PAPER

An Optimum Half-Hot Code Assignment Algorithm for Input Encoding and Its Application to Finite State Machines

Yasunori NAGATA†, Masao MUKAIDONO††, *and* Chushin AFUSO†, *Members*

SUMMARY In this paper, a new optimum input encoding algorithm with m-out-of-$2m$ code which is called Half-Hot Code is presented. By applying Half-Hot Code to the input encoding in PLA-based digital system, the logic functions of the system turn out to be unate functions, thus, the number of bit-lines of PLA may be reduced [1]. The proposed method further reduces the number of product-lines of PLA optimally. In this code assignment procedure, computed Boolean subspaces satisfying suggested two conditions are assigned to each partitioned subset of digital input variables which are obtained by disjoint minimization or other techniques. As an experiment to evaluate the method, the state assignment for finite state machines of two-level implementation is considered. Specifically, the proposed Half-Hot Code assignment is compared with arbitrary Half-Hot Code assignment. The results show that the optimum encoding is superior to an arbitrary assignment up to about 24% in the number of product-lines of PLA.

key words: *finite state machine, half-hot code, input encoding problem, PLA, unate Boolean function*

1. Introduction

Half-Hot Code (HHC), in which the Hamming weight m is a half of the code length $2m$, is mainly utilized in fault tolerant technologies, and is well known as one of optimal unidirectional error detecting codes. Recently, D.J. Rosenkrants [1] pointed out that by applying HHC assignment to the PLA system the number of the bit-lines may be reduced considerably. Rosenkrantz also estimates the advantage of HHC encoding on the assumption that optimum HHC assignment is obtained.

In this paper, an optimum input encoding algorithm with Half-Hot Code is presented. The aim of the proposed method is to reduce not only the number of bit-lines but also the number of product-lines of the AND-plane of PLA optimally. The outline of the proposed strategy is to find Boolean subspace which fits to the classified input variables by using disjoint minimization or other techniques. These subsets of input are processed to suitable form for HHC encoding, and the Boolean subspaces must satisfy two conditions proposed here. Then the HHCs that are contained in the subspace are assigned to each input of a group. Since this method supposes the given code-length encoding, in case that constrained encoding can not be performed, appropriate unassigned HHCs are now assigned while preserving optimal cost of PLA.

The remainder of the paper begins with a background section containing some basic definitions. Section 3 describes the optimum input encoding algorithm. The results of comparison of the proposed and arbitrary HHC input encoding appear in Sect. 4.

Manuscript received June 25, 1994.
Manuscript revised May 10, 1995.

†The authors are with the Department of Electrical and Electronics Engineering, University of the Ryukyus, Okinawa-ken, 903–01 Japan.
††The author is with the Department of Computer Science, Meiji University, Kawasaki-shi, 214 Japan.

2. Background and Definitions

2.1 Half-Hot Code and Unate Functions

$\lceil m/2 \rceil$-out-of-m code or $\lfloor m/2 \rfloor$-out-of-m code is called optimal unidirectional error detecting code (UEC). In this paper, m-out-of-$2m$ code of which exactly a half of the total bits are equal to "1" is defined as Half-Hot Code (HHC).

Namely, the characteristics of HHC are as follows. (a) HHC is one of the unordered code which containes maximum number of codewords. In the case of HHC of code length $2m$, the maximum number of codewords n_{max} can be given by Eq. (1).

$$n_{max} = \begin{pmatrix} 2m \\ m \end{pmatrix} \cong 2^{2m}/\sqrt{\pi m} \tag{1}$$

(b) The Boolean function of which each minterm expressed by HHC can be always transformed into a unate function. This leads to an advantage in PLA implementation that each PLA bit-line needs only complemented or uncomplemented variables [1].

A unate function is a form in which only one of the two literals for each variable appears in the sum of products, for example $f = x_1\overline{x_2} + x_1x_3 + \overline{x_2}x_3$. In this paper, only positive functions which are constructed with uncomplemented variables are considered for simplicity.

Definition 1: Operation $\mathcal{F}\{x\}$ is defined as

$$\mathcal{F}\{x\} = \begin{cases} 1 & : x = 1 \\ * & : x = 0 \text{ or } * \end{cases}$$

where $*$ is either 0 or 1 (don't care).

