# コンピューターの仕組み (その 1)

山本昌志 *

2008 年 1 月 18 日

**概 要**

これからコンピューターの仕組みを学習する．ここでは，コンピューターの基本構成と順序回路を学習する．

## 1 本日の学習内容

これから，二回に分けて，コンピューターの仕組みを学習する．まず第一回目は，コンピューターの基本構成と機械語，そして組み合わせ回路について学ぶ．これらの内容については，諸君は一通り学習している．ほとんどの内容は復習になる．

本日の学習のゴールは，以下の通りである．

- コンピューターとチューリング機械の関係が分かり，コンピューターの動作方法が理解できる．
- 機械語とは何か?—ということが分かる．
- 論理演算，ブール代数，組み合わせ回路が分かる．

本日の範囲は，教科書 [1] の pp.154–168 である．

## 2 計算の実現機構

現代のコンピューターは，チューリング機械を半導体を使って実現させたものとなっている．チューリング機械のヘッドが CPU に，テープがメモリーに相当する．ヘッドの中で内部状態を記憶するものがレジスターで，テープから読み出したデータと内部状態から動作を決めるものが CPU の制御装置となる．

コンピューターのメモリーには，コンピューターが取り扱う情報が格納されている．その情報とは，プログラムとデータである．CPU はメモリーからプログラムを読み取り，その内容に応じて，データを書き換える．コンピューターができることは，たったこれだけで，それを高速に行う．1 秒間に数十億回もこの動作を行う．本当に，高速に動作するチューリング機械そのものである．

おもしろいことに，プログラムとデータが同じメモリーに格納される．普通に考えると，データとプログラムは別の場所に記憶させた方が良さそうである [1]．コンピューターが発明されるよりも先に，チューリング機械が考えられたことによるのかもしれない．歴史的経緯はわからないが，同じメモリーにプログラムとデータが格納されているコンピューターをノイマン型と言う．

*独立行政法人　秋田工業高等専門学校　電気情報工学科

[1] ハーバードアーキテクチャーと言われるコンピューターは，データとプログラムは記憶場所が異なる．ただ，この方式のコンピューターは少数．マイクロチップテクノロジー社の PIC はハーバードアーキテクチャー．

### 2.1 コンピューターの基本構成

コンピューターは様々な装置から構成されるが，もっとも基本的な形は CPU(Central Processing Unit) と主記憶装置 (メインメモリー) である．この 2 つがあればコンピューターができあがる．CPU を一つに IC(集積回路) にまとめたものが MPU(Micro Processing Unit) と言う．Intel 社の「Pentium」は，もっとも有名な MPU である．

あとは教科書の通り説明する．

### 2.2 機械語レベルのプログラム例

諸君は，アセンブリ言語「Casl II」を学習済みなので，このあたりことはわかっているはずである．教科書を用いて説明する．

## 3 論理演算と組合せ回路

### 3.1 真理値表と論理関数

教科書の最初に述べている 2 進数の加算については，諸君はよく知っているはずである．また，後の説でも説明するので，ここでは省略する．

論理関数 $F$ は，論理変数 $x_1$, $x_2$, $x_3$, $\cdots$ があるとき，

$$Z = F(x_1, x_2, x_3, \cdots) \tag{1}$$

のようになっているものを言う．これらの論理変数 $(x_1, x_2, x_3, \cdots)$ の取りうる値の集合は，$\{0, 1\}$ である．論理関数の値 $Z$ の値の集合も $\{0, 1\}$ である．こららの論理変数と論理関数の値のすべての組み合わせを書き出したものを真理値表という．論理演算の取り扱う集合は $\{0, 1\}$ と有限なので，真理値表が書ける．普通の関数だとこういうわけにはいかない．とびとびの値を書くことはできるが，それでは関数を完全に表しているとはいえない．諸君は，論理関数と真理値表について，十分知っていると思われるので，ここでは細かい説明はしない．

