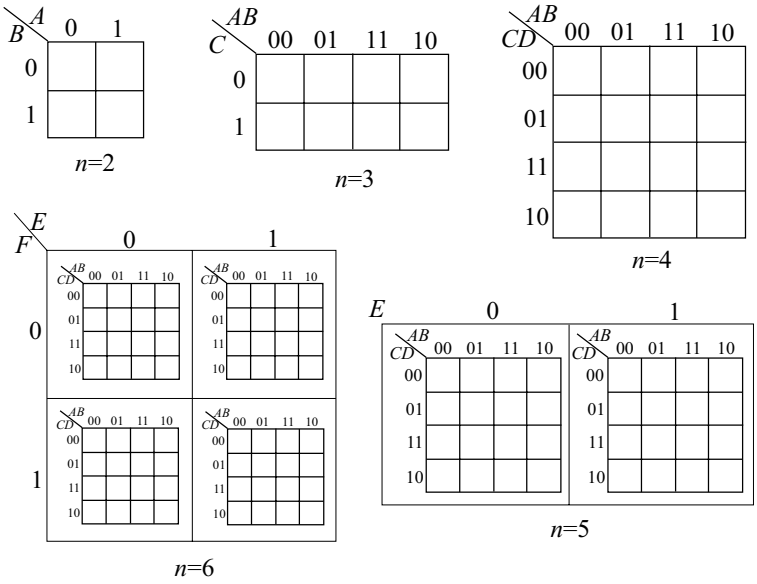

図 1.3　カルノー図の枠組み（$n = 2, 3, 4, 5, 6$ の場合）

示す.

[3]　カルノー図による最小項と標準積和形の表現

n 変数論理関数 $f(X_1, X_2, \ldots, X_n)$ 用カルノー図の枠組みでは，ある変数値の組 $(x_1, x_2, \ldots, x_n)$ （$x_i = 0$ か 1) に対して，それを座標ラベルにして一意にマス目が最小項として識別できる．そのマス目に，最小項 $f(x_1, x_2, \ldots, x_n)$ の値（論理値，“0” か “1”）を記入する.

これで，論理関数 f の**カルノー図**による表現が完成する.

注意　カルノー図のマス目には論理値 “1” か “0” を記入する．“1” が記入されていないところは “0” であるから，“1” だけを記入し，“0” のマス目は空白にしておいても紛れはない.

もう一つ重要なことがある．カルノー図のマス目は最小項を示しているので，“1” が記入されたマス目の座標ラベルを「最小項を示すリテラル（の AND)」として読み取り，それらを OR で結んでできる論理式が**標準積和形**である.

出現する頻度が高いので，特に，前述の手順の 3（ド・モルガンの定理）を活用しなければならない．

例　題

2.4　図 2.28の NAND 回路を解析しなさい．

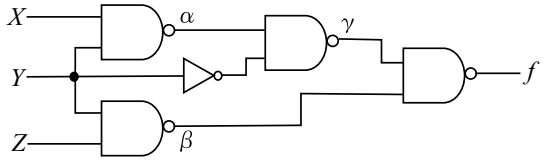

図 2.28　NAND 回路例（例題 2.4）

［解］

$$\alpha = \overline{X\,Y}$$
$$\beta = \overline{Y\,Z}$$
$$\gamma = \overline{\alpha\,\overline{Y}} = \overline{\alpha} + Y = X\,Y + Y = Y$$
$$f = \overline{\gamma\,\beta} = \overline{\gamma} + \overline{\beta} = \overline{Y} + Y\,Z$$

2.3.2　組み合わせ回路の最適化設計

組み合わせ回路の最適化設計の目的や評価指標，および，評価手法の概要について学ぶ．

[1]　組み合わせ回路の設計

任意の論理関数から組み合わせ回路を設計する場合には，元の論理関数の形式に応じた論理回路を設計するのが最も簡単である．

たとえば，積和形論理関数から AND-OR 回路を，和積形論理関数から OR-AND 回路を，一般的な AND/OR 形式の論理関数から一般的な AND/OR 回路を，それぞれ設計することは簡単である．また，任意の論理関数は標準積和形あるいは標準和積形のいずれかに必ず変換できることを利用すれば，変換後の標準形の論理関数から AND/OR 回路を簡単に設計できる．

