وزارة التعليم العالي والبحث العلمي
الجامعة التقنية الشمالية
معهد الإدارة التقني نينوى

الحقيبة التعليمية

القسم العلمي: تقنيات أنظمة الحاسوب

اسم المقرر: التصميم المنطقي

المرحلة / المستوى: الأولى

الفصل الدراسي: الاول

السنة الدراسية: 2025

اسم المقرر:	التصميم المنطقي					
القسم:	تقنيات أنظمة الحاسوب					
الكلية:	معهد الإدارة التقني نينوى					
المرحلة / المستوى	الاولى					
الفصل الدراسي:	الاول					
عدد الساعات الاسبوعية:	نظري	1	عملي	2		
عدد الوحدات الدراسية:	3					
الرمز:						
نوع المادة	نظري		عملي		كلهما	✓
هل يتوفر نظير للمقرر في الاقسام الاخرى	كلا					
اسم المقرر النظير (ان وجد)	كلا					
القسم	كلا					
رمز المقرر النظير	كلا					
معلومات تدريسي المادة						
اسم مدرس (مدرسي) المقرر:	**لميس علي حسين**					
اللقب العلمي:	**مدرس**					
سنة الحصول على اللقب	**2025**					
الشهادة:	**ماجستير**					
سنة الحصول على الشهادة	**2025**					
عدد سنوات الخبرة (تدريس)	**10**					

الوصف العام للمقرر

يتناول هذا القسم المقدمة لموضوع التصميم المنطقي، حيث يقدم نظرة عامة عن مفهوم التصميم المنطقي وأهميته في علوم الحاسوب والهندسة. كما يتطرق إلى تاريخ وتطور التصميم المنطقي ودوره في تطبيقات الحياة اليومية والتكنولوجيا. يسلط الضوء على الأهداف والمبادئ الأساسية للتصميم المنطقي وكيفية تطبيقها في بناء الأنظمة والبرمجيات. يتضمن هذا القسم أيضًا العلاقة بين التصميم المنطقي والرياضيات والمنطق وكيف يساهم كل منهما في تعزيز التفكير النقدي وحل المشاكل بطريقة منطقية ونافعة.

الاهداف العامة

- التصميم المنطقي يمثل جزءاً أساسياً من مجال أنظمة وعلوم الحاسوب حيث يساعد في فهم وتحليل وتصميم الدوائر الإلكترونية والأنظمة الرقمية. يتناول التصميم المنطقي العمليات الرقمية وكيفية تنفيذ وتحقيق الوظائف الحسابية والمنطقية باستخدام الدوائر الإلكترونية. يشمل أيضاً التصميم المنطقي تحليل وفهم البوابات المنطقية والعلاقة بينها وبين النظام الثنائي.
- تصميم المنطقي له أهمية كبيرة في مجالات متعددة مثل تصميم الدوائر الإلكترونية والأنظمة الكهربائية والحاسوبية. يساعد التصميم المنطقي في فهم وتحليل الأنظمة الرقمية والعمليات الحسابية بشكل دقيق ومنطقي. ويساعد في تصميم وبناء الدوائر الإلكترونية التي تنفذ وتنقل الإشارات المنطقية بفعالية، مما يسهم في تطوير التكنولوجيا والابتكار في مجالات الهندسة وتقنية المعلومات.
-

الأهداف الخاصة

- معرفة الأنظمة العددية
- التحويلات ما بين الأنظمة العددية
- الاسهاب في دراسة انظام الثنائي والذي يعتبر الأساس في الثورة الرقمية
- البوابات المنطقية الاساسية

الأهداف السلوكية او نواتج التعلم

- فهم الأنظمة العددية المختلفة: سيتمكن الطلاب من التعرف على الأنظمة العددية الثنائية والعشرية والسادسة عشرية والثمانية والتحويل بينها.
- تحدد الأداء الذي يسعى المعلم إلى إحداثه لدى المتعلم وشروط حدوثه ومستوى التمكن المطلوب في الاداء.
- أي انها تحدث تغيرا في سلوك المتعلم

- **أهداف تدريسية:**

بعد الانتهاء من الدرس (المحاضرة) سيكون الطالب قادرا على ان:

- يعرف مختلف الأنظمة العددية
- كيفية اجراء التحويلات ما بين الأنظمة العددية
- البوابات المنطقية الاساسية

المتطلبات السابقة

- معرفة الأنظمة العددية
- النظام العشري

الأهداف السلوكية او مخرجات التعليم الأساسية		
ت	تفصيل الهدف السلوكي او مخرج التعليم	آلية التقييم
1	التعرف على الأنظمة العددية من خلال شرح الأسس والمراتب العددية لكل نظام	امتحان نظري + امتحان عملي
2	النظام الثنائي، النظام الثنائي، النظام السادس عشر	امتحان نظري + امتحان عملي
3	التحويلات بين الأنظمة	امتحان نظري + امتحان عملي
4	العمليات الحسابية	امتحان نظري + امتحان عملي
5	أسس البوابات والدوائر المنطقية	امتحان نظري + امتحان عملي
6	الجبر البولي	امتحان نظري + امتحان عملي
7	نظرية ديمو رجان	امتحان نظري + امتحان عملي
8	خارطة كارنوف	امتحان نظري + امتحان عملي

أساليب التدريس (حدد مجموعة متنوعة من أساليب التدريس لتناسب احتياجات الطلاب ومحتوى المقرر)

الاسلوب او الطريقة	مبررات الاختيار
1. الأمثلة	الأمثلة هي أدوات تعليمية تستخدم لتوضيح المفاهيم والنظريات من خلال تقديم حالات محددة. الأمثلة الواقعية (من الحياة اليومية). الأمثلة النظرية (تعتمد على المفاهيم المجردة). الأمثلة الموجهة (تستخدم في التعليم المبرمج).
2. التدريبات	التدريبات تهدف إلى تعزيز الفهم وتطوير المهارات من خلال التطبيق العملي تدريبات فردية. تدريبات جماعية. تدريبات تفاعلية (استخدام الألعاب التعليمية). تدريبات تحليلية (تحليل الحالات والدراسات).
3. الوسائل والمثيرات والتقنيات	الوسائل التعليمية تشمل كل ما يساعد على نقل المعلومة بفعالية: الوسائل السمعية والبصرية (الفيديوهات، التسجيلات الصوتية). اللوحات التعليمية والنماذج. المثيرات التعليمية: استخدام القصص. المسابقات. الأنشطة العملية. التقنيات: استخدام التكنولوجيا في التعليم (السبورة الذكية، التطبيقات التعليمية).
4.مواقع الكترونية علمية	مواقع علمية معروفة: • Google Scholar: للبحث عن الأبحاث والمقالات العلمية. • Khan Academy: لتعلم مواد أكاديمية متنوعة. • Coursera: لدورات تعليمية من جامعات عالمية. • edX: منصة أخرى للتعليم الإلكتروني من مؤسسات تعليمية مرموقة.
5.تطبيقات (APLICATIONS)	تطبيقات تعليمية مفيدة: Duolingo: لتعلم اللغات. Kahoot!: لإنشاء اختبارات تفاعلية. Quizlet: لإنشاء بطاقات تعليمية. Microsoft Teams و Zoom: للاجتماعات والورش التعليمية عبر الإنترنت.
6. الورش والندوات الطلابية	الورش والندوات تعتبر فرصة لتعميق الفهم من خلال التفاعل المباشر والمناقشات. أنواع الورش والندوات: ورش العمل العملية (لتطوير المهارات الفنية). ندوات أكاديمية (لمناقشة الأبحاث والدراسات). ندوات توجيهية (للإرشاد الأكاديمي والمهني).

الفصل الاول من المحتوى العلمي					
عنوان الفصل	الوقت				
التوزيع الزمني	النظري	العملي	العنوان الفرعي	طريقة التدريس	التقنيات	طرق القياس
الأسبوع الأول	1	2	التعرف على الأنظمة العددية من خلال شرح الأسس والمراتب العددية لكل نظام	محاضرة نظرية + وعملية	عرض تقديمي، شرح، أمثلة، أسئلة وأجوبة، مشاركة الطالب على اللوحة	مراجعة عامة
الأسبوع الثاني	1	2	النظام الثنائي	محاضرة نظرية + وعملية	عرض تقديمي، شرح، أمثلة، أسئلة وأجوبة، مشاركة الطالب على اللوحة	اختبار نظري وعملي
الأسبوع الثالث	1	2	التحويل بين النظام العشري والثنائي	محاضرة نظرية + وعملية	عرض تقديمي، شرح، أمثلة، أسئلة وأجوبة، مشاركة الطالب على اللوحة	اختبار نظري وعملي
الأسبوع الرابع	1	2	العمليات الحسابية على النظام الثنائي	محاضرة نظرية + وعملية	عرض تقديمي، شرح، أمثلة، أسئلة وأجوبة، مشاركة الطالب على اللوحة	اختبار نظري وعملي
الأسبوع الخامس	1	2	النظام الثماني	محاضرة نظرية + وعملية	عرض تقديمي، شرح، أمثلة، أسئلة وأجوبة، مشاركة الطالب على اللوحة	اختبار نظري وعملي
الأسبوع السادس	1	2	التحويل بين النظام الثماني والثنائي والعشري	محاضرة نظرية + وعملية	عرض تقديمي، شرح، أمثلة، أسئلة وأجوبة، مشاركة الطالب على اللوحة	اختبار نظري وعملي
الأسبوع السابع	1	2	النظام السادس عشر	محاضرة نظرية + وعملية	عرض تقديمي، شرح، أمثلة، أسئلة وأجوبة، مشاركة الطالب على اللوحة	اختبار نظري وعملي
الأسبوع الثامن	1	2	التحويل بين النظام السادس عشر والثنائي والعشري	محاضرة نظرية + وعملية	عرض تقديمي، شرح، أمثلة، أسئلة وأجوبة، مشاركة الطالب على اللوحة	اختبار نظري وعملي
الأسبوع التاسع	1	2	البوابات المنطقية	محاضرة نظرية + وعملية	عرض تقديمي، شرح، أمثلة، أسئلة وأجوبة، مشاركة الطالب على اللوحة	اختبار نظري وعملي
الأسبوع العاشر	1	2	الجبر البولي	محاضرة نظرية + وعملية	عرض تقديمي، شرح، أمثلة، أسئلة وأجوبة، مشاركة الطالب على اللوحة	اختبار نظري وعملي
الأسبوع الحادي عشر	1	2	ديمورجان	محاضرة نظرية + وعملية	عرض تقديمي، شرح، أمثلة، أسئلة وأجوبة، مشاركة الطالب على اللوحة	اختبار نظري وعملي
الأسبوع الثاني عشر	1	2	SOP	محاضرة نظرية + وعملية	عرض تقديمي، شرح، أمثلة، أسئلة وأجوبة، مشاركة الطالب على اللوحة	اختبار نظري وعملي
الأسبوع الثالث عشر	1	2	POS	محاضرة نظرية + وعملية	عرض تقديمي، شرح، أمثلة، أسئلة وأجوبة، مشاركة الطالب على اللوحة	اختبار نظري وعملي

الأسبوع الرابع عشر	1	2	K-Map	محاضرة نظرية + وعملية	عرض تقديمي، شرح، أمثلة، أسئلة وأجوبة، مشاركة الطالب على اللوحة	اختبار نظري وعملي
الأسبوع الخامس عشر	1	2	BCD	محاضرة نظرية + وعملية	عرض تقديمي، شرح، أمثلة، أسئلة وأجوبة، مشاركة الطالب على اللوحة	اختبار نظري وعملي

