A PHASE-STATE REDUCTION AND ASSIGNMENT METHOD BASED ON THE FLOW CHART FOR THE LOGICAL DESIGN OF CONTROL UNITS

By

P. KALMÁR

Department of Process Control, Technical University, Budapest

Received July 25, 1976

Presented by Prof. Dr. A. FRIGYES

1. Introduction

The mass production of digital equipment requires the automatization of the development and design processes. In this paper a logical design method is described for the computer-aided realization of the control units in logical systems. The control function is supposed to be concentrated in one or more blocks in the system. The design method described is based on the flow chart of the control function as the initial characterization of the problem to be solved. The concentration of the control function into control units is a result of functional decomposition which is easy to build up, taking into consideration the verbal description of the system to be designed. The system having been decomposed into control units and into blocks of internal tasks, the flow chart of the control must describe the internal control signal changes in addition to the external ones.

The design method described in the paper produces the realization of the control blocks in fix structures determined in advance.

2. Structure of the Control Units

The proposed structure is based on the requirements of the computer-aided method and on the phase-register method described in the reference [1]. The proposed structure is shown in Figure 1 and the denotations are as follows:

The model is denoted by

$$VE = [X, Z, F, V, f_z, f_f, f_v, f_c] ,$$

where $X : x^1, x^2 \dots x^a$ is the set of the input combinations,

$Z : z^1, z^2 \dots z^b$ the set of the output combinations,

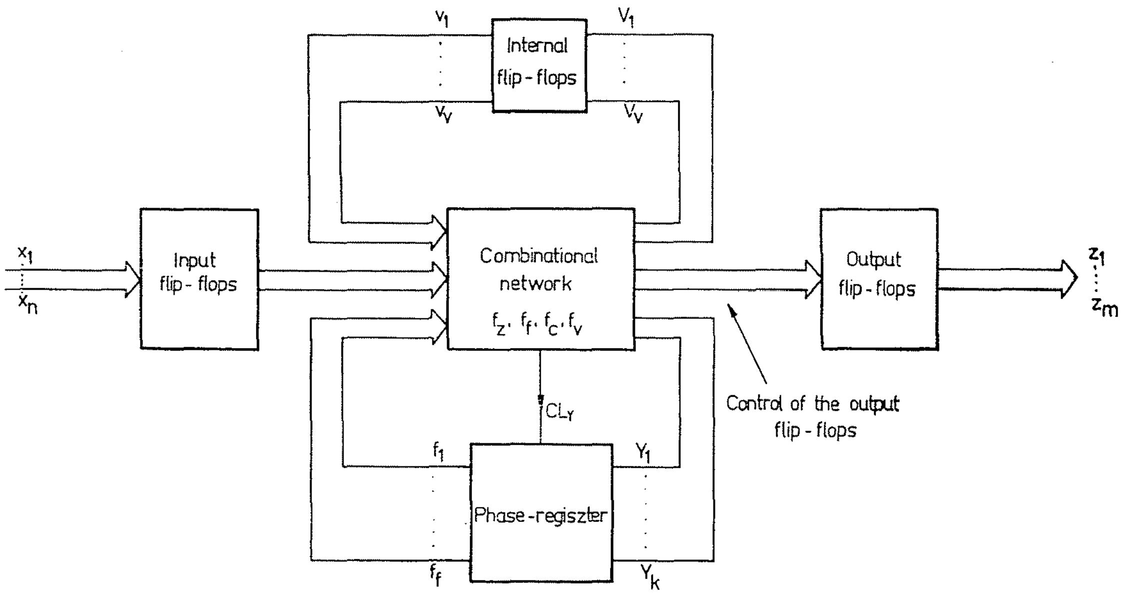


Fig. 1

$V: v^1, v^2 \ldots v^c$ the set of the internal variables,

$f_f: XxFxV \rightarrow F$ the mapping producing the next phase-state,

$f_v: XxFxV \rightarrow V$ the mapping producing the next internal variable states

$f_z: XxFxV \rightarrow Z$ the mapping producing the output combination

$f_c: XxFxV \rightarrow CL_y$ the mapping producing the clock-signal for the phase-register.