によって達成できるが，両者を同時に満足する最適化設計が可能であろうか？ここでは，それを確かめるために，最適化設計の評価指標について考える．

定義 2.28（段数）　論理回路の入力端子から出力端子に至るまでに通過する論理ゲート数を**段数**という．

既出であるが，入力端子から出力端子に向かって電気信号が流れることから，入力端子に近い方を「前段」，それから遠い（出力端子に近い）方を「後段」とそれぞれいう．

注意　特に断りがない場合には，多入力論理（AND と OR）ゲートも「1 個」および「1 段」と数える．したがって，論理関数（論理式）における論理積項 $(P_1 \cdot P_2 \cdot \cdots \cdot P_m)$ あるいは論理和項 $Q_1 + Q_2 + \cdots + Q_n$ は，それぞれ 1 個の m 入力 AND ゲートあるいは 1 個の n 入力 OR ゲートで実現できる．ただし，P_i や Q_i は，リテラルでなくても，論理式であれば何でも構わない．多入力ゲートを実際に電気的に実現すると，等価な複数個・複数段の 2 入力ゲートによって構成する組み合わせ回路よりも空間サイズ，時間サイズとも小さくなる．

最適化の目的はⒶとⒷであるので，**論理回路**の最適化設計に対する評価指標は次の 2 種類である．

1. **空間最適化**：「論理回路を構成するのに必要な論理ゲート（信号線も含む）の総数」を最小にする．
2. **時間最適化**：「論理回路の入力端子から出力端子に至るまでに通過する論理ゲートの**段数**（定義 2.28）」を最小にする．入力端子から出力端子に至るまでの経路（パス (path)）は普通複数あるが，時間最適化では，このうち最も多い段数（最長）の経路（**クリティカルパス** (critical path) という）の段数を最小にする．

これら 2 種類の最適化はいずれも「論理ゲート数を最小にする」と定義しているが，残念ながら,「総数」と「段数」は同じことを意味していないので，2 種類の最適化を同時に（同じ論理回路として）達成するのは難しい．

注意　論理回路の最適化設計で対象とする論理ゲートは，特に断りがないときは，AND ゲートと OR ゲートである．基本論理ゲートのうち，NOT ゲートやある信号の NOT は，NOT 機構の電気的な作り方（本書では，ブラックボックスとした）の観点から見ると，AND や OR に比べると簡単であり，論理ゲート数としては無視で

定義 2.29（次元）　論理関数の論理積 (AND) 項と論理和 (OR) 項を，多項演算も許して，1 つずつカッコでくくると，カッコの重なり（ネスト (nest) という）ができる．多重のカッコのそれぞれは，最内側を最優先とし，外側へ向かって低くなる演算順位を示す．最内側から外側に向かって，$1, 2, \cdots$ と演算順位を付けたとき，これを**次元**あるいは**レベル** (level) という．また，その論理式における最大次元（最外側のカッコの次元）を単に**次元**ということがある．次元は，実際には冗長で省略できるカッコも数える．

たとえば，積和形や和積形は 2 次元，$((X \cdot \overline{Y} \cdot Z) \cdot (Y + (\overline{Z} \cdot \overline{X})))$ は 3 次元である．また，後者の論理式において，「$\overline{Z} \cdot \overline{X}$ は第 1 次元の論理積演算」，「$Y + (\overline{Z} \cdot \overline{X})$ は第 2 次元の論理和演算」などという．

空間サイズと時間サイズのそれぞれの最適化は，**論理関数**（論理式）では，次の操作となる．

1. 論理関数の**空間最適化**：「その論理関数に現れる**リテラル総数**」を最小にする．リテラル総数が最小であれば，「1 つのカッコでくくれる多項論理積（AND）演算および多項論理和（OR）演算の総数（和）」（本書では，これを**多項論理演算数**といい，“S_T”と表記する）も最小である．
2. 論理関数の**時間最適化**：多項論理積 (AND) 演算（項）と多項論理和 (OR) 演算（項）による**次元**を最小にする．次元の定義（定義 2.29）における「最大次元」が論理回路における「クリティカルパスの段数」に対応する．

注意　前述の論理関数上での最適化操作で使っている多項論理積 (AND) 項や多項論理和 (OR) 項においては，AND や OR で結ぶ各項は論理式であり，リテラルとは限らない．たとえば，$Y \cdot Z + \overline{X} + X \cdot \overline{Y} + \overline{Z}$ では，最外側（第 2 次元）の論理和演算を構成する演算項は，$Y \cdot Z$, $\overline{X}$, $X \cdot \overline{Y}$, $\overline{Z}$ の 4 項（論理式）である．

注意　論理積項や論理和項が多項である場合には，それらの演算項は，論理回路上での「多入力論理ゲート」に対応する．論理回路の空間サイズを 2 入力論理ゲートの総数で換算する場合には，S_T ではなく，論理関数（論理式）の 2 項演算総数（“$\cdot$”と“$+$”の総数）を数えればよい．また，リテラル総数は (2 項演算総数 $+1$) である．

