



Aljabar Boolean dan Gerbang Logika

Q Fadlan, S.ST, M.Kom

Kus Andriadi, S.Kom., M.T.I

Formal Education :

- **S1 Teknologi Informasi** - Universitas Budi Luhur
- **S2 Magister Teknologi Informasi**- Universitas Indonesia
- **S3 Doktor Ilmu Komputer** - BINUS (on going, exp. 2025)

Currently work in one of e-commerce in Indonesia as **Sr. Engineering Manager/Lead Principal SDE.**

WA : 0856-9250-9990

Email : kus@stmik.tazkia.ac.id

AGENDA

1. Pengenalan Aljabar Boolean
2. Dasar Aljabar Boolean
3. Gerbang Logika Dasar
4. Implementasi Gerbang Logika pada Komputer
5. Praktikum Simulator Gerbang Logika



PENDAHULUAN

STMIK
TAZKIA

CAPAIAN PEMBELAJARAN

1. Memahami konsep dasar Aljabar Boolean dan operasi-operasinya dalam sistem digital
2. Menganalisis dan merancang rangkaian logika menggunakan gerbang-gerbang dasar
3. Menerapkan teknik penyederhanaan rangkaian logika untuk optimasi sistem
4. Mengimplementasikan rangkaian logika menggunakan simulator
5. Mengaitkan konsep gerbang logika dengan komponen-komponen dalam arsitektur komputer

KOMPETENSI YANG DIHARAPKAN

1. Memahami dan menerapkan konsep dasar Aljabar Boolean serta hukum-hukumnya dalam perancangan rangkaian logika digital
2. Menganalisis dan merancang rangkaian kombinasional dan sekuensial menggunakan berbagai jenis gerbang logika dasar
3. Mengoperasikan simulator gerbang logika untuk membuat dan menguji rangkaian digital seperti adder, decoder, dan flip-flop
4. Mengembangkan kemampuan berpikir logis dan sistematis dalam optimasi rangkaian logika untuk efisiensi sistem
5. Mengidentifikasi dan menjelaskan implementasi rangkaian logika dalam komponen-komponen dasar arsitektur komputer

MANFAAT DALAM ARSITEKTUR KOMPUTER

1. Memberikan pemahaman fundamental tentang cara kerja komputer di level hardware, termasuk proses pengolahan data biner dan operasi logika dasar
2. Mengembangkan kemampuan dalam perancangan dan optimasi sistem digital yang menjadi dasar dari komponen-komponen komputer modern
3. Membangun fondasi kuat untuk memahami arsitektur processor, memory, dan ALU (Arithmetic Logic Unit) dalam sistem komputer
4. Meningkatkan kemampuan analisis dan troubleshooting hardware yang diperlukan dalam pengembangan dan pemeliharaan sistem komputer
5. Menyediakan dasar pengetahuan yang esensial untuk pengembangan karir di bidang hardware development dan sistem terintegrasi

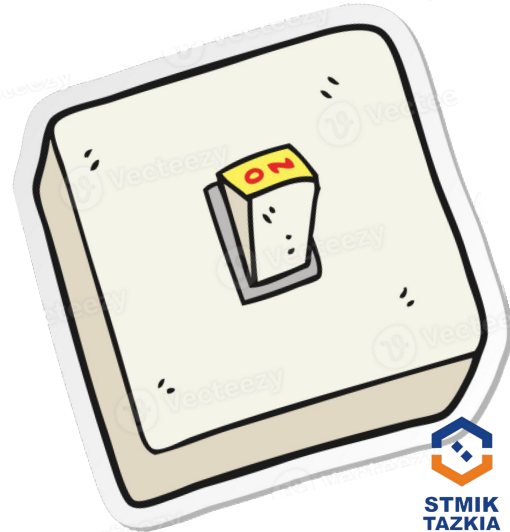
DEFINISI ALJABAR BOOLEAN

Aljabar Boolean adalah cabang matematika yang khusus dirancang untuk menangani operasi logika dengan menggunakan dua nilai (biner). Sistem ini menggunakan tiga operasi dasar yaitu AND (dan), OR (atau), dan NOT (tidak) untuk mengolah nilai-nilai biner. Aljabar Boolean menjadi pondasi penting dalam perancangan sistem digital termasuk rangkaian komputer, karena komputer pada dasarnya bekerja dengan mengolah nilai 0 dan 1.

