JUMANTAKA
Halaman Jurnal: http://jurnal.stmik-dci.ac.id/index.php/jumantaka/
Halaman LPPM STMIK DCI: http://lppm.stmik-dci.ac.id/

PENYEDERHAAN FUNGSI BOOLEAN DENGAN METODE QUINE MC-CLUSKEY

Ismayanto[1] , Andri Sukmaindrayana[2],
[1]Mahasiswa, Teknik Informatika STMIK DCI
puckhong@gmail.com
[2]Dosen, Teknik Informatika STMIK DCI
sukmaindrayana@gmail.com

ABSTRAK

Aljabar merupakan salah satu cabang matematika yang mempelajari tentang pemecahan masalah menggunakan simbol-simbol sebagai pengganti konstanta dan variabel. Sedangkan variabel adalah simbol pengganti suatu bilangan yang belum diketahui nilainya secara jelas.

Boolean adalah suatu tipe data yang hanya mempunyai dua nilai, yaitu *true* atau *false* (benar atau salah). Pada beberapa bahasa pemrograman, nilai *true* bisa digantikan 1 dan nilai *false* digantikan 0.

Quine-McCluskey adalah sebuah metode yang digunakan untuk menyederhanakan fungsi Boolean, khususnya fungsi Boolean yang memiliki jumlah peubah yang besar (di atas 6 buah). Metode ini mengubah sebuah fungsi Boolean menjadi sebuah himpunan bentuk prima, dimana sebanyak mungkin peubah dieliminasi (dihilangkan) secara maksimal, hingga didapat fungsi Boolean yang paling sederhana.

Kata Kunci : Aljabar, Boolean, *Quine-McCluskey*

I. PENDAHULUAN

Fungsi Boolean seringkali mengandung operasi – operasi yang tidak perlu, literal atau suku – suku yang berlebihan. Oleh karena itu, kita dapat menyederhanakan fungsi Boolean lebih lanjut. Menyederhanakan fungsi Boolean artinya mencari bentuk fungsi lain yang ekivalen tetapi dengan jumlah literal atau operasi yang lebih sedikit. Penyederhanaan fungsi Boolean disebut juga minimisasi fungsi. Dipandang dari segi aplikasi aljabar Boolean, fungsi Boolean yang lebih sederhana berarti rangkaian logikanya juga lebih sederhana (menggunakan jumlah gerbang logika lebih sedikit).

Salah satu metode yang dapat digunakan untuk menyederhakan fungsi Boolean adalah metode Quine-McCluskey (metode tabulasi). Metode peta Karnaugh hanya cocok digunakan jika fungsi Boolean mempunyai jumlah peubah paling banyak 6 buah. Jika jumlah peubah yang terlibat pada suatu fungsi Boolean lebih dari 6 buah maka penggunaan peta Karnaugh menjadi semakin rumit, sebab ukuran peta semakin besar. Selain itu, metode peta Karnaugh lebih sulit diprogram dengan komputer karena diperlukan pengamatan visual untuk mengidentifikasi *minterm* – *minterm* yang akan dikelompokkan. Untuk itu diperlukan metode penyederhanaan yang lain yang dapat diprogram dan dapat

digunakan untuk fungsi Boolean dengan sembarang jumlah peubah. Metode alternatif tersebut adalah metode Quine-McCluskey yang dikembangkan oleh W.V. Quine dan E.J. McCluskey pada tahun 1950.

Berdasarkan uraian di atas, penulis bermaksud merancang suatu perangkat lunak bantu pemahaman yang mampu menunjukkan tahapan – tahapan minimisasi fungsi Boolean dengan metode *Quine-McCluskey*.

II. LANDASAN TEORI

2.1 Definisi Aljabar Boolean

Menurut Lipschutz, Seymour & Marc Lars Lipson dalam bukunya '*2000 Solved Problems in Discrete Mathematics*', McGraw-Hill, 1992 Misalkan *B* adalah himpunan yang didefinisikan pada dua operator biner, + dan ., dan sebuah operator uner,'. Misalkan 0 dan 1 adalah dua elemen yang berbeda dari *B*. Maka, tupel <*B*, +, ., ', 0, 1> disebut aljabar Boolean jika untuk setiap *a*, *b*, *c* $\in$ *B* berlaku aksioma (sering dinamakan juga Postulat Huntington) berikut :

