**LAPORAN**

**PENYEDERHANAAN RANGKAIAN**



Disusun untuk Memenuhi Tugas Mata Kuliah :

Logika Digital dan Sistem Digital

Disusun Oleh :

1. Andika Febrianto 18.11.0030
2. Refri Riyanto 18.11.0007
3. Ilham Fatkhul Qorib 18.11.0170

**PROGRAM STUDI TEKNIK INFORMATIKA**

**SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER**

**AMIKOM PURWOKERTO**

**PURWOKERTO**

**2019**

KATA PENGANTAR

Puji syukur alhamdulillah kami panjatkan ke hadirat Tuhan Yang Maha Esa, karena telah melimpahkan rahmat-Nya berupa kesempatan dan pengetahuan sehingga laporan ini bisa selesai pada waktunya. Laporan yang berjudul “Sistem Operasi Bilangan dan Pengkodean” disusun untuk memenuhi tugas mata kuliah Logika Digital dan Sistem Digital oleh Bapak Agus Pramono, M. T.

Terima kasih juga kami ucapkan kepada teman-teman yang telah berkontribusi dengan memberikan ide-idenya sehingga laporan ini bisa disusun dengan baik dan rapi.

Kami berharap semoga laporan ini bisa menambah pengetahuan para pembaca. Namun terlepas dari itu, kami memahami bahwa laporan ini masih jauh dari kata sempurna, sehingga kami sangat mengharapkan kritik serta saran yang bersifat membangun demi terciptanya makalah selanjutnya yang lebih baik lagi.

**DAFTAR ISI**

**KATA PENGANTAR 2**

**DAFTAR ISI 3**

**BAB I PENDAHULUAN 4**

**1.1 Latar Belakang 4**

**BAB II PEMBAHASAN 5**

**Pendahuluan Sistem Digital dan Logika Digital 5**

**Metode Penyederhanaan Rangkaian Logika 6**

**BAB III PENGERTIAN ALJABAR BOOLEAN 7**

**1.1 Dasar Operasi Logika 7**

**1.2 Operasi-operasi Dasar Logika Yang Dijelaskan Tabel Kebenaran 8**

**1.3 Hukum Aljabar Boolean 9**

**BAB IV PENYEDERHANAAN RANGKAIAN LOGIKA**

**2.1 Penyederhanaan Secara Aljabar 13**

**2.2 Metode Peta Karnaugh (K-Map) 14**

**2.3 Pembentukan Peta Karnaugh 15**

**2.4 Kuad 20**

**2.5 Oktet 21**

**2.6 Metode Tabulasi (Quine Mc Cluskey) 24**

**BAB v PENUTUP 31**

* 1. **Kesimpulan 31**
  2. **DAFTAR PUSTAKA 32**

**BAB I**

**PENDAHULUAN**

* 1. **Latar Belakang**

Perkembangan teknologi digital saat ini semakin maju. Masyarakat saat ini telah menjadikan komputer sebagai alat yang tidak bisa lepas dalam membantu pekerjaan mereka. Ini mungkin karena mesin tersebut menghasilkan fungsi aritmatika dengan ketelitian dan kecepatan yang sangat menakjubkan. Bab ini membicarakan beberapa sistem bilangan dan pengkonversiannya yang dapat menjumlahkan dan mengurangkan. Penambahan dan pengurangan dikerjakan dalam biner.

Telah kita maklum bersama bahwa elektronika mengalami perkembangan yang pesat dan cukup menakjubkan. Dengan perkembangan tersebut kadang-kadang kita agak repot dalam mengikutinya sehingga elektronika suatu ketika teramati seperti belantara yang membingungkan. Suatu rangkaian elektronika terdiri atas komponen-komponen dengan nama yang aneh-aneh, parameter-parameternya sering tidak sederhana, serta teori yang rumit. Untuk itulah hanya satu jawaban untuk megikuti perkembangan tersebut yaitu teruslah untuk bekerja keras dalam bidang elektronika ini. Analisis dan penelitian yang tidak kenal lelah dapat mendorong kita memahami dan menemukan hal-hal yang baru pada dunia elektronika ini.

**BAB II**

**PEMBAHASAN**

**Pendahuluan Sistem Digital dan Logika Digital**

Dalam sains dan teknologi, demikian pula berbagai bidang kehidupan yang lainnya, kita selalu dihadapkan pada yang namanya besaran yaitu sesuatu yang dapat kita amati, kita ukur, dan analisis. Dalam pekerjaan seperti itu kita membutuhkan sejumlah peralatan. Harapan kita adalah dapat menyajikan harga suatu besaran dengan cepat dan tepat. Pada dasarnya ada dua cara menyajikan harga numerik suatu besara yaitu secara analog dan digital. Penyajian secara analog dapat kita ambil contoh spedometer kendaraan dimana penyimpangan jarum menunjukkan harga tertentu dan mengikuti laju kendaraan yang ber-sangkutan. Dalam kasus tersebut besaran laju kendaraan dianalogikan dengan penyimpangan jarum spedometer. Masih banyak contoh lainnya silahkan anda mengamati sendiri dan mendiskusikan dengan teman-teman anda. Ciri khas penyajian secara analog dapat berada pada

sebarang nilai tidak ada nilai terlarang kecuali di luar batas kemampuan. Penyajian secara digital, dapat kita ambil contoh yang banyak ditemukan seperti jam digital. Waktu berubah secara kontinyu namun jam digital tidak dapat menunjukkan secara kontinyu. Penampilan waktu hanya dapat berubah pada tingkat paling kecil (menit atau detik). Dengan demikian penampilan waktu tersebut berubah secara diskrit. Ciri khas penyajian besaran secara digital adalah hanya berada pada nilai-nilai tertentu yang diskrit. Piranti elektronika sekarang ini menuju pada keadaan otomatisasi, minimisasi, dan digitasi. Otomatisasi menjadikan segala pekerjaan dapat dilakukan secara mudah dan akurat seolah dapat diselesaikan dengan sendirinya. Minimisasi menjadikan piranti elektronika menjadi semakin kecil dan kompak, tidak membutuhkan ruang yang besar tetapi kinerjanya sangat handal. Digitasi menjadikan pengolahan data semakinmenguntungkan dengan beberapa kelebihan antara lain : a. Lebih tegas karena data ditampilkan dalam dua keadaan YA atau TIDAK, MATI atau HIDUP, 1 atau 0, 0 volt atau 5 volt dan sebagainya. b. Mudah dikelola seperti disimpan dalam bentuk memori, mudah ditransmisikan, mudah dimunculkan kembali, mudah diolah tanpa penurunan kualitas. c. Lebih tahan terhadap gangguan atau lebih sedikit terkena gangguan. d. Kebutuhan dayanya yang rendah.

