ま　え　が　き

本書では，コンピュータの基本である論理回路の豊富で高度な内容を，初めての方でも十分納得できるように，やさしく丁寧に体系的に述べている。できる限り直感的に納得できるように考慮しながらも，同時に論理的であるように心がけて，基礎から高度な内容までスムーズに理解できるように構成している。

本文中で説明すると難解になりそうなところでは，例題や演習問題を解くことによって理解が深まるようにすると共に，その理解度をチェックできるようにしている。論理回路の本質をわかりやすく，直感的にも理論的にも理解できるような教科書となることが本書の重要な目的である。「わかりやすいけれど浅い，高度な内容だけれどわかりにくい」教科書とならないことに気をつけて構成している。

論理回路の基礎的な事柄として，ブール代数，論理関数の表現と簡単化，順序回路の原理について述べ，コンピュータのハードウェアの基礎として，論理素子を組み合わせた組合せ論理回路や，フリップフロップを記憶素子として用いた順序回路の設計法について述べている。特に，一般の教科書では詳しく取り上げられることが少ない主乗法標準形の意味やその簡単化についても記述している。

また，コンピュータにますます高速処理が要求されてきている状況に鑑み，高速演算方式についても一章を設け，近年発展が著しいプログラマブル論理素子についても説明し，アナログ世界とディジタル世界を関連づけるために，アナログ，ディジタル変換の原理についても述べている。最近のディジタル回路（論理回路）はどんどん複雑になり，多くのディジタル回路は“専用の言語”を使って設計されるようになっている。そのため，「ハードウェア記述言語

HDL による論理設計」を付録としてつけた。

本書はディジタル回路である論理回路に重点を置いているので，その基本素子である AND 素子や OR 素子の構成などの電子回路に関する記述を最後の章（9 章）にしている。電子回路に関する知識が先に必要であると思われる方は，9 章をはじめに学習してから 1 章にお進みください。

2013 年 8 月

曽和　将容
範　　公可

目　　次

1 数の表現

2 ブール代数と論理関数

3　論理関数の簡単化

4　いろいろな組合せ論理回路

5 フリップフロップ

6 順 序 回 路

7 アナログ–ディジタル変換

8 高速演算方式

9 基本論理素子の電子回路

付　録　ハードウェア記述言語 HDL による論理設計

COMPUTER SCIENCE TEXTBOOK SERIES

1 数の表現

論理回路では情報を0と1の2文字によって表す。ここでは，0と1とで数値や文字を表す方法について述べる。

1.1 2 進 数

0と1を用いて数値を表現するには2進数を使う。2進数は0110のように表される。2進数の各桁は**ビット**（**bit**）と呼ばれており，8桁を1**バイト**（**byte**）と呼ぶ。2進数の値，例えば101の値は，各桁の値に2のべき乗を掛けることにより

$$1\times2^2+0\times2^1+1\times2^0=4+0+1=5$$

と求められる。

3ビットの2進数では**表1.1**のように0から7まで表すことができる。4ビットでは0から15まで（**表1.2**），8ビットでは0から255，16ビットでは0か

表1.1 2進数

10進数	2進数
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

表 1.2　4 ビット 2 進数と 8 進数，16 進数表示

10 進数	2 進数表示	16 進数表示	8 進数表示
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	8	10
9	1001	9	11
10	1010	A	12
11	1011	B	13
12	1100	C	14
13	1101	D	15
14	1110	E	16
15	1111	F	17

ら 65 535 で表すことができる。

数が大きくなる場合は，10 進数のキロである 1 000 に近い 1 024（2^{10}）が 1 **K**（**ケイ**または**キロ**）として使われる。1 024 K は 1 **M**（**メガ**），1 024 M は 1 **G**（**ギガ**），1 024 G は 1 **T**（**テラ**）と呼ばれる。

2 進数 n ビットで表すことのできる値は 0 から (2^n-1) の 2^n 個である。2 進数の最上位桁 は **MSB**（most significant bit），最下位桁は **LSB**（least significant bit）と呼ばれる。

2 進数は 0 と 1 の羅列で表すので，人間にとっては扱いにくい数値表現である。そのため，表 1.2 に示すように，3 桁または 4 桁をひとかたまりとして扱う方法が考えられている。3 桁では 0 から 7 まで表現できるので **8 進数**（octal number）として，また 4 桁では 0 から 15 まで表現できるので **16 進数**（hexadecimal number）として表現する。

16 進数では 0 から 15 を表す文字が必要である。0 から 9 までは 10 進数の文字そのものが使えるが，10 から 15 を表す文字は 10 進数にはないので新たな文字が必要である。この文字として A，B，C，D，E，F を使う。すなわち，10 を 16 進数で表すのに A を，11 を表すのに B を，12 を表すのに C を，13 を表すのに D を，14 を表すのに E を，15 を表すのに F を用いる。