Definition 2: Since each column of HHC corresponds to each Boolean variable respectively, a Boolean subspace which is obtained from HHC of code length $2m$

with operation $\mathcal{F}$ is said to be *minimal unate cube.* A minimal unate cube includes 2^m minterms. In general, a unate cube is the vector representation of a unate product term. Thus the variable dependency of a unate cube is m at most. Moreover, unate cubes which contain m' bits of 1's ($0 < m' < m$) and $2m - m'$ bits of $*$'s can be defined and described as $\{1, *\}^{2m}$: e.g. $\langle x_1, x_2, x_3, x_4 \rangle = \langle 1 * ** \rangle$ in code length four. A unate cube includes $2^{|*|}$ Boolean minterms, where $|*|$ is the number of don't care columns.

Note that the unate product term obtained from each $\begin{pmatrix} 2m \\ m \end{pmatrix}$ minimal unate cube refers to unate minterm. The unate minterm is the product term which is composed of m Boolean variables.

Definition 3: The expanded logical product operation "$\wedge$" and logical sum operation "$\vee$" which are said to be conjunction and disjunction, respectively, are defined on pseudo-Boolean elements $\{0, 1, *$ (0 or 1), ϕ (null character : neither 0 nor 1)$\}$ as follows [2].

$\wedge$	0	1	$*$	ϕ
0	0	ϕ	0	ϕ
1	ϕ	1	1	ϕ
$*$	0	1	$*$	ϕ
ϕ	ϕ	ϕ	ϕ	ϕ

$\vee$	0	1	$*$	ϕ
0	0	$*$	$*$	0
1	$*$	1	$*$	1
$*$	$*$	$*$	$*$	$*$
ϕ	0	1	$*$	ϕ

Theorem 1: $\mathcal{F}$ operation of an HHC or bit-wise disjunction of HHCs provides a unate cube.

Proof: It is an injection that each $\begin{pmatrix} 2m \\ m \end{pmatrix}$ minimal unate cube corresponds to each different unate minterm respectively. Since every HHC of code length $2m$ has $2^m - 1$ adjacent binary codewords which are not HHCs (i.e. adjacent minterm), each HHC composes a unique unate minterm respectively. From a disjunction of a combination of some unate minterms containing an HHC, a Boolean minterm cover is obtained, then a unate cube can be derived with $\mathcal{F}$ operation, e.g. $\mathcal{F}\langle 1100 \rangle \vee \mathcal{F}\langle 1010 \rangle = \mathcal{F}\{\langle 1100 \rangle \vee \langle 1010 \rangle\} = \langle 1 * ** \rangle$. Q.E.D.

To avoid a misapprehension, a terminology is defined. Let *inputs* to FSMs be said to be primary inputs to distinguish from "inputs" of *input encoding.* When the state assignment problem of FSMs are transformed into input encoding, regard the state variables of FSM as input variables of input encoding.

2.2 Disjoint Minimization

De Micheli [2] proposed an innovative technique for state assignment problem for finite state machines (FSM's). An FSM is implemented as a combinational logic system with memories and feedback. State assignment problem is known as an NP-complete problem, i.e. the state to be encoded are both inputs as well as outputs of the combinational logic. Traditional techniques [5] try to estimate the effect of encoding on the possible simplification of combinational component. In the procedure of [2], De Micheli applies logic minimization before the code assignment, and this is called *disjoint minimization.* Disjoint minimization is able to transform the state assignment problem into a constrained input encoding. The overview of disjoint minimization is as follows.

The state transition table of a bound filter for an example

S	$I=0$	$I=1$
s_1	$s_1/0$	$s_2/0$
s_2	$s_1/0$	$s_3/0$
s_3	$s_1/0$	$s_4/1$
s_4	$s_5/3$	$s_4/1$
s_5	$s_3/1$	$s_4/1$

can be represented by the symbolic cover illustrated below.

I	S	S'	Z
0	s_1	s_1	0
0	s_2	s_1	0
0	s_3	s_1	0
0	s_4	s_5	1
0	s_5	s_3	1
1	s_1	s_2	0
1	s_2	s_3	0
1	s_3	s_4	1
1	s_4	s_4	1
1	s_5	s_4	1

The symbolic cover is the arrangement of primary inputs, present states, next states, and outputs. Thus the symbolic cover is described as a four tupple (I, S, S', Z). Then, express each S and S' by positional cube notations:

I	S	S'	Z
0	10000	10000	0
0	01000	10000	0
0	00100	10000	0
0	00010	00001	1
0	00001	00100	1
1	10000	01000	0
1	01000	00100	0
1	00100	00010	1
1	00010	00010	1
1	00001	00010	1

Positional cube notation is the 1-hot code where the column corresponding to the state number is hot in which code length equals the number of states : N_s. Compress this cover by using multiple-valued Boolean minimization technique, then

I	S	S'	Z
0	11100	10000	0
0	00010	00001	1
0	00001	00100	1
1	10000	01000	0
1	01000	00100	0
1	00111	00010	1

can be obtained. The present states S are grouped for each (I, S', Z) symbols independently, i.e. each symbolic implicant of the cover is (I, S', Z)-disjoint.

