ELEKTRONIKA DIGITAL

Undang-Undang Republik Indonesia Nomor 28 Tahun 2014 tentang Hak Cipta

Fungsi dan sifat hak cipta Pasal 4
Hak Cipta sebagaimana dimaksud dalam Pasal 3 huruf a merupakan hak eksklusif yang terdiri atas hak moral dan hak ekonomi.
Fungsi dan sifat hak cipta Pasal 4
Ketentuan sebagaimana dimaksud dalam Pasal 23, Pasal 24, dan Pasal 25 tidak berlaku terhadap:

a. Penggunaan kutipan singkat Ciptaan dan/atau produk. Hak Terkait untuk pelaporan peristiwa aktual yang ditujukan hanya untuk keperluan penyediaan informasi aktual;
b. Penggandaan Ciptaan dan/atau produk Hak Terkait hanya untuk kepentingan penelitian ilmu pengetahuan;
c. Penggandaan Ciptaan dan/atau produk Hak Terkait hanya untuk keperluan pengajaran, kecuali pertunjukan dan Fonogram yang telah dilakukan Pengumuman sebagai bahan ajar; dan
d. Penggunaan untuk kepentingan pendidikan dan pengembangan ilmu pengetahuan yang memungkin kan suatu Ciptaan dan/atau produk Hak Terkait dapat digunakan tanpa izin Pelaku Pertunjukan, Produser Fonogram, atau Lembaga Penyiaran.

Sanksi Pelanggaran Pasal 113

1. Setiap Orang yang dengan tanpa hak melakukan pelanggaran hak ekonomi sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf i untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 1 (satu) tahun dan/atau pidana denda paling banyak Rp100.000.000 (seratus juta rupiah).
2. Setiap Orang yang dengan tanpa hak dan/atau tanpa izin Pencipta atau pemegang Hak Cipta melakukan pelanggaran hak ekonomi Pencipta sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf c, huruf d, huruf f, dan/atau huruf h untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 3 (tiga) tahun dan/atau pidana denda paling banyak Rp500.000.000,00 (lima ratus juta rupiah).

Bagus Haryadi

ELEKTRONIKA DIGITAL

WAWASANILMU

ELEKTRONIKA DIGITAL

Edisi Pertama

Cetakan Pertama: Oktober, 2023

Ukuran: 15,5 cm x 23 cm; Halaman: viii + 97

wi.2023.0315

Penulis:
Bagus Haryadi

Editor : *Wahyu Kurniawadi*
Cover : *Maulana Arifin*
Tata letak : *Dita Yuni Setiawati*

Penerbit
Wawasan Ilmu

Anggota IKAPI (215/JTE/2021)
Leler RT 002 RW 006 Desa Kaliwedi Kec. Kebasen Kab. Banyumas Jawa Tengah 53172
Email : naskah.wawasanilmu@gmail.com
Web : https://wawasanilmu.co.id/

ISBN :

PRAKATA

Selamat datang dalam Buku Ajar Elektronika Digital. Buku ini merupakan panduan komprehensif yang dirancang untuk membantu mahasiswa dalam mempelajari konsep dasar dan aplikasi praktis dalam bidang elektronika digital. Dalam era modern yang dipenuhi teknologi, pemahaman tentang komponen elektronika, sirkuit digital, dan gerbang logika menjadi sangat penting.

Buku ini disusun dengan tujuan memberikan penjelasan yang jelas dan terstruktur mengenai konsep-konsep dasar elektronika digital, mulai dari gerbang logika hingga desain rangkaian yang lebih kompleks. Kami percaya bahwa pembelajaran yang kuat dalam elektronika digital akan membekali pembaca dengan keterampilan yang esensial, baik untuk keperluan akademik maupun profesional.

Materi dalam buku ini disajikan dengan pendekatan yang mudah dipahami, tanpa mengorbankan keakuratan dan kedalaman konsep. Setiap bab dilengkapi dengan contoh kasus, ilustrasi, dan latihan yang dirancang untuk menguji pemahaman pembaca serta merangsang pemikiran kreatif dalam merancang dan menganalisis rangkaian digital.

Kepada pihak-pihak yang berkontribusi dalam penyusunan buku ini, kami ucapkan banyak terima kasih. Semoga agar Buku Ajar Elektronika Digital ini bisa menjadi panduan bagi para pembaca dalam memahami keilmuan di bidang elektronika digital. Selamat belajar, dan semoga buku ini menginspirasi perjalanan Anda dalam eksplorasi ilmu elektronika.

Penulis

DAFTAR ISI

BAB 1 SISTEM BILANGAN

A. Tujuan Pembelajaran

Pembelajaran tentang sistem bilangan membantu mahasiswa dalam memahami berbagai macam jenis bilangan yang digunakan dalam teknologi digital. Selain itu, sistem bilangan biner banyak digunakan dalam berbagai sirkuit elektronik dan komunikasi digital sehingga pembelajaran ini bisa membantu mahasiswa untuk memahami dasar-dasar kinerja perangkat elektronik.

B. Pendahuluan

Dalam dunia teknologi, sistem bilangan digunakan secara luas dalam komputasi dan pemrograman. Sistem ini meliputi berbagai bilangan antara laian: biner, oktal, desimal, dan heksadesimal yang biasa disebut sebagai "sistem bilangan basis multi". Dalam sistem ini, berbagai sistem bilangan digunakan untuk merepresentasikan nilai numerik dengan basis yang berbeda.

C. Sistem Bilangan Biner (Basis-2)

Untuk merepresentasikan nilai numerik dalam sistem bilangan biner digunakan basis 2. Dalam sistem ini, setiap angka direpresentasikan sebagai 0 atau 1. Sistem bilangan biner memiliki aplikasi utama dalam teknologi digital, karena sangat cocok dengan karakteristik dasar sirkuit elektronik yang memiliki dua keadaan diskrit: mati (0) dan hidup (1). Basis sistem bilangan biner adalah 2, yang berarti hanya ada dua simbol yang digunakan: 0 dan 1. Setiap angka dalam sistem bilangan biner disebut "digit biner" atau lebih umum dikenal sebagai "70" (singkatan dari "binary digit"). Bit merupakan unit terkecil dalam representasi bilangan biner.

Sistem bilangan biner sangat penting dalam representasi data digital. Dalam sirkuit elektronik, komponen seperti transistor dapat berada dalam dua keadaan, yang sesuai dengan 0 (*no current*) dan 1 (*current*). Karena hal inilah maka sistem bilangan biner digunakan untuk merepresentasikan data dalam berbagai perangkat elektronik

seperti: komputer, calculator, dll. Selain itu, operasi logika dalam elektronika digital seperti AND, OR, dan NOT dapat diartikan langsung dalam sistem bilangan biner. Berikut adalah contoh bilangan biner:

1. $0010\ 1000\ 0111_2$
2. $1000\ 0001\ 1111_2$
3. $1110\ 1110\ 1110\ 1110_2$
4. $1111\ 0000\ 1111\ 0000\ 1111\ 0000_2$

D. Sistem Bilangan Oktal (Basis-8)

Dalam sistem ini, basis 8 digunakan untuk merepresentasikan nilai numerik. Setiap angka dalam sistem oktal direpresentasikan sebagai angka dari 0 hingga 7. Sistem bilangan oktal juga biasa digunakan dalam konteks komputer dan elektronika digital.

Basis sistem bilangan oktal adalah 8, yang berarti hanya ada 8 bilangan (0 s.d 7). Setiap angka dalam sistem bilangan oktal disebut "digit oktal". Setiap digit oktal merepresentasikan angka dalam basis 8. Misalnya, dalam angka oktal 745, digit pertama mewakili nilai 7, digit kedua mewakili nilai 4, dan digit ketiga mewakili nilai 5.

Contoh bilangan oktal:

1. 745_8
2. 432156_8
3. 1234_8
4. 55555_8

E. Sistem Bilangan Desimal (Basis-10)

Sistem ini sering disebut dengan istilah sitem basis-10 yang merupakan sistem bilangan yang paling umum digunakan dalam pemrosesan informasi manusia. Dalam sistem bilangan desimal, setiap angka direpresentasikan sebagai angka dari 0 hingga 9. Ini sangat sesuai dengan cara kita menghitung, mengukur, dan berinteraksi dengan alam.

Walaupun dasar sirkuit elektronik adalah logika biner (0 dan 1), pemrosesan data dalam kebanyakan komputer dan perangkat elektronik seringkali dilakukan dalam sistem bilangan desimal. Ini karena sistem bilangan desimal lebih intuitif bagi manusia, yang membuatnya lebih mudah untuk berinteraksi dengan teknologi.

Dalam komputasi, perangkat keras seperti CPU (Central Processing Unit) dapat memproses instruksi dan melakukan perhitungan menggunakan operasi matematika dalam sistem bilangan desimal. Pemrograman komputer juga sering menggunakan representasi desimal untuk merepresentasikan data, menghitung, dan memproses hasil. Misalnya, angka dalam kode program biasanya ditulis dalam sistem bilangan desimal.

Contoh bilangan desimal:

1. 272727_{10}
2. 89898989_{10}
3. 234523452345_{10}

F. Sistem Bilangan heksadesimal (Basis-16)

Sistem ini dikenal sebagai sistem bilangan basis-16 yang emnngunakan 16 angka untuk merepresentasikan nilai numerik. Sistem ini menggabungkan angka 0-9 dengan huruf A-F untuk merepresentasikan nilai dalam bentuk yang lebih ringkas dan mudah diinterpretasikan. Sistem bilangan heksadesimal memiliki berbagai aplikasi dalam dunia teknologi, khususnya dalam pemrograman, desain sirkuit, dan komunikasi data.

Sistem bilangan heksadesimal sering digunakan dalam komputer dan teknologi karena kemampuannya untuk merepresentasikan nilai biner yang lebih panjang dengan lebih efisien. Misalnya, dua digit heksadesimal dapat merepresentasikan 8 bit (1 byte) dalam biner. Konversi dari biner ke heksadesimal atau sebaliknya melibatkan pengelompokan atau pemecahan nilai biner ke dalam kelompok empat digit heksadesimal, yang membuatnya lebih mudah diinterpretasikan.

Contoh bilangan heksadesimal:

1. $5678AB_{16}$
2. $AB1960CDEF_{16}$
3. $5115AABBCCDD_{16}$

G. Binary Coded Decimal (BCD)

BCD merupakan representasi numerik dimana setiap kombinasi binernya mewakili setiap digit desimalnya (0-9). Dalam representasi ini, setiap digit desimal dipecah menjadi empat bit biner terpisah, yang masing-masing mewakili angka dari 0 hingga 9. BCD adalah cara untuk menyimpan angka desimal dalam bentuk biner tanpa perlu mengkonversi sepenuhnya menjadi biner.

Dalam BCD, setiap digit desimal diwakili oleh empat bit, sehingga digit-desimal tunggal akan menggunakan empat bit. Misalnya, angka 5 diwakili oleh 0101 dalam BCD. Namun, BCD memiliki pembatasan tertentu dan tidak mengambil keuntungan penuh dari representasi biner.

BCD umumnya digunakan dalam aplikasi di mana perhitungan dan representasi numerik penting, seperti pada perangkat tampilan tujuh segmen yang digunakan dalam jam digital, alat pengukur, panel kontrol, dan lain sebagainya. Kelebihan BCD adalah kemampuannya untuk mewakili angka desimal dengan cara yang lebih intuitif dan langsung dibandingkan dengan representasi biner

tradisional. Namun, penggunaan BCD juga dapat memerlukan lebih banyak ruang penyimpanan dibandingkan dengan representasi biner murni.

Tabel BCD adalah tabel yang memperlihatkan hubungan antara angka desimal dan representasi biner BCD yang sesuai. Dalam BCD, setiap angka desimal dipecah menjadi empat bit biner yang merepresentasikan digit-desimal tersebut. Berikut adalah tabel BCD untuk angka 0 hingga 9:

Angka Desimal	Representasi BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Dalam tabel ini, setiap angka desimal memiliki representasi BCD yang terdiri dari empat bit biner. Misalnya, angka 5 memiliki representasi BCD 0101, dan angka 9 memiliki representasi BCD 1001.

Tabel BCD digunakan sebagai referensi saat mengkonversi antara angka desimal dan representasi BCD atau saat bekerja dengan perangkat yang menggunakan representasi BCD, seperti tampilan tujuh segmen pada perangkat digital.

H. Konversi Bilangan Digital

Konversi bilangan digital merupakan langkah-langkah untuk mengubah representasi suatu bilangan. Setiap sistem bilangan memiliki cara tersendiri untuk diinterpretasikan dan dihitung. Konversi antar sistem bilangan ini penting dalam pemrograman, desain sirkuit, dan analisis data.

1. Konversi bilangan basis-2 ke bilangan basis-8

Proses konversi ini melibatkan pengubahan representasi bilangan biner menjadi bilangan basis-8. Konversi ini melibatkan pengelompokan digit-digit biner ke dalam kelompok 3 digit dan mengubah setiap kelompok menjadi digit oktal yang sesuai. Berikut adalah langkah-langkah untuk mengkonversi bilangan biner ke oktal:

a. **Tentukan Nilai**: Mulailah dengan bilangan biner yang akan diubah ke oktal.

b. **Tambah Digit Nol**: Jika jumlah digit dalam bilangan biner tidak habis dibagi tiga, tambahkan digit 0 di depan bilangan biner hingga jumlah digit habis dibagi tiga. Ini dilakukan untuk memastikan bahwa setiap kelompok memiliki tiga digit.

c. **Kelompokkan Digit**: Kelompokkan digit-digit biner ke dalam kelompok tiga digit, mulai dari digit paling kanan.

d. **Tentukan Nilai untuk Setiap Kelompok**: Konversi setiap kelompok tiga digit biner ke nilai desimal. Kemudian, ubah nilai desimal tersebut ke dalam digit oktal yang sesuai.

Kelompok Biner	Nilai Desimal	Nilai Oktal
000	0	0
001	1	1

010	2	2
011	3	3
100	4	4
101	5	5
110	6	6
111	7	7

e. Gabungkan Nilai Oktal: Gabungkan digit oktal dari setiap kelompok untuk mendapatkan bilangan oktal akhir.

Contoh: konversikan bilangan biner 1101001_2 ke dalam bilangan octal.

Cara mengerjakan:

1. Tentukan nilai bilangan yang akan diubah: 1101001
2. Tambahkan digit nol di depan bilangan biner: 001101001
3. Kelompokkan digit dalam kelompok tiga digit: 001 101 001
4. Konversi setiap kelompok ke nilai oktal: 1 3 1
5. Gabungan nilai oktal=131

Jadi, bilangan biner 110101_2 setara dengan bilangan oktal 131_8.

2. Konversi bilangan basis-2 ke bilangan basis-10

Proses konversi ini adalah mengubah representasi bilangan basis-2 ke desimal. Setiap digit dalam bilangan biner mewakili pemangkatan 2. Berikut adalah langkah-langkah pengkonversiannya:

a. Tentukan Nilai: Tentukan bilangan binernya.

b. Tentukan Panjang Bilangan: Hitung panjang bilangan biner (jumlah digit).

c. **Mulai dari Kiri ke Kanan**: Mulai dari digit paling kiri (digit tertinggi) dan bergerak ke kanan, tetapkan pangkat dua yang sesuai dengan posisi digit.

d. **Hitung Nilai Desimal**: Untuk setiap digit biner, kalikan digit tersebut dengan 2^pangkat dan jumlahkan hasilnya. Pangkat dimulai dari 0 untuk digit terkanan dan meningkat seiring perpindahan ke kiri.

e. **Total Nilai**: Jumlahkan semua hasil dari langkah sebelumnya untuk mendapatkan nilai desimal akhir.

Contoh: konversikan bilangan basis-2 ke bilangan basis-10 berikut ini: 1101_2

Cara mengerjakan:

1. Tentukan nilai yang akan diubah: 1101
2. Menghitung panjang digit bilangan biner : 4
3. Menentukan pangkat 2 dari setiap digit biner:

   1 * 2^3 (pangkat 3) = 8

   1 * 2^2 (pangkat 2) = 4

   0 * 2^1 (pangkat 1) = 0

   1 * 2^0 (pangkat 0) = 1
4. Menghitung nilai desimal = 8 + 4 + 0 + 1 = 13
5. Total nilai = 13

Jadi, bilangan biner 1101_2 setara dengan bilangan desimal 13_{10}.

