



# EKSPLORASI ALGORITMA MINIMISASI LOGIC

Kelompok 7

Atadila Belva Ganya -18320015

Diandra Ramadhani Putri Naja -18320023

Reina Puteri Ramadhani-18320039

# Table of Contents

---

**Pendahuluan**   **01**   

   **04**   **List Fungsi & Kegunaannya**

**Teori Dasar**   **02**   

   **05**   **Test Case**

**Analisis Algoritma**   **03**   

   **06**   **Demo**




# Pendahuluan

---

**Minimisasi *logic boolean algebra*** merupakan salah satu metode yang digunakan untuk memperkecil ukuran sebuah rangkaian digital. Secara umum, terdapat dua buah metode yang dapat digunakan untuk melakukan minimisasi *logic boolean algebra*, yakni metode Karnaugh Map (K-map) dan Quine-McCluskey (metode tabular). **Metode Karnaugh Map (K-Map)** memanfaatkan array dua dimensi sebagai representasi grafis dari *truth table* fungsi-fungsi *logic* yang setiap kotaknya mewakili minterm dalam bentuk *boolean*. Sedangkan **metode tabular** melibatkan dua buah tahapan, yakni menemukan seluruh *prime implicants* pada sebuah fungsi, kemudian memanfaatkannya untuk mencari *essential prime implicants* dalam fungsi terkait.

Keterbatasan metode karnaugh map dalam penggunaan jumlah variabel yang dapat ditinjau pada ekspresi fungsional terkait dan implementasinya dalam sebuah program komputer membuat metode tabular sebagai metode yang akan diterapkan dalam program eksplorasi algoritma minimasi logic. Metode tabular dapat memberikan *input* jumlah variabel lebih dari lima, hingga tak terbatas dan tahapan utama yang diterapkan pada metode tabular tidak banyak memanfaatkan kemampuan visual manusia, sehingga lebih mudah untuk diterapkan dalam sebuah program komputer.



# Teori Dasar

## A. Sinyal Digital dan Biner

**Sinyal digital** atau sinyal diskrit merupakan sinyal yang pada waktu tertentu dapat merepresentasikan satu set berhingga dari beberapa kemungkinan value. Input serta output digital dari sistem digital dapat diekspresikan melalui sinyal-sinyal dengan value biner (binary).

**Biner (binary)** merupakan representasi dari sinyal digital, yang terdiri atas satu dari dua kemungkinan value, yakni 1 (on) atau 0 (off). Sebuah sinyal biner tunggal dikenal dengan sebutan bit (binary digit). Jika diinginkan representasi nilai lebih dari 1, maka akan dibutuhkan digit lainnya,  $2^1$  hingga  $2^n$

contoh:

$$\begin{array}{cccc} 1 & 0 & 1 & \\ 2^3 & 2^2 & 2^1 & 2^0 \end{array}$$

biner "101", dapat disebut sebagai "satu nol satu, basis dua", mengekspresikan nilai  $1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 5$  pada sistem angka basis sepuluh.

# Teori Dasar



## B. Minimisasi Rangkaian Digital Kombinasional

Dalam mengonstruksi rangkaian digital, optimalisasi ukuran, kecepatan, serta konsumsi *power*. cara untuk meningkatkan kecepatan dan konsumsi *power* adalah dengan mereduksi ukuran rangkaian. Dengan memanfaatkan teorema aljabar boolean, sebuah ekspresi fungsional dapat disederhanakan. Sehingga, ukuran dari rangkaian digital yang direpresentasikan oleh ekspresi fungsional terkait dapat diperkecil.

Terdapat dua buah metode yang dapat diterapkan untuk menyederhanakan sebuah ekspresi fungsional *Boolean*, yakni Karnaugh map (K-map) dan Quine-McCluskey (metode tabular).