1. Identitas
   i. $a + 0 = a$
   ii. $a \cdot 1 = a$
2. Komutatif
   i. $a + b = b + a$
   ii. $a \cdot b = b \cdot a$
3. Distributif
   i. $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$
   ii. $a + (b \cdot c) = (a + b) \cdot (a + c)$
4. Komplemen
   Untuk setiap $a \in B$ terdapat elemen unik $a' \in B$ sehingga
   i. $a + a' = 1$
   ii. $a \cdot a' = 0$

Elemen 0 dan 1 adalah dua elemen unik yang berada di dalam *B*. 0 disebut elemen terkecil dan 1 disebut elemen terbesar. Kedua elemen unik dapat berbeda – beda pada beberapa aljabar Boolean (misalnya ι dan *U* pada himpunan, *False* dan *True* pada proposisi), namun secara umum kita tetap menggunakan 0 dan 1 sebagai dua elemen unik yang berbeda. Elemen 0 disebut elemen *zero*, sedangkan elemen 1 disebut elemen *unit*. Operator + disebut operator penjumlahan, . disebut operator perkalian, dan ' disebut operator komplemen.

Terdapat perbedaan antara aljabar Boolean dengan aljabar biasa untuk aritmetika bilangan riil :

1. Hukum distributif yang pertama, $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$ sudah dikenal di dalam aljabar biasa, tetapi hukum distributif yang kedua, $a + (b \cdot c) = (a + b) \cdot (a + c)$, benar untuk aljabar Boolean, tetapi tidak benar untuk aljabar biasa.
2. Aljabar Boolean tidak memiliki kebalikan perkalian (*multiplicative inverse*) dan kebalikan penjumlahan; karena itu, tidak ada operasi pembagian dan pengurangan di dalam aljabar Boolean.
3. Aksioma nomor 4 pada definisi 2.1 mendefinisikan operator yang dinamakan *komplemen* yang tidak tersedia pada aljabar biasa.
4. Aljabar biasa memperlakukan himpunan bilangan riil dengan elemen yang tidak berhingga banyaknya. Sedangkan aljabar Boolean memperlakukan himpunan elemen *B* yang sampai sekarang belum didefinisikan, tetapi pada aljabar Boolean dua-nilai, *B* didefinisikan sebagai himpunan dengan hanya dua nilai, 0 dan 1.

2.2 Fungsi Boolean

Fungsi Boolean (disebut juga fungsi biner) adalah pemetaan dari Bn 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*.

Misalkan ekspresi Boolean dengan *n* peubah adalah $E(x_1, x_2, ..., x_n)$. Menurut definisi di atas, setiap pemberian nilai – nilai kepada peubah $x_1, x_2, ..., x_n$ merupakan suatu pasangan terurut ganda-*n* di dalam daerah asal B^n dan nilai ekspresi tersebut adalah bayangannya di dalam daerah hasil *B*. Dengan kata lain, 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}. Contoh pasangan terurut ganda-3 misalnya (1, 0, 1) yang berarti $x = 1$, $y = 0$, dan $z = 1$ sehingga $f(1, 0, 1) = 1 . 0 . 1 + 1' . 0 + 0' . 1 = 0 + 0 + 1 = 1$.

2.3 Komplemen Fungsi Boolean

Bila sebuah fungsi Boolean dikomplemenkan, kita memperoleh fungsi komplemen. Fungsi komplemen berguna pada saat kita melakukan penyederhanaan fungsi Boolean. Fungsi komplemen dari suatu fungsi *f*, yaitu *f* ' dapat dicari dengan dua cara berikut :

