TUTORIAL 1 COMBINATIONAL LOGIC CIRCUIT

Overview

- **We will review the following concept in this tutorial:**
- **Combinational logic circuit**
  - No memory, output(s) solely determined by input(s)
- **Truth table and logic function**
- **Two-level logic and PLA**
- **Simplification with Boolean Algebra and K-map**
- **Circuit design**

- **Work with two practical examples**
  - Bit comparator
  - Encoder

Digital Logic Circuit

- **Two types of digital logic circuits inside a computer:**
  - **Combinational logic circuits**
    - Logic circuits that do not have memory
    - The output depends only on the current inputs and the circuit
    - They can be specified fully with a truth table or a logic equation
  - **Sequential logic circuits**
    - Logic circuits that have memory
    - The output depends on both the current inputs and the value stored in memory (called state)

Circuit Design Process

- **A simple logic design process involves**
  - Problem specification
  - Truth table derivation
  - Derivation of logical expression
  - Simplification of logical expression
  - Implementation

Review of Boolean Algebra

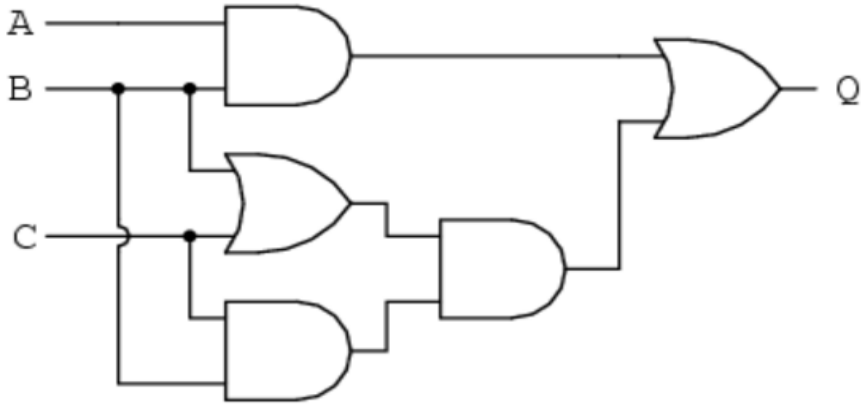
- **Boolean algebra consists of**
  - Boolean variables (with values equal to either ‘0’ or ‘1’)
  - Binary operators: AND (·), OR (+), NOT (‘)
- **Any logic function can be expressed as a two-level logic expression, either as**
  - Sum-of-Products (SoP) representation, or
  - Product-of-Sums (PoS) representation
- **The AND, OR, and NOT operations form a functionally complete set (namely, universal gates), as they can specify any logic function.**

Basic Laws of Boolean Algebra

Name	AND Form	OR Form
Identity Law	$1A = A$	$0 + A = A$
Null Law	$0A = 0$	$1 + A = 1$
Idempotent Law	$AA = A$	$A + A = A$
Inverse Law	$A\overline{A} = 0$	$A + \overline{A} = 1$
Commutative Law	$AB = BA$	$A + B = B + A$
Associative Law	$(AB)C = A(BC)$	$(A + B) + C = A + (B + C)$
Distributive Law	$A + BC = (A + B)(A + C)$	$A(B + C) = AB + AC$
Absorption Law	$A(A + B) = A$	$A + AB = A$
De Morgan's Law	$\overline{AB} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A}\overline{B}$

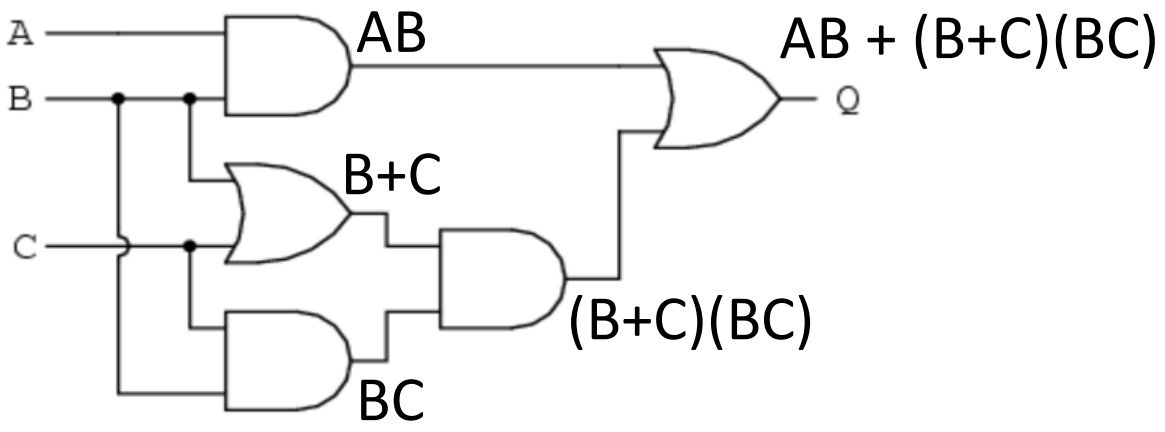
Boolean Algebra Exercise 1

- Find the direct Boolean expression for the following circuit
- Simplify the Boolean expression using Boolean algebra
- Draw the new circuit for the simplified Boolean expression

