PENYEDERHANAAN SIRKUIT LOGIKA DENGAN METODE QUINE-McCLUSKEY

Oleh :

MUHAJIR ARMAN

H 111 97 001

JURUSAN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS HASANUDDIN

2002

PENYEDERHANAAN SIRKUIT LOGIKA DENGAN METODE QUINE-McCLUSKEY

Skripsi

melengkapi tugas-tugas dan memenuhi

syarat-syarat untuk mencapai

gelar sarjana

Oleh :

MUHAJIR ARMAN

H 111 97 001

JURUSAN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS HASANUDDIN

2002

PENYEDERHANAAN SIRKUIT LOGIKA DENGAN METODE QUINE-McCLUSKEY

Disetujui oleh :

Pembimbing Utama

Drs. Locky Haryanto, MS., MSc., MA
NIP. 131 289 844

Pembimbing Pertama

Drs. M. Saleh AF.
NIP. 130 675 575

Pembimbing Kedua

A. Kresna Jaya, SSi., MSi.
NIP. 132 259 231

Pada tanggal : **2002**

KATA PENGANTAR

Puji syukur Penulis panjatkan kehadirat Allah SWT, karena atas berkah, rahmat, dan hidayah-Nyalah sehingga penulisan skripsi ini dapat diselesaikan. Penulis menyadari adanya kekurangan dan kelemahan dalam penulisan skripsi ini yang disebabkan karena keterbatasan yang dimiliki oleh Penulis.

Skripsi ini merupakan hasil kerja yang tidak akan berhasil tanpa dukungan, dorongan, petunjuk, dan keterlibatan pihak lain. Oleh karena itu, pada kesempatan ini tidaklah berlebihan jika Penulis menyampaikan ucapan terima kasih dan penghargaan mendalam kepada yang terhormat :

1. Bapak Rektor Universitas Hasanuddin beserta stafnya.
2. Bapak Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin beserta stafnya.
3. Bapak Ketua Jurusan Matematika Universitas Hasanuddin.
4. Bapak Drs. Raupong,MSi selaku ketua dosen penguji dan Bapak Drs. Muh. Zakir, MSi selaku sekretaris dosen penguji.
5. Bapak Drs. Loeky Haryanto, MS, MSc, MA, selaku pembimbing utama, Bapak Drs. M. Saleh AF, selaku pembimbing pertama dan Bapak A. Kresna Jaya, SSi, MSi selaku pembimbing kedua yang senantiasa memberikan bimbingan dan pengetahuan serta perhatian sejak awal sampai akhir penulisan skripsi ini.

6. Bapak dan Ibu staf dosen Jurusan Matematika Universitas Hasanuddin yang telah banyak membantu mahasiswa dalam menyalurkan ilmunya.
7. Staf tata usaha Jurusan Matematika Universitas Hasanuddin yang telah memberikan pelayanan yang baik.
8. Kedua orang tua Penulis yang tercinta dan tersayang ayahanda Drs. H. Abdul Rahim dan ibunda Hj. Salmah, serta saudara-saudaraku yang tercinta Muallim, SS, Muzayidah, S.TP, Mushawwir Arman yang telah memberikan dukungan hingga selesainya skripsi ini.
9. Rekan-rekan mahasiswa sejurusan Matematika di Himpunan Mahasiswa Matematika Universitas Hasanuddin.
10. Rekan-rekan mahasiswa angkatan '97 Jurusan Matematika Universitas Hasanuddin khususnya kepada **Ardi, Fandy, Mahmud, Agus, Nana, Temma, Ade, Ati, Irna, Ophie,** dan yang tercinta adinda **Herlina Toaha,** serta rekan-rekan seangkatan yang tidak disebutkan namanya yang telah memberikan motivasi kepada Penulis.

Semoga budi baik dan bantuan yang telah disumbangkan ini mendapat imbalan yang berlipat ganda dari Allah SWT. Amin

Makassar, Juni 2002

PENULIS

ABSTRAK

Tulisan ini menguraikan tentang penyederhanaan fungsi Boolc dengan berdasarkan metode ***Quine-McCluskey***. Metode ini, yang dikenal sebagai 'metode tabulasi', bermanfaat pada penyederhanaan fungsi Boole dengan banyak variabel, misalnya lebih dari enam variabel.

Setiap ekspresi Boole bisa dinyatakan dalam bentuk baku yang disebut bentuk *jumlah hasilkali*. Dalam bentuk baku ini, sembarang ekspresi Boole disajikan melalui matriks Marquand, yang mirip dengan peta Karnaugh.

Sembarang ekspresi Boole bisa dinyatakan oleh beberapa sirkuit logika yang berbeda, tetapi ekuivalen. Dua sirkuit logika dikatakan ekuivalen jika dan hanya jika setiap masukan yang sama pada masing-masing sirkuit, memberikan luaran yang sama.

Metode ***Quine-McCluskey*** bertujuan untuk memperoleh sirkuit logika yang paling sederhana dari suatu ekspresi Boole. Metode ini menyederhanakan ekspresi Boole dalam bentuk jumlah hasilkali. Implikan utama dari jumlah ini kemudian diperiksa untuk melihat apakah beberapa dari implikan utama ini berlebihan, sehingga bisa lebih disederhanakan lagi.

ABSTRACT

This paper describes a Boolean function minimization based on ***the Quine-McCluskey Method***. The method, which is also known as the 'tabular method', is particularly useful when the functions have a large number of variables, e.g. six variables.

Each Boolean expression is represented by a standard form, called a ***sum of products***. In this form, any Boolean expression can be represented by a Marquand matrix, which is similar to Karnaugh map.

Any Boolean expression can be represented by different, but equivalent, logic circuits. Two logic circuits are equivalent if and only if the same inputs into each circuit will return the same outputs.

The ***Quine-McCluskey*** aims to find the simplest logic circuit of a Boolean expression. The method investigates any expression in the standard sum of product form and then examines if some prime implicants in the sum are redundant. If they are, then they will be reduced to simpler forms.

DAFTAR ISI

BAB I

PENDAHULUAN

I.1 LATAR BELAKANG

Konsep aljabar Boole pertama kali dikemukakan oleh George Boole pada tahun 1847. Konsep yang diperkenalkan dinamakan " *The Mathematical Analysis of Logic*". Boole menguraikan secara formal dan menggunakan pernyataan logika dalam kerangka aljabar.

Konsep aljabar Boole ini merupakan salah satu bentuk aplikasi matematika yang dapat digunakan untuk model rangkaian perlengkapan elektronik. Variabel-variabel yang dipakai dalam Boole hanya dapat mengambil satu harga dari dua harga yang mungkin diambilnya. Kedua harga ini dapat dipresentasikan dengan simbol 0 dan 1. Harga ini dapat menjadi output dan input dalam suatu rangkaian.

Rangkaian dapat dibangun dengan menggunakan elemen dasar yang memiliki dua kondisi yang berbeda. Elemen dasar ini berupa saklar yang dapat disusun dalam posisi "*on*" atau dalam posisi "*off*". Pada tahun 1938 Claude Shannon menunjukkan bagaimana aturan dasar dari logika digunakan untuk mendisain rangkaian. Aturan ini menjadi bentuk yang mendasari aljabar Boole. Operasi dari suatu rangkaian didefinisikan oleh suatu fungsi Boole yang mengkhususkan nilai output untuk setiap himpunan dari input.

Langkah pertama dalam membangun suatu rangkaian adalah bagaimana menyatakan ekspresi Boole melalui suatu rangkaian dan menggambarkan fungsi Boole dari rangkaian ini dengan suatu ungkapan yang dibangun dengan menggunakan operasi aljabar Boole. Suatu rangkaian dapat terdiri dari ribuan bahkan jutaan rangkaian. Suatu rangkaian dapat disederhanakan untuk memperoleh rangkaian yang paling sederhana. Oleh karena itu diperlukan suatu prosedur penyederhanaan jumlahan hasilkali perluasan.

Permasalahan yang dibahas adalah bagaimana penyederhanaan suatu rangkaian logika yang telah didisain dengan menggunakan suatu prosedur penyederhanaan perluasan jumlahan hasilkali (*sum of products expansion*) yang dikenal dengan nama **Metode Quine-McCluskey.**

Berdasarkan alasan-alasan di atas, maka penulis tertarik untuk membahas lebih mendalam masalah tersebut dan menuangkan dalam bentuk tulisan dengan judul :

"PENYEDERHANAAN SIRKUIT LOGIKA DENGAN METODE QUINE - McCLUSKEY"

Sekaligus sebagai tugas akhir untuk memenuhi salah satu syarat dalam meraih gelar sarjana pada Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin.

I.2 TUJUAN PENULISAN

Adapun tujuan dari penulisan skripsi ini sebagai berikut :

a. Menunjukkan dua ekspresi Boole yang berbeda tetapi ekivalen, dinyatakan melalui dua rangkaian (sirkuit) yang berbeda.
b. Menunjukkan metode Quine-McCluskey untuk mendapatkan rangkaian paling sederhana.
c. Mempelajari dan menunjukkan fasilitas operator-operator logika dalam program MapleV.

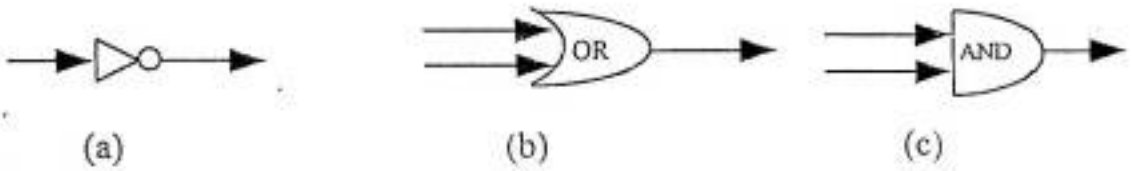
I.3 RUMUSAN / BATASAN MASALAH

Berdasarkan latar belakang, maksud dan tujuan penulisan, maka fokus utama dalam tulisan ini akan membahas bagaimana proses penyederhanaan jumlahan hasilkali Boole yang ditunjukkan oleh rangkaian dengan menggunakan metode Quine-McCluskey.

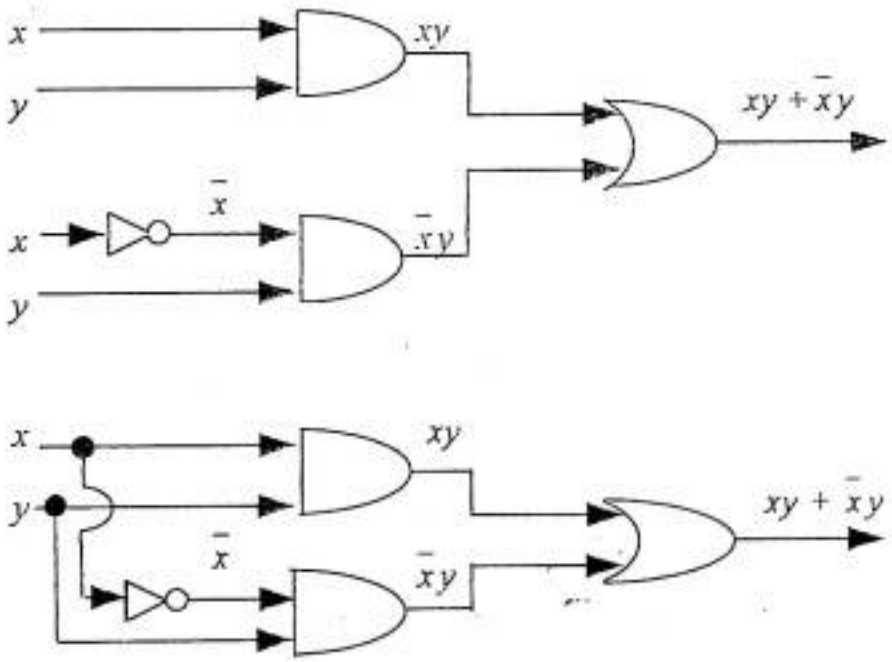
I.4 TINJAUAN PUSTAKA

Elemen dasar dari rangkaian dinamakan *gates* (gerbang). Setiap tipe dari gerbang menjalankan suatu operasi Boole. Dengan penggunaan gerbang ini, dapat diterapkan aturan aljabar Boole terhadap disain rangkaian. Rangkaian yang dipelajari diberikan output yang hanya bergantung pada input, dan tidak pada keadaan arus dari rangkaian. Dengan kata lain, rangkaian ini tidak mempunyai kapabilitas memori. Rangkaian demikian disebut rangkaian kombinasi atau *gating networks*.

Berdasarkan [5, hal. 604-605], rangkaian kombinasi akan dikonstruksi dengan menggunakan tiga tipe elemen. Pertama yaitu *inverter*, gerbang *OR*, gerbang *AND*. Illustrasi ketiga elemen tersebut dapat ditunjukkan seperti pada gambar 1.1.



Gambar 1.1

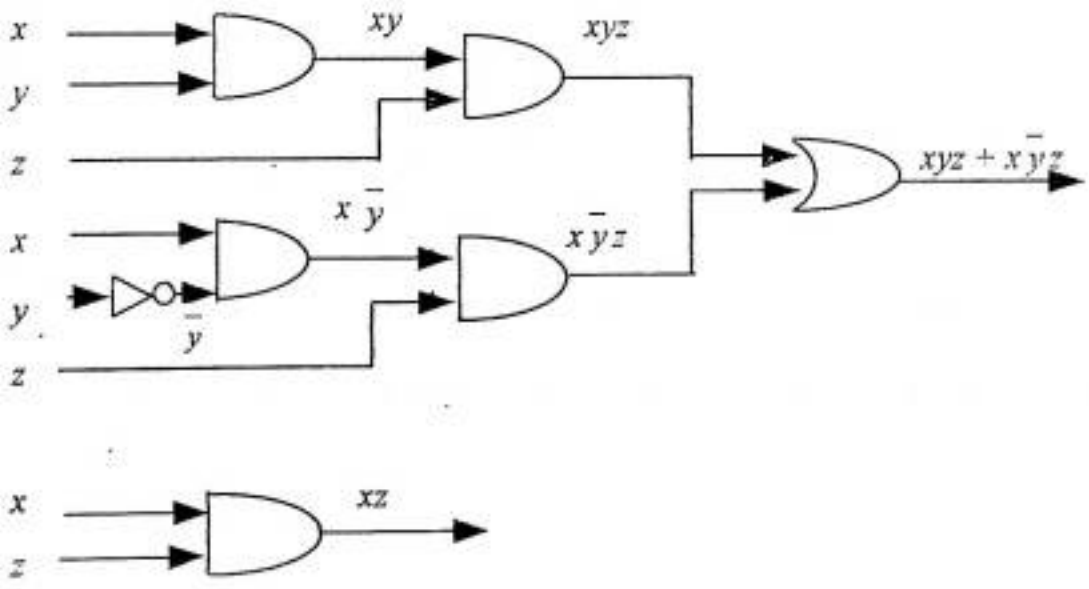


Gambar 1.2

Gambar 1.2 menggambarkan dua cara penggambaran gerbang dengan nilai input x dan y. Dalam hal ini juga menunjukkan bahwa output dari gerbang mungkin digunakan sebagai input oleh satu atau lebih elemen lainnya. Kedua gambar menunjukkan bahwa rangkaian menghasilkan output $xy + \bar{x}y$.

