論文 95-8-8-10

Minimum Unicode One-Shot State Assignment and its Realization Using Graph-Theoretic Terms

Yong Jin Kwon*, Shuzo Yajima** Regular Members

그래프 이론적인 수법을 이용한 최소 단부호 단발 상태할당과 그 실현

正會員 權 容 珍*, 矢島脩三**

ABSTRACT

We clarify characteristics and present an optimization on the One-shot state assignment which has been known by the Hamming universal One-shot state assignment. It if possible to perform the One-shot state assignment for all asynchronous sequential machines by the Hamming universal One-shot state assignment. However, the assignment requires $2^{\lceil \log_2 n \rceil}-1$ state variables and 2m-1/m,m=[$\log_2 n$] binary vectors per state for a flow table with n states, even when [$\log_2 n$] state variables and only one binary vector per state are sufficient for the One-shot state assignment for the flow table. We first provide a necessary and sufficient condition for unicode One-shot state assignment, which is described by the recursive form of matrices. An algorithm for unicode One-shot state assignments which requires [$\log_2 n$] state variables is developed in graph-theoretic terms, where n is the number of internal states of a given flow table. Besides, we propose an upper bound on the number of gate inputs required for unicode One-shot state assignment realizations, and compare the upper bound with results in other papers. Finally, the topics in this paper are overviewed and several future works are discussed.

key words: *unicode One-shot, state assignment, minimum, asynchronous, sequential circuits, gracuits, graph-theoretic terms, hypercube embedding*

要 約

본 논문에서는 Hamming의 만능 단발 상태할당으로만 알려져 있는 단발 상태할당의 성질과 특징을 분명히 하고, 이

*한국항공대학교 통신정보공학과
Dept. of Telecomm. & Information Engeneering
Hankuk Aviation University
**Dept. of Information Science, Faculty of Engineering Kyoto University, Kyoto-Shi, 606-01, JAPAN
論文番號：95059-0210
接受日字：1995年 2月 10日

상태할당을 최적화하기 위한 수법을 제안한다. Hamming의 만능 단발 상태할당을 이용해서 n개의 상태를 갖는 비동기 순서회로에 단발 상태할당을 할 경우, 모든 경우에 대해서 $2^{\lceil \log_2 n \rceil}-1$개의 상태변수와 각 상태 당 $2^{m-1}/m, m=\lceil \log_2 n \rceil$개의 2진 벡터를 필요로 한다. 본 논문에서는 우선 단부호에 의한 단발 상태할당을 위한 필요 충분 조건을 명확히한다. 이 조건은 행렬의 제귀적인 형태로 주어진다. 또한 $\lceil \log_2 n \rceil$개의 상태변수만을 사용하는 단부호에 의한 단발 상태할당의 알고리즘을 제안한다. 나아가 이 단부호에 의한 단발 상태할당의 실현에 필요한 gate input의 갯수에 대한 상한(upper bound)을 제시하고, 다른 수법에 의한 상한과 비교, 검토하여 본 상태할당의 유효성과 실용성을 분명히 한다.

I. Introduction

Sequential machines play a major role in digital systems. Digitalcomputers are very complex examples of sequential machines and involvea combination of various sequential machines.

The computer-aided synthesis of sequential machines can be partitionedinto several tasks. This paper addresses the state assignment-problem. The state assignment problem can be stated as an optimizationproblem: find an assignment of binary vectors for the states of asequential machine that will minimize some costs of the resulting circuits.

The state assignment problem has been the object of extensivetheoretical research. Harmains [Har61], Karp [Kar64], andKohavi [Koh70] developed algebraic methods based on partitiontheory. Their approach was based on a reduced dependence criterion,which led to a good assignment. However, no theoretical result waspresented that related reduced dependencies to optimal sequentialmachine implementations. Moreover, no systematic procedure wasdeveloped that could be used to encode large sequential machines. However the computer-aided synthesis of sequential machines is aprerequisite for coping with the complexity of very large-scaleintegrated (VLSI) circuits.

Meanwhile, asynchronous sequential machines are widely used in many ofpresent day digital logic circuits. Even in synchronously designedcomputer circuitry, asynchronous sequential devices such as latches,triggers, counters, etc., are widely used together with combinationalelements to realize complex sequential functions. However, due to thepossible presence of races and hazards among asynchronous sequentialmachines, they are usually more difficult to design than synchronoussequential machines.

Furthermore, while every state assignment, which has a unique binaryvector for every internal state of a synchronous sequential machine,may yield a correct circuit realization, it is not a sufficientcondition for asynchronous machine realizations. The state assignmentfor asynchronous sequential machines must ensure that no criticalraces exist among the state variables [Yaj87] [Ung66]. We areconcerned with the state assignment for asynchronous sequentialmachines in this paper.

State assignments for asynchronous sequential machines can beclassified into two main categories: 1) multiple transition-time (MTT)state assignment and 2) single transition-time (STT) state assignment[Ung66]. Although STT state assignments generally require morestate variables than the MTT state assignment. STT state assignmentslead to faster circuits. For this reason. STT state assignments havebeen extensively studied [Yaj87]. [Ung66]. [Tra66]. [Liu63]. The STT stateassignments have been earlier classified into 1) unicode and 2)multicode state assignment [Yaj87]. [Ung66]. Although it isknown that multicode STT state assignments may require fewer statevariables than unicode STT state assignments. no general techniques toderive efficient multicode STT state assignments areknown [Ung66]. One of known general techniques is. due toHuffman [Huf55]. named the Hamming universal One-shot stateassignment. and requires$(2^{\lceil \log_2 n \rceil}-1)$state variables for asynchronous sequential machines with n states.where $\lceil x \rceil$ is the least integer greater than or equalto x.

In this paper. we clarify characteristics and present an optimization on the One-shot stateassignment which has been known by the Hamming universal One-shotstate assignment. It is possible to perform the One-shot stateassignment for all asynchronous sequential machines by the Hamminguniversal One-shot state assignment. However. the assignment requires $(2^{\lceil \log_2 n \rceil}-1)$ state variables and $(2^{m-1}/m, m=\lceil \log_2 n \rceil)$ binaryvectors per state for a flow table with n states. even when $(\log_2 n)$ state variables and only one binary vectorper state are sufficient for the One-shot state assignment for theflow table.

We first provide a necessary and sufficient condition for unicode One-shot state assignments. which is described by the recursive formof matrices. An algorithm for unicode One-shot state assignments which requires $\lceil \log_2 n \rceil$ state variables is developedin graph-theoretic terms. where n is the number of internal statesof a given flow table.

Besides. we propose an upper bound on the number of gate inputsrequired for unicode One-shot state assignment realizations. andcompare the upper bound with results in otherpapers [Tan71].