Metode ini dapat digunakan untuk mengkonversi bilangan biner apa pun menjadi bilangan desimal. Proses ini digunakan untuk menghitung nilai desimal dari masing-masing digit biner sesuai dengan pangkat dua yang sesuai dengan posisinya dalam bilangan.

3. Konversi bilangan basis-2 ke bilangan basis-16

Proses dalam konversi ini melibatkan mengubah representasi bilangan biner ke basis-16. Konversi ini melibatkan pengelompokan digit-digit biner ke dalam kelompok 4 digit dan mengubah setiap kelompok menjadi digit heksadesimal yang sesuai. Berikut adalah langkah-langkah untuk mengkonversi bilangan biner ke basis-16:

a. **Tentukan Nilai**: Mulailah dengan bilangan yang akan diubah.

b. **Tambah Digit Nol**: Jika jumlah digit dalam bilangan biner tidak habis dibagi 4, tambahkan digit 0 di depan bilangan biner hingga jumlah digit habis dibagi 4. Ini dilakukan untuk memastikan bahwa setiap kelompok memiliki 4 digit.

c. **Kelompokkan Digit**: Kelompokkan digit-digit bilangan basis-2 ke dalam kelompok 4 digit dari digit paling kanan.

d. **Tentukan Nilai untuk Setiap Kelompok**: Konversi setiap kelompok 4 digit biner ke nilai desimal. Kemudian, ubah nilai desimal tersebut ke dalam digit heksadesimal yang sesuai.

Kelompok Biner	Nilai Desimal	Nilai Heksadesimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8

1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

4. Gabungkan Nilai Heksadesimal

Gabungkan digit heksadesimal dari setiap kelompok untuk mendapatkan bilangan heksadesimal akhir.

Contoh: konversikan bilangan basis-2 berikut ini ke bilangan heksadesimal: 110101110_2

Cara mengerjakan:

a. Tentukan nilai bilangan biner yang akan diubah: 110101110

b. Tambahkan digit nol di depan bilangan biner: 0011 0101 1100.

c. Konversi setiap kelompok empat digit ke nilai heksadesimal: 0011= 3, 0101= 5, 1100= C.

d. Tentukan nilai setiap kelompok: 3 5 C

e. Gabungkan nilai heksadesimal: 35C

Jadi, bilangan biner 110101110_2 setara dengan bilangan heksadesimal $35C_{16}$.

5. Konversi bilangan basis-8 ke bilangan basis-2

Proses konversi ini melibatkan pengubahan representasi

bilangan dari basis-8 menjadi basis-2 dimana dalam setiap digit oktalnha mewakili 3 digit biner yang sesuai. Berikut adalah langkah-langkah untuk mengkonversi bilangan oktal ke biner:

a. **Tentukan Nilai**: Mulailah dengan bilangan oktal yang akan diubah ke biner.

b. **Konversi Setiap Digit**: Ubah setiap digit oktal menjadi representasi biner yang sesuai, dengan tiga digit biner untuk setiap digit oktal.

c. **Gabungkan Representasi Biner**: Gabungkan representasi biner dari setiap digit oktal untuk mendapatkan bilangan biner akhir.

Contoh: Konversikan bilangan basis-8 berikut ke dalam bilangan basis-2: 745_8

Cara mengerjakan:

1. Tentukan nilai yang akan dikonversi: 745
2. Konversikan setiap digit oktal ke dalam representasi biner:
   - Digit 7: 111
   - Digit 4: 100
   - Digit 5: 101
3. Gabungkan representasi biner: 111100101

Jadi, bilangan oktal 745_8 setara dengan bilangan biner 111100101_2.

6. Konversi bilangan basis-8 ke bilangan basis-10

Proses konversi ini melibatkan mengubah representasi bilangan dari basis-8 menjadi basis-10. Setiap digit dalam bilangan oktal mewakili kekuasaan 8 yang meningkat dengan pangkat. Berikut adalah langkah-langkah untuk mengkonversi bilangan oktal ke desimal:

a. **Tentukan Nilai**: Mulailah dengan bilangan oktal yang

akan diubah ke desimal.

b. **Tentukan Panjang Bilangan**: Hitung panjang bilangan oktal (jumlah digit).

c. **Mulai dari Kiri ke Kanan**: Mulai dari digit paling kiri (digit tertinggi) dan bergerak ke kanan, tetapkan pangkat kekuasaan delapan yang sesuai dengan posisi digit.

d. **Hitung Nilai Desimal**: Untuk setiap digit oktal, kalikan digit tersebut dengan 8 pangkat dan jumlahkan hasilnya. Pangkat dimulai dari 0 untuk digit terkanan dan meningkat seiring perpindahan ke kiri.

e. **Total Nilai**: Jumlahkan semua hasil dari langkah sebelumnya untuk mendapatkan nilai desimal akhir.

Contoh: konversikan bilangan oktal 645_8 ke dalam bilangan basis-10:

Cara mengerjakan:

1. Tentukan nilai yang akan diubah: 546_8
2. 5 * 8^0 (pangkat 0) = 5
3. 4 * 8^1 (pangkat 1) = 32
4. 6 * 8^2 (pangkat 2) = 384
5. Total nilai desimal = 5 + 32 + 384 = 421

Jadi, bilangan oktal 645_8 setara dengan bilangan desimal 421_{10}.

7. Konversi bilangan basis-8 ke bilangan basis-16

Proses konversi bilangan ini melibatkan pengubahan representasi bilangan basis-8 ke dalam bilangan basis-16. Berikut adalah langkah-langkah untuk mengkonversi bilangan oktal ke heksadesimal:

a. **Tentukan Nilai**: Mulailah dengan bilangan oktal yang akan diubah ke heksadesimal.

b. **Konversi ke Biner**: Ubah setiap digit oktal menjadi representasi biner yang sesuai. Misalnya, ubah digit 3 menjadi 011, digit 4 menjadi 100, dan seterusnya.

c. **Gabungkan Representasi Biner**: Gabungkan semua representasi biner dari setiap digit oktal untuk mendapatkan bilangan biner akhir.

d. **Konversi Biner ke Heksadesimal**: Setelah mendapatkan bilangan biner akhir, kelompokkan digit biner menjadi kelompok 4 digit, lalu ubah setiap kelompok menjadi digit heksadesimal yang sesuai.

e. **Gabungkan Nilai Heksadesimal**: Gabungkan digit heksadesimal dari setiap kelompok untuk mendapatkan bilangan heksadesimal akhir.

Contoh: konversikan bilangan octal berikut ke dalam bilangan heksadesimal: 545_8

Cara mengerjakan:

1. Tentukan bilangan yang akan dikonversi: 545_8
2. Konversi setiap digit oktal ke representasi biner:
   - Digit 5: 101
   - Digit 4: 100
   - Digit 5: 101
3. Gabungkan representasi biner: 101100101
4. Konversi bilangan biner ke bilangan heksadesimal:
   - Kelompok biner: 1011 0010 1
   - Konversi ke heksadesimal: B 2 1
5. Gabungkan nilai heksadesimal: B21

Jadi, bilangan oktal 545_8 setara dengan bilangan heksadesimal $B21_{16}$.

8. Konversi bilangan basis-10 ke bilangan basis-2

Konversi basis-10 ke biner melibatkan pengubahan representasi bilangan dari basis-10 menjadi basis-2. Proses ini melibatkan pembagian bilangan basis-10 dengan 2 secara berulang dan mencatat sisa hasil bagi dari setiap pembagian. Berikut adalah langkah-langkah untuk mengkonversi bilangan desimal ke basis-2:

a. **Tentukan Nilai**: Mulailah dengan bilangan desimal yang akan diubah ke biner.

b. **Pembagian dan Pencatatan Sisa**: Bagi bilangan desimal dengan 2 dan catat sisa hasil bagi. Lanjutkan proses ini hingga hasil bagi menjadi 0. Pencatatan sisanya dari bawah ke atas akan memberikan representasi biner yang benar.

c. **Catat Hasil**: Catat semua sisa hasil bagi dari langkah sebelumnya untuk mendapatkan representasi biner.

d. **Balik Urutan Sisa**: Balik urutan sisa hasil bagi yang telah dicatat pada langkah sebelumnya untuk mendapatkan representasi biner yang benar.

Contoh: konversikan bilangan basis-10 berikut ini ke bilangan biner: 13_{10}

Cara mengerjakan:

1. Tentukan nilai bilangan yang akan diubah: 13
2. - Bagi 13 dengan 2: 13 ÷ 2 = 6, sisa = 1
   - Bagi 6 dengan 2 : 6 ÷ 2 = 3, sisa = 0
   - Bagi 3 dengan 2 : 3 ÷ 2 = 1, sisa = 1
   - Bagi 1 dengan 2 : 1 ÷ 2 = 0, sisa = 1
3. Catatan sisa hasil bagi: 1101
4. Balik urutan sisa: 1101

Jadi, bilangan desimal 13_{10} setara dengan bilangan biner 1101_2.

9. Konversi bilangan basis-10 ke bilangan basis-8

Proses konversi ini melibatkan mengubah representasi bilangan dari basis-10 ke dalam bilangan basis-8. Berikut adalah langkah-langkah untuk mengkonversi bilangan desimal ke bilangan oktal:

a. Tentukan Nilai: Mulailah dengan bilangan desimal yang akan diubah ke oktal.

b. Pembagian dan Pencatatan Sisa: Bagi bilangan desimal dengan 8 dan catat sisa hasil bagi. Lanjutkan proses ini hingga hasil bagi menjadi 0. Pencatatan sisanya dari bawah ke atas akan memberikan representasi oktal yang benar.

c. Catat Hasil: Catat semua sisa hasil bagi dari langkah sebelumnya untuk mendapatkan representasi oktal.

d. Balik Urutan Sisa: Balik urutan sisa hasil bagi yang telah dicatat pada langkah sebelumnya untuk mendapatkan representasi oktal yang benar.

Contoh: konversikan bilangan desimal 45_{10} ke bilangan oktal:

1. Tentukan nilainya: 45
2. Bagi 45 dengan 8: 45 ÷ 8 = 5, sisa = 5
3. Bagi 5 dengan 8: 5 ÷ 8 = 0, sisa = 5

   Catatan sisa hasil bagi: 55
4. Balik urutan sisa: 55

Jadi, bilangan desimal 45_{10} setara dengan bilangan oktal 55_8.

10. Konversi bilangan basis-10 ke bilangan basis-16

Proses konversi ini melibatkan pengubahan representasi

bilangan dari basis-10 menjadi bilangan basis-16. Selain itu, representasi huruf A-F diperlukan untuk mewakili angka 10 hingga 15. Berikut adalah langkah-langkah untuk mengkonversi bilangan basis-10 ke bilangan heksadesimal:

a. Tentukan Nilai: Mulailah dengan bilangan desimal yang akan diubah ke heksadesimal.

b. Pembagian dan Pencatatan Sisa: Bagi bilangan basis-10 dengan 16 dan catat sisa hasil baginya. Lanjutkan proses ini hingga hasilnya menjadi 0. Pencatatan sisanya dari bawah ke atas akan memberikan representasi heksadesimal yang benar.

c. Catat Hasil: Catat semua sisa hasil bagi dari langkah sebelumnya. Jika hasil bagi adalah 10, 11, 12, 13, 14, atau 15, gantilah angka tersebut dengan huruf A sampai F, sesuai urutan.

d. Balik Urutan Sisa: Balik urutan sisa hasil bagi yang telah dicatat pada langkah sebelumnya untuk mendapatkan representasi heksadesimal yang benar.

Contoh: konversikan bilangan desimal 255_{10} ke dalam bilangan basis-16:

Cara mengerjakan:

1. Tentukan nilainya: 255
2. Bagi 255 dengan 16: 255 ÷ 16 = 15, sisa = 15
3. Bagi 15 dengan 16: 15 ÷ 16 = 0, sisa = 15

   Catatan sisa hasil bagi: FF
4. Balik urutan sisa: FF

Jadi, bilangan desimal 255_{10} setara dengan bilangan heksadesimal FF_{16}.

11. Konversi bilangan basis-16 ke bilangan basis-2

Proses konversi ini melibatkan pengubahan representasi

bilangan dari basis-16 menjadi basis-2. Proses ini mengubah setiap digit heksadesimal menjadi representasi biner yang sesuai.

Berikut adalah langkah-langkah untuk mengkonversi bilangan basis-16 ke bilangan biner:

a. **Tentukan Nilai**: Mulailah dengan bilangan heksadesimal yang akan diubah ke biner.

b. **Konversi Setiap Digit**: Ubah setiap digit heksadesimal menjadi representasi biner yang sesuai. Misalnya, ubah digit 0 menjadi 0000, digit 1 menjadi 0001, dan seterusnya hingga digit F menjadi 1111.

c. **Gabungkan Representasi Biner**: Gabungkan representasi biner dari setiap digit heksadesimal untuk mendapatkan bilangan basis-2 akhir.

Contoh: konversikan bilangan heksadesimal $B3_{16}$ ke bilangan biner:

Cara mengerjakan:

1. Tentukan nilai yang akan diubah: B3
2. Konversi setiap digit heksadesimal ke representasi biner:
   - Digit B: 1011
   - Digit 3: 0011
3. Gabungkan representasi biner: 10110011

Jadi, bilangan heksadesimal $B3_{16}$ setara dengan bilangan biner 10110011_2.

12. Konversi bilangan basis-16 ke bilangan basis-8

Proses konversi ini melibatkan pengubahan representasi bilangan dari basis-16 ke dalam bilangan basis-8. Proses ini melibatkan dua tahap: pertama, mengkonversi bilangan heksadesimal menjadi bilangan biner, dan kedua, mengkonversi bilangan biner tersebut ke bilangan oktal. Berikut adalah

langkah-langkah untuk mengkonversi bilangan heksadesimal ke bilangan oktal:

a. **Tentukan Nilai**: Mulailah dengan bilangan heksadesimal yang akan diubah ke oktal.

b. **Konversi ke Biner**: Ubah setiap digit heksadesimal menjadi representasi biner yang sesuai. Misalnya, ubah digit 0 menjadi 0000, digit 1 menjadi 0001, dan seterusnya hingga digit F menjadi 1111.

c. **Gabungkan Representasi Biner**: Gabungkan representasi biner dari setiap digit heksadesimal untuk mendapatkan bilangan biner akhir.

d. **Konversi Biner ke Oktal:** Setelah mendapatkan bilangan biner akhir, konversikan bilangan basis-2 ke basis-8 dengan cara mengelompokkan digit biner menjadi kelompok tiga digit, lalu ubah setiap kelompok menjadi digit oktal yang sesuai.

e. **Gabungkan Nilai Oktal**: Gabungkan digit oktal dari setiap kelompok untuk mendapatkan bilangan oktal akhir.

Contoh: konversikan bilangan heksadesimal B3A ke bilangan basis-8:

Cara mengerjakan:

1. Tentukan nilainya: B3A
2. Konversi setiap digit heksadesimal ke representasi biner:
   - Digit B: 1011
   - Digit 3: 0011
   - Digit A: 1010
3. Gabungkan representasi biner: 101100111010
4. Konversi bilangan biner ke bilangan oktal:
   - Kelompok biner: 101 100 111 010

- Konversi ke oktal: 5 4 7 2

5. Gabungkan nilai oktal: 5472

Jadi, bilangan heksadesimal $B3A_{16}$ setara dengan bilangan oktal 5472_8.

13. Konversi heksadesimal ke decimal

Proses konversi ini melibatkan pengubahan representasi bilangan basis-16 menjadi basis-10. Setiap digit heksadesimal harus dikonversi menjadi nilai desimal yang sesuai, dengan mempertimbangkan angka 10 hingga 15 yang direpresentasikan oleh huruf A hingga F. Berikut adalah langkah-langkah untuk mengkonversi bilangan heksadesimal ke bilangan desimal:

a. **Tentukan Nilai**: Mulailah dengan bilangan heksadesimal yang akan diubah ke desimal.

b. **Konversi Setiap Digit**: Ubah setiap digit heksadesimal menjadi nilai desimal yang sesuai.

c. **Hitung Nilai Desimal**: Hitung nilai desimalnya yang sesuai.