Misalkan suatu rangkaian seperti pada gambar 1.3. Rangkaian mempunyai output 1 jika dan hanya jika $x = y = z = 1$ atau $x = z = 1$ dan $y = 0$. Perluasan jumlahan hasilkali dari rangkaian ini adalah $xyz + x\bar{y}z$. Dua hasil perluasan ini berbeda tepat satu variabel yaitu y. Kombinasinya dapat dibentuk sebagai berikut :

$$\begin{aligned} xyz + x\bar{y}z &= (y + \bar{y})(xz) \\ &= 1 \cdot (xz) \\ &= xz \end{aligned}$$



Gambar 1.3

Metoda Quine-McCluskey digunakan untuk menyederhanakan rangkaian tersebut di atas yaitu menyatakan setiap *minterm* (didefinisikan pada bab III) dengan

suatu notasi biner dan kemudian mengelompokkan *term* ini sesuai index dalam notasi binernya. Seperti yang ditunjukkan pada tabel 1.1. Semua *Boolean products* dapat dibentuk seperti yang ditunjukkan pada daftar 2.

Tabel 1.1

Daftar 1		Daftar 2	
	x y z		*x y z*
(05)	1 0 1 [x]	(05,07)	1 - 1 [1]
(07)	1 1 1 [x]		

Tabel 1.2

Minterm	05	07
xz	**X**	**X**

Hasil yang tidak digunakan untuk membentuk hasil dalam variabel terkecil adalah *xz*. Dalam tabel 1.2 ditunjukkan bahwa hasil yang diperoleh terdapat dalam *minterm* yang akan disederhanakan dan *xz* merupakan implikan utama terpenting. Akibatnya *xz* adalah hasil terakhir dari penyederhanaan rangkaian, lihat [5, hal. 619 – 621]

I.5 SISTEMATIKA PEMBAHASAN

Adapun sistematika pembahasan dalam penulisan ini terdiri dari lima bab yaitu: BAB I merupakan bab pendahuluan yang menguraikan tentang latar belakang, batasan masalah, tujuan penulisan, tinjauan pustaka, sistématika pembahasan. BAB II merupakan bab pembahasan mengenai aljabar Boole antara lain aksioma aljabar Boole, teorema *de Morgan*, operasi aljabar Boole, dan dualitas. BAB III pembahasan mengenai fungsi Boole dan rangkaian logika. BAB IV berisi pembahasan tentang penyederhanaan rangkaian logika dengan metode Quine-McCluskey. BAB V merupakan bab penutup yang meliputi kesimpulan dari semua pembahasan serta saran-saran yang dianggap perlu untuk dikemukakan.

BAB II

ALJABAR BOOLE

II.1 ALJABAR BOOLE

Jika suatu variabel dipakai dalam rumus aljabar, biasanya dianggap bahwa variabel tersebut dapat mengambil setiap harga numerik. Misalnya dalam persamaan $2x - 5y = z$ dapat dianggap bahwa x, y, z dapat mengambil harga suatu bilangan real.

Variabel-variabel yang dipakai dalam aljabar Boole memiliki karakteristik yang khas, namun variabel tersebut hanya dapat mengambil satu harga dari dua harga yang mungkin diambilnya. Kedua harga ini dapat direpresentasikan dengan simbol 0 dan 1. konsep ini akan dijelaskan berikut ini.

Definisi 2.1 (Aljabar Boole)

Suatu aljabar Boole adalah suatu sistem aljabar $(B, +, \cdot, \bar{}\,; 0,1)$ *yang terdiri dari himpunan B, dua operasi biner* $\cdot$ *dan* $+$ *, dan suatu operasi uner* $-$ *sedemikian sehingga untuk* $x, y, z \in B$ *berlaku :*

(i)	$(x + y)$ *dan* $(x \cdot y)$ *di* B	*Klosur (Closure)*
(ii)	$x + 0 = x$	*Elemen identitas*
	$x \cdot 1 = x$	
(iii)	$(x + y) = (y + x)$	*Komutatif (Commutative)*
	$(x \cdot y) = (y \cdot x)$	

(*iv*) $(x+y)+z = x+(y+z)$ *Asosiatif (Assosiative)*

$(x\,.\,y)\,.\,z = x\,.\,(y\,.\,z)$

(*v*) $x+(y\,.\,z) = (x+y)\,.\,(x+z)$ *Distributif (Distributive)*

$x\cdot(y+z) = x\,.\,y + x\,.\,z$

(*vi*) $x+\bar{x}=1$ *Komplemen (Complement)*

$x\,.\,\bar{x}=0$

Teorema 2.1 (Idempoten)

Misalkan $(B, +, .\,, \bar{\ })$ aljabar Boole

Misalkan $x \in B$, maka

$$x+x=x$$

$$x\,.\,x=x$$

Bukti :

$x+x=(x+x).1$ *Identitas*

$=(x+x).(x+\bar{x})$ *Komplemen*

$=x+(x\,.\,\bar{x})$ *Distributif*

$=x+0$ *Komplemen*

$=x$ *Identitas*

dan,

$x\,.\,x=(x.x)+0$ *Identitas*

$=(x\,.x)+(x.\bar{x})$ *Komplemen*

$=x\,.\,(x+\bar{x})$ *Distributif*

$=x\,.\,1$ *Komplemen*

$=x$ *Identitas*

Teorema 2.2 (Operasi 0 dan 1)

Misalkan $(B, +, ., \bar{})$ aljabar Boole

Misalkan $x \in B$, maka

$$x + 1 = 1$$

$$x \,.\, 0 = 0$$

Bukti :

$x + 1 = x + (x + \bar{x})$	*Komplemen*
$= (x + x) + \bar{x}$	*Asosiatif*
$= x + \bar{x}$	*Idempotent*
$= 1$	*Komplemen*

Bahwa $x \,.\, 0 = 0$ merupakan akibat dari prinsip dualitas.

Teorema 2.3

Misalkan $(B, +, ., \bar{})$ aljabar Boole

Misalkan $x, y \in B$, maka

$$y \,.\, x + x = x$$

$$(y + x) \,.\, x = x$$

Bukti :

$y.x + x = y.x + x.1$	*Identitas*
$= y.x + 1.x$	*Komutatif*
$= (y+1).x$	*Distributif*
$= 1.x$	*Operasi* 0 *dan* 1
$= x$	*Identitas*

II.2 TEOREMA *DE MORGAN*

Teorema 2.4 (*De Morgan*)

Misalkan $B = \{0,1\}$

untuk sembarang x dan y di B, berlaku :

$$\overline{x+y} = \bar{x}.\bar{y}$$

$$\overline{x.y} = \bar{x} + \bar{y}$$

Bukti :

Perhatikan bahwa

$$\begin{aligned}(x+y)+(\bar{x}.\bar{y}) &= [(x+y)+\bar{x}]\,[(x+y)+\bar{y}] \\ &= [(x+\bar{x})+y]\,[x+(y+\bar{y})] \\ &= [1+y]\,[x+1] \\ &= 1.1 \\ &= 1\end{aligned}$$

dan,

$$(x+y).(\bar{x}.\bar{y}) = [(x.(\bar{x}.\bar{y})] + [y.(\bar{x}.\bar{y})]$$

$$= [(x.\bar{x}).\bar{y}] + [(y.\bar{y}).\bar{x}]$$

$$= [0.\bar{y}] + [0.\bar{x}]$$

$$= 0 + 0$$

$$= 0$$

Jadi $\bar{x}.\bar{y}$ adalah komplemen bagi $x+y$ artinya,

$$\overline{x+y} = \bar{x}.\bar{y} \qquad (2.2.1)$$

Bahwa $\overline{x.y} = \bar{x} + \bar{y}$ merupakan akibat dari prinsip dualitas.

Komplemen setiap ungkapan Boole atau sebagian dari setiap ungkapan dapat ditentukan dengan menggunakan teorema ini. Dalam aturan ini, dua langkah dipakai untuk mencari komplemen :

1. Simbol " + " diganti dengan simbol " • " atau sebaliknya.
2. Setiap suku dalam ungkapan diambil komplemennya.

Penggunaan teorema *De Morgan* dapat ditunjukkan dengan mencari komplemen dari ungkapan

$$(x+y.z) \qquad (2.2.2)$$

Pertama perlu ditunjukkan bahwa tanda kali dihilangkan ungkapan dapat ditulis sebagai

$$x + (yz) \qquad (2.2.3)$$

Untuk mengambil komplemennya, simbol " + " diganti dengan simbol " • " dan kedua suku itu dikomplementasikan, sehingga diperoleh :

$$\bar{x} \cdot (\overline{yz}) \qquad (2.2.4)$$

Kemudian suku terakhir dikomplementasikan, menghasilkan :

$$\bar{x} \cdot (\bar{y} + \bar{z}) \qquad (2.2.5)$$

Maka diperoleh,

$$\overline{(x+yz)} = \bar{x} \cdot (\bar{y} + \bar{z}) \qquad (2.2.6)$$

II.3 OPERASI ALJABAR BOOLE

Dalam operasi penjumlahan dan perkalian bilangan umumnya digunakan operasi biner untuk menentukan solusinya. Misalnya, persamaan paling sederhana adalah $a + x = b$ untuk operasi penjumlahan dan $a \cdot b$ untuk operasi perkalian.

Demikian pula untuk membahas aljabar Boole digunakan operasi biner $\wedge$ dan operasi biner $\vee$, dan sebuah operasi uner $'$. Konsep ini akan menjadi jelas jika didefinisikan atas himpunan B. Operasi $\wedge$ dapat direpresentasikan sebagai operasi perkalian logis dengan simbol "." dan operasi $\vee$ dapat direpresentasikan sebagai operasi penjumlahan logis dengan simbol "+".

Definisi 2.2 (Jumlahan Boole)

Misal F dan G fungsi Boole berderajat n

Jumlahan Boole F + G didefinisikan sebagai :

$$(F+G)(x_1, \ldots, x_n) = F(x_1, \ldots, x_n) + G(x_1, \ldots, x_n)$$

***Definisi 2.3* (*Hasilkali Boole*)**

Misal F dan G fungsi Boole berderajat n

Hasilkali Boole F . G didefinisikan sebagai :

$$(F \cdot G)(x_1, \ldots, x_n) = F(x_1, \ldots, x_n) \cdot G(x_1, \ldots, x_n)$$

***Definisi 2.4* (*Komplemen*)**

Misal F fungsi Boole berderajat n

Komplemen dari fungsi Boole F adalah $\overline{F}$ *, didefinisikan sebagai :*

$$\overline{F}(x_1, \ldots, x_n) = \overline{F(x_1, \ldots, x_n)}$$

Definisi 2.5

Misalkan $B = \{0,1\}$

Misalkan F dan G fungsi Boole

Fungsi Boole F dan G dari n variabel ekivalen jika hanya jika :

$$F(b_1, b_2, b_3, \ldots, b_n) = G(b_1, b_2, b_3, \ldots, b_n) \; : b_1, b_2, b_3, \ldots \; di\; B$$

Teorema 2.5

Dalam sembarang aljabar Boole, $\overline{0} = 1$.

Bukti :

Akan ditunjukkan bahwa $0 + 1 = 1$ dan $0 \cdot 1 = 0$

Menurut definisi, persamaan pertama adalah hukum identitas $0 + x = x$ untuk $x = 1$.

Dan persamaan kedua adalah hukum identitas $x \cdot 1 = x$ untuk $x = 0$.

Teorema 2.6

Dalam Aljabar Boole, jika suatu unsur memiliki komplemen, maka komplemen ini tunggal.

Bukti :

Misalkan suatu unsur x memiliki dua komplemen y dan z. Artinya

$$x \vee y = 1 \qquad x \wedge y = 0$$

$$x \vee z = 1 \qquad x \wedge z = 0$$

Maka diperoleh

$$\begin{aligned} y &= y \wedge 1 \\ &= y \wedge (x \vee z) \\ &= (y \wedge x) \vee (y \wedge z) \\ &= 0 \vee (y \wedge z) \\ &= (x \wedge z) \vee (y \wedge z) \\ &= (x \vee y) \wedge z \\ &= 1 \wedge z \\ &= z \end{aligned}$$

II.4 DUALITAS

Prinsip dualitas merupakan suatu konsep penting yang banyak dijumpai diberbagai tempat dan situasi. Dalam identitas aljabar Boole merupakan suatu pasangan (khusus untuk double komplemen). Untuk menjelaskan antara dua identitas dalam setiap pasangan digunakan konsep dualitas. Dualitas dari ekspresi Boole diperoleh dari pertukaran antara jumlahan Boole dan hasil kali Boole dan pertukaran nilai 0 dan 1.

Definisi 2.6 (Dualitas)

Dualitas dari ekspresi Boole adalah ekspresi yang diperoleh dari pertukaran jumlahan Boole dengan hasilkali Boole dan pertukaran nilai 0 *dengan* 1.

Contoh :

Misalkan suatu ekspresi Boole

$$x \,.\, (y + 0) \text{ , dan}$$

$$\bar{x} \,.\, 1 + (y + z) \qquad (2.4.1)$$

Kedua ekspresi Boole tersebut dapat ditentukan dualitasnya dengan cara mempertukarkan tanda "+" dan tanda " . " , nilai 0 dan 1 pada ekspresi ini.

Sehingga diperoleh

$$x + (y \,.\, 1), \text{ dan}$$

$$\bar{x} + 0 \,.\, (y \,.\, z) \qquad (2.4.2)$$

BAB III

FUNGSI BOOLE DAN RANGKAIAN LOGIKA

III.1 EKSPRESI BOOLE DAN FUNGSI BOOLE

***Definisi 3.1* (*Variabel Boole*)**

Misalkan $B = \{0,1\}$

Variabel x disebut variabel Boole jika menerima nilai hanya dari B.

***Definisi 3.2* (*Literal/Minterm*)**

Literal adalah variabel Boole atau komplemennya. Minterm dari variabel Boole x_1 , x_2 , x_3 ,....,x_n *adalah hasilkali Boole* y_1 , y_2 , y_3 ,...., y_n *dimana* $y_i = x_i$ *atau* $y_i = \bar{x}_i$.

Minterm merupakan hasilkali dari n literal dengan satu literal untuk setiap variabel.