This paper is organized in the following manner: In Chapter 2. stateassignments for asynchronous sequential machines are brieflyintroduced and the notation to be used and the assumption being madeis contained. In Chapter 3. we explain the One-shot state assignmentin short. and a necessary and sufficient condition for unicodeOne-shot state assignments and a exhaustive algorithm are proposed.and the number of state variables required by the proposed algorithmis shown. Chapter 4 are allowed itsspace for representing an upper bound on the number of gate inputsrequired for One-shot state assignment realizations.In Chapter 5. the conclusion of this paper and future problems are stated.

Ⅱ. Preliminaries

This chapter introduces basic definitions and assumptions which will benecessary later in the paper. and overviews state assignments ofasynchronous sequential machines

1. Basic Concepts

Some assumptions being made.basic notations to be used. and definitions ofseveral terms are given in this section.

A *sequential machine of Moore type (Mealy type. respectively) M* is a 6-tuple $(S, \Delta, \Gamma, \delta, \lambda, S_{init})$. S is a finite set of states. Δ is an alphabet from which input symbols are chosen. Γ is alphabet from which outputsymbols are chosen. δ is a state transition functionwhich maps from $S\times\Delta$ to λ. is an output function which maps from S to Γ $(S\times\Delta$ to $\Gamma)$ and S_{init} is the set of initial states. The sequential machines considered in this paper are asynchronousand of the Mealy ones. A sequential machine can be also defined by a flow table that gives the state transition (next state) function and the output function in a table form, as shown in Fig.1.

A circuit representation of an asynchronous sequential machine to bedesigned is shown in Fig.2. The $\triangle_i$s in this figure are the inserted delays in the feedback paths.The following assumptions are made on asynchronoussequential machines (for which the state assignments are to bemade).

a) The asynchronous sequential machines are represented by *normal mode flow tables* [Ung66] (for normal mode flow tables the next state entries are equal to the next stable states or, equivalently, each unstable state leads directly to a stable state).

b) The asynchronous circuit realizing the given normalmode flow table is to operate in fundamental mode: em *i.e.*,the inputs to the circuit are not changed until it is a stablestate.

c) The delays in the feedback paths in Fig.2 are adequate toeliminate hazards [Ung66]. Assumptions a)~c) have to be satisfied for the generalapplication of most of the known results on asynchronous state assignments [Ung66], [Huf55]. Elimination of delays is possible under certain conditions [Ung68], [AFM68].

Definition 2.1 *stable state*

A state $s_i \in S$ is said to be a stable state under input $i_j \in I$ if and only if $\delta(s_i, i_j) = s_i$. □

Definition 2.2 *race*

Such a situation whereby more than one state variable mustchange in the course of a transition is called a race. □

Definition 2.3 *critical race*

Such a situation, where the final stable state reached by thecircuit depends on the order in which the state variables change, is referred to as a critical race. □

An example of the critical race is illustrat-

X / Q	x = 0	x = 1
a	b, 0	c, 1
b	a, 0	d, 1
c	d, 0	c, 0
d	c, 0	a, 0

Figure 1. A flow table

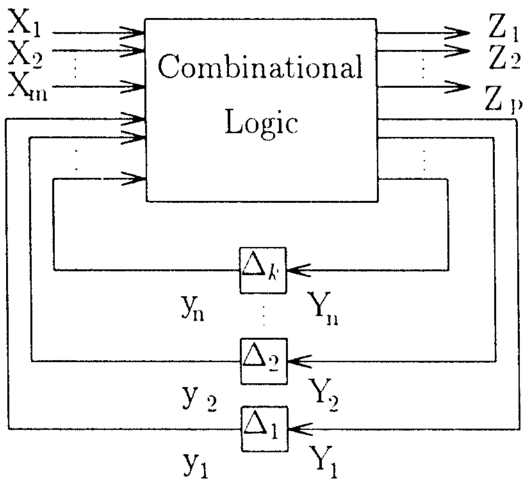


Figure 2. Circuit representation of an asynchronous sequential machine

ed in Fig.3, which should be considered as a portion of a larger flow table. Thetransition from state 4 to state 7 involves a race between y_1 and y_2 which is critical since if y_1 wins, the system will stop in state 6. However, correct operationresults if y_2 wins, as well as if the race ends in a tie.

Definition 2.4 *state assignment*

A state assignment is defined that of assigning statesof binary-valued state variables (binary vectors) to the states (rows) of a flow table

2. State Assignments for Asynchronous Sequential Circuits

This section overviews state assignments for asynchronous sequentialcircuits.

Having arrived at a flow table that satisfactorily describes the desired sequential function, we must, as the next step in the synthesis, select a finite set of binary state variables and assign to each row of the flow table one or more states of these variables. Once this has beenaccomplished, it is a straightforward matter to obtain truth tablesspecifying the circuit outputs and next values of the state variablesas functions of the current values of the input and state variables.

However, while every state assignment, which has a unique binary vector for every internal state of a synchronous sequential circuit, may yield a correct circuit realization, it is not a sufficient conditionfor asynchronous sequential circuit realizations. The state assignment for asynchronous sequential circuits must ensurethat no critical races exist among the state variables.

As a result of the above discussion, it should be evident that the stateassignment problem is by no means a simple one. Not only must each row be assigned a unique code, but the codes must be so interrelated that notransition involves a critical race. It is desirable to achieve this endwhile minimizing transition time(that is, the number of steps for eachtransition), the number of state variables, and the complexity of the resulting logic circuit.

		$y_1\,y_2\,y_3$
4	7 , 0	0 0 1
5	7 , 0	0 1 1
6	⑥, 1	1 0 1
7	⑦, 0	1 1 1

Figure 3. An example of critical race

Q \ X	X_1	X_2	X_3	X_4
1	①	2	①	5
2	3	②	1	5
3	③	4	1	③
4	5	④	1	④
5	⑤	4	1	⑤

Figure 4. A flow table

State-assignment techniques for asynchronous sequential machines can be classified intotwo main categories: 1) multiple transition time(MTT) state assignments and 2) singletransition time (STT) state assignments. However, STT state assignments are mainly concernedin this paper and described here.

Single Transition-Time assignments

Speed of operation is often an important considerationin the design of switching circuits, and it is notunusual to pay for it with significant increases inthe number of components used. A key factor influencing the speed of operation of asynchronous sequentialcircuits is the maximum number of transition times requiredfor an inter-state transition(i.e. a transition betweenstates of a flow table). State assignments for which a single transition time is always sufficient for any transition are called single transition-time (STT) state assignments. STT assignments may involve simultaneous changes in several state variables for certaintransitions, where transitions will occur through noncritical races among all the variablesdistinguishing the initial and final states.

The STT assignments consist of unicode STT assignments and multicode STT assignments. The unicode STT assignmentswere first introduced by Liu [Liu63] and later further developed by Tracey [Tra66] in a more sophisticated way.In these assignments, a single binary vector is assigned to each state ofthe flow table, and all state variables which must change in a given transition are allowed to change simutaneously without critical races.

