BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Algoritma

Algoritma pemrograman dalam perangkat lunak minimisasi fungsi boolean dengan metode Quine-McCluskey ini dibagi menjadi 3 bagian yaitu :

1. Algoritma Data *Input*.
2. Algoritma Penghasil Tahapan – Tahapan Penyederhanaan Fungsi Boolean dengan metode Quine-McCluskey.
3. Algoritma Fungsi Pendukung.

4.1.1 Algoritma Data Input

Algoritma ini dirancang dalam bentuk fungsi (*function*) dengan nama 'CekInput'. Apabila fungsi ini mengembalikan nilai *True*, maka *input* data *valid* dan proses akan dilanjutkan ke tahapan penyederhanaan. Apabila fungsi mengembalikan nilai *False*, maka fungsi akan memunculkan pesan kesalahan berkaitan dengan kesalahan yang telah dibuat dan *user* harus mengoreksi kesalahan untuk dapat melanjutkan ke proses berikutnya.

Berikut merupakan algoritma fungsi 'CekInput' yang mengembalikan nilai *boolean* (*true* / *false*),

1. Set CekInput = *True*.
2. Penggal *input* nomor *term* berdasarkan huruf ';' dan simpan hasilnya pada variabel *array* TTerm.

3. Jika jumlah *array* pada TTerm < 0, maka munculkan pesan kesalahan ‘Term belum diisi !’, set CekInput = *False* dan keluar dari fungsi. Jika tidak, maka lanjutkan ke algoritma berikutnya.
4. Selanjutnya untuk memeriksa apakah terdapat nomor *term* yang melebihi batas maksimal nomor *term* yaitu (2 ^ banyak_peubah) – 1, lakukan prosedur berikut,
    a. Set Max = (2 ^ banyak_peubah) – 1.
    b. Untuk T = 0 sampai jumlah *array* TTerm, lakukan prosedur berikut,
        i. Set TTerm(T) = Val(TTerm(T)).
        ii. Jika Val(TTerm(T)) < 0 atau Val(TTerm(T)) > Max, maka munculkan pesan kesalahan ‘Input term di luar batas’, set *CekInput* = False dan keluar dari fungsi.
5. Untuk memeriksa apakah terdapat nomor *term* yang sama, lakukan prosedur berikut. Untuk S = 0 sampai jumlah *array* TTerm, lakukan langkah berikut,
    a. Untuk T = 0 sampai jumlah *array* TTerm, cek Jika S <> T dan TTerm(S) = TTerm(T), maka munculkan pesan kesalahan ‘Ditemukan term ‘” & Tterm (S) & “’ lebih dari 1 buah’, set CekInput = False dan keluar dari fungsi.

4.1.2 Algoritma Penghasil Tahapan – Tahapan Penyederhanaan Fungsi Boolean dengan metode Quine-McCluskey

Algoritma ini merupakan inti dari perangkat lunak. Fungsi dari algoritma ini adalah melakukan proses penyederhanaan terhadap *input* fungsi Boolean dengan metode Quine-McCluskey dan sekaligus menghasilkan tahapan – tahapan penyederhanaan dalam format *text*. Tahapan – tahapan penyederhanaan ini dimaksudkan sebagai fasilitas pembelajaran bagi *user*. Tahapan penyederhanaan ini juga dapat disimpan (format *rich text file* (*.rtf)), dibuka dengan aplikasi *Word Processing* seperti *Microsoft Word* atau *WordPad*, dan dicetak ke *printer*. Algoritma ini dirancang dalam bentuk fungsi dengan nama 'ProsesQuineMcCluskey'. Fungsi ini menghasilkan tahapan – tahapan penyederhanaan, menyimpannya ke variabel *array* cLangkah dan mengembalikan hasil minimisasi. Algoritma dari fungsi 'ProsesQuineMcCluskey' adalah sebagai berikut,

1. Untuk mengerjakan langkah-1 dari metode Quine McCluskey, lakukan langkah berikut. Untuk N1 = 0 sampai jumlah *array* InputTerm.
    a. Set MTerm(N1).Term = InputTerm(N1), set MTerm(N1).Bit = bilangan biner dari InputTerm(N1) dengan panjang bilangan sebesar banyak_peubah, set MTerm(N1).Level = 1, set MTerm(N1).Kelompok = banyak jumlah bit '1' dari bilangan biner, set MTerm(N1).Check = *False*.
    b. Set Pos(N1) = N1.

c. Untuk langkah penyelesaian, set cLangkah(1) = cLangkah(1) & vbCrlf & MTerm(N1).Term & " = " & MTerm(N1).Bit.

2. Untuk mengerjakan langkah-2 sampai langkah-5 dari metode Quine McCluskey, set nLevel1 = 1. Selama bSederhana = *False* (masih bisa dikelompokkan atau belum sederhana), lakukan langkah berikut.
   a. Set nLevel = nLevel + 1.
   b. Periksa apakah terdapat *term* yang memiliki perbedaan bit sebanyak 1 buah untuk *level* yang sama. Apabila ada, maka bentuk sebuah bentuk prima yang baru. Untuk N1 = 0 sampai jumlah *array* dari MTerm, lakukan prosedur berikut.
      i. Jika MTerm(N1).Level = nLevel maka lakukan *looping* dari N2 = N1 + 1 sampai jumlah array MTerm, cek jika MTerm(N2).Level = nLevel maka lakukan langkah berikut.
         1. Ambil banyak perbedaan bit, dengan men-set DF = GetDifference(MTerm(N1).Bit, MTerm(N2).Bit).
         2. Jika DF.Banyak = 1(jumlah bit yang berbeda sebanyak 1 buah), maka lanjutkan ke poin 3.
         3. Set MTerm(N1).Check = *True*, set MTerm(N2).Check = *True*.
         4. Set N3 = jumlah *array* MTerm + 1. Bentuk ulang dimensi *array* dari MTerm menjadi N3.

5. Set MTerm(N3).Term = MTerm(N1).Term & "," & MTerm(N2).Term. Ini adalah bentuk prima yang baru.
6. MTerm(N3).Bit = Left(MTerm(N1).Bit, DF.Posisi – 1) & "-" & Mid(MTerm(N1).Bit, DF.Posisi + 1, Len(MTerm(N1).Bit) – DF.Posisi).
7. Set MTerm(N3).Check = False
8. Set MTerm(N3).Level = nLevel + 1.
9. Set MTerm(N3).Kelompok = MTerm(N1).Kelompok.
10. Set nLevel1 = nLevel + 1

c. Set bSederhana = *True*.

d. Cek apakah masih bisa disederhanakan lagi. Untuk N1 = 0 sampai jumlah *array* dari MTerm, lakukan langkah berikut.

   i. Jika MTerm(N1).Level = nLevel + 1, maka lakukan *looping* dari N2 = N1 + 1 sampai jumlah *array* MTerm, lakukan langkah berikut.

      1. Jika MTerm(N2).Level = nLevel + 1, maka set DF = GetDifference(MTerm(N1).Bit, MTerm(N2).Bit).
      2. Jika DF.Banyak = 1 maka set bSederhana = False dan keluar dari *looping*.

   ii. Jika bSederhana = False, maka keluar dari *looping*.

3. Ambil semua bentuk prima yang tidak memiliki tanda *check* dan *update* ke bentuk prima (Langkah 6.1 dari metode Quine McCluskey). Algoritmanya adalah sebagai berikut,
    a. Set N5 = 0.
    b. Bentuk ulang dimensi *array* variabel Prima menjadi 0.
    c. Untuk N1 = 0 sampai jumlah *array* dari MTerm, lakukan langkah berikut,
        i. Jika MTerm(N1).Check = *False* maka lakukan langkah berikut,
            1. Urutkan *term* dengan men-set cTemp = Urutkan(MTerm(N1).Term)
            2. Cek apakah bentuk prima sudah pernah ada dengan set bSama = False, lakukan *looping* dari N2 = 1 sampai jumlah *array* dari variabel Prima, jika Prima(N2).Term = cTemp maka set bSama = *True* dan keluar dari *looping*.
            3. Jika bSama = False, maka lakukan langkah berikut,
                a. Set N3 = Ubound(Prima) + 1.
                b. Bentuk ulang dimensi *array* Prima menjadi N3.
                c. Set Prima(N3).Term = cTemp.
                d. Set Prima(N3).Check = False.
                e. Set Prima(N3).Nilai = 0.

f. Jika panjang dari Prima(N3).Term > N5 maka, set N5 = panjang dari Prima(N3).Term.

4. Beri tanda ‘x’ untuk setiap *term* yang tercakup di dalam bentuk prima yang terpilih. Algoritmanya adalah sebagai berikut,
    a. Bentuk ulang dimensi *array* variabel MTermP menjadi jumlah *array* dari variabel InputTerm.
    b. Untuk N1 = 0 sampai jumlah *array* variabel InputTerm, lakukan langkah berikut,
        i. Set MTermP(N1).Term = InputTerm(N1).
        ii. Set MTermP(N1).BanyakX = 0.
        iii. Set MTermP(N1).Check = *False*.
        iv. Set MTermP(N1).Bintang = *False*.
    c. Untuk N1 = 1 sampai jumlah *array* variabel Prima, lakukan langkah berikut,
        i. Untuk N2 = 0 sampai jumlah *array* variabel MTermP, cek jika MTermP(N2).Term terdapat di dalam list dari Prima(N1).Term) maka set MTermP(N2).BanyakX = MTermP(N2).BanyakX + 1.
5. Tandai ‘*’ untuk semua MTermP yang memiliki 1 buah tanda ‘x’ (Langkah 7.a.1 dari metode Quine McCluskey). Algoritmanya adalah sebagai berikut,

a. Untuk N1 = 0 sampai jumlah *array* dari variabel MTermP, cek jika MTermP(N1).BanyakX = 1 maka set MTermP(N1).Bintang = *True*.

6. Beri tanda *check* pada semua bentuk prima yang mencantumkan *term* yang memiliki tanda '*' (Langkah 7.a.2 dari metode Quine McCluskey). Algoritmanya adalah sebagai berikut,
    a. Untuk N1 = 0 sampai jumlah *array* dari variabel MTermP, lakukan langkah berikut,
        i. Jika MTermP(N1).Bintang = *True*, maka lakukan *looping* dari N2 = 1 sampai jumlah *array* dari variabel Prima, cek jika MTermP(N1).Term tercantum di dalam Prima(N2).Term, maka set Prima(N2).Check = *True*.
7. Beri tanda *check* di MTermP untuk semua MTermP yang tercantum dalam bentuk prima yang memiliki tanda *check* (Langkah 7.b dari metode Quine McCluskey). Algoritmanya adalah sebagai berikut,
    a. Untuk N1 = 0 sampai jumlah *array* dari variabel Prima, lakukan langkah berikut,
        i. Jika Prima(N1).Check = *True*, maka lakukan looping dari N2 = 0 sampai jumlah array variabel MTermP, cek jika MTermP(N2).Term tercantum di dalam Prima(N1).Term, maka set MTermP(N2).Check = *True*.
8. Periksa apakah masih ada MTermP yang belum ter-*check*, apabila ada, maka pilih bentuk prima yang mencakup paling banyak MTermP yang

belum ter-*check* (Langkah 7.c & 7.d dari metode Quine McCluskey). Algoritmanya adalah sebagai berikut,

a. Set bAllCheck = FAllCheck(MTermP).
b. Jika bAllCheck = *False*, maka lanjutkan algoritma pada poin c.
c. Set cTemp = “”.
d. Untuk N1 = 0 sampai jumlah *array* dari variabel MTermP, lakukan langkah berikut,
    i. Jika MTermP(N1).Check = *False*, maka
        1. Jika cTemp <> “” maka cTemp = cTemp & “, “.
        2. Set cTemp = cTemp & MTermP(N1).Term.
        3. Untuk N2 = 1 sampai jumlah *array* dari variabel Prima, cek jika MTermP(N1).Term tercantum di dalam Prima(N2).Term maka Prima(N2).Nilai = Prima(N2).Nilai + 1.
e. Bentuk ulang dimensi *array* variabel Pos menjadi jumlah *array* dari variabel Prima.
f. Untuk N1 = 1 sampai jumlah *array* dari variabel Pos, set Pos(N1) = N1.
g. Kemudian, lakukan pengurutan variabel Prima berdasarkan nilai yang dimiliki. Pengurutan yang dilakukan menggunakan metode *buble sort*.
    i. Untuk N1 = 1 sampai jumlah *array* dari variabel Pos – 1, lakukan langkah berikut.

ii. Untuk N2 = N1 + 1 sampai jumlah array dari variabel Pos, lakukan langkah berikut.

iii. Jika Prima(Pos(N1)).Nilai < Prima(Pos(N2)).Nilai, maka

1. Set N3 = Pos(N1).
2. Set Pos(N1) = Pos(N2).
3. Set Pos(N2) = N3.

iv. Jika Prima(Pos(N1)).Nilai = Prima(Pos(N2)).Nilai, maka pilih bentuk prima yang paling panjang.