The result of disjoint minimization provides the Constraint Matrix (C.M.) denoted as **M** from entry for S of the above table.

$$\mathbf{M} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

The column of **M** corresponds to each input (each state in this example) and the row corresponds to input groups (or state groups). **M** indicates constrained relation of states (or inputs). Namely, **M** provides that the states (or inputs) in each state (input) group should be assigned to the codewords that is included in a minimal cube. After the disjoint minimization, state assignment problem for FSMs is transformed to a constrained input encoding in which each row $\mathbf{a}_i.$ of **M** should be encoded to have minimal cost. The number of kinds of the product terms of multiple output functions is equal to the number of product-lines and this number is used as cost of PLAs.

3. HHC Optimum Input Encoding Algorithm

In this section, the optimum input encoding algorithm using Half-Hot Code is formulated. The algorithm consists of reduction of **M** and optimum encoding procedures. Also the optimality of this algorithm and how the number of bit-lines decreases are discussed.

3.1 Extended Constraint Matrix

As a preliminary task, the reduction procedure applying to a constraint matrix for efficient encoding is presented. The procedure is reduction of the number of rows in the constraint matrix without losing any information of optimum encoding with HHCs. The reduction procedure avoids some redundant decisions and iterations in the program.

The derived constraint matrix is denoted as $\mathbf{M}^R$, and the number of rows which indicates the number of input groups is denoted n_g.

≪ *Operations to derive* $\mathbf{M}^R$ ≫

op.1 Remove the rows with all 1's from **M**.

op.2 In two rows $\mathbf{a}_i.$ and $\mathbf{a}_j.$ of **M**, if $\mathbf{a}_j.$ includes $\mathbf{a}_i.$ ($\mathbf{a}_i. \leqq \mathbf{a}_j.$) then remove $\mathbf{a}_i.$ from **M**. (The duplicated rows are also deleted in this operation.)
Note that the $\mathbf{a}_i. \leqq \mathbf{a}_j.$ means $a_{ik} \leq a_{jk}$ for all corresponding elements in the $\mathbf{a}_i.$ and $\mathbf{a}_j.$.

op.3 Sort each row of **M** after enumerating the weight of each row : $|\mathbf{a}_i.| = w1(i)$ where $i = 1$–n_g.
($w1(i)$ provides the number of 1s in the i-th row.)

op.4 If $a_{ik} = 1$ and $a_{jk} = 1$ then change a_{jk} to 2, since unate spaces $\mathbf{u}_j$ and $\mathbf{u}_i$ can *intersect* for all $1 \leqq i < j$, where $|\mathbf{a}_i.| \geqq |\mathbf{a}_j.|$.
Note that an HHC assignment which includes intersections can not always be found for a given code length.

op.5 Remove such rows with all 2's from **M**, and enumerate $w2(i)$: the number of 2's of each row.

After all, these operations derives a n_s(:number of inputs or states) × n_g(:number of groups) matrix $\mathbf{M}^R$ whose elements are 0, 1 or 2, and the $\mathbf{M}^R$ is composed of the minimum cardinarity of rows covering all inputs.

Example-1: **M** of bound filter in 2.B is now reduced to:

$$\mathbf{M}^R = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 2 & 1 & 1 \end{bmatrix}$$

and $n_g = 2$.

The input that belongs to two or more groups, for example s_3 of example-1, is referred to as *intersection*. The $\mathbf{M}^R$ is the extended constraint matrix that has some further information of intersections for optimum encoding.

The goal of input encoding for PLA-based digital systems is to find an input encoding corresponding to a PLA implementation of minimal area.

For this aim, two concepts are useful [2].

(i) Find the encoding of minimum code length among the encodings that minimize the number of product-lines of the PLA.

(ii) Find the encoding that minimizes the number of product-lines of PLA among encodings of a given code length.

Both literatures [2] and [3] assume the above (i) for natural binary coding, and therefore the code length is increased by 1-bit at a time. Increasing 1-bit at a time is also possible in the case of HHC assignment. However, the positions of $m+1$-out-of-$2m+1$ or $m+1$-out-of-$2m+2$ codewords are modified from m-out-of-$2m$ in a Boolean space. In short, the above encoding can not be performed continuously and the hardware cost will be

inferior to the optimum natural encoding with situation (i).