One of the possibilities of realizing the model in Figure 1 is the use of two clock signals denoted by CL1 and CL2, respectively. The input flip-flops are triggered by clocks CL1, and CL2 the internal and output flip-flops. The mapping f_c is represented by a clock-inhibitor block, which is convenient for the realization of the phase register. This solution ensures that the clock signal is transmitted to the phase register only when a phase-state change is desirable. (Figure 2.)

3. Flow Chart of the Control Unit

The flow chart of the control function can be constructed on the basis of the flow chart and the block structure of the system to be designed. The flow chart consists of instruction-sequences. The following flow chart instructions can be defined:

1. Set or reset the output flip-flop z_m or the internal flip-flop v_i.

2. Conditional jump depending on the value of the input variable x_j or of the internal variable v_i.

3. Wait or continue depending on the value of the input variable x_i.
4. Wait or continue depending on the time (Delay instruction).

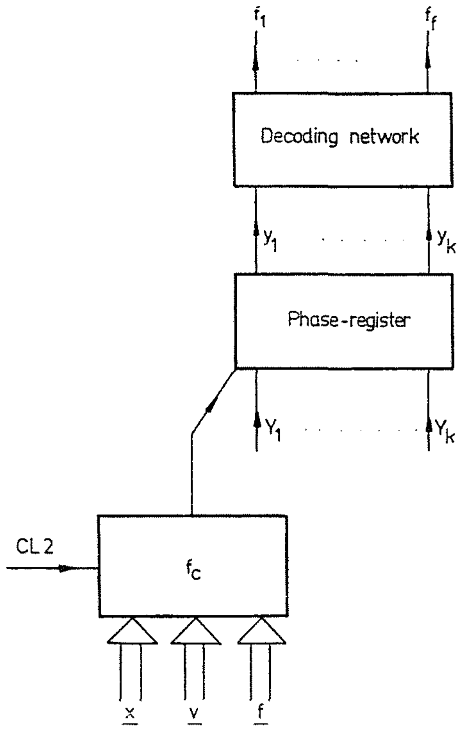


Fig. 2

4. Procedure for Reducing the Phase-States

A phase-state reducing procedure can be built up in the following steps:

1. Forming the directed graph of the given flow-chart.
2. Appointing the possible through-pass ways belonging to every node of the directed graph.
3. Constructing the flow charts corresponding to the through-pass ways; appointing the possible ranges of the phase-state transitions.
4. Calculating the solution with the minimum number of phase-states.
5. Constructing the transition table.

For the easy handling of the flow chart, a directed graph representation appears to be useful. The directed graph of a flow-chart is a graph, the nodes of which represent the conditional jump instruction and the instructions following the junction points of two or more flow-chart branches. A directed branch leads from the node q_k to the node q_l, if there exists a branch in the flow chart from the instruction corresponding to q_k to the instruction corresponding to q_l.

The properties of the directed graph-nodes of a flow chart are summarized in Figure 3.

It can be seen that one or two branches go further from every node of the directed graphs.

The through-pass way corresponding to the node q_i of a directed graph is a sequence of branches denoted by

$$\hat{h}_i \dots . \hat{h}_m, \hat{h}_n \dots . \hat{h}_n$$

where $\hat{h}$ may represent h or $\bar{h}$.

This sequence starts with the branch $\hat{h}_i$ and ends with a branch leading to q_i. The branch h_m leads to the node q_m. (The through-pass way may consist of only one branch.)

Suppose that the initial state of a flow chart corresponds to the node q_i of the directed graph. For appointing the phase-states, the through-pass ways belonging to q_i are to be determined. All possible through-pass ways must be taken into consideration, if the directed graph does not contain any circles. If the directed graph does contain a circle, then only the through-pass ways will be important that have no redundant parts. This means that among the through-pass ways there must not be any similar partial ways next to each other. For example:

$$h_i \dots . \underbrace{h_m, h_n, h_0}, \underbrace{h_m, h_n, h_0} \dots . h_n$$

The through-pass ways can be determined by step-by-step reduction of the directed graph. The rules of reducing the nodes are shown in Figure 4. (During reduction the properties of the nodes may change as compared with Figure 3.)