A *universal* unicode STT assignment, also called the (2,2) separating system, is a way to give a valid unicode STT assignment to anyasynchronous sequential machine of an arbitrary number of states *regardless of* the configuration of its flow table.The study based on the (i, j) separating systems for constructing universal unicode assignments originated fromthe work of Friedman *et al.* [FGU69]. So far, the (2,2) and (2,1) separating systems were investigated by them [FGU69], Mago[Mag69], [Mag70], and Pradhan *et al.* [PR76].

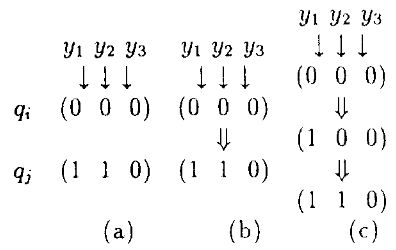
If more than one binary vectors are assigned to someor all states of a flow table, it may be possible toobtain STT assignments with fewer state variables thanunicode STT assignments, which we refer to as multicode STT assignments. The multicode STT assignment, tailored for a specific flow table with fewerstate variables than the best unicode STT assignment,was originally developed by Frieds [Ung66]. A universal multicode STT assignment for a specificnumber of states was given in a rather specialized way by Friedman *et al.* [FGU69].

Recently, a method has been given for constructing multicode STTassignments which requires $2 \cdot m$ state variables for 2^m states under the assumptions that the set of input states is divided intotwo disjoint subsets and all allowed input transitions occur from elements of one subset into elements of the other subset [KR78].

Reference [Ung69] gives a general background for this paper.

Multiple Transition-Time assignments

For all transition from a state s_i to s_j in a flow table, if it is possible to assigna binary vector to s_i such that the binaryvector is adjacent to a binary vector of s_j ,a state assignment which has no critical racesis obtained. However this is not always possible byone binary vector per state. For example,in



(a)An inter-state transition (b)STT (c)MTT

Figure 5. An example of STT and MTT

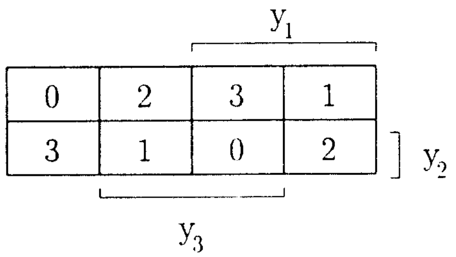


Figure 6. A universal One-shot state assignment for four-state flow table

the flow table shown in Fig.4. there exist following transitions: $\delta(1, X_2)=2$, $\delta(2, X_1)=3$, $\delta(3, X_3)=1$. Clearly, it is impossible to assign those stateswith adjacency ensured, by one binary vector per state. Hence state assignments in which adjacency betweentransitions is ensured by introducing *transient states* are called multicode transitiontime assignments. Fig.5 shows an example of STT and MTT. The binary vector (100) correspond to a transient state

Ⅲ The One-Shot State Assignment Using Graph-Theoretic

1.One-Shot State Assignment

The *One-shot* state assignment is a restricted class of STT state assignment, having the added property that only a single state variable changes for each inter-state transition. The solution having presented is in the form of a class ofuniversal assignments. The One-shot feature is significant if some price must be paid whena state variable changes state, perhaps inenergy, heat dissipation, component deterioration, or risk of malfunction. Such an assignment mightalso have value as a tool in coping with certaintheoretical questions.

A universal One-shot assignment for an n-state flow table must imply at least n-1 state variables,since from a binary vector representing state i, it must be possible to reach any of n - 1 other states by changing a single variable. These pointsare nicely illustrated by Fig.6. which depictsa three-variable universal One-shot assignment validfor all four-state flow tables(flow table states arenumbered from 0 to n-1 in this section). A set R_i which is consist of binary vectorsassigned to state i has two members,located as far apart as possible, and each binary vector in R_i is adjacent to a member of R_j for every $i \neq j$. The number of state variables is n-1 and in this case, it is also the minimum number required to assign all four-state flow tables even for assignments notof the STT variety. Hence this assignment isinteresting and useful apart from its illustrative value. The generalization of this assignment to larger tables, with $n=2^k$ is the *Hamming universal One-shot assignment* described below.

For the benefit of those familiar with theHamming single-error correcting codes,The Hamming universal One-shot assignmentmay be stated that the binary vectorsassigned to R_0 are those corresponding to Hamming code words with no errors and that, for $i > 0$, R_i consists of those words with errors in position i.

Theorem 3.1 *The Hamming assignment described above isuniversal One-shot assignment.*

(Proof)(By[Yaj87])

[1] Any code word ($x \in R_0$) is adjacent to $x \oplus e_i \in R_i (i \neq 0)$, where e_i is a binary vector which contains nonzerodigit in only position i and zero digit in the other positions. It is obvious by the definition.

[2] Any code word(binary vector) $y \in R_i$ is adjacent to $y \oplus e_k \in R_j$ for every $i \neq j$, where $[k] = [i] \oplus [j]$, $[k]$ is the binary value of k and $\oplus$ denotes the addition modulo-2.

It is obvious that both are adjacent.Hence if $y \oplus e_k \in R_j$ is proved, it is sufficient. For that, show the em *syndrome of* $y \oplus e_k$ is $[j]$.

$$[H(y \oplus e_k)^T = Hy^T \oplus He_k^T = [i] \oplus [k] = [j]] \ ,$$

where H is the parity check matrix of the Hamming error correcting code. Thus,

$$R_i \oplus e_k = R_j$$

□

2. Unicode One-Shot State Assignment

In this section, we discuss a unicode One-shotstate assignment referred to as a One-shotstate assignment with only one binary vector per state. *A necessary and sufficient-condition* for the unicode One-shot state assignmentis stated, and an assignment algorithm developedin graph-theoretic terms is proposed. The result of this algorithm minimizes the number of state variables. Besides, an upper bound on the number of gate inputs required for unicode One-shot state assignment realizations is shown in this chapter.

2.1 A Necessary and Sufficient Condition

In this subsection, a necessary and sufficient condition for the unicode One-shot state assignment is discussed.

Consider n state variables, $y_1, y_2, \cdots, y_i, \cdots, y_n$,where each y_i can take a value of either 0 or 1. Then, the set B of binary vectors of n bits($y_1, y_2, \cdots, y_n$) has 2n distinct elements. Two binary vectors of B are said to be adjacent,if the two binary vectors differ in exactly one bit. On the other hand, an n-cube is considered as a labeled graphsuch that each vertex of the n-cube is labeled as a binary vector of B. Namely, the coordinate of each vertex in an n-cube can be regarded as a label. If two vertices u and v of the n-cube are adjacent, i.e., there is an edge(u, v) in the n-cube, their labels are adjacent. Hereafter, a graph is considered as an undirected andconnected graph in this paperuntil otherwise is stated or implied.

