TUGAS BESAR EL2008 – PEMECAHAN MASALAH DENGAN C

Program Minimisasi Logic

Farhan Hakim Iskandar (13220007)
Fitra Nurindra (13220011)
Muhammad Daffa Daniswara (13220043)
EL2008-Pemecahan Masalah dengan C
Teknik Elektro - Sekolah Teknik Elektro dan Informatika ITB

Abstrak

Pada tugas besar PMC ini, mahasiswa dibagi perkelompok untuk megeksplor metode-metode dalam minimisasi logic dan mengimplementasikannya pada program dengan bahasa C. Spesifikasi dibebaskan perkelompok.

Kata kunci: minimisasi, boolean, tabular, quine-mccluskey.

1. LATAR BELAKANG PERMASALAHAN

Secara umum, permasalahan yang terdapat pada tugas besar Mata Kuliah Pemecahan Masalah Dengan C (EL2008) adalah mengenai minimisasi fungsi logika. Sehingga secara garis besar, kita diminta untuk membuat program yang dapat melakukan minimisasi suatu fungsi logika. Minimisasi fungsi logika merupakan suatu proses menyederhanakan suatu fungsi *boolean algebra*. Setiap fungsi *boolean algebra* dapat dinyatakan dalam bentuk *sum of product* (minterm) atau *product of sum* (maxterms). Dalam melakukan penyederhanaan fungsi ekspresi *boolean algebra* dapat menggunakan beberapa cara, diantaranya yaitu metode manipulasi *boolean algebra*, Karnaugh Maps, dan Quine-McCluskey atau Metode Tabular. Beberapa literatur merekomendasikan penggunaan metode Quine-McCluskey karena dianggap paling baik dalam penyederhanaan dan dapat digunakan untuk variable fungsi *boolean algebra* yang besar. Metode tabulasi menggunakan tahap-tahap penyederhanaan yang jelas dan teratur dalam penyederhanaan suatu fungsi aljabar.

Minimisasi logika fungsi aljabar Boolean diperlukan untuk mengurangi penggunaan kompleksitas sirkuit supaya menghasilkan program atau sirkuit yang efisien untuk digunakan. Selain itu, penyederhanaan fungsi *boolean algebra* juga dapat mengurangi biaya dan penggunaan *logic gate* pada suatu sirkuit atau rangkaian tanpa mengubah hasil keluaran rangkaian tersebut. Penyederhanaan fungsi logika juga dapat mengurangi kerusakan pada sirkuit *logic gate* karena mengurangi beban kerja chip pada rangkaian tersebut. Jika rangkaian *logic gate* pada kehidupan sehari-hari tidak diminimisasi, atau dengan kata lain tidak dioptimalisasi, barang-barang digital yang digunakan saat ini tidak akan secepat itu dan lebih mahal karena komponen-komponennya yang tidak disederhanakan.

2. EKSPLORASI ALGORITMA MINIMISASI LOGIC

Logic minimization adalah proses menyederhanakan ekspresi boolean. Tujuan dari logic minimization ialah untuk memudahkan serta meningkatkan efisiensi suatu sistem. Pada saat eksplorasi metode, kami mencoba beberapa algoritma logic minimization, yakni Boolean Algebra, Karnaugh Map, dan Quine-McCluskey atau Metode Tabular. Kelompok kami memutuskan untuk menggunakan algoritma Quine-McCluskey untuk diimplementasikan ke dalam program yang akan kelompok kami buat untuk menyelesaikan permasalahan yang terdapat dalam tugas besar kali ini.

Algoritma Quine-McCluskey ini dipilih karena jika dibandingkan dengan metode lain, metode ini lebih mudah diimplementasikan ke dalam program dan direalisasikan secara komputasional. Hal ini disebabkan karena metode ini dilakukan banyak pengulangan atau looping. Selain itu, algoritma ini juga cocok untuk penyederhanaan logic dengan variabel yang banyak. Tahapan metode tabular ini adalah penentuan prime implicant dengan mencari matched pairs dari minterm. Selanjutnya penentuan minimum cover, yakni memilih prime implicant yang telah didapat untuk mencakup semua suku dan menghasilkan fungsi yang paling sederhana (essential prime implicant).

Secara umum, algoritma Quine-McCluskey didasarkan atas prinsip reduksi, misalkan terdapat suatu fungsi boolean algebra sebagai berikut.

$$f(A,B) = AB + AB' = A$$

Dalam fungsi tersebut, A dapat berupa variabel atau tim dan B adalah variabel. Hal tersebut berarti bahwa ketika dua buah minterm mengandung variabel yang sama hanya berbeda

dalam satu variabel, maka mereka bisa digabungkan bersama dan membentuk suatu ekspresi baru yang lebih sederhana. Kemudian proses reduksi tersebut akan terus dilakukan sampai tidak ada suku yang dapat digabungkan lagi. Istilah lain yang menandakan bahwa beberapa minterm tidak dapat direduksi lagi yaitu prime implicant (PI). Langkah terakhir dari algoritma QM ini adalah memilih set yang terdapat di dalam PI yang mengandung paling sedikit kemungkinan jumlah PI dan mencakup semua minterm yang terdapat dalam ekspresi fungsi boolean algebra. PI yang dipilih disebut dinamakan essential prime implicant (EPI). EPI tersebut mewakili ekspresi akhir dari fungsi boolean algebra yang paling minimum.

3. SPESIFIKASI DAN DESKRIPSI SIMULASI

Oleh karena algoritma yang kelompok kami gunakan adalah algoritma Quine-McCluskey, maka dapat diketahui bahwa *input* yang akan diberikan dari program simulasi yang akan kami buat adalah berupa informasi mengenai ekspresi fungsi *boolean algebra* yang ingin dicari ekspresi minimumnya.

Misalkan diberikan suatu fungsi *boolean algebra* sebagai berikut.

$$f(A,B,C,D) = \sum m(1,3,7,12,13,14,15)$$

Dari ekspresi fungsi *boolean algebra* tersebut, kelompok kami memutuskan bahwa *input* yang perlu diberikan oleh *user* untuk memberikan informasi mengenai ekspresi fungsi *boolean algebra* yang ingin dicari ekspresi minimumnya yaitu:

- Banyaknya variabel dari fungsi *boolean algebra*.
- Banyaknya minterm secara keseluruhan (termasuk *don't care* minterm).
- Banyaknya *don't care* minterm.
- Indeks minterm secara keseluruhan (termasuk *don't care* minterm) dalam bentuk desimal.
- Indeks *don't care* minterm dalam bentuk desimal.

Dalam setiap pemberian *input* program simulasi yang dibuat diharapkan dapat memvalidasi nilai *input* yang diberikan agar program simulasi dapat berjalan dengan baik. Validasi tersebut diantaranya:

- Banyaknya variabel maksimal dari fungsi *boolean algebra* hanya sebanyak 10 variabel dan jumlah variabel minimalnya adalah sebanyak 1 variabel.
- Banyaknya minterm keseluruhan haruslah ($0 < Tminterm \leq 2^{nVariable}$).
- Banyakya *don't care* minterm haruslah ($0 \leq DCminterm < Tminterm$).
- Indeks minterm yang diberikan harus dimulai dari indeks terkecil hingga indeks terbesar dan indeks minterm haruslah ($0 \leq IdxMT < 2^{nVariable}$).
- Indeks *don't care* minterm yang diberikan harus dimulai dari indeks terkecil hingga indeks terbesar dan indeks *don't care* minterm haruslah ($0 \leq IdxDCMT < 2^{nVariable}$).

Kemudian setelah *input* yang diberikan sudah valid, maka program simulasi akan memulai proses minimisasi fungsi logika. Proses awal yang akan dilakukan yaitu mengubah indeks minterm yang telah diberikan dalam bentuk desimal menjadi ke dalam bentuk binary. Kemudian algoritma Quine-McCluskey pun akan mulai dilakukan. Secara umum proses yang terdapat dalam program simulasi yang akan dibuat oleh kelompok kami adalah sebagai berikut (penjelasan algortima lebih lanjut akan terdapat pada bagian *flowchart* dan *data flow diagram*).

1. Langkah awal akan dibuat sebuah tabel (tabel 0) yang berisi minterm yang telah dikelompokkan ke dalam beberapa grup berdasarkan banyaknya angka 1 dalam representasi binary-nya. Untuk contoh ekspresi fungsi *boolean algebra* di atas akan di dapatkan tabel sebagai berikut.

