + A\,\overline{B}\,\overline{C} + A\,B\,\overline{C}$$
$$\equiv X_1 \quad + X_2 \quad + X_3 \quad + X_4$$

This form is called the "sum of products" form ("sum" corresponding to OR and "product" corresponding to AND).

We can construct the truth table for X in a systematic manner:

(1) Enumerate all possible combinations of A, B, C.
Since each of A, B, C can take two values (0 or 1), we have 2^3 possibilities.

(2) Tabulate $X_1 = \overline{A}\,B\,\overline{C}$, etc. Note that X_1 is 1 only if $\overline{A} = B = \overline{C} = 1$ (i.e., $A = 0$, $B = 1$, $C = 0$), and 0 otherwise.

(3) Since $X = X_1 + X_2 + X_3 + X_4$,
X is 1 if any of X_1, X_2, X_3, X_4 is 1; else X is 0.
$\rightarrow$ tabulate X.

"Sum of products" form

$$X = X_1 + X_2 + X_3 + X_4 = \overline{A}\,B\,\overline{C} + \overline{A}\,B\,C + A\,\overline{B}\,\overline{C} + A\,B\,\overline{C}$$

A	B	C	X_1	X_2	X_3	X_4	X
0	0	0					
0	0	1					
0	1	0					
0	1	1					
1	0	0					
1	0	1					
1	1	0					
1	1	1					

"Sum of products" form

$$X = X_1 + X_2 + X_3 + X_4 = \overline{A}\,B\,\overline{C} + \overline{A}\,B\,C + A\,\overline{B}\,\overline{C} + A\,B\,\overline{C}$$

A	B	C	X_1	X_2	X_3	X_4	X
0	0	0					
0	0	1					
0	1	0	1				
0	1	1					
1	0	0					
1	0	1					
1	1	0					
1	1	1					

“Sum of products” form

$$X = X_1 + X_2 + X_3 + X_4 = \overline{A}\,B\,\overline{C} + \overline{A}\,B\,C + A\,\overline{B}\,\overline{C} + A\,B\,\overline{C}$$

A	B	C	X_1	X_2	X_3	X_4	X
0	0	0	0				
0	0	1	0				
0	1	0	1				
0	1	1	0				
1	0	0	0				
1	0	1	0				
1	1	0	0				
1	1	1	0				

"Sum of products" form

$$X = X_1 + X_2 + X_3 + X_4 = \overline{A}\,B\,\overline{C} + \overline{A}\,B\,C + A\,\overline{B}\,\overline{C} + A\,B\,\overline{C}$$

A	B	C	X_1	X_2	X_3	X_4	X
0	0	0	0				
0	0	1	0				
0	1	0	1				
0	1	1	0	1			
1	0	0	0				
1	0	1	0				
1	1	0	0				
1	1	1	0				

"Sum of products" form

$$X = X_1 + X_2 + X_3 + X_4 = \overline{A}\,B\,\overline{C} + \overline{A}\,B\,C + A\,\overline{B}\,\overline{C} + A\,B\,\overline{C}$$

A	B	C	X_1	X_2	X_3	X_4	X
0	0	0	0	0			
0	0	1	0	0			
0	1	0	1	0			
0	1	1	0	1			
1	0	0	0	0			
1	0	1	0	0			
1	1	0	0	0			
1	1	1	0	0			

“Sum of products” form

$$X = X_1 + X_2 + X_3 + X_4 = \overline{A}\,B\,\overline{C} + \overline{A}\,B\,C + A\,\overline{B}\,\overline{C} + A\,B\,\overline{C}$$

A	B	C	X_1	X_2	X_3	X_4	X
0	0	0	0	0			
0	0	1	0	0			
0	1	0	1	0			
0	1	1	0	1			
1	0	0	0	0	1		
1	0	1	0	0			
1	1	0	0	0			
1	1	1	0	0			

"Sum of products" form

$$X = X_1 + X_2 + X_3 + X_4 = \overline{A}\,B\,\overline{C} + \overline{A}\,B\,C + A\,\overline{B}\,\overline{C} + A\,B\,\overline{C}$$

A	B	C	X_1	X_2	X_3	X_4	X
0	0	0	0	0	0		
0	0	1	0	0	0		
0	1	0	1	0	0		
0	1	1	0	1	0		
1	0	0	0	0	1		
1	0	1	0	0	0		
1	1	0	0	0	0		
1	1	1	0	0	0		

"Sum of products" form

$$X = X_1 + X_2 + X_3 + X_4 = \overline{A}\,B\,\overline{C} + \overline{A}\,B\,C + A\,\overline{B}\,\overline{C} + A\,B\,\overline{C}$$

A	B	C	X_1	X_2	X_3	X_4	X
0	0	0	0	0	0		
0	0	1	0	0	0		
0	1	0	1	0	0		
0	1	1	0	1	0		
1	0	0	0	0	1		
1	0	1	0	0	0		
1	1	0	0	0	0	1	
1	1	1	0	0	0		

"Sum of products" form

$$X = X_1 + X_2 + X_3 + X_4 = \overline{A}\,B\,\overline{C} + \overline{A}\,B\,C + A\,\overline{B}\,\overline{C} + A\,B\,\overline{C}$$

A	B	C	X_1	X_2	X_3	X_4	X
0	0	0	0	0	0	0	
0	0	1	0	0	0	0	
0	1	0	1	0	0	0	
0	1	1	0	1	0	0	
1	0	0	0	0	1	0	
1	0	1	0	0	0	0	
1	1	0	0	0	0	1	
1	1	1	0	0	0	0	

"Sum of products" form

$$X = X_1 + X_2 + X_3 + X_4 = \overline{A}\,B\,\overline{C} + \overline{A}\,B\,C + A\,\overline{B}\,\overline{C} + A\,B\,\overline{C}$$

A	B	C	X_1	X_2	X_3	X_4	X
0	0	0	0	0	0	0	
0	0	1	0	0	0	0	
0	1	0	1	0	0	0	1
0	1	1	0	1	0	0	1
1	0	0	0	0	1	0	1
1	0	1	0	0	0	0	
1	1	0	0	0	0	1	1
1	1	1	0	0	0	0	

"Sum of products" form

$$X = X_1 + X_2 + X_3 + X_4 = \overline{A}\,B\,\overline{C} + \overline{A}\,B\,C + A\,\overline{B}\,\overline{C} + A\,B\,\overline{C}$$

A	B	C	X_1	X_2	X_3	X_4	X
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	1	0	0	1
1	0	0	0	0	1	0	1
1	0	1	0	0	0	0	0
1	1	0	0	0	0	1	1
1	1	1	0	0	0	0	0

Consider a function Y of three variables A, B, C:

$$
\begin{aligned}
Y &= (A + B + C) \cdot (A + B + \overline{C}) \cdot (\overline{A} + B + \overline{C}) \cdot (\overline{A} + \overline{B} + \overline{C}) \\
&\equiv Y_1 \cdot Y_2 \cdot Y_3 \cdot Y_4
\end{aligned}
$$

Consider a function Y of three variables A, B, C:

$$Y = (A + B + C) \cdot (A + B + \overline{C}) \cdot (\overline{A} + B + \overline{C}) \cdot (\overline{A} + \overline{B} + \overline{C})$$
$$\equiv Y_1 \cdot Y_2 \cdot Y_3 \cdot Y_4$$

This form is called the "product of sums" form ("sum" corresponding to OR, and "product" corresponding to AND).

Consider a function Y of three variables A, B, C:

$$
\begin{aligned}
Y &= (A+B+C)\cdot(A+B+\overline{C})\cdot(\overline{A}+B+\overline{C})\cdot(\overline{A}+\overline{B}+\overline{C})\\
&\equiv Y_1 \cdot Y_2 \cdot Y_3 \cdot Y_4
\end{aligned}
$$

This form is called the "product of sums" form ("sum" corresponding to OR, and "product" corresponding to AND).

We can construct the truth table for Y in a systematic manner:

Consider a function Y of three variables A, B, C:

$$Y = (A + B + C) \cdot (A + B + \overline{C}) \cdot (\overline{A} + B + \overline{C}) \cdot (\overline{A} + \overline{B} + \overline{C})$$
$$\equiv Y_1 \cdot Y_2 \cdot Y_3 \cdot Y_4$$

This form is called the "product of sums" form ("sum" corresponding to OR, and "product" corresponding to AND).

We can construct the truth table for Y in a systematic manner:

(1) Enumerate all possible combinations of A, B, C.
Since each of A, B, C can take two values (0 or 1), we have 2^3 possibilities.

Consider a function Y of three variables A, B, C:

$$
\begin{aligned}
Y &= (A + B + C) \cdot (A + B + \overline{C}) \cdot (\overline{A} + B + \overline{C}) \cdot (\overline{A} + \overline{B} + \overline{C}) \\
&\equiv Y_1 \cdot Y_2 \cdot Y_3 \cdot Y_4
\end{aligned}
$$

This form is called the "product of sums" form ("sum" corresponding to OR, and "product" corresponding to AND).

We can construct the truth table for Y in a systematic manner:

(1) Enumerate all possible combinations of A, B, C.
Since each of A, B, C can take two values (0 or 1), we have 2^3 possibilities.

(2) Tabulate $Y_1 = A + B + C$, etc. Note that Y_1 is 0 only if $A = B = C = 0$;
Y_1 is 1 otherwise.

Consider a function Y of three variables A, B, C:

$$Y = (A+B+C)\cdot(A+B+\overline{C})\cdot(\overline{A}+B+\overline{C})\cdot(\overline{A}+\overline{B}+\overline{C})$$
$$\equiv Y_1 \cdot Y_2 \cdot Y_3 \cdot Y_4$$

This form is called the "product of sums" form ("sum" corresponding to OR, and "product" corresponding to AND).

We can construct the truth table for Y in a systematic manner:

(1) Enumerate all possible combinations of A, B, C.
Since each of A, B, C can take two values (0 or 1), we have 2^3 possibilities.

(2) Tabulate $Y_1 = A+B+C$, etc. Note that Y_1 is 0 only if $A=B=C=0$;
Y_1 is 1 otherwise.

(3) Since $Y = Y_1\,Y_2\,Y_3\,Y_4$,
Y is 0 if any of Y_1, Y_2, Y_3, Y_4 is 0; else Y is 1.
$\rightarrow$ tabulate Y.

"Product of sums" form

$$Y = Y_1 \, Y_2 \, Y_3 \, Y_4 = (A + B + C)\,(A + B + \overline{C})\,(\overline{A} + B + \overline{C})\,(\overline{A} + \overline{B} + \overline{C})$$

A	B	C	Y_1	Y_2	Y_3	Y_4	Y
0	0	0					
0	0	1					
0	1	0					
0	1	1					
1	0	0					
1	0	1					
1	1	0					
1	1	1					

"Product of sums" form

$$Y = Y_1\,Y_2\,Y_3\,Y_4 = (A + B + C)\,(A + B + \overline{C})\,(\overline{A} + B + \overline{C})\,(\overline{A} + \overline{B} + \overline{C})$$

A	B	C	Y_1	Y_2	Y_3	Y_4	Y
0	0	0	0				
0	0	1					
0	1	0					
0	1	1					
1	0	0					
1	0	1					
1	1	0					
1	1	1					

"Product of sums" form

$$Y = Y_1\,Y_2\,Y_3\,Y_4 = (A + B + C)\,(A + B + \overline{C})\,(\overline{A} + B + \overline{C})\,(\overline{A} + \overline{B} + \overline{C})$$

A	B	C	Y_1	Y_2	Y_3	Y_4	Y
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				

"Product of sums" form

$$Y = Y_1\,Y_2\,Y_3\,Y_4 = (A + B + C)\,(A + B + \overline{C})\,(\overline{A} + B + \overline{C})\,(\overline{A} + \overline{B} + \overline{C})$$

A	B	C	Y_1	Y_2	Y_3	Y_4	Y
0	0	0	0				
0	0	1	1	0			
0	1	0	1				
0	1	1	1				
1	0	0	1				
1	0	1	1				
1	1	0	1				
1	1	1	1				

"Product of sums" form

$$Y = Y_1\,Y_2\,Y_3\,Y_4 = (A + B + C)\,(A + B + \overline{C})\,(\overline{A} + B + \overline{C})\,(\overline{A} + \overline{B} + \overline{C})$$

A	B	C	Y_1	Y_2	Y_3	Y_4	Y
0	0	0	0	1			
0	0	1	1	0			
0	1	0	1	1			
0	1	1	1	1			
1	0	0	1	1			
1	0	1	1	1			
1	1	0	1	1			
1	1	1	1	1			

"Product of sums" form

$$Y = Y_1\,Y_2\,Y_3\,Y_4 = (A + B + C)\,(A + B + \overline{C})\,(\overline{A} + B + \overline{C})\,(\overline{A} + \overline{B} + \overline{C})$$

A	B	C	Y_1	Y_2	Y_3	Y_4	Y
0	0	0	0	1			
0	0	1	1	0			
0	1	0	1	1			
0	1	1	1	1			
1	0	0	1	1			
1	0	1	1	1	0		
1	1	0	1	1			
1	1	1	1	1			

“Product of sums” form

$$Y = Y_1\,Y_2\,Y_3\,Y_4 = (A + B + C)\,(A + B + \overline{C})\,(\overline{A} + B + \overline{C})\,(\overline{A} + \overline{B} + \overline{C})$$

A	B	C	Y_1	Y_2	Y_3	Y_4	Y
0	0	0	0	1	1		
0	0	1	1	0	1		
0	1	0	1	1	1		
0	1	1	1	1	1		
1	0	0	1	1	1		
1	0	1	1	1	0		
1	1	0	1	1	1		
1	1	1	1	1	1		

“Product of sums” form

$$Y = Y_1 Y_2 Y_3 Y_4 = (A + B + C)(A + B + \overline{C})(\overline{A} + B + \overline{C})(\overline{A} + \overline{B} + \overline{C})$$

A	B	C	Y_1	Y_2	Y_3	Y_4	Y
0	0	0	0	1	1		
0	0	1	1	0	1		
0	1	0	1	1	1		
0	1	1	1	1	1		
1	0	0	1	1	1		
1	0	1	1	1	0		
1	1	0	1	1	1		
1	1	1	1	1	1	0	

"Product of sums" form

$$Y = Y_1 Y_2 Y_3 Y_4 = (A + B + C)(A + B + \overline{C})(\overline{A} + B + \overline{C})(\overline{A} + \overline{B} + \overline{C})$$

A	B	C	Y_1	Y_2	Y_3	Y_4	Y
0	0	0	0	1	1	1	
0	0	1	1	0	1	1	
0	1	0	1	1	1	1	
0	1	1	1	1	1	1	
1	0	0	1	1	1	1	
1	0	1	1	1	0	1	
1	1	0	1	1	1	1	
1	1	1	1	1	1	0	

$$Y = Y_1\,Y_2\,Y_3\,Y_4 = (A + B + C)\,(A + B + \overline{C})\,(\overline{A} + B + \overline{C})\,(\overline{A} + \overline{B} + \overline{C})$$

A	B	C	Y_1	Y_2	Y_3	Y_4	Y
0	0	0	0	1	1	1	0
0	0	1	1	0	1	1	0
0	1	0	1	1	1	1	
0	1	1	1	1	1	1	
1	0	0	1	1	1	1	
1	0	1	1	1	0	1	0
1	1	0	1	1	1	1	
1	1	1	1	1	1	0	0

"Product of sums" form

$$Y = Y_1 Y_2 Y_3 Y_4 = (A + B + C)\,(A + B + \overline{C})\,(\overline{A} + B + \overline{C})\,(\overline{A} + \overline{B} + \overline{C})$$

A	B	C	Y_1	Y_2	Y_3	Y_4	Y
0	0	0	0	1	1	1	0
0	0	1	1	0	1	1	0
0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1
1	0	1	1	1	0	1	0
1	1	0	1	1	1	1	1
1	1	1	1	1	1	0	0

"Product of sums" form

$$Y = Y_1\,Y_2\,Y_3\,Y_4 = (A + B + C)\,(A + B + \overline{C})\,(\overline{A} + B + \overline{C})\,(\overline{A} + \overline{B} + \overline{C})$$

A	B	C	Y_1	Y_2	Y_3	Y_4	Y
0	0	0	0	1	1	1	0
0	0	1	1	0	1	1	0
0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1
1	0	1	1	1	0	1	0
1	1	0	1	1	1	1	1
1	1	1	1	1	1	0	0

Note that Y is identical to X (seen two slides back). This is an example of how the same function can be written in two seemingly different forms (in this case, the sum-of-products form and the product-of-sums form).

