This article is under the direction of Prof. Chi-Tsong Chen. The author especially wish to thank Prof. Min-Wen Du for his helpful discussions and encouragement.

Appendix Superposition principle

An example is given here to explain the superposition principle. For the same machine used in example 2, case 1

the impulse response response is 101010.....

If the input is given as follows:

```
          1st impulse
          :  2nd impulse
          :  :
Input     0001000010001001001 00
          :  :  :  :  101 --responding sequence due to the 5th impulse
          :  :  :  101010
          :  :  101010101 --responding sequence due to the 3rd impulse
          :  1010101010101
          101010101010101010 --responding sequence due to the 1st impulse
Output    00010101111101000 0101
```

By adding the five responding sequences, we find the output sequence. It is the same as the output sequence in the checking sequence.

References

1. Zvi Kohavi: "Switching and Finite Automata Theory", New York, McGraw-Hill, 1970, pp. 530, Computer Science Series.
2. F. C. Hennie : "Fault Dectecting Experiments for Sequential Circuits" Proc. Fifth Ann. Symp. Switching Circuit Theory and Logical Design, Princeton, N. J., November, 1964. pp 95-110.

最少無靜態障礙組合邏輯線路之計算機傾向合成法

A Note on Computer Oriented Synthesis of Minimum Static Hazard-Free Combinational Logic CKT

陳 正 Cheng Chen

Department of Computer Science, N.C.T.U.

(Received August 12, 1976)

ABSTRACT — Hazard phenomenon is one of the most important problem in digital system design. It is due to unequal delay time of each gate in combinational logic cKT. For any given switching function of within 4 variables, we can detect and eliminate it from Karnaugh map easily [7,8]. If the number of variables of given switching function is more than four, then this method is tedius and time consuming. It will be shown that by using modified extraction algorithm, we can obtain minimum static hazard-free combinational logic conveniently.

Index terms: Static hazard in 1(0)'s, static hazard with kth order, *-product operation, #-operation, extremal of degree 1, extremal of degree k, Karnaugh map, tabular method, modified extraction algorithm, k-cube, minimum static hazard-free cover, prime implicant.

It is well known that hazard is one of the most serious problem in digital system design [1]. There are several kinds of hazards occurred in digital cKT, such as static hazard, dynamic hazard, function hazard and essential hazard etc [1]. Among them, Static hazard is the most basic one and often causes the malfunction of combinational logic or asynchronous sequential logic. Originally, the theory of static hazards has been developed by Huffman [2] and Mcclusky [3] who gave some basic mathematical framework of Boolean algebra about static hazard, Yoeli and Rinon [4] and Eichelberger used the ternary structure to study and detect the static hazard in combinational cKT. Abraham Kandel [6]developed an algorithm to minimize the static hazard free logic by using digraph structure. If the number of variables in any given switching function is large (e.g. more than 4), then all of these method are tedius and time-consuming. Dietmeyer [1] developed a powerful algorithmetic synthesis method to minimize the switching functions. In this paper we use *-operation to generate the set of all 1-cubes of the given switching function easily. Also, we define the extremal of degree 1 instead of extremal. And then develope an algorithm to generate all extremals of degree 1. Finally we use the modified extraction algorithm to select a minimum cost of irredundant prime implicants with the generated extremals of degree 1 to form a minimum static hazard-free cover of the given switching function. This method is easily to be processed by the computer and we will give the flow chart of the whole algorithms at the end of the paper.

Before going on, we make the following basic assumptions [1]:

1. The given switching function is in disjunctive normal form with single value i.e. it is given by the canonical sum of product form.
2. To minimize the given switching function is to minimize the number of terms and the number of literals in each term.
3. The complement of each variable in the given switching function is unavailable. In this paper, we use the symbol $\bar{x}$ to denote the complement form of literal x.

We will explain the concept of static hazard in the given switching logic briefly. At first, we give the following definitions of static hazard.

Definition 1. In the given combinational logic cKT having n input variable if some input variable changes cause the output value has temporarily incorrect value "0" (1) due to unequal delays in the cKT. Then we say that this combinational logic cKT has static hazard in 1(0)'s [7].

