AN APPLICATION OF MULTIPLE-VALUED LOGIC TO A DESIGN OF MASTERSLICE GATE ARRAY LSI

Tsutomu SASAO *

Mathematical Science Department
IBM T.J.Watson Research Center
P.O.Box 218, Yorktown Heights, N.Y. 10598
* On sabbatical from Department of Electronic Engineering, Osaka University
Suita 565, Japan

Abstract

MACDAS(Multi-level AND-OR Circuit Design Automation System) is a system for the design of fan-in limited multi-level multi-output circuits: it is used for the design of masterslice LSI of NOR/OR gate arrays. In MACDAS, 1) a given expression is simplified; 2) the input variables are paired to form an expression of four-valued variables; 3) output phases are optimized; 4) the expression is minimized; 5) the expression is factored to solve fan-in limitation problem; 6) the expression is transformed into AND-OR multi-level circuit. Transformation into NOR/OR circuit is done by anothor system. MACDAS has been programmed in FORTRAN and statistical data has been obtained to demonstrate its efficiency in terms of gate counts.

I. Introduction

As integration size of LSI increases, time and cost for the logic design increase rapidly. In the logical design of LSI, we have to obtain a circuit of gate (AND, OR, NOT etc.) level which satisfies desired specification. Now, in most logical design practice, circuits are designed manually and then their errors are detected by logic simulation. No practical automatic design method is known except for the circuits having regular structure such as two-level AND-OR circuits (Fig.1) or PLA's. However, two-level circuits are impractical for complex functions and PLA's are unsuitable for high speed applications.

Gate arrays are widely used for custom LSI design for their low development costs. They are faster than PLA's and can be used for high speed applications. In the gate arrays, each gate has fan-in limitation. For example, if the gates in Fig.2(a) have maximum fan-in 3, 9-input gates must be realized as shown in Fig.2(b). We might design a circuit of Fig.1 and then replace each gate by ones shown in Fig.2(b). But, circuits designed in this way have too many gates. To solve this problem, factoring algorithms have been developled[1]-[2]. However, these methods are still impractical because most circuits designed by these methods (Fig.3) have too many gates compared with manually designed ones.

Design method in this paper use two-variable function generators (TVFG's). A TVFG generates all the function of one and two variables. When we use ECL technology, a TVFG can be realized as shown in Fig.4. Each gate is assumed to realize both NOR and OR outputs.

The design steps of MACDAS are as follows:

1) Pair the input variables to form variables of four-values.
2) Find a near optimal output phase assignment.
3) Minimize the expression of four-valued variables, and derive two-level AND-OR circuit with TVFG's (Fig.5).
4) Factor the expression and derive multi-level AND-OR circuit (Fig.6).
5) Expand TVFG's and replace each gate by NOR/OR gate (Fig.10).

Because MACDAS considers the properties of given function, it often produces better circuits than manually designed ones. MACDAS is an example of practical applications of multiple-valued logic for the design of binary LSI's. It is written in FORTRAN about 10k steps.

II. Definitions and Basic Properties.

In this section, several definitions and basic properties are shown[3].

Definition 2.1: Let $X=(x_1,x_2,\ldots,x_n)$. $X=(X_1,X_2,\ldots,X_r)$ is a partition of X if $\{X_1\}\cup\{X_2\}\cup\ldots\cup\{X_r\}=\{X\}$, where $\{X_i\}\cap\{X_j\}=\phi$ $(i\neq j)$, and $\{X_i\}\neq\phi$.

Definition 2.2: Let $X=(x_1,x_2,\ldots,x_t)$, $\underline{a}\in B^t$, and $B=\{0,1\}$. $X^{\underline{a}}=1$ (if $X=\underline{a}$) and $=0$ (if $X\neq\underline{a}$). Let $S\subseteq B^n$. $X^S=\bigvee_{\underline{a}_i\in S} X^{\underline{a}_i}$. An arbitrary t-variable logic function can be represented by a literal X^S.

Theorem 2.1: Let $X=(X_1,X_2,\ldots,X_r)$ be a partition of $X=(x_1,x_2,\ldots,x_n)$. An arbitrary logic function can be represented by the following expression:

$$f(X_1,X_2,\ldots,X_r)=\bigvee_{(S_1,S_2,\ldots,S_r)} X_1^{S_1}\cdot X_2^{S_2}\cdot\ldots\cdot X_r^{S_r} \quad \text{----(1.1)}$$

where $S_i\subseteq B^{n_i}$ and n_i denotes the number of variables in X_i.

Definition 2.3: Let $P_i=\{0,1,\ldots,p_i-1\}$.
A function $f:\times_{i=1}^{n} P_i \to B$ is called binary function.

Binary function is a generalization of two-valued logic function.

