MACDAS: Multi-level AND-OR Circuit Synthesis using Two-Variable Function Generators

Tsutomu SASAO

Department of Electronic Engineering
Osaka University
Suita 565, Japan

Abstract: MACDAS(Multi-level AND-OR Circuit Design Automation System) designs a multi-level circuit with fan-in limited AND-OR gates. In MACDAS, a given specification is converted into an AND-OR two-level circuit; input variables are paired to produce an AND-OR two-level circuit with two-variable function generators; some of the outputs are complemented to obtain a circuit with fewer AND gates; the circuit is transformed into a multi-level fan-in limited AND-OR circuit; and finally the circuit is optimized by local transformations. MACDAS has been programmed in FORTRAN and C, and runs on a personal computer. Both arithmetic and control circuits are designed to show the performance of MACDAS.

I. Introduction

Over the years, a number of automatic synthesis system have been developed. Effective programs exist to go from high-level register-transfer specification to PLA's. Logic minimizers such as ESPRESSO-II can now treat standard PLA's with hundred inputs and outputs[BRA 84a].

For random logic synthesis, at least two different methods are known. The first one is a rule based local transformation [DAR 84]. The second one is an algebraic method for optimizing multi-level logic[DIE 78]. Brayton and McMullen developed algebraic tools for designing multi-level networks called 'weak division' and 'strong division', and successfully applied them to design single-ended cascode voltage switch circuits[BRA 84b]. Bartlett and Hachtel used weak division to design gate array or standard cell library implementation[BAR 85]. de Geus and Cohen made a system which first synthesizes a multi-level circuit by weak division and then optimizes it in the target technology by using a rule-based expert system[DEG 85]. The circuit generated by their system are comparable in area and speed to ones designed by experts.

In this paper, a method of optimizing a multi-level combinational circuit targeted towards a gate array is presented. In the gate array, each gate has a fan-in limitation. The straightforward design method is, first to derive a two-level circuit by using PLA minimizers, and then replace each gate by ones with fan-in limitations. But circuit designed in this way have too many gates. To solve this problem, factoring algorithms have been developed [DIE 78]. Unfortunately, these methods are still impractical because most circuits designed by these methods have too many gates compared with manually designed ones.

The design method in this paper use TVFG's(Two-Variable Function Generators). The TVFG generates all functions of one and two-variables. The design steps of MACDAS[SAS 82] are as follows:
1) The specification of the circuit is given by a truth table, a netlist of the circuit diagram, or an arithmetic expression.
2) It is converted into an AND-OR two-level circuit.
3) The input variables are partitioned into pairs to produce an AND-OR two-level circuit with TVFG's.
4) Some of the outputs are complemented to obtain a circuit with fewer AND gates.
5) The circuit is converted into a multi-level fan-in limited AND-OR circuit.
6) Finally, the circuit is optimized by local transformations.

MACDAS uses two recently developed PLA optimization techniques. The first one is the optimal assignment of the input variables to PLA's with two-bit decoders. Optimally designed PLA's with two-bit decoders are, on the average, 20 to 30 percent smaller than the standard PLA's[SAS 81]. The second one is the optimal selection of the output phases. For some functions, the complement are easier to realize than the given functions. We can choose for each output, whether to implement the function or its complement, and obtain correct output function by properly adding inverters. The author have shown that the optimized arithmetic PLA's are 5 to 10 percent smaller than output phase trivial ones [SAS 84]. By using these optimization techniques, MACDAS first obtains a two-level circuit with at least 30 percent fewer products than standard minimization technique.

Because MACDAS use these optimization techniques of PLA's as well as local transfomations, it often produces better circuits than manually designed ones. It is written in FORTRAN and C, and runs on a personal computer using MS-DOS.

II. Logic Design using TVFG's

In this section, the author propose a design method using TVFG's. In this method, the input variables are partitioned into pairs. Each pair represents a super variable which takes four values 00,01,10, or 11. For a super variable X, a literal of X is defined as follows:

$X^S = 0$ (if $X \notin S$), and $X^S = 1$ (if $X \in S$),
where $S \subseteq (00,01,10,11)$.

There are $2^4=16$ different literals , of which 14 literals are non-trivial. By using these literals, we can represent an arbitrary two-valued logic function.

Theorem 2.1: An arbitrary two-valued logic function $f(x_1,x_2,\ldots,x_n)$ $(n=2r)$ can be represented by the following expression:

$$f(X_1,X_2,\ldots,X_r) = \bigvee_{(S_1,S_2,\ldots,S_r)} X_1^{S_1} \cdot X_2^{S_2} \cdot \ldots \cdot X_r^{S_r} , \qquad \text{-----(2.1)}$$

where $X_i=(x_{2i-1},x_{2i})$, $S_i \subseteq (00,01,10,11)$, and $i=1,2,\ldots,r$.

Fig.2.1 is a TVFG which generates all the non-trivial literals. By using TVFG's, we can realize a two-level AND-OR circuit shown in Fig.2.2.

Theorem 2.2: A two-level circuit with TVFG's (Fig.2.2) realizes a sum-of-products expression having a form (2.1).

Now consider the design of the circuit with TVFG's. In gate arrays, each connection usually has less cost than a gate. Therefore, we define the minimum two-level AND-OR circuit with TVFG's as follows:

0738-100X/86/0000/0086$01.00 ©1986 IEEE

Definition 2.1: An AND-OR two-level circuit with TVFG's is said to be minimum if the following conditions hold.
1) The number of the AND gates is minimum.
2) The number of connections is minimum in the condition that 1) is satisfied.

Because the number of products in (2.1) is equal to the number of AND gates, and the number of non-trivial literals in each product is equal to the number of inputs of each AND gate, we can define the minimum sum-of-products expression as follows:

Definition 2.2: A sum-of-products expression (2.1) is said to be minimum if the following conditions hold.
1) The number of products in (2.1) is minimum.
2) The total number of the non-trivial literals in (2.1) is minimum in the condition that 1) is satisfied.