# Teori Dasar



## B. Minimisasi Rangkaian Digital Kombinasional

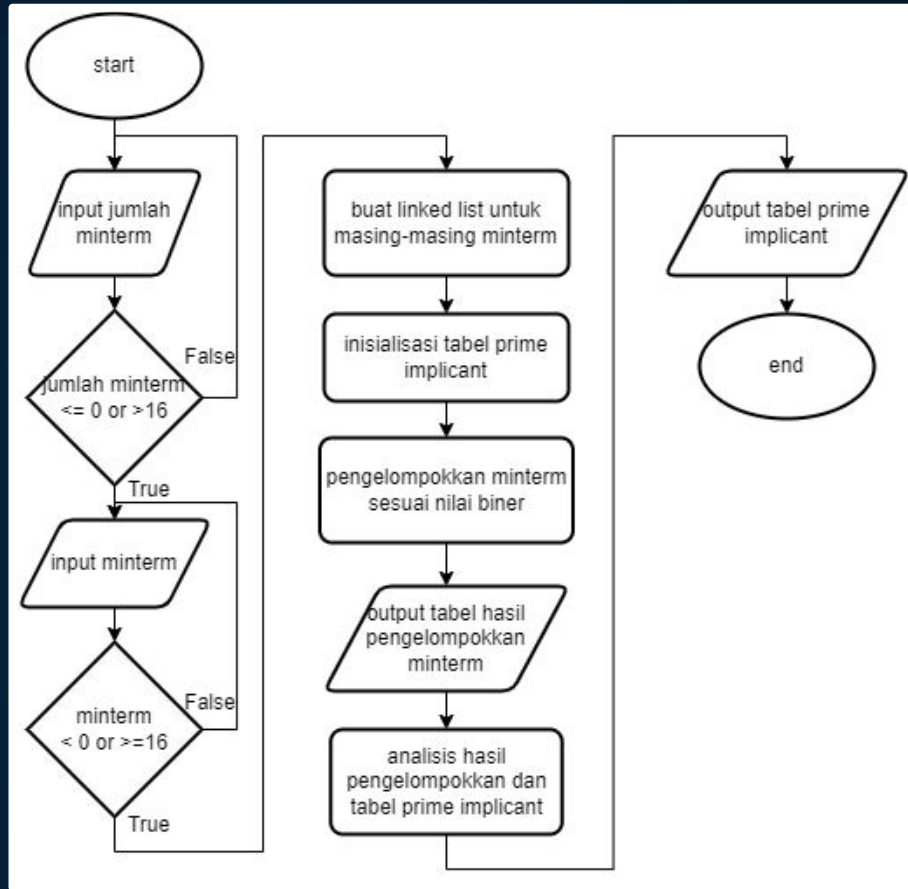
**Karnaugh map** merupakan metode pemanfaatan array dua dimensi sebagai representasi grafis dari *truth table* fungsi-fungsi *logic* yang setiap kotaknya mewakili sebuah minterm dalam fungsi *Boolean*, yang kemudian minterm-minterm tersebut akan dikelompokkan.

Metode karnaugh map memiliki keterbatasan dalam hal jumlah variabel yang dapat ditinjau pada ekspresi fungsional terkait, yakni minimal dua variabel dan maksimal lima variabel, serta implementasinya yang tidak mudah untuk diterapkan pada sebuah program komputer.

**Quine-McCluskey (metode tabular)** hadir dengan syarat jumlah variabel dapat melebihi lima (tidak terbatas) dan lebih mudah untuk diimplementasikan dalam program komputer. Secara umum, terdapat tahapan dalam melaksanakan minimisasi *logic* menggunakan metode tabular hingga mencapai kondisi dimana tidak dapat dilakukan tahapan tersebut lagi dan mendapat bentuk yang paling sederhana.



