BAB I

GERBANG LOGIKA DASAR & ALJABAR BOOLEAN

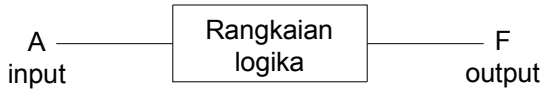
A. Tabel Kebenaran (Truth Table)

Tabel kebenaran merupakan tabel yang menunjukkan pengaruh pemberian level logika pada input suatu rangkaian logika terhadap keadaan level logika outputnya. Melalui tabel kebenaran dapat diketahui watak atau karakteristik suatu rangkaian logika. Oleh karena itu, tabel kebenaran mencerminkan watak atau karakteristik suatu rangkaian logika. Tabel kebenaran harus memuat seluruh kemungkinan keadaan input tergantung pada jumlah variabel input atau jumlah saluran input dari suatu rangkaian logika, dan mengikuti rumus :

Jumlah seluruh kemungkinan input = 2^n, dengan n merupakan jumlah variabel atau saluran input rangkaian .

Contoh :

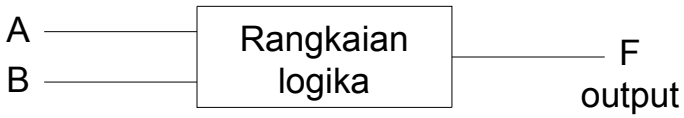
1. Rangkaian logika dengan 1 variabel input, maka jumlah seluruh kemungkinan input = $2^1 = 2$



Tabel kebenaran:

Input (A)	Output (F)
0	…..
1	…..

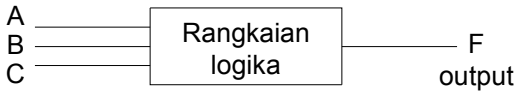
2. Rangkaian logika dengan 2 variabel input, maka jumlah seluruh kemungkinan input = $2^2 = 4$



Tabel kebenaran:

Input		Output
A	B	F
0	0	…..
0	1	…..
1	0	…..
1	1	…..

3. Rangkaian logika dengan 3 variabel input, maka jumlah seluruh kemungkinan input = $2^3 = 8$



Tabel kebenaran:

Input			Output
A	B	C	F
0	0	0	…..
0	0	1	…..
0	1	0	…..
0	1	1	…..
1	0	0	…..
1	0	1	…..
1	1	0	…..
1	1	1	…..

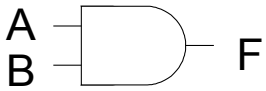
B. Gerbang Logika Dasar

Gerbang-gerbang dasar logika merupakan elemen rangkaian digital dan rangkaian digital merupakan kesatuan dari gerbang-gerbang logika dasar yang membentuk fungsi pemrosesan sinyal digital. Gerbang dasar logika terdiri dari 3 gerbang utama, yaitu AND

Gate, OR Gate, dan NOT Gate. Gerbang lainnya seperti NAND Gate, NOR Gate, EX-OR Gate dan EX-NOR Gate merupakan kombinasi dari 3 gerbang logika utama tersebut.

1. AND Gate

Gerbang AND merupakan salah satu gerbang logika dasar yang memiliki 2 buah saluran masukan (input) atau lebih dan sebuah saluran keluaran (output). Suatu gerbang AND akan menghasilkan sebuah keluaran biner tergantung dari kondisi masukan dan fungsinya. Prinsip kerja dari gerbang AND adalah kondisi keluaran (output) akan berlogic 1 bila semua saluran masukan (input) berlogic 1. Selain itu output akan berlogic 0. Simbol gerbang logika AND 2 input :



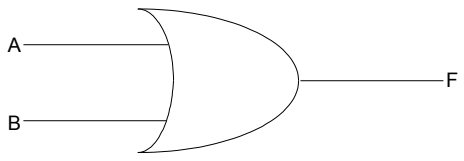
dengan persamaan Boolean fungsi AND adalah F = A.B (dibaca F = A AND B).
Tabel kebenaran:

input		Output
A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

2. OR Gate

Gerbang OR merupakan salah satu gerbang logika dasar yang memiliki 2 buah saluran masukan (input) atau lebih dan sebuah saluran keluaran (output). Berapapun jumlah saluran masukan yang dimiliki oleh sebuah gerbang OR, maka tetap memiliki prinsip kerja yang sama dimana kondisi keluarannya akan berlogic 1 bila salah satu atau semua saluran masukannya berlogic 1. Selain itu output berlogic 0.

Simbol gerbang logika OR 2 input :



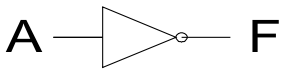
dengan persamaan Boolean fungsi OR adalah F = A+B (dibaca F = A OR B).

Tabel kebenaran:

input		Output
A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

3. NOT Gate

Gerbang NOT sering disebut dengan gerbang inverter. Gerbang ini merupakan gerbang logika yang paling mudah diingat. Gerbang NOT memiliki 1 buah saluran masukan (input) dan 1 buah saluran keluaran (output). Gerbang NOT akan selalu menghasilkan nilai logika yang berlawanan dengan kondisi logika pada saluran masukannya. Bila pada saluran masukannya berlogic 1 maka pada saluran keluarannya akan berlogic 0 dan sebaliknya. Simbol gerbang logika NOT :

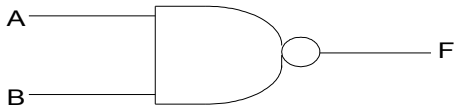


Tabel kebenaran:

Input (A)	Output (F)
0	1
1	0

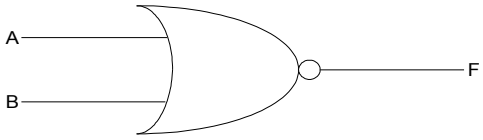
4. NAND Gate

Gerbang NAND merupakan kombinasi dari gerbang AND dengan gerbang NOT dimana keluaran gerbang AND dihubungkan ke saluran masukan dari gerbang NOT. Karena keluaran dari gerbang AND di"NOT"kan maka prinsip kerja dari gerbang NAND merupakan kebalikan dari gerbang AND. Outputnya merupakan komplemen atau kebalikan dari gerbang AND, yakni memberikan keadaan level logic 0 pada outputnya jika dan hanya jika keadaan semua inputnya berlogika 1. Simbol gerbang logika NAND 2 input :



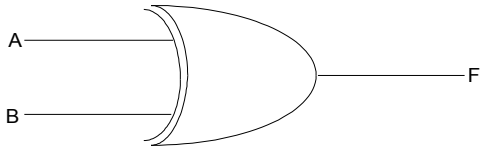
4. NOR Gate

Sama halnya dengan NAND Gate, gerbang NOR merupakan kombinasi dari gerbang OR dengan gerbang NOT dimana keluaran gerbang OR dihubungkan ke saluran masukan dari gerbang NOT. Karena keluaran dari gerbang OR di"NOT"kan maka prinsip kerja dari gerbang NOR merupakan kebalikan dari gerbang OR. Outputnya merupakan komplemen atau kebalikan dari gerbang OR, yakni memberikan keadaan level logic 0 pada outputnya jika salah satu atau lebih inputnya berlogika 1. Simbol gerbang logika NOR 2 input :



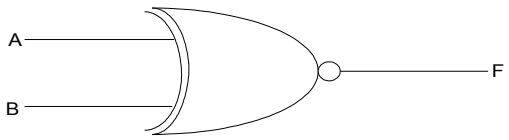
5. EX-OR Gate

EX-OR singkatan dari Exclusive OR dimana jika input berlogic sama maka output akan berlogic 0 dan sebaliknya jika input berlogic beda maka output akan berlogic 1. Simbol gerbang logika EX-OR 2 input :



6. EX-NOR

EX-NOR gate adalah kebalikan dari EX-OR gate dimana jika input berlogic sama maka output akan berlogic 1 dan sebaliknya jika input berlogic beda maka output akan berlogic 0. Simbol gerbang logika EX-NOR 2 input :



BAB II
RANGKAIAN LOGIKA KOMBINASI

A. Pengertian Logika Kombinasi

Logika kombinasi merupakan salah satu jenis rangkaian logika yang keadaan outputnya hanya tergantung pada kombinasi-kombinasi inputnya saja.

B. Bentuk-bentuk Persamaan Logika

Selain menggunakan symbol elemen logika, deskripsi rangkaian logika kombinasi dapat dilakukan dengan menggunakan persamaan logika. Secara umum persamaan logika diklasifikasikan ke dalam 2 bentuk, yakni *Sum Of Product (SOP)* dan *Product Of Sum (POS)*. Dari masing-masing bentuk persamaan tersebut dapat diklasifikasikan lagi menjadi bentuk standar dan tidak standar.

1. Bentuk Sum Of Product (SOP)

SOP merupakan persamaan logika yang mengekspresikan operasi OR dari suku-suku berbentuk operasi AND. Secara sederhana dapat dikatakan bahwa SOP adalah bentuk persamaan yang melakukan operasi OR terhadap AND. Bentuk SOP ini terdiri dari 2 macam, yaitu SOP standar dan SOP tidak standar. SOP standar adalah persamaan logika SOP yang setiap sukunya mengandung semua variabel input yang ada, sedangkan SOP standar merupakan persamaan logika SOP yang tidak setiap sukunya mengandung semua variabel input. Pada bentuk SOP standar, setiap sukunya dinamakan *minterm*, disingkat dengan m(huruf kecil). *Minterm* bersifat unik, yakni untuk semua kombinasi input yang ada hanya terdapat satu kombinasi saja yang menyebabkan suatu *minterm* bernilai 1. Dengan kata lain, suatu persamaan logika dalam bentuk SOP, dapat dilihat dari outputnya yang berlogic 1. Tanda sigma (Σ) digunakan sebagai pengganti operator-operator penjumlahan (operasi logika OR).

2. Bentuk Product Of Sum (POS)

POS merupakan suatu persamaan logika yang mengekspresikan operasi AND dari suku-suku berbentuk operasi OR atau dengan kata lain POS adalah bentuk persamaan yang meakukan operasi AND terhadap OR. Bentuk POS ini terdiri dari 2 macam, yaitu POS standar dan POS tidak standar. POS standar adalah persamaan logika POS yang setiap sukunya mengandung semua variabel input yang ada, sedangkan POS standar merupakan persamaan logika POS yang tidak setiap sukunya mengandung semua variabel input. Pada bentuk POS standar, setiap sukunya dinamakan *maxterm*, disingkat

dengan M (huruf besar). Sama halnya dengan *minterm*, *maxterm* juga bersifat unik, yakni untuk semua kombinasi input yang ada hanya terdapat satu kombinasi saja yang menyebabkan suatu *maxterm* bernilai 0. Dengan kata lain, suatu persamaan logika dalam bentuk POS, dapat dilihat dari outputnya yang berlogic 0. Tanda phi (Π) digunakan sebagai pengganti operator-operator perkalian (operasi logika AND).

3. Penyederhanaan Secara Aljabar

Bentuk suatu persamaan logika baik dalam bentuk SOP maupun POS yang diperoleh dari tabel kebenaran umumnya jika diimplementasikan ternyata merupakan bentuk impelementasi yang tidak efisien. Oleh karena itu, setiap persamaan logika yang akan diimplementasikan ke dalam bentuk rangkaian logika pada dasarnya dapat dilakukan jika persamaan logika tersebut sudah dalam bentuk minimum, yaitu dengan tahap minimisasi. Tahap minimisasi merupakan suatu cara untuk memanipulasi atau menyederhanakan suatu persamaan logika dengan menggunakan teorema aljabar Boolean, diagram venn, karnaugh map, dam sebagainya. Dengan menyederhanakan suatu persamaan logika sebelum persamaan tersebut diimplementasikan ke dalam bentuk rangkaian, terdapat beberapa keuntungan yang dapat diperoleh, yaitu :

- Mengurangi jumlah komponen yang diperlukan.
- Mengurangi biaya yang diperlukan.
- Waktu yang diperlukan untuk menyusun rangkaian lebih sedikit.
    - Respon/tanggapan rangkaian menjadi lebih cepat karena delay/tundaan rangkaian berkurang.
    - Ukuran/dimensi fisik rangkaian lebih kecil.
    - Bobot rangkaian lebih ringan.
    - Rangkaian akan lebih mudah dianalisa.

4. Metode Karnaugh Map

Selain menggunakan teorema aljabar Boolean, agar suatu persamaan logika dengan cepat dapat diketahui sudah dalam bentuk minimum atau masih perlu

diminimumkan dapat digunakan metode Karnaugh Map. Keuntungan yang diperoleh dari penyederhanaan persamaan logika dengan menggunakan K-map ditinjau dari persamaan akhir yang dihasilkan selalu merupakan persamaan yang tersederhana. Pembahasan lebih lanjut tentang karnaugh map akan dijelaskan pada bab minimasi.

BAB III
TEKNIK MINIMISASI DAN IMPLEMENTASI

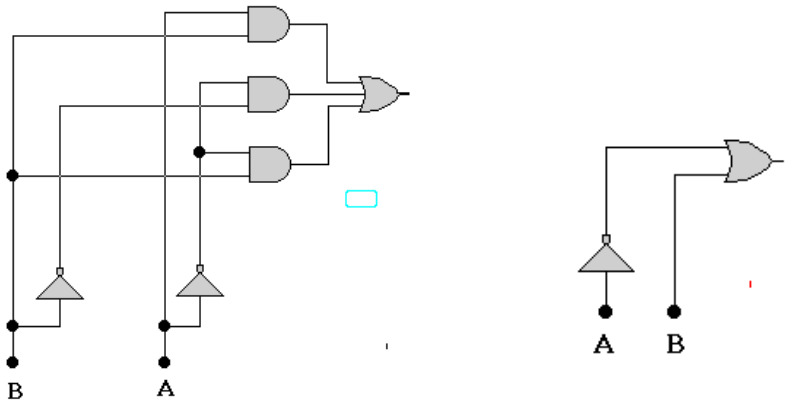
A. Teknik Minimasi

Teknik minimisasi dalam ilmu digital adalah suatu teknik yang digunakan untuk menyederhanakan suatu persamaan logika. Mengapa suatu persamaan logika perlu disederhanakan?

Suatu persamaan logika perlu disederhanakan agar jika persamaan logika itu kita buat menjadi sebuah rangkaian logika kita bisa ;

- Mengurangi jumlah komponen yang digunakan
- Mengurangi jumlah biaya yang diperlukan
- Mempersingkat waktu untuk merangkai
- Menghasilkan respon rangkaian lebih cepat karena delay rangkaian berkurang
- Memperkecil dimensi fisik rangkaian
- Menganalisa rangkaian dengan mudah

Berikut adalah contoh rangkaian yang belum diminimisasi dan rangkaian yang sudah diminimisasi.



Sebelum diminimisasi sesudah diminimisasi

Bagaimanakah cara menyederhanakan persamaan logika?

Berikut beberapa metoda untuk menyederhanakan persamaan suatu logika diantaranya ;

- Aljabar Boolean
- Diagram Venn

- Karnaugh Map
- Quinne -Mc.Cluskey

1. Teorema Aljabar Boolean

Aljabar Boolean sangat penting peranannya di dalam proses perancangan maupun analisis rangkaian logika. Untuk memperoleh hasil rancangan yang berupa suatu persamaan logika yang siap diimplementasikan, diperlukan tahap pemberlakuan kaidah-kaidah perancangan. Salah satunya adalah aljabar Boolean. Aljabar Boolean merupakan aljabar yang diberlakukan pada variabel yang bersifat diskrit, dan oleh karena itu, aljabar ini cocok diberlakukan pada variabel yang ada pada rangkaian logika. Terdapat 2 jenis teorema aljabar Boolean yakni teorema variabel tunggal dan teorema variabel jamak. Setiap teorema baik yang bersifat tunggal maupun jamak selalu memiliki teorema rangkapnya.

a. Sifat Idempoten (sama)
   - $X + X = X$
   - $X \bullet X = X$

b. Sifat Absorpsi (menghilanghkan)
   - $X + (X \bullet Y) = X$
   - $X \bullet (X + Y) = X$

c. Teorema Identitas
   - $X + Y = Y$
   - $X \bullet Y = Y$

   (Jika $X = Y$)

d. Teorema Komplemen
   - Jika $X + Y = 1$, atau
   - Jika $X \bullet Y = 0$,

   Maka $X = \overline{Y}$

e. Teorema Involution

- $X = \overline{\overline{X}}$

f. Teorema Van De Morgan

- $\overline{X + Y + Z} = \overline{X} \bullet \overline{Y} \bullet \overline{Z}$
- $\overline{X \bullet Y \bullet Z} = \overline{X} + \overline{Y} + \overline{Z}$

2. Postulate Huntington

a. Postulate 1

- $X + 0 = X \rightarrow X \bullet 1 = X$
- $X \bullet 0 = 0 \rightarrow X + 1 = 1$

b. Postulate 2

- $X + Y = Y + X$
- $X \bullet Y = Y \bullet X$

c. Postulate 3

- $X \bullet (Y + Z) = (X \bullet Y) + (X \bullet Z)$
- $X + (Y \bullet Z) = (X + Y) \bullet (X + Z)$

d. Postulate 4

- $X + (Y + Z) = (X + Y) + Z$
- $X \bullet (Y \bullet Z) = (X \bullet Y) \bullet Z$

e. Postulate 5

- $\overline{X} + X = 1$
- $\overline{X} \bullet X = 0$

3. Diagram Venn

Salah satu cara untuk memudahkan untuk melukiskan hubungan antara variable dalam aljabar boolean adalah dengan menggunakan diagram venn. Diagram ini terdiri dari sebuah segi empat yang didalamnya dilukis lingkaran-lingkaran yang mewakili variabelnya, satu lingkaran untuk setiap variabelnya. Masing-masing lingkaran itu diberi

nama menurut variable yang diwakilinya. Ditentukan bahwa semua titik diluar lingkaran itu tidak dimiliki oleh variable tersebut. Misalnya lingkaran dengan nama *A,* jika dalam lingkaran itu dikatakan bernilai 1, maka diluar a dikatakan bernilai 0. untuk dua lingkaran yang bertumpang tindih, terdapat empat daerah dalam segiempat tersebut.

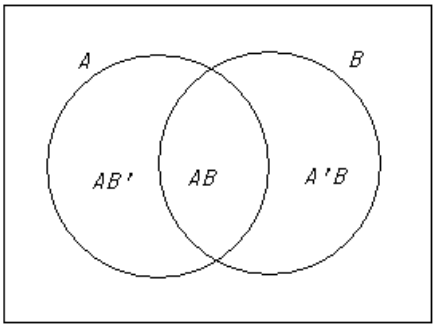
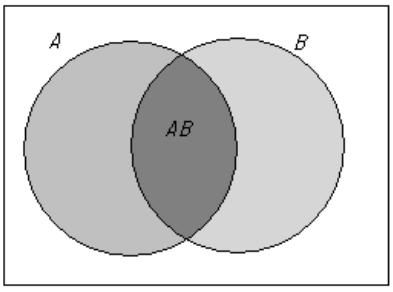
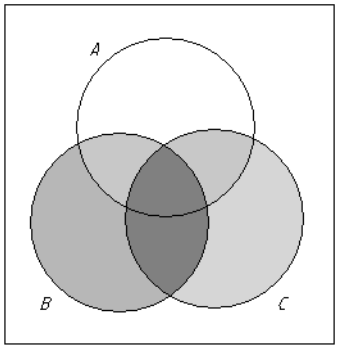


Diagram venn dapat digunakan untuk melukiskan postulate aljabar boole atau untuk membuktikan berlakunya aljabar Boolean. Gambar berikut menunjukan bahwa daerah yang dimiliki oleh *AB* terletak dalam lingkaran *A* sehingga $A+AB = A$.



Gambar berikut ,menunjukan hukum distributive $A(B+C) = AB+AC$

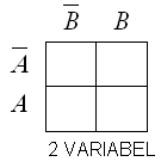


Dalam lingkaran itu tampak tiga lingkaran yang bertumpang tindih, satu untuk masing-masing variable A, B dan C. dengan demikian dimungkinkan untuk membedakan delapan daerah yang terpisah dalam diagram venn dengan variable itu. Dalam hal ini hokum distributiv dibuktikan dengan menunjukan bahwa daerah yang memotong lingkaran *A* dengan daerah yang meliputi *B* atau *C* adalah daerah yang sama yang dimiliki oleh *AB* atau *A*.

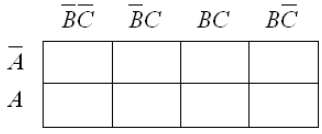
4. Karnaugh Map

Aturan penyederhanaan persamaan logika dengan K-map ;

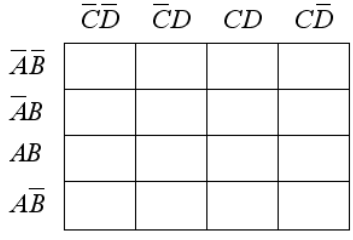
a. Untuk persamaan logika yang terdiri dari n variable diperlukan K-map dengan 2^n kotak. Penomoran kotak berurutan berdasarkan kode gray.



