

# Digital Logic

ENEL 111

## Digital systems

- A digital system is a system whose inputs and outputs fall within a discrete, finite set of values
- Two main types
  - Combinational
    - Outputs dependent only on current input
  - Sequential
    - Outputs dependent on both past and present inputs

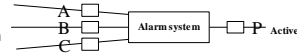


## Combinational Logic Circuits

- Aims
  - To express the inputs and outputs of a system in binary form
  - To develop the relationships between these inputs and outputs as a truth table
  - To simplify the Boolean expression using algebra or Karnaugh maps
  - To select suitable electronic devices to implement the required function

## Example

- Consider a buzzer which sounds when :
  - The lights are on and
  - The door is open and
  - No key is in the ignition



Variable	Value	Situation
A	1	Lights are on
A	0	Lights are off
B	1	Door is open
B	0	Door is closed
C	1	Key is in ignition
C	0	Key is out of ignition
P	1	Buzzer is on
P	0	Buzzer is off

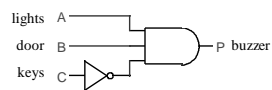
## Example

### ■ Truth Table

- A Truth Table can be used to show the relationships between :
  - the 3 inputs and
  - the single output

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

- Implementation as a circuit using logic gates



## Summary

- Inputs and Outputs are expressed in Binary Form
- A truth table showing relationships between inputs and outputs is constructed
- A circuit is built to implement the circuit

## This lecture

- Truth tables for primitive functions
- Boolean notation
- Sum of products
- Boolean algebra

## Truth Tables and Boolean Notation

### ■ Circuits with one input

- Buffer  $P = A$ 

A	P
0	0
1	1

- Not  $P = \bar{A}$ 

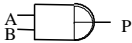
A	P
0	1
1	0

## Basic AND / OR

### ■ Circuits with two Inputs


□ AND  $P = A \cdot B$

A	B	P
0	0	0
0	1	0
1	0	0
1	1	1



□ OR  $P = A + B$

A	B	P
0	0	0
0	1	1
1	0	1
1	1	1

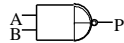


## Basic NAND / NOR

### ■ Problems with two Inputs


□ NAND  $P = \overline{A \cdot B}$

A	B	P
0	0	1
0	1	1
1	0	1
1	1	0



□ NOR  $P = \overline{A + B}$

A	B	P
0	0	1
0	1	0
1	0	0
1	1	0




## Basic XOR / XNOR

### ■ Circuits with two Inputs:


□ XOR  $P = A \oplus B$

A	B	P
0	0	0
0	1	1
1	0	1
1	1	0



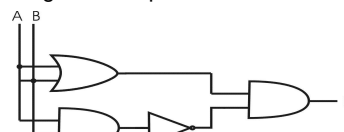
□ XNOR  $P = \overline{A \oplus B}$

A	B	P
0	0	1
0	1	0
1	0	0
1	1	1



## Primitive gates

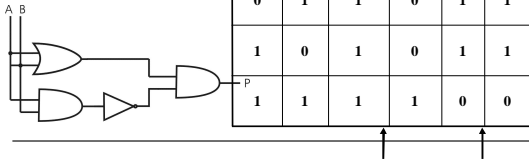
- All circuits can actually be made using AND, OR and NOT gates if required.



In terms of components used, it is generally easier to build inverting functions. They typically require less transistors and also work faster than their non-inverting cousins.

## Exercise

Complete the truth table for this circuit and name the equivalent primitive function/gate.



## Not Symbol

- You should be aware that

not A and not B  $\overline{A}.\overline{B}$

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

and

not (A and B) equivalent to NAND  $\overline{A.B}$

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

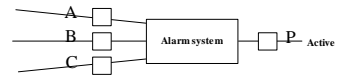
are different.

## Combinational Logic Circuits

- Reminder from our overview
  - To express the inputs and outputs of a system in binary form
  - To develop the relationships between these inputs and outputs as a truth table
  - To simplify the Boolean expression using algebra or Karnaugh maps
  - To select suitable electronic devices to implement the required function

## Our Example

- Consider a buzzer which sounds when :

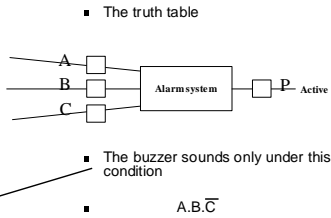


- The lights are on **and**
- The door is open **and**
- No key is in the ignition

Variable	Value	Situation
A	1	Lights are on
A	0	Lights are off
B	1	Door is open
B	0	Door is closed
C	1	Key is in ignition
C	0	Key is out of ignition
P	1	Buzzer is on
P	0	Buzzer is off

Very simple!