Definition 2. In the given combinational logic cKT having n input variables, we say that it has static hazard with kth order if this static hazard in 1's is due to only k input variables changes, $1 \leq k \leq n$.

Since the static hazard with 1st order is more often than not occurring in not only combinational logic but also in asynchronous sequential logic, we usually call it as static hazard briefly and the static hazard with kth order ($k \geq 2$) as high-order static hazard. In this paper, we only discuss about minimizing the static hazard-free logic. To minimize the high-order static hazard -free logic is easily extended by the same principle.

Basically, it is convenient to detect static hazard by using Karnaugh map [7,8]. Consider the following example:

Example 1. Given $f_1(x_1,x_2,x_3,x_4)=\bar{x}_1x_2\bar{x}_3\bar{x}_4+\bar{x}_1x_2\bar{x}_3x_4+\bar{x}_1x_2x_3\bar{x}_4+x_1\bar{x}_2x_3x_4+x_1x_2\bar{x}_3x_4$
$+x_1x_2x_3\bar{x}_4+x_1x_2x_3x_4=\Sigma m_4,m_5,m_6,m_{11},m_{13},m_{14},m_{15}$.

Its Karnaugh map is given by

x_1x_2 \ x_3x_4	00	01	11	10
00	0	0	0	0
01	(1)	1	0	(1)
11	0	1	(1)	1
10	0	0	1	0

Fig 1. Karnaugh map of f_1

As shown in Fig.1, we find that $f_1(1,1,1,1)=1$ and $f_1(1,1,1,0)$ is still equals to 1. But due to the delay of inverter in x_4, $\bar{x}_4$ will not go to 1 as fast as x_1 go to 0. So there will be a short temporary "o" at output. This is the incorrect output value for function f_1. If we indude the implicant $x_1x_2x_3$, then the output value of f_1 will be independent of the change of x_4 and this static hazard can be eliminated. By this way, we can detect all static hazard from Karnaugh map such that all those adjacent 1's are not included in minimum cover.

When the number of input variables is large (e.g. ten or more) the detection of all static hazard from Karnaugh map or by tabular method [7] is very tedius and time-consuming. In order to develope the algorithms, we give some basic important definitions [1].

Definition 3 Let $f(x_1,x_2,\cdots x_n)$ be any given switching function of n variables. The set of all minterms of f is called the base of f.

For example, $f(x_1,x_2,x_3)=\Sigma m_2,m_3,m_4$, the set $\{\bar{x}_1x_2\bar{x}_3,\bar{x}_1x_2x_3,x_1\bar{x}_2\bar{x}_3\}$ is the base of $f(x_1,x_2,x_3)$

Definition 4. Let $f(x_1,x_2,\cdots,x_n)$ be the given switching function of n variables. We can represent each element of the base of $f(x_1,\cdots,x_n)$ by n-tuple $(I_1,I_2,\cdots,I_n)$ such that $I_i=\begin{cases}0 \text{ if } \bar{x}_i \text{ appears in that term}\\ 1 \text{ if } x_i \text{ appears in that term}\end{cases}$

Briefly, we denote $I_1I_2\cdots I_n$ instead of $(I_1,I_2,\cdots I_n)$

Each n-tuple in the base of $f(x_1,\cdots x_n)$ is called one o-cube of $f(x_1,x_2,\cdots,x_n)$. For example, $\{\bar{x}_1x_2\bar{x}_3,\bar{x}_1x_2x_3,x_1\bar{x}_2\bar{x}_3\}$ is the base of $f(x_1,x_2,x_3)$ shown above. We can represent it as {010,011,100}, where 010, 011 and 100 are all 0-cubes of f $(x_1,\cdots x_n)$.