Contoh :

Suatu *minterm* bernilai 1 untuk satu dan hanya satu kombinasi dari nilai variabelnya. *Minterm* y_1 , y_2 , y_3 ,...., y_n bernilai satu jika hanya jika setiap y_i bernilai 1 dan ini terjadi jika hanya jika $x_i = 1$ ketika $y_i = x_i$ dan $x_i = 0$ ketika $y_i = \bar{x}_i$. *Minterm* $\bar{x}_1$ x_2 $\bar{x}_3$ x_4 x_5 bernilai 1 jika $x_1 = x_3 = 0$ dan $x_2 = x_4 = x_5 = 1$ dan sama dengan 0 untuk yang lainnya.

Definisi 3.3 (Ekspresi Boole)

Ekspresi Boole dalam variabel $x_1, x_2, x_3, ..., x_n$ *didefinisikan berlaku :*

$0, 1, x_1, x_2, x_3, ..., x_n$ *adalah ekspresi Boole*

Jika E_1 *dan* E_2 *adalah ekspresi Boole, maka* E_1, $(E_1\ E_2)$, *dan* $(E_1 + E_2)$ *adalah ekspresi Boole*

Contoh :

Misalkan $E(x_1, x_2, ..., x_n)$ adalah sebuah ekspresi Boole n peubah di dalam aljabar Boole $(B, +, ., \bar{})$. Yang dimaksud dengan pemberian nilai pada peubah-peubah $x_1, x_2, ..., x_n$ ialah pemberian unsur-unsur di dalam B untuk menjadi nilai-nilai bagi peubah-peubah tersebut. Untuk pemberian nilai kepada peubah-peubah, dapat dievaluasi ekspresi $E(x_1, x_2, ..., x_n)$ dengan cara mensubstitusikan peubah-peubah di dalam ekspresi itu dengan nilai-nilai mereka. Misalkan, bagi ekspresi Boole

$$E(x_1, x_2, ..., x_n) = (x_1 + x_2)(\overline{x_1} + \overline{x_2})(\overline{x_2 + x_3}) \qquad (3.1.1)$$

di dalam aljabar Boole $(\{0,1\}, +, ., \bar{})$, pemberian nilai $x_1 = 0$, $x_2 = 1$, $x_3 = 0$ menghasilkan

$$E(0,1,0) = (0 + 1)(\overline{0} + \overline{1})(\overline{1+0})$$

$$= 1 . 1 . 0$$

$$= 0$$

***Definisi 3.4* (*Fungsi Boole*)**

Misal $x_1, x_2, ..., x_n$ *sebagai variabel pada suatu aljabar Boole B. Suatu pemetaan F dari B kepada dirinya sendiri adalah suatu fungsi Boole n variabel, dinyatakan dengan* $F(x_1, x_2, ..., x_n)$, *bila dapat disusun menurut aturan-aturan sebagai berikut*

1. *Bila* $F(x_1, x_2, ..., x_n)$ *adalah suatu fungsi Boole, maka* $\overline{F[x_1 x_2 ... x_n]}$ *adalah suatu fungsi Boole.*
2. *Bila* $F_1(x_1, x_2, ..., x_n)$ *dan* $F_2(x_1, x_2, ..., x_n)$ *adalah fungsi Boole, maka*

$$F_1(x_1, x_2, ..., x_n) + F_2(x_1, x_2, ..., x_n) \text{ dan}$$

$$F_1(x_1, x_2, ..., x_n) \cdot F_2(x_1, x_2, ..., x_n)$$

adalah fungsi Boole.

Untuk melihat bagaimana menspesifikasikan suatu fungsi dari B^n ke B melalui sebuah ekspresi Boole $E(x_1, x_2, ..., x_n)$ yaitu misalkan setiap pemberian nilai-nilai kepada peubah-peubah $x_1, x_2, ..., x_n$ merupakan suatu pasangan terurut ganda-n di dalam daerah asal (domain) B^n, dan misalkan nilai ekspresi itu, $E(x_1, x_2, ..., x_n)$ adalah bayangannya di dalam daerah hasil (range) B. Misalkan, ekspresi Boole

$$(\overline{x_1} x_2 \overline{x_3}) + (x_1 \overline{x_2}) + (x_1 x_3) \qquad (3.1.2)$$

di dalam aljabar Boole $(\{0,1\}, +, ., \bar{})$ mendefinisikan suatu fungsi.

Teorema 3.1

Ada 2^{2^n} pemetaan yang berlainan yang memetakan dari himpunan hasilkali Kartesian $\{0,1\}^n$ ke dalam $\{0,1\}$.

Bukti :

Dalam wilayah (domain)nya, terdapat $2 \times 2 \times 2 \times \ldots \times 2 = 2^n$ elemen. Karena masing-masing dari 2^n elemen ini dapat mempunyai salah satu dari kedua nilai 0 dan 1 dengan tidak saling bergantungan, maka ada

$$\underbrace{2 \times 2 \times \ldots \times 2}_{2^n} = 2^{2^n} \qquad (3.1.3)$$

pemetaan yang berlainan.

Dalam fungsi Boole berderajat n terdapat 2^n ganda-n dari 0 dan 1 yang berbeda. Dengan demikian fungsi Boole merupakan penempatan 0 dan 1 ke setiap 2^n ganda-n yang berbeda, sehingga menurut aturan perkalian terdapat 2^{2^n} fungsi Boole yang berbeda.

Contoh :

Suatu fungsi Boole berderajat 2 adalah fungsi dari suatu himpunan dengan empat elemen yaitu pasangan dari elemen $B = \{0,1\}$ ke B, sehingga terdapat $2^{2^2} = 2^4 = 16$ fungsi Boole berderajat 2 yang berbeda.

III.2 RANGKAIAN LOGIKA

Aljabar Boole digunakan untuk rangkaian dari perlengkapan elektronik. Setiap input dari perlengkapan tersebut dapat dipresentasikan sebagai suatu anggota dari $B = \{0,1\}$. Setiap rangkaian dapat didesain dengan menggunakan aturan dari aljabar Boole. Elemen dasar dari suatu rangkaian dinamakan gerbang (*gate*). Setiap tipe dari gerbang tersebut menjalankan suatu operasi Boole.

Operasi + dan operasi - secara fisis direalisasikan dengan memakai dua jenis rangkaian elektronik yang disebut gerbang *OR* dan gerbang *AND*.

***Definisi 3.5*(*Gerbang*)**

Suatu rangkaian elektronik yang beroperasi atas dasar satu atau lebih signal input untuk menghasilkan signal output disebut gerbang (gate).

***Definisi 3.6* (*Gerbang OR*)**

Suatu rangkaian elektronik yang menerima nilai dari dua atau lebih variabel Boole sebagai input dan jumlahan Boolenya sebagai output disebut gerbang OR.

Secara skematis gerbang *OR* digambarkan sebagai berikut :

Gambar 3.1

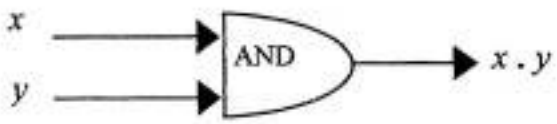
Output dari gerbang *OR* didefinisikan dengan input x dan y, $x + y$ sedemikian sehingga

$$x + y = \begin{cases} 1 & ; jika\ x = 1\ atau\ y = 1 \\ 0 & ; x, y\ yang\ lain \end{cases}$$

***Definisi 3.7* (*Gerbang AND*)**

Rangkaian elektronik yang menerima nilai dari dua atau lebih variabel Boole sebagai input dan hasilkali Boolenya sebagai output disebut gerbang AND.

Secara skematis gerbang *AND* digambarkan sebagai berikut :



Gambar 3.2

Output dari gerbang *AND* didefinisikan oleh x, y sedemikian sehingga nilai dari $x \cdot y$ diberikan :

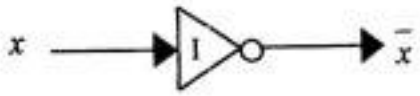
$$x \cdot y = \begin{cases} 1 & ; jika\ x = 1\ dan\ y = 1 \\ 0 & ; x, y\ yang\ lain \end{cases}$$

Operasi komplementasi secara fisis direalisasikan oleh sebuah gerbang atau rangkaian yang disebut *inverter* (gerbang *NOT*).

***Definisi 3.8* (*Gerbang inverter*)**

Rangkaian elektronik yang menerima nilai dari suatu variabel Boole sebagai input dan menghasilkan komplemennya sebagai output disebut gerbang inverter.

Secara skematis gerbang *inverter* digambarkan sebagai berikut :



Gambar 3.3

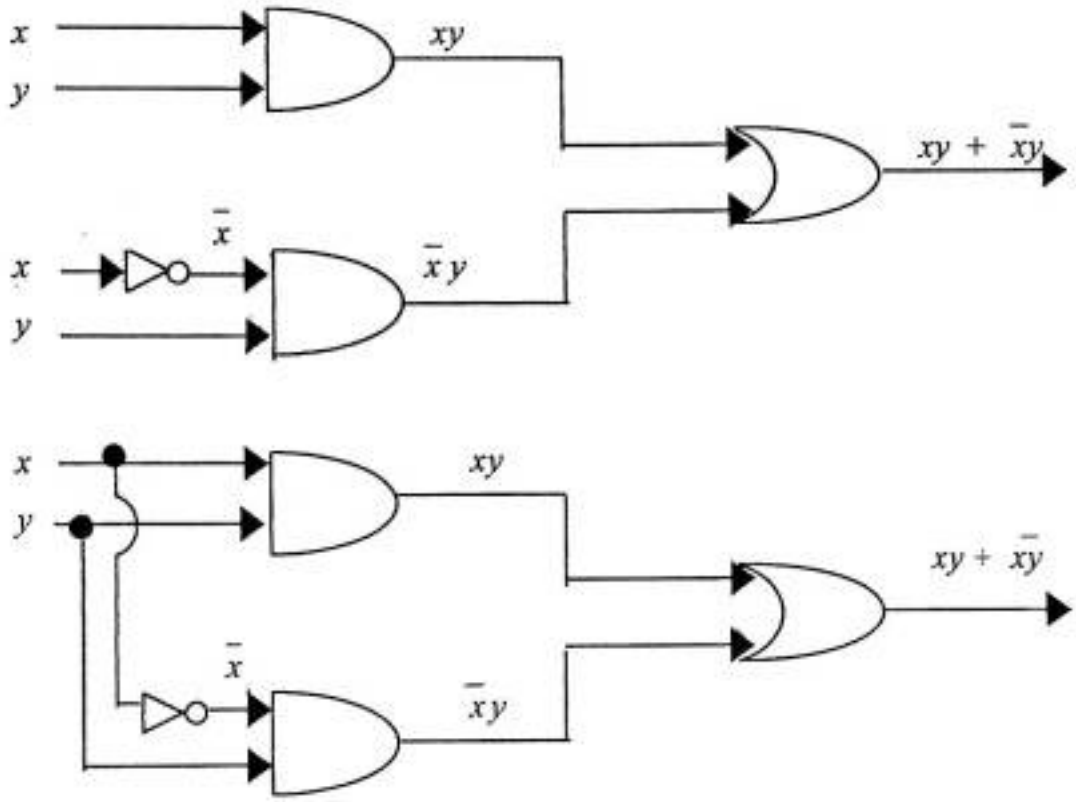
Output dari gerbang *inverter* didefinisikan sebagai :

$$\bar{x} = \begin{cases} 1 & ; jika\ x = 0 \\ 0 & ; jika\ x = 1 \end{cases}$$

III.3 KOMBINASI RANGKAIAN

Rangkaian kombinasi dapat dibangun dengan menggunakan suatu kombinasi dari gerbang *OR*, gerbang *AND*, dan gerbang *inverter*. Ketika kombinasi dari rangkaian dibentuk, beberapa gerbang dapat menjadi suatu input dari rangkaian lainnya. Gerbang *OR*, gerbang *AND*, dan gerbang *inverter* dapat dihubungkan untuk membentuk jaringan penggerbangan atas rangkaian logika.

Misalkan suatu rangkaian seperti pada gambar :



Gambar 3.4

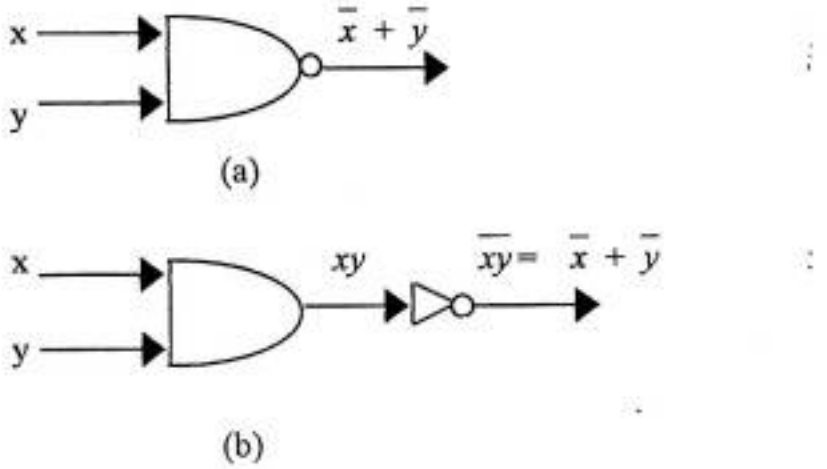
Gambar 3.4 menunjukkan dua cara penggambaran gerbang dengan nilai input yang sama. Dalam hal ini dapat ditunjukkan bahwa output dari suatu gerbang dapat digunakan sebagai input oleh satu atau lebih gerbang lainnya. Kedua gambar menunjukkan bahwa rangkaian menghasilkan output yang sama yaitu $xy + \bar{x}y$.

III.3.1 GERBANG *NAND*

Selain rangkaian logika dapat berupa gerbang *OR*, gerbang *AND*, dan gerbang *inverter*, terdapat pula gerbang yang merupakan kombinasi dari gerbang-gerbang tersebut. Gerbang ini antara lain : gerbang *NAND* dan gerbang *NOR*.

Gerbang *NAND* merupakan gerbang yang operasinya dapat menggunakan kombinasi gerbang *AND* yang diikuti oleh gerbang *inverter*.

Secara skematis gerbang *NAND* diperlihatkan dalam gambar berikut :



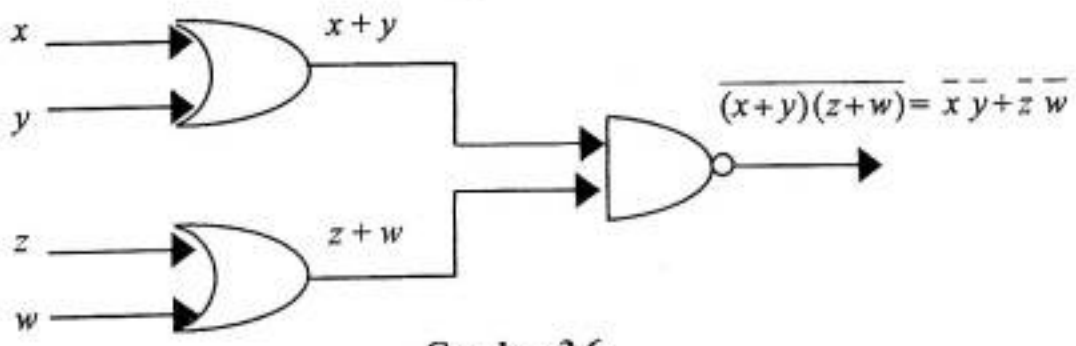
Gambar 3.5

Operasi gerbang (a) dapat dianalisa dengan menggunakan kombinasi gerbang *AND* yang diikuti oleh gerbang *inverter* seperti pada gambar 3.5 (b).

