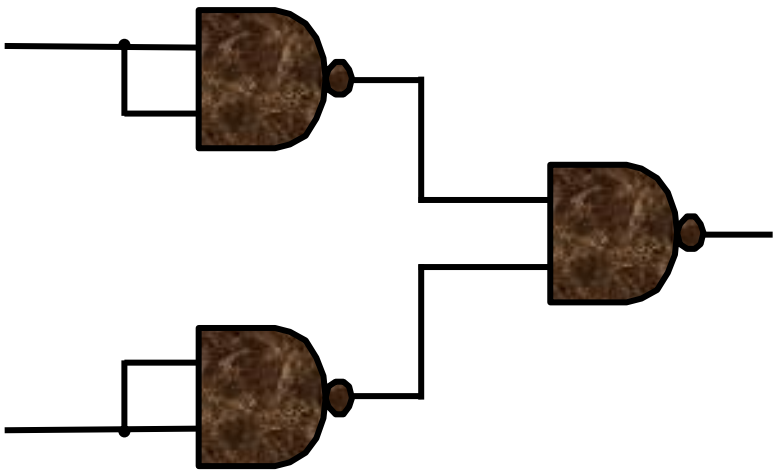
PENGANTAR SISTEM DIGITAL



Dr. Wawan Setiawan, M.Kom.

Bab 1
Pendahuluan

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

1.1 Sistem Analog dan Digital

Dalam sain dan teknilogi, demikian pula berbagai bidang kehidupan yang lainnya, kita selalu dihadapkan pada yang namanya besaran yaitu sesuatu yang dapat kita amati, kita ukur dan analisis. Dalam pekerjaan seperti itu kita membutuhkan sejumlah peralatan. Harapan kita adalah dapat menyajikan harga suatu besaran dengan cepat dan tepat. Pada dasarnya ada dua cara menyajikan harga numerik suatu besara yaitu secara analog dan digital.

Penyajian secara analog dapat kita ambil contoh spedometer kendaraan dimana penyimpangan jarum menunjukkan harga tertentu dan mengikuti laju kendaraan yang ber-sangkutan. Dalam kasus tersebut besaran laju kendaraan dianalogikan dengan penyimpangan jarum spedometer. Masih banyak contoh lainnya silahkan anda mengamati sendiri dan mendiskusikan dengan teman-teman anda. Ciri khas penyajian secara analog dapat berada pada

sebarang nilai tidak ada nilai terlarang kecuali di luar batas kemampuan.

Penyajian secara digital, dapat kita ambil contoh yang banyak ditemukan seperti jam digital. Waktu berubah secara kontinyu namun jam digital tidak dapat menunjukkan secara kontinyu. Penampilan waktu hanya dapat berubah pada tingkat paling kecil (menit atau detik). Dengan demikian penampilan waktu tersebut berubah secara diskrit. Ciri khas penyajian besaran secara digital adalah hanya berada pada nilai-nilai tertentu yang diskrit.

Piranti elektronika sekarang ini menuju pada keadaan otomatisasi, minimisasi, dan digitasi. Otomatisasi menjadikan segala pekerjaan dapat dilakukan secara mudah dan akurat seolah dapat diselesaikan dengan sendirinya. Minimisasi menjadikan piranti elektronika menjadi semakin kecil dan kompak, tidak membutuhkan ruang yang besar tetapi kinerjanya sangat handal. Digitasi menjadikan pengolahan data semakinmenguntungkan dengan beberapa kelebihan antara lain :

a. Lebih tegas karena data ditampilkan dalam dua keadaan YA atau TIDAK, MATI atau HIDUP, 1 atau 0, 0 volt atau 5 volt dan sebagainya.
b. Mudah dikelola seperti disimpan dalam bentuk memori, mudah ditransmisikan, mudah dimunculkan kembali, mudah diolah tanpa penurunan kualitas.
c. Lebih tahan terhadap gangguan atau lebih sedikit terkena gangguan.
d. Kebutuhan dayanya yang rendah.

Dalam prakteknya elektronika analog dan elektronika digital dapat saling melengkapi karena masing-masing memiliki keungulan dan kelemahan tersendiri.

Bab 2

Sistem Bilangan

Banyak system bilangan yang digunakan pada piranti digital, dan yang biasa digunakan ialah system-sistem bilangan biner, oktal, decimal, dan heksa decimal. Sedangkan, dalam kehidupan sehari-hari sangat akrab dengan system bilangan decimal,(dasaan, basis-10, atau radiks-10). Meskipun system decimal sangat akrab dengan kita, tetapi system tersebut tidak mudah diterapkan didalam mesin digital. System bilangan yang paling mudah diterapkan didalam mesin digital adalah system bilangan biner (basis-2) karena system tersebut hanya memiliki 2 simbol angka yang sesuai dengan 2 keadaan yang berbeda didalam mesin. Untuk memudahkan pembahasan, kita membagi system bilangan menjadi basis-10 dan basis ke-n, dimana n > 2. Sehingga dikenal banyak system seperti basis –2, basis-3, …, basis-8, …,basis-10, …, basis-16, dan seterusnya. Semua system bilangan tersebut termasuk kedalam system bilangan berbobot, artinya nilai sustu angka tergantung dari posisi relatifnyaterhadap koma atau angka satuan. Misalnya bilangan 5725,5 dalam decimal. Ketiga angka 5 memiliki nilai yang berbeda, angka 5 paling kanan bernilai lima persepuluhan, angka 5 yang tengah bernilai lima satuan sedangkan angka 5 yang lainnya bernilai lima ribuan.

Untuk membedakan suatu bilangan dalam system bilangan tertentu digunakan konvensi notasi. Untuk basis-n kita menggunakan indeks n atau tanda lain yang disepakati. Sebagai contoh bilangan ‘11’ basis-2 akan ditulis dalam bentuk ‘11_2’ untuk mencegah terjadinya salah pengertian dengan bilangan ‘11_8’, ‘11_{10}’ atau ‘11_{16}’ dan seterusnya. Kadang-kadang indeks tersebut tidak dicantumkan jika basis tersebut sudah jelas. Misalkan secara khusus sedang membahas bilangan basis-8, maka bilangan-bilangan dalam pembahasan tersebut tidak disertai indeks. Sering pula

dalam konvensi tersebut dijumpai bahwa suatu bilangan yang tidak disertai indeks berarti bilangan tesebut dinyatakan dalam decimal atau basis-10. selanjutnya dikenal beberapa cara menyatakan suatu bilangan dalam basis-16, atau heksa-desimal. Cara-cara tersebut adalah dengan menyertakan indeks 16, atau di belakang bilangan diikuti dengan huruf 'h', atau sebelum bilangan itu dicantumkan huruf 'h' atau tanda '#'atau tanda'$'. Contoh 96_{16} = 96h = H16 = # 96 = $96.

2.1 basis-10 (desimal)

Dalam system decimal (basis-10) mempunyai symbol angka (numeric) sebanyak 10 buah symbol, yaitu 0, 1, 2, 3, 4, 5, 6, 7, 8, dan 9. nilai suatu bilangan dalam basis-10 dapat dinyatakan sebagai $\sum(N \times 10^a)$ dengan N = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 dan a = …, -3,-2, -1, 0, 1, 2, 3, …(bilangan bulat yang menyatakan posisi relatif N terhadap koma atau satuan).

Contoh :

$325_{10} = 3 \times 10^2 + 2 \times 10^1 + 5 \times 10^0$

$0,61_{10} = 0 \times 10^0 + 6 \times 10^{-1} + 11 \times 10^{-2} + 6 \times 10^1 + 1 \times 10^{-2}$

$9407,108_{10} = 9 \times 10^3 + 4 \times 10^2 + 7 \times 10^0 + 1 \times 10^{-1} + 8 \times 10^{-3}$

2.2 Basis-2 (biner)

Dalam system biner (basis-2) mempuyai symbol angka (numeric) sebanyak 2 buah symbol, yaitu 0 dan 1. nilai suatu bilangan basis-2 dalam basis-10 dapat dinyatakan sebagai $\sum(N \times 2^a)$ dengan N = 0 atau 1; dan a = …, -3, -2, -1, 0, 1, 2, 3, …(bilangan bulat dalam decimal yang menyatakan posisi relatif N terhadap koma atau satuan).

Contoh :

1101_2 $= 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^0$
$= 8 + 4 + 1$
$= 13_{10}$
$0,101$ $= 0 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3}$
$= 0 + 0,5 + 0 + 0,125$
$= 0,62510$

$11,01$ $= 1 \times 21 + 1 \times 20 + 1 \times 2\text{-}2$
$= 2 + 1 + 0,25$
$= 3,25_{10}$

2.3 Basis-8 (octal)

Dalam system octal (basis-8) mempunyai symbol angka (numerik) sebanyak 8 buah symbol, yaitu 0, 1, 2, 3, 4, 5, 6, dan 7 . nilai suatu bilangan basis-8 dalam basis 10 dapat dinyatakan sebagai $\sum(N \times 8^a)$ dengan N = 0, 1, 2,3, 4, 5, 6, dan 7; dan a = …,-3, -2, -1, 0, 1, 2, 3, …(bilangan bulat dalam decimal yang menyatakan posisi relatif N terhadap koma atau satuan).

Contoh :
$647,35_8 = 6 \times 8^2 + 4 \times 8^1 + 7 \times 8^0 + 3 \times 8^{-1} + 5 \times 8^{-2}$
$= 384 + 32 + 7 + 0,375 + 0,078125$
$= 423,453125_{10}$

2.4 Sis-16 (heksa-desimal)

Dalam system heksa-desimal (basis-16) mempunyai symbol angka (numerik) sebanyak 16 buah symbol. Karena angka yang telah dikenal ada 10 maka perlu diciptakan 6 buah symbol angka lagi yaitu A, B, C, D, E, dan F dengan nilai A16 = 1010, B16 = 1110, C16 = 1210, D16 =1310, E16 = 1410 dan F16 =1510. Dengan demikian symbol angka-angka untuk system heksa=decimal adalah 0, 1, 2,3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, dan F. nilai suatu bilangan basis-16

dalam basis-10 dapat dinyatakan sebagai $\sum(Nx\ 16^a)$ dengan N = 0, 1, 2,3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, dan 15; dan a = …, -3, -2, -1, 0, 1, 2, 3, …(bilangan bulat dalam decimal yang menyatakan posisi relatif N terhadap koma atau satuan).

Contoh :

$$584AED_{16} = 5 \times 16^5 + 8 \times 16^4 + 4 \times 16^3 + 10 \times 16^2 + 14 \times 16^1 + 13 \times 16^0$$
$$= 5242880 + 524288 + 16384 + 2560 + 224 + 13$$
$$= 14{,}06640625_{10}$$

2.5. Konversi (Pengubahan) Bilangan

Ada kalanya kita perlu menyatakan suatu bilangan dalam basis yang berbeda atau mengubah (mengkorversi) suatu bilangan dari satu basis yang satu ke basis yang lain. Misalkan, kenversi bilangan dari basis-n ke basis-10, kenversi bilangan dari basis-10 ke basis-n, atau konversi dari basis-n ke basis-m. untuk kenversi bilangan dari basis-n ke basis-10telah dikemukakan ketika menyatakan nilai suatu bilangan dari basis-n ke dalam basis-10.

2.6 Konversi bilangan basis-10 ke basis-n

Setidaknya ada 2 cara untuk mengubah bilangan decimal menjadi bilangan dalam basis selain 10. Cara pertama, biasanya untuk bilangan yang kecil, adalah kebalikan dari proses kenversi bilangan dari basis-n (selain 10) ke basis-10 (decimal). Bilangan decimal itu dinyatakan sebagai jumlah dari suku-suku yang setiap suku merupakan hasil kali suatu angka (N) dan bilangan pangkat bulat, kemudian angka-angka tersebutdituliskan dalam posisi yang sesuai. Secara umum dapat dituliskan sebagai :

$$(\text{Bilangan})_{10} = \sum(N \times n^a)$$

dengan N menyatakan symbol angka yang diijinkan dalam basis-n, n menyatakan basis bilangan yang dituju, a

merupakan bilangan bulat dalam basis-10 yang menyatakan positif relatif N terhadap koma atau satuan, dan semua posisi yang tercakup harus diperhitungka. Untuk lebih jelasnya perhatikan beberapa ilustrasi berikut ;

(1). Ubahlah bilangan 9810 ke dalam basis-2 yang setara !

$98_{10} = \sum(N \times n^a)$
$= \sum(N \times 2^a)$
$= N \times 64 + n \times 32 + N \times 2^1$
$= 1 \times 2^6 + 1x2^5 + 1x2^1$ (semua posisi belum diperhitungkan)
$= 1 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1x\ 2^1 + 0 \times 2^0$
$= 1100010$
$= 1100010_2$

Perhatikan bahwa 0 ditempatkan dalam posisi 2^4, 2^3, 2^2, dan 2^0 karena semua posisi harus diperhitungkan.

(2). Ubahkah bilangan 1368_{10} ke dalam basis-8 yang setara !

$136810 = \sum(N \times n^a)$
$= \sum(N \times 8^a)$
$= N \times 512\ N \times 64 + Nx8$
$= 2 \times 8^3 + 5 \times 8^2 + 3 \times 8^1$
(semua posisi belum diperhitungkan)
$= 2 \times 83 + 5 \times 82 + 3 \times 81 + 0 \times 80$
$= 2530$
$= 2530_8$.

Perhatikan bahwa 0 ditempatkan dalam posisi 8^0 karena semua posisi harus diperhitungkan.

(3). Ubahlah bilangan 19006_{10} ke dalam heksadesimal yang setara !

$$
\begin{aligned}
1900610 &= \sum(N \times n^a) \\
&= \sum(N \times 16^a) \\
&= N \times 4096 + N \times 256 + N \times 16 \\
&\quad + N \times 16^0 \\
&= 4 \times 16^3 + A \times 16^2 + 3 \times 16^1 \\
&\quad + 14 \times 16^0
\end{aligned}
$$

(semua posisi belum diperhitungkan)

$$
\begin{aligned}
&= 4A3E \\
&= 4A3E_{16}
\end{aligned}
$$

Cara kedua dikenal dengan **pembagian berulang.** Cara ini sangat baik untuk bilangan decimal yang kecil maupun yang besar. Cara konversi ialah membagi bilangan decimal dan hasil baginya secara berulang dengan basis tujuan kemudian menuliskan sisanya hingga diperoleh hasil bagi 0. Hasil kenversinya adalah menuliskan sisa pertama pada posisi yang paling kecil dan sisa terakhir pada posisi yang paling besar. Untuk lebih jelasnya perhatikan beberapa ilustrasi berikut :

(1). Ubahlah bilangan 98_{10} ke dalam basis-2 yang setara !

98 / 2 = 49, sisa 0
49 / 2 = 24, sisa 1
24 / 2 = 12, sisa 0
12 / 2 = 6, sisa 0
6 / 2 = 3, sisa 0
3 / 2 = 1, sisa 1
1 / 2 = 0, sisa 1

Sisa dituliskan dari bawah : 1 1 0 0 0 1 0

(2). Ubahlah bilangan 1368_{10} ke dalam basis-8 yang setara !

1368 / 8 = 171, sisa 0
171 / 8 = 21, sisa 3
21 / 8 = 2, sisa 5
2 / 8 = 0, sisa 2

Sisa dituliskan dari bawah : 2 5 3 0
Jadi $1368_{10} = 2530_{8}$.

(3). Ubahlah bilangan 19006_{10} ke dalam heksadesimal yang setara !

19006 = 1187, sisa 14 (=E)
6
1187 = 74, sisa 3
16
74 = 4, sisa 10 (=A)
16
4 = 0, sisa 4
16
Sisa dituliskan dari bawah :4 A 3 E
Jadi $19006_{10} = 4A3E_{16}$

Untuk mengubah bilangan decimal **tidak bulat** dilakukan dengan dua tahap. Tahap pertama mengubah bagian bulat (disebelah kiri tanda koma) dengan cara seperti yang telah dijelaskan diatas. Tahap kedua mengubah bagian pecahannya (disebelah kanan tanda koma) dengan cara bahwa **bilangan pecahan** dikalikan berulang-ulang dengan **basis tujuan** sampai hasil perkalian terakhir sama dengan 0 setelah angka disebelah kiri tanda koma dari hasil kali setiap perkalian diambil. Selanjutnya angka-angka disebelah kiri koma yang diambil tadi dituliskan secara berderet dari kiri kekanan. Untuk lebih jelasnya perhatikan ilustrasi pada konversi bilangan $98,375_{10}$ menjadi basis-2 yang setara ! tahap pertama mengubah bilangan bulat 98_{10} ke dalam basis-2 yang hasilnya adalah 1100010_2. tahap Kedua mengubah bilangan pecahan $0,375_{10}$ ke dalam basis-2 sebagai berikut :

0,375 x 2 = 0,75 dan angka disebelah kiri koma adalah 0
0,75 x 2 = 1,5 dan angka disebelah kiri koma adalah 1
0,5 x 2 = 1,0 dan angka disebelah kiri koma adalah 1.

Hasil pengambilan angka disebelah kiri koma adalah : 0,011. Selanjutnya hasil konversi kedua tahap tersebut digabungkan sesuai dengan posisinya. Hasil gabungannya adalah 1100010,011. Dengan demikian $98,375_{10}$ = $1100010,011_{2}$.

Contoh berikutnya adalah mengubah bilangan $1368,25_{10}$ kedalam basis-8 yang setara. Tahap pertama adalah mengubah bagian bulatnya (disebelah kiri koma) yaitu 1368_{10} ke dalam basis-8 yang hasilnya telah diperoleh sebesar 2530_{8}. tahap kedua adalah mengubah bagian pecahannya (disebelah kanan keoma) yaitu $0,25_{10}$ ke dalam basis-8 dengan cara sevagai berikut :

0,25 x 8 = 2,0 dan bilangan disebelah kiri koma adalah 2. Setelah 2 diambil maka sisanya adalah 0 dan proses perkalian berakhir.