DEFINISI ALJABAR BOOLEAN

Contoh penggunaan sederhana:

- Jika tombol A DAN tombol B ditekan (operasi AND), lampu akan menyala
- Jika tombol A ATAU tombol B ditekan (operasi OR), bel akan berbunyi
- Jika saklar dalam kondisi TIDAK menyala (operasi NOT), kipas akan berputar



SEJARAH ALJABAR BOOLEAN



SEJARAH ALJABAR BOOLEAN

1. Era Awal (1847-1854)

- George Boole (1815-1864), seorang matematikawan Inggris, memperkenalkan Aljabar Boolean pertama kali dalam bukunya "The Mathematical Analysis of Logic" (1847)
- Boole kemudian menyempurnakan idenya dalam buku "An Investigation of the Laws of Thought" (1854)
- Ia mengembangkan sistem matematika untuk merepresentasikan dan memanipulasi logika dengan menggunakan simbol dan persamaan

SEJARAH ALJABAR BOOLEAN

2. Pengembangan Awal (1930-an)

- Claude Shannon (1916-2001), dalam tesisnya di MIT tahun 1937 berjudul "A Symbolic Analysis of Relay and Switching Circuits"
- Shannon menunjukkan bahwa Aljabar Boolean dapat digunakan untuk:
 - Merancang rangkaian switching elektrik
 - Mengoptimalkan rangkaian relay telepon
 - Menjadi dasar rangkaian digital modern

SEJARAH ALJABAR BOOLEAN

3. Era Komputer Digital (1940-1950)

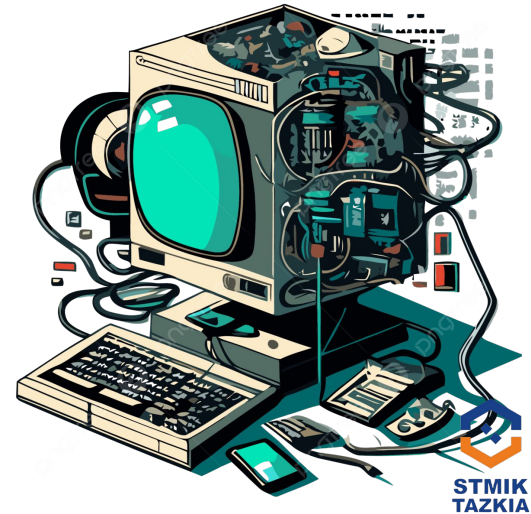
- John von Neumann menggunakan prinsip Aljabar Boolean dalam perancangan komputer EDVAC
- Pengembangan komputer ENIAC dan UNIVAC menggunakan konsep rangkaian logika berbasis Aljabar Boolean
- Mulai digunakan secara luas dalam perancangan rangkaian digital

SEJARAH ALJABAR BOOLEAN

4. Perkembangan Modern (1950-sekarang)

- Penemuan transistor (1947) dan rangkaian terintegrasi/IC (1958) membuat implementasi Aljabar Boolean menjadi lebih efisien
- Pengembangan gerbang logika IC memungkinkan miniaturisasi rangkaian digital
- Menjadi dasar perkembangan mikroprosesor dan komputer modern

Pemahaman sejarah ini menunjukkan bagaimana ide matematika abstrak yang dikembangkan oleh George Boole berevolusi menjadi fondasi teknologi modern yang kita gunakan sehari-hari. Perkembangan ini terus berlanjut dengan munculnya teknologi-teknologi baru yang tetap mengandalkan prinsip-prinsip dasar Aljabar Boolean.



PENTINGNYA DALAM SISTEM DIGITAL

Representasi Informasi:

- **Biner:** Semua data dalam komputer, dari teks hingga gambar, pada dasarnya direpresentasikan sebagai kombinasi dari 0 dan 1. Aljabar Boolean memungkinkan kita untuk memanipulasi representasi biner ini.
- **Logika:** Aljabar Boolean menyediakan cara untuk merepresentasikan logika, seperti kondisi benar (true) atau salah (false). Ini sangat penting dalam pengambilan keputusan dalam program.

