DESIGN OF MICROPROGRAMMED CONTROL UNITS USING SEPARATE MICROPROGRAM FIELDS ENABLED BY STATES

P. ARATÓ

Department of Process Control,
Technical University, H-1521 Budapest

Received June 1, 1987

Abstract

In this report a method is outlined for the realisation of high-speed microprogrammed control units. The processor is derived from the synchronous phase register structure. The organisation of the micoprogram storage is determined by the states (phases). It means that a separate field of the microprogram storage belongs to each state. In this way, the exponential growth of the number of the necessary memory units can be reduced significantly. This advantage is derived mainly from the separate handling of the storage fields, because each field mutually corresponds to one state, in which only a subset of the input and output signals is affected. The elements of these subsets are determined by the design procedure of the control unit. Thus, the minimization of the Boolean functions, defining the combinational part of the control unit, and the state reduction can be utilized in the simplification of a microprogrammed realisation.

In the paper, the minimal forms of the Boolean functions and the minimal number of states (phases) are assumed; the influence of the separate micoprogram storage fields on the number of the necessary memory units is examined; a rule is given for constructing the microprograms.

Introduction

The control unit design methods based on flow-chart [1]—[4] provides the expressions of the Boolean functions representing the combinational part of the control unit to be realized in a uniform fixed hardware structure. If the combinational part is built from memory units (EPROM-s), then the remaining parts of the uniform fixed hardware structure can be considered to be drawn together as a processor. This yields a kind of microprogrammed structure. In this case, the word-organisation of the microinstructions is determined by the networks drawn together in the processor.

One of the uniform fixed hardware structure used frequently for control units, is the synchronous phase register structure [4], [6], [7], [9], [12], [13] shown in Fig. 1. The combinational part (C) may be realized by memory units. The remaining parts are the phase register, the clock-enabling network, the

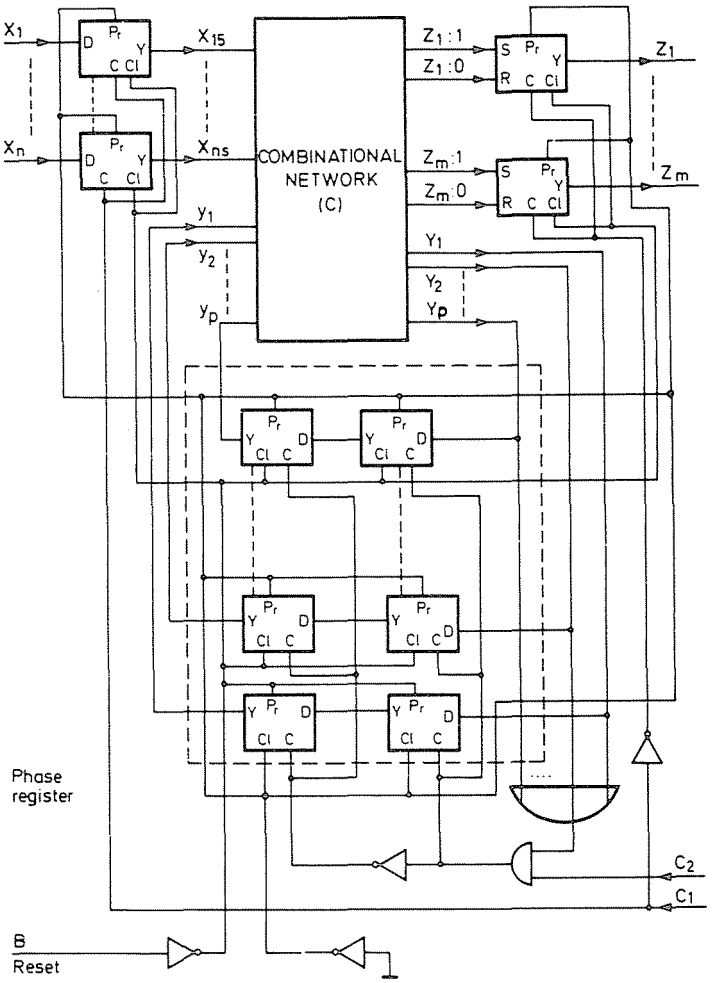

Fig. 1

input-output flip-flops and the clock generator. These remaining networks may be considered to form the processor according to Fig. 2.

