CAD Logic Synthesis Algorithms for Multilevel-Multi-Output NOR Gate Circuits

Yaw-Koung Tsai
Lecturer of Electronic Engineering Department

Abstract

A fully computer-aided design procedure for the synthesis and minimization of multilevel-multi-output NOR gate logic networks as encountered in the synthesis of VLSI logic circuits is presented in this paper, A fast and very powerful technique using logic Zero-One-Interaction (ZOI) of permissible cubes is among the salient features in the synthesizing algorithm developed. Only uncomplemented inputs are involved in this automatic design technique, so fewer NOR gates are required in the synthesized logic network. The algorithm developed is feasible for any number of input variables and any incompletely specified functions. The algorithm is implemented in C language using software approach of our developed powerful data structure. Permissible-Cube-Related-Maxterm (PCRM) graph to generate the set of required permissible cubes automatically. The automatic level-reduction and gate-reduction of logic networks using CAD approach is also described. The CAD algorithms provide automatic synthesis of nearly minimal multilevel-multi-output NOR gate logic networks.

應用反相或閘多相輸出電路實現CAD邏輯運算法

蔡曜光

德霖技術學院電子科 講師

摘 要

本篇論文所提述的是：以反相或閘多相輸出邏輯電路(NOR)，來達到超大型積體邏輯電路(VLSI)的極小化設計，並實現電腦輔助設計(CAD)功能。

此篇研究的貢獻在於推論發展一套相當快速及有效的被允許的 0-1 特徵交互作用(ZOI)之技術理論。此理論的實現法則僅以未定(uncomplemented)輸入的反相或閘邏輯電路(NOR)，來達到複雜的邏輯電路的自動設計之技術。此運算法則可運用於多組不確定函數的多相數入之電路中。吾人使用 C 語言程式軟體來發展此套強而有力的數據架構即：可允許立體關聯多組態模組圖(Permissible-Cube-Related-Maxterm (PCRM))。應用此可允許立體關聯多組態模組圖(PCRM)，可大量降低電腦輔助設計的多層電路元件，並達到迅速及極小化的功效。

Ⅰ. Introduction

With the progress of VLSI technology, tens of millions of transistors can be placed on a single chip. The automation of logic synthesis can result in a reduction in design time, significant improvements of the circuit complexity, and can guarantee functional correctness.

The subject of two-level logic synthesis is well developed and well understood (Brayton 1984[3]). Exact techniques which provide minimum representations of the given logic are known (cf. Muroga 1982[26], Dagenais 1986 [8], Rudell 1985[2]). There have also been two generations of programs for generating near minimum logic representations (cf. MINI(Hong 1974[16])), ESPRESSO-II (Brayton 1984[4]), ESPRESSO-MV (Rudell and Sangiovanni-Vincetelli 1985[5]). In short, this is a well-developed science.

In contrast, multilevel logic synthesis has been studied less, is more difficult, and is relatively new. Nevertheless, multilevel logic synthesis has received most attention from CAD researchers because：

(a) It enables circuitry to be shared between the multiple functions;

(b) There is usually an area/delay trade-off for the implementation of a Boolean function;

(c) It is the basis for transforming the original representation into primitives that are more easily implementable.

This paper proposes a logic synthesis system using the permissible cube approach to obtain multilevel NOR gate logic networks (Eisenberg 1969[12], Muroga 1989[27]). NOR gate (or NAND gates for dual functions) logic networks are fundamental to integral circuit design such as TTL, CMOS, NMOS, and ECL (Davidson and Matze 1968[31], Davidson 1969[10]). Only uncomplemented input variables are needed by using the zero-one-interaction (ZOI) algorithm to synthesize NOR gate logic networks. This method can significantly reduce the circuit complexity of NOR gate networks.

The PCRM (Permissible-Cube-Related-Maxterm) graph ZOI system is used to generate and locate permissible cubes which are required for multilevel NOR gate logic synthesis. After all permissible cubes are generated, a primitive realization of the multilevel NOR gate network is obtained. The system developed also includes a level-reduction technique by which each primitive realization can be reduced to a three-level network without any increase in the number of gates. The level-reduction reduces circuit delay time due to its multilevel structure. Finally, a gate-reduction technique eliminates redundant gates in the synthesized primitive NOR gate network.

This paper is organized as follows:Section Ⅱ gives some definitions to help describe the algorithms used in the proposed system, and a system overview is given in section Ⅲ. All the major algorithms of the proposed system are detailed in section Ⅳ and section Ⅴ presents CAD of NOR gate logic networks and some exprimental results. Finally, conclusion is given in section Ⅵ.

Ⅱ. Background and Definitions

Permissible cubes are obtained from PCRM graphs to design multilevel NOR gate logic networks with uncomplemented inputs. The ZOI algorithm differs from the conventional product-of -sum (POS) algorithm in that it only uses uncomplemented inputs in a synthesized logic network. This section provides some basic definitions and concepts which are helpful in describing the algorithms used in the proposed logic synthesis system.

Definition 1

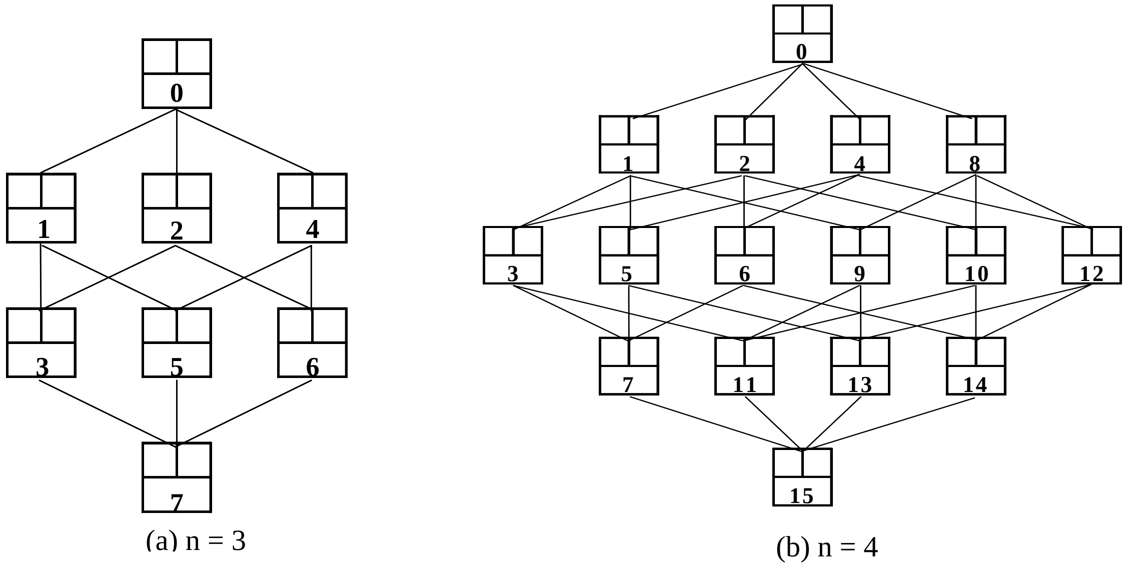
Two maxterms are said to be ***adjacent to each other*** if their bit distance is one.