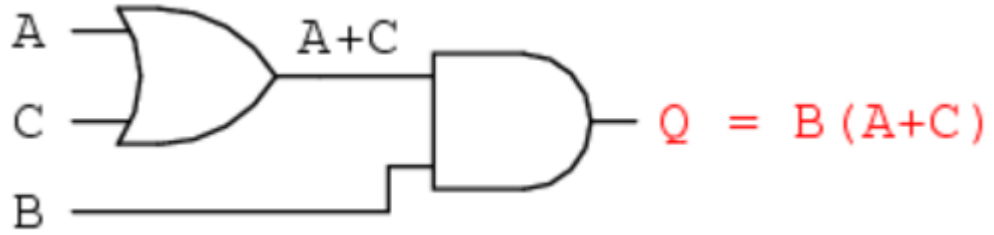


Boolean Algebra Exercise 1

- Find the direct Boolean expression for the following circuit
- Simplify the Boolean expression using Boolean algebra
- Draw the new circuit for the simplified Boolean expression

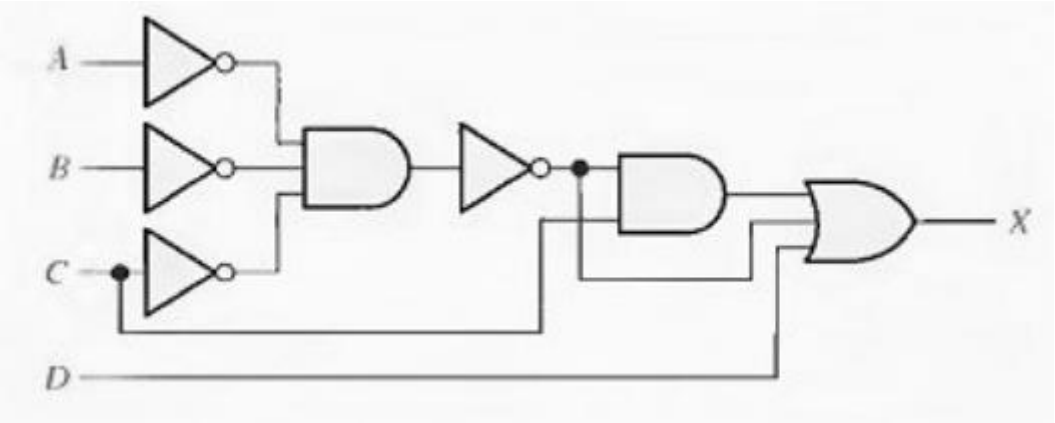


Q = AB + (B+C)(BC)
= AB + BBC + BCC
= AB + BC + BC
= AB + BC
= B(A+C)