1. Cek jika UBound(Split(Prima(Pos(N1)).Term, ",")) < UBound(Split(Prima(Pos(N2)).Term, ",")), maka tukar posisi.

a. Set N3 = Pos(N1).
b. Set Pos(N1) = Pos(N2).
c. Set Pos(N2) = N3.

h. Set cPrima = “”.

i. Set N1 = 0.

j. Selama bAllCheck = *False*, lakukan langkah berikut,

i. Set N1 = N1 + 1.

ii. Set Prima(Pos(N1)).Check = *True*.

iii. Jika cPrima <> “” maka set cPrima = cPrima & “, “.

iv. Untuk N2 = 0 sampai jumlah *array* dari variabel MTermP, lakukan langkah berikut,

1. Jika MTermP(N2).Check = False, maka cek jika MTermP(N2).Term tercantum dalam Prima(Pos(N1)).Term) maka set MTermP(N2).Check = *True*.

v. Set bAllCheck = FAllCheck(MTermP).

9. Set HasilMinimisasi = “”.
10. Ambil bentuk prima yang ter-*check* dan ubah menjadi bentuk kanonik. Algoritmanya adalah sebagai berikut,
    a. Untuk N1 = 1 sampai jumlah *array* dari variabel Prima, lakukan langkah pada poin b berikut.
    b. Jika Prima(N1).Check maka,
        i. Set Prima(N1).BentukKanonik = PrimaToKanonik(Prima(N1).Term).
        ii. Jika cBentuk = “SOP”, maka,
            1. Jika HasilMinimisasi <> “” maka set HasilMinimisasi = HasilMinimisasi & “ + “.
            2. Set HasilMinimisasi = HasilMinimisasi & Prima(N1).BentukKanonik.
        iii. Jika cBentuk = “POS”, maka,
            1. Set HasilMinimisasi = HasilMinimisasi & Prima(N1).BentukKanonik.
11. Kembalikan hasil minimisasi.
    a. Set ProsesQuineMcCluskey = HasilMinimisasi.

4.1.3 Algoritma Fungsi Pendukung

Fungsi-fungsi pendukung yang digunakan dalam proses penyederhanaan adalah sebagai berikut :

1. Fungsi ‘GetDifference(pBit1 As String, pBit2 As String)’ mengembalikan banyak dan posisi *bit* yang berbeda dari pBit1 dan pBit2.
    a. Untuk F1 = 1 sampai panjang dari variabel pBit1, lakukan langkah berikut,
        i. Jika Mid(pBit1,F1,1) <> Mid(pBit2, F1, 1) maka set nBeda = nBeda + 1, set nPosisi = F1.
    b. Set GetDifference.Banyak = nBeda.
    c. Set GetDifference.Posisi = nPosisi.
2. Fungsi ‘Urutkan (pcDeretAngka As String)’ mengembalikan deretan angka yang terurut. Pengurutan dilakukan dengan menggunakan metode *bubble-sort*.
    a. Pisah pcDeretAngka berdasarkan huruf “,” dan simpan hasilnya ke A1.
    b. Untuk nU1 = 0 sampai jumlah *array* dari variabel A1 -1, lakukan langkah berikut,
        i. Untuk nU2 = nU1 + 1 sampai jumlah *array* dari variabel A1, lakukan langkah berikut,
            1. Jika A1(nU1) > A1(nU2), maka tukar posisi dengan men-*set* nU3 = A1(nU1), set A1(nU1) = A1(nU2) dan set A1(nU2) = nU3.
    c. Untuk nU1 = 0 sampai jumlah *array* dari variabel A1, lakukan langkah berikut,

i. Jika nU1 > 0, maka set Urutkan = Urutkan & “,”.

ii. Urutkan = Urutkan & A1(nU1).

3. Fungsi ‘IsInList(pcAngka As String, pcDeretan As String)’ mengembalikan nilai *boolean*, apakah pcAngka berada di dalam pcDeretan.
    a. Set bFound = False.
    b. Pisah pcDeretan berdasarkan huruf “,” dan simpan ke dalam variabel *array* AList.
    c. Untuk F1 = 0 sampai jumlah *array* dari variabel AList, lakukan langkah berikut,
        i. Jika pcAngka = AList(F1), maka set bFound = *True*, dan keluar dari *looping*.
    d. Set IsInList = bFound.
4. Fungsi ‘FAllCheck(pMTermP() As TMTermP)’ melakukan pengecekan apakah semua *term* pMTermP sudah ter-*check*.
    a. Set FAllCheck = *True*.
    b. Untuk F1 = 0 sampai jumlah *array* dari pMTermP, lakukan langkah berikut,
        i. Jika pMTermP(F1).Check = *False*, maka set FAllCheck = *False* dan keluar dari *looping*.
5. Fungsi ‘GetBit1(pcBit as String)’ mengembalikan banyak bit ‘1’ dari pcBit.
    a. Untuk F1 = 1 sampai panjang dari pcBit, lakukan langkah berikut.
        i. Jika karakter ke-F1 dari pcBit = “1”, maka set nByk = nByk + 1.

b. Set GetBit1 = nByk.

6. Fungsi ‘PrimaToKanonik(pcBentukPrima As String)’ mengembalikan hasil konversi dari bentuk prima ke bentuk kanonik SOP / POS.
    a. Pisah pcBentukPrima berdasarkan huruf “,” dan simpan ke variabel *array* TmpBit.
    b. Untuk F1 = 0 sampai jumlah *array* dari TmpBit, set TmpBit(F1) = Bilangan biner dari TmpBit(F1).
    c. Bentuk ulang dimensi *array* dari variabel BitSama menjadi jumlah peubah.
    d. Untuk F1 = 1 sampai jumlah peubah, set BitSama(F1) = *True*.
    e. Untuk F1 = 1 sampai jumlah peubah, lakukan langkah berikut,
        i. Untuk F2 = 0 sampai jumlah *array* pada TmpBit,
            1. Jika F2 <> 0 dan karakter ke-F1 dari TmpBit(F2) <> cLast maka set BitSama(F1) = *False*, dan keluar dari *looping*.
            2. Set cLast = karakter ke-F1 dari TmpBit(F2).
    f. Untuk F1 = 1 sampai jumlah *array* pada variabel Peubah, lakukan langkah berikut,
        i. Jika BitSama(F1) = *True*, maka lakukan langkah pada poin (ii).
        ii. Jika cBentuk = “SOP”, maka,
            1. Set Hasil = Hasil & Peubah(F1).
            2. Jika karakter ke-F1 dari TmpBit(0) = “0”, maka set cHasil = cHasil & “’ “.

iii. Jika cBentuk = “POS”, maka,

1. Jika Hasil <> “” maka set Hasil = Hasil & “ + “.
2. Set Hasil = Hasil & Peubah(F1).
3. Jika karakter ke-F1 dari TmpBit(0) = “1”, maka set cHasil = cHasil & “’ “.

g. Jika cBentuk = “POS”, maka set Hasil = “(“ & Hasil & “)”.

h. Set PrimaToKanonik = Hasil.

4.2 Implementasi Sistem

Implementasi sistem program ini mencakup spesifikasi kebutuhan perangkat keras (*hardware*) dan spesifikasi perangkat lunak (*software*).

4.2.1 Spesifikasi Perangkat Keras

Program ini direkomendasikan untuk dijalankan dengan menggunakan perangkat keras (*hardware*) yang mempunyai spesifikasi minimum sebagai 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*.

4.2.2 Spesifikasi Perangkat Lunak

Adapun perangkat lunak (*software*) yang digunakan untuk menjalankan aplikasi ini minimal menggunakan sistem operasi *MS-Windows*98.

4.2.3 Analisis Pengguna

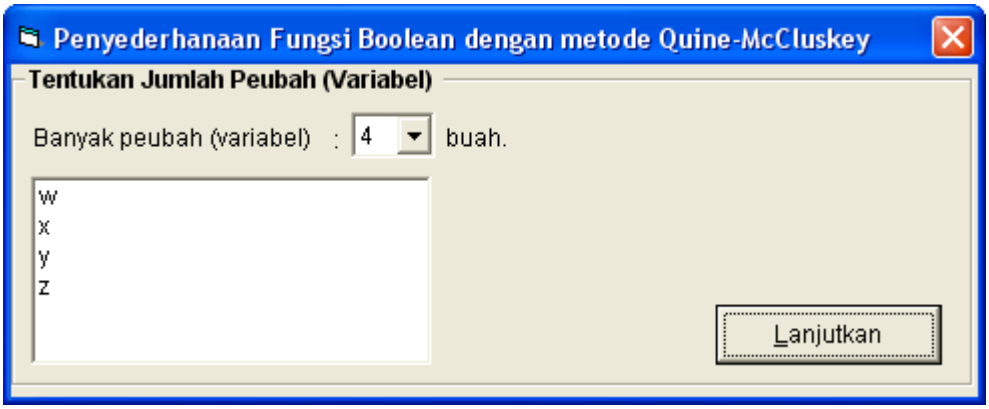
Pengguna adalah kalangan mahasiswa yang minimal memiliki wawasan tentang ilmu komputer, teknologi informasi, fungsi aljabar, expresi Boolean, dan gerbang logika.