In this way, the synchronous phase register structure can be considered as a microprogrammed one. The microprogram is to be constructed from the Boolean expressions, provided by the hardware design procedure. The word-organisation of the microinstructions is determined by the processor.

This approach yields a faster microprogrammed control unit with a smaller microprogram storage than the full software or the simple flow table implementation approaches [2], [3], [9], [14]. The speed is higher, because there is a method for the state assignment [12], which ensures the highest speed

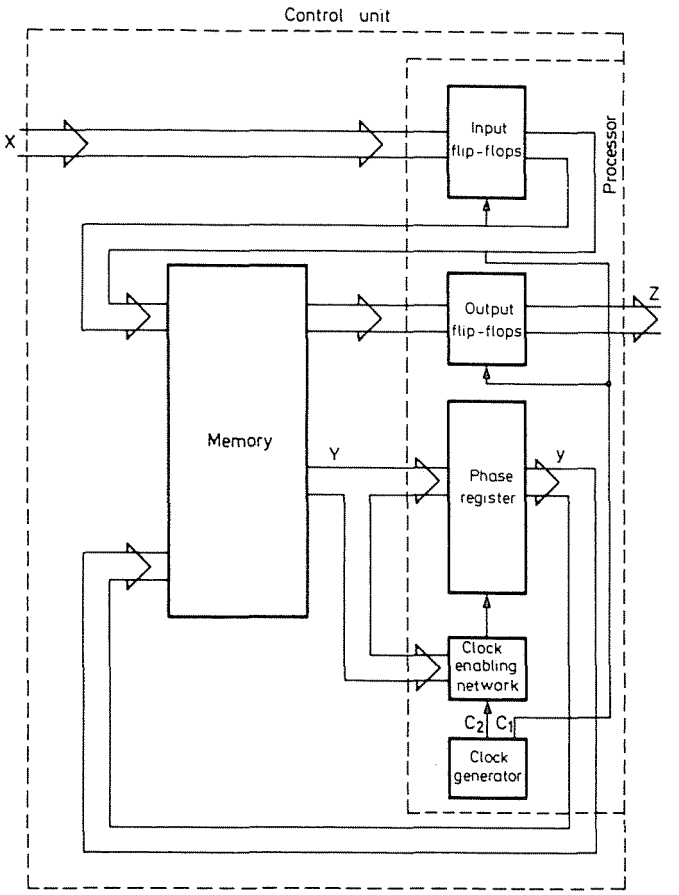

Fig. 2

allowed by the building blocks. Furthermore, the synchronous phase register structure can be modified slightly, in order to operate with a one-phase clock [12]. This modification may provide twice as high a speed than the structure in Fig. 1.

The smaller size of microprogram storage is achieved by handling the storage in separate fields defined by the states. It means that each field mutually corresponds to one state, in which only a subset of the input and output signals is affected. The elements of these subsets are determined by the design procedure of the control unit. In each field of the storage, a different bit-map may be formed. Thus, the numbers of the necessary address and data lines in each storage field can be reduced according to the simplicity of the Boolean expressions provided by the design procedure. In this way, the exponential growth of the number of the necessary memory units is significantly reducible.

Later in this paper, the minimal forms of the Boolean functions and the minimal number of states (phases) will be assumed; the influence of the separate micoprogram storage fields on the number of the necessary memory units will be examined and a rule will be given for constructing the microprograms.