これらの論理関数（論理式）上での 2 種類の最適化は目標が異なるので，両者を両立させるのは難しい．また，両者を同時に満足する最適化手法はない．

2.3.8　多段論理最小化

2 段論理回路という枠を取り外して，空間最適化をより追求する最適化設計手法について考えてみよう．

[1]　論理式の変形による多段論理最小化

論理関数における空間最適化を図る方法の 1 つに，**ファクタリング** (factoring) がある．ファクタリングは，定理 1.10（分配則）を用いて，論理式から共通項をくくり出す操作である．

ファクタリングができると，リテラル総数すなわち 2 項論理演算数（“$\cdot$” と “$+$” の総数）だけでなく，多項論理演算数（多項論理積演算数と多項論理和演算数の和）S_T も減る．

たとえば，分配則の式 1.26 そのものも，

$$((X \cdot Y) + (X \cdot Z)) = (X \cdot (Y + Z))$$

という「X のくくり出し（ファクタリング）」である．この左辺の S_T は “3” であるが，X をくくり出した右辺の S_T は “2” に減っている．

ファクタリングは,「論理積項や論理和項の項数を減らす」操作であるから，論理回路上での「論理ゲートへの入力信号線を減らす」操作に対応する．また，S_T が減る場合には必要な論理ゲート総数も減る．論理回路の空間サイズを 2 入力論理ゲートで換算する場合には，ファクタリング後の論理関数に対応する論理回路の空間サイズ（2 入力論理ゲートの総数）は減る．

しかし，一方で，ファクタリングは次元を増やす操作であり，時間最適化とは逆行する．

定義 2.34（多段論理最小化）　「論理回路が多段になる（時間サイズが増大する）ことを許す」という前提で論理回路の空間最適化を図ることを**多段論理最小化**という．

論理関数（論理式）に対して行う多段論理最小化は,「次元が増えることを犠牲にして，リテラル総数（2 項論理演算数）や多項論理演算数 S_T を減らす」操作にあたる．

具体的に，ファクタリングによる多段論理最小化について，次の例題によって考えてみよう．

例　題

2.14　式 $f(W,X,Y,Z) = W\,X\,Z + \overline{W}\,\overline{X}\,Y + W\,X\,Y + \overline{W}\,\overline{X}\,Z$ をファクタリングして空間最適化しなさい．また，その結果と時間最適化との関係について説明しなさい．

［解］　与式は 2 次元であり，そのリテラル総数は “12”，S_T は “5” である．これをファクタリングすると，

$$\begin{aligned} f(W,X,Y,Z) &= \big((W\cdot X\cdot Z) + \big(\overline{W}\cdot\overline{X}\cdot Y\big) \\ &\qquad + (W\cdot X\cdot Y) + \big(\overline{W}\cdot\overline{X}\cdot Z\big)\big) \\ &= \big((W\cdot X\cdot (Y+Z)) + \big(\overline{W}\cdot\overline{X}\cdot (Y+Z)\big)\big) \\ &= \Big(\big((W\cdot X) + (\overline{W}\cdot\overline{X})\big)\cdot (Y+Z)\Big) \end{aligned}$$

となる．S_T は “5” で変わらないが，リテラル総数は “6” に減る．また，結果は 3 次元となり，時間サイズは悪くなっている．

ファクタリングは「共通項のくくり出し」という明確な操作である．しかし，それ以外の式変形（たとえば，種々の定理を利用する）操作を用いて，任意の形式の論理式に対して空間最適化あるいは時間最適化を行うのは，難しい．その理由は以下の通りである．

- 最適化の限界（これ以上最適化できない）が分かりにくい．
- 適用する変形操作を見つける一定の規則がないので，場当り的に操作をしなければならない．
- 変形過程が異なると最適化結果が異なる場合がある．

例　題

2.15　式 $f(X,Y,Z) = X\,Y + Y\,Z + Z\,\overline{X}$ をカルノー図や Q-M 法を使わない式変形操作だけで空間最適化しなさい．

表 2.34　多数決関数例の主項－最小項表

	最小項		√ 3	√ 5	√ 6	√ 7
	主項					
√	3, 7	p	◎			○
√	5, 7	q		◎		○
√	6, 7	r			◎	○

表 2.35　重み付き多数決関数例の真理値表（横書き）

A	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
B	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
C	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
D	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
M	0	0	0	0	0	0	1	1	0	1	1	1	1	1	1	1

となる．多段論理最小化によって，多項論理演算数 S_T は “5” から “4” に，リテラル総数は 8 個から 6 個に，それぞれ減る．これに対応する AND/OR 回路は図 2.57となる．

