

Title: Ternary Digital System: Concepts and Applications

Authors: A P Dhande, V T Ingole, V R Ghiye

Published by SM Online Publishers LLC

Copyright © 2014 SM Online Publishers LLC

ISBN: 978-0-9962745-0-0

All book chapters are Open Access distributed under the Creative Commons Attribution 3.0 license, which allows users to download, copy and build upon published articles even for commercial purposes, as long as the author and publisher are properly credited, which ensures maximum dissemination and a wider impact of the publication. Upon publication of the eBook, authors have the right to republish it, in whole or part, in any publication of which they are the author, and to make other personal use of the work, identifying the original source.

Statements and opinions expressed in the book are these of the individual contributors and not necessarily those of the editors or publisher. No responsibility is accepted for the accuracy of information contained in the published chapters. The publisher assumes no responsibility for any damage or injury to persons or property arising out of the use of any materials, instructions, methods or ideas contained in the book.

First published October, 2014

Online Edition available at www.smgebooks.com

For reprints, please contact us at ebooks@esciencemedicine.com

Ternary Logic Minimization

INTRODUCTION

We already discuss in chapter 1 Boolean algebra which is used for synthesis of logical expressions. It is also demonstrated that a logical expression can be realized using the logic gates. The number of input terminals for gate required for the realization of logical expression in general gets reduced considerably if the expression can be simplified. The simplification of logical expression is very important as it saves the hardware required to design a specific system.

Basically digital circuits are categories in two ways:

- 1) Combinational circuits
- 2) Sequential circuits

In combinational circuit the output at any instant of time depends upon the input present at that instant of time. This means there is no memory in this circuit. There are other type of circuit in which the output at any instant of time depend upon the present input as well as past inputs. This means that there are elements used to store past information. These elements are known as memory. Such circuits are known as sequential circuits.

Three methods are adopted for the realization of ternary circuit.

- 1) Adaption of Karnaugh map method
- 2) Adaption of Quine method: Adaption to ternary case
- 3) Adaption of Scheinman's Binary method: Adaption to ternary case.

STANDARD REPRESENTATION FOR LOGIC FUNCTION

Logical functions are expressed in terms of logical variables. The values assumed by the logical functions as well as the logical variable are in ternary form. Any arbitrary functions can be expressed as in Sum of Product (SOP) forms. Equations 4.1 shows SOP form of equation.

$$Y = (A \cdot B \cdot C) + (A \cdot B \cdot \bar{C}) \quad (1)$$

FUNCTION MINIMIZATION

One of the basic of logical design is the simplification of a given function. The problem of

determining the optimal representation of a given function is very complicated and has not been solved even for the binary case [1]. For function minimization only the cost factor of switching elements (Transistors / Fetes) is considered. Others factors such as speed of operation of ratability are not considered [2].

Let us define some basic terms that are modifications of their binary equivalents

Literal: $x_i^{a_i}$ is defined as literal where $a_i=0, 2, 01, 12, x, 1$.

Inclusion: The ternary n- variable function f_1 include the ternary n-variable function f_2

(Notation: $f_1 \geq f_2$

If for every $\alpha \in L^n$

$$f_1(\alpha) \geq f_2(\alpha)$$

Implicants: Two types of implicants of a ternary n-variable function F are defined.

1) g-type implicants - Φ_g is a product of irredundant satisfying the conditions

$$\Phi_g(\alpha) \leq f(\alpha) \quad \text{For every } \alpha \in L^n$$

2) h-type implicants- $\Phi_h(\alpha)$ is a product of irredundant literals satisfying the conditions

$$1. \Phi_h(\alpha) \leq f(\alpha) \quad \text{For every } \alpha \in L^n$$

Primary implicants:

1) A primary g- type implicant: Ψ_g of the ternary function f is g- type implicant of f satisfying one of the following conditions:

for every g- type implicant - Φ_g of f

$$a) \quad \Psi_g \leq \Phi_g \leq f \Rightarrow \Phi_g = \Psi_g$$

Or

$$b) \quad \Psi_g(\alpha) \cdot \Phi_g(\alpha) \leq \Phi_g(\alpha) \cdot 1$$

For at least one $\alpha \in L^n$ such that $\Psi_g(\alpha) \cdot \Phi_g(\alpha) \neq 0$ and one g-type implicant Φ_g of f

2) A primary h- type implicant: Ψ_h of the function f is an h- type implicant of f satisfying the following conditions

$$\Psi_h \cdot 1 \leq \Phi_h \cdot 1 \leq f \Rightarrow \Psi_h = \Phi_h$$

For every h- type implicant of f , and

$$\Psi_h \cdot 1 \not\leq \Phi_g$$

For every g- type implicant Φ_g of f .

A representation of a function f ,

$$f = \sum \Phi_g + 1. \sum \Phi_h$$

Where the Φ_g and Φ_h are g- type implicants of f , respectively, is quasi-basic. If only literals of the type $x^0, x^2, x^{01}, x^{02}, x^x$ are included, it is basic.

Simplicity Criterion

Cost function related to a basic representation of a ternary function f is defined as the total cost of the diodes and transistors needed for the realization of the function. A basic representation is called minimal when the value of the cost function is minimal.

As stated before we assume the AND (.) OR (+) circuitry to consists of diodes

And the unary function x^0, x^2, x^{01}, x^{02} realized by one triode (transistor) each. The function x^x is to be simply identical with x .

Theorem 2: Any ternary function f has a minimal basic representation of the form

$$f = \sum \Psi_g + 1. \sum \Psi_h$$

Where the Ψ_g and Ψ_h are primary g- type and h- type implicants respectively.

Proof: Let $f = \sum \Phi_g + 1. \sum \Phi_h$ be a minimal basic representation of the function f , where at least one of the implicants Φ_i is not primary. We have only the following two possibilities:

Φ_i Is a g-type implicant. Clearly we can find a primary implicant Ψ_g such that

$$\Phi_i < \Psi_g \leq f$$

Evidently the basic representation of Ψ_g does not contain more literals than Φ_i . Also; literals of the type x^x which appear in Φ_i cannot be replaced by literals of the type $x^{ai} a_i \neq x$ in Ψ . Thus, the representation obtained by replacing Φ_i by Ψ_g will also be minimal.

Φ_i is an h-type implicant. In this case we can find either a primary h-type implicant Ψ_h such that

$$\Phi_i < \Psi_h$$

Or a primary g-type implicant Ψ_g such that

$$1. \Phi_i < \Psi_g$$

Again similar considerations as before show that the replacement of Φ_i by Ψ_h or Ψ_g will result in a minimal representation of f .