Consider a function X of three variables A, B, C:
$X = A\,B\,\overline{C} + \overline{A}\,B\,C + \overline{A}\,B\,\overline{C}$

Consider a function X of three variables A, B, C:

$X = A\,B\,\overline{C} + \overline{A}\,B\,C + \overline{A}\,B\,\overline{C}$

This form is called the *standard* sum-of-products form, and each individual term (consisting of all three variables) is called a "minterm."

Consider a function X of three variables A, B, C:

$X = A\,B\,\overline{C} + \overline{A}\,B\,C + \overline{A}\,B\,\overline{C}$

This form is called the *standard* sum-of-products form, and each individual term (consisting of all three variables) is called a "minterm."

In the truth table for X, the numbers of 1s is the same as the number of minterms, as we have seen in an example.

Standard sum-of-products form

Consider a function X of three variables A, B, C:

$X = A\,B\,\overline{C} + \overline{A}\,B\,C + \overline{A}\,B\,\overline{C}$

This form is called the *standard* sum-of-products form, and each individual term (consisting of all three variables) is called a "minterm."

In the truth table for X, the numbers of 1s is the same as the number of minterms, as we have seen in an example.

X can be rewritten as,

$$\begin{aligned} X &= A\,B\,\overline{C} + \overline{A}\,B\,(C + \overline{C}) \\ &= A\,B\,\overline{C} + \overline{A}\,B. \end{aligned}$$

Consider a function X of three variables A, B, C:

$$X = A\,B\,\overline{C} + \overline{A}\,B\,C + \overline{A}\,B\,\overline{C}$$

This form is called the *standard* sum-of-products form, and each individual term (consisting of all three variables) is called a "minterm."

In the truth table for X, the numbers of 1s is the same as the number of minterms, as we have seen in an example.

X can be rewritten as,

$$\begin{aligned} X &= A\,B\,\overline{C} + \overline{A}\,B\,(C + \overline{C}) \\ &= A\,B\,\overline{C} + \overline{A}\,B. \end{aligned}$$

This is also a sum-of-products form, but not the standard one.

Consider a function X of three variables A, B, C:

$X = (A + B + C)\,(\overline{A} + B + C)\,(\overline{A} + \overline{B} + C)$

Consider a function X of three variables A, B, C:

$X = (A + B + C)\,(\overline{A} + B + C)\,(\overline{A} + \overline{B} + C)$

This form is called the *standard* product-of-sums form, and each individual term (consisting of all three variables) is called a "maxterm."

Standard product-of-sums form

Consider a function X of three variables A, B, C:

$X = (A + B + C)\,(\overline{A} + B + C)\,(\overline{A} + \overline{B} + C)$

This form is called the *standard* product-of-sums form, and each individual term (consisting of all three variables) is called a "maxterm."

In the truth table for X, the numbers of 0s is the same as the number of maxterms, as we have seen in an example.

Consider a function X of three variables A, B, C:

$X = (A + B + C)\,(\overline{A} + B + C)\,(\overline{A} + \overline{B} + C)$

This form is called the *standard* product-of-sums form, and each individual term (consisting of all three variables) is called a "maxterm."

In the truth table for X, the numbers of 0s is the same as the number of maxterms, as we have seen in an example.

X can be rewritten as,

$$\begin{aligned} X &= (A + B + C)\,(\overline{A} + B + C)\,(\overline{A} + \overline{B} + C) \\ &= (A + B + C)\,(\overline{A} + C + B)\,(\overline{A} + C + \overline{B}) \\ &= (A + B + C)\,(\overline{A} + C + B\,\overline{B}) \\ &= (A + B + C)\,(\overline{A} + C). \end{aligned}$$

Consider a function X of three variables A, B, C:

$X = (A + B + C)\,(\overline{A} + B + C)\,(\overline{A} + \overline{B} + C)$

This form is called the *standard* product-of-sums form, and each individual term (consisting of all three variables) is called a "maxterm."

In the truth table for X, the numbers of 0s is the same as the number of maxterms, as we have seen in an example.

X can be rewritten as,

$$\begin{aligned} X &= (A + B + C)\,(\overline{A} + B + C)\,(\overline{A} + \overline{B} + C) \\ &= (A + B + C)\,(\overline{A} + C + B)\,(\overline{A} + C + \overline{B}) \\ &= (A + B + C)\,(\overline{A} + C + B\,\overline{B}) \\ &= (A + B + C)\,(\overline{A} + C). \end{aligned}$$

This is also a product-of-sums form, but not the standard one.

The "don't care" condition

I want to design a box (with inputs A, B, C, and output S) which will help in scheduling my appointments.

$A \equiv$ I am in town, and the time slot being suggested for the appointment is free.

$B \equiv$ My favourite player is scheduled to play a match (which I can watch on TV).

$C \equiv$ The appointment is crucial for my business.

$S \equiv$ Schedule the appointment.

I want to design a box (with inputs A, B, C, and output S) which will help in scheduling my appointments.

$A \equiv$ I am in town, and the time slot being suggested for the appointment is free.

$B \equiv$ My favourite player is scheduled to play a match (which I can watch on TV).

$C \equiv$ The appointment is crucial for my business.

$S \equiv$ Schedule the appointment.

The following truth table summarizes the expected functioning of the box.

A	B	C	S
0	X	X	0
1	0	X	1
1	1	0	0
1	1	1	1

The "don't care" condition

I want to design a box (with inputs A, B, C, and output S) which will help in scheduling my appointments.

$A \equiv$ I am in town, and the time slot being suggested for the appointment is free.

$B \equiv$ My favourite player is scheduled to play a match (which I can watch on TV).

$C \equiv$ The appointment is crucial for my business.

$S \equiv$ Schedule the appointment.

The following truth table summarizes the expected functioning of the box.

A	B	C	S
0	X	X	0
1	0	X	1
1	1	0	0
1	1	1	1

Note that we have a new entity called X in the truth table.

The “don’t care” condition

I want to design a box (with inputs A, B, C, and output S) which will help in scheduling my appointments.

$A \equiv$ I am in town, and the time slot being suggested for the appointment is free.

$B \equiv$ My favourite player is scheduled to play a match (which I can watch on TV).

$C \equiv$ The appointment is crucial for my business.

$S \equiv$ Schedule the appointment.

The following truth table summarizes the expected functioning of the box.

A	B	C	S
0	X	X	0
1	0	X	1
1	1	0	0
1	1	1	1

Note that we have a new entity called X in the truth table.

X can be 0 or 1 (it does not matter) and is therefore called the “don’t care” condition.

I want to design a box (with inputs A, B, C, and output S) which will help in scheduling my appointments.

$A \equiv$ I am in town, and the time slot being suggested for the appointment is free.

$B \equiv$ My favourite player is scheduled to play a match (which I can watch on TV).

$C \equiv$ The appointment is crucial for my business.

$S \equiv$ Schedule the appointment.

The following truth table summarizes the expected functioning of the box.

A	B	C	S
0	X	X	0
1	0	X	1
1	1	0	0
1	1	1	1

Note that we have a new entity called X in the truth table.

X can be 0 or 1 (it does not matter) and is therefore called the “don’t care” condition.

Don’t care conditions can often be used to get a more efficient implementation of a logical function.

* A Karnaugh map ("K-map") is a representation of the truth table of a logical function.

* A Karnaugh map ("K-map") is a representation of the truth table of a logical function.
* A K-map can be used to obtain a "minimal" expression of a function in the sum-of-products form or in the product-of-sums form.

* A Karnaugh map ("K-map") is a representation of the truth table of a logical function.
* A K-map can be used to obtain a "minimal" expression of a function in the sum-of-products form or in the product-of-sums form.
* A "minimal" expression has a minimum number of terms, each with a minimum number of variables. (For some functions, it is possible to have more than one minimal expressions, i.e., more than one expressions with the same complexity.)

* A Karnaugh map ("K-map") is a representation of the truth table of a logical function.
* A K-map can be used to obtain a "minimal" expression of a function in the sum-of-products form or in the product-of-sums form.
* A "minimal" expression has a minimum number of terms, each with a minimum number of variables. (For some functions, it is possible to have more than one minimal expressions, i.e., more than one expressions with the same complexity.)
* A minimal expression can be implemented with fewer gates.

K-maps

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	X
1	0	1	0
1	1	0	0
1	1	1	1

K-maps

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	X
1	0	1	0
1	1	0	0
1	1	1	1

C \ AB	00	01	11	10
0				
1				

K-maps

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	X
1	0	1	0
1	1	0	0
1	1	1	1

C \ AB	00	01	11	10
0				
1	1			

K-maps

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	X
1	0	1	0
1	1	0	0
1	1	1	1

C \ AB	00	01	11	10
0		1		
1	1			

K-maps

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	X
1	0	1	0
1	1	0	0
1	1	1	1

C \ AB	00	01	11	10
0		1		X
1	1			

K-maps

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	X
1	0	1	0
1	1	0	0
1	1	1	1

C \ AB	00	01	11	10
0		1		X
1	1		1	

K-maps

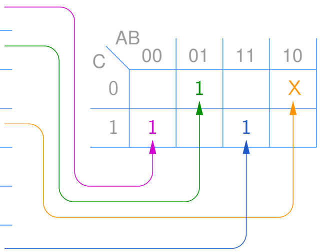
A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	X
1	0	1	0
1	1	0	0
1	1	1	1

C \ AB	00	01	11	10
0		1		X
1	1		1	

$\Rightarrow$

C \ AB	00	01	11	10
0	0	1	0	X
1	1	0	1	0

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	X
1	0	1	0
1	1	0	0
1	1	1	1



C \ AB	00	01	11	10
0	0	1	0	X
1	1	0	1	0

* A K-map is the same as the truth table of a function except for the way the entries are arranged.

K-maps

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	X
1	0	1	0
1	1	0	0
1	1	1	1

C \ AB	00	01	11	10
0		1		X
1	1		1	

$\Rightarrow$

C \ AB	00	01	11	10
0	0	1	0	X
1	1	0	1	0

* A K-map is the same as the truth table of a function except for the way the entries are arranged.
* In a K-map, the adjacent rows or columns differ only in *one* variable. For example, in going from the column $A\,B = 01$ to $A\,B = 11$, there is only one change, viz., $A = 0 \rightarrow A = 1$.

K-maps: example with four variables

A	B	C	D	Y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	X
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

$\Rightarrow$

CD \ AB	00	01	11	10
00	0	0	0	0
01	1	1	0	X
11	1	0	1	1
10	1	0	0	0

K-maps

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

$$X_1 = A\,B\,\overline{C}\,\overline{D}$$

K-maps

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

$$X_1 = A\,B\,\overline{C}\,\overline{D}$$

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

$$X_2 = \overline{A}\,\overline{C}\,D$$

K-maps

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

$X_1 = A\,B\,\overline{C}\,\overline{D}$

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

$X_2 = \overline{A}\,\overline{C}\,D$

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

$X_3 = A\,C$

K-maps

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

$$X_1 = A\,B\,\overline{C}\,\overline{D}$$

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

$$X_2 = \overline{A}\,\overline{C}\,D$$

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

$$X_3 = A\,C$$

*

No. of variables	No. of 1's
4	2^0
3	2^1
2	2^2

K-maps

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

$X_1 = A\,B\,\overline{C}\,\overline{D}$

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

$X_2 = \overline{A}\,\overline{C}\,D$

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

$X_3 = A\,C$

*

No. of variables	No. of 1's
4	2^0
3	2^1
2	2^2

* The 1's can be enclosed by a rectangle in each case.

K-maps

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

$$X_1 = A\,B\,\overline{C}\,\overline{D}$$

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

$$X_2 = \overline{A}\,\overline{C}\,D$$

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

$$X_3 = A\,C$$

K-maps

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

$X_1 = A\,B\,\overline{C}\,\overline{D}$

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

$X_2 = \overline{A}\,\overline{C}\,D$

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

$X_3 = A\,C$

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

$Y = X_1 + X_2 + X_3$

K-maps

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

$X_1 = A\,B\,\overline{C}\,\overline{D}$

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

$X_2 = \overline{A}\,\overline{C}\,D$

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

$X_3 = A\,C$

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

$Y = X_1 + X_2 + X_3$

* We are interested in identifying a *minimal* expression from the given K-map.

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

$X_1 = A\,B\,\overline{C}\,\overline{D}$

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

$X_2 = \overline{A}\,\overline{C}\,D$

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

$X_3 = A\,C$

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

$Y = X_1 + X_2 + X_3$

* We are interested in identifying a *minimal* expression from the given K-map.
* Minimal: smallest number of terms, smallest number of variables in each term
  $\rightarrow$ smallest number of rectangles containing 2^k 1's, each as large as possible

K-maps

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

What is the logical function (Y) represented by this K-map?

K-maps

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

What is the logical function (Y) represented by this K-map?

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

What is the logical function (Y) represented by this K-map?

* There are 2^1 1's forming a rectangle $\rightarrow$ we can combine them.

K-maps

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

What is the logical function (Y) represented by this K-map?

* There are 2^1 1's forming a rectangle $\rightarrow$ we can combine them.
* The product term is 1 if $B = 1$, and $C = 0$.

K-maps

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

What is the logical function (Y) represented by this K-map?

* There are 2^1 1's forming a rectangle $\rightarrow$ we can combine them.
* The product term is 1 if $B = 1$, and $C = 0$.
* The product term does not depend on A.

K-maps

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

What is the logical function (Y) represented by this K-map?

* There are 2^1 1's forming a rectangle $\rightarrow$ we can combine them.
* The product term is 1 if $B = 1$, and $C = 0$.
* The product term does not depend on A.

$\rightarrow Y = B\,\overline{C}$

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

Can the 1s shown in the K-map be combined?

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

Can the 1s shown in the K-map be combined?

Although the number of 1's is a power of 2 (2^1), they cannot be combined because they are not adjacent (i.e., they do not form a rectangle).

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

Can the 1s shown in the K-map be combined?

Although the number of 1's is a power of 2 (2^1), they cannot be combined because they are not adjacent (i.e., they do not form a rectangle).

$\rightarrow$ the function $(A\,B\,\overline{C} + A\,\overline{B}\,C)$ cannot be minimized.

K-maps

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

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

Can the 1's shown in the K-map be combined?

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

Can the 1's shown in the K-map be combined?

Let us redraw the K-map by changing the order of the columns cyclically.

K-maps

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

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

Can the 1's shown in the K-map be combined?

Let us redraw the K-map by changing the order of the columns cyclically.

K-maps

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

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

Can the 1's shown in the K-map be combined?

Let us redraw the K-map by changing the order of the columns cyclically.

The two 1's are, in fact, adjacent and can be combined to give $\overline{B}\,\overline{C}$.

K-maps

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

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

Can the 1's shown in the K-map be combined?

Let us redraw the K-map by changing the order of the columns cyclically.

The two 1's are, in fact, adjacent and can be combined to give $\overline{B}\,\overline{C}$.

K-maps

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

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

Can the 1's shown in the K-map be combined?

Let us redraw the K-map by changing the order of the columns cyclically.

The two 1's are, in fact, adjacent and can be combined to give $\overline{B}\,\overline{C}$.

$\rightarrow$ Columns $A\,B = 00$ and $A\,B = 10$ in the K-map on the left are indeed "logically adjacent" (although they are not geometrically adjacent) since they differ only in one variable (A).