2 VARIABEL



3 VARIABEL



4.VARIABEL

	$\bar{D}\bar{E}\bar{F}$	$\bar{D}\bar{E}F$	$\bar{D}EF$	$\bar{D}E\bar{F}$	$DE\bar{F}$	DEF	$D\bar{E}F$	$D\bar{E}\bar{F}$
$\bar{A}\bar{B}\bar{C}$								
$\bar{A}\bar{B}C$								
$\bar{A}BC$								
$\bar{A}B\bar{C}$								
$AB\bar{C}$								
ABC								
$A\bar{B}C$								
$A\bar{B}\bar{C}$								

6 VARIABEL

A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

b. Memasukan data dari truth table ke dalam K-map

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	0	1	0
$\bar{A}B$	1	1	1	1
AB	1	1	0	0
$A\bar{B}$	0	0	0	0

c. Penyederhanaan dilakukan dengan menggabungkan kotak-kotak yang bersebelahan dengan anggota sebanyak 2^m kotak dan formasi kotak membentuk segi empat ($0 \leq m \leq n$).

d. Setiap kelompok dalam K-map akan membentuk satu suku dalam persamaan hasil penyederhanaan, dan jumlah variabel yang terkandung dalam suatu suku tergantung kepada jumlah kotak/daerah dalam suatu kelompok

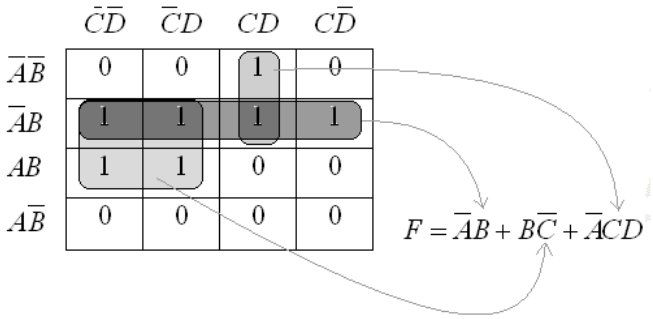
e. Dalam K-map dengan n variabel, suatu kelompok yang memiliki 2^m kotak merupakan suatu suku dengan (n-m) variabel.

f. Jumlah kelompok (group) dalam suatu K-map harus dibuat seminimal mungkin.

g. Jumlah anggota (kotak) dalam suatu kelompok harus dibuat semaksimal mungkin

	$\overline{C}\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\overline{B}$	0	0	1	0
$\overline{A}B$	1	1	1	1
AB	1	1	0	0
$A\overline{B}$	0	0	0	0

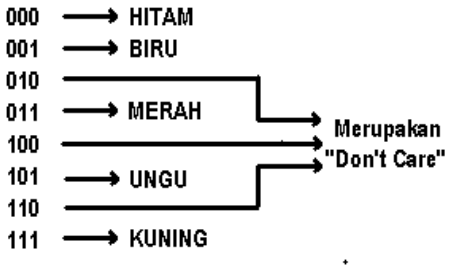
h. Proses pengelompokan dilakukan sampai seluruh kotak yang berlogik 1 tergabung dalam pengelompokan.



Don't care adalah Kombinasi input yang tidak pernah digunakan, tidak dipakai dalam sistem.

Contoh:

Don't care pada K-map 3 variabel (8 kombinasi warna input tetapi hanya 5 warna yang digunakan)



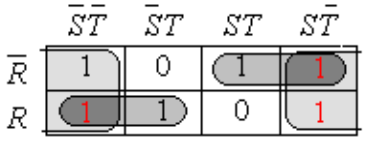
R	S	T	U
0	0	0	1
0	0	1	0
0	1	0	d
0	1	1	1
1	0	0	d
1	0	1	1
1	1	0	d
1	1	1	0

d = don't care

	$\bar{S}\bar{T}$	$\bar{S}T$	ST	$S\bar{T}$
$\bar{R}$	1	0	1	d
R	d	1	0	d

Don't care boleh dibuat logik 1 atau logik 0 tergantung pada posisi yang menguntungkan

Pada M-map diatas nilai d lebih menguntungkan jika berlogik 1



$$U = \bar{T} + R\bar{S} + \bar{R}S$$

$$U = \bar{T} + (R \oplus S)$$

5. Metoda Quine - Mc. Cluskey

Untuk menyederhanakan suatu persamaan logika empat variable, K-map memang metode yang paling efektif. Akan tetapi jika persamaan itu lebih dari empat variable metode ini akan mengalami kesulitan. Metode Quine Mc. Cluskey adalh salah satu cara yang memungkinkan untuk menyederhanakan suatu persamaan logika lebih dari empat variable.

Berikut langkah-langkahnya ;

Bila diberikan persamaan logika $F = \Sigma(0,3,7,8,9,13)$

a. Nyatakan masing-masing unsur minterm kedalam kode biner

0 = 0000

3 = 0011

7 = 0111

8 = 1000

9 = 1001

13 = 1011

b. Tentukan jumlah logik 1 dalam suatu angka biner sebagai indeks dari angka. Kumpulkan semua angka biner yang berindeks sama menjadi satu kelompok pada tabel 1

0 = 0000 → jumlah logik 1 = 0

3 = 0011→ jumlah logic 1 = 2

7 = 0111→ jumlah logic 1 = 3

8 = 1000 → jumlah logic 1 = 1

9 = 1001→ jumlah logic 1 = 2

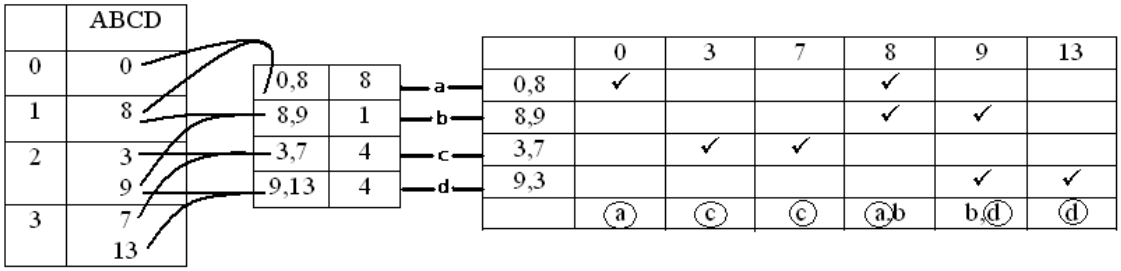
13 = 1011→ jumlah logic 1 = 3

	ABCD
0	0
1	8
2	3 9
3	7 13

c. Bandingkan antara tiap unsur mulai dari indeks terkecil dengan tiap unsur dari indeks sesudahnya. Nilai unsur dari indeks pertama harus lebih kecil dari nilai unsur indeks sesudahnya. Apabila terdapat selisih 2^n maka boleh digabung. Langkah ini akan menghasilkan kelompok baru.

d. Lakukan kembali langkah c sampai tidak ada lagi selisih 2^n.

e. Tiap kelompok diberi nama.



	0	3	7	8	9	13
0,8	✓			✓		
8,9				✓	✓	
3,7		✓	✓			
9,3					✓	✓
	ⓐ	ⓒ	ⓒ	ⓐ,b	b,ⓓ	ⓓ

f. Untuk penyelesaian, kita ambil satu nama yang mewakili tiap angka (a, b, c atau d). Pengambilan nama harus seminimal mungkin.

Sehingga akan didapat

$F = a + c + d$

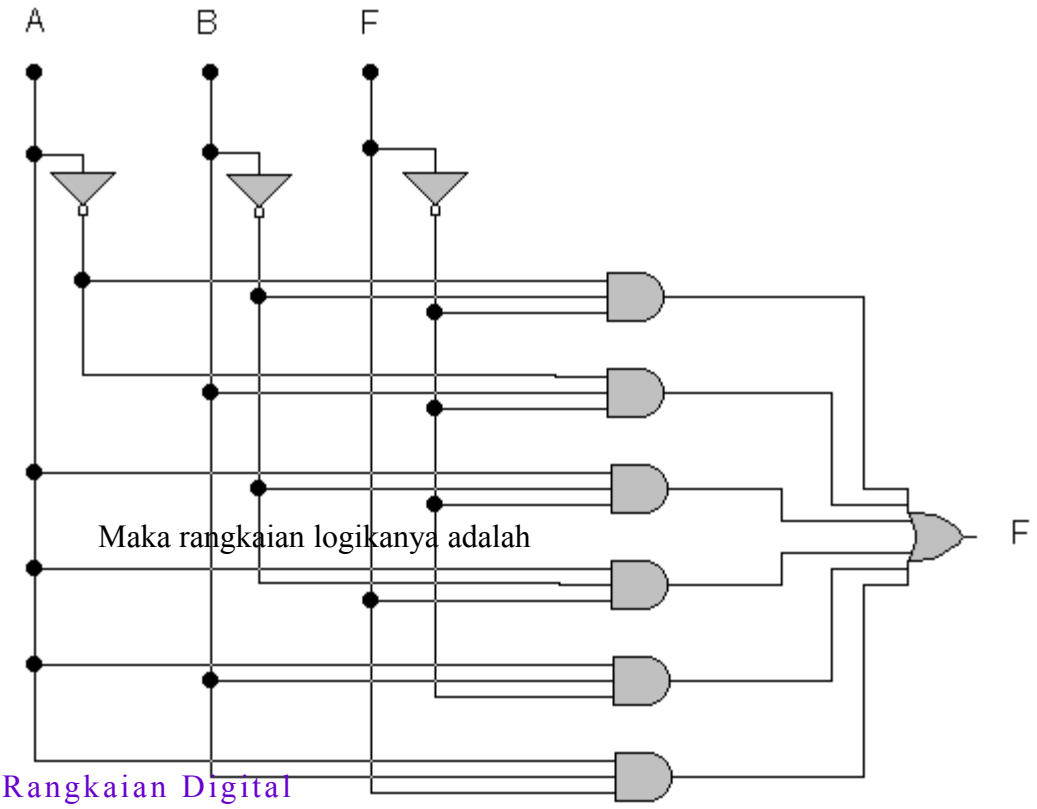
$= 0000 + 0011 + 1001$

$= \underline{1000 + 0111 + 1011}$

$= -000 + 0\text{-}11 + 10\text{-}1$

$= \overline{B}\overline{C}\overline{D} + \overline{A}CD + A\overline{B}D$

Sebagai contoh sederhanakan persamaan logika pada table kebenaran dibawah ini.



Maka rangkaian logikanya adalah

Persamaan diatas dapat disederhanakan dengan beberapa metode yang telah dijelaskan diatas.

- Dengan aljabar Boolean

$$
\begin{aligned}
F &= \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + A\bar{B}\bar{C} + A\bar{B}C + AB\bar{C} + ABC \\
&= \bar{A}\bar{C}(\bar{B} + B) + A(\bar{B}\bar{C} + \bar{B}C + B\bar{C} + BC) \\
&= \bar{A}\bar{C}(1) + A(1) \\
&= \bar{A}\bar{C} + A \\
&= A + \bar{C}
\end{aligned}
$$

- Dengan K-map

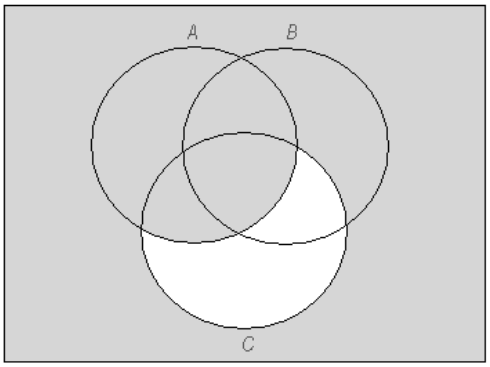
A	B	C	F
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	1	0	0	1
A	1	1	1	1

$$F = A + \overline{C}$$

- Dengan diagram venn

$$F = \overline{A}\overline{B}\overline{C} + \overline{A}B\overline{C} + A\overline{B}\overline{C} + A\overline{B}C + AB\overline{C} + ABC$$



Dari gambar disamping kita bisa lihat lingkaran *A* terisi oleh arsiran sedangkan lingkaran *C* tidak terisi oleh arsiran hanya sebagian yang terisi dan itupun sudah terwakili oleh lingkaran *A*. jadi $F = A + \overline{C}$

- Dengan Quine Mc-Cluskey

$$F = \overline{A}\overline{B}\overline{C} + \overline{A}B\overline{C} + A\overline{B}\overline{C} + A\overline{B}C + AB\overline{C} + ABC$$

000 010 100 101 110 111 → Biner
0 2 4 5 6 7 → Desimal

Jumlah logik 1	*ABCD*
0	0
1	2
	4
2	5
	6
3	7

0,2	2
0,4	4
2,6	4
4,5	1
4,6	2
5,7	2
6,7	1

0,2,4,6	4
0,4,2,6	2
4,5,6,7	2
4,6,5,7	1

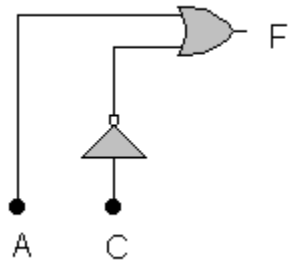
		0	2	4	5	6	7
0,2,4,6	a	✓	✓	✓		✓	
4,5,6,7	b			✓	✓	✓	✓
		a	a	a b	b	a b	b

$F = a + b \rightarrow (0,2,4,6) + (4,5,6,7)$

ABC	ABC
000	100
010	101
100	110
110	111
$0 = \overline{C}$	$1 = A$

$F = A + \overline{C}$

Jadi penyederhanaan persaaan logika diatas dapat diimplementasikan dalam rangkaian sebagai berikut ;



$F = A + \overline{C}$

B. Teknik Implementasi

Implementasi merupakan suatu teknik untuk merealisasikan suatu persamaan logika ke dalam bentuk rangkaian logika. Teknik implementasi sangat penting peranannya dalam perencanaan system-sistem diital.

Salah satu tujuan yang hendak dicapai dalam teknik implementasi ini adalah meralisasikan suatu persamaan logika dengan menggunakan jenis-jenis komponen yang banyak terdapat di pasaran serta dengan memperhatikan segi ekonomis dan kecepatan respon rangkaian.

Gerbang-gerbang Nand dan Nor mempunyai kelebihan dibandingkan dengan gerbang logika lainnya karena dengan menggunakan gerbang logika Nand dan Nor dapat diperoleh fungsi-fungsi And, Or, Ex-Or, Ex-Nor maupun Not gate.

Penulisan persamaan logika bias dilakukan dengan 2 metoda yaitu metoda SOP (Sum Of Product) yang mengacu pada logic 1 pada output dan metoda POS (Product Of Sum) yang mengacu pada logic 0 pada output.

1. Representasi Numerik dari persamaan SOP

Penulisan persamaan logika dalam bentuk SOP untuk persamaan yang memilki jumlah suku dan variable yang banyak biasanya relative panjang. Caranya adalah dengan melakukan representasi numerik.

Contoh:

F = A'B'C + A'BC + AB'C + ABC

Dapat disingkat menjadi:

f(A,B,C)= $\sum$ (1,3,5,7)

Dimana angka decimal 1,3,5,7 merupakan nilai biner dari suku A'B'C, A'BC, AB'C, dan ABC. Dalam suatu persamaan Sop, setiap suku yang mempunyai jumlah variable lengkap (diwakili oleh seluruh variable yang digunakan disebut minterm (disingkat m)).

Untuk membedakan suatu minterm dari minterm yang lain, masing-masing minterm diberikan symbol tersendiri, yaitu dengan menggunakan huruf kecil m dengan subskrip sesuai dengan nilai desimalnya. Misalnya minterm A'B'C diberi symbol m_0; minterm A'BC diberi symbol m_1, dll.

2. Representasi Numerik dari persamaan POS

Penulisan persamaan logika output dalam bentuk product of sum juga dapat disederhanakan menggunakan cara representasi numeric. Caranya dengan mencari ekuivalen biner dari masing-masing sukunyakemudian merubah nilai biner dari masing-masing sukunya, kemudian merubah nilai biner tersebut ke dalam bilangan decimal.

Contoh:

$F = (A+B+C).(A+B'+C).(A'+B+C).(A'+B'+C)$

Disingkat menjadi:

$f(A,B,C) = \pi(0,2,4,6)$

Dimana angka decimal 0 menggantikan suku (A+B+C) yang mempunyai nilai biner 000; angka decimal 2 menggantikan suku (A+B'+C) yang mempunyai nilai biner 010; angka decimal 4 menggantikan suku (A'+B'+C) yang mempunyai biner 100; angka decimal 6 menggantikan suku (A'+B'+C) yang mempunyai biner 110.

Pada suatu persamaan POS, setiap suku yang mempunyai jumlah variable lengkap (terwakili oleh variable yang digunkan) disebut maxterm (disingkat M). Untuk membedakan suatu maxterm dari maxterm yang lain, masing-masing maxterm diberikan symbol tersendiri, yaitu dengan menggunakan huruf besar M dengan subskrip sesuai dengan nilai desimalnya. Misalnya maxterm (A+B+C) diberi symbol M_0; maxterm (A+B'+C) diberi symbol M_1, dll.

3. Merubah persamaan SOP ke POS dan sebaliknya

Representasi numeric juga dapat digunakan untuk memudahkan dalam merubah suatu persamaan logika dari bentuk Sum Of Product (SOP) menjadi Product Of Sum (POS).

Contoh:

$f(A,B,C) = A'B'C + A'BC' + AB'C + ABC$

Dalam representasi numeric, ditulis:

$f(A,B,C) = m_1 + m_2 + m_3 + m_4$

Atau $f(A,B,C) = \sum (1,2,5,7)$

Dalam bentuk POS, dapat ditulis:

$f(A,B,C) = \pi\ (0,3,4,6)$

Atau $f(A,B,C) = M_0.M_3.M_4.M_6$

Atau $f(A,B,C) = (A+B+C).(A+B'+C').(A'+B+C).(A'+B'+C)$

Pada contoh diatas, persamaan SOP terdiri dari 3 variabel input yaitu A,B,C, dengan demikian akan terdapat 23 = 8 kombinasi input (dalam angka decimal : 0,1,2,3,4,5,6,7). Dengan kata lain terdapat sebanyak 8 minterm. Dalam persamaan SOP di atas hanya terdiri dari 4 buah minterm (m_1; m_2 ; m_5 dan m_7). Perhatikan bahwa nagka-angka subskrip yang digunakan adalah 1; 2; 5 dan 7, sisanya yaitu angka-angka 0;3;4 dan 6 akan menjadi subskrip untuk maxterm persamaan dalam bentuk POS.

Jadi,

$f(A,B,C) = \sum (1,2,5,7) = \pi(0,3,4,6)$

atau,

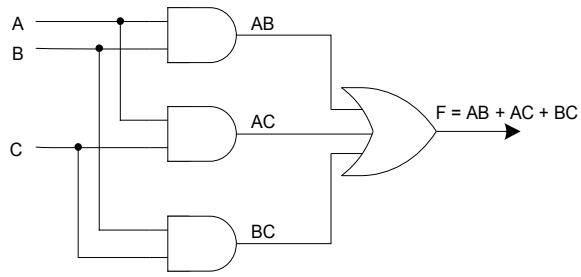
$f(A,B,C) = m_1 + m_2 + m_5 + m_7 = M_0.M_3.M_4.M_6$

4. Implementasi persamaan SOP dengan gerbang Nand

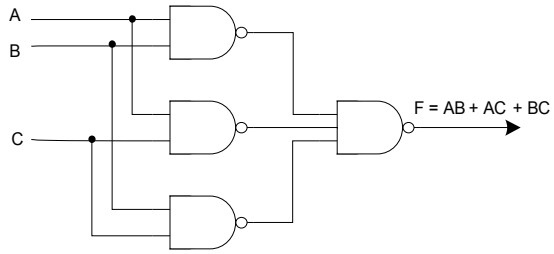
Suatu persamaan dalam bentuk SOP dapat diimplementasikan atau direalisasikan hanya dengan menggunakan gerbang-gerbang NAND. Misalnya, untuk persamaan SOP berikut:

$F = AB + AC + BC$

Implementasi rangkaiannya adalah:



Rangkaian diatas dapat diganti hanya dengan menggunaan gerbang NAND sbb:



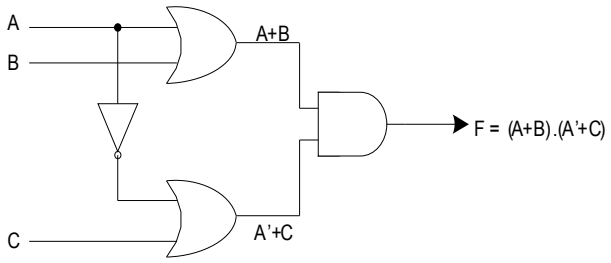
5. Implementasi persamaan POS dengan gerbang Nor

Setiap persamaan logika output yang berada dalam bentuk POS dapat langsung diimplementasikan dengan menggunakan gerbang-gerbang NOR.