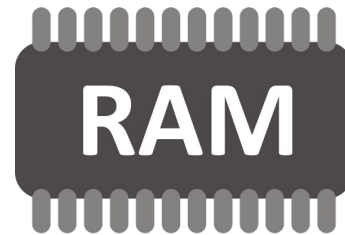
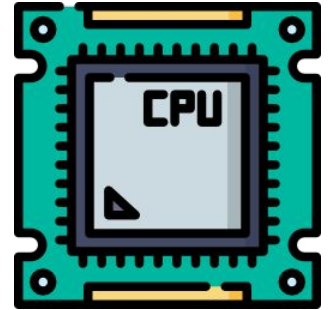
Perancangan Rangkaian Digital:

- Gerbang Logika: Aljabar Boolean digunakan untuk merancang gerbang logika seperti AND, OR, NOT, yang merupakan blok bangunan dasar dari semua sirkuit digital.
- Sirkuit Kombinasional: Sirkuit yang outputnya hanya bergantung pada input saat ini, seperti adder, multiplexer, dan decoder, dirancang menggunakan aljabar Boolean.
- Sirkuit Sekuensial: Sirkuit yang outputnya bergantung pada input saat ini dan keadaan sebelumnya, seperti flip-flop dan counter, juga menggunakan aljabar Boolean dalam desainnya.

APLIKASI DALAM KOMPUTER MODERN

Rangkaian Logika:

- Gerbang Logika: Aljabar Boolean digunakan untuk merancang gerbang logika seperti AND, OR, NOT, yang merupakan blok bangunan dasar dari semua sirkuit digital.
- Processor: Setiap operasi yang dilakukan oleh processor, dari perhitungan aritmatika hingga pengambilan keputusan, didasarkan pada operasi Boolean.
- Memori: Cara data disimpan dan diambil dari memori juga melibatkan operasi Boolean.



APLIKASI DALAM KOMPUTER MODERN

Bahasa Pemrograman:

- Tipe Data Boolean: Hampir semua bahasa pemrograman memiliki tipe data Boolean yang hanya memiliki dua nilai: benar (true) atau salah (false).
- Operator Logika: Operator seperti AND, OR, NOT digunakan untuk membuat ekspresi logika dalam program.
- Struktur Kontrol: Struktur seperti if-else, while, dan for menggunakan ekspresi Boolean untuk mengontrol aliran eksekusi program.



APLIKASI DALAM KOMPUTER MODERN

Python

```
umur = 20
memiliki_tiket = True

if umur >= 18 and memiliki_tiket:
    print("Anda boleh masuk")
```



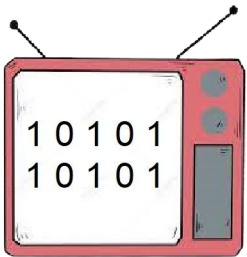


DASAR ALJABAR BOOLEAN

**STMIK
TAZKIA**

VARIABEL DAN OPERATOR BOOLEAN

Variabel Boolean adalah sebuah variabel yang hanya dapat menyimpan dua nilai, yaitu benar (true) atau salah (false). Dalam konteks komputer, nilai ini seringkali diwakilkan dengan angka biner 1 (untuk true) dan 0 (untuk false).