$n$ 個の論理変数からつくれる論理関数は，いくつ作れるだろうか? 無限に作れる—と思うかもしれないが，それは間違いである．論理変数と論理関数の値は $\{0, 1\}$ の有限の個数から成るので，それらの組み合わせの数は有限である．$n$ 個の論理変数の取りうる値の組み合わせは，$2^n$ である．これら $2^n$ に対して論理関数の値は，それぞれ $0, 1$ の 2 通りあるので，可能な論理関数の数は $2^{2^n}$ となる．

相異なる $2^{2^n}$ 個の論理関数は，たった三つの論理演算 $\{AND, OR, NOT\}$ で作ることができる [2]．この証明には，教科書では数学的帰納法 [3] を使っている．証明は，以下の通り．これは教科書 [1] の証明と全く同じ内容である．

1. 論理変数が一つの場合，その入力の組み合わせの集合 $\{(0), (1)\}$ である．可能な論理関数の値は，表 1 の通りになる．合わせて，論理関数も示す．以上より，論理変数が一つの場合，論理関数は $AND, OR, NOT$

[2] $AND$ は'$\cdot$'，$OR$ は'+',$NOT$ は" と書いても良い．計算を行うときはこの方が簡単．

[3] 有限回の議論で可算無限個の対象に対する命題を証明するための数学の論法. 具体的な手順は，(1) 命題 $P(0)$ は真であることを証明，(2)$P(k)$ が真であれば $P(k+1)$ も真であることを証明 である．これから，すべての $P(k)$ が真であることが言える．

表 1: 1 変数の論理変数と論理関数の組み合わせ

| 論理変数 $x_1$ | 論理関数の値 | 論理関数 |
|---|---|---|
| $\{(0),(1)\}$ | $\{(0),(0)\}$ | $x_1 \cdot \overline{x_1}$ |
| $\{(0),(1)\}$ | $\{(0),(1)\}$ | $x_1$ |
| $\{(0),(1)\}$ | $\{(1),(0)\}$ | $\overline{x_1}$ |
| $\{(0),(1)\}$ | $\{(1),(1)\}$ | $x_1 + \overline{x_1}$ |

で表すことができる．

2. $k$ 個の論理変数がある場合の論理関数 $o_i(x_1, x_2, x_3, \cdots, x_k)$ は，次のように展開できる．

$$
\begin{aligned}
o_i &= \begin{cases} p_i(x_1, x_2, x_3, \cdots, x_{k-1}) & x_k = 0 \text{ のとき} \\ q_i(x_1, x_2, x_3, \cdots, x_{k-1}) & x_k = 1 \text{ のとき} \end{cases} \\
&= [p_i(x_1, x_2, x_3, \cdots, x_{k-1}) \cdot \overline{x_k}] + [q_i(x_1, x_2, x_3, \cdots, x_{k-1}) \cdot x_k]
\end{aligned} \tag{2}
$$

これで，論理変数が $k-1$ 個の場合に成り立てば，$k$ 個の場合にも成り立つことがわかる．

以上より，任意の数の論理変数の場合でも，論理関数は $AND, OR, NOT$ で表現できることが証明できた．従って，演算子 $AND, OR, NOT$ の組み合わせは完備性を有している．どんな論理回路でも，$AND, OR, NOT$ で作ることができる．

## 3.2 ブール代数

ブール代数はこれまでも，繰り返し使ってきたはずである．ここでは，詳細な説明はしない．忘れた者は，付録 A を見よ．

## 3.3 MIL 記法

論理素子 (ゲート素子) を表すための記号として，一般には MIL 記号が使われる．これは，実際の回路の配線を表すことができるので便利である．その MIL 記号とその真理値表を以下に示す．

○印がつくと否定を表すことに注意すること．入力に丸印がついた場合，その否定がゲート入力になる．

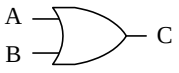


図 1: OR 素子


図 2: AND 素子

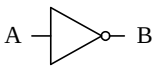


図 3: NOT 素子

表 2: OR の真理値表

| $A$ | $B$ | $A+B$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

表 3: AND の真理値表

| $A$ | $B$ | $A \cdot B$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

表 4: NOT の真理値表

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

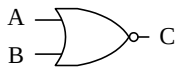


図 4: NOR 素子

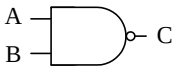


図 5: NAND 素子

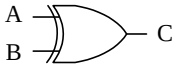


図 6: XOR 素子

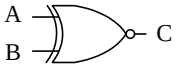


図 7: 一致素子

表 5: NOR の真理値表

| $A$ | $B$ | $\overline{A+B}$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