Contoh: konversikan bilangan heksadesimal $1A3_{16}$ ke bilangan basis-10

Cara mengerjakan:

1. Tentukan nilainya: 1A3
2. Konversi setiap digit heksadesimal menjadi nilai desimal:
   - Digit 1: 1
   - Digit A: 10
   - Digit 3: 3
3. Hitung nilai desimal: (1 * 16^2) + (10 * 16^1) + (3 * 16^0) = 256 + 160 + 3 = 419

Jadi, bilangan heksadesimal $1A3_{16}$ setara dengan bilangan desimal 419_{10}.

I. Evaluasi / Soal Latihan

1. Konversikan bilangan biner 101101 ke desimal.
2. Konversikan bilangan oktal 34 ke heksadesimal.
3. Konversikan bilangan heksadesimal 1A ke biner.
4. Konversikan bilangan desimal 147 ke oktal.
5. Konversikan bilangan BCD 0101 1001 ke desimal.
6. Konversikan bilangan oktal 63 ke desimal.
7. Konversikan bilangan heksadesimal 2E ke BCD.
8. Konversikan bilangan heksadesimal FF ke desimal..

BAB 2
GERBANG LOGIKA

A. Tujuan Pembelajaran

Mahasiswa bisa mengetahui berbagai macam simbol gerbang logika dan kegunaan dalam rangkaian. Selain itu, mahasiswa diharapkan bisa menjelaskan konsep gerbang logika, tabel kebenaran dan aplikasinya.

B. Pendahuluan

Dalam dunia modern yang penuh dengan perangkat elektronik yang canggih, kita seringkali melihat berbagai jenis perangkat, mulai dari komputer hingga ponsel pintar. Di balik kemampuan dan kompleksitas perangkat tersebut, ada dasar-dasar yang memungkinkan semuanya berfungsi dengan cara yang terkoordinasi dan terprediksi. Salah satu fondasi utama dari semua ini adalah apa yang dikenal sebagai "gerbang logika."

Gerbang logika merupakan komponen pembangun dari sirkuit-sirkuit yang memungkinkan perangkat elektronik untuk melakukan operasi matematika dan logika yang kompleks. Gerbang logika memungkinkan kita untuk memanipulasi sinyal digital, yang merupakan representasi dari angka biner (0 dan 1) yang mendasari semua komputasi modern.

Dalam materi ini akan dipelajari dasar-dasar gerbang logika. Kita akan memahami apa itu gerbang logika? mengapa mereka penting?? dan bagaimana mereka digunakan untuk membangun sirkuit-sirkuit yang lebih kompleks? Selain itu, juga akan dibahas berbagai macam gerbang logika dasar sebagai komponen dasar untuk membangun rangkaian logika serta bagaimana interaksi gerbang satu dengan gerbang lain bisa melakukan operasi logika yang lebih kompleks.

Penting untuk diingat bahwa walaupun gerbang logika mungkin terlihat sederhana pada permukaan, peran dan kontribusi mereka dalam dunia teknologi sangatlah besar. Mereka adalah "batu bata"

yang membentuk dasar semua perangkat digital di sekitar kita. Oleh karena itu, memahami konsep dasar gerbang logika adalah langkah pertama yang sangat penting dalam memahami dunia elektronika digital yang semakin berkembang pesat.

Mari kita memulai perjalanan kita dalam dunia gerbang logika, memahami bagaimana sinyal digital diubah dan diproses, serta mengungkap potensi luar biasa dari dasar-dasar ini dalam membentuk teknologi yang menggerakkan dunia modern..

C. Gerbang NOT

Gerbang NOT juga dikenal sebagai "gerbang inverter." Fungsinya adalah untuk membalikkan (invert) status sinyal masukan.

Dalam konteks digital, sinyal umumnya direpresentasikan dalam bentuk bit 0 (tidak ada tegangan) dan bit 1 (tegangan ada). Jadi, jika sinyal masukan ke gerbang NOT adalah 0, maka keluaran akan menjadi 1, dan sebaliknya. Simbol gerbang NOT biasanya ditunjukkan dengan tanda panah yang mengarah keluar dari lingkaran.

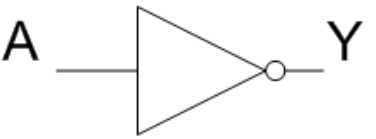
Misalnya:

- Masukan: 0
- Keluaran: 1

Atau:

- Masukan: 1
- Keluaran: 0

Gerbang NOT merupakan gerbang yang populer digunakan dalam rangkaian elektronika untuk mengubah atau menginversi status sinyal sesuai dengan kebutuhan perancangan



Gambar 2.1. Simbol gerbang logika NOT

Tabel 2.1. Tabel kebenaran untuk gerbang logika NOT

Masukan	Keluaran
0	1
1	0

Berikut ini adalah contoh-contoh Gerbang logika NOT yang ditanam dalam sebuah Integrated Circuit (IC):

1. **IC 7407:** IC ini merupakan gerbang NOT buffer dengan output open-collector. Ini memungkinkan Anda untuk memperluas jumlah sirkuit output dengan menghubungkan beberapa gerbang pada rangkaian yang sama.
2. **IC 4049:** Meskipun tidak secara eksklusif gerbang NOT, IC ini mengandung enam gerbang invert (NOT) yang dapat digunakan dalam berbagai aplikasi.
3. **IC 4049UB:** Ini adalah variasi dari IC 4049 yang memiliki output dengan tipe CMOS (Complementary Metal-Oxide-Semiconductor), yang berarti memiliki karakteristik daya yang lebih rendah daripada IC logika TTL.
4. **IC 4069:** Serupa dengan IC 4049, IC 4069 juga memiliki enam gerbang NOT invert di dalamnya.

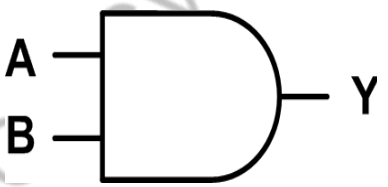
5. **IC 74LS04:** Ini adalah IC seri TTL (Low-power Schottky Transistor-Transistor Logic) yang memiliki enam gerbang NOT dalam satu paket.

6. **IC 74HC04:** Ini adalah IC jenis High-Speed CMOS (HCMOS) yang memiliki enam gerbang NOT dalam satu paket. Ia bekerja pada tegangan rendah dan memiliki kecepatan lebih tinggi dibandingkan dengan beberapa seri lainnya.

D. Gerbang AND

Gerbang AND memiliki 2 atau lebih masukan dan memiliki 1 keluaran. Fungsinya adalah untuk menghasilkan keluaran hanya ketika semua inputnya adalah logika "1" (tinggi).

Secara simbolik, gerbang AND direpresentasikan dengan simbol "Λ" atau tanda ".". Simbol "Λ" menggambarkan dua input yang masuk ke atas, dan garis keluaran keluar dari bawah.



Gambar 2.2. Simbol gerbang AND

Tabel kebenaran untuk gerbang AND dengan dua masukan sebagai berikut:

Tabel 2.2. Tabel kebenaran gerbang AND

A	B	Keluaran
0	0	0
0	1	0

1	0	0
1	1	1

Fungsi gerbang AND berperan dalam menghasilkan keluaran "1" hanya ketika semua inputnya adalah "1" (tinggi). Jika salah satu atau semua input adalah "0" (rendah), maka keluaran akan menjadi "0". Ini mencerminkan operasi "dan" dalam logika boolean, di mana keluaran akan "benar" hanya jika semua kondisi adalah "benar".

Contoh sederhana dari penggunaan gerbang AND dalam kehidupan sehari-hari adalah pada pengaturan *traffic light*. Bayangkan sebuah *traffic light* yang dikendalikan oleh rangkaian elektronika menggunakan gerbang AND untuk mengontrol warna lampu hijau. Mari kita katakan bahwa lampu hijau harus menyala hanya ketika kedua kondisi berikut terpenuhi: sinyal untuk jalan utama hijau (A) dan sinyal untuk jalan samping hijau (B).

- Input A: Jika sinyal untuk jalan utama hijau adalah "1" (aktif).
- Input B: Jika sinyal untuk jalan samping hijau adalah "1" (aktif).

Jadi, kita dapat menggunakan gerbang AND untuk menghasilkan keluaran yang akan menghidupkan lampu hijau hanya jika kedua kondisi tersebut terpenuhi. Jika salah satu atau kedua masukan adalah "0", maka keluaran gerbang AND akan menjadi "0," dan lampu hijau akan mati.

Gerbang AND juga digunakan dalam pembentukan rangkaian logika yang lebih kompleks, seperti pembuatan pintu gerbang logika lainnya seperti NAND, NOR, dan XOR. Kombinasi gerbang-gembang ini memungkinkan untuk membangun sirkuit-sirkuit yang dapat melakukan berbagai tugas logika dan perhitungan matematis.

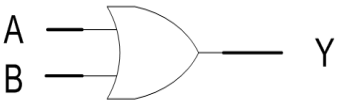
Dalam perancangan rangkaian digital, gerbang AND adalah salah satu dasar yang penting. Mempelajari bagaimana gerbang ini bekerja dan bagaimana mereka dapat digunakan dalam

berbagai aplikasi adalah langkah pertama dalam memahami dunia elektronika digital yang lebih luas.

E. Gerbang OR

Gerbang OR berfungsi untuk menghasilkan keluaran "1" jika setidaknya salah satu dari masukannya adalah "1" (tinggi). Dengan kata lain, gerbang OR melakukan operasi "atau" dalam logika boolean.

Simbol grafis yang sering digunakan untuk gerbang OR adalah "+" atau tanda "v" di dalam lingkaran. Simbol ini menggambarkan dua input yang masuk dari atas, dan garis keluaran keluar dari bawah.



$$Y = A + B$$

Gambar 2.3. Simbol gerbang logika OR

Berikut ini adalah tabel kebenaran dari gerbang OR dengan 2 masukan:

Tabel 2.3. Tabel kebenaran gerbang OR

A	B	Keluaran
0	0	0
0	1	1

1	0	1
1	1	1

Dalam tabel kebenaran di atas, keluaran akan menjadi "1" jika setidaknya salah satu dari input A atau B adalah "1". Hanya jika kedua inputnya "0", maka keluaran gerbang OR akan menjadi "0".

Contoh sederhana dari aplikasi gerbang logika OR adalah **Pintu Otomatis**. Bayangkan Anda memiliki pintu otomatis yang harus dibuka jika ada gerakan di depan sensor atau jika tombol pengendali ditekan. Di sini, kita dapat menggunakan gerbang OR untuk mengontrol pintu. Mari kita katakan bahwa pintu harus terbuka jika sensor mendeteksi gerakan (A) atau tombol ditekan (B).

- Input A: Jika sensor mendeteksi gerakan adalah "1" (aktif).
- Input B: Jika tombol ditekan adalah "1" (aktif).

Dalam hal ini, gerbang OR akan menghasilkan keluaran yang akan membuka pintu jika setidaknya ada salah satu dari kondisi tersebut terpenuhi. Jika keduanya adalah "0", maka pintu akan tetap tertutup.

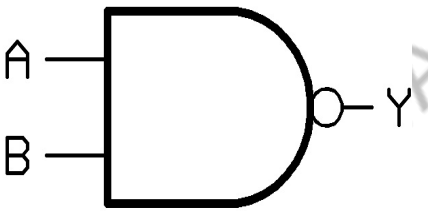
Sama seperti gerbang AND, gerbang OR juga digunakan dalam pembentukan rangkaian logika yang lebih kompleks. Gerbang OR sering digunakan dalam pembentukan pintu gerbang logika lainnya, seperti NAND, NOR, XOR, dan lainnya, untuk membangun sirkuit-sirkuit yang dapat melakukan operasi logika yang lebih kompleks.

Dalam perancangan rangkaian digital, pemahaman tentang gerbang OR dan cara mereka berinteraksi dengan gerbang-gerbang lainnya adalah kunci untuk merancang sirkuit logika yang sesuai dengan tujuan dan aplikasi tertentu.

F. Gerbang NAND

Gerbang NAND adalah gerbang logika dasar dalam elektronika digital yang menghasilkan keluaran yang berkebalikan dari hasil operasi gerbang AND. Fungsinya adalah untuk menghasilkan keluaran "0" hanya jika semua inputnya adalah "1" (tinggi). Dengan kata lain, gerbang NAND melakukan operasi "dan" dalam logika boolean, tetapi keluarannya dibalik atau di-"nand"-kan.

Simbol untuk gerbang NAND adalah sebagai berikut:



Gambar 2.4. Simbol gerbang NAND

Berikut ini adalah tabel kebenaran untuk gerbang NAND dengan 2 masukan:

Tabel 2.4. Tabel kebenaran gerbang NAND

A	B	Keluaran
0	0	1
0	1	1
1	0	1
1	1	0

Dalam tabel kebenaran di atas, keluaran gerbang NAND akan menjadi "0" hanya jika kedua input A dan B adalah "1". Jika salah

satu atau kedua input adalah "0", maka keluaran gerbang NAND akan menjadi "1".

Salah satu aspek menarik dari gerbang NAND adalah bahwa dengan menggabungkan beberapa gerbang NAND bisa mengimplementasikan gerbang logika lainnya seperti gerbang NOT, OR, dan AND. Ini dikenal sebagai "universal gate" karena kemampuannya untuk membangun semua gerbang dasar lainnya.

Sebagai contoh:

- Gerbang NOT: Anda dapat mengimplementasikan gerbang NOT dengan menghubungkan input yang sama ke kedua input gerbang NAND dan menghubungkan output mereka.
- Gerbang OR: Anda dapat mengimplementasikan gerbang OR dengan menghubungkan input ke gerbang NAND dan menggunakan input negasi (input NOT) untuk kedua masukan gerbang NAND.
- Gerbang AND: Anda dapat mengimplementasikan gerbang AND dengan menghubungkan dua input ke gerbang NAND dan menggunakan input negasi (input NOT) untuk kedua masukan tersebut.

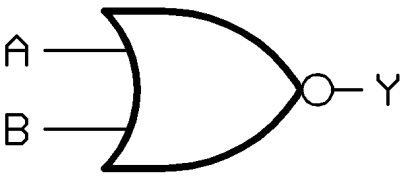
Gerbang NAND juga digunakan dalam berbagai rangkaian logika dan perangkat digital. Karena kemampuannya untuk membangun semua gerbang dasar lainnya, gerbang NAND memainkan peran penting dalam perancangan sirkuit-sirkuit logika yang lebih kompleks.

G. Gerbang NOR

Gerbang NOR adalah gerbang logika dasar dalam elektronika digital yang menghasilkan keluaran "1" hanya jika semua inputnya adalah "0" (rendah). Fungsinya adalah untuk menghasilkan keluaran yang berkebalikan dari hasil operasi gerbang OR. Gerbang NOR melakukan operasi "atau" dalam logika boolean, tetapi keluarannya

dibalik atau di-"nor"-kan.

Simbol untuk gerbang logika NOR adalah sebagai berikut:



Gambar 2.5. Simbol gerbang NOR

Berikut ini adalah tabel kebenaran dari gerbang NOR menggunakan 2 masukan:

Tabel 2.5. Tabel Kebenaran NOR

A	B	Keluaran
0	0	1
0	1	0
1	0	0
1	1	0

Dalam tabel kebenaran di atas, keluaran gerbang NOR akan menjadi "1" hanya jika kedua input A dan B adalah "0". Jika salah satu atau kedua input adalah "1", maka keluaran gerbang NOR akan menjadi "0".

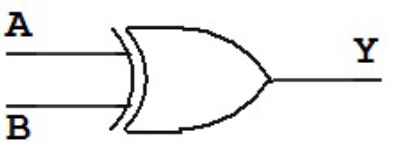
Seperti halnya gerbang NAND, gerbang NOR juga memiliki kemampuan untuk mengimplementasikan gerbang logika lainnya. Anda dapat menggunakan gerbang NOR untuk mengimplementasikan gerbang NOT, AND, dan OR.