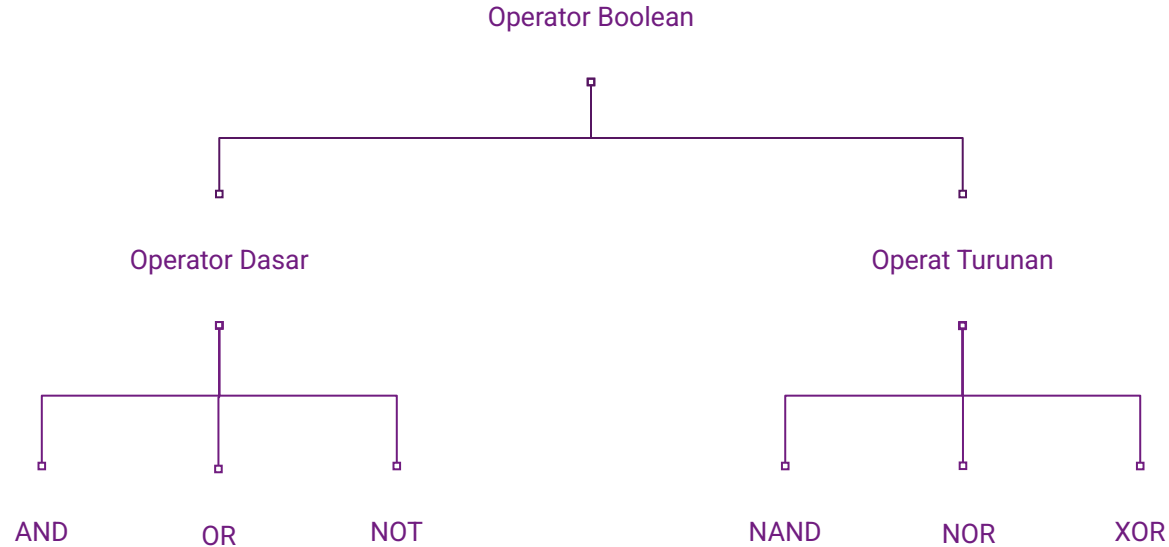


VARIABEL DAN OPERATOR BOOLEAN

Mengapa penting dalam komputer?

- **Pengambilan keputusan:** Variabel Boolean digunakan untuk membuat keputusan dalam program. Misalnya, jika suatu kondisi bernilai true, maka program akan menjalankan satu set instruksi, sedangkan jika false, maka program akan menjalankan instruksi yang lain.
- **Representasi data:** Variabel Boolean juga digunakan untuk merepresentasikan data yang bersifat biner, seperti status suatu perangkat (aktif atau tidak aktif), hasil perbandingan (sama atau tidak sama), dan sebagainya.
- **Implementasi logika:** Variabel Boolean adalah dasar dari logika digital, yang merupakan fondasi dari semua operasi komputer.

VARIABEL DAN OPERATOR BOOLEAN



VARIABEL DAN OPERATOR BOOLEAN

Simbol Operator

1. AND = • atau &&
2. OR = + atau ||
3. NOT = ' atau !
4. NAND = NAND
5. NOR = NOR
6. XOR = XOR



VARIABEL DAN OPERATOR BOOLEAN

OPERATOR AND

Output 1 hanya jika semua input bernilai 1 atau akan menghasilkan **true** jika kedua kondisi bernilai **true**

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

VARIABEL DAN OPERATOR BOOLEAN

OPERATOR OR

Output 1 jika salah satu input bernilai 1 atau akan menghasilkan **true** jika salah satu atau kedua kondisi bernilai **true**

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

VARIABEL DAN OPERATOR BOOLEAN

OPERATOR NOT

Membalikkan nilai input

A	NOT A
1	0
0	1

VARIABEL DAN OPERATOR BOOLEAN

OPERATOR NAND

Kombinasi AND diikuti NOT atau NAND menghasilkan **true** jika setidaknya salah satu input bernilai **false**

A	B	A NAND B
0	0	1
0	1	1
1	0	1
1	1	0

VARIABEL DAN OPERATOR BOOLEAN

OPERATOR NOR

Kombinasi OR diikuti NOT atau NOR menghasilkan **true** hanya jika kedua input bernilai **false**

A	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0

VARIABEL DAN OPERATOR BOOLEAN

OPERATOR XOR

Output 1 jika input berbeda atau XOR menghasilkan **true** hanya jika salah satu input bernilai **true** dan yang lainnya bernilai **false**