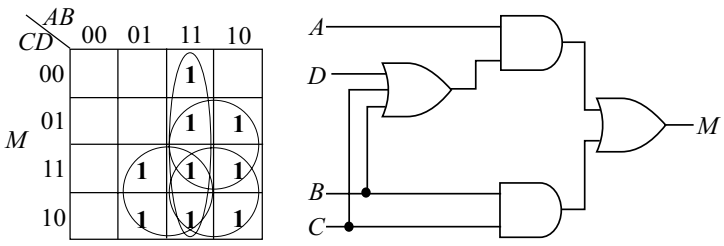

図 2.57　重み付き多数決関数例の多段論理最小化 AND/OR 回路（カルノー図付き）

これをカルノー図（省略）などに写すことによって，入力条件式と出力

表 3.22　2 ビット列比較回路の拡大入力要求表（状態遷移表）

I	Q_2	Q_1	Q_2^+	Q_1^+	P	D_2	D_1
0	0	0	0	1	0	0	1
0	0	1	0	0	0	0	0
0	1	0	0	0	1	0	0
0	1	1	—	—	—	—	—
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	1	0	0	0	0	0	0
1	1	1	—	—	—	—	—

関数を求める．ドントケアを利用すると，出力 P を論理最小化できる．

$$\begin{cases} P = \overline{I}\,Q_2 + I\,Q_1 \\ D_1 = \overline{I}\,\overline{Q_2}\,\overline{Q_1} \quad D_2 = I\,\overline{Q_2}\,\overline{Q_1} \end{cases}$$

となる．

3.3.3　同期式順序回路設計における最適化

組み合わせ回路設計における最適化（論理最小化）手法を同期式順序回路設計に適用する手順については，前の 3.3.1 項の [3]（180ページ）で学んだ．本項では，それ以外の，順序回路設計独特の最適化手法について紹介する．

[1]　完全指定順序回路における状態数の決定と最小化

前の 3.3.1 項の [3]（180ページ）で学んだ同期式順序回路設計手順の 1 で示したように,「順序回路の**状態**および**状態数**を決定する」ことが順序回路を実現するための第一歩である．ここでは，この順序回路設計の初期段階で可能な最適化について考える．

順序回路の状態数は必要となるフリップフロップ数に直接影響する．必要なフリップフロップ個数が多くなればなるほど，フリップフロップ入力条件式の

定義 3.14（順序回路の等価性）　順序回路 S_a と S_b が同一の入力（端子）と同一の出力（端子）を持つ．S_a のすべての状態 s_i に対して，S_b がそれと等価な状態 s_j を持ち，かつ，S_b のすべての状態 s_j に対して，S_a がそれと等価な状態 s_i を持つとき,「順序回路 S_a と S_b とは**等価**である」あるいは「S_a と S_b とは**等価**な順序回路である」という．これを $S_a \equiv S_b$ と表記する．

この順序回路の**等価性**は，言い換えると,「s_i と s_j を順序回路 S_a, S_b それぞれの初期状態とするとき，S_a, S_b に任意の入力系列を与えると，それぞれの順序回路が同じ出力系列を生成する」ことを表している．これが「まったく同じ動作」の厳密な（数学的）定義である．

定義 3.15（順序回路の最小化）　定義 3.14 にしたがうと,「ある順序回路 S と等価で状態数が最小の順序回路 S^* を求める」ことが「順序回路の**最小化**」といえる．また，このようにして作った順序回路 S^* を「順序回路の**最小形**あるいは**最簡形**」という．

状態数の最小化については [1] で与えた．すなわち，等価な状態を併合して状態数を最小化して作った順序回路 S^* は，状態名の付け替えを除くと，一意に定まる．この S^* が順序回路 S の最小形である．

[3]　不完全指定順序回路における状態数の決定

次に，状態（遷移）や出力にドントケアを含む**不完全指定順序回路**における状態最小化について考える．

不完全指定順序回路については，3.3.2 項（183ページ）で既に学んでいる．また，そこではドントケアの状態そのものを利用して順序回路の論理最小化を図る方法についても紹介した．

ここでは，状態数の最適化（最小化）にドントケアを利用する方法について考える．

まず，ドントケアをさらに一般化して，状態数の最小化に利用する．

3.3.2 項では，**ドントケア**は「状態」であった．したがって，状態遷移表では，ドントケアである現状態に対応する行の次状態欄や出力欄および入力要求欄すべてに “—” を記入した．

表 3.34　3 ビット 2 進カウンタの拡大入力要求表（状態遷移表）

Q_3	Q_2	Q_1	Q_3^+	Q_2^+	Q_1^+	J_3	K_3	J_2	K_2	J_1	K_1
0	0	0	0	0	1	0	−	0	−	1	−
0	0	1	0	1	0	0	−	1	−	−	1
0	1	0	0	1	1	0	−	−	0	1	−
0	1	1	1	0	0	1	−	−	1	−	1
1	0	0	1	0	1	−	0	0	−	1	−
1	0	1	1	1	0	−	0	1	−	−	1
1	1	0	1	1	1	−	0	−	0	1	−
1	1	1	0	0	0	−	1	−	1	−	1