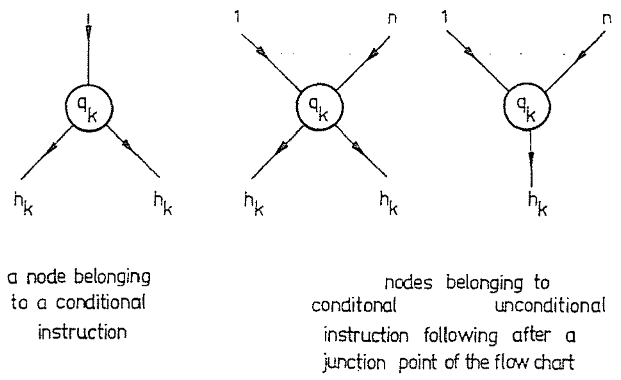

1
n
1
n
q_k
q_k
q_k
h_k
h_k
h_k
h_k
h_k
a node belonging to a conditional instruction
nodes belonging to
conditonal
unconditional
instruction following after a junction point of the flow chart

Fig. 3

Knowing the through-pass ways of the directed graph, partial flow charts can be derived. *A partial flow chart* corresponds to one of the irredundant through-pass ways from an initial state and back to it. The partial flow charts have no branching because the conditional jump instructions (except the wait instruction) are represented only by one of their branches.

The partial flow charts can be constructed in two steps:

— Deriving the flow chart parts corresponding to the branches of the directed graph

— Assembling the flow chart parts corresponding to the branch-sequences of the through-pass ways.

After having constructed the partial flow charts, separating rules are to be applied on them to appoint the ranges of the phase-state transitions. The separating rules detailed below suppose the realization described in Chapter 1 and one of their aims is to ensure a systematic design method.

4.1. *Separating Rules*

Rule 1. The delay instruction can be realized by a phase-state transition (or transitions) placed before the instruction. In the case of synchronous realization there is a fixed minimum delay between the execution times of the instructions of two successive phase-states. The fixed minimum delay time depends only on the frequency of the clock signal.

Rule 2. Two points of a partial flow chart must be separated by a phase-state transition, if

— they are defined by a conditional jump instruction of an internal variable v_i and by a wait instruction of an input variable x_j, and

— the instruction sequence

$$S_u = u_1, u_2 \ldots\ldots u_{n-2}, u_{n-1}, u_n$$

between them contains an instruction changing the variable v_i, and

— there exists no such partial flow chart, in which

$$\ldots\ldots\ldots\ldots u_{n-2}, u_{n-1}, u_n \subseteq S_u$$

would hold for the instruction sequence between the conditional instructions of v_i and x_j.

This rule is illustrated in Figure 5. Separating the phase-states in this way, any erroneous operation of the system can be prevented. Suppose that the flow chart part shown in Figure 5 is realized in one-phase-state, and $v_i = 1$ and $x_j = 0$. The clock CL2 triggers $z_k = 1$ and $v_j = 0$. If x_j is not changed by the next pulse of CL1, then no phase-state transition occurs.

