第3章　演算装置

3.1 負の数の表現　：純小数：桁数 m

(1) ２の補数

数 X の絶対値：X^*

X^* の２の補数 $\overline{X^*}$

$X^*+\overline{X^*}=1$

固定小数点
m桁の整数なら、2^m

$\overline{X^*}$ の2の補数は X^*

２進表現の各桁を反転

０を１に，１を０に

最下位桁に１（2^{-m}）を加算

$X^*=.010$ に対して

$\overline{X^*}=.101+.001=.110$

(2)1の補数

X^*の1の補数 $\overline{X^*}$

$X^* + \overline{X^*} = 1 - 2^{-m}$

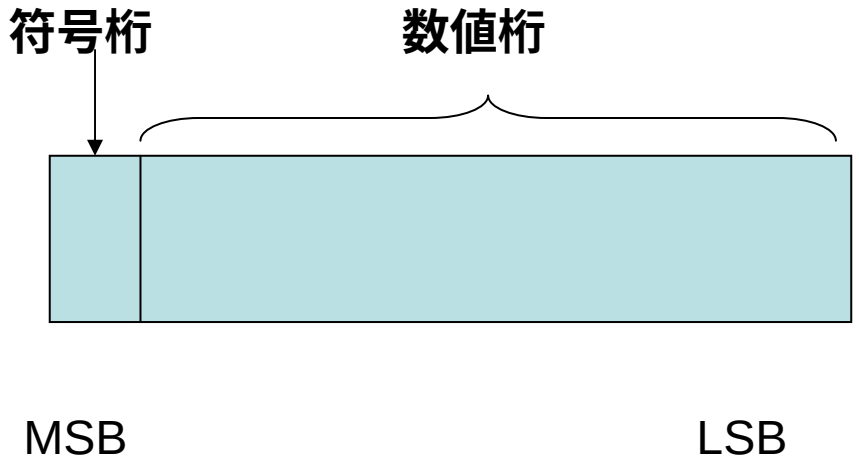
Xの2進表現の各桁反転

$X^* = .101$

$\overline{X^*} = .010$

3.1.2負の数の表現

- X^*を表すには
  - 絶対値表示
  - ２の補数表示
  - １の補数表示



符号桁

０：正,１：負

数値桁

(1)絶対値表示

正のとき　0. $X*$

負のとき　1. $X*$

(2)２の補数表示

正のとき　0. $X*$

負のとき　1. $\overline{X*}$

２の補数表示（0.X^*，1.$\overline{X^*}$）が

与えられたとき，その数表現

の表す数値は，

X^***：正のとき**

-X^***=-(1-**$\overline{X^*}$**)=-1+**$\overline{X^*}$**：負のとき**

1 $\overline{X^*}$ $\longrightarrow$ $\bar{1}$ $\overline{X^*}$**とみなす**

-1の記号

$\overline{X^*}$の２の補数X^*に－を付けた値

$$X^* + \overline{X^*} = 1$$

符号なし数：符号桁を含めて(m+1)桁の数Ｘ’

Unsigned Numbers

２の補数表示

正のとき　0. X^*

負のとき　1. $\overline{X^*}$

符号なし２進数の数値

$X' = X^*$ **正のとき**

$X' = 1 + \overline{X^*}$ **負のとき**

(3) １の補数表示

0.X^*：正のとき

1.$\overline{\overline{X^*}}$負のとき

１の補数表示（0.X^*,1.$\overline{\overline{X^*}}$）が与えられたとき、その数の表す値は

X^*:正のとき

$-X^*=-(1-\overline{\overline{X^*}}-2^{-m})=-1+2^{-m}+\overline{\overline{X^*}}$：負のとき

3．2　シフト

左1ビットシフト　X2　　0110→1100

右1ビットシフト　÷2　　0110→0011

3.2 シフト

3.2.1 数表現とシフト

Ｘが正のとき

左１ビットシフト：$2X^*$

右１ビットシフト：$X^*/2$

Ｘが負のとき

(1)絶対値表示

①符号桁を除いて数値桁のみシフト

②左１ビットシフト：

最下位数値桁に０詰

最上位数値桁からの１のあふれ

オーバフロー

③右１ビットシフト：

最上位数値桁に０詰

(2) ２の補数表示

①符号桁を含めてシフト

②左１ビットシフト：

最下位数値桁に０詰

シフトの結果，符号桁反転

オーバフロー

$1 a_{-1} a_{-2} \cdots a_{-m}$

$a_{-1} a_{-2} \cdots a_{-m} 0$

a_{-1}：０　オーバフロー

③右１ビットシフト：

符号桁には元の値を詰

$1 a_{-1} a_{-2} \cdots a_{-m}$

$1\ 1 a_{-1} a_{-2} \cdots (a_{-m})$

証明

(a)左１ビットシフト＝２Ｘの証明

Ｘは負数（-$X*$）

数値桁$\overline{X*}$

数値桁の最上位桁：０のとき

（−0.5 以下の数のとき）

左１ビットシフト：オーバフロー

最上位桁：１のとき

$\overline{X*}$の左１ビットシフト

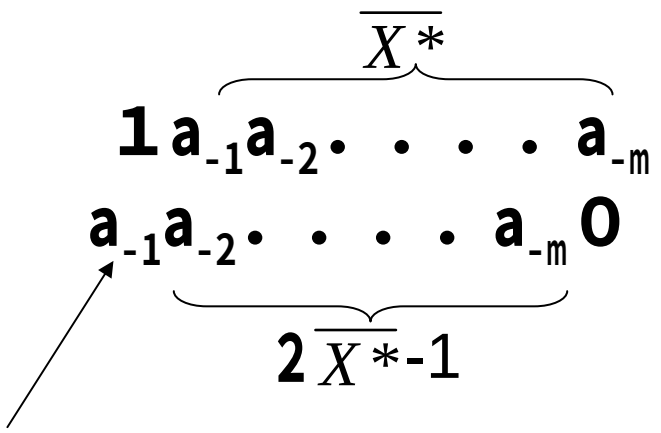
数値桁：2$\overline{X*}$-1

符号桁：１

２の補数表示で数値桁が

2$\overline{X*}$-1 の数Ｚの数値

Z=-1+(2$\overline{X*}$-1)=-2(1-$\overline{X*}$)=-2$X*$

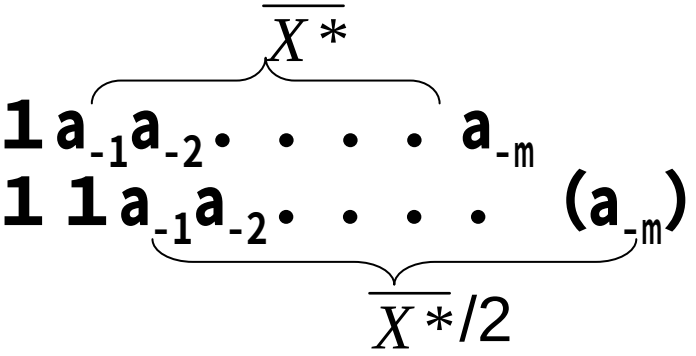


例　1.101(-0.375)

　　1.010(-0.75)

(b)右１ビットシフト：X／２の証明

　右１ビットシフト



　　数値桁：$\overline{X*}$/2+1/2

　　符号桁：１

　　数値桁$\overline{X*}$/2+1/2 をもつ数Ｚの値

　　Z=-1+（$\overline{X*}$/2+1/2）=-$X*$/2

例　1.101　　(-0.375)

　　1.110(1)　(-0.1875)

(3) １の補数表示

左１ビットシフト：

最下位数値桁に１詰

例　1.101(-0.25)

1.010+0.001=1.011(-0.5)

例　1.100　(-0.375)

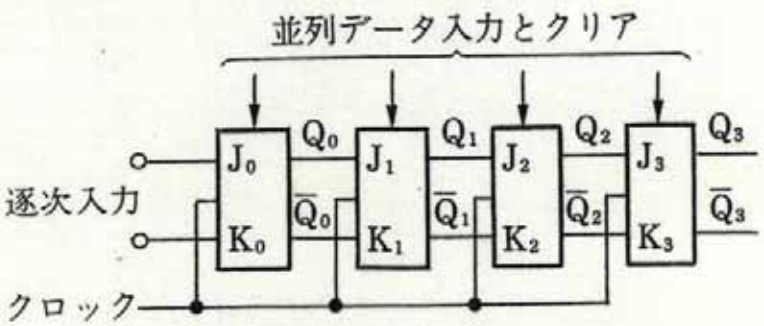
1.110(0)(-0.1875)

1.110　(-0.125)　あふれ

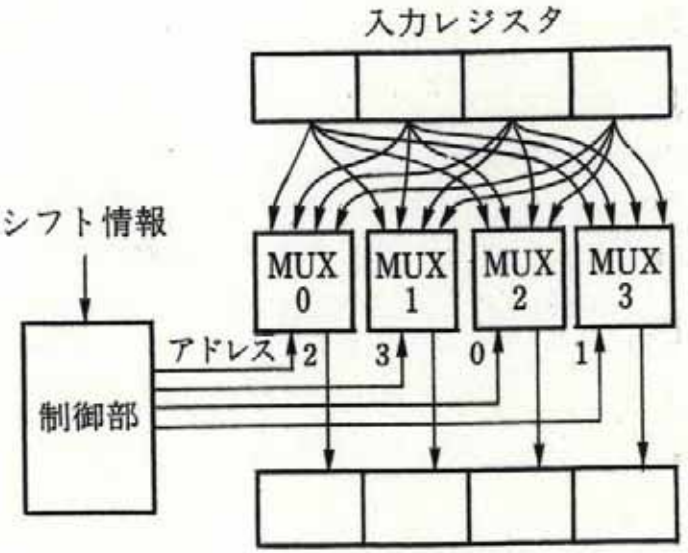
3.2.2 シフト回路

逐次シフト

バレル（並列）シフト

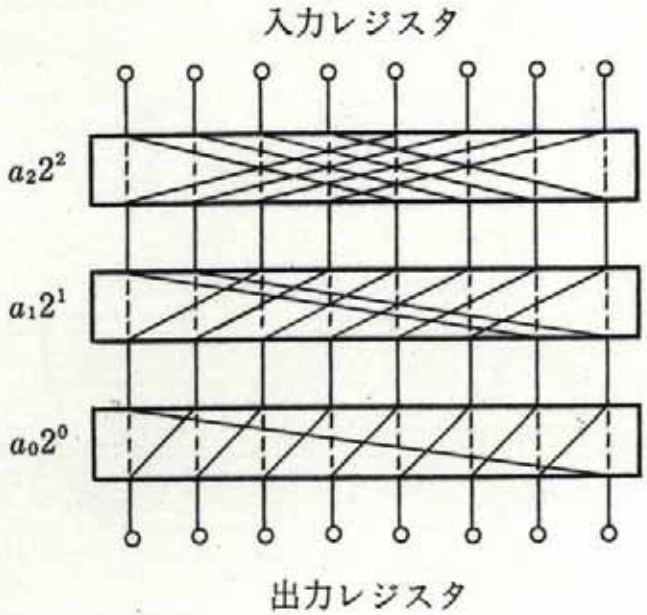


(a) 逐次シフト回路



MUX：マルチプレクサ

(b) バレルシフト回路-1



(c) バレルシフト回路-2

図 2.1 シフト回路

3．3　加減算

3.3 加減算

3.3.1 加算アルゴリズム

$$Z=X+Y$$

(1)絶対値表示

(2) ２の補数表示

・符号桁も含めて(m+1)桁の符号なし２進数の加算を行った結果($X'+Y'$)が正しい結果を与える

・オーバフロー条件は符号桁からのキャリーと符号桁へのキャリーの排他的論理和⊕ が１

証明

①　X>0、Y>0 の時

X'=X^*, Y'=Y^*であるので、

X'+Y'=X^*+Y^*

X'+Y'：正しい答え

オーバフロー：X^*+Y^*が１以上

0　　X^*

0　　Y^*

＜ーキャリなし　＜ーキャリあり

②　X>0、Y<0 のとき

X’=X^*，Y’=1+$\overline{Y^*}$であるので、

X’+Y’=X^*+1+$\overline{Y^*}$

=X^*+1+1-Y^*

=2+X^*-Y^*

$X^* \geqq Y^*$のとき，２は符号桁からの桁上げであるので無視すると X’+Y’の最上位桁は０，下位桁は$X^* - Y^*$

$X^* < Y^*$ のとき，

X’+Y’=1+1-($Y^* - X^*$)

$=1+\overline{Y^* - X^*}$

符号桁：１

数値桁：正の数$Y^* - X^*$ の２の補数

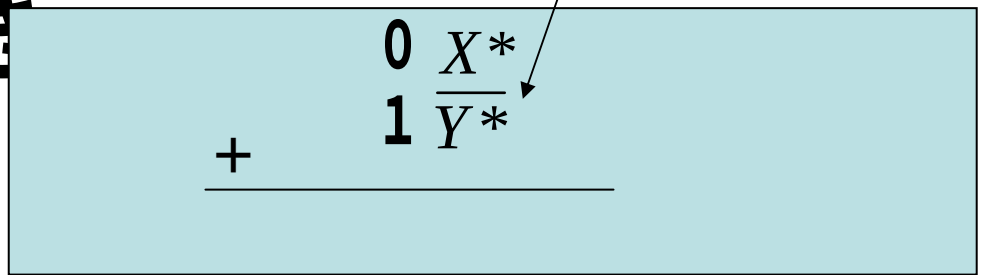
③　X<0、Y>0 のとき

②と同様。

符号桁へのキャリがあって、さらに符号桁からキャリ

符号桁へキャリなし

符号桁そのまま



④　X<0、Y<0 のとき

X'=1+$\overline{X*}$，Y'=1+$\overline{Y*}$

X'+Y'=2+(1-$X*$)+(1-$Y*$)

=2+1+(1-($X*$+$Y*$))

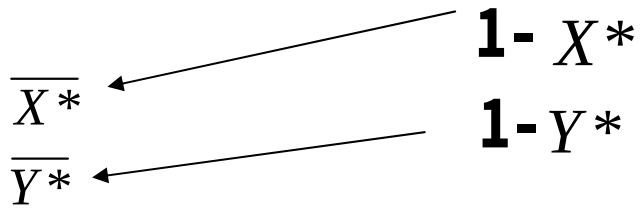
=2+1+$\overline{X*+Y*}$

２は符号桁からの桁上げ：無視

符号桁は１

数値桁：正数($X*$+$Y*$)の２補数

$X*$+$Y*$が１より大：オーバフロー



＜－　キャリ有り＜－キャリなし

例

0.101　　(0.625)

1.101(-0.375)

+1.110(-0.250)　　+

(3) １の補数表示

　１の補数表示では,符号桁を含めた加算（X’+Y’）を行い，符号桁からの桁上げがあると-m 桁に加える（循環桁上げと呼ぶ）ことで加算が実行される.