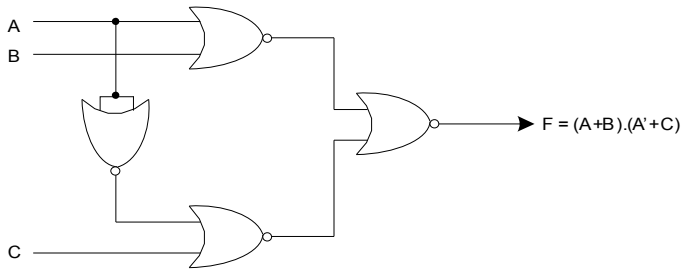
Sebagai contoh, dibawah ini deberikan suatu persamaan logika dalam bentuk POS:

F = (A+B).(A'+C)

Persamaan diatas dapat diimplementasikan dengan menggunakan beberapa jenis gate sbb:



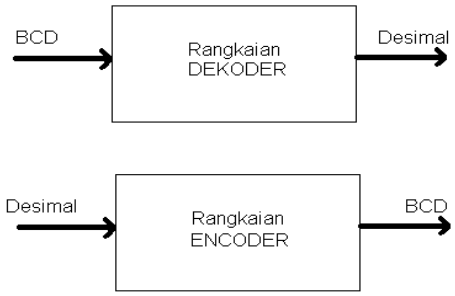
Akan tetapi, persamaan di atas dapat pula diimplementasikan hanya dengan menmggunakan gerbang-gerbang NOR sbb:



BAB IV
ENCODER DAN DECODER

Encoder adalah suatu rangkaian logika yang berfungsi untuk mengkonversikan kode yang lebih dikenal oleh manusia ke dalam kode yang kurang dikenal manusia. Decoder adalah suatu rangkaian logika yang berfungsi untuk mengkonversikan kode yang kurang dikenal manusia kedalam kode yang lebih dikenal manusia.

Contoh

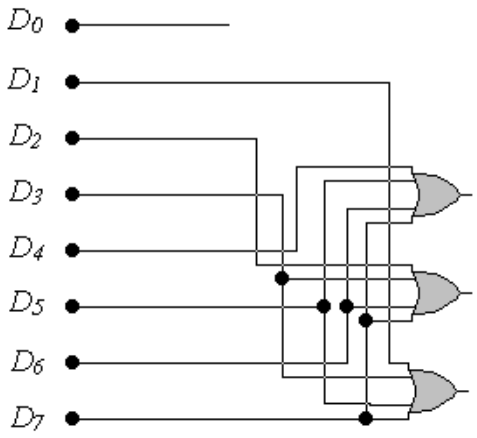


A. Encoder Oktal ke Biner

ENCODER oktal ke biner ini terdiri dari delapan input, satu untuk masing-masing dari delapan angka itu, dan tiga output yang menghasilkan bilangan binernya yang sesuai. Rangkaian itu terdiri dari gerbang OR. Berikut tabel kebenarannya.

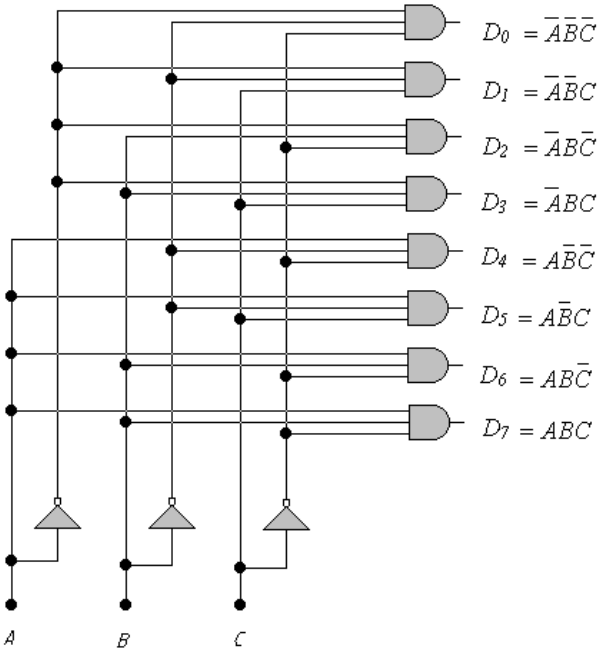
Input								Output		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	A	B	C
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

Diandaikan hanya ada satu saluran input dengan logik 1 untuk setiap kalinya, seelain dari itu input tersebut tidak mempunyai arti. Tampak bahwa rangkaian itu mempunyai delapan input yang dapat memberikan 2^8 kemungkinan kombinasi, tetapi hanya delapan kombinasi yang mempunyai arti.



B. Decoder Biner ke Octal

Pada decoder dari biner ke oktal ini terdapat tiga input yaitu A, B dan C yang mewakili suatu bilangan biner tiga bit dan delapan output yang yaitu D_0 sampai dengan D_7 yang mewakili angka oktal dari 0 sampai dengan

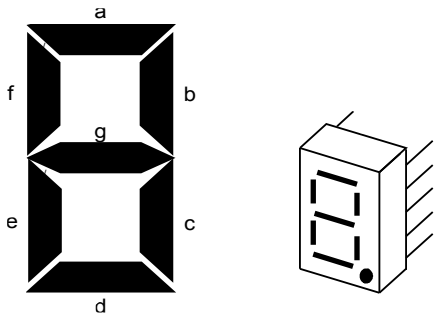


Dalam hal ini unsur informasinya adalah delapan angka oktal. Sandi untuk informasi diskrit ini terdiri dari bilangan biner yang diwakili oleh tiga bit. Kerja dekorder ini dapat lebih jelas tampak dari hubungan input dan output yang ditunjukan pada tabel kebenaran dibawah ini. Tampak bahwa variabel outputnya itu hanya dapat mempunyai sebuah logk 1 ntuk setiap kombinasi inputnya. Saluran output yang nilainya sama dengan 1 mewakili angka oktal yang setara dengan bilangan biner pada saluran inputnya

Input			Output							
A	*B*	*C*	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

C. Peraga 7 segmen

Untuk menampilkan bilangan yang dikeluarkan oleh decoder akan dapat dipakai sebuah penampil 7-segmen *(seven segment display)*. Penampil ini terdiri dari 7-segmen yang tersusun membentuk angka-angka, ditunjukkan pada Gb.C1.



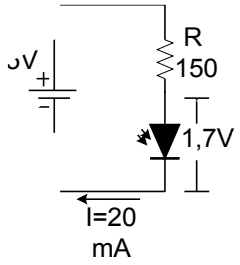
Gb.C1

Cara mengidentifikasi segmen-segmen dalam penampil 7-segmen

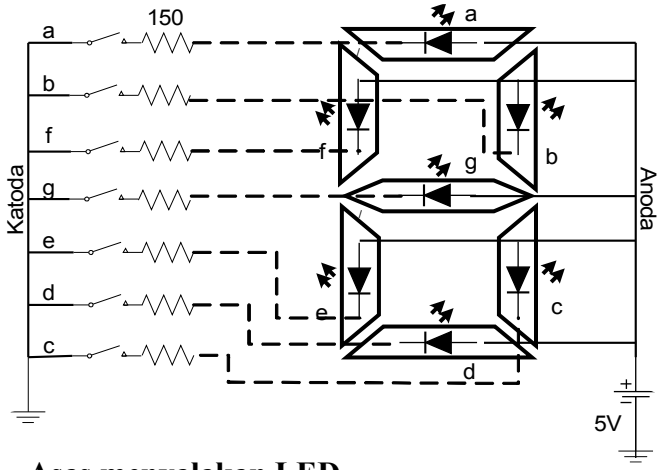
Segmen-segmen ditandai dengan huruf-huruf a, b, c, d, e, f dan g. setiap segmen dapat diisi sebuah filamen yang akan berpijar apabila diaktifkan. Jenis penampil semacam ini disebut penampil pijar *(incandescent display)*. Cara memijarkan tidak beda dengan lampu-lampu pijar biasa.

Jenis penampil lain adalah yang segmen-segmennya mengandung tabung gas *(gas discharge tube)*, yang beroperasi dengan tegangan tinggi. Penampil ini berpendar dengan warna jingga. Ada pula penampil pendaran *(fluorescent tube)* yang mengeluarkan cahaya kehijauan, dan beroperasi dengan tegangan rendah.

Penampil yang banyak dipakai adalah yang menerapkan LED *(Light Emitting Diode)*. Untuk menyalakan LED diterapkanlah sirkit seperti pada Gb.C2. R=150Ω berfungsi untuk membatasi arus agar bertahan pada 20mA. Tanpa R, LED akan terbakar. Pada LED akan terdapat tegangan kira-kira 1,7V.



Gb.C2. Sirkit untuk menyalakan LED



Gb.C3

Asas menyalakan LED.

LED yang dibumikan (lewat R=150 Ω) akan menyala

Setiap segmen didalam penampil pada Gb.C1 berisi satu LED. Adapun asasnya hubungan LED ditunjukkan dalam Gb.C3, yaitu anoda-anoda disatukan dan diberi potensial +Vcc (5V). katodalah yang diberi logik 0 atau 1 dari dekoder lewat R=150Ω. Apabila saklar ditutup, maka katoda yang bersangkutan memperoleh logik 0 dan LED itupun menyala, sebab sirkit baterai tertutup. Pada Gb.C4 ditunjukkan angka-angka yang akan dapat ditampilkan oleh tujuh segmen.

Gb.C4

Angka-angka yang akan dapat ditampilkan oleh 7-segmen

Sebagai contoh, untuk menyalakan atau menampilkan angka 6, maka saklar a, c, d, e, f, dan g harus ditutup, sehingga segmen-segmen a, c, d, e, f, dan g pun menyala. Dalam pelaksanaan praktek, segmen-segmen a hingga g dikoneksikan langsung pada keluaran a hingga g pada dekoder. Keluaran yang aktif akan meng-*ground*-kan segmen yang berkoneksi padanya, sehingga segmen tersebut menyala. Contoh, keluaran pada dekoder (a, b, c) aktif, maka output-output itu masing-masing meng-*ground*-kan katodanya LED yang ada di segmen a, b, dan c, sehingga tampilah 7.

C. Decoder BCD ke Desimal

Rangkaian Dekoder BCD ke desimal ditunjukan pada gambar D2. Unsur informasi dalam hal ini adalah sepuluh angka desimal yang diwakili oleh sandi BCD. Masing-masing keluarannya sama dengan 1 hanya bila variabel masukannya membentuk suatu kondisi bit yang sesuai dengan angka desimal yang diwakili oleh sandi BCD itu. Tabel D2 menunjukkan hubungan masukan dan keluaran dekoder tersebut. Hanya sepuluh kombinasi masukan pertama yang berlaku untuk penentuan sandi itu, enam berikutnya tidak digunakan dan menurut definisi, merupakan keadaan tak acuh. Jelas keadaan tak acuh itu pada perencanaannya digunakan untuk menyederhanakan fungsi

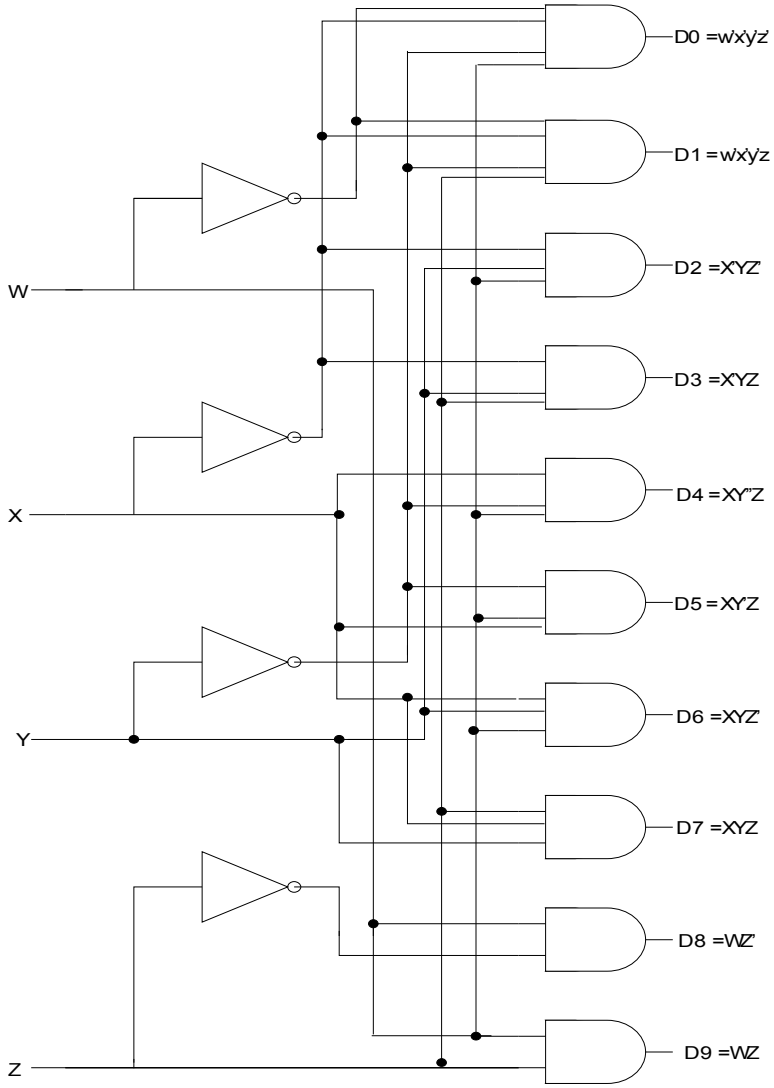
keluarannya, jika tidak setiap gerbang akan memerlukan empat masukan. Untuk kelengkapan analisis tabel D2 memberikan semua keluaran termasuk enam kombinasi yang tidak terpakai dalam sandi BCD itu; tetapi jelas keenam kombinasi tersebut tidak mempunyai arti apa-apa dalam rangkaian itu.

Dekoder dan enkoder itu banyak sekali dipakai dalam sistem digital. Dekoder tersebut berguna untuk memperagakan unsur informasi diskret yang tersimpan dalam register. Misalnya suatu angka desimal yang disandikan dalam BCD dan tersimpan dalam register empat sel dapat diperagakan dengan pertolongan rangkaian dekoder BCD ke desimal dimana keluaran keempat sel biner tersebut diubah sehingga menyalakan 10 lampu penunjuk. Lampu penunjuk itu dapat berupa angka peraga *(display digit)*, sehingga suatu angka desimal akan menyala bila keluaran dekoder yang sesuai adalah logika 1. Rangkaian dekoder juga berguna untuk menentukan isi register dalam proses pengambilan keputusan. Pemakaiannya yang lain adalah untuk membangkitkan sinyal waktu dan sinyal urutan untuk keperluan pengaturan.

Tabel D2

Tabel kebenaran decoder BCD ke desimal

Masukan				Keluaran									
w	x	y	z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9
0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0	0	0	1	0	0	0
0	1	1	1	0	0	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	1	0	0	0	0	0	0	0	0	0	1
1	0	1	0	0	0	1	0	0	0	0	0	1	0
1	0	1	1	0	0	0	1	0	0	0	0	0	1
1	1	0	0	0	0	0	0	1	0	0	0	1	0
1	1	0	1	0	0	0	0	0	1	0	0	0	1
1	1	1	0	0	0	0	0	0	0	1	0	1	0
1	1	1	1	0	0	0	0	0	0	0	1	0	1

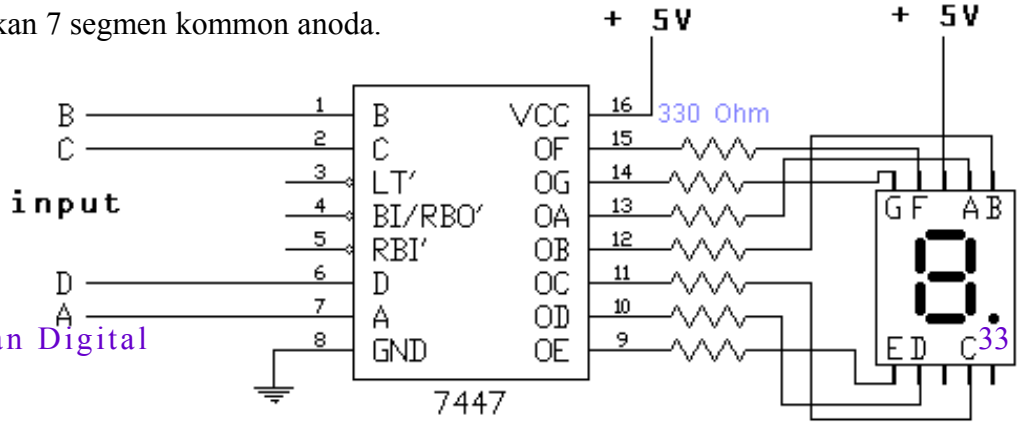


Gb.D2

Dekoder BCD ke decimal

Decoder BCD ini ada 2 macam yaitu yang outputnya aktif level tinggi dan yang outputnya aktif rendah sehingga membutuhkan 7 segmen yang berbeda. Untuk aktif level tinggi menggunakan 7 segmen kommon katoda, sedangkan untuk aktif level rendah menggunakan 7 segmen kommon anoda.

Contoh *rangkaian*



Decoder

BCD to 7

segmen

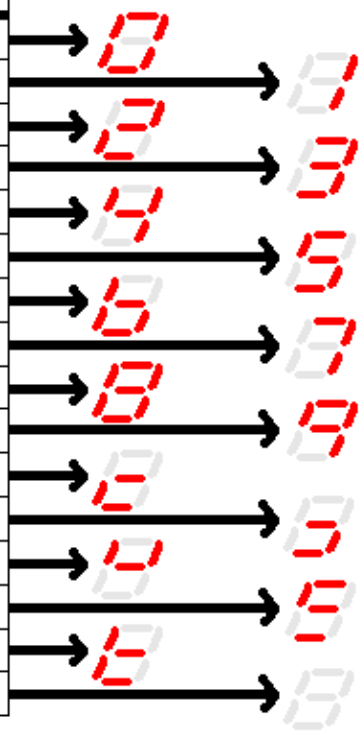
kommon

anoda

Tabel Kebenaran Decoder common anoda

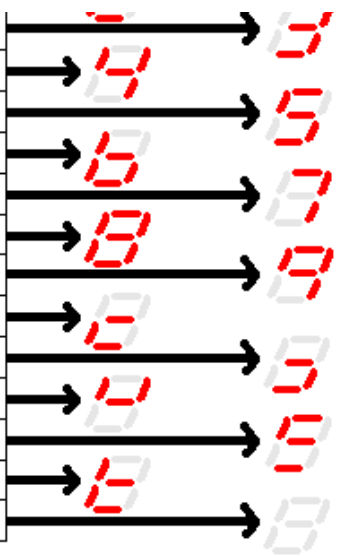
kommom anoda

D	C	B	A	g	f	e	d	c	b	a
0	0	0	0	1	0	0	0	0	0	0
0	0	0	1	1	1	1	1	0	0	1
0	0	1	0	0	1	0	0	1	0	0
0	0	1	1	0	1	1	0	0	0	0
0	1	0	0	0	0	1	1	0	0	1
0	1	0	1	0	0	1	0	0	1	0
0	1	1	0	0	0	0	0	0	1	1
0	1	1	1	1	1	1	1	0	0	0
1	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	1	1	1
1	0	1	1	0	1	1	0	0	1	1
1	1	0	0	0	0	1	1	1	0	1
1	1	0	1	0	0	1	0	1	1	0
1	1	1	0	0	0	0	0	1	1	1
1	1	1	1	1	1	1	1	1	1	1



Tabel Kebenaran Decoder common katoda

0	0	1	1	1	0	0	1	1	1	1
0	1	0	0	1	1	0	0	1	1	0
0	1	0	1	1	1	0	1	1	0	1
0	1	1	0	1	1	1	1	1	0	0
0	1	1	1	0	0	0	0	1	1	1
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	0	0	1	1	1
1	0	1	0	1	0	1	1	0	0	0
1	0	1	1	1	0	0	1	1	0	0
1	1	0	0	1	1	0	0	0	1	0
1	1	0	1	1	1	0	1	0	0	1
1	1	1	0	1	1	1	1	0	0	0
1	1	1	1	0	0	0	0	0	0	0

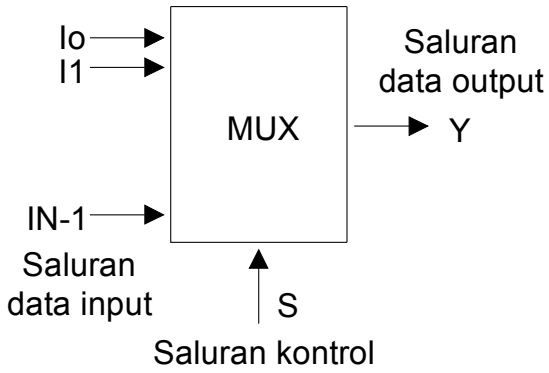


Dengan demikian untuk peraga 7 segmen jenis common cathode memerlukan decoder dengan output jenis active high untuk menyalakan setiap segmennya, sedangkan untuk peraga 7 segmen jenis common anode memerlukan decoder dengan output jenis active low.

