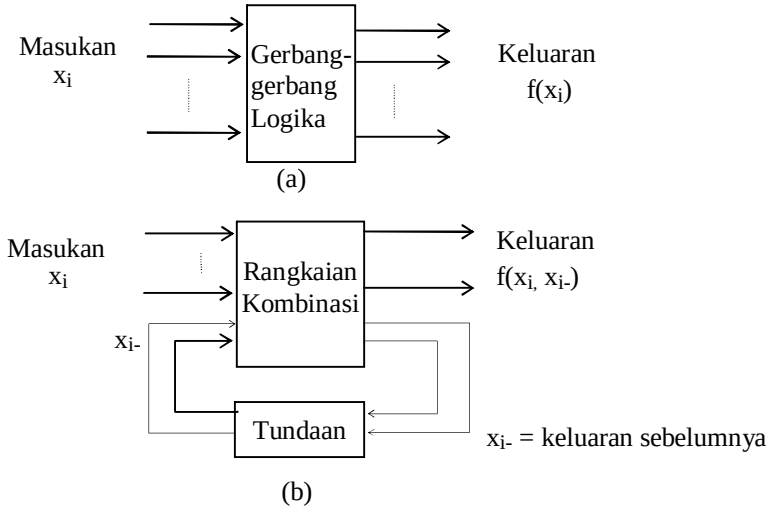
5
RANGKAIAN KOMBINASI

Semua rangkaian logika dapat digolongkan atas dua jenis, yaitu rangkaian kombinasi (combinational circuit) dan rangkaian berurut (sequential circuit). Perbedaan kedua jenis rangkaian ini terletak pada sifat keluarannya. Keluaran suatu rangkaian kombinasi setiap saat hanya ditentukan oleh masukan yang diberikan saat itu. Keluaran rangkaian berurut pada setiap saat, selain ditentukan oleh masukannya saat itu, juga ditentukan oleh keadaan keluaran saat sebelumnya, jadi juga oleh masukan sebelumnya. Jadi, rangkaian berurut tetap mengingat keluaran sebelumnya dan dikatakan bahwa rangkaian ini mempunyai ingatan (memory). Kemampuan mengingat pada rangkaian berurut ini diperoleh dengan memberikan tundaan waktu pada lintasan balik (umpan balik) dari keluaran ke masukan. Secara diagram blok, kedua jenis rangkaian logika ini dapat digambarkan seperti pada Gambar 5.1.



Gambar 5.1. Model Umum Rangkaian Logika
(a) Rangkaian Kombinasi
(b) Rangkaian Berurut

Perhatikan bahwa rangkaian berurut juga dibangun dari rangkaian kombinasi. Rangkaian ini menerima masukan melalui rangkaian kombinasi dan mengeluarkan keluarannya juga melalui rangkaian kombinasi. Jadi, rangkaian kombinasi merupakan dasar dari seluruh rangkaian logika. Sinyal yang diumpan-balik dalam rangkaian berurut umumnya merupakan keluaran elemen memori didalamnya, yang pada dasarnya juga dibangun dari rangkaian kombinasi. Tundaan dalam lintasan umpan balik itu boleh jadi hanya tundaan yang disumbangkan oleh rangkaian kombinasi pada lintasan tersebut, tetapi boleh jadi ditambahkan dengan sengaja. Umpan balik ini tidak ada pada rangkaian kombinasi. Rangkaian berurut ini akan diuraikan belakangan. Bab ini hanya akan menguraikan rangkaian kombinasi.

5.1 Perancangan Rangkaian Kombinasi

Rangkaian kombinasi mempunyai komponen-komponen masukan, rangkaian logika, dan keluaran, tanpa umpan balik. Persoalan yang dihadapi dalam perancangan (design) suatu rangkaian kombinasi adalah memperoleh fungsi Boole beserta diagram rangkaiannya dalam bentuk susunan gerbang-gerbang.

Seperti telah diterangkan sebelumnya, fungsi Boole merupakan hubungan aljabar antara masukan dan keluaran yang diinginkan. Langkah pertama dalam merancang setiap rangkaian logika adalah menentukan apa yang hendak direalisasikan oleh rangkaian itu yang biasanya dalam bentuk uraian kata-kata (verbal). Berdasarkan uraian kebutuhan ini ditetapkan jumlah masukan yang dibutuhkan serta jumlah keluaran yang akan dihasilkan. Masing-masing masukan dan keluaran diberi nama simbolis. Dengan membuat tabel kebenaran yang menyatakan hubungan masukan dan keluaran yang diinginkan, maka keluaran sebagai fungsi masukan dapat dirumuskan dan disederhanakan dengan cara-cara yang telah diuraikan dalam bab-bab sebelumnya. Berdasarkan persamaan yang diperoleh ini, yang merupakan fungsi Boole dari pada rangkaian yang dicari, dapat digambarkan diagram rangkaian logikanya Ada kalanya fungsi Boole yang sudah disederhanakan tersebut masih harus diubah untuk memenuhi kendala yang ada seperti jumlah gerbang dan jenisnya yang tersedia, jumlah masukan setiap gerbang, waktu perambatan melalui keseluruhan gerbang (tundaan waktu), interkoneksi antar bagian-bagian rangkaian, dan kemampuan setiap gerbang untuk mencatu (drive) gerbang berikutnya.

Harga rangkaian logika umumnya dihitung menurut cacah gerbang dan cacah masukan keseluruhannya. Ini berkaitan dengan cacah gerbang yang dikemas dalam setiap kemasan. Gerbang-gerbang logika yang tersedia di pasaran pada umumnya dibuat dengan teknologi rangkaian terpadu (Integrated Circuit, IC). Pemaduan (integrasi) gerbang-gerbang dasar seperti NOT, AND, OR, NAND, NOR, XOR pada umumnya dibuat dalam skala kecil (Small Scale Integration, SSI) yang mengandung 2 sampai 6 gerbang dalam setiap kemasan. Kemasan yang

paling banyak digunakan dalam rangkaian logika sederhana berbentuk DIP (Dual-In-line Package), yaitu kemasan dengan pen-pen hubungan ke luar disusun dalam dua baris sejajar. Kemasan gerbang-gerbang dasar umunya mempunyai 14-16 pen, termasuk pen untuk catu daya positif dan nol (Vcc dan Ground). Setiap gerbang dengan 2 masukan membutuhkan 3 pen (1 pen untuk keluaran) sedangkan gerbang 3 masukan dibutuhkan 4 pen. Karena itu, satu kemasan 14 pen dapat menampung hanya 4 gerbang 2 masukan atau 3 gerbang 3 masukan.

Dalam praktek kita sering terpaksa menggunakan gerbang-gerbang yang tersedia di pasaran yang kadang-kadang berbeda dengan kebutuhan rancangan kita. Gerbang yang paling banyak tersedia di pasaran adalah gerbang-gerbang dengan 2 atau 3 masukan. Umpamanya, dalam rancangan kita membutuhkan gerbang dengan 4 atau 5 masukan dan kita akan mengalami kesulitan memperoleh gerbang seperti itu. Karena itu kita harus mengubah rancangan sedemikian sehingga rancangan itu dapat direalisasikan dengan gerbang-gerbang dengan 2 atau 3 masukan.

Kemampuan pencatuan daya masing-masing gerbang juga membutuhkan perhatian. Setiap gerbang mampu mencatu hanya sejumlah tertentu gerbang lain di keluarannya (disebut sebagai fan-out). Ini berhubungan dengan kemampuan setiap gerbang dalam menyerap dan mencatu arus listrik. Dalam perancangan harus kita yakinkan bahwa tidak ada gerbang yang harus mencatu terlalu banyak gerbang lain di keluarannya. Ini sering membutuhkan modifikasi rangakaian realisasi yang berbeda dari rancangan semula. Mengenai karakteristik elektronik gerbang-gerbang logika dibahas dalam Lampiran A.

5.2 Rangkaian AND dan OR

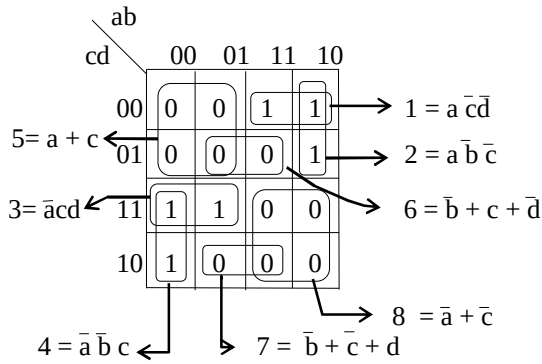
Masalah waktu perambatan terutama penting untuk sistem yang membutuhkan kecepatan tinggi. Kegagalan satu bagian rangkaian memenuhi kebutuhan waktu yang ditentukan dapat membawa kegagalan keseluruhan sistem dalam penyerempakan (synchronization), bukan hanya tak terpenuhinya kecepatan yang diinginkan. Untuk memenuhi tuntutan waktu ini, kadang-kadang kita perlu menambah atau mengurangi cacah tingkat (level) rangkaian logika yang kita rancang, yaitu menambah atau mengurangi cacah gerbang dalam deretan terpanjang yang menghubungkan masukan dengan keluaran. Semakin banyak tingkatnya, semakin panjang pula tundaan waktu yang dialami sinyal masukan untuk mencapai keluaran. Perlu diperhitungkan bahwa selain mengubah tundaan waktu, pada umumnya, pengubahan tingkat suatu rangkaian logika juga akan mengubah cacah gerbang dan cacah masukan gerbang yang diperlukan, dan kerena itu mengubah harga realisasi rangkaian.

Pada umumnya, penambahan tingkat (penambahan waktu tunda) suatu realisasi fungsi dalam bentuk jumlah-perkalian dapat dilakukan dengan menguraikan suku-suku fungsi tersebut, sedangkan penambahan tingkat realisasi fungsi dalam bentuk perkalian-jumlah dapat dilakukan dengan mengalikan beberapa suku

fungsi bersangkutan. Sebagai contoh, kita perhatikan fungsi:

$$f(a,b,c,d)= \Sigma\, m(2,3,7,8,9,12) \qquad (5.1)$$

Peta Karnaugh untuk fungsi ini ditunjukkan pada Gambar 5.2. Untuk menyatakan fungsi ini sebagai jumlah-perkalian, maka kita melakukan penggabungan semua kotak yang berisi 1 seperti yang ditunjukkan dengan penggabungan 1, 2, 3 dan 4 pada Gambar 5.2.