From the discussion in [1], adding two more bits to a natural coding increases the number of bit-lines by 4 (by 6 in FSMs), and multiplies the number of inputs that can be encoded by 4. Adding two more bits to an HHC encoding increases the number of bit-lines by 2 (by 3 in FSMs) where m is the new number of code length and multiplies the number of inputs that can be encoded by $4(1-1/\lceil m/2 \rceil)$, which can be derived from:

$$\frac{\binom{2n+1}{n}}{\binom{2n}{n}} \cdot \frac{\binom{2n+2}{n+1}}{\binom{2n+1}{n}} = 2(1-1/(n+1)) \cdot 2$$

$$= 4(1-1/(n+1))$$

Therefore, when optimum HHC encoding is obtained, the PLA cost may be reduced certainly.

3.2 HHC Optimum Encoding

A constraint matrix M (or $\mathbf{M}^R$) provides that the states (or inputs) in a state group (input group) in the M should be encoded to the codewords which contained in a cube. Given a state group and a state assignment, the *face* is the minimal cube (the minimal subspace) containing the assigned codewords. In a constrained relation, *face of each group does not intersect the codeword assigned to any state (or input) not contained in the group.* Incidentally, a constraint matrix M which expresses a constraint relation provides a minimal symbolic cover. Thus a code assignment satisfying the constraint relation gives as many as implicants as the minimal symbolic cover.

Since the proposed HHC optimum encoding procedure here stands on (ii) with code length given by Eq. (1), there are code assignments which do not satisfy all of the constrained relation. However, the procedure aims to obtain the minimal cost of PLA restraining the number of encodings, that is, the number of assigned codewords which do not satisfy the constrained relation.

≪ *Encoding Algorithm* ≫

$\{ac\}$: set of the assigned codewords.

$\{dc\}$: set of the codewords included in the unate subspaces which have been accepted in step2, where {dc} is not contained in {ac} : $\{ac\} \cap \{dc\} = \phi$.

$\{ac\}^*$ **and** $\{dc\}^*$, which appear in step2 first, are intially ϕ respectively.

step1. Generate all HHCs of $2m$-code-length and denote this codeword set {hhc}. Initialize both {ac} and {dc} to empty sets: $\{ac\} = \phi$, $\{dc\} = \phi$.

step2. For a row $\mathbf{a}_j$. ($j = 1$–n_g) in a constraint matrix $\mathbf{M}^R$, find a unate cube (unate space) $\mathbf{u}_j$ which satisfies conditions (c1) and (c2) below, where $\mathbf{u}_j \in \{1, *\}^{2m}$.

$$(c1) \quad each\ element\ of\ \{ac\} \notin \mathbf{u}_j - \{ac\}^* \quad (2)$$

$$(c2) \quad w1(j) \leqq \begin{pmatrix} 2m-|\mathbf{u}_j| \\ m-|\mathbf{u}_j| \end{pmatrix} - |\{dc\}^*| - |\{ac\}^*| \quad (3)$$

where $\{ac\}^*$ is the set of HHCs assigned to states (inputs) of $a_{jk} = 2$ and $\{dc\}^* \equiv \{dc\} \cap \mathbf{u}_j$. Note that $|\mathbf{u}_j|$ is the number of unate minterms in a unate cube $\mathbf{u}_j$.
Iterate step2 to find $\mathbf{u}_j$ containing as many the number of HHCs as possible in the range of $w1(j) \leqq |\mathcal{C}| \leqq w1(j) + w2(j)$, where $|\mathcal{C}|$ is the number of HHCs.

step3. Assign codewords $\{\mathcal{C}\} \in \mathbf{u}_j$ to the states (inputs) in $\mathbf{a}_j$., where $\mathcal{C} \notin \{dc\}$ and $\mathcal{C} \notin \{ac\}$.

step4. Let $\{ac\} := \{ac\} + \{\mathcal{C}\}$, $\{dc\} := \{dc\} + \{d\}$, where $d \in \mathbf{u}_j$ and $d \notin \{ac\}$. Also, add the HHC (s) assigned to states (inputs) of $a_{jk} = 2$ to the $\{ac\}^*$, and update the $\{dc\}^*$ to $\{dc\} \cap \mathbf{u}_j$.

step5. If all inputs are encoded then stop, else let $\{hhc\} := \{hhc\} - \{\mathcal{C}\} - \{d\}$ and go to step2 with increment j for $\mathbf{a}_j$..

step6. **(6.1)** Assign a codeword $\mathcal{C}'$ which is contained in neither {ac} nor {dc} under no consideration of any constrained relation.

(6.2) Assign a codeword d in {dc} of $\mathbf{u}_i$ where i counts from $j-1$ to 1 since $\mathbf{u}_{j-1} \leqq \mathbf{u}_{j-2} \leqq \cdots \leqq \mathbf{u}_1$. Then the constrained relation of $\mathbf{u}_i$ will be canceled.