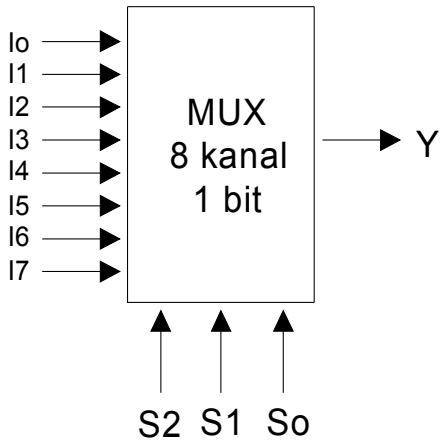
BAB V
MULTIPLEXER DAN DEMULTIPLEXER

Multiplexer dapat didefinisikan sebagai suatu rangkaian logika yang dapat menerima beberapa saluran data input yang terdiri dari 1 bit/lebih secara paralel dan pada outputnya hanya dilewatkan salah satu saluran data yang terpilih. Saluran data input yang terpilih dikontrol oleh beberapa saluran control yang sering disebut sebagai saluran pemilih (input select). Jumlah saluran control berkaitan erat dengan jumlah saluran data

input yang akan dikontrol. Multiplekser sering juga disebut dengan selector data. Diagram sebuah multiplekser secara umum :



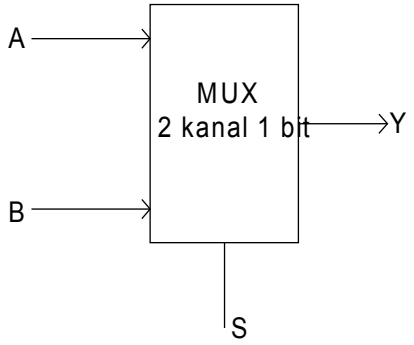
contoh multiplekser 8 kanal 1 bit :



Contoh soal :

1. Rancanglah sebuah MUX 2 kanal 1 bit.

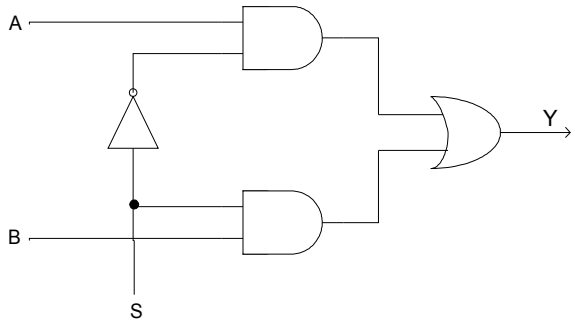
Jawab :



Tabel kebenaran MUX 2 kanal 1 bit

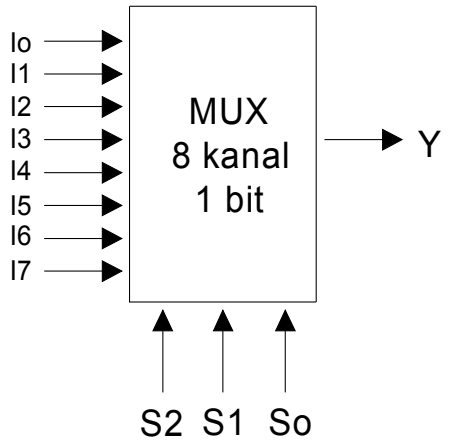
Selector (S)	Output (Y)
0	S'A
1	SB

Rangkaian dalam MUX 2 kanal 1 bit



2. Rancanglah MUX 8 kanal 1 bit.

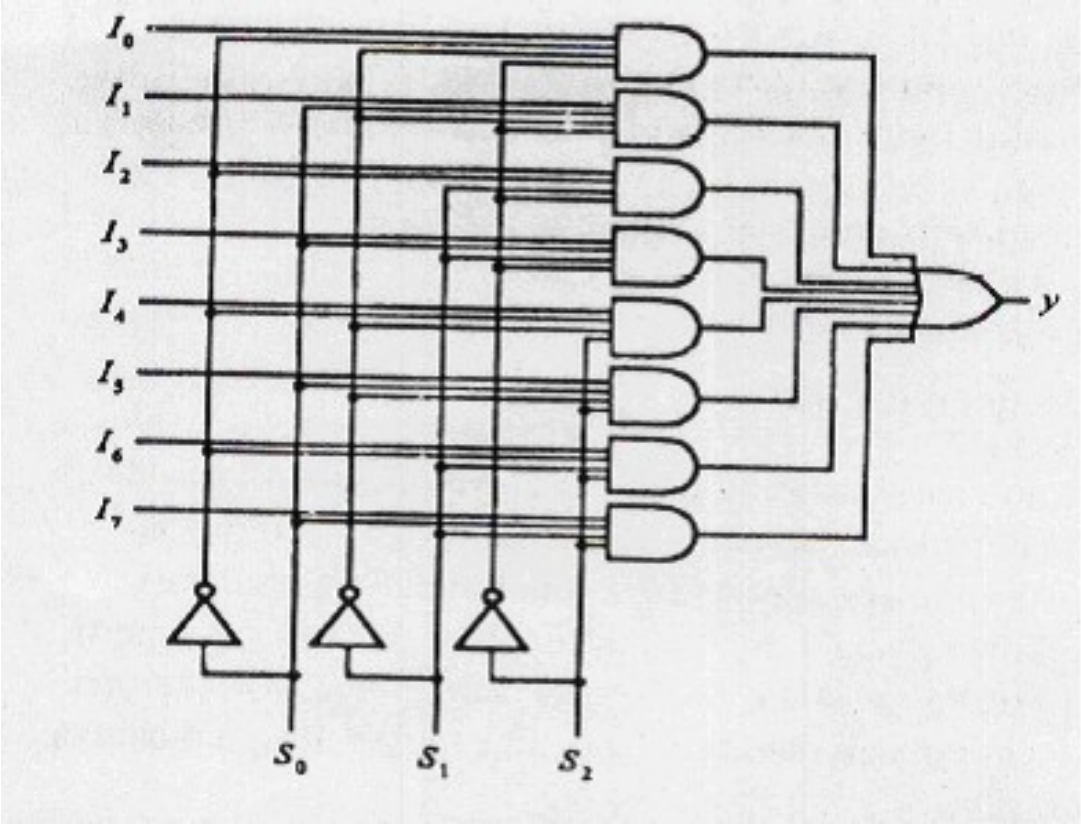
Jawab :



Tabel kebenaran MUX 8 kanal 1 bit

S_2	S_1	S_0	Output (Y)
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

Rangkaian dalam MUX 8 kanal 1 bit



BAB VI

PENJUMLAHAN DAN PENGURANGAN

A. Penjumlahan

1. Half Adder

Tabel penambahan pada gambar 1*(a)* dapat kita anggap sebagai tabel kebenaran. Angka yang ditambahkan ada pada posisi masukan tabel. Pada gambar 3*(a)*, masukan ini merupakan kolom masukan *A* dan *B*. Tabel kebenaran membutuhkan dua kolom keluaran, satu kolom untuk jumlah dan satu kolom untuk pindahan.

MASUKAN		KELUARAN	
B	*A*	Σ	C_o
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1
Penambahan digit biner		Jumlah (XOR)	Disimpan (AND)

(a)

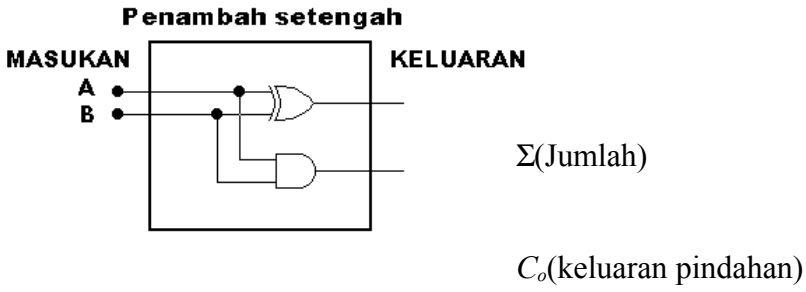
(b)

Gambar 3 Penambahan setengah. *(a)* Tabel kebenaran. *(b)* simbol blok.

Kolom jumlah diberi label dengan simbol Σ. Kolom pindahan diberi label dengan C_o. C_o singkatan untuk keluaran pindahan atau *carry out*. Simbol blok yang cocok untuk penambahan yang memberikan fungsi tabel kebenaran tersebut diperlihatkan pada gambar 3*(b)*. Rangkaian ini disebut rangkaian *penambah setengah*. Rangkaian-penambah-setengah mempunyai masukan (*A*,*B*) dan dua keluaran (Σ,C_o).

Lihat dengan teliti tabel kebenaran penambah-setengah pada gambar 3*(a)*. Bagaimana bentuk boolean yang diperlukan untuk keluaran C_o? Bentuk boolean itu ialah $A \cdot B = C_o$ kita membutuhkan dua gerbang AND dua masukan untuk membuat keluaran C_o.

Sekarang bagaimana bentuk boolean untuk jumlah keluaran (Σ) dari setengah penambahan pada gambar 3*(a)*? Bentuk boolean tersebut ialah $\overline{A} \cdot B + A \cdot B = \Sigma$. Kita dapat menggunakan dua gerbang AND dan satu gerbang OR untuk melakukan pekerjaan ini. Bila dilihat lebih dekat, anda akan mendapatkan bahwa pola ini juga merupakan gerbang XOR. Kemudian bentuk boolean yang disederhanakan menjadi $A \oplus B = \Sigma$. Dengan kata lain kita hanya memerlukan satu gerbang XOR 1-masukan untuk menghasilkan keluaran jumlah tersebut.



Gambar 4 Diagram logika untuk penambah setengah

Dengan menggunakan gerbang AND dua masukan, suatu diagram simbol logika untuk penambahan setengah kita nyatakan pada gambar 4. rangkaian penambah_setengah hanya menambahkan kolom LSB (kolom 1) pada persoalan penambah biner. Untuk bagian 2-an, 4-an, 8-an, 16-an dan sebagainya, dalam penambahan biner, harus kita gunakan rangkaian yang disebut *penambah lengkap*.

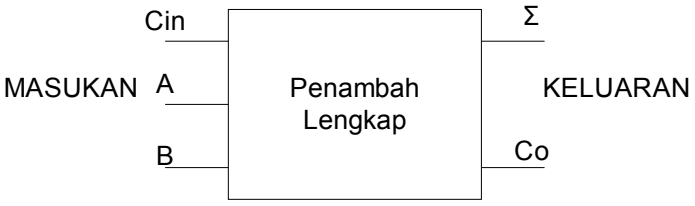
2. Full Adder

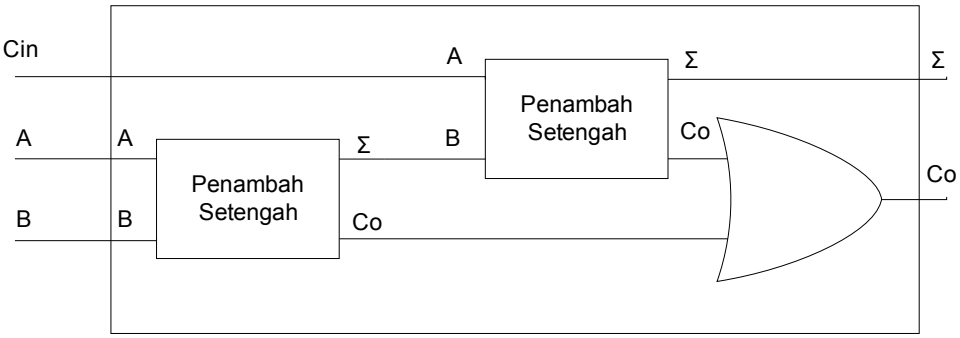
Gambar 5 merupakan bentuk singkat dari tabel penambahan biner, dengan situasi **1** + **1** + **1**. tabel kebenaran pada gambar 5*(a)* memperlihatkan semua kombinasi yang mungkin dari *A, B,* dan C_{in} (masukan pindahan). Tabel kebenaran ini untuk panambah lengkap. Penambah lengkap digunakan untuk semua harga bagian biner, kecuali bagian

1-an. Bila diinginkan suatu masukan pindahan tambahan maka kita harus gunakan penambah lengkap. Diagram blok dari penambah lengkap diperlihatkan pada gambar 5(b). Penambah lengkap mempunyai 3 masukan ; C_{in}, A, dan B. Untuk mendapatkan keluaran Σ dan C_o tiga masukan tersebut harus kita tambahkan.

Gambar 5(a).Tabel Kebenaran Full Adder

MASUKAN			KELUARAN	
C	B	A	Σ	Co
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1
Pindahan + B +A			Ju mlah	Carry Out



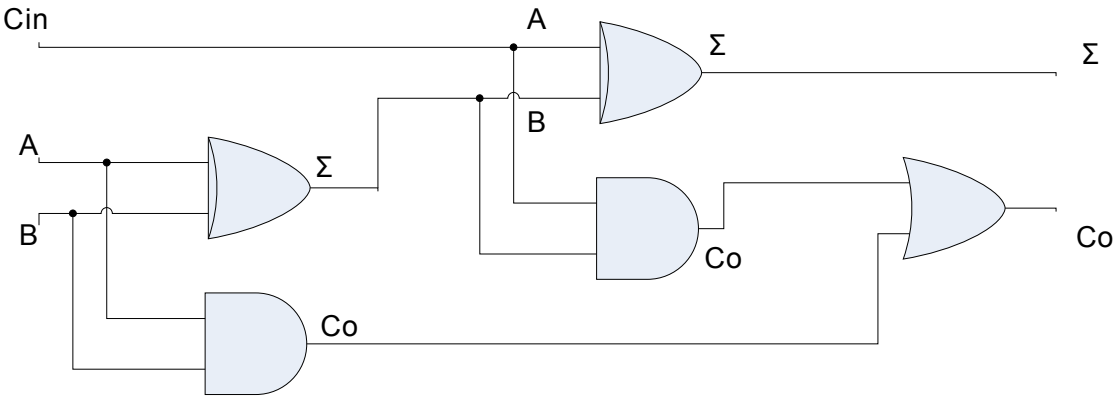


Gambar 5(b). Diagram blok dari penambah lengkap

Salah satu metode yang paling mudah untuk membentuk logika gabungan penambahan lengkap digambarkan pada gambar 5*(c).* Digunakan rangkaian setengah-penambah dan gerbang OR. Bentuk rangkaian ini adalah $A \oplus B \oplus C = \Sigma$. Ekspresi untuk keluaran pindahan adalah $A \cdot B + C_{in} \cdot (A \oplus B) = C_o$. Rangkaian logika pada gambar 6*(a)* merupakan suatu penambah lengkap, rangkaian ini berdasarkan diagram blok yang menggunakan dua buah setengah-penambah seperti gambar 5*(c).* Dibawah diagram logika ini, terdapat rangkaian logika yang lebih mudah dirangkai. Gambar 5(b) berisi dua gerbang XOR dan tiga gerbang NAND, yang memungkinkan rangkaian sangat mudah dirangkai.

Penambah setengah dan penambah lengkap kita gunakan bersama-sama. Untuk persoalan pada gambar 2*(a),* kita membutuhkan penambah setengah dan penambah lengkap yang merupakan rangkaian sederhana. Dengan demikian, banyak diantara rangkaian ini dibutuhkan untuk menambah persoalan yang lebih panjang.

Rangkaian penambah setengah dan penambah lengkap banyak digunakan sebagai bagian unit logika aritmatik (ALU, arithmatic logic unit) dari suatu mikroprosesor. ALU dari mikroprosesor dapat juga mengurangi penggunaan rangkaian penambah setengah dan penambah lengkap yang sama. Pada akhir bab ini, kita akan menggunakan penambah untuk membentuk pengurangan biner.



3. Parallel Adder

Penjumlahan penuh yang telah diperkenalkan dalam pasal 6.2 membentuk jumlah dua bit dan bawaan sebelumnya. Dua bilangan biner n bit masimg-masing dapat dijumlahkan dengan rangkaian tersebut. Untuk membuktikannya dengan contoh khas, tinjau dua bilangan biner, A = 1011 dan B = 0011, yang jumlahnya adalah S = 1110. bila suatu pasangan bit dijumlahkan dengan suatu penjumlahan penuh, rangkaian itu menghasilkan bawaan yang akan digunakan dengan pasangan bit pada kedudukan yang lebih berarti yang lebih tinggi. Hal itu ditunjukkan dalam tabel 8.1

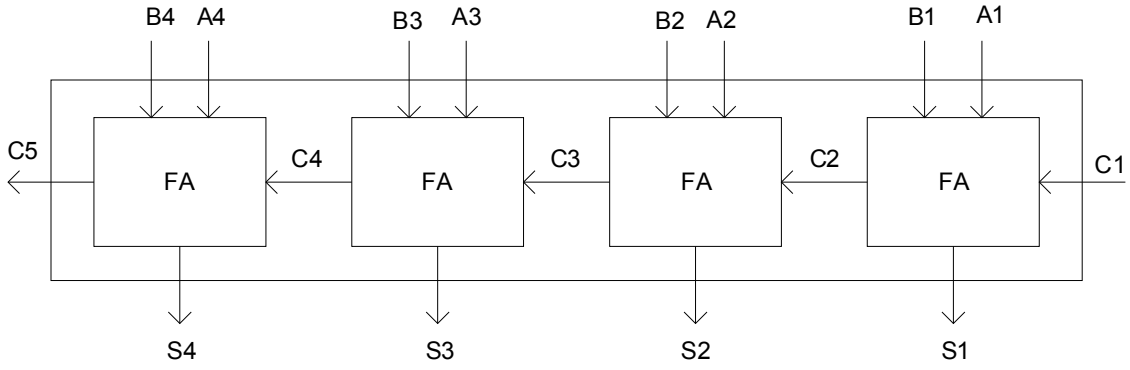
Tabel 8.1 Penjumlah biner parallel

MASUKAN			KELUARAN	
C	B	A	Σ	Co
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1
Pindahan + B +A			Jumlah	Carry Out

Dalam tabel 8.1 itu, bit dijumlahkan oleh penjumlah penuh, dengan dimulai dari kedudukan berarti terendah (subskrip 1), untuk membentuk bit jumlah dan bit bawaan. Masukan dan keluaran rangkaian penjumlahan penuh pada gambar 6.7 juga ditunjukkan dalam tabel 8.1. Bawaanmasukan C1 pada kedudukan berarti terendah harus 0. Nilai Ci+1 dalam suatu kedudukan berarti tertentu adalah bawaan keluaran penjumlahan penuh itu. Nilai tersebut dipindahkan ke bawaan masukan penjumlahan penuh yang menjumlah bit itu satu kedudukan berarti lebih tinggi ke kiri. Bit jumlah itu dibangkitkan berawal dari kedudukan terkanan dan tersedia segera setelah bit bawaan sebelumnya didapatkan.

Jumlah dua bilangan biner n-bit, A dan B, dapat diperoleh dalam dua cara: secara seri atau parallel. Cara seri hanya menggunakan satu rangkaian penjumlahan penuh dan suatu peralatan penyimpan untuk menahan bawaan keluaran yang dihasilkan. Pasangan bit dalam A dan B dipindahkan secara seri, satu demi satu, melalui penjumlahan penuh tunggal untuk menghasilkan sederetan bit keluaran sebagai jumlahnya. Bawaan keluaran yang tersimpan dari suatu pasangan bit itu digunakan sebagai bawaan masukan untuk pasangan bit berikutnya. Cara seri ini akan ditinjau lebih lanjut dalam Bab Sembilan. Cara parallel menggunakan n rangkaian penjumlahn penuh, dan semua bit pada A dan B dikenakan secara serentak. Bawaan keluaran dari suatu penjumlah penuh dihubungkan ke bawaan masukan penjumlah penuh satu kedudukan di kirinya. Segera setelah bawaan itu dihasilkan, bit jumlah yang benar muncul dari keluaran jumlah semua penjumlah penuh itu.

Suatu penjumlah paralel biner adalah suatu fungsi digital yang menghasilkan jumlah aritmatika dua bilangan biner secara paralel. Fungsi itu terdiri dari sejumlah penjumlahan penuh yang dihubungkan secara bertigkat, dengan bawaan keluaran dari suatu penjumlah penuh yang dihubungkan ke bawaan masukan penjumlahan penuh berikutnya.



Gambar 8.2 Penjumlah paralel 4 bit

Gambar 8.2 menunjukkan hubungan empat rangkaian penjumlah penuh (full adder) untuk memberikan suatu penjumlah paralel 4 bit. Bit yang ditambah A dan bit penambah B ditunjukkan oleh bilangan subskrip dari kanan ke kiri, dengan subskrip I

yang menyatakan bit tingkat rendah. Bawaan itu dihubungkan secara berantai sepanjang penjumlahan penuh tersebut. Bawaan masukan ke penjumlah itu adalah C1 dan bawaan keluarannya adalah C5. Keluaran S menghasilkan bit jumlah yang diperlukan. Bila rangkaian penjumlah penuh 4 bit itu dikemas dalam suatu kemasan IC, kemasan itu mempunyai empat kutub untuk bit ditambah, empat kutub untuk bit penambah, empat kutub untuk it jumlah, dan dua kutub untuk bawaan masukan dan keluaran. Contoh penjumlahan penuh 4 bit semacam itu adalah TTL jenis IC 74283.

