# Buku Ajar

# Teknik Digital

***Untuk Mahasiswa Teknik Elektro***
***Dan Program Studi Serumpun***

Dr. Muchlas, M.T.

Universitas Ahmad Dahlan
Yogyakarta, 2020

# Buku Ajar

# Teknik Digital

***Untuk Mahasiswa Teknik Elektro***
***Dan Program Studi Serumpun***

Dr. Muchlas, M.T.

Universitas Ahmad Dahlan
Yogyakarta, 2020

# KATA PENGANTAR

Alhamdulillah puji syukur dipanjatkan kepada Allah Subhanahu Wa Ta'ala yang telah memberikan petunjukNya sehingga penulisan Buku Ajar Teknik Digital ini dapat diselesaikan dengan baik.

Seluruh materi dalam buku ini dikelompokkan ke dalam tujuh bab yang telah disusun secara urut dan sistematis sehingga pembaca dapat memperoleh pengetahuan yang utuh terhadap rangkaian digital. Ketujuh bab itu adalah Rangkaian dan Sistem Digital, Sistem Bilangan dan Sistem Kode, Gerbang Logika Dasar dan Aljabar Boole, Rangkaian Logika Kombinasi, Logika Kombinasi Dalam Kemasan IC, Rangkaian Logika Sekuensi, Pencacah dan Register. Selain diperuntukkan bagi mahasiswa program studi Teknik Elektro, buku ajar ini juga dapat digunakan oleh para mahasiswa program studi lain yang serumpun seperti Teknik Informatika, Teknik Komputer, Sstem Informasi, Ilmu Komputer, dan bahkan oleh para mahasiswa Ilmu Fisika dan Pendidikan Fisika yang mengambil mata kuliah sejenis dengan Teknik Digital.

Melalui buku ajar ini, mahasiswa diharapkan dapat memahami secara komprehensif dasar-dasar analisis dan perancangan rangkaian logika yang merupakan landasan bagi pengembangan kompetensi utama di lingkungan program studi Teknik Elektro atau program studi-program studi serumpunnya.

Kepada semua pihak yang telah membantu penyusunan buku ajar ini diucapkan terimakasih. Semoga bantuan tersebut menjadi amal sholeh dan mendapat imbalan pahala dari Allah Subhanahu Wa Ta'ala.

Dengan berbagai kekurangannya, buku ajar ini diharapkan dapat memberikan manfaat sesuai dengan fungsinya. Masukan-masukan dari siapapun sangat dinanti demi perbaikan buku ajar ini.

Yogyakarta, September 2020

Muchlas

# DAFTAR ISI

Halaman

# DAFTAR TABEL

# DAFTAR GAMBAR

## KOMPETENSI DASAR I

Mahasiswa dapat menjelaskan aspek-aspek penting dalam Teknik Digital

## TUJUAN PEMBELAJARAN I

Agar mahasiswa dapat:
1. mendefinisikan rangkaian dan sistem digital
2. menyebutkan contoh-contoh rangkaian dan sistem digital dalam bidang elektronika
3. membedakan ciri-ciri rangkan dan sistem digital
4. menjelaskan representasi besaran digital dengan tegangan listrik, diode, transistor, saklar, dan indikator LED (*light emitting diode*)

## GARIS BESAR MATERI I

Pada bagian awal ini anda akan mempelajari pengertian rangkaian dan sistem digital. Pengetahuan tentang kedua istilah tersebut sangat penting karena tujuan deskripsi seluruh materi yang ada pada buku ini adalah agar anda memperoleh pengetahuan tentang dasar-dasar perancangan dan analisis rangkaian digital atau rangkaian logika. Kegiatan perancangan rangkaian digital/logika pada dasarnya merupakan kegiatan mengimplementasikan atau merealisasikan rangkaian digital/logika atas dasar adanya karakteristik atau watak yang diinginkan. Sedangkan analisis rangkaian digital/logika merupakan kegiatan melakukan identifikasi watak dari suatu rangkaian digital/logika yang telah ada. Karena titik berat pembahasan buku ini adalah pada kegiatan perancangan dan analisis rangkaian digital/logika, maka pengertian rangkaian digital/logika itu sendiri perlu dipahami sejak awal. Perancangan dan analisis rangkaian digital/logika tidak akan memberikan manfaat yang signifikan dalam kehidupan sehari-hari jika tidak dikaitkan dengan aplikasi rangkaian digital/logika. Pembahasan aplikasi rangkaian digital/logika akan terkait erat dengan sistem digital yang dalam kehidupan sehari-hari berwujud berbagai peralatan sistem elektronika seperti *personal computer* (PC). Oleh karena itu anda akan diperkenalkan pula dengan pengertian sistem digital dalam tinjauan sistem elektronika.

Melalui bagian awal ini anda juga akan diperkenalkan dengan berbagai representasi besaran digital dalam bidang elektronika. Dalam rangkaian atau sistem digital, data-data yang diproses berbentuk diskrit, artinya data itu hanya memiliki dua keadaan saja. Jika data digital itu berbentuk tegangan listrik, maka rangkaian atau sistem digital hanya akan menganggap bahwa tegangan yang akan diproses itu termasuk level logika rendah atau level logika tinggi. Materi-materi pada bagian ini akan memperkenalkan pada anda berbagai representasi data digital, baik representasi yang umum digunakan pada bagian input maupun output suatu rangkaian logika. Pemahaman terhadap representasi besaran ini menjadi penting karena dalam perancangan dan analisis rangkaian digital/logika, besaran digital dinyatakan dalam simbol 0 dan 1, sedangkan dalam kenyataannya besaran digital tersebut direpresentasikan dalam berbagai bentuk seperti keadaan saklar yang tertutup atau terbuka, dan lampu yang menyala atau padam. Dengan memahami berbagai representasi besaran digital dalam bidang elektronika tersebut, pada satu sisi anda diharapkan memiliki kemampuan mengidentifikasi besaran digital dalam kehidupan sehari-hari dan dapat menyatakannya dalam simbol biner 0 dan 1, dan pada sisi lain anda memiliki kemampuan mengimplementasikan besaran digital yang dinyatakan dalam simbol 0 dan 1 ke dalam bentuk representasi yang sesuai. Dengan kemampuan itu, pada akhirnya anda akan memperoleh kemudahan di dalam melakukan proses perancangan maupun analisis rangkaian digital/logika.

# BAB I
# RANGKAIAN DAN SISTEM DIGITAL

## A. Rangkaian Digital

Rangkaian digital atau sering pula disebut dengan rangkaian logika adalah kesatuan dari elemen-elemen logika yang membentuk suatu fungsi **pemrosesan sinyal digital**.


**Gambar 1. Penjelasan pengertian rangkaian digital/logika**

Pada gambar 1 terlihat bahwa baik input maupun output rangkaian digital merupakan sinyal digital dan outputnya memberikan fungsi pemrosesan sinyal digital.

## B. Sistem Digital

Sistem digital adalah kesatuan dari beberapa rangkaian digital/logika, dan elemen/gerbang logika untuk suatu tujuan pengalihan tenaga/energi. Contoh sistem digital dalam bidang komputasi adalah komputer digital. Komputer digital merupakan kesatuan dari beberapa rangkaian digital/logika, dan elemen/gerbang logika untuk tujuan pengalihan tenaga dalam bidang komputasi. *Keyboard* merupakan periferal input (bagian luar sistem komputer) yang berfungsi menerima data dari pengguna. Untuk memasukkan data, *keyboard* harus ditekan sehingga memperoleh energi mekanik dan oleh sistem komputer data itu diproses, selanjutnya melalui bagian outputnya, data dikirim ke monitor yang merupakan periferal output dalam bentuk energi cahaya atau ke *printe*r dalam bentuk energi mekanik. Demikian pula *disk drive*, *harddisk drive*, CD-ROM *drive* melayani pemasukan data ke sistem komputer dengan memutar *disk* dalam

bentuk energi mekanik dan oleh sistem komputer diproses serta dikeluarkan ke periferal output dalam bentuk energi cahaya pada monitor atau energi mekanik pada *printer*.


**Gambar 2. Komputer sebagai suatu sistem elektronika digital dalam bidang komputasi**

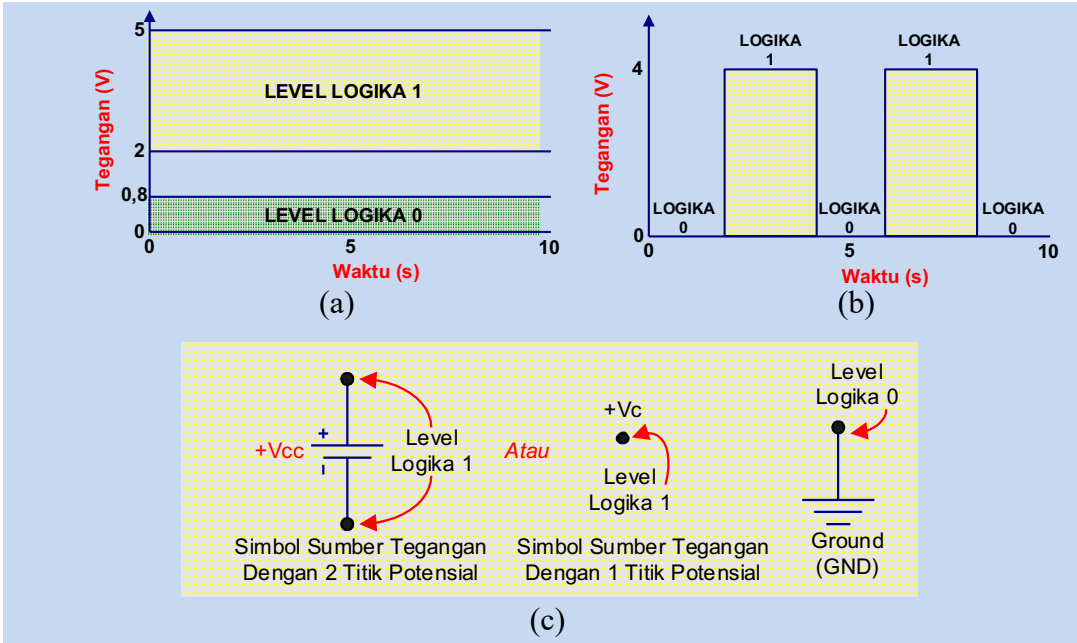
Selain komputer, contoh sistem digital yang lain adalah *handphone* yang merupakan sistem digital dalam bidang komunikasi dan *storage oscilloscope* yang merupakan sistem digital dalam bidang instrumentasi. Berikut ini adalah tabel yang dapat digunakan untuk membedakan antara rangkaian dan sistem digital.

**Tabel 1. Perbedaan rangkaian dan sistem digital**

| Rangkaian Digital | Sistem Digital |
|---|---|
| 1. Merupakan bagian dari sistem digital, bagian-bagiannya terdiri atas beberapa elemen/gerbang logika.<br>2. Outputnya membentuk fungsi pemrosesan sinyal digital.<br>3. Input dan outputnya berupa sinyal digital. | 1. Bagian-bagiannya terdiri atas beberapa rangkaian digital, gerbang logika, dan komponen elektronika lainnya.<br>2. Outputnya merupakan fungsi pengalihan tenaga.<br>3. Input dan outputnya berupa suatu tenaga/energi. |

## C. Representasi Besaran Digital

Besaran digital merupakan besaran yang sifatnya diskrit, yakni besaran yang hanya memiliki dua keadaan saja. Dalam analisis dan perancangan sistem/rangkaian digital, kedua keadaan tersebut dinamakan keadaan biner yakni keadaan rendah atau level logika 0 dan keadaan tinggi atau level logika 1. Dalam tegangan listrik, besaran digital diwujudkan dalam berbagai bentuk seperti ditunjukkan pada gambar 3.

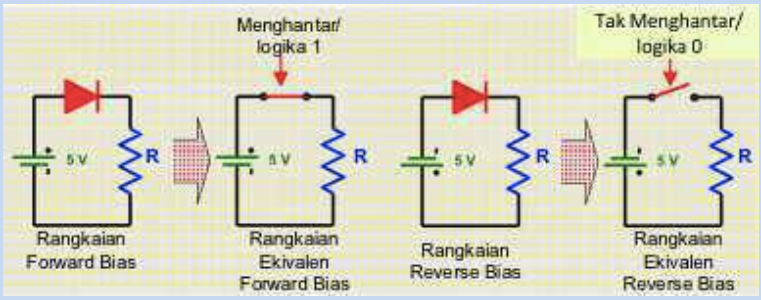



**Gambar 3. Representasi besaran digital dengan tegangan listrik**

**Lihat juga:**

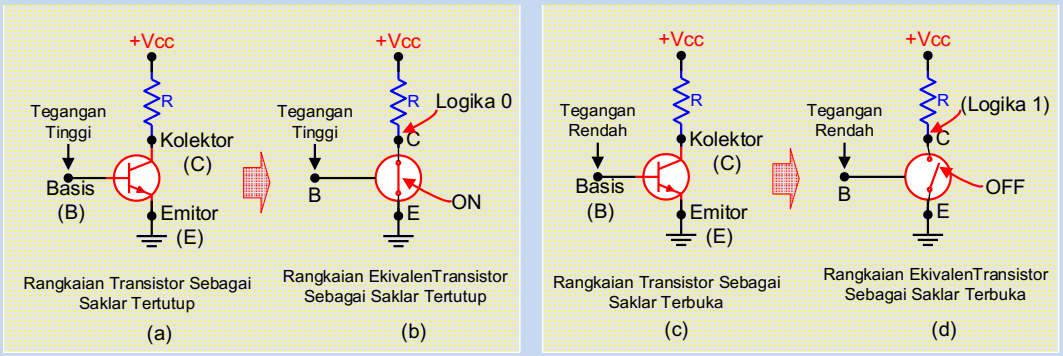
- Ragam tegangan rangkaian digital untuk TTL dan CMOS, definisi tegangan masukan untuk logika tinggi dan rendah ($V_{IH}$ dan $V_{IL}$), serta tegangan keluaran ($V_{OH}$ dan $V_{OL}$) pada **Lampiran 1**
- *Noise margin* pada IC digital pada **Lampiran 2**

Bentuk besaran digital yang lain dalam bidang elektronika adalah keadaan piranti diode dan transistor. Diode dalam keadaan menghantar (*conducting*) menunjukkan keadaan level logika 1 dan diode dalam keadaan tak menghantar (*nonconducting*) menunjukkan keadaan level logika 0. Diode merupakan piranti elektronika yang terbuat dari bahan semikonduktor yang memiliki dua buah elektrode yakni anode dan katode. Sifat diode adalah jika anode diberi tegangan yang lebih tinggi dari pada katodenya maka diode dalam keadaan menghantar. Keadaan tak menghantar akan terjadi jika tegangan pada kedua elektrodenya sama atau tegangan anode lebih rendah dari tegangan katodenya.



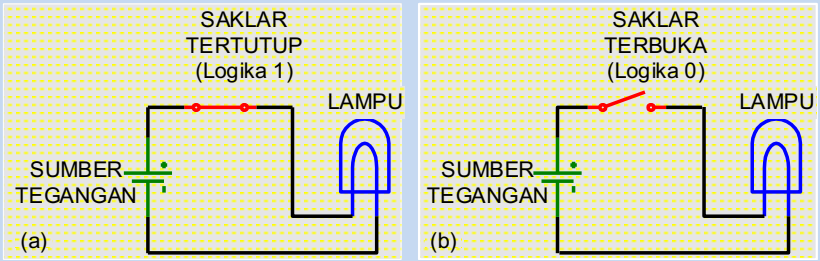

**Gambar 4. Representasi besaran digital dengan diode**

Jika besaran digital direpresentasikan dengan sifat transistor, maka ketika transistor dalam keadaan jenuh dapat dianggap merepresentasikan level logika 1 karena bersifat seperti saklar tertutup, dan apabila transistor berada pada keadaan mati dapat dianggap merepeperesentasikan level logika 0 karena bersifat seperti saklar terbuka. Transistor merupakan piranti elektronika dengan tiga elektrode yakni basis, kolektor dan emitor yang dapat berfungsi sebagai penguat (amplifier) maupun saklar. Dalam hal ini basis sebagai elektrode pengendali yang akan menentukan keadaan transistor. Untuk memberikan keadaan transistor seolah-olah seperti saklar tertutup antara elektrode emitor dan kolektor, maka pada basis harus diberi tegangan tinggi, dan sebaliknya jika diinginkan keadaan transistor seolah-olah seperti saklar terbuka antara emitor dan kolektor, maka tegangan basis harus rendah.



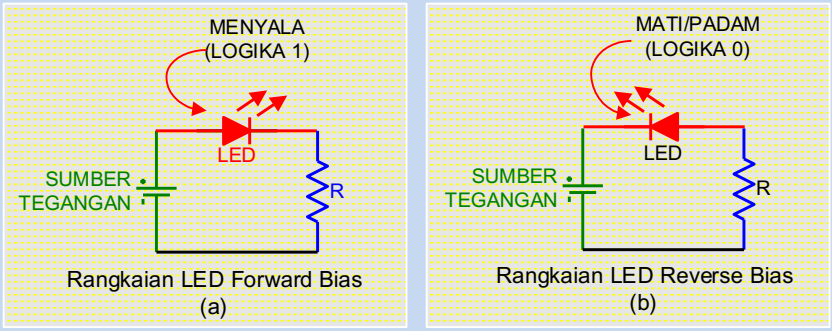

**Gambar 5. Representasi besaran digital dengan transistor**

Representasi lain adalah dengan saklar yang sebenarnya, dalam hal ini saklar tertutup merepresentasikan level logika 1 dan saklar terbuka merepresentasikan level logika 0. Biasanya representasi level logika dengan saklar ini digunakan pada bagian input dari suatu rangkaian logika, sedangkan level logika pada bagian outputnya direpresentasikan dengan menggunakan indikator LED (light emitting diode).




**Gambar 6. Representasi besaran digital dengan saklar**

LED merupakan suatu piranti elektronika yang memiliki dua buah elektrode yakni anode dan katode. Jika tegangan antara anode dan katode lebih besar dari tegangan ambang (*threshold*), maka LED akan menyala, dan sebaliknya akan mati. Keadaan LED yang menyala merepresentasikan level logika 1, sedangkan keadaan LED mati merepresentasikan level logika 0.




**Gambar 7. Representasi level logika menggunakan LED**

Jadi, representasi besaran digital dalam bidang elektronika khususnya pada rangkaian/sistem digital kebanyakan dinyatakan dalam bentuk sesuai tabel berikut ini.

**Tabel 2. Representasi besaran digital pada bidang elektronika**

| Level Logika 0 | Level Logika 1 |
|---|---|
| Tegangan listrik 0 s.d. 0,8 V | Tegangan listrik 2 s.d. 5 V |
| Titik Potensial Referensi 0 (ground) | Titik Potensial Catu Daya +Vcc |
| Diode dengan *reverse bias* | Diode dengan *forward bias* |
| Transistor dalam keadaan mati (cut-off) | Transistor dalam keadaan jenuh (saturated) |
| Saklar dalam keadaan terbuka | Saklar dalam keadaan tertutup |
| Lampu atau LED dalam keadaan padam | Lampu atau LED dalam keadaan menyala |

**Lihat juga:**

- Watak *floating input* IC TTL gerbang logika pada **Lampiran 3**
- Pengantarmukaan TTL/CMOS pada **Lampiran 4**
- Sifat keluaran gerbang TTL *open collector* dan *totem pole* serta teknik pemanfaatannya pada **Lampiran 5**
- *Fanin* dan *fanout* pada IC gerbang logika pada **Lampiran 6**
- *Propagation delay* pada gerbang logika pada **Lampiran 7**
- Peredaman *noise* dengan kapasitor bypass pada **Lampiran 8**

## D. Soal Latihan

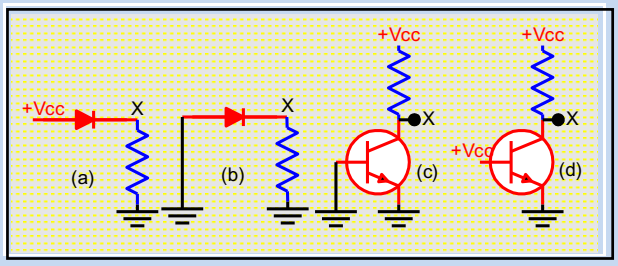
1. Apa perbedaan pokok rangkaian dan sistem digital dari sisi tugasnya! Jelaskan pula perbedaan keduanya dari sisi besaran yang diproses!

2. Apa yang dimaksud dengan elemen logika dan berikan contohnya!
3. Sebuah rangkaian logika memiliki input dan output seperti pada gambar berikut ini.


**Gambar 8. Diagram blok untuk soal nomor 3 pada Bab I**

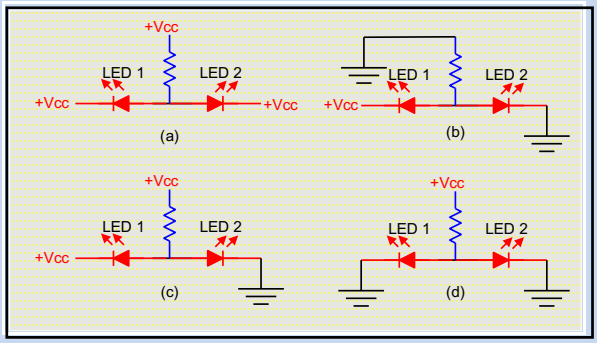
Jika pada input-1 diberi tegangan sebesar +0,1 volt, pada input-2 diberi tegangan +4 volt, serta memberikan tegangan pada outputnya sebesar 3,5 volt, tuliskan keadaan level logika pada input-1, input-2, dan output rangkaian logika tersebut!

4. Representasikan data digital 16-bit 1010110011110111 dalam bentuk gelombang kotak!
5. Tentukan keadaan level logika dari gambar a, b, c, dan d berikut ini serta berikan alasan mengapa anda menentukan keadaan tersebut!




**Gambar 9. Representasi besaran digital dengan diode dan transistor untuk soal nomor 5 pada Bab I**

6. Perhatikan rangkaian LED berikut ini:




**Gambar 10. Representasi besaran digital dengan LED untuk soal nomor 6 pada Bab I**

Tentukan keadaan level logika LED 1 dan LED 2 serta jelaskan alasannya untuk gambar a, b, c, dan d!

7. Implementasikan rangkaian-rangkaian dalam bentuk gambar blok yang terdiri dari tiga buah input dan sebuah output dengan ketentuan input dan output rangkaian direpresentasikan dengan:
   a. INPUT-1, INPUT-2, INPUT-3 dengan SAKLAR, OUTPUT dengan LED
   b. INPUT-1, INPUT-2, INPUT-3 dengan SAKLAR, OUTPUT dengan Transistor.
8. Soal nomor 8 dan nomor 9 adalah jenis pilihan ganda. Untuk mengerjakan soal-soal tersebut, pilih satu jawaban yang tepat di antara opsi jawaban yang tersedia!
   LED akan berfungsi sebagai peraga logika tinggi dalam suatu rangkaian dengan kondisi:
   a. Tegangan anode sama dengan tegangan katode
   b. Tegangan katode lebih positif dibandingkan tegangan anode
   c. Tegangan anode lebih positif dibandingkan tegangan katode
   d. Terpasang dalam konfigurasi *reverse bias*
   e. Tegangan anode dan tegangan katode sama-sama positif
9. Berikut ini merupakan sistem digital, kecuali:
   a. *Handphone*
   b. Multimeter *moving coil*
   c. Timbangan buah dengan peraga tujuh segmen
   d. Pengendali robot
   e. *Storage oscilloscope*

## KOMPETENSI DASAR II

1. Mahasiswa memahami konsep berbagai sistem bilangan dalam teknik digital
2. Mahasiswa memahami konsep berbagai sistem kode dalam teknik digital.

## TUJUAN PEMBELAJARAN II

Agar mahasiswa dapat:
1. mengetahui arti sistem bilangan desimal
2. mengetahui arti sistem bilangan biner dalam konteks sistem desimal
3. mengetahui arti sistem bilangan oktal dalam konteks sistem desimal
4. mengetahui arti sistem bilangan heksadesimal dalam konteks sistem desimal
5. mengubah berbagai sistem bilangan ke sistem desimal
6. mengubah sistem bilangan desimal ke berbagai sistem bilangan
7. mengubah sistem biner ke sistem oktal dan heksadesimal dengan cepat
8. mengubah sistem oktal dan heksadesimal ke sistem biner dengan cepat
9. mengubah sistem desimal ke sistem kode BCD atau sebaliknya
10. mengubah sistem desimal ke sistem kode XS-3 atau sebaliknya
11. mengubah sistem desimal ke sistem kode Gray atau sebaliknya
12. mengubah bilangan, huruf dan simbol ke dalam kode ASCII berparitas ganjil dan genap atau sebaliknya

## GARIS BESAR MATERI II

Pada bab I telah dipelajari bahwa sistem maupun rangkaian digital/logika bekerja dengan menggunakan besaran digital yang disimbolkan dengan 0 untuk level logika rendah, dan 1 untuk level logika tinggi. Oleh karena hanya menggunakan dua buah simbol, suatu sistem/rangkaian digital pada dasarnya bekerja dengan menggunakan sistem bilangan biner. Agar diperoleh kemudahan dalam melakukan analisis dan perancangan sistem/rangkaian digital, perlu dipahami terlebih dahulu sistem bilangan biner. Pada bab ini anda akan mempelajari sistem bilangan biner yang mencakup pengertian sistem bilangan itu sendiri, konversinya ke sistem bilangan lain, dan konversi sebaliknya dari sistem bilangan lain ke sistem bilangan biner. Selain itu, melalui bab ini anda akan mempelajari pula sistem bilangan oktal dan heksadesimal. Penjelasan terhadap ketiga sistem bilangan tersebut akan dilakukan dalam konteks sistem bilangan desimal, karena desimal merupakan sistem bilangan yang paling sering digunakan dalam kehidupan sehari-hari.

Bagian akhir dari bab ini menjelaskan beberapa sistem kode yang digunakan untuk menyajikan data digital. Terdapat berbagai macam sistem kode seperti desimal dikode biner atau *binary-coded decimal* (BCD), *gray*, excess-3, kode peraga 7-segmen, dan ASCII . Jika penyajian data hanya menggunakan sistem bilangan, maka penyajian tersebut sangat terbatas yakni hanya dapat menyajikan data dalam bentuk bilangan positif saja. Dengan menggunakan sistem kode dapat disajikan berbagai macam jenis data seperti bilangan, simbol maupun huruf ke dalam besaran digital. Selain itu, dengan sistem kode juga dapat disajikan bilangan positif maupun bilangan negatif, dan bahkan bilangan pecahan dengan titik desimal. Melalui bab ini anda akan mempelajari pula berbagai sistem kode yang ada mencakup konsep dasar maupun cara pembangkitannya.

# BAB II
# SISTEM BILANGAN DAN SISTEM KODE

## A. Sistem Bilangan

### 1. Bilangan Desimal

Desimal merupakan sistem bilangan dengan basis 10, artinya digit/angka yang digunakan untuk menyajikannya berjumlah 10 buah yakni: 0, 1, 2, 3, 4, 5, 6, 7, 8, dan 9, serta setiap digit penyusunnya memiliki bobot kepangkatan $10^n$ dengan n merupakan bilangan bulat positif dan negatif.

Contoh:

Bilangan $(5346)_{10}$ atau $5346_{10}$ memiliki arti:

$$5346_{10} = \mathbf{5} \times 10^3 + \mathbf{3} \times 10^2 + \mathbf{4} \times 10^1 + \mathbf{6} \times 10^0$$

Angka-angka penyusun bilangan desimal disebut **digit**. Digit yang menempati posisi paling kiri yakni 5 memiliki bobot terbesar sehingga dinamakan **Most Significant Digit (MSD)** sedangkan digit paling kanan dinamakan **Least Significant Digit (LSD)** yang berarti digit dengan bobot terkecil. Untuk bilangan desimal bulat 5346, hubungan antara digit-digit penyusunnya dengan bobotnya dapat disajikan seperti berikut ini.

| | $10^3$ | $10^2$ | $10^1$ | $10^0$ | ← Bobot bilangan desimal bulat |
|---|---|---|---|---|---|
| Bilangan desimal → | **5** | **3** | **4** | **6** | |
| | **MSD** | | | **LSD** | |

Contoh lain, bilangan desimal 0,25 memiliki arti:

$$0{,}25 = \mathbf{2} \times 10^{-1} + \mathbf{5} \times 10^{-2}$$

Berdasarkan contoh tersebut, hubungan digit-digit penyusunnya dengan bobotnya pada bilangan desimal pecahan 0,25 dapat disajikan seperti berikut ini.

| | | | $10^{-1}$ | $10^{-2}$ | ← Bobot bilangan desimal pecahan |
|---|---|---|---|---|---|
| Bilangan desimal → | 0 | , | **2** | **5** | |
| | | | **MSD** | **LSD** | |

### 2. Bilangan Biner

Biner merupakan sistem bilangan dengan basis 2, artinya dalam sistem ini digit yang digunakan berjumlah 2 buah yakni 0 dan 1, serta setiap digit penyusunnya (dinamakan bit) memiliki bobot kepangkatan $2^n$ dengan n merupakan bilangan bulat positif dan negatif.

Contoh:

Bilangan biner $(10110)_2$ atau $10110_2$, dalam konteks bilangan desimal memiliki arti:

$$10110_2 = \underline{\mathbf{1} \times 2^4} + \underline{\mathbf{0} \times 2^3} + \underline{\mathbf{1} \times 2^2} + \underline{\mathbf{1} \times 2^1} + \underline{\mathbf{0} \times 2^0} = 22_{10}$$

$$16_{10} \quad 0 \quad 4_{10} \quad 2_{10} \quad 0$$

Bit dengan bobot terbesar dinamakan **Most Significant Bit (MSB)** dan bit paling kanan dengan bobot terkecil dinamakan **Least Significant Bit (LSB).** Bobot bilangan biner ditunjukkan sebagai berikut:

| | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ | ← Bobot bilangan biner bulat |
|---|---|---|---|---|---|---|
| Bilangan biner → | 1 | 0 | 1 | 1 | 0 | |
| | **MSB** | | | | **LSB** | |

Untuk bilangan biner pecahan, bobot bit MSB dimulai dari $2^{-1}$. Contoh bilangan biner $0{,}101_2$ memiliki arti:

$$0{,}101_2 = \mathbf{1} \times 2^{-1} + \mathbf{0} \times 2^{-2} + \mathbf{1} \times 2^{-3} = 0{,}5 + 0 + 0{,}125 = 0{,}525_{10}$$

dan hubungan antara bit-bit penyusunnya dengan bobotnya adalah sebagai berikut.

| | | $2^{-1}$ | $2^{-2}$ | $2^{-3}$ | ← Bobot bilangan biner pecahan |
|---|---|---|---|---|---|
| Bilangan biner → | 0 , | 1 | 0 | 1 | |
| | | **MSB** | | **LSB** | |

### 3. Bilangan Oktal

Oktal merupakan sistem bilangan dengan basis 8. Dalam sistem ini digit yang digunakan berjumlah 8 buah yakni: 0, 1, 2, 3, 4, 5, 6, dan 7, serta bobot yang dimiliki oleh setiap digit penyusunnya adalah kepangkatan $8^n$ dengan n merupakan bilangan bulat positif dan negatif.

Contoh:

Bilangan oktal $(215)_8$ atau $215_8$ dalam konteks bilangan desimal memiliki arti:

$$215_8 = \underline{\mathbf{2} \times 8^2} + \underline{\mathbf{1} \times 8^1} + \underline{\mathbf{5} \times 8^0} = 141_{10}$$

$$128_{10} \quad 8_{10} \quad 5_{10}$$

Setiap bilangan penyusun pada sistem oktal disebut **digit** dan bobotnya ditunjukkan sebagai berikut:

| | $8^2$ | $8^1$ | $8^0$ | ← Bobot bilangan oktal bulat |
|---|---|---|---|---|
| Bilangan oktal → | 2 | 1 | 5 | |
| | **MSD** | | **LSD** | |

Untuk bilangan oktal pecahan bobotnya merupakan kepangkatan negatif dari 8, contoh bilangan oktal $0,14_8$ memiliki arti:

$$0,14_8 = 1 \times 8^{-1} + 4 \times 8^{-2} = \frac{1}{8} + \frac{4}{64} = \frac{2}{16} + \frac{1}{16} = \frac{3}{16}$$

dan hubungan antara digit-digit penyusunnya dengan bobotnya adalah:

| | | $8^2$ | $8^1$ | ← Bobot bilangan oktal pecahan |
|---|---|---|---|---|
| Bilangan oktal → | 0 , | 1 | 4 | |
| | | **MSD** | **LSD** | |

**4. Bilangan Heksadesimal**

Heksadesimal merupakan sistem bilangan dengan basis 16, artinya simbol digit yang digunakan berjumlah 16 yakni 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, dan F, serta bobot setiap digit penyusunnya adalah kepangkatan $16^n$ dengan n merupakan bilangan bulat positif dan negatif. **Contoh:**
Bilangan $(BE5)_{16}$ atau $BE5_{16}$ dalam konteks sistem desimal memiliki arti:

$$\underset{\downarrow}{\mathbf{11} \times 16^2} + \underset{\downarrow}{\mathbf{14} \times 16^1} + \underset{\downarrow}{\mathbf{5} \times 16^0} = 3045_{10}$$
$$2816_{10} \quad 224_{10} \quad 5_{10}$$

Bobot digit-digit heksadesimal pada contoh ditunjukkan sebagai berikut.

| | $16^2$ | $16^1$ | $16^0$ | ← Bobot bilangan heksadesimal bulat |
|---|---|---|---|---|
| Bilangan heksadesimal → | B | E | 5 | |
| | **MSD** | | **LSD** | |

Untuk bilangan heksadesimal pecahan, bobot terbesar dimulai dari $16^{-1}$. Contoh bilangan heksadesimal pecahan adalah $0,C8_{16}$. Bilangan heksadesimal tersebut memiliki nilai desimal:

$$0,C8_{16} = 12 \times 16^{-1} + 8 \times 16^{-2} = \frac{12}{16} + \frac{8}{256} = \frac{24}{32} + \frac{1}{32} = \frac{25}{32}$$

dan hubungan antara digit-digit penyusunnya dengan bobotnya dapat dituliskan sebagai berikut:

| | | $16^{-1}$ | $16^{-2}$ | ← Bobot bilangan heksadesimal pecahan |
|---|---|---|---|---|
| Bilangan heksadesimal → | 0 , | C | 8 | |
| | | **MSD** | **LSD** | |

**B. Konversi Sistem Biner, Oktal, dan Heksadesimal ke Sistem Desimal**

Uraian tentang pengertian sistem bilangan biner, oktal, dan heksadesimal dalam konteks bilangan desimal yang telah dikemukakan di depan pada dasarnya merupakan penjelasan cara melakukan konversi dari semua sistem bilangan ke sistem bilangan desimal.

Prinsip dari konversi ini adalah menjumlahkan nilai dari setiap digit/bit suatu bilangan yang telah dikalikan dengan bobotnya.

**1. Konversi Sistem Biner ke Desimal**

**a. Bilangan Bulat**

Contoh:

$1101_2$ = .....$_{10}$

$1101_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 8+4+0+1 = 13_{10}$

**b. Untuk Bilangan Pecahan**

Contoh:

$1101,11_2$= .....$_{10}$

$1101,11_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2}$

$1101,11_2 = 8 + 4 + 0 + 1 + 0,5 + 0,25 = 13,75_{10}$

**2. Konversi Sistem Oktal ke Desimal**

**a. Bilangan Bulat**

Contoh:

$154_8$ = .....$_{10}$

$154_8 = 1 \times 8^2 + 5 \times 8^1 + 4 \times 8^0 = 64+40+4 = 108_{10}$

**b. Bilangan Pecahan**

Contoh:

$154,67_8$ = .....$_{10}$

$154,67_8 = 1 \times 8^2 + 5 \times 8^1 + 4 \times 8^0 + 6 \times 8^{-1} + 7 \times 8^{-2}$

$154,67_8 = 64 + 40 + 4 + 6/8 + 7/64 = 92,86_{10}$

**3. Konversi Sistem Heksadesimal ke Desimal**

**a. Bilangan Bulat**

Contoh:

$5B_{16}$ = .....$_{10}$

$5B_{16} = 5 \times 16^1 + 11 \times 16^0 = 80+11 = 91_{10}$

**b. Bilangan Pecahan**

Contoh:

$A7,C1_{16}$ = .....$_{10}$

$A7,C1_{16} = 10 \times 16^1 + 7 \times 16^0 + 12 \times 16^{-1} + 1 \times 16^{-2}$

$A7,C1_{16} = 160 + 7 + 12/16 + 1/256 = 167,75_{10}$

**C. Konversi Sistem Desimal ke Sistem Biner, Oktal, dan Heksadesimal**

Konversi sistem desimal ke sistem bilangan lain dapat dilakukan dengan dua cara yakni metode nilai digit dan metode bagi. Pemilihan terhadap metode yang digunakan tergantung pada kebiasaan seseorang.

## 1. Konversi Sistem Desimal ke Sistem Biner

### a. Metode Nilai Digit

Konversi sistem desimal ke sistem biner dengan metode ini dilakukan dengan cara menuliskan terlebih dahulu bobot bilangan biner pecahan dan bobot bilangan biner bulat misalnya dimulai dari $2^{-3}$ sampai dengan $2^{8}$. Penulisan bobot-bobot bilangan biner tersebut diurutkan dari bobot terkecil sampai dengan bobot terbesar dari kanan ke kiri.

Contoh:

$21_{10}$ = .....$_2$
$227_{10}$ = .....$_2$
$227{,}625_{10}$ = .....$_2$

Penyelesaian dari contoh di atas dengan metode nilai digit adalah sebagai berikut:

**Batas desimal**

| | $2^8$ | $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ | $2^{-1}$ | $2^{-2}$ | $2^{-3}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 256 | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | 0,5 | 0,25 | 0,125 |
| $21_{10}$ = | | | | | 1 | 0 | 1 | 0 | 1 | | | |
| $227_{10}$ = | | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | | | |
| $227{,}625_{10}$= | | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |

**Gambar 11. Metode nilai digit**

Dengan cara di atas, maka dapat diperoleh hasil konversi bilangan desimal 21 ke biner sebagai berikut:

$21_{10} = 10101_2$

Untuk memeriksa kebenaran hasil konversi, dapat dilakukan konversi balik dari bilangan biner $10101_2$ ke sistem desimal sebagai berikut:

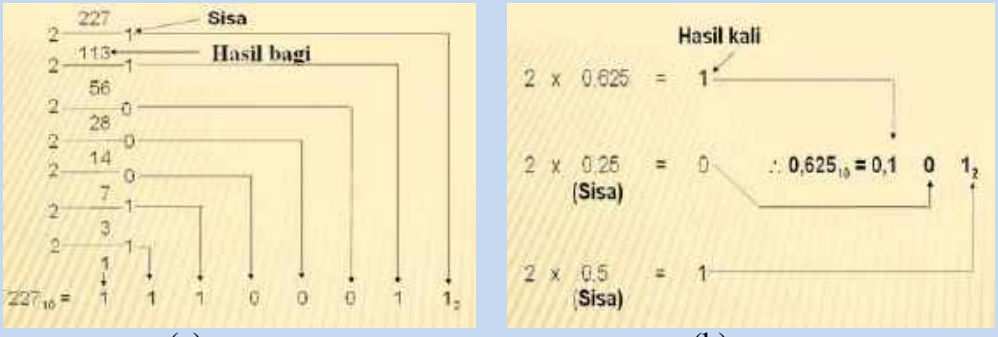
$10101_2 = 1x2^4 + 0x2^3 + 1x2^2 + 0x2^1 + 1x2^0$
$10101_2 = 16 + 0 + 4 + 0 + 1 = 10101_2 = 21_{10}$

Dengan cara yang sama, yakni metode nilai digit dapat diperoleh hasil konversi bilangan desimal 227 dan 227,625 sebagai berikut:

$227_{10} = 11100011_2$
$227{,}625_{10} = 11100011{,}101_2$

**b. Metode Bagi Dua**

Konversi bilangan desimal ke sistem biner dengan metode bagi dua dapat dilakukan dengan cara seperti ditunjukkan pada gambar 12.

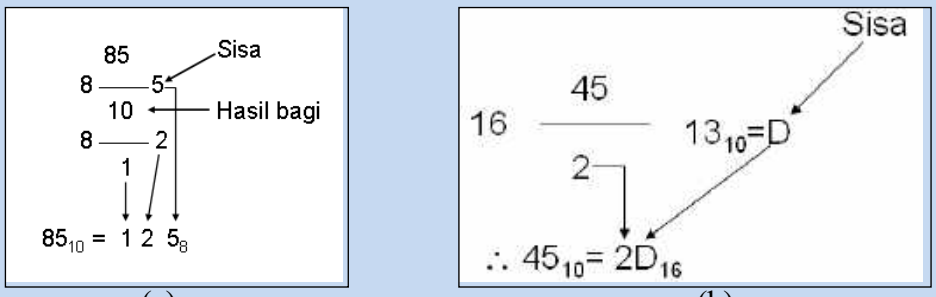



(a) (b)

**Gambar 12. Konversi bilangan desimal ke sistem biner menggunakan metode bagi dua: (b) bulat, (b) pecahan**

**2. Konversi Sistem Desimal ke Sistem Oktal dan Heksadesimal**

Konversi sistem desimal ke sistem oktal dan ke sistem heksadesimal dapat dilakukan dengan metode bagi, dan dalam hal ini faktor pembaginya adalah 8 untuk oktal dan 16 untuk heksadesimal. Contoh: konversikan bilangan desimal 85 ke sistem oktal dan sistem heksadesimal! Penyelesaiannya dapat dilakukan dengan cara:




(a) (b)

**Gambar 13. Konversi bilangan desimal bulat ke sistem (a) oktal dan (b) heksadesimal**

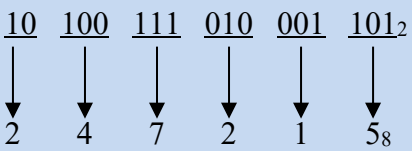
**D. Konversi Sistem Biner ke Sistem Oktal dan Heksadesimal**

**1. Konversi Sistem Biner ke Sistem Oktal**

Konversi sistem biner ke sistem oktal dilakukan dengan cara mengelompokkan bilangan biner menjadi kelompok 3 bit dimulai dari LSB. Selanjutnya setiap kelompok bilangan biner tersebut dikonversi ke sistem oktal.

**Contoh:**
$10100111010001101_2$= .....$_8$

Penyelesaian konversinya adalah sebagai berikut:

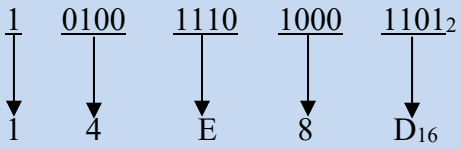



Jadi, $10100111010001101_2 = 247215_8$.

**2. Konversi Sistem Biner ke Sistem Heksadesimal**

Konversinya dilakukan dengan cara mengelompokkan bilangan biner menjadi kelompok 4 bit dimulai dari LSB. Selanjutnya setiap kelompok bilangan biner tersebut dikonversi ke sistem heksadesimal**. Contoh:**
$10100111010001101_2$= .....$_{16}$

Proses konversinya sebagai berikut:




Sehingga $10100111010001101_2$= $14E8D_{16}$

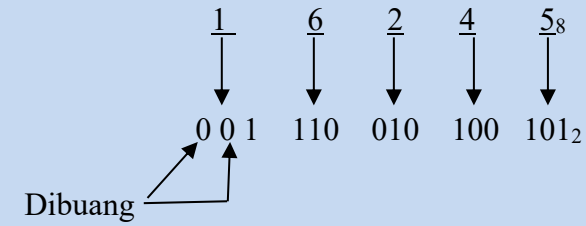
**E. Konversi Sistem Oktal dan Heksadesimal ke Sistem Biner**

**1. Konversi Sistem Oktal ke Sistem Biner**

Konversi suatu bilangan oktal menjadi bilangan biner dilakukan dengan cara mengubah setiap bilangan oktal ke dalam bilangan biner 3-bit, dan jika ditemukan bit 0 pada MSB, bit tersebut dibuang.

Contoh:

$16245_8$= .....2




Dengan demikian $16245_8$=1 110 010 100 $101_2$

### 2. Konversi Sistem Heksadesimal ke Sistem Biner

Konversi ini dilakukan dengan cara mengubah setiap bilangan heksadesimal ke dalam bilangan biner 4-bit. Jika ditemukan bit 0 pada MSB, bit tersebut dibuang. **Contoh:**

$62B1C_{16}$= .....$_2$

| 6 | 2 | B | 1 | $C_{16}$ |
|---|---|---|---|---|
| 0110 | 0010 | 1011 | 0001 | $1100_2$ |

Dibuang

Jadi, $62B1C_{16}$= 110 0010 1011 0001 $1100_2$.

## F. Sistem Kode

Data yang diproses di dalam sistem digital disusun dengan menggunakan kode tertentu. Terdapat berbagai macam sistem kode seperti desimal dikode biner atau *binary-coded decimal* (BCD), *gray*, excess-3, kode peraga 7-segmen, dan ASCII

Kode-kode tersebut disusun dengan suatu cara menggunakan bilangan biner yang membentuk kelompok tertentu. Kelompok bilangan biner yang membentuk suatu kode dibedakan penyebutannya. Kode biner **4-bit** dinamakan **nibble**, contoh 1101, 1010, dan1001. Kode biner **8-bit** dinamakan **byte**, contoh: 10011100, dan 10101010. Dalam hal ini 1 byte=8-bit, 1Kilo byte=1KB=1024 byte=$2^{10}$ byte. Kode biner **16-bit** dinamakan **word**, contoh: 1001110010101010, dan kode biner **32-bit** dinamakan **double word.**

### 1. Kode BCD *(Binary-Coded Decimal)*

Kode BCD atau Desimal Dikode Biner sering ditulis dalam bentuk BCD-8421 menggunakan kode biner 4-bit untuk merepresentasikan masing-masing digit desimal dari suatu bilangan.

Contoh: Tulis dalam bentuk kode BCD-8421 bilangan desimal; 529!

Jawab:

| 5 | 2 | 9 | ← DESIMAL |
|---|---|---|---|
| 0101 | 0010 | 1001 | ← BCD-8421 |

Dalam sistem kode BCD terdapat 6 buah kode yang tidak dapat digunakan (*invalid code*) yakni: 1010, 1011, 1100, 1101, 1110, dan 1111, sehingga hanya ada 10 buah kode BCD yang valid yakni kode-kode untuk merepresentasikan bilangan desimal 0 s.d. 9. Untuk lebih memahami kode BCD, coba perhatikan konversi kode BCD ke sistem desimal berikut ini!

a. Ubah 0110 1000 0011 $1001_{BCD}$ ke sistem desimal!
Jawab:

0110 1000 0011 $1001_{BCD}$ ⟶ Sistem BCD
6 8 3 9 ⟶ Sistem Desimal

b. Ubah 0111 1100 $0001_{BCD}$ ke sistem desimal!
Jawab:

0111 1100 $0001_{BCD}$ ⟶ Sistem BCD
7 1 ⟶ Sistem Desimal
⟶ Kode yang tidak dapat digunakan (*invalid)*, menunjukkan terjadi kesalahan pada kode BCD.

**2. Kode Excess-3 (XS-3)**

Sistem kode lain yang mirip dengan BCD adalah Excess-3. Untuk menyusun kode XS-3 dari suatu bilangan desimal, masing-masing digit dari suatu bilangan desimal yang akan dikode dengan XS-3, ditambah dengan 3 desimal, kemudian hasilnya dikonversi seperti cara pada konversi BCD.
Contoh: Tulis dalam bentuk kode XS-3 bilangan desimal 12!

Jawab:

1 2 ⟶ Sistem Desimal
3 3
\+ \+
4 5
0100 0101 ⟶ Sistem XS-3

Pada XS-3, terdapat 6 kode yang tidak dapat digunakan yakni: 0000, 0001, 0010, 1101, 1110, dan 1111. Perhatikan contoh konversi berikut ini!

a. Ubah kode XS-3: 1001 1100 $0101_{XS-3}$ ke sistem desimal!
Jawab:

1001 1100 $0101_{XS-3}$ ⟶ Sistem Kode XS-3
9 12 5
3 3 3
\- \- \-
6 9 2 ⟶ Sistem desimal

b. Ubah kode XS-3: 0111 0001 $1010_{XS-3}$ ke sistem desimal!
Jawab:

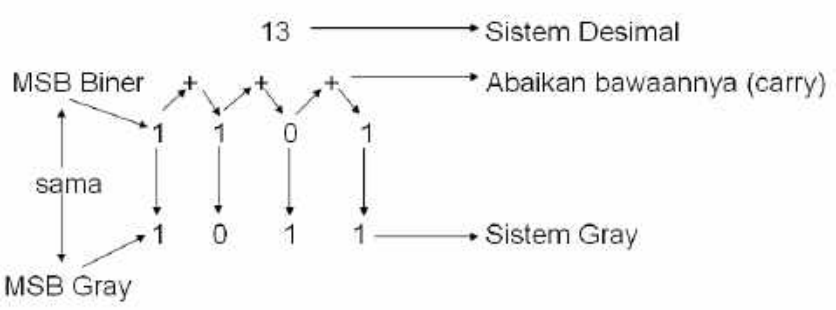
0111 0001 $1010_{XS-3}$ ⟶ Sistem Kode XS-3
7 1 10
3 3 3
\- \- \-
4 -2 7 ⟶ Sistem desimal

Kode XS-3 salah (invalid)

### 3. Kode Gray

Kode *gray* memiliki keunikan yakni setiap kali kode itu berubah nilainya secara berurutan misalnya dari 2 ke 3 atau dari 5 ke 6, hanya terdapat 1-bit saja yang berubah. Contoh: jika nilai kode *gray* berubah dari 2 ke 3, maka kode *gray* berubah dari 0011 ke 0010. Kode gray biasanya digunakan sebagai data yang menunjukkan posisi dari suatu poros mesin yang berputar. Cara mengubah bilangan desimal menjadi kode *gray* adalah sebagai berikut:
Ubah $13_{10}$ dalam bentuk kode Gray!

Jawab:



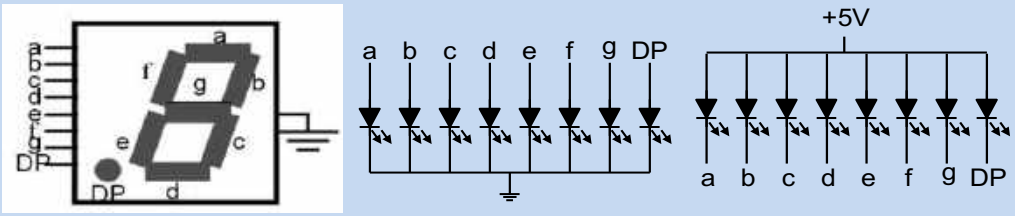

Langkah konversi:
1. MSB biner=MSB gray
2. Tambahkan MSB biner dengan bit kanannya, hasilnya adalah kode gray setelah MSB
3. Dengan mengabaikan carry, lakukan langkah ke-2 untuk semua bit yang ada

**Gambar 14. Konversi sistem desimal ke sistem gray**

Jadi, $13_{10}$ dalam bentuk kode *gray* adalah $1011_{GRAY}$.

### 4. Kode *7-Segment Display*

Hasil pemrosesan sinyal dari suatu rangkaian digital merupakan sinyal digital dalam bentuk kode-kode biner. Jika hasil tersebut tetap disajikan dalam bentuk aslinya yakni kode biner, maka kita akan mengalami kesulitan di dalam membacanya karena kita tidak terbiasa menggunakan kode biner dalam kehidupan sehari-hari. Kebiasaan kita adalah menggunakan sajian bilangan dalam bentuk desimal. Agar menjadi mudah dibaca, maka kode-kode biner tersebut perlu diubah tampilannya menggunakan tampilan desimal. Piranti yang digunakan untuk menampilkan data dalam bentuk desimal adalah LED *7-Segment Display* atau dinamakan peraga 7-segmen saja. Untuk menampilkan bilangan desimal, peraga ini memerlukan penggerak berbentuk kode-kode biner. Bentuk peraga 7-segmen ditunjukkan pada gambar 15 berikut ini.



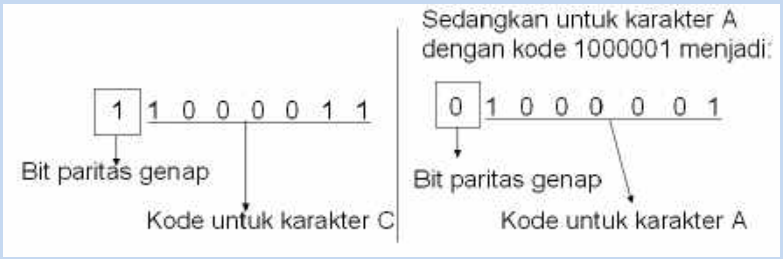

(a) (b) (c)

**Gambar 15. Peraga 7-segmen**

Setiap segmen dari peraga tersebut berupa LED yang susunannya membentuk suatu konfigurasi tertentu. Gambar 15 (a) menunjukkan wujud peraga 7-segmen dilihat dari atas, sedangkan gambar 15 (b) menunjukkan segmen-segmen peraga 7-segmen jenis *common cathode*. Pada jenis ini diperlukan sinyal tinggi untuk menyalakan setiap segmennya. Pada gambar 15 (c) ditunjukkan segmen-segmen peraga 7-segmen jenis *common anode* yang memerlukan sinyal rendah untuk menyalakan setiap segmennya.

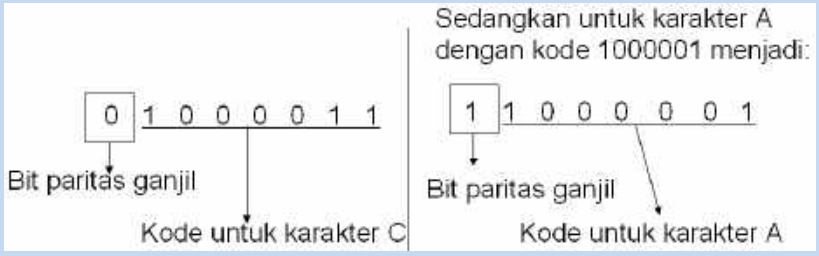
## 5. Kode ASCII

ASCII (*American Standard Code for Information Interchange*) merupakan kode biner untuk merepresentasikan bilangan, huruf, dan simbol, sehingga disebut juga kode alfanumerik. Terdapat dua jenis kode ASCII yakni berparitas genap dan ganjil. Contoh: Kode ASCII untuk karakter C adalah 1000011, memiliki jumlah bit 1 ganjil yakni 3 buah dan karakter A adalah 1000001, memiliki jumlah bit 1 genap yakni 2 buah. Untuk menyusun menjadi kode ASCII berparitas genap dilakukan sebagai berikut:




**Gambar 16. Bit paritas genap untuk karakter C dan A**

Sedangkan kode ASCII berparitas ganjil disusun dengan cara:




**Gambar 17. Bit paritas ganjil untuk karakter C dan A**

Tabel 3 menunjukkan nilai heksadesimal dari kode ASCII 7-bit. Untuk memperoleh nilai biner dari kode ASCII tersebut, anda harus mengubah nilai heksadesimal dari suatu kode ASCII ke dalam nilai biner. Contoh, pada tabel tercantum nilai heksadesimal dari karakter C adalah 43, maka nilai binernya adalah 100 0011. Untuk karakter A yang memiliki nilai heksadesimal 41, nilai binernya adalah 100 0001.

**Tabel 3. Nilai heksadesimal untuk beberapa kode ASCII 7-bit.**

| Simbol | Kode ASCII | Simbol | Kode ASCII | Simbol | Kode ASCII | Simbol | Kode ASCII | Simbol | Kode ASCII |
|---|---|---|---|---|---|---|---|---|---|
| ! | 21 | 4 | 34 | G | 47 | Z | 5A | m | 6D |
| ” | 22 | 5 | 35 | H | 48 | [ | 5B | n | 6E |
| # | 23 | 6 | 36 | I | 49 | \ | 5C | o | 6F |
| $ | 24 | 7 | 37 | J | 4A | ] | 5D | p | 70 |
| % | 25 | 8 | 38 | K | 4B | ^ | 5E | q | 71 |
| & | 26 | 9 | 39 | L | 4C | _ | 5F | r | 72 |
| ' | 27 | : | 3A | M | 4D | ` | 60 | s | 73 |
| ( | 28 | ; | 3B | N | 4E | a | 61 | t | 74 |
| ) | 29 | < | 3C | O | 4F | b | 62 | u | 75 |
| * | 2A | = | 3D | P | 50 | c | 63 | v | 76 |
| + | 2B | > | 3E | Q | 51 | d | 64 | w | 77 |
| , | 2C | ? | 3F | R | 52 | e | 65 | x | 78 |
| - | 2D | @ | 40 | S | 53 | f | 66 | y | 79 |
| . | 2E | A | 41 | T | 54 | g | 67 | z | 7A |
| / | 2F | B | 42 | U | 55 | h | 68 | { | 7B |
| 0 | 30 | C | 43 | V | 56 | i | 69 | \| | 7C |
| 1 | 31 | D | 44 | W | 57 | j | 6A | } | 7D |
| 2 | 32 | E | 45 | X | 58 | k | 6B | ~ | 7E |
| 3 | 33 | F | 46 | Y | 59 | l | 6C | DEL | 7F |

Ringkasan nilai biner, oktal, dan heksadesimal, serta nilai kode BCD, XS-3, Gray, dan kode peraga 7-segmen untuk bilangan desimal 0 sampai dengan 15 disajikan pada tabel 4.

**Tabel 4. Nilai berbagai sistem bilangan dan kode untuk bilangan desimal 0 s.d. 15**

| Desimal | Biner | Oktal | Heksa-desimal | BCD | | Gray | Peraga 7-segmen | |
|---|---|---|---|---|---|---|---|---|
| | | | | 8421 | XS-3 | | abcdefg | Display |
| 0 | 0 | 0 | 0 | 0000 | 0011 | 0000 | 1111110 | 0 |
| 1 | 1 | 1 | 1 | 0001 | 0100 | 0001 | 0110000 | 1 |
| 2 | 10 | 2 | 2 | 0010 | 0101 | 0011 | 1101101 | 2 |
| 3 | 11 | 3 | 3 | 0011 | 0110 | 0010 | 1111001 | 3 |
| 4 | 100 | 4 | 4 | 0100 | 0111 | 0110 | 0110011 | 4 |
| 5 | 101 | 5 | 5 | 0101 | 1000 | 0111 | 1011011 | 5 |
| 6 | 110 | 6 | 6 | 0110 | 1001 | 0101 | 1011111 | 6 |
| 7 | 111 | 7 | 7 | 0111 | 1010 | 0100 | 1110000 | 7 |
| 8 | 1000 | 10 | 8 | 1000 | 1011 | 1100 | 1111111 | 8 |
| 9 | 1001 | 11 | 9 | 1001 | 1100 | 1101 | 1110011 | 9 |
| 10 | 1010 | 12 | A | 0001 0000 | 0100 0011 | 1111 | 1111101 | A |
| 11 | 1011 | 13 | B | 0001 0001 | 0100 0100 | 1110 | 0011111 | B |
| 12 | 1100 | 14 | C | 0001 0010 | 0100 0101 | 1010 | 0001101 | C |
| 13 | 1101 | 15 | D | 0001 0011 | 0100 0110 | 1011 | 0111101 | D |
| 14 | 1110 | 16 | E | 0001 0100 | 0100 0111 | 1001 | 1101111 | E |
| 15 | 1111 | 17 | F | 0001 0101 | 0100 1000 | 1000 | 1000111 | F |

## G. Soal Latihan

1. Tulislah bobot bilangan:
   a. biner 12 bit, terdiri atas 10 bit bilangan biner bulat dan 2 bit bilangan biner pecahan!
   b. oktal 6 digit, terdiri atas 4 digit bilangan oktal bulat dan 2 digit bilangan oktal pecahan!
   c. heksadesimal 5 digit, terdiri atas 3 digit bilangan bulat dan 2 digit pecahan!

2. Ubahlah ke dalam sistem desimal nilai:
   a. $1000110_2$
   b. $11010010{,}011_2$
   c. $345_8$
   d. $153{,}24_8$
   e. $12AB_{16}$
   f. $2F1{,}E2_{16}$

3. Lakukan konversi ke sistem biner dengan menggunakan metode nilai digit dan metode bagi dua bilangan desimal berikut ini, dan lakukan konversi balik untuk memeriksa kebenaran konversi yang anda lakukan!
   a. $268_{10}$
   b. $513_{10}$
   c. $1025_{10}$
   d. $0{,}375_{10}$
   e. 71,875

4. Ubahlah ke dalam sistem oktal, dan heksadesimal bilangan desimal berikut ini dan lakukan konversi balik untuk memeriksa kebenaran konversi yang anda lakukan!
   a. $259_{10}$
   b. $5000_{10}$

5. Ubahlah $101101010101110111001_2$ ke sistem oktal dan heksadesimal!

6. Lakukan konversi nilai $2BF1A_{16}$ dan nilai $16243_8$ ke sistem biner!

7. Susunlah ke dalam kode BCD, Gray, dan XS-3 untuk bilangan desimal $458_{10}$!

8. Apa yang dimaksud dengan kode salah (invalid code) pada sistem kode BCD dan XS-3?

9. Lakukan konversi ke nilai desimal kode $000101001000_{BCD}$, $1001001101110110_{BCD}$, $0101000000001100001_{BCD}$, dan $1010011100111001_{XS\text{-}3}$, $1000101101100111010 1_{XS\text{-}3}$, serta $011101101001110001010101_{XS\text{-}3}$!