表 6: NAND の真理値表

| $A$ | $B$ | $\overline{A \cdot B}$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

表 7: XOR の真理値表

| $A$ | $B$ | $A \oplus B$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

表 8: 一致の真理値表

| $A$ | $B$ | $A \odot B$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

### 3.4 組み合わせ回路の例

組み合わせ回路とは，入力からの信号のみから出力が決まる回路のことを言う．一方，入力信号と以前の信号から出力が決まる回路を順序回路といい，フリップフロップ等がある．フリップフロップは次回の学習とし，ここでは簡単に組み合わせ回路について学習する．

いかなる組み合わせ回路でも，AND と OR，NOT から作ることができる．先に証明したように，この組み合わせは完備性を備えている (完全系) からである．このことは，諸君はよく知っている内容である．かなり学習したし，実験も行っている．したがって，ここでは組み合わせ回路について，詳細な説明はしない．教科書の組み合わせ回路に付いては，説明するまでもなく分かるであろう．

# 4　演算回路

## 4.1　加算器

下の桁からの桁上げを考慮しない加算器を半加算器，それを考慮するものを全加算器と言う．これらについても，諸君はすでに学習済みであろう．ここでは，教科書に沿って簡単に復習する．

表 9 が半加算器の真理値表である．この真理値表を実現する論理回路はどうなるか? 真理値表から，論理回路を求める方法はいろいろ有るが，この程度で有れば主加法標準形に直すのが簡単．それぞれは，つぎのようになる．

$$s = \bar{x} \cdot y + x \cdot \bar{y} = x \oplus y \tag{3}$$

$$C_{out} = x \cdot y \tag{4}$$

表 9: 半加算器の真理値表

| $x$ | $y$ | $C_{out}$ | $s$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

表 10: 全加算器の真理値表

| $x$ | $y$ | $C_{in}$ | $C_{out}$ | $s$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |

この半加算器の論理回路は，教科書 [1] の p.164 の図 7.7 の通りである．半加算器を組み合わせて全加算器を作ることができる．教科書 [1] の p.164 の図 7.8 のようになる．

## 4.2　n ビット加算器の高速化

省略．興味のある者は教科書を読め．

## 4.3　減算器

コンピューターでは減算は，負の数の加算として計算する．負の数をどのようにして表現するか?—ということが問題となる．1 の補数と 2 の補数として表すことができる．1 の補数の表現では，負の数の絶対値のビットを反転させる．これは，現在ではあまり使われないので，細かい説明はしない．それに対して，2 の補数はかなり使われるので，以降，詳細に説明する．

### 4.3.1 負の整数の表現

整数が 16 ビットで表現される場合を例にとって，負の数の表現方法を説明する．負の整数は，2 の補数を使って，コンピューター内部では表現する．それを図 8 に示すが，手順は，次の通りである．