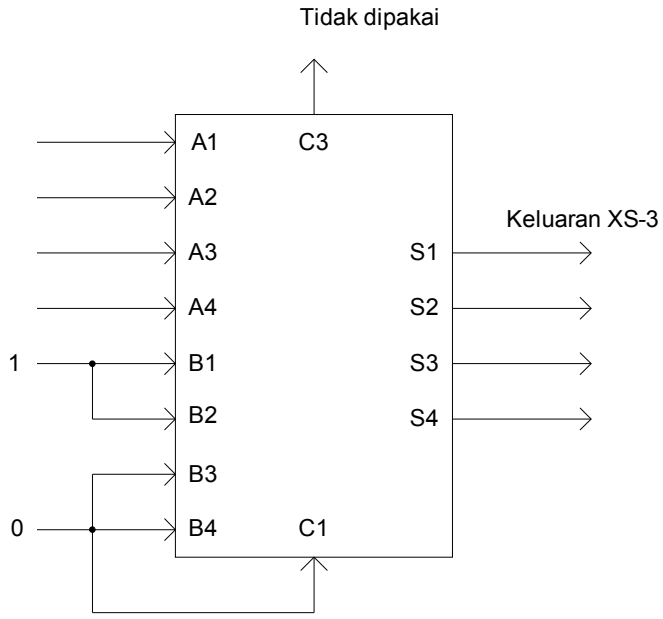
Suatu penjumlahan biner n-bit memerlukan n penjumlah penuh. Rangkaian itu dapat dibuat dari IC penjumlah penuh 4 bit, 2 bit, dan 1 bit dengan menghubungkan beberapa kemasan secara bertingkat. Bawaan keluaran dari suatu kemasan harus dihubungkan ke bawaan masukan yang lain dengan bit tingkat tinggi berikutnya.

Penjumlahan penuh 4 bit adalah suatu contoh khas fungsi MSI. Penjumlahan itu dapat digunakan dalam berbagai pemakaian yang meliputi operasi aritmatika. Dapat dibuktikan bahwa bila rancangan rangkaian itu dilakukan dengan cara klasik memerlukan satu tabel kebenaran dengan 2^9=512 baris, karena terdapat sembilan masukan ke rangkaian tersebut. Dengan cara iterasi pemakaian fungsi yang diketahui secara bertingkat-tingkat, dapat diperoleh suatu implementasi sederhana dan teratur rapi.

Contoh lain penggunaan MSI penjumlah biner 4 bit itu untuk fungsi logika acak diberikan dalam contoh 8.1.

Contoh 8.1 Rancangkan suatu pengubah sandi BCD ke XS-3.

Jawab:



Gambar 8.3 Pengubah sandi BCD ke XS-3

Rangkaian itu telah dirancang dalam pasal 6.4 dengan cara klasik. Rangkaian yang diperoleh dari rancangan itu ditunjukkan dalam gambar 6.10 dan memerlukan 11 gerbang. Bila diimplementasikan dengan gerbang SSI, akan memerlukan 3 kemasan IC dan 15 hubungan kawat dalam (tidak meliputi hubungan masukan dan keluaran). Pengamatan pada tabel kebenaran menunjukkan bahwa sandi setara XS-3 dapat diperoleh dari sandi BCD dengan menambahkan biner 0011. Penambahan tersebut dapat dengan mudah diimplementasikan dengan pertolongan suatu rangkaian MSI penjumlah penuh 4 bit, seperti yang ditunjukkan dalam gambar 8.3. Angka BCD diberikan ke masukan A. Masukan B dibuat tetap sama dengan 0011. Hal itu dilakukan dengan menggunakan logika-1 ke B1 dan B2 dan logika 0 ke B3,B4, dan C1. Logika 1 dan logika 0 itu adalah sinyal fisik yang nilainya tergantung pada keluarga logika IC yang dipakai. Untuk rangkaian TTL, logika 1 setara 3,5 volt, dan logika 0 setara dengan tanah. Keluaran S pada rangkaian itu memberikan sandi XS-3 yang setara dengan angka BCD masukannya.

Implementasi tersebut memerlukan satu kemasan IC dan lima hubungan kawat, tidak termasuk kawat masukan dan keluarannya.

4. BCD Adder

Komputer atau kalkulator yang melaksanakan operasi aritmatika langsung dalam sistem bilangan decimal mewakili bilangan decimal dalam bentuk sandi biner. Suatu penjumlah semacam itu harus menggunakan rangkaian aritmatika yang menerima bilangan desimal yang disandikan dan memberikan hasilnya dalam sandi yang telah disetujui. Untuk penjumlahan biner, untuk setiap kalinya cukup ditinjau sepasang bit yang berarti dan suatu bawaan sebelumnya. Suatu penjumlah desimalmemerlukan sekurang-kurangnya sembilan masukan dan lima keluaran karena empat bit diperlukan untuk menyandikan masing-masing bilangan desimal dan rangkaian itu harus mempunyai sebuah bawaan masukan dan sebuah bawaan keluaran. Tentu saja, terdapat berbagai macam rangkaian penjumlah desimal yang dapat dibuat, tergantung pada sandi yang dipergunakan untuk mewakili angka desimal itu.

Rancangan suatu rangkaian kombinasi sembilan masukan, lima keluaran menurut metoda klasik akan memerlukan suatu tabel kebenaran dengan 2^9=512 isian. Banyak di antara kombinasi masukan itu adalah keadaan tak acuh, karena masing-masing masukan andi biner mempunyai enam kombinasi yang tidak terpakai. Fungsi Boole yang disederhanakan untuk rangkaian itu dapat diperoleh dengan suatu cara tabel yang dihasilkan oleh komputer, dan hasilnya mungkin akan merupakan suatu hubungan antar gerbang dengan pola yang tidak teratur. Suatu prosedur lainnya adalah menjumlah bilangan itu dengan rangkaian penjumlah penuh, dengan memperhitungkan kenyataan bahwa enam kombinasi dalam masing-masing maukan 4 bit itu tidak terpakai. Keluarannya harus disesuaikan sedemikian hingga hanya kombinasi biner yang merupakan kombinasi untuk sandi decimal itu saja yang dihasilkan.

Dalam bagian ini akan ditinjau suatu penjumlahan aritmatika dua angka decimal dalam BCD, bersama-sama dengan suatu bawaan yang mungkin dari suatu tingkat sebelumnya. Karena masing-masing angka masukan itu tidak melebihi 9, jumlah keluarannya tidak dapat lebih dari 9 + 9 +1 = 19, 1 dalam jumlah itu adalah bawaan masukan. Penjumlah itu membentuk jumlah dalam bentuk biner dan menghasilkan suatu

hasil yang dapat berkisar dari 0 sampai dengan 19. Bilangan biner tersebut diberikan dalam tabel 8.2 dan diberi tanda dengan lambing K, Z8, Z4, Z2, dan Z1. K adalah bawaan, dan subskrip di bawah huruf Z mewakili bobot 8, 4, 2, dan 1 yang dapat diberikan ke empat bit dalam sandi BCD. Kolom pertama dalam tabel itu memberikan jumlah biner sebagaimana yang muncul pada keluaran suatu penjumlah biner 4 bit. Jumlah keluaran dua angka desimal harus diwakili dalam BCD dan harus muncul dalam bentuk yang diberikan dalam kolom kedua pada tebel itu. Masalahnya adalah mencari suatu aturan sederhana sehingga bilangan biner dalam kolom pertama dapat diubah menjadi perwakilan bilangan itu dalam angka BCD yang benar pada kolom kedua.

Tabel 8.2 Penurunan penjumlah BCD

Jumlah Biner					Jumlah BCD					Desimal
K	Z8	Z4	Z2	Z1	C	S8	S4	S2	S1	
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	2
0	0	0	1	1	0	0	0	1	1	3
0	0	1	0	0	0	0	1	0	0	4
0	0	1	0	1	0	0	1	0	1	5
0	0	1	1	0	0	0	1	1	0	6
0	0	1	1	1	0	0	1	1	1	7
0	1	0	0	0	0	1	0	0	0	8
0	1	0	0	1	0	1	0	0	1	9
0	1	0	1	0	1	0	0	0	0	10
0	1	0	1	1	1	0	0	0	1	11
0	1	1	0	0	1	0	0	1	0	12
0	1	1	0	1	1	0	0	1	1	13
0	1	1	1	0	1	0	1	0	0	14
0	1	1	1	1	1	0	1	0	1	15
1	0	0	0	0	1	0	1	1	0	16
1	0	0	0	1	1	0	1	1	1	17
1	0	0	1	0	1	1	0	0	0	18
1	0	0	1	1	1	1	0	0	1	19

Dalam memeriksa isitabel itu, tampak bahwa bila jumlah biner itu sama dengan atau kurang dari 1001, bilangan BCD yang bersesuaian identik, dan oleh karenanya tidak diperlukan perubahan. Bila jumlah biner itu lebih besar dari 1001, didapatkan suatu

perwakilan BCD yang tidak sah. Penambahan biner 6 (0110) ke jumlah biner itu mengubahnya menjadi perwakilan BCD yang benar dan juga menghasilkan bawaan keluaran yang diperlukan.

Rangkaian logika yang menyidik pembetulan yang diperlukan itu dapat diturunkan dari isian tabel tersebut. Jelas bahwa suatu pembetulan diperlukan bila jumlah biner itu mempunyai suatun bawaan keluaran K = 1. Enam kombinasi yang lain dari 1010 ampai dengan 1111 yang memerlukan pembetulan , mempunyai suatu 1 dalam kedudukan Z8. Untuk membedakan hal itu dari biner 1000 dan 1001 yang juga mempunyai suatu 1 dalam kedudukan Z8, ditetapkan lebih lanjut bahwa Z4 atau Z2 harus mempunyai suatu 1. Persyaratan untuk suatu pembetulan dan suatu bawaan keluaran dapat dinyatakan oleh fungsi Boole:

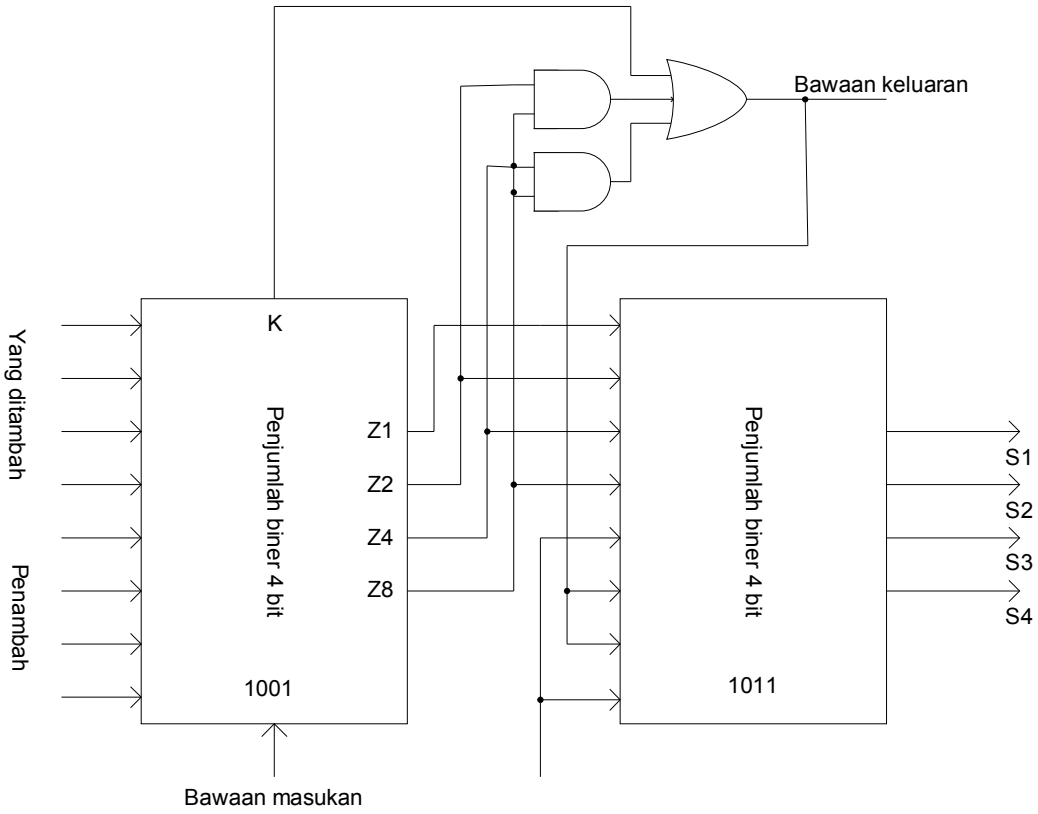
$$\mathbf{C = K + Z8Z4 + Z8Z2}$$

Bila C = 1, perlu ditambahkan 0110 ke jumlah biner itu dan menyediakan suatu bawaan keluaran untuk tingkat berikutnya.

Untuk menambahkan 0110 ke jumlah biner itu, digunakan suatu penjumlah biner 4 bit kedua seperti yang ditunjukkan dalam gambar 8.4. Kedua angka decimal, bersama-sama dengan bawaan masukannya, mula-mula ditambahkan ke penjumlah biner 4 bit yang di kiri untuk menghasilkan jumlah biner itu. Bila bawaan keluaran itu sama dengan 0, tidak ada yang ditambahkan ke jumlah biner itu. Bila sama dengan 1, biner 0110 ditambahkan ke jumlah biner itu melalui penjumlah biner 4 bit yang di kanan. Bawaan keluaran yang dihasilkan dari penjumlah biner bawah itu dapat diabaikan karena hal itu mencatu informasi yang sudah tersedia di kutub bawaan keluaran.

Penjumlah BCD itu dapat dibentuk dengan tiga kemasan IC. Masing-masing dari penjumlah 4 bit itu adalah suatu fungsi MSI dan ketiga gerbang untuk logika pembetulan itu memerlukan satu kemasan SSI. Akan tetapi penjumlah BCD itu telah tersedia dalam satu rangkaian MSI (TTL IC jenis 82S83 adalah suatu penjumlah BCD).

Suatu penjumlah paralel desimal yang menjumlahkan n angka desimal memerlukan n tingkat penjumlah BCD. Bawaan keluaran dari suatu tingkat harus dihubungkan ke bawaan masukan tingkat lebih tinggi berikutnya.



5. Komplemen 1 Adder

Angka positif dalam system komplemen-1 bertanda adalah sama seperti di dalam sistem angka besaran bertanda, akan tetapi angka negatifnya berbeda. Untuk angka negatif ini dinyatakan dalam bentuk komplemen-1. sebagai contoh, bentuk komplemen-1 dari -19 untuk suatu sistem digital 6 bit adalah komplemen dari 010011 (+19), yaitu sama dengan 101100(-19). Begitu pula, oleh karena nol plus adalah 000000, maka nol minus untuk sistem angka komplemen bertanda 1 adalah 111111.

Diatas telah digambarkan tentang penambahan dari dua angka besaran bertanda. Selanjutnya akan digambarkan penambahan dari dua angka komplemen bertanda 1. Perbedaan utama antara kedua penambahan adalah bahwa pada penambahan dari dua angka komplemen 1, bit tanda nya ditambahkan bersama-sama dengan bit besaran. Dengan kata lain, bit tanda ditambahkan sebagaiman bit besaran.

Kasus 1

N_1 dan N_2 adalah positif.

Aturan 1

Bila N_1 dan N_2 adalah positif, tambahkan angka bertanda (tanda dan besaran). Bila bit tanda menunjukkan 1, berarti menyatakan suatu luapan (overflow).

Sebagai contoh, perhatikan penambahan 19 dan 10. Dalam bentuk komplemen-1, 19 adalah 010011 dan 10 adalah 001010, yang jumlahnya adalah :

```
  010011 (+19)
+001010 (+10)
--------

  011101  (+29)
```

19 tambah 19 adalah

```
  010011 (+19)
+010010 (+19)
--------

  100110  (+38)
```

karena bit-tanda adalah 1, berarti menyatakan suatu luapan (overflow)

Kasus 2

N_1 dan N_2 adalah negatif.

Aturan 2

Bila dua angka negatif ditambahkan, selalu terjadi muatan dekat-ujung, yang dihasilkan oleh dua bit-tanda dari angka yang ditambahkan. Muatan ini ditambahkan kepada posisi bit-tanda terkecil.

a) Bila bit-tanda dari angka yang dihasilkan adalah 1, berarti menyatakan bahwa jawaban adalah benar.

b) Bila bit tanda dari angka yang dihasilkan adalah 0, berarti menyatakan suatu luapan (overflow).

Sebagai contoh, jumlah -19 dan -10 :

101100 (-19)
+110101 (-10)

011101 (+29)

19 tambah 19 adalah

010011 (+19)
+010010 (+19)
100110 (+38)

karena bit-tanda adalah 1, berarti menyatakan suatu luapan (overflow)

Kasus 3

N1 dan N2 mempunyai tanda yang berbeda.

Aturan 3

Bila dua angka, N1 dan N2, yang berbeda tanda ditambahkan dan angka positif adalah lebih besar, maka akan terjadi muatan dekat ujung yang harus ditambahkan kepada digit tanda terkecil. Bila angka negatif adalah lebih besar, maka tidak akan terjadi muatan seperti itu.

Sebagai contoh, jumlah 19 dan -10 serta jumlah -19 dan 10 adalah :

010011	(+19)		101100	(-19)
+110101	(-10)	dan	+001010	(+10)
1001000			110110	(-9)
↘+1				
001001(+9)				

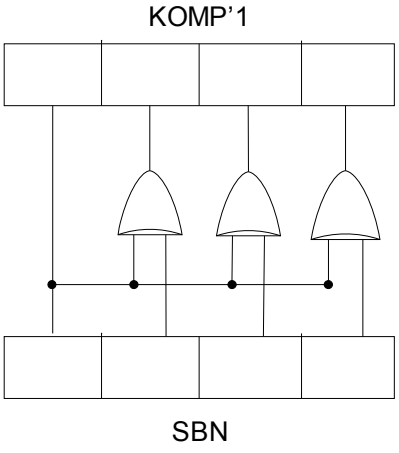
Untuk membuat rangkaian adder dari bilangan komplemen 1, maka terlebih dahulu dibutuhkan suatu rangkaian yang bisa mengkonversi bilangan dari SBN (Signed

Binary Number) ke komplemen'1. Perhatikan tabel di bawah, tabel tersebut menunjukkan perubahan bilangan dari SBN ke komplemen'1.

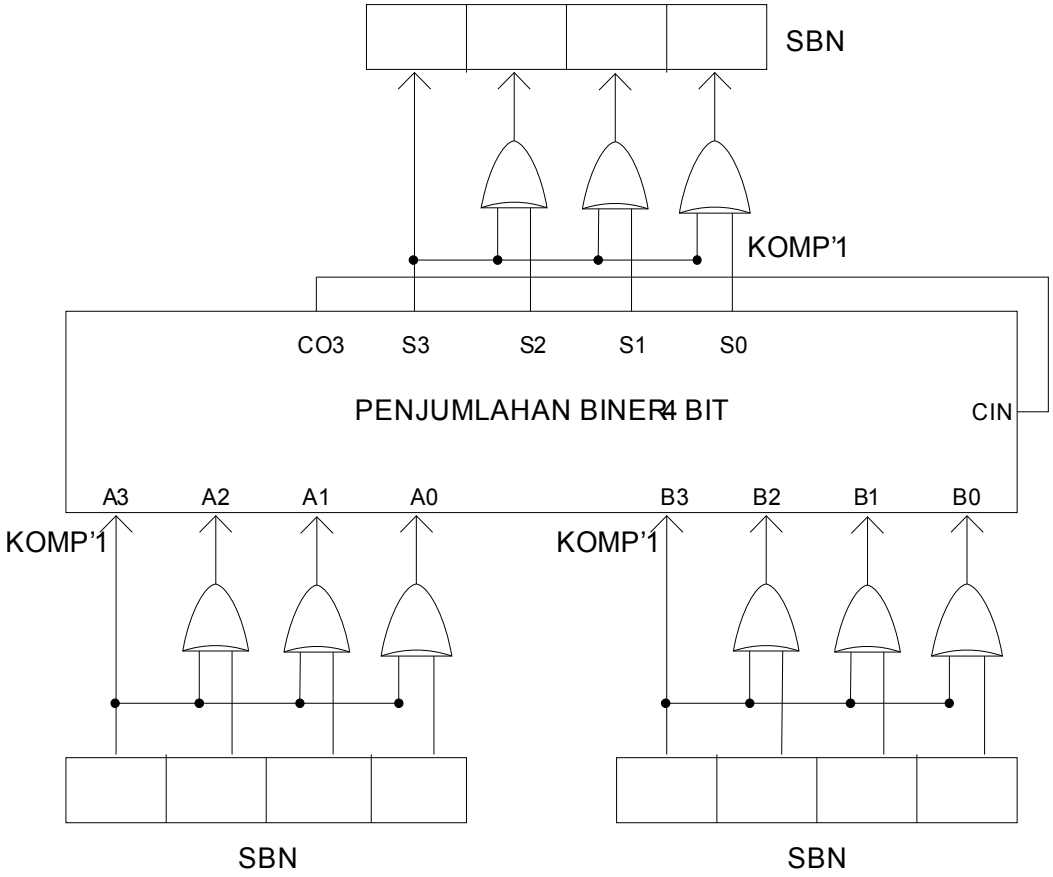
SBN	Komp'1
1011	1100
1101	1010
0110	0110
0101	0101

Dari tabel di atas maka dapat dianalisa, pada digit pertama tidak mengalami perubahan, pada digit selanjutnya mengalami perubahan sesuai dengan (Gerbang EX-OR). Tabel kebenaran untuk EX-Or gate adalah

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0



Rangkaian penjumlah SBN 4 bit yang menerapkan sistem komplemen 1



6. Komplemen 2 Adder

Dalam system ini suatu angka positif dinyatakan dalam bentuk yang sama seperti dalam dua sistem lainnya. Sedangkan angka negatif adalah dalam bentuk komplemen 2. Sebagai contoh, -10 dalam system digital 6 bit adalah 110110. ini diperoleh dari:

-10 = -32 + 22

110110

hanya terdapat nol plus, yaitu semua nol; sedangkan nol minus tidak berlaku.