Digital Logic Design

خارطة القياس المعتمدة

المحتوى التعليمي	عناوين الفصول	الأهمية النسبية	الأهداف السلوكية					عدد الفقرات
			المعرفة	الفهم	التطبيق	التحليل	التقييم	
			النسبة					
الأسبوع الأول	التعرف على الأنظمة العددية من خلال شرح الأسس والمراتب العددية لكل نظام	%10	%6.66	%6	%5	%5	%4	1
الأسبوع الثاني	النظام الثنائي	%5	%6.66	6%	5%	5%	4%	1
الأسبوع الثالث	التحويل بين النظام العشري والثنائي	%5	%6.66	6%	5%	5%	4%	1
الأسبوع الرابع	العمليات الحسابية على النظام الثنائي	%5	%6.66	6%	5%	5%	4%	1
الأسبوع الخامس	النظام الثماني	%5	%6.66	6%	5%	5%	4%	1
الأسبوع السادس	التحويل بين النظام الثماني والثنائي والعشري	%5	%6.66	6%	5%	5%	4%	1
الاسبوع السابع	النظام السادس عشر	%8	%6.66	6%	5%	5%	4%	1
الأسبوع الثامن	التحويل بين النظام السادس عشر والثنائي والعشري	%5	%6.66	6%	5%	5%	4%	1
الأسبوع التاسع	البوابات المنطقية	%10	%6.66	6%	5%	5%	4%	1
الأسبوع العاشر	الجبر البولي	%7	%6.66	6%	5%	5%	4%	1
الأسبوع الحادي عشر	ديمو رجان	%6	%6.66	6%	5%	5%	4%	1
الأسبوع الثاني عشر	SOP	%7	%6.66	6%	5%	5%	4%	1
الأسبوع الثالث عشر	POS	%5	%6.66	6%	5%	5%	4%	1
الأسبوع الرابع عشر	K-Map	%12	%6.66	6%	5%	5%	4%	1
الأسبوع الخامس عشر	BCD	%5	%6.66	6%	5%	5%	4%	1
المجموع	15	%100	%100	%90	%75	%75	%60	**15**

المحتويات (لكل فصل في المقرر)

رقم المحاضرة:	
عنوان المحاضرة:	
اســـــم المدرس:	
الفئة المستهدفة:	
الهدف العام من المحاضرة:	
الأهداف السلوكية او مخرجات التعلم:	-1 -2 -3
استراتيجيات التيسير المستخدمة	
المهارات المكتسبة	
طرق القياس المعتمدة	

1. الاسئلة القبلية
2. المحتوى العلمي
3. محتويات الفصل
4. الأسئلة البعدية في نهاية الحقيبة

- **المصادر الاساسية :**

1- **Teorey, T. J., Lightstone, S. S., Nadeau, T., & Jagadish, H. V. (2011). Database modeling and design: logical design. Elsevier.**

2- **Armstrong, J. R., & Gray, F. G. (1993). Structured logic design with VHDL. Prentice-Hall, Inc.**

3- **Bergmann, M., Moor, J., & Nelson, J. (1998). The logic book (Vol. 3). New York: McGraw-Hill.**

- **المصادر المقترحة:**

1- **Holdsworth, B., & Woods, C. (2002). Digital logic design. Elsevier.**

2- **LEWIN, D. P. D. (2013). Design of logic systems. Springer.**

الفصل الأول

الأنظمة العددية

تقسم الأنظمة العددية الى أربعة اقسام

1- النظام العشري (Decimal)

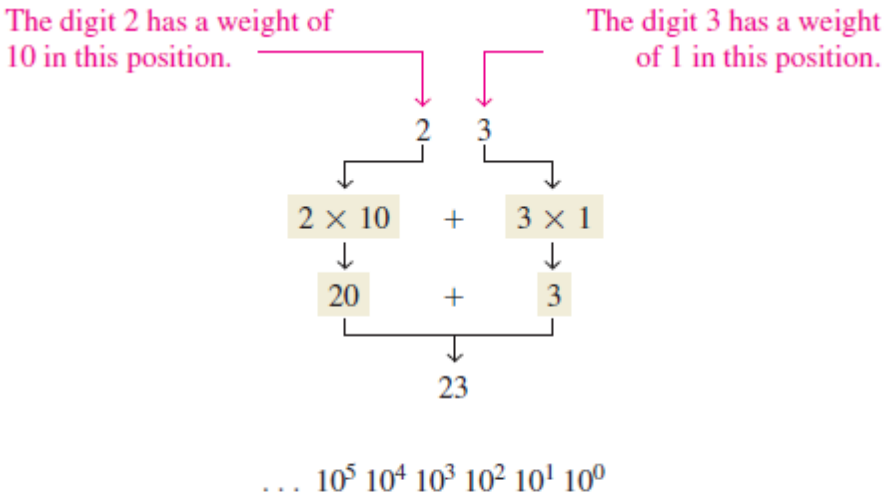
2- النظام الثنائي (Binary)

3- النظام الثماني (Octal)

4- النظام السادس عشر (Hexadecimal)

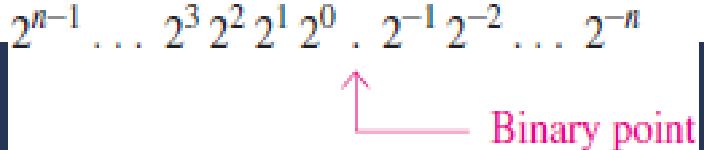
1- النظام العشري

في نظام الأعداد العشرية، يمثل كل من الأرقام العشرة (0 إلى 9) كمية معينة. كما تعرف، الرموز العشرة (الأرقام) لا تحدك للتعبير عن عشر كميات مختلفة فقط، لأنك تستخدم الأرقام المختلفة في المواضع المناسبة داخل الرقم للإشارة إلى مقدار الكمية. يمكنك التعبير عن الكميات حتى تسعة قبل نفاذ الأرقام؛ إذا كنت ترغب في التعبير عن كمية أكبر من تسعة، فإنك تستخدم رقمين أو أكثر، وموقع كل رقم داخل الرقم يخبرك بمقدار الكمية التي يمثلها. إذا كنت ترغب على سبيل المثال في التعبير عن الكمية ثلاثة وعشرين، فإنك تستخدم الرقم 2 للإشارة إلى كمية العشرين والرقم 3 للإشارة إلى كمية الثلاثة.



2- النظام الثنائي

نظام الأعداد الثنائي هو نظام عددي يستخدم فقط الرقمين 0 و 1 لتمثيل الأعداد. يعتبر هذا النظام أساسيًا في علم الحاسوب وتقنية المعلومات. يستخدم النظام الثنائي في تخزين ومعالجة البيانات في الحاسوب وغالبًا ما يكون النظام الثنائي أكثر كفاءة من النظام العشري في تمثيل البيانات. لتحويل الأعداد من النظام العشري إلى النظام الثنائي، يتم تقسيم العدد العشري على 2 وحفظ باقي القسمة كرقم ثنائي، ثم يتم تكرار هذه العملية حتى يصبح القسم المقسوم صفرًا.



Binary weights.

Positive Powers of Two (Whole Numbers)									Negative Powers of Two (Fractional Number)					
2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
256	128	64	32	16	8	4	2	1	1/2 0.5	1/4 0.25	1/8 0.125	1/16 0.625	1/32 0.03125	1/64 0.015625

الأرقام الثنائية هي التي تحتوي على الأرقام 0 و1 فقط. وتعبر كل رقم ثنائي عن حالة فردية معينة. على سبيل المثال، الرقم الثنائي 101 يعبر عن القيمة 5 في النظام العشري. ويمكن تمثيل الأعداد الثنائية في شكل سلاسل رقمية تتكون من الأرقام 0 و1 فقط، حيث يمثل كل موضع في السلسلة قيمة مضروبة في قوة معينة من 2.

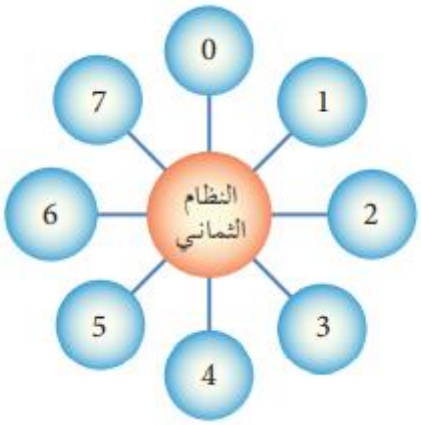
يمكن تحويل الأعداد من النظام العشري إلى النظام الثنائي باستخدام طرق مختلفة مثل قسمة العدد العشري على 2 وتحويل الأعداد بشكل يدوي أو باستخدام الأدوات الحاسوبية. بينما يمكن تحويل الأعداد الثنائية إلى العشرية عن طريق تفسير كل رقم ثنائي على أنه قيمة مضروبة في قوة معينة من 2 وجمعها معًا للحصول على القيمة العشرية المكافئة.

جدول الأرقام المكافئة للأرقام بالنظام العشري وما يقابلها بالنظام الثنائي

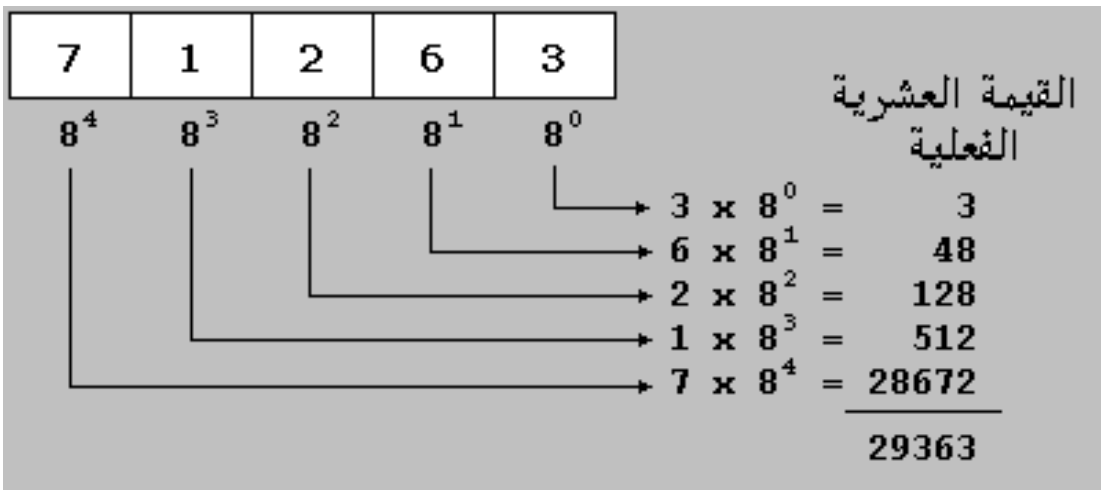
Decimal Number	Binary Number			
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

3- النظام الثماني

الأرقام الثمانية هي الأرقام التي يعتمدها النظام الثماني في تمثيل الأعداد، وهي الأرقام (0، 1، 2، 3، 4، 5، 6، 7). وتمثل هذه الأرقام قيم مختلفة بشكل مماثل للأرقام في النظام العشري، ولكن بالأرقام الثمانية. ويتم استخدام الأرقام الثمانية في الحوسبة وتخزين البيانات والعمليات الحسابية اللاحقة.



عملية التحويل بين النظام العشري والنظام الثماني تتضمن تحويل الأعداد من تمثيلها بالأرقام العشرية إلى تمثيلها بالأرقام الثمانية والعكس. يتطلب ذلك استخدام قواعد محددة لتحويل القيم بين النظامين دون فقدان الدقة. وتعد عملية التحويل بين النظامين مهمة في عالم الحوسبة وهندسة البرمجيات حيث قد تتطلب الحاجة إلى تمثيل البيانات بصيغة معينة بناءً على متطلبات النظام أو التطبيق.