トカウンタと同じであることに注意してほしい．回路図は省略する．

$$\begin{cases} J_3 = Q_2\,Q_1 \\ K_3 = Q_2\,Q_1 \end{cases} \qquad \begin{cases} J_2 = Q_1 \\ K_2 = Q_1 \end{cases} \qquad \begin{cases} J_1 = 1 \\ K_1 = 1 \end{cases}$$

---

2 進カウンタには，

- ビット長（状態数）が増えても回路構成は簡単である．

という特徴がある．

(B) **グレイコード カウンタ**

1.3.3 項の [2]（30ページ）で説明したように，カルノー図の座標ラベルとして使っている 2 進数列は**グレイコード** である．このグレイコードの並び順に状態遷移するカウンタを**グレイコード カウンタ**という．

たとえば，2 ビットグレイコードカウンタの状態遷移図は図 3.29となる．状態遷移する状態間では，Q_1, Q_2 のどちらか 1 つだけしか反転（“0”⇔“1”）しないことがグレイコード（カウンタ）の特徴である．

この 2 ビットグレイコードカウンタを 2 個の JK フリップフロップ FF_1, FF_2 を用いて設計してみよう．状態遷移図より状態遷移表と拡大入力要求表（表 3.35）を得る．

- 硬貨投入口は 1 個とし，一時に 50 円硬貨と 100 円硬貨のいずれかだけを受け付ける．
- 品物の値段は 150 円とする．

この自動販売機を D フリップフロップを用いて設計しなさい．

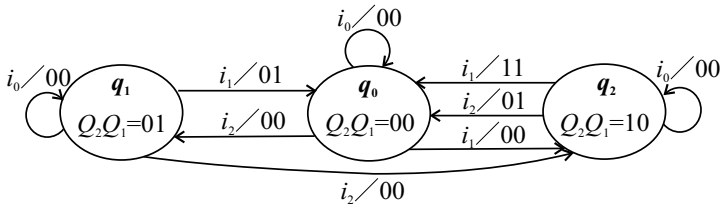

図 3.39　自動販売機（例題 3.17）の状態遷移図（状態ラベルによる）

表 3.40　自動販売機（例題 3.17）の状態遷移表（状態ラベルによる）

現状態 q	次状態 q^+ ／出力 MG		
	入力 i_0	入力 i_1	入力 i_2
q_0	q_0 ／ 00	q_2 ／ 00	q_1 ／ 00
q_1	q_1 ／ 00	q_0 ／ 01	q_2 ／ 00
q_2	q_2 ／ 00	q_0 ／ 11	q_0 ／ 01

表 3.41　自動販売機（例題 3.17）の状態遷移表（拡大入力要求表）

Q_2	Q_1	Q_2^+ Q_1^+						M G					
		$I_2\,I_1$						$I_2\,I_1$					
		00		01		10		00		01		10	
0	0	0	0	1	0	0	1	0	0	0	0	0	0
0	1	0	1	0	0	1	0	0	0	0	1	0	0
1	0	1	0	0	0	0	0	0	0	1	1	0	1

［解］　まず，状態を決定し，それから状態割り当てを行う．仕様より，入力（ラベル）を硬貨無投入：i_0，100 円硬貨投入：i_1，50 円硬貨投入：

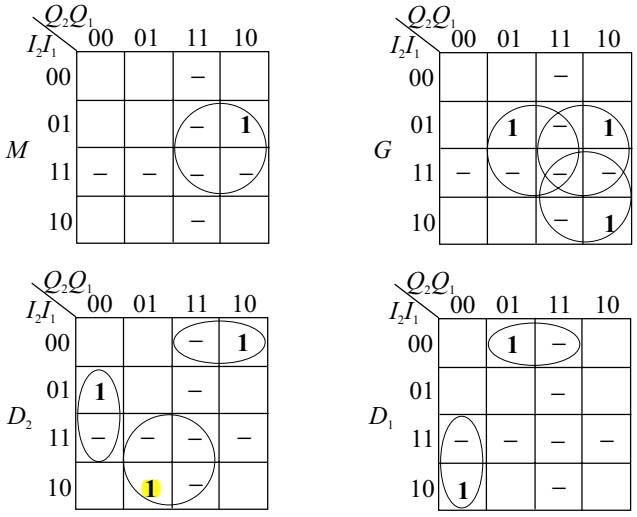

図 3.40　自動販売機（例題 3.17）のカルノー図

i_2 とする．そして，状態（ラベル）を 0 円を入力した状態：q_0，50 円を入力した状態：q_1，100 円を入力した状態：q_2 の 3 状態とする．状態遷移図は図 3.39，それによって作成した状態遷移表は表 3.40，となる．q_0, q_1, q_2 のそれぞれはある入力によって出力（値）が異なるから，状態数のこれ以上の最小化はできない．2 個の D フリップフロップ $Q_2\,Q_1$（並び順に注意）への状態ラベル q_0, q_1, q_2 の状態割り当てを

