Logic Functions over Galois Field GF(4)

Logic Functions over Galois Field GF(4)

Zensho NAKAO*

Abstract The elements of Galois field GF(4) are represented by four numerals $\{0, 1, 2, 3\}$; it is shown that all quaternary logic functions can be expressed in a sort of standard form as polynomial functions over GF(4); the two field operators of GF(4) are proposed as basic logic gates and are used as basic building blocks in the representation of the logic functions.

1. Introduction

In recent papers, Hachimine and Zukeran [HZ1] proposed a set of four-valued logic functions and demonstrated the completeness of the system; they also designed several quaternary logic circuits [HZ2].

The objective of this note is to show that how Galois field GF(4) (i.e., a finite field of four elements) can be used effectively to represent quaternary logic functions such as the ones studied in [HZ1, HZ2] in standard polynominal forms as was done in [N1, NZK].

2. Preliminaries

The set $A_2 = \{0, 1\}$ of two symbols 0, 1 can be made into a Boolean algebra by furnishing it with two binary operations $\vee$, $\wedge$ and one unary operation $^{-}$ which are defined by the following (truth) tables:

$\vee$	0	1
0	0	1
1	1	1

$\wedge$	0	1
0	0	0
1	0	1

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

Table 1. Boolean operators

On A_2, introduce two binary operations +, • (or juxtaposition) by the (truth) tables:

+	0	1
0	0	1
1	1	0

•	0	1
0	0	0
1	0	1

Table 2. Field operators

Received: October 31, 1984

*Department of Electronics Engineering and Computer Science

The algebraic operations now transform the structure on A_2 into that of Galois field GF(2). In fact, the Boolean algebraic structure $(A_2; \vee, \wedge, \bar{\ })$ and the field structure (GF(2); +, •) are related by the following transformation formulas:

$$\text{(1)} \qquad \begin{array}{ll} x \wedge y = xy & xy = x \wedge y \\ x \vee y = x + y + xy & x + y = (x \wedge \bar{y}) \vee (\bar{x} \wedge y) \\ \bar{x} = x + 1 & \end{array}$$

Since GF(2) is a field, the operations in Table 2 define subtraction and division implicitly (namely, the charts are used backward), which are the operations impossible in the Boolean algebra A_2. Also, it is known in field theory that we can always enlarge GF(2) to the extension field $GF(2^n)$ (n is a natural number) which possesses exactly 2^n elements by adjoining a zero of some irreducible polynomial over GF(2).

In this note we let n = 2 and use GF(4) in studying $A_4 = \{0, 1, 2, 3\}$ with a quaternary logic structure. We can obtain GF(4) as the splitting field of the irreducible polynomial $p(x) = x^2 + x + 1$ over GF(2) by adjoining a solution α of $p(x) = 0$ in an extension field of GF(2). Since GF(4) is a two dimensional vector field over GF(2) with the base $\{1, \alpha\}$, where $\alpha^2 = \alpha + 1$, GF(4) has four elements $0, 1, \alpha, \alpha + 1$. For ease of reference and computation, we introduce two numerals 2 and 3, and let $\alpha = 2$ and $\alpha + 1 = 3$; we obtain the following addition and multiplication tables which are in fact the truth tables for the binary operations:

+	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

•	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

Table 3. Field operators

Recall that (GF(4); +) is Klein's Viergruppe (hence, not cyclic), and that (GF(4) – {0}; •) is a cyclic group of order 3; both are Abelian (i.e., the tables are symmetric with respect to the main diagonal).

It is to be noted that there are quaternary logic functions on A_4 which cannot be expressed as standard Boolean functions, i.e., $(A_4; \vee, \wedge, \bar{\ })$ is not a complete system.