Gambar 5.2. Peta Karnaugh untuk $f = \Sigma\, m(2,3,7,8,9,12)$
$f = 1 + 2 + 3 + 4 = 5.6.7.8$

Fungsi minimum yang kita peroleh adalah:

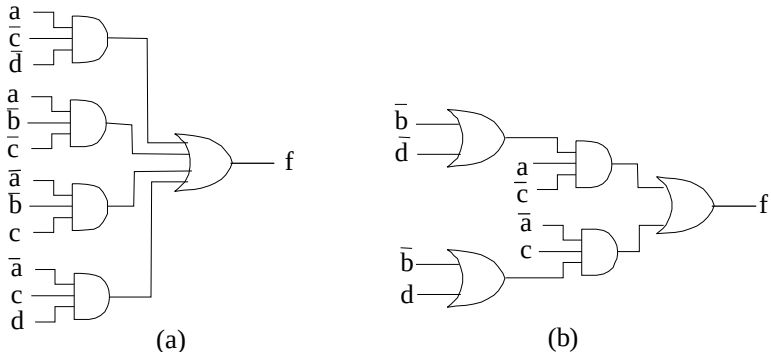
$$f = ac\bar{d} + ab\bar{c} + \bar{a}cd + \bar{a}\bar{b}c \qquad (5.2)$$

Realisasi langsung persamaan di atas akan menghasilkan rangkaian 2 tingkat AND-OR, yaitu rangkaian yang terdiri atas gerbang AND pada masukan diikuti oleh gerbang OR pada keluaran, seperti ditunjukkan pada Gambar 5.3(a). Rangkaian ini membutuhkan 5 gerbang dan 16 masukan.

Untuk menambah cacah tingkatnya menjadi 3, persamaan (5.2) di atas dapat diuraikan menjadi berbentuk :

$$f = ac\bar{d} + ab\bar{c} + \bar{a}cd + \bar{a}\bar{b}c = a\bar{c}(\bar{b}+\bar{d}) + \bar{a}c(\bar{b}+d) \qquad (5.3)$$

Seperti ditunjukkan pada Gambar 5.3(b), realisasi fungsi ini merupakan rangkaian OR-AND-OR 3-tingkat yang membutuhkan 12 masukan untuk 5 gerbang. Kelihatan di sini bahwa dengan penambahan cacah tingkat kita dapat mengurangi cacah masukan gerbang [bandingkan dengan 16 masukan untuk 5 gerbang pada Gambar 5.3(a)]. Perhatikan bahwa kedua rangkaian pada Gambar 5.3 memakai gerbang OR pada keluarannya. Ini merupakan ciri rangkaian untuk fungsi dalam bentuk jumlah-perkalian.



Gambar 5.3. Rangkaian contoh dengan minimisasi sukumin

Untuk mengekspansikan pers. (5.1) di atas dalam bentuk perkalian-jumlah, kita harus menggabungkan semua sukumax (kotak berisi 0) seperti yang ditunjukkan dengan penggabungan sukumin (5,6,7,8) pada Gambar 5.2, dengan fungsi minimum:

$$f = (a + c)(\bar{b} + c + \bar{d})(\bar{b} + \bar{c} + d)(\bar{a} + \bar{c}) \qquad (5.4)$$

Fungsi ini menjurus kepada realisasi seperti yang ditunjukkan pada Gambar 5.4(a), yang merupakan rangkaian OR-AND 2 tingkat dengan harga 5 gerbang dan 14 masukan.

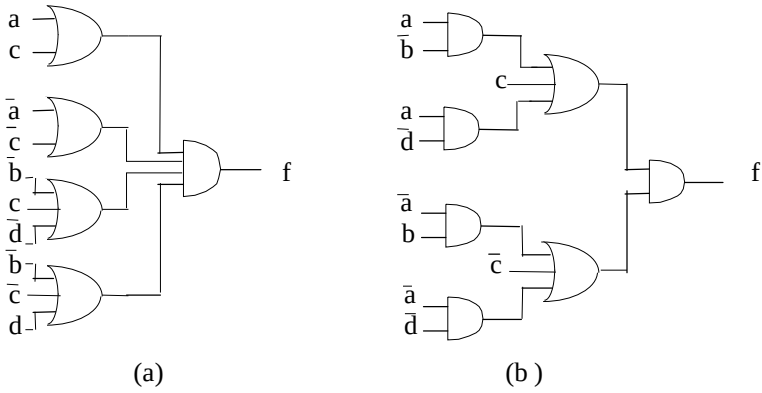
Dengan mengatur letak suku-sukunya dan dengan menggunakan rumus dari Bab 2, $(x + y)(x + z) = x + yz$, pers. (5.4) dapat ditulis dalam bentuk

$$f = (c + a)(c + \bar{b} + \bar{d})(\bar{c} + \bar{a})(\bar{c} + b + \bar{d})$$
$$= \{c + a(c + \bar{b} + \bar{d})\}\{(\bar{c} + \bar{a}(\bar{c} + b + \bar{d})\}$$
$$= \{c + a(\bar{b} + \bar{d})\}\{(\bar{c} + \bar{a}(b + \bar{d})\}$$
$$= (c + a\bar{b} + a\bar{d})(\bar{c} + \bar{a}b + \bar{a}\bar{d})$$

Realisasi persamaan ini merupakan rangkaian AND-OR-AND 3 tingkat, seperti yang ditunjukkan pada Gambar 5.4 (b) dengan kebutuhan gerbang sebanyak 7 dan 16 masukan. Jelas bahwa rangkaian ini lebih mahal dari rangkaian AND-OR 2-tingkat sebelumnya (7 : 5 gerbang dengan 16 : 16 masukan).

Kalau diperhatikan keempat rangkaian pada Gambar 5.3 dan Gambar 5.4, dapat dilihat bahwa rangkaian Gambar 5.4 (a) adalah rangkaian yang terbaik untuk rangkaian dua tingkat dan Gambar 5.3(b) adalah yang terbaik untuk tiga tingkat. Biasanya, hal ini baru dapat diketahui setelah menggambarkan rangkaian,

baik dengan keluaran AND maupun dengan keluaran OR. Untuk melihat cacah tingkat dan kebutuhan gerbang dan masukan tanpa harus menggambarkan rangkaian secara lengkap dapat dilakukan dengan menggambarkan diagram pohon persamaan fungsi yang dihadapi.

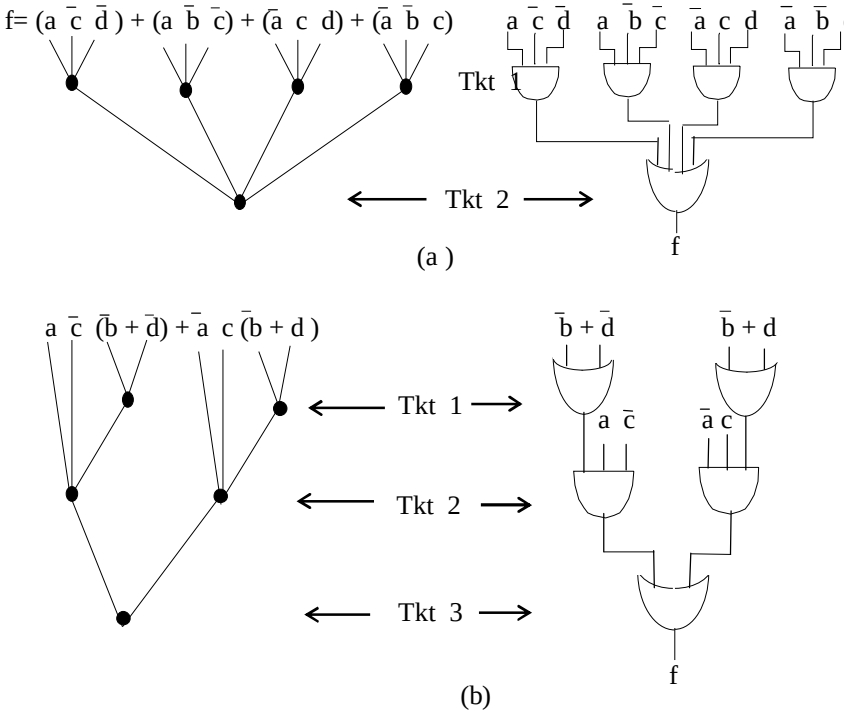


Gambar 5.4. Rangkaian contoh dengan penyederhanaan sukumax

Diagram pohon (tree diagram) suatu rangkaian logika merupakan gambaran rangkaian logika dengan keluaran sebagai akar (root) dan peubah masukan sebagai cabang/ranting/daunnya. Setiap gerbang digambarkan sebagai titik-cabang/simpul (node) dengan masukan sebagai ranting dan keluaran sebagai cabang. Pada setiap simpul dicantumkan cacah masukan gerbang yang bersangkutan. Gerbang keluaran, yaitu gerbang paling akhir pada rangkaian itu, hanya ditunjukkan sebagai simpul dan cabang tanpa lengan keluaran.

Untuk fungsi dalam pers. (5.3), sebagai contoh, diagram pohonnya dapat digambarkan seperti pada Gambar 5.5. Dari diagram ini dapat segera dilihat cacah tingkatnya dan kebutuhan gerbang dan masukan gerbang. Misalnya dari diagram pohon pada Gambar 5.5(b), segera terlihat bahwa rangkaiannya merupakan rangkaian 3 tingkat yang menggunakan 5 gerbang dengan cacah masukan gerbang adalah 2 + 2 + 3 + 3 + 2 = 12.

Suku yang sama dalam persamaan fungsi yang direalisasikan, yang menunjukkan adanya gerbang yang digunakan lebih dari satu kali, tetap digambarkan sebagai simpul terpisah dalam diagram pohon, tetapi hanya satu dari simpul yang sama itu yang diberi angka cacah masukan. Simpul tanpa angka masukan dalam diagram pohon tidak diikut-sertakan dalam perhitungan, cacah gerbang dan cacah masukan.



Gambar 5.5. Contoh diagram pohon 2 dan 3 tingkat.

5.3 Rangkaian NOR dan NAND

Seperti telah pernah disinggung di bagian depan, karena pemakaiannya yang lebih umum, gerbang-gerbang NOR dan NAND pada umumnya lebih mudah diperoleh di pasaran. Karena itu, desain sistem yang kecil-kecil yang memakai komponen rangkaian terpadu (Integrated Circuit, IC) skala kecil (SSI) dan menengah (MSI) sering diusahakan memakai gerbang-gerbang NOR dan NAND. Hal ini tentunya tidak perlu dilakukan dalam mendesain sistem dalam rangkaian terpadu skala besar (LSI, Large Scale Integration) atau yang sangat besar (VLSI, Very Large Scale Integration) yang semakin banyak digunakan.