4.3 Implementasi Perangkat Lunak

4.3.1 Contoh Kasus Implementasi Perangkat Lunak

Sebagai contoh, user meng-*input* data sebagai berikut:

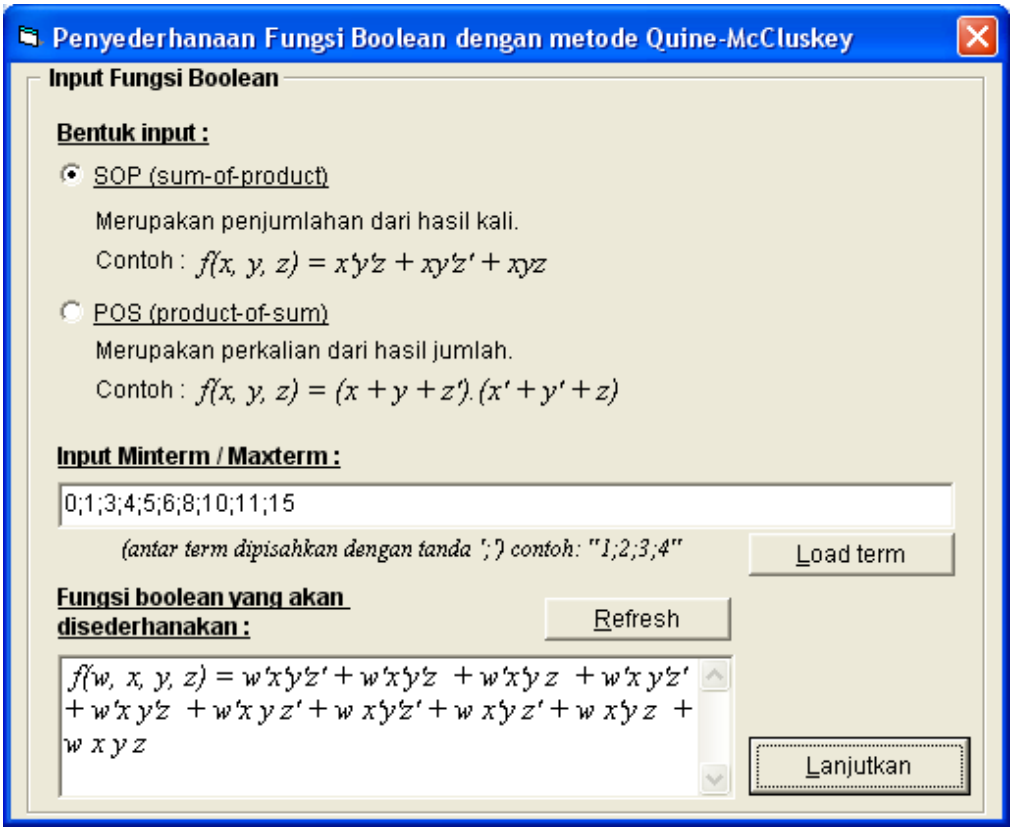
1. Banyak peubah (variabel) = 4 buah, yaitu : *w*, *x*, *y* dan *z*.



Gambar 4.1 Proses penentuan jumlah jumlah variabel

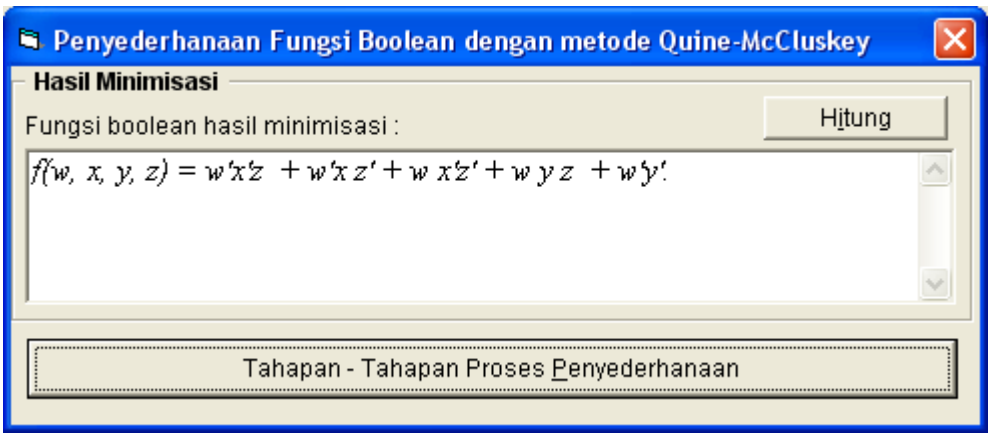
2. Bentuk fungsi *input* : SOP (*sum-of-product*)

3. *Input* nomor *minterm* : 0;1;3;4;5;6;8;10;11;15



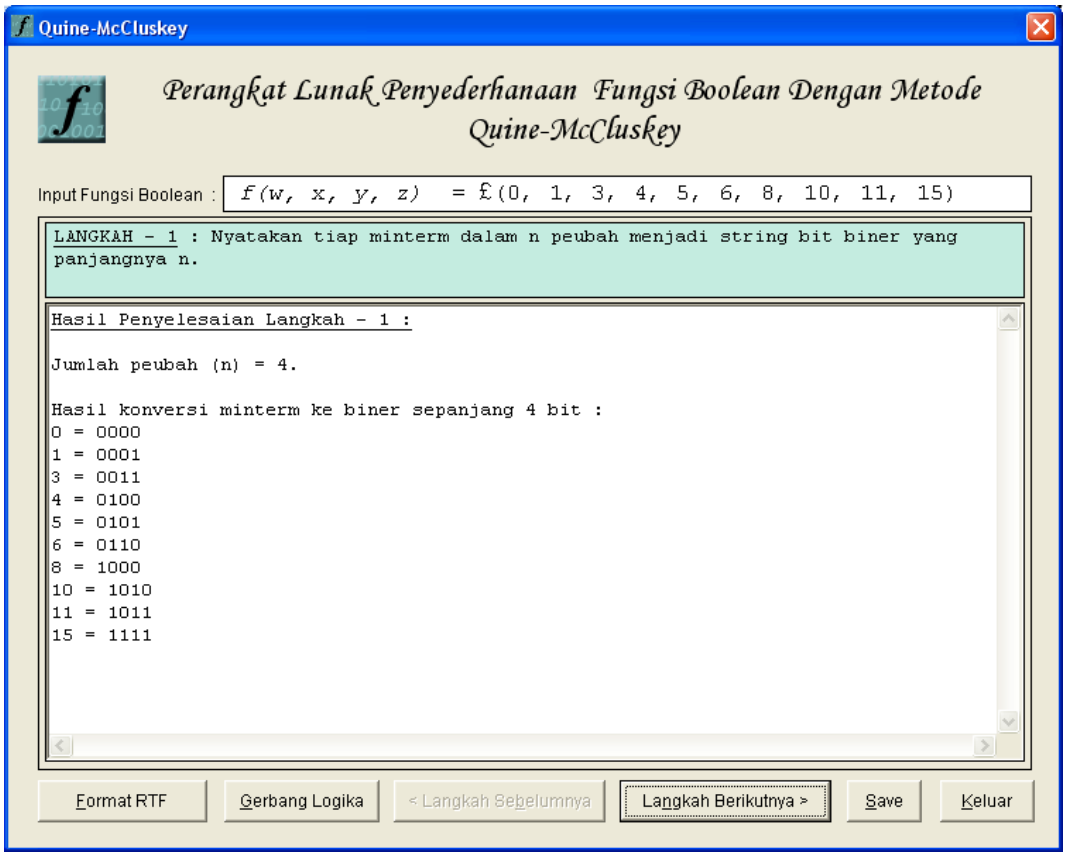
Gambar 4.2 Proses penginputan term dan bentuk kanonik

4. Hasil penyederhanaan : $f(w, x, y, z) = w'x'z+w'xz'+wx'z'+wyz+w'y'$.

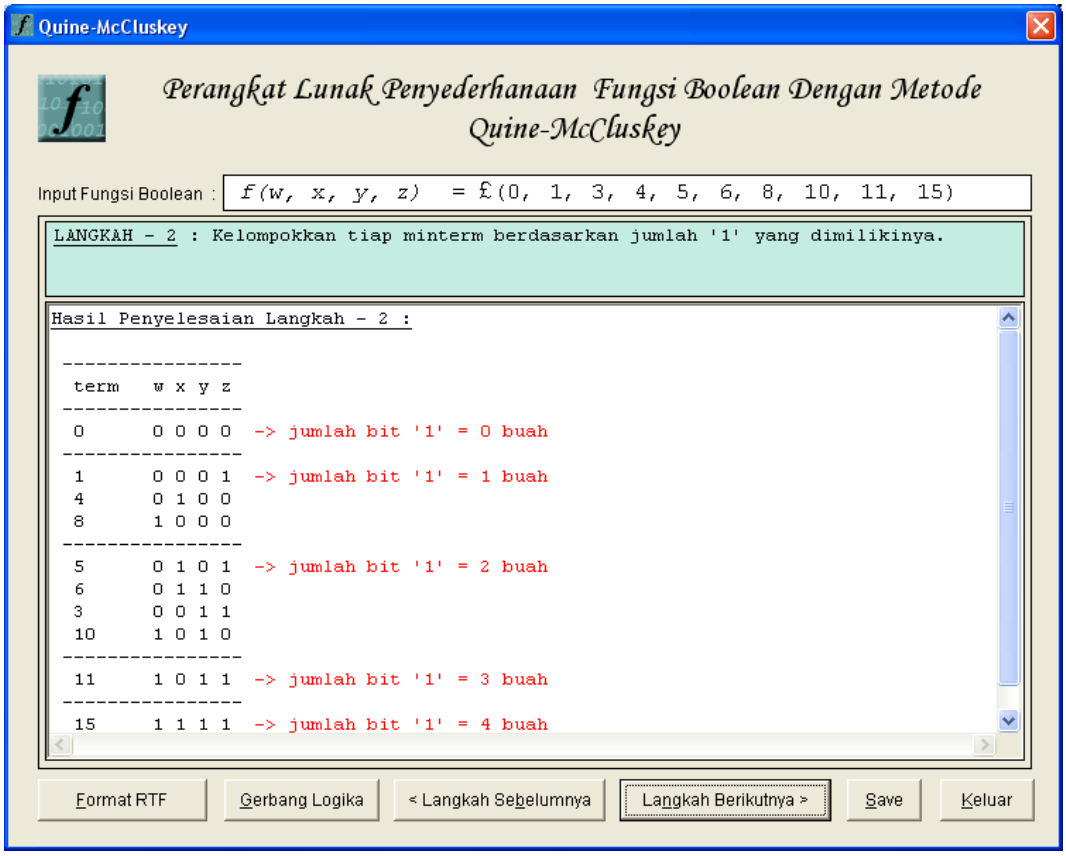


Gambar 4.3 Fungsi Boolean hasil minimisasi

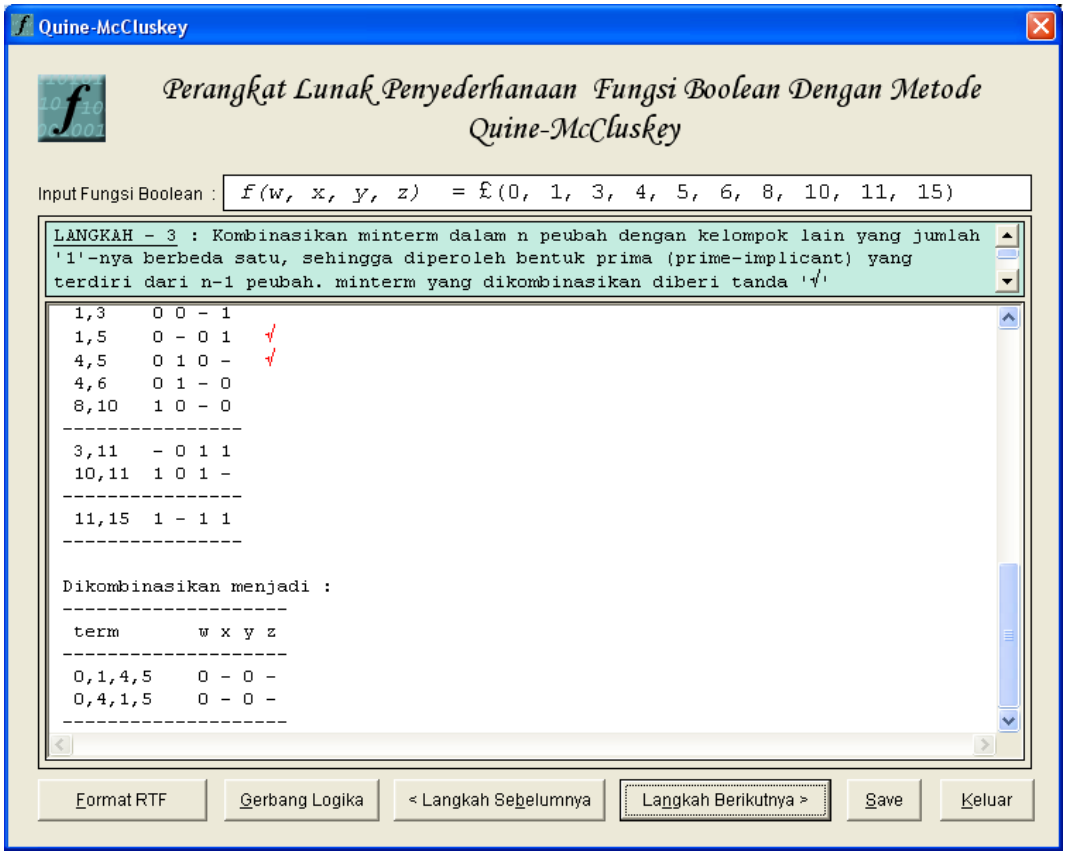
5. *Tahapan – tahapan proses penyederhanaan yang dihasilkan adalah sebagai* berikut:



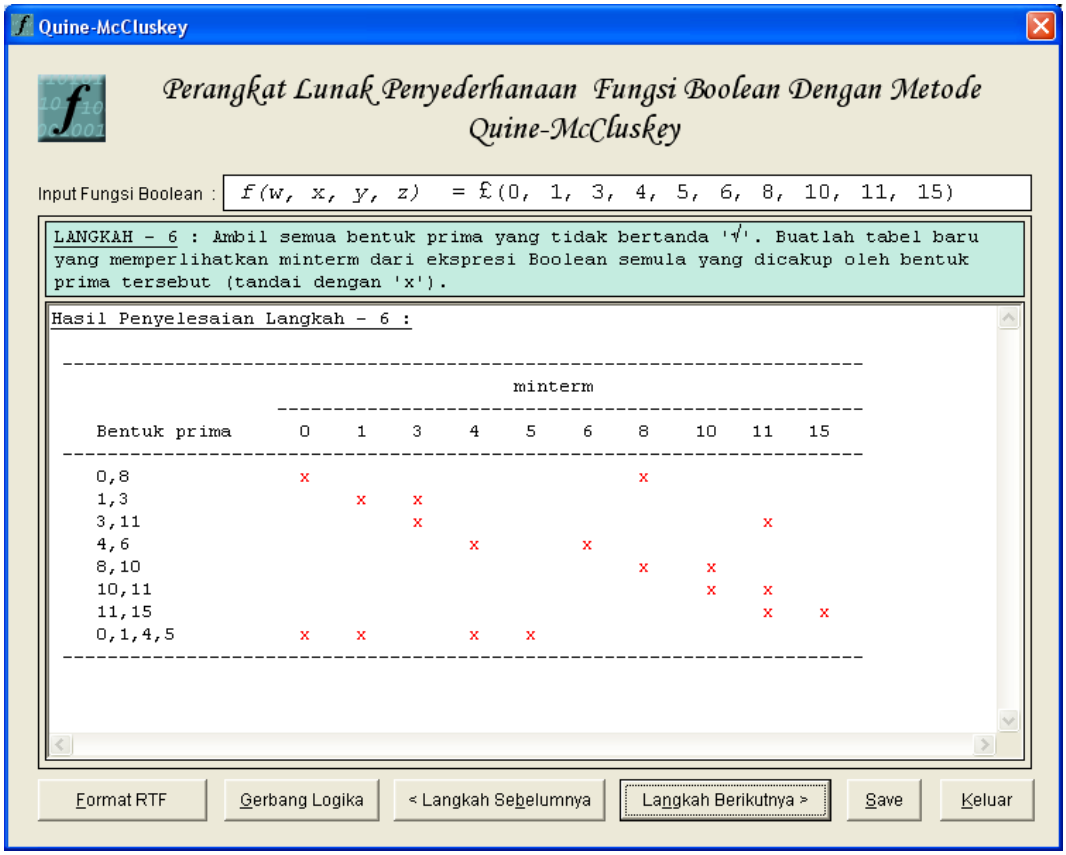
Gambar 4.4 Tampilan Langkah-1 proses minimisasi



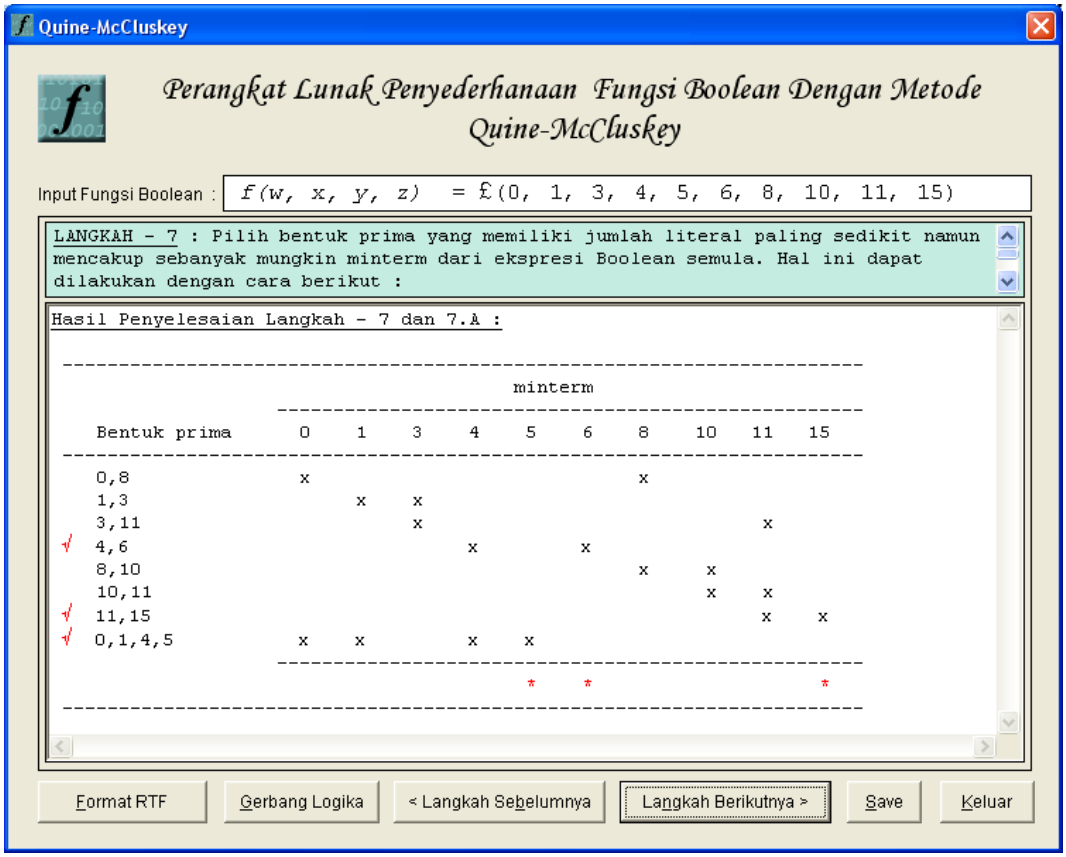
Gambar 4.5 Tampilan Langkah-2 proses minimisasi



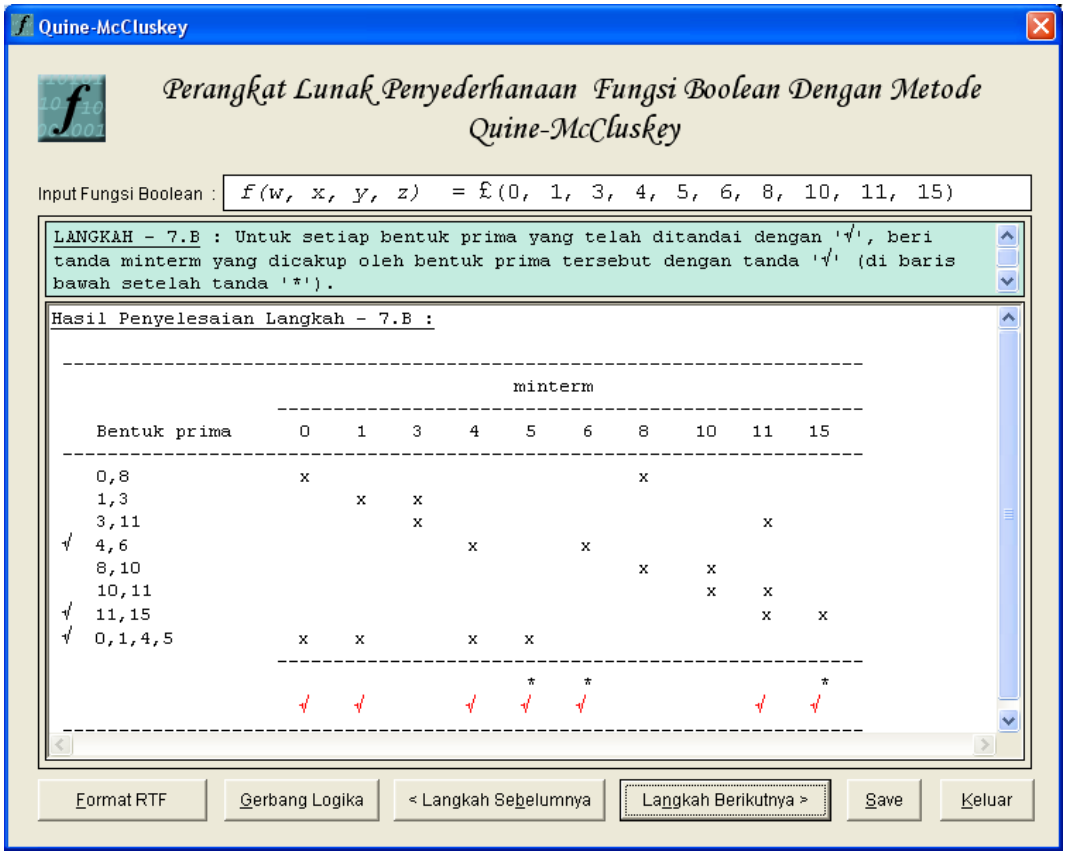
Gambar 4.6 Tampilan Langkah-3,4,5 proses minimisasi