$$q_0 \Rightarrow 00 \quad q_1 \Rightarrow 01 \quad q_2 \Rightarrow 10$$

とする．$Q_2\,Q_1 = 11$ はドントケアとする．また，入力（端子）$I_2\,I_1$（並び順に注意）への入力ラベル i_0, i_1, i_2 の割り当てを

$$i_0 \Rightarrow 00 \quad i_1 \Rightarrow 01 \quad i_2 \Rightarrow 10$$

とする．$I_2 I_1 = 11$ はドントケアとする．状態ラベルによる状態遷移表（表 3.40）を論理値による状態遷移表（拡大入力要求表，ドントケアは省略）に変換すると，表 3.41となる．これをカルノー図（図 3.40）など

$\overline{X_i}$ を入れ替え，演算記号の $\cdot$（AND）と $+$（OR）を入れ替えて，

$$\overline{L} = f\left(\overline{X_1}, X_1, \overline{X_2}, X_2, \ldots, \overline{X_n}, X_n, +, \cdot\right)$$

と形式的に表せる．このとき，次が成り立つ．

$$\begin{aligned} & f\left(X_1, \overline{X_1}, X_2, \overline{X_2}, \ldots, X_n, \overline{X_n}, \cdot, +\right) \\ & = \overline{f\left(\overline{X_1}, X_1, \overline{X_2}, X_2, \ldots, \overline{X_n}, X_n, +, \cdot\right)} \end{aligned} \tag{1.38}$$

証明　2 変数から始めて，数学的帰納法によって n 変数に拡張する．（省略）

定義 1.6（双対関数）　ド・モルガンの定理（定理 1.16）における n 変数論理関数 f に対して，すべての変数の否定（NOT）をとり，かつ，論理関数全体に NOT を施して得る n 変数論理関数を**双対関数**という．本書では，f の双対関数を f^d と表記する．すなわち，形式的には，

$$f^d = \overline{f\left(\overline{X_1}, X_1, \overline{X_2}, X_2, \ldots, \overline{X_n}, X_n, \cdot, +\right)}$$

と表せる．

注意　双対な論理式や双対関数を得る操作では，演算記号の AND(“$\cdot$”) と OR(“$+$”) は入れ替えない．この点がド・モルガンの定理と異なる．すなわち，L^d は $\overline{L}$ ではない．

ここで，ド・モルガンの定理（定理 1.16）の式 1.38によって，

$$\begin{aligned} f^d &= \overline{f\left(\overline{X_1}, X_1, \overline{X_2}, X_2, \ldots, \overline{X_n}, X_n, \cdot, +\right)} \\ &= f\left(X_1, \overline{X_1}, X_2, \overline{X_2}, \ldots, X_n, \overline{X_n}, +, \cdot\right) \end{aligned}$$

である．これを f と比較すると，「論理関数 f の**双対関数** f^d は，f の AND($\cdot$) と OR($+$) を入れ替えて得る」ことが分かる．

定義 1.7（自己双対関数）　n 変数論理関数 f とその双対関数 f^d が同値である（等しい）とき，すなわち，

$$f = f^d$$

ならば，f を**自己双対関数**という．

奇数個（1 個か 3 個）のものの論理和.

[問題 2.2]　すべての最小項の包含条件（定理 2.7）U は次の通り.

$$\begin{aligned} U &= p\,t\,(p+q)\,(r+t)\,(q+s)\,(r+s) = p\,t\,(q+s)\,(r+s) \\ &= p\,t\,(s+q\,r) = p\,s\,t + p\,q\,r\,t \end{aligned}$$

したがって，最小積和形 f_m は次の通り.

$$f_m = p + s + t$$

[問題 2.3]　最小積和形は次の通り.

$$\overline{A}\,\overline{B}\,C + \overline{B}\,\overline{D} + C\,\overline{D}$$

ファクタリングは，$\overline{B}$，C あるいは $\overline{D}$ をくくり出す形の 3 通りある．いずれもリテラル総数は “6” である．たとえば，$\overline{D}$ をくくり出して多段最小化した論理式は次の通り.

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

[問題 2.4]　AND/OR 回路を合成してから，NAND 回路にする．回路図は図 4.9.

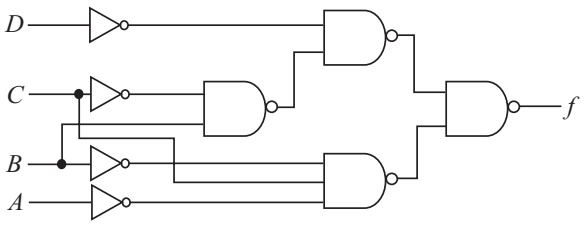

図 4.9　NAND 回路（問題 2.4 の解）

[問題 2.5]　AND/OR 回路を合成してから，NAND 回路に変換すると図 4.10.