4- النظام السادس عشر

❖ Decimal to Binary conversion:

EX:

- $(50)_{10}$ $(?)_2$

2	50	0	LSB
2	25	1	
2	12	0	
2	6	0	
2	3	1	
2	1	1	MSB

The answer is $(110010)_2$

❖ Decimal to Octal conversion:

- $(30.5)_{10} \longrightarrow (?)_8$

8	30	6	LSB	$0.5 \times 8 = 4.0$
8	3	3	MSB	

The answer is $(36.4)_8$

❖ Decimal to Hexadecimal conversion:

EX:

- $(28.3)_{10} \longrightarrow (?)_{16}$

16	28	12	LSB	$0.3 \times 16 = 4.8$	4
16	1	1	MSB	$0.8 \times 16 = 12.8$	12
				$0.8 \times 16 = 12.8$	12

The answer is $(1C.4CC)_{16}$

❖ Binary to Decimal conversion:

EX:

- $(110)_2$ $(?)_{10}$

$$1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 6$$

The answer is $(6)_{10}$

- $(1011.1100)_2 (?)_{10}$

$$1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 . 1 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 0 \times 2^{-4} = 11.75$$

The answer is $(11.75)_{10}$

❖ Binary to Octal conversion:

each three bits from right to left represent a number.

EX:

- $(111010011)_2 \longrightarrow (723)_8$
- $(110100001)_2 \longrightarrow (641)_8$

❖ Binary to Hexadecimal conversion:

each four bits from right to left represent a number.

- $(1101110001010001)_2 \longrightarrow (DC51)_{16}$

❖ Octal to Decimal conversions:

- $(371)_8 \longrightarrow (?)_{10}$

$1 \times 8^0 + 7 \times 8^1 + 3 \times 8^2 = 249$

The answer is $(249)_{10}$

❖ Octal to binary conversion:

each number is representing in binary using three bits.

- $(752)_8 \longrightarrow (111101010)_2$
- $(631)_8 \longrightarrow (110011001)_2$

EX:

- $(1BF)_{16} \longrightarrow (?)_{10}$

$F \times 16^0 + B \times 16^1 + 1 \times 16^3 = 447$

The answer is $(447)_{10}$

❖ **exadecimal to decimal conversion:**

❖ **Hexadecimal to Binary conversion:**

number is representing in binary using four bits.

EX:

- $(A35C)_{16} \longrightarrow (1010001101011100)_2$

Arithmetic Operation in number systems

Decimal arithmetic: can be applied the arithmetic operation in decimal number system. But the Subtraction using complement

The Subtraction using 9's complement and 10's complement:

Firstly, find the 9's complement and 10's complement

The equation to find 9's complement is

$$r^n - r^{-m} - N$$

The equation to find **10's complement** is

$$r^n - N$$

Ex:

Number	7	19	43.7
9's	2	80	56.2
10's	3	81	56.3

To Subtraction using **9's complement**

1. Take the 9's for negative number
2. The operation convert from subtract to add
3. If appear carry, move this carry; no carry take 9's for result.

Ex:

When smaller number is to be subtracted from larger one

Regular subtraction	Subtraction using 9's complement
678 - 234 444	678 +765 ← (9's complement of 234) 1443 + 1 444

When larger number is to be subtracted from smaller one

Regular subtraction	Subtraction using 9's complement
228 - 485 - 257	228 +514 ← (9's complement of 485) 742 (No carry indicates –ve value) ↓ - 257 (9's complement of result)

To Subtraction using **10's complement**

1. Take the 10's for negative number
2. The operation convert from subtract to add
3. If appear carry, ignore this carry; no carry take the 10's for result.

Ex:

When smaller number is to be subtracted from larger one

Regular subtraction

```
 678
-234
 444
```

Subtraction using 10's complement

```
 678
+766  ←(10's complement of 234)
(1)444 ←(Ignore the carry)
  ↓
 444
```

When larger number is to be subtracted from smaller one

Regular subtraction

```
 228
-485
-257
```

Subtraction using 10's complement

```
 228
+515  ←(10's complement of 485)
 743  (No carry indicates –ve value)
  ↓
-257 (10's complement of result)
```

Binary Arithmetic:

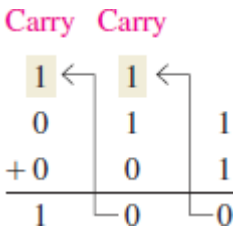
Binary arithmetic is essential in all digital computers and in many other types of digital systems

- ✓ **Binary Addition**

The four basic rules for adding binary digits (bits) are as follows:

0 + 0 = 0	Sum of 0 with a carry of 0
0 + 1 = 1	Sum of 1 with a carry of 0
1 + 0 = 1	Sum of 1 with a carry of 0
1 + 1 = 10	Sum of 0 with a carry of 1

Ex:

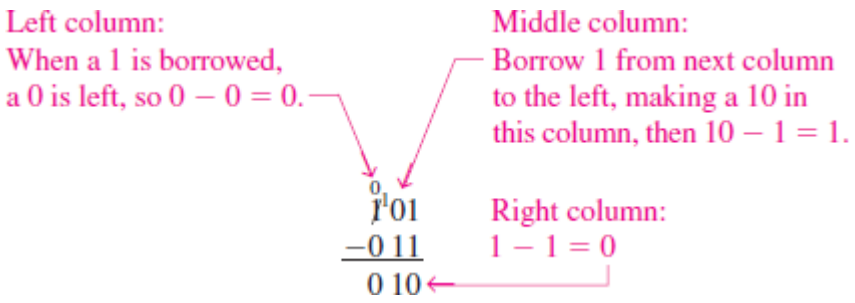


- ✓ **Binary Subtraction**

The four basic rules for subtracting bits are as follows:

$0 - 0 = 0$
$1 - 1 = 0$
$1 - 0 = 1$
$10 - 1 = 1$ $\quad 0 - 1$ with a borrow of 1

Ex:



The Subtraction using 1's complement and 2's complement:

Firstly, find the 1's complement and 2's complement The equation to find 1's complement is

$$r^n - r^{-m} - N$$

The equation to find 2's complement is

$$r^n - N$$

Ex:

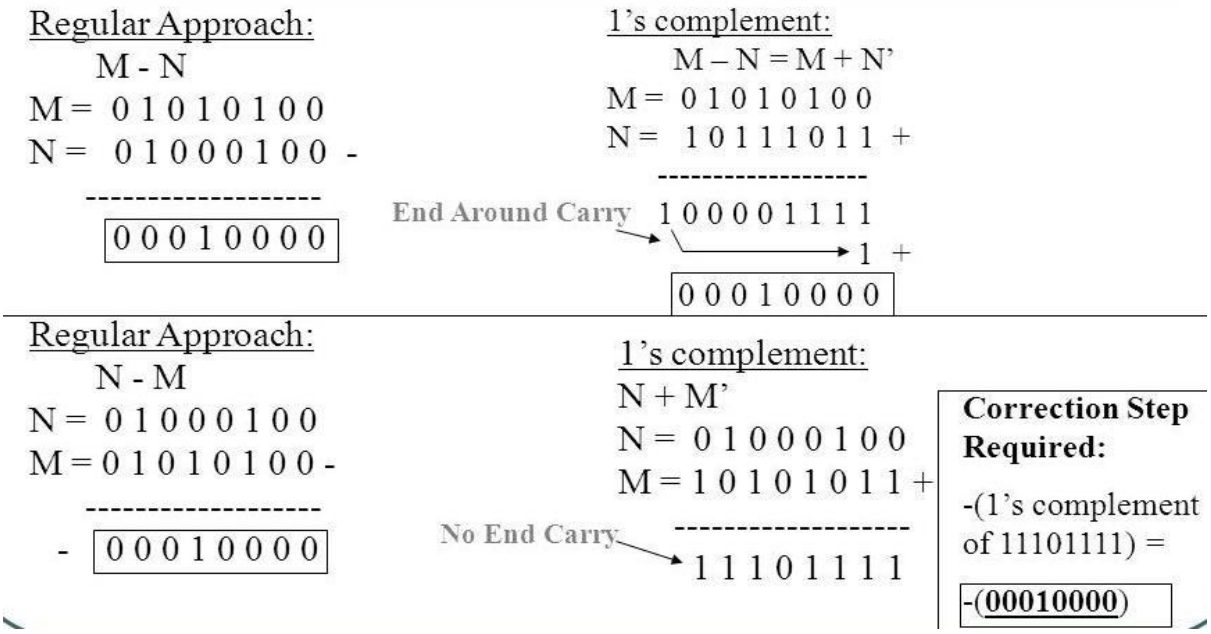
Number	111	1010
1's	000	0101
2's	001	0110

To Subtraction using **1's complement**

1. Take the 1's for negative number
2. The operation convert from subtract to add
3. If …carry, move this carry.

...no carry ,take 1's for final number

Ex:



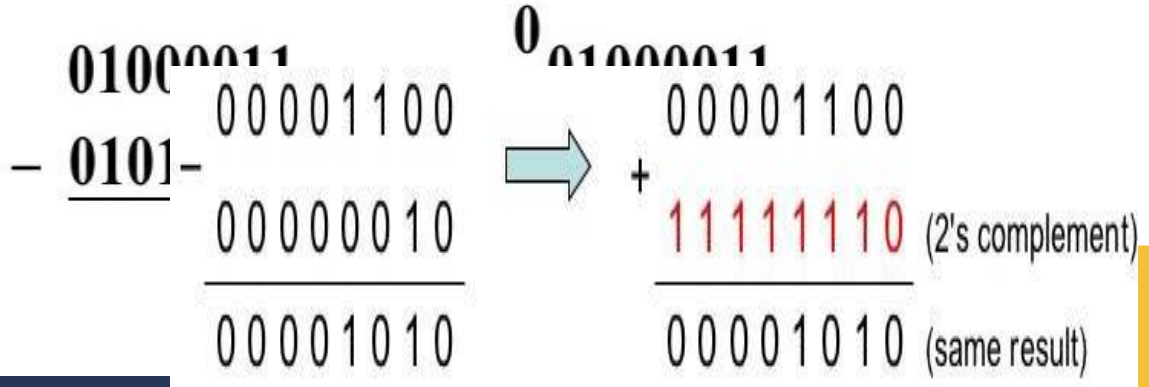
To Subtraction using **2's complement**

1. Take the 2's for negative number
2. The operation convert from subtract to add
3. If …carry, ignore this carry.

...no carry ,take 2's for final number

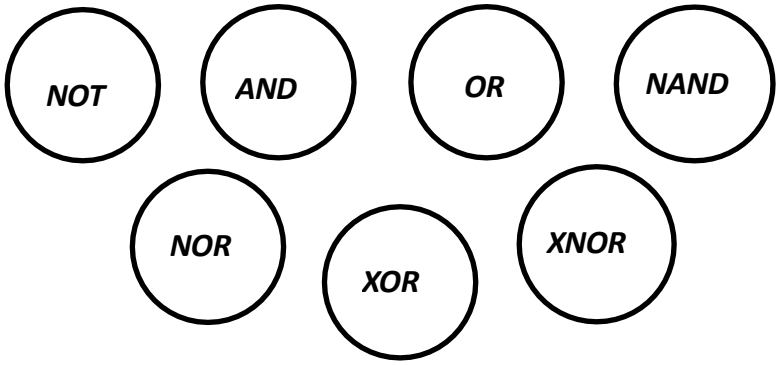
Ex:

- **Find $01000011_2 - 01010100_2$**



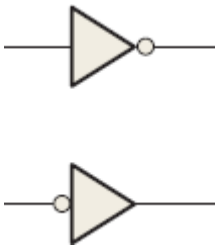
Logic gates

The logic gate is the basic building block in digital systems. Logic gates operate with binary numbers. Gates are therefore referred to as binary logic gates. All voltages used with logic gates will be either HIGH or LOW. In this lecture, a HIGH voltage will mean a binary 1. A LOW voltage will mean a binary 0.



