

KONSEP DASAR ALJABAR BOOLEAN, GERBANG LOGIKA,

6

DAN HUKUM ALJABAR BOOLEAN

OBJEKTIF :

1. Mampu Memahami tentang konsep dasar aljabar boolean, gerbang logika, dan hukum aljabar boolean
2. Mahasiswa Mampu Menggunakan *Software* Netbeans dalam membuat program tentang aljabar boolean.

1. ALJABAR BOOLEAN

Definisi:

Misalkan terdapat

- Dua operator biner: $+$ dan \times
- Sebuah operator uner: $'$.
- B : himpunan yang didefinisikan pada operator $+$, \times , dan $'$
- 0 dan 1 adalah dua elemen yang berbeda dari B .

Tupel $(B, +, \times, ')$

disebut **aljabar Boolean** jika untuk setiap $a, b, c \in B$ berlaku aksioma-aksioma atau postulat Huntington berikut:

1. *Closure*/tertutup: (i) $a + b \in B$
(ii) $a \cdot b \in B$
2. Identitas: (i) $a + 0 = a$
(ii) $a \cdot 1 = a$
3. Komutatif: (i) $a + b = b + a$
(ii) $a \cdot b = b \cdot a$

4. Distributif: (i) $a \times (b + c) = (a \times b) + (a \times c)$

(ii) $a + (b \cdot c) = (a + b) \cdot (a + c)$

5. Komplemen¹: (i) $a + a' = 1$

(ii) $a \cdot a' = 0$

Untuk mempunyai sebuah aljabar Boolean, harus diperlihatkan:

1. Elemen-elemen himpunan B ,
2. Kaidah operasi untuk operator biner dan operator uner,
3. Memenuhi postulat Huntington.

Aljabar Boolean Dua-Nilai

Aljabar Boolean dua-nilai:

- $B = \{0, 1\}$
- operator biner, $+$ dan \times
- operator uner, $'$
- Kaidah untuk operator biner dan operator uner

a	b	$a \cdot b$
0	0	0
0	1	0
1	0	0
1	1	1

a	a'
0	1
1	0

Cek apakah memenuhi postulat Huntington:

1. *Closure* : jelas berlaku , karena hasil operasi + dan \times juga bernilai 0 atau 1.
2. Identitas: jelas berlaku karena dari tabel dapat kita lihat bahwa:
 - (i) $0 + 1 = 1 + 0 = 1$
 - (ii) $1 \times 0 = 0 \times 1 = 0$
3. Komutatif: jelas berlaku dengan melihat simetri tabel operator biner.
4. Distributif: (i) $a \times (b + c) = (a \times b) + (a \times c)$ dapat ditunjukkan benar dari tabel operator biner di atas dengan membentuk tabel kebenaran:

a	b	c	$b + c$	$a \times (b + c)$	$a \times b$	$a \times c$	$(a \times b) + (a \times c)$
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

(ii) Hukum distributif $a + (b \times c) = (a + b) \times (a + c)$ dapat ditunjukkan benar dengan membuat tabel kebenaran dengan cara yang sama seperti (i).

5. Komplemen: jelas berlaku karena Tabel 7.3 memperlihatkan bahwa:

(i) $a + a' = 1$, karena $0 + 0' = 0 + 1 = 1$ dan $1 + 1' = 1 + 0 = 1$

(ii) $a \times a = 0$, karena $0 \times 0' = 0 \times 1 = 0$ dan $1 \times 1' = 1 \times 0 = 0$

Karena kelima postulat Huntington dipenuhi, maka terbukti bahwa $B = \{0, 1\}$ bersama-sama dengan operator biner $+$ dan \times operator komplemen $'$ merupakan aljabar Boolean.