After some reduction of $f(x_1,\cdots x_n)$, if the kth literal disappeared in one term, then the kth position in its n-tuple will be replaced by x. For example, $f(x_1,x_2,x_3)=\bar{x}_1x_2\bar{x}_3+\bar{x}_1x_2x_3+x_1\bar{x}_2\bar{x}_3=\bar{x}_1x_2+x_1\bar{x}_2\bar{x}_3$. In this case, x_3 disappears in $\bar{x}_1x_2$ and we represent $\bar{x}_1x_2$ as 01x and it can be referred to as a 1-cube of f, Similarly, if there is k "x" in one cube, then we call it k-cube of f. By this way, we can represent any given switching function of n variables by a set of n-tuples with k-cubes etc., $0\leq k\leq n$. If k=n, i.e. all elements of n-tuple are "x", then it is called universal cubes of f, denoted as μ_n.

Definition 5. The ON-array and OFF-array of the given switching function $f(x_1,x_2,\cdots x_n)$ are the set of those cubes which map f to the value of 1 and 0 respectively. Similarly, the DC-array of f is the set of those cubes of which the value of f are undefined.

Definition 6. (*-product) Let $a=(a_1,a_2,\cdots a_n)$, $b=(b_1,b_2,\cdots b_n)$ be two cubes, we define *-product operation, a*b, as follows:

$$a*b=\begin{cases}\Phi \text{ if } a_i*b_i=\Phi \text{ for more than one i, } 1\leq i<n\\ c \text{ if } c_i=\begin{cases}a_i*b_i, \text{ where } a_i*b_i\neq\Phi\\ x \text{ when } a_i*b_i=\Phi, \text{ i.e. only one } \Phi \text{ in a*b.}\end{cases}\end{cases}$$

where a_i*b_i is defined as:

a_i \ b_i	0	1	x
0	0	Φ	0
1	Φ	1	1
x	0	1	x

For example, if a=0100, b=0111
then a*b=01ΦΦ=Φ
if a=0100, b=0101
then a*b=010x

Definition 7. (Absorb operation) Let a,b are two n-tuples, we define a.ABS.b as follows:

$$a.ABS.b=\begin{cases} b \text{ if } a_i.ABS.b_i=\varepsilon \text{ for all } i=1,\cdots,n \\ \phi \text{ if } a_i.ABS.b_i=\phi \text{ for some } i \end{cases}$$

where $a_i.ABS.b_i$ is defined in the following table.

a_i \ .ABS. \ b_i	0	1	x
0	ε	Φ	ε
1	Φ	ε	ε
x	Φ	Φ	ε

For example, a=01x b=011, a.ABS.b=ϕ, b.ABS.a=a

If a.ABS.b=b then we say that cube b covers cube a or cube a is covered by cube b.

Let $C=\{c_1,c_2,\cdots c_p\}$ be a set of cubes. a is another cube we define a.ABS.$C=\{a.ABS.c_1, a.ABS.c_2\cdots,a.ABS.c_p\}$. We also define, ABS(C), the absorb operation on set C to be the absorb operation between each other element in C.

For example,

$$C=\left\{\begin{matrix}000\\100\\0x0\\x10\\x11\\x1x\end{matrix}\right\} \qquad ABS(C)=\left\{\begin{matrix}100\\0x0\\x1x\end{matrix}\right\}$$

From this, let A,B be two sets of cubes, the cube union operation, denoted as $\cup$, is defined as $A\cup B=ABS(AUB)$ where AUB is the set union of A and B.

For example, $A=\{\begin{matrix}000\\x11\end{matrix}\}$, $B=\{\begin{matrix}0x0\\x1x\end{matrix}\}$

$$A \cup B=ABS\left\{\begin{matrix}000\\x11\\0x0\\x1x\end{matrix}\right\}=\{\begin{matrix}0x0\\x1x\end{matrix}\}$$

$$a\#b=\begin{cases} a \text{ if } a_i\#b_i=\Phi \text{ for some } i \\ \phi \text{ if } a_i\#b_i=\varepsilon \text{ for all } i \\ \bigcup_i(a_1,a_2,\cdots\bar{b}_i,\cdots,a_n), \text{ otherwise, where } a_i\#b_i=\alpha_i\varepsilon\{0,1\}.\forall_i \end{cases}$$