Defining the storage fields by the Boolean expressions of the combinational part

The "1-from-n" code of the states in a synchronous phase register structure involves that the Boolean expressions realizing the combinational part have the forms as follows [6], [13]:

$$z_i : 1 = \sum_{k=1}^{p} (y_k \cdot F_{z_i}^k)$$

$$z_i : 0 = \sum_{k=1}^{p} (y_k \cdot F_{\bar{z}_i}^k)$$

$$Y_i = \sum_{k=1}^{p} (Y_k \cdot F_{y_i}^k)$$

where y_k is the k-th secondary variable;

$\left\{ \begin{matrix} F_{z_i}^k \\ F_{\bar{z}_i}^k \\ F_{y_i}^k \end{matrix} \right\}$ is the identifying function [13], the value 1

of which defines the input conditions for the change

$$\text{of} \left\{ \begin{matrix} \text{the output variable } z_i \text{ from 0 to 1} \\ \text{the output variable } z_i \text{ from 1 to 0} \\ \text{the secondary variable } Y_i \text{ from 0 to} \end{matrix} \right\} \text{in the}$$

state y_k. Following from the "1-from-n" code, $y_k = 1$ represents a state called y_k.

$$\text{If the change of} \quad \left\{ \begin{matrix} z_i \text{ from 0 to 1} \\ z_i \text{ from 1 to 0} \\ Y_i \text{ from 0 to 1} \end{matrix} \right\}$$

is not specified in a given state y_k, then the value of

$$\begin{Bmatrix} F^k_{z_i} \\ F^k_{\bar{z}_i} \\ Y^k_i \end{Bmatrix}$$

is a constant 0 independently from the input conditions.

The variable $\begin{Bmatrix} z_i : 1 \\ z_i : 0 \\ Y_i \end{Bmatrix}$

denotes the logical sum of the conditions for changing

$$\begin{Bmatrix} z_i \text{ from 0 to 1} \\ z_i \text{ from 1 to 0} \\ Y_i \text{ from 0 to 1} \end{Bmatrix}.$$

$\sum_{k=1}^{p}$ denotes the logical sum of the expressions between the brackets according to k.

Example

Let it be assumed that the expressions realizing the combinational part are as follows:

$$z_i : 1 = y_1 x_2 x_6 + y_3 x_2 x_6$$

$$z_1 : 0 = y_2(\bar{x}_3 x_5 + \bar{x}_1)$$

$$z_2 : 1 = y_1 \bar{x}_1$$

$$z_2 : 0 = y_1(\bar{x}_5 x_6 + \bar{x}_3 x_5)$$

$$z_3 : 1 = y_1(\bar{x}_1 \bar{x}_3 + \bar{x}_3 x_5 + \bar{x}_2 x_4) + y_2 \bar{x}_3 x_5$$

$$z_3 : 0 = y_1 \bar{x}_1 x_3 + y_3 x_2 x_6$$

$$z_4 : 1 = y_1 \bar{x}_5 x_6 + y_3 x_2 x_6$$

$$z_4 : 0 = y_1(\bar{x}_3 x_5 + \bar{x}_2 x_4) + y_2 \bar{x}_3 x_5$$

$$Y_1 = y_2 x_3 x_5$$

$$Y_2 = y_1 x_2 x_6 + y_3 x_2 x_6$$

$$Y_3 = y_1(\bar{x}_3 x_5 + \bar{x}_1 \bar{x}_3) + y_2 \bar{x}_3 x_5$$

In this way, the algebraic forms of the identifying functions are also given. For example

$$F^1_{z_2} = \bar{x}_1; \quad F^1_{\bar{z}_2} = \bar{x}_5 x_6 + \bar{x}_3 x_5; \quad F^2_{Y_1} = x_3 x_5;$$

$$F^2_{z_2} \equiv F^3_{z_2} \equiv F^2_{\bar{z}_2} \equiv F^3_{z_2} \equiv F^3_{Y_1} \equiv F^1_{y_1} \equiv \text{ etc.} \ldots \equiv 0$$

It can be observed that there exist input combinations which cause, for example:

$$F^1_{z_2} = F^1_{\bar{z}_2} = 1$$

It is obvious that these input combinations do not occur in the state y_1 during the specified operation of the control unit. In the opposite case, the state would have been otherwise defined by the design procedure [12], [13].