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

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

Can the 1's shown in the K-map be combined?

Let us redraw the K-map by changing the order of the columns cyclically.

The two 1's are, in fact, adjacent and can be combined to give $\overline{B}\,\overline{C}$.

$\rightarrow$ Columns $A\,B=00$ and $A\,B=10$ in the K-map on the left are indeed "logically adjacent" (although they are not geometrically adjacent) since they differ only in one variable (A).

We could have therefore combined the 1's without actually redrawing the K-map.

K-maps

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

K-maps

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

K-maps

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

$$\Rightarrow \quad X_1 = \overline{B}\,\overline{D}$$

K-maps

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

$$\Rightarrow \quad X_1 = \overline{B}\,\overline{D}$$

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

K-maps

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

$$X_1 = \overline{B}\,\overline{D}$$

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

K-maps

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

$$X_1 = \overline{B}\,\overline{D}$$

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

$$X_2 = \overline{B}\,\overline{C}$$

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

Standard sum-of-products form:

$$X_1 = \underline{A\,B\,\overline{C}\,\overline{D}} + \underline{A\,B\,\overline{C}\,D} + \underline{\overline{A}\,B\,\overline{C}\,D}$$

Since the number of minterms is not a power of 2, they cannot be combined into a single term; however, they can be combined into two terms:

K-maps

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

Standard sum-of-products form:

$$X_1 = \underline{A\,B\,\overline{C}\,\overline{D}} + \underline{A\,B\,\overline{C}\,D} + \underline{\overline{A}\,B\,\overline{C}\,D}$$

Since the number of minterms is not a power of 2, they cannot be combined into a single term; however, they can be combined into two terms:

$$\begin{aligned} X_1 &= A\,B\,\overline{C}\,\overline{D} + A\,B\,\overline{C}\,D + A\,B\,\overline{C}\,D + \overline{A}\,B\,\overline{C}\,D \qquad \text{(using Y=Y+Y)} \\ &= A\,B\,\overline{C}\,(\overline{D} + D) + B\,\overline{C}\,D\,(A + \overline{A}) \\ &= A\,B\,\overline{C} + B\,\overline{C}\,D \end{aligned}$$

K-maps

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

Standard sum-of-products form:

$$X_1 = \underline{A\,B\,\overline{C}\,\overline{D}} + \underline{A\,B\,\overline{C}\,D} + \underline{\overline{A}\,B\,\overline{C}\,D}$$

Since the number of minterms is not a power of 2, they cannot be combined into a single term; however, they can be combined into two terms:

$$\begin{aligned} X_1 &= A\,B\,\overline{C}\,\overline{D} + A\,B\,\overline{C}\,D + A\,B\,\overline{C}\,D + \overline{A}\,B\,\overline{C}\,D \qquad \text{(using Y=Y+Y)} \\ &= A\,B\,\overline{C}\,(\overline{D} + D) + B\,\overline{C}\,D\,(A + \overline{A}) \\ &= A\,B\,\overline{C} + B\,\overline{C}\,D \end{aligned}$$

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

Standard sum-of-products form:

$$X_1 = \underline{A\,B\,\overline{C}\,\overline{D}} + \underline{A\,B\,\overline{C}\,D} + \underline{\overline{A}\,B\,\overline{C}\,D}$$

Since the number of minterms is not a power of 2, they cannot be combined into a single term; however, they can be combined into two terms:

$$\begin{aligned} X_1 &= A\,B\,\overline{C}\,\overline{D} + A\,B\,\overline{C}\,D + A\,B\,\overline{C}\,D + \overline{A}\,B\,\overline{C}\,D \qquad \text{(using Y=Y+Y)} \\ &= A\,B\,\overline{C}\,(\overline{D} + D) + B\,\overline{C}\,D\,(A + \overline{A}) \\ &= A\,B\,\overline{C} + B\,\overline{C}\,D \end{aligned}$$

X_1:

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

K-maps

X_1:

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

K-maps

X_1:

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

K-maps

X_1:

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

K-maps

X_1:

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

X_1:

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

$$\Rightarrow \quad X_1 = \overline{A}\,\overline{C} + \overline{B}\,\overline{C} + A\,B\,C\,D + A\,\overline{B}\,\overline{D}$$

Z:

CD \ AB	00	01	11	10
00	0	0	X	0
01	1	1	0	0
11	0	0	0	0
10	1	X	1	1

K-maps

Z:

CD \ AB	00	01	11	10
00	0	0	X	0
01	1	1	0	0
11	0	0	0	0
10	1	X	1	1

Since X represents a "don't care" condition, we can assign 0 or 1 to the corresponding minterm to arrive at a minimal expression.

Z:

CD \ AB	00	01	11	10
00	0	0	X	0
01	1	1	0	0
11	0	0	0	0
10	1	X	1	1

$\Rightarrow$

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

Since X represents a “don’t care” condition, we can assign 0 or 1 to the corresponding minterm to arrive at a minimal expression.

K-maps

Z:

CD \ AB	00	01	11	10
00	0	0	X	0
01	1	1	0	0
11	0	0	0	0
10	1	X	1	1

$\Rightarrow$

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

Since X represents a "don't care" condition, we can assign 0 or 1 to the corresponding minterm to arrive at a minimal expression.

K-maps

Z:

CD \ AB	00	01	11	10
00	0	0	X	0
01	1	1	0	0
11	0	0	0	0
10	1	X	1	1

$\Rightarrow$

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

$\Rightarrow$

$$Z = C\,\overline{D} + \overline{A}\,\overline{C}\,D$$

Since X represents a "don't care" condition, we can assign 0 or 1 to the corresponding minterm to arrive at a minimal expression.

Binary numbers

$\boxed{\text{Decimal (base 10) system}}$

$$3\ 1\ 7 = 3 \times 10^2 + 1 \times 10^1 + 7 \times 10^0$$

10^2 10^1 10^0

$\boxed{\text{Decimal (base 10) system}}$

$$3\ 1\ 7 = 3 \times 10^2 + 1 \times 10^1 + 7 \times 10^0$$

10^2 10^1 10^0

* Digits: 0,1,2,..,9

* example: 4 1 5 3

most significant digit

least significant digit

Binary numbers

Decimal (base 10) system

$$3\ 1\ 7 = 3 \times 10^2 + 1 \times 10^1 + 7 \times 10^0$$

10^2 10^1 10^0

* Digits: 0,1,2,..,9

* example: 4 1 5 3

most significant digit

least significant digit

Binary (base 2) system

$$1\ 0\ 1\ 1\ 1 = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$
$$= 23 \text{ (in decimal)}$$

2^4 2^3 2^2 2^1 2^0

Decimal (base 10) system

$3\ 1\ 7 = 3 \times 10^2 + 1 \times 10^1 + 7 \times 10^0$

10^2 10^1 10^0

* Digits: 0,1,2,..,9

* example: 4 1 5 3

most significant digit

least significant digit

Binary (base 2) system

$1\ 0\ 1\ 1\ 1 = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

$= 23$ (in decimal)

2^4 2^3 2^2 2^1 2^0

* Bits: 0,1

* example: 1 0 0 1 1 0

most significant bit (MSB)

least significant bit (LSB)

Addition of binary numbers

Decimal (base 10) system

	10^4	10^3	10^2	10^1	10^0	weight
		3	1	7	9	first number
+						
		8	0	1	5	second number
	1			1		carry
	1	1	1	9	4	sum

Addition of binary numbers

Decimal (base 10) system

	10^4	10^3	10^2	10^1	10^0	
						weight
+		3	1	7	9	first number
		8	0	1	5	second number
	1			1		carry
	1	1	1	9	4	sum

Binary (base 2) system

	2^4	2^3	2^2	2^1	2^0	
						weight
+		1	0	1	1	first number (dec. 11)
		1	1	1	0	second number (dec. 14)
	1	1	1			carry
	1	1	0	0	1	sum (dec. 25)

Addition of binary numbers

Decimal (base 10) system

	10^4	10^3	10^2	10^1	10^0	weight
		3	1	7	9	first number
+		8	0	1	5	second number
	1			1		carry
	1	1	1	9	4	sum

Binary (base 2) system

	2^4	2^3	2^2	2^1	2^0	weight
		1	0	1	1	first number (dec. 11)
+		1	1	1	0	second number (dec. 14)
	1	1	1			carry
	1	1	0	0	1	sum (dec. 25)

* $0 + 1 = 1 + 0 = 1 \rightarrow \mathrm{S} = 1,\ \mathrm{C} = 0$

Addition of binary numbers

Decimal (base 10) system

	10^4	10^3	10^2	10^1	10^0	weight
		3	1	7	9	first number
+		8	0	1	5	second number
	1			1		carry
	1	1	1	9	4	sum

Binary (base 2) system

	2^4	2^3	2^2	2^1	2^0	weight
		1	0	1	1	first number (dec. 11)
+		1	1	1	0	second number (dec. 14)
	1	1	1			carry
	1	1	0	0	1	sum (dec. 25)

* $0 + 1 = 1 + 0 = 1 \rightarrow \mathsf{S} = 1,\ \mathsf{C} = 0$
* $1 + 1 = 10$ (dec. 2) $\rightarrow \mathsf{S} = 0,\ \mathsf{C} = 1$

Addition of binary numbers

Decimal (base 10) system

	10^4	10^3	10^2	10^1	10^0	
		3	1	7	9	first number
+		8	0	1	5	second number
	1			1		carry
	1	1	1	9	4	sum

Column headings are labelled "weight".

Binary (base 2) system

	2^4	2^3	2^2	2^1	2^0	
		1	0	1	1	first number (dec. 11)
+		1	1	1	0	second number (dec. 14)
	1	1	1			carry
	1	1	0	0	1	sum (dec. 25)

Column headings are labelled "weight".

* $0+1=1+0=1 \rightarrow S=1,\ C=0$
* $1+1=10$ (dec. 2) $\rightarrow S=0,\ C=1$
* $1+1+1=11$ (dec. 3) $\rightarrow S=1, C=1$

Addition of binary numbers

example

2^4	2^3	2^2	2^1	2^0	weight
	1	0	1	1	first number
+	1	1	1	0	second number
1	1	1	0	–	carry
1	1	0	0	1	sum

Addition of binary numbers

example

	2^4	2^3	2^2	2^1	2^0	weight
		1	0	1	1	first number
$+$						
		1	1	1	0	second number
	1	1	1	0	–	carry
	1	1	0	0	1	sum

general procedure

		2^N		2^2	2^1	2^0	weight
		A_N	$\cdots$	A_2	A_1	A_0	first number
$+$							
		B_N	$\cdots$	B_2	B_1	B_0	second number
	C_N	C_{N-1}	$\cdots$	C_1	C_0		carry
		S_N		S_2	S_1	S_0	sum

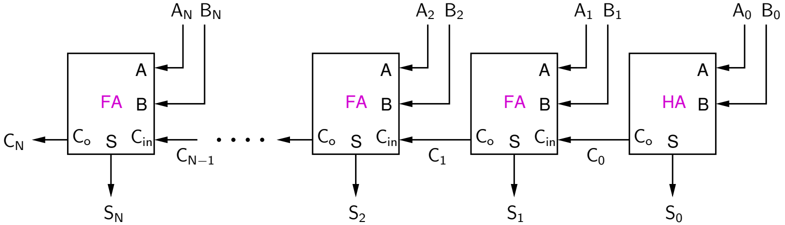
Addition of binary numbers

example

	2^4	2^3	2^2	2^1	2^0	weight
		1	0	1	1	first number
+		1	1	1	0	second number
	1	1	1	0	–	carry
	1	1	0	0	1	sum

general procedure

		2^N		2^2	2^1	2^0	weight
		A_N	$\cdots$	A_2	A_1	A_0	first number
+		B_N	$\cdots$	B_2	B_1	B_0	second number
	C_N	C_{N-1}	$\cdots$	C_1	C_0		carry
		S_N		S_2	S_1	S_0	sum



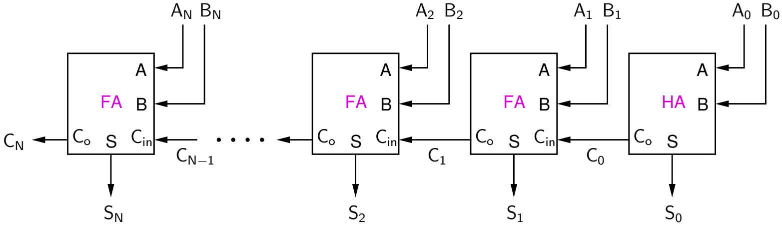
Addition of binary numbers

example

	2^4	2^3	2^2	2^1	2^0	weight
		1	0	1	1	first number
+		1	1	1	0	second number
	1	1	1	0	–	carry
	1	1	0	0	1	sum

general procedure

		2^N		2^2	2^1	2^0	weight
		A_N	$\cdots$	A_2	A_1	A_0	first number
+		B_N	$\cdots$	B_2	B_1	B_0	second number
	C_N	C_{N-1}	$\cdots$	C_1	C_0		carry
		S_N		S_2	S_1	S_0	sum



* The rightmost block (corresponding to the LSB) adds two bits A_0 and B_0; there is no input carry. This block is called a "half adder."

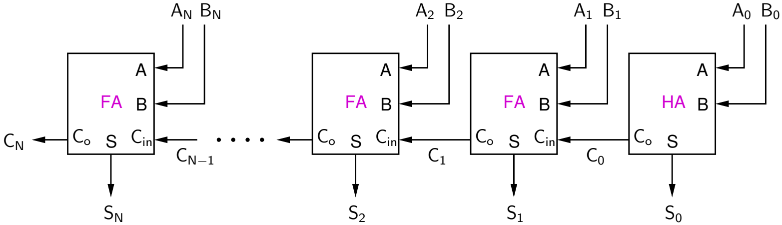
Addition of binary numbers

example

	2^4	2^3	2^2	2^1	2^0	weight
+		1	0	1	1	first number
		1	1	1	0	second number
	1	1	1	0	–	carry
	1	1	0	0	1	sum

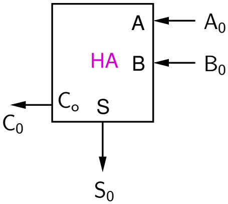
general procedure

		2^N		2^2	2^1	2^0	weight
+		A_N	$\cdots$	A_2	A_1	A_0	first number
		B_N	$\cdots$	B_2	B_1	B_0	second number
	C_N	C_{N-1}	$\cdots$	C_1	C_0		carry
		S_N		S_2	S_1	S_0	sum



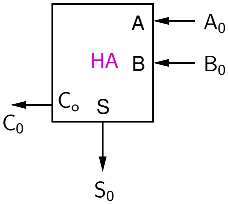
* The rightmost block (corresponding to the LSB) adds two bits A_0 and B_0; there is no input carry. This block is called a “half adder.”
* Each of the subsequent blocks adds three bits (A_i, B_i, C_{i-1}) and is called a “full adder.”