End Around Carry

① X>0，Y>0 のとき

X’+Y’=$X^{*}+Y^{*}$

② X>0，Y<0 のとき

X’=X^{*}，Y’=1+$\overline{\overline{Y^{*}}}$であるので，

X’+Y’=$X^{*}+1+\overline{\overline{Y^{*}}}$

$=X^{*}+1+(1-2^{-m}-Y^{*})$

$=2+X^{*}-2^{-m}-Y^{*}$

$X^* > Y^*$のとき，符号桁からの桁上げ

があり，これを最下位桁に加えると，
符号桁０，数値桁$X^* - Y^*$

$X^* < Y^*$のとき，

X ’+Y ’=1+1−2^{-m}−($Y^* - X^*$)=1+$\overline{\overline{Y^* - X^*}}$

③　X<0，Y>0
　②と同様
④X<0，Y<0 のとき
　X ’+Y ’=2+$\overline{\overline{X^*}}$+$\overline{\overline{Y^*}}$
　　=2+1−2^{-m}−X^*+1−2^{-m}−Y^*

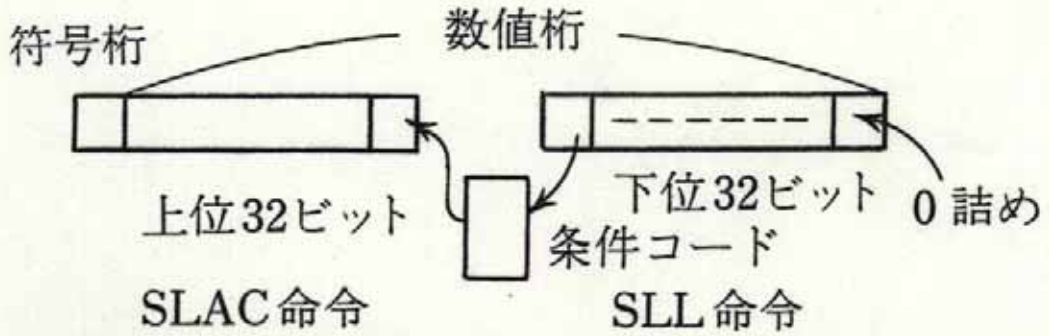
　　=2+1+1−2^{-m}−($X^* + Y^*$)−2^{-m}

２は符号桁からの桁上げであるので，これを最下位桁に加算する
符号桁：１

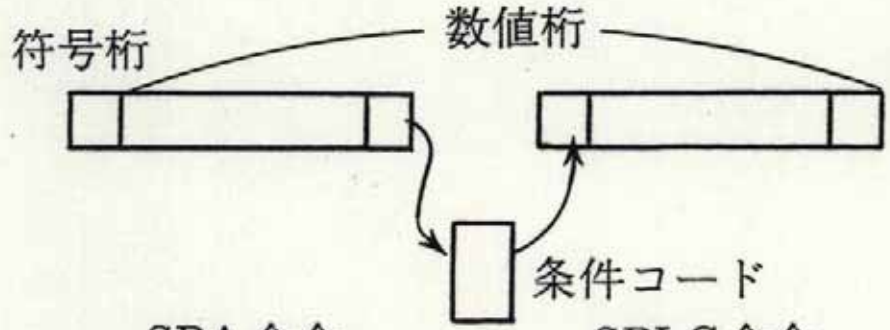
例

0.101 (0.625)	1.101(-0.25)
+1.110(-0.125)	+1.110(-0.125)
10.011	11.011
---->1	---->1
0.100 (0.5)	1.100(-0.375)

多倍長演算



(a) 算術1ビット左シフト



(b) 算術1ビット右シフト

図 3.2 多倍長算術シフト

3.4 高速加算

3.4.1 逐次桁上げ方式（リップルキャリー）

全加算器（FA，full adder）

$S_i = X_i \oplus Y_i \oplus C_{i-1}$

$C_i = X_i Y_i + C_{i-1}(X_i \oplus Y_i)$

$C_i = X_i Y_i + C_{i-1}(X_i + Y_i)$

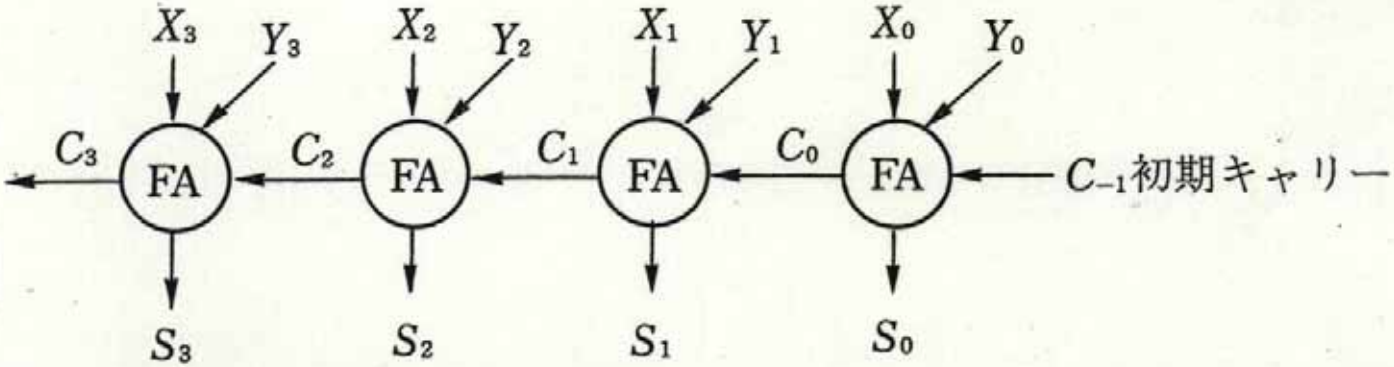
半加算器（HA，half adder）

$S_i = X_i \oplus Y_i$

$C_i = X_i Y_i$

表 3.1　全加算器の真理値表

出力		入力		
C_i	S_i	X_i	Y_i	C_{i-1}
0	0	0	0	0
0	1	0	0	1
0	1	0	1	0
1	0	0	1	1
0	1	1	0	0
1	0	1	0	1
1	0	1	1	0
1	1	1	1	1



FA：全加算器

図 3.3　リップルキャリー方式

キャリー伝搬

1111+0001=10000：桁数に等しい

　X_i=Y_i=1の桁から発生

　$X_j \oplus Y_j$=1なる桁ｊ（ｊ＞ｉ）通過

　$X_k \oplus Y_k$=0なる桁（ｋ＞ｊ）まで

加算時間：O(m)

例（リップルキャリー方式）

```
  011100110
+ 000110010
-----------
  100011000
  ←←← ←←
キャリーの伝搬
```

3.4.2 桁上げ先見（carry look-ahead）

方式

$G_i=X_iY_i$

$T_i=X_i+Y_i$

とおくと，

$C_i=G_i+T_iC_{i-1}$

$=G_i+T_i(G_{i-1}+T_{i-1}C_{i-2})$

$=G_i+T_i(G_{i-1}+T_{i-1}(G_{i-2}+T_{i-2}C_{i-3})$

$=G_i+T_iG_{i-1}+T_iT_{i-1}G_{i-2}+\cdots+$

$T_iT_{i-1}\cdots T_1(G_0+T_0C_0)$

$=G_i+T_iG_{i-1}+T_iT_{i-1}G_{i-2}+\cdots+$

$T_iT_{i-1}\cdots T_1G_0+T_iT_{i-1}\cdots T_0C_{-1}$

C_{-1}：初期キャリー

全加算器

$S_i=X_i\oplus Y_i\oplus C_{i-1}$

$C_i=X_iY_i+(X_i+Y_i)C_{i-1}$

4 ビット先見回路

$$C_0=G_0+T_0C_{-1}$$

$$C_1=G_1+T_1C_0=G_1+T_1(G_0+T_0C_{-1})$$

$$=G_1+T_1G_0+T_1T_0C_{-1}$$

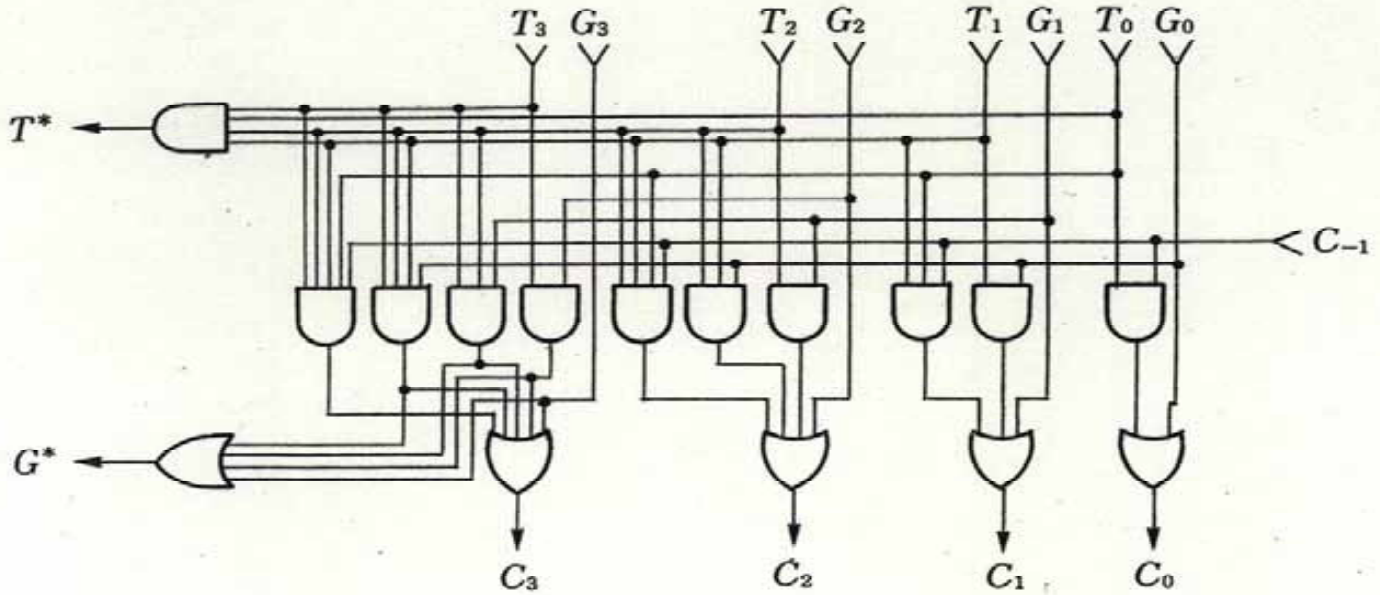
$$C_2=G_2+T_2(G_1+T_1(G_0+T_0C_{-1}))$$

$$=G_2+T_2G_1+T_2T_1G_0+T_2T_1T_0C_{-1}$$

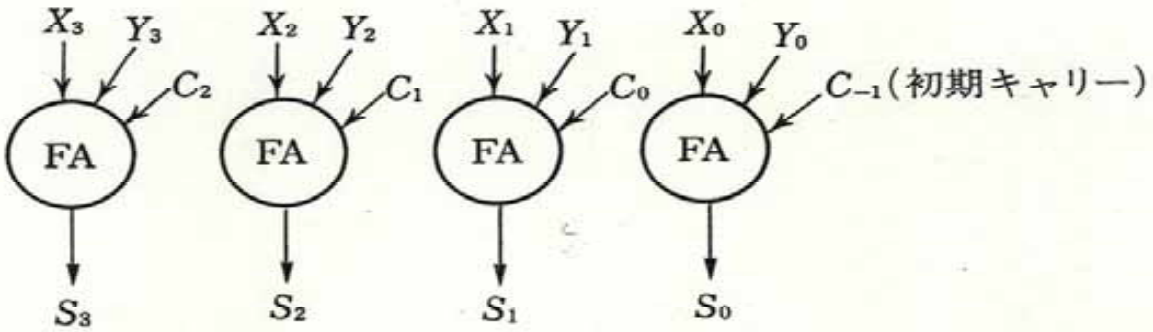
$$C_3=G_3+T_3(G_2+T_2(G_1+T_1(G_0+T_0C_{-1})))$$

$$=G_3+T_3G_2+T_3T_2G_1+T_3T_2T_1G_0+T_3T_2T_1T_0C_{-1}$$

$$=G^*+T^*C_{-1}$$



(a) 桁上げ先見器



(b) 加算器

図 3.4 桁上げ先見器

ファンインとファンアウト

桁上げ先見回路で

まず

$C_3 = G^*_0 + T^*_0 C_{-1}$

$C_7 = G^*_1 + T^*_1 C_3$

$C_{11} = G^*_2 + T^*_2 C_7$

$C_{15} = G^*_3 + T^*_3 C_{11}$

次に $C_{4,5,6,8,9,10,12,13,14}$

計算時間：$O(\log m)$

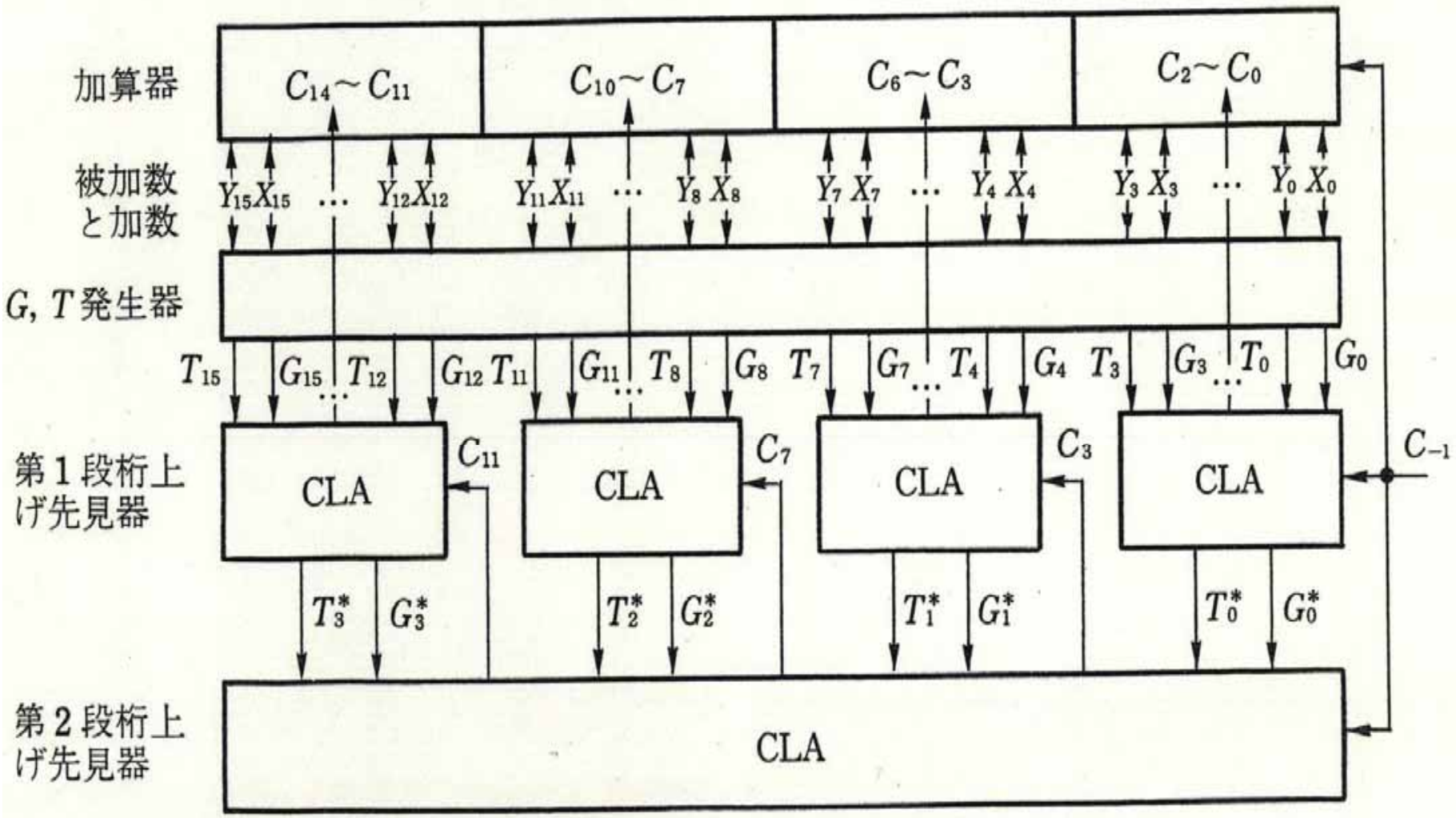


図 3.5　2段桁上げ先見加算器の構成（16ビット）

3.4.3 桁上げ保存方式

全加算器：変換器と見なす

3入力加算→2入力加算

X＋Y＋Z→S＋2C

桁数に無関係に定数時間

表 3.1 全加算器の真理値表

出力		入力		
C_i	S_i	X_i	Y_i	C_{i-1}
0	0	0	0	0
0	1	0	0	1
0	1	0	1	0
1	0	0	1	1
0	1	1	0	0
1	0	1	0	1
1	0	1	1	0
1	1	1	1	1