Hence, the minimization of a two-level AND-OR circuit with TVFG's is reduced to the minimization of the expression of form (2.1). Minimization of the expression can be done by a multiple-valued minimizer such as MINIMON 74], MINI-II[SAS 84], and ESPRESSO-MV[RUD 85].

Theorem 2.1 and 2.2 are natural extension of ordinary two-valued switching theory. In two-valued case, there exist $2^2=4$ literals $(0,1,\bar{x},x)$, where x and $\bar{x}$ are non-trivial. In this paper, the circuit which generates x and $\bar{x}$ is called OVFG (One-variable Function Generator).

The AND-OR two-level circuit with TVFG's have the following features:
1) The number of AND gates is smaller than ones with OVFG's. For example, the number of AND gates is, on the average, 32 percent smaller than conventional ones for randomly generated functions of 8 variables with 102 minterms[SAS 81].
2) The number of connections to AND gates is smaller than ones with OVFG's. Because the number of the super variables is a half of the number of the ordinary variables, the maximum number of inputs to each AND gate is at most r=n/2.

When the circuit with TVFG's still has fan-in problem, the circuit is transformed into a multi-level AND-OR circuit using a factoring algorithm.

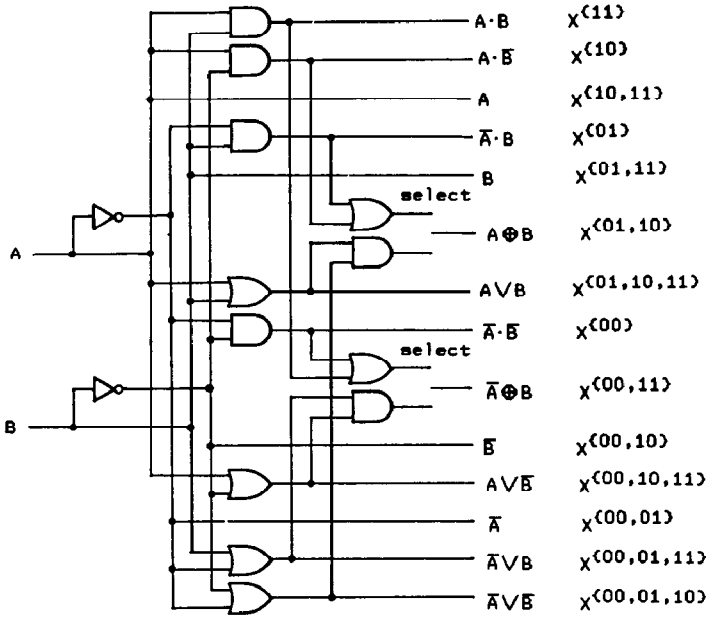


Fig.2.1 Two-Variable Function Generator

Example 2.1: Let us design a two-bit adder using 3-input AND and OR gates.

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

Design Method Using OVFG's (Conventional method):

1) Simplified expression for two-bit adder using the literals of OVFG's are obtained by the maps in Fig.2.4 as follows:

$$z_0=x_0\bar{y}_0 \vee \bar{x}_0y_0 \quad ,$$

$$z_1=x_1\bar{x}_0\bar{y}_1 \vee x_1\bar{y}_0\bar{y}_1 \vee \bar{x}_1\bar{x}_0y_1 \vee \bar{x}_1\bar{y}_0y_1 \vee x_1x_0y_1y_0 \vee \bar{x}_1x_0\bar{y}_1y_0,$$

$$z_2=x_1y_1 \vee x_1x_0y_0 \vee x_0y_1y_0 . \qquad \text{----(2.2)}$$

Fig.2.5(a) shows 3-input AND-OR realization of (2.2), where the numbers in the gates show the numbers of 3-input gates necessary to implement the 4-input or 6-input gates. Note that 22 gates are necessary to realize this circuit.

2) After factoring the expressions, we have

$$z_0=x_0\bar{y}_0 \vee \bar{x}_0y_0,$$

$$z_1=(x_1\bar{y}_1)\cdot(\bar{x}_0\vee\bar{y}_0)\vee(\bar{x}_1y_1)(\bar{x}_0\vee\bar{y}_0) \vee(\bar{x}_1\vee y_1)\cdot(x_1\vee\bar{y}_1)\cdot(x_0\cdot y_0),$$

$$z_2=x_1y_1\vee(x_1\vee y_1)\cdot(x_0y_0), \qquad \text{-----(2.3)}$$

Fig.2.5(b) shows the multi-level realization of (2.3). Note that 20 gates are used in this circuit.

Design method using TVFG's (The proposed method)

1) In the two-bit adder, the output functions are symmetric with respect to (x_1,y_1) and (x_0,y_0). So, we make super variables $X_1=(x_1,y_1)$ and $X_2=(x_0,y_0)$. MACDAS finds this pairng. Simplified expressions using the literals for these super variables are obtained by the maps in Fig.2.6 as follows:

$$z_0=X_2^{(01,10)} \quad ,$$

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

$$z_2=X_1^{(11)}\vee X_1^{(01,10,11)}\cdot X_2^{(11)}.$$

Note that in the multiple-valued logic, we have more possibility of making larger prime implicants than two-value logic. This is why the minimized multiple-valued expressions usually contain fewer products than two-valued ones.