Dengan memakai hukum deMorgan yang diberikan dalam bab 2, rangkaian yang tersusun atas OR dan AND dapat diubah menjadi susunan gerbang-gerbang NOR dan NAND. Prinsip yang penting dalam hal ini adalah kenyataan bahwa jika

suatu fungsi atau peubah dikomplemenkan dua kali, maka hasilnya kembali ke bentuk sebenarnya, yaitu :

$$\overline{\overline{(f)}} = f$$

Perhatikan fungsi minimum dalam bentuk jumlah perkalian:

$$f = \bar{A}B + \bar{B}\bar{D} + C\bar{D} + A\bar{B}\bar{C} \tag{5.5}$$

Dengan menggunakan hukum de Morgan, fungsi ini dapat diubah kebentuk-bentuk lain sebagai berikut:

$$f = \overline{\overline{\bar{A}B + \bar{B}\bar{D} + C\bar{D} + A\bar{B}\bar{C}}}$$

$$f = \overline{(\overline{\bar{A}B})\,(\overline{\bar{B}\bar{D}})\,(\overline{C\bar{D}})\,(\overline{A\bar{B}\bar{C}})} \tag{5.6}$$

$$= \overline{(A+\bar{B})\,(B+D)\,(\bar{C}+D)\,(\bar{A}+B+C)} \tag{5.7}$$

$$= (\overline{A+\bar{B}}) + (\overline{B+D}) + (\overline{\bar{C}+D}) + (\overline{\bar{A}+B+C}) \tag{5.8}$$

Pers. (5.5) mewakili bentuk dasar rangkaian AND-OR, pers.(5.6) mewakili bentuk dasar rangkaian NAND-NAND, pers.(5.7) mewakili rangkaian dasar OR-NAND, dan pers. (5.8) mewakili rangkaian dasar NOR-OR yang juga dapat diubah menjadi rangkaian NOR-NOR-NOT. Rangkaian-rangkaian ini digambarkan pada Gambar 5.6.

Kalau kita menginginkan rangkaian yang hanya mengandung gerbang NOR, lebih baik kita mulai dengan bentuk perkalian-jumlah sebagai ganti bentuk jumlah-perkalian. Fungsi di atas, sebagai contoh, sebenarnya merupakan fungsi jumlah-perkalian:

$$f(A,B,C,D) = \Sigma\, m(0,2,4,5,6,7,8,9,10,14)$$

atau fungsi perkalian-jumlah:

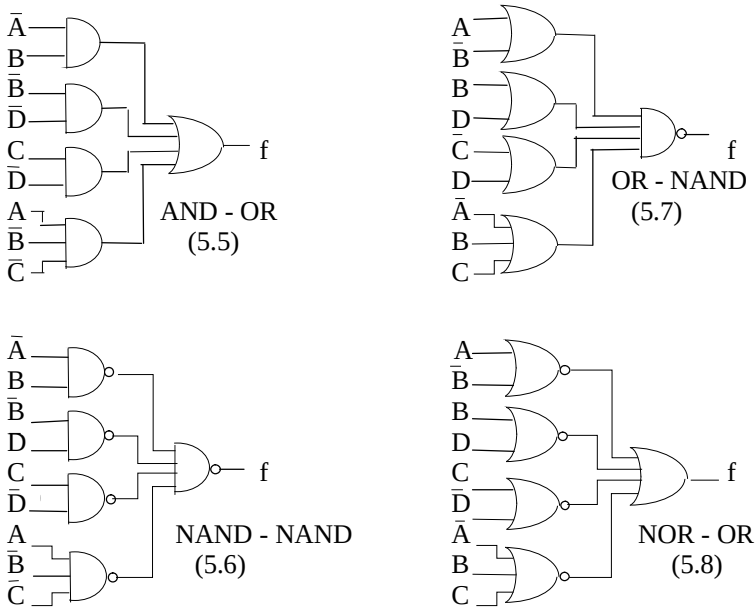
$$f(A,B,C,D) = \prod M(1,3,11,12,13,15)$$

yang dapat digambarkan dalam peta Karnaugh seperti ditunjukkan dalam dalam Gambar 5.7.

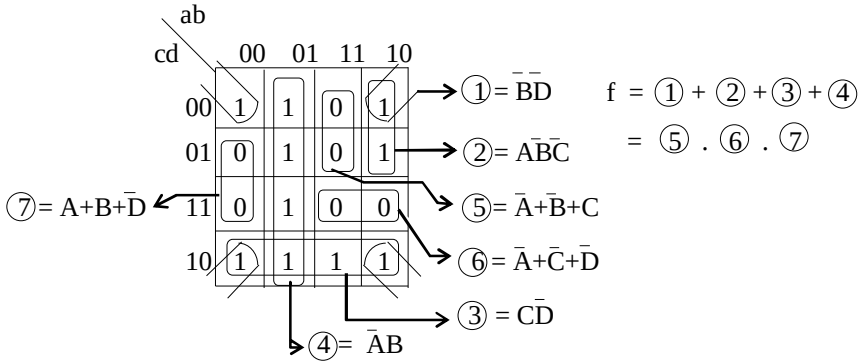
Dari gambar ini dapat dilihat bahwa fungsi di atas dapat dinyatakan sebagai:

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

dengan harga 4 gerbang dengan 12 masukan yang lebih murah dibanding pers. (5.8) yang berharga 5 gerbang dengan 13 masukan.



Gambar 5.6. Bentuk bentuk dasar Rangkaian dua tingkat.



Gambar 5.7. Peta Karnaugh fungsi pada pers. (5.5) .

5.4 Rangkaian Penjumlah

Operasi yang paling mendasar dalam suatu sistem digital adalah penjumlahan; hampir semua operasi aljabar dapat dilaksanakan dengan operasi penjumlahan. Rangkaian penjumlah yang paling sederhana dan mendasar adalah penjumlah yang menjumlahkan dua angka biner. Untuk mengetahui bentuk rangkaian yang dibutuhkan kita lihat hukum penjumlahan dua angka biner sebagai berikut :

$0 + 0 = 0$ $\quad$ $1 + 0 = 1$

$0 + 1 = 1$ $\quad$ $1 + 1 = 0$ carry 1 $= 10$

(simpan)

Kalau kita perhatikan operasi perjumlahan diatas, kita akan lihat bahwa perjumlahan sama dengan operasi OR dengan pengecualian untuk keadaan kedua angka yang dijumlahkan berharga 1. Juga dapat dilihat bahwa hasil perjumlahan adalah 1 bila kedua angka tidak sama sedangkan bila kedua angka yang dijumlahkan sama, maka hasilnya adalah 0. Tetapi untuk kedua operand = 1, maka akan dihasilkan simpanan (carry). Simpanan ini harus diperhitungkan bila penjumlahan dilakukan untuk bit yang lebih mahal (lebih tinggi nilainya) dan untuk itu, tentunya, harus dideteksi.

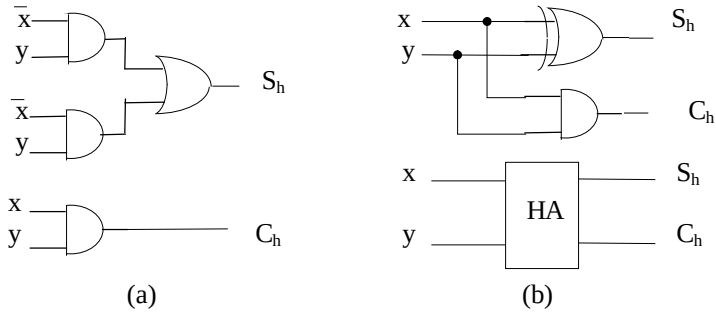
Dengan menyebut kedua angka yang dijumlahkan sebagai x dan y, hasil perjumlahan sebagai S (sum), dan simpanan sebagai C (carry), maka tabel kebenaran untuk rangkaian penjumlahan diatas dapat dibuat sebagai berikut:

x	y	S_h	C_h
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Dari tabel kebenaran ini dapat diperoleh persamaan:

$$S_h = \bar{x}\, y + x\bar{y} = x \oplus y\,; \quad C_h = xy \qquad (5.9)$$

Pemberian subskrip **h** kepada S dan C pada persamaan ini ditujukan untuk menunjukkan sifatnya sebagai penjumlah paruh (half adder). Penamaan sebagai penjumlah paruh ini muncul dari ketidak-mampuan menerima simpanan (carry) dari hasil penjumlah untuk bit dengan bobot dibawahnya. Rangkaian logika untuk pers. (5.9) digambarkan pada Gambar 5.8.



Gambar 5.8. Rangkaian dan simbol penjumlah paruh

Sekarang perhatikan penjumlahan dua bilangan biner yang masing-masing terdiri dari 3 bit: 101 + 111. Dengan memakai hukum perjumlahan yang disebutkan sebelumnya, maka diperoleh hasil perjumlahan sebagai berikut:

$$\begin{array}{l} 1\ 1\ 1 \quad \longleftarrow \text{ simpanan (carry)} \\ \ \ 1\ 0\ 1 \\ \ \ 1\ 1\ 1 \\ \hline 1\ 1\ 0\ 0 \end{array}$$

Perhatikan bahwa untuk bit kedua, ketiga, dan seterusnya, dari kanan, diperlukan juga masukan untuk penjumlahan simpanan dari perjumlahan bit di kanannya, dan ini tidak dimiliki oleh penjumlah yang telah diuraikan sebelumnya. Penjumlah yang memiliki 3 masukan disebut "Penjumlah penuh" (Full Adder).

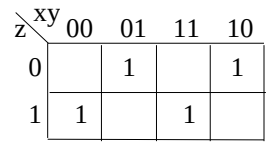
Karena, seperti yang akan ditunjukkan kemudian, satu penjumlah penuh dapat dibentuk dari dua penjumlah dengan dua masukan, maka penjumlah dua masukan disebut "Penjumlah Paruh" (Half Adder).

Tabel kebenaran dan peta Karnaugh penjumlah penuh dapat dibuat seperti ditunjukkan pada Gambar 5.9. Dari tabel kebenaran dan peta Karnaugh pada Gambar 5.9 dapat diperoleh persamaan Sum dan Carry sebagai berikut :

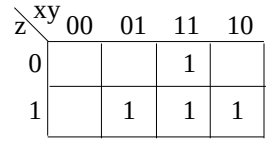
$$\begin{aligned} S_f &= \bar{x}\,\bar{y}\,z + x\,y\,z + \bar{x}\,y\,\bar{z} + x\,\bar{y}\,\bar{z} \\ &= (\bar{x}\,\bar{y} + x\,y)\,z + (\bar{x}\,y + x\,\bar{y})\,z \\ &= \overline{(x \oplus y)}\,z + (x + y)\,\bar{z} \\ &= (x \oplus y) \oplus z \end{aligned} \qquad (5\text{-}10)$$

$$C_f = xy + xz + yz = xy + (\bar{x}\,y + x\bar{y})z = xy + (x \oplus y)\,z$$

x	y	z	S	C
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



S_f



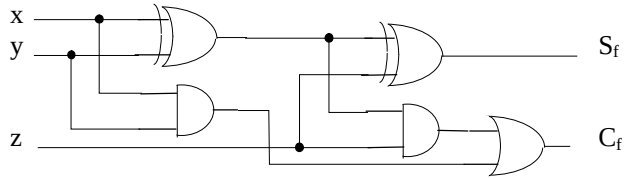
C_f

Gambar 5.9. Tabel Kebenaran dan Peta Penjumlah penuh

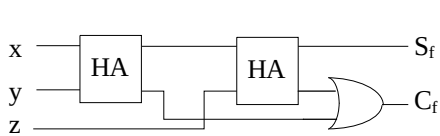
Perhatikan bahwa z pada persamaan-persamaan di atas dapat dipakai untuk masukan bagi simpanan pada penjumlahan sebelumnya. Kalau kita bandingkan pers.(5.10) dengan pers.(5.9), maka dapat dilihat bahwa:

$$S_f = S_h \oplus z \qquad (5.11)$$
$$C_f = z\, S_h + C_h$$

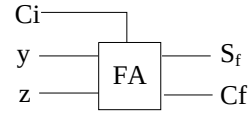
Diagram rangkaian persamaan ini ditunjukkan pada Gambar 5.10 yang juga menunjukkan bahwa satu penjumlah penuh dapat dibuat dari 2 penjumlah paruh.