H. Gerbang XOR dan XNOR

1. Gerbang XOR

Gerbang XOR (Exclusive OR) adalah gerbang logika dasar dalam elektronika digital yang menghasilkan keluaran "1" jika jumlah input yang berstatus "1" (tinggi) adalah ganjil. Dalam gerbang XOR, keluaran akan menjadi "1" hanya jika jumlah input "1" adalah jumlah yang ganjil, yaitu 1, 3, 5, dan seterusnya. Jika jumlah input "1" adalah genap, keluaran gerbang XOR akan menjadi "0".

Simbol gerbang logika XOR adalah sebagai berikut:



Gambar 2.6. Simbol gerbang logika XOR

Berikut ini adalah tabel kebenaran dari gerbang XOR dengan 2 masukan:

Tabel 2.6. Tabel kebenaran XOR

A	B	Keluaran
0	0	0
0	1	1
1	0	1
1	1	0

Dalam tabel kebenaran di atas, keluaran gerbang XOR akan menjadi "1" hanya jika jumlah input "1" adalah ganjil, yaitu dalam

kasus A=0, B=1 atau A=1, B=0. Jika jumlah input "1" adalah genap (A=0, B=0 atau A=1, B=1), maka keluaran gerbang XOR akan menjadi "0".

Gerbang XOR sering digunakan dalam berbagai aplikasi, termasuk dalam:

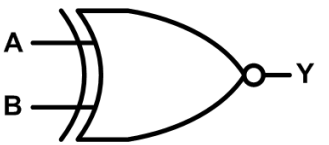
- **Deteksi Paritas:** Dalam komunikasi digital, gerbang XOR digunakan untuk mendeteksi kesalahan atau kehilangan data dalam pengiriman.
- **Pemrograman Binary:** Dalam pemrograman, gerbang XOR dapat digunakan untuk melakukan operasi pada bit-bit dalam representasi binary.
- **Kriptografi:** Dalam enkripsi dan dekripsi data, gerbang XOR digunakan dalam operasi kriptografi untuk mengamankan informasi.

Gerbang XOR juga merupakan komponen penting dalam pembentukan rangkaian logika yang lebih kompleks, seperti pembuatan gerbang XNOR (Exclusive NOR), penambahan binary, dan bahkan dalam perancangan algoritma logika digital yang lebih rumit.

2. Gerbang XNOR

Gerbang XNOR (Exclusive NOR) adalah gerbang logika dasar dalam elektronika digital yang menghasilkan keluaran "1" jika jumlah input yang berstatus "1" (tinggi) adalah genap. Gerbang XNOR melakukan operasi yang merupakan kebalikan dari gerbang XOR. Dalam gerbang XNOR, keluaran akan menjadi "1" hanya jika jumlah input "1" adalah genap, yaitu 0 atau 2.

Gerbang logika XNOR disimbolkan dalam gambar berikut:



Gambar 2.7. Simbol gerbang logika XNOR

Berikut ini adalah tabel kebenaran untuk gerbang XNOR dengan 2 masukan:

Tabel 2.7. Tabel kebenaran gerbang XNOR

A	B	Keluaran
0	0	1
0	1	0
1	0	0
1	1	1

Dalam tabel kebenaran di atas, keluaran gerbang XNOR akan menjadi "1" hanya jika jumlah input "1" adalah genap, yaitu dalam kasus A=0, B=0 atau A=1, B=1. Jika jumlah input "1" adalah ganjil (A=0, B=1 atau A=1, B=0), maka keluaran gerbang XNOR akan menjadi "0".

Gerbang XNOR memiliki berbagai aplikasi dalam dunia elektronika digital, termasuk:

- **Penilaian Kesamaan:** Dalam perbandingan data digital, gerbang XNOR dapat digunakan untuk memeriksa apakah dua data adalah sama atau berbeda.
- **Pemrograman Binary:** Dalam pemrograman dan komputasi, gerbang XNOR digunakan dalam operasi pada bit-bit dalam representasi binary.
- **Kalkulator Binary:** Gerbang XNOR juga digunakan dalam pembuatan kalkulator binary, di mana dua angka binary dijumlahkan dan hasilnya diubah menjadi bentuk biner.

I. Evaluasi / Soal Latihan

1. Jika input suatu gerbang NOT adalah 0, apa keluarannya?
2. Bagaimana keluaran gerbang NOT jika inputnya adalah 1?
3. Jika dipasang dua gerbang NOT berturut-turut, apa yang akan menjadi keluaran dari kombinasi ini?
4. Jika jumlah gerbang NOT yang dirangkai secara seri jumlahnya ganjil, bagaimana dengan keluarannya?
5. Jika kedua input suatu gerbang AND adalah 1, apa keluarannya?
6. Bagaimana keluaran gerbang AND jika salah satu inputnya adalah 0?
7. Gambarkan simbol gerbang logika AND.
8. Bagaimana keluaran gerbang OR jika kedua inputnya adalah 0?
9. Gambarkan simbol gerbang logika OR.
10. Jika kedua input suatu gerbang NAND adalah 1, apa keluarannya?
11. Bagaimana keluaran gerbang NAND jika salah satu inputnya adalah 0?
12. Jika salah satu dari dua input suatu gerbang NOR adalah 0, apa keluarannya?
13. Bagaimana keluaran gerbang NOR jika kedua inputnya adalah 1?
14. Bagaimana keluaran gerbang XOR jika kedua inputnya adalah 1?
15. Apa keluaran dari gerbang XNOR jika kedua inputnya adalah 0?
16. Bagaimana keluaran gerbang XNOR jika salah satu inputnya adalah 1?

BAB 3
RANGKAIAN GERBANG LOGIKA

A. Tujuan Pembelajaran

Dalam bab ini mahasiswa diharapkan bisa merancang rangkaian gerbang logika dari berbagai macam gerbang logika dasar. Selain itu, mahasiswa juga mampu untuk membuat penyederhanaan dari rangkaian gerbang logika yang kompleks ke dalam bentuk rangkaian sederhana.

B. Pendahuluan

Rangkaian gerbang logika merupakan kumpulan gerbang-gerbang logika dasar yang dihubungkan bersama-sama untuk melakukan operasi logika yang lebih kompleks. Gerbang logika adalah unit dasar dalam perancangan sirkuit digital dimana satu atau lebih masukan akan menghasilkan keluaran berdasarkan aturan logika tertentu. Rangkaian gerbang logika bertugas mengimplementasikan berbagai macam fungsi logika (a.l. operasi matematika, pemrosesan data, pengontrolan). Rangkaian gerbang logika membentuk dasar dari semua perangkat digital, dari komputer hingga ponsel pintar.

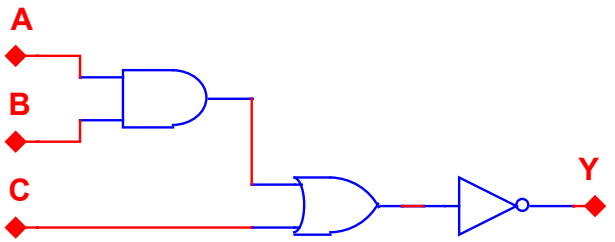
C. Rangkaian gerbang logika

Berbagai macam rangkaian gerbang logika dapat dibuat menggunakan gerbang-gerbang logika dasar. Berikut ini contoh-contoh rangkaian gerbang logika:

1. Rangkaian penentu logika (Logic Gate Circuit)

   Misalkan kita ingin membuat rangkaian yang menghasilkan keluaran "1" hanya jika kedua masukannya adalah "1". Selain itu, kita ingin menambahkan kemampuan untuk mengubah keluaran dengan tombol pengendali (C) jika ditekan.

Rangkaian:



Gambar 3.1. *Logic Gate Circuit*

Tabel 3.1. Tabel Kebenaran *Logic Gate Circuit*

A	B	C	Keluaran
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

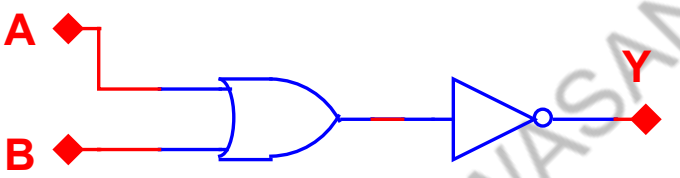
- Masukan A dan B dihubungkan ke gerbang AND, sehingga keluaran gerbang AND adalah "1" hanya ketika kedua masukannya adalah "1".
- Masukan tombol C dihubungkan ke gerbang OR bersama dengan keluaran gerbang AND. Ini berarti bahwa jika tombol C ditekan (input C=1), keluaran gerbang OR akan menjadi "1" terlepas dari keluaran gerbang AND.
- Selanjutnya keluaran gerbang OR disambungkan ke gerbang Inverter yang akan menghasilkan keluaran yang dibalik. Hal ini sama artinya jika keluaran gerbang OR adalah "1", keluaran

gerbang NOT akan menjadi "0", dan sebaliknya.

Rangkaian ini dapat berfungsi sebagai rangkaian penentu logika sederhana di mana keluaran hanya akan menjadi "1" dalam kasus tertentu sesuai dengan tabel kebenaran di atas.

2. Rangkaian pintu otomatis

Berikut ini adalah rangkaian gerbang logika untuk aplikasi pintu otomatis.



Gambar 3.2. Rangkaian pintu otomatis

Dua input: Sensor gerakan (A) dan tombol pengendali (B).

- Gerbang OR untuk menghubungkan input dari sensor gerakan (A) dan tombol pengendali (B).
- Gerbang NOT untuk menghasilkan keluaran negasi dari gerbang OR.

Tabel 3.2. Tabel Kebenaran rangkaian pintu otomatis:

A	B	Keluaran
0	0	1
0	1	1
1	0	1
1	1	0

- Jika sensor gerakan mendeteksi gerakan (A=1) atau tombol pengendali ditekan (B=1), maka keluaran dari gerbang OR akan menjadi "1".
- Namun, karena kita ingin pintu gerbang terbuka hanya ketika

keluaran gerbang OR adalah "1", kita menggunakan gerbang NOT untuk membalikkan keluaran tersebut. Sehingga, jika keluaran gerbang OR adalah "1", keluaran gerbang NOT akan menjadi "0", dan sebaliknya.

Dengan rangkaian ini, kita dapat menciptakan sistem pintu gerbang otomatis yang akan terbuka jika sensor gerakan mendeteksi gerakan atau tombol pengendali ditekan.

Seringkali rangkaian gerbang logika yang dibuat membutuhkan gerbang-gerbang logika yang banyak. Namun rangkain yang kompleks tersebut dapat dibuat penyederhanaannya sebagiamana akan dijelaskan pada sub bab berikutnya.

D. Aljabar Boolean

Ada berbagai cara untuk melakukan penyederhanaan rangkaian gerbang logika. Aljabar Boolean adalah salah satu metode untuk melakukan penyederhanaan tersebut. Metode ini merupakan suatu cabang matematika yang berkaitan dengan manipulasi simbol-simbol biner (0 dan 1) untuk menganalisis, menyederhanakan, dan merancang ekspresi dan fungsi logika. Aljabar Boolean dinamai sesuai dengan matematikawan Inggris, George Boole, yang mengembangkan konsep ini pada abad ke-19. Aljabar Boolean sangat penting dalam perancangan sirkuit logika digital, komputer, dan teknologi terkait lainnya.

Penyederhanaan gerbang logika menggunakan Aljabar Boolean melibatkan penggunaan hukum-hukum aljabar dan identitas Boolean untuk mereduksi ekspresi logika menjadi bentuk yang lebih sederhana dan efisien. Berikut langkah-langkah umum dalam penyederhanaan gerbang logika menggunakan Aljabar Boolean:

1. **Tuliskan Ekspresi Awal:** Mulailah dengan menuliskan ekspresi logika awal dalam bentuk simbol-simbol logika (AND, OR, NOT) atau dalam bentuk fungsi logika.
2. **Gunakan Hukum-Hukum Aljabar Boolean:** Gunakan hukum-

hukum aljabar Boolean seperti hukum distribusi, hukum De Morgan, hukum komutatif, hukum asosiatif, dan hukum identitas untuk menyederhanakan ekspresi. Beberapa contoh hukum yang umum digunakan.

- **Hukum Distribusi:** A . (B + C) = (A . B) + (A . C)
- **Hukum De Morgan untuk AND:** NOT (A . B) = NOT A + NOT B
- **Hukum De Morgan untuk OR:** NOT (A + B) = NOT A . NOT B

3. **Gunakan Identitas Boolean:** Identitas Boolean adalah pernyataan yang benar dalam aljabar Boolean. Beberapa identitas yang sering digunakan antara lain:
   - Identitas Zero: A . 0 = 0
   - Identitas Satu: A + 1 = 1
   - Identitas Invers: A + A' = 1 (A' adalah NOT A)
   - Identitas Pemersatu: A + A = A
   - Identitas Penyederhanaan: A × A = A
4. **Gunakan Hukum-Hukum Sirkuit Khusus:** Tergantung pada jenis sirkuit atau ekspresi yang Anda kerjakan, ada hukum-hukum khusus yang dapat membantu dalam penyederhanaan, seperti hukum NAND dan NOR.
5. **Sederhanakan Lebih Lanjut:** Lakukan penyederhanaan lebih lanjut dengan menghilangkan term-term yang redundan atau yang tidak berpengaruh pada hasil akhir.
6. **Uji Kembali dengan Tabel Kebenaran:** Setelah penyederhanaan, uji ekspresi baru dengan tabel kebenaran untuk memastikan bahwa ekspresi baru memberikan hasil yang sesuai dengan yang diharapkan.
7. **Implementasikan dalam Rangkaian:** Setelah penyederhanaan, Anda dapat mengimplementasikan ekspresi yang telah

disederhanakan ke dalam rangkaian gerbang logika dengan lebih efisien.

Penyederhanaan menggunakan Aljabar Boolean membantu mengurangi kompleksitas sirkuit dan meningkatkan efisiensi operasional. Ini juga berguna dalam mengidentifikasi dan mengatasi masalah dalam perancangan sirkuit logika digital.

Contoh penyederhanaan rangkaian gerbang logika menggunakan Aljabar Boolean:

Contoh 3.1.

Ekspresi awal: F = (A + B) . (A + C)

Langkah-langkah penyederhanaan:

- **Langkah 1:** Gunakan hukum distribusi

  F = A . (B + C)

- **Langkah 2:** Gunakan hukum De Morgan

  F = A . (B' . C')

- **Langkah 3:** Gunakan identitas Boolean (Invers)

  F = A . (NOT B) . (NOT C)

- **Langkah 4:** Gunakan hukum De Morgan lagi (AND menjadi OR)

  F = A . NOT (B + C)

- **Langkah 5:** Gunakan hukum distribusi lagi (NOT dari OR)

  F = A . (B' . C')

- **Langkah 6:** Gunakan identitas Boolean (Invers) lagi

  F = A . (NOT B) . (NOT C)

Dengan demikian, ekspresi logika awal F = (A + B) . (A + C) telah disederhanakan menjadi F = A . (NOT B) . (NOT C).

Contoh 3.2.

Ekspresi awal: F = (A + B) . (C + D) + (A + B) . (C + D')

Langkah-langkah penyederhanaan:

Langkah 1: Gunakan hukum distribusi

- F = (A + B) . (C + D + D') // Distribusi ke kedua bagian
- F = (A + B) . (C + 1) // Karena D + D' = 1
- F = (A + B) ×.1 // Karena C + 1 = 1
- F = A + B // Karena X × 1 = X

Langkah 2: Penyederhanaan lebih lanjut (Identitas Penyederhanaan)

- F = A + B // Tidak dapat lebih disederhanakan

Dengan demikian, ekspresi logika awal F = (A + B) . (C + D) + (A + B) . (C + D') telah disederhanakan menjadi F = A + B.

Dalam Aljabar Boolean dikenal juga berbagai macam bentuk fungsi yang berhubungan dengan rangkaian gerbang logika. Bentuk-bentuk fungsi tersebut antara lain:

8. Bentuk Sum of Product (SOP)

Bentuk fungsi Sum of Products (SOP) adalah bentuk ekspresi logika dalam Aljabar Boolean di mana kita menggabungkan beberapa produk (AND) yang dijumlahkan (OR) bersama untuk menggambarkan kondisi di mana output fungsi logika dinyalakan (1). SOP juga dikenal dengan istilah "Canonical Form" karena setiap minterm yang menyebabkan output dinyalakan direpresentasikan secara eksplisit.