- **The Inverter Gate:**

The inverter (NOT circuit) performs the operation called inversion or complementation. The inverter changes one logic level to the opposite level. In terms of bits, it changes a 1 to a 0 and a 0 to a 1. Standard logic symbols for the inverter are shown



Truth Table

Input A	Output B
0	1
1	0

- **The AND Gate:**

The AND gate is one of the basic gates that can be combined to form any logic

function. An AND gate can have two or more inputs and performs what is known as logical multiplication. The standard logic symbol

Input A	Input B	Output X
0	0	0
0	1	0
1	0	0
1	1	1

- **The OR Gate**

The OR gate is another of the basic gates from which all logic functions are constructed. An OR gate can have two or more inputs and performs what is known as logical addition.
The standard logic symbols

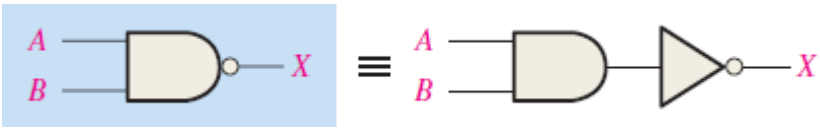
Truth Table
For a 2-input OR gate, output X is HIGH when either input A or input B is HIGH, or when both A and B are HIGH; X is LOW only when both A and B are LOW.

Input A	Input B	Output X
0	0	0
0	1	1
1	0	1
1	1	1

- **The NAND Gate:**

The NAND gate is a popular logic element because it can be used as a universal gate; that is, NAND gates can be used in combination to perform the AND, OR, and inverter

operations. The term NAND is a contraction of NOT-AND and implies an AND function with a complemented (inverted) output. The standard logic symbol



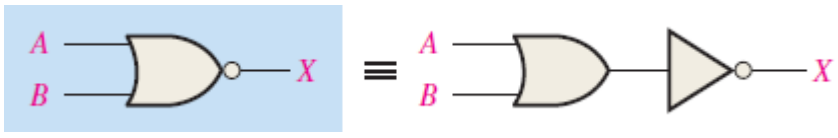
Truth Table

For a 2-input NAND gate, output X is LOW only when inputs A and B are HIGH; X is HIGH when either A or B is LOW, or when both A and B are LOW.

Input A	Input B	Output X
0	0	1
0	1	1
1	0	1
1	1	0

- **The NOR Gate**

The NOR gate, like the NAND gate, is a useful logic element because it can also be used as a universal gate; that is, NOR gates can be used in combination to perform the AND, OR, and inverter operations. The term NOR is a contraction of NOT-OR and implies an OR function with an inverted (complemented) output. The standard logic symbol



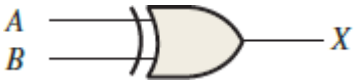
Truth Table

For a 2-input NOR gate, output X is LOW when either input A or input B is HIGH, or when both A and B are HIGH; X is HIGH only when both A and B are LOW.

Input A	Input B	Output X
0	0	1
0	1	0
1	0	0
1	1	0

The Exclusive-OR Gate:

The exclusive-OR gate performs modulo-2 addition. The XOR gate has only two inputs and the standard symbols for an exclusive-OR (XOR for short) gate



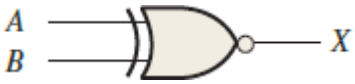
Truth table:

For an exclusive-OR gate, output X is HIGH when input A is LOW and input B is HIGH, or when input A is HIGH and input B is LOW; X is LOW when A and B are both HIGH or both LOW.

Input A	Input B	Output X
0	0	0
0	1	1
1	0	1
1	1	0

- **The Exclusive-NOR Gate**

Like the XOR gate, an XNOR has only two inputs. Standard symbols for an exclusive- NOR (XNOR) gate



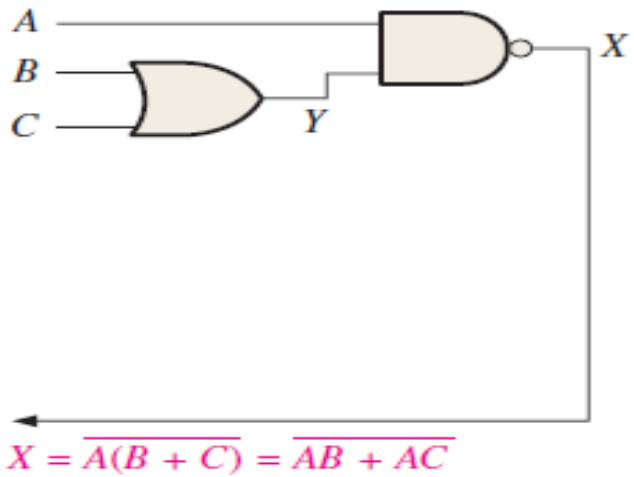
Truth table:

For an exclusive-NOR gate, output X is LOW when input A is LOW and input B is HIGH, or when A is HIGH and B is LOW; X is HIGH when A and B are both HIGH or both LOW.

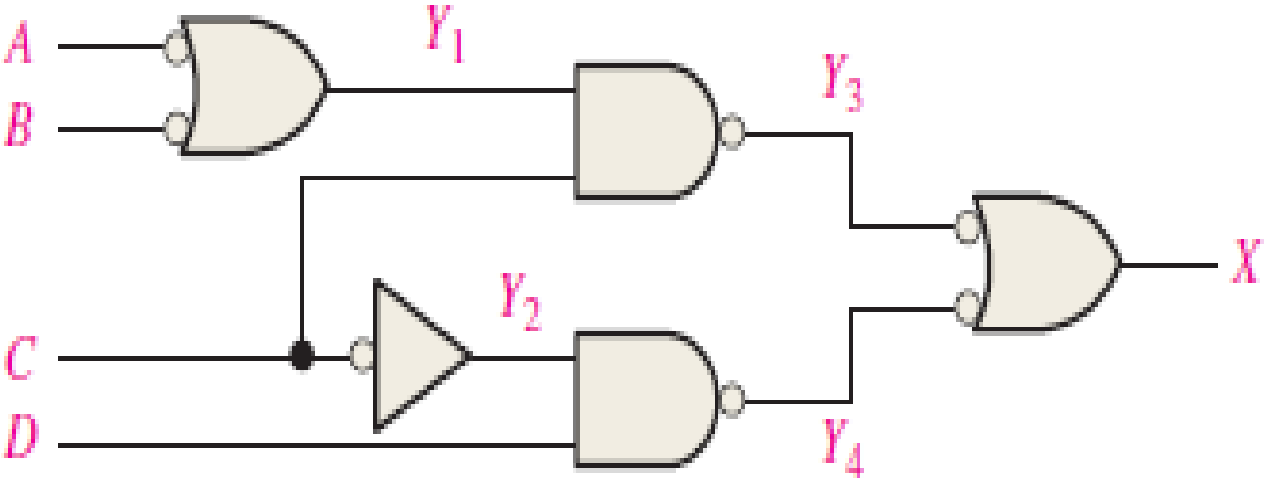
Input A	Input B	Output X
0	0	1
0	1	0
1	0	0
1	1	1

Pulse Waveform Operation

The operation of each gate is the same for pulse waveform inputs as for constant-level inputs. The output of a logic circuit at any given time depends on the inputs at that particular time, so the relationship of the time-varying inputs is of primary importance Ex:



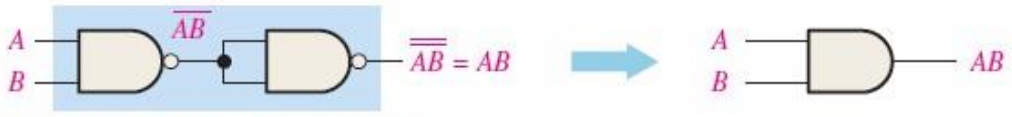
$$X = \overline{A(B + C)} = \overline{AB + AC}$$



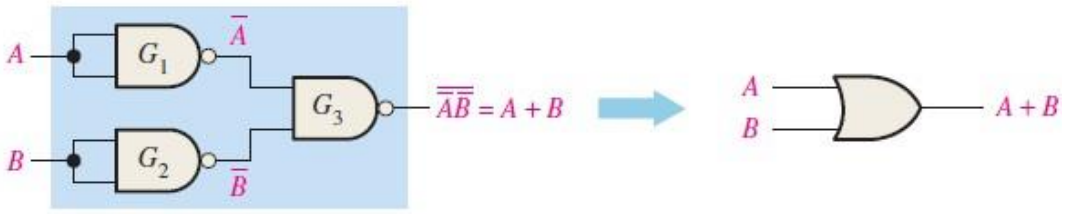
The NAND Gate as a Universal Logic Element

The NAND gate is a universal gate because it can be used to produce the NOT, the AND, the OR, and the NOR functions. An inverter can be made from a NAND gate by connecting all of the inputs together and creating, in effect, a single input, as shown in Figure (a) for a 2-input gate. An AND function can be generated by the use of NAND gates alone, as shown in Figure (b). An OR function can be produced with only NAND gates, as illustrated in part (c). Finally, a NOR function is produced as shown in part (d).

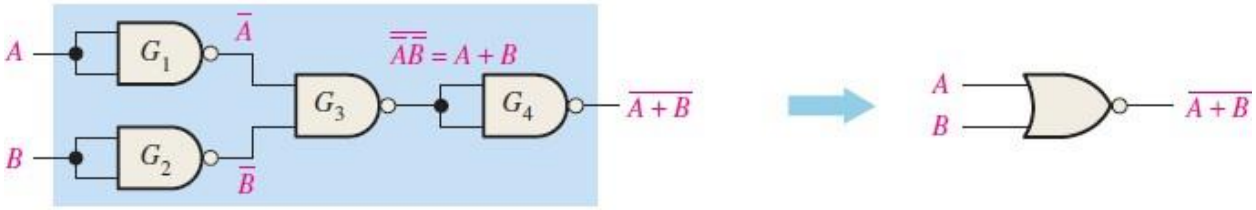
(a) One NAND gate used as an inverter



(b) Two NAND gates used as an AND gate



(c) Three NAND gates used as an OR gate



(d) Four NAND gates used as a NOR gate

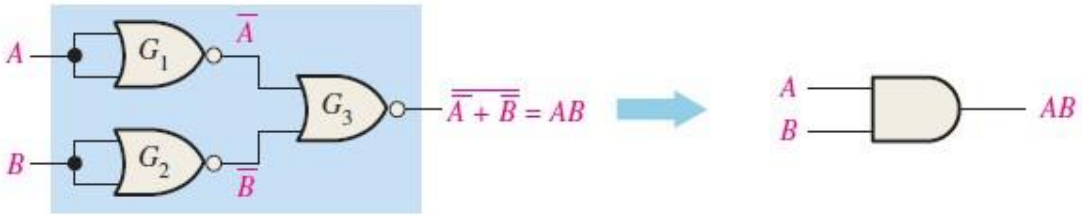
The NOR Gate as a Universal Logic Element

Like the NAND gate, the NOR gate can be used to produce the NOT, AND, OR, and

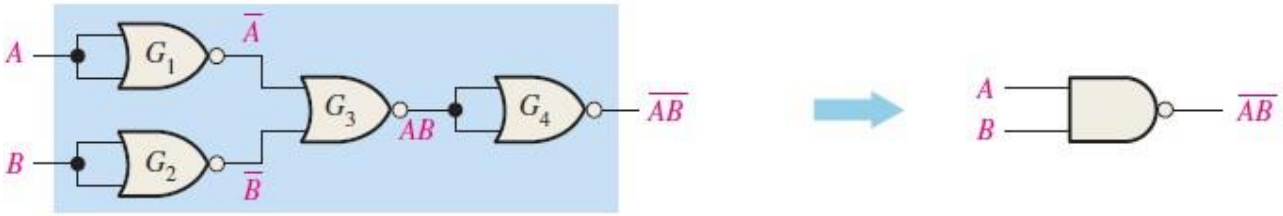
NAND functions. A NOT circuit, or inverter, can be made from a NOR gate by connecting all of the inputs together to effectively create a single input, as shown in Figure(a) with a 2-input example. Also, an OR gate can be produced from NOR gates, as illustrated in Figure (b). An AND gate can be constructed by the use of NOR gates, as shown in Figure(c). And Figure (d) shows how NOR gates are used to form a NAND function.