In the rest of the section, we quote the necessary results from Galois theory:

$$\text{(2)} \qquad (x + y)^2 = x^2 + y^2 \ \forall\, x, y \in GF(4)$$

$$\text{(3)} \qquad x + x = 0 \ \forall\, x \in GF(4)$$

Any function from a finite field into itself is known to be a polymomial function (use Lagrange's interpolation formula, for example); the following results give us the necessary formulas for our specific purpose [T]:

(4) If $f: GF(4) \longrightarrow GF(4)$ is a function, then f can be expressed in a polynomial form over GF(4):

$$f(x) = a_0 + a_1x + a_2x^2 + a_3x^3,$$

where $a_0 = f(0)$, $a_i = \sum_{x \in GF(4)} x^{3-i} f(x)$, $(1 \leq i \leq 3)$.

(5) If $f: GF(4) \times GF(4) \longrightarrow GF(4)$ is a mapping, then f can be realized as a polynomial mapping over GF(4) in two variables:

$$f(x, y) = \sum_{0 \leq i, j \leq 3} a_{ij} x^i y^j,$$

where $a_{00} = f(0, 0)$, $a_{i0} = \sum_{x \in GF(4)} x^{3-i} f(x, 0)$, $a_{0i} = \sum_{y \in GF(4)} y^{3-i} f(0, y)$,

$a_{ij} = \sum_{x, y \in GF(4)} x^{3-i} y^{3-j} f(x, y)$, $(1 \leq i, j \leq 3)$.

Note that in formulas (4) and (5) above, all calculations must be done in Galois field GF(4), and also that (5) can be generalized to mappings

$$f: GF(4)^n \longrightarrow GF(4)$$

of n (≥ 1) variables. With this rigid structure, we will be able to express all quaternary logic functions as polynomial functions on GF(4) or $GF(4)^2$ (or $GF(4)^n$, if necessary) of total degree at most 6.

3. Translation of A_4 to GF(4)

It is proved that (MIN, MAX, $x^{(0,1)}$, 1, 2) forms a complete system of quaternary logic functions in [HZ1]; we are going to express all those quaternary logic functions explicitly as polynomials over GF(4), which is a consequence of the evident fact that (GF(4); +, •) is another complete system of quaternary logic functions. The BASIC programs used are included in the Appendix.

a. MIN[x, y]

Operation tables for the binary operators MIN[x, y] and MAX[x, y] are given below:

MIN	0	1	2	3
0	0	0	0	0
1	0	1	1	1
2	0	1	2	2
3	0	1	2	3

MAX	0	1	2	3
0	0	1	2	3
1	1	1	2	3
2	2	2	2	3
3	3	3	3	3

Table 4. MIN and MAX operators

By using formulas for a_{ij} in (5), we can obtain the following results:

$$a_{00} = a_{01} = a_{10} = a_{02} = a_{20} = a_{03} = a_{30} = a_{11} = a_{33} = 0 \tag{6}$$
$$a_{22} = 1,\quad a_{12} = a_{21} = a_{13} = a_{31} = 2,\quad a_{23} = a_{32} = 3$$

Thus we get a polynomial expression for MIN[x, y]:

$$\mathrm{MIN}[x, y] = 2xy^2 + 2xy^3 + 2x^2y + 2x^3y + x^2y^2 + 3x^2y^3 + 3x^3y^2 \tag{7}$$

By using (2) and collecting like terms, we can change MIN[x, y] into a less formidable polynomial in elementary symmetric functions $(x + y)$ and (xy):

$$\mathrm{MIN}[x, y] = xy\ \{2(x + y) + 2(x + y)^2 + xy[1 + 3(x + y)]\} \tag{8}$$

b. MAX[x, y]

Computing similarly as in MIN[x, y], we can derive the following results:

$$a_{00} = a_{02} = a_{20} = a_{03} = a_{30} = a_{11} = a_{33} = 0,\quad a_{23} = a_{32} = 3 \tag{9}$$
$$a_{01} = a_{10} = a_{22} = 1,\quad a_{12} = a_{21} = a_{13} = a_{31} = 2$$

