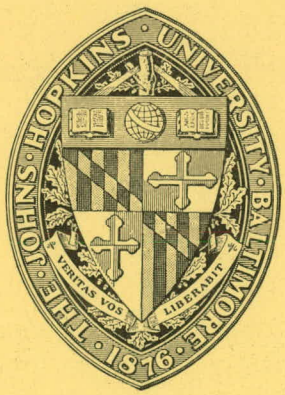
HOPKINS COMPUTER RESEARCH REPORTS
REPORT #30
DECEMBER 1973

A SURVEY OF TECHNIQUES TO REDUCE/MINIMIZE THE CONTROL PART/ROM OF A MICROPROGRAMMED DIGITAL COMPUTER

BY

TILAK AGERWALA

RESEARCH PROGRAM IN COMPUTER SYSTEMS ARCHITECTURE
COMPUTER SCIENCE PROGRAM
THE JOHNS HOPKINS UNIVERSITY
BALTIMORE, MARYLAND



DISCLAIMER

A SURVEY OF TECHNIQUES TO REDUCE/MINIMIZE THE CONTROL PART/ROM OF A MICROPROGRAMMED DIGITAL COMPUTER

BY

Tilak Agerwala

*This work was supported, in part, by the U. S. Atomic Energy Commission under contract AT (11-1 3288)

A SURVEY OF TECHNIQUES TO REDUCE/MINIMIZE THE CONTROL PART/ROM OF A MICRO-PROGRAMMED DIGITAL COMPUTER

I. Introduction

In this report a survey of the research to date in microprogram minimization is presented. We discuss the works of Glushkov, Casaglia, et. al., Flynn and Rosin, Mischenko, Schwartz, Grasselli and Montanari and Das et. al. The techniques are classified into three broad categories: the Glushkov Approach, the Ad Hoc or Engineering Approach and the Schwartz Approach. The authors' views are summarized in the conclusion but the survey presents more than sufficient detail for the reader to draw his own.

II. The Glushkov Approach

Glushkov's main aim is to construct an abstract model of an electronic computer based on an enlarged concept of an automaton [6,7]. This abstract model enables him to formalize a whole series of problems important from the point of view of logical design. One of these problems is the minimization of the control unit.

Glushkov views a digital system as a composition of two automata, called respectively, the operational and control automata. (These are also called "operation part" (OP) and "control part" (CP) in the literature [1].) The OP is generally a finite Moore automaton[1]. All the combinatorial and and sequential networks needed to perform the logical and arithmetic functions of the computer can be included in the OP. The number of states of the OP is thus enormous and Glushkov feels that it may be better to replace the finite

---

1. A finite state automaton M is a quintuple $M = (I,O,S,\delta,\lambda)$ where I,O and S are finite, nonempty sets of input states respectively; δ: $I \times S \to S$ is the state transition function, λ is the output function such that $\lambda : I \times S \to O$ for Mealy machines, $\lambda : S \to O$ for Moore machines.

OP by an infinite, multiregister automaton of the type described in [4].

The output signals of the OP are the strings of values of logical conditions $X_1, X_2, \ldots, X_m$ formed during certain elementary operations. Each logical condition X_i is associated with a subset U_i of the set of states of the OP. $X_i = 1$ if and only if the current state belongs to U_i. Thus the output depends only on the current state and not on the signal at the input. Let the set of output signals be $\{y_i\}$. The input signals $a_1, a_2, \ldots a_n$ of the OP are identified with certain transformations of the set of states of this automaton. Thus each input signal contains the commands for executing the elementary operations (micro operations).

The CP which works together with the OP is a finite Moore or Mealy automaton. The input signals to the CP coincide with the output signals of the OP and the output signals of the CP with the input signals of the OP. The number of states of the CP will usually be relatively small.

In general, there is a third part to this dual system, i.e. Memory and I/O, (MIO). Information passes from MIO to the OP and the results of micro operations pass from the OP to MIO. MIO also send information to the CP regarding their own functioning and also about which instruction is to be executed next. The CP communicates with MIO and initiates their operations from time to time. The general system is shown in figure 1.

However, by assuming that memory and I/O are buffered, the buffer registers can be considered to be part of the OP and the reduced scheme shown in figure 2 is obtained.

Let the CP be a Mealy automaton. The problem then is to reduce the number of states of this automaton. For a completely specified finite state automaton the first step would be to try and minimize the number of states by partitioning them into equivalence classes. (For details the reader is referred to [10]). Since the minimal partition is unique, there is a unique minimal solution. The CP may be incompletely specified to start off with because for some combinations of

states and inputs the output values may not be critical and are left unspecified. The following point is more important: i.e. the CP and OP are acting in conjunction. Thus, even if it is possible to select any state of the OP as the initial state, it is not, generally speaking, possible for all a priori conceivable sequences of input signals to arrive at the input of the CP. Because of this it may be possible to convert the original completely specified minimal machine to an incompletely specified one. The incompletely specified machin can then be minimized by known techniques [10]. We will illustrate Glushkov's technique [8] by means of an example. Let the outputs of the OP be $\{y_1, y_2\}$ and inputs $\{a_1, a_2\}$. Let the OP be described by the table in figure 3.