The repeated application of the above procedure will finally lead to a representation as required by Theorem 2.

MINIMIZATION BY MAP METHOD

Although map methods provide a convenient approach to logical design their applicability in the ternary case is limited due to the large number of combinations, to functions of up to three variables.

Two Variables Maps

A two-variable map is given in Figure 4.1. The outputs $C_1 \dots C_9$ can have the values $C_i = 0, 1, 2, d$ where d stands for “don’t care.” Arrays of $3 \times 1, 3 \times 2, 3 \times 3, 1 \times 2, 3 \times 1, 3 \times 2, 3 \times 3, 1 \times 2$, and 2×2 cells of the map correspond to implicants of the two-variable function, represented by the map. Similar to the binary case the principle of the map Methods-consists, therefore, of joining corresponding cells into arrays large as possible. The corresponding implicants will be written in the form

$$F = g + h.1$$

The joining of cell is carried out according to the rules.

- 1) Arrays of $3 \times 1, 3 \times 2$ or 3×3 cells may be formed provided the output values coincide with the values of one of the inputs.
- 2) It permitted to join (properly) adjacent cells with equal output values.
- 3) Multiple uses of cells for different arrays are permitted.
- 4) Cells with the output value 2 can be considered as “don’t care” for the formation of arrays with the output value 1.

X1 \ X2	0	1	2
0	c ₁	c ₂	c ₃
1	c ₄	c ₅	c ₆
2	c ₇	c ₈	c ₉

Fig 4 1: Ternary two variable map

X1 \ X2	0	1	2
0	1	1	2
1	1	2	2
2	1	0	2

Fig 4. 2: Simplification of 2-variable function by map method

Example: a map representing a two –variable ternary function is shown in Figure 4.2. The arrays selected are also included in this figure 4.2. Thus the resulting representation of f is

$$f = x_2^2 + x_1^0 1 x_1^1 2 x_2^1 2 + 1. [x_1^0 1 + x_2^0]$$

Three-variable maps

For three variables functions we can draw a three-dimensional map Figure 4.3. But it is more convenient to use the two dimensional map shown in Figure 4.4. In this map C_i is considered adjacent to and but not to the method of using this map is similar to the two variable case and will now be illustrated by an example.

Example: A ternary three –variable function is represented in Figure 4.5. The arrays marked lead to the following results.

$$f = x_1^0 . x_2 . x_3^0 + x_1^0 . x_2^1 . x_3^{12} + 1 \cdot x_1^0$$

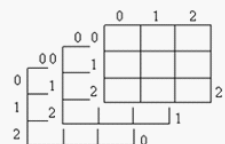


Fig 4.3: Three dimensional 3-variable map

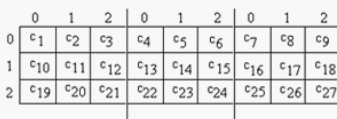


Fig 4.4: Two dimensional three variable map

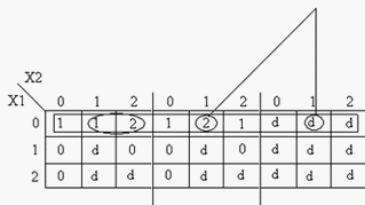


Fig 4.5: Simplification of 3-variable function by map method

Examples on MAP Minimization Method

Example 1) Design a circuit for ternary output using map method that will have intermediate level output only when one of the inputs is 1.

Solution: Let A and B be the two inputs. Truth Table 4.1 for the circuit is given below along with K-map.

Truth table 4.1		
Input A	Input B	Output
0	0	0
0	1	1
0	2	0
1	0	1
1	1	1
1	2	1
2	0	0
2	1	1
2	2	0

By considering above truth table using ternary algebra, we can write output equation for 1's term as $F = A^0 B^1 + A^1 B^0 + A^1 B^1 + A^1 B^2 + A^2 B^1$ which is not justified output for circuit implementation as more number of logic gates are required. By using map minimization technique we can reduce number of logic gates required for circuit implementation. Figure 4.6 shows K-map for truth Table 4.1

B	A		
	0	1	2
0	0	1	0
1	1	1	1
2	0	1	0

K -map for Table 4.1

According to rules explained in above, we can group all 1's term. There is no ant 2; s term for grouping. Therefore the output equation becomes $F = 0 + 1 \cdot (A^1 + B^1)$. Figure 4.7 shows circuit implementation for truth Table 4.1

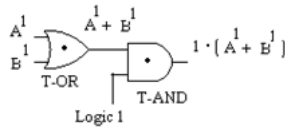


Figure 4.7: Circuit implementation for truth Table 4.1

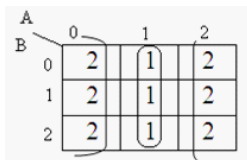
Example 2) from the given truth Table 4.2, writes k-map, derive minimized output equation and implement the circuit.

Solution: Referring to k-map here grouping of 2's and 1's is possible. As per the rules explained above of 2's as 3x2 is shown by simple bracket and 1' by 1x3 using simple bracket. Therefore the final minimized output equation is $F = A^0 + A^2 + 1 \cdot (A^1)$. The circuit for given output equation is shown in Figure 4.8.

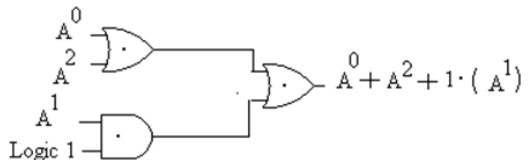
The examples on three variable k-maps are given in chapter 5.

Truth Table 4.2:

A	B	Output
0	0	2
0	1	2
0	2	2
1	0	1
1	1	1
1	2	1
2	0	2
2	1	2
2	2	2



(a)



(b)

Figure 4.8: (a) K-map for truth table (b) Circuit for given K-map.

THE QUINE METHOD: ADAPTION TO TERNARY CASE

By theorem 2 it is obvious that a minimal basic representation of a given ternary function may be obtained from its primary applicants. We shall now discuss a method of deriving all primary applicants which is an adaptation of Quine's binary procedure [3]. The method suggested is systematic and can be easily mechanized. Furthermore, the procedure can be started from any basic representation.

Let us introduce some additional concepts required in the sequel.

The following definitions refer to implicants of a given ternary n -variable function f .