Definition 3.1 *Edge index* ρ

Let us consider a labeled graph of which each vertex is labeled as a distinct binary vector of n bits. For two vertices x and x′ of the graph, an edge(x, x′) of the graph is said to have an edge index ρ *if and only if the binary vector assigned to x is adjacent tothe binary vector assigned to x′ and the two binary vectorsdiffer in the* ρ*th digit. Other edges have no indices.* □

For example, it is easy to decide edge indices for all theedges of an n-cube.

Definition 3.2 *Set of edge indices*

Let us consider a subgraph of an n-cubein which each edge is labeled as an edge index.For two vertices x and x′ of the subgraph,we define $E_c(x, x')$ *as a setof edge indices such that each edge index appears an oddnumber of times on a path(x, x′), where a* *path (x,x′) is an alternating sequence of vertices andedges, starting at x and ending at x′, in which all vertices are distinct.*

Notice that $E_c(x,x')$ is independent of the consideredpath. $|E_c(x,x')|$ is the distance (x,x') on the cube, where $|T|$ denotes the size ofa set T.

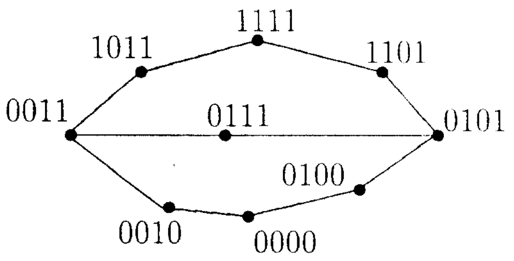
For simplicity, a set of edge indices $\{\rho_1,\rho_2\},\cdots,\rho_k\}$ is often denoted by $\rho_1\rho_2\cdots\rho_k$, if there is no danger of confusion.

Definition 3.3 *Relative coordinates of the vertices*

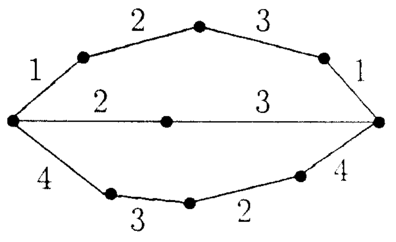
Let us consider a subgraph of an n-cube. We consider some vertex r of the subgraph as a referencevertex. We then associate to each vertex x of the graph (x≠r) its relative coordinatewith regard to r: $E_c(r,x)$. However the relative coordinate of r is 0. □

Definition 3.4 *Graph associated with a flow table*

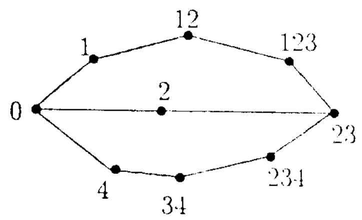
An undirected graph associated with a flow table G is made as follows: Each vertex in G corresponds to a stateof the flow table, and



: a subgraph S of 4-cube

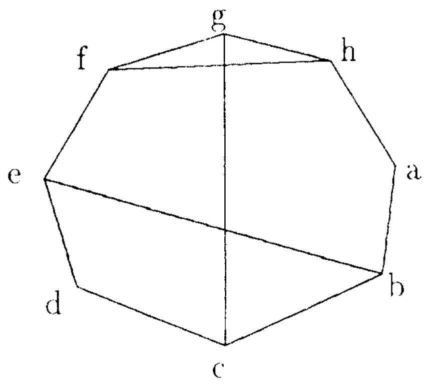


: edge indices in S

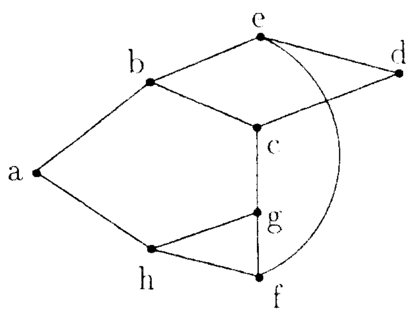


· relatibe cordinates of S

Figure 7. Edge indices and relative coordinates



a) a connected graph



b) a layer representation for a)

Figure 8. An example of the layer representation

each edge of G representstransition between the two states corresponded to the two end-vertices of the edge. However self-loops are omitted. □

Definition 3.5 *Layer representation associated with a graph G*

Let r be a reference vertex of G. We generate a partition of G into an ordered set ofblocks or layers $C_1, C_2, \cdots, C_i, \cdots$}, so that for a vertex x of G, $x \in C_i \Leftrightarrow d(r,x) = i$, where $d(r,x)$ is the distance between r and x. This partition is called a layer representation associated with G. □

Definition 3.6 *SA_n illustrated in Fig.9 is said to be the n-cube checking matrix.* □

Definition 3.7 *Extended adjacency matrix*

Let $A = (a_{ij})$ be a $|S| \times |T|$ adjacency matrix of a bipartite graph G[S,T;E] defined by

$$a_{ij} = \begin{cases} 1 & \text{if } s_i \in S \text{ is adjacent to } t_j \in T \\ 0 & \text{otherwise} \end{cases}$$

, where |x| is the size of a set x.

Suppose that EA is a square matrix made of A such thatthe number of rows and columns of EA is to be 2^n, where $2^{n-1} < k \le 2^n$, k is the larger one betweenthe number of rows and columns of A, and n any integer. For that, 0-row or column vectorsmay be inserted into in EA. Then EA is said to be anextended adjacency matrix of $A = (a_{ij})$. □

Fig.10 shows examples of the One-shot state assignmentin which graphs associated with flow tables are illustrated,and the solutions to the graphs are minimum, taking into account the number of state variables and binary vectors per vertex.Having less vertices than the graph of Fig.10(b), the graph of Fig.10(a) is required more state variablesand binary vectors per vertex for One-shot state assignment than that of Fig.10(b). What makes it possible to perform the unicode One-

$$SA_m = \left(\begin{array}{c|c} SA_{m-1} & \begin{matrix} d & & 0 \\ & \ddots & \\ 0 & & d \end{matrix} \\ \hline \begin{matrix} d & & 0 \\ & \ddots & \\ 0 & & d \end{matrix} & SA_{m-1} \end{array}\right) \begin{matrix} 2^{m-2} \\ \\ 2^{m-2} \end{matrix} \qquad d = [0 \mid 1]$$
$$\qquad\qquad 2^{m-2} \qquad\quad 2^{m-2}$$

$$SA_{m-1} = \left(\begin{array}{c|c} SA_{m-2} & \begin{matrix} d & & 0 \\ & \ddots & \\ 0 & & d \end{matrix} \\ \hline \begin{matrix} d & & 0 \\ & \ddots & \\ 0 & & d \end{matrix} & SA_{m-2} \end{array}\right) \begin{matrix} 2^{m-3} \\ \\ 2^{m-3} \end{matrix} \qquad \cdots\cdots$$
$$\qquad\qquad 2^{m-3} \qquad\quad 2^{m-3}$$

$$SA_3 = \left(\begin{array}{cc|cc} d & d & d & 0 \\ d & d & 0 & d \\ \hline d & 0 & d & d \\ 0 & d & d & d \end{array}\right) \qquad SA_2 = \begin{pmatrix} d & d \\ d & d \end{pmatrix}$$

Figure 9. Forms of SA_n

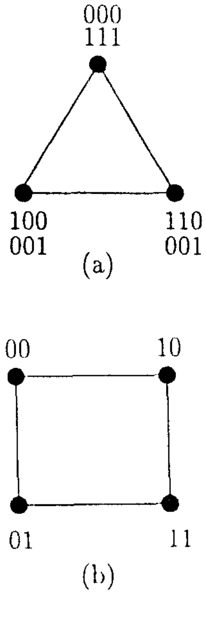


Figure 10. Examples of the One-shot assignment

shot state assignment,as shown in Fig.10(b)?