Dalam bentuk fungsi SOP, setiap kombinasi input yang menghasilkan output "1" direpresentasikan sebagai produk dari variabel input yang mungkin ditemukan dalam kombinasi tersebut. Produk ini mencakup semua variabel input yang ada dalam kombinasi. Setelah semua produk dibentuk, produk-produk tersebut dijumlahkan (OR) bersama untuk mendapatkan ekspresi akhir yang

mewakili fungsi logika. Penjumlahan ini menggambarkan bahwa output akan menjadi "1" jika salah satu dari produk-produk tersebut aktif.

Contoh:

Misalnya, jika kita memiliki dua variabel input A dan B, dan kita ingin mewakili fungsi logika "1" saat A=0 dan B=1 atau saat A=1 dan B=0, maka ekspresi SOP-nya akan menjadi: F = (A' . B) + (A . B')

Dalam ekspresi ini, kita memiliki dua produk (A' . B) dan (A . B') yang dijumlahkan bersama.

Bentuk SOP memberikan representasi yang sangat eksplisit tentang kondisi di mana output akan menjadi "1". Dengan menganalisis tabel kebenaran dan menterjemahkannya ke dalam bentuk SOP, kita dapat memahami bagaimana sirkuit logika bekerja secara rinci. Meskipun bentuk SOP sangat berguna dalam menganalisis dan mendesain sirkuit logika, bentuk SOP bisa menjadi panjang jika jumlah variabel input atau jumlah kombinasi yang harus dipertimbangkan sangat besar.

9. Bentuk Product of Sum (POS)

Bentuk Product of Sums (POS) adalah bentuk lain dari ekspresi logika dalam Aljabar Boolean. Dalam bentuk ini, kita menggabungkan beberapa penjumlahan (OR) yang dikalikan (AND) bersama untuk menggambarkan kondisi di mana output fungsi logika mati (0). POS juga dikenal dengan istilah "Canonical Form" karena setiap maxterm yang menyebabkan output mati direpresentasikan secara eksplisit.

Dalam POS, setiap kombinasi input yang menghasilkan output "0" direpresentasikan sebagai penjumlahan (OR) dari variabel input yang tidak mungkin ditemukan dalam kombinasi tersebut. Penjumlahan ini mencakup semua variabel input yang tidak muncul dalam kombinasi. Setelah semua penjumlahan dibentuk, penjumlahan-penjumlahan tersebut dikalikan (AND) bersama untuk mendapatkan ekspresi akhir yang mewakili fungsi logika. Pengalihan ini menggambarkan bahwa output akan menjadi "0" hanya jika salah satu dari penjumlahan tersebut tidak aktif.

Contoh:

Misalkan kita memiliki tiga variabel input A, B, dan C, dan kita ingin mewakili fungsi logika "0" saat input adalah 101 (A=1, B=0, C=1) atau 110 (A=1, B=1, C=0).

Ekspresi POS-nya akan menjadi:

F = (A' + B + C) × (A + B' + C')

- Kedua penjumlahan (A' + B + C) dan (A + B' + C') mewakili kombinasi input yang menghasilkan output "0."
- Dua penjumlahan ini dikalikan (AND) bersama untuk menghasilkan ekspresi akhir yang menggambarkan kondisi di mana output fungsi logika mati (0).

10. Bentuk minterm

Bentuk minterm adalah salah satu bentuk ekspresi logika dalam Aljabar Boolean yang mewakili kombinasi input yang menghasilkan output "1". Dalam bentuk ini, setiap minterm mewakili satu kombinasi input yang tepat yang menyebabkan output fungsi logika menjadi aktif (1).

Dalam bentuk minterm, setiap kombinasi input yang menghasilkan output "1" direpresentasikan sebagai produk (AND) dari variabel input yang ada dalam kombinasi tersebut. Minterm mencakup semua variabel input yang ada dalam kombinasi yang menghasilkan output "1".

Contoh:

Misalnya, jika kita memiliki dua variabel input A dan B, dan kita ingin mewakili fungsi logika "1" saat A=1 dan B=0, maka minterm-nya akan menjadi:

m(1, 0) = A . B'

Dalam minterm ini, produk (AND) A . B' mewakili kombinasi input A=1 dan B=0 yang menghasilkan output "1".

11. Bentuk maxterm

Bentuk maxterm adalah bentuk ekspresi logika dalam Aljabar Boolean yang mewakili kombinasi input yang menghasilkan output "0". Dalam bentuk ini, setiap maxterm mewakili satu kombinasi input yang tepat yang menyebabkan output fungsi logika menjadi tidak aktif (0).

Dalam bentuk maxterm, setiap kombinasi input yang menghasilkan output "0" direpresentasikan sebagai penjumlahan (OR) dari variabel input yang ada dalam kombinasi tersebut.

Maxterm mencakup semua variabel input yang ada dalam kombinasi yang menghasilkan output "0". Bentuk maxterm juga sangat terkait dengan tabel kebenaran. Setiap baris dalam tabel kebenaran yang memiliki output "0" akan diwakili oleh satu maxterm.

Contoh:

Misalkan kita memiliki fungsi logika 3 variabel masukan (A, B, dan C) yang dinyatakan dalam bentuk maxterm sebagai berikut:

F = M(0, 1, 2, 4, 5, 6)

Dalam maxterm ini, angka-angka yang disebutkan dalam tanda kurung adalah kombinasi input yang menyebabkan output fungsi logika menjadi "0". Masing-masing angka mewakili baris dalam tabel kebenaran di mana output adalah "0".

Artinya, kita perlu mencari kombinasi input yang tepat yang menyebabkan output fungsi logika menjadi "0" sesuai dengan maxterm di atas. Kita dapat melihat tabel kebenaran berikut:

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

0 | 1 | 1 | 0

1 | 0 | 0 | 0

1 | 0 | 1 | 0

1 | 1 | 0 | 0

1 | 1 | 1 | 0

Dari tabel kebenaran di atas, kita dapat melihat bahwa kombinasi input (0, 1, 2, 4, 5, 6) menghasilkan output "0," sesuai dengan maxterm yang diberikan.

E. Peta Karnaugh

Selain menggunakan Aljabar Boolean, cara menyederhanakan rangkaian gerbang logika adalah menggunakan peta Karnaugh. Peta Karnaugh, juga dikenal sebagai "Karnaugh Map" atau "K-Map," adalah alat visual yang digunakan dalam aljabar Boolean untuk menyederhanakan dan menganalisis ekspresi logika. Peta Karnaugh membantu dalam penyederhanaan ekspresi logika dengan cara mengelompokkan kombinasi input yang memiliki nilai keluaran yang sama. Ini adalah teknik yang sangat berguna dalam perancangan sirkuit digital yang kompleks.

Berikut adalah penjelasan rinci tentang Peta Karnaugh:

1. Bentuk Matriks:

- Peta Karnaugh berbentuk matriks persegi dengan ukuran yang bervariasi tergantung pada jumlah variabel input yang terlibat dalam ekspresi logika.
- Setiap baris dan kolom mewakili satu kombinasi input, dan setiap sel mewakili nilai keluaran yang sesuai.

2. **Notasi Biner:**

   - Setiap sel dalam Peta Karnaugh diisi dengan nilai biner yang sesuai berdasarkan tabel kebenaran.

   - Misalnya, jika ada dua variabel input (A dan B), maka ada empat kombinasi input yang mungkin (00, 01, 10, 11), dan setiap kombinasi memiliki nilai keluaran yang sesuai (0 atau 1).

3. **Grupkan Minterm:**

   - Tujuan utama Peta Karnaugh adalah untuk mengelompokkan minterm yang memiliki nilai keluaran yang sama.

   - Minterm adalah representasi biner dari kombinasi input yang menghasilkan nilai keluaran "1."

4. **Pengelompokan:**

   - Dalam Peta Karnaugh, minterm yang bersebelahan atau dalam bentuk kelompok yang bisa berupa kotak dua kali dua, kotak dua kali empat, dan sebagainya, dikelompokkan bersama.

   - Kelompok ini memiliki nilai keluaran yang sama dan dapat digunakan untuk menyederhanakan ekspresi logika.

5. **Penyederhanaan:**

   - Setelah minterm dikelompokkan, ekspresi logika dapat disederhanakan dengan mengambil setiap kelompok yang berisi minterm "1" dan mengambil nilai variabel yang tetap konsisten di seluruh kelompok.

   - Setiap kelompok yang hanya berisi minterm "1" digunakan sebagai kontributor dalam ekspresi penyederhanaan.

6. **Batasan dan Ukuran Peta:**

   - Peta Karnaugh menjadi kurang praktis saat jumlah variabel input meningkat. Dalam kasus yang lebih rumit, ukuran

peta bisa sangat besar.

- Penting untuk membatasi ukuran peta sehingga masih dapat dibaca dan dianalisis dengan mudah.

7. Kelebihan dan Kekurangan:

- Kelebihan Peta Karnaugh adalah kemampuannya dalam menyederhanakan ekspresi logika dengan cara visual yang mudah dimengerti.
- Kekurangannya adalah dalam mengatasi ekspresi yang memiliki banyak variabel input atau yang sangat kompleks.

F. Metode Quine McCluskey

Metode Quine McCluskey adalah salah satu metode yang bisa digunakan untuk membuat suatu rangkaian gerbang logika yang kompleks menjadi rangkaian yang lebih sederhana. Metode Quine McCluskey adalah teknik yang digunakan untuk menyederhanakan ekspresi logika Boolean yang kompleks menjadi bentuk yang lebih sederhana dan efisien. Metode ini adalah salah satu cara paling umum dalam penyederhanaan ekspresi logika dalam bentuk sum of products (SOP) dengan menggunakan pendekatan algoritmik.

Berikut adalah langkah-langkah umum dalam Metode Quine McCluskey:

1. Membuat Tabel Minterm:

- Mulailah dengan membuat tabel yang berisi daftar minterm yang muncul dalam ekspresi logika. Setiap baris dalam tabel mewakili satu minterm dan diberi nomor urut (yang juga disebut "minterm ID") yang sesuai dengan representasi biner dari kombinasi input.

2. Kelompokkan Minterm dengan Jumlah 1 Bit yang Berbeda:

- Dalam langkah ini, Anda akan mengelompokkan minterm

dalam tabel berdasarkan jumlah bit "1" yang berbeda dalam masing-masing minterm.

- Minterm yang memiliki jumlah bit "1" yang berbeda hanya satu disebut "minterm esensial."

3. Kelompokkan Minterm dengan Jumlah 1 Bit yang Sama:

- Sekarang, Anda akan mengelompokkan minterm yang memiliki jumlah bit "1" yang sama untuk membuat kelompok-kelompok yang lebih besar.

- Anda akan membandingkan minterm dalam setiap kelompok dan mencari kombinasi input yang hanya memiliki perbedaan satu bit. Jika ada perbedaan hanya dalam satu bit, kelompok-kelompok ini dapat digabungkan.

4. Membuat Tabel Implicant Esensial:

- Setelah kelompok-kelompok dikombinasikan sebanyak mungkin, Anda membuat tabel yang berisi implicant esensial (minterm yang tidak dapat digabungkan lebih lanjut).

- Tabel ini akan membantu Anda melacak minterm yang sudah digunakan dalam proses penyederhanaan.

5. Pilih Implicant untuk Penyederhanaan:

- Anda akan memilih implicant yang akan digunakan dalam penyederhanaan ekspresi. Tujuannya adalah untuk mencakup semua minterm dalam tabel minterm sembari menggunakan jumlah implicant yang sekecil mungkin.

6. Bentuk Ekspresi Sederhana:

- Dengan implicant yang dipilih, Anda akan membentuk ekspresi logika sederhana dengan menggabungkan implicant-impliant tersebut.

7. Penyederhanaan Lanjutan Jika Perlu:

- Jika ekspresi masih bisa disederhanakan lebih lanjut,

Anda dapat menggunakan teknik seperti Peta Karnaugh atau hukum-hukum Aljabar Boolean untuk melakukan penyederhanaan tambahan.

G. Evaluasi / Soal Latihan

1. Sederhanakan ekspresi logika berikut menggunakan Peta Karnaugh:

   F = AB' + A'C + BC + AC'

2. Gunakan metode Quine-McCluskey untuk menyederhanakan ekspresi logika berikut:

   F = Σ(1, 2, 4, 5, 7)

3. Sederhanakan ekspresi logika berikut menggunakan Peta Karnaugh dan lakukan penyederhanaan lanjutan jika diperlukan:

   F = Σ(0, 1, 2, 3, 5, 7)

4. Gunakan metode Quine-McCluskey dan Peta Karnaugh untuk menyederhanakan ekspresi logika berikut:

   F = Σ(0, 1, 2, 5, 6, 7).

BAB 4
RANGKAIAN LOGIKA KOMBINASIONAL

A. Tujuan Pembelajaran

Dalam bab ini, mahasiswa diharapkan bisa menjelaskan dan merangkai berbagi macam rangkaian logika kombinasional. Selain itu, mahasiswa bisa menerangkan kegunaan rangkaian kombinasional ini dalam aplikasi bidang elektronika.

B. Pendahuluan

Rangkaian logika kombinasional adalah dasar dari dunia elektronika digital yang memungkinkan untuk merancang dan memahami cara sirkuit-sirkuit elektronik bekerja untuk melakukan operasi logika dan aritmatika. Dalam dunia yang semakin terkoneksi secara digital, pemahaman tentang rangkaian logika kombinasional menjadi semakin penting.

Dalam bab ini akan dipelajari tentang penerapan IC digital sebagai gerbang logika yang membentuk dasar operasi sirkuit digital. Selain itu, akan dijelaskan juga tentang konsep dasar rangkaian kombinasional dan bagaimana sirkuit-sirkuit tersebut melakukan pengolahan data masukan.

C. Enkoder

Enkoder atau encoder adalah perangkat atau sistem elektronik yang berfungsi untuk mengubah sinyal masukan ke dalam bentuk yang sesuai atau dapat diterima oleh perangkat lain atau sistem. Enkoder sering digunakan dalam berbagai konteks, terutama dalam elektronika digital dan komunikasi. Berikut adalah contoh-contoh aplikasi enkoder:

Konversi Sinyal: Fungsi utama enkoder adalah mengonversi sinyal masukan dari satu bentuk ke bentuk lain. Ini dapat mencakup

konversi dari sinyal analog ke digital, konversi antara format komunikasi, atau bahkan konversi antara sistem bilangan seperti biner ke heksadesimal.

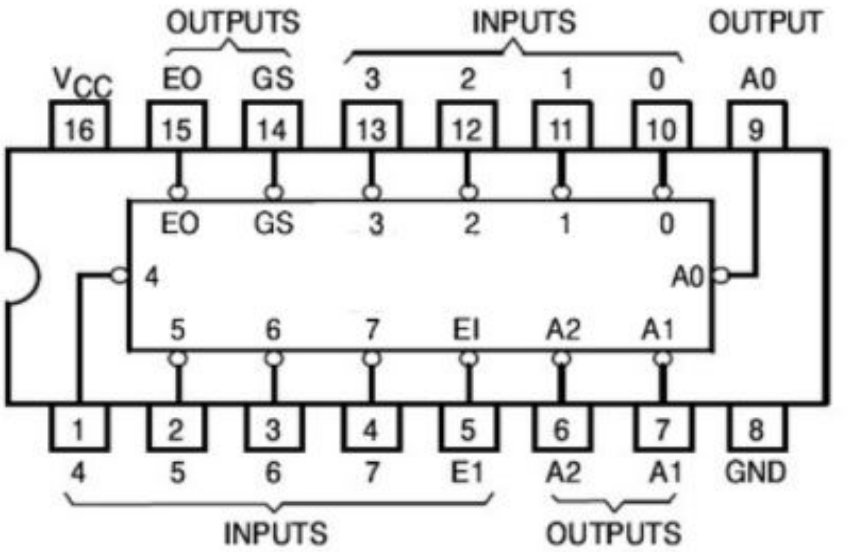
Pengukuran dan Posisi: Salah satu penggunaan umum enkoder adalah dalam pengukuran dan pemantauan posisi. Enkoder rotasi digunakan untuk mengukur perputaran atau posisi dalam mesin atau peralatan. Mereka menghasilkan pulsa elektrik yang sesuai dengan perubahan posisi.