Therefore, a polynomial expression for MAX[x, y] is:

$$\mathrm{MAX}[x, y] = x + y + 2x^2y + 2xy^2 + 2x^3y + 2xy^3 + x^2y^2 + 3x^3y^2 + 3x^2y^3 \tag{10}$$

Rewriting the result above as a polynomial in $(x + y)$ and (xy), we get:

$$\mathrm{MAX}[x, y] = (x + y) + xy[2(x + y) + 2(x + y)^2 + xy + 3xy(x + y)] \tag{11}$$

c. $x^{\{0,2\}}$

The unary operator $x^{\{0,2\}}$ is defined by:

$$x^{\{0,2\}} = 3 \text{ if } x \in \{0, 2\} \tag{12}$$
$$= 0 \text{ otherwise}$$

A truth table for the operator is given below:

x	0	1	2	3
$x^{\{0,2\}}$	3	0	3	0

Table 5. Truth table for $x^{\{0,2\}}$

Repeated applications of the formulas for a_i in (4) yield:

$$a_0 = 3,\quad a_1 = 2,\quad a_2 = 1,\quad a_3 = 0 \tag{13}$$

Hence, we obtain a polynomial function for $x^{\{0,2\}}$:

$$x^{\{0,2\}} = 3 + 2x + x^2 = (1 + x)(3 + x) \tag{14}$$

Putting pieces obtained together, we demonstrated that the algebraic structure (GF(4); +, •) provides an effective method for deriving standard forms of the quaternary logic functions.

4. Other translations

For completeness of presentation, we include the polynomial formulas for the binary operator NOR[x, y]; the unary operators ${}^{(k)}x$ and $x^{(k)}$ which are discussed in [HZ2]. Their definitions follow:

(15) $$\mathrm{NOR}[x, y] = 3 + \mathrm{MAX}[x, y]$$

(16) $${}^{(k)}x \equiv x + k \pmod 4, \quad k = 0, 1, 2, 3$$

(17) $$\begin{aligned} x^{(k)} &= 3 \text{ if } x = k \qquad k = 0, 1, 2, 3 \\ &= 0 \text{ otherwise} \end{aligned}$$

Polynomial representations for the operators are given in the following:

(18) $$\begin{aligned} \mathrm{NOR}[x, y] &= 3 + \mathrm{MAX}[x, y] \\ &= 3 + (x + y) + xy[2(x + y) + 2(x + y)^2 + xy + 3xy(x + y)] \end{aligned}$$

(19) $${}^{(k)}x = (x + k) + xk[3 + 2(x + k) + xk], \quad k = 0, 1, 2, 3$$

(20) $$x^{(k)} = 3[1 + (x + k)^3], \quad k = 0, 1, 2, 3$$

5. Conclusions

We found that all quaternary logic functions of one or two variables can be realized as polynomial functions over Galois field GF(4) in one or two variables of total degree at most 6. An obvious advantage for having polynomial expressions is that we can formally manipulate the elements of A_4, i.e., GF(4) with four arithmetic operations of GF(4) itself as we normally do with the real (or complex) number field.

If we can design (x + y) and (xy) as basic logic circuit elements*, then we can easily construct other circuits such as MIN, MAX, $x^{\{0,2\}}$, NOR, ${}^{(k)}x$ and $x^{(k)}$. For illustration, take MIN, MAX and $x^{\{0,2\}}$. We introduce two logic gate symbols for the operators (x + y) and (xy):

Figure 1. Circuit symbols for (x + y) and (xy)

Then the following diagrams present one possible design for each function:

*The logic circuit elements were designed by Mr. Chotei Zukeran, Department of Electrical Engineering, Ryukyu University; and are included as Figures in the Appendix.