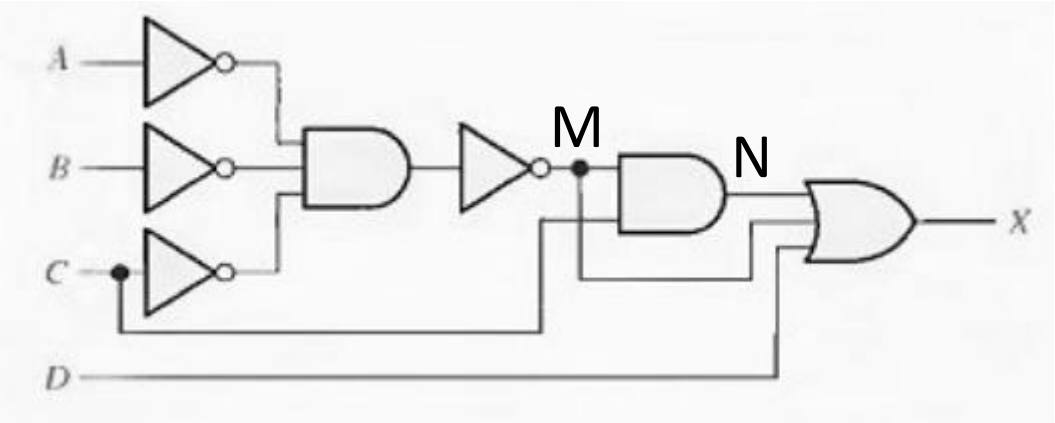
Boolean Algebra Exercise 2

- **Simplify the combinational logic circuit shown below to a minimum form**



Boolean Algebra Exercise 2

- **Simplify the combinational logic circuit shown below to a minimum form**



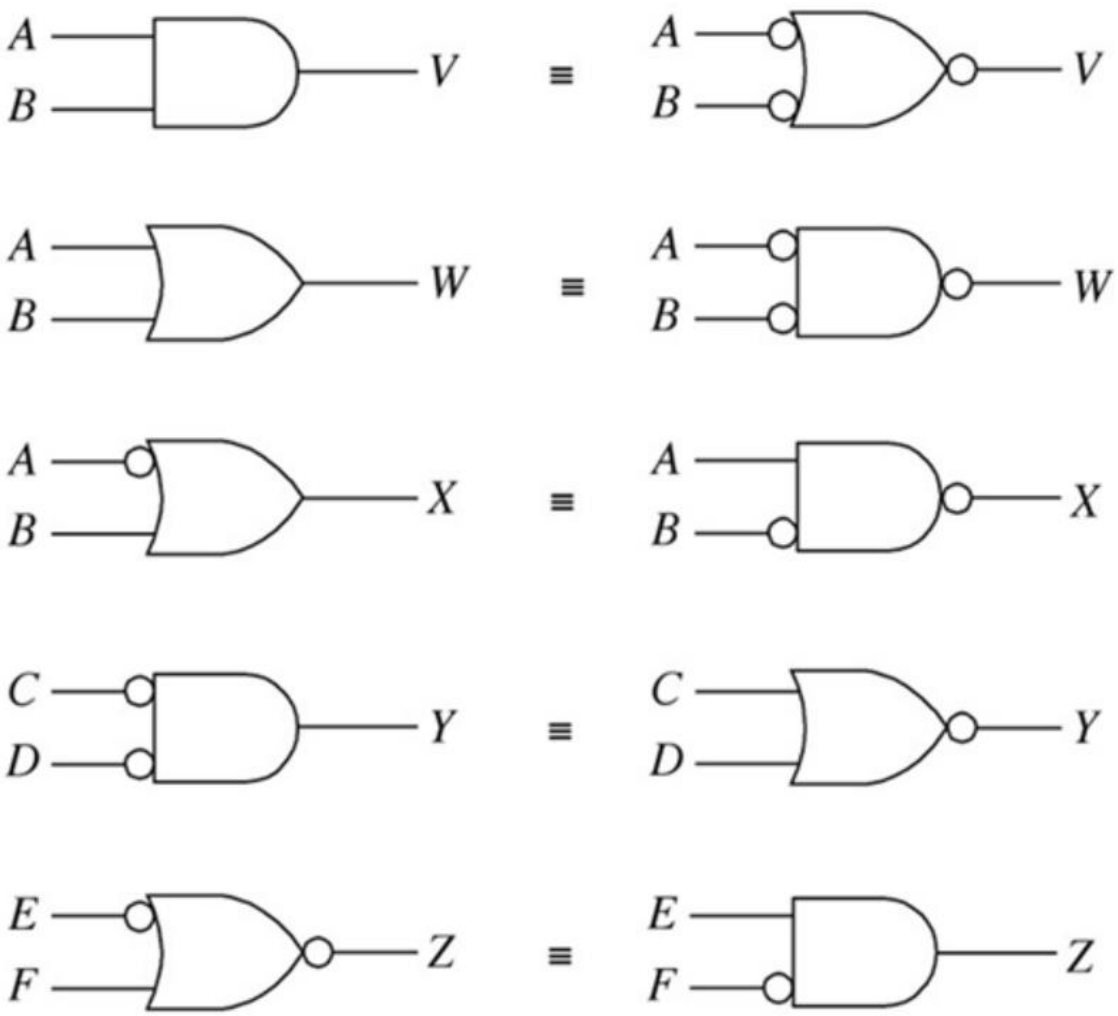
$M = \overline{\overline{A}\overline{B}\overline{C}} = A + B + C$

Using De Morgan's Law (Bubble Pushing)

N = CM = C(A + B + C)
= AC + BC + CC
= AC + BC + C
= C(A + B + 1)
= C(1) = C

X = D + M + N
= D + (A + B + C) + C
= A + B + C + D

De Morgan's Law (Bubble Pushing)



- Not Gates before and after AND/OR gates can be represented just as the "Bubble".
- "Convert" the gate between AND/OR, as well as invert the "Bubbles" at all the inputs and the output.

