Lecture 11:
Digital Logic

Announcements

- Midterm Grades Released
  - If you have any questions, email me or any of the Tas
  - Overall Class Average: ~81 %
- Project 3 released
  - Due August 11th 11:55pm, 2 Weeks
  - Due day before Final, so plan accordingly
  - No Extension
- Lectures until final:
  - Digital Logic (Chapter 4 in book)
- Rest of lecture time / recitation today:
  - Overview of Programming Assignment 3

Logic Design

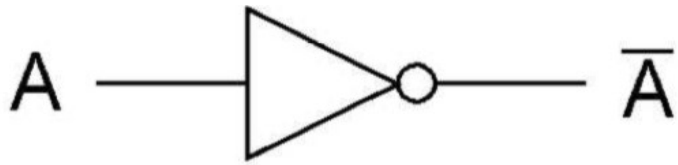
- How does your processor perform various operations?

Logic Gates

- Transition from representing information to implementing them

- Logic gates are simple digital circuits
  - Take one or more binary inputs
  - Produce a binary output
  - Truth table: relationship between the input and the output

Not Gate



(or A')

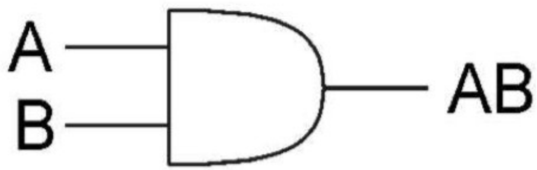
Truth table

In	Out
0	1
1	0

Simplest Gate

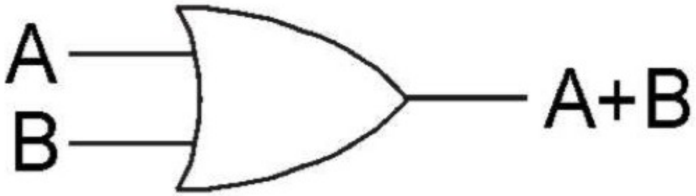
And Gate

A	B	C
0	0	0
0	1	0
1	0	0
1	1	1



AND

Or Gate



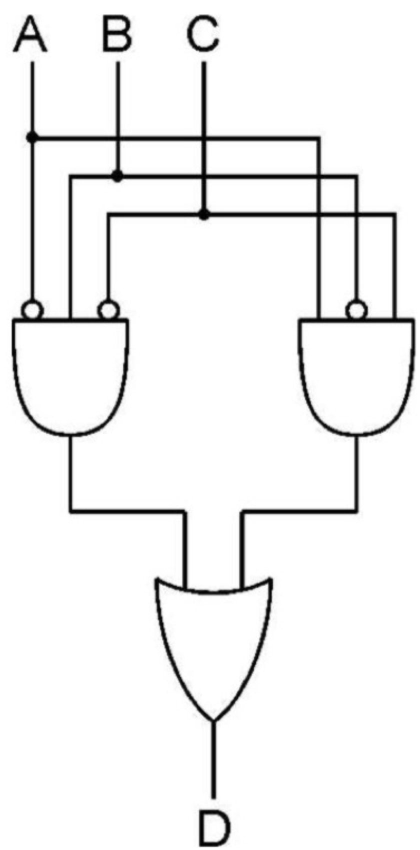
OR

A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

Logical Completeness

- Can implement any truth table with Not, Or,And gates.

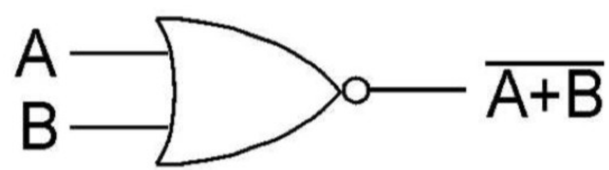
A	B	C	D
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0



1. AND combinations that yield a "1" in the truth table.
2. OR the results of the AND gates.

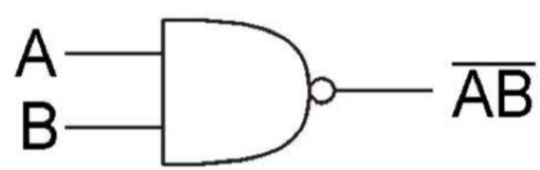
NAND and NOR gates

A	B	C
0	0	1
0	1	0
1	0	0
1	1	0



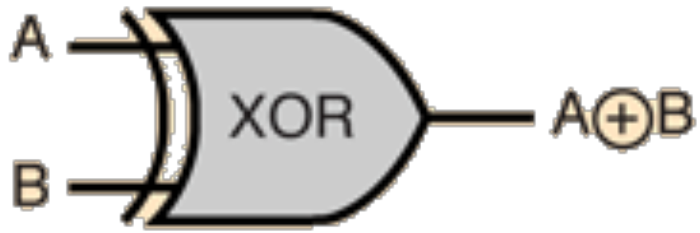
NOR

A	B	C
0	0	1
0	1	1
1	0	1
1	1	0



NAND

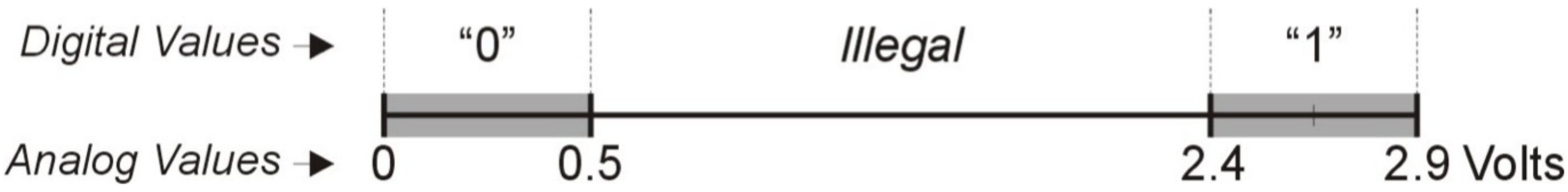
Xor Gate



A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

Beneath the Digital Abstraction

- Digital system uses discrete values
  - Represent it with continuous variables (voltage, etc)
  - Also must handle noise
- Transistors used to implement logical functions
- Voltage used to represent 0 or 1

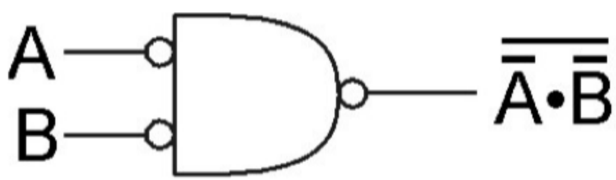


Transistor

- Microprocessors contain millions (billions) of transistors
  - Intel Pentium 4 (2000): 48 million
  - IBM/Apple PowerPC G5 (2003): 58 million
- A transistor acts as a switch
- Combined to implement logic functions (AND, OR, NOT..)
- Combined to build higher-level structures (Adder, Decoder..)
- Combined to build processor

DeMorgan's Law

- Converting AND to OR (and some NOT).

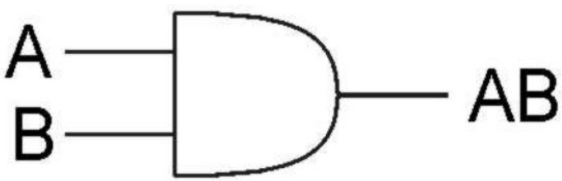


A	B	$\overline{A}$	$\overline{B}$	$\overline{A} \cdot \overline{B}$	$\overline{\overline{A} \cdot \overline{B}}$
0	0	1	1	1	0
0	1	1	0	0	1
1	0	0	1	0	1
1	1	0	0	0	1

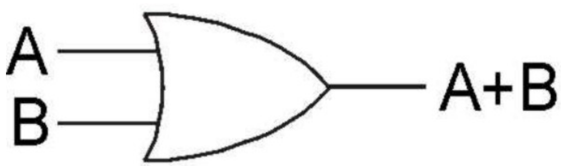
- In general,
  - (1) $\overline{PQ} = \overline{P} + \overline{Q}$
  - (2) $\overline{P+Q} = \overline{P}\,\overline{Q}$

Recap

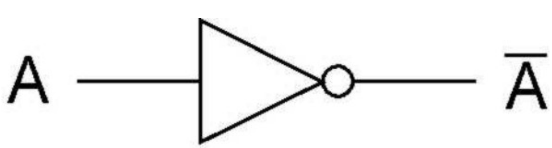
- 6 Widely used logic gates
- And Gate



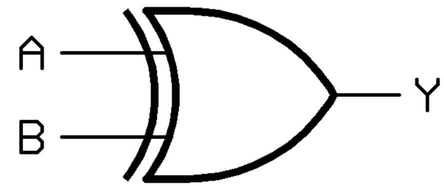
- Or Gate



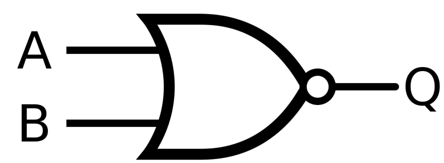
- Not Gate



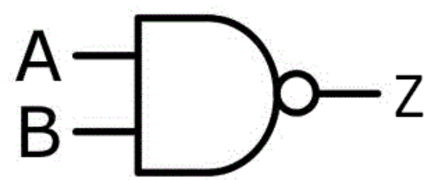
- Xor Gate



- Nor Gate

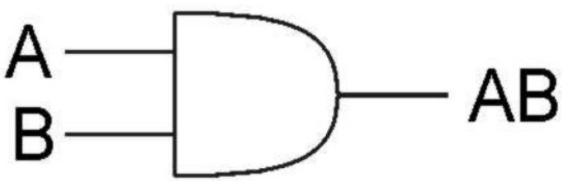


- Nand Gate

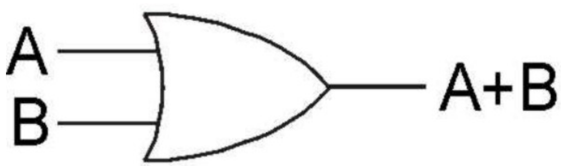


Transformation

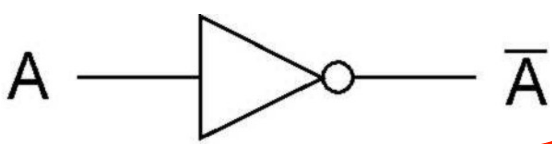
- 6 Widely used logic gates
- And Gate



- Or Gate



- Not Gate



- Xor Gate

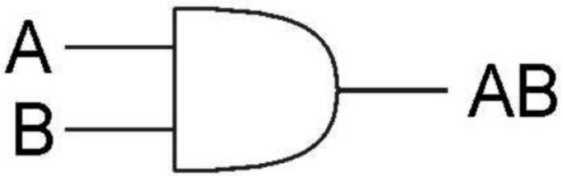
Can express any
logic circuit with these as we
have seen previously.