If the combinational part is realized by memory units according to Fig. 2, then the numbers of the necessary input and output lines of the whole storage are, with the notations of Fig. 3:

$$r \geq n + p$$

$$s \geq 2m + p$$

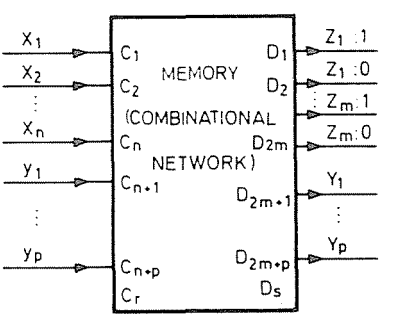

Fig. 3

If the codes of the states are assumed to be decoded at the input of the storage from "1-from-n" to binary, then

$$r \geq n + \lfloor {}^2\log p \rfloor,$$

where $\lfloor {}^2\log p \rfloor$ denotes the smallest integer, which is higher or equal to ${}^2\log p$.

It is well-known that the value of r is the most dominant factor, which determines, in most cases, how many memory units are necessary for the realisation of the whole storage. If the number of the necessary inputs exceeds the value of r belonging to the memory units to be applied, then the number of the necessary memory units for the whole storage grows exponentially depending on the extent of exceeding.

This exponential growing can be reduced if the expressions, realizing the combinational part, will be handled in groups defined by the states. In this way, the input conditions will be collected for each state, which cause the output and secondary changes specified in that state.

Let the identifying functions be grouped for our example in this way:

$$F^1_{z_1},\ F^1_{z_2},\ F^1_{\bar{z}_2},\ F^1_{z_3},\ F^1_{z_4},\ F^1_{\bar{z}_3},\ F^1_{\bar{z}_4},\ F^1_{Y_2},\ F^1_{Y_3}$$

$$F^2_{\bar{z}_1},\ F^2_{z_3},\ F^2_{\bar{z}_4},\ F^2_{Y_1},\ F^2_{\bar{Y}_3}$$

$$F^3_{z_1},\ F^3_{\bar{z}_3},\ F^3_{z_4},\ F^3_{Y_2}$$

Let X/F^k denote the set of the input variables, on which at least one of the identifying functions, belonging to y_k, depends.

In our example:

$$X/F^1 = \{x_1, x_2, x_3, x_4, x_5, x_6\}$$

$$X/F^2 = \{x_1, x_3, x_5\}$$

$$X/F^3 = \{x_2, x_6\}$$

Let Z/y_k denote the set of the output changing variables $(z_i : 1, z_i : 0)$, which may have the value 1 in the state y_k.

In our example:

$$Z/y_1 = \{z_1 : 1, z_2 : 1, z_2 : 0, z_3 : 1, z_4 : 1, z_4 : 0\}$$

$$Z/y_2 = \{z_1 : 0, z_3 : 1, z_4 : 0\}$$

$$Z/y_3 = \{z_1 : 1, z_3 : 0, z_4 : 1\}$$

Let Y/y_k denote the set of the secondary changing variables (Y_i), which may have the value 1 in the state y_k.

In our example:

$$Y/y_1 = \{Y_2, Y_3\}$$

$$Y/y_2 = \{Y_1, Y_3\}$$

$$Y/y_3 = \{Y_2\}$$

With the above notations, the whole storage, realizing the combinational part in Fig. 3, can be built up from memory units according to Fig. 4 making use of the enable inputs (E). The sets X/F^k, Z/y_k, Y/y_k are established by wiring at the inputs of the memory units. The outputs $z_i:1, z_i:0$, Y_i can be formed also by a simple wiring, because the memory units are supposed to have "tri-state" outputs. The different sets X/F^k, Z/y_k, Y/y_k, may have a different number of elements and so, memory units with different capacity can be mixed in wiring the whole storage.

The content of the storage is determined by the expressions realizing the combinational part. This content is considered as the microprogram and it can be constructed for Fig. 4 as follows.