**METODE PENYEDERHANAAN RANGKAIAN LOGIKA**

* Penyederhanaan Secara Aljabar
* Peta Karnaugh
* Tabulasi (Quine Mc.Kluskey)

**BAB III**

**PENGERTIAN ALJABAR BOOLEAN**

Aljabar boolean merupakan aljabar yang berhubungan dengan variabel-variabel biner dan operasi-operasi logik. Variabel-variabel diperlihatkan dengan huruf-huruf alfabet, dan tiga operasi dasar dengan AND, OR dan NOT (komplemen). Fungsi boolean terdiri dari variabel-variabel biner yang menunjukkan fungsi, suatu tanda sama dengan, dan suatu ekspresi aljabar yang dibentuk dengan menggunakan variabel-variabel biner, konstanta-konstanta 0 dan 1, simbol-simbol operasi logik, dan tanda kurung.  
  
    Suatu fungsi boolean bisa dinyatakan dalam tabel kebenaran. Suatu tabel kebenaran untuk fungsi boolean merupakan daftar semua kombinasi angka-angka biner 0 dan 1 yang diberikan ke variabel-variabel biner dan daftar yang memperlihatkan nilai fungsi untuk masing-masing kombinasi biner.  
  
    Aljabar boolean mempunyai 2 fungsi berbeda yang saling berhubungan. Dalam arti luas, aljabar boolean berarti suatu jenis simbol-simbol yang ditemukan oleh George Boole untuk memanipulasi nilai-nilai kebenaran logika secara aljabar. Dalam hal ini aljabar boolean cocok untuk diaplikasikan dalam komputer. Disisi lain, aljabar boolean juga merupakan suatu struktur aljabar yang operasi-operasinya memenuhi aturan tertentu.

**1.1 DASAR OPERASI LOGIKA:**  
    Memberikan batasan yang pasti dari suatu keadaan, sehingga suatu keadaan tidak dapat berada dalam dua ketentuan sekaligus.  
Dalam logika dikenal aturan sbb :

* Suatu keadaan tidak dapat dalam keduanya benar dan salah sekaligus
* Masing-masing adalah benar / salah.
* Suatu keadaan disebut benar bila tidak salah.

Dalam ajabar boolean keadaan ini ditunjukkan dengan dua konstanta : LOGIKA ‘1’ dan ‘0’  
Operasi-operasi dasar logika dan gerbang logika :  
Pengertian GERBANG (GATE) :

* Rangkaian satu atau lebih sinyal masukan tetapi hanya menghasilkan satu sinyal keluaran.
* Rangkaian digital (dua keadaan), karena sinyal masukan atau keluaran hanya berupa tegangan tinggi atau low ( 1 atau 0 ).
* Setiap keluarannya tergantung sepenuhnya pada sinyal yang diberikan pada masukan-masukannya.

**1.2** **OPERASI-OPERASI DASAR LOGIKA YANG DIJELASKAN TABEL KEBENARAN**

**Operasi INVERS (NOT)*****Operasi INVERS / NOT*** merupakan suatu operasi yang menghasilkan keluaran nilai kebalikannya. Operasi INVERS / NOT dilambangkan dengan tanda ( ¯ ) diatas variabel atau tanda single apostrope ( ‘ ). Operasi ini akan mengubah logik 1(benar) menjadi 0(salah) dan sebaliknya, akan mengubah logik 0(salah) menjadi logik 1(benar).

A               a'  
  
0                1  
  
1                 0

**Operasi AND**  
***Operasi AND*** merupakan operasi boolean yang yang akan memghasilkan nilai 1 ketika dipasangkan dengan 1 pula.           Operasi AND dilambangkan dengan dot ( . ). Operasi ini hanya akan menghasilkan nilai benar jika kedua variabel bernilai benar, selain itu akan bernilai salah.

**Operasi OR**  
***Operasi OR*** merupakan operasi yang hanya akan menghasilkan nilai benar(1) jika salah satu variabelnya bernilai benar(1) serta akan menghasilkan nilai salah jika kedua variabelnya bernilai salah. Operasi OR dilambangkan dengan plus (+).  
**Operasi Turunan / Operasi logika NOR**  
***Operasi NOR*** merupakan perpaduan dari operasi OR dan INVERS / NOT. Operasi NOR kan menghasilkan keluaran OR yang di inverskan. Operasi NOR mempunyai dua buah lambang yaitu lambang OR (+) dan INVERS / NOT ( ‘ ).  
**Operasi logika NAND *Operasi NAND*** merupakan perpaduan dari operasi AND dan INVERS / NOT. Operasi NAND akan menghasilkan keluaran AND yang di inverskan. Operasi NAND mempunyai dua buah lambang yaitu lambang AND ( . ) dan INVERS / NOT ( ‘ ).  
**Operasi logika EXOR**  
***EXOR*** berarti exklusive OR berarti “yang satu atau yang satunya tapi tidak keduanya”. Operasi XOR akan menghasilkan keluaran 1(benar) jika jumlah masukan yang bernilai 1(benar) berjumlah ganjil. Operasi XOR merupakan hasil dari (a’.b) + (a.b’) atau biasa ditulis a    b. Tabel kebenaran untuk operasi XOR:

**Operasi logika EXNOR *EXNOR*** berarti exklusive NOR berarti “yang satu atau yang satunya tapi tidak keduanya”. Operasi ini akan menghasilkan keluaran 1(benar) jika jumlah masukan yang bernilai 1(benar) berjumlah genap atau tidak ada sama sekali. Operasi XOR merupakan hasil dari a’+b .  a+b’ atau biasa ditulis a’     b’ atau (a      b)’.

## 1.3 HUKUM ALJABAR BOOLEAN

Dengan menggunakan Hukum Aljabar Boolean ini, kita dapat mengurangi dan menyederhanakan Ekspresi Boolean yang kompleks sehingga dapat mengurangi jumlah Gerbang Logika yang diperlukan dalam sebuah rangkaian Digital Elektronika.

**Dibawah ini terdapat 6 tipe Hukum yang berkaitan dengan Hukum Aljabar Boolean**

### **Hukum Komutatif (Commutative Law)**

Hukum Komutatif menyatakan bahwa penukaran urutan variabel atau sinyal Input tidak akan berpengaruh terhadap Output Rangkaian Logika.

