MODUL PRAKTIKUM

TEKNIK DIGITAL

Oleh
Team Laboratorium

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS DR.SOETOMO SURABAYA
2005

Tata Tertib Praktikum
Laboratorium Teknik Informatika
Fakultas Teknik Universitas Dr. Soetomo Surabaya

PEMAKAIAN LABORATORIUM

1. Yang diperbolehkan menggunakan fasilitas Laboratorium adalah Mahasiswa Teknik Informatika Fakultas Teknik Universitas Dr. Soetomo Surabaya yang terdaftarpada BAAK dan aktif paling tidak pada tahun akademik yang sedang berlangsung.
2. Laboratorium Teknik Informatika hanya dapat digunakan pada jam kerja yang telah ditentukan . Di luar jam tersebut, pemkaian Laboratorium harus seijin pejabat yang berwenang.
3. Peralatan dan semua fasilitas Laboratorium tidak diperbolehkan dipindah tempatkan, kecuali atas seijin pejabat yang berwenang.
4. Mahasiswa yang sedang menggunakan laboratorium dilarang membawa keluar fasilitas tanpa ijin pejabat yang berwenang
5. Mahasiswa dilarang membawa teman/Mahasiswa luar kedalam Laboratorium tanpa ijin pimpinan/penanggung jawab Laboratorium.

KEWAJIBAN PRAKTIKAN

1. Hadir selambat-lambatnya 10 menit sebelum praktikum dimulai.
2. Hadir untuk seluruh acara praktikum agar dapat memenuhi keutuhan absensi sebagai persyaratan kelulusan.
3. Mengikuti test/ujian/quiz praktikum yang dilakukan oleh asisten yang bersangkutan.
4. Mengumpulkan laporan dan tugas lain yang diberikan oleh asisten/dosen.
5. Kerusakan alat akibat keteledoran dan kesengajaan pemakain menjadi tanggung jawab pemakai.
6. Menjaga keserasian, ketenangan dan kebersihan ruangan Laboratorium.
7. Merapikan kembali semua peralatan setelah selesai memakainya.
8. Minimal mengikuti 80% (delapan puluh persen) dari seluruh kegiatan praktikum dalam 1 semester.

LARANGAN PRAKTIKAN BAGI PRAKTIKAN SELAMA PRAKTIKUM

1. Dilarang makan, minum dan merokok didalam Laboratorium.
2. Membuat gaduh selama jalannya praktikum, sehingga mengganggu konsntrasi praktikan lainnya.
3. Keluar masuk laborattorium tanpa seijin asisten.

SANKSI

1. Terhadap pelanggaran TATA TERTIB diatas, asisten berhak menjatuhkan sanksi dengan peraturan berlaku :
2. Pelanggaran Point II.4. tidak diperbolehkan mengikuti praktikum
3. Pelanggaran Point II.8. Tidak diperkenankan mengikuti ujian.
4. Pelanggaran Point lainnya dikenakan sanksi teguran sampai sanksi akademik.

KATA PENGANTAR

Puji syukur kehadirat Tuhan YME atas terselesaikannya buku petunjuk praktikum Teknik Digital . Sesuai dengan motto kami yaitu “Tiada Hari Tanpa Peningkatan Mutu”, maka buku ini adalah suatu realisasi untuk terus berusaha meningkatkan mutu praktikum.

Struktur dari buku ini diharapkan memenuhi standar dari sebuah buku petunjuk praktikum dengan bagian pada setiap BABnya terdapat: tujuan, materi, teori, alat dan bahan, prosedur praktikum, percobaan dan latihan. Untuk peserta praktikum diharapkan sudah membaca seluruh bagian dari buku petunjuk praktikum ini sebelum praktikum dimulai dan melakukan praktikum sesuai dengan prosedur yang ditetapkan. Instruktur praktikum mempunyai tugas memberikan arahan tentang pelaksanaan praktikum sekaligus memberikan bimbingan dalam penyelesaian setiap percobaan maupun latihan.

Demikian kata pengantar ini yang mencoba memberikan sedikit arahan tentang tujuan dibuatnya buku petunjuk praktikum ini. Selanjutnya untuk masa yang mendatang akan terus dikembangkan kualitasnya, baik dari segi substansi maupun metode penyusunan dan penyampaiannya. Akhir kata tiada gading yang tak retak, dan demi peningkatan kualitas dari buku petunjuk praktikum ini, kami selalu mengharapkan saran dan kritik membangun dari para pembaca.

Surabaya, Agustus 2005

Team Laboratorium

DAFTAR ISI

Halaman

PERTEMUAN I
DASAR-DASAR LOGIKA

1. Tujuan

a. Mahasiswa mengenal dasar-dasar logika, operasi-operasi yang berlaku dan teknik matematis yang digunakan untuk menyelesaikan persoalan-persoalan logika.
b. Mahasiswa mengenal implementasi gerbang-gerbang ke dalam bentuk hardware (IC / Integrated Circuits)

2. Materi

a. Operasi-operasi logika dasar
b. Tabel kebenaran
c. Gerbang-gerbang logika (Logic Gates)
d. Aljabar Boolean
e. Rangkaian ekivalen
f. Sekilas tentang IC TTL

3. Teori

a. Operasi-operasi logika dasar

Ada beberapa operasi-operasi dasar pada suatu rangkaian logika dan untuk menunjukkan suatu perilaku dari operasi-operasi tersebut biasanya ditunjukkan dengan menggunakan suatu tabel kebenaran. Tabel kebenaran berisi statemen-statemen yang hanya berisi:

- Benar yang dilambangkan dengan huruf "T" kependekan dari "True" atau bisa juga dilambangkan dengan angka 1. atau
- Salah yang dilambangkan dengan huruf "F" kependekan dari "False" atau bisa juga dilambangkan dengan angka 0.

Adapun operasi-operasi dasar logika adalah sebagai berikut:

➢ Operasi INVERS

Operasi invers ini dilambangkan dengan tanda "¯" diatas variabel atau tanda single apostrope " ' ". Operasi invers ini akan mengubah logic benar/1 menjadi logic salah/0 dan begitu pula sebaliknya akan mengubah logic salah/0 menjadi logic benar/1. operasi ini dapat ditunjukkan dengan tabel kebenaran sebagai berikut:

A	$\bar{A}$
0	1
1	0

➢ Operasi AND

Operasi AND dilambangkan dengan dot (.). Operasi ini hanya akan menghasilkan nilai benar jika kedua variabel bernilai benar, selain itu akan bernilai salah. Sehingga operasi ini dapat ditabelkan sebagai berikut:

A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

➢ **Operasi OR**

Operasi OR dilambangkan dengan dot (+). Operasi ini hanya akan menghasilkan nilai benar jika salah satu variabelnya bernilai benar. Sehingga operasi ini dapat ditabelkan sebagai berikut:

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

b. Tabel Kebenaran

Tabel kebenaran adalah tabel yang menunjukkan kombinasi input beserta outputnya pada suatu kasus logika. TABEL KEBENARAN berguna sekali untuk menganalisa suatu fungsi logika. Ada kalanya suatu kasus logika ditunjukkan oleh suatu fungsi logika atau suatu tabel kebenaran. Untuk mempermudah pemahaman perhatikan contoh berikut.

Contoh:
Tunjukkan nilai kebenaran dari suatu fungsi:
F = AB'C + ABC'

Tabel kebenarannya dapat digambarkan sebagai berikut:

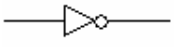
A	B	C	B'	C'	AB'C	ABC'	AB'C + ABC'
0	0	0	1	1	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	1	0	0	0
0	1	1	0	0	0	0	0
1	0	0	1	1	0	0	0
1	0	1	1	0	1	0	1
1	1	0	0	1	0	1	1
1	1	1	0	0	0	0	0

c. Gerbang-gerbang logika (Logic Gates)

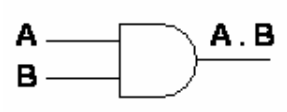
Gerbang-gerbang logika yang khususnya dipakai di dalam komputer digital, dibuat dalam bentuk IC (Integrated Circuit) yang terdiri atas transistor-transistor, diode dan komponen-komponen lainnya. Gerbang-gerbang logika ini mempunyai bentuk-bentuk tertentu yang dapat melakukan operasi-operasi INVERS, AND, OR serta NAND, NOR, dan XOR (Exclusive OR). NAND merupakan gabungan AND dan INVERS sedangkan NOR merupakan gabungan OR dan INVERS.

Masing-masing gerbang tersebut bersama operasinya digambarkan sebagai berikut:

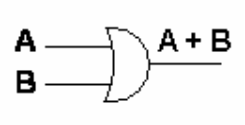
INVERS



AND

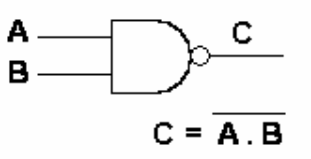


OR



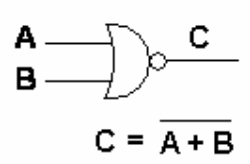
NAND

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



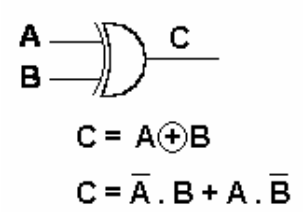
NOR

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



XOR

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



d. Aljabar Boolean

Teori-teori aljabar boolean ini merupakan aturan-aturan dasar hubungan antara variabel-variabel boolean. Aturan ini digunakan untuk memanipulasi dan menyederhanakan suatu rangkaian logika ke dalam bentuk yang bervariasi. Adapun teori-teori aljabar boolean ini dapat kita rangkum menjadi bentuk-bentuk seperti berikut ini:

Dalil-dalil Boolean (Boolean postulates)

P1: X= 0 atau X=1
P2: 0 . 0 = 0
P3: 1 + 1 = 1
P4: 0 + 0 = 0
P5: 1 . 1 = 1
P6: 1 . 0 = 0 . 1 = 0
P7: 1 + 0 = 0 + 1 = 1

Theorema Aljabar Boolean

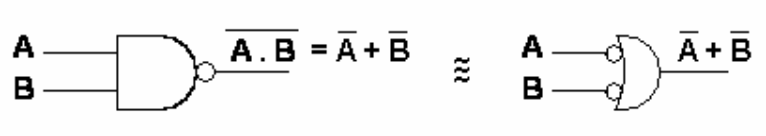
T1: Commutative Law
   a. A + B = B + A
   b. A . B = B . A
T2: Associative Law
   a. (A + B) + C = A + (B + C)
   b. (A . B) . C = A . (B . C)
T3: Distributive Law
   a. A . (B + C) = A . B + A . C
   b. A + (B . C) = (A + B) . (A + C)
T4: Identity Law
   a. A + A = A
   b. A . A = A
T5: Negation Law
   1. (A') = A'
   2. (A')' = A
T6: Redundant Law
   a. A + A . B = A
   b. A . (A + B) = A
T7: 0 + A = A
   1 . A = A
   1 + A = 1
   0 . A = 0
T8: A' + A = 1
   A' . A = 0
T9: A + A' . B = A + B
   A . (A' + B) = A . B
T10: De Morgan's Theorem

   a. $\overline{(A + B)} = \overline{A} . \overline{B}$

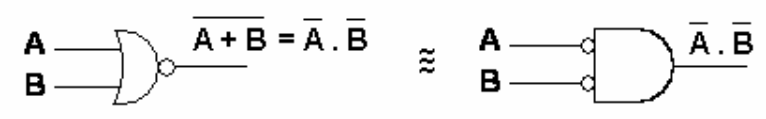
   b. $\overline{(A . B)} = \overline{A} + \overline{B}$

e. Rangkaian ekivalen

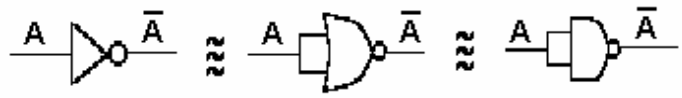
Dalam mendesain rangkaian logika seringkali kita diminta untuk menggunakan gerbang-gerbang NAND atau NOR saja. Untuk memudahkan pelaksanaan desain tersebut , maka diberikan rangkaian ekivalen dari gerbang NAND dan NOR yaitu sebagai berikut:



NAND sama dengan INVERS - OR

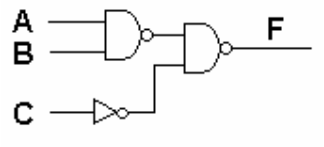


NOR sama dengan INVERS – AND

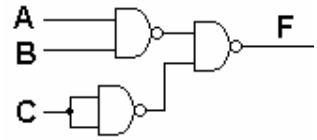


kesamaan INVERS

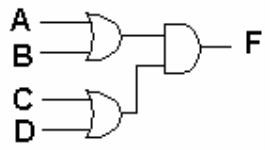
contoh 1.1:
Ubahlah rangkaian dibawah ini menjadi rangkaian yang hanya terdiri dari gerbang NAND saja.