Hasil pengambilan angka disebelah kiri koma adalah : 0,2. Selanjutnya hasil konvrsi kedua tahap tersebut digabung sesuai dengan posisinya. Hasil gabungannya ialah 2530,2. Dengan demikian $1368,25_{10}$ = $2530,2_{8}$.Perlu dicatat bahwa tidak semua pecahan mudah dikonversi. Ada kalanya hasil konversi bilangan yang tepat. Sebagaimana pecahan 2/3 yang dikonversikan ke dalam bentuk menghasilkan 0,666666…. Dimana angka 6 tidak akan pernah berakhir. Misalnya bilangan $34,275_{10}$ diubah ke dalam bilangan basis-8 yang setara. Bagian bulatnya menghasilkan $4 \times 8^{1} + 2x\ 8^{0}$ atau 42_{8}. Sedangkan bagian pecahannya dikonversi dengan vara berikut :

0,275 x 8 = 2,2 dan angka disebelah kiri koma adalah 2
0,2 x 8 = 1,6 dan angka disebelah kiri koma adalah 1
0,6 x 8 = 4,8 dan angka disebelah kiri koma adalah 4
0,8 x 8 = 6,4 dan angka disebelah kiri koma adalah 6
0,4 x 8 = 3,2 dan angka disebelah kiri koma adalah 3

0,2 x 8 = 1,6 dan angka disebelah kiri koma adalah 1 dan seterusnya.
Jadi $34,275_{10}$ = 42,214631463…….
Dimana angka 1463 tidak pernah berakhir.

2.7 Konversi bilangan dari basis-n ke basis-m (keduanya bukan basis-10)

Untuk mengkonversi suatu bilangan basis-n (bukan basis-10) menjadi bilangan basis-m (bukan basis-10) dengan n ≠ m diperlukan konversi ke basis-10 sebagai perantara. Kita telah akrab dengan bilangan basis-10. Dengan demikian perlu dua tahap konversi. Tahap pertama mengkonversi bilangan dari basis-n ke basis-10, dan tahap kedua mengkonversi bilangan hasil tahap pertama (dalam basis-10) menjadi basis-m. sebagai contoh ubahlah bilangan 237_8 menjadi bilangan yang setara dalam basis-5 !

Tahap 1 : $237_8 = 2 \times 8^2 + 3\times8^1 + 7\times8^0$
$= 128 + 24 + 7$
$= 159_{10}$.
Tahap 2 : $159_{10} = 1 \times 5^3 + 1 \times 5^2 + 1 \times 5^1 + 4 \times 5^0$
$= 1114_5$
Jadi $237_8 = 114_5$

Contoh berikutnya adalah mengubah bilangan $52DA_{16}$ ke dalam basis-12 yang setara.
Tahap 1 : $52DA_{16} = 5 \times 16^3 + 2 \times 16^2 + 13 \times 16^1 + 10 \times 16^0$
$= 21210_{10}$
Tahap 2 : 21210_{10}
$= 1\times12^4+0\times12^3+4\times12^2+3\times12^1+6\times12^0$
$=1043612$
Jadi $52DA_{16} = 10436_{12}$.

2.8 Operasi Bilangan

Dalam system decimal telah dikenal dengan baik mengenai operasi dasar bilangan, yakni penjumlahan, pengurangan, perkalian, dan pembagian. Operasi-operasi bilangan tersebut juga dapat dikenakan pada system bilangan yang lain seperti dalam system-sistem bilangan biner, basis-5, octal, heksa-desimal, dan seterusnya. Tetapi pembahasan operasi kali ini lebih banyak pada system biner, sedangkan untuk system bilangan yang lain akan dikemukakan contoh hanya apabila dipandang perlu. Prinsip-prinsip penggunaan operasi bilangan itu sama dengan yang diterapkan pada system decimal. Oleh karena belum akrab dengan system bilangan selain decimal, maka untuk memudahkan pelaksanaan operasi hitung perlu pertolongan table operasi.

Untuk system biner perhatikan table operasi berikut :

Tabel 2.1 : Penjumlah biner

+	0	1
0	0	1
1	1	10

Tabel 2.2 : Perkalian biner

x	0	1
0	0	0
1	0	1

Sebagaimana pada system decimal, bilangan biner dapat dijumlahkan, dikurangkan, dikalikan dan dibagi. Dimulai dari operasi penjumlahan pada bilangan biner. Penjumlahan antara dua bilangan biner dikerjakan dengan cara yang sama seperti pada penjumlahan bilangan decimal, bahkan penjumlahan pada bilangan biner lebih sederhana, persoalannya adalah orang tidak terbiasa dengan system biner. Berdasarkan pada table penjumlahan untuk bilangan buner hanya ada 4 (empat) hal yang dapat terjadi, yakni :

1) 0 + 0 = 0
2) 0 + 1 = 1 + 0 = 1
3) 1 + 1 = 10 = 0 + simpanan (carry) 1 untuk posisi berikutnya, dan
4) 1 + 1 + 1 = 11 = 1 + simpanan (carry) 1 untuk posisi berikutnya.

Hal yang ke empat terjadi dua bit pada posisi tertentu ialah 1 dan ada simpanan dari posisi sebelumnya.
Berikut beberapa contoh penjumlahan dua bilangan biner :

```
1  1  0  1  1   1  0  0  1    1  1 , 0  1  1
1  1  0  1  1   1  1  1  1    1  0 , 1  1  0
__________ +    _________+    ___________+
1  1  0  1  1 0 1  1  0  0  1 1  0 , 0  0 1
```

Didalam mesin digital penjumlahan antara lebih dari dua bilangan biner pada saat yang sama tidak terjadi, karena rengkaian digital yang melaksanakan penjumlahan hanya dapat menangani dua bilangan pada saat yang bersamaan. Jika lebih dari dua bilangan yang ditambahkan, maka bilangan pertama dan ke dua dijumlahkan lebih adahulu dan hasil penjumlahan itu baru ditambahkan pada bilangan ke tiga, dan seterusnya. Penjumlahan biner merupakan operasi aritmetika yang paling penting dalam system digital, karena operasi-operasi pengurangan, perkalian, dan

pembagian dapat dikerjakan dengan hanya dengan prinsip penjumlahan. Misalnya pengurangan dapat dibentuk dari penjumlahan dengan bilangan negatif. Perkalian tidak lain merupakan penjumlahan yang berulang. Sedangkan pembagian adalah pengurangan yang berulang. Selanjutnya dikemukakan dahulu contoh operasi lain bilangan biner dengan cara aljabar.

Pengurangan :

```
    1  0 pinjaman (borrow)
   1  1  0   1  1   bilangan yang dikurangi
   1  0  1   1  0   bilangan pengurang
   _____________-
   0  0  1   0  1   hasil pengurangan
```

perkalian :

```
      1  1  0  1    bilangan yang dikalikan
         1  1  0   bilangan pengali
      _________x
        0  0  0  0
        1  1  1  0
        1  1  0  1
        ______________+
        1  0  0  1  1  1  0
```

hasil operasi perkalian

```
pembagian :
1   0   1   1    (hasil bagi)
1   0   1        (pembagi)

0 1 1 1          (bilangan yang dibagi)
1 0    1
_______________-
1   1   1
1   0   1
_______________-
1   0   1
1   0   1
___________-
        0
```

Untuk pembanding, berikutnya dikemukakan operasi bilangan basis-5 dengan pertolongan table operasi berikut :

Tabel 2.3 : Penjumlahan basis-5

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	10
2	2	3	4	10	11
3	3	4	10	11	12
4	4	10	11	12	13

Tabel 2.4 : perkalian basis-5

X	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	2	11	13
3	0	3	11	14	22
4	0	4	13	22	31

Penjumlahan :

```
      1   1   1          simpanan (carry)
      4   3   3   2₅     bilangan pertama
      3   3   4   4₅     bilangan ke dua
    ____________________+
   1   3   2   3   1₅    hasil penjumlahan
```

Pengurangan :

```
  10 10           pinjaman (borrow)
  4   3   0   2₅   bilangan yang dikurangi
  1   4   3   1₅   bilangan pengurang
  ____________________-
  2   3   2   1₅   hasil pengurangan
```

Perkalian :

```
  3    4₅         bilangan yang dikalikan
  4    2₅         bilangan pengali
  ___________x
  1    2    3
  3    0    1
  ________________+
  3    1     3    3₅ hasil perkalian
```

Pembagian :

```
  1   0   4   2₅       (hasil bagi)(pembagi)
  4₅ 4   3   2   3₅      (yang dibagi)
  4 _____________-
     0   3   2
         3   1
       _______-
     0   1   3
     0   1   3
     ________-
           0
```

Operasi bilangan pada basis yang lain prinsipnya sama. Untuk basis lebih dari 10 kemungkinan terasa sulit oleh karena belum terbiasa.

Sebagaimana diketahui bahwa mesin digital, seperti computer dan kalkulatir, hanya dapat mengolah data yang sifatnya biner. Lagi pula, mesin digital tersebut hanya mengenal operasi penjumlahan dan tidak mengenal operasi pengurangan. Sedangkan mesin digital megolah bilangan negatif sama baiknya dengan mengolah bilangan positif. Oleh karena itu, operasi pengurangan harus bisa disajikan dalam bentuk penjumlahan. Selain itu, untuk keperluan penyimpanan dan pembacaan bilangan diperlukan kejelasan tentang tanda dari suatu bilangan merupakan bilangan positif atau negatif. Seperti yang telah dikenal bilangan positif diberi tanda '+' atau tanpa tanda didepan bilangan yang bersangkutan, sedangkan bilangan negatif dengan tanda '-'. Untuk ini diperlukan tanda bilangan. Hal ini biasanya dikerjakan dengan menambahkan bit lain dedalam suatu bilangan. Bit tambahan itu disebut sebagai bit-tanda (sign bit). Dalam system biner tanda '+' dan "-" digantikan dengan '0' dan '1'. Pada umumnya bit tanda itu berupa tanda '0' untuk menyatakan sustu bilangan positif, dan '1' untuk bilangan negatif. Sebagai contoh :

+1011 atau 1011 dituliskan sebagai 01011
-1011 dituliskan sebagai 11101.

Untuk menyatakan bilangan negatif dalam mesin digital da[at digunakan beberapametode. Dua metode yang paling dikenal ialah metode komplemen 1 dan metode komplemen 2.

Bentuk komplemen 1 dari suatu bilangan biner diperoleh secara sederhana dengan mengubah setiap 0 dalam bilangan itu menjadi 1 dan setiap 1 didalam bilangan itu menjadi 0. Dengan kata lain mengubah setiap bit menjadi komplemennya. Sebagai contoh komplemen 1 dari bilangan 011010 adalah 100101. Ketika bilangan negatif disajikan

dalam komplemen 1, maka bit tandanya adalah 1 dan besar bilangannya dikonversi menjadi bentuk komplemen 1.

Misalnya bilangan -57_{10} dapat disajikan menjadi biner negatif sebagai:

-57_{10} = 1 111001 (besar bilangan dalam biner)
= 1 000110 (bentuk komplemen 1)

Perhatikan bahwa bit tanda tidak ikut dikomplemenkan tetapi dipertahankan sebagai 1 untuk menunjukan bahwa bilangan itu negatif. Contoh lain bilangan negatif yang disajikan dalam bentuk komplemen 1 adalah :

-14_{10} = 10001
-326_{10} =1010111001
-7_{10} =100

Agar lebih memahami metode komplemen 1 ini akan dilakukan operasi pengurangan pada bilangan biner 1101 – 1011. Pada operasi tersebut dapat ditambahkan 1111 asalkan diingat bahwa hasilnya nanti kelebihan 1111. Kelebihan dihilangkan agar diperoleh hasil yang sebenarnya.

Selanjutnya perhatikan :

1101 + (1111 – 1011) = 1101 + **0100** = 1 0001
Bilangan 0100 pada operasi itu merupakan komplemen 1 dari bilangan -1011. Kelebihan 1111 dihilangkan dengan cara menambahkan EAC (End Around Carry = simpanan memutar) kepada LSD (Least Significant Digit = digit bobot terkecil). Dengan demikian hasil yang sebenarnya adalah :

0001 + 1 (EAC) = 0010.

Bila operasi dilakukan dengan bit tanda, maka yang diubah ke dalam bentuk komplemen 1 adalah bagian besar bilangan saja (tidak termasuk bit tanda). Untuk contoh diatas :

```
0 101   (bilangan positif yang dikurangi)
1  0100 (1 dari 1011 yang negatif)
______-
1 0  0001

  _______-    (EAC)
  0  0010   (hasil +0010 karena bit tanda 0)
```

contoh tersebut merupakan salah satu kemungkinan dari pengurangan A – B = C dengan A, B > 0 dan A < B, sehingga C < 0 (negatif). Perhatikan contoh berikut :

```
  1001 –1110 = 0  1001 + 1  1110


      0  1001 (yang dikurangi)
      1  0001 (komplemen 1 dari pengurang)
      _______+
      1 1010
           0  (EAC)
       ______+
        11010
```

(bukan hasil dengan negatif,bit tanda 1).

Bila hasilnya negatif,maka hasil tersebut bukan hasil yang sebenarnya. Hasil yang sesungguhnya adalah komplemen 1 dari besar bilangan hasil tersebut. Jadi, hasil yang sesungguhnya adalah 1 0101 = -0101.
Kemungkinan berikutnya adalah A < 0 dan B > 0, sehingga C < 0. Dalam hal ini harus diperhatikan bahwa banyaknya digit hasil operasi tidak boleh melebihi digit soal.

Sehingga demi amannya harus ditambahkan satu digit yakni 0 sebagai MSD (Mosat Significant Digit = digit paling berbobot). Tambahan digit ini tidak akan mengubah besar bilangan yang bersangkutan. Sebagai contoh akan dicoba operasi pada –1101 – 1011 sebagai berikut :
Demi amannya soal diubahnya menjadi -01101 – 01011
Dengan bit tanda soal menjadi
1 01101 +1 01011

110010 (komplemen 1 dari yang dikurangi)
110100 (komplemen 1 dari pengurang)
_____+
11 00110
1 (EAC)
_____+
1 00111
(hasilnya masih negatif, belm hasil yang sebenarnya)

hasil yang sebenarnya adalah komplemen 1 dari bilangan 1 00111 yaitu 1 11000 (ingat bit tidak turut dikomplemenkab) atau -11000.

Selanjutnya metode komplemen 2. Bentuk komplemen 2 dari suatu bilangan biner ditentukan dengan cara mengambil komplemen 1 dari dari bilangan itu kemudian menambahkan 1 pada posisi LSB (Least Significant Bit = bit paling tudak berbobot). Sebagai ilustrasi akan diubah bilangan bimer 111001 ($=57_{10}$) ke dalam bentuk komplemen 2 –nya.

```
1 1 1 0 0 1   (bilangan awal)
  0 0 0 1 1 0   (bentuk komplemen 1)
              1   (menambahkan 1 pada LSB)
   ___________+
     0 0 0 1 1 1
```

Jadi, -57_{10} yang disajikan dalam bentuk komplemen 2 dituliskan sebagai 1 000111. Ingat bahwa bit paling kiri merupakan bit tanda dan 6 bit yang lain merupakan bertuk awal komplemenn 2 dari besar bilangan awalnya. Operasi pengurangan dengan metode komplemen2 juga memiliki 3 kemungkinan yaitu
A – B = C dengan C > 0, A- B = C dengan A < B, dan A – b = C dengan A < 0, B > 0. Untuk setiap kemungkinan tersebut, perhatikan contoh-contoh berikut !

```
  1)   1110 – 0101 = 0  1110 + 1  0101]
               0  1110    (bilangan positif
                          yang dikurangi)
               1  1011    komplemen 2 dari
                          pengurangannya)
              ________+
             10  1001    (EAC,  bit  tanda  0,dan  hasil
    sesungguhnya)
```

hasil yang sebenarnya diperoleh dengan menghilangkan EAC-nya, jadi hasil yang sebenarnya adalah 0 1001 atau +1001 atau 1001

2) 10011 – 11001 = 0 10011 + 1 11001

```
  0 10011  (bilangan positif
           yang dikurangi)
  1 00111  (komplemen 2 dari
            pengurangannya)
  _______+
  01 11010 (EAC, bit tanda 1, dan hasil
```

belum sesungguhnya).

Karena hasil masih negatif, maka hasil yang sebenarnya adalah komplemen 2 dari 11010 yaitu 00110. jadi hasil pengurangannya adalah 1 001110 atau –00110.

3) -011011-011101=1011011+1 011101

```
1 100101  (komplemen 2 dari yang dikurangi)
1 100011  (komplemen 2 dari pengurangnya)
_________+
11 001000 (EAC,bit tanda 1,dan hasil belum sesungguhnya)
```

Karena hasilnya negatif (dengan bit tanda 1), maka setelah menghitung EAC, kemudian mengambil komplemen 2 dari 001000 dan diperoleh 111000. Jadi hasil sebenarnya dari pengurangan tersebut adalah 1 111000 atau –111000.

Diantara kedua metoda yang dikemukakan diata, maka metode komplemen 2 paling banyak digunakan. Karena dengan metode komplemen 2 memungkinkan untuk membentuk operasi pengurangan hanya menggunakan operasi penjumlahan yang

sesungguhnya. Ini berarti bahwa mesin digital dapat menggunakan ranglaian yang sama untuk melaksanakan kedua operasi tersebut. Dengan demikian mendapat penghematan dalam perangkat kerasnya.

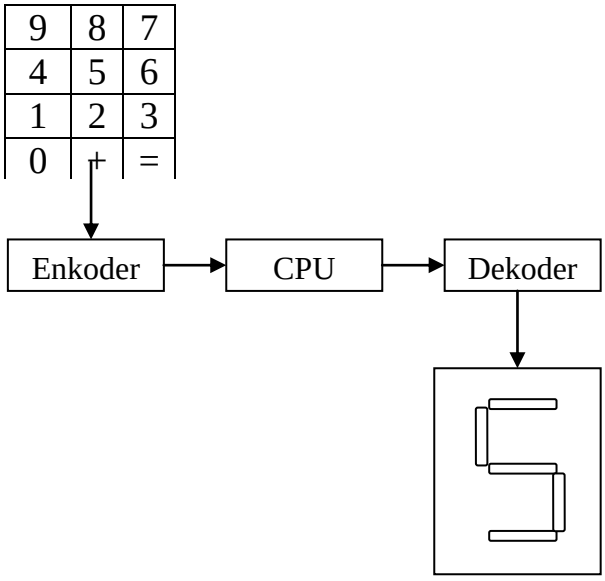
Pengubahan dari bentuk komplemen ke nilai biner yang sesungguhnya asngat sederhana. Dari bentuk komplemen 1 ke biner susungguhnya ditempuh dengan cara meng-komplemen-kan setiap bit lagi. Sedangkan dari bentuk komplemen 2 ke nilai biner yang sesungguhnya dilakukan dengan meng-komplemen-kansetiap bit dan kemudian menamnahkan 1 pada LSB. Dalam 2 hal itu, pengubahan kembali ke bentuk biner yang sesungguhnya ditempuh melalui proses yang sama yang telah ditempuh untuk menghasilkan bentuk komplemennya.