- $\overline{A + B} = \overline{A}\overline{B}$
- $\overline{AB} = \overline{A} + \overline{B}$
- $\overline{\overline{A}\overline{B}\overline{C}} = A + B + C$

Boolean Algebra Exercise 3

- **Simplify the following Boolean expression**

$$AB + \overline{AB}CD + \overline{CD}EF$$

Boolean Algebra Exercise 3

- **Simplify the following Boolean expression**

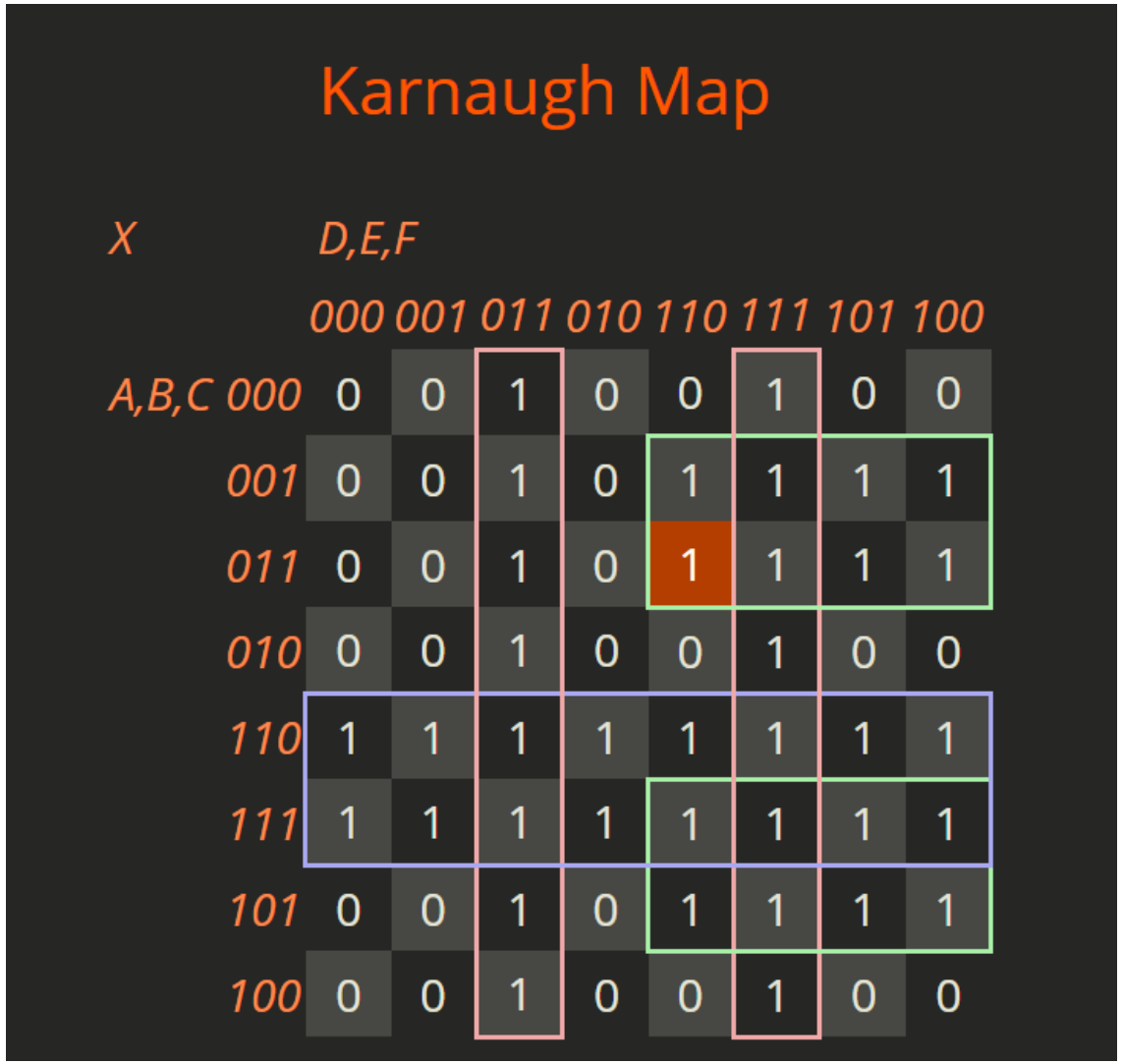
$$
\begin{aligned}
& AB + \overline{AB}CD + \overline{CD}EF \\
&= (AB(1 + CD)) + \overline{AB}CD + \overline{CD}EF \\
&= (AB + ABCD) + \overline{AB}CD + \overline{CD}EF \\
&= AB + (ABCD + \overline{AB}CD) + \overline{CD}EF \\
&= AB + CD(AB + \overline{AB}) + \overline{CD}EF \\
&= AB + CD + \overline{CD}EF \\
&= AB + (CD + CDEF) + \overline{CD}EF \\
&= AB + CD + EF(CD + \overline{CD}) \\
&= AB + CD + EF
\end{aligned}
$$

Boolean Algebra Exercise 3 (Truth Table)

- **Simplification via Boolean Algebra axioms can be tedious.**
  - Especially when the number of inputs increases.