Jika salah satu saluran input gerbang *NAND* berisi dua masukan, misalkan

$$(x + y) \text{ dan } (z + w) \tag{3.3.1}$$

seperti yang ditunjukkan dalam gambar berikut :



Gambar 3.6

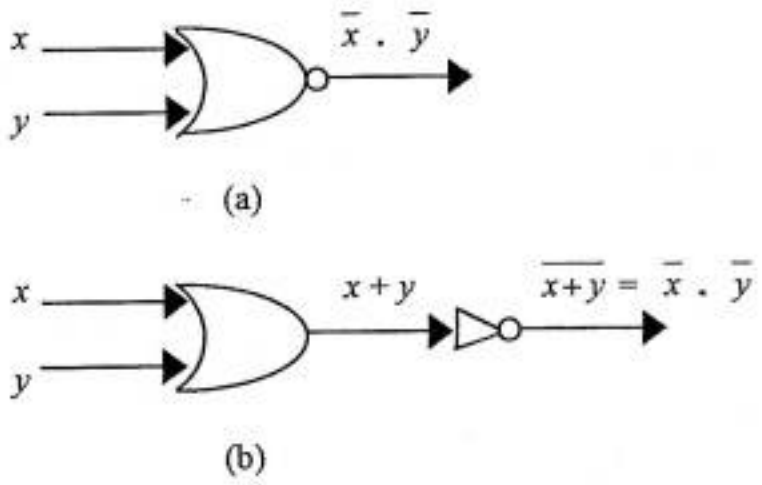
maka output gerbang *NAND* diperoleh

$$\overline{(x+y)(z+w)} = \bar{x}\,\bar{y} + \bar{z}\,\bar{w} \qquad (3.3.2)$$

III.3.2 GERBANG *NOR*

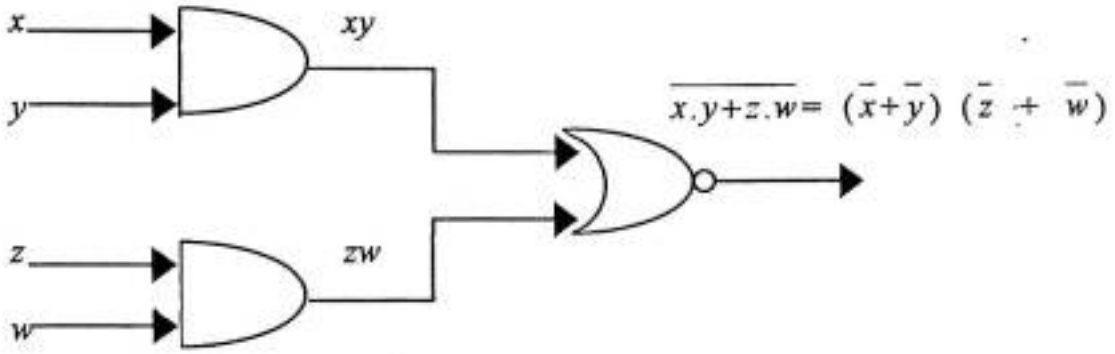
Gerbang *NOR* merupakan gerbang yang operasinya dapat menggunakan kombinasi dua buah gerbang yaitu gerbang *OR* dan gerbang *inverter*. Gambar 3.7 (a) menunjukkan gerbang *NOR*, dan kombinasinya yang dibentuk dari gerbang *OR* dan gerbang *inverter* ditunjukkan dalam gambar 3.7 (b).

Secara skematis gerbang *NOR* dapat diperlihatkan dalam gambar berikut :



Gambar 3.7

Jika salah satu saluran input dari gerbang *NOR* berisi dua input, misalkan x . y dan z . w seperti pada gambar berikut



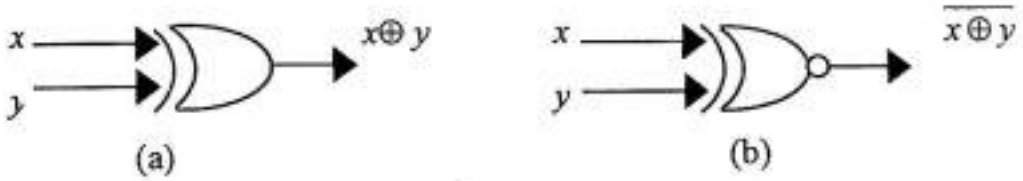
Gambar 3.8

maka output dari rangkaian diperoleh

$$\overline{x.y+z.w} = (\bar{x}+\bar{y})\,(\bar{z}+\bar{w}) \qquad (3.3.3)$$

III.3.3 GERBANG *EXCLUSIVE-OR*

Secara skematis gerbang *Exclusive-OR* ditunjukkan dalam gambar berikut



Gambar 3.9

Gambar 3.9(a) menunjukkan gerbang *Exclusive-OR*. Output dari gerbang ini adalah 1 jika dan hanya jika salah satu dari kedua inputnya bernilai 1, tetapi tidak kedua-duanya. Jika gerbang *Exclusive-OR* dirangkaikan dengan gerbang *inverter* maka akan diperoleh suatu gerbang baru yaitu gerbang *Exclusive-NOR* yang

ditunjukkan dalam gambar 3.9(b). Gerbang *Exclusive-NOR* memberikan output 1 ketika kedua inputnya 0 atau kedua inputnya 1.

III.4 PENAMBAH (*ADDERS*)

Misalkan penjumlahan dari dua digit ditunjukkan sebagai berikut :

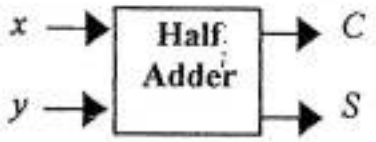
$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 10$$

Penjumlahan dari dua digit tunggal dapat memberikan hasil bilangan dua digit. Dengan demikian, jika dibangun suatu rangkaian untuk menggambarkan fungsi ini maka rangkaian yang didesain disebut *half adder* (penambah setengah). Operasi dari suatu rangkaian setengah penambah adalah menambahkan dua bit. Rangkaian setengah penambah mempunyai dua input dan dua output. Dua input tersebut adalah x dan y, sedangkan dua outputnya adalah jumlah S (*the sum*) dari x dan y, dan suatu bit muatan (*the carry bit*) yang dinyatakan dengan C. Jumlah S dari x dan y dapat bernilai 0 dan 1 seperti yang diperlihatkan pada gambar 3.10 (b). pada gambar yang sama terlihat bahwa bit muatan C adalah 1 hanya jika x dan y keduanya bernilai 1. Dari tabel diperoleh

$$C = xy \tag{3.4.1}$$

$$S = \bar{x}y + x\bar{y} \tag{3.4.2}$$

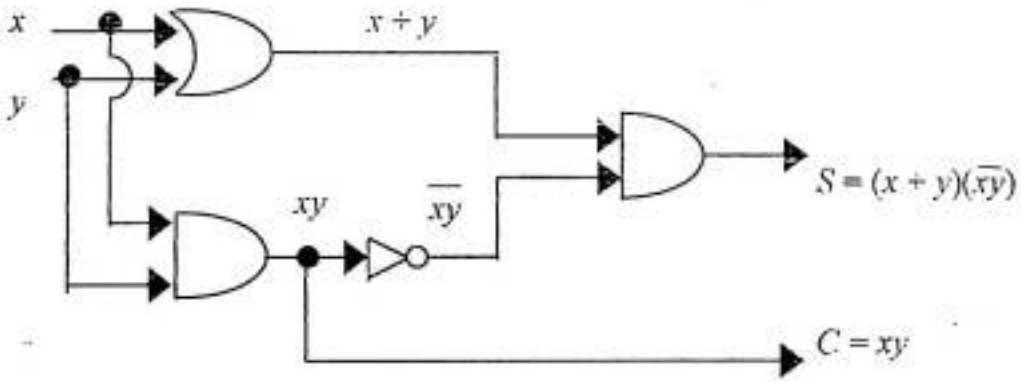
Suatu perwujudan gerbang *AND-OR* dari *half* adder (penambah setengah) diperlihatkan pada gambar 3.10 (c) dan diagram bloknya pada gambar 3.10 (a).



(a)

x	y	C	S
1	1	1	0
1	0	0	1
0	1	0	1
0	0	0	0

(b)

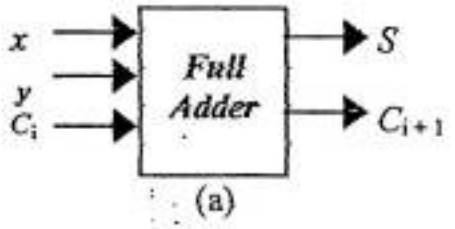


(c)

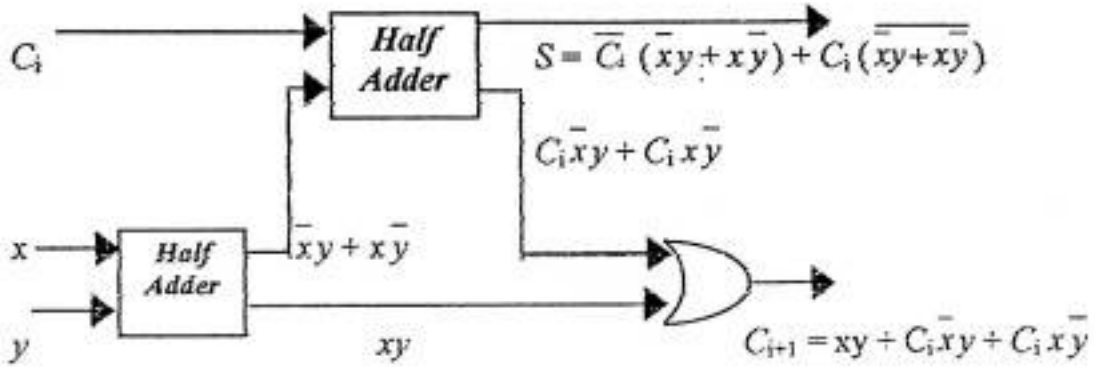
Gambar 3.10

Selanjutnya, pada rangkaian penambah (*adder*) diperlihatkan satu masukan lagi menjadi input ketiga. Suatu penambah yang mempunyai tiga input adalah penambah penuh (*full adder*) seperti yang ditunjukkan pada gambar 3.11 (a) di mana C_i menyatakan muatan dari penambah sebelumnya dan C_{i+1} menyatakan muatan (*the carry*) yang baru.

Rangkaian *full adder* ditunjukkan dalam gambar berikut:



(a)



(b)

Gambar 3.11

Dua output dari rangkaian *full adder* diberikan sebagai berikut :

$$C_{i+1} = xy + \bar{x}yC_i + x\bar{y}C_i \qquad (3.4.3)$$

$$S = \bar{C}_i(\bar{x}y + x\bar{y}) + C_i(\overline{\bar{x}y + x\bar{y}})$$

$$= \bar{x}\,\bar{y}C_i + \bar{x}y\bar{C}_i + xyC_i + x\bar{y}\,\bar{C}_i \qquad (3.4.4)$$

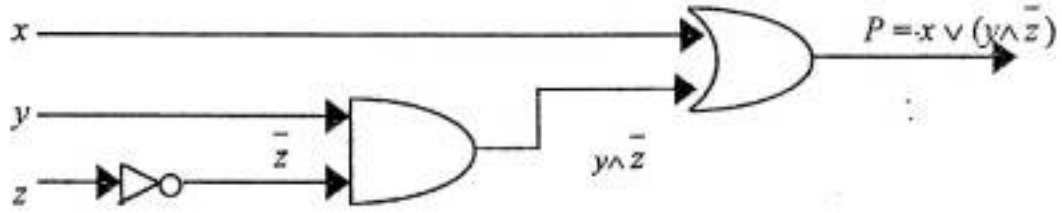
III.5 APLIKASI RANGKAIAN LOGIKA

Misalkan akan dirancang suatu rangkaian elektronik yang akan membunyikan alarm di dalam mobil bila suhu mesin telah melebihi 200^{o} F atau jika gigi perseneling mobil telah dimasukkan dan pengemudi belum mengenakan sabuk pengaman. Jelas bahwa akan diperoleh relasi :

$$P \equiv x \vee (y \wedge \bar{z}) \qquad (3.5.1)$$

dalam hal ini P adalah pernyataan " Rangkaian elektronik mengeluarkan bunyi," x adalah pernyataan " suhu melebihi 200^{o} F," y adalah pernyataan "gigi perseneling mobil telah dimasukkan," z adalah pernyataan "sabuk pengaman telah dikenakan." Untuk menyusun suatu rangkaian elektronik yang akan berprilaku P, pertama-tama pernyataan direpresentasikan dengan sinyal listrik. Misalnya, merepresentasikan pernyataan dengan suatu tegangan listrik. Jika pernyataan itu benar, akan direpresentasikan dengan suatu tegangan tinggi (misalnya $6V$), dan jika pernyataan itu salah, akan direpresentasikan oleh suatu tegangan rendah (misalnya $0V$). Jadi, dalam aplikasi tersebut, berpadanan dengan pernyataan x, suatu sinyal tegangan tinggi akan muncul jika suhu mesin melebihi 200^{o} F, dan suatu sinyal tegangan rendah akan muncul jika suhu mesin tidak melebihi 200^{o} F. Demikian pula, berpadangan dengan pernyataan P, suatu sinyal tegangan tinggi, yang dapat digunakan untuk membunyikan alarm, akan muncul jika syarat-syarat untuk membunyikan alarm dipenuhi, dan suatu sinyal tegangan rendah, yang tidak cukup untuk membunyikan

alarm, akan muncul jika syarat-syarat untuk membunyikan alarm tidak dipenuhi. Rangkaian ini dapat digambarkan sebagai berikut :



Gambar 3.12

BAB IV

PENYEDERHANAAN RANGKAIAN LOGIKA DENGAN METODE QUINE-McCLUSKEY

IV.1 PENYEDERHANAAN RANGKAIAN

Suatu rangkaian elektronik dapat tersusun atas ribuan bahkan jutaan rangkaian yang dibangun dari kombinasi gerbang-gerbang logika. Efisiensi dari suatu rangkaian kombinasi bergantung susunan dari gerbang dalam suatu rangkaian. Proses desain suatu rangkaian kombinasi diawali dengan menentukan tabel output untuk setiap kombinasi dari nilai input. Untuk menentukan anggota dari gerbang logika yang ditunjukkan oleh suatu rangkaian akan digunakan perluasan jumlahan hasilkali (*sum of products expansion*) dari suatu rangkaian. Penyederhanaan rangkaian dapat dikatakan mencari bentuk minimal dari fungsi penyambungan yang diberikan. Pada bagian ini akan dibahas mengenai prosedur penyederhanaan rangkaian dengan menggunakan metode Quine-McCluskey. Dan sebagai tambahan akan diuraikan tentang metode lain yaitu metode peta Karnaugh.