Bab 3
Sistem Sandi

Pada mesin digital, baik instruksi (perintah) maupun informasi (data) diolah dalam bentuk biner. Karena mesin digital hanya dapat 'memahami' data dalam bentuk biner. Kita sering menggunakan mesin-mesin digital seperti jam digital, multimerter digital, thermometer digital, kalkulator, computer, dan sebagainya. Tampilan yang langsung dapat dilihat berupa angka decimal atau kumpulan huruf latin yang dikenal dalam keseharian, padahal proses yang terjadi didalam mesin-mesin tersebut berbentuk biner. Sedangkan instruksi maupun informasi dalam bentuk biner tidak disukai karena diluar kebiasaan sehingga terasa sangat rumit dan kurang praktis. Kita telah terbiasa dengan huruf latin dari A sampai Z dan angka-angka dari 0, 1, 2, …, sampai 9. Sehingga apabila disajikan bilangan atau kata dalam bentuk biner tidak segera dapat diketahui maknanya. Misalnya pada sederet bit biner 00010111, kita tidak segera tahu bahwa deretan bit biner itu menyatakan bilangan atau huruf. tahu bahwa deretan bit itu menyatakan bilangan atau huruf. Jika bilangan, dereten bit tersebut dapat menunjukan bilangan 17_{16} atau 23_{10}. Agar deretan bit 00010111 dapat tampil sebagai bilangan 17_{16} atau 23_{10} diperlukan teknik atau rangkaian tertentu. Sebaliknya, agar 17_{16} atau 23_{10} dapat dikenali oleh suatu mesin digital sebagai 00010 111 juga diperlukan teknik atau rangkaian tertentu.

Dalam pemakaian kakulator, bilangan yang dimasukan melalui tombol kunci (tuts) perlu diubah dari bentuk decimal menjadi biner. Sebaliknya bilangan yang muncul pada tampilan kalkulator mengalami proses pengubahan dari bentuk biner ke dalam format 7-segmen yang umumnya berbentuk decimal. Perhatikan ilustrasi pengubahan tampilan kalkulator pada gambar 3.1. Kita akan memasukan

bilangan decimal 5 dengan cara menekan tombol kunci 5. Rangkaian encoder (penyandi) mengubah decimal 5 menjadi biner 0101. Unit pengolah pada kalkulator (CPU : Central Processing Unit) menerima bilangan itu dalam bentuk biner 0101 karena CPU hanya dapat mengolah data dalam bentuk biner. Selanjutnya rangkaian decoder (pembaca sandi) mengubah bilangan biner 0101 kembali menjadi bentuk decimal 5. Akhirnya yang muncul dalam tampilan adalah decimal 5 seperti semula. Dari ilustrasi tersebut memperlihatkan terjadinya proses pengubahan dari satu jenis sandi (kode) system bilangan menjadi jenis sandi system bilangan yang lain. Awalnya dari sandi decimal menjadi biner, dan akhirnya dari sandi bener menjadi sandi decimal. Suatau rangkaian pengubah suatu pesan bermakna (misal decimal) menjadi satu sandi tertentu (missal biner) disebut **encoder** (penyandi). Sedangkan, sebaliknya, rangkaian pengubah suatu sandi tertentu kembali menjadi makna yang sebenarnya disebut **dekoder** (pembaca sandi).



Gambar 3.1:Pengubahan tampilan kalkulator.

3.1 Sandi BCD (Biner Code Decimal)

Kita telah terbiasa dan akrap dengan system bilangan decimal dan karenanya system ini dianggap sebagai sandi yang paling bermakna. Dalam design digital biasa menampilkan bilangan dalam bentuk decimal. Sedangkan sedangkan proses komputasi dalam mesin bentuk digital dalam bentuk biner. Jika hasil komputasi tetap ditampilkan dalam bentuk biner, kita mengalmi hambatan atau bahkan sulit memahaminya, karena kita tidak terbiasa dengan bilangan yang tampil dalam bentuk biner. Jadi tampilan decimal lebih mudah dipahami daripada tampilan biner. Oleh karena itu diperlukan suatu cara penyandian dari biner ke decimal dan sebaliknya.

Sebagai contoh bilangan decimal 25 dan 43 masing-masing disandikan sebagai berikut :

$25_{10} = 11001_2$

$43_{10} = 101011_2$

Kita lihat bahwa sembarang bilangan decimal dapat disajikan dalam bentuk biner yang setara. Sekelompok 0 dan 1 dalam bentuk biner dapat dipikirkan sebagai penggambaran sandi suatu bilangan decimal. Dua contoh diatas memperlihatkan bahwa setiap angka biner mempunyai nilai sesuai dengan posisinya (satuan, duaan, empatan, dan seterusnya). Dalam contoh diatas semua digit bilangan decimal disandikan langsung, atau sebaliknya semua pernyataan biner menyandikan suatu bilangan decimal, jadi bukan digit perdigit yang disandikan. Dalam bentuk jenis lain bilangan-bilangan 0, 1, 2, 3, 4, 5, 6, 7, 8, dan 9 disandikan sendiri-sendiri. Dengan demikian untuk menyatakan bilangan decimal lebih dari satu digit,maka setiap digitnya disandikan sendiri. Salah satu system sandi yang cukup terkenal adalah BCD atau decimal yang disandikan biner. Karena digit decimal yang terbesar adalah 9, maka diperlukan 4 bit biner untuk menyandi setiap digit. Susunan 4 bit biner tersebut menghasilkan 6 kombinasi yang berbeda, tetapi hanya diperlukan 10

kombinasi diantaranya. Untuk menyatakan bilangan decimal N digit diperlukan N x 4 bit biner. Untuk bilangan bulat, kelompok 4 bit ke 2 adalah puluhan, kelompok 4 bit ketiga merupakan ratusan, dan seterusnya. Sebagai contoh bilangan decimal 468 (terdiri dari 3 digit)memerlukan 3 kelompok masing-masing 4bit seperti tampak pada table 3.3 berikut :

Table 3.1:Desimal 468 disajikan dengan BCD

Desimal	4	6	8
BCD	0 1 0 0	0 1 1 0	1 0 0 0
Bobot	800 400 200 100	80 40 20 10	8 4 2 1

Sekali lagi, setiap digit decimal diubah secara langsung menjadi biner yang setara. Perlu dicatat bahwa 4 bit biner selalu digunakan untuk setiap digit. Dengan demikian sandi 4 bit biner yang digunakan adalah dari 0000, 0001, 0010, 0011, …,hingga 1001. Dalam BCD tidak digunakan sandi-sandi 1010, 1011, 1100,1101,1110. dan 1111. Jika sembarang bilangan 4 bit yang terlarang itu terjadi pada mesin yang menggunakan sandi BCD, mka biasanya akan terjadi indikasi terjadinya kesalahan. Tampaknya penulisan dengan cara BCD ini merupakan pemborosan bit, karena 4 bit biner dapat melambangkan 16 bilangan (pada BCD hanya 10). Tetapi keuntungannya kita tidak perlu menulisakan bilangan yang lebih besar 9 (tidak dikenal A, B, …, F), sehinga BCD cocok untuk memperagakan bilangan decimal, cukup dengan mengubah setiap karakter BCD menjadi bilangan decimal yang diinginkan.

3.2. Sandi Excess-3(XS-3)

Kita juga menjumpai sandi bilangan lain yang menarik yang kadang-kadang sangat bermanfaat. Misalnya sadi Excess-3 (XS-3). Jenis sandi XS-3 ini seperti BCD, terdiri dari kelompok 4 bit untuk melambangkan sebuah digit decimal. Sandi Xs-3 untuk bilangan decimal dibentuk dengan cara yang sama seperti BCD kecuali bahwa 3 ditambahkan pada setiap digit decimal sebelum penyandian ke binernya. Misalkan untuk menyandi bilangan decimal 5 dalam XS-3, pertama kali menambahkan 3 kepada 5 yang menghasilkan 8, kemudian 8 disandikan ke dalam biner 4 bit yang setara, yaitu 1000.

5 + 3 = 8 ⟶ 1000.

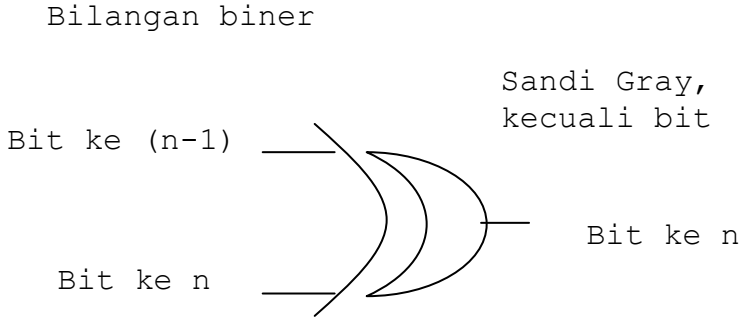
Sadi XS-3 hanya mengginakan 10 dari 16 kelompok sandi 4 bit yang mungkin. Kelompok sandi yang tidak valid (terlarang) pada sandi XS-3 adalah 0000, 0001,0010, 1101, 1110, dan 1111.

3.3. Sandi Gray

Sandi Grau termasuk kedalam system sandi tak berbobot karena posisi bit dalam kelompok sandi tidak memiliki nilai bobot tertentu. Dengan demikian sandi gray tidak cocok dalam operasi aritmetika, dan aplikasinya banyak dijumpai dalam piranti input/ouput dan ADC. Sandi Gray merupakan sandi yang berubah minimum karena sifatnya yang hanya berubah satu bit dalam kelompok apabila berubah satu digit bilangan ke digit bilangan berikutnya. Hal ini dapat mencegah terjadinya kesalahan dalan trasisi perubahan apabila bit yang berubah lebin dari satu, kemungkinan besar perubahan itu terjadi tidak bersamaan (satu bit lebih dulu berubah dari yang lain). Misalnya perubahan dari decimal 7 (binernya 0111) menjadi 8 (binernya 1000) yang selutuh bit mengalami perubahan yang biasanya dapat bertransisi dahulu ke biner 1111 (decimal 15). Kejadian 1111 tersebut sebenarnya hanya sementara

tetapi dapat menimbulkan operasi yang dapat mengacu unsur-unsur yang dikendalikan bit tersebut. Aturan untuk mengubah biner ke sandi Gray adalah sebagai barikut :

a. Bit pertama (paling kiri) sandi Gray sama dengan bit pertama dari bilangan biner.
b. Bit ke dua sandi Gray sama dengan EX-OR dari bit pertama dan bit ke dua bilangan biner. (EX-OR : sama dengan 1 bila kedua bit biner itu berbeda, dan 0 bila sama).
c. Bit sandi Gray ke tiga sama dengan EX-OR bit ke dua dan bit ke tiga bilangan biner.
d. Dan seterusnya, perhatukan gambar 3.2 yang merupakan gerbang EX-OR untuk mengubah bit-bit bilangan biner ke dalam sandi Gray, kecuali bit pertama.



Gambar 3.2:Pengubah sandi biner ke Gray

Sebagai contoh mengubah bilangan biner 10110 ke dalam sandi Gray (hasilnya 11101) adalah sebagai berikut :

```
1 0 1 1 0   (sandi biner)
1 1 1 0 1   (Sandi Gray)
Bit pertama
```

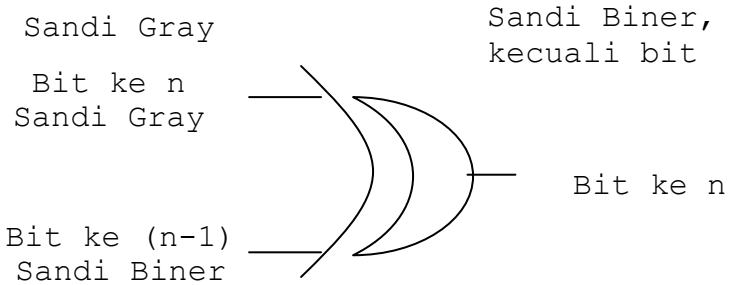
Selanjutnya untuk mengubah dari samdi Gray ke biner digunakan langkah-langkah (yang berlawanan dengan cara mengubah biner ke sandi Gray) adalah sebagai berikut :

a. bit pertama biner sama dengan bit pertama sandi Gray.
b. Bila bit ke dua sandi Gray 0, bit biner ke dua sama dengan yang pertama, bila bit sandi Gray ke dua 1, bit biner ke dua adalah kebalikan dari bit biner pertama.
c. Langkan b diulang untuk setiap bit berikutnya.

Sebagai contoh mengubah sandi Gray 1101 ke dalam biner hasilnya adalah 1001, seperti tampak pada ilustrasi berikut :

```
1  1     0       1   (sandi Gray)
1  0     0       1   (Sandi biner)
```

Ternyata setiap bit biner (kecuali yang pertama) diperoleh dengan mencari EX-OR dari bit sandi Gray yang sesuai dan bit biner sebelumnya. Perhatikan gambar 3.3 berikut !



Gambar 3.3 : Pengubah sandi Gray ke biner

Sebagai contoh mengybah sandi Gray 1101 ke dalam biner yang hasilnya 1001.

```
1   1     0       1   (sandi Gray)
1   0     0       1   (Sandi biner)
```

3.4 Sandi ASCII

Jika perhatikan tombol kunci pada computer, setidaknya ada 87 tombol kunci yang baik berupa huruf besar dan kecil, angka, tanda khusus, maupun tombol dengan fungsi khusus. Computer harus mampu menangani informasi numeric maupun non numeric, sehingga computer harus mampu mengenali berbagai sandi yang mencangkup angka, huruf, tanda, dan fungsi tertentu. Sandi-sandi ini dikelompokan sebagai sandi alpanumerik (alphabed and numeric). Sejumlah tombol yang lengkap dan memadai yang diperlukan ini meliputi (1) 26 tombol untuk huruf kecil, (2) 26 tombol untuk huruf besar, (3) 10 tombol untuk digit angka, dan (4) sekitar 25 tombol untuk tanda maupun fungsi khusus sepserti +, /, %, $, @, #, Esc,Insert,Page Up, dan seterusnya. Untuk menampilkan setidaknya 87 karakter yang berbeda tersebut dengan sandi biner setidaknya 7 bit. Demikian 7 bit akan diperoleh 2^7=128 sandi biner yang mungkin. Sebagai contoh sandi biner 1010101 menampilkan huruf U. sandi alpanumerik yang paling terkenal adalah sandi ASCII (American Standard Code for Information Interchange) yang digunakan oleh hampir seluruh computer. Pada table 3.2 berikut ini dikemukakan sebagian sandi ASCII.

Tabel 3.2 : Sebagian Sandi ASCII

Tanda	ASCII 7 bit	Tanda	ASSCII 7 bit
A	100 0001	S	101 0011
B	100 0010	T	101 0100
C	100 0011	U	101 0101
D	100 0100	V	101 0110
E	100 0101	W	101 0111
F	100 0110	X	101 1000
G	100 0111	Y	101 1001
H	100 1000	Z	101 1010
I	100 1001	0	011 0000
J	100 1010	1	011 0001
K	100 1011	2	011 0010
L	100 1100	3	011 0011
M	100 1101	4	011 0100
N	100 1110	5	011 0101
O	100 1111	6	011 0110
P	101 0000	7	011 0111
Q	101 0001	8	011 1000
R	101 0010	9	011 1001

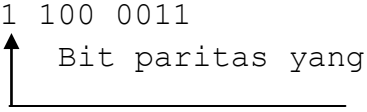
Sandi ASCII selengkapnya dapat dilihat pada daftar yang lebih lengkap di lrar buku ini. Sebagai contoh, seorang computer memasukan suatu pernyataan dari papan kunci berupa tulisan STOP yang maksudnya memerintah computer untuk memeintah suatu computer menghetikan suatu program, maka sandi biner yang dikenali computer adalah sebagai berikut:

S T O P
101 0011 101 0100 100 1111 101 0000

3.5 Bit Paritas

Pemindahan data dari suatu tempat ke tempat yang lain pada umumnya dalam bentuk biner. Misalnya perpindaha data dari computer ke kaset tape, pemindahan informasi jalur telepon, pengambilan data dari memori computer yang ditempatkan dalam unit aritmetik, dan sebagainya. Proses data yang dipindahkan tersebut dapat mengalami kesalahan sekalipun pirantinya telah dirancang sedemikian canggih. Meskipun terjadinya kesalahan itu relative kecil, tetapi dapat menghasilkan sesuatu yang tidak berguna bagi dan bahkan sangat fatal. Sehingga diperluka mekanisme pemeriksaan data untuk memperkecil kemungkinan terjadinya kesalahan data. Salah satu cara yang sangat terkenal untuk mendeteksi kesalahan adalah metoda peritas. Didalam sekelompok data ditambahkan bit yang disebut bit paritas. Jadi bit paritas merupakan tambahan yang disertakan ke dalam sekelompok sandi yang sedang dipindahkan dari setu tempat ke tempat yang lain. Bit paritas dapat berupa 0 dan 1 tergantung pada banyaknya angka 1 yang dimuat di dalam kelompok sandi itu, sehingga dikenal paritas genap dan paritas ganjil.

Pada metode paritas genap, nilai bit paritas dipilih sedemikian hingga banyaknya angka 1 dalam suatu kelompok sandi (termasuk bit paritas) berjumlah genap. Sebagai contoh suatu kelompok sandi 100 0011 yang merupakan huruf C pada sandi ASCII. Kelomsok sandi itu memiliki 1 sebanyak 3. Selanjutnya akan ditambahkan bit paritas 1 untuk membuat banyaknya angka 1 berjumlah genap. Kelompok sandi yang baru, termasuk bit paritas, kemudian menjadi



Jika suatu kelompok sandi berisi dalam jumlah genap, maka bit paritas yang ditambahkan bernilai 0. Sebagai contoh, suatu kelompok sandi 100 0001 (sandi ASCII untuk huruf A)

akan ditandai dengan bit paritas 0, sehingga diperoleh sandi yang baru (termasuk bit paritas) yaitu 0 100 0001.