1. Cara pertama : menggunakan hukum De Morgan

   Hukum De Morgan untuk dua buah peubah, x_1 dan x_2 adalah

   (i) $(x_1 + x_2)' = x_1'x_2'$

   (ii) dan dualnya : $(x_1 . x_2)' = x_1' + x_2'$

   Hukum De Morgan untuk tiga buah peubah, x_1, x_2 dan x_3 adalah

   (i) $(x_1 + x_2 + x_3)' = (x_1 + y')$, yang dalam hal ini $y = x_2 + x_3$

   $= x_1'y'$

   $= x1'(x_2 + x_3)'$

   $= x_1'x_2'x_3'$

   (ii) dan dualnya : $(x_1 . x_2 . x_3)' = x_1' + x_2' + x_3'$

   Hukum De Morgan untuk *n* buah peubah, $x_1, x_2, ... ,x_n$, adalah

   (iii) $(x_1 + x_2 + ... + x_n)' = x_1' x_2' ... x_n'$

   (iv) dan dualnya : $(x_1 . x_2 . ... . x_n)' = x_1' + x_2' + ... + x_n'$

2. Cara kedua : menggunakan prinsip dualitas.

   Tentukan dual dari ekspresi Boolean yang merepresentasikan *f*, lalu komplemenkan setiap literal di dalam dual tersebut. Bentuk akhir yang diperoleh menyatakan fungsi komplemen. Sebagai contoh, Misalkan $f(x, y, z) = x(y'z' + yz)$, maka dual dari ekspresi Booleannya adalah

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

   Komplemenkan tiap literal dari dual di atas menjadi

   $x' + (y + z) (y' + z')$ = *f* '. Jadi, $f'(x, y, z) = x' + (y + z) (y' + z')$.

2.4 Metode *Quine Mc-Cluskey*

Metode peta Karnaugh hanya cocok digunakan jika fungsi Boolean mempunyai jumlah peubah yang tidak besar. Jika peubah yang terlibat pada suatu fungsi Boolean dalam jumlah yang besar maka penggunaan peta Karnaugh menjadi semakin rumit, sebab ukuran peta bertambah besar. Selain itu, metode peta Karnaugh lebih sulit diprogram dengan komputer karena diperlukan pengamatan visual untuk mengidentifikasi *minterm* – *minterm* yang akan dikelompokkan. Untuk itu diperlukan metode penyederhanaan yang lain yang dapat diprogram dan dapat digunakan untuk fungsi Boolean dengan sembarang jumlah peubah. Metode alternatif tersebut adalah metode Quine-McCluskey.

Metode Quine-McCluskey adalah sebuah metode yang digunakan untuk menyederhanakan fungsi Boolean,

khususnya fungsi Boolean yang memiliki jumlah peubah yang besar (di atas 6 buah). Metode Quine-McCluskey dikembangkan oleh W.V. Quine dan E.J. McCluskey pada tahun 1950.

Metode ini mengubah sebuah fungsi Boolean menjadi sebuah himpunan bentuk prima, dimana sebanyak mungkin peubah dieliminasi (dihilangkan) secara maksimal, hingga didapat fungsi Boolean yang paling sederhana.

III. ANALISIS SISTEM

3.1 Pembahasan

Proses penyederhanaan fungsi Boolean dengan metode Quine-McCluskey mempunyai 7 (tujuh) langkah pengerjaan untuk menyederhanakan fungsi Boolean dalam bentuk SOP (*sum-of-product*) atau bentuk POS (*product-of-sum*), seperti yang telah dijelaskan pada bab 2 sebelumnya.

Metode Quine-McCluskey menyederhanakan fungsi Boolean dengan menggabungkan *minterm / maxterm* menjadi himpunan bentuk prima (*prime-implicant*), dimana sebanyak mungkin peubah dieliminasi (dihilangkan) secara maksimal. *Minterm / maxterm* yang digabung untuk membentuk sebuah bentuk prima harus memiliki tepat satu buah peubah yang nilainya berbeda (komplemen dan non-komplemen). Bentuk prima yang terbentuk juga dapat digabung untuk membentuk bentuk prima lainnya, apabila memiliki satu buah peubah yang nilainya berbeda. Prosedur ini dilakukan berulang kali hingga tidak terdapat bentuk prima yang dapat terbentuk lagi.

Sebagai contoh, dilakukan penyederhanaan terhadap fungsi Boolean dalam bentuk SOP : $f(w, x, y, z) = \sum$ (0, 1, 3, 4, 5, 6, 8, 10, 11, 15). Langkah – langkah penyederhanaan fungsi Boolean dengan metode Quine-McCluskey adalah sebagai berikut:

LANGKAH-1 : Nyatakan tiap *minterm* dalam n peubah menjadi string bit biner yang panjangnya n. (Pada contoh ini, jumlah peubah adalah 4 sehingga n = 4)

0 = 0000
1 = 0001
3 = 0011
4 = 0100
5 = 0101
6 = 0110
8 = 1000
10 = 1010
11 = 1011
15 = 1111

LANGKAH-2 : Kelompokkan tiap *minterm* berdasarkan jumlah '1' yang dimilikinya.