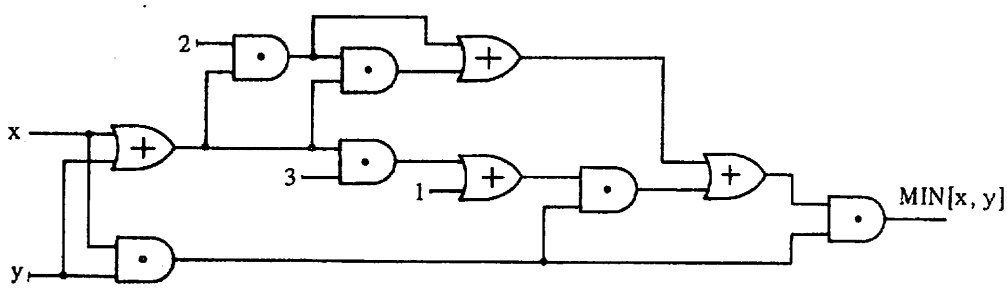

Figure 2. MIN[x, y] gate

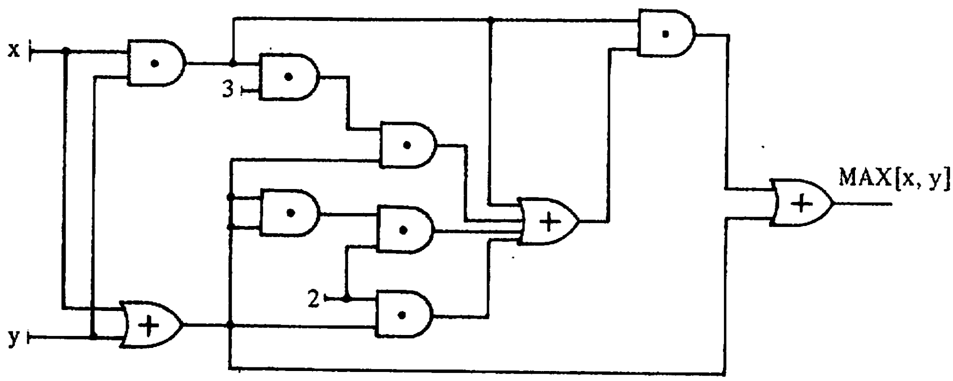

Figure 3. MAX[x, y] gate

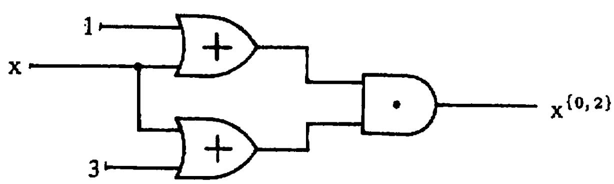

Figure 4. Logic gate for $x^{\{0,2\}}$

Some properties of quanternary logic functions over GF(4) will be discussed further in the forthcoming paper [N2].

References

[HZ1] Hachimine, G and Zukeran, C: A proposal for a complete set of four valued logic functions and its circuit construction, *Trans. IECE* (Japanese), 1981, pp. 901-2.

[HZ2] ____________________: Construction of quaternary logic functions, *Trans IECE* (Japanese), 1983, pp. 302-308.

[N1] Nakao, Z: A field-theoretic view of logic functions, *Bull. Faculty of Engineering, Univ. of the Ryukyus,* No. 28, 1984, pp. 55-66.

[N2] ______: *Vector-valued representation of quaternary logic functions over Galois field GF(4)* (Pre-print).

[NZK] Nakao, Z, Zukeran, C and Kyan, S: *Quaternary logic functions over Galois field GF(4)* (Pre-print).

[T] Takahashi, I: *Combinatorics and its applications* (Japanese), Iwanami 1979, pp. 199-204.