In a viewpoint of graph theory,what it is possible to perform the unicode One-shot state assignment is the same as it is possible to embed a graph associated witha flaw table in an n-cube. *i.e.*,the graph is a subgraph of an n-cube. Then, several properties concerned withsubgraphs of n-cube are discussed as follows.

Lemma 3.1 *A graph is bipartite if and only if all its cycles are even.*

(Proof) (By [Har69])

If G is a graph, then its vertex set V can be partitioned into two sets V_1 and V_2 so that every edge of G joints a vertex of V_1 with a vertex of V_2. Thus, every cycle $V_1\ V_2 \cdots V_n\ V_1$ in G necessarily has its oddly subscripted vertices in V_1, say, and the others in V_2, so that its length n is even.

For the converse, we assume, without loss of generality,that G is connected(for otherwise we can consider the components of G separately). Take any vertex $v_1 \in \$$, and let V_1 consist of v_1 and all vertices at evendistance from v_1, while $V_2 = V - V_1$. Since all the cycles of G are even, every edge of G joints a vertex of V_1 with a vertex of V_2. For suppose there is an edge joining two vertices of V_1. Then the union of geodesics from v_1 to v and from v_1 to u together with the edge uv contains an odd cycle, a contradiction. □

Lemma 3.2 A graph G is bipartite if and only if, a reference vertexbeing arbitrarily chosen, the layer representation associated with G has no edge whose ends are in the same layer.

(Proof) By lemma 3.1 and the definition of the layer representations associated with graphs, it is obvious. □

Lamma 3.3 *If vertices of a graph, $x, x' \in C_i$ and x is adjacent to x' in the layer representation associated with the graph,then there exists at least one cycle of oddlength including edge(x, x').*

(Proof) Note bipartite graphs have no cycle of odd length. If there exist edges whose ends are in the samelayer of the layer representation, the graph described by thelayer representation is not abipartite graph. □

Definition 3.8 Product of a graph G_1 and G_2: $G_1 \times G_2$

Let G_1 and G_2 be graphs.Consider any two vertices $u = (u_1, u_2)$ and $v = (v_1, v_2)$ in $V = V_1 \times V_2 = \{(a,b) \mid a \in V_1$ and $b \in V_2\}$, where V_i is a set of vertices of G_i(i = 1,2). Then u and v are adjacent in $G_1 \times G_2$ whenever $u_1 = v_1$ and u_2 is adjacent to v_2, or $u_2 = v_2$ and u_1 is adjacent to v_1, where two vertices u and v are said to be adjacent ifthere is an edge(u, v).

Lemma 3.4 *There exists no cycle of odd length in an n-cube.*

(Proof) Define n-cube Q_n recursively as $Q_1 = K_2$ and $Q_n = K_2 \times Q_{n-1}$, where K_2 denotes the complete graph with 2 vertices.Clearly, Q_2 is consisted of one cycle of even length.Assume Q_{n-1} is satisfied with this lemma. Then it is easy to show by the definition of the operation $\times$ that $Q_n = K_2 \times Q_{n-1}$ has no cycle of odd length. Thus, there exists no cycle of odd length in an n-cube.

Lemma 3.5 *If a graph consists of one spanning cycle, and thelength of the cycle is even,then it is possible to perform the unicode One-shot state assignment for the graph.*

(Proof) By the above lemmas, it is obvious. □

Lemma 3.6 *If a graph consists of one span-*

ning cycle, and thelength of the cycle is odd,then it is possible to perform the One-shot state assignment for the graph, using two binaryvectors per vertex.

(Proof) Let the length of a spanning cycle be n. since $2n$ is even, clearly, there exist cycles of $2n$ length in an n-cube. Select a cycle amongcycles of $2n$ length, and match the selected cycle to the graph on keeping the adjacency relation between verticessuch that beginning a vertex of the selected cycle andthe graph, vertices of the graph correspond to verticesof the selected cycle two times. Then, the One-shot state assignment is obtained.

Theorem 3.2 *The unicode One-shot state assignmentfor a graph with k vertices is possibleif and only if the graph is a bipartite graph G[S,T:E] and an extended adjacency matrix SA made from G can be transformed into SA_n shown in Fig.9 by permutations ofthe rows and columns of SA.*

(Proof) *Let SA_n (see to Fig.11 be the worst case of SA_n with all the d = 1. Then, we prove that SA_n' is a necessary and sufficient condition for n-cube.*

First, we show that SA_n' is made from n-cube.Consider an edge of an n-cube, and remove all the edges paralleled with the edge from n-cube. Then we obtain two (n-1)-cubes G_1, G_2 such that(G_1 = $[s_{kp}, t_{lq}:E']$, G_2 = $[s_{lq}, t_{kp}]:E'']$), where ($s_{kp}, s_{lq} \in S$, $t_{lq}, t_{kp} \in T$, p, q = $1, 2, \ldots, 2^{n-2}$) and s_{kp}, $t_{lq} \in G_1$ is connected with $t_{kp}, s_{lq} \in G_2$ by all the edges removed, respectively. Fig.12 shows G_1 and G_2, and the matrix form of G_1 and G_2 is illustrated in Fig.13. The same operation described above is applied to G_1 and G_2 recursively such that if necessary, the 1's of the matrices of ((s_{kp}, t_{kp}) and (s_{lq}, t_{lq})) is adequately moved to the main diagonals, respectively. Thus, we can obtain the matrix SA_n'.