For M_0(0000), M_2(0010), and M_3(0011) of a 4-input-variable Boolean function. M_0 and M_2, M_2 and M_3 are adjacent, while M_0 and M_3 are not adjacent.

Definition 2

A ***PCRM graph*** is a modified Karnaugh map. Each vertex of the PCRM graph stands for a maxterm; an edge stands for a pair of adjacent maxterms.

For a given switching function with n input variables, its PCRM graph has 2^n vertices, and each vertex has n adjacent vertices. It is easy to verify that a PCRM graph is a regular graph of dimension n (Mchugh 1990[25]). Some PCRM graphs with two, three and four variables are illustrated in Fig. 1.



(a) n = 3

(b) n = 4

Fig. 1 PCRM graphs : (a) n = 3; (b) n = 4

Definition 3

A ***primary cube*** is a o-cube (a maxterm) with all the input variables complemented. Obviously, the primary cube represents the maxterm M_0.

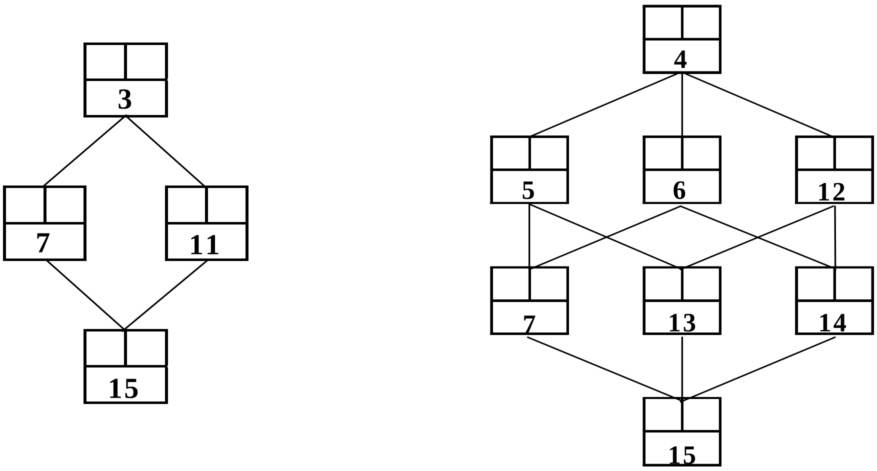
Definition 4

A ***termination cube*** is a o-cube with all the input variables uncomplemented. Obviously, the termination cube represents the maxterm M_2^{n-1}, where n is the number of variables in the given function.

Definition 5

A ***permissible cube*** $P_{x,y}$ is an r-cube, with 0≦r≦n, which contains the primary cube. $P_{x,y}$ corresponds to a permissible cube for gate $G_{x,y}$, where $G_{x,y}$ is denoted as the yth gate in level x of the synthesized NOR gate network.

By definition 5, the primary cube itself is a permissible cube. Each permissible r-cube corresponds to a Boolean sum of (n-r) uncomplemented input variables. Some examples are illustrated in Fig. 2 From definition 5 and the examples shown in Fig 2, a regular graph of degree r can represent a permissible cube. This is helpful when searching for necessary permissible cubes in the primitive realization.



(a) r = 2

(b) r = 3

Fig 2 Examples of permissible cubes for n = 4: (a) r = 2; (b) r = 3

《*Example 1*》

Given the switching function

$$\boldsymbol{f(x_1,x_2,x_3,x_4)=\prod(0,4,8,11,13)\cdot\prod_{d=0}(9,10)\cdot\prod_{d=1}(3,6),}$$

Its PCRM graph is shown in Fig. 3(a). We have $P_{1,1}$ ={0,8},$P_{1,2}$ ={0,4}. From the r-cube

(r = 2)   in Fig. 3(d) and using zero-one-interaction logic, we obtain $P_{2,1}$={0,4,8,12}.

Defintion 6

Let L_x be the ***level characteristic value*** of the *x*th logic level in the synthesized NOR gate network. L_1=0(1) if the logic value of the primary cube is 0(1). L_x has the property that $L_{x+1}=\overline{L_x}$.

Definition 7

$IS_{x,y}=\{\alpha \mid \alpha \in P_{x,y}$ and whose logic value is the same as that of $L_x\}$, where $IS_{x,y}$ is called the ***inhibiting set*** of $G_{x,y}$.

In example 1, we can find that $IS_{1,1}$ = {0,8} since L_1=0 and the logic value of o-cubes M_0 and M_8 are both zero. Similarily, we have $IS_{1,2}$={0,4}and $IS_{2,1}$={12}.

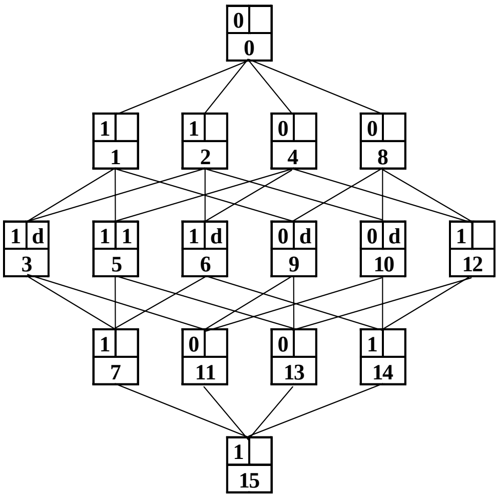
Definition 8

$VS_{x,y}=\{\beta \mid \beta \in P_{x,y}$ and $\beta \notin IS_{x,y}\}$, where $VS_{x,y}$ is a ***non-inhibiting set*** of gate $G_{x,y}$.
In example 1,$P_{2,1}$={0,4,8,12}, $IS_{2,1}$={12}, we obtain $VS_{2,1}$={0,4,8}.