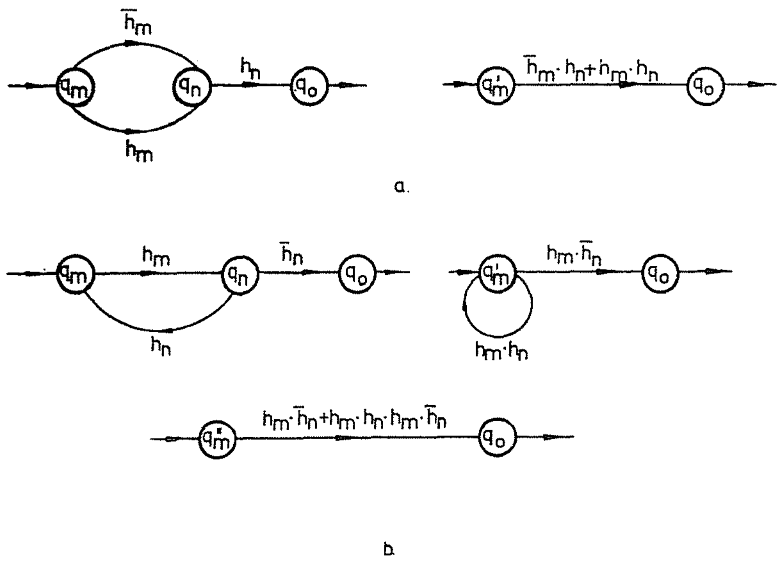


Fig. 4

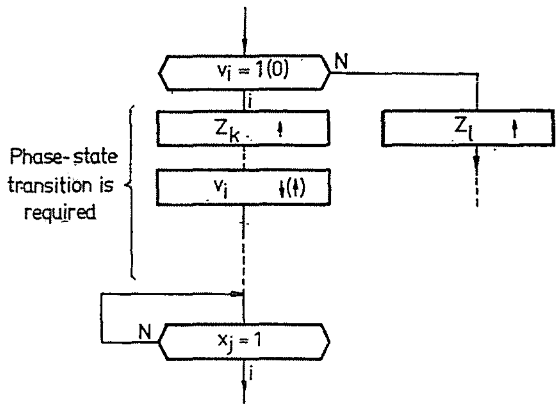
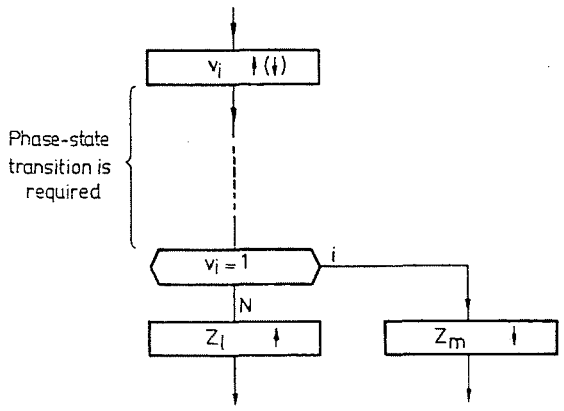


Fig. 5

The next CL2 would be $z_l = 1$. Obviously, this operation does not correspond to the flow chart.

This error can be avoided by establishing a phase-state transition between the conditional jump instructions corresponding to the variables v_i and x_j. Suppose that the initial conditions are the same as have been before: $x_j = 0$; $v_i = 1$. The phase-state transition is caused by the pulse CL2 arriving in he first phase-state. In the new phase-state — according to the flow chart — no signal-change dependent on v_i can occur, thus z_l will remain 0.

Rule 3. Two points of a partial flow chart must be separated by a phase-state transition if



Fib. 6

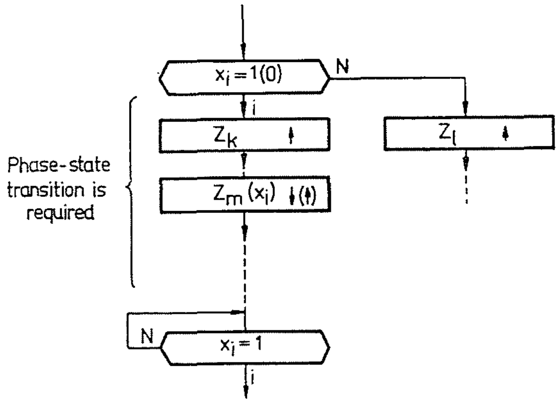


Fig. 7

— they are defined by a conditional jump instruction of an input variable x_i and by a wait instruction of another input variable x_j and

— the instruction sequence S_u changes an output variable z_m, which causes x_i to change in an asynchronous way[1], and

— there exists no such partial flow chart, in which

$$\ldots\ldots\ldots u_{n-2},\ u_{n-1},\ u_n \subseteq S_u$$

would hold for the instruction sequence between the conditional instructions of x_i and x_j.