(a) One NOR gate used as an inverter

(b) Two NOR gates used as an OR gate



(c) Three NOR gates used as an AND gate



(d) Four NOR gates used as a NAND gate

Boolean Algebra and Logic Simplification

Boolean algebra is the mathematics of digital logic. A basic knowledge of Boolean algebra is indispensable to the study and analysis of logic circuits

Laws and Rules of Boolean Algebra

As in other areas of mathematics, there are certain well-developed rules and laws that must be followed in order to properly apply Boolean algebra

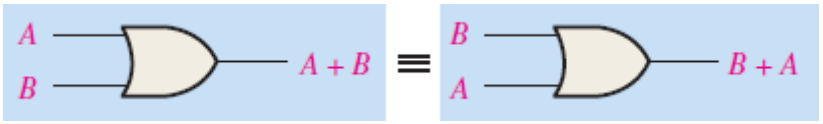
Laws of Boolean Algebra:

The basic laws of Boolean algebra—the commutative laws for addition and multiplication, the associative laws for addition and multiplication, and the distributive law—are the same as in ordinary algebra. Each of the laws is illustrated with two or three variables, but the number of variables is not limited to this.

- **Commutative Laws**

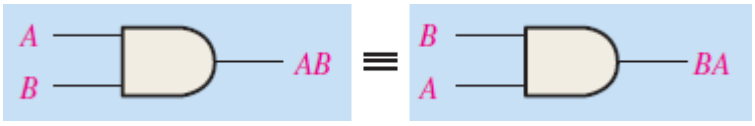
The commutative law of addition for two variables is written as

$$A + B = B + A$$



The commutative law of multiplication for two variables is

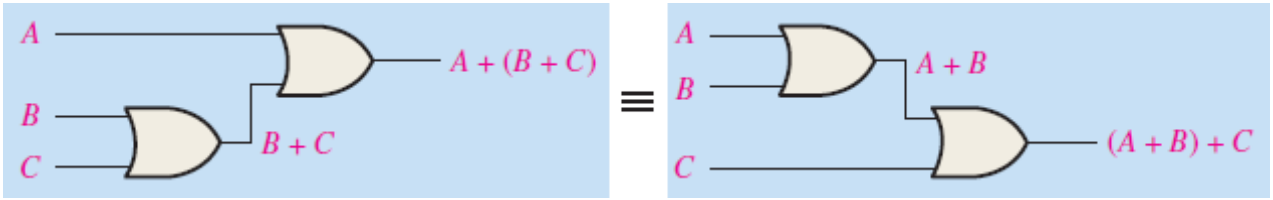
$$AB = BA$$



- **Associative Laws**

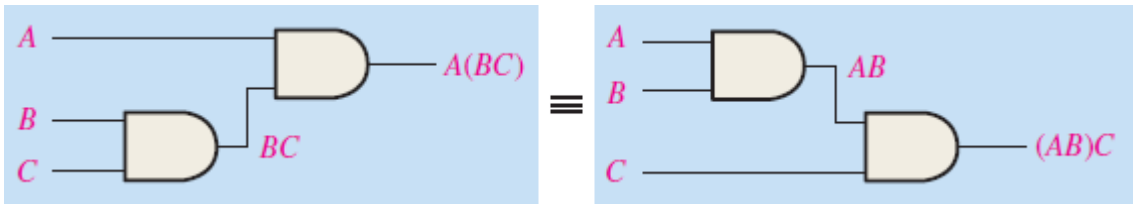
The associative law of addition is written as follows for three variables:

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



The associative law of multiplication is written as follows for three variables:

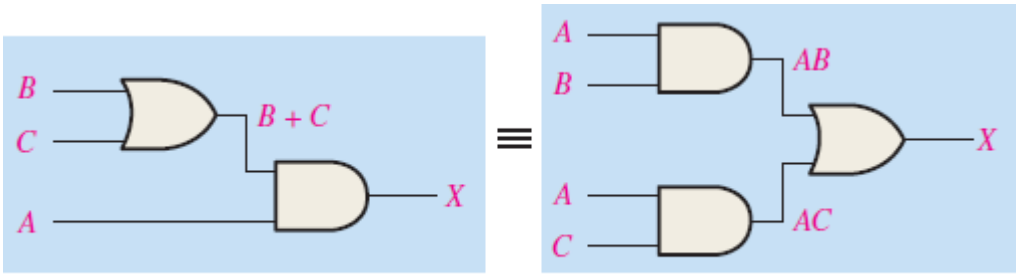
$$A(BC) = (AB)C$$



- **Distributive Law**

The distributive law is written for three variables as follows:

$$A(B + C) = AB + AC$$



$X = A(B + C)$ $X = AB + AC$

Rules of Boolean Algebra

Table below lists 12 basic rules that are useful in manipulating and simplifying Boolean
expressions. Rules 1 through 9 will be viewed in terms of their application to logic gates.
Rules 10 through 12 will be derived in terms of the simpler rules.

Basic rules of Boolean algebra.

1. $A + 0 = A$	7. $A \cdot A = A$
2. $A + 1 = 1$	8. $A \cdot \overline{A} = 0$
3. $A \cdot 0 = 0$	9. $\overline{\overline{A}} = A$
4. $A \cdot 1 = A$	10. $A + AB = A$
5. $A + A = A$	11. $A + \overline{A}B = A + B$
6. $A + \overline{A} = 1$	12. $(A + B)(A + C) = A + BC$

A, B, or C can represent a single variable or a combination of variables.

Ex: Using Boolean algebra techniques, simplify this expression:

$$AB + A(B + C) + B(B + C)$$

Step 1: Apply the distributive law to the second and third terms in the expression, as follows:

$$AB + AB + AC + BB + BC$$

Step 2: Apply rule 7 ($BB = B$) to the fourth term.

$$AB + AB + AC + B + BC$$

Step 3: Apply rule 5 ($AB + AB = AB$) to the first two terms.

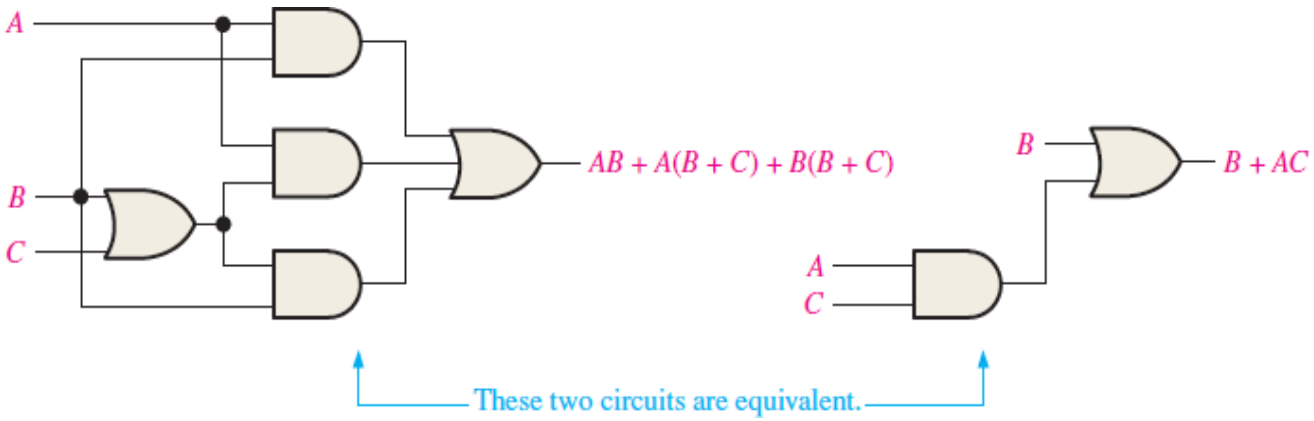
$$AB + AC + B + BC$$

Step 4: Apply rule 10 ($B + BC = B$) to the last two terms.

$$AB + AC + B$$

Step 5: Apply rule 10 ($AB + B = B$) to the first and third terms.

$$B + AC$$



DeMorgan's Theorems

DeMorgan, a mathematician who knew Boole, proposed two theorems that are an important part of Boolean algebra. In practical terms, DeMorgan's theorems provide mathematical verification of the equivalency of the NAND and negative-OR gates and the equivalency of the NOR and negative-AND gates

- DeMorgan's first theorem is stated as follows:

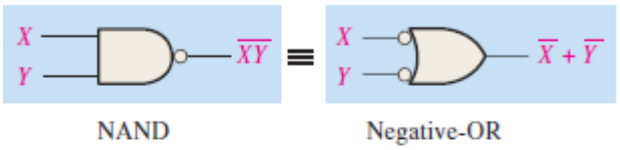
The complement of a product of variables is equal to the sum of the complements of the variables.

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

- DeMorgan's second theorem is stated as follows:

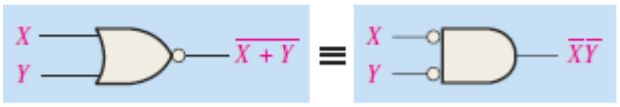
The complement of a sum of variables is equal to the product of the complements of the variables.

$$\overline{X + Y} = \overline{X}\,\overline{Y}$$



NAND Negative-OR

Inputs		Output	
X	Y	$\overline{XY}$	$\overline{X}+\overline{Y}$
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0



NOR Negative-AND

Inputs		Output	
X	Y	$\overline{X+Y}$	$\overline{X}\,\overline{Y}$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

Ex: Apply DeMorgan's theorems to each expression:

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

Step 1: Identify the terms to which you can apply DeMorgan's theorems, and think of each term as a single variable. Let $\overline{A + B\overline{C}} = X$ and $D(\overline{E + \overline{F}}) = Y$.

Step 2: Since $\overline{X + Y} = \overline{X}\,\overline{Y}$,

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

Step 3: Use rule 9 ($\overline{\overline{A}} = A$) to cancel the double bars over the left term (this is not part of DeMorgan's theorem).

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

Step 4: Apply DeMorgan's theorem to the second term.

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

Step 5: Use rule 9 ($\overline{\overline{A}} = A$) to cancel the double bars over the $E + \overline{F}$ part of the term.

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

Ex: Simplify the following Boolean expression:

$$[A\overline{B}(C + BD) + \overline{A}\,\overline{B}]C$$

Step 1: Apply the distributive law to the terms within the brackets.

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

Step 2: Apply rule 8 ($\overline{B}B = 0$) to the second term within the parentheses.

$$(A\overline{B}C + A \cdot 0 \cdot D + \overline{A}\,\overline{B})C$$

Step 3: Apply rule 3 ($A \cdot 0 \cdot D = 0$) to the second term within the parentheses.

$$(A\overline{B}C + 0 + \overline{A}\,\overline{B})C$$

Step 4: Apply rule 1 (drop the 0) within the parentheses.