Penambahan dua angka positif tidak akan dibahas karena penambahan ini adalah sama seperti system komplemen 1.

Kasus 1

N1 dan N2 adalah negatif

Aturan 1

Bila N_1 dan N_2 adalah negatif, muatan harus diperhatikan. Muatan ini dihasilkan dari jumlah dua bit tanda 1. selanjutnya, bit tanda dari jumlah harus 1, karena bernilai negatif. Bila 0 menunjukkan positif dalam bit tanda, berarti menyatakan suatu luapan (overflow).

Sebagai contoh

```
  101101 (-19)
+110110 (-10)
--------
1100011  (-29)
```

diabaikan

dan

```
  101101    (-19)
+ 101101    (-19)
--------
 1011010
```


menyatakan luapan

Kasus 2

N1 dan N2 mempunyai tanda yang berbeda.

Aturan 2

Suatu muatan ditimbulkan bila jumlah adalah positif. Dalam kasus ini, muatan diabaikan. Bila jumlah nya adalah negatif, maka tidak ditimbulkan muatan (carry).

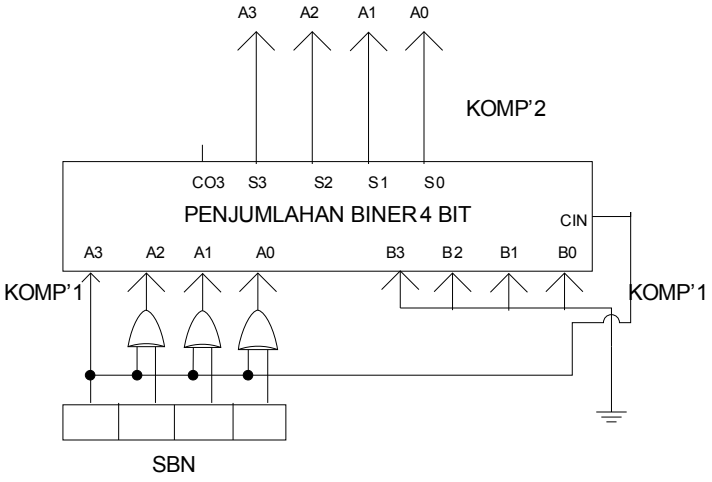
Sebagai contoh, jumlah 19 dan -10 serta jumlah -19 dan 10 adalah :

010011	(+19)		101101	(-19)
+110110	(-10)	dan	+001010	(+10)
1001001			110111	(-9)

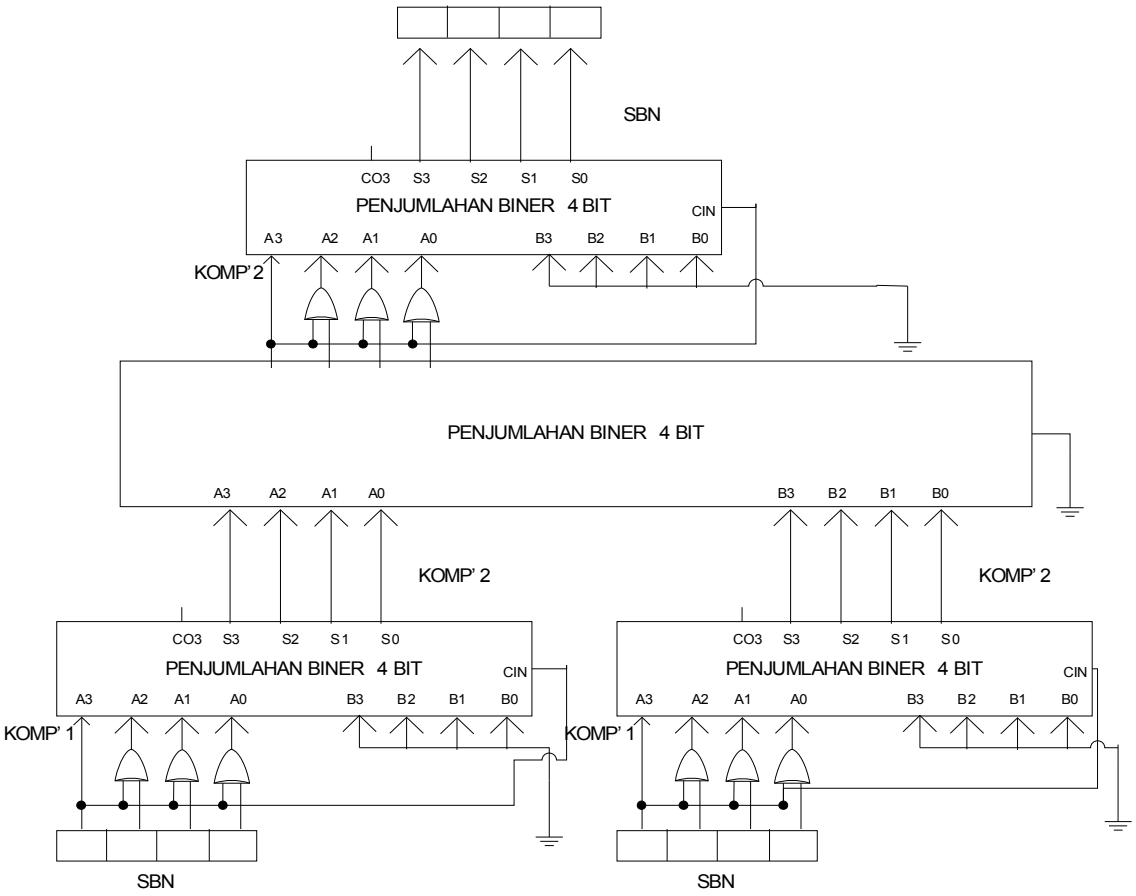
diabaikan

Untuk membuat rangkaian adder dari bilangan komplemen 2, maka terlebih dahulu dibutuhkan suatu rangkaian yang bisa mengkonversi bilangan dari SBN (Signed Binary Number) ke komplemen'2.

Rangkaian pengkonversi bilangan dari SBN ke komplemen 2



Rangkaian penjumlah SBN 4 bit yang menerapkan sistem komplemen 2



7. Carry Look Ahead Adder

Bila panjang penambah-jajar perambatan muatan khusus naik, maka waktu yang diperlukan untuk menyelesaikan penambahan juga naik sebesar waktu tunda (delay time) per tingkat untuk setiap bit yang ditambahkan. Penambahan pandangan muka muatan *(the carry look ahead adder)* mengurangi waktu tunda muatan (time delay) dengan mengurangi jumlah gerbang yang dilewati sinyal muatan. Tabel kebenaran untuk penambah penuh diperlihatkan lagi pada tabel 6, pada tabel ini disertai juga kondisi di mana terjadi pembangkitan muatan. Isian 1, 2, 7, dan 8 memberikan contoh di mana muatan keluaran Ci bebas terhadap Ci-1. Pada isian 1 dan 2, muatan keluaran selalu 0, dan pada isian 7 dan 8 muatan keluaran selalu satu. Hal ini dikenal dengan kombinasi pembangkitan muatan. Isian 3, 4, 5, 6 memperlihatkan kombinasi masukan di mana muatan keluaran tergantung kepada muatan masukkan. Dengan kata lain, Ci adalah 1 hanya jika Ci-1 bernilai 1. hal ini disebut kombinasi perambatan muatan. Andaikan

bahwa G1 menyatakan kondisi pembangkitan muatan 1 dari tingkat I dari penambah jajar dan pi menyatakan kondisi perambatan muatan dari tingkat yang sama.

Isian	Ai	Bi	Ci-1	Ci	Kondisi
1	0	0	0	0	Tidak ada pembangkitan muatan
2	0	0	1	0	
3	0	1	0	0	Perambatan muatan
4	0	1	1	1	
5	1	0	0	0	
6	1	0	1	1	
7	1	1	0	1	Pembangkitan muatan
8	1	1	1	1	

Tanpa menyimpang dari kebiasaan, ambil penambahan dari dua angka biner 4 bit

$A = A_4A_3A_2A_1$

Dan,

$B = B_4B_3B_2B_1$

Dari tabel di atas, fungsi (penyambungan) perambatan muatan dan pembangkitan muatan dalam unsure Ai dan Bi, i=1, 2, 3, dan 4, diperoleh

$Gt = AtBt$

$Pt = At + Bt = At \oplus Bt$

Muatan keluaran kesatuan dari tingkat ke I dapat dinyatakan dalam unsure Gi, Pi, dan Ci-1, yang merupakan muatan keluaran kesatuan dari tingkat ke (i-1), sebagai

$\mathbf{Ci = G_t + P_i * C_{i-1}}$

Sebagai contoh, untuk i=1, 2, 3, dan 4, Ct menjadi

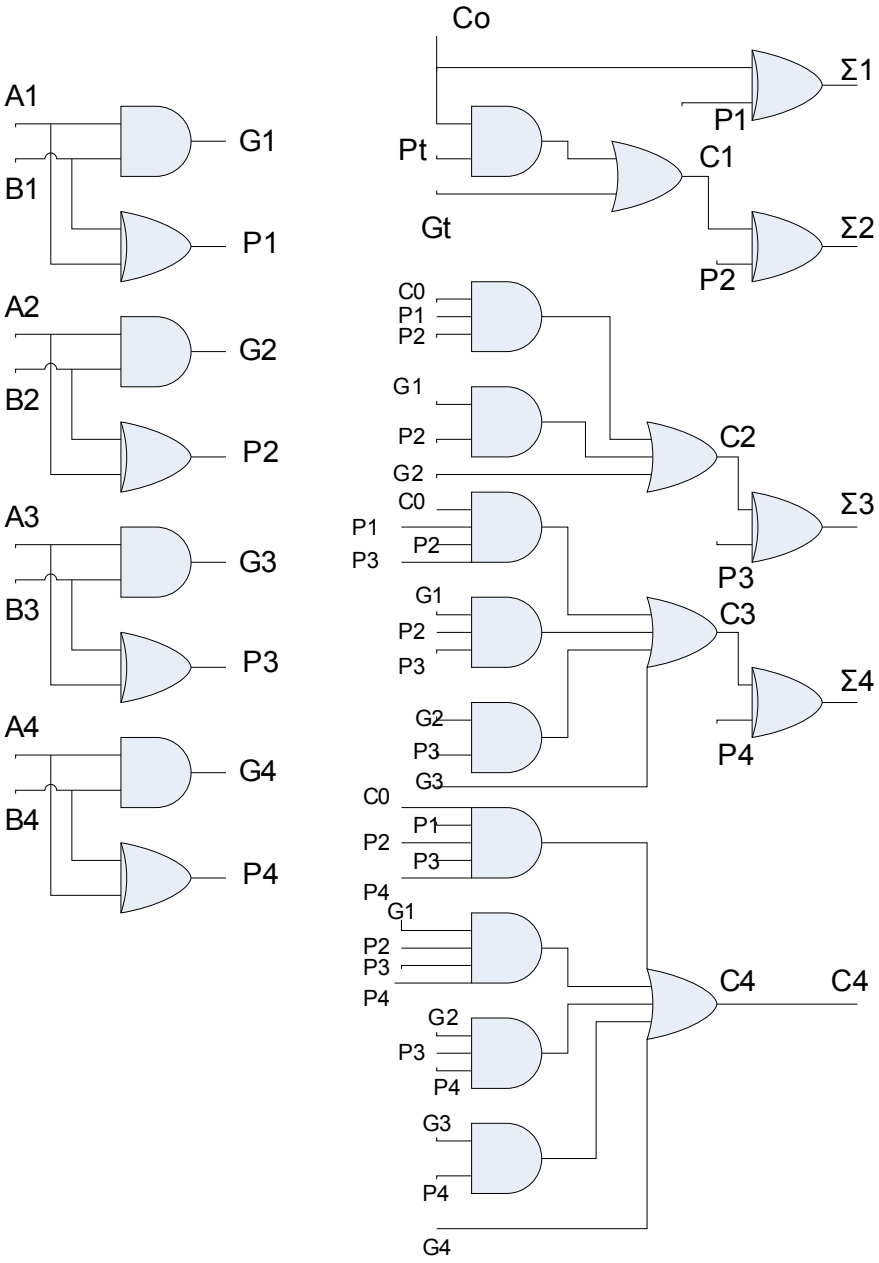
$C_1 = G_1 + P_1C_0$

$C_2 = G_2 + P_2C_1 = G_2 + P_2G_1 + P_2P_1C_0$

$C_3 = G_3 + P_3C_2 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1C_0$

$C_4 = G_4 + P_4C_3 = G_4 + P_4G_3 + P_4P_3G_2 + P_4P_3P_2G_1 + P_4P_3P_2P_1C_0$

Jumlah Σ dari A dan B: $\Sigma = C_4\Sigma_4\Sigma_3\Sigma_2\Sigma_1$, dimana $\Sigma_t = A_t \oplus B_t \oplus C_{i-1}$



Sebagai contoh,

C_0=0 (misalkan) Σ_1=0

A_1=1; G_1=1 C_1=1 Σ_2=0

B_1=1; P_1=0

A_2=0; G_2=0 C_2=1 Σ_3=1

B_2=1; P_2=1

A_3=0; G_3=0 C_3=0 Σ_4=1

B_3=0; P_3=0

A_4=1; G_4=0 C_4=0

B_4=0; P_4=1

Periksa : A=1001 (9)

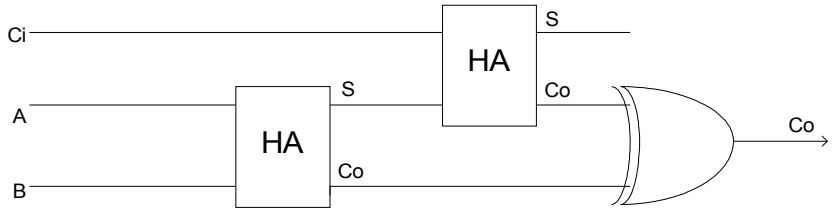
+B=0011 (3)

Σ=1100 (12)

Contoh Soal

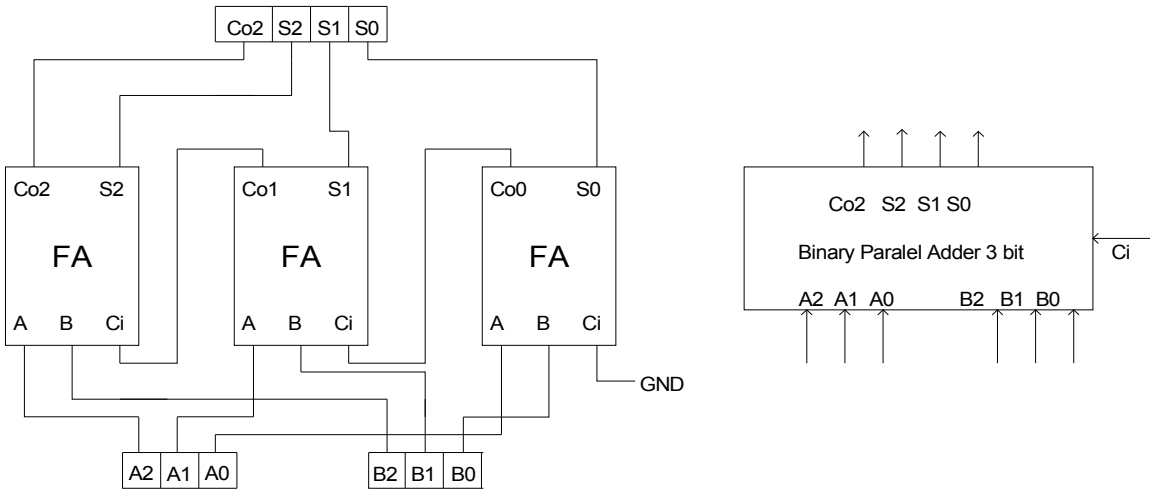
Rancanglah suatu Full Adder (FA) yang dibentuk dari Half Adder (HA)

Jawab:



Rancanglah suatu penjumlah biner yang dapat menjumlahkan 2 data biner 3 bit

Jawab :



B. Pengurangan

Dalam proses pengurangan biner, dapat ditemukan jenis pengurang paro (Half Subtractor) dan pengurang penuh (Full Subtractor). Proses pengurangan dapat dilakukan secara komplemen ataupun biner secara langsung. Ingatlah kaidah-kaidah bagi pengurangan biner ;

0-0 = 0 dengan pinjaman 0

0-1 = 1 dengan pinjaman 1

1-0 = 0 dengan pinjaman 0

1-1 = 0 dengan pinjaman 0

Tabel 5.4 meringkaskan hasil-hasil ini dengan memberikan daftar kaidah pengurangan bagi A-B

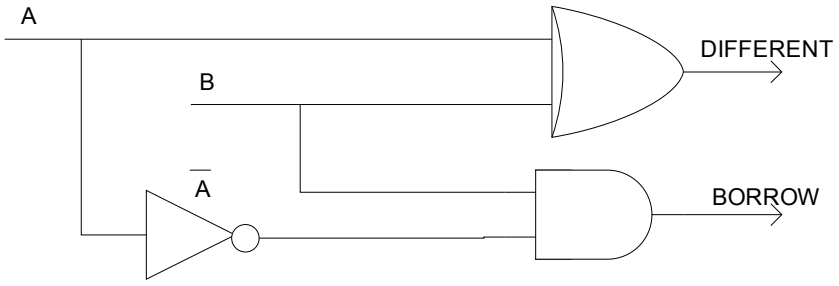
A	B	Bo	D
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

Ket : *Bo* = *Borrow*

D = Different

Rangkaian logika mana yang mempunyai tabel kebenaran seperti tabel 5.4?. pertama, keluaran Different adalah 1 bi;la A dan B berbeda. Maka, kita dapat menggunakan sebuah gerbang EX-OR untuk menghasilkan keluaran different ini. Selanjutnya, keluaran borrow adalah 1 hanya bila A adalah 0 dan B adalah 1. kita dapat memperoleh keluaran pinjaman ini dengan meng-AND-kan A dan B.

Gambar 5.16 memperlihatkan salah satu cara untuk membangun suatu rangkaian half subtractor yang mengurangkan sebuah angka biner dari angka lainnya. Rangkaian pada gambar 5.16 mempunyai tabel kebenaran identik dengan tabeel 5.4. Dapat Anda lihat bahwa pinjaman (borrow) hanya ada bila A= 0 dan B = 1. selanjutnya, keluaran pinjaman (different) adalah sesuai bagi masing-masing di antara keempat kemungkinan kombinasi A-B.

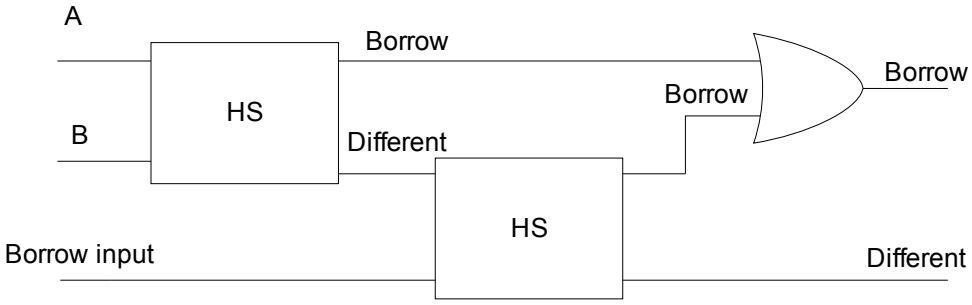


Gambar 5.16 Half Subtractor

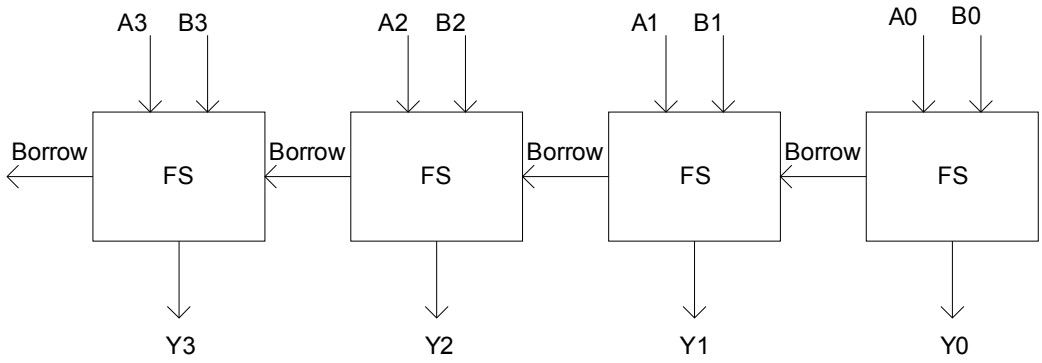
Half subtractor hanya menangani 2 bit pada suatu saat dan hanya dapat digunakan bagi kolom paling ringan (least significant) pada suatu masalah pengurangan. Untuk menangani kolom yang lebih tinggi, kita membutuhkan pengurang penuh (full subtractor). Gambar 5.17 memperlihatkan sebuah full subtractor; rangkaian ini menggunakan dua buah half adder dan sebuah OR gate.

Half dan full subtractor adalah analog dengan half dan full adder; dengan menggandengkan half dan full subtractor seperti terlihat pada gambar 5.18, diperoleh suatu sistem yang secara langsung mengurangkan B3B2B1B0 dari A3A2A1A0.