Komunikasi: Enkoder juga digunakan dalam komunikasi data. Sebagai contoh, dalam jaringan komputer, enkoder mengubah data menjadi bentuk yang dapat ditransmisikan melalui jaringan dan kemudian diubah kembali oleh dekoder di sisi penerima.

Keamanan: Dalam beberapa aplikasi, enkoder digunakan untuk tujuan keamanan. Sebagai contoh, enkoder dapat digunakan untuk mengamankan komunikasi data dengan mengubahnya ke dalam bentuk yang tidak dapat dimengerti oleh pihak yang tidak berwenang tanpa kunci dekripsi yang sesuai.

Salahsatu komponen yang bisa digunakan sebagai encoder adalah IC 74LS148. Komponen ini bisa digunakan untuk mengubah sinyal analog menjadi sinyal digital atau Analog to Digital Conventer(ADC). Yang perlu diperhatikan dalam penggunaan IC decoder ini adalah perubahan-perubahan sinyal masukan terhadap perubahan-perubahan sinyal keluarannya.

Konfigurasi pin dan tabel kebenaran masukan dan keluaran IC 74LS148 dapat dilihat pada gambar di bawah ini.



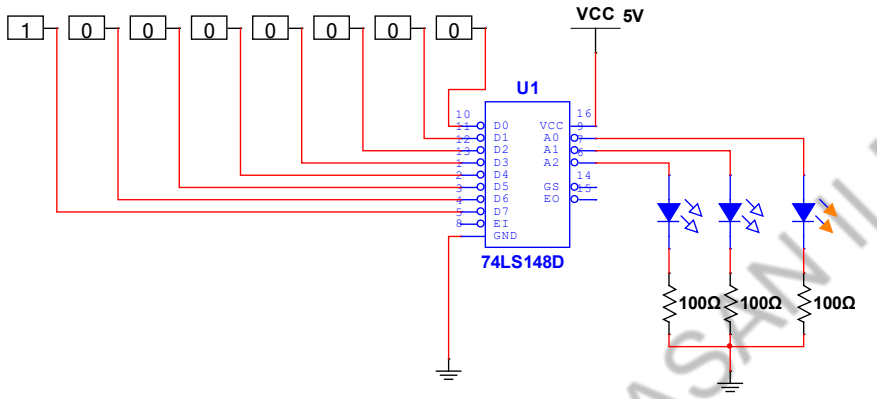
Gambar 4.1. Konfigurasi IC 74LS148

Berikut ditampilkan tabel kebenaran dari IC 74LS148

Tabel 4.1. Tabel kebenaran IC 74LS148

INPUTS									OUTPUTS				
EI	0	1	2	3	4	5	6	7	A2	A1	A0	GS	EO
H	X	X	X	X	X	X	X	X	H	H	H	H	H
L	H	H	H	H	H	H	H	H	H	H	H	H	L
L	X	X	X	X	X	X	X	L	L	L	L	L	H
L	X	X	X	X	X	X	L	H	L	L	H	L	H
L	X	X	X	X	X	L	H	H	L	H	L	L	H
L	X	X	X	X	L	H	H	H	L	H	H	L	H
L	X	X	X	L	H	H	H	H	H	L	L	L	H
L	X	X	L	H	H	H	H	H	H	L	H	L	H
L	X	L	H	H	H	H	H	H	H	H	L	L	H
L	L	H	H	H	H	H	H	H	H	H	H	L	H

Salahsatu penerapan IC74LS148 dalam rangkaian disajikan dalam gambar berikut:



Gambar 4.2. Penerapan IC 74LS148 dalam rangkaian

Pada gambar 4.2 input dibuat berurutan dari 0 s.d 7 dari kanan ke kiri, sedangkan outputnya dibuat urutan dari A2 s.d. A0 (sesuai penempatan urutan bit priority bilangan biner). Jika input bit 7 bernilai 1 artinya sinyal masukan 10000000 atau HLLLLLLL maka sinyal keluarannya 001 atau LLH. Jika input bit 6 bernilai 1 artinya sinyal masukan X1000000 atau XHLLLLLL maka sinyal keluarannya 010 atau LHL. Jika input bit 5 bernilai 1 artinya sinyal masukan XX100000 atau XXHLLLLL maka sinyal keluarannya 011 atau LH1. Begitu seterusnya menyesuaikan dengan tabel kebenaran pada tabel 4.1. Selanjutnya rangkaian ini bisa dikembangkan sebagai rangkaian ADC.

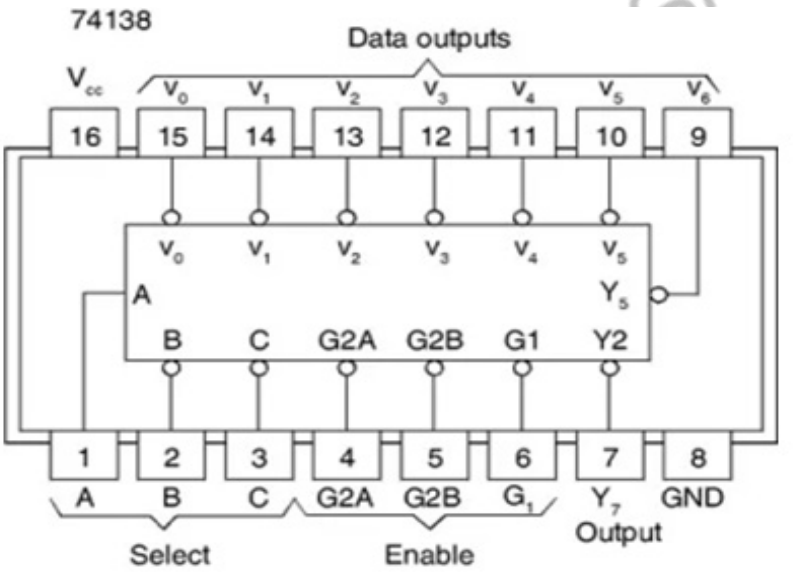
D. Dekoder

Dalam elektronika digital, sebutan "decoder" mengacu pada sebuah komponen atau perangkat logika yang digunakan untuk mengonversi kode atau sinyal masukan dalam bentuk bilangan biner menjadi satu atau lebih sinyal keluaran aktif dalam bentuk pola logika. Decoder sering digunakan dalam berbagai aplikasi,

termasuk dalam rangkaian logika, sistem kontrol, dan pemrosesan sinyal digital.

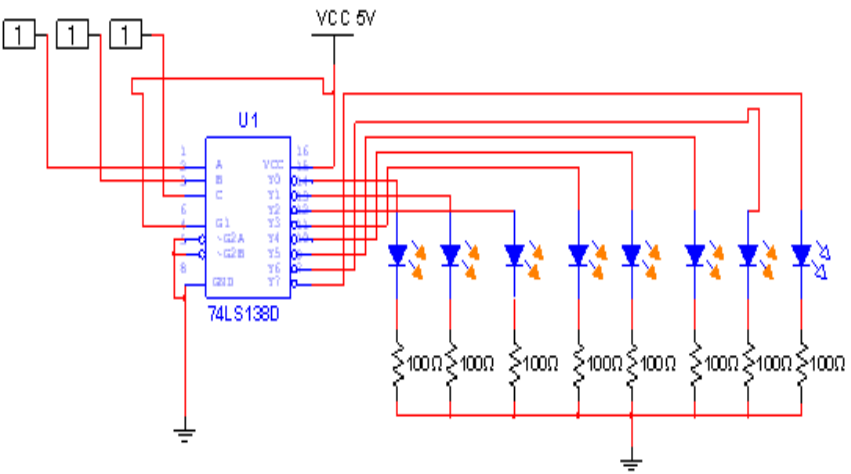
Dekoder menerima input biner dan menghasilkan output yang sesuai berdasarkan pola yang telah ditentukan. Misalnya, jika sebuah decoder memiliki tiga input (A, B, dan C), maka akan ada 2^3 atau 8 kombinasi input yang mungkin (0-7 dalam bilangan desimal). Setiap kombinasi input ini akan menghasilkan keluaran yang berbeda sesuai dengan pola logika yang telah diprogram.

Salah satu contoh IC decoder adalah IC 74LS138. Gambar 4.3 menampilkan konfigurasi dari IC 74LS138.



Gambar 4.3. Konfigurasi IC 74LS138

Contoh penerapan dekoder menggunakan IC 74LS138 dengan 3 input dan 8 output disajikan dalam Gambar 4.4.



Gambar 4.4. Rangkaian decoder menggunakan IC74LS138

Dari rangkaian di atas dapat terlihat jika $A_2A_1A_0$ = 111 maka keluarannya Y7 = 0 (rendah), dan keluaran yang lain sama dengan 1 (tinggi).

Sedangkan tabel kebenaran IC 74LS138 adalah sebagai berikut:

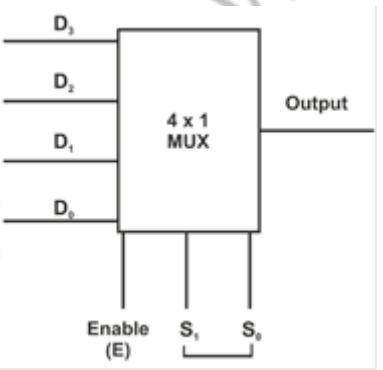
Tabel 4.2. Tabel Kebenaran IC74LS138

Input				Output							
E1	A2	A1	A0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
0	X	X	X	1	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1	1	1	1	0
1	0	0	1	1	1	1	1	1	1	0	1
1	0	1	0	1	1	1	1	1	0	1	1
1	0	1	1	1	1	1	1	0	1	1	1
1	1	0	0	1	1	1	0	1	1	1	1
1	1	0	1	1	1	0	1	1	1	1	1
1	1	1	0	1	0	1	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1	1	1

E. Multiplexer (MUX)

Multiplexer, sering disingkat sebagai "MUX," adalah perangkat elektronik dalam elektronika digital yang digunakan untuk menggabungkan beberapa sinyal masukan ke dalam satu sinyal keluaran tunggal. Ini adalah komponen penting dalam perancangan sirkuit digital dan memiliki banyak aplikasi dalam pemrosesan data, pengontrolan, dan pengalihan sinyal.

Multiplexer memiliki beberapa input (n) dan satu output (1). Itu memungkinkan pengguna untuk memilih salah satu dari input dan mengirimkannya ke output. Pemilihan input yang akan dikirimkan ke output dikendalikan oleh sinyal kontrol. Jika Anda memiliki 2^n input, Anda akan memerlukan log2(n) bit sinyal kontrol untuk memilih input yang sesuai.



Gambar 4.5. Simbol Multiplexer 4X1

Kontrol (S1 dan S0) digunakan untuk memilih salah satu dari empat data yang tersedia dan data ini akan direfleksikan pada sisi keluaran. Dengan cara ini pengguna dapat memilih sinyal yang dibutuhkan di antara banyak sinyal yang tersedia.

Sedangkan tabel kebenaran dari MUX 4x1 ditambilkan dalam Tabel 4.5.

Tabel 4.5. Tabel kebenaran MUX 4x1

INPUT				OUTPUT				No. Tabel
D0	D1	D2	D3	S0	S1	Y	Keterangan	
0	X	x	X	0	0	0	D0	1
1	X	x	x	0	0	1		2
X	0	X	X	0	1	0	D1	3
X	1	X	X	0	1	1		4
X	X	0	X	1	0	0	D2	5
X	X	1	X	1	0	1		6
X	X	X	0	1	1	0	D3	7
x	X	x	1	1	1	1		8

Multiplexer digunakan dalam berbagai aplikasi, termasuk dalam pemrosesan data dan komunikasi. Misalnya, dalam pemrosesan data, multiplexer digunakan untuk memilih input dari beberapa sensor atau sumber data dan mengirimkannya ke unit pemrosesan yang tepat. Dalam komunikasi, multiplexer digunakan untuk menggabungkan beberapa saluran suara atau data ke dalam satu saluran komunikasi.

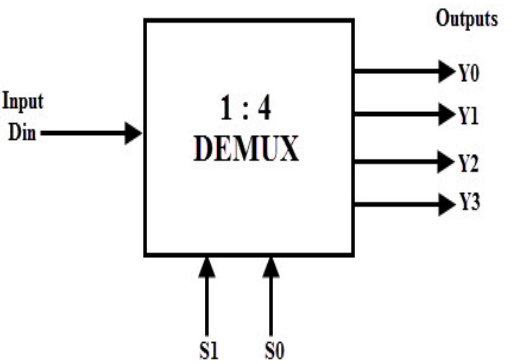
Multiplexer juga sering digunakan dalam pemilihan input dalam perangkat digital yang lebih kompleks. Misalnya, dalam memori komputer, multiplexer digunakan untuk memilih alamat memori yang tepat yang akan diakses oleh CPU. Penggunaan multiplexer mengurangi jumlah kabel dan jalur yang diperlukan dalam sirkuit digital. Ini membuat desain sirkuit lebih efisien dan menghemat sumber daya.

F. Demultiplexer (DEMUX)

Sebaliknya dari multiplexer adalah demultiplexer (sering disingkat sebagai "DEMUX"). Demultiplexer digunakan untuk membagi sinyal dari satu input menjadi beberapa output berdasarkan sinyal kontrol. Misalnya, dalam komunikasi, demultiplexer dapat

digunakan untuk membagi saluran komunikasi tunggal menjadi beberapa saluran berdasarkan alamat tujuan.

Jumlah output dalam demultiplexer biasanya bergantung pada jumlah bit sinyal kontrol yang digunakan. Jika Anda memiliki n bit sinyal kontrol, maka demultiplexer akan memiliki 2^n output, di mana setiap output mewakili kombinasi tertentu dari sinyal kontrol.



Gambar 4.6. Simbol DEMUX 1X4

Sedangkan tabel kebenaran untuk DEMUX disajikan dalam tabel 4.5.

Tabel 4.5. Tabel kebenaran DEMUX

Input			**Output**			
S0	S1	Din	Y0	Y1	Y2	Y3
0	0	0	0	X	X	X
0	0	1	1	X	X	X
0	1	0	X	0	X	X
0	1	1	X	1	X	X
1	0	0	X	X	0	X
1	0	1	X	X	1	X
1	1	0	X	X	X	0
1	1	1	X	X	X	1

Penggunaan demultiplexer dapat menghemat kabel dan jalur dalam sirkuit digital, mengurangi kebingungan dalam perutean sinyal, dan memungkinkan fleksibilitas dalam pengiriman data ke berbagai tujuan. Demultiplexer sering digunakan bersamaan dengan multiplexer. Ini memungkinkan untuk mengirimkan data dari satu sumber ke beberapa tujuan. Misalnya, data dari satu sumber dapat dimasukkan ke dalam multiplexer, dan kemudian demultiplexer digunakan di ujung penerimaan untuk membagi data tersebut menjadi beberapa output yang sesuai dengan tujuan yang berbeda.

G. Evaluasi / Soal Latihan

1. Apa fungsi utama dari sebuah enkoder dan decoder dalam konteks elektronika digital?
2. Bagaimana cara sebuah enkoder mengonversi data input menjadi data output?
3. Sebuah enkoder dengan 4 input memiliki berapa banyak output jika itu adalah enkoder biasa?
4. Bagaimana cara sebuah dekoder bekerja dalam konteks pemetaan alamat memori?
5. Jika Anda memiliki 8 input yang ingin Anda hubungkan ke 3 output, berapa banyak jalur input yang dibutuhkan dalam sebuah multiplexer?
6. Jika Anda memiliki sebuah demultiplexer dengan 4 jalur input dan 2 output, berapa banyak jalur output yang akan dipilih oleh sinyal kontrol?

.

BAB 5
FLIP-FLOP

A. Tujuan Pembelajaran

Mahasiswa mampu untuk menjelaskan tentang flip-flop dan rangkaian kombinasinya. Selain itu mahasiswa juga mampu untuk merancang rangkaian flip-flop.