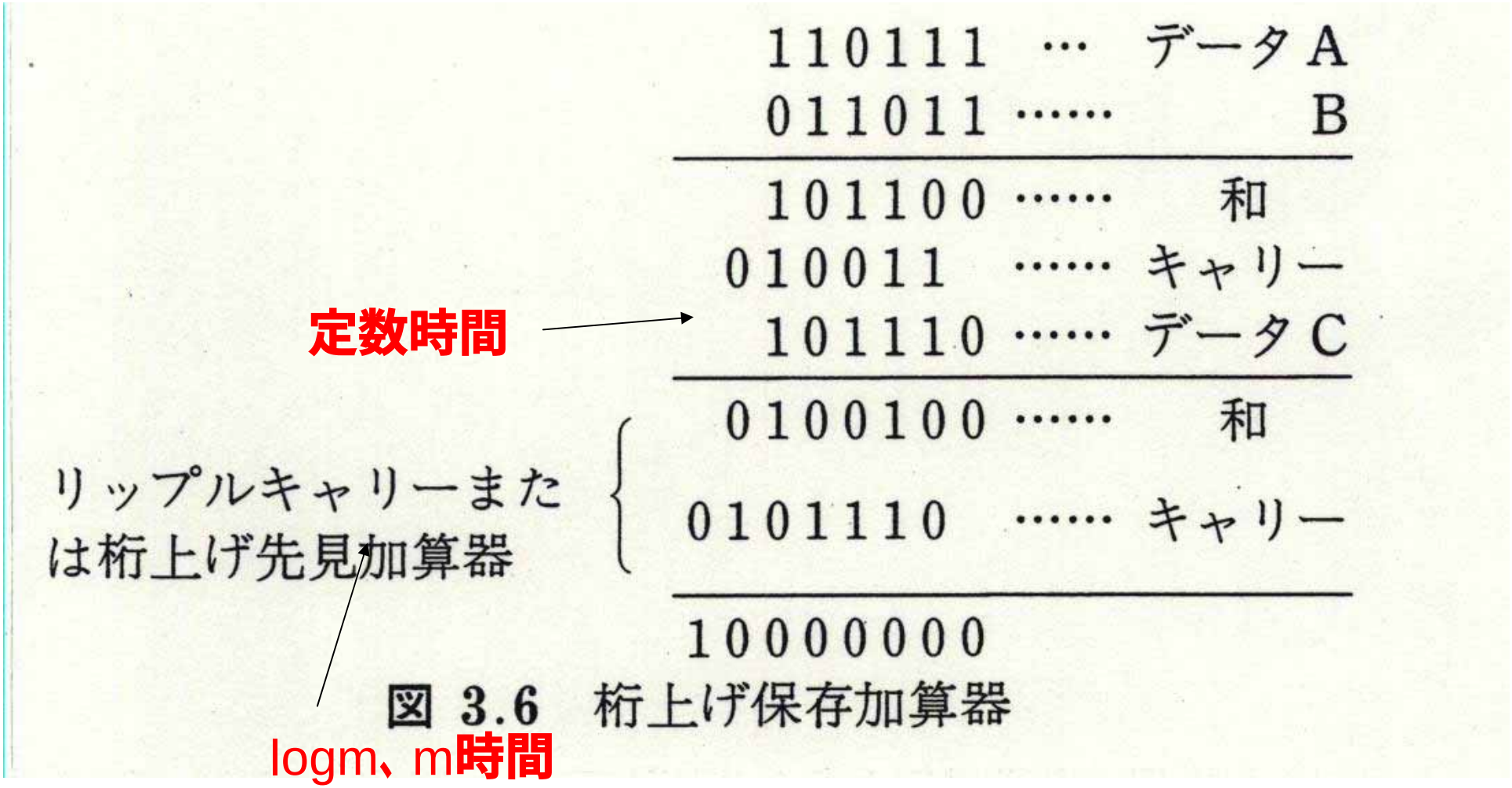


図 3.6 桁上げ保存加算器

3.4.4桁上げ選択法

```
  1010  1100
+0100  0111
```

```
  1010          1100
+0100         +0111
  1110         10011
```

選択

```
  1010
+0101
  1111
```

上位桁　キャリあり、なしの
2つの場合の計算

3.4.5桁上げ完了方式

キャリーの伝播：$X_i Y_i = 1$ で発生
$X_j \oplus Y_j = 1$ を通過
$X_k \oplus Y_k = 0$ で終了

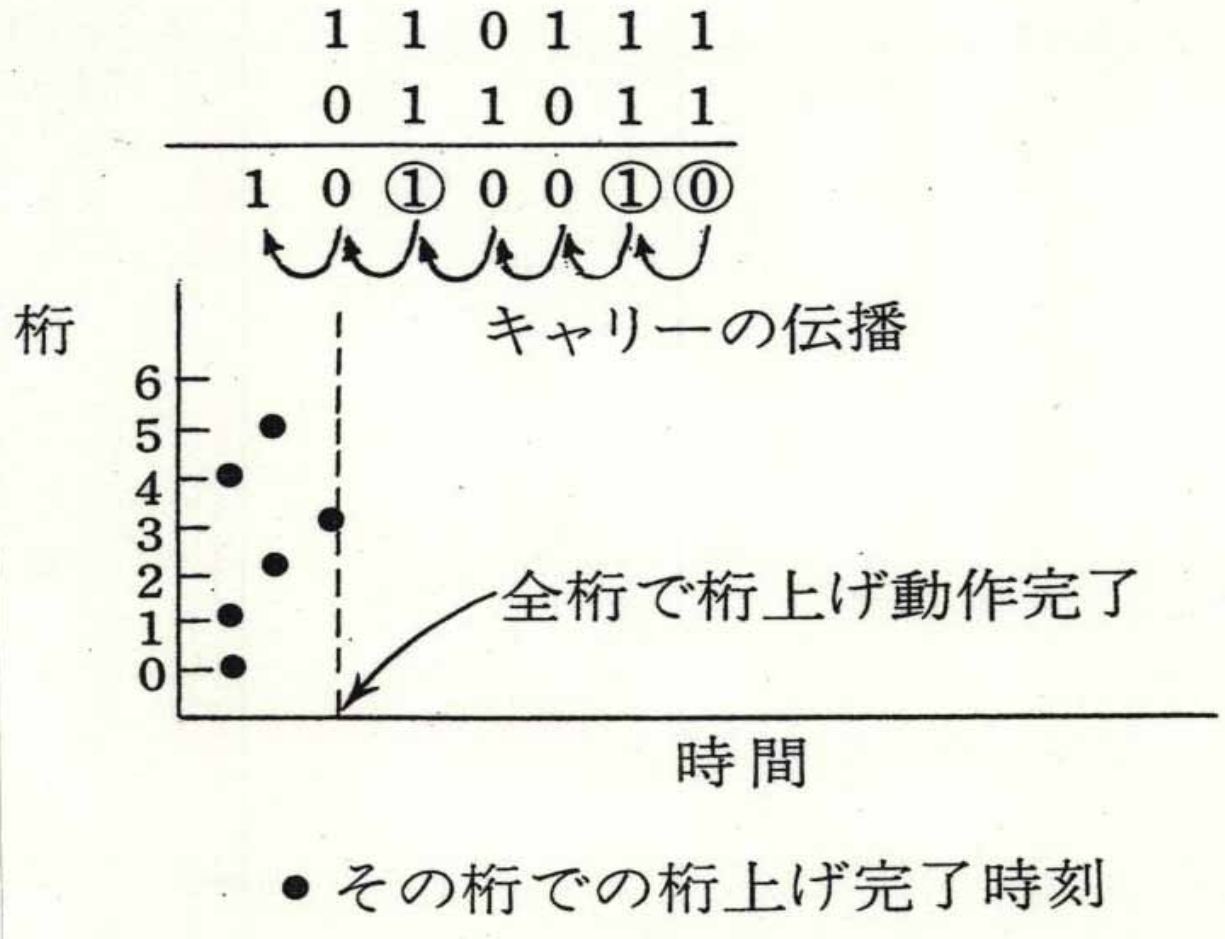


図 3.7　桁上げ完了通知加算器

3.4.6 冗長2進法

０，１，$\bar{1}$

様々な２進数の表現

$0.0101 \quad 0.011\bar{1} \quad 0.1\bar{1}01$

①２進数－冗長２進数変換

２の補数表示：符号１を　$\bar{1}$　とする

-1

表 3.2　冗長2進加算[4)]

タイプ	被加数 (X_I)	加　数 (Y_I)	1つの下位の桁 (X_{I-1}, Y_{I-1})	桁上げ (C_I)	中間和 (S_I)
〈1〉	1	1	—	1	0
〈2〉	1	0	両方とも正*	1	$\bar{1}$
	0	1	少なくとも一方負	0	1
〈3〉	0	0	—	0	0
〈4〉	1	$\bar{1}$			
	$\bar{1}$	1			
〈5〉	0	$\bar{1}$	両方とも正*	0	$\bar{1}$
	$\bar{1}$	0	少なくとも一方負	$\bar{1}$	1
〈6〉	$\bar{1}$	$\bar{1}$	—	$\bar{1}$	0

*：正は1または0を表す.

②冗長２進加算

i-1 桁の１，$\bar{1}$キャリーが i+1 桁に伝播しないようにする

③冗長２進数－２進数変換

例（冗長２進方式）

$$
\begin{array}{rl}
 & 011100110 \\
+ & 000110010 \\
\hline
 & 01\bar{1}0\bar{1}0\bar{1}00 \quad \text{和} \\
 & 011110110 \quad \text{キャリー} \\
\hline
 & 1001\bar{1}1000 \quad \text{キャリー} \\
 & \downarrow \quad \text{(キャリーの伝搬なし)} \\
 & 100101000 \\
- & 000010000 \quad \text{冗長２進－２進変換} \\
\hline
 & 100011000
\end{array}
$$

例（リップルキャリー方式）

011100110

\+ 000110010

100011000

←←←　←←

キャリーの伝搬

2の補数の減算

- 符号なし数と見なして減算
- 符号桁への借り ⊕ 符号桁からの借り＝1
  オーバフロー

⊕

0.110 (0.75)	1.010 (-0.75)
-0.011 (0.375)	-1.100 (-0.5)
0.011 (0.375)	0.011 (0.375)
-1.100 (-0.5)	-1.010 (-0.75)
1.100 (-0.5)	1.100 (-0.5)
-0.010 (0.25)	-0.110 (0.75)

0.110 (0.75)
-0.011 (0.375)
0.011 (0.375)

1.010 (-0.75)
-1.100 (-0.5)
1.110 (-0.25)

0.011 (0.375)
-1.100 (-0.5)
0.111 (0.875)

0.011 (0.375)
-1.010 (-0.75)
1.001 (1.125)

借り、ボロー

1.100 (-0.5)
-0.010 (0.25)
1.010(-0.75)

1.100 (-0.5)
-0.110 (0.75)
0.110 (-1.25)

オーバフロー

符号桁への借り $\rightarrow 0.X^*$

$-1.\overline{Y^*}$

符号桁への借り

$$X' - Y' = 2 + X^* - (1 + \overline{Y^*})$$

$$= 1 + X^* - (1 - Y^*)$$

$$= X^* + Y^*$$

符号桁0のためには符号桁からの借りがあるはず

3．5　乗算

3.5.1 アルゴリズム

(1)絶対値表示

１０進

```
      0.125
    x 0.223
-------------
        375
       250
      250
-------------
   0.027875
```

２進

```
      0.1010
    x 0.1101
-------------
  0．00001010
  0．001010
  0．01010
-------------
  0．10000010
```

高速化の方式

①加算のスキップ

②加算時間の短縮

桁上げ先見法、桁上げ保存法

③多重ビットシフト

例 0.1101x0.1110

1101x10＝11010,1101x11＝100111

```
          0.1101
        x0.1110
  ---------------
  0.00011010   乗数下位２ビット
  0.100111     乗数上位２ビット
  ---------------
  0.10110110
```

④高基数

(2)２の補数表示

①Robertson の方法

Ｙの２の補数表示：

$(y_0 y_{-1},,,y_{-m+1} y_{-m})$

数値：$Y=-y_0+y_{-1}2^{-1}+y_{-2}2^{-2}+\cdots+y_{-m}2^{-m}$

$XY=-xy_0+xy_{-1}2^{-1}+xy_{-2}2^{-2}+\cdots+xy_{-m}2^{-m}$

補正操作

- X>0,Y>0 のとき

  X と y_{-i} との部分積

  　ｉ桁分符号桁０の拡張

- X<0,Y>0 のとき

  X と y_{-i} との部分積

  　ｉ桁分符号桁１の拡張

例

```
      1.1101
    x0.0110
  ---------
  1.1111101      １の拡張に注意
  1.111101
 ----------
  1.11101110
```