1. 絶対値を 2 進数のビットパターンで表現し，その反転を行う．
2. 反転されたビットパターンに 1 を加算する．

このようにしてできたビットパターンをメモリーに記憶させ，それを負の数として取り扱う．

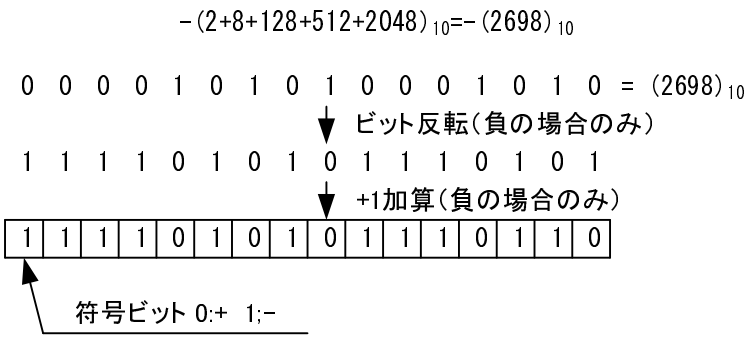


図 8: 負の整数をメモリーに格納する方法

この方法のメリットは，減算が加算器でできることである．補数表現のイメージは，図 9 の通りである．車の距離計に似ている．

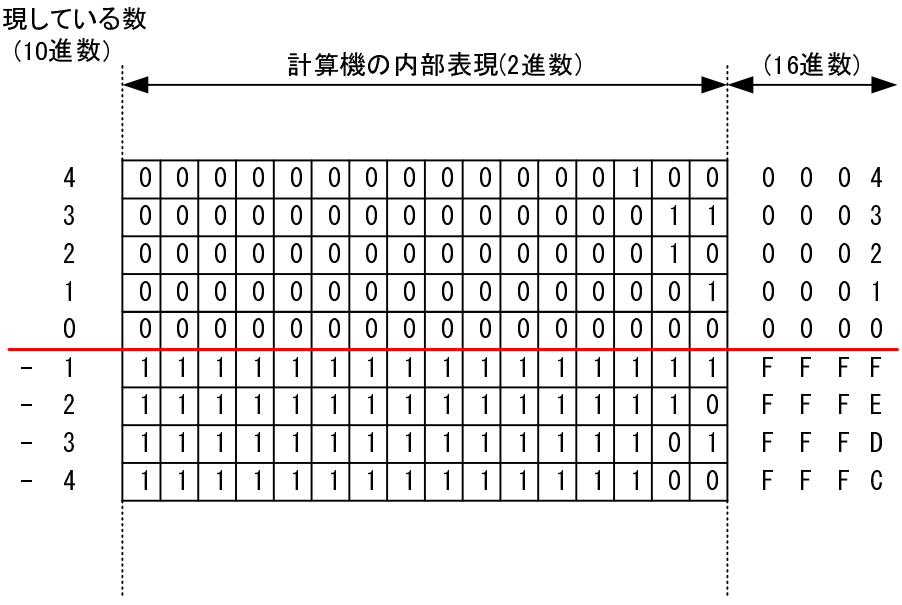


図 9: 距離計イメージ (2 の補数表示)

それでは，なぜ，補数表現だと，減算が加算器で可能なのだろうか? 減算の演算は，負の数の加算と同じである．したがって，図 9 のように負の数を表現すると，負の整数の加算は正の整数の加算と同じと分かるであろう．したがって，加算器で減算が可能となる．実際，正の数の減算を行うときは，ビットの反転と+1 加算を実施して，加算器で計算する．イメージは，図 9 の通りであるが，もう少し，理論的に説明をおこなうとしよう．ある正の整数を $x$ とする．その負の数，$-x$ は補数表現では，

$$[-x] = (FFFF - x + 1)_{16} \tag{5}$$

となる．左辺の $[-x]$ が $-x$ の意味である．[　] の意味は，括弧内の負の整数を計算機内部の表現を表している．これは，私が作った表記なので，一般には用いられていない．右辺の $FFFF - x$ がビット反転になっている．ここでは，16 ビットで整数を表現しようとしているので，$FFFF$ から $x$ を引いてビット反転させている．疑問に思う者は実際に計算して見よ．それに 1 を加えて，補数の表現としている．つぎに，ある整数 $y$ を考えて，$y - x$ を計算してみよう．

$$[y - x] = (y + FFFF - x + 1)_{16} \tag{6}$$

$FFFF - x + 1$ は，あらかじめ計算されて，コンピューター内部のメモリーに格納されているので，$[y - x]$ は加算器で可能である．これは，あたりまえです．重要なことは，この結果が，負の場合，2 の補数表現になっており，正の場合，そのままの値になっていることである．

演算の結果，$y - x$ が負になる場合を考えよう．すると式 (6) は，

$$[y - x] = (FFFF - (x - y) + 1)_{16} \tag{7}$$

と変形できる．この場合，絶対値が $(x - y)$ なので，絶対値のビット反転と+1 加算となっていることが理解できる．つぎに，$y - x$ が正になる場合を考えましょう．すると式 (6) は，

$$\begin{aligned}[y - x] &= (y - x + FFFF + 1)_{16} \\ &= (y - x + 10000)_{16}\end{aligned} \tag{8}$$