(a)



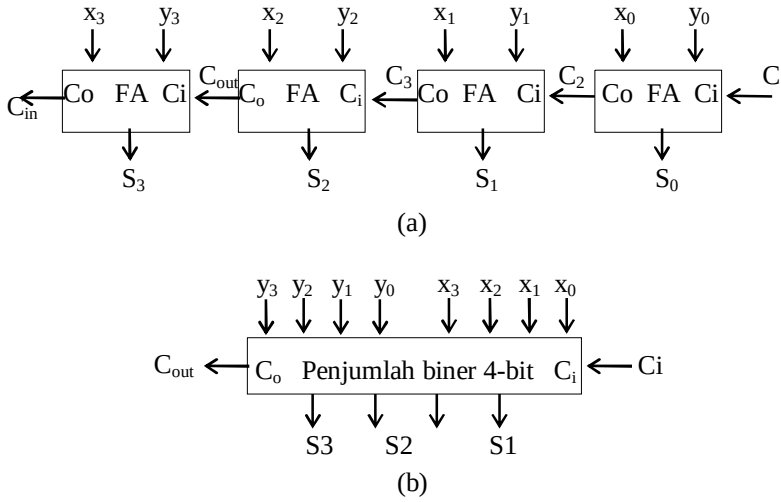
(b)



(c)

Gambar 5.10. Rangkaian penjumlah penuh.

Penjumlah untuk beberapa bit dapat dibentuk dengan menghubungkan beberapa buah penjumlah penuh, Carry keluaran (carry-out) penjumlah bit rendah diumpankan ke Carry masukan (carry-in) penjumlah bit lebih tinggi. Dalam Gambar 5.11(a) ditunjukkan hubungan 4 penjumlah penuh yang membentuk penjumlah biner 4-bit yang secara diagram dapat juga digambarkan seperti pada Gambar 5.11(b). Jenis penjumlah lain seperti penjumlah BCD atau penjumlah Desimal dapat disusun dengan cara yang sama. Di pasaran tersedia rangkaian terpadu penjumlah penuh untuk cacah bit tertentu, misalnya penjumlah 4-bit SN7483.



Gambar 5.11. Hubungan penjumlah biner 4-bit.

5.5 Rangkaian pengurang

Seperti telah diterangkan di bab sebelumnya, operasi pengurangan dapat dilaksanakan dengan penjumlahan dengan komplemen bilangan pengurang. Ini tidak berarti bahwa rangkaian untuk pengurangan tidak dapat dibentuk. Yang perlu diperhatikan dalam operasi pengurangan adalah adanya pinjaman (borrow) dari bit yang lebih mahal bila bit pengurang lebih besar dari pada bit yang dikurangi.

Sebagaimana pada penjumlah, dalam hal pengurangan juga ada pengurang paruh dan pengurang penuh. Kalau kedua operand disebut x dan y, selisih disebut D (difference) dan pinjaman disebut B (borrow), maka tabel kebenaran pengurang paruh dapat dibuat seperti pada Gambar 5.12.

x	y	D	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

(a)

x	y	z	D	B
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

(b)

Gambar 5.12. Tabel-tabel kebenaran rangkaian pengurang
(a) pengurang paruh: D = x - y
(b) pengurang penuh: D = x - y - z

Dari Gambar 5.12 dapat diperoleh persamaan:

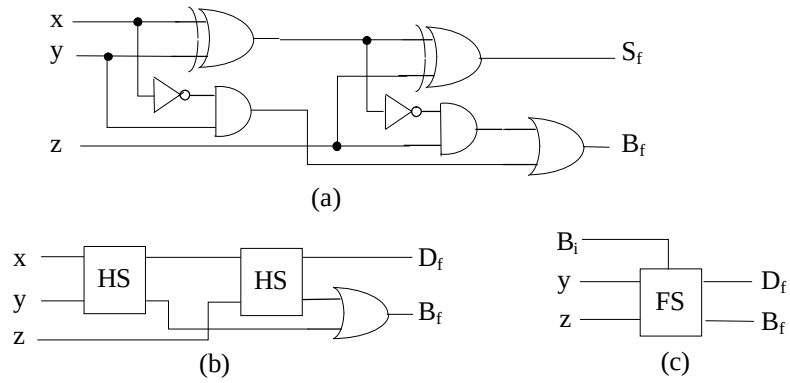
$$D_h = \bar{x}y + x\bar{y} = x \oplus y$$
$$B_h = \bar{x}y \qquad (5.12)$$

Pada Gambar 5.12(b) ditunjukkan tabel kebenaran pengurang penuh dan dari tabel tersebut diperoleh persamaan:

$$D_f = \bar{x}\bar{y}z + \bar{x}y\bar{z} + x\bar{y}\bar{z} + xyz$$
$$= (\bar{x}\bar{y} + xy)z + (\bar{x}y + x\bar{y})\bar{z} = (x \oplus y) \oplus z \qquad (5.13)$$
$$B_f = \bar{x}\bar{y}z + \bar{x}y\bar{z} + \bar{x}yz = \bar{x}y + (\bar{x}\bar{y} + xy)z = \bar{x}y + z\,\overline{(x \oplus y)}$$

Perhatikan bahwa jumlah untuk penjumlah paruh sama dengan selisih untuk pengurang paruh dan jumlah untuk penjumlah penuh mempunyai persamaan yang sama dengan selisih untuk pengurang penuh. Selanjutnya, pinjaman B untuk pengurang penuh akan sama dengan simpanan pada penjumlah penuh bila x digantikan dengan $\bar{x}$. Jadi dapat dilihat bahwa pengurang penuh dapat dibuat dari penjumlah penuh dengan memberikan inverter pada masukan x bagi logika pinjamannya. Rangkaian pengurang penuh ditunjukkan pada Gambar 5.13

.



Gambar 5.13. Rangkaian pengurang penuh.

5.6 Pengubah Kode

Informasi yang diolah dalam sistem digital dapat dikodekan secara berbeda dari satu sistem ke sistem lain. Bila dua sistem digital yang menganut sistem pengkodean yang berbeda hendak kita gabung, maka kita membutuhkan pengubahan kode dari kode satu mesin kekode mesin yang lain. Hanya dengan pengubah kode ini kedua mesin menjadi "compatible" (dapat disambung/hubungkan satu sama lain). Jelaslah kalau masukan rangkaian pengubah kode itu merupakan kode-kode biner dalam satu sistem kode, misalnya A, yang dipakai mesin x, maka keluarannya haruslah kode biner dalam sistem kode yang lain, misalnya B, yang dikenal oleh mesin y. Sebagai contoh, perhatikanlah sistem kode BCD dan Excess-3 yang telah diterangkan dalam Bab 1, yang kembali ditunjukkan pada Gambar 5.14(a). Karena baik kode BCD maupun kode XS-3 terdiri atas 4 angka biner (bit), maka rangkaian pengubah kode yang dibutuhkan akan mempunyai 4 masukan dan 4 keluaran.

Misalkanlah peubah masukan kita sebut a, b, c, d, dan peubah keluaran kita sebut P, Q, R, S. Maka tabel kebenaran rangkaian yang dicari adalah hubungan yang ditunjukkan dalam tabel diatas. Perlu dicatat kembali bahwa untuk kode yang lebih besar dari 9 desimal, kombinasi masukannya merupakan abaikan (don't care). Dalam Gambar 5.14(b) ditunjukkan peta Karnaugh untuk keempat fungsi keluaran. Dari peta tersebut dapat diperoleh persamaan-persamaan keluaran sebagai berikut :

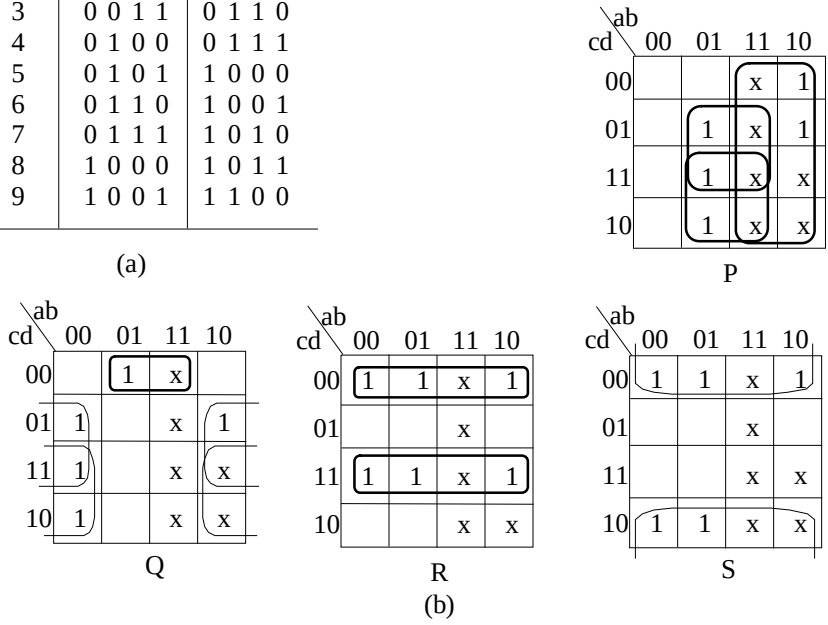
$P = a + bc + bd$ $\qquad$ $R = \bar{c}\bar{d} + cd$

$Q = \bar{b}c + \bar{b}d + b\bar{c}\bar{d}$ $\qquad$ $S = \bar{d}$

yang rangkaian realisasinya dapat digambarkan dengan mudah.

Angka desimal	Kode BCD a b c d	Kode XS-3 P Q R S
0	0 0 0 0	0 0 1 1
1	0 0 0 1	0 1 0 0
2	0 0 1 0	0 1 0 1
3	0 0 1 1	0 1 1 0
4	0 1 0 0	0 1 1 1
5	0 1 0 1	1 0 0 0
6	0 1 1 0	1 0 0 1
7	0 1 1 1	1 0 1 0
8	1 0 0 0	1 0 1 1
9	1 0 0 1	1 1 0 0

(a)



Gambar 5.14. Pengubah kode BCD ke XS-3

Perlu diperhatikan bahwa walaupun ada suku yang dapat dipakai secara bersama, seperti suku cd, ternyata pemakaian suku bersama ini tidak mempersederhana fungsi secara keseluruhan karena untuk S suku ini bergabung membentuk $\bar{d}$.