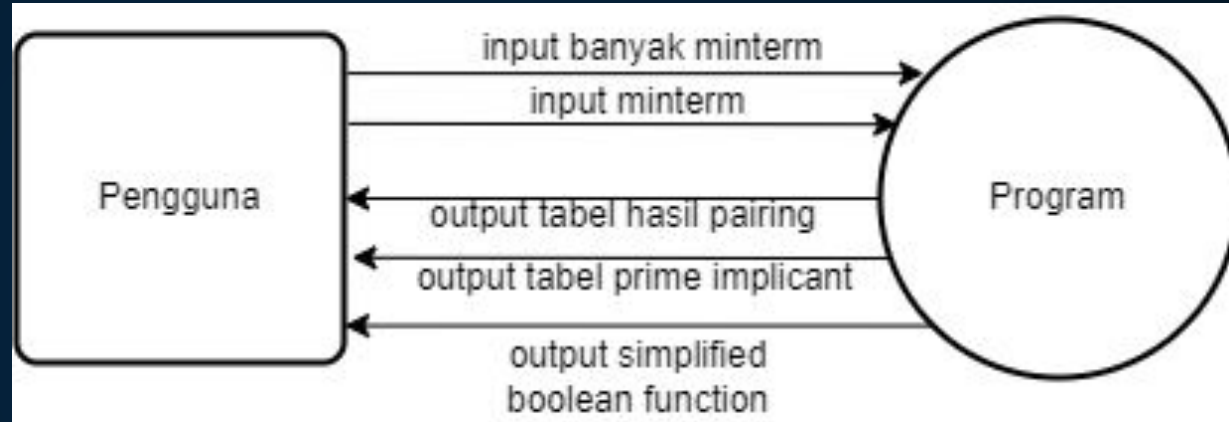
# Analisis Algoritma



Gambar 1.1. Flowchart



# Analisis Algoritma

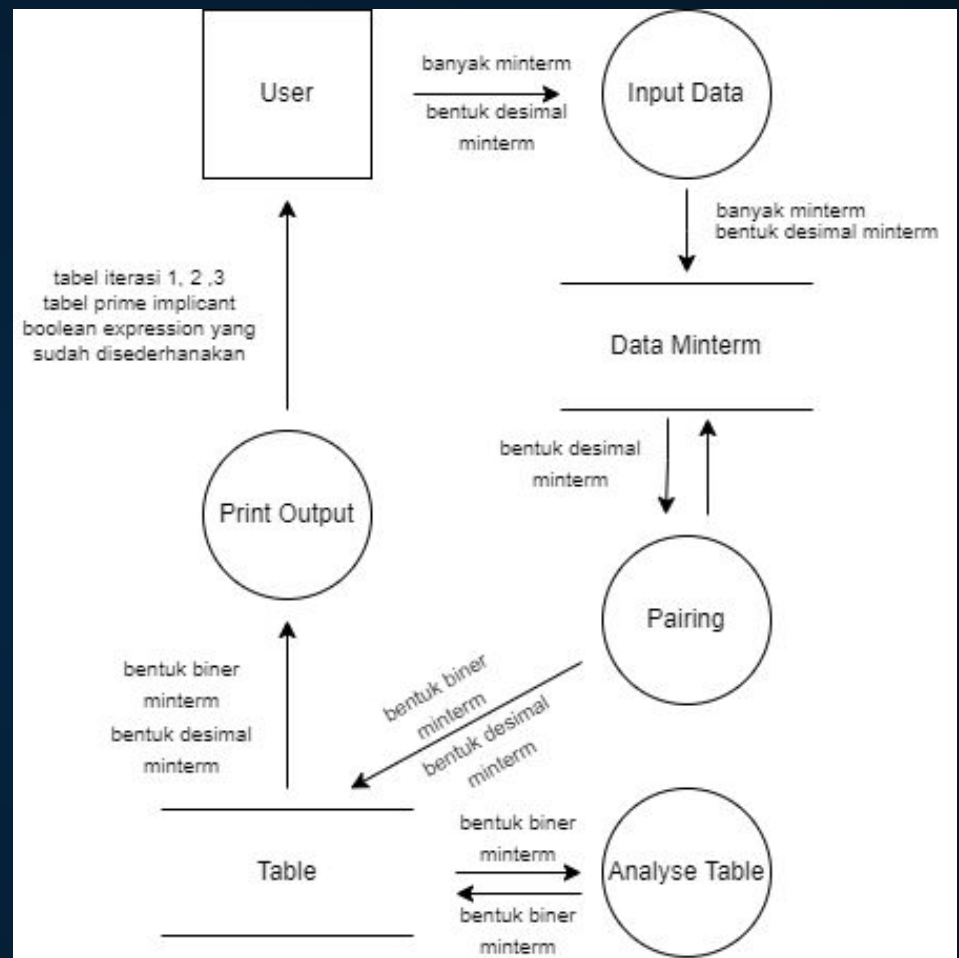


Gambar 1.2. DFD level 0





# Analisis Algoritma



Gambar 1.3. DFD level 1



# Analisis Algoritma

- Pertama, program akan meminta *input* berupa banyaknya minterm yang akan di minimisasi.
- Selanjutnya, program akan meminta bentuk desimal (dari 0 s.d. 15) dari minterm-minterm terkait sesuai jumlah yang telah di-*input*-kan.

Tabel representasi minterm dalam bentuk desimal dan biner yang dapat di-*input*-kan pada program. Nilai desimal pada tabel minterm diperoleh dari konversi biner ke desimal.

Variabel yang digunakan untuk melambangkan nilai 0 adalah huruf kecil dan huruf kapital melambangkan nilai biner 1. Setiap desimal akan direpresentasikan dalam 4 bit, sehingga banyaknya minterm yang dapat di minimisasi maksimal 16 minterm dengan *range* desimal minterm dari 0 s.d. 15.

AB \ CD	00	01	10	11
00	0 abcd	4 aBcd	8 Abcd	12 ABcd
01	1 abcD	5 aBCD	9 AbcD	13 ABcD
10	2 abCd	6 aBCd	10 AbCd	14 ABCd
11	3 abCD	7 aBCD	11 AbCD	15 ABCD

Gambar 1.4. Tabel Minterm



# Analisis Algoritma

---

Program akan melakukan minimisasi logic expression menggunakan metode tabular. Dalam minimisasi logic expression menggunakan metode ini, terdapat beberapa tahap yang dilakukan, yaitu sebagai berikut :

**Iterasi 1**, mula-mula, dilakukan konversi desimal minterm menjadi binernya, sesuai dengan yang telah dituliskan pada Tabel 1-1. Tabel Minterm. Kemudian, dari bentuk biner minterm tersebut, dilakukan pengurutan terhadap minterm-minterm yang ada, lalu dikelompokkan.