Half adder implementation



A	B	C_o	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Half adder implementation

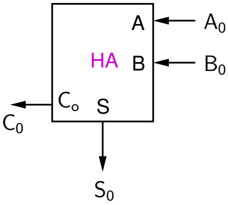


A	B	C_o	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

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

$$C_o = A\,B$$

Half adder implementation

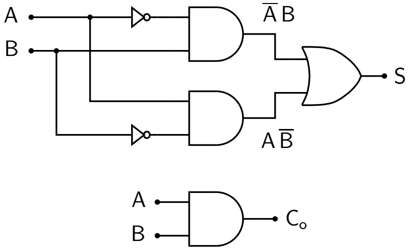


A	B	C_o	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

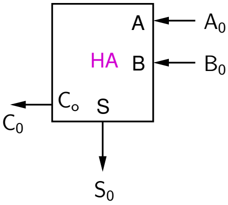
$$S = \overline{A}\,B + A\,\overline{B} = A \oplus B$$

$$C_o = A\,B$$

Implementation 1



Half adder implementation

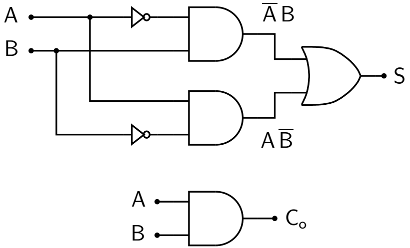


A	B	C_o	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

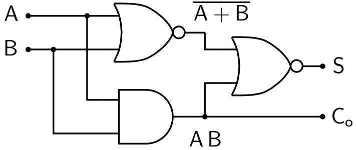
$$S = \overline{A}\,B + A\,\overline{B} = A \oplus B$$

$$C_o = A\,B$$

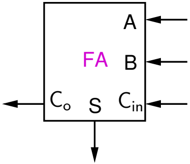
Implementation 1



Implementation 2

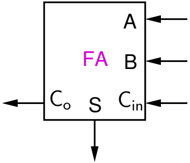


Full adder implementation



A	B	C_{in}	C_o	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Full adder implementation



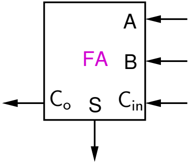
A	B	C_{in}	C_o	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

S:

C_{in} \ AB	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$$S = \overline{A}\,B\,\overline{C}_{in} + A\,\overline{B}\,\overline{C}_{in} + \overline{A}\,\overline{B}\,C_{in} + A\,B\,C_{in}$$

Full adder implementation



A	B	C_{in}	C_o	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

S:

C_{in} \ AB	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$$S = \overline{A}\,B\,\overline{C}_{in} + A\,\overline{B}\,\overline{C}_{in} + \overline{A}\,\overline{B}\,C_{in} + A\,B\,C_{in}$$

C_o:

C_{in} \ AB	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$$C_o = A\,B + B\,C_{in} + A\,C_{in}$$

Implementation of functions with only NAND gates

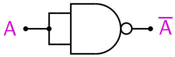
The NOT, AND, OR operations can be realised by using only NAND gates:

Implementation of functions with only NAND gates

The NOT, AND, OR operations can be realised by using only NAND gates:

NOT

$$\overline{A} = \overline{A \cdot A}$$

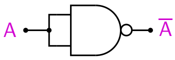


Implementation of functions with only NAND gates

The NOT, AND, OR operations can be realised by using only NAND gates:

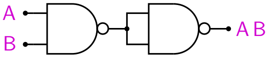
NOT

$$\overline{A} = \overline{A \cdot A}$$



AND

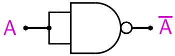
$$A \cdot B = \overline{\overline{A \cdot B}}$$



The NOT, AND, OR operations can be realised by using only NAND gates:

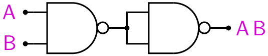
NOT

$$\overline{A} = \overline{A \cdot A}$$



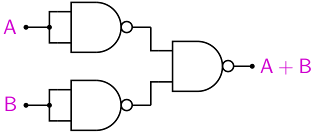
AND

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



OR

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



Implementation of functions with only NAND gates

Implement $\mathsf{Y} = \mathsf{A}\,\mathsf{B} + \mathsf{B}\,\mathsf{C}\,\overline{\mathsf{D}} + \overline{\mathsf{A}}\,\mathsf{D}$ using only NAND gates.

Implementation of functions with only NAND gates

Implement $Y = A\,B + B\,C\,\overline{D} + \overline{A}\,D$ using only NAND gates.

$$\overline{A} = \overline{A \cdot A}$$

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

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

Implementation of functions with only NAND gates

Implement $\mathsf{Y} = \mathsf{A\,B} + \mathsf{B\,C\,\overline{D}} + \mathsf{\overline{A}\,D}$ using only NAND gates.

$$\mathsf{Y} = \overline{\overline{\mathsf{A\,B}} \cdot \overline{\mathsf{B\,C\,\overline{D}}} \cdot \overline{\mathsf{\overline{A}\,D}}}$$

$$\overline{\mathsf{A}} = \overline{\mathsf{A} \cdot \mathsf{A}}$$

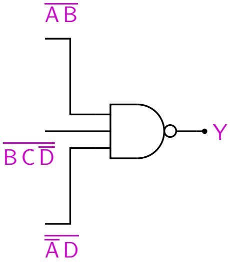
$$\mathsf{A} \cdot \mathsf{B} = \overline{\overline{\mathsf{A} \cdot \mathsf{B}}}$$

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

Implementation of functions with only NAND gates

Implement $Y = A\,B + B\,C\,\overline{D} + \overline{A}\,D$ using only NAND gates.

$$Y = \overline{\overline{A\,B} \cdot \overline{B\,C\,\overline{D}} \cdot \overline{\overline{A}\,D}}$$



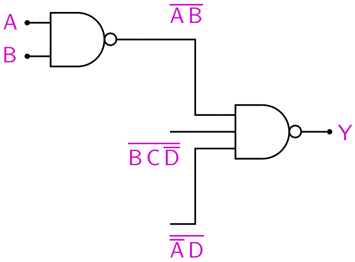
$$\overline{A} = \overline{A \cdot A}$$

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

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

Implement $Y = A\,B + B\,C\,\overline{D} + \overline{A}\,D$ using only NAND gates.

$$Y = \overline{\overline{A\,B} \cdot \overline{B\,C\,\overline{D}} \cdot \overline{\overline{A}\,D}}$$



$$\overline{A} = \overline{A \cdot A}$$

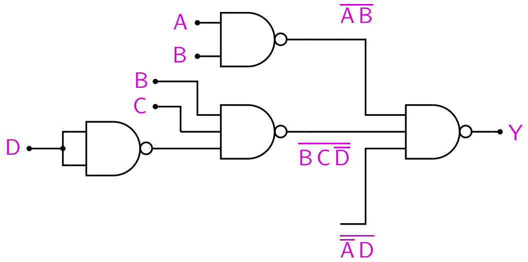
$$A \cdot B = \overline{\overline{A \cdot B}}$$

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

Implementation of functions with only NAND gates

Implement $Y = AB + BC\overline{D} + \overline{A}D$ using only NAND gates.

$$Y = \overline{\overline{AB} \cdot \overline{BC\overline{D}} \cdot \overline{\overline{A}D}}$$



$$\overline{A} = \overline{A \cdot A}$$

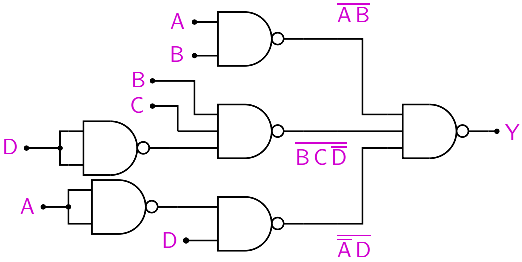
$$A \cdot B = \overline{\overline{A \cdot B}}$$

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

Implementation of functions with only NAND gates

Implement $Y = A\,B + B\,C\,\overline{D} + \overline{A}\,D$ using only NAND gates.

$$Y = \overline{\overline{A\,B} \cdot \overline{B\,C\,\overline{D}} \cdot \overline{\overline{A}\,D}}$$



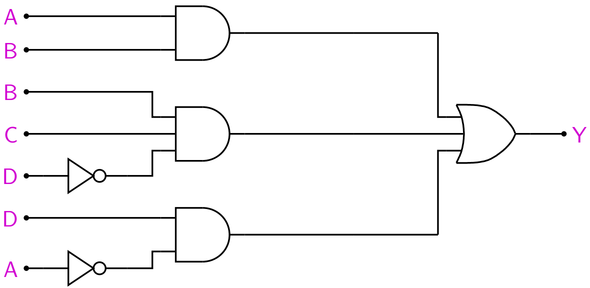
$$\overline{A} = \overline{A \cdot A}$$

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

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

Implementation of functions with only NAND gates

Implement $Y = A\,B + B\,C\,\overline{D} + \overline{A}\,D$ using only NAND gates.



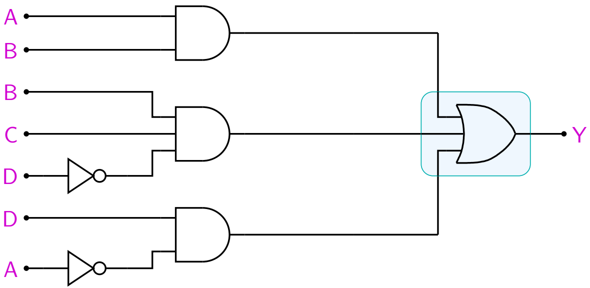
$\overline{A} = \overline{A \cdot A}$

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

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

Implementation of functions with only NAND gates

Implement $Y = A\,B + B\,C\,\overline{D} + \overline{A}\,D$ using only NAND gates.

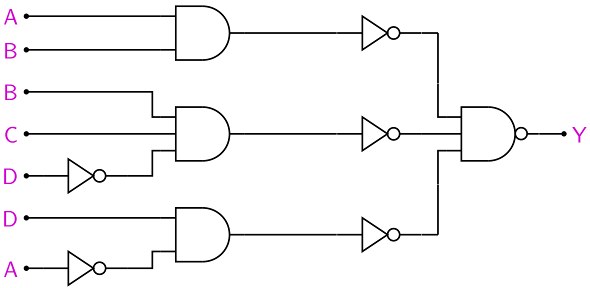


$\overline{A} = \overline{A \cdot A}$

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

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

Implement $Y = A\,B + B\,C\,\overline{D} + \overline{A}\,D$ using only NAND gates.

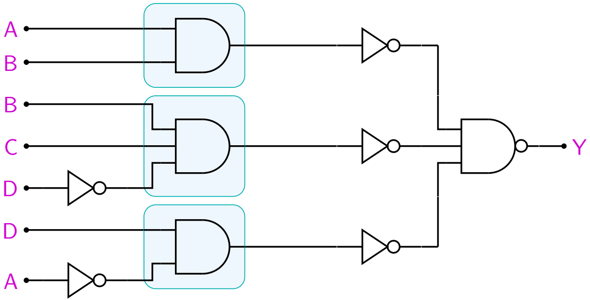


$\overline{A} = \overline{A \cdot A}$

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

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

Implement $Y = A\,B + B\,C\,\overline{D} + \overline{A}\,D$ using only NAND gates.



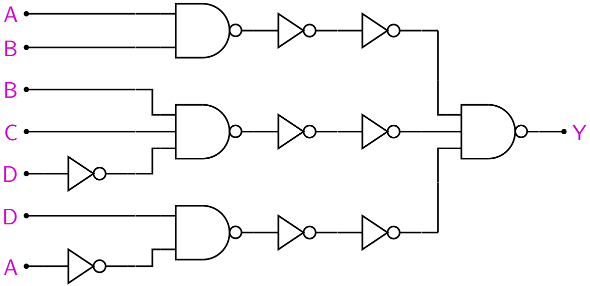
$$\overline{A} = \overline{A \cdot A}$$

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

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

Implementation of functions with only NAND gates

Implement $Y = A\,B + B\,C\,\overline{D} + \overline{A}\,D$ using only NAND gates.

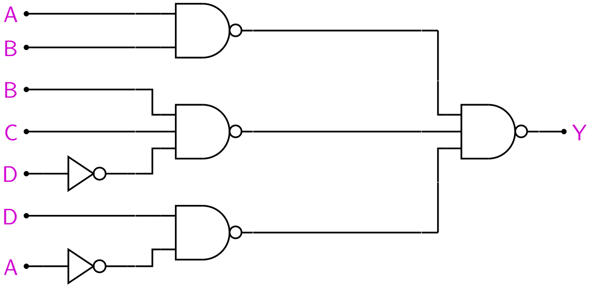


$$\overline{A} = \overline{A \cdot A}$$

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

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

Implement $\mathrm{Y} = \mathrm{A\,B} + \mathrm{B\,C\,\overline{D}} + \mathrm{\overline{A}\,D}$ using only NAND gates.

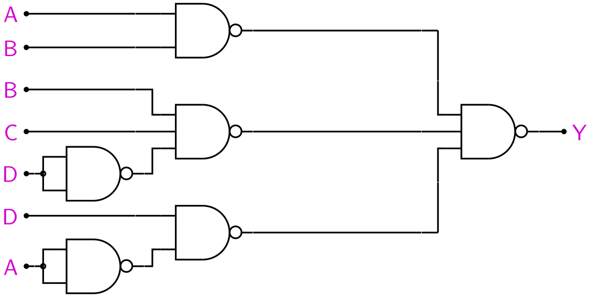


$\overline{\mathrm{A}} = \overline{\mathrm{A} \cdot \mathrm{A}}$

$\mathrm{A} \cdot \mathrm{B} = \overline{\overline{\mathrm{A} \cdot \mathrm{B}}}$

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

Implement $Y = A\,B + B\,C\,\overline{D} + \overline{A}\,D$ using only NAND gates.



$\overline{A} = \overline{A \cdot A}$

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

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

Implementation of functions with only NAND gates

Implement $Y = A + B + C$ using only 2-input NAND gates.

Implement $Y = A + B + C$ using only 2-input NAND gates.

$$\overline{A} = \overline{A \cdot A}$$

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

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

Implementation of functions with only NAND gates

Implement $Y = A + B + C$ using only 2-input NAND gates.

$$Y = (A + B) + C$$

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

$$\overline{A} = \overline{A \cdot A}$$

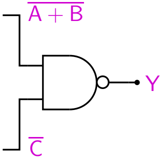
$$A \cdot B = \overline{\overline{A \cdot B}}$$

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

Implementation of functions with only NAND gates

Implement $Y = A + B + C$ using only 2-input NAND gates.

$$Y = (A + B) + C$$
$$= \overline{\overline{(A + B)} \cdot \overline{C}}$$



$$\overline{A} = \overline{A \cdot A}$$

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

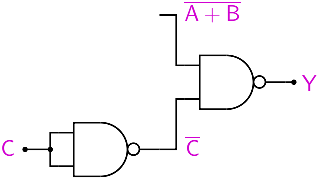
$$A + B = \overline{\overline{A} \cdot \overline{B}}$$

Implementation of functions with only NAND gates

Implement $Y = A + B + C$ using only 2-input NAND gates.

$$Y = (A + B) + C$$

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



$$\overline{A} = \overline{A \cdot A}$$

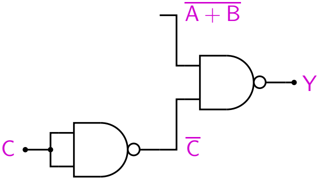
$$A \cdot B = \overline{\overline{A \cdot B}}$$

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

Implementation of functions with only NAND gates

Implement $Y = A + B + C$ using only 2-input NAND gates.

$$
\begin{aligned}
Y &= (A + B) + C \\
&= \overline{\overline{(A + B)} \cdot \overline{C}} \\
&= \overline{\overline{\overline{\overline{A} \cdot \overline{B}}} \cdot \overline{C}}
\end{aligned}
$$



$\overline{A} = \overline{A \cdot A}$

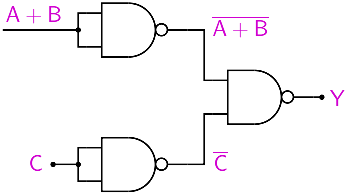
$A \cdot B = \overline{\overline{A \cdot B}}$

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

Implementation of functions with only NAND gates

Implement $Y = A + B + C$ using only 2-input NAND gates.

$$
\begin{aligned}
Y &= (A + B) + C \\
&= \overline{\overline{(A + B)} \cdot \overline{C}} \\
&= \overline{\overline{\overline{\overline{A} \cdot \overline{B}}} \cdot \overline{C}}
\end{aligned}
$$