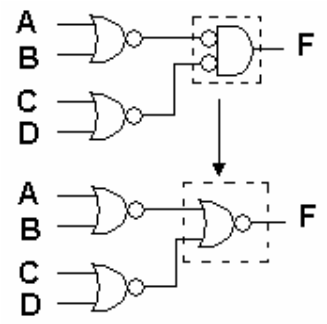
jawab:
karena kesetaraan gerbang INVERS maka rangkaian menjadi:



contoh 1.2:
Ubahlah rangkaian dibawah ini menjadi rangkaian yang hanya terdiri dari gerbang NOR saja.

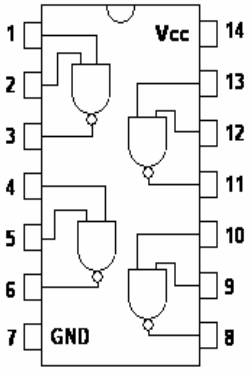


jawab:



f. Sekilas tentang IC TTL

Selama ini kita hanya mengenal symbol-symbol suatu gerbang logika. Di dalam prakteknya suatu gerbang-gerbang logika ini dikemas dalam suatu IC (integrated circuits). Salah satu diantaranya yang terkenal adalah TTL (transistor-transistor logic). Setiap IC TTL ini mempunyai seri-seri tersendiri yang sudah ditetapkan oleh pabrik. Untuk lebih jelasnya berikut ini ada lah salah satu data book dari TTL seri 74 yaitu **SN74LS00**.



SN74LS00

Seri 74LS (low power dengan Scottky-clamp diodes), untuk seri yang sama seperti seri 74L (low power) seri 74H (high power) dan seri 74S (fast speed).
Penggunaan scottky diodes dengan rangkaian transistor paling banyak memberikan transistor switching tercepat karena waktu propagasinya terpendek, sedangkan 74H memboroskan tenaga terbesar dan menangani arus output terbesar. IC TTL ini hanya akan bekerja jika pin-pin power IC tersebut (GND untuk arus minus dan Vcc untuk arus plus) dihubungkan dengan sumber tegangan.

4. Alat dan Bahan

a. Buku praktikum
b. Papan Socket IC
c. IC SN74LS00 dan kabel

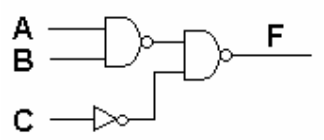
5. Prosedur Praktikum

a. Peserta telah membaca dan mempelajari materi praktikum.

b. Pada awal pertemuan instruktur menerangkan teori dan cara kerja penggunaan IC dan papan socket IC.
c. Jika terjadi alarm pada papan socket IC berarti telah terjadi kesalahan pemasangan rangkaian IC dan segeralah mematikan tombol power pada papan socket IC.

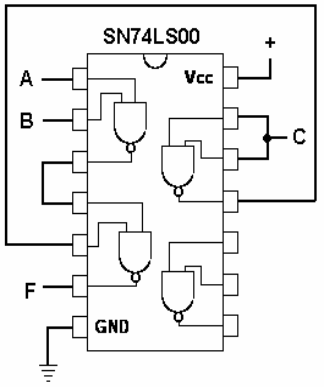
6. Percobaan

Rancanglah dengan menggunakan IC SN74LS00 seminimal mungkin untuk membuat rangkaian berikut ini:



buatlah fungsi logika untuk F dan buatlah tabel kebenarannya. Kemudian cocokkan dengan hasil percobaan anda.

jawaban:
gambar rangkaian:



Fungsi logika:

$$F = \overline{\overline{(A \cdot B)} \cdot \overline{C}}$$

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

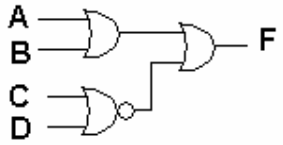
$$F = (A \cdot B) + C$$

Tabel kebenaran:

A	B	C	A.B	(A.B)+C
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	0	1
1	0	0	0	0
1	0	1	0	1
1	1	0	1	1
1	1	1	1	1

7. Latihan

a. Buatlah fungsi logika untuk rangkaian berikut ini kemudian buatlah tabel kebenarannya. Implementasikan rangkaian tersebut hanya dengan menggunakan IC SN74LS00 seminimal mungkin. Periksa kesesuaian tabel anda dengan output pada rangkaian yang anda buat.



Buat pula rangkaian aslinya seperti diatas kemudian bandingkan hasilnya dengan rangkaian yang anda buat dengan menggunakan IC SN74LS00. Hasil outputnya harus sama.

b. Buatlah tabel kebenaran untuk fungsi:

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

Kemudian Implementasikan rangkaian tersebut dengan menggunakan IC SN74LS00 seminimal mungkin. Periksa kesesuaian tabel anda dengan output pada rangkaian yang anda buat.

PERTEMUAN II
ANALISA MASALAH UNTUK MEMBUAT SUATU RANCANGAN LOGIKA

1. Tujuan

a. Mampu mengubah suatu fungsi aljabar menjadi bentuk yang paling sederhana dan menerapkan kedalam tabel kebenaran kemudian mengimplementasi-kannya ke rangkaian kombinasional.
b. Mampu merancang rangkaian kombinasional dari analisa tabel kebenaran.
c. Mampu mendesain rangkaian kombinasional dengan menggunakan IC seminimal mungkin.

2. Materi

a. identifikasi masalah ke dalam tabel kebenaran
b. penyederhanaan fungsi logika dengan K-Map
c. penyelesaian logika dari tabel kebenaran dengan menggunakan metode SOP dan POS dan implementasi pada rancangan rangkaian logikanya.

3. Teori

i. Identifikasi masalah ke dalam tabel kebenaran

Ada kalanya suatu kasus logika disajikan dalam bentuk suatu fungsi logika atau suatu diagram gerbang-gerbang logika yang belum tersaji secara efisien. Oleh sebab itu penting bagi perancang rangkaian logika untuk mengerti bagaimana merancang suatu rangkaian logika dari setiap masalah yang dihadapi.

- **Apa yang dilakukan jika kasus yang disajikan adalah suatu fungsi logika?**

Pada kasus ini kita harus meneliti dahulu apakah fungsi logika yang disajikan tersebut sudah dalam bentuk yang paling sederhana/efisien ataukah masih dalam bentuk yang apa adanya(belum paling sederhana).

- **Apa yang dilakukan jika suatu fungsi sudah dalam bentuk yang paling sederhana?**

Jika kita mempunyai semua gerbang bisa memenuhi semua gerbang logika yang ada pada fungsi tersebut, segeralah merancangnya. Tetapi jika kita ingin mengubah menjadi satu macam type gerbang saja seperti NAND atau NOR kita harus mengubah fungsi tersebut menjadi bentuk seperti berikut:

Untuk membuat rangkaian hanya dari gerbang **NAND**:
Fungsi:

$$F = B\,(A + \bar{C}) + D$$

ubahlah fungsi tersebut menjadi bentuk SOP (Sum of Product), sehingga menjadi:

$$F = AB + B\bar{C} + D$$

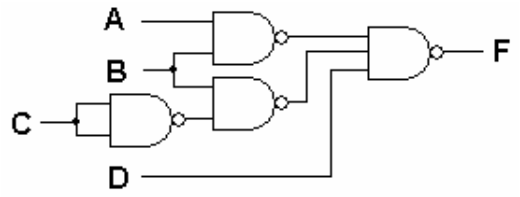
double-bar fungsi tersebut menjadi:

$$F = \overline{\overline{AB + B\bar{C} + D}}$$

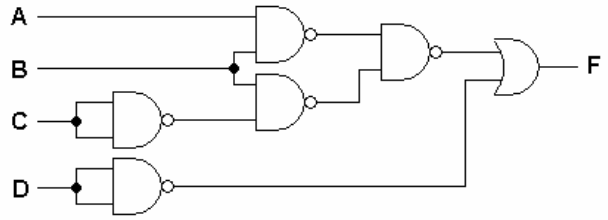
operasikan bar yang terbawah dari double bar, sehingga menjadi:

$$F = \overline{\overline{A}B} \cdot \overline{B\overline{C}} \cdot \overline{D}$$

rangkaian kombinasionalnya:



jika hanya dibuat dari gerbang NAND dengan 2 input saja dan gerbang OR:



Untuk membuat rangkaian hanya dari gerbang **NOR:**
Fungsi:

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

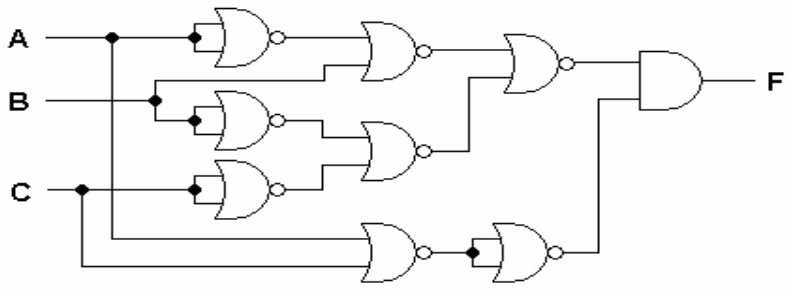
karena sudah dalam bentuk POS, maka langsung double bar fungsi tsb:

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

operasikan bar yang terbawah dari double bar, sehingga menjadi:

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

rangkaian kombinasionalnya jika hanya memakai NOR dengan 2 input:



- **Apa yang dilakukan jika fungsi belum dalam bentuk yang paling sederhana?**

Sederhanakan dahulu fungsi tersebut dengan metode penyederhanaan fungsi aljabar boolean atau dengan menggunakan K-Map (akan dibahas pada sub bab penyederhanaan fungsi logika dengan K-Map). Sesudah itu rancanglah rangkaian kombinasionalnya. Sama seperti proses pembuatan yang sudah dijelaskan diatas.

- **Apa yang akan dilakukan jika kasus disajikan dalam bentuk tabel kebenaran?**

Dengan adanya tabel kebenaran kita bisa berbuat lebih banyak diantaranya:
Membentuk fungsi logika secara efisien (dengan cara penyederhanaan dengan K-Map)
Membuatnya ekspresi fungsi logika ke arah SOP atau POS (akan dibahas pada sub bab penyelesaian logika dari tabel kebenaran dengan menggunakan metode SOP dan POS dan implementasi pada rancangan rangkaian logikanya).