**iterasi 2**, dilakukan perbandingan antara setiap elemen (minterm). Perbandingan antara dua elemen tersebut akan menghasilkan komparasi elemen dengan hanya satu perbedaan digit bit yang dimiliki. Proses perbandingan akan kerap dilanjutkan hingga dihasilkan kelompok komparasi antara kelompok minterm yang memiliki jumlah bit berdigit "1" terbanyak dengan yang sebelumnya. Dalam pengaplikasiannya, digit bit yang sama akan ditulis kembali, sementara digit bit yang berbeda akan dituliskan sebagai underscore (  $\_$  ) atau strip ( - ).



# Analisis Algoritma

Pada **iterasi 3** dilakukan perbandingan antara setiap elemen pada kelompok-satu-iterasi-dua dengan setiap elemen pada kelompok-dua-iterasi-dua. Hasil komparasi yang didapat berupa sekelompok minterm dengan perbedaan pada salah satu digit bit-nya. Proses perbandingan ini akan kerap dilakukan hingga setiap elemen pada kelompok terakhir iterasi dua dibandingkan dengan setiap elemen pada kelompok sebelumnya. Jika syarat proses perbandingan dipenuhi, akan dituliskan kembali seluruh digit bit yang sama dan menuliskan digit bit yang berbeda dengan underscore ( \_ ) atau strip ( - ).

**Tabel Prime Implicants** didapat dengan melakukan listing terhadap fungsi-fungsi aljabar paling sederhana dari setiap minterm yang sebelumnya telah di-input-kan. Baris pertama pada tabel prime implicants akan merepresentasikan minterm-minterm yang pada awal program di-input-kan, secara terurut. Kolom pertama pada tabel prime implicants akan merepresentasikan bentuk aljabar dari digit-digit bit sekelompok minterm yang paling sederhana (prime implicants). Selanjutnya, dilakukan penandaan (checklist) terhadap kotak dengan posisi kolom (minterm) dan baris (prime implicants) yang saling berkaitan. Jadi, bentuk aljabar pada kolom pertama (prime implicants) dengan minterm essential prime implicants, menjadi jawaban atau hasil akhir penyederhanaan logic boolean algebra menggunakan metode tabular.

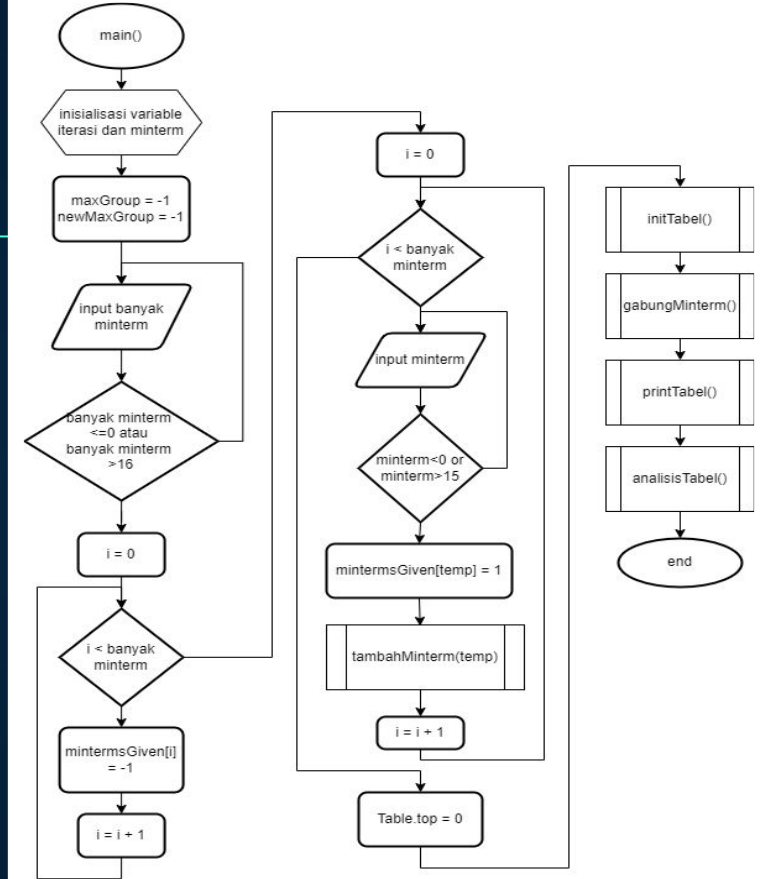


# Analisis Algoritma

Pada main function, program akan melakukan inisialisasi untuk beberapa variabel. Kemudian program akan meminta input berupa banyaknya minterm yang akan di minimisasi.