This rule is illustrated by the flow chart shown in Figure 6 and the necessity of the phase-state separation can be explained in the same way as in Rule 2.

[1] The input variable x_i can change immediately after z_m has changed, and it is under the influence of the line delays only. This situation will be denoted by $z_m(x_i)$.

Rule 4. Two points of a partial flow chart must be separated by a phase-state transition, if they are defined by a set or reset instruction of an internal variable v_i and by a conditional instruction depending on the internal variable v_i (Figure 7).

For the sake of the correct operation, the value of the internal variable v_i must be ensured to be correct (as it is defined by the flow chart) while being scanned.

Rule 5. Two points of a partial flow chart must be separated by a phase-state transition if they are defined by a set and a reset (or a reset and a set) instruction of an output variable z_l or of an internal variable v_i. It is not necessary to separate the instructions $z_l \uparrow (\downarrow)$ and $z_l \downarrow (\uparrow)$ if in the instruction sequence between them contains a wait instruction of x_j, such as

$$z_l\,(x_j) \uparrow (\downarrow),$$

which means that the execution of the set and reset instruction depends on the value of x_j.

The separation is necessary for constructing the Boolean functions corresponding to the mapping f_f. If $z_l(x_j) \uparrow (\downarrow)$ does not hold, then in the case of a realization with $J-K$ flip flops

$J_{zl} \;= F_i \cdot f_j$
$K_{zl} \;= F_i \cdot f_k$ will be satisfied, where
$F_i \;=$ denotes the variable representing the i-th phase-state
$f_j \;=$ and f_k are terms determined by the partial flow chart.

It can be seen that, following from the properties of the partial flow chart, $f_j \neq f_k$.

The consequence of this is that the clock pulse CL2 will change the value of z_l and the network will get into the next phase-state.

Rule 6. Two points of a partial flow chart must be separated by a phase-state transition if they are defined by wait instructions corresponding to different values of an input variable x_i. If this separation turns out to be the only one in the partial flow chart, then it need not be done.

The Boolean functions of f_t are easily constructed with the separation, because it cannot occur that both $x_j = 1$ and $x_j = 0$ would be conditions of getting further along the flow chart in the same phase-state.

Rule 7. A phase-state transition is to be established at the beginning of the flow chart in order to ensure the transition into the initial phase-state.

Before constructing the partial flow charts, each instruction of the flow chart must be provided with an index, in order to identify the exact places of the instruction sequences resulting from the application of the above rules.

4.2. *The Reducing Procedure*

Applying the separation rules, each partial flow chart can be divided into instruction sequences. Let $N_1, N_2 \ldots\ldots$ denote the set of the instruction sequences belonging to all of the partial flow charts separated according to the rules described above. (The instruction sequences may overlap each other.) The minimal number of the phase-states required for proper operation can be obtained if an instruction set is found which contains the fewest possible instructions and these instructions distinguish the instruction sequences $N_1, N_2 \ldots\ldots N_k$ at least once. The calculation of this minimal instruction set can be made by the covering method after reduction[1] of the set $N_1, N_2 \ldots\ldots N_k$. The instructions of the reduced set are considered to be variables of the covering function written in sum-of-products form. Each product represents an instruction set as a solution of the problem. The minimal solutions can be obtained from the products consisting of the fewest variables. Each of the minimal solutions is suitable for the decision about the exact places of the phase-state transitions, and a transition table can be drawn up, which is a useful tool for the phase-state assignment methods.

5. A Shift-Register Assignment of the Phase-States

The shift-register is one of the possible logical networks for the realization of the phase-states in the structure shown in Figure 1. If the phase-state transitions allow a shift-register realization, then the state assignment will have to meet the following requirements:

— For the number of the state variables $p = [\log^2 f]$ must hold, where f denotes the number of the phase-states,

— A one-to-one correspondence must be ensured,

— In the case of binary shift-registers, the flip-flops must have a minimal number of control inputs.