Contoh lain, misalnya kita hendak menampilkan kode BCD ke penampil lampu LED 7-segmen. Dalam Gambar 5.15 digambarkan penamaan segmen-segmen LED 7-segmen dan tabel kebenaran pengubahan kode yang dibutuhkan. Berdasarkan tabel kebenaran tersebut dapat disusun peta Karnaugh seperti yang juga ditunjukkan dalam gambar tersebut. Dengan meminimalkan masing-masing fungsi dapat diperoleh fungsi realisasinya sebagai berikut:

$a(A,B,C,D)= \Sigma\, m(0,2,3,5,7,8,9) = \bar{A}\bar{B}\bar{D} + \bar{A}CD + \bar{A}B\bar{D} + A\bar{B}\bar{C}$

$b(A,B,C,D)= \Sigma\, m(0,1,2,3,4,7,8,9) = \bar{A}\bar{B} + \bar{B}\bar{C} + \bar{A}\bar{C}\bar{D} + \bar{A}CD$

$c(A,B,C,D)= \Sigma\, m(0,1,3,4,5,6,7,8,9) = \bar{A}D + \bar{A}B + \bar{B}\bar{C}$

$d(A,B,C,D)= \Sigma\, m(0,2,3,5,6,8) = \bar{A}\bar{B}C + \bar{A}C\bar{D} + \bar{B}\bar{C}\bar{D} + \bar{A}B\bar{C}D$

$e(A,B,C,D)= \Sigma\, m(0,2,6,8) = \bar{A}C\bar{D} + \bar{B}\bar{C}\bar{D}$

$f(A,B,C,D)= \Sigma\, m(0,4,5,6,8,9) = \bar{B}\bar{C}\bar{D} + \bar{A}B\bar{C} + \bar{A}B\bar{D} + A\bar{B}\bar{C}$

$g(A,B,C,D)= \Sigma\, m(2,3,4,5,6,8,9) = \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}B\bar{D} + A\bar{B}\bar{C}$

Namun, karena konversi kode ini merupakan fungsi keluaran ganda, maka akan diperoleh realisasi yang lebih murah bila diminimalkan secara bersama, yaitu dengan mendahulukan penggabungan suku-suku bersama daripada penggabungan suku-suku masing-masing fungsi secara terpisah. Dalam Gambar 5.15 ditunjukkan penggabungan yang harus digunakan pada satu fungsi diberi nomor berlingkaran, yaitu: 1 untuk a; 2 untuk a , f dan g; 3 untuk b; 4 untuk b; 5 untuk d dan e; 6 untuk d dan e; 7 untuk f, 8 untuk f dan g; dan 9 untuk g; . Dalam fungsi-fungsi lain penggabungan ini dapat digantikan dengan penggabungan lain. Penggabungan bersama ini menghasilkan fungsi minimum sebagai berikut:

$a(A,B,C,D) = ① + ② + ⑤ + ⑨$
$= \bar{A}BD + A\bar{B}\bar{C} + \bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}C$

$b(A,B,C,D) = ② + ③ + ④ + \bar{A}\bar{B}$
$= A\bar{B}\bar{C} + \bar{A}\bar{C}\bar{D} + \bar{A}CD + \bar{A}\bar{B}$

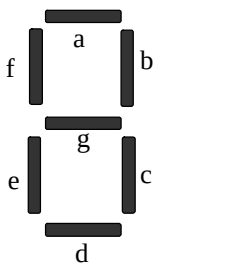
$c(A,B,C,D) = ② + ③ + ⑦ + \bar{A}D$
$= A\bar{B}\bar{C} + \bar{A}\bar{C}\bar{D} + \bar{A}B\bar{D} + \bar{A}D$

$d(A,B,C,D) = ⑤ + ⑥ + ⑨ + \bar{A}B\bar{C}D$
$= \bar{B}\bar{C}\bar{D} + \bar{A}C\bar{D} + \bar{A}\bar{B}C + \bar{A}B\bar{C}D$

$e(A,B,C,D) = ⑤ + ⑥$
$= \bar{B}\bar{C}\bar{D} + \bar{A}C\bar{D}$

$f(A,B,C,D) = ② + ⑤ + ⑦ + ⑧$
$= \bar{B}\bar{C}\bar{D} + \bar{B}\bar{C}\bar{D} + \bar{A}B\bar{D} + \bar{A}B\bar{C}$

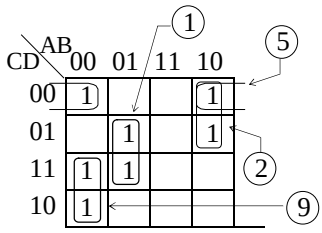
$g(A,B,C,D) = ② + ⑥ + ⑧ + ⑨$
$= A\bar{B}\bar{C} + \bar{A}C\bar{D} + \bar{A}B\bar{C} + \bar{A}\bar{B}C$



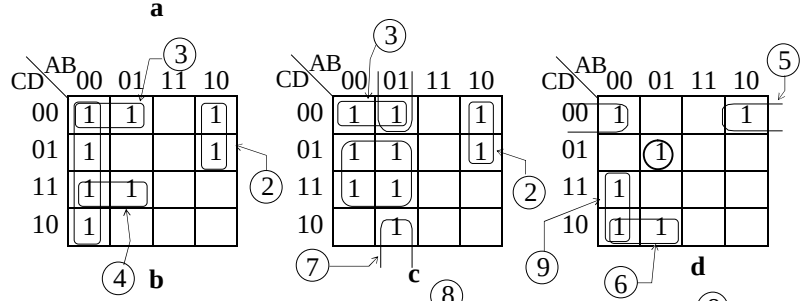
(a)

Desimal	BCD	LED 7-segmen
	A B C D	a b c d e f g
0	0 0 0 0	1 1 1 1 1 1 0
1	0 0 0 1	0 1 1 0 0 0 0
2	0 0 1 0	1 1 0 1 1 0 1
3	0 0 1 1	1 1 1 1 0 0 1
4	0 1 0 0	0 1 1 0 0 1 1
5	0 1 0 1	1 0 1 1 0 1 1
6	0 1 1 0	0 0 1 1 1 1 1
7	0 1 1 1	1 1 1 0 0 0 0
8	1 0 0 0	1 1 1 1 1 1 1
9	1 0 0 1	1 1 1 0 0 1 1
10,11	1 0 1 x	0 0 0 0 0 0 0
12,13, 14,15	1 1 x x	0 0 0 0 0 0 0

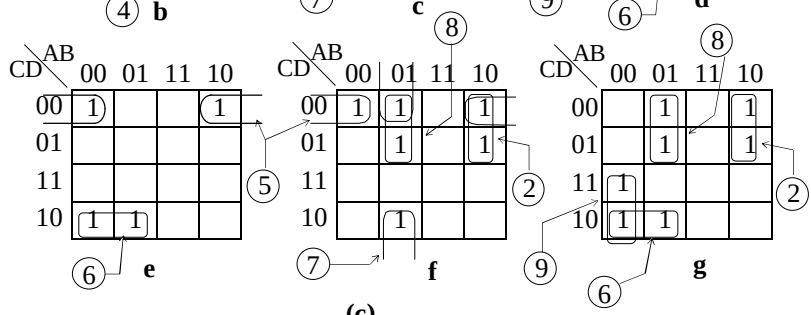
(b)



a



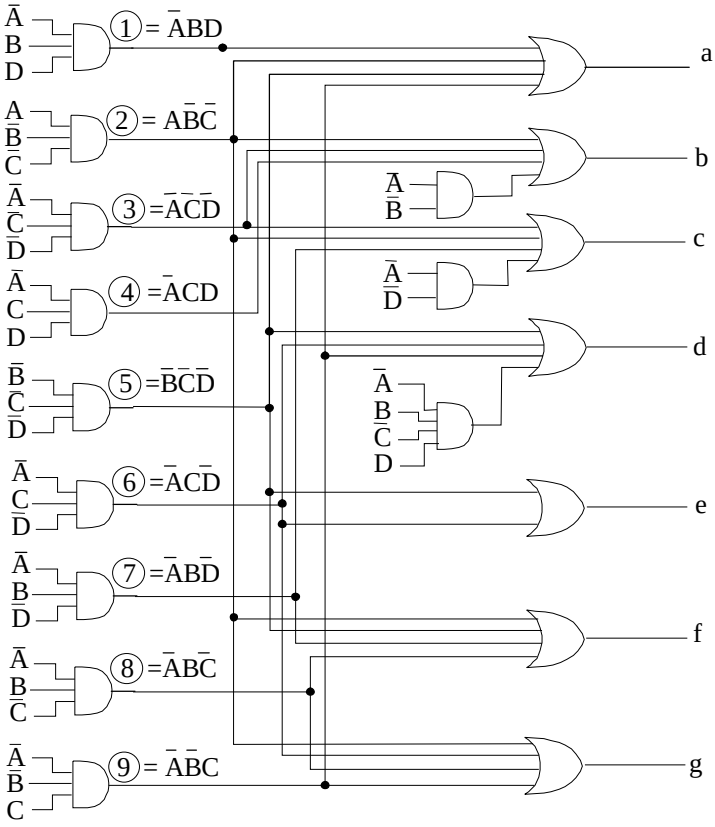
b **c** **d**



e **f** **g**

(c)

Gambar 5.15. Konversi kode BCD ke LED 7-segmen



Gambar 5.16. Rangkaian Pengubah kode BCD ke LED 7-segmen

Perhatikan penghematan yang dicapai dengan menggunakan bebrapa gerbang secara bersama untuk realisasi beberapa segmen.

Di pasaran tersedia pengubah kode BCD-ke-LED 7-segmen dalam bentuk rangkaian terpadu (IC) skala menengah (MSI, Medium Scale Integration) dengan nama “BCD-to-Seven-Segmen Driver/Decoder dengan nomor tipe 46, 47, 48 dan 49, misalnya 7447, 74L47, 74LS47 dan sebagainya. Pada Gambar 5.17 ditunjukkan logika dan rangkaian dari IC tipe 46 dan 47.

Desimal/	Masukan			BI/	Keluaran ke LED 7-seg							
Fungsi	LT	RBI	D C B A	BR0	a	b	c	d	e	f	g	
0	1	1	0 0 0 0	1	1	1	1	1	1	1	0	0
1	x	1	0 0 0 1	1	0	1	1	0	0	0	0	1
2	1	x	0 0 1 0	1	1	1	0	1	1	0	1	2
3	1	x	0 0 1 1	1	1	1	1	1	0	0	1	3
4	1	x	0 1 0 0	1	0	1	1	0	0	1	1	4
5	1	x	0 1 0 1	1	1	0	1	1	0	1	1	5
6	1	x	0 1 1 0	1	0	0	1	1	1	1	1	6
7	1	x	0 1 1 1	1	1	1	1	0	0	0	0	7
8	1	x	1 0 0 0	1	1	1	1	1	1	1	1	8
9	1	x	1 0 0 1	1	1	1	1	0	0	1	1	9
10	1	x	1 0 1 0	1	0	0	0	1	1	0	1	10
11	1	x	1 0 1 1	1	0	0	1	1	0	0	1	11
12	1	x	1 1 0 0	1	0	1	0	0	0	1	1	12
13	1	x	1 1 0 1	1	1	0	0	1	0	1	1	13
14	1	x	1 1 1 0	1	0	0	0	1	1	1	1	14
15	1	x	1 1 1 1	1	0	0	0	0	0	0	0	15
BI	x	x	x x x x	0	0	0	0	0	0	0	0	
RBI	1	0	0 0 0 0	0	0	0	0	0	0	0	0	
B0	0	x	x x x x	1	1	1	1	1	1	1	1	

Catatan : LT= Lamp Test 1
RBI=Ripple Blanking Input
BI/RB0= Blanking Input/ Ripple Blanking Output

Gambar 5.17. Rangkaian terpadu pengubah BCD-ke-LED 7-segmen tipe 47.

