컴퓨터를 이용한 순차 논리 회로의 설계 (비동기 순차 논리 회로의 경우)

論 文
33~2~1

Computer-Aided Design of Sequential Logic Circuits (Case of Asynchronous Sequential Logic Circuits)

金 炳 喆*·趙 東 燮**·黃 熙 隆***
(Byung-Chul Kim · Dong-Sub Cho · Hee-Yeung Hwang)

요 약

본 논문은 비동기 순차 회로를 설계할 때 필요한, 레이스 (race) 없는 내부 상태의 지정과 조합 회로망의 최소화 단계에 대해, 컴퓨터를 이용한 방법을 제안하고자 한다.

제안된 알고리즘 (algorithm)은 포트란 (FORTRAN)으로 작성되었으며, 그 결과로 레이스없는 상태를 지정할 수 있고, 또한 소자 (gate)의 수와 연결선 수를 줄여서 최소에 가까운 비용으로 원하는 회로를 설계할 수 있음을 알 수 있다.

제안된 알고리즘에 따른 컴퓨터 프로그램에 의해, 몇가지 예를 제시했으며, 끝으로 이 결과를 다른 방법들에 의한 것과 비교해 보았다.

Abstract

This paper is concerned with a computer-aided state assignment, that is, coding race-free internal states of asynchronous sequential circuits, and a method for minimizing the combinational network of asynchronous sequential circuits.

The FORTRAN version of the proposed algorithm results in race-free state assignments and reduction of the number of connections and gates with near minimal hardware cost.

Some examples are designed by the proposed computer program to illustrate the algorithm in this paper. Finally, results are compared with those of the other methods.

I. Introduction

Asynchronous sequential circuits are widely used in most of present day digital circuits, together with combinational network to realize complex sequential functions.

The general form of the asynchronous sequential circuit is combinational network with feedback. And all inputs and outputs are level signals which may remain at either value (0 or 1) for arbitrary periods and may change at arbitrary times. However, due to the absence of clock pulses, there may races and hazards among the circuit variables. Therefore, these circuits are usually more difficult to design than synchronous sequential circuits.

One of the important steps in design process is obtaining a valid state assignment for a given sequential functions. The state assignment for an asynchronous sequential circuit must ensure that no critical race exists among the state variables [1), 2)]

There are many kinds of effective algorithms for obtaining a valid state assignment [3), 4), 5)].

By these algorithms, we can assign valid states to an asynchronous sequential circuit which operates

*正 會 員：서울大 工大 電子計算機工學科 碩士課程
**正 會 員：서울大 工大 電子計算機工學科 博士課程
***正 會 員：서울大 工大 電子計算機工學科 教授
接受日字：1983年 8 月16日

correctly under some constrainst, even though they require a large amount of hand tailoring or are very difficult to program it on a digital computer.

And the other important step is minimizing the output functions. After assigning a valid state, combinational network which generate appropriate outputs and next states are minimized in the conventional method using map or table[6),7)]

But these output functions resulting from the minimization method above are not optimal because of the common terms of output functions. And accordings to the increase in the number of inputs and outputs, it is very difficult, if not impossible, to minimize the output functions by hand working.

In this papper, a computer program, MASK-LASEQ. is suggested, which assigns valid states and minimizes the output functions. A reduced flow table[1)] which is a method for describing the characteristics of a sequential circuit, is used for input.

The proposed state assignment procedure is straightforward. easy to apply, and ensures a race-free state assignment for any asynchronous sequential circuit. And the implemention of this procedure on a computer is accomplished with ease. Especially, the generation of the intermediate transient states is simple, thereby we can reduce the efforts in removing the race problem. But this procedure requires more state variables than the minimum. And the operating speed of the resulting circuit is adversely affected because some transitions between states are accomplished through intermediate transient states. This procedure is described in more detail in section 3.

And the outputs of this procedure are used for inputs of multiple output function minimization method, MASK-MULTI[10),11)]. MASK-MULTI is an improved algorithm. in which the common terms of the output functions are considered. In section 2, the basic idea of constructing inputs for MASK-MULTI is suggested in order to use MASK-MULTI method in sequential circuit design. And then state assignment procedure mainly aimed at the elimination of critical races is discussed in section 3. Finally, in section 4, asynchronous sequential circuit is implemented by the proposed procedure, MASK-LASEQ. and corresponding computer program is also discussed.

2. Property of Level-Mode Sequential Circuits

The formal definition of a asynchronous sequential circuit (ASC) is given below.

Definition 1: (Asynchronous sequential circuit)

ASC is siven by the 5-tuple ASC=(X, Q, Z, G, F), where:

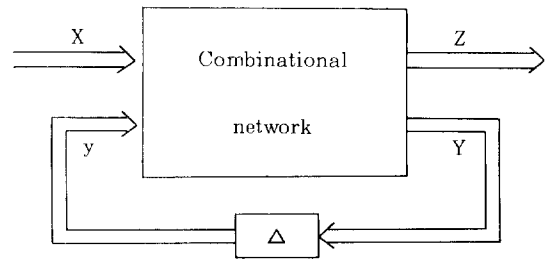
a) X=(x1, x2, ... , xn) is a set of input variables with the following properties;
   i) input variable x i ϵX is voltage levels,
   ii) input variable xiϵX is never changed unless the circuit is stable internally.

b) Q=(a1, a2, . . . , ar) is a set of internal states of ASC, where each ai can be expressed by yi

c) Z=(z1, z2,···zm) is a set of output variables.

d) G=(g1, g2, ... , gr) is a set of next-state functions with the following properties;
   i) the next-state variable Yi is given by
   Yi=gi (y1, y2, ... , yr, x1, x2, . . . , xn)
   ii) the next-state q(t+1) is given by q(t+1)= G(X, q(t))

e) F=(f1, f2, ... , fm) is a set of output functions such that the present value of the output variable zi (i=1, 2, . . . , m) is given by Zi=fi (X, qt).



X : Input variables
Z : Output variables
y : State variables
Y : Next-state variables

Fig. 1. Basic model of livel-mode sequential circuit.

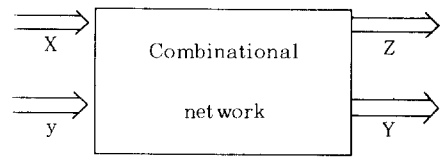


Fig. 2. Combinational network from sequential circuit.

Fig. 1. shows the general block diagram of ASC, which consists of two parts, a combinational network and a memory section. Delay elements, denoted by – [Δ] –, are used for memory section.

The combinational network can be characterized by G. and F. These functions are time-independent equations, i.e., they are valid at any time that all the inputs and outputs are stable. Therefore, the combinational network can be considered independently with the memory section (see Fig. 2). This combinational network can be treated as multiple output switching circuit. Then, we can now apply MASK-MULTI method to minimize this combinational network of sequential circuit, whose inputs and outputs are (y1, y2, ..., yr, x1, x2, ..., xn) and (z1, z2, ..., zm, Y1, Y2, ..., Yr), respectively.

3. Reduced One-Hot Assignment Procedure

In asynchronous sequential circuits, a transition from one stable state to another occurs only in response to changes in the inputs. When the input is changed, some difficulties may arise as a result of the different delays associated with the various sates, if two or more internal state variables are required to change their values simultaneously. Therefore, the assignment of binary values to the rows of the required flow table must ensure that the circuit will operate correctly even if different delays are associated with the internal states.

For the convenient description of the proposed procedure, some definitions are listed below.

Definition 2: (Stable state)

A state ai is said to be stable if the next state is also ai under some input X, i.e., G(X, ai)=ai.

Definition 3: (Direct transition)

A direct transition from state qi to qj under input X, denoted by qj=G(X, qi), is a transition whereby all internal state variables that undergo a change are excited simultaneously.

Definition 4: (Race)

When the binary code of the next state differs from the code of the present state in two or more bits, the circuit is said to be racing from the present state to the next state.

Table 1. Reduced flow table

q \ $x_1\ x_2$	input 00	01	11	10	
q_1	q_1	q_3	q_1	q_1	$q_1 \leftarrow 00$
q_2	q_1	q_3	q_2	q_2	$q_2 \leftarrow 01$
q_3	q_3	q_3	q_2	q_1	$q_3 \leftarrow 11$

Definition 5: (Critical race)

If a race condition exists and there is a possibility that unequal transmission delays may cause the circuit to reach stable state other than the one intended, the race is called critical; all others are non-critical.

Definition 6: (The condition of critical race)

A direct transition qj=G(X, qi) races critically with the direct transition ql=G(X, qk) if the possibility exists that unequal transmission delays may cause these two transitions share a common internal state.

A reduced flow table and its state assignment, for example, are shown in Table 1. Assume that input is 00 and internal state is q3, and then the input is changed to 10. If direct transition occurs, the next state is given by

G(10, 11) = G(10, 00) = q1.

Whereas if unequal transmission delay causes y2 changed first, instead of going directly to internal state q1, the next state is given by

G(10, 11) = G(10, 01) = q2.

Because internal state q2 is also stable under input 10, the circuit can't go to the internal state q1. Thus the circuit operation will be incorrect, i.e., a transition q1=G(10, 11) races critically with the transition q2= G(10, 11). Such a situation must be avoided in case of hardware design. We will explain the premises on which our method is based, as well as the procedure which implement it.

Definition 7: (Base matrix)

Let B_n be a N x N binary identity matrix, which is called base matrix in this paper. Then the set of binary n-tuple B_n^i is the i-th column of the B_n matrix.

Definition 8: (Operator ⊕)

For each set of binary n-tuple X and Y, Let ⊕ be

	STATE	CODE
$B_3 = \begin{bmatrix}100\\010\\001\end{bmatrix}$	q_1	$B_3^1 = 100$
	q_2	$B_3^2 = 010$
	q_3	$B_3^3 = 001$
	$q_1 q_2$	$B_3^1 \oplus B_3^2 = 110$
	$q_1 q_3$	$B_3^1 \oplus B_3^3 = 101$
	$q_2 q_3$	$B_3^2 \oplus B_3^3 = 011$

Fig 3. Example of one-hot assignment.

the modular-2 addition operator such that

$X \oplus Y = <(x1+y1).mod.2, ..., (xn+yn).mod.2>$.

Definition 9: (One-hot assignment)

A one-hot assignment[15] is a state assignment in which n state variables are used to encode n states such that each state qi is assigned a binary code B_n^i and each state transition qi= G(x, qj) goes through intermediate transient state $B_n^i \oplus B_n^j$.

It should be noted that each state has a single logic 1 bit by one-hot assignment.

For example, Fig. 3 shows the result of one-hot assignment for the reduced flow table in Table I. The state variable change sequences of state transition q1=(10, q3) are as follow:

G(10, 001)=G(10,101)=G(10, 100)=q1.

Theorem 1: Critical races can be avoided by directing the circuit through intermediate transient state from which the stable state is different only one bit position, before it reaches it's final destination.[16]

Proof: It is shown that there is no race in the state transition qi=G(X, qj). The intermediate transient state, denoted by qiqj is different only one bit position from qi and qj. That is, state transition qiqj=G(X,qj) causes only i-th bit position set, and qi=G(X,qiqj) causes only j-th bit position reset. Therefore, there is no race in the state variable change sequences of the state transition qi=G(x, qj).

The basic concept of one-hot assignment is adapted to the proposed state assignment method in this paper.

Definition 10: $({}_kB_n^i)$

Let B_n be N x N binary identity matrix. The the set of binary (n-1) tuple ${}_kB_n^i$ is the i-th column of the matrix obtained from B_n by deleting the k-th row.

Definition 11: (Reduced one-hot assignment)

Reduced one-hot assignment is a state assignment in which (n-1) state variables are used to encode n state such that each state qi is assigned a binary code ${}_kB_n^i$ and each state transition qi=G(X, qj) occurs directly where i=k or j=k, or goes through intermediate transient state ${}_kB_n^i \oplus {}_kB_n^j$ where $k \neq i$ and $k \neq j$.

Lemma 1: If i is equal to k in ${}_kB_n^i$, then ${}_kB_n^i$ will be the set of binary (n-1) tuple having all its components equal to 0.

Proof) B_n^i is the set of binary n tuple having all its components equal to 0 except the i-th component, equal to 1. By the way, ${}_kB_n^i$ is the set of binary (n-1) tuple obtained from B_n^i by deleting the k-th component. If k is equal to i, then unique component equal to 1 is deleted from B_n^i. So, all components of ${}_kB_n^i$ are equal to 0.

Lemma 2: If k is greater than i, then ${}_kB_n^i$ is equal to B_{n-1}^i. And if k is less than i, then ${}_kB_n^i$ is equal to B_{n-1}^{i-1}.

Proof) By Lemma 1, ${}_kB_n^k$ is the set of binary (n-1) tuple having all its components equal to 0. And ${}_kB_n^i$ $(k \neq i)$ has only one component equal to 1. The i-th component is equal to 1 if k is greater than i. And because of the presence of ${}_kB_n^k$, the (i-1)'th component is equal to 1 if k is less than i. Therefore it is clear that ${}_kB_n^i$ is equal to B_{n-1}^i if k is greater than i, and ${}_kB_n^i$ is equal to B_{n-1}^{i-1} if k is less than i.

Theorem 2: A reduced one-hot assignment provides a critical race-free state assignment for ASC and less state transition delays than one-hot assignment, which results in the improved operating speed.

Proof: It is shown that state transition qi=G(X,qj) can occur directly or go through intermediate transient state ${}_kB_n^i \oplus {}_kB_n^j$ without races.

Case 1: By Lemma 1, binary code of state qi(qj) consists of (n-1) conjective 0's, if k is equal to i(j). Therefore, state qi(qj) is different only one bit

position from state qj(qi). State transition qi=G (X, qj) occurs directly, i.e., causes only j−th bit position reset (i-th bit position set). Races can be avoided, so critical races never occur. And there is no need of intermediate transient state, so the state transition delays can be reduced.

Case 2: By Lemma 2, state qi(qj) is equal to B^{i}_{n-1} (B^{j}_{n-1}), if k is greater than i(j). And state qi(qj) is equal to B^{i-1}_{n-1} (B^{j-1}_{n-1}), if k is less than i(j). Therefore, the state transition qi=G(X, qj) goes through intermediate transient state qiqj, to avoid critical race. The intermediate transient state qiqj is listed below.

i) $k > i$ and $k > j$ --> $B^{i}_{n-1} + B^{j}_{n-1}$ ---> ${}_kB^{i}_{n} \oplus {}_kB^{j}_{n}$

ii) $k > i$ and $k < j$ --> $B^{i}_{n-1} + B^{j-1}_{n-1}$ ---> ${}_kB^{i}_{n} \oplus {}_kB^{j}_{n}$

iii) $k < i$ and $k > j$ --> $B^{i-1}_{n-1} + B^{j}_{n-1}$ ---> ${}_kB^{i}_{n} \oplus {}_kB^{j}_{n}$

iv) $k < i$ and $k < j$ --> $B^{i-1}_{n-1} + B^{j-1}_{n-1}$ ---> ${}_kB^{i}_{n} \oplus {}_kB^{j}_{n}$

Therefore, critical race can be avoided by using intermediate transient state ${}_kB^{i}_{n} \oplus {}_kB^{i}_{n}$.

By the procedure above, we obtain a successful race-free state assignment method. And by the characteristics of the procedure, there exists only n! ways of coding for the n-state reduced flow table[13)14)] The output function is not considered, because the realization of the next state function usually plays a much more important part.

For example, consider the reduced flow table in Table 1. Since this table has 3 states, we can easily note that 2 state variables are needed, and there may be 6 ways of coding for this table.

STATE	A	B	C	D	E	F
q_1	${}_1B^{1}_{3}$	${}_1B^{1}_{3}$	${}_2B^{1}_{3}$	${}_2B^{3}_{3}$	${}_3B^{1}_{3}$	${}_3B^{2}_{3}$
q_2	${}_1B^{2}_{3}$	${}_1B^{3}_{3}$	${}_2B^{2}_{3}$	${}_2B^{2}_{3}$	${}_3B^{2}_{3}$	${}_3B^{1}_{3}$
q_3	${}_1B^{3}_{3}$	${}_1B^{2}_{3}$	${}_2B^{3}_{3}$	${}_2B^{1}_{3}$	${}_3B^{3}_{3}$	${}_3B^{3}_{3}$

According to the proposed method, a programmed procedure is listed below.

```
Procedure STATE-ASSIGN (niv, ns, nf)
integer niv, ns, nf, state
global mask, MB(1:2**(ns+niv-1)), FLOWTB(1:ns,
    1:niv, 1:2), BASE(1:ns, 1:ns), PREINT(1:2**(ns+
    niv-1), 1:nf)
// Initialize MB and PREINT array //
for i<-1 to 2**(ns+niv-1) do
    state <- i-1
    for j <- 1 to (ns+niv-1) do
        MB(1, ns+niv-j) <- state-state/2*2;
        state <- state/2;
    endfor
    for k <- 1 to nf do
        PREINT(i,k) <- don't care term;
    endfor
endfor
// Construct state table according to "next state and
    "output" " //
for state <-1 to ns
    // Calculate entry addr. for the current state in
        the state table, using matrix BASE //
    current-state(1:ns-1) <-BASE (state, 1:ns) except
        k-th column
    stpointerl <-entry addr. for the current state;
    for input <-0 to2** niv-1
        If (current entry in state table is unspecified)
            then go to next;
endif
        output <-FLOWTB(state, input, 2);
        next-state-base <-FLOWTB(state, input, 1);
        next-state(1:ns-1) <-BASE(next-state-base, 1:ns)
            except k-th column;
        case
            next state is stable or
            transition occurs directly :
            PREINT (stpointer1+input, 1:nf) <-next-
            state(1:ns-1) and binary value for output;
            transition occurs indirectly:
            // Generate intermediate transient state
                and calculate it's appropriate entry
                addr. in state table //
            transient-state(1:ns) <-
                current-state(1:ns-1).OR.next-state
                (1:ns-1);
            stpointer2 <-entry addr. for the inter-
                mediate transient state;
            PREINT (stpointer 2+input, 1:ns-1) <-
                next-state(1:ns-1);
        PREINT(stpointer1+input, 1:nf) <-
```

transient-state(1:ns-1) and binary value for output;
endcase
endfor
endfor
end STATE-ASSIGN

The procedure STATE-ASSIGN results in the binary-coded state table which consists of MB and PREINT arrays. These arrays are used for inputs of the MASK-MULTI method.

Now then, it is apprent that the proposed method is better than one-hot assignment method. For example, binary values are assigned to the states shown in Table 1, by one-hot assignment and reduced one-hot assignment. The resulting binary-coded reduced flow tables are shown in Table 2. In the case of the reduced one-hot assignment, only one case using ${}_1B_3^1$, ${}_1B_3^2$, ${}_1B_3^3$ is represented.

In Table 2.(a), three state variables are needed, and states 011,101, and 110 are used for intermediate transient states. In Table 2.(b), two state variables are needed and state 11 is used for intermediate transient state. By the existence of the all zero entry state, we can reduce one state variable, so that we can reduce the hardware cost, and also we can improve the operating speed of the resulting circuit. The improved operating speed arises from the direct transition to and from the all zero entry state. And the reason for the hardware cost reduction can be explained through the example shown in section 4.

Table 2. Binary-coded reduced flow table

$y_1\,y_2\,y_3$ \ $x_1\,x_2$	00	01	11	10
001	001	101	001	001
010	011	110	010	010
100	100	100	110	101
011	001	–	–	–
101	–	100	–	001
110	–	100	010	–

($q_1 \leftarrow 001$, $q_2 \leftarrow 010$, $q_3 \leftarrow 100$)

(a) Under one-hot assignment

$y_1\,y_2$ \ $x_1\,x_2$	00	01	11	10
00	00	10	00	00
01	00	11	01	01
10	10	10	11	00
11	–	10	01	–

($q_1 \leftarrow 00$, $q_2 \leftarrow 01$, $q_3 \leftarrow 10$)

(b) Under reduced one-hot assignment

4. Computer Programming of Asynchronous Sequential Circuit Design

In order to identify how to implement the proposed procedure on the computer, the following steps are given.

Step 1) Read the reduced flow table(FLOWTB) row by row. Each row consists of set of pairs, next state and output.

Step 2) Set up base matrix (BASE) for the convenient state assignment.

Step 3) Select a column k in base matrix, which is not used for state assignment.

Step 4) Using the procedure STATE-ASSIGN, construct the state table (MB and PREINT) according to the next states and outputs specified in the reduced flow table. Note that the unspecified entries and unused states are considered as don't care terms. The resulting state table is used for the input of the next step.

Step 5) Minimize the output functions specified in the state table, using MASK-MULTI which considers the commonality of the output functions. Now, we can obtain the minimized next state functions and output functions for the assignment in Step 4).

Step 6) Write the result.

Step 7) If all possible ways of state assignments are examined, terminte this procedure. Otherwise, go

Table 3. Reduced flow table

q \ $x_1\,x_2$	00	01	11	10
q_1	q_1\1	q_4\–	q_3\–	q_1\1
q_2	q_1\–	q_2\1	q_4\–	q_2\0
q_3	q_4\–	q_2\–	q_3\1	q_1\–
q_4	q_4\0	q_4\0	q_4\0	q_2\–

to Step 3). We explained in section 3 that this procedure will iterate n! times for the n-state reduced flow table.

Now then, let's compare the result of MASK-LASEQ with that of the conventional method, i.e., single output function minimization method with one-hot assignment. For example, a reduced flow table with 4 states and 2 input lines is shown in Table 3. According to the proposed computer program, we can obtain 24 state tables for the given table in Table 3. We can easily determine that the state assignment shown in Fig. 4 results in the minimal hardware cost circuit. That is, the number of minterms which must be generated are fewer than any others.

q \ $x_1 x_2$	00	01	11	10	
001	001\1	000\–	101\–	001\1	$q_1 \leftarrow {}_4B_4^3$
010	011\–	010\–	000\–	010\0	$q_2 \leftarrow {}_4B_4^2$
100	000\–	110\–	100\1	101\–	$q_3 \leftarrow {}_4B_4^1$
000	000\0	000\0	000\0	010\–	$q_4 \leftarrow {}_4B_4^4$
011	001\–	–––	–––	–––	
101	–––	–––	100\–	001\–	
110	–––	010\–	–––	–––	

(a) Input of the proposed algorithm MASK-LASEQ

I INPUT VARIABLES I						I COMMONALITY FOR SELECTED PARAMOUNT PI				
I	A	B	C	D	E	I	Y1	Y2	Y3	Y4
I	1	×	0	1	0	I	1	0	1	0
I	×	1	×	0	0	I	0	0	1	0
I	1	×	×	0	1	I	0	1	0	0
I	×	×	1	×	0	I	0	0	1	1
I	×	1	0	0	×	I	0	1	0	1
I	1	0	×	×	1	I	1	0	0	1
I	0	×	0	1	0	I	0	1	0	0
I	0	×	1	1	1	I	1	0	1	0

(b) Minimized output functions by MASK-LASEQ

Fig. 4. The result of the forth state assignment

The resulting output functions in sum of product form are shown as follows;

$$Y1 = A\overline{C}D\overline{E} + A\overline{B}E + \overline{A}CDE$$
$$Y2 = A\overline{D}E + B\overline{C}\overline{D} + \overline{A}\overline{C}D\overline{E}$$
$$Y3 = A\overline{C}D\overline{E} + B\overline{D}\overline{E} + \overline{A}CDE + C\overline{E}$$
$$Y4 = A\overline{B}E + B\overline{C}\overline{D} + C\overline{E}$$

Y1, Y2, and Y3 represent next state, and Y4 represents output. And A, B, and C mean next state variable and D, E mean input variable. In Fig. 5, binary-coded reduced flow table constructed by one-hot assignment is presented together with output functions minimized by K-map method. Here, Y1, Y2, Y3, and Y4 represent the next state, Y5 represents output. And A, B, C, and D mean next state variable, and E, F mean output variable.

In Fig. 5, the number of minterms needed to obtain the correctly operating sequential circuit is 14. In contrast, 8 minterms are needed by MASK-LASEQ. So, we can clearly note that the hardware cost can

q \ $x_1 x_2$	00	01	11	10	
0001	0001\1	1001\–	0101\–	0001\–	$q_1 \leftarrow B_4^4$
0010	0011\–	0010\1	1010\–	0010\0	
0100	1100\–	0110\–	0100\1	0101\–	$q_2 \leftarrow B_4^3$
1000	1000\0	1000\0	1000\0	1010\–	
0011	0001\–	–––	–––	–––	$q_3 \leftarrow B_4^2$
0101	–––	–––	0100\–	0001\–	
0110	–––	0010\–	–––	–––	$q_4 \leftarrow B_4^1$
1001	–––	1000\–	–––	–––	
1010	–––	–––	1000\–	0010\–	
1100	1000\–	–––	–––	–––	

$$Y_1 = B\overline{E}\overline{F} + \overline{B}\,\overline{C}\overline{D} + D\overline{E}F + CEF$$
$$Y_2 = DEF + \overline{A}\,\overline{C}\overline{D}$$
$$Y_3 = \overline{A}C\overline{D} + AE\overline{F} + B\overline{E}F$$
$$Y_4 = \overline{A}B\overline{C} + C\overline{E}\,\overline{F} + BE\overline{F}$$
$$Y_5 = \overline{A}\overline{E} + \overline{A}\overline{C}$$

Fig. 5. Binary-coded reduced flow table by one-hot assignment and result from the conventional method.

be reduced.

And also we can understand the merit of the application of MASK-MULTI to asynchronous sequential circuit design. If we minimize each Yi(i=1,2,3,4) specified in Fig. 4 seperately, using K-map method, we can get the result as follows;

$$Y1 = A\overline{B}E + A\overline{C}D + CDE$$
$$Y2 = A\overline{D}E + B\overline{C}\overline{D} + \overline{A}\overline{C}D\overline{E}$$
$$Y3 = \overline{A}CD + AD\overline{E} + B\overline{D}E + C\overline{E}$$
$$Y4 = B\overline{D} + A + C$$

From the above result, it is clear that 11 minterms (excluding A, C) are necessary to obtain the output functions. For these resulting functions, the common terms are not considered. The proposed method, however, minimizes the output functions not seperately but simultaneously. From the result of MASK-LASEQ, ACDE may cover ACD in Y1 and ADE in Y3, and ACDE may cover CDE in Y1 and ACD in Y3. And A, C, and BD in Y4 can be covered by the minterms which are already generated, ABE, CE, BCD. So, there is no need of generating any other minterms to obtain the function Y4. Therefore, only 8 minterms can cover all the output functions. Now then, we can conclude that the number of minterms in necessity, that is, the hardware cost for the results circuit, can be reduced by MASK-LASEQ.

5. Conclusion

The algorithm for state assignment and minimization method, which leads to race-free asynchronous sequential circuit, has been presented with some examples. This algorithm can be easily carried out on computer. Furthermore, it is possible to reduce the number of connections and gates with near minimal hardware cost. And a large number of inputs and outputs can be easily handled because they don't matter in computer programming. Note that some portions of this program should be changed for Large inputs and outputs. The example in section 4 illustrates that the proposed computer program produces the nearly optimized output functions.

However, the number of state variables increases as the number of states in the reduced flow table increases. So, reduced one-hot assignment method is not always adequate. Therefore, another programmable state assignment method with minimal state variables must be considered.

And for the convenience of the asynchronous sequential circuit design, the flow table reduction step must be added to this program.

References

1) D.A. Huffman; "The systhesis of sequential switching circuits," J. Franklin Inst., vol. 257, pp. 161-190, 275-303, Mar. 1954.
2) S.H. Unger; Theory of Asynchronous Sequential Switching Circuits. New York: Wiley, 1969.
3) C.J. Ten; "State Assignments for Asynchronous Sequential Machines," IEEE Trans. Comput., vol. c-20, pp. 382-391, Apr. 1971.
4) J.H. Tracey; "Internal state assignments for asynchronous sequential machines," IEEE Trans. Electronic Computers, vol. EC-15, pp. 551-560, Aug. 1966.
5) T. Nanya and Y. Tohma; "Universal MT state Assignments for Asynchronous Sequential Machines," IEEE Trans. Comput., vol. c-28, pp. 811-818, Nov. 1979.
6) E.J. McCluskey; "Minimization of Boolean functions," Bell Syst . Tech. J., vol. 35, pp. 1417-1424, Nov. 1956.
7) W.V. Quine; "A way to simplify truth functions," Amer. Math. Mon. Vol. 62, pp. 627-631, Nov. 1955.
8) C.N. Liu; "A state variable assignment method for asynchronous sequential circuits," J. Ass. Comput. Mach., vol. 10, pp. 209-216, Apr. 1963.
9) S.C. Lee; Digital circuits and logic design. Prentice-Hall, 1976.
10) Cho Dong Sub and Hwang Hee Yeung; "A computer algorithm for implementing the Multiple Output Switching Functions," KIEE, vol. 29, No. 10, pp. 678-688, Oct. 1980.
11) Cho Dong Sub and Hwang Hee Yeung; "On the minimization of the switching function by MASK method," KIEE, vol. 28, No. 11, pp. 801-808, Nov. 1979.
12) Z. Kohavi; Switching and Finite Automata Theory. McGraw-Hill, 1970.

13) F.J. Hill and G.R. Peterson; Introduction to Switching Theory & Logical Design. John Wiely & Sons, 1974.
14) H.C. Torns; Switching Circuit Theory and Logic Design. Addison-Wesley, 1972.
15) S.H. Unger; Asynchronous Sequential Switching Circuits. New York: Wiley, 1969.
16) B.I. Dervisglu and H.A. Sholl; "Theory and Design of Mixed-Mode Sequential Machines," IEEE Trans. Comput., vol. c-29, pp. 639-647, Jul. 1980.