Appendix

a. BASIC programs

```
10  PR# 0: REM  Printer off
12  HOME
14  DIM M(3, 3), A(3, 3), B(3), F(3)
16  FOR I = 0 TO 3
18  FOR J = 0 TO 3
20  READ M(I, J): REM  Define multiplication
22  NEXT J
24  NEXT I
26  DATA  0, 0, 0, 0, 0, 1, 2, 3, 0, 2, 3, 1, 0, 3, 1, 2
28  FOR I = 0 TO 3
30  FOR J = 0 TO 3
32  READ A(I, J): REM  Define addition
34  NEXT J
36  NEXT I
38  DATA  0, 1, 2, 3, 1, 0, 3, 2, 2, 3, 0, 1, 3, 2, 1, 0
40  HOME
42  PRINT  "Type in: f(0), f(1), f(2), f(3)"
44  PRINT
46  FOR X = 0 TO 3
48  INPUT F(X): REM  Define the logic function
50  NEXT X
52  PRINT
54  PRINT  "Hit any key to continue";
56  GET A$
58  HOME
60  PR# 1: REM  Printer on
62  PRINT : REM  Display the function table
64  PRINT "x"; "———>"; "f(x)"
66  PRINT
68  FOR X = 0 TO 3
70  PRINT X; "———>"; F(x)
72  NEXT X
74  PR# 0: REM  Printer off
76  PRINT
78  REM  The coefficients are determined
80  B(0) = F(0)
82  B(1) = 0
84  FOR X = 0 TO 3
86  I = M(X, X)
88  J = M(I, F(X))
```

```
90   B(1) = A(B(1), J)
92   NEXT X
94   B(2) = 0
96   FOR X = 0 TO 3
98   J = M(X, F(X))
100  B(2) = A(B(2), J)
102  NEXT X
104  B(3) = 0
106  FOR X = 0 TO 3
108  J = F(X)
110  B(3) = A(B(3), J)
112  NEXT X
114  REM  Output the result in polynomial form
116  PR# 1:  REM  Printer on
118  PRINT  "f(x) ="; B(0); "+"; B(1); "x+"; B(2); "xx+"; B(3); "xxx"
120  PRINT
122  PR# 0:  REM  Printer off
124  PRINT  "Continue=any key; Stop=Q";
126  GET  A$
128  IF  A$ = "Q"  THEN  132
130  GOTO  40
132  END
```

```
10   PR# 0:  REM  Printer off
12   HOME
14   DIM  M(3, 3), A(3, 3), B(3, 3), F(3, 3)
16   FOR  I = 0  TO  3
18   FOR  J = 0  TO  3
20   READ  M(I, J):  REM  Define multiplication
22   NEXT  J
24   NEXT  I
26   DATA  0, 0, 0, 0, 0, 1, 2, 3, 0, 2, 3, 1, 0, 3, 1, 2
28   FOR  I = 0  TO 3
30   FOR  J = 0  TO 3
32   READ  A(I, J):  REM  Define addition
34   NEXT  J
36   NEXT  I
38   DATA  0, 1, 2, 3, 1, 0, 3, 2, 2, 3, 0, 1, 3, 2, 1, 0
40   HOME
42   PRINT  "Type in: f(0, 0), f(0, 1), f(0, 2), f(0, 3), f(1, 0), f(1, 1), f(1, 2), f(1, 3), f(2, 0),
     f(2, 1), f(2, 2), f(2, 3), f(3, 0), f(3, 1), f(3, 2), f(3, 3)"
44   PRINT
```

```
46  FOR I = 0 TO 3
48  FOR J = 0 TO 3
50  INPUT F(I, J): REM  Define the logic function
52  NEXT J
54  NEXT I
56  PRINT
58  PRINT "Hit any key to continue";
60  GET A$
62  HOME
64  PR# 1: REM  Printer on
66  PRINT : REM  Display the function table
68  PRINT "(x, y)";"--- >";"f(x, y)"
70  PRINT