となる．10000 は計算機内部では，桁上がりを示す．16 ビットの表示では無視される．したがって，内部の表現は，正しく表せる．

コーヒーブレイク

この方法で負の数を表すことは，1970 年頃には常識となったようである．驚いたことに，負の数をこの補数で表すアイディアは，パスカルが最初です．パスカルは，パスカリーヌという歯車式計算機を 1642 年頃に製作しています．そこの減算を加算器で行うために，補数というものを考えたようです．

#### 4.3.2　ビット反転と+1 加算の意味

$x$ を正の整数として，$-x$ をコンピューターの内部で表現する場合，

1. $x$ をビット反転する．
2. +1 加算する．

の操作で得られたものその内部表現になる．式で表すと，

$$[-x] = (FFFF - x + 1)_{16} \tag{9}$$

である．この操作の意味を調べてみよう．結論から言うと，この操作は符号反転 (-1 乗算) の操作になっている．それを示すために，もう一度この操作を繰り返してみる．すると，

$$\begin{aligned} \{FFFF - (FFFF - x + 1) + 1\}_{16} &= (x)_{16} \\ &= [x] \end{aligned} \tag{10}$$

となる．このことから，この操作は，符号反転であることが理解できる．式 (9) は，$x$ の符号反転を示しており，式 (10) は $-x$ の符号反転を示している．

式 (9) は，-x のコンピューターの内部表現を表している．従って，元の x を求めるためには，その逆の操作

1. 1 減算 (-1 加算) する．
2. ビット反転する．

をすればよく，式だと

$$\begin{aligned} [FFFF - \{(FFFF - x + 1) - 1\}] &= (x)_{16} \\ &= [x] \end{aligned} \tag{11}$$

となる．しかし，元の表現を得るためには，式 (10) の演算でも良いはずである．式 (11) を変形すると，容易に式 (10) を導くことができる．これらのことから，以下の結論を導くことができる．

- ビット反転と+1 加算の操作は，整数のコンピューターの内部での表現の符号反転である．
- この符号反転の反対の操作である 1 減算とビット反転の操作は，ビット反転と 1 加算と同じ操作である．

## 4.4 ALU

CPU で実際に計算を司る部分を ALU(Arithmetic and Logic Unit) と言う．加算と減算を行う ALU は，教科書の p.168 の図 7.12 で可能である．その動作は，下図の通り．

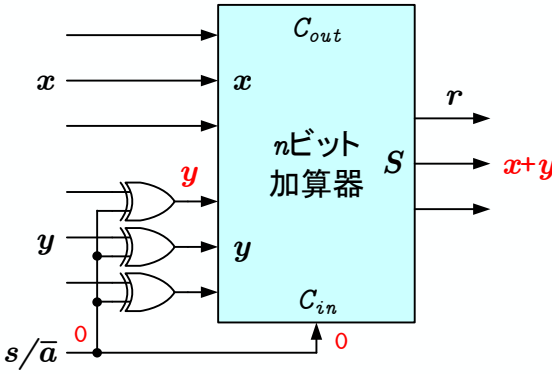


図 10: $s/\bar{a}$ に 0 を入力して ALU を加算器として動作させている．$y$ の入力の排他的論理和 (XOR) の片側に 0 を入力しているので，$y$ が出力される．また，桁上がりの入力は 0 なので，加算となる．