***Definisi 4.1* (*Sum of Products Expansion*)**

Misalkan $B = \{0,1\}$

Suatu ekspresi Boole di dalam $(B, -, \cdot, \bar{})$ *dikatakan berada dalam bentuk normal disjungtif jika merupakan suatu sum beberapa minterm* (*sum of products expansion*)

Contoh :

Misalkan x, y, z ekspresi Boole

maka

$$F(x, y, z) = x\,y\bar{z} + x\bar{y}\,\bar{z} + \bar{x}\,y\bar{z} \qquad (4.1.1)$$

merupakan suatu ekspresi Boole dalam bentuk normal disjungtif dengan *minterm* $x\,y\bar{z}$, $x\bar{y}\,\bar{z}$, $\bar{x}\,y\bar{z}$.

Definisi 4.2 (Product of Sums Expansion)

Misalkan $B = \{0,1\}$

Suatu ekspresi Boole di dalam $(B, -, \cdot, \bar{})$ *dikatakan berada dalam bentuk normal konjungtif jika merupakan suatu product beberapa maxterm (product of sums expansion)*

Contoh :

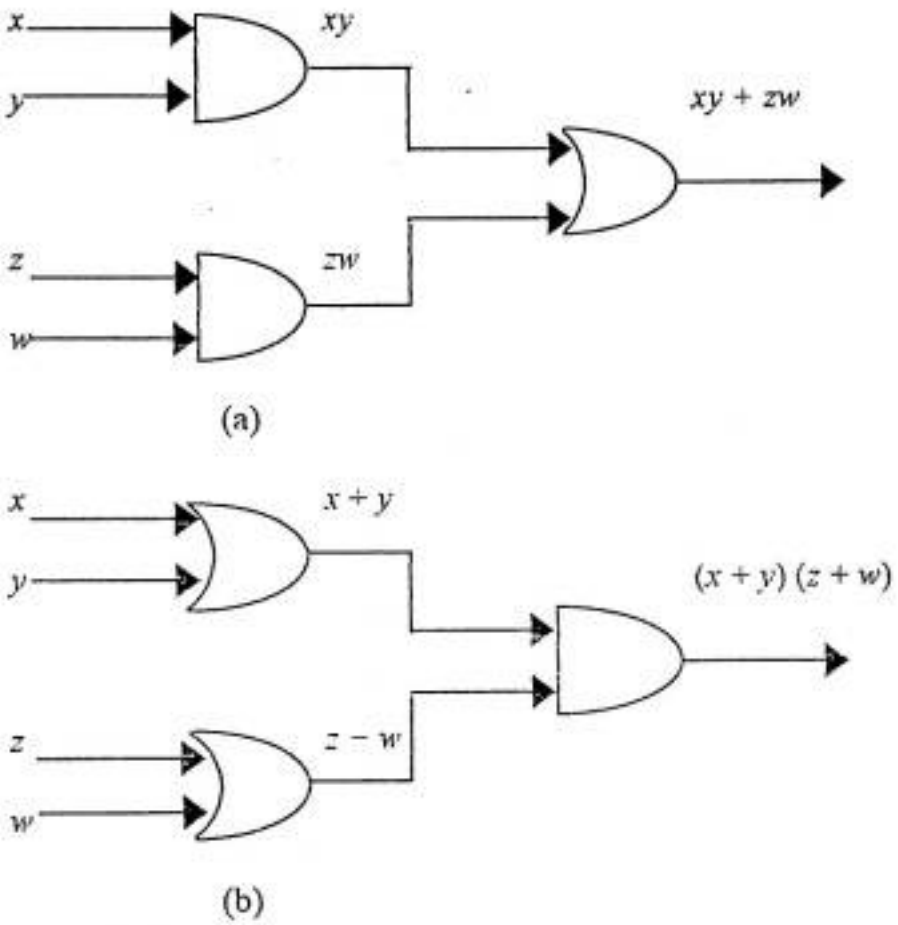
Misalkan x, y, z ekspresi Boole

maka

$$F(x, y, z) = (x + y + \bar{z})\,(x + \bar{y} + \bar{z}) \qquad (4.1.2)$$

merupakan suatu ekspresi Boole dalam bentuk normal konjungtif dengan *maxterm* $(x + y + \bar{z}), (x + \bar{y} + \bar{z})$.

Misalkan suatu rangkaian seperti gambar berikut :



Gambar 4.1

Gambar 4.1 menunjukkan 2 buah jaringan penggerbangan. Gambar 4.1(a) menunjukkan jaringan jumlahan hasilkali dan gambar 4.1(b) menunjukkan jaringan hasilkali jumlahan. Jaringan penggerbangan untuk ungkapan jumlahan hasilkali dalam bentuk "konvensional" adalah ungkapan yang mengandung paling sedikit dua hasilkali dan dengan paling sedikit dua variabel dalam setiap suku hasilkali akan menjadi jaringan gerbang AND ke gerbang OR. Sedangkan ungkapan jumlahan akan

menjadi jaringan gerbang *OR* ke gerbang *AND* seperti yang ditunjukkan pada gambar.

IV.1.1 METODE PETA KARNAUGH

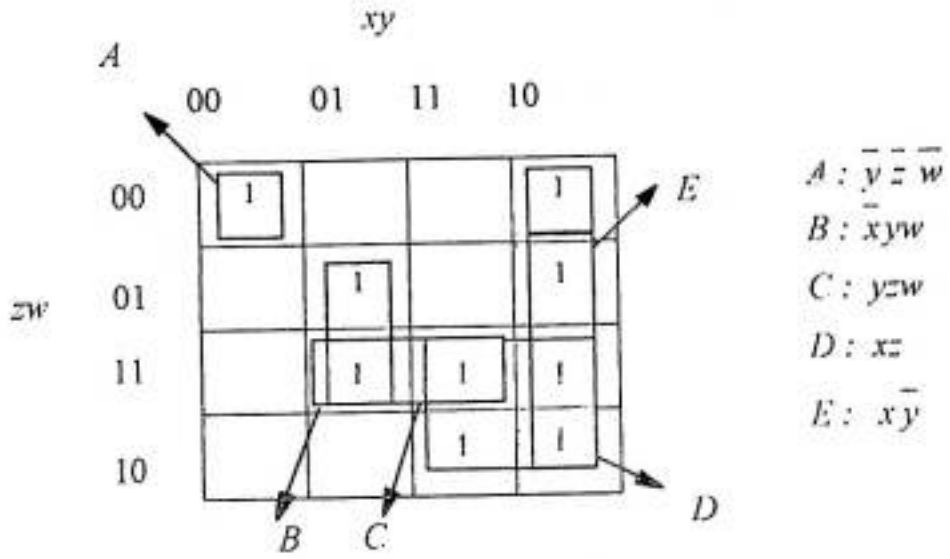
Metode Karnaugh merupakan metode yang tepat untuk menyatakan fungsi penyambungan secara grafik yang disebut "peta Karnaugh". Fungsi penyambungan dalam bentuk jumlahan hasilkali dan hasilkali jumlahan, dan hasil penyederhanaannya, bila dinyatakan dengan peta Karnaugh akan memberikan hasil yang sama. Dengan demikian, dalam peta Karnaugh dipusatkan pada pengelompokan suku-suku bertetangga atau berdekatan yang akan menghasilkan kesederhanaan, dan tidak mengutamakan bagaimana kedekatan yang ada di dalam diagram karena dalam peta Karnaugh dapat memperagakan kedekatan yang ada di dalam diagram jumlahan hasilkali. Prosedur untuk penciutan fungsi penyambungan dengan cara peta Karnaugh digambarkan sebagai berikut :

1. Mencari semua implikan utama.
2. Mencari implikan utama terpenting
3. Mencari set terkecil dari implikan utama yang mencakup semua implikan utama terpenting untuk menutup semua 1 dalam peta Karnaugh.

Misalkan fungsi Boole empat variabel.

$$F(x, y, z, w) = \bar{x}\,\bar{y}\,\bar{z}\,\bar{w} + x\bar{y}\,\bar{z}\,\bar{w} + \bar{x}y\bar{z}w + x\bar{y}\,\bar{z}w + \bar{x}yzw + xyzw + x\bar{y}zw + xy\bar{z}\bar{w} + x\bar{y}z\bar{w} \qquad (4.1.3)$$

Fungsi penyambungan tersebut mempunyai peta Karnaugh seperti yang diperlihatkan pada gambar berikut :



Gambar 4.2

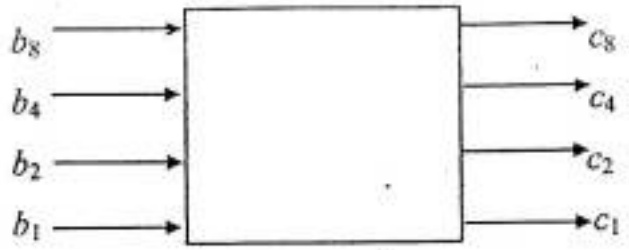
Dari gambar 4.2 ditunjukkan bahwa implikan *A, B, C, D,* dan *E* adalah hasil penyederhanaan dua/lebih *minterm.* Sebagai contoh, untuk implikan $B = \bar{x}yw$ merupakan hasil penyederhanaan dua *minterm* $\bar{x}y\bar{z}w$ dan $\bar{x}yzw$ yang ditunjukkan oleh digit 0101 dan 0111. Kedua *minterm* ini merupakan suku-suku bertetangga sehingga dapat disederhanakan menjadi digit 01–1 yang dinyatakan dengan $\bar{x}yw$. Dari implikan utama, yang merupakan implikan utama terpenting adalah :

$$\bar{y}\bar{z}\bar{w},\ \bar{x}yw, xz, x\bar{y} \qquad (4.1.4)$$

Oleh karena dalam penyederhanaan ini, set implikan utama terpenting mencakup semua unsur fungsi dalam peta Karnaugh, maka bentuk jumlahan hasilkali minimal dari fungsi adalah

$$F(x, y, z, w) = xz + x\bar{y} + \bar{x}yw + \bar{y}\bar{z}\bar{w} \qquad (4.1.5)$$

Misalkan akan ditentukan suatu fungsi Boole dari output rangkaian seperti pada gambar berikut dengan menggunakan peta Karnaugh



Gambar 4.3

Dengan tabel input dan output sebagai berikut

Tabel 4.1

Input digit					Output digit				
Desimal	Binary				Binary				Desimal
	b_8	b_4	b_2	b_1	c_8	c_4	c_2	c_1	
0	0	0	0	0	1	0	0	1	9
1	0	0	0	1	1	0	0	0	8
2	0	0	1	0	0	1	1	1	7
3	0	0	1	1	0	1	1	0	6
4	0	1	0	0	0	1	0	1	5
5	0	1	0	1	0	1	0	0	4
6	0	1	1	0	0	0	1	1	3
7	0	1	1	1	0	0	1	0	2
8	1	0	0	0	0	0	0	1	1
9	1	0	0	1	0	0	0	0	0

Misalkan akan ditentukan c_4, dengan menggunakan metode peta Karnaugh. Berdasarkan tabel 4.1 bahwa c_4 bernilai 1 diempat posisi. Sebagai contoh, 1 pertama pada kolom c_4 berada dibaris ketiga (input 2 dan output 7) bersesuaian dengan nilai input.

$$b_8 = 0, b_4 = 0, b_2 = 1, \text{dan } b_1 = 0 \qquad (4.1.6)$$

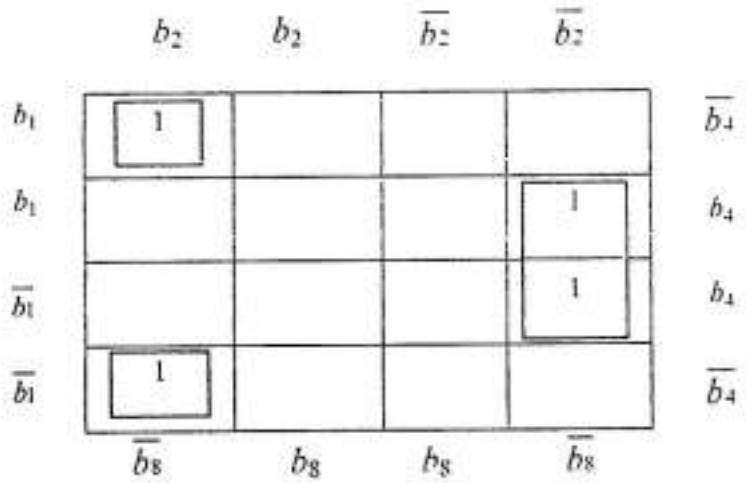
dalam bentuk hasilkali akan diperoleh

$$\overline{b_8}\,\overline{b_4}\;b_2\overline{b_1} = 1 \qquad (4.1.7)$$

untuk ketiga nilai 1 dalam kolom c_4 dapat diperoleh dari hasilkali input b_i yang menghasilkan 1, sehingga jika semua hasilkali dijumlahkan akan diperoleh

$$\overline{b_8}\,\overline{b_4}\;b_2\overline{b_1} + \overline{b_8}\,\overline{b_4}\;b_2\,b_1 + \overline{b_8}\,b_4\,\overline{b_2}\,\overline{b_1} + \overline{b_8}\,b_4\,\overline{b_2}\,b_1 \qquad (4.1.8)$$

Dengan demikian untuk sembarang baris dalam tabel 4.1 jumlah hasilkali dalam persamaan 4.18 menghasilkan c_4. Untuk menyederhanakan persamaan 4.18 akan digunakan metode peta Karnaugh, seperti yang ditunjukkan dalam gambar berikut



Gambar 4.4

Dua *term* yang bersesuaian dengan kedua suku ini adalah $b_2\overline{b_4}\;\overline{b_8}$ dan $\overline{b_2}\,b_4\,\overline{b_8}$, sehingga persamaan untuk c_4 adalah

$$c_4 = b_2\overline{b_4}\;\overline{b_8} + \overline{b_2}\,b_4\,\overline{b_8} \qquad (4.1.9)$$

Dengan cara yang sama persamaan untuk c_1, c_2, dan c_8 dapat ditentukan.