where $a_i\#b_i$ is defined as

Definition 8. (#-product) Let a,b be two cubes of n-tuples, we define the sharp operator, a#b, as follows:

	a_i \ b_i	0	1	x
	0	ε	Φ	ε
	1	Φ	ε	ε
	x	1	0	ε

For example, if a=xlx, b=010, then

$$\begin{array}{r} x1x \\ \#\ 010 \\ \hline 1\varepsilon 1 \end{array}$$

i.e. a#b={11x,x11}

if a=x10, b=x1x, then

$$\begin{array}{r} x10 \\ \#\ x1x \\ \hline \varepsilon\varepsilon\varepsilon \end{array}$$

i.e. a#b=Φ

We find that if a covers b then the result of a#b is the set of those cubes not covered by b but covered by a and if b covers a then the result of a#b is empty.

Now, we can generate all 1-cubes of the given switching function, by the following algorithm:

Algorithm 1 (1-cube generator)

Let ON be the ON-array of the given switching function and |ON| denotes the number of elements in ON.

For $i=1,2,3,\cdots,$ |ON| in turn

$j=i+1,\cdots,$ |ON| Do the following:

Form ON_i*ON_j

If $ON_i*ON_j\neq\Phi$, then $A\leftarrow A\cup\{ON_i*ON_j\}$

otherwise, continue this process.

where ON_i, ON_j are the ith and jth elements in ON.

Finally, the set of all 1-cubes in ON will be included in set A.

Example 2 As in example 1, $f(x_1,x_2,x_3,x_4)=\Sigma m_4,m_5,m_6,m_{11},m_{13},m_{14},m_{15}$

$$ON \left\{ \begin{matrix} 0100 \\ 0101 \\ 0110 \\ 1011 \\ 1101 \\ 1110 \\ 1111 \end{matrix} \right.$$

Perform the algorithm 1, we have

$ON_1 * ON_2 = 010x \quad ON_1 * ON_3 = 01x0$

$ON_1 * ON_4 = \phi \cdots$

Finally, we obtain

$$A = \left\{ \begin{matrix} x110 \\ 1x11 \\ 11x1 \\ 111x \\ 010x \\ 01x0 \\ x101 \end{matrix} \right.$$

which is the set of all 1-cubes in f.

In order to find the minimum static hazard free logic, we have to select a minimum number of prime implicants which cover the set of all generated 1-cubes. Now, we give the definition of the extremal of degree 1.

Definition 9. The prime implicant p of the given switching function f is called the extremal of degree 1 if it covers a 1-cube in A uniquely. Where A is the set of all 1-cubes generated by algorithm 1

By this definition, we find that an extremal of a given switching function is an extremal of degree o which covers a o-cube of f uniquely. In general, we can define the extremal of degree k ($0 \leq k \leq n$) as a prime implicant which covers a k-cube of f uniquely.

Example 3 Let $f(x_1, x_2, x_3, x_4) = \Sigma m_4, m_5, m_6, m_7, m_9, m_{13}, m_{14}, m_{15}$

Its Karnaugh map is given by

x_1x_2 \ x_3x_4	00	01	11	10
00	(1)	0	0	0
01	0	1	1	1
11	0	1	1	1
10	0	1	0	0

Fig.2. Karnaugh map of f_2

We find that the two 2-cubes x1x1 and x11x are extremals of degree 1 since they cover 1-cubes 01x1 and 011x uniquely and respectively.

THEOREM 1. Let $A=\{\alpha^1, \alpha^2, \cdots\}$ and $PI=\{\pi^1, \pi^2, \cdots\}$ be the set of all 1-cubes and prime implicants of the given switching function $f(x_1, \cdots, x_n)$ respectively. And $PI-\pi^i$ denotes that π^i is deleted from PI. We comput $e=\alpha^j \# (PI-\pi^i)$ for some $\alpha^j \varepsilon A$ and $\pi^i \varepsilon PI$, where $|PI|$ and $|A|$ are the number of elements in PI and A respectively. If $e \neq \Phi$ then π^i is the extremal of degree 1 covering the 1- cube α^j.