Hasil Penyelesaian Langkah - 2 :

```
))))))))))))))))
 term   w x y z
))))))))))))))))
 0      0 0 0 0  -> jumlah bit '1' = 0 buah
))))))))))))))))
 1      0 0 0 1  -> jumlah bit '1' = 1 buah
 4      0 1 0 0
 8      1 0 0 0
))))))))))))))))
 5      0 1 0 1  -> jumlah bit '1' = 2 buah
 6      0 1 1 0
 3      0 0 1 1
 10     1 0 1 0
))))))))))))))))
 11     1 0 1 1  -> jumlah bit '1' = 3 buah
))))))))))))))))
 15     1 1 1 1  -> jumlah bit '1' = 4 buah
))))))))))))))))
```

LANGKAH-3 : Kombinasikan *minterm* dalam n peubah dengan kelompok lain yang jumlah '1'-nya berbeda satu, sehingga diperoleh bentuk prima (*prime-implicant*) yang terdiri dari n-1 peubah. *minterm* yang dikombinasikan diberi tanda 'v'

LANGKAH-4 : Kombinasikan *minterm* dalam n - 1 peubah dengan kelompok lain yang jumlah '1'-nya berbeda satu, sehingga diperoleh bentuk prima yang terdiri dari n-2 peubah.

LANGKAH-5 : Teruskan langkah 4 sampai diperoleh bentuk prima yang sesederhana mungkin.

Hasil Penyelesaian Langkah-3, 4 dan 5 :

term	w x y z	
0	0 0 0 0	v
1	0 0 0 1	v
4	0 1 0 0	v
8	1 0 0 0	v
5	0 1 0 1	v
6	0 1 1 0	v
3	0 0 1 1	v
10	1 0 1 0	v
11	1 0 1 1	v
15	1 1 1 1	v

Dikombinasikan menjadi :

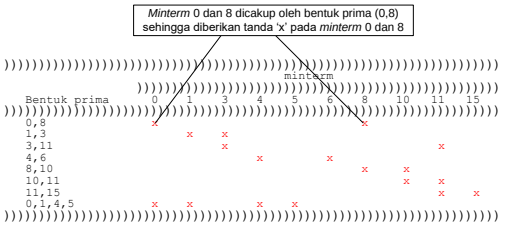
term	w x y z	
0,1	0 0 0 -	v
0,4	0 - 0 0	v
0,8	- 0 0 0	
1,3	0 0 - 1	
1,5	0 - 0 1	v
4,5	0 1 0 -	v
4,6	0 1 - 0	
8,10	1 0 - 0	
3,11	- 0 1 1	
10,11	1 0 1 -	
11,15	1 - 1 1	

Dikombinasikan menjadi :

term	w x y z
0,1,4,5	0 - 0 -
0,4,1,5	0 - 0 -

LANGKAH-6 : Ambil semua bentuk prima yang tidak bertanda 'v'. Buatlah tabel baru yang memperlihatkan *minterm* dari ekspresi Boolean semula yang dicakup oleh bentuk prima tersebut (tandai dengan 'x').

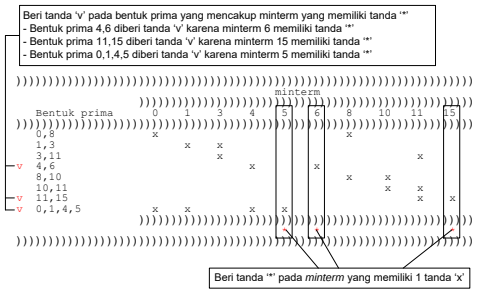
Hasil Penyelesaian Langkah - 6 :



LANGKAH-7 : Pilih bentuk prima yang memiliki jumlah literal paling sedikit namun mencakup sebanyak mungkin minterm dari ekspresi Boolean semula. Hal ini dapat dilakukan dengan cara berikut :

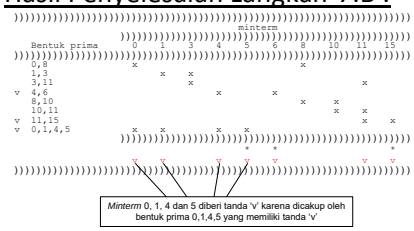
LANGKAH-7.A : Tandai kolom-kolom yang mempunyai satu buah tanda 'x' dengan tanda '*', lalu beri tanda 'v' di sebelah kiri bentuk prima yang mencakup minterm yang mempunyai tanda '*' tersebut. Bentuk prima ini telah dipilih untuk fungsi Boolean sederhana.