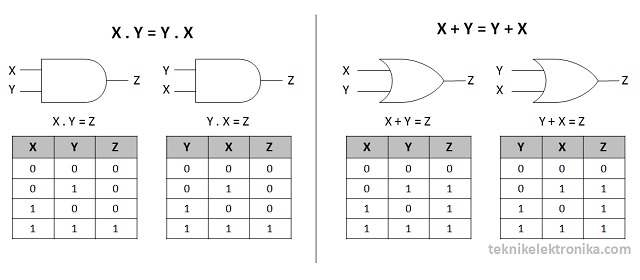
Contoh :

**Perkalian (Gerbang Logika AND)**

X.Y = Y.X

**Penjumlahan (Gerbang Logika OR)**

X+Y = Y+X

Catatan : Pada penjumlahan dan perkalian, kita dapat menukarkan posisi variabel atau dalam hal ini adalah sinyal Input, hasilnya akan tetap sama atau tidak akan mengubah keluarannya.**[](https://teknikelektronika.com/wp-content/uploads/2016/01/Hukum-Komutatif.jpg?x27780)**

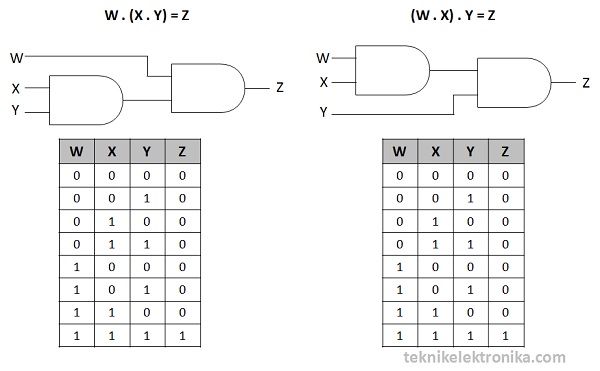
### Hukum Asosiatif (Associative Law)

Hukum Asosiatif menyatakan bahwa urutan operasi logika tidak akan berpengaruh terhadap Output Rangkaian Logika.

Contoh :

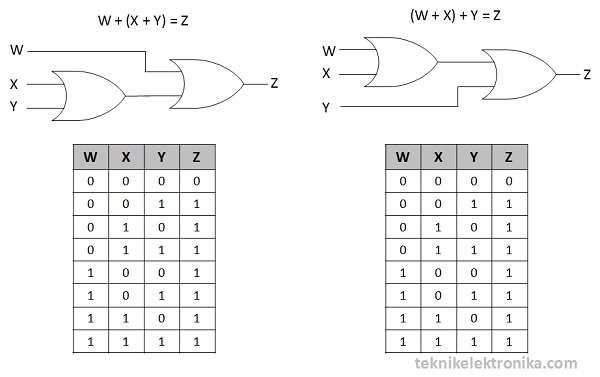
**Perkalian (Gerbang Logika AND)**

W . (X . Y) = (W . X) . Y

**[](https://teknikelektronika.com/wp-content/uploads/2016/01/Hukum-Asosiatif.jpg?x27780)**

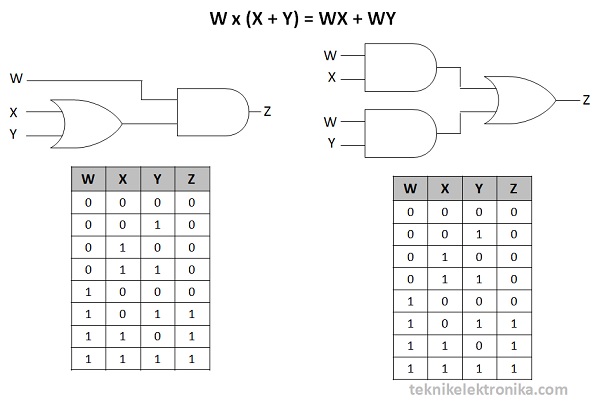
**Penjumlahan (Gerbang Logika OR)**

W + (X + Y) = (W + X) + Y

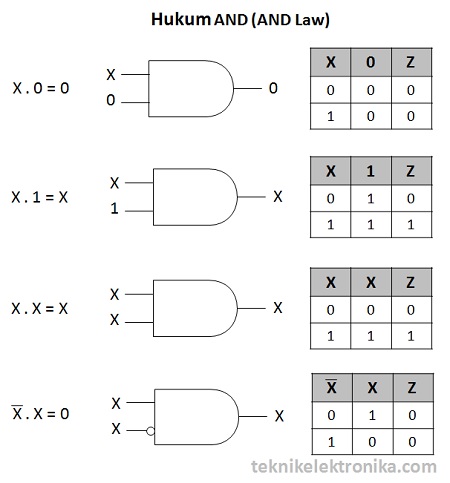
**[](https://teknikelektronika.com/wp-content/uploads/2016/01/Hukum-Asosiatif-OR.jpg?x27780)**

Catatan : Pada penjumlahan dan perkalian, kita dapat mengelompokan posisi variabel dalam hal ini adalah urutan operasi logikanya, hasilnya akan tetap sama atau tidak akan mengubah keluarannya. Tidak peduli yang mana dihitung terlebih dahulu, hasilnya tetap akan sama. Tanda kurung hanya sekedar untuk mempermudah mengingat yang mana akan dihitung terlebih dahulu.

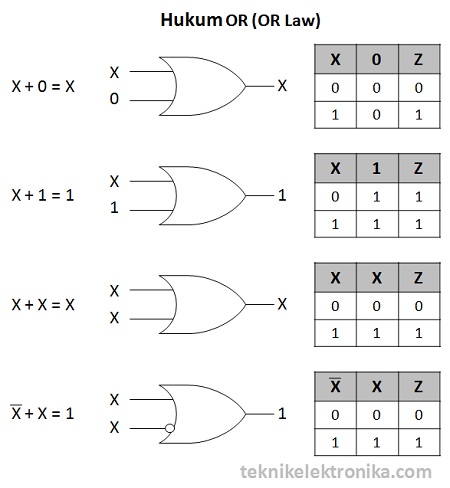
### Hukum Distributif

Hukum Distributif menyatakan bahwa variabel-variabel atau sinyal Input dapat disebarkan tempatnya atau diubah urutan sinyalnya, perubahan tersebut tidak akan mempengaruhi.**[](https://teknikelektronika.com/wp-content/uploads/2016/01/Hukum-Distributif.jpg?x27780)**

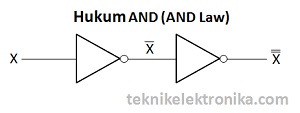
### Hukum AND (AND Law)

Disebut dengan Hukum AND karena pada hukum ini menggunakan Operasi Logika AND atau perkalian. Berikut ini contohnya :**[](https://teknikelektronika.com/wp-content/uploads/2016/01/Hukum-AND.jpg?x27780)**

### Hukum OR (OR Law)

Hukum OR menggunakn Operasi Logika OR atau Penjumlahan. Contoh:**[](https://teknikelektronika.com/wp-content/uploads/2016/01/Hukum-OR-.jpg?x27780)**

### Hukum Inversi (Inversion Law)

Hukum Inversi menggunakan Operasi Logika NOT. Hukum Inversi ini menyatakan jika terjadi Inversi ganda (kebalikan 2 kali) maka hasilnya akan kembali ke nilai aslinya.**[](https://teknikelektronika.com/wp-content/uploads/2016/01/Hukum-NOT.jpg?x27780)**

Jadi, jika suatu Input (masukan) diinversi (dibalik) maka hasilnya akan berlawanan. Namun jika diinversi sekali lagi, hasilnya akan kembali ke semula.