2) An optimal output phase can be obtained by using a method shown in [SAS 84]. MACDAS finds a near optimal output phase. In this case, it is $(\bar{z}_2z_1z_0)$. In other words, the complement of z_2 is

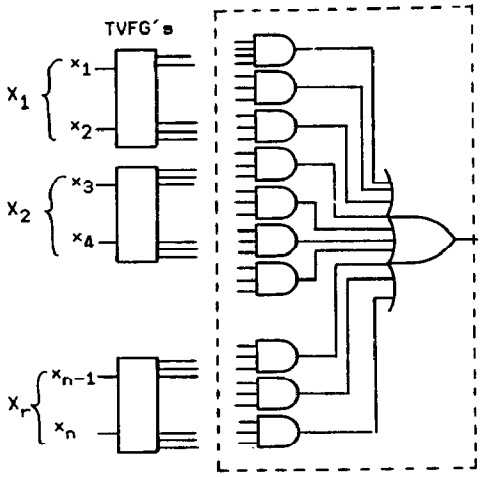


Fig.2.2 Two-level AND-OR circuit with TVFG's

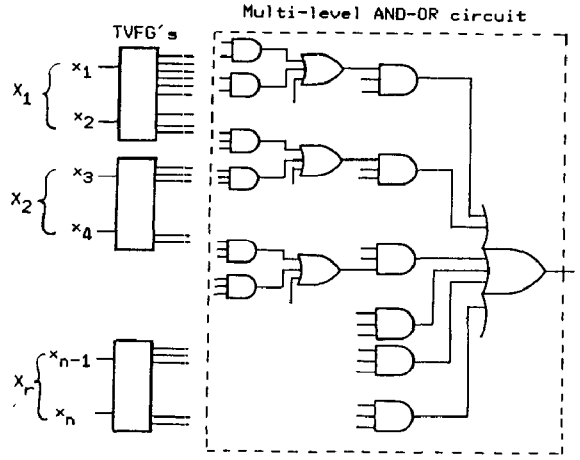


Fig.2.3 Multi-level AND-OR circuit with TVFG's

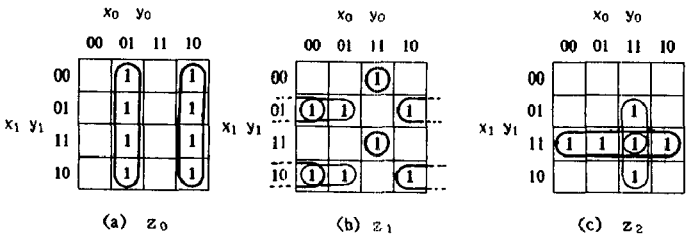


Fig.2.4 Maps for Two-bit Adder using OVFG's

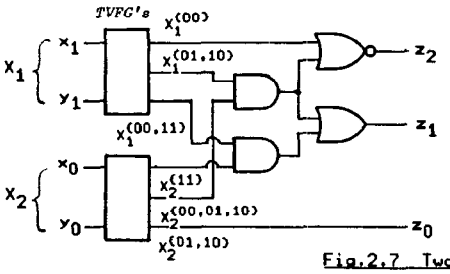


Fig.2.7 Two-bit adder using TVFG's

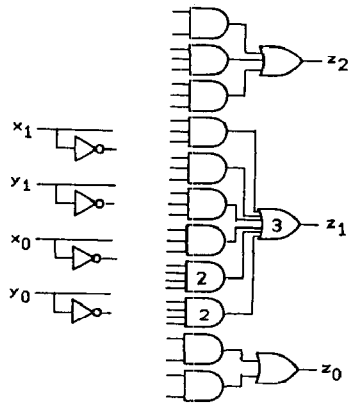


Fig.2.5(a)

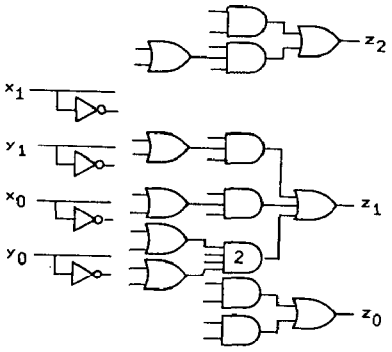


Fig.2.5(b)

Fig.2.5 Two-bit adder by AND-OR circuit with OVFG's.

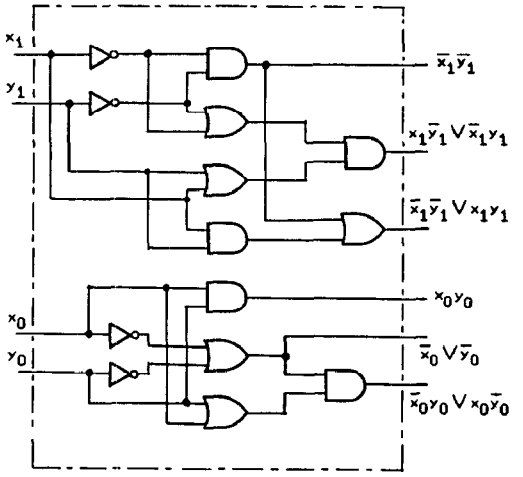


Fig.2.8 TVFG's after Macro Expansion

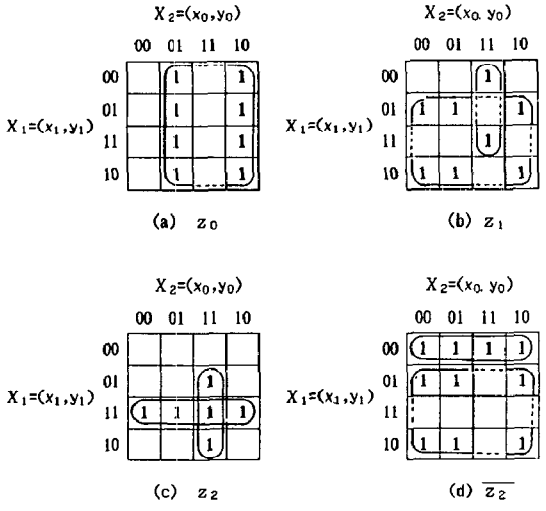


Fig.2.6 Maps for Two-bit Adder using TVFG's

```
/*************************************************************
/        TWO BIT ADDER WITHOUT CARRY INPUT
/
/                          A2    A1
/          +)              B2    B1
/          ------------------------
/                    S3    S2    S1      ---- SUM
/                          C2    C1      ---- CARRY
/-------------------------------------------------------------
FUNCTION                          /This is a FUNCTION statement.
    CARRY(A,B,C)=A*B+B*C+C*A      / * denotes AND ; + denotes OR
    SUM  (A,B,C)=A@B@C            / @ denotes EXCLUSIVE OR.
    ZERO (A)    =A*-A             /Constant zero function.
INPUT                             /This is an INPUT statement.
    A1,B1,A2,B2                   /List of the input variables.
OUTPUT                            /This is an OUTPUT statement.
    S1,S2,S3                      /List of the output variables.
CONNECT                           /This is a CONNECT statement.
    C0=ZERO(A1)                   /C0 denotes the carry input
    S1=SUM    (A1,B1,C0)          /which is equal to zero.
    C1=CARRY  (A1,B1,C0)
    S2=SUM    (A2,B2,C1)
    S3=CARRY  (A2,B2,C1)
END                               /This is an END statement.
/*************************************************************
```