$\overline{A} = \overline{A \cdot A}$

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

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

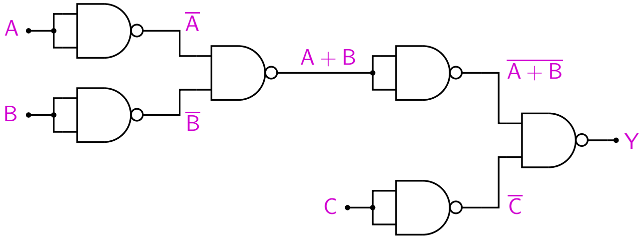
Implementation of functions with only NAND gates

Implement $Y = A + B + C$ using only 2-input NAND gates.

$$Y = (A + B) + C$$

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

$$= \overline{\overline{\overline{A} \cdot \overline{B}} \cdot \overline{C}}$$



$$\overline{A} = \overline{A \cdot A}$$

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

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

Implementation of functions with only NOR gates

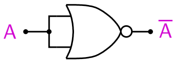
The NOT, AND, OR operations can be realised by using only NOR gates:

Implementation of functions with only NOR gates

The NOT, AND, OR operations can be realised by using only NOR gates:

NOT

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

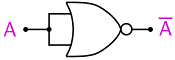


Implementation of functions with only NOR gates

The NOT, AND, OR operations can be realised by using only NOR gates:

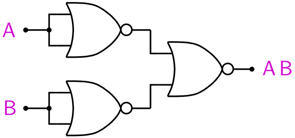
NOT

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



AND

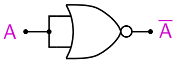
$$A \cdot B = \overline{\overline{A} + \overline{B}}$$



The NOT, AND, OR operations can be realised by using only NOR gates:

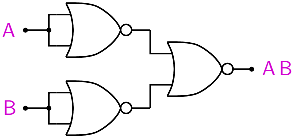
NOT

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



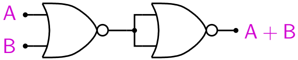
AND

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



OR

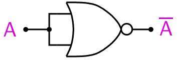
$$A + B = \overline{\overline{A + B}}$$

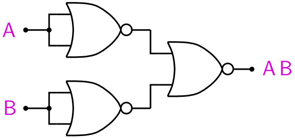


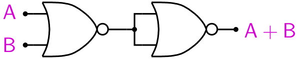
Implementation of functions with only NOR gates

The NOT, AND, OR operations can be realised by using only NOR gates:

NOT	AND	OR
$\overline{A} = \overline{A+A}$	$A \cdot B = \overline{\overline{A}+\overline{B}}$	$A+B = \overline{\overline{A+B}}$







Implementation of functions with only NOR (or only NAND) gates is more than a theoretical curiosity. There are chips which provide a "sea of gates" (say, NOR gates) which can be configured by the user (through programming) to implement functions.

Implement $Y = AB + BC\overline{D} + \overline{A}D$ using only NOR gates.

Implement $Y = A\,B + B\,C\,\overline{D} + \overline{A}\,D$ using only NOR gates.

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

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

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

Implement $Y = A\,B + B\,C\,\overline{D} + \overline{A}\,D$ using only NOR gates.

$$Y = \overline{\overline{A\,B + B\,C\,\overline{D} + \overline{A}\,D}}$$

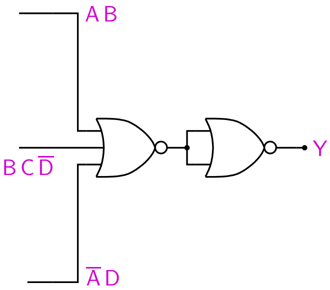
$$\overline{A} = \overline{A + A}$$

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

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

Implement $Y = A\,B + B\,C\,\overline{D} + \overline{A}\,D$ using only NOR gates.

$$Y = \overline{\overline{A\,B + B\,C\,\overline{D} + \overline{A}\,D}}$$



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

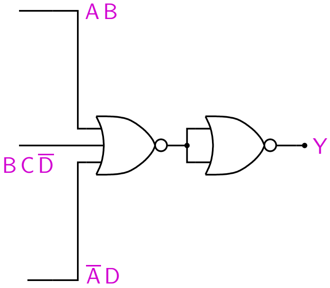
$$A + B = \overline{\overline{A + B}}$$

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

Implement $Y = A\,B + B\,C\,\overline{D} + \overline{A}\,D$ using only NOR gates.

$$Y = \overline{\overline{A\,B + B\,C\,\overline{D} + \overline{A}\,D}}$$

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



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

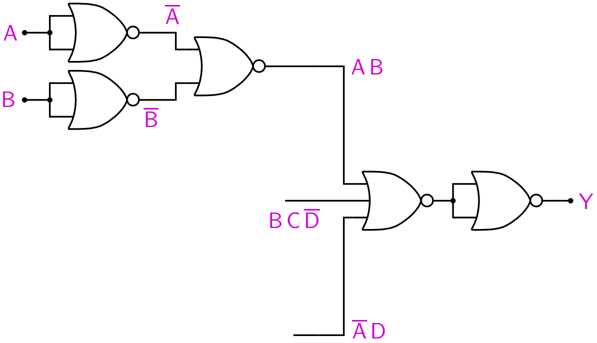
$$A + B = \overline{\overline{A + B}}$$

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

Implement $Y = A\,B + B\,C\,\overline{D} + \overline{A}\,D$ using only NOR gates.

$$Y = \overline{\overline{A\,B + B\,C\,\overline{D} + \overline{A}\,D}}$$

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



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

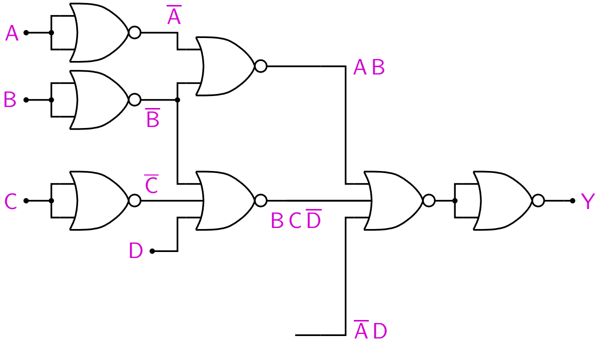
$$A + B = \overline{\overline{A + B}}$$

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

Implement $Y = AB + BC\overline{D} + \overline{A}D$ using only NOR gates.

$$Y = \overline{\overline{AB + BC\overline{D} + \overline{A}D}}$$

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



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

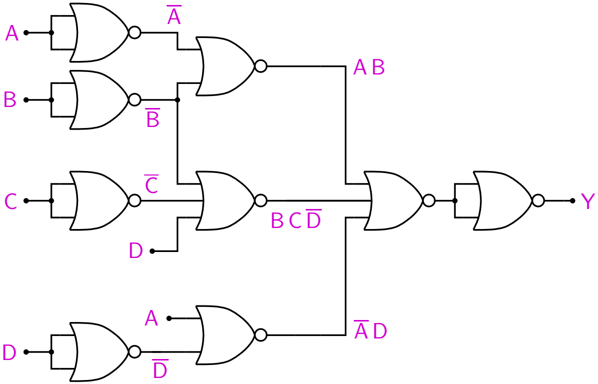
$A + B = \overline{\overline{A + B}}$

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

Implement $Y = A\,B + B\,C\,\overline{D} + \overline{A}\,D$ using only NOR gates.

$$Y = \overline{\overline{A\,B + B\,C\,\overline{D} + \overline{A}\,D}}$$

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

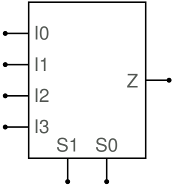


$$\overline{A} = \overline{A + A}$$

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

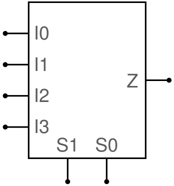
$$A \cdot B = \overline{\overline{A} + \overline{B}}$$

Multiplexers



S1	S0	Z
0	0	I0
0	1	I1
1	0	I2
1	1	I3

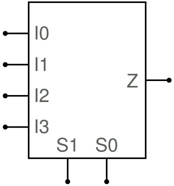
Multiplexers



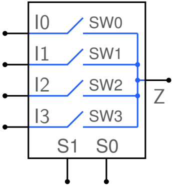
S1	S0	Z
0	0	I0
0	1	I1
1	0	I2
1	1	I3

* A multiplexer or data selector (MUX in short) has N Select lines, 2^N input lines, and it *routes* one of the input lines to the output.

Multiplexers

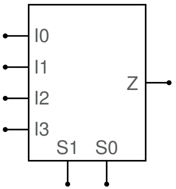


S1	S0	Z
0	0	I0
0	1	I1
1	0	I2
1	1	I3

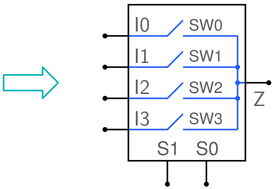


* A multiplexer or data selector (MUX in short) has N Select lines, 2^N input lines, and it *routes* one of the input lines to the output.

Multiplexers

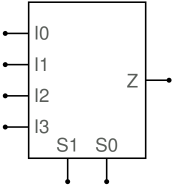


S1	S0	Z
0	0	I0
0	1	I1
1	0	I2
1	1	I3

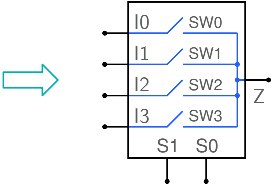


* A multiplexer or data selector (MUX in short) has N Select lines, 2^N input lines, and it *routes* one of the input lines to the output.
* Conceptually, a MUX may be thought of as 2^N switches. For a given combination of the select inputs, only one of the switches closes (makes contact), and the others are open.

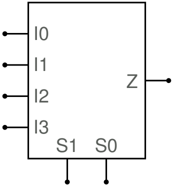
Multiplexers



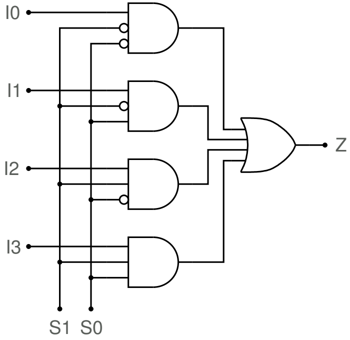
S1	S0	Z
0	0	I0
0	1	I1
1	0	I2
1	1	I3

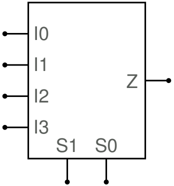


* A multiplexer or data selector (MUX in short) has N Select lines, 2^N input lines, and it *routes* one of the input lines to the output.
* Conceptually, a MUX may be thought of as 2^N switches. For a given combination of the select inputs, only one of the switches closes (makes contact), and the others are open.
* SEQUEL file: `mux_test_1.sqproj`

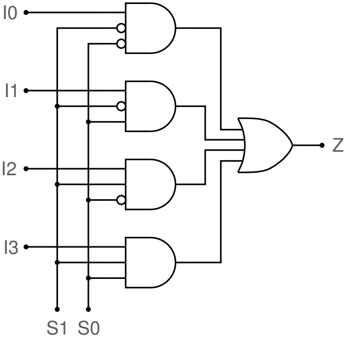


S1	S0	Z
0	0	I0
0	1	I1
1	0	I2
1	1	I3

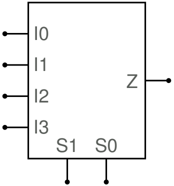




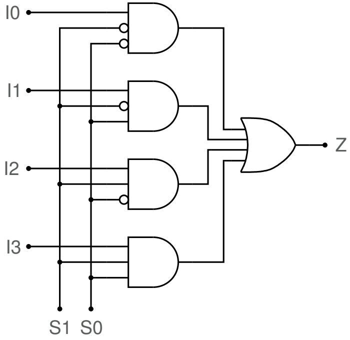
S1	S0	Z
0	0	I0
0	1	I1
1	0	I2
1	1	I3



* A 4-to-1 MUX can be implemented as,
  $Z = I_0\,\overline{S_1}\,\overline{S_0} + I_1\,\overline{S_1}\,S_0 + I_2\,S_1\,\overline{S_0} + I_3\,S_1\,S_0$.
  For a given combination of S_1 and S_0, only one of the terms survives (the others being 0). For example, with $S_1 = 0$, $S_0 = 1$, we have $Z = I_1$.



S1	S0	Z
0	0	I0
0	1	I1
1	0	I2
1	1	I3

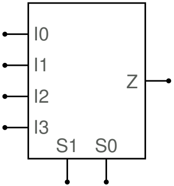


* A 4-to-1 MUX can be implemented as,

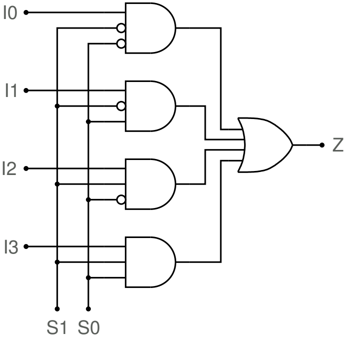
  $Z = I_0\, \overline{S_1}\, \overline{S_0} + I_1\, \overline{S_1}\, S_0 + I_2\, S_1\, \overline{S_0} + I_3\, S_1\, S_0\,.$

  For a given combination of S_1 and S_0, only one of the terms survives (the others being 0). For example, with $S_1 = 0$, $S_0 = 1$, we have $Z = I_1$.
* Multiplexers are available as ICs, e.g., 74151 is an 8-to-1 MUX.

Multiplexers



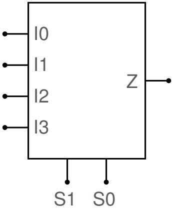
S1	S0	Z
0	0	I0
0	1	I1
1	0	I2
1	1	I3



* A 4-to-1 MUX can be implemented as,

  $Z = I_0\,\overline{S_1}\,\overline{S_0} + I_1\,\overline{S_1}\,S_0 + I_2\,S_1\,\overline{S_0} + I_3\,S_1\,S_0\,.$

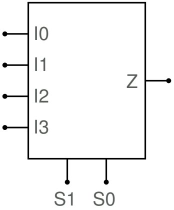
  For a given combination of S_1 and S_0, only one of the terms survives (the others being 0). For example, with $S_1 = 0$, $S_0 = 1$, we have $Z = I_1$.
* Multiplexers are available as ICs, e.g., 74151 is an 8-to-1 MUX.
* ICs with *arrays* of multiplexers (and other digital blocks) are also available. These blocks can be configured ("wired") by the user in a programmable manner to realise the functionality of interest.



S1	S0	Z
0	0	I0
0	1	I1
1	0	I2
1	1	I3

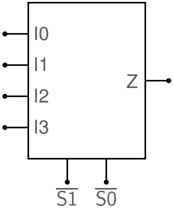
Select inputs are active high.

Active high and active low inputs/outputs



S1	S0	Z
0	0	I0
0	1	I1
1	0	I2
1	1	I3

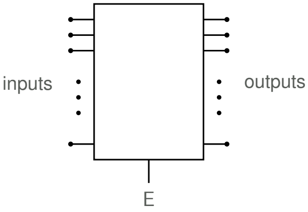
Select inputs are active high.



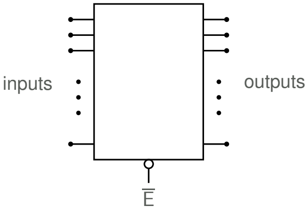
$\overline{S1}$	$\overline{S0}$	Z
1	1	I0
1	0	I1
0	1	I2
0	0	I3

Select inputs are active low.

Enable (E) pin

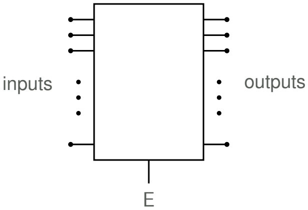


Active high enable pin

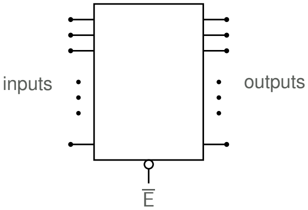


Active low enable pin

Enable (E) pin



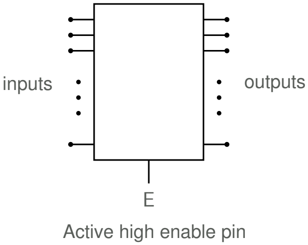
Active high enable pin

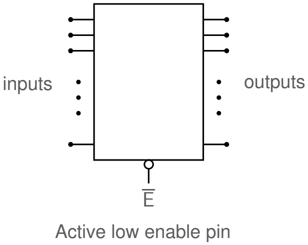


Active low enable pin

* Many digital ICs have an "Enable" (E) pin. If the Enable pin is active, the IC functions as desired; else, it is "disabled," i.e., the outputs are set to some default values.