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

Step 5: Apply the distributive law.

$$A\overline{B}CC + \overline{A}\,\overline{B}C$$

Step 6: Apply rule 7 ($CC = C$) to the first term.

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

Step 7: Factor out $\overline{B}C$.

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

Step 8: Apply rule 6 ($A + \overline{A} = 1$).

$$\overline{B}C \cdot 1$$

Step 9: Apply rule 4 (drop the 1).

$$\overline{B}C$$

Standard Forms of Boolean Expressions

All Boolean expressions, regardless of their form, can be converted into either of two standard forms: the sum-of-products form or the product-of-sums form

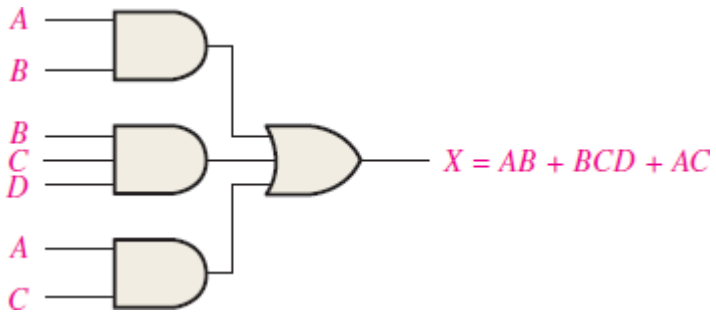
- The Sum-of-Products (SOP) Form

a term consisting of the product (Boolean multiplication) of literals (variables or their complements). When two or more product terms are summed by Boolean addition, the resulting expression is a sum-of-products (SOP). Symbol ∑
Some examples are

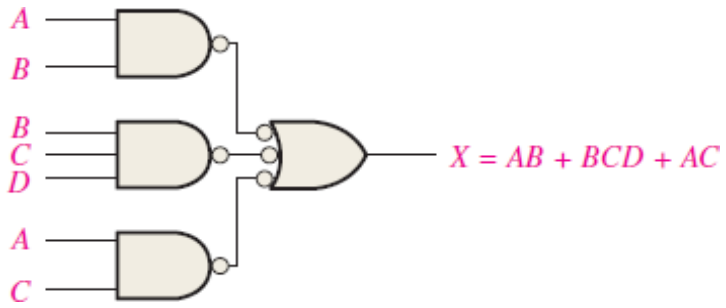
$$AB + ABC$$
$$ABC + CDE + \overline{B}C\overline{D}$$
$$\overline{A}B + \overline{A}B\overline{C} + AC$$

Implementation of a SOP Expression

Implementing an SOP expression simply requires ORing the outputs of two or more AND gates. A product term is produced by an AND operation, and the sum (addition) of two or more product terms is produced by an OR operation. Therefore, an SOP expression can be implemented by AND-OR logic in which the outputs of a number (equal to the number of product terms in the expression) of AND gates connect to the inputs of an OR gate, as shown in figure below for the expression AB + BCD + AC. The output X of the OR gate equals the SOP expression.



NAND gates can be used to implement an SOP expression. By using only NAND gates, an AND/OR function can be accomplished, as illustrated in figure below. The first level of NAND gates feed into a NAND gate that acts as a negative-OR gate. The NAND and negative-OR inversions cancel and the result is effectively an AND/OR Circuit.



Converting Product Terms to Standard SOP

Each product term in an SOP expression that does not contain all the variables in the domain can be expanded to standard form to include all variables in the domain and their
complements. As stated in the following steps, a nonstandard SOP expression is converted into standard form using Boolean algebra rule 6 $(A + \overline{A} = 1)$

Step 1: Multiply each nonstandard product term by a term made up of the sum of a missing variable and its complement. This results in two product terms. As you know, you can multiply anything by 1 without changing its value.

Step 2: Repeat Step 1 until all resulting product terms contain all variables in the domain in either complemented or uncomplemented form. In converting a product term to standard form, the number of product terms is doubled for each missing variable.

Ex: Convert the following Boolean expression into standard SOP form:

$$A\overline{B}C + \overline{A}\,\overline{B} + AB\overline{C}D$$

In this case, four standard product terms are the result.

The third term, $AB\overline{C}D$, is already in standard form. The complete standard SOP form of the original expression is as follows:

$$A\overline{B}C + \overline{A}B + AB\overline{C}D = A\overline{B}CD + A\overline{B}\overline{C}\overline{D} + \overline{A}BCD + \overline{A}B\overline{C}D + \overline{A}\overline{B}CD + \overline{A}\overline{B}\overline{C}\overline{D} + AB\overline{C}D$$

Binary Representation of a Standard Product Term

A standard product term is equal to 1 for only one combination of variable values. For example, the product term $A\overline{B}C\overline{D}$ is equal to 1 when A = 1, B = 0, C = 1, D = 0, as shown below, and is 0 for all other combinations of values for the variables.

$$A\overline{B}C\overline{D} = 1 \cdot \overline{0} \cdot 1 \cdot \overline{0} = 1 \cdot 1 \cdot 1 \cdot 1 = 1$$

In this case, the product term has a binary value of 1010 (decimal ten).

Remember, a product term is implemented with an AND gate whose output is 1 only if each of its inputs is 1. Inverters are used to produce the complements of the variables as required.

An SOP expression is equal to 1 only if one or more of the product terms in the expression is equal to 1.

Ex: Determine the binary values for which the following standard SOP expression is equal to 1:

$$ABCD + A\overline{B}\overline{C}D + \overline{A}\overline{B}\overline{C}\overline{D}$$

The term $ABCD$ is equal to 1 when $A = 1, B = 1, C = 1$, and $D = 1$.

$$ABCD = 1 \cdot 1 \cdot 1 \cdot 1 = 1$$

The term $A\overline{B}\overline{C}D$ is equal to 1 when $A = 1, B = 0, C = 0$, and $D = 1$.

$$A\overline{B}\overline{C}D = 1 \cdot \overline{0} \cdot \overline{0} \cdot 1 = 1 \cdot 1 \cdot 1 \cdot 1 = 1$$

The term $\overline{A}\overline{B}\overline{C}\overline{D}$ is equal to 1 when $A = 0, B = 0, C = 0$, and $D = 0$.

$$\overline{A}\overline{B}\overline{C}\overline{D} = \overline{0} \cdot \overline{0} \cdot \overline{0} \cdot \overline{0} = 1 \cdot 1 \cdot 1 \cdot 1 = 1$$

The SOP expression equals 1 when any or all of the three product terms is 1.

Converting SOP Expressions to Truth Table Format

SOP expression is equal to 1 only if at least one of the product terms is equal to 1. A truth table is simply a list of the possible combinations of input variable values and the corresponding output values (1 or 0). For an expression with a domain of two variables, there are four different combinations of those variables ($2^2 = 4$).

For an expression with a domain of three variables, there are eight different combinations

of those variables ($2^3 = 8$). For an expression with a domain of four variables, there are sixteen different combinations of those variables ($2^4 = 16$), and so on.

The first step in constructing a truth table is to list all possible combinations of binary values of the variables in the expression. Next, convert the SOP expression to standard

form if it is not already. Finally, place a 1 in the output column (X) for each binary value
that makes the standard SOP expression a 1 and place a 0 for all the remaining binary values.

Ex: Develop a truth table for the standard SOP expression

$$\overline{A}\,\overline{B}C + A\overline{B}\,\overline{C} + ABC.$$

Inputs			Output	
A	*B*	*C*	*X*	Product Term
0	0	0	0	
0	0	1	1	$\overline{A}\,\overline{B}C$
0	1	0	0	
0	1	1	0	
1	0	0	1	$A\overline{B}\,\overline{C}$
1	0	1	0	
1	1	0	0	
1	1	1	1	ABC

➢ The Product-of-Sums (POS) Form

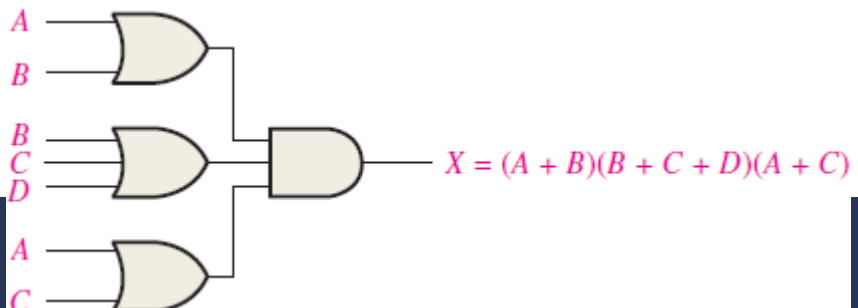
Term consisting of the sum (Boolean addition) of literals (variables or their complements). When two or more sum terms are multiplied, the resulting expression is a product-of-sums (POS). Symbol ∏

Some examples are

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

Implementation of a POS Expression

Implementing a POS expression simply requires ANDing the outputs of two or more OR gates. A sum term is produced by an OR operation, and the product of two or more sum terms is produced by an AND operation. Therefore, a POS expression can be implemented by logic in which the outputs of a number (equal to the number of sum terms in the expression) of OR gates connect to the inputs of an AND gate.



Converting a Sum Term to Standard POS

Each sum term in a POS expression that does not contain all the variables in the domain can be expanded to standard form to include all variables in the domain and their complements.

As stated in the following steps, a nc $(A \cdot \overline{A} = 0)$ OS expression is converted into standard form using Boolean algebra rule 8

Step 1: Add to each nonstandard product term a term made up of the product of the missing variable and its complement. This results in two sum terms. As you know, you can add 0 to anything without changing its value.

Step 2: Apply rule 12 : A + BC = (A + B)(A + C)

Step 3: Repeat Step 1 until all resulting sum terms contain all variables in the domain in either complemented or uncomplemented form.

Ex: Convert the following Boolean expression into standard POS form:

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

The domain of this POS expression is A, B, C, D. Take one term at a time. The first term, $A + \overline{B} + C$, is missing variable D or $\overline{D}$, so add $D\overline{D}$ and apply rule 12 as follows:

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

The second term, $\overline{B} + C + \overline{D}$, is missing variable A or $\overline{A}$, so add $A\overline{A}$ and apply rule 12 as follows:

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

The third term, $A + \overline{B} + \overline{C} + D$, is already in standard form. The standard POS form of the original expression is as follows:

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

Binary Representation of a Standard Sum Term

A standard sum term is equal to 0 for only one combination of variable values. For example,

the sum term $A + \overline{B} + C + \overline{D}$ is 0 when A = 0, B = 1, C = 0, and D = 1, as shown below, and is 1 for all other combinations of values for the variables.

$$A + \overline{B} + C + \overline{D} = 0 + \overline{1} + 0 + \overline{1} = 0 + 0 + 0 + 0 = 0$$

In this case, the sum term has a binary value of 0101 (decimal 5). Remember, a sum term is implemented with an OR gate whose output is 0 only if each of its inputs is 0. Inverters are used to produce the complements of the variables as required.

A POS expression is equal to 0 only if one or more of the sum terms in the expression is equal to 0.

Ex: Determine the binary values of the variables for which the following standard POS expression is equal to 0:

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

The term $A + B + C + D$ is equal to 0 when $A = 0, B = 0, C = 0$, and $D = 0$.

$$A + B + C + D = 0 + 0 + 0 + 0 = 0$$

The term $A + \overline{B} + \overline{C} + D$ is equal to 0 when $A = 0, B = 1, C = 1$, and $D = 0$.

$$A + \overline{B} + \overline{C} + D = 0 + \overline{1} + \overline{1} + 0 = 0 + 0 + 0 + 0 = 0$$

The term $\overline{A} + \overline{B} + \overline{C} + \overline{D}$ is equal to 0 when $A = 1, B = 1, C = 1$, and $D = 1$.

$$\overline{A} + \overline{B} + \overline{C} + \overline{D} = \overline{1} + \overline{1} + \overline{1} + \overline{1} = 0 + 0 + 0 + 0 = 0$$

The POS expression equals 0 when any of the three sum terms equals 0.