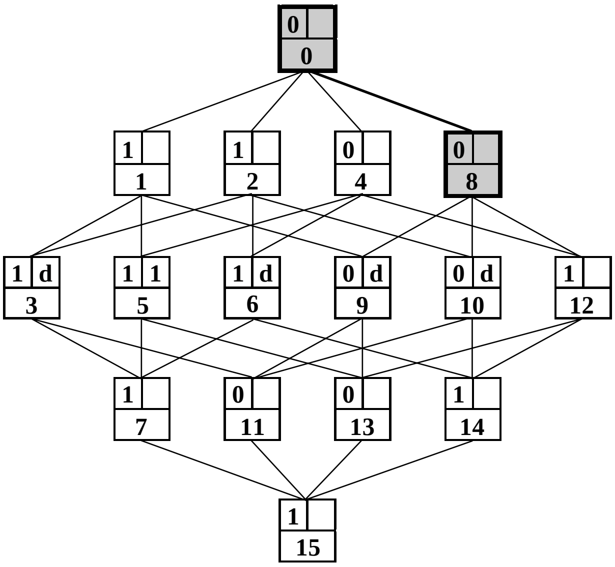
Definition 9

$GI_{x,y}$ is an input set corresponding to the synthesized NOR gate $G_{x,y}$ in the NOR logic network. $GO_{x,y}$ is the output of $G_{x,y}$.

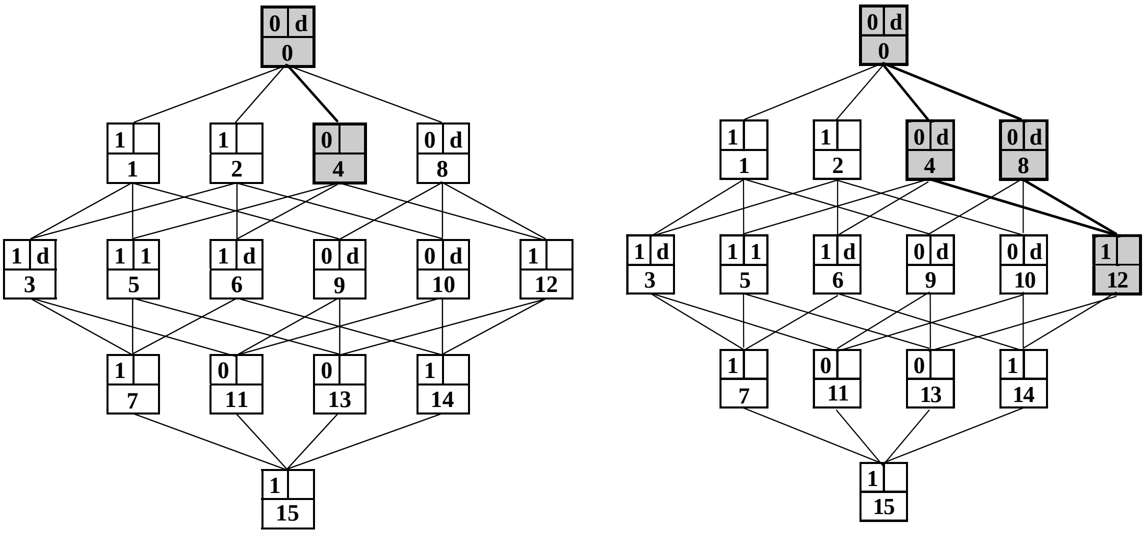
In example 1, we have $GI_{2,1}=\{x_3,x_4,GO_{1,1},GO_{1,2}\}$ The construction of $GI_{x,y}$ will be given in section Ⅳ.



(a): PCRM graph for
$f(x_1,x_2,x_3,x_4)=\prod(0,4,8,9,10,11,13)$



(b): permissible cube $P_{1,1}$={0,8}
obtained from *r*-cube (*r* = 1)



(c): permissible cube $P_{1,2}$={0,4}
obtained from *r*-cube (*r* = 1)

(d): permissible cube $P_{2,1}$={0,4,8,12}
obtained from *r*-cube (*r* = 2)

Fig. 3 (a)PCRM graph for $f(x_1,x_2,x_3,x_4)=\Pi(0,4,8,9,10,11,13)$; (b), (c), (d) permissible cube obtained from r-cube.

Ⅲ. System Overview

The logic synthesis system is carried out with four main procedures：
1)Primitive realization of multilevel NOR gate network.
2)Level reduction of the primitive realization into a three-level NOR gate network.
3)Gate reduction of the three-level NOR gate network.
4)Gate reduction of multiple output functions.

The input equation should be in the canonical product-of-sum form. The logic synthesis system constructs PCRM graphs in accordance with the input equation. The required permissible cubes are generated for each level characteristic value until each vertex of the PCRM graph has been covered. A permissible cube means a synthesizing gate existing in a level. Each gate has two kind of inputs:one is uncomplemented input variables corresponding to its permissible cube; the other is other gate outputs from a previous level if it has a non-inhibiting set. A multilevel NOR gate logic network is constructed according to the permissible cubes obtained from the PCRM graph. This process is called primitive realization of multilevel NOR gate network.

A primitive realization for a given function may be a circuit of more than three levels. To minimize the logic levels in the logic network, a level-reduction technique is used by which each primitive realization can be reduced to a three-level NOR gate network without increasing the total number of gates. The validity of the technique is proved in section Ⅳ.B.

The gate-reduction technique reduces redundant gates in the three-level NOR gate network. The non-inhibiting set of the final output gate of the three-level NOR logic network realized is actually equal to the union of inhibiting sets of gates at the second logic level. Our logic synthesis system realizes a minimum number of gates covering the non-inhibiting set of the final output gate. Some gates are redundant at the second level and can be eliminated. The details are described in section Ⅳ.C.

The final procedure of our logic synthesis system is to establish a multiple output network list of the input functions. In this step, we eliminate common shared gates of input functions.