- X>0，Y<0 または X<0，Y<0

  補正：Ｘの補数を加算

例

```
      1.1101
    x1.0101
  ---------
  1.11111101
  1.111101
+0.0011---X の補数
 ----------
 10.00100001
```

②Booth 法

$Y=\Sigma_{1}^{m+1}(-y_{-k+1}+y_{-k})2^{-k+1}$

$y_{-(m+1)}=0$

$(y_{-k+1},y_{-k})=(0,1)$：1

$(1,0)$：$\bar{1}$

$(0,0)$：0

$(1,1)$：0

変換（リコード）

$0.1010(0)\rightarrow 1.\bar{1}1\bar{1}0$

$$Y=-y_0+y_{-1}2^{-1}+y_{-2}2^{-2}+\cdots+y_{-m}2^{-m}$$

$$\Downarrow$$

$$\begin{aligned}Y=&-y_0+y_{-1}\\&+(-y_{-1}+y_{-2})\,2^{-1}\\&+(-y_{-2}+y_{-3})\,2^{-2}\\&+(-y_{-3}+y_{-4})\,2^{-3}\\&\quad\vdots\\&+(-y_{-m}+y_{-(m+1)})\,2^{-m}\end{aligned}$$

例 1.0011∗0.1010

1.0011∗ $\bar{1}$ =0.1101

$$
\begin{array}{r}
1.0011 \\
*1.\bar{1}1\bar{1}0 \\
\hline
0.0001101 \\
1.110011 \\
0.01101 \\
1.0011 \\
\hline
11.01111110
\end{array}
$$

乗数：0.11110 のように

　１の連が長いとき

　$1.000\bar{1}$ となり、加算２回

　絶対値の小さな負の数

乗数：0 の連が長いとき

$$
\begin{array}{r}
0.0011 \\
*1.1110 \\
\hline
0.00000000 \\
0.0000011 \\
0.000011 \\
0.00011 \\
1.1101 \\
\hline
1.11111010
\end{array}
$$

$$
\begin{array}{r}
0.0011 \\
*0.00\bar{1}0 \\
\hline
0.00000000 \\
1.1111101 \\
\hline
1.11111010
\end{array}
$$

3.5.2 逐次型乗算器

Booth 法

①Q=E=0 または Q=E=1 のとき，
　ACC を算術右シフト

②Q=1,E=0 のとき,X の補数を
　求め,ACC に加算後,算術右シフト

③Q=0,E=1 のとき,X を ACC に
　加算後,算術右シフト

②Booth 法

$Y=\Sigma_1^{m+1}(-y_{-k+1}+y_{-k})2^{-k+1}$

$y_{-(m+1)}=0$

$(y_{-k+1},y_{-k})=(0,1)$: 1

$(1,0)$: $\bar{1}$

$(0,0)$: 0

$(1,1)$: 0

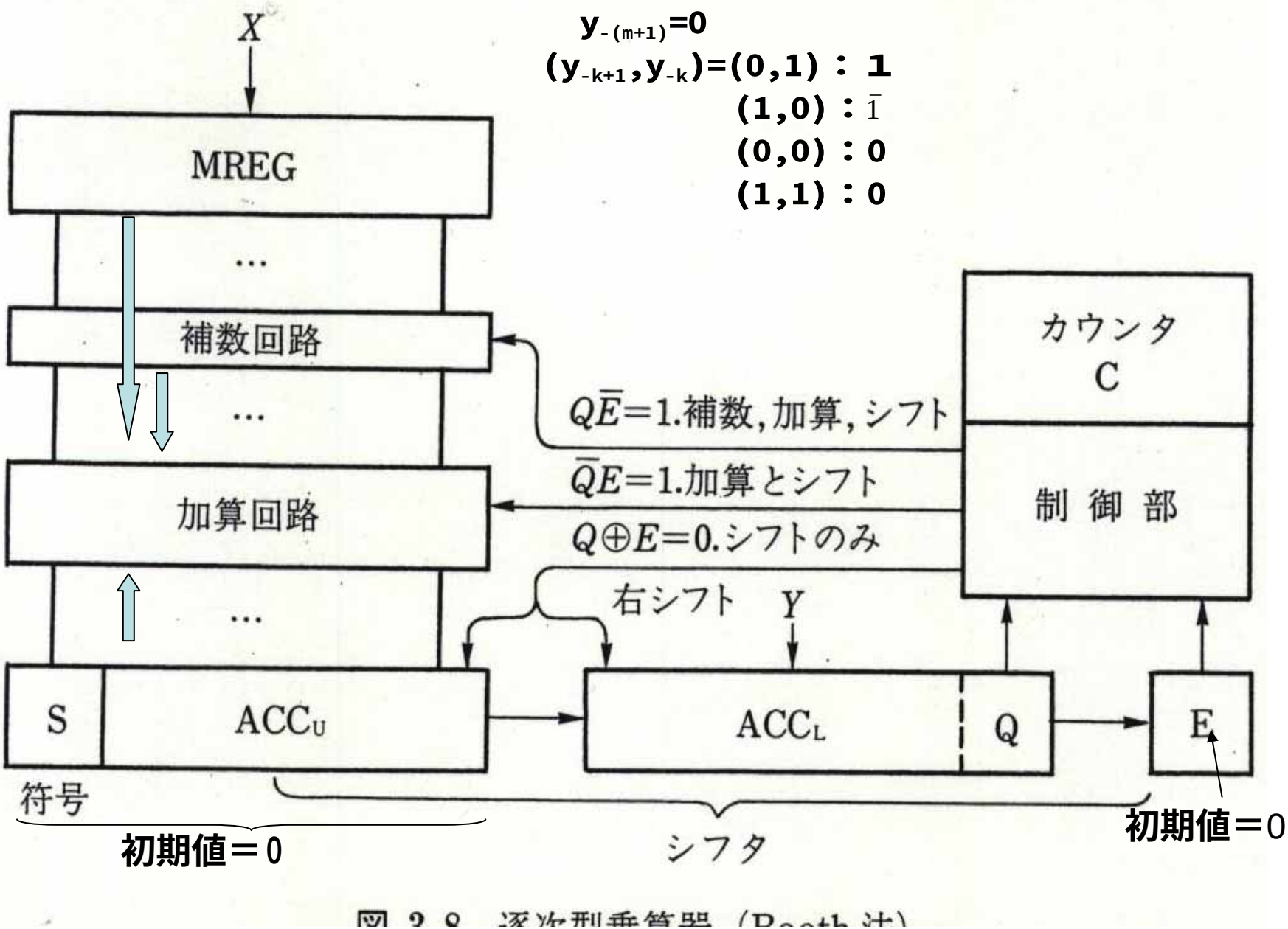


図 2.8 逐次型乗算器（Booth 法）

3.5.3 並列型乗算器

①シフト操作をなくした高速な乗算器

②逐次型乗算器の制御部なし

③多段配置の加算器群にはフィードバックがなく,パイプライン構成可能

(1)配列乗算器

桁上げ保存加算器

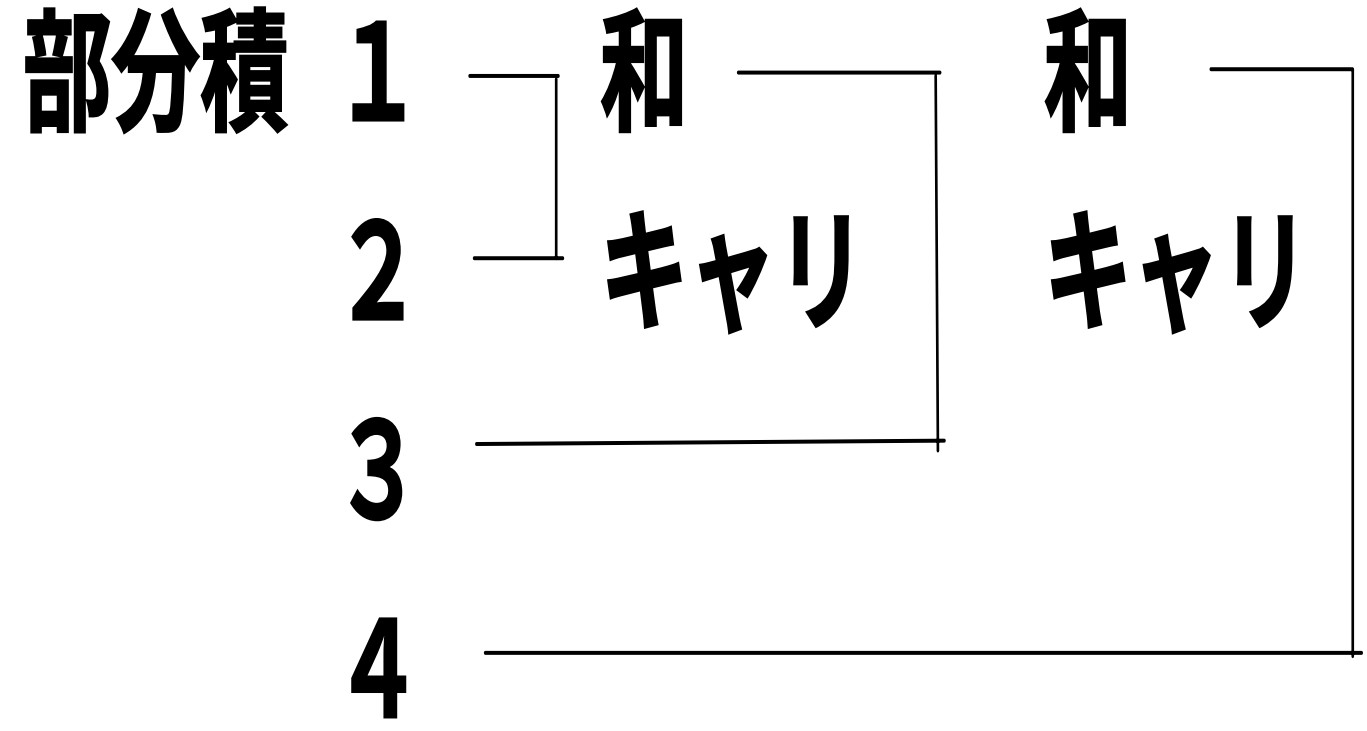
桁上げ先見加算器（最終段）

FA の数：m(m-1)

加算器の段数：m

乗算時間：O(m)

VLSI レイアウト：規則構造



桁上げ保存　最後桁上げ先見

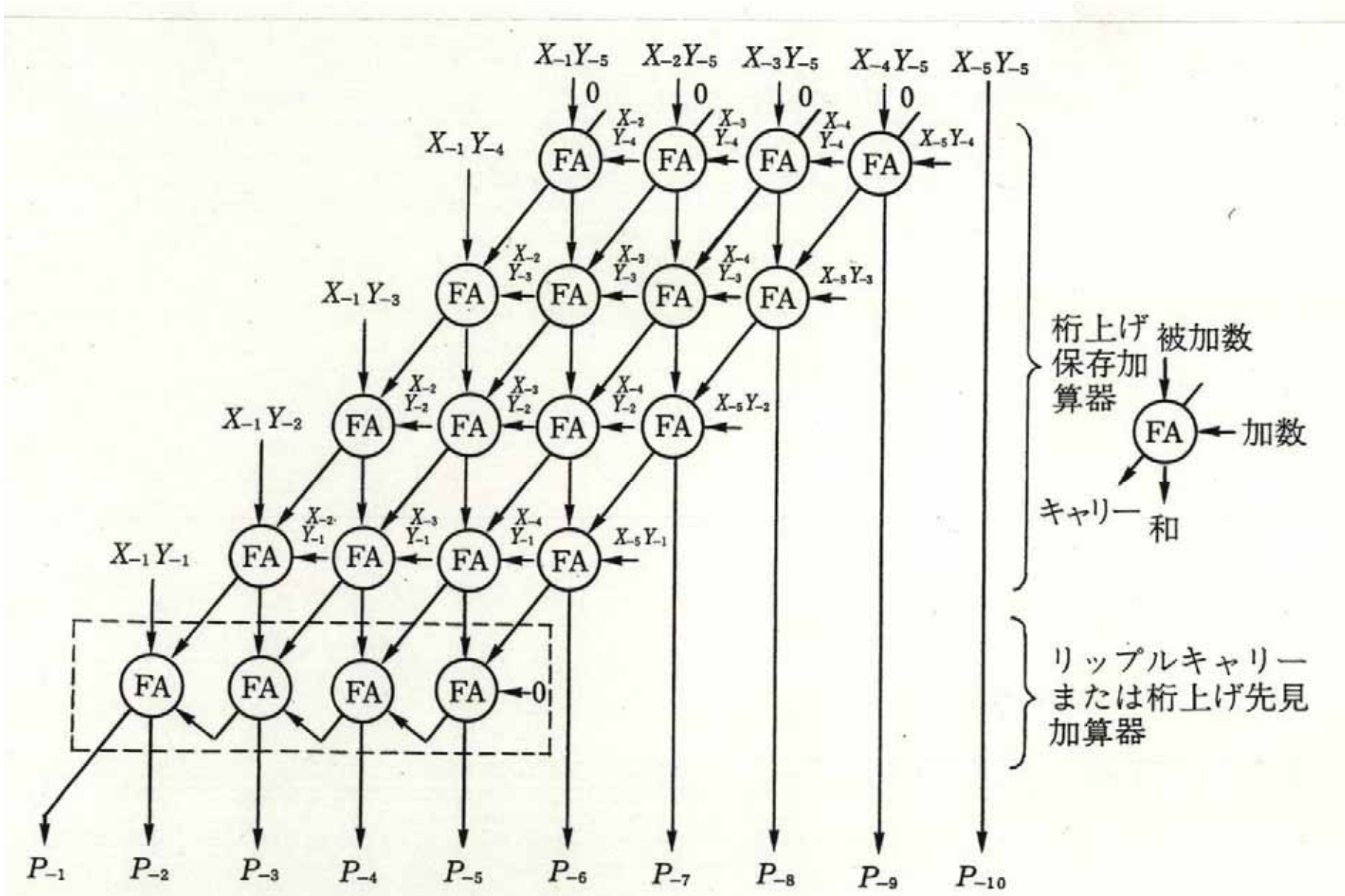

$X_{-1}Y_{-5}$ $X_{-2}Y_{-5}$ $X_{-3}Y_{-5}$ $X_{-4}Y_{-5}$ $X_{-5}Y_{-5}$
0
$X_{-1}Y_{-4}$
$X_{-5}Y_{-4}$
$X_{-1}Y_{-3}$
$X_{-5}Y_{-3}$
$X_{-1}Y_{-2}$
$X_{-5}Y_{-2}$
$X_{-1}Y_{-1}$
$X_{-5}Y_{-1}$
FA
桁上げ保存加算器
被加数
加数
キャリー
和
リップルキャリーまたは桁上げ先見加算器
P_{-1} P_{-2} P_{-3} P_{-4} P_{-5} P_{-6} P_{-7} P_{-8} P_{-9} P_{-10}