10. Perhatikan kode BCD dan XS-3 berikut ini:
    a. $1001001101011100_{BCD}$
    b. $010101110000000110010110_{BCD}$
    c. $101100001000_{BCD}$
    d. $0110010110110101_{XS\text{-}3}$
    e. $110000100110_{XS\text{-}3}$
    Tentukan kode mana yang salah dan kemukakan alasan mengapa kode tersebut anda nyatakan salah!

11. Tulislah kode peraga 7-segmen untuk menampilkan angka 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, dan F sesuai jenis peraga yang digunakan dengan mengisi tabel berikut ini:

**Tabel 5. Kode peraga 7-segmen untuk soal nomor 11 pada Bab II**

| Kode Peraga 7-Segmen | | Tampilan |
|---|---|---|
| Jenis Common Cathode | Jenis Common Anode | |
| a b c d e f g | a b c d e f g | |
| ..... | ..... | 0 |
| ..... | ..... | 1 |
| ..... | ..... | 2 |
| ..... | ..... | 3 |
| ..... | ..... | 4 |
| ..... | ..... | 5 |
| ..... | ..... | 6 |
| ..... | ..... | 7 |
| ..... | ..... | 8 |
| ..... | ..... | 9 |
| ..... | ..... | A |
| ..... | ..... | B |
| ..... | ..... | C |
| ..... | ..... | D |
| ..... | ..... | E |
| ..... | ..... | F |

12. Tulislah kode ASCII berparitas genap dan berparitas ganjil untuk karakter H, S, dan N!

13. Kode ASCII berikut ini berparitas genap ataukah ganjil!
    a. 01000010
    b. 01001111
    c. 11010011
    Dengan menggunakan tabel 3, tentukan kode-kode tersebut mewakili karakter apa?

14. Soal nomor 14 dan 15 adalah jenis pilihan ganda. Untuk mengerjakannya, pilih satu jawaban yang paling tepat dari opsi yang tersedia. Kode BCD yang valid dari beberapa kode berikut ini adalah:
    a. 1001 1100 1111 0001
    b. 0000 0101 1110 0011 1001 1100
    c. 1011 0001 0010 0011 0111
    d. 0110 0111 0000 0100 1100
    e. 0111 0101 0000 0001 1000 1001

15. Hasil konversi bilangan 12 desimal ke dalam sistem kode BCD adalah:
    a. 1100
    b. 0000 1100
    c. 1100 0000
    d. 0001 0010
    e. 0000 0001 0010

## KOMPETENSI DASAR III

1. Mahasiswa memahami watak gerbang logika dasar
2. Mahasiswa memahami prinsip-prinsip deskripsi, evaluasi dan implementasi rangkaian logika
3. Mahasiswa memahami watak gerbang universal NOR dan NAND serta hukum-hukum aljabar Boole sebagai dasar analisis dan perancangan rangkaian logika

## TUJUAN PEMBELAJARAN III

Agar mahasiswa dapat:

1. mengetahui arti konstanta dan variabel Boole pada gerbang logika
2. mengetahui manfaat tabel kebenaran dalam analisis dan perancangan rangkaian logika
3. menggambarkan simbol dan menuliskan persamaan logika gerbang-gerbang logika dasar
4. menyebutkan seri IC TTL gerbang logika dasar dan menggambarkan susunan pin yang tersedia
5. menjelaskan watak gerbang-gerbang logika dasar melalui tabel kebenaran dan diagram waktu
6. mendeskripsikan rangkaian logika menggunakan persamaan maupun simbol gerbang logika
7. mengevaluasi output rangkaian logika dalam bentuk deskripsi simbol dan persamaan Boole
8. mengimplementasikan rangkaian logika dari persamaan menjadi rangkaian
9. menggambar rangkaian logika menggunakan simbol gerbang dan koneksi secara benar
10. menggambarkan simbol dan menuliskan persamaan logika gerbang NOR dan NAND
11. menyebutkan seri IC TTL gerbang NOR dan NAND serta menggambarkan susunan *pin* nya
12. menjelaskan watak gerbang NOR dan NAND melalui tabel kebenaran dan diagram waktu
13. menuliskan ekspresi dari teorema variabel tunggal dan jamak pada aljabar Boole
14. menjelaskan watak universal gerbang NOR dan NAND
15. melakukan minimalisasi rangkaian menggunakan teorema de Morgan.

## GARIS BESAR MATERI III

Secara umum bagian ini terdiri atas dua meteri pokok yakni gerbang logika dasar yang merupakan elemen rangkaian/sistem digital dan aljabar Boole yang merupakan alat untuk keperluan perancangan dan analisis rangkaian digital/logika. Bagian ini diawali dengan penjelasan tentang tabel kebenaran yakni suatu tabel yang digunakan untuk menunjukkan pengaruh pemberian input terhadap keadaan output pada suatu rangkaian logika. Tabel kebenaran sesungguhnya menunjukkan watak suatu rangkaian logika sehingga pengetahuan tentang teknik penyusunan maupun interpretasi terhadap tabel kebenaran sangat diperlukan bagi anda yang ingin merancang dan menganalisis rangkaian logika dengan mudah dan baik.

Pada bab ini akan diperkenalkan gerbang logika dasar dalam bentuk definisi, simbol, ekspresi Boole atau persamaan logika, maupun wataknya. Memahami gerbang logika dasar merupakan landasan bagi kegiatan perancangan dan analisis rangkaian logika. Hal itu dikarenakan gerbang logika merupakan elemen penyusun rangkaian logika. Selanjutnya anda akan diperkenalkan cara mendeskripsikan rangkaian logika dengan menggunakan persamaan logika. Pemahaman terhadap materi ini diperlukan agar anda dapat melakukan evaluasi terhadap output rangkaian logika dengan cepat. Selain itu, dengan pemahaman tersebut anda juga dapat melakukan karakterisasi atau penentuan karakteristik suatu rangkaian logika secara teoritik dengan cepat dan mudah.

Hasil rancangan rangkaian logika yang berbentuk ekspresi Boole tidak akan dapat berguna jika tidak diimplementasikan. Pada bab ini akan diperkenalkan juga cara mengimplementasikan rangkaian logika dari suatu persamaan logika yang diketahui.

Selain gerbang logika dasar AND, OR, dan NOT, melalui bab ini akan dikenalkan juga gerbang yang memiliki sifat universal yakni NAND dan NOR. Pemahanan terhadap watak gerbang-gerbang tersebut penting karena keduanya bersifat universal dapat menggantikan gerbang logika dasar dalam membangun semua rangkaian logika.

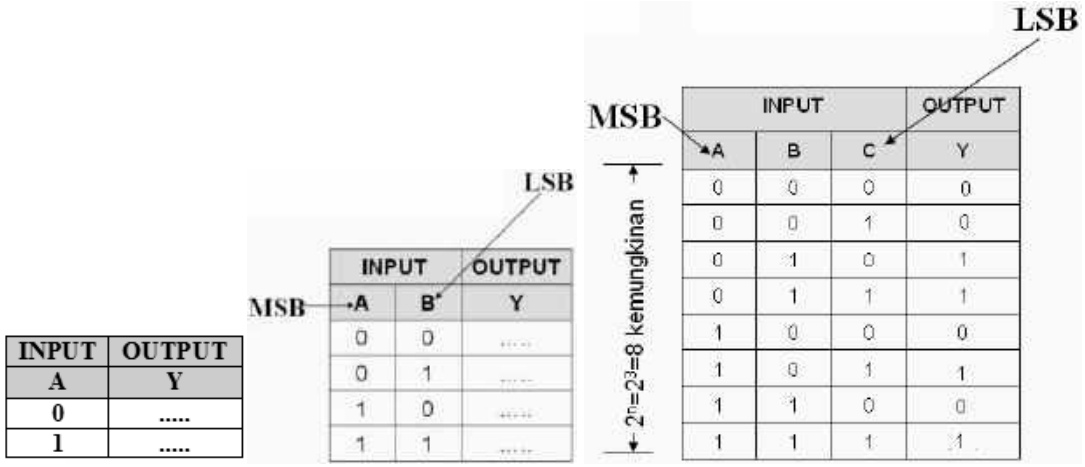
Selanjutnya akan dijelaskan kegiatan perancangan dan analisis rangkaian logika yang merupakan kompetensi atau keahlian yang diharapkan dapat anda capai dalam mempelajari buku ini. Sebagai landasan bagi kegiatan perancangan dan analisis, diperkenalkan juga berbagai teorema Aljabar Boole.

# BAB III
# GERBANG LOGIKA DASAR DAN ALJABAR BOOLE

## A. Tabel Kebenaran (Truth Table)

Tabel kebenaran atau *truth table* merupakan tabel yang menunjukkan pengaruh pemberian level logika pada input suatu rangkaian logika terhadap keadaan level logika outputnya. Contoh tabel kebenaran untuk rangkaian logika dengan 1 input, 2 input dan 3 input ditunjukkan pada tabel 6 berikut ini.

**Tabel 6. Tabel kebenaran rangkaian logika berbagai jumlah variabel input**




(a) (b) (c)

Kolom Y diisi sesuai dengan karakteristik rangkaian logikanya, tabel 6 (c) menunjukkan contoh tabel kebenaran rangkaian detektor bilangan prima 3-bit!

## B. Gerbang Logika Dasar

### 1. Gerbang OR

Gerbang OR didefinisikan sebagai gerbang logika yang memberikan keadaan logika 1 (tinggi) pada outputnya, jika keadaan salah satu atau lebih inputnya berlogika 1 (tinggi). Tabel kebenarannya untuk OR-2 input ditunjukkan pada tabel 7 berikut ini.

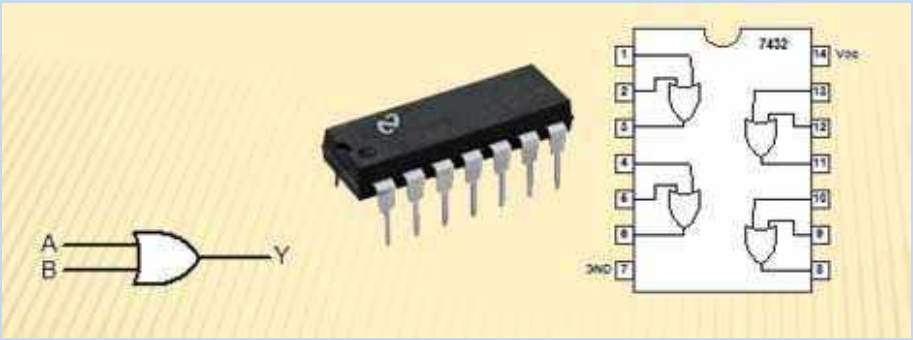
**Tabel 7. Tabel kebenaran gerbang OR 2 input**

| INPUT | | OUTPUT |
|---|---|---|
| A | B | Y |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Persamaan logika atau ekspresi Boole output gerbang OR dinyatakan dengan persamaan:

**Y=A + B** atau **Y=A or B** **persamaan (1)**

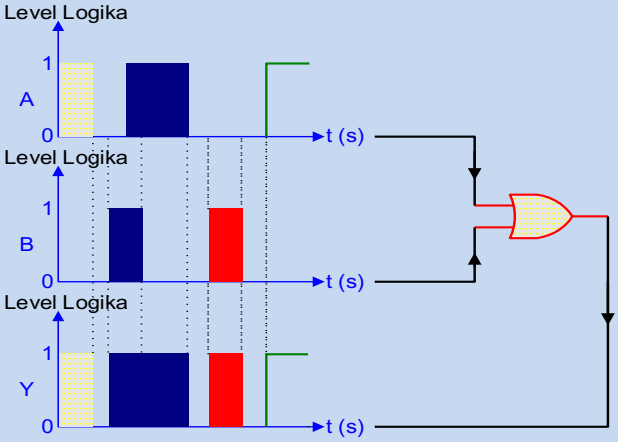
Sedangkan simbol dan IC TTL (transistor transistor logic) yang menyediakan fungsi gerbang OR yakni seri 7432 disajikan pada gambar 18.




(a) (b) (c)

**Gambar 18. (a) Simbol gerbang OR, (b) bentuk nyata IC 7432, (c) susunan pin**

Selain dengan input berupa saklar, watak gerbang logika juga dapat dipelajari dengan memberikan input berupa gelombang kotak. Contoh watak gerbang OR dengan input berupa gelombang kotak ditunjukkan pada gambar 19 berikut ini.




**Gambar 19. Watak gerbang OR dengan input gelombang kotak**

## 2. Gerbang AND

Gerbang AND didefinisikan sebagai gerbang logika yang memberikan keadaan level logika 1 (tinggi) pada outputnya, jika dan hanya jika semua keadaan inputnya berlevel logika 1 (tinggi). Tabel kebenaran gerbang AND 2 input ditunjukkan pada tabel 8.

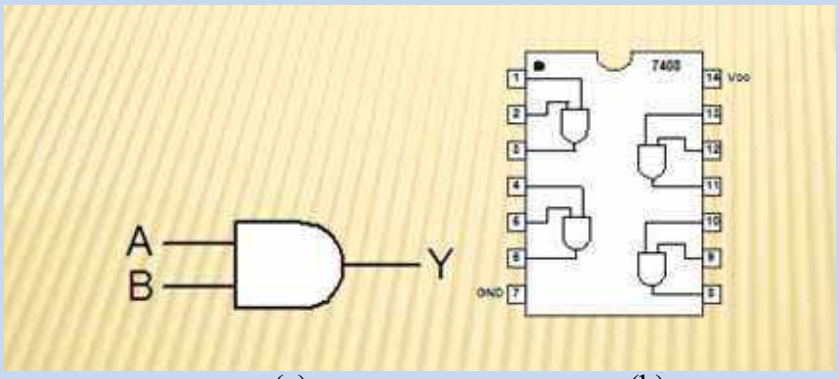
**Tabel 8. Tabel kebenaran gerbang AND 2 input**

| INPUT | | OUTPUT |
|---|---|---|
| A | B | Y |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Persamaan logika untuk gerbang AND 2 input adalah:

**Y = A .B** atau **Y = AB** atau **Y = A and B** **persamaan (2)**

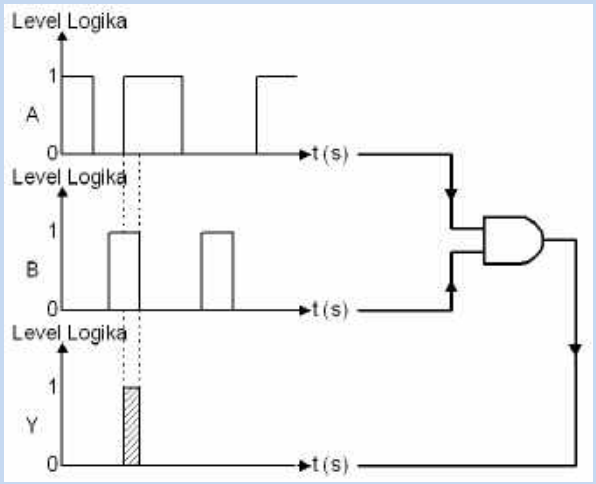
dan simbol serta susunan IC yang menyediakan fungsi AND ditunjukkan gambar 20.




(a) (b)

**Gambar 20. Gerbang AND: (a) simbol, (b) susunan pin IC 7408**

Untuk input berbentuk gelombang kotak, watak atau respons output gerbang AND 2 input ditunjukkan seperti contoh pada gambar 21.




**Gambar 21. Watak gerbang AND terhadap input berbentuk gelombang kotak**

### 3. Gerbang NOT

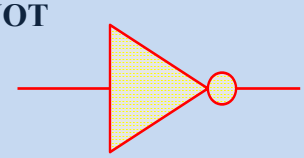
NOT merupakan gerbang logika yang memberikan keadaan level logika 1 (tinggi) pada outputnya, jika keadaan inputnya berlevel logika 0 (rendah) atau sebaliknya gerbang ini akan memberikan keadaan level logika 0 (rendah) pada outputnya jika keadaan inputnya berlevel 1 (tinggi). Persamaan logika gerbang NOT adalah:

**Y= not A** atau **$Y=\overline{A}$** **persamaan (3)**

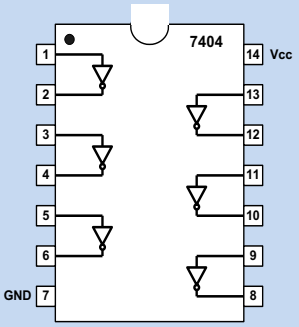
Sedangkan tabel kebenaran, simbol dan susunan pin IC gerbang NOT adalah:

**Tabel 9. Tabel Kebenaran NOT**

| INPUT | OUTPUT |
|---|---|
| A | Y |
| 0 | 1 |
| 1 | 0 |

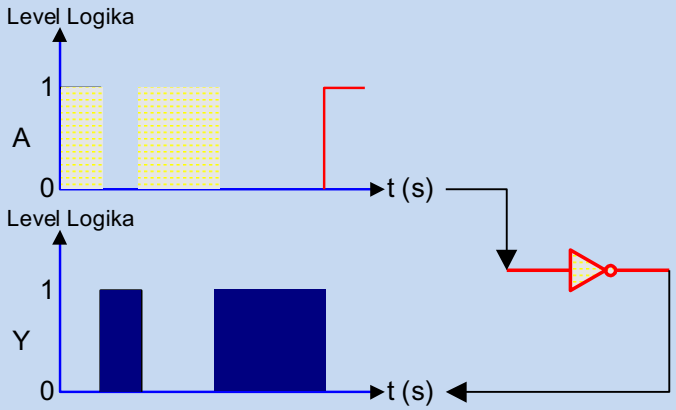


**Gambar 22. Simbol gerbang NOT**




**Gambar 23. Susunan pin IC NOT seri 7404**

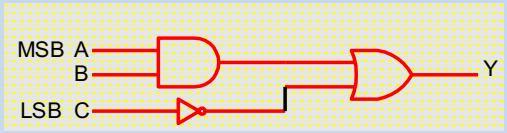
Watak gerbang NOT terhadap input gelombang kotak ditunjukkan pada gambar 48 berikut ini.




**Gambar 24. Watak gerbang NOT terhadap input berbentuk gelombang kotak**

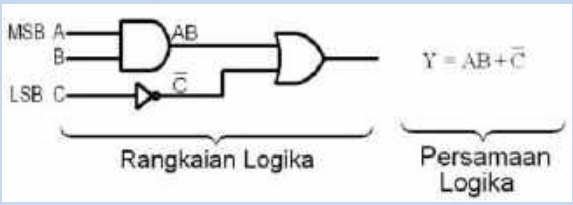
### C. Mendeskripsikan Rangkaian Logika

Rangkaian logika dapat dideskripsikan dalam dua bentuk yakni dengan menggunakan simbol elemen logika dan menggunakan persamaan logika/ekspresi Boole. **Contoh:** Deskripsikan rangkaian berikut ini dengan menggunakan persamaan logika!




**Gambar 25. Contoh rangkaian logika yang akan dideskripsikan dengan persamaan**

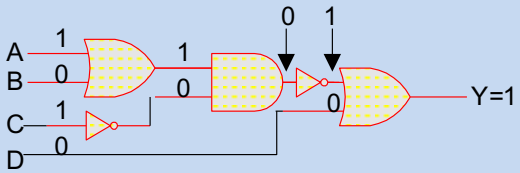
Untuk dapat mendeskripsikan rangkaian logika ke dalam bentuk persamaan logika, harus dituliskan terlebih dahulu persamaan logika pada setiap output gerbang penyusun rangkaian tersebut. Selanjutnya, penulisan persamaan logika terhadap operasi yang dilakukan oleh gerbang terakhir akan menghasilkan persamaan logika dari rangkaian tersebut.




**Gambar 26. Cara mendeskripsikan rangkaian dengan persamaan logika**

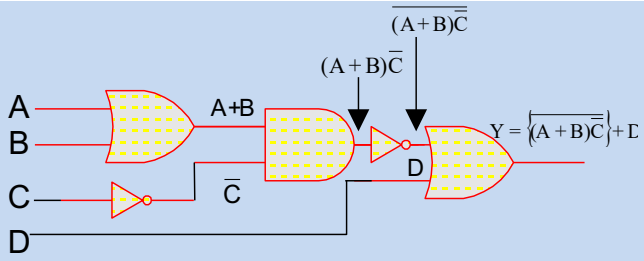
## D. Mengevaluasi Output Persamaan Logika

Evaluasi output rangkaian logika dapat dilakukan dengan menggunakan deskripsi simbol gerbang seperti di bawah ini:




**Gambar 27. Evaluasi output rangkaian logika dengan deskripsi simbol**

Tetapi cara seperti itu akan menjadi kurang efisien, karena setiap output gerbang harus dievaluasi, sedangkan jumlah kemungkinan input dengan 4 variabel input adalah 16 buah, dapat dibayangkan betapa repotnya mengevaluasi output rangkaian logika itu untuk semua kemungkinan input yang ada! Agar evaluasi output rangkaian logika menjadi mudah, sebaiknya rangkaian dideskripsikan terlebih dahulu menjadi persamaan logika seperti berikut ini.




**Gambar 28. Evaluasi output rangkaian logika dengan deskripsi persamaan**

Dengan menggunakan persamaan logika yang telah ditemukan yakni $Y=\{\overline{(A+B)\overline{C}}\}+D$, dapat dengan mudah disusun tabel 10.

**Tabel 10. Tabel kebenaran persamaan logika $Y=\{\overline{(A+B)\overline{C}}\}+D$**

| INPUT | | | | OUTPUT SETIAP GERBANG | | | | OUTPUT |
|---|---|---|---|---|---|---|---|---|
| A | B | C | D | A+B | $\overline{C}$ | $(A+B)\overline{C}$ | $\overline{(A+B)\overline{C}}$ | Y |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |

## E. Mengimplementasikan Rangkaian Logika

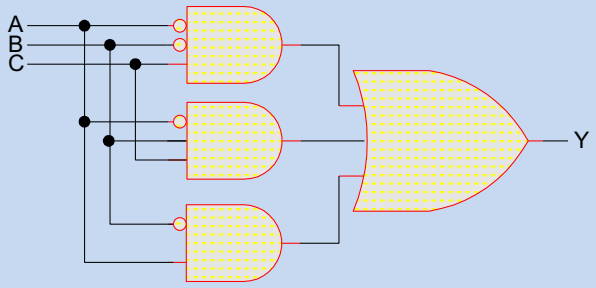
Untuk mempelajari implementasi rangkaian logika perhatikan contoh berikut ini! Implementasikan persamaan logika $Y=\overline{A}\,\overline{B}C+\overline{A}BC+A\overline{B}$ ke dalam bentuk rangkaian logika!

**Jawab:**

Berdasarkan persamaan tersebut kebutuhan gerbang adalah:

**OR 3-input : 1 buah** **AND 2-input : 1 buah**
**AND 3-input : 2 buah** **NOT : 4 buah**

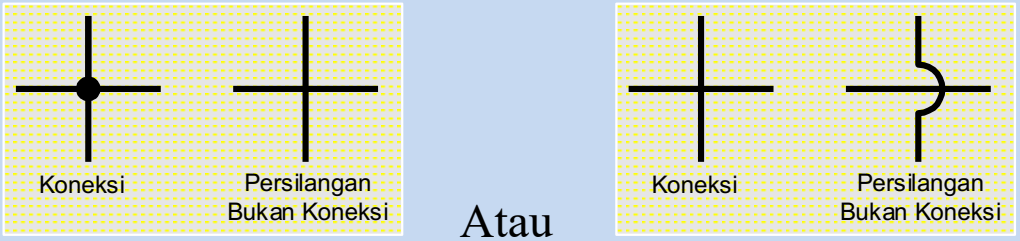
Jadi, rangkaian logika untuk persamaan $Y = \overline{A}\,\overline{B}C + \overline{A}BC + A\overline{B}$ adalah:




**Gambar 29. Rangkaian logika untuk persamaan $Y = \overline{A}\,\overline{B}C + \overline{A}BC + A\overline{B}$**

Agar penyusunan rangkaian logika dapat dilakukan dengan benar, perlu memperhatikan hal-hal sebagai berikut:

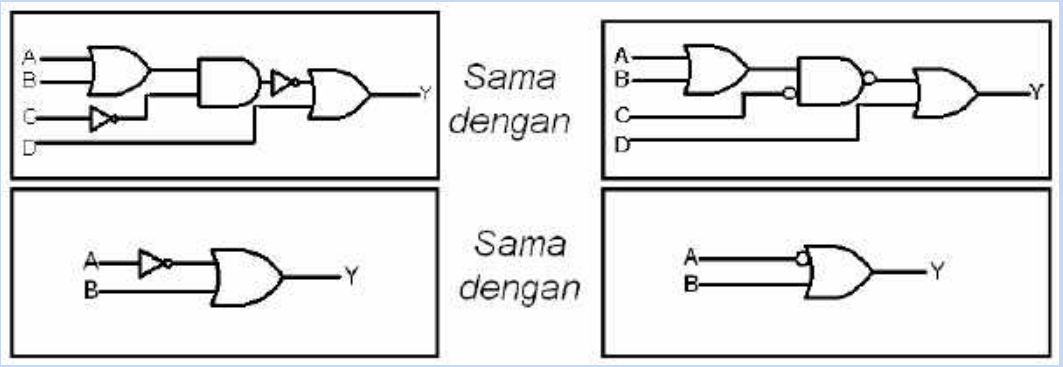
1. Interseksi rangkaian menggunakan simbol sebagai berikut:




(a) (b)

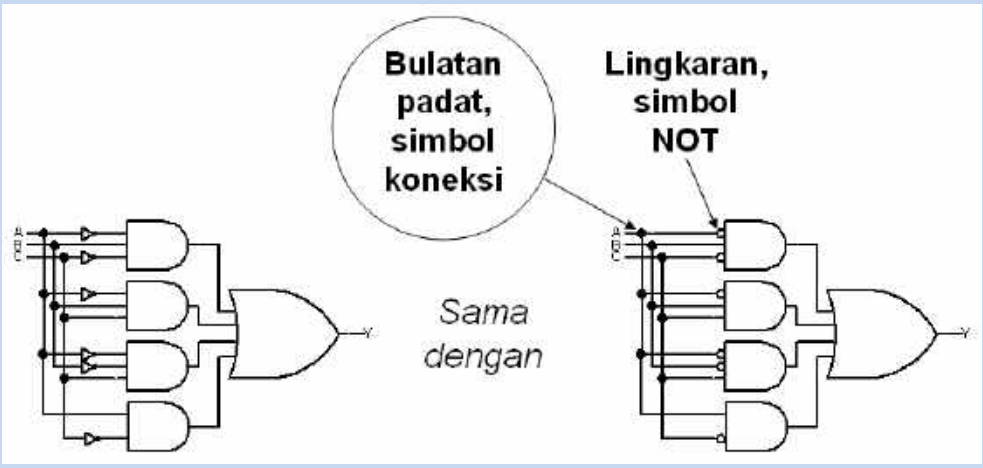
**Gambar 30. Simbol interseksi rangkaian logika**

2. Pada rangkaian yang mengandung gerbang NOT, simbol gerbang NOT dapat diganti dengan lingkaran yang menempel pada gerbang yang dihubungkannya. Contoh:



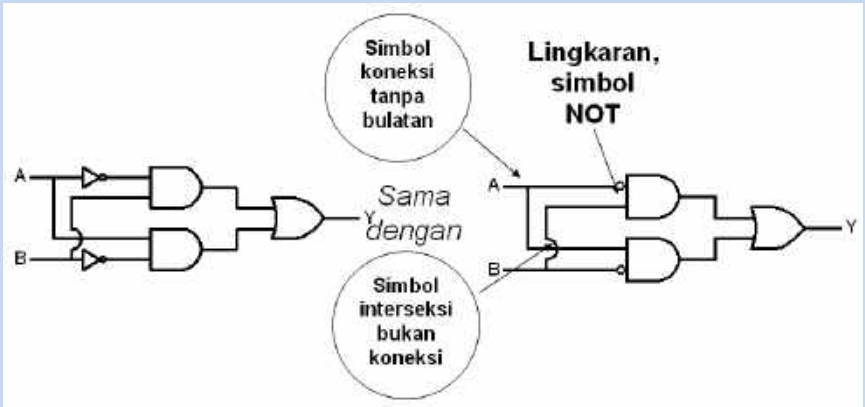

**Gambar 31. Pengganti simbol gerbang NOT pada rangkaian logika**

Contoh berikut ini menunjukkan rangkaian logika yang mengandung interseksi dan gerbang NOT.




**Gambar 32. Rangkaian logika yang mengandung interseksi dan gerbang NOT**

Contoh rangkaian lain yang mengandung interseksi dan gerbang NOT:




**Gambar 33. Rangkaian logika yang mengandung interseksi dan gerbang NOT**

## F. Gerbang NOR dan Gerbang NAND

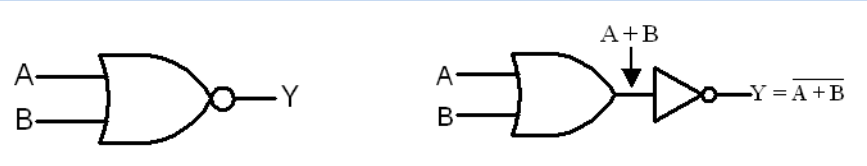
Selain gerbang logika dasar AND, OR, dan NOT, terdapat dua buah gerbang yakni NAND dan NOR yang sangat penting peranannya dalam perancangan dan analisis rangkaian logika. Kedua gerbang tersebut sangat penting karena memiliki sifat universal, yakni dapat menggantikan gerbang logika dasar dalam membangun semua rangkaian logika.

### 1. Gerbang NOR

NOR merupakan gerbang logika gabungan dari gerbang OR dan gerbang NOT. Outputnya kebalikan dari output gerbang OR. Persamaan logikanya adalah:

$$Y = \overline{A + B} \qquad \textbf{persamaan (4)}$$

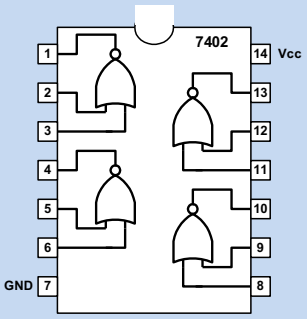
Simbol, tabel kebenaran dan IC nya ditunjukkan gambar berikut ini.




**Gambar 34. Simbol gerbang NOR dan rangkaian ekivalennya**

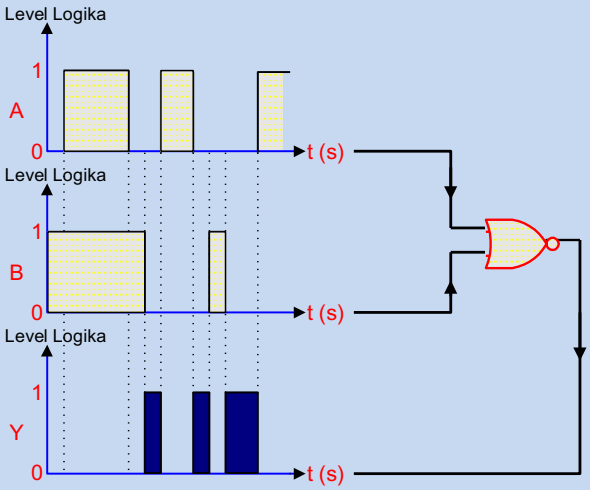
**Tabel 11. Tabel kebenaran gerbang NOR**

| INPUT | | OUTPUT OR | OUTPUT NOR |
|---|---|---|---|
| A | B | A+B | $Y = \overline{A+B}$ |
| 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 |




**Gambar 35. IC NOR 7402**

Jika gerbang NOR inputnya berupa gelombang kotak, wataknya ditunjukkan pada gambar 36 berikut ini.



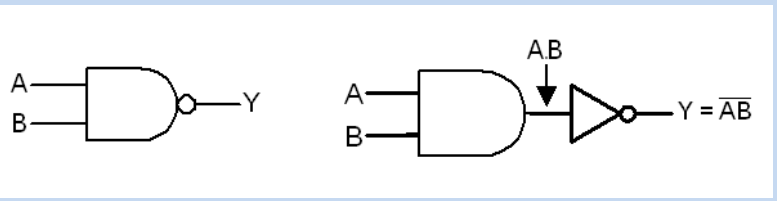

**Gambar 36. Watak gerbang NOR dengan input gelombang kotak**

## 2. Gerbang NAND

NAND adalah gerbang logika yang di dalamnya terdapat gabungan gerbang AND dan gerbang NOT. Outputnya merupakan kebalikan dari gerbang AND dengan persamaan logika:

$$Y = \overline{AB} \qquad \textbf{persamaan (5)}$$

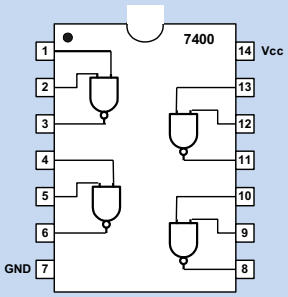
Simbol, tabel kebenaran dan IC nya ditunjukkan gambar berikut ini.




**Gambar 37. Simbol gerbang NOR dan rangkaian ekivalennya**

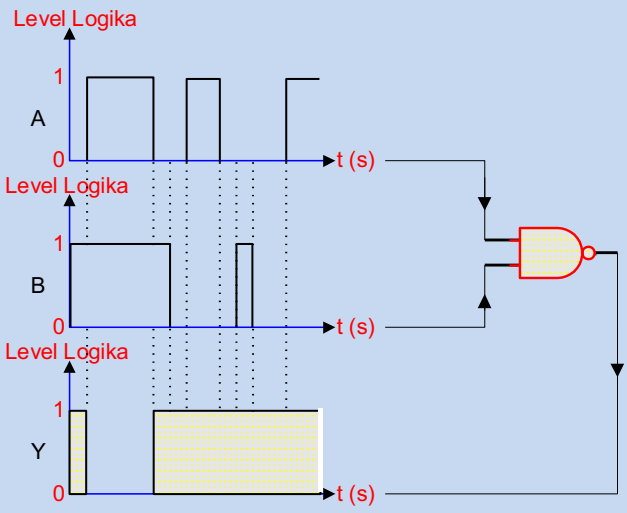
**Tabel 12. Tabel kebenaran gerbang NAND**

| INPUT | | OUTPUT AND | OUTPUT NAND |
|---|---|---|---|
| A | B | AB | Y = $\overline{AB}$ |
| 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |




**Gambar 38. IC NAND 7400**

Jika pada kedua inputnya diberikan gelombang kotak, respons output gerbang NAND 2 input ditunjukkan pada gambar berikut ini.



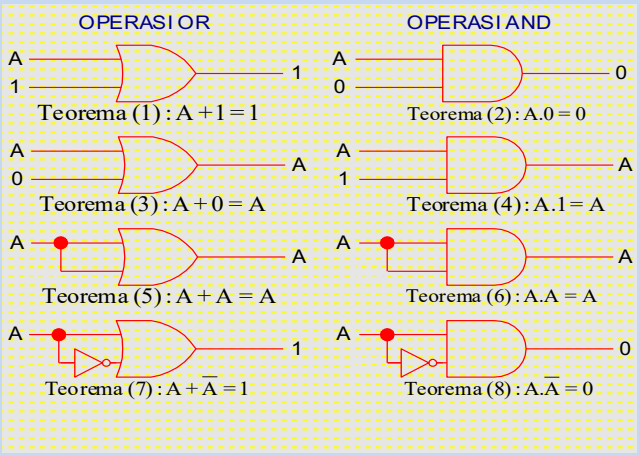

**Gambar 39. Watak gerbang NAND dengan input gelombang kotak**

## G. Teorema-teorema Aljabar Boole

Aljabar Boole sangat penting peranannya di dalam proses perancangan maupun analisis rangkaian logika. Terdapat dua jenis teorema dalam aljabar Boole yakni teorema variabel tunggal dan teorema variabel jamak. Setiap teorema baik yang bersifat tunggal maupun jamak selalu memiliki teorema rangkapnya.

## 1. Teorema Variabel Tunggal

Teorema variabel tunggal aljabar Boole diturunkan dari operasi logika dasar OR, AND, dan NOT. Penurunan teorema variabel tunggal ditunjukkan pada gambar berikut ini.




**Gambar 40. Penurunan teorema variabel tunggal**

Perhatikan bahwa teorema pada operasi AND dapat diperoleh melalui teorema pada operasi OR atau sebaliknya. Untuk memperoleh suatu teorema dari terorema yang diketahui, lakukan dengan:

a. Mengubah tanda + menjadi dot (.) atau sebaliknya
b. Mengubah 1 menjadi 0 atau sebaliknya

Berdasarkan gambar 40, dapat dituliskan teorema-teorema aljabar Boole untuk variabel tunggal seperti tersaji pada tabel 13 berikut ini.

**Tabel 13. Teorema-teorema aljabar Boole untuk variabel tunggal**

| Teorema | Ekspresi | Sifat Rangkap |
|---|---|---|
| Satu dan Nol | Teorema (1): A+1=1 | Teorema (2): A.0=0 |
| Identitas | Teorema (3): A+0=A | Teorema (4): A.1=A |
| Idempoten | Teorema (5): A+A=A | Teorema (6): A.A=A |
| Komplemen | Teorema (7): $A + \overline{A} = 1$ | Teorema (8): $A.\overline{A} = 0$ |
| Involusi | Teorema (9): $\overline{\overline{A}} = A$ | - |

### 2. Teorema Variabel Jamak

Teorema-teorema variabel jamak aljabar Boole umumnya sama dengan teorema-teorema pada aljabar biasa seperti ditunjukkan pada tabel berikut ini.

**Tabel 14. Teorema-teorema aljabar Boole untuk variabel jamak**

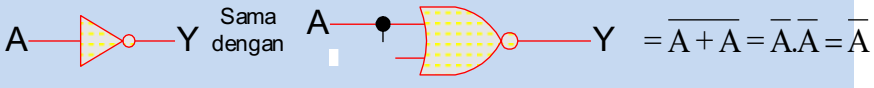
| Teorema | Ekspresi | Sifat Rangkap |
|---|---|---|
| Komutatif | Teorema (10): A+B=B+A | Teorema (11): AB=BA |
| Asosiatif: | Teorema (12): A+(B+C)=(A+B)+C | Teorema (13): A(BC)=(AB)C |
| Distributif: | Teorema (14): A+BC=(A+B)(A+C) | Teorema (15): A(B+C)=AB+AC |
| Absorpsi | Teorema (16): A+AB=A<br>Teorema (18): $A+\overline{A}B=A+B$ | Teorema (17): A(A+B)=A<br>Teorema (19): $A(\overline{A}+B)=AB$ |
| De Morgan | Teorema (20): $\overline{A+B+...}=\overline{A}.\overline{B}....$ | Teorema (21): $\overline{A.B....}=\overline{A}+\overline{B}+...$ |

## H. Universalitas Gebang NOR dan NAND

Gerbang NOR dan NAND memiliki sifat universal, artinya semua gerbang logika atau rangkaian logika dapat disusun dengan menggunakan gerbang NOR saja atau NAND saja.

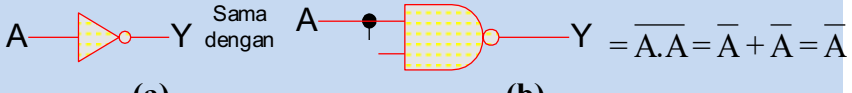
### 1. NOR/NAND sebagai gerbang NOT

Gambar 41 menunjukkan gerbang NOT yang dibentuk dari gerbang NOR dan gambar 42 NOT yang dibentuk dari gerbang NAND.




**(a)** **(b)**

**Gambar 41. Gerbang NOT dengan NOR**

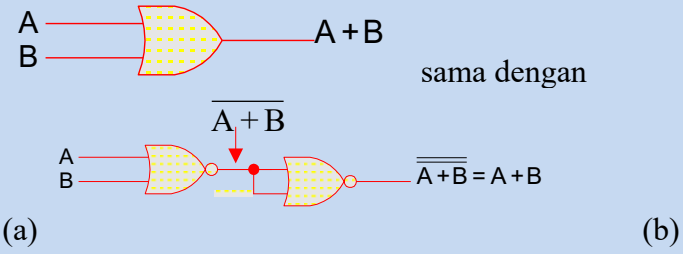



**(a)** **(b)**

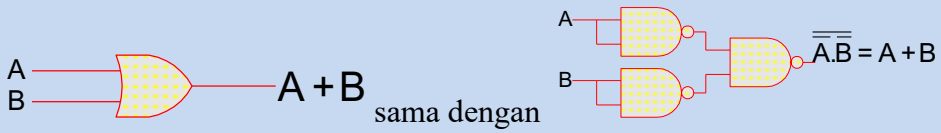
**Gambar 42. Gerbang NOT dengan NAND**

**2. NOR/NAND sebagai gebang OR**

Gerbang OR yang dibentuk dari gerbang NOR ditunjukkan pada gambar 43 dan gerbang OR dari gerbang NAND ditunjukkan pada gambar 44.



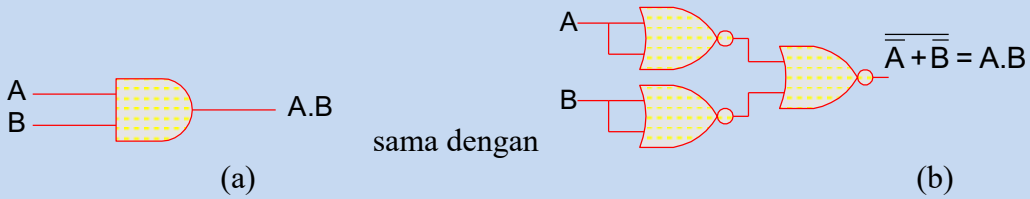

**Gambar 43. Gerbang OR dengan menggunakan NOR**



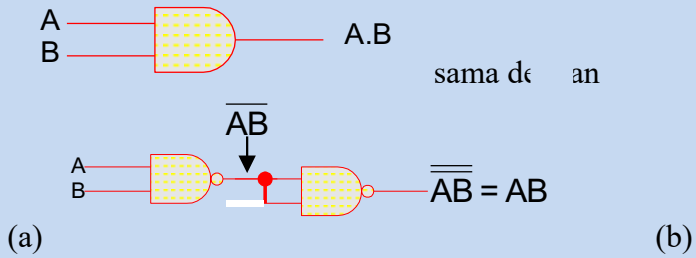

**Gambar 44. Gerbang OR dengan NAND**

**3. Gerbang AND dengan NOR/NAND**

Gerbang AND yang dibentuk dari gerbang NOR ditunjukkan pada gambar 45 dan gerbang AND dari gerbang NAND ditunjukkan pada gambar 46.




**Gambar 45. Gerbang AND dengan NOR**

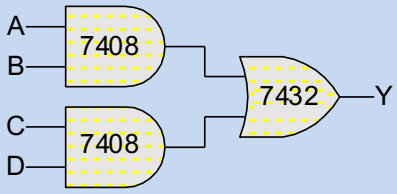



**Gambar 46. Gerbang AND dengan NAND**

Keuntungan menggunakan gerbang NAND dan NOR adalah implementasinya menjadi lebih efisien.

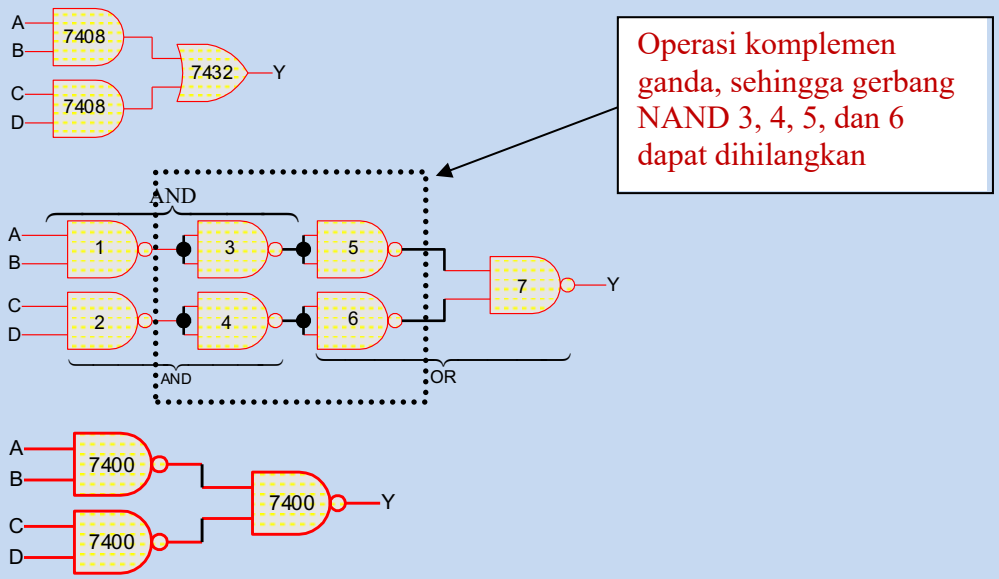
**Contoh:**

Suatu persamaan logika Y=AB+CD memerlukan sebuah gerbang OR 2-input dan dua buah gerbang AND 2-input. Jika persamaan tersebut diimplementasikan dengan gerbang OR dan AND, hasilnya adalah:




**Gambar 47. Implementasi Y=AB+CD dengan gerbang AND dan OR**

Impelementasi tersebut memerlukan sebuah gerbang OR 2 input dan dua buah gerbang AND 2 input, atau memerlukan dua buah IC yakni IC OR 7432 dan IC AND 7408. Tetapi, jika rangkaian tersebut diimplementasikan dalam bentuk NAND hanya memerlukan sebuah IC NAND 7400. Coba perhatikan penyusunan rangkaian dalam bentuk NAND berikut ini! Untuk membentuk persamaan Y=AB+CD dalam bentuk NAND, dilakukan dengan cara sebagai berikut:




**Gambar 48. Implementasi Y=AB+CD dengan gerbang NAND**

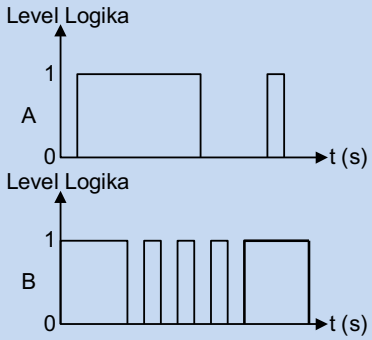
Terlihat bahwa dengan menggunakan gerbang NAND saja, implementasi Y=AB+CD cukup menggunakan 3 buah gerbang NAND atau 1 buah IC 7400. Untuk membuktikan bahwa kedua implementasi tersebut memiliki watak yang sama, lakukan percobaan berikut ini!

## I. Soal Latihan

1. Susun tabel kebenaran rangkaian logika dengan tiga input yakni A (MSB), B, dan C (LSB) dan satu output yakni Y yang memberikan keadaan output bernilai tinggi jika:

   a. input A dan input B berbeda!
   b. input A dan C sama!
   c. input B dan C berbeda!

2. Perhatikan bentuk gelombang input berikut ini!

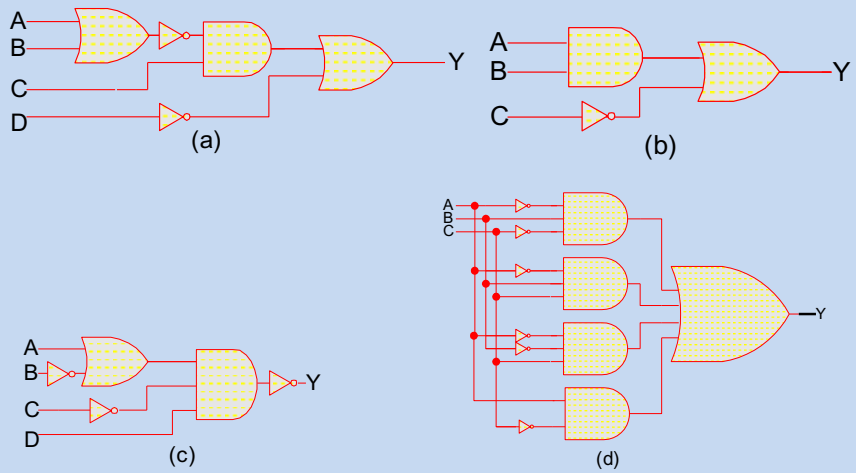



**Gambar 49. Bentuk gelombang input untuk soal nomor 2 Bab 3.**

   Gambarkan bentuk gelombang outputnya jika kedua input tersebut dihubungkan dengan input gerbang: OR, AND, NOR dan NAND! Gambarkan pula gelombang outputnya jika input B dihubungkan dengan input gerbang NOT.

3. Gambarkan bentuk gelombang output rangkaian $Y = \overline{A} + B$ jika inputnya berbentuk gelombang kotak seperti pada gambar 49!

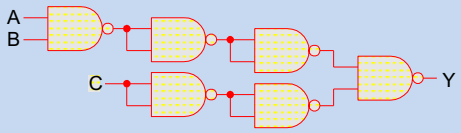
4. Deskripsikan rangkaian berikut ini ke dalam bentuk persamaan logika!




**Gambar 50. Rangkaian untuk soal nomor 3 Bab 3**

   Susun tabel kebenaran dari rangkaian pada gambar 50 (b), dan 50 (d)!

5. Implementasikan persamaan berikut ini ke dalam bentuk rangkaian logika!
   a. $X = (A + B)(A + \overline{C})B$
   b. $Y = AB\overline{C} + \overline{A}B\overline{C} + B\overline{C}$
   c. $Z = \overline{(A + \overline{D}) \bullet (\overline{B} + C)}$
6. Susunlah rangkaian $Y = AB + C$ dengan menggunakan gerbang AND dan OR! Susun pula rangkaian hanya dengan gerbang NAND saja! Tunjukkan bahwa dalam bentuk NAND rangkaian tersebut hanya memerlukan sebuah IC 7400!
7. Dengan menggunakan gerbang AND dan OR, susunlah rangkaian $Y = (A + B)(A + C)$! Susun pula rangkaian hanya dengan menggunakan gerbang NOR saja! Tunjukkan bahwa dalam bentuk NOR rangkaian tersebut hanya memerlukan sebuah IC 7402!
8. Perhatikan rangkaian berikut ini!




**Gambar 51. Rangkaian untuk soal nomor 9 Bab 3**

   Ubahlah rangkaian tersebut ke dalam bentuk gerbang AND, OR, dan NOT! Susunlah tabel kebenarannya!
9. Lakukan eliminasi/penghilangan gerbang-gerbang pada gambar 51 yang membentuk operasi komplemen ganda sehingga dihasilkan rangkaian yang cukup diimplementasikan dengan sebuah IC 7400!
10. Soal nomor 10 dan 11 adalah soal jenis pilihan ganda. Kerjakan dengan memilih satu jawaban yang paling tepat dari opsi yang tersedia. Teorema-teorema aljabar Boole yang semuanya benar di bawah ini adalah:
    a. A+1=1, A+A=2A, A.0=0
    b. A+1=1, A+A=2A, A+0=A
    c. A+1=1, A+A=A, A+0=A
    d. A.1=1, A+A=A, A+1=A
    e. A+1=0, A+A=A, A+1=A
11. Persamaan $Y = \overline{A} + \overline{B} + \overline{C}$ menurut de Morgan sama dengan persamaan:
    a. $Y = \overline{\overline{A} + \overline{B} + \overline{C}}$
    b. $Y = \overline{A\ B\ C}$
    c. $Y = \overline{A + B + C}$
    d. $Y = \overline{\overline{A + B + C}}$
    e. $Y = \overline{\overline{A\ B\ C}}$

## KOMPETENSI DASAR IV

1. Mahasiswa memahami dasar-dasar analisis dan perancangan logika kombinasi menggunakan Aljabar Boole
2. Mahasiswa memahami dasar-dasar analisis dan perancangan logika kombinasi menggunakan metode peta Karnaugh

## TUJUAN PEMBELAJARAN IV

Agar mahasiswa dapat:

1. mendefinisikan pengertian rangkaian logika kombinasi
2. menyebutkan dan menjelaskan bentuk persamaan logika
3. menerapkan konsep Aljabar Boole untuk meminimalisasi suatu rangkaian logika
4. merancang rangkaian logika kombinasi sederhana
5. memperoleh bentuk persamaan minimum dari persamaan logika yang diketahui menggunakan metode peta Karnaugh
6. memperoleh bentuk persamaan minimum dari tabel kebenaran yang diketahui menggunakan metode peta Karnaugh
7. mengubah persamaan minimum ke dalam bentuk NOR atau NAND

## GARIS BESAR MATERI IV

Logika kombinasi merupakan salah satu jenis rangkaian logika yang keadaan outputnya hanya tergantung pada kombinasi input-inputnya saja. Selain rangkaian logika kombinasi, terdapat pula rangkaian logika sekuensi yang outputnya merupakan fungsi dari keadaan output sebelumnya. Pada bagian III telah diuraikan berbagai teorema aljabar Boole yang sangat diperlukan dalam proses perancangan rangkaian logika kombinasi. Telah pula dikemukakan di muka bahwa pada tahap akhir proses perancangan rangkaian logika kombinasi akan dihasilkan persamaan logika. Dalam hal ini, setiap persamaan logika yang akan diimplementasikan perlu diuji terlebih dahulu bentuk minimumnya. Implementasi perasamaan logika ke dalam bentuk rangkaian logika pada dasarnya dapat dilakukan jika persamaannya sudah dalam bentuk minimum. Tahap minimalisasi rangkaian logika diperlukan agar diperoleh rangkaian dengan watak yang sama namun dengan jumlah gerbang yang paling sedikit. Rangkaian dengan jumlah gerbang yang paling sedikit akan lebih murah harganya, dan dari segi tata letak komponennya akan lebih sederhana. Bagian ini akan memperkenalkan kepada anda metode pengujian bentuk minimum dari persamaan logika maupun prosedur minimalisasi rangkaian logika dengan menggunakan aljabar Boole.

Selain dengan aljabar Boole, pengujian bentuk minimum dan minimalisasi suatu rangkaian logika juga dapat dilakukan dengan menggunakan peta Karnaugh. Bagian ini akan memperkenalkan pula kepada anda metode peta Karnaugh untuk pengujian bentuk minimum dan minimalisasi suatu rangkaian logika.

Rangkaian logika dianggap memiliki bentuk minimum jika jumlah gerbang-gerbang penyusunnya paling sedikit. Selain itu, suatu rangkaian dapat dianggap minimum jika gerbang-gerbang penyusunnya terdiri hanya satu jenis saja. Melalui bagian ini anda akan diperkenalkan juga metode konversi suatu bentuk persamaan logika ke dalam bentuk NAND saja atau NOR saja agar dapat diperoleh suatu rangkaian logika yang implementasinya hanya menggunakan satu jenis gerbang saja.

# BAB IV
# RANGKAIAN LOGIKA KOMBINASI

Logika kombinasi merupakan rangkaian logika yang outputnya hanya tergantung pada kombinasi input-inputnya saja, dan tidak tergantung pada keadaan output sebelumnya atau rangkaian logika yang outputnya tidak tergantung pada waktu.

## A. Bentuk-bentuk Persamaan Logika

### 1. Bentuk S*um of Product* (SOP)

SOP merupakan persamaan logika yang berbentuk operasi OR dari suku-suku berbentuk operasi AND.

Contoh:

$X = \overline{A}\,\overline{B}C + A\,\overline{B}\,\overline{C} + A\,B\,\overline{C} + A\,B\,C$ **persamaan (6)**

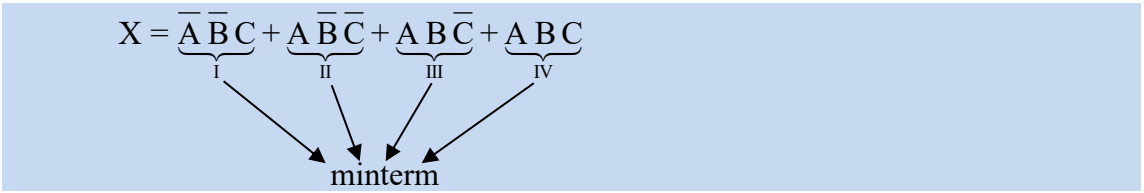
$Y = A\,\overline{B} + \overline{A}\,B$ **persamaan (7)**

$P = A\,\overline{B}C + \overline{A}\,B + B\,\overline{C} + \overline{A}$ **persamaan (8)**

$Q = \overline{A}\,B + \overline{B}\,C + \overline{A}\,\overline{C}$ **persamaan (9)**

Persamaan (6) dan persamaan (7) merupakan contoh SOP bentuk standar karena setiap sukunya mengandung semua variabel input yang ada, sedangkan persamaan (8) dan persamaan (9) contoh SOP bentuk tak standar karena tidak setiap sukunya mengandung semua variabel input.

Pada bentuk SOP standar, setiap sukunya dinamakan *minterm*, disingkat dengan m (huruf kecil). Perhatikan fungsi X berikut ini:




*Minterm* bersifat unik, yakni untuk semua kombinasi input yang ada hanya terdapat satu kombinasi saja yang menyebabkan suatu *minterm* bernilai 1. Misal jika terdapat input A=0, B=0, dan C=1 maka hanya terdapat sebuah *minterm* yang bernilai 1 yakni:

$\overline{A}\,\overline{B}C = \overline{0}\,\overline{0}1 = 1.1.1 = 1.$

Dengan demikian untuk suatu input yang memberikan nilai 1 pada salah satu *minterm* yang ada, fungsi SOP standar selalu bernilai 1. Karena pasangan input yang menyebabkan output bernilai 1 adalah 001 maka minterm tersebut yakni suku I dinamakan minterm 1 ($m_1$).
Jadi, fungsi X dapat ditulis:

$$X = \underbrace{\overline{A}\,\overline{B}\,C}_{m_1} + \underbrace{A\,\overline{B}\,\overline{C}}_{m_4} + \underbrace{A\,B\,\overline{C}}_{m_6} + \underbrace{A\,B\,C}_{m_7} \rightarrow \text{Cara Penulisan I} \qquad \textbf{persamaan (10)}$$

$$X = m_1 + m_4 + m_6 + m_7$$

$$X(A,B,C) = \sum m(1,4,6,7) \rightarrow \text{Cara Penulisan II} \qquad \textbf{persamaan (11)}$$

Perhatikan bahwa fungsi X akan bernilai 1 untuk input-input yang bernilai desimal 1, 4, 6, dan 7, sesuai dengan nama-nama *minterm* penyusunnya. Dengan demikian tabel kebenaran untuk fungsi X adalah:

**Tabel 15. Tabel kebenaran fungsi** $X = \overline{A}\,\overline{B}C + A\,\overline{B}\,\overline{C} + A\,B\,\overline{C} + A\,B\,C$

| INPUT | | | OUTPUT | |
|---|---|---|---|---|
| A | B | C | X | |
| 0 | 0 | 0 | 0 | |
| 0 | 0 | 1 | 1 | $m_1 = \overline{A}\,\overline{B}\,C$ |
| 0 | 1 | 0 | 0 | |
| 0 | 1 | 1 | 0 | |
| 1 | 0 | 0 | 1 | $m_4 = A\,\overline{B}\,\overline{C}$ |
| 1 | 0 | 1 | 0 | |
| 1 | 1 | 0 | 1 | $m_6 = AB\overline{C}$ |
| 1 | 1 | 1 | 1 | $m_7 = ABC$ |

## 2. Bentuk *Product of Sum* (POS)

POS merupakan persamaan logika berbentuk operasi AND dari suku-suku berbentuk operasi OR.

Contoh:

$$R = (\overline{A} + \overline{B} + C)(A + B + \overline{C})(A + \overline{B} + C)(\overline{A} + B + C) \qquad \textbf{persamaan (12)}$$

$$S = (\overline{A} + \overline{B})(A + B) \qquad \textbf{persamaan (13)}$$

$$T = (\overline{A} + \overline{B} + C)(\overline{A} + B + C)(A + \overline{C}) \qquad \textbf{persamaan (14)}$$

$$U = (A + B)(B + \overline{C})(\overline{A} + \overline{C}) \qquad \textbf{persamaan (15)}$$

Persamaan (12) dan persamaan (13) merupakan POS standar karena setiap sukunya mengandung semua variabel input yang ada, sedangkan persamaan (14) dan persamaan (15) adalah POS tak standar karena setiap sukunya mengandung semua variabel input.

Pada bentuk POS standar, setiap sukunya dinamakan *maxterm*, disingkat dengan M (huruf besar). Perhatikan fungsi R berikut ini:

$$R = \underbrace{(\overline{A}+\overline{B}+C)}_{I}\underbrace{(A+B+\overline{C})}_{II}\underbrace{(A+\overline{B}+C)}_{III}\underbrace{(\overline{A}+B+C)}_{IV}$$

Maxterm

Seperti halnya *minterm*, *maxterm* juga bersifat unik Dalam hal ini, untuk semua kombinasi input hanya terdapat satu kombinasi saja yang menyebabkan suatu *maxterm* bernilai 0. Misal jika terdapat input A=1, B=1, dan C=0 maka hanya terdapat sebuah *maxterm* yang bernilai 0 yakni:

$$\overline{A}+\overline{B}+C = \overline{1}+\overline{1}+0 = 0+0+0 = 0.$$

Jadi, untuk suatu input yang memberikan nilai 0 pada salah satu *maxterm* yang ada, fungsi POS standar selalu bernilai 0. Karena pasangan input yang menyebabkan output bernilai 0 salah satunya adalah 110 maka *maxterm* tersebut yakni suku I dinamakan *maxterm* 6 ($M_6$). Jadi R dapat ditulis:

$$R = \underbrace{(\overline{A}+\overline{B}+C)}_{M_6}\underbrace{(A+B+\overline{C})}_{M_1}\underbrace{(A+\overline{B}+C)}_{M_2}\underbrace{(\overline{A}+B+C)}_{M_4} \rightarrow \text{Cara penulisan I}$$

**persamaan (16)**

$$R = M_6 \bullet M_1 \bullet M_2 \bullet M_4$$

$$R(A,B,C) = \prod M(1,2,4,6) \rightarrow \text{Cara penulisan II}$$ **persamaan (17)**

Tabel kebenaran untuk R dapat disusun dengan mudah sebagai berikut:

**Tabel 16. Tabel kebenaran fungsi** $R = (\overline{A}+\overline{B}+C)(A+B+\overline{C})(A+\overline{B}+C)(\overline{A}+B+C)$

| INPUT | | | OUTPUT | |
|---|---|---|---|---|
| A | B | C | R | |
| 0 | 0 | 0 | 1 | |
| 0 | 0 | 1 | 0 | $M_1 = A+B+\overline{C}$ |
| 0 | 1 | 0 | 0 | $M_2 = A+\overline{B}+C$ |
| 0 | 1 | 1 | 1 | |
| 1 | 0 | 0 | 0 | $M_4 = \overline{A}+B+C$ |
| 1 | 0 | 1 | 1 | |
| 1 | 1 | 0 | 0 | $M_6 = \overline{A}+\overline{B}+C$ |
| 1 | 1 | 1 | 1 | |

## B. Mengubah Fungsi Bentuk Tak Standar Menjadi Bentuk Standar

Jika suatu persamaan logika bentuk SOP Tak Standar ingin diubah menjadi bentuk SOP Standar, maka dapat dilakukan dengan cara seperti contoh berikut ini:

Ubahlah fungsi:

a.. $Y = A\,\overline{B}\,\overline{C} + \overline{B}\,C$

b. $Y = A\,B + \overline{C}$

menjadi bentuk standar!