Ekspresi Boolean

- Misalkan $(B, +, \times, ')$ adalah sebuah aljabar Boolean. Suatu ekspresi Boolean dalam $(B, +, \times, ')$ adalah:
 - (i) setiap elemen di dalam B ,
 - (ii) setiap peubah,
 - (iii) jika e_1 dan e_2 adalah ekspresi Boolean, maka $e_1 + e_2$, $e_1 \times e_2$, e_1' adalah ekspresi Boolean

Contoh:

0

1

a

b

c

$a + b$

$a \times b$

$a' \times (b + c)$

$a \times b' + a \times b \times c' + b'$, dan sebagainya

Mengevaluasi Ekspresi Boolean

- Contoh: $a' \times (b + c)$

jika $a = 0$, $b = 1$, dan $c = 0$, maka hasil evaluasi ekspresi:

$$0' \times (1 + 0) = 1 \times 1 = 1$$

- Dua ekspresi Boolean dikatakan **ekivalen** (dilambangkan dengan '=') jika keduanya mempunyai nilai yang sama untuk setiap pemberian nilai-nilai kepada n peubah. Contoh:

$$a \times (b + c) = (a \cdot b) + (a \times c)$$

Contoh. Perhatikan bahwa $a + a'b = a + b$.

Penyelesaian:

a	b	a'	$a'b$	$a + a'b$	$a + b$
0	0	1	0	0	0
0	1	1	1	1	1
1	0	0	0	1	1
1	1	0	0	1	1

- Perjanjian: tanda titik (\times) dapat dihilangkan dari penulisan ekspresi Boolean, kecuali jika ada penekanan:

$$(i) \quad a(b + c) = ab + ac$$

$$(ii) \quad a + bc = (a + b)(a + c)$$

$$(iii) \quad a \times 0, \text{ bukan } a0$$

Prinsip Dualitas

Misalkan S adalah kesamaan (*identity*) di dalam aljabar Boolean yang melibatkan operator $+$, \times , dan komplemen, maka jika pernyataan S^* diperoleh dengan cara mengganti

\times dengan $+$

$+$ dengan \times

0 dengan 1

1 dengan 0

dan membiarkan operator komplemen tetap apa adanya, maka kesamaan S^* juga benar. S^* disebut sebagai *dual* dari S .

Contoh.

$$(i) (a \times 1)(0 + a') = 0 \text{ dualnya } (a + 0) + (1 \times a') = 1$$

$$(ii) a(a' + b) = ab \quad \text{dualnya } a + a'b = a + b$$

2. GERBANG LOGIKA

Pengertian GERBANG (GATE) :

- ♦ Rangkaian satu atau lebih sinyal masukan tetapi hanya menghasilkan satu sinyal keluaran.
- ♦ Rangkaian digital (dua keadaan), karena sinyal masukan atau keluaran hanya berupa tegangan tinggi atau low (1 atau 0).
- ♦ Setiap keluarannya tergantung sepenuhnya pada sinyal yang diberikan pada masukan-masukannya.

Gerbang logika atau gerbang logik adalah suatu entitas dalam elektronika dan matematika Boolean yang mengubah satu atau beberapa masukan logik menjadi sebuah sinyal keluaran logik. Gerbang logika terutama diimplementasikan secara elektronis menggunakan dioda atau transistor, akan tetapi dapat pula dibangun menggunakan

susunan komponen-komponen yang memanfaatkan sifat-sifat elektromagnetik (*relay*), cairan, optik dan bahkan mekanik.

Dalam logika dan bidang teknik yang memakainya, konjungsi, atau dan, adalah operator logika dalam kalkulus proposisional. Hasil dari dua proposisi juga disebut konjungsi mereka. Hasil konjungsi adalah benar jika *kedua* proposisinya benar; jika tidak, hasilnya adalah salah.

Dalam logika dan bidang teknik yang memakainya, disjungsi, atau atau, adalah operator logika dalam kalkulus proposisional. Hasil dari dua proposisi juga disebut disjungsi mereka. Hasil disjungsi adalah salah jika *kedua* proposisinya salah; jika tidak, hasilnya adalah benar.

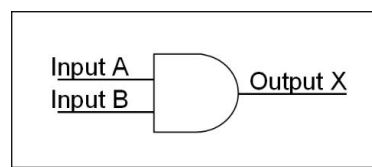
Dalam logika dan bidang teknik yang memakainya, negasi, atau tidak, adalah operator logika dalam kalkulus proposisional. Hasil dari dua proposisi juga disebut negasi mereka. Hasil negasi adalah benar jika proposisinya salah; jika tidak, hasilnya adalah salah.