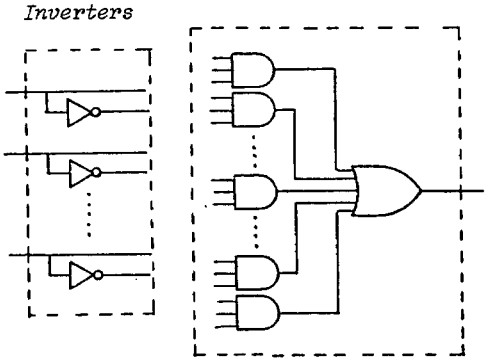


Fig.1 Two-level AND-OR circuit

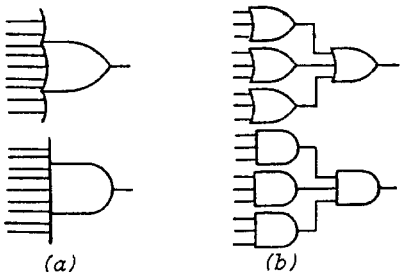


Fig.2 Realization of 9-input gates

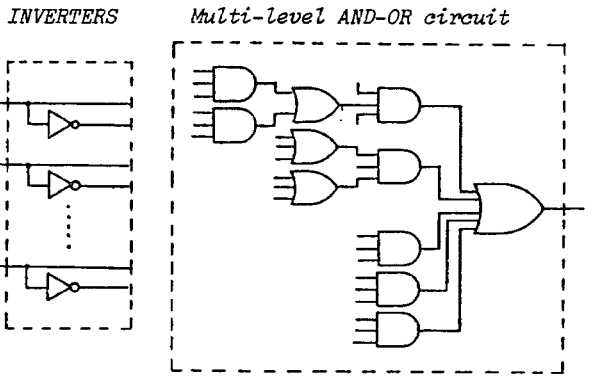
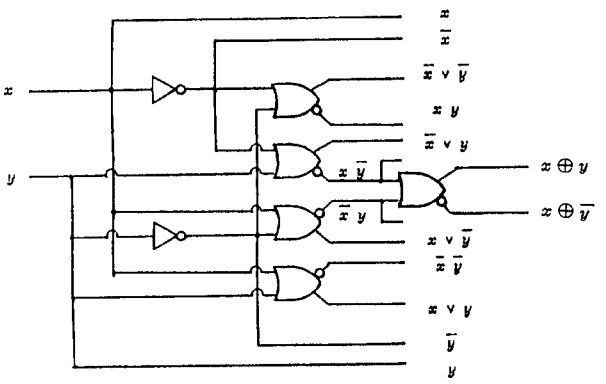
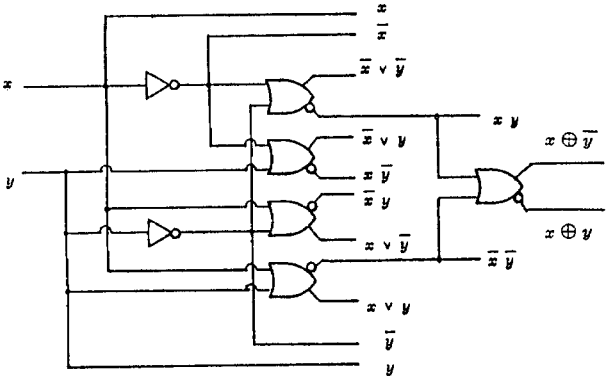


Fig.3 Multi-level AND-OR circuit



(a) Type I TVFG



(b) Type II TVFG

Fig.4 Two-Variable Function Generators

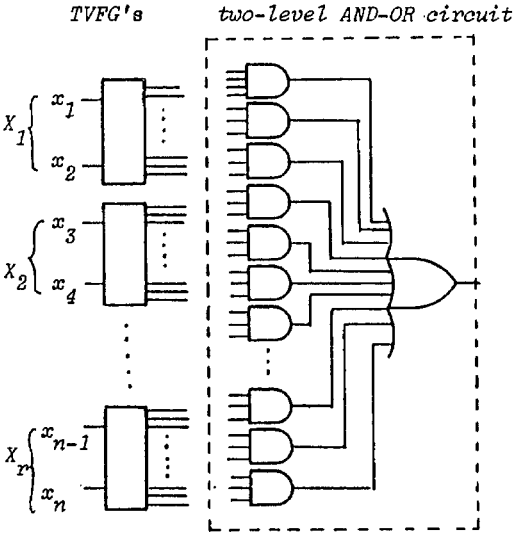


Fig.5 Two-level AND-OR circuit with TVFG's

Theorem 2.2: An arbitrary binary function can be represented by (1.1), where $S_i \subseteq P_i$ and $X_i^{S_i} = 0$ (if $X_i \notin S_i$) and $=1$ (if $X_i \in S_i$).

Binary functions can be represented by positional cubes.

Definition 2.4: A positional cube of a term $X_1^{S_1} \cdot X_2^{S_2} \cdot \ldots \cdot X_n^{S_n}$ is

$$a_0^1 \cdot a_1^1 \ldots a_{p_1-1}^1 \; a_0^2 a_1^2 \ldots a_{p_2-1}^2 \; \ldots \; a_0^n a_1^n \ldots a_{p_n-1}^n ,$$

where $a_j^i = 0$ (if $j \notin S_i$) and $=1$ (if $j \in S_i$).

A term and its cube are often used interchangeably.

Example 2.1: (a) When $p_1=p_2=p_3=2$. (Ordinary logic function).

Product term	Cube
$X_1^0 \cdot X_2^1 \cdot X_3^1$	10-01-01
$X_1^{\{0,1\}} \cdot X_2^0 \cdot X_3^0$	11-10-10
$X_1^{\{0,1\}} \cdot X_2^{\{0,1\}} \cdot X_3^{\{0,1\}} = 1$	11-11-11

(b) When $p_1=2$, $p_2=3$, and $p_3=4$.

Product term	Cube
$X_1^0 \cdot X_2^1 \cdot X_3^2$	10-010-0010
$X_1^{\{0,1\}} \cdot X_2^2 \cdot X_3^{\{0,1,3\}}$	11-001-1101
$X_1^{\{0,1\}} \cdot X_2^{\{0,1,2\}} \cdot X_3^{\{0,1,2,3\}} = 1$	11-111-1111

Theorem 2.3: Let $n=2r$, and $|X_i|=2$. The circuit in Fig.5 realizes a function represented by (1.1).

(Proof) Each TVFG realizes an arbitrary function $X_i^{S_i}$ of two variables(i=1,2,...,r). AND gates realize product terms, and the OR gate realizes logical sum. Hence, Fig.5 realizes (1.1).

(Remark) In MACDAS, each variable X_i consists of two binary variables. The reason for this is as follows.

1) When $|X_i| = 1$, we have conventional two-level AND-OR circuits (Fig.1).
2) When $|X_i| = 2$, the number of distinct non-constant functions is $2^4 - 2 = 14$. We can realize a TVFG by using 7 NOR/OR gates (Fig.4).
3) When $|X_i| = 3$, the number of distinct non-constant functions is $2^8 - 2 = 254$, and we need too many gates to realize a three-variable function generator.

The realization of Fig.5 has the following features:

1) The number of gates and connections are smaller than a conventional two-level circuit (Fig.1).
2) The number of gates can be further reduced by considering the assignment of input variables to the TVFG's.

Theorem 2.4[3]: Let t(f) denote the number of product terms in a minimal sum-of-products expression in (1.1).