図 3.9 配列乗算器

(2)Wallace トリー乗算器

部分積を３つずつのグループ分割

グループ内で FA で加算

　３変数加算―＞２変数加算

結果を更に３変数づつグループ化

最終段の２変数：桁上げ先見加算

加算すべき部分積個数 m

$m->(2/3)m->(2/3)^2m->...->(2/m)m$

レベル数：L

$2/m=(2/3)^L$

$L=(\log m-1)/(\log 3-1)$

$=1.71(\log m-1)$

計算時間:$O(\log m)$

加算器の数:m^2

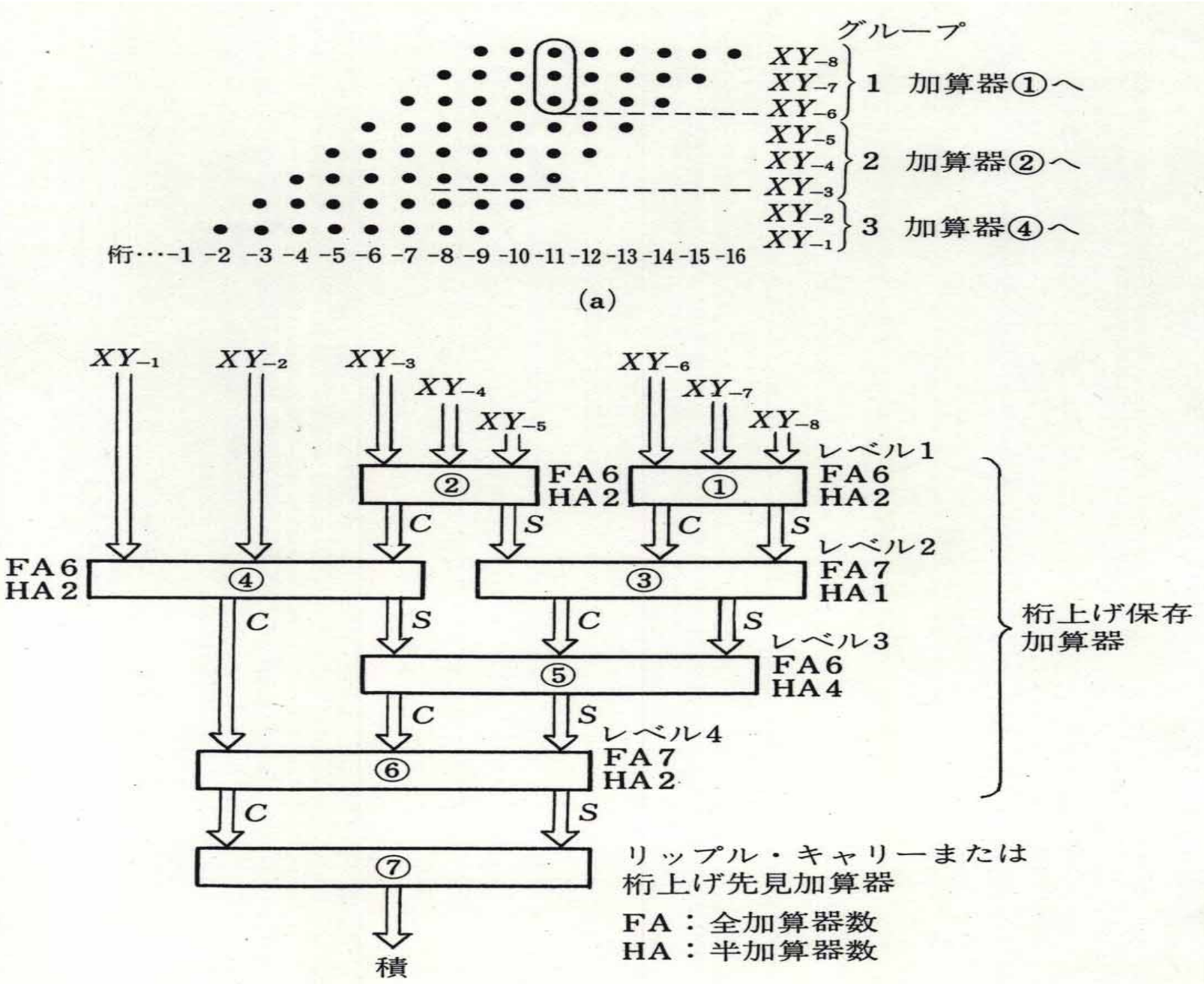


図 3.10 Wallace トリー乗算器

(3) ＲＯＭ乗算器

m*m 桁乗算

2m ビットのアドレス線のメモリ

関数表

メモリ容量

m=16 のとき，$32x2^{32}$ ビット

m=8 のとき，$16x2^{16}$ ビット

8*8 乗算

X_L と Y_L，X_H と Y_L，

X_L と Y_H，X_H と Y_H

メモリ・アクセス４回

加算３回，シフト３回

$XY=\{(X+Y)^2-(X-Y)^2\}/4$ の利用

加減算３回，シフト１回，

メモリアクセス２回

m=16 の場合の ROM 容量

$32x2^{16}$ ビット

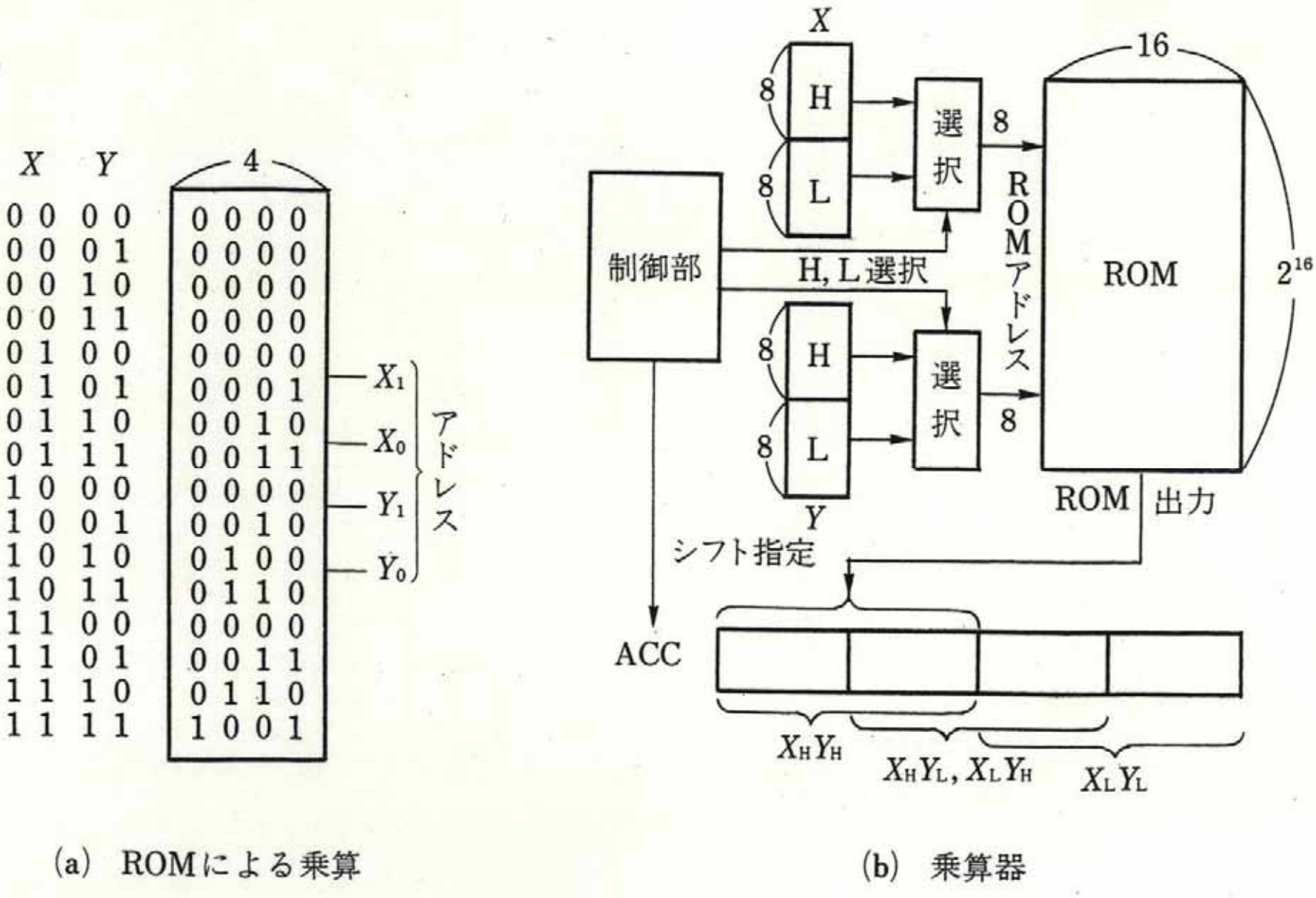

X
Y
4
0 0 0 0
0 0 0 1
0 0 1 0
0 0 1 1
0 1 0 0
0 1 0 1
0 1 1 0
0 1 1 1
1 0 0 0
1 0 0 1
1 0 1 0
1 0 1 1
1 1 0 0
1 1 0 1
1 1 1 0
1 1 1 1
X_1
X_0
Y_1
Y_0
アドレス
(a) ROMによる乗算
X
16
8
H
8
L
選択
8
ROMアドレス
制御部
H, L選択
ROM
2^{16}
8
H
8
L
選択
8
Y
ROM
出力
シフト指定
ACC
$X_H Y_H$
$X_H Y_L, X_L Y_H$
$X_L Y_L$
(b) 乗算器

図 3.11 ROM 乗算器

(4)冗長２進乗算器

m個の部分積の総和

２進木加算網で実現

加算の木の段数：logm

計算時間：O(logm)

レイアウト：規則的

２の補数表示乗算

被乗数、乗数の符号桁１のとき，$\overline{1}$

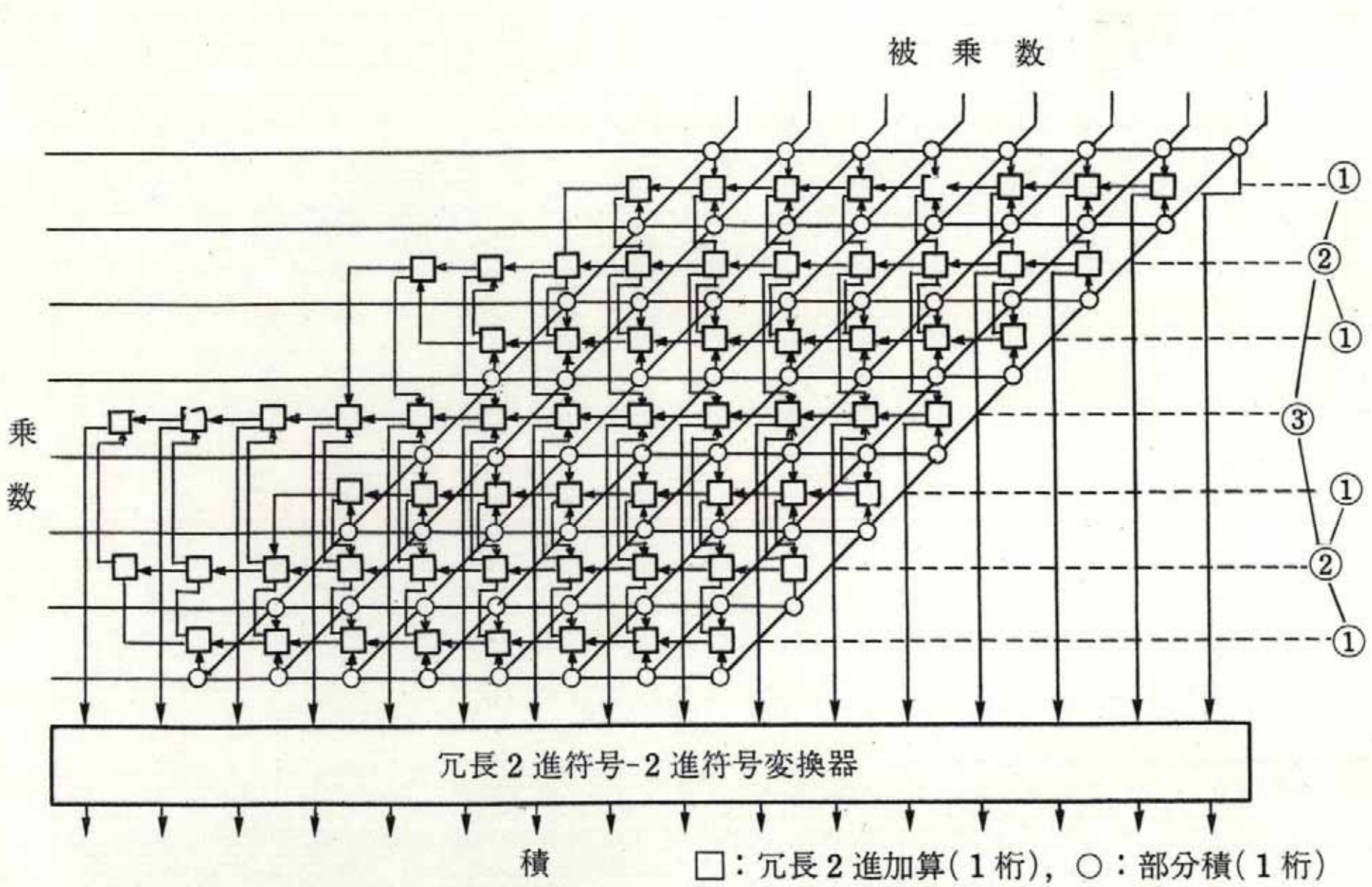


図 3.12　冗長2進加算木による高速乗算器のハードウェア構成[高木直史, 安浦寛人, 矢島脩三：冗長2進加算木を用いたVLSI向き高速乗算器, 電子通信学会論文誌, J 66-D, 6, pp. 683-690 (1983)]

3.6 除算

引戻し法，引放し法，

SRT 法，反復法

3.6.1 引戻し法（restoring　method）

通常の正の１０進数の除算

0≦部分剰余＜除数

例　0.358538÷0.556

商　除数 部分剰余

整数桁　0.358538=0x0.556+0.358538

x10

小数１桁　3.58538= 6x0.556+0.24938

x 10

小数２桁　2.4938 =4x0.556+0.2698

x10

小数３桁　2.698　=4ｘ0.556＋0.474

商:0.644,剰余:0.000474

2進引戻し法

被除数X：小数2m桁

除数Y：　小数m桁

商Q：　　小数m桁

$X=QY+R$

商の最下位桁：2^{-m}であるので、

剰余Rは

$R<Y2^{-m}$

X≧Y：オーバフロー

以後X<Y

初期部分剰余 R_0

$R_0=X$

次に，R_0 を２倍（左シフト）して，Ｙを引き R_1 とする．これが正または０ならば，Q_1＝１とする．負ならば Q_1＝０とし，Ｙを R_1 に加算して返す（restore）．この結果