Reiterate step6. If all inputs are encoded then stop.

Procedures from step1 to step5 process the encoding satisfying constrained relation $\mathbf{a}_j$ ($j = 1$–n_g). However, if any constrained relation can not be satisfied because of the limitation of code length, then step6 is going to be executed. The program in pseudo-C language and the explanation of the procedure will be related below.
($|\mathbf{u}_i|$ is the number of HHCs in the unate cube $\mathbf{u}_i$.)

```
// OPEN: OPtimum input ENcoding
generate {hhc}; {ac}={dc}=φ;
for (a1.···a_ng.) {
    while (w2(i) >= 0) {
```

```
        candidates u_i;
        if(!(c1)) continue;
        if(!(c2)) continue;
        code assignment;          //C ∈ u_i
        goto cont1;
    }
    if (|u_i| < w1(i)) goto substep (6.1);
    substep (6.1); //minimal unate cube assign-
                   //ment
    if (n_s) break;
    if (candidates u_i = φ) goto substep (6.2);
    substep (6.2);    //{dc} code assignment
    goto substep (6.1);
    cont1:;
}
```

In step1, all HHCs of code length $2m$ are generated, and denote the set of codewords as {hhc}. The subset of assigned codes and the set of don't care codewords up to this iteration are denoted {ac} and {dc}, respectively. The don't care codeword {dc} is not used for encoding but is contained in $\mathbf{u}_k$ $(k = 1-j-1)$ which have appeared in step3. Step2 searches an appropriate unate space (unate cube) $\mathbf{u}_j$ which corresponds to a row of constraint matrix which represents the input set to be encoded, i.e. for the cardinality of input of $\mathbf{a}_j.$, $\mathbf{u}_j$ must be large enough to assign. To satisfy this condition, the number of codewords {dc} included in $\mathbf{u}_j$ represented by $|\{dc\}^*|$ and the number of codewords $|\{ac\}^*|$, which is intersection, should be substracted from the number of HHCs in $\mathbf{u}_j$. These two conditions are represented by Eqs. (4) and (5). The checking of (c1) proceeds as follows.

```
candidates u_j:
do{                          //checking (c1)
    5mm {C} ← {ac};
    {ac} ← {ac} − {C};
    if (C ∈ {ac}*) continue;
    if (C ∈ u_j) goto candidates u_j;
}while({ac} = φ);
checking (c2):
```

The $\mathbf{u}_j$ is seeked with conditions (c1) and (c2) iteratively for w2 (i) times.

In step3 the codewords included in $\mathbf{u}_j$ can be assigned to inputs of $\mathbf{a}_j.$ arbitrarily with the constraint relation. Step4 updates {ac} and {dc}. In step5, if all inputs are encoded then end, else the {hhc} will be updated and go to step2.

From step1 to step5, the unate faces satisfying conditions (c1) and (c2) are seeked to perform constrained encoding.

In step6, substep (6.1) assignes the remainder codewords not applied in constrained encoding of step1 to step5. In substep (6.2), if there is not any unused codeword stragling all over the $2m$ demension Boolean space, then assign {dc} codewords of $\mathbf{u}_i$ where i is from $j-1$ to 1. When $\mathbf{u}_j$ is processed in (6.2), the constrained relation of $\mathbf{u}_i$ is canceled.

Although the above procedure executes generation and testing, the following codeword inclusion check condition makes the procedure faster and simpler. A Half-Hot Code inclusion relation to a unate cube can be checked by Eq. (4) or Eq. (5).

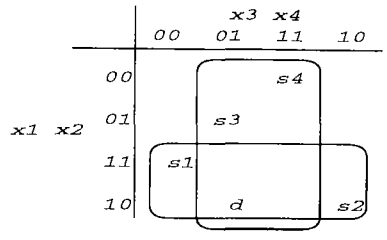
$$\mathcal{C} \notin \mathbf{u} \Leftrightarrow \mathcal{C} \wedge \mathbf{u} = \Phi \tag{4}$$

$$\mathcal{C} \in \mathbf{u} \Leftrightarrow \mathcal{C} \wedge \mathbf{u} = \mathcal{C} \neq \Phi \tag{5}$$

where $\mathcal{C}$ is an HHC and $\mathbf{u}$ is a unate subspace, "$\Leftrightarrow$" means iff and "Φ" is a no-codeword string. Operation $\wedge$ is conjunction which appeared in the previous section.