Converting POS Expressions to Truth Table Format

Recall that a POS expression is equal to 0 only if at least one of the sum terms is equal to 0. To construct a truth table from a POS expression, list all the possible combinations of binary values of the variables just as was done for the SOP expression. Next, convert the POS expression to standard form if it is not already.

Finally, place a 0 in the output column (X) for each binary value that makes the expression a 0 and place a 1 for all the remaining binary values.

Ex: Determine the truth table for the following standard POS expression:

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

Inputs			Output	
A	*B*	*C*	*X*	Sum Term
0	0	0	0	$(A + B + C)$
0	0	1	1	
0	1	0	0	$(A + \overline{B} + C)$
0	1	1	0	$(A + \overline{B} + \overline{C})$
1	0	0	1	
1	0	1	0	$(\overline{A} + B + \overline{C})$
1	1	0	0	$(\overline{A} + \overline{B} + C)$
1	1	1	1	

The Karnaugh Map

A Karnaugh map provides a systematic method for simplifying Boolean expressions and, if properly used, will produce the simplest SOP or POS expression possible. A Karnaugh map is similar to a truth table because it presents all of the possible values of input variables and the resulting output for each value. Instead of being organized into columns and rows like a truth table, the Karnaugh map is an array of cells in which each cell represents a binary value of the input variables. The cells are arranged in a way so that simplification of a given expression is simply a matter of properly grouping the cells.

Karnaugh maps can be used for expressions with two, three, four, and five variables, The number of cells in a Karnaugh map, as well as the number of rows in a truth table, is equal to the total number of possible input variable combinations. For three variables, the number of cells is $2^3 = 8$. For four variables, the number of cells is $2^4 = 16$.

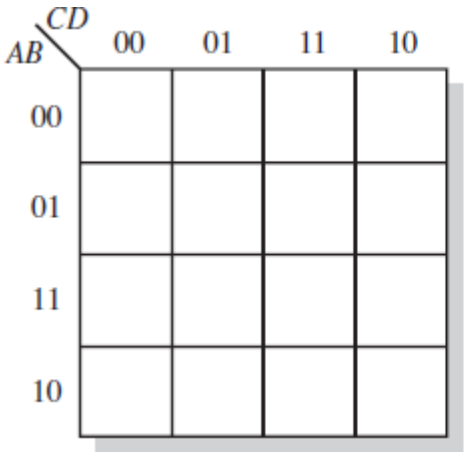
The 3-Variable Karnaugh Map

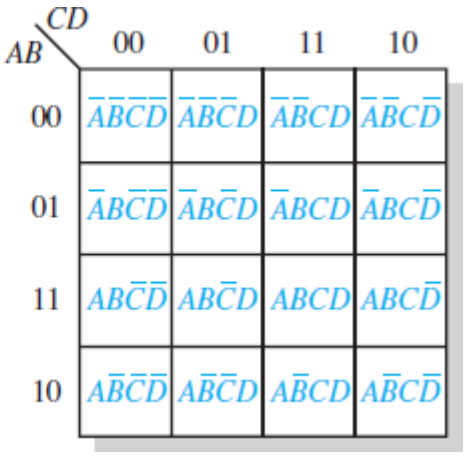
The 3-variable Karnaugh map is an array of eight cells. In this case, A, B, and C are used for the variables although other letters could be used.

A \ BC	00	01	11	10
0	A'B'C' 0	A'B'C 1	A'BC 3	A'BC' 2
1	AB'C' 4	AB'C 5	ABC 7	ABC' 6

The 4-Variable Karnaugh Map

The 4-variable Karnaugh map is an array of sixteen cells. Binary values of A and B are along the left side and the values of C and D are across the top.





Karnaugh Mapping a Standard SOP Expression

For an SOP expression in standard form, a 1 is placed on the Karnaugh map for each product term in the expression. Each 1 is placed in a cell corresponding to the value of a product term.

Ex: Map the following standard SOP expression on a Karnaugh map:

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

The standard product term in the expression.

$$\underset{001}{\overline{A}\,\overline{B}C} + \underset{010}{\overline{A}B\overline{C}} + \underset{110}{AB\overline{C}} + \underset{111}{ABC}$$

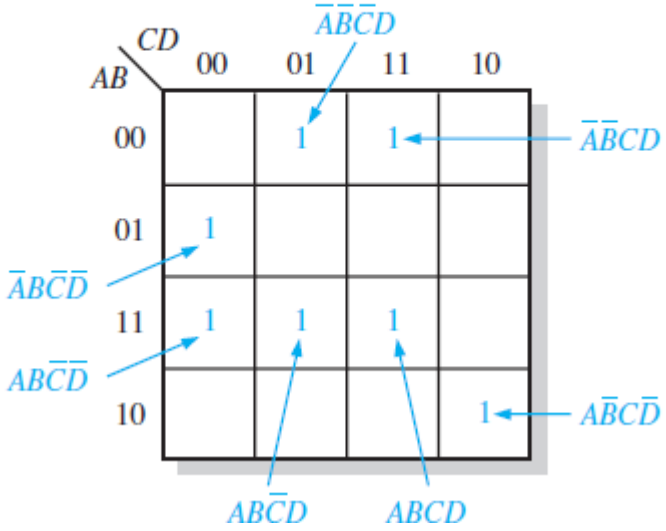
	$\overline{B}\overline{C}$	$\overline{B}C$	BC	$B\overline{C}$
$\overline{A}$		1	1	1
A			1	

Ex: Map the following standard SOP expression on a Karnaugh map:

$$\overline{A}\,\overline{B}CD + \overline{A}B\overline{C}\,\overline{D} + AB\overline{C}D + ABCD + AB\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + A\overline{B}CD$$

The standard product term in the expression.

$$\underset{0011}{\overline{A}\,\overline{B}CD} + \underset{0100}{\overline{A}B\overline{C}\,\overline{D}} + \underset{1101}{AB\overline{C}D} + \underset{1111}{ABCD} + \underset{1100}{AB\overline{C}\,\overline{D}} + \underset{0001}{\overline{A}\,\overline{B}\,\overline{C}D} + \underset{1010}{A\overline{B}CD}$$



Mapping a Nonstandard SOP Expression

A Boolean expression must first be in standard form before you use a Karnaugh map

Ex: Map the following SOP expression on a Karnaugh map:

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

The SOP expression is obviously not in standard form because each product term does not have three variables. The first term is missing two variables, the second term is missing one variable, and the third term is standard. First expand the terms numerically as follows:

$$\begin{array}{lll} \overline{A} & + A\overline{B} & + AB\overline{C} \\ 000 & 100 & 110 \\ 001 & 101 & \\ 010 & & \\ 011 & & \end{array}$$

Map each of the resulting binary values by placing a 1 in the appropriate cell of the 3-variable Karnaugh map

	$\overline{B}\overline{C}$	$\overline{B}C$	BC	$B\overline{C}$
$\overline{A}$	1	1	1	1
A	1	1		1

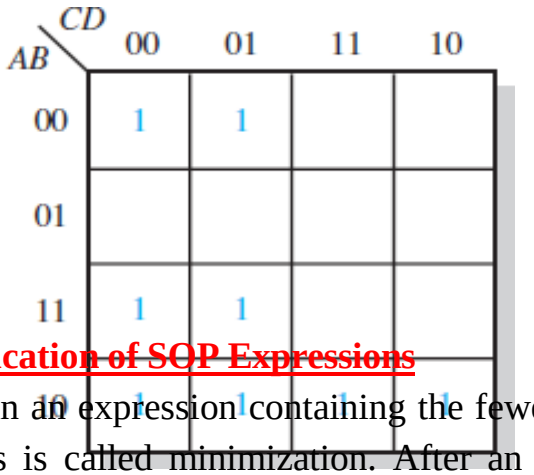
Ex: Map the following SOP expression on a Karnaugh map:

$$\overline{B}\overline{C} + A\overline{B} + AB\overline{C} + A\overline{B}C\overline{D} + \overline{A}B\overline{C}D + A\overline{B}CD$$

The SOP expression is obviously not in standard form because each product term does not have four variables. The first and second terms are both missing two variables, the third term is missing one variable, and the rest of the terms are standard. First expand the terms by including all combinations of the missing variables numerically as follows:

$\overline{B}\overline{C}$	$A\overline{B}$	$AB\overline{C}$	$A\overline{B}C\overline{D}$	$\overline{A}B\overline{C}D$	$A\overline{B}CD$
0000	1000	1100	1010	0001	1011
0001	1001	1101			
1000	1010				
1001	1011				

Map each of the resulting binary values by placing a 1 in the appropriate cell of the 4-variable Karnaugh map



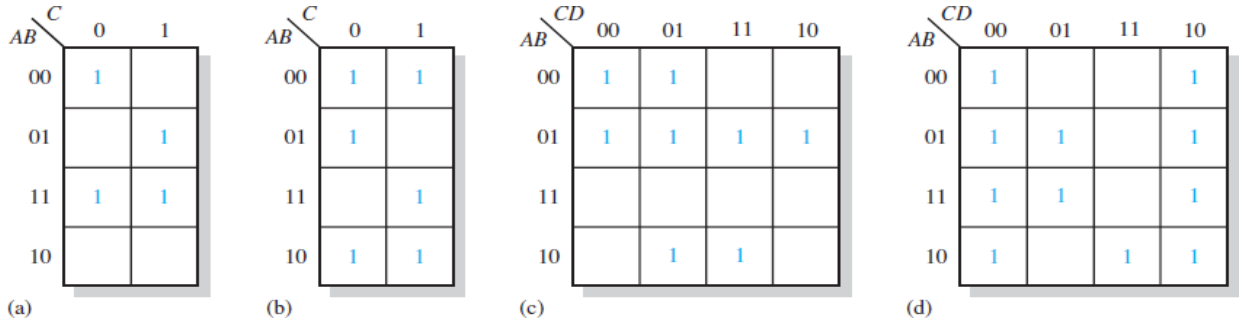
Karnaugh Map Simplification of SOP Expressions

The process that results in an expression containing the fewest possible terms with the fewest possible variables is called minimization. After an SOP expression has been mapped, a minimum SOP expression is obtained by grouping the 1s and determining the minimum SOP expression from the map. The Karnaugh map according to the following rules by enclosing those adjacent cells containing 1s. The goal is to

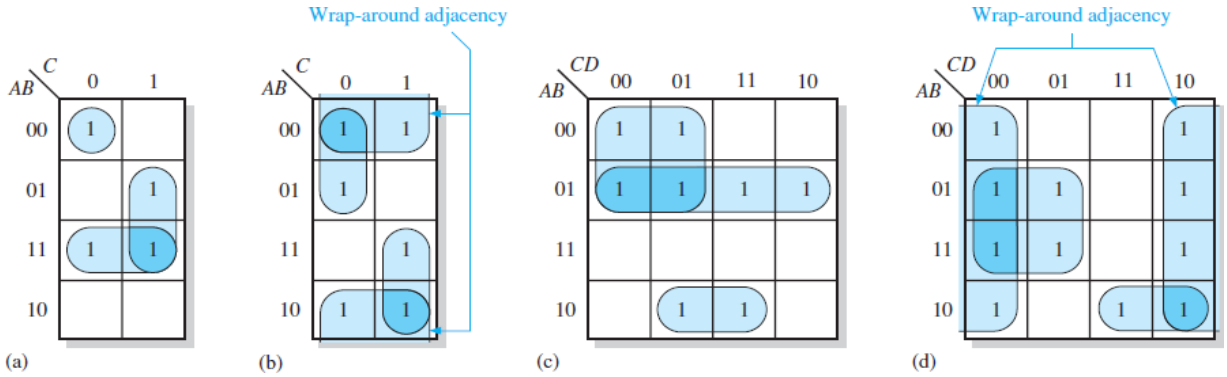
maximize the size of the groups and to minimize the number of groups.