Disjoint Implicants

1) Two g -type implicants

$$\Phi_{gj} = x_1^{a_{j1}} \dots x_i^{a_{ji}} \dots x_n^{a_{jn}}$$

$$\Phi_{gk} = x_1^{a_{k1}} \dots x_i^{a_{ki}} \dots x_n^{a_{kn}}$$

Are disjoint in the coordinate i , if

$$x_i^{a_{j1}} \cdot x_i^{a_{ki}} \neq 2$$

Example: The two g -type implicants

$$\Phi_{g1} = x_1^0 \cdot x_2^1 \cdot x_3^x \cdot x_4^2$$

$$\Phi_{g2} = x_1^2 \cdot x_2^{12} \cdot x_3^1 \cdot x_4^x$$

Are disjoint in the coordinates 1 and 3

2) Two h -type implicants

$$\Phi_{hj} = x_1^{a_{j1}} \dots x_i^{a_{ji}} \dots x_n^{a_{jn}}$$

$$\Phi_{hk} = x_1^{a_{k1}} \dots x_i^{a_{ki}} \dots x_n^{a_{kn}}$$

Are disjoint in the coordinates i , if

$$x_i^{a_{ji}} \cdot x_i^{a_{ki}} = 0$$

Example: The two h -type implicants

$$\Phi_{h1} = x_1^0 \cdot x_2^1 \cdot x_3^x \cdot x_4^2$$

$$\Phi_{h2} = x_1^1 \cdot x_2^{12} \cdot x_3^1 \cdot x_4^x$$

Are disjoint only in coordinate 1

Consensus

Let Φ_k and Φ_l ,

$$\Phi_k = x_1^{a_{k1}} \dots x_i^{a_{ki}} \dots x_n^{a_{kn}}$$

$$\Phi_l = x_1^{a_{l1}} \dots x_i^{a_{li}} \dots x_n^{a_{ln}}$$

Be two implicants of the same type and i any coordinate.

If the literal x_i^x appears in an h -type implicants we first replace it by x_i^{12} . (This step does not change the function f)

If now for every coordinate $j, j \neq i$, the two implicants are not disjoint, the coordinate i satisfy

$$x_i^{a_{ki}} + x_i^{a_{li}} = x_i^\alpha$$

$$\alpha = \Delta, 01, 12$$

The consensus in the coordinate i (notation: $\Phi_k \overset{i}{*} \Phi_l$) is obtained by

$$\Phi_m = \Phi_k \overset{i}{*} \Phi_l = x_1^{a_1} \dots x_i^\alpha \dots x_n^{a_n}$$

Where

$$x_j^{a_j} = x_j^{a_{kj}} . x_j^{a_{lj}}, j \neq i$$

Joining of implicants

Consider a g-type implecant Φ_g and an h-type implicant Φ_h

$$\Phi_g = x_1^{a_{g1}} \dots x_i^{a_{gi}} \dots x_n^{a_{gn}}$$

$$\Phi_h = x_1^{a_{h1}} \dots x_i^{a_{hi}} \dots x_n^{a_{hn}}$$

Where a_{gi} “contains” the value 2, a_{hi} the value 1 ($a_i = 12$ “contains” 1 and 2, $a_i = \Delta$ “contains” the value 0, 1, 2 etc.) and for every coordinate $j, j \neq i$ at least one value of x_j satisfies the condition

$$x_j^{a_{gj}} . x_j^{a_{hj}} = 2$$

The joining of Φ_g and Φ_h in coordinate i is then defined by

$$\Phi_g \overset{i}{0} \Phi_h = x_1^{a_1} \dots x_i^x \dots x_n^{a_n}$$

And

$$x_j^{a_j} = x_j^{a_{gj}} . x_j^{a_{hj}}, j \neq i$$

Example: Let

$$\Phi_g = x_1^0 . x_2^{12} . x_3^{01}$$

$$\Phi_h = x_1^{01} . x_2^1 . x_3^1$$

For the coordinate 2 the conditions for joining are satisfied and

$$\Phi_g \overset{2}{0} \Phi_h = x_1^0 . x_2^x . x_3^1$$

We now discuss some properties of consensus and the join.

Theorem 3: Let Φ_j and Φ_k be two implicants of the same type, having a consensus in the coordinate i . Then

$$\Phi_j \overset{i}{*} \Phi_k \leq \Phi_j + \Phi_k$$

Proof: Let

$$\Phi_j = x_1^{a_{j1}} \dots x_i^{a_{ji}} \dots x_n^{a_{jn}}$$

$$\Phi_k = x_1^{a_{k1}} \dots x_i^{a_{ki}} \dots x_n^{a_{kn}}$$

We define

$$\Phi_j = x_i^{a_{ji}} \Phi_j$$

$$\Phi_k = x_i^{a_{ki}} \Phi_k$$

It is clear that

$$x_i^{a_{ji}} \Phi_j \leq \Phi_j + \Phi_k$$

And therefore

$$x_i^{a_{ji}} \cdot \Phi_j \cdot \Phi_k \leq \Phi_j + \Phi_k$$

In a similar manner we obtain

$$x_i^{a_{ki}} \Phi_j \cdot \Phi_k \leq \Phi_j + \Phi_k$$

By addition of the last two equations we obtain

$$(x_i^{a_{ji}} + x_i^{a_{ki}}) \Phi_j \cdot \Phi_k \leq \Phi_j + \Phi_k$$

The expression appearing on the left-hand side of the inequality is by definition the consensus in the coordinate i . The theorem is thus proved.

Theorem 4: Let Φ_g be a g-type implicant and Φ_h an h-type implicant having a join in the coordinate i . Then

$$\Phi_g \overset{i}{\underset{0}{\cdot}} \Phi_h = \Phi_g + 1 \cdot \Phi_h$$

The proof is similar to that of theorem 3.

As an immediate result of theorem 4 and 5 we obtain the following.

Theorem 5: A ternary function f given in a quasi-basic representation will not change if the following implicants are added:

- 1) g-type implicants which are the consensus of two g-type implicants or the join of a g-type implicants and an h-type implicants.
- 2) h-type implicants, which are the consensus of two h-type implicants.

Algorithm for determining all the primary implicants

Let f be a ternary n - variable function given in any quasi -basic representations.

$$f = \sum \Phi_{gi} + 1 \cdot \sum \Phi_{hi}$$

It is clear that we can represent this function in the following way:

$$f = \sum \Phi_{gi} + 1 \cdot \left[\sum \Phi_{hi} + \sum_d \Phi_{gi} \right]$$

The expression $\sum_j \Phi_{gi}$ indicates that the combinations corresponding to the Φ_{gi} –scan be taken as “don’t care” states for the computation of $\sum \Phi_{hi}$.