Perbedaan Gambar 5.17 dengan Gambar 5.15 (b) hanya pada kode untuk 10 s/d 15 dan adanya kendali masukan LT, RBI dan BI/RBO pada IC tipe 47. Dalam Gambar 5.15(b), semua kode untuk 10-15 akan menghasilkan keluaran 0 sedangkan pada Gambar 5.17 dibentuk kode gambar tambahan. Pada Gambar 5.17, BI (Blanking Input) harus dibuat berlogika 1 agar ada keluaran yang hidup (logika 1); bila BI= 0, semua keluaran akan berlogika 0. Sinyal RBI (Ripple Blanking Input) mengendalikan penampilan angka nol, yaitu RBI= 0 untuk menampilkan nol. Sinyal BI/RBO (Blanking Input/Ripple Blanking Output)= 1 dan LT (Lamp Test)= 0 membuat semua sinyal keluaran berlogika 1. Dalam operasi normal, LT selalu dibuat berlogika 1. Perhatikan bahwa sinyal masukan yang dibiarkan terbuka akan berlogika 1.

5.7 Multiplexer

Multiplexer dikenal juga dengan nama Data Selector (Pemilih Data). Fungsi peralatan ini adalah untuk melalukan salah satu dan hanya satu dari sekumpulan masukan-datanya ke keluarannya. Pemilihan masukan mana yang dilalukan ditentukan oleh kombinasi sinyal yang diberikan pada masukan kendalinya (control input). Untuk memilih data yang ke i, artinya membuat keluaran sama dengan data masukan ke i, yaitu $Z = I_i$, sukumin yang dibentuk oleh peubah kendali, sebut saja m_i, dibuat berlogika 1. Jadi, untuk memilih salah satu dari 2^n masukan, dibutuhkan n bit kendali. Dengan demikian keadaan/logika keluaran multiplexer, disingkat dengan MUX, dapat dituliskan sebagai:

$$Z = \sum_{i=0}^{2^n-1} m_i I_i$$

Sebagai contoh, untuk MUX dengan 4 masukan data, yang disebut juga sebagai MUX 4 x 1, ada n = 2 sinyal kendali. Bila kita sebut sinyal kendali tersebut sebagai A dan B, dan masukan datanya disebut I_0, I_1, I_2 dan I_3, maka persamaan keluarannya dapat ditulis :

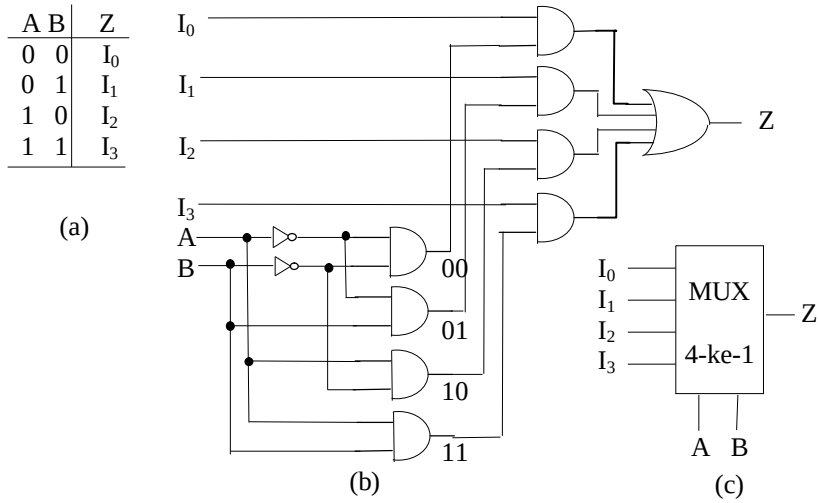
$$Z = m_0 I_0 + m_1 I_1 + m_2 I_2 + m_3 I_3$$
$$= \bar{A}\,\bar{B}\,I_0 + \bar{A}\,B\,I_1 + A\,\bar{B}\,I_2 + A\,B\,I_3$$

Fungsi logika MUX 4 x 1 ini dapat ditunjukkan dalam bentuk tabel kebenaran seperti yang ditunjukkan pada Gambar 5.18 (a) dan rangkaiannya dapat disusun seperti pada Gambar 5.18(b). Simbol Mux ditunjukkan pada Gambar 5.18 (c). Dalam bentuk rangkaian terpadu (IC), MUX yang paling umum dipergunakan adalah kemasan-kemasan yang terdiri atas 4, 2, atau 1 MUX dalam satu kemasan, yaitu MUX 2x1 (quadraple 2-to-1) misalnya SN74LS258, MUX 4x1 (dual 4-to-1) misalnya SN74LS253, MUX 8x1 (single 8-to-1) misalnya SN74LS251, dan MUX 16x1 (single 16-to-1). Penggunaan serpih-serpih MUX ini memegang peranan yang sangat penting dalam sistem digital yang menerapkan sistem bus di mana beberapa kumpulan data yang berbeda fungsinya dilalukan pada sekumpulan kawat yang sama dalam kurun waktu yang berbeda, umpamanya multiplexing data dan alamat (address) memori pada komputer.

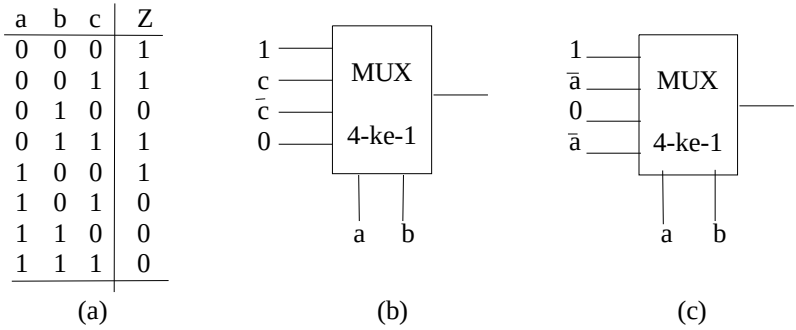
Walaupun multiplexer secara khusus berfungsi sebagai pemilih data, peralatan ini juga dapat digunakan untuk merealisasikan rangkaian kombinasi secara umum. Sebagai contoh, perhatikanlah fungsi yang tabel kebenarannya ditunjukkan pada Gambar 5.19(a).

Secara lengkap fungsi ini dapat dituliskan sebagai:

$$Z = \bar{a}\,\bar{b}\,\bar{c} + \bar{a}\,\bar{b}\,c + \bar{a}\,b\,\bar{c} + a\,\bar{b}\,\bar{c}$$
$$= \bar{a}\,\bar{b} + \bar{a}\,b\,c + a\,\bar{b}\,\bar{c}$$



Gambar 5.18. Rangkaian dasar Multiplexer.



Gambar 5.19. Realisasi dengan Multiplexer.

Perhatikan bahwa Z = 1 bila $\bar{a}\,\bar{b} = 1$. Bila $\bar{a}\,b = 1$, Z= c dan bila $a\bar{b} = 1$ maka Z= $\bar{c}$. Untuk kombinasi a dan b yang lain, Z= 0 (Ingat bahwa Z= 1 hanya bila salah satu sukumin penyusunnya berharga 1). Jadi, fungsi Z dapat direalisasikan dengan MUX dengan memilih **a** dan **b** sebagai masukan kendali dan **c** sebagai masukan data, seperti ditunjukkan dalam Gambar 5.19(b). Variasi lain dapat dibuat dengan memilih b dan c sebagai masukan kendali dan a sebagai masukan data. Dalam hal ini Z ditulis sebagai:

$$Z = \bar{a}\,\bar{b}\,\bar{c} + \bar{a}\,\bar{b}\,c + \bar{a}\,b\,c + a\,\bar{b}\,\bar{c}$$
$$= \bar{b}\,\bar{c} + \bar{a}\,\bar{b}\,c + \bar{a}\,b\,c$$

dan realisasinya seperti ditunjukkan pada Gambar 5.19(c).

5.8 Dekoder

Dekoder juga merupakan peralatan yang sangat umum pemakaiannya. Ditinjau dari keadaan aktif keluarannya, dekoder dapat dibedakan atas ***"non-inverted-outputs"*** dan ***"inverted-outputs"***. Rangkaian dekoder ***"inverted-outputs"*** menghasilkan satu, dan hanya satu, dari sejumlah keluarannya yang berlogika 0 sedangkan keluaran lain semua berlogika 1. Sebaliknya, dalam dekoder ***non-inverted-outputs*** hanya satu keluaran yang berlogika 1 sedangkan keluaran lain berlogika 0. Pemilihan keluaran ditentukan oleh sekumpulan masukan kendali. Untuk n sinyal kendali dapat dipilih satu dari 2^n keluaran yang berlogika 0 atau 1 (tergantung atas jenis keluaran dibalik atau tidak (inverted atau non-inverted outputs). Fungsi keluaran dekoder dapat dituliskan sebagai berikut:

non-inverted: $z_i = m_i$, i= 0,1,.. (2^n-1)

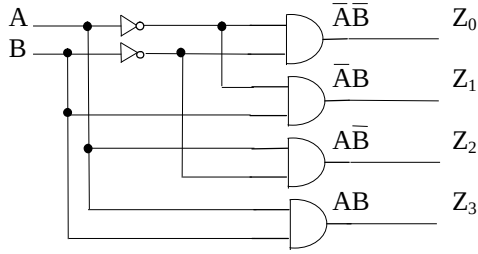
inverted: $z_i = \overline{m}_i = M_i$, i= 0,1,... (2^n-1)

dengan m_i dan M_i adalah sukumin dan sukumax dari peubah masukan/kendali.

Untuk dekoder 2x4 yang keluarannya tak dibalik, maka tabel kebenarannya dapat dibuat sebagai berikut:

a	b	z_0	z_1	z_2	z_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Perhatikan bahwa hanya 1 dari ke 4 keluarannya yang berkeadaan 1 sedangkan selebihnya berkeadaan 0. Dalam bentuk gerbang-gerbang diskrit skala kecil, rangkaian logika dekoder ini dapat disusun seperti pada Gambar 5.20. Dalam praktek sehari-hari, dekoder seperti yang ditunjukkan pada gambar ini jarang digunakan karena ketersediaan serpih dekoder khusus dalam skala menengah (MSI) yang lebih murah dan lebih rapi.