72  FOR I = 0 TO 3
74  FOR J = 0 TO 3
76  PRINT "(";I;", ";J;")";"---- >"; F(I, J)
78  NEXT J
80  NEXT I
82  PR# 0: REM  Printer off
84  PRINT
86  REM  The coefficients are determined
88  B(0, 0) = F(0, 0)
90  B(1, 0) = 0: B(0, 1) = 0
92  FOR X = 0 TO 3
94  I = M(X, X)
96  J = M(I, F(X, 0)): K = M(I, F(0, X))
98  B(1, 0) = A(B(1, 0), J): B(0, 1) = A (B(0, 1), K)
100 NEXT X
102 B(2, 0) = 0: B(0, 2) = 0
104 FOR X = 0 TO 3
106 J = M(X, F(X, 0)): K = M (X, F(0, X))
108 B(2, 0) = A(B(2, 0), J): B(0, 2) = A(B(0, 2), K)
110 NEXT X
112 B(3, 0) = 0: B(0, 3) = 0
114 FOR X = 0 TO 3
116 J = F(X, 0): K = F(0, X)
118 B(3, 0) = A(B(3, 0), J): B(0, 3) = A(B(0, 3), K)
120 NEXT X
122 B(1, 1) = 0
124 FOR X = 0 TO 3
126 FOR Y = 0 TO 3
128 I1 = M(X, X): I2 = M(Y, Y)
```

```
130   J1 = M(I1, I2): J2 = M(J1, F(X, Y))
132   B(1, 1) = A(B(1, 1), J2)
134   NEXT Y
136   NEXT X
138   B(1, 2) = 0: B(2, 1) = 0
140   FOR X = 0 TO 3
142   FOR Y = 0 TO 3
144   I1 = M(X, X): I2 = Y: K1 = X: K2 = M(Y, Y)
146   J1 = M(I1, I2): J2 = M(J1, F(X, Y))
148   L1 = M(K1, K2): L2 = M(L1, F(X, Y))
150   B(1, 2) = A(B(1, 2), J2); B(2, 1) = A(B(2, 1), L2)
152   NEXT Y
154   NEXT X
156   B(1, 3) = 0: B(3, 1) = 0
158   FOR X = 0 TO 3
160   FOR Y = 0 TO 3
162   I1 = M(X, X): K2 = M(Y, Y)
164   J1 = M(I1, F(X, Y)): J2 = M(K2, F(X, Y))
166   B(1, 3) = A(B(1, 3), J1): B(3, 1) = A(B(3, 1), J2)
168   NEXT Y
170   NEXT X
172   B(2, 2) = 0
174   FOR X = 0 TO 3
176   FOR Y = 0 TO 3
178   I1 = M(X, Y): I2 = M(I1, F(X, Y))
180   B(2, 2) = A(B(2, 2), I2)
182   NEXT Y
184   NEXT X
186   B(2, 3) = 0: B(3, 2) = 0
188   FOR X = 0 TO 3
190   FOR Y = 0 TO 3
192   I1 = M(X, F(X, Y)): I2 = M(Y, F(X, Y))
194   B(2, 3) = A(B(2, 3), I1): B(3, 2) = A(B(3, 2), I2)
196   NEXT Y
198   NEXT X
200   B(3, 3) = 0
202   FOR X = 0 TO 3
204   FOR Y = 0 TO 3
206   B(3, 3) = A(B(3, 3), F(X, Y))
208   NEXT Y
210   NEXT X
212   REM Output the result in polynomial form
214   PR# 1: REM Printer on
```

```
216  PRINT  "f(x, y) ="; B(0, 0): "+"; B(1, 0): 'x+"; B(0, 1); "y+"; B(2, 0); "xx+"; B(1, 1);
     "xy+"; B(0, 2); "yy+";B(3, 0); "xxx+"; B(2, 1); "xxy+"; B(1, 2); "xyy+"; B(0, 3);
     "yyy+"; B(3, 1); "xxxy+"; B(2, 2); "xxyy+"; B(1, 3); "xyyy+"; B(3, 2); "xxxyy+";
     B(2, 3); "xxyyy+"; B(3, 3); "xxx"
218  PRINT
220  PR# 0: REM  Printer off
222  PRINT  "Continue=any key; Stop=Q";
224  GET A$
226  IF A$ = "Q" THEN 230
228  GOTO 40
230  END
```