- **Apa yang akan dilakukan jika kasus disajikan dalam bentuk diagram gerbang logika?**

Jika dengan diagram gerbang kita dapat analisa apakah diagram gerbang tersebut sudah benar-benar efisien. Oleh karena itu kita cek dahulu apakah gerbang tersebut sudah cukup sederhana. Caranya dengan menelusuri dahulu diagram gerbang tersebut untuk mendapatkan fungsi logikanya. Jika ekspersi fungsi logikanya sudah sederhana segeralah merancang rangkaian kombinasionalnya. Jika belum sederhana lakukan proses penyederhanan. Bisa dengan penyederhanaan aljabar boolean atau dengan K-Map.

ii. Penyederhanaan fungsi logika dengan K-Map

Salah satu metode penyederhanaan fungsi logika untuk maksimal 4 variabel dapat dilakukan dengan metode K-Map (Karnaugh Map). Sebab jika lebih dari 4 variabel kita menggunakan metode Quine Mc Cluskey. Adapun contoh penyederhanaan fungsi logika dengan menggunakan K-Map adalah sebagai berikut:
Contoh:
Sederhanakan fungsi logika dengan 3 variabel berikut ini :

$$F = A\bar{B} + \bar{A}BC + B\bar{C} + \bar{B}$$

Karena bentuk ekspresi fungsi diatas adalah SOP maka pada matrik K-Map kita letakkan angka 1. Sehingga K-Map tersebut akan tampak seperti:

A \ BC	00	01	11	10
0			1	1
1	1	1	1	1

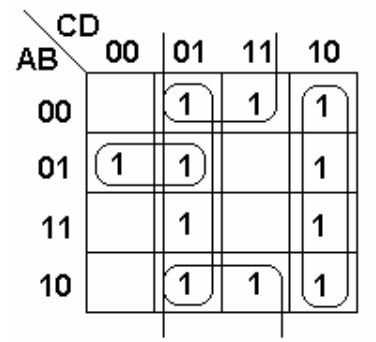
sehingga dari K-Map tersebut didapat penyederhanaan fungsi sebagai berikut:

$$F = A + B$$

Contoh 2.1:
Sederhanakan fungsi logika dengan 4 variabel berikut ini :

$$F = \bar{A}B\bar{C} + \bar{B}D + \bar{A}B\bar{C}D + A\bar{C}D + C\bar{D}$$

Maka K-Map akan berbentuk seperti :



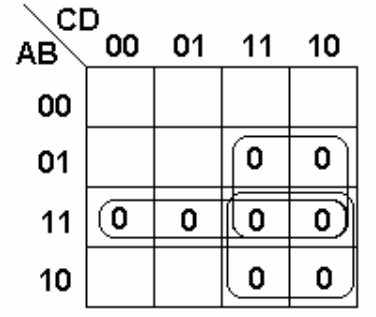
sehingga dari K-Map tersebut didapat penyederhanaan fungsi sebagai berikut:

$$F = \bar{C}D + C\bar{D} + \bar{B}D + \bar{A}B\bar{C}$$

contoh 2.2:
Sederhanakan fungsi logika dengan 4 variabel berikut ini :

$$F = (\bar{A} + \bar{B} + C) . (\bar{A} + \bar{C}) . (\bar{B} + \bar{C})$$

karena bentuk ekspresi fungsi diatas adalah POS, maka kita tempatkan 0 pada K-Map. Sehingga K-Map akan tampak seperti berikut:



hasil penyederhanaan K-Map adalah:

$$F = (\bar{A} + \bar{B}) . (\bar{B} + \bar{C}) . (\bar{A} + \bar{C})$$

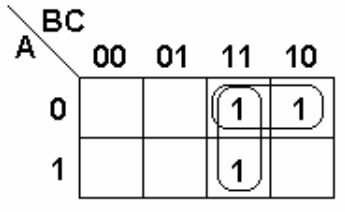
iii. Penyelesaian logika dari tabel kebenaran dengan menggunakan metode SOP dan POS dan implementasi pada rancangan rangkaian logikanya.

Jika diberikan suatu tabel kebenaran dari suatu kasus maka kita bisa menggunakan metode SOP atau POS untuk merancang suatu rangkaian kombinasionalnya. Seperti yang telah dijelaskan diatas. Untuk menentukan suatu rancangan biasanya kita menghendaki suatu rancangan yang paling efisien. Dengan adanya tabel kebenaran kita dapat menentukan mana diantara metode yang paling efisien untuk diimplementasikan. Untuk menentukan metode mana yang paling efisien, kita lihat bagian output pada tabel kebenaran tersebut. Jika jumlah output yang mempunyai nilai 1 lebih sedikit dari jumlah output yang mempunyai nilai 0, maka kita bisa menentukan bahwa metode SOP yang lebih efisien. Jika jumlah output yang mempunyai nilai 0 lebih sedikit dari jumlah output yang mempunyai nilai 1, maka kita bisa menentukan metode POS yang lebih efisien.

Contoh 2.3:
Buatlah rangkaian kombinasional untuk mengimplentasikan tabel kebenaran dibawah ini:

A	B	C	OUTPUT
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Karena output dengan nilai 1 lebih sedikit maka kita gunakan metode SOP. Dan untuk teknik penyederhanaannya kita langsung gunakan K-Map (karena masih 3 variabel). Sehingga K-Map akan berbentuk:



Ekspresi fungsi logikanya dari hasil K-Map tersebut adalah:

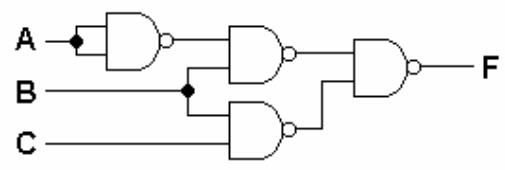
$$F = \bar{A}B + BC$$

Karena bentuk fungsi logikanya adalah SOP kita dapat merancang rangkaian kombinasionalnya dari gerbang NAND saja, yaitu dengan cara memberi double bar pada fungsi tersebut kemudian operasikan bar yang terbawah. Fungsi akan menjadi:

$$F = \overline{\overline{\bar{A}B + BC}}$$

$$F = \overline{\overline{\bar{A}B} \cdot \overline{BC}}$$

Sehingga rangkaian kombinasionalnya menjadi:

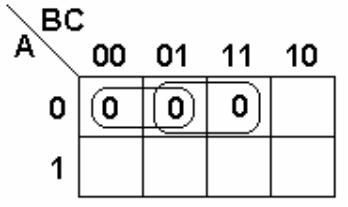


Contoh 2.4:

Buatlah rangkaian kombinasional untuk mengimplentasikan tabel kebenaran berikut ini:

A	B	C	OUTPUT
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Karena output dengan nilai 0 lebih banyak maka kita gunakan metode POS. Sehingga K-Map akan berbentuk:



Ekspresi fungsi logikanya dari hasil K-Map tersebut adalah:

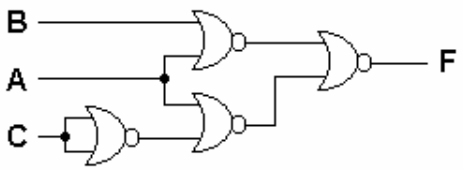
$$F = (A + B) . (A + \bar{C})$$

Dari fungsi logika tersebut kita dapat merancang rangkaian kombinasionalnya dari gerbang NOR saja dengan cara memberi double bar kemudian bar terbawah dioperasikan sehingga:

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

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

Dan rangkaian kombinasionalnya:



Kadangkala suatu hasil dari tabel disajikan dalam bentuk fungsi. Dan kita akan mengenal symbol "Σ" melambangkan operasi SOP sehingga yang ditampilkan adalah output yang mempunyai nilai 1 dan symbol "Π" melambangkan operasi POS sehingga yang ditampilkan adalah ouput yang mempunyai nilai 0.

Contoh 2.5:
F(A, B, C) = Σ (0, 3, 5, 7)

Maksud dari fungsi diatas adalah fungsi tersebut mempunyai 3 variabel input dan output yang mempunyai nilai 1 adalah 0, 3, 5, dan 7 (tanda Σ melambangkan SOP).

Jika fungsi yang disajikan adalah:
F(A, B, C) = Π (0, 3, 5, 7)

Maksudnya adalah fungsi tersebut mempunyai 3 variabel input dan output yang mempunyai nilai 0 adalah 0, 3, 5, dan 7 (tanda Π melambangkan POS).

4. Alat dan Bahan

a. Buku praktikum
b. Papan Socket IC
c. IC SN74LS00 dan kabel

5. Prosedur Praktikum

a. Peserta telah membaca dan mempelajari materi praktikum.
b. Peserta merancang rangkaian pada lembar kerja kemudian diimplementasikan ke dalam papan socket.
c. Jika terjadi alarm pada papan socket IC berarti telah terjadi kesalahan pemasangan rangkaian IC dan segeralah mematikan tombol power pada papan socket IC.

6. Percobaan

Rancanglah dengan menggunakan IC SN74LS00 seminimal mungkin untuk membuat rangkaian kombinasional seperti pada contoh 2.3.

7. Latihan

a. Buatlah rangkaian kombinasional dengan hanya menggunakan gerbang NOR untuk fungsi:
   F(A, B, C, D) = Π (2, 3, 6, 8, 10, 13, 14)
b. Buatlah rangkaian kombinasional untuk mengimplementasikan tabel kebenaran berikut ini dengan menggunakan IC seminimal mungkin. Maksimal 2 macam type gerbang.

A	B	C	OUTPUT 1	OUTPUT 2
0	0	0	0	1
0	0	1	0	1
0	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	0	1	1	0
1	1	0	0	1
1	1	1	0	1

PERTEMUAN III
RANGKAIAN ARITHMATIKA

1. Tujuan

a. Mampu memahami dan merancang rangkaian arithmatika
b. Mampu merancang rangkaian penambahan (adder) dan pengurangan (substractor)

2. Materi

a. Half Adder
b. Full Adder
c. Full Subtractor
d. Bilangan tak bertanda
e. Bilangan bertanda
f. Pengurangan dengan operasi complement

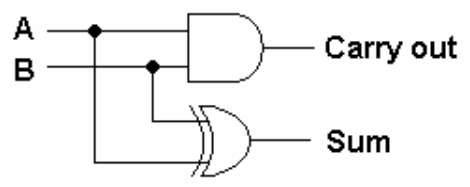
3. Teori

a. Half Adder

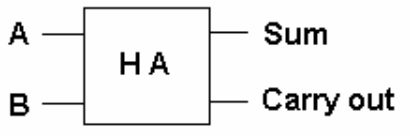
Adalah suatu operasi penjumlahan dua bit biner tanpa menyertakan carry-in nya. Half adder ini dapat dibuat tabel kebenarannya sebagai berikut:

A	B	SUM	Carry Out
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Dari tabel kebenaran tersebut kita dapat merancang rangkaian kombinasionalnya menjadi:



Jika kita buat diagram menurut rangkaian kombinasional diatas half adder tersebut menjadi:

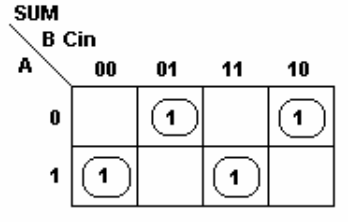


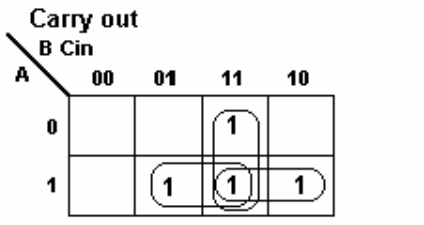
b. Full Adder

Adalah operasi penjumlahan dua bit biner dengan menyertakan carry-in nya. Tabel kebenaran untuk full adder ini adalah sebagai berikut:

A	B	Carry in	Sum	Carry out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Dengan K-Map kita bisa merancang rangkaian full addernya sebagai berikut:



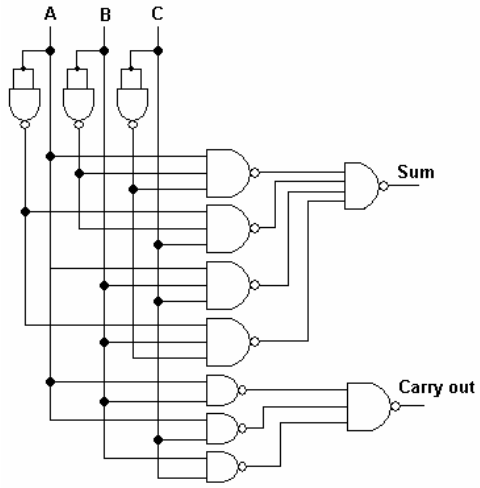


C = Carry in

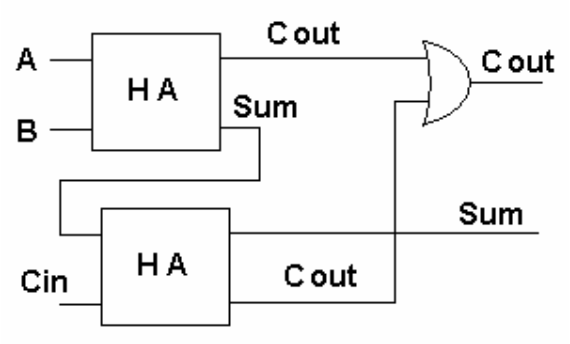
$Sum = A\bar{B}\bar{C} + \bar{A}\bar{B}C + ABC + \bar{A}B\bar{C}$

Carry out = AB + AC + BC

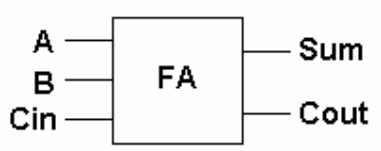
Rangkaian kombinasionalnya:



atau dapat juga rangkaian full adder tersebut dibuat dari 2 buah rangkaian half adder. Sehingga bentuknya menjadi seperti berikut:



Rangkaian tersebut dapat dibuat diagram logikanya menjadi:

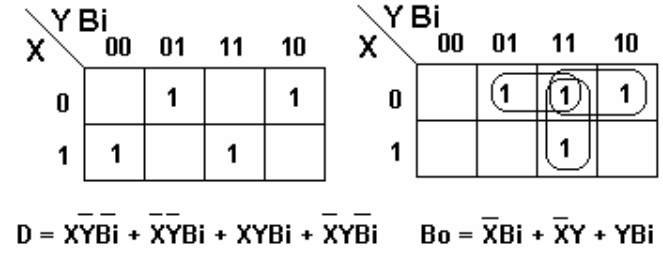


c. Full substractor

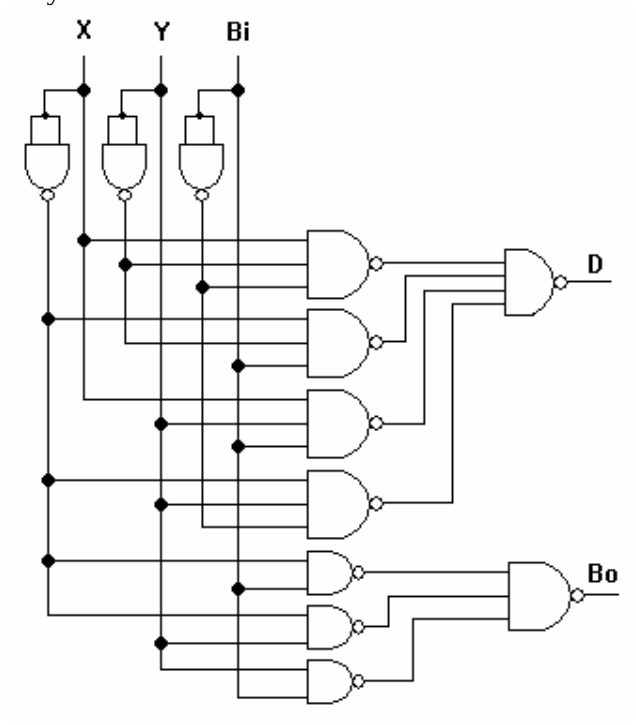
Rangkaian logika lainnya yang dapat dikelompokkan sebagai unit rangkaian arithmatika adalah full substractor. Full substractor ini merupakan operasi pengurangan dua bit biner yang mengikutsertakan borrow-in nya di dalam operasi pengurangannya. Tabel kebenaran dari full substractor ini adalah sebagai berikut:

X	Y	Bin	D	Bout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

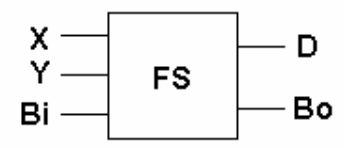
Dengan menggunakan K-Map untuk mencari fungsi logika yang paling sederhana.



Rangkaian logikanya adalah:



Rangkaian diatas dapat dibuat diagram logikanya sebagai berikut:



d. Bilangan tak bertanda

Pada beberapa aplikasi kadangkala tidak memperhatikan tanda negatif dan positip, karena hanya berkonsentrasi pada besaran nilai. Sebagai contoh pada angka biner 8 bit nilai besaran yang terkecil adalah 0000 0000 dan yang terbesar adalah 1111 1111. Maka dari itu total kisaran angka biner 8 bit adalah dari 0000 0000 (00H) s/d 1111 1111 (FFH). Besaran nilai biner 8 bit setara dengan decimal dari 0 s/d 255. seperti yang anda lihat bahwa kita tidak menyertakan tanda + dan – pada bilangan decimal tersebut.

Jika kita menggunakan 16 bit, maka kisaran biner adalah:
0000 0000 0000 0000 (0000H) s/d 1111 1111 1111 1111 (FFFFH).
Yang setara dengan bilangan decimal dari 0 s/d 65.535

Contoh 3.1:
Penjumlahan bilangan decimal 150 dan 85 dengan menggunakan 8 bit.

Jawab:
150 → 1001 0110 → 96H
85 → 0101 0101 → 55H

	1001 0110			96H
+	0101 0101		+	55H
	1110 1011			EBH

1110 1011 → EBH → 235

Contoh 3.2:
Pengurangan bilangan decimal 150 dan 85 dengan menggunakan 8 bit.
Jawab:

	1001 0110			96H
-	0101 0101		-	55H
	0100 0001			41H

0100 0001 → 41H → 65

- Batasan-batasan

Microcomputer generasi pertama hanya dapat memproses 8 bit dalam satu waktu. Oleh karena itu semua operasi-operasi arithmatika (baik penjumlahan atau pengurangan) haruslah menghasilkan besaran pada kisaran 0 s/d 255. Jika suatu besaran lebih besar dari 255, berarti kita harus menggunakan operasi arithmatika 16 bit. 8 bit pertama baru kemudian 8 bit berikutnya.

- *Overflow*

Pada penjumlahan 8 bit antara dua bilangan tak bertanda yang menghasilkan besaran lebih besar dari 255 akan menyebabkan overflow yaitu carry pada bit ke-9. Pada sebagian besar microprocessor mempunyai rangkaian logic yang disebut carry flag. Rangkaian ini mendeteksi sebuah carry yang berada pada bit ke-9 dan memberi tanda bahwa hasil yang diperoleh pada operasi 8 bit tersebut invalid.

Contoh 3.3:
Tunjukkan dengan menggunakan operasi 8 bit untuk 175_{10} ditambah 118_{10}.

Jawab:

	175
+	118
	293

Karena jawabannya lebih dari 255, yang akan terjadi jika kita menggunakan operasi 8 bit adalah:

	AFH				1010 1111
+	76H			+	0111 0110
	125H		Overflow →		1 0010 0101

Karena 8 bit, maka hanya hasil 8 bit kebawah saja yang digunakan. Sehingga:
0010 0101 → 25H → 37
seperti yang anda lihat, bahwa jawabannya salah.

e. Bilangan bertanda

Bilangan bertanda sangat penting artinya untuk operasi arithmatika yang lebih kompleks. Angka-angka decimal –1, -2, -3 dst yang menunjukkan besaran negatif akan sama jika dilambangkan dengan bilangan biner -001, -010, dan -011 secara berurutan. Karena semua kode harus dilambangkan dengan 0 dan 1, maka tanda + dilambangkan dengan 0 dan – dilambangkan dengan 1. Sehingga -001, -010, dan -011 dikodekan sebagai 1001, 1010 dan 1011.

Pada angka-angka selanjutnya, MSB selalu menunjukkan tanda dan bit sisanya adalah besaran angka. Berikut ini adalah contoh konversi besaran bilangan bertanda:

+7 → 0000 0111
-16 → 1001 0000
+25 → 0000 0000 0001 1001
-128 → 1000 0000 1000 0000

- **Kisaran besaran bilangan bertanda**

Seperti telah kita ketahui, bilangan tak bertanda 8 bit dapat mewakili 0 s/d 255. Jika kita menggunakan besaran bertanda nilai yang diwakili menjadi berkurang dari 255 menjadi 127, karena 1 bit dipergunakan sebagai wakil dari tanda besaran (+ atau -).

Contoh 3.4:
1000 0001 → (-1) 0000 0001 → (+1)
1111 1111 → (-127) 0111 1111 → (+127)

- **1'S Complement**

adalah bilangan biner yang dihasilkan dari menginvers-kan setiap bit-nya. 1'S complement dari biner 1000 adalah 0111

- **2'S Complement**

adalah bilangan biner yang dihasilkan dari 1'S complement ditambah 1. jika dibuat rumus menjadi:

2'S complement = 1'S complement + 1

sebagai contoh, 2'S complement dari 1011 adalah:

1011 → 0100 (1'S complement)
0100 + 1 → 0101 (2'S complement)

- **Odometer biner**

Adalah cara yang baik sekali untuk memahami gambaran tentang 2'S complement. Pada umumnya microcomputer menggunakan 2'S complement untuk menggambarkan bilangan negatif dan positif.

Seperti telah kita ketahui bahwa tanda + diwakili dengan 0 dan – diwakili dengan 1 pada MSB bit-bit biner. Pada odometer, bilangan negatif merupakan 2'S complement dari bilangan positifnya, sehingga:

Besaran	Positif	Besaran	Negatif
1	0001	-1	1111
2	0010	-2	1110
3	0011	-3	1101

4	0100	-4	1100
5	0101	-5	1011
6	0110	-6	1010
7	0111	-7	1001
8	-	-8	1000

Kecuali untuk bilangan terakhir, bilangan negatif merupakan 2'S complement dari bilangan positifnya.

f. Pengurangan dengan complement

Pengurangan dua buah bilangan adalah sama dengan penjumlahan antara bilangan yang dikurangi dengan complement pengurangnya. Pada microcomputer selalu menggunakan biner untuk melakukan operasi arithmatika.

Contoh 3.5:

Hitunglah : 83 – 16 dengan menggunakan 8 bit.

Biner setiap bilangan:

83 → 0101 0011

16 → 0001 0000

16 akan dikirimkan 2'S complement-nya (karena minus) menjadi :

-16 → 1111 0000

sehingga :

83		0101 0011	
- 16	+	1111 0000	
67	Carry don't care	1 0100 0011	→ 0100 0011 → 43H → 67

Contoh 3.6:

Hitunglah : 14 – 108 dengan menggunakan 8 bit.

Biner setiap bilangan:

+14 → 0000 1110

+108 → 0110 1100

108 akan dikirimkan 2'S complement-nya (karena minus) menjadi :

-108 → 1001 0100

sehingga:

14		0000 1110	
- 108	+	1001 0100	
-94	Tanpa carry	1010 0010	→ A2H → -94

4. Alat dan Bahan

a. Buku praktikum
b. Papan Socket IC
c. IC TTL dan kabel

5. Prosedur Praktikum

a. Peserta telah membaca dan mempelajari materi praktikum.
b. Peserta merancang rangkaian pada lembar kerja kemudian diimplementasikan ke dalam papan socket.