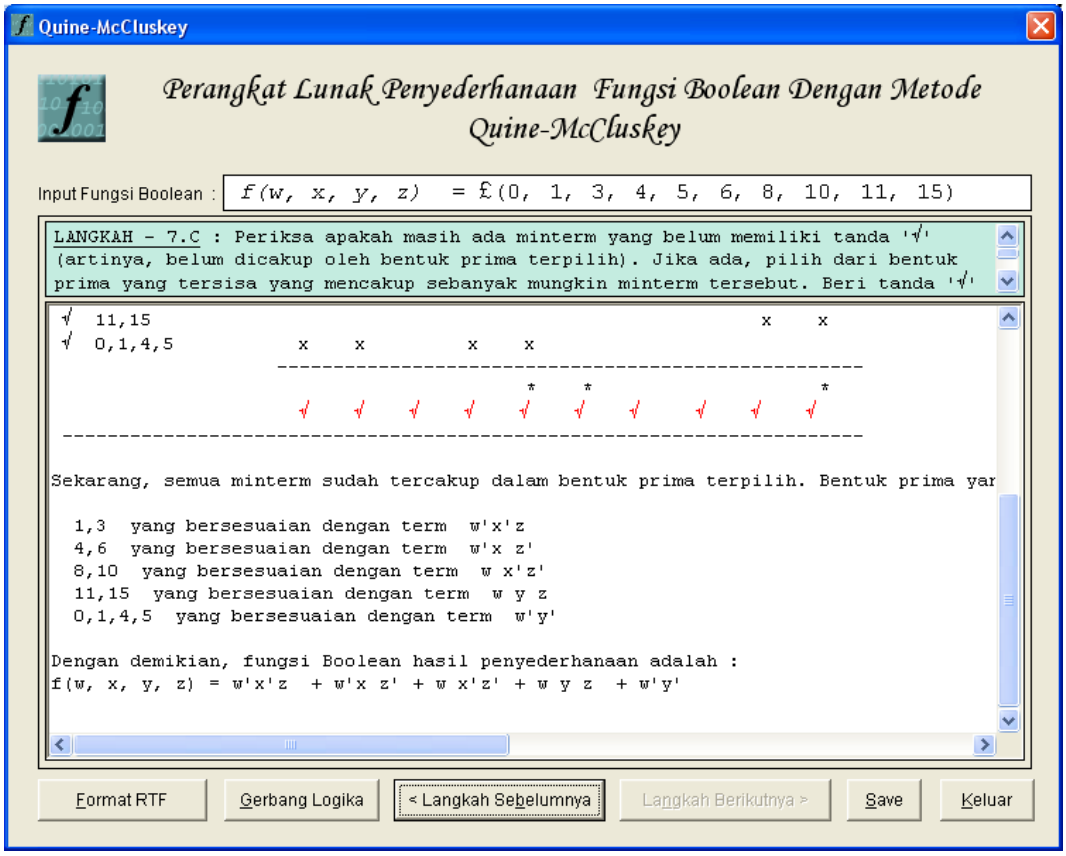
Gambar 4.7 Tampilan Langkah-6 proses minimisasi



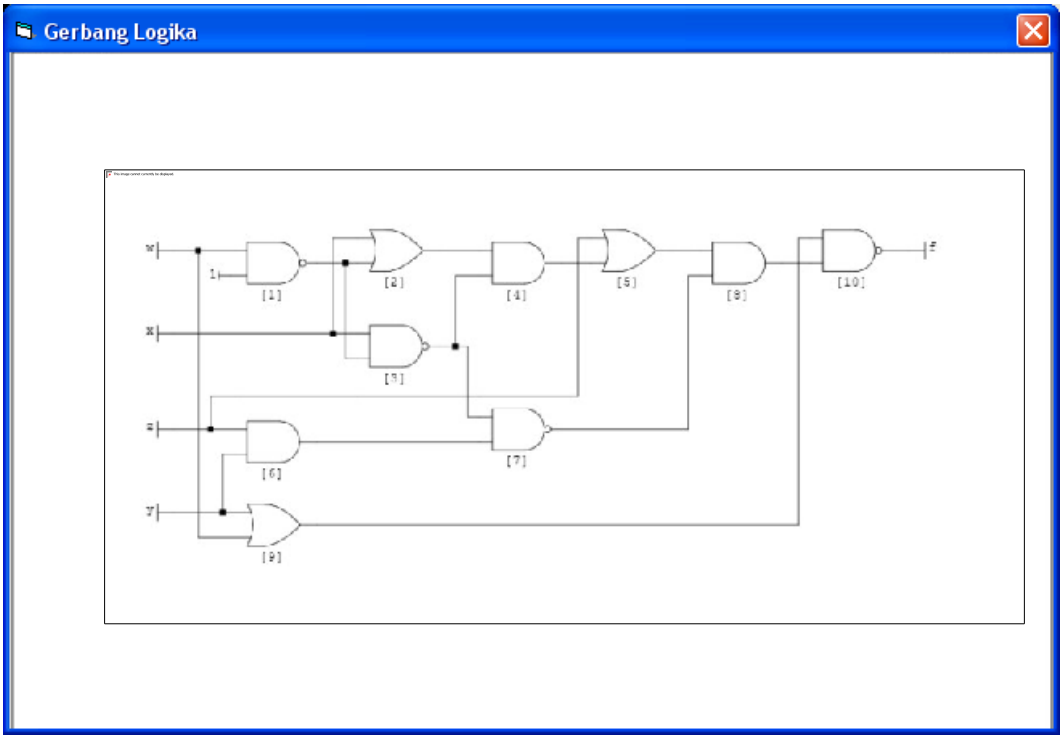
Gambar 4.8 Tampilan Langkah-7,7A proses minimisasi



Gambar 4.9 Tampilan Langkah-7B proses minimisasi



Gambar 4.10 Tampilan Langkah-7C,7D proses minimisasi

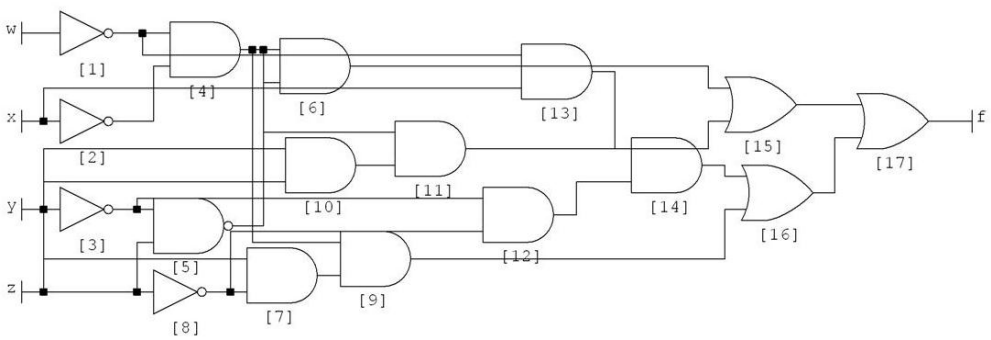


Gambar 4.11 Tampilan Gerbang logika fungsi hasil minimisasi

Satu contoh perbandingan antara gerbang logika dari suatu fungsi dengan gerbang logika dari fungsi yang sama yang sudah disederhanakan:

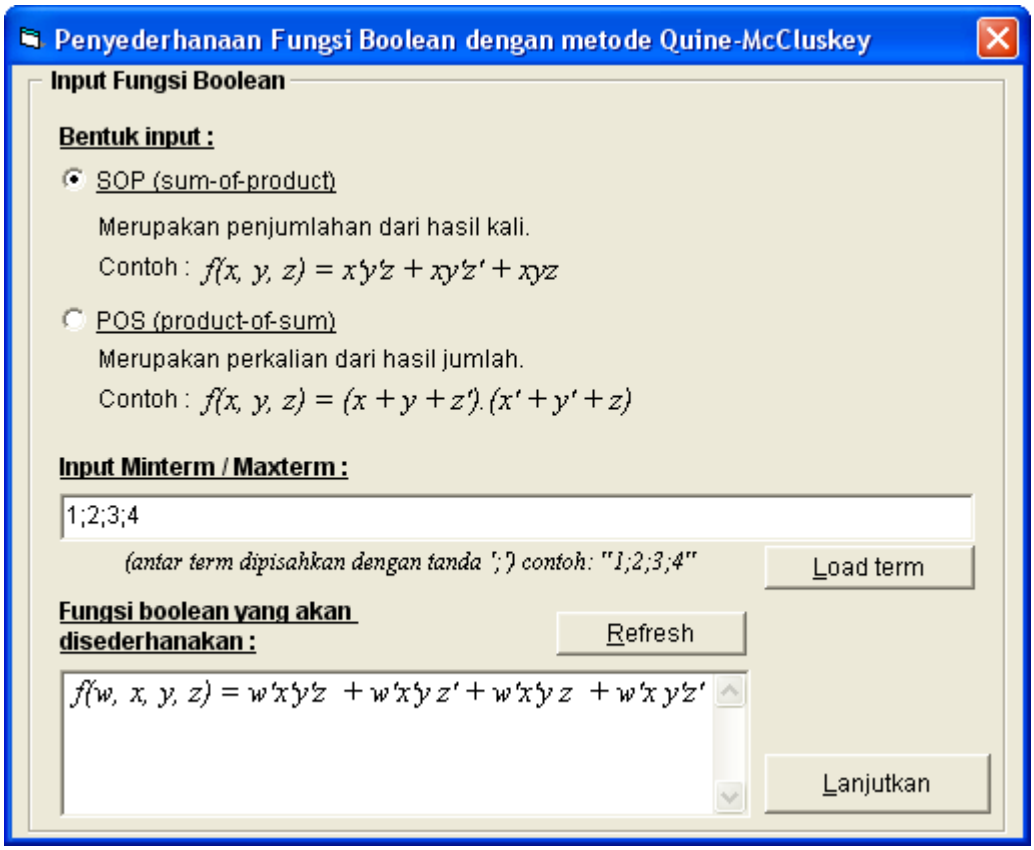
f(w, x, y, z) = w'x'y'z + w'x'y z' + w'x'y z + w'x y'z'

Gerbang logika yang dibentuk dari fungsi tersebut adalah:

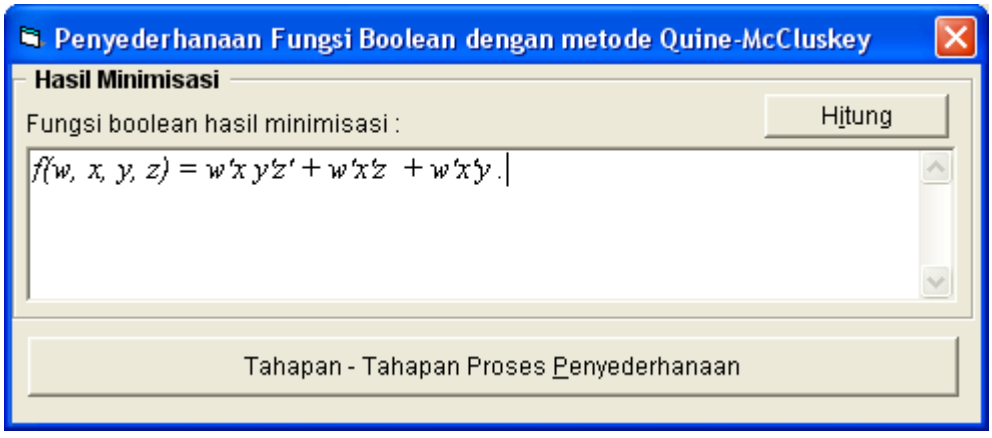


Gambar 4.12 Gerbang Logika fungsi sebelum disederhanakan

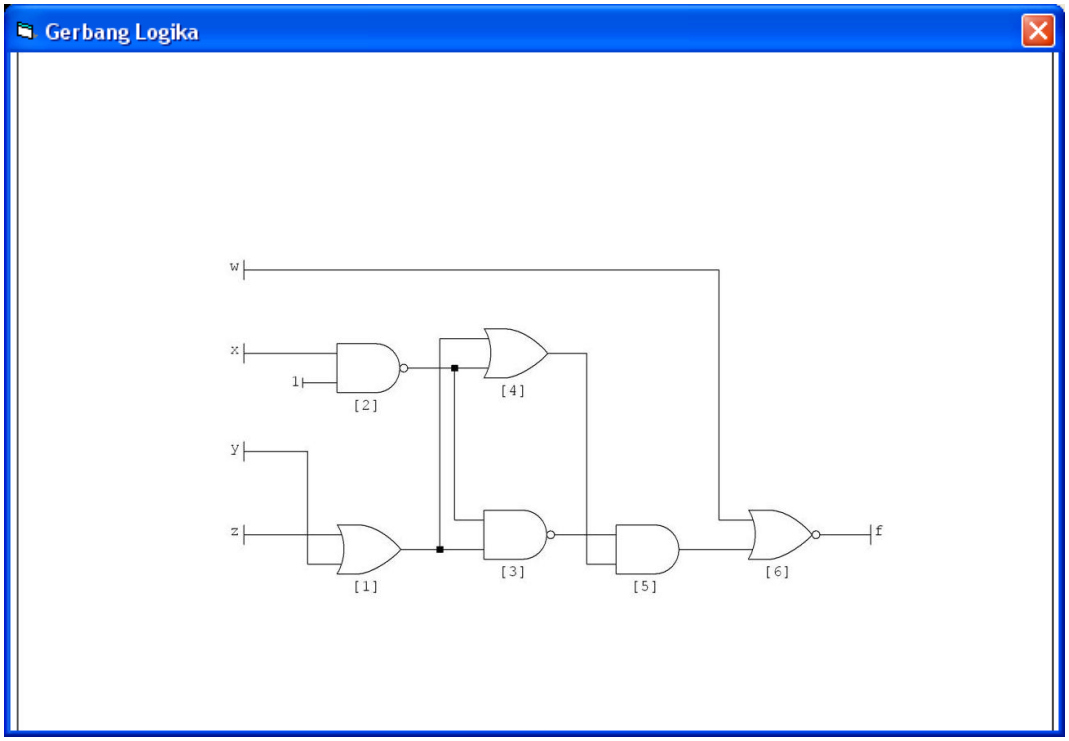
Setelah fungsi tersebut disederhanakan dengan menggunakan perangkat lunak yang telah dibangung adalah sebagai berikut:



Gambar 4.13 Tampilan Input term dan bentuk kanonik



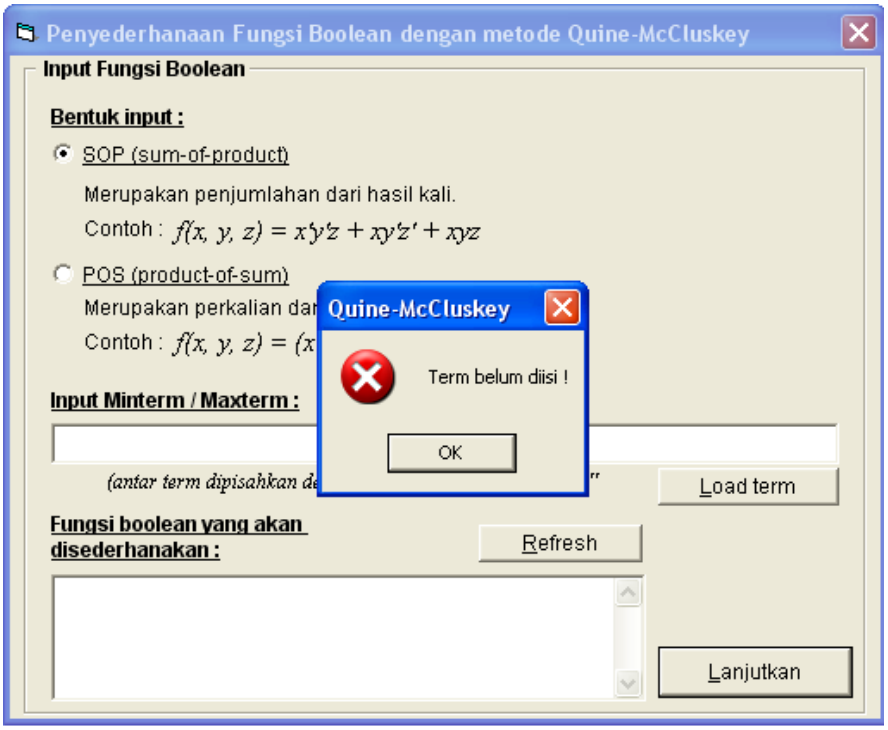
Gambar 4.14 Tampilan Fungsi Boolean hasil minimisasi



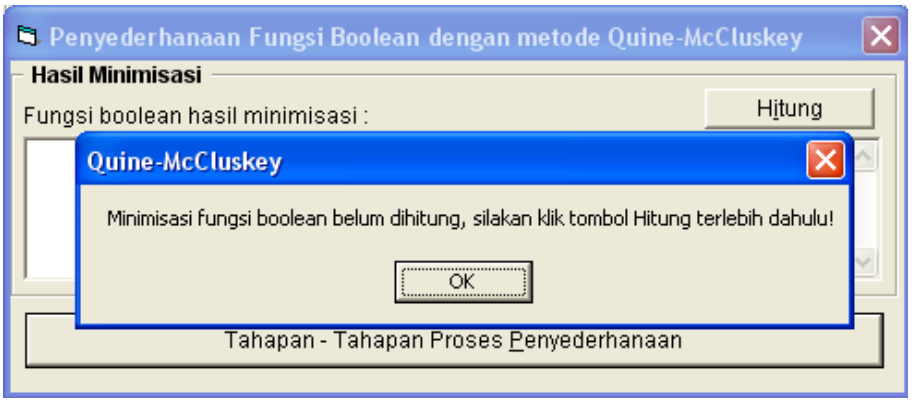
Gambar 4.15 Tampilan Gerbang logika fungsi hasil minimisasi

Terlihat perbedaan yang sangat significan ketika gerbang logika dibentuk dari suatu fungsi Boolean yang belum disederhanakan dengan fungsi Boolean setelah proses penyederhanaan.

4.3.2 Error Handler Dalam Implementasi Program



Gambar 4.16 Tampilan Alert ketika term belum diisi



Gambar 4.17 Tampilan Alert ketika menekan tombol tahapan sebelum tombol hitung

4.3.3 Pengujian Alpha

Pengujian dilakukan terhadap perangkat lunak guna memastikan bahwa perangkat lunak dapat berjalan dengan benar sesuai kebutuhan dan tujuan yang diharapkan. Proses pengujian terhadap perangkat lunak penyederhanaan ini akan menggunakan *black box* dan *white box testing*.

4.3.3.1 Skenario Pengujian Perangkat Lunak

Tabel 4.18 Skenario Pengujian Perangkat Lunak

No	Bagian Pengujian	Butir Uji	Jenis Pengujian
1.	Menu Utama	Tampilkan menu mulai penyederhanaan	*Black box*
		Tampilkan form about	*Black box*
		Eksekusi tombol “Keluar”	*Black box*
2.	Menentukan Jumlah Variabel	Pilih banyak variabel	*Black box*
		Eksekusi tombol “Lanjutkan”	*Black box*
3.	Menentukan Inputan Term	Melilih bentuk baku	*Black box*
		Menginputkan term	*Black box*
		Mengeluarkan kombinasi term yang sudah di simpan sebelumnya	*Black box*
		Eksekusi tombol “Refresh”	*Black box*
		Eksekusi tombol “Lanjutkan”	*Black box*
4.	Proses Hitung	Eksekusi tombol “Hitung”	*Black box*

	Minimisasi	Eksekusi tombol "Tahapan – Tahapan Proses Penyederhanaan"	*Black box*
5.	Langkah-Langkah Penyederhanaan	Menampilkan format '.rtf'	*Black box*
		Menampilkan gerbang logika	*Black box*
		Menyimpan kombinasi term	*Black box*
		Eksekusi tombol "Keluar"	*Black box*
6.	Alur Proses Quine-McCluskey	Langkah-langkah penyederhanaan Quine-McCluskey	*White Box*
7.	Algoritma Pendukung	*Bubble Sort*	*White Box*

4.3.3.2 Bagian Pengujian Perangkat Lunak

1. Pengujian Menu Utama

Pengujian menu utama merupakan proses dimana pengguna menyentuh menu-menu yang ada pada tampilan menu utama.

Tabel 4.19 Pengujian Menu Utama

Kasus dan Hasil Uji			
Masukan	Yang Diharapkan	Pengamatan	Kesimpulan
Data Pilih Menu Mulai	Sistem memulai tahapan proses penyederhanaan	Sistem menampilkan tahapan pertama proses penyederhanaan	[√] Berhasil [ ] Tidak Berhasil
Data Pilih Menu About	Sistem menampilkan form about	Sistem menampilkan form about	[√] Berhasil [ ] Tidak Berhasil
Data Pilih Menu Keluar	Keluar dari perangkat lunak	Keluar dari perangkat lunak	[√] Berhasil [ ] Tidak Berhasil

2. Menentukan Jumlah Variabel

Pengujian form menentukan jumlah variabel merupakan tahapan pertama proses penyederhanaan oleh perangkat lunak.

Tabel 4.20 Pengujian Menentukan Jumlah Variabel

Kasus dan Hasil Uji			
Masukan	Yang Diharapkan	Pengamatan	Kesimpulan
Data Pilih Banyak Variabel	Sistem menampilkan jumlah variabel dan menginputkan kedalam sistem	Sistem menampilkan banyaknya variabel dan menginputkan nilainya ke sistem	[√] Berhasil [ ] Tidak Berhasil
Eksekusi tombol "Lanjutkan"	Sistem menampilkan proses berikutnya	Sistem menampilkan form Input Bentuk Kanonik	[√] Berhasil [ ] Tidak Berhasil

3. Menentukan Inputan Term

Pengujian menentukan inputan term merupakan tahapan proses penyederhanaan setelah input variabel.

Tabel 4.21 Pengujian Menentukan Inputan Term

Kasus dan Hasil Uji			
Masukan	Yang Diharapkan	Pengamatan	Kesimpulan
Data Pilih Bentuk Baku	Bentuk baku (kanonik) terpilih dan dimasukkan ke sistem	Salah satu bentuk baku terpilih	[√] Berhasil [ ] Tidak Berhasil
Menginputkan Term	Kombinasi term ditentukan	Area kombinasi inputan term terisi	[√] Berhasil [ ] Tidak Berhasil
Eksekusi Tombol "Refresh"	*Generate* fungsi logika	Sistem menampilkan fungsi logika	[√] Berhasil [ ] Tidak Berhasil

Eksekusi Tombol "Lanjutkan"	Menuju ke tahapan proses berikutnya	Sistem menampilkan form proses hitung	[√] Berhasil [ ] Tidak Berhasil

4. Proses Hitung Minimisasi

Pengujian proses hitung minimisasi adalah tahapan proses penyederhanaan setelah penentuan variabel dan bentuk baku.

Tabel 4.22 Pengujian Hitung Minimisasi

Kasus dan Hasil Uji			
Masukan	Yang Diharapkan	Pengamatan	Kesimpulan
Eksekusi Tombol "Hitung"	Sistem melakukan proses minimisasi	Sistem menampilkan fungsi hasil minimisasi	[√] Berhasil [ ] Tidak Berhasil
Eksekusi Tombol "Tahapan-Tahapan Proses Penyederhanaan"	Sistem menampilkan form tahapan penyederhanaan	Sistem menampilkan tahapan-tahapan penyederhanaan	[√] Berhasil [ ] Tidak Berhasil

5. Langkah-Langkah Penyederhanaan

Pengujian proses langkah-langkah penyederhanaan adalah tahapan dalam menampilkan langkah-langkah penyederhanaan Quine.McCluskey.