b. Logic gates

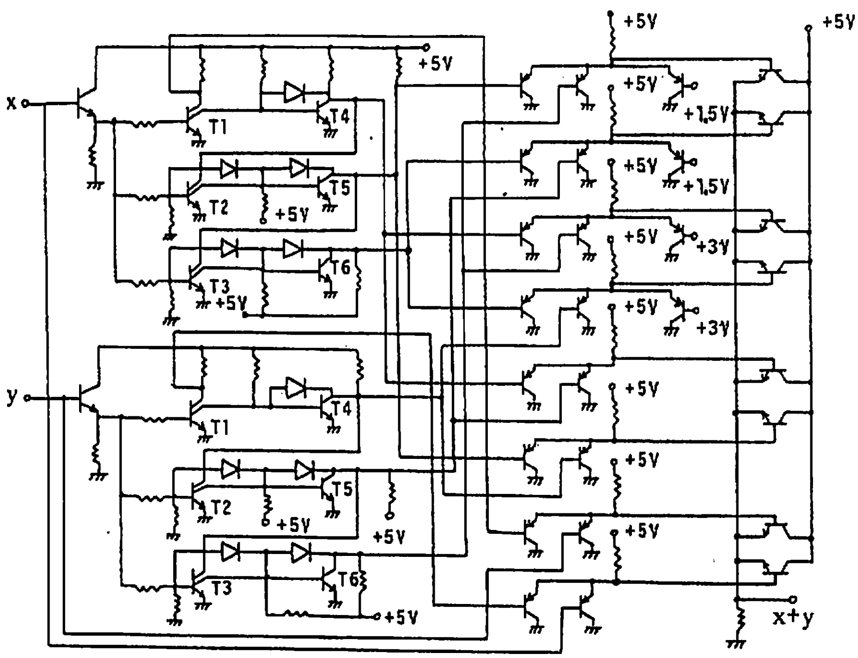

Figure A. 1. (x+y) gate

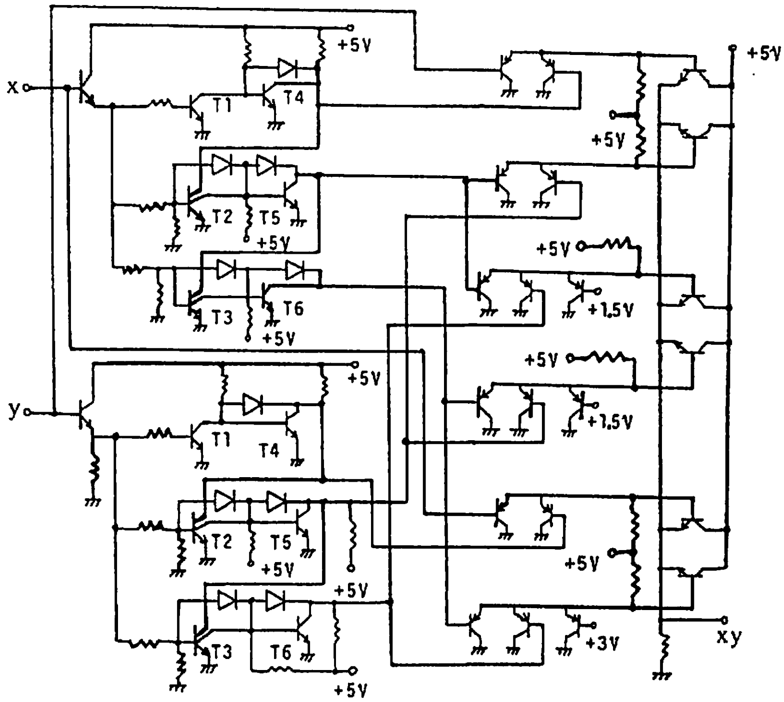

Figure A.2. (xy) gate