Hasil Penyelesaian Langkah-7 dan 7.A :



LANGKAH-7.B : Untuk setiap bentuk prima yang telah ditandai dengan 'v', beri tanda *minterm* yang dicakup oleh bentuk prima tersebut dengan tanda 'v' (di baris bawah setelah tanda '*').

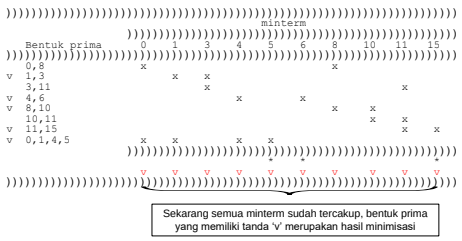
Hasil Penyelesaian Langkah-7.B :



LANGKAH-7.C : Periksa apakah masih ada minterm yang belum memiliki tanda 'v' (artinya, belum dicakup oleh bentuk prima terpilih). Jika ada, pilih dari bentuk prima yang tersisa yang mencakup sebanyak mungkin minterm tersebut. Beri tanda 'v' untuk setiap bentuk prima yang dipilih itu serta minterm yang dicakupnya.

LANGKAH-7.D : Ulangi langkah 7.C sampai seluruh minterm sudah dicakup oleh bentuk prima

Hasil Penyelesaian Langkah - 7.C dan 7.D :
Sampai tahap ini, masih ada *minterm* yang belum tercakup dalam bentuk prima terpilih, yaitu 3, 8, 10. Untuk mencakup *minterm* tersebut, kita pilih bentuk prima (8,10) dan (1,3) karena mencakup *minterm* 3, 8 dan 10 sekaligus.



Sekarang, semua minterm sudah tercakup dalam bentuk prima terpilih. Bentuk prima yang terpilih adalah :

1, 3 yang bersesuaian dengan *term* *w'x'z* (*minterm* 1 (0001) dan *minterm* 3 (0011) memiliki persamaan bit pada posisi 1, 2 dan 4 sehingga dapat dikatakan *minterm* 1 dan *minterm* 3 merupakan fungsi Boolean *w'x'z*).

4, 6 yang bersesuaian dengan *term* *w'xz'*.

8, 10 yang bersesuaian dengan *term* *wx'z'*.

11, 15 yang bersesuaian dengan *term* *wyz.*

0, 1, 4, 5 yang bersesuaian dengan *term* *w'y'*.

IV. PERANCANGAN SISTEM

4.1 Pembahasan

Perangkat lunak bantu pemahaman ini dirancang dengan menggunakan bahasa pemrograman *Microsoft Visual Basic 6.0* dengan beberapa komponen standar seperti *Command Button, Text Box, Rich Text Box, Option Button, Label, Shape*, dan sebagainya.

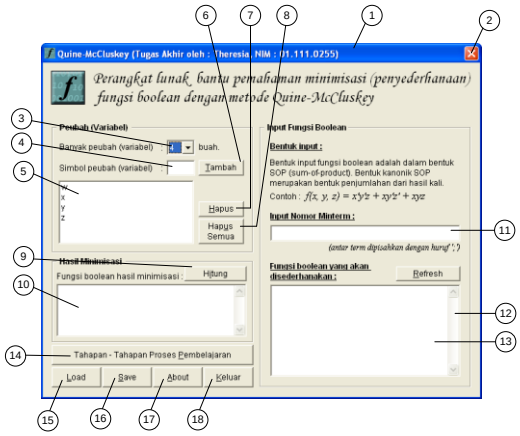
Perangkat lunak bantu pemahaman ini memiliki beberapa *form*, antara lain :

1. *Form Input*.
2. *Form* Pembelajaran Quine-McCluskey.
3. *Form* Langkah – Langkah Penyederhanaan dalam Format *Text*.
4. *Form About*.