Proof: Consider $e=\alpha^j \# (PI-\pi^i)$ for some $\alpha^j \varepsilon A$ and $\pi^i \varepsilon PI$

If $e=\phi$, then by definition 8 α^j is not covered by $PI-\pi^i$. But each α^j must be covered by some prime implicants in PI. Then, π^i is the only prime implicant covering α^j. i.e. π^i is the extremal of degree 1 covering α^j.#

By this theorem, we obtain the following algorithm.

Algorithm 2 (Extremals of degree 1 generator)

Let $f(x_1, x_2, \cdots, x_n)$ be the given switching function

1. Form ON, OFF and DC of f
2. Generate the set of all 1-cubes of f by algorithm 1 and then store them in A.
3. Generate the set of all prime implicants of f by iterative consensus method [1] and then store them in PI.
4. For each $j=1,2,\cdots,|A|$ do the following:
   (i) For each $i=1,2,\cdots, |PI|$ do the following:
   (1) PI $PI-\pi^i$
   (2) $C \leftarrow \alpha^j \# PI$
   (3) If $C=\phi$, $PI \leftarrow PI \cup \pi^i$, $i \leftarrow i+1$ go to (1)
   If $C \neq \phi$, $B \leftarrow C \# DC$ and test B
   (4) If $B=\phi$, $PI \leftarrow PI \cup \pi^i$ and $i \leftarrow i+1$ go to (1)
   If $B \neq \phi$, $MC \leftarrow MC \cup \pi^i$, $DC \leftarrow DC \cup \pi^i$, $j \leftarrow j+1$
   go to (i)

In this algorithm, we pick up one element π^i from PI and perform the operation $\alpha^j \# (PI-\pi^i)$. If the result is empty, we restore π^i in PI. Otherwise we test $C \# DC$, where DC is to store those cubes of don't care and those selected extremals of degree 1. If the result of $C \# DC$ is not empty, then π^i is an extremal of degree 1. We store it in a set MC. Otherwise, we restore it in PI. Repeat this process for each π^i and α^j, we can generate every extremal of degree 1 iteratively.

Next, we will choose the minimum cost of prime implicants in addition to extremals of degree 1 to cover all 1-cubes in A and those 0-cubes of function f not covered by 1-cubes. At first, we define the following cost criteria [1]:

Definition 10. Let π^i and π^j be two cubes of n=tuple, we say that π^i is less than π^j if the following two conditions are satisfied:

(i) $\$(\pi^i) \geq \$(\pi^j)$, where the cost sign $\$$ is defined as $\$(\pi^i)=n-r$ when π^i is r-cube.

(ii) $(\pi^i \# DC) \# \pi^j = \Phi$, where DC is the result of algorithm 2.

By this definition, condition (i) is to show that the number of literals in π^i is more or equal to the number of literals in π^j, condition (ii) is to show that those cubes covered by π^i but not covered by DC are also covered by π^j. Then, we delete all "less than" cubes from PI and those 1-cubes covered by extremals of degree 1 from A. In the remaining cubes of PI we select the secondary extremals of degree 1. Here, the secondary extremals of degree 1 are those prime implicants in PI, each of which covers uniquely some remaining 1-cubes in A. And put those secondary extremals of degree 1 in MC 8 DC as well. Again, we delete the "less than" cubes in PI and repeat the process until the following three cases:

(i) the set A and PI are empty

(ii) the set A is empty and PI is not empty

(iii) the set A is not empty and there is no secondary extremals of degree 1

In cases (i) and (ii) we have found the minimum static hazard free cover and stop the process. In case (iii), a cyclic structure exists and we can solve it by using branching method as follows: [1]

Picking one of the remaining prime implicant in PI and

(i) assuming it is a part of the minimum cost cover, and proceeding to recognize less-than and extremals to find the remainder of the cover

(ii) assuming it is not a part of the minimum cost cover and proceeding to find a cover based upon this assumption.

Then the costs of the two covers are compared and the lower cost cover selected.

Now, we have the following algorithm.