Ⅳ. Logic Synthesis

The input switching function is transformed into canonical product-of-sum form. The output is a three-level network list. The algorithms will be described as follows. Also, an example is given to illustrate the logic synthesis algorithm.

A. Primitive realization of multilevel NOR gate network

This section presents a systematic realization procedure for multilevel NOR gate logic networks. An example illustrates the algorithms step by step.

Step 1. Construct a PCRM graph in accordance with the given switching function. Considering example 1 again, we construct a PCRM graph as in Fig 3(a).

Step 2. L_1=0(1) if the input symbol corresponding to the primary cube is in 0(1) set in the given function. For the given function *f*, the primary cube represents the input symbol (0000), which is in the 0-set. Therefore, L_1 is 0.

Step 3. Find all permissible cubes：

1. Choose those L_x-labelled 0-cubes(maxterms) and those $\overline{L_x}$-labelled 0-cubes which have already been inhibited by gates at previous levels.
2. Exclude those permissible cubes previously covered.

Then, the inhibiting and non-inhibiting sets can be found. Any permissible cube which satisfies the above process 1 describes a gate whose inputs consist of uncomplemented input variables and outputs from former (*x*-m)-level gates, where m=1,3,... and *x*-*m*>0. Process 2 excludes those unnecessary gates. In our example, we have *x*=1 and L_1=0. Furthermore, none of the 0-cubes have been inhibited by any gate, yet. Examination of the PCRM graph in Fig. 3(a) shows that there are two permissible cubes, which are $P_{1,1}$={0,8} and $P_{1,2}$={0,4} shown in Fig. 3(b) and 3(c) respectively. This brings about gates, $G_{1,1}$ and $G_{1,2}$ with the following descriptions：

$$\mathbf{P_{1,1}=\{0,8\},\quad IS_{1,1}=\{0,8\},}\quad \text{and}\quad \mathbf{VS_{1,1}=\varphi\ ;}$$

$$\mathbf{P_{1,2}=\{0,4\},\quad IS_{1,2}=\{0,4\},}\quad \text{and}\quad \mathbf{VS_{1,2}=\varphi}$$

Step 4. Find the uncomplemented input variables for the gates, $G_{x,j}$, generated in step 3. In this step, we can easy determine the uncomplemented input variables. Fig. 2 shows some examples of permissible cubes. Vertices with same number of 1' s in their coordinate set are placed at same level. The primary cube is the only one located at the top level. Also, there is only one vertex located at the bottom level. This maxterm is called the ***leading*** maxterm of permissible cube. The uncomplemented input variable of $G_{x,j}$ exists if the value of the corresponding variable location of the leading maxterm is logic 0. For example, in Fig. 3(b) the leading maxterm of $P_{1,1}$ is 8(1000), hence the uncomplemented input variables are x_2, x_3, x_4, and $GI_{1,1}=\{x_2,x_3,x_4\}$. Similarily, in Fig. 3(c) the leading maxterm of $P_{1,2}$ is 4(0100). We therefore obtain $GI_{1,2}=\{x_1,x_3,x_4\}$.

Step 5. Select a minimum number of $IS_{m,i}$'s is to cover$VS_{x.j}$, where m=x-1, x-3, x-5,... and find the inputs for second part of the gates generated in step 3, which are outputs from the gates at level x-1 or lower levels. In our example, we have $VS_{1,1}=\varphi$ and $VS_{1,2}=\varphi$. There is no other gate output needed to feed the gates at first level. We therefore obtain the gate description for the gates at first level：

$$\mathbf{GI_{1,1}=\{x_2,x_3,x_4\}}\quad \text{and}\quad \mathbf{GI_{1,2}=\{x_1,x_3,x_4\}}$$

Step 6. If the termination cube is covered in a permissible cube and IS($\boldsymbol{x}$) is equal to the o-set of the given function, where $IS(x)=IS_{x,1}\cup IS_{x,2}\cup \dots \cup IS_{x,j}$, a primitive realization is obtained. Otherwise, increase the value of $\boldsymbol{x}$ by 1 and set $L_{x+1}=\overline{L_x}$ and return to step 3.

Note that if the termination cube is covered there exists only one permissible cube at level x. This means that there exists only one gate at level $\boldsymbol{x}$. The fact that $IS(\boldsymbol{x})=IS_{x,1}$ is the 0-set of the given function means that the inhibiting set of the output gate $G_{x,1}$ inhibits those elements in the 0-set of the given switching function. From definition 4, the output of gate $G_{x.1}$ is the final gate of the given switching function. The resulting synthesized network is called a ***primitive realization network*** because techniques given below will usually yield a circuit with fewer logic levels and fewer gates.

Continuing the illustrative example, since the termination cube has not yet been covered, we increase the value of x by 1, set $L_2=\overline{L_1}=1$, and return to step 3. From Fig 3, we obtain

$$\mathbf{P_{2,1}=\{0,4,8,12\},\quad IS_{2,1}=\{12\},\quad VS_{2,1}=\{0,4,8\};}$$

Proceed to step 4 and step 5. From step 4, the uncomplemented input variables for gate $G_{2,1}$

are $\boldsymbol{x}_3$ and $\boldsymbol{x}_4$, because the leading maxterm is M_{12}(1100). From step 5, the inputs for second part of gate $G_{2,1}$ are $GO_{1,1}$ and $GO_{1,2}$ because
($IS_{1,1} \cup IS_{1,2}$) $\supset VS_{2,1}$ We have

$$\mathbf{GI_{2,1}=\{\mathit{x_3},\mathit{x_4},\ GO_{1,1}\ ,GO_{1,2}\}}$$

Similarly, we have

$$\mathbf{GI_{2,2}=\{\mathit{x_1},\ GO_{1,2}\},\ GI_{3,1}=\{\mathit{x_3},\ GO_{2,1},GO_{2,2}\},\ GI_{3,2}=\{\mathit{x_2},\ GO_{2,2}\}}\ ;$$

and the output gate

$$\mathbf{GI_{4,1}=\{GO_{3,1},\ GO_{3,2}\}}$$

The primitive realization network is shown in Fig. 4.