We shall define two operations that can be carried out on the representations () without changing f .

Let the representation contain g - type implicants $\Phi_{gi}, \Phi_{gj}, \Phi_{gk}, i \neq j, i \neq k$ such that

$$\Phi_{gi} \leq \Phi_{gj} \text{ Or } \Phi_{gi} \cdot \Phi_{gk} \not\leq \Phi_{gk} \cdot 1$$

For at least one combination of the variables where $\Phi_{gi} \cdot \Phi_{gk} \neq 0$ then Φ_{gi} can be cancelled.

Alternatively let the representation contain h - type implicants Φ_{hi}, Φ_{hj} such that

$$\Phi_{hi} \leq \Phi_{hj}, i \neq j$$

Or

$$1. \Phi_{hi} \leq \Phi_{gj}$$

Where Φ_{gj} is a g - type implicants of (57). Then Φ_{hi} can be cancelled.

2) If a consensus or a join can be formed from two implicants of the representation [3] the resulting implicant can be added.

Theorem 6: By the iterative use or the above operations on a ternary function f representation as in () a quasi-basic representation is obtained containing all the primary implicants and only such implicants.

Proof: It is obvious that the use of the above two operations does not change the given function f .

We now show that if a primary implicant Ψ_i of f is not yet included in the representation, the procedure is not yet terminated. For this purpose we have to distinguish between the following cases.

Case 1: let us assume that there exists a g - type primary implicant Ψ_g that does not yet appear

In the representation, i.e.

$$\Psi_g \leq f$$

$$\Psi_g \not\leq \Phi_{gi}$$

For every Φ_{gi} in (19)

Case 1a: No g - type implicant Φ_g of f properly includes Ψ_g .

In this case there must exist a g - type implicants Φ_g such that

$$1) \Phi_g \leq \Psi_g$$

$$2) \Phi_g \not\subseteq \Phi_{gi}$$

3) Φ_g Does not properly include an implicant satisfying (2)

Two alternatives are still possible.

Φ_g Contains literals of all the variables where exactly one of the literals is of the type x^0, x^1 or x^2 Let

$$\Phi_g = a_1^{a_1} \dots a_i^{a_i} \dots a_n^{a_n}$$

Where $a_i = 01$ or 12

We can separate this expression into the two following ones:

$$\Phi_{g1} = x_1^{a_1} \dots x_i^{a_{i1}} \dots x_n^{a_n}$$

$$\Phi_{g2} = x_1^{a_1} \dots x_i^{a_{i2}} \dots x_n^{a_n}$$

Such that

$$x_i^{a_{i1}} + x_i^{a_{i2}} = a_i^{a_i}$$

$$x_i^{a_{i1}} . x_i^{a_{i2}} = 0$$

Due to property 3 of Φ_g the representation must contain implicants Φ_{gj} and Φ_{gk} of the form

$$\Phi_{gj} = x_i^{b_{i1}} \Phi_{gj}$$

$$\Phi_{gk} = x_i^{b_{i2}} \Phi_{gk}$$

Where

$$x_i^{b_{i1}} \geq x_i^{a_{i1}}$$

$$x_i^{b_{i2}} \geq x_i^{a_{i2}}$$

We can separate this expression into the two following ones:

$$\Phi_{g1} = x_1^{a_1} \dots x_i^{a_{i1}} \dots x_n^{a_n}$$

$$\Phi_{g2} = x_1^{a_1} \dots x_i^{a_{i2}} \dots x_n^{a_n}$$

Such that

$$x_i^{a_{i1}} + x_i^{a_{i2}} = a_i^{a_i}$$

$$x_i^{a_{i1}} . x_i^{a_{i2}} = 0$$

Of Φ_g the representation must contain implicants Φ_{gj} and Φ_{gk} of the form

Where

$$x_i^{b_{i1}} \geq x_i^{a_{i1}}$$

$$x_i^{b_{i2}} \geq x_i^{a_{i2}}$$

And the literals of $\overline{\Phi_{gj}}$ and $\overline{\Phi_{gk}}$ are part of the literals appearing in Φ_g

Hence the following consensus may be formed.

$$\Phi_{g1} = \Phi_{gj} \overset{i}{*} \Phi_{gh} \geq \Phi_g$$

Where

$$\Phi_{g1} \not\leq \Phi_{gi}$$

Thus the second operation is possible.

Φ_g Contains literals of all the variables when exactly one of them is of the type x^x and all the other of the type x^0, x^1 or x^2 .

i) We can show similarly to a) that a join may be added. Clearly these two alternatives are the only that can exist.

Case 1b: Ψ_g doesn't satisfy the conditions of case 1a but

$$\Psi_g \cdot \Phi_{gj} \leq \Phi_{gj} \cdot 1$$

For at least one combination of the variables, such that $\Psi_g \cdot \Phi_j \neq 0$, and at least one g- type implicants Φ_{gj} .

In this case the representation must contain at least one Φ_{gj} such that

$$\Psi_g \cdot \Phi_{gi} \leq \Phi_{gi} \cdot 1$$

For at least one combination of the variables such that $\Psi_g \cdot \Phi_{gi} \neq 0$.

The same Φ_{gi} appears in the expression $1 \cdot \sum_d \Phi_{gi}$ and therefore the join may be added.

$$\Psi_g \overset{i}{\underset{0}{*}} \Phi_{gi}$$

Case 2: There exists an h-type implicant Ψ_h not contained in the representation. Similarly to case 1 it is easily shown that a consensus may be added.

(Note: a literal x_i^x appearing in an h-type implicant can be changed, as we have seen, to x_i^{12} .)

Let us now show that it is impossible for the procedure to terminate as long as the representation contains an implant Φ_i which is no primary. As every implicant Φ_i which is not primary is included by primary implicant there are two possibilities.

1) This primary implicant appears in the representation and Φ_i can therefore be cancelled.

2) This primary implicant does not yet appear and therefore the procedure is not yet terminated.

It is also clear that the procedure is finite. Implicants which have been cancelled will not be added again implicants, and which have been added once will not be added again.

The applications of the algorithms described will be illustrated by an example.