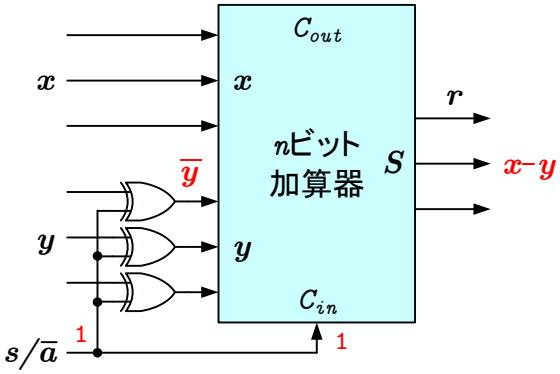


図 11: $s/\bar{a}$ に 1 を入力して ALU を減算器として動作させている．$y$ の入力の排他的論理和 (XOR) の片側に 1 を入力しているので，各ビットが反転した $\bar{y}$ が出力される．さらに，桁上がりの入力は 1 となっている．これは，$-y$ を加算する回路になっている．すなわち，減算である．

# A　ブール代数

## A.1　公理

ブール代数の基本的な演算と変数は，次の通りである．

- 2 項演算子 $+,\cdot$ と単項演算子 $\bar{}$ が定義されている．それぞれ加法と乗法，および否定の演算子である．
- 使われる変数は，0 と 1 のみ．

これらの関係を示す，ブール代数の公理は次の通りである．

**公理 A.1 (ブール代数)**

$$\begin{aligned}
&\text{交換法則} && x+y=y+x, && x\cdot y=y\cdot x && (12)\\
&\text{分配法則} && x\cdot(y+C)=(x\cdot y)+(x\cdot C), && x+(y\cdot C)=(x+y)\cdot(x+C) && (13)\\
&\text{単位元} && x+0=x, && x\cdot 1=x && (14)\\
&\text{否定} && x+\bar{x}=1, && x\cdot\bar{x}=0 && (15)
\end{aligned}$$

これで，ブール代数が定義できる．通常の演算と似ているが，次の点で異なる．

- 式 (13) の 2 つある分配法則のうちの一つが，数の計算の分配法則にはない．
- 補元は逆元に似ていますが，ちょっとことなる．

## A.2　定理

証明はしないが，ブール代数には次のような定理がある．

**定理 A.1 (双対の定理)** ブール代数では，元の式の $+$ と $\cdot$，0 と 1 を交換してできる式を元の式の「双対」(dual) と呼ぶ．ブール代数において，ある定理が成り立つならば，その定理の双対もまた成り立つ．

**定理 A.2 (演算の諸定理)**

$$\begin{aligned}
&\text{結合則} && x+(y+C)=(x+y)+C, && x\cdot(y\cdot C)=(x\cdot y)\cdot C && (16)\\
&\text{吸収則} && x+(x\cdot y)=x, && x\cdot(x+y)=x && (17)\\
&\text{巾等律} && x+x=x, && x\cdot x=x && (18)\\
& && x+1=1, && x\cdot 0=0 && (19)\\
& && x+(\bar{x}+y)=1, && x\cdot(\bar{x}\cdot y)=0 && (20)\\
& && (x+\bar{y})\cdot(\bar{x}+y)=(x\cdot y)+(\bar{x}\cdot\bar{y}), && (x\cdot\bar{y})+(\bar{x}\cdot y)=(x+y)\cdot(\bar{x}+\bar{y}) && (21)\\
&\text{ド・モルガン} && \overline{(x+y)}=\bar{x}\cdot\bar{y}, && \overline{(x\cdot y)}=\bar{x}+\bar{y} && (22)
\end{aligned}$$

**定理 A.3 (二重否定)**

$$\bar{\bar{x}} = x \tag{23}$$

**定理 A.4** $x + \bar{y} = 1$ かつ $x \cdot \bar{y} = 0$ ならば，$x = y$ である．

### A.2.1 真理値表

先に述べたように，ブール代数の変数の集合は $0, 1$ である．そして，演算子は $+$ と $\cdot$, $\bar{}$だ．変数も演算子も少ないので，すべての組み合わせを表にすることができる．それを表 11 から表 13 に示す．このように，変数の全ての組み合わせを示して，その演算結果を示すものを真理値表と呼ぶ．