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

HUKUM DAN TEOREMA BOOLEAN

01	HUKUM KOMUTATIF	<ul style="list-style-type: none">• $A \text{ OR } B = B \text{ OR } A$ (Komutatif OR)• $A \text{ AND } B = B \text{ AND } A$ (Komutatif AND)• Urutan operasi tidak mempengaruhi hasil
02	HUKUM ASOSIATIF	<ul style="list-style-type: none">• $(A \text{ OR } B) \text{ OR } C = A \text{ OR } (B \text{ OR } C)$ (Asosiatif OR)• $(A \text{ AND } B) \text{ AND } C = A \text{ AND } (B \text{ AND } C)$ (Asosiatif AND)• Pengelompokan operasi tidak mempengaruhi hasil
03	HUKUM DISTRIBUTIF	<ul style="list-style-type: none">• $A \text{ AND } (B \text{ OR } C) = (A \text{ AND } B) \text{ OR } (A \text{ AND } C)$ (Distributif AND atas OR)• $A \text{ OR } (B \text{ AND } C) = (A \text{ OR } B) \text{ AND } (A \text{ OR } C)$ (Distributif OR atas AND)• Operasi perkalian dapat didistribusikan ke penjumlahan dan sebaliknya
04	DE MORGAN'S LAW	<ul style="list-style-type: none">• $\text{NOT}(A \text{ OR } B) = \text{NOT}(A) \text{ AND } \text{NOT}(B)$• $\text{NOT}(A \text{ AND } B) = \text{NOT}(A) \text{ OR } \text{NOT}(B)$• Komplemen dari penjumlahan = perkalian dari komplemen• Komplemen dari perkalian = penjumlahan dari komplemen

HUKUM DAN TEOREMA BOOLEAN

05

HUKUM IDENTITAS

- $A \text{ AND } 1 = A$
- $A \text{ OR } 0 = A$

06

HUKUM KOMPLEMEN

- $A \text{ AND NOT } A = 0$
- $A \text{ OR NOT } A = 1$
- $\text{NOT}(\text{NOT } A) = A$

07

HUKUM NOL DAN SATU

- $A \text{ OR } 1 = 1$
- $A \text{ AND } 0 = 0$
- $A \text{ OR } A = A$
- $A \text{ AND } A = A$

PENYEDERHANAAN EKSPRESI BOOLEAN

Penyederhanaan ekspresi Boolean adalah proses mengubah suatu ekspresi Boolean yang kompleks menjadi bentuk yang lebih sederhana namun tetap setara secara logika. Ekspresi Boolean sendiri adalah pernyataan yang terdiri dari variabel-variabel Boolean (bernilai benar atau salah) yang dihubungkan oleh operator-operator Boolean seperti AND, OR, NOT, NAND, NOR, dan XOR.

PENYEDERHANAAN EKSPRESI BOOLEAN

TUJUAN PENYEDERHANAAN:

1. ***Meningkatkan efisiensi rangkaian digital:*** Mengurangi jumlah gerbang logika yang dibutuhkan, sehingga mengurangi biaya dan konsumsi daya.
2. ***Mempermudah analisis:*** Ekspresi yang sederhana lebih mudah dipahami dan dianalisis.
3. ***Memperbaiki keandalan:*** Rangkaian yang lebih sederhana cenderung lebih andal karena memiliki lebih sedikit komponen.

PENYEDERHANAAN EKSPRESI BOOLEAN

CONTOH PENYEDERHANAAN

$$F = AB' + AB + A'B = (A \text{ AND NOT } B) \text{ OR } (A \text{ AND } B) \text{ OR } (\text{ NOT } A \text{ AND } B)$$

Langkah 1: Gunakan Hukum Distributif

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

Langkah 2: Gunakan Hukum Komplemen ($B' + B = 1$)

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

PENYEDERHANAAN EKSPRESI BOOLEAN

Langkah 3: Gunakan Hukum Distributif

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

Langkah 4: Gunakan Hukum Komplemen ($A + A' = 1$)

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

PENYEDERHANAAN EKSPRESI BOOLEAN

Bentuk Sederhana dari

$F = AB' + AB + A'B$ adalah $F = A + B$

Pembuktian ($A = 1$ dan $B = 0$)