For the converse, in the case that SA_n' is given, first 2-cube is made from SA_2', and 3-cube is made by connecting two 2-cubes with 2^2 paralleledоткedges. Then by applying this operation successively,we can obtain an n-cube. □

Corollary 3.1 *If a bipartite graph [S,T:E] with n vertices are satisfied with thenecessary and sufficient condition of theorem 3.2, the number of state variables used in the unicode One-shot state assignment for the graph is $\lceil \log_2 n \rceil$, except $|S|$ or $|T|$ = 1.*

2.2 An algorithm developed in graph theo-

$$SA_m' = \left(\begin{array}{c|c} SA_{m-1}' & \begin{matrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{matrix} \\ \hline \begin{matrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{matrix} & SA_{m-1}' \end{array}\right) \begin{array}{l} 2^{m-2} \\ \\ 2^{m-2} \end{array}$$
$$\qquad\quad 2^{m-2} \qquad\qquad 2^{m-2}$$

Figure 11. Form of SA_n'

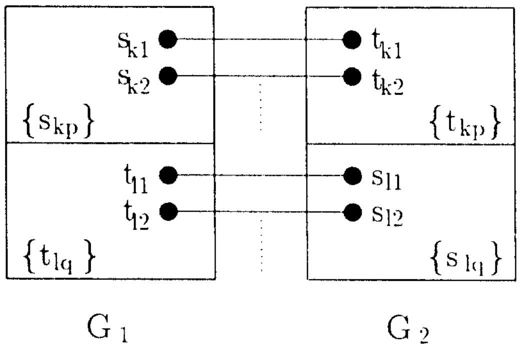


Figure 12. G_1 and G_2

retic terms

In this section, we show an algorithmfor the unicode One-shot assignment, whichis stated in graph-theoretic terms.

An algorithm discussed below permitsus to find an exhaustive search procedurefor the One-shot state assignment. For a givenlayer of the layer representation for a graph, we form the product of the possibilities for eachvertex of the layer, taking into account theadjacency condition. We also have to avoid thegeneration of equivalent binary vectors (*i.e.*, equivalent by permutation of the directionindices). The algorithm follows in details :

step 1 : Transform a given flow table to a graph G associated with it.

step 2 : Let S be a vertex of G havingmaximum degree. Generate the layer representationunder S as a reference vertex.

step 3 : Assign a relative coordinate 0 to S, and look for the relative coordinates of C_{j+1} layer from C_j as follows:

- If $x' \in C_j$, $x \in C_{j+1}$ and x' is adjacent to x, then n possibilities for therelative coordinate of x taking into accountits adjacency relation with x' will be

$p(x \mid x') = \{E_c(S, x') \cup \{k\}\} - \{E_c(S, x') \cap \{k\}\}$

where $A-B$ denotes the set yielded on removing the element of set B from set A, and k describes, $(1, 2, \cdots, \lceil \log_2 n \rceil)$. Hence we choose one among $p(x \mid x')$ for the relative coordinate of x such that the chosen relative coordinateis not used for other vertices.

More generally, if x is adjacent tothe vertices $x_1, x_2, \cdots, x_m$ of C_j, then the possibilities for therelative coordinate of x taking into accountits adjacency relation with $x_1, x_2, \cdots, x_m$ are

$p(x \mid x_1, x_2, \cdots x_m) = p(x \mid x1) \cap p(x \mid x_1) \cap \cdots \cap p(x \mid x_m)$.

step 4 : If all possible relative coordinatesare used until C_j, then *back tracking* method is operated.

An example for the above algorithm is illustratedin Fig.14. Transforming G_1 shown in Fig.14-(a) to a layer representation,

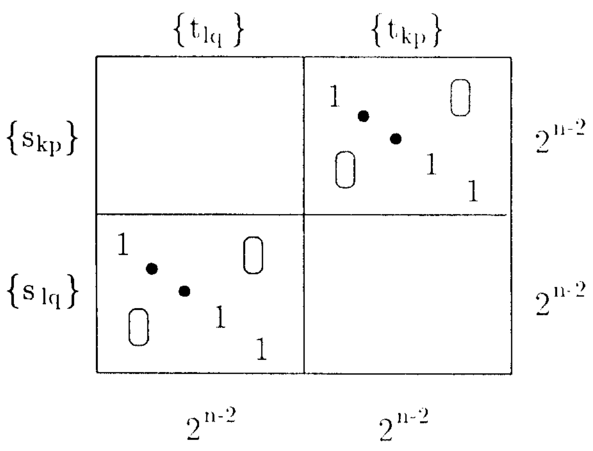
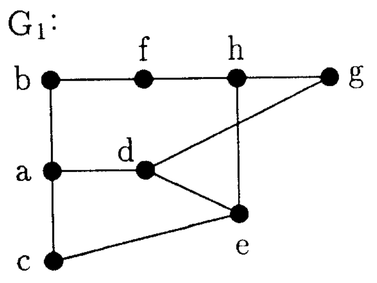
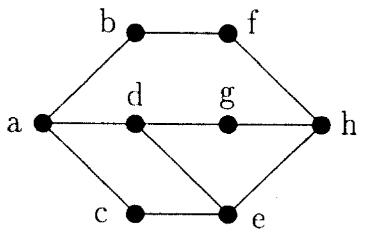


Figure 13. Matrix form of G_1 and G_2



(a)



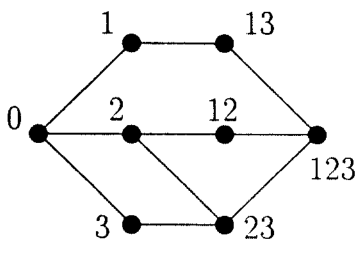
(b)

$$\begin{array}{c|cccc} & b & c & d & h \\ \hline a & 1 & 1 & 1 & 0 \\ e & 0 & 1 & 1 & 1 \\ g & 0 & 0 & 1 & 1 \\ f & 1 & 0 & 0 & 1 \end{array}$$

(c)

$$\begin{array}{c|cccc} & b & c & d & h \\ \hline a & 1 & 1 & 1 & 0 \\ f & 1 & 0 & 0 & 1 \\ g & 0 & 0 & 1 & 1 \\ e & 0 & 1 & 1 & 1 \end{array}$$

(d)



(e)

Figure 14. An example of the exhaustive algorithm

Fig.14-(b) is obtained. So, G_1 is bipartite since no edge whose ends in the same layer exists. Then an extended adjacency matrix EA is made as like Fig.14-(c). By permuting the rows and columns of EA, SA_3(Fig.14-(d)) is obtained. We see then that the unicode One-shot state assignment for G_1 is performed with 3 state variables, as like Fig.14(e).

Ⅳ. Upper Bound on the Number of Gate Input Required for One-Shot State Assignment Realizations

It is useful to have an easily computed

upper bound on the logic necessary to realize a given flow table. Hence in this section, the number of gate inputsrequired for unicode One-shot state assignment realizations is discussed.

For the discussion, we will give the four followingassumptions:

1. The flow table is normal mode.

A normal mode flow table is one in which any transition leads directly to a stable state and no output is required to change more than once during atransition.