For an arbitrary function : $t(f) \leq 2^{n-2}$;
For a function symmetric with respect to$\{X_i\}$(i=1,...,r) : $t(f) \leq 3^{r-1}$;
For a parity function : $t(f) \leq 2^{r-1}$,

where $n=2r$ and $|X_i|=2$.

An s-input AND gate is denoted by s-AND, and an s-input OR gate is denoted by s-OR.

Lemma 2.1: The number of s-AND's to realize an n-AND ($n > s$) is $[(n+s-3)/(s-1)]$, where [x] denotes the integer part of x.

Example 2.2: Consider the number of AND/OR gates necessary to realize a parity function p of n-variables ($n=2r$).

1) When p is realized in the circuit structure of Fig.1.
   n-AND : 2^{n-1} gates .
   2^{n-1}-OR: 1 gate.
2) When each gate is realized by s-AND/OR's in Fig.1.
   s-AND : $2^{n-1} \cdot [\frac{n+s-3}{s-1}]$ gates.
   s-OR : $[\frac{2^{n-1}+s-3}{s-1}]$ gates.
3) When p is realized in the circuit structure of Fig.5.
   r-AND : 2^{r-1} gates.
   2^{r-1}-OR: 1 gate.
4) When each gate is realized by s-AND/OR's in Fig.5.
   s-AND : $2^{r-1} \cdot [\frac{r+s-3}{s-1}]$ gates.
   s-OR : $[\frac{2^{r-1}+s-3}{s-1}]$ gates.

For n=10 and s=3, the above numbers are

1) 10-AND: 512 gates
   512-OR : 1 gate
2) 3-AND: 2560 gates
   3-OR : 256 gates
3) 5-AND: 16 gates
   16-OR : 1 gate
4) 3-AND: 32 gates
   3-OR : 8 gates (End of the example)

From above example, it is clear that circuits using TVFG's require less gates than conventional ones for parity functions. In VI, it will be shown that circuits using TVFG's require less gates for randomly generated functions.

III. Outline of the Design System

In this section, we briefly describe the outline of MACDAS.

Algorithm 3.1: MACDAS

1) Derive truth table from the given specification.
2) Obtain logical expressions of two-valued variables from the truth table. Minimize them.
3) Pair the input variables by considering the properties of the function (see Appendix).
4) Treat each paired variables as a four-valued one, and obtain expressions of four-valued variables.
5) Optimize the output phase assignment (see the next section).
6) Minimize the expressions of four-valued variables [4].

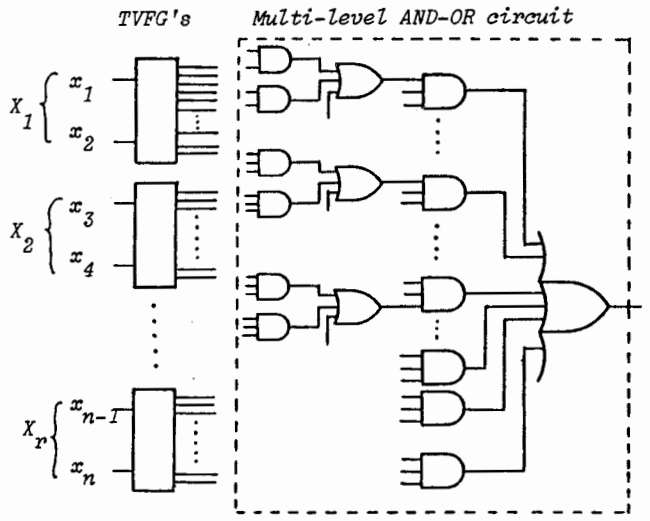


Fig.6 Multi-level AND-OR circuit with TVFG's

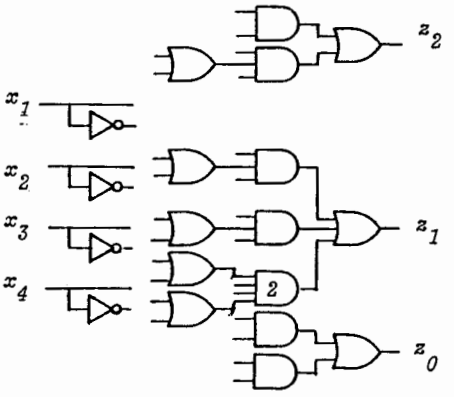


Fig.8 Multi-lvel realization of Two-bit Adder

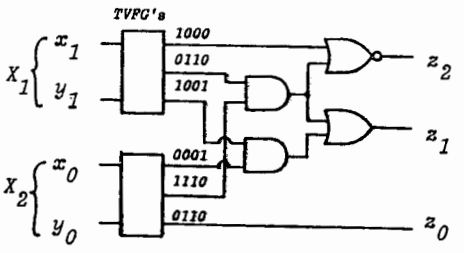


Fig. 9 Two-bit Adder

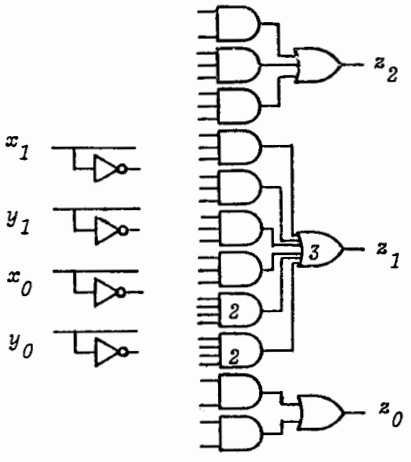


Fig.7 Two-level realization of Two-bit Adder

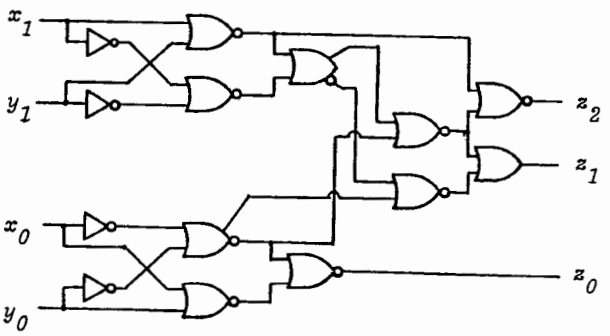


Fig. 10 Two-bit adder by NOR/OR gates.