表 11: $x + y$ の真理値表

| $x$ | $y$ | $x + y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

表 12: $x \cdot y$ の真理値表

| $x$ | $y$ | $x \cdot y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

表 13: $\bar{x}$ の真理値表

| $x$ | $\bar{x}$ |
|---|---|
| 0 | 1 |
| 1 | 0 |

# B 加算標準形と乗算標準形

## B.1 真理値表から論理式を導く

加算標準形と乗算標準形は，真理値表から論理関数 (論理式) を導くときに便利である．これらの標準系について，真理値表を使って説明する．

### B.1.1 加算標準形

3 変数 (入力)$x_1, x_2, x_3$ を含む論理関数で，その出力を $Z$ とする．例えば，表 14 のようなものである．各変数を一つずつ含む式を標準項 (canonical term) と言う．例えば，$x_1 \cdot x_2 \cdot x_3$ や $\bar{x_1} \cdot \bar{x_2} \cdot x_3$，$x_1 + x_2 \cdot x_3$，$x_1 \cdot \bar{x_2} + x_3$，$x_1 + x_2 + x_3$，$x_1 + \bar{x_2} + \bar{x_3}$ などがそれに当たる．このうち最も項が少ないもの，例えば $x_1 \cdot x_2 \cdot x_3$ や $\bar{x_1} \cdot \bar{x_2} \cdot x_3$ を最小項 (minterm) と呼ぶ．項というのは数学で学習したものと同じで，乗算のかたまりを 1 つの項として数える．要するに最小項の標準項が乗算のみで構成されており，その項数は 1 となる．一方，$x_1 + x_2 + x_3$，$x_1 + \bar{x_2} + \bar{x_3}$ のように最も項数の多いものを最大項 (maxterm) と呼ぶ．最大項の項数は，変数の数と同じになる．$x_1 + x_2 \cdot x_3$ や $x_1 \cdot \bar{x_2} + x_3$ は，項数 2 で最小項でも最大項でもない．

準備ができたので，表 14 で示されている真理値表から，論理式を導き出す．まずは，加算標準形で最小項を利用した方法を示す．

表 14 を見よ．入力変数が $x_1 = 1$ の場合 $x_1$，$x_1 = 0$ の場合 $\bar{x_1}$ と表す．$x_2$ や $x_3$ も同様である．そして，それらを乗算して最小項を書き出す．つまり，表 14 の最小項ようにする．入力が 8 種類あるので 8 個の最小項を得ることができる．おのおのの最小項は，それに応じた入力の時のみ 1 になる．例えば，$(x_1, x_2, x_3)$ の入力が $(0, 0, 0)$ の場合，その出力が 1 になる論理関数は，表から分かるように $\bar{x_1} \cdot \bar{x_2} \cdot \bar{x_3}$ のみである．$(1, 0, 1)$ の場合は $x_1 \cdot \bar{x_2} \cdot x_3$ というような具合である．

そうして，今度は出力 $Z$ を注目する．出力 $Z$ のうち，$Z = 1$ となる最小項を次のように加算する．

$$Z = \bar{x_1} \cdot x_2 \cdot x_3 + x_1 \cdot \bar{x_2} \cdot x_3 + x_1 \cdot x_2 \cdot \bar{x_3} + x_1 \cdot x_2 \cdot x_3 \tag{24}$$

これが，元の真理値表を表す論理式になる．式 (24) は 4 つの最小項を含み，それぞれが 1 になる場合を加算しているので，元の真理値表を表すのはあたりまえ．

このように最小項の和で表すことを加算標準形，あるいは主加法標準展開 (principal disjunctive canonical expansion) と言う．もう一度，手順をまとめると以下のようになる．

1. 真理値表に従い，入力が $x_1 = 0$ のとき $\bar{x_1}$，$x_1 = 1$ のとき $x_1$ と表す．$x_2$，$x_3$，$\cdots$ と入力変数の数だけ同じように表す．
2. そうして，それらを乗算して，最小項を作る．これを全ての入力の組み合わせについて行う．
3. 次に，出力 $Z = 1$ の場合の最小項を加算する．これで加算標準形が完成する．