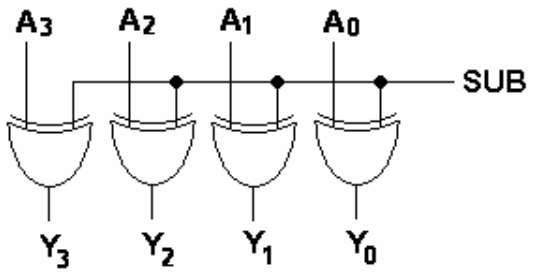
c. Jika terjadi alarm pada papan socket IC berarti telah terjadi kesalahan pemasangan rangkaian IC dan segeralah mematikan tombol power pada papan socket IC.

6. Percobaan

Buatlah adder sekaligus substractor 4 bit untuk menjumlahkan dan mengurangkan dua bilangan biner 4 bit.

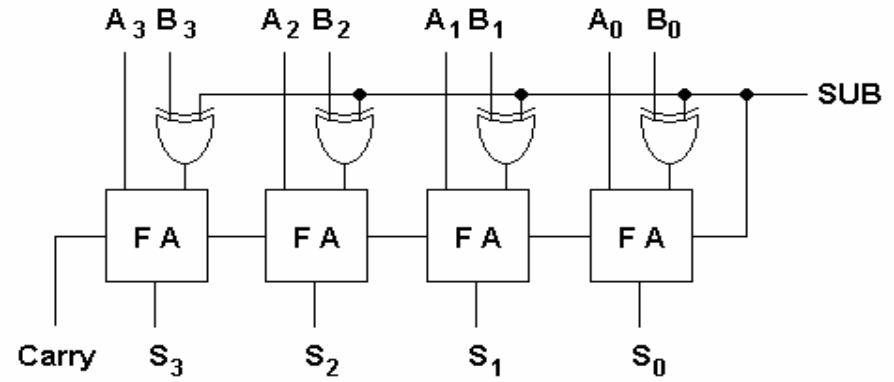
	A_3	A_2	A_1	A_0			A_3	A_2	A_1	A_0
+	B_3	B_2	B_1	B_0		-	B_3	B_2	B_1	B_0
	S_3	S_2	S_1	S_0			S_3	S_2	S_1	S_0

Rangkaian diagram logika 1'S complement untuk 4 bit:



jika SUB bernilai 0 maka $Y_3Y_2Y_1Y_0 = A_3A_2A_1A_0$ dan jika SUB bernilai 1 maka $Y_3Y_2Y_1Y_0 = A_3A_2A_1A_0$.

Rangkaian diagram Adder-Substractor (aplikasi paralel full adder) adalah:



7. Latihan.

a. Buatlah rangkaian gerbang logikanya untuk pengurangan 4 bit dengan menggunakan paralel full substractor.

b. Buatlah rangkaian yang outputnya adalah multiplication (perkalian) dengan bilangan 2 dari bilangan yang diinputkan. Jumlah input dan output adalah 4 bit.

PERTEMUAN IV
DATA PROCESSING CIRCUITS
(RANGKAIAN PEMROSES DATA)

1. Tujuan

Mahasiswa dapat mempelajari dan mengimplementasikan rangkaian logika untuk memproses data-data biner.

2. Materi

a. Multiplexer
b. Demultiplexer
c. Decoder
d. Encoder

3. Teori

a. MULTIPLEXER

Multiplexer adalah suatu rangkaian yang mempunyai banyak input dan hanya mempunyai satu output. Dengan menggunakan selector, kita dapat memilih salah satu inputnya untuk dijadikan output. Sehingga dapat dikatakan bahwa multiplexer ini mempunyai n input, m selector , dan 1 output. Biasanya jumlah inputnya adalah 2^m selectornya. Adapun macam dari multiplexer ini adalah sebagai berikut:

- Multiplexer 4x1 atau 4 to 1 multiplexe r
- Multiplexer 8x1 atau 8 to 1 multiplexer
- Multiplexer 16x1 atau 16 to 1 multiplexer dsb.

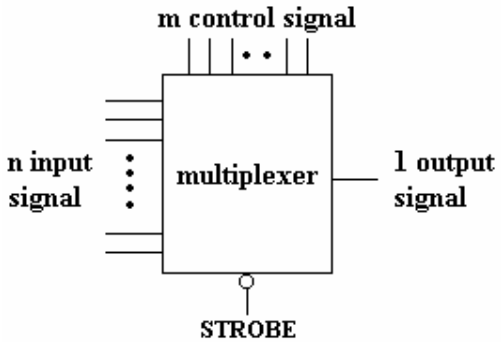
Gambar 4.1. berikut adalah symbol dari multiplexer 4x1 yang juga disebut sebagai "data selector" karena bit output tergantung pada input data yang dipilih oleh selector. Input data biasanya diberi label D_0 s/d Dn. Pada multiplexer ini hanya ada satu input yang ditransmisikan sebagai output tergantung dari kombinasi nilai selectornya. Kita misalkan selectornya adalah S_1 dan S_0, maka jika nilai :

$$S_1\ S_0 = 00$$

Maka outputnya (kita beri label Y) adalah :

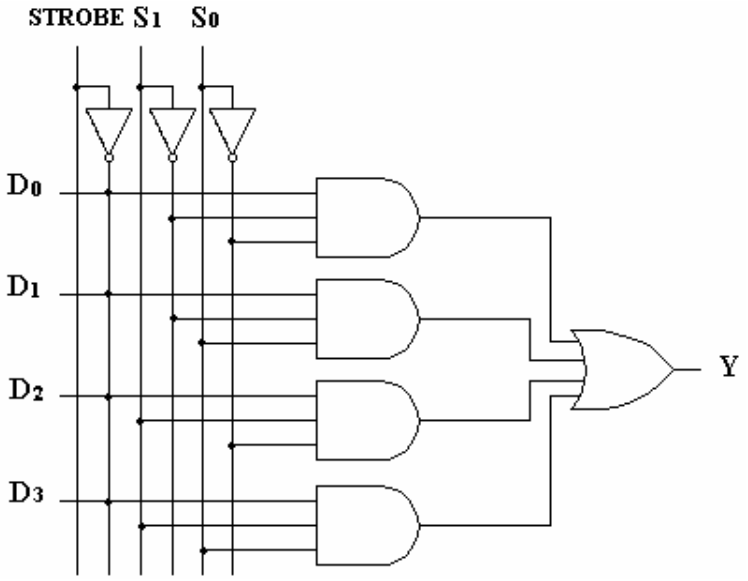
$$Y = D_0$$

Jika D_0 bernilai 0 maka Y akan bernilai 0, jika D_0 bernilai 1 maka Y akan bernilai 1.



gb.4.1. symbol multiplexer

Adapun rangkaian multiplexer 4x1 dengan menggunakan strobe atau enable yaitu suatu jalur bit yang bertugas mengaktifkan atau menonaktifkan multiplexer, dapat kita lihat pada gambar 4.2 berikut ini.



Gambar.4.2. Rangkaian multiplexer 4x1

Tabel. 4.1. True table multiplexer 4x1

Strobe	S_1	S_0	Output
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3
1	X	X	0

- **LOGIKA MULTIPLEXER DAN IMPLEMENTASI FUNGSI BOOLEAN**

Suatu desain dari rangkaian logic biasanya dimulai dengan membuat tabel kebenaran. Seperti telah kita ketahui bahwa kita mengenal ada 2 macam metode yang diterapkan pada tabel kebenaran, yaitu metode *sum of product* (SOP) dan metode *product of sum* (POS). Nah pada bagian ini kita kenalkan dengan metode yang ketiga yaitu *multiplexer solution.*

Pada kenyataannya, kita dapat merancang suatu multiplexer 8x1 dari multiplexer 4x1 atau multiplexer 16x1 dari multiplexer 8x1 dan seterusnya. Jika kita anggap **selector** sebagai **n**, maka kita dapat membuat multiplexer 2^nx1 dari multiplexer 2^{n-1}x1. Dengan kata lain kita memfungsikan multiplexer 2^{n-1}x1 sebagai multiplexer 2^nx1.

Jika kita menterjemahkan suatu kasus sebagai suatu fungsi F :

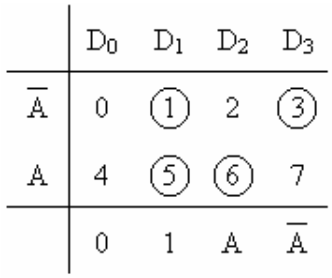
$$F(A, B, C) = \sum (1, 3, 5, 6)$$

Dimana parameter fungsi tersebut A, B, C adalah merupakan selector dari multiplexer dan sisi sebelah kanan fungsi adalah output yang diinginkan dari multiplexer. Tanda $\sum$ beserta parameter berikutnya adalah merupakan bentuk SOP

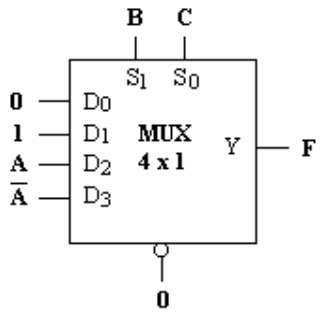
(sum of product), dimana hal ini menandakan hanya minterm yang mempunyai nilai 1 saja yang diikutkan sebagai parameter. Tabel kebenaran dari fungsi diatas adalah sebagai berikut:

Tabel 4.2.

Minterm	A	B	C	F
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0



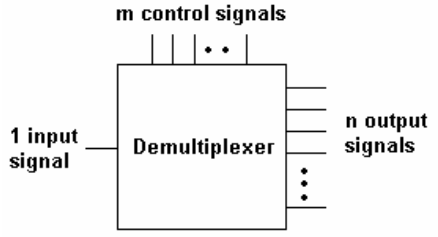
(b) implementasi tabel



(c) Implementasi multiplexer.

b. DEMULTIPLEXER

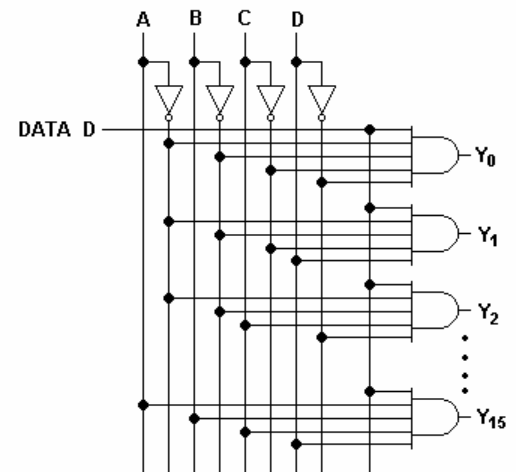
Demultiplexer berarti satu ke banyak. Sebuah demultiplexer adalah suatu rangkaian logic yang mempunyai satu input dan mempunyai banyak output. Dengan menggunakan control signal, kita dapat mengarahkan input signal ke salah satu outputnya. Gambar 4.3 mengilustrasikan ide dasar dari demultiplexer yang mempunyai 1 input signal, *m* control signal, dan *n* output signal.



Gambar 4.3 Demultiplexer.