4.1.1 *Form Input*

Form Input berfungsi sebagai *form* untuk memasukkan *input* terhadap perangkat lunak, seperti: banyak peubah,

simbol peubah, bentuk fungsi dan *input term* pada fungsi Boolean. *Input* fungsi Boolean dapat disimpan (*save*) dalam bentuk *file* (*exstension* QMC) dan dibuka (*load*) kembali. Pada *form* ini, juga terdapat fasilitas untuk menghitung dan mendapatkan hasil minimisasi fungsi Boolean secara langsung (tanpa melihat langkah – langkah pemahaman).



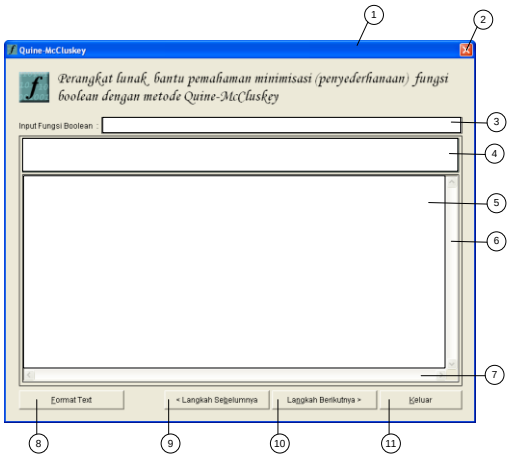
Gambar 4.1
Rancangan *Form Input*

Keterangan :

1. *title bar*, berisikan tulisan 'Quine-McCluskey (Tugas Akhir oleh : Theresia, NIM : 31.111.0255)'.
2. Tombol 'Close', berfungsi untuk menutup *Form Input*.
3. *combobox*, berfungsi sebagai tempat memilih banyak peubah (variabel).
4. *textbox* 'Simbol peubah', berfungsi sebagai tempat pengisian simbol peubah.
5. *listbox*, sebagai tempat menampilkan simbol peubah yang telah di-*input*.
6. Tombol 'Tambah' untuk menambahkan simbol peubah yang diisikan pada daerah-4 ke *listbox* pada daerah-5.
7. Tombol 'Hapus' untuk menghapus simbol peubah yang dipilih pada daerah-5.
8. Tombol 'Hapus Semua' untuk menghapus semua simbol peubah pada daerah-5.
9. Tombol 'Hitung' untuk mendapatkan hasil minimisasi secara langsung dan ditampilkan di *textbox* pada daerah 10.
10. *textbox* 'Hasil Minimisasi' sebagai tempat menampilkan hasil minimisasi.
11. *textbox* 'Input Nomor Minterm', sebagai tempat pengisian nomor-nomor *term* pada fungsi Boolean.
12. Tombol 'Refresh' untuk mengubah bentuk *minterm* yang telah di-input ke bentuk fungsi Boolean.
13. *textbox* 'Fungsi Boolean', sebagai tempat untuk menampilkan fungsi Boolean.
14. Tombol 'Tahapan – Tahapan Proses Pembelajaran' untuk menampilkan *For* Pembelajaran Quine-McCluskey.
15. Tombol 'Load' untuk membuka *file input* (*.QMC) yang telah disimpan sebelumnya.
16. Tombol 'Save' untuk menyimpan *input* fungsi ke dalam bentuk file (*.QMC).
17. Tombol 'About' untuk menampilkan *Form About.*
18. Tombol 'Keluar' untuk keluar dari perangkat lunak.

4.1.2 *Form* Pembelajaran *Quine Mc-Cluskey*

Form Pembelajaran Quine-McCluskey berfungsi sebagai *form* untuk menampilkan langkah-langkah minimisasi fungsi yang telah di-*input* dengan metode Quine-McCluskey. Pada *form* ini, setiap langkah minimisasi dapat ditelusuri langkah per langkah hingga didapat hasil minimisasi.