表 14: 3 入力の真理値表と最小項

| 入力 | | | 出力 | 最小項 |
|---|---|---|---|---|
| $x_1$ | $x_2$ | $x_3$ | $Z$ | |
| 0 | 0 | 0 | 0 | $\bar{x_1} \cdot \bar{x_2} \cdot \bar{x_3}$ |
| 0 | 0 | 1 | 0 | $\bar{x_1} \cdot \bar{x_2} \cdot x_3$ |
| 0 | 1 | 0 | 0 | $\bar{x_1} \cdot x_2 \cdot \bar{x_3}$ |
| 0 | 1 | 1 | 1 | $\bar{x_1} \cdot x_2 \cdot x_3$ |
| 1 | 0 | 0 | 0 | $x_1 \cdot \bar{x_2} \cdot \bar{x_3}$ |
| 1 | 0 | 1 | 1 | $x_1 \cdot \bar{x_2} \cdot x_3$ |
| 1 | 1 | 0 | 1 | $x_1 \cdot x_2 \cdot \bar{x_3}$ |
| 1 | 1 | 1 | 1 | $x_1 \cdot x_2 \cdot x_3$ |

### B.1.2 乗算標準形

加算標準形では出力 $Z$ の値が 1 になるものに着目したが，乗算標準形では $Z$ が 0 になるものに着目する．加算標準形とは逆に，入力変数が $x_1 = 1$ の場合 $\bar{x_1}$，$x_1 = 0$ の場合 $x_1$ と表す．$x_2$ や $x_3$ も同様である．そして，それらを加算して最大項を書き出す．つまり，表 15 のようにする．おのおのの最大項は，それに応

じた入力の時のみ 0 になる．例えば，$(x_1, x_2, x_3)$ の入力が $(0, 0, 0)$ の場合，その出力が 0 になる論理関数は，表から分かるように $x_1 + x_2 + x_3$ のみである．$(1, 0, 1)$ の場合は $\bar{x_1} + x_2 + \bar{x_3}$ という具合である．

そうして，今度は出力 $Z$ を注目する．出力 $Z$ のうち，$Z = 0$ となる最大項を次のように乗算する．

$$Z = (x_1 + x_2 + x_3) \cdot (x_1 + x_2 + \bar{x_3}) \cdot (x_1 + \bar{x_2} + x_3) \cdot (\bar{x_1} + x_2 + x_3) \tag{25}$$

これは，元の真理値表を表す論理式になっている．式 (25) は 4 つの最大項を含み，それぞれが 0 になる場合を乗算しているので，元の真理値表を表すのはあたりまえ．

このように最大項の積で表すことを乗算標準形，あるいは主乗法標準展開 (principal conjunctive canonical expansion) と言う．もう一度の手順をまとめると以下のようになる．

1. 真理値表に従い，入力が $x_1 = 0$ のとき $x_1$，$x_1 = 1$ のとき $\bar{x_1}$ と表す．$x_2$，$x_3$，$\cdots$ と入力変数の数だけ同じように表す．
2. そうして，それらを加算して，最大項を作る．これを全ての入力の組み合わせについて行う．
3. 次に，出力 $Z = 0$ の場合の最大項を乗算する．これで乗算標準形が完成する．

表 15: 3 入力の真理値表と最大項

| 入力 | | | 出力 | 最大項 |
|---|---|---|---|---|
| $x_1$ | $x_2$ | $x_3$ | $Z$ | |
| 0 | 0 | 0 | 0 | $x_1 + x_2 + x_3$ |
| 0 | 0 | 1 | 0 | $x_1 + x_2 + \bar{x_3}$ |
| 0 | 1 | 0 | 0 | $x_1 + \bar{x_2} + x_3$ |
| 0 | 1 | 1 | 1 | $x_1 + \bar{x_2} + \bar{x_3}$ |
| 1 | 0 | 0 | 0 | $\bar{x_1} + x_2 + x_3$ |
| 1 | 0 | 1 | 1 | $\bar{x_1} + x_2 + \bar{x_3}$ |
| 1 | 1 | 0 | 1 | $\bar{x_1} + \bar{x_2} + x_3$ |
| 1 | 1 | 1 | 1 | $\bar{x_1} + \bar{x_2} + \bar{x_3}$ |

## 参考文献

[1] 河合慧（編）. 情報. 東京大学出版会, 2006.