$R_1=2R_0-Q_1Y$

以下同様

$2R_{i-1}-Y \geqq 0$ なら $Q_i=1$

$2R_{i-1}-Y<0$ なら　$Q_i=0$

として

$R_i=2R_{i-1}-Q_iY$

$R_0 = X$

$R_1 = 2R_0 - Q_1 Y \quad X2^{-1}$

$R_2 = 2R_1 - Q_2 Y \quad X2^{-2}$

・・・

$R_i = 2R_{i-1} - Q_i Y \quad X2^{-i}$

・・・

$R_m = 2R_{m-1} - Q_m Y \quad X2^{-m}$

$X = \{ \Sigma_1^m Q_i 2^{-i} \} Y + R_m 2^{-m}$

商$(0.Q_1Q_2 \cdots Q_m)$

剰余 $R_m 2^{-m}$

$Y2^{-m}$ より小さいことの証明

$0<R_{i-1}<Y$ と仮定すると，

$2R_{i-1}>Y$ のとき、

$R_i=2R_{i-1}-Y=R_{i-1}-Y+R_{i-1}<R_{i-1}<Y$

$2R_{i-1}<Y$ のとき

$R_i=2R_{i-1}<Y$

$R_0=X<Y$ であるので、帰納法によって

$R_m<Y$．したがって $R=R_m2^{-m}<Y2^{-m}$

例（引戻し法．絶対値表示）

X=0.1011000000

Y=0.11101

R_0=X　　=0.10110

R_1=2R_0-Y=0.01111　　Q_1=1

R_2=2R_1-Y=0.00001　　Q_2=1

R_3=2R_2　=0.00010　　Q_3=0

R_4=2R_3　=0.00100　　Q_4=0

R_5=2R_4　=0.01000　　Q_5=0

Q=0.11000、R=$R_5 2^{-5}$=0.0000001

3.6.2 引 放 し 法 （ non-restoring method）

１０進法の引き放し法

商の桁(-9,-8,･･･-1,1,･･･,8,9)

商 除数 部分剰余

整数桁 0.358538= 0x0.556+0.358538

x10

小数１桁 3.58538 = 7x0.556-0.30662

x10

小数２桁-3.0662 =-5x0.556-0.2862

x10

小数３桁 –2.862 =-6x0.556+0.474

小数１桁で商:６，７の選択の余地

７を選択したため,部分剰余が負

小数２，３桁で商に負数をたて補正

商:0.7-0.05-0.006=0.644

部分剰余の絶対値≦除数の絶対値

２進引き放し法

商の桁$\bar{1}$または１

２の補数表示に適した方法

$|X|<|Y|$

初期部分剰余

$R_0=X$として、

$R_i=2R_{i-1}-Q_iY$

R_{i-1}とＹが同符号のときQ_i＝１

異符号のとき$Q_i=\bar{1}$

$$R_0 = X$$

$$R_1 = 2R_0 - Q_1 Y$$

$$\bullet \quad \bullet \quad \bullet$$

$$R_i = 2R_{i-1} - Q_i Y$$

$$\bullet \quad \bullet \quad \bullet$$

$$R_m = 2R_{m-1} - Q_m Y$$

$$X = \{ \Sigma_1^m Q_i 2^{-i} \} Y + R_m 2^{-m}$$

剰余｜２$^{-m}$Rm｜＜｜２$^{-m}$Y｜の証明

$|R_{i-1}|<|Y|$とすると

$R_i=2R_{i-1}\pm Y$

（+Y は R_{i-1} と Y が異符号，-Y は同符号）

より，

$$R_i^2-Y^2=4R_{i-1}^2\pm 4R_{i-1}Y$$
$$=4R_{i-1}^2(1\pm Y/R_{i-1})<0$$

したがって，$|R_i|<|Y|$

また，$|R_0|<|Y|$．帰納法により，$|R_m|<|Y|$

$1,\bar{1}$ **から１，０表現への変換**

$$0.\bar{1}\bar{1}\bar{1}1 \rightarrow \bar{1}.1\bar{1}\bar{1}1. \rightarrow \bar{1}.01\bar{1}1 \rightarrow \bar{1}.0011$$

２の補数表現で１．００１１

例（引放し法１，$\bar{1}$表現）

X=0.10000000,Y=1.0110

R_0=X　　=0.1000

$R_1=2R_0+Y$=0.0110　R0 と Y 異符号 $Q_1=\bar{1}$

$R_2=2R_1+Y$=0.0010　R1 と Y 異符号 $Q_2=\bar{1}$

$R_3=2R_2+Y$=1.1010　R2 と Y 異符号 $Q_3=\bar{1}$

$R_4=2R_3-Y$=1.1110　R3 と Y 同符号 $Q_4=1$

Q=0.$\bar{1}\bar{1}\bar{1}$1=$\bar{1}$.0011

R=$R_4 2^{-4}$=1.11111110

引放し法（１，０表現）

ＸとＹが同符号のとき，$Q_{-1}=1$

ＸとＹが異符号のとき，$Q_{-1}=0$

初期部分剰余 R0 を

$R_0=X+(1-2Q_{-1})Y$

$R_i=2R_{i-1}+(1-2Q_{i-1})Y$

R_{i-1} とＹが同符号：$Q_{i-1}=1$

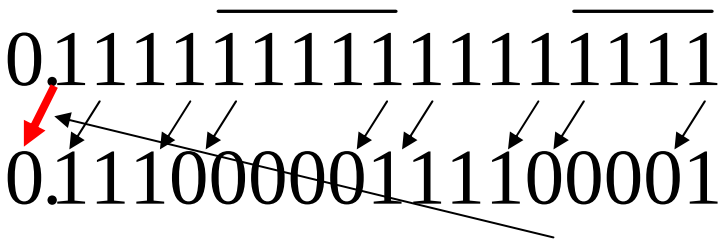
異符号：$Q_{i-1}=0$

$$X=\{2(Q_{-1}-1)+2^{-m}+\Sigma_0^{m-1}2^{-i}Q_i\}Y+2^{-m}R_m$$

引放し法１，$\bar{1}$との関連

X,Y が同符号のとき

1,$\bar{1}$方式



1,$\bar{1}$方式で１なら一つ前を１　　これは0にして欲しい

$\bar{1}$なら一つ前を０

$R_0=X$

$R_1=2X-Y,....$

$R_i=2R_{i-1}-Q_iY$

$R_0=X$ではX,Y＞０で$Q_0=1$となる。

$R_0=X-Y$では、$R_0<0$、Y＞0で

$R_1=2R_0+Y=2X-Y$、$Q_0=0$となる

1,0 方式

$R_1=2X-Y$

$R_i=2R_{i-1}+(1-2Q_{i-1})Y$

R_{i-1}とYが同符号：$Q_{i-1}=1$

異符号：$Q_{i-1}=0$

$Q_0=0$，$R_0=X-Y$ とすればよい

例（引放し法１，０表現）

X=0.10000000

Y=1.0110

R_0=X+Y　=1.1110 X と Y は異符号

Q_{-1}=0

R_1=2R_0-Y=0.0110 R_0 と Y は同符号 Q_0=1

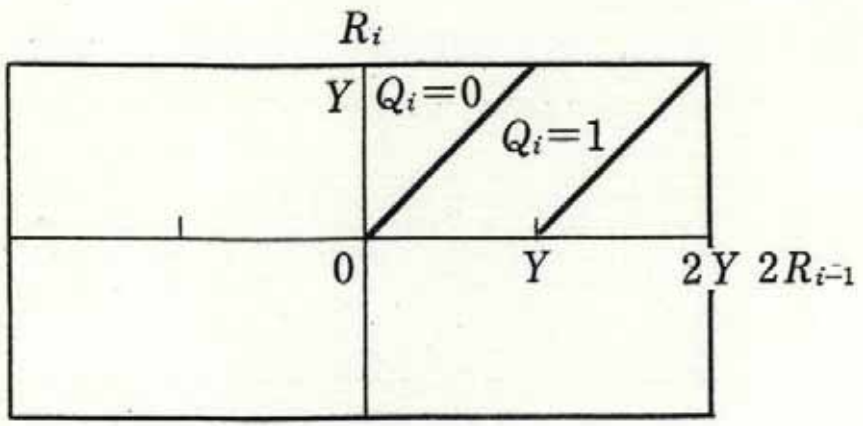
R_2=2R_1+Y=0.0010 R_1 と Y は異符号 Q_1=0

R_3=2R_2+Y=1.1010 R_2 と Y は異符号 Q_2=0

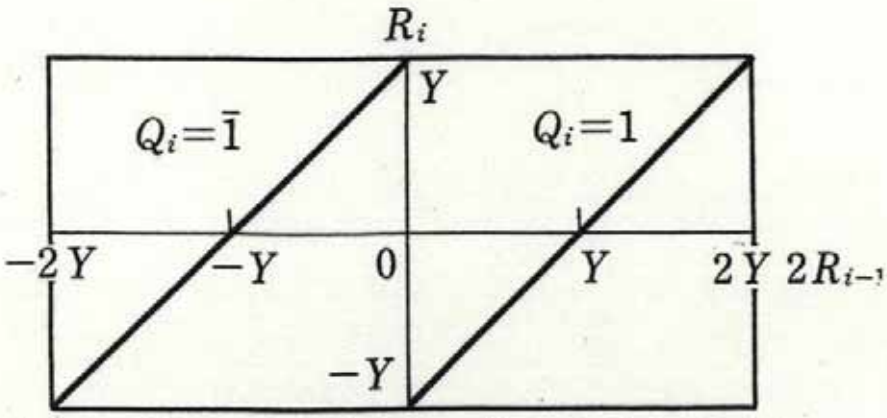
R_4=2R_3-Y=1.1110 R_3 と Y は同符号 Q_3=1

Q=1.001+0.0001=1.0011

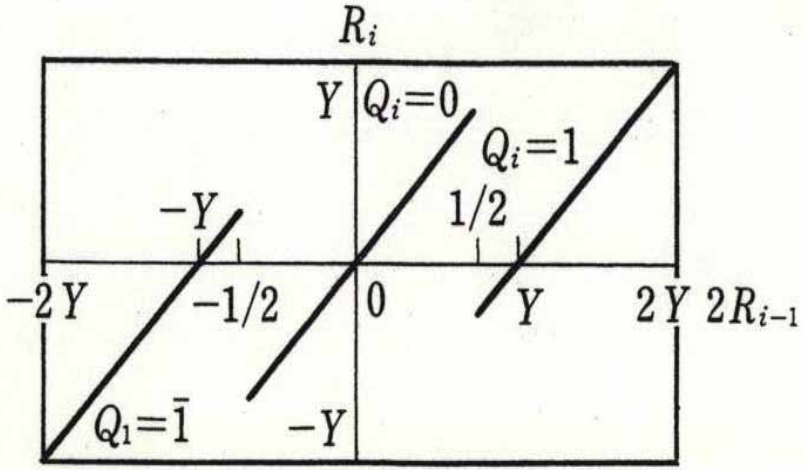
R=$2^{-4}R_4$=1.11111110



(a) 引戻し法



(b) 引放し法(商桁に1,$\bar{1}$のみ許す場合)