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



Slightly more complex

□ Consider my car which complains by sounding a buzzer when I have left the lights on or left the car in gear (not in Park) and taken the keys out of the ignition:

A (lights on = 1)	B (in gear = 1)	C (keys out = 0)	P (buzzer)	what I've done
0	0	0	0	
0	0	1	0	
0	1	0	1	left in gear
0	1	1	0	
1	0	0	1	left lights on
1	0	1	0	
1	1	0	1	both
1	1	1	0	

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

Minimization

■ The expression can be simplified in one of two ways:

- via algebra
- via Karnaugh maps

■ to  $A.\bar{C} + B.\bar{C}$

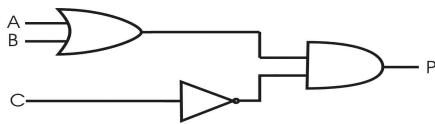
■ as the following truth table shows:

Truth table shows the same result

A	B	C	P	$A.\bar{C}$	$B.\bar{C}$	$A.\bar{C} + B.\bar{C}$
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	1	0	1	1
0	1	1	0	0	0	0
1	0	0	1	1	0	1
1	0	1	0	0	0	0
1	1	0	1	1	1	1
1	1	1	0	0	0	0

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

Means fewer logic gates are required



## Minterms and Maxterms

Notice the truth table has all possible combinations of A,B and C included:

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

The minterm is obtained from the "product" of A,B and C by AND-ing them

$$\bar{A}.B.C$$

The maxterm is obtained from the "sum" of A,B and C by OR-ing them and inverting inputs

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

## Sum of Products/Product of Sums

- For all combinations of inputs for which the output is a logical true:
  - Combining the minterms with OR gives the sum-of-products
- For all combinations of inputs for which the output is a logical false:
  - Combining the maxterms with AND gives the product-of-sums.

## From our example:

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

sum-of-products:

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

product-of-sums:

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

Normally the expression is derived using sum-of-products although product-of-sums yields fewer terms when there are more 1 outputs than 0 outputs.

## Exercise

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

Write out the sum-of-products expression for the truth table :

$$\underline{\quad} \cdot \underline{\quad} \cdot \underline{\quad} + \underline{\quad} \cdot \underline{\quad} \cdot \underline{\quad} + \underline{\quad} \cdot \underline{\quad} \cdot \underline{\quad} + \underline{\quad} \cdot \underline{\quad} \cdot \underline{\quad}$$

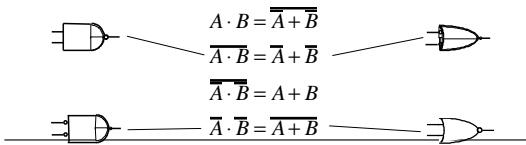
$$A \cdot B \cdot C + A \cdot B \cdot C + A \cdot B \cdot C + A \cdot B \cdot C$$

## Summary

- A circuits desired outputs can be specified in terms
- An boolean (logical) expression can be derived from the truth table.
- The boolean expression can then be simplified now we see how...

## Algebraic Laws

- DeMorgan's Laws
  - The AND and OR functions can be shown to be related to each other through the following equations:



## DeMorgan

- DeMorgan's Laws
  - Example: Implement the expression  $A \cdot B + C \cdot D$  using only NAND gates
    - NOT the individual terms
    - Change the sign
    - NOT the lot

## Boolean Algebraic Laws

Tautology (Idempotent)	$A \cdot A = A$ $A + A = A$
Complementary	$A \cdot \bar{A} = 0$ $A + \bar{A} = 1$
Operating with logic 0 and logic 1	$A \cdot 0 = 0$ $A \cdot 1 = A$ $A + 0 = A$ $A + 1 = 1$
Commutative	$A \cdot B = B \cdot A$ $A + B = B + A$
Associative	$(A \cdot B) \cdot C = A \cdot (B \cdot C) = A \cdot (B \cdot C)$
Distributive	$A \cdot (B + C) = A \cdot B + A \cdot C$ $A + (B \cdot C) = (A + B) \cdot (A + C)$

## Basic rules of Boolean Algebra

- Example: Simplify the following Expression

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

$$= A$$

$$A \cdot B + A \cdot C + A \cdot \bar{C} + A \cdot \bar{B} \quad \text{distributive}$$

$$A \cdot (B + \bar{B}) + A \cdot (C + \bar{C}) \quad \text{re-distribute}$$

$$A \cdot 1 + A \cdot 1 \quad \text{complementary}$$

$$A + A \quad \text{op with logic 1}$$

$$A \quad \text{idempotent}$$

## Exercises

- You should be able to:
  - Construct truth tables given boolean expressions
  - Compare expressions using truth tables
  - Produce a sum-of-products form from a truth table by combining minterms
  - Simplify the resulting expression algebraically
  - Represent the expression as a circuit using logic gates