Gambar 5.20. Rangkaian dasar Dekoder 2x4

Salah satu serpih dekoder yang sangat banyak digunakan dalam sistem digital adalah dekoder 3-ke-8, yang tersedia dengan nomor tipe, antara lain, 74LS138 (buatan TI,Texas Instruments) dan 8205 (buatan Intel) dari jenis keluaran dibalik yang simbol logika dan tabel kebenarannya ditunjukkan pada Gambar 5.21. Perhatikan bahwa selain sinyal kendali pemilihan, pada umumnya juga disediakan tambahan kendali: $G_{2A,}$ G_{2B} dan G_1 pada 74LS138 dan E_3, E_2 dan E_1 pada 8205. Dengan kendali tambahan ini dimungkinkan pengoperasian yang tidak satu pun keluaran berkeadaan aktif dan, dalam praktek, ini sangat membantu menyederhanakan realisasi rancangan kita.

Sebagaimana multiplexer, dekoder juga dapat digunakan untuk merealisasikan fungsi-fungsi logika kombinasi. Sebagai contoh, perhatikan fungsi:

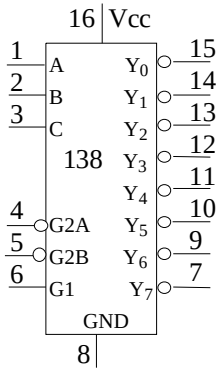
$$f = \Sigma\, m(3,5,6,7) = \bar{a}\, b\, c + a\, \bar{b}\, c + a\, b\, \bar{c} + a\, b\, c$$

$$= \overline{\bar{m}_3\, \bar{m}_5\, \bar{m}_6\, \bar{m}_7}$$

Sukumin-sukumin $\bar{m}_3$, $\bar{m}_5$, $\bar{m}_6$ dan $\bar{m}_7$ sudah tersedia sebagai keluaran dekoder dengan keluaran dibalik seperti 74LS138 atau 8205 di atas. Karena itu, untuk merealisasikan fungsi ini kita cukup mengambil keluaran dekoder yang menghasilkan sukumin-sukumin penyusun fungsi tersebut dan mengumpankan nya ke satu NAND. Dengan menggunakan dekoder 74LS138 di atas, misalnya, kita dapat tuliskan persamaan fungsi di atas sebagai berikut:

$$f = \overline{Y_3\, Y_5\, Y_6\, Y_7}$$

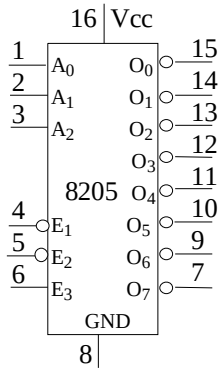
Dalam hal ini kita tidak perlu melakukan penyederhanaan fungsi. Untuk realisasi ini kita harus memberikan sinyal semestinya bagi kendali tambahan kendali yaitu: $G_1\, G_{2A}\, G_{2B} = 100$ untuk 74LS138 dan $E_3\, E_2\, E_1 = 100$ untuk 8205 seperti ditunjukkan dalam Gambar 5.22.



G1	G2	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
1	0	0	0	1	0	1	1	1	1	1	1	1
1	0	0	0	1	1	0	1	1	1	1	1	1
1	0	0	0	1	1	1	0	1	1	1	1	1
1	0	0	0	1	1	1	1	0	1	1	1	1
1	0	0	0	1	1	1	1	1	0	1	1	1
1	0	0	0	1	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	1	0	1
1	0	0	0	1	1	1	1	1	1	1	1	0
x	1	x	x	x	1	1	1	1	1	1	1	1
0	x	x	x	x	1	1	1	1	1	1	1	1

G2 = G2A+G2B

(a)



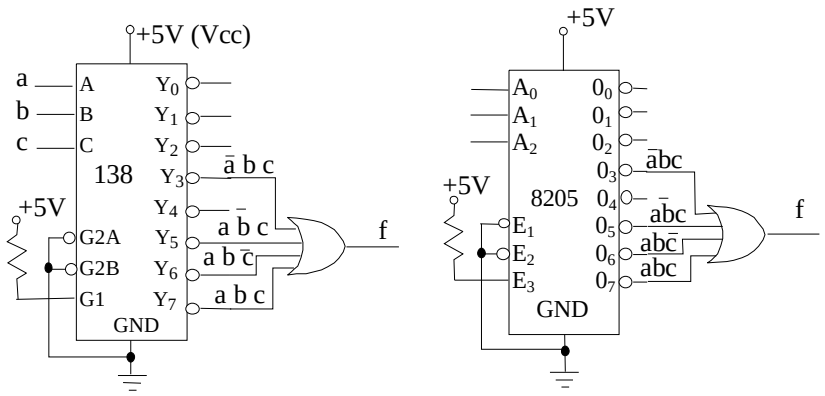
E_1	E_2	E_3	A_2	A_1	A_0	0_0	0_0	0_0	0_0	0_0	0_0	0_0	0_0
0	0	1	0	0	0	0	1	1	1	1	1	1	1
0	0	1	0	0	1	1	0	1	1	1	1	1	1
0	0	1	0	1	0	1	1	0	1	1	1	1	1
0	0	1	0	1	1	1	1	1	0	1	1	1	1
0	0	1	1	0	0	1	1	1	1	0	1	1	1
0	0	1	1	0	1	1	1	1	1	1	0	1	1
0	0	1	1	1	0	1	1	1	1	1	1	0	1
0	0	1	1	1	1	1	1	1	1	1	1	1	0
x	x	0	x	x	x	1	1	1	1	1	1	1	1
1	x	x	x	x	x	1	1	1	1	1	1	1	1
x	1	x	x	x	x	1	1	1	1	1	1	1	1

(b)

Gambar 5.21. Simbol logika dan tabel kebenaran dekoder 3x8.
(a) 74LS138, (b) 8205

5.9 ROM

ROM, singkatan daripada Read Only Memory, merupakan rangkaian penginget (memori) yang hanya dapat dibaca tanpa dapat ditulis. Pada dasarnya, ROM hanyalah rangkaian kombinasi dengan keluaran ganda (multiple output). Pemberian nama memori bagi rangkaian ini didasarkan atas kenyataan bahwa setiap kombinasi masukan tertentu akan memberikan kombinasi keluaran sesuai dengan rangkaian yang sudah disusun sebelumnya. Kombinasi masukan ini dipandang sebagai alamat sel memorinya sedangkan kombinasi keluarannya dipandang sebagai data yang disimpannya.



Gambar 5.22. Realisasi dengan dekoder 138 dan 8205

Penulisan data isi pengingat ini membutuhkan proses yang berbeda dari operasi normal. Ada jenis ROM yang hanya dapat ditulisi dalam proses fabrikasi dengan masker (mask) khusus; ada yang dapat difabrikasi dengan sejenis elemen lebur yang lebih murah harganya, biasa disebut sebagai field-programmable ROM atau Eraseable Programmable (EPROM). Jenis pertama, yang biasa disebut MROM (Mask ROM) atau ROM saja, merupakan jenis ROM yang paling mahal dan hanya dibuat dalam jumlah besar, lebih dari 1000 serpih (chip). EPROM merupakan jenis ROM yang sangat banyak dipergunakan dan sangat membantu dalam pekerjaan perancangan dan pembuatan pertama kali suatu sistem digital. Penghapusan isi EPROM yang banyak dilakukan adalah dengan memberikan sinar ultra-violet (UV) yang cukup banyak sehingga sering juga disebut UVEPROM. Tetapi sekarang sudah banyak juga dipakai yang dapat dihapus dengan aliran listrik tegangan agak tinggi, jenis ini disebut juga sebagai EAROM (Electrically Alterable ROM) atau EEPROM (Electrically Eraseable Programmable ROM).

ROM terdiri atas 2 bagian: **dekoder** di bagian masukan dan **matrik OR** di bagian keluaran. Bagian masukan menyediakan sukumin dan bagian keluaran menjumlahkan (meng-OR-kan) sukumin penyusun fungsi keluaran. Matrik OR pada keluaran direalisasikan dengan diode yang berfungsi sebagai saklar (switch). Diode saklar bagi sukumin penyusun dihubungkan, diode saklar bagi sukumin yang bukan penyusun dilepas/diputus. Kombinasi yang timbul pada keluaran dapat dipilih dengan memberikan kombinasi tertentu pada masukan pemilih pada dekoder. Jadi, pemilihan kombinasi masukan sama dengan pemilihan alamat dalam pengertian memori secara umum. Kumpulan keluaran dapat dipandang seba-

gai satu kesatuan membentuk kata data (data word) dan cacah saluran keluaran merupakan cacah bit dalam satu kata dan disebut sebagai panjang kata.

Untuk melihat bagaimana struktur internal suatu ROM, kita perhatikan fungsi-fungsi berikut:

$$f_0 = \Sigma\, m(2,3,5,6,7) = m_2\, m_3\, m_3\, m_6\, m_7$$
$$= \bar{a}\,b\bar{c} + \bar{a}\,bc + a\bar{b}c + ab\bar{c} + abc$$

$$f_1 = \Sigma\, m(0,1,2,6) = m_0\, m_1\, m_2\, m_6$$
$$= \bar{a}\,\bar{b}\,\bar{c} + \bar{a}\,\bar{b}\,c + \bar{a}\,b\bar{c} + ab\bar{c}$$

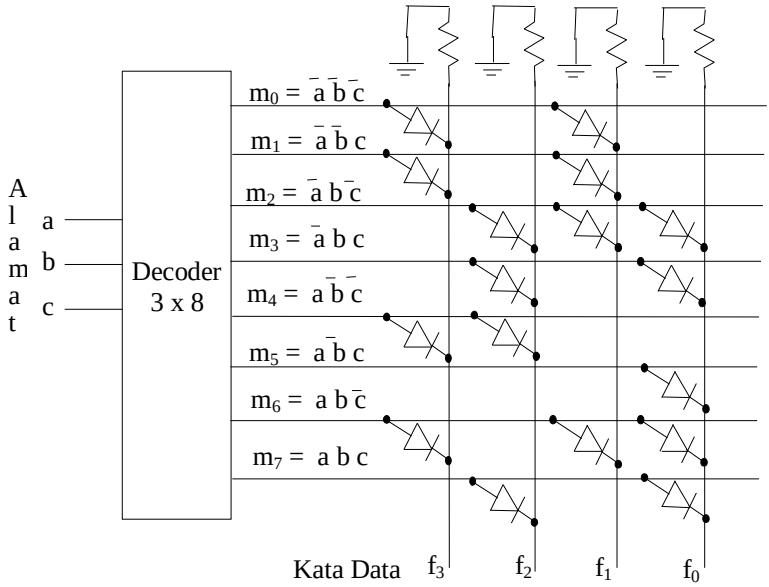
$$f_2 = \Sigma\, m(2,3,4,7) = m_2\, m_3\, m_4\, m_7$$
$$= \bar{a}\,b\bar{c} + \bar{a}\,bc + a\bar{b}\bar{c} + abc$$

$$f_3 = \Sigma\, m(0,1,4,6) = m_0\, m_1\, m_4\, m_6$$
$$= \bar{a}\,\bar{b}\,\bar{c} + \bar{a}\,\bar{b}\,c + a\bar{b}\,\bar{c} + a\,b\bar{c}$$

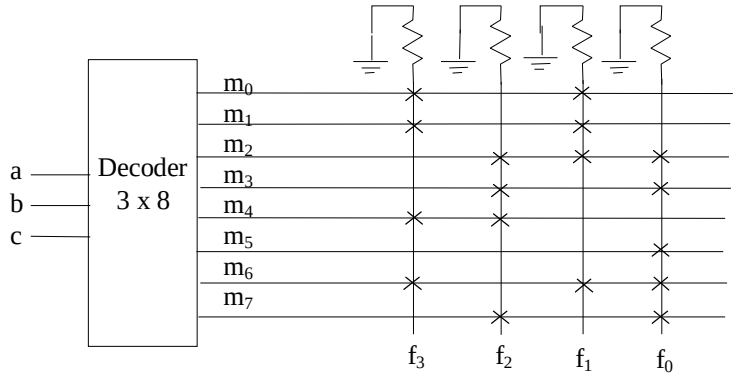
Untuk merealisasikan fungsi dengan ROM, semua sukumin penyusun disajikan dengan lengkap. Berdasarkan sukumin-sukmin tersebut ditentukan diode mana yang disambung dalam ROM. Realisasi ROM untuk fungsi-fungsi di atas ditunjukkan pada Gambar 5.23.