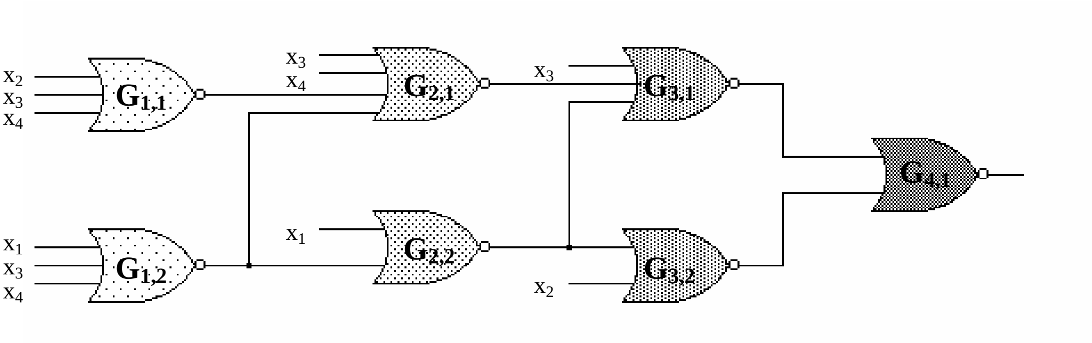


Fig. 4 Primitive realization for example 1

B. Level Reduction

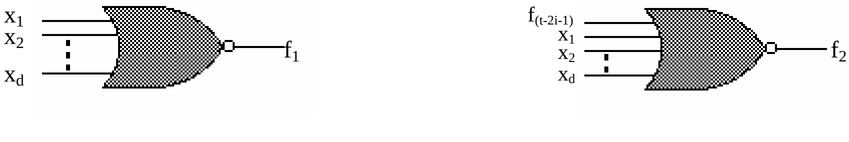
As shown in the previous section, a primitive realization for a given function may be a circuit with more than three levels. We propose a level-reduction technique by which every primitive realization can be reduced to a three-level network without increasing the total number of gates. The technique depends on the following theorem.

Theorem 1

In a primitive realization, any output of gate $G_{(t\text{-}2i\text{-}1),y}$ can feed the output gate $G_{t,1}$, where $G_{t,1}$ is the synthesized output gate of NOR logic network, and i is a positive integer.

《*Proof*》

Let $f_{(t\text{-}2i\text{-}1),y}$ be the output function of gate $G_{(t\text{-}2i\text{-}1),y}$ with gate level (t-2i-1) and f_1 be the realized NOR gate function. Assume that $G_{(t\text{-}2i\text{-}1),y}$ does not feed $G_{t,1}$ directly. Now connect the output of $G_{(t\text{-}2i\text{-}1),y}$ directly to $G_{t,1}$ and let f_2 be the function realized by the new network. The situation at $G_{t,1}$ before and after the application of $f_{(t\text{-}2i\text{-}1),y}$ is depicted in Fig. 5.



(a) (b)

Fig. 5(a) Output gate of original NOR network; (b) Output gate of new network

We have

$$f_1 = \overline{(x_1 + x_2 + \ldots + x_3)} = \overline{x_1} \cdot \overline{x_2} \cdot \ldots \cdot \overline{x_d},$$

$$f_1 = \overline{(f_{(t-2i-1),y} + x_1 + x_2 + \ldots + x_d)} = \overline{f_{(t-2i-1),y}} \cdot f_1$$

Then,

$$f_2 = \overline{f_{(t-2i-1),y}} \cdot f_1$$

There are two cases to be considered for any maxterm M_x inputs to the network.

1. For M_x with level characteristic value equals to 1, $f_{(t-2i-1),y}=0$. We have

$$f_1 = 1 \text{ and } f_2 = \overline{f_{(t-2i-1),y}} \cdot 1 = 1$$

2. For M_x with level characteristic value equals to 0, $f_{(t-2i-1),y} = 1$. We have

$$f_1 = 0 \text{ and } f_2 = \overline{f_{(t-2i-1),y}} \cdot 0 = 0$$

We have proved that

$$f_2 = f_1$$

In our example, t=4 and i=1, the output of gate $G_{1,1}$ and $G_{1,2}$ can feed the output gate $G_{4.1}$ directly. The resulting circuit is shown in Fig. 6.

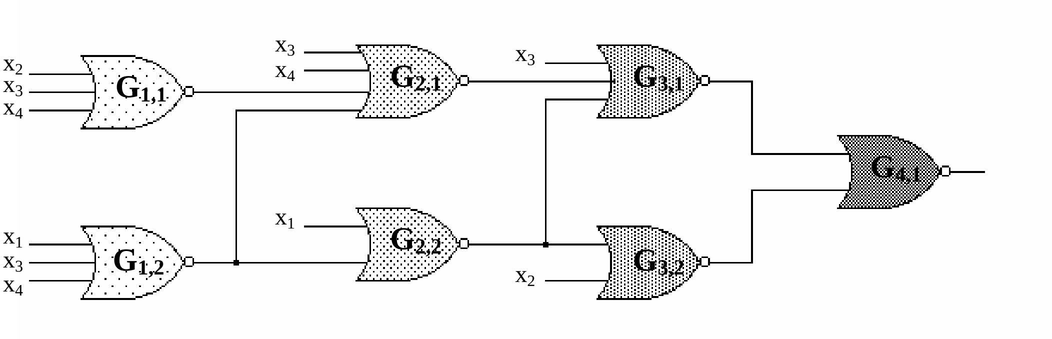


Fig. 6 Realization of example 1

Theorem 2

In a feedback-free NOR network, a gate which feeds the output gate need not feed any other gate.