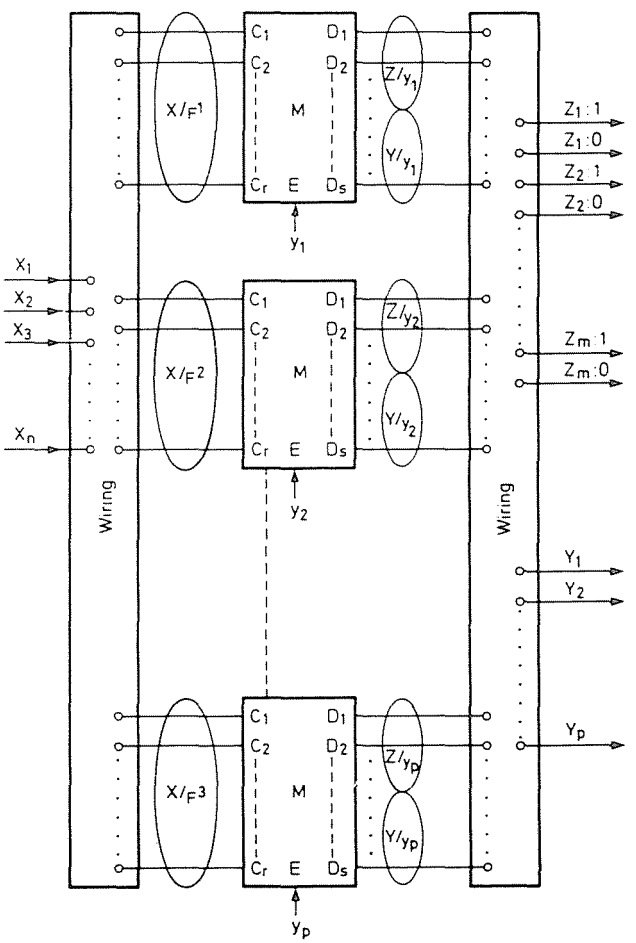

Fig. 4

Constructing the microprogram

The number of the necessary outputs of the memory units enabled in the state y_k is equal to the number of elements of the set $Z/y_k \bigcup Y/y_k$. Thus, the bit-map of a memory unit is determined by the elements of the set $Z/y_k \bigcup Y/y_k$.

Let $\begin{Bmatrix} C^k_{z_i} \\ C^k_{\bar{z}_i} \\ C^k_{Y_i} \end{Bmatrix}$ denote the set of the binary combinations, interpreted on the elements of the set

$$\begin{Bmatrix} X/F^k_{z_i} \\ X/F^k_{\bar{z}_i} \\ X/F^k_{Y_i} \end{Bmatrix}, \qquad \text{for which} \qquad \begin{Bmatrix} F^k_{z_i}=1 \\ F^k_{z_i}=1 \\ F^k_{Y_i}=1 \end{Bmatrix}$$

Let $\begin{Bmatrix} D^k_{z_i} \\ D^k_{\bar{z}_i} \\ D^k_{Y_i} \end{Bmatrix}$ denote the output of the memory unit, enabled in the state y_k, which contributes to the forming of the output

$$\begin{Bmatrix} z_i : 1 \\ z_i : 0 \\ Y_i \end{Bmatrix}.$$

With the above notations, the content of the memory unit enabled in the state y_k can be determined by the rule as follows:

The bit $\begin{Bmatrix} D^k_{z_i} \\ D^k_{\bar{z}_i} \\ D^k_{Y_i} \end{Bmatrix}$ must have the value 1 in each address, which corresponds to the elements of the set $\begin{Bmatrix} C^k_{z_i} \\ C^k_{\bar{z}_i} \\ C^k_{Y_i} \end{Bmatrix}$

In the remaining addresses, the bit $\begin{Bmatrix} D^k_{z_i} \\ D^k_{\bar{z}_i} \\ D^k_{Y_i} \end{Bmatrix}$ must have a 0 value.