**BAB IV**

# **PENYEDERHANAAN RANGKAIAN LOGIKA**

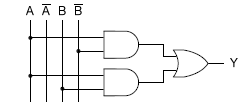
**2.1 Penyederhanaan Secara Aljabar**

Bentuk persamaan logika *sum of minterm* dan *sum of maxterm* yang diperoleh dari tabel kebenaran umumnya jika diimplementasikan ternyata merupakan bentuk implementasi yang tidak efisien. Dalam hal ini, setiap persamaan logika yang akan diimplementasikan perlu diuji terlebih dahulu ke dalam bentuk yang paling minimum.

Tahap minimalisasi rangkaian logika diperlukan agar diperoleh rangkaian dengan fungsi yang sama namun menggunakan gerbang yang paling sedikit. Rangkaian dengan jumlah gerbang yang paling sedikit akan lebih murah harganya, dan dari segi tata letak komponennya akan lebih sederhana.

Salah satu cara untuk menguji bentuk minimum dari suatu persamaan logika dan meminimalkannya adalah dengan menggunakan aljabar Boole.

**Contoh 4.1** Sederhanakan rangkaian di bawah ini :



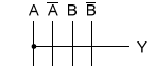
**Jawab :**

*Y* = *AB* + *AB* = *A*(*B* + *B*)

*Y* = *A*

Berdasarkan penyederhanaan persamaan di atas, rangkaian tersebut bisa

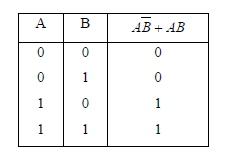
disederhanakan menjadi :



Sehingga dalam kasus ini, untuk mendapatkan keluaran cukup

menggunakan seutas kawat dan tidak memerlukan gerbang sama sekali.

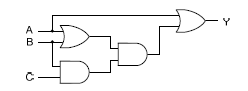
Pembuktian dengan mengunakan table kebenaran :

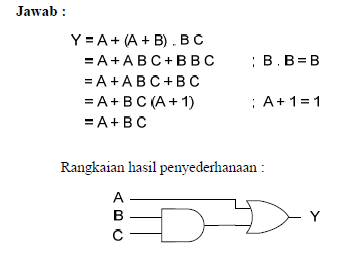


Untuk Y = *AB* + *AB* \_ Y = 1 jika **A=1 dan B=0** atau **A=1 dan B=1**

Dari pembuktian di atas, ternyata benar bahwa : Y = *AB* + *AB* = A

**Contoh 4.2** Sederhanakan rangkaian di bawah ini :





**2.2 Metode Peta Karnaugh (K-Map)**

Meskipun aljabar Boole merupakan suatu sarana yang berguna untuk

menyederhanakan pernyataan logika, belum dapat dipastikan bahwa pernyataan

yang disederhanakan dengan aljabar Boole itu merupakan pernyataan yang paling

sederhana. Prosedur meminimumkan itu agak sulit dirumuskan karena tidak

adanya aturan yang jelas untuk menentukan langkah manipulasinya.

61

Metode peta karnaugh memberikan suatu prosedur yang mudah dan

langsung dalam proses penyederhanaan fungsi Boole. Metode pemetaan itu

awalnya diusulkan oleh Veitch, lalu dimodifikasi oleh Karnaugh. Itulah alasannya

namanya dikenal sebagai diagram Veitch atau Peta Karnaugh (K-Map).

**2.3 Pembentukan Peta Karnaugh**

Peta Karnaugh yang digunakan dalam penyederhanaan fungsi Boole

merupakan sebuah tabel kebenaran dengan bentuk lain. Oleh karena itu jumlah

kombinasi yang ada dalam suatu tabel kebenaran sama dengan jumlah kombinasi

yang diperlukan oleh peta tersebut. Jadi, untuk *n* variabel input akan

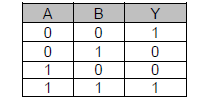
menghasilkan 2n kombinasi minterm yang diwakili dalam bentuk segiempat. Peta

Karnaugh untuk 2 variabel memerlukan 22 atau 4 segiempat, peta karnaugh 3

variabel mempunyai 23 atau 8 segiempat, dan seterusnya.

**Peta 2 Variabel**

Contoh tabel kebenaran yang mempunyai 2 variabel input :



Cara menyusun peta karnaugh untuk tabel kebenaran di atas :

a. Karena tabel kebenaran memilik 2 variabel input, buat 4 segiempat, dimana

kolom vertical diisi dengan *A* dan *A*, sedangkan baris horizontal diisi dengan

*B* dan *B*.

b. Carilah output yang bernilai 1, lalu buat persamaan *sum of minterm* untuk

tabel tersebut.

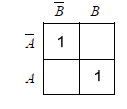
*Y* = *AB* + *AB*

Dari persamaan di atas, diketahui bahwa keluarannya akan bernilai 1 untuk

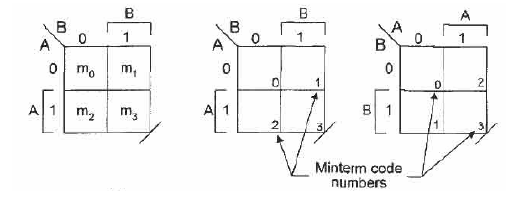
*A B* = 1 dan *A B* = 1. Tuliskan angka satu pada tempat yang bersesuaian.