Pendekatan lain dapat digunakan untuk menentukan c_4, yaitu dengan hanya mengamati input dan output pada tabel 4.1. Pendekatan ini berbeda dengan metode peta Karnaugh. Dari tabel diperoleh bahwa

1. Kolom b_2 dan c_2 sama

$$c_2 = b_2 \tag{4.1.10}$$

2. Setiap entri dalam kolom c_1 merupakan komplemen dari entri yang bersesuaian dalam kolom b_1 sehingga

$$c_1 = \overline{b_1} \tag{4.1.11}$$

3. Terdapat 1 dalam kolom c_4 jika terdapat 1 dalam kolom b_2 atau kolom b_4, tetapi tidak kedua-duanya.

$$c_4 = (b_2 + b_4)(\overline{b_2} + \overline{b_4}) \tag{4.1.12}$$

Dari persamaan 4.1.12 dengan menggunakan hukum distributif akan diperoleh

$$\begin{aligned} c_4 &= (b_2 + b_4)(\overline{b_2} + \overline{b_4}) \\ &= b_2\,\overline{b_2} + b_2\,\overline{b_4} + \overline{b_2}\,b_4 + b_4\overline{b_4} \\ &= b_2\,\overline{b_4} + \overline{b_2}\,b_4 \end{aligned} \tag{4.1.13}$$

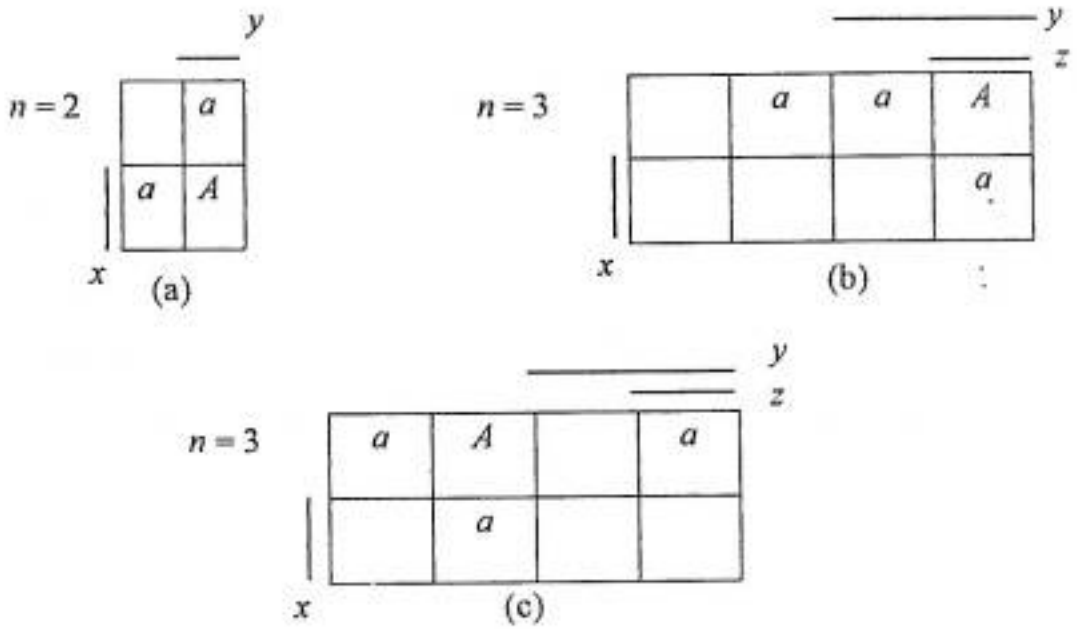
Hasil ini lebih sederhana dibanding hasil yang diperoleh dengan menggunakan metode Karnaugh yaitu tanpa $\overline{b_8}$. Akan tetapi persamaan ini benar karena $\overline{b_8} = 1$ untuk semua baris dengan nilai kolom $c_4 = 1$. Dengan kata lain, jika $b_2\,\overline{b_4} + \overline{b_2}\,b_4$ dikalikan $\overline{b_8}$ maka diperoleh nilai yang sama pada persamaan 4.1. 9 Dalam pendekatan peta Karnaugh $\overline{b_8}$ bukan variabel redundan, hal ini disebabkan karena

hanya 10 dari 16 kemungkinan kombinasi b_i yang digunakan pada tabel 4.1.

IV.1.2 METODE QUINE-McCLUSKEY

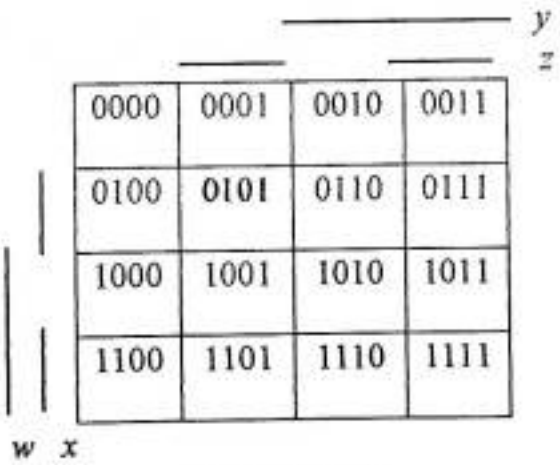
Metode Quine-McCluskey merupakan salah satu prosedur untuk penyederhanaan ekspansi jumlahan hasilkali yang ditunjukkan oleh suatu rangkaian. Metode Quine-McCluskey dikenal sebagai metode tabulasi (*tabular method*). Metode ini dapat digunakan untuk menyederhanakan fungsi Boole untuk jumlah/banyak variabel.

Suatu ekspresi Boole dikatakan mempunyai tetangga (*neighbors*) jika dua buah ekspresi tersebut yang merupakan hasilkali dari n literal nampak saling komplemen di satu posisi digit dan bersesuaian di posisi lainnya. Untuk menentukan tetangga dari suatu *minterm* akan digunakan "matriks Marquand". Dengan matriks Marquand akan diberikan cara mekanis (yang dapat diprogram) dan rekursif untuk mendapatkan semua tetangga dari suatu *minterm*. Tetangga dari *minterm* selanjutnya akan digunakan dalam metode Quine-McCluskey. Gambar berikut menunjukkan suatu matriks Marquand dari *minterm* n variabel untuk memperoleh suatu tetangga dari *minterm A*. Basis pencarian tetangga diawali dari matriks Marquand berukuran 2×2 yang mengandung hanya dua peubah. Jika *minterm* terdiri atas n digit/variabel maka *minterm* tersebut mempunyai n buah tetangga.



Gambar 4.5

Dari gambar 4.5 (a) *minterm* $A = xy$ mempunyai tetangga (*neighbors*) $\bar{x}y$ dan $x\bar{y}$ yang ditunjukkan oleh *a*. Tetangga dari *minterm A* pada gambar 4.5(b) dapat dicari dengan membuat persegiempat dalam. Jika *A* berada di luar persegi empat tersebut maka dua tetangga dari *A* berada di dalam persegi empat tersebut yang sebaris dengan *minterm A*. Sedangkan sisanya berada di kolom yang sama. Jika *minterm A* berada di dalam persegi empat maka tetangga dari A berada di luar persegi empat tersebut. Gambar berikut menunjukkan matriks Marquand dari suatu *minterm* empat variabel.



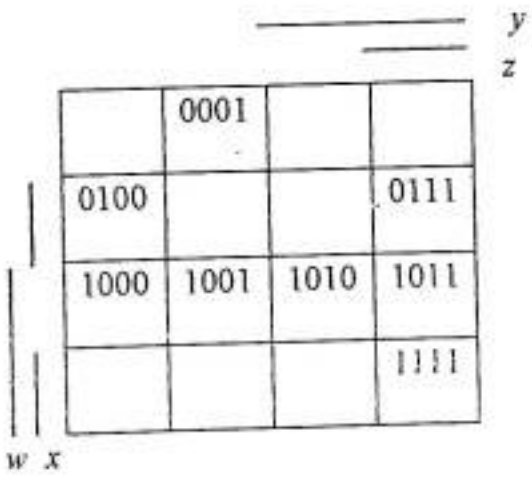
Gambar 4.6

Gambar 4.6 merupakan matriks Marquand untuk empat variabel. Tetangga dari *minterm* 0101 dapat diperoleh dengan cara membuat persegiempat dalam pada bagian vertikal. Sehingga *minterm* 0101 berada di dalam persegi empat tersebut. Dengan demikian terdapat dua tetangga dari *minterm* 0101 yaitu 0100 dan 0111 yang sebaris dengan 0101 berada di luar persegi empat vertikal. Tetangga dari *minterm* 0101 yang lain dapat diperoleh dengan cara yang sama yaitu membuat persegi empat horizontal sehingga diperoleh dua tetangga yang lain, 0001 dan 1101. Untuk *minterm* yang lain tetangga dapat diperoleh dengan cara yang sama.

Misalkan fungsi Boole empat variabel

$$F(w, x, y, z) = wxyz + \bar{w}xyz + w\bar{x}yz + wx\bar{y}\bar{z} + w\bar{x}\bar{y}z + \bar{w}\bar{x}yz + \bar{w}x\bar{y}\bar{z} + w\bar{x}\bar{y}\bar{z} \qquad (4.1.14)$$

setiap *minterm* dalam fungsi dapat dibentuk ke dalam matriks Marquand sebagai berikut



Gambar 4.7

setiap *minterm* dipasangkan dengan setiap tetangganya. Sehingga berdasarkan urutan dalam matriks Marquand diperoleh :

(1,9), (7,15), (8,10), (9,11), (10,11), (11,15) (4.1.15)

setiap pasangan *minterm* dengan tetangganya akan diperoleh suatu variabel yang redundan yang selanjutnya merupakan bagian dari proses penyederhanaan rangkaian dengan metode Quine-McCluskey. Sebagai contoh, (1,9) = (0001,1001) akan menghasilkan –001. Demikian seterusnya untuk pasangan yang lain.

Penyederhanaan rangkaian logika dengan menggunakan metode Quine-McCluskey akan dilakukan dengan berdasarkan pada langkah-langkah penyederhanaan ekspresi jumlahan hasilkali sebagai berikut :

1. Menyatakan setiap *minterm* fungsi Boole dengan *n variabel* dalam notasi biner atau *bit string*, index, dan nilai desimalnya.
2. Mengelompokkan notasi biner dari *minterm* berdasarkan indexnya. Semua notasi biner dengan index yang sama dikumpulkan dalam suatu kelompok. Semua kelompok index didaftar dalam suatu kolom mulai index terkecil. Untuk setiap kelompok, notasi biner disusun mulai dari nilai desimal ekivalennya yang terkecil.
3. Membandingkan notasi biner dari *minterm* dalam kelompok index terkecil dengan kelompok index yang lebih besar, dan menghilangkan semua variabel redundan. Variabel pada posisi yang dihilangkan diberi tanda strip (–).

4. Memeriksa satu persatu semua *minterm* yang dimasukkan ke dalam kombinasi. unsur yang tertinggal adalah implikan utama.
5. Mengulangi langkah 3 dan 4 sampai reduksi tidak mungkin lagi. Dengan cara demikian akan diperoleh himpunan dari semua implikan utama.
6. Menyusun tabel implikan utama dan mencari implikan utama terpenting dari fungsi. Dalam tabel, masing-masing kolom memuat angka desimal di bagian atas yang berhubungan dengan *minterm-minterm* dalam bentuk jumlahan hasilkali dari fungsi yang diberikan. Sedangkan masing-masing baris berhubungan dengan salah satu dari implikan utama.
7. Membuat tanda X di bawah masing-masing nilai desimal yang merupakan unsur yang diisikan dalam implikan utama yang dinyatakan oleh baris tersebut.
8. Mencari semua kolom yang berisi tanda kali tunggal dan diberi tanda kurung. Baris-baris demikian adalah implikan utama terpenting.
9. Mencakup semua implikan utama terpenting dalam jumlahan hasilkali minimal. Hasil ini merupakan bentuk minimal, jadi tidak perlu langkah lebih lanjut.

IV.2 PENYEDERHANAAN RANGKAIAN LOGIKA DENGAN METODE QUINE-McCLUSKEY

Penyederhanaan rangkaian logika dengan menggunakan metode ini akan dilakukan berdasarkan prosedur penyederhanaan yang telah dijelaskan sebelumnya. Dan proses penyederhanaannya akan dibahas dengan menjelaskan terlebih dahulu

beberapa aturan yang digunakan dalam metode ini dan dilanjutkan dengan menunjukkan metodologinya dalam penyederhanaan rangkaian logika.

IV.2.1 ATURAN DASAR METODE QUINE-McCLUSKEY

Dalam metode Quine-McCluskey digunakan aturan dasar $x + \bar{x} = 1$, dengan penggunaan notasi biner selain notasi desimal. Suatu variabel yang bernilai benar didefinisikan sebagai notasi biner 1 dan bernilai salah dengan notasi biner 0. Variabel yang hilang setelah kombinasi diberi tanda strip (-).

Misalkan $F(x, y, z)$ suatu fungsi Boole dari tiga variabel :

$x\bar{y}z$	:	Notasi biner 101
$\bar{x}\,y\bar{z}$	:	Notasi biner 010
$x\bar{y}$	:	Notasi biner 10 -
xz	:	Notasi biner 1 - 1

Misalkan fungsi Boole empat variabel :

$$F(x, y, z, w) = Sum\ (0011,1011) = \bar{x}\,\bar{y}\,z\,w + x\bar{y}\,z\,w = \bar{y}\,z\,w \qquad (4.2.1)$$

Kedua *minterm* dapat dikombinasikan sebagai :

x	y	z	w
0	0	1	1
1	0	1	1
-	0	1	1

Misalkan fungsi Boole empat variabel :

$$F(x, y, z, w) = Sum\ (0111,1011) = \bar{x}\, y\, z\, w + x\, \bar{y}\, z\, w \qquad (4.2.2)$$

Kedua *minterm* tidak dapat dikombinasikan ;

x	y	z	w
0	1	1	1
1	0	1	1

?

Kedua *minterm* tersebut tidak dapat dikombinasikan karena berbeda lebih dari satu posisi digit. Oleh karena itu, aturan dasar kedua dari metode Quine-McCluskey adalah untuk dua *term* yang akan dikombinasikan harus berbeda satu dan hanya satu posisi digit.