例えば，2 進数 01010010101 は 16 進数表示では

0101 1010 1111　→　5AF

のように，5AF と表される。8 進数表示では

010 110 101 111　→　2657

のように，2657 と表される。

4 ビットで表すことができる 0 から 15 のうち，0 から 9 までを使って 10 進数を表す方法がある。これを **2 進化 10 進符号**（**BCD コード**：binary coded decimal code）と呼ぶ。例えば，2 進化 10 進符号で 10 進数 365 を表現するのに，2 進数の 3（0011），6（0110），5（0101）をそのまま並べて

0011 0110 0101

と表現する。この値は

$1\times2^9+1\times2^8+1\times2^6+1\times2^5+1\times2^2+1\times2^0=512+256+64+32+4+1=869$

ではなく，10 進数 365 であることに注意が必要である。

1.2　2 進数の負数表現

2 進数による負の値の表現には，**表 1.3**（a），（b）に示すように，**1 の補数表現**（補数表現）と **2 の補数表現**（縮小基数表現）が使われる。

表 1.3　負の数の表現方法

（a）　1 の補数表現

10 進数	2 進数
− 3	100
− 2	101
− 1	110
− 0	111
+ 0	000
+ 1	001
+ 2	010
+ 3	011

0 の表現 2 個

（b）　2 の補数表現

10 進数	2 進数
− 4	100
− 3	101
− 2	110
− 1	111
+ 0	000
+ 1	001
+ 2	010
+ 3	011

負の表現が 1 個増える

0 の表現 1 個

1 の補数表現の各桁は，(基数−1) から正の 2 進数の各桁を引くことによって求められる。2 進数では基数は 2 であるので，(基数−1) は (2−1=1) から 1 である。したがって，**1 の補数表現の各桁は，1 から正の 2 進数の各桁を減じて求める**ということができる。

例えば，011 の最上位桁は (2−1)−0=1，第 2 桁は (2−1)−1=0，第 1 桁

は (2−1)−1=0 となり，1 の補数表現として 100 が求められる。

正の 2 進数表現　　011

1 の補数表現　　100

この例からわかるように，**1 の補数は結果的にもとの 2 進数の 1，0 を入れ替えたものとなる**。

3 桁の 2 進数では 8 個の数値表現が可能であり，表 1.3（a）に示すように，1 の補数では −3 から +3 までを表現することができる。8 個のうち 2 個が +0（2 進数では 000）と −0（2 進数では 111）の表現に使われおり，少し表現効率が悪い。

2 の補数表現は，1 の補数に 1 を加算したものである。表（b）のように零の表現には 000 一つが使われているだけで，表現できる数が一つ増える。そのため 2 の補数表現では，−4 から +3 まで表現することができる。この利点からコンピュータでの計算には 2 の補数表現が使われることが多い。

1 の補数，2 の補数表現では，正数のとき最上位ビットが 0，負数では 1 となるので，このビットは正負の符号を表す**符号ビット**と呼ばれることも多い。

【例題 1.1】

（1） 4 ビットで表現できる数の範囲を，正数表現のみの場合と，1 と 2 の補数表現を用いる場合について書け。

解答　正の整数のみ　　0 から 15

1 の補数表現　　−7 から +7

2 の補数表現　　−8 から +7

（2） 0101 1101，0111 0010 の 2 の補数表現を求めよ。

解答　0101 1101　→　1010 0011

0111 0011　→　1000 1101　◇

2 の補数を使った計算では，減算を負数の加算として行う。例えば，10 進数 3−1=2 の計算は，3 ビットの 2 の補数を使うと

$$\begin{array}{rr} 3 & 011 \\ +)\ -1 & +)\ 111 \\ \hline 2 & 1\boxed{010} \end{array}$$

となって，010（10進数で2）が得られる。結果の4ビット目は，計算が3ビット2進数の計算であるので無視する。

2の補数を使った計算では，その補数が表現できる範囲（**値域**）を超えた結果となる計算はできない。例えば，3ビットの場合，10進数 3+2=5 の計算は

$$\begin{array}{rr} 3 & 011 \\ +)\ 2 & +)\ 010 \\ \hline 5 & 101 \end{array} \quad \rightarrow \quad -3$$

となり，5となるべきところが−3になってしまう。これは5を表現したくても5の表現方法がないからである。このような現象を，計算結果が3ビットの表現値域を超えたということで**オーバフロー**（overflow）という。

したがって，3ビットの場合，2の補数を使う計算では，計算結果が−4〜+3までの間になるようにしなければならない。2の補数を使った計算の結果がオーバフローしているかどうかは，次のようにして確かめることができる。