Tabel 4.23 Pengujian Langkah-Langkah Penyederhanaan

Kasus dan Hasil Uji			
Masukan	Yang Diharapkan	Pengamatan	Kesimpulan
Data Tampilan Format '.rtf'	Sistem menampilkan langkah-langkah dalam format '.rtf'	Sistem menampilkan format '.rtf'	[√] Berhasil [ ] Tidak Berhasil
Data Tampilan Gerbang Logika	Sistem memberikan	Sistem menampilkan	[√] Berhasil

	tampilan gerbang logika	form gambar gerbang logika	[ ] Tidak Berhasil
Menyimpan Kombinasi Term	Sistem melakukan penyimpanan kombinasi term	Sistem menampilkan form penyimpanan	[√] Berhasil [ ] Tidak Berhasil
Eksekusi Tombol "Keluar"	Keluar dari perangkat lunak	Sistem *close*	[√] Berhasil [ ] Tidak Berhasil

6. Alur Proses Quine-McCluskey

Pengujian proses Quine-McCluskey merupakan pengujian dari pengimplementasian metode Quine-McCluskey dalam proses penyederhanaan fungsi Boolean pada perangkat lunak ini.

a. *Flowchart* Quine-McCluskey

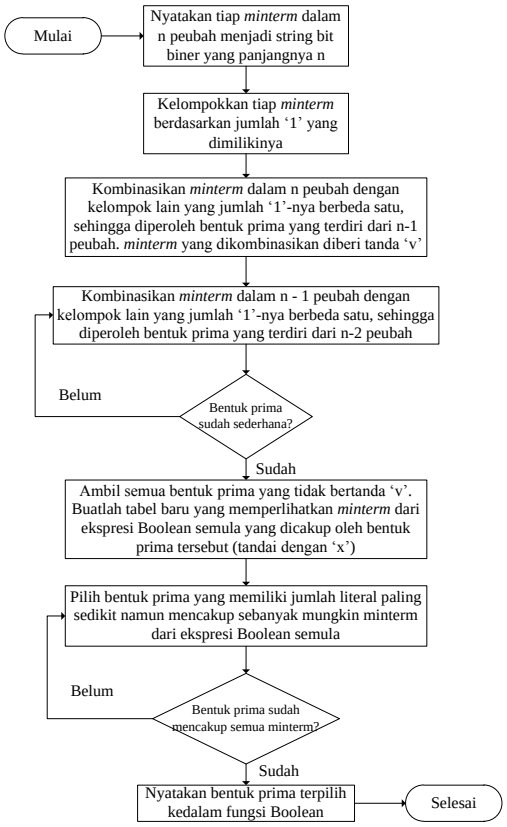


Diagram 4.24 Flowchart Skema Quine-McCluskey

b. *Flow graph* Quine-McCluskey

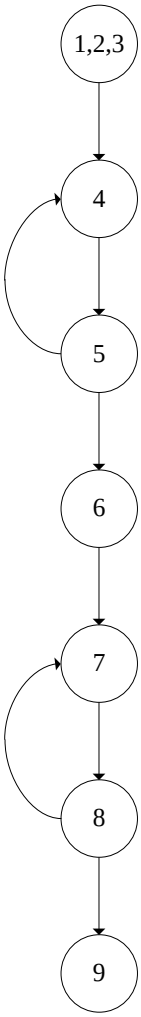
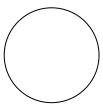


Diagram 4.25 Flowgraph Quine-McCluskey

Keterangan :



: Menggambarkan Kondisi

: Menggambarkan Aksi

c. *Cyclomatic Complexity* V(G)

E = 8, N = 7

V(G) = E – N + 2

V(G) = 8 – 7 + 2

V(G) = 3

Keterangan : E = Jumlah Aksi

N = Jumlah Kondisi

d. *Independent Path*

Path 1 = (1.2.3) – (4) – (5) – (6) – (7) – (8) – (9)

Path 2 = (1.2.3) – (4) – (5) – (4) – (5) – (6) – (7) – (8) – (9)

Path 3 = (1.2.3) – (4) – (5) – (6) – (7) – (8) – (7) – (8) – (9)

e. *Graph Matriks*

Tabel 4.26 Graph Matriks Proses Quine-McCluskey

	1,2,3	4	5	6	7	8	9	E
1,2,3		1						
4			1					
5		1		1				1
6					1			
7						1		
8					1		1	1
9								
						E Sum + 1		3

Keterangan :

1. Baris dan kolom merepresentasikan simpul
2. Nilai 1 merepresentasikan adanya keterhubungan antar simpul

f. Hasil Uji Proses Quine-McCluskey

Tabel 4.27 Hasil Pengujian Proses Quine-McCluskey

Kasus dan Hasil Uji				
No. *Path*	Node (n), input kondisi	Yang Diharapkan	Pengamatan	Kesimpulan
1.	(1.2.3) – (4) – (5) – (6) – (7) – (8) – (9)	Penyederhanaan Quine-McCluskey	Sistem menjalankan dan menampilkan proses penyederhanaan Quine-McCluskey	[√] Berhasil [ ] Tidak Berhasil
2.	(1.2.3) – (4) – (5) – (4) – (5) – (6) – (7) – (8) – (9)	Kombinasikan minterm selama bentuk prima belum sederhana	Sistem melakukan kombinasi minterm sampai bentuk prima sederhana	[√] Berhasil [ ] Tidak Berhasil
3.	(1.2.3) – (4) – (5) – (6) – (7) – (8) – (7) – (8) – (9)	Pemilihan bentuk prima selama masih ada term yang belum ter cek	Bentuk prima dipilih sampai semua bentuk prima mencakup semua term	[√] Berhasil [ ] Tidak Berhasil

7. Algoritma Pendukung

Pengujian algoritma pendukung ini adalah pengujian terhadap algoritma yang ada dalam langkah-langkah proses penyederhanaan. Tentunya algoritma yang sangat berpengaruh terhadap jalannya proses penyederhanaan. Algoritma tersebut adalah algoritma pengurutan dengan *bubble sort*.

a Algoritma *bubble sort*

```
1       For N1 = 1 To UBound(Pos) - 1
2               For N2 = N1 + 1 To UBound(Pos)
                  'Urut dari besar ke kecil
3                 If Prima(Pos(N1)).Nilai < Prima(Pos(N2)).Nilai
                  Then
                  'Tukar Posisi
4                 N3 = Pos(N1)
                  Pos(N1) = Pos(N2)
                  Pos(N2) = N3
5                 ElseIf Prima(Pos(N1)).Nilai = Prima(Pos(N2)).Nilai
```

```
                Then
                   'Jika sama, maka pilih bentuk prima yang paling
                   panjang
6                  If UBound(Split(Prima(Pos(N1)).Term, ",")) < _
                         UBound(Split(Prima(Pos(N2)).Term, ","))
                   Then
                   'Tukar Posisi
7                  N3 = Pos(N1)
                   Pos(N1) = Pos(N2)
                   Pos(N2) = N3
8                  End If
                End If
9            Next N2
10      Next N1
```

b. *Flow graph bubble sort*

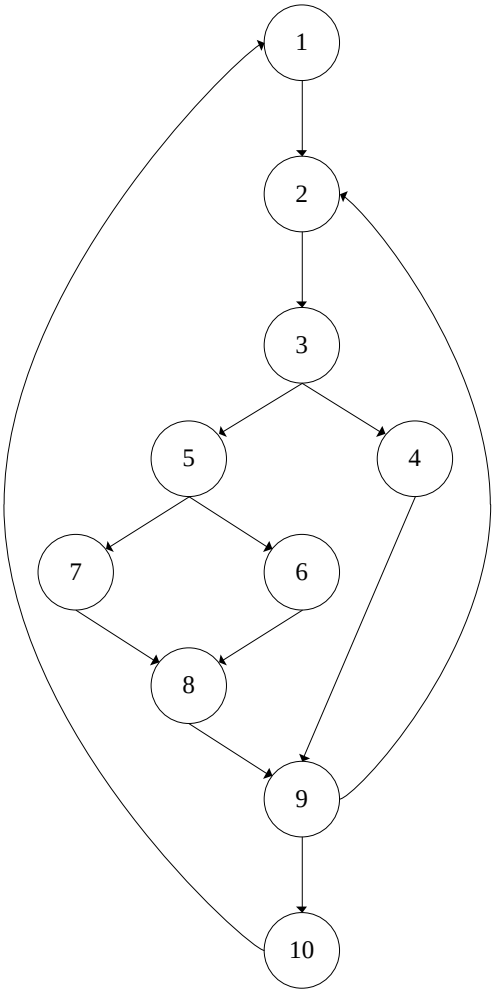


Diagram 4.28 Flowgraph Bubble Sort

c. *Cyclomatic Complexity* V(G)

E = 13, N = 10

V(G) = E – N + 2

V(G) = 13 – 10 + 2

V(G) = 5

d. *Independent Path*

Path 1 = (1) – (2) – (3) – (4) – (9) – (10)

Path 2 = (1) – (2) – (3) – (5) – (6) – (8) – (9) – (10)

Path 3 = (1) – (2) – (3) – (5) – (7) – (8) – (9) – (10)

Path 4 = (1) – (2) – (3) – (4) – (9) – (2) – (3) – (4) – (9) – (10)

Path 5 = (1) – (2) – (3) – (4) – (9) – (10) – (1)

e. *Graph Matriks*

Tabel 4.29 Graph Matriks Bubble Sort

	1	2	3	4	5	6	7	8	9	10	E
1		1								1	1
2			1								
3				1	1						1
4									1		
5						1	1				1
6								1			
7								1			
8									1		
9		1								1	1
10	1										
									E Sum + 1		5

f. Hasil Uji *Bubble Sort*

Tabel 4.30 Pengujian Bubble Sort

Kasus dan Hasil Uji				
No. *Path*	Node (n), input kondisi	Yang Diharapkan	Pengamatan	Kesimpulan
1.	(1), (2), (3), (4), (9), (10)	Tukar posisi ketika N1 < N2	Sistem menukar posisi N1 dengan N2	[√] Berhasil [ ] Tidak Berhasil
2.	(1), (2), (3), (5), (6), (8), (9), (10)	Tukar posisi ketika N1 < N2	Sistem menukar posisi N1 dengan N2	[√] Berhasil [ ] Tidak Berhasil
3.	(1), (2), (3), (5), (7), (8), (9), (10)	Ketika N1 > N2 keluar dari kondisi	Sistem keluar dari kondisi dan menjalankan aksi berikutnya	[√] Berhasil [ ] Tidak Berhasil
4.	(1) – (2) – (3) – (4) – (9) – (2) – (3) – (4) – (9) – (10)	Mengulang deretan aksi selama N1 < UBound	Sistem mengerjakan deretan aksi berulang sampai N1=UBound	[√] Berhasil [ ] Tidak Berhasil
5.	(1), (2), (3), (4), (9), (10), (1)	Mengulang deretan aksi selama N1 < UBound	Sistem mengerjakan deretan aksi berulang sampai N1=UBound	[√] Berhasil [ ] Tidak Berhasil

4.3.4 Pengujian Beta

Pengujian beta merupakan pengujian yang dilakukan secara objektif, dimana pengujian dilakukan secara langsung ke kalangan yang memang menjadi target *user* atau pengguna perangkat lunak penyederhanaan ini. Pengujian dilakukan dengan cara membagikan kuesioner kepada para pengguna (dalam kasus ini targetnya adalah mahasiswa) guna mendapatkan data tentang kepuasan pengguna terhadap perangkat lunak penyederhanaan yang telah dibangun ini. Dengan kandungan *point* sesuai dengan tujuan dibangunnya perangkat lunak penyederhanaan Quine-McCluskey ini.