Maka bentuk peta karnaugh 2 variabel untuk tabel kebenaran di atas

adalah sebagai berikut :

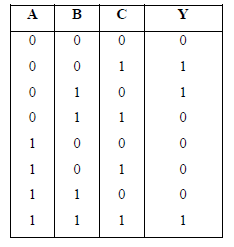


Peletakan posisi suku minterm untuk peta 2 variabel adalah sebagai berikut :



**Peta 3 Variabel**

Contoh tabel kebenaran yang mempunyai 3 variabel input :

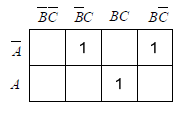


Jumlah kotak untuk 3 variabel input adalah 23 atau 8. Persamaan sum of minterm

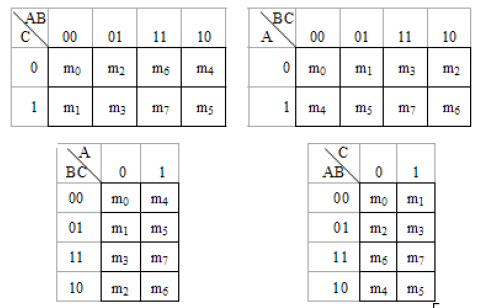
untuk table di atas adalah :

*Y* = *ABC* + *ABC* + *ABC*

Maka bentuk peta karnaughnya akan menjadi :

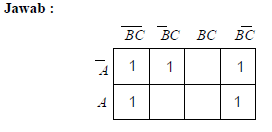


Peletakan posisi suku minterm untuk peta 3 variabel adalah sebagai berikut :



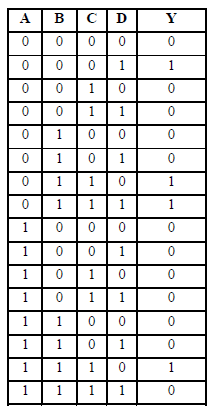
Tuangkanlah persamaan berikut ke dalam peta karnaugh !

FABC = \_ m (0,1,2,4,6)



**Peta 4 Variabel**

Contoh tabel kebenaran yang mempunyai 4 variabel input :

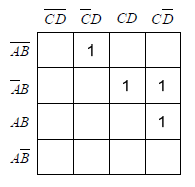


Jumlah kotak untuk 4 variabel input adalah 24 atau 16. Persamaan *sum of minterm*

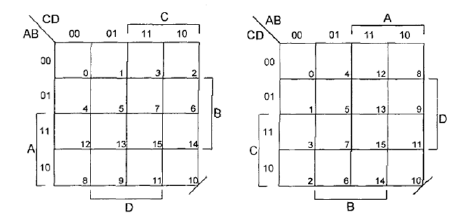
untuk tabel 4.3 di atas adalah :

*Y* = *ABCD* + *ABCD* + *ABCD* + *ABCD*

Maka bentuk peta karnaughnya akan menjadi :

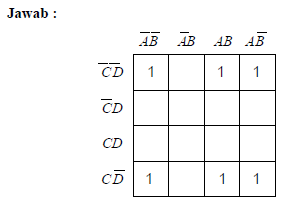


Peletakan posisi suku minterm untuk peta 4 variabel adalah sebagai berikut :



Tuangkanlah persamaan berikut ke dalam peta karnaugh !

FABCD = \_ m (0,2,8,10,12,14 )



**Penyederhanaan Karnaugh**

Peta karnaugh dapat digunakan untuk menyederhanakan rangkaian logika.

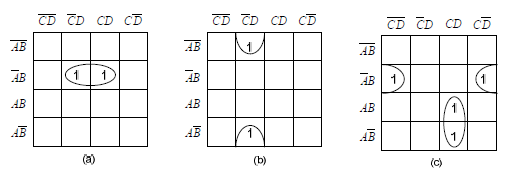
Tetapi sebelum memahami bagaimanan hal tersebut terjadi, perlu dipahami

terlebih dahulu mengenai *pasangan*, *kuad* (kelompok berempat), dan *oktet*

(pasangan berdelapan).

**Pasangan**

Gambar berikut menunjukkan bentuk pasangan pada peta karnaugh :



Peta dari gambar 4.1(a) berisi satu pasangan angka 1 yang saling

berdekatan dalam arah horizontal. Angka 1 pertama menyatakan *ABCD*, dan

angka 1 yang kedua menyatakan *ABCD* . Bila kita melihat pada angka 1 pertama

dan angka 1 kedua, ada satu variabel yang mengalami perubahan dari bentuk *C*

menjadi C. Untuk hal ini kita bisa menghapus variabel yang berubah tersebut,

sedangkan variabel yang tidak berubah diambil sebagai bentuk yang telah

disederhanakan. Sehingga bentuk persamaan yang telah disederhanakan untuk

gambar 4.1(a) adalah : *Y* = *ABD* . Berikut bukti secara aljabar :

Persamaan sum of minterm untuk gambar 4.1(a) adalah :

*Y* = *ABCD* + *ABCD*

Faktorisasi menghasilkan :

*Y* = *ABD*(*C* + *C*)

Karena (*C* +*C*) =1, maka persamaan di atas dapat direduksi menjadi :

*Y* = *ABD*

Untuk memudahkan identifikasi, biasanya angka 1 yang berdekatan diberi

tanda lingkaran. Dengan cara seperti ini kita lebih mudah untuk mengenali adanya

variabel dan komplemennya yang tidak muncul lagi dalam persamaan Boole.

Selanjutnya bayangkan kita mengambil peta karnaugh dan menggulungnya

sedemikian rupa, sehingga tepi atas bersentuhan dengan tepi bawah (seperti

terlihat pada gambar 4.1(b), dan tepi kiri bersentuhan dengan tepi kanan (seperti

terlihat pada gambar 4.1(c), maka hal tersebut juga akan membentuk pasangan.

Hal itu disebut dengan **penggulungan peta (Rolling)**.

Pada gambar 4.1(b), variabel *A* berubah menjadi *A*, sehingga variabel

tersebut bisa dihilangkan. Persamaan yang diambil menjadi bentuk yang telah

disederhanakan adalah minterm yang terdiri dari variabel-variabel yang tetap atau

tidak mengalami perubahan. Maka persamaan yang telah disederhanakan untuk

gambar 4.1(b) adalah : *Y* = *BCD*

Jika di dalam peta karnaugh terdapat lebih dari satu pasangan, maka

dilakukan operasi OR untuk semua minterm yang telah disederhanakan. Pada

gambar 4.1(c) terdapat dua buah pasangan, pasangan pertama *ABCD* dan

*ABCD*, pasangan kedua *ABCD* dan *ABCD* . Untuk pasangan pertama dapat

disederhanakan menjadi *ABD* dan pasangan kedua disederhanakan menjadi

*ACD* , sehingga persamaan Boole-nya adalah :

*Y* = *ABD* + *ACD*

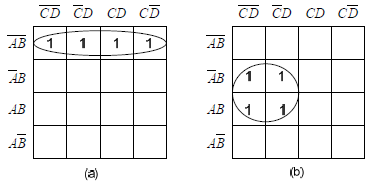
**2.4 Kuad**

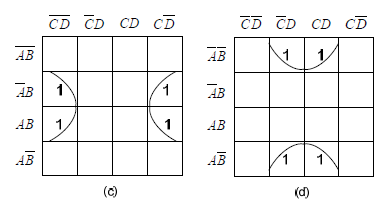
Kuad adalah kelompok yang terdiri dari 4 buah angka 1 yang berdekatan.