(c) SRT法

割算演習

(1) 0.1000÷0.1010

(2) 0.1000÷1.0110

(3) 1.0111÷0.1101

(4) 1.0111÷1.0110

a. $X=0.1000$, $Y=0.1010$, XとYは同符号 $q_0=1$

$r_0=X-Y=1.1110$ r_0とYは異符号 $q_1=0$

$r_1=2r_0+Y=0.0110$ r_1とYは同符号 $q_2=1$

$r_2=2r_1-Y=0.0010$ r_2とYは同符号 $q_3=1$

$r_3=2r_2-Y=1.1010$ r_3とYは異符号 $q_4=0$

$r_4=2r_3+Y=1.1110$

$Q=0.110+0.0001=0.1101$

$R_4=2^{-4}r_4=1.11111110$

b. $X=0.1000$, $Y=1.0110$, XとYは異符号 $q_0=0$

$r_0=X+Y=1.1110$ r_0とYは同符号 $q_1=1$

$r_1=2r_0-Y=0.0110$ r_1とYは異符号 $q_2=0$

$r_2=2r_1+Y=0.0010$ r_2とYは異符号 $q_3=0$

$r_3=2r_2+Y=1.1010$ r_3とYは同符号 $q_4=1$

$r_4=2r_3-Y=1.1110$

$Q=1.001+0.0001=1.0011$

$R_4=2^{-4}r_4=1.11111110$

c. $X=1.0111$, $Y=0.1101$, XとYは異符号 $q_0=0$

$r_0=X+Y=0.0100$ r_0とYは同符号 $q_1=1$

$r_1=2r_0-Y=1.1011$ r_1とYは異符号 $q_2=0$

$r_2=2r_1+Y=0.0011$ r_2とYは同符号 $q_3=1$

$r_3=2r_2-Y=1.1001$ r_3とYは異符号 $q_4=0$

$r_4=2r_3+Y=1.1111$

$Q=1.010+0.0001=1.0101$

$R_4=2^{-4}r_4=1.11111111$

d. $X=1.0111$, $Y=1.0110$, XとYは同符号 $q_0=1$

$r_0=X-Y=0.0001$ r_0とYは異符号 $q_1=0$

$r_1=2r_0+Y=1.1000$ r_1とYは同符号 $q_2=1$

$r_2=2r_1-Y=1.1010$ r_2とYは同符号 $q_3=1$

$r_3=2r_2-Y=1.1110$ r_3とYは同符号 $q_4=1$

$r_4=2r_3-Y=0.0110$

$Q=0.111+0.0001=0.1111$

$R_4=2^{-4}r_4=0.00000110$

q0,q1,q2,q3,q4 を q-1,q0,q1,q2,q3に変更

3.6.4 収束法（convergent　mehtod）

除数の逆数計算

被除数と乗算

(1)Newton-Raphson 法

$f(X)=0$ の根を求める

X_0 としたとき，(X_0，$f(X_0)$）を通り，$f(x)$ に接する直線は

$$Y-f(X_0)=f'(X_0)(X-X_0)$$

となる。X 軸との交点を X_1 とすると，

$$X_1=X_0-f(X_0)/f'(X_0)$$

一般に，

$$X_{i+1}=X_i-f(X_i)/f'(X_i)$$

f(X)=1/X-D とすると

D：除数

ｆ（X）＝０の根：Ｄの逆数

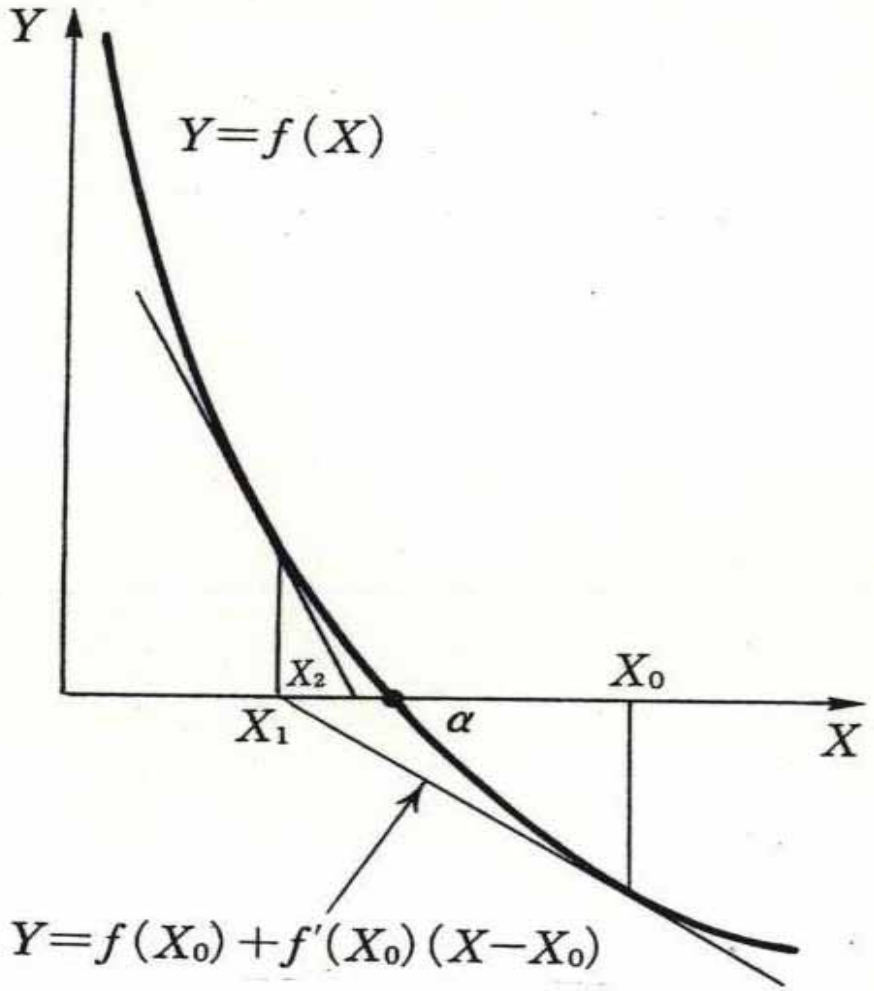


図 3.14　Newton-Raphson 法

漸化式

$$X_{i+1}=X_i(2-DX_i)$$

$1/2 \leqq D<1$ に正規化

例　$D=(0.675487)_{10}$

$1/D=1.4804133$（電卓で計算）

①$X_0=1$ を初期値

$X_1=1.3245130$

$X_2=1.4639957$

$X_3=1.4802314$

$X_4=1.4804134$

②X_0=1.6 を初期値

D を 0.625（２進で 0.101）で近似し、

その逆数、1.6 をテーブルに格納

Ｄ：２進数

　　上位３桁をアドレスとして表参照

X_1=1.4707532

X_2=1.4803504

X_3=1.4804134

③X_0=1.4545454 を初期値

D を 0.65625（２進で 0.10101）で近似

逆数 1.4545454 をテーブルに記憶

Ｄ（２進数）の上位５桁をアドレスとして表参照

X_1=1.4799613

X_2=1.4804132

(2)Goldschmidt の方法

$1/D=(1/D)(R_0/R_0)(R_1/R_1)\cdots(R_n/R_n)$

$DxR_0xRx\cdots xR_n \to 1$

となるような数 $R_0,\cdots, R_n$ が得られれば，

$1/D=R_0xR_1x\cdots xR_n$

$D=1-\delta$

$0<\delta \leqq 1/2$

$1/(1-\delta)=1+\delta+\delta^2+\delta^3+\delta^4+\delta^5\cdots$

$=(1+\delta)(1+\delta^2)(1+\delta^4)(1+\delta^8)\cdots$

に着目して、

$$1/(1-\delta) = \frac{(1+\delta)(1+\delta^2)\cdots(1+\delta^{2n})}{(1-\delta)(1+\delta)(1+\delta^2)\cdots(1+\delta^{2n})}$$

$$= \frac{(1+\delta)(1+\delta^2)\cdots(1+\delta^{2n})}{(1-\delta^{2n+1})}$$

$$1/(1-\delta)=(1+\delta)(1+\delta^2)\cdots(1+\delta^{2n})$$

$2n$：2^n

例

D=(0.675487)10 のとき，δ=0.324513

$(1+\delta)$=1.3245130

$(1+\delta)(1+\delta^2)$=1.4639957

$(1+\delta)(1+\delta^2)(1+\delta^4)$=1.4802313

$(1+\delta)(1+\delta^2)(1+\delta^4)(1+\delta^8)$=1.48041３

3.6.5 除算器

(1)逐次型除算器

(2)配列型並列除算器

X

Y

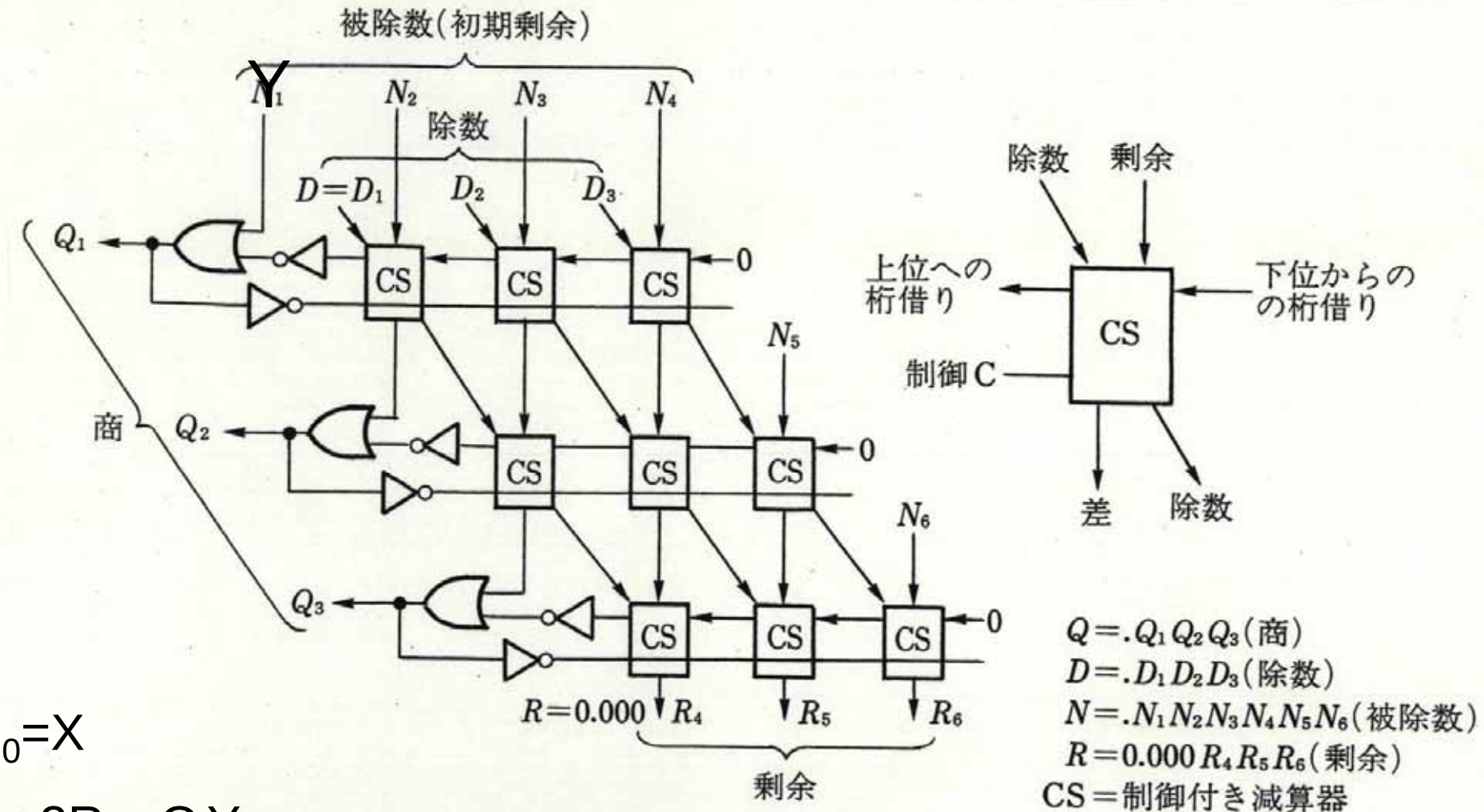


図 3.15 配列型並列除算器（引戻し法）

$R_0=X$

$R_i=2R_{i-1}-Q_iY$

3.7 浮動小数点演算

mr^e

m：仮数，r：基数，e：指数

3.7.1 四則演算

(1)加減算

$m_1 2^{e1} + m_2 2^{e2}$

$e_1 > e_2$

手順

- 指数の桁合わせ

  $e_1 - e_2$

- 仮数の右シフト

  $m_2 2^{-(e1-e2)} 2^{e1}$ となるので、m_2 を右に

  $(e_1 - e_2)$桁算術シフト

- 仮数部の加減算

  $m_1 + m_2 2^{-(e1-e2)}$

- 正規化

  仮数第１桁を１

$0.00112^{10} \rightarrow 0.1100 2^{8}$

(2)乗算

$m_1 2^{e1} \cdot m_2 2^{e2} = m_1 \cdot m_2 2^{e1+e2}$

- 指数部の加算
- 仮数部の乗算
- 正規化

(3)除算

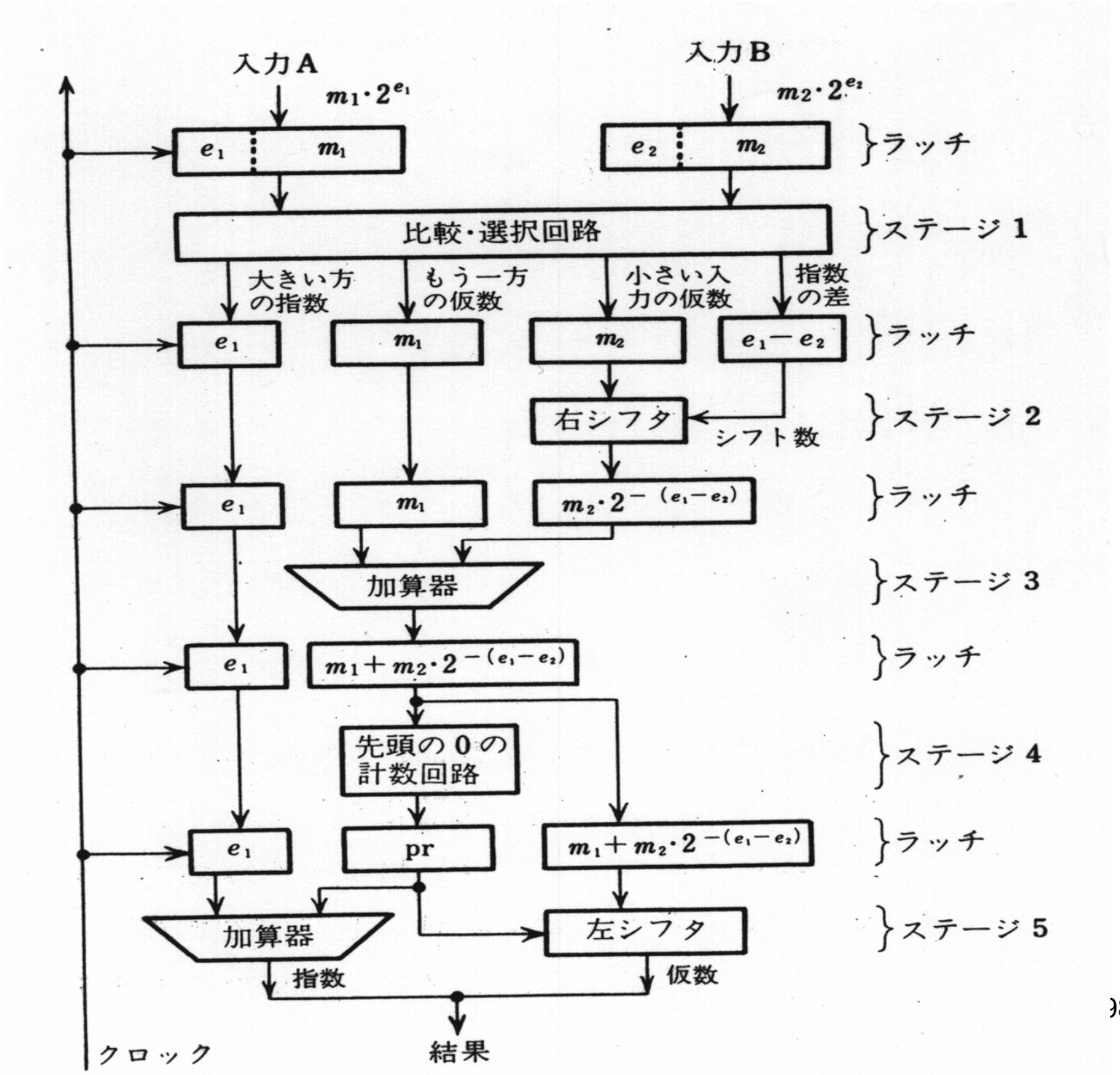
$(m_1 2^{e1}) / (m_2 2^{e2}) = (m_1 / m_2) 2^{e1-e2}$

- 被除数の桁合わせ

  $(m_1/m_2)<1$ となるように

  m_1 を右シフト，e_1 に+1

- 仮数部の除算
- 指数部の減算


入力A
$m_1 \cdot 2^{e_1}$
入力B
$m_2 \cdot 2^{e_2}$
e_1 m_1
e_2 m_2
ラッチ
比較・選択回路
ステージ 1
大きい方の指数
もう一方の仮数
小さい入力の仮数
指数の差
e_1
m_1
m_2
$e_1 - e_2$
ラッチ
右シフタ
シフト数
ステージ 2
e_1
m_1
$m_2 \cdot 2^{-(e_1-e_2)}$
ラッチ
加算器
ステージ 3
e_1
$m_1 + m_2 \cdot 2^{-(e_1-e_2)}$
ラッチ
先頭の0の計数回路
ステージ 4
e_1
pr
$m_1 + m_2 \cdot 2^{-(e_1-e_2)}$
ラッチ
加算器
左シフタ
ステージ 5
指数
仮数
クロック
結果