Group	Minterm	Representasi Binary (ABCD)
Group 0	m1	0001
Group 1	m3 m12	0011 1100
Group 2	m7 m13 m14	0111 1101 1110
Group 3	m15	1111

Tabel 3.1. Tabel Inisialisasi

2. Kemudian antara 2 buah minterm yang memiliki 1 buah perbedaan akan saling dipasangkan dan hasilnya akan disimpan dalam sebuah tabel (tabel 1). Tabel yang diperoleh adalah sebagai berikut.

Group	*Matched-Pairs*	Representasi Binary (ABCD)
Group 0	m1-m3	00X1
Group 1	m3-m7 m12-m13 m12-m14	0X11 110X 11X0
Group 2	m7-m15 m13-m15 m14-m15	X111 11X1 111X

Tabel 3.2. Tabel Reduksi 1

3. Kemudian akan dicari kembali 2 buah pasangan minterm yang memiliki 1 buah perbedaan yang kemudian akan saling dipasangkan dan hasilnya akan disimpan dalam sebuah tabel (tabel 2). Tabel yang diperoleh adalah sebagai berikut.

Group	*Matched-Pairs*	Representasi Binary (ABCD)
Group 0	m1-m3	00X1
Group 1	m3-m7 m12-m13-m14-m15 m12-m14-m13-m15	0X11 11XX 11XX
Group 2	m7-m15	X111

Tabel 3.3. Tabel Reduksi 2

4. Kemudian untuk pasangan minterm yang memiliki representasi biner sama akan direduksi sehingga diperoleh hasil akhir berupa tabel prime implicant yang menandakan bahwa tidak ada lagi pasangan minterm yang dapat direduksi. Tabel yang diperoleh adalah sebagai berikut.

Group	*Matched-Pairs*	Representasi Binary (ABCD)	*Minimized Form*
Group 0	m1-m3	00X1	A'B'D
Group 1	m3-m7	0X11	A'CD
Group 2	m7-m15	X111	BCD
Group 3	m12-m13-m14-m15	11XX	AB

Tabel 3.4. Tabel Prime Implicant (PI)

5. Kemudian dari tabel prime implicant tersebut akan dipilih set yang terdapat di dalam PI yang mengandung paling sedikit kemungkinan jumlah PI dan mencakup semua minterm yang terdapat dalam ekspresi fungsi boolean algebra sehingga diperoleh tabel essential prime implicant sebagai berikut.

Prime Implicant	*Minterm Involved*	*Minterms given in the problem*						
		1	**3**	**7**	**12**	**13**	**14**	**15**
A'B'D	(1,3)	**X**	X					
A'CD	(3,7)		X	X				
BCD	(7,15)			X				X
AB	(12,13,14,15)				X	X	X	X

Tabel 3.5. Tabel Essential Prime Implicant (EPI)

Kemudian dari tabel EPI tersebut diperoleh hasil akhir atau *output* berupa hasil minimisasi dari ekspresi fungsi logika awal, *output* yang diperoleh adalah sebagai berikut.

$$f(A,B,C,D) = A'B'D + A'CD + AB$$

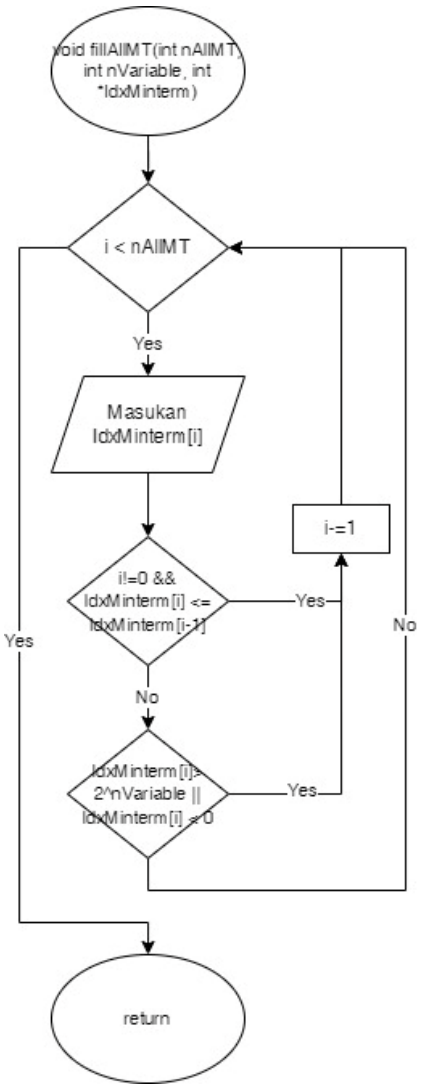
4. FLOWCHART

Untuk gambar flowchart yang lebih jelas secara keseluruhan dapat dilihat pada link berikut:

https://github.com/fitranurindra/Logic-Minimization/tree/main/Flowchart

1. *Flowchart* procedure fillallmt()

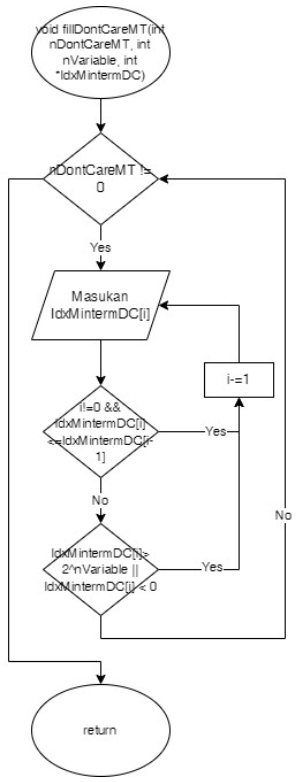
Fungsi ini digunakan untuk menerima input index minterm dari pengguna. Program akan memvalidasi input apakah index minterm lebih dari 2^(jumlah variabel) atau bernilai negatif. Untuk input kedua dan seterusnya divalidasi apakah index minterm ke-i lebih besar dari index minterm ke-(i-1).



Gambar 4.1. Flowchart Procedure fillAllMT()

2. *Flowchart* Procedure fillDontCareMT()

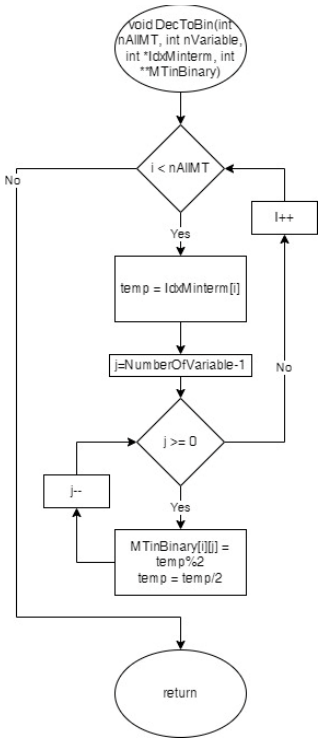
Fungsi ini digunakan untuk menerima input index don't care dari pengguna. Program akan memvalidasi input apakah index don't care lebih dari 2^(jumlah variabel) atau bernilai negatif. Untuk input kedua dan seterusnya divalidasi apakah index don't care ke-i lebih besar dari index minterm ke-(i-1).



Gambar 4.2. Flowchart Procedure fillDontCareMT ()

3. *Flowchart* Procedure DecToBin()

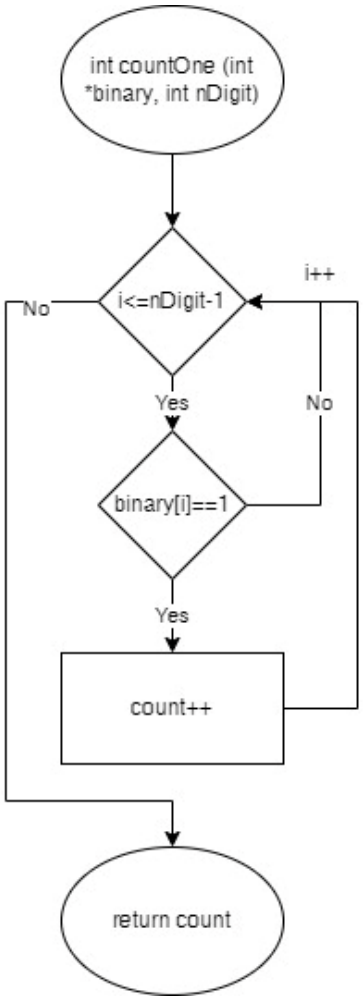
Fungsi ini digunakan untuk mengubah tiap index minterm menjadi representasi binernya. Konversi dari desimal ke biner menggunakan operasi modulo dan pembagian oleh angka 2.