Metode paritas ganjil digunakan dengan cara yang persis sama kecuali bahwa bit paritas dipilih sedemikian jumlah angka 1 (termasuk bit paritas) adalah ganjil. Sebagai contoh, untuk kelompok sandi 100 001 diberi bit paritas 1 sehingga diperoleh sandi baru sebagai 1 100 001. untuk kelompok sandi 100 0011 dikenal bit paritas 0 dan diperoleh sandi baru yakni 0 100 0011.

Terlepas dari paritas genap atau ganjil yang digunakan, bit paritas menjadi bagian yang nyata dari suatu sandi. Penambahan bit paritas kepada sandi ASCII 7 bit menghasilkan sandi 8 bit. Sehingga bit paritas diperlakukan seperti bit-bit lain di dalam sandi tersebut. Bit paritas digunakan untuk mendeteksi kesalahan bit tunggal yang terjadi selama pemindahan dari satu tempat ke temoat yang lain. Sebagai ilustrasi akan dipindahkan huruf A dan digunakan paritas ganjil. Kode yang dipindahkan berupa :

1 100 0001

Ketika rangkaian penerima menerima sandi ini, ia akan memeriksa untuk mengetahui bahwa sandi itu berisi 1 dalam ganjil (termasuk bit paritas). Sehingga pemerima akan menganggap bahwa sandi itu diterima benar. Selanjutnya dianggap bahwa karena suatu gangguan atau kegagalan, maka penerima sebenarnya menerima sandi sebagai:

1 100 0000

Penerima akan mendapatkan bahwa sandi tersebut berisi 1 dalam jumlah genap, ini memberitahu penerima bahwa pasti terjadi kesalahan sandi, karena sebelumnya antara pengirim dan penerima sandi telah setuju untuk menggunakan paritas ganjil, tidak ada cara bahwa penerima dapat memberitahukan bit mana yang mengalami kesalahan, karena ia tidak tahu sandi apa yang dimaksudkan.

Selanjutnya menjadi jelas bahwa metode paritas ini tidak akan bekerja jika terjadi 2 bit yang salah, seba 2

kesalahan tidak akan mengubah genap-ganjilnya jumlah 1 dalam sandi itu. Metode paritas hanya digunakan dalam keadaan di mana kemungkinan kesalahan satu bit kecil dan kemungkinan kesalahan dua bit boleh dikatakan tidak ada.

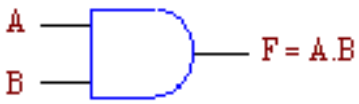
Bab 4
Gerbang Logika Dasar

Gerbang logika (logika gate) merupakan dasar pembentukan sistem digital. Gerbang logika beroperasi dengan bilangan biner, oleh karena itu gerbang tersebut di sebut gerbang logika biner. Tegangan yang digunakan dalam gerbang logika adalah TINGGI (HIGH) atau RENDAH (LOW). Dalam pembahasan ini di pakai logika positif, tinggi berarti biner 1, rendah berarti biner 0.

Semua sistem digital pada dasarnya hanya merupakan kombinasi gerbang logika dasar yaitu gerbang AND, gerbang OR, dan gerbang NOT (inverter).

4.1. Gerbang AND

Suatu gerbang AND biasanya mempunyai dua atau lebih dari dua masukan dan satu keluaran. Keluaran dari suatu gerbang AND akan menempati keadaan 1, jika dan hanya jika semua masukan menempati keadaan 1, bila salah satu masukannya menempati keadaan 0 maka keluarannya akan menempati keadaan 0. Simbol gerbang AND diperlihatkan dalam gambar 2.1, dan tabel kebenarannya diperlihatkan dalam tabel 4.1.



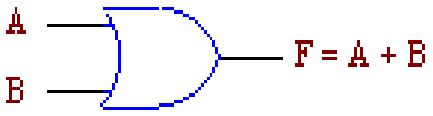
Gambar 4.1 Simbol Gerbang AND

Tabel 4.1 Tabel Kebenaran Gerbang AND

Masukkan		Keluaran (F)
A	B	
0	0	0
0	1	0
1	0	0
1	1	1

4.2. Gerbang OR

Gerbang OR mempunyai dua atau lebih dari dua masukan dan satu keluaran. Cara operasinya mengikuti definisi sebagai berikut : keluaran dari suatu gerbang OR menunjukkan keadaan 1 jika satu atau lebih dari satu masukkannya berada pada keadaan 1, keluarannya akan menempati keadaan 0 hanya apabila semua masukannya berada pada keadaan 0. Simbol gerbang OR diperlihatkan dalam gambar 4.2, dan tabel kebenarannya diperlihatkan dalam tabel 4.2.



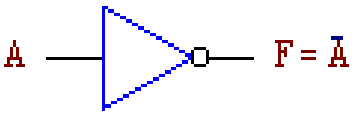
Gambar 4.2 Simbol Gerbang OR

Tabel 4.2 Tabel Kebenaran Gerbang OR

Masukkan		Keluaran (F)
A	B	
0	0	0
0	1	1
1	0	1
1	1	1

4.3 Gerbang NOT (inverter)

Gerbang NOT hanya mempunyai satu masukan dan satu keluaran, dan melakukan operasi logika peniadaan (negation) sesuai dengan definisi berikut : keluaran dari gerbang NOT akan mengambil keadaan 1 jika dan hanya jika masukanya tidak mengambil keadaan 1. Simbol gerbang NOT diperlihatkan dalam gambar 4.3, dan tabel kebenarannya diperlihatkan dalam tabel 4.3.



Gambar 4.3 Simbol Gerbang NOT

Tabel 4.3 Tabel Kebenaran Gerbang NOT

Masukkan (A)	Keluaran (F)
0	1
1	0

4.4 Rangkaian Logika Kombinasional

Rangkaian logika kombinasional terdiri dari kombinasi gerbang- gerbang logika dasar yang akan menghasilkan keluaran sebagai hasil tanggapan adanya dua atau lebih dari dua variable masukan. Keluarannya itu tergantung pada kombinasi gerbang yang digunakan dan masukannya saat itu juga, tanpa memandang masukan sebelumnya.

Perencanaan rangkaian logika kombinasi-onal berawal dari uraian garis besar yang dinyatakan dengan kata-kata untuk suatu masalah dan berakhir dengan suatu diagram logika atau suatu himpunan fungsi Boole yang dapat diimplementasukan menbjadi suatu rangkaian logika.

Prosedur itu meliputi beberapa langkah berikut :

1. Pernyataan masalah yang direncanakan
2. Penetapan banyaknya variable masukan yang tersedia dan variable keluaran yang diperlukan

3. Pemberian lambang huruf untuk setiap variable masukan dan keluaran
4. Penulisan tabel kebenaran yang mendefinisikan hubungan antara masukan dengan keluaran
5. Penulisan persamaan keluaran yang paling sederhana
6. Implementasi rangkaian

Bab 5
Aljabar Boole dan Peta Karnaugh

5.1 Pengertian Aljabar Boole

Dikenal banyak macam aljabar seperti aljabar biasa, aljabar himpunan, aljabar vector, aljabar boole, dan lain-lain. Dalam setiap aljabar memiliki postulat,aturan main dan operasi sendiri-sendiri. Aljabar boole berbeda dengan aljabar biasa atau aljabar yang lain. Aljabar boole diciptakan pada abad 19 oeh George Boole sebagai suatu system untuk menganalisa secara matematis mengenai logika. Alnabar boole didasarkan pada pernyataan logika bernilai benar atau salah. Ternyata, aljabar boole ini menjadi alat yang sangat ampuh untuk merancang maupun menganalisa rancangan digital. Selanjutnya, dalam aljabar boole baik konstantamaupun nilai dari suatu variable hanya diijinkan memiliki 2 kemungkinan nilai (biner) yaitu 0 dan1. variable aljabar boole sering digunakan untuk menyajikan suatu tingkat tegangan pada terminal suatu rangkaian. Terminal itu dapat berupa kawat atau saluran masukan /keluaran suatu rangkaian. Misalnya 0 sering digunakan untuk menandai suatu jangkauan tegangan dari 0 volt sampai dengan 0,8 volt. Sedangkan 1 sering digunakan untuk jangkauan tegangan dari 2 volt hingga 5 volt. Dengan demikian tanda 0 dan 1 tidak menggambarkan bilanga yang sebenarnya tetapi menyatakan keadaan suatu variable tegangan.

Aljabar boole digunakan untuk menyatakan pengaruh berbagai rangkaian digital pada masukan-masukan logika, dan untuk memani- pulasi variable logika dalam menentukan cara terbaik pada pelaksanaan (kinerja) fungsi rangkaian tertentu. Oleh karena hanya ada 2 nilai yang mungkin,

aljabar boole lebih cocok digunakan untuk rangkaian digital digital dibandingkan dengan aljabar yang lain. Dalam aljabar boole tidak ada pecahan, decimal, bilangan negatif, akar kwadrat, akar pangkat tiga, logaritma, bilangan imajiner, dan sebagainya. Kenyataannya, dalam aljabar boole hanya mengenal 3 (tiga) operasi dasar, yaitu :

1) Penjumlahan logika atau OR denan symbol operasi '+' (tanda plus).
2) Perkalian logika atau AND dengan symbol operasi '.' (tanda titik) atau tanpa tanda sama sekali.
3) Komplementasi atau NOT (atau inverse) dengan symbol operasi '¯' (garis diatas variable).

Atau operasi OR, AND dan NOT pada dua tingkat logika 0 dan 1 dapat dirangkum sebagai berikut :

OR	**AND**	**NOT**
0 + 0 = 0	0 . 0 = 0	0 = 1
0 + 1 = 1	0 . 1 = 0	1 = 0
1 + 0 = 1	1 . 0 = 0	
1 + 1 = 1	1 . 1 = 1	

Selanjutnya akan terlibat bahwa aljabar boole dapat digunakan sebagai salah satu cara untuk menganlisa rangkaian logika dan menyatakan operasinya secara metematik, terutama untuk mendapatkan konfigurasi rangkaian yang paling sederhana (paling sedikit jumlah komponen).

5.2 Teorema dalam Aljabar Boole

Sebagaimana telah dikemukakan sebelumnya bahwa dalam setiap aljabar memiliki potulat, aturan main , dan operasi sendiri-sendiri. Ketiga hal tersebut saling terkait dan terangkum dalam istilah teorema. Berdasarkan

torema dalam aljabar boole dapat membantu menyederhanakan pernyataan dan rangkaian logika. Sekali lagi dalam aljabar boole setiap konstanta dan setiap variable hanya dapat bernilai 0 atau 1. Teorema dalam aljabar boole meliputi :

1) A . 0 = 0
2) A . 1 = A
3) A . A = A
4) A . $\bar{A}$ = 0
5) A + 0 = A
6) A + 1 = 1
7) A + A = A
8) A + $\bar{A}$ = 1.

Teorema 1) hingga 8), variable A sebenarnya dapat menyajikan suatu pernyataan yang ber-isi lebih dari satu variable. Sebagai contoh, bentuk yang lebih sederhana dari pernyataan XY + XY dapat ditentukan dengan memisalkan XY = A . kemudian diperoleh A + A = A. Dengan demikian XY + XY = XY.

Teorema berikut mencakup lebih dari satu variable, yaitu:

9) A + B = B + A (komutatif OR)
10) A . B = B . A (komutatif AND)
11) A + (B + C) = (A +B) + C = A + B + C (asosiatif OR)
12) A(BC) = (AB)C = ABC (asosiatif OR)
13) A(B + C) = AB + AC (distributi OR)
14) (A + B)(C + D) AC + BC + AD + BD (distributif AND)
15) A + AB = A
16) A + AB = A + B

Masih ada dua teorema dalam aljabar boole yang sangat penting yang disumbangkan oleh matematikawan De Morgan.

Kedua toerema de Morgan tesebut adalah:

$$17)\quad \overline{(A + B)} = \bar{A} \cdot \bar{B}$$

$$18)\quad \overline{(A \cdot B)} = \bar{A} \cdot \bar{B}.$$

Teorema de Morgan itu tidak hanya berlaku untuk dua variable, selain A dan B masing-masing terdiri dari lebih dari satu variable tetapi operasi OR atau AND dapat diteruskan pada variable berikutnya, seperti :

$$\overline{(A + B + C + D + \ldots)} = \bar{A} \cdot \bar{B} \cdot \bar{C} \cdot \bar{D} \cdot \ldots.$$

$$\overline{(A \cdot B \cdot C \cdot D \cdot \ldots)} = \bar{A} + \bar{B} + \bar{C} + \bar{D} + \ldots$$

Meminimalisasi Rangkaian Logika secara Analitis

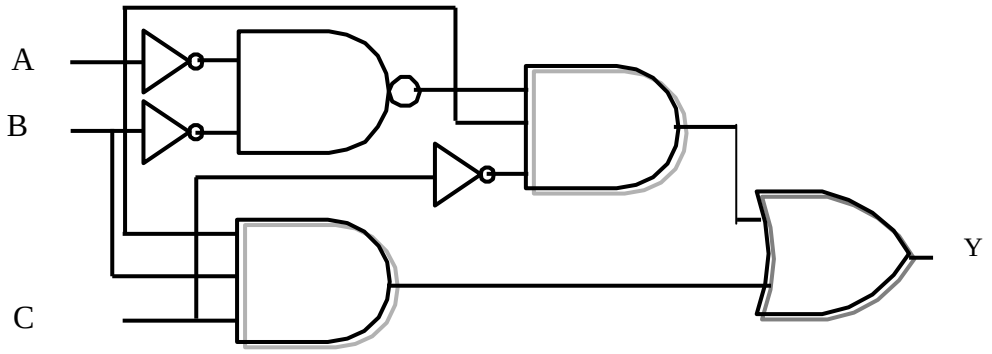
Realisasi rangkaian logika dengan fungsi tertentu dari suatu pernyataan logika pada umumnya tidak unik, artinya ada berracam-macam konfigurasi rangkaian dengan fungsi yang sama. Tentu saja diinginkan cara ataupun konfigurasi yang paling sederhana, atau paling mudah dilaksanakan. Dengan rangkaian yang sederhana memiliki banyak keuntungan misalnya lebih ekonomis, tidak rumit sehingga memminimalkan kemungkinan terjadinya kesalahan, mengurangi efek pembebanan dan segalanya. Banyak yang mencari metode terbaik untuk keperluan penyederhanaan itu. Salah satu metode penyederhanaan rangkaian logika adalah dengan metoda analistis. Metode analistis ini menggunakan torema-teorema aljabarboole.

Sebagai ilustrasi marilah mencari bentuk yang paling sederhana setidaknya dari pernyataan atau fungsi logika berikut ;

```
Y   = ABC +AB (A . C)
    = ABC + AB (A + C)
    = ABC + AB(A + C)
    = ABC + AAB + ABC
    = ABC + AB +ABC
    = AC +(B + B) + AB
    = AC (1) + AB
    = A (C + B)
```

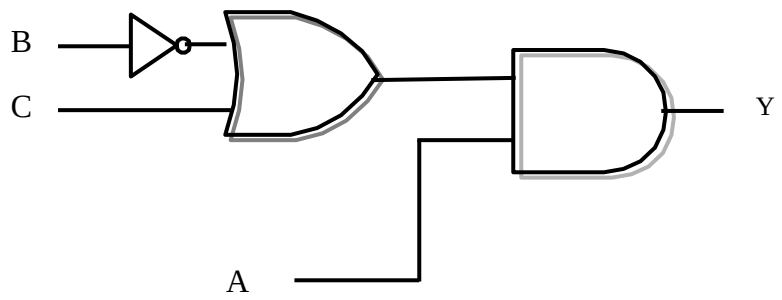
Jadi Y = ABC + AB (A . C) memiliki bentuk yang paling sederhana Y = A (C + B).

Realisasi rangkaian Y = ABC + AB (A . C) dapat diperhatikan pada gambar 5.1 berikut:



Gambar 5.1 : Realisasi Y = ABC + AB (A.C)

Sedangkan realisasi dari Y = A (C + B) dapat dilihat pada gambar 5.2 dibawah ini :



Gambar 5.2 : Realisasi Y = A (C + B)

Jika dibandingka dengan seksama, rangkaian yang ditujukan pada gambar 5.2 jauh lebih sederhana dari pada rangkaian pada gambar 5.1 padahal kedua rangkaian memiliki fungsi yang sama.

Selanjutnya, untuk lebih memahami penggunaan postulat boole dalam rangka penyederhanaan suatu pernyataan logika, perhatikanlah dengan seksam acontoh-contoh berikut ini !

Contoh 1 :
Buktikanlah bahwa (A +B)(A + C) = A + BC !

Penyelesaian :

$$
\begin{aligned}
(A+B)(A+C) &= AA + AC + AB + BC \\
&= A + AC + AB + BC \\
&= A + AB + AC + BC \\
&= A\,(1 + B) + C\,(A + B) \\
&= A + C\,(A + B) \\
&= A + AC + BC \\
&= A\,(1 + C) + BC \\
&= A + BC.
\end{aligned}
$$

Contoh 2 :
Sederhanakan pernyataan logika berikut ini!

$$ABC + ABC + ABC + ABC$$

Penyelesaian :

$$ABC + ABC + ABC + ABC$$
$$= A\,(BC + BC) + A\,(BC + BC)$$
$$= A\,\{C(B + B)\} + A\,\{C(B + B)\}$$
$$= AC + AC$$
$$= C.$$

Contoh 3 :
Carilah bentuk uang paling sederhana dari pernyataan logika :

$$ABCD + ABCD + ABCD + ABCD$$

Penyelesaian :

$$ABCD + ABCD + ABCD + ABCD = ABC(D + D) + ABC\,(D + D)$$
$$= ABC + ABC$$
$$= AB\,(C + C)$$
$$= AB.$$

Contoh 4 : Carilah bentuk sederhana dari pernyataan berikut :

(A+B+C)(A+B+C)(A+B+C)(A+B+C)

Penelesaian :
(A+B+C)(A+B+C)(A+B+C) (A+B+C)
= (A + B)(A + B)(A + C)
= B (A+ C)

Contoh 5 :
Buktikanlah bahwa
(A+B+C) + (A+B+C) + (A+B+C) = B(A+C)

Penyelesaian :
(A+B+C) + (A+B+C) + (A+B+C)= A(BC+BC)+ ABC)
= AB + ABC
= B(A + AC)
= AB + BC
= B (A + C)

Aljabar Boole antara lain berperan untuk menyederhanakan fungsi logika. Sedangkan fungsi logika sangat penting untuk merancang rangkaian logika. Realisasi dari suatu fungsi logika tidaklah unik, melainkan bermacam-macamdengan hasil yang sama, sehingga dimungkinkan untuk mencari cara yang paling sederhana atau yang paling mudah untuk direalisasikan. Dalam merancang rangkaian digital juga perlu menerapkan fungsi ekonomi, yakni tujuan tercapai dengan resiko, waktu, dan biaya sesedikit mungkin. Tujuan dibuatnya rangkaian elektronik adalah untuk mendapatkan rangkaian dengan fungsi tertentu. Para perancang rangkaian elektronik telah banyak mengeluarkan tenaga danmeluangkan waktu guna menemukan cara yang tepat untuk dapat merancang rangkaian elktronik dengan komponen sesedikit mungkin (minimal)

untuk mendapatkan hasil yang sama. Dengan jumlah komponen minimal, selain lebih ekonomis tetapi secara teknis lebih mengurangi kerumitan rangkaian, konsumsi daya lebih rendah, dan mengurangi efek pembebanan rangkaian. Cara-cara yang dapat ditempuh untuk tujuan tersebut meliputi cara analitis (dengan aljabar Boole), cara grafis, maupun dengan menggunakan computer.

Peta Karnough digunakan sebagai cara penyederhanaan persamaan logika secara grafis, atau dapat pula dipandang sebagai metoda untuk merubah suatu table kebenaran ke rangkaian logika yang sesuai secara sederhana dan rapi. Keuntungan penggunaan peta Karnough adalah dapat melihat bentuk umum persoalan dan memungkinkannya melakukan penyederhanaan dengan cepat. Dengan demikian, minimalisasi logika kombinasional dengan metoda peta Karnough dapat memperoleh hasil yang lebih cepat bila dibandingkan dengan metoda analitis. Meskipun secara prinsip metode peta Karnough dapat digunakan dapat digunakan untuk menyelesaikan persoalan dengan sejiumlah variable masukan, tetapi secara praktishanya efektif (terbatas) untuk enam variable saja. Sedangkan untuk persoalan yang melibatkan lebih dari enam variable masukan akan sangat baik jika diselesaikan dengan bantuan program computer.

5.3 Bentuk Standar Fungsi Boole

Bentuk Boole sangat penting untuk merancang rangkaian digital, khususnya rangkaian logika. Sedangkan aljabar Boole sangat berperan dalam penyedehanaan fungsi Boole. Batasan fungsi Boole tentu saja memenuhi operasi-operasi dalam aljabar Boole. Pernyataan logika AND, OR, NOT, dan kombinasinya dipenuhi oleh fungsi Boole. Dengan demikian fungsi Boole merupakan fungsi yang menyatakan hubungan antara variable-variabel masukan dan keluaran ddalam rangkaian logika. Jika suatu fungsi Boole memiliki variable-variabel masukan A, B, C, D … dan variable

keluarannya adalah Y, maka hubungan antara variable-variabel masukan dan keluaran tersebut secara umum dapat dinyatakan sebagai :

$$Y = f\ (A,B,C,D,\ldots)$$

1. Jumlah dari Hasil Kali (Sum Of Product)

Untuk memahami hubungan fungsi antara fungsi Boole, table kebenaran, dan peta Karnough terlebih dahulu ditinjau suatu bentuk khusus dari persamaan (5-1) sebagai:

$$Y = f\ (A,B,C) = \overline{A}C + B\overline{C}$$

Tabel kebenaran dari persamaan (5-2) tampak pada table 5.1 sebagai :

Tabel 5.1

Baris ke-	**A**	**B**	**C**	$\overline{AC}$	$\overline{BC}$	$Y = \overline{AC} + \overline{BC}$
0	0	0	0	0	0	0
1	0	0	1	1	0	1
2	0	1	0	0	1	1
3	0	1	1	1	0	1
4	1	0	0	0	0	0
5	1	0	1	0	0	0
6	1	1	0	0	1	1
7	1	1	1	0	0	0

Dengan memperhatikan nomor baris dimana Y = 1, dapat diperoleh :

Y = 1
= baris 1 atau baris 2 atau baris 3 atau baris 6
= 001 + 010 + 011 + 110

$= \bar{A}\,\bar{B}C + A\bar{B}\bar{C} + A\bar{B}C + AB\bar{C}$

Fungsi boole seperti disajikan pada persamaan (5-3) merupakan bentuk standar jumlah dari hasil kali (sum of product). Jika diperhatikan dengan seksama, setiap bentuk sum of product memenuhi sifat-sifat sebagai berikut :

a. Fungsi tersebut merupukan jumlahan (OR) dari suku-suku
b. Setiap suku berupa perkalian (AND) dari variable variable
c. Semua variable muncul pada setiap suku (bentuk kanonik)

Setiap suku dari fungsi Boole dalam bentuk sum of product juga disebut **minterm** (suku minimum). Untuk menyingkat penulisan, setiap minterm diberi symbol **m** yang diikuti angka indeks menurut nomor barisnya.

Untuk persamaan (6-3) dapat dituliskan kembali sebagai :

Y = ABC + ABC + ABC + ABC
= 001 + 010 + 011 + 110
= $m_1 + m_2 + m_3 + m_4$
= Σ m (1,2,3,6).

Secara sederhana menterm atau sum of product dapat disajikan dengan cara sebagai berikut :

Nyatakanlah A,B,C,D…. dengan 1 dan $\overline{A}$, $\overline{B}$, $\overline{C}$, $\overline{D}$, … dengan 0

a. Nyatakanlah kombinasi bener setiap suku menjadi decimal
b. Nyatakanlah Y = Σ m (n), dengan n merupakan nilai decimal dari setiap suku.

Contoh 1 :

Y = ƒ (A,B,C)

$$= \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C} + \overline{A}\,\overline{B}\,\overline{C} + A\,B\,C$$
$$= 111 + 110 + 101 + 011 + 000$$
$$= m_7 + m_6 + m_5 + m_1 + m_0$$
$$= \Sigma m\ (0,1,5,6,7)$$

Contoh 2 :

Y = ƒ (A,B,C,D)
= Σ m (0,2,5,6,7,13).
= $m_0 + m_2 + m_5 + m_6 + m_7 + m_{13}$
= 0000+0010+0101 +0110+0111+1101

$$= \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + A\,\overline{B}\,CD + ABC\overline{D} + AB\overline{C}D + ABCD$$

2. Hasil Kali dari Jumlah (Product of Sum)

Berdasarkan table kebenaran dari dari persamaan (5-2) dapat juga diperhatikan nomor baris dimana Y = 1 atau Y = 0, dan selajutnya dapat dituliskan sebagai berikut:

$$Y = 1$$
$$= \text{baris } 0 \text{ atau } 4 \text{ atau } 5 \text{ atau } 7$$
$$= 000 + 100 + 101 + 111$$
$$= \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + A\bar{B}C + AB\bar{C}$$

Dengan sifat $\overline{AB} = \bar{A} + \bar{B}$ dan $\overline{A + B} = \bar{A}\bar{B}$ persamaan (6-5) dapat dituliskan menjadi

$$Y = \bar{Y} = (\bar{A}\bar{B}\bar{C}) + (\bar{A}\bar{B}C) + (\bar{A}BC) + (A\bar{B}C)$$
$$= (\bar{A}\bar{B}\bar{C})(\bar{A}\bar{B}\bar{C})(\bar{A}B\bar{C})(A\bar{B}\bar{C})$$
$$= (A+B+C)(A+B+C)(A+B+C)(A+B+C$$

Fungsi Boole seperti disajikan pada persamaan (6-6) merupakan bentuk standar hasil kali dari jumlah (Product of Sum). Jika diperhatikan dengan seksama setiap bentuk product of sum memenuhi sifat-sifat:

a. Fungsi-fungsi tersebut terdiri dari factor
b. Setiap factor berupa jumlahan (OR) dari variable-variabel
c. Semua variable fungsi muncul pada setiap factor (bentuk kanonik).

Setiap factor dari fungsi Boole dalam bentuk product of sum juga disebut maxterm (suku maximum). Untuk menyingkat penulisan, setiap maxterm diberi simbo **M** yang diikuti dengan angka indeks menurut nomor barisnya.

Untuk persamaan (5-6) dapat dituliskan kembali sebagai :

$$Y = (\bar{A}+\bar{B}+C)\ (\bar{A}+B+\bar{C})\ (\bar{A}+B+C)\ (A+\bar{B}+\bar{C})$$
$$= 000.\ 100\ .\ 101\ .\ 111$$
$$= M_0.M_4.M_5.M_7$$
$$= \prod M\ (0,4,5,7). \qquad (5\text{-}7)$$

Secara sederhana maxterm atau product of sum dapat disajikan dengan cara sebagai berikut :

a. Nyatakanlah $\bar{A},\bar{B},\bar{C},\bar{D}\ldots$ dengan 0 dan A, B, C, D,…dengan 1
b. Nyatakanlah kombinasi biner setiap factor menjadi decimal (n)
c. Nyatakanlah $Y = \prod M\ (n)$, dengan n merupakan nilai decimal dari setiap factor .

Contoh 3 :

Ubahlah minterm $Y = f\ (A,B,C) = A\bar{B}\bar{C} + A\bar{B}C + AB\bar{C} + ABC$ menjadi maksterm !

$$Y = A\bar{B}\bar{C} + A\bar{B}C + AB\bar{C} + ABC$$
$$= 111 + 100 + 010 + 011$$
$$= 7 + 4 + 2 + 3$$
$$= \Sigma\ m\ (2,3,4,7)$$

Sedangkan bentuk makstermnya adalah :

$$Y = Y\ f(A,B,C)$$
$$= \prod M\ (0,1,5,6)$$
$$= 000\ .\ 001\ .\ 101\ .\ 110$$
$$= (A+B+C)\ (A+B+\bar{C})\ (\bar{A}+B+\bar{C})\ (\bar{A}+B+C)$$

2. Peta Karnough(Peta K)

Peta Karhough digunakan sebagai salah satu metode untuk menyederhanakan fungsi Boole (pernyataan logika). Peta kaornough merupakan penggambaran secara grafik semua kombinasi variable-variabel yang terlibat dalam suatu pernyataan logika. Dengan demikian peta Karnough merupakan metode untuk menunjukan hubungan antara variable masukan dan keluaran yang diinginkan. Peta Karnough terdiri dari kolom dan baris bergantung pada banyaknya variable yang terlibat dalam suatu pernyataan logika. Beberapa catatan tentang peta Karnough adalah sebagai berikut :

a. Jika ada m variable untuk kolom dan n variable untuk baris, maka diperlukan 2^m kolom dan 2^n baris yang membentuk $2^{(m+n)}$ kotak atau sel. Jumlah kotak tersebut sama dengan banyaknya baris dalam table kebenaran. Hal ini juga berarti bahwa banyaknya variable fungsi logika adalah (m+n).
b. Nilai dari kombinasi variable pada setiap sel digunakan untuk memberikan nomor sel yang bersangkutan. Nilai tersebut menunjukan nomor baris pada table kebenaran.
c. Sel-sel pada peta Karnough digunakan untuk meletakan suku minterm atau factor maksterm yang sesuai.
d. Tanda 1 digunakan untuk menyatakan bahwa suatu sel berisi minterm, sedangkan tanda 0 menyatakan bahwa sel itu berisi maksterm.

Untuk lebih jelasnya marilah kita perhatikan beberapa contoh berikut, langsung saja dimulai dari fungsi logika dengan 3 variabel masukan sebagai Y = ƒ (A,B,C). Tabel kebenaran fungsi logika tersebut ditentukan seperti tampak pada table 5-2 berikut ini.

Tabel 5.2

Nomor Baris	Masukan			Keluaran
	A	**B**	**C**	**Y**
0	0	0	0	1
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
7	1	1	1	0

Berdasarkan pada table 6-2 dapat dituangkan dalam peta Karnough dengan beberapa cara. Cara pertama m = 2 (cacah variable untuk kolom ada 2, yaitu A dan B)., dan 1 (cacah variable untuk baris ada 1, yaitu C). Dengan demikian cacah kolom ada $2^m = 2^2 = 4$, dan cacah baris ada $2^n = 2^1 = 2$.

Peta Karnough untuk cara tersebut adalah sebagai berikut:

AB **C**	**AB** **00**	**AB** **01**	**AB** **11**	**AB** **10**
C : 0	000 0	010 2	110 6	100 4
C : 1	001 1	011 3	111 7	101 5

Cara ke dua kita tetapkan m = 1 (cacah variable untuk kolom ada 1, yaitu A), dan n = 2 (cacah variable untuk baris ada 2, yaitu B dan C). Dengan demikian cacah kolom ada $2^m = 2^1 = 2$, dan cacah baris ada $2^n = 2^2 = 4$.

Peta Karnough untuk cara tersebut adalah sebagai berikut:

BC **A**	**A** **0**	**A** **1**
B C : 00	000 0	100 4
B C : 01	001 1	101 5
BC : 11	011 3	111 7
BC : 10	010 2	110

Untuk kedua cara diatas diatas masing-masing memiliki cacah sel yang sama, yaitu $2^{(m+n)} = 2^{(2+1)} = 2^{(1+2)} = 2^3 = 8$. Perhatikan bahwa nomor sel ditujnukan oleh kombinasi biner dari variable yang bersilangan di sel itu. Tetapkan dahulu bahwa kita akan memilih menyatakan fungsi logika dalam bentuk sum of product yang berarti pula kita menggunakan bentuk minterm. Sehingga berdasarkan table 6-2 kita ambil nomor baris dimana Y = 1, yaitu terjadi pada baris-baris nomor 0, 1, 2, dan 6. Dengan demikian, kita menempatkan 1 ke dalam sel-sel yang bernomor 0, 1, 2, dan 6 tadi.

Setelah bentuk minterm tersebut diisikan pada sel-sel yang sesuai akan diperoleh peta Karnough seperti berikut:

AB / **C**	**AB** **00**	**AB** **01**	**AB** **11**	**AB** **10**
C : 0	0 1	2 1	6 1	4
C : 1	1 1	3	7	5

Atau :

A / **BC**	**A** **1**	**A** **1**
BC : 00	0 1	4
BC : 01	1 1	5
BC : 11	3	7
BC : 10	2 1	6 1

Pernyataaan sum of product untuk keluaran Y pada peta karnough yang telah diisi dengan 1 dapat diperoleh dengan cara **meng-OR-kan** bersama seluruh sel yang berisi satu. Pada peta Karnough dengan tiga variable baik untuk keadaan m = 2 dan n = 1 maupun keadaan m = 1 dan n = 2 seperti diatas, maka pernyataan logika setiap sel yang berisi 1 adalah ABC (sel 0), ABC (sel 1), ABC (sel2), dan ABC (sel 6), sehingga pernyataan untuk keluarannya adalah Y = ABC + ABC + ABC + ABC.

Tetapi pernyatan keluaran demikian (peng-OR-an) masih dapat disederhanakan lagi dengan cara mengelompokan sel-sel yang berdekatan dalam peta Karnough yang berisi 1. Proses penggabungan tersebut dinamakan operasi pengelompokan (looping). Dasar pengelompokan itu adalah postulat yang berbentuk A + A = 1.

Kelompok 1 : sel 0 dengan sel 1

$$: \bar{A}\,\bar{B}\,\bar{C} + A\,\bar{B}\,\bar{C}$$

$$: \bar{A}\,\bar{B}\,(\bar{C} + C)$$

$$: \bar{A}\,\bar{B}\,(1)$$

Kelompok 2 : sel 2 dengan sel 6

$$: \bar{A}\,\bar{B}\,C + A\,\bar{B}\,C$$

$$: (\bar{A} + A)\bar{B}C$$

$$: (1)\bar{B}C$$

$$: \bar{B}C.$$

Sel 0 dan sel 2 juga dikelompokan karena kedua sel juga saling berdekatan. Tetapi karena sel 0 telah dikelompokan dalam kelompok-1 dan sel 2 dalam kelompok-2, maka kedua sel tersebut tidak perlu dikelompokan lagi. Jika semua sel telah dikelompokan, maka hasil akhirnya diperoleh dengan cara meng-OR-kan semua kelompok yang dihasilkan.

Bab 6
Dekoder (Demultiplekser) dan Multiplekser

6.1 Dekoder BCD ke Desimal

Dekoder ialah suatu rangkaian logika kombinasional yang berfungsi untuk mengubah kode bahasa mesin (biner) menjadi kode bahasa yang dapat dimengerti manusia. Dekoder BCD ke Desimal mengubah kode biner menjadi bentuk decimal. Untuk merencanakan decoder BCD ke decimal terlebih dahulu perlu menentukan cara kerjanya. Keluaran yang diperlukan adalah dalam bentuk desimal, sehingga saluran keluaran yang diperlukan sebanyak 10 saluran. Setelah banyaknya saluran keluaran ditentukan, maka dapat diketahui saluran masukan yang diperlukan sebanyak 4 saluran. Setiap saluran masukan misalkan diberi lambang huruf A, B, C, dan D, sedangkan setiap saluran keluaran misalkan diberi lambing huruf Y_0 sampai Y_9. Untuk keluaran aktif high, salah satu keluaran akan menempati keadaan 1 sesuai dengan kombinasi masukan yang diberikan, sedangkan saluran keluaran yang lainnya akan menempati keadaan 0. Hubungan masukan dan keluarannya diperlihatkan dalam tabel 6.1.

Tabel 6.1
Tabel Kebenaran Dekodert BCD ke Desimal

Masukan				**Keluaran**									
D	C	B	A	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	Y_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**

Setelah membuat tabel kebenaran, langkah selanjutnya menuliskan persamaan keluaran berdasarkan tabel kebenaran yang telah di buat.

Dengan menggunakan '*sum of product*', diperoleh persamaan keluaran sebagai berikut:

$$Y_O = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} \qquad Y_1 = \overline{A}\,\overline{B}\,\overline{C}D$$

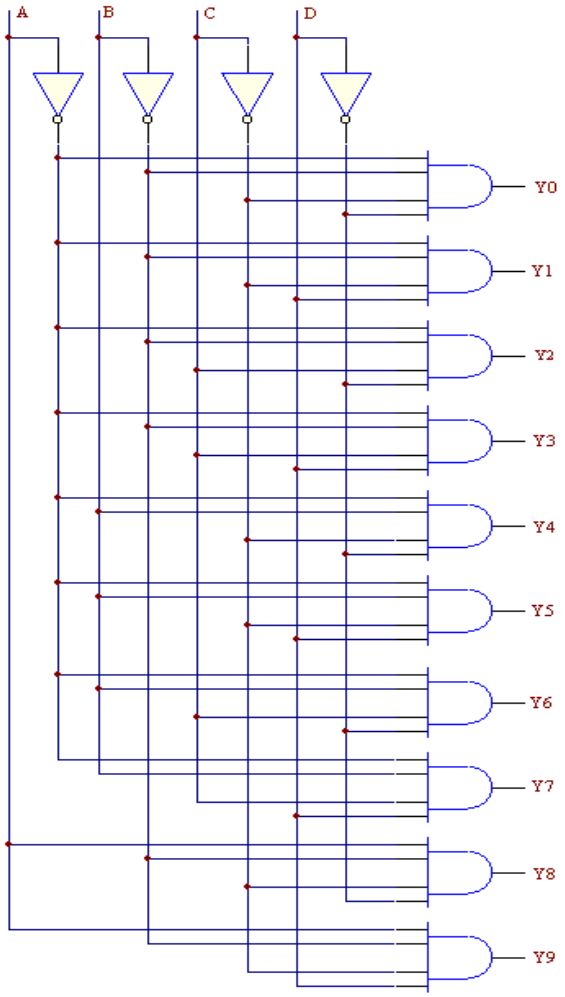
$$Y_2 = \overline{A}\,\overline{B}C\,\overline{D} \qquad Y_3 = \overline{A}\,\overline{B}CD$$

$$Y_4 = \overline{A}B\,\overline{C}\,\overline{D} \qquad Y_5 = \overline{A}B\,\overline{C}\,D$$

$$Y_6 = \overline{A}BC\,\overline{D} \qquad Y_7 = \overline{A}BC\,D$$

$$Y_8 = A\,\overline{B}\,\overline{C}\,\overline{D} \qquad Y_9 = A\,\overline{B}\,\overline{C}\,D$$

Langkah terakhir, dari persamaan keluaran dapat diimplementasikan ke dalam bentuk rangkaian, seperti diperlihatkan dalam gambar 6.1.

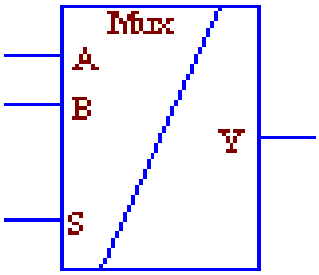


Gambar 6.1 Rangkaian Dekoder BCD ke Desimal

6.2 Multiplekser

Multiplekser berfungsi sebagai data selector. Data masukan yang terdiri dari N sumber, di pilih salah satu dan diteruskan kepada suatu saluran tunggal. Masukan data dapat terdiri dari beberapa jalur dengan masing-masing jalur dapat terdiri dari satu atau lebih dari satu bit. Masukan selektor terdiri dari satu atau lebih dari satu bit, tergantung dari banyaknya jalur masukan. Keluaran hanya terdiri dari satu jalur satu atau lebih dari satu bit.

Perencanaan multiplekser di mulai dari penentuan banyaknya jalur masukan dan banyaknya bit, dan diakhiri dengan implementasi rangkaian sesuai dengan prosedur perancangan rangkaian logika kombinasional. Misalnya perencanaan multiplekser satu bit 2 ke 1, dengan simbol seperti diperlihatkan dalam gambar 2.5.



Gambar 6.2 Multiplekser 1 bit 2 ke 1

Masukan data terdiri dari dua jalur A dan B, serta satu keluaran Y. Dari banyaknya saluran dapat ditentukan banyaknya bit masukan selektor S yaitu satu bit. Hubungan masukan dan keluaran didefinisikan dengan tabel kebenaran seperti yang diperlihatkan dalam tabel 6.2.

Tabel 6.2
Tabel kebenaran Multiplekser 1 bit 2 ke 1

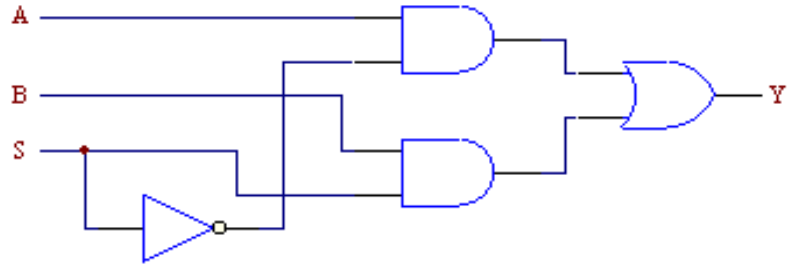
Masukan			**Keluaran**
A	B	S	**(Y)**
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Jika kondisi selektor S = 0, maka keluaran Y = A, jika S = 1 maka Y = B. Dari tabel kebenaran di atas, dengan menggunakan 'sum of product'.

Diperoleh persamaan keluaran sebagai berikut:

$$Y = \overline{A}BS + A\overline{B}\overline{S} + AB\overline{S} + ABS$$
$$= \overline{S}A(B + \overline{B}) + SB(A + \overline{A})$$
$$= \overline{S}A + SB$$

Langkah terakhir adalah mengimplementasikan persamaan keluaran ke dalam bentuk gambar rangkaian seperti diperlihatkan dalam gambar 6.3.

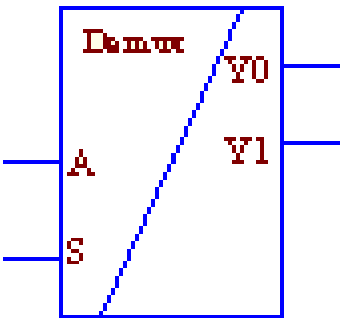


Gambar 6.3 Multiplekser 1 bit 2 ke 1

6.3 Demultiplekser

Demultiplekser berfungsi sebagai data distributor data. Demultiplekser menyalurkan sinyal biner (data serial) pada salah satu dari N saluran keluaran yang tersedia, dan pemilihan saluran khusus tersebut ditentukan melalui alamatnya. Masukan data dapat terdiri dari beberapa bit. Keluaran terdiri dari beberapa jalur, masing-masing jalur terdiri dari satu atau lebih dari satu bit. Masukan selector terdiri dari satu atau lebih dari satu bit tergantung pada banyaknya jalur keluaran.

Simbol demultiplekser 1 bit 1 ke 2, diperlihatkan dalam gambar 6.4. Masukan ditandai dengan huruf A. Keluaran terdiri dari dua jalur Y_0 dan Y_1. Karena keluarannya terdiri dari dua jalur, maka masukan selektor S hanya satu bit.



Gambar 6.4 Demultiplekser 1 bit 1 ke 2

Hubungan masukan dan keluaran didefinisikan dengan tabel kebenaran seperti diperlihatkan dalam tabel 6.3.

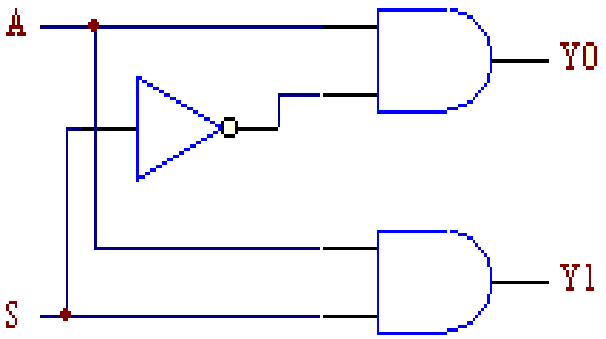
Tabel 6.3
Tabel Kebenaran Demultiplekser 1 bit 1 ke 2

Masukan		**Keluaran**	
S	A	Y0	Y1
0	0	0	0
0	1	1	0
1	0	0	0
1	1	0	1

Dengan menggunakan 'sum of product', dari tabel kebenaran diatas dapat diperoleh persamaan sebagai berikut :

$Y_0 = \overline{S}A \quad Y_1 = SA$

Persamaan keluaran Y_0 dan Y_1 dapat diimplementasikan ke dalam bentuk gambar rangkaian seperti diperlihatkan dalam gambar 6.5.



Gambar 6.5 Demultiplekser 1 bit 1 ke 2

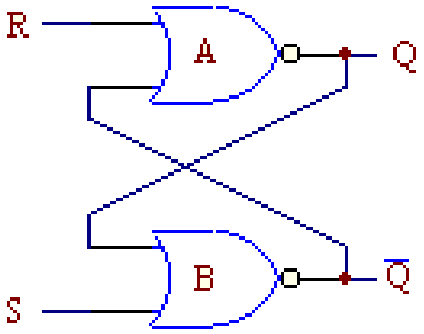
Bab 7
Rangkaian Sekuensial

7.1 Rangkaian Logika Sekuensial

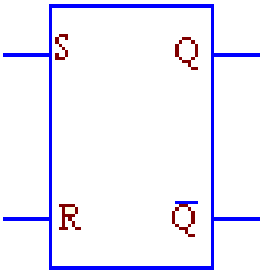
Pada rangkaian logika sekuensial, keadaan keluaran selain ditentukan oleh keadaan masukan juga ditentukan oleh keadaan keluaran sebelumnya. Hal itu menunjukkan bahwa rangkaian logika sekuensial harus mempunyai pengingat (memory), atau kemampuan untuk menyimpan informasi. Rangkaian dasar yang dapat dipakai untuk membentuk rangkaian logika sekuensial adalah latch dan flip-flop. Perbedaan latch dan flip-flop terletak pada masukan clock. Pada flip-flop dilengkapi dengan masukan clock, sedangkan pada latch tidak. Flip-flop hanya akan bekerja pada saat transisi pulsa clock dari tinggi ke rendah atau dari rendah ke tinggi, tergantung dari jenis clock yang digunakan. Transisi pulsa clock dari rendah ke tinggi di sebut transisi positif, sedangkan transisi tinggi ke rendah di sebut transisi negatif.

7.2 Set-Reset Latch (S-R Latch)

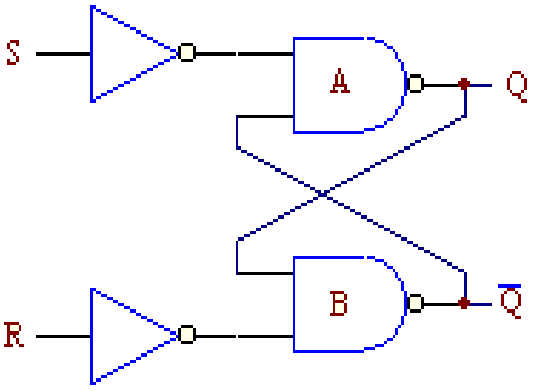
S-R latch dapat dibentuk dari dua buah gerbang NOR atau dari dua buah gerbang NAND, dengan cara mengumpanbalikkan keluaran gerbang yang satu ke salah satu masukan gerbang lainnya. S-R latch dengan gerbang NOR diperlihatkan dalam gambar 7.1 dan simbolnya diperlihatkan pada gambar 7.2, sedangkan S-R latch dengan gerbang NAND diperlihatkan dalam gambar 7.3 dan simbolnya diperlihatkan dalam gambar 7.3.



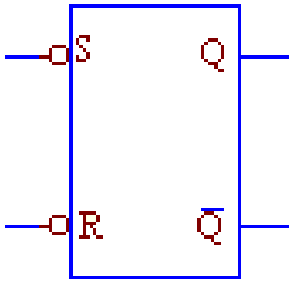
Gambar 7.29 S-R Latch dari Gerbang NOR



Gambar 7.2 Simbol S-R Latch dari Gerbang NOR



Gambar 7.3 S-R Latch dari Gerbang NAND



Gambar 7.4 Simbol S-R Latch
dengan Gerbang NAND

Untuk menganalisis rangkaian pada gambar 7.1, harus di ingat bahwa keluaran gerbang NOR adalah 0 jika salah satu masukannya dalam kondisi 1, dan keloaran gerbang NOR adalah 1 jika semua masukannya dalam kondisi 0. Sebagai titik awal diandaikan, masukan set (S) adalah 1, dan masukan reset (R) adalah 0. Karena gerbang NOR B mempunyai sebuah masukan 1, maka keluarannya ($\overline{Q}$) akan dalam kondisi 0. Keluaran $\overline{Q}$ diumpanbalikkan ke masukan gerbang NOR A, sehingga semua masukan gerbang NOR A = 0, yang mengakibatkan keluarannya (Q) dalam kondisi 1. Bila sekarang masukan S dan R di beri masukan 0, maka yang mempunyai salah satu masukan satu adalah gerbang B sehingga keluarannya tetap 0. Sedangkan kedua masukan gerbang A adalah 0, sehingga keluarannya tetap 1. Bila setelah itu masukan R diberi lojik 1 dan masukan S diberi lojik 0, maka $\overline{Q}$ akan berubah menjadi 0 karena semua masukan gerbang NOR A adalah 1. Sebaliknya keluaran NOR B akan berubah menjadi 1 karena semua masukannya 0. Bila sekarang masukan S dan R semuanya kembali di beri masukan 0, maka $\overline{Q}$ akan tetap 0 karena masukan gerbang NOR A mempunyai salah satu masukan 1. Keluaran Q juga akan tetap 1 karena semua masukan NOR A adalah 0. Bila masukan S dan R semuanya diberi lijik 1 maka keluaran Q dan $\overline{Q}$

akan 0. Dalam praktek keadaan semacam itu harus dihindari.

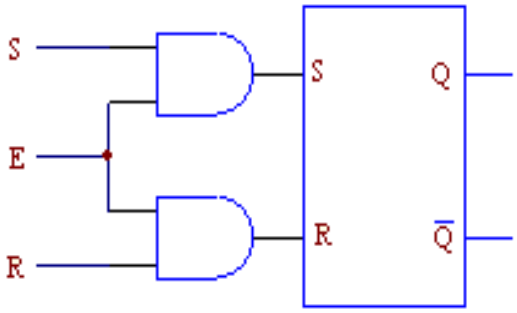
Dari analisa di atas, dapat disimpulkan bahwa bila S = 1 dan R = 0 maka keluaran Q akan menjadi 1, keadaan seperti ini disebut keadaan set. Bila S = 0 dan R = 1 maka keluaran Q akan menjadi 0, keadaan seperti ini disebut keadaan reset. Bila S = 0 dan R = 0, maka keluaran Q akan tetap seperti sebelumnya, keadaan seperti ini disebut keadaan mengingat (memory). Bila S dan R semuanya 1 maka keluaran Q akan sama dengan keluaran $\overline{Q}$, keadaan ini harus dihindari.

Rangkaian pada gambar 2.11 dapat di analisis dengan cara yang sama, dan akan menghasilkan hasil yang sama. Tabel kebenaran S-R latch diperlihatkan pada tabel 7.1.

Tabel 7.1
Tabel Kebenaran S-R Latch

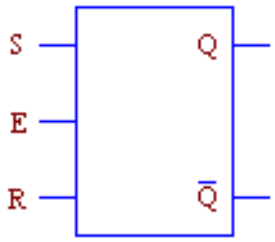
Masukan		**keluaran**		**Ket.**
S	R	Q	$\overline{Q}$	
1	0	1	0	Set
0	0	1	0	Memory
0	1	0	1	Reset
0	0	0	1	Memory
1	1	?	?	Terlarang

Untuk keperluan tertentu, S-R latch kadang-kadang dilengkapi dengan input enable. S-R latch yang dilengkapi dengan input enable hanya akan bekerja bila input enabelnya dalam kondisi 1 untuk enable aktif high, atau pada kondisi 0 untuk enable aktif low. S-R latch dengan enable dapat di buat dengan menambahkan dua buah gerbang AND yang di hubungkan dengan S-R latch tanpa enable seperti diperlihatkan dalam gambar 7.5.



Gambar 7.5 S-R Latch dengan Enabel

Simbol umum S-R latch dengan enable diperlihatkan dalam gambar 7.6, sedangkan tabel kebenarannya diperlihatatkan dalam tabel 7.2.



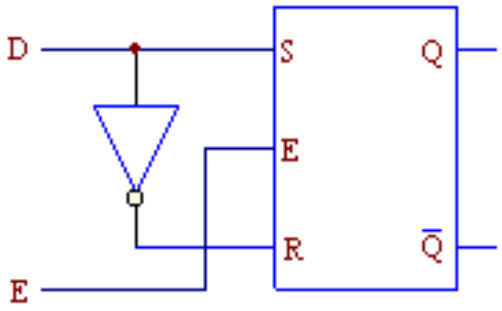
Gambar 7.6 Simbol S-R Latch dengan Enabel

Tabel 7.2
Tabel Kebenaran S-R Latch dengan Enabel

E	**S**	**R**	**Q**	$\overline{Q}$	**Ket.**
0	X	X	Q	$\overline{Q}$	Memori
1	0	0	Q	$\overline{Q}$	Memori
1	1	0	0	0	Set
1	0	1	1	1	Reset
1	1	1	?	?	Terlarang

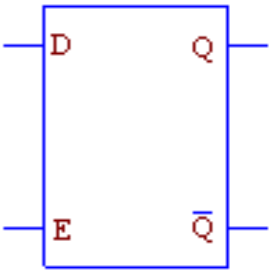
7.3 Delay Latch (D Latch)

D latch berfungsi untuk menyimpan data satu bit sementara waktu. Masukannya ada dua, yaitu masukan D dan masukan enable. D latch dapat di buat dengan menambahkan satu buah inverter ke S-R latch yang dihubungkan seperti dalam gambar 7.7, sehingga masukan S dan R selalu berlawanan.



Gambar 7.7 Rangkaian D Latch

Simbol umum D latch diperlihatkan dalam gambar 2.16.



Gambar 7.8 Simbol Umum D latch

Tabel kebenaran D latch diperlihatkan dalam tabel 7.3.

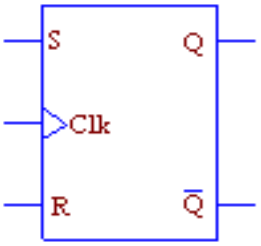
Tabel 7.3
Tabel kebenaran D Latch

Masukan		**Keluaran**	
E	D	Q_n	Q_{n+1}
0	X	0	0
0	X	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

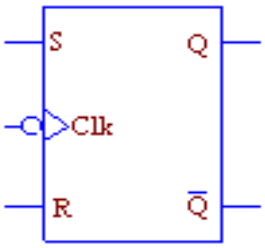
Dimana :
Q_n adalah keadaan output Q sekarang
Q_{n+1} adalah keadaan output Q selanjutnya

7.4 S-R Flip-Flop

Cara kerja rangkaian R-S flip-flop tidak jauh berbeda dengan rangkaian S-R latch. Perbedaannya adalah pada R-S flip-flop digunakan sinyal clock. Simbol S-R flip-flop dengan clock transisi positif diperlihatkan dalam gambar 2.17, sedangkan simbol flip-flop dengantransisi negatif diperlihatkan dalam gambar 7.8.

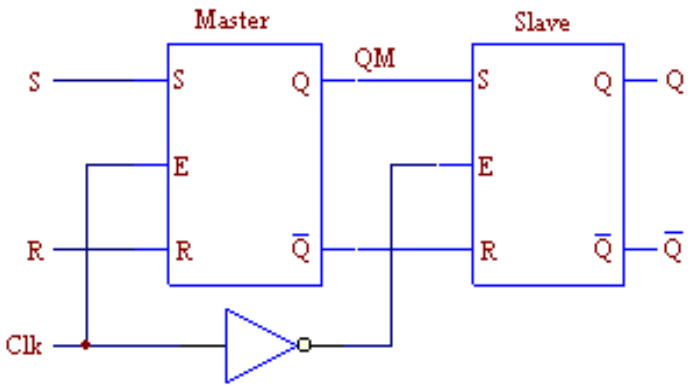


Gambar 7.8 Simbol S-R Flip-Flop dengan Clock Transisi Positif



Gambar 7.9 Simbol S-R Flip-Flop dengan Clock Transisi Negatif

S-R flip-flop dapat di buat dari dua buah S-R latch dengan satu buah inverter yang dihubungkan seperti dalam gambar 7.10.



Gambar 7.10 S-R Flip-flop dari S-R Latch

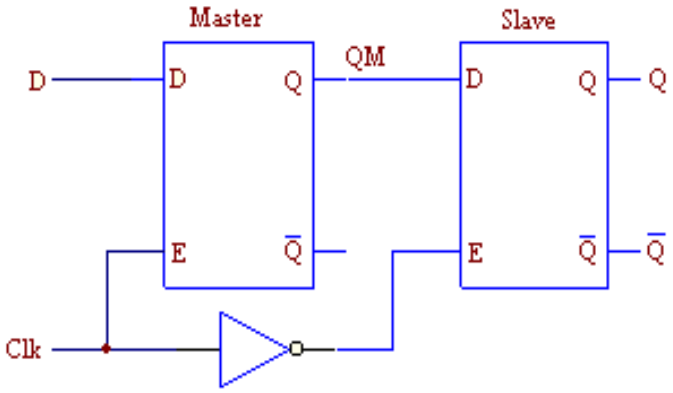
S-R latch peertama di sebut master, sedangkan S-R latch yang kedua disebut slave. Ketika masukan clock di beri lojik 1, maka S-R latch yang bekerja adalah master, sedangkan slave tidak bekerja. Data masih tersimpan pada output master (QM). Supaya QM diteruskan ke Q maka masukan clock harus diubah menjadi 0. Pada keadaan ini S-R latch yang bekerja adalah slave.
Tabel kebenarean S-R flip-flop diperlihtkan dalam tabel 7.4.

Tabel 7.4
Tabel Kebenaran S-R Flip-flop

S	R	Q_n	Q_{n+1}	Ket.
0	0	0	0	Memori
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0	X	Terlarang
1	1	1	X	

7.5 D Flip-Flop

D flip-flop dapat dibuat dari dua buah D latch yang dihubungkan seperti dalam gambar 7.11.



Gambar 7.11 D flip-flop dari D Latch

Untuk memindahkan data dari masukan D ke keluaran Q, diperlukan pulsa clock yang berubah dari 1 ke 0. Tabel kebenaran D flip-flop diperlihatkan dalam tabel 7.5.

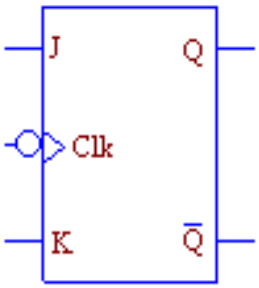
Tabel 7.5
Tabel Kebenaran D Flip-flop

D	Q_n	Q_{n+1}
0	0	0
0	1	0
1	0	1
1	1	1

7.6 JK Flip-Flop

JK flip-flop dirancang untuk menghilangkan keadaan terlarang pada R-S flip-flop. Pada keadaan terlaarang ini, JK di buat untuk selalu berlawanan dengan keadaan sebelumnya, sementara keadaan lainnya sama dengan R-S flip-flop.

Suatu JK flip-flop dengan transisi negative diperlihatkan dalam gambar 2.21.



Gambar 7.12 Simbol JK Flip-flop

Tabel kebenaran JK flip-flop diperlihatkan dalam tabel 7.6.

Tabel 7.6
Tabel Kebenaran JK Flip-flop

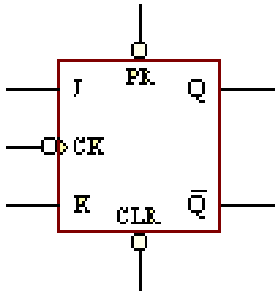
J	K	Q_n	Q_{n+1}	Ket.
0	0	0	0	Memori
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0	1	Toggle
1	1	1	0	

Dengan melihat perubahan Q_n ke Q_{n+1} dari tabel kebenaran JK flip-flop diatas, dapat di buat tabel transisi JK flip-flop seperti diperlihatkan dalam tabel 7.7.

Tabel 7.7
Tabel Transisi JK Flip-flop

Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Untuk keperluan tertentu kadang-kadang JK flip-flop dilengkapi input preset dan clear seperti diperlihatkan dalam gambar 7.14.



Gambar 7.14 JK Flip-flop dengan Preset dan Clear

Bila PR = 1 dan CLR = 0, flip-flop akan reset. Bila PR = 0 dan CLR = 1, flip-flop akan set. Bila PR = CL R = 0, flip-flop berada pada kondisi terlarang. Pada ketiga keadan preset dan clear di atas, masukan J, K, dan Clock tidak berpengaruh. Supaya flip-flop dapat bekerja, preset dan clear harus berada pada kondisi 1.

Bab 8
Pencacah

8.1 Counter (Pencacah)

Pencacah adalah suatu rangkaian logika sekuensial yang dapat berfungsi untuk menghitung jumlah pulsa yang masuk dan akan dinyatakan dalam bentuk biner. Pada umumnya counter di bentuk dari beberapa buah flip-flop yang jumlahnya disesuaikan dengan kebutuhan. Menurut cara pemberian pulsa clock, pencacah dapat di bagi ke dalam :

1. Pencacah tak sinkron
2. Pencacah sinkron

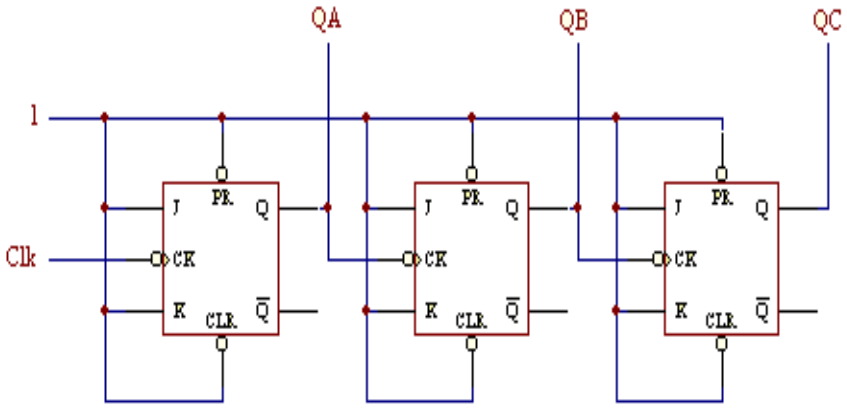
Sedangkan menurut urutan hitungan yang terbentuk pada keluarannya, pencacah dapat di bagi ke dalam :

1. Pencacah naik
2. Pencacah turun
3. Pencacah naik-turun

8.2 Pencacah Tak Sinkron

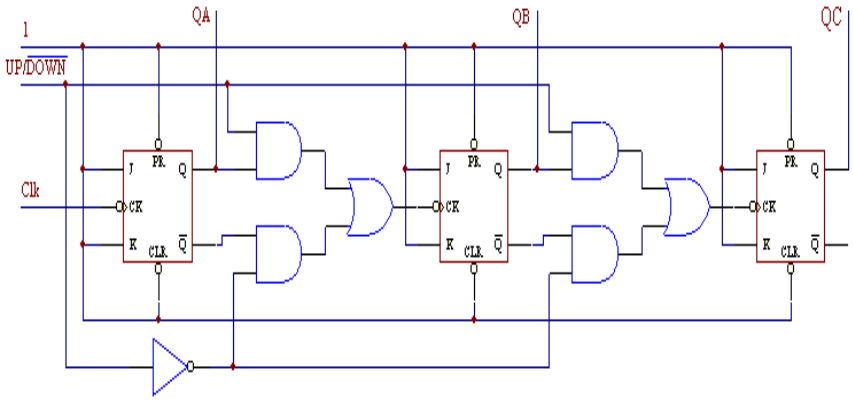
Pada pencach tak sinkron, pulsa clock hanya diberikan kepada masukan clock salah satu flip-flop, sedangkan untuk sinyal clock flip-flop yang lainnya di ambil dari keluaran flip-flop sebelumnaya. Sinyal clock dapat diambil dari keluaran Q atau $\overline{Q}$ tergantung dari jenis masukan clock yang digunakan dan urutan cacahan yang diinginkan. Clock transisi positif, bila dihubungkan dengan Q akan menghasilkan pencacah turun, sedangkan bila dihubungkan dengan $\overline{Q}$ akan menghasilkan pencacah naik. Begitu pula untuk clock transisi negatif, bila dihubungkan dengan Q akan menghasilkan pencacah naik, sedangkan bila dihubungkan dengan $\overline{Q}$ akan menghasilkan pencacah turun.

Dari berbagai jenis pencacah, yang paling sederhana adalah yang melakukan cacahan mengikuti urutan bilangan biner. Suatu pencacah biner dengan n bit terdiri dari n buah flip-flop dapat menghitung dalam bilangan biner dari 0 sampai $(2^n - 1)$. Gambar 8.1 memperlihatkan rangkaian pencacah naik biner tak sinkron 3 bit yang dapat menghitung dari 0 sampai 7. Flip-flop yang digunakan adalah JK flip-flop dengan masuka clock transisi negatif. Supaya didapatkan pencacah naik. Masukan clock dihibungkan dengan Q JK flip-flop sebelumnya. Masukan J dan K di beri lojik 1 supaya flip-flop berada pada kondisi toggle.



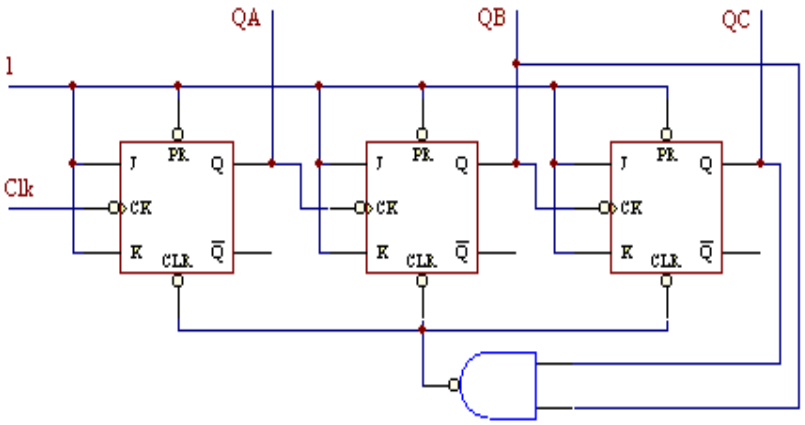
Gambar 8.1 Pencacah Biner Naik Tak Sinkron 3 bit

Pencacah biner tak sinkron naik-turun diperlihatkan dalam gambar 2.24. Dalam perencanaan pencacah jenis ini, selain flip-flop, juga diperlukan beberapa buah gerbang logika dasar untuk menguntrol supaya pencacah dapat bekerja mencacah naik atau turun.



Gambar 8.2 Pencacah Biner
Tak Sinkron Naik-turun 3 bit

Di dalam sistem digital, ada kalanya diperlukan hitungan dengan basis N yang bukan merupakan pangkat bilangan 2. Pencacah seperti ini disebut modulo N. Modulo adalah jumlah cacahan tertinggi yang dapat dilakukan oleh suatu pencacah. Untuk membuat pencacah jenis ini yaitu dengan cara menghubungkan keluaran pencacah biner tak sinkron yang berada pada keadaan 1 pada saat cacahan ke N, ke masukan clear tiap flip-flop, melalui gerbang AND untuk clear aktif high, atau dengan gerbang NAND untuk clear aktif low. Sebagai contoh untuk membuat pencacah modulo 6 dari gambar 8.2. Pada cacahan ke 6, keluaran yang berada pada keadaan 1 adalah QB dan QC. Flip-flop yang digunakan menggunakan jenis clear aktif low, maka QB dan QC dihubungkan ke clear tiap-tiap flip-flop dengan menggunakan gerbang NAND. Dengan demikianpada cacahan ke 6, pencacah akan reset sehingga semua keluaran pencacah kembali ke keadaan 0. Pencacah modulo 6 diperlihatkan dalam gambar 8.3.



Gambar 8.3 Pencacah Modulo 6

8.3 Pencacah Sinkron.

Pulsa clock pada pencacah sinkron diberikan secara serempak. Masukan clock tiap-tiap flip-flop dihubungkan bersama. Perencanaan pencacah sinkron berawal dari penentuan N cacahan tertinggi dan diakhiri dengan realisasi rangkaian. Prosedur itu meliputi beberapa langkah berikut:

1. Penentuan N cacahan tertinggi
2. Penentuan jumlah flip-flop yang digunakan dengan rumus $2^n = N$
3. Penentuan jenis flip-flop yang dipakai
4. Penulisan tabel keadaan pencacah dengan melihat tabel transisi flip-flop yang digunakan
5. Penulisan dan penyederhanaan persamaan masukan
6. Realisasi rangkaian

Pencacah sinkron modulo 6 diperlihatkan dalam gambar 8.4. Banyaknya flip-flop yang digunakan adalah $2^n = 6$, maka diperoleh n = 3 buah. Jenis flip-flop yang digunakan adalah JK flip-flop. Setelah ditentukan jenis flip-flop yang digunakan dapat dibuat tabel keadaan seperti yang diperlihatkan dalam tabel 8.1.

Tabel 8.1
Tabel Keadaan Pencacah Sinkron Modulo 6

Sekarang			**Selanjutnya**			**Masukan**					
QC	QB	QA	QC	QB	QA	JC	KC	JB	KB	JA	KA
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	0	0	0	X	1	0	X	X	1

Dari tabel 8.1, setelah melalui penyederhanaan dengan metoda Karnough-Map, diperoleh persamaan masukan J dan K sebagai berikut :

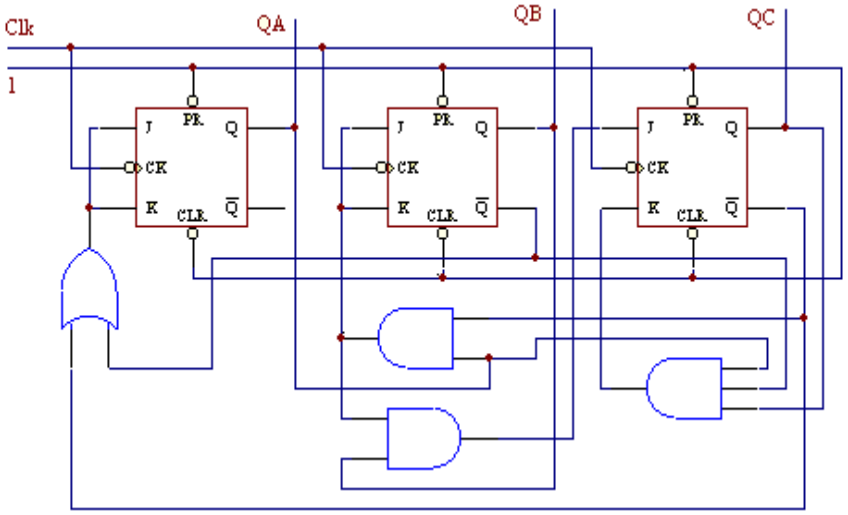
$$J_A = K_A = \overline{Q_B} + \overline{Q_C}$$

$$J_B = K_B = Q_A \overline{Q_C}$$

$$J_C = Q_A Q_B \overline{Q_C}$$

$$K_C = Q_A \overline{Q_B} Q_C$$

Persamaan di atas dapat direalisasikan dengan gambar rangkaian seperti yang diperlihatkan dalam gambar 8.4.