The design method developed for the state assignment selects the minimal partial solution from the set of the shift-register partitions of minimal length. The method summarized in this paper is based on NICHOL's algorythm [2], but a new procedure is proposed requiring less calculation to determine the shift-register partitions.

[1] The basis of the reduction is that N_i can be neglected if $N_j \subset N_i$.

Based on the transition table, the decomposition tree [3]

$$P_0, \delta P_0, \delta^2 P_0, \ldots\ldots \delta^n P_0 \;\Big|\; \delta^n P_0 = \delta^{n+1} P_0 \quad \text{or} \quad \delta^{n-1} P_0 = \{I\}$$

can be constructed, where

P_0 denotes the $\{0\}$ trivial partition;

$\{I\}$: the trivial partition;

the meaning of $\delta\, P$ is given by the following definition:

If $P = \{b_1; b_2; \ldots . b_r\}$ is an internal state partition and δb is the union of all successors of the states of b, then δP is that partition obtained from $\{\delta b_1; \delta b_2 \ldots\ldots \delta b_r\}$ by identifying any blocks δb_i and δb_j that are chain-connected [3]

δ^{iP} means the i-times application of the operator δ on the partition P.

The shift-register partitions of different length can be calculated by means of the composition tree:

$$P_m^{i+1} = \prod_{j=0}^{i} \delta^{-j} (\delta^i P_0)_{bm}$$

where

$(\delta^i P_0)_b$ is a binary partition formed from $(\delta^i P_0)$;

$i = 1,2 \ldots\ldots n$ denotes the level of the composition tree [3];

m is the index of the two-block partitions obtainable from the partitions on a given level of the composition tree;

$\#\ (\delta^i P_0)$ denotes the number of the blocks of the partition $\delta^i P_0$;

the meaning of $\delta^{-1} P$ is given by the following definition:

If $P = \{b_1; b_2; \ldots . b_r\}$ is an internal state partition and $\delta^{-1} P$ is the union of all predecessors of the states of b, then $\delta^{-1} P$ is that partition obtained from $\{\delta^{-1} b, \ \delta^{-1} b_2 \ldots . \delta^{-1} b_r\}$ by identifying any blocks $\delta^{-1} b_i$ and $\delta^{-1} b_j$ that are row-connected [3]

$\delta^{-i} P$ means the i-times applications of the operator δ^{-1} on the partition P.

The set of the shift-register partitions of minimal length [2] $(l \geq 2)$ can be selected from the set of the partitions calculated in the above way. The basis of selection is that P_i^l will be neglectable if $P_j < P_i$ [2].

Suppose that the partition p_m^{i+1} is one of the partitions of a minimal solution [2]. The phase-state assignment can be derived from the binary partitions

$$(\delta^i P_0)_{bm}, \ \delta^{-1} (\delta^i P_0)_{bm}, \ldots . \ \delta^{-i} (\delta^i P_0)_{bm} .$$

The following requirements must be satisfied in the assignment:

— The values of the state variable y_j must be similar in the states belonging to the same block of the partition $\delta^{-j} (\delta^i P_0)_{bm}$.

— If the block S_j^l of the partition $\delta^{-j}(\delta^i P_0)_{bm}$ and the block S_{j+1}^l of the partition $\delta^{-(j+1)}(\delta^i P_0)_{bm}$ lead into each other, then the values of the state variable y_j must be the same in the states belonging to S_j^l as the values of the state variable y_{j+1} in the states belonging to S_{j+1}^l.

The result of the phase-state assignment procedure can be summarized in the encoded transition table.