- **Truth Table and K-map can be a faster way to simplify.**
  - See Excel File for full truth table.

Boolean Algebra Exercise 3 (K-map)

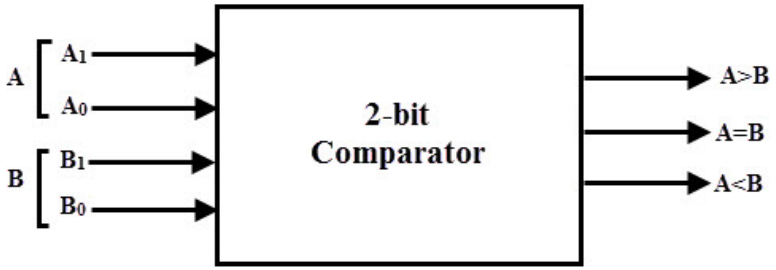


- **Three groupings:**
  - Blue: AB
  - Green: CD
  - Red: EF
  - X = AB + CD + EF

- **6-variable K-map adjacency:**
  - Top and bottom rows.
  - Left and right columns.
  - Rows and Columns where only 1 input changes, i.e., (011, 111)
  - This is why the Green grouping seems disjointed but is actually adjacent. Similar for the Red grouping.

2-bit Comparator

- Here we'll be designing circuits to compare 2-bit binary numbers.
- Suppose we have two 2-bit numbers A & B at the inputs, and three outputs as A>B, A==B, A<B
- Only one of the three outputs would be true accordingly if A is greater than or equal to or less than B.
- We'll practice the circuit design for f(A==B), try to work on f(A>B) and f(A<B) by yourself



Solution: 2-bit Comparator Truth Table

$A(A_1A_0)$	$B(B_1B_0)$	**f (A>B)**	**f (A==B)**	**f (A<B)**
00	00	*0*	*1*	*0*
00	01	*0*	*0*	*1*
00	10	*0*	*0*	*1*
00	11	*0*	*0*	*1*
01	00	*1*	*0*	*0*
01	01	*0*	*1*	*0*
01	10	*0*	*0*	*1*
01	11	*0*	*0*	*1*
10	00	*1*	*0*	*0*
10	01	*1*	*0*	*0*
10	10	*0*	*1*	*0*
10	11	*0*	*0*	*1*
11	00	*1*	*0*	*0*
11	01	*1*	*0*	*0*
11	10	*1*	*0*	*0*
11	11	*0*	*1*	*0*

Solution: Logic Function for f(A==B)

- **Four output rows of f(A==B) that has value of '1':**
  - Four corresponding input rows:
    (A=00,B=00) , (A=01,B=01) , (A=10,B=10) , (A=11,B=11).
  - The four 1-minterms are:

$$\overline{A_1}\,\overline{A_0}\,\overline{B_1}\,\overline{B_0}\,,\ \overline{A_1}A_0\overline{B_1}B_0\,,\ A_1\overline{A_0}B_1\overline{B_0}\,,\ A_1A_0B_1B_0$$

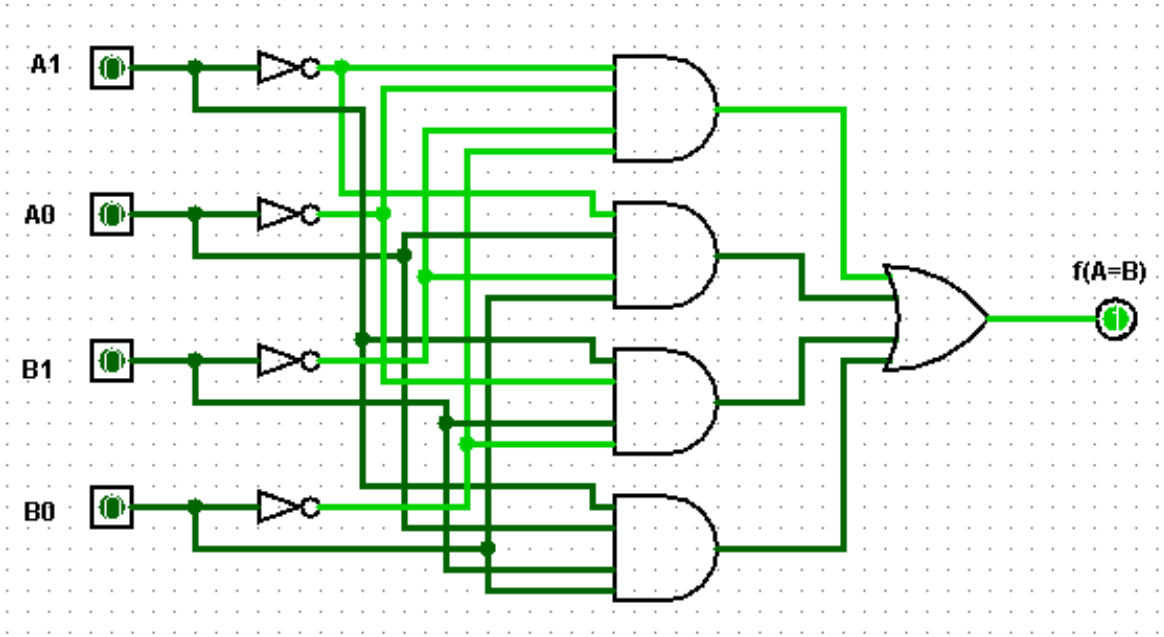
- **The output is the OR operation of the four 1-minterms:**