RINGKASAN JENIS-JENIS GERBANG LOGIKA

Ada 7 gerbang logika dasar : AND, OR, NOT, NAND, NOR, Ex-OR, Ex-NOR.



Gerbang AND



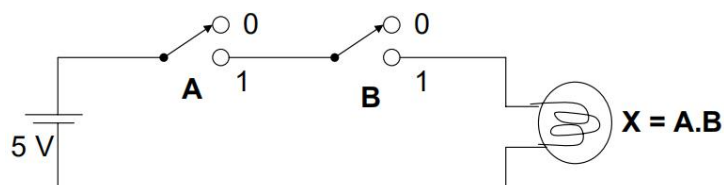
Operasi AND :

- Jika Input A AND B keduanya **HIGH**, maka output X akan **HIGH**
- Jika Input A atau B salah satu atau keduanya **LOW** maka output X akan **LOW**

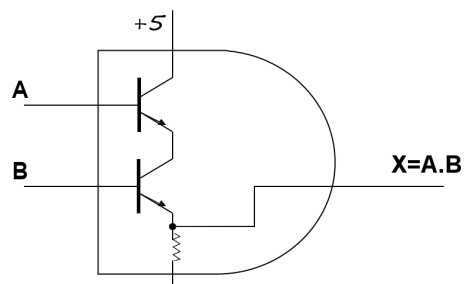
Tabel Kebenaran gerbang AND – 2 input

INPUT		Output
A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

Cara kerja Gerbang AND :

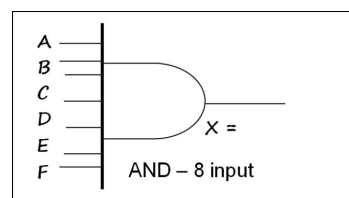
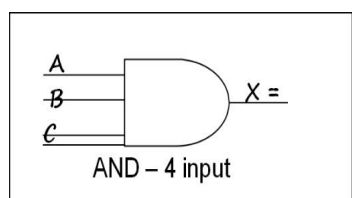


Analogi elektrik gerbang AND



Gerbang AND dengan switch Transistor

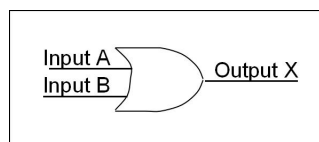
Gerbang AND dengan banyak Input



Tabel Kebenaran AND-4 input

INPUT				Output X
A	B	C	D	
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	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Gerbang OR



Simbol gerbang logika OR

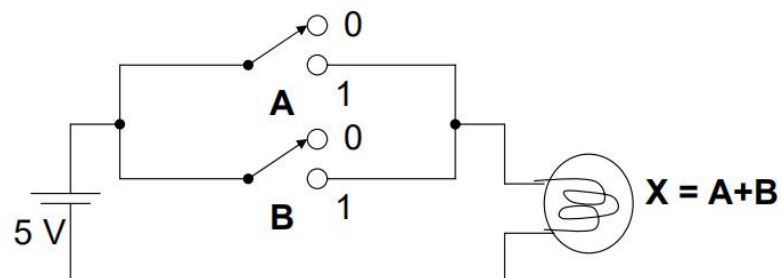
Operasi OR :

- Jika Input A OR B atau keduanya **HIGH**, maka output X akan **HIGH**
- Jika Input A dan B keduanya **LOW** maka output X akan **LOW**

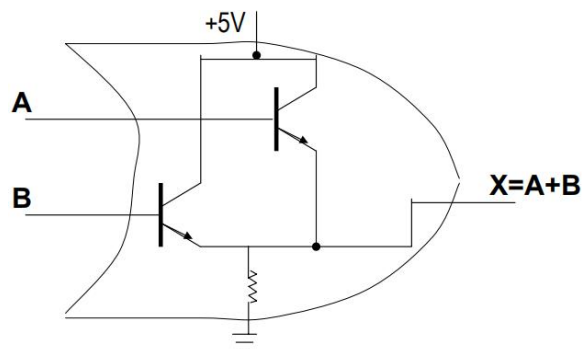
Tabel Kebenaran gerbang OR – 2 input

INPUT		Output
A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

Cara kerja Gerbang OR :