➢ **1 TO 16 DEMULTIPLEXER**

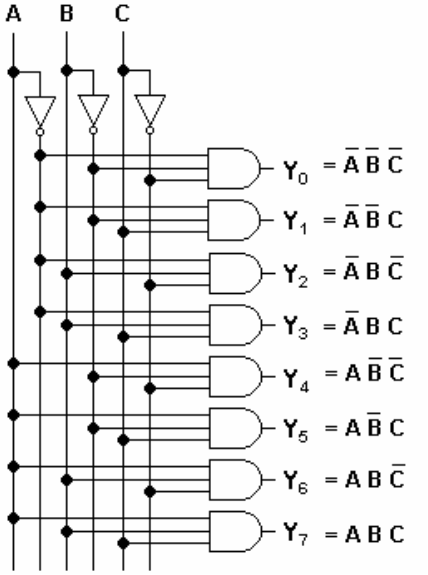
Gambar 4.4 menunjukkan 1 to 16 demultiplexer. Input diberi label *D*. Bit data *D* ditransmisikan ke output tergantung pada nilai input control *ABCD*. Jika *ABCD* bernilai 0000, maka gerbang AND teratas enable/aktif dan gerbang AND lainnya akan disable/ tidak aktif. Oleh karena itu bit data *D* hanya ditransmisikan ke output Y_0, sehingga $Y_0=D$. Jika *D* bernilai 0, maka Y_0 bernilai 0. Jika *D* bernilai 1, maka Y_0 bernilai 1. Jika input control bernilai 1111, maka semua gerbang AND akan disable kecuali gerbang AND terbawah. Kemudian *D* hanya diransmisikan ke output Y_{15}, dan $Y_{15}=D$.



Gambar 4.4 Rangkaian 1 to 16 demultiplexer.

c. DECODER

Jika kita perhatikan decoder ini sebenarnya mirip dengan demultiplexer, dengan satu pengecualian yaitu pada decoder ini tidak mempunyai data input. Input hanya digunakan sebagai data control.



Gambar 4.5 Rangkaian 1 to 8 decoder.

Dalam hal ini adalah *ABCD*. Seperti yang ditunjukkan pada gambar 4.5, circuit logic ini disebut *1 of 8 decoder*, karena hanya 1 dari 8 jalur output yang bernilai 1. Sebagai contoh, ketika *ABCD* = 0001, maka hanya output Y_1 yang akan bernilai 1. Begitu juga jika *ABCD* = 0100, maka hanya output Y_4 yang mempunyai output 1 dan seterusnya.

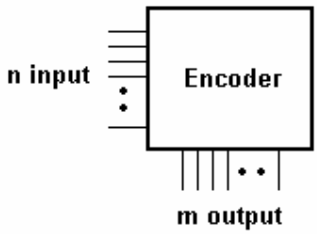
Operasi pada decoder dapat dijelaskan lebih lanjut dari hubungan input-output, seperti pada tabel 4.3 Amatilah pada variabel output yang mana, satu sama lainnya saling eksklusif, karena hanya ada satu output yang bernilai 1 pada satu waktu. Jalur output ditunjukkan dengan minterm yang ekivalen dengan angka biner.

Tabel 4.3 Tabel kebenaran 1 to 8 decoder.

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

d. ENCODER

Sebuah encoder mengkonversikan input signal yang aktif menjadi output signal yang dikodekan. Pada gambar 4.6 mengilustrasikan suatu encoder. Dimana ada sejumlah n jalur input, dan hanya salah satunya yang aktif. Internal logic di dalam encoder mengkonversikan input yang aktif menjadi output kode-kode biner sebanyak m bit.



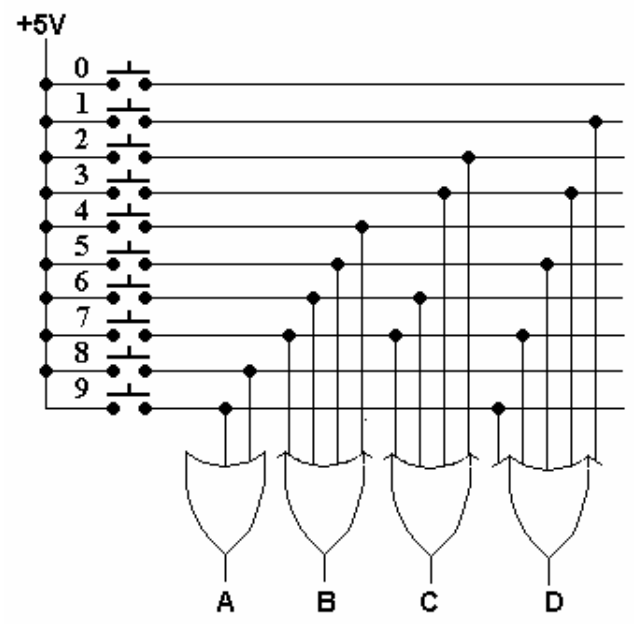
Gambar 4.6 Enc oder.

➢ **DECIMAL TO BCD ENCODER**

Gambar 4.7 menunjukkan suatu type encoder yang sudah umum yaitu decimal to BCD encoder. Switch dengan penekan tombol mirip dengan tombol kalkulator dihubungkan dengan tegangan Vcc. Jika tombol 3 ditekan, maka gerbang-gerbang OR pada jalur C dan D akan mempunyai input bernilai 1. Oleh karena itu maka outputnya menjadi :

ABCD = 0011

Dan seterusnya.



Gambar 4.7 Decimal to BCD encoder.

4. Alat dan Bahan

a. Buku praktikum
b. Papan Socket IC
c. IC TTL dan kabel

5. Prosedur Praktikum

a. Peserta telah membaca dan mempelajari materi praktikum.
b. Peserta merancang rangkaian pada lembar kerja kemudian diimplementasikan ke dalam papan socket.
c. Jika terjadi alarm pada papan socket IC berarti telah terjadi kesalahan pemasangan rangkaian IC dan segeralah mematikan tombol power pada papan socket IC.

6. Percobaan

Buatlah rangkaian kombinasional untuk implementasi 1 to 8 demultiplexer. Ubahlah rangkaian pada gambar 4.4 (1 to 16 demultiplexer) untuk menjadi 1 to 8 demultiplexer.

7. Latihan

a. Buatlah implementasi full adder dengan menggunakan 1 to 8 decoder dan 2 buah gerbang OR
b. Buatlah BCD to DECIMAL decoder

PERTEMUAN V
RANGKAIAN SEQUENTIAL

1. Tujuan

Mahasiswa dapat mengerti cara kerja rangkaian sequential beserta aplikasinya

2. Materi

a. Flip-flop
b. Rangkaian dasar flip-flop
c. RS flip-flop dengan clock
d. D flip-flop
e. JK flip-flop
f. T flip-flop
g. Tabel eksitasi flip-flop
h. Procedure desain

3. Teori

Pada dasarnya rangkaian logika dibagi menjadi dua jenis, yaitu rangkaian kombinasional yang telah kita pelajari pada bab sebelumnya dan rangkaian sequential.

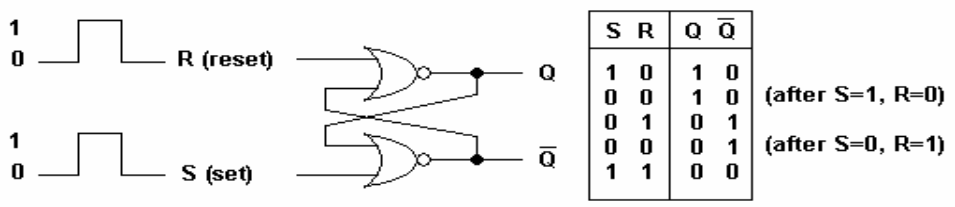
- Rangkaian kombinasional adalah suatu rangkaian yang outputnya hanya tergantung pada kombinasi inputnya.
- Rangkaian sequential adalah suatu rangkaian yang outputnya tidak hanya tergantung pada kombinasi inputnya tetapi juga tergantung pada output sebelumnya.

a. FLIP-FLOP

Adalah suatu rangkaian yang dapat menyimpan state biner (sepanjang masih terdapat power pada rangkaian) sampai terjadi perubahan pada sinyal inputnya.

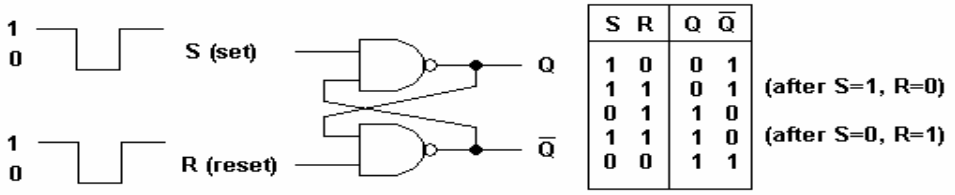
b. RANGKAIAN DASAR FLIP-FLOP

Flip-flop dapat dibuat dari dua buah gerbang NAND atau NOR berikut ini:



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

Gambar 5.1 Rangkaian dasar flip-flop dengan gerbang NOR

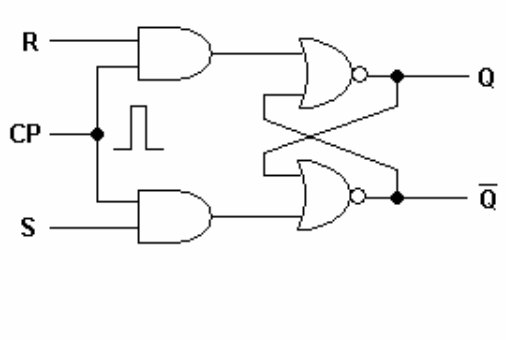


S	R	Q	$\overline{Q}$	
1	0	0	1	
1	1	0	1	(after S=1, R=0)
0	1	1	0	
1	1	1	0	(after S=0, R=1)
0	0	1	1	

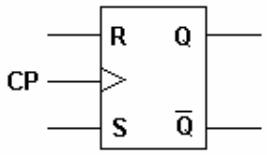
Gambar 5.2 Rangkaian dasar flip-flop dengan gerbang NAND.

c. RS FLIP-FLOP DENGAN CLOCK

Dengan menambah beberapa gerbang pada bagian input rangkaian dasar, flip-flop tersebut hanya dapat merespon input selama terdapat clock pulsa. Output dari flip-flop tidak akan berubah selama clock pulsanya 0 meskipun terjadi perubahan pada inputnya. Output flip-flop hanya akan berubah sesuai dengan perubahan inputnya jika clock pulsa bernilai 1.



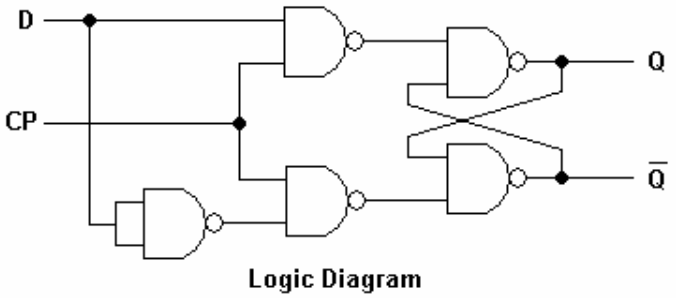
Q	S	R	Q (t + 1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	indeterminate
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	indeterminate

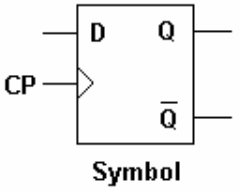


Gambar 5.3 RS flip-flop dengan clock.

d. D FLIP -FLOP

D flip-flop merupakan modifikasi dari RS flip-flop memakai clock. Input D disalurkan secara langsung ke S.





