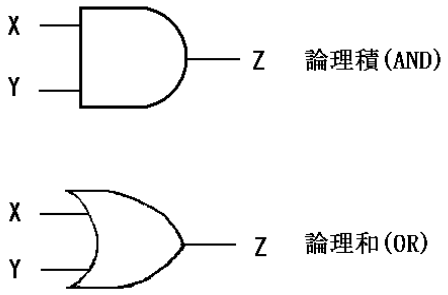
JAPLA 研究会資料 2008/5/24

Ｊによるブール代数とスイッチング回路網の計算

西川　利男

１．はじめに



コンピュータのディジタル論理の基本となるスイッチング回路網はブール代数に基礎をおくシャノン(C. Shannon)のスイッチング代数として計算され、回路の設計がなされる[1]。しかし実際の込み入ったディジタル回路ではブール代数の手計算だけで行うのは相当困難である。

Ｊではビット操作のための　AND(*.)やOR(+.)などのプリミティブを備えていて、２値演算(Binary Calculation)が容易に行える。これを用いて、スイッチング論理代数の計算を行うツールを作ってみた。

２．Ｊによるスイッチング論理代数の実現

シャノンのスイッチング代数では２値(0, 1)のいずれかの値を持つ論理変数に対して、論理和(OR)、論理積(AND)、否定(NOT)の３つの演算を組み合わせた一種の代数計算を行う。演算の結果は真理値表(Truth Table)として示される。

Ｊでこれらを実現するためには、まず論理変数の集合を次のようにボックスを用いて表現することにした。

２変数では

```
   AB =: , {0 1;0 1
   AB
+---+---+---+---+
|0 0|0 1|1 0|1 1|
+---+---+---+---+
```

となり、

３変数では

```
   ABC =: , {0 1;0 1;0 1
   ABC
+-----+-----+-----+-----+-----+-----+-----+-----+
|0 0 0|0 0 1|0 1 0|0 1 1|1 0 0|1 0 1|1 1 0|1 1 1|
+-----+-----+-----+-----+-----+-----+-----+-----+
```

のようになる。

一方、論理関数はＪのプリミティブを使って、次のように定義できる。

```
   and =: *.
   or =: +.
   not=: -.
   xor =: 1:`0:@.=
```

参考書

[1] 遠山武「現代電気電子工学の基礎」p.270-1, オーム社(1982).

```
  nand =: not@and
  nor =: not@or
```

これを用いると、次のように演算される。

```
   and /L:0 AB
+-+-+-+-+
|0|0|0|1|
+-+-+-+-+
   or /L:0 AB
+-+-+-+-+
|0|1|1|1|
+-+-+-+-+
   xor /L:0 AB
+-+-+-+-+
|0|1|1|0|
+-+-+-+-+
```

これらの結果は以下のようにまとめて真理値表として表すことができる。

```
   ('and';'or';'nand';'nor';'xor') tru_tables 'AB'
+---+-+-+-+-+-+
|0 0|0|0|1|1|0|
+---+-+-+-+-+-+
|0 1|0|1|1|0|1|
+---+-+-+-+-+-+
|1 0|0|1|1|0|1|
+---+-+-+-+-+-+
|1 1|1|1|0|0|0|
+---+-+-+-+-+-+
```

真理値表を得る動詞 tru_tables の定義は末尾のＪのリストを参照のこと。

ここで有名なド・モルガン(de Morgan)の定理を確かめてみよう。

```
    not (X or Y) = (not X) and (not Y)
NB. de Morgan's Theorem
demorg0 =: 3 : 0 L:0
'X Y' =. y.
-. X +. Y
)

demorg1 =: 3 : 0 L:0
'X Y' =. y.
(-.X) *. (-.Y)
)

   demorg0 AB
+-+-+-+-+
|1|0|0|0|
+-+-+-+-+
   demorg1 AB
+-+-+-+-+
|1|0|0|0|
+-+-+-+-+
```

３．いくつかの問題例

３．１

次の論理式

Z = X・Y’ + X’・Y’ + X’・Y　　（ここでX’はXの否定を表す）

は否定論理積nand (X・Y)’となることを示せ。

Jで直ちにつぎのようなプログラムを作りさえすればよい。

```
test1 =: 3 : 0 L:0
'X Y' =. y.
((-.X) *. Y) +. ((-.X) *. (-.Y)) +. (X *. (-.Y))
)
```

次のように確かめられる。

```
   test1 AB
+-+-+-+-+
|1|1|1|0|
+-+-+-+-+
   nand /L:0 AB
+-+-+-+-+
|1|1|1|0|
+-+-+-+-+
```

３．２

次の３値の論理式

f = X’・Y’・Z + X’・Y・Z　　（ここでX’はXの否定を表す）

を簡単にし、かつ真理値表を計算せよ。

「電気電子事典」p.1528-1530

「付録E ブール代数とスイッチング代数」の例題

Jによる論理関数の定義は次のようになる。

```
f0 =: 3 : 0 L:0
'X Y Z' =. y.
((-.X) *. (-.Y) *. Z) +. ((-.X) *. Y *. Z)
)
```