7) Partition the function by the output.
8) Factor the expressions considering the fan-in of the gates[1]-[2].
9) Realize multi-level AND-OR circuit.
10) Expand TVFG's, convert each gate by NOR/OR gate, and reduce inverters.

* Step 1) and 10) will be done in other systems[6]-[7].

Example 3.1: Let's design a two-bit adder:

$$\begin{array}{rcc} & x_1 & x_0 \\ +) & y_1 & y_0 \\ \hline z_2 & z_1 & z_0 \end{array}$$

1) The truth table for the above adder is shown in Table 1.
2) Simplified expression of two-valued variables are:

$$z_2=x_1^1y_1^1 \vee x_1^1x_0^1y_0^1 \vee x_0^1y_1^1y_0^1 ,$$

$$z_1=x_1^1x_0^0y_1^1 \vee x_1^1y_0^0y_1^1 \vee x_1^0x_0^0y_1^1 \vee x_1^0y_0^0y_1^1 \vee x_1^1x_0^1y_1^1y_0^1 \vee x_1^0x_0^1y_1^0y_0^1 ,$$

$$z_0=x_0^1y_0^0 \vee x_0^0y_0^1 \qquad \text{----(3.1)}$$

Table 2 is the cubical representation of above expressions. Fig.7 shows 3-AND/OR realization of (3.1). It contains 22 gates. By factoring (3.1), we have

$$z_2=x_1^1y_1^1 \vee (x_1^1 \vee y_1^1)(x_0^1y_0^1),$$

$$z_1=(x_1^1y_1^0)(x_0^0 \vee y_0^0) \vee (x_1^0y_1^1)(x_0^0 \vee y_0^0) \vee (x_1^0 \vee y_1^1)(x_1^1 \vee y_1^0)(x_0^1y_0^1),$$

$$z_0=x_0^1y_0^0 \vee x_0^0y_0^1 \qquad \text{----(3.2)}$$

Fig.8 shows the 3-AND/OR realization of (3.2). It contains 20 gates.
3) In the two-bit adder, the output functions are partially symmetric with respect to x_1 and y_1, and x_0 and y_0. So we partition $X=(x_1,x_0,y_1,y_0)$ into $X_1=(x_1,y_1)$ and $X_2=(x_0,y_0)$.
4) By minimizing expressions of four-valued variables, we have

$$z_2=X_1^{11} \vee X_1^{\{01,10,11\}}X_2^{11} ,$$

$$z_1=X_1^{\{01,10\}}X_2^{\{00,01,10\}} \vee X_1^{\{00,11\}}X_2^{11} ,$$

$$z_0=X_2^{\{01,10\}} \qquad \text{----(3.3)}$$

Cubical representation [4] of (3.3) is shown in Table 3.
5) An optimal output phase can be obtained by Algorithm 4.1 (shown in the next section). In this case, it is $X_3=(\bar{z}_2z_1z_0)$. The corresponding cubical representation is shown in Table 4.
6)---- 9) Fig.9 shows the AND/OR realization of this function.
10) By using type II TVFG's, converting each gate by NOR/OR, and reducing inverters, we have the circuit in Fig.10. Note that it contains only 14 gates.

IV. Optimal Output Phase Assignment for Multiple-output Function

When realizing a multiple-output function $(f_0,f_1,\ldots,f_{m-1})$, we have the option to realize either f_i or $\bar{f}_i$ for each output. This is because ECL gates have both NOR and OR outputs[5].

The optimal output phase assignment of multiple-output function is to choose output phases which minimize the number of product terms. In the case of k-output functions, there are 2^k different phase assignments. Because the exhaustive method requires 2^k minimizations, an efficient heuristic method has been desired[4].

This section considers a method which obtains a good solution efficiently.

Definition 4.1: An n-input m-output binary function is defined by

$$f_i : P^n \to B , \quad (i=0,1,\ldots,m-1), \quad P=\{0,1,\ldots,p-1\}.$$

A positive phase characteristic function is difined as

$$F_p : P^n \times M \to B , \text{ where } M=\{0,1,\ldots,2m-1\}, \text{ and}$$

$$F_p(x_1,x_2,\ldots,x_n,j)=f_j(x_1,x_2,\ldots,x_n) \quad (j=0,1,\ldots,m-1)$$
$$=0 \quad (j=m,\ldots,2m-1)$$

A negative phase characteristic function is defined as

$$F_N : P^n \times M \to B , \text{ where}$$

$$F_N(x_1,x_2,\ldots,x_n,j)=0 \quad (j=0,1,\ldots,m-1)$$
$$=\bar{f}_{j-m}(x_1,x_2,..,x_n) \quad (j=m,\ldots,2m-1)$$

A double phase characteristic function is defined as

$$F_D : P^n \times M \to B, \text{ where}$$

$$F_D(x_1,x_2,\ldots,x_n,j)=f_j(x_i,x_2,\ldots,x_n) \quad (j=0,1,\ldots,m-1)$$
$$=\bar{f}_{j-m}(x_1,x_2,..,x_n) \quad (j=m,\ldots,2m-1)$$

Lemma 4.1: F_P, F_N, and F_D can be represented by the following expression:

$$F(x_1,x_2,\ldots,x_n,y) = \bigvee_{(S_1,S_2,\ldots,S_n,R)} x_1^{S_1}\cdot x_2^{S_2}\cdot \ldots \cdot x_n^{S_n}\cdot Y^R ,$$

where $S_i \subseteq P$ and $R \subseteq M$.