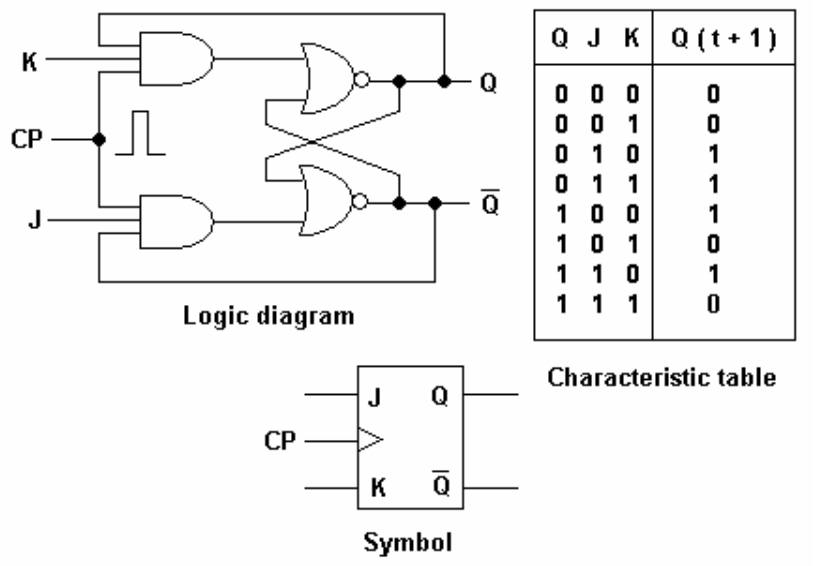
Q	D	Q (t + 1)
0	0	0
0	1	1
1	0	0
1	1	1

Characteristic table

Gambar 5.4 D flip-flop dengan clock.

e. JK FLIP-FLOP

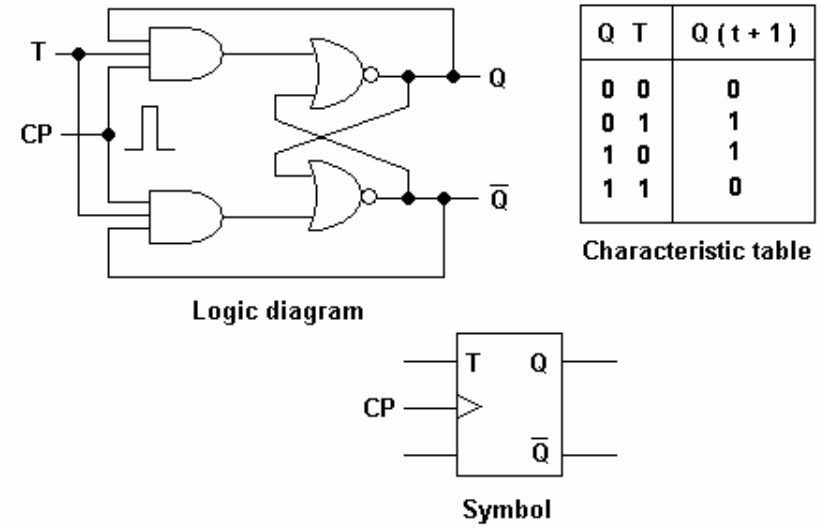
State-state yang tidak didefinisikan pada RS flip-flop, pada JK flip-flop ini state tersebut didefinisikan. Jika pada RS flip-flop kondisi R dan S sama dengan 1, maka kondisi seperti ini tidak didefinisikan, maka pada JK flip-flop jika kondisi J dan K sama dengan 1 maka output JK flip-flop tersebut adalah komplemen dari output sebelumnya. Dalam hal ini J setara dengan S dan K setara dengan R. untuk lebih jelasnya kita perhatikan diagram dibawah ini.



Gambar 5.5 JK flip-flop dengan clock.

f. T FLIP-FLOP

Adalah versi JK flip-flop dengan single input. T flip-flop mempunyai kemampuan yaitu membuat toggle seperti pada tabel dibawah ini..



Gambar 5.6 T flip-flop dengan clock.

g. TABEL EKSITASI FLIP-FLOP

Dibawah ini adalah karakteristik tabel dari berbagai type flip-flop. Nilai X menandakan bahwa nilainya dapat diisi kedua-duanya yaitu 0 dan 1.

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

(a) RS flip-flop

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

(b) JK flip-flop

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

(c) D flip-flop

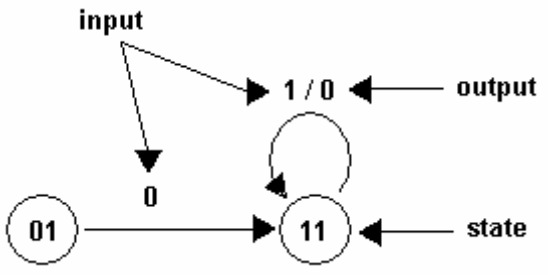
Q (t)	Q (t+1)	T
0	0	0
0	1	1
1	0	1
1	1	0

(d) T flip-flop

h. PROCEDURE DESAIN

Apabila kita akan membuat suatu rangkaian sequential dengan clock biasanya dimulai dari kumpulan spesifikasi rangkaian dalam bentuk diagram state sehingga nantinya didapatkan daftar fungsi boolean. Berbeda dengan rangkaian kombinasional yang sepenuhnya dapat dibuat dari representasi tabel kebenaran, rangkaian sequential ini harus dibuat dahulu diagram statenya agar dapat diketahui tahap-tahap state yang seharusnya diproses, sehingga kita dapat menentukan rangkaian kombinasionalnya.

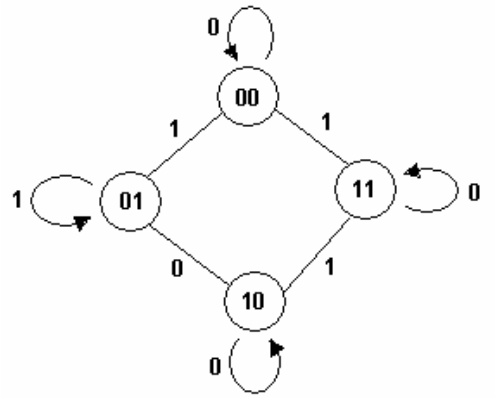
State diagram mempunyai bentuk:



Gambar 5.7 Konvensi state diagram.

Contoh 5.1 :

Buatlah rangkaian sequential dengan menggunakan JK flip-flop untuk state diagram berikut ini:



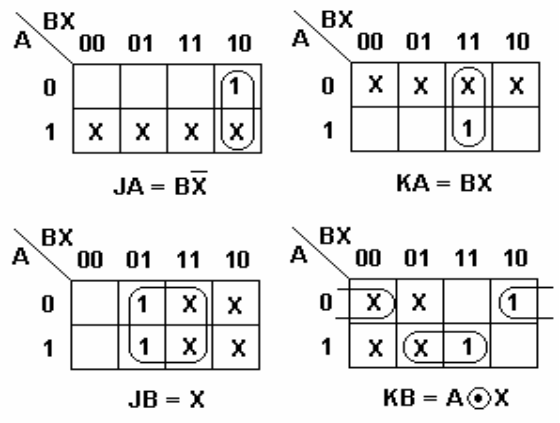
Tabel 5.1 Tabel state

Present state		Next state			
		X = 0		X = 1	
A	B	A	B	A	B
0	0	0	0	0	1
0	1	1	0	0	1
1	0	1	0	1	1
1	1	1	1	0	0

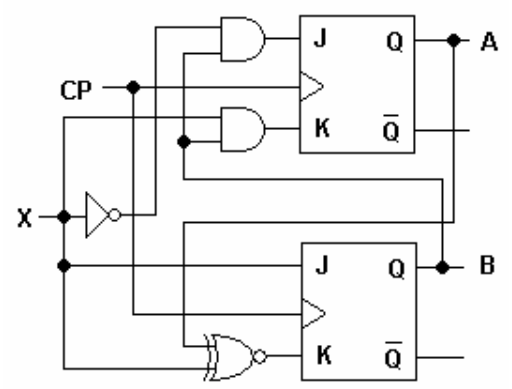
Tabel 5.2 tabel eksitasi

Input dari rangkaian kombinasional Present state		input	Next state		Output dari rangkaian kombinasional Input flip-flop			
A	B	X	A	B	JA	KA	JB	KB
0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	X	1	X
0	1	0	1	0	1	X	X	1
0	1	1	0	1	0	X	X	0
1	0	0	1	0	X	0	0	X
1	0	1	1	1	X	0	1	X
1	1	0	1	1	X	0	X	0
1	1	1	0	0	X	1	X	1

Dari tabel eksitasi tersebut dapat dibuat K-Map untuk setiap input flip-flop yaitu:



Rangkaian sequentialnya adalah:



4. Alat dan Bahan

e. Buku praktikum
f. Papan Socket IC
g. IC TTL dan kabel

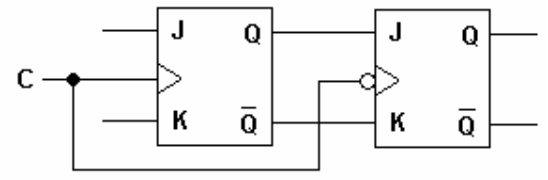
5. Prosedur Praktikum

a. Peserta telah membaca dan mempelajari materi praktikum.
b. Peserta merancang rangkaian pada lembar kerja kemudian diimplementasikan ke dalam papan socket.
c. Jika terjadi alarm pada papan socket IC berarti telah terjadi kesalahan pemasangan rangkaian IC dan segeralah mematikan tombol power pada papan socket IC.

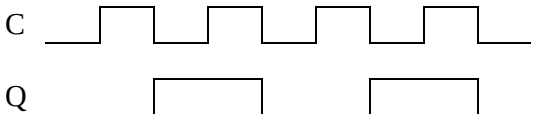
6. Percobaan

Buatlah rangkaian kombinasional untuk merancang suatu master slave flip-flop dengan menggunakan JK flip-flop.

Rangkaian diagram:



waveform:



7. Latihan

a. Suatu rangkaian sequential terdiri dari dua buah flip-flop (A dan B) sebuah input x dan sebuah output y. Input dan output flip-flop mempunyai fungsi-fungsi sebagai berikut:

$$JA = x\overline{B} + B$$

$$KA = \overline{x}B$$

$$JB = \overline{A}$$

$$KB = \overline{x} + A$$

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

PERTEMUAN VI
COUNTER

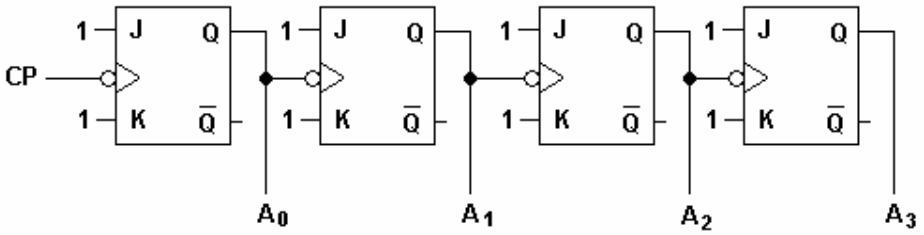
1. Tujuan
Mahasiswa dapat mempelajari dan mengimplementasikan rangkaian counter (pencacah)

2. Materi
a. Ripple counter
b. Modulus counter
c. Synchronous counter

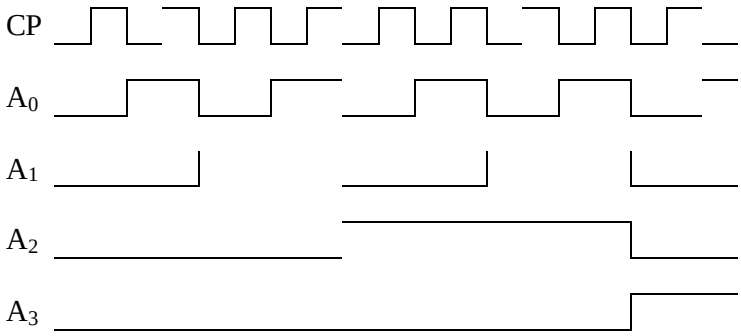
3. Teori

a. RIPPLE COUNTER
Selain register digunakan untuk transfer data, register juga mempunyai kegunaan lain yaitu sebagai counter (pencacah). Counter dapat dibedakan menjadi dua macam, yang pertama adalah Ripple counter dimana output flip-flop yang satu digunakan sebagai clock pulsa pada flip-flop lainnya. Sedangkan synchronous counter semua flip-flop mempunyai clock pulsa yang sama.