The following three variables ternary function is given:

Example 3) Suppose given ternary function is:

$$f = x_1^0 x_2^0 x_3^0 + x_1^1 x_2^1 x_3^0 + x_1^0 x_2^2 x_3^0 + x_1^2 x_2^0 x_3^1 + x_1^0 x_2^0 x_3^1 + x_1^2 x_2^1 x_3^2 + x_1^0 x_2^1 x_3^0 + x_1^1 x_2^2 x_3^0 + x_1^1 x_2^2 x_3^1 + 1 \cdot [x_1^2 x_2^1 x_3^1 + x_1^2 x_2^1 x_3^2 + x_1^2 x_2^2 x_3^2 + x_1^0 x_2^2 x_3^2] + [x_1^1 x_2^0 x_3^2 + x_1^1 x_2^1 x_3^2 + x_1^1 x_2^2 x_3^2] d$$

The brackets marked by d correspond to “don’t care” combinations.

The given implicants are written in two separate columns, one for the g-type and the second h-type implicants. The implicants corresponding to the “don’t care” combination will be marked by d. Obviously all the g-type implicants appear as d’s in the h column.

To simplify the notations we shall indicate an implicant $x_1^{a1} \dots x_n^{an} \dots$ by the a’s only. The literal x^{12} will be written as 2, the literal x^{01} as 0 and x^Δ as Δ .

For example, the four variable implicants $x_1^{01} \cdot x_2^1 \cdot x_4^{12}$ will be written as $01\Delta 2$.

The procedure is best started from g-type implicants. All the g-type implicants may then be added as “don’t care” to the h column.

Clearly an implicant obtained from “don’t care” implicants only is also a “don’t care” implicants. Throughout this procedure “don’t care” implicants obtained must be marked as such (by the letter ‘d’) this information is of course essential for the last step, the selection of the primary implicants.

Figure 4.9 shows the complete procedure of forming consensus implicants from g-type implicants of f. This figure indicates how every implicant is obtained or cancelled.

Not every implicant added entails the canceling of the original implicants. For example the addition of implicant 9, obtained from implicant 1 and 3, results in their canceling. However, implicant 11 is obtained from 2 and 9 but only implicant 2 may be cancelled after implicant 11 has been added. Implicant 12 is obtained from 5 and 7 but neither of the original implicants can be cancelled.

The procedure for h-type implicants is similar. After adding all g-type implicant marked as ‘d’, the list shown in Figure 4.9 is obtained. The next step consists of generating all join-implicants. Figure 4.10 gives this procedure. The two columns are listed again (cancelled implicants have been omitted.)

Let us examine some examples. The g-type implicant $(0 \Delta 0)$ can be joined with $0 \Delta 0$ (d) which appears in the h column. The result $0 X 0$ is included by $0 \Delta 0$, but it satisfies the condition

$$(0 X 0). (0 \Delta 0) \leq 0 \Delta 0.1$$

For at least one input combination. Therefore 0 X 0 is primary, and thus added to the list. Another join implicant is 2 X 1,

$$2 \ 2 \ 0 \overset{2}{\underset{0}{\Delta}} = 2 \ X \ 1$$

This expression is not included by any other g-type implicants appearing in the representations and satisfies therefore the first condition for primary implicant.

It is clear that the implicant obtained by joining “don’t care” implicants need not be added.

Having reached the stage where it is no longer possible to apply the operations of the algorithms, we know by theorem 6 that all the primary implicants of the function have been derived. The function may now be represented as sum of all its primary implicants. Implicants known “don’t care” are of course, omitted at this stage.

The result according to Figure 4.10 is

$$f = x_1^0 x_3^0 + x_1^{01} x_2^{12} x_3^0 + x_1^{01} x_2^0 x_3^0 + x_1^0 x_2^0 x_3^0 + x_1^0 x_2^0 x_3^{01} + x_2^0 x_3^0 + x_1^{01} x_2^x x_3^0 + x_1^0 x_2^x x_3^0 + x_1^x x_2^x x_3^{01} + x_1^x x_2^x x_3^{12} + x_1^x x_2^x x_3^0 + x_1^{01} x_2^{12} x_3^x x_3^0 + 1 \cdot [x_1^2 x_2^{12} x_3^0 + x_2^2 x_3^0 + x_1^{12} x_2^{12} x_3^0 + x_1^2 x_2^2]$$

g-type implicants	h-type implicants
0 Δ 0	2 $\frac{1}{2}$ $\frac{1}{2}$
0 $\frac{1}{2}$ 0	Δ 2 2
0 0 2	$\frac{1}{2}$ $\frac{1}{2}$ 2
0 0 Δ	2 2 Δ
2 2, 0'	1 Δ 2 d
1 Δ 2 d	0 Δ 0 d
Δ 2 0	0' $\frac{1}{2}$ 0 d
0 X 0	0' 0 2 d
0' X 0	0 0 Δ d
0 0 X	Δ 2 0 d
2 X 1	
X 2 0	
1 X 2	

Joining procedure

Figure 4.9: The joining procedure.

Implicant Number	Implicant	Obtained from	Cancelled by
1	0, 0' 0	given	9
2	1 $\frac{1}{2}$ 0		11
3	0 2 0		9
4	2 2 0		10
5	0 0 $\frac{1}{2}$		13
6	2 2 $\frac{1}{2}$		14
7	1, 0' $\frac{1}{2}$ d		15
8	1 $\frac{1}{2}$ 2 d	$\frac{2}{1 \cdot 3}$ $\frac{2}{2 \cdot 4}$ $\frac{1}{2 \cdot 9}$ $\frac{5}{5 \cdot 7}$ $\frac{5}{5 \cdot 9}$ $\frac{6}{6 \cdot 10}$ $\frac{7}{7 \cdot 8}$ $\frac{9}{9 \cdot 10}$	15
9	0 Δ 0		
10	$\frac{1}{2}$ 2 0		16
11	0' $\frac{1}{2}$ 0		
12	0' 0 2		
13	0 0 Δ		
14	2 2 0'		
15	1 Δ 2 d		
16	Δ 2 0		

The consensus procedure for g-type implicants

g-type implicants	h-type implicants
0, 0' 0	2 $\frac{1}{2}$ $\frac{1}{2}$
1 $\frac{1}{2}$ 0	2 2 2
0 2 0	0 2 2
2 2 0	1 Δ 2 d
0 0 $\frac{1}{2}$	0 Δ 0 d
2 2 $\frac{1}{2}$	0' $\frac{1}{2}$ 0 d
1, 0' $\frac{1}{2}$ d	0' 0 2 d
1 $\frac{1}{2}$ 2 d	0 0 Δ d
0 Δ 0	2 2 0' d
$\frac{1}{2}$ 2 0	Δ 2 0 d
0' $\frac{1}{2}$ 0	2 $\frac{1}{2}$ 2
0' 0 2	$\frac{1}{2}$ 2 2
0 0 Δ	2 $\frac{1}{2}$ $\frac{1}{2}$
2 2 0'	Δ 2 2
1 Δ 2 d	$\frac{1}{2}$ 2 2
Δ 2 0	2 2 Δ

Consensus finding procedure for g and h type implicants

Figure 4.10: The consensus finding procedure for g-type & h-type implicant.