「**最上位桁からの桁上げと，最上位桁への桁上げが両方あるか両方ないとき，計算はオーバフローしていない。**」

1.3 アスキーコード

人間世界で使っているアルファベットや記号などの文字と，0，1で表されるディジタル世界との情報交換用に**アスキー**（**ASCII**：American Standard Code for Information Interchange）**コード**と呼ばれる米国の情報交換用標準コードが使われてきた。アスキーコードは，**表 1.4** に示すように7ビットでアルファベットや記号を表す。例えば，ronri はアスキーコードでは

r	o	n	r	i
111 0010	110 1111	110 1110	111 0010	110 1001

となる。行替え（LF：0001010）や1字戻り（BS：0001000），ベル（BEL：0000111）などもある。数字1の表現は 011 0001，2の表現は 011 0010 である。

コンピュータでは1バイトが基準となることが多いので，最上位ビットに0

表 1.4　ASCII コード

上位 / 下位				6	0	0	0	0	1	1	1	1
				5	0	0	1	1	0	0	1	1
3	2	1	0	4	0	1	0	1	0	1	0	1
0	0	0	0		NUL	DLE	SP	0	@	P	`	p
0	0	0	1		SOH	DC1	!	1	A	Q	a	q
0	0	1	0		STX	DC2	″	2	B	R	b	r
0	0	1	1		ETX	DC3	#	3	C	S	c	s
0	1	0	0		EOT	DC4	$	4	D	T	d	t
0	1	0	1		ENQ	NAK	%	5	E	U	e	u
0	1	1	0		ACK	SYN	&	6	F	V	f	v
0	1	1	1		BEL	ETB	′	7	G	W	g	w
1	0	0	0		BS	CAN	(	8	H	X	h	x
1	0	0	1		HT	EM	)	9	I	Y	i	y
1	0	1	0		LF	SUB	*	:	J	Z	j	z
1	0	1	1		VT	ESC	+	;	K	[	k	{
1	1	0	0		FF	FS	,	<	L	\	l	|
1	1	0	1		CR	GS	−	=	M	]	m	}
1	1	1	0		SO	RS	.	>	N	^	n	~
1	1	1	1		SI	US	/	?	O	_	o	DEL

を加えて 8 ビットに拡張することも多い。

演　習　問　題

【1】 4 ビットの 1 の補数，2 の補数を書け。

【2】 ASCII コードで kairo を表せ。

【3】 3 ビットの 2 の補数表現を用いて次の計算を行い，結果の正否を確かめよ。
（1） 3+(−1)=　　（2） (−2)+(−1)=　　（3） 2+3=

【4】 8 ビットからなる 2 の補数を使って次の計算を行い，結果の正否を確認せよ。
（1） 124−67=　　（2）124+72=
（3） −15−21=　　（4）127−124=

【5】 2 の補数を使った計算で，「最上位桁への桁上げと最上位桁からの桁上げの両方があるか両方がないとき，計算はオーバフローしていない」ことを説明せよ。

索　　引

―― 著 者 略 歴 ――

曽和　将容（そわ　まさひろ）
1974 年　名古屋大学大学院博士課程修了
（電気・電子工学専攻）
工学博士
1974 年　群馬大学助手
1976 年　群馬大学助教授
1987 年　名古屋工業大学教授
1993 年　電気通信大学教授
2009 年　電気通信大学名誉教授

範　　公可（ファム　コンカ）
1988 年　上智大学理工学部電気・電子工学科卒業
1992 年　上智大学大学院博士後期課程修了
（電気・電子工学専攻）
博士（工学）
1992 年　上智大学助手
1996 年　東京情報大学講師
2000 年　電気通信大学助教授
2007 年　電気通信大学准教授
現在に至る

論　理　回　路

Logical Circuits

Ⓒ Masahiro Sowa, Cong-Kha Pham 2013

2013 年 9 月 20 日　初版第 1 刷発行

検印省略

著　　者　曽　和　将　容
　　範　　公　可
発 行 者　株式会社　コ ロ ナ 社
　　代 表 者　牛 来 真 也
印 刷 所　新日本印刷株式会社

112-0011　東京都文京区千石4-46-10

発行所　株式会社　**コ　ロ　ナ　社**

CORONA PUBLISHING CO., LTD.

Tokyo Japan

振替 00140-8-14844・電話（03）3941-3131（代）

ホームページ　http://www.coronasha.co.jp

ISBN 978-4-339-02705-1　（阿部）（製本：愛千製本所）

Printed in Japan

本書のコピー，スキャン，デジタル化等の無断複製・転載は著作権法上での例外を除き禁じられております。購入者以外の第三者による本書の電子データ化及び電子書籍化は，いかなる場合も認めておりません。

落丁・乱丁本はお取替えいたします