2. The next-state logic isimplemented with a two-stage *AND-OR* circuit.

3. The *NOT* inputs are permitted.

4. On the gates, there is no *fan-in* constraint.

Definition 4.1 Gate inputs

In a two-stage AND-OR *circuit, inputs of* AND *and* OR *gate are called* gate inputs. □

First at all, the number of gate inputs required for the next-state function is consideredfor the *one-hot* state assignment andTan's USTT [Tan71], together with that of state variables required forthe two state assignments.

In the *one-hot* state assignment, if in a column I offlow tables, there is a transition from state p to state q, then we specify the logic so thatfirst y_q(y_q is the qth state variable)goes *on* (y_p is of course initially *on*) and then y_p goes *off*. The general form of the next-state function for y_i is $Y_i = F_{i1} + y_i\overline{F}_{i2}$. The F_{i1}-function is a summation of terms representing unstabletotal states (*i.e.*, a sum of unstable states in each input column of flow tables) which lead to state i. Hence F_{q1} includes a term Iy_p. The F_{i2}-function is a sum of terms corresponding to next states reached from state i. The $y_i\overline{F}_{i2}$-termserves to held y_i on until the y-variablecorresponding to the next state goes on. For the $p \rightarrow q$ transition, F_{p2} has a term y_q. Each unstable state produces one component ofan F_{i1}-expression and adds one literal to an F_{i2}-product. Thus, if m is the number of input variables, r the number of states, and u the number ofunstable total states, then an upper bound on the numberof gate inputs is

$$2r+u(m+3)$$

,an expression arrived at by computing the termsand literals added to the F_{i1}-and F_{i2}-expressions be cause of each unstable entry. The number of state variables required for the one-hot state assignment is r. Of course, those components associated withthe state variables increase in number linearlywith r. Particularly when the number of unstablestates is relatively small, the amount of logicneeded in one-hot circuits is usually not excessive compared with what is needed for stateassignment requiring fewer state variables

On the other hand,in Tan's USTT assignment, the following is a summary of theassignment scheme.

1. Order the distinct destination sets $D_{i,q}$ of a given flow table,where the set of states $D_{i,q} = \{p \mid \delta(p, I_i) = q\}$ is called a destinationset under input I_i.

2. Associate a unique variable y'_j for each element $D_i[j]$ in the set of ordereddestination sets such that $y'_j = 1$ for $s \in D_i[j]$ and $y'_j = 0$ for $s \notin D_i[j]$.

3. For each variable y'_j, express the next-state equation as a sum of simple product-terms(*i.e.*, a product terms such that it containsat most one state variable) such that a simple productterm $x_e y'^{e}_t$ is contained in the equation ifand only if states in $D_e[t]$ are involved in 1-transitions with respect to y'_j.

The number of gate inputs required for Tan's USTT algorithm is

$\Sigma D_i(m_I+m+1)$ $(1 \leq i \leq m_I)$

,where D_i is the number of stable states in the ith input column(shortly, ΣD_i is called stable total states), m_I the number of input columns in a flow table,and m the number of input variables.The bound derived above increases as a functionof the number of stable total states of the flow table. It is interesting to compare this bound with that obtained by the one-hot assignment. Notice thatthis bound tends to increase as a function of thenumber of unstable states of the flow table.Therefore, for flow tables having less stable statesthan unstable states, the bound derived in Tan's USTT will, in general, be better than that derived from the one-hot assignment. The upper boundon the number of state variables required for Tan's USTT algorithmis $d = \Sigma D_i (1 \leq i \leq m_I)$. Note that those components associated with the state variablesincrease in number linearly d, but not the number of rows(states)of flow states.

In the unicode One-shot assignment, we obtain the following:

Theorem 4.1 *For a given flow table, if the following hold, then the number of gate inputsrequired for unicode One-shot assignmentrealizations is less than or equal to*

$u(k_s+M-1)+\Sigma u_i$ $(1 \leq i \leq k_s)$.

u : *the number of unstable total states*

k_s : *the number of state variables*

M : *the number of input variables*

u_i : *the number of ones that are contained in binary vectors assigned to unstable states in position* i.

(Proof) Each unstable state produces one product term.Each product term contains M input variables and k_{s-1} state variables. The equation Y_i is consisted of u_i product terms. Therefore, an upper bound on the number of gate inputs is $u(k_s+M-1)+\Sigma u_i$ $(1 \leq i \leq k_s)$. □

Since $\Sigma u_i (1 \leq i \leq k_s)$ can be replaced by $u * k_s$, we habe the follwing result.

Corollary 4.1 Given a flow table, the number of gate inputs requiredfor unicode One-shot assignment realizations is less than or equal to

$$u(2 * k_s+M-1).$$

□

Notice that the one-hot state assignment requires2-step transition while the One-shot state assignmentwill result in circuits operating in single transition time. Therefore, the One-shotstate assignment will be able to bring forthgood results for *control circuits* thanthe one-hot state assignment even though there exists no difference between the number of gate inputs requiredfor the two state assignments. Remember that why the asynchronous modeis adopted in sequential circuits like control circuits. Because of the speed of operation. In Tan's USTT,the number of state variables is dependent of the number of stable total states of the flow table.Hence the number of *flip-flops* in unicode One-shot assignment realizations is less thanthat in Tan's USTT, since the former is $\log_2 n$, where n is the number of states.

For example, the flow table for a simple counter is illustrated in Fig.15. The one-hot state assignment will require $2r+u(m+3) = 2 * 6+6(1+3) = 36$ gate inputs and 6 state variables. In Tan's USTT, the number of gate inputs required is $d(m_I+m+1) = 6(2+1+1) = 24$, and the number of state variables is $\Sigma D_i = 3+3 = 6$. However for the One-shot state assignment developed in this chapter, we have $u = 6$, $M = 1$, and $k_s = 3$. So the upper

Q \ X	0	1
1	①	2
2	3	②
3	③	4
4	5	④
5	⑤	6
6	1	⑥

Figure 15. A simple counter

$y_1y_2y_3$	$Y_1Y_2Y_3$	
	$x=0$	$x=1$
$1 \rightarrow 000$	000	001
$2 \rightarrow 001$	011	001
$3 \rightarrow 011$	011	010
$4 \rightarrow 010$	110	010
$5 \rightarrow 110$	110	100
$6 \rightarrow 100$	000	100

Figure 16. A One-shot state assignment for Fig.15

bound is equal to $u(k_s+M-1)+\Sigma u_i = 6(3+1-1)+(2+3+2)=25$. where Σu_i is obtained from aunicode One-shot state assignment for the flow table.shown in Fig.\ref{fig:unils}.This result is better than the others.taking the number of gate inputs and state variables into consideration.

Besides, the number of gate inputs required for *output-equation* in unicode One-shot assignmentrealization is $Z(k_s+M)$. where Z is the number of total ones appeared in output entries of flow table. In multicode One-shot assignmentper state, the number of gate inputs required for the next-state equations is $L * u(k_s+M+1)+\Sigma u_i$. and the number of gate inputs required forthe output equations is $L * Z(k_s+M)$

Ⅴ. Remarks and Discussions

The One-shot state assignment problem for asynchronoussequential machines is considered. Up to now, the One-shot state assignment has only known bythe Hamming universal One-shot state.