Fig.4.2 Circuit Description of Two-bit Adder

realized instead of z_2 . And, we have

$\bar{z}_2 = x_1^{(01,10)} \cdot x_2^{(00,01,10)} \vee x_1^{(00)}$.

Note that the first term of $\bar{z}_2$ is equal to the first term of z_1. Thus the term can be shared by two expressions. Fig.2.7. shows the adder using TVFG's.

3) Next, we must expand TVFG's into ANDs, ORs and inverters. This process is called macro expansion. In this case, we have the options to choose AND or OR gates to make literals such as $X^{(00,11)}$ and $X^{(01,10)}$ (See Fig.2.1). We choose gates to make the resulting circuit as simple as possible. Only the gates whose outputs are actually used are realized. Other gates will be deleted. Fig.2.8 shows the TVFG's after macro expansion. When the resulting TVFG's have a pair of gates whose outputs are complementary, one is realized by connecting an inverter to the other to reduce the gate count. For example, in Fig.2.8, $\bar{x}_1 \cdot \bar{y}_1$ can be realized by $x_1 \vee y_1$, $\bar{x}_1 \vee \bar{y}_1$ can be realized by $x_1 \cdot y_1$, and $\bar{x}_0 \vee \bar{y}_0$ can be realized by $x_0 \cdot y_0$. Thus, we can obtain TVFG's with only 10 gates shown in Fig.2.9. This process is called TVFG gate reduction.

4) Fig.2.10 shows the final circuit. Note that only 15 gates are used. (End of example)

III Multi-level Circuit Generator

MACDAS generates a multi-level multi-output fan-in limited AND-OR circuit with TVFG's or OVFG's. In this section, we show a method for converting a two-level circuit into multi-level fan-in limited circuit. The method is primarily based on [DIE 78]. In order to explain the idea of the algorithm as simply as possible, we use an example of two-valued variables. However, the real factoring algorithm treats literals of both two-valued variables and four-valued variables.

3.1 Finding a Factor

Suppose that a two-level AND-OR circuit shown in Fig.3.1(a) is given. If we can find a set of h AND gates which have a common factor consisting of w literals, then we can realize a multi-level circuit shown in Fig.3.1(b), where the number of input lines of the first h AND gates are reduced by w.

Example 3.1: Suppose that we have to realize the following array by using AND and OR gates with fan-in limitation of four.

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
*	0	0	0	1	*	1	1	*	0
*	0	1	1	1	*	1	1	*	0
*	0	1	0	1	*	1	0	*	0
*	0	0	1	0	*	1	0	*	0

In the i-th column of the array, 1 denotes x_i , 0 denotes $\bar{x}_i$, and * denotes a don't care.

1) When we realize the above array in a straightforward way as shown in Fig.3.2(a), we need 9 gates.

2) When we use $\bar{x}_2 \cdot x_7 \cdot x_{10}$ as a common factor, and realize the circuit shown in Fig.3.2(b), we can resolve the fan-in limitation problems, and need only 6 gates. (End of example)

In general, suppose that the circuit shown in Fig.3.1(a) is given. If we can find a common factor consisting of w literals for h products, we can realize a circuit shown in Fig.3.1(b). By factoring, we can often resolve the fan-in limitation problem of AND gates and the OR gate, and thus can reduce the total number of gates. The number of the interconnections reduced by the factoring is given by $h \cdot w - (w+2) = w \cdot (h-1) - 2$. It is not easy to find a factor which maximally reduce the number of gates. Therefore, we try to find a factor which maximally reduce the number of interconnections.

We use $F(w,h) = w \cdot (h-1)$ as a figure of merit function, where w is called width and h is called height of the factor.

Example 3.2: Suppose that we have the following array.

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
*	0	0	0	1	*	1	1	*	0
*	0	1	1	1	*	1	1	*	0
*	0	1	0	1	*	1	0	*	0
*	0	0	1	0	*	1	0	*	0
1	*	0	0	0	1	*	*	*	*
*	1	1	0	0	1	1	*	*	1
0	*	1	0	0	1	*	*	*	*
1	*	1	1	1	*	*	*	*	*
1	*	0	*	1	*	*	0	*	*
*	1	*	*	*	*	0	1	0	*

1) We can find a factor having a maximum figure of merit with 9: the first 4 rows have a common factor with width 3 ($\bar{x}_2 \cdot x_7 \cdot \bar{x}_{10}$) and height 4.

2) For the rest of the the array, we can find a factor having a maximum figure of merit with 6: the next 3 rows have a common factor with width 3 ($\bar{x}_4 \cdot \bar{x}_5 \cdot x_6$) and height 3.

3) For the rest of the array, we can find a factor having a maximum figure of merit with 2: the last two rows but one have a common factor with width 2 ($x_1 \cdot x_5$) and height 2. (End of example)

The algorithm to find a factor with a maximum figure of merit is similar to [DIE 69], except that MACDAS use both two-valued variables (which have three different literals) and four-valued variables (which have 15 different literals).