Input yang diberikan pengguna harus berjumlah lebih dari 0 dan kurang dari sama dengan 16. Apabila tidak, program akan melakukan looping dengan menampilkan pesan kesalahan dan meminta pengguna untuk memasukkan ulang banyak minterm.

Setelah benar, program akan meminta minterm yang akan di minimisasi sesuai dengan jumlah yang telah diberikan sebelumnya. Pada input ini, pengguna diminta untuk memasukkan input minterm dari rentang 0 - 15.



Gambar 1.5. Flowchart Main Function



# Analisis Algoritma

---

Apabila pengguna memberikan angka yang tidak sesuai dengan batasan input, program akan kembali menampilkan pesan kesalahan dan meminta pengguna memasukkan ulang minterm yang diinginkan.

Setiap minterm akan dibuat linked list baru dengan struktur data Node untuk menyimpan informasi dari setiap minterm menggunakan fungsi tambahMinterm().

setelah input berhasil dilakukan, program akan memanggil fungsi initTable() untuk menginisialisasi tabel prime implicant dengan mengisi setiap sel dalam tabel -1 untuk menandakan sel tersebut kosong.

Berikutnya akan dipanggil fungsi gabungMinterm() untuk melakukan pengelompokkan. Lalu tabel prime implicant akan ditampilkan pada output dengan memanggil fungsi printTable().

Tabel prime implicant akan dianalisis menggunakan fungsi analisisTabel() dan menampilkan hasil akhir berupa minterm yang telah disederhanakan.

# List Fungsi & Kegunaannya

No	Nama Fungsi	Kegunaan
1.	Struktur Data dan Variable Global	Terdiri dari struct vektor, struct node, dan struct implicantsTable
2.	tambahMinterm(int)	Digunakan untuk membuat linked list baru untuk menyimpan minterms yang diberikan
3.	buatNode(int)	Digunakan untuk membuat linked list baru pada setiap minterm untuk menyimpan informasi dari minterm tersebut.
4.	gabungMinterm()	Digunakan untuk melakukan pairing atau pengelompokan pada minterm. sampai proses pengelompokan tidak lagi dapat dilakukan.
5.	print()	Digunakan untuk menampilkan minterm dan minterm pasangannya serta nilai biner dalam setiap iterasi yang telah dilakukan.
6.	printTable()	Digunakan untuk menampilkan tabel prime implicant yang didapat dari informasi pada struktur data Table.

# List Fungsi & Kegunaannya

No	Nama Fungsi	Kegunaan
7.	<code>buatNodePair(node*, node*)</code>	digunakan untuk membuat linked list baru hasil pengelompokkan dua minterm sesuai dengan linked list yang diberikan pada parameter fungsi.
8.	<code>isiBiner(node*,node*, node*)</code>	digunakan untuk mengisi informasi nilai biner pada linked list baru hasil gabungan dari dua linked list minterm sebelumnya.
9.	<code>initTable()</code>	digunakan untuk menginisialisasi struktur data <code>implicantsTable</code> pada <code>Table.brr</code> dengan menyimpan seluruh nilai awal dengan nilai -1 untuk menandakan bahwa sel pada tabel tersebut masih kosong.
10.	<code>ifPairingPossible(node *, node*)</code>	digunakan untuk mengecek apakah terdapat perubahan atau perbedaan satu bit.
11.	<code>ifMintermPresentInImplicant(int,int)</code>	melakukan pengecekan apakah minterm tertentu ada dalam di suatu baris implicant.
12.	<code>tambahPair(node*, node*)</code>	digunakan untuk membuat linked list untuk menyimpan minterms yang berpasangan.



# List Fungsi & Kegunaannya

No	Nama Fungsi	Kegunaan
13.	tambahTabel()	menambahkan informasi yang diperoleh setelah proses iterasi dan pengelompokkan yang dilakukan pada minterm yang diberikan.
14.	analisisTabel()	digunakan untuk menganalisa tabel prime implicants untuk memperoleh essential prime implicant yang kemudian akan menjadi bentuk sederhana dari boolean function yang diberikan oleh user.
15.	konversiBiner(int)	digunakan untuk mengkonversi nilai biner ke dalam notasi variabelnya, kemudian menampilkannya sebagai output dari program.
16.	cariMax(int*)	digunakan untuk mencari prime implicant yang memiliki nilai minterm tertinggi yang tidak digunakan saat itu dan mengembalikan nilainya.
17.	jumlahImplicants(int, int*)	digunakan untuk menghitung banyaknya implicant yang ada di minterm tertentu
18.	hapusMinterm(int)	digunakan untuk menghapus minterm yang telah ditambahkan menjadi essential prime implicant dan kemudian dikonversi kedalam bentuk notasi variabel boolean dan dikeluarkan menjadi output hasil penyederhanaan.