- Nor Gate

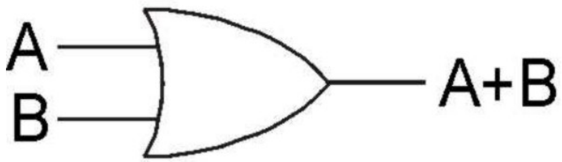
- Nand Gate

Transformation

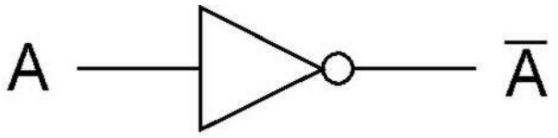
- 6 Widely used logic gates
- And Gate



- Or Gate



- Not Gate



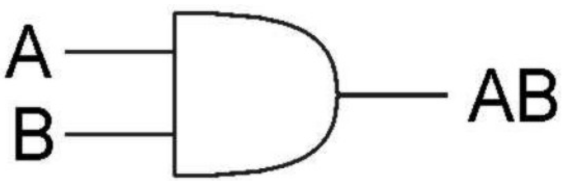
- Xor Gate

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

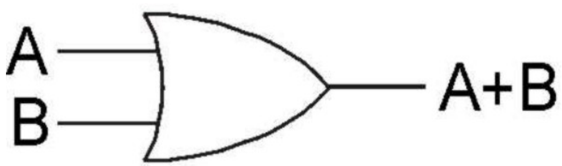
- Nor Gate
- Nand Gate

Transformation

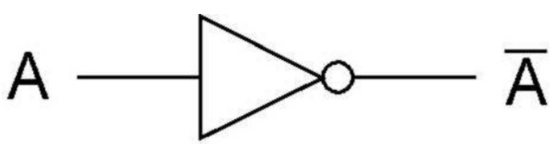
- 6 Widely used logic gates
- And Gate



- Or Gate



- Not Gate



- Xor Gate

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

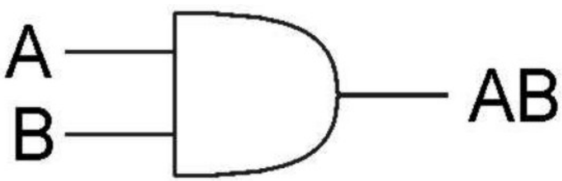
- Nor Gate

$$\overline{A + B}$$

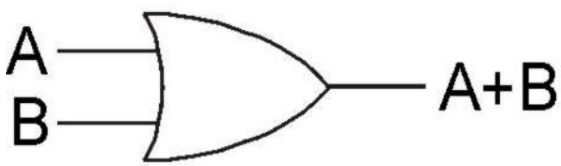
- Nand Gate

Transformation

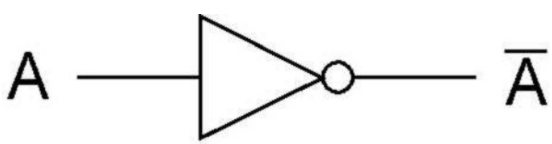
- 6 Widely used logic gates
- And Gate



- Or Gate



- Not Gate



- Xor Gate

$A\bar{B} + B\bar{A}$

- Nor Gate

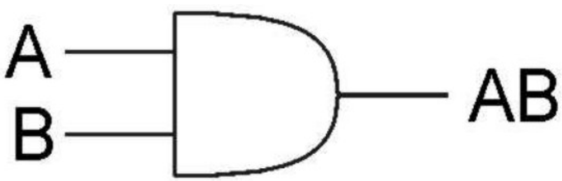
$\overline{A+B}$

- Nand Gate

$\overline{AB}$

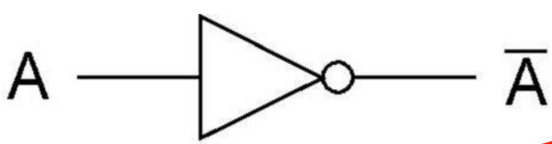
Transformation

- 6 Widely used logic gates
- And Gate



- Or Gate

- Not Gate



- Xor Gate

In fact…
You just need
these two

- Nor Gate

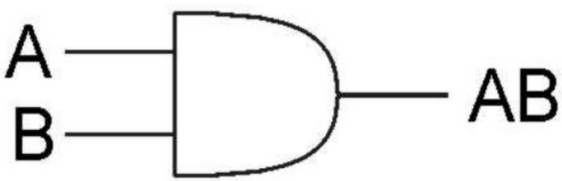
$\overline{A + B}$

- Nand Gate

$\overline{AB}$

Transformation

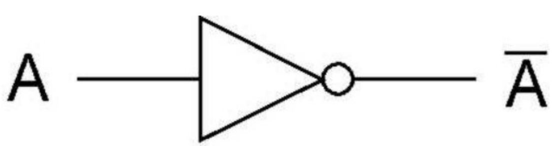
- 6 Widely used logic gates
- And Gate



- Or Gate

$\overline{A}\,\overline{B}$

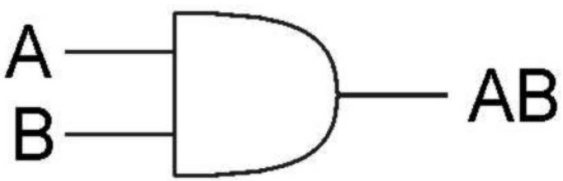
- Not Gate



- Xor Gate
- Nor Gate
- Nand Gate

Transformation

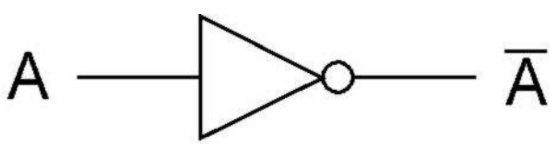
- 6 Widely used logic gates
- And Gate



- Or Gate

$$\overline{\overline{A}\,\overline{B}}$$

- Not Gate



- Xor Gate

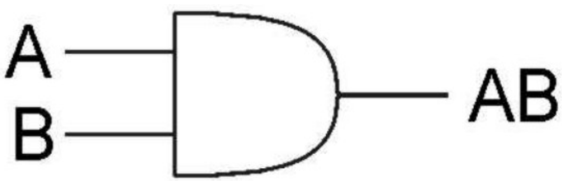
$$\overline{\overline{A\overline{B}}\,\overline{\overline{A}B}}$$

- Nor Gate

- Nand Gate

Transformation

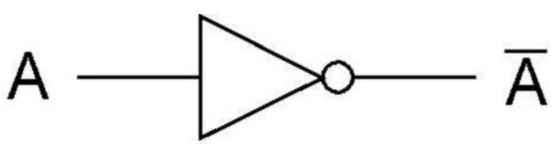
- 6 Widely used logic gates
- And Gate



- Or Gate

$$\overline{\overline{A}\,\overline{B}}$$

- Not Gate



- Xor Gate

$$\overline{\overline{A\overline{B}}\;\overline{\overline{A}B}}$$

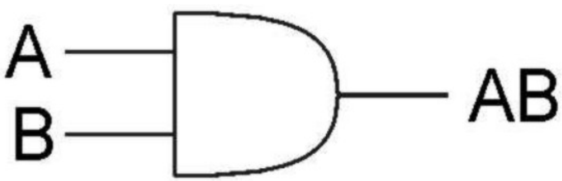
- Nor Gate

$$\overline{A}\,\overline{B}$$

- Nand Gate

Transformation

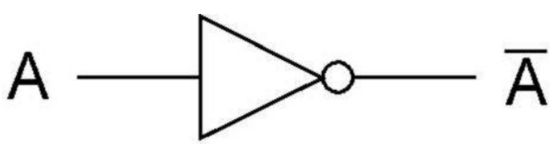
- 6 Widely used logic gates
- And Gate



- Or Gate

$\overline{\overline{A}\,\overline{B}}$

- Not Gate



- Xor Gate

$\overline{\overline{A\overline{B}}\;\overline{\overline{A}B}}$

- Nor Gate

$\overline{A}\,\overline{B}$

- Nand Gate

$\overline{AB}$

Transformation

- 6 Widely used logic gates
- And Gate

- Or Gate

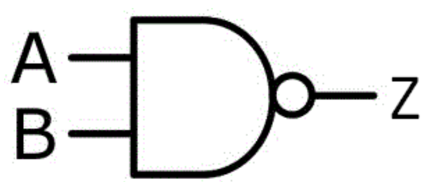
- Not Gate

- Xor Gate

Side note:
Or You just need this Nand Gate to do anything

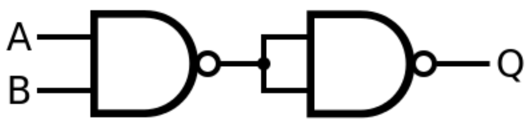
- Nor Gate

- Nand Gate

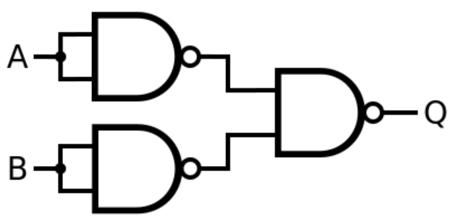


Transformation

- 6 Widely used logic gates
- And Gate

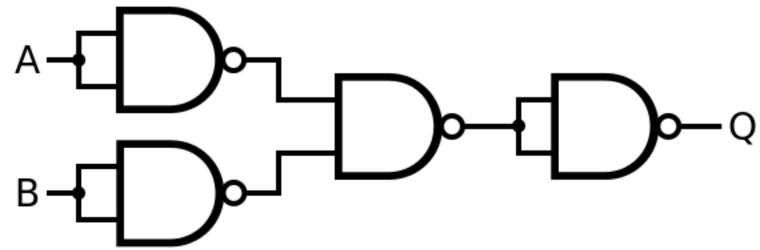


- Or Gate

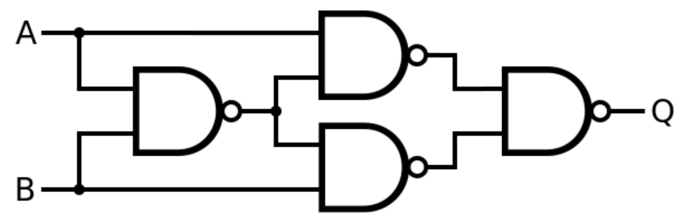


- Not Gate

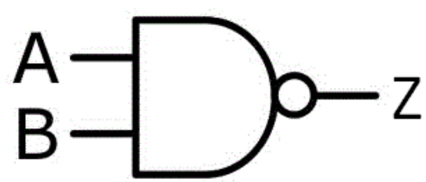
- Xor Gate



- Nor Gate



- Nand Gate



Transformation

- 6 Widely used logic gates

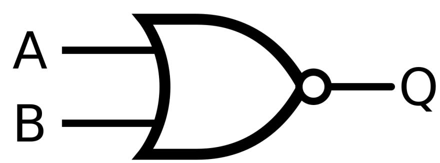
Side note:
Or You just need this
Nor Gate to do anything

- Or Gate

- Not Gate

- Xor Gate

- Nor Gate



- Nand Gate

Transformation

- 6 Widely used logic gates
- And Gate

- Or Gate

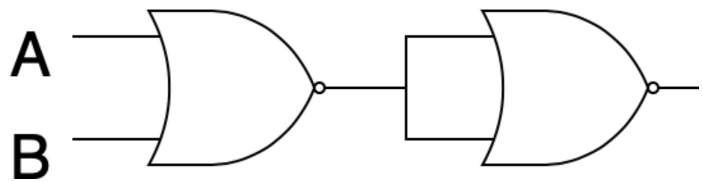
- Not Gate

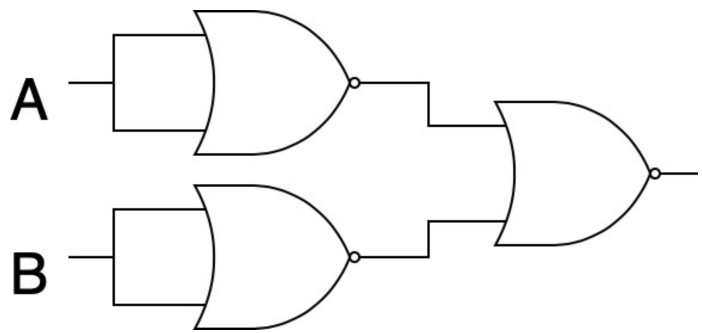
- Xor Gate

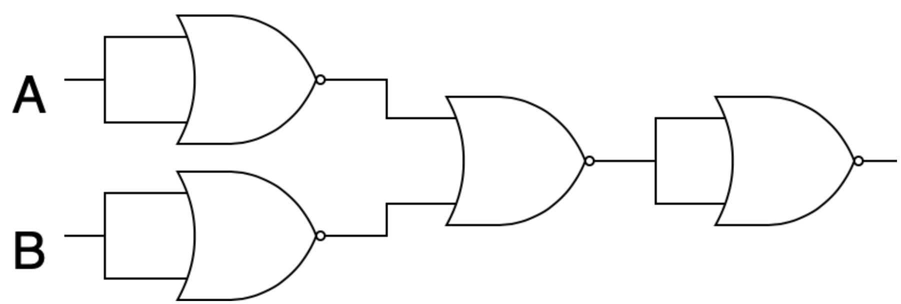
And many more: Imagine all the fun questions you'll be seeing in the final exam