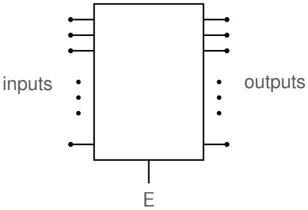
Enable (E) pin



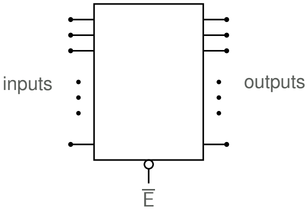


* Many digital ICs have an “Enable” (E) pin. If the Enable pin is active, the IC functions as desired; else, it is “disabled,” i.e., the outputs are set to some default values.
* The Enable pin can be active high or active low.

Enable (E) pin



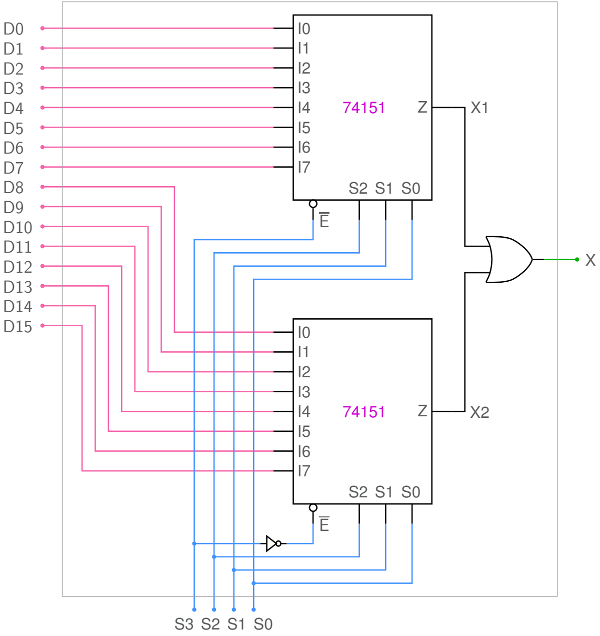
Active high enable pin



Active low enable pin

* Many digital ICs have an "Enable" (E) pin. If the Enable pin is active, the IC functions as desired; else, it is "disabled," i.e., the outputs are set to some default values.
* The Enable pin can be active high or active low.
* If the Enable pin is active low, it is denoted by $\overline{\text{Enable}}$ or $\overline{\text{E}}$. When $\overline{\text{E}}=0$, the IC functions normally; else, it is disabled.

Using two 8-to-1 MUXs to make a 16-to-1 MUX

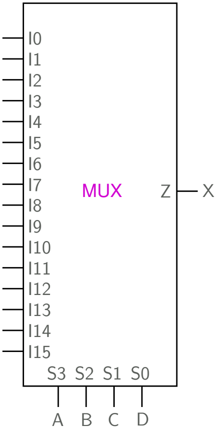


S3	S2	S1	S0	X
0	0	0	0	D0
0	0	0	1	D1
0	0	1	0	D2
0	0	1	1	D3
0	1	0	0	D4
0	1	0	1	D5
0	1	1	0	D6
0	1	1	1	D7
1	0	0	0	D8
1	0	0	1	D9
1	0	1	0	D10
1	0	1	1	D11
1	1	0	0	D12
1	1	0	1	D13
1	1	1	0	D14
1	1	1	1	D15

Implement $X = A\,\overline{B}\,\overline{C}\,D + \overline{A}\,B\,\overline{C}\,\overline{D}$ using a 16-to-1 MUX.

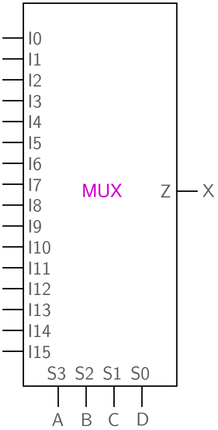
Implement $X = A\,\overline{B}\,\overline{C}\,D + \overline{A}\,B\,\overline{C}\,\overline{D}$ using a 16-to-1 MUX.

A	B	C	D	X
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0



Implement $X = A\,\overline{B}\,\overline{C}\,D + \overline{A}\,B\,\overline{C}\,\overline{D}$ using a 16-to-1 MUX.

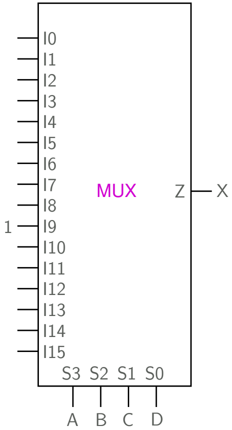
A	B	C	D	X
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0



* When $A\,\overline{B}\,\overline{C}\,D = 1$, we want $X = 1$.
  $A\,\overline{B}\,\overline{C}\,D = 1 \rightarrow A = 1,\ B = 0,\ C = 0,$
  $D = 1$, i.e., the input line corresponding to 1001 (I9) gets selected.
  $\rightarrow$ Make $\text{I9} = 1$.

Implement $X = A\,\overline{B}\,\overline{C}\,D + \overline{A}\,B\,\overline{C}\,\overline{D}$ using a 16-to-1 MUX.

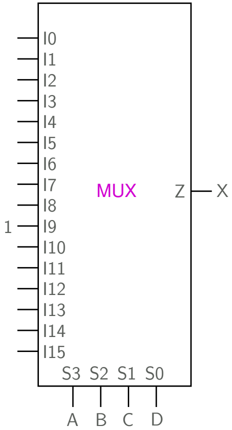
A	B	C	D	X
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0



* When $A\,\overline{B}\,\overline{C}\,D = 1$, we want $X = 1$.
  $A\,\overline{B}\,\overline{C}\,D = 1 \rightarrow A = 1,\ B = 0,\ C = 0,$
  $D = 1$, i.e., the input line corresponding to 1001 (I9) gets selected.
  $\rightarrow$ Make $\mathrm{I}9 = 1$.

Implement $X = A\,\overline{B}\,\overline{C}\,D + \overline{A}\,B\,\overline{C}\,\overline{D}$ using a 16-to-1 MUX.

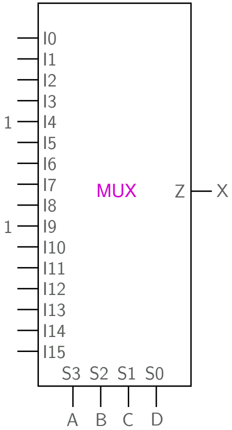
A	B	C	D	X
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0



* When $A\,\overline{B}\,\overline{C}\,D = 1$, we want $X = 1$.
  $A\,\overline{B}\,\overline{C}\,D = 1 \rightarrow A = 1,\ B = 0,\ C = 0,$
  $D = 1$, i.e., the input line corresponding to 1001 (I9) gets selected.
  $\rightarrow$ Make $\text{I9} = 1$.
* Similarly, when $\overline{A}\,B\,\overline{C}\,\overline{D} = 1$, we want $X = 1$.
  $\rightarrow$ Make $\text{I4} = 1$.

Implement $X = A\,\overline{B}\,\overline{C}\,D + \overline{A}\,B\,\overline{C}\,\overline{D}$ using a 16-to-1 MUX.

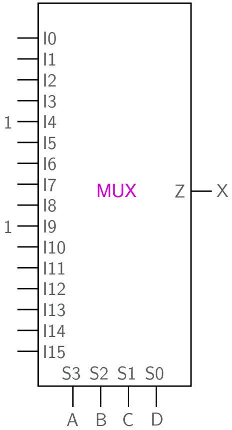
A	B	C	D	X
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0



* When $A\,\overline{B}\,\overline{C}\,D = 1$, we want $X = 1$.
  $A\,\overline{B}\,\overline{C}\,D = 1 \rightarrow A = 1,\ B = 0,\ C = 0,$ $D = 1$, i.e., the input line corresponding to 1001 (I9) gets selected.
  $\rightarrow$ Make $\text{I9} = 1$.
* Similarly, when $\overline{A}\,B\,\overline{C}\,\overline{D} = 1$, we want $X = 1$.
  $\rightarrow$ Make $\text{I4} = 1$.

Implement $X = A\,\overline{B}\,\overline{C}\,D + \overline{A}\,B\,\overline{C}\,\overline{D}$ using a 16-to-1 MUX.

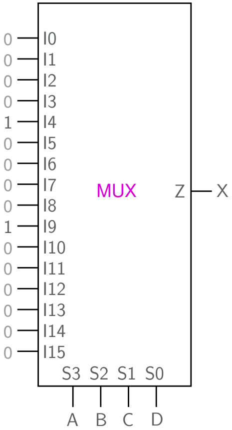
A	B	C	D	X
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0



* When $A\,\overline{B}\,\overline{C}\,D = 1$, we want $X = 1$.
  $A\,\overline{B}\,\overline{C}\,D = 1 \rightarrow A = 1,\ B = 0,\ C = 0,$ $D = 1$, i.e., the input line corresponding to 1001 (I9) gets selected.
  $\rightarrow$ Make I9 $= 1$.
* Similarly, when $\overline{A}\,B\,\overline{C}\,\overline{D} = 1$, we want $X = 1$.
  $\rightarrow$ Make I4 $= 1$.
* In all other cases, X should be 0.
  $\rightarrow$ connect all other pins to 0.

Implement $X = A\,\overline{B}\,\overline{C}\,D + \overline{A}\,B\,\overline{C}\,\overline{D}$ using a 16-to-1 MUX.

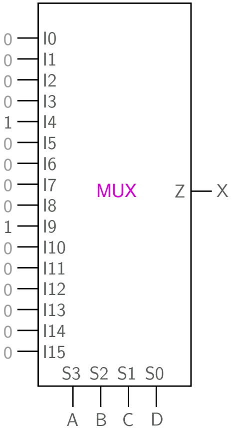
A	B	C	D	X
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0



* When $A\,\overline{B}\,\overline{C}\,D = 1$, we want $X = 1$.
  $A\,\overline{B}\,\overline{C}\,D = 1 \rightarrow A = 1,\ B = 0,\ C = 0,$ $D = 1$, i.e., the input line corresponding to 1001 (I9) gets selected.
  $\rightarrow$ Make I9 $= 1$.
* Similarly, when $\overline{A}\,B\,\overline{C}\,\overline{D} = 1$, we want $X = 1$.
  $\rightarrow$ Make I4 $= 1$.
* In all other cases, X should be 0.
  $\rightarrow$ connect all other pins to 0.

Implement $X = A\,\overline{B}\,\overline{C}\,D + \overline{A}\,B\,\overline{C}\,\overline{D}$ using a 16-to-1 MUX.

A	B	C	D	X
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

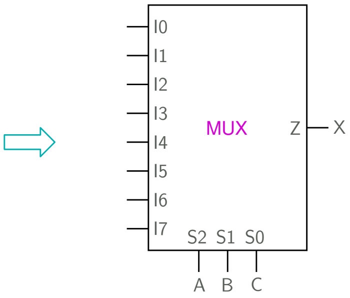


* When $A\,\overline{B}\,\overline{C}\,D = 1$, we want $X = 1$.
  $A\,\overline{B}\,\overline{C}\,D = 1 \rightarrow A = 1,\ B = 0,\ C = 0,$ $D = 1$, i.e., the input line corresponding to 1001 (I9) gets selected.
  $\rightarrow$ Make I9 $= 1$.
* Similarly, when $\overline{A}\,B\,\overline{C}\,\overline{D} = 1$, we want $X = 1$.
  $\rightarrow$ Make I4 $= 1$.
* In all other cases, X should be 0.
  $\rightarrow$ connect all other pins to 0.
* In this example, since the truth table is organized in terms of $ABCD$, with A as the MSB and D as the LSB (the same order in which A, B, C, D are connected to the select pins), the design is simple: connect
  I0 to X(0 0 0 0),
  I1 to X(0 0 0 1),
  I2 to X(0 0 1 0), etc.

Implement $X = A\,\overline{B}\,\overline{C}\,D + \overline{A}\,B\,\overline{C}\,\overline{D}$ using an 8-to-1 MUX.

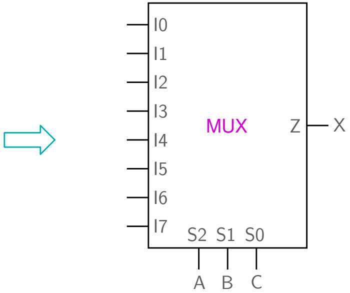
Implement $X = A\,\overline{B}\,\overline{C}\,D + \overline{A}\,B\,\overline{C}\,\overline{D}$ using an 8-to-1 MUX.

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	$\overline{\text{D}}$
0	1	1	0
1	0	0	D
1	0	1	0
1	1	0	0
1	1	1	0



Implement $X = A\,\overline{B}\,\overline{C}\,D + \overline{A}\,B\,\overline{C}\,\overline{D}$ using an 8-to-1 MUX.

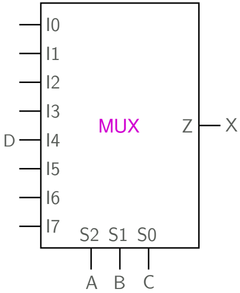
A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	$\overline{D}$
0	1	1	0
1	0	0	D
1	0	1	0
1	1	0	0
1	1	1	0



* When $A\,\overline{B}\,\overline{C} = 1$, i.e., $A = 1$, $B = 0$, $C = 0$, we have $X = D$.
  $\rightarrow$ connect the input line corresponding to 100 (I4) to D.

Implement $X = A\,\overline{B}\,\overline{C}\,D + \overline{A}\,B\,\overline{C}\,\overline{D}$ using an 8-to-1 MUX.

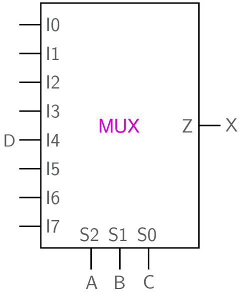
A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	$\overline{\text{D}}$
0	1	1	0
1	0	0	D
1	0	1	0
1	1	0	0
1	1	1	0



* When $A\,\overline{B}\,\overline{C} = 1$, i.e., $A = 1$, $B = 0$, $C = 0$, we have $X = D$.
  $\rightarrow$ connect the input line corresponding to 100 (I4) to D.

Implement $X = A\,\overline{B}\,\overline{C}\,D + \overline{A}\,B\,\overline{C}\,\overline{D}$ using an 8-to-1 MUX.

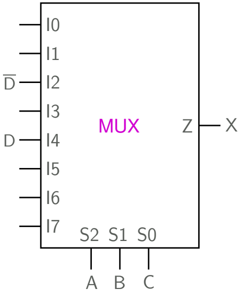
A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	$\overline{D}$
0	1	1	0
1	0	0	D
1	0	1	0
1	1	0	0
1	1	1	0



* When $A\,\overline{B}\,\overline{C} = 1$, i.e., $A = 1$, $B = 0$, $C = 0$, we have $X = D$.
  $\rightarrow$ connect the input line corresponding to 100 (I4) to D.
* When $\overline{A}\,B\,\overline{C} = 1$, i.e., $A = 0$, $B = 1$, $C = 0$, we have $X = \overline{D}$.
  $\rightarrow$ connect the input line corresponding to 010 (I2) to $\overline{D}$.

Implement $X = A\,\overline{B}\,\overline{C}\,D + \overline{A}\,B\,\overline{C}\,\overline{D}$ using an 8-to-1 MUX.