3.2 Finding the Best Realization

Each time a factor with the maximum figure of merit is found, the algorithm compares the costs for three different realizations shown in Fig.3.2, and choose the most economical one. Type 1 realization shown in (a) is a straightforward AND-OR realization. Type 2 realization shown in (b) is the factored realization. Type 3 realization shown in (c) is an OR-AND realization, which can be obtained by 1) complementing a function, 2) minimizing the expression, and 3) complementing every inputs.

Example 3.3: Let's realize a circuit for the array in Example 3.2 by using 4-input gates.

1) For the first part of the array, three different realizations are shown in Fig.3.2. The numbers in the gates denote the numbers of 4-input gates to realize the gates with more inputs. Because Type 2 realization requires the fewest gates, we choose Type 2 realization.
2) For the second part of the array, Type 2 realization requires the fewest gates.
3) For the third part of the array, Type 3 realization requires the fewest gates.
4) For the last part of the array, we have only Type 1 realization.
5) Fig.3.3 shows the final circuit. (End of example)

When Type 2 or Type 3 realization has the fewest gates, we decide to use the factoring algorithm recursively or not by using the following Criterion : Let DIF be the number of additional gates to resolve the fan-in limitation problems in the factored circuit of Type 2 or Type 3 realization. If DIF > 3, then we recursively apply the factoring algorithm.

Example 3.4: Suppose that Type 2 realization is given as Fig.3.4(a). In this case, DIF=4, because we need 3 additional AND gates and one OR gate to esolve the fan-in limitation problems. Therefore, we recursively apply the factoring algorithm to the circuit which is inside of the dotted line in Fig. 3.4(a). In this case, a factor with width 2 and height 3 is found. Thus, we can save two gates by the recursive factoring. (End of example).

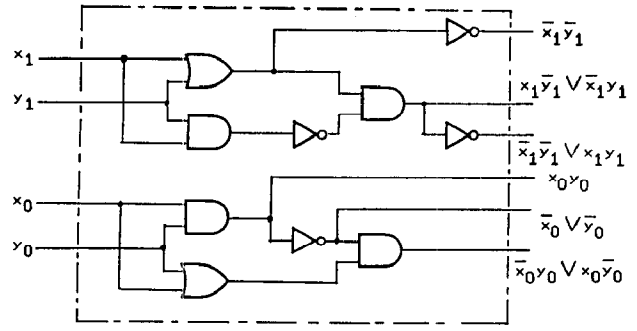


Fig.2.9 TVFG's after Gate Reduction

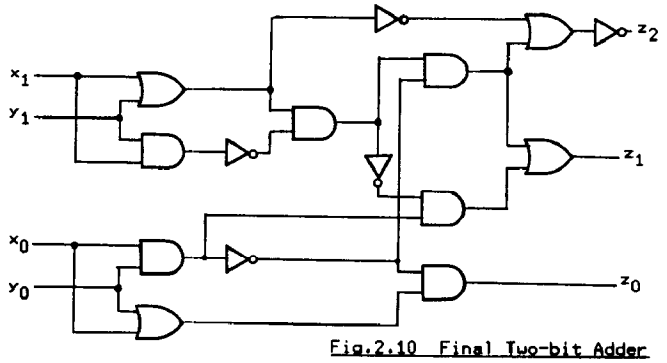
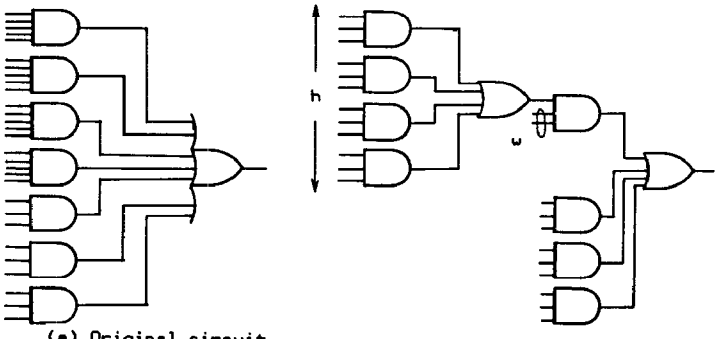


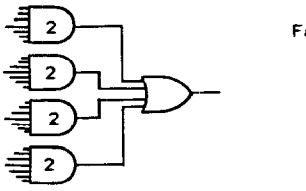
Fig.2.10 Final Two-bit Adder



(a) Original circuit

(b) Factored circuit

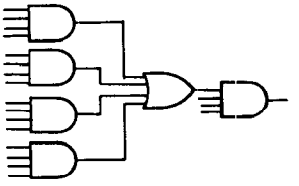
Fig.3.1 Factoring of two-level circuit



Fan-in limitation is 4.

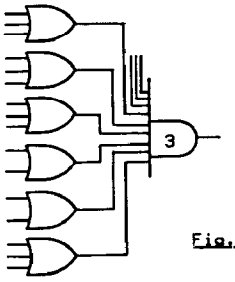
9 gates

(a) Type 1 realization



6 gates

(b) Type 2 realization



9 gates

(c) Type 3 realization

Fig.3.2 Three types of realization

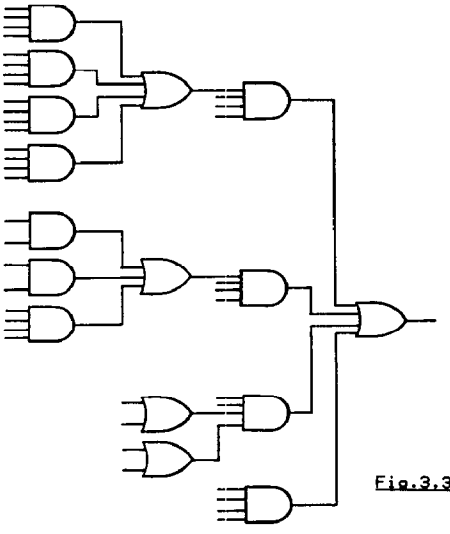


Fig.3.3 Final circuit

Given PLA (144 products)

↓ Logic minimization by MINI2 (245 sec)