Example 4.1: A positive phase characteristic function of two-bit adder in Example 3.1 is

$$F_p = \left\{ \begin{array}{l} 0001\text{-}1111\text{-}100000 \\ 0111\text{-}0001\text{-}100000 \\ 0110\text{-}1110\text{-}010000 \\ 1001\text{-}0001\text{-}010000 \\ 1111\text{-}0110\text{-}001000 \end{array} \right\}$$

A negative phase characteristic function is

$$F_N = \left\{ \begin{array}{l} 1111\text{-}1001\text{-}000001 \\ 1000\text{-}1111\text{-}000100 \\ 0110\text{-}1110\text{-}000100 \\ 0110\text{-}0001\text{-}000010 \\ 1001\text{-}1110\text{-}000010 \end{array} \right\}$$

Table 1 Truth table for two-bit adder

x_1	x_0	y_1	y_0	z_2	z_1	z_0
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	0
0	0	1	1	0	1	1
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	0	0
1	0	0	0	0	1	0
1	0	0	1	0	1	1
1	0	1	0	1	0	0
1	0	1	1	1	0	1
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	1
1	1	1	1	1	1	0

Table 2 Cubical representation of two-bit adder

x_1 x_0	y_1 y_0	z_2	z_1	z_0
01-11-	01-11-1	0	0	
01-01-	11-01-1	0	0	
11-01-	01-01-1	0	0	
01-10-	10-11-0	1	0	
01-11-	10-10-0	1	0	
10-10-	01-11-0	1	0	
10-11-	01-10-0	1	0	
01-01-	01-01-0	1	0	
10-01-	10-01-0	1	0	
11-01-	11-10-0	0	1	
11-10-	11-01-0	0	1	

Table 3 Cubical representation of (3.3)

X_1	X_2	z_2	z_1	z_0
0001-	1111-1	0	0	
0111-	0001-1	0	0	
0110-	1110-0	1	0	
1001-	0001-0	1	0	
1111-	0110-0	0	1	

Table 4 Cubical representation of output phase optimized function.

X_1	X_2	$\bar{z}_2$	z_1	z_0
1000-	1111-1	0	0	
0110-	1110-1	1	0	
1001-	0001-0	1	0	
1111-	0110-0	0	1	

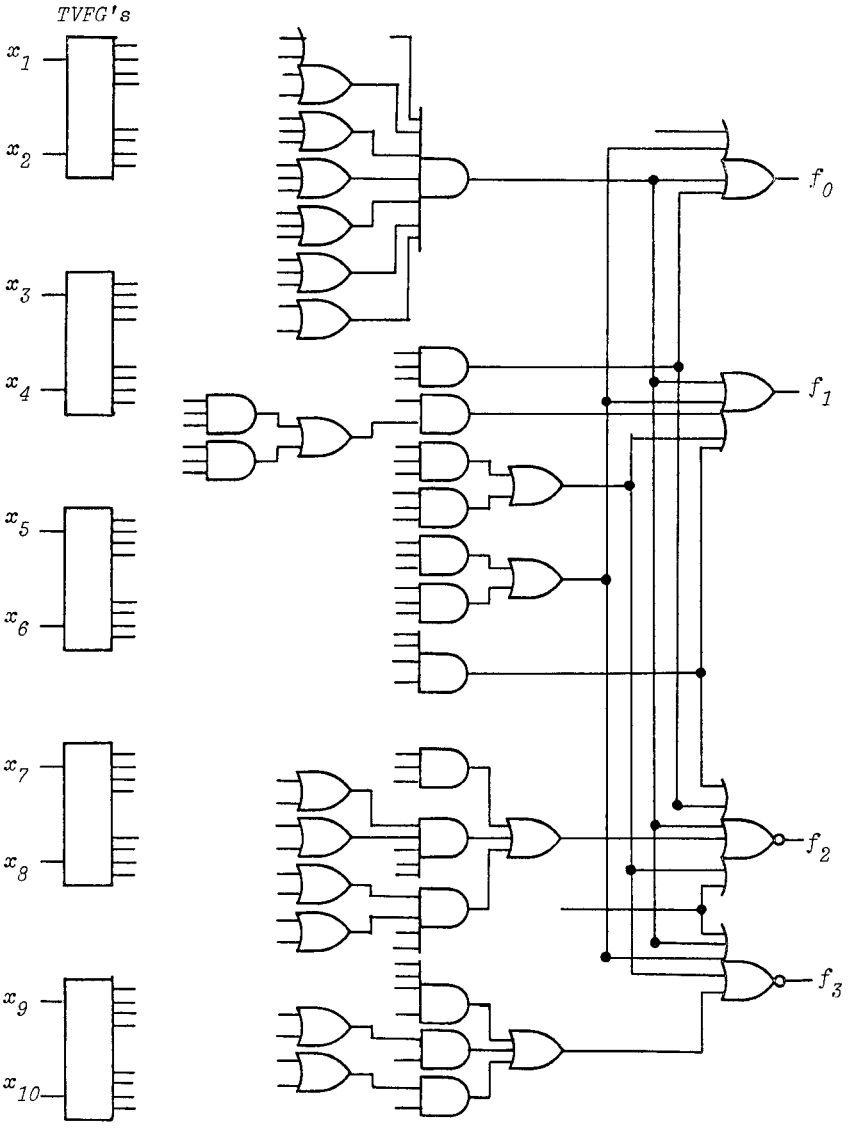


Fig.11 Multilevel AND-OR circuit with TVFG's for ROM pattern

A double phase characteristic function is

$$F_D = F_P \vee F_N .$$

F_D can be minimized to

$$F_D^* = \left\{ \begin{array}{l} 0001\text{-}1111\text{-}100000 \\ 0110\text{-}1110\text{-}010100 \\ 1001\text{-}0001\text{-}010000 \\ 0110\text{-}0001\text{-}100010 \\ 1111\text{-}0110\text{-}001000 \\ 1000\text{-}1111\text{-}000100 \\ 1001\text{-}1110\text{-}000010 \\ 1111\text{-}1001\text{-}000001 \end{array} \right\}$$

Theorem 4.1[3]: To minimize the number of distinct product terms of the expression for the functions $(f_0,f_1,\ldots,f_{m-1})$, it is sufficient to obtain a minimal sum-of-products expression for F_p.

Definition 4.2: Let F_D^* be a minimal sum-of-products expression for F_D. A connection matrix $G = \{g_{ij}\}$ for F_D^* is the output part of F_D^*. g_{ij} =1 iff j-th output of i-th cube is one.