6. Determination of the Control Functions

The control functions realizing the mappings f_z, f_v, f_f and f_c can be constructed systematically with the use of the partial flow charts as follows:

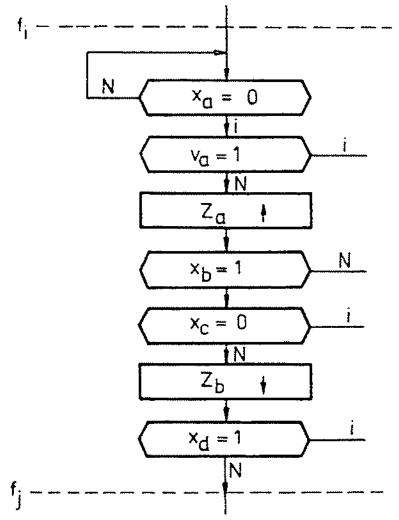
a) One of the terms of the control functions belonging to a phase-state is easily constructed from the partial flow chart provided with the phase-transition points. The variables of a term (belonging to the control functions of an output variable z_m or an internal variable v_i) are:

— the input and internal variables occurring in conditional jump and wait instructions between the beginning of the phase-state and the instruction, the control term of which is being constructed (Figure 8),

— the variables representing the phase-state in which the term is being constructed.

The terms of the same control inputs occurring in several phase-states must be summarized logically.

b) The control function of the flip-flops realizing the phase-states are derivable from the functions f_{i-j} and from the transition table.



$$JZ_a = f_i \cdot \bar{x}_a \cdot \bar{v}_a$$

$$KZ_b = f_i \cdot \bar{x}_a \cdot \bar{v}_a \cdot x_b \cdot x_c$$

$$f_{i-j} = f_i \cdot \bar{x}_a \cdot \bar{v}_a \cdot x_b \cdot x_c \cdot \bar{x}_d$$

Fig. 8

— The functions f_{i-j} represent the conditions of the transition from the phase-state f_i to the phase-state f_j. The functions f_{i-j} can be constructed in the same way as described in *a)*. In this case the determination of a term begins at the *i*-th and ends the *j*-th phase-state (Figure 8).

c) The clock signal of the phase-register is

$$CL_y = f_c \cdot CL$$

The enable-control function f_c is generated by the logical sum of all functions f_{i-j}:

$$f_c = \sum_{i,j=1}^{f} f_{i-j}\Big|_{i \neq j}$$

where

f is the number of the phase-states,

$f_{i-j} = 0$, if there is no transition from the *i*-th to the *j*-th phase-state.

The control functions constructed by the method described above have not necessarily the simplest form; they are two-level Boolean functions and one of the well-known methods can be applied for simplification.

The design procedures outlined in this paper have been realized by a program package for the computer ODRA 1204.

Summary

The paper outlines a new computer-aided logical design method for the realization of the control functions in arbitrary digital equipment. The method is based on the flow chart of the control unit presumed to be concentrated and to be described by flow chart. A systematic procedure is presented for appointing the phase-state transitions and for reducing the phase-states in a phase-register realization. The design procedures consider the flow chart as the description of the control function to be realized and suppose the control block to have a fixed structure. After phase-state reduction the method is suitable for developing several phase-state assignment procedures. As an example for the assignment a modification of NICHOL's algorythm for shift-register realizations is proposed.

References

1. ARATÓ, P.—KALMÁR, P.—KONDOROSI, K.: Számítógépek és perifériák (Computers and peripherals) Tankönyvkiadó, Budapest, 1973, J 5—1032 (A students' textbook in Hungarian).
2. NICHOLS, A. J.: Minimal Shift-Register Realizations of Sequentional Machines, IEEE Transactions on Electronic Computers, Volume EC—14, Oct. 1965, pp. 688—700.
3. DAVIS, W. A.: On Shift-Register Realisations for Sequential Machines, IEEE Conf. Rec. on Switching Circuit Theory and Logical Design 1965, pp. 71—83.
4. KALMÁR, P.: A Phase-state Reduction and Assignment Method Based on the Flow Chart, A doctoral dissertation in Hungarian, Budapest, 1975.

Dr. Péter KALMÁR, H-1521 Budapest