$$f(A == B) = \overline{A_1}\,\overline{A_0}\,\overline{B_1}\,\overline{B_0} + \overline{A_1}A_0\overline{B_1}B_0 + A_1\overline{A_0}B_1\overline{B_0} + A_1A_0B_1B_0$$

- **Unfortunately, no further simplification possible.**

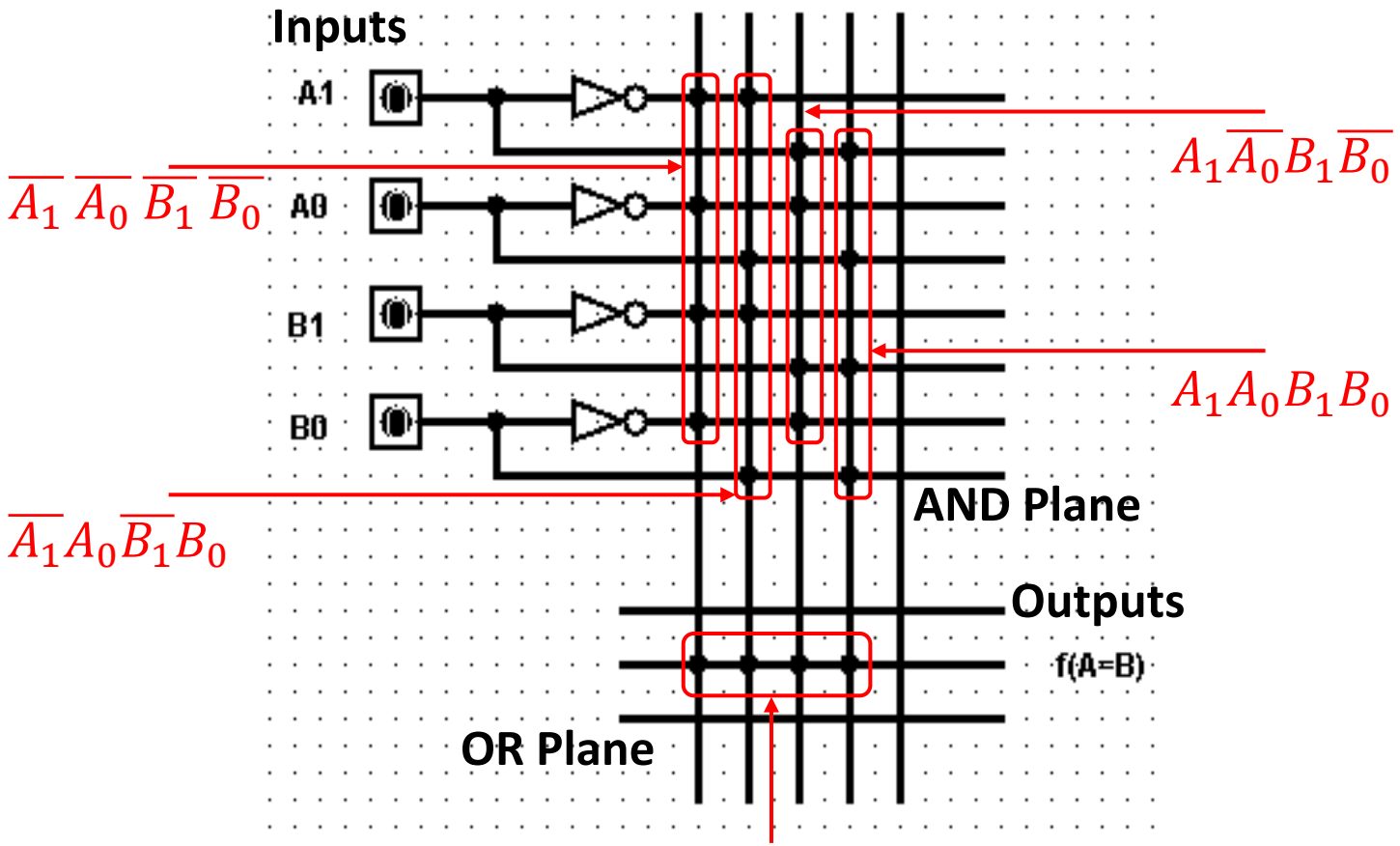
Solution: Circuit for f(A==B)