Example-2: From *example-1*, $\mathrm{M}^R = \begin{bmatrix} \mathbf{a}_1. \\ \mathbf{a}_2. \end{bmatrix} = \begin{bmatrix} 11100 \\ 00211 \end{bmatrix}$ is a constraint matrix. Then $\mathbf{u}_1 =< 1 * ** >$ can be obtained from $\mathbf{a}_1.$, since $\{ac\} = \phi \notin \mathbf{u}_1$ and w1 $(1) = 3 \leqq \begin{pmatrix} 4-1 \\ 2-1 \end{pmatrix} - 0 - 0 = 3$. Then $\begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} 1100 \\ 1010 \\ 1001 \end{bmatrix}$ is a codeword assignment. In the second iteration, $\mathbf{a}_i. = [00211], \{ac\} = \{1100, 1010, 1001\}$, $\{ac\}^* = \phi$ then $\{ac\}^* \notin \mathbf{u}_2 - < 1001 >$ and w1 $(2) = 2 \leqq \begin{pmatrix} 4-1 \\ 2-1 \end{pmatrix} - 0 - 1 = 2$. A unate cube excluding any codewords of $\{ac\} - \{ac\}^*$ is $< ***1 >$. Thus, $\begin{bmatrix} s_4 \\ s_5 \end{bmatrix} = \begin{bmatrix} 0101 \\ 0011 \end{bmatrix}$. Since this example does not perform step6, the encoding satisfies constrained relations completely.

Example-3: For $\mathrm{M}^R = \begin{bmatrix} 1100 \\ 0011 \end{bmatrix}$,



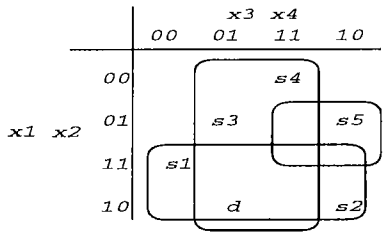
is obtained, where s_i $(i = 1 \cdots 4)$ are input symbols. The each loop in the above K-map shows a unate cube.

Example-4: For $\mathrm{M}^R = \begin{bmatrix} 110000 \\ 001100 \\ 000011 \end{bmatrix}$, the encodings of $s_1 \cdots s_4$ are similar to Example-3. To encode s_5, substep (6.1) is used and

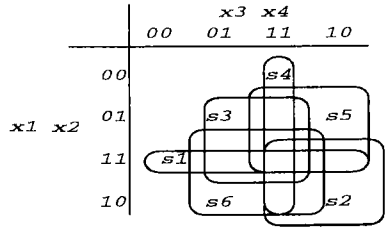
Table 1 The Experimental Results of HHC Assignment for FSMs.: The number of product-lines.

EXP.No. \ N_S		5	6	7	10	15	20	21	30	40	50	60	70	75	80
Ex.1	opt.	8	10	10	15	23	32	36	46	62	78	104	139	119	121
	arb.	8	12	13	19	28	38	41	60	78	98	119	139	150	160
Ex.2	opt.	8	10	9	16	22	37	35	46	61	78	102	138	117	126
	arb.	10	12	13	19	27	37	42	59	78	99	119	138	150	160
Ex.3	opt.	7	10	11	14	25	37	32	47	61	78	104	135	114	123
	arb.	7	10	12	17	29	39	40	59	79	97	120	139	150	160
Ex.4	opt.	8	10	10	15	25	34	33	45	62	78	105	139	119	126
	arb.	9	12	14	20	29	40	40	58	78	100	119	139	150	160
Ex.5	opt.	9	11	10	16	26	39	32	46	63	79	104	139	114	126
	arb.	10	11	12	18	30	39	42	58	78	99	117	139	150	160
MEANS	opt.	8	10.2	10	15.2	24.2	35.8	33.6	46	61.8	78.2	103.8	138	116.6	124.4
	arb.	8.6	10.4	12.8	18.6	28.6	38.6	41	58.8	78.2	98.6	118.8	138.8	150	160
USED CODE/ TOTAL CODE		5/6	6/6	7/20	10/20	15/20	20/20	21/70	30/70	40/70	50/70	60/70	70/70	75/252	80/252

opt.:optimum HHC encoding arb.:arbitrary HHC encoding



is obtained. To encode s_6, substep (6.2) is used and



is obtained. This result consist of six unate minterms (minimal unate cubes).

The optimality of the procedure is shown by (a), (b) and (c) below.

(a) procedures from step1 to step5 are almost the same constrained encoding as in [2] and [3]. These procedures seek the minimal dimention subspace containing the encoding of an input group. The HHC assigned to $a_{ik} = 2$ can be the joint of two or more faces for optimum HHC encoding.