Dalam kenyataannya, kehadiran sebuah kuad berarti terhapusnya dua variabel

beserta komplemennya dari persamaan Boole yang bersangkutan. Berikut contoh

bentuk kuad yang dimungkinkan pada peta karnaugh :





Proses penyederhanaan persamaannya sama dengan yang telah dijelaskan

pada pasangan, yaitu : hilangkan variabel yang berubah atau yang berbeda, dan

ambil variabel yang tetap atau sama. Sehingga dengan cara tersebut :

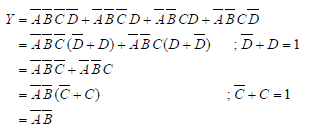
Persamaan untuk peta gambar 4.2(a) menjadi : *Y* = *AB*

Persamaan untuk peta gambar 4.2(b) menjadi : *Y* = *BC*

Persamaan untuk peta gambar 4.2(c) menjadi : *Y* = *BD*

Persamaan untuk peta gambar 4.2(d) menjadi : *Y* = *BD*

Pembuktian secara aljabar untuk gambar 4.2(a) :



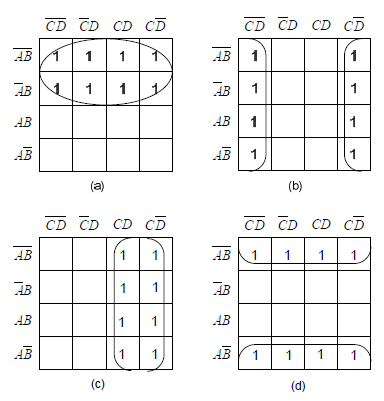
**2.5 Oktet**

Oktet adalah kelompok dari delapan angka 1 yang berdekatan. Sebuah

oktet berarti menghapus tiga variabel dan komplemen-komplemennya dari

persamaan Boole yang bersangkutan. Berikut contoh bentuk kuad yang

dimungkinkan pada peta karnaugh :



Proses penyederhanaan persamaannya juga sama dengan yang telah

dijelaskan pada pasangan dan kuad, yaitu : hilangkan variabel yang berubah atau

yang berbeda, dan ambil variabel yang tetap atau sama. Sehingga dengan cara

tersebut :

Persamaan untuk peta gambar 4.3(a) menjadi : *Y* = *A*

Persamaan untuk peta gambar 4.3(b) menjadi : *Y* = *D*

Persamaan untuk peta gambar 4.3(c) menjadi : *Y* = *C*

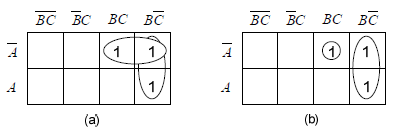
Persamaan untuk peta gambar 4.3(d) menjadi : *Y* = *B*

**Kelompok yang Bertumpang Tindih (*Overlapping*)**

Dalam melingkari kelompok dalam peta Karnaugh, dimungkinkan untuk

menggunakan angka 1 tertentu lebih dari satu kali., seperti yang terlihat pada

gambar 4.7 berikut :



Persamaan yang telah disederhanakan untuk kelompok yang bertumpang

tindih gambar 4.4(a) adalah : *Y* = *AB* + *BC* , sedangkan persamaan untuk gambar

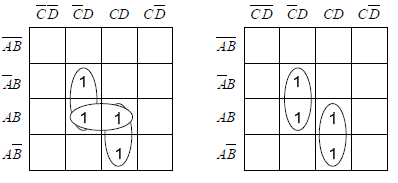
4.4(b) adalah : *Y* = *ABC* + *BC* . Dapat dilihat dari 2 persamaan tersebut, bahwa

penyederhanaan tanpa menggunakan konsep bertumpang tindih akan

menghasilkan persamaan yang lebih kompleks.

**Kelompok Kelebihan (*Redundant*)**

Kelompok kelebihan (redundant dapat dilihat pada gambar 4.8 berikut ini :



Pada gambar 4.5(a), terlihat bahwa ada tiga pasangan yang telah

dilingkari. Persamaan Boole-nya adalah : *Y* = *BCD* + *ACD* + *ABD* . Sedangkan

persamaan untuk gambar 4.5(b) adalah : *Y* = *BCD* + *ACD* . Dapat dilihat dari 2

persamaan tersebut bahwa dalam penyederhanaan, jika angka 1-nya telah habis

dikelompokkan, maka jangan buat pengelompokkan untuk angka 1 kelompok

yang satu dengan angka 1 kelompok yang lain.

**Keadaan Tak Acuh (*Don’t Care Condition*)**

Angka 1 dan 0 dalam table kebenaran menunjukkan bahwa kombinasi

variable input akan membuat fungsi outputnya bernilai 1 atau 0. Dalam

prakteknya, terdapat kombinasi variable input yang tidak pernah ada. Sebagai

contoh, kode BCD hanya menggunakan kombinsi variable input 0000 sampai

dengan 1001 (mengkodekan angka decimal 0 sampai dengan 9), sedangkan 1010

sampai dengan 1111 tidak boleh muncul dalam operasi normalnya. Sehingga

keluaran dari fungsi 1010 sampai dengan 1111 tidak perlu diperhatikan karena

dijamin tidak akan pernah ada, keadaan ini disebut dengan **Keadaan Acuh (*Don’t***

***Care Condition*)**

Keadaan don’t care tersebut dimanfaatkan dalam Peta Karnaugh untuk

mendapatkan penyederhanaan lebih lanjut pada fungsinya. Untuk membedakan

keadaan don’t care ini dengan 1 dan 0, digunakan tanda silang (X).

Dalam pengelompokan peta Karnaugh, X hanya digunakan untuk

menyumbang pengelompokan angka 1 yang lebih luar. Sehingga X tidak perlu

digunakan jika tidak menyumbang untuk pengelompokan angka 1 yang lebih luas.

Jadi, pemilihannya hanya tergantung pada penyederhanaan yang paling

menguntungkan. Untuk lebih jelasnya dapat dilihat pada contoh 4.4 dan 4.5.

Berdasarkan penjelasan di atas, dapat disimpulkan bahwa langkah-langkah

untuk penyederhanaan rangkaian logika dengan menggunakan peta Karnaugh

adalah :

1. Masukkan output yang bernilai 1 ke dalam peta Karnaugh untuk setiap

minterm yang bersesuaian pada tabel kebenaran.