Algorithm 3 (Modified extraction algorithm)

Let $f(x_1, x_2, \cdots, x_n)$ be the given switching function

1. Generate all 1-cubes of f by algorithm 1 and store them in A.
2. Generate all extremals of degree 1 from PI by algorithm 2 and store them in set MC and add them to DC.
3. Delete all extremals of degree 1 generated in 2. from PI and those 1-cubes in A covered by them.
4. Test the following cases:
   (i) If A and PI are all empty, then a minimum cover with static hazard-free is found in MC. Stop the algorithm.
   (ii) If A is empty and PI is not empty, then add these elements of PI in MC and MC is a minimum cover with static hazard-free. Stop the algo-

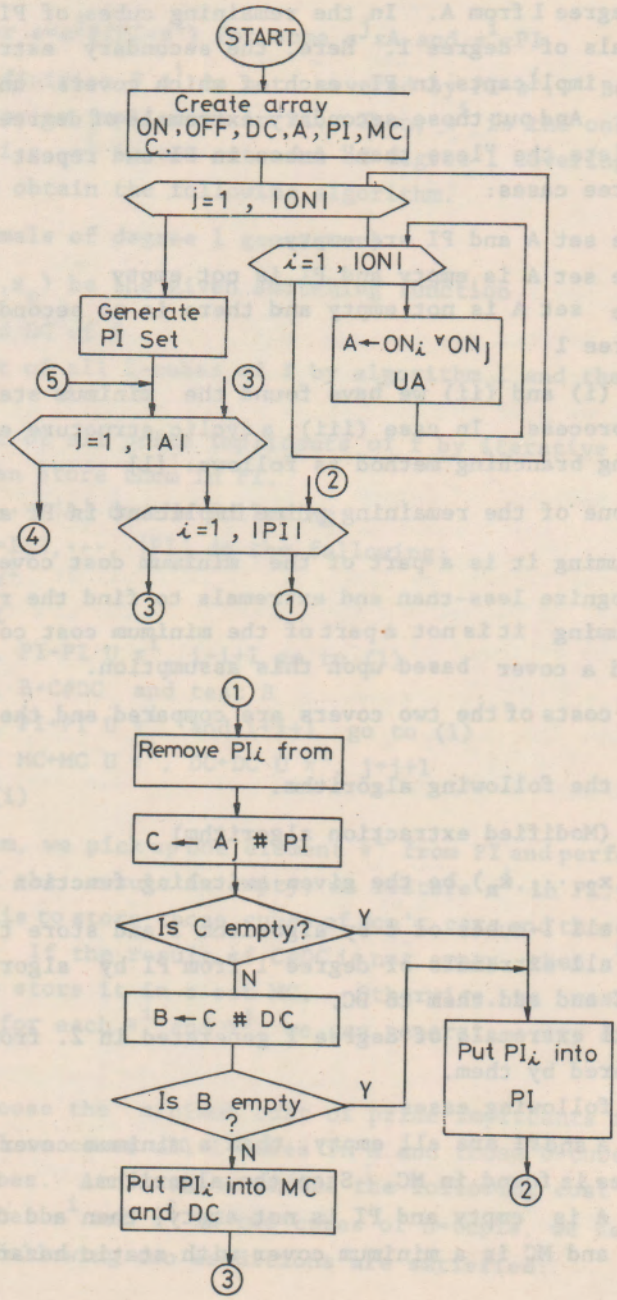
rithm.