# Test Case 1

Jika nilai input banyak minterm yang dimasukkan salah, maka akan ditampilkan keluaran/output peringatan salah. Selanjutnya, akan dilakukan iterasi memasukkan ulang banyak minterm yang akan diminimalisasi. jika nilai input minterm yang dimasukkan benar, maka selanjutnya akan diminta masukkan minterm yang akan diminimalisasi

banyak  
minterm>16

```
[MINIMIZATION LOGIC GATE]
Masukkan banyak minterm yang akan diminimalisasi :
17
Input yang dimasukkan salah! Jumlah minterm yang diberikan harus lebih dari 0 dan kurang dari sama dengan 16!
Masukkan banyak minterm yang akan diminimalisasi :
```

input banyak  
minterm <0

```
[MINIMIZATION LOGIC GATE]
Masukkan banyak minterm yang akan diminimalisasi :
-1
Input yang dimasukkan salah! Jumlah minterm yang diberikan harus lebih dari 0 dan kurang dari sama dengan 16!
Masukkan banyak minterm yang akan diminimalisasi :
```

input banyak  
minterm =0

```
[MINIMIZATION LOGIC GATE]
Masukkan banyak minterm yang akan diminimalisasi :
0
Input yang dimasukkan salah! Jumlah minterm yang diberikan harus lebih dari 0 dan kurang dari sama dengan 16!
Masukkan banyak minterm yang akan diminimalisasi :
```

# Test Case 2

Jika nilai input minterm yang akan diminimalisasi  $<0$  atau  $>15$  maka akan menampilkan output yang memberitahu bahwa input yang diberikan salah, lalu akan dilakukan iterasi memasukkan ulang nilai input minterm yang akan diminimalisasi hingga benar dan jumlahnya sesuai dengan input banyak minterm yang akan diminimalisasi. Nilai input yang benar akan disimpan.

```
[MINIMIZATION LOGIC GATE]
Masukkan banyak minterm yang akan diminimalisasi :
4
Masukkan minterm yang akan diminimalisasi :
1
-1
Input yang diberikan salah, masukkan ulang :
2
14
17
Input yang diberikan salah, masukkan ulang :
3
[Iterasi - 1]
1  0001
2  0010
3  0011
14 1110
[Iterasi - 2]
1,3  00-1
2,3  001-
[Tabel Prime Implicants]
ABCD  14
abD   1   3
abC   2   3
Boolean Expression telah disederhanakan menjadi : abD + abC + ABCd
Process returned 1 (0x1)   execution time : 33.708 s
Press any key to continue.
```

# Test Case 3

Nilai input banyak minterm yang akan di minimalisasi dan input minterm yang akan diminimalisasi benar akan ditampilkan keluaran/output berupa tabel iterasi-1, tabel iterasi-2, tabel prime implicants, dan boolean expression yang sudah disederhanakan

→ misal, minterm yang diinputkan adalah 1, 2, 4, 6

## [ Iterasi-1 ]

Group	Minterm	Biner				
G0	1	0	0	0	1	
	2	0	0	1	0	
	4	0	1	0	0	
G1	6	0	1	1	0	

## [ Iterasi-2 ]

Group	Minterm	Biner				
G0	2, 6	0	-	1	0	
	4, 6	0	1	-	0	

## [ Iterasi-3 ]

Group	Minterm	Biner				
-	-	-	-	-	-	

Tidak ada kelompok lain pada tabel iterasi 2 selain G0, sehingga elemen-elemennya tidak dapat dibandingkan dengan kelompok hasil iterasi 2 lainnya

# Test Case 3

Nilai input banyak minterm yang akan di minimalisasi dan input minterm yang akan diminimalisasi benar akan ditampilkan keluaran/output berupa tabel iterasi-1, tabel iterasi-2, tabel prime implicants, dan boolean expression yang sudah disederhanakan