[問題 2.6]　最小積和形は f と同じ．ドントケアがある場合は次の通り.

$$\overline{B}\,\overline{C} + A\,D + A\,C$$

公理 2（NOT）　“0” の NOT は “1”，“1” の NOT は “0” である．

$$\overline{0} = 1 \tag{1.1}$$

$$\overline{1} = 0 \tag{1.2}$$

公理 3（AND）　“1” どうしの AND だけが “1” であり，外（ほか）の 3 通りの組み合わせの AND は “0” である．

$$0 \cdot 0 = 0 \tag{1.3}$$

$$0 \cdot 1 = 0 \tag{1.4}$$

$$1 \cdot 0 = 0 \tag{1.5}$$

$$1 \cdot 1 = 1 \tag{1.6}$$

公理 4（OR）　“0” どうしの OR だけが “0” であり，外の 3 通りの組み合わせの OR は “1” である．

$$0 + 0 = 0 \tag{1.7}$$

$$0 + 1 = 1 \tag{1.8}$$

$$1 + 0 = 1 \tag{1.9}$$

$$1 + 1 = 1 \tag{1.10}$$

1.2.2　論理関係を表す数式

論理代数では，論理や論理どうしの関係を数式によって表現する．

[1]　論理関係と論理式

3 種類の論理演算を使うと様々な論理の関係が数式によって表現できる．論理の関係を**論理関係**，論理関係を表す数式を**論理式**，とそれぞれいう．

例　題

1.1　「『A である』かつ『B でない』，または，『C でない』か『D である』」という論理関係を論理式で表しなさい．

［解］

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

論理式は論理変数あるいは論理定数（“0” か “1” の論理値）と 3 種類の論理演算記号を用いて記述できる．論理式中のすべての論理変数に論理値を代入すれば論理式そのものの値（論理値）も一意に決まる．

論理値が等しい論理式は**同値**であり，その関係は算術演算と同様に “=” によって表せる．左辺と右辺が “=” によって結ばれた論理式がある場合,「その論理式は成立している」という．

例 題

1.2 例題 1.1 の論理式が論理値として「“1” をとる」場合，および，論理関係「『A である』かつ『D でない』」と同値でない場合，との 2 つの論理関係を論理式で表しなさい．

［解］

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

$$A \cdot \overline{B} + (\overline{C} + D) \neq A \cdot \overline{D}$$

---

[2] 基本論理演算の演算順位

3 種類の基本論理演算である NOT, AND, OR 間に演算順位を与える．

定義 1.4（基本論理演算順位） 3 種類の基本論理演算の優先順位は，高い方から低い方へ，次の順とする．

$$\text{NOT}(\text{“}\overline{\ }\text{”}) \rightarrow \text{AND}(\text{“}\cdot\text{”}) \rightarrow \text{OR}(\text{“}+\text{”})$$

この暗黙の演算順位を変更したいときは，算術演算と同様にカッコ ( ) によって明示する．最内側のカッコの演算が最優先され，外側へ行くにしたがって優先順位が低くなる．同一カッコ内では定義 1.4 の演算順位にしたがう．

$\overline{X_i}$ を入れ替え，演算記号の $\cdot$（AND）と $+$（OR）を入れ替えて，

$$\overline{L} = f\left(\overline{X_1}, X_1, \overline{X_2}, X_2, \ldots, \overline{X_n}, X_n, +, \cdot\right)$$

と形式的に表せる．このとき，次が成り立つ．

$$\begin{aligned} & f\left(X_1, \overline{X_1}, X_2, \overline{X_2}, \ldots, X_n, \overline{X_n}, \cdot, +\right) \\ & \quad = \overline{f\left(\overline{X_1}, X_1, \overline{X_2}, X_2, \ldots, \overline{X_n}, X_n, +, \cdot\right)} \end{aligned} \tag{1.38}$$

証明　2 変数から始めて，数学的帰納法によって n 変数に拡張する．（省略）

定義 1.6（双対関数）　ド・モルガンの定理（定理 1.16）における n 変数論理関数 f に対して，すべての変数の否定（NOT）をとり，かつ，論理関数全体に NOT を施して得る n 変数論理関数を**双対関数**という．本書では，f の双対関数を f^d と表記する．すなわち，形式的には，

$$f^d = \overline{f\left(\overline{X_1}, X_1, \overline{X_2}, X_2, \ldots, \overline{X_n}, X_n, \cdot, +\right)}$$