(b) Step6-(6.1) is the procedure that assigned HHCs excluding $\mathbf{F}$, where $\mathbf{F} = \begin{bmatrix} \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_i \end{bmatrix}$ for $\mathbf{a}_i$.. The code assignment on this step does not satisfy constrained relations, but it preserves a constrained relations obtained in step1 to 5.

(c) Step6-(6.2) is the process that assigns codeword $d \in \{dc\}$. The code assignment on this step never satisfies constrained relation, and the constrained relation of $\mathbf{u}_i$ including d will be cancelled. To preserve optimality under a code length limitation, the d included in $\mathbf{u}_i$ is assigned to an input where $|\mathbf{u}_i|$ is as smaller as possible.

While this procedure completely or partly satisfies constraint relations, an optimum HHC encoding can be obtained.

4. Experimental Results

The experiments treated the problem of state encoding for finite state machines (FSMs). First, the state transition table of FSM is processed with disjoint minimization, and we have a new input encoding problem. Then the input encoding with HHC program is executed, which consist of proposed procedures. Finally, unate function optimizer program provides the number of product-lines of PLA designed here.

Disjoint minimization program, HHC optimum encoding and arbitrary HHC encoding program and unate function optimizing program are constructed in C-language on personal computer. In this case finite state machines are limited to Moore type, but it is easy to change to Mealy type in general. As input data, FSMs of 5–80 states and 2 primary inputs are applied. Arbitrary HHC assignment exploits the same code length that the proposed HHC assignment procedure has. (refer to Eq. (1).) The results of comparison between strategied encoding and arbitrary encoding are shown in Table 1, and the means are shown in Fig. 1 (b). Figure 1 (a), additionaly, shows the rate of utilization of codeword defined as

$r =$ (utilized codeword) / (total number of HHC) $\times$ 100 [%].

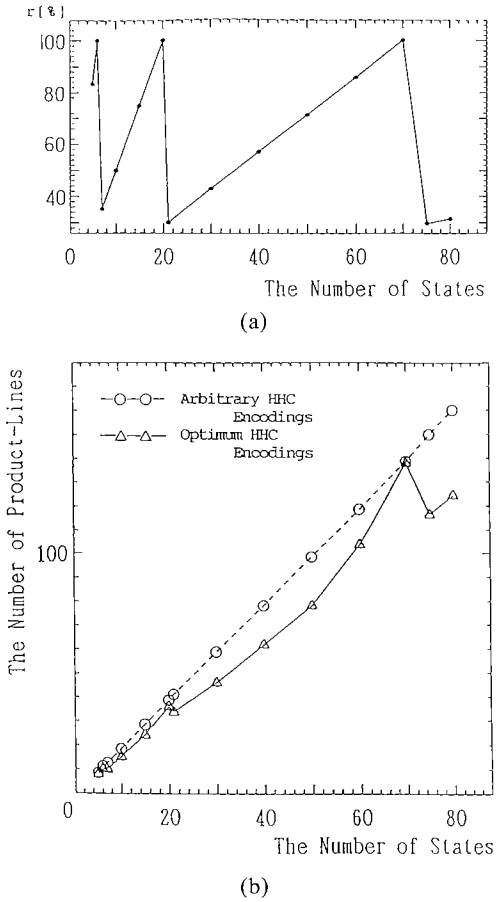


Fig. 1 (a) The number of states versus the rates of utilization of HHC. (b) Comparison between optimum HHC assignments and arbitrary HHC assignments.

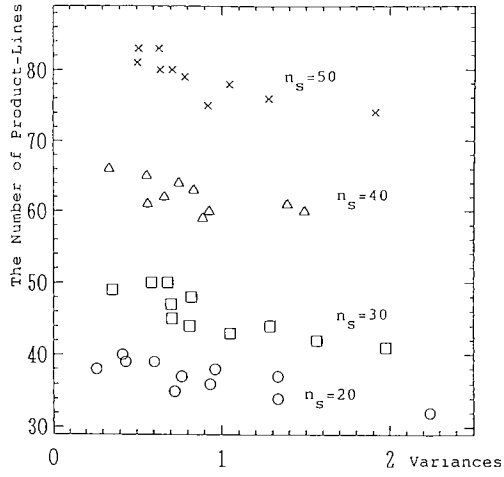


Fig. 2 The number of product-lines versus the variances of $|a_{i\cdot}|$ of $\mathbf{M}^R$.