これは以下の関数と等しい。

```
f1 =: 3 : 0 L:0
'X Y Z' =. y.
(-. X) *. Z
)
```

実行すると次のようになる。

```
   f0 ABC
+-+-+-+-+-+-+-+-+
|0|1|0|1|0|0|0|0|
+-+-+-+-+-+-+-+-+
   f1 ABC
+-+-+-+-+-+-+-+-+
|0|1|0|1|0|0|0|0|
+-+-+-+-+-+-+-+-+
```

なお、手計算では分配則と相補則(Y’+ Y = 1)とを使って次のように得られるが、極めて技巧的であり、いつでも可能というわけではない。

f = X’・Y’・Z+ X’・Y・Z
= X’・(Y’+ Y) ・Z
= X’・1・Z
= X’・Z

```
NB. Boolean Algebra for Digital Switch Circuit
NB.   by Toshio Nishikawa 2008/5/19
AB =: , {0 1;0 1
ABC =: , {0 1;0 1;0 1

NB. de Morgan's Theorem
demorg0 =: 3 : 0 L:0
'X Y' =. y.
-. X +. Y
)

demorg1 =: 3 : 0 L:0
'X Y' =. y.
(-.X) *. (-.Y)
)

demorg3_0 =: 3 : 0 L:0
'X Y Z' =. y.
-. X +. Y +. Z
)

demorg3_1 =: 3 : 0 L:0
'X Y Z' =. y.
(-.X) *. (-.Y) *. (-.Z)
)

NB. 例題
NB. 「電気電子事典」p.1528-1530
NB.  「付録E ブール代数とスイッチング代数」
f0 =: 3 : 0 L:0
'X Y Z' =. y.
((-.X) *. (-.Y) *. Z) +. ((-.X) *. Y *. Z)
)

f1 =: 3 : 0 L:0
'X Y Z' =. y.
(-. X) *. Z
)

NB. 「電気電子事典」example p. 1530
g0 =: 3 : 0 L:0
'X Y Z' =. y.
G01 =. (-. X) *. (-. Y) *. Z
G02 =. (-. X) *. Y *. Z
G01 +. G02
)
```

```
g1 =: 3 : 0 L:0
'X Y Z' =. y.
G11 =. X +. (-. Y) +. Z
G12 =. (-. X) +. Y +. Z
G13 =. (-. X) +. Y +. (-. Z)
G11 *. G12 *. G13
)

NB.  Usages =>  g0 tru_table ABC
NB.             g1 tru_table ABC
tru_table =: 1 : 0
fun =. u.
y. ,"(0) fun y.
)

NB. display truth tables for several logic functions
and =: *.
or =: +.
not=: -.
xor =: 1:`0:@.=
nand =: not@and
nor =: not@or

NB.   ('and';'or';'nand';'nor';'xor') tru_tables 'AB'
tru_tables =: 3 : 0
:
f =. > {.x.
h =. }. x.
F =. ". f, ' /L:0 ', y.
TABLE =. ("y.), "0 F
while. (# h) > 0
 do.
   f =. > {. h
   F =. ". f, ' /L:0 ', y.
   TABLE =. TABLE, "(1 0) F
   h =. }. h
 end.
TABLE
)
NB. 加藤肇, 見城尚志, 高橋久
NB. 「図解わかる電子回路」p.283 講談社ブルーバックス
NB. build OR circuit from composite AND device
or_from_and =: 3 : 0 L:0
'X Y' =. y.
-. (-.X) *. (-.Y)
```

```
)
NB.   or_from_and AB
NB. +-+-+-+-+
NB. |0|1|1|1|
NB. +-+-+-+-+
NB.   +. / L:0 AB
NB. +-+-+-+-+
NB. |0|1|1|1|
NB. +-+-+-+-+

NB. ===============================================
NB. デジタル技術検定２級制御部門, 問題(4)
kentei0 =: 3 : 0 L:0
'X Y Z' =. y.
K0 =: X +. (-.Y) +. Z
K1 =: (-.X) +. Y +. Z
K2 =: (-.X) +. (-.Y) +. Z
K3 =: X +. (-.Y) +. (-.Z)
K0 *. K1 *. K2 *. K3
)

kentei =: 3 : 0 L:0
'X Y Z' =. y.
K40 =: (-.X) *. Y *. (-.Z)
K41 =: X *. (-.Y) *. (-.Z)
K42 =: X *. Y *. (-.Z)
K43 =: (-.X) *. Y *. Z
K40 +. K41 +. K42 +. K43
)

kentei1 =: 3 : 0 L:0
-. kentei y.
)


p0 =: 3 : 0 L:0
'X Y Z' =. y.
X +. (-.Y) +. Z
)

q0 =: 3 : 0 L:0
'X Y Z' =. y.
(-.X) *. Y *. (-.Z)
)
```