Minimized AND-OR circuit with OVFG's (127 products)

↓ Optimal input variable assignment by ASS (5 sec)

AND-OR circuit with TVFG's (127 products)

↓ Logic minimization by MINI2 (48 sec)

Minimized AND-OR circuit with TVFG's (37 products)

↓ Output phase optimization by OPTOUT (115 sec)

Output phase optimized AND-OR circuit with TVFG's (26 products)

↓ Multi-level circuit generation by FACTI2 (58 sec)

Multi-level AND-OR circuit (81.5 gates)

Fig.5.1 Computation time for RD73 by MACDAS

(By a PC-98XA personal computer utilizing an 8-MHz Intel 80286 microprocesor)

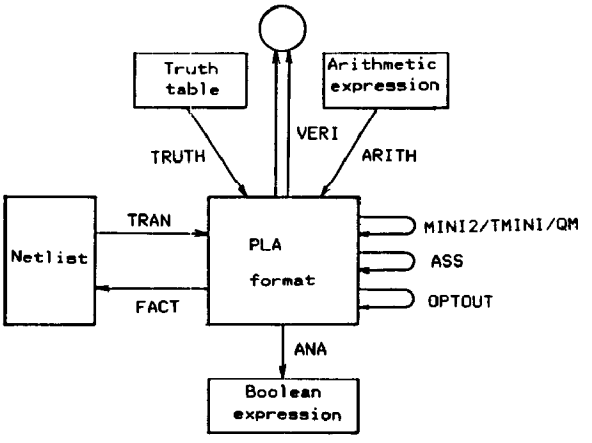


Fig.4.1 Tools in MACDAS

3.3 Calculation of Circuit Costs

The number of t-input gates to realize an n-input gate is given by

NGATES(n,t)=0 when n=1
=1+GTYPE*[(n-2)/(t-1)] when n ≥ 2,

where [a] denotes the integer part of a, and GTYPE is 1 or 2 depending on the type of the gates in the target technology. When we use ECL gates with both NOR and OR outputs, GTYPE=1. Whereas, when we use TTL or MOS gates of NAND or NOR, GTYPE=2, because we need additional inverters to change the polarities as shown in Fig.3.5.

3.4 Duplicated Gate Reduction

When we realize multiple-output function, the given array is partitioned into sub-functions according to their output patterns, and each function is realized as stated before. The external output function is realized by appropriately ORing the some of sub-functions. Then the program checks if there are gates whose inputs are identical. When such gates exist, only one gate is generated and others are deleted.

Example 3.5: In Fig.2.5(b), two-input OR gates for z_1 have identical inputs, and realize the same function $(x_0^0 \vee y_0^0)$. Therefore one gate can be deleted from the circuit. (End of example)

3.5 Gate Merging

When the resulting circuit has cascaded AND's or OR's as shown in Fig.3.6(a), we can merge the gates into one as shown in Fig.3.6(b). This operation is called gate merging.

3.6 Factorization on the Network

When there are gates whose inputs exceed the fan-in limitation, factorization on the network is applied. Consider the circuit in Fig.3.7(a). Suppose that the fan-in limitation of each gate is three. Then, two AND gates and two OR gate exceed fan-in limitation. In this case, an AND gate and an OR gate are created to resolve the fan-in problem. This process is similar to the weak division.

IV. Tools in MACDAS

MACDAS is currently used for academic research of logic design and logical complexity analysis, and consists of a set of tools shown in Fig.4.1. Each tool can be used as a command of MS-DOS. By combining these commands, we can make a batch command which automatically synthesize an AND-OR multi-level circuit from a given specification.

4.1 PLA generators

TRAN accepts a circuit description which consists of macro definitions and netlist, and generates a PLA format. Fig.4.2 shows the circuit description of a two-bit adder.

ARITH generates PLA formats for adders, multipliers, and other arithmetic functions. Each arithmetic function is described by FORTRAN statements.

4.2 PLA Minimizers

MACDAS has the following three strong multiple-valued logic minimizers.

MINI-II is an improved version of MINI[HON 74] with an essential prime implicant detection algorithm [SAS 84] and a fast recursive complementation algorithm[SAS 85a].

TMINI is similar to MINI-II, but minimizes large PLA's under limited memory space. In TMINI, a hardware tautology checker(HART) is used to accelerate the minimization process [SAS 85b]. Generation of prime implicants, detection of all the essential prime implicants and verification of the correctness of the minimization are accelerated by HART.

QM is an improved version of Quine-McCluskey algorithm. It obtains an absolute minimum PLA for a small problem. QM use a sparse matrix technique to store a large covering table efficiently, finds an absolute optimum solution quickly by using recently developed heuristics, and then proves its optimality by using two new bounding algorithms. It has obtained absolute minimum solutions and proved their minimality for SQR6(47 products in 5.4 min and proved its minimality in 7.8 min), and 5XP1(63 products in 14 min and proved its minimality in 0.5 min) on a PC 98XA personal computer utilizing an 8-MHz Intel 80286 microprocessor. Minimum solutions of these functions have not been obtained before [BRA 84a], [DAG 85]. It has also obtained a 121-product solution for MLP4 in 19.4 minutes.

4.3 Other PLA Optimization Tools

MACDAS has the following two distinguished PLA optimizers, by which we can obtain 30 percent smaller PLA's than standard methods[SAS 84].

ASS which finds a near optimal two-bit partition of the input variables.

OPTOUT which finds a near optimal output phase assignment.