Tabel *Prime Implicants*

<i>Prime Implicants</i> \ Minterm	1	2	4	6
aCd		✓		✓
aBd			✓	✓
abcD	✓			

→ *prime implicants* : aCd, aBd, abcD

→ *essential prime implicants* : abcD, aCd, aBd

→ fungsional aljabar yang telah disederhanakan:  $abcD + aCd + aBd$

# Test Case 3

Nilai input banyak minterm yang akan di minimalisasi dan input minterm yang akan diminimalisasi benar akan ditampilkan keluaran/output berupa tabel iterasi-1, tabel iterasi-2, tabel prime implicants, dan boolean expression yang sudah disederhanakan

```
[MINIMIZATION LOGIC GATE]
Masukkan banyak minterm yang akan diminimalisasi :
4
Masukkan minterm yang akan diminimalisasi :
1
2
4
6
[Iterasi - 1]
1 0001
2 0010
4 0100
6 0110
[Iterasi - 2]
2,6 0-10
4,6 01-0
[Tabel Prime Implicants]
abcD 1
aCd 2 6
aBd 4 6
Boolean Expression telah disederhanakan menjadi : abcD + aCd + aBd
Process returned 1 (0x1) execution time : 9.472 s
Press any key to continue.
```

# Test Case 4

Nilai input banyak minterm yang akan di minimalisasi dan input minterm yang akan di minimalisasi benar akan ditampilkan keluaran/output berupa tabel iterasi-1, tabel iterasi-2, iterasi-3 tabel prime implicants, dan boolean expression yang sudah disederhanakan

→ misal, minterm yang diinputkan adalah 0, 5, 6, 9, 10, 7, 13, 14, 15

[ Iterasi-1 ]

Group	Minterm	Biner			
G0	0	0	0	0	0
G1	5	0	1	0	1
	6	0	1	1	0
	9	1	0	0	1
	10	1	0	1	0
G2	7	0	1	1	1
	13	1	1	0	1
	14	1	1	1	0
G3	15	1	1	1	1

# Test Case 4

Nilai input banyak minterm yang akan di minimalisasi dan input minterm yang akan diminimalisasi benar akan ditampilkan keluaran/output berupa tabel iterasi-1, tabel iterasi-2, iterasi-3 tabel prime implicants, dan boolean expression yang sudah disederhanakan

[ Iterasi-2 ]

Group	Minterm	Biner			
G0	5, 7	0	1	_	1
	5, 13	_	1	0	1
	6, 7	0	1	1	_
	6, 14	_	1	1	0
	9, 13	1	_	0	1
	10, 14	1	_	1	0
G1	7, 15	_	1	1	1
	13, 15	1	1	_	1
	14, 15	1	1	1	_

[ Iterasi-3 ]

Group	Minterm	Biner			
G0	5,7 - 13, 15	_	1	_	1
	6,14 - 7,15	_	1	1	_



# Test Case 4

Nilai input banyak minterm yang akan di minimalisasi dan input minterm yang akan di minimalisasi benar akan ditampilkan keluaran/output berupa tabel iterasi-1, tabel iterasi-2, iterasi-3 tabel prime implicants, dan boolean expression yang sudah disederhanakan

**Tabel Prime Implicants**

Minterm Prime Implicants	0	5	6	7	9	10	13	14	15
BD		✓		✓					
BC			✓	✓				✓	✓
AcD					✓		✓		
ACd						✓		✓	
abcd	✓								