パイプラインの基本方式

（１）同期式パイプライン

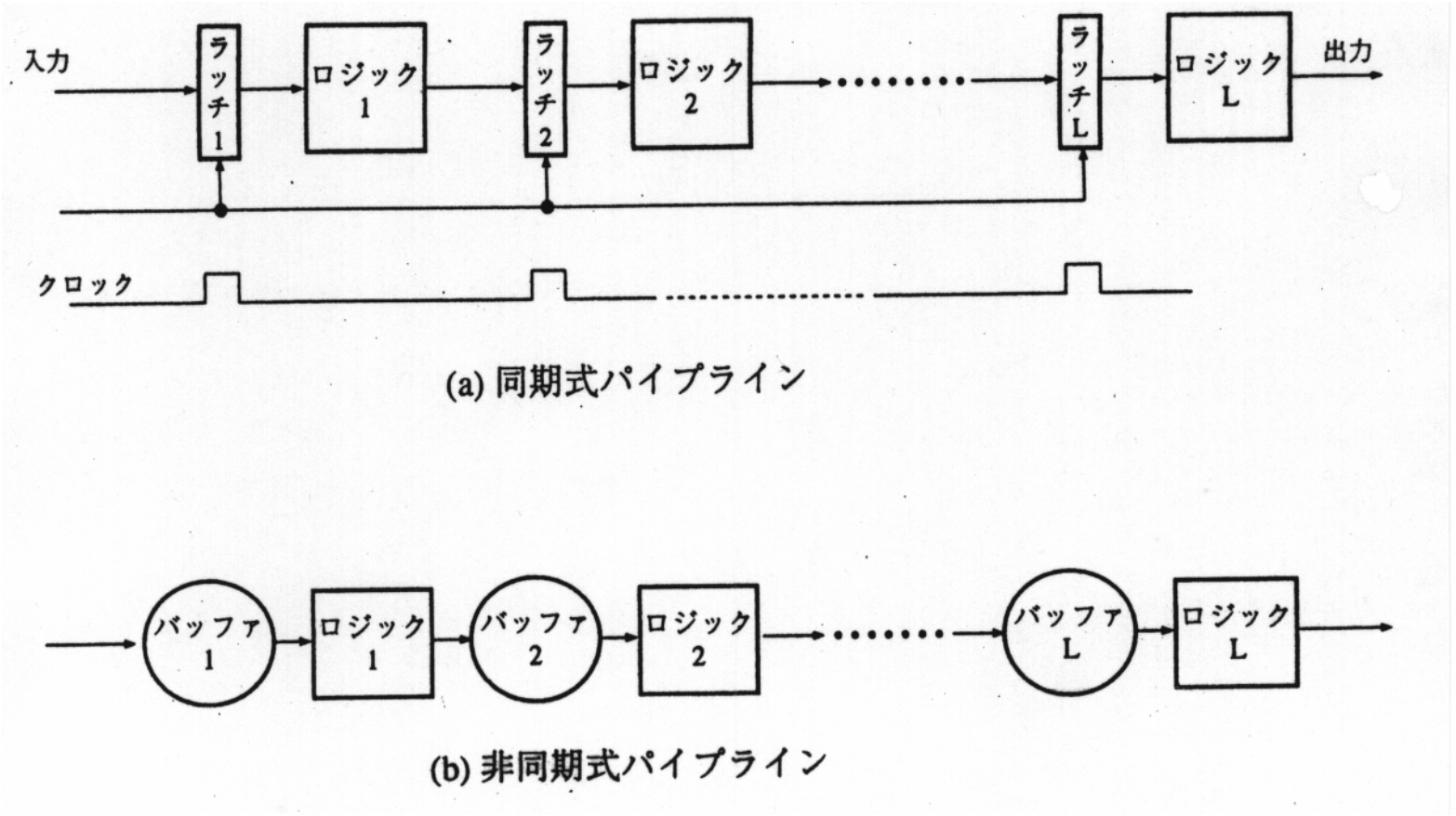
半性能長：最大性能（$1/\tau_v$）の半分の
性能となるデータ要素数

演算時間

$$T=S+(N-1+L)\tau_v \qquad （１）$$

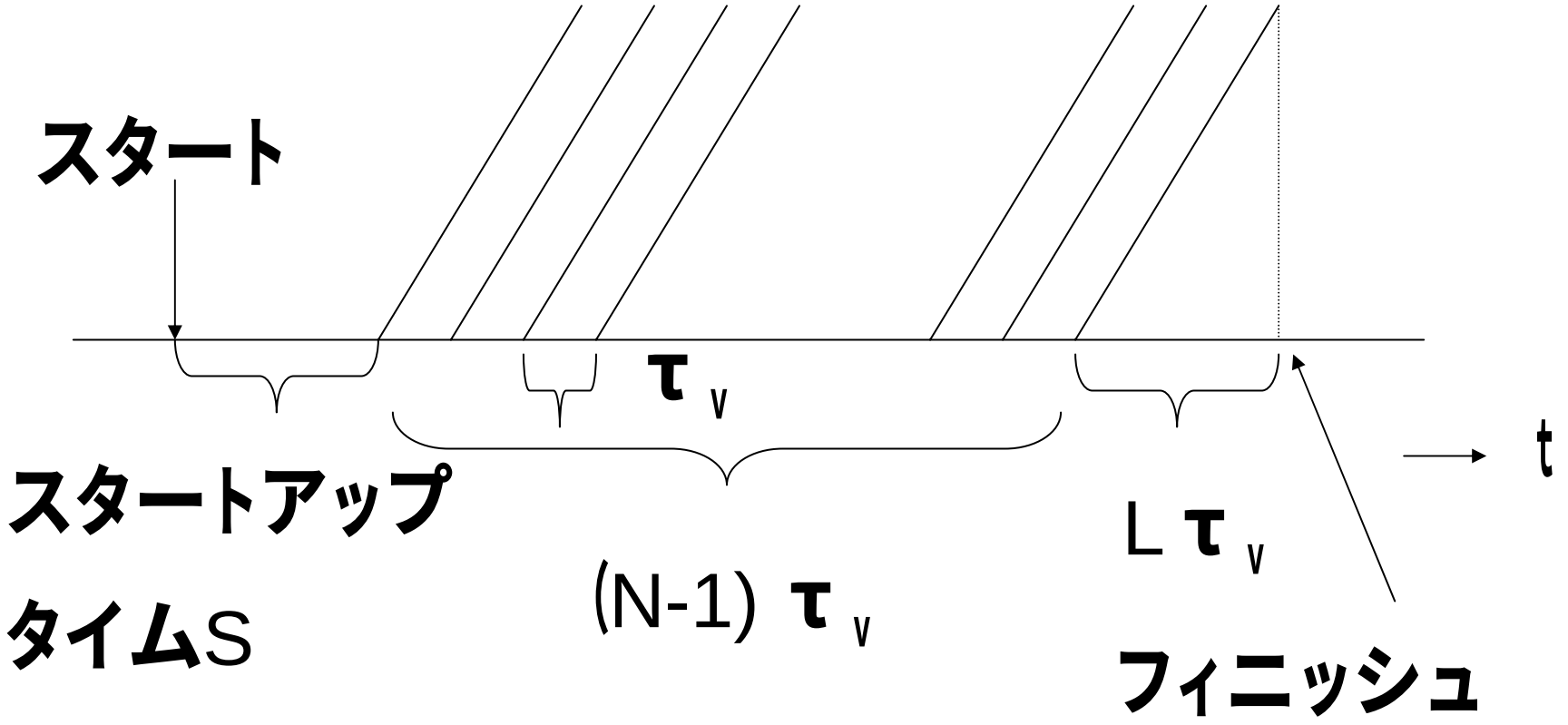
$$=(N+N_{1/2})\tau_v \qquad （２）$$

$$1/(2\tau_v)=N_{1/2}/(S+(N_{1/2}-1+L)\tau_v)$$



(a) 同期式パイプライン

(b) 非同期式パイプライン


スタート
τv
t
スタートアップ
タイムS
(N-1) τv
L τv
フィニッシュ

3.7.3 浮動小数点数の標準化

IBM 形式の浮動小数点数の形式

IEEE（アメリカの電気電子学会．
アイトリプルイーと読む)）標準

(1)単精度浮動小数点

符号 s	指数 8 e	仮数 23 f

符号(s)１ビット，

指数(e)８ビット（げたばき１２７）

仮数(f) ２３ビット

基数２

①e=255 で f≠0 なら数の値は NaN（非数）

②e=255 で f=0 なら数の値は$(-1)^s\infty$

③0<e<255 なら数の値は　正規化数

$(-1)^s 2^{e-127}(1.f)$

④e=0、f≠0 なら数の値は

$(-1)^s 2^{-126}(0.f)$　非正規化数

⑤e=f=0 なら数の値は$(-1)^s 0$

(2)倍精度浮動小数点

符号(s)１ビット，

指数（ｅ）11 ビット（げたばき 1023），

仮数（ｆ）52 ビット

(3)IEEE 標準の特徴

①簡約表現

基数が２

仮数の第１桁は暗黙的に１

②非正規化数

正規化表現２$^{-126}$：最小数

非正規化（denormalize）表現可

③数値としての∞

∞の数値

１／０は∞を値

④非数の導入

$\sqrt{-1}$，0/0，∞-∞：NaN（Not a Number）

⑤丸めの方式の指定

４つの丸めの方式

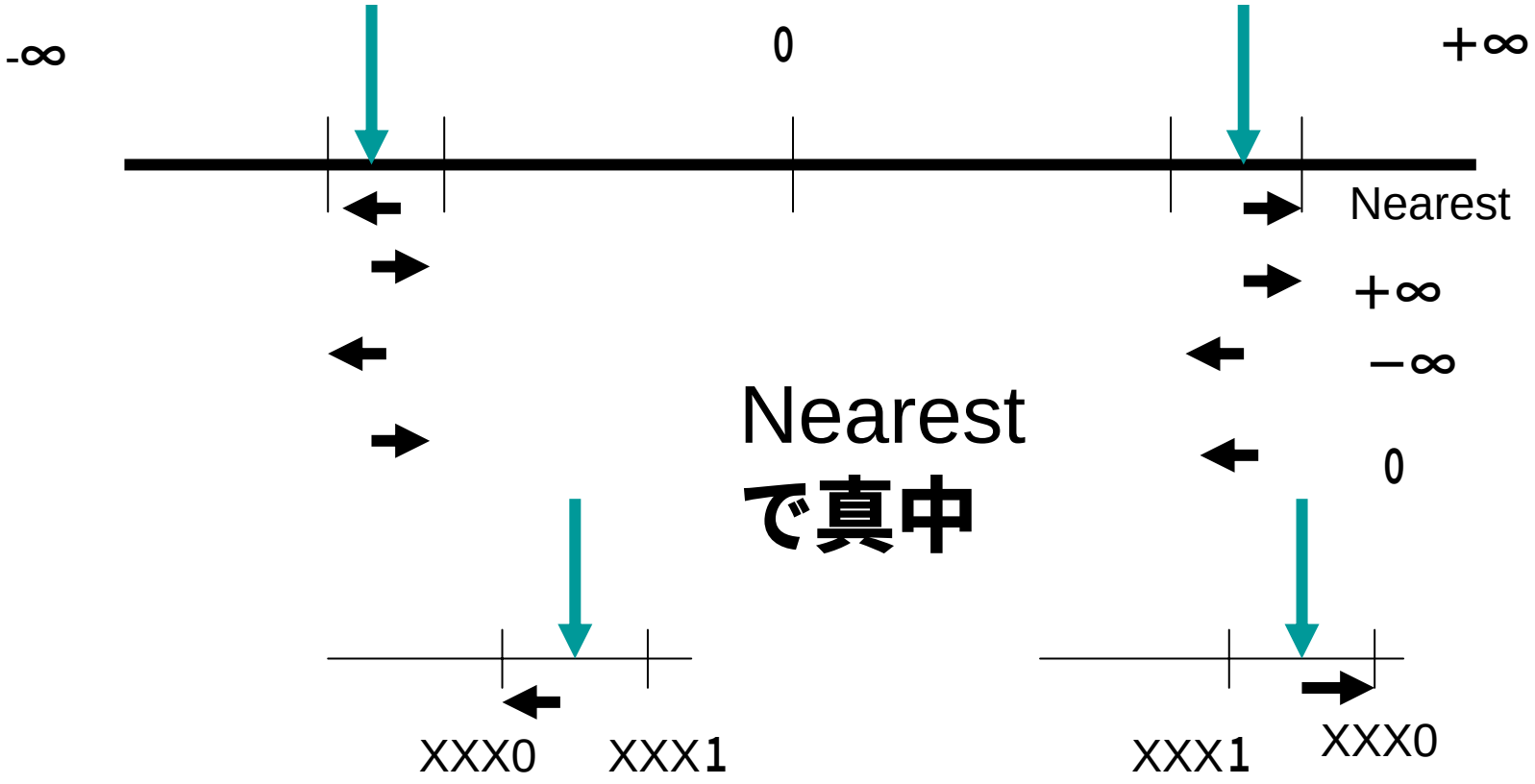
・最も近い数への丸め

・+∞方向への丸め

・-∞方向への丸め

・0 方向への丸め

丸め方式



$X_0 = X$、$X_1 = (X_0 - Y) + Y$, $X_2 = (X_1 - Y) + Y$

$X_n = (X_{n-1} - Y) + Y$

四捨五入

X=1, Y=−0.555, $X_0 - Y$ = 1.555　1.56,

X_1 = 1.56 − 0.555 = 1.005　1.01

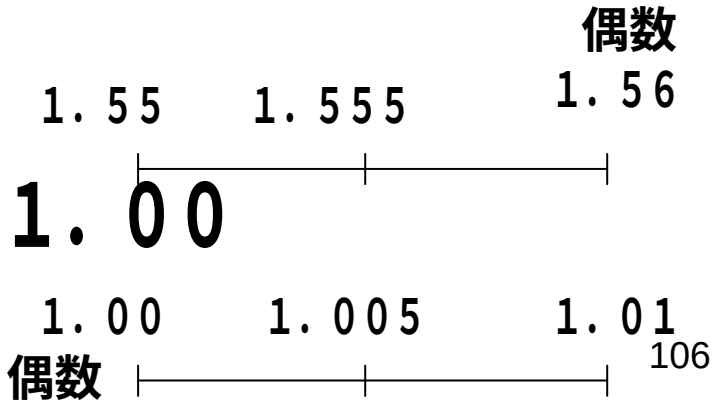
$X_1 - Y$ = 1.01 + 0.555 = 1.565　1.57,

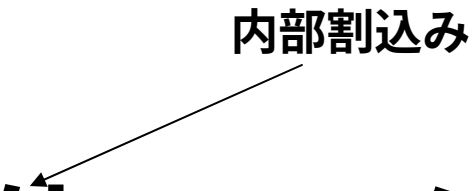
X_2 = 1.57 − 0.555 = 1.015　1.02

偶数に丸め

X_0-Y=1.555　1.56、

X_1=1.56-0.555=1.005　1.00





⑥例外：マスク可能

- オーバフロー：正規化数より大2^{128}
- アンダフロー：非正規化数発生・演算
- 演算異常：NaN の発生時
- ゼロデバイド
- 精度異常（inexact）：1/３など