Example 4.2: The connection matrix for F_D^* of Example 4.1 is

$$G = \left\{ \begin{array}{l} 100000 \\ 010100 \\ 010000 \\ 100010 \\ 001000 \\ 000100 \\ 000010 \\ 000001 \end{array} \right\} \begin{array}{l} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \\ p_7 \\ p_8 \end{array}$$

Definition 4.3: Let G be a connection matrix. A covering function Q of $G = \{g_{ij}\}$ is

$$Q(p_1,p_2,\ldots,p_t) =$$

$$\bigwedge_{j=0}^{m-1} \left[\{ \bigwedge_{i=0}^{t} (p_i \vee \bar{g}_{ij}) \} \vee \{ \bigwedge_{i=0}^{t} (p_i \vee \bar{g}_{i,j+m}) \} \right] .$$

An assignment vector of a product $p_{a_1}p_{a_2}\ldots p_{a_s}$ is defined as

$$g = \bigvee_i (g_{i0}, g_{i1}, \cdots, g_{i,(2m-1)}) ,$$

where $i=a_1, a_2,\ldots, a_s$.

Example 4.3: The covering function of G in Example 4.2 is

$$Q(p_1,p_2,\ldots,p_8)$$
$$=(p_1p_4 \vee p_2p_6)\cdot(p_2p_3 \vee p_4p_7)\cdot(p_5 \vee p_8)$$
$$=p_1p_2p_3p_4p_5 \vee p_1p_2p_3p_4p_8 \vee p_2p_3p_5p_6 \vee p_2p_3p_6p_8$$
$$\vee p_1p_4p_5p_7 \vee p_1p_4p_7p_8 \vee p_2p_4p_5p_6p_7 \vee p_2p_4p_6p_7p_8.$$

Algorithm 4.1: Near optimal output phase assignment for F.

1) Obtain the double phase characteristic function F_D, and minimize it.
2) Obtain a covering function Q, and expand it into product terms.
3) Find a product term $p_{a_1}p_{a_2}\ldots p_{a_s}$, which has (near) minimal number of letters.
4) Obtain the output assignment vector $g=(g_0,g_1,\ldots,g_{2m-1})$ for the product obtained in 3).
5) Obtain the output assignment $F^*=(f_0^*,f_1^*,\ldots,f_{m-1}^*)$, where

$$f_i^* = \left\{ \begin{array}{ll} f_i & (\text{if } g_i=1) \\ \bar{f}_i & (\text{if } g_i=0) \end{array} \right. \quad (i=0,1,\ldots,m-1)$$

Theorem 4.2: Multiple-output function $F^* = (f_0^*,f_1^*,\ldots,f_{m-1}^*)$ can be realized at most s product terms, where F^* and s are obtained in Algorithm 4.1.

Example 4.4: In Example 4.3, the product term $p_2p_3p_5p_6$ has 4 letters and it is minimal. Subset of F_D^* corresponding to $p_2p_3p_5p_6$ is

$$C = \left\{ \begin{array}{l} 0110\text{-}1110\text{-}010100 \\ 1001\text{-}0001\text{-}010000 \\ 1111\text{-}0110\text{-}001000 \\ 1000\text{-}1111\text{-}000100 \end{array} \right\}$$

So F can be realized at most 4 terms and the assignment vector for it is represented by

$$g = (0,1,0,1,0,0) \vee (0,1,0,0,0,0) \vee (0,0,1,0,0,0) \vee (0,0,0,1,0,0)$$
$$=(0,1,1,1,0,0)$$

Obtained output phase assignment is $(\bar{f}_0f_1f_2)$.

For the function which has don't cares, the step 1 of Algorithm 4.1 should be modified as follows:

1) Obtain double phase characteristic function F_D, and double phase don't care characteristic function H_D (Definition 4.4). Minimize F_D by using H_D.

Definition 4.4: Let an n-input m-output function $h_i : B^n \to B$ $(i=0,1,\ldots,m-1)$ denote unspecified part of the function: i.e., i-th function is undefined iff h_i=1.

A don't care characteristic function is defined as

$H : B^n \times M \to B$, where

$$H(x_1,x_2,\ldots,x_n,j)=h_j(x_1,x_2,\ldots,x_n) \quad (j=0,1,\ldots,m-1)$$
$$=0 \quad (j=m,\ldots,2m-1)$$

A double phase don't care characteristic function is defined as

$H_D : B^n \times M \to B$, where

$$H_D(x_1,x_2,\ldots,x_n,j)= h_j(x_1,x_2,\ldots,x_n) \quad (j=0,1,\ldots,m-1)$$
$$=h_{j-m}(x_1,x_2,\ldots,x_n) \quad (j=m,\ldots,2m-1)$$

Example 4.5: Consider the following 3-input 3-output function with don't cares:

$$F = \left\{ \begin{array}{l} 10\text{-}10\text{-}10\text{-}100 \\ 10\text{-}10\text{-}01\text{-}100 \\ 10\text{-}01\text{-}10\text{-}011 \\ 10\text{-}01\text{-}01\text{-}101 \\ 01\text{-}10\text{-}10\text{-}011 \\ 01\text{-}10\text{-}01\text{-}110 \\ 01\text{-}01\text{-}10\text{-}001 \end{array} \right\} \qquad DC = \left\{ \begin{array}{l} 10\text{-}10\text{-}01\text{-}011 \\ 10\text{-}01\text{-}01\text{-}010 \\ 01\text{-}10\text{-}10\text{-}100 \\ 01\text{-}10\text{-}01\text{-}001 \\ 01\text{-}01\text{-}01\text{-}111 \end{array} \right\}$$

Table 5. Number of products and gates to realize various functions *

Source (in/out/product)		Ordinary circuits		Circuits using TVFG's	
		Output phase non-optimized (Fig.1/Fig.3)	Output phase optimized (Fig.1/Fig.3)	Output phase non-optimized (Fig.5/Fig.6)	Output phase optimized (Fig.5/Fig.6)
ROM pattern (10/4/511)	products	58	37	38	27
	gates	(226/132)	(176/114)	(87/73)	(50/48)
ROM pattern (10/4/1024)	products	16	7	11	7
	gates	(34/23)	(21/17)	(16/13)	(16/13)
3-bit multiplier (6/6/49)	products	31	31	21	19
	gates	(83/77)	(78/74)	(35/36)	(31/35)
2-bit adder (4/3/15)	products	11	9	5	4
	gates	(18/16)	(16/14)	(5/5)	(4/4)
4-bit adder (8/5/255)	products	75	61	17	14
	gates	(191/90)	(158/76)	(31/26)	(24/21)

* These values do not count buffers, inverters, nor gates for TVFG's. Maximum fan-in of each gate is 3.