Jawab:

a. $Y = A\,\overline{B}\,\overline{C} + \overline{B}\,C$

$Y = A\,\overline{B}\,\overline{C} + \overline{B}\,C(A + \overline{A})$ Ingat : $A + \overline{A} = 1$

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

$Y(A,B,C) = \sum m(1,4,5)$

b. $Y = A\,B + \overline{C}$

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

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

$Y = A\,B\,C + \underbrace{A\,B\,\overline{C} + A\,B\,\overline{C}}_{A\,B\,\overline{C}} + \overline{A}\,B\,\overline{C} + A\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,\overline{C}$

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

$Y(A,B,C) = \sum m(0,2,4,6,7)$

## C. Memperoleh Persamaan Bentuk Standar dari Tabel Kebenaran

Contoh: tuliskan persamaan logika bentuk SOP Standar dan POS Standar yang diperoleh dari tabel kebenaran berikut ini!

**Tabel 17. Tabel kebenaran yang akan ditentukan persamaan logikanya**

| INPUT | | OUTPUT | |
|---|---|---|---|
| A | B | Y | |
| 0 | 0 | 0 | $\rightarrow M_0 = A + B$ |
| 0 | 1 | 1 | $\rightarrow m_1 = \overline{A}\,B$ |
| 1 | 0 | 1 | $\rightarrow m_2 = A\,\overline{B}$ |
| 1 | 1 | 0 | $\rightarrow M_3 = \overline{A} + \overline{B}$ |

Dari tabel 17, persamaan logika yang diperoleh adalah:

a. Bentuk SOP Standar :

$Y = m_1 + m_2$

$Y = \overline{A}\,B + A\,\overline{B}$

$Y(A,B) = \sum m(1,2)$

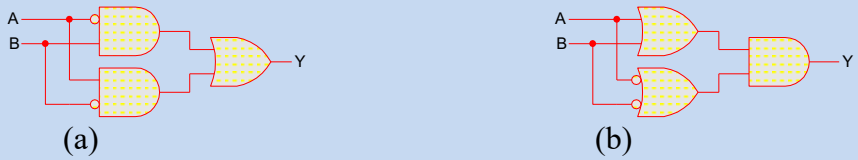
b. Bentuk POS Standar :

$Y = M_0 \,.\, M_3$

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

$Y(A,\ B) = \prod\ M(0,3)$

Rangkaian untuk kedua bentuk tersebut adalah:




(a) (b)

**Gambar 52. Rangkaian untuk (a) $Y(A,B) = \sum m(1,2)$ dan (b) $Y(A, B) = \prod M(0,3)$**

## D. Penyederhanaan Secara Aljabar

Untuk mempermudah proses implementasi rangkaian logika, langkah pertama yang perlu dilakukan adalah mengasumsikan bahwa setiap rangkaian logika memiliki bentuk yang tidak efisien, selanjutnya dilakukan pengujian bentuk minimumnya, jika belum minimum diteruskan dengan penyederhaan, dan akhirnya diimplementasikan.

Contoh:

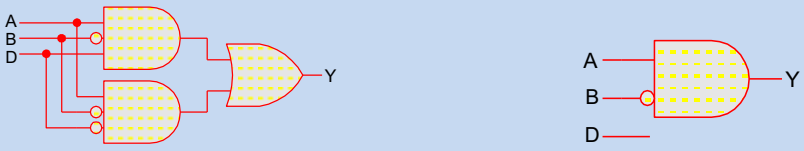
1. Sederhanakan persamaan $Y = A\,\overline{B}\,D + A\,\overline{B}\,\overline{D}$!

Jawab : $Y = A\,\overline{B}\,D + A\,\overline{B}\,\overline{D}$

$= A\,\overline{B}(D + \overline{D})$

$\therefore Y = A\,\overline{B}$ Ingat : $D + \overline{D} = 1$

Rangkaiannya adalah:




(a) (b)

**Gambar 53. (a) Rangkaian $Y = A\,\overline{B}\,D + A\,\overline{B}\,\overline{D}$ yang belum disederhanakan, (b) rangkaian yang telah disederhanakan**

Perhatikan bahwa persamaan Y dimanipulasi sedemikian rupa sehingga sukunya mengandung faktor $(D + \overline{D})$. Faktor tersebut dapat dieliminasi atau dihilangkan karena menurut teorema komplemen (teorema 7), faktor $(D + \overline{D})$ bernilai 1. Hal ini merupakan

salah satu cara minimalisasi rangkaian dengan Aljabar Boole yakni dengan mengarahkan persamaan agar mengandung faktor seperti pada teorema 7.

Untuk membuktikan watak kedua rangkaian itu sama, disusun tabel kebenaran keduanya sebagai berikut.

**Tabel 18. Table kebenaran** $Y = A\,\overline{B}\,D + A\,\overline{B}\,\overline{D}$ dan $Y = A\,\overline{B}$

| INPUT | | | OUTPUT TIAP GERBANG | | | | OUTPUT | |
|---|---|---|---|---|---|---|---|---|
| A | B | D | $\overline{B}$ | $\overline{D}$ | $A\,\overline{B}\,D$ | $A\,\overline{B}\,\overline{D}$ | $Y = A\,\overline{B}\,D + A\,\overline{B}\,\overline{D}$ | $Y = A\,\overline{B}$ |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

2. Sederhanakan persamaan $X = (\overline{A} + B)(A + B)$!

Jawab : $X = (\overline{A} + B)(A + B)$

$= \overline{A}\,A + \overline{A}\,B + B\,A + B\,B$

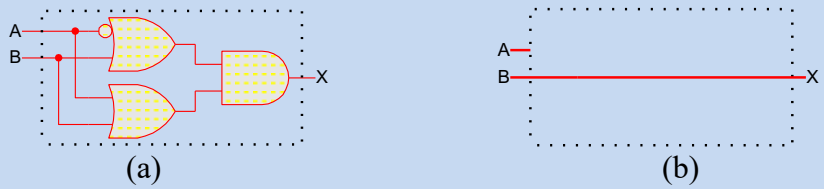
$= 0 + \overline{A}\,B + B\,A + B$ ingat $\overline{A}\,A = 0$, dan $B\,B = B$

$= \overline{A}\,B + A\,B + B$

$= B(A + \overline{A} + 1)$

$\therefore X = B$ ingat $A + \overline{A} + 1 = 1$

Rangkaiannya:




(a) (b)

**Gambar 54. (a) Rangkaian** $X = (\overline{A} + B)(A + B)$ **yang belum disederhanakan, (b) rangkaian yang telah disederhanakan**

Pada contoh 2 ditunjukkan bahwa persamaan dimanipulasi sedemikian rupa sehingga sukunya mengandung faktor $(A + \overline{A} + 1)$. Berdasarkan teorema 1 atau teorema satu dan nol, faktor tersebut bernilai 1 sehingga dapat dieliminasi.

3. Sederhanakan persamaan $Z = ACD + \overline{A}BCD$!

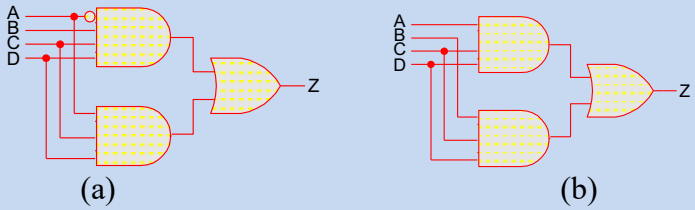
Jawab : $Z = ACD + \overline{A}BCD$

$= CD(A + \overline{A}B)$

$= CD(A + B)$ ingat $A + \overline{A}B = A + B$

$\therefore Z = ACD + BCD$

Rangkaiannya:




(a) (b)

**Gambar 55. (a) Rangkaian $Z = ACD + \overline{A}BCD$ yang belum disederhanakan, (b) rangkaian yang telah disederhanakan**

Pada contoh 3, persamaan diarahkan agar sukunya mengandung faktor berbentuk $(A + \overline{A}B)$. Menurut teorema absorpsi atau teorema 18, faktor $(A + \overline{A}B)$ sama dengan (A+B).

4. Sederhanakan persamaan $Y = \overline{(\overline{A} + C)(B + \overline{D})}$!

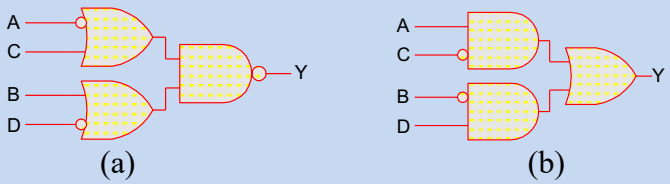
Jawab : $Y = \overline{(\overline{A} + C)(B + \overline{D})}$

$= \overline{(\overline{A} + C)} + \overline{(B + \overline{D})}$ ingat teorema de Morgan

$= (\overline{\overline{A}} . \overline{C}) + (\overline{B} . \overline{\overline{D}})$ ingat teorema de Morgan

$\therefore Y = A\overline{C} + \overline{B}D$

Rangkaiannya:




(a) (b)

**Gambar 56. (a) Rangkaian $Y = \overline{(\overline{A} + C)(B + \overline{D})}$ yang belum disederhanakan, (b) rangkaian yang telah disederhanakan**

5. Sederhanakan persamaan $Z = ABC + A\overline{B}(\overline{\overline{A}\,\overline{C}})$!

Jawab : $Z = ABC + A\overline{B}(\overline{\overline{A}\,\overline{C}})$

$Z = ABC + A\overline{B}(\overline{\overline{A}} + \overline{\overline{C}})$ ingat teorema de Morgan!

$Z = ABC + A\overline{B}(A + C)$ ingat operasi komplemen ganda!

$Z = ABC + A\overline{B}A + A\overline{B}C$

$Z = ABC + A\overline{B} + A\overline{B}C$ ingat teorema idempoten $AA = A$

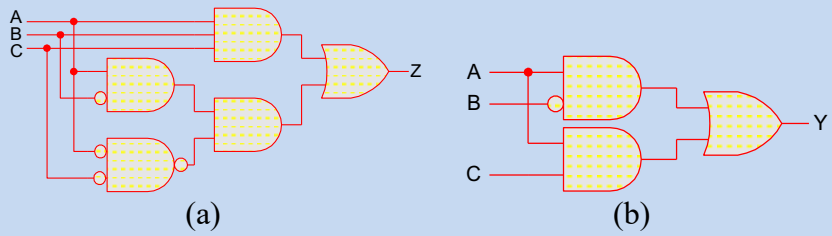
$Z = A\overline{B} + (A\overline{B}C + ABC)$

$Z = A\overline{B} + AC(\overline{B} + B)$

$Z = A\overline{B} + AC$ ingat $\overline{B} + B = 1$

$\therefore Z = A(\overline{B} + C)$

Rangkaiannya:




**Gambar 57. (a) Rangkaian $Z = ABC + A\overline{B}(\overline{\overline{A}\,\overline{C}})$ yang belum disederhanakan, (b) rangkaian yang telah disederhanakan**

6. Sederhanakan fungsi $Y = ABC + AB\overline{C} + A\overline{B}C$!

Jawab :

Cara I : $Y = ABC + AB\overline{C} + A\overline{B}C$

$= AB(C + \overline{C}) + A\overline{B}C$

$= AB(1) + A\overline{B}C$ ingat teorema komplemen : $C + \overline{C} = 1$

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

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

$= A(B + C)$ ingat teorema distributif : $B + \overline{B}C = B + C$

$\therefore Y = AB + AC$

Cara II : $Y = ABC + AB\overline{C} + A\overline{B}C$

$= ABC + AB\overline{C} + A\overline{B}C + ABC$ ingat teorema idempoten : $ABC + ABC = ABC$

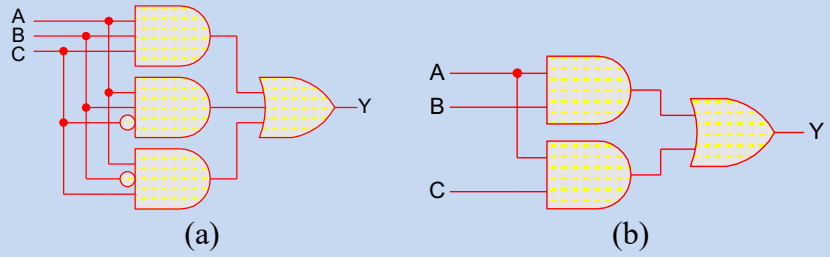
$= (ABC + AB\overline{C}) + (A\overline{B}C + ABC)$

$= AB(C + \overline{C}) + AC(\overline{B} + B)$

$= AB(1) + AC(1)$ ingat teorema komplemen $\overline{B} + B = 1$ dan $C + \overline{C} = 1$

$\therefore Y = AB + AC$ atau $Y = A(B + C)$

Rangkaiannya:




**Gambar 58. (a) Rangkaian $Y = ABC + A\bar{B}\bar{C} + \bar{A}\bar{B}C$ yang belum disederhanakan, (b) rangkaian yang telah disederhanakan**

## E. Metode Peta Karnaugh

### 1. Memperoleh Bentuk Minimum dari Persamaan yang Diketahui

Langkah pertama cara memperoleh bentuk minimum dari persamaan logika menggunakan metode Peta Karnaugh (Peta-K) adalah dengan memastikan bahwa persamaan tersebut dalam bentuk standar.

Contoh: Sederhanakan fungsi $Y = \bar{A}BC + A\bar{B}C + ABC + B\bar{C}$ dengan menggunakan Aljabar Boole dan Peta Karnaugh!

**Jawab:**

**Dengan menggunakan Aljabar Boole:**

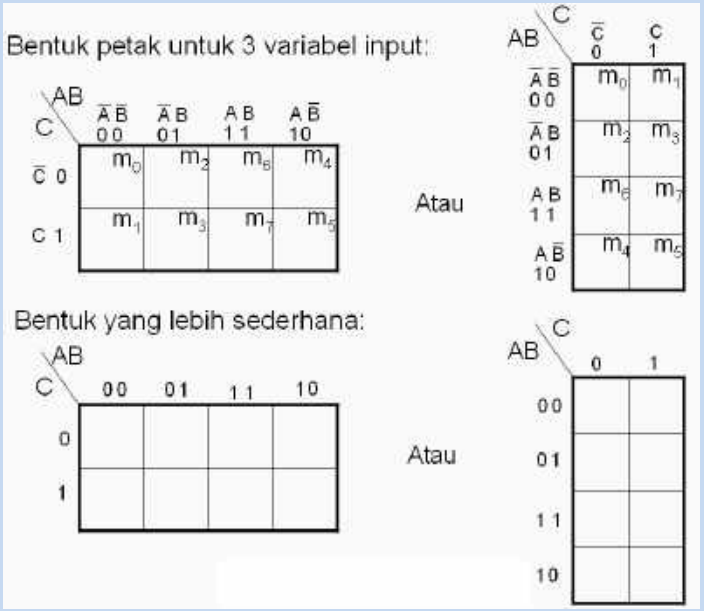
$$Y = \bar{A}BC + A\bar{B}C + ABC + B\bar{C}$$
$$Y = \bar{A}BC + A\bar{B}C + ABC + B\bar{C} + ABC$$
$$Y = (\bar{A}BC + ABC) + (A\bar{B}C + ABC) + B\bar{C}$$
$$Y = BC(\bar{A} + A) + AC(\bar{B} + B) + B\bar{C}$$
$$Y = BC(1) + AC(1) + B\bar{C}$$
$$Y = BC + AC + B\bar{C}$$
$$Y = AC + (BC + B\bar{C})$$
$$Y = AC + B(C + \bar{C})$$
$$Y = AC + B(1)$$
$$Y = AC + B \qquad \textbf{persamaan (18)}$$

**Dengan metode Peta Karnaugh:**

a. **Langkah pertama**: memastikan bahwa persamaan yang akan disederhanakan berbentuk standar. Perhatikan bahwa bentuk persamaannya adalah SOP yang tidak standar yakni $Y = \overline{A}BC + A\overline{B}C + ABC + B\overline{C}$. Persamaan tersebut perlu distandarkan terlebih dahulu menjadi:

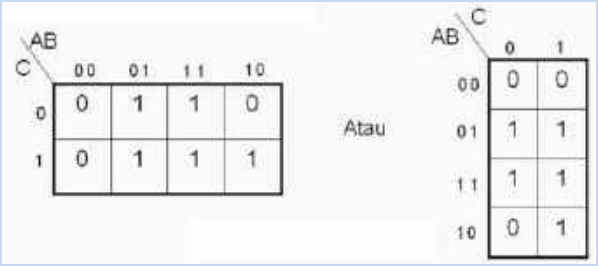
$$Y = \overline{A}BC + A\overline{B}C + ABC + B\overline{C}$$
$$Y = \overline{A}BC + A\overline{B}C + ABC + B\overline{C}(A + \overline{A})$$
$$Y = \overline{A}BC + A\overline{B}C + ABC + AB\overline{C} + \overline{A}\,B\overline{C}$$
$$Y = m_3 + m_5 + m_7 + m_6 + m_2$$
$$Y(A,B,C) = \sum m(2,3,5,6,7) \qquad \textbf{persamaan (19)}$$

b. **Langkah kedua**: menyusun petak-petak sebanyak $2^n$ dengan n adalah jumlah variabel input.




**Gambar 59. Bentuk petak pada Peta Karnaugh untuk 3 variabel input**

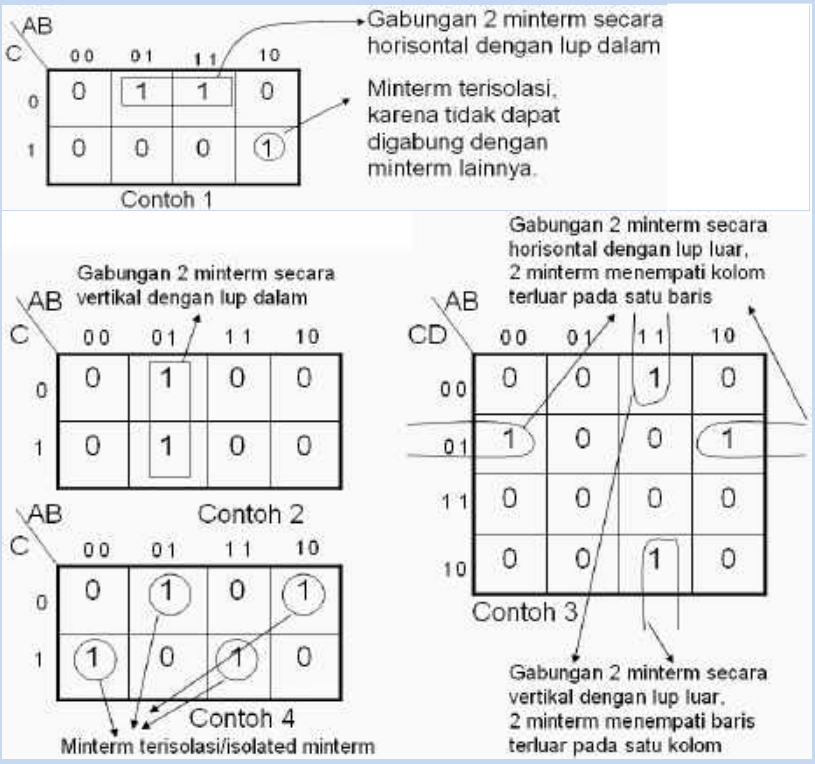
c. **Langkah ketiga**: memasukkan *minterm* persamaan ke dalam petak-petak yang sesuai.




**Gambar 60. Peta Karnaugh untuk persamaan** $Y(A,B,C) = \sum m(2,3,5,6,7)$

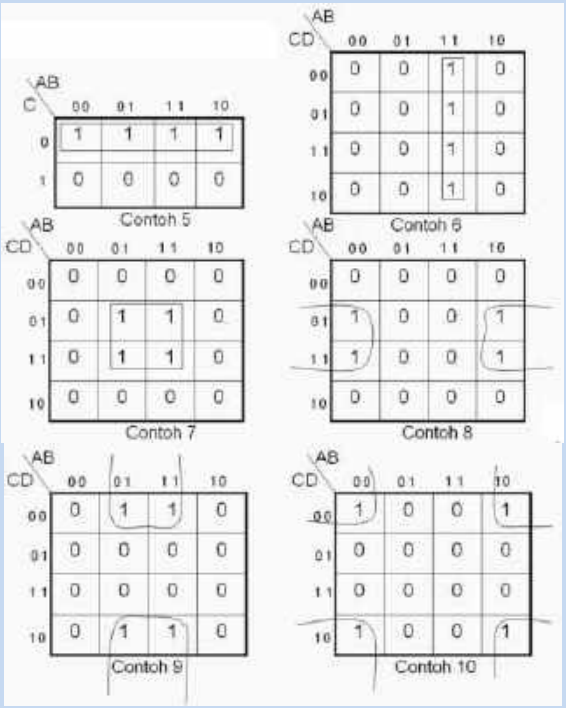
d. **Langkah keempat**: memberi tanda lup (kalang) pada setiap *minterm* yang terisolasi.

**1) Gabungan 2 minterm**




**Gambar 61. Contoh gabungan dua minterm dan minterm terisolasi**

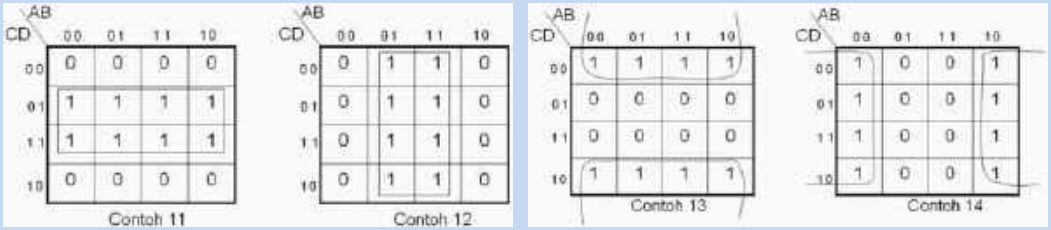
**2) Gabungan 4 minterm (Quad)**




**Gambar 62. Contoh peta Karnaugh yang mengandung *Quad***

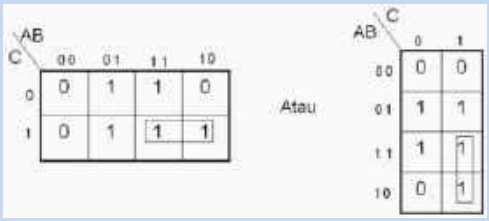
**3) Gabungan 8 minterm (Octet)**

Gabungan 8 minterm dapat ditemukan pada contoh 11, contoh 12, contoh 13, dan contoh 14 pada gambar berikut ini.



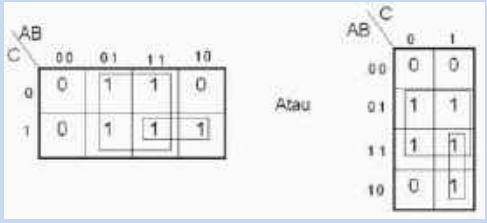

**Gambar 63. Contoh peta Karnaugh yang mengandung *Octet***

e. **Langkah kelima**: memberi tanda lup pada *minterm* yang hanya dapat bergabung dengan 1 *minterm* lainnya (gabungan dua *minterm*).




**Gambar 64. Tanda lup untuk gabungan dua minterm**

**f. Langkah keenam**: memberi tanda lup pada gabungan empat *minterm*.

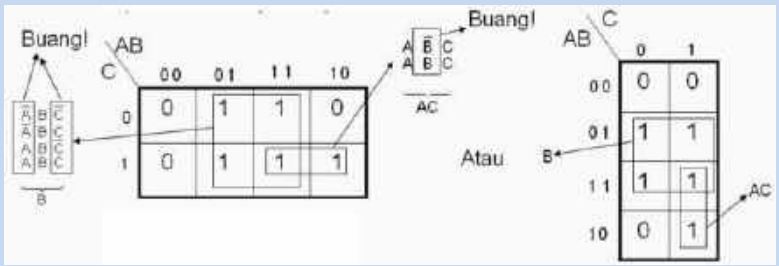



**Gambar 65. Tanda lup untuk gabungan empat minterm**

g. **Langkah ketujuh**: memberi tanda lup pada gabungan 8-*minterm*. Pada contoh tersebut tidak terdapat *octet*.

h. **Langkah kedelapan**: membuang variabel-variabel yang berbeda dan menggunakan variabel-variabel yang sama sebagai suku persamaan dari gabungan *minterm* yang diperoleh. Untuk *minterm* yang terisolasi, suku persamaannya tetap tidak mengalami reduksi.

i. **Langkah kesembilan:** membentuk persamaan minimum dengan cara melakukan operasi OR terhadap suku-suku persamaan yang diperoleh dari gabungan *minterm*.




**Gambar 66. Mereduksi suku persamaan pada gabungan *minterm***

Sesuai langkah kesembilan persamaan minimum yang dihasilkan adalah:

$$Y = AC + B \qquad \textbf{persamaan (20)}$$

## 2. Memperoleh Bentuk Minimum dari Tabel Kebenaran

Susunlah rangkaian yang paling minimum dari tabel kebenaran berikut ini!

**Tabel 19. Tabel kebenaran contoh penyederhanaan fungsi dengan peta Karnaugh**

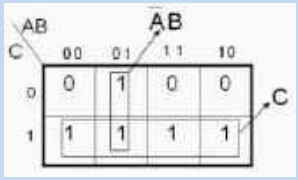
| INPUT | | | OUPUT | |
|---|---|---|---|---|
| A | B | C | Y | |
| 0 | 0 | 0 | 0 | |
| 0 | 0 | 1 | 1 | $m_1 = \bar{A}\bar{B}C$ |
| 0 | 1 | 0 | 1 | $m_2 = \bar{A}B\bar{C}$ |
| 0 | 1 | 1 | 1 | $m_3 = \bar{A}BC$ |
| 1 | 0 | 0 | 0 | |
| 1 | 0 | 1 | 1 | $m_5 = A\bar{B}C$ |
| 1 | 1 | 0 | 0 | |
| 1 | 1 | 1 | 1 | $m_7 = ABC$ |

**Jawab:**

Dari tabel 19 dapat diperoleh persamaan SOP standar:

$$Y(A,B,C) = \sum m(1,2,3,5,7) \qquad \textbf{persamaan (21)}$$

Peta Karnaugh untuk fungsi Y yang diperoleh dari tabel kebenaran tersebut adalah:




**Gambar 67. Peta Karnaugh untuk persamaan (21)**

Dari peta-K tersebut terlihat bahwa fungsi Y belum minimum, dan masih dapat disederhanakan menjadi:

$$Y = \bar{A}B + C \qquad \textbf{persamaan (22)}$$

## 3. Kondisi Diabaikan (Don't Care Condition)

Rangkaian-rangkaian logika yang telah dipelajari di muka hampir semuanya selalu memberikan output 1 atau 0 untuk suatu kombinasi input yang diberikan. Selain itu, terdapat pula rangkaian logika dengan beberapa kombinasi input yang dalam kenyataannya tidak pernah ada. Contoh untuk rangkaian-rangkaian logika dengan input kode BCD, inputnya hanya ada 10 kombinasi yakni 0000 sampai dengan 1001, karena kode BCD memang hanya merepresentasikan bilangan desimal yang memiliki nilai 0 sampai dengan 9. Dengan kata lain, terdapat 6 buah kombinasi input yang tidak pernah ada yakni

1010 sampai dengan biner 1111. Untuk rangkaian yang input-inputnya tidak pernah ada, outputnya tidak dinyatakan dalam nilai 1 atau 0 melainkan diberi tanda X yang berarti keadaan diabaikan (*don't care condition*). Tabel 20 berikut ini menunjukkan watak rangkaian detektor bilangan prima dengan input kode BCD.

**Tabel 20. Contoh tabel kebenaran yang mengandung keadaan diabaikan: Detektor Bilangan Prima Dengan Input Kode BCD**

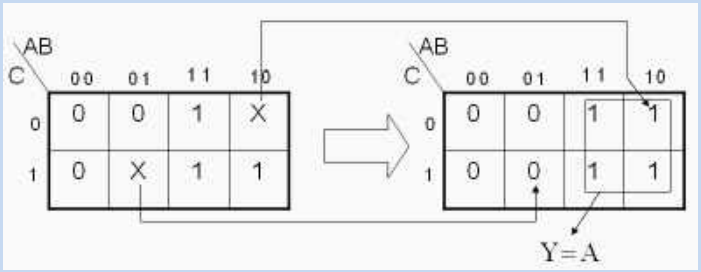
| INPUT | | | | OUTPUT |
|---|---|---|---|---|
| A | B | C | D | Y |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | X |
| 1 | 0 | 1 | 1 | X |
| 1 | 1 | 0 | 0 | X |
| 1 | 1 | 0 | 1 | X |
| 1 | 1 | 1 | 0 | X |
| 1 | 1 | 1 | 1 | X |

Contoh lain dari tabel kebenaran yang mengandung keadaan diabaikan ditunjukkan pada tabel berikut ini.

**Tabel 21. Contoh lain tabel kebenaran dengan kondisi diabaikan**

| INPUT | | | OUTPUT |
|---|---|---|---|
| A | B | C | Y |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | X |
| 1 | 0 | 0 | X |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

Peta Karnaugh untuk tabel 21 ditunjukkan pada gambar 68. Pada peta Karnaugh yang mengandung kondisi diabaikan, X dapat dipilih bernilai 0 atau 1. Pemilihan nilai X dilakukan sedemikian rupa sehingga dari peta Karnaugh tersebut dapat diperoleh persamaan yang paling sederhana.




(a) (b)

**Gambar 68. Peta Karnaugh untuk tabel 21**

## F. Bentuk NAND dan NOR Rangkaian Logika

Persamaan yang diperoleh melalui penyederhanaan baik menggunakan aljabar Boole maupun peta Karnaugh umumnya dalam bentuk SOP (AND-OR) atau bentuk POS (OR-AND). Contoh persamaan (20) yakni $Y = AC + B$ adalah bentuk SOP hasil penyederhanaan dari persamaan $Y = \overline{A}BC + A\overline{B}C + ABC + B\overline{C}$. Selain dalam hal jumlah gerbang, penyederhanaan rangkaian logika juga dapat dilakukan dengan mengarahkan agar rangkaian hanya terdiri dari satu jenis gerbang saja. Gerbang NAND dan NOR memiliki sifat universal sehingga semua rangkaian logika dapat disusun hanya dengan gerbang NAND saja atau NOR saja. Cara mengubah bentuk SOP ke dalam bentuk NAND dapat dilakukan dengan cara sebagai berikut.

1. Pastikan bahwa persamaan dalam bentuk SOP
2. Lakukan operasi komplemen ganda
3. Berlakukan teorema de Morgan

Contoh: Ubahlah $Y = \overline{A}B + C$ menjadi bentuk NAND!

**Jawab:**

$Y = \overline{A}\,B + C \rightarrow$ sudah dalam bentuk SOP

$Y = \overline{\overline{A}\,B + C} \rightarrow$ operasi komplemen ganda

$Y = \overline{(\overline{A}\,B).(\overline{C})} \rightarrow$ de Morgan **persamaan (23)**

Bentuk NAND dari $Y = \overline{A}\,B + C$

Sedangkan untuk mengubah bentuk SOP atau POS ke dalam bentuk NOR saja dapat dilakukan dengan cara:

1. Pastikan bahwa persamaan dalam bentuk POS
2. Lakukan operasi komplemen ganda
3. Berlakukan teorema de Morgan

Contoh: Ubahlah $Y = \overline{A}\,B + C$ menjadi bentuk NOR!

**Jawab:**

$Y = \overline{A}\,B + C \rightarrow$ masih berbentuk SOP

Diubah menjadi POS :

$Y = (\overline{A} + C)(B + C) \rightarrow$ bentuk POS **persamaan (24)**

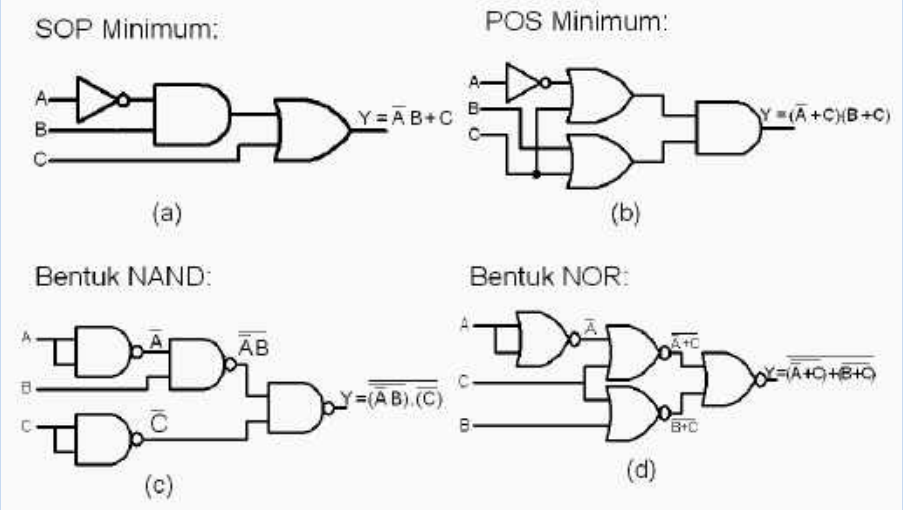
$\overline{\overline{(\overline{A} + C)(B + C)}} \rightarrow$ operasi komplemen ganda

$Y = \overline{\overline{(\overline{A} + C)} + \overline{(B + C)}} \rightarrow$ de Morgan

**persamaan (25)**

Bentuk NOR dari $Y = \overline{A}\,B + C$

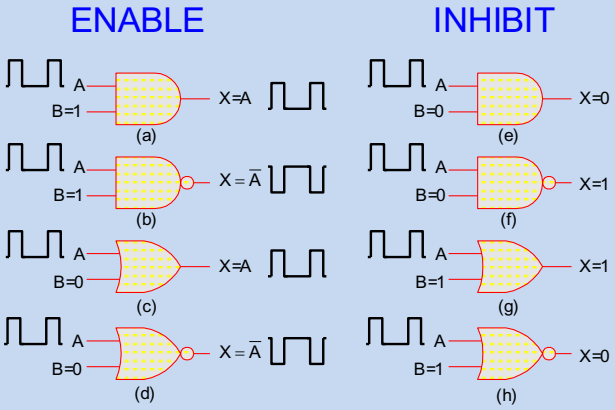
Dengan demikian, penyederhanaan fungsi $Y = \overline{A}BC + A\overline{B}C + ABC + B\overline{C}$ menghasilkan 4 bentuk rangkaian yakni bentuk SOP seperti pada persamaan (20), bentuk POS pada persamaan (24), bentuk NAND pada persamaan (23), dan bentuk NOR pada persamaaan (25). Implementasi dari rangkaian-rangkaian tersebut dapat digambarkan dalam bentuk rangkaian logika seperti pada gambar 69. Jika dilihat dari IC yang dibutuhkan, implementasi bentuk NAND dan NOR akan memberikan tingkat efisiensi yang tinggi karena keduanya, masing-masing hanya memerlukan satu buah IC saja.




**Gambar 69. Implementasi Persamaan Logika Dalam Bentuk SOP minimum, POS minimum, NAND, dan NOR**

## G. Rangkaian *Enable* dan *Inhibit*

Dalam implementasinya, rangkaian logika kadang kala perlu dilengkapi dengan rangkaian pengontrol yang berfungsi untuk meneruskan atau menghambat inputnya. Susunan gerbang yang memberikan fungsi meneruskan inputnya dinamakan rangkaian *enable*, dan susunan gerbang yang menyebabkan suatu input tidak dapat diteruskan dinamakan rangkaian *inhibit*.

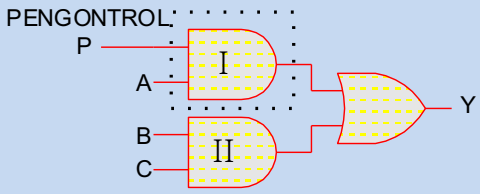



**Gambar 70. Rangkaian *enable* dan *inhibit***

Perhatikan contoh aplikasi dari rangkaian *enable* dan *inhibit* berikut ini! Rancanglah rangkaian yang dapat melakukan operasi Y=A+BC untuk pengontrol P bernilai tinggi (P=1) dan operasi Y=BC untuk pengontrol P yang bernilai rendah (P=0).

**Jawab:**

Disusun terlebih dahulu rangkaian Y=A+BC. Selanjutnya, pada input A diberi rangkaian *enable* dan *inhibit* menggunakan gerbang AND.




**Gambar 71. Rangkaian Y=A+BC untuk P=1 dan Y=BC untuk P=0**

Jika P bernilai tinggi (P=1) maka akan terjadi keadaan *enable* yakni input A diteruskan ke output gerbang AND I sehingga output Y merupakan fungsi Y=A+BC. Namun, jika P=0, gerbang AND I bersifat *inhibit* dan output gerbang AND I bernilai rendah, sehingga output rangkaian merupakan fungsi Y=BC.

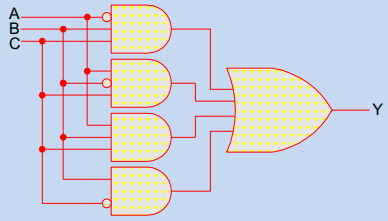
## H. Soal Latihan

1. Jelaskan pengertian *sum of product, standard sum of product, product of sum, standard product of sum, minterm* dan *maxterm*! Berikan contoh dari masing-masing pengertian tersebut!

2. Tuliskan persamaan SOP Standar dan POS Standar yang dihasilkan dari tabel kebenaran suatu rangkaian yang memberikan output 1 jika inputnya merupakan bilangan prima 3-bit!

3. Persamaan-persamaan berikut ini mana yang tidak termasuk bentuk SOP dan kemukakan alasannya:
   a. $W = RS\overline{T} + \overline{R}S\overline{T} + \overline{T}$
   b. $X = A\overline{DC} + \overline{A}DC$
   c. $Y = MN\overline{P} + (M + \overline{N})P$
   d. $Z = AB + \overline{A}B\overline{C} + A\overline{B}\,\overline{C}D$

4. Tuliskan persamaan SOP Standar dan POS Standar dari rangkaian yang menghasilkan persamaan SOP Tak Standar berikut ini:




**Gambar 72. Rangkaian untuk soal nomor 4 Bab IV**

5. Susunlah tabel kebenaran dari persamaan berikut ini:
   a. $X(A,B,C) = \sum m(5,6,7)$
   b. $Y(K,L,M,N) = \sum m(7,8,9,10)$
   c. $Z(A,B,C,D) = \sum m(1,15)$
   d. $P(W,X,Y,Z) = \sum m(0,1,4,5,10,11,14,15)$
   e. $Q(X_1,X_2,X_3) = \prod M(1,3,5,7)$
   f. $R(A,B,C,D) = \prod M(0,2,4,10,12,14)$
   g. $S(X_1,X_2,X_3,X_4) = \prod M(2,3,7,9,15)$

6. Susun persamaan pada soal nomor 5 dalam bentuk penulisan cara II!

7. Susun persamaan berikut ini dalam bentuk penulisan cara I:
   a. $Y = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}BC + A\overline{B}\,\overline{C} + ABC$
   b. $Z = \overline{X}_1\overline{X}_2X_3\overline{X}_4 + \overline{X}_1X_2X_3\overline{X}_4 + X_1X_2X_3X_4$
   c. $S = \overline{K}LM\overline{N} + KLMN$
   d. $T = \overline{W}\,\overline{X}\,\overline{Y}\,\overline{Z} + \overline{W}X\overline{Y}Z + \overline{W}XYZ + W\,\overline{X}\,\overline{Y}Z + WX\overline{Y}\,\overline{Z} + WX\overline{Y}Z + WXYZ$
   e. $U = (A+B+C)\,(A+\overline{B}+C)(A+\overline{B}+\overline{C})\;\;(\overline{A}+B+\overline{C})(\overline{A}+\overline{B}+C)$
   f. $W = (\overline{A}+\overline{B}+C+D)\;\;(A+B+\overline{C}+\overline{D})$
   g. $X = (Y_1+Y_2+Y_3+Y_4)(Y_1+\overline{Y}_2+Y_3+Y_4)(Y_1+\overline{Y}_2+\overline{Y}_3+\overline{Y}_4)(\overline{Y}_1+Y_2+\overline{Y}_3+Y_4)$

8. Ubahlah bentuk persamaan logika berikut ini menjadi bentuk standar!
   a. $X = A + BC$
   b. $Y = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}BC + B\,\overline{C}$
   c. $Z = (A+\overline{B})(\overline{A}+C)$

9. Tulislah persamaan logika bentuk SOP standar dan POS standar yang diperoleh dari tabel kebenaran berikut ini!

Tabel 22

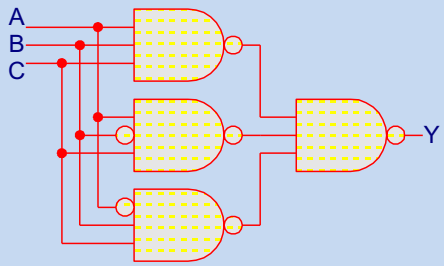
| INPUT | | | OUTPUT |
|---|---|---|---|
| A | B | C | Y |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

Tabel 23

| INPUT | | | OUTPUT |
|---|---|---|---|
| A | B | C | Y |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

10. Sederhanakan fungsi berikut ini dengan menggunakan aljabar Boole:
    a. $X = \overline{A}C(\overline{\overline{A}BD}) + \overline{A}\,B\,\overline{C}\,\overline{D} + A\overline{B}C$
    b. $Y = (\overline{A} + B)(A + B + D)\overline{D}$
    c. $Z = A\overline{B}C + \overline{A}BD + \overline{CD}$
    d. $X = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}BC + ABC + A\overline{B}\,\overline{C} + A\overline{B}C$
    e. $Y = (B + \overline{C})(\overline{B} + C) + \overline{\overline{A} + B + \overline{C}}$
    f. $Z = \overline{(C + D)} + \overline{A}C\overline{D} + A\overline{B}\,\overline{C} + \overline{A}\,\overline{B}CD + AC\overline{D}$

11. Perhatikan rangkaian berikut ini:




**Gambar 73. Rangkaian untuk soal nomor 11 Bab IV**

Tulislah persamaan output rangkaian pada gambar 73, dan sederhanakan persamaan tersebut dengan menggunakan aljabar Boole!

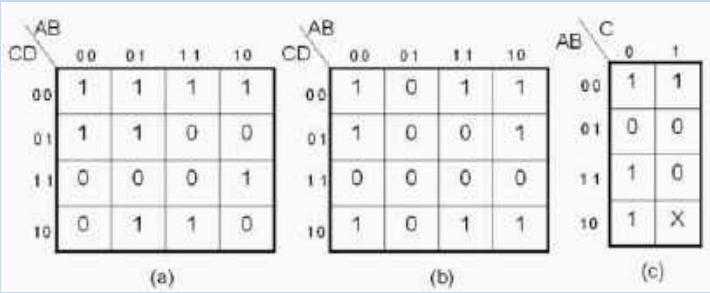
12. Sederhanakan persamaan $X = \overline{A}\,\overline{B}\,C + \overline{A}BC + ABC + A\overline{B}\,\overline{C} + AC$ dengan peta Karnaugh! Susunlah persamaan yang sudah sederhana tersebut ke dalam bentuk NAND saja dan NOR saja! Gambarkan semua bentuk rangkaian yang telah disederhanakan tersebut!

13. Ubahlah persamaan $Y = (A + \overline{B})(A + C)$ ke dalam bentuk NAND saja dan ke dalam bentuk NOR saja! Gambarkan ketiga bentuk rangkaian tersebut!

14. Sederhanakan dengan menggunakan peta Karnaugh persamaan yang dihasilkan dari gambar 73! Bandingkan hasilnya dengan jawaban soal nomor 11!

15. Sederhanakan fungsi standar yang dihasilkan dari tabel 22 dan tabel 23 menggunakan peta Karnaugh dan gambarkan rangkaian hasil penyederhanaannya!

16. Perhatikan peta Karnaugh berikut ini!




**Gambar 74. Peta Karnaugh untuk soal nomor 16 Bab IV**

Tentukan bentuk minimum dari peta Karnaugh pada gambar 74 (a), (b), dan (c)!

17. Dengan menggunakan rangkaian *enable* dan *inhibit*, rancanglah rangkaian logika dengan input A, pengontrol B, dan output X dan Y yang beroperasi:
    a. Ketika B=0, output X mengikuti input A, dan output Y bernilai 0
    b. Ketika B=1, output X akan bernilai 0, dan output Y akan mengikuti A.

18. Untuk soal nomor 18 dan 19, pilih satu jawaban yang paling tepat dari pilihan yang ada! Bentuk paling sederhana dari persamaan
    $Y = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,B\,\overline{C} + A\,B\,\overline{C} + A\,\overline{B}\,\overline{C} + \overline{A}\,BC$ adalah:
    a. $Y = \overline{A}\ B + \overline{C}$
    b. $Y = AB + C$
    c. $Y = A + \overline{B}\,C$
    d. $Y = A\overline{B}\overline{C} + C$
    e. $Y = A + BC$

19. Bentuk NAND dari persamaan pada soal nomor 18 yang paling tepat adalah:
    a. $Y = \overline{\overline{A} \bullet \overline{BC}}$
    b. $Y = \overline{\overline{\overline{A}\ B} \bullet C}$
    c. $Y = \overline{\overline{\overline{A}\ B} \bullet \overline{C}}$
    d. $Y = \overline{A \bullet \overline{\overline{B}C}}$
    e. $Y = \overline{\overline{\overline{A}\ B} \bullet C}$

**KOMPETENSI DASAR V**

1. Mahasiswa memahami watak dan cara kerja modul-modul logika kombinasi mencakup komparator, *half adder,* dan *full adder*
2. Mahasiswa dapat memahami watak dan cara kerja modul-modul logika kombinasi multiplekser, demultiplekser, enkoder dan dekoder

**TUJUAN PEMBELAJARAN V**

1. mendefinisikan pengertian rangkaian komparator (XOR dan XNOR), half adder, full adder, multiplekser, demultiplekser, enkoder dan dekoder
2. menyusun tabel kebenaran dan persamaan logika dari rangkaian komparator (XOR dan XNOR), half adder, full adder, multiplekser, demultiplekser, enkoder dan dekoder
3. menggambarkan rangkaian komparator (XOR dan XNOR), half adder, full adder, multiplekser, demultiplekser, enkoder dan dekoder dalam bentuk SOP, POS, NAND, NOR serta simbol
4. menjelaskan cara kerja rangkaian komparator (XOR dan XNOR), half adder, full adder, multiplekser, demultiplekser, enkoder dan dekoder
5. menyebutkan seri IC TTL XOR, XNOR, full adder, multiplekser, demultiplekser, enkoder dan dekoder serta menggambarkan susunan pin yang tersedia

**GARIS BESAR MATERI V**

Pada bagian ini akan dibahas beberapa rangkaian logika kombinasi yang disediakan dalam bentuk rangkaian terpadu atau *integrated circuits* (IC). Rangkaian-rangkaian kombinasi itu antara lain pembanding atau komparator, penjumlah (*adder*), *multiplexer*, *demultiplexer*, *encoder* dan *decoder*. Penjelasan tentang rangkaian-rangkaian tersebut dilakukan dengan mendefinisikan terlebih dahulu watak dari setiap rangkaian yang akan dipelajari. Selanjutnya, berdasarkan definisi disusun tabel kebenaran dan dari tabel kebenaran diturunkan persamaan logikanya dan akhirnya dideskripsikan implementasinya atau bentuk rangkaiannya. Dengan cara tersebut diharapkan Anda dapat memperoleh kemudahan di dalam mempelajari rangkaian-rangkaian logika kombinasi.

Rangkaian logika kombinasi pertama yang akan dipelajari adalah komparator 1-bit. Komparator merupakan rangkaian logika yang berfungsi membandingkan keadaan input-inputnya. Komparator jenis *non-equality* akan memberikan output tinggi jika keadaan input-inputnya berbeda dan komparator jenis *equality* akan memberikan output tinggi jika keadaan input-inputnya sama. Rangkaian komparator sangat penting peranannnya di dalam perancangan rangkaian logika kombinasi karena digunakan sebagai landasan pembangunan rangkaian *adder*, dan rangkaian *adder* merupakan dasar dari rangkaian aritmetika yang terdapat di dalam sistem komputer digital. Melalui bagian ini Anda akan diperkenalkan dengan kedua jenis komparator tersebut.

Rangkaian kedua yang akan dipelajari adalah penjumlah (*adder*). Terdapat dua jenis penjumlah yakni *half adder* dan *full adder*. *Half adder* merupakan penjumlah yang tidak menyertakan bawaan sebelumnya, sedangkan *full adder* adalah penjumlah yang menyertakan bawaan sebelumnya. Penjelasan bagian ini dimulai dari rangkaian *adder* 1-bit dan diakhiri dengan penjelasan *full adder* paralel n-bit yang merupakan dasar bagi pengembangan rangkaian aritmetika.

Melalui bagian ini anda akan diperkenalkan juga dengan rangkaian *multiplexer* dan *demultiplexer*. Rangkaian *multiplexer* berfungsi memilih data digital yang ada pada inputnya untuk diteruskan ke outputnya, sedangkan *demultiplexer* berfungsi mendistribusikan data digital yang ada pada inputnya ke salah satu dari beberapa saluran output yang tersedia.

Data yang akan dikirim dari satu lokasi ke lokasi yang lain dalam suatu rangkaian/sistem digital perlu disandikan atau dikodekan terlebih dahulu. Bagian ini akan memperkenalkan pula kepada Anda rangkaian *encoder* yang berfungsi membangkitkan kode dari suatu data yang ada. Demikian pula ketika kode tersebut ingin dibaca diperlukan pengubah kode. Anda juga akan diperkenalkan dengan rangkaian *decoder* yang berfungsi mengubah suatu kode menjadi data aslinya.

Bagian akhir dari bab ini akan menjelaskan implementasi berbagai rangkaian logika kombinasi dengan menggunakan IC *multiplexer* dan *decoder*.

# BAB V
# LOGIKA KOMBINASI DALAM KEMASAN IC

## A. Komparator

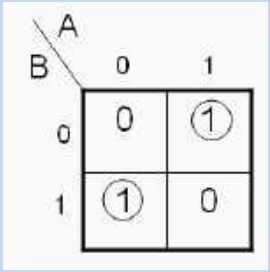
Komparator biner merupakan rangkaian logika yang berfungsi membandingkan keadaan logika input-inputnya. Jenis komparator biner terdiri atas *non-equality comparator* dan *equality comparator*.

### 1. Non-Equality Comparator

*Non-equality comparator* didefinisikan sebagai rangkaian logika yang memberikan keadaan output tinggi jika keadaan input-inputnya berbeda. Berdasarkan definisi tersebut dapat disusun tabel kebenaran untuk *non-equality comparator* sebagai berikut:

**Tabel 24. Tabel kebenaran *non-equality comparator***

| INPUT | | OUTPUT | |
|---|---|---|---|
| A | B | Y | |
| 0 | 0 | 0 | $M_0 = A + B$ |
| 0 | 1 | 1 | $m_1 = \overline{A}B$ |
| 1 | 0 | 1 | $m_2 = A\overline{B}$ |
| 1 | 1 | 0 | $M_3 = \overline{A} + \overline{B}$ |




**Gambar 75. Peta Karnaugh *non-equality comparator***

Persamaan output dari *non-equality comparator* dalam bentuk SOP minimum adalah

$$Y = \overline{A}B + A\overline{B}$$
$$Y(A,B) = \sum m(1,2) \qquad \text{persamaan (26)}$$

dan dalam bentuk POS minimum adalah

$$Y = (A + B)(\overline{A} + \overline{B})$$
$$Y(A,B) = \prod M(0,3) \qquad \text{persamaan (27)}$$

Dengan melakukan operasi komplemen ganda dan memberlakukan teorema de Morgan terhadap persamaan (26) dan persamaan (27) maka dapat diperoleh bentuk NAND dan NOR. Bentuk NAND diperoleh dengan cara sebagai berikut:

$$Y = \overline{A}\,B + A\,\overline{B}$$
$$Y = \overline{\overline{\overline{A}\,B + A\,\overline{B}}}$$
$$Y = \overline{\overline{\overline{A}\,B}\,.\,\overline{A\,\overline{B}}} \qquad \text{persamaan (28)}$$

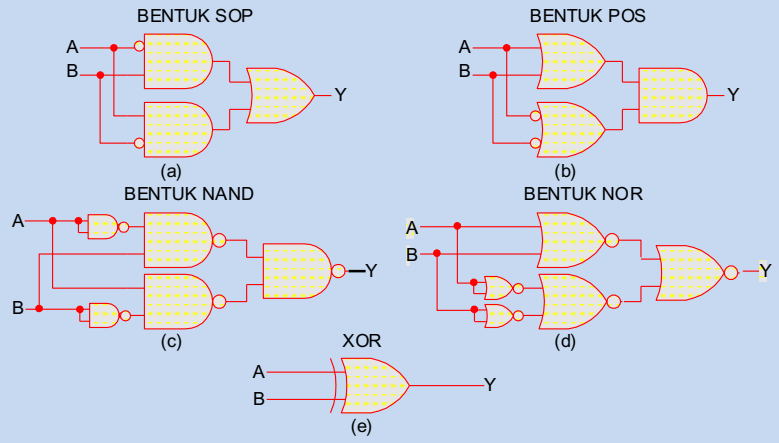
Sedangkan bentuk NOR diperoleh dengan cara:

$$Y = (A + B)(\overline{A} + \overline{B})$$
$$Y = \overline{\overline{(A + B)(\overline{A} + \overline{B})}}$$
$$Y = \overline{\overline{(A + B)} + \overline{(\overline{A} + \overline{B})}} \qquad \text{persamaan (29)}$$

Kecuali dapat diimplementasikan dengan bentuk SOP, POS, NAND, dan NOR, *non-equality comparator* juga dapat diimplementasikan dengan gerbang *exclusive-OR* (EXOR atau XOR). Simbol gerbang XOR ditunjukkan pada gambar 76 (e), dan persamaan logikanya adalah:

$$Y = A \oplus B \qquad \text{persamaan (30)}$$

Implementasi *non-equality comparator* dalam berbagai bentuk ditunjukkan pada gambar 76 berikut ini.




**Gambar 76. Berbagai bentuk implementasi *non-equality comparator***

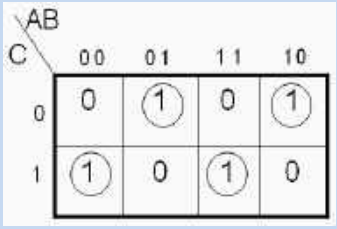
Jika dicermati tabel kebenarannya (tabel 24), kecuali membentuk fungsi sebagai komparator, output gerbang XOR juga membentuk fungsi sebagai detektor jumlah ganjil. Coba susun tabel kebenaran rangkaian detektor jumlah ganjil dengan 3-input, bagaimana bentuk rangkaiannya?

**Jawab:**

Disusun terlebih dahulu tabel kebenaran seperti ditunjukkan pada tabel 25 dan peta Karnaugh seperti pada gambar 76b.

**Tabel 25. Tabel kebenaran detektor jumlah ganjil 3 input**

| INPUT | | | OUPUT |
|---|---|---|---|
| A | B | C | Y |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |




**Gambar 76b. Peta Karnaugh detektor jumlah ganjil 3 input**

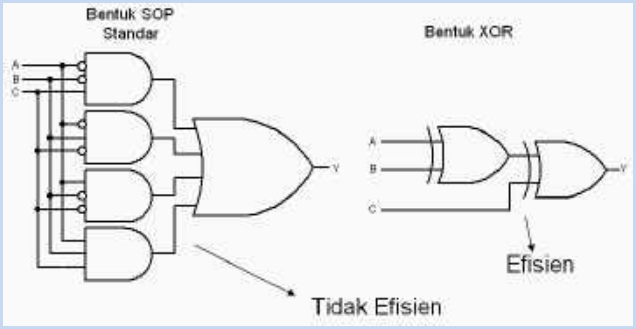
Dari peta Karnaugh terlihat bahwa semua *minterm* yang ada terisolasi, sehingga jika diimplementasikan dengan SOP bentuknya adalah SOP standar sebagai berikut:

$$Y = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\,\overline{C} + ABC$$

$$Y = \sum m(1,2,4,7) \qquad \text{persamaan (31)}$$

dan hal ini merupakan implementasi yang tidak efisien. Oleh karena tabel kebenarannya membentuk fungsi detektor jumlah ganjil, maka akan lebih efisien jika diimplementasikan dengan gerbang XOR sehingga bentuk persamaannya adalah:

$$Y = A \oplus B \oplus C \qquad \text{persamaan (32)}$$



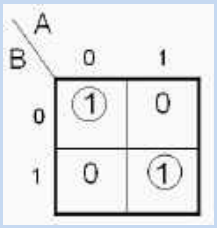

**Gambar 77. Implementasi detektor jumlah ganjil 3-input**

## 2. Equality Comparator

*Equality comparator* merupakan rangkaian logika yang memberikan keadaan output tinggi jika keadaan input-inputnya sama. Tabel kebenaran untuk rangkaian ini ditunjukkan pada tabel 26.

**Tabel 26. Tabel kebenaran *equality comparator***

| INPUT | | OUTPUT | |
|---|---|---|---|
| A | B | Y | |
| 0 | 0 | 1 | $\rightarrow m_0 = \overline{A}\,\overline{B}$ |
| 0 | 1 | 0 | $\rightarrow M_1 = A + \overline{B}$ |
| 1 | 0 | 0 | $\rightarrow M_2 = \overline{A} + B$ |
| 1 | 1 | 1 | $\rightarrow m_3 = A\,B$ |




**Gambar 78. Peta Karnaugh *equality comparator***

Dari peta Karnaugh terlihat bahwa semua *minterm* yang ada terisolasi, sehingga fungsi Y yang diperoleh berbentuk standar dan sudah merupakan bentuk yang minimum. Persamaan output dalam bentuk SOP minimum dari *equality comparator* adalah

$$Y = \overline{A}\,\overline{B} + A\,B$$
$$Y(A,B) = \sum m(0,3) \qquad \text{persamaan (32)}$$

dan dalam bentuk POS minimum adalah

$$Y = (A + \overline{B})(\overline{A} + B)$$
$$Y(A,B) = \prod M(1,2) \qquad \text{persamaan (33)}$$

Dengan melakukan operasi komplemen ganda dan memberlakukan teorema de Morgan terhadap persamaan (32) dan persamaan (33) dapat diperoleh bentuk NAND dan NOR dari *equality comparator*. Bentuk NAND *equality comparator* diperoleh dengan cara sebagai berikut

$$Y = \overline{A}\,\overline{B} + A\,B$$
$$Y = \overline{\overline{A}\,\overline{B} + A\,B}$$
$$Y = \overline{\overline{A}\,\overline{B}} \;.\; \overline{A\,B} \qquad \text{persamaan (34)}$$

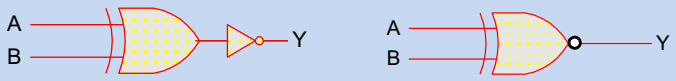
Sedangkan bentuk NOR diperoleh dengan cara

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

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

$$Y = \overline{\overline{(A + \overline{B})} + \overline{(\overline{A} + B)}} \qquad \text{persamaan (35)}$$

Kecuali dalam bentuk SOP, POS, NAND, dan NOR, *equality comparator* juga dapat diperoleh dengan melakukan operasi NOT terhadap rangkaian *non-equality comparator* atau operasi NOT terhadap gerbang XOR:




(a) (b)

**Gambar 79. (a) Operasi NOT terhadap XOR, (b) simbol gerbang XNOR**

Bukti bahwa operasi NOT terhadap fungsi *non-equality comparator* atau operasi NOT terhadap XOR menghasilkan fungsi *equality comparator* ditunjukkan melalui operasi-operasi berikut ini:

$Y = \overline{\overline{A}B + A\overline{B}}$ → Operasi NOT terhadap *non-equality comparator*

$Y = \overline{\overline{A}B} \,.\, \overline{A\overline{B}}$

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

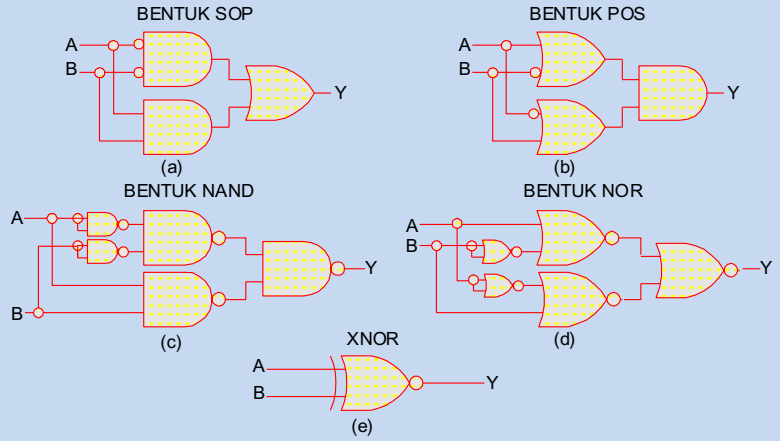
$Y = (A + \overline{B})(\overline{A} + B)$ → Hasilnya adalah *equality comparator*

Dalam praktek, fungsi yang dihasilkan oleh operasi NOT terhadap XOR disediakan dalam bentuk gerbang XNOR, dan simbolnya ditunjukkan pada gambar 79 (b), sedangkan persamaan logikanya adalah:

$$Y = A \odot B \qquad \text{persamaan (36)}$$

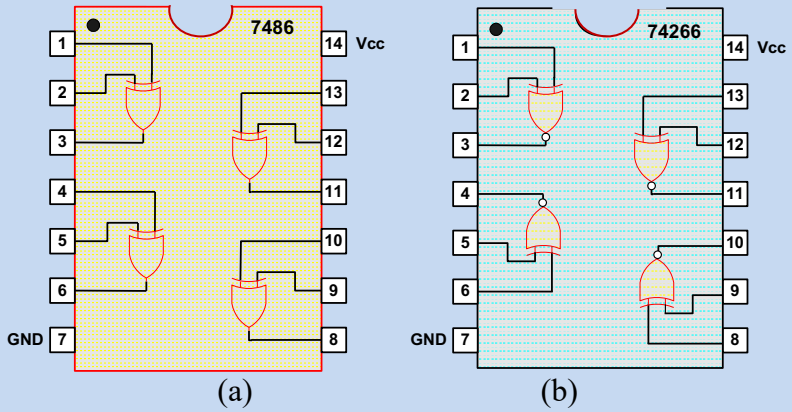
Jika dicermati tabel kebenarannya, kecuali membentuk fungsi sebagai komparator, output gerbang XNOR juga membentuk fungsi sebagai detektor jumlah genap. Perhatikan tabel 26, jika jumlah pasangan inputnya genap, maka outputnya 1. Susun tabel kebenaran detektor jumlah genap untuk 3-input! Dengan menggunakan peta Karnaugh, selidiki apakah bentuk standarnya masih dapat disederhanakan? Jika tidak dapat, bagaimanakah Anda memperoleh bentuk implementasi paling efisien dari detektor jumlah genap 3-input tersebut?