If the address fields corresponding to $C^k_{z_i}$ and $C^k_{\bar{z}_i}$ are overlapping, that is $C^k_{z_i}$ and $C^k_{\bar{z}_i}$ are not disjointed, then the values of the bits $D^k_{z_i}$ and $D^k_{\bar{z}_i}$ may be arbitrary in the common address field. This case can occur only for the input combinations causing $F^k_{z_i} = F^k_{\bar{z}_i} = 1$, which is excluded by a proper state definition [12], [13].

X/F^2	x_1	x_3	x_5	$D^2_{z_1}$	$D^2_{z_3}$	$D^2_{\bar{z}_4}$	$D^2_{Y_1}$	$D^2_{Y_3}$
	C_1	C_2	C_3	D_1	D_2	D_3	D_4	D_5
	0	0	0	1	0	0	0	0
	0	0	1	1	1	1	0	1
	0	1	0	1	0	0	0	0
	0	1	1	1	0	0	1	0
	1	0	0	0	0	0	0	0
	1	0	1	1	1	1	0	1
	1	1	0	0	0	0	0	0
	1	1	1	0	0	0	1	0

Fig. 5

The above rule yields the content of the memory unit enabled by the state y_2 in our example as shown in Fig. 5. The identifying functions belonging to the state y_2 determine that the outputs of this memory unit correspond to the bits $D^2_{\bar{z}_1}$, $D^2_{z_3}$, $D^2_{\bar{z}_4}$, $D^2_{Y_1}$, $D^2_{Y_3}$. The identifying functions belonging to the state y_2:

$$F^2_{\bar{z}_1} = \bar{x}_3 x_5 + \bar{x}_1$$

$$F^2_{z_3} = \bar{x}_3 x_5$$

$$F^2_{\bar{z}_4} = \bar{x}_3 x_5$$

$$F^2_{Y_1} = x_3 x_5$$

$$F^2_{Y_3} = \bar{x}_3 x_5$$

Assuming the bit order $x_1 x_3 x_5$:

$$C^2_{\bar{z}_1} = \{000, 001, 010, 011, 101\}$$

$$C^2_{z_3} = C^2_{\bar{z}_4} = C^2_{Y_3} = \{001, 101\}$$

$$C^2_{Y_1} = \{011, 111\}$$

Calculating the effect of the storage fields separated by the states

In this examination, the pessimistic case will be assumed, where every X/F^k set contains all of the input variables. Even in this pessimistic case, it can be proved, by a very simple calculation, that the solution shown in Fig. 4

reduces the exponential growing of the number of the necessary memory units, if the network to be realized has more inputs than r. Let it be assumed that the number of outputs on one memory unit is satisfactory for the realisation of the whole storage.

At first, let the case $n \geq r$ be examined. In this case, the number of the necessary memory units according to Fig. 3 is 2^{n+p-r}.

In the solution shown in Fig. 4, the number of the necessary memory units can be calculated as follows:

$$p \cdot 2^{n-r}$$

The solution of Fig. 4 is more advantageous than that in Fig. 3, if

$$2^{n+p-r} > p \cdot 2^{n-r},$$

that is

$$2^p \cdot 2^{n-r} > p \cdot 2^{n-r}$$

$$2^p > p,$$

which is always fulfilled.

Now, let the case $n < r$, $n + p > r$ be examined. In this case, the solution in Fig. 4 needs p memory units, but according to Fig. 3, the number of the necessary memory units remains

$$2^{n+p-r}$$

The solution of Fig. 4 is advantageous if

$$2^{n+p-r} > p,$$

that is

$$n + p - r > {}^2\log p$$

$$p - {}^2\log p > r - n.$$

This result shows that the value of $r-n$ determines the minimal value of p, for which the solution in Fig. 4 becomes advantageous, even if the pessimistic assumption for the set X/F^k is held.

After this, let both cases be examined, assuming that the secondary combinations in Fig. 3 arrive decoded into binary form at the inputs of the storage.