Table 6 Number of products and gates for randomly generated functions. *

Source (in/out/product)		Ordinary circuts (Fig.1/Fig.3)	Circuits using TVFG's (Fig.5 /Fig.6)
(10/1/128)	products	91	77
	gates	(447/266)	(192/152)
(10/1/256)	products	132	104
	gates	(615/357)	(260/215)
(10/1/348)	products	153	123
	gates	(688/392)	(307/240)
(10/1/512)	products	161	123
	gates	(693/415)	(307/252)
(10/1/640)	products	163	121
	gates	(647/383)	(302/260)
(8/4/128)	products	88	77
	gates	(358/259)	(194/149)
(8/4/256)	products	125	106
	gates	(477/357)	(265/214)
(8/4/384)	products	143	123
	gates	(519/391)	(305/252)

* These values do not count buffers, inverters, nor gates for TVFG's. Maximum fan-in of each gate is 3.

1) The double phase characteristic functions are:

$$F_D = \left\{\begin{array}{l} 10\text{-}10\text{-}10\text{-}100000 \\ 10\text{-}10\text{-}01\text{-}100000 \\ 10\text{-}01\text{-}10\text{-}011000 \\ 10\text{-}01\text{-}01\text{-}101000 \\ 01\text{-}10\text{-}10\text{-}011000 \\ 01\text{-}10\text{-}01\text{-}110000 \\ 01\text{-}01\text{-}10\text{-}001000 \\ 10\text{-}01\text{-}10\text{-}000100 \\ 10\text{-}10\text{-}10\text{-}000011 \\ 10\text{-}10\text{-}01\text{-}000011 \\ 01\text{-}10\text{-}10\text{-}000100 \\ 01\text{-}10\text{-}01\text{-}000001 \\ 01\text{-}01\text{-}01\text{-}000111 \\ 01\text{-}01\text{-}10\text{-}000110 \end{array}\right\} \quad H_D = \left\{\begin{array}{l} 10\text{-}10\text{-}01\text{-}011011 \\ 10\text{-}01\text{-}01\text{-}010010 \\ 01\text{-}10\text{-}10\text{-}100100 \\ 01\text{-}10\text{-}01\text{-}001001 \\ 01\text{-}01\text{-}01\text{-}111111 \end{array}\right\}$$

2) F_D is minimized to F_D^*.

$$F_D^* = \left\{\begin{array}{l} 11\text{-}11\text{-}01\text{-}101000 \\ 10\text{-}10\text{-}11\text{-}100011 \\ 01\text{-}10\text{-}11\text{-}011000 \\ 10\text{-}01\text{-}10\text{-}011100 \\ 01\text{-}01\text{-}11\text{-}001110 \end{array}\right\}$$

3) The connection matrix for F_D^* is

$$G = \left\{\begin{array}{l} 101000 \\ 100011 \\ 011000 \\ 011100 \\ 001110 \end{array}\right\} \begin{array}{l} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{array}$$

4) The covering function Q is
$Q(p_1, p_2, \ldots, p_5)$

$$= (p_1p_2 \vee p_4p_5) \cdot (p_3p_4 \vee p_2p_5) \cdot (p_1p_3p_4p_5 \vee p_2)$$

$$= p_1p_2p_3p_4p_5 \vee p_1p_2p_3p_4 \vee p_1p_2p_3p_4p_5 \vee p_1p_2p_5 \vee p_1p_3p_4p_5 \vee p_2p_3p_4p_5 \vee p_1p_2p_3p_4p_5 \vee p_2p_4p_5 .$$

5) The product $p_1p_2p_5$ has three letters. Subset of F_D^* corresponding to $p_1p_2p_5$ is

$$C = \left\{\begin{array}{l} 11\text{-}11\text{-}01\text{-}101000 \\ 10\text{-}10\text{-}11\text{-}100011 \\ 01\text{-}01\text{-}11\text{-}001110 \end{array}\right\}$$

Obtained output phase is $(f_0\bar{f}_1\bar{f}_2)$.

V. Experimental Results

Table 5 shows the selected results for several practical functions. For each functions, four different realizations (Fig.1, Fig.3, Fig.5, and Fig.6) are compared. Maximum fan-in of each gate is assumed to 3.Each value does not count the number of buffers, inverters, nor gates for TVFG's.

Fig.11 shows the realization (type Fig.6) for the first function of Table 5. Manual design required 110 gates to realize this function. It took 56.4 sec. to minimize an expression of 511 terms into 58 terms, 1.1 sec. to assign the input variables to the decoders, 23 sec. to assign output phase and minimize it into 27 terms, 13 sec. to realize AND-OR multi-level circuit of 48 gates.

Besides the functions of Table 5, many practical functions were examimed. In the case of practical functions with many inputs and outputs, our program failed to obtain better output phase assignments than trivial ones.

Table 6 shows the results for randomly generated functions. In the case of 4-output functions, all the (near) optimal output phase assignments were trivial.

VI. Observation

Table 6 shows that even if the functions have no special properties, circuits using TVFG's require far less gates than ordinary ones. For example, to realize the first function of Table 6, Fig.3 realization requires 266 gates and 10 invertors; whereas Fig.6 realization requires only 152 gates and 7x5 =35 gates for TVFG's. We can save (266+10)-(152+35) =89 gates by using TVFG's.