- $F = AB' + AB + A'B = 1.1 + 1.0 + 0.0 = 1 + 0 + 0 = 1$
- $F = A + B = 1 + 0 = 1$

LATIHAN SOAL

Sederhanakan Ekspresi Boolean Berikut ini:

1. $F = A'B'C + AB'C + ABC + ABC'$
2. $H = (A + B)(A + C)(B + C') + ABC$
3. $J = A'BC + ABC + AB'C + ABC' + A'BC'$
4. $Q = (A + B)(A' + C)(B + C)(A + B' + C')$
5. $X = AB'C + A'BC + ABC + (A + B + C)(A' + B' + C')$



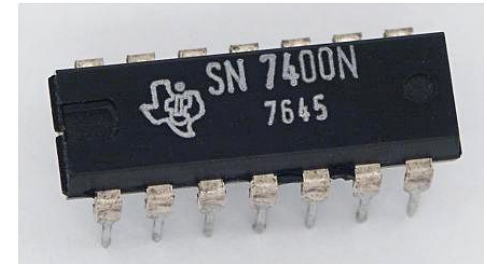
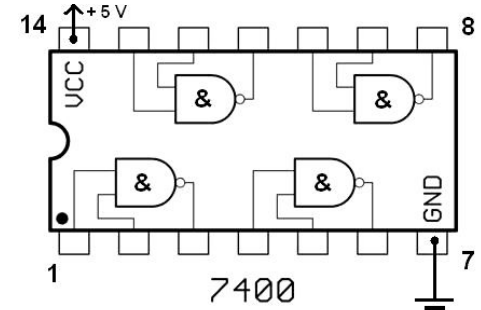
GERBANG LOGIKA DASAR

STMIK
TAZKIA

PENGENALAN GERBANG LOGIKA

Pengertian Dasar Gerbang Logika

- Gerbang logika adalah blok dasar untuk membangun rangkaian digital
- Bekerja dengan nilai biner (0 dan 1)
- Mengolah input menjadi output berdasarkan fungsi logika tertentu
- Seperti "saklar pintar" yang menghasilkan output sesuai aturan tertentu



Gerbang logika NAND TTL tipe 7400 yang

OPERATOR BOOLEAN VS GERBANG LOGIKA

Operator Boolean



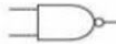

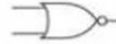


- **Konsep:** Operator Boolean adalah simbol-simbol yang digunakan dalam pemrograman untuk melakukan operasi logika pada nilai Boolean (true atau false).
- **Fungsi:** Operator ini digunakan untuk membandingkan nilai, membuat keputusan, dan mengontrol aliran program.
- **Contoh:** AND (&&), OR (||), NOT (!).
- **Tingkat Abstraksi:** Berada pada tingkat abstraksi yang lebih tinggi, digunakan dalam bahasa pemrograman untuk menulis kode.

Gerbang Logika

- **Konsep:** Gerbang logika adalah komponen dasar dalam elektronika digital yang melakukan operasi logika pada sinyal biner (1 atau 0).
- **Fungsi:** Gerbang logika mengimplementasikan fungsi Boolean secara fisik dalam sirkuit elektronik.
- **Contoh:** AND, OR, NOT, XOR, NAND, NOR.
- **Tingkat Abstraksi:** Berada pada tingkat abstraksi yang lebih rendah, merupakan implementasi fisik dari operasi Boolean.