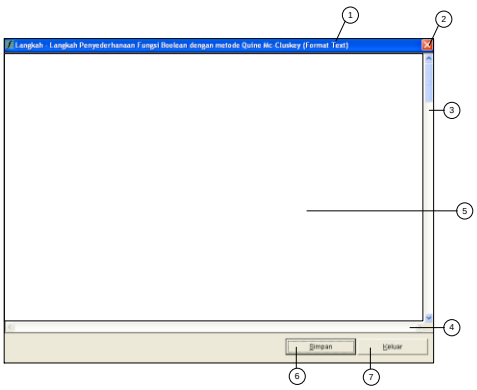
Gambar 4.2
Rancangan *Form* Pembelajaran *Quine Mc-Cluskey*

Keterangan :

1. *Title bar*, berisikan tulisan 'Quine-McCluskey'.
2. Tombol 'Close', berfungsi untuk menutup *Form* Pembelajaran Quine-McCluskey.
3. Label, untuk menampilkan *input* fungsi Boolean.
4. T*extbox*, untuk menampilkan langkah-langkah yang dilakukan.
5. T*extbox*, untuk menampilkan hasil eksekusi langkah-langkah yang dilakukan.
6. *Vertical scrollbar*, untuk menggulung *textbox* pada daerah-5 secara vertikal.
7. *Horizontal scrollbar*, untuk menggulung *textbox* pada daerah-5 secara horizontal.
8. Tombol 'Format Text', untuk menampilkan *form* langkah-langkah penyederhanaan dalam bentuk *text*.
9. Tombol 'Langkah Sebelumnya', untuk menampilkan langkah sebelumnya.
10. Tombol 'Langkah Berikutnya', untuk menampilkan langkah berikutnya.
11. Tombol 'Keluar', untuk keluar dari *form* dan kembali ke *Form Input*.

4.1.3 *Form* Langkah – Langkah Penyederhanaan dalam Format Text

Apabila langkah – langkah minimisasi pada *form* pembelajaran dijabarkan langkah per langkah, maka pada *form* ini ditampilkan semua hasil eksekusi langkah secara bersamaan (sekaligus). Hal ini dimaksudkan agar *user* dapat menyimpan langkah - langkah penyederhanaan dalam bentuk *rich text file* (*.rtf). *File* hasil penyimpanan dapat digunakan sebagai alat pendukung dalam kegiatan belajar mengajar, karena *file* dapat dibuka dan dicetak pada aplikasi *Microsoft Word*.



Gambar 4.3
Rancangan *Form Langkah – Langkah Penyederhanaan dalam Format Text*

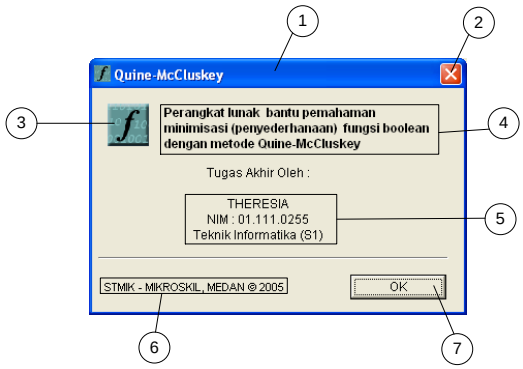
Keterangan :

1. *Title bar*, berisikan tulisan 'Langkah – Langkah Penyederhanaan Fungsi Boolean dengan metode Quine Mc-Cluskey (Format Text)'.
2. Tombol 'Close', berfungsi untuk menutup *form*.
3. *Vertical scrollbar*, berfungsi untuk menggulung *textbox* pada daerah-5 secara vertikal.
4. *Horizontal scrollbar*, berfungsi untuk menggulung *textbox* pada daerah-5 secara horizontal.

5. *Textbox*, berfungsi sebagai tempat menampilkan langkah-langkah penyederhanaan fungsi Boolean.
6. Tombol 'Simpan', berfungsi untuk menyimpan langkah – langkah penyederhanaan dalam format *rich text file* (*.rtf).
7. Tombol 'Keluar', berfungsi untuk keluar dari *form*.

4.1.4 *Form About*

Form About berfungsi sebagai *form* untuk menampilkan informasi mengenai pembuat perangkat lunak.



Gambar 4.4
Rancangan *Form About*

Keterangan :

1. *Title bar*, berisikan tulisan 'Quine-McCluskey'.
2. Tombol 'Close', berfungsi untuk menutup *form*.
3. Logo atau gambar *icon* perangkat lunak.
4. Nama perangkat lunak.
5. Identitas pembuat perangkat lunak.
6. Nama kampus, kota dan tahun pembuatan perangkat lunak.
7. Tombol 'OK' untuk menutup *Form About*.