➢ BINARY RIPPLE COUNTER
Binary ripple counter terdiri dari flip-flop complement (T flip-flop dan JK flip-flop) yang dirangkai secara seri dengan setiap output flip-flop dihubungkan pada input CP flip-flop yang mempunyai urutan lebih tinggi.



Gambar 6.1 Binary ripple counter 4 bit.



Tabel 6.1 Tabel kebenaran binary ripple counter

A_3	A_2	A_1	A_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0

b. MODULUS COUNTER

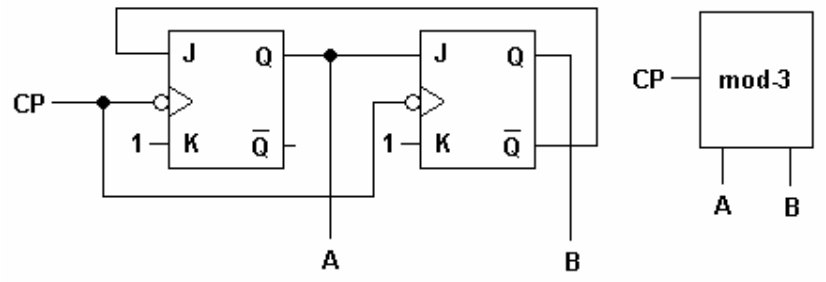
Rangkaian counter-counter diatas hanya akan kembali ke keadaan semula (reset) dalam 2n pulsa. Apabila kita menginginkan suatu flip-flop akan reset setelah hitungan tertentu (selain hitungan 2n) maka counter ini disebut sebagai modulus counter.

- **COUNTER MOD-3**

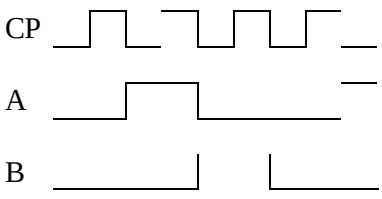
Berikut ini adalah counter mod 3 dengan menggunakan JK flip-flop.

Tabel 6.2 tabel kebenaran

B	A	count
0	0	0
0	1	1
1	0	2
0	0	0

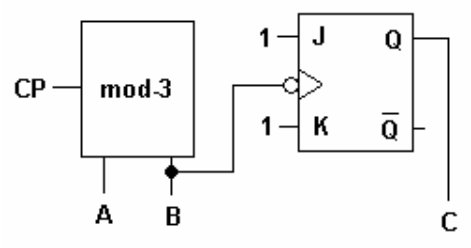


Gambar 6.2 Diagram logika dan logic blok

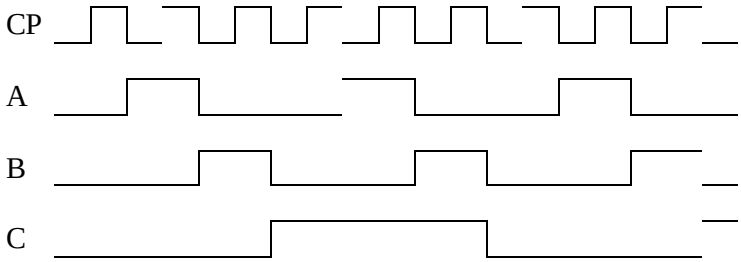


➢ **COUNTER MOD-6**

Dasar dari flip-flop adalah counter mod-2. Sehingga jika kita ingin membuat counter mod-4 kita dapat membangunnya dari dua buah mod-2 yang dihubungkan secara seri. Untuk counter mod-6 kita dapat membangunnya dari counter mod-3 dan counter mod-2 yang dirangkai secara seri (3 x 2 = 6) seperti pada gambar 6.3 berikut:



Gambar 6.3 3 x 2 mod-6 counter



c. SYNCHRONOUS COUNTER

Perbedaan antara synchronous counter dengan ripple counter adalah semua flip-flop mempunyai pulsa clock yang sama. Prosedure design suatu synchronous counter sama seperti pada prosedure desain flip-flop (pertemuan V).

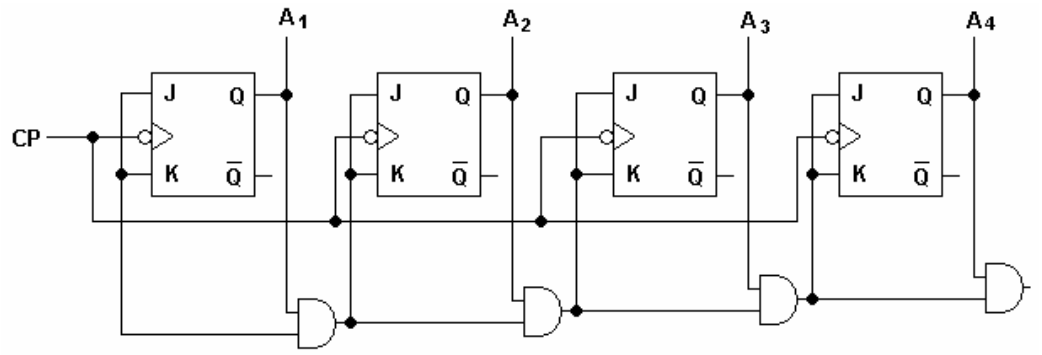
➢ **BINARY COUNTER**

Desain dari synchronous binary counter ini sederhana, dimana flip-flop pada urutan terendah dicomplement-kan dengan setiap pulsa. Flip-flop pada posisi lebih tinggi akan dicomplement-kan jika semua bit pada flip-flop yang lebih rendah bernilai 1.

Sebagai contoh:

Jika present state suatu counter 4 bit yaitu $A_4A_3A_2A_1$ = 0011, maka state berikutnya adalah 0100.

A_1 selalu dicomplement-kan. A_2 dicomplement-kan karena present state A_1 = 1. A_3 dicomplement-kan karena A_2A_1 = 11. Tetapi A_4 tidak dicomplement-kan karena present state $A_3A_2A_1$ = 011 yang mana tidak terdiri dari 1 semua.



Gambar 6.4 rangkaian dari synchronous binary counter 4 bit

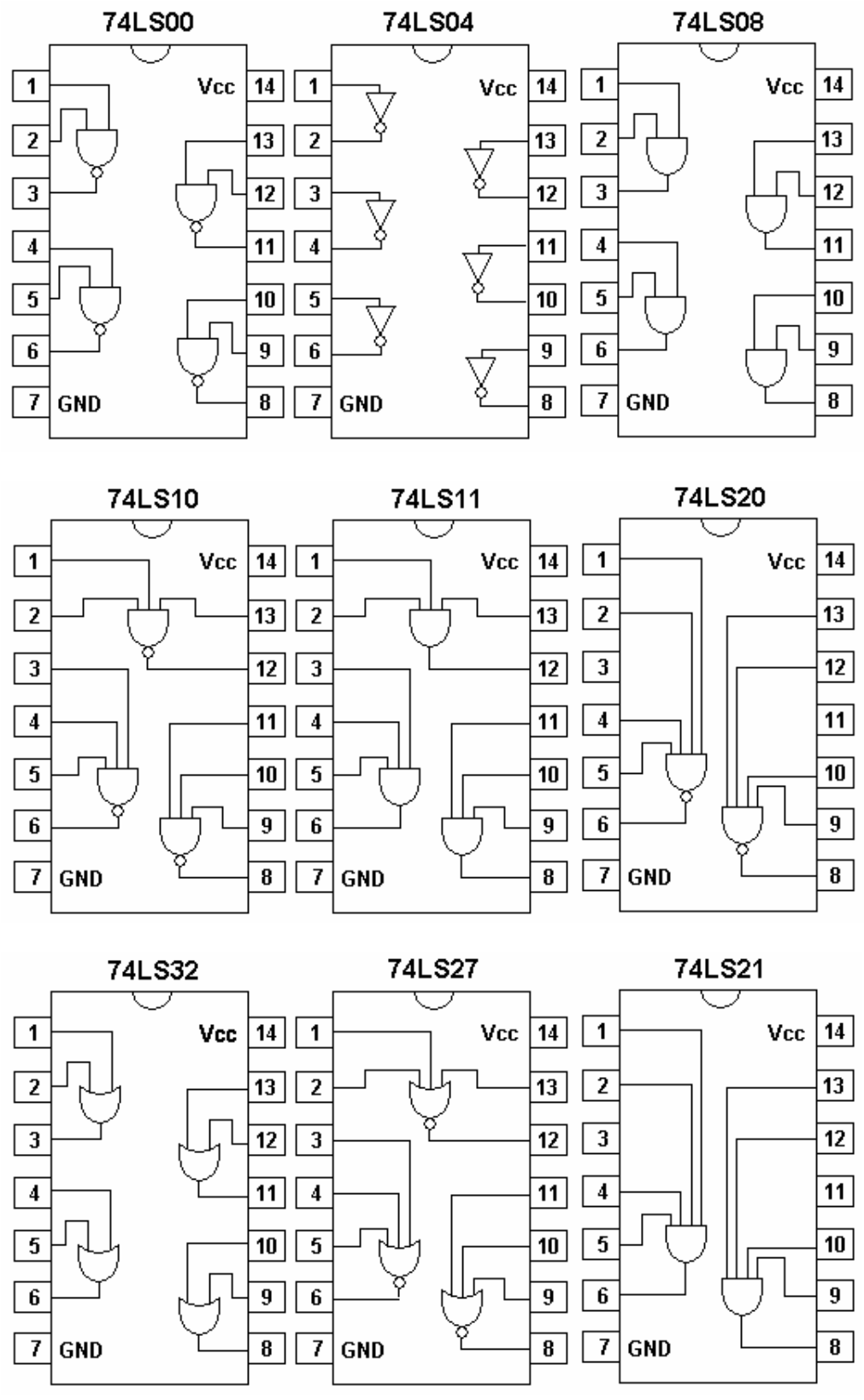
4. Alat dan Bahan
   a. Buku praktikum
   b. Papan Socket IC
   c. IC TTL dan kabel

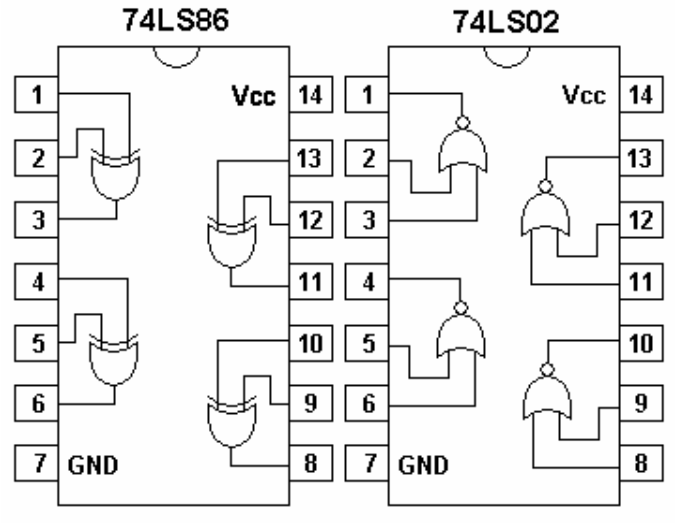
5. Prosedur Praktikum
   a. Peserta telah membaca dan mempelajari materi praktikum.
   b. Peserta merancang rangkaian pada lembar kerja kemudian diimplementasikan ke dalam papan socket.
   c. Jika terjadi alarm pada papan socket IC berarti telah terjadi kesalahan pemasangan rangkaian IC dan segeralah mematikan tombol power pada papan socket IC.

6. Percobaan
Implementasikan counter mod-6 pada gambar 6.3 diatas.

7. Latihan
   Buatlah synchronous binary counter 3 bit dengan menggunakan JK flip-flop

DATA BOOK TTL CIRCUIT




74LS86
Vcc
GND
1
2
3
4
5
6
7
14
13
12
11
10
9
8
74LS02
Vcc
GND
1
2
3
4
5
6
7
14
13
12
11
10
9
8