Ministry of high Education and Scientific Research
Southern Technical University
Technological institute of Basra
Department of Electronic Techniques

Learning package

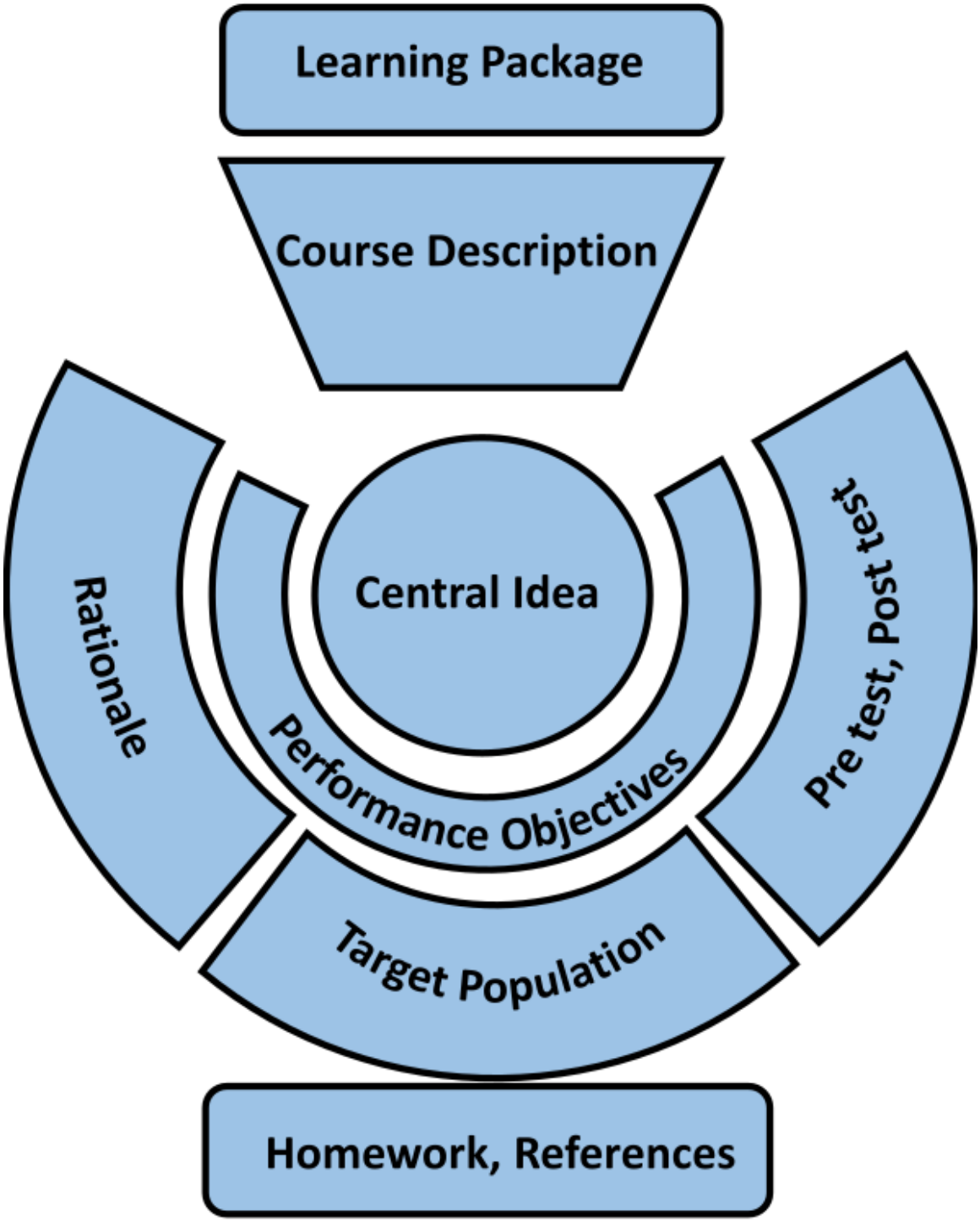
Digital circuits 1 +2

For

First year students

By

Dr. Mahmood A. Al-Shareeda
Lecturer
Dep. Of Electronic Techniques
2025


Learning Package
Course Description
Rationale
Central Idea
Performance Objectives
Pre test, Post test
Target Population
Homework, References

نموذج وصف المقرر **Course**

1- اسم المقرر:
الدوائر الرقمية1
2- رمز المقرر:
E112
3- الفصل / السنة:
فصلي
4- تاريخ إعداد هذا الوصف:
14/ 02/ 2024
5- أشكال الحضور المتاحة:
حضوري فقط
6- عدد الساعات الدراسية (الكلي)/ عدد الوحدات (الكلي):
وحدات604 ساعة فصليا / 4 ساعة اسبوعياً/
7- اسم مسؤول المقرر الدراسي (اذا اكثر من اسم يذكر)
الاسم: م.د محمود عارف لفته Mahmood.alshareedah@stu.edu.iq
8- اهداف المقرر

1. الرقمية: تمكين الطلاب من فهم تطوير الفهم الأساسي للإلكترونيات المبادئ الأساسية للدوائر الرقمية، بما في ذلك المكونات الرقمية الأساسية مثل البوابات، الادارات، شفت رجستر. 2. تطبيق المفاهيم النظرية: تعزيز القدرة على تطبيق المفاهيم النظرية في تصميم وتحليل الدوائر الإلكترونية.	• ..... • ..... • .....

3. تنمية المهارات العملية: توفير التدريب العملي من خلال التجارب المختبرية، مما يمكن الطلاب من اكتساب المهارات اللازمة لبناء واختبار الدوائر الإلكترونية. 4. تكنولوجيا المعلومات والاتصالات: فهم دور الإلكترونيات في تكنولوجيا المعلومات والاتصالات وتطبيقاتها العملية. 5. تعزيز التفكير النقدي: تشجيع الطلاب على التفكير النقدي والتحليلي في حل المشكلات المتعلقة بالإلكترونيات. 6. تحضير الطلاب لسوق العمل: إعداد الطلاب لدخول سوق العمل من خلال توفير المعرفة والمهارات المطلوبة في مجال الإلكترونيات. 7. التوجيه نحو التعلم المستمر: تحفيز الطلاب على متابعة التعلم الذاتي وتطوير مهاراتهم في مجال الإلكترونيات.	

9- استراتيجيات التعليم والتعلم

الاستراتيجية	1- استراتيجية التعليم تخطيط المفهوم التعاوني. 2- استراتيجية التعليم العصف الذهني. 3- استراتيجية التعليم سلسلة الملاحظات

10- بنية المقرر

الأسبوع	الساعات	مخرجات التعلم المطلوبة	اسم الوحدة او الموضوع	طريقة التعلم	طريقة التقييم
1	ساعة4	1- فهم تطبيقات الإلكترونيات 2-تطوير مهارات التفكير النقدي وحل المشكلات من	الأنظمة العددية	القيام بتجارب مختبرية لبناء واختبار الدوائر الرقمية. هذا يعزز الفهم النظري واكتساب المهارات العملية الطلب التغذية الراجعة من المعلمين والزملاء	الامتحانات الأسبوعية والشهرية واليومية والتحرير
2	ساعة4		النظام الثنائي		
3	ساعة4		النظام العشري		
4	ساعة4		نظام السادس عشر		
5	ساعة4		تحويل من الثنائي الى العشري		
6	ساعة4		البوابات المنطقية		

7	ساعة4	خلال تحليل الدوائر واكتشاف الأخطاء. 3-القدرة على استخدام أدوات المختبر الإلكترونية، مثل الملتيمتر، ومولدات الإشارة، وأجهزة الأوسيلوسكوب. 4-تحليل الدوائر الرقمية	تمثيل البوابات المنطقية الجبر البوليني نظريتا دي موركان خارطة كارنو رطة كارنو لثلاثة متغيرات رطة كارنو لاربع متغيرات المقارن الرقمي المقارن ذي المرتبتين. مفك الشفرات الترميز لترميز العشري الى الثنائي	ـديد نقـاط القوة والضعف مراجعـة المفـاهيم كل دوري وتطبيقها مسائل جديدة لتعزيز الذاكرة والفهم. ســتخدام البرامج ليميـة والتطبيقـات اعلية لفهم المفـاهيم كل أفضـــل، مثل محاكاة الدوائر شجيع البحث الذاتي مواضـيع جديدة في كـتـرونـيـات تكشـاف التطورات الحديثة	يـــــة وامتحـان نـهـايـة الفصل.
8	ساعة4				
9	ساعة4				
10	ساعة4				
11	ساعة4				
12	ساعة4				
13	ساعة4				
14	ساعة4				
15	ساعة4				

11- تقييم المقرر

توزيع كالتالي: 20 درجة امتحانات نظرية لمنتصف الفصل الاول. 20 درجة امتحانات عملية لمنتصف الفصل الاول.10 درجات امتحانات يومية و تقييم مستمر . 50 درجة امتحان نهاية الفصل

12- مصادر التعلم والتدريس

الكتب المقررة المطلوبة (المنهجية أن وجدت)	Holdsworth, Brian, and Clive Woods. Digital logic design. Elsevier, 2002. الالكترونيك الرقمي المتقدم ((ترجمة ضـــياء مهدي –محمود شكر-ياسر خليل))
المراجع الرئيسة (المصادر)	Alam, Mansaf, and Bashir Alam. *Digital Logic Design*. PHI Learning Pvt. Ltd., 2015.

الكتب والمراجع الساندة التي يوصى بها (المجلات العلمية، التقارير....)	Dally, William James, and R. Cu Harting. *Digital design: a systems approa* Cambridge University Press, 2012.
المراجع الإلكترونية ، مواقع الانترنيت	https://zlibrary-asia.se/ https://www.researchgate.net/

موذج وصف المقرر

1- اسم المقرر:
الدوائر الرقمية2
2- رمز المقرر:
E123
3- الفصل / السنة:
فصلي
4-تاريخ إعداد هذا الوصف:
14/ 02/ 2024
5-أشكال الحضور المتاحة:
حضوري فقط
6- عدد الساعات الدراسية (الكلي)/ عدد الوحدات (الكلي):

وحدات604 ساعة فصليا / 4 ساعة اسبوعياً/
7- اسم مسؤول المقرر الدراسي (اذا اكثر من اسم يذكر)
الاسم: م.د محمود عارف لفته Mahmood.alshareedah@stu.edu.iq

8- اهداف المقرر

1. الرقمية: تمكين الطلاب من فهم تطوير الفهم الأساسي للإلكترونيات المبادئ الأساسية للدوائر الرقمية، بما في ذلك المكونات الرقمية الأساسية مثل البوابات، الادارات، شفت رجستر. 2. تطبيق المفاهيم النظرية: تعزيز القدرة على تطبيق المفاهيم النظرية في تصميم وتحليل الدوائر الإلكترونية. 3. تنمية المهارات العملية: توفير التدريب العملي من خلال التجارب المختبرية، مما يمكن الطلاب من اكتساب المهارات اللازمة لبناء واختبار الدوائر الإلكترونية. 4. تكنولوجيا المعلومات والاتصالات: فهم دور الإلكترونيات في تكنولوجيا المعلومات والاتصالات وتطبيقاتها العملية. 5. تعزيز التفكير النقدي: تشجيع الطلاب على التفكير النقدي والتحليلي في حل المشكلات المتعلقة بالإلكترونيات. 6. تحضير الطلاب لسوق العمل: إعداد الطلاب لدخول سوق العمل من خلال توفير المعرفة والمهارات المطلوبة في مجال الإلكترونيات. 7. التوجيه نحو التعلم المستمر: تحفيز الطلاب على متابعة التعلم الذاتي وتطوير مهاراتهم في مجال الإلكترونيات.	• ..... • ..... • .....

9- استراتيجيات التعليم والتعلم

الاستراتيجية	1- استراتيجية التعليم تخطيط المفهوم التعاوني.

2- استراتيجية التعليم العصف الذهني. 3- استراتيجية التعليم سلسلة الملاحظات	

10- بنية المقرر

الأسبوع	الساعات	مخرجات التعلم المطلوبة	اسم الوحدة او الموضوع	طريقة التعلم	طريقة التقييم
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	ساعة4 ساعة4 ساعة4 ساعة4 ساعة4 ساعة4 ساعة4 ساعة4 ساعة4 ساعة4 ساعة4 ساعة4 ساعة4 ساعة4 ساعة4	1- فهم تطبيقات الإلكترونيات 2-تطوير مهارات التفكير النقدي وحل المشكلات من خلال تحليل الدوائر واكتشاف الأخطاء. 3-القدرة على استخدام أدوات المختبر الإلكترونية، مثل الملتيمتر، ومولدات الإشارة، وأجهزة الأوسيلوسكوب.	دائرة نصف الجامع دائرة نصف الطارح دائرة الجامع التام دائرة الطارح التام المراجيح العدادات العداد التموجي العداد التزامني التوالي العداد التزامني المتوازي سجلات الازاحه دوائر الذاكرة حويل من رقمي الى نظيري حويل من نظيري الى رقمي بطريقة العداد ADC التصاعدي باستخدام المعداد ADC تصاعدي تنازلي محول الفولتية الى تردد	القيام بتجارب عملية لبناء واختبار الدوائر الرقمية. هذا يز الفهم النظري سب المهارات العملية طلب التغذية الراجعة المعلمين والزملاء حديد نقاط القوة والضعف مراجعة المفاهيم كل دوري وتطبيقها مسائل جديدة لتعزيز الذاكرة والفهم. ستخدام البرامج ليمية والتطبيقات اعلية لفهم المفاهيم كل أفضل، مثل محاكاة الدوائر شجيع البحث الذاتي مواضيع جديدة في كترونيات تكشاف التطورات الحديثة	الامتحانات الأسبوعية والشهرية واليومية والتحريرية وامتحان نهاية الفصل.

			4-تحليل الدوائر الرقمية		

11- تقييم المقرر
توزيع كالتالي: 20 درجة امتحانات نظرية لمنتصف الفصل الاول. 20 درجة امتحانات عملية لمنتصف الفصل الاول10. درجات امتحانات يومية و تقييم مستمر . 50 درجة امتحان نهاية الفصل
12- مصادر التعلم والتدريس

الكتب المقررة المطلوبة (المنهجية أن وجدت)	Holdsworth, Brian, and Clive Woods. Digital logic design. Elsevier, 2002. الالكترونيك الرقمي المتقدم ((ترجمة ضياء مهدي –محمود شكر-ياسر خليل))
المراجع الرئيسة (المصادر)	Alam, Mansaf, and Bashir Alam. *Digital Logic Design*. PHI Learning Pvt. Ltd., 2015.
الكتب والمراجع الساندة التي يوصى بها (المجلات العلمية، التقارير....)	Dally, William James, and R. Cu Harting. *Digital design: a systems approa* Cambridge University Press, 2012.
المراجع الإلكترونية ، مواقع الانترنيت	https://zlibrary-asia.se/ https://www.researchgate.net/

Topic 1: Number Systems

1 / A – Target Population :-

This topic is intended for first-year students enrolled in the Department of Electronic Techniques at the Technological Institute of Basra, Southern Technical University. These students are beginning their journey into digital electronics and require foundational knowledge to understand how computers and digital systems process information numerically.

1 / B – Rationale :-

Number systems form the backbone of digital logic. All digital electronics—from microprocessors to programmable circuits—operate using binary and other number systems such as octal and hexadecimal. Students must be proficient in understanding and converting between these systems to analyze digital signals, design circuits, and interpret processor-level operations. Without this foundational skill, further study in logic design, coding, and digital communication becomes ineffective.

This unit prepares learners to:

Understand how numerical values are stored, transmitted, and manipulated in digital devices.

Connect abstract concepts of base systems to practical hardware applications.

Develop fluency in binary, octal, decimal, and hexadecimal conversions, enabling ease of transition to logic gate and memory unit topics later in the course.

1 / C – Central Idea :-

This module focuses on:

Types of Number Systems: Decimal (Base-10), Binary (Base-2), Octal (Base-8), Hexadecimal (Base-16)

Representation: How numbers are expressed in each system using place value.

Conversions:

Between any base to decimal

Decimal to binary, octal, and hexadecimal

Binary to octal/hex and vice versa

Codes:

Binary-Coded Decimal (BCD)

Excess-3 Code

Gray Code

These concepts build critical skills for digital circuit analysis and programming logic operations.

1 / D – Performance Objectives :-

By the end of this unit, the student will be able to:

Knowledge

Identify and describe different number systems and their characteristics.

Recognize binary equivalents of decimal numbers and vice versa.

Skills

Perform accurate conversions between decimal, binary, octal, and hexadecimal systems.

Represent numbers using BCD, Gray, and Excess-3 codes.

Competence

Analyze binary-coded data in practical applications such as digital counters and microcontrollers.

Interpret number formats used in computer architecture, digital displays, and logic circuits.

Chapter one
Numeral systems

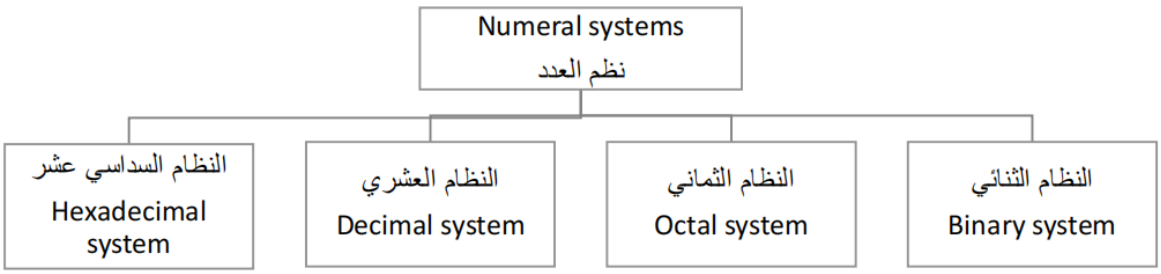
نظام الأعداد هو مجموعة من الرموز أو الحروف المستخدمة لتمثيل الأرقام والأعداد. يحدد النظام كيفية ترتيب الرموز واستخدامها لتعبير عن القيم العددية. يستخدم نظام الأعداد في مختلف المجالات مثل الرياضيات، الحوسبة، البرمجة، والهندسة لتمثيل الأرقام وتسهيل العمليات الحسابية والتحليلية.

Types of Number Systems: أنواع نظم الأعداد

There are four types of number systems commonly used in mathematics and computing. They are: هناك أربعة أنواع من نظم الأعداد التي تُستخدم بشكل شائع في الرياضيات والحوسبة. وهي:

1. **Decimal Number System**
   نظام الأعداد العشرية
2. **Binary Number System**
   نظام الأعداد الثنائية
3. **Octal Number System**
   نظام الأعداد الثمانية

4. **Hexadecimal Number System**
   نظام الأعداد الست عشرية



DECIMAL NUMBER SYSTEM:- نظام الأعداد العشرية

- يحتوي نظام الأعداد العشرية على عشرة رموز فريدة هي: 0، 1، 2، 3، 4، 5، 6، 7، 8، 9.
- في النظام العشري، يتم استخدام 10 رموز، لذلك فإن القاعدة أو الأساس هو 10.
- هو نظام ذو وزن موضعي.
- القيمة المرتبطة بالرمز تعتمد على موقعه بالنسبة للفاصلة العشرية.

الاستخدامات:

- يُستخدم في جميع الأنشطة الحياتية اليومية مثل العد، القياس، المعاملات المالية، وغيرها.
- النظام الأساسي في الرياضيات والإحصاء.

أساس النظام	10									
رموز النظام	0	1	2	3	4	5	6	7	8	9
مراتب النظام	10^0	10^1	10^2	10^3	10^4	10^5	10^6	10^7	10^8	10^9
	1	10	100	1000	10000	....	....	....	....	....

نظام الأعداد الثنائية Binary Number System

- نظام الأعداد الثنائية هو نظام ذو وزن موضعي.
- القاعدة أو الأساس في هذا النظام هو 2.
- يحتوي على رمزين مستقلين.
- الرموز المستخدمة هي 0 و 1.
- الرقم الثنائي يُسمى "بت.(bit) "
- الفاصلة الثنائية تفصل بين الجزء الصحيح والجزء الكسري .

الاستخدامات:

- يُستخدم في جميع الأجهزة الرقمية مثل الكمبيوترات، الهواتف الذكية، وأي جهاز يستخدم الدوائر الإلكترونية.
- يُستخدم لتمثيل البيانات، العمليات الحسابية، وتخزين المعلومات في الكمبيوتر.

أساس النظام	2	
رموز النظام	0	1
مراتب النظام	2^0	2^1
	1	2

نظام الأعداد الثمانية Octal Number System

- هو أيضًا نظام ذو وزن موضعي.
- القاعدة أو الأساس في هذا النظام هو 8.
- يحتوي على 8 رموز مستقلة: 0، 1، 2، 3، 4، 5، 6، و 7.
- قاعدته $8 = 2^3$، كل مجموعة من 3 بتات في النظام الثنائي يمكن تمثيلها برمز في النظام الثماني.

الاستخدامات:

- يُستخدم في البرمجة كاختصار لتمثيل الأعداد الثنائية .

•يُستخدم في أنظمة التشغيل القديمة مثل UNIX لتمثيل أذونات الملفات.

• يُستخدم في تمثيل الأرقام الثنائية بشكل أكثر اختصارًا وكفاءة.

• يُستخدم في بعض الأجهزة الرقمية لتسهيل التعامل مع البيانات الثنائية .

•يُستخدم في بعض تطبيقات الشبكات والاتصالات لتمثيل المعلومات بشكل أكثر تنظيمًا.

أساس النظام	8							
رموز النظام	0	1	2	3	4	5	6	7
مراتب النظام	8^0	8^1	8^2	8^3	8^4	8^5	8^6	8^7
	1	8	64	512	4096	....	....	....

نظام الأعداد الست عشرية Hexadecimal Number System

- نظام الأعداد الست عشرية هو نظام ذو وزن موضعي.
- القاعدة أو الأساس في هذا النظام هو 16.
- الرموز المستخدمة هي 0، 1، 2، 3، 4، 5، 6، 7، 8، 9، A، B، C، D، E وF.
- القاعدة 16 = 2^4، كل مجموعة مكونة من 4 بتات في النظام الثنائي يمكن تمثيلها برمز في النظام الست عشري.

أساس النظام	16															
رموز النظام	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
مراتب النظام	16^0	16^1	16^2	16^3	16^4	16^5	16^6	16^7	16^8	16^9	16^A	16^B	16^C	16^D	16^E	16^F
	1	16	256	..	..	..	..	..	..	..	..	..	..	..	..	..

الاستخدامات:

- يُستخدم في البرمجة والأنظمة الرقمية لتمثيل الأعداد بطريقة أكثر اختصارًا من النظام الثنائي.
- يستخدم بشكل واسع في الحوسبة لتحديد العناوين في الذاكرة والأوامر.
- يُستخدم في تمثيل الألوان في تصميم الويب (مثل اللون .(#FF5733
- يُستخدم في تمثيل الأعداد في بعض العمليات الحسابية المعقدة في البرمجة.

مقارنة بين أنظمة الأعداد العشرية، الثنائية، الثمانية، والست عشرية

تُستخدم أنظمة الأعداد المختلفة في مجالات متعددة مثل الرياضيات، الحوسبة، والبرمجة. لكل نظام خصائص فريدة تجعله مناسبًا لاستخدامات معينة. في هذا الجدول، نعرض مقارنة بين أربعة من أشهر أنظمة الأعداد: العشرية، الثنائية، الثمانية، والست عشرية. يسلط الجدول الضوء على الفروق بين هذه الأنظمة من حيث الاستخدام، التطبيقات العملية، الكفاءة في الأجهزة، والحداثة.

النظام العددي	القاعدة	الرموز المستخدمة	الاستخدامات الرئيسية	التطبيقات	الأجهزة	الحداثة والتطور
نظام الأعداد العشرية	10	0، 1، 2، 3، 4، 5، 6، 7، 8، 9	العد اليومي، القياس، المعاملات المالية، الرياضيات، الإحصاء	العمليات الحسابية اليومية، المعاملات المالية، القياسات العلمية	جميع الأجهزة اليومية (حاسوب، آلة حاسبة، إلخ)	النظام الأكثر شيوعًا في الحياة اليومية، الأساس في الرياضيات
نظام الأعداد الثنائية	2	0، 1	تمثيل البيانات في الأجهزة الرقمية، عمليات الحوسبة، البرمجة	الحوسبة، شبكات الكمبيوتر، الخوارزميات، التشفير	جميع الأجهزة الإلكترونية الحديثة (كمبيوترات، هواتف، أجهزة ذكية)	الأساس في تصميم الكمبيوتر والأجهزة الرقمية، يعتمد عليه بشكل كامل
نظام الأعداد الثمانية	8	0، 1، 2، 3، 4، 5، 6، 7	اختصار لتمثيل الأعداد الثنائية، في البرمجة القديمة، أنظمة التشغيل	تمثيل الأعداد الثنائية بشكل أبسط، الأذونات في UNIX	الأجهزة القديمة والأنظمة التي تعتمد على التمثيل الثنائي	يُستخدم بشكل أقل في الأنظمة الحديثة، لكن مفيد في البرمجة القديمة
نظام الأعداد الست عشرية	16	0، 1، 2، 3، 4، 5، 6، 7، 8، 9، A، B، C، D، E، F	اختصار للأعداد الثنائية، البرمجة، تحديد الألوان في الويب، العناوين في الذاكرة	البرمجة الحديثة، الحوسبة، الشبكات، تمثيل الألوان في HTML	أجهزة الكمبيوتر الحديثة، أجهزة الاتصالات الرقمية	الأكثر استخدامًا في البرمجة الحديثة، الأجهزة الرقمية، تصميم الويب

ملاحظات إضافية:

- **النظام العشري** :الأكثر شيوعًا في الحياة اليومية ويعتمد عليه الناس بشكل رئيسي في جميع الأنشطة الحياتية.
- **النظام الثنائي** :يُستخدم في جميع الأجهزة الرقمية والكمبيوترات لأنه يمثل البيانات في شكل منطقي بسيط (0 و 1).

- **النظام الثماني** :يُستخدم في بعض البرمجيات القديمة مثل أنظمة UNIX لتبسيط التعامل مع الأعداد الثنائية.
- **النظام الست عشري** :أصبح معيارًا شائعًا في البرمجة الحديثة والأنظمة الرقمية نظرًا لقدرته على تمثيل الأعداد الثنائية بشكل مضغوط وواضح.

التحويل من نظام عددي إلى آخر

لماذا نحتاج إلى تحويل الأعداد من نظام عددي إلى آخر؟

تحويل الأعداد بين الأنظمة العددية المختلفة يعد من المهارات الأساسية في العديد من المجالات، خاصة في **الرياضيات** و**الهندسة** و**علوم الكمبيوتر** .هناك عدة أسباب تجعل من المهم فهم كيفية تحويل الأعداد من نظام عددي إلى آخر:

التعامل مع البيانات في الحوسبة:

- الأنظمة الثنائية والست عشرية تُستخدم في الحواسيب لأنها تمثل البيانات الرقمية بفعالية.
- التحويل بين الأنظمة يُسهل فهم البيانات، حيث يُستخدم الست عشري كاختصار للثنائي.

البرمجة والتطوير البرمجي:

- تحويل الأعداد يُستخدم لتمثيل العناوين في الذاكرة والألوان في البرمجة.
- المبرمجون في البرمجة منخفضة المستوى يعتمدون على الأنظمة الثنائية والست عشرية.

التكامل بين الأجهزة والبرمجيات:

- البيانات يتم تمثيلها بالثنائي بين الأجهزة، لكن البرمجيات تستخدم أحيانًا الأنظمة العشرية أو الست عشرية لتسهيل التفاعل.
- مثل تحويل عناوين IP من النظام العشري إلى الثنائي في الشبكات.

الحسابات الرياضية والتمثيل المكاني:

- بعض العمليات الحسابية تتطلب استخدام النظام العشري أو الثنائي.
- النظام الثماني يُستخدم أحيانًا في التطبيقات القديمة أو لتمثيل البيانات بشكل مختصر.

التعامل مع الأنظمة القديمة أو المشروحة:

- الأنظمة القديمة قد تستخدم النظام الثماني أو الثنائي، لذلك يحتاج المحللون لتحويل الأعداد لفهم البيانات.

الفهم العميق للبيانات:

- فهم التحويلات بين الأنظمة يُساعد في تحليل البيانات وكيفية تخزينها في الذاكرة.
- التحويلات تُساعد العلماء والمبرمجين في تحسين الأداء والكفاءة في الحوسبة.

في النهاية:

إذا لم نتمكن من تحويل الأعداد بين الأنظمة العددية المختلفة، سنواجه صعوبة في التعامل مع البيانات المتنوعة وتطبيقاتها في مختلف المجالات مثل البرمجة، الحوسبة، شبكات الكمبيوتر، والهندسة. لذلك، تعلم كيفية تحويل الأعداد بين الأنظمة المختلفة ليس فقط يساعد في تنفيذ العمليات الحسابية، بل أيضًا يمكننا من استخدام الأدوات التكنولوجية بشكل أكثر فعالية.

التحويل بين الأنظمة العددية المختلفة وأسباب التحويل

نوع التحويل	مثال	سبب التحويل
من النظام العشري إلى الثنائي	العشري 13 → الثنائي 1101	**التمثيل في الأجهزة الرقمية**: الحواسيب والمعالجات تعمل على أساس النظام الثنائي (0 و 1). التحويل ضروري لفهم كيفية تخزين البيانات ومعالجتها في الأجهزة الإلكترونية.
من النظام الثنائي إلى العشري	الثنائي 1101 → العشري 13	**فهم البيانات**: عند التعامل مع البيانات الثنائية في البرمجة أو داخل الأنظمة الرقمية، قد يكون من الأسهل تفسيرها في النظام العشري لاستخدامها في الحسابات اليومية أو التطبيقات الرياضية.
من النظام العشري إلى الثماني	العشري 65 → الثماني 101	**التقليل من الحجم**: النظام الثماني كان يُستخدم في الماضي لتمثيل البيانات الثنائية بشكل مختصر، وهو مفيد عند التعامل مع الأجهزة أو الأنظمة التي تعتمد على الثنائي.
من النظام الثماني إلى العشري	الثماني 101 → العشري 65	**فهم البيانات**: لتحويل الأرقام من النظام الثماني إلى العشري يساعد في التعامل مع القيم بشكل أكثر دقة، خاصة عند البرمجة أو تحليل البيانات في التطبيقات الحسابية.
من النظام العشري إلى الست عشري	العشري 254 → الست عشري FE	**التمثيل الفعّال في البرمجة**: النظام الست عشري يُستخدم بشكل شائع في البرمجة، خاصة لتمثيل الذاكرة والعناوين (مثل الأكواد اللونية في البرمجة). التحويل يسهل التعامل مع البيانات الكبيرة بكفاءة.
من النظام الست عشري إلى العشري	الست عشري FE → العشري 254	**تحليل البيانات**: تحويل الأرقام الست عشرية إلى العشرية يساعد على تفسيرها بسهولة في التطبيقات الحسابية والعلمية، حيث تكون القيم الست عشرية عادة جزءًا من تمثيلات الذاكرة أو العناوين.

سبب الحاجة للتحويل بين الأنظمة العددية:

- **الأنظمة الرقمية** :الحواسيب والأجهزة الرقمية تستخدم النظام الثنائي لتمثيل البيانات داخليًا، لكن البرمجة والأنظمة الأخرى تعتمد على الأنظمة العشرية أو الست عشرية لتوفير سهولة الفهم والتفاعل.

- **اختصار البيانات** :الأنظمة مثل **الست عشري** و **الثماني** تُستخدم لتقليص حجم البيانات المُعالجة أو المخزنة، مما يسهل العمل مع الأرقام الكبيرة بطريقة أكثر فاعلية.
- **التفاعل بين الأجهزة** :بعض الأنظمة القديمة أو الأجهزة قد تتطلب تحويلات بين الأنظمة العددية المختلفة لضمان التوافق بين الأجهزة أو التطبيقات المختلفة.

Binary Number System

التحويل من النظام الثنائي إلى النظام العشري

في هذه الطريقة، يتم تحويل العدد الثنائي إلى عشري عن طريق ضرب كل رقم ثنائي في **وزنه الموضعي** (أي 2 مرفوعة إلى القوة المناسبة بناءً على موقع الرقم في العدد) ثم جمع النتائج.

For example:

(i) Convert $(10101)_2$ to decimal.

Solution :

(Positional weight) $2^4\ 2^3\ 2^2\ 2^1\ 2^0$

Binary number 10101

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

$= 16 + 0 + 4 + 0 + 1$

$= (21)_{10}$

(ii) Convert $(111.101)_2$ to decimal.

Solution:

$(111.101)_2 = (1 \times 2^2) + (1 \times 2^1) + (1 \times 2^0) + (1 \times 2^{-1}) + (0 \times 2^{-2}) + (1 \times 2^{-3})$

$= 4 + 2 + 1 + 0.5 + 0 + 0.125$

$= (7.625)_{10}$

التحويل من النظام الثنائي إلى النظام الثماني

لتحويل الأعداد من النظام الثنائي إلى النظام الثماني، يتم تقسيم الأعداد الثنائية إلى مجموعات من 3 بتات لكل مجموعة، بدءًا من نقطة الفاصلة الثنائية والتوجه نحو اليسار واليمين.

النظام الثماني	النظام الثنائي
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

For example:

(i) Convert $(101111010110.110110011)_2$ into octal.

Solution :

Group of 3 bits are	101	111	010	110	.	110	110	011
Convert each group into octal =	5	7	2	6	.	6	6	3

The result is $(5726.663)_8$

(ii) Convert $(10101111001.0111)_2$ into octal.

Solution :

Binary number		10	101	111	001	.	011	1
Group of 3 bits are	=	010	101	111	001	.	011	100
Convert each group into octal =		2	5	7	1	.	3	4

The result is $(2571.34)_8$

التحويل من النظام الثنائي إلى النظام الست عشري:

لتحويل الأعداد من النظام الثنائي إلى النظام الست عشري، يتم تقسيم الأعداد الثنائية إلى مجموعات مكونة من 4 بتات لكل مجموعة، بدءًا من نقطة الفاصلة الثنائية وعلى كلا الجانبين (اليسار واليمين).

النظام الست عشري	النظام الثنائي
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

For example:
(i) Convert $(1011011011)_2$ into hexadecimal.

Solution:

Given Binary number		10	1101	1011
Group of 4 bits are		0010	1101	1011
Convert each group into hex	=	2	D	B

The result is $(\mathbf{2DB})_{16}$

(ii) Convert $(01011111011.011111)_2$ into hexadecimal.

Solution:

Given Binary number		010	1111	1011	.	0111	11
Group of 3 bits are	=	0010	1111	1011	.	0111	1100
Convert each group into octal	=	2	F	B	.	7	C

The result is $(\mathbf{2FB.7C})_{16}$

DECIMAL NUMBER SYSTEM

نظام الأعداد العشري

التحويل من النظام العشري إلى النظام الثنائي:

الطريقة :في التحويل من النظام العشري إلى النظام الثنائي، يتم تحويل الأعداد الصحيحة إلى النظام الثنائي باستخدام **القسمة المتكررة** على **القاعدة (الأساس)** التي هي 2 (لأننا ننتقل إلى النظام الثنائي).

قسّم العدد العشري المعطى بشكل متتابع على 2، ثم اقرأ الجزء الصحيح وباقي القسمة من الأسفل إلى الأعلى للحصول على العدد الثنائي المكافئ. بالنسبة للجزء الكسري، اضربه في 2. احتفظ بالعدد الصحيح في الناتج كما هو، ثم اضرب الجزء الكسري الجديد في الناتج في 2. تستمر هذه العملية ويتم قراءة الأعداد الصحيحة في النواتج من الأعلى إلى الأسفل.

For example:
(i) Convert $(52)_{10}$ into binary.

2 | 52
2 | 26 — 0
2 | 13 — 0
2 | 6 — 1
2 | 3 — 0
2 | 1 — 1
0 — 1

Result of $(52)_{10}$ is **$(110100)_2$**

(ii) Convert $(105.15)_{10}$ into binary.

Solution:

Integer part		**Fraction part**
2 | 105		0.15 x 2 = 0.30
2 | 52	— 1	0.30 x 2 = 0.60
2 | 26	— 0	0.60 x 2 = 1.20
2 | 13	— 0	0.20 x 2 = 0.40
2 | 6	— 1	0.40 x 2 = 0.80
2 | 3	— 0	0.80 x 2 = 1.60
2 | 1	— 1	
0	— 1	

Result of $(105.15)_{10}$ is **$(1101001.001001)_2$**

التحويل من النظام العشري إلى النظام الثماني:

لتحويل العدد الصحيح العشري المعطى إلى النظام الثماني، قسم العدد المعطى بشكل متتابع على 8 حتى يصبح الناتج 0. أما لتحويل الأعداد الكسرية العشرية إلى النظام الثماني، فاضرب الجزء الكسري من العدد العشري والأجزاء الكسرية التالية في 8 بشكل متتابع حتى يصبح الناتج 0 أو حتى تحصل على الدقة المطلوبة.

For example:
(i) Convert $(378.93)_{10}$ into octal.

Solution:

8 | 378		0.93 x 8 = 7.44
8 | 47	— 2	0.44 x 8 = 3.52
8 | 5	— 7	0.52 x 8 = 4.16
0	— 5	0.16 x 8 = 1.28

Result of $(378.93)_{10}$ is **$(572.7341)_8$**

التحويل من النظام العشري إلى النظام الست عشري:

التحويل من النظام العشري إلى النظام الست عشري مشابه تمامًا للتحويل إلى النظام الثماني، ولكن باستخدام القاعدة 16 بدلًا من 8.

For example:
(i) Convert $(2598.675)_{10}$ into hexadecimal.

Solution:

	Remainder Decimal	Hex		Hex
16 | 2598			0.675 x 16 = 10.8	A
16 | 162	— 6	6	0.800 x 16 = 12.8	C
16 | 10	— 2	2	0.800 x 16 = 12.8	C
0	— 10	A	0.800 x 16 = 12.8	C

Result of $(2598.675)_{10}$ is **$(A26.ACCC)_{16}$**

نظام الأعداد الثمانية(Octal Number System

التحويل من النظام الثماني إلى النظام الثنائي:

لتحويل عدد ثماني معطى إلى النظام الثنائي، استبدل كل رقم ثماني بنظيره الثنائي المكون من 3 بتات.

For example:

Convert $(367.52)_8$ into binary.

Solution:

Given Octal number is		3	6	7 .	5	2
Convert each group octal to binary	=	011	110	111 .	101	010

Result of $(367.52)_8$ is **$(011110111.101010)_2$**

التحويل من النظام الثماني إلى النظام العشري:

لتحويل عدد من النظام الثماني إلى النظام العشري، قم بضرب كل رقم في العدد الثماني في وزن موقعه (أي القوة المناسبة للأساس 8)، ثم اجمع جميع القيم الناتجة.

For example: -

Convert $(4057.06)_8$ to decimal

Solution:

$$(4057.06)_8 = 4 \times 8^3 + 0 \times 8^2 + 5 \times 8^1 + 7 \times 8^0 + 0 \times 8^{-1} + 6 \times 8^{-2}$$
$$= 2048 + 0 + 40 + 7 + 0 + 0.0937$$
$$= (2095.0937)_{10}$$

Result is **$(2095.0937)_{10}$**

التحويل من النظام الثماني إلى النظام الست عشري:

لتحويل العدد الثماني إلى النظام الست عشري، قم أولًا بتحويل العدد الثماني إلى النظام الثنائي، ثم قم بتحويل العدد الثنائي إلى النظام الست عشري.

For example :-

Convert $(756.603)_8$ to hexadecimal.

Solution :-

Given octal no.		7	5	6	.	6	0	3
Convert each octal digit to binary	=	111	101	110	.	110	000	011
Group of 4bits are	=	0001	1110	1110	.	1100	0001	1000
Convert 4 bits group to hex.	=	1	E	E	.	C	1	8

Result is $(\mathbf{1EE.C18})_{16}$

نظام الأعداد الست عشري(Hexadecimal Number System)

التحويل من النظام الست عشري إلى النظام الثنائي:

لتحويل العدد الست عشري إلى النظام الثنائي، استبدل كل رقم ست عشري بمجموعته الثنائية المكونة من 4 بتات.

For example:

Convert $(3A9E.B0D)_{16}$ into binary.

Solution:

Given Hexadecimal number is		3	A	9	E	.	B	0	D
Convert each hexadecimal digit to 4 bit binary	=	0011	1010	1001	1110	.	1011	0000	1101

Result of $(3A9E.B0D)_8$ is **$(0011101010011110.101100001101)_2$**

التحويل من النظام الست عشري إلى النظام العشري:

لتحويل العدد الست عشري إلى النظام العشري، قم بضرب كل رقم في العدد الست عشري في وزن موقعه (أي القوة المناسبة للأساس 16)، ثم اجمع جميع القيم الناتجة.

For example: -
Convert $(A0F9.0EB)_{16}$ to decimal

Solution:

$$(A0F9.0EB)_{16} = (10 \times 16^3)+(0 \times 16^2)+(15 \times 16^1)+(9 \times 16^0)+(0 \times 16^{-1})+(14 \times 16^{-2})+(11 \times 16^{-3})$$
$$= 40960 + 0 + 240 + 9 + 0 + 0.0546 + 0.0026$$
$$= (41209.0572)_{10}$$

التحويل من النظام الست عشري إلى النظام الثماني:

لتحويل العدد الست عشري إلى النظام الثماني، قم أولًا بتحويل العدد الست عشري إلى النظام الثنائي، ثم قم بتحويل العدد الثنائي إلى النظام الثماني.

For example :-
Convert $(B9F.AE)_{16}$ to octal.

Solution :-

Given hexadecimal no.is		B		9		F	.	A		E
Convert each hex. digit to binary	=	1011		1001		1111	.	1010		1110
Group of 3 bits are	=	101	110	011	111		.	101	011	100
Convert 3 bits group to octal.	=	5	6	3	7		.	5	3	4

Result is $(5637.534)_8$

1. Binary Number System

- التحويل من النظام الثنائي إلى النظام العشري
- التحويل من النظام الثنائي إلى النظام الثماني
- التحويل من النظام الثنائي إلى النظام الست عشري

2. Decimal Number System

- التحويل من النظام العشري إلى النظام الثنائي
- التحويل من النظام العشري إلى النظام الثماني
- التحويل من النظام العشري إلى النظام الست عشري

3. Octal Number System

- التحويل من النظام الثماني إلى النظام الثنائي
- التحويل من النظام الثماني إلى النظام العشري
- التحويل من النظام الثماني إلى النظام الست عشري

4. Hexadecimal Number System

- التحويل من النظام الست عشري إلى النظام الثنائي
- التحويل من النظام الست عشري إلى النظام العشري
- التحويل من النظام الست عشري إلى النظام الثماني

Post-Test

Answer the following:

Convert the decimal number 177 into its 8-bit binary equivalent (first convert to octal, then to binary).

Convert the decimal number 423 into its hexadecimal form.

Convert $(537)_8$ to decimal.

What is the Gray Code for decimal number 5?

Convert binary number $(10011101)_2$ to its octal equivalent.

Key Answers

177 → Octal: $(261)_8$ → Binary: 010110001

423 ÷ 16 = 26 R7; 26 ÷ 16 = 1 R10 (A); 1 ÷ 16 = 0 R1 → $(1A7)_{16}$

$(537)_8 = 5\times64 + 3\times8 + 7 = 320 + 24 + 7 = 351_{10}$

Decimal 5 → Gray Code: 0111

$(10011101)_2$ = 001 001 110 101 (grouped) = $(1175)_8$

Homework Assignments

Convert the octal number $(641)_8$ to decimal. (Answer: 369)

Convert $(146)_{10}$ to octal and then to binary. (Answer: 222_8 and 010010010_2)

Convert the binary number $(10011101)_2$ to octal. (Answer: 235_8)

Write the next three numbers in this octal counting sequence: 624, 625, 626, …

Convert the decimal number $(975)_{10}$ to binary by first converting to octal. (Answer: 1111001111)

Convert binary number $(1010111011)_2$ to decimal by first converting to octal. (Answer: 699)

Convert $(24CE)_{16}$ to decimal. (Answer: 9422)

Convert $(3117)_{10}$ to hexadecimal, then from hex to binary. (Answer: $C2D_{16}$ and 110000101101_2)

Convert $(1001011110110101)_2$ to hex. (Answer: $97B5_{16}$)

Write the next four numbers in this hexadecimal sequence: E9A, E9B, E9C, E9D, …

Convert $(3527)_8$ to hex. (Answer: 757_{16})

Topic 2: Logic Gates

1 / A – Target Population :-

This topic is designed for **first-year undergraduate students** enrolled in the **Department of Electronic Techniques** at the **Technological Institute of Basra**. These students are beginning their studies in digital electronics and require a solid understanding of logic gates, which serve as the essential building blocks for all digital systems.

1 / B – Rationale :-

Logic gates are fundamental to digital electronics. They are used to construct every digital device—from simple switches to advanced microprocessors. A thorough understanding of logic gates enables students to analyze, simulate, and design digital circuits that form the core of embedded systems, computers, and control devices.
This topic introduces learners to:

- **Digital decision-making** using binary inputs and outputs
- **Boolean logic** in a practical and visual context
- **Circuit design** using symbolic logic

It establishes the **intellectual framework** required for more advanced topics such as Boolean algebra, Karnaugh maps, memory units, and arithmetic logic units (ALUs).

1 / C – Central Idea :-

The topic revolves around the **operation, representation, and truth tables** of basic and compound logic gates. It covers:

1. **Basic Logic Gates**: AND, OR, NOT
2. **Compound and Universal Gates**: NAND, NOR, XOR, XNOR
3. **Truth Tables**: Demonstrating input-output behavior
4. **Symbolic and Boolean Representation**: Standard gate symbols and algebraic expressions
5. **Logic Diagrams**: Interpreting and constructing logic circuits
6. **Universal Functionality**: How NAND and NOR can replace all other gates

1 / D – Performance Objectives :-

Upon completing this unit, the student will be able to:

Knowledge

- Identify standard logic gates and their Boolean expressions
- Describe the behavior of each gate using truth tables

Skills

- Construct truth tables for multi-input logic gates
- Build simple circuits using AND, OR, NOT, and compound gates
- Represent logic gates using symbolic and Boolean formats

Competence

- Analyze digital logic conditions using gate diagrams
- Select appropriate gates to meet design criteria
- Implement logic functions using universal gates (NAND/NOR)

البوابات المنطقية LOGIC GATES

البوابات المنطقية (Logic Gates) هي الدوائر الإلكترونية التي تستخدم لتنفيذ العمليات المنطقية الأساسية على المدخلات الرقمية (0 و 1) في الأنظمة الرقمية. هذه البوابات هي المكونات الأساسية التي يعتمد عليها جميع الأجهزة الرقمية، من الكمبيوترات إلى الهواتف الذكية، لتنفيذ العمليات الحسابية، منطقية والتحكمية.

أهمية البوابات المنطقية:

1. **أساس الأنظمة الرقمية :**تعتبر البوابات المنطقية حجر الأساس لجميع الدوائر الرقمية. فهي تتحكم في جميع العمليات الحسابية والمنطقية في الأجهزة الإلكترونية مثل الكمبيوترات، الميكروكنترولر، والأجهزة المدمجة.
2. **التحكم والتشغيل :**من خلال البوابات المنطقية يمكن التحكم في تدفق الإشارات الرقمية داخل الأجهزة. البوابات مثل AND و OR و NOT تعمل على تحديد كيفية معالجة البيانات والإشارات.
3. **تحويل العمليات إلى دوائر كهربائية :**البوابات المنطقية تتحكم في تحويل البيانات المدخلة إلى مخرجات معينة من خلال عمليات منطقية، مما يسمح بتطوير أجهزة مثل المعالجات، الذاكرة، وأجهزة التحكم.
4. **تصميم الدوائر الرقمية :**من خلال فهم البوابات المنطقية، يمكن للطلاب تصميم وبناء دوائر رقمية معقدة، مثل العدادات، المسجلات، والأنظمة البسيطة باستخدام المعالجات الدقيقة.

أين يتم استخدام البوابات المنطقية؟

- **أنظمة الحوسبة :**في جميع المعالجات الدقيقة (الموبايلات، الكمبيوترات)، يتم استخدام البوابات المنطقية لتنفيذ العمليات الحسابية، والقرارات المنطقية.
- **الدوائر الرقمية :**مثل أنظمة التحكم في الآلات، الأجهزة الكهربائية، الأجهزة الرقمية في السيارات، وغيرها من الأنظمة التي تحتاج إلى منطق ثنائي.
- **الأنظمة المدمجة :(Embedded Systems)** تستخدم البوابات المنطقية في تصميم الدوائر المدمجة داخل الأجهزة مثل أجهزة التحكم، الأجهزة الطبية، والأدوات الذكية.
- **الاتصالات :**تُستخدم البوابات المنطقية في تحسين الاتصالات الرقمية من خلال تدفق البيانات عبر الشبكات.

أنواع مختلفة من البوابات المنطقية:

جدول الحقيقة (Truth Table) هو جدول يُستخدم لعرض كافة القيم الممكنة للمدخلات والمخرجات في الدوائر المنطقية. يعتمد على أنواع البوابات المنطقية مثلAND ، OR، NOT، وغيرها. يتم استخدام هذا الجدول لفهم كيفية عمل البوابات المنطقية في جميع الحالات الممكنة.

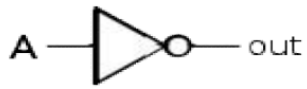
رموز المنطق:(Logic Symbols)

في الدوائر الرقمية، يتم استخدام **رموز** معينة لتمثيل **البوابات المنطقية** المختلفة. كل بوابة منطقية لها رمز خاص بها يُستخدم في الرسم التخطيطي للدوائر لتوضيح طريقة عملها.

بوابة) NOT العكسي:(

- بوابة **NOT**، والتي تُسمى أيضًا **العكسي**، تحتوي على مدخل واحد ومخرج واحد.
- هي جهاز يكون مخرجه دائمًا مكملًا للمدخل الخاص به.
- مخرج بوابة **NOT**يكون في حالة منطقية 1 عندما يكون المدخل في حالة منطقية 0، وفي حالة منطقية 0 عندما يكون المدخل في حالة منطقية 1.

Logic Symbol



Truth table

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

بوابة:AND

- **بوابة AND** تحتوي على مدخلين أو أكثر ولكن لها مخرج واحد فقط.
- **المخرج** يكون في حالة) **1صح** (فقط عندما يكون كل مدخل من المدخلات في حالة) **1صح**.(
- إذا كان **أحد المدخلات** في حالة) **0خطأ**(، فإن المخرج سيكون في حالة) **0خطأ** (بغض النظر عن حالة المدخل الآخر.

Logic Symbol

Truth Table

		OUTPUT
A	B	Q=A . B
0	0	0
0	1	0
1	0	0
1	1	1

بوابة:OR

- **بوابة OR** قد تحتوي على مدخلين أو أكثر ولكن لها مخرج واحد فقط.
- **المخرج** يكون في حالة) **1صح** (حتى لو كان أحد المدخلات فقط في حالة) **1صح**.(
- **المخرج** يكون في حالة) **0خطأ** (فقط عندما يكون كل المدخلات في حالة) **0خطأ**.(

Logic Symbol

Truth Table

INPUT		OUTPUT
A	B	Q=A + B
0	0	0
0	1	1
1	0	1
1	1	1

بوابة:NAND

- **بوابة NAND** هي مزيج من **بوابة AND** و **بوابة.NOT**
- **المخرج** يكون في حالة) **0خطأ** (عندما تكون جميع المدخلات في حالة) **1صح(**، ولأي مجموعة أخرى من المدخلات، يكون **المخرج** في حالة) **1صح.(**

Logic Symbol

Truth Table

INPUT		OUTPUT
A	B	Q= $\overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

بوابة:NOR

- **بوابة NOR** هي مزيج من **بوابة OR** و **بوابة.NOT**
- **المخرج** يكون في حالة) **1صح** (فقط عندما تكون جميع المدخلات في حالة) **0خطأ(**، ولأي مجموعة أخرى من المدخلات، يكون **المخرج** في حالة) **0خطأ.(**

Logic Symbol

Truth Table

INPUT		OUTPUT
A	B	Q= $\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

بوابة:EXCLUSIVE-OR (X-OR)

- **بوابة X-OR** هي دائرة منطقية ذات مدخلين ومخرج واحد.
- **المخرج** يكون في حالة) **1صح** (عندما يكون أحد المدخلات فقط في حالة) **1صح** .(وعندما يكون كلا المدخلين في حالة) **0خطأ** (أو عندما يكون كلا المدخلين في حالة) **1صح**(، يكون **المخرج** في حالة) **0خطأ**.(

Logic Symbol

INPUTS are **A** and **B**

OUTPUT is **Q** = A $\oplus$ B

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

Truth Table

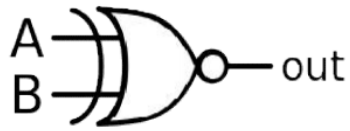
INPUT		OUTPUT
A	B	Q = A $\oplus$ B
0	0	0
0	1	1
1	0	1
1	1	0

بوابة:EXCLUSIVE-NOR (X-NOR)

- **بوابة X-NOR** هي مزيج من **بوابة X-OR** و **بوابة NOT**.

- **بوابة X-NOR** هي دائرة منطقية ذات مدخلين ومخرج واحد.
- **المخرج** يكون في حالة (**1صح** (فقط عندما يكون كلا المدخلين في نفس الحالة (إما كلاهما 0 أو كلاهما 1).
- **المخرج** يكون في حالة (**0خطأ** (عندما يكون أحد المدخلات في حالة (**0خطأ** (والآخر في حالة (**1صح**.(

Logic Symbol



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

$= A\ XNOR\ B$

INPUT		OUTPUT
A	B	OUT = A XNOR B
0	0	1
0	1	0
1	0	0
1	1	1

Post-Test

Answer the following:

1. Draw the logic symbol and truth table for a 2-input AND gate.
2. What is the output of an OR gate when both inputs are 0?
3. Fill in the truth table for a 2-input XOR gate.
4. What type of gate is considered a "universal gate"?
5. Construct a Boolean expression for the following circuit: an OR gate whose output goes into a NOT gate.
6. Draw how a NOR gate can be used to make a NOT gate.

Key Answers

1. AND Gate Truth Table:

A	**B**	**X = A.B**
0	0	0
0	1	0
1	0	0
1	1	1

2. Output = 0 (since OR only outputs 1 if at least one input is 1)
3. XOR Truth Table:

A	**B**	**X = A⊕B**
0	0	0
0	1	1
1	0	1
1	1	0

4. NAND and NOR are both universal gates.
5. Boolean Expression: X = ¬(A + B) or X = (A + B)'
6. Connect both inputs of a NOR gate to the same input (A) → output = ¬A

Homework Assignments

1. **Truth Tables Practice**: Draw the truth tables for the following gates: NOR, NAND, XNOR.
2. **Boolean Expression**: Write Boolean expressions for:
   - A circuit with A AND B, then NOT the result.
   - A circuit with A OR B, ANDed with NOT C.
3. **Symbol Practice**: Draw symbols for XOR and XNOR gates.
4. **Application**: Identify one real-world application for each of the following gates:
   - AND
   - OR
   - XOR
   - NAND
5. **Challenge**: Use only NAND gates to replicate an AND gate and a NOT gate.

Topic 3: Representation of Logic Gates

/ A – Target Population :-

This topic is designed for **first-year students** in the **Department of Electronic Techniques** at the **Technological Institute of Basra**, who are just beginning their exposure to digital system design. These students must be able to **read,**

interpret, and create logic diagrams, which is an essential skill in both academic labs and future engineering workplaces.

1 / B – Rationale :-

While understanding logic gates individually is important, the **ability to represent and connect them in circuit form** is what enables students to transition from theory to practical application. Circuit diagrams and logic symbols are the **language of digital electronics**, used in:

- Schematics
- IC datasheets
- Circuit design software
- Documentation and troubleshooting

Mastery of these representations empowers students to:

- Read and design digital logic circuits
- Interpret Boolean algebra in physical form
- Debug and simulate circuits effectively

1 / C – Central Idea :-

This unit introduces the standard **graphical and algebraic representations** of logic gates. Key components include:

1. **Standard Symbols** of AND, OR, NOT, NAND, NOR, XOR, XNOR
2. **Boolean Expressions** associated with each gate and circuit
3. **Truth Tables** to match inputs and outputs
4. **Multi-Gate Circuits**: Combining several logic gates in one diagram
5. **Translation** between logic expressions, truth tables, and circuit diagrams

1 / D – Performance Objectives :-

After completing this unit, students will be able to:

Knowledge

- Recognize and label standard logic gate symbols

- Understand the Boolean expressions related to gates and gate combinations

Skills

- Draw circuit diagrams from Boolean expressions
- Translate truth tables into logic circuit schematics
- Use gates to represent compound Boolean operations

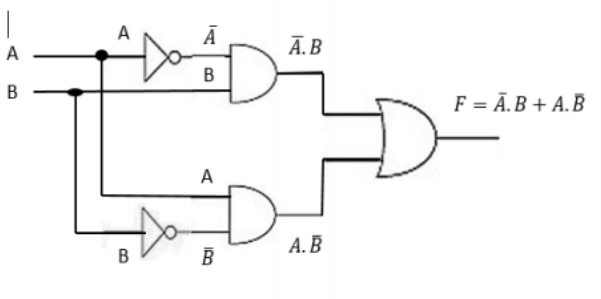
Competence

- Design and interpret logic circuits used in combinational systems
- Apply representation skills in simulation tools or hardware implementation

س1/ ارسم المخطط المنطقي للمعادلة المنطقية الاتية:

$$F = \bar{A}.B + A.\bar{B}$$

الحل:

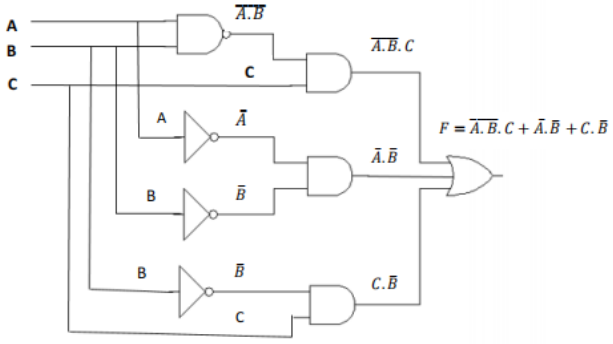


A	B	$\bar{A}$	$\bar{B}$	$\bar{A}.B$	$A.\bar{B}$	F
0	0	1	1	0	0	0
0	1	1	0	1	0	1
1	0	0	1	0	1	1
1	1	0	0	0	0	0

س2/ارسم المخطط المنطقي للمعادلة المنطقية الاتية:

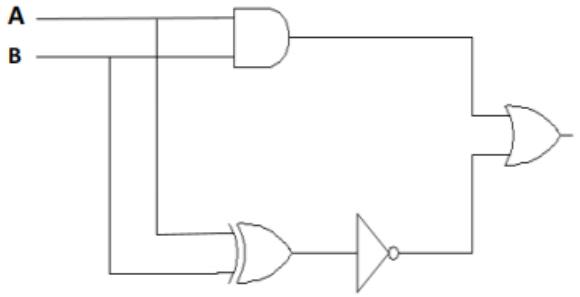
$$F = \overline{A.B}.C + \bar{A}.\bar{B} + C.\bar{B}$$

الحل:



A	B	C	$\bar{A}$	$\bar{B}$	$\overline{A.B}$	$\overline{A.B}.C$	$\bar{A}.\bar{B}$	$C.\bar{B}$	F
0	0	0	1	1	1	0	1	0	1
0	0	1	1	1	1	1	1	1	1
0	1	0	1	0	1	0	0	0	0
0	1	1	1	0	1	1	0	0	1
1	0	0	0	1	1	0	0	0	0
1	0	1	0	1	1	1	0	1	1
1	1	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0

س3/ اكتب المعادلة المنطقية للمخطط المنطقي الاتي واكتب جدول الحقيقة:

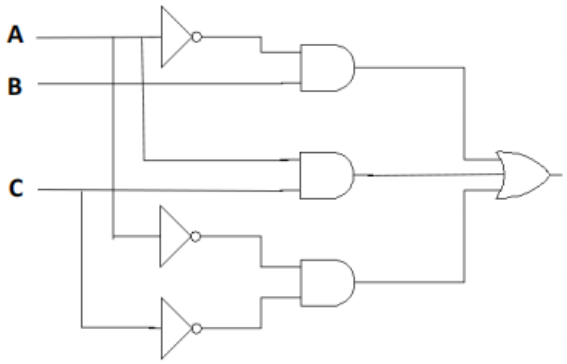


الحل:

$$F = A.B + \overline{A \oplus B}$$

A	B	A.B	A⊕B	$\overline{A \oplus B}$	F
0	0	0	0	1	1
0	1	0	1	0	0
1	0	0	1	0	0
1	1	1	0	1	1

س4/اكتب المعادلة المنطقية للمخطط المنطقي الاتي واكتب جدول الحقيقة:



الحل:

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

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

Post-Test

Answer the following:

1. Draw the standard logic symbol for an **XOR** gate.
2. Write the **Boolean expression** for the circuit consisting of an AND gate with inputs A and B, followed by a NOT gate.
3. Given the expression **X = A + B·C**, draw the corresponding logic circuit.
4. Complete the **truth table** for the Boolean expression **X = ¬A + B**.
5. What is the output of the circuit when A=1 and B=0 in the following:
    - AND gate
    - OR gate
    - NOR gate

Key Answers

1. XOR symbol: An OR symbol with an extra curved line on the input side.
2. Boolean Expression: X = ¬(A·B)
3. Circuit:
    - B and C go to an AND gate → output1
    - A goes to another input line
    - output1 and A go to an OR gate → X
4. Truth Table:

A	**B**	**¬A**	**X = ¬A + B**
0	0	1	1
0	1	1	1
1	0	0	0
1	1	0	1

5. Gate outputs with A=1, B=0:
    - AND: 1·0 = **0**
    - OR: 1+0 = **1**
    - NOR: NOT(1+0) = **0**

Homework Assignments

1. **Draw logic symbols** for the following gates: NAND, NOR, XNOR.
2. **Write Boolean expressions** for each of the following circuits:
    - A circuit with A and B entering an OR gate, then output to a NOT gate.
    - A circuit with three inputs: A AND B, then the result ORed with C.
3. **Create truth tables** for:
    - $X = A \cdot \neg B$
    - $X = (A + B)'$
4. **Translate** the following Boolean expressions into logic diagrams:
    - $X = A \cdot B + \neg C$
    - $X = \neg(A + \neg B)$
5. **Challenge**: Given a circuit diagram of three gates connected in series (AND → NOT → OR), write the Boolean expression and generate the truth table.

Topic 4: Boolean Algebra

1 / A – Target Population :-

This topic is tailored for **first-year undergraduate students** in the **Department of Electronic Techniques** at the **Technological Institute of Basra**. The content assumes students have prior knowledge of logic gates and truth tables and are now ready to manipulate logic expressions mathematically and reduce circuit complexity through algebraic techniques.

---

1 / B – Rationale :-

In digital systems, circuit efficiency matters. More components mean more cost, size, and power consumption. Boolean Algebra provides a **powerful tool for simplifying digital logic circuits**. Using algebraic rules, students can reduce complex expressions to their simplest forms, enabling:

- Efficient circuit design with fewer gates
- Easier troubleshooting and logic analysis
- Seamless transition to advanced topics like Karnaugh Maps, programmable logic devices, and VHDL

Boolean Algebra also strengthens **abstract reasoning** and aligns closely with programming logic, making it foundational in electronics and computer engineering education.

---

1 / C – Central Idea :-

This topic introduces students to the **language and rules of Boolean logic**, which form the algebraic basis for designing and simplifying digital logic circuits. Core concepts include:

1. **Boolean Variables and Expressions**: Using A, B, C to represent logic signals
2. **Basic Boolean Operations**: AND (·), OR (+), NOT (¬ or ')
3. **Boolean Laws and Theorems**:
    - Identity laws
    - Null laws
    - Idempotent laws
    - Involution and complement
    - Distributive, associative, and commutative laws
4. **Simplification Techniques**:
    - Combining like terms
    - Applying absorption and consensus theorems
    - Removing redundancy
5. **Equivalence Verification**: Using truth tables or simplification to confirm logic equivalence

---

1 / D – Performance Objectives :-

By the end of this unit, students will be able to:

Knowledge

- Define Boolean variables and expressions
- List and describe Boolean laws and rules

Skills

- Simplify logic expressions using Boolean algebra
- Translate between expressions, truth tables, and logic diagrams
- Identify equivalent expressions and functions

Competence

- Apply simplification techniques to optimize circuit designs
- Use Boolean algebra to troubleshoot, verify, and implement digital logic functions in real circuits

الجبر البولياني"Boolean Algebra"

ما هو الجبر البولياني؟

الجبر البولياني (Boolean Algebra) هو فرع من الرياضيات يتعامل مع المتغيرات التي يمكن أن تأخذ فقط قيمتين :**صحيح** (1)أو **خطأ** .(0)يعتمد الجبر البولياني على العمليات المنطقية مثل **الـ AND** و **OR**و **NOT**، ويستخدم لوصف والتحليل والتصميم للدوائر المنطقية الرقمية وأنظمة الحوسبة.

أهمية الجبر البولياني:

1. **الأساس في تصميم الدوائر المنطقية:**
الجبر البولياني هو الأساس في تصميم **الدوائر المنطقية الرقمية** مثل **الذاكرة، المعالجات،** و **الدوائر الإلكترونية** .يتم استخدامه لتبسيط العمليات المعقدة وتصميم الدوائر بكفاءة.
2. **تحليل الدوائر المنطقية:**
يُستخدم الجبر البولياني لتحليل وتبسيط الدوائر الرقمية، مما يساهم في تقليل تعقيد الدوائر وتقليل عدد البوابات المنطقية (gates) المطلوبة لتحقيق وظيفة معينة.
3. **في الحوسبة:**
يستخدم الجبر البولياني بشكل أساسي في **المعالجات** و **الأنظمة الرقمية** حيث يتم تمثيل البيانات والمنطق باستخدام الأرقام الثنائية (0 و 1).
4. **في البرمجة والأنظمة المنطقية:**
الجبر البولياني مهم في **البرمجة المنطقية** وفي تحديد القيم المنطقية مثل الصحة أو الخطأ في **الشروط** و **الحلقات** في البرمجيات.
5. **في تحسين الدوائر الرقمية:**
يساعد الجبر البولياني في **تحسين الدوائر**، حيث يُستخدم لتقليل **عدد المدخلات** أو **عدد البوابات المنطقية** في تصميم الدوائر الرقمية، مما يحسن **الكفاءة** و **أداء الدائرة**.

ماذا يستخدم الجبر البولياني؟

- **الدوائر المنطقية الرقمية** :مثل دوائر **المعالجات، الذاكرة،** و **وحدات التحكم**.
- **تصميم الأنظمة الرقمية** :يتم استخدام الجبر البولياني لتصميم **الدوائر الإلكترونية** التي تستخدم في الآلات والأجهزة الذكية.
- **الخوارزميات** :يُستخدم الجبر البولياني في تطوير الخوارزميات التي تعتمد على المنطق الثنائي مثل الخوارزميات المستخدمة في الذكاء الاصطناعي أو البرمجة المنطقية.
- **أنظمة التحكم** :يُستخدم في تصميم **أنظمة التحكم** مثل **أنظمة التحكم في الطيران** أو **الأنظمة الآلية**.
- **الشبكات الرقمية** :يُستخدم الجبر البولياني في تحليل **الشبكات الرقمية** ومعالجة البيانات في التطبيقات مثل **الشبكات العصبية**.

قوانين التكامل:(Complementation Laws)

مصطلح التكامل يعني ببساطة **العكس**، أي تغيير 0 إلى 1 و 1 إلى 0. القوانين الخمسة للتكامل هي كما يلي:

Law 1: $\overline{0} = 1$
Law 2: $\overline{1} = 0$
Law 3: if $A = 0$, then $\overline{A} = 1$
Law 4: if $\overline{A} = 1$, then $\overline{\overline{A}} = 0$
Law 5: $\overline{\overline{A}} = 0$ (double complementation law)

قوانين:OR (OR Laws)

القوانين الأربعة لقانون OR هي كما يلي:

Law 1: $A + 0 = 0$(Null law)
Law 2: $A + 1 = 1$(Identity law)
Law 3: $A + A = A$
Law 4: $A + \overline{A} = 1$

قوانين:AND (AND Laws)

القوانين الأربعة لقانون AND هي كما يلي:

Law 1: $A \cdot 0 = 0$(Null law)
Law 2: $A \cdot 1 = 1$(Identity law)
Law 3: $A \cdot A = A$
Law 4: $A \cdot \overline{A} = 0$

قوانين التبادلية:(Commutative Laws)

تسمح القوانين التبادلية بتغيير ترتيب المتغيرات في عمليات AND أو OR. هناك قانونان تبادليان كما يلي:

Law 1: A + B = B + A

Proof

A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	1

=

B	A	B+ A
0	0	0
0	1	1
1	0	1
1	1	1

Law 2: A . B = B . A

Proof

A	B	A . B
0	0	0
0	1	0
1	0	0
1	1	1

=

B	A	B. A
0	0	0
0	1	0
1	0	0
1	1	1

This law can be extended to any number of variables. For example
A.B. C = B. C. A = C. A. B = B. A. C

قوانين التجميع:(Associative Laws)

تسمح قوانين التجميع بتجميع المتغيرات. هناك قانونان للتجميع كما يلي:

Law 1: (A + B) + C = A + (B + C)

Proof

A	B	C	A+B	(A+B)+C
0	0	0	0	0
0	0	1	0	1
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

=

A	B	C	B+C	A+(B+C)
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	1
1	0	0	0	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Law 2: (A .B) C = A (B .C)

Proof

A	B	C	AB	(AB)C
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	0	0
1	1	0	1	0
1	1	1	1	1

=

A	B	C	B.C	A(B.C)
0	0	0	0	0
0	0	1	0	0
0	1	0	0	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

This law can be extended to any number of variables. For example
A(BCD) = (ABC)D = (AB) (CD)

قوانين التوزيع:(Distributive Laws)

تسمح قوانين التوزيع بالعامل أو التوسيع في التعبيرات. هناك قانونان للتوزيع كما يلي:

Law 1: A (B + C) = AB + AC

Proof

A	B	C	B+C	A(B+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

=

A	B	C	AB	AC	A+(B+C)
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	1	1

Law 2: A + BC = (A+B) (A+C)

Proof **RHS** = (A+B) (A+C)
= AA + AC + BA + BC
= A + AC + AB + BC
= A (1+ C + B) + BC
= A. 1 + BC (1 +C + B = 1 + B = 1)
= A + BC
= **LHS**

6) Inversion Law

A" = A

7)

1. A + (A . B) = A

Proof: A (1+ B) = A . 1 = A

Post-Test

1. Simplify the Boolean expression:
   X = A·B + A·B'
2. Apply Boolean laws to reduce:
   X = A + A·B
3. Which law justifies this step:
   A + A = A?

4. Determine the simplified form of:
   X = (A + B)·A
5. Which law is applied here:
   A + A'·B = A + B?

Answer Key

1. X = A·(B + B') = A·1 = **A**
2. X = A(1 + B) = A·1 = **A**
3. **Idempotent Law**
4. X = A·(A + B) = **A** (by Absorption Law)
5. **Distributive + Absorption Law**

Homework

1. **Simplify** the following using Boolean laws:
   - a) X = A + A·B
   - b) X = AB + A'B
   - c) X = (A + B)(A + B')
   - d) X = A·B + A·B' + A'·B
2. **Identify the laws** used in each simplification.
3. **Translate** the following logic circuit into a Boolean expression and simplify:
   - A and B go into an OR gate
   - Output and A again go into an AND gate
   - Output is X
4. **Create truth tables** for both:
   - Original expression
   - Simplified expression
     Confirm their equivalence.
5. **Advanced Challenge**: Simplify using a combination of DeMorgan's Theorems and basic laws:
   - X = ¬(¬A + B) + A·B

Topic 5: DeMorgan’s Theorems

1 / A – Target Population :-

This topic is intended for **first-year students** in the **Department of Electronic Techniques** at the **Technological Institute of Basra**, who already have foundational knowledge in Boolean algebra and logic gates. These students are now ready to explore logic manipulation using DeMorgan's Theorems to simplify and transform logic expressions, especially in cases involving **inversion and universal gate applications**.

---

1 / B – Rationale :-

DeMorgan's Theorems are essential in digital circuit design and simplification. They allow for the **transformation of logic expressions containing NOT operations applied to grouped variables**, and are widely used in:

- Designing logic circuits using only NAND or NOR gates
- Reducing the complexity of expressions in programmable logic
- Ensuring compatibility with IC logic families and design constraints

These theorems also reinforce **critical reasoning in algebraic manipulation** and set the stage for implementing minimized logic functions in hardware and software-based logic systems.

---

1 / C – Central Idea :-

This unit centers around **two logical equivalences** known as DeMorgan's Theorems, which allow the breakdown of inverted expressions involving AND and OR:

1. **Theorem 1**
   $\neg(A \cdot B) = \neg A + \neg B$
2. **Theorem 2**
   $\neg(A + B) = \neg A \cdot \neg B$

These theorems are vital in:

- Converting logic using universal gates (NAND, NOR)

- Rewriting expressions to simplify logic circuits
- Implementing logic functions without using NOT gates separately

---

1 / D – Performance Objectives :-

Upon completing this unit, students will be able to:

Knowledge

- State and describe DeMorgan's Theorems
- Identify inverted logic groups in expressions and circuits

Skills

- Apply DeMorgan's Theorems to simplify Boolean expressions
- Convert SOP to POS using DeMorgan's approach
- Redraw logic circuits using NAND or NOR gates only

Competence

- Design and optimize circuits using DeMorgan transformations
- Analyze logic outputs in complex gate configurations
- Apply these theorems in real-world systems and microprocessor design

نظرية دي مورغان:(De Morgan's Theorem)

نظرية دي مورغان هي قاعدة هامة في الجبر البولياني تستخدم لتحويل العمليات المنطقية بين **النفي** و **العوامل** في التعبيرات المنطقية. تقدم هذه النظرية طريقتين لتحويل النفي (NOT) من حيث العمليات ANDو OR.

- نظرية دي مورغان تساعد في تبسيط التعبيرات المنطقية وتحويلها من شكل إلى آخر باستخدام النفي.
- على سبيل المثال، إذا كان لديك تعبير معقد يحتوي على عمليات AND أو OR مع نفي، يمكنك استخدام نظرية دي مورغان لتحويله إلى شكل أبسط أو أكثر ملاءمة للتطبيقات المختلفة.

Law 1: $\overline{A + B} = \overline{A} \cdot \overline{B}$

Proof

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

=

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

ينص هذا القانون على أن متمم جمع المتغيرات يكون مساويًا لجداء متمماتها الفردية.

Law 2: $\overline{A \cdot B} = A + B$

Proof

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

=

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

ينص هذا القانون على أن متمم حاصل ضرب المتغيرات يكون مساويًا لجمع متمماتها الفردية.

ملاحظة: متمم الدالة يعني نفي الدالة.

Y=A+B.C

$$\bar{Y} = \overline{A + B.C}$$

$$\overline{Y} = \overline{A}.\overline{B.C}$$

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

ملاحظة: إذا تكرر الرمز مرتين او أكثر يحذف ويبقى رمز واحد فقط.

Y=A.B+A.B

=A.B(~~1+1~~)

=A.B

Y=A'.B'+A'.B'+A'.B'

=A'.B'(~~1+1+1~~)

=A'.B'

مثال 1: بسط المعادلة الاتية:

F=A+A.B

الحل:

F=A+A.B

F=A ~~(1+B)~~

F=A

مثال 2: بسط المعادلة الاتية:

$$F=A.B+A.B'$$

الحل:

$F=A.B+A.B'$

$F=A\,(B+B')$

$F=A$

مثال 3: بسط المعادلة الاتية:

$$F=A(A'+B)$$

الحل:

$F=A(A'+B)$

$F=A.A'+A.B \longrightarrow F=A.B$

مثال 4: بسط المعادلة الاتية:

$$F=A.B+B.C(B+C)$$

الحل:

$F=A.B+B.C(B+C)$

$F=A.B+B.B.C+B.C.C$

$F=A.B+B.C+B.C$

$F=A.B+B.C$

$F=B(A+C)$

مثال 5: بسط المعادلة الاتية:

$$F=A.A+A.C+A.B+B.C$$

الحل:

$F=A.A+A.C+A.B+B.C$

$F=A +A.C+A.B+B.C$

$F=A(1+C+B)+B.C$

$F=A+B.C$

مثال 6: بسط المعادلة الاتية:

F=A.B+A(A+C)+B(A+C)

الحل:

F=A.B+A(A+C)+B(A+C)

F=A.B+A.A+A.C+A.B+B.C

F=A.B+A+A.C+B.C

F=A(B+1+C)+B.C

F=A+B.C

Example 3: Prove that:

a'b + b'c + ac' = ab' + bc' + a'c

Solution:

The left side:

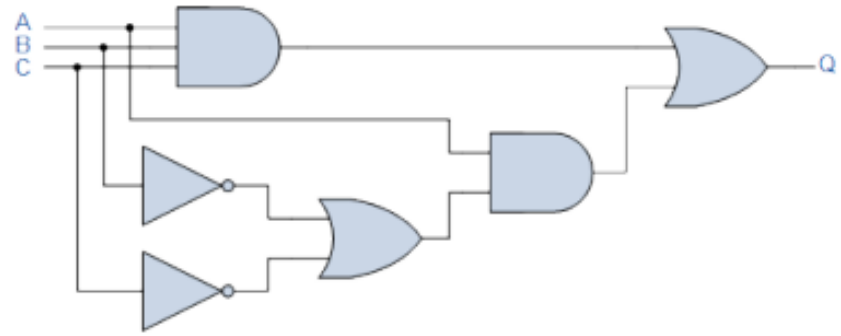
= a'b(c + c') + (a + a')b'c + a(b + b')c'

= a'bc +a'bc' + ab'c + a'b'c + abc' + ab'c'

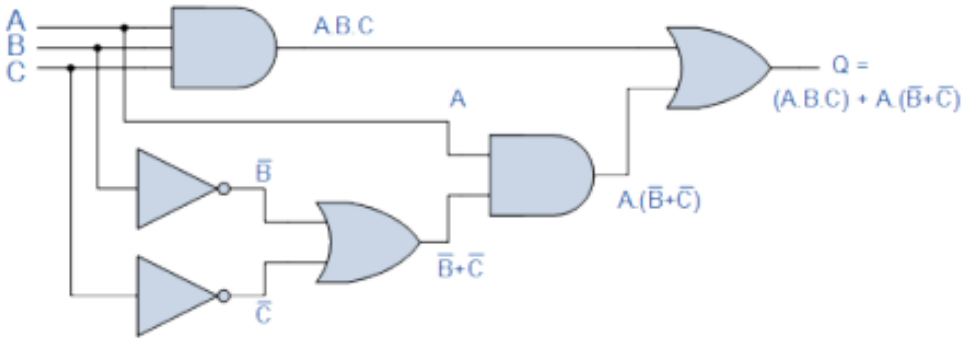
= ab'(c + c') + bc' (a + a') + a'c (b + b')

= ab' + bc' + a'c = **The right side.**

Example 1: Find the Boolean algebra expression for the following system.



Solution:



A	B	C	A.B.C	$\overline{B}$	$\overline{C}$	$\overline{B+C}$	A.($\overline{B+C}$)	Q
0	0	0	0	1	1	1	0	0
0	0	1	0	1	0	1	0	0
0	1	0	0	0	1	1	0	0
0	1	1	0	0	0	0	0	0
1	0	0	0	1	1	1	1	1
1	0	1	0	1	0	1	1	1
1	1	0	0	0	1	1	1	1
1	1	1	1	0	0	0	0	1

Post-Test

1. Apply DeMorgan's Theorem to simplify:
   X = ¬(A·B)
2. Apply DeMorgan's Theorem to this expression:
   X = ¬(A + B·C)
3. Use DeMorgan's Theorem to transform:
   X = ¬(A + B + C)
4. Convert the expression using NAND gates only:
   X = ¬(A·B)
5. Convert the expression using NOR gates only:
   X = ¬(A + B)

Key Answers

1. **¬(A·B) = ¬A + ¬B**
2. **¬(A + B·C) = ¬A · ¬(B·C) = ¬A · (¬B + ¬C)**
3. **¬(A + B + C) = ¬A · ¬B · ¬C**
4. Use a **NAND gate** with inputs A and B; output is ¬(A·B)
5. Use a **NOR gate** with inputs A and B; output is ¬(A + B)

Homework

1. **Simplify using DeMorgan's Theorems**:
   - a) ¬(A + ¬B)
   - b) ¬(A·(B + C))
   - c) ¬(A·B·C)
2. **Draw logic circuits** for the following expressions using only:
   - a) NAND gates: ¬(A·B + C)
   - b) NOR gates: ¬(A + B·C)
3. **Write the Boolean expression** for each circuit below:
   - Circuit with two AND gates feeding an OR gate, then passed through a NOT gate
   - A NOR gate with inputs A and B
4. **True or False? Justify**:
   - a) ¬A + ¬B = ¬(A·B)
   - b) ¬(A + B·C) = ¬A · ¬(B·C)

5. **Challenge**: Simplify the expression
   ¬[(A + B)' + (C·D)'] using DeMorgan's Theorem and Boolean algebra.

Topic 6: Karnaugh Map (K-Map)

1 / A – Target Population :-

This topic is tailored for **first-year students** in the **Department of Electronic Techniques** at the **Technological Institute of Basra**, who have already covered logic gates and Boolean algebra. Students now require visual and systematic tools like K-Maps to minimize complex logic expressions and reduce hardware requirements in circuit design.

---

1 / B – Rationale :-

Boolean simplification using algebra becomes increasingly difficult and error-prone with more variables. The **Karnaugh Map (K-Map)** offers a powerful visual method for **simplifying logic expressions** efficiently. It directly supports:

- Reducing the number of logic gates in a circuit
- Designing optimal digital systems
- Preparing students for programmable logic devices (PLDs) and VHDL logic modeling

The K-Map method is used by circuit designers to achieve **cost-effective, faster, and more power-efficient** circuits.

---

1 / C – Central Idea :-

The K-Map is a grid-based tool used to simplify Boolean expressions using **visual grouping of 1s (minterms)**. The topic covers:

1. **2-variable, 3-variable, and 4-variable K-Maps**
2. **Filling the map** using minterm indices
3. **Grouping rules**:
    - Groups of 1, 2, 4, 8 (must be powers of 2)
    - Must be adjacent (horizontally or vertically)
    - Wrapping around the edges is allowed
4. **Simplified Expression Extraction** from grouped cells
5. **"Don't Care" Conditions** for flexible optimization
6. **Conversion between truth tables, expressions, and K-Maps**

---

1 / D – Performance Objectives :-

After completing this unit, students will be able to:

Knowledge

- Define what a K-Map is and its purpose in digital logic
- Recognize the structure of 2-, 3-, and 4-variable K-Maps

Skills

- Populate K-Maps from truth tables or minterm expressions
- Identify and form optimal groupings of adjacent 1s
- Derive simplified Boolean expressions using SOP (Sum of Products) or POS (Product of Sums)

Competence

- Apply K-Maps to reduce circuit complexity
- Design circuits with fewer logic gates
- Evaluate and compare results from Boolean algebra and K-Maps

خارطة كارنوف(Karnaugh Map)

ما هي خارطة كارنوف؟

خارطة كارنوف هي طريقة رسومية لتبسيط الدوال الجبرية المنطقية (Boolean Functions) من خلال ترتيب القيم الممكنة لجميع المدخلات في جدول مشابه للجدول الذي يستخدم في الجبر البولياني.

2.كيفية رسم خارطة كارنوف:

خارطة كارنوف تكون عبارة عن شبكة تحتوي على خلايا تمثل كافة المخرجات الممكنة للوظيفة المنطقية بناءً على قيم المدخلات. عدد الخلايا يعتمد على عدد المدخلات.

- **2متغيرات** :**(A, B)** تكون الخارطة عبارة عن جدول x2.2
- **3متغيرات** :**(A, B, C)** تكون الخارطة عبارة عن جدول x4.2
- **4متغيرات** :**(A, B, C, D)** تكون الخارطة عبارة عن جدول x4.4

كل خلية في الخريطة تحتوي على قيمة مخرجة من الوظيفة المنطقية. يتم استخدام الأرقام الثنائية أو العشرية لتمثيل هذه القيم.

3.أهمية خارطة كارنوف:

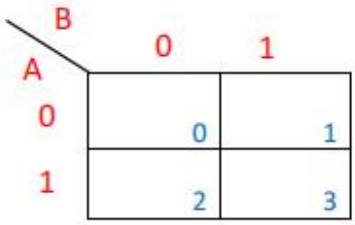
تساعد خارطة كارنوف في تبسيط المعادلات الجبرية المنطقية عن طريق البحث عن مجموعات من القيم المتشابهة (سواء 1 أو 0) يمكن دمجها باستخدام القواعد الجبرية لتبسيط التعبير.

4.كيفية استخدام خارطة كارنوف لتبسيط الدوال المنطقية:

1. **رسم الخارطة** :أولاً، قم برسم الجدول بناءً على عدد المدخلات في الوظيفة.
2. **ملء الخلايا بالقيم المنطقية** :ضع القيم الناتجة من الوظيفة المنطقية في الخلايا.
3. **إيجاد المجموعات** :ابحث عن مجموعات من الخلايا المتجاورة التي تحتوي على نفس القيمة (1 أو 0).
    - يجب أن تكون هذه المجموعات من 1 أو 2 أو 4 أو 8 خلايا.
4. **تبسيط المعادلة** :بعد تحديد المجموعات، يمكنك كتابة المعادلة المبسطة باستخدام القيم المشتركة بين الخلايا المجتمعة.

خارطة كارنوف للمتغيرين (Karnaugh Map for Two Variables) هي أداة قوية وبسيطة تساعد في تبسيط الدوال المنطقية ذات المتغيرين باستخدام رسم بياني. تُستخدم بشكل رئيسي لتقليص تعبيرات المنطق الجبري المعقدة إلى شكل أبسط يمكن تنفيذه بسهولة في دوائر منطقية باستخدام بوابات منطقية أقل.

1.خارطة كارنوف لمتغيرين $2^2=4$



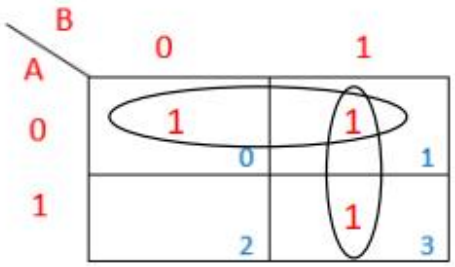
ملاحظة:- اذا كان بالسؤال موجودة علامة ∑ يعني نعوض بالخلايا رقم 1

ملاحظة:- اذا كان بالسؤال موجودة علامة π يعني نعوض بالخلايا رقم 0

مثال1:

$$f(A,B)=\sum(0,1,3)$$

الحل:

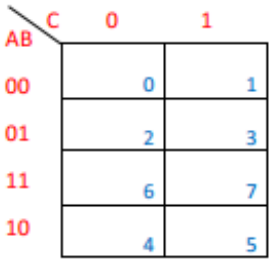


$$Y = \bar{A}.\bar{B}.B + B.\bar{A}.A$$

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

خارطة كارنوف للثلاث متغيرات (Karnaugh Map for Three Variables) هي أداة مرئية تُستخدم لتبسيط الدوال المنطقية ذات الثلاث مدخلات (A, B, C) بطريقة سهلة وفعّالة. عند استخدام خارطة كارنوف لثلاث متغيرات، يتم ترتيب القيم المنطقية في شبكة تتكون من 8 خلايا، والتي تمثل جميع التركيبات الممكنة للمتغيرات الثلاثة. الهدف من هذه الخارطة هو تبسيط الدوال المعقدة عن طريق دمج القيم المتشابهة (1 أو 0) في مجموعات.

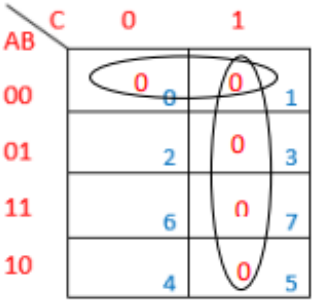
2.خارطة كارنوف لثلاثة متغيرات $2^3=8$



مثال1:

$f(A,B,C)=\pi(0,1,3,5,7)$

الحل:



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

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

Post-Test

1. Draw a **3-variable K-Map** and fill it with 1s for the function:
   F(A,B,C) = Σm(1,2,3,5)
2. For **F = Σm(0,1,2,5,7)**, group the minterms and write the simplified SOP expression.
3. Explain what makes two minterms “adjacent” on a K-Map.
4. How would you represent the following using K-Map?
   F = A'B + AB'
5. What is the advantage of using a **“don’t care” condition** in a K-Map?

Key Answers

1. K-Map filled:

A\BC	00	01	11	10
0	0	1	1	1
1	0	1	0	0

2. Grouping:
    - Group of 2 (0,1): A'B'C' + A'B'C = A'B'
    - Group of 2 (2,3): A'BC
    - Group of 1 (5): AB'C
    - Result: **F = A'B' + A'BC + AB'C**
3. Adjacent = cells differing by **only one variable**; side-by-side or top-bottom in Gray code order.
4. You fill 1s in cells corresponding to (A=0, B=1) and (A=1, B=0) → Then group and extract **A'B + AB'**
5. Don't care conditions enable **larger groupings**, leading to **simpler expressions and fewer gates.**

Homework

1. **Complete the following K-Map exercises:**
    - a) F = Σm(0,1,2,5,6,7)
    - b) F = Σm(1,3,7) with d = (2,6)
    - c) F = Σm(4,5,6,7,12,13,14,15)
2. **Given a truth table**, create the K-Map and derive simplified SOP and POS expressions.
3. **Find and highlight redundant terms** in the expression: **F = A'B + AB + A'B'C**
4. **Draw the circuit diagram** of a logic function minimized using K-Map: **F = Σm(1,3,5,7)**
5. **Compare two methods**:
    - Simplify F = A·B + A·B' using Boolean Algebra
    - Simplify the same using a K-Map
    - Explain which method is easier and why.

Topic 7: Digital Comparator

1 / A – Target Population :-

This topic is designed for **first-year students** in the **Department of Electronic Techniques** at the **Technological Institute of Basra** who have already mastered logic gates, Boolean algebra, and Karnaugh map simplification. These students are now prepared to understand **comparison logic**, which is essential for decision-making in digital circuits and control systems.

---

1 / B – Rationale :-

A **comparator** is a digital circuit used to **compare two binary values** and determine whether one is greater than, less than, or equal to the other. Comparators are essential in:

- Arithmetic logic units (ALUs)
- Digital control systems
- Sorting algorithms
- Digital counters and memory addressing
- Microprocessors and programmable controllers (PLCs)

Understanding how comparators work enables students to design systems that can make **decisions based on logic conditions**, a foundational element in computer engineering and embedded systems.

---

1 / C – Central Idea :-

This unit focuses on the **concept, design, and implementation** of **digital comparators**, especially **1-bit and 2-bit** versions. It introduces:

1. **Definition and Function**: What comparators do in digital systems
2. **1-Bit Comparator Logic**:
   - A > B
   - A < B
   - A = B
3. **2-Bit Comparator Extension**: Handling higher-order bits
4. **Truth Tables** and **Boolean Equations** for comparison
5. **Implementation using Basic Gates**
6. **Real-world applications** and digital ICs (e.g., 7485 comparator chip)

---

1 / D – Performance Objectives :-

After completing this unit, students will be able to:

Knowledge

- Define the function of a digital comparator
- Describe the logic conditions for A > B, A = B, and A < B

Skills

- Design 1-bit and 2-bit comparators using logic gates
- Write Boolean expressions for comparison outcomes
- Create truth tables for comparator circuits

Competence

- Apply comparators in real digital applications
- Analyze multi-bit comparison logic
- Integrate comparators into larger digital control circuits

(Digital Comparator)المقارن الرقمي

المقارن الرقمي هو دائرة رقمية تُستخدم لمقارنة قيمتين عدديتين (عادة في النظام الثنائي) وتحديد العلاقة بينهما، مثل:

- **أكبر من** .(Greater than)
- **أقل من** .(Less than)
- **تساوي** .(Equal)

وظيفة المقارن الرقمي

- مقارنة القيم الثنائية.(Binary Numbers)
- إخراج إشارات تحدد العلاقة بين القيم المُدخلة:
  - إذا كانت القيمة الأولى أكبر.
  - إذا كانت القيمة الثانية أكبر.
  - إذا كانت القيمتان متساويتين.

مكونات المقارن الرقمي

يتكون المقارن الرقمي بشكل أساسي من بوابات منطقية) مثل (AND, OR, XOR تُستخدم لتحديد العلاقة بين مدخلاته. يعتمد تصميمه على عدد البتات (Bits) التي يتم مقارنتها.

أنواع المقارن الرقمي

1. **مقارن أحادي البت:(1-bit Comparator)**
   - يُستخدم لمقارنة رقمين مكوّنين من بت واحد.
   - الإخراج يكون:
     - A>B: إذا كانت القيمة الأولى أكبر.
     - A<B: إذا كانت القيمة الثانية أكبر.
     - A=B: إذا كانت القيمتان متساويتين.
2. **مقارن متعدد البتات:(Multi-bit Comparator)**
   - يُستخدم لمقارنة أرقام تحتوي على أكثر من بت واحد) مثل 4bit-، 8.(bit-
   - يُقارن كل بت في الرقم الأول مع البت المقابل له في الرقم الثاني، بدءًا من أكثر البتات أهمية.(MSB - Most Significant Bit)

دائرة المقارن الرقمي

يمكن تصميم المقارن الرقمي باستخدام بوابات منطقية:

- بوابات XOR للكشف عن الفرق بين القيم.
- بوابات AND و ORلتحديد إذا ما كانت قيمة أكبر أو أصغر.

مثال: مقارن 1-bit

مدخلات: A و B.

الإخراج:

1. $\overline{B} \cdot A = B < A$
2. $B \cdot \overline{A} = B > A$
3. $\overline{A \oplus B} = B = A$

1. المعادلة الأولى:

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

- $\overline{B}$ تعني **عكس قيمة** B (أي إذا كان $B = 0$، يصبح $\overline{B} = 1$ والعكس).
- عند ضرب $\overline{B}$ في A باستخدام بوابة AND (·):
- إذا كان $A = 1$ و$B = 0$، سيكون الناتج $1 \rightarrow$ أي أن $B < A$.
- في أي حالة أخرى، سيكون الناتج 0.
- إذن، هذه المعادلة تعني أن **الإخراج يكون 1 فقط عندما يكون** $B < A$.

تطبيقات المقارن الرقمي

- **أنظمة التحكم :**لمعرفة إذا كانت قيمة معينة تجاوزت حدًا معينًا.
- **المعالجات الدقيقة :**لمقارنة القيم أثناء العمليات الحسابية.
- **وحدات الحساب والمنطق :(ALU)** لتحديد العمليات المناسبة بناءً على المقارنة.
- **أنظمة الفرز :(Sorting Systems)** لترتيب القيم الرقمية.
- **العدادات الرقمية :**لمعرفة ما إذا تم الوصول إلى قيمة معينة.

1-Bit Comparator

السؤال:

قم بإكمال جدول الحقيقة التالي لمقارن 1 بت:

$B = A$	$B > A$	$B < A$	B	A
?	?	?	0	0
?	?	?	1	0
?	?	?	0	1
?	?	?	1	1

الإجابة:

$B = A$	$B > A$	$B < A$	B	A
1	0	0	0	0
0	1	0	1	0
0	0	1	0	1
1	0	0	1	1

Post-Test

1. Write the Boolean expression for:
   A > B when A and B are 1-bit inputs.
2. Complete the truth table for a 1-bit comparator.
3. What logic condition must be true for **A = B**?
4. If A = 11 (binary) and B = 10, which condition is true?
5. Design a logic circuit using AND, OR, and NOT gates for the case:
   A = B in a 1-bit comparator.

Key Answers

1. A > B = A·B'
2. Truth table:

A	B	A > B	A = B	A < B
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

3. A = B = A'·B' + A·B
4. A > B → since 11 > 10
5. A = B = A'·B' + A·B → implement using NOT and AND gates into OR gate

Homework Assignments

1. **Draw the logic diagram** of a 1-bit comparator for A > B, A = B, A < B.
2. **Write Boolean expressions** for a 2-bit comparator.
   Let A = A1A0 and B = B1B0
3. **Create a truth table** for all possible input combinations of a 2-bit comparator.
4. **Design a comparator circuit** that gives an output = 1 when A ≤ B.
5. **Application scenario**:
   In a digital temperature control system, explain how a comparator would be used to compare the input from a sensor with a threshold.

Topic 8: Two-Level Comparator

1 / A – Target Population :-

This topic is designed for **first-year students** in the **Department of Electronic Techniques** at the **Technological Institute of Basra**, who have already studied 1-bit and 2-bit digital comparators. These students are now ready to understand how comparators can be scaled and interconnected to handle multi-bit binary comparisons in real digital systems.

---

1 / B – Rationale :-

Simple comparators compare 1 or 2 bits, but most real-world applications—like microprocessors, digital controllers, or memory address decoding—require comparing **multi-bit binary numbers**. A **two-level comparator** is a

hierarchical circuit that compares two or more **multi-bit binary values** by cascading smaller comparators.

Understanding this structure helps students:

- Build scalable digital systems
- Handle logic design for data sorting, priority encoding, and digital control
- Learn real-world logic integration using **modular thinking**

It also prepares students for advanced topics like **priority encoders, arithmetic units**, and **control logic in CPUs**.

---

1 / C – Central Idea :-

The focus of this unit is understanding and building **multi-bit comparators using multiple comparison levels**. Key ideas include:

1. **Cascading Comparators**
    - Connecting 1-bit or 2-bit comparator blocks to compare 3, 4, or more bits
    - Logic flow: compare from most significant bit (MSB) down
    - Use of control signals: A = B, A > B, A < B
2. **Two-Level Structure**
    - First level: compare lower bits (e.g., A0A1 vs B0B1)
    - Second level: compare higher bits (e.g., A2A3 vs B2B3)
    - Result determined based on MSB comparator unless equal, then check LSB
3. **Example: 4-bit Comparator Using Two 2-bit Stages**
    - Use two 2-bit comparators: one for A3A2 vs B3B2, another for A1A0 vs B1B0
    - Use output of higher-order stage to enable or block result from lower stage
4. **Enable and Cascade Logic**
    - How equality cascades work (EQ must be 1 from upper to allow LSB compare)
    - Chip example: 7485 comparator (4-bit)

---

1 / D – Performance Objectives :-

By the end of this unit, students will be able to:

Knowledge

- Describe the concept and logic flow of multi-level comparison
- Understand when to use hierarchical comparator design

Skills

- Design a 4-bit comparator using 1-bit or 2-bit comparators
- Implement cascading logic for multi-stage comparisons
- Write Boolean expressions for comparator hierarchy

Competence

- Build practical multi-bit comparison circuits
- Integrate comparators into larger digital decision systems
- Optimize comparison logic in programmable hardware environments

Comparison Hierarchy

- Start from MSB: if A3 > B3 → A > B
- If A3 = B3, compare A2 and B2, then A1/B1, then A0/B0

Two-Level Comparator Block Design

- First comparator compares high bits
- Second comparator is enabled only when high bits are equal

Cascade Connection

- EQ output of stage 1 feeds enable of stage 2
- Output logic:
  - A > B = (A3 > B3) + (A3 = B3)·(A2 > B2) + …
  - A = B = EQ(H) · EQ(L)

Example Truth Logic

- For A3A2A1A0 = 1011 and B3B2B1B0 = 1001
- A3 = B3 (1), A2 = 0 < B2 = 0 → A = B
- But A1 = 1 > B1 = 0 → A > B

المثال الثاني: مقارنة 2-بت

المطلوب:

استخدام جدول الحقيقة لمقارنة رقمين ثنائيين $A = {}_0A_1A$ و $B = {}_0B_1B$، واستخراج العلاقات.

الحل:

1. المدخلات: $A, {}_0A_1$ و $B, {}_0B_1$.
2. الإخراج:

- إذا كانت البتات الأعلى $({}_1A_1, B)$ مختلفة:
  - $B < A \rightarrow {}_1B < {}_1A$.
  - $B > A \rightarrow {}_1B > {}_1A$.
- إذا كانت البتات الأعلى متساوية، يُقارن البتات الأقل $({}_0A_0, B)$.

$B = A$	$B > A$	$B < A$	${}_0B_1B$	${}_0A_1A$
1	0	0	00	00
0	1	0	01	00
0	1	0	10	00
0	1	0	11	00
0	0	1	00	01
1	0	0	01	01
0	1	0	10	01
0	1	0	11	01
0	0	1	00	10
0	0	1	01	10
1	0	0	10	10
0	1	0	11	10
0	0	1	00	11
0	0	1	01	11
0	0	1	10	11
1	0	0	11	11

Post-Test

1. What is the role of the **higher-order comparator** in a multi-level comparison?
2. In a 4-bit comparison (A3A2A1A0 vs B3B2B1B0), which bit is checked first and why?

3. How does equality at one level affect the next level in a two-level comparator?
4. Design a 4-bit comparator using **two 2-bit comparator blocks**.
5. Write a Boolean logic expression to describe:
    - A > B in a cascaded 2-level comparator
    - A = B

Key Answers

1. It makes the **first comparison decision**—if bits differ here, lower bits are ignored.
2. The **most significant bit (MSB)** is checked first because it has the highest weight.
3. The next level is only **enabled** if the higher level reports equality (A = B).
4. Use two 2-bit comparators:
    - First block compares A3A2 to B3B2
    - Second block compares A1A0 to B1B0
    - EQ from block 1 enables block 2
5. Boolean logic:
    - A > B = (A3 > B3) + (A3 = B3)·(A2 > B2) + …
    - A = B = (A3 = B3)·(A2 = B2)·(A1 = B1)·(A0 = B0)

Homework Assignments

1. **Draw a block diagram** for a 4-bit comparator using two 2-bit comparators.
2. **Explain in writing** how comparator outputs can be used in a traffic control system.
3. **Using logic gates**, build a truth table for a 3-bit comparator (A2A1A0 vs B2B1B0).
4. **Design and simulate** a 4-bit comparator using Logisim or Tinkercad Circuits.
5. **Challenge**: Implement a 4-bit comparator with only 1-bit comparator units and minimal logic gates.

Topic 9: Encoders

1 / A – Target Population :-

This topic is designed for **first-year students** in the **Department of Electronic Techniques** at the **Technological Institute of Basra**. At this stage, students are ready to explore how digital circuits translate real-world inputs into binary signals, which is essential for working with sensors, input devices, and digital control systems.

---

1 / B – Rationale :-

In digital systems, **encoders and decoders (codebreakers)** are used to **convert data between different formats**—for example, translating a keypress or sensor signal into a binary number. Encoders simplify inputs, while decoders expand binary outputs into human-readable or actuator-driven signals.

These components are used in:

- Keypads and keyboard systems
- Microcontroller input/output handling
- Data compression and signal transmission
- Error detection and correction systems
- Communication buses and address lines

Understanding how these circuits work prepares students for **real-life digital systems design**, such as **input encoding in control panels** and **address decoding in memory systems**.

---

1 / C – Central Idea :-

This unit introduces the **design, function, and logic implementation** of digital **encoders and codebreakers (decoders)**. It includes:

1. **Encoder Circuits**
   - Convert 2^n input lines into an n-bit binary output
   - Example: 8-to-3 encoder (8 inputs → 3-bit output)
   - Priority encoders: handle multiple simultaneous inputs with hierarchy
2. **Decoder (Codebreaker) Circuits**
   - Convert n-bit binary input to 2^n output lines
   - Example: 3-to-8 decoder (3-bit input → 8 outputs)
   - Used for display driving, memory addressing
3. **Truth Tables and Boolean Logic**
   - Construct encoder/decoder truth tables
   - Derive logic expressions for outputs
4. **Practical Applications**
   - Keypad scanning
   - BCD to 7-segment encoding
   - Binary-to-address line decoding

---

1 / D – Performance Objectives :-

By the end of this unit, students will be able to:

Knowledge

- Define encoders and decoders and their roles in digital systems
- Distinguish between standard and priority encoders

Skills

- Draw and explain encoder/decoder truth tables
- Derive Boolean expressions for encoding logic
- Design small encoder/decoder circuits using logic gates

Competence

- Apply encoding/decoding logic in real-world digital systems
- Integrate encoders and decoders into digital input/output subsystems
- Simulate and debug encoder/decoder behavior using software tools

الترميز(Encoding)

الترميز (Encoding) في الدوائر الرقمية

◆ ما هو الترميز؟

الترميز **(Encoding)**هو عملية **تحويل البيانات من شكل إلى آخر** باستخدام **مجموعة محددة من القواعد أو القيم الرقمية** .في الدوائر الرقمية، يتم استخدام الترميز لتحويل البيانات إلى **شكل ثنائي (Binary)**يمكن للأنظمة الرقمية التعامل معه بسهولة.

◆ أهمية الترميز في الإلكترونيات والاتصالات

- يستخدم في **تمثيل البيانات** داخل الحاسوب.
- يسهل **نقل البيانات** في الأنظمة الرقمية والاتصالات.
- يقلل من **عدد الخطوط والأسلاك** في التصميم الرقمي.
- يساعد في **ضغط البيانات** وتحسين الكفاءة.

◆ أنواع الترميز الرقمي(Types of Encoding)

هناك عدة أنواع من الترميز تستخدم في الدوائر الرقمية، ومن أهمها:

1 الترميز الثنائي(Binary Encoding)

- هو أبسط شكل من أشكال الترميز حيث يتم تمثيل البيانات باستخدام **0و 1.**
- كل قيمة رقمية تُعطى تسلسلًا من البتات.(Bits)

✅ **مثال على تمثيل الأرقام العشرية في النظام الثنائي:**

الرقم العشري	التمثيل الثنائي
0	0000
1	0001
2	0010
3	0011
4	0100

2️⃣ ترميز BCD (Binary Coded Decimal)

- يستخدم لتمثيل **الأرقام العشرية** باستخدام **4 بت لكل رقم عشري**.
- هذا الترميز مهم في **الآلات الحاسبة** و**شاشات العرض الرقمية**.

✅ **مثال على BCD:**

الرقم العشري	BCD (4 بت)
0	0000
1	0001
2	0010
3	0011
9	1001

🔹 **تطبيق عملي:**
تُستخدم **شاشات العرض الرقمية (Segment Displays-7)** في الأجهزة الإلكترونية التي تعرض الأرقام باستخدام BCD.

3️⃣ الترميز الرمادي (Gray Code)

- يتميز بأنه **يختلف فقط في بت واحد بين كل رقمين متتالين**.
- يقلل من **أخطاء التغيير المفاجئ للبتات** في الأنظمة الرقمية.
- يُستخدم في **أجهزة الاستشعار الدوارة (Rotary Encoders)**.

✅ **مثال على الترميز الرمادي مقابل الثنائي:**

الرقم العشري	الترميز الثنائي	الترميز الرمادي
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110

🔹 **تطبيق عملي:**
يستخدم في **أنظمة تحديد المواقع** و**الروبوتات** حيث تحتاج أجهزة الاستشعار إلى تقليل الأخطاء في الحسابات.

4️⃣ ترميز ASCII (American Standard Code for Information Interchange)

- يُستخدم لتمثيل **الأحرف والرموز** في الحواسيب.
- يتكون من **7 أو 8 بت** لتمثيل كل حرف أو رمز.

✅ **مثال على ASCII:**

الحرف	ASCII (8 بت)
A	01000001
B	01000010
C	01000011
a	01100001
b	01100010

🔹 **تطبيق عملي:**
يستخدم في **أنظمة تشغيل الحواسيب** والتواصل بين الأجهزة.

5️⃣ ترميز هافمان (Huffman Encoding)

- يُستخدم لضغط البيانات بناءً على **تكرار الرموز**.
- الرموز الأكثر تكرارًا تأخذ **عددًا أقل من البتات**، بينما الرموز الأقل تكرارًا تأخذ **عددًا أكبر من البتات**.
- يقلل حجم البيانات أثناء **الإرسال والتخزين**.

✅ **مثال على ترميز هافمان:** إذا كانت لدينا الحروف التالية مع تكرارها:

- A (50 مرة)
- B (20 مرة)
- C (15 مرة)
- D (10 مرات)
- E (5 مرات)

يتم تخصيص رموز قصيرة للحروف الأكثر تكرارًا، مما يقلل الحجم الكلي للبيانات.

🔹 **تطبيق عملي:**
يُستخدم في **ضغط الملفات** مثل **ملفات ZIP و JPEG**.

الخطوات: 1️⃣ الحرف **A** هو الأكثر تكرارًا، لذا يحصل على **أقصر رمز**.
2️⃣ الحرف **E** هو الأقل تكرارًا، لذا يحصل على **أطول رمز**.
3️⃣ يتم بناء **شجرة هوفمان** بحيث تكون الأحرف الأقل تكرارًا في **المستويات الأعمق**.
4️⃣ يتم تخصيص **رمز ثنائي لكل حرف** بناءً على موقعه في الشجرة.

✅ **النتيجة (رموز هوفمان المحتملة لكل حرف):**

الحرف	عدد التكرارات	الرمز الثنائي (مثال)
A	50	0
B	20	10
C	15	110
D	10	1110
E	5	1111

🎯 **لاحظ:**

- الحرف **A** حصل على **أقصر رمز (0)** لأنه الأكثر تكرارًا.
- الحرف **E** حصل على **أطول رمز (1111)** لأنه الأقل تكرارًا.

🔹 مقارنة بين أنواع الترميز المختلفة

نوع الترميز	الاستخدام الرئيسي	عدد البتات
Binary	تمثيل الأرقام	غير محدد
BCD	شاشات العرض الرقمية	4 بت لكل رقم عشري
Gray Code	أجهزة الاستشعار	يختلف
ASCII	الحروف والرموز في الحواسيب	7 أو 8 بت لكل حرف
Huffman	ضغط البيانات	متغير

◆ الفرق بين الترميز ومفك الشفرات

العنصر	الترميز (Encoder)	مفك الشفرات (Decoder)
الوظيفة	تحويل البيانات إلى شكل مشفر	فك التشفير وإرجاع البيانات الأصلية
عدد المدخلات	$^n2$	n
عدد المخرجات	n	$^n2$
مثال عملي	تحويل الحروف إلى ASCII	تحويل BCD إلى رقم عشري

1. **حوّل الرقم العشري 13 إلى:**
   - النظام **الثنائي**
   - **ترميز BCD**
   - **ترميز Gray Code**
2. إذا كان لدينا الحرف **'Z'**، فما هو **ترميزه في ASCII**؟
3. أكمل الجدول التالي للترميز الرمادي:

الرقم العشري	الترميز الثنائي	الترميز الرمادي
0	0000	?
1	0001	?
2	0010	?
3	0011	?

Post-Test

1. What is the function of a **priority encoder**?
2. Design a **4-to-2 encoder** and write its truth table.
3. How many outputs would a **3-to-8 decoder** have?
4. Given inputs A = 1, B = 0, C = 1 to a 3-to-8 decoder, which output Y is activated?
5. Write the logic expression for A0 in a 4-to-2 encoder.

Key Answers

1. A priority encoder outputs the binary code of the **highest-priority active input**, ignoring others.
2. Truth Table (4-to-2 Encoder):

D3	D2	D1	D0	A1	A0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

3. A 3-to-8 decoder has **8 outputs**.
4. Input A=1, B=0, C=1 → binary 101 = **Y5** is activated.
5. A0 = D1 + D3

Homework Assignments

1. **Design truth tables** for:
   - a) A 3-to-8 decoder
   - b) A 2-to-4 encoder
2. **Write Boolean expressions** for outputs of a 4-to-2 encoder.
3. **Draw the circuit diagram** of a 2-to-4 decoder using AND, OR, and NOT gates.
4. **Real-world application**: Explain how a priority encoder could be used in a fire alarm control system with multiple sensors.
5. **Challenge**:
   - Design a logic system that takes a 4-key input and outputs a 2-bit binary code using **priority encoder logic**.
   - Add an "Invalid" signal output when no key is pressed.

Topic 10: Digital Decoder

1 / A – Target Population :-

This topic is designed for **first-year students** in the **Department of Electronic Techniques** at the **Technological Institute of Basra**. These students are already familiar with binary number systems, encoders, and basic digital circuits, and are now prepared to understand how binary data is **decoded and distributed** using logic circuitry.

---

1 / B – Rationale :-

A **decoder** is an essential component in digital electronics used to **convert binary codes into uniquely activated outputs**. Decoders are widely used in:

- **Memory addressing** and data selection
- **Display driving** (e.g., BCD to 7-segment)
- **Multiplexing and demultiplexing**
- **Instruction decoding** in CPUs and microcontrollers

Learning how to implement and use decoders helps students design systems where **binary input must control multiple output lines**, making them vital for digital design and automation.

---

1 / C – Central Idea :-

This unit introduces the **digital decoder**, which performs the reverse function of an encoder by **translating n-bit binary input into 2^n unique outputs**. Key points covered include:

1. **Definition and Functionality**
    - A digital decoder takes binary input and activates **only one output line** corresponding to the input code.
2. **Types of Decoders**
    - **2-to-4 decoder**: 2-bit input, 4 outputs
    - **3-to-8 decoder**: 3-bit input, 8 outputs
    - **4-to-16 decoder**: 4-bit input, 16 outputs
    - With or without **enable input**
3. **Logic Implementation**
    - Outputs expressed as **AND combinations** of input variables
    - Use of inverters (NOT gates) to handle logic '0'
4. **Truth Tables and Logic Expressions**
    - Example for 2-to-4 decoder:
        - Y0 = A'·B'
        - Y1 = A'·B
        - Y2 = A·B'
        - Y3 = A·B
5. **IC Examples**
    - 74138 (3-to-8 decoder with enable)
    - 7442 (BCD to decimal decoder)
    - 7447 (BCD to 7-segment decoder/driver)

---

1 / D – Performance Objectives :-

After completing this topic, students will be able to:

Knowledge

- Define a digital decoder and describe its function
- Recognize standard decoder types and truth tables

Skills

- Construct logic circuits for 2-to-4 and 3-to-8 decoders
- Write Boolean equations for decoder outputs
- Simulate decoder behavior using digital tools

Competence

- Apply decoders to memory systems, display circuits, and input-output control
- Select appropriate decoder ICs for given logic tasks

- Integrate decoders with microcontroller I/O ports

مفك الشفرات(Decoder)

تعريف مفك الشفرات(Decoder)

مفك الشفرات **(Decoder)**هو دائرة رقمية تأخذ **مدخلات مشفرة** على شكل عدد **ثنائي** وتقوم بتحويلها إلى **خرج غير مشفر**، حيث يتم تنشيط أحد خطوط الخرج بناءً على قيمة المدخل.

- مفك الشفرات يستقبل إدخالًا مكونًا من n بتات، وينتج 2^n مخرجات.
- كل خرج يمثل واحدة من التوليفات الممكنة للمدخلات.

أهمية وفائدة مفك الشفرات

- **تستخدم في وحدات التحكم الرقمية**، مثل الذاكرة والمعالجات.
- **تستخدم في شاشات العرض الرقمية**، مثل **7-segment display**.
- **تستخدم في اختيار خطوط البيانات** في أنظمة الاتصالات ومعالجة الإشارات.

أنواع مفك الشفرات

1. مفك الشفرات الثنائي إلى العشري (Binary to Decimal Decoder)

- **يدخل عدد ثنائي** (مثل 3 بتات).
- **يخرج إشارة واحدة من أصل 8** (لأن $2^3 = 8$)، مما يمثل القيم العشرية من 0 إلى 7.

المدخلات $A_0 A_1 A_2$	المخرجات $Y_0 - Y_7$ (الخط المفعل = 1)
000	$Y_0 = 1$, والباقي 0
001	$Y_1 = 1$, والباقي 0
010	$Y_2 = 1$, والباقي 0
011	$Y_3 = 1$, والباقي 0
100	$Y_4 = 1$, والباقي 0
101	$Y_5 = 1$, والباقي 0
110	$Y_6 = 1$, والباقي 0
111	$Y_7 = 1$, والباقي 0

يستخدم هذا النوع في **تحكم وحدات الذاكرة**، حيث يتم تحديد أي موقع ذاكرة سيتم تفعيله.

2.مفك الشفرات 2 إلى 4(2-to-4 Decoder)

يأخذ **مدخلين ثنائيين** وينتج **4مخرجات**، حيث يتم تفعيل أحدها فقط بناءً على قيمة المدخل.

- **جدول الحقيقة لمفك الشفرات 2-إلى-4:**

$_0Y$	$_1Y$	$_2Y$	$_3Y$	$_0A$	$_1A$
1	0	0	0	0	0
0	1	0	0	1	0
0	0	1	0	0	1
0	0	0	1	1	1

✍️ تطبيق:

يستخدم في **أنظمة التوجيه الرقمية** و **تحكم المعالجات في اختيار الأجهزة الطرفية**.

3.مفك الشفرات 3 إلى 8(3-to-8 Decoder)

- يحتوي على **3مدخلات**.
- يحتوي على **8مخرجات**.
- يتم تفعيل مخرج واحد فقط لكل إدخال.

- **جدول الحقيقة لمفك الشفرات 3-إلى-8:**

$_0Y$	$_1Y$	$_2Y$	$_3Y$	$_4Y$	$_5Y$	$_6Y$	$_7Y$	$_0A$	$_1A$	$_2A$
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	1	1	0
0	0	0	0	1	0	0	0	0	0	1
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	0	1	1
0	0	0	0	0	0	0	1	1	1	1

✍️ تطبيق:

يستخدم في **شاشات العرض الرقمية** لتفعيل رقم معين على **Segment Display-7**.

أسئلة حول جدول الحقيقة لمفك الشفرات(Decoder Truth Table Questions)

إذا كانت المدخلات $A_1 = 1$، $A_0 = 0$، فما هو المخرج الذي سيتم **تفعيله (1)**؟
(اختر الإجابة الصحيحة):

1. Y_3
2. Y_2
3. Y_1
4. Y_0

باستخدام جدول الحقيقة لمفك الشفرات **2 إلى 4**، ما هو خرج الدائرة إذا كان المدخل $A_1A_0 = 11_2$؟
(اختر الإجابة الصحيحة):

1. $Y_3 = 1$
2. $Y_2 = 1$
3. $Y_1 = 1$
4. $Y_0 = 1$

أكمل جدول الحقيقة التالي لمفك شفرات **3 إلى 8**:

Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	A_0	A_1	A_2
?	?	?	?	?	?	?	?	0	0	0
?	?	?	?	?	?	?	?	1	0	0
?	?	?	?	?	?	?	?	0	1	0
?	?	?	?	?	?	?	?	1	1	0
?	?	?	?	?	?	?	?	0	0	1
?	?	?	?	?	?	?	?	1	0	1
?	?	?	?	?	?	?	?	0	1	1
?	?	?	?	?	?	?	?	1	1	1

🔹 **سؤال متابعة:**

- إذا كان المدخل $A_2A_1A_0 = 101_2$، فما هو المخرج الذي سيتم **تفعيله**؟

Post-Test

1. What is the purpose of a **digital decoder** in a digital system?
2. Complete the truth table for a **3-to-8 decoder**.
3. Write the Boolean expressions for the outputs of a 2-to-4 decoder.
4. What happens when the **enable input** of a decoder is low (0)?
5. How many outputs would a 4-to-16 decoder have?

Key Answers

1. A decoder activates a **single output line** based on binary input, useful in **addressing or selecting** functions.
2. Truth table for 3-to-8 decoder (input A2A1A0):

A2	A1	A0	Y0–Y7 Active
0	0	0	Y0
0	0	1	Y1
0	1	0	Y2
0	1	1	Y3
1	0	0	Y4

A2	A1	A0	Y0–Y7 Active
1	0	1	Y5
1	1	0	Y6
1	1	1	Y7

3. From earlier:
    - Y0 = A1'·A0'
    - Y1 = A1'·A0
    - Y2 = A1·A0'
    - Y3 = A1·A0
4. Decoder becomes **active** (outputs work as intended) when **Enable = 0**, depending on design
5. A 4-to-16 decoder has **16 outputs**

Homework Assignments

1. **Design a 3-to-8 decoder** using logic gates and draw the full diagram.
2. **Derive the logic equations** for all outputs of a 3-to-8 decoder.
3. **Explain** how a 4-to-16 decoder could be used to select one of 16 memory blocks.
4. **Simulation Exercise**: Build a 2-to-4 decoder in Logisim and verify outputs.
5. **Advanced**: Add an **enable pin** to a 3-to-8 decoder design and describe its behavior in active and inactive states.

Topic 11: Adders and Subtractors

1 / A – Target Population :-

This topic is intended for **first-year students** in the **Department of Electronic Techniques** at the **Technological Institute of Basra**. These students have a foundation in binary number systems, logic gates, and flip-flops, and are now ready to learn how digital circuits perform **arithmetic operations**, particularly **binary addition and subtraction**.

---

1 / B – Rationale :-

Addition and subtraction are **core functions** in digital electronics. They are essential for:

- Arithmetic Logic Units (ALUs)
- Microprocessor design
- Signal processing
- Memory address manipulation
- Digital counters and accumulators

Understanding how **adders and subtractors** work prepares students to design and troubleshoot complex **arithmetic circuits**, which are the foundation of computing.

---

1 / C – Central Idea :-

This topic introduces the design, logic, and implementation of digital circuits that perform **binary addition and subtraction**, including:

1. **Half-Adder and Full-Adder**
    - Half-Adder: Adds 2 bits (A + B)
    - Full-Adder: Adds A + B + Carry-in (C-in)
2. **Binary Subtraction using 2's Complement**
    - Subtraction as A + (–B)
    - Invert B and add 1, then perform addition
3. **Combinational Circuit Design**
    - Truth tables and Boolean expressions for sum and carry
    - Logic diagrams using AND, OR, XOR gates
4. **Integrated Circuits (ICs)**
    - IC 7483: 4-bit full adder
    - Circuit expansion: Multi-bit adder/subtractor designs
5. **Practical Applications**
    - Used in calculators, processors, embedded systems
    - Fundamental for signed/unsigned arithmetic

---

1 / D – Performance Objectives :-

By the end of this unit, students will be able to:

Knowledge

- Define half-adder and full-adder circuits
- Explain how subtraction can be implemented using addition logic

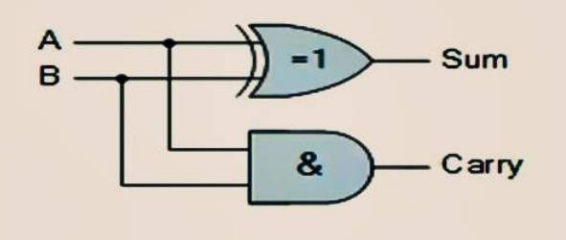
Skills

- Design 1-bit and 4-bit adder/subtractor circuits
- Derive Boolean expressions for sum and carry
- Implement subtraction using 2’s complement

Competence

- Build and simulate multi-bit adders/subtractors
- Apply arithmetic logic in digital systems
- Evaluate and troubleshoot logic-level arithmetic errors

دائرة نصف الجامع(Half Adder)



دائرة **نصف الجامع (Half Adder)** هي إحدى أبسط الدوائر المنطقية في الإلكترونيات الرقمية، وتستخدم لإجراء عملية جمع رقمين ثنائيين (Bitwise Addition) دون الأخذ في الاعتبار أي قيمة حمل من عملية سابقة.

هذه الدائرة تتعامل مع إدخالين A) و (B وتمتلك خرجين:

- **المجموع :(Sum)** يمثل ناتج الجمع البسيط.
- **الحمل :(Carry)** يمثل الحمل الناتج الذي يتم تمريره إلى المرحلة التالية في حالة الجمع المتعدد البتات.

جدول الحقيقة لدائرة نصف الجامع

Carry (C)	Sum (S)	B	A
0	0	0	0
0	1	1	0
0	1	0	1
1	0	1	1

التعبير المنطقي لنصف الجامع

يعتمد نصف الجامع على بوابتين منطقيتين أساسيتين:

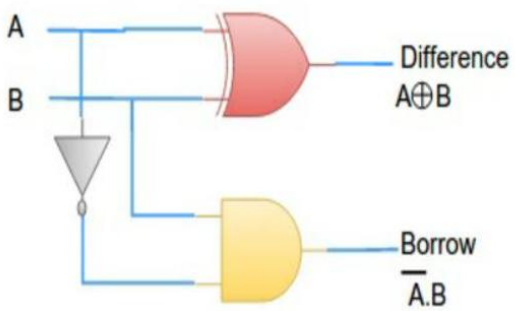
- **بوابة XOR** لحساب المجموع.(S)
- **بوابة AND** لحساب الحمل.(C)

يمكن كتابة المعادلات المنطقية كما يلي:

$$B \oplus A = S$$

$$B \cdot A = C$$

دائرة نصف الطارح(Half Subtractor)



دائرة **نصف الطارح (Half Subtractor)** هي دائرة منطقية تُستخدم لحساب **الطرح الثنائي (Binary Subtraction)**بين بتين A) و B) دون الأخذ في الاعتبار أي اقتراض (Borrow) من العمليات السابقة. وهي مشابهة لنصف الجامع ولكنها تتعامل مع عملية الطرح بدلاً من الجمع.

تمتلك هذه الدائرة:

- **خرج الفرق (Difference - D)**: وهو ناتج طرح البتات.

- **خرج الاقتراض :(Borrow - B_out)** وهو الإشارة التي تدل على الحاجة إلى الاقتراض في حالة كان المطروح (B) أكبر من المطروح منه.(A)

جدول الحقيقة لدائرة نصف الطارح:

Borrow (B_out)	Difference (D)	B	A
0	0	0	0
1	1	1	0
0	1	0	1
0	0	1	1

المعادلات المنطقية لنصف الطارح

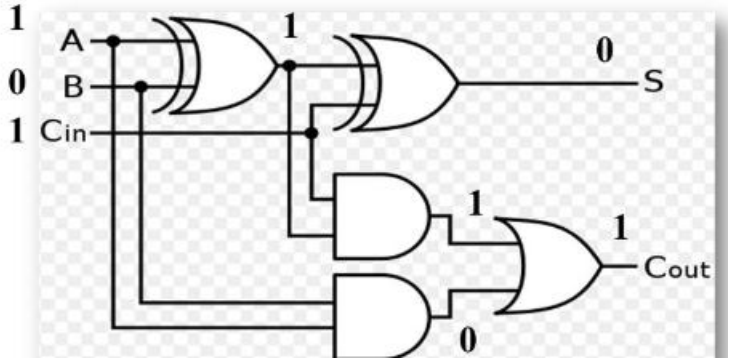
- الفرق (Difference) يتم حسابه باستخدام بوابة XOR:

$$B \oplus A = D$$

- الاقتراض (Borrow) يتم حسابه باستخدام بوابة **NOT AND (A'B)**:

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

دائرة الجامع التام(Full Adder)



دائرة **الجامع التام (Full Adder)** هي دائرة منطقية تستخدم لجمع **ثلاثة بتات ثنائية** بدلاً من اثنين فقط كما هو الحال في **نصف الجامع** .هذه الدائرة تأخذ في الاعتبار حمل الإدخال(**Carry In - Cin**) ، مما يجعلها أساسية في بناء المجمعات متعددة البتات مثل وحدات الجمع في المعالجات الرقمية.

المدخلات والمخرجات:

- **المدخلات:**
  - **A** : أول رقم ثنائي.
  - **B** : ثاني رقم ثنائي.
  - **Cin** : حمل الإدخال القادم من المرحلة السابقة.
- **المخرجات:**
  - **Sum (S)** : ناتج الجمع.
  - **Carry Out (Cout)** : ناتج الحمل الذي يتم تمريره إلى المرحلة التالية.

جدول الحقيقة لدائرة الجامع التام

Carry Out (Cout)	Sum (S)	Cin	B	A
0	0	0	0	0
0	1	1	0	0
0	1	0	1	0
1	0	1	1	0
0	1	0	0	1
1	0	1	0	1
1	0	0	1	1
1	1	1	1	1

المعادلات المنطقية لدائرة الجامع التام

يمكننا تحليل دائرة الجامع التام إلى معادلتين منطقيتين:

- المجموع (Sum)

$$Cin \oplus B \oplus A = S$$

- الحمل الخارج (Cout)

$$((B \oplus A) \cdot Cin) + (B \cdot A) = Cout$$

مثال عملي لجمع عددين من 4 بت باستخدام جامع تام

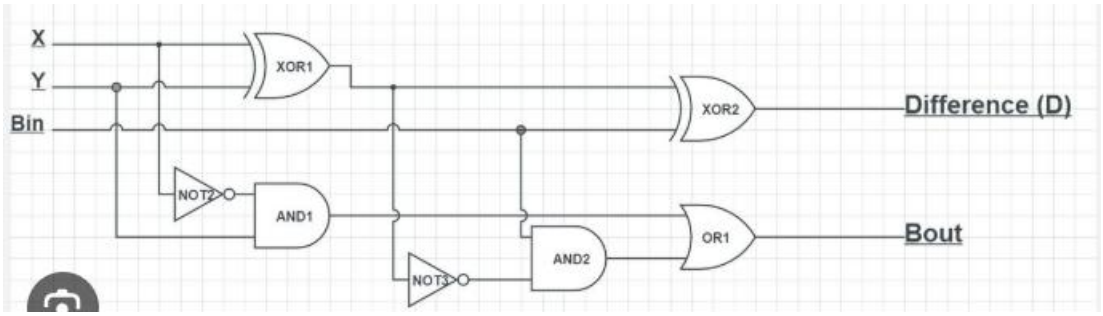
جمع العددين 1011_2 و 1101_2

Carry Out	Sum	Carry In	B	A
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	1	1	1

✅ النتيجة:

$$1011_2 + 1101_2 = 11000_2$$

دائرة الطارح التام(Full Subtractor)



دائرة **الطارح التام (Full Subtractor)** هي دائرة منطقية تُستخدم لتنفيذ **عملية الطرح الثنائية** بين ثلاثة مدخلات، مع الأخذ في الاعتبار **الاقتراض الداخلي (Borrow In - Bin)** من العملية السابقة. وهي تعد التطوير الأساسي لدائرة **نصف الطارح.(Half Subtractor)**

المدخلات والمخرجات:

- **المدخلات:**
  - **A** : المطروح منه.(Minuend)
  - **B** : المطروح.(Subtrahend)
  - **Bin** : الاقتراض الداخل من العملية السابقة.
- **المخرجات:**
  - **Difference (D)** : ناتج الطرح.
  - **Borrow Out (Bout)** : إشارة الاقتراض إلى المرحلة التالية.

جدول الحقيقة لدائرة الطارح التام

Borrow Out (Bout)	Difference (D)	Bin	B	A
0	0	0	0	0
1	1	1	0	0
1	1	0	1	0
1	0	1	1	0
0	1	0	0	1
0	0	1	0	1
0	0	0	1	1
1	1	1	1	1

المعادلات المنطقية لدائرة الطارح التام

- **الفرق (Difference)**

$$Bin \oplus B \oplus A = D$$

- **الاقتراض الخارج (Borrow Out)**

$$(Bin \cdot \overline{(A \oplus B)}) + (B \cdot \overline{A}) = Bout$$

مثال عملي لطرح عددين من 4 بت باستخدام الطارح التام

طرح العدد 1011_2 من 1101_2

Borrow Out	Difference	Borrow In	B	A
0	0	0	1	1
0	1	0	0	1
1	1	0	1	0
1	1	1	1	1

✅ **النتيجة:**

$$1101_2 - 1011_2 = 0010_2$$

Post-Test

1. Write the **truth table for a half-adder**.
2. Derive the **Boolean expressions** for the **sum and carry** of a full-adder.
3. How do you perform **subtraction using 2's complement**?
4. What is the result of adding A = 1010 and B = 0110 using binary addition?
5. Implement a **4-bit adder** using full-adders. How many full-adders are needed?

Key Answers

1. Half-Adder Truth Table (see above).
2. Full-Adder:
   - Sum = $A \oplus B \oplus Cin$
   - Carry = $A \cdot B + Cin \cdot (A \oplus B)$
3. Subtraction via 2's complement:
   - Invert B
   - Add 1
   - Add result to A
4. 1010 + 0110 = 10000 (in 5-bit output, result = 0000 with carry = 1)
5. A 4-bit adder uses **4 full-adders** in cascade (each handles one bit)

Homework Assignments

1. **Design a full-adder circuit** using logic gates and draw the complete diagram.
2. **Implement binary subtraction** using 2's complement for:
   - a) 1010 – 0011
   - b) 0111 – 0100
3. **Draw a block diagram** for a 4-bit adder-subtractor circuit using full-adders and XOR gates.

4. **Challenge**: Simulate an 8-bit adder-subtractor circuit in Logisim and test with signed numbers.
5. **Explain** how overflow is detected in binary addition using the carry out and sign bit

Topic 12: Flip-Flop Circuits

1 / A – Target Population :-

This topic is designed for **first-year students** in the **Department of Electronic Techniques** at the **Technological Institute of Basra**, who have already studied combinational logic circuits. These students are now ready to explore **sequential logic**, beginning with the fundamental unit: the **flip-flop**.

---

1 / B – Rationale :-

While combinational circuits produce outputs based only on current inputs, **flip-flops introduce memory** into digital systems. Flip-flops are essential for:

- **Data storage** (1-bit memory)
- **Clocked operations** in CPUs and microcontrollers
- **Counters, registers, and shift registers**
- **Timing and synchronization** in digital systems

Understanding flip-flops enables students to build systems that **remember, store, and toggle data**, paving the way to sequential logic analysis and design.

---

1 / C – Central Idea :-

Flip-flops are **bistable devices** that store a single bit of binary data. This topic introduces:

1. **Types of Flip-Flops**
    - **SR (Set-Reset)**

    - **JK (Jack-Kilby)**
    - **D (Data or Delay)**
    - **T (Toggle)**
2. **Inputs, Outputs, and Truth Tables**
    - Inputs: clock signal, data/control bits
    - Outputs: Q and $\overline{Q}$
3. **Clocked Behavior**
    - Edge-triggered flip-flops (rising/falling edge)
    - Master-slave configuration
4. **Applications**
    - Latching data
    - Edge detection
    - Frequency division
    - Counter and timer designs

---

1 / D – Performance Objectives :-

By the end of this topic, students will be able to:

Knowledge

- Identify types of flip-flops and their functions
- Understand truth tables and timing diagrams

Skills

- Design circuits using SR, D, JK, and T flip-flops
- Analyze timing diagrams and predict flip-flop behavior
- Build sequential logic using flip-flops

Competence

- Apply flip-flops in counter/register design
- Integrate memory behavior in digital systems
- Simulate and test sequential circuits using flip-flops

المراجيح **(Flip-Flops)**هي **دوائر منطقية ثنائية الحالة** تُستخدم لتخزين **بت واحد من البيانات** .تعتبر هذه الدوائر من أساسيات **المنطق التتابعي(Sequential Logic)** ، حيث تعتمد حالتها الحالية على مدخلاتها **وعلى حالتها السابقة**.

- **الاستخدامات الشائعة للمراجيح**:

- تخزين البيانات في **الذاكرة العشوائية.(RAM)**
- بناء **مسجلات الإزاحة.(Shift Registers)**
- توليد النبضات الزمنية. **(Clock Generation)**
- بناء **عدادات رقمية.(Counters)**

- **الفرق بين المنطق التوافقي والتتابعي**:

- **المنطق التوافقي (Combinational Logic)** يعتمد فقط على المدخلات الحالية.
- **المنطق التتابعي (Sequential Logic)** يعتمد على المدخلات والحالة السابقة للدائرة.

أنواع المراجيح(Flip-Flops)

هناك عدة أنواع من المراجيح، وكل منها لها خصائصها ووظيفتها الخاصة. أهم هذه الأنواع:

نوع المرجوحة	الرمز المختصر	المعادلة الأساسية	الاستخدامات
مرجوحة SR (Set-Reset)	SR Flip-Flop	**تعيين (Set)** → S = 1 **إعادة تعيين (Reset)** → R = 1	تخزين البيانات، التحكم في الإشارات
مرجوحة D (Data Flip-Flop)	D Flip-Flop	Q(next) = D	مسجلات الإزاحة، الذاكرة
مرجوحة JK	JK Flip-Flop	J=1, K=0 → Set J=0, K=1 → Reset J=1, K=1 → Toggle	العدادات، أنظمة التحكم
مرجوحة T (Toggle Flip-Flop)	T Flip-Flop	يتغير T=1 → Q	العدادات الرقمية

شرح تفصيلي لكل نوع من المراجيح

🔹 3.1 مرجوحة SR (Set-Reset Flip-Flop)

📌 **تعريف:**

هي أبسط أنواع المراجيح، تحتوي على **مدخلين رئيسيين**:

Set (S): عند 1، يتم **ضبط** (Set) خرج المرجوحة إلى 1.

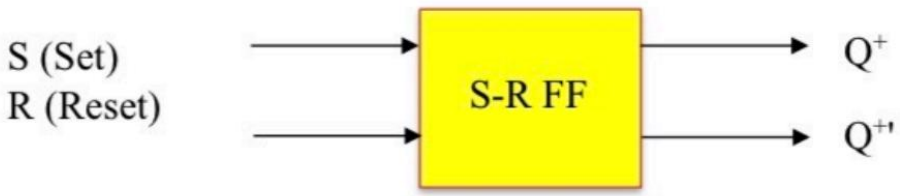
Reset (R): عند 1، يتم **إعادة الضبط** (Reset) للخرج إلى 0.

📌 **جدول الحقيقة لمرجوحة SR**

S	R	Q (الحالة الجديدة)	Q' (المتممة)
0	0	Q (الحالة السابقة)	Q' (الحالة السابقة)
0	1	0	1
1	0	1	0
1	1	غير محدد	غير محدد

📌 **العيب الرئيسي لمرجوحة SR:**

عند S = 1 و R = 1، تكون الحالة **غير معرفة** مما يجعلها غير مستقرة.



Q_0: is the previous output.

Q^+: is the next output.

🔹 3.2 مرجوحة D (Data Flip-Flop)

📌 **تعريف:**

تُعرف أيضًا باسم **Delay Flip-Flop** لأنها تؤخر قيمة المدخل حتى النبضة التالية. تحتوي على:

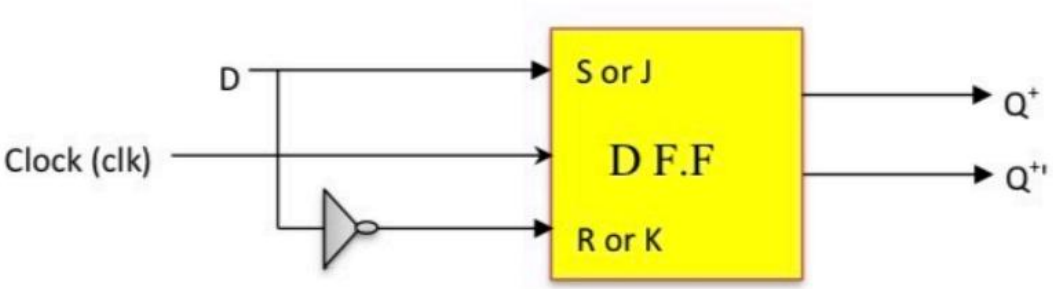
- **مدخل D (Data)**: يحدد قيمة الخرج عند النبضة القادمة.
- **نبضة الساعة (Clock - CLK)**: يتم تحديث الحالة فقط عند النبضة الفعالة.

📌 **جدول الحقيقة لمرجوحة D**

D	CLK	Q (الحالة الجديدة)
0	↑	0
1	↑	1

📌 **الميزة الأساسية:**

- تمنع الحالات غير المحددة الموجودة في مرجوحة **SR**.
- تستخدم على نطاق واسع في **المسجلات والذاكرة**.



🔹 3.3 مرجوحة JK (JK Flip-Flop)

📌 **تعريف:**

تحل مشكلة **الحالة غير المحددة** في مرجوحة SR، حيث:

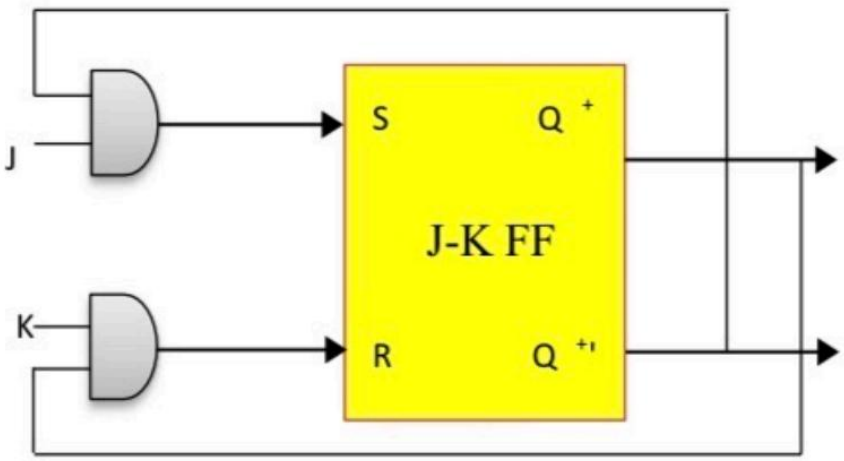
- عندما يكون J = 1 و K = 1، يتغير الخرج (Toggle).

📌 **جدول الحقيقة لمرجوحة JK**

J	K	Q (الحالة الجديدة)
0	0	Q (الحالة السابقة)
0	1	0
1	0	1
1	1	عكس الحالة السابقة

📌 **الميزة الرئيسية:**

- أكثر استقرارًا من SR.
- يستخدم في **العدادات الرقمية** وأنظمة التحكم.



🔹 3.4 مرجوحة T (Toggle Flip-Flop)

📌 **تعريف:**

تعتمد على مبدأ **التبديل (Toggle)**، حيث:

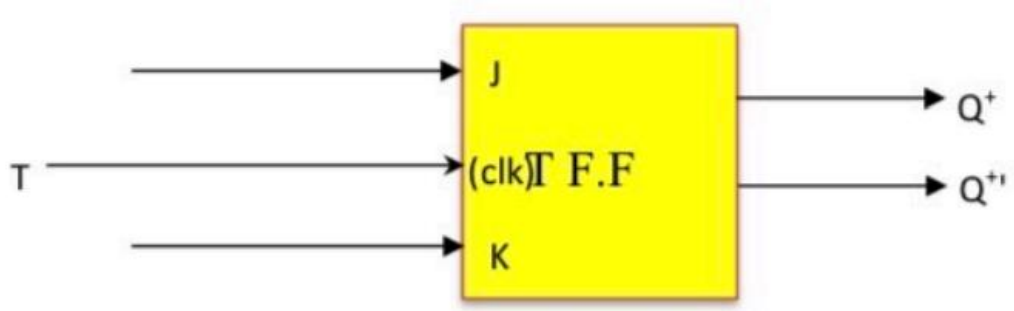
- إذا كان T = 0، تبقى الحالة كما هي.
- إذا كان T = 1، يتم **عكس الحالة السابقة**.

📌 **جدول الحقيقة لمرجوحة T**

Q (الحالة الجديدة)	CLK	T
Q (الحالة السابقة)	↑	0
عكس الحالة السابقة	↑	1

📌 **الاستخدامات:**

- تستخدم في **العدادات الرقمية**.
- مفيدة لإنشاء **تقسيم التردد** (Frequency Divider).



Post-Test

1. What is the function of a flip-flop in digital circuits?
2. Write the truth table for a **D flip-flop**.
3. Which flip-flop is best used in **binary counters**?
4. What happens to Q in an **SR flip-flop** if S=R=1?
5. Describe the difference between a **D flip-flop** and a **T flip-flop**.

Key Answers

1. A flip-flop **stores 1 bit** and can remember a logic state based on control input and clock.
2. D Flip-Flop Truth Table:

Clock	D	Q(next)
↑	0	0
↑	1	1

3. **T flip-flop** is ideal for binary counters.
4. **S=R=1** is an **invalid state** in an SR flip-flop.
5. D flip-flop **copies input to output** on clock edge; T flip-flop **toggles** output when T=1.

Homework Assignments

1. **Draw logic symbols and truth tables** for all four basic flip-flops.
2. **Simulate** a T flip-flop using a JK flip-flop by wiring J=K=1.
3. **Design a 2-bit counter** using T flip-flops.
4. **Explain** why D flip-flops are widely used in registers.
5. **Timing Diagram Exercise**: Predict Q output for a D flip-flop over 5 clock cycles for a given D input stream.

Topic 13: **Counters**

1 / A – Target Population :-

This lesson is designed for **first-year students** in the **Department of Electronic Techniques** at the **Technological Institute of Basra**, who have already covered flip-flops and basic sequential circuits. They are now ready to understand how **counters** are built using flip-flops and how they function in timing and control applications.

---

1 / B – Rationale :-

Counters are sequential logic circuits that count pulses or events. They are widely used in:

- **Timers and digital clocks**
- **Event counters in processors**
- **Frequency division**
- **Step-by-step control systems**
- **Memory address sequencing**

Understanding counters is fundamental to designing systems with **state progression**, **data indexing**, or **synchronized timing**.

---

1 / C – Central Idea :-

This unit introduces **binary and BCD counters** built from flip-flops. It covers:

1. **Definition and Classification**
    - Synchronous vs. Asynchronous counters

    - Up-counters, down-counters, and up/down counters
    - Binary vs. decimal (BCD) counters
2. **Asynchronous (Ripple) Counters**
    - Output of one flip-flop feeds the next clock input
    - Simple but suffers from propagation delay
3. **Synchronous Counters**
    - All flip-flops clocked together
    - Faster, less glitch-prone
4. **Modulo Counters**
    - Count from 0 to N–1 (mod-N)
    - Example: Mod-10 BCD counter
5. **Logic Design**
    - Truth tables and state diagrams
    - T and JK flip-flop based designs
    - Reset and enable control lines
6. **Common ICs**
    - 7490: Decade counter
    - 74193: 4-bit up/down counter
    - 4017: Johnson counter (10-output)

---

1 / D – Performance Objectives :-

By the end of this topic, students will be able to:

Knowledge

- Define the function and types of digital counters
- Identify differences between synchronous and asynchronous designs

Skills

- Design a 3-bit and 4-bit binary counter
- Create timing diagrams and state sequences
- Implement BCD counters and understand carry-out behavior

Competence

- Apply counters in control, timing, and memory applications
- Simulate counters using logic software
- Modify counters for custom modulus counting (e.g., Mod-6, Mod-12)

العدادات الرقمية(Counters)

العداد **(Counter)**هو **دائرة منطقية متتابعة** تستخدم لتتبع عدد النبضات الزمنية (Clock Pulses) في **نظام رقمي** .يتكون العداد من مجموعة من **المراجيح (Flip-Flops)** المرتبطة ببعضها لتشكيل عدد من الحالات الثنائية.

🔹 استخدامات العدادات الرقمية:

✅ **حساب النبضات الزمنية** في الدوائر الرقمية.
✅ **قياس الزمن** والفواصل الزمنية في الأنظمة المضمنة.
✅ **تقسيم الترددات** في مولدات الإشارات.
✅ **توليد الأنماط الرقمية** في الأنظمة المضمنة.
✅ **إنشاء مؤقتات (Timers)** في أجهزة الحاسوب والأنظمة المدمجة

أنواع العدادات الرقمية

يمكن تصنيف العدادات بناءً على عدة عوامل، مثل اتجاه العد، طريقة تشغيلها، وطريقة التوصيل بين مراحلها.

🔹 (1) العدادات المتزامنة وغير المتزامنة

النوع	طريقة العمل	المزايا	العيوب
العداد غير المتزامن (Asynchronous Counter)	كل مرجوحة تعتمد على خرج المرجوحة السابقة، مما يؤدي إلى تأخير زمني بين المراحل.	بسيط التصميم، يحتاج عدد أقل من البوابات.	بطيء بسبب تأخير الانتشار بين المراحل.
العداد المتزامن (Synchronous Counter)	كل المراجيح تستقبل نبضة الساعة نفسها، مما يجعل التغيير يحدث في نفس الوقت.	أسرع، لا يوجد تأخير زمني بين المراحل.	أكثر تعقيدًا في التصميم.

🔹 (2) العدادات الصاعدة والهابطة

النوع	طريقة العد
العداد الصاعد (Up Counter)	يعد تصاعديًا من 0 → 1 → 2 → 3 → ...
العداد الهابط (Down Counter)	يعد تنازليًا من ... → 3 → 2 → 1 → 0
العداد القابل للعكس (Up/Down Counter)	يمكنه العد تصاعديًا أو تنازليًا باستخدام إشارة تحكم.

🔹 (3) العدادات ذات الأساس العشري والثنائي

النوع	عدد الحالات
العداد الثنائي (Binary Counter)	يعد باستخدام النظام الثنائي، أي من 0000 → 0001 → 0010 → ...
العداد العشري (BCD Counter - Decimal Counter)	يعد من 0000 إلى 1001 (0 إلى 9)، ثم يعود إلى الصفر.

🔹 (1) عداد غير متزامن 4-بت (Asynchronous 4-bit Counter)

- يتكون من 4 مراجيح T Flip-Flop متصلة بحيث يعمل كل منها كمقسم للتردد.
- كل مرجوحة تتغير حالتها عند **الحافة الهابطة (Falling Edge)** للساعة القادمة من المرجوحة السابقة.

🔹 **جدول الحقيقة لعداد غير متزامن 4-بت:**

Q0 (LSB)	Q1	Q2	Q3 (MSB)	Clock Pulse
0	0	0	0	0
1	0	0	0	1
0	1	0	0	2
1	1	0	0	3
0	0	1	0	4
1	0	1	0	5
0	1	1	0	6
1	1	1	0	7
0	0	0	1	8

🔹 جدول الحقيقة لعداد متزامن 4-بت (Synchronous 4-bit Counter)

- **عداد متزامن** يعني أن جميع المراجيح (Flip-Flops) تتغير حالتها في نفس الوقت عند نبضة الساعة (Clock Pulse).
- يبدأ العداد من 0000 ويزيد بمقدار 1 عند كل نبضة ساعة.

Q0 (LSB)	Q1	Q2	Q3 (MSB)	نبضة الساعة (Clock Pulse)
0	0	0	0	0
1	0	0	0	1
0	1	0	0	2
1	1	0	0	3
0	0	1	0	4
1	0	1	0	5
0	1	1	0	6
1	1	1	0	7
0	0	0	1	8
1	0	0	1	9
0	1	0	1	10
1	1	0	1	11
0	0	1	1	12
1	0	1	1	13

🔹 (3) عداد BCD (عشري)

- يعد من 0 إلى 9 ثم يعيد نفسه.
- يستخدم في شاشات العرض الرقمية (7-Segment Display).

🔹 جدول الحقيقة لعداد BCD:

Q0 (LSB)	Q1	Q2	Q3 (MSB)	Clock Pulse
0	0	0	0	0
1	0	0	0	1
0	1	0	0	2
1	1	0	0	3
0	0	1	0	4
1	0	1	0	5
0	1	1	0	6
1	1	1	0	7
0	0	0	1	8
1	0	0	1	9
0	0	0	0	10 (إعادة)

سجلات الإزاحة(Shift Registers)

سجلات الإزاحة (Shift Registers) هي **دوائر رقمية متتابعة** تتكون من مجموعة من **المراجيح (Flip-Flops)**متصلة معًا بحيث يمكنها **تخزين ونقل البيانات** داخل النظام الرقمي. يتم نقل البيانات داخل السجل عبر كل نبضة ساعة **(Clock Pulse)**، مما يسمح بتحريك البيانات إما إلى اليمين أو إلى اليسار.

🔹 استخدامات سجلات الإزاحة

✅ تخزين البيانات مؤقتًا في **الأنظمة الرقمية**.
✅ تحويل البيانات من **التوازي إلى التسلسل والعكس**.
✅ توليد **أنماط البيانات** في الدوائر الرقمية.
✅ تنفيذ العمليات الحسابية **كالضرب والقسمة**.
✅ **توليد إشارات التأخير** في الأنظمة الرقمية.

أنواع سجلات الإزاحة

يمكن تصنيف سجلات الإزاحة وفقًا **لاتجاه حركة البيانات** و**طريقة الإدخال والإخراج**.

نوع السجل	اختصار الاسم	اتجاه الإزاحة	الوصف
السجل بإزاحة يمينية (Serial-In, Serial-Out - SISO)	SISO	➡️ اليمين فقط	يتم إدخال البيانات وإخراجها بشكل متسلسل، ويستخدم في تحويل البيانات بين الأنظمة التسلسلية.
السجل بإزاحة يسارية (Serial-In, Serial-Out - SISO)	SISO	⬅️ اليسار فقط	مشابه لسابقه ولكن البيانات تتحرك نحو اليسار.
السجل بإزاحة يمينية مع إدخال متوازي (Parallel-In, Serial-Out - PISO)	PISO	إدخال متوازي ➡️ إخراج متسلسل	يتم إدخال البيانات دفعة واحدة وإخراجها **بتًا تلو الآخر**.
السجل بإزاحة يسارية مع إدخال متوازي (Parallel-In, Serial-Out - PISO)	PISO	إدخال متوازي ⬅️ إخراج متسلسل	يعمل بنفس الطريقة ولكن بتحريك البيانات لليسار.
السجل بإزاحة في الاتجاهين (Bidirectional Shift Register)	BIDIR	اليمين ➡️⬅️ واليسار	يمكن تحريك البيانات في الاتجاهين، ويتم التحكم باتجاه الإزاحة بإشارة خارجية.
السجل بإزاحة كاملة (Universal Shift Register)	USR	إدخال ➡️⬅️ وإخراج من	يسمح بالإدخال والإخراج بنمط متسلسل أو متوازي في أي اتجاه.

📌 **جدول الحقيقة لـ SISO (إزاحة يمينية)**

Clock	المدخل (In)	Q3 (MSB)	Q2	Q1	Q0 (LSB)	الخروج (Out)
0	1	0	0	0	0	0
1	0	1	0	0	0	0
2	1	0	1	0	0	0
3	1	1	0	1	0	0
4	0	1	1	0	1	1

- **يتم إدخال البيانات بتًا واحدًا في كل نبضة ساعة، وتنتقل إلى اليمين تدريجيًا حتى تصل إلى الخرج.**

(2)سجل إزاحة تسلسلي مع إدخال متوازي(PISO - Parallel-In Serial-Out)

- يتم تحميل البيانات دفعة واحدة ثم إخراجها بشكل متسلسل.
- يستخدم عند الحاجة إلى **إرسال بيانات متوازية عبر خطوط تسلسلية**.

📌 جدول الحقيقة لـ PISO

الخروج (Out)	Q0 (LSB)	Q1	Q2	Q3 (MSB)	تحميل البيانات (Load Enable)	Clock
0	1	1	0	1	1	0
1	1	1	0	1	0	1
1	1	1	1	0	0	2
1	1	1	1	1	0	3
1	1	1	1	1	0	4

- عند تحميل البيانات (`Load Enable = 1`) يتم تخزين القيمة مباشرة في السجل.
- بعد ذلك، تبدأ البيانات بالخروج بتًا تلو الآخر عند كل نبضة ساعة.

(3)سجل إزاحة ثنائي الاتجاه(Bidirectional Shift Register)

- يسمح بتحريك البيانات **إلى اليمين أو اليسار**.
- يحتوي على **إشارة تحكم تحدد الاتجاه**.

📌 جدول الحقيقة لسجل إزاحة ثنائي الاتجاه

Q0	Q1	Q2	Q3	اتجاه الإزاحة (Direction)	Clock
1	0	0	0	0 (إزاحة يمين)	0
0	1	0	0	0	1
0	0	1	0	1 (إزاحة يسار)	2
0	0	0	1	1	3

◆ أسئلة إكمال جدول الحقيقة

1. أكمل القيم الناقصة في جدول الحقيقة لسجل الإزاحة المتسلسل بإزاحة يمينية (SISO):

الخروج (Out)	Q0 (LSB)	Q1	Q2	Q3 (MSB)	المدخل (In)	Clock
?	0	0	0	0	1	0
?	0	0	0	1	0	1
?	0	0	1	?	1	2
?	0	1	?	?	1	3
1	1	?	?	1	0	4

2. أكمل القيم الناقصة في جدول الحقيقة لسجل الإزاحة المتوازي إلى التسلسلي (PISO):

الخروج (Out)	Q0 (LSB)	Q1	Q2	Q3 (MSB)	تحميل البيانات (Load Enable)	Clock
?	1	1	0	1	1	0
?	1	1	0	1	0	1
?	1	1	1	?	0	2
1	1	1	?	1	0	3
1	1	?	1	1	0	4

Post-Test

1. What is the difference between synchronous and asynchronous counters?
2. Draw the state table for a **3-bit binary counter**.
3. How many states does a **Mod-6 counter** have?
4. What happens in a BCD counter after the state **1001**?
5. Design a **4-bit synchronous up-counter** using T flip-flops.

Key Answers

1. **Asynchronous**: flip-flops clocked in sequence → slower, delays.
   Synchronous: all flip-flops clocked together → faster, cleaner transitions.
2. See above table for 3-bit counter (states 0–7).
3. A Mod-6 counter has **6 states** (000 to 101).
4. It resets to **0000** after 1001 (decimal 9).
5. T flip-flop design (as per logic for T inputs):
   - T0 = 1

- T1 = Q0
- T2 = Q0·Q1
- T3 = Q0·Q1·Q2

Homework Assignments

1. **Draw timing diagrams** for a 3-bit asynchronous and synchronous counter.
2. **Design a Mod-5 counter** using JK flip-flops and draw its state diagram.
3. **Simulate a 4-bit binary counter** in Logisim and verify output.
4. **Challenge**: Add a reset and enable function to your 4-bit counter design.
5. **Explain** an application of counters in real-world devices (e.g., elevator, stopwatch, vending machine).

Topic 14: **Memory Circuits**

1 / A – Target Population :-

This topic is designed for **first-year students** in the **Department of Electronic Techniques** at the **Technological Institute of Basra**. Students are expected to have prior understanding of flip-flops and registers. This lesson introduces how memory elements are structured and used to store digital information in systems.

---

1 / B – Rationale :-

Memory circuits are essential to all digital systems—from microcontrollers to high-speed computers. They allow temporary or permanent data storage and retrieval. By studying memory circuits, students can:

- Understand how digital devices **store bits**
- Differentiate between **volatile and non-volatile** memory
- Build foundational knowledge for **RAM, ROM, and EEPROM** systems
- Apply memory concepts in **register file**, **buffering**, and **data transfer**

This knowledge is crucial for any field involving digital electronics, embedded systems, or processor design.

---

1 / C – Central Idea :-

Memory circuits are built using **sequential logic elements** like flip-flops, combined to store multi-bit values. This unit covers:

1. **Classification of Memory**

    - **Volatile** (e.g., RAM – Random Access Memory)
    - **Non-Volatile** (e.g., ROM – Read-Only Memory, EEPROM)
2. **Basic Memory Elements**
    - **1-bit memory cell** using flip-flop
    - **Register** (group of flip-flops)
    - **Memory array** (rows/columns of storage cells)
3. **Addressing Mechanism**
    - Memory organized into addressable locations
    - Uses decoders to select memory cells
4. **Read/Write Operations**
    - Write = Store data into memory location
    - Read = Retrieve stored data
    - Control signals (Read Enable, Write Enable, Chip Select)
5. **Memory Expansion**
    - Word size vs. memory depth
    - Cascading chips for large memory blocks

---

1 / D – Performance Objectives :-

By the end of this unit, students will be able to:

Knowledge

- Define key types of memory: RAM, ROM, EEPROM
- Understand the structure and function of memory arrays

Skills

- Design simple 1-bit and 4-bit memory circuits using flip-flops
- Create a memory module using decoders and registers
- Interpret timing diagrams for memory operations

Competence

- Apply memory logic to register files, microcontroller memory, and buffer systems
- Integrate memory with control logic in digital systems
- Evaluate memory performance and data integrity

- دوائر الذاكرة(Memory Circuits)

- **دوائر الذاكرة (Memory Circuits)** هي وحدات رقمية تُستخدم **لتخزين البيانات واسترجاعها** في الأنظمة الإلكترونية. تُعد الذاكرة أحد المكونات الأساسية في أجهزة الحاسوب والمعالجات والأنظمة المدمجة.

- 🔹 **تصنيف الذاكرة في الدوائر الرقمية**

يمكن تصنيف الذاكرة وفقًا **لطريقة التخزين، إمكانية الكتابة والقراءة، وسرعة الوصول**.

نوع الذاكرة	الاختصار	الوصف
الذاكرة العشوائية (Random Access Memory)	**RAM**	ذاكرة سريعة تستخدم لتخزين البيانات أثناء تشغيل النظام.
الذاكرة الدائمة (Read Only Memory)	**ROM**	ذاكرة غير قابلة للكتابة تُستخدم لتخزين التعليمات الأساسية للنظام.
الذاكرة الديناميكية (Dynamic RAM)	**DRAM**	ذاكرة تحتاج إلى تحديث مستمر للحفاظ على البيانات.
الذاكرة الساكنة (Static RAM)	**SRAM**	ذاكرة أسرع من DRAM ولكن أكثر تكلفة وتستهلك طاقة أكثر.
الذاكرة القابلة للمسح والبرمجة (Erasable Programmable ROM)	**EPROM**	ذاكرة ROM يمكن إعادة برمجتها باستخدام الأشعة فوق البنفسجية.
الذاكرة القابلة للمسح إلكترونيًا (Electrically Erasable Programmable ROM)	**EEPROM**	ذاكرة ROM يمكن إعادة كتابتها كهربائيًا، مثل ذاكرة **Flash Memory**.

-

2. مكونات دوائر الذاكرة الأساسية

- 🔹 **أي دائرة ذاكرة تتكون من المكونات التالية:**

1. **الخلايا الثنائية (Memory Cells)**: وحدات تخزين البيانات التي تمثل "0" أو "1".
2. **خطوط العنوان (Address Bus)**: تُستخدم لاختيار موقع البيانات داخل الذاكرة.
3. **خطوط البيانات (Data Bus)**: تُستخدم لنقل البيانات بين الذاكرة والمعالج.
4. **إشارات التحكم (Control Signals)**: تحدد عمليات القراءة والكتابة إلى الذاكرة.

-
-

3. أنواع دوائر الذاكرة

🔹 (1) دائرة ذاكرة RAM الأساسية

- تتكون من **عدة مراجيح D Flip-Flops** تُستخدم لتخزين البيانات الثنائية.
- تتطلب **إشارة ساعة (Clock Signal)** للتحكم في عمليات **القراءة والكتابة**.

📌 **مخطط مبسط لذاكرة RAM باستخدام مراجيح D:**

```
Address Lines → [D Flip-Flop] → Data Output
```

📌 **جدول الحقيقة لذاكرة RAM (قراءة وكتابة)**

العملية	إشارة القراءة (Read Enable - RE)	إشارة الكتابة (Write Enable - WE)
كتابة البيانات في الخلايا	0	1
قراءة البيانات من الخلايا	1	0
لا تغيير (الاحتفاظ بالبيانات)	0	0
غير صالح	1	1

-

🔹 (2) دائرة ذاكرة ROM الأساسية

- تتكون من **بوابات منطقية ثابتة** تُخزّن بيانات لا يمكن تغييرها بعد التصنيع.
- لا تحتوي على **إشارة كتابة**، فقط يمكن **قراءة البيانات**.

📌 **مخطط مبسط لذاكرة ROM باستخدام بوابات منطقية:**

```
Address Lines → [Decoder] → [Fixed Logic Gates] → Data Output
```

📌 **جدول الحقيقة لذاكرة ROM**

عنوان (Address)	البيانات المخزنة (Stored Data)
0000	1010
0001	1100
0010	0110
0011	1001

-
-
-

Post-Test

1. What is the difference between **volatile** and **non-volatile** memory?
2. How many flip-flops are needed to build an **8-bit register**?

3. What is the role of a **decoder** in a memory circuit?
4. Explain the difference between **read** and **write** operations in memory.
5. Design a **4 × 4 memory array** using D flip-flops and address lines.

Key Answers

1. **Volatile memory** loses data when power is off (e.g., RAM); **non-volatile** retains data (e.g., ROM).
2. 8 D flip-flops (each stores 1 bit).
3. The decoder selects the **active memory location** based on the address input.
4. **Read** retrieves stored data; **Write** changes or loads new data into memory.
5. A 4×4 array requires:
    - 4 rows × 4 bits = 16 flip-flops
    - 2-bit address input
    - A 2-to-4 decoder
    - Read/Write control logic

Homework Assignments

1. **Draw a block diagram** of a 4-bit register with write enable control.
2. **Explain** the difference between ROM, RAM, and EEPROM in terms of use cases.
3. **Design a 2 × 4 memory module** using flip-flops, and show how address selection works.
4. **Challenge**: Simulate a 4-bit RAM with read/write control using Logisim.
5. **Write a short report** on how memory is used in a microcontroller system.

Topic 15: Analog-to-Digital and Digital-to-Analog Converters (ADC & DAC)

1 / A – Target Population :-

This topic is prepared for **first-year students** in the **Department of Electronic Techniques** at the **Technological Institute of Basra**. The students should have a basic understanding of binary numbers, digital logic levels, and basic electronics principles. They are now ready to explore how real-world analog signals interact with digital systems through **conversion circuits**.

---

1 / B – Rationale :-

Most real-world signals (temperature, sound, voltage) are **analog**, while computers and microcontrollers operate in **digital** form. ADCs and DACs are essential to **interface between analog and digital domains**, and are widely used in:

- **Sensors and data acquisition systems**
- **Embedded systems and microcontrollers**
- **Audio processing**
- **Industrial automation**

Understanding ADCs and DACs equips students to bridge the analog and digital worlds in measurement, control, and signal processing.

---

1 / C – Central Idea :-

This unit introduces the principles, operation, and application of **Analog-to-Digital Converters (ADC)** and **Digital-to-Analog Converters (DAC)**, including:

1. **Analog-to-Digital Converter (ADC)**
    - Converts a continuous analog voltage into a digital binary value
    - Key concepts: **Resolution**, **Sampling**, **Quantization**, and **Conversion time**
2. **Types of ADCs**
    - **Successive Approximation (SAR)**
    - **Flash ADC**
    - **Dual Slope ADC**
3. **Digital-to-Analog Converter (DAC)**
    - Converts a digital binary input into an analog output voltage or current
    - Types: **R-2R Ladder**, **Weighted Resistor DAC**
4. **Performance Parameters**
    - Resolution (e.g., 8-bit, 10-bit, 12-bit)
    - Accuracy, Linearity, Conversion Rate
    - Reference Voltage
5. **Applications**
    - ADC: Temperature sensors, microphones, battery monitors
    - DAC: Audio output, analog controls, waveform generators

1 / D – Performance Objectives :-

By the end of this topic, students will be able to:

Knowledge

- Describe the function of ADCs and DACs
- List types of ADC and DAC architectures

Skills

- Interpret block diagrams of ADC/DAC systems
- Calculate digital output for given analog input and vice versa
- Select appropriate resolution based on signal needs

Competence

- Apply ADCs and DACs in sensor systems and embedded applications
- Evaluate performance specs such as bit resolution and sampling rate
- Simulate analog-digital conversions using software tools

التحويل من رقمي إلى نظيري(Digital to Analog Conversion - DAC)

- المحول الرقمي إلى النظيري **(Digital to Analog Converter - DAC)** هو **دائرة إلكترونية** تقوم بتحويل **الإشارات الرقمية** (المكونة من "0" و "1") إلى **إشارة نظيرية** مستمرة يمكن استخدامها في الأجهزة الحقيقية مثل **مكبرات الصوت، أجهزة العرض، والأنظمة المدمجة**.
- هذا التحويل مهم لأن **الأجهزة الإلكترونية تتعامل مع بيانات رقمية، بينما البيئة الطبيعية تعمل بإشارات نظيرية**.

🔹 **أمثلة على استخدامDAC**

✅ تشغيل **ملفات الصوت الرقمية**)مثل MP3) عبر مكبرات الصوت.
✅ عرض الصور الرقمية على **الشاشات التناظرية** مثل شاشات CRT القديمة.
✅ التحكم في **المحركات الكهربائية** باستخدام إشارات رقمية.
✅ أنظمة التحكم **في السيارات والطائرات** لتحويل البيانات الرقمية إلى أوامر كهربائية.

مبدأ عمل المحول الرقمي إلى النظيري(DAC)

🔹 كيف يتم تحويل القيم الرقمية إلى إشارة نظيرية؟

- في النظام الرقمي، يتم تمثيل القيم باستخدام **عدد معين من البتات**.
- يتم تحويل هذه القيم إلى **جهد كهربائي (Voltage)** أو **تيار كهربائي (Current)** متناسب مع القيمة الرقمية.
- الإشارة الناتجة تكون **متدرجة(Staircase Signal)** ، ويتم تنعيمها باستخدام **مرشحات (Filters)**للحصول على إشارة نظيرية سلسة.

3. أنواع المحولات الرقمية إلى النظيرية (DAC)

هناك عدة أنواع من **دوائر DAC**، ولكل منها طريقة مختلفة في التحويل:

النوع	طريقة العمل	المميزات	العيوب
مقسم المقاومات (Resistor String DAC)	يستخدم شبكة مقاومات لتمثيل القيم الرقمية كمستويات جهد	سهل التنفيذ، دقة جيدة	يحتاج عدد كبير من المقاومات في القيم الكبيرة
شبكة R-2R (R-2R Ladder DAC)	يستخدم شبكة من المقاومات بنسبة 1:2 لتحويل القيم الرقمية إلى جهد	كفاءة عالية، عدد أقل من المقاومات	يحتاج إلى دقة عالية في المقاومات
محول التيار المرجح (Weighted Current DAC)	يولد تيارات متناسبة مع كل بت ويجمعها للحصول على الجهد النهائي	سريع، سهل التطبيق	يحتاج إلى دقة عالية في مصادر التيار
محول دلتا-سيجما (Delta-Sigma DAC)	يستخدم تعديل عرض النبضة (PWM) للحصول على خرج تناظري سلس	دقة عالية، يقلل الضوضاء	معقد التنفيذ، يحتاج معالجة إضافية

4. جدول الحقيقة لمحول رقمي إلى نظيري 3-بت

في هذا المثال، لدينا 3 DAC-بت يحول القيم الرقمية إلى جهد نظيري بين 0V و 7V.

القيمة الرقمية (3-bit Binary)	القيمة العشرية (Decimal Value)	الجهد النظيري (Analog Voltage)
000	0	0.0V
001	1	1.0V
010	2	2.0V
011	3	3.0V
100	4	4.0V
101	5	5.0V
110	6	6.0V
111	7	7.0V

📌 في هذا المثال، يتم تعيين كل مستوى رقمي لمستوى جهد محدد، بحيث تكون الخطوة بين كل مستوى = 1V.

التحويل من نظيري إلى رقمي(Analog to Digital Conversion - ADC)

- المحول **النظيري إلى الرقمي (Analog to Digital Converter - ADC)** هو **دائرة إلكترونية** تقوم بتحويل **الإشارات النظيرية المستمرة** (مثل الصوت، درجة الحرارة، الجهد الكهربائي) إلى **إشارة رقمية** يمكن معالجتها في الأجهزة الرقمية مثل **الكمبيوتر والمعالجات الدقيقة والميكروكنترولر**.
- البيئة الطبيعية تعمل بإشارات **نظيرية مستمرة**، بينما تعمل الحواسيب والإلكترونيات الحديثة بالإشارات **الرقمية (0 و 1)**، لذا فإن **ADCضروري لنقل البيانات من العالم الحقيقي إلى العالم الرقمي**.

🔹 أمثلة على استخدامADC

✅ **الميكروفونات الرقمية** التي تحول الصوت إلى بيانات رقمية يمكن تخزينها أو إرسالها.
✅ **مستشعرات الحرارة** التي تقيس درجة الحرارة وتحولها إلى بيانات يمكن قراءتها رقميًا.
✅ **الكاميرات الرقمية** التي تحول الضوء إلى بيانات بكسل رقمية.
✅ **أنظمة التحكم في السيارات** التي تعتمد على بيانات مستشعرات الوقود والسرعة ودرجة الحرارة.

مبدأ عمل المحول النظيري إلى الرقمي(ADC)

🔹 **كيف يتم تحويل القيم النظيرية إلى إشارة رقمية؟**

1. **أخذ العينة (Sampling):** يتم أخذ قراءات دورية من الإشارة النظيرية عند **معدّل زمني ثابت**.
2. **التكميم (Quantization):** يتم تعيين كل قيمة نظيرية إلى أقرب مستوى رقمي متاح.
3. **الترميز (Encoding):** يتم تمثيل القيمة الرقمية بتنسيق ثنائي.(Binary Code)

📌 **كلما زاد عدد البتات في المحول(ADC Resolution) ، زادت دقة التحويل.**

3. أنواع المحولات النظيرية إلى الرقمية (ADC)

النوع	طريقة العمل	المميزات	العيوب
ADC التقريبي المتتابع (Successive Approximation) (ADC - SAR	يقارن الإشارة النظيرية بقيم مرجعية بترتيب متتابع حتى يحصل على القيمة الدقيقة	دقة عالية، استهلاك طاقة منخفض	أبطأ من الأنواع الأخرى
ADC التكاملي (Integrating) (ADC	يحول الإشارة إلى وقت زمني ويحسب مدتها للحصول على القيمة الرقمية	دقيق جدًا، مناسب للقياسات البطيئة	بطيء نسبيًا
ADC الفوري (Flash ADC)	يستخدم شبكة من المقارنات للحصول على القيمة الرقمية فورًا	أسرع نوع، مناسب للتطبيقات عالية السرعة	يحتاج عددًا كبيرًا من المقارنات، مكلف
ADC دلتا-سيجما (Delta-Sigma) (ADC	يعتمد على زيادة دقة العينات وتقليل الضوضاء	دقة عالية جدًا، مناسب للصوت والتطبيقات الطبية	يتطلب معالجة رقمية بعد التحويل

4. جدول الحقيقة لمحول نظيري إلى رقمي (ADC) - مثال لـ 3-بت

نفترض أن لدينا **ADC 3-بت** يقوم بتحويل إشارة جهد نظيري بين **0V و 7V** إلى **قيم رقمية**.

القيمة الرقمية (3-bit Binary)	القيمة العشرية (Decimal Value)	الجهد النظيري (Analog Voltage)
000	0	0.0V
001	1	1.0V
010	2	2.0V
011	3	3.0V
100	4	4.0V
101	5	5.0V
110	6	6.0V
111	7	7.0V

📌 في هذا المثال، يتم تعيين كل مستوى جهد إلى رمز ثنائي معين، بحيث تكون الخطوة بين كل مستوى = 1V.

Post-Test

1. What is the role of an ADC in a digital system?
2. Calculate the digital output of an **8-bit ADC** with Vref = 5V when the analog input is 3V.
3. What are two common types of ADC architecture?
4. What is the analog output of a **4-bit DAC** with Vref = 4V and digital input = 1100?
5. What is the resolution (in volts per step) of a 10-bit ADC with Vref = 3.3V?

Key Answers

1. An ADC converts analog voltage signals into digital binary numbers that digital circuits can process.
2. Output = (3 / 5) × 255 = 153 (binary: 10011001)
3. **Successive Approximation** and **Flash ADC**
4. Decimal input = 12 → (12/16) × 4V = **3.0V**

5. Resolution = 3.3V / 1024 = **~0.00322 V/step**

Homework Assignments

1. Draw the **block diagram** of a Successive Approximation ADC and explain each block.
2. Calculate the **digital output** of a 10-bit ADC with Vref = 2.5V and input voltage = 1V.
3. For a 4-bit DAC, plot the analog output for all digital input values from 0000 to 1111.
4. **Compare** Flash ADC and SAR ADC in terms of speed, cost, and complexity.
5. **Simulate** an analog-to-digital system using any logic simulation software (e.g., Tinkercad or Proteus).

Topic 16: Voltage-to-Frequency Converter (VFC)

1 / A – Target Population :-

This topic is designed for **first-year students** in the **Department of Electronic Techniques** at the **Technological Institute of Basra**, who have studied analog signals, op-amps, and basic digital interfacing. They are now ready to understand **signal conditioning circuits**, especially converting analog voltages into frequency-based outputs.

---

1 / B – Rationale :-

A **Voltage-to-Frequency Converter (VFC)** converts a continuous analog voltage into a frequency signal that can be **counted or measured digitally**. VFCs are essential in:

- **Digital voltmeters and frequency counters**
- **Analog sensor interfacing** (e.g., temperature, pressure)
- **Long-distance signal transmission** using frequency instead of voltage
- **Embedded systems** with limited ADC capabilities

VFCs are often used when digital systems need to **sense or process analog values** via time-based measurements.

---

1 / C – Central Idea :-

This unit introduces how a Voltage-to-Frequency Converter operates, its components, and applications. The core concepts include:

1. **Principle of Operation**
   - VFCs output a frequency that is **proportional to input voltage**: f_out ∝ V_in
   - The higher the voltage, the faster the pulse generation
2. **Circuit Components**
   - **Integrator**: Converts input voltage to a ramp signal
   - **Comparator**: Monitors ramp against threshold
   - **Schmitt Trigger or flip-flop**: Creates square wave output
   - **Reset/Discharge mechanism**: To repeat the cycle
3. **Types of VFCs**
   - Op-amp based VFC
   - IC-based (e.g., LM331, AD654)
4. **Applications**
   - Sensor signal conditioning
   - Long-distance analog signal transmission via frequency
   - Measurement systems using microcontrollers or counters

---

1 / D – Performance Objectives :-

By the end of this topic, students will be able to:

Knowledge

- Understand the function and purpose of VFCs
- Recognize the core components of a VFC circuit

Skills

- Design a basic VFC using op-amps and comparators
- Calculate output frequency for given input voltage
- Interpret frequency outputs using timing diagrams

Competence

- Apply VFCs in sensor-to-microcontroller interfacing
- Simulate VFC performance under different voltages
- Select appropriate ICs for VFC design

محولة الفولتية إلى التردد(Voltage to Frequency Converter - VFC)

- محولة الفولتية إلى التردد **(Voltage to Frequency Converter - VFC)**هي **دائرة إلكترونية** تقوم بتحويل **إشارة جهد نظيري (Analog Voltage)** إلى **إشارة ترددية رقمية (Frequency Signal)**تتناسب مع قيمة الجهد المدخل.
- هذا النوع من المحولات يستخدم عندما يكون **إرسال الإشارة عبر التردد أكثر كفاءة من إرسالها كجهد نظيري**، كما هو الحال في أنظمة الاتصالات والتحكم الصناعي.

🔹 تطبيقات محولةVFC

✅ قياس الجهد الكهربائي باستخدام **عدادات التردد**.
✅ تحويل الإشارات التناظرية إلى نبضات **PWM**في الأنظمة الرقمية.
✅ إرسال البيانات عبر **المسافات الطويلة** في أنظمة القياس والتحكم الصناعي.
✅ معالجة الإشارات في **أجهزة الاستشعار الرقمية**.

2. مبدأ عمل محولة الفولتية إلى التردد (VFC)

- عندما يتم إدخال **إشارة جهد نظيري**، تقوم الدائرة بتحويله إلى **تردد معين** بحيث:

$$V_{in} \times k = f_{out}$$

حيث:

- f_{out} هو **التردد الناتج**.
- k هو **ثابت التحويل** (يعتمد على تصميم الدائرة).
- V_{in} هو **الجهد المدخل**.

📌 **كلما زاد الجهد المدخل، زاد التردد الناتج بشكل خطي.**

جدول الحقيقة لمحولة الفولتية إلى التردد(VFC)

يُظهر الجدول التالي العلاقة بين **الجهد المدخل والتردد الناتج** لمحول يعمل بتردد بين **1 كيلوهرتز و 10 كيلوهرتز**.

الجهد المدخل (V_in)	التردد الناتج (F_out) - كيلوهرتز
0V	kHz 1
1V	kHz 2
2V	kHz 3
3V	kHz 4
4V	kHz 5
5V	kHz 6
6V	kHz 7
7V	kHz 8
8V	kHz 9
9V	kHz 10

📌 **في هذا المثال، كل فولت يضيف 1 كيلوهرتز إلى التردد الناتج.**

إحدى أسهل الطرق لتنفيذ VFC هي باستخدام **المؤقت 555 (Timer 555)**.

🔹 المكونات المطلوبة:

- مؤقت Timer 555.
- مقاومة $R1$ و $R2$.
- مكثف C.
- مصدر جهد (V_in) **متغير**.

🔹 المخطط الأساسي للدائرة:

```
Edit  Copy

V_in → [مضخم عمليات] → [مؤقت 555] → [خرج نبضي F_out]
```

📌 **عندما يزيد الجهد المدخل (V_in)، يزيد تردد النبضات الناتجة (F_out).**

🔹 معادلة التردد الناتج:

$$F_{out} = V_{in} \times \frac{1.44}{C \times (R1 + 2R2)}$$

📌 **توضح هذه المعادلة كيف يعتمد التردد الناتج على الجهد المدخل والمكونات الإلكترونية.**

A **Voltage-to-Frequency Converter (VFC)** is designed using a **555 Timer** in astable mode. The circuit operates with the following component values:

- $R1 = 2k\Omega$
- $R2 = 8k\Omega$
- $C = 15nF$

The output frequency is given by the equation:

$$F_{out} = \frac{1.44}{(R1 + 2R2) \times C} \times V_{in}$$

(a) Calculate the output frequency (F_{out}) when the input voltage is $V_{in} = 5V$.
(b) Find the maximum output frequency when $V_{in} = 10V$.
(c) What is the minimum frequency when $V_{in} = 0V$?

📌 Solution: Voltage-to-Frequency Calculation

We are given the formula:

$$F_{out} = \frac{1.44}{(R1 + 2R2) \times C} \times V_{in}$$

With the given values:

- $R1 = 2k\Omega$
- $R2 = 8k\Omega$
- $C = 15nF$

Now, let's calculate the output frequency for different input voltages:

(a) Calculate F_{out} when $V_{in} = 5V$:

$$F_{out} = \frac{1.44}{(2000 + 2(8000)) \times (15 \times 10^{-9})} \times 5$$

$$F_{out} = 26.67kHz$$

(b) Calculate F_{out} when $V_{in} = 10V$ (Maximum Frequency):

$$F_{out} = 53.33kHz$$

(c) Calculate F_{out} when $V_{in} = 0V$ (Minimum Frequency):

$$F_{out} = 0Hz$$

📌 New Question: Voltage-to-Frequency Conversion

A **Voltage-to-Frequency Converter (VFC)** is designed using a **555 Timer** in astable mode. The circuit parameters are given as:

- $R1 = 1.5k\Omega$
- $R2 = 6.8k\Omega$
- $C = 22nF$

The output frequency is determined by the equation:

$$F_{out} = \frac{1.44}{(R1 + 2R2) \times C} \times V_{in}$$

(a) Calculate the output frequency F_{out} when the input voltage is $V_{in} = 3V$.

(b) Find the maximum output frequency when $V_{in} = 12V$.

(c) What is the minimum frequency when $V_{in} = 0V$?

Post-Test

1. What is the **main function** of a VFC?
2. In a VFC, how does the output frequency relate to the input voltage?
3. What role does the **integrator** play in a VFC?
4. If a VFC outputs 5 kHz for 2.5V input, what frequency would it output at 1.25V?
5. List one advantage and one application of using a VFC over a direct ADC.

Key Answers

1. It converts an analog input voltage into a digital **frequency** output signal.
2. It is **directly proportional**: as V_in increases, f_out increases.
3. The integrator converts the DC voltage into a **linearly increasing ramp** signal.
4. Assuming linearity:
   1.25V → 2.5 kHz (half the voltage → half the frequency)
   5.
   - **Advantage**: More immune to analog noise and line resistance
   - **Application**: Remote sensor reading over long cables (e.g., pressure sensor to digital counter)

Homework Assignments

1. **Draw the block diagram** of a VFC circuit using op-amp integrator and comparator.
2. **Simulate a basic VFC** using any circuit simulator and verify frequency output for 1V, 2V, 3V inputs.
3. **Research and summarize** the specifications of LM331 or AD654 VFC ICs.
4. **Design Challenge**: Create a circuit where 0–5V input produces 0–1000 Hz output using a VFC and verify the formula.
5. **Explain** why VFCs are useful in industrial automation where interference is common.