Berdasarkan persamaan (32), (33), (34), (35), dan (36), *equality comparator* dapat diimplementasikan dalam bentuk-bentuk seperti ditunjukkan pada gambar 80.




**Gambar 80. Berbagai bentuk implementasi *equality comparator***

Nomor seri rangkaian terpadu atau piranti IC yang menyediakan fungsi XOR 2-input adalah 7486 dan nomor seri IC untuk XNOR 2-input adalah 74266. Spesifikasi pin kedua seri IC tersebut ditunjukkan pada gambar 81.




**Gambar 81. Spesifikasi pin IC untuk (a) XOR (7486), dan (b) XNOR (74266)**

## B. Penjumlah Biner (Adder)

Penjumlah biner merupakan rangkaian logika kombinasi yang berfungsi melakukan operasi penjumlahan bilangan biner. Penjumlah biner 1-bit terdiri atas *half adder* dan *full adder*.

## 1. Half Adder

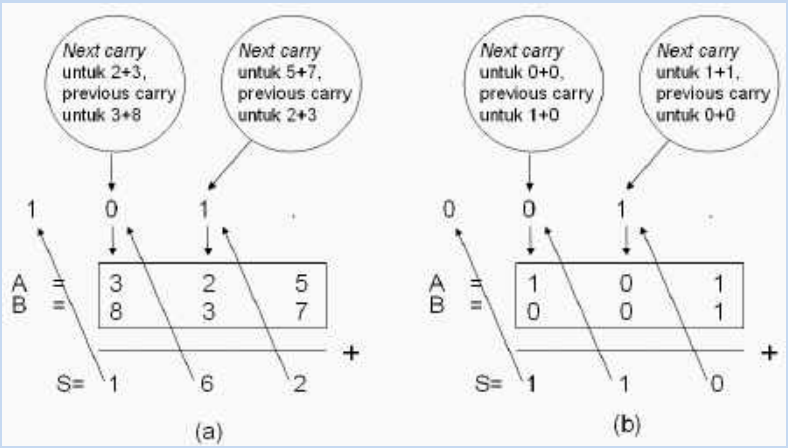
*Half adder* merupakan rangkaian penjumlah yang tidak menyertakan bawaan sebelumnya (*previous carry*) pada inputnya. Dari definisi tersebut dapat disusun tabel kebenaran sebagai berikut:

**Tabel 27. Tabel kebenaran *half adder***

| INPUT | | OUTPUT | |
|---|---|---|---|
| A | B | S | Cn |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

Keterangan:
A: *Augend* (bilangan yang dijumlahkan)
B: *Addend* (bilangan penjumlah)
S: *Sum* (hasil penjumlahan)
$C_n$: *Next carry* (bawaan berikutnya)

Untuk memahami pengertian *previous carry* dan *next carry* coba perhatikan penjumlahan bilangan desimal dan bilangan biner berikut ini!




**Gambar 82. Ilustrasi penjumlahan: (a) bilangan desimal, (b) bilangan biner**

Pada gambar 82 (a) ditunjukkan penjumlahan bilangan desimal 325 dengan 837. Penjumlahan dimulai dari LSD yakni 5 ditambah 7 hasilnya 2 dengan *next carry* 1. Selanjutnya *next carry* yang dihasilkan dari 5 ditambah 7 dijumlahkan dengan 2 dan 3, hasilnya 6 dengan *next carry* 0. Dalam hal ini *next carry* yang dihasilkan dari 5 ditambah 7 merupakan *previous carry* bagi 2 ditambah 3. Seterusnya, *next carry* 0 yang dihasilkan dari penjumlahan 2 dan 3 dijumlahkan dengan 3 dan 8 dan hasilnya adalah 1 dengan *next carry* 1. Hasil penjumlahan akhir adalah 162 dengan *next carry* 1 atau 1162.

Untuk penjumlahan bilangan biner 101 dan 001 pada gambar 134 (b) prosesnya juga dimulai dari bit dengan bobot terkecil (LSB) yakni 1 ditambah 1 hasilnya 0 dengan *next carry* 1. Dalam hal ini *next carry* yang dihasilkan dari 1 ditambah 1 merupakan *previous carry* bagi 0 ditambah 0, sehingga menghasilkan 1 dengan *next carry* 0. Selanjutnya, *next carry* 0 ditambahkan dengan penjumlahan 1 dan 0 dan hasilnya 1 dengan *next carry* 0. Hasil penjumlahan akhir adalah 110 dengan *next carry* 0.

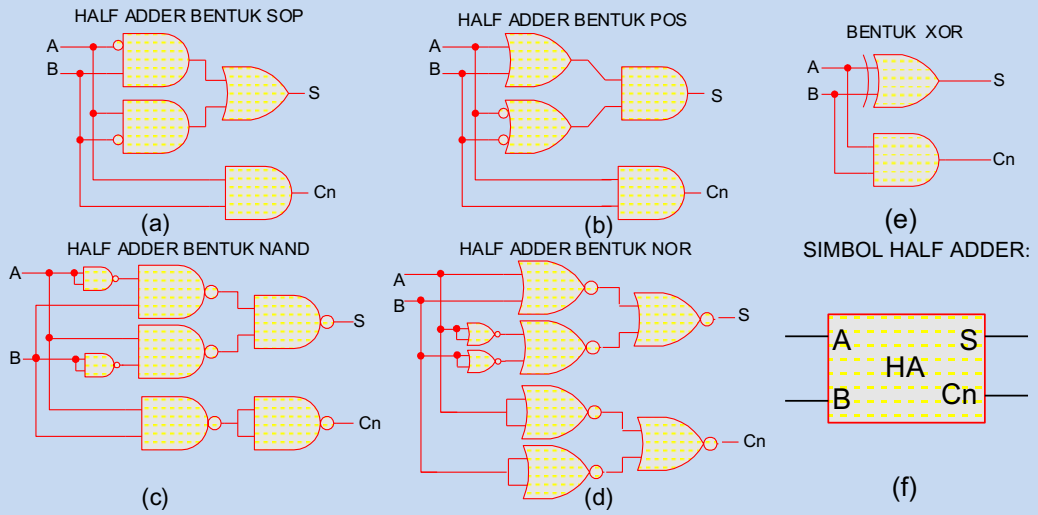
Perhatikan bahwa output S membentuk fungsi *non-equality comparator* atau XOR dan output $C_n$ membentuk operasi AND. Jadi, persamaan output *half adder* untuk hasil penjumlahan dapat ditulis:

$$S = A \oplus B \qquad \text{persamaan (37)}$$

dan untuk bawaan berikutnya:

$$C_n = AB \qquad \text{persamaan (38)}$$

Pada bagian muka telah dijelaskan bahwa fungsi *non-equality comparator* dapat diimplementasikan dalam lima bentuk yakni SOP, POS, NAND, NOR, dan XOR. Oleh karena output S merupakan fungsi *non-equality comparator*, maka *half adder* juga dapat diimplementasikan ke dalam lima bentuk seperti ditunjukkan pada gambar 83.



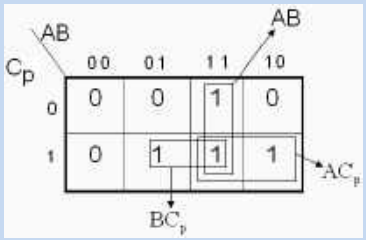

**Gambar 83. Implementasi dan simbol *half adder***

## 2. Full Adder

*Full adder* adalah rangkaian penjumlah yang menyertakan bawaan sebelumnya (*previous carry*) pada inputnya. Atas dasar pengertian tersebut, tabel kebenaran *full adder* 1-bit dapat disusun sebagai berikut.

**Tabel 28.Tabel kebenaran *full adder* 1-bit**

| INPUT | | | OUPUT | |
|---|---|---|---|---|
| A | B | Cp | S | Cn |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |




**Gambar 84. Peta Karnaugh untuk $C_n$**

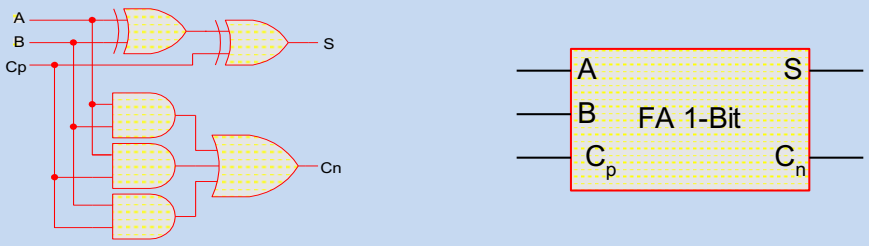
Dalam hal ini $C_p$ adalah *previous carry* (bawaan sebelumnya). Dari tabel kebenaran terlihat bahwa output S membentuk fungsi detektor jumlah ganjil, dan di muka telah ditunjukkan bahwa fungsi tersebut dapat diimplementasikan secara efisien dengan menggunakan XOR 3-input, jadi:

$$S = A \oplus B \oplus C_p \qquad \text{persamaan (39)}$$

Sedangkan untuk menentukan persamaan output $C_n$, disusun terlebih dahulu peta Karnaugh untuk $C_n$ seperti ditunjukkan pada gambar 84. Berdasarkan peta Karnaugh tersebut dapat diperoleh persamaan untuk $C_n$ sebagai berikut:

$$C_n = AB + AC_p + BC_p \qquad \text{persaamaan (40)}$$

Berdasarkan persamaan (39) dan (40), implementasi rangkaian *full adder* 1-bit dapat dinyatakan dalam bentuk seperti pada gambar 85 (a), dan simbolnya ditunjukkan pada gambar 85 (b).




(a) (b)

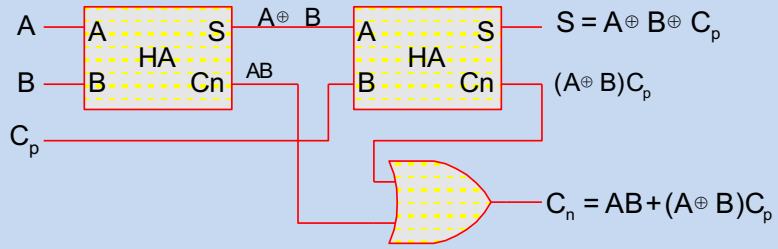
**Gambar 85. Full adder: (a) implementasi dengan XOR, (b) simbol**

Selain dapat diimplementasikan dengan XOR, *full adder* 1-bit juga dapat diimplementasikan dengan menggunakan *half adder*. Jika *full adder* diimplementasikan dengan *half adder*, maka persamaan output untuk S tetap seperti pada persamaan (39) yakni $S = A \oplus B \oplus C_p$, namun persamaan output untuk $C_n$, perlu dimodifikasi menjadi:

$$
\begin{aligned}
C_n &= AB + AC + BC_p \\
&= AB + AC_p(B + \overline{B}) + BC_p(A + \overline{A}) \\
&= AB + (ABC_p + A\overline{B}C_p) + (ABC_p + \overline{A}BC_p) \\
&= (AB + ABC_p + ABC_p) + (A\overline{B}C_p + \overline{A}BC_p) \\
&= AB(1 + C_p + C_p) + (A\overline{B} + \overline{A}B)C_p \\
&= AB(1) + (A\overline{B} + \overline{A}B)C_p \\
&= AB + (A\overline{B} + \overline{A}B)C_p \\
C_n &= AB + (A \oplus B)C_p
\end{aligned}
$$

persamaan (41)

Dengan menggunakan persamaan (39) dan (41), implementasi *full adder* dengan menggunakan *half adder* dapat dideskripsikan seperti pada gambar 86.

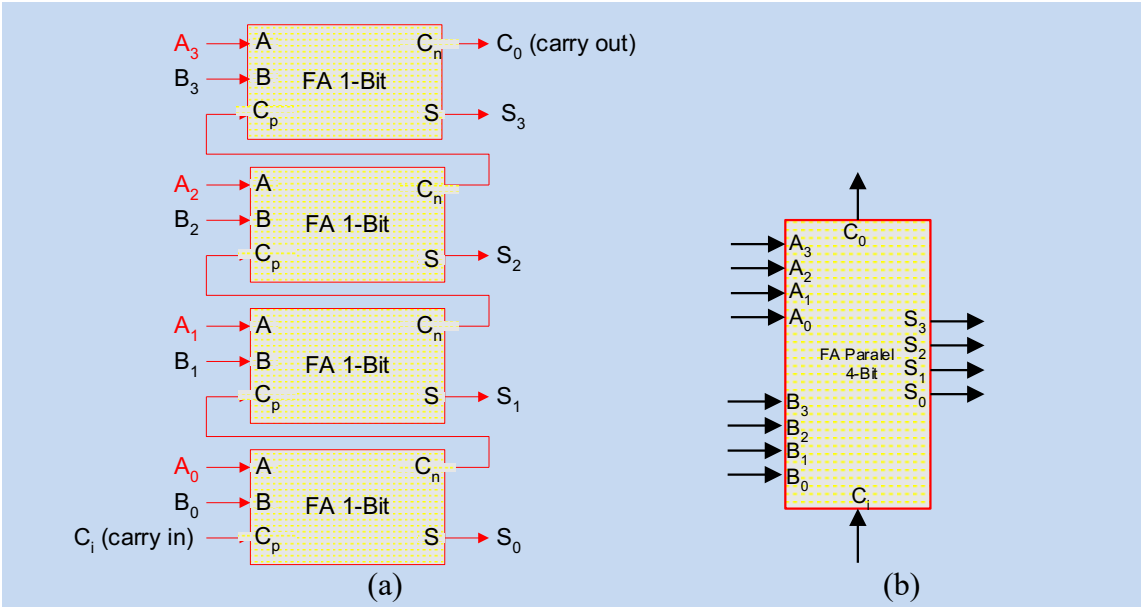



**Gambar 86. Implementasi *full adder* dengan *half adder***

### 3. *Full Adder* Paralel

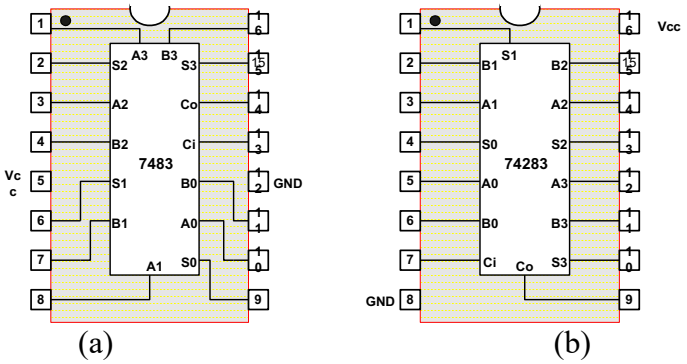
*Full adder* paralel merupakan rangkaian logika yang melakukan proses penjumlahan data biner n-bit. Contoh *full adder* paralel 4-bit memiliki input $A_3$, $A_2$, $A_1$, $A_0$ untuk input A, dan $B_3$, $B_2$, $B_1$, $B_0$ untuk input B serta $S_3$, $S_2$, $S_1$, dan $S_0$ untuk output S. Gambar 87 (a) menunjukkan *full adde*r paralel 4-bit, dan gambar 87 (b) menunjukkan simbolnya. *Full adder* paralel 4-bit dibangun dengan menempatkan 4 buah *full adder* 1-bit secara berjajar misalnya dari bawah ke atas. Selanjutnya input dan output *full adder*

terbawah ditetapkan sebagai input dan output dengan bobot terkecil atau LSB yakni $A_0$, $B_0$, dan $S_0$. Input *previous carry* ($C_p$) pada *full adder* terbawah ditetapkan sebagai input *carry* ($C_i$) dari *full adder* paralel. *Next carry* ($C_n$) dari setiap *full adder* 1-bit dihubungkan dengan $C_p$ dari *full adder* 1-bit berikutnya dan seterusnya, sedangkan $C_n$ dari *full adder* 1-bit teratas ditetapkan sebagai output *carry* ($C_o$) dari *full adder* paralel.




**Gambar 87. *Full adder* paralel 4-bit: (a) rangkaian, dan (b) simbol**

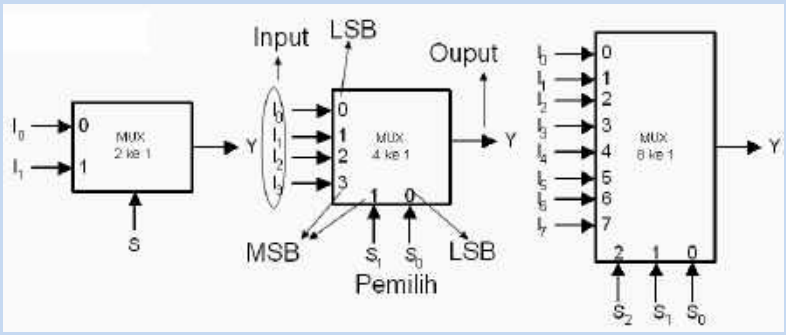
IC TTL yang menyediakan fungsi *full adder* paralel adalah seri 7483 dan seri 74283. Gambar 88 (a) menunjukkan spesifikasi pin dari IC 7483 dan gambar 88 (b) untuk seri 74283 yang merupakan *full adde*r paralel 4-bit.




**Gambar 88. Spesifikasi pin IC *full adder* paralel 4-bit (a) seri 7483, dan (b) seri 74283**

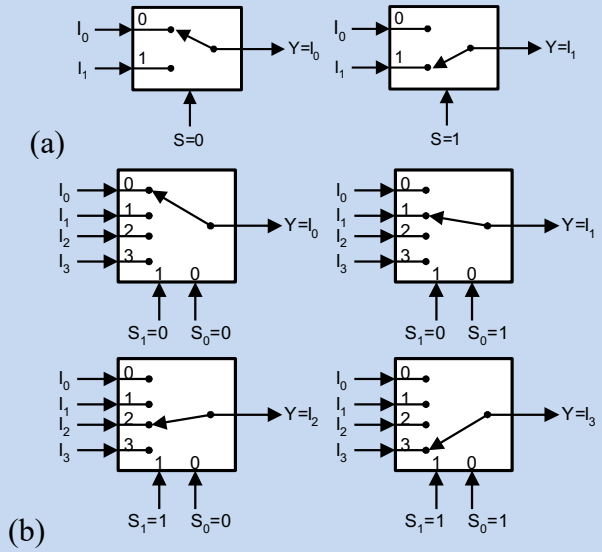
## C. Multiplexer

*Multiplexer* merupakan rangkaian logika yang berfungsi memilih data yang ada pada inputnya untuk disalurkan ke outputnya dengan bantuan sinyal pemilih atau sinyal kontrol. Kata *multiplexer* sering dikemukakan dalam bentuk singkatannya yakni MUX. *Multiplexer* disebut juga sebagai pemilih data (*data selector*). Jumlah input *multiplexer* adalah $2^n$ (n=1,2,3,…) dengan n merupakan jumlah bit sinyal pemilih, sehingga terdapat MUX 2 ke 1 dengan 1-bit sinyal pemilih, MUX 4 ke 1 dengan 2-bit sinyal pemilih, MUX 8 ke 1 dengan 3-bit sinyal pemilih, dan seterusnya. Simbol *multiplexer* untuk berbagai jumlah input ditunjukkan pada gambar 89.




**Gambar 89. Simbol *multiplexer*: (a) MUX 2 ke 1, (b) MUX 4 ke 1, dan (c) MUX 8 ke 1**

Bagaimanakah cara kerja *multiplexer*? Untuk memahaminya, coba perhatikan rangkaian analogi *multiplexer* pada gambar 90!




**Gambar 90. Rangkaian analogi: (a) MUX 2 ke 1, (b) MUX 4 ke 1**

Pada gambar 90 (a) ditunjukkan rangkaian analogi atau perumpamaan dari MUX 2 ke 1, dan pada gambar 90 (b) untuk MUX 4 ke 1. *Multiplexer* dapat diumpamakan seperti saklar putar, dalam hal ini pemindahan saklar dilakukan dengan memberikan sinyal pemilih. Untuk MUX 2 ke 1 pada gambar 90 (a), pemberian sinyal pemilih 0 (S=0) menyebabkan data pada input 0 dipilih untuk disalurkan ke Y. Demikian pula pemberian sinyal pemilih 1 (S=1) menyebabkan data pada input 1 dipilih untuk disalurkan ke Y. Untuk MUX 4 ke 1 pada gambar 148 (b), pemberian sinyal pemilih $S_1S_0$=00 yang bernilai 0 menyebabkan input yang bersesuaian dengan nilai sinyal pemilihnya yakni $I_0$ akan dipilih untuk disalurkan ke outputnya sehingga Y=$I_0$. Sedangkan pemberian sinyal pemilih $S_1S_0$=01 menyebabkan input 1 dipilih, $S_1S_0$=10 menyebabkan input 2 dipilih, dan $S_1S_0$=11 menyebabkan input 3 dipilih untuk disalurkan ke output Y. Rangkaian di atas merupakan rangkaian analogi *multiplexer* dengan menggunakan saklar putar (*rotary switch*), dan rangkaian yang sesungguhnya terdiri atas gerbang-gerbang logika dasar yang membentuk fungsi penyaklaran.

Berdasarkan definisi *multiplexer* dan rangkaian analoginya, dapat disusun tabel kebenaran dari MUX 4 ke 1 sebagai berikut:

**Tabel 29. Tabel kebenaran MUX 4 ke 1**

| PEMILIH | | OUTPUT | |
|---|---|---|---|
| $S_1$ | $S_0$ | Y | |
| 0 | 0 | $I_0$ | $\rightarrow \bar{S}_1\bar{S}_0I_0$ |
| 0 | 1 | $I_1$ | $\rightarrow \bar{S}_1S_0I_1$ |
| 1 | 0 | $I_2$ | $\rightarrow S_1\bar{S}_0I_2$ |
| 1 | 1 | $I_3$ | $\rightarrow S_1S_0I_3$ |

*Multiplexer* pada dasarnya adalah rangkaian logika berbentuk AND-OR atau SOP. Berdasarkan tabel kebenarannya, maka dapat diperoleh *product* atau suku persamaan SOP $\bar{S}_1\bar{S}_0I_0$, $\bar{S}_1S_0I_1$, $S_1\bar{S}_0I_2$, dan $S_1S_0I_3$ sehingga persamaan output MUX 4 ke 1 adalah:

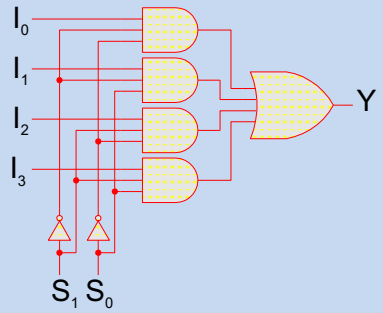
$$Y = \bar{S}_1\bar{S}_0I_0 + \bar{S}_1S_0I_1 + S_1\bar{S}_0I_2 + S_1S_0I_3 \qquad \text{persamaan (42)}$$

Berdasarkan persamaan output MUX 4 ke 1 pada persamaan (42) dapat dijelaskan cara kerja *multiplexer* sebagai berikut, jika sinyal pemilih $S_1S_0$=00 maka:

$$Y = \bar{S}_1\bar{S}_0I_0 + \bar{S}_1S_0I_1 + S_1\bar{S}_0I_2 + S_1S_0I_3$$
$$Y = \bar{0}.\bar{0}.I_0 + \bar{0}.0.I_1 + 0.\bar{0}.I_2 + 0.0.I_3$$
$$Y = 1.1\ .I_0 + 1.0.I_1\ + 0.1.\ I_2 + 0.0.I_3$$
$$Y = I_0$$

Dengan menggunakan persamaan (42) dapat ditunjukkan bahwa jika pemilih MUX 4 ke 1 diberi sinyal $S_1S_0$=00, maka Y=$I_0$. Hal itu berarti pemberian sinyal pemilih $S_1S_0$=00 menyebabkan input 0 ($I_0$) pada MUX 4 ke 1 disalurkan ke output Y.

Berdasarkan persamaan outputnya pada persamaan (42), maka rangkaian MUX 4 ke 1 dapat disusun sebagai berikut:

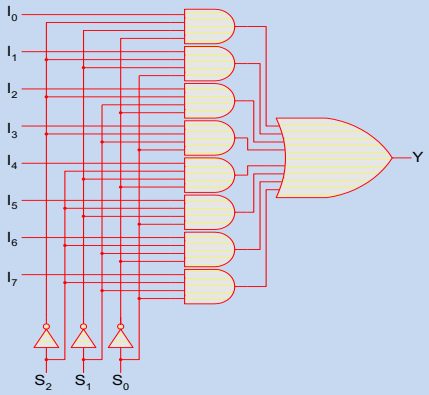



**Gambar 91. Rangkaian MUX 4 ke 1**

Dengan menggunakan cara penurunan yang sama dengan MUX 4 ke 1, persamaan output untuk MUX 8 ke 1 dapat ditulis:

$$Y = \bar{S}_2\bar{S}_1\bar{S}_0I_0 + \bar{S}_2\bar{S}_1S_0I_1 + \bar{S}_2S_1\bar{S}_0I_2 + \bar{S}_2S_1S_0I_3$$
$$+ S_2\bar{S}_1\bar{S}_0I_4 + S_2\bar{S}_1S_0I_5 + S_2S_1\bar{S}_0I_6 + S_2S_1S_0I_7 \qquad \text{persamaan (43)}$$

Berdasarkan persamaan (43), rangkaian MUX 8 ke 1 dapat digambarkan sebagai berikut:



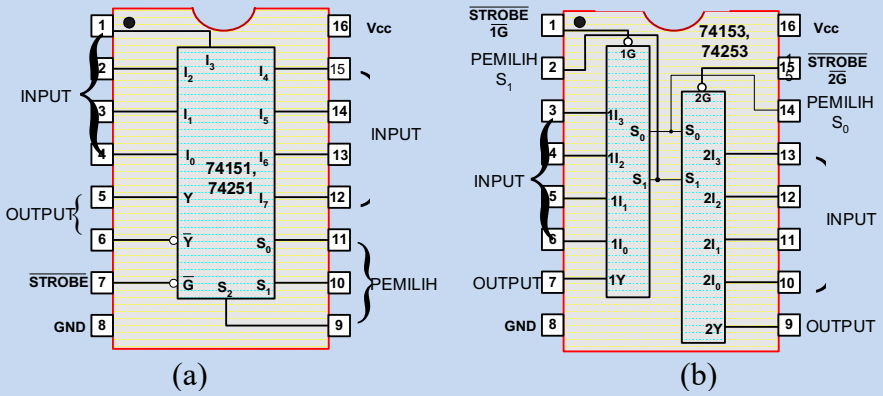

**Gambar 92. Rangkaian MUX 8 ke 1**

IC TTL yang menyediakan fungsi *multiplexer* terdiri atas berbagai seri seperti ditunjukkan pada tabel 30.

**Tabel 30. IC yang menyediakan fungsi *multiplexer***

| Seri | Spesifikasi |
|---|---|
| 74151 | MUX 8 ke 1 |
| 74153 | MUX 4 ke 1, 2 buah |
| 74157 | MUX 2 ke 1, 4 buah |
| 74158 | MUX 2 ke 1, 4 buah, outputnya membalik |
| 74251 | MUX 8 ke 1, outputnya 3-status |
| 74253 | MUX 4 ke 1, 2 buah, outputnya 3-status |
| 74257 | MUX 2 ke 1, 4 buah, outputnya 3-status |
| 74258 | MUX 2 ke 1, 4 buah, outputnya membalik, outputnya 3-status |

Spesifikasi pin IC *multiplexer* untuk seri 74151, 74251, 74153, dan 74253 ditunjukkan pada gambar 93.



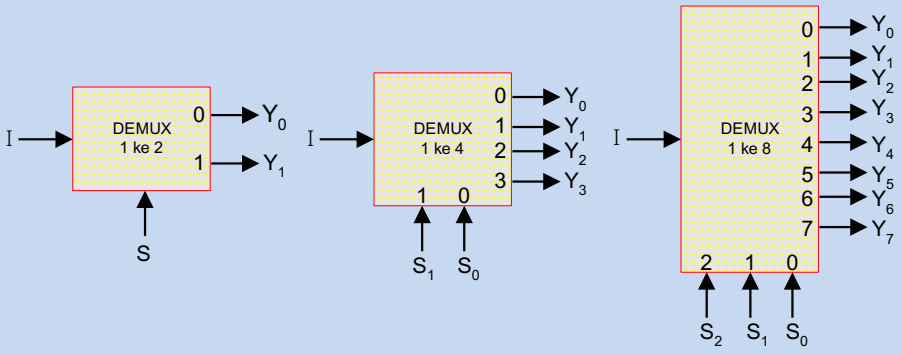

**Gambar 93. Spesifikasi pin IC *multiplexer* seri (a) 74151 dan 74251, (b) 74153 dan 74253**

Perhatikan bahwa pada kedua IC *multiplexer* tersebut terdapat pin STROBE yang berfungsi mengaktifkan piranti tersebut. Oleh karena jenisnya *active-low*, maka untuk mengaktifkan *multiplexer* pin STROBE diberi sinyal rendah atau 0.

## D. Demultiplexer

*Demultiplexer* merupakan rangkaian logika yang berfungsi menyalurkan data yang ada pada inputnya ke salah satu dari beberapa outputnya dengan bantuan sinyal pemilih atau sinyal kontrol. Dalam penyebutannya, *demultiplexer* sering dikemukakan dalam bentuk singkatannya saja yakni DEMUX. *Demultiplexer* disebut juga sebagai penyalur data (*data distributor*), dan fungsinya merupakan kebalikan dari fungsi *multiplexer*.

Jumlah output *demultiplexer* adalah $2^n$ (n=1,2,3,…), dalam hal ini n merupakan jumlah bit sinyal pemilih, sehingga terdapat DEMUX 1 ke 2 dengan 1-bit sinyal pemilih, DEMUX 1 ke 4 dengan 2-bit sinyal pemilih, DEMUX 1 ke 8 dengan 3-bit sinyal pemilih, dan seterusnya. Simbol *demultiplexer* ditunjukkan pada gambar 94.

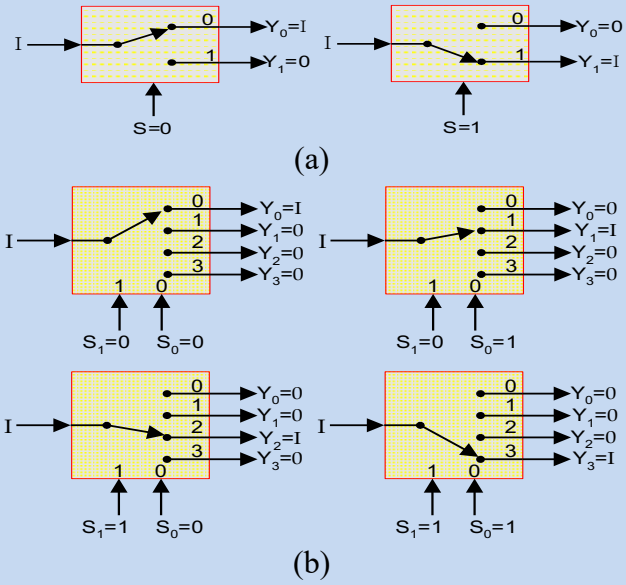



**Gambar 94. Simbol *demultiplexer*: (a) DEMUX 1ke 2, (b) DEMUX 1 ke 4, dan (c) DEMUX 1ke 8**

Bagaimanakah cara kerja *demultiplexer*? Agar diperoleh kemudahan dalam memahaminya coba perhatikan rangkaian analogi *demultiplexer* menggunakan saklar putar seperti ditunjukkan pada gambar 95.

Perhatikan rangkaian analogi DEMUX 1 ke 2 pada gambar 95 (a)! Pemberian sinyal pemilih 0 (S=0) menyebabkan data pada input I disalurkan atau didistribusikan ke output 0 sehingga $Y_0$=I dan $Y_1$=0. Sedangkan pemberian sinyal pemilih 1 (S=1) menyebabkan data pada input I disalurkan ke output 1 sehingga $Y_0$=0 dan $Y_1$=I. Selanjutnya perhatikan rangkaian analogi DEMUX 1 ke 4 pada gambar 95 (b)! Pemberian sinyal pemilih $S_1S_0$=00 menyebabkan data I pada input *demultiplexer* disalurkan ke output 0 sehingga $Y_0$=I dan output lainnya bernilai 0. Demikian pula pemberian sinyal pemilih $S_1S_0$=01 menyebabkan input I disalurkan ke output 1 sehingga $Y_1$=I, $S_1S_0$=10

menyebabkan input I disalurkan ke output 2 sehingga $Y_2$=I, serta $S_1S_0$=11 menyebabkan input I disalurkan ke output 3 sehingga $Y_3$=I. Perlu ditegaskan di sini bahwa ketika sebuah output *demultiplexer* sedang menyalurkan data inputnya, maka output-output yang lain akan bernilai rendah atau 0.




**Gambar 95. Rangkaian analogi *demultiplexer*: (a) DEMUX 1 ke 2, dan (b) DEMUX 1 ke 4**

Dari definisi *demultiplexer* dan rangkaian analoginya, dapat disusun tabel kebenaran dari DEMUX 1 ke 4 sebagai berikut:

**Tabel 31. Tabel kebenaran DEMUX 1 ke 4**

| PEMILIH | | OUTPUT | | | |
|---|---|---|---|---|---|
| $S_1$ | $S_0$ | $\overline{Y}_0$ | $\overline{Y}_1$ | $\overline{Y}_2$ | $\overline{Y}_3$ |
| 0 | 0 | I | 0 | 0 | 0 |
| 0 | 1 | 0 | I | 0 | 0 |
| 1 | 0 | 0 | 0 | I | 0 |
| 1 | 1 | 0 | 0 | 0 | I |

*Demultiplexer* pada dasarnya adalah kumpulan gerbang AND. Berdasarkan tabel 31, dapat diperoleh operasi AND untuk setiap output DEMUX 1 ke 4 yakni $\overline{S}_1\overline{S}_0I$, $\overline{S}_1S_0I$, $S_1\overline{S}_0I$, dan $S_1S_0I$, sehingga untuk DEMUX 1 ke 4 persamaan outputnya adalah:

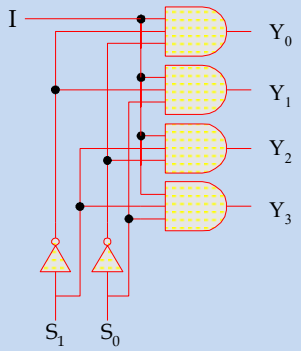
$$Y_0 = \bar{S}_1 \bar{S}_0 \, I$$
$$Y_1 = \bar{S}_1 S_0 \, I$$
$$Y_2 = S_1 \bar{S}_0 \, I$$
$$Y_3 = S_1 S_0 \, I$$

persamaan (44)

Untuk sinyal pemilih $S_1S_0$=00, maka output DEMUX 1 ke 4 adalah:

$$Y_0 = \bar{0} . \bar{0} . I = 1 . 1 . I = I$$
$$Y_1 = \bar{0} . 0 . I = 1 . 0 . I = 0$$
$$Y_2 = 0 . \bar{0} . I = 0 . 1 . I = 0$$
$$Y_3 = 0 . 0 . I = 0$$

Terlihat bahwa untuk sinyal pemilih $S_1S_0$=00, input I akan disalurkan ke output 0 ($Y_0$=0), dan output lainnya bernilai 0. Berdasarkan persamaan outputnya, rangkaian DEMUX 1 ke 4 dapat disusun seperti ditunjukkan pada gambar 96.




**Gambar 96. Rangkaian DEMUX 1 ke 4**

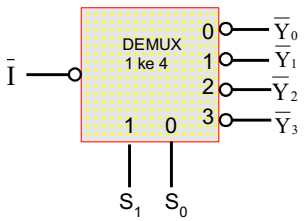
Jika diperhatikan rangkaian DEMUX 1 ke 4 pada gambar 96, terlihat bahwa hanya input yang bernilai 1 saja yang memungkinkan semua gerbang AND pada *demultiplexer* tersebut bersifat *enable*. Hal itu berarti bahwa *demultiplexer* akan aktif jika inputnya bernilai tinggi atau 1. Sebaliknya, input dengan nilai rendah atau 0 akan menyebabkan semua gerbang AND pada *demultiplexer* bersifat *inhibit* sehingga semua outputnya keadaannya rendah atau 0. Oleh karena *demultiplexer* akan aktif jika keadaan inputnya tinggi maka *demultiplexer* tersebut dikatakan memiliki input jenis *active-high*.

Perhatikan kembali gambar 96! Terlihat bahwa output *demultiplexer* yang dipilih untuk menyalurkan data input akan bernilai tinggi atau 1, sedangkan output yang tidak dipilih bernilai rendah atau 0. Oleh karena output yang aktif memberikan keadaan logika tinggi atau 1 maka dikatakan *demultiplexer* tersebut memiliki output jenis *active-high*. Dalam kebanyakan aplikasi, rangkaian *demultiplexer* dirancang dengan input dan output jenis *active-low*. Suatu rangkaian logika dikatakan memiliki input jenis *active-low* jika rangkaian tersebut dapat aktif untuk input yang bernilai rendah. Sedangkan output rangkaian logika dikatakan *active-low* jika rangkaian tersebut memberikan output rendah ketika aktif.

Tabel kebenaran untuk rangkaian DEMUX 1 ke 4 dengan input dan output jenis *active-low* dapat disusun seperti pada tabel 32.

**Tabel 32. Tabel kebenaran DEMUX 1 ke 4: input dan ouput jenis *active-low***

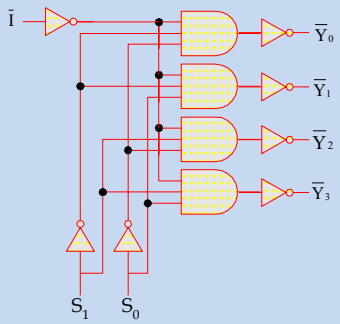
| INPUT | | | OUTPUT | | | |
|---|---|---|---|---|---|---|
| ENABLE | PEMILIH | | | | | |
| $\bar{I}$ | $S_1$ | $S_0$ | $\overline{Y}_0$ | $\overline{Y}_1$ | $\overline{Y}_2$ | $\overline{Y}_3$ |
| 1 | X | X | 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 |




**Gambar 97. Simbol DEMUX 1 ke 4: input dan ouput jenis *active-low***

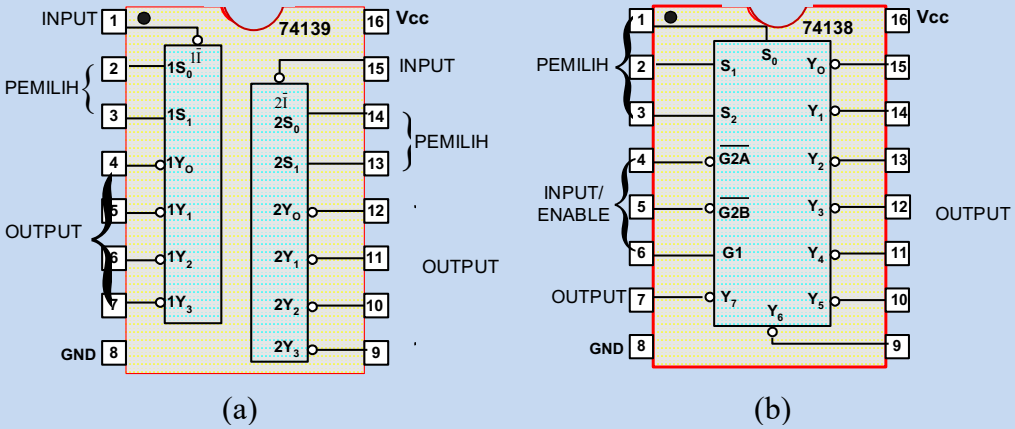
Perhatikan bahwa pada rangkaian logika dengan input dan output jenis *active-low*, notasinya diberi tanda komplemen seperti $\bar{I}$ untuk input, dan $\overline{Y}_3$, $\overline{Y}_2$, $\overline{Y}_1$, $\overline{Y}_0$ untuk output. Pada tabel 32 ditunjukkan pula bahwa untuk input bernilai tinggi atau 1, pemberian sinyal pemilih untuk semua kemungkinan, akan memberikan keadaan tidak aktif. Simbol DEMUX 1 ke 4 untuk jenis ini ditunjukkan pada gambar 97.

Berdasarkan tabel 32, dapat disusun rangkaian DEMUX 1 ke 4 untuk input dan output jenis *active-low* seperti pada gambar 98.




**Gambar 98. Rangkaian DEMUX 1 ke 4: input dan output jenis *active-low***

Dalam bentuk IC, fungsi *demultiplexer* disediakan antara lain oleh IC dengan nomor seri 74138 untuk DEMUX 1 ke 8, dan seri 74139 yang di dalamnya terkandung dua buah DEMUX 1 ke 4. IC lain yang menyediakan fungsi *demultiplexer* adalah seri 74155 dan seri 74156 yang menyediakan dua buah DEMUX 1 ke 4, serta seri 74159 yang menyediakan fungsi DEMUX 1 ke 16. Gambar 99 menunjukkan spesifikasi pin IC dengan seri 74138 dan 74139.

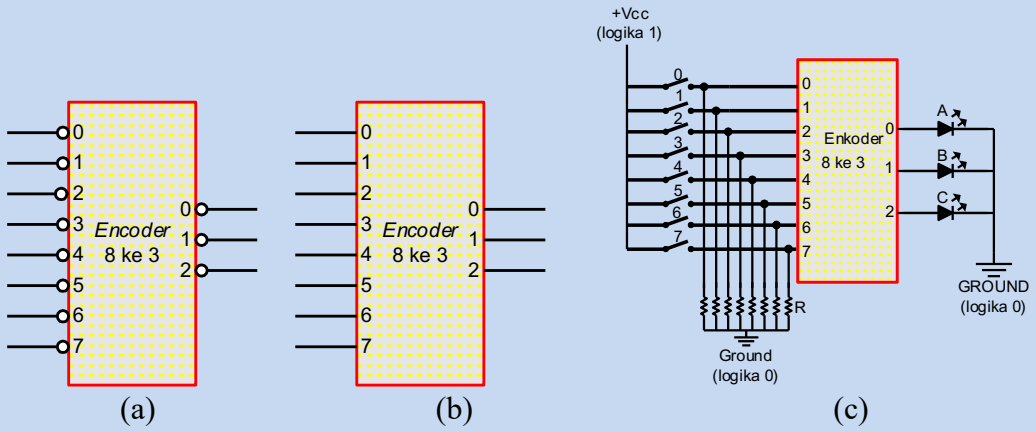



**Gambar 99. Spesifikasi pin IC *demultiplexer*: (a) seri 74139, dan (b) seri 74138**

## E. Encoder

*Encoder* merupakan rangkaian logika yang berfungsi mengubah data yang ada pada inputnya menjadi kode-kode biner pada outputnya. Contoh *encoder* oktal ke biner atau disebut juga *encoder* 8 ke 3, berfungsi mengubah data bilangan oktal pada inputnya

menjadi kode biner 3-bit pada outputnya. Simbol *encoder* oktal ke biner ditunjukkan pada gambar 100 (a) dan 100 (b).




**Gambar 100. *Encoder* oktal ke biner:**
**(a) simbol untuk input dan output jenis *active-low*, (b)jenis *active-high*, dan (c) rangkaian eksperimen**

Dari rangkaian percobaan pada gambar 100 (c) dapat diperoleh tabel kebenaran seperti disajikan pada tabel 33.

**Tabel 33. Tabel kebenaran *encoder* 8 ke 3**

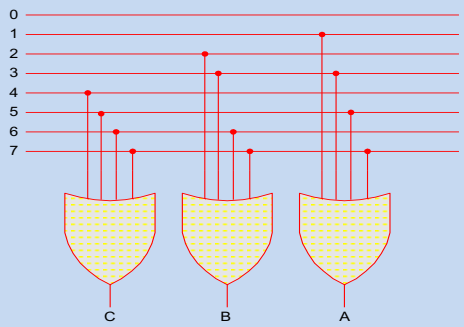
| INPUT | | | | | | | | OUTPUT | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | C (MSB) | B | A (LSB) |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

Pada tabel 33, nilai logika 1 di bagian input menunjukkan saklar dalam keadaan tertutup (ON), sedangkan nilai logika 0 menunjukkan saklar dalam keadaan terbuka (OFF). Pada bagian output, nilai logika 1 menunjukkan LED dalam keadaan menyala, dan nilai logika

0 menunjukkan LED padam. Berdasarkan tabel kebenarannya, dapat disusun persamaan output rangkaian *encoder* 8 ke 3 seperti pada persamaan berikut ini:

$$A = 1 + 3 + 5 + 7$$
$$B = 2 + 3 + 6 + 7 \qquad \text{persamaan (45)}$$
$$C = 4 + 5 + 6 + 7$$

Perhatikan kembali tabel 33! Pada bagian output A terdapat 4 nilai logika tinggi atau 1 yang bersesuaian dengan penekanan tombol 1, tombol 3, tombol 5, dan tombol 7 sehingga output A merupakan operasi OR dari input 1, input 3, input 5, dan input 7 atau A=1+3+5+7. Pada output B terdapat 4 nilai logika tinggi yang bersesuaian dengan penekanan tombol 2, tombol, 3, tombol 6, dan tombol 7, sehingga B=2+3+6+7. Sedangkan pada output C, nilai logika tingginya bersesuaian dengan penekanan tombol 4, tombol 5, tombol 6, dan tombol 7, sehingga C=4+5+6+7. Atas dasar persamaan logika outputnya rangkaian *encoder* 8 ke 3 dapat disusun seperti ditunjukkan pada gambar 101.




**Gambar 101. Rangkaian *encoder* 8 ke 3**

*Encoder* pada gambar 101 di atas merupakan *encoder* yang jenisnya bukan prioritas, artinya untuk menghasilkan kode biner pada outputnya, hanya boleh ada 1 saklar saja pada inputnya yang tertutup (ON), perhatikan kembali tabel kebenarannya! Pada umumnya, rangkaian *encoder* dalam kemasan IC yang dijual di pasaran merupakan *encoder* prioritas, artinya un tuk membangkitkan kode biner pada outputnya, saklar pada input tertinggi saja yang diperhatikan atau diprioritaskan. Keadaan saklar-saklar selain saklar pada input tertinggi diabaikan. Tabel 34 (a) menunjukkan contoh tabel kebenaran *encoder* prioritas 8

ke 3 untuk input dan output jenis *active-high*, dan tabel 34 (b) untuk input dan output jenis *active-low*.

**Tabel 34. Tabel kebenaran *encoder* prioritas 8 ke 3 dengan input dan output jenis: (a) *active-high,* dan (b*) active-low*.**

(a)

| INPUT | | | | | | | | OUTPUT | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | C (MSB) | B | A (LSB) |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| X | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| X | X | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| X | X | X | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| X | X | X | X | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| X | X | X | X | X | 1 | 0 | 0 | 1 | 0 | 1 |
| X | X | X | X | X | X | 1 | 0 | 1 | 1 | 0 |
| X | X | X | X | X | X | X | 1 | 1 | 1 | 1 |

(b)

| INPUT | | | | | | | | OUTPUT | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $\bar{0}$ | $\bar{1}$ | $\bar{2}$ | $\bar{3}$ | $\bar{4}$ | $\bar{5}$ | $\bar{6}$ | $\bar{7}$ | $\bar{C}$ | $\bar{B}$ | $\bar{A}$ |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| X | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| X | X | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| X | X | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| X | X | X | X | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| X | X | X | X | X | 0 | 1 | 1 | 0 | 1 | 0 |
| X | X | X | X | X | X | 0 | 1 | 0 | 0 | 1 |
| X | X | X | X | X | X | X | 0 | 0 | 0 | 0 |

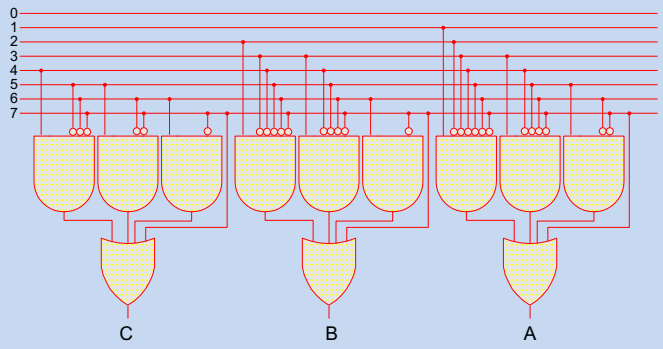
Dalam hal ini X adalah keadaan diabaikan, artinya X dapat bernilai 0 atau 1, dan keduanya tidak diperhatikan. Perhatikan tabel 34 (a) baris ke-3 dari atas! Pada baris ini saklar tertinggi yang ditekan adalah saklar 2. Dengan demikian saklar 2 merupakan saklar yang keadaannya diprioritaskan atau diperhatikan. Saklar-saklar di bawah saklar 2 yakni saklar 1 dan saklar 0 diberi tanda X untuk menunjukkan bahwa kedua saklar itu keadaannya dibaikan. Dari tabel 34 (a), dapat diperoleh persamaan output *encoder* prioritas untuk input dan output jenis *active-high* sebagai berikut:

$$A = 1\bar{2}\bar{3}\bar{4}\bar{5}\bar{6}\bar{7} + 3\bar{4}\bar{5}\bar{6}\bar{7} + 5\bar{6}\bar{7} + 7$$
$$B = 2\bar{3}\bar{4}\bar{5}\bar{6}\bar{7} + 3\bar{4}\bar{5}\bar{6}\bar{7} + 6\bar{7} + 7 \qquad \text{persamaan (46)}$$
$$C = 4\bar{5}\bar{6}\bar{7} + 5\bar{6}\bar{7} + 6\bar{7} + 7$$

Persamaan A diperoleh dari baris-baris pada kolom A yang memberikan output 1. Suku I berhubungan dengan baris ke-2, suku II berhubungan dengan baris ke-4, suku III berhubungan dengan baris ke-6, dan suku IV berhubungan dengan baris ke-8 dari tabel 34 (a). Perhatikan tabel 34 (a) baris ke-2! Saklar tertinggi yang ditekan atau dalam keadaan

ON adalah saklar 1, sedangkan keadaan saklar 0 dapat diabaikan, dan saklar 2 sampai dengan saklar 7 keadaannya OFF. Suku yang dihasilkan dari keadaan tersebut merupakan operasi AND dari saklar ON tertinggi dan komplemen-komplemen dari saklar-saklar OFF sehingga suku I adalah $1\bar{2}\bar{3}\bar{4}\bar{5}\bar{6}\bar{7}$. Perhatikan baris ke-4! Pada baris ini A bernilai 1 untuk keadaan saklar ON tertinggi saklar 3, dan dalam hal ini keadaan saklar 0 sampai dengan saklar 2 diabaikan, serta saklar 4 sampai dengan saklar 7 OFF. Dengan demikian suku yang dihasilkan adalah $3\bar{4}\bar{5}\bar{6}\bar{7}$. Pada baris ke-6 terlihat bahwa A bernilai 1 untuk keadaan saklar ON tertinggi adalah saklar 5, dan saklar 6 sampai dengan saklar 7 keadaannya OFF sehingga suku yang dihasilkan adalah $5\bar{6}\bar{7}$. Pada baris ke-8, A bernilai 1 untuk keadaan saklar ON tertinggi adalah saklar 7 dan selain saklar tersebut keadaannya diabaikan sehingga suku ke-4 adalah 7. Oleh karena suku-suku A diperoleh dari output yang bernilai 1 maka bentuk persamaannya adalah SOP sehingga $A = 1\bar{2}\bar{3}\bar{4}\bar{5}\bar{6}\bar{7} + 3\bar{4}\bar{5}\bar{6}\bar{7} + 5\bar{6}\bar{7} + 7$. Perhatikan tabel 34 (a) kolom B baris ke3, ke-4, ke-7, dan ke-8! Dengan menggunakan penurunan yang sama dengan output A, maka untuk output B diperoleh suku-suku persamaan $2\bar{3}\bar{4}\bar{5}\bar{6}\bar{7}$, $3\bar{4}\bar{5}\bar{6}\bar{7}$, $6\bar{7}$, dan $7$, sehingga output B adalah $B = 2\bar{3}\bar{4}\bar{5}\bar{6}\bar{7} + 3\bar{4}\bar{5}\bar{6}\bar{7} + 6\bar{7} + 7$. Perhatikan pula tabel 34 (a) kolom C baris ke-5, ke-6, ke-7, dan ke-8! Dari sana dapat diperoleh $C = 4\bar{5}\bar{6}\bar{7} + 5\bar{6}\bar{7} + 6\bar{7} + 7$.

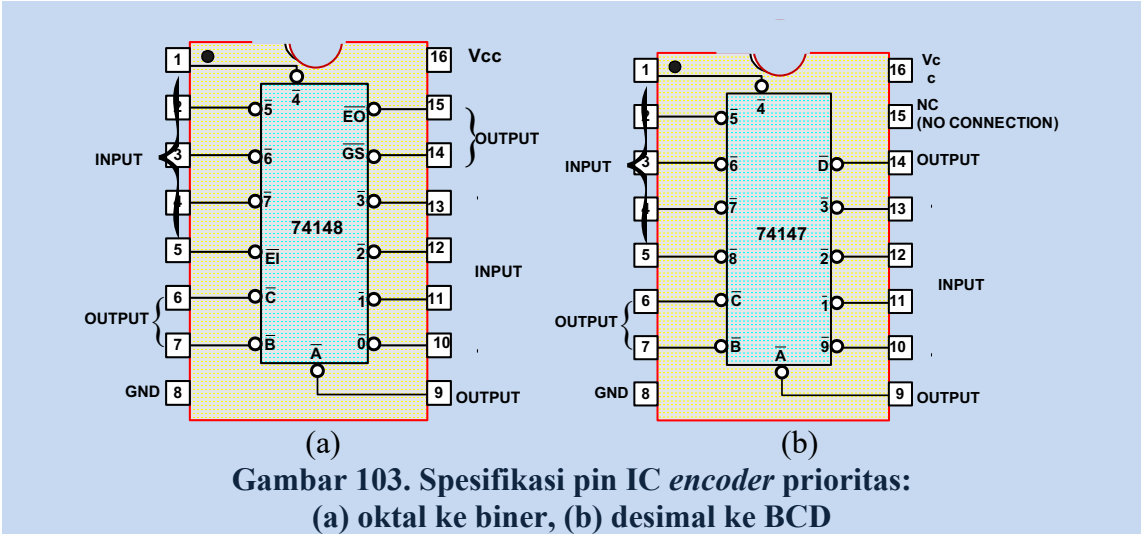
Berdasarkan persamaan (46), rangkaian *endcoder* prioritas oktal ke biner untuk input dan output jenis *active-high* dapat disusun seperti pada gambar 102.




**Gambar 102. Rangkaian *encoder* prioritas oktal ke biner input dan output jenis *active-high***

Selain *encoder* oktal ke biner terdapat pula *encoder* desimal ke BCD atau *encoder* 10 ke 4. Coba susun tabel kebenaran untuk *encoder* desimal ke biner jenis *active-high* dan *active-low*!

Dalam bentuk kemasan IC jenis TTL, *encode*r prioritas oktal ke biner disediakan oleh seri 74148, sedangkan *encode*r prioritas desimal ke BCD seri 74147. Gambar 103 menunjukkan spesifikasi pin untuk kedua IC tersebut. Terlihat bahwa kedua IC tersebut menyediakan fungsi *encoder* jenis *active-low* baik untuk input maupun outputnya.




**Gambar 103. Spesifikasi pin IC *encoder* prioritas: (a) oktal ke biner, (b) desimal ke BCD**

Untuk IC seri 74148 pada inputnya terdapat pin $\overline{EI}$ (*enable input*) jenis *active-low*. Pin tersebut berfungsi sebagai pengendali untuk mengaktifkan input. Jika $\overline{EI}$ bernilai tinggi atau 1, maka input *encoder* bersifat *inhibit*, artinya semua inputnya diblokir sehingga *encoder* tidak aktif. Untuk mengaktifkan *encoder*, $\overline{EI}$ diberi nilai rendah atau 0 sehingga semua inputnya bersifat *enable*. Pabrik IC tersebut juga telah melengkapi output 74148 dengan pin $\overline{OE}$ dan $\overline{GS}$. Pin-pin output tambahan disediakan untuk memungkinkan perluasan penggunaan IC 74148 tanpa memerlukan rangkaian eksternal, karena dalam beberapa aplikasi biasanya keduanya diperlukan.

## F. Decoder

*Decoder* merupakan rangkaian logika yang berfungsi mengkode ulang atau menafsirkan kode-kode biner yang ada pada inputnya menjadi data asli pada outputnya, dan fungsinya merupakan kebalikan dari fungsi *encode*r. Contoh: *decoder* 2 ke 4 berfungsi menafsirkan kode-kode biner 2-bit menjadi data asli bilangan desimal 0 sampai dengan 3.

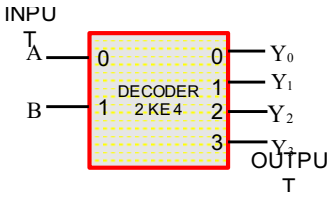
*Decoder* biner ke oktal atau *decoder* 3 ke 8 berfungsi menafsirkan kode-kode biner 3-bit menjadi data asli sistem oktal. *Decoder* BCD ke desimal atau *decoder* 4 ke 10 berfungsi menafsirkan kode-kode BCD menjadi bilangan desimal. *Decoder* BCD ke peraga 7 segmen berfungsi mengubah kode-kode BCD menjadi kode-kode penggerak peraga 7 segmen.

### 1. *Decoder* 2 ke 4

Dari definisi, dapat disusun tabel kebenaran suatu *decoder* misalnya *decoder* 2 ke 4 dengan input dan output jenis *active-high* seperti ditunjukkan pada tabel 35.

**Tabel 35.Tabel kebenaran *decoder* 2 ke 4**

| INPUT | | OUTPUT | | | |
|---|---|---|---|---|---|
| B | A | $Y_0$ | $Y_1$ | $Y_2$ | $Y_3$ |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 |

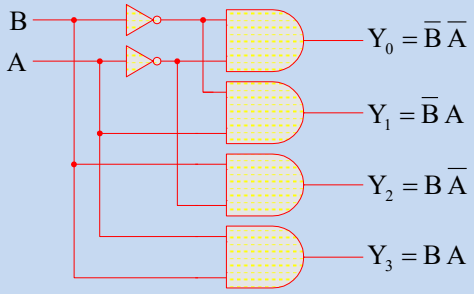



**Gambar 104. Simbol *decoder* 2 ke 4, input dan ouput jenis *active-high***

Pada baris ke-1 terlihat bahwa output $Y_0$ bernilai tinggi dan output lainnya rendah jika inputnya berupa kode 00. Hal itu berarti *decoder* menafsirkan kode 00 biner sebagai 0 desimal. Demikian pula jika inputnya berupa kode 01 pada baris ke-2 maka output $Y_1$ bernilai 1 dan output lainnya 0 yang berarti *decoder* menafsirkan kode 01 biner sebagai 1 desimal. Selanjutnya, jika inputnya berupa kode biner 10, *decoder* akan menafsirkannya menjadi 2 desimal yang ditandai dengan output $Y_2$ bernilai 1 dan output lainnya bernilai 0. Untuk input berupa kode biner 11 output $Y_3$ bernilai 1 dan output lainnya bernilai 0 yang menunjukkan bahwa *decoder* menafsirkan kode biner 11 sebagai 3 desimal. Terlihat bahwa pada *decoder* dengan output jenis *active-high*, hanya terdapat 1 buah output yang bernilai tinggi untuk suatu input tertentu sedangkan output lainnya bernilai 0. Dengan demikian pada *decoder* yang memiliki output jenis *active-low*, hanya terdapat 1 buah output yang bernilai 0 dan output lainnya bernilai 1 untuk suatu keadaan input tertentu.