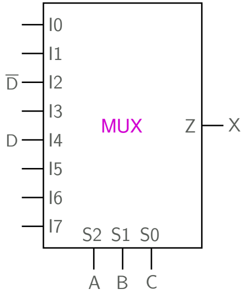
A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	$\overline{\text{D}}$
0	1	1	0
1	0	0	D
1	0	1	0
1	1	0	0
1	1	1	0



* When $A\,\overline{B}\,\overline{C} = 1$, i.e., $A = 1$, $B = 0$, $C = 0$, we have $X = D$.
  $\rightarrow$ connect the input line corresponding to 100 (I4) to D.
* When $\overline{A}\,B\,\overline{C} = 1$, i.e., $A = 0$, $B = 1$, $C = 0$, we have $X = \overline{D}$.
  $\rightarrow$ connect the input line corresponding to 010 (I2) to $\overline{D}$.

Implement $X = A\,\overline{B}\,\overline{C}\,D + \overline{A}\,B\,\overline{C}\,\overline{D}$ using an 8-to-1 MUX.

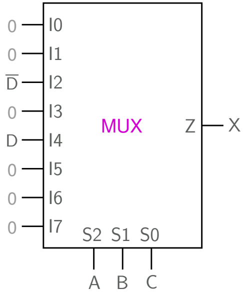
A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	$\overline{D}$
0	1	1	0
1	0	0	D
1	0	1	0
1	1	0	0
1	1	1	0



* When $A\,\overline{B}\,\overline{C} = 1$, i.e., $A = 1$, $B = 0$, $C = 0$, we have $X = D$.
  $\rightarrow$ connect the input line corresponding to 100 (I4) to D.
* When $\overline{A}\,B\,\overline{C} = 1$, i.e., $A = 0$, $B = 1$, $C = 0$, we have $X = \overline{D}$.
  $\rightarrow$ connect the input line corresponding to 010 (I2) to $\overline{D}$.
* In all other cases, X should be 0.
  $\rightarrow$ connect all other pins to 0.

Implement $X = A\,\overline{B}\,\overline{C}\,D + \overline{A}\,B\,\overline{C}\,\overline{D}$ using an 8-to-1 MUX.

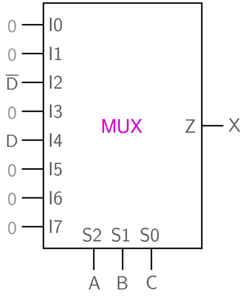
A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	$\overline{D}$
0	1	1	0
1	0	0	D
1	0	1	0
1	1	0	0
1	1	1	0



* When $A\,\overline{B}\,\overline{C} = 1$, i.e., $A = 1$, $B = 0$, $C = 0$, we have $X = D$.
  $\rightarrow$ connect the input line corresponding to 100 (I4) to D.
* When $\overline{A}\,B\,\overline{C} = 1$, i.e., $A = 0$, $B = 1$, $C = 0$, we have $X = \overline{D}$.
  $\rightarrow$ connect the input line corresponding to 010 (I2) to $\overline{D}$.
* In all other cases, X should be 0.
  $\rightarrow$ connect all other pins to 0.

Implement $X = A\,\overline{B}\,\overline{C}\,D + \overline{A}\,B\,\overline{C}\,\overline{D}$ using an 8-to-1 MUX.

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	$\overline{D}$
0	1	1	0
1	0	0	D
1	0	1	0
1	1	0	0
1	1	1	0



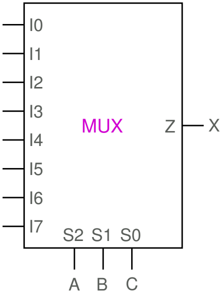
* When $A\,\overline{B}\,\overline{C} = 1$, i.e., $A = 1$, $B = 0$, $C = 0$, we have $X = D$.
  $\rightarrow$ connect the input line corresponding to 100 (I4) to D.
* When $\overline{A}\,B\,\overline{C} = 1$, i.e., $A = 0$, $B = 1$, $C = 0$, we have $X = \overline{D}$.
  $\rightarrow$ connect the input line corresponding to 010 (I2) to $\overline{D}$.
* In all other cases, X should be 0.
  $\rightarrow$ connect all other pins to 0.
* Home work: Implement the same function (X) with $S2 = B$, $S1 = C$, $S0 = D$.

A	B	C	D	X
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Implement the function X using an 8-to-1 MUX.

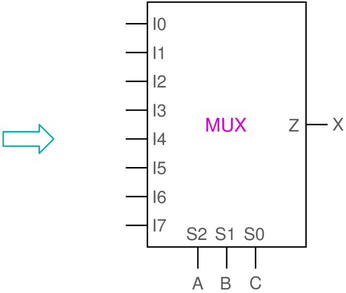
A	B	C	D	X
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Implement the function X using an 8-to-1 MUX.



A	B	C	D	X
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

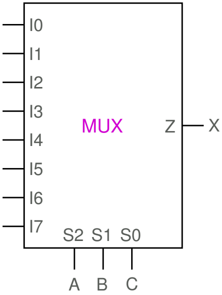
Implement the function X using an 8-to-1 MUX.



A	B	C	D	X
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Implement the function X using an 8-to-1 MUX.

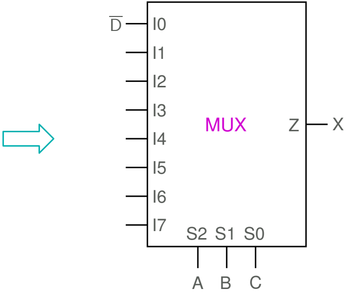
* When $ABC = 000$, $X = \overline{D} \rightarrow \text{I0} = \overline{D}$.



A	B	C	D	X
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Implement the function X using an 8-to-1 MUX.

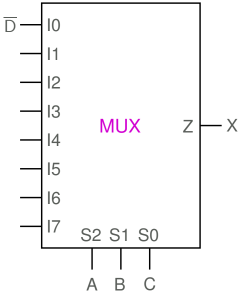
* When $ABC = 000$, $X = \overline{D} \rightarrow \text{I0} = \overline{D}$.



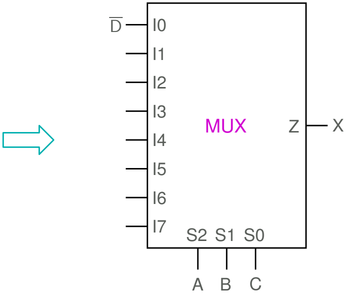
A	B	C	D	X
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Implement the function X using an 8-to-1 MUX.

* When $ABC = 000$, $X = \overline{D} \rightarrow \text{I0} = \overline{D}$.



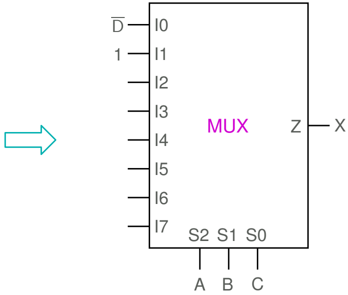
A	B	C	D	X
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0



Implement the function X using an 8-to-1 MUX.

* When $ABC = 000$, $X = \overline{D} \rightarrow$ I0 $= \overline{D}$.
* When $ABC = 001$, $X = 1 \rightarrow$ I1 $= 1$, and so on.

A	B	C	D	X
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0



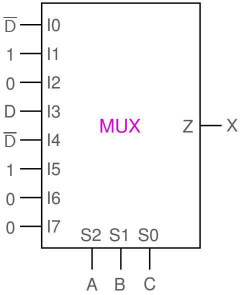
Implement the function X using an 8-to-1 MUX.

* When $ABC = 000$, $X = \overline{D} \rightarrow \text{I0} = \overline{D}$.
* When $ABC = 001$, $X = 1 \rightarrow \text{I1} = 1$, and so on.

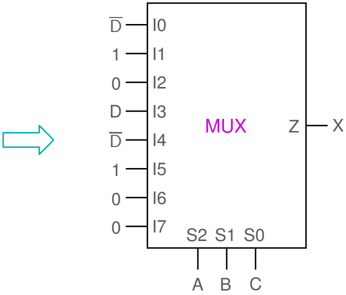
A	B	C	D	X
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Implement the function X using an 8-to-1 MUX.

* When $ABC = 000$, $X = \overline{D} \rightarrow \mathrm{I0} = \overline{D}$.
* When $ABC = 001$, $X = 1 \rightarrow \mathrm{I1} = 1$, and so on.



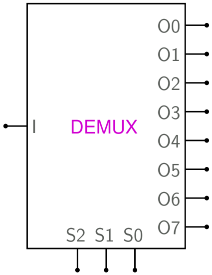
A	B	C	D	X
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0



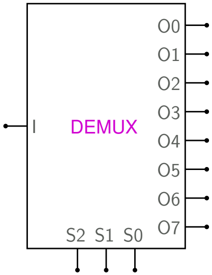
Implement the function X using an 8-to-1 MUX.

* When $ABC = 000$, $X = \overline{D} \rightarrow$ I0 $= \overline{D}$.
* When $ABC = 001$, $X = 1 \rightarrow$ I1 $= 1$, and so on.
* Home work: repeat with S2 $= B$, S1 $= C$, S0 $= D$.

S2	S1	S0	O0	O1	O2	O3	O4	O5	O6	O7
0	0	0	I	0	0	0	0	0	0	0
0	0	1	0	I	0	0	0	0	0	0
0	1	0	0	0	I	0	0	0	0	0
0	1	1	0	0	0	I	0	0	0	0
1	0	0	0	0	0	0	I	0	0	0
1	0	1	0	0	0	0	0	I	0	0
1	1	0	0	0	0	0	0	0	I	0
1	1	1	0	0	0	0	0	0	0	I



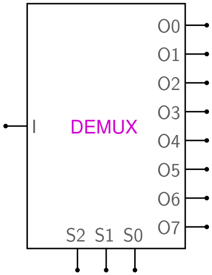
S2	S1	S0	O0	O1	O2	O3	O4	O5	O6	O7
0	0	0	I	0	0	0	0	0	0	0
0	0	1	0	I	0	0	0	0	0	0
0	1	0	0	0	I	0	0	0	0	0
0	1	1	0	0	0	I	0	0	0	0
1	0	0	0	0	0	0	I	0	0	0
1	0	1	0	0	0	0	0	I	0	0
1	1	0	0	0	0	0	0	0	I	0
1	1	1	0	0	0	0	0	0	0	I



* A demultiplexer takes a *single* input (I) and *routes* it to one of the output lines (O0, O1, $\cdots$).

Demultiplexers

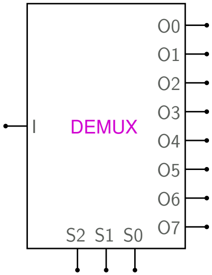
S2	S1	S0	O0	O1	O2	O3	O4	O5	O6	O7
0	0	0	I	0	0	0	0	0	0	0
0	0	1	0	I	0	0	0	0	0	0
0	1	0	0	0	I	0	0	0	0	0
0	1	1	0	0	0	I	0	0	0	0
1	0	0	0	0	0	0	I	0	0	0
1	0	1	0	0	0	0	0	I	0	0
1	1	0	0	0	0	0	0	0	I	0
1	1	1	0	0	0	0	0	0	0	I



* A demultiplexer takes a *single* input (I) and *routes* it to one of the output lines (O0, O1, $\cdots$).
* For N Select inputs (S0, S1, $\cdots$), the number of output lines is 2^N.

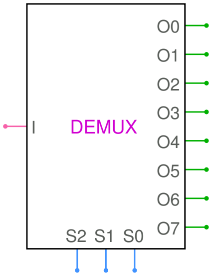
Demultiplexers

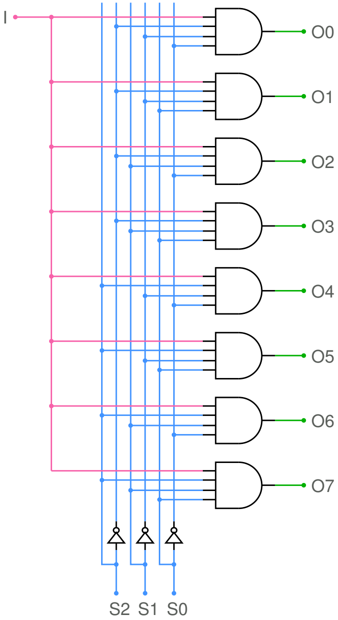
S2	S1	S0	O0	O1	O2	O3	O4	O5	O6	O7
0	0	0	I	0	0	0	0	0	0	0
0	0	1	0	I	0	0	0	0	0	0
0	1	0	0	0	I	0	0	0	0	0
0	1	1	0	0	0	I	0	0	0	0
1	0	0	0	0	0	0	I	0	0	0
1	0	1	0	0	0	0	0	I	0	0
1	1	0	0	0	0	0	0	0	I	0
1	1	1	0	0	0	0	0	0	0	I



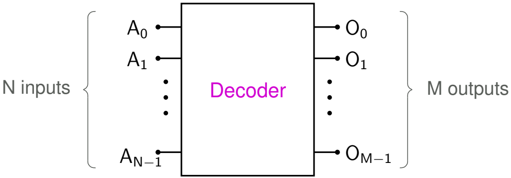
* A demultiplexer takes a *single* input (I) and *routes* it to one of the output lines (O0, O1, $\cdots$).
* For N Select inputs (S0, S1, $\cdots$), the number of output lines is 2^N.
* SEQUEL file: `demux_test_1.sqproj`

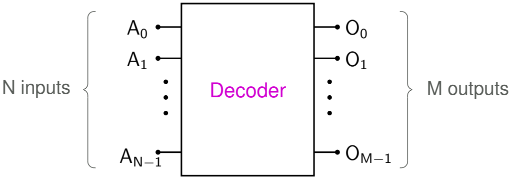
Demultiplexer: gate-level diagram





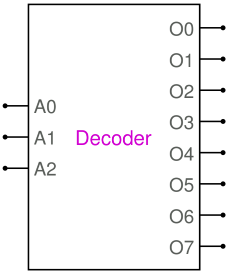
Decoders





* For each input combination, an associated bit pattern appears at the output.

3-to-8 decoder (1-of-8 decoder)



A2	A1	A0	O0	O1	O2	O3	O4	O5	O6	O7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

* Example:

  Decimal 75

Binary-Coded-Decimal (BCD) encoding

* Example:

Decimal	75
Binary	1001011

Binary-Coded-Decimal (BCD) encoding

* Example:

Decimal	75
Binary	1001011
BCD	0111 0101

Binary-Coded-Decimal (BCD) encoding

* Example:

Decimal	75
Binary	1001011
BCD	0111 0101

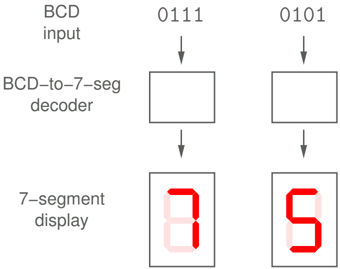
* BCD coding is commonly used to display numbers in electronic systems.

Binary-Coded-Decimal (BCD) encoding

* Example:

Decimal	75
Binary	1001011
BCD	0111 0101

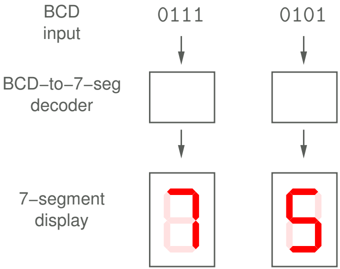
* BCD coding is commonly used to display numbers in electronic systems.



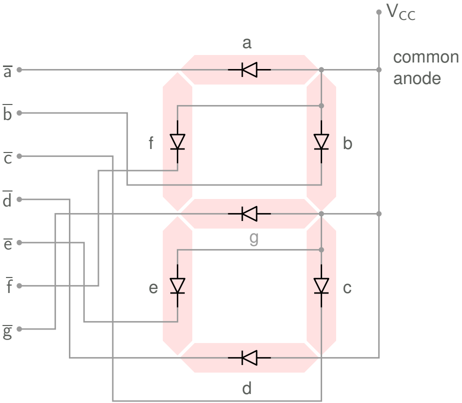
- Example:

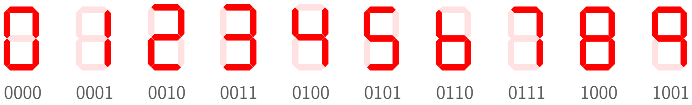
Decimal	75
Binary	1001011
BCD	0111 0101

- BCD coding is commonly used to display numbers in electronic systems.