V. Experimental Results

5.1 Description of Benchmarks

Table 5.1 shows 10 benchmarks used to evaluate the performance of MACDAS. Dr.Aart J. de Geus, the Session Chairman of Logic Synthesis and Optimization, collected these benchmarks and distributed diskette to each author of the session. To make the performance comparable to other systems, he proposed that the circuit cost be counted by 2-input gates equivalents. After simple examination of the benchmarks, the first 5 ones are found to be special functions which can be generated by simple programs: SYM9(=9SYM) is a symmetric function which is very difficult to minimize[HON 74, SAS 85b]. RD53, RD73 and SAO1 are identical to WGT(5),WGT(7), and WGT(8), respectively. WGT(n) is an n-input function which represents the number of 1's in the inputs. The number of products in a PLA for WGT(n) is (2^n-1) when the output phase is original and $2^n - C_n\,[n/2]$ when the output phase is optimal [SAS 86]. F2, VG2, BW, SAO2 seems to be control PLA's, but the author is not sure. DUKE2 is a control PLA. The benchmark called ALUPLA was given by a circuit description instead of PLA format, but MACDAS failed to convert it into a PLA due to memory size overflow.

5.2 Number of Product Terms

Table 5.1 compares the number of products in AND-OR two-level circuits having the following four different realizations:

1) Circuit with OVFG's with output phases original.
2) Circuit with OVFG's with output phases optimal.
3) Circuit with TVFG's with output phases original.
4) Circuit with TVFG's with output phases optimal.

For all functions, circuits with TVFG's require fewer products than ones with OVFG's. In all functions but 4, the number of product terms are reduced by optimizing the output phases.

All logic minimization were done by MINI-II. By using QM instead of MINI-II, we can obtain better solution, especially for the circuit with TVFG's. But, it takes more computation time.

5.3 Circuit areas

Table 5.2 compares the area of circuits having the four different realizations. Area are counted in terms of 2-input gate equivalents; an inverter counts for .5 area unit. It is assumed that only the true input signals are available. In Table 5.2, columns for PO denote the number of complemented

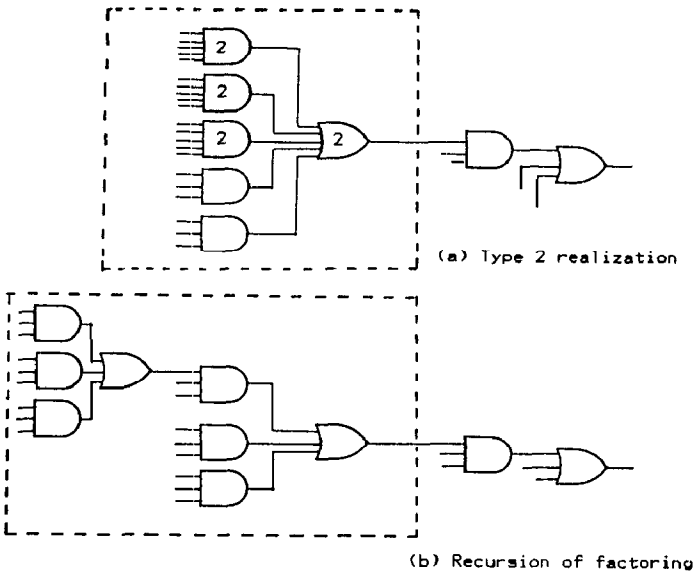


Fig.3.4 Recursion of factorization algorithm

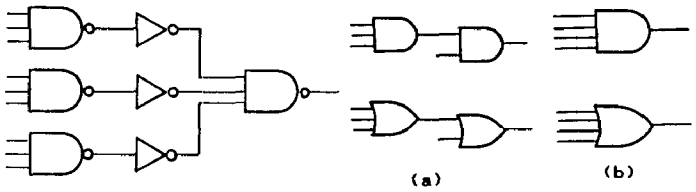


Fig.3.6 Gate Merging

Fig.3.5 Realization of 9-input NAND

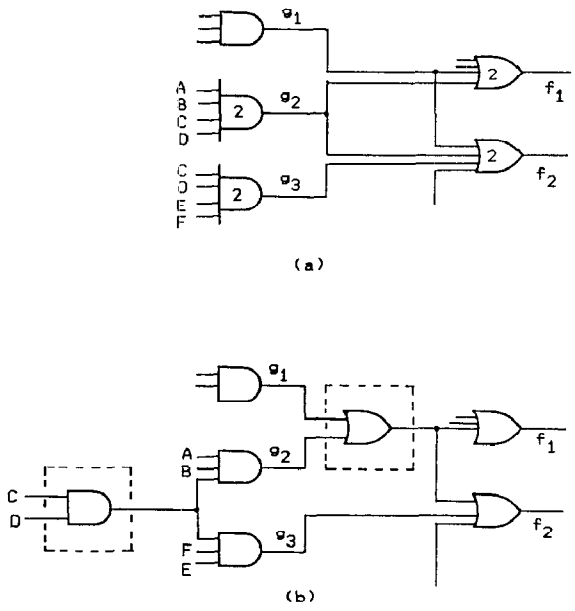


Fig.3.7 Factorization on the Network

Table 5.1 Number of Products

INPUT DATA				Circuits with OVFG's		Circiuts with TVFG's	
Circuit Name	# In	# Ou	# Cube	Output Phase Orig.	Output Phase Optml.	Output Phase Origl.	Output Phase Optml.
SYM9	9	1	420	87	72	27	27
RD53	5	3	32	31	22	12	10
RD73	7	3	141	127	93	37	26
SA01	8	4	256	255	186	54	38
5XP1	7	10	75	67	59	47	41
F2	4	4	12	8	8	6	6
VG2	25	8	110	110	110	88	88
BW	5	28	87	25	24	24	24
SA02	10	4	58	58	38	38	28
DUKE2	22	29	87	86	86	76	76

#in: Number of inputs
#ou: Number of outputs
#Cube: Number of cubes(products) of the given function.

ALUPLA was given by a circuit description instead of a PLA format; MACDAS failed to convert it into PLA format due to memory size overflow.