- Nand Gate

Question

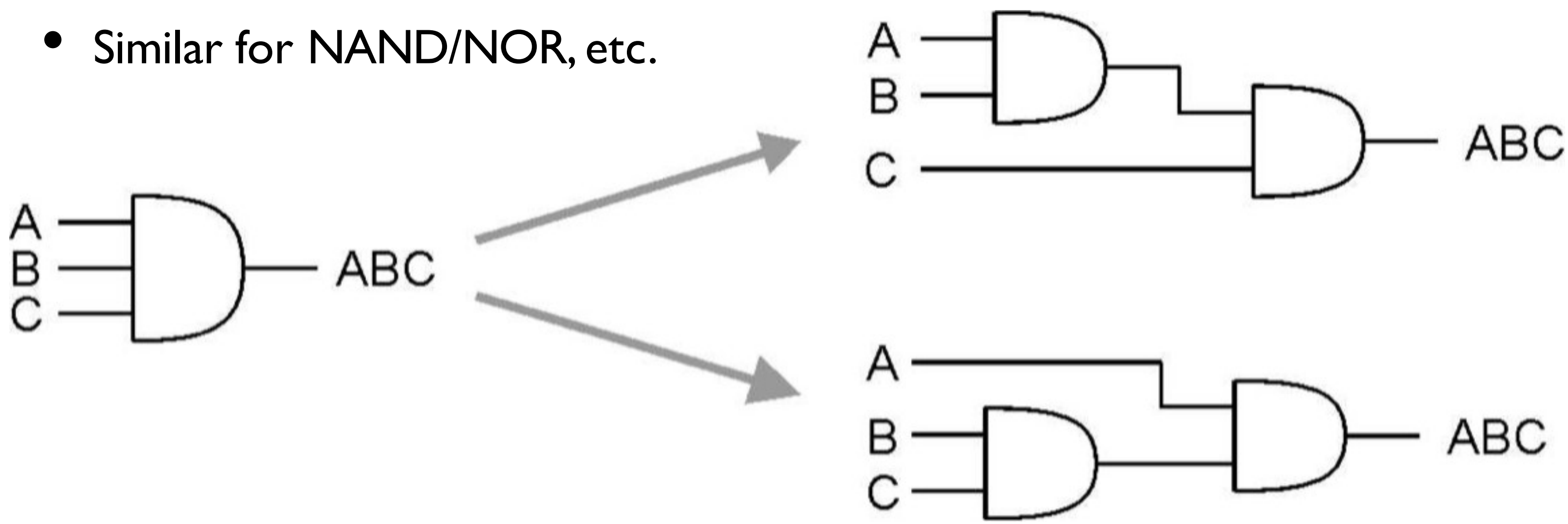
- Which of these circuits using NOR gates is equivalent to an AND gate?
- A: 

- B: 

- C: 

More than 2 Inputs?

- AND/OR can take any number of inputs:
  - AND = 1 if all inputs are 1
  - OR = 1 if any input is 1.
  - Similar for NAND/NOR, etc.



So … Circuit Design?

- You have a circuit you want to build
- Derive a truth table for this circuit
- Derive Boolean expression for the truth table
- Then build the circuit based on the boolean expression
- The tricky part is how do you optimize this circuit?

Truth Table to Boolean Expression

sensor inputs

A	B	C	Output
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

- Given a circuit, isolate the rows in which the output of the circuit is 1

Truth Table to Boolean Expression

sensor inputs

A	B	C	Output
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

sensor inputs

A	B	C	Output	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$\overline{A}BC = 1$
1	0	0	0	
1	0	1	1	$A\overline{B}C = 1$
1	1	0	1	$AB\overline{C} = 1$
1	1	1	1	$ABC = 1$

- A product term that contains exactly one instance of every variable is called a minterm

Minterm

A	B	C	minterm
0	0	0	m0 $\bar{A}\bar{B}\bar{C}$
0	0	1	m1 $\bar{A}\bar{B}C$
0	1	0	m2 $\bar{A}B\bar{C}$
0	1	1	m3 $\bar{A}BC$
1	0	0	m4 $A\bar{B}\bar{C}$
1	0	1	m5 $A\bar{B}C$
1	1	0	m6 $AB\bar{C}$
1	1	1	m7 ABC

- A product term in which all variables appear once.
- Each minterm evaluates to 1 in exactly one case. All other case, it evalutes to 0.

Truth Table to Boolean Expression

sensor inputs

A	B	C	Output
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

sensor inputs

A	B	C	Output	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$\overline{A}BC = 1$
1	0	0	0	
1	0	1	1	$A\overline{B}C = 1$
1	1	0	1	$AB\overline{C} = 1$
1	1	1	1	$ABC = 1$

- Given expressions for each row, build a larger Boolean expression. This is called a sum-of-products (SOP) form.

$$Output = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

Truth Table to Boolean Expression

sensor inputs

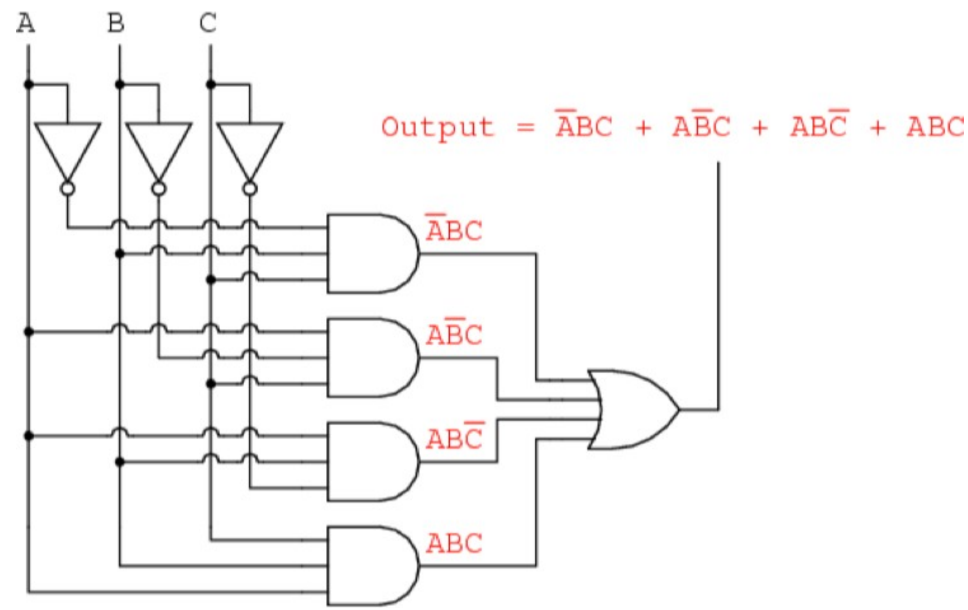
A	B	C	Output
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$\overline{A}BC = 1$

$A\overline{B}C = 1$

$AB\overline{C} = 1$

$ABC = 1$



- Finally build the circuit
- However, SoP forms are usually not minimal. We must optimize.

Canonical Forms

- There are two canonical forms:
  - Sum of Products (SOP)

$$F = \overline{Y}\,\overline{Z} + X\overline{Y}Z + XY\overline{Z}$$

  - Product of Sums (POS)

$$F = (Y + Z)(\overline{X} + Y + Z)(\overline{X} + \overline{Y} + Z)$$

Converting PoS to SoP

- Multiply through:

$$F = (Y + Z)(\overline{X} + Y + Z)(\overline{X} + \overline{Y} + Z)$$

Converting PoS to SoP

- Multiply through:

$$F = (Y + Z)(\overline{X} + Y + \overline{Z})(\overline{X} + \overline{Y} + Z)$$

$$F = (\overline{X}Y + Y + Y\overline{Z} + \overline{X}Z + YZ + Z\overline{Z})(\overline{X} + \overline{Y} + Z)$$

Converting PoS to SoP

- Multiply through:

$$F = (Y + Z)(\overline{X} + Y + \overline{Z})(\overline{X} + \overline{Y} + Z)$$

$$F = (\overline{X}Y + Y + Y\overline{Z} + \overline{X}Z + YZ + Z\overline{Z})(\overline{X} + \overline{Y} + Z)$$

$$F = (Y + \overline{X}Z + YZ)(\overline{X} + \overline{Y} + Z)$$

Converting PoS to SoP

- Multiply through:

$$F = (Y + Z)(\overline{X} + Y + \overline{Z})(\overline{X} + \overline{Y} + Z)$$

$$F = (\overline{X}Y + Y + Y\overline{Z} + \overline{X}Z + YZ + Z\overline{Z})(\overline{X} + \overline{Y} + Z)$$

$$F = (Y + \overline{X}Z + YZ)(\overline{X} + \overline{Y} + Z)$$

$$F = \overline{X}Y + Y\overline{Y} + YZ + \overline{X}Z + \overline{X}\overline{Y}Z + \overline{X}Z + \overline{X}YZ + Y\overline{Y}Z + YZ$$

Converting PoS to SoP

- Multiply through:

$$F = (Y + Z)(\overline{X} + Y + \overline{Z})(\overline{X} + \overline{Y} + Z)$$

$$F = (\overline{X}Y + Y + Y\overline{Z} + \overline{X}Z + YZ + Z\overline{Z})(\overline{X} + \overline{Y} + Z)$$

$$F = (Y + \overline{X}Z + YZ)(\overline{X} + \overline{Y} + Z)$$

$$F = \overline{X}Y + Y\overline{Y} + YZ + \overline{X}Z + \overline{X}\,\overline{Y}Z + \overline{X}Z + \overline{X}YZ + Y\overline{Y}Z + YZ$$

$$F = \overline{X}Y + YZ + \overline{X}Z$$

Converting SoP to PoS

- Complement, multiply through, then complement

$$F = \overline{Y}\,\overline{Z} + X\overline{Y}Z + XY\overline{Z}$$

Converting SoP to PoS

- Complement, multiply through, then complement

$$F = \overline{Y}\overline{Z} + X\overline{Y}Z + XY\overline{Z}$$

$$\overline{F} = (Y + Z)(\overline{X} + Y + \overline{Z})(\overline{X} + \overline{Y} + Z)$$