《Proof》

Let $G_{i.j}$ be a gate whose output, $f_{i.j}$, feeds other gates in addition to the output gate. Furthermore, let x_1, x_2,... x_n be the input variables to the network. The function realized by the network can be expressed as

$$f_1 = \overline{f_{i,j} + f(f_{i,j}, x_1, x_2, \ldots, x_n)}$$

$$f_1 = \overline{f_{i,j}} \times \overline{f(f_{i,j}, x_1, x_2, \ldots, x_n)}$$

After all connections from $G_{i,j}$, except the one to the output gate, have been removed, the resulting function becomes

$$f_2 = \overline{f_{i,j} + f(0, x_1, x_2, \ldots, x_n)}$$

$$f_1 = \overline{f_{i,j}} \times \overline{f(0, x_1, x_2, \ldots, x_n)}$$

Note that the removal of an input to a NOR gate is equivalent to assigning a constant 0 to that specific input. There are two cases to be considered：

1. $\boldsymbol{f_{i,j} = 0}$. We have

$$f_1 = \bar{0} \times \overline{f(0, x_1, x_2, \ldots, x_n)} = \overline{f(0, x_1, x_2, \ldots, x_n)},$$

$$f_2 = \bar{0} \times \overline{f(0, x_1, x_2, \ldots, x_n)} = \overline{f(0, x_1, x_2, \ldots, x_n)}$$

2. $\boldsymbol{f_{i,j} = 1}$. We have

$$f_1 = \overline{1 + f(1, x_1, x_2, \ldots, x_n)} = 0,$$

$$f_2 = \overline{1 + f(0, x_1, x_2, \ldots, x_n)} = 0$$

We conclude that

$$\boldsymbol{f_2 = f_1}$$

Applying theorem 2 to the circuit in Fig. 6, we obtains the logic circuit shown in Fig. 7 by removing the connections from $G_{1,1}$ and $G_{1,2}$ to $G_{2,1}$ and $G_{2,2}$, The resulting circuit is a three-level logic network .

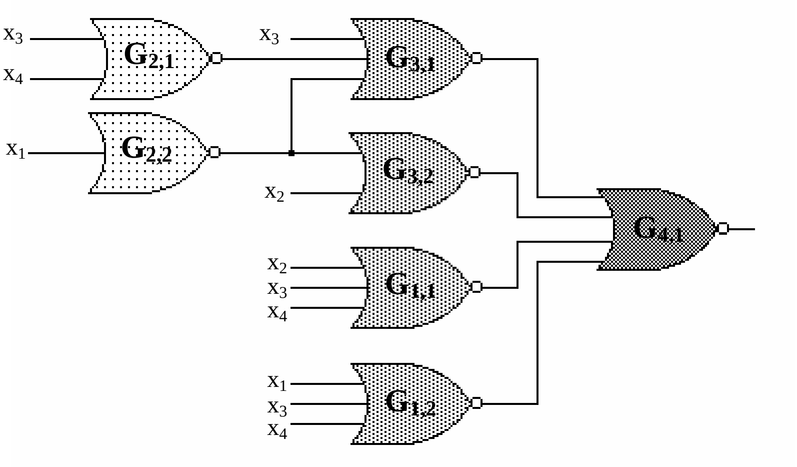


Fig. 7 Three-level realization for example 1

The level-reduction technique is described as follows.

Step 1. Feed any gate, $G_{i,j}$, where L_i=0, to the output gate.

Step 2. Remove the connections from $G_{i,j}$ to any other gates except the output gate.

The application of the level-reduction technique will always render a primitive realization into a simplified realization with no more than three levels without increasing the total number of gates.

C. Gate reduction of the three-level NOR gate network

In addition to the level-reduction technique, we propose a gate-reduction technique to minimize the number of gates in the realized three-level logic network.

Theorem 3

In a synthesized three-level NOR gate network, a gate at second NOR logic level is redundant and can be removed if all of its elements of its inhibiting set are covered by the other gates at second NOR logic level.

《*Proof*》

Let $G_{2,i}$ be a gate in a reduced three-level NOR gate network with an inhibiting set $IS_{2,i}$ with k elements. Because the k elements of the inhibiting set of gate $G_{2,i}$ are already covered by other gates at the second level, we have

$$\mathbf{IS_{2,\mathit{i}} \subset IS(2)^*}$$

where

$$\mathbf{IS(2)=IS_{2,1} \cup IS_{2,2} \cup \ldots \cup IS_{2,\mathit{i}} \cup \ldots \cup IS_{2,\mathit{j}}}$$

and

$$\mathbf{IS(2)^*=IS_{2,1} \cup IS_{2,2} \cup \ldots \cup IS_{2,\mathit{i}-1} \cup IS_{2,\mathit{i}+1} \cup \ldots \cup IS_{2,\mathit{j}}}$$

are the inhibiting sets of the second level in the NOR gate logic network before and after removing the gate $G_{2,i}$, respectively. Therefore IS(2)*=IS(2), since $IS_{2,\mathrm{i}}$ is a redundant inhibiting set. This theorem is proved.

The gate-reduction technique is described as follows.

Step 1. Implement the inhibiting set

$$\mathbf{IS(2)=\cup IS_{i,\mathit{j}}}$$

Step 2. Select a minimum number of $IS_{\mathrm{i},j}$, to cover IS(2). Eliminate the redundant gates in the realized three-level NOR logic network.

Continuing example 1, we find the inhibiting sets for the gates feeding the output gate.

$$\mathbf{IS_{3,1}=\{9,13\},\ IS_{3,2}=\{8,9,10,11\},\ IS_{1,1}=\{0,8\},}\quad \text{and} \quad \mathbf{IS_{1,2}=\{0,4\}}$$

We then have

$$\mathbf{IS(2)=IS_{3,1} \cup IS_{3,2} \cup IS_{1,1} \cup IS_{1,2}=\{0,4,8,9,10,11,13\}}$$

Then, we have a minimum cover of inhibiting set of second logic level:
{ $IS_{3,1}$, $IS_{3,2}$, $IS_{1,2}$ } and can eliminate gate $G_{1,1}$. The final circuit is shown in Fig 8.