Table 5.2 Circuit areas in terms of 2-input gate equivalents

Circuit Name	Circuits with OVFG's			Circuits with TVFG's		
	Output Phase Orgnl.	Output Phase Optml.	PO	Output Phase Orgnl.	Output Phase Optml.	PO
SYM9	217.5	119.0	1	91.5	91.5	0
RD53	69.5	54.0	1	37.5	32.0	1
RD73	173.5	146.0	1	94.0	81.5	1
SA01	305.0	279.5	1	129.0	116.0	2
5XP1	162.5	154.5	2	125.5	118.0	2
F2	24.0	24.0	0	21.0	21.0	0
VG2	148.5	148.5	0	212.5	212.5	0
BW	130.5	123.0	17	123.5	123.5	0
SA02	148.0	139.0	2	127.5	117.0	2
DUKE2	337.5	337.5	0	301.5	301.5	0

PO denotes the number of complemented outputs.
An inverter counts for .5 area unit.
Only the true inputs are available.

outputs. Except for VG2, circuits with TVFG's require smaller areas than ones with OVFG's.

5.4 Computation time

Fig.5.1 shows the computation time for RD73. It took 471 seconds to generate a 81.5-gate circuit from a given PLA format by PC-98XA personal computer utilizing an 8-MHz Intel 80286 microprocessor.

5.5 Additional experiments

In addition to the benchmarks, the author tested ADR4, MLP4, NRM4(=DIST), ROT8(=ROOT),SQR6 RDM8(=F51M), GARY, X1DN, X6DN and Z4. The first 6 examples are arithmetic PLA's which are generated by simple programs. The author have been using these benchmarks since 1980 to test the performance of various logic minimizers [SAS 84, SAS 85b, SAS 86]. The PLA generator ARITH in 4.1 generates these arithmetic PLA's. Other examples are ESPRESSO-II benchmarks. Again, both TVFG's and output phase optimization were quite effective for arithmetic funcitons, but for other (control) PLA's output phase optimization were not so effective. See Table 5.3 and 5.4. The area for the additional examples are better or comparable to ones obtained by using weak division[DEG85, BAR 85].

6. Conclusion

MACDAS is useful tool to design multi-level circuit especially for arithmetic circuits. Two level circuits with TVFG's are usually more compact than ones with OVFG's, and they are technology independent. Therefore, they are good start points of multi-level logic synthesis. Application of weak division instead of factoring to the circuit with TVFG's is also promissing [BAR 85].

Acknowledgement

The author is grateful to Dr. A. de Geus who arranged the Logic Synthesis and Optimization Session. I am also grateful to Mr.M. Higashida who worked for the local transformation program. Part of this work was supported by Grant in Aid for Scientific Research of the Ministry of Education, Science, and Culture of Japan.

Rererence

[BAR 85] ICCD-85, pp.411-415, Oct. 1985
[BRA 84a] R.K.Brayton et.al, Logic Minimization Algorithms for VLSI Synthesis, Kluwer Academic Pub.
[BRA 84b] ICCD-84,pp.23-28, Oct. 1984.
[DAG 85] 22nd DAC, pp.667-673, June 1985.
[DAR 84] IBM Res. Dev.pp.537-545,Sept. 1984.
[DEG 85] IEEE Design and Test, pp.22-32, Aug. 1985.
[DIE 78] D.L.Dietmeyer,Logic Design of Digital Systems, Allyn and Bacon,Inc., Boston, 1978.
[ENO 85] ISCAS-85, pp.659-662, June 1985.
[HON 74] IBM Res. Dev. pp.443-458, Sept. 1974.
[HOS 84] CICC pp.356-360, May 1984.
[MUR 79] S.Muroga, Logic Design and Switching Theory, Wiley-Interscience, New York 1979.
[RUD 85] CICC-85 pp.230-234, May 1985.
[SAS 81] IEEE TC, C-30, pp.635-643, Sept. 1981.
[SAS 82] ISMVL-82,pp.45-54, May 1982.
[SAS 84] IEEE TC, C-33, pp.879-894, Oct.1984.
[SAS 85a]IEEE TC, C-34, pp.131-140, Feb.1985.
[SAS 85b]ICCD-85, pp.713-718, Oct. 1985.
[SAS 86] T.Sasao, Programmable Logic Arrays: How to make and How to Use,(in Japanese) NIKKAN KOGYOU Pub. Co.Tokyo, April 1986.

Table 5.3 Number of Products(Additional)

INPUT DATA				Circuits with OVFG's		Circiuts with TVFG's	
Circuit Name	# In	# Ou	# Cube	Output Phase Orig.	Output Phase Optml.	Output Phase Origl.	Output Phase Optml.
ADR4	8	5	255	75	61	17	14
MLP4	8	8	225	126	112	89	77
NRM4	8	5	255	120	106	82	70
ROT8	8	5	255	57	48	41	35
SQR6	6	12	63	51	41	41	38
RDM8	8	8	255	76	76	52	52
GARY	15	11	167	107	107	92	92
X1DN	27	6	110	110	110	80	80
X6DN	39	5	121	81	81	63	63
Z4	7	4	59	59	45	16	13

#in: Number of inputs
#ou: Number of outputs
#Cube: Number of cubes(products) of the given function.

Table 5.4 Circuit areas in terms of 2-input gate equivalents(Additional)

Circuit Name In-Out	Circuits with OVFG's			Circuits with TVFG's		
	Output Phase Orgnl.	Output Phase Optml.	PO	Output Phase Orgnl.	Output Phase Optml.	PO
ADR4	76.0	69.5	1	39.5	39.5	1
MLP4	382.0	328.0	2	250.5	233.5	3
NRM4	343.0	293.0	2	260.5	234.0	2
ROT8	146.0	125.5	3	142.0	121.0	3
SQR6	150.0	130.5	5	113.0	109.0	2
RDM8	150.0	150.0	0	138.0	138.0	0
GARY	396.5	396.5	0	369.0	369.0	0
X1DN	181.5	181.5	0	140.5	140.5	0
X6DN	292.5	292.5	0	273.0	273.0	0
Z4	102.5	71.0	1	48.5	42.5	1

PO denotes the number of complemented outputs.
An inverter counts for .5 area unit.
Only the true inputs are available.