The last step of the simplification procedure is the selection of a minimal cost basic representation out of the list of primary implicants.

The application of McCluskey's method to our example yields the following representations:

$$f = x_1^{01} . x_2^{12} . x_3^0 + x_1^0 . x_2^0 + x_1^2 . x_2^2 . x_3^{01} + 1 . [x_2^2 . x_3^2 + x_1^2 . x_2^{12} . x_3^{12}]$$

Procedure for Minimization

Comparing with other minimization techniques it will be found that the ‘_’ variable of the redundant term takes three possible values 0, 1 and 2 in the triplet of three other forms. The remaining (n-1) variable of the redundant term are each identical to the corresponding (n-1) variable of the triplet, where “_” in any of the triplet term is taken as being equal to 0, 1, 2 for purpose of identity comparison.

Minimization procedure for the function $z = \sum ABCDE \dots$

1. Expand each term of given or function into all its minimum polynomials (e.g. $x_1^2 x_3^1 x_4^1 + x_2^0 x_4^2 + x_1^1 x_2^0 x_3^2 = 2011 + 0002 + 1020$) since x_2 is absent it is treated as 0 and so on for remaining variables.
2. Add together the values of each of the n variable in each midterm and establish the order of each mean term.
3. Make column of the all midterms
4. Delete the each duplication in each column
5. Compare each term in the column with all other term in next other columns.
6. For forming the decimal order, write order 1 such that alembic sum of total no of variable is one under order one, repeat the procedure for decimal order till final order get obtained. In this case it is 6.
7. Repeat the procedure 4,5 and 6 till term looking for complete variable is obtained.
8. The remaining terms shows the final minimized result.

Example 4) given ternary four variable function

$$f = x_1^2 x_3^1 x_4^1 + x_2^0 x_4^2 + x_1^1 x_2^0 x_3^2 + x_1^1 x_2^0 x_4^0 + x_1^1 x_2^0 x_3^0 x_4^1 + x_1^1 x_3^1 x_4^1 + x_2^2 x_4^1 + x_1^1 x_2^1 x_4^1$$

By expanding above terms, we can sum up midterms as:

$x_1^2 x_3^1 x_4^1$	$x_2^0 x_4^2$	$x_1^1 x_2^0 x_3^2$	$x_1^1 x_2^0 x_4^0$	$x_1^1 x_2^0 x_3^0 x_4^1$	$x_1^1 x_3^1 x_4^1$	$x_2^2 x_4^1$	$x_1^1 x_2^1 x_4^1$
2011	0002	1020	1000	1001	1011	0201	1101
2111	1002	1021	1010		1111	1201	1111
2211	2002	1022	1020		1211	2201	1121
	0012					0211	
	1012					1211	
	2012					2211	
	0022					0221	
	1022					1221	
	2022					2221	

Tabulating in resultant decimal order

Decimal order	0	1	2	3	4	5	6	7	8
List of all midterms	—	1000	1010 0002 1001	1020 1002 0012 1020* 1011 0201 1101	1012 1021 1111 1201 1111* 0022 0211 2011	2012 1022 1211 2201 1121 1022* 1211* 0221 2111	2022 2211 1221 2211	2221	—

Compare each term with all other terms looking for complete variable.

1000	1000	0002	0002	1010
1001	1010	1002	0012	1011
1002	1020	2002	0022	1012
100-	10-0	-002	00-2	101-
1001	1001	1020	1002	0012
1101	1011	1021	1012	1012
1201	1021	1022	1022	2012
1-01	10-1	102-	10-2	-012
1011	0201	0201	1101	2002
1111	0211	1201	1111	2012
1211	0221	2201	1121	2022
1-11	02-1	-201	11-1	20-2
1021	1201	0022	0211	2201
1121	1211	1022	1211	2211
1221	1221	2022	2211	2221
1-21	12-1	-022	-211	22-1
0221	2011			
1221	2111			
2221	2211			
-221	2-11			

Tabulating above lines in '-' order.

xxx-	xx-x	x-xx	-xxx
	10-0		
	00-2		-002
	10-1	1-01	-102
100-	10-2	1-11	-201
101-	02-1	1-21	-022
102-	11-1	2-11	-211
	20-2		-221
	12-1		
	22-1		

Comparing each term in each column, looking for complete variable

100-	10-0	00-2	10-1
101-	10-1	10-2	11-1
102-	10-2	20-2	12-1
10--	10--	-0-2	1--1
02-1	1-01	-002	-201
12-1	1-11	-012	-211
22-1	1-21	-022	-221
-2-1	1--1	-0-2	-2-1

Tabulating above surface in '-' order.

xx--	x-x-	-xx-	-x-x	x--x	--xx
10--			-0-2 -2-1	1--1	

No further variables are available in above tabulation. Therefore this is the minimization, as terms by themselves.

Therefore given expression for output is: $f = x_1^1 x_2^0 + x_4^2 x_2^0 + x_2^2 x_4^1 + x_1^1 x_4^1 + x_1^2 x_3^1 x_4^1$

SCHEINMAN'S BINARY METHOD: ADAPTION TO TERNARY CASE

By following the method [3], which is an adaptation of Scheinman's binary method [4], near to optimal solutions can be derived. The advantage of this method is that the procedure is m previous one, as some of the primary nonessential implicants are cancelled automatically during the simplification process. Here, again the last step is the selection of a minimal representation from the list of implicants obtained. This selection is made in the same way as in the modified Quine procedure described in the previous section.

On the other hand this method is only applicable to function given in canonical sum form.

The method is based on the iterated use of the expansion Theorem. The given function is again separated into g-type and h-type implicants. i.e,

$$f = g + 1.h$$

Each of these secondary functions, g and h, can be expanded in the following way:

$$g(x_1, \dots, x_n) = x_1^0 g(0, x_2, \dots, x_n) + x_1^1 g(0, x_2, \dots, x_n) + x_1^2 g(0, x_2, \dots, x_n)$$

and similarly for h.