The reason why the circuits with TVFG's require far less gates even for randomly generated functions can be considered as follows:

1) In the realization of Fig.3, fan-outs exist only in the inputs and outputs of inverters; whereas in the realization of Fig.6, fan-outs exist both inside and outputs of TVFG's.
2) By using TVFG's, ECL gates display OR and NOR output capability as shown in Fig.10. (No systematic design method for ECL logic circuits have been known except for the integer programming method[8]).
3) When the input variables are paired, the number of terms necessary to represent the function decreases about 10-30% for randomly generated functions[3].
4) By using TVFG's, we can reduce fan-in of the gates inside the broken line of Fig.6.

VII. Conclusion

Main results obtained are as follows:

1) Circuits with TVFG's usually require less gates than conventional ones.
2) Circuits with TVFG's can be designed by using binary functions of 4-valued variables.
3) Output phase optimized circuits often require less gates than non-optimized ones.
4) Near optimal output phase assignments can be obtained by double phase characteristic functions.
5) MACDAS realizes good circuits. It often produces circuits having less gates than manually designed ones.

Acknowledgment

The author wish to thank Prof. H.Terada for his encouragement and Mr. K. Ishikawa for his discussion and programming.

Rererences

[1] S.Y.H.Su and C-W Nam,"Computer-aided synthesis of multiple-output multilevel NAND networks with fan-in and fan-out constraints," IEEE Trans. on Comput., vol.C-20, No.12, pp.1445-1455, Dec. 1971.

[2] D.L.Dietmeyer, Logical Design of Digital Systems, Allyn and Bacon, 1971.

[3] T. Sasao, "Multiple-valued decomposition of generalized Boolean functions and the complexity of programmable logic arrays," IEEE Trans. on Comput., vol.C-30, No.9, pp.635-643, Sept. 1981.

[4] S.J.Hong.R.G.Cain, and D.L.Ostapko, "MINI: A heuristic approach for logic minimization," IBM J.Res. Deveolp., vol.18,pp.443-458, Sept. 1974.

[5] R.J.Blumberg and S.Brenner, "A 1500 gate, random logic, large scale integration (LSI) master-slice," IEEE J. Solid-State Circuits, vol.SC-14, No.5, pp.818-822, Oct. 1979.

[6] N.Kawato, T.Saito, F. Maruyama, and T.Uehara, "Design and verification of large scale computers by using DDL," 16th Design Automation Conference, pp.375-381, June 1979.

[7] S. Nakamura et. al.,"LORES-Logic reorganization system," 15th Design Automation Conference, pp. 250-260, June 1978.

[8] C.R.Baugh, C.S.Chandersekaran, R.S.Swee, and S. Muroga,"Optimal networks of NOR-OR gates for functions of three variables," IEEE Trans. on Comput., vol.C-21, No.2,pp.153-160, Feb.1972.

[9] K.Ishikawa, T. Sasao, and H.Terada, "An assignment method for programmable logic arrays with decoders, " (in Japanese) Trans. IECE Japan (to be published).

Appendix

In this appendix, we briefly describe the details of the algorithms. We assume that n=2r.

AP.1 Near Optimal Assignment of Input Variables to the TVFG's.

Definition AP.1: Let I={1,2,...,n} be the set of subscripts of variables in {X}. The partition of I which corresponds to the partition of {X} is denoted by Π. The number of terms in a minimal sum-of-products expression for f(X) under the partition Π is denoted by $t(f:\Pi)$.

Definition AP.2: Let an expression which represents $f(x_1,x_2,\ldots,x_n)$ be P. The number of distinct terms which are obtained by deleting literals of x_i and x_j from P is denoted by $q(i,j)$.

Example AP.1: Let

$$f= x_1^0x_2^0x_3^1x_4^1 \vee x_1^0x_2^1x_3^1x_4^0 \vee x_1^1x_2^0x_3^0x_4^1 \vee x_1^1x_2^0x_3^1x_4^0 \vee x_1^1x_2^1x_3^0x_4^0 .$$

The terms which are obtained by deleting the literals of x_3 and x_4 are

$$x_1^0x_2^0 , \ x_1^0x_2^1 , \ x_1^1x_2^0 , \ x_1^1x_2^0 , \ x_1^1x_2^1 .$$

The number of distinct terms is 4. So, we have $q(3,4)=4$. Similarly, we have $q(1,3)=q(2,4)=3$, $q(1,2)=q(1,4)=q(2,3)=4$.

Lemma AP.2: Let Π_{ij}= {[1],[2],...,[i,j],..,[n]}.

$$t(f: \Pi_{ij}) \le q(i,j) \le 2^{n-2}.$$

$t(f: \Pi_{ij})$ denotes the number of terms in a minimal sum-of-products expression when x_i and x_j are paired to form a 4-valued variables.

The smaller $t(f:\Pi_{ij})$, the simpler the expression for f becomes. Because it takes much computation time to obtain $t(f: \Pi_{ij})$, we use an upper bound $q(i,j)$ instead.

Definition AP.3: An assignment graph for an n-variable function $f(x_1,x_2,\ldots,x_n)$ is a complete graph satisfying the following conditions:

1) G has n nodes (n=2r).
2) The weight of the edge (i,j) is q(i,j).

Algorithm AP.1:

1) Obtain a near minimal sum-of-products expression for f.
2) Obtain the assignment graph for f.
3) Cover every node by disjoint edges so as to minimize the sum of the weights of the edges.
4) Obtain the partition of the variables corresponding to the edges.

Example AP.2: Consider the function in AP.1.

1) Given expression is minimal.
2) Fig.AP.1 shows the assignment graph for the function of Example AP.1.
3) Edges (1,3) and (2,4) cover all the nodes of G. The sum of the weights is 3+3=6 and is the minimum.
4) The partition of X is $X=(X_1,X_2)$, where $X_1=(x_1,x_3)$ and $X_2=(x_2,x_4)$.

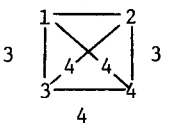


Fig.AP.1

Although Algorithm AP.1 is quite simple, it obtains good assinments which are on the average 10% better than trivial ones[9]. In Algorithm AP.1, most time are spent for obtaining a near minimal sum-of-products expressions: other time is relatively short.