$n \geq r$:

$$2^{n+2\log p-r} > p \cdot 2^{n-r}$$

$$2^{2\log p} \cdot 2^{n-r} > p \cdot 2^{n-r}$$

$$2^{2\log p} > p$$

This inequality is fulfilled, when the value of p if not equal to any integer power of 2. The design methods [7], [12], [13] provide minimized identifying functions and so, the sets X/F^k contain generally less elements than n. Thus, the

factor beside p in the initial inequality is significantly smaller than 2^{n-r}. As a consequence, the number of the necessary memory units is practically reduced by the solution in Fig. 4 even in this case.

$n<r, \qquad n+\lfloor {}^2\log p \rfloor > r$:

$$2^{n+\lfloor {}^2\log p \rfloor - r} > p,$$

that is

$$n+\lfloor {}^2\log p \rfloor - r > {}^2\log p$$

$$\lfloor {}^2\log p \rfloor - {}^2\log p > r-n,$$

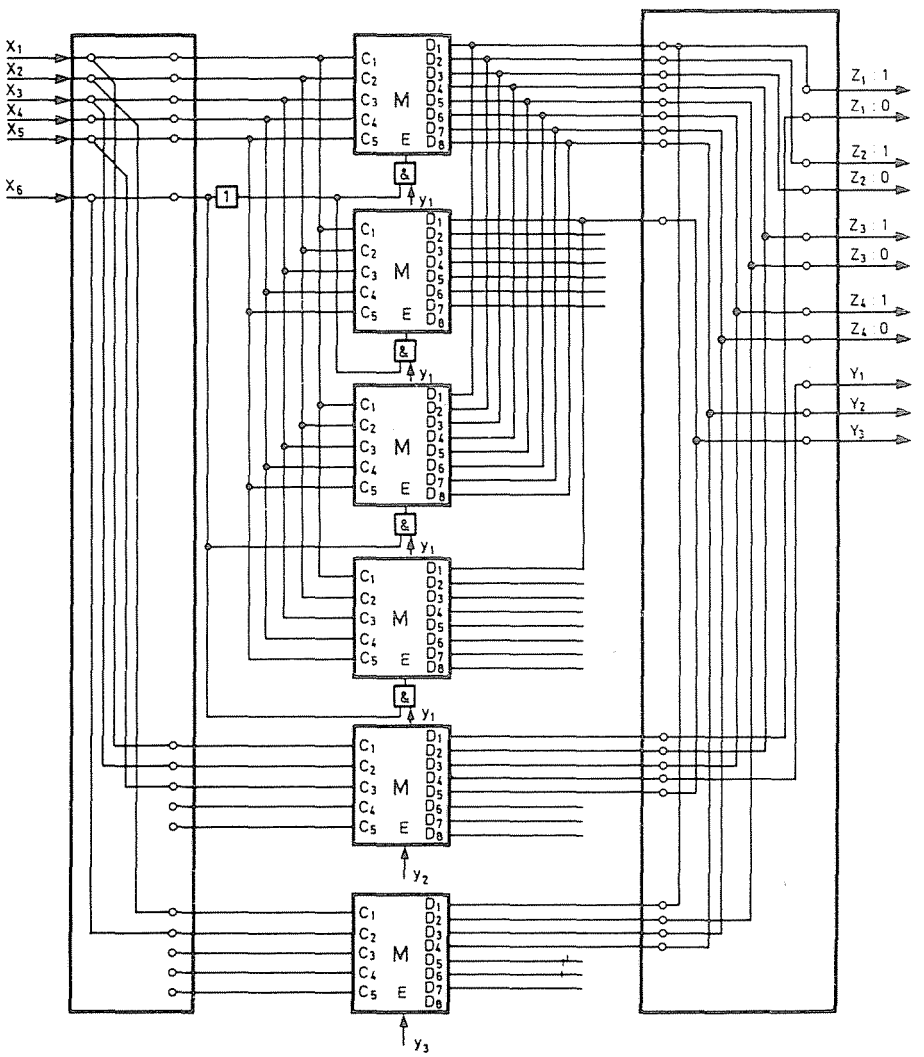

Fig. 6

which cannot be valid, because the minimal value of $r-n$ is 1 in this case. However, the design methods [7], [12], [13] reduce the number of elements of the sets Z/y_k and Y/y_k beside simplifying the identifying functions. Thus, the solution in Fig. 4 may be advantageous, even if the solution in Fig. 3 were completed with the input decoder for the secondary combinations.