Penambah dan pengurang memberikan rangkaian rangkaian dasar yang dibutuhkan bagi aritmatika biner; perkalian dan pembagian dapat dilakukan dengan penambahan dan pengurang berulang (dibahas dalam bab-bab selanjutnya, setelah kita membahas register).



Gambar 5.17 Full Subtractor



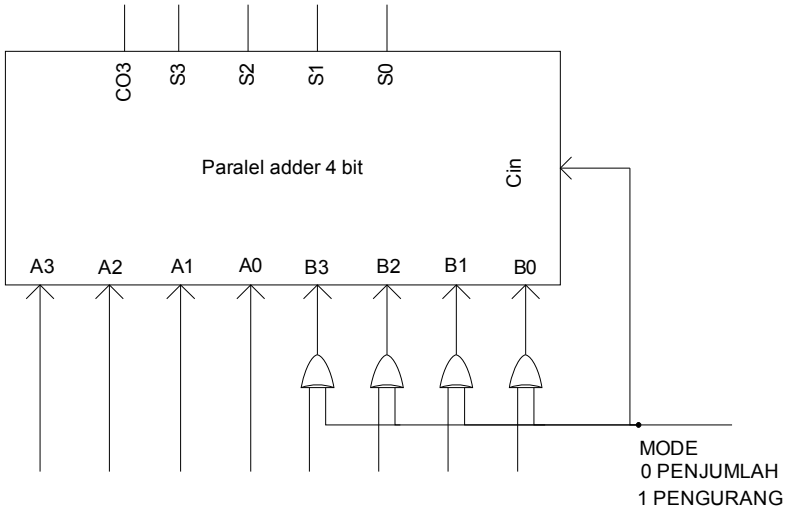
Gambar 5.18 Pengurang Paralel Biner 4 bit

Contoh Soal ;

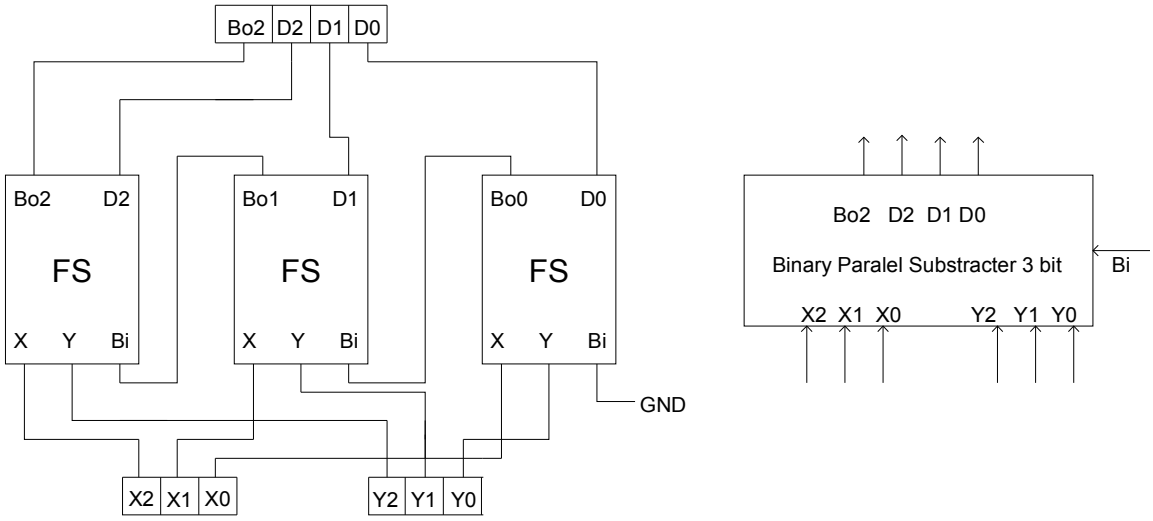
1) Rancang suatu rangkaian penjumlah / pengurang yang dapat menjumlahkan atau mengurangkan 2 data biner komplemen 2 (4 bit) dengan keteneuan sebagai berikut :

- Jika input mode operasi 0, maka rangkaian berfungsi sebagai adder
- Jika input mode operasi 1, maka rangkaian berfungsi sebagai subtractor

Jawab :



2) Rancanglah suatu pengurang biner yang dapat mengurangkan 2 data biner 3 bit



BAB VII
SISTEM SANDI

A. Sandi Biner

Sebuah bit, menurut definisi adalah sebuah angka biner (binary digit). Bila digunakan dalam hubungan dengan suatu sandi biner, sebuah bit merupakan suatu besaran biner yang sama dengan 0 atau 1. Untuk mewakili suatu kelompok yang terdiri dari 2^n unsure yang berbeda dalam suatu sansi biner akan memerlukan paling sedikit n bit itu. Hal itu adalah karena dimungkinkan untuk menyusun n bit itu dalam 2^n cara yang berlainan. Meskipun banyaknya bit minimum yang diperlukan untuk menjadikan 2^n besaran yang berbeda itu adalah n, tidak ada batas maksimum banyaknya bit yang dapat dipergunakan untuk suatu sandi biner. Jadi untuk m karakter yang diwakili sebagai sandi biner, diperlukan sekurang-kurangnya n bit yang diperoleh menurut hubungan berikut : $2^n \geq m$. Berbagai macam sandi untuk bilangan decimal dapat diperoleh dengan mengatur 4 bit atau lebih dalam 10 kombinasi yang berlainan. Beberapa diantaranya ditunjukkan seperti pada tabel berikut :

Bilangan Decimal	BCD 8421	XS-3	8-4-2-1	2421	Bikuiner 5043210	Sandi gray
0	0000	0011	0000	0000	0100001	0000
1	0001	0100	0111	0001	0100010	0001
2	0010	0101	0110	0010	0100100	0011
3	0011	0110	0101	0011	0101000	0010
4	0100	0111	0100	0100	0110000	0110
5	0101	1000	1011	1011	1000001	0111
6	0110	1001	1010	1100	1000010	0101
7	0111	1010	1001	1101	1000100	0100

8	1000	1011	1000	1110	1001000	1100
9	1001	1100	1111	1111	1010000	1101
10	-	-	-	-	-	1111
11	-	-	-	-	-	1110
12	-	-	-	-	-	1010
13	-	-	-	-	-	1011
14	-	-	-	-	-	1001
15	-	-	-	-	-	1000

B. Sandi BCD

BCD (Binary Coded Decimal-desimal yang disandikan biner) merupakan penetapan langsung dari setara binernya. Sandi tersebut juga dikenal sebagai sandi BCD 8421 yang menunjukkan bobot untuk masing-masing kedudukan bitnya. Oleh sebab itu, seringkali sandi BCD dikatakan sebagai sandi berbobot. Kolom kedua pada tabel diatas menunjukkan tabel sandi BCD itu. Sebagai contoh, bilangan decimal 1996 dapat disandikan menurut BCD sebagai : 1996 = 0001 1001 1001 0110. Perlu diperhatikan bahwa pengubahan suatu bilangan decimal ke bilangan biner berbeda dengan penyandian suatu bilangan decimal, meskipun dalam kedua hal tersebut hasilnya sama-sama berupa suatu deretan bit. Untuk sandi BCD ini, sandi bilangan decimal 0 sampai 9 sama dengan bilangan biner setaranya. Namun untuk diatas 9, sandi BCD berbeda dengan bilangan biner setaranya. Misalnya setar biner untuk 11 adalah 1011, tetapi sandi BCD untuk 11 adalah 0001 0001. Oleh karena itu, perlu diingat bahwa suatu deretan bit (angka) 0 dan 1 dalam suatu system digital kadang-kadang mewakili suatu bilangan biner dan pada saat yang lain merupakan informasi diskrit yang ditentukan oleh suatu sandi biner tertentu. Keunggulan utama sandi BCD adalah mudahnya mengubah dari dan ke bilangan decimal. Sedangkan kerugiannya adalah sandi yang tidak akan berlaku untuk operasi metematika yang hasilnya melebihi 9. Sandi BCD hanya menggunakan 10 dari 16 kombinasi yang tersedia. 6 kelompok bit yang tidak terpakai adalah 1010, 1011, 1100, 1101, 1110, dan 1111. Sandi BCD merupakan sandi radiks campuran, dalam setiap kelompok 4 bitnya merupakan sistem biner, tetapi merupakan decimal untuk kelompok demi kelompoknya.

C. Sandi Excess-3 (XS-3)

Sandi XS-3 (yang berasal dari excess-3, artinya kelebihan 3) merupakan sandi penting lainnya yang erat hubungannya dengan sandi BCD. Sesuai dengan namanya, penetapannya diperoleh dari nilai binernya, sama seperti pada sandi BCD dan menambahnya dengan 3. Kolom ketiga pada tabel diatas menunjukkan sandi XS-3 tersebut. Sebagai contoh, untuk mengubah 23 menjadi sandi XS-3 adalah sebagai berikut : 23 = 0101 0110 , dengan ditambah 3 untuk setiap angka decimal yang diketahui dan hasilnya diubah menjadi bilangan biner setaranya akan menghasilkan sandi XS-3 yang diminta. Seperti halnya pada BCD, sandi XS-3 hanya menggunakan 10 dari 16 kombinasi yang tersedia. 6 kelompok bit yang tidak digunakan adalah 0000, 0001, 0010, 1101, 1110, dan 1111. Sandi XS-3 adalah sandi tidak berbobot karena tidak seperti halnya pada sandi BCD yang kedudukan bitnya mempunyai bobot tertentu. Sandi XS-3 merupakan sandi yang mengkomplemenkan dirinya sendiri. Hal itu terjadi karena setiap komplemen-1 dari bilangan XS-3 adalah komplemen-9 dari bilangan desimalnya. Misalnya, 0101 dalam sandi XS-3 mewakili angka decimal 2. Komplemen-1 0101 adalah 1010 yang merupakan angka decimal 7 dan 7 adalah komplemen-9 dari 2. Sandi XS-3 mempunyai keunggulan dibandingkan dengan sandi BCD karena semua operasi penjumlahan untuk XS-3 berlangsung seperti penjumlahan biner biasa dan juga karena XS-3 merupakan sandi yang mengkomplemenkan dirinya sendiri. Pengurangan dengan komplemen-1 dan komplemen-2 dapat dilakukan untuk sandi XS-3.

D. Sandi Gray

Sandi Gray merupakan suatu sandi 4 bit tanpa bobot dan tidak sesuai untuk operasi aritmatika. Sandi Gray ini sangat berguna untuk peralatan masukan/keluaran (input/output devices), pengubah analog ke digital dan peralatan tambahan lainnya. Pada tabel diatas kolom paling kanan menunjukkan perwakilan sandi gray untuk bilangan 0 sampai 15. Terlihat bahwa setiap perubahan dari 1 bilangan decimal yang 1 dengan yang berikutnya hanya 1 bit dalam sandi gray itu yang berubah. Itulah sebabnya sandi gray digolongkan ke kelompok sandi perubahan-minimum (minimum-change code).

1. Perubahan Biner ke Gray

Inilah cara untuk mengubah dari biner ke Gray :

Desimal	Sandi Gray	Biner
0	0000	0000
1	0001	0001
2	0011	0010
3	0010	0011
4	0110	0100
5	0111	0101
6	0101	0110
7	0100	0111
8	1100	1000
9	1101	1001
10	1111	1010
11	1110	1011
12	1010	1100
13	1011	1101
14	1001	1110
15	1000	1111
…	…	…

Angka Gray pertama sama dengan angka biner pertama. Tambahkan masin-masing pasangan bit berdampingan untuk mendapatkan angka Gray berikutnya.

Abaikan setiap bawaan [5].

Contoh merupakan cara terbaik untuk menjelaskan perubahan dari biner ke Gray. Ambilah bikangan biner 1100. Inilah cara untuk mencari bilangan sandi Gray yang bersangkutan :

- **LANGKAH 1** Angka Gray pertama sama dengan angka biner pertama**.**

1 1 0 1 biner

1 Gray

- **LANGKAH 2** Selanjutnya, tambahkan 2 bit pertama pada bilangan biner, dengan mengabaikan setiap bawaan. Jumlahnya merupakan angka Gray berikutnya.

  1 1 0 0 biner

  1 0 Gray

ket : [5] hal ini secara formal disebut penambahan mod-2, atau penambahan OR-eksklusif. Keempat kaidah bagi penambahan jenis ini adalah : 0 + 0 = 0, 0 + 1 = 1, 1 + 0 =1, 1 + 1 = 0

Dengan perkataan lain, tambahkan 2 bit pertama pada bilangan biner untuk mendapatkan 1 + 1 = 0 dengan bawaan 1. Tuliskan angka 0, namun abaikan angka 1.

- **LANGKAH 3** Tambahkan 2 angka biner berikutnya untuk mendapatkan angka Gray berikutnya.

  1 1 0 0 biner

  1 0 1 Gray

- **LANGKAH 4** Tambahkan 2 angka biner terakhir untuk mendapatkan angka Gray.

  1 1 0 0 biner

  1 0 1 0 Gray

Oleh karenanya, 1010 adalah ekivalen sandi-Gray bagi bilangan biner 1100.

2. Perubahan Gray ke biner

Untuk mengubah dari *sandi Gray ke biner*, kita menggunakan metoda yang serupa, tetapi tidak tepat sama. Sekali lagi, contoh merupakan cara yang terbaik untuk metode ini. Marilah kita mengubah sandi Gray 101110101 kembali ke ekivalen binernya.

- **LANGKAH 1** Ulangilah angka paling berbobot

  1 0 1 1 1 0 1 0 1 Gray

  1 biner

- **LANGKAH 2** Tambahkan *secara diagonal* seperti terliha di bawah ini untuk mendapatkan angka biner berikutnya.

  1 0 1 1 1 0 1 0 1 Gray

  1 1 biner

  (1 + 0 = 1)

- **LANGKAH 3** Lanjutkan menambahkan *secara diagonal* untuk mendapatkan angka-angka biner selanjutnya.

  1 0 1 1 1 0 1 0 1 Gray

  1 1 0 1 0 0 1 1 0 biner

Dengan metode ini, Anda dapat ,engubah Gray ke biner dan sebaliknya bilamana dibutuhkan

E. Parity Bit

Parity Bit adalah digit "1" atau yang ditempatkan pada kelompok bit dari suatu sandi yang berfungsi untuk mengetahui adanya kecacatan (validasi) atau kesalahn dari kelompok bit yang berupa data input. Parity Bit dapat dibagi menjadi 2, yaitu :

1. Parity genap (Odd Parity), dipakai untuk membuat agar jumlah dari digit 1 pada kelompok sandi menjadi genap. Misalnya bila jumlah digit 1 semula sudah genap, maka paritynya adalah 0. Jika jumlah digit 1 semula ganjil, maka bit paritynya adalah 1 sehingga jumlah digit 1 akan menjadi genap.
2. Parity Ganjil (Even Parity), dipakai untuk membuat agar jumlah dari digit 1 pada kelompok bit menjadi ganjil. Misalnya bila jumlah digit 1 semula sudah ganjil, maka

paritynya adalah 0. Jika jumlah digit 1 semula genap, maka bit paritynya adalah 1 sehingga jumlah digit 1 akan menjadi ganjil

Tabel Parity pada sandi BCD 8421

Decimal	Parity Genap	Parity Ganjil
0	00000	10000
1	10001	00001
2	10010	00010
3	00011	10011
4	10100	00100
5	00101	10101
6	00110	10110
7	10111	00111
8	11000	01000
9	01001	11001

BAB VIII
SISTEM KOMPLEMEN

Dalam Sistem decimal terdapat sistem komplemen yang terbagi kedalam tiga kelompok, yaitu :

1. Pada bilangan decimal yaitu komplemen 9 (komplemen ganjil) dan komplemen 10 (komplemen genap).

2. Pada bilangan biner yaitu komplemen 1 (komplemen ganjil) dan komplemen 2 (komplemen genap).
3. Pada bilangan hexadecimal yaitu komplemen 15 (komplemen ganjil) dan komplemen 16 (komplemen genap).

A. Komplemen 1

Komplemen 1 bagi suatu bilangan biner adalah bilangan yang terjadi bila kita mengubah masing-masing 0 menjadi 1 dan masing-masing 1 menjadi 0. Dengan perkataan lain, komplemen 1 bagi 100 adalah 011. Komplemen 1 bagi 1001 adalah 0110. Contoh lain komplemen 1 adalah :

1010 mempunyai komplemen 1 berbentuk 0101

1110 mempunyai komplemen 1 berbentuk 0001

0011 mempunyai komplemen 1 berbentuk 1100

B. Komplemen 2

Komplemen 2 adalah bilangan biner yang terjadi bila kita menambahkan 1 kepada kepada komplemen 1, yakni komplemen 2 = komplemen 1 + 1. Sebagai contoh, untuk mencari komplemen 2 bagi 1011, pertama-tama carilah komplemen 1 nya yaitu 0100. Selanjutnya tambahkan 1 kepada 0100 untuk mendapatkan 0101 (komplemen 2 bagi 1011). Contoh lain komplemen 2 dari 11001, yakni komplemen 1 nya adalah 00110 dan setelah ditambahkan 1 menjadi 00111 (komplemen 2 bagi 11001).

C. Komplemen 9 dan Komplemen 10

Komplemen 9 (sama dengan komplemen 1) diperoleh dengan mengurangkan masing-masing angka decimal dari 9. Sebagai contoh, komplemen 9 bagi 25 diperoleh sebagai berikut :

```
  99
 -25
 ---
  74  (komplemen 9 bagi 25)
```

Sebagai contoh lain, komplemen 9 bagi 6291 adalah :

```
 9999
-6291
```

3708 (komplemen 9 bagi 6291)

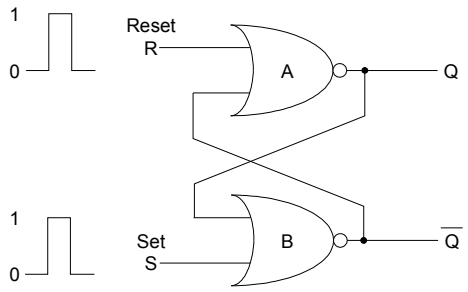
Komplemen 10 (sama dengan komplemen 2) bagi suatu bilangan bulat decimal satu lebih besar daripada komplemen 9 nya. Sebagai contoh dari komplemen 9 diatas, yaitu komplemen 10 untuk 25 adalah komplemen 9 + 1 = 74 + 1 = 75 dan komplemen 10 untuk 6291 adalah komplemen 9 + 1 = 3708 +1 = 3709. Komplemen 9 dan komplemen 10 dapat digunakan untuk mengurangkan bilangan decimal.

BAB IX
FLIP-FLOP

Rangkaian flip-flop dapat mempertahankan suatu keadaan biner dalam waktu yang tak terbatas sampai suatu sinyal masukan baru datang untuk mengubah keadaan itu. Perbedaan utama diantara berbagai jenis flip-flop itu adalah banyaknya masukan yang dimiliki dan perilaku bagaimana masukan itu mempengaruhi keadaan biner dalam flip-flop tersebut.

A. Rangkaian Flip-flop Dasar

Suatu rangkaian flip-flopdapat disusun dengan dua gerbang NOR atau dua gerbang NAND. Susunan itu ditunjukkan pada gambar a dan gambar b. Masing-masing rangkaian itu membentuk suatu flip-flop dasar yang merupakan dasar pengembangan bagi jenis-jenis flip-flop yang lain. Hubungan silang dari salah satu gerbang ke masukan gerbang yang lain merupakan suatu jalur umpan-balik. Dengan alas an itu rangkaian tersebut dapat digolongkan kepada rangkaian urutan tak-serempak. Masing-masing flip-flop itu mempunyai dua keluaran, Q dan Q dan dua masukan, set dan reset. Masukan set membuat flip-flop menjadi dalam keadaan set atau bernilai logik 1 pada keluaran normalnya (Q), dan masukan reset membuat flip-flop menjadi dalam keadaan bebas (*clear*) atau mempunyai nilai logik 0 pada keluaran normalnya. Jenis flip-flop ini sering dikenal sebagai flip-flop RS gandengan langsung (*direct coupled RS flip-flop*), R dan S merupakan huruf pertama nama masukannya.



Gambar a

Rangkaian flip-flop dasar dengan gerbang NOR

Tabel 1

Tabel kebenaran flip-flop dasar dengan gerbang NOR

S	R	Q	$\overline{Q}$	
1	0	1	0	
0	0	1	0	
0	1	0	1	(setelah S = 1, R = (
0	0	0	1	(setelah S = 0, R = '
1	1	0	0	

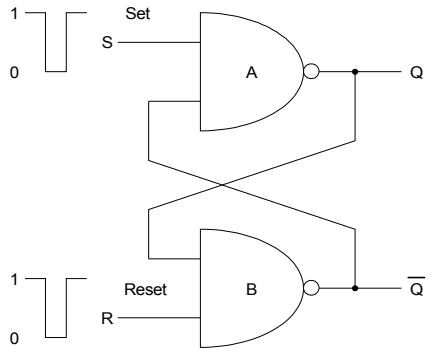
Untuk menganalisis rangkaian pada gambar a, harus diingat bahwa keluaran suatu gerbang NOR adalah 0 jika salah satu masukannya sama dengan 1 dan keluaran gerbang NOR adalah 1 hanya jika semua masukannya sama dengan 0. Sebagai titik awal, misalnya masukan set adalah 1 dan masukan reset sama dengan 0. Karena gerbang B mempunyai sebuah masukan 1, keluaran Q' harus sama dengan 0 yang mengakibatkan kedua masukan ke gerbang A itu sama dengan 0 dan keluarannya Q sama dengan 1. Bila masukan set dikembalikan ke 0, keluarannya tetap sama. Hal itu adalah karena keluaran Q tetap 1 sehingga masih ada sebuah masukan 1 pada gerbang B, yang selanjutnya membuat keluaran Q' tetap 0. Akibatnya kedua masukan ke gerbang A sama dengan 0 dan keluaran Q tetap sama dengan 1. Dengan cara yang sama dapat dibuktikan bahwa suatu 1 pada masukan reset akan mengubah keluaran Q menjadi 0 dan Q' menjadi 1. Bila masukan reset itu dikembalikan ke 0, keluarannya tidak berubah.