Converting SoP to PoS

- Complement, multiply through, then complement

$$F = \overline{Y}\overline{Z} + X\overline{Y}Z + XY\overline{Z}$$

$$\overline{F} = (Y + Z)(\overline{X} + Y + \overline{Z})(\overline{X} + \overline{Y} + Z)$$

$$\overline{F} = \overline{X}Y + YZ + \overline{X}Z$$

Converting SoP to PoS

- Complement, multiply through, then complement

$$F = \overline{Y}\overline{Z} + X\overline{Y}Z + XY\overline{Z}$$

$$\overline{F} = (Y + Z)(\overline{X} + Y + \overline{Z})(\overline{X} + \overline{Y} + Z)$$

$$\overline{F} = \overline{X}Y + YZ + \overline{X}Z$$

$$F = (X + \overline{Y})(\overline{Y} + \overline{Z})(X + \overline{Z})$$

Algebraic Optimization

- Simply use the rules of Boolean logic

$$\overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

Algebraic Optimization

- Simply use the rules of Boolean logic

$$\bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$$

$$BC(\bar{A} + A) + A\bar{B}C + AB\bar{C}$$

Algebraic Optimization

- Simply use the rules of Boolean logic

$$\overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

$$BC(\overline{A} + A) + A\overline{B}C + AB\overline{C}$$

$$BC + A\overline{B}C + AB\overline{C}$$

Algebraic Optimization

- Simply use the rules of Boolean logic

$$\overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

$$BC(\overline{A} + A) + A\overline{B}C + AB\overline{C}$$

$$BC + A\overline{B}C + AB\overline{C}$$

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

Algebraic Optimization

- Simply use the rules of Boolean logic

$$\overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

$$BC(\overline{A} + A) + A\overline{B}C + AB\overline{C}$$

$$BC + A\overline{B}C + AB\overline{C}$$

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

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

Algebraic Optimization

- Simply use the rules of Boolean logic

$$\overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

$$BC(\overline{A} + A) + A\overline{B}C + AB\overline{C}$$

$$BC + A\overline{B}C + AB\overline{C}$$

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

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

$$AB + BC + A\overline{B}C$$

Algebraic Optimization

- Simply use the rules of Boolean logic

$$AB + BC + A\overline{B}C$$

Algebraic Optimization

- Simply use the rules of Boolean logic

$$AB + BC + A\bar{B}C$$

$$BC + A(B + \bar{B}C)$$

Algebraic Optimization

- Simply use the rules of Boolean logic

$$AB + BC + A\bar{B}C$$

$$BC + A(B + \bar{B}C)$$

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

Algebraic Optimization

- Simply use the rules of Boolean logic

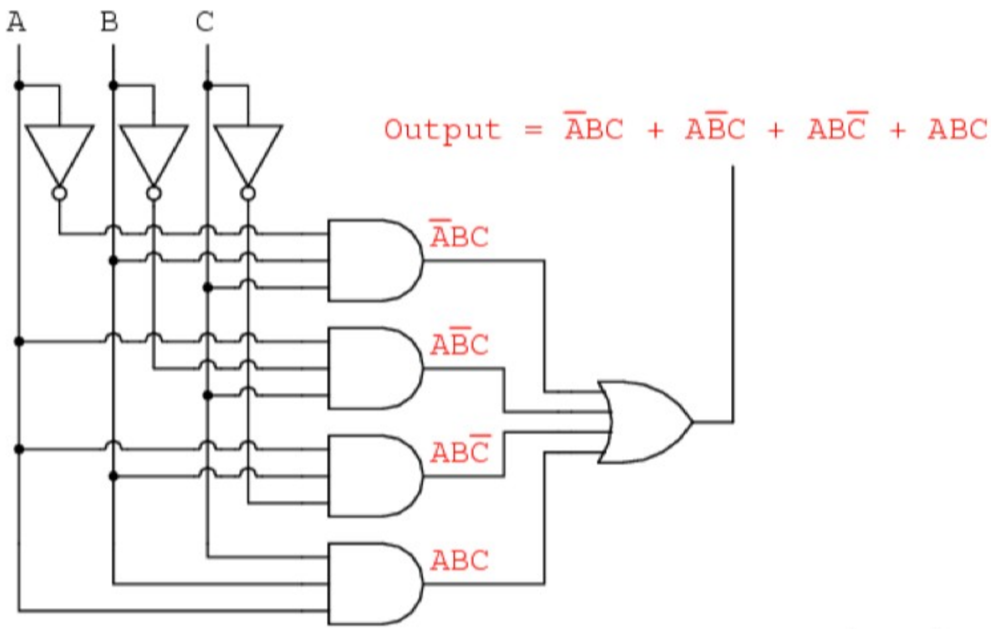
$$AB + BC + A\bar{B}C$$

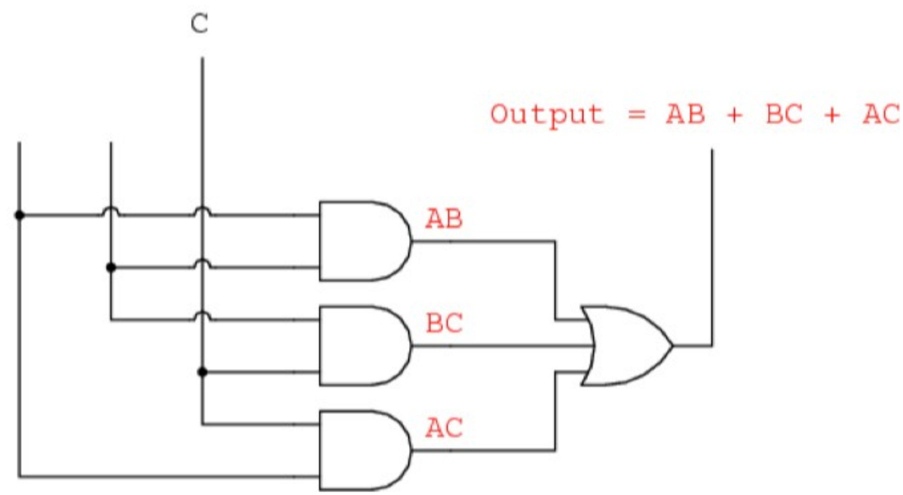
$$BC + A(B + \bar{B}C)$$

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

$$AB + BC + AC$$

Algebraic Optimization





Question

- Simplify the following expression:

$$\overline{A}\overline{B}C + \overline{A}BC + A\overline{B}C + ABC$$

- A: A
- B: B
- C: C
- D: AC + BC
- E: A + C

Question

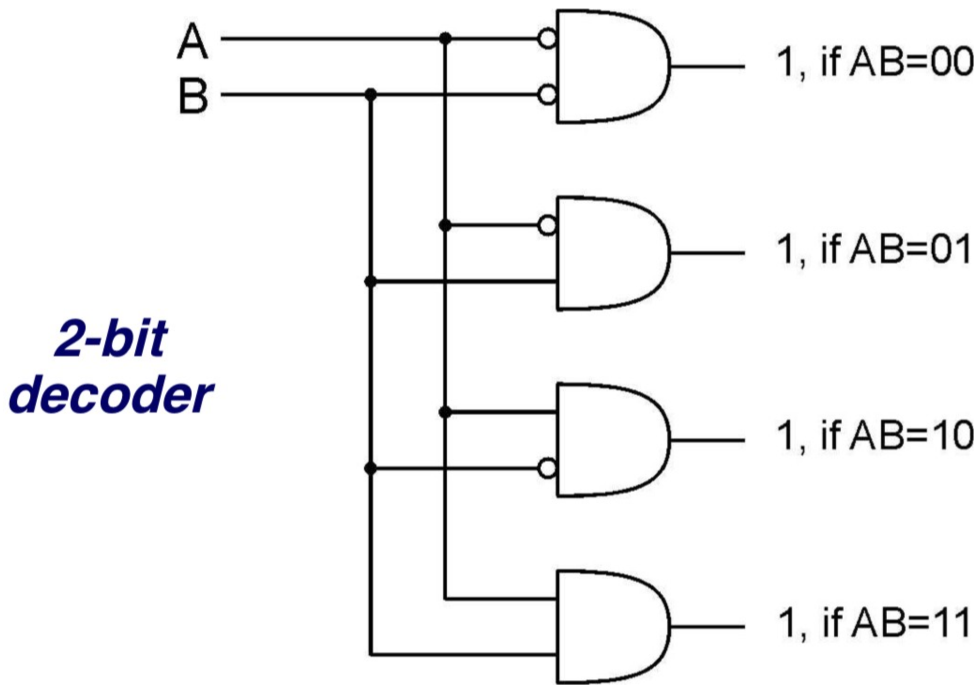
- Simplify the following expression:

$$\overline{A}\overline{B}C + \overline{A}BC + A\overline{B}C + ABC$$

- A: A
- B: B
- C: C
- D: AC + BC
- E: A + C

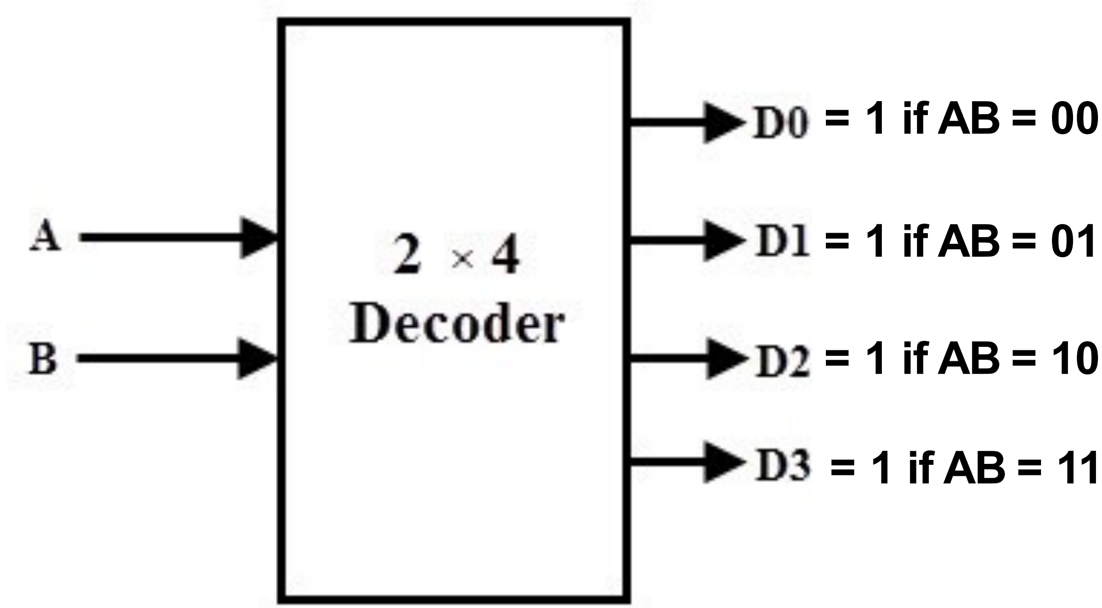
Decoder

- n inputs, 2^n outputs
- Exactly one output is 1 for a single possible input pattern