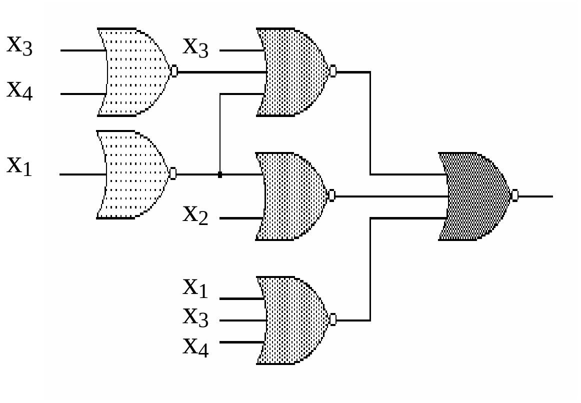


Fig. 8 Final synthesized circuit for example 1

x_1 x_2 x_3 x_4	f_1 f_2 f_3 f_4		x_1 x_2 x_3 x_4	f_1 f_2 f_3 f_4
0 0 0 0	0 0 0 0		1 0 0 0	0 0 0 1
0 0 0 1	1 0 0 0		1 0 0 1	1 0 0 1
0 0 1 0	0 1 0 0		1 0 1 0	0 1 0 1
0 0 1 1	1 1 0 0		1 0 1 1	0 0 0 0
0 1 0 0	0 0 1 0		1 1 0 0	0 0 1 1
0 1 0 1	1 0 1 0		1 1 0 1	0 0 0 0
0 1 1 0	0 1 1 0		1 1 1 0	0 0 0 0
0 1 1 1	0 0 0 0		1 1 1 1	0 0 0 0

Fig. 9 Truth table for F2

D. Gate reduction of multiple output functions

In this step, we examine common gates in the first level sharing the multiple functions. That is, any two gates of different output functions with same inputs are merged. Similarly, the gates at second level are checked for common sharing to obtain the simplest network. Consider a typical example described in Fig. 9. Some gates are shared by different functions. Fig. 10 shows the resulting circuits of these four functions. In Fig. 10, gate $G_{1,1}$ is shared by f_1 and f_4; gate $G_{1,2}$ is shared by f_1 and f_3, etc.

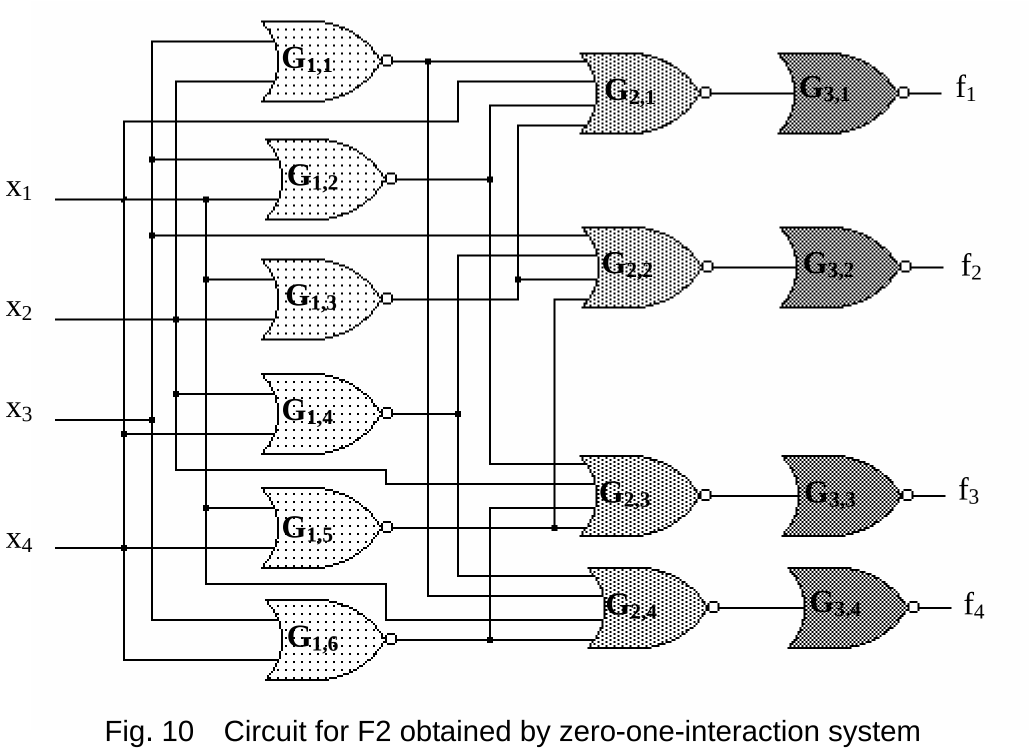


Fig. 10 Circuit for F2 obtained by zero-one-interaction system

Ⅴ. CAD Design And Experimental Results

The above algorithm is suitable for hardware logic design of multilevel NOR gate logic networks. The algorithm is implemented in C language using software approach of our developed powerful data structures, PCRM graph. This logic synthesis system has been implemented on a Sun SPARC-10/512 workstation. The CAD automatic logic design reduces design time sub-stantially.

The experimental results of some benchmarks are listed in Table Ⅰ and compared with those on CUP(Kuo, 1987)[20], MIS(Brayton 1987)[6] and VAR(Tai 1991)[29]. All of our results are better than those by CUP and VAR, and some are better than MIS.

In the primitive realization step. we use an exhaustive method to generate permissible cubes. This results in a long execution time. This situation can be improved by using a greedy method in future studies.

Table Ⅰ: Experimental results of some benchmarks

	I/O	CUP	MIS	VAR	PCRM
F2	4/4	36	24	23	22
RD53	5/3	152	52	52	52
SQR6	6/12	221	131	158	154
RD73	7/3	840	118	84	82
5XP1	7/10	294	137	280	256
ARD4	8/5	340	48	46	46
MLP4	8/8	790	331	399	328

I/O： Primary Input/Output
CUP： Number of literals by using CUP
MIS： Number of literals by using MIS
VAR： Number of literals by using VAR
PCRM： Number of literals by using PCRM

Ⅵ. Conclusion

A fully computer-aided design procedure for the automatic design of incompletely specified multilevel multioutput NOR gate logic network is given in this paper. We have used the technique of logic Zero-One-Interaction of permissible cubes and PCRM graphs to set up algorithms for the synthesis of multilevel NOR gate logic networks. Also, we used level-reduction and gate-reduction techniques to yield a circuit with less time delay and fewer gates. The automation design of logic synthesis can result in a reduction in design time and yields significant improvements of the circuitry complexity which is very important in reducing the VLSI chip area. The developed algorithms are implemented in software approach. The automatic design of NOR gate logic networks is feasible for any number of input variables. The developed design tool can obtain nearly minimum number of NOR gates in the synthesized logic network as shown by our experiments in Table Ⅰ.

Acknowledgment

We would like to express thanks to Dr. Wang, Dean of School of Engineering, Chung-Yung. Christian University, for his financial support in SUN SPARC-10/512 workstation.