Index menyatakan banyaknya notasi biner 1 yang ditunjukkan oleh suatu *minterm.*

Contoh :

Misalkan fungsi Boole tiga variabel :

$$F(x, y, z) = Sum\ (000, 010, 100, 111) \qquad (4.2.3)$$

index masing-masing notasi biner dari fungsi adalah :

Index 0 : 000

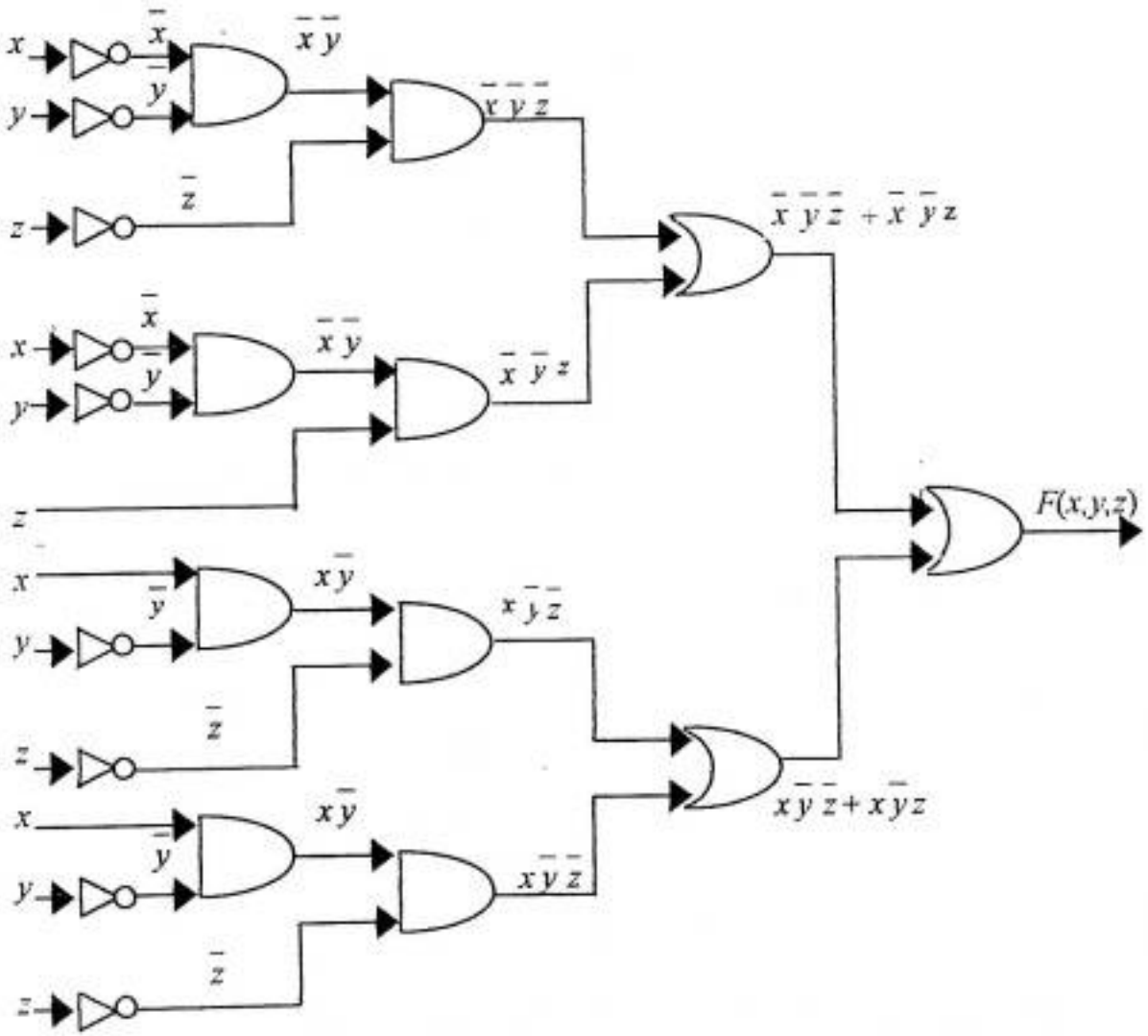
Index 1 : 010, 100

Index 3 : 111

Setelah dilakukan proses penyederhanaan akan diperoleh suatu implikan utama dari fungsi. Implikan utama adalah suatu implikan yang bukan berupa subset dari implikan lain dari fungsi.

IV.2.2 METODOLOGI

Misalkan suatu rangkaian logika yang ditunjukkan berikut ini :



Gambar 4.8

Misalkan suatu fungsi Boole tiga variabel yang ditunjukkan oleh rangkaian tersebut

$$F(x, y, z) = \bar{x}\,\bar{y}\,\bar{z} + \bar{x}\,\bar{y}\,z + x\,\bar{y}\,\bar{z} + x\,\bar{y}\,z \qquad (4.2.4)$$

Rangkaian logika pada gambar 4.8 akan disederhanakan dengan menggunakan metode Quine-McCluskey.

Langkah pertama yaitu menyatakan atau mengubah fungsi Boole ke dalam notasi biner dengan nilai index dan nilai desimal.

$$F(x, y, z) = Sum\ (000, 001, 100, 101) \qquad (4.2.5)$$

$\bar{x}\,\bar{y}\,\bar{z}$: Notasi biner 000 : Index 0 : Nilai desimal 0

$\bar{x}\,\bar{y}\,z$: Notasi biner 001 : Index 1 : Nilai desimal 1

$x\,\bar{y}\,\bar{z}$: Notasi biner 100 : Index 1 : Nilai desimal 4

$x\,\bar{y}\,z$: Notasi biner 101 : Index 2 : Nilai desimal 5

Tabulasi dalam bentuk daftar berdasarkan kelompok index dan nilai desimalnya.

Tabel 4.2

Daftar 1		Daftar 2		Daftar 3	
	x y z		*x y z*		*x y z*
(0)	0 0 0 [x]	(0,1)	0 0 – [x]	(0,1,4,5)	– 0 –
(1)	0 0 1 [x]	(0,4)	– 0 0 [x]	(0,4,1,5)	– 0 –
(4)	1 0 0 [x]	(1,5)	– 0 1 [x]		
(5)	1 0 1 [x]	(4,5)	1 0 – [x]		

Dari daftar 1, *term* dikelompokkan berdasarkan index dan hanya satu digit yang berbeda dari satu index ke index yang lainnya. *Term* dari daftar pertama dipisahkan ke dalam kelompok-kelompok pada daftar kedua. Tanda [x] hanya

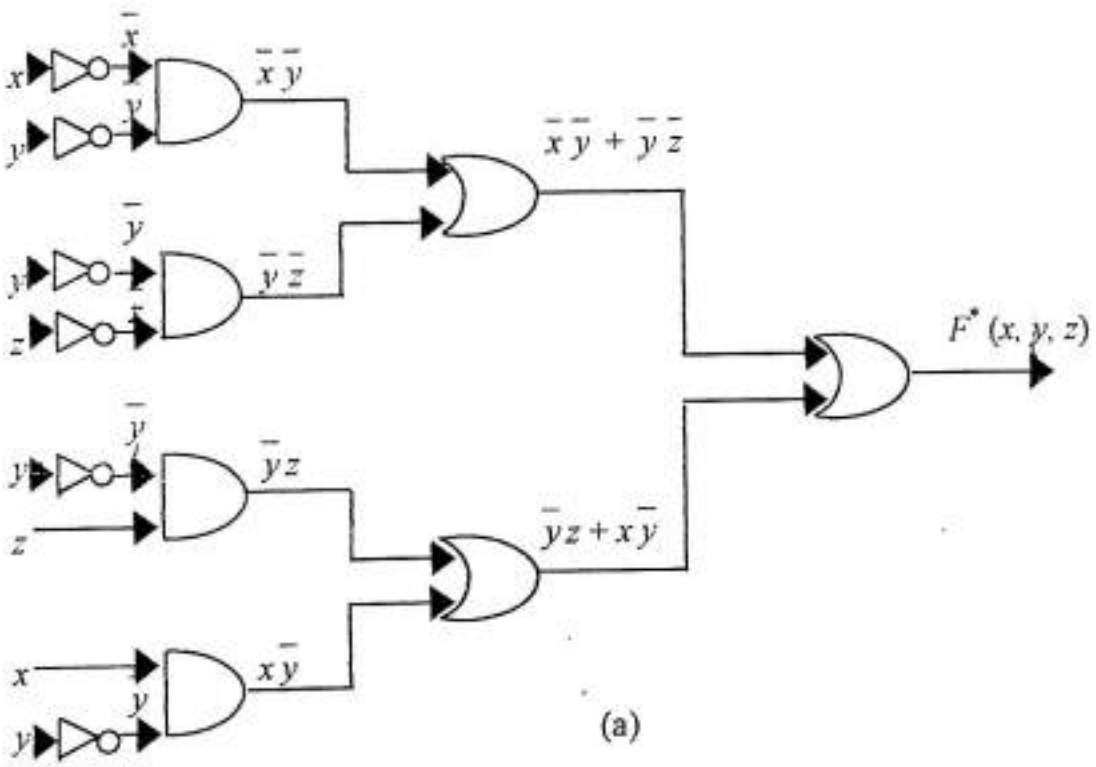
menunjukkan bahwa satu *term* telah dikombinasikan dengan *term* lainnya. Dari daftar kedua diperoleh implikan utama yaitu :

$$F^*(x, y, z) = \bar{x}\,\bar{y} + \bar{y}\,\bar{z} + \bar{y}z + x\,\bar{y} \qquad (4.2.6)$$

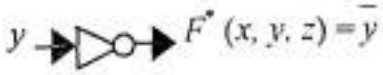
Dari daftar ketiga diperoleh hasil penyederhanaan ekspresi :

$$F^*(x, y, z) = \bar{y} \qquad (4.2.7)$$

Dengan demikian, dari persamaan (4.2.6) diperoleh rangkaian hasil penyederhanaan sebagai berikut :



(a)



(b)

Gambar 4.9

Implikan utama yang diperoleh pada persamaan (4.2.6) dan (4.2.7) dinyatakan sebagai rangkaian logika dalam gambar 4.9. Rangkaian logika ini merupakan hasil penyederhanaan dari rangkaian logika pada gambar 4.8.

Misalkan fungsi Boole empat variabel

$$F(x, y, z, w) = \bar{x}\bar{y}\bar{z}\bar{w} + \bar{x}\bar{y}\bar{z}w + \bar{x}\bar{y}z\bar{w} + \bar{x}\bar{y}zw + \bar{x}y\bar{z}w + \bar{x}yzw +$$

$$x\bar{y}\bar{z}\bar{w} + x\bar{y}z\bar{w} + xy\bar{z}\bar{w} + xy\bar{z}w + xyzw \qquad (4.2.8)$$

dalam notasi biner

$$F(x, y, z, w) = Sum\ (0000, 0001, 0010, 0011, 0101, 0111, 1000, 1010, 1100,$$

$$1101, 1111) \qquad (4.2.9)$$

dalam notasi index

$$F(x, y, z, w) = Sum\ (0, 1, 1, 2, 2, 3, 1, 2, 2, 3, 4) \qquad (4.2.10)$$

dalam bentuk desimal

$$F(x, y, z, w) = Sum\ (0, 1, 2, 3, 5, 7, 8, 10, 12, 13, 15) \qquad (4.2.11)$$

Fungsi ini akan dikombinasikan dengan pengelompokan berdasarkan index masing-masing *minterm*. Dalam tabel 4.3 setiap *minterm* dari fungsi yang ditunjukkan dalam notasi biner dikombinasikan dari satu index ke index yang lain dengan berdasarkan bahwa setiap *minterm* yang dikombinasikan harus berbeda tepat satu notasi biner dan berada pada posisi yang sama. hasil kombinasi ditunjukkan dalam daftar 2 dan 3. Implikan utama yang diperoleh ditunjukkan dengan tanda [1] dan seterusnya. Dari implikan utama ini dibentuk fungsi pada persamaan (4.2.12).

Kombinasinya dapat ditunjukkan dalam daftar berikut:

Tabel 4.3

Daftar 1		Daftar 2		Daftar 3	
	x y z w		*x y z w*		*x y z w*
(00)	0 0 0 0 [x]	(00, 01)	0 0 0 - [x]	(00, 01, 02, 03)	0 0 - - [3]
(01)	0 0 0 1 [x]	(00, 02)	0 0 - 0 [x]	(00, 02, 01, 03)	0 0 - -
(02)	0 0 1 0 [x]	(00, 08)	- 0 0 0 [x]	(00, 02, 08, 10)	- 0 - 0 [4]
(08)	1 0 0 0 [x]	(01, 03)	0 0 - 1 [x]	(00, 08, 02, 10)	- 0 - 0
(03)	0 0 1 1 [x]	(01, 05)	0 - 0 1 [x]	(01, 03, 05, 07)	0 - - 1 [5]
(05)	0 1 0 1 [x]	(02, 03)	0 0 1 - [x]	(01, 05, 03, 07)	0 - - 1
(10)	1 0 1 0 [x]	(02, 10)	- 0 1 0 [x]	(05, 07, 13, 15)	- 1 - 1 [6]
(12)	1 1 0 0 [x]	(08, 10)	1 0 - 0 [x]	(05, 13, 07, 15)	- 1 - 1
(07)	0 1 1 1 [x]	(08, 12)	1 - 0 0 [1]		
(13)	1 1 0 1 [x]	(03, 07)	0 - 1 1 [x]		
(15)	1 1 1 1 [x]	(05, 07)	0 1 - 1 [x]		
		(05, 13)	- 1 0 1 [x]		
		(12, 13)	1 1 0 - [2]		
		(07, 15)	- 1 1 1 [x]		
		(13, 15)	1 1 - 1 [x]		

Implikan utama dari fungsi yang diperoleh dari penyederhanaan ini adalah

$$F(x, y, z) = \bar{x}\,\bar{y} + \bar{y}\,\bar{w} + \bar{x}w + yw + x\bar{z}\,\bar{w} + x\,y\bar{z} \qquad (4.2.12)$$

Selanjutnya, dibentuk tabel 4.3 dengan setiap hasilkali Boole yang telah diperoleh pada persamaan (4.2.12) untuk bagian baris dan setiap *minterm* dari fungsi Boole pada persamaan (4.2.8) untuk bagian kolom. Dalam tabel diletakkan tanda X pada

jika *minterm* dari persamaan (4.2.8) dibentuk oleh setiap hasilkali dalam persamaan (4.2.12).

Tabel 4.4

Minterm	00	01	02	03	05	07	08	10	12	13	15
$\bar{x}\,\bar{y}$	X	X	X	X							
$\bar{y}\,\bar{w}$ *	X		X				X	(X)			
$\bar{x}w$		X		X	X	X					
yw *					X	X				X	(X)
$x\bar{z}\,\bar{w}$							X		X		
$xy\bar{z}$									X	X	

Dari tabel 4.3, yw dan $\bar{y}\,\bar{w}$ adalah implikan utama yang terpenting. yw termasuk dalam *minterm* dengan desimal 5, 7, 13, dan 15. $\bar{y}\,\bar{w}$ pada *minterm* dengan desimal 0, 2, 8, dan 10. *Minterm* lain dari fungsi yang merupakan syarat perlu adalah 1, 3, dan 12. *Minterm* 1 dipenuhi oleh implikan utama $\bar{x}\,\bar{y}$ dan $\bar{x}w$, demikian pula dengan *minterm* 3. *Minterm* 12 dipenuhi oleh $x\bar{z}\,\bar{w}$ dan $xy\bar{z}$. *Minterm* $\bar{x}w$ dan $xy\bar{z}$ merupakan dua implikan utama yang tidak menjadi syarat perlu.