→ *prime implicants* : BD, BC, AcD, ACd, abcd

→ *essential prime implicants* : abcd, BD, BC, AcD, ACd

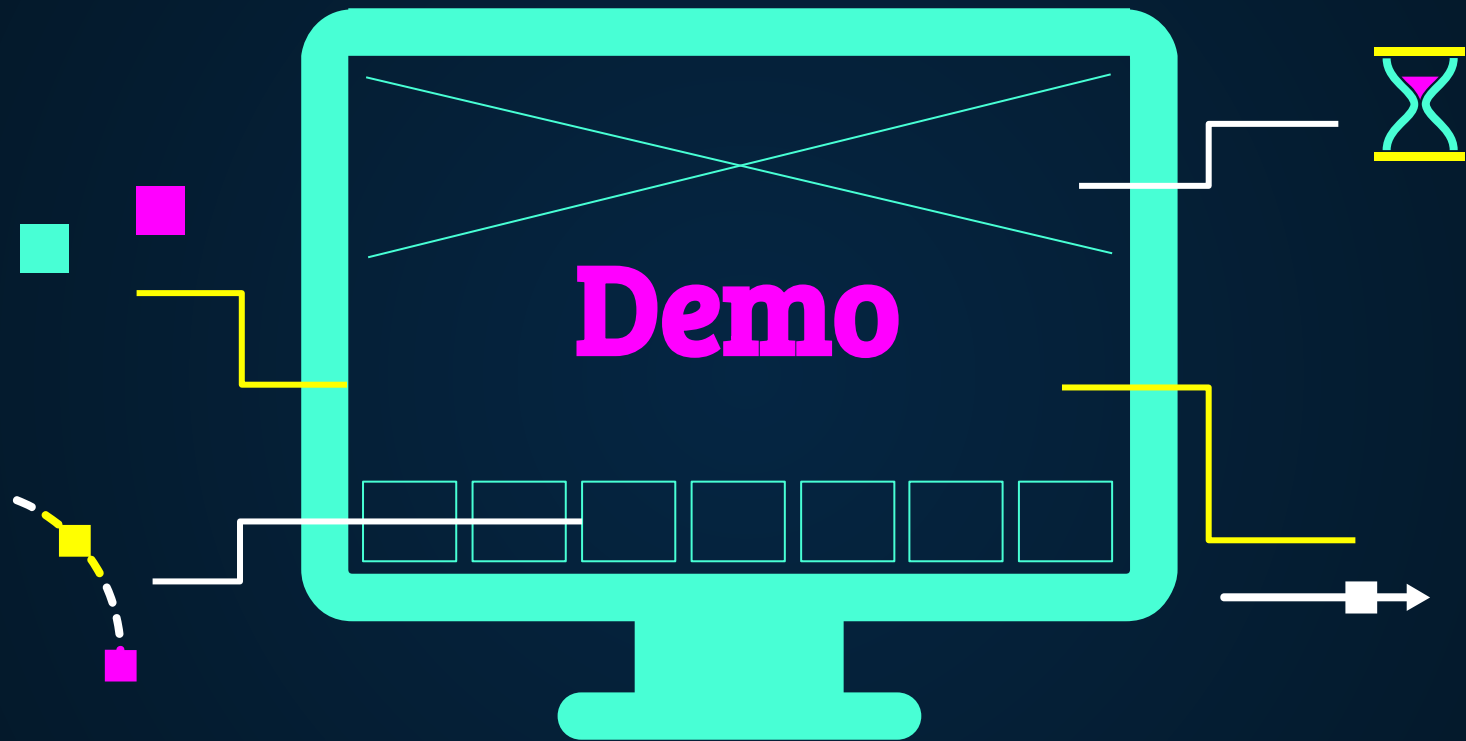
→ fungsional aljabar yang telah disederhanakan:  $abcd + BD + BC + AcD + ACd$

# Test Case 4

Nilai input banyak minterm yang akan di minimalisasi dan input minterm yang akan diminimalisasi benar akan ditampilkan keluaran/output berupa tabel iterasi-1, tabel iterasi-2, iterasi-3 tabel prime implicants, dan boolean expression yang sudah disederhanakan

```
[MINIMIZATION LOGIC GATE]
Masukkan banyak minterm yang akan diminimalisasi :
9
Masukkan minterm yang akan diminimalisasi :
0
5
6
9
10
7
13
14
15
[Iterasi - 1]
0 0000
5 0101
6 0110
9 1001
10 1010
7 0111
13 1101
14 1110
15 1111
```

```
[Iterasi - 2]
5,7 01-1
5,13 -101
6,7 011-
6,14 -110
9,13 1-01
10,14 1-10
7,15 -111
13,15 11-1
14,15 111-
[Iterasi - 3]
5,7,13,15 -1-1
5,13,7,15 -1-1
6,7,14,15 -11-
6,14,7,15 -11-
[Tabel Prime Implicants]
abcd 0
AcD 9 13
AcD 10 14
BD 5 7 13 15
BC 6 7 14 15
Boolean Expression telah disederhanakan menjadi :  $abcd + BD + BC + AcD + ACd$ 
```



# Kesimpulan

---

Di mata kuliah sebelumnya, kami lebih familiar dengan penggunaan Karnaugh Map. Namun untuk tugas besar kali ini, kami menggunakan Tabular Method untuk minimisasi logic boolean algebra. Hal ini dikarenakan, penggunaan Tabular Method dapat digunakan untuk variabel yang lebih banyak sedangkan untuk Karnaugh Map lebih terbatas. Melalui hal tersebut, kami memperoleh ilmu baru terkait metode lain yang dapat dilakukan untuk meminimisasi *logic boolean algebra*.

Pada mata kuliah sebelumnya, kami lebih menitikberatkan pada analisis manual minimisasi *logic boolean algebra*, sementara pada tugas besar ini, kami menitikberatkan analisis dalam implementasinya di sebuah program komputer. Sehingga, kami kembali menggali ilmu terkait penggambaran program dalam bentuk *flowchart* dan DFD, serta diperlukan analisis kembali terhadap beberapa contoh implementasinya yang telah ada di internet.

Dengan adanya keterbatasan variabel yang terjadi pada program yang disusun, kami merasa program masih dapat dikembangkan agar variabel-variabel yang dapat di-*input*-kan lebih dari 4.



# TERIMA KASIH

---