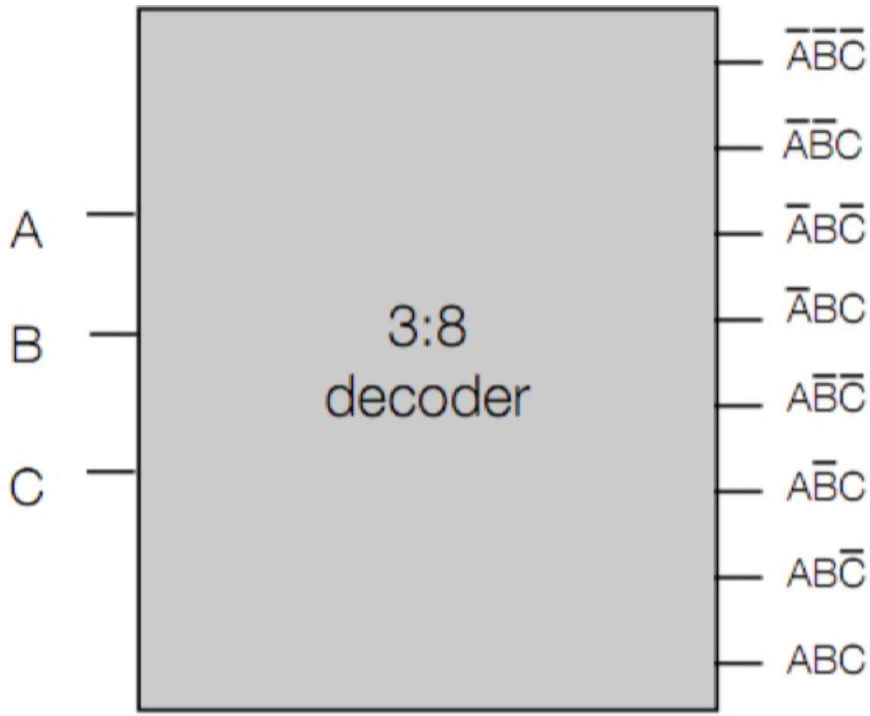
Decoder circuit

- n inputs, 2^n outputs
- Exactly one output is 1 for a single possible input pattern



Decoder Circuit

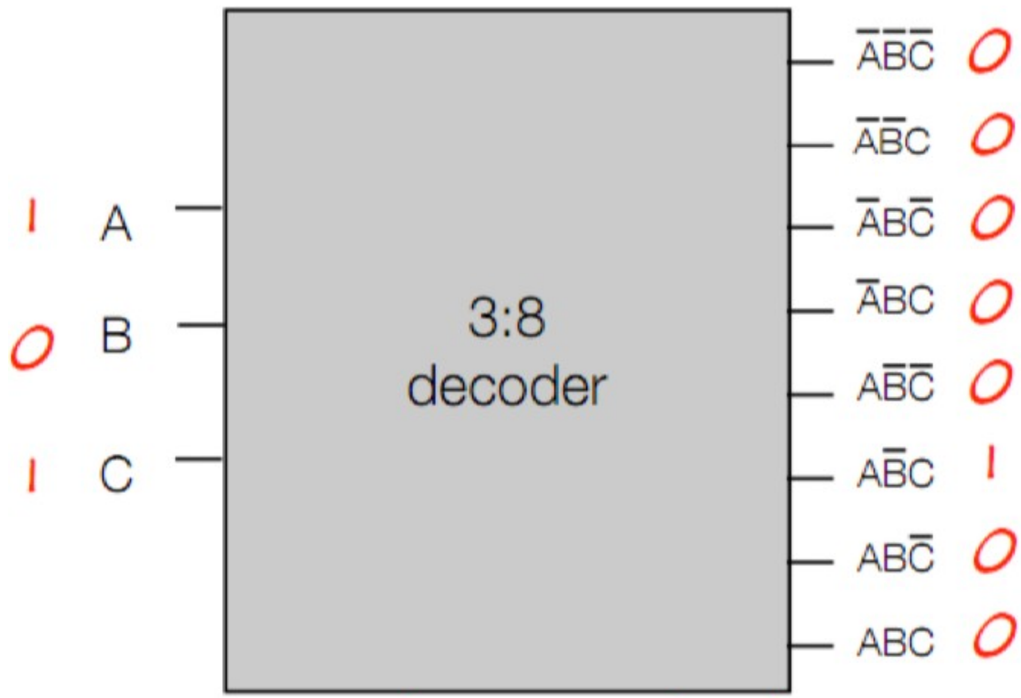
- Converts n-bit input to 2^n bit output



"Standard" Decoder: i^{th} output = 1, all others = 0, where i is the binary representation of the input (ABC)

Decoder Circuit

- Converts n-bit input to 2^n bit output



e.g., ABC = 101 (i=5)

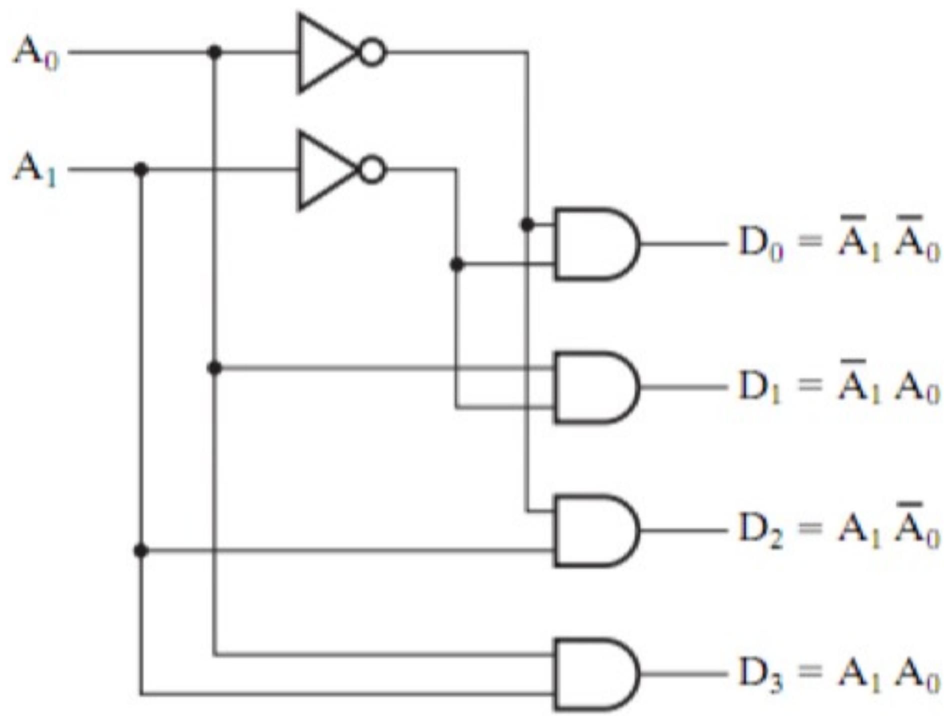
"Standard" Decoder: i^{th} output = 1, all others = 0,

where i is the binary representation of the input (ABC)

Internal of 2:4 Decoder

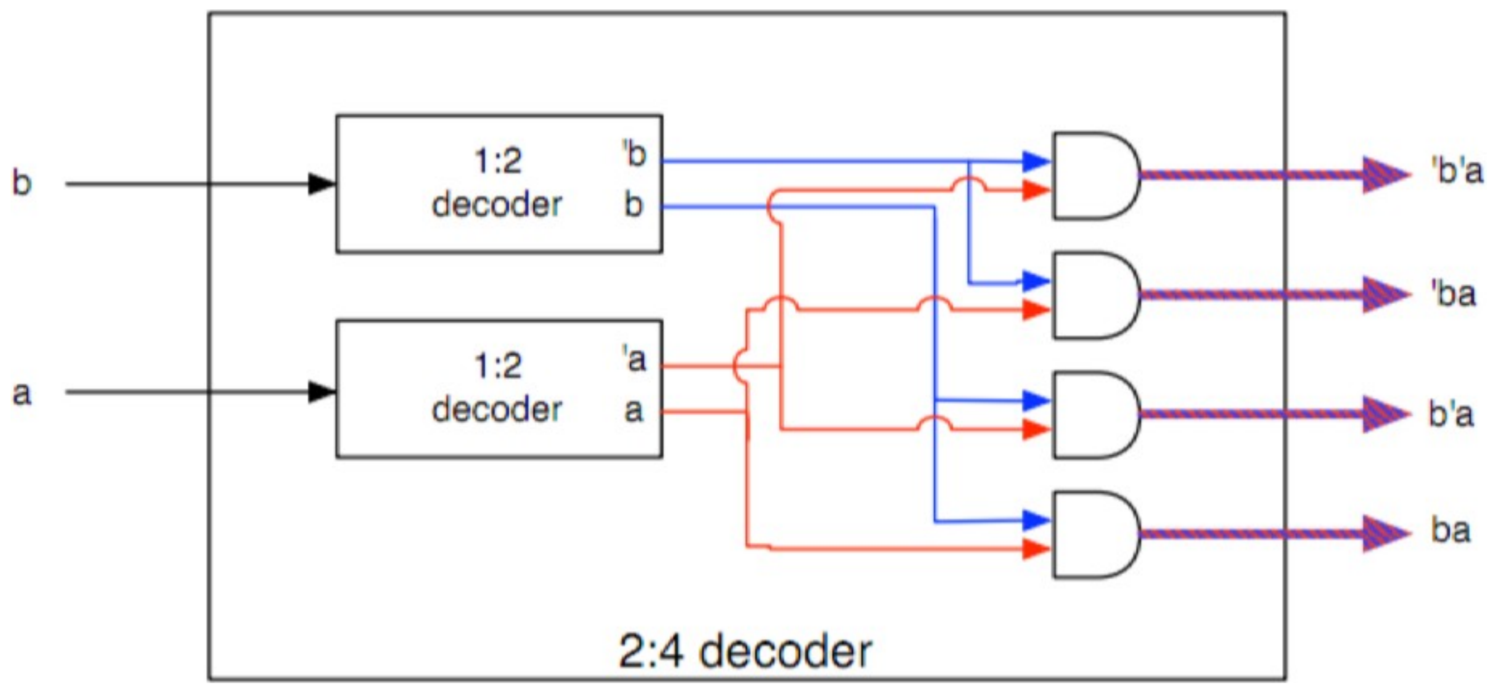
A_1	A_0	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

(a)