V. IMPLEMENTASI SISTEM

5.1 Spesifikasi Perangkat Keras dan Perangkat Lunak

Program ini direkomendasikan untuk dijalankan dengan menggunakan perangkat keras (*hardware*) yang mempunyai spesifikasi berikut :

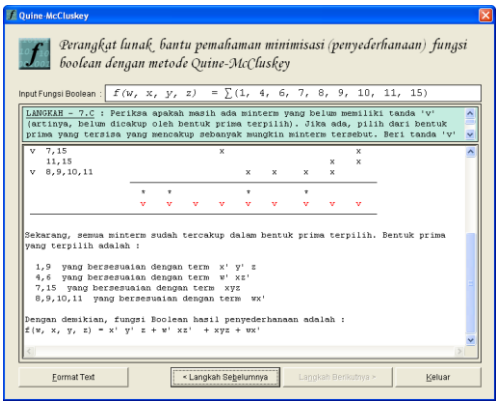
1. Prosesor Intel Pentium IV 1,6 Ghz.
2. Memory 128 MB.
3. Harddisk 10 GB.
4. VGA *card* 64 MB.
5. Monitor dengan resolusi 800 × 600 *pixel*.
6. *Keyboard* dan *Mouse*.

Adapun perangkat lunak (*software*) yang digunakan untuk menjalankan aplikasi ini adalah lingkungan sistem operasi *MS-Windows*98 atau *MS-Windows* NT/2000/XP.

5.2 Pengujian Program

Sebagai contoh, penulis meng-*input* data sebagai berikut.

1. Bentuk fungsi *input* : SOP (*sum-of-product*)
2. Banyak peubah (variabel) = 4 buah, yaitu *w*, *x*, *y* dan *z*.
3. *Input* nomor *minterm* : 1;4;6;7;8;9;10;11;15.
4. Hasil penyederhanaan, didapat : $f(w, x, y, z) = x'y'z + w'xz' + xyz + wx'$.



Gambar 5.1
Proses penyederhanaan fungsi Boolean (contoh-1)

VI. KESIMPULAN DAN SARAN

6.1 Kesimpulan

Setelah menyelesaikan perancangan perangkat lunak bantu pemahaman minimisasi fungsi Boolean dengan metode Quine-McCluskey ini, penulis menarik kesimpulan sebagai berikut :

1. Perangkat lunak mampu menyederhanakan fungsi Boolean dalam bentuk SOP (*sum-of-product*) dan POS (*product-of-sum*).
2. Perangkat lunak mampu menyederhanakan fungsi Boolean dengan jumlah peubah yang besar (maksimum 10 buah).
3. Perangkat lunak menunjukkan setiap langkah atau tahapan dalam proses penyederhanaan fungsi Boolean dengan metode Quine-McCluskey.

6.2 Saran

Penulis ingin memberikan beberapa saran yang mungkin dapat membantu dalam pengembangan perangkat lunak bantu pemahaman ini yaitu :

1. Perangkat lunak dapat dikembangkan dengan menerima *input* berupa fungsi Boolean yang langsung diisi oleh *user*.
2. Pemahaman proses penyederhanaan dapat dikembangkan dengan menambahkan animasi proses penjelasan yang dibangun dengan aplikasi *Macromedia Flash*.

DAFTAR PUSTAKA

Ario Suryokusumo, ***Microsoft Visual Basic 6.0***, PT. Elex Media Komputindo, 2001.

Djoko Pramono, **Mudah menguasai *Visual Basic 6***, PT. Elex Media Komputindo, 2002.

http://134.193.15.25/vu/course/cs281/lectures/simplification/quine-McCluskey.html(diakses pada tanggal 2 Februari 2017)

http://www.seattlerobotics.org/encode/200106/qmccmin.htm#1.3(diakses pada tanggal 15 januari 2017)

Informatika Bandung, **Pengantar Logika Matematika,** 2004.

Retno Hendrawati,IR,MT & Bambang Hariyanto, **Logika Matematika,** Informatika Bandung, 2000.

Rinaldi Munir, *Matematika Diskrit*, Informatika Bandung, 2005.

Wikipedia, Bolean Algebra (http://en.wikipedia.org/wiki/Boolean_algebra)(diakses pada tanggal 20 Desember 2016)

Yulianeu A, 2016, Sistem Berkas, LPPM STMIK DCI, Tasikmalaya.