B. Pendahuluan

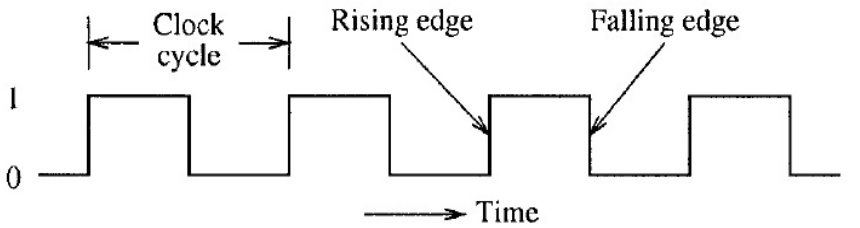
Flip-flop adalah salah satu komponen penting dalam elektronika digital. Ini adalah sirkuit digital yang digunakan untuk menyimpan dan mengubah status logika dua keadaan (biasanya 0 dan 1) secara persisten. Flip-flop digunakan dalam berbagai aplikasi, termasuk penyimpanan data, pengatur waktu, dan penggerak dalam rangkaian digital. Flip-flop memiliki dua keadaan dasar: SET (1) dan RESET (0). Ketika flip-flop diatur ke SET, itu akan menghasilkan keluaran 1, dan ketika diatur ke RESET, itu akan menghasilkan keluaran 0. Prinsip kerjanya mirip dengan sakelar elektronik yang dapat mengganti keadaan.

Ada beberapa jenis flip-flop yang berbeda, yang paling umum adalah:

- Flip-Flop SR (Set-Reset): Memiliki dua input (R dan S) dan digunakan untuk mengatur keadaan flip-flop secara eksplisit.
- Flip-Flop JK: Memiliki tiga input (J, K, dan CLK - Clock) dan dua output (Q dan ~Q). Ini adalah jenis yang paling fleksibel dan sering digunakan dalam aplikasi yang lebih kompleks.
- Flip-Flop T (Toggle): Memiliki input tunggal (T) dan dapat mengganti keadaan flip-flop saat clock berubah.
- Flip-Flop D (Data): Memiliki input tunggal (D) dan digerakkan oleh clock. Output akan menjadi sama dengan input pada saat tertentu.

C. Clock

Dalam konteks sistem digital, "clock" mengacu pada sinyal periodik yang digunakan untuk mengatur waktu dan sinkronisasi operasi komponen-komponen dalam perangkat elektronik seperti komputer, mikrokontroler, dan berbagai perangkat digital lainnya. Sinyal clock memiliki dua aspek penting: frekuensi dan duty cycle. Frekuensi clock mengukur berapa kali sinyal clock berubah dari keadaan rendah (biasanya 0) ke keadaan tinggi (biasanya 1) dalam satu detik. Frekuensi ini diukur dalam hertz (Hz). Semakin tinggi frekuensi clock, semakin banyak operasi yang dapat dilakukan dalam satu unit waktu, dan perangkat memiliki potensi untuk menjadi lebih cepat dalam eksekusi tugas-tugasnya. Sedangkan Duty cycle mengukur perbandingan antara durasi sinyal clock dalam keadaan tinggi dengan total periode sinyal clock. Duty cycle biasanya dinyatakan dalam persentase. Misalnya, duty cycle 50% berarti bahwa sinyal clock berada dalam keadaan tinggi selama setengah dari periode total sinyal clock. Duty cycle yang seimbang membantu memastikan operasi yang stabil dan sinkron antara komponen-komponen dalam sistem digital.



Gambar 5.1. Timing diagram dari clock

Sinyal clock digunakan untuk mengkoordinasikan operasi seluruh komponen dalam sistem digital. Setiap operasi dalam sistem biasanya diawali atau diakhiri pada tepat satu titik dalam periode sinyal clock. Ini memastikan bahwa semua komponen dalam sistem bergerak dalam langkah yang terkoordinasi dan terhindar dari konflik yang dapat terjadi jika operasi-operasi ini tidak diatur dengan baik.

D. Latch NAND dan NOR

1. Latch NAND

Latch NAND adalah sirkuit digital dasar yang digunakan untuk menyimpan informasi biner dengan menggunakan gerbang NAND sebagai komponen utamanya. Latch NAND umumnya dikenal dengan nama lain, yaitu "SR latch" (Set-Reset latch), karena prinsip kerjanya melibatkan input "Set" (S) dan "Reset" (R). Latch NAND digunakan untuk memahami dasar kerja dari elemen penyimpanan dalam sirkuit digital sebelum kemudian diperkenalkan jenis-jenis flip-flop yang lebih kompleks.

Prinsip Kerja latch NAND adalah sebagai berikut:

a Latch NAND memiliki dua input, yaitu input "Set" (S) dan input "Reset" (R).

b. Ketika sinyal "Set" diaktifkan (S=1), keluaran latch menjadi 1 (hidup).

c. Saat sinyal "Reset" diaktifkan (R=1), keluaran latch menjadi 0 (mati).

d. Jika keduanya dimatikan (S=0 dan R=0), maka keadaan latch tetap dalam kondisi sebelumnya.

Tabel 5.1. Tabel kebenaran untuk SR latch menggunakan gerbang NAND

S	R	Q(t)	Q(t+1)
0	0	0	0
0	1	0	0
1	0	0	1
1	1	0	-

Latch NAND digunakan sebagai dasar untuk memahami konsep penyimpanan dalam sirkuit digital. Namun, sifat khususnya yang bisa menyebabkan keadaan tak terdefinisi (undefined state) pada saat kedua input S dan R aktif, membuatnya kurang praktis dibandingkan dengan jenis flip-flop lain yang lebih canggih dan stabil.

2. Latch NOR

Latch NOR sebagaimana latch NAND adalah sirkuit digital dasar yang digunakan untuk menyimpan informasi biner dengan menggunakan gerbang NOR sebagai komponen utamanya.

Prinsip kerja latch NOR adalah sebagai berikut:

a. Latch NOR memiliki dua input, yaitu input "Set" (S) dan input "Reset" (R).

b. Ketika sinyal "Set" diaktifkan (S=0), keluaran latch menjadi 0 (mati). S

c. aat sinyal "Reset" diaktifkan (R=0), keluaran latch menjadi 1 (hidup).

d. Jika keduanya dimatikan (S=1 dan R=1), maka keadaan latch tetap dalam kondisi sebelumnya.

Tabel 5.2. Tabel kebenaran untuk SR latch menggunakan gerbang NOR

S	R	Q(t)	Q(t+1)
1	1	0	-
1	0	1	0
0	1	0	1
0	0	1	1

Latch NOR digunakan sebagai dasar untuk memahami konsep penyimpanan dalam sirkuit digital. Namun, seperti latch

SR yang menggunakan gerbang NAND, sifat khususnya yang bisa menyebabkan keadaan tak terdefinisi (undefined state) saat kedua input S dan R aktif, membuatnya kurang praktis dibandingkan dengan jenis flip-flop yang lebih canggih dan stabil.

E. Flip-Flop SR

Flip-Flop SR (Set-Reset) adalah jenis dasar dari sirkuit flip-flop yang memiliki dua input: S (set) dan R (reset). Flip-flop ini memungkinkan kita untuk mengatur (set) atau mereset (reset) nilainya berdasarkan input yang diberikan.

Prinsip Kerja Flip-flop SR adalah sebagai berikut:

1. Flip-Flop SR memiliki dua keadaan: diatur (set) dan direset (reset), yang tercermin dalam dua keluaran yang disebut Q dan Q' (inverse dari Q).
2. Saat input S=1 dan R=0, flip-flop akan berada dalam keadaan diatur (nilai Q=1).
3. Saat input S=0 dan R=1, flip-flop akan berada dalam keadaan direset (nilai Q=0).
4. Saat input S=0 dan R=0, flip-flop akan mempertahankan keadaan sebelumnya.

Tabel 5.3. Tabel kebenaran untuk flip flop SR:

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

1	1	0	-
1	1	1	-

Flip-Flop SR sering digunakan dalam kasus di mana kita ingin secara manual mengatur atau mereset suatu keadaan. Namun, kelemahannya adalah kemungkinan adanya situasi tak terdefinisi saat kedua input S dan R berada di level tinggi (1), yang dapat mengakibatkan keadaan tak terduga. Untuk mengatasi situasi tak terdefinisi, variasi dari flip flop SR seperti Flip-Flop D dengan fitur "Enable" (D dengan Enable) digunakan lebih sering karena lebih stabil. Meskipun flip flop SR cukup sederhana, dalam prakteknya lebih umum menggunakan jenis flip-flop yang lebih kompleks seperti Flip-Flop D, JK, atau T, karena mereka memberikan lebih banyak kontrol dan stabilitas terhadap perubahan input.

F. Flip-Flop JK

Flip-Flop JK adalah jenis flip-flop yang lebih canggih dan sering digunakan dalam sirkuit digital untuk penyimpanan dan sinkronisasi data. Nama "JK" berasal dari "Jack Kilby," seorang insinyur yang berkontribusi besar dalam pengembangan sirkuit terpadu. Flip-Flop JK memiliki tiga input: J (set), K (reset), dan clock (C).

Prinsip Kerja:

1. Flip-Flop JK memiliki dua keadaan: 0 (mati) dan 1 (hidup), yang tercermin dalam dua keluaran yang disebut Q dan Q' (inverse dari Q).
2. Input J dan K memungkinkan kita mengontrol perubahan keadaan flip-flop.
3. Saat clock aktif (berpindah dari 0 ke 1 atau sebaliknya), input J dan K diambil sebagai acuan untuk mengubah keadaan flip-flop.

4. Saat J=1 dan K=0, flip-flop akan diatur (nilai Q=1).
5. Saat J=0 dan K=1, flip-flop akan direset (nilai Q=0).
6. Saat J=1 dan K=1, flip-flop akan beralih (toggle) ke keadaan sebaliknya dari sebelumnya.

Tabel 5.4. Tabel kebenaran untuk flip flop JK:

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

Flip-Flop JK adalah jenis flip-flop yang sangat fleksibel dan dapat digunakan untuk berbagai tujuan, termasuk pembangunan register dan penghitungKemampuan toggle dari flip-flop JK memungkinkannya digunakan dalam aplikasi seperti pembangkit gelombang osilator dan penghitung pembalik (ripple counter). Flip-Flop JK memiliki variasi lain seperti Flip-Flop JK Master-Slave, di mana dua flip-flop JK terhubung untuk menghasilkan output yang lebih stabil.

G. Flip-Flop T

Flip-Flop T (Toggle) adalah jenis flip-flop yang memiliki fungsi khusus untuk mengalihkan (toggle) nilai keluaran setiap kali clock dinyalakan. Flip-Flop T memiliki dua input: T (toggle) dan clock (C).

Prinsip Kerja:

1. Flip-Flop T memiliki dua keadaan: 0 (mati) dan 1 (hidup), yang tercermin dalam dua keluaran yang disebut Q dan Q' (inverse dari Q).
2. Saat clock aktif (berpindah dari 0 ke 1 atau sebaliknya), input T diambil sebagai acuan untuk mengubah keadaan flip-flop.
3. Jika T=1, maka nilai flip-flop akan dibalik (toggle) saat clock aktif.
4. Jika T=0, maka nilai flip-flop akan tetap sama saat clock aktif.

Tabel 5.5. Tabel kebenaran untuk flip flop T

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

Flip-Flop T umumnya digunakan dalam aplikasi di mana perlu menghasilkan osilasi (getaran) atau gelombang persegi yang dibalik secara teratur. Dapat digunakan dalam pembalik gelombang osilator atau pembangkit gelombang.

Flip-Flop T dapat diimplementasikan dalam berbagai cara, termasuk dengan menggunakan JK flip-flop dengan input J dan K yang dihubungkan ke T. Meskipun Flip-Flop T memiliki kegunaan spesifik dalam pembangkit gelombang osilator dan aplikasi serupa, penggunaannya lebih terbatas dibandingkan dengan jenis flip-flop lain seperti Flip-Flop D atau Flip-Flop JK.

H. Flip-Flop D

Flip-Flop D (Data) adalah jenis flip-flop yang digunakan untuk menyimpan data dan memungkinkan input data baru untuk diambil saat clock aktif. Flip-Flop D memiliki satu input: D (data) dan satu input clock (C).

Prinsip Kerja:

1. Flip-Flop D memiliki dua keadaan: 0 (mati) dan 1 (hidup), yang tercermin dalam dua keluaran yang disebut Q dan Q' (inverse dari Q).
2. Saat clock aktif (berpindah dari 0 ke 1 atau sebaliknya), nilai dari input D diambil dan disimpan dalam flip-flop.
3. Keluaran Q akan mengikuti nilai input D saat clock aktif.
4. Keluaran Q akan tetap tidak berubah saat clock tidak aktif.

Tabel 5.6. Tabel kebenaran untuk flip flop D:

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

Flip-Flop D adalah jenis flip-flop yang paling umum digunakan dan memiliki aplikasi yang luas dalam perancangan sirkuit digital diantaranya digunakan dalam memori register, pembangkit gelombang, dan berbagai aplikasi penyimpanan dan sinkronisasi data. Flip-Flop D dapat memiliki variasi dengan fitur tambahan, seperti Flip-Flop D dengan Enable (D dengan Enable) yang memungkinkan pengendalian tambahan terhadap penyimpanan data.

I. Evaluasi / Soal Latihan

1. Bagaimana prinsip kerja flip-flop D saat clock aktif?
2. Apa perbedaan antara flip-flop JK dan flip-flop T dalam hal prinsip kerja dan fungsinya?
3. Sebutkan salah satu aplikasi praktis untuk flip-flop T dalam sirkuit digital.
4. Apa arti dari "flip-flop tersinkronisasi" dan mengapa sinkronisasi penting dalam flip-flop?
5. Jelaskan mengapa flip-flop RS dapat mengalami keadaan tak terdefinisi saat input S dan R berada pada level tinggi (1).
6. Apa fungsi dari input "Enable" pada flip-flop D dengan Enable?
7. Apa tujuan utama dari menggunakan flip-flop master-slave, dan mengapa hal ini dapat mengatasi masalah race condition?.

BAB 6
REGISTER DAN COUNTER

A. Tujuan Pembelajaran

Dengan memahami konsep register dan counter, mahasiswa dapat merancang dan mengimplementasikan sirkuit digital yang lebih kompleks dan efisien. Hal ini juga sangat penting dalam pengembangan perangkat digital seperti mikrokontroler, FPGA, dan sistem digital yang lebih besar.

B. Pendahuluan

Dalam era teknologi yang semakin maju, perangkat elektronik telah menjadi bagian tak terpisahkan dalam kehidupan kita. Dari perangkat komputasi hingga kendaraan otomotif, hampir semua aspek kehidupan kita melibatkan penggunaan sirkuit digital yang rumit. Dua komponen penting yang membantu mengatur dan mengolah informasi biner dalam sirkuit digital adalah "register" dan "counter."

Register adalah elemen penyimpanan data yang mampu mengadopsi beberapa bit informasi pada suatu waktu. Dalam dunia sirkuit digital, register berperan seperti "kotak penyimpanan" yang dapat memegang data, angka, atau nilai-nilai penting lainnya. Kemampuan register dalam menyimpan dan mengubah data dengan kecepatan tinggi memungkinkan operasi paralel yang efisien, memainkan peran krusial dalam banyak aplikasi, termasuk dalam pemrosesan data, pengaturan waktu, dan pemindahan informasi antara komponen-komponen sistem.

Counter, di sisi lain, adalah komponen yang memiliki kemampuan menghitung suatu peristiwa atau kejadian tertentu. Dengan kemampuan ini, counter memungkinkan pengukuran waktu, frekuensi, dan kuantitas kejadian dengan akurat. Counters dapat diatur untuk memulai dari nol dan menghitung hingga batas tertentu, serta kembali ke nilai awal. Fungsinya bukan hanya terbatas pada penghitungan sederhana, tetapi juga berperan dalam pembangkit gelombang, pembagi frekuensi, dan pembangunan

rangkaian kontrol yang mengandalkan sekuensi peristiwa.

Dalam pandangan yang lebih luas, pemahaman tentang register dan counter merupakan fondasi yang krusial bagi pengembangan sirkuit digital yang lebih kompleks. Kemampuan merancang, memahami, dan menerapkan konsep ini akan memungkinkan para ilmuwan dan perancang sistem untuk menciptakan solusi yang efisien dan inovatif dalam berbagai bidang, termasuk teknologi informasi, komunikasi, otomatisasi, dan banyak lagi.