Bila sebuah 1 diberikan bersama-sama ke masukan set dan reset, kedua keluarannya Q dan Q' menjadi 0. Dalam praktek keadaan semacam itu harus dihindari. Suatu flip-flop mempunyai dua keadaan stabil. Bila Q = 1 dan Q' = 0 dikatakan flip-flop itu dalam keadaan set (atau keadaan 1). Dan Q = 0 dan Q' = 1 merupakan keadaan bebas (atau keadaan 0). Keluaran Q dan Q' merupakan komplemen antara yang satu dengan yang lain dan dikatakan sebagai keluaran normal dan komplemen flip-flop tersebut. Keadaan biner suatu flip-flop diambil dari nilai keluaran normalnya.

Dalam operasi normal, kedua masukan suatu flip-flop akan tetap 0 kecuali bila keadaan flip-flop itu akan diubah. Pengenaan 1 sesaat ke masukan set menyebabkan flip-flop itu menjadi dalam keadaan set. Masukan set itu harus kembali ke 0 sebelum suatu 1 diberikan ke masukan resetnya. Pengenaan 1 sesaat ke masukan reset menyebabkan flip-flop tersebut menjadi dalam keadaan bebas kembali. Bila kedua masukannya itu mula-

mula sama dengan 0, dan bila suatu 1 dikenakan ke masukan set sedangkan flip-flop itu dalam keadaan set atau bila sebuah 1 yang diberikan ke masukan reset sedangkan flip-flop itu dalam keadaan bebas, maka keadaan keluarannya tidak akan berubah. Bila sebuah 1 dikenakan sekaligus ke masukan set dan reset, kedua keluarannya akan sama dengan 0. Keadaan itu tidak terdefinisi dan biasanya dihindari. Jika kedua masukan itu menjadi 0 kembali, keadaan flip-flop menjadi tak tentu dan tergantung pada masukan mana yang menerima 1 lebih lama sebelum kembali ke 0.

Rangkaian flip-flop dasar NAND pada gambar b bekerja dengan kedua masukannya dalam keadaan normal sama dengan 1 kecuali bila keadaan flip-flop itu akan diubah. Pengenaan 0 sesaat ke masukan set menyebabkan keluaran Q menjadi 1 dan Q menjadi 0 membuat flip-flop menjadi dalam keadaan set. Setelah masukan set itu kembali ke 1, 0 sesaat pada masukan reset akan menyebabkan keadaan flip-flop menjadi bebas. Bila kedua masuka itu menjadi 0 bersama-sama, kedua keluaran pada flip-flop itu sama dengan 1, suatu keadaan yang harus dihindari dalam praktek.



Gambar b

Rangkaian flip-flop dasar dengan gerbang NAND

Tabel 2

Tabel kebenaran flip-flop dasar dengan gerbang NAND

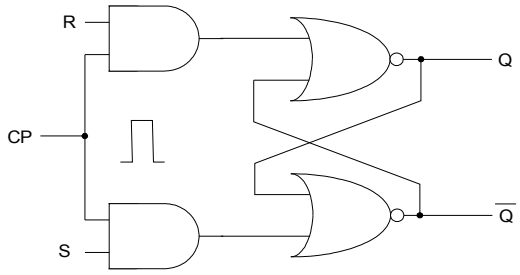
S	R	Q	$\overline{Q}$	
1	0	0	1	
1	1	0	1	
0	1	1	0	(setelah S = 1, R =
1	1	1	0	(setelah S = 0, R =
0	0	1	1	

B. Flip-flop RS

Suatu flip-flop dasar pada dasarnya adalah suatu rangkaian urutan tak serempak. Dengan menambah suatu gerbang ke setiap masukan rangkaian dasar itu, flip-flop tersebut dapat diubah untuk menanggapi masukan selama adanya suatu pulsa waktu. Flip-flop RS menurut waktu yang ditunjukkan pada gambar a itu terdiri dari flip-flop NOR dasar dengan dua gerbang AND. Keluaran kedua gerbang AND tersebut tetap 0 selama pulsa waktu (yang diberi lambang CP - *clock pulse*) sama dengan 0, tanpa memandang nilai masukan S dan R nya. Bila pulsa waktu itu menjadi 1, informasi dari masukan S dan R diijinkan untuk masuk ke flip-flop dasar tersebut. Keadaan set tercapai dengan S = 1, R = 0, dan CP = 1. untuk mengubahnya menjadi keadaan bebas, masukan S harus 0, R= 1, dan CP = 1. Dengan masukan R dan S yang keduanya sama dengan 1, adanya pulsa waktu akan menyebabkan kedua keluaran flip-flop itu sesaat sama dengan 0. Bila pulsa waktu itu hilang, keadaannya menjadi tak tentu, dapat dalam keadaan set atau bebas, tergantung apakah masukan set atau reset yang lebih lama sama dengan 1 sebelum berubah menjadi 0 pada akhir pulsa waktu tersebut.

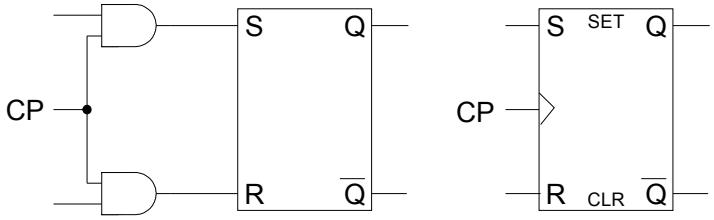
Tanggapan flip-flop menurut waktu merupakan praktek yang umum dijumpai dalam sistem digital karena perubahan dalam sistem itu umumnya diinginkan terjadi serentak menurut kendali sumber waktu. Oleh sebab itu, flip-flop menurut waktu disebut sebagai suatu rangkaian urutan serempak.

Dua lambang untuk flip-flop RS ditunjukkan pada gambar b. Gerbang AND dengan masukan pulsa waktu dapat dilukis diluar lambang tersebut, atau suatu lambang dengan tanda CP digunakan untuk menunjukkan bahwa keluaran flip-flop tersebut tidak akan terpengaruh kecuali bila ada pulsa waktu pada masukan yang bertanda CP itu.



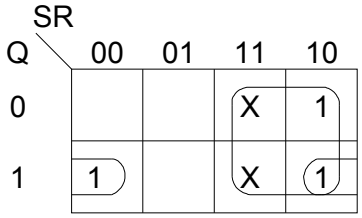
Gambar a

Diagram logika



Gambar b

Lambang tanpa dan dengan pulsa waktu



$Q(t + 1) = S + \overline{R}Q$

$SR = 0$

Gambar c

Persamaan karakteristik

Tabel 1

Tabel karakteristik flip-flop RS menurut waktu

Q	S	R	Q(t + 1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	tak tentu
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	tak tentu

Dalam praktek flip-flop menurut waktu itu seringkali diinginkan untuk membuat flip-flop tersebut dalam keadaan set atau bebas tanpa harus menunggu datangnya pulsa waktu. Untuk itu umumnya flip-flop menurut waktu selalu dilengkapi dengan masukan set atau reset langsung. Masukan langsung itu sering diberi label SET atau CLR (*clear*-bebas) untuk membedakannya dengan masukan S (set) dan R (reset) yang bekerja menurut waktu seperti yang ditunjukkan pada gambar b.

Pada awal penggunaan suatu flip-flop sering tidak dapat diramal perilakunya, dalam hal semacam itu masukan SET dan CLR berguna untuk mengawali operasi suatu sistem dengan keadaan flip-flop yang terdefinisi. Persamaan karakteristik flip-flop itu diturunkan dari gambar c. Persamaan itu memberikan nilai keadaan berikutnya sebagai fungsi keadaan sekarang dan masukan-masukannya. Persamaan karakteristik itu adalah pernyataan aljabar untuk informasi biner pada tabel karakteristiknya. Dua keadaan tak tentu pada flip-flop itu ditandai dengan x dalam peta itu karena dapat bernilai 1 atau 0.

akan tetapi hubungan SR = 0 harus dimasukan sebagai bagian persamaan karakteristik itu untuk menunjukkan bahwa S dan R tidak dapat sama dengan 1 secara serentak.

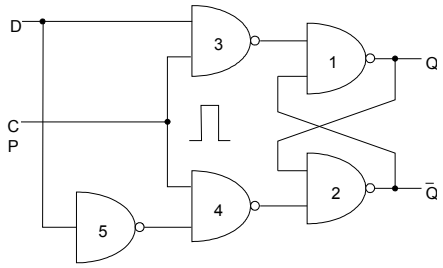
Tabel karakteristik flip-flop tersebut ditunjukkan pada tabel 1. Tabel itu merupakan ringkasan operasi flip-flop dalam bentuk tabel. Q adalah keadaan biner flip-flop pada suatu waktu yang diketahui (yang dikatakan sebagai keadaan sekarang), kolom R dan S memberikan nilai yang dapat terjadi untuk masukannya dan Q (t + 1) adalah keadaan flip-flop setelah timbulnya suatu pulsa waktu (dikatakan sebagai keadaan berikutnya).

C. Flip-flop D

Flip-flop D yang ditunjukkan pada gambar a merupakan modifikasi flip-flop RS menurut waktu. Gerbang NAND 1 dan 2 membentuk suatu flip-flop dasar. Gerbang 3 dan 4 mengubahnya menjadi suatu flip-flop menurut waktu. Masukan D langsung diberikan ke masukan S dan komplemennya melalui gerbang 5, dikenakan ke masukan R. Selama masukan pulsa waktu 0, gerbang 3 dan 4 mempunyai nilai 1 pada keluarannya, tanpa memandang nilai masukannya yang lain. Hal itu sesuai dengan persyaratan bahwa kedua masukan flip-flop NAND dasar tersebut (gambar b) pada awalnya mempunyai nilai logika 1. Masukan D dicacah (*sampled*) selama adanya pulsa waktu. Jadi pada saat masukan D sama dengan 1, keluaran gerbang 3 menjadi 0 sehingga mengakibatkan flip-flop itu menjadi dalam keadaan set (kecuali bila flip-flop itu telah berada dalam keadaan set sebelumnya). Jika masukan D itu sama dengan 0, keluaran gerbang 4 menjadi 0 yang mengubah flip-flop tersebut menjadi dalam keadaan bebas.

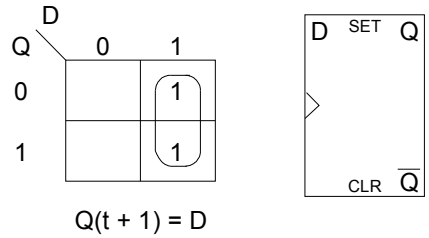
Flip-flop D itu mendapatkan namanya karena kemampuannya memindahkan ‘data’ ke dalam flip-flop. Rangkaian flip-flop itu pada dasarnya adalah rangkaian flip-flop RS dengan sebuah pembalik di masukan R nya. Adanya penambahan pembalik itu mengurangi banyaknya masukan dari dua menjadi satu. Disamping itu karena keluaran Q

tidak akan menerima masukan D sampai datangnya suatu pulsa waktu, bentuk itu sering juga disebut sebagai flip-flop tertunda (delay flip-flop).



Gambar a

Diagram logika



Gambar b dan c

Lambang dan persamaan karakteristik

Lambang untuk flip-flop D menurut waktu itu diberikan pada gambar b. Seperti halnya dengan setiap flip-flop menurut waktu, flip-flop D juga dilengkapi dengan masukan SET dan CLR. Persamaan karakteristiknya diturunkan dengan peta karnaugh di (c) dan tabel karakteristik flip-flop D itu diberikan oleh tabel 6.2. Persamaan karakteristik itu membuktikan bahwa keadaan berikutnya pada flip-flop tersebut sama seperti masukan D dan tidak tergantung pada nilai keadaan sekarang.

Tabel 1

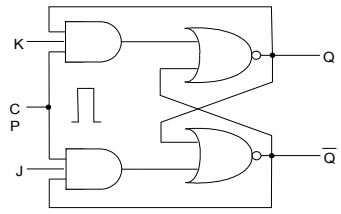
Tabel karakteristik flip-flop D

Q	D	Q(t + 1)
0	0	0
0	1	1
1	0	0
1	1	1

D. Flip-flop JK

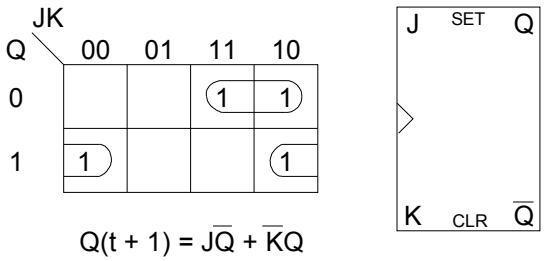
Flip-flop ini merupakan perbaikan dari flip-flop RS sehingga keadaan tak tentu pada jenis RS menjadi terdefinisi untuk jenis JK tersebut. Masukan J dan K berlaku seperti masukan R dan S (perhatikan bahwa untuk suatu flip-flop JK, huruf J adalah utnuk set dan huruf K untuk bebas). Bila masukan J dan K diberikan secara serentak, nilai flip-flop itu berubah menjadi komplemennya, yaitu jika mula-mula Q = 1, akan berubah menjadi Q = 0 dan sebaliknya.

Suatu flip-flop JK menurut waktu ditunjukkan pada gambar a. Keluaran diAND-kan dengan masukan K dan CP sehingga flip-flop itu dibebaskan selama suatu pulsa waktu hanya jika Q sebelumnya sama dengan 1.



Gambar a

Diagram logika



Gambar b dan c

Lambang dan persamaan karakteristik

Demikian pula keluaran Q flip-flop tersebut diAND-kan dengan masukan J dan CP sehingga flip-flop itu dapat diset dengan pulsa waktu hanya jika Q sebelumnya sama dengan 1. bila baik J maupun K sama dengan 1, keadaan Q akan selalu berubah tanpa memandang bagaimana keadaan Q tersebut sebelum pulsa waktu diberikan. Jadi jika Q sama dengan 1, keluaran gerbang AND yang diatas menjadi 1 dan flip-flop itu dibebaskan. Tampak bahwa jika sinyal CP itu tetap 1 setelah keluarannya dikomplemenkan, flip-flop itu akan berubah menjadi suatu keadaan yang baru.

Lambang dan persamaan karakteristik flip-flop JK itu diberikan pada gambar b dan c. Tabel karakteristik flip-flop itu diberikan oleh tabel 1.

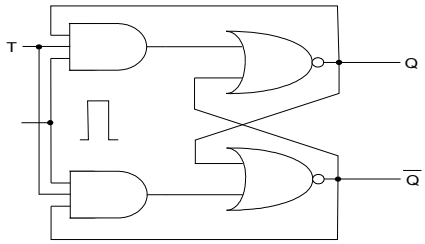
Tabel 1

Tabel karakteristik flip-flop JK

Q	J	K	Q(t + 1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

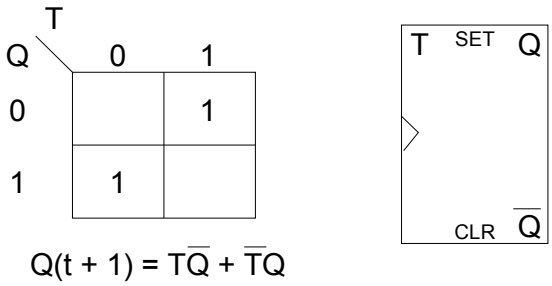
E. Flip-flop T

Flip-flop ini adalah flip-flop JK dengan masukan tunggal. Seperti yang tampak pada gambar a, flip-flop T itu didapatkan dari jenis JK jika kedua masukannya dijadikan satu. Nama 'T' (*toggle*-artinya saklar pengalih dua keadaan) itu diperoleh karena kemampuan flip-flop itu untuk mengubah keadaannya. Apapun keadaan sekarang flip-flop T itu akan berubah menjadi komplemennya setiap kali pulsa waktu diberikan pada saat masukan T itu bernilai 1.Lambang dan persamaan karakteristik flip-flop T itu ditunjukkan pada gambar b dan c. Tabel karakteristiknya diberikan oleh tabel 1. Keempat jenis flip-flop yang diperkenalkan diatas dapat tersedia dalam keadaan tanpa pengaturan waktu. Flip-flop tanpa masukan waktu tersebut berguna untuk operasi tak-serempak. Keempat jenis itu merupakan jenis yang umum dijumpai dalam rangkaian digital dan tersedia di pasaran.



Gambar a

Diagram logika



Gambar b dan c

Lambang dan karakteristik

Tabel 1

Tabel karakteristik flip-flop T

Q	T	Q(t + 1)
0	0	0
0	1	1
1	0	1
1	1	0

BAB X

REGISTER

Register merupakan blok logika yang sangat penting dalam kebanyakan system digital. Register sering digunakan untuk menyimpan (sementara) informasi biner yang muncul pada keluaran sebuah matriks pengkodean. Di samping itu, register sering digunakan untuk menyimpan (sementara) data biner yang sedang didekode. Maka register membentuk suatu kaitan yang sangat penting antara system digital utama dan kanal-kanal masukan/keluaran.

Register biner juga membentuk basis beberapa operasi aritmatika yang sangat penting. Sebagai contoh, operasi-operasi komplementasi, perkalian dan pembagian seringkali diwujudkan dengan menggunakan register.

Register geser dengan sangat mudah dapat dimodifikasi untuk membentuk berbagai jenis pecacah. Pecahan-pecahan ini memberikan beberapa keuntungan yang sangat berbeda.

Register Geser seri (Serial Shift Register)

Register tidak lebih daripada sekelompok flip-flop yang dapat digunakan untuk menyimpan sebuah bilangan biner. Harus terdapat sebuah flip-flop bagi masing-masing bit dalam bilangan biner tersebut. Tentunya flip-flop harus dihubungkan sedemikian hingga bilangan biner dapat dimasukan keluar dan kedalam register. Sekelompok flip-flop yang dihubungkan untuk melaksanakan salah satu atau kedua fungsi ini disebut register geser (shift regester).

Shift Register adalah suatu register yang mempunyai kemampuan untuk menggeser data 1 bit ke kiri atau ke kanan setiap kali mendapat satu pulsa clock. Secara umum terdapat 3 jenis shift register, yaitu :

1. Shift-Left Register, yaitu suatu register yang dapat menggeser data 1 bit ke kiri setiap kali mendapat pulsa satu clock.
2. Shift-Right Register, yaitu suatu register yang dapat menggeser data 1 bit ke kanan setiap kali mendapat pulsa satu clock.

3. Shift-Left/Right Register, yaitu suatu register yang dapat menggeser data 1 bit ke kiri atau ke kanan setiap kali mendapat pulsa satu clock; tergantung kepada level logic yang diberikan pada “Mode Input” dari register tersebut.

Ditinjau dari cara pemasukan dan pengeluaran data, terdapat 4 jenis shift register, yaitu :

1. Shift Register SISO (serial in serial out), yaitu shift register yang dapat menerima dan mengeluarkan data secara seri. Untuk memasukkan dan mengeluarkan data secara seri diperlukan sebanyak n pulsa clock.
2. Shift Register SIPO (serial in paralel out), yaitu shift register yang dapat menerima data secara seri dan mengeluarkan data secara paralel.
3. Shift Register PISO (paralel in serial out), yaitu shift register yang dapat menerima data secara paralel dan mengeluarkan data secara seri.
4. Shift Register PIPO (paralel in paralel out), yaitu shift register yang dapat menerima dan mengeluarkan data secara paralel.

Terdapat dua metode untuk menggeser informasi biner kedalam suatu register. Yang pertama berkenaan dengan pergeseran informasi kedalam register bit demi bit secara seri (berderet) dan metode ini mengarah kepada pengembangan register geser seri (serial shift register). Metode yang kedua berkenaan dengan penggeseran semua bit ke dalam register pada saat yang sama dan mengarah kepada pengembangan register geser paralel (paralel shift register). Register geser dibahas dalam bagian ini, dan register paralel dibahas dalam bagian selanjutnya.

DAFTAR PUSTAKA

Catatan kuliah Digital. 2006.

Jobsheet Digital.2006.

Mismail, Budiono. 1998. *Dasar-dasar Rangkaian Logika Digital*. Bandung : ITB.

S, Wasito dan B.Hernawan.1986. *Tehnik Digit.* Jakarta Selatan : Karya Utama.

Sulaeman, Entis. 2003. *Rangkaian Logika & Digit*. Bandung : Politeknik TEDC.