Berdasarkan tabel kebenaran pada tabel 35, terlihat bahwa output $Y_0$=1 berhubungan dengan input B=0 dan A=0 atau BA=00. Agar input 00 tersebut dapat menghasilkan nilai 1 pada $Y_0$, maka kedua input itu harus dikomplemenkan sehingga $Y_0 = \overline{B}\,\overline{A}$. Berikutnya terlihat pula bahwa output $Y_1$=1 berhubungan dengan input 01.

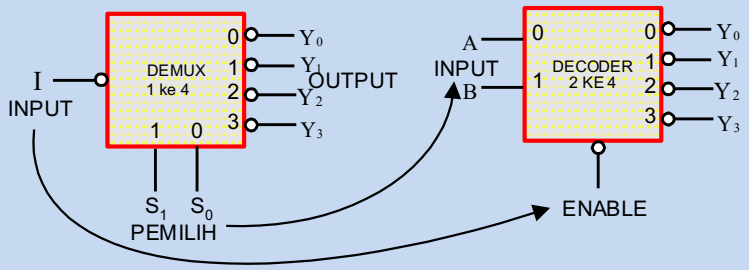
Supaya input tersebut dapat menghasilkan nilai 1 pada $Y_1$, maka nilai B harus dikomplemenkan sehingga $Y_1 = \overline{B}\,A$. Selanjutnya, karena $Y_2$=1 berhubungan dengan input 10, maka $Y_2 = B\,\overline{A}$, dan karena $Y_3$=1 berhubungan dengan input 11, maka $Y_3 = B\,A$. Jadi, persamaan output *decoder* 2 ke 4 dan rangkaiannya adalah:




persamaan (47)

**Gambar 105. Rangkaian *decoder* 2 ke 4**

Perhatikan bahwa tabel kebenaran rangkaian *decoder* 2 ke 4 pada tabel 35 bentuknya mirip dengan tabel kebenaran rangkaian *demultiplexer* 1 ke 4 pada tabel 31. Hal tersebut menunjukkan bahwa rangkaian *decoder* pada dasarnya adalah rangkaian *demultiplexer*. Contoh: *demultiplexer* 1 ke 4 adalah *decoder* 2 ke 4, *demultiplexer* 1 ke 8 adalah *decoder* 3 ke 8, dan *demultiplexer* 1 ke 16 adalah *decoder* 4 ke 16. Dalam hal ini, jika *demultiplexer* akan difungsikan sebagai *decoder*, maka input *demultiplexer* difungsikan sebagai input *enable* dari *decoder*, dan pemilih *demultiplexer* menjadi input *decoder*. Perhatikan gambar yang menunjukkan hubungan antara *demultiplexer* dan *decoder* berikut ini!

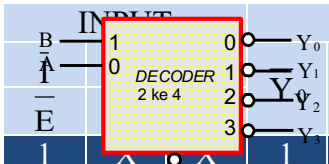



**Gambar 106. Simbol DEMUX 1 ke 4 dan *decoder* 2 ke 4**

Dalam kemasan IC TTL, *decoder* 2 ke 4 disediakan oleh seri 74139 yang juga menyediakan fungsi DEMUX 1 ke 4. Pada umumnya input-input *decoder* dalam kemasan IC berjenis *active-high*, sedangkan outputnya bervariasi di antara *active-high* dan *active-low*. Tabel kebenaran untuk IC 74139 yang menyediakan fungsi decoder 2 ke 4 ditunjukkan pada tabel 36.

**Tabel 36. Tabel kebenaran *decoder* 2 ke 4 dengan *enable* dan ouput jenis *active-low***

| INPUT | | | OUTPUT | | | |
|---|---|---|---|---|---|---|
| $\overline{E}$ | B | A | $\overline{Y}_0$ | $\overline{Y}_1$ | $\overline{Y}_2$ | $\overline{Y}_3$ |
| 1 | X | X | 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 |




**Gambar 107. Simbol *decoder* 2 ke 4: *enable* dan ouput jenis *active-low***

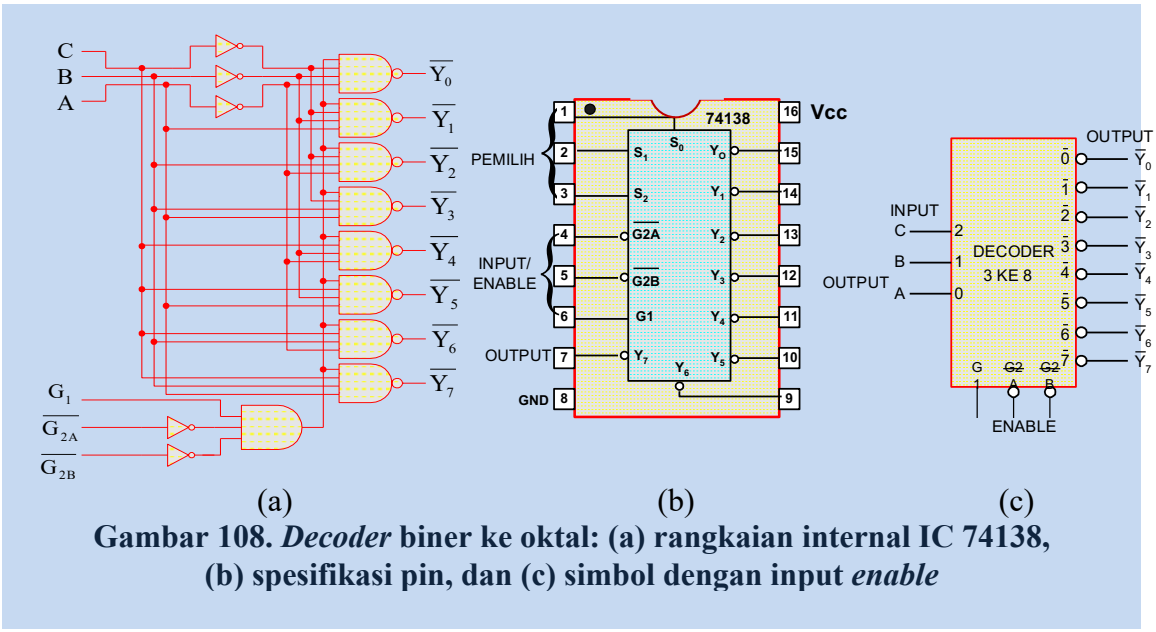
## 2. *Decoder* Biner ke Oktal

Rangkaian *decoder* biner ke oktal atau *decoder* 3 ke 8 berfungsi mengubah kode-kode biner 3-bit pada inputnya menjadi bilangan oktal pada outputnya. Decoder ini dalam kemasan IC disediakan oleh IC 74138 dengan spesifikasi pin seperti ditunjukkan pada gambar 99 (b) di muka atau gambar 108 (b). Berdasarkan lembar data (*data sheet*) yang diterbitkan oleh pabriknya, tabel kebenaran 74138 sebagai *decoder* 3 ke 8 ditunjukkan pada tabel 37. Terlihat bahwa output *decoder* 74138 menggunakan tanda NOT, yang berarti jenis outputnya adalah ACTIVE-LOW.

**Tabel 37. Tabel kebenaran *decoder* 3 ke 8 IC 74138**

| INPUT | | | | | OUTPUT | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $G_1$ | G2 | $S_2$ C | $S_1$ B | $S_0$ A | $\overline{Y}_0$ | $\overline{Y}_1$ | $\overline{Y}_2$ | $\overline{Y}_3$ | $\overline{Y}_4$ | $\overline{Y}_5$ | $\overline{Y}_6$ | $\overline{Y}_7$ |
| X | 1 | X | X | X | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | X | X | X | X | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

Keterangan: $G2 = \overline{G_{2A}} + \overline{G_{2B}}$



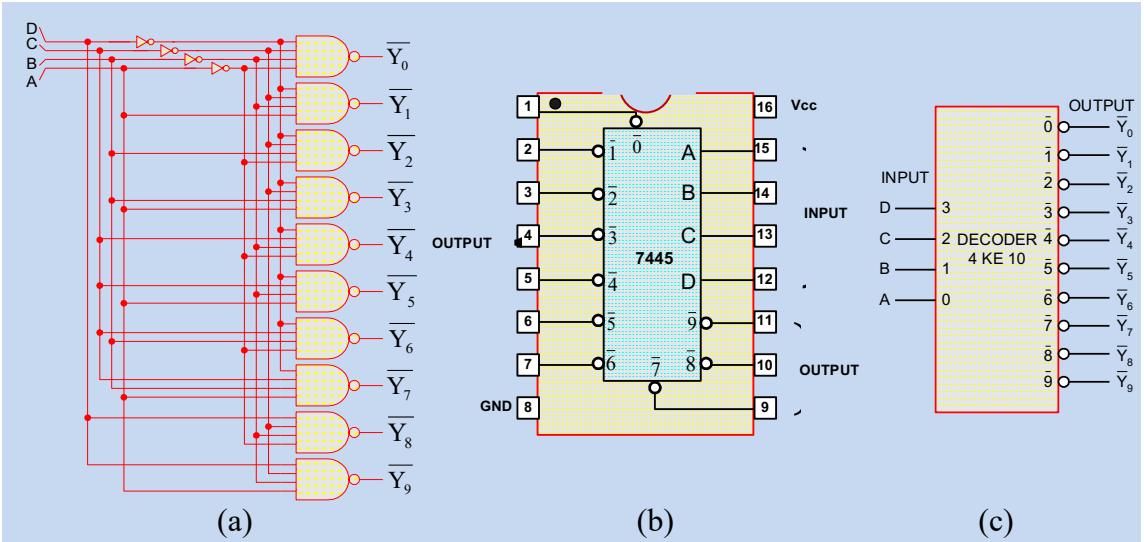

**Gambar 108. *Decoder* biner ke oktal: (a) rangkaian internal IC 74138, (b) spesifikasi pin, dan (c) simbol dengan input *enable***

### 3. *Decoder* BCD ke Desimal

*Decoder* yang berfungsi menafsirkan kode-kode BCD ke nilai desimal atau dinamakan pula *decoder* 4 ke 10 disediakan oleh IC TTL dengan seri 7442, 7445, 74145, 74445, dan 74141. Tabel kebenaran *decoder* BCD ke desimal dari berbagai IC tersebut ditunjukkan pada tabel 38, dan spesifikasi pin, serta rangkaiannya pada gambar 109.

**Tabel 38. Tabel kebenaran *decoder* BCD ke desimal dengan output *active-low***

| NO | INPUT | | | | OUTPUT | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | D | C | B | A | $\overline{Y}_0$ | $\overline{Y}_1$ | $\overline{Y}_2$ | $\overline{Y}_3$ | $\overline{Y}_4$ | $\overline{Y}_5$ | $\overline{Y}_6$ | $\overline{Y}_7$ | $\overline{Y}_8$ | $\overline{Y}_9$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 3 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 5 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 6 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 7 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 8 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 9 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| KODE TIDAK SAH | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

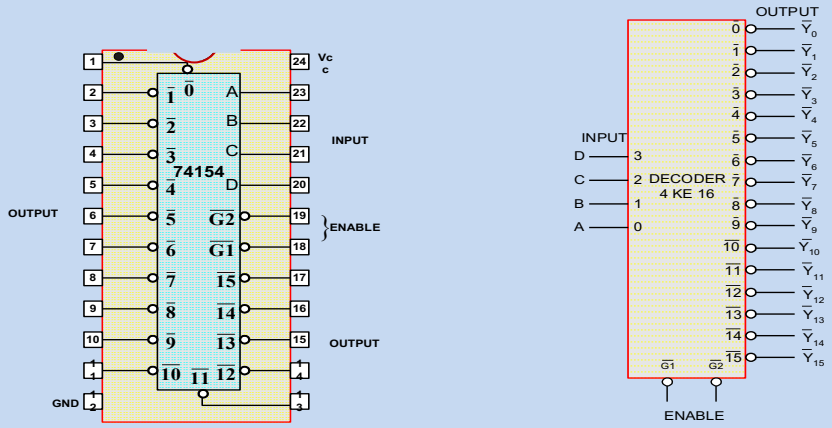



**Gambar 109. *Decoder* BCD ke desimal: (a) rangkaian internal IC 7442, 7445, 74145, dan 74445, (b) spesifikasi pin, (c) simbol tanpa input *enable***

## 4. *Decoder* 4 ke 16

*Decoder* 4 ke 16 menyediakan 16 saluran output sebagai saluran-saluran yang menampilkan hasil tafsiran terhadap kode 4-bit yang dimasukkan melalui inputnya. Dalam kemasan IC, *decoder* ini disediakan oleh IC dengan nomor seri 74154 dan 74159. Kedua

IC tersebut menyediakan fungsi *decoder* 4 ke 16 dengan output jenis *active-low*. Rangkaian internal IC 74154 atau 74159 ditunjukkan pada gambar 110 (a), dan spesifikasi pin untuk kedua IC itu pada gambar 110 (b). Terlihat bahwa *decoder* tersebut memiliki 2 buah input *enable* $\overline{G1}$ dan $\overline{G2}$ jenis *active-low*. Hal itu menunjukkan bahwa rangkaian *decoder* pada IC 74154 atau 7 4159 akan aktif hanya jika kedua input *enable* tersebut bernilai rendah atau $\overline{G1}$=0 dan $\overline{G2}$=0.

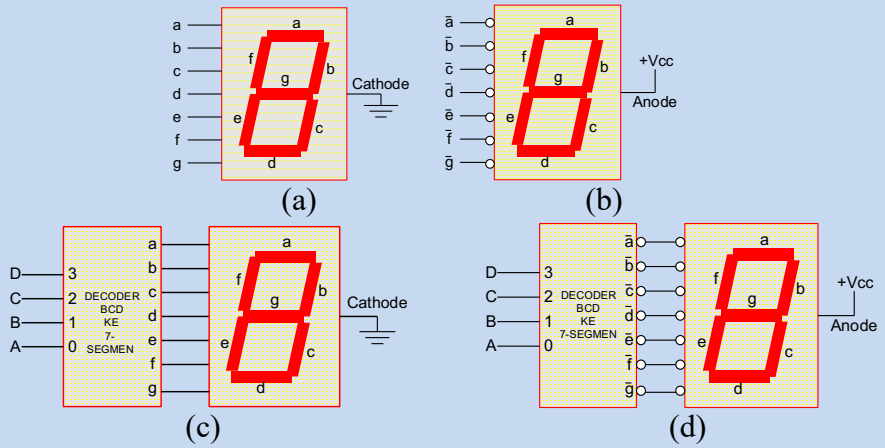



(a) (b)

**Gambar 110. *Decoder* 4 ke 16: (a) spesifikasi pin IC 74154 atau 74159, (b) simbol dengan input *enable***

## 5. *Decoder* BCD ke Peraga 7-Segmen

Agar data dalam bentuk kode BCD dapat langsung ditampilkan pada peraga 7-segmen, maka diperlukan rangkaian *decoder* yang menghasilkan sinyal-sinyal penggerak peraga 7-segmen. Pada Bab II di muka telah dijelaskan bahwa peraga 7-segmen terdiri atas dua jenis yakni *common anode* dan *common cathode*. Peraga 7-segmen jenis *common anode* memerlukan sinyal rendah dan jenis *common cathode* memerlukan sinyal tinggi untuk menyalakan segmen-segmennya. Secara umum, peraga 7-segmen memiliki input 7 buah yakni a, b, c, d, e, f, dan g yang digunakan untuk menyalakan segmen-segmennya.



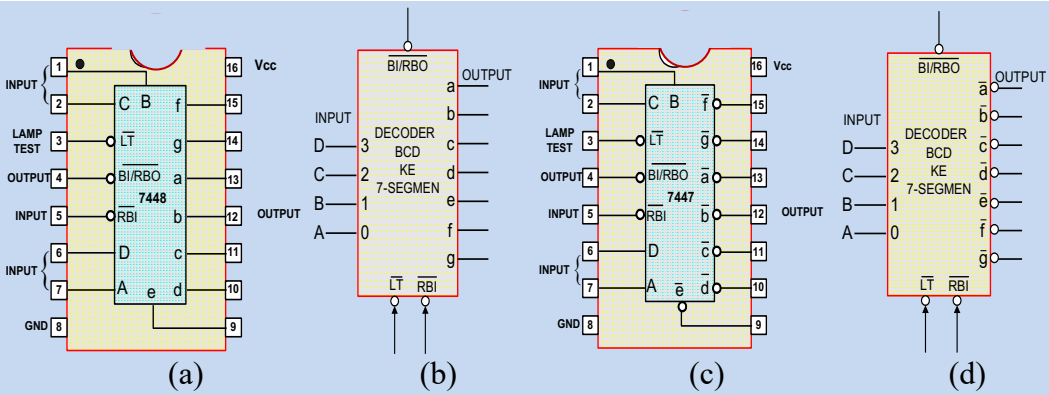

**Gambar 111. Peraga 7-segmen: (a) jenis *common cathode*, (b) jenis *common anode*, (c), dan (d) hubungannya dengan *decoder***

Dari gambar 111 (c) terlihat bahwa peraga 7-segmen jenis *common cathode* memerlukan *decoder* dengan output jenis *active-high* untuk menyalakan segmen-segmennya. Sedangkan jenis *common anode* memerlukan *decoder* dengan output jenis *active-low* seperti ditunjukkan pada gambar 111 (d). Untuk *decoder* BCD ke peraga 7-segmen dengan output jenis *active-high* (gambar 111 c) tabel kebenarannya dapat disusun seperti disajikan pada tabel 39.

**Tabel 39. Tabel kebenaran *decoder* BCD ke 7-segmen dengan output *active-high***

| INPUT | | | | OUTPUT | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| D | C | B | A | a | b | c | d | e | f | g | TAMPILAN |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 2 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 3 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 4 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 5 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 7 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 8 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 9 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | c |
| 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | ɔ |
| 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | u |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | ⊂ |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | t |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

Dalam kemasan rangkaian terpadu, *decoder* BCD ke peraga 7-segmen untuk output jenis *active-high* disediakan oleh IC dengan nomor seri 7448 atau 74248 dan untuk output jenis *active-low* disediakan oleh seri 7446, 74246, 7447, 74247, 74347, dan 74447. Spesifikasi pin dan simbol untuk IC tersebut ditunjukkan pada gambar 112.




**Gambar 112. *Decoder* BCD ke peraga 7-segmen:**
**(a) spsesifikasi pin 7448 dan 74248, (b) simbol untuk output *active-high*,**
**(c) spesifikasi pin 7446, 74246, 7447, 74247, 74347, dan 74447,**
**(d) simbol untuk output *active-low***

Selain menyediakan output untuk menggerakkan peraga 7-segmen jenis *common cathode*, IC ini menyediakan pula output $\overline{\text{RBO}}$. Untuk mengaktifkan *decoder*, IC ini dilengkapi dengan input pengendali $\overline{\text{LT}}$, $\overline{\text{BI}}$ dan $\overline{\text{RBI}}$. Kendali $\overline{\text{LT}}$ digunakan untuk *test lamp*, dan jika $\overline{\text{LT}}$ bernilai rendah maka *decoder* tidak aktif karena akan menyalakan semua segmen yang ada. Agar *decoder* dapat bekerja menafsirkan kode-kode BCD menjadi sinyal-sinyal penggerak peraga 7-segmen pada outputnya, maka $\overline{\text{LT}}$ dan $\overline{\text{RBI}}$ harus tinggi. Pengontrol $\overline{\text{RBI}}$ digunakan untuk mengatur penampilan angka 0, jika $\overline{\text{RBI}}$ bernilai rendah maka angka 0 desimal tidak ditampilkan.

## G. Soal Latihan

Soal berikut ini adalah jenis pilihan ganda. Kerjakan dengan cara memilih satu jawaban yang paling tepat dari opsi yang tersedia.

1. Soal nomor Fungsi *non-equality comparator* sama dengan fungsi gerbang:
   a. XNOR
   b. NOR
   c. XOR
   d. OR
   e. NAND

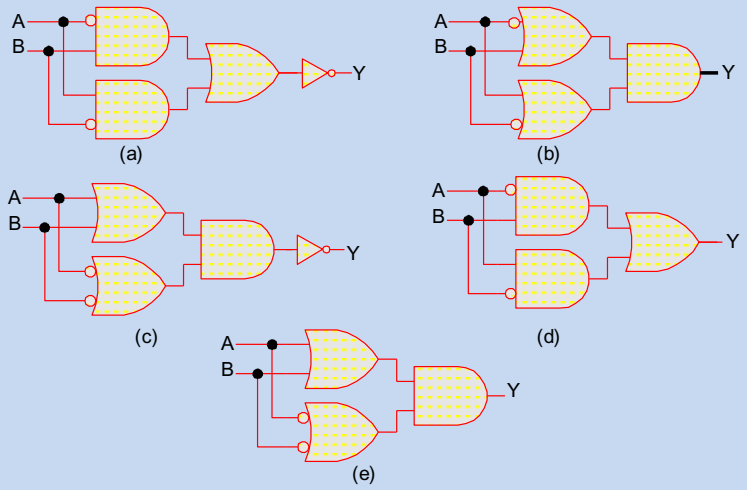
2. Definisi yang tepat dari *full adder* adalah:
   a. Penjumlah dengan dua buah input dan satu buah output
   *b.* Penjumlah dengan output mengandung *carry*
   c. Penjumlah dengan output mengandung *sum* dan *carry*
   d. Penjumlah yang melibatkan *carry* (bawaan) sebelumnya pada inputnya
   e. Penjumlah dengan dua input
3. Dalam kehidupan sehari-hari, multiplekser banyak diterapkan pada peralatan:
   a. SPST (*single-pole single-throw*) atau saklar tunggal
   b. DPDT (*double-pole double-throw*) atau saklar dengan kutub dan terminal ganda
   c. SPDT (*single-pole double-throw*) atau saklar dengan kutub tunggal terminal ganda
   d. Gabungan SPST dan DPDT
   e. Gabungan SPDT dan DPDT
4. Demultiplekser adalah rangkaian logika yang berfungsi:
   a. Mendistribusikan data yang ada pada inputnya ke output tertentu yang dipilih
   b. Memilih data yang ada pada inputnya dengan bantuan sinyal pemilih
   c. Mencampur data yang ada pada inputnya
   d. Menahan data yang ada pada inputnya dengan bantuan sinyal pemilih
   e. Meneruskan data yang ada pada inputnya
5. Jenis enkoder dengan input dan output bertanda not adalah jenis
   a. Input dan output ACTIVE-LOW
   b. Input dan output ACTIVE-HIGH
   c. Input ACTIVE-HIGH, output ACTIVE-LOW
   d. Input ACTIVE-LOW, output ACTIVE-HIGH
   e. Enkoder prioritas jenis ACTIVE-LOW
6. Rangkaian dekoder dengan 4 input bertanda NOT dan 10 output adalah jenis:
   a. Dekoder BCD ke desimal jenis input dan output ACTIVE-HIGH
   b. Dekoder BCD ke desimal jenis input dan output ACTIVE-LOW
   c. Dekoder BCD ke desimal jenis input ACTIVE-LOW, output ACTIVE-HIGH
   d. Dekoder BCD ke desimal jenis input ACTIVE-HIGH, output ACTIVE-LOW
   e. Bukan rangkaian dekoder

Soal-soal berikut ini adalah jenis uraian (esai).

7. Tunjukkan persamaan berikut ini yang merupakan fungsi *non-equality comparator* dan berikan alasannya!

   a. $Y = \overline{\overline{A}B + A\overline{B}}$
   b. $Y = (A + B)(\overline{A} + \overline{B})$
   c. $Y = \overline{A}BC + \overline{A}B\overline{C} + A\overline{B}$
   d. $Y = \overline{A}B\overline{C} + AB\overline{C} + A\overline{B}C$
   e. $Y = A(A + B)(\overline{A} + \overline{B})(A + B)\overline{A}$

8. Perhatikan rangkaian-rangkaian pada gambar 113 berikut ini! Tunjukkan rangkaian mana yang memberikan fungsi *equality comparator* dan berikan alasannya!




**Gambar 113. Rangkaian untuk soal nomor 8 Bab V**

9. Perhatikan peta Karnaugh berikut ini:

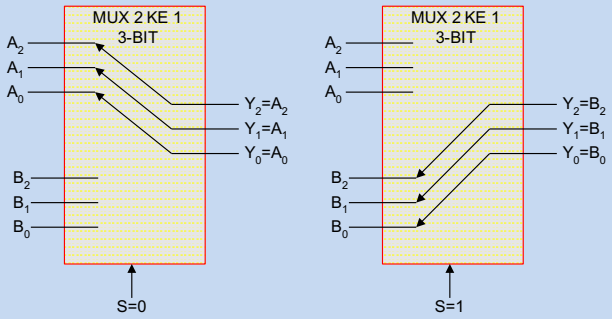
| CD \ AB | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 1 | 0 | 1 |
| 01 | 1 | 0 | 1 | 0 |
| 11 | 0 | 1 | 0 | 1 |
| 10 | 1 | 0 | 1 | 0 |

Implementasikan peta Karnaugh di samping ini dengan menggunakan gerbang XOR!

**Gambar 114. Peta Karnaugh untuk soal nomor 3 Bab V**

10. Implementasikan $Y(A,B,C,D) = \sum m(0,3,5,6,9,10,12,15)$ menggunakan gerbang XNOR!

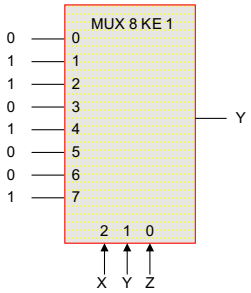
11. Susun rangkaian *full adder* paralel 6-bit, dan tunjukkan operasi rangkaian tersebut dalam melakukan operasi aritmetika +5+4, +5-4, -5+4, dan -5-4!
12. Gambar berikut ini menunjukkan rangkaian analogi *multiplexer* 2 ke 1 dengan panjang data 3-bit!




**Gambar 115. Rangkaian untuk soal nomor 12 Bab V**

Berdasarkan rangkaian analogi tersebut, susun tabel kebenaran, persamaan output untuk $Y_2$, $Y_1$, dan $Y_0$ serta gambarkan rangkaian logikanya!

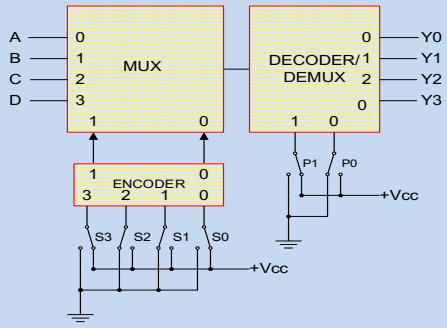
13. Suatu *multiplexer* 8 ke 1 pada inputnya terpasang data-data seperti pada gambar 116.




Tulislah keadaan output Y untuk berbagai nilai sinyal kendali XYZ!

**Gambar 116. Keadaan input MUX 8 ke 1 untuk soal nomor 7 Bab V**

14. Susunlah rangkaian DEMUX 1 ke 4 yang memiliki 2 buah input yakni $\overline{G}$, dan I dengan output jenis *active-low*!
15. Susun tabel kebenaran, persamaan output, dan rangkaian *encoder* desimal ke kode XS-3 dengan output jenis *active-high*! Susun pula rangkaian yang sama untuk output jenis *active-low*!
16. Dengan menggunakan IC 74147 *encoder* desimal ke BCD susunlah rangkaian *encoder* oktal ke biner!
17. Perhatikan diagram blok berikut ini!




**Gambar 117. Diagram blok untuk soal nomor 17 Bab V**

Dengan anggapan *encoder* yang digunakan adalah jenis prioritas, tulislah keadaan output *decoder* $Y_0$, $Y_1$, $Y_2$, dan $Y_3$!

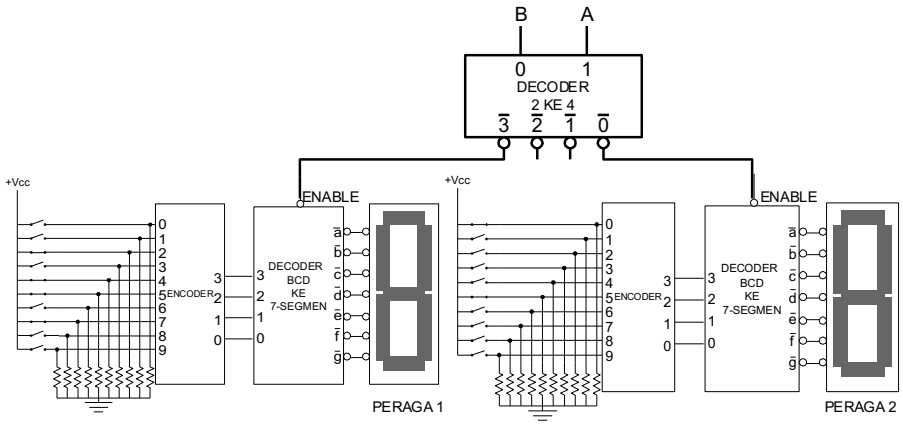
18. Dengan menggunakan gambar 117, dan KA menunjukkan posisi saklar ke kanan, serta KI posisi saklar ke kiri, isilah tabel berikut ini:

**Tabel 40. Tabel untuk soal nomor 12 Bab V**

| SAKLAR ENCODER | | | | SAKLAR DECODER | | OUTPUT DECODER/DEMUX | | | |
|---|---|---|---|---|---|---|---|---|---|
| S3 | S2 | S1 | S0 | P1 | P0 | Y0 | Y1 | Y2 | Y3 |
| KI | KI | KI | KA | KA | KI | ….. | ….. | ….. | ….. |
| KI | KA | KA | KI | KA | KA | ….. | ….. | ….. | ….. |
| KI | KI | KA | KI | KI | KI | ….. | ….. | ….. | ….. |
| KA | KA | KI | KI | KI | KA | ….. | ….. | ….. | ….. |
| KI | KI | KA | KI | KA | KI | ….. | ….. | ….. | ….. |
| KI | KA | KI | KI | KI | KI | ….. | ….. | ….. | ….. |

19. Implementasikan rangkaian *full adder* 1-bit dengan menggunakan (a) dua buah *multiplexer* 8 ke 1, dan (b) dua buah *decoder* 3 ke 8.

20. Dari gambar berikut ini, apa yang tejadi dengan peraga 1 dan peraga 2 jika diberikan sinyal AB=00, AB=01, AB=10, dan AB=11?




**Gambar 118. Rangkaian untuk soal nomor 14 Bab V**

## KOMPETENSI DASAR VI

1. Mahasiswa memahami watak dan cara kerja elemen rangkaian sekuensial flip-flop (FF)
2. Mahasiswa memahami dasar-dasar analisis dan perancangan rangkaian logika sekuensial

## TUJUAN PEMBELAJARAN VI

Agar mahasiswa dapat:
1. mendefinisikan pengertian rangkaian sekuensial
2. mendefinisikan flip-flop jenis SR, JK, D, dan T.
3. menggambarkan rangkaian dan simbol serta menuliskan persamaan Boole FFSR, JK, D, serta T
4. menjelaskan watak flip-flop jenis SR, JK, D, dan T melalui tabel kebenaran dan diagram waktu
5. menjelaskaan cara kerja dari flip-flop SR, JK, D, serta T
6. menyebutkan seri IC TTL dan susunan pin yang tersedia dari flip-flop SR, JK, D, serta T
7. membuat tabel kebenaran dan diagram transisi dari suatu persamaan/rangkaian sekuensial
8. menuliskan persamaan output dari rangkaian sekuensial yang diketahui
9. membuat diagram transisi dari definisi rangkaian sekuensial
10. menuliskan persamaan berdasarkan diagram transisi yang diperoleh
11. menggambar rangkaian berdasarkan persamaan sekuensial yang diketahui

## GARIS BESAR MATERI VI

Logika sekuensi merupakan rangkaian logika yang keadaan outputnya selain tergantung pada keadaan input-inputnya juga tergantung pada keadaan output sebelumnya. Dalam aplikasinya, rangkaian logika sekuensi banyak digunakan di dalam sistem komputer. Hal itu disebabkan dalam sistem komputer banyak data dikirim dari satu tempat ke tempat lainnya secara berurutan, sehingga memerlukan rangkaian sekuensi untuk menangani transfer data tersebut. Rangkaian logika ini didefinisikan pula sebagai rangkaian logika yang outputnya tergantung pada waktu. Bagian-bagian rangkaian logika sekuensi terdiri atas rangkaian logika kombinasi dan unit penyimpan. Melalui bagian ini Anda akan diperkenalkan terlebih dahulu dengan unit penyimpan dalam suatu rangkaian sekuensi yang dinamakan flip-flop. Flip-flop merupakan elemen rangkaian logika sekuensi yang berfungsi menyimpan data 1-bit, sehingga elemen ini dinamakan pula memori 1-bit. Berbagai jenis flip-flop akan diperkenalkan seperti flip-flop Set-Reset, flip-flop J-K, flip-flop J-K *master-slave*, flip-flop D, dan flip-flop T.

Salah satu ciri rangkaian logika sekuensi adalah pengaktifan dari elemen-elemennya dilakukan dengan menggunakan pulsa *clock* (detak). *Clock* merupakan pulsa atau denyut listrik periodik yang memiliki periode tertentu. Melalui bagian ini Anda akan diperkenalkan dengan karakteristik *clock* yang diperlukan untuk mengaktifkan elemen rangkaian sekuensi dan sekaligus akan dikenalkan pula dengan karakteristik flip-flop berdasarkan jenis pulsa aktivasinya atau pemicunya.

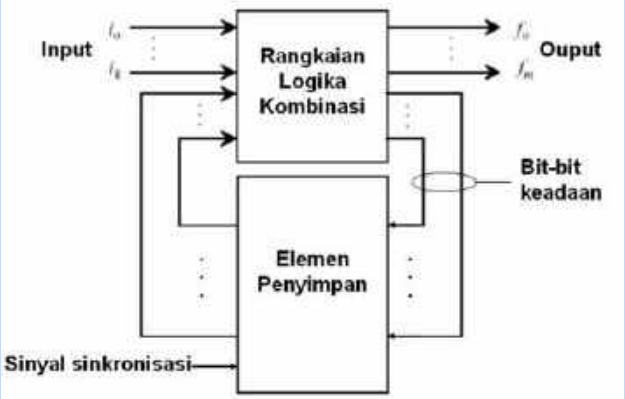
Selanjutnya, Anda akan diberi penjelasan tentang pengertian rangkaian logika sekuensi dan prosedur analisisnya yang mencakup penyusunan tabel keadaan (*state table*) dari suatu rangkaian sekuensi yang diketahui, penggambaran diagram keadaan (*state diagram*) dari tabel keadaan yang diperoleh, dan penurunan persamaan keadaan serta persamaan output yang mencerminkan karakteristik dari rangkaian sekuensi yang dianalisis.

Bagian akhir dari bab ini akan menjelaskan kepada Anda tentang teknik perancangan rangkaian logika sekuensi. Penjelasannya mencakup cara pendefinisian atau penetapan watak rangkaian logika sekuensi yang akan dirancang, cara penyusunan tabel keadaan atau diagram transisi keadaan berdasarkan definisi watak rangkaian, penentuan spesifikasi elemen penyimpan meliputi banyak dan jenis flip-flop yang digunakan, dan penyusunan tabel eksitasi flip-flop untuk menentukan fungsi input flip-flop penyusun rangkaian dan fungsi output rangkaian, serta penggambaran rangkaian sekuensi atas dasar persamaan output dan fungsi-fungsi input flip-flop yang diperoleh.

# BAB VI
# RANGKAIAN LOGIKA SEKUENSI

## A. Pengertian Logika Sekuensi

Pada bab V di muka telah dijelaskan bahwa secara umum terdapat dua jenis rangkaian logika yakni logika kombinasi dan logika sekuensi. Rangkaian jenis pertama outputnya hanya tergantung pada keadaan input-inputnya saja, sedangkan pada rangkaian logika sekuensi selain outputnya tergantung pada keadaan input-inputnya juga tergantung pada keadaan output sebelumnya. Oleh karena itu, pada rangkaian logika sekuensi terdapat unit penyimpan untuk mengingat keadaan output sebelumnya. Diagram blok rangkaian logika sekuensi ditunjukkan pada gambar 119.




**Gambar 119. Diagram blok rangkaian sekuensi**

Rangkaian logika kombinasi pada gambar 119 melakukan pemrosesan terhadap sinyal-sinyal input dan salah satu outputnya adalah bit-bit keadaan yang merupakan perangsang (*excitation*) bagi elemen penyimpan. Bit-bit perangsang ini akan menentukan keadaan yang harus disimpan dalam elemen penyimpan yang berguna untuk menentukan keadaan output berikutnya. Terlihat bahwa output rangkaian logika sekuensi merupakan fungsi dari keadaan input sekarang dan keadaan-keadaan output penyimpan yang mewakili keadaan output sebelumnya.

Dalam operasinya, bagian elemen penyimpan pada rangkaian logika sekuensi dilengkapi dengan input sinyal sinkronisasi. Sinyal ini berbentuk *clock* yang merupakan pulsa listrik periodik. Fungsi utama *clock* pada rangkaian ini adalah untuk sinkronisasi

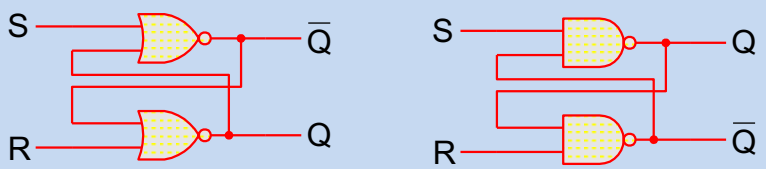
yakni suatu tindakan mengaktifkan beberapa elemen penyimpan secara bersama-sama. Atas dasar pengaktifan elemen penyimpanannya, rangkaian logika sekuensi terbagi menjadi rangkaian sekuensi serempak (*synchronous sequential circuit*) dan rangkaian sekuensi tak serempak (*asynchronous sequential circuit*). Rangkaian logika sekuensi serempak merupakan rangkaian logika sekuensi yang bekerja memroses suatu sinyal input ketika suatu sinyal sinkronisasi mengaktifkan semua elemen penyimpanannya secara bersama-sama. Sedangkan rangkaian sekuensi tak serempak merupakan rangkaian logika sekuensi yang pengaktifan elemen-elemen penyimpannya berdasarkan urutan sinyal yang masuk sehingga elemen-elemen penyimpannya tidak bekerja secara bersamaan.

**B. Flip-flop**

Elemen penyimpan rangkaian logika sekuensi adalah flip-flop. Flip-flop merupakan sel biner yang mampu menyimpan data 1-bit, sehingga sel ini dinamakan pula memori 1-bit. Ciri-ciri flip-flop yang paling menonjol adalah memiliki dua buah output, yakni satu buah untuk output dari data yang disimpan dan lainnya merupakan komplemennya. Berbagai jenis flip-flop dapat ditemukan pada bidang teknik digital di antaranya adalah flip-flop Set-Reset, flip-flop JK, flip-flop JK Slave-Master, Flip-flop D, dan flip-flop T.

**1. Flip-Flop Set-Reset**

Sesuai dengan namanya, flip-flop set-reset atau disingkat flip S-R merupakan memori yang melakukan penyimpanan data dengan cara memberi sinyal pada input Set (S) dan Reset (R) yang dimilikinya. Gambar berikut ini menunjukkan rangkaian flp-flop set-reset.




**Gambar 120. Rangkaian flip-flop Set-Reset: (a) menggunakan gerbang NOR, dan (b) dengan gerbang NAND**

Jika $Q_n$ merupakan keadaan output sekarang dan $Q_{n-1}$ keadaan output sebelumnya, maka persamaan output flip-flop S-R yang dibangun dengan menggunakan gerbang NOR seperti pada gambar 120 (a), dalam keadaan tak stabil, dapat dinyatakan dalam:

$$Q_n = \overline{R + \overline{Q_{n-1}}}$$
$$\overline{Q_n} = \overline{S + Q_{n-1}} \quad \text{persamaan (48)}$$

Sedangkan untuk flip-flop yang dibangun dengan gerbang NAND, persamaan outputnya dinyatakan dalam:

$$Q_n = \overline{S.\overline{Q_{n-1}}}$$
$$\overline{Q_n} = \overline{R.Q_{n-1}} \quad \text{persamaan (49)}$$

Dengan menggunakan persamaan (48), dalam keadaan stabil, watak atau karakteristik flip-flop dengan gerbang NOR dapat dituangkan dalam suatu tabel kebenaran seperti ditunjukkan pada tabel 41.

**Tabel 41. Tabel kebenaran flip-flop S-R dengan gerbang NOR**

| INPUT | | | OUTPUT | | |
|---|---|---|---|---|---|
| S | R | $Q_{n-1}$ | $Q_n$ | $\overline{Q_n}$ | KEADAAN |
| 1 | 0 | 0 | 1 | 0 | Set ($Q_n$=1) |
| 1 | 0 | 1 | 1 | 0 | |
| 0 | 1 | 1 | 0 | 1 | Reset ($Q_n$=0) |
| 0 | 1 | 0 | 0 | 1 | |
| 0 | 0 | 0 | 0 | 1 | Tetap ($Q_n$=$Q_{n-1}$) |
| 0 | 0 | 1 | 1 | 0 | |
| 1 | 1 | 0 | ? | ? | Terlarang ($Q_n$=?) |
| 1 | 1 | 1 | ? | ? | |

Tabel kebenaran tersebut dapat disederhanakan lagi menjadi tabel 42 dan untuk flip-flop S-R dengan gerbang NAND pada tabel 43.

**Tabel 42. Tabel kebenaran sederhana flip-flop S-R dengan NOR**

| INPUT | | OUTPUT |
|---|---|---|
| S | R | $Q_n$ |
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 0 | 0 | $Q_{n-1}$ |
| 1 | 1 | ? |

**Tabel 43. Tabel kebenaran sederhana flip-flop S-R dengan**

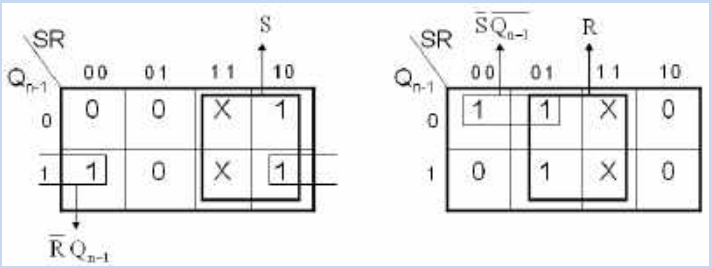
| INPUT | | OUTPUT |
|---|---|---|
| S | R | $Q_n$ |
| 1 | 0 | 0 |
| 0 | 1 | 1 |
| 0 | 0 | ? |
| 1 | 1 | $Q_{n-1}$ |

Berdasarkan tabel 41, dapat disusun persamaan output flip-flop S-R jenis NOR untuk keadaan stabil, dan perlu dibuat terlebih dahulu tabel 44 yang merupakan tabel penyesuaian terhadap tabel 41.

**Tabel 44. Tabel kebenaran flip-flop S-R untuk penyusunan peta Karnaugh**

| INPUT | | | OUTPUT | |
|---|---|---|---|---|
| S | R | $Q_{n-1}$ | $Q_n$ | $\overline{Q_n}$ |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | X | X |
| 1 | 1 | 1 | X | X |

Dari tabel tersebut dapat disusun peta Karnaugh untuk $Q_n$ dan $\overline{Q_n}$ sebagai berikut:




(a) (b)

**Gambar 121. Peta Karnaugh output flip-flop S-R untuk: (a) $Q_n$, dan (b) $\overline{Q_n}$**

Atas dasar peta Karnaugh yang telah disusun pada gambar 121, dapat diperoleh persamaan output flip-flop S-R sebagai berikut:

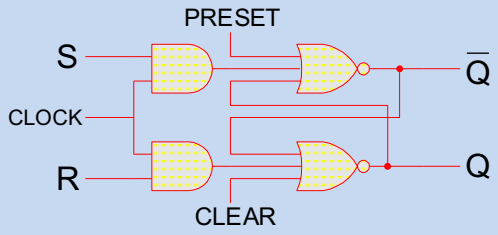
$$Q_n = S + \overline{R}Q_{n-1}$$
$$\overline{Q_n} = R + \overline{S}\,\overline{Q_{n-1}}$$
$$SR = 0 \qquad \text{persamaan (50)}$$

$Q_n$ dan $\overline{Q_n}$ pada persamaan (50) merupakan persamaan output flip-flop S-R yang berlaku pada keadaan stabil. Pencantuman syarat SR=0 diperlukan karena flip-flop S-R tidak boleh

diberi input S dan R tinggi atau S=1 dan R=1. Dengan kata lain, $Q_n$ dan $\overline{Q_n}$ berlaku jika SR=0.

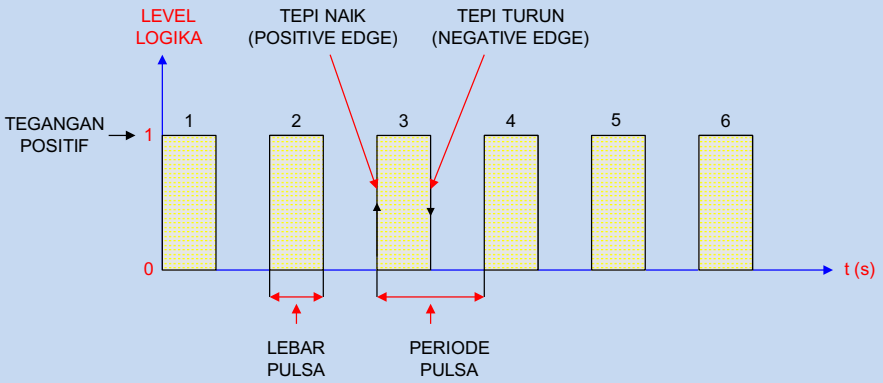
## 2. Flip-flop S-R Canggih

Flip-flop S-R yang telah dibahas di muka konfigurasinya sangat sederhana yakni hanya terdiri dari dua gerbang NOR atau dua gerbang NAND dan hanya memiliki input set dan reset saja. Selain itu, terdapat flip-flop S-R jenis yang lebih rumit, oleh karenanya disebut flip-flop S-R canggih (*sophisticated S-R flip flop*). Kerumitan yang ada pada flip-flop jenis ini disebabkan karena adanya tambahan fungsi seperti input *clock* untuk sinkronisasi atau pengaktifan, sehingga elemen penyimpan ini dinamakan juga *clocked set reset flip flop*, atau flip-flop S-R yang dilengkapi dengan *clock*. Selain input *clock*, flip-flop ini dilengkapi juga dengan input *preset* dan *clear*. Input *preset* digunakan untuk memberikan set awal, dan aksinya tidak terpengaruh oleh *clock,* dan input *clear* digunakan untuk memberikan reset awal, dan aksinya juga tidak terpengaruh oleh *clock*. Pulsa sinkronisasi *clock* hanya berpengaruh terhadap input S dan R, dalam hal ini S dan R akan memberikan pengaruh pada watak flip-flop jika ada input *clock*. Rangkaian flip-flop S-R jenis ini disajikan pada gambar 122.




**Gambar 122. Rangkaian flip-flop S-R yang dilengkapi dengan clock**

Dari gambar 122 terlihat bahwa flip-flop jenis ini memiliki fasilitas input *clock* sebagai input pengaktifan atau sinkronisasi. *Clock* adalah pulsa atau denyut listrik periodik yang berfungsi mengaktifkan elemen/rangkaian logika. Pulsa *clock* dapat digambarkan sebagai berikut:

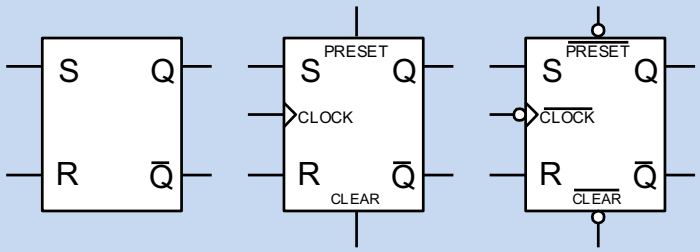



**Gambar 123. Pulsa *clock***

Pada gambar 123 ditunjukkan deretan pulsa *clock* terdiri atas 6 pulsa yakni pulsa ke-1 sampai dengan pulsa ke-6. Hal yang paling penting dari informasi *clock* ini adalah selain pulsa-pulsa itu memiliki periode dan lebar pulsa, juga setiap pulsa memiliki dua keadaan. Keadaan pertama adalah pulsa berubah dari keadaan rendah atau 0 ke keadaan tinggi 1, dan kejadian itu terjadi pada tepi naik atau tepi positif dari pulsa. Keadaan kedua adalah pulsa berubah dari keadaan tinggi atau 1 ke keadaan rendah atau 0, dan hal itu terjadi pada tepi turun atau tepi negatif. Pada umumnya, pengaktifan elemen logika yang dilakukan oleh *clock* hanya terjadi pada kedua peristiwa itu, artinya elemen atau rangkaian logika hanya akan aktif pada saat pulsa berubah dari 0 ke 1 atau pada saat berubah dari 1 ke 0 tergantung jenisnya. Berdasarkan pulsa pengaktifannya, elemen logika dibagi menjadi elemen-elemen yang diaktifkan pada tepi naik dinamakan elemen jenis *positive-edge triggered,* dan elemen-elemen yang diaktifkan pada tepi turun disebut elemen jenis *negative-edge triggered*. Jika suatu elemen atau rangkaian logika diaktifkan menggunakan *clock* seperti pada gambar 123, maka hanya akan terdapat 6 keadaan aktif karena *clock* tersebut hanya memiliki 6 tepi naik atau 6 tepi turun saja. Dalam hal ini, elemen logika itu akan diaktifkan tidak pada sembarang waktu namun hanya pada saat-saat tertentu saja yakni saat terjadinya tepi naik atau saat terjadinya tepi turun.

Simbol flip-flop S-R sederhana dan canggih yang dilengkapi dengan *clock* ditunjukkan pada gambar 124. Pada gambar 124 (a) diperlihatkan simbol flip-flop S-R sederhana, sedangkan gambar 124 (b) dan (c) simbol untuk flip-flop S-R canggih yang dilengkapi dengan *clock*. Gambar 124 (b) menunjukkan simbol untuk jenis *positive-edge*

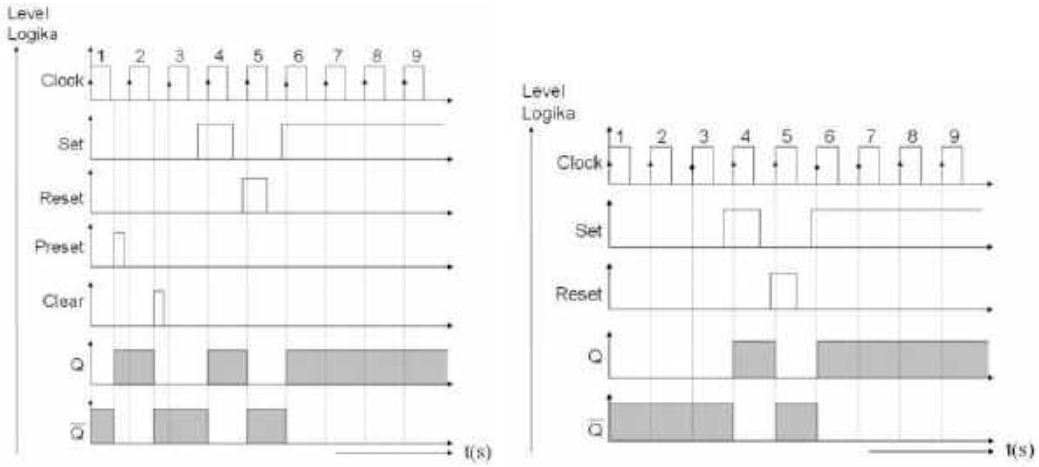
*triggered* dengan input *preset* serta input *clear* jenis *active-high*, sedangkan gambar 124 (c) simbol untuk jenis *negative-edge triggered* dengan *preset* serta *clear* jenis *active-low*.




(a) (b) (c)

**Gambar 124. Simbol flip-flop S-R: (a) sederhana, (b) dan (c) canggih**

Untuk mempelajari watak elemen logika yang dilengkapi dengan *clock* umumnya digunakan suatu diagram waktu. Dengan menggunakan diagram waktu dapat diamati watak elemen atau rangkaian logika setiap waktu. Gambar 125 menunjukkan diagram waktu untuk flip-flop S-R yang dilengkapi dengan *clock*.



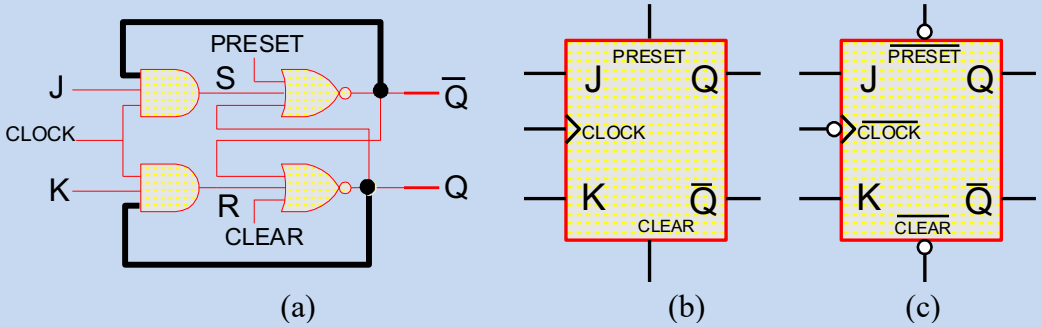

(a) (b)

**Gambar 125. Diagram waktu flip-flop S-R: (a) *preset* dan *clear* diaktifkan, dan (b) tanpa *preset* dan *clear***

### 3. Flip-flop J-K

Kelemahan flip-flop S-R adalah munculnya output yang tidak dapat didefinisikan ketika input S dan R tinggi untuk jenis NOR dan rendah untuk jenis NAND. Untuk menanggulangi munculnya keadaan tersebut, maka dikembangkan flip-flop J-K. Jadi, flip-

flop J-K dibangun untuk mengantisipasi keadaan *terlarang* pada flip-flop S-R, dan rangkaiannya ditunjukkan pada gambar 126.




(a) (b) (c)

**Gambar 126. Flip-flop J-K: (a) rangkaian, (b) simbol untuk jenis *positive-edge triggered*, dan (c) simbol untuk jenis *negative-edge triggered***

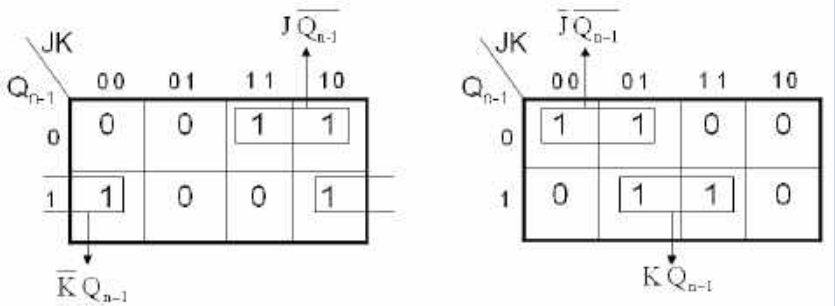
Konfigurasi tersebut telah dapat menghilangkan keadaan terlarang yang tejadi pada flip-flop S-R. Untuk melihat pengaruh pemberian J dan K pada output flip-flop, perhatikan tabel 45 berikut ini dengan asumsi *clock* bernilai tinggi, *preset* dan *clear* bernilai rendah.

**Tabel 45. Tabel kebenaran flip-flop J-K**

| INPUT | | | | | OUTPUT | | |
|---|---|---|---|---|---|---|---|
| J | K | $S = J\overline{Q_{n-1}}$ | $R = KQ_{n-1}$ | $Q_{n-1}$ | $Q_n$ | $\overline{Q_n}$ | KEADAAN |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | Set ($Q_n$=1) |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | Reset ($Q_n$=0) |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | Tetap ($Q_n$=$Q_{n-1}$) |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | |
| 1 | 1 | 1 | 0 | 0 | 1 | 0 | Komplemen ($Qn=\overline{Q_{n-1}}$) |
| 1 | 1 | 0 | 1 | 1 | 0 | 1 | |

Perhatikan bahwa pemberian input J=1 dan K=1 menjadikan output flip-flop melakukan pembalikan terhadap keadaan output sebelumnya.

Dari tabel 45, dengan J, K, dan $Q_{n-1}$ sebagai input dan $Q_n$ sebagai output, maka dapat disusun peta Karnaugh untuk menentukan persamaan output flip-flop J-K sebagai berikut:

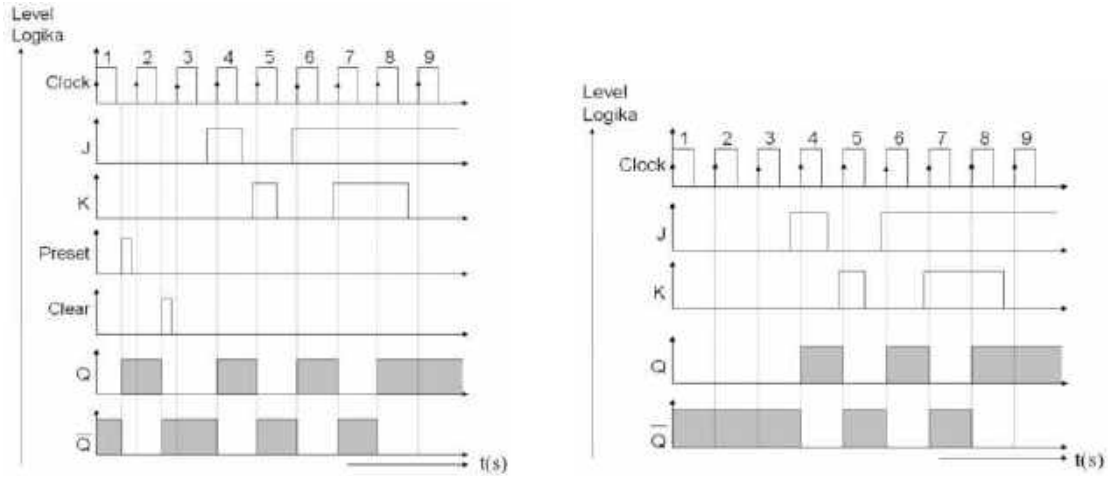



**Gambar 127. Peta Karnaugh output flip-flop J-K untuk: (a) $Q_n$, dan (b) $\overline{Q_n}$**

Dengan demikian persamaan output flip-flop J-K yang dihasilkan dari peta Karnaugh pada gambar 127 dapat dituliskan seperti pada persamaan (51).

$$Q_n = J\overline{Q_{n-1}} + \overline{K}Q_{n-1}$$
$$\overline{Q_n} = \overline{J}\overline{Q_{n-1}} + KQ_{n-1} \qquad \text{persamaan (51)}$$

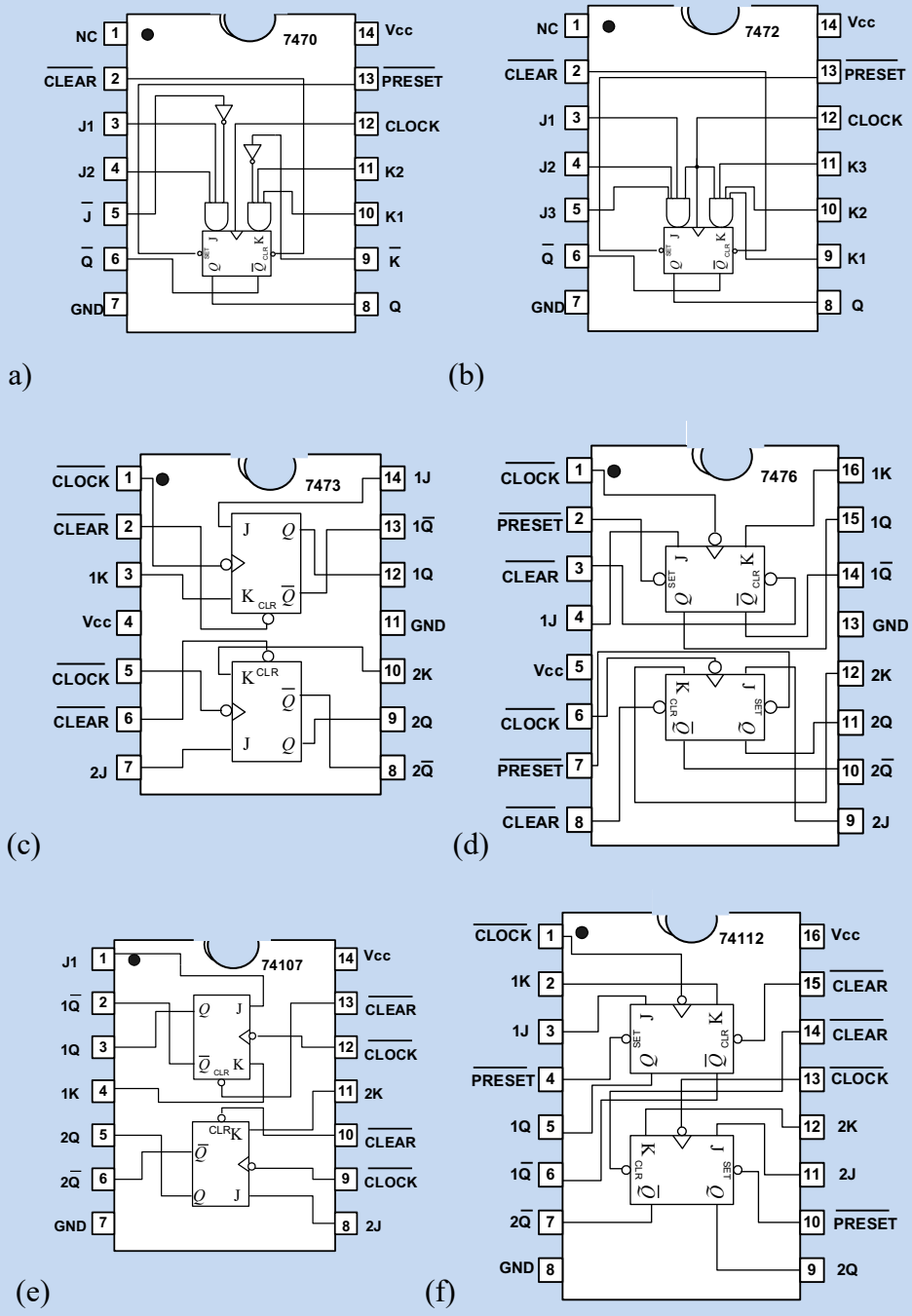
Gambar 128 adalah contoh diagram waktu dari flip-flop J-K jenis *positive-edge triggered* dan *preset*, serta *clear* jenis *active-high*. Pulsa *clock* yang diberikan sebanyak 9 buah dimulai dari pulsa ke-1 sampai dengan pulsa ke-9.