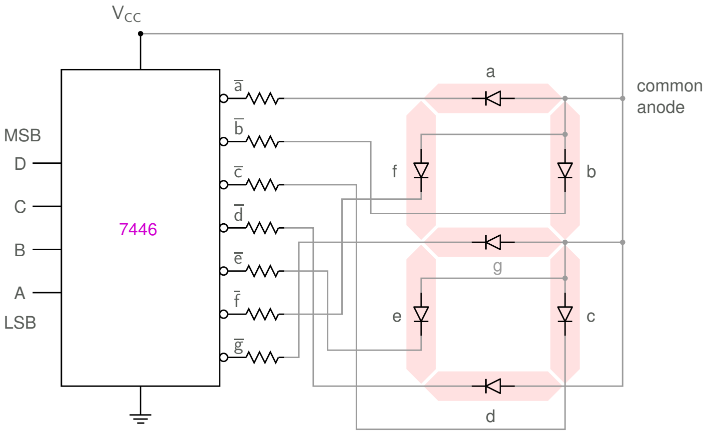


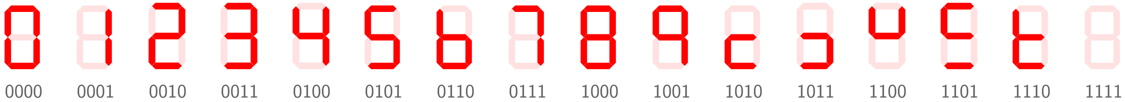
- In some electronic systems (e.g., calculators), all computations are performed in BCD.


V_{CC}
a
$\overline{a}$
common
anode
$\overline{b}$
f
b
$\overline{c}$
$\overline{d}$
g
$\overline{e}$
e
c
$\overline{f}$
$\overline{g}$
d

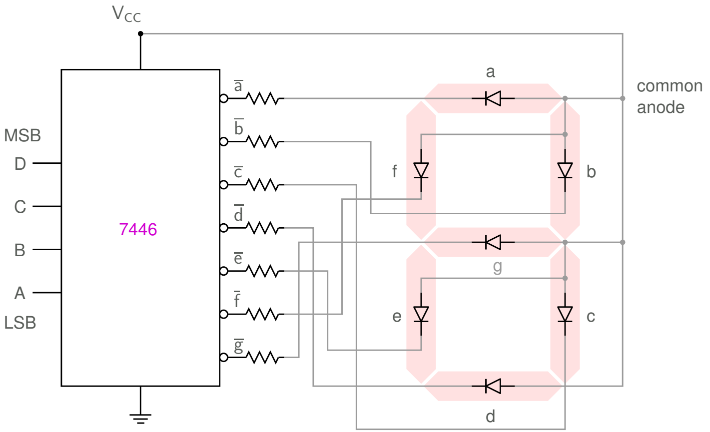

0000
0001
0010
0011
0100
0101
0110
0111
1000
1001

BCD-to-7 segment decoder

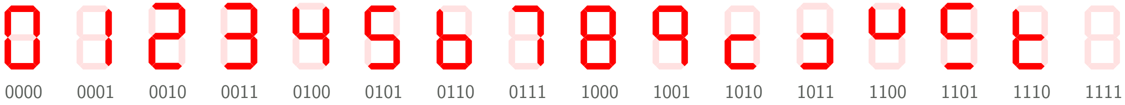




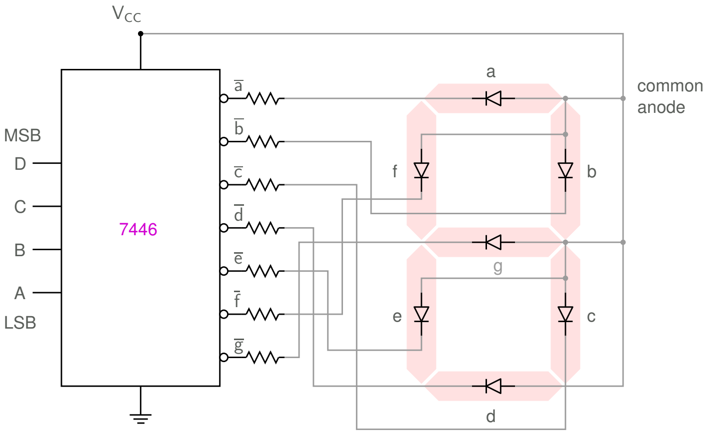
BCD-to-7 segment decoder



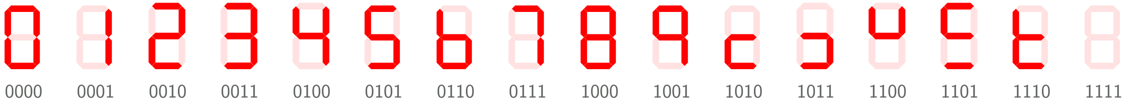
* The resistors serve to limit the diode current. For $V_{CC} = 5\ V$, $V_D = 2\ V$, and $I_D = 10\ \mathrm{mA}$, $R = 300\ \Omega$.



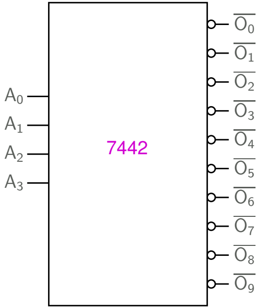
BCD-to-7 segment decoder



* The resistors serve to limit the diode current. For $V_{CC} = 5\ V$, $V_D = 2\ V$, and $I_D = 10\,\mathrm{mA}$, $R = 300\,\Omega$.
* Home work: Write the truth table for $\overline{c}$ (in terms of D, C, B, A). Obtain a minimized expression for $\overline{c}$ using a K map.

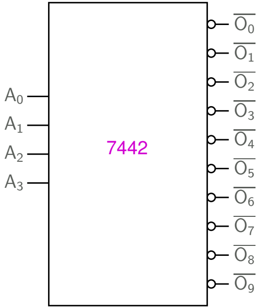


M. B. Patil, IIT Bombay



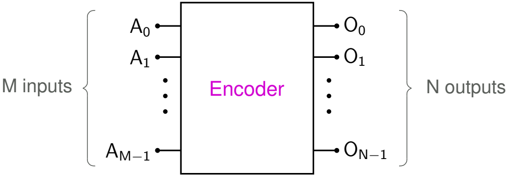
A_3	A_2	A_1	A_0	Active output
0	0	0	0	$\overline{O_0}$
0	0	0	1	$\overline{O_1}$
0	0	1	0	$\overline{O_2}$
0	0	1	1	$\overline{O_3}$
0	1	0	0	$\overline{O_4}$
0	1	0	1	$\overline{O_5}$
0	1	1	0	$\overline{O_6}$
0	1	1	1	$\overline{O_7}$
1	0	0	0	$\overline{O_8}$
1	0	0	1	$\overline{O_9}$
1	0	1	0	none
1	0	1	1	none
1	1	0	0	none
1	1	0	1	none
1	1	1	0	none
1	1	1	1	none

BCD-to-decimal decoder

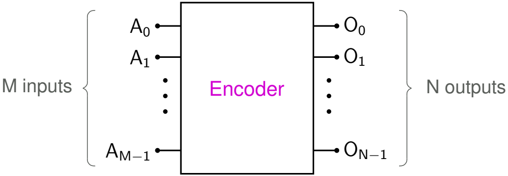


A_3	A_2	A_1	A_0	Active output
0	0	0	0	$\overline{O_0}$
0	0	0	1	$\overline{O_1}$
0	0	1	0	$\overline{O_2}$
0	0	1	1	$\overline{O_3}$
0	1	0	0	$\overline{O_4}$
0	1	0	1	$\overline{O_5}$
0	1	1	0	$\overline{O_6}$
0	1	1	1	$\overline{O_7}$
1	0	0	0	$\overline{O_8}$
1	0	0	1	$\overline{O_9}$
1	0	1	0	none
1	0	1	1	none
1	1	0	0	none
1	1	0	1	none
1	1	1	0	none
1	1	1	1	none

Encoders

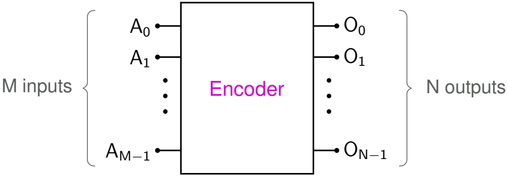


Encoders



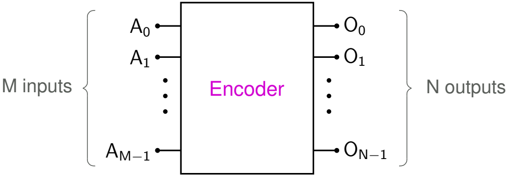
* Only one input line is assumed to be active. The binary number corresponding to the active input line appears at the output pins.

Encoders



* Only one input line is assumed to be active. The binary number corresponding to the active input line appears at the output pins.
* The N output lines can represent 2^N binary numbers, each corresponding to one of the M input lines, i.e., we can have $M = 2^N$. Some encoders have $M < 2^N$.

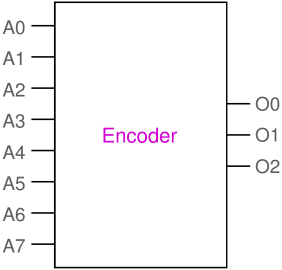
Encoders



* Only one input line is assumed to be active. The binary number corresponding to the active input line appears at the output pins.
* The N output lines can represent 2^N binary numbers, each corresponding to one of the M input lines, i.e., we can have $M = 2^N$. Some encoders have $M < 2^N$.
* As an example, for $N = 3$, we can have a maximum of $2^3 = 8$ input lines.

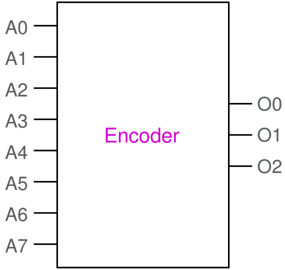
Encoders

8–to–3 encoder example



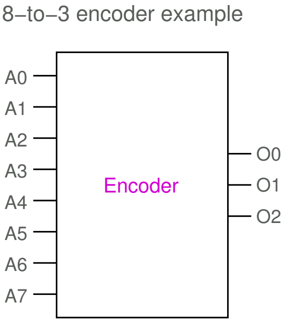
A0	A1	A2	A3	A4	A5	A6	A7	O2	O1	O0
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

8–to–3 encoder example



A0	A1	A2	A3	A4	A5	A6	A7	O2	O1	O0
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

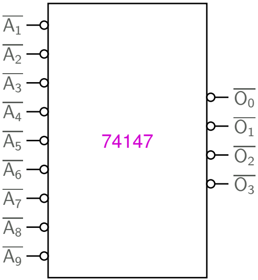
* Note that only one of the input lines is assumed to be active.



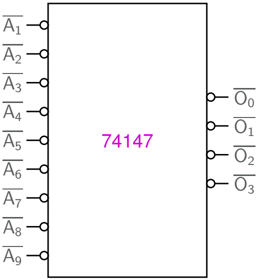
A0	A1	A2	A3	A4	A5	A6	A7	O2	O1	O0
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

* Note that only one of the input lines is assumed to be active.
* What if two input lines become simultaneously active?
  → There are "priority encoders" which assign a *priority* to each of the input lines.

74147 decimal-to-BCD priority encoder



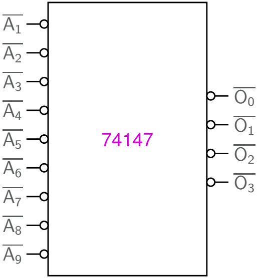
$\overline{A_1}$	$\overline{A_2}$	$\overline{A_3}$	$\overline{A_4}$	$\overline{A_5}$	$\overline{A_6}$	$\overline{A_7}$	$\overline{A_8}$	$\overline{A_9}$	$\overline{O_3}$	$\overline{O_2}$	$\overline{O_1}$	$\overline{O_0}$
1	1	1	1	1	1	1	1	1	1	1	1	1
X	X	X	X	X	X	X	X	0	0	1	1	0
X	X	X	X	X	X	X	0	1	0	1	1	1
X	X	X	X	X	X	0	1	1	1	0	0	0
X	X	X	X	X	0	1	1	1	1	0	0	1
X	X	X	X	0	1	1	1	1	1	0	1	0
X	X	X	0	1	1	1	1	1	1	0	1	1
X	X	0	1	1	1	1	1	1	1	1	0	0
X	0	1	1	1	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1	0



$\overline{A_1}$	$\overline{A_2}$	$\overline{A_3}$	$\overline{A_4}$	$\overline{A_5}$	$\overline{A_6}$	$\overline{A_7}$	$\overline{A_8}$	$\overline{A_9}$	$\overline{O_3}$	$\overline{O_2}$	$\overline{O_1}$	$\overline{O_0}$
1	1	1	1	1	1	1	1	1	1	1	1	1
X	X	X	X	X	X	X	X	0	0	1	1	0
X	X	X	X	X	X	X	0	1	0	1	1	1
X	X	X	X	X	X	0	1	1	1	0	0	0
X	X	X	X	X	0	1	1	1	1	0	0	1
X	X	X	X	0	1	1	1	1	1	0	1	0
X	X	X	0	1	1	1	1	1	1	0	1	1
X	X	0	1	1	1	1	1	1	1	1	0	0
X	0	1	1	1	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1	0

* Note that the higher input lines get priority over the lower ones.
  For example, $\overline{A_7}$ gets priority over $\overline{A_1}$, $\overline{A_2}$, $\overline{A_3}$, $\overline{A_4}$, $\overline{A_5}$, $\overline{A_6}$. If $\overline{A_7}$ is active (low), the binary output is 1000 (i.e., 0111 inverted bit-by-bit) which corresponds to decimal 7, *irrespective of* $\overline{A_1}$, $\overline{A_2}$, $\overline{A_3}$, $\overline{A_4}$, $\overline{A_5}$, $\overline{A_6}$.

74147 decimal-to-BCD priority encoder



$\overline{A_1}$	$\overline{A_2}$	$\overline{A_3}$	$\overline{A_4}$	$\overline{A_5}$	$\overline{A_6}$	$\overline{A_7}$	$\overline{A_8}$	$\overline{A_9}$	$\overline{O_3}$	$\overline{O_2}$	$\overline{O_1}$	$\overline{O_0}$
1	1	1	1	1	1	1	1	1	1	1	1	1
X	X	X	X	X	X	X	X	0	0	1	1	0
X	X	X	X	X	X	X	0	1	0	1	1	1
X	X	X	X	X	X	0	1	1	1	0	0	0
X	X	X	X	X	0	1	1	1	1	0	0	1
X	X	X	X	0	1	1	1	1	1	0	1	0
X	X	X	0	1	1	1	1	1	1	0	1	1
X	X	0	1	1	1	1	1	1	1	1	0	0
X	0	1	1	1	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1	0

* Note that the higher input lines get priority over the lower ones.
  For example, $\overline{A_7}$ gets priority over $\overline{A_1}$, $\overline{A_2}$, $\overline{A_3}$, $\overline{A_4}$, $\overline{A_5}$, $\overline{A_6}$. If $\overline{A_7}$ is active (low), the binary output is 1000 (i.e., 0111 inverted bit-by-bit) which corresponds to decimal 7, *irrespective of* $\overline{A_1}$, $\overline{A_2}$, $\overline{A_3}$, $\overline{A_4}$, $\overline{A_5}$, $\overline{A_6}$.
* The lower input lines are therefore shown as "don't care" (X) conditions.