1. A group must contain either 1, 2, 4, 8, or 16 cells, which are all powers of two. In the case of a 3-variable map, $2^3 = 8$ cells are the maximum group.
2. Each cell in a group must be adjacent to one or more cells in that same group, but all cells in the group do not have to be adjacent to each other.
3. Always include the largest possible number of 1s in a group in accordance with rule1.
4. Each 1 on the map must be included in at least one group. The 1s already in a group can be included in another group as long as the overlapping groups include noncommon 1s.

Ex:

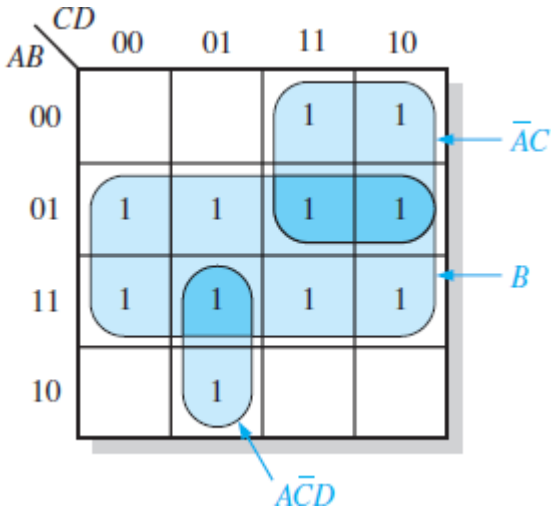


The groupings are



Determining the Minimum SOP Expression from the Map

Ex: Determine the product terms for the Karnaugh map and write the resulting minimum SOP expression



The resulting minimum SOP expression is the sum of these product terms:

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

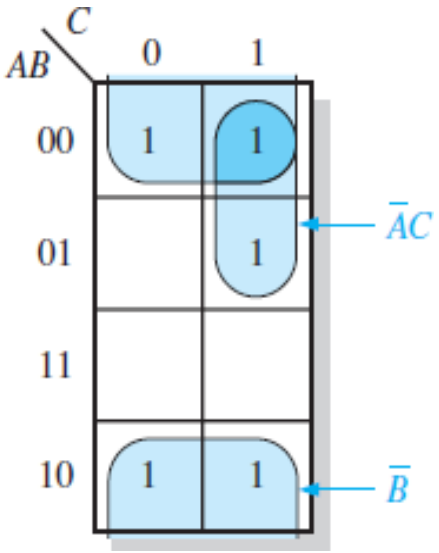
Ex: Use a Karnaugh map to minimize the following standard SOP expression:

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

The binary values of the expression are

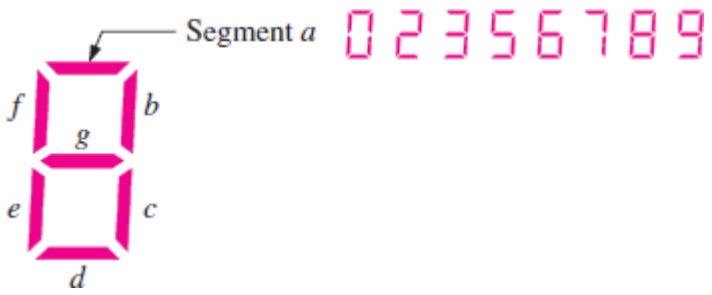
$$101 + 011 + 001 + 000 + 100$$

Map the standard SOP expression and group the cells



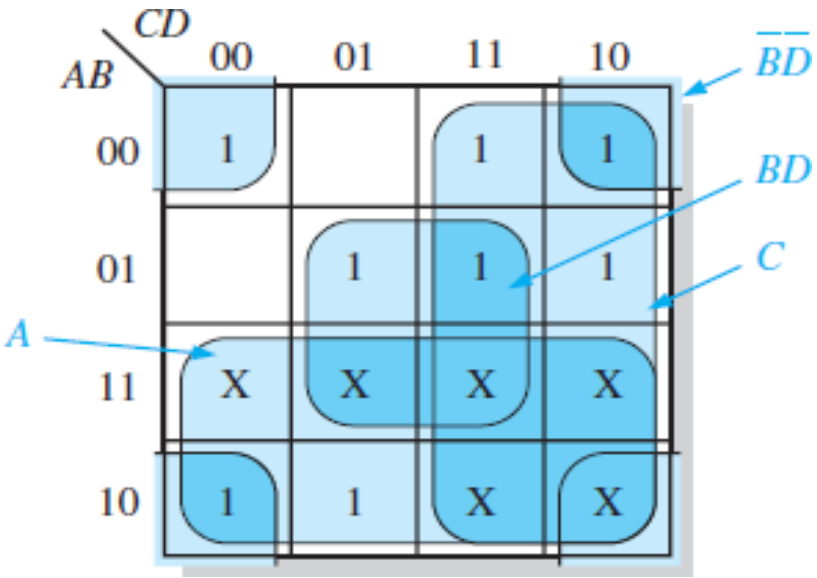
"Don't Care" Conditions

Ex: In a 7-segment display, each of the seven segments is activated for various digits. For example, segment **a** is activated for the digits 0, 2, 3, 5, 6, 7, 8, and 9. Since each digit can be represented by a BCD code, derive an SOP expression for segment a using the variables ABCD and then minimize the expression using a Karnaugh map.



The expression for segment **a** is

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



From the Karnaugh map, the minimized expression for segment **a** is

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

Mapping a Standard POS Expression

For a POS expression in standard form, a 0 is placed on the Karnaugh map for each sum term in the expression. Each 0 is placed in a cell corresponding to the value of a

sum term.

When a POS expression is completely mapped, there will be a number of 0s on the Karnaugh map equal to the number of sum terms in the standard POS expression. The cells that do not have a 0 are the cells for which the expression is 1.The following steps the mapping process:

Step 1: Determine the binary value of each sum term in the standard POS expression. This is the binary value that makes the term equal to 0.

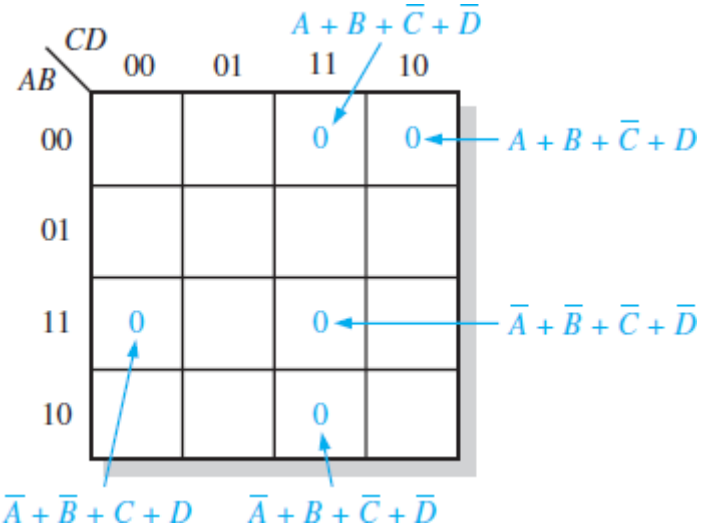
Step 2: As each sum term is evaluated, place a 0 on the Karnaugh map in the corresponding cell.

Ex: Map the following standard POS expression on a Karnaugh map:

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

Evaluate the expression as shown below and place a 0 on the 4-variable Karnaugh map for each standard sum term in the expression

$$\underset{1100}{(\overline{A} + \overline{B} + C + D)}\underset{1011}{(\overline{A} + B + \overline{C} + \overline{D})}\underset{0010}{(A + B + \overline{C} + D)}\underset{1111}{(\overline{A} + \overline{B} + \overline{C} + \overline{D})}\underset{0011}{(A + B + \overline{C} + \overline{D})}$$



Karnaugh Map Simplification of POS Expressions

The process for minimizing a POS expression is basically the same as for an SOP expression except that you group 0s to produce minimum sum terms instead of grouping 1s to produce minimum product terms. The rules for grouping the 0s are the same as those for grouping the 1s.

Ex: Use a Karnaugh map to minimize the following standard POS expression, and Also,

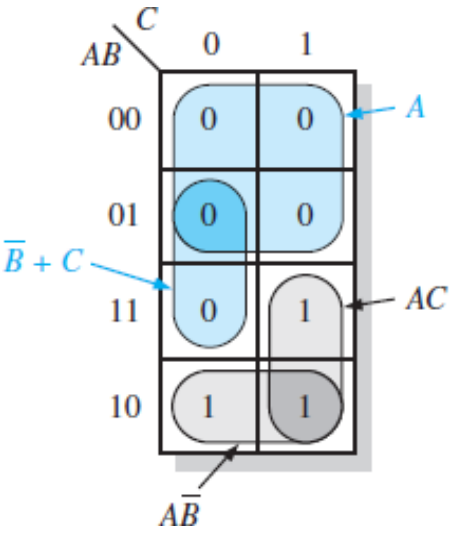
derive the equivalent SOP expression.

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

The combinations of binary values of the expression are

$$(0 + 0 + 0)(0 + 0 + 1)(0 + 1 + 0)(0 + 1 + 1)(1 + 1 + 0)$$

Map the standard POS expression and group the cells



Notice how the 0 in the 110 cell is included into a 2-cell group by utilizing the 0 in the 4-cell group. The sum term for each blue group is shown in the figure and the resulting minimum POS expression is

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

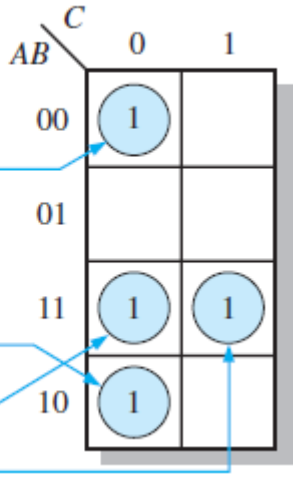
Grouping the 1s as shown by the gray areas yields an SOP expression that is equivalent to grouping the 0s.

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

Mapping Directly from a Truth Table

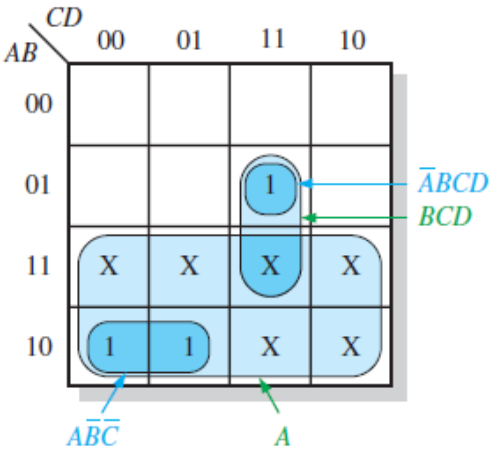
$X = \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + AB\bar{C} + ABC$

Inputs A	B	C	Output X
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1



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

Don't cares



(a) Truth table

(b) Without "don't cares" $Y = A\bar{B}\bar{C} + \bar{A}BCD$
With "don't cares" $Y = A + BCD$


First
Semester
Complete