(a) (b)

**Gambar 128. Contoh diagram waktu flip-flop J-K: (a) *preset* dan *clear* diaktifkan, dan (b) tanpa *preset* dan *clear***

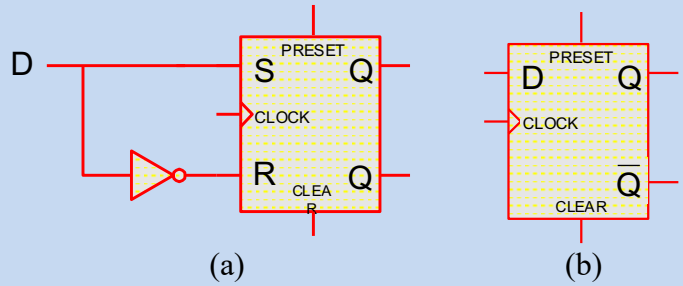
Dalam kemasan IC, flip-flop J-K disediakan oleh IC dengan nomor seri 7470 (J-K *positive-edge triggered*), 7472 (J-K *master-slave*), 7473, 7476, 74107, 74109, dan 74112 (J-K *negative-edge triggered*) dengan spesifikasi pin seperti pada gambar 129.




**Gambar 129. Spesifikasi pin IC flip-flop J-K**

## 4. Flip-flop D

Selain flip-flop S-R dan J-K terdapat pula flip-flop D. Sesuai dengan namanya, input flip-flop ini adalah D. Flip-flop D dibangun dengan menggunakan flip-flop S-R seperti ditunjukkan pada gambar 130.




**Gambar 130. Flip-flop D: (a) rangkaian, dan (b) simbol**

Dengan adanya gerbang NOT yang masuk ke input R, maka setiap input yang diumpankan ke D akan memberikan keadaan yang berbeda pada input S dan R. Dengan demikian hanya akan terdapat dua keadaan dari S dan R yakni S=0 dan R=1 atau S=1 dan R=0. Jadi, output flip-flop D juga hanya memiliki dua keadaan yakni keadaan set atau keadaan reset. Tabel kebenaran flip-flop D ditunjukkan pada tabel 46.

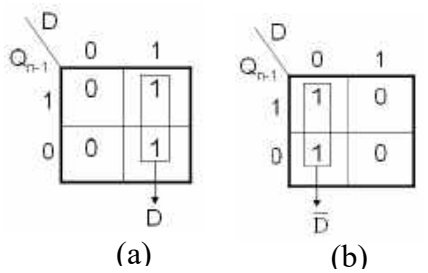
**Tabel 46. tabel kebenaran flip-flop D**

| INPUT | | OUTPUT | | |
|---|---|---|---|---|
| D | $Q_{n-1}$ | $Q_n$ | $\overline{Q_n}$ | KEADAAN |
| 0 | 0 | 0 | 1 | Reset (Qn=0) |
| 0 | 1 | 0 | 1 | Reset (Qn=0) |
| 1 | 0 | 1 | 0 | Set (Qn=1) |
| 1 | 1 | 1 | 0 | Set (Qn=1) |

Tabel yang lebih sederhana dapat disusun seperti disajikan pada tabel 47, sedangkan peta Karnaugh untuk menentukan persamaan output Qn dan $\overline{Q_n}$ ditunjukkan pada gambar 131.

**Tabel 47. Tabel kebenaran sederhana flip-flop D**

| INPUT | OUTPUT | |
|---|---|---|
| D | $Q_n$ | KEADAAN |
| 0 | 0 | Reset ($Q_n$=0) |
| 1 | 1 | Set($Q_n$=1) |

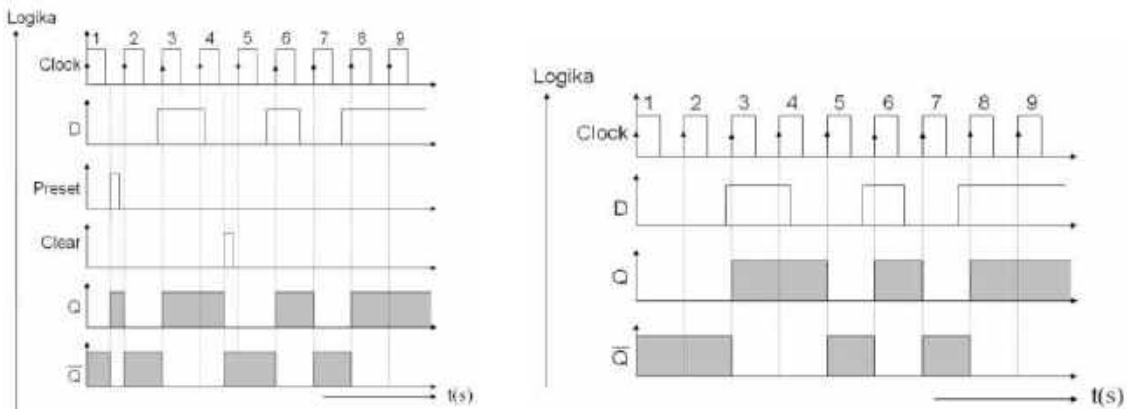



**Gambar 131. Peta Karnaugh flip-flop D untuk (a) $Q_n$, dan (b) $\overline{Q}$**

Berdasarkan peta Karnaugh pada gambar 131, persamaan output flip-flop D adalah:

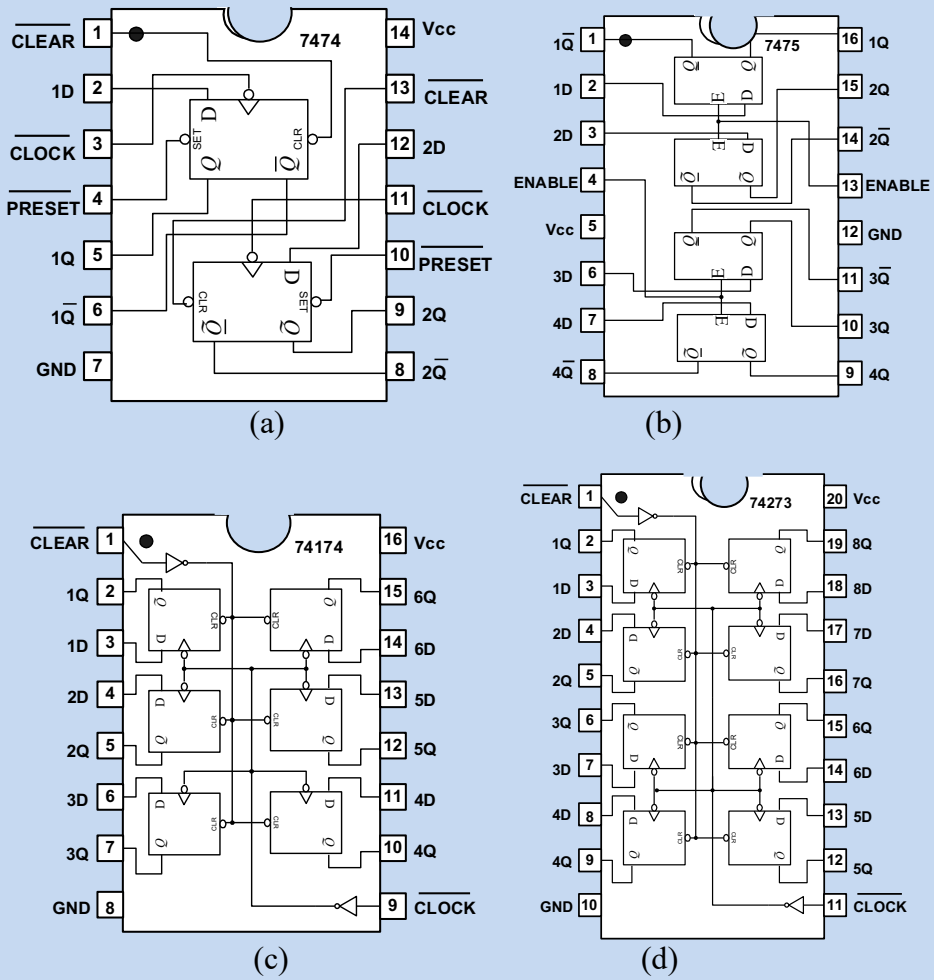
$$Q_n = D \text{ dan } \overline{Q_n} = \overline{D} \qquad \text{persamaan (52)}$$

Contoh diagram waktu untuk flip-flop D ditunjukkan pada gambar 132!




**Gambar 132. Contoh diagram waktu flip-flop D: (a) *preset* dan *clear* diaktifkan, (b) tanpa *preset* dan *clear***

IC yang menyediakan fungsi flip-flop D antara lain 7474, 7475, 74174, 74175, 74273, 74363, 74364, 74373, 74374, dan 74377. Spesifikasi pin seri-seri IC tersebut ditunjukkan pada gambar 133.

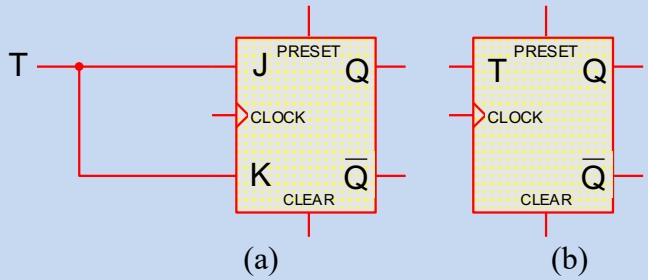



**Gambar 133. Spesifikasi pin IC flip-flop D: (a) 7474, (b) 7475, (c) 74174, 74175, (d) 74273, 74363, 74364, 74373, 74374, dan 74377**

## 5. Flip-flop T

Telah dibahas di muka bahwa flip-flop J-K memiliki watak membalik keadaan output sebelumnya jika input J dan K diberi nilai tinggi. Dengan menggunakan flip-flop J-K yang kedua inputnya dihubungkan menjadi satu maka akan diperoleh flip-flop yang memiliki watak membalik ouput sebelumnya jika inputnya tinggi, dan outputnya akan tetap

jika inputnya rendah. Flip-flop yang berfungi seperti itu dinamakan flip-flop T. Rangkaian flip-flop T ditunjukkan pada gambar 134, dan tabel kebenarannya dapat diperoleh berdasarkan tabel kebenaran flip-flop J-K. Oleh karena flip-flop T merupakan flip-flop J-K dengan dua keadaan yakni J=0, K=0 dan J=1, K=1, maka tabel kebenaran flip-flop T adalah tabel kebenaran flip-flop J-K dengan dua keadaan tersebut!




**Gambar 134. Flip-flop T: (a) rangkaian, dan (b) simbol**

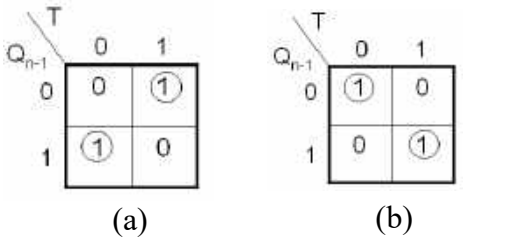
**Tabel 48. Tabel kebenaran flip-flop T**

| INPUT | | OUTPUT | | |
|---|---|---|---|---|
| T | $Q_{n-1}$ | $Q_n$ | $\overline{Q_n}$ | KEADAAN |
| 0 | 0 | 0 | 1 | Tetap($Q_n$=$Q_{n-1}$) |
| 0 | 1 | 1 | 0 | Tetap($Q_n$=$Q_{n-1}$) |
| 1 | 0 | 1 | 0 | Membalik ($Q_n = \overline{Q_{n-1}}$) |
| 1 | 1 | 0 | 1 | Membalik ($Q_n = \overline{Q_{n-1}}$) |

Tabel kebenaran yang lebih sederhana untuk flip-flop T dapat disajikan dalam bentuk seperti pada tabel 49, sedangkan peta Karnaugh untuk menentukan persamaan output flip-flop J-K ditunjukkan pada gambar 135.

**Tabel 49. Tabel kebenaran sederhana flip-flop T**

| INPUT | OUTPUT | |
|---|---|---|
| T | $Q_n$ | KEADAAN |
| 0 | $Q_{n-1}$ | Tetap |
| 1 | $\overline{Q_{n-1}}$ | Membalik |

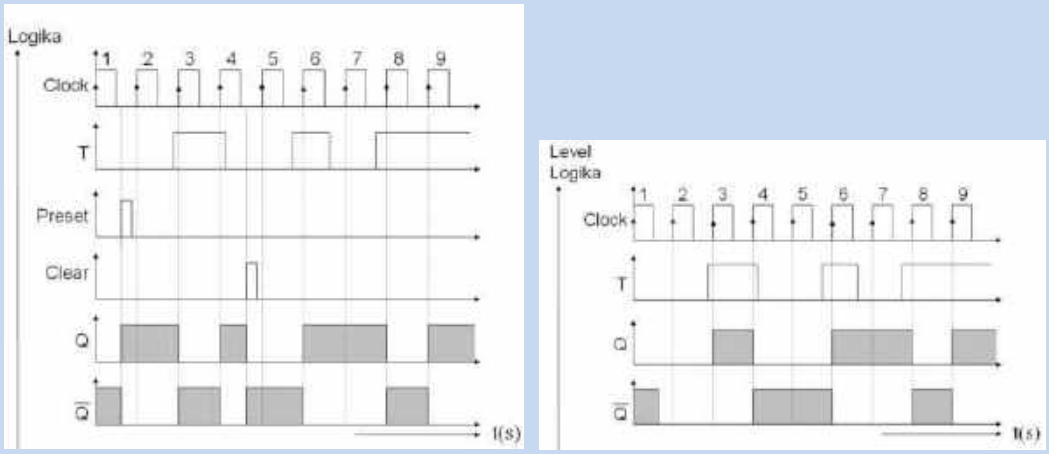



**Gambar 135. Peta Karnaugh flip-flop T untuk (a) $Q_n$, dan (b) $\overline{Q}$**

Dari peta Karnaugh pada gambar 135 terlihat bahwa persamaan output flip-flop T bentuknya adalah SOP standar karena semua minterm yang ada terisolasi, sehingga:

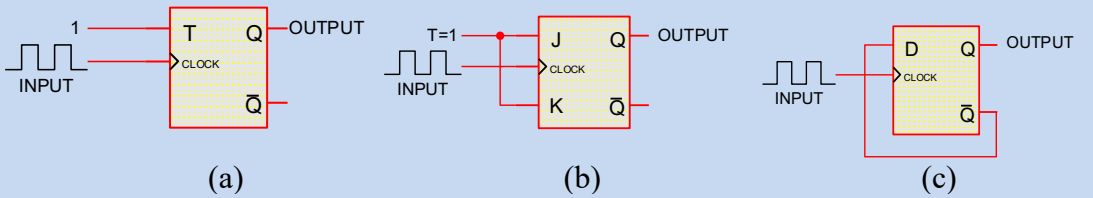
$$Q_n = T\,\overline{Q_{n-1}} + \overline{T}\,Q_{n-1}$$
$$\overline{Q_n} = \overline{T}\,\overline{Q_{n-1}} + T\,Q_{n-1} \qquad \text{persamaan (53)}$$

Pengaruh pemberian keadaan input terhadap output flip-flop T dapat ditunjukkan dengan contoh diagram waktu seperti pada gambar 136.




**Gambar 136. Contoh diagram waktu flip-flop T: (a) *preset* dan *clear* diaktifkan, (b) tanpa *preset* dan *clear***

Jika input flip-flop T dipertahankan tinggi, maka setiap terjadinya pulsa *clock* akan menyebabkan keadaan outputnya berubah. Dalam banyak aplikasi diperlukan elemen yang memiliki watak sebagai *toggle* (saklar dua keadaan) yakni outputnya berubah setiap suatu input *clock* diumpankan. Implementasi elemen tersebut dapat dilakukan dalam berbagai bentuk. Bentuk pertama adalah menggunakan flip-flop J-K yang membentuk konfigurasi flip-flop T dengan T=1, dan bentuk kedua menggunakan flip-flop D yang komplemen outputnya diumpankan ke input D. Gambar 137 menunjukkan berbagai implementasi elemen yang dapat melakukan pembalikan terhadap keadaan output sebelumnya untuk setiap terjadinya pulsa *clock*.




(a) (b) (c)

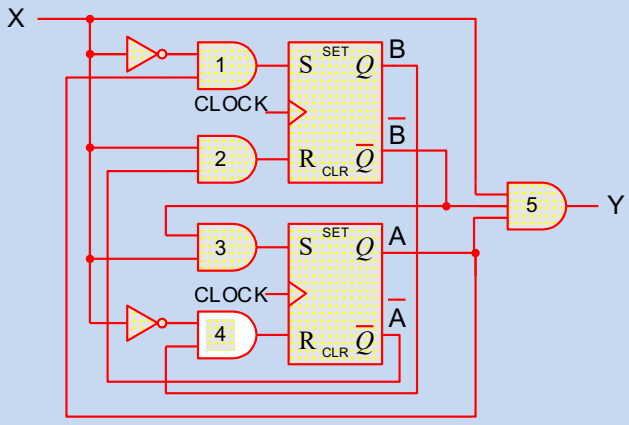
**Gambar 137. Rangkaian *toggle* dengan: (a) flip-flop T, (b) flip-flop JK, dan (c) flip-flop D**

## C. Analisis Rangkaian Sekuensi

Analisis rangkaian logika sekuensi merupakan kegiatan mengidentifikasi watak rangkaian logika sekuensi. Dengan melakukan analisis, suatu rangkaian logika sekuensi dapat segera diketahui wataknya sehingga mudah di dalam mempelajarinya. Prosedur analisis rangkaian logika sekuensi mencakup penyusunan tabel keadaan (*state table*) dari suatu rangkaian sekuensi yang diketahui, penggambaran diagram transisi keadaan (*state transition diagram*) dari tabel keadaan yang diperoleh, dan penurunan persamaan keadaan serta persamaan output yang mencerminkan karakteristik dari rangkaian sekuensi yang dianalisis. Agar Anda dapat memperoleh pemahaman yang baik terhadap cara analisis rangkaian logika sekuensi, berikut ini akan diperkenalkan dua buah contoh analisis.

**Contoh pertama:**

Lakukan analisis terhadap rangkaian sekuensi pada gambar 138 berikut ini!




**Gambar 138. Contoh rangkaian sekuensi untuk kegiatan analisis**

Langkah pertama kegiatan analisis rangkaian logika sekuensi adalah menentukan terlebih dahulu tabel keadaan. Selain variabel input dan ouput, dalam tabel keadaan terkandung informasi tentang keadaan sebelumnya dan keadaan sekarang dari variabel-variabel keadaan yakni output dari elemen penyimpannya. Untuk menyusun tabel keadaan, perlu dimisalkan keadaan awal atau keadaan sebelumnya dari output flip-flop penyusunnya. Dalam berbagai aplikasi, keadaan awal umumnya bernilai 0, dan untuk memperoleh keadaan tersebut flip-flop dilengkapi dengan fasilitas *clear* dan *preset*. Namun demikian beberapa rangkaian logika sekuensi memiliki keadaan awal yang berbeda dan bahkan keadaan awalnya ada yang tidak didefinisikan. Walaupun terdapat keadaan awal yang beragam namun tetap dapat digunakan untuk keperluan analisis rangkaian sekuensi. Dalam contoh ini keadaan awal ditentukan bernilai 0, artinya output kedua flip-flop S-R pada rangkaian bernilai 0 sehingga $A_{n-1}=0$ dan $B_{n-1}=0$.

Agar lebih mudah di dalam menentukan keadaan sekarang dari output rangkaian Y maupun output flip-flop $A_n$ dan $B_n$, perlu dirumuskan terlebih dahulu persamaan output AND 1, AND 2, AND 3, AND 4, dan AND 5. Dari rangkaian gambar 138 dapat diperoleh persamaan (54).

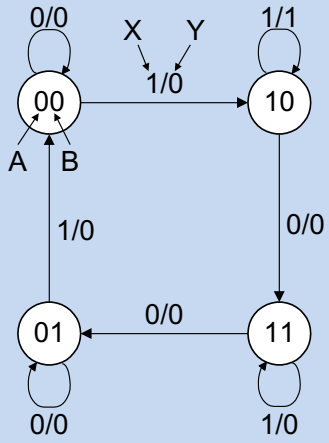
$$S_B = \overline{X}\, A_{n-1}$$
$$R_B = X\, \overline{A_{n-1}}$$
$$S_A = X\, \overline{B_{n-1}}$$
$$R_A = \overline{X}\, B_{n-1}$$
$$Y = X\, A_{n-1}\, \overline{B_{n-1}} \qquad \text{persamaan (54)}$$

Dari persamaan (54) dapat ditentukan tabel keadaan berikut ini:

**Tabel 50. Tabel keadaan rangkaian gambar 138**

| KEADAAN SEBELUMNYA | | KEADAAN SEKARANG | | | | OUTPUT | |
|---|---|---|---|---|---|---|---|
| | | X=0 | | X=1 | | X=0 | X=1 |
| $A_{n-1}$ | $B_{n-1}$ | $A_n$ | $B_n$ | $A_n$ | $B_n$ | Y | Y |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 |

Berdasarkan tabel keadaan yang diperoleh tersebut dapat disusun diagram transisi keadaan yang mencerminkan perubahan-perubahan yang terjadi setiap rangkaian sekuensi itu memperoleh pulsa pemicuan dari *clock*. Diagram transisi keadaan untuk tabel 50 disajikan pada gambar 139. Cara melukis diagram transisi keadaan dimulai dari keadaan awal atau keadaan sebelumnya. Pada contoh ini keadaan sebelumnya dari output flip-flop A dan B adalah $A_{n-1}$=0 dan $B_{n-1}$=0 atau disingkat 00. Dari baris ke-1 tabel 50 terlihat bahwa untuk X=0 akan memberikan output Y=0 dan untuk keadaan sebelumnya 00 tidak memberikan perubahan pada output flip-flop A dan B pada keadaan sekarang. Keadaan ini digambarkan dengan menggunakan lingkaran kiri atas yang di dalamnya terdapat simbol keadaan sebelumnya yakni 00, dan karena tidak ada perubahan maka di atas lingkaran tersebut diberi tanda lup yang di atasnya diberi tanda 0/0 yang menunjukkan keadaan tersebut berlaku untuk X=0 (angka 0 sebelah kiri) dan menghasilkan output Y=0 (angka 0 sebelah kanan). Selanjutnya, untuk X=1 keadaan sebelumnya 00 memberikan perubahan 10 pada output kedua flip-flop dan memberikan nilai Y=0. Diagramnya digambar dengan cara memberi tanda panah dari lingkaran kiri atas ke lingkaran kanan atas yang di dalamnya dicantumkan nilai perubahannya yakni 10, dan di atas garis berpanah tersebut diberi tanda 1/0 yang menunjukkan input X=1 dan output Y=0. Jika keadaan-keadaan semua baris pada tabel 50 dituangkan dalam diagram transisi keadaan, maka akan diperoleh diagram seperti pada gambar 139.



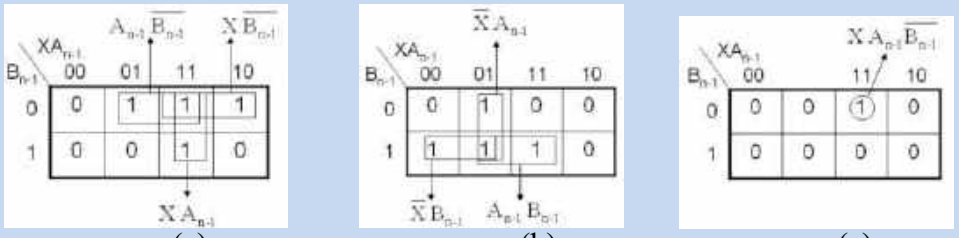

**Gambar 139. Diagram transisi keadaan untuk tabel 50.**

Dengan menggunakan diagram transisi keadaan tersebut, maka watak rangkaian sekuensi dapat segera diidentifikasi. Dari diagram itu terlihat dengan jelas pengaruh pemberian input dan keadaan sebelumnya terhadap output rangkaian. Selain menggunakan diagram transisi keadaan, watak rangkaian sekuensi juga dapat dicerminkan oleh persamaan keadaan dan persamaan outputnya. Untuk menurunkan persamaan-persamaan tersebut, tabel 50 perlu disesuaikan menjadi tabel 51.

**Tabel 51. Tabel keadaan penyesuaian dari tabel 50**

| INPUT | KEADAAN SEBELUMNYA | | KEADAAN SEKARANG | | OUTPUT |
|---|---|---|---|---|---|
| X | $A_{n-1}$ | $B_{n-1}$ | $A_n$ | $B_n$ | Y |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 |

Untuk memperoleh persamaan output, dari tabel 51 disusun terlebih dahulu peta Karnaugh untuk $A_n$, $B_n$, dan Y. Perlu dikemukakan bahwa persamaan output rangkaian sekuensi terdiri atas output rangkaian dan output dari elemen-elemen penyimpannya yang menunjukkan pengaruh pemberian input terhadap perubahan keadaan sebelumnya ke keadaan sekarang.



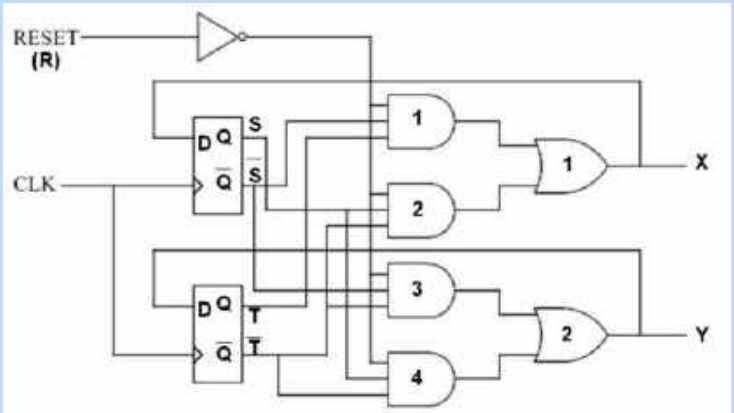

**Gambar 140. Peta Karnaugh tabel 51 untuk (a) $A_n$, (b) $B_n$, dan (c) Y**

Dari peta Karnaugh tersebut, persamaan output rangkaian sekuensi pada gambar 140 dapat ditulis seperti pada persamaan (55).

$$A_n = A_{n-1}\,\overline{B_{n-1}} + X\,\overline{B_{n-1}} + X\,A_{n-1}$$
$$B_n = A_{n-1}\,B_{n-1} + \overline{X}\,A_{n-1} + \overline{X}\,B_{n-1}$$
$$Y = X\,A_{n-1}\,\overline{B_{n-1}} \qquad \text{persamaan (55)}$$

**Contoh kedua:**

Lakukan analisis terhadap rangkaian logika sekuensi pada gambar 141!




**Gambar 141. Rangkaian untuk contoh analisis kedua**

Seperti pada contoh pertama, pada contoh kedua ini analisis rangkaian juga diawali dengan menyusun terlebih dahulu tabel keadaan. Berdasarkan rangkaian pada gambar 141 dapat disusun tabel keadaan seperti pada tabel 52. Perhatikan bahwa rangkaian sekuensi pada contoh ini menggunakan flip-flop D sebagai elemen penyimpannya. Sesuai dengan watak flip-flop D, hal itu berarti output flip-flop sekarang sama dengan input D sebelumnya. Oleh karena input D flip-flop itu diambilkan dari output X dan input D flip-flop lainnya diambilkan dari output Y, maka output flip-flop S sama dengan output X dan output flip-flop T sama dengan output Y.

Untuk reset (R) bernilai tinggi maka X dan Y bernilai 0 karena semua gerbang AND dalam keadaan *disable* yakni pada input gerbang-gerbang AND tersebut terdapat nilai 0 sehingga outputnya 0. Untuk keadaan ini, pada tabel keadaan diisi nilai 0 semua pada kolom keadaan sekarang dan kolom output X dan Y bagian R=1.

Jika R=0, dan keadaan sebelumnya $S_{n-1}$=0 dan $T_{n-1}$=0, maka pada input gerbang AND 1 dan AND 2 terdapat nilai 0 sehingga X=0. Pemberian nilai tersebut juga menyebabkan semua input AND 3 bernilai 1 sehingga output gerbang ini bernilai 1, dan walaupun output gerbang AND 4 bernilai 0, namun karena output gerbang AND 3 dan AND 4 dioperasikan dengan gerbang OR maka output Y=1. Karena $S_n$=X dan $T_n$=Y maka $S_n$=0 dan $T_n$=1. Keaadaan-keadaan tersebut tercantum pada tabel 52 baris ke-1.

Selanjutnya, jika R=0, $S_{n-1}$=0 dan $T_{n-1}$=1 akan menyebabkan output AND 1 bernilai 1 dan output AND 2 bernilai 0 sehingga X=1, dan output AND 3 serta AND 4 bernilai 0 sehingga Y=0. Pemberian nilai-nilai tersebut juga akan menyebabkan $S_n$=1 dan $T_n$=0. Keadaan-keadaan itu tertuang pada baris ke-2 tabel 52.

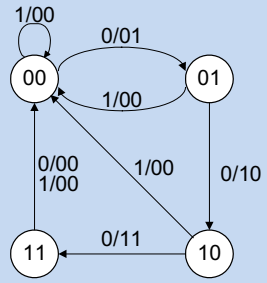
Seterusnya, jika R=0, $S_{n-1}$=1 dan $T_{n-1}$=0 akan menjadikan output gerbang AND 1 rendah dan output gerbang AND 2 tinggi sehingga X=1, dan output AND 3 rendah serta output AND 4 tinggi sehingga Y=1. Oleh karena X=1 dan Y=1 maka $S_n$=1 dan $T_n$=0. Keadaan-keadaan tersebut tercantum pada baris ke-3 tabel 52.

Terakhir, jika R=0, $S_{n-1}$=1 dan $T_{n-1}$=1 akan menyebebabkan semua output gerbang AND bernilai 0 sehingga X=0 dan Y=0 serta $S_n$=0 dan $T_n$=0.

**Tabel 52. Tabel keadaan rangkaian pada gambar 141**

| KEADAAN SEBELUMNYA | | KEADAAN SEKARANG | | | | OUTPUT | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | R=0 | | R=1 | | R=0 | | R=1 | |
| $S_{n-1}$ | $T_{n-1}$ | $S_n$ | $T_n$ | $S_n$ | $T_n$ | X | Y | X | Y |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Dari tabel keadaan tersebut dapat dilukis diagram transisi keadaan seperti ditunjukkan pada gambar 142.



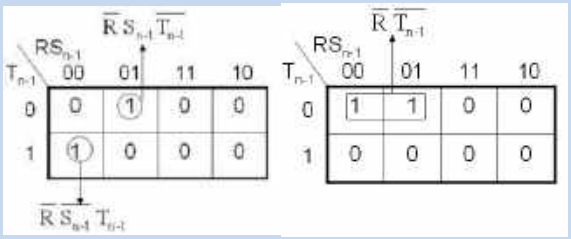

**Gambar 142. Diagram transisi keadaan untuk tabel 52**

Untuk menurunkan persamaan keadaan dan persamaan output, dilakukan penyesuaian tabel 52 menjadi tabel 53.

**Tabel 53. Tabel keadaan penyesuaian dari tabel 52**

| INPUT | KEADAAN SEBELUMNYA | | KEADAAN SEKARANG | | OUTPUT | |
|---|---|---|---|---|---|---|
| X | $S_{n-1}$ | $T_{n-1}$ | $S_n$ | $T_n$ | X | Y |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 |

Peta Karnaugh untuk tabel 53 dapat dilukis seperti pada gambar 143.




**Gambar 143. Peta Karnaugh tabel 53 untuk (a) $S_n$ dan X, dan (b) $T_n$ dan Y**

Berdasarkan peta Karnaugh pada gambar 143, persamaan keadaan dan persamaan output rangkaian sekuensi gambar 141 dapat ditulis seperti pada persamaan (56).

$$S_n = \overline{R}\,\overline{S_{n-1}}\,T_{n-1} + \overline{R}\,S_{n-1}\,\overline{T_{n-1}}$$
$$T_n = \overline{R}\,\overline{T_{n-1}}$$
$$X = \overline{R}\,\overline{S_{n-1}}\,T_{n-1} + \overline{R}\,S_{n-1}\,\overline{T_{n-1}}$$
$$Y = \overline{R}\,\overline{T_{n-1}} \qquad \text{persamaan (56)}$$

Hasil akhir analisis rangkaian yang berbentuk tabel keadaan, diagram transisi keadaan, maupun persamaan output tersebut telah menunjukkan bahwa rangkaian yang dianalisis bersifat sebagai *counter* yang berfungsi menghitung pulsa yang memicunya. Terlihat bahwa hanya sebanyak 4 buah pulsa yang mampu dihitungnya sehingga rangkaian tersebut dinamakan *counter* modulo-4. Selain dengan menggunakan pendekatan teoritis seperti yang telah dilakukan di atas, analisis rangkaian ini juga dapat dilakukan secara eksperimen.

## D. Perancangan Rangkaian Sekuensi

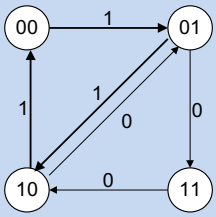
Perancangan rangkaian sekuensi dimaksudkan untuk memperoleh rangkaian logika sekuensi dari suatu watak rangkaian yang telah ditetapkan. Prosedur kegiatan perancangan ini umumnya diawali dengan pendefinisian atau penetapan watak rangkaian logika sekuensi yang akan dirancang. Langkah ini dapat berbentuk penyusunan tabel kebenaran, tabel keadaan atau diagram transisi keadaan. Langkah berikutnya adalah penentuan spesifikasi elemen penyimpan mencakup banyak dan jenis flip-flop yang digunakan, dan penyusunan tabel eksitasi flip-flop untuk menentukan fungsi input flip-flop penyusun rangkaian dan fungsi output rangkaian. Langkah terakhir adalah penggambaran rangkaian sekuensi atas dasar persamaan output dan fungsi-fungsi input flip-flop yang diperoleh.

**Contoh:**

Rancanglah rangkaian logika sekuensi dengan dua buah flip-flop dan sebuah input R. Untuk setiap adanya pulsa *clock,* jika R=1 output kedua flip-flop akan memberikan nilai desimal dengan urutan 0, 1, 2, dan kembali ke urutan semula. Apabila R=0, output kedua flip-flop akan bernilai desimal 3, 2, 1, dan kembali ke urutan semula.

**Jawab:**

Berdasarkan definisi watak tersebut, dapat dilukis diagram transisi keadaannya seperti pada gambar 144. Perhatikan bahwa rangkaian ini tidak memiliki output pada bagian logika kombinasinya, dan hanya memiliki output pada masing-masing flip-flopnya saja sehingga pada setiap garis yang menunjukkan transisi keadaan hanya dicantumkan keadaan inputnya saja yakni 1 atau 0.




**Gambar 144. Diagram transisi keadaan untuk rangkaian yang sedang dirancang**

Berdasarkan diagram transisi yang diperoleh pada gambar 144, dapat diturunkan tabel keadaan seperti disajikan pada tabel 54. Pada tabel tersebut digunakan notasi $Q_{1(n-1)}$ dan $Q_{0(n-1)}$ yang menunjukkan keadaan sebelumnya dari output flip-flop 1 dan output flip-flop 0 serta $Q_{1(n)}$ dan $Q_{0(n)}$ yang merupakan keadaan sekarang dari output kedua flip-flop.

Karena untuk keadaan sebelumnya yakni $Q_{1(n-1)}$=0 dan $Q_{0(n-1)}$=0, serta R=0, tidak terdapat peristiwa transisi keadaan, maka pada keadaan sekarang output flip-flop $Q_{1(n)}$ dan $Q_{0(n)}$ diberi tanda X, sedangkan untuk R=1 akan menyebabkan output sekarang dari kedua flip-flop bernilai 01. Keadaan-keadaan tersebut tertuang pada baris ke-1 tabel 54. Untuk keadaan sebelumnya $Q_{1(n-1)}$=0, $Q_{0(n-1)}$=1, dan R=0 memberikan keadaan sekarang 11, dan untuk R=1 memberikan keadan sekarang 10. Keadaan-keadaan tersebut tertulis pada baris ke-2 tabel 54. Selanjutnya, untuk keadaan sebelumnya 10, jika R=0 akan memberikan keadaan sekarang 01, dan jika R=1 akan memberikan keadaan sekarang 00 seperti tercantum pada baris ke-3 tabel 54. Seterusnya, jika keadaan sebelumnya 11, dan R=0 akan memberikan keadaan sekarang 10, tetapi untuk R=1 tidak terdapat transisi keadaan sehingga pada keadaan sekarang diberi tanda X.

**Tabel 54. Tabel keadaan rangkaian yang sedang dirancang**

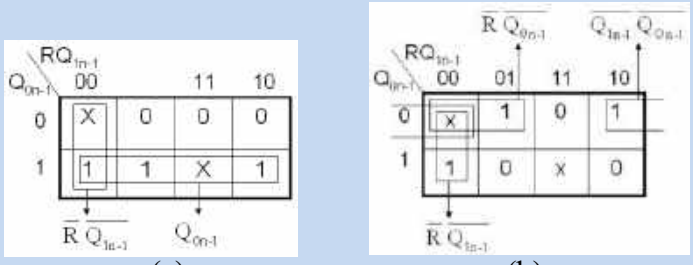
| KEADAAN SEBELUMNYA | | KEADAAN SEKARANG | | | |
|---|---|---|---|---|---|
| | | R=0 | | R=1 | |
| $Q_{1(n-1)}$ | $Q_{0(n-1)}$ | $Q_{1(n)}$ | $Q_{0(n)}$ | $Q_{1(n)}$ | $Q_{0(n)}$ |
| 0 | 0 | X | X | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | X | X |

Atas dasar tabel keadaannya dapat disusun tabel eksitasi dari flip-flop. Dalam hal ini flip-flop yang digunakan sebanyak dua buah dan jenisnya adalah flip-flop D yang memiliki watak setiap kali adanya pulsa *clock,* keadaan outputnya sama dengan keadaan inputnya. Tabel eksitasi ditunjukkan pada tabel 55. Pada tabel tersebut terlihat bahwa keadaan input kedua flip-flop yang diisikan sama dengan keadaan sekarang dari output flip-flop, hal itu karena watak flip-flop D akan memberikan output yang sama nilainya dengan input D setiap suatu pulsa *clock* diberikan.

**Tabel 55. Tabel eksitasi flip-flop D untuk tabel 54**

| INPUT | KEADAAN SEBELUMNYA | | KEADAAN SEKARANG | | KEADAAN INPUT FLIP-FLOP | |
|---|---|---|---|---|---|---|
| R | $Q_{1n-1}$ | $Q_{0n-1}$ | $Q_{1n}$ | $Q_{0n}$ | $D_1$ | $D_0$ |
| 0 | 0 | 0 | X | X | X | X |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | X | X | X | X |

Setelah diperoleh tabel eksitasi, langkah selanjutnya adalah menggambar peta Karnaugh untuk menentukan fungsi input flip-flop dan fungsi output rangkaian. Dalam hal ini telah ditentukan bahwa output rangkaian sama dengan output masing-masing flip-flop yakni $Q_{1n}$ dan $Q_{0n}$. Peta Karnaugh untuk D1, dan D0 ditunjukkan pada gambar 145.



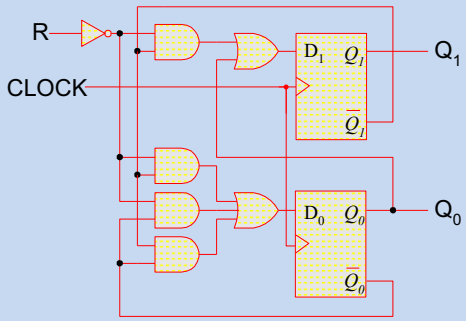

(a) (b)

**Gambar 145. Peta Karnaugh tabel 69 untuk (a) $D_1$ dan (b) $D_0$**

Dari peta Karnaugh pada gambar 145, dapat ditentukan fungsi input dari D1 dan D2 seperti ditunjukkan pada persamaan (57).

$$D_1 = \overline{R}\,\overline{Q_{1n-1}} + Q_{On-1}$$
$$D_O = \overline{R}\,\overline{Q_{1n-1}} + \overline{R}\,\overline{Q_{On-1}} + \overline{Q_{1n-1}}\,\overline{Q_{On-1}} \quad \text{persamaan (57)}$$

Dengan menggunakan persamaan (57) dapat disusun rangkaian sekuensi hasil rancangan seperti pada gambar 146.



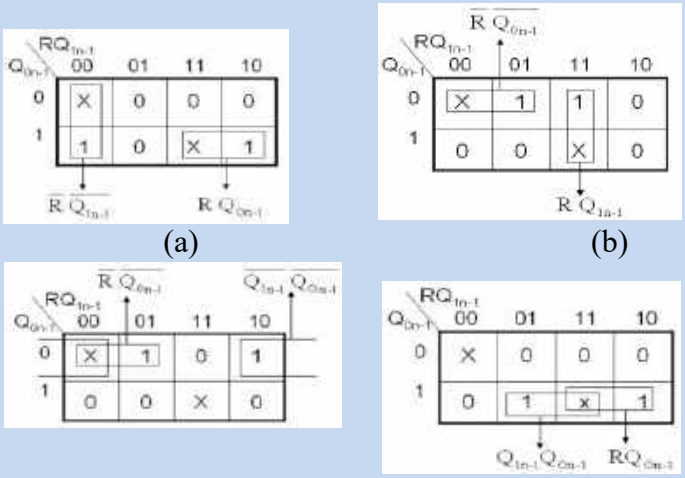

**Gambar 146. Rangkaian sekuensi hasil rancangan dengan flip-flop D**

Bagaimana rangkaiannya jika elemen penyimpannya menggunakan flip-flop J-K? Untuk menyusun rangkaian sekuensi tersebut dengan menggunakan flip-flop J-K, perlu disusun terlebih dahulu tabel eksitasi flip-flop J-K seperti pada tabel 56.

**Tabel 56. Tabel eksitasi flip-flop J-K untuk tabel 54**

| INPUT | KEADAAN SEBELUMNYA | | KEADAAN SEKARANG | | KEADAAN INPUT FLIP-FLOP | | | |
|---|---|---|---|---|---|---|---|---|
| R | $Q_{1n-1}$ | $Q_{0n-1}$ | $Q_{1n}$ | $Q_{0n}$ | $J_1$ | $K_1$ | $J_0$ | $K_0$ |
| 0 | 0 | 0 | X | X | X | X | X | X |
| 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | X | X | X | X | X | X |

Karena menggunakan flip-flop J-K, maka pengisian keadaan input flip-flop berdasarkan watak flip-flop J-K. Perhatikan baris ke-2 tabel 56! Untuk R=0 dan keadaan sebelumnya $Q_{1n-1}$=0 dan $Q_{0n-1}$=1 menyebabkan keadaan sekarang $Q_{1n}$=1 dan $Q_{0n}$=1. Keadaan tersebut memberikan arti bahwa output flip-flop 1 dalam keadaan set sehingga keadaan inputnya adalah $J_1K_1$=10 dan flip-flop 0 keadaan outputnya tetap sehingga $J_0K_0$=00. Dengan menggunakan cara seperti yang telah dijelaskan untuk baris ke-2 tersebut, dapat dihasilkan keadaan input flip-flop untuk semua keadaan yang mungkin dan hasilnya dituangkan pada tabel 56. Berdasarkan tabel 56, dapat disusun peta Karnaugh untuk $J_1$, $K_1$, $J_0$, dan $K_0$ seperti pada gambar 147.

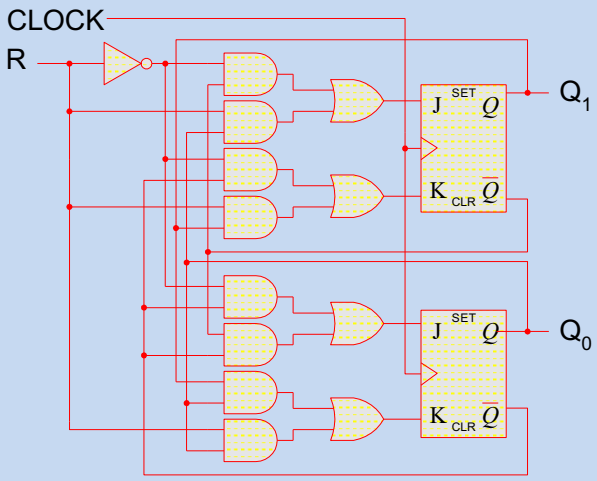



**Gambar 147. Peta Karnaugh untuk (a) $J_1$, (b) $K_1$, (c) $J_0$, dan (d) $K_0$**

Dari peta Karnaugh pada gambar 147 dapat diperoleh fungsi input kedua flip-flop, dan persamaan tersebut disajikan pada persamaan (58).

$$J_1 = \overline{R}\,\overline{Q_{1n-1}} + RQ_{On-1}$$
$$K_1 = \overline{R}\,\overline{Q_{On-1}} + RQ_{1n-1}$$
$$J_O = \overline{R}\,\overline{Q_{On-1}} + \overline{Q_{1n-1}}\,\overline{Q_{On-1}}$$
$$K_O = Q_{1n-1}Q_{On-1} + RQ_{On-1} \qquad \text{persamaan (58)}$$

Dengan menggunakan persamaan (58) dapat disusun rangkaian sekuensi hasil rancangan seperti ditunjukkan pada gambar 148.




**Gambar 148. Rangkaian sekuensi hasil rancangan dengan flip-flop J-K**

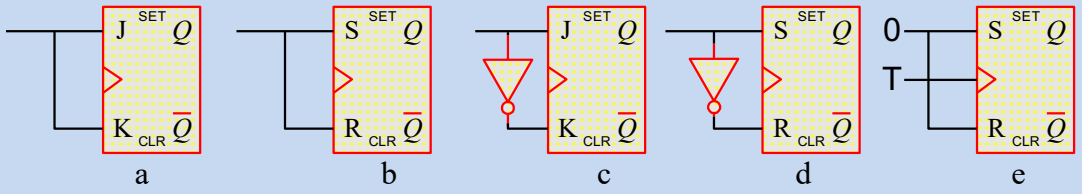
Pada contoh perancangan rangkaian sekuensi ini telah dihasilkan dua buah rangkaian sekuensi yang memberikan watak sama sesuai dengan diagram transisi keadaan yang ditetapkan. Berdasarkan contoh perancangan rangkaian logika sekuensi yang telah dikemukakan tersebut dapat diambil kesimpulan bahwa prosedur perancangan rangkaian logika sekuensi meliputi pendefinisian watak rangkaian, penuruan definisisi ke tabel keadaan, penentuan jumlah dan jenis flip-flop yang digunakan, penuruan tabel eksitasi flip-flop, penurunan fungsi input flip-flop dan output rangkaian, dan penggambaran rangkaian atas dasar persamaan yang diperoleh.

## E. Soal Latihan

Soal nomor 1 sampai dengan nomor 5 adalah jenis pilihan ganda. Kerjakan dengan cara memilih satu jawaban yang paling tepat dari opsi yang tersedia.

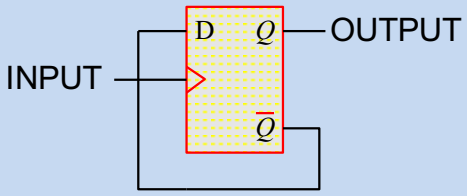
1. Definisi flip-flop yang paling tepat adalah:
   a. Flip-flop merupakan rangkaian logika yang berfungsi menyimpan data
   b. Flip-flop adalah memori 1-bit
   c. Flip-flop adalah elemen terkecil dari rangkaian sekuensial
   d. Flip-flop adalah elemen rangkaian pencacah
   e. Flip-flop adalah elemen rangkaian register
2. Flip-flop yang berfungsi membalik output yang lalu dinamakan:
   a. Flip-flop D
   b. Flip-flop SR
   c. Flip-flop JK
   d. Flip-flop SR-clocked
   e. Flip-flop T
3. Jenis flip-flop yang akan aktif jika sinyal *clock* berubah dari 0 ke 1 adalah:
   a. Edge-triggered
   b. JK Master-Slave
   *c.* SR-*Cloked*
   d. Negative edge-triggered
   e. Positive edge-triggered

4. Gambar yang benar untuk memperoleh watak flip-flop T adalah:




**Gambar 148b. Pilihan untuk soal nomor 4 Bab VI**

5. Rangkaian berikut ini memiliki watak:



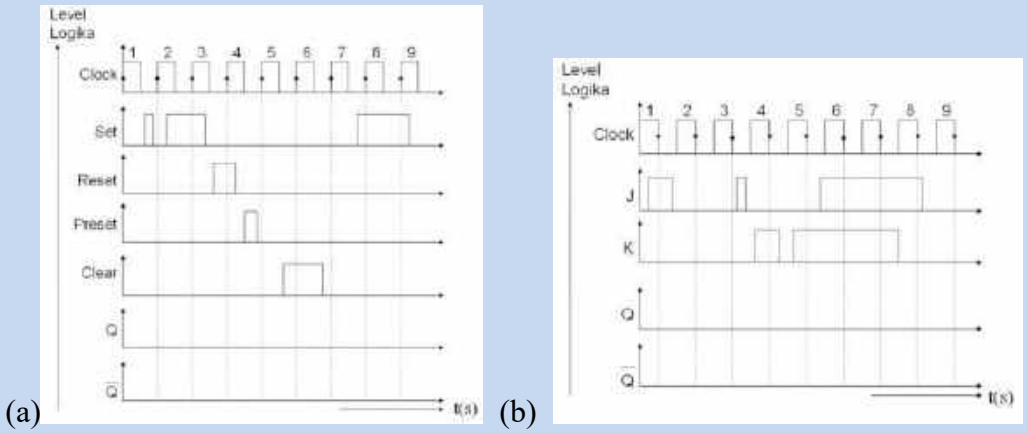

**Gambar 148c. Rangkaian untuk untuk soal nomor 5 Bab VI**

a. Mempertahankan keadaan outputnya setiap ada sinyal *clock*
b. Selalu bernilai tinggi setiap ada sinyal *clock*
c. Selalu bernilai rendah setiap ada sinyal *clock*
d. Membalik keadaan output sebelumnya setiap ada sinyal *clock*
e. Sinyal *clock* tidak berpengaruh terhadap outputnya

Soal-soal berikut ini adalah jenis uraian (esai).

6. Suatu rangkaian logika sekuensi memiliki sebuah input R satu bit dan output Y empat bit.
   a. Berapa jumlah flip-flop yang diperlukan sebagai unit penyimpan rangkaian tersebut?
   b. Jika unit penyimpannya menggunakan flip-flop D, gambarkan diagram blok rangkaian dengan menampilkan semua flip-flop yang diperlukan, dan hubungan antara jalur bit-bit keadaan dari rangkaian kombinasi ke input semua flip-flop, dan jalur output semua flip-flop ke rangkaian kombinasi.
   c. Gambarkan pula diagram blok rangkaian jika flip-flop yang digunakan jenis J-K!

7. Mengapa flip-flop S-R tidak dapat digunakan untuk input S dan R tinggi? Jelaskan jawaban anda dengan menggunakan analisis diagram waktu untuk keadaan tak stabil dan keadaan stabil!
8. Perhatikan diagram waktu pada gambar 149!

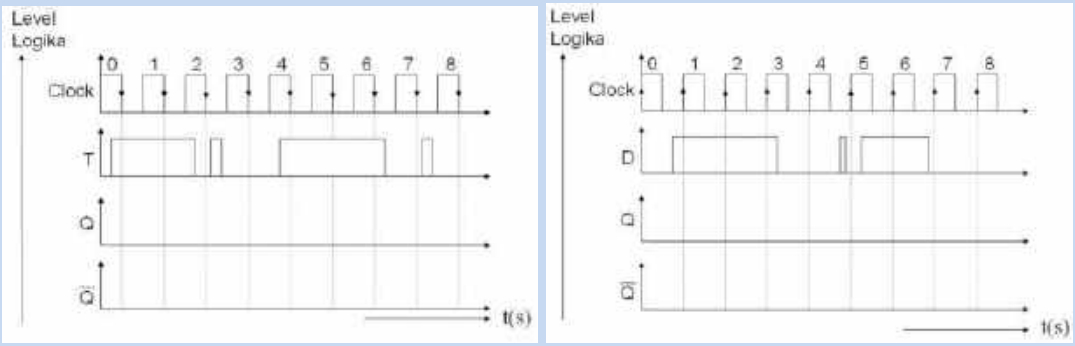



(a) (b)

**Gambar 149. Diagram waktu untuk soal nomor 8, dan 9 Bab VI**

Gambarkan bentuk output flip-flop S-R jika input-input yang diberikan bentuknya seperti pada diagram waktu gambar 149 (a)!

9. Jika input-input flip-flop bentuknya seperti pada diagram waktu gambar 149 (b), gambarkan bentuk output flip-flop J-K *positive-edge triggered*!
10. Perhatikan diagram waktu pada gambar 150!

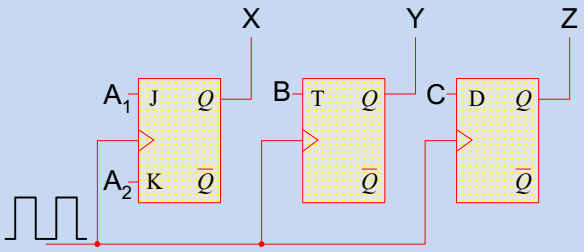



(a) (b)

**Gambar 150. Diagram waktu untuk soal nomor 10, dan 11 Bagian VI**

Gambarkan bentuk output flip-flop T jika input yang diberikan bentuknya seperti pada diagram waktu gambar 150 (a)! Gambarkan pula bentuk outputnya jika flip-flop yang digunakan dari jenis *positive-edge triggered*!

11. Gambarkan output flip-flop D jika bentuk inputnya seperti pada diagram waktu gambar 150 (b)!
12. Perhatikan rangkaian pada gambar 151!



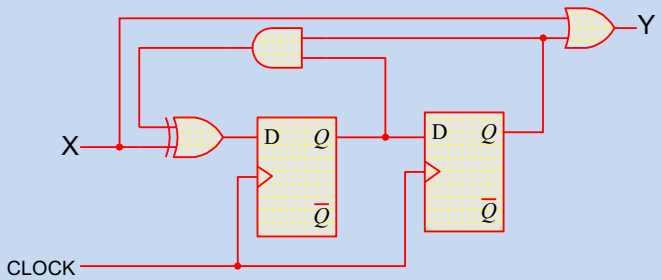

**Gambar 151. Rangkaian untuk soal nomor 12 Bab VI**

Berdasarkan rangkaian pada gambar 151, jika keadaan sebelumnya dari semua output flip-flop adalah 0, tentukan keadaan output flip-flop untuk setiap terjadinya pulsa *clock*! Gunakan tabel 57 di bawah ini!

**Tabel 57. Tabel untuk soal nomor 12 Bab VI**

| CLOCK | INPUT FLIP-FLOP | | | | OUTPUT FLIP-FLOP | | |
|---|---|---|---|---|---|---|---|
| | $A_1$ | $A_2$ | B | C | X | Y | Z |
| Ke-1 | 0 | 0 | 1 | 0 | .... | .... | .... |
| Ke-2 | 1 | 0 | 1 | 0 | .... | .... | .... |
| Ke-3 | 0 | 0 | 1 | 1 | .... | .... | .... |
| Ke-4 | 0 | 1 | 1 | 1 | .... | .... | .... |
| Ke-5 | 1 | 1 | 1 | 0 | .... | .... | .... |

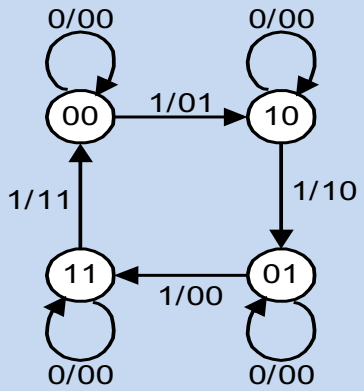
13. Perhatikan rangkaian sekuensi berikut ini!




**Gambar 152. Rangkaian sekuensi untuk soal nomor 13 Bab VI**

Lakukan analisis terhadap rangkaian pada gambar 152 untuk menentukan tabel keadaan, diagram transisi keadaan, dan persamaan keadaan serta persamaan output.

14. Rancanglah rangkaian logika sekuensi yang memiliki diagram transisi keadaan seperti pada gambar 153a menggunakan: (a) Flip-flop D, (b) Flip-flop J-K, dan (c) Flip-flop T.




**Gambar 153. Diagram transisi keadaan untuk soal nomor 14 Bab VI**

## KOMPETENSI DASAR VII

Mahasiswa memahami watak dan cara kerja modul-modul logika sekuensial mencakup rangkaian pencacah dan register

## TUJUAN PEMBELAJARAN VII

Agar mahasiswa dapat:
1. mendefinisikan pengertian pencacah dan register
2. menjelaskan cara kerja pencacah dan register
3. merancang pencacah sinkron dan asinkron berbagai modulo menggunakan flip-flop JK, D dan T
4. menggambarkan rangkaian internal IC pencacah 7493 dan susunan pin yang tersedia
5. merancang pencacah asinkron berbagai modulo menggunakan IC 7493
6. merancang register paralel dan geser menggunakan flip-flop maupun IC register
7. menjelaskan berbagai jenis transfer register

## GARIS BESAR MATERI VII

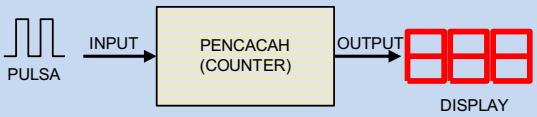
Pencacah dan register merupakan rangkaian sekuensi yang aplikasinya sangat luas dalam sistem digital. Pencacah atau *counter* merupakan rangkaian logika sekuensi yang berfungsi mencacah atau menghitung jumlah pulsa *clock* yang masuk. Menurut jumlah pulsa yang dapat dicacah, terdapat jenis pencacah modulo-X dengan X menunjukkan jumlah pulsa yang dapat dicacah. Pencacah seperti pencacah modulo-5, outputnya akan *reset* pada saat terjadinya *clock* ke-5. Setiap jenis pencacah tersebut dapat berupa pencacah tak serempak maupun pencacah serempak. Selain itu, terdapat pula jenis pencacah naik-turun (*up-down counter*), pencacah ring dan pencacah Johnson. Melalui bagian ini Anda akan diperkenalkan dengan analisis dan perancangan berbagai jenis pencacah tersebut. Untuk menambah pengetahuan praktis Anda, akan diperkenalkan juga berbagai seri IC yang menyediakan fungsi pencacah baik jenis tak serempak maupun serempak. Uraian bagian pencacah diakhiri dengan memperkenalkan kepada Anda aplikasi dari pencacah yakni rangkaian pencacah frekuensi dan jam digital. Penjelasan terhadap kedua rangkaian tersebut mencakup cara kerja dan perancangannya, sehingga setelah selesai mempelajari bagian ini diharapkan Anda dapat menyusun sendiri kedua rangkaian itu.

Bagian kedua bab ini akan memperkenalkan kepada Anda tentang register. Jika flip-flop merupakan elemen penyimpan satu bit, maka register merupakan kumpulan flip-flop yang dapat menyimpan data beberapa bit di dalamnya. Secara umum register terbagi menjadi dua yakni register paralel dan register geser. Jenis pertama merupakan register yang cara penyimpanan datanya dilakukan secara serempak, dan jenis kedua merupakan register yang cara penyimpanan datanya secara berurutan bit demi bit dan dilakukan dengan menggeser bit-bit yang ada di dalam elemen register. Berdasarkan cara memasukkan data ke dalam inputnya untuk disimpan dan cara mengeluarkan data melalui outputnya, register dapat dibedakan menjadi register dengan input dan output paralel *(parallel in-parallel out*) disingkat PIPO, register dengan input paralel dan output seri (*parallel in-serial out*) disingkat PISO, register dengan input seri dan output seri (*serial in-serial out*) disingkat SISO, register dengan input seri dan output paralel (*serial in-parallel out*) disingkat SIPO. Melalui bagian ini Anda akan diperkenalkan dengan analisis dan perancangan berbagai jenis register tersebut. Akan diperkenalkan pula beberapa IC yang menyediakan fungsi register paralel maupun geser yang dapat digunakan untuk menyediakan fungsi register PIPO, PISO, SISO, maupun PISO. Bagian akhir dari bab ini akan memperkenalkan kepada Anda tranfer data yang banyak digunakan di dalam aplikasi rangkaian digital yakni transfer paralel dan seri. Untuk menyelenggarakan transfer paralel diperlukan dua buah register yang dapat berperan sebagai register paralel. Sedangkan untuk seri, implementasinya memerlukan dua buah register yang dapat berperan sebagai register geser.