Untuk penyederhanaan penggambaran, elemen saklar pada Gambar 5.23(a) sering digambarkan dengan simpul berupa titik atau tanda x seperti pada Gambar 5.23(b). Dari Gambar 5.23 dapat dilihat bahwa bila sukumin m_i akan muncul sebagai salah satu penyusun di satu keluaran ROM, maka elemen saklar (switching element) yang menghubungkan keluaran dekoder m_i dengan saluran keluaran tersebut dibuat terhubung. Karena sukumin m_6, misalnya, penyusun bagi fungsi f_0, maka saluran m_6 dihubungkan ke saluran f_0. Bila m_6 berlogika 1, maka f_0 akan ikut berlogika 1. Tetapi bila m_6 dan semua saluran m_i yang lain yang terhubung ke f_0 berlogika 0, maka f_0 akan mengikuti tegangan *ground* (0 V) dan juga berlogika 0.

Dalam realisasi dengan ROM ini, walaupun ada sukumin keluaran dekoder yang tak digunakan, tak tersambung ke salah satu saluran keluaran, kombinasi alamat bersangkutan tetap didekodekan. Jadi, ditinjau dari segi memori, ada sel yang tak pernah diakses.



(a)



(b)

Gambar 5.23. Contoh realisasi dengan ROM

5.10 PLA dan PAL

PLA (Programmed Logic Array) hampir sama dengan ROM dalam fungsi yang dilakukannya. Jadi PLA juga digunakan untuk merealisasikan fungsi-fungsi logika yang berkeluaran ganda dengan masukan juga ganda (MIMO, Multiple Input Multiple Output). Perbedaan antara PLA dengan ROM terletak pada rangkaian awalnya. Dekoder yang menentukan alamat kata yang akan dibaca pada ROM, dalam PLA digantikan dengan matriks AND untuk menghasilkan sukumin dari peubah masukan. Dalam hal ini, hanya sukumin yang dibutuhkan saja yang direalisasikan sehingga fabrikasinya lebih hemat dibandingkan ROM, terutama jika banyak sukumin yang tak digunakan.

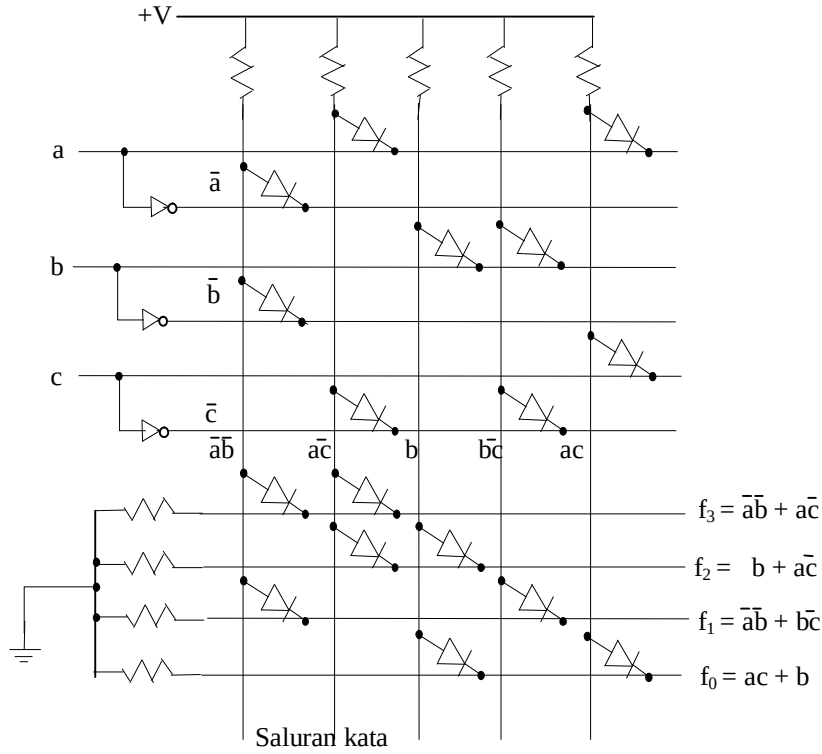
Dalam Gambar 5.24 ditunjukkan realisasi fungsi-fungsi yang ditunjukkan pada Gambar 5.23 sebelumnya. Realisasi rangkaian tersebut diperoleh setelah dilakukan penyederhanaan fungsi menjadi:

$f_0 = \Sigma\, m(2,3,5,6,7) = a\,c + b$
$f1 = \Sigma\, m(0,1,2,6) = \bar{a}\,\bar{b} + b\,\bar{c}$
$f_2 = \Sigma\, m(2,3,4,7) = b + a\,\bar{c}$
$f3 = \Sigma m(0,1,4,6) = \bar{a}\,\bar{b} + \bar{a}\,c$

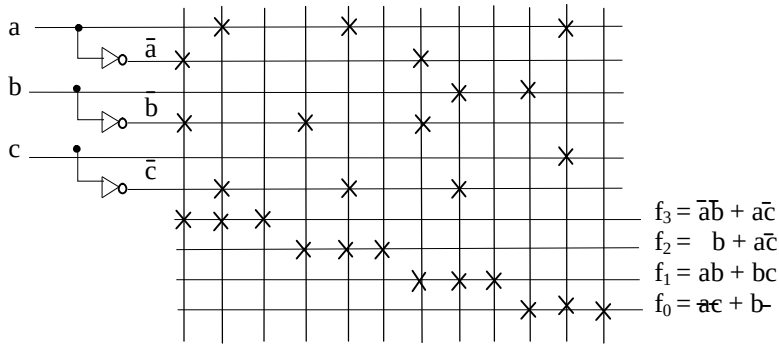
Perhatikan bahwa pemasangan elemen saklar antara saluran masukan dengan sumber tegangan positif membentuk operasi AND karena satu saluran kata yang terhubung ke sumber tegangan positif akan berlogika 1 hanya bila semua saluran masukan yang terhubung ke saluran kata tersebut berlogika 1. Seperti pada ROM, saluran kata dan saluran keluaran dihubungkan secara OR. Jadi sebenarnya PLA merupakan kombinasi matriks AND dan matriks OR. Perhatikan juga bahwa realisasi dengan PLA membutuhkan proses penyederhanaan (minimisasi) terlebih dahulu, berbeda dengan ROM yang membutuhkan penyediaan semua sukumin.

PAL (Programmable Array Logic) merupakan rangkaian kombinasi terprogram yang semakin luas penggunaannya dalam merealisasikan sistem logika dengan banyak keluaran dan tersedia dalam bentuk serpih rangkaian terpadu sekala menengah dan besar (MSI dan LSI). Perbedaan antara PAL dan PLA terletak pada rangkaian keluarannya. Kalau dalam PLA, baik matrik AND di bagian masukan maupun matriks OR di bagian keluaran dapat diprogram, maka dalam PAL hanya matrik AND di bagian masukan yang dapat diprogram, sedangkan matriks OR di bagian keluaran sudah terhubung tetap.

Fungsi-fungsi yang sebelumnya telah direalisasikan dengan ROM dan PLA, dapat direalisasikan dalam PAL dengan hubungan seperti ditunjukkan Gambar 5.25. Semua jalur alamat yang bertanda x pada satu jalur keluaran matriks AND (bagian masukan) membentuk satu sukumin dan semua sukumin yang bertanda x pada satu jalur keluaran di-OR-kan dan menghasilkan fungsi keluaran f. Untuk membentuk sukumin ac bagi fungsi f_0, sebagai contoh, maka alamat a dan c dihubungkan ke jalur yang pada gilirannya akan membentuk f_0.



Gambar 5.24. Realisasi dengan PLA



Gambar 5.25. Realisasi dengan PAL

5.11 Soal Latihan

1. Gambarkanlah diagram pohon 3 tingkat dan 4 tingkat daripada fungsi:

   $f = (AB + C)(D + E + FG) + H$

2. Dengan menggunakan gerbang OR dan AND, tentukanlah realisasi minimum dengan logika 2 tingkat dan 3 tingkat dari fungsi

   $f = \Sigma\, m\,(0,3,4,6,7,8,11,12,13,15)$

   Tentukanlah harga realisasi masing-masing ?

3. Dengan menggunakan gerbang NOR dalam logika 2 tingkat realisasikanlah fungsi minimum dari fungsi

   $f= \Sigma\, m(0,2,5,8,10,11,12,13,15,18,21,26,27,28,29,30,31)$

   Ulangi untuk realisasi dengan 3 tingkat NOR.

4. Buatlah rangkaian yang dapat menjumlahkan dua bilangan BCD (Binary Coded Decimal) 1 digit. Lanjutkan untuk dua bilangan BCD 3 digit.

5. Realisasikanlah fungsi minimum dari

   $f(a,b,c,d) = \Sigma\, m\,(0,1,2,4,5,7,11,15)$

   dengan menggunakan multiplexer (MUX) 8-ke-1 dengan masukan kontrol a,c, dan d. Lakukan juga dengan MUX 4-ke-1, tentukan sendiri masukan kontrolnya.

6. Realisasikanlah fungsi dalam soal nomor 5 dengan menggunakan dua buah dekoder 3 x 8.

7. Rancanglah rangkaian logika untuk mengubah kode dari BCD ke Gray dan gambarkan rangkaiannya.

8. Realisasikanlah fungsi pengubah kode dalam soal nomor 7 dengan menggunakan: a. ROM b. PLA