Jadi, fungsi yang telah diminimumkan adalah :

$$F^*(x, y, z, w) = \bar{y}\,\bar{w} + yw + \bar{x}\,\bar{y} + x\bar{z}\,\bar{w} \qquad (4.2.13)$$

IV.3 APLIKASI DALAM PROGRAM MAPLE V

Aljabar Boole dapat dipelajari dengan menggunakan program MapleV, terutama mengenai fasilitas-fasilitas operator logika. Secara umum dalam program MapleV operator-operator yang digunakan antara lain : *&or*, *&and*, dan *&not* yang masing-masing menunjukkan operasi +, operasi $\cdot$, dan komplementasi $\bar{}$. Untuk mempelajari operator ini digunakan fungsi tipe ***with (logic)***.

Fungsi ***logic[bsimp]*** digunakan untuk menyederhanakan ekspansi jumlahan hasilkali (*sum of products expansion*) dari ekspresi Boole yang diberikan.

Misalkan fungsi Boole :

$$F(x, y, z) = x\,y\,z + x\,\bar{y}\,z \tag{4.3.1}$$

$$F(x, y, z) = \bar{x}\,\bar{y}\,\bar{z} + \bar{x}\,\bar{y}z + x\bar{y}\,\bar{z} + x\bar{y}z \tag{4.3.2}$$

$$F(x,\ y,\ z,\ w) = \bar{x}\,\bar{y}\,\bar{z}\,\bar{w} + \bar{x}\,\bar{y}\,\bar{z}w + \bar{x}\,\bar{y}z\bar{w} + \bar{x}\,\bar{y}zw + \bar{x}y\bar{z}w + \bar{x}yzw + x\bar{y}\,\bar{z}\,\bar{w} + x\bar{y}z\bar{w} + x\bar{y}\bar{z}\,\bar{w} + x\bar{y}\bar{z}w + x\,y\,z\,w \tag{4.3.3}$$

Jumlahan hasilkali yang ditunjukkan dalam fungsi di atas dapat disederhanakan dalam program MapleV seperti yang ditunjukkan dalam lampiran.

Fungsi ***logic[dual]*** merupakan fungsi dalam MapleV untuk mencari dualitas dari suatu ekspresi Boole yang diberikan. Misalkan dari fungsi Boole yang diberikan pada persamaan (4.3.1), (4.3.2), dan (4.3.3). Dualitas dari fungsi ini akan ditunjukkan dengan program MapleV pada lampiran.

Fungsi ***logic*[*bequal*]** merupakan fungsi dalam program MapleV untuk menguji apakah dua buah ekspresi Boole yang diberikan ekivalen secara logika. Jika hasil yang diperoleh benar (*true*), maka kedua ekspresi Boole ekivalen, dan jika salah (*false*), maka kedua ekspresi tidak ekivalen.

Misalkan teorema *De Morgan* :

$$\overline{x+y} = \bar{x}.\bar{y}$$

$$\overline{x.y} = \bar{x}+\bar{y} \qquad (4.3.4)$$

dan misalkan fungsi Boole pada persamaan (4.3.1), (4.3.2), dan (4.3.3). Ekspresi Boole ini akan ditentukan ekivalennya dengan menggunakan program MapleV seperti yang ditunjukkan pada lampiran .

Fungsi ***logic*** **[*canon*]** merupakan fungsi dalam program MapleV untuk mengubah ekspresi Boole yang diberikan kebentuk jumlahan hasilkali kanonik. Ekspresi Boole pada persamaan (4.3.1), (4.3.2), dan (4.3.3) akan diubah kebentuk jumlahan hasilkali kanonik seperti yang ditunjukkan pada lampiran.

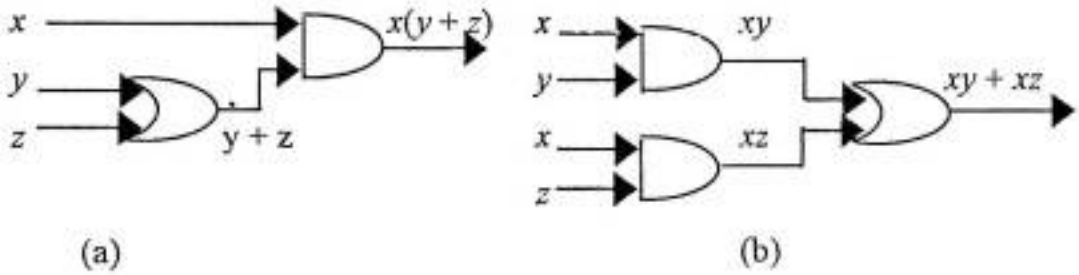
BAB V

PENUTUP

V.1 KESIMPULAN

Dari pembahasan di atas dapat disimpulkan bahwa :

1. Dua ekspresi Boole yang berbeda, tetapi ekivalen dapat dinyatakan melalui dua rangkaian yang berbeda. Misalkan, ekspresi Boole $x\ (y + z)$ dapat dinyatakan melalui dua/lebih rangkaian yang berbeda, misalnya :



Gambar 5.1

2. Metode Quine-McCluskey adalah salah satu metode yang dapat diimplementasikan dalam suatu program rangkaian paling sederhana dengan prosedur penyederhanaan ekspresi jumlahan hasilkali dalam suatu rangkaian untuk memperoleh implikan utama dari fungsi Boole, berapapun banyak peubahnya.
3. Melalui pengujian dengan program MapleV dapat disimpulkan bahwa hasil penyederhanaan ekspresi Boole yang diperoleh sama ataupun ekivalen dengan hasil penyederhanaan dengan menggunakan metode Quine-McCluskey, meskipun dalam program ini tidak ditunjukkan prosedur dari metode Quine-McCluskey (lampiran).

V.2 SARAN-SARAN

Tulisan ini masih jauh dari kesempurnaan, karena hanya membahas metode Quine-McCluskey untuk mendapatkan rangkaian paling sederhana. Oleh karena itu, penulis menyarankan agar penyederhanaan rangkaian dengan metode lainnya dikembangkan lebih jauh lagi.

DAFTAR PUSTAKA

Bartee, Thomas C., *Dasar Komputer Digital*, Edisi Keenam, Erlangga, 1986.

Dossey John A., *Discrete Mathematics*, third Edition, Addison-Wesley Educational Publishers, Inc.

Lee Samuel C., *Digital Circuits and Logic Design*, Prentice-Hall, Inc., Englewood Cliffs., U.S.A., 1976.

Liu, Chung L ., *Elements of Discrete Mathematics*, Second Edition, Mc Graw-Hill International Edition, 1977.

Maurer, Stephen B., Antony Ralston, *Discrete Algorithmic Mathematics*, Addison-Wesley Publishing Company, Inc. 1991.

Peter, Fletcher, Hoyle, Patty, *Foundations of Discrete Mathematics*, PWS-KENT Publishing Company, 1991.

Rosen, Kenneth H., *Discrete Mathematics and Its Applications*, Fourth Edition, Mc Graw-Hill Internatinal Edition, 1999.

```
> with(logic);
```

[*bequal*, *bsimp*, *canon*, *convert/MOD2*, *convert/frominert*, *convert/toinert*, *distrib*, *dual*, *environ*, *randbool*, *satisfy*, *tautology*]

```
> bsimp(((x &and y)&and z)&or((x &and(&not y))&and z));
```

$$x \,\&\text{and}\, z$$

```
> canon(((x &and y)&and z)&or((x &and(&not y))&and z),{x,y,z},MOD2);
```

$$x\, z$$

```
> canon(((x &and y)&and z)&or((x &and(&not y))&and z),{x,y,z},DNF);
```

$$\&\text{and}(x, y, z)\ \&\text{or}\ \&\text{and}(x, \&\text{not}(y), z)$$

```
> canon(((x &and y)&and z)&or((x &and(&not y))&and z),{x,y,z},CNF);
```

$$\&\text{and}(\&\text{or}(x, \&\text{not}(y), \&\text{not}(z)), \&\text{or}(x, z, \&\text{not}(y)), \&\text{or}(x, y, \&\text{not}(z)), \&\text{or}(x, y, z), \&\text{or}(y, z, \&\text{not}(x)), \&\text{or}(z, \&\text{not}(y), \&\text{not}(x)))$$

```
> bsimp(((&not x)&and(&not y))&and(&not z))&or(((&not x)&and(&not y))&an
> d z)&or((x &and (&not y))&and(&not z))&or((x &and (&not y))&and z);
```

$$((\&\text{and}(\&\text{not}(y), \&\text{not}(x), \&\text{not}(z))\ \&\text{or}\ ((\&\text{not}(x)\ \&\text{and}\ \&\text{not}(y))\ \&\text{and}\ z))\ \&\text{or}\ ((x\ \&\text{and}\ \&\text{not}(y))\ \&\text{and}\ \&\text{not}(z)))\ \&\text{or}\ ((x\ \&\text{and}\ \&\text{not}(y))\ \&\text{and}\ z)$$

```
>
> canon((((&not x)&and(&not y))&and(&not z))&or(((&not x)&and(&not y))&a
> nd z)&or((x &and (&not y))&and(&not z))&or((x &and (&not y))&and z),{x,y,
> z},CNF);
```

$$\&\text{and}(\&\text{or}(\&\text{not}(x), \&\text{not}(y), \&\text{not}(z)), \&\text{or}(\&\text{not}(x), z, \&\text{not}(y)), \&\text{or}(\&\text{not}(z), \&\text{not}(y), x), \&\text{or}(\&\text{not}(y), z, x))$$

```
> canon((((&not x)&and(&not y))&and(&not z))&or(((&not x)&and(&not y))&a
```

```
> nd z)&or((x &and (&not y))&and(&not z))&or((x &and (&not y))&and z),{x,y,
> z},DNF);
```

$$\&or(\&and(\&not(x), \&not(y), \&not(z)), \&and(\&not(x), \&not(y), z), \&and(x, \&not(y), \&not(z)), \&and(x, \&not(y), z))$$

```
> canon((((&not x)&and(&not y))&and(&not z))&or(((&not x)&and(&not y))&a
> nd z)&or((x &and (&not y))&and(&not z))&or((x &and (&not y))&and z),{x,y,
> z},MOD2);
```

$$1 + y$$

```
> canon(((&and(&not(y),&not(x),&not(z))&or((&not(x)&and&not(y))&and z))&
> or((x &and &not(y))&and &not(z)))&or((x &and &not(y))&and z),{x,y,z},MOD
> 2);
```

$$1 + y$$

```
> bsimp(((((&not x)&and(&not y))&and(&not z))&and(&not w))&or((((&not x)
> &and(&not y))&and(&not z))&and w)&or((((&not x)&and(&not y))&and z)&a
> nd(&not w))&or((((&not x)&and (&not y))&and z)&and w)&or((((&not x)&an
> d y)&and(&not z))&and w)&or((((&not x)&and y)&and z)&and w)&or(((x &a
> nd (&not y))&and(&not z))&and(&not w))&or(((x &and (&not y))&and z)&an
> d(&not w))&or(((x &and y)&and(&not z))&and (&not w))&or (((x &and y)&a
> nd(&not z))&and w)&or (((x &and y)&and z)&and w));
```

$$\&or(\&and(x, y, \&not(z)), \&not(y)\ \&and\ \&not(x), \&not(y)\ \&and\ \&not(w), y\ \&and\ w)$$

```
>
> canon(((((&not x)&and(&not y))&and(&not z))&and(&not w))&or((((&not x)
> &and(&not y))&and(&not z))&and w)&or((((&not x)&and(&not y))&and z)&a
> nd(&not w))&or((((&not x)&and (&not y))&and z)&and w)&or((((&not x)&an
> d y)&and(&not z))&and w)&or((((&not x)&and y)&and z)&and w)&or(((x &a
> nd (&not y))&and(&not z))&and(&not w))&or(((x &and (&not y))&and z)&an
> d(&not w))&or(((x &and y)&and(&not z))&and (&not w))&or (((x &and y)&a
> nd(&not z))&and w)&or (((x &and y)&and z)&and w),{x,y,z,w},MOD2);
```

$$1 + y\ w + y + x\ w + x\ y\ z\ w + x\ y\ z + x\ y$$

```
>
> canon(&or(&and(x,y,&not(z)),&not(y)&and&not(x),&not(y)&and&not(w),y &
> and w),{x,y,z,w},MOD2);
```

$$1 + y + x\,y\,z + y\,x + w\,x + y\,w + w\,x\,y\,z$$

```
> dual(((x &and y)&and z)&or((x &and(&not y))&and z));
```

$$((x \,\&\text{or}\, y)\,\&\text{or}\, z)\,\&\text{and}\,((x\,\&\text{or}\,\&\text{not}(y))\,\&\text{or}\, z)$$

```
>
> dual(((&not x)&and(&not y))&and(&not z))&or(((&not x)&and(&not y))&and
> z)&or((x &and (&not y))&and(&not z))&or((x &and (&not y))&and z);
```

$$((((\&\text{not}(x)\,\&\text{or}\,\&\text{not}(y))\,\&\text{or}\,\&\text{not}(z))\,\&\text{or}$$
$$((\&\text{not}(x)\,\&\text{and}\,\&\text{not}(y))\,\&\text{and}\, z))\,\&\text{or}$$
$$((x\,\&\text{and}\,\&\text{not}(y))\,\&\text{and}\,\&\text{not}(z)))\,\&\text{or}$$
$$((x\,\&\text{and}\,\&\text{not}(y))\,\&\text{and}\, z)$$

```
> bequal((&not (x&and y)), ((&not x)&or(&not y)));
```

true

```
> bequal((&not(x &or y)), ((&not x)&and (&not y)));
```

true

```
> bequal(((x &and y)&and z)&or((x &and(&not y))&and z), x &and z);
```

true

```
>
> bequal((((&not x)&and(&not y))&and(&not z))&or(((&not x)&and(&not y))&a
> nd z)&or((x &and (&not y))&and(&not z))&or((x &and (&not y))&and z),(((&
> not x)&and(&not y))&or((&not y)&and(&and z))&or ((&not y)&and z)&or(x &
> and (&not y))));
```

true

```
>
> bequal((((&not x)&and(&not y))&and(&not z))&or(((&not x)&and(&not y))&a
> nd z)&or((x &and (&not y))&and(&not z))&or((x &and (&not y))&and z),(&n
> ot y));
```

true

```
> canon(x &or(&not x),{x},MOD2);
```

1

```
> convert(x &or (&not x),MOD2,'expanded');
```

```
                                    1
> convert(x &or (&not x),frominert);
                                  true
> canon(x &and(&not x),{x},MOD2);
                                    0
> convert(x &and(&not x),MOD2,'expanded');
                                    0
> convert(x &and(&not x),frominert);
                                 false
> canon((x &or(&not x))&or(x &and(&not x)),{x},MOD2);
                                    1
>
```