と表せる．

ここで，ド・モルガンの定理（定理 1.16）の式 1.38によって，

$$\begin{aligned} f^d &= \overline{f\left(\overline{X_1}, X_1, \overline{X_2}, X_2, \ldots, \overline{X_n}, X_n, \cdot, +\right)} \\ &= f\left(X_1, \overline{X_1}, X_2, \overline{X_2}, \ldots, X_n, \overline{X_n}, +, \cdot\right) \end{aligned}$$

である．これを f と比較すると，「論理関数 f の**双対関数** f^d は，f の AND($\cdot$) と OR($+$) を入れ替えて得る」ことが分かる．

注意　厳密には，双対関数を得る操作では，定義 1.5（11 ページ）による双対な論理式を得る操作と同じく，論理演算記号の AND(“$\cdot$”) と OR(“$+$”) の入れ替えの外に，論理定数の “0” と “1” の入れ替えも適用する．双対を得るこれらの操作では，論理変数の否定はとらない点がド・モルガンの定理と異なる．

定義 1.7（自己双対関数）　n 変数論理関数 f とその双対関数 f^d が同値である（等しい）とき，すなわち，

$$f = f^d$$

ならば，f を**自己双対関数**という．

ま　え　が　き

本書では,「コンピュータサイエンス」を情報科学，情報工学，計算機科学，計算機工学などの総称として用いている．

そして，本書は，大学学部，高等専門学校，専修学校のコンピュータサイエンス系学科における「論理回路」と「論理設計」の教科書として書き下ろした．

その内容は,（社）情報処理学会が策定した「大学の理工系学部情報系学科のためのコンピュータサイエンス教育カリキュラム J97」の U-1 **論理回路**；U-6 **論理設計**；の 2 科目に準拠している．

本書では，コンピュータサイエンスを支える論理代数とそのハードウェアによる実現である論理回路との関係について，電気に関する専門知識がなくても理解できるように，解き明かしている．また，古典的な知識や理論だけではなく，最新の理論や実用的な手法についても平易に解説している．各所で，コンピュータハードウェアの基本原理である「論理回路」を実例として紹介し，理論と実際との関連に興味をつなげるようにしている．

J97 では，この分野の講義を，① **論理回路**：数学的な概念（ソフトウェア）による組み合わせ回路や順序回路といった論理回路（ハードウェア）の実現；② **論理設計**：論理回路の効率の良い設計手法の理論と実際；の 2 科目に分けて学習することを提案している．

本書の構成では，①には，1 章，2 章の 2.1 と 2.2 節，3 章の 3.1，3.2 節が，②には，2 章の 2.3〜2.5 節，3 章の 3.3〜3.5 節，4 章が，それぞれ対応している．また，J97 では，それぞれの講義科目に，1.5 時間× 15 回（週 1 回で半年）をあてることも推奨している．この分類を参考にして，各機関での実現可能性に合わせて，本書を教科書として活用してもらえば幸いである．

本書についてのコメントや意見は著者（shibayam@kit.jp）への E メールで

送られたい．適切なコメント等については，今後の増刷や改訂時に対応する．それらの対応によるあるいは自主的な修正のたびに，その情報を著者の公式ホームページ（http://www.ark.is.kit.ac.jp/~shibayam/）で公開する．

2 章を中心とする「組み合わせ回路とその設計」および 3 章を中心とする「同期式順序回路とその設計」のそれぞれに関する演習問題の補遺については，同じ公式ホームページ（http://www.ark.is.kit.ac.jp/~shibayam/）において，＜学部の講義（シラバス）＞ ・ ＜論理設計＞ → ＜成績評価の方法＞ ・ ＜過去問＞ ・ ＜各年度＞ → とたどる各ページで公開している．

これらの演習問題は全問とも，本書を教科書とする講義の履修者に対して，学期末に，著者が課したオリジナルな定期試験問題である．各年度の問題はすべてが組み合わせ回路（[A]）と同期式順序回路（[B]）の各 1 問をセットとしており，全部で 18 セット（計 36 問）の演習問題を掲載している．各セットの標準的な解答時間はセット（2 問）当たり 90 分である．

この公式ホームページに掲載している演習問題は，組み合わせ回路と同期式順序回路それぞれについて，理論と実際のそれぞれを問う基本問題から発展問題までの順序立てた小問で構成してある．したがって，本書の各章末に収録してある演習問題に比べると，問題それぞれが総合的かつ体系的で完備している．

学習成果の確認のための一助としてチャレンジすることを勧める．

いつもながら，丁寧に査読をしてくれた同僚・平田博章 氏に深謝する．

最後に，我が家の構成員：妻・真木子，4 人の子供たち・風野（ふうや），すみれ，ののみ，蒼宙（あおき）に，「君らの健康が本書執筆の推進剤だよ！ お・お・き・に」．

1999 年 初夏 および 2013 年 初春
京都・松ヶ崎あるいは岩倉にて

柴 山　　潔

下線部の Url をクリックすると，ブラウザで当該ページを開きます．