# BAB VII
# PENCACAH DAN REGISTER

## A. Pencacah

Pencacah atau *counter* merupakan rangkaian logika sekuensi yang berfungsi mencacah atau menghitung jumlah pulsa *clock* yang masuk. Menurut jumlah pulsa yang dapat dicacah, terdapat jenis pencacah modulo-$2^n$ (n=1, 2, 3, 4, …), contoh pencacah modulo-4, pencacah modulo-8, dan pencacah modulo-16. Jika clock ke-0 dinyatakan sebagai keadaan awal pencacah, jumlah pulsa yang dapat dicacah oleh pencacah modulo-4 adalah 4 buah yakni pulsa ke-0, ke-1, ke-2, ke-3, dan pada pulsa *clock* ke-4, output pencacah ini akan *reset* kembali ke 0. Pada pencacah modulo-8, output akan reset pada *clock* ke-8 sehingga pencacah ini hanya mampu mencacah pulsa *clock* ke-0 sampai dengan pulsa *clock* ke-7. Selain pencacah modulo-$2^n$ terdapat pula pencacah seperti modulo-5, modulo-6, dan modulo-10. Diagram blok pencacah ditunjukkan gambar 154.




**Gambar 154. Diagram blok pencacah**

Sedangkan menurut pengaktifan elemen penyimpanannya dan dalam hal ini elemen penyimpan pencacah adalah flip-flop, terdapat pencacah jenis tak serempak atau pencacah tak sinkron (*asynchronous counter*), dan pencacah jenis serempak atau pencacah sinkron (*synchronous counte*). Pada pencacah tak serempak, elemen-elemen penyusunnya yakni flip-flop bekerja secara tidak serempak ketika pencacah tersebut diberi input pulsa, dan pada pencacah serempak elemen-elemen penyusunnya bekerja secara bersama-sama ketika ada pulsa masuk ke inputnya.

Prosedur perancangan kedua jenis pencacah tersebut agak berbeda. Untuk pencacah serempak prosedur perancangannya sama dengan prosedur perancangan rangkaian sekuensial seperti telah dijelaskan di muka. Sedangkan untuk rangkaian pencacah tak serempak prosedur perancangannya lebih sederhana dan akan dijelaskan terlebih dahulu.

## 1. Pencacah Tak Serempak

Untuk merancang pencacah tak serempak, perlu ditetapkan terlebih dahulu modulo dari pencacah yang akan dirancang. Untuk modulo-$2^n$, prosedur perancangannya mengikuti urutan sebagai berikut:

a. Tetapkan modulo pencacah yang akan dirancang, misalnya akan dirancang pencacah tak serempak modulo-8 atau modulo-$2^n$ dengan n=3.

b. Tentukan jumlah dan jenis flip-flop yang akan digunakan. Jumlah flip-flop yang digunakan adalah n buah. Jika akan digunakan flip-flop J-K, maka sediakan flip-flop J-K sebanyak n buah, dalam hal ini 3 buah.

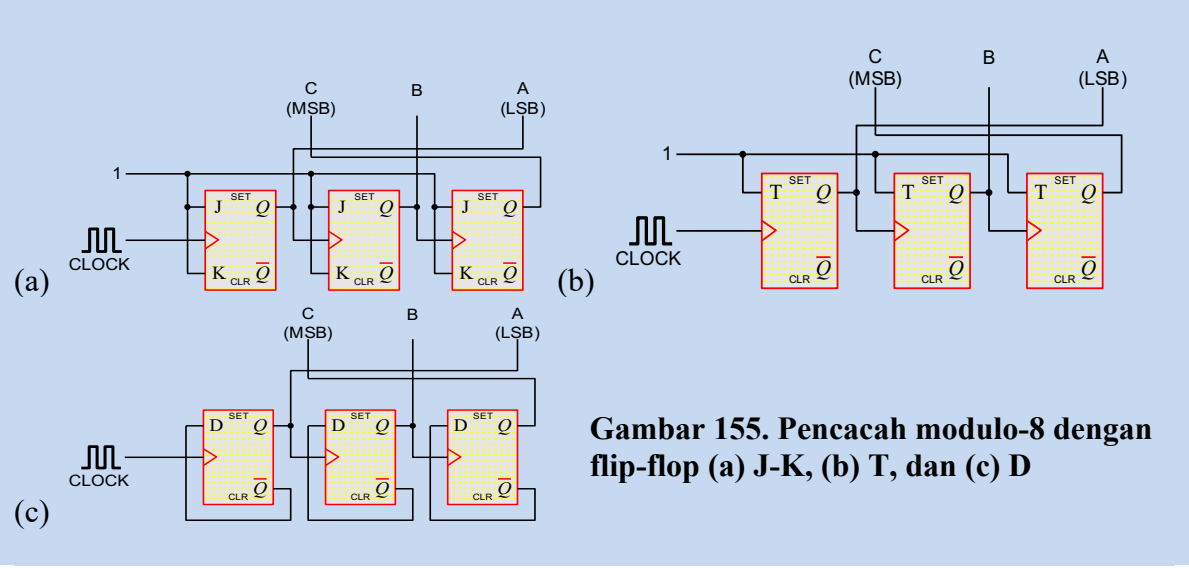
c. Lakukan pengaturan input-input flip-flop yang digunakan. Untuk flip-flop J-K, hubungkan semua input J dan input K dengan level logika 1. Untuk flip-flop T, hubungkan semua input T dengan level logika 1, dan untuk flip-flop D, hubungkan tiap input D dengan komplemen outputnya.

d. Berikan input pencacah ke input *clock* flip-flop paling kiri.

e. Hubungkan output flip-flop paling kiri dengan input *clock* flip-flop di sebelah kanannya dan seterusnya.

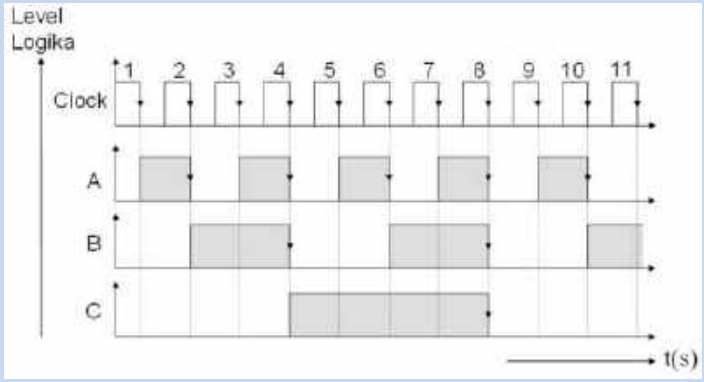
f. Ambil output pencacah melalui setiap output flip-flop. Ingat: output flip-flop paling kiri adalah LSB dan ouput flop paling kanan adalah MSB.

Melalui prosedur tersebut, dapat disusun rangkaian pencacah tak serempak modulo-8 dengan flip-flop J-K, flip-flop T maupun flip-flop D seperti ditunjukkan pada gambar 155.




**Gambar 155. Pencacah modulo-8 dengan flip-flop (a) J-K, (b) T, dan (c) D**

Cara kerja pencacah tak sinkron modulo-8 ini dapat dijelaskan sebagai berikut. Perhatikan gambar 155 (a)! Semua input flip-flop J-K dihubungkan dengan logika 1, hal itu berarti bahwa setiap ada pulsa *clock* yang masuk, output flip-flop itu akan berubah.
Anggap ketiga flip-flop J-K tersebut dari jenis *negative-edge triggered*, dan memilki output dari kiri ke kanan A, B, dan C. Agar cara kerja pencacah dapat dipelajari dengan mudah perlu digunakan diagram waktu. Perhatikan diagram waktu pencacah modulo-8 pada gambar 156!




**Gambar 156. Diagram waktu pencacah tak serempak modulo-8**

Anggap pada saat awal output pencacah CBA=000. Pada *clock* ke-1 setelah terjadinya tepi turun, flip-flop A terpicu sehingga A=1, dan flip-flop yang lain belum terpicu sehingga outputnya masih bernilai 0. Pada keadaan ini output pencacah adalah CBA=001. Pada *clock* ke-2 setelah terjadinya tepi turun, output flip-flop A akan membalik output sebelumnya sehingga menjadi rendah, dan perubahan ini akan memicu flip-flop B sehingga B=1, pada sisi lain output C tetap rendah karena belum terpicu. Untuk keadaan ini output pencacah menjadi CBA=010. Selanjutnya pada saat terjadinya tepi turun *clock* ke-3 flip-flop A terpicu sehingga outputnya berubah dari rendah ke tinggi menjadi A=1, flip-flop B tidak terpicu karena output A sebagai pemicunya berubah dari rendah ke tinggi sehingga B tetap tinggi. Demikian pula dengan flip-flop C, karena belum terpicu outputnya masih rendah sehingga untuk keadaan ini output pencacah menjadi CBA=011. Pada tepi turun *clock* ke-4, flip-flop A terpicu sehingga outputnya terbalik menjadi A=0, dan perubahan ini memicu flip-flop B sehingga output B juga terbalik menjadi B=0. Karena B berubah dari tinggi ke rendah maka outputnya memicu flip-flop C sehingga C berubah menjadi C=1. Pada keadaan ini output pencacah menjadi CBA=100. Pada tepi turun pulsa *clock* ke-5, flip-flop A terpicu sehingga outputnya berubah dari A=0 menjadi A=1, flip-flop B dan C tidak terpicu sehingga outputnya tetap B=0 dan C=1. Untuk

keadaan ini output pencacah adalah CBA=101. Pada tepi turun pulsa *clock* ke-6, flip-flop A terpicu sehingga A=0, output flip-flop A memicu flip-flop B sehingga outputnya berubah menjadi B=1, dan flip-flop C tidak terpicu sehingga C=1.

Pada keadaan ini output pencacah menjadi CBA=110. Selanjutnya, pada tepi turun pulsa *clock* ke-7, flip-flop A terpicu sehingga outputnya A=1, B tetap, C tetap, dan output pencacah menjadi CBA=111. Pada saat terjadinya pulsa turun *clock* ke-8, ketiga flip-flop terpicu, dan karena keadaan output awalnya tinggi maka akan berubah menjadi reset. Keadaan tersebut menyebabkan output pencacah menjadi CBA=000. Jika keadaan-keadaan tersebut dituangkan dalam suatu tabel, maka akan diperoleh tabel kebenaran pencacah tak sinkron modolo-8 seperti disajikan pada tabel 58.

**Tabel 58. Tabel kebenaran pencacah tak serempak modulo-8**

| CACAH (COUNT) | OUTPUT | | |
|---|---|---|---|
| | C | B | A |
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 2 | 0 | 1 | 0 |
| 3 | 0 | 1 | 1 |
| 4 | 1 | 0 | 0 |
| 5 | 1 | 0 | 1 |
| 6 | 1 | 1 | 0 |
| 7 | 1 | 1 | 1 |
| 8 | 0 | 0 | 0 |

Berdasarkan diagram waktu pada gambar 156, untuk pencacah tak serempak modulo-8 jika pada inputnya dimasukkan pulsa *clock* dengan periode T, maka pada output B memberikan pulsa dengan periode 2T, dan pada output C memberikan pulsa dengan periode 4T. Dengan kata lain jika output flip-flop diambil pada C maka rangkaian tersebut berfungsi sebagai pembagi frekuensi, dalam hal ini frekuensi pulsa pada output C nilainya seperempat frekuensi *clock*. Pencacah modulo-$2^n$ yang lain dapat dirancang menggunakan cara yang sama dengan cara perancangan pencacah modulo-8.

Perancangan yang telah dilakukan di muka adalah untuk pencacah tak serempak modulo-$2^n$. Untuk pencacah selain modulo tersebut seperti pencacah modulo-5, modulo-7, modulo-9, dan modulo 14, perancangannya mengikuti prosedur sebagai berikut:

a. Tetapkan modulo pencacah yang akan dirancang, misalnya pencacah tak serempak modulo-5.

b. Tentukan jumlah dan jenis flip-flop yang akan digunakan. Pencacah modulo-5 harus menampilkan angka terbesar pada outputnya 4 desimal atau 100 biner, sehingga diperlukan penampil 3 bit. Untuk menampilkan data 3-bit diperlukan 3 buah flip-flop. Seperti pada perancangan pencacah modulo $2^n$, flip-flop yang digunakan dapat dari jenis J-K, T maupun D. Untuk pencacah tak serempak modulo yang lain, penentuan jumlah flip-flop yang diperlukan juga dilakukan dengan cara yang sama, misal pencacah tak serempak modulo-9 memerlukan 4 buah flip-flop karena pencacah ini akan menampilkan angka maksimum sebesar 8 dan dalam biner angka tersebut ditampilkan dalam format 4-bit yakni 1000.

c. Lakukan pengaturan input-input flip-flop yang digunakan. Untuk flip-flop J-K, hubungkan semua input J dan input K dengan level logika 1. Untuk flip-flop T, hubungkan semua input T dengan level logika 1, dan untuk flip-flop D, hubungkan tiap input D dengan komplemen outputnya.

d. Berikan input pencacah ke input *clock* flip-flop paling kiri.

e. Hubungkan output flip-flop paling kiri dengan input *clock* flip-flop di sebelah kanannya dan seterusnya.

f. Ambil output pencacah melalui setiap output flip-flop. Ingat: output flip-flop paling kiri adalah LSB dan ouput flip-flop paling kanan adalah MSB.

g. Susun tabel kebenaran pencacah tak serempak yang sedang dirancang, dalam hal ini pencacah tak serempak modulo-5.

**Tabel 59. Tabel kebenaran pencacah tak serempak modulo-5**

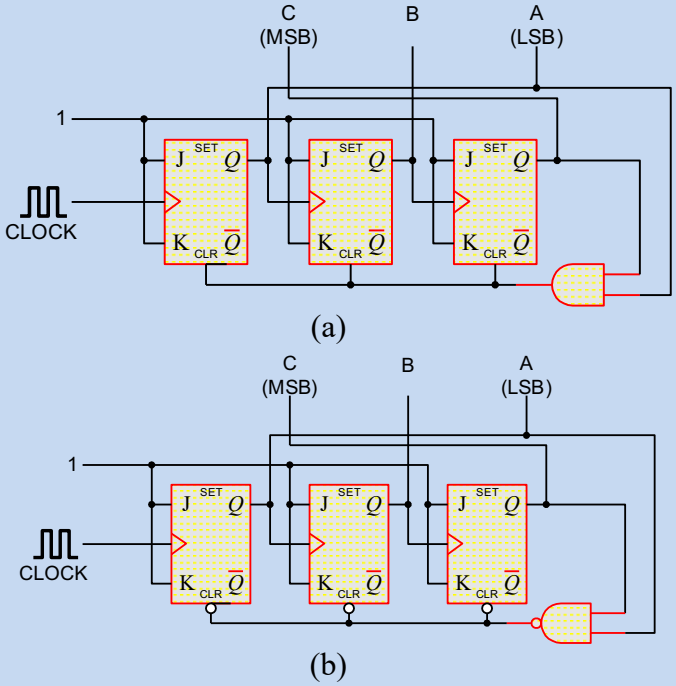
| CACAH (COUNT) | OUTPUT | | |
|---|---|---|---|
| | C | B | A |
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 2 | 0 | 1 | 0 |
| 3 | 0 | 1 | 1 |
| 4 | 1 | 0 | 0 |
| 5 | 0 | 0 | 0 |

h. Dari tabel kebenaran terlihat bahwa ketika pencacah memberikan output bernilai 5 desimal atau CBA=101 biner, karena harus *reset* yakni semua outputnya 0 maka pada c*lock*-5 output C dan A keduanya harus berubah dari 1 ke 0. Perubahan tersebut dapat dilakukan dengan menghubungkan *clear* setiap flip-flop dengan suatu gerbang. Gerbang harus dapat membangkitkan sinyal yang diperlukan untuk *clear* jika inputnya C=1 dan A=1.

i. Jika jenis *clear* pada flip-flop adalah *active-low* (*clear* jika diberi 0), maka gerbang yang digunakan harus dapat membangkitkan sinyal 0 jika kedua inputnya 1 yakni berasal dari C dan A. Untuk keadaan ini gerbang yang digunakan untuk melakukan *clear* adalah NAND.

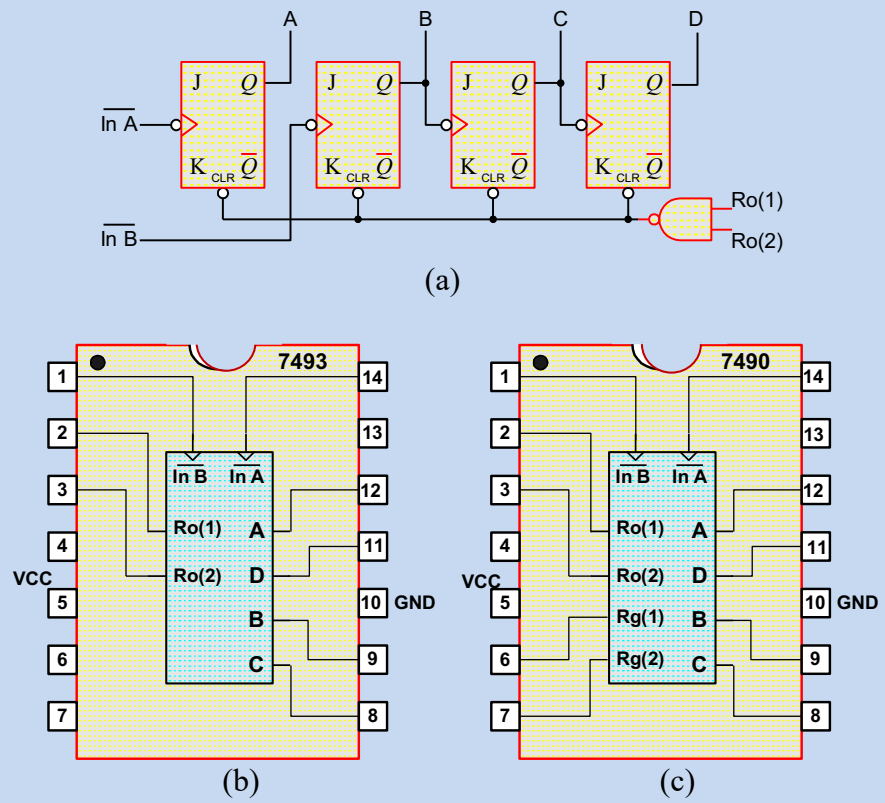
j. Jika jenis *clear* pada flip-flop adalah *active-high* (*clear* jika diberi 1), maka gerbang yang digunakan harus dapat membangkitkan sinyal 1 jika kedua inputnya 1 yakni berasal dari C dan A. Untuk keadaan ini gerbang yang digunakan untuk melakukan *clear* adalah AND.

k. Angggap flip-flop yang digunakan memiliki *clear* jenis *active-high*. Hubungkan output yang akan dinolkan (reset), dalam hal ini output C dan A ke input gerbang AND, dan output gerbang AND tersebut dihubungkan ke *clear* semua flip-flop. Rangkaian pencacah tak serempak modulo-5 ditunjukkan pada gambar 157.



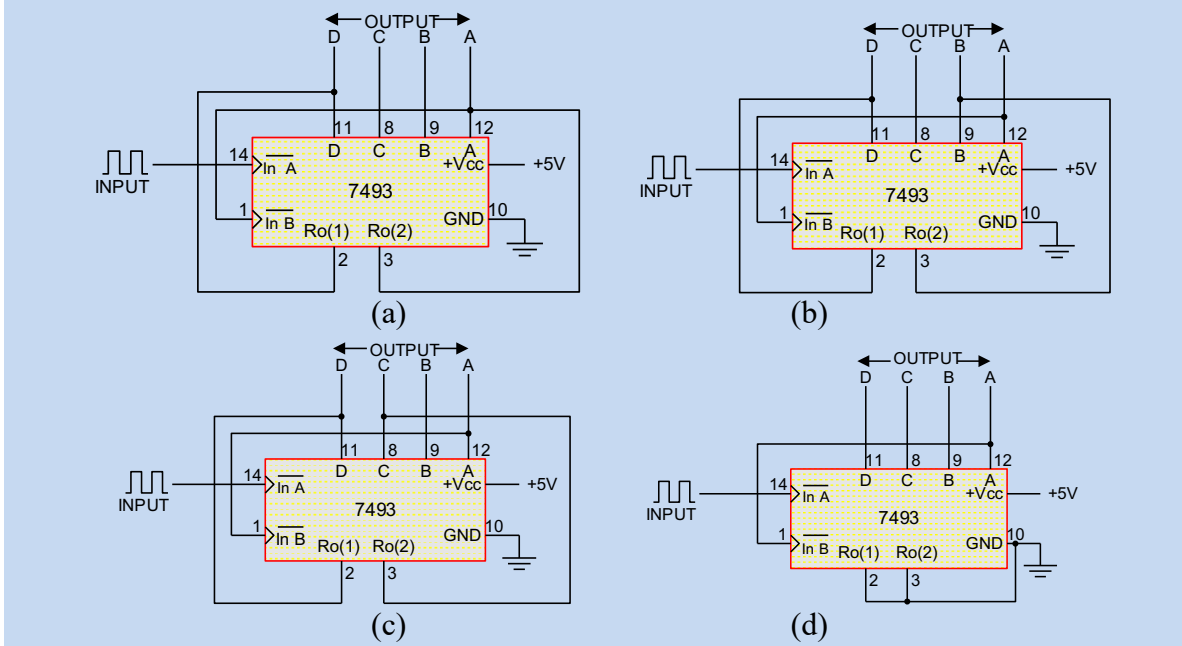

**Gambar 157. Rangkaian pencacah modulo-5 dengan flip-flop J-K: (a) *clear* jenis *active-high*, dan (b) *clear* jenis *active-low***

Dalam praktek, fungsi pencacah tak serempak disediakan oleh beberapa IC di antaranya adalah seri 7490 sebagai pencacah modulo-10 atau pencacah *decade* atau pembagi 10, 7492 sebagai pencacah pembagi 12, dan 7493 sebagai pencacah biner 4-bit. Rangkaian internal IC 7493 ditunjukkan pada gambar 158 (a).Spesisikasi pin untuk IC itu ditunjukkan pada gambar 158 (b) dan untuk IC 7490 ditunjukkan pada gambar 158 (c).



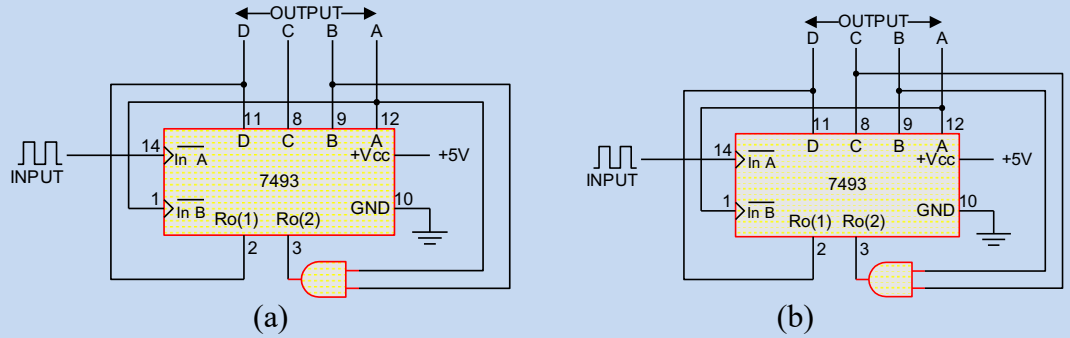

**Gambar 158. IC pencacah tak sinkron: (a) rangkaian internal IC 7493, (b) spesifikasi pin IC 7493, dan (c) spesifikasi pin IC 7490**

Dengan menggunakan IC 7493 dapat disusun rangkaian pencacah sampai dengan modulo-16. Gambar 159 menunjukkan rangkaian pencacah tak serempak berbagai modulo menggunakan IC 7493.



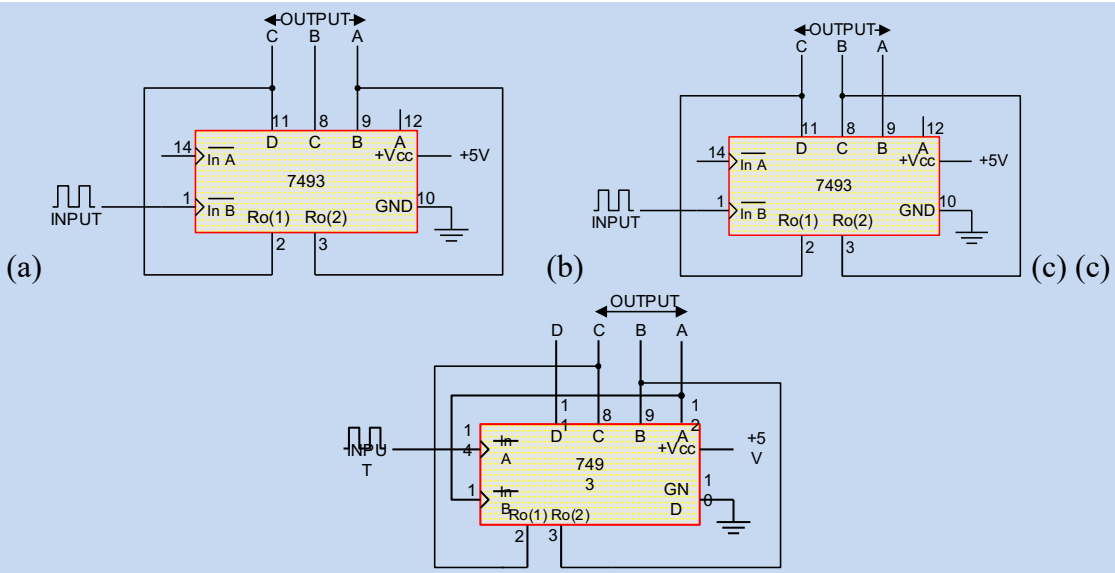

**Gambar 159. IC 7493 sebagai pencacah tak serempak: (a) modulo-9, (b) modulo-10, (c) modulo-12, dan (d) modulo-16**

Karena gerbang NAND sebagai fasilitas *clear* yang ada di dalam rangkaian internal 7493 hanya memiliki dua buah input, maka untuk pencacah yang memerlukan *clear* terhadap 3 buah outputnya seperti pencacah modulo-11, modulo-13, modulo-14, dan modulo-15 diperlukan rangkaian luar atau rangkaian tambahan. Gambar 160 menunjukkan rangkaian tambahan yang diperlukan untuk membangun pencacah serempak modulo-11, dan modulo-14.




**Gambar 160. IC 7493 sebagai pencacah tak serempak: (a) modulo-11, dan (b) modulo-14**

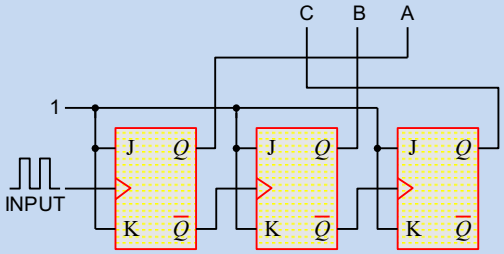
Bagaimana memanfaatkan IC 7493 sebagai pencacah tak sinkron modulo-8 ke bawah? Untuk pencacah modulo-5 dan modulo-6, dapat memanfaatkan 3 flip-flop yang tersedia dengan input pencacah diumpankan ke $\overline{\text{In B}}$ dan output diambil dari D, C, B pada IC 7493 sebagai output pencacah C, B, A seperti ditunjukkan pada gambar 161 (a) untuk pencacah modulo-5 dan gambar 161 (b) untuk pencacah modulo-6. Selain itu, jika diinginkan $\overline{\text{In A}}$ sebagai input, maka untuk pencacah modulo-6 dapat pula menggunakan rangkaian pada gambar 161 (c).




**Gambar 161. IC 7493 sebagai pencacah tak serempak: modulo-5, (b) dan (c) modulo-6**

Pencacah tak serempak yang telah dibahas di muka adalah pencacah naik (*up counter*), yakni outputnya memberikan urutan naik. Kecuali dapat memberikan urutan naik, suatu pencacah juga dapat memberikan urutan turun pada outputnya, dan pencacah seperti ini dinamakan pencacah turun (*down counter*).

Perbedaan rancangan pencacah turun dengan pencacah naik untuk jenis tak serempak terletak pada input *clock* dari flip-flop tahap berikutnya setelah flip-flop pertama. Flip-flop pertama adalah flip-flop paling kiri yang menerima input pulsa *clock*. Pada pencacah naik, input *clock* flip-flop tahap berikutnya berasal dari output flip-flop sebelumnya (Q), sedangkan pada pencacah turun berasal dari komplemen output flip-flop sebelumnya ($\overline{Q}$). Rangkaian pencacah turun modulo-8 jenis tak serempak ditunjukkan pada gambar 162.




**Gambar 162. Pencacah turun tak serempak modulo-8**

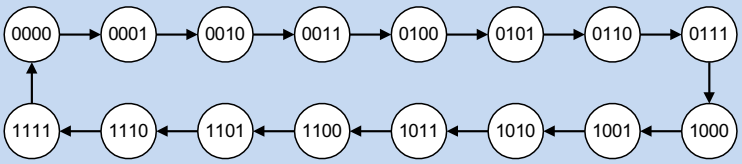
## 2. Pencacah Serempak

Pencacah serempak merupakan rangkaian sekuensial serempak, sehingga perancangannya dilakukan dengan menggunakan prosedur seperti pada perancangan rangkaian sekuensial serempak.

**Contoh 1:** Rancang pencacah serempak modulo-16!

**Jawab:**

Langkah pertama dalam merancang rangkaian ini adalah menyusun diagram transisi keadaan berdasarkan definisi watak. Pencacah serempak modulo-16 adalah pencacah yang keadaan outputnya akan *reset* pada *clock* ke-16. Dari definisi watak tersebut dapat dilukis diagram transisi keadaan seperti ditunjukkan pada gambar 163.




**Gambar 163. Diagram transisi keadaan pencacah modulo-16**

Karena unit rangkaian kombinasi pada rangkaian pencacah serempak tidak memiliki input, dan output rangkaian diambil dari output unit penyimpannya maka diagram transisinya hanya menggambarkan keadaan transisi dari output elemen-elemen penyimpannya saja. Langkah selanjutnya adalah menyusun tabel keadaan berdasarkan diagram transisi keadaan yang diperoleh. Dari gambar 163, dapat disusun tabel keadaan dari rangkaian pencacah serempak modolo-16 seperti pada tabel 60.

**Tabel 60. Tabel keadaan rangkaian pencacah serempak modulo-16**

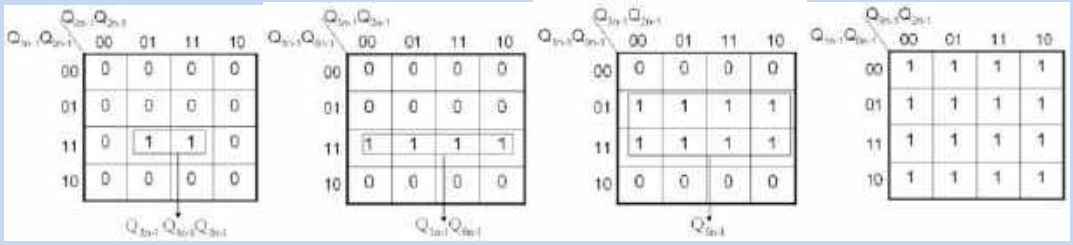
| KEADAAN SEBELUMNYA | | | | KEADAAN SEKARANG | | | |
|---|---|---|---|---|---|---|---|
| $Q_{3(n-1)}$ | $Q_{2(n-1)}$ | $Q_{1(n-1)}$ | $Q_{0(n-1)}$ | $Q_{3(n)}$ | $Q_{2(n)}$ | $Q_{1(n)}$ | $Q_{0(n)}$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

Berdasarkan tabel 60, dapat disusun tabel eksitasi flip-flop yang digunakan. Misal akan digunakan flip-flop T, maka tabel eksitasinya dapat disusun seperti pada tabel 61.

**Tabel 61. Tabel eksitasi flip-flop T untuk tabel 60**

| KEADAAN SEBELUMNYA | | | | KEADAAN SEKARANG | | | | KEADAAN INPUT FLIP-FLOP | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $Q_{3(n-1)}$ | $Q_{2(n-1)}$ | $Q_{1(n-1)}$ | $Q_{0(n-1)}$ | $Q_{3(n)}$ | $Q_{2(n)}$ | $Q_{1(n)}$ | $Q_{0(n)}$ | $T_3$ | $T_2$ | $T_1$ | $T_0$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

Selanjutnya, berdasarkan tabel eksitasi tersebut disusun peta Karnaugh untuk menentukan fungsi dari masing-masing input pada flip-flop yang digunakan.



(a) (b) (c) (d)

**Gambar 164. Peta Karnaugh tabel 75 untuk: (a) $T_3$, (b) $T_2$, (c) $T_1$, dan (d) $T_0$**

Dari peta Karnaugh tersebut dapat diturunkan fungsi input setiap flip-flop seperti ditunjukkan pada persamaan (59).

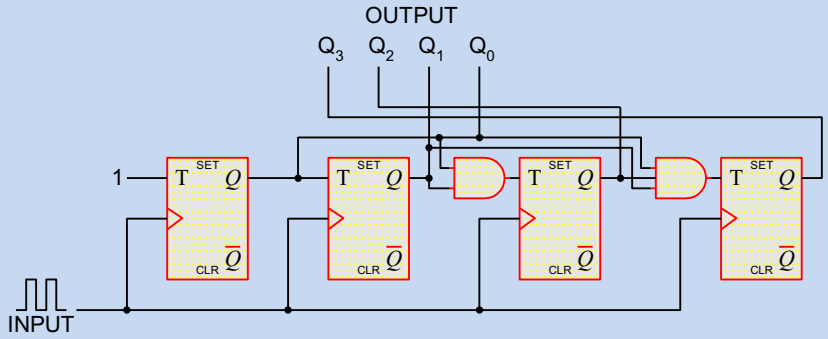
$$T_3 = Q_{2n-1}Q_{1n-1}Q_{0n-1}$$

$$T_2 = Q_{1n-1}Q_{On-1}$$

$$T_1 = Q_{On-1}$$

$$T_0 = 1 \qquad \text{persamaan (59)}$$

Berdasarkan persamaan (59) dapat disusun rangkaian pencacah serempak modulo-16 seperti ditunjukkan pada gambar 165.

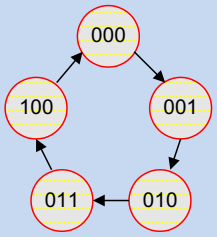



**Gambar 165. Rangkaian pencacah serempak modulo-16 dengan flip-flop T**

**Contoh 2:** Rancang pencacah serempak modulo-5!

**Jawab:**

Pencacah modulo-5 adalah pencacah yang outputnya *reset* pada pulsa *clock* ke-5. Dari definisi tersebut dapat disusun diagram transisi keadaan seperti ditunjukkan pada gambar 166.




**Gambar 166. Diagram transisi keadaan pencacah modulo-5**

Dari gambar 166 dapat diturunkan tabel keadaan pencacah modulo-5 seperti ditunjukkan pada tabel 62.

**Tabel 62. Tabel keadaan rangkaian pencacah serempak modulo-5**

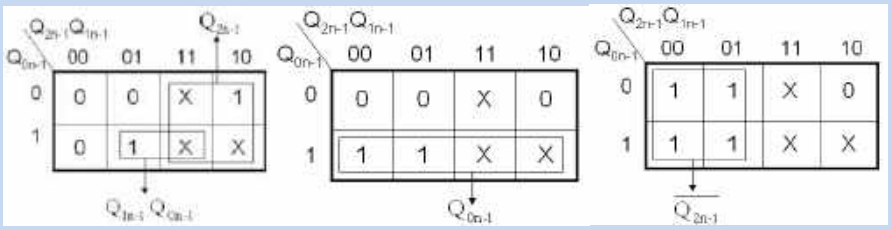
| KEADAAN SEBELUMNYA | | | KEADAAN SEKARANG | | |
|---|---|---|---|---|---|
| $Q_{2(n-1)}$ | $Q_{1(n-1)}$ | $Q_{0(n-1)}$ | $Q_{2(n)}$ | $Q_{1(n)}$ | $Q_{0(n)}$ |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | X | X | X |
| 1 | 1 | 0 | X | X | X |
| 1 | 1 | 1 | X | X | X |

Karena tidak terdapat keadaan transisi untuk keadaan sebelumnya yang bernilai 101, 110, dan 111, maka untuk keadaan-keadaan tersebut output sekarang diberi tanda X. Selanjutnya, dari tabel keadaan dapat disusun tabel eksitasi flip-flop yang digunakan. Untuk flip-flop T, tabel eksitasinya ditunjukkan pada tabel 63.

**Tabel 63. Tabel eksitasi flip-flop T untuk tabel 62**

| KEADAAN SEBELUMNYA | | | KEADAAN SEKARANG | | | KEADAAN INPUT FLIP-FLOP | | |
|---|---|---|---|---|---|---|---|---|
| $Q_{2n-1}$ | $Q_{1n-1}$ | $Q_{0n-1}$ | $Q_{2n}$ | $Q_{1n}$ | $Q_{0n}$ | $T_2$ | $T_1$ | $T_0$ |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | X | X | X | X | X | X |
| 1 | 1 | 0 | X | X | X | X | X | X |
| 1 | 1 | 1 | X | X | X | X | X | X |

Peta Karnaugh untuk tabel 63 ditunjukkan pada gambar 167.




(a) (b) (c)

**Gambar 167. Peta Karnaugh tabel 62 untuk (a) $T_2$, (b) $T_1$, dan (c) $T_0$**

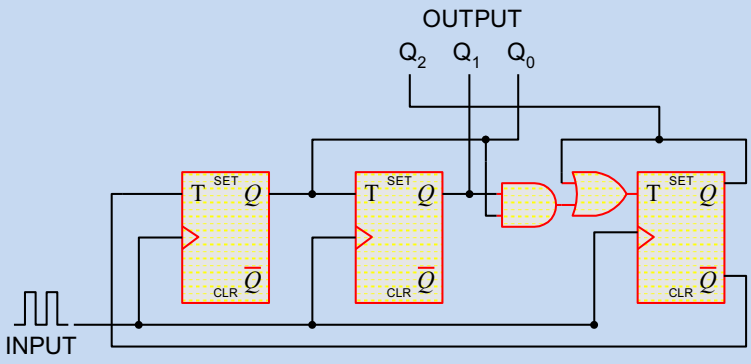
Sedangkan persamaan yang diperoleh dari peta Karnaugh pada gambar 254 dituliskan pada persamaan (60).

$$T_2 = Q_{1n-1}Q_{On-1} + Q_{2n-1}$$

$$T_1 = Q_{On-1}$$

$$T_0 = \overline{Q_{2n-1}}$$ persamaan (60)

Dari persamaan (60) dapat disusun rangkaian pencacah serempak modulo-5 dengan flip-flop T seperti ditunjukkan pada gambar 168.

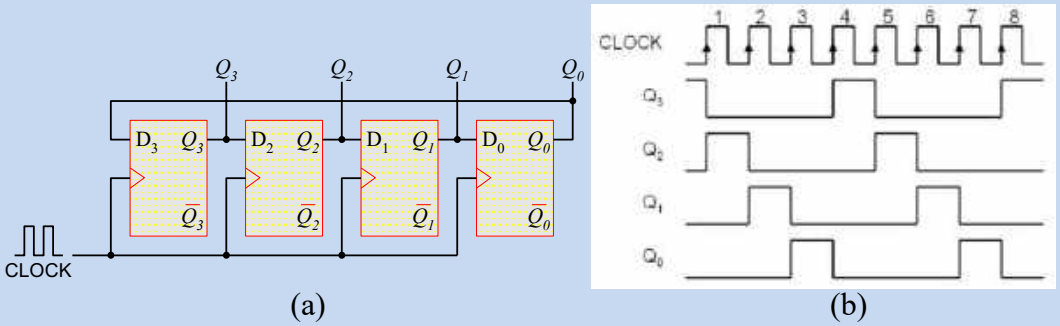



**Gambar 168. Rangkaian pencacah serempak modulo-5**

### 3. Pencacah Ring dan Pencacah Johnson

Dalam banyak aplikasi sering diperlukan suatu pencacah yang dapat membangkitkan pulsa secara berurutan pada outputnya. Misalnya pada *clock* ke-0 output $Q_3$ bernilai tinggi dan output lainnya rendah. Pada *clock* ke-1 output $Q_2$ tinggi lainnya rendah dan seterusnya. Pencacah yang memiliki watak dapat membangkitkan pulsa secara berurutan dinamakan pencacah ring. Pencacah ini dapat dibangun dengan menggunakan flip-flop D, dan

rangkaiannya ditunjukkan pada gambar 169 (a), sedangkan gambar 169 (b) dan tabel 64 menunjukkan diagram waktu dan tabel keadaannya.




**Gambar 169. Pencacah ring: (a) rangkaian, dan (b) diagram waktu**

Anggap mula-mula $Q_3$ tinggi dan output lainnya rendah. Pada *clock* ke-1, $Q_2$ menjadi tinggi karena input flip-flop $D_2$ sama dengan $Q_3$ yakni tinggi, sedangkan output lainnya rendah. Output $Q_3$ menjadi rendah karena input flip-flop $D_3$ sama dengan output $Q_0$ yakni rendah. Pada *clock* ke-2, $Q_1$ menjadi tinggi karena input flip-flop $D_1$ sama dengan output $Q_2$ yakni tinggi, dan output yang lain rendah. Pada *clock* ke-3, output $Q_0$ menjadi tinggi karena input flip-flop $D_0$ sama dengan output $Q_1$ yakni tinggi dan output lainnya rendah. Pada *clock* ke-4 kembali $Q_3$ menjadi tinggi karena input $D_3$ sama dengan output $D_0$ yakni tinggi, dan output lainnya rendah. Karena urutan outputnya kembali lagi pada *clock* ke-4, maka pencacah ring seperti ini termasuk pencacah modulo-4. Keadaan-keadaan tersebut dapat dituangkan dalam bentuk tabel keadaan seperti ditunjukkan pada tabel 64.

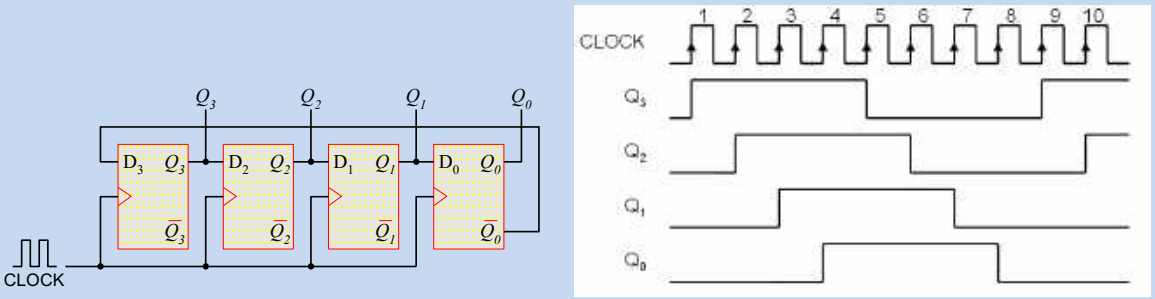
**Tabel 64. Tabel keadaan pencacah ring modulo-4**

| PULSA CLOCK | $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 2 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 0 | 1 |
| 4 | 1 | 0 | 0 | 0 |
| 5 | 0 | 1 | 0 | 0 |
| 6 | 0 | 0 | 1 | 0 |
| 7 | 0 | 0 | 0 | 1 |
| 8 | 1 | 0 | 0 | 0 |

Untuk mengoperasikan pencacah ring harus ada pemberian nilai awal sehingga terdapat sebuah flip-flop saja yang outputnya bernilai 1 dan flip-flop yang lain outputnya bernilai 0, misalnya $Q_3Q_2Q_1Q_0$=1000. Keadaan ini dapat diperoleh jika flip-flop yang digunakan

dilengkapi dengan input *preset* dan *clear*. Jadi, sebelum pulsa *clock* diberikan, salah satu flip-flop penyusun pencacah ring diberikan *preset* dan lainnya *clear*.

Selain pencacah ring, terdapat pula pencacah yang wataknya hampir mirip dengan pencacah ring. Pencacah ini dinamakan pencacah Johnson yang dapat diperoleh dari pencacah ring yang dimodifikasi. Gambar 170 (a) menunjukkan rangkaian hasil modifikasi pencacah ring menjadi pencacah Johnson dan gambar 170 (b) menunjukkan diagram waktunya.




(a) (b)

**Gambar 170. Pencacah Johnson: (a) rangkaian, dan (b) diagram waktu**

Cara kerja pencacah Johnson dapat dijelaskan sebagai berikut. Anggap mula-mula keadaan output pencacah 0000 sehingga $\overline{Q_0}$=1, dan karena $\overline{Q_0}$ dihubungkan dengan input $D_3$ maka $D_3=\overline{Q_0}$=1. Pada *clock* ke-1, karena input $D_3$ tinggi dan input $D_2$, $D_1$, $D_0$ rendah maka output $Q_3$ tinggi dan output lainnya rendah atau $Q_3Q_2Q_1Q_0$=1000. Pada *clock* ke-2, input $D_3$, $D_2$ keadaannya tinggi dan input $D_1$, $D_0$ rendah sehingga $Q_3Q_2Q_1Q_0$=1100, pada *clock* ke-3 input $D_3$, $D_2$, dan $D_1$ keadaannya tinggi dan input $D_0$ rendah sehingga $Q_3Q_2Q_1Q_0$=1110, dan pada *clock* ke-4 input $D_3$, $D_2$, $D_1$, dan $D_0$ tinggi sehingga $Q_3Q_2Q_1Q_0$=1111. Pada *clock* ke-5, $\overline{Q_0}$ rendah sehingga input $D_3$ juga rendah. Karena input $D_3$ rendah dan input $D_2$, $D_1$, $D_0$ tinggi maka $Q_3Q_2Q_1Q_0$=0111, pada *clock* ke-6 input $D_3$, $D_2$ rendah dan input $D_1$, $D_0$ tinggi sehingga $Q_3Q_2Q_1Q_0$=0011, pada *clock* ke-7 input $D_3$, $D_2$, $D_1$ rendah dan input $D_0$ tinggi sehingga $Q_3Q_2Q_1Q_0$=0001, dan pada *clock* ke-8 input $D_3$, $D_2$, $D_1$, $D_0$ rendah sehingga $Q_3Q_2Q_1Q_0$=0000. Pada *clock* ke-9 output pencacah kembali seperti urutan semula. Keadaan-keadaan tersebut dapat dituangkan dalam bentuk tabel keadaan seperti disajikan pada tabel 65.

**Tabel 65. Tabel keadaan pencacah Johnson**

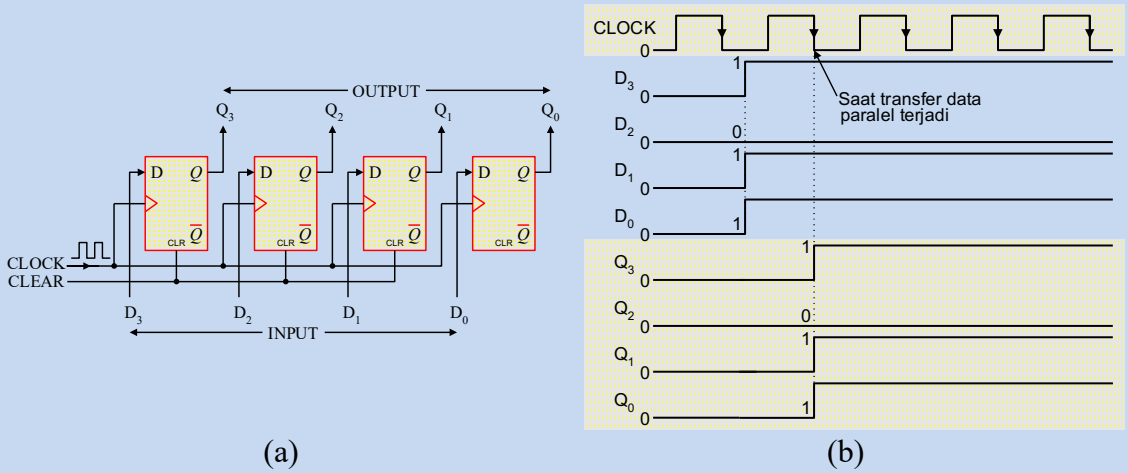
| PULSA CLOCK | $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 2 | 1 | 1 | 0 | 0 |
| 3 | 1 | 1 | 1 | 0 |
| 4 | 1 | 1 | 1 | 1 |
| 5 | 0 | 1 | 1 | 1 |
| 6 | 0 | 0 | 1 | 1 |
| 7 | 0 | 0 | 0 | 1 |
| 8 | 0 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 0 |
| 10 | 1 | 1 | 0 | 0 |

## B. Register

Telah dikemukakan di muka bahwa flip-flop merupakan elemen logika yang berfungsi menyimpan data. Data-data yang disimpan pada elemen tersebut berbentuk keadaan biner yang dapat berupa angka maupun huruf yang disusun dalam format kode seperti BCD dan ASCII. Oleh karena data-data itu berbentuk suatu keadaan biner yang panjangnya lebih dari satu bit maka untuk menyimpannya diperlukan elemen yang terdiri atas beberapa flip-flop, dan elemen seperti itu dinamakan register.

### 1. Register Paralel

Perhatikan rangkaian register yang disusun seperti ditunjukkan pada gambar 171 (a) berikut ini.



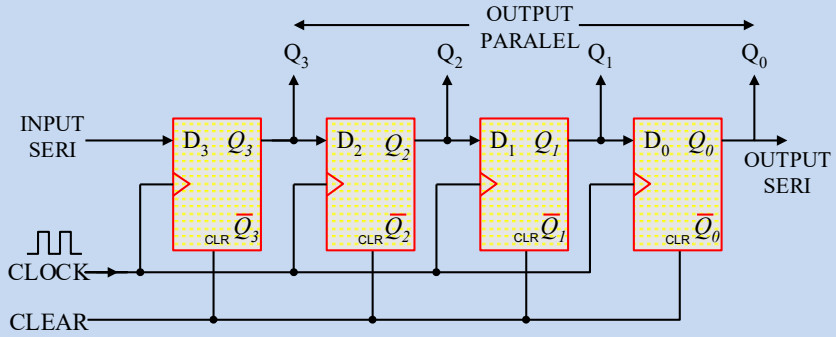

(a) (b)

**Gambar 171. Register paralel 4-bit: (a) rangkaian, (b) cara penyimpanan data 1011**

Gambar 171 (a) menunjukkan register paralel karena memiliki input dan output berupa saluran data paralel dengan panjang n-bit atau dalam contoh ini 4-bit yang dibangun dari kumpulan flip-flop D. Pada register ini data dimasukkan ke dalamnya secara serempak melalui saluran $D_3D_2D_1D_0$. Demikian pula ketika register tersebut akan dibaca outputnya, data dikeluarkan secara serempak melalui $Q_3Q_2Q_1Q_0$. Prinsip penyimpanan data pada register adalah memindahkan data yang ada pada inputnya ke outputnya. Penyimpanan data pada register paralel dilakukan dengan cara menempatkan data yang akan disimpan pada input paralel, dan untuk memindahkan data tersebut ke outputnya dilakukan dengan memberikan sebuah pulsa *clock*. Gambar 171 (b) menunjukkan ilustrasi cara penyimpanan data pada register paralel. Pada gambar tersebut dianggap register melakukan penyimpanan data 1011. Mula-mula ditempatkan data pada saluran input register yakni $D_3D_2D_1D_0$=1011, dan saat terjadinya tepi turun dari *clock* data dipindah ke output register sehingga $Q_3Q_2Q_1Q_0$=1011.

## 2. Register Geser

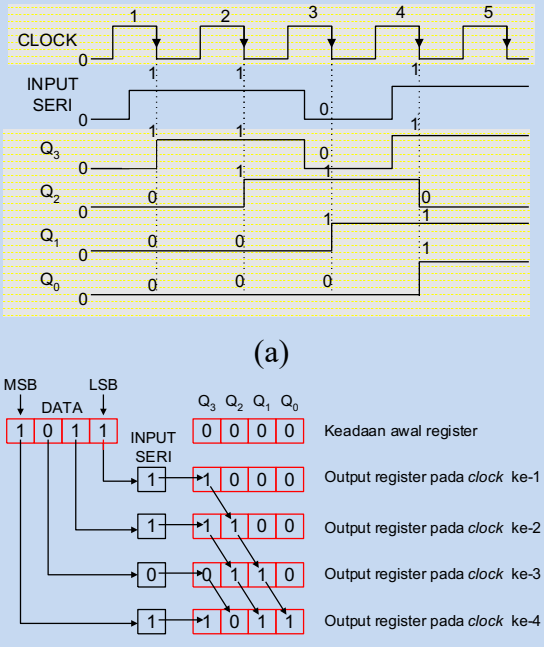
Selain register paralel yang dapat menyimpan data secara serempak, terdapat pula register geser yang melakukan penyimpanan data secara seri dengan memasukkan data bit demi bit. Disebut register geser karena dalam memindahkan data dari input ke outputnya, register ini melakukan penggeseran bit yang ada di dalam elemen-elemennya. Gambar 172 menunjukkan rangkaian register geser 4-bit yang memiliki 1-bit input dan 1-bit output seri, serta 4-bit output paralel.




**Gambar 172. Rangkaian register geser 4-bit**

Untuk memahami cara kerja register geser dalam menyimpan data, perhatikan ilustrasi pada gambar 173! Anggap data yang akan disimpan adalah 1011 dan keadaan mula-mula isi register masih kosong sehingga $Q_3Q_2Q_1Q_0$=0000. Mekanisme penyimpanan datanya dilakukan dengan memasukkan terlebih dahulu bit LSB dari data yang akan disimpan ke bagian elemen

MSB register. Untuk memulai penyimpanan data dipasang terlebih dahulu data MSB yakni 1 pada input seri. Tepi turun pulsa *clock* ke-1 akan memicu semua input flip-flop. Oleh karena pada input flip-flop $D_3$ terpasang data 1 maka $Q_3$=1. Pada sisi lain input flip-flop $D_2$, $D_1$, dan $D_0$ bernilai 0 sehingga $Q_3Q_2Q_1Q_0$=1000. Sebelum ada *clock* ke-2 dipasang lagi data berikutnya yakni 1 pada input seri. Tepi turun *clock* ke-2 menyebabkan semua input flip-flop kembali terpicu. Oleh karena $D_3$=1 (berasal dari input seri) dan $D_2$=$Q_3$=1, maka $Q_3$ dan $Q_2$ bernilai 1. Pada sisi lain $D_1$ dan $D_0$ bernilai 0 sehingga $Q_3Q_2Q_1Q_0$=1100. Keadaan tersebut menyebabkan seolah-olah isi $Q_3$ digeser ke posisi $Q_2$. Sebelum ada *clock* ke-3, data berikutnya yakni 0 dipasang pada input seri, dan pada tepi turun *clock* ke-3 semua input flip-flop kembali terpicu menyebabkan $Q_3$=0 (berasal dari input seri), $Q_2$ dan $Q_1$ bernilai 1. Pada sisi lain $Q_0$=0 sehinga $Q_3Q_2Q_1Q_0$=0110. Keadaan tersebut menyebabkan seolah-olah isi $Q_3$ digeser ke $Q_2$ dan isi $Q_2$ digeser ke $Q_1$. Selanjutnya, sebelum tepi turun *clock* ke-4 terjadi, pada input seri dipasang data yang terakhir yakni 1. Oleh karena keadaan sebelumnya $D_2$=$Q_3$=0, $D_1$=$Q_2$=1, dan $D_0$=$Q_1$=1 maka setelah terjadinya tepi turun *clock* ke-4 menjadikan $Q_2$=0 dan $Q_1$=1 dan $Q_0$=1. Pada sisi lain pemasangan data 1 pada input seri menyebabkan $Q_3$=1 sehingga $Q_3Q_2Q_1Q_0$=1011. Keadaan ini menyebabkan seolah-olah isi $Q_3$ digeser ke $Q_2$, isi $Q_2$ digeser ke $Q_1$, dan isi $Q_1$ digeser ke $Q_0$. Terlihat bahwa pada register geser 4-bit, penyimpanan data yang dilakukan memerlukan waktu sebanyak 4 siklus *clock*, sebab setelah *clock* ke-4 isi register sama dengan data yang dimasukkan yakni 1011.

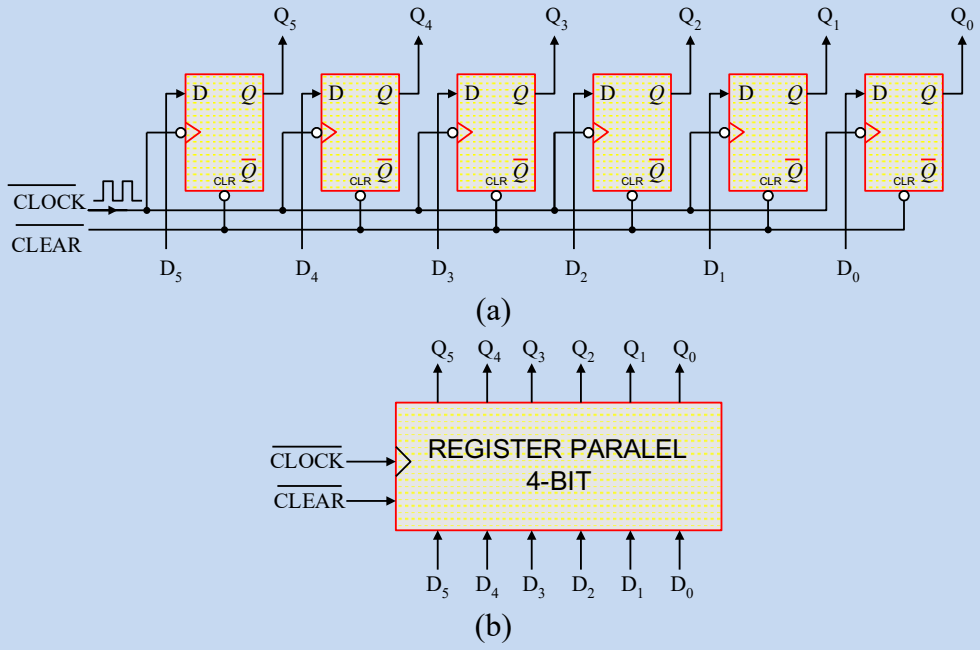



**Gambar 173. Ilustrasi penyimpanan data 1011 pada register geser**

## 3. IC Register

Berdasarkan cara memasukkan data ke dalam inputnya untuk disimpan dan cara mengeluarkan data melalui outputnya, register dapat dibedakan menjadi register dengan input dan output paralel *(parallel in-parallel out*) disingkat PIPO, register dengan input paralel dan output seri (*parallel in-serial out*) disingkat PISO, register dengan input seri dan output seri (*serial in-serial out*) disingkat SISO, register dengan input seri dan output paralel (*serial in-parallel out*) disingkat SIPO.

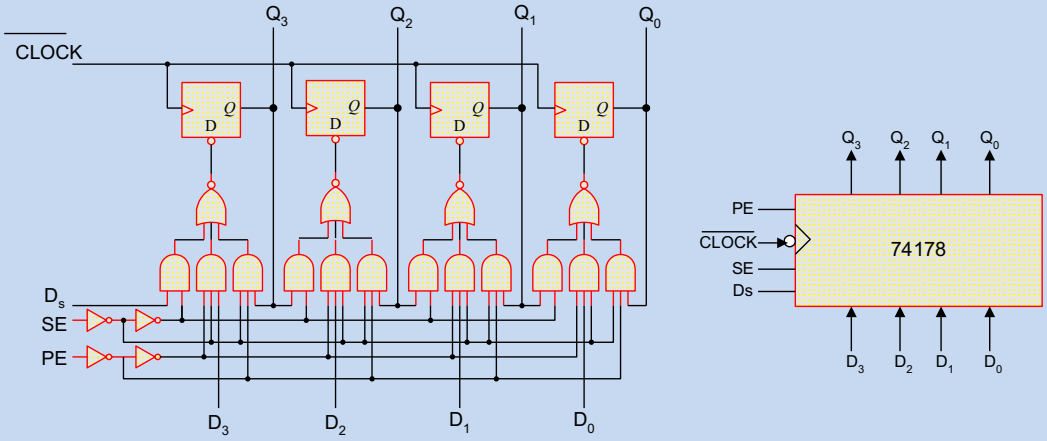
Salah satu IC yang menyediakan fungsi register PIPO adalah seri 74174 yang spesifikasi pinnya seperti ditunjukkan pada gambar 133 (c) di muka. Rangkaian internal dan simbol untuk IC ini ditunjukkan pada gambar 174. IC ini merupakan register paralel 6-bit yang dibangun menggunakan flip-flop D, dan memiliki input *clear* jenis *active-low*.




**Gambar 174. Register paralel 6-bit dari IC 74174: (a) rangkaian internal, dan (b) simbol**

Selain seri 74174, IC lain yang menyediakan fungsi register PIPO adalah seri 74178. IC ini memiliki input paralel 4-bit ($D_0$ sampai dengan $D_3$), output paralel 4-bit ($Q_0$ sampai dengan $Q_3$) , input seri 1-bit (Ds), 2 buah input *enable* yakni PE (*parallel enable*), dan SE (*serial enable*). Dengan demikian register ini dapat dioperasikan sebagai register paralel dan register geser 4-bit, sehingga dapat berperan sebagai PIPO, SIPO, SISO, maupun PISO. Bahkan dengan memberikan sedikit tambahan koneksi pada rangkaian luar, IC ini dapat

dimanfaatkan sebagai pencacah ring. Rangkaian internal IC 74178 dan simbolnya ditunjukkan pada gambar 175.