2. Melingkari oktet, kuad, dan pasangan yang ada pada peta. Jangan lupa

melakukan proses penggulungan dan penandaan kelompok-kelompok

yang bertumpang tindih untuk memperoleh pengelompokan yang sebesar

mungkin. Jika perlu gunakan bit don’t care untuk pengelompokan yang

leih besar.

3. Melingkari sisa-sisa angka 1 yang terisolasi atau yang tidak bisa

dikelompokkan.

4. Menghapus kelompok-kelompok kelebihan (jika ada).72

5. Menuliskan persamaan Boole dalam pernyataan operasi OR dari hasil

penyederhanaan semua kelompok yang dilingkari

.

**2.6 Metode Tabulasi (Quine Mc Cluskey)**

Metode penyederhanaan dengan peta tidak mudah dilakukan untuk jumlah

variable yang besar. Metode tabulasi dapat mengatasi kesulitan tersebut. Metode

tabulasi pertama kali dirumuskan oleh Quine dan selanjutnya diperbaiki oleh

McCluskey, sehingga metode ini dikenal dengan metode Quine McCluskey.

Metode penyederhanaan dengan tabulasi terdiri dari dua bagian, yaitu :

1. Penentuan Prime Implicant

Mencari semua suku (term) yang merupakan calon untuk dicantumkan dalam

fungsi yang disederhanakan itu. Suku tersebut dinamakan prime implicant.

2. Pemilihan Prime Implicant

Memilih di antara semua suku prime implicant yang tersedia itu yang akan

memberikan pernyataan yang paling sederhana.

Metode Quine-McCluskey

1. Nyatakan tiap minterm dalam n peubah menjadi string bit yang panjangnya n, peubah komplemen →0, peubah bukan komplemen →1
2. Kelompokkan minterm berdasarkan jumlah 1 yang dimilikinya
3. Kombinasikan minterm dalam n peubah dengan kelompok lain yang jumlah 1 nya berbeda satu sehingga diperoleh bentuk prima yang terdiri dari n-1 peubah. Mintern yang dikombinasikan diberi tanda √
4. Kombinasikan minterm dalam n-1 peubah dengan kelompok lain yang jumlah 1 nya berbeda 1, sehingga diperoleh bentuk prima yang terdiri dari n-2 peubah
5. Teruskan langkah 4 sampai diperoleh bentuk prima yang sesederhana mungkin
6. Ambil semua bentuk prima yang tidak bertanda √. Buat tabel baru yang memperlihatkan minterm dari ekspresi Boolean semula yang dicakup oleh bentuk prima tersebut (tandai dengan x). Setiap minterm harus dicakup oleh paling sedikit satu buah bentuk prima
7. Pilih bentuk prima yang memiliki jumlah literal paling sedikit namun mencakup sebanyak mungkin minterm dari ekspresi Boolean semua.

Metode Quine-McCluskey

Langkah 7 terdiri dari langkah-langkah sebagai berikut:

a)

Tandai kolom-kolom yang mempunyai satu buah tanda x dengan tanda \* lalu beri tanda √ di sebelah kiri bentuk prima yang berasosiasi dengan tanda \* tersebut. Bentuk prima ini telah dipilih untuk fungsi Boolean sederhana

b)

Untuk setiap bentuk prima yang telah ditandai dengan √

, beri tanda minterm yang dicakup oleh bentuk prima tersebut dengan tanda

√

c)

Periksa apakah masih ada minterm yang belum dicakup oleh bentuk prima terpilih, jika ada peilih dari bentuk prima yang tersisa yang mencakup sebanyak mungkin minterm tersebut. Beri tanda √ bentuk prima yang dipilih itu serta minterm yang dicakupnya

d)

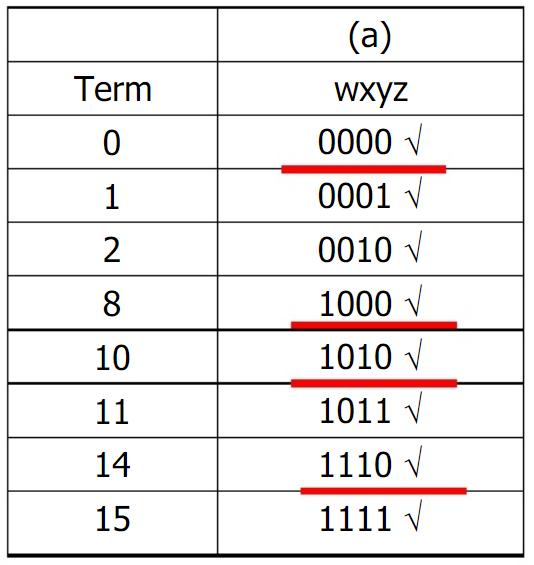
Ulangi langkah c sampai seluruh minterm sudah dicakup oleh semua bentuk prima

Sederhanakan fungsi Boolean f(v,w,x,y,z) =∑(0,1,2,8,10,11,14,15)

Nyatakan tiap minterm dalam n peubah menjadi string bit yang panjangnya n, peubah komplemen

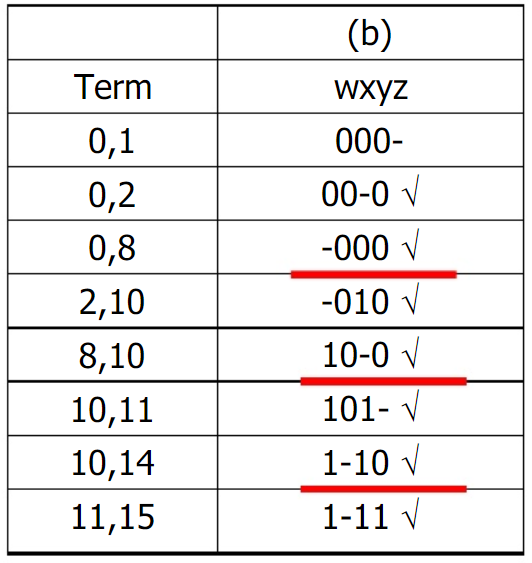
→ 0, peubah bukan komplemen → 1

Kelompokkan minterm berdasarkan jumlah 1 yang dimilikinya

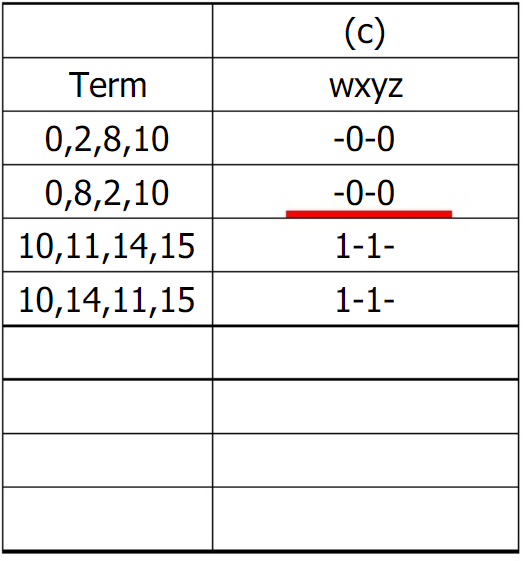


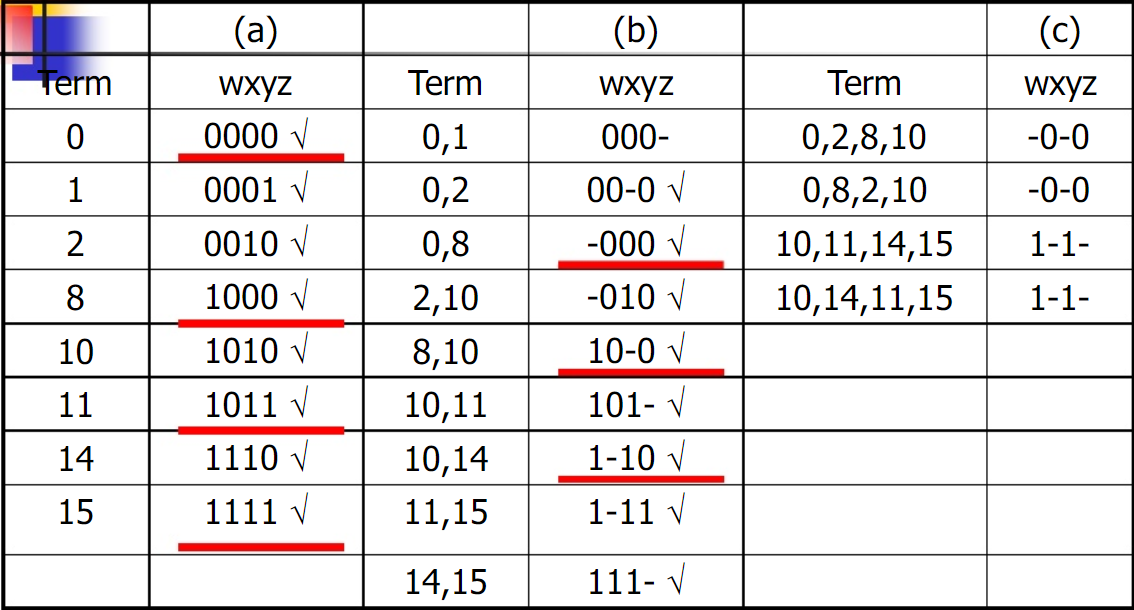
Kombinasikan minterm dalam n peubah dengan kelompok lain yang jumlah 1 nya berbeda satu sehingga diperoleh bentuk prima yang terdiri dari n-1 peubah.

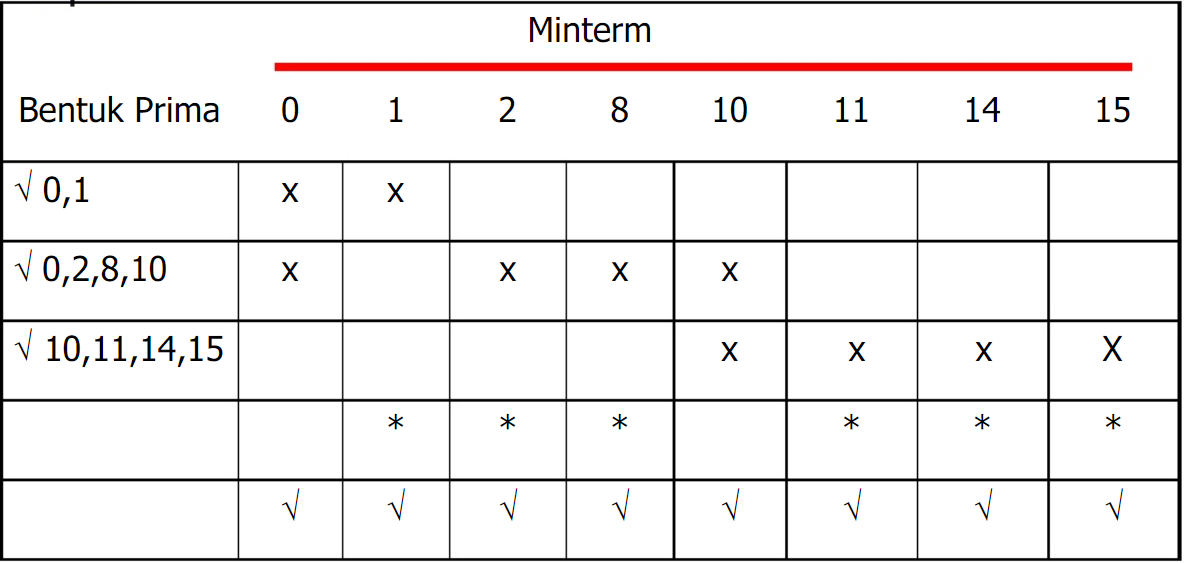
Mintern yang dikombinasikan diberi tanda√



Kombinasikan minterm dalam n-1 peubah dengan kelompok lain yang jumlah 1 nya berbeda 1, sehingga diperoleh bentuk prima yang terdiri dari n-2 peubah







Bentuk Prima yang terpilih adalah:

* 0,1
* 0,2,8,10
* 10,11,14,15
* Maka f(w,x,y,z) =

w’x’y+x’z’+wy

Yang bersesuaian dengan:

* w’x’y
* X’z’
* Wy

**BAB V**

**PENUTUP**

**3.1 Kesimpulan**

Penyederhanaan rangkain berfungsi untuk menyederhanaan suatu rangkaian supaya dapat melakukan pekerjaan menjadi lebih cepat dan tidak memakan waktu yang lama.

Di penyederhanaan rangkaian ini menggunakan 3 metode penyederhanaan

* 1. Aljabar Boolean
  2. K-Map
  3. Metode Quine-McCluskey

Yang membuat suatu rangkaian menjadi lebih sederhana

**DAFTAR PUSTAKA**

**DAFTAR PUSTAKA**

**https://nalasumarna.blogspot.com/2017/09/pengertian-aljabar-boolean.html**

**https://teknikelektronika.com/pengertian-aljabar-boolean-hukum-aljabar-boolean/**

**https://smallpdf-production-files.s3.eu-west-1.amazonaws.com/606e664b412b5598cc216f29b1d431b0.docx?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Cred**

**https://www.academia.edu/8418111/Pertemuan\_6\_metode\_tabulasi**