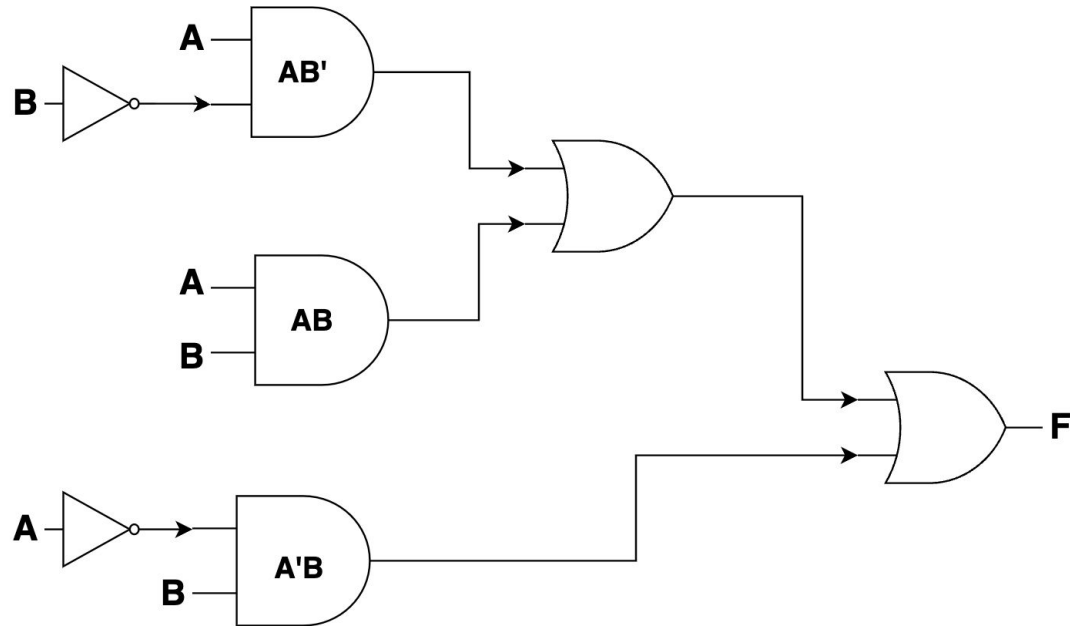
JENIS-JENIS GERBANG LOGIKA

- AND Gate
- OR Gate
- NOT Gate
- NAND Gate
- NOR Gate
- XOR Gate
- XNOR Gate

Name	NOT	AND	NAND	OR	NOR	XOR	XNOR																																																																																																
Alg. Expr.	\overline{A}	AB	\overline{AB}	$A+B$	$\overline{A+B}$	$A\oplus B$	$\overline{A\oplus B}$																																																																																																
Symbol																																																																																																							
Truth Table	<table><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>	A	X	0	1	1	0	<table><tr><th>B</th><th>A</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>	B	A	X	0	0	0	0	1	0	1	0	0	1	1	1	<table><tr><th>B</th><th>A</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>	B	A	X	0	0	1	0	1	1	1	0	1	1	1	0	<table><tr><th>B</th><th>A</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>	B	A	X	0	0	0	0	1	1	1	0	1	1	1	1	<table><tr><th>B</th><th>A</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>	B	A	X	0	0	1	0	1	0	1	0	0	1	1	0	<table><tr><th>B</th><th>A</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>	B	A	X	0	0	0	0	1	1	1	0	1	1	1	0	<table><tr><th>B</th><th>A</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>	B	A	X	0	0	1	0	1	0	1	0	0	1	1	1
A	X																																																																																																						
0	1																																																																																																						
1	0																																																																																																						
B	A	X																																																																																																					
0	0	0																																																																																																					
0	1	0																																																																																																					
1	0	0																																																																																																					
1	1	1																																																																																																					
B	A	X																																																																																																					
0	0	1																																																																																																					
0	1	1																																																																																																					
1	0	1																																																																																																					
1	1	0																																																																																																					
B	A	X																																																																																																					
0	0	0																																																																																																					
0	1	1																																																																																																					
1	0	1																																																																																																					
1	1	1																																																																																																					
B	A	X																																																																																																					
0	0	1																																																																																																					
0	1	0																																																																																																					
1	0	0																																																																																																					
1	1	0																																																																																																					
B	A	X																																																																																																					
0	0	0																																																																																																					
0	1	1																																																																																																					
1	0	1																																																																																																					
1	1	0																																																																																																					
B	A	X																																																																																																					
0	0	1																																																																																																					
0	1	0																																																																																																					
1	0	0																																																																																																					
1	1	1																																																																																																					