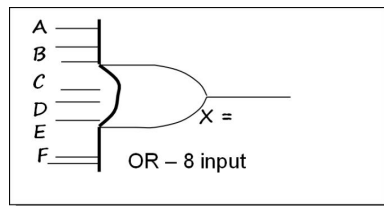
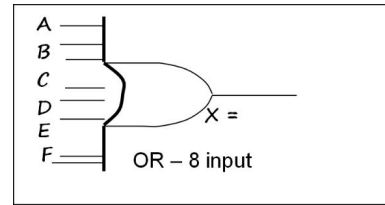
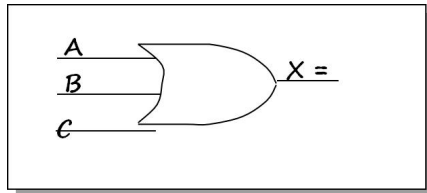


Analogi elektrikal gerbang OR



Gerbang OR dengan switch Transistor

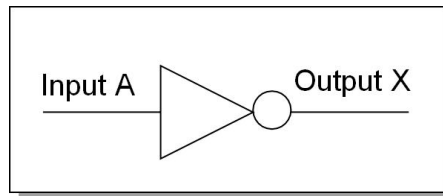
Gerbang OR dengan banyak Input



Tabel Kebenaran OR-3 input

INPUT			Output
A	B	C	X
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Gerbang NOT / INVERTER



Simbol gerbang logika NOT

Operasi NOT :

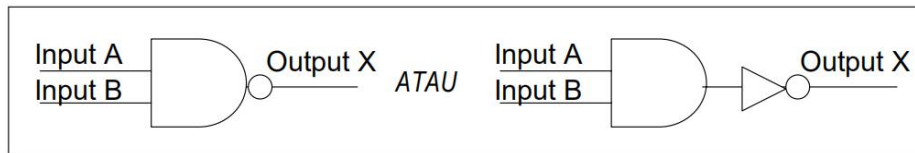
- Jika Input A **HIGH**, maka output X akan **LOW**
- Jika Input A **LOW**, maka output X akan **HIGH**

Tabel Kebenaran Gerbang Not / Inverter

INPUT A	Output X
0	1
1	0

$$X = \overline{A}$$

Gerbang NAND



Simbol gerbang logika NAND

Operasi NAND :

- Merupakan Inversi (kebalikan) dari operasi AND
- Jika Input A AND B keduanya **HIGH**, maka output X akan **LOW**
- Jika Input A atau B atau keduanya **LOW**, maka output X akan **HIGH**

Tabel Kebenaran gerbang NAND

INPUT		Output
A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

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

Gerbang NAND dengan banyak Input

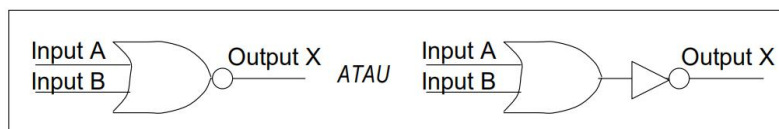


Tabel Kebenaran NAND-3 input

Tabel kebenaran 3 input

INPUT			Output
A	B	C	X
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Gerbang NOR



Simbol gerbang logika NOR

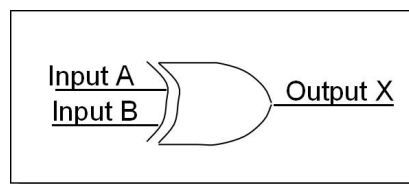
Operasi NOR :

- Merupakan Inversi (kebalikan) dari operasi OR
- Jika Input A dan B keduanya **LOW**, maka output X akan **HIGH**
- Jika Input A OR B salah satu atau keduanya **HIGH**, maka output X akan **LOW**

Tabel Kebenaran gerbang NOR

INPUT		Output
A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

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

Gerbang Ex-OR

Simbol gerbang logika Ex-OR

Operasi Ex-OR :

- Ex-OR adalah kependekan dari Exclusive OR
- Jika salah satu dari kedua inputnya HIGH (bukan kedua-duanya), maka output X akan HIGH
- Jika kedua inputnya bernilai LOW semua atau HIGH semua, maka output X akan LOW

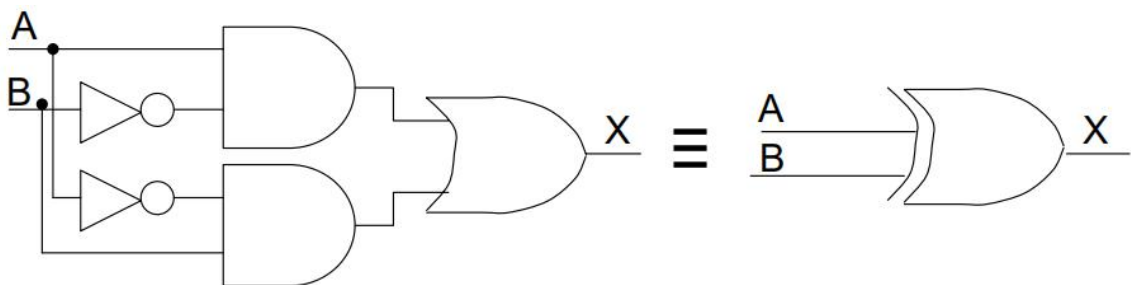
Tabel Kebenaran Gerbang Ex-OR

INPUT		OUTPUT
A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

$$X = A \oplus B$$

Berdasarkan Tabel Kebenaran di atas (yang bernilai output = 1), Ex-OR dapat disusun dari gerbang dasar : AND, OR dan NOT

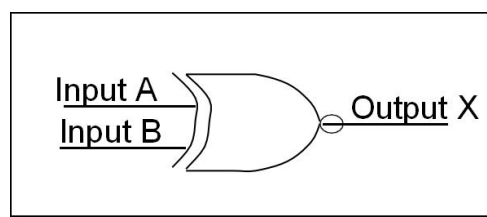
Persamaan EX-OR (dari AND, OR dan NOT) : $X = \overline{A}B + A\overline{B}$



Gerbang Ex-OR dari AND, OR, NOT

Simbol logika Ex-OR

Gerbang Ex-NOR



Simbol gerbang logika Ex-NOR

Operasi Ex-NOR :

- Ex-NOR merupakan kebalikan dari Ex-OR
- Jika salah satu dari kedua inputnya HIGH (bukan kedua-duanya), maka output X akan LOW
- Jika kedua inputnya bernilai LOW semua atau HIGH semua, maka output X akan HIGH

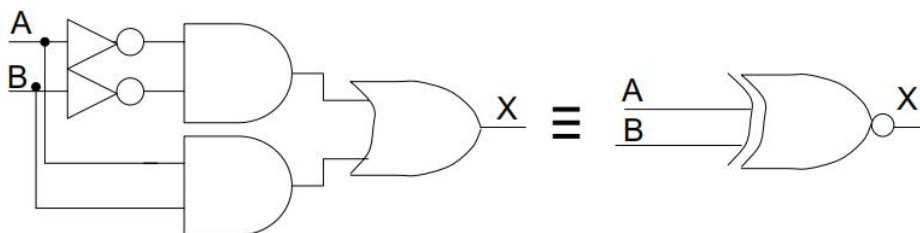
Tabel Kebenaran Gerbang Ex-NOR

INPUT		OUTPUT
A	B	X
0	0	1
0	1	0
1	0	0
1	1	1

$$X = \overline{A \oplus B}$$

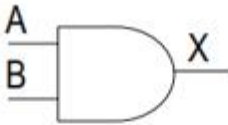
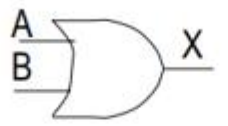
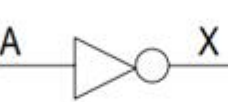
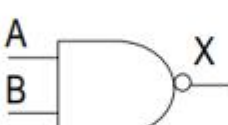



Berdasarkan Tabel Kebenaran di atas (yang bernilai output = 1), Ex-NOR dapat disusun dari gerbang dasar : AND, OR dan NOT Persamaan EX-NOR (dari AND, OR dan NOT) :

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



Gerbang Ex-NOR dari AND, OR, NOT

Simbol logika Ex-NOR

No	NAMA	TIPE IC	Simbol Logika	Persamaan	Tabel Kebenaran																		
1	AND	7408		$X=A.B$	<table><tr><th colspan="2">INPUT</th><th>Output</th></tr><tr><th>A</th><th>B</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	INPUT		Output	A	B	X	0	0	0	0	1	0	1	0	0	1	1	1
INPUT		Output																					
A	B	X																					
0	0	0																					
0	1	0																					
1	0	0																					
1	1	1																					
2	OR	7432		$X=A+B$	<table><tr><th colspan="2">INPUT</th><th>Output</th></tr><tr><th>A</th><th>B</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	INPUT		Output	A	B	X	0	0	0	0	1	1	1	0	1	1	1	1
INPUT		Output																					
A	B	X																					
0	0	0																					
0	1	1																					
1	0	1																					
1	1	1																					
3	NOT	7404		$X=\overline{A}$	<table><tr><th>INPUT</th><th>Output</th></tr><tr><th>A</th><th>X</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	INPUT	Output	A	X	0	1	1	0										
INPUT	Output																						
A	X																						
0	1																						
1	0																						
4	NAND	7400		$X=\overline{A.B}$	<table><tr><th colspan="2">INPUT</th><th>Output</th></tr><tr><th>A</th><th>B</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	INPUT		Output	A	B	X	0	0	1	0	1	1	1	0	1	1	1	0
INPUT		Output																					
A	B	X																					
0	0	1																					
0	1	1																					
1	0	1																					
1	1	0																					
5	NOR	7402		$X=\overline{A+B}$	<table><tr><th colspan="2">INPUT</th><th>Output</th></tr><tr><th>A</th><th>B</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	INPUT		Output	A	B	X	0	0	1	0	1	0	1	0	0	1	1	0
INPUT		Output																					
A	B	X																					
0	0	1																					
0	1	0																					
1	0	0																					
1	1	0																					
6	Ex-OR	7486		$X=A\oplus B$	<table><tr><th colspan="2">INPUT</th><th>OUTPUT</th></tr><tr><th>A</th><th>B</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	INPUT		OUTPUT	A	B	X	0	0	0	0	1	1	1	0	1	1	1	0
INPUT		OUTPUT																					
A	B	X																					
0	0	0																					
0	1	1																					
1	0	1																					
1	1	0																					
7	Ex-NOR			$X=\overline{A\oplus B}$	<table><tr><th colspan="2">INPUT</th><th>OUTPUT</th></tr><tr><th>A</th><th>B</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	INPUT		OUTPUT	A	B	X	0	0	1	0	1	0	1	0	0	1	1	1
INPUT		OUTPUT																					
A	B	X																					
0	0	1																					
0	1	0																					
1	0	0																					
1	1	1																					

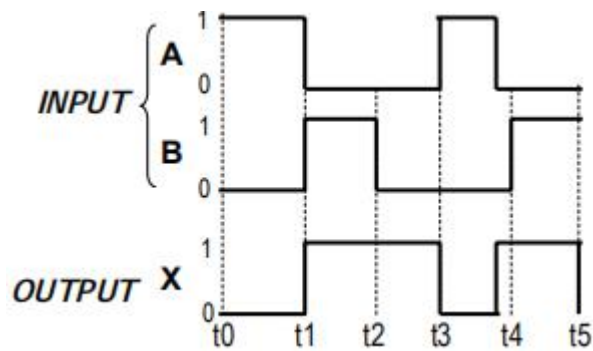
ANALISA PE-WAKTU-AN

Cara penganalisaan response output terhadap kombinasi input-inputnya pada periode waktu tertentu,

Cara penganalisaan yang lain adalah dengan Tabel Kebenaran

Peralatan yang digunakan disebut : *Timing Diagram* (Diagram pe-waktu-an).

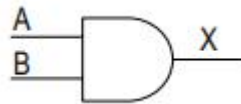
Bentuk Timing Diagram :



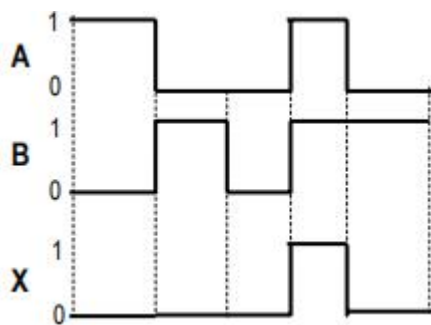
Contoh :

1. Buatlah timing diagram untuk mendapatkan output dari gerbang AND

berikut ini :



Jawab :

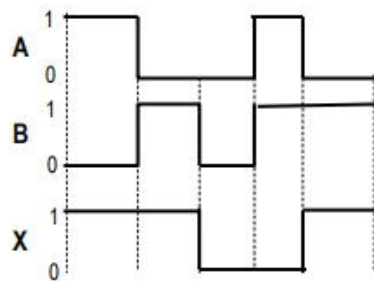


2. Buatlah timing diagram untuk mendapatkan output dari gerbang Ex-OR

berikut ini :



Jawab :



3. HUKUM ALJABAR BOOLEAN

1. Hukum identitas: (i) $a + 0 = a$ (ii) $a \times 1 = a$	2. Hukum idempoten: (i) $a + a = a$ (ii) $a \times a = a$
3. Hukum komplemen: (i) $a + a' = 1$ (ii) $aa' = 0$	4. Hukum dominansi: (i) $a \times 0 = 0$ (ii) $a + 1 = 1$
5. Hukum involusi: (i) $(a')' = a$	6. Hukum penyerapan/absorpsi: (i) $a + ab = a$ (ii) $a(a + b) = a$
7. Hukum komutatif: (i) $a + b = b + a$ (ii) $ab = ba$	8. Hukum asosiatif: (i) $a + (b + c) = (a + b) + c$ (ii) $a (b c) = (a b) c$
9. Hukum distributif: (i) $a + (b c) = (a + b) (a + c)$ (ii) $a (b + c) = a b + a c$	10. Hukum De Morgan: (i) $(a + b)' = a' b'$ (ii) $(ab)' = a' + b'$

11. Hukum 0/1 (i) $0' = 1$ (ii) $1' = 0$	12. (i) $a + a'b = a + b$ (ii) $a(a' + b) = ab$
---	---

Contoh 7.3. Buktikan (i) $a + a'b = a + b$ dan (ii) $a(a' + b) = ab$

Penyelesaian:

$$\begin{aligned}
 \text{(i)} \quad a + a'b &= (a + ab) + a'b \text{ (Penyerapan)} \\
 &= a + (ab + a'b) \quad \text{(Asosiatif)} \\
 &= a + (a + a')b \text{ (Distributif)} \\
 &= a + 1b \quad \text{(Komplemen)} \\
 &= a + b \quad \text{(Identitas)}
 \end{aligned}$$

(ii) adalah dual dari (i).

Aljabar Boolean sebagai Lattice

Suatu aljabar boolean adalah *lattice complemented*, yaitu lattice yang terbatas dan semua elemennya mempunyai komplemen.

Fungsi Boolean

- **Fungsi Boolean** (disebut juga fungsi biner) adalah pemetaan dari B^n ke B melalui ekspresi Boolean, kita menuliskannya sebagai

$$f: B^n \rightarrow B$$

- yang dalam hal ini B^n adalah himpunan yang beranggotakan pasangan terurut ganda- n (*ordered n -tuple*) di dalam daerah asal B .
- Setiap ekspresi Boolean tidak lain merupakan fungsi Boolean.
- Misalkan sebuah fungsi Boolean adalah

$$f(x, y, z) = xyz + x'y + y'z$$

Fungsi f memetakan nilai-nilai pasangan

terurut ganda-3 (x, y, z) ke himpunan $\{0, 1\}$.

Contohnya, $(1, 0, 1)$ yang berarti $x = 1$, $y = 0$, dan $z = 1$

sehingga $f(1, 0, 1) = 1 \times 0 \times 1 + 1' \times 0 + 0' \times 1 = 0 + 0 + 1 = 1$

Contoh.

Contoh-contoh fungsi Boolean yang lain:

1. $f(x) = x$

2. $f(x, y) = x'y + xy' + y'$

3. $f(x, y) = x' y'$

4. $f(x, y) = (x + y)'$

5. $f(x, y, z) = xyz'$

- Setiap peubah di dalam fungsi Boolean, termasuk dalam bentuk komplemennya, disebut literal.

Contoh: Fungsi $h(x, y, z) = xyz'$ pada contoh di atas terdiri dari 3 buah literal, yaitu x , y , dan z' .

Contoh. Diketahui fungsi Boolean $f(x, y, z) = xy z'$, nyatakan h dalam tabel kebenaran.

Penyelesaian :

x	y	z	$f(x, y, z) = xy z'$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Penyederhanaan Fungsi Boolean

Contoh.

$f(x, y) = x'y + xy' + y'$ disederhanakan menjadi $f(x, y) = x' + y'$

Penyederhanaan fungsi Boolean dapat dilakukan dengan cara:

1. Secara aljabar
2. Menggunakan Peta Karnaugh

Penyederhanaan Secara Aljabar

Contoh:

$$\begin{aligned}
 1. \quad f(x, y) &= x + x'y \\
 &= (x + x')(x + y) \\
 &= 1 \times (x + y) \\
 &= x + y
 \end{aligned}$$

$$2. f(x, y, z) = x'y'z + x'yz + xy'$$

$$= x'z(y' + y) + xy'$$

$$= x'z + xz'$$

$$3. f(x, y, z) = xy + x'z + yz = xy + x'z + yz(x + x')$$

$$= xy + x'z + xyz + x'yz$$

$$= xy(1 + z) + x'z(1 + y) = xy + x'z$$

Peta Karnaugh

a. Peta Karnaugh dengan dua perubah y

		y	
		0	1
x	0	$x'y'$	$x'y$
	1	xy'	xy

b. Peta dengan tiga peubah

				yz			
				00	01	11	10
x	0	$x'y'z'$	$x'y'z$	$x'yz$	$x'yz'$		
	1	$xy'z'$	$xy'z$	xyz	xyz'		

Contoh. Diberikan tabel kebenaran, gambarkan Peta Karnaugh.

x	y	z	$f(x, y, z)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

		yz			
		00	01	11	10
x	0	0	0	0	1
	1	0	0	1	1

c. Peta dengan empat peubah

				yz				
				00	01	11	10	
m_0	m_1	m_3	m_2	wx 00	$w'x'y'z'$	$w'x'y'z$	$w'x'yz$	$w'x'yz'$
m_4	m_5	m_7	m_6	01	$w'xy'z'$	$w'xy'z$	$w'xyz$	$w'xyz'$
m_{12}	m_{13}	m_{15}	m_{14}	11	$wxy'z'$	$wxy'z$	$wxyz$	$wxyz'$
m_8	m_9	m_{11}	m_{10}	10	$wx'y'z'$	$wx'y'z$	$wx'yz$	$wx'yz'$

Contoh. Diberikan tabel kebenaran, gambarkan Peta Karnaugh.

w	x	y	z	$f(w, x, y, z)$
0	0	0	0	0
0	0	0	1	1
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	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

		yz			
		00	01	11	10
wx	00	0	1	0	1
	01	0	0	1	1
	11	0	0	0	1
	10	0	0	0	0

CONTOH PROGRAM JAVA

```
// Mendemonstrasikan operator-operator logika boolean.

public class LogikaBool {

    public static void main(String args[]) {

        boolean a = true;

        boolean b = false;

        boolean c = a | b;

        boolean d = a & b;

        boolean e = a ^ b;

        boolean f = (!a & b) | (a & !b);

        boolean g = !a;

        System.out.println(" a = " + a);

        System.out.println(" b = " + b);

        System.out.println(" a|b = " + c);

        System.out.println(" a&b = " + d);

        System.out.println(" a^b = " + e);

        System.out.println(" !a&b|a&!b = " + f);

        System.out.println(" !a = " + g);

    }

}
```

OUTPUT

```
a = true
b = false
a|b = true
a&b = false
a^b = true
!a&b|a&!b = true
!a = false
```