Starting with the expressions of our example, the solution in Fig. 4 yields the realisation shown in Fig. 6. It has been assumed that memory units with 5 outputs and 8 outputs are available. In this case, the number of the elements in the sets X/F^1 and $Z/y_1 \cup Y/y_1$ require the usual network of four memory units to realize the storage field belonging to the state y_1. In spite of this, the solution with separate storage fields, using altogether six memory units, is more advantageous than the solution requiring $2^{6+3-5}=16$ memory units according to Fig. 3. Even if the solution in Fig. 3 were completed with the secondary decoder, it would require $2^{6+\lfloor 2\log 3 \rfloor - 5}=8$ memory units.

Conclusions

If the microprogrammed control unit is derived from the synchronous phase register structure according to Fig. 2, then the microprogram storage can be realized as shown in Fig. 4 based on the Boolean expressions provided by the hardware design procedures. By this means, the exponential growing of the number of the necessary memory units can be reduced to a great extent compared with the usual solution shown in Fig. 3. Thus, the minimization of the identifying functions and the state reduction produce an essential effect on the microprogrammed structure of the control unit.

The processor derived from the synchronous phase register structure and the state definition may yield a different bit interpretation of the microinstructions in each storage field enabled by different states. These different bitmaps make it possible to apply memory units with less address input and data outputs compared with the usual solutions.

References

1. CLARE, C. R.: Designing Logic Systems Using State Machines. McGraw-Hill Book Company, 1973.
2. WENDT, S.: Entwurf komplexer Schaltwerke. Springer Verlag, 1974.
3. GRASS, W.: Steuerwerke (Entwurf von Schaltwerken mit Festwertspeichern). Springer-Verlag, 1978.
4. KALMÁR, P.: A Phase-State Reduction and Assignment Method Based on the Flow Chart of Control Units. Period. Polytech. Electr. Eng. *20*, 365—376, (1976).
5. TERPLÁN, S.: A Design Method of Asynchronous Sequential Circuits Based on Flow Diagram. MTA SZTAKI Reports 137/1982.

6. ARATÓ, P.—TERPLÁN, S.—KALMÁR, P.—GRANTNER, J.: Methoden zum Entwurf logischer Schaltungen aus gegebene Ablaufplänen. Zeitschrift für elektrische Informations- und Energietechnik 7, 426—436. (1977).
7. KALMÁR, P.: Logic Design of Control Units; "Cand. of Sc." degree* dissertation Budapest, 1978.
8. TERPLÁN, S.: Design of Asynchronous Logic Network Specified by Flow-Chart; "Cand. of Sc." degree* dissertation, Budapest, 1981.
9. ARATÓ, P.: Design of Logic Systems; University textbook* Tankönyvkiadó, Budapest, 1985.
10. DERVISOGLU: A Hard programmable Control Unit Design Using VLSI Technology. IEEE Transactions on Computers, pp. 800—810. October, 1981.
11. DAVID, R.: Modular Design of Asynchronous Circuits Defined by Graphs. IEEE Transactions on Computers, pp. 727—738 August, 1977.
12. BÁNYAI, E.: A New Method for the Design of Control Units in Synchronous Phase Register Structure. Scientific Student Seminar Report* TU Budapest, Faculty for Electrical Engineering. 1986.
13. ARATÓ, P.: Control Unit Design Based on Prescribed Input and Output Changes. "Dr of Sc" degree* dissertation Budapest, 1984.
14. BALOGH, B.: Synthesis of Digital Control Units Based on Flow Chart "Cand. of Sc" degree* dissertation Budapest, 1986.

Prof. Dr Péter ARATÓ H-1521 Budapest

* In Hungarian