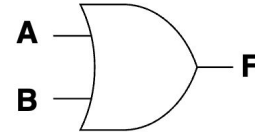
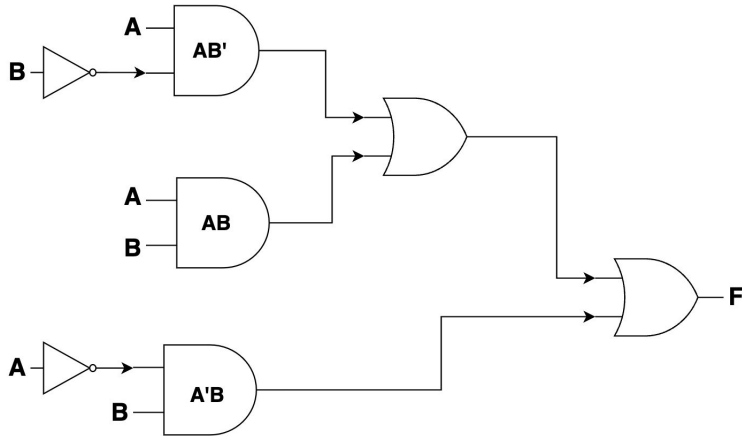
OPERATOR BOOLEAN VS GERBANG LOGIKA

$$F = AB' + AB + A'B$$



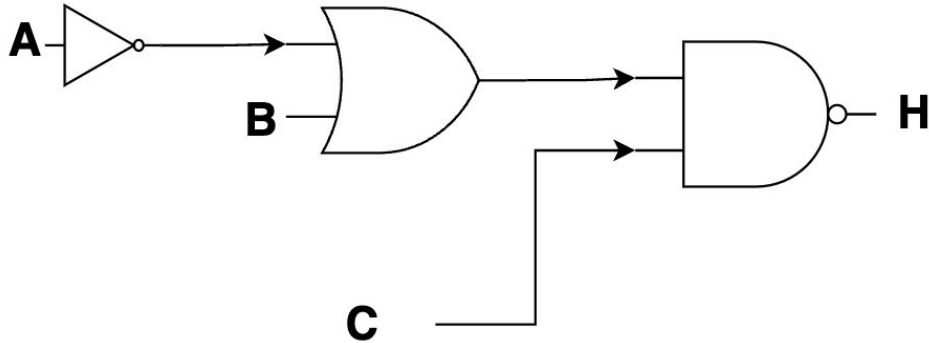
OPERATOR BOOLEAN VS GERBANG LOGIKA

$$F = AB' + AB + A'B = A + B$$



LATIHAN SOAL

- a) Tuliskan persamaan boolean dari rangkaian tersebut
- b) Buat tabel kebenaran
- c) Implementasikan rangkaian tersebut menggunakan gerbang NAND saja



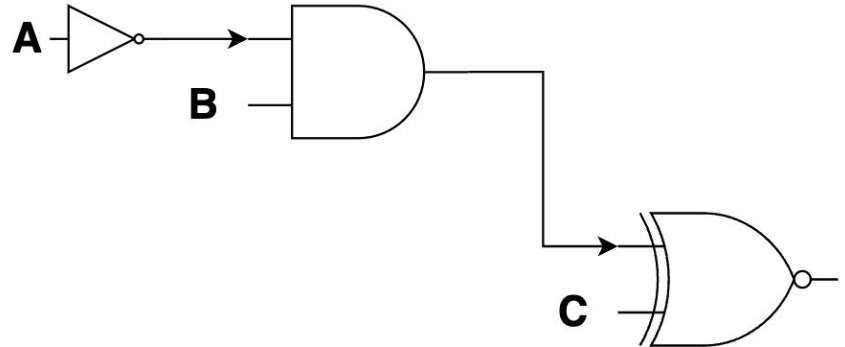
LATIHAN SOAL

2. Implementasikan gerbang XOR menggunakan hanya gerbang AND, OR, dan NOT.

3. a) Tulis persamaan boolean untuk rangkaian tersebut

b) Tentukan output Y untuk input berikut:

- $A=1, B=1, C=0$
- $A=0, B=1, C=1$
- $A=1, B=1, C=1$



LATIHAN SOAL

4. Buat gerbang logika dari fungsi berikut ini:

- $X = AB'C + A'BC + ABC + (A + B + C)(A' + B' + C')$