4.3.4.1 Kuesioner

Kuesioner disebarkan menggunakan teknik *Simple Random Sampling* dimana anggota *sample* dari populasi dilakukan secara acak tanpa memperhatikan strata yang ada dalam populasi[7].

Kuesioner disebarkan kepada 30 orang mahasiswa, dari hasil kuesioner tersebut akan dilakukan perhitungan agar dapat diambil kesimpulan terhadap penilaian perangkat lunak penyederhanaan Quine-McCluskey. Kuesioner terdiri dari 7 pernyataan dengan menggunakan skala 1 sampai 5 (skala likert).

Untuk mencari nilai persentase dari masing-masing pendapat kuesioner digunakan rumus skala likert [7] sebagai berikut :

$$P = \frac{S}{skorideal} \times 100$$

Keterangan :

P = Nilai persentase yang dicari

S = Jumlah frekuensi dikalikan dengan skor yang ditetapkan tiap jawaban

skorideal = Skor tertinggi dikalikan dengan jumlah sampel

Berikut ini adalah hasil persentase masing-masing pendapat yang sudah dihitung nilainya dengan menggunakan rumus skala likert.

1. Perangkat lunak ini membantu dalam proses penyederhanaan fungsi Boolean.

Tabel 4.31 Perhitungan Persentase Pernyataan ke-1

Pernyataan	**Keterangan**	**Skor**	**Frekuensi**	**S**
1	Sangat Setuju	5	21	105
	Setuju	4	9	36
	Ragu-ragu	3	-	
	Tidak Setuju	2	-	
	Sangat Tidak Setuju	1	-	
Jumlah			**30**	**141**

$$P = \frac{141}{150} \times 100\% = 94\%$$

Berdasarkan hasil persentase nilai diatas, maka dapat disimpulkan bahwa penilaian terhadap perangkat lunak penyederhanaan Quine-McCluskey adalah

94% dari yang diharapkan 100%, atau bisa dikategorikan sebagai sangat setuju. Berikut kriteria interpretasi skor :



Gambar 4.32 Interpretasi Skor Pernyataan ke-1

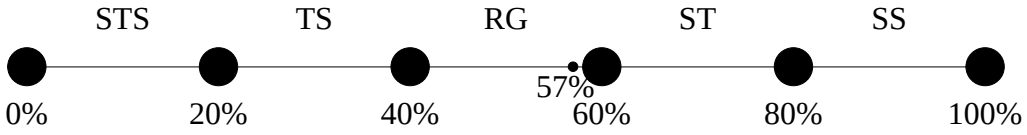
2. Perangkat lunak ini membantu dalam proses penyederhanaan gerbang logika.

Tabel 4.33X Perhitungan Persentase Pernyataan ke-2

Pernyataan	**Keterangan**	**Skor**	**Frekuensi**	**S**
2	Sangat Setuju	5	2	10
	Setuju	4	4	16
	Ragu-ragu	3	11	33
	Tidak Setuju	2	13	26
	Sangat Tidak Setuju	1	-	-
Jumlah			**30**	**85**

$$P = \frac{85}{150} \times 100\% = 57\%$$

Berdasarkan hasil persentase nilai diatas, maka dapat disimpulkan bahwa penilaian terhadap perangkat lunak penyederhanaan Quine-McCluskey adalah 59% dari yang diharapkan 100%, atau bisa dikategorikan sebagai ragu-ragu. Berikut kriteria interpretasi skor :



Gambar 4.34 Interpretasi Skor Pernyataan ke-2

3. Penjelasan langkah penyederhanaan, membantu user memahami prosedur Quine-McCluskey.

Tabel 4.35 Perhitungan Persentase Pernyataan ke-3

Pernyataan	Keterangan	Skor	Frekuensi	S
3	Sangat Setuju	5	5	25
	Setuju	4	19	76
	Ragu-ragu	3	6	18
	Tidak Setuju	2	-	-
	Sangat Tidak Setuju	1	-	-
Jumlah			**30**	**119**

$$P = \frac{119}{150} \times 100\% = 79\%$$

Berdasarkan hasil persentase nilai diatas, maka dapat disimpulkan bahwa penilaian terhadap perangkat lunak penyederhanaan Quine-McCluskey adalah 79% dari yang diharapkan 100%, atau bisa dikategorikan sebagai setuju. Berikut kriteria interpretasi skor :



Gambar 4.36 Interpretasi Skor Pernyataan ke-3

4. Kesalahan proses tabulasi Quine-McCluskey tidak ditemukan lagi.

Tabel 4.37 Perhitungan Persentase Pernyataan ke-4

Pernyataan	Keterangan	Skor	Frekuensi	S
4	Sangat Setuju	5	18	90
	Setuju	4	12	48
	Ragu-ragu	3	-	-
	Tidak Setuju	2	-	-
	Sangat Tidak Setuju	1	-	-
Jumlah			**30**	**138**

$$P = \frac{138}{150} \times 100\% = 92\%$$

Berdasarkan hasil persentase nilai diatas, maka dapat disimpulkan bahwa penilaian terhadap perangkat lunak penyederhanaan Quine-McCluskey adalah 92% dari yang diharapkan 100%, atau bisa dikategorikan sebagai sangat setuju. Berikut kriteria interpretasi skor :



Gambar 4.38 Interpretasi Skor Pernyataan ke-4

5. Mengurangi *human error* dalam proses penyederhanaan fungsi Boolean.

Tabel 4.39 Perhitungan Persentase Pernyataan ke-5

Pernyataan	**Keterangan**	**Skor**	**Frekuensi**	**S**
5	Sangat Setuju	5	20	100
	Setuju	4	10	40
	Ragu-ragu	3	-	-
	Tidak Setuju	2	-	-
	Sangat Tidak Setuju	1	-	-
Jumlah			**30**	**140**

$$P = \frac{140}{150} \times 100\% = 93\%$$

Berdasarkan hasil persentase nilai diatas, maka dapat disimpulkan bahwa penilaian terhadap perangkat lunak penyederhanaan Quine-McCluskey adalah 93% dari yang diharapkan 100%, atau bisa dikategorikan sebagai sangat setuju. Berikut kriteria interpretasi skor :



Gambar 4.40 Interpretasi Skor Pernyataan ke-5

6. Mengurangi *human error* dalam proses pembangkitan gerbang logika.

Tabel 4.41 Perhitungan Persentase Pernyataan ke-6

Pernyataan	**Keterangan**	**Skor**	**Frekuensi**	**S**
6	Sangat Setuju	5	20	100
	Setuju	4	10	40
	Ragu-ragu	3	-	-
	Tidak Setuju	2	-	-
	Sangat Tidak Setuju	1	-	-
Jumlah			**30**	**140**

$$P = \frac{140}{150} \times 100\% = 93\%$$

Berdasarkan hasil persentase nilai diatas, maka dapat disimpulkan bahwa penilaian terhadap perangkat lunak penyederhanaan Quine-McCluskey adalah 93% dari yang diharapkan 100%, atau bisa dikategorikan sebagai sangat setuju. Berikut kriteria interpretasi skor :



Gambar 4.42 Interpretasi Skor Pernyataan ke-6

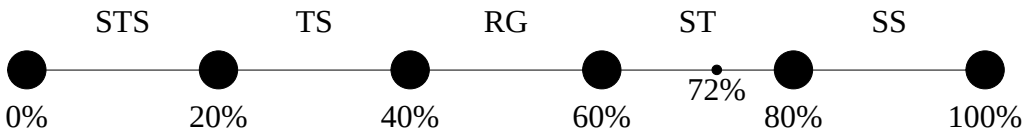
7. Mengefisienkan waktu penyederhanaan.

Tabel 4.43 Perhitungan Persentase Pernyataan ke-7

Pernyataan	Keterangan	Skor	Frekuensi	S
7	Sangat Setuju	5	6	30
	Setuju	4	11	44
	Ragu-ragu	3	8	24
	Tidak Setuju	2	5	10
	Sangat Tidak Setuju	1	-	-
Jumlah			**30**	**108**

$$P = \frac{108}{150} \times 100\% = 72\%$$

Berdasarkan hasil persentase nilai diatas, maka dapat disimpulkan bahwa penilaian terhadap perangkat lunak penyederhanaan Quine-McCluskey adalah 72% dari yang diharapkan 100%, atau bisa dikategorikan sebagai setuju. Berikut kriteria interpretasi skor :



Gambar 4.44 Interpretasi Skor Pernyataan ke-7

4.3.4.2 Kesimpulan Pengujian Beta

Berdasarkan hasil kuesioner yang diberikan kepada para mahasiswa, sebagian besar mengatakan setuju bahwa perangkat lunak penyederhanaan fungsi Boolean Quine-McCluskey ini telah sesuai dengan tujuan yang ingin dicapai, yaitu membantu menyederhanakan fungsi Boolean beserta gerbang logikanya, membantu pemahaman proses tabulasi Quine-McCluskey, serta mengefisienkan waktu penyederhanaan dan implementasi gerbang logikanya. Juga mengurangi kesalahan-kesalahan akibat *human error*.

4.3.5 Kelemahan Metode Quine-McCluskey

Untuk masalah praktis yang selalu melibatkan lima atau lebih variabel, Quine [15] merancang prosedur dimana redundansi bisa dihilangkan dengan membandingkan setiap istilah dari fungsi dalam bentuk simbolik dengan istilah lain. Prosedur ini cocok untuk penggunaan komputer digital untuk mengurangi jumlah memanipulasi diperlukan untuk minimalisasi.

McCluskey [16] memodifikasi metode Quine dengan menggunakan representasi biner dari variabel sehingga perbandingan istilah akan lebih mudah beradaptasi untuk manipulasi komputer. Metode McCluskey melibatkan manipulasi berdasarkan karakter numerik sederhana yang terdiri dari angka nol, satu dan tanda hubung. Jika menggunakan komputer, maka ruang penyimpanan yang dibutuhkan kecil karena pekerjaannya berkonsentrasi hanya pada produk dasar yang terkait dengan unit dan nilai-nilai fungsional yang terperinci. Namun sudah lebih nyaman dan kompak untuk menentukan fungsi *switching* sebagai

suatu himpunan bilangan desimal. Dengan demikian Abraham dan Nordhal [17] memodifikasi prosedur ini lebih lanjut dengan memperkenalkan suatu prosedur dimana *prime implicants* [18] dapat ditemukan langsung dari representasi desimal.

Di dalam pemilihan ketentuan jumlah yang minimum dari grafik *prime implicants* yang diperoleh, metode McCluskey yang memiliki kerugian dari ketergantungannya pada keterampilan dalam menentukan baris dasar, setelah baris dasar primer pada tabel sudah ditandai. Dalam masalah sederhana, kelemahan ini tidak mungkin tidak begitu signifikan, tetapi ketika meningkatnya kompleksitas grafik *prime implicants*, ini akan menjadi lebih sulit dalam mengevaluasi manfaat relatif dari berbagai alternatif dari jumlah minimum yang cukup besar tanpa adanya proses *try & error* yang dapat dipertimbangkan. Oleh karena itu untuk mengatasi masalah yang sangat kompleks dimana jumlah variabel melebihi enam, metode aljabar untuk mengevaluasi jumlah minimum menggunakan penandaan desimal.