$$f(A == B)$$
$$= \overline{A_1}\,\overline{A_0}\,\overline{B_1}\,\overline{B_0} + \overline{A_1}A_0\overline{B_1}B_0 + A_1\overline{A_0}B_1\overline{B_0} + A_1A_0B_1B_0$$



Solution: PLA Implementation

- The same circuit can be equivalently represented by a **programmable logic array** (PLA) circuit.

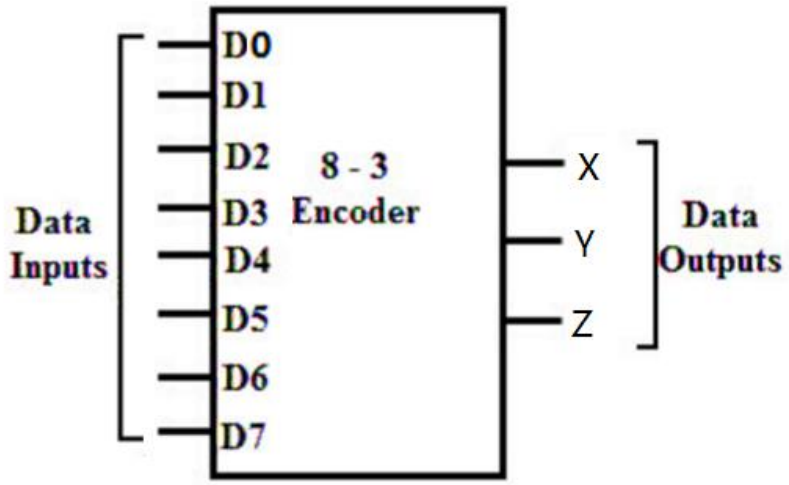


$$f(A == B) = \overline{A_1}\,\overline{A_0}\,\overline{B_1}\,\overline{B_0} + \overline{A_1}A_0\overline{B_1}B_0 + A_1\overline{A_0}B_1\overline{B_0} + A_1A_0B_1B_0$$

8-to-3 Encoder

- An encoder (2^N-to-N encoder) is a logical block with an 2^N-bit input and N 1-bit outputs, which performs the inverse function of a decoder.

- Example (8-to-3 encoder)
  - 8 inputs (D_0, D_1, …, D_7) and 3 outputs (X, Y, Z)

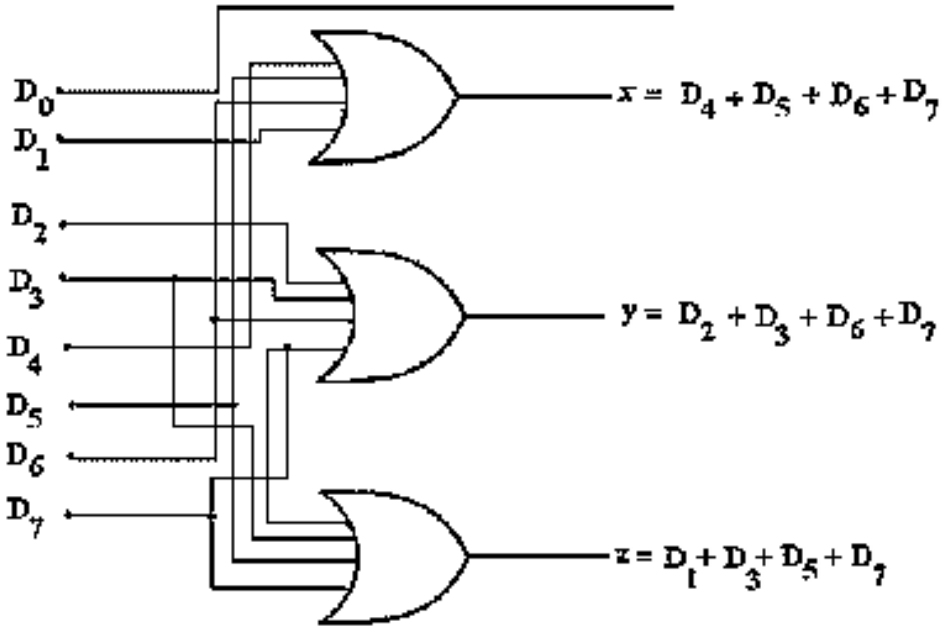


8-to-3 Encoder Truth Table

D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	**X**	**Y**	**Z**
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*