From Fig. 1 (b), the number of product-lines with arbitrary encoding is almost in linear relation to the number of states. From Fig. 1 (a) and (b), in the range of high utilization of codeword, the number of product-lines of strategied encoding is very close to that of arbitrary encoding. As the extreme case of $r = 100\,[\%]$, both encoding essentially give the same results. If r is low, again the results of the both encodings are not so different. The reason for this is that the Boolean space in the case of low value of r is large comparatively. And thus, the corresponding unate product terms are well minimized regardless of code assignments.

Therefore, the proposed procedure is efficient in all ranges of r, especially in the range of $r = 40\,[\%]$–$90\,[\%]$ the effect of the proposed method is remarkable. The optimum encoding is superior up to about 24% to arbitrary encoding in this experiment.

Figure 2 shows the number of product-lines versus variances of the cardinalities of the states (inputs) in state groups (in input groups) of FSMs, i.e. horizontal axis indicates the variance of $\mathbf{a}_1. \cdots \mathbf{a}_{n_g}.$ in

$$\mathbf{M}^R = \begin{bmatrix} \mathbf{a}_1. \\ \vdots \\ \mathbf{a}_{n_g}. \end{bmatrix}.$$

The results of Fig. 2 show that the effectiveness of encoding has no close relation to the structure of FSMs (structure of state transition) in HHC optimum encoding. However, weak relations between the PLA-costs and the variances can be observed.

5. Conclusions

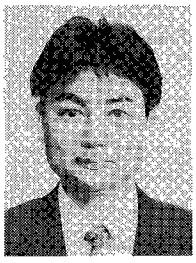
An optimum input encoding algorithm with Half-Hot Code (m-out-of-$2m$ code) is presented. To find an optimum encoding under the condition of minimum code length, the proposed algorithm consists of two parts; in the first part, constrained encoding is performed as far as possible, and in the second part unconstrained encoding is performed while holding down the increase of cost.

As an experiment, the proposed procedure was applied to state assignment of PLA-based sequential machines. An interesting aspect of the results of the experiment is that for arbitrary encoding the cost of hardware (the number of product-lines) is directly proportional to the number of states. In the range of $r = 40\,[\%]$–$90\,[\%]$ (rate of utilization of HHC) the effect of proposed optimum encoding is remarkable. Also the cost of the PLA is less dependent on the variance of each row of constraint matrix which represents the structure of machines.

References

[1] D.J. Rosenkrantz, "Half-hot state assignment for finite state machines," IEEE Trans. Compt., vol.39, no.5 pp.700–702, March 1990.
[2] G.D. Micheli, R.K. Brayton, and A. Sangiovanni-Vincentelli, "Optimal state assignment for finite state machines," IEEE Trans. CAD, vol.CAD-4, no.3 pp.269–285, Jan. 1985.

[3] S. Yang and S.J. Ciesielski, "Optimal and suboptimum algorithms for input encoding and its relationship to logic minimization," IEEE Trans. CAD, vol.10, no.1 pp.4–12, Jan. 1991.
[4] T. Sasao, "Tautology checking algorithm for multiple-valued input binary functions and their application," Proc. 14th Int. Symp. Multi. Val. Logic, 1984.
[5] Y. Nagata, C. Zukeran, and C. Afuso, "A state assignment for p-valued sequential machines," IEICE Trans. D-I, vol.J-72-D-I, pp.248–253, April 1989.



Yasunori Nagata was born in Japan in 1960. He received his B.S. and M.S. degrees in electronics and information engineering from University of the Ryukyus, Okinawa, in 1984 and 1987, respectively. He is currently an Assistant Professor in the Department of Electrical and Electronics Engineering, University of the Ryukyus. His main reserch interests are in multiple-valued logic, fault-tolerant systems and computer-aided logic design. He is a member of IEEE Computer, Circuits and Systems, and Information Theory Societies.

Masao Mukaidono was born in Japan in 1942. He received his B.S., M.S., and Ph.D. degrees in electrical engineering from Meiji University, Kawasaki, in 1965, 1968, and 1973, respectively. He is currently a Professor in the Department of Computer Science, Faculty of Science and Technology, Meiji University. His main research interests are in multiple-valued logic, fuzzy set theory and its applications, fault- tolerant computing systems, fail-safe systems, and computer-aided logic design. Dr. Mukaidono is a member of the IEEE Computer Society and Information Processing Society of Japan, and president of Japan Society for Fuzzy Theory and Systems.

Chushin Afuso received a B.E. from Yokohama National University, Yokohama, Japan, in 1956 and an M.S. and ph.D. in Electrical Engineering from the University of Illinois in 1961 and 1968, respectively. He has been with Washington University, the University of Illinois and Yamagata University. Heis currently a Professor at the University of the Ryukyus. His main interests is in digital and analog electronic circuits and multiple-valued logic circuits.