**Gambar 175. IC 74178: (a) rangkaian internal, dan (b) simbol**

Untuk mengoperasikan IC ini sebagai register paralel maupun seri perlu diatur pemberian nilai logika pada input PE dan SE. Berdasarkan lembar data yang dikeluarkan oleh pabriknya, nilai dari input-input *enable* yang diperlukan untuk operasi register paralel adalah PE=1 dan SE=0, untuk operasi register geser PE=X (*don't care condition*) dan SE=1, sedangkan jika PE=0 dan SE=0 register tidak aktif.

## 4. Transfer Register

Operasi yang berhubungan dengan data yang tersimpan di dalam register atau flip-flop dinamakan operasi mikro (*microoperation*) seperti *load*, *clear*, *shift*, dan *rotate*.. *Load* adalah operasi untuk memuati atau mengisi data ke dalam register, *clear* merupakan operasi menghapus data dalam register, *shift* atau geser adalah operasi untuk menggeser posisi data dalam register ke kiri atau ke kanan, dan *rotate* merupakan operasi untuk memutar data ke kiri atau ke kanan. Selain itu, terdapat pula operasi mikro aritmetika seperti penambahan, pengurangan, perkalian, pembagian, *increment* (penambahan dengan 1) dan *decrement* (pengurangan dengan 1) terhadap isi suatu register, serta operasi mikro logika seperti AND, OR, dan NOT. Beberapa dari operasi mikro tersebut seperti *load*, *clear* dan *shift* telah Anda pelajari melalui berbagai percobaan di muka.

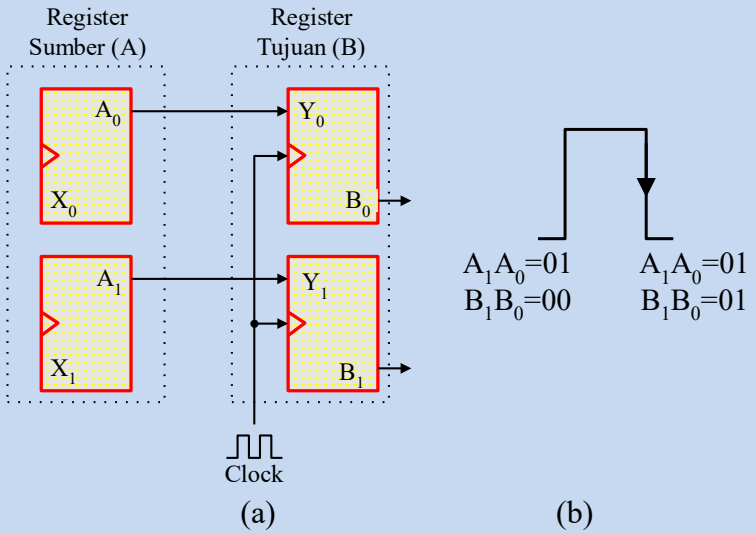
Selain itu, operasi mikro juga meliputi operasi transfer yakni pemindahan data dari satu register ke register yang lain. Pada operasi ini, isi suatu register yang dipindah ke register lain, setelah operasi dilakukan keadaannya tetap atau tidak berubah. Dengan kata lain, operasi transfer merupakan proses penyalinan data. Dalam hal ini, register yang isinya disalin

dinamakan register sumber (*source register*) dan register penampung data salinan dinamakan register tujuan (*destination register*). Mekanisme transfer data dapat dilakukan dengan berbagai cara antara lain transfer paralel dan transfer seri.

**a. Transfer Data Paralel**

Pada transfer data paralel, pemindahan atau penyalinan data dari register sumber ke register tujuan dilaksanakan secara serempak. Contoh dalam kehidupan sehari-hari transfer paralel dapat dianalogikan dengan peristiwa memasukkan sekelompok orang ke dalam stadion secara bersama-sama lewat sebuah pintu yang lebar. Dalam konteks register, hal ini berarti, semua data yang tersimpan pada setiap flip-flop yang merupakan elemen-elemen register sumber disalin secara serempak ke register tujuan. Untuk menyelenggarakan operasi transfer data paralel diperlukan register sumber dan tujuan jenis PIPO.

Gambar 176 menunjukkan ilustrasi transfer paralel. Pada gambar tersebut ditunjukkan sebuah register sumber (register A) dan register tujuan (register B) yang di dalamnya memiliki dua buah elemen flip-flop. Notasi yang digunakan pada register sumber adalah $X_1$ dan $X_0$ masing-masing input MSB dan LSB, serta $A_1$ dan $A_0$ masing-masing sebagai output MSB dan LSB. Sedangkan pada register tujuan digunakan notasi $Y_1$ dan $Y_0$ sebagai input MSB dan LSB, serta $B_1$ dan $B_0$ sebagai output MSB dan LSB.



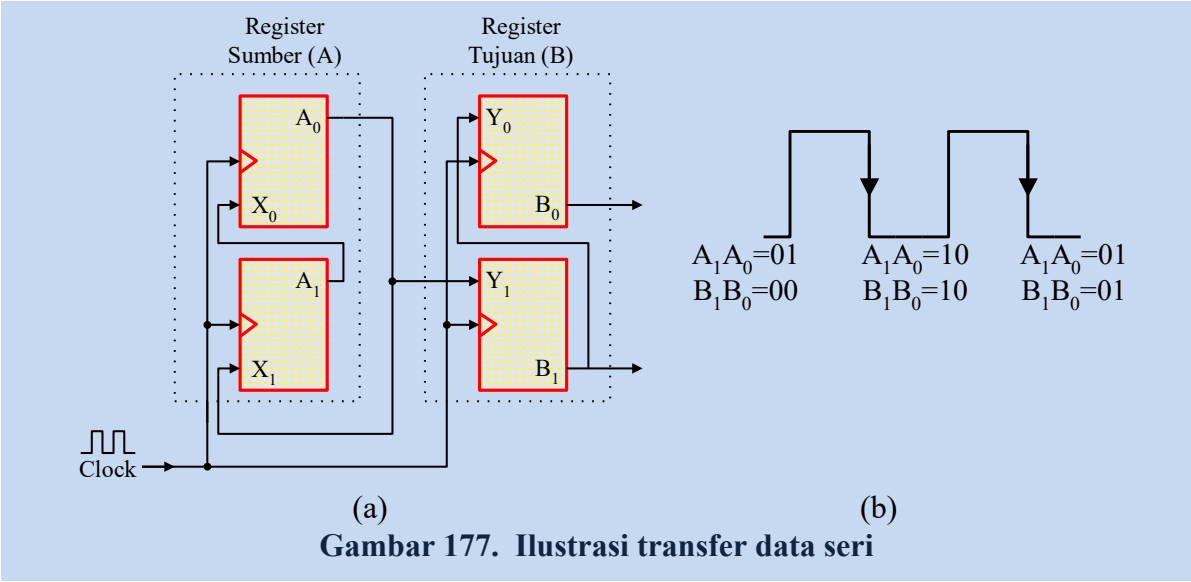

**Gambar 176. Ilustrasi transfer data paralel**

Pada transfer paralel, output setiap elemen register sumber dihubungkan dengan input setiap elemen register tujuan. Anggap keadaan output mula-mula dari register sumber adalah $A_1A_0$=01 dan output register tujuan $B_1B_0$=00. Jika pada setiap flip-flop penyusun register tujuan diberikan sebuah pulsa *clock* maka isi register sumber akan disalin secara serempak ke register tujuan sehingga output keduanya sama yakni $A_1A_0 = B_1B_0$=01.

Agar Anda lebih memahami mekanisme transfer data paralel dari suatu register ke register lain coba lakukan percobaan berikut.

**b. Transfer Data Seri**

Pada transfer data seri, pemindahan data dilakukan bit demi bit. Dalam kehidupan sehari-hari transfer ini mirip dengan memasukkan sekelompok orang ke stadion tetapi lewat pintu yang sempit, sehingga harus dilakukan satu per satu. Untuk menyelenggarakan proses ini diperlukan register geser atau register seri yang susunannya seperti pada gambar 177 (a). Transfer ini juga memerlukan operasi *rotate* atau putar sehingga output LSB register sumber ($A_0$) selain dihubungkan ke input MSB register tujuan ($Y_1$), juga diumpankan ke inputnya sendiri yakni input MSB ($X_1$).




**Gambar 177. Ilustrasi transfer data seri**

Anggap mula-mula isi register sumber $A_1A_0$=01 dan isi register tujuan $B_1B_0$=00. Perhatikan rangkaian di luar register sumber pada gambar 177 (a)! Pada *clock* pertama isi $A_0$ dipindah ke $B_1$ lewat $Y_1$ sehingga $B_1B_0$=10. Sekarang perhatikan rangkaian di dalam register sumber! Pada saat yang bersamaan di dalam register sumber isi $A_1$ digeser ke $A_0$ lewat $X_0$ dan isi $A_0$ diputar ke $A_1$ sehingga isi register sumber menjadi $A_1A_0$=10. Pada *clock* kedua, dalam register tujuan isi $B_1$ digeser ke posisi $B_0$, dan isi $A_0$ dipindah ke $B_1$ sehingga isi register tujuan menjadi $B_1B_0$=01. Pada bagian register sumber, isi $A_1$ digeser ke posisi $A_0$ dan isi $A_0$ diputar ke posisi $A_1$ sehingga isi register sumber menjadi $A_1A_0$=01. Proses transfer berhenti karena isi kedua register telah sama yakni 01. Terlihat bahwa untuk register 2-bit proses transfer data memerlukan dua buah pulsa *clock*.

## C. Soal Latihan

Soal nomor 1 sampai dengan nomor 4 adalah jenis pilihan ganda. Kerjakan dengan cara memilih satu jawaban yang paling sesuai dari opsi yang tersedia.

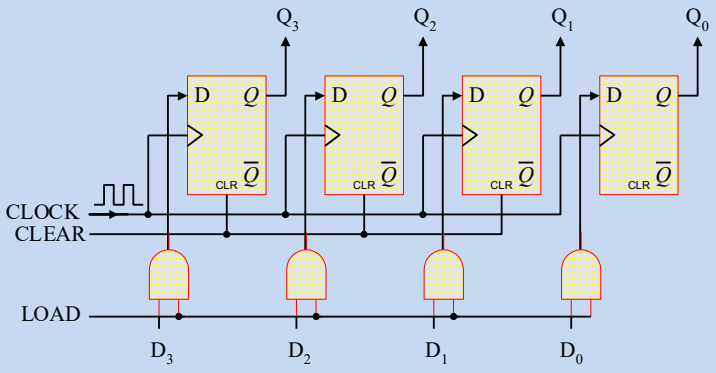
1. Rangkaian sekuensial adalah rangkaian logika yang outputnya:
   a. tidak dilengkapi dengan memori
   b. tidak tergantung pada waktu
   c. tergantung pada keadaan inputnya
   d. tergantung pada keadaan output sebelumnya
   e. tergantung *clock*
2. Pengertian pencacah (*counter*) yang paling tepat adalah:
   a. Rangkaian logika yang outputnya diambil dari output flip-flop penyusunnya
   b. Rangkaian logika yang berfungsi menjumlah sinyal pada input-inputnya
   c. Rangkaian logika yang mengandung elemen flip-flop
   d. Rangkaian logika yang berfungsi menyimpan data dalam satu baris memori
   e. Rangkaian logika sekuensial yang berfungsi menjumlah pulsa *clock* yang masuk ke inputnya
3. Pencacah yang setiap elemennya bekerja secara tidak bersamaan dinamakan:
   a. Pencacah Sinkron
   b. Pencacah Asinkron
   c. Pencacah Pulsa
   d. Pencacah Ring
   e. Pencacah Up-Down
4. *Shift register* melakukan penyimpan data dengan cara:
   a. Data dimasukkan secara serempak dengan menggeser bit-bit nya
   b. Data dimasukkan secara serial dimulai dari data MSB
   c. Data dimasukkan secara serial dimulai dari data LSB
   d. Data dimasukkan secara serempak dalam waktu yang bersamaan
   e. Data dimasukkan dengan cara menggeser bit-bit nya dimulai dari MSB

Soal-soal berikut ini adalah soal bentuk uraian (esai).

5. Berapa jumlah flip-flop yang diperlukan untuk membangun pencacah modulo-6, modulo-9, modulo-17, modulo-25, dan modulo-50?
6. Pencacah modulo-5 diberi input pulsa *clock* dengan frekuensi 2 MHz. Hitung frekuensi gelombang kotak pada output flip-flop terakhir (MSB)!

7. Rancang pencacah tak serempak modulo-9 dan modulo-24 menggunakan flip-flop J-K dengan:
   a. *clear* jenis *active-high*, dan
   b. *clear* jenis *active-low*
8. Susun rangkaian pencacah tak serempak modulo-4 dan modulo-13 menggunakan IC 7493!
9. Rancang rangkaian pencacah serempak modulo-6 menggunakan flip-flop T!
10. Rancang rangkaian pencacah serempak naik-turun modulo-7 menggunakan:
    a. flip-flop J-K, dan
    b. flip-flop T

    Anggap pencacah tersebut memiliki input pengontrol urutan C. Jika C=1 output pencacah memberikan urutan naik dan apabila C=0 output pencacah memberikan urutan turun.
11. Susun rangkaian pencacah bertingkat modulo-50 menggunakan dua buah IC 7490. Lengkapi pencacah tersebut dengan peraga 7-segmen untuk menampilkan outputnya. Gunakan IC 7447 sebagai *decoder* BCD ke peraga 7-segmen jenis *common anode*! Susun pula rangkaian yang sama dengan menggunakan dua buah IC 74193!
12. Susun rangkaian register paralel 7-bit menggunakan flip-flop D dan tunjukkan dengan diagram waktu cara register tersebut menyimpan data 1100101.
13. Jelaskan mekanisme penyimpanan data pada soal 12 ke dalam register geser!
14. Perhatikan rangkaian register paralel dengan pengontrol LOAD berikut ini!




**Gambar 178. Rangkaian untuk soal nomor 14 dan 15 Bab VII**

Jika data 1011 dipasang pada input register, jelaskan cara penyimpanan data tersebut ke dalam register!

15. Gambarkan diagram waktu dari rangkaian pada gambar 178, jika data 1011 dipasang pada input register, dan sinyal LOAD=1 diberikan di antara *clock* ke-2 dan ke-3 sampai dengan *clock* ke-5.
16. Tunjukkan cara menyimpan data 1011 pada IC register 74178 secara paralel maupun seri! Lengkapi penjelasan Anda dengan gambar simbol dari 74178 yang telah diatur untuk keperluan operasi-operasi tersebut.
17. Susun rangkaian untuk menyelenggarakan transfer paralel register A ke register B. Anggap kedua register tersebut masing-masing merupakan register paralel 8-bit yang dibangun dari dua buah IC 74178. Jelaskan cara mengisi register A dengan data 10011001 dan memindahkan isi tersebut ke register B. Susun diagram waktu untuk operasi transfer tersebut jika data pada register A tersedia pada *clock* ke-2 dan sinyal PE=1 pada register B diberikan di antara *clock* ke-3 dan ke-4 sampai dengan *clock* ke-6!

# DAFTAR PUSTAKA

Elektuur (Alih bahasa: Wasito). 1996. *Data Sheet Book 1*. Jakarta: PT Elex Media Komputindo.

Hall, D. V. 1993. *Microprocessors and Interfacing: Programming and Hardware, 2/E.* Lake Forest: Glencoe Division of Macmillan/McGraw-Hill School Publishing Company.

Hill, F. J. and Peterson, G. R. 1981. *Switching Theory and Logical Design*. New York: John Wiley & Sons, Inc.

Malvino, A. P. and Brown J. A. *Digital Computer Electronics*. Lake Forest: Glencoe Division of Macmillan/McGraw-Hill School Publishing Company.

Mano, M. M. 1992. *Computer System Architecture (3rd Edition)*. Englewood Cliff: Prentice Hall, Inc.

Mismail, B. 1998. *Dasar-Dasar Rangkaian Logika Digital*. Bandung: Penerbit ITB.

Murdocca, M. and Heuring, V. P. 1999. *Principles of Computer Architecture*. Englewood Cliff: Prentice Hall, Inc.

Sicard, E. and Xi, C. 2003. *Dsch2 Commands.* Http://intrage.insa-tlse.fr/~etienne

Smith, R. J. and Dorf, R. C. 1992. *Circuits, Devices and Systems*. New York: John wiley & Sons, Inc.

Tocci, R. J. & Widmer, R. S. 2001. *Digital Systems*: *Principles and Applications*, *8th Edition*. Englewood Cliff: Prentice Hall, Inc.

www.digital.ni.com/public.nsf/allkb/826C981B3D0D3A9786256DD300188620. *How Do I Interface TTL Signals With CMOS Circuits?*

www.highered.mcgraw hill.com/sites/dl/free/0073126349/443702/Chapter05.pdf

www.interfacebus.com/voltage_threshold.html. *Logic Threshold Voltage Levels IC Specifications and Simple Interfacing*.

**Lampiran 1.**

## Ragam Tegangan IC TTL dan CMOS

Secara umum keluarga besar IC dibagi menjadi dua yakni TTL (*transistor-transistor* logic) dan CMOS (*complementary meoductor*). Anggota keluarga keduanya ditunjukkan oleh tabel berikut ini.

**Tabel 66.**
**Keluarga IC TTL**

| Keluarga TTL | |
|---|---|
| Nama | Awalan Nomor Seri IC |
| Standard TTL | 74XX |
| Low Power TTL (L) | 74LXX |
| Schottky TTL (S) | 74SXX |
| High Speed TTL (H) | 74HXX |
| Low-power schottky TTL (LS) | 74LSXX |
| Advanced- Schottky TTL (AS) | 74ASXX |
| Advanced Low Power Schottky (ALS) | 74ALSXX |
| Fast (Advanced –Schottky) TTL (F) atau (AST) | 74SXX atau 74ASTXX |

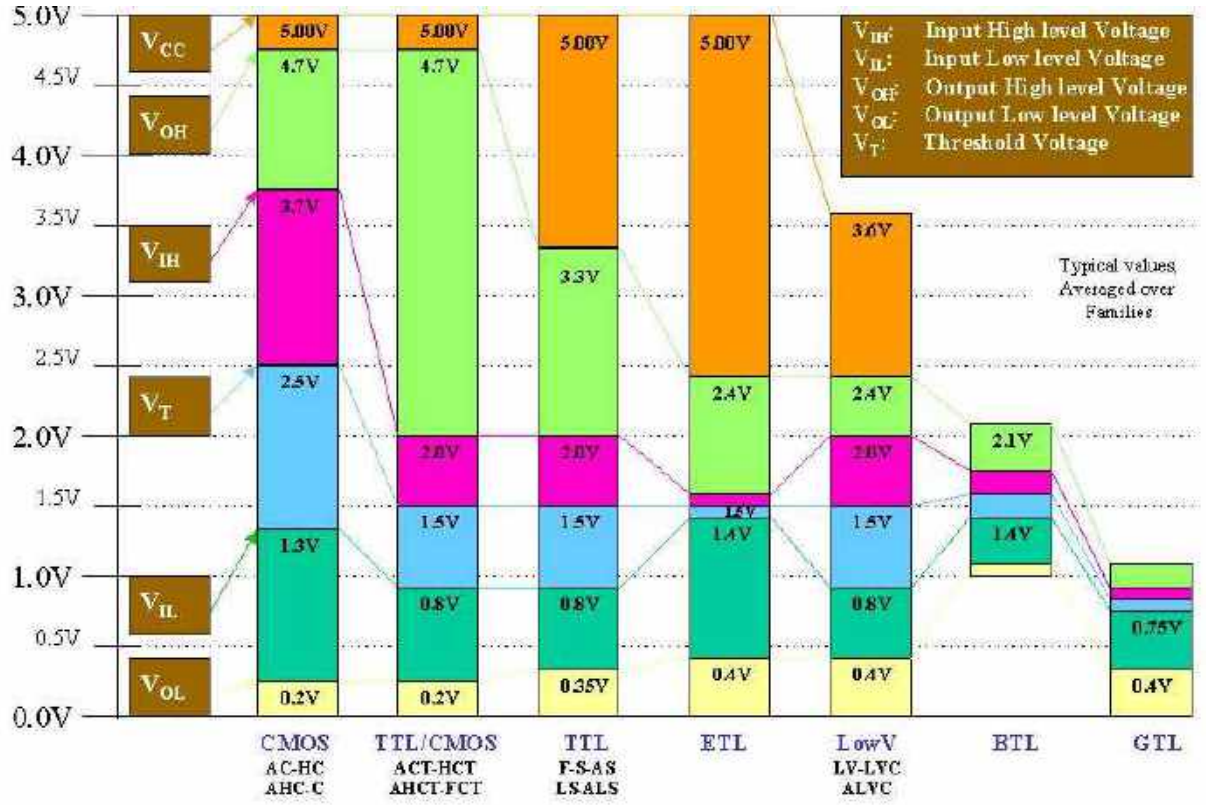
Sedangkan keluarga IC CMOS ditunjukkan tabel berikut ini.

**Tabel 67.**
**Keluarga IC CMOS**

| Keluarga CMOS | |
|---|---|
| Nama | Awalan Nomor Seri IC |
| Standard CMOS (C) | 74CXX |
| High Speed CMOS (HC) | 74HCXX |
| Advanced CMOS (AC) | 74ACXX |
| Advanced High Speed CMOS (AHC) | 74AHCXX |
| Gabungan CMOS dan TTL: | |
| Advanced CMOS compatible TTL (ACT) | 74ACTXX |
| High-speed CMOS compatible TTL (HCT) | 74HCTXX |
| Advanced-High-Speed CMOS compatible TTL (AHCT) | 74AHCTXX |
| Fast- CMOS compatible TTL (FCT) | 74FCTXX |

**Lanjutan Lampiran 1.**

Level tegangan IC keluarga TTL dan CMOS ditunjukkan pada gambar berikut ini.




Sumber: www.interfacebus.com

**Gambar 179. Level Tegangan IC Keluarga TTL dan CMOS**

Dari gambar 179, berdasarkan tegangan catunya, terdapat IC TTL maupun CMOS dengan tegangan catu 5 V dan 3,3 V. Jenis IC TTL/CMOS dengan tegangan catu 5 V adalah IC dengan awalan nomor seri 74FXX, 74SXX, 74ASXX, 74LSXX, 74ALSXX untuk TTL, dan 74ACXX, 74HCXX, 74AHCXX, dan 74CXX untuk CMOS. Sedangkan jenis IC dengan tegangan catu 3,3 V adalah IC TTL/CMOS dengan awalan nomor seri 74LVXX, 74LVCXX, dan 74ALVCXX.

**Keterangan tambahan untuk gambar 179:**

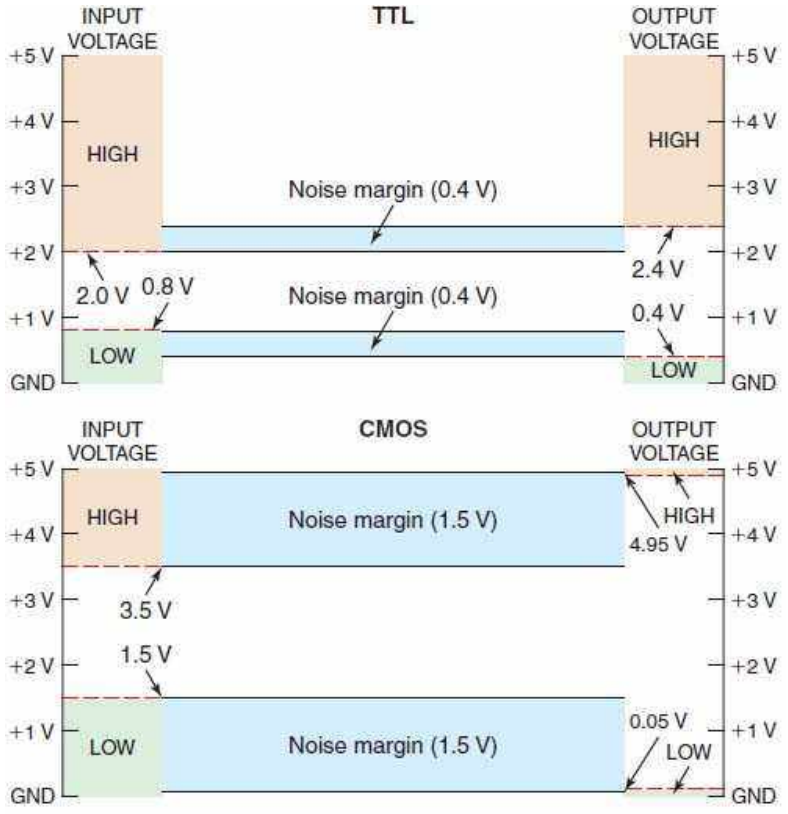
$V_{CC}$: tegangan yang diberikan pada pin catu daya
$V_{IH}$: *voltage input high*, merupakan tegangan positif minimum yang terpasang pada input yang akan diterima oleh piranti/IC sebagai logika tinggi
$V_{IL}$: *voltage input low*, adalah tegangan positif maksimum yang terpasang pada input yang akan diterima oleh piranti/IC sebagai logika rendah
$V_{OL}$: *voltage output low*, merupakan tegangan positif maksimum dari output yang dianggap oleh piranti/IC sebagai nilai maksimum posisitf logika rendah
$V_{OH}$: *voltage output high*, merupakan tegangan positif maksimum dari output yang dianggap oleh piranti/IC sebagai nilai minimum positif logika tinggi

**Lampiran 2.**

## *Noise Margin*

Salah satu kelebihan dari penggunaan CMOS sebagai piranti dalam sistem digital adalah selain kebutuhan dayanya rendah, juga kemampuannya yang baik dalam meredam *noise* yang tidak diinginkan. Dalam teknik digital, *noise* diartikan sebagai tegangan yang tidak diinginkan, biasanya muncul akibat induksi dari kabel-kabel penghubung maupun konduktor-konduktor pada papan rangkaian tercetak yang dapat berpengaruh pada level logika input sehingga menyebabkan kesalahan pada outputnya.

Kekebalan rangkaian digital terhadap *noise* disebut juga sebagai *noise margin* yang didefinisikan sebagai batas nilai tegangan yang tidak diinginkan yang dapat ditolak oleh piranti. *Noise margin* untuk TTL dan CMOS ditunjukkan pada gambar berikut ini.



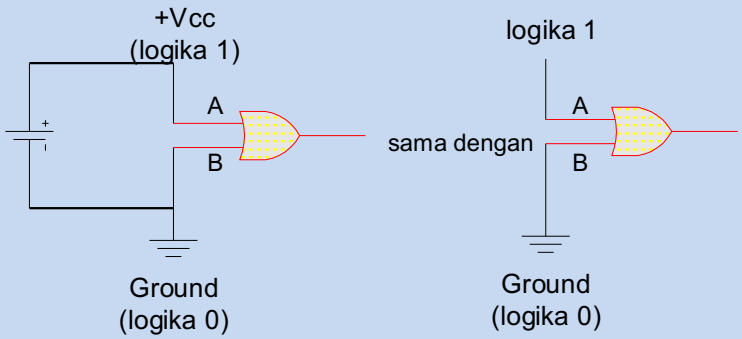

**Gambar 180. *Noise Margin* IC TTL dan CMOS**
(Sumber: http://highered.mcgraw-hill.com/sites/dl/free/0073126349/443702/Chapter05.pdf)

CMOS memiliki *noise margin* yang lebih baik dibandingkan TTL. Dengan *noise margin* sebesar 1,5 V, CMOS menjadi lebih tidak peka dibandingkan TTL dengan noise margin sebesar 0,4 V.

**Lampiran 3.**

### *Floating Input*

Salah satu watak dari gerbang-gerbang logika pada IC keluarga TTL adalah level logika inputnya tinggi ketika dalam keadaan mengambang (*floating*). Perhatikan contoh pada gambar berikut ini.



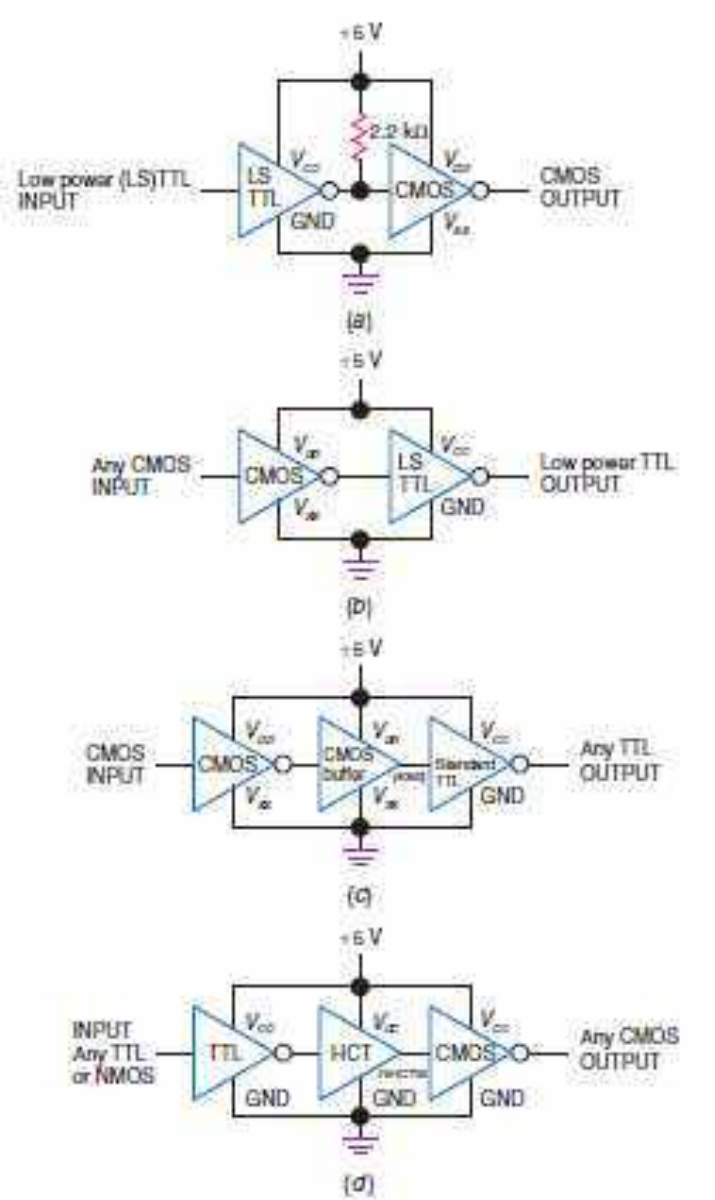

**Gambar 181. Watak Input Mengambang Pada Gerbang TTL**

Pada gambar 181 kiri, input A bernilai logika tinggi (1) karena dihubungkan dengan catu daya +Vcc dan pada gambar sebelah kiri input A tetap memiliki nilai logika tinggi (1) walaupun tidak dihubungkan dengan +Vcc. Hal ini menandakan bahwa input mengambang (*floating input*) pada gerbang-gerbang TTL memiliki nilai logika tinggi.

**Lampiran 4.**

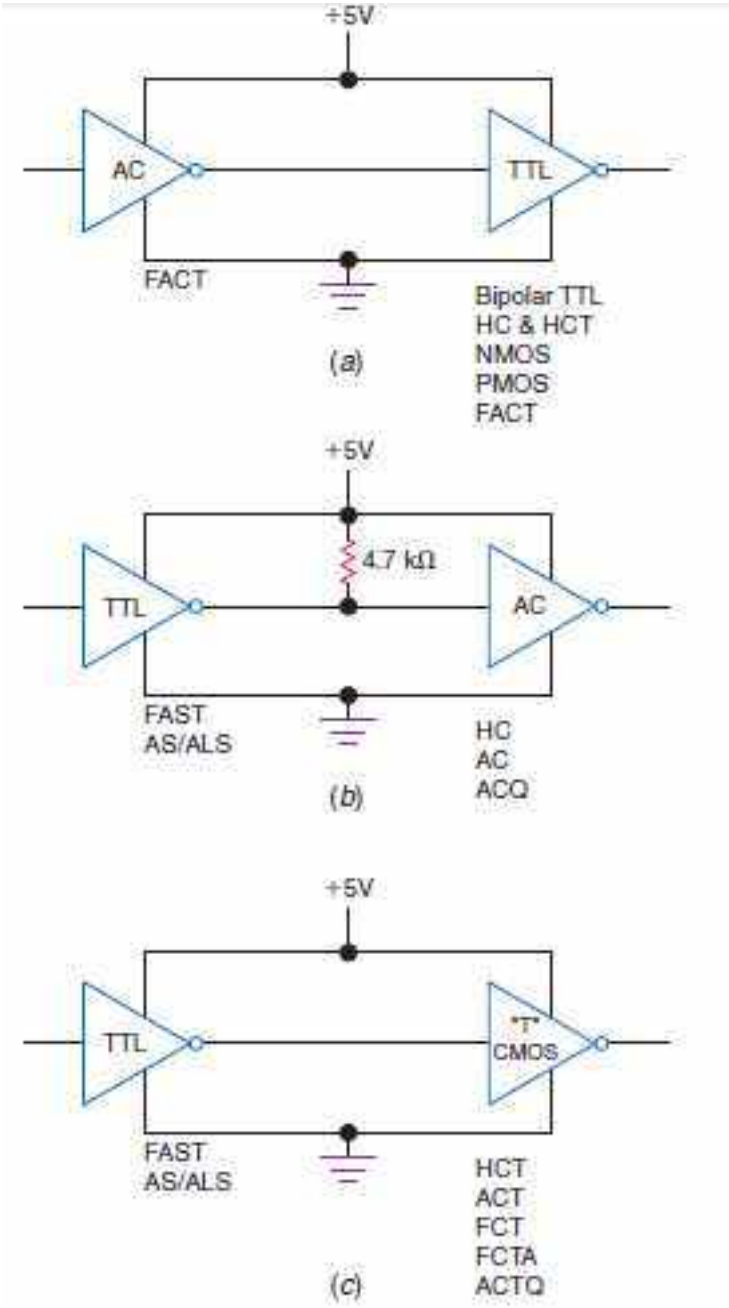
## Teknik *Interface* TTL-CMOS, CMOS-TTL, dan antar TTL

Dalam perancangan rangkaian atau sistem digital seringkali digunakan kombinasi IC sebagai piranti-piranti yang digunakan. Agar setiap jenis IC yang digunakan dapat bekerja secara wajar/normal, perlu dilengkapi dengan rangkaian antarmuka pada piranti-piranti yang digunakan seperti pada gambar-gambar berikut ini.



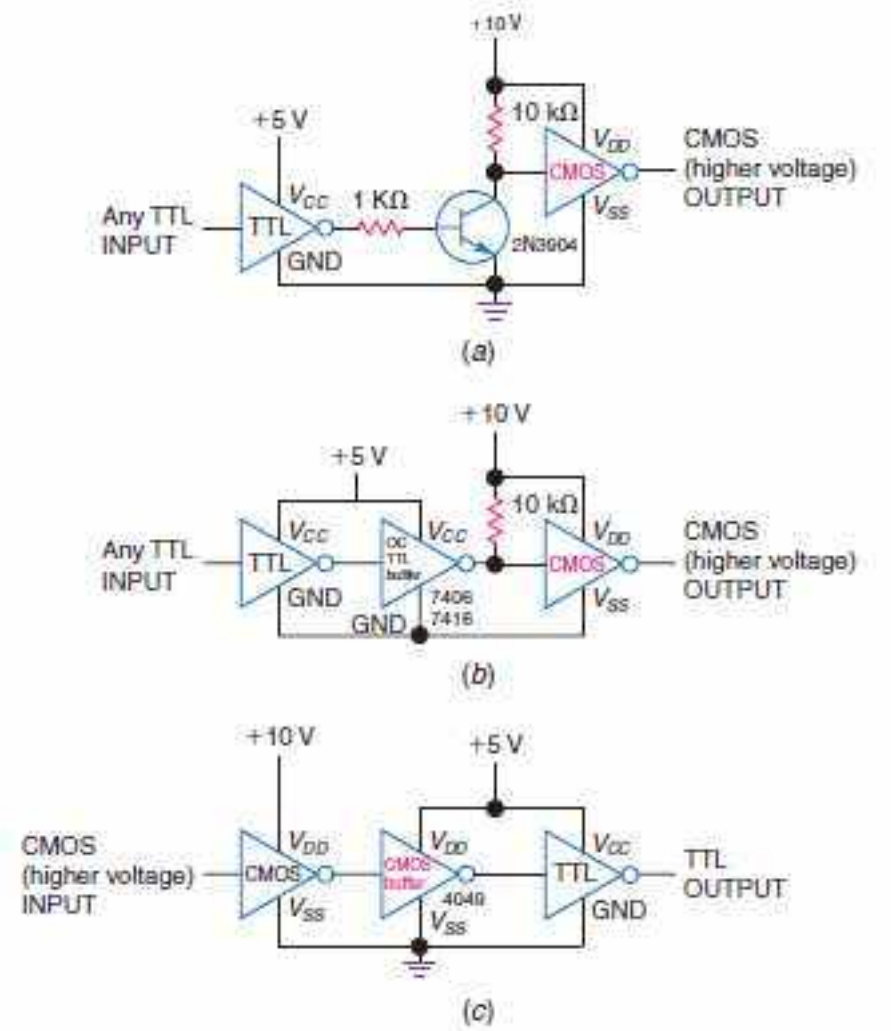

Gambar 182. *Interface* TTL dan CMOS Untuk Tegangan Catu +5V:(a) *Interface* Lowpower Schottky TTL ke CMOS Menggunakan Resistor *Pull-Up*, (b) *Interface* CMOS ke *Lowpower* TTL, (c) *Interface* CMOS ke TTL Standar Menggunakan CMOS Buffer, (d) TTL ke CMOS Menggunakan IC 74HCT00

**Lanjutan Lampiran 4.**




Gambar 183. *Interfacing FACT (Fairchild Advanced CMOS Technology)*, salah satu jenis IC CMOS Modern, Dengan Keluarga Lainnya: (a) FACT ke Keluarga TTL, (b) TTL ke FACT Menggunakan Resistor *Pull-Up*, (c) *TTL* ke CMOS "T"

**Lanjutan Lampiran 4.**




Gambar 184. *Interfacing* TTL dan CMOS Untuk Catu Daya Berbeda: (a) TTL ke CMOS Menggunakan Transistor Driver, (b) TTL ke CMOS Menggunakan IC Buffer TTL Open Collector, (c) CMOS ke TTL Menggunakan IC Buffer CMOS

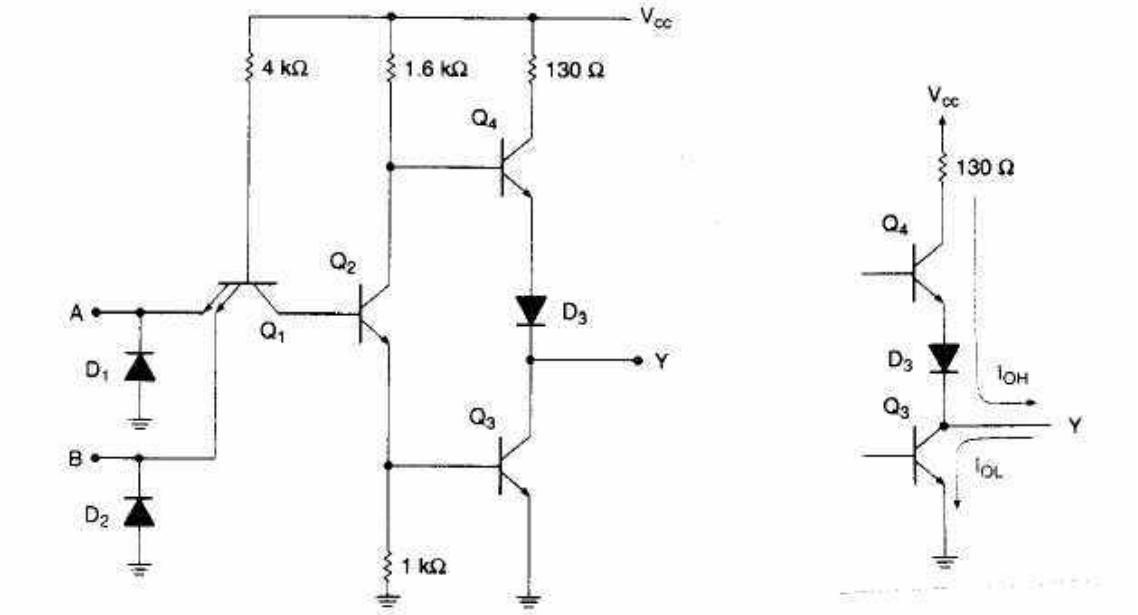
**Lampiran 5.**

## Output *Totem Pole* dan Output *Open Collector*

Salah satu karakteristik IC keluarga TTL adalah memiliki konfigurasi output jenis *totem pole* dan *open collector.* Untuk suatu nomor seri IC TTL, pengguna dapat memilih jenis konfigurasi output yang diinginkan.

### Output Totem Pole

*Totem pole* merupakan jenis output yang paling umum terdapat pada IC TTL. Gambar berikut ini adalah rangkaian internal IC 7400 yang menyediakan gerbang NAND dengan output jenis *totem pole*. Transistor Q3 dan Q4 disusun seperti tiang *totem* dan bekerja secara bergantian. Jika Q3 OFF, Q4 ON, maka arus $I_{OH}$ sebesar maksimum 0.4 mA mengalir lewat Q4, D3 dan Y menyebabkan output Y bernilai mendekati VCC atau berlogika tinggi. Sedangkan jika Q3 ON, Q4 OFF, arus $I_{OL}$ sebesar maksimum 1,6 mA mengalir dari Y lewat Q3 ke *ground,* menyebabkan output Y bernilai mendekati tegangan *ground* atau berlogika rendah.



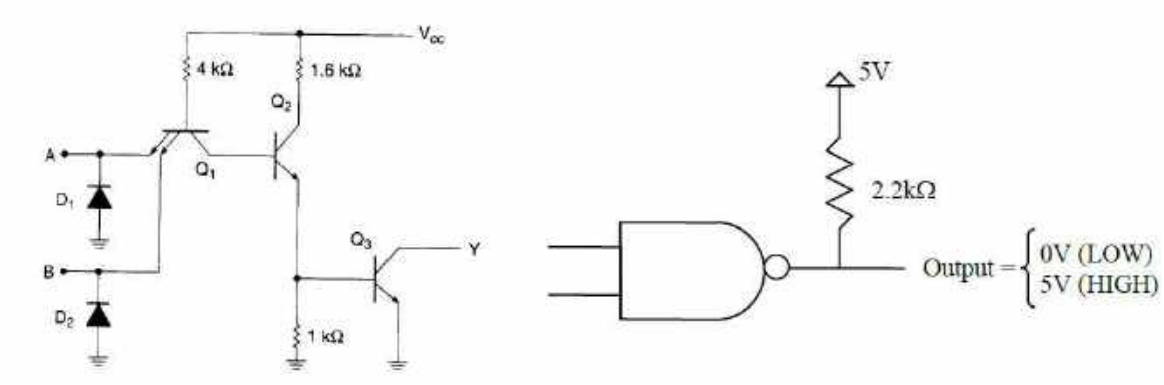

(a) (b)

**Gambar 185. (a) Rangkaian Internal IC TTL 7400 Dengan Konfigurasi Output *Totem Pole*, (b) Jalur Arus Pada**

**Lanjutan Lampiran 5.**

**Output *Open-Collector***

Jenis output yang lain dari IC TTL adalah *open-collector*. Gambar berikut ini adalah rangkaian internal IC 7401 yang menyediakan fungsi NAND dengan output jenis *open collector.*



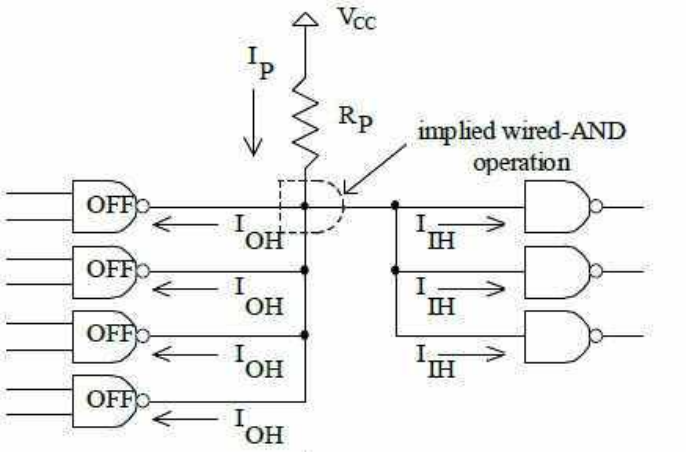

(a) (b)

Gambar 186. (a) Rangkaian Internal IC 7401 Dengan Output Jenis *Open-Colletor*, (b) Output Gerbang NAND Dengan *Pull-Up* Resistor

Output *open-collector* menggunakan sebuah transistor yakni Q3. Jika transistor Q3 ON maka Y terhubung dengan *ground* sehingga output Y bernilai logika rendah. Namun jika transistor Q3 OFF, maka *collector* dalam keadaan terbuka, sehingga perlu dipasang resistor eksternal *pull-up* seperti gambar 186 (b) agar outputnya memberikan nilai logika tinggi .

Dibandingkan dengan output *totem pole*, penggunaan IC TTL dengan output jenis *open-collector* dapat memberikan keuntungan yakni:

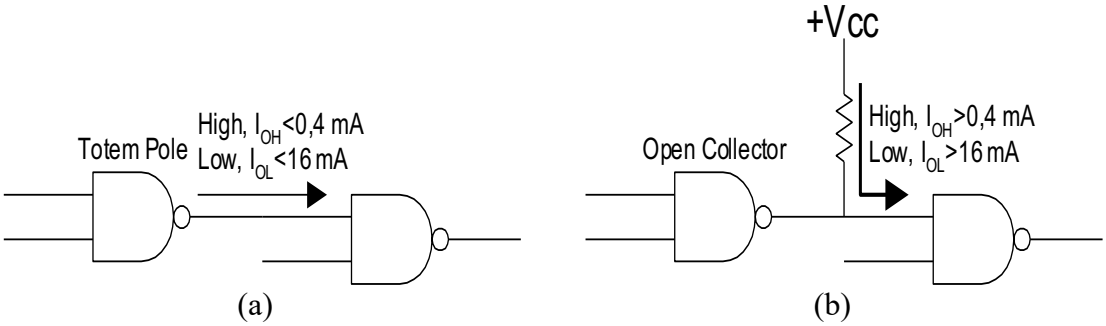
1. Jika diinginkan operasi AND terhadap banyak outputnya, pada jenis *open collector,* output-outputnya dapat disambungkan secara langsung menjadi satu tanpa merusak IC. Hal ini tidak bisa dilakukan pada IC TTL jenis output *totem pole*, karena dapat merusak piranti tersebut. Gambar berikut ini adalah ilustrasi peng-AND-an (ANDing) dari output-output gerbang *open collector.*




**Gambar 187. Operasi AND-ing Pada Output *Open Collector***

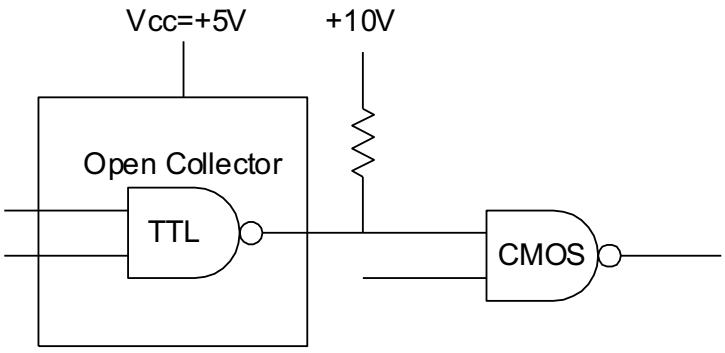
**Lanjutan Lampiran 5.**

2. Dapat meningkatkan level arus output. Gerbang-gerbang pada IC TTL standar dengan output jenis *totem pole* hanya mampu menyediakan arus output logika tinggi sebesar 0,4 mA dan logika rendah sebesar 1,6 mA. Namun, beberapa gerbang TTL jenis output *open collector* mampu memberikan arus dengan level lebih tinggi.




**Gambar 188. (a) Output *Totem Pole* Hanya Menyediakan Arus Maksimum 0,4 mA ($I_{OH}$) dan 16 mA ($I_{OL}$) , (b) Open Collector Dapat Menyediakan Arus Output Lebih Besar**

3. Dengan menggunakan gerbang jenis *open collector,* menjadikan *interfacing* TTL dengan berbagai keluarga IC logika lain yang memiliki tegangan berbeda, mudah dilakukan. Ouput IC TTL *open collector* dengan Vcc=+5V dapat memberi umpan tegangan 10V ke input IC CMOS, jika outputnya dilakukan *pull-up* menggunakan tegangan +10V.




**Gambar 189. *Interfacing* TTL (Vcc=5V) ke CMOS (Vcc=10V) Mudah Dilakukan Pada TTL Dengan Output *Open Collector***

**Lampiran 6.**

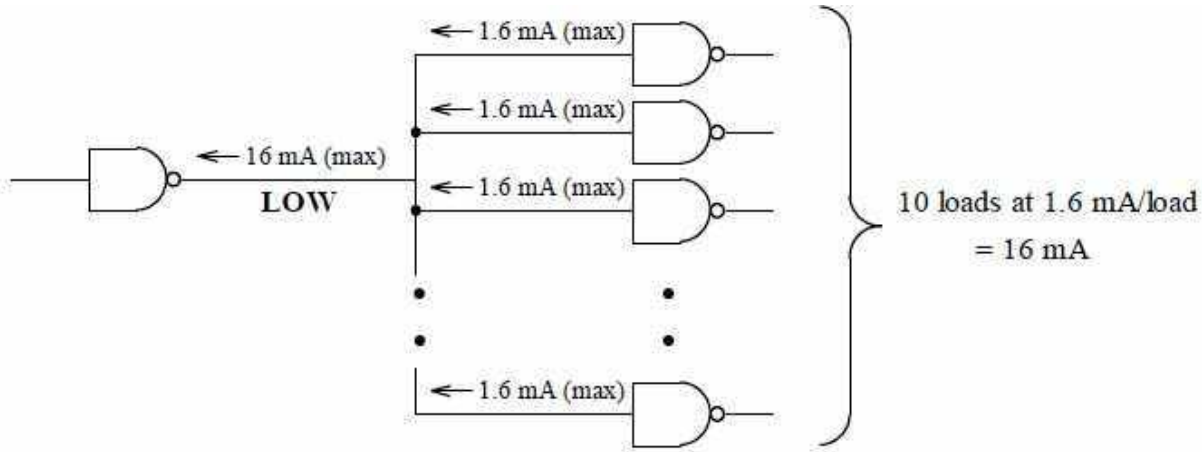
## *Fanin* dan *Fanout*

*Fanin* didefinisikan sebagai jumlah maksimum input yang masih dapat diterima oleh suatu gerbang logika. Dalam hal ini, jika jumlah input melebihi *fanin,* output gerbang akan menjadi salah atau tidak terdefinisi masuk level rendah atau tinggi.

*Fanout* adalah jumlah beban standar yang masih dapat dihubungkan dengan output suatu gerbang logika. Jumlah beban standar yang dimaksud adalah perbandingan besarnya arus output suatu gerbang terhadap arus input pada setiap gerbang-gerbang yang dihubungkan dengan output tersebut, atau dapat ditulis:

$$\text{fanout} = \frac{I_{OL}(\max)}{I_{IL}(\max)} \quad \text{atau} \quad \text{fanout} = \frac{I_{OH}(\max)}{I_{IH}(\max)}$$

Gerbang-gerbang TTLstandar memiliki *fanout* sebesar 10. Ilustrasi *fanout* dapat ditunjukkan melalui gambar berikut ini.




**Gambar 190. Ilustrasi fanout dari suatu gerbang logika**

**Lampiran 7.**

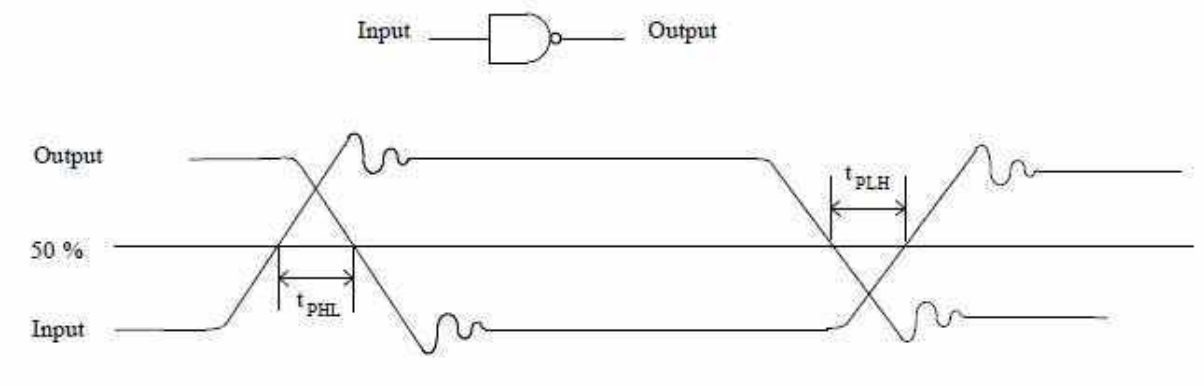
## Tunda Perambatan (*Propagation Delay*)

Tunda perambatan adalah waktu yang diperlukan oleh gerbang dalam mengubah level logika dari tinggi ke rendah atau sebaliknya. Tunda perambatan pada gerbang logika biasanya berbeda ketika berubah dari rendah ke tinggi dan dari tinggi ke rendah, sehingga terdapat dua jenis tunda perambatan yakni:

$t_{PLH}$: tunda perambatan ketika output berubah dari rendah ke tinggi

$t_{PHL}$: tunda perambatan ketika output berubah dari tinggi ke rendah

Ilustrasi kedua tunda perambatan tersebut ditunjukkan pada gambar berikut ini.

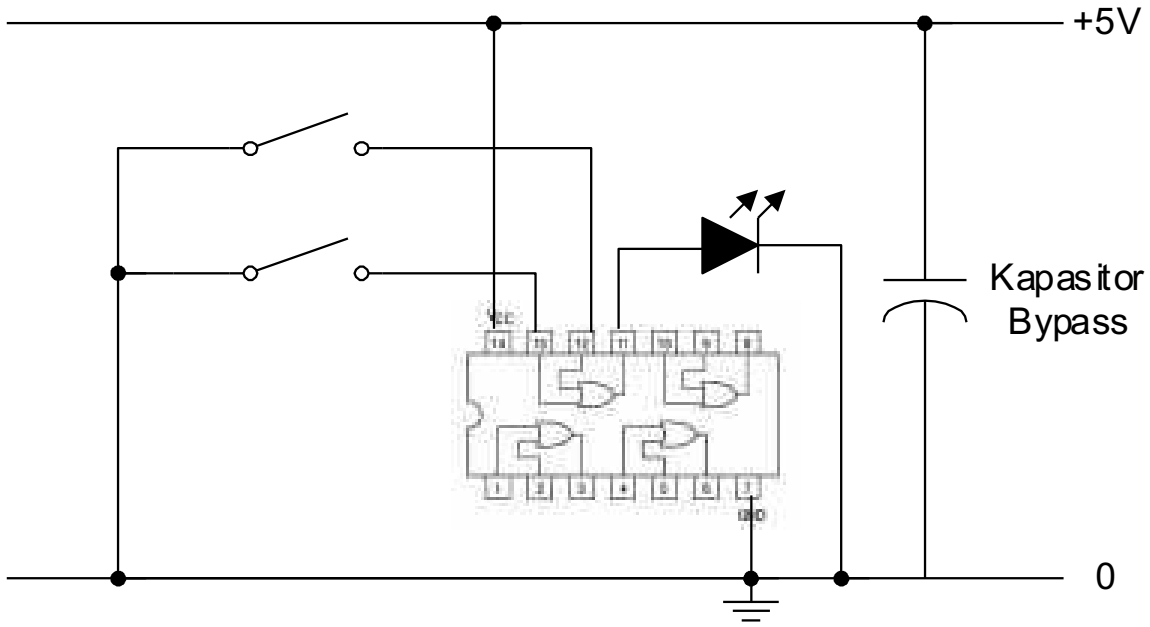



**Gambar 191. Ilustrasi Tunda Perambatan pada Gerbang Logika**

**Lampiran 8.**

## Kapasitor *By Pass* Pada Catu Daya

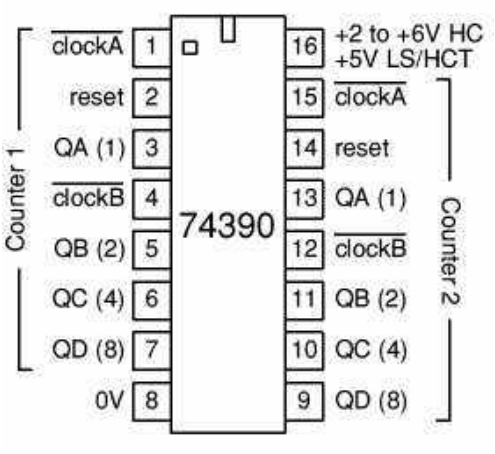
Dalam sebuah rangkaian memungkinkan munculnya *noise* atau tegangan-tegangan yang tidak dikehendaki masuk dalam inputnya. *Noise* dapat ditimbulkan karena induksi arus dari kabel-kabel penghubung maupun konduktor-konduktor pada papan rangkaian tercetak yang dapat berpengaruh pada level logika input sehingga menyebabkan kesalahan pada outputnya. Untuk meredam munculnya *noise* ini perlu dipasang kapasitor *bypass* sedekat mungkin pada kaki catudaya setiap IC logika yang digunakan.



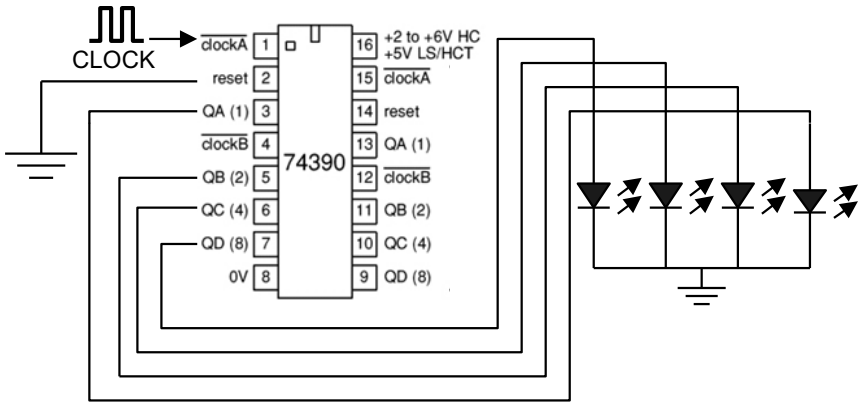

**Gambar 192. Contoh Pemasangan Kapasitor By Pass**

**Lampiran 9.**

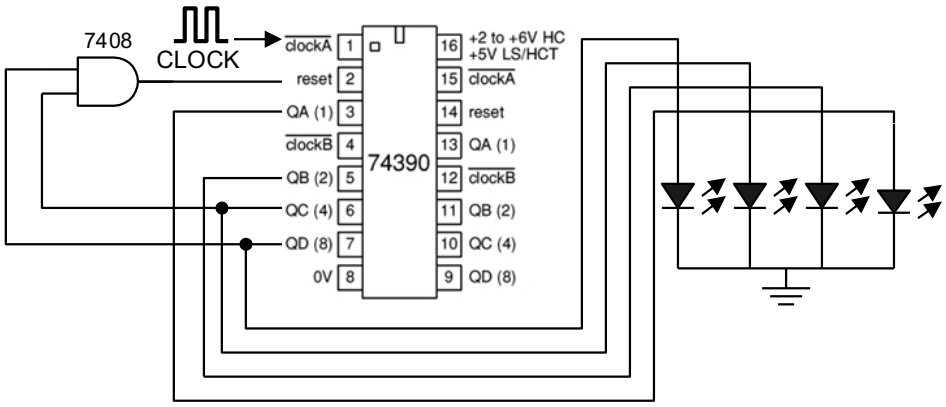
Pinout IC 74393:




Pencacah modulo-16 menggunakan IC 74393:

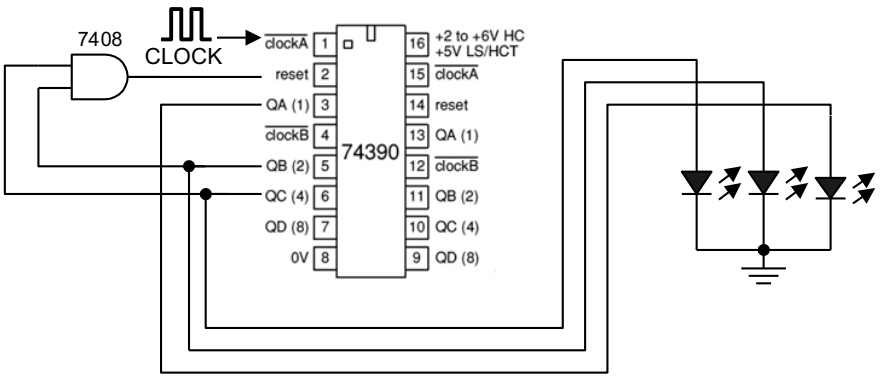



Pencacah modulo-12 menggunakan IC 74393:



**Lanjutan lampiran 9.**

Pencacah modulo-6 menggunakan IC 74393:




Rangkaian register geser (SIPO) 4-bit menggunakan IC 74164: