PERIODICA POLYTECHNICA SER. EL. ENG. VOL. 34, NO. 2, PP. 99-111 (1990)

A MICROPROGRAM DESIGN METHOD BASED ON MEMORY BLOCKS SEPARATED BY STATES[1]

K. PÁSZTOR-VARGA

Computer and Automation Institute
Budapest, Hungary

Received: October 11, 1989.

Abstract

In this report the connections of the synthesis method of P. Arató with the automata theory are examined. A method is elaborated for realizing the automata obtained by the above synthesis. This method is based on the results obtained here and uses some earlier results concerning the realization with memory elements enabled by states. A realization with memory elements enabled by states is obtained here, too, where the input and output constraints coming from the set of disposable memory elements are taken into consideration.

A special method is elaborated for minimizing the number of variables needed in the realization of Boolean functions. The procedure is based on a new method for searching prime implicants.

The conditions of the used group of states as enabling combination is investigated and the way of its application is shown.

We point out that a post qualification of the specification is given by the result of the method minimizing the number of variables of the Boolean functions. The procedure is illustrated on an example.

Keywords: Automata theory, Boolean functions, microprogramming, logic synthesis.

Introduction

The form of the Boolean (switching) functions describing the combinatorial part of control units and the constraints concerning this form are determined by the technology of production.

In ARATÓ (1988) a method is outlined for the realization of high speed microprogrammed control units. The processor is derived from the synchronous phase register structure. The organization of the microprogram storage is determined by the states, which means that a separate microprogram field of the microprogram storage belongs to each state. To realize this correspondence a suitable statement signal is sent to the enabled input

[1]Research supported by the Hungarian National Fund for Scientific Research (OTKA). Grant No. 438.

of the memory element realizing a microprogram. The author shows that in this case the exponential growth of the number of the necessary memory units can be reduced significantly. In this way a natural reduction of the number of address lines and output lines is possible because in one state only a subset of variables is affected. In ARATÓ (1988) a rule is given for constructing microprograms. It is worth developing this idea and analysing the automata obtained by the design in ARATÓ (1987).
In this paper two main problems are examined.

1. The problem of the microprogram field separation by groups of states together with some combination of the input variables.
2. The problem of the decomposition of the output variables.

Ad 1. A minimizing method for Boolean functions (PÁSZTORNÉ-VARGA, 1987) is used. In that paper the duality property of Boolean algebras is examined and an effective synthesis algorithm for Boolean functions is derived from the results. Here these results form the basis of synthesis and analysis taking into consideration the constraints concerning the maximal number of input lines of the disposable memory elements. To the solution a convenient classification of input variables is constructed, which results in an optimal microprogram.
Ad 2. A decomposition algorithm is given for the output variables. It assures the minimal microprogram field number in the realization.

This work based on the above-mentioned results and on the theory of the deterministic automata gives a solution minimizing the number of the input and output variables of the Boolean functions, defining the combinatorial part of the control unit and reducing the number of the necessary memory elements.

Notations and Definitions

B^n — the set of the binary n-tuples or the n-dimensional Boolean space;
X — the set of the input combinations, $X^i \in X$, n_x is the number of the signals in a combination;
x_i — one input signal, $i = 1, .., n_x$;
Z — the set of the output combinations, $Z^i \in Z$, n_z is the number of the signals in a combination. z_i is one output signal, $i = 1, .., n_z$;
y — the set of the current states, $y^j \in y$, n_y is the number of the states;
Y — the set of the next states, $Y^i \in Y$;
X_s^t — the $X^s \to X^t$ change of input combination;

Z_s^t — the $Z^s \to Z^t$ change of output combination;

$z_i{:}1$ — the i-th output variable change from 0 to 1;

$z_i{:}0$ — the i-th output variable change from 1 to 0;

J_s^t — set of the prescribed input combinations occurring in a state y^j and causing the output change Z_s^t. The elements of J_s^t are called J points;

T_s^t — set of the prescribed input combinations occurring in a state y^j and inhibiting the output change Z_s^t . The elements of T_s^t are called T points;

F_s^t — the output change Z_s^t is defined by the function F_s^t , which is a not completely defined mapping $B^n \to B^m$, where n and m are the number of the input and the output variables, respectively. $(n = n_x + n_y,\ n = 2 * n_z + n_y)$. We call it *defining Boole function,* to express the relationship with the *identifying function* in ARATÓ (1987);

F_s^t and F_k^l — are *compatible* in the theory of not completely defined Boolean functions, if the set of the J points of F_s^t and the set of the T points of F_k^l are disjoint and conversely. The relation between the compatibility of functions and the compatible output changes (defined in ARATÓ (1987)) is treated in Section 3;

F_s^t and F_k^l — Boolean functions are called *disjoints* if it is true that the set of the J points of F_s^t is a subset of the T points of F_k^l and conversely.

Optimization of Defining Functions; Some Specialities

Generally a logic control procedure can be realized by a finite deterministic automaton $A(X, Y, Z, \delta, \gamma)$, where X is the set of the input signals, Y is the set of the states and Z is the set of the output signals. If this automaton is a Mealy, then δ is mapping $X \times Y \to Y$ describing state transitions and γ is a mapping $X \times Y \to Z$ describing the outputs. If this automaton is a Moore, then γ does not depend on the input, so γ is a mapping $Y \to Z$. In case of two-valued realizations X, Y, Z are n_x, n_y, n_z dimensional Boole spaces, or their subspaces satisfying some restrictions. Correspondingly the δ and γ are Boolean vector-functions or systems of Boolean functions.That is, these functions may be optimalized as a set of functions identifying output signals. The way of the optimization of the functions δ and γ is influenced by the electronic elements used in the realization.

The result of the synthesis described in ARATÓ (1987) is an automaton $B(X, y, Z, \rho)$, where ρ is a set of identifying functions of out-

put changes $F_{z_i:1}$, $F_{z_i:0}$, $(i = 1, 2, ..., n_z)$ and of state changes $F_{y_j:1}$, $F_{y_j:0}$ $(j = 1, 2, ..., n_y)$ belonging to different states. We regard the vector function $X \times y \times Z \to y \times Z$ defining the *output changes* and the *state change*. Then ρ is the *output defining function*. The difference between the output identifying function (ARATÓ, 1987) and output defining function is that the state variables and output variables are handled together in the output defining function. It can be considered that the states of the automaton B are vectors $\langle \mathbf{y}^s, \mathbf{Z}^l \rangle$, where $\mathbf{y}^s$ is rebuckled and $\mathbf{Z}^l$ is memorized. Then the number of the states of the automaton B corresponding to a state $\mathbf{y}^s$ is equal to the number of the different output changes of the automaton B in the state $\mathbf{y}^s$. In an arbitrary state $\langle \mathbf{y}^s, \mathbf{Z}^l \rangle$ the $\mathbf{y}^s$ marks the Mealy state $\mathbf{y}^s$ and $\mathbf{Z}^l$ is an output combination from which a $\mathbf{Z}_l^m$ output change exists in the state $\mathbf{y}^s$. Consequently ρ is a vector function giving the condition of the transition from the vector $\langle \mathbf{y}^s, \mathbf{Z}^l \rangle$ to the vector $\langle \mathbf{y}^j, \mathbf{Z}^m \rangle$. Later on ρ is denoted by F_l^m because the output change $\mathbf{Z}_l^m$ is described by F_l^m.

In ARATÓ (1987) the identifying functions F_l^m are optimized directly. In ARATÓ (1988) these functions are grouped according to the states $\mathbf{y}^s$ before the optimalization, since memory elements enabled by states are used in the realization. Heuristically in this grouping the number of input variables is less than originally and the exponential growth of the number of the necessary memory units can be reduced significantly.

Furthermore, we deal with the optimalization under input-output constraints of the Boolean vector functions giving a mapping $B^n \to B^m$. Our aim is to elaborate an optimal realization of the function ρ, belonging to the automaton B (ARATÓ, 1987), with memory elements enabled by states. The catalogue of the memory elements is given.

Remark 1. In general the defining functions F_l^m are not completely determined. Their values are given in the points J and T (ARATÓ, 1987, 1988), or in the points 1 and 0 (PÁSZTORNÉ-VARGA, 1987) according to another terminology. The synthesis in ARATÓ (1987) gives the points J and T.

Remark 2. Let $\langle \mathbf{y}^s, \mathbf{Z}^l \rangle$ and $\langle \mathbf{y}^s, \mathbf{Z}^k \rangle$ denote two different states of the automaton B with a common state component $\mathbf{y}^s$. In such states there are no constraints concerning the disjunctivity of the set of J points of the defining functions belonging to different output changes. Since in a given state of a Mealy automaton an input combination determines the output, in the states $\langle \mathbf{y}^s, \mathbf{Z}^l \rangle$ and $\langle \mathbf{y}^s, \mathbf{Z}^k \rangle$ the defining functions F_l^m and F_k^r may be compatible only in case of compatible output changes (ARATÓ, 1987).

Remark 3. There are no constraints concerning the compatibility of the defining functions F_l^m and F_k^r belonging to the states $\langle \mathbf{y}^s \mathbf{Z}^l \rangle$ and $\langle \mathbf{y}^t, \mathbf{Z}^k \rangle$, where $\mathbf{y}^s$ and $\mathbf{y}^t$ are different Mealy states.

It follows from remarks 1 and 2 that it is worth optimizing together only defining functions belonging to different states with a common state component $\mathbf{y}^s$, which are compatible.

Consequences of the Realization

Let $V_i = (x_{i1}, \ldots, x_{iq})$ denote the set of variables appearing in the prime implicants realizing the defining functions belonging to the state y^i. Let $K_i = (k_{i1}, \ldots, k_{ik})$ denote the set of the output changes and state changes in the state y^i. For a memory element enabled by the state y^i, the number of the address lines may not be less than iq and the number of the output lines may not be less than ik. In the realization, all variables x_{il} correspond to an address line c_j and all output changes k_{is} correspond to an output line. A prime implicant associates a deterministic value only with those address lines which correspond to variables appearing in the prime implicant. In principle to *realize a prime implicant relating to the set* V_i it is necessary to give all possible combinations of 0.1 to the address lines corresponding to the variables not appearing in the prime implicant. In the given state, all prime implicants define one output combination change. For the addresses realizing the prime implicant, the output lines corresponding to output variables affected by the above output change have the value 1, the others in K_i have the value 0.

Example 1. (The T points are not marked)

Table 1
Prime implicants corresponding to the output changes

	Prime implicants	Output changes	J – points
1.	$x_1 \neg x_2$	$z_1 : 0,\ z_3 : 1,\ y^k$	$x_1 \neg x_2 x_3 x_4 x_5$
			$x_1 \neg x_2 \neg x_3 x_4 x_5$
			$x_1 \neg x_2 x_3 \neg x_4 x_5$
2.	$x_1 x_2 x_4$	$z_1 : 1,\ z_3 : 1$	$x_1 x_2 x_3 x_4 x_5$
3.	$x_2 x_5$	$z_1 : 1,\ z_4 : 0$	$\neg x_1 x_2 x_3 \neg x_4 x_5$
			$x_1 x_2 \neg x_3 x_4 x_5$

$V = (x_1, x_2, x_4, x_5), \qquad K = (z_1 : 0, z_4 : 0, z_1 : 1, z_3 : 1, y^k)$

The synthesis procedure described in ARATÓ (1987) guarantees that if in the cases marked by $*$ the output changes are not contradictory (parallel output changes (ARATÓ, 1987)), then in the realization the row appears once in which the output combination is the union of the original ones. The merged row is:

$$* \quad 1111 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0 \quad - \quad - \quad -$$

Table 2
Correspondence of variables, correspondence of values of the prime implicants A), and their microprogram (realization) B):

	Address lines						Output lines								Output change
	1	2	3	4	5		1	2	3	4	5	6	7	8	
variables	1	2	4	5	–		$z_1:0$	$z_4:0$	$z_1:1$	$z_3:1$	y^k	–	–	–	
1.	1	0	–	–			1	0	0	1	1	–	–	–	
2.	1	1	1	–			0	0	1	1	0	–	–	–	A)
3.	–	1	–	1			0	1	1	0	0	–	–	–	
1.	1	0	0	0			1	0	0	1	1	–	–	–	
	1	0	0	1	♯		1	0	0	1	1	–	–	–	
	1	0	1	0			1	0	0	1	1	–	–	–	
	1	0	1	1	♯		1	0	0	1	1	–	–	–	
2.	1	1	1	0			0	0	1	1	0	–	–	–	
*	1	1	1	1	♯		0	0	1	1	0	–	–	–	B)
3.	0	1	0	1	♯		0	1	1	0	0	–	–	–	
	1	1	1	1	♯		0	1	1	0	0	–	–	–	
	1	1	0	1			0	1	1	0	0	–	–	–	
*	1	1	1	1	♯		0	1	1	0	0	–	–	–	

In our case, the following two statements are true related to the realization.

Statement 1. *If $V = (x_1, ..., x_r)$ is the set of the variables occurring in the conjunctions $p_1, .., p_j$, then the realization of a p_i, $1 <= i <= j$, relating to the set V is equivalent to the set of all 0.1 sequences of length r covering at least one J point covered by p_i, too.*

Proof. Because of the construction, all points J of the conjunctions $p_1, ..., p_j$ are covered at least by one sequence 0.1. Applying statement 1. the microprogram in example 1 contains only the rows marked with ♯.

Statement 2. *If F_s^t and F_k^l are disjointed, then in their realization relating to the set V of the variables occurring in their prime implicants, there is no common 0.1 sequence.*

Proof. If the statement is not true then there are such $p_i \rightarrow F_s^t$ and $p_j \rightarrow F_k^l$ that their common variables are negated equally. The common sequence obtained covers J points belonging to F_s^t and J points belonging to F_k^l, too. This contradicts the fact that F_s^t and F_k^l are disjointed.

As the defining functions belonging to not compatible output changes are disjointed, the realization described in statement 1. gives a suitable result.

In case of the realization with memory elements (microprogram), the aim of the function optimization is not the reduction of the number of the prime implicants, but the reduction of the number of different variables occurring in prime implicants necessary to the realization.

Because of the input-output constraints it can happen that more than one memory element is needed to realize a defining function. In this case, a signal controlling an output variable may appear in different memory elements. In the realization they are connected with a common wire. As a wire cannot receive more than one signal at the same time, it is necessary to assure the disjunctivity of the sets of J points of the function covered by the parts realized in different memory elements.

When the number of the input variables exceeds the input constraint, the conjunction of some input variables may be connected to the enabled input. To reduce the size of memory needed, it is suitable to analyse the disjunctive normal form of F_s^t. As in ARATÓ (1987), (1988) the identifying functions define the output change, two defining functions belong to one output. One of the defining functions determines the change $0 \rightarrow 1$ and the other the change $1 \rightarrow 0$ (*Fig. 4* in ARATÓ (1988)). Then the number of the defining functions is twice the number of the output variables.

A Procedure to the Realization with Memory Elements Enabled by the Group of States

The realization with memory elements enabled by the state may be extended to the realization with memory elements *enabled by the group of states* (by the disjunction of the states) if the J points of the defining function F_s^t belonging to the states connected to the enabled input are the same.

We have to solve three problems in the procedure:

1. Find all minimized disjunctive normal forms of the different defining functions F_s^t belonging to the states. Select the minimal forms of the defining functions F_s^t belonging to one state in which the number of variables is minimal.
2. Select the group of states to enable memory elements. Point out the defining functions F_s^t which are realized in the memory element enabled by this group of states.
3. Adequately select the enabling input variables.

Remark. If we determine the minimal number of variables to the realization of all defining functions F_s^t, then that is *a post investigation of the specification.* That is, if the automaton may be realized with fewer input variables than it was prescribed in the specification, then there is another, more effective specification, too.

The procedures developed to solve the different problems are illustrated on an example of ARATÓ (1987, 1988).

Table 3
The sets J_n^m and T_n^m obtained by the synthesis (ARATÓ, (1987), p. 136)

Current state	J and T points		Next state
(y^i)	$(J_n^m, T_n^m)(\delta)$		$(Y^j)(\gamma)$
y^1	$J_1^2 = \{X^2\}$	$T_1^2 = \{X^3, X^6, X^4, X^1, X^7, X^5\}$	
	$J_1^3 = \{X_3\}$	$T_1^3 = \{X^2, X^6, X^4, X^1, X^7, X^5\}$	Y^3
	$J_1^3/1 = \{X^4\}$	$T_2^3/1 = \{X^2, X^3, X^6, X^1, X^7, X^5\}$	Y^3
	$J_2^3/2 = \{X^7\}$	$T_2^3/2 = \{X^1, X^5, X^6, X^2, X^3, X^4\}$	
	$J_3^1 = \{X^1\}$	$T_3^1 = \{X^7, X^5, X^6, X^2, X^3, X^4\}$	
	$J_2^4 = \{X^5\}$	$T_2^4 = \{X^7, X^1, X^6, X^2, X^3, X^4\}$	Y^2
y^2	$J_4^3 = \{X^4\}$	$T_4^3 = \{X^7, X^1, X^6, X^5\}$	Y^3
	$J_4^2 = \{X^1\}$	$T_4^2 = \{X^7, X^6, X^5, X^4\}$	
	$J_2 = \{X^6\}$	$T_2 = \{X^4, X^5, X^7, X^1\}$	Y^1
y^3	$J_3^4 = \{X^5\}$	$T_3^4 = \{X^1, X^2, X^3, X^4, X^6, X^7\}$	Y^2

Table 4
Prescribed input and output combinations

X	x_1	x_2	x_3	x_4	x_5	x_6	x_7
X^1	0	0	1	0	0	0	0
X^2	1	0	1	0	0	1	0
X^3	0	0	0	0	0	0	1
X^4	1	0	0	0	1	0	0
X^5	1	1	1	0	1	1	0
X^6	1	1	1	1	1	0	1
X^7	1	0	1	1	1	1	1

Z	z_1	z_2	z_3	z_4
Z^1	0	1	0	0
Z^2	0	0	0	1
Z^3	0	0	1	0
Z^4	1	0	0	1

Ad 1. By applying the procedure described in PÁSZTORNÉ-VARGA (1987) all prime implicants of the functions F_s^t are generated. These are listed in *Table 5* according to the order in *Table 3*. In case of this example the functions may be realized with any of their prime implicants.

If the above covers are available, then a covering problem must be solved, which is the search of the forms of the defining functions F_i^j belonging to a given state which contain a minimal number of variables.

Table 5
Defining functions

F_1^2 :	$\{\neg x_5 x_6,\ x_1 x_3 \neg x_5,\ \neg x_2 \neg x_4 x_6,\ \neg x_2 x_6 \neg x_7,\ x_1 \neg x_2 x_3 \neg x_7,\ x_1 \neg x_2 x_3 \neg x_4\}$
F_1^3 :	$\{\neg x_1 \neg x_3,\ \neg x_3 \neg x_5,\ \neg x_1 x_7,\ \neg x_4 x_7,\ \neg x_5 x_7,\ \neg x_2 x_6 x_7\}$
$F_2^3/1$:	$\{x_1 \neg x_3,\ \neg x_3 x_5,\ \neg x_3 \neg x_7,\ \neg x_2 \neg x_4 x_5,\ \neg x_4 x_5 \neg x_6,\ x_5 \neg x_6 \neg x_7,$ $x_1 \neg x_4 \neg x_6,\ x_1 \neg x_2 \neg x_6,\ x_1 \neg x_6 \neg x_7\}$ **
$F_2^3/2$:	$\{\neg x_2 x_4,\ \neg x_2 x_5,\ \neg x_2 x_7,\ x_4 x_6,\ x_6 x_7\}$
F_2^4 :	$\{x_2 \neg x_4,\ x_2 x_6,\ x_2 \neg x_7,\ x_3 \neg x_4 x_5,\ \neg x_4 x_5 x_6,\ x_5 x_6 \neg x_7,\ x_3 x_5 \neg x_7\}$ ***
F_3^1 :	$\{\neg x_1 x_3,\ \neg x_1 \neg x_7,\ \neg x_5 \neg x_6 \neg x_7,\ \neg x_2 x_3 \neg x_6,\ x_3 \neg x_4 \neg x_6,\ x_3 \neg x_5 \neg x_6,$ $x_3 \neg x_6 \neg x_7\}$ *
F_4^3 :	$\{\neg x_3,\ x_a \neg x_2 \neg x_4,\ \neg x_1 \neg x_2 \neg x_7,\ x_1 \neg x_4 \neg x_6,\ x_1 \neg x_6 \neg x_7,\ x_5 \neg x_2 \neg x_4,$ $\neg x_1 \neg x_2 \neg x_7,\ x_1 \neg x_4 \neg x_6,\ x_1 \neg x_6 \neg x_7\}$ **
F_4^2 :	$\{\neg x_1,\ \neg x_5,\ x_3 \neg x_6,\ \neg x_2 x_3 \neg x_4,\ \neg x_2 x_3 \neg x_7\}$ *
F_2 :	$\{x_2 x_4,\ x_2 \neg x_6,\ x_2 x_7,\ x_4 \neg x_6,\ \neg x_6 x_7,\ x_1 x_3 \neg x_6,\ x_3 x_5 \neg x_6\}$
F_3^4 :	$\{x_2 \neg x_4,\ x_2 x_6,\ x_2 \neg x_7,\ x_3 \neg x_4 x_5,\ x_3 x_5 \neg x_7,\ \neg x_4 x_5 x_6,\ x_5 x_6 \neg x_7\}$ ***

The algorithm begins with a table (T1) in which the input variables are associated to the columns. The rows are associated to the prime implicants of the functions. In the row of a prime implicant there is a + in the columns of the variables occurring in the prime implicant (*Table 6*).

In the table x_i is designated by i and the variables occurring in a prime implicant are given by the list of the indices of the variables occurring in it. The table (Tk), $k = 2, .., n$, is constructed from (Tk-1) in the third step of the algorithm. The sequence $(i_1, .., i_n)$ below the sign (Tk) shows the variables selected during the construction of (Tk).

The algorithm

0. k:=0.
1. Search for variables occurring alone in a row of (Tk). If there are such variables, then select only one from a cover of an F_i^j. Then step 3 follows.
2. Search for a column of (Tk) which contains for maximal number of F_i^j the sign +. If there are such columns, then choose only one and select the variable belonging to this column.
3. $k := k + 1$. Generate a new table (Tk) in which the columns of the selected variables do not appear. For functions with prime implicants having selected variables cancel all prime implicants not containing selected variables and cancel the selected variables from the other prime implicants. There is no change in the subtable, if the prime implicants of a function do not contain selected variables. If the new table does not contain any columns, then the set of the selected vari-

Table 6
Functioning of the algorithm

	(T1)								(T2) (7)						(T3) (6)			(T4) (5)	(T5) (2)	
	1	2	3	4	5	6	7		1	2	3	4	5	6	1	2	5	2		
5,6	−	−	−	−	+	+	−	−												
1,3,5	+	−	+	−	+	−	−	−												
2,4,6	−	+	−	+	−	+	−	−												F_1^2
2,6,7	−	+	−	−	−	+	+	2,6	−	+	−	−	−	+	2	−	+	−	2 +	x
1,2,3,7	+	+	+	−	−	−	+	1,2,3	+	+	+	−	−	−	−					
1,2,3,4	+	+	+	+	−	−	−	−												
1,3	+	−	+	−	−	−	−	−												
3,5	−	−	+	−	+	−	−	−												
1,7	+	−	−	−	−	−	+	1	+	−	−	−	−	−	−					F_1^3
4,7	−	−	−	+	−	−	+	4	−	−	−	+	−	−	−					
5,7	−	−	−	−	+	−	+	5	−	−	−	−	+	−	−					
2,6,7	−	+	−	−	−	+	+	2,6	−	+	−	−	−	+	2	−	+	−	2 +	x
1,3	+	−	+	−	−	−	−	−												
3,5	−	−	+	−	+	−	−	−												
3,7	−	−	+	−	−	−	+	3	−	−	+	−	−	−	−					
2,4,5	−	+	−	+	+	−	−	−												
4,5,6	−	−	−	+	+	+	−	−												$F_2^3/1$
5,6,7	−	−	−	−	+	+	+	5,6	−	−	−	−	+	+	5	−	−	+	x	
1,4,5	+	−	−	+	−	+	−	−												
1,2,6	+	+	−	−	−	+	−	−												
1,6,7	+	−	−	−	−	+	+	1,6	+	−	−	−	−	+	1	+	−	−	−	
2,4	−	+	−	+	−	−	−	−												
2,5	−	+	−	−	+	−	−	−												
2,7	−	+	−	−	−	−	+	2	−	+	−	−	−	−	−					$F_2^3/2$
4,6	−	−	−	+	−	+	−	−												
6,7	−	−	−	−	−	+	+	6	−	−	−	−	−	+	x					
2,4	−	+	−	+	−	−	−	−												
2,6	−	+	−	−	−	+	−	−												
2,7	−	+	−	−	−	−	+	2	−	+	−	−	−	−	−					
3,4,5	−	−	+	+	+	−	−	−												F_2^4
4,5,6	−	−	−	+	+	+	−	−												
5,6,7	−	−	−	−	+	+	+	5,6	−	−	−	−	+	+	5	−	−	+	x	
3,5,7	−	−	+	−	+	−	+	3,5	−	−	+	−	+	−	−					
1,3	+	−	+	−	−	−	−	−												
1,7	+	−	−	−	−	−	+	1	+	−	−	−	−	−	−					
5,6,7	−	−	−	−	+	+	+	5,6	−	−	−	−	+	+	5	−	−	+	x	
2,3,6	−	+	+	−	−	+	−	−												F_3^1
3,4,6	−	−	+	+	−	+	−	−												
3,5,6	−	−	+	−	+	+	−	−												
3,6,7	−	−	+	−	−	+	+	3,6	−	−	+	−	−	+	3	−	+	−	−	

ables are sufficient for the realization. The algorithm ends. Otherwise step 1 follows.

For the defining functions F_i^j belonging to state 1, the table (T1) and the functioning of the algorithm is illustrated in *Table 6.*

According to the result, all functions belonging to state 1 can be realized with the variables x_2, x_5, x_6, x_7. *Table 7* shows this:

Table 7
Realization of functions in state 1

column (A) the minimal forms of the functions
column (B) the realization in case of four address lines
column (C) the output changing belonging to the functions

(A)	(B)	(C)		
$F_1^2 = \neg x_2 x_6 \neg x_7$	$\neg x_2 \neg x_5 x_6 \neg x_7$	$z_2:0$	$z_4:1$	
$F_1^3 = \neg x_2 x_6 x_7$	$\neg x_2 \neg x_5 x_6 x_7$	$z_2:0$	$z_3:1$	Y^3
$F_2^3/1 = x_5 \neg x_6 \neg x_7$	$\neg x_2 x_5 \neg x_6 \neg x_7$	$z_4:0$	$z_3:1$	Y^3
$F_2^3/2 = x_6 x_7$	$\neg x_2 x_5 x_6 x_7$	$z_4:0$	$z_3:1$	
$F_2^4 = x_5 x_6 \neg x_7$	$x_2 x_5 x_6 \neg x_7$	$z_1:1$		Y^2
$F_3 = \neg x_5 \neg x_6 \neg x_7$	$\neg x_2 \neg x_5 \neg x_6 \neg x_7$	$z_3:0$	$z_2:1$	

Ad 2. A memory element may be enabled by a group of states, if the functions realized in it are defining functions in all states from the group and all functions affect the same output changes. In our example, if the memory element realizes $F_2^3/1$ from state y^1 and F_4^3 from state y^2 with the output changes $z_3:1, z_4:0, Y^3$ then the memory may be enabled by $y^1 \vee y^2$.

Ad 3. If the necessary number of inputs exceeds the input constraint by n_q, then it is useful to select n_q columns in which there are minimal numbers of different 0.1 combinations. The enabling variables are the variables belonging to these columns. The needed memory number ahead of 2^{nq} is the number of the appearing combinations.

An Illustration

We show realizations for the above example. The constraints: input number may be maximum 12, output number may be maximum 8.

Table 8 shows in any state the groups of input variables, which are suitable to realize the defining functions. As it is easy to see for all states two groups of variables may be suitable. In version 1 the group consists of 2,3,4,5,6 and in version 2 the group consists of 2,3,5,6,7. In *Table 8* the rows belonging together are marked with the version numbers.

Table 8
The variables needed in the states

state	1	2	3	4	5	6	7	version
1	−	+	+	−	−	+	+	2
	−	+	−	−	+	+	+	2
	−	+	+	+	+	+	−	1
2	+	−	+	−	−	+	−	
	−	−	+	−	+	+	−	2
	+	+	+	+	−	−	−	
	−	+	+	+	+	−	−	1
	+	+	+	−	−	−	+	
3	−	+	−	+	−	−	−	1
	−	+	−	−	+	−	−	1,2
	−	+	−	−	−	−	+	
	−	−	+	+	+	−	−	1
	−	−	+	−	+	−	+	
	−	−	−	+	+	+	−	1
	−	−	−	−	+	+	+	

Table 9
Version 2 with the variables 2,3,4,5,6

memory 1 enabled by state 1									function
23456	$z_2:0$	$z_3:0$	$z_4:0$	$z_2:1$	$z_3:1$	$z_4:1$	Y^3		$y^1(\neg x_5 x_6 \vee$
12345	1	2	3	4	5	6	7		
01001	1	0	0	0	0	1	0	F_1^2	$\neg x_3 \neg x_5 \vee \neg x_3 x_5 \vee$
00001	1	0	0	0	1	0	1	F_1^3	$\neg x_2 x_4 \vee$
00010	0	0	1	0	1	0	1	$F_2^3/1$	$\neg x_2 x_3 \neg x_6)$
01111	0	0	1	0	1	0	0	$F_2^3/2$	
01000	0	1	0	1	0	0	0	F_3^1	
memory 2 enabled by state 1									
23456	$z_1:1$	Y^2							
12345	1	2							$y^1 x_2 \neg x_4$
11011	1	1							
memory 3 enabled by state 2									
23456	$z_1:0$	$z_4:0$	$z_3:1$	Y^1	Y^3				
12345	1	2	3	4	5				
00010	1	1	1	0	1			F_4^3	$y^2(\neg x_3 \vee x_3 \neg x_6 \vee x_2 x_4)$
01000	1	0	0	0	0			F_4^2	
11110	0	0	0	1	0			F_2	
memory 4 enabled by state 3									
23456	$z_3:0$	$z_1:1$	$z_4:1$	Y^2					
12345	1	2	3	4					$y^3 x_2 \neg x_4$
11011	1	1	1	1				F_3^4	

As the post-investigation of the specification, *Table 8* shows that ahead of the original seven variables five variables are enough to specify the task. The variable x_1 can be cancelled and one of the variables x_4 ,x_7 can be cancelled, too.

In our example the input constraint does not cause a problem. In state 1, the number of output changes is 9 (*Table 7*), then due to the output constraint, the defining functions in state 1 are not realizable with one memory element. As F_2^4 in the state 1 and F_3^4 in the state 2 are compatible, the problem is solved by their realization with a memory element enabled by $y^1 \vee y^3$, where the outputs are $z_1 : 1, Y^2$ (version 1). In our example the solution is simpler if the defining functions in state 1 are realized with two memory elements (version 2).

In version 1, the number of the realizing memory is greater by 1, but in another case, the above type of realization may be more effective.

Conclusion

A special method is elaborated for minimizing the number of variables needed in the realization of Boolean functions. The realization with memory elements causes constraints of the number of the input and the output variables. Such constraints and other specialities of this realization are taken into consideration. As a consequence of the algorithm minimizing the number of input variables a post qualification of the specification is obtained.

References

ARATÓ, P. (1987): Specification and Realisation of Logic Control Procedures on the Basis of Prescribed Input-Output Changes. *Periodica Polytechnica Ser. El. Eng.* Vol. 31. Nos. 3-4, pp. 99–153.

ARATÓ, P. (1988): Design of Microprogrammed Control Units Using Separate Microprogram Fields Enabled by States. *Periodica Polytechnica Ser. El. Eng.* Vol. 32. Nos. 2-4, pp. 97–110.

PÁSZTORNÉ VARGA, K. (1987): A Boole függvények Boole algebrájának strukturális tulajdonságait felhasználó Boole függvény optimalizációs módszer. *Alkalmazott Matematikai Lapok 13* (1987-88) pp. 69–96. (in Hungarian)

Address:

K. PÁSZTOR-VARGA
Computer and Automation Institute
P. O. Box 63, Budapest, Hungary
H-1502