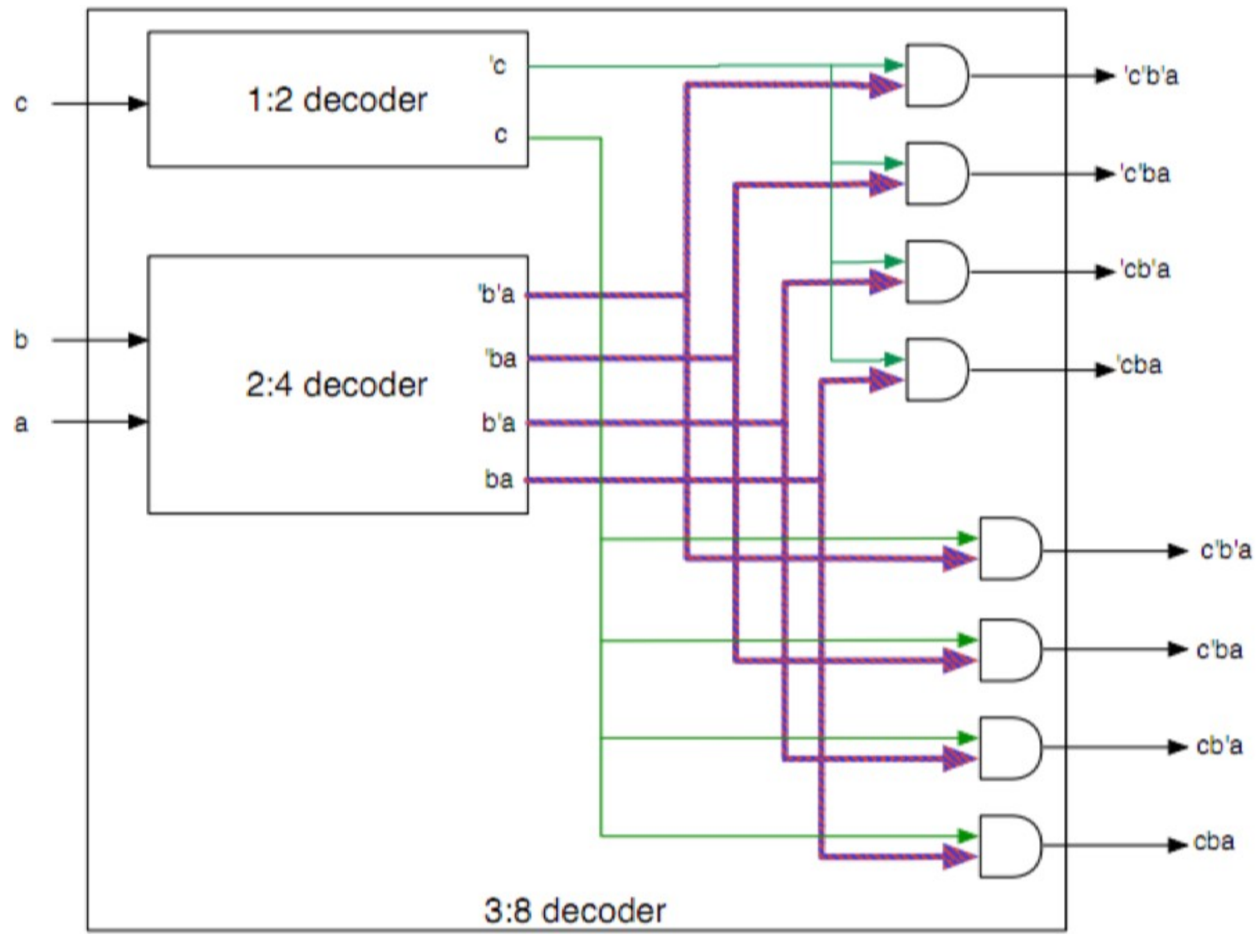


(b)

2:4 Decoder from 1:2 Decoders



3:8 Decoder from Smaller Decoders



Encoder

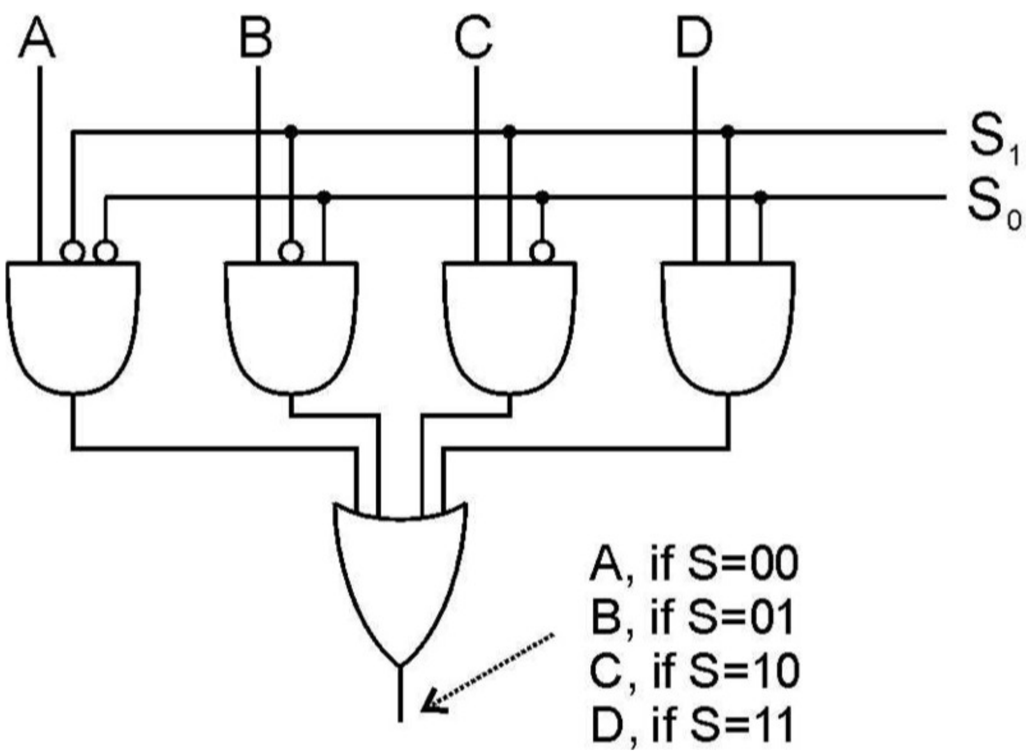
- Inverse of decoder:

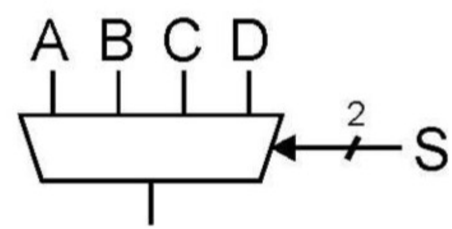
TABLE 3-7
Truth Table for Octal-to-Binary Encoder

Inputs								Outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

Multiplexer (MUX)

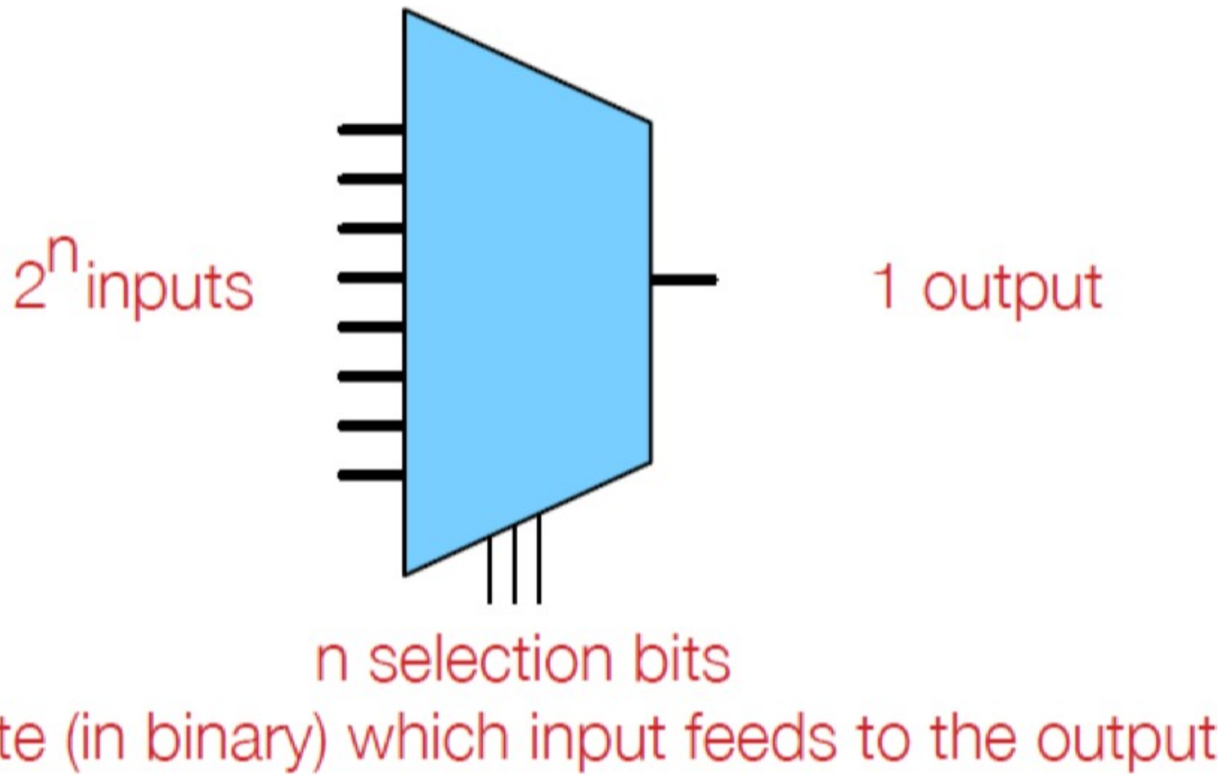
- 2^n inputs, n-bit selector, one output
  - Output equals one of the inputs, depending on the selector