The main contributions of this paper are: 1) A necessaryand sufficient condition for the unicode One-shot state assignment is proposed, where the condition is provided for the first time. 2) An algorithm for it is developed in graph-theoretic terms.The algorithm requires $\lceil \log_2 n \rceil$ state variables for a flow table with n states for which it is possible to performthe unicode One-shot state assignment. 3) we present an easily calculated upper bound on thenumber of the gate inputs required for unicode One-shotstate assignment realizations by two-stage AND-OR circuits. It is shown thatunicode One-shot state assignment realizationsmay be smaller than the one-hot state assignment realizations.taking both state variables and gate inputsinto consideration.

For future work, the complexity of the One-shot state assignment is not yet known. It is supposed to belong tomore difficult class than the problem of edge-colorings of graphs in time complexity. Efforts to clarify the time complexity of the One-shot state assignment are necessary.

From the practical view point, the implementation of sequentialmachines in VLSI circuits design has to satisfy two major requirements:

i) regular and structured design that can be supported by computer-aided tools;

ii) size and performance of the silicon implementation.

These requirements imply to us to develop a stateassignment taking the circuit realization intoconsideration. It is supposed to be very difficult, however.

Acknowledgment

We would like to thank all the members be Professor Yajima's Laboratory at Kyoto University for their useful discusstions and great supports throughout this research.

References

[AFM68] D. B. Armstrong, A. A. Friedman, and P. R. Menon. Realization of asynchronous sequential circuits without inserted delay eleements. *IEEE Trans. Comput.*, vol. C-17 : pp.129~139, Feb. 1968.

[Bry86] R. E. Bryant. Graph-based algorithms for boolean function manipulation., *IEEE Trans. Comput.*, Vol.C-35, 8 : pp.677~691, 1986.

[ea83] Y. Tojma et al. *Sequential Machines. Iwanami* Syoten, 1983.

[FGU69] A. D. Friedman, R. L. Graham, and J. D. Ullman. Universal single transition time asynchronous state assignments. *IEEE Trans. Comput.*, vol. C-18 : pp.541~547, June. 1969.

[G70] Mago G. Asynchnorous sequential circuits with (2,1) type state assignments. in IEEE Conf. Rec., 1970 11th Annu. Symp. Switching and Automata Theory, pages pp.109~113, 1970.

[Har61] J. Hartmanis. On the state assignment problem for sequential machines 1. *IRE Trans. Electron. Comput.*, vol. EC-10 : pp.157~165. Jun. 1961.

[Har69] F. Harary. *GRAPH THEORY.* Addision Wesley, 1969.

[Huf55] D. A. Huffman. A study of the memory requirements of sequential switching circuits. *Res. Lab. Electron., M.I.T.*, Tech. Rep. 923, Mar. 14, 1955.

[Kar64] R. Karp. Some techniques for state assignment for synchronous sequential machines. *IEEE Trans. Electron. Comput.*, vol. EC-13 : pp.1160~1179, Dec. 1964.

[Koh70] Z. Kohavi. *Switching and Finite Automata Theory*. McGraw-Hill, New York, 1970.

[KR78] J. G. Kuhl and S. M. Reddy. A multicode single transition-time state assignment for asynchronous sequential machines. *IEEE Trans. Comput.*, vol. C-27(No.10) : pp.927~934, Oct. 1978.

[Liu63] C. N. Liu. A state variable assignment method for asynchronous sequential switching circuits. *J. Ass. Comput. Mach.*, vol. 10 : pp.209~216, Apr. 1963.

[Mag69] G. Mago. Universal state assignments for asynchronous sequential circuits. *IEEE Computer Group Repository*, R-69-6, 1969.

[MIY90] S. Minato, N. Ishiura, and S. Yajima. Shared binary decision diagram with attributed edges for effcient boolean function manipulation. *Proc. 27th Design Automat. Conf.*, pages pp.52~57, 1990.

[PR76] D. K. Pradhan and S. M. Reddy. Techniques to construct (2,1) separating systems from linear error-correcting codes. *IEEE Trans. Comput.*, Vol. C-25, Sept.

1976.

[Sas89] T. Sasao. On the optimal design of multiple-valued pla's. *IEEE, Trans. Comput.*, vol. 38(No.4), April. 1989.

[Tan71] C. J. Tan. State assignments for asynchronous sequential machines. *IEEE Trans. Comput.*, vol. C-20 : pp.382~391, Apr. 1971.

[Tra66] J. H. Tracey. Internal state assignments for asynchronous sequential machines. *IEEE Trans. Electron.* Comput., vol. EC-15 : pp.551~560, Aug. 1966.

[Ung66] S. H. Unger. *Asynchronous Sequential Switching Circuits.* Wiley-Interscience, New York, 1966.

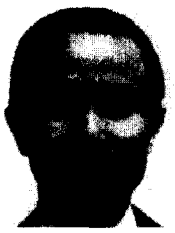
[Ung68] S. H. Unger. A row assignment for delay-free realizations of folw tables without essential hazards. *IEEE Trans. Comput.*, vol. C-17 : pp.146~152, Feb. 1968.

[Ung69] S. H. Unger. *Asychronous Sequential Switching Circuits.* John Wiley & Sons, Inc., 1969.

[Yaj87] S. Yajima. Asynchronous sequential machines. *Lecture Notes in Switching and Automata Theory Ⅱ*, 1987. in Japanse.

權 容 珍(Kwon Yong Jin) 정회원

Kwon, Yong-Jin was born in Seoul, Korean, in 1964. He received the B.E. degree in eletronic engineering from Hankuk Avation University, Seoul, Korea, and M.E. and Ph. D. degrees in information science from Kyoto University, Kyoto, Japan, in 1986, 1990 and 1994, respectively. Since 1994 he has been a Professor in the Telecommunication and Information Engineering Department, Hankuk A viation University, Seoul, Korea. His current interests include logic synthesis of sequential circuits, algorithm design and software engineering.



矢島脩三(Yajima Shuzo) 정회원

Shuzo Yajima was born in Takarazuka, Japan, on December 6, 1933. He received the B.E., M.E., and Ph. D. Degrees in electrical engineering from Kyoto University, Kyoto, Japan, in 1956, 1958, and 1964, respectively. He developed Kyoto University's first digital computer, KDC-I, in 1960. In 1961 he joined the faculty of Kyoto University. Since 1971 he has been a Professor in the Department of Information Science, Falculty of Engineering, Kyoto University, engaged in research and education in logic circuits, switching, and automata theory. Dr.Yajima was a Trustees of Japan and Chairman of the Technical Committee on Automata and Languages of the Institute. He served on the Board of Directors of the Information processing Society of Japan.