Dalam bab ini akan dijelaskan tentang konsep dasar, prinsip kerja, jenis-jenis, serta aplikasi praktis dari register dan counter dalam dunia sirkuit digital. Semakin mendalam pemahaman kita terhadap elemen-elemen ini, semakin siap kita menghadapi tantangan dan peluang di era digital yang terus berkembang..

C. Register

Register adalah tempat penyimpanan data berbasis bit yang mampu menampung beberapa nilai data secara paralel. Setiap bit dalam register dapat merepresentasikan nilai biner 0 atau 1. Register digunakan untuk menyimpan informasi sementara, melakukan operasi data paralel, serta mentransfer data antara komponen-komponen sirkuit.

Register menerima input data dan clock. Saat clock aktif, data input ditransfer ke dalam register dan tersimpan. Register memiliki kemampuan membaca dan mengubah data secara cepat, memungkinkan operasi paralel yang efisien.

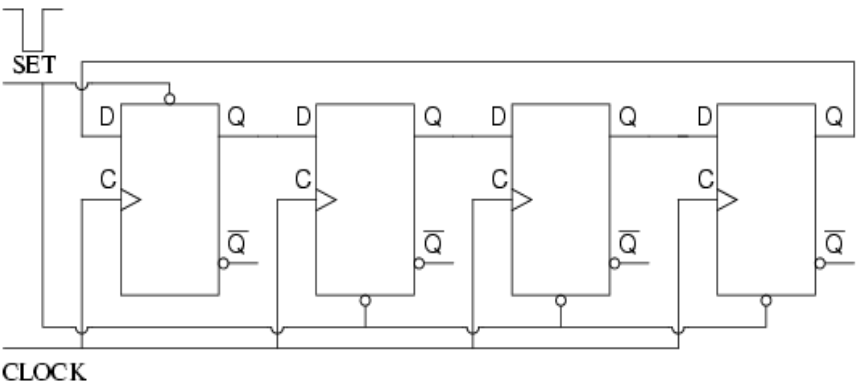
Jenis-Jenis Register:

1. **Register Paralel:** Data diinput dan dioutput secara paralel.
2. **Register Geser (Shift Register):** Data dapat digeser ke kiri atau kanan dalam register.
3. **Register Serial:** Data diinput dan dioutput secara serial, satu bit pada satu waktu.

Aplikasi Praktis Register:

1. **Memori:** Register digunakan dalam memori komputer untuk penyimpanan data.
2. **Pemrosesan Data:** Dalam operasi aritmatika dan logika.
3. **Komunikasi Serial:** Pada perangkat seperti komunikasi serial atau transmisi data.
4. **Pemindahan Data:** Dalam operasi load/store dalam mikroprosesor.

Berikut adalah contoh rangkaian register geser ring counter:



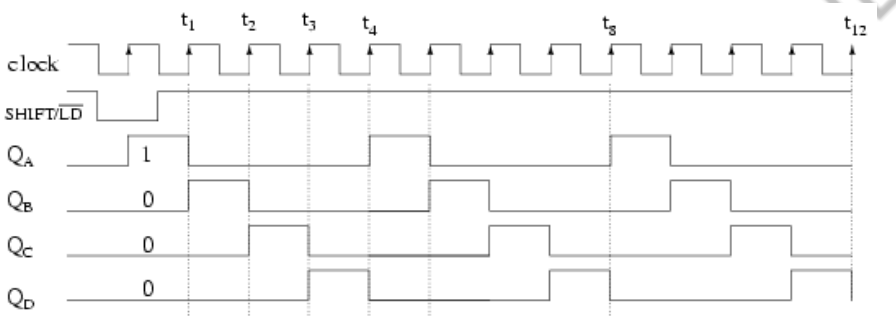
Gambar 6.1. Rangkaian register geser ring counter

Sebelum kita membahas ring counter, mari kita pahami terlebih dahulu apa itu register geser. Register geser adalah rangkaian digital yang digunakan untuk menggeser data secara sekuensial dari satu elemen penyimpanan ke elemen penyimpanan berikutnya. Ini sering digunakan untuk menggeser data melalui serial input dan output.

Ring counter adalah jenis khusus dari register geser. Perbedaannya terletak pada cara data digeser. Pada ring counter, data digeser dalam bentuk lingkaran, yang berarti output dari elemen register terakhir terhubung kembali ke input elemen register pertama, membentuk loop tertutup. Ini menciptakan urutan

tertutup dari nilai-nilai biner yang digunakan untuk menghitung.

Ring counter bekerja dengan menggeser satu bit data pada setiap langkah clock. Saat clock sinyal aktif, bit pertama akan berpindah ke bit kedua, bit kedua ke bit ketiga, dan seterusnya. Bit terakhir akan kembali ke bit pertama, sehingga menciptakan efek loop.



Gambar 6.2. Timing diagram rangkaian register geser ring counter

D. Counter

Counter adalah komponen yang menghitung perubahan kejadian atau waktu dalam bentuk digit biner. Counter dapat diatur untuk menghitung mundur, menghitung naik, atau menghitung keduanya.

Counter menerima sinyal clock yang menggerakkan penghitungan. Setiap kali clock aktif, nilai counter dapat bertambah atau berkurang tergantung pada pengaturannya.

Jenis-Jenis Counter:

1. **Counter Binari:** Menghitung dalam basis biner.
2. **Counter BCD (Binary Coded Decimal):** Menghitung dalam format desimal berkode biner.

3. **Counter Up/Down:** Dapat menghitung naik, turun, atau keduanya.

4. **Counter Ripple:** Bit yang lebih rendah mempengaruhi bit yang lebih tinggi.

Aplikasi Praktis Counter:

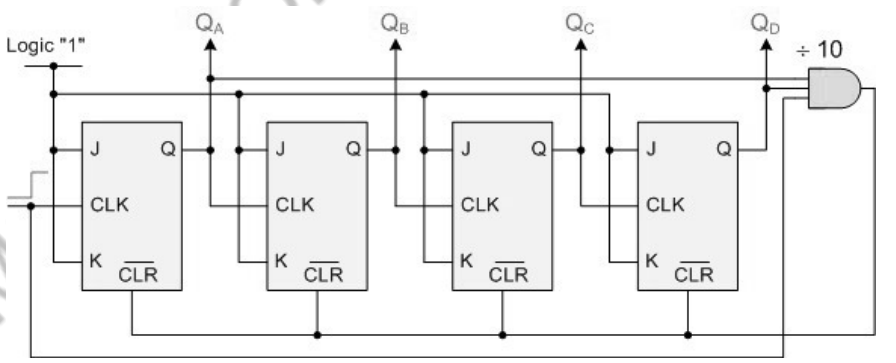
1. **Penghitungan Waktu:** Dalam jam digital, timer, serta perangkat pengukuran waktu lainnya.

2. **Pembangkit Gelombang Osilator:** Dalam pembangkit gelombang persegi dan osilator.

3. **Penghitungan Kejadian:** Dalam perangkat hitung jumlah barang, pergerakan, atau kejadian.

4. **Pemrograman Sekuensial:** Dalam otomatisasi dan kontrol industri.

Counter dapat menghitung bilangan biner dari urutan bawah ke atas yang disebut dengan up-counter. Jika kita menggunakan 4 buah flip-flop, maka hitungan tertinggi yang diperoleh adalah bilangan biner 1111_2. Counter yang dapat menghitung sampai 1111 disebut 4 bit binary counter. Berikut adalah gambar rangkaian up-counter:



Gambar 6.3. Rangkaian up-counter

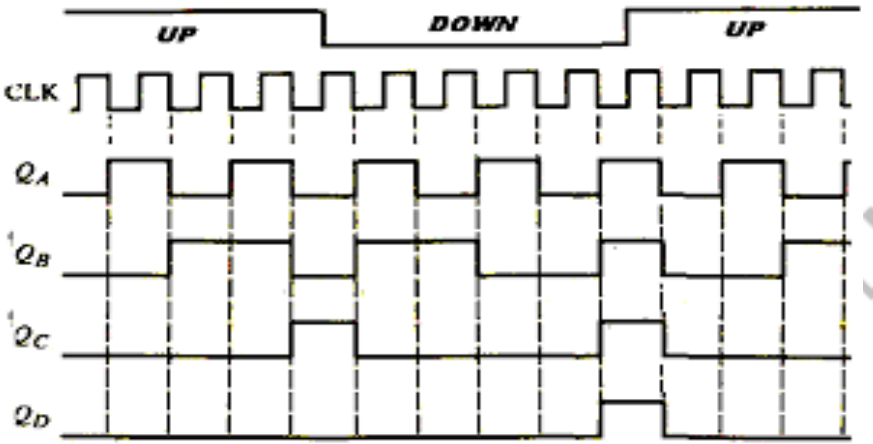
Setiap flip-flop JK digunakan untuk merepresentasikan satu bit dalam hitungan biner. Oleh karena itu, dalam counter 4-bit, kita

memiliki empat flip-flop JK yang diberi label sebagai Q0, Q1, Q2, dan Q3, masing-masing mewakili bit ke-0, bit ke-1, bit ke-2, dan bit ke-3 dalam hitungan biner. Setiap flip-flop JK memiliki dua input, yaitu J dan K, serta satu input Clock (CLK) yang digunakan untuk menggerakkan perubahan nilai. Awalnya, semua flip-flop diatur pada keadaan awal, yaitu 0000 dalam notasi biner, yang sesuai dengan 0 dalam notasi desimal. Setiap kali sinyal clock berdenyut atau berubah dari 0 ke 1, semua flip-flop berpotensi mengalami perubahan nilai sesuai dengan prinsip kerja flip-flop JK.

Prinsip kerja flip-flop JK adalah jika J=1 dan K=0, maka flip-flop akan mengubah keadaan menjadi 1, jika J=0 dan K=1, maka flip-flop akan mengubah keadaan menjadi 0, dan jika J=K=1, maka flip-flop akan beraksi sebagai toggle, yang berarti nilai akan diubah dari 0 ke 1 atau dari 1 ke 0 tergantung pada keadaan sebelumnya.

Ketika counter mulai, flip-flop paling rendah (Q0) akan menghitung dari 0 hingga 1 pada setiap naiknya sinyal clock. Ketika Q0 mencapai nilai 1 dan kemudian naik lagi, ia akan kembali ke 0 dan pada saat itu akan memberi "tumpukan" atau "carry" ke flip-flop berikutnya (Q1). Ini berlanjut dengan "penumpukan" bit ke bit berikutnya. Misalnya, ketika Q1 mencapai 1, ia akan mengaktifkan penumpukan pada Q2, dan seterusnya. Ketika seluruh counter mencapai nilai 1111 (15 dalam notasi desimal), maka setiap flip-flop akan berubah menjadi 1 saat naiknya sinyal clock berikutnya.

Sedangkan time diagram untuk rangkaian up-counter adalah sebagai berikut:



Gambar 6.4. Timing diagram rangkaian up-counter

E. Evaluasi / Soal Latihan

1. Apa yang dimaksud dengan register paralel dalam konteks sirkuit digital? Berikan contoh penggunaan register paralel.
2. Jelaskan konsep dari register geser (shift register) dan sebutkan salah satu aplikasi praktisnya.
3. Bagaimana prinsip kerja register serial? Berikan contoh situasi di mana penggunaan register serial sangat bermanfaat.
4. Sebuah counter binari mulai dari 0 dan menghitung hingga 15. Berikan representasi biner dari nilai ketika counter mencapai 9.
5. Apa yang dimaksud dengan counter BCD (Binary Coded Decimal)? Bagaimana representasi BCD dari angka 7?
6. Sebutkan perbedaan antara counter up dan counter down dalam sirkuit digital. Berikan contoh penggunaan masing-masing jenis counter.

7. Jelaskan konsep counter ripple. Mengapa counter ripple dapat lebih rentan terhadap delay dibandingkan dengan counter sinkron?

8. Bagaimana counter digunakan dalam penghitungan waktu pada jam digital? Jelaskan prinsip kerjanya.

9. Mengapa counter digunakan dalam pembangkit gelombang osilator? Berikan contoh jenis osilator yang menggunakan counter.

10. Dalam sebuah pabrik, sebuah counter digunakan untuk menghitung berapa banyak produk yang diproduksi setiap hari. Pada hari pertama, counter diatur menjadi 0. Jika pada hari kedua terdapat 350 produk yang diproduksi, berapakah nilai counter pada akhir hari kedua?.

DAFTAR PUSTAKA

Thomas L. Floyd, "Digital Fundamentals"

Donald P. Leach, Albert Paul Malvino, Goutam Saha, "Digital Principles and Applications"

Michael D. Ciletti, "Digital Design: Principles and Practices"

M. Morris Mano, Michael D. Ciletti, "Digital Design"

John F. Wakerly, "Digital Design: Principles and Practices"

Charles H. Roth Jr., "Fundamentals of Logic Design"

Ronald J. Tocci, Neal S. Widmer, Greg Moss, "Digital Systems: Principles and Applications"

John M. Yarbrough, "Digital Logic: Applications and Design"

GLOSARIUM

1. **Aljabar Boolean**: Cabang matematika yang menggunakan operasi logika untuk memodelkan operasi pada data biner.
2. **Bit**: Singkatan dari "binary digit." Merupakan unit dasar dalam sistem bilangan yang mempunyai nilai ‘0’ dan ‘1’.
3. **Clock**: Sinyal osilator periodik yang digunakan untuk mengkoordinasikan operasi dalam rangkaian digital.
4. **Counter**: Rangkaian sekuen yang menghitung atau menghasilkan urutan nilai biner berurutan.
5. **Decoder**: Sirkuit yang digunakan untuk mengonversi kode biner menjadi bentuk lain, seperti angka desimal atau tampilan 7-segment.
6. **Demultiplexer (DEMUX)**: Sirkuit yang digunakan untuk membagi satu input menjadi beberapa output berdasarkan sinyal kontrol.
7. **Elektronika Digital**: Cabang ilmu elektronika yang berfokus pada pengolahan dan manipulasi sinyal diskrit dalam bentuk kode biner (0 dan 1). Ini melibatkan desain, analisis, dan implementasi sirkuit digital yang mendasarkan operasinya pada prinsip-prinsip aljabar boolean.
8. **Encoder:** Sirkuit yang mengubah satu set input biner menjadi kode yang lebih padat atau representasi yang lebih sederhana, sering digunakan dalam komunikasi dan pemrosesan data.
9. **Flip-Flop**: Sirkuit digital yang digunakan untuk menyimpan dan mengganti status logika dua keadaan, biasanya 0 dan 1.
10. **Gerbang Logika**: Sirkuit elektronika yang melakukan operasi logika pada sinyal-sinyal biner.

11. **Kode Biner**: Representasi angka dalam sistem bilangan biner menggunakan kombinasi bit, di mana setiap bit mewakili nilai 0 atau 1.

12. **Latch D:** Jenis latch yang mengubah output sesuai dengan nilai inputnya pada sinyal clock yang aktif.

13. **Memori**: Komponen elektronik yang digunakan untuk menyimpan dan mengambil data dalam sistem komputer atau perangkat digital.

14. **Multiplexer (MUX)**: Sirkuit untuk menggabungkan beberapa masukan menjadi satu keluaran berdasarkan sinyal kontrol.

15. **Peta Karnaugh**: Metode visual untuk menyederhanakan fungsi boolean dengan mengelompokkan kombinasi masukannya.

16. **Register**: Sekumpulan flip-flop yang digunakan untuk menyimpan data sementara dalam operasi pengolahan data.

17. **Tabel Kebenaran**: Tabel yang menunjukkan semua kemungkinan masukan dan keluaran untuk suatu gerbang atau rangkaian logika.

INDEKS

BIOGRAFI PENULIS

Bagus Haryadi, seorang dosen dan peneliti di Program studi Fisika, Universitas Ahmad Dahlan. Dia menyelesaikan studi S1 di tempat yang sama dengan dia bekerja sekarang. Kelulusan Magisternya diperoleh di Institut Teknologi Bandung, sedangkan Gelar PhD nya didapatkan dari National Dong Hwa University, Taiwan. Penulis memiliki keilmuan di bidang Fisika khususnya Fisika Elektronika dan Instrumentasi. Selain itu, dia juga aktif membuat video pembelajaran di channel youtube @bagusharyadijogja.