Gambar 4.3. Flowchart Procedure DecToBin()

4 *Flowchart* Function countOne()

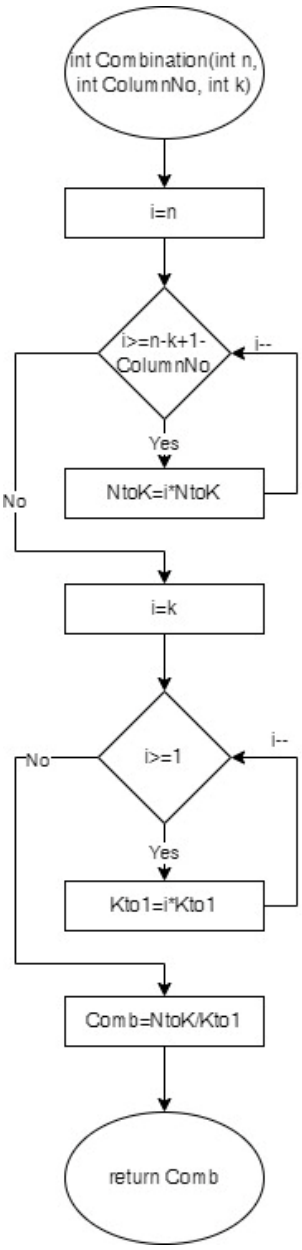
Fungsi ini digunakan untuk mengecek jumlah angka 1 pada representasi biner minterm. Prosedurnya adalah dengan mengecek tiap digit dari representasi biner minterm.



Gambar 4.4. Flowchart Procedure countOne()

5. *Flowchart* Function Combination()

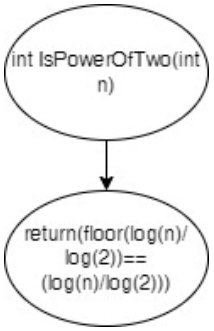
Fungsi ini digunakan untuk mencari jumlah kombinasi dari 3 input yang diberikan.



Gambar 4.5. Flowchart Procedure Combination()

6. *Flowchart* Function IsPowerOfTwo()

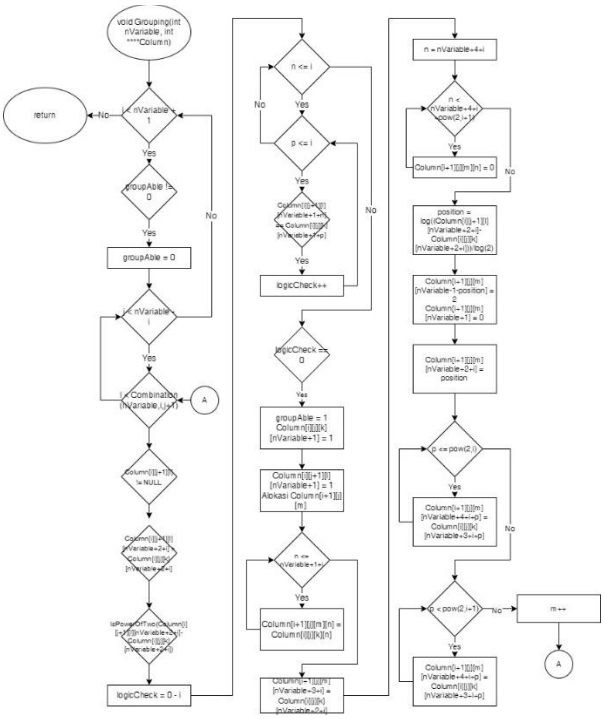
Fungsi ini digunakan untuk mengecek apakah input merupakan hasil pangkat dari 2.



Gambar 4.6. Flowchart Procedure IsPowerOfTwo()

7. *Flowchart* Procedure Grouping()

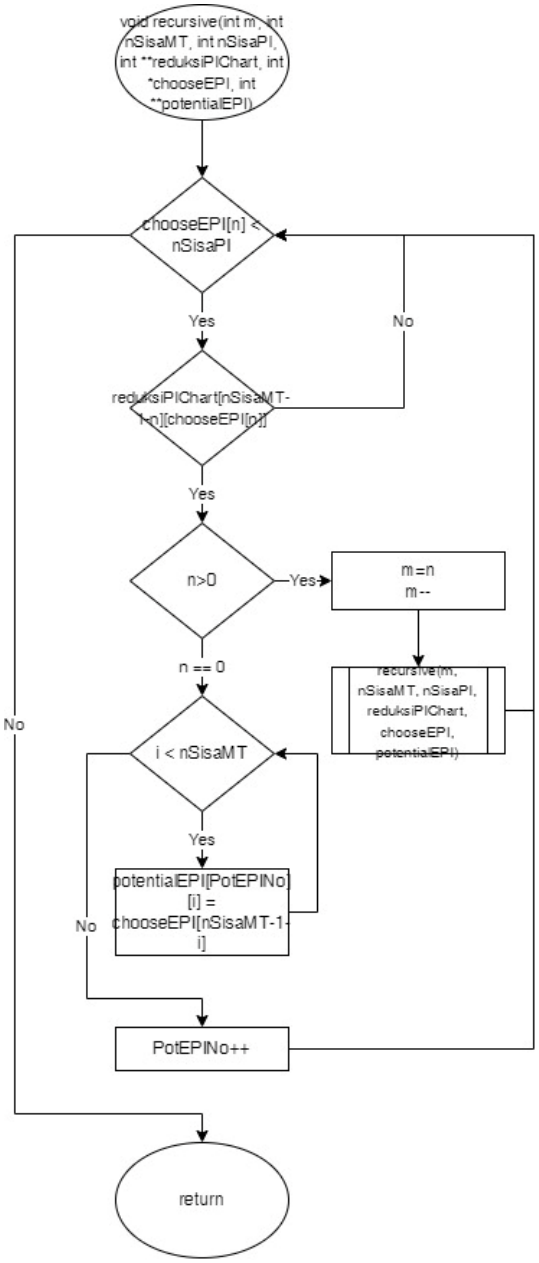
Fungsi ini digunakan untuk mengelompokkan minterm apabila dua buah minterm memiliki perbedaan hanya pada salah satu digitnya. Grouping dilakukan sampai minterm sudah tidak dapat disederhanakan kembali sehingga dihasilan prime implicant.



Gambar 4.7. Flowchart Procedure Grouping()

8. *Flowchart* Procedure recursive()

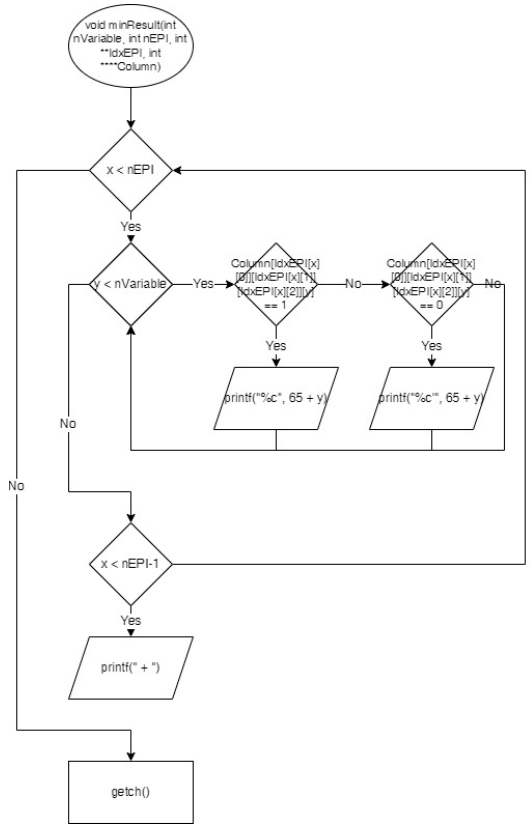
Fungsi ini digunakan untuk mencari prime implicant paling efisien (mencakup semua minterm yang ada) dari prime implicant yang ada.



Gambar 4.8. Flowchart Procedure recursive()

9. *Flowchart* Procedure minResult()

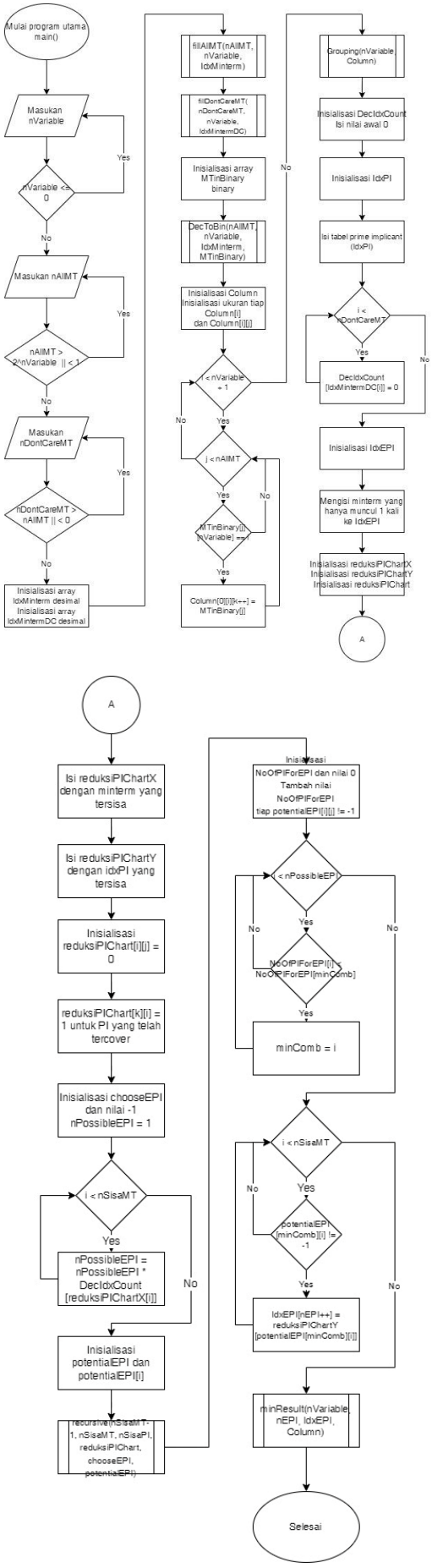
Fungsi ini digunakan untuk mencetak hasil minimisasi. Fungsi yang didapatkan akan mengecek dahulu apakah nilai truthnya 1 atau 0. Apabila bernilai 0, fungsi boolean akan dicetak menggunakan apostrophe dan jika 1 akan dicetak tanpa apostrophe.



Gambar 4.9. Flowchart Procedure minResult()

10. *Flowchart* Program Utama

Program akan menerima input jumlah variabel, jumlah minterm dan jumlah don't care. Kemudian program akan menerima dan memvalidasi input minterm menggunakan fungsi fillAllMT. Selanjutnya program juga akan menerima dan memvalidasi input don't care menggunakan fungsi fillDontCareMT. Program kemudian akan mengonversi minterm tersebut ke representasi binernya dengan fungsi DecToBin. Biner tersebut kemudian akan dikelompokkan berdasarkan banyaknya angka 1 di digitnya. Setelah itu tiap minterm akan dikelompokkan kembali apabila binernya hanya memiliki perbedaan pada salah satu digit. Proses tersebut berlangsung hingga tidak dapat dikelompokkan lagi dan didapatkan prime implicant. Prime implicant kemudian akan dihapus untuk minterm don't carenya. Dari sana prime implicant kemudian akan dicari essential prime implicant menggunakan fungsi recursive. Setelah selesai, akan dicetak fungsi aljabar boolean hasil minimisasi menggunakan fungsi minResult.



Gambar 4.10. Flowchart Program Utama

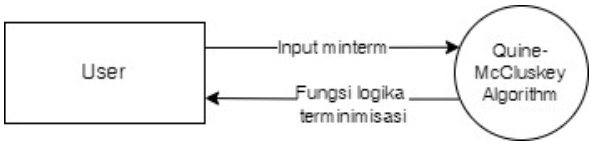
5. DATA FLOW DIAGRAM

Untuk gambar DFD yang lebih jelas secara keseluruhan dapat dilihat pada link berikut:

https://github.com/fitranurindra/Logic-Minimization/tree/main/DFD

1. *Data Flow Diagram (DFD) Level 0*

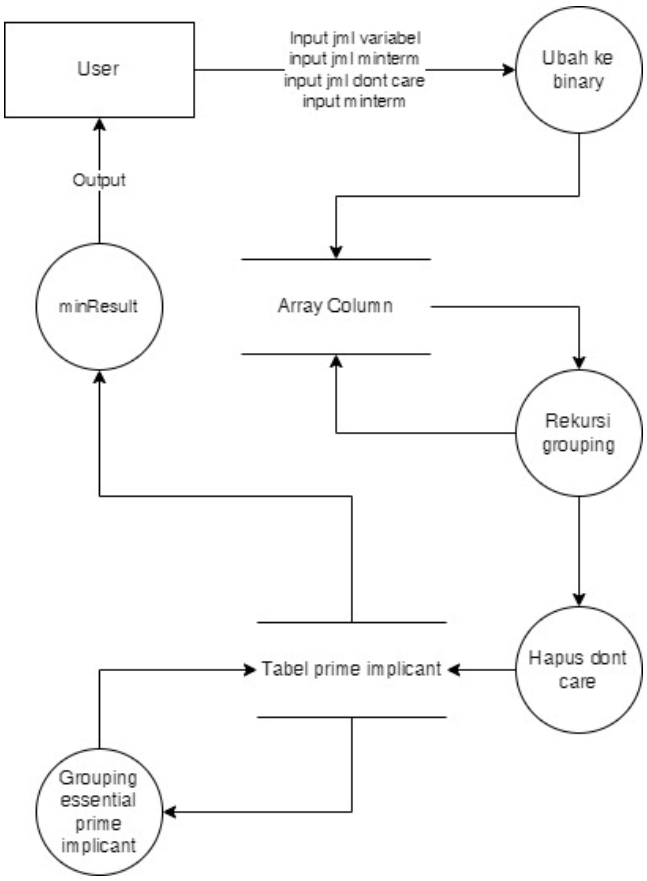
Program akan menerima input Program akan menerima input jumlah variabel, jumlah minterm dan jumlah don't care. Kemudian program juga akan menerima input minterm dan don't care dari pengguna. Input tersebut kemudian akan diproses menggunakan Quine-McCluskey Algorithm sehingga didapatkan output sebuah fungsi aljabar boolean yang sudah terminimisasi.



Gambar 5.1. DFD Level 0

2. *Data Flow Diagram (DFD) Level 1*

Program akan menerima input Program akan menerima input jumlah variabel, jumlah minterm dan jumlah don't care. Kemudian program juga akan menerima input minterm dan don't care dari pengguna. Minterm tersebut kemudian diconvert ke binary oleh program dan disimpan pada Column. Isi dari Column tersebut kemudian akan dikelompokkan berdasarkan banyaknya angka 1 di digitnya. Setelah itu tiap minterm akan dikelompokkan kembali apabila binernya hanya memiliki perbedaan pada salah satu digit. Proses tersebut berlangsung hingga tidak dapat dikelompokkan lagi dan didapatkan prime implicant. Hasil prime implicant tersebut kemudian akan dihapus minterm don't carenya kemudian dimasukkan ke dalam tabel prime implicant. Lalu prime implicant kemudian akan dicari essential prime implicant menggunakan fungsi recursive. Setelah selesai, akan dicetak fungsi aljabar boolean hasil minimisasi menggunakan fungsi minResult.



Gambar 5.2. DFD Level 1

6. LINK GITHUB *SOURCE CODE*

Berikut ini merupakan *link repository* GitHub dari kelompok kami.

https://github.com/fitranurindra/Logic-Minimization

Adapun penjelasan dari hierarki dan struktur *file* dalam *branch main* di *repository* GitHub tersebut adalah sebagai berikut.

1. Folder DFD

   Pada folder ini berisi seluruh gambar *data flow diagram* dari algoritma yang digunakan pada program minimisasi fungsi logika yang telah dibuat.

2. Folder *Flowchart*

   Pada folder ini berisi seluruh gambar *flowchart* dari algoritma yang digunakan pada program minimisasi fungsi logika yang telah dibuat.

3. LAP_TUBES_EL2008_KELOMPOK 6 (IEEE).pdf

   Pada file ini berisi laporan dari tugas besar Mata Kuliah Pemecahan Masalah dengan C (EL2008) mengenai minimisasi fungsi logika oleh kelompok 6 dengan format IEEE.

4. LAP_TUBES_EL2008_KELOMPOK 6.pdf

   Pada file ini berisi laporan dari tugas besar Mata Kuliah Pemecahan Masalah dengan C (EL2008) mengenai minimisasi fungsi logika oleh kelompok 6.

5. PPT_TUBES_EL2008_KELOMPOK 6.pdf

   Pada file ini berisi bahan presentasi dari tugas besar Mata Kuliah Pemecahan Masalah dengan C (EL2008) mengenai minimisasi fungsi logika oleh kelompok 6 dalam format PDF.

6. PPT_TUBES_EL2008_KELOMPOK 6.pptx

   Pada file ini berisi bahan presentasi dari tugas besar Mata Kuliah Pemecahan Masalah dengan C (EL2008) mengenai minimisasi fungsi logika oleh kelompok 6 dalam format PPTX.

7. README.md

   File markdown yang berisi penjelasan singkat mengenai spesifikasi dari rancangan simulasi program untuk melakukan minimisasi fungsi logika yang telah dibuat oleh kelompok 6.

8. main.c

   Pada file ini berisi implementasi dari program utama dalam bahasa C yang berisi algoritma untuk melakukan minimisasi fungsi logika yang telah dibuat oleh kelompok 6.

9. minimisasi_lib.c

   Pada file ini berisi implementasi program dalam bahasa C untuk library minimisasi (library yang menyimpan setiap fungsi-fungsi yang digunakan untuk melakukan minimisasi fungsi logika seperti fungsi fillAllMT(), fillDontCareMT(), DecToBin(), countOne(), Combination(), IsPowerOfTwo(), Grouping(), recursive(), minResult()).

10. minimisasi_lib.h

    Pada file ini berisi header dari library minimisasi (library yang menyimpan setiap fungsi-fungsi yang digunakan untuk melakukan minimisasi fungsi logika seperti fungsi fillAllMT(), fillDontCareMT(), DecToBin(), countOne(), Combination(), IsPowerOfTwo(), Grouping(), recursive(), minResult()).

7. SIMULASI PROGRAM

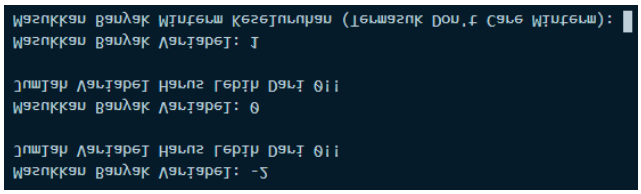
Untuk mengukur keakuratan program, dilakukan *benchmarking* atau perbandingan hasil minimisasi fungsi logika dari program yang telah dibuat dengan *logic minimization calculator* dari internet yaitu

https://www.charlie-coleman.com/experiments/kmap/.

Simulasi program yang akan dilakukan akan dibagi menjadi beberapa test kasus (*case test*) sebagai berikut.

Kasus 1: *Input* jumlah variabel *invalid*

Pada kasus 1 ini, program akan diberikan *input* jumlah variabel yang *invalid*. Program akan terus melakukan validasi hingga *user* memberikan *input* jumlah variabel yang *valid*. *Input* jumlah variabel yang *valid* adalah harus lebih besar dari 0. Untuk *input* jumlah variabel yang *invalid* diperoleh hasil pengujian sebagai berikut.

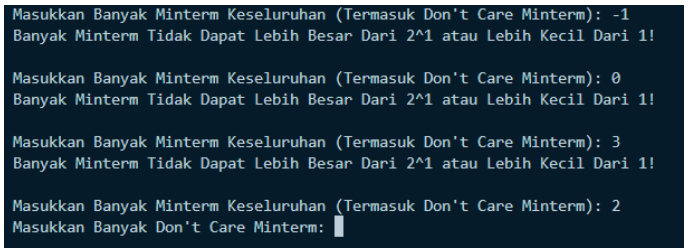


Gambar 7.1. Hasil Pengujian Kasus 1 (*Input* Jumlah Variabel *Invalid*)

Dari hasil pengujian tersebut dapat diketahui bahwa telah diperoleh hasil sesuai dengan yang diinginkan bahwa skema validasi telah berjalan sebagaimana mestinya.

Kasus 2: *Input* banyak minterm keseluruhan *invalid*

Pada kasus 2 ini, program akan diberikan *input* banyak minterm keseluruhan *invalid*. Program akan terus melakukan validasi hingga *user* memberikan *input* banyak minterm keseluruhan yang *valid*. *Input* banyak minterm keseluruhan yang *valid* adalah harus lebih besar dari 0 dan kurang dari $2^{nVariable} + 1$. Untuk *input* banyak minterm keseluruhan yang *invalid* diperoleh hasil pengujian sebagai berikut.



Gambar 7.2. Hasil Pengujian Kasus 2 (*Input* Banyak Minterm Keseluruhan *Invalid*)

Dari hasil pengujian tersebut dapat diketahui bahwa telah diperoleh hasil sesuai dengan yang diinginkan bahwa skema validasi telah berjalan sebagaimana mestinya.

Kasus 3: *Input* banyak *don't care* minterm *invalid*

Pada kasus 3 ini, program akan diberikan *input* banyak *don't care* minterm *invalid*. Program akan terus melakukan validasi hingga *user* memberikan *input* banyak *don't care* minterm yang *valid*. *Input* banyak *don't care* minterm yang *valid* adalah harus lebih besar dari sama dengan 0 dan kurang dari banyak minterm keseluruhan. Untuk *input* banyak *don't care* minterm yang *invalid* diperoleh hasil pengujian sebagai berikut.

```
Masukkan Banyak Minterm Keseluruhan (Termasuk Don't Care Minterm): 2
Masukkan Banyak Don't Care Minterm: 2
Banyak Don't Care Minterm Tidak Dapat Lebih Besar Dari Sama Dengan Banyak Minterm Keseluruhan atau Lebih Kecil Dari 0!

Masukkan Banyak Don't Care Minterm: 3
Banyak Don't Care Minterm Tidak Dapat Lebih Besar Dari Sama Dengan Banyak Minterm Keseluruhan atau Lebih Kecil Dari 0!

Masukkan Banyak Don't Care Minterm: -1
Banyak Don't Care Minterm Tidak Dapat Lebih Besar Dari Sama Dengan Banyak Minterm Keseluruhan atau Lebih Kecil Dari 0!

Masukkan Banyak Don't Care Minterm: 0

Masukkan Minterm Keseluruhan (Termasuk Don't Care Minterm)!

Masukkan Indeks Minterm ke-1 (Dalam Urutan Meningkat):
```

Gambar 7.3. Hasil Pengujian Kasus 3 (*Input* Banyak *Don't Care* Minterm *Invalid*)

Dari hasil pengujian tersebut dapat diketahui bahwa telah diperoleh hasil sesuai dengan yang diinginkan bahwa skema validasi telah berjalan sebagaimana mestinya.

Kasus 4: *Input* indeks minterm *invalid*

Pada kasus 4 ini, program akan diberikan *input* indeks minterm *invalid*. Program akan terus melakukan validasi hingga *user* memberikan *input* indeks minterm yang *valid*. *Input* indeks minterm yang *valid* adalah harus dimulai dari indeks terkecil hingga indeks terbesar (terurut membesar) serta *input* indeks minterm yang *valid* haruslah lebih besar dari sama dengan 0 dan kurang dari $2^{nVariable}$. Untuk *input* indeks minterm yang *invalid* diperoleh hasil pengujian sebagai berikut.

```
Masukkan Indeks Minterm ke-1 (Dalam Urutan Meningkat): 1
Masukkan Indeks Minterm ke-2 (Dalam Urutan Meningkat): 0
Input Indeks Minterm Tidak Dalam Urutan Meningkat
Masukkan Kembali Seluruh Indeks Minterm Dari Awal

Masukkan Indeks Minterm ke-1 (Dalam Urutan Meningkat): -1

Indeks Minterm Harus Lebih Besar Dari Sama Dengan 0 dan Lebih Kecil Dari 2
Masukkan Kembali Seluruh Indeks Minterm Dari Awal

Masukkan Indeks Minterm ke-1 (Dalam Urutan Meningkat): 2

Indeks Minterm Harus Lebih Besar Dari Sama Dengan 0 dan Lebih Kecil Dari 2
Masukkan Kembali Seluruh Indeks Minterm Dari Awal

Masukkan Indeks Minterm ke-1 (Dalam Urutan Meningkat): 0
Masukkan Indeks Minterm ke-2 (Dalam Urutan Meningkat): 1

Fungsi Logika Setelah Minimisasi Dalam Bentuk SOP:
```

Gambar 7.4. Hasil Pengujian Kasus 4 (*Input Indeks Minterm Invalid*)

Dari hasil pengujian tersebut dapat diketahui bahwa telah diperoleh hasil sesuai dengan yang diinginkan bahwa skema validasi telah berjalan sebagaimana mestinya.

Kasus 5: *Input* indeks *don't care* minterm *invalid*

Pada kasus 5 ini, program akan diberikan *input* indeks *don't care* minterm *invalid*. Program akan terus melakukan validasi hingga *user* memberikan *input* indeks *don't care* minterm yang *valid*. *Input* indeks *don't care* minterm yang *valid* adalah harus dimulai dari indeks terkecil hingga indeks terbesar (terurut membesar) serta *input* indeks *don't care* minterm yang *valid* haruslah lebih besar dari sama dengan 0 dan kurang dari $2^{nVariable}$. Untuk *input* indeks *don't care* minterm yang *invalid* diperoleh hasil pengujian sebagai berikut.

```
Masukkan Indeks Minterm yang Merupakan Don't Care Minterm!

Masukkan Indeks Don't Care Minterm ke-1 (Dalam Urutan Meningkat): 2
Masukkan Indeks Don't Care Minterm ke-2 (Dalam Urutan Meningkat): 3
Masukkan Indeks Don't Care Minterm ke-3 (Dalam Urutan Meningkat): 1
Input Indeks Don't Care Minterm Tidak Dalam Urutan Meningkat
Masukkan Kembali Seluruh Indeks Don't Care Minterm Dari Awal

Masukkan Indeks Don't Care Minterm ke-1 (Dalam Urutan Meningkat): -1

Indeks Don't Care Minterm Harus Lebih Besar Dari Sama Dengan 0 dan Lebih Kecil Dari 8
Masukkan Kembali Seluruh Indeks Don't Care Minterm Dari Awal

Masukkan Indeks Don't Care Minterm ke-1 (Dalam Urutan Meningkat): 8

Indeks Don't Care Minterm Harus Lebih Besar Dari Sama Dengan 0 dan Lebih Kecil Dari 8
Masukkan Kembali Seluruh Indeks Don't Care Minterm Dari Awal

Masukkan Indeks Don't Care Minterm ke-1 (Dalam Urutan Meningkat): 0
Masukkan Indeks Don't Care Minterm ke-2 (Dalam Urutan Meningkat): 1
Masukkan Indeks Don't Care Minterm ke-3 (Dalam Urutan Meningkat): 2

Fungsi Logika Setelah Minimisasi Dalam Bentuk SOP:
```

Gambar 7.5. Hasil Pengujian Kasus 5 (*Input Indeks Don't Care Minterm Invalid*)

Dari hasil pengujian tersebut dapat diketahui bahwa telah diperoleh hasil sesuai dengan yang diinginkan bahwa skema validasi telah berjalan sebagaimana mestinya.

Kasus 6: Kasus *input* 1 variabel

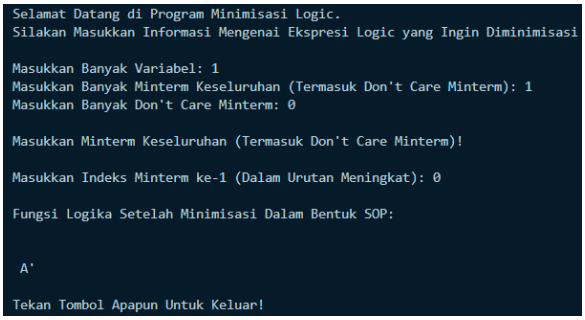
Pada kasus 6 ini, program akan diberikan *input* 1 variabel. Program akan melakukan minimisasi dari ekspresi fungsi logika berikut.

$$f(A) = \sum m(0)$$

Sehingga inputnya adalah:

- Banyak variabel = 1
- Banyak minterm keseluruhan = 1
- Banyak *don't care* minterm = 0
- Indeks minterm keseluruhan = 0

Untuk kasus ini diperoleh hasil pengujian sebagai berikut.



Gambar 7.6. Hasil Pengujian Kasus 6 (*Input* 1 Variabel)

Ketika dilakukan pengujian dengan menggunakan logic minimization calculator dari internet diperoleh hasil sebagai berikut.

Gambar 7.7. Hasil Pengujian Kasus 6 (*Input* 1 Variabel Dengan *Logic Minimization Calculator* Dari Internet)

Dari kedua hasil pengujian tersebut dapat diketahui bahwa telah diperoleh hasil yang sama dan sesuai dengan yang diharapkan, sehingga simulasi program telah sebagaimana mestinya untuk kasus *input* 1 variabel.

Kasus 7: Kasus input 2 variabel tanpa don't care

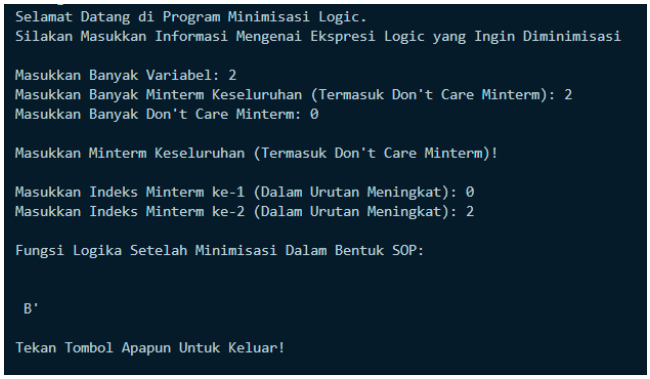
Pada kasus 7 ini, program akan diberikan input 2 variabel tanpa don't care. Program akan melakukan minimisasi dari ekspresi fungsi logika berikut.

$$f(A,B) = \sum m(0,2)$$

Sehingga inputnya adalah:

- Banyak variabel = 2
- Banyak minterm keseluruhan = 2
- Banyak don't care minterm = 0
- Indeks minterm keseluruhan = 0, 2

Untuk kasus ini diperoleh hasil pengujian sebagai berikut.



Gambar 7.8. Hasil Pengujian Kasus 7 (*Input* 2 Variabel Tanpa *Don't Care*)

Ketika dilakukan pengujian dengan menggunakan *logic minimization calculator* dari internet diperoleh hasil sebagai berikut.

Gambar 7.9. Hasil Pengujian Kasus 7 (Input 2 Variabel Tanpa Don't Care Dengan Logic Minimization Calculator Dari Internet)

Dari kedua hasil pengujian tersebut dapat diketahui bahwa telah diperoleh hasil yang sama dan sesuai dengan yang diharapkan, sehingga simulasi program telah sebagaimana mestinya untuk kasus *input* 2 variabel tanpa *don't care*.

Kasus 8: Kasus input 2 variabel dengan don't care

Pada kasus 8 ini, program akan diberikan input 2 variabel dengan don't care. Program akan melakukan minimisasi dari ekspresi fungsi logika berikut.

$$f(A,B) = \sum m(0,3) + \sum d(2)$$

Sehingga inputnya adalah:

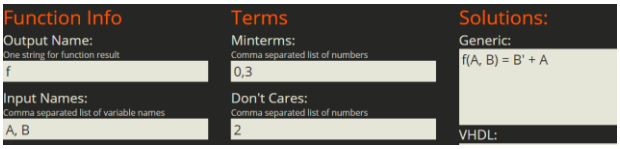
- Banyak variabel = 2
- Banyak minterm keseluruhan = 3
- Banyak don't care minterm = 1
- Indeks minterm keseluruhan = 0, 2, 3
- Indeks don't care minterm = 2

Untuk kasus ini diperoleh hasil pengujian sebagai berikut.

```
Selamat Datang di Program Minimisasi Logic.
Silakan Masukkan Informasi Mengenai Ekspresi Logic yang Ingin Diminimisasi

Masukkan Banyak Variabel: 2
Masukkan Banyak Minterm Keseluruhan (Termasuk Don't Care Minterm): 3
Masukkan Banyak Don't Care Minterm: 1

Masukkan Minterm Keseluruhan (Termasuk Don't Care Minterm)!

Masukkan Indeks Minterm ke-1 (Dalam Urutan Meningkat): 0
Masukkan Indeks Minterm ke-2 (Dalam Urutan Meningkat): 2
Masukkan Indeks Minterm ke-3 (Dalam Urutan Meningkat): 3

Masukkan Indeks Minterm yang Merupakan Don't Care Minterm!

Masukkan Indeks Don't Care Minterm ke-1 (Dalam Urutan Meningkat): 2

Fungsi Logika Setelah Minimisasi Dalam Bentuk SOP:


 B' + A

Tekan Tombol Apapun Untuk Keluar!
```

Gambar 7.10. Hasil Pengujian Kasus 8 (Input 2 Variabel Dengan Don't Care)

Ketika dilakukan pengujian dengan menggunakan logic minimization calculator dari internet diperoleh hasil sebagai berikut.



Gambar 7.11. Hasil Pengujian Kasus 8 (Input 2 Variabel Dengan Don't Care Dengan Logic Minimization Calculator Dari Internet)

Dari kedua hasil pengujian tersebut dapat diketahui bahwa telah diperoleh hasil yang sama dan sesuai dengan yang diharapkan, sehingga simulasi program telah sebagaimana mestinya untuk kasus input 2 variabel dengan don't care.

Kasus 9: Kasus input 3 variabel tanpa don't care

Pada kasus 9 ini, program akan diberikan input 3 variabel tanpa don't care. Program akan melakukan minimisasi dari ekspresi fungsi logika berikut.

$$f(A,B,C) = \sum m(0,1,4,6)$$

Sehingga inputnya adalah:

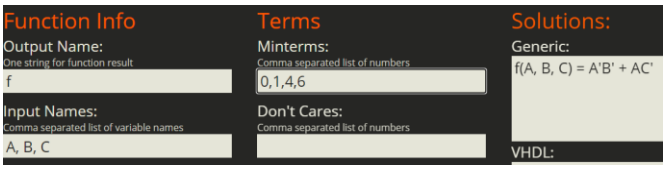
- Banyak variabel = 3
- Banyak minterm keseluruhan = 4
- Banyak don't care minterm = 0
- Indeks minterm keseluruhan = 0, 1, 4, 6

Untuk kasus ini diperoleh hasil pengujian sebagai berikut.

```
Selamat Datang di Program Minimisasi Logic.
Silakan Masukkan Informasi Mengenai Ekspresi Logic yang Ingin Diminimisasi

Masukkan Banyak Variabel: 3
Masukkan Banyak Minterm Keseluruhan (Termasuk Don't Care Minterm): 4
Masukkan Banyak Don't Care Minterm: 0

Masukkan Minterm Keseluruhan (Termasuk Don't Care Minterm)!

Masukkan Indeks Minterm ke-1 (Dalam Urutan Meningkat): 0
Masukkan Indeks Minterm ke-2 (Dalam Urutan Meningkat): 1
Masukkan Indeks Minterm ke-3 (Dalam Urutan Meningkat): 4
Masukkan Indeks Minterm ke-4 (Dalam Urutan Meningkat): 6

Fungsi Logika Setelah Minimisasi Dalam Bentuk SOP:


 A'B' + AC'

Tekan Tombol Apapun Untuk Keluar!
```

Gambar 7.12. Hasil Pengujian Kasus 9 (Input 3 Variabel Tanpa Don't Care)

Ketika dilakukan pengujian dengan menggunakan logic minimization calculator dari internet diperoleh hasil sebagai berikut.



Gambar 7.13. Hasil Pengujian Kasus 9 (Input 3 Variabel Tanpa Don't Care Dengan Logic Minimization Calculator Dari Internet)

Dari kedua hasil pengujian tersebut dapat diketahui bahwa telah diperoleh hasil yang sama dan sesuai dengan yang diharapkan, sehingga simulasi program telah sebagaimana mestinya untuk kasus input 3 variabel tanpa don't care.

Kasus 10: Kasus input 3 variabel dengan don't care

Pada kasus 10 ini, program akan diberikan input 3 variabel dengan don't care. Program akan melakukan minimisasi dari ekspresi fungsi logika berikut.

$$f(A,B,C) = \sum m(0,1,4,7) + \sum d(3)$$

- Sehingga inputnya adalah:
- Banyak variabel = 3
- Banyak minterm keseluruhan = 5
- Banyak don't care minterm = 1
- Indeks minterm keseluruhan = 0, 1, 3, 4, 7
- Indeks don't care minterm = 3

Untuk kasus ini diperoleh hasil pengujian sebagai berikut.

```
Selamat Datang di Program Minimisasi Logic.
Silakan Masukkan Informasi Mengenai Ekspresi Logic yang Ingin Diminimisasi

Masukkan Banyak Variabel: 3
Masukkan Banyak Minterm Keseluruhan (Termasuk Don't Care Minterm): 5
Masukkan Banyak Don't Care Minterm: 1

Masukkan Minterm Keseluruhan (Termasuk Don't Care Minterm)!

Masukkan Indeks Minterm ke-1 (Dalam Urutan Meningkat): 0
Masukkan Indeks Minterm ke-2 (Dalam Urutan Meningkat): 1
Masukkan Indeks Minterm ke-3 (Dalam Urutan Meningkat): 3
Masukkan Indeks Minterm ke-4 (Dalam Urutan Meningkat): 4
Masukkan Indeks Minterm ke-5 (Dalam Urutan Meningkat): 7

Masukkan Indeks Minterm yang Merupakan Don't Care Minterm!

Masukkan Indeks Don't Care Minterm ke-1 (Dalam Urutan Meningkat): 3

Fungsi Logika Setelah Minimisasi Dalam Bentuk SOP:


 B'C' + BC + A'B'

Tekan Tombol Apapun Untuk Keluar!
```

Gambar 7.14. Hasil Pengujian Kasus 10 (Input 3 Variabel Dengan Don't Care)

Ketika dilakukan pengujian dengan menggunakan logic minimization calculator dari internet diperoleh hasil sebagai berikut.

Gambar 7.15. Hasil Pengujian Kasus 10 (Input 3 Variabel Dengan Don't Care Dengan Logic Minimization Calculator Dari Internet)

Dari kedua hasil pengujian tersebut dapat diketahui bahwa telah diperoleh hasil yang sama dan sesuai dengan yang diharapkan, sehingga simulasi program telah sebagaimana mestinya untuk kasus input 3 variabel dengan don't care.

Kasus 11: Kasus input 4 variabel tanpa don't care

Pada kasus 11 ini, program akan diberikan input 4 variabel tanpa don't care. Program akan melakukan minimisasi dari ekspresi fungsi logika berikut.

$$f(A,B,C,D) = \sum m(1,3,7,12,13,14,15)$$

Sehingga inputnya adalah:

- Banyak variabel = 4
- Banyak minterm keseluruhan = 7
- Banyak don't care minterm = 0
- Indeks minterm keseluruhan = 1, 3, 7, 12, 13, 14, 15

Untuk kasus ini diperoleh hasil pengujian sebagai berikut.

```
Selamat Datang di Program Minimisasi Logic.
Silakan Masukkan Informasi Mengenai Ekspresi Logic yang Ingin Diminimisasi

Masukkan Banyak Variabel: 4
Masukkan Banyak Minterm Keseluruhan (Termasuk Don't Care Minterm): 7
Masukkan Banyak Don't Care Minterm: 0

Masukkan Minterm Keseluruhan (Termasuk Don't Care Minterm)!

Masukkan Indeks Minterm ke-1 (Dalam Urutan Meningkat): 1
Masukkan Indeks Minterm ke-2 (Dalam Urutan Meningkat): 3
Masukkan Indeks Minterm ke-3 (Dalam Urutan Meningkat): 7
Masukkan Indeks Minterm ke-4 (Dalam Urutan Meningkat): 12
Masukkan Indeks Minterm ke-5 (Dalam Urutan Meningkat): 13
Masukkan Indeks Minterm ke-6 (Dalam Urutan Meningkat): 14
Masukkan Indeks Minterm ke-7 (Dalam Urutan Meningkat): 15

Fungsi Logika Setelah Minimisasi Dalam Bentuk SOP:


 A'B'D + A'CD + AB

Tekan Tombol Apapun Untuk Keluar!
```

Gambar 7.16. Hasil Pengujian Kasus 11 (Input 4 Variabel Tanpa Don't Care)

Ketika dilakukan pengujian dengan menggunakan logic minimization calculator dari internet diperoleh hasil sebagai berikut.

Gambar 7.17. Hasil Pengujian Kasus 11 (Input 4 Variabel Tanpa Don't Care Dengan Logic Minimization Calculator Dari Internet)

Dari kedua hasil pengujian tersebut dapat diketahui bahwa telah diperoleh hasil yang sama dan sesuai dengan yang diharapkan, sehingga simulasi program telah sebagaimana mestinya untuk kasus input 4 variabel tanpa don't care.

Kasus 12: Kasus input 4 variabel dengan don't care

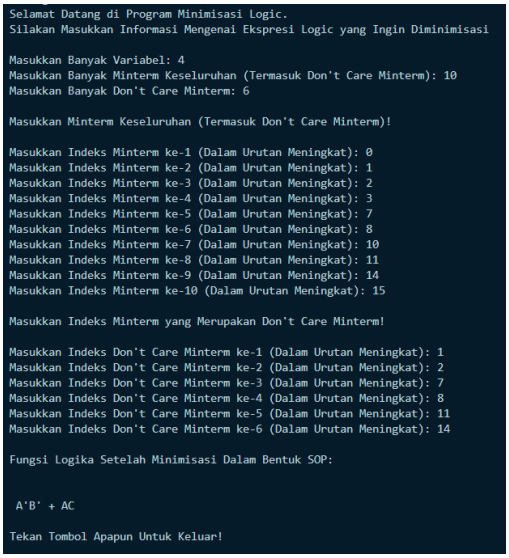
Pada kasus 12 ini, program akan diberikan input 4 variabel dengan don't care. Program akan melakukan minimisasi dari ekspresi fungsi logika berikut.

$$f(A,B,C,D) = \sum m(0,3,10,15) + \sum d(1,2,7,8,11,14)$$

Sehingga inputnya adalah:

- Banyak variabel = 4
- Banyak minterm keseluruhan = 10
- Banyak don't care minterm = 6
- Indeks minterm keseluruhan = 0, 1, 2, 3, 7, 8, 10, 11, 14, 15
- Indeks don't care minterm = 1, 2, 7, 8, 11, 14

Untuk kasus ini diperoleh hasil pengujian sebagai berikut.



Gambar 7.18. Hasil Pengujian Kasus 12 (Input 4 Variabel Dengan Don't Care)

Ketika dilakukan pengujian dengan menggunakan logic minimization calculator dari internet diperoleh hasil sebagai berikut.

Gambar 7.19. Hasil Pengujian Kasus 12 (Input 4 Variabel Dengan Don't Care Dengan Logic Minimization Calculator Dari Internet)

Dari kedua hasil pengujian tersebut dapat diketahui bahwa telah diperoleh hasil yang sama dan sesuai dengan yang diharapkan, sehingga simulasi program telah sebagaimana mestinya untuk kasus input 4 variabel dengan don't care.

8. KESIMPULAN DAN *LESSON LEARNED*

Logic minimization adalah proses menyederhanakan ekspresi boolean dan tujuan dari *logic minimization* ialah untuk memudahkan serta meningkatkan efisiensi suatu sistem. Pada program minimisasi fungsi *boolean algebra* dengan menggunakan bahasa C yang telah dibuat oleh kelompok kami, dapat diketahui bahwa program telah dapat dijalankan dengan baik dan mendapatkan hasil sesuai dengan yang diharapkan. Program minimisasi dengan menggunakan metode Quine-McCluskey dapat diimplementasikan dalam bahasa pemrograman C. Program minimisasi fungsi logika yang telah dibuat oleh kelompok kami memanfaatkan metode tabulasi menggunakan suku minterms untuk mendapatkan jumlahnya dan dilakukan pengelompokan sampai membentuk *essential prime implicant* yang menunjukkan bahwa fungsi *boolean algebra* telah selesai disederhanakan. Penerapan metode Quine-McCluskey pada program bahasa C memanfaatkan beberapa *looping* untuk menyederhanakan fungsi *boolean algebra*. Algoritma *logic minimization* sangat mementingkan kompleksitas waktu dan ruang terutama jika variable fungsi logika yang diberikan cukup besar.

Proses minimisasi yang dilakukan oleh program yang telah kami buat ini dapat disimpulkan bahwa telah berhasil dilakukan karena fungsi *boolean algebra* yang rumit dapat disederhanakan menjadi suatu fungsi yang paling sederhana mengikuti aturan *(SOP)*. Jika dibandingkan secara manual, penyederhanaan fungsi *boolean algebra* dengan masukan tertentu membutuhkan waktu lebih lama karena banyaknya proses reduksi hingga memeroleh kondisi *prime implicant*. Namun, dengan adanya program minimisasi fungsi logika, penyederhanaan fungsi logika dapat dilakukan dalam waktu yang relatif singkat (dengan syarat variabel masukkan tidak lebih dari 10 variabel). Selain itu, dengan adanya program minimisasi ini, pengguna juga dapat menghemat waktu dan juga biaya pembuatan sirkuit logika dengan baik. Sehingga secara umum, program *logic minimization* yang telah dibuat oleh kelompok kami sudah terealisasi dengan baik sesuai dengan spesifikasi.

9. PEMBAGIAN TUGAS DALAM KELOMPOK

Berikut ini merupakan tabel pembagian tugas dari kelompok kami dalam pengerjaan tugas besar pada Mata Kuliah Pemecahan Masalah Dalam Bahasa C.

Anggota	Kontribusi				
	Pembuatan File Presentasi	Pembuatan File Laporan	Pembuatan Flowchart	Pembuatan Data Flow Diagram	Pembuatan Source Code
Farhan Hakim Iskandar (13220007)	Berkontribusi dalam pembuatan file presentasi secara keseluruhan	Berkontribusi dalam pembuatan file laporan dalam format IEEE	-	-	Berkontribusi dalam pencarian referensi algoritma minimisasi logic
Fitra Nurindra (13220011)	Berkontribusi dalam pembuatan presentasi terutama pada bagian demonstrasi dan perbaikan file	Berkontribusi dalam pembuatan file laporan secara keseluruhan baik dalam format biasa maupun dalam format IEEE	-	-	Berkontribusi dalam pencarian referensi algoritma minimisasi logic dan juga dalam pembuatan source code program secara keseluruhan

Muhammad Daffa Daniswara (13220043)	-	Berkontribusi dalam pembuatan laporan terutama pada bagian flowchart dan DFD	Berkontribusi dalam pembuatan flowchart secara keseluruhan	Berkontribusi dalam pembuatan DFD secara keseluruhan	Berkontribusi dalam pencarian referensi algoritma minimisasi logic

Tabel 9.1. Tabel Pembagian Kontribusi Kelompok

DAFTAR PUSTAKA

[1] https://github.com/Es1chUbJyan9/32bit_Quine-McCluskey_and_Petrick_Method_in_C. Diakses pada 16 April 2022.

[2] https://www.geeksforgeeks.org/logic-functions-and-minimization/. Diakses pada 28 April 2022.

[3] https://www.sciencedirect.com/topics/computer-science/logic-minimization. Diakses pada 1 Mei 2022.

[4] https://www.geeksforgeeks.org/quine-mccluskey-method/. Diakses pada 5 Mei 2022.

[5] https://www.geeksforgeeks.org/minimization-of-boolean-functions/. Diakses pada 8 Mei 2022.

[6] https://www.charlie-coleman.com/experiments/kmap/. Diakses pada 19 Mei 2022.

[7] Stephen Brown and Zvonko Vranesic, Fundamentals of Digital Logic with VHDL Design, McGraw-Hill Ed., New York-Amerika, 2009.

[8] Dan, Rotar. (2010). Software For The Minimization Of The Combinational Logic Functions. https://www.incdmtm.ro/editura/document e/pag.%2095-99.%20Software%20for%20The%20Minimization%20of%20The%20Combinational%20Logic%20Functions.pdf.