REFERENCES

Bartlett, K., Brayton, R., Hachtel, G., Jacoby, R., Morrison, C., Rudell, R., and Sangiovanni-Vincentelli, A.: ‘Multilevel logic minimization using implicit don`t cares.’ *IEEE Trans. Computer-Aided Design*, 7, 1988, pp. 723-740.

Rudell, R., and Sangiovanni-Vincen telli, A.: ‘ ESPRESSO_MV:algorithms for multiple-valued logic minimization.’ *Proceedings of the Custom Integrated Circuits Conference*. Portland, Oregorr, 1985, pp. 230-234.

Brayton, R., Hachtel, G., McMullen, C., and Sangiovanni-Vincentelli, A.: ‘Logic Minimization Algorithms for VLSI Synthesis’ (Boston, Mass:Kluwer Academic), 1984.

Brayton, R., Hachtel, G., McMullen, C., and Sangiovanni-Vincentelli, A.,: ‘1984,ESPRESSO-II: A new logic minimizer for programmable logic arrays.’ *IEEE Custom Integrated Circuits Conference*, Rochester, N.Y. 1984.

Rudell. R., Sangiovanni-Vincentelli, A.,and Demicheli, G.: ‘A finite-state machine synthesis system.’ *International Symposium on Circuits and Systems*. Kyoto, Japan, 1985, pp. 647-650

Brayton, R., Hachtel, G., Sangiovanni-Vincentelli, A., and Wang, A.: ‘ MIS: A multiple-level logic optimization system.’ *IEEE Trans. Computer-Aided Design*, 1987, pp. 1062-1081.

Chen, S. S., and Chen, L. C.: ‘ CAD automatic design of multiple level NOR gate, networks using permissible cubes and cascade digraph.’ *The Chung Yuan Journal*, 21, 1992, pp. 512-534.

Dagenais, M., Agarwal, V., and Rumin, N.: ‘ MCBOOLE: A new procedure for exact, logic minimization.’ *IEEE Trans. Computer-Aided Design, CAD-5*, 1986.

Darringer, J., Brand, D., Gerbi, J., Joyner, W., and Trevillyan, L., ‘ LSS: A system for Production Logic Synthesis.’ IBM J. Res. Develop, 1984.

Davidson, E., and Metze, G.: ‘ Comments on an algorithm for synthesis of multiple, output combination logic.’ *IEEE Trans. Computers(short notes)*, C-17, 1969, pp. 1091-1092.

Dussault, J., Liaw, C., and Tong, M.: ‘ A high level synthesis tool for MOS chip design.’ *Proc. 22nd Design Automation Conf., Albuquerque*, NM. 1985.

Eisenberg, H.: ‘ An algorithm for the NAND gate realization of switching functions. Master`s thesis,’ Cornell University, Ithaca, N.Y. 1969.

Eris E., and Muzio, J.: ‘ Spectral testing of circuit realizations based on linearizations.’ *Proc. IEE Comput. & Digital Tech.*, 133, 1986, pp. 73-78.

Gregory, D., Bartlett, K., Degeus, K., and Hachtel, G.: ‘ SOCRATES: A system for automatically synthesizing and optimizing combinational logic.’ 23rd *ACM/IEEE Design Automation Conf.*, 1986.

Gurunath, B., and Biswas, N.: ‘ An Algorithm for Multiple Output Minimization.’ *IEEE Trans. Computer-Aided Design*, 8, 1989, pp. 1007-1012.

Hong, S., Cain, R., and Ostapko, D.: ‘ MINI: A heuristic approach for logic minimization. IBM.’ *Journal Research Development*, 18, 1974, pp. 443-458.

Hwang, T., Owens, R., and Irwin, M.: ‘Efficiently computing communication complexity for multilevel logic synthesis.’ *IEEE Trans. Computer-Aided Design,* 11, 1992.

Hurst, S.: ‘*The Logical Processing of Digital Signals.*’ (New York: Crane-Russak), 1978.

Hurst, S., Miller, D., and Muzio, J.： ‘Spectral Techniques in Digital Logic.’ (London, U.K: Academic), 1985.

Kuo, Y. S.: ‘Generation essential primes for a Boolean function with multiple-valued inputs.’ *IEEE Trans. Comput.,* C-3, 1987, pp. 356-359.

Kaushik De, Ballkrishna Ramkumar, and Prithviraj Banerjee,: ‘A Portable Parallel Algorithm for Logic Synthesis Using Transduction.’ *IEEE Trans. Computer-Aided Design*, 13(5), 1994, pp. 566-580.

Landman H.: ‘Logic synthesis at Sun.’ *IEEE Conference Paper,* 1989, CH2686-4/89/0000/0469.

McCluskey, E.: ‘Minimization of Boolean functions.’ *Bell System Tech. J.,* 35. 1956, pp. 1417-1444.

Malik, A., Brayton, R., Newton, A., and Sangiovanni-Vincentelli, A.: ‘Two-level minimization of multivalued functions with large offsets.’ *IEEE Trans. Computer-Aided Design,* 42, 1993.

Mchugh, J.: ‘Alorithm Graph Theory’ (Englewood Cliffs, N. J.: Prentice-Hall, 1990).

Muroga, S.: ‘VLSI System Design: When and How to Design Very-Large-Scale Integrated Circuits,’ (New York: Wiley, 1982).

Muroga, S, Kambayashi, Y., Chilai, H., and Culliney, J.: ‘The transduction method design of logic network based on permissible function.’ *IEEE Transaction Computer,* 38(10), 1989.

Schafer, I., and Perkowski, M.: ‘Synthesis of multilevel multiplexer circuits for incompletely specified multioutput Boolean functions with mapping to multiplexer.’ *IEEE Trans. Computer-Aided Desig Integrated circuits and system,* 12, 1993.

Tai, S. C., Du, M. W., and Lee, R, C.: ‘A transformational approach to synthesizing combinational circuits.’ *IEEE Trans. Computer-Aided Design,* 10(3), 1991, pp.286-295.

Tokmen, V.: ‘Disjoint decomposability of multi-valued functions by spectral means.’ *in Proc. IEEE Int. Symp. on Multiple-valued Logic,* 1980, pp. 88-93.

Davidson, E., and Metze, G.: ‘Comments on an algorithm for synthesis of multiple output combination logic.’ *IEEE Trans on Computers.* 17, pp. 1091-1092