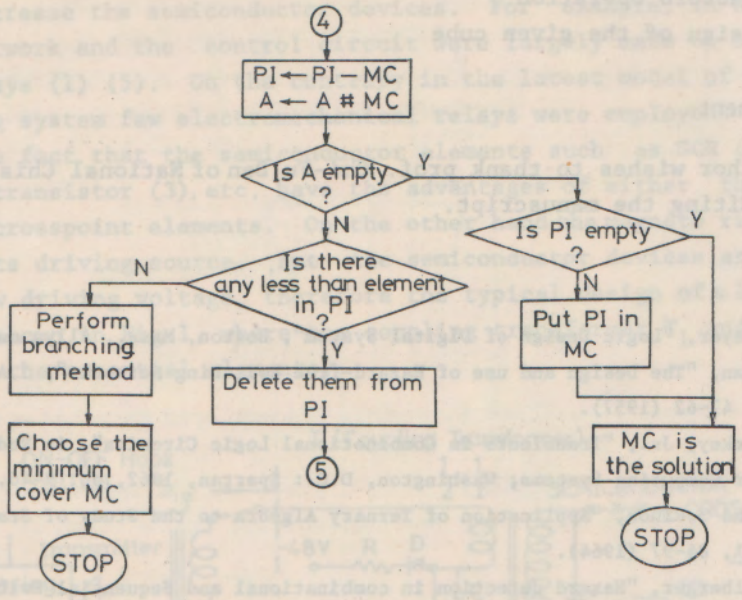
(iii) If A is not empty. then test the following cases:

(a) If one or more less than prime implicants exist in PI, delete them and then go to 2.

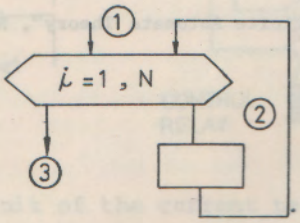
(b) If no less than prime implicant exists in PI, it is a cyclic structure and perform the branching method.

In summary, we give the following rough flow chart:





In this flow chart, we define the following diagram:



The above diagram is defined as the Do Loop structure where ①, ②, ③ are the entry, body and exit of the loop.

The whole algorithm has been implemented in DEC/System-10 by using FORTRAN IV language. The program can be used for educational purpose or some logic design applications.

List of Symbols

ON	ON-array of the given switching function $f(x_1,\cdots,x_n)$
OFF	OFF-array of the given switching function $f(x_1,\cdots x_n)$
DC	DC-array of the given switching function $f(x_1,\cdots x_n)$
*	star-product operation
#	sharp-product operation
U	set union

U cube union operation
.ABS. cube absorb operation
$ cost sign of the given cube

Acknowledgment

The author wishes to thank prof. Chi-Fu Den of National Chiao Tung University for editing the manuscript.

References

1. D.L.Dietmeyer, "Logic Design of Digital System", Boston, Mass., Allyn and Bacon, 1971.
2. D.A.Huffman, "The Design and use of Hazard-free Switching Networks", J.Ass. Compt. Mach., 4, 47-62 (1957).
3. E.J.McCluskey, Jr., "Transients in Combinational Logic Circuits", in Redundancy Techniques for Computing Systems, Washington, D.C.: Spartan, 1962, pp. 9-46.
4. M.Yoeli and S.Rinon, "Application of Ternary Algebra to the Study of Static Hazard", J. ACM. 11, 84-97 (1964).
5. E.B.Eichelberger, "Hazard detection in combinational and Sequential Switching Circuits" in Proc. 5th Annu. Symp. Switching Circuit Theory and Logic Design, 1964, pp. 111-120.
6. A.Kandel, "Note on Hazard Elimination", IEEE Trans. on Comput. 955-956 (Oct. 1973).
7. Hill & Peterson, "Introduction to Switching Theory and Logic Design", New York, John Wiley, 1974, pp. 394-398.
8. Zvi Kohavi, "Switching and Finite Automata Theory", New York, McGraw-Hill, 1970, pp. 202-206.

電話振鈴之改革—— 電子振鈴

An Innovative Ringing Technique in Telephone System – The Electronic Tone Ringer

張天鵬 Tian-Pong Chang

Department of Communication Engineering, N.C.T.U.

(Received November 11, 1976)

ABSTRACT – A new electronic tone ringer is presented. Basically the circuit of the new ringer can be drived by a 3.5 KHz ringing source, 3 volts in amplitude. The new ringer has the advantages over the conventional magneto bell ringer in the following respects. System reliability, flatness of frequency response, weight and cost. A prototype was built and tested with satisfactory results.

Since No 1 ESS_5 was developed a decade ago, it is of a general tendency