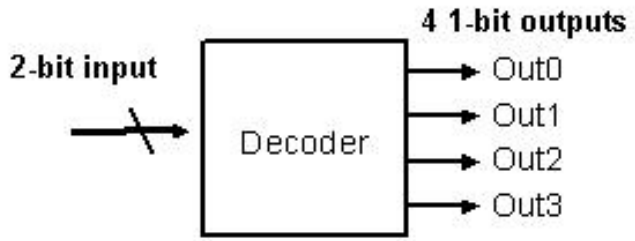
Solution: Logic Function and Circuit

- $X = D_4 + D_5 + D_6 + D_7$
- $Y = D_2 + D_3 + D_6 + D_7$
- $Z = D_1 + D_3 + D_5 + D_7$



Extra Exercise: Decoder

- A decoder takes a single N-bit input and outputs 2^N 1-bit signals. The 1-bit output corresponds to the N-bit input bit pattern is true while all other outputs are false.
- The following figure shows a block diagram for a 2-to-4 decoder.



A 2-bit decoder

Questions

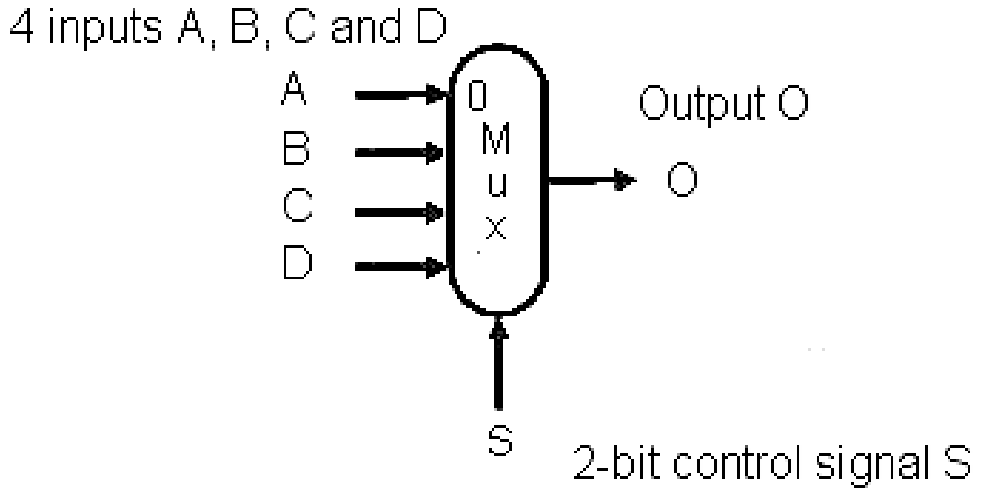
- Why can a 2-bit input generate 4 outputs in the decoder?
- If the input bits are 11, what will happen to the outputs of the decoder?
- Is it possible to have more than one outputs asserted (=1)?
- Name two potential uses of the decoder.
- Implement the decoder using Logisim.

Questions

- **Why can a 2-bit input generate 4 outputs in the decoder?**
  - $2^2 = 4$.
- **If the input bits are 11, what will happen to the outputs of the decoder?**
  - Out3 = 1, Out2 = 0, Out1= 0, Out0 = 0.
- **Is it possible to have more than one outputs asserted (=1)?**
  - Logically no for Line Decoders. Can be a feature in special decoders.
- **Name two potential uses of the decoder.**
  - Reduce the number of control lines needed, if only 1-out-of-N devices need to be active at any single timeframe.
  - Convert digital signals to analog signals, e.g., synchronize multiple motors.

Extra Exercise: Multiplexor

- A multiplexor is a devices that given the control signal, selects one of the inputs to be forwarded to the output. The following figure shows a 4-input multiplexor.
- 4-to-1 multiplexor



Questions

- If the inputs A/B are 32-bit in width, what is the data width of the Output O?
- What is the maximum number of inputs if the control signal is 10-bit in width?
- What is the bit-width of the control signal for the multiplexor if there are 9 inputs?

Questions

- **If the inputs A/B/C/D are 32-bit in width, what is the data width of the Output O?**
  - 32-bit.
- **What is the maximum number of inputs if the control signal is 10-bit in width?**
  - $2^{10} = 1024$.
- **What is the bit-width of the control signal for the multiplexor if there are 9 inputs?**
  - 4 bits. $2^4 = 16$, while $2^3 = 8$ is insufficient.