The following explanations deals with the g function. Clearly the same considerations apply to the h function also.

The expansion of g about x_1 can be written as

$$g(x_1, \dots, x_n) = x_1^0 r_0 + x_1^1 r_1 + x_1^2 r_2$$

The residue functions r_0, r_1 and r_2 are, of course, sums of implicants.

We distinguish between the following possibilities.

1) The three residue functions are equal

$$r_0 = r_1 = r_2 = r$$

In this case we write

$$g = x_1^0 r_0 + x_1^1 r_1 + x_1^2 r_2 = (x_1^0 + x_1^1 + x_1^2) r = r$$

This means that the appearance of x_1 in the secondary function g is not required.

2) There exists a residue function r such that

$$r \leq r_0$$

$$r \leq r_1$$

$$r \leq r_2$$

In this case

$$g = x_1^0(r_0 - r) + x_1^1(r_1 - r) + x_1^2(r_2 - r) + r$$

Where e.g., r_0 is r are the impliants of r_0 not appearing in r ?

1) There exists a residue function r such that

$$r \leq r_0$$

$$r \leq r_1$$

Or

$$r \leq r_1$$

$$r \leq r_2$$

In this case

$$g = x_1^0(r_0 - r) + x_1^1(r_1 - r) + x_1^2r_2 + x_1^{01}.r$$

Or

$$g = x_1^0r_0 + x_1^1(r_1 - r) + x_1^2(r_2 - r) + x_1^{12}.r$$

2) If none of the above conditions are satisfied the original expansion of g about x_1 is left unchanged,

$$g = x_1^0r_0 + x_1^1r_1 + x_1^2r_2$$

The Scheinman's technique is based on the iterated use of the above expansion rules.

Product of literals $x_1^{a_1}, \dots, x_n^{a_n}$ appearing in the canonical sum form are called *atoms*.

The atoms will be written in two columns corresponding to g-type and h-type atoms. They will be denoted the same way as previously, e.g., $x_1^0.x_2^1.x_3^2$ is written as 012. "Don't care" atoms are marked by $\surd$. The g atoms are also included in the h column, marked by $\surd$.

Following are the Simplifications Rules

1) The first expansion is about x_1 . The column considered (g or h) is split into three columns with the headings x_1^0 , x_1^1 and x_1^2 . The entries in these columns correspond to the atoms containing x_1^0 , x_1^1 or x_1^2 , respectively. Only the exponent of $x_2, \dots, x_n$ are listed in these sub columns.

2) Entries derived from "don't care" entries are also marked as such.

3) If the columns of x_1^0 and x_1^1 contain the same entry this entry is also listed an additional column with the heading x_1^{01} , and the original entries are marked by $\surd$. Similarly if the same entry

appears in both the x_1^1 and x_1^2 columns it is also listed in an additional column x_1^{12} and the original entries are marked by $\surd$. If an entry appears in all the three columns x_1^0 , x_1^1 and x_1^2 it is listed in both the columns x_1^{12} and furthermore in a new column x_1^Δ .

In the three original columns and in the columns x_1^{01} and x_1^{12} this entry is marked by $\surd$.

4) Entries appearing in corresponding positions in the expansion for g and h are now examined. If the same entry appears in column x_1^1 in h and column x_1^2 in g it is listed in an additional column in g headed by x_1^x , and the original entries are marked by $\surd$.

If the same entry appears both in the x_1^Δ and x_1^x columns in g , it is marked by $\surd$ in the x_1^x column.

5) If all entries of a column are marked $\surd$, the column is cancelled.

6) Each of the derived columns is expanded about x_2 and so forth, for the other variables.

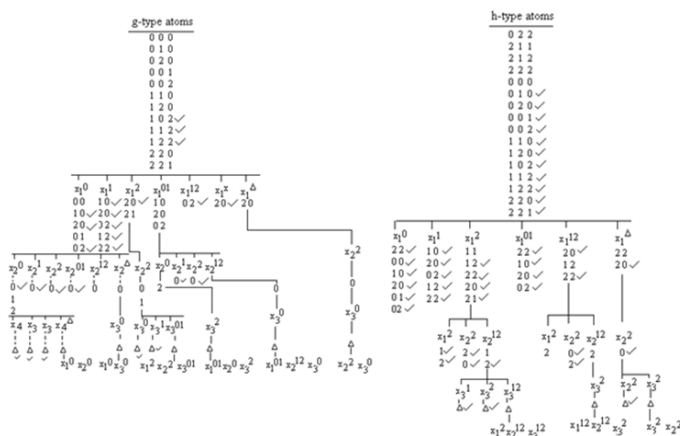
In the expansion about the last variable all the residues are Δ (where Δ denotes nonappearance).

At the end of this procedure the resulting implicants are found by moving upwards from any lowest entry which has not been cancelled and writing down the product of the literals heading the columns we pass while moving upwards.

The result obtained is the sum of g -type and h -type primary implicants. Although not all the primary implicants appear, it can be easily seen that the essential do. Clearly as not all the primary implicants appear only near –to – optimal representation can be expected. The complete procedure above is illustrated with the example.

Example 5)

$$f = x_1^0 x_2^0 x_3^0 + x_1^0 x_2^1 x_3^0 + x_1^0 x_2^2 x_3^0 + x_1^0 x_2^0 x_3^1 + x_1^0 x_2^1 x_3^1 + x_1^0 x_2^2 x_3^1 + x_1^1 x_2^0 x_3^0 + x_1^1 x_2^1 x_3^0 + x_1^1 x_2^2 x_3^0 + x_1^1 x_2^0 x_3^1 + x_1^1 x_2^1 x_3^1 + x_1^1 x_2^2 x_3^1 + 1 \cdot [x_1^1 x_2^1 x_3^1 + x_1^1 x_2^1 x_3^2 + x_1^1 x_2^2 x_3^2 + x_1^0 x_2^2 x_3^2] + [x_1^1 x_2^0 x_3^2 + x_1^1 x_2^1 x_3^2 + x_1^1 x_2^2 x_3^2] d$$



Scheinman's method simplification procedure

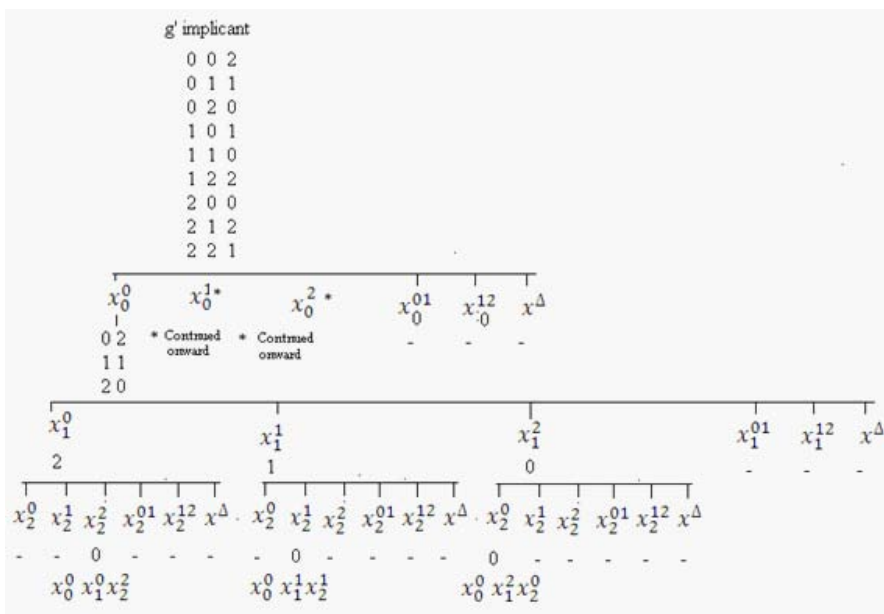
The result is therefore

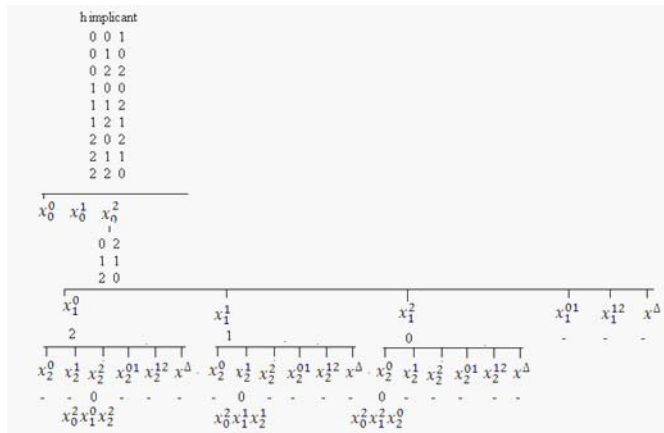
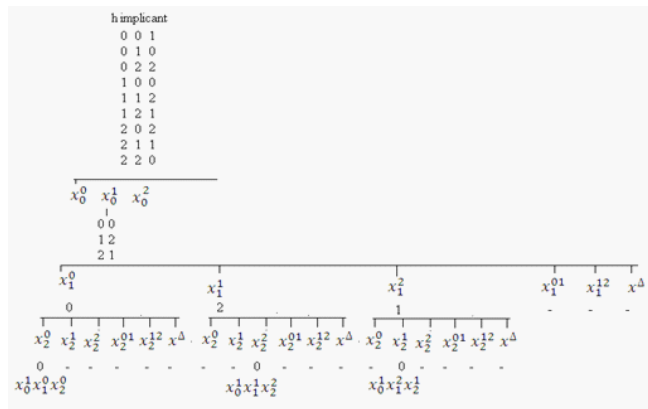
$$f = x_1^0 \cdot x_2^0 + x_1^0 \cdot x_3^0 + x_1^{01} \cdot x_2^0 \cdot x_3^2 + x_1^{01} \cdot x_2^{12} \cdot x_3^0 + x_2^2 \cdot x_3^0 + x_1^2 \cdot x_2^2 \cdot x_3^{01} + 1 \cdot [x_1^2 \cdot x_2^{12} \cdot x_3^{12} + x_1^{12} \cdot x_2^{12} \cdot x_3^2 + x_2^2 \cdot x_3^2]$$

Example 6) Consider full adder Table 4.3 for minimization by using modified Schiman binary method.

Truth table 4.3

x_0	x_1	$C_n = x_2$	Sum	Carry _{out}
0	0	0	0	0
0	0	1	1	0
0	0	2	2	0
0	1	0	1	0
0	1	1	2	0
0	1	2	0	1
0	2	0	2	0
0	2	1	0	1
0	2	2	1	1
1	0	0	1	0
1	0	1	2	0
1	0	2	0	1
1	1	0	2	0
1	1	1	0	1
1	1	2	1	1
1	2	0	0	1
1	2	1	1	1
1	2	2	2	1
2	0	0	2	0
2	0	1	0	1
2	0	2	1	1
2	1	0	0	1
2	1	1	1	1
2	1	2	2	1
2	2	0	1	1
2	2	1	2	1
2	2	2	0	2





In this example the procedure yields all the implicants participating in the minimal-cost representation obtained previously and will therefore lead to the same solution. However, this is not true in general.

Ternary switching appears especially promising connection with control circuits, coding circuits as special purpose computers. Further investigations as the advantage of such application are recommended.

There are other minimization methods described by [5-7]. T .C. Yang & A. S. Wojcik have suggested an algorithm for the minimization of ternary switching function which gives all prime implicants and a minimal cost static and dynamic hazard free ternary switching function. P.W. Besslich [6] described a heuristic method to obtain near minimal covers of multivalued switching function. S. C. Kleen [7] has described Algebra for regular ternary logic function. Y. Yamamoto and M. Mukaidono [8] considered special ternary functions for treating ambiguity in circuits.

References:

1. Caldwell SH. Switching Circuits and Logical Design. New York: Wiley. 1958.
2. McCluskey EJ, Bartee TC. A Survey of Switching Circuit Theory. New York: McGraw-Hill. 1962.
3. Yoeli M, Rosenfeld G. Logical Design of ternary switching circuits. IEEE Trans Computer. 1965; 14: 19-29.
4. Scheinman AH. A method for simplifying Boolean function. The Bell system technical journal. 1962; 1175-1480.
5. Yang TC, Wojcik AS. A minimization Algorithm for ternary switching functions. Tutorial on MVL minimization.
6. Besslich PW. Heuristic Minimization of MVL functions. A Direct Cover Approach. IEEE Tran On Computer. 1985; 35: 130-143.
7. Leen KSC. Introduction to mathematics. Amsterdam: North Holland. 1952; 332-340.
8. Yamamoto Y, Mukaidono M. Meaningful Special classes of Ternary logic functions Regular Ternary Logic Functions –Regular Ternary Logic Functions & Ternary Majority Function Proc. IEEE Tran Computer. 1988; 37: 799-806.