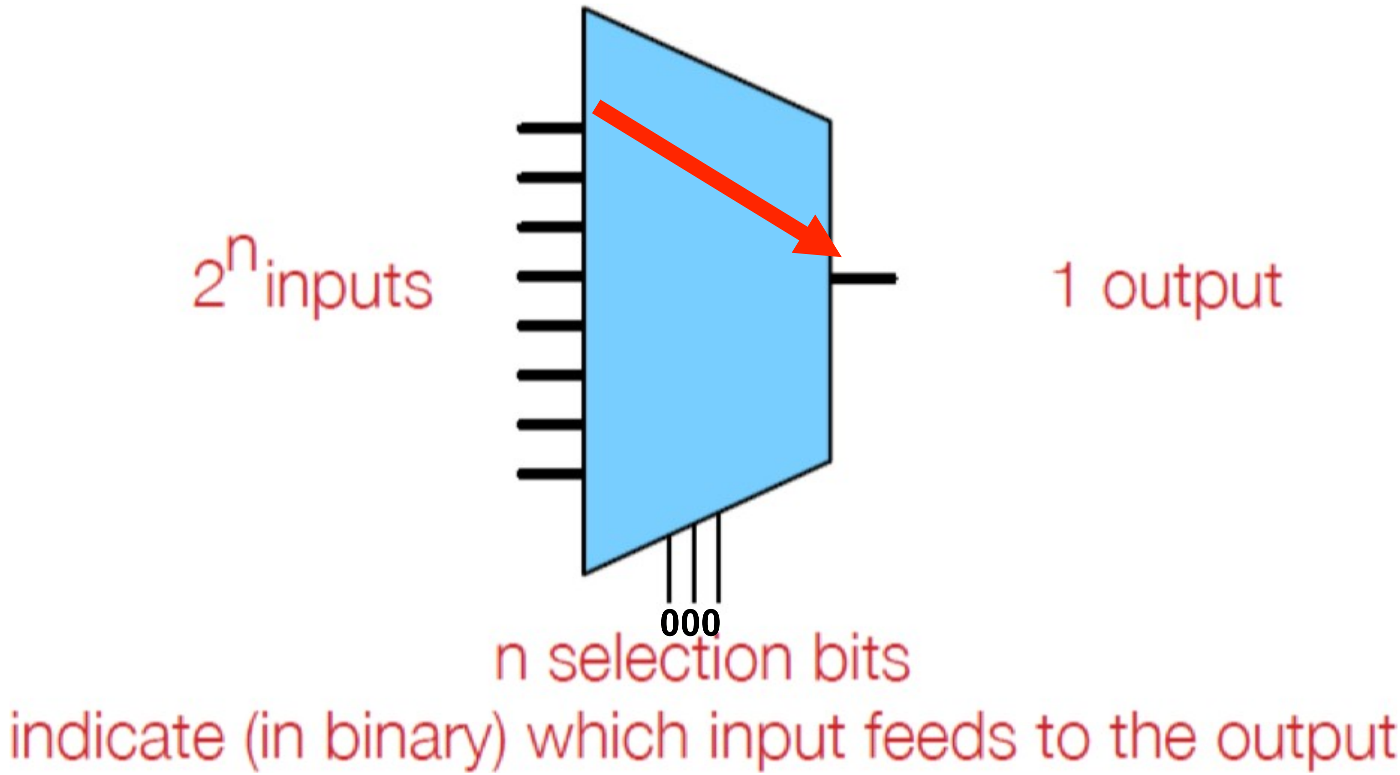


4-to-1 MUX

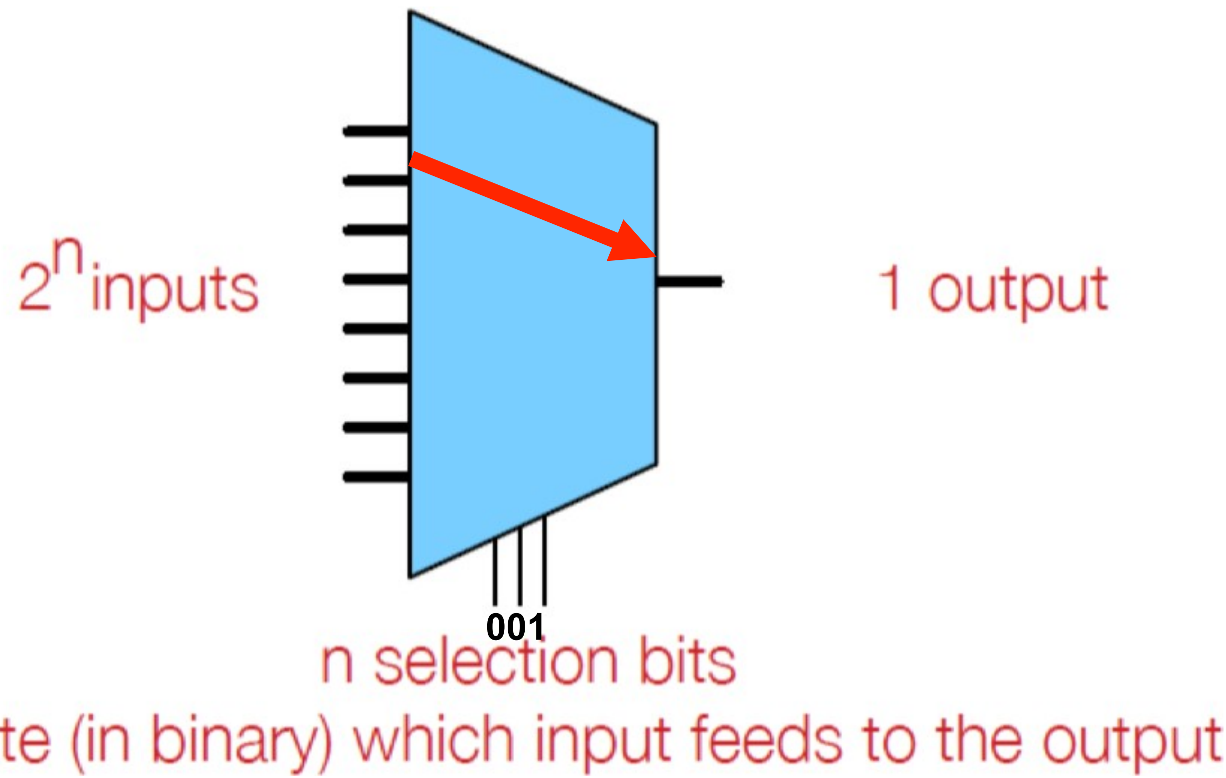
Multiplexer (MUX) Circuit



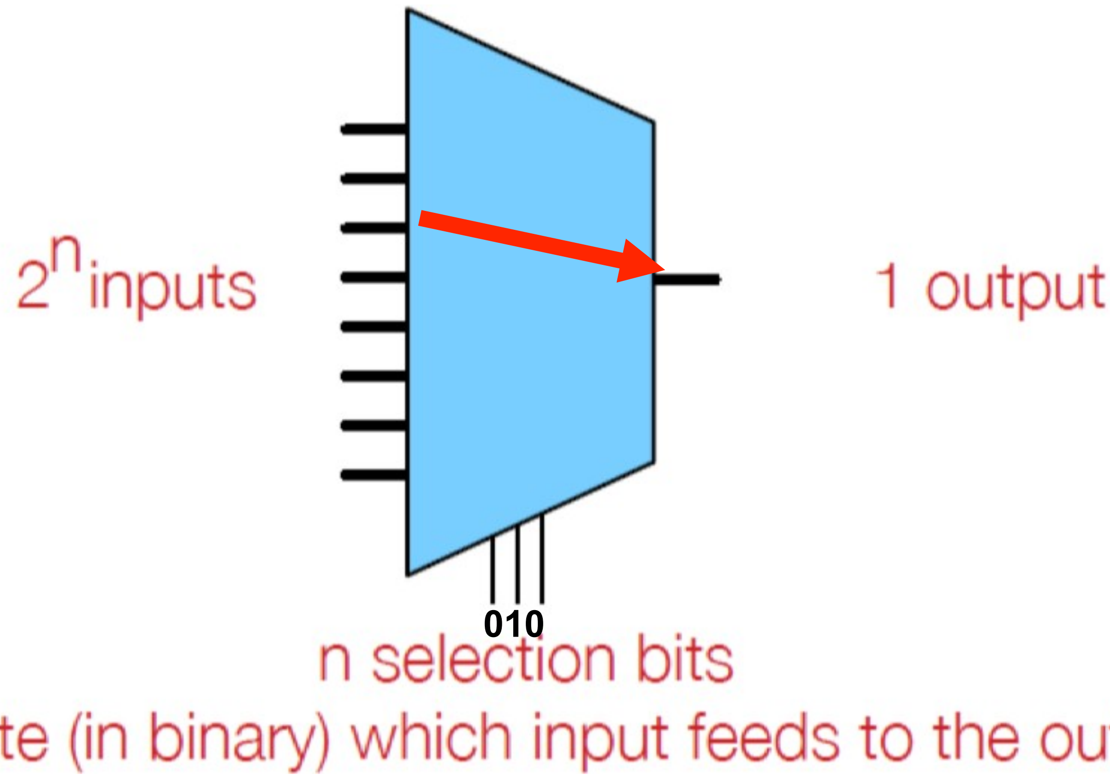
Multiplexer (MUX) Circuit



Multiplexer (MUX) Circuit



Multiplexer (MUX) Circuit



Functions with Decoders and Multiplexers

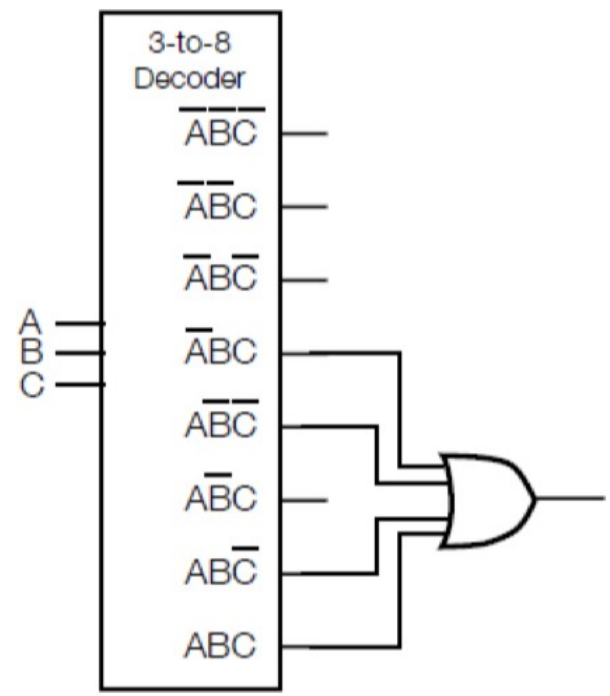
- e.g., $F = A\overline{C} + BC$

A	B	C	minterm	F
0	0	0	$\overline{A}\overline{B}\overline{C}$	0
0	0	1	$\overline{A}\overline{B}C$	0
0	1	0	$\overline{A}B\overline{C}$	0
0	1	1	$\overline{A}BC$	1
1	0	0	$A\overline{B}\overline{C}$	1
1	0	1	$A\overline{B}C$	0
1	1	0	$AB\overline{C}$	1
1	1	1	ABC	1

Functions with Decoders and Multiplexers

- e.g., $F = A\overline{C} + BC$

A	B	C	minterm	F
0	0	0	$\overline{A}\,\overline{B}\,\overline{C}$	0
0	0	1	$\overline{A}\,\overline{B}C$	0
0	1	0	$\overline{A}B\overline{C}$	0
0	1	1	$\overline{A}BC$	1
1	0	0	$A\overline{B}\,\overline{C}$	1
1	0	1	$A\overline{B}C$	0
1	1	0	$AB\overline{C}$	1
1	1	1	ABC	1

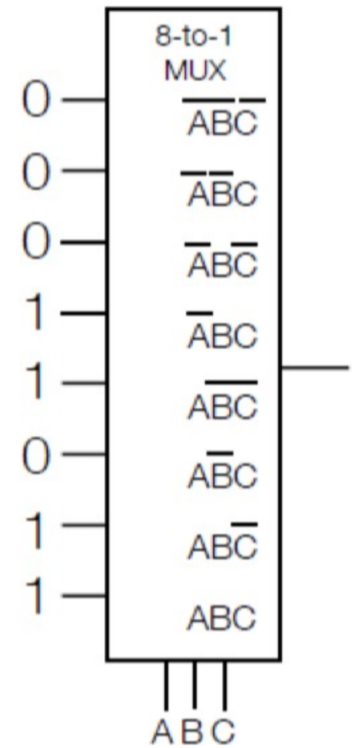


- OR minterms for which F should evaluate to 1

Functions with Decoders and Multiplexers

- e.g., $F = A\overline{C} + BC$

A	B	C	minterm	F
0	0	0	$\overline{A}\,\overline{B}\,\overline{C}$	0
0	0	1	$\overline{A}\,\overline{B}C$	0
0	1	0	$\overline{A}B\overline{C}$	0
0	1	1	$\overline{A}BC$	1
1	0	0	$A\overline{B}\,\overline{C}$	1
1	0	1	$A\overline{B}C$	0
1	1	0	$AB\overline{C}$	1
1	1	1	ABC	1

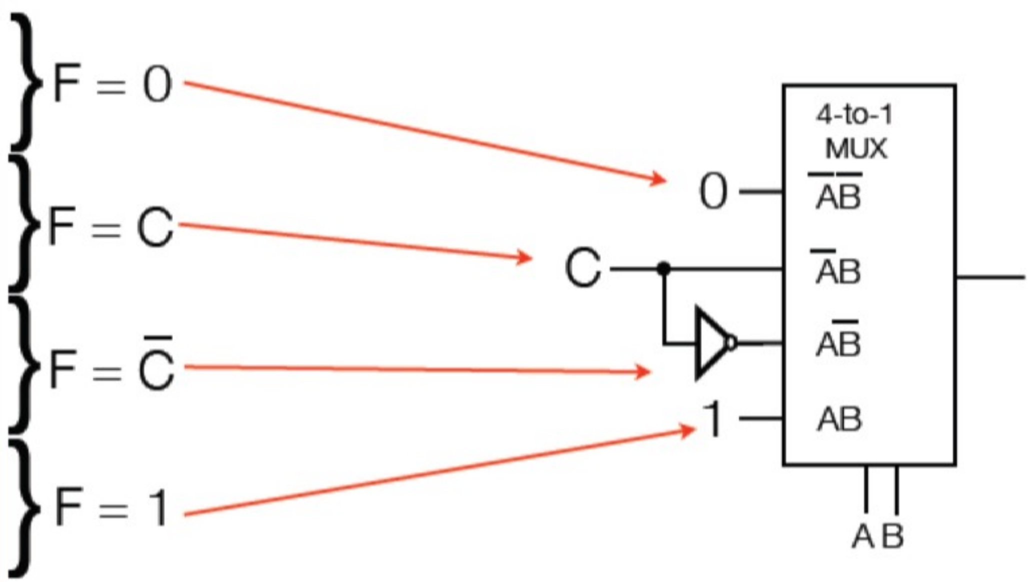


- Feed the value of F for each minterm in the input

Using Smaller mux:

- We can use 4-to-1 mux with a trick:
- Every two rows have same A and B value. The output F depends on the value C.