Gambar 8.4 Pencacah Sinkron Modulo 6

Bab 9
Multivibrator

9.1 Multivibrator

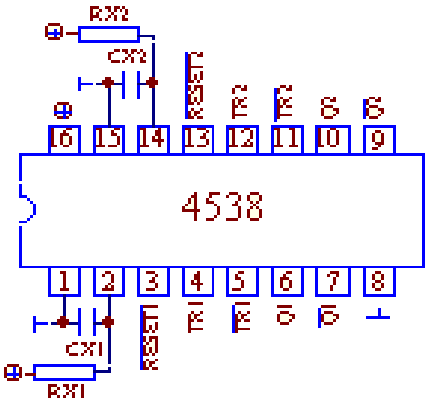
Dalam perencanaan suatu sistem digital, kadang-kadang diperlukan suatu alat yang dapat membangkitkan pulsa untuk masukan clock, atau suatu alat yang berfungsi sebagai debuonce switch. Alat seperti ini digolongkan pada sub sistem yang di sebut multivibrator. Di lihat dari cara kerjanya, multivibrator di bagi ke dalam tiga jenis, yaitu :

1. Moltivibrator monostabil
2. Multivibrator bistabil
3. Multivibrator astabil

9.1.1 Multivibrator Monostabil

Multivibrator monostabil adalah penggetarganda yang keluarannya hanya mempunyai satu keadaan stabil. Diperlukan satu pulsa trigger untuk membuat perangkat ini berada pada kondisi goyah. Setelah beberapa saat sesuai dengan jangka waktu yang telah ditetapkan berada dalam kondisi goyah, ia akan kembali ke kondisi stabil. Pada saat kondisi goyah, multivibrator jenis ini tidak dapat di beri pulsa trigger ulang, sampai kembali ke kondisi stabil. Multivibrator monostabil dapat di pakai debuance switch, untuk memanjangkan atau memendekkan pulsa, atau sebagai unsur tunda.

Salah satu IC multivibrator monostabil adalah dari jenis cmos type 4538. IC ini berisi dua buah multivibrator monostabil. Diagram koneksi IC 4538 diperlihatkan dalam gambar 9.1.



Gambar 9.1 Diagram Koneksi IC 4538

Transisi dari tinggi ke rendah di jalan masuk $\overline{TR}$ bila jalan masuk TR adalah rendah, atau transisi dari rendah ke tinggi di jalan masuk TR bila jalan masuk $\overline{TR}$ sedang tinggi, akan membangkitkan denyut positif di jalan keluar Q dan denyut negative di jalan keluar $\overline{Q}$ bila jalan masuk reset menjadi tinggi. Rendah di jalan masuk reset memaksa jalan keluar $\overline{Q}$ rendah, jalan keluar Q tinggi dan melarang sebarang denyut berikutnya sampai jalan masuk reset menjadi tinggi.

Lebar pulsa ditentukan oleh dua komponen ekstern yaitu R_X dan C_X dengan rumus :

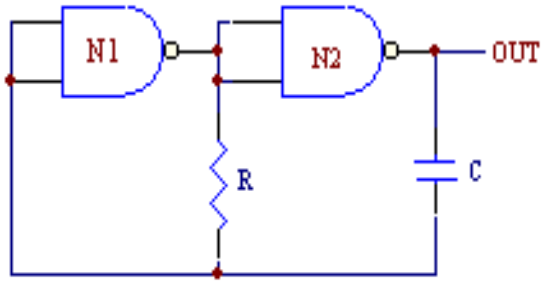
$$T = R_X \cdot C_X$$

9.1.2 Multivibrator Bistabil

Multivibrator bistabil adalah multivibrator yang mempunyai dua kondisi stabil. Diperlukan dua pulsa trigger untuk menyelesaikan satu daur operasi, dari kkondisi stabil 1 ke kondisi stabil 2, dan kembali lagi ke kondisi stabil 1. Multivibrator jenis ini dapat di buat dari JK flip-flop yang dioperrasikan dalam keadaan toggle.

9.1.3 Multivibrator Astabil

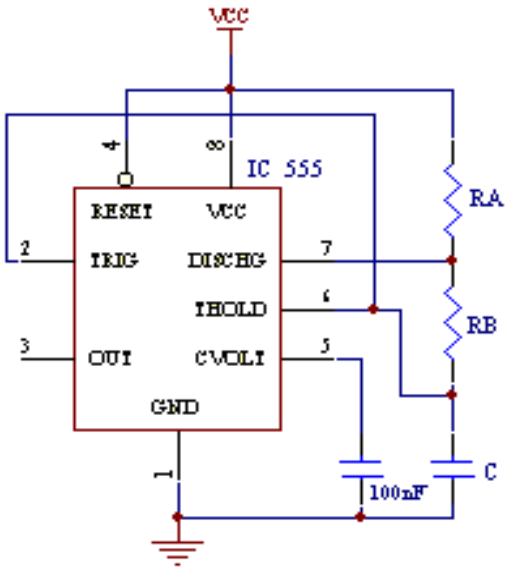
Multivibrator astabil adalah multivibrator yang tiak mempunyai keadaan stabil. Multivibratr jenis ini digunakan sebagai pembangkit pulsa. Multivibrator astabil yang di bentuk dari dua buah gerbang NAND diperlihatkan dalam gambar 9.2.



Gambar 9.2 Multivibrator Astabil dari Gerbang NAND

Rangkaian dalam gambar 9.2 bekerja atas dasar pengisian dan pengosongan kapasitor. Misalnya pada waktu dihidupkan, keluaran N2 dalam keadaan tinggi, C meneruskannya ke masukan N1 sehingga keluaran N1 rendah. C mulai dikosongkan melalui R, tegangan pada masukan N1 turun. Saat mencapai tegangan ½ V_{CC}, keluaran N2 berubah menjadi rendah, C meneruskannya ke masukan N1, sehiungga keluaran N1 menjadi tinggi. C di isi oleh keluaran N1 melalui R. Saat mencapai tegangan ½ V_{CC}, keluaran N2 tinggi kembali. Begitulah seterusnya kejadian ini akan berulang, sehingga pada keluaran akan dihasilkan suatu getaran yang mirip gelombang persegi. Frekuensi keluaran ditentukan oleh nilai R dan C.

Selain dari gerbang NAND, multivibrator astabil dapat juga di buat dari IC 555, seperti yang diperlihatkan dalam gambar 9.3.



Gambar 9.3 Multivibrator Astabil dari IC 555

Waktu charge dan discharge juga frekuensi tidak tergantung pada tegangan supply, tetapi hanya tergantung pada RA, RB, dan C. Waktu charge dapat di terntukan oleh:

$t_1 = 0,693\ (RA+RB)C$

sedangkan waktu dischargenya adalah :

$t_2 = 0,693\ RB.C$

Jadi total periodanya adalah :

$T = t_1 + t_2$

$= 0,693\ (RA+2RB)C$

dengan frekuensi keluaran :

$f = 1/T = 1,44/\{(RA+2RB)C\}$

Duty cycle-nya adalah :

$D = RB/(RA+2RB)$

Bab 10
Peraga Tujuh Segmen

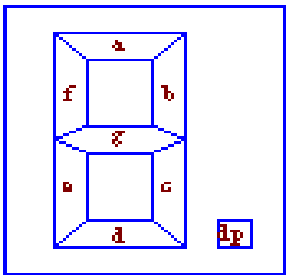
10.1 Peraga Seven Segmen

Prinsip dasar antara led dengan peraga tujuh segmen adalah sama. Peraga tujuh segmen terdiri dari tujuh buah led yang disusun membentuk suatu karakter angka. Peraga tujuh segmen ini sering digunakan pada alat-alat ukur digital, dimana parameter yang terukur dapat langsung dibaca.

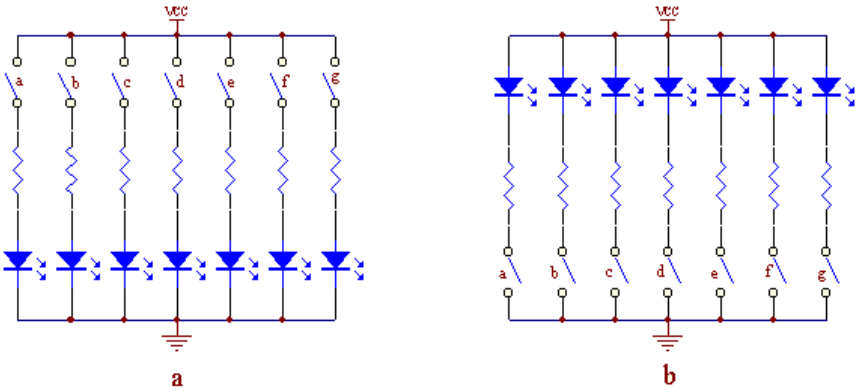
Ada dua jenis peraga tujuh segmen dilihat dari hubungan bersamanya (common) :

1. Anoda bersama (Common Anoda)
   Semua terminal anoda dari led yang membentuk tujuh segmen digabung menjadi satu, dan terminal-terminal katoda sebagai masukan. Terminal bersama selalu dihubungkan pada sumber tegangan positif dari catu daya, sehingga arus masuk melalui common dan keluar melalui terminal katoda.
2. Katoda Bersama (Common Katoda)
   Semua terminal katoda dari led yang membentuk tujuh segmen digabung menjadi satu, dan terminal-terminal anoda sebagai masukan. Terminal bersama selalu dihubungkan pada ground dari catu daya.

Gambar 10.1 merupakan peraga tujuh segmen dan gambar 10.2 merupakan rangkaian ekuivalent tujuh segmen common katoda dan common anoda.



Gambar 10.1 Peraga Tujuh Segmen



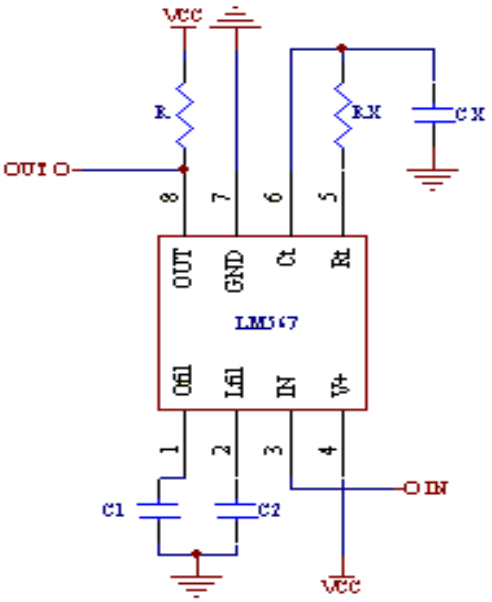
Gambar 10.2 Rangkaian Ekuivalen Tujuh Segmen
(a). Common Katoda (b). Common Anoda

10.2 Tone Decoder

Tone decoder berfungsi untuk mengkonversikan sinyal audio dengan frekuensi tertentu sesuai dengan talaannya ke dalam bentuk sinyal digital. LM 567 merupakan salah satu IC tone decoder yang umum digunakan. Komponen ini mampu

memberikan respon terhadap suatu frekuensi yang sesuai dengan talaan RC-nya dalam waktu kurang dari satu detik. Pada waktu terdapat sinyal masukan , bila frekuensinya sesuai, keluaran IC ini akan berada pada kondisi rendah. Sebaliknya bila frekuensinya tidak sesuai, keluaran IC ini akan berada pada kondisi tinggi.

Diagram koneksi IC LM 567 diperlihatkan dalam gambar 10.3.



Gambar 10.3 Diagram Koneksi IC LM 567

Frekuensi talaan ditentukan oleh Rx dan Cx dengan rumus sebagai berikut:

$$Fo = \frac{1}{1{,}1.Rx.Cx}$$

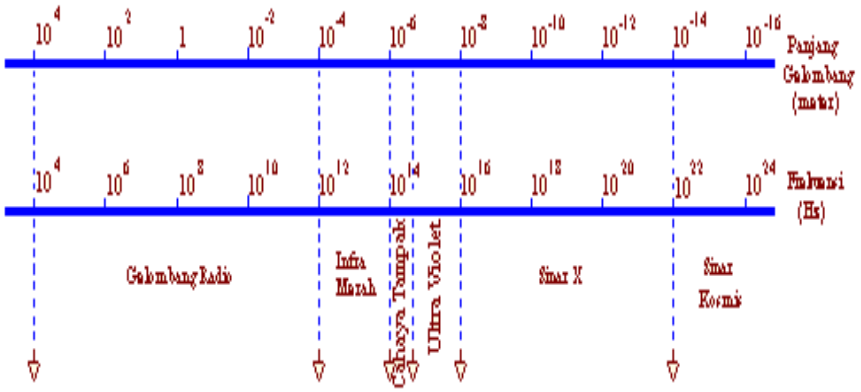
Sedangkan bandwidthnya adalah

$$BW = \frac{1070.Vi}{Fo.C2}$$

Fo dalam satuan Hz
Rx dalam satuan KOhm
Cx dalam satuan uF
Vi dalam satuan V(rms), Vi≤200mV(rms)

10.3 Infra Merah

Cahaya infra merah merupakan cahaya yang tidak tampak. Jika di lihat dengan spektroskop cahaya, cahaya infra merah akan tampak pada spectrum elektromagnetik dengan panjang gelombang di atas panjang gelombang cahaya merah. Spektrum elektromagnetik diperlihatkan dalam gambar 2.33. Dengan panjang gelombang ini maka cahaya infra merah akan tidak tampak oleh mata, namun radiasi panas yang ditimbulkan masih dapat terasa. Cahaya infra merah walaupun mempunyai panjang gelombang yang sangat panjang, tetapi tetap tidak dapat menembus bahan-bahan yang tidak dapat melewatkan cahaya yang nampak sehingga cahaya infra merah tetap mempunyai karakteristik seperti halnya cahaya yang tampak oleh mata.



Gambar 10.4 Spektrum Gelombang Elektromagnetis

Penggunaan cahaya infra merah sebagai media komunikasi sangat menguntungkan, karena sampai saat ini belum ada aturan yang membatasi penggunaan cahaya ini. Namun demikian modulasinya harus menggunakan sinyal carrier dengan frekuensi tertentu, untuk memperjauh jarak transmisi. Untuk transmisi data yang menggunakan media udara sebagai perantara, biasanya menggunakan frekuensi carrier antara 30 KHz sampai dengan 40 KHz.

Sinyal infra merah dapat dihasilkan dari LED khusus yaitu LED infra merah. Sedangkan sebagai sensornya dapat menggunakan photo dioda atau photo transistor. Sensor ini akan mengubah energi cahaya, dalam hal ini energi cahaya infra merah menjadi sinyal listrik. Semakin besar intensitas cahaya merah yang diterima, maka sinyal pulsa listrik yang dihasilkan akan semakin baik. Dalam praktek, sinyal infra merah yang di terima intensitasnya sangat kecil, sehingga perlu dikuatkan. Selain itu agar tidak terganggu oleh sinyal cahaya lain, maka sinyal listrik yang dihasilkan oleh sensor infra merah harus di filter terlebih dahulu pada frekuensi sinyal carrier.

10.4 Transistor Sebagai Saklar

1. Kondisi Cut Off Transistor

Pada gambar 10.4 memperlihatkan yang di rangkai common emitor dimana resisitansi beban R_L di anggap terhubung seri. Tegangan total yang terdapat pada ujung-ujung rangkaian seri sama dengan tegangan catunya (V_{CC}) dan diberi notasi V_{RL} dan V_{CE} . Menurut hukum kirchoff :

$$V_{CC} = V_{CE} + V_{R\ L}$$

Arus kolektor I_C mengalir melalui R_L dan besar tegangan V_R adalah I_C . R_L sehingga :

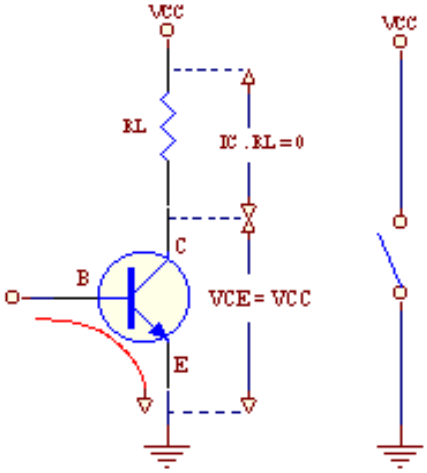
$$V_{CC} = V_{CE} + (I_C.R_L)$$

Misalnya basis memperoleh bias reverse yang sedemikian besar sehingga memutuskan (cut off) arus kolektor, dan dalam keadaan ini arus kolektor sama dengan nol.

Dari persamaan diatas diperoleh :

$I_c \cdot R_L = 0$ sehingga $V_{CC} = V_{CE}$

Bila transistor di anggap sebagai saklar, maka pada keadaan ini saklar tersebut berada dalam keadaan terbuka.



Gambar 10.4 Rangkaian Common Emittor

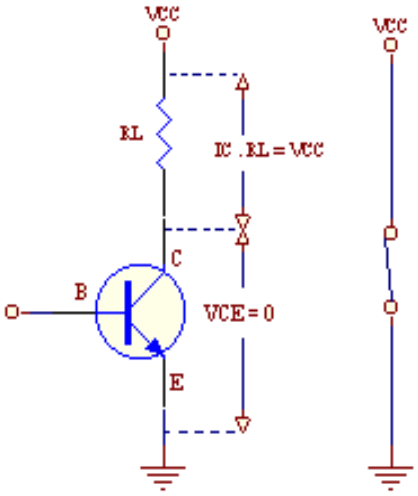
2 Kondisi Saturasi Transistor

Bila sekarang basis di beri bias forward sampai pada titik dimana seluruh tegangan V_{CC} muncul sebagai drop tegangan pada R_L, maka pada keadaan ini :

$$V_{RL} = I_C \cdot R_L = V_{CC}$$
$$V_{CC} = V_{RL} + V_{CE}$$
$$V_{CE} = V_{CC} - V_{RL} = 0$$

Dengan demikian bila I_c diperbesar pada suatu titik dimana seluruh tegangan V_{CC} muncul pada R_L, maka tidak tersisa tegangan pada kolektor. Keadaan seperti ini dikatakan sebagai kondisi saturasi dari transistor

tersebut. Dan jika transistor di anggap sebagai saklar, maka saklar tersebut dalam keadaan tertutup.



Gambar 10.5 Transistor
Dalam Kondisi Saturasi

Daftar Pustaka

1. Fadeli AR., Dasar-Dasar Elektronika, Fisika UGM, Yogjakarta, 1998.
2. Fajar Purwanto HM. dkk. , Elektronika Dasar, Fisika UPI, Bandung, 1999.
3. Nur Muhammad, Elektronika Dasar, Fisika ITS, Surabaya, 1984.
4. Sutrisno, Elektronika : Teori dan Penerapannya, Jilid 1 dan 2, Fisika ITB, Bandung, 1986.