A	B	C	minterm	F
0	0	0	$\overline{A}\,\overline{B}\,\overline{C}$	0
0	0	1	$\overline{A}\,\overline{B}C$	0
0	1	0	$\overline{A}B\overline{C}$	0
0	1	1	$\overline{A}BC$	1
1	0	0	$A\overline{B}\,\overline{C}$	1
1	0	1	$A\overline{B}C$	0
1	1	0	$AB\overline{C}$	1
1	1	1	ABC	1

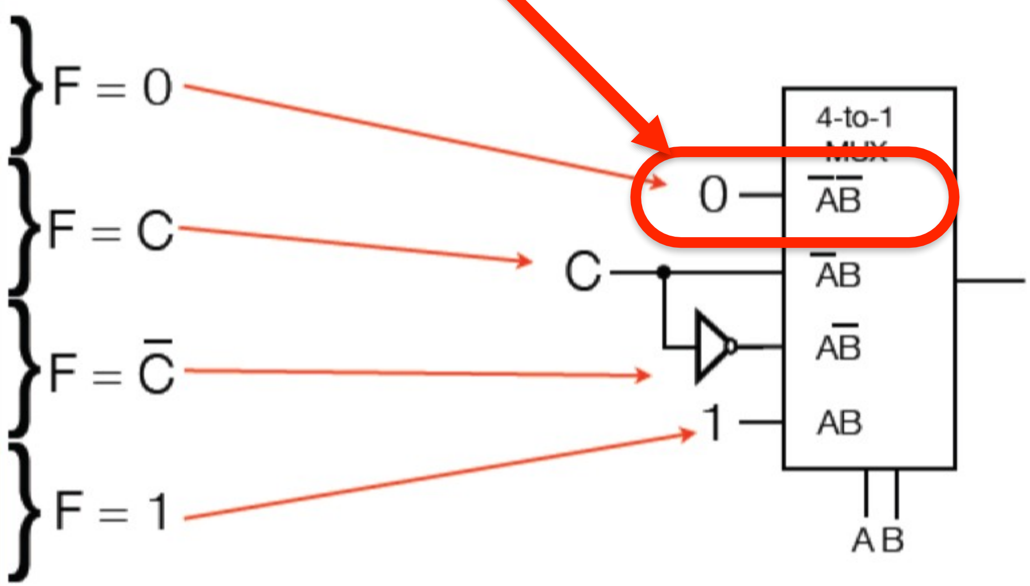


Using Smaller mux:

- We can use 4-to-1 mux with a trick:
- Every two rows have same A and B value. The output F depends on the val

If AB = 00, then F = 0

A	B	C	minterm	F
0	0	0	$\bar{A}\bar{B}\bar{C}$	0
0	0	1	$\bar{A}\bar{B}C$	0
0	1	0	$\bar{A}B\bar{C}$	0
0	1	1	$\bar{A}BC$	1
1	0	0	$A\bar{B}\bar{C}$	1
1	0	1	$A\bar{B}C$	0
1	1	0	$AB\bar{C}$	1
1	1	1	ABC	1



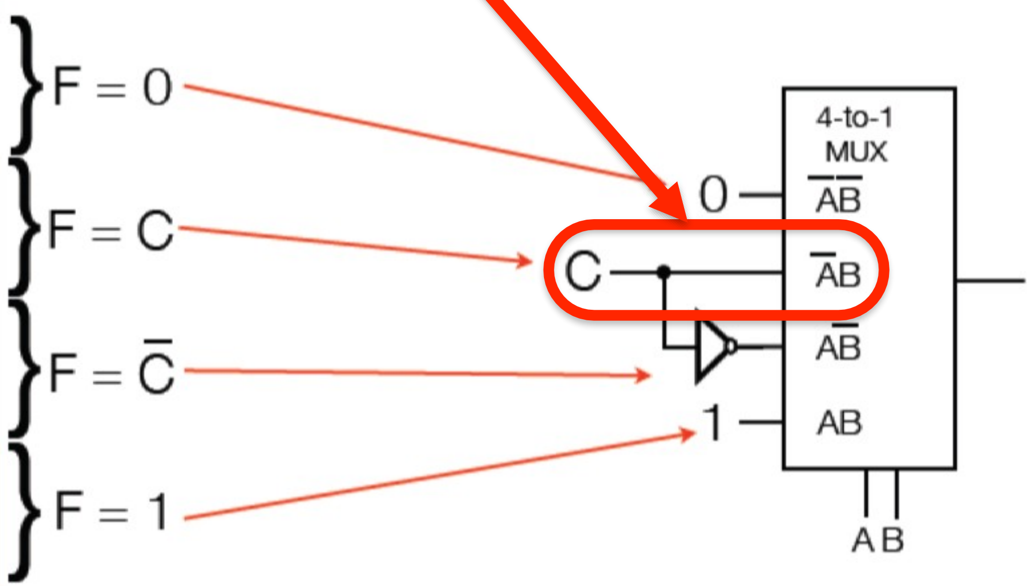
Using Smaller mux:

- We can use 4-to-1 mux with a trick:

- Every two rows have same A and B value. The output F depends on the valu

If AB = 01, then F = C

A	B	C	minterm	F
0	0	0	$\bar{A}\bar{B}\bar{C}$	0
0	0	1	$\bar{A}\bar{B}C$	0
0	1	0	$\bar{A}B\bar{C}$	0
0	1	1	$\bar{A}BC$	1
1	0	0	$A\bar{B}\bar{C}$	1
1	0	1	$A\bar{B}C$	0
1	1	0	$AB\bar{C}$	1
1	1	1	ABC	1



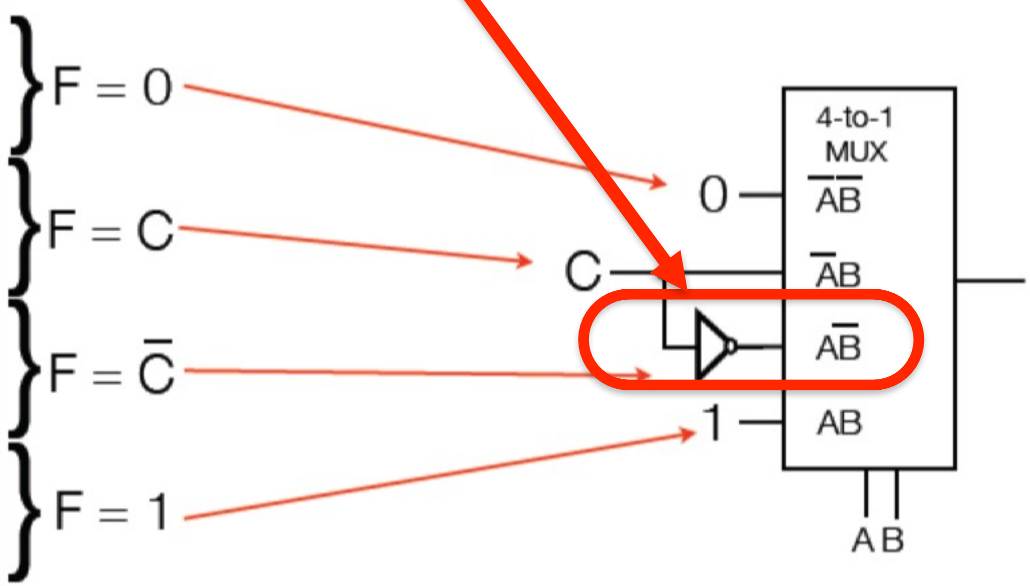
Using Smaller mux:

- We can use 4-to-1 mux with a trick:

- Every two rows have same A and B value. The output F depends on the valu

If AB = 10, then F = $\overline{C}$

A	B	C	minterm	F
0	0	0	$\overline{A}\,\overline{B}\,\overline{C}$	0
0	0	1	$\overline{A}\,\overline{B}C$	0
0	1	0	$\overline{A}B\overline{C}$	0
0	1	1	$\overline{A}BC$	1
1	0	0	$A\overline{B}\,\overline{C}$	1
1	0	1	$A\overline{B}C$	0
1	1	0	$AB\overline{C}$	1
1	1	1	ABC	1



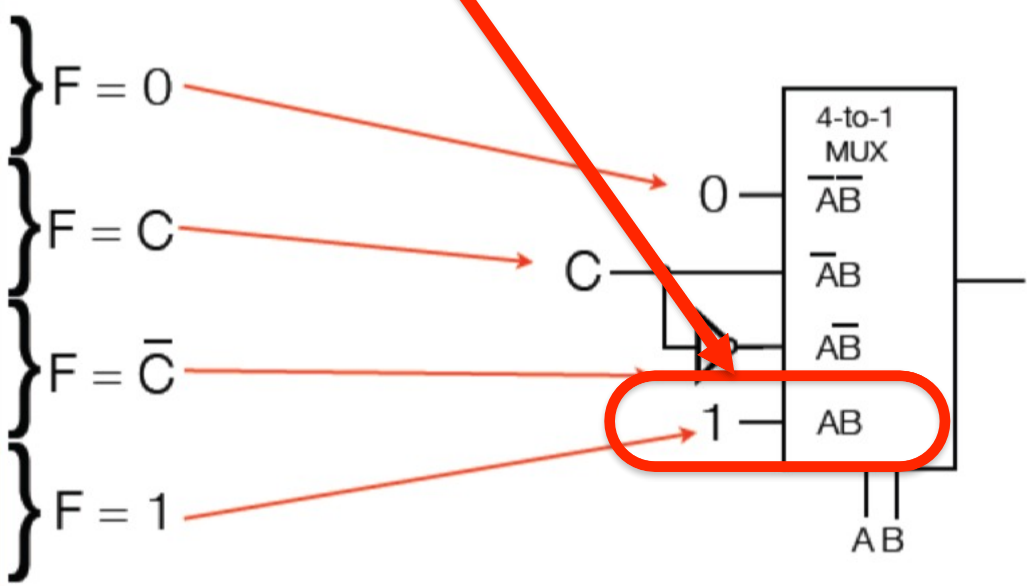
Using Smaller mux:

- We can use 4-to-1 mux with a trick:

- Every two rows have same A and B value. The output F depends on the valu

If AB = 11, then F = 1

A	B	C	minterm	F
0	0	0	$\bar{A}\bar{B}\bar{C}$	0
0	0	1	$\bar{A}\bar{B}C$	0
0	1	0	$\bar{A}B\bar{C}$	0
0	1	1	$\bar{A}BC$	1
1	0	0	$A\bar{B}\bar{C}$	1
1	0	1	$A\bar{B}C$	0
1	1	0	$AB\bar{C}$	1
1	1	1	ABC	1



Another Example

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

A	B	C	minterm	F
0	0	0	$\overline{A}\,\overline{B}\,\overline{C}$	1
0	0	1	$\overline{A}\,\overline{B}C$	1
0	1	0	$\overline{A}B\overline{C}$	0
0	1	1	$\overline{A}BC$	1
1	0	0	$A\overline{B}\,\overline{C}$	1
1	0	1	$A\overline{B}C$	0
1	1	0	$AB\overline{C}$	1
1	1	1	ABC	0

$F = 1$

$F = C$

$F = \overline{C}$

$F = \overline{C}$