The CP thus has inputs $\{y_1, y_2\}$ and outputs $\{a_1, a_2\}$. Let it be described by the table in figure 3.

Assume also that initially the CP is in state 1 and the only input that can arrive is y_1. The OP is first examined. The output signals of the OP, i.e. y_1 and y_2 are identified with the set of states marked by these signals. Therefore $y_1 = \{p_1, p_3\}$ and $y_2 = \{p_2, p_4\}$. Let S_{a_j} denote the state entered when the current input is a_j and current state is S. Let $y_i\ a_j$ denote the union of all sets y_k which contain states of the form S_{a_j} where $S \in y_i$. Thus $y_1 a_1 = y_2$, $y_2\ a_1 = y_2$, $y_1\ a_2 = (y_1, y_2)$ and $y_2\ a_2 = y_1$. For any set of output signals $M = \{y_{i_1}, \ldots, y_{i_k}\}$ let M_{a_j} denote the union of all sets $y_{i_1}\ a_j, \ldots y_{i_k}\ a_j$. Thus, $y_1\ a_1 = y_2$, $y_1\ a_2 = (y_1, y_2)$, $y_2\ a_1 = y_2$, $y_2\ a_2 = y_1$, $(y_1, y_2)\ a_1 = (y_1, y_2)$, and $(y_1, y_2)\ a_2 = (y_1, y_2)$. Define $M(y_i, a_j)$ to be $y_1\ a_j$ if $y_i \in M$ and ϕ otherwise. Therefore, $y_1(y_1, a_1) = y_2$, $y_1(y_1, a_2) = (y_1, y_2) = y_1, y_2$, $y_2(y_2, a_1) = y_2$, $y_2(y_2, a_2) = y_1$, $(y_1, y_2)(y_1, a_1) = (y_1, y_2)(y_2, a_1) = y_2$, $(y_1, y_2)(y_1, a_2) = (y_1, y_2)$, $(y_1, y_2)(y_2, a_2) = y_1$. All other combinations have a value ϕ. The produce MN where N is any set of pairs (y_i, a_j) is defined as the union of the products Mq for all $q \in N$.

The CP is now examined. For any pair (b_s, b_r) of this automaton define B_{sr} to be the set of all pairs (y_i, a_j) such that the effect of the input y_i is to make the CP pass from b_s to b_r. Thus, $B_{11} = (y_2, a_1)$, $B_{12} = (y_1, a_2)$, $B_{13} = \phi$, $B_{21} = \phi$, $B_{22} = (y_1, a_1)$, $B_{23} = (y_2, a_2)$, $B_{31} = (y_2, a_2)$, $B_{32} = \phi$, $B_{33} = (y_1, a_2)$. Define $M_k^\circ = \phi$ for all $k \neq 1$ and M_1° to be the set of signals that can reach the CP initially. Thus $M_1^\circ = y_1$. Define next, the following recurrence relations:

$$M_k^{j+1} = M_k^j \cup \bigcup_{i=1}^{p} M_k^j \; B_{i_k} \qquad (k = 1, 2, \ldots p)$$

where p is the number of states in the CP. The sequence $\{M_k^i\}$, $j = o, 1, \ldots$ will stabilize at some stage j (i.e. $M_k^{j+1} = M_k^j$ for all k). Let M_k, $k = 1,2,\ldots p$ denote the stabilized values. For simultaneous operation of the OP and CP, when CP is in state k the only signals that can possibly arrive at its input are those belonging to M_k. In the particular example under discussion, the sequences obtained are as follows:

$$M_1^1 = (y_1), \quad M_2^1 = (y_2), \quad M_3^1 = \phi$$

$$M_1^2 = (y_1), \quad M_2^2 = (y_2), \quad M_3^2 = (y_1)$$

$$M_1^3 = (y_1), \quad M_2^3 = (y_2), \quad M_3^3 = (y_1, y_2)$$

$$M_1^4 = (y_1), \quad M_2^4 = (y_2), \quad M_3^4 = (y_1, y_2)$$

We thus conclude that in state 1, y_2 can never arrive at the input of the CP and in state 2, y_1 cannot arrive. The corresponding entries can now be deleted from the table in figure 4 to yield the incompletely specified CP of figure 5.

States 1 and 2 are now consistent and can be combined to yield a single state. Figure 6 represents the final reduced CP.

Comments:

The example presented was a very simple one and even then the procedure

to obtain the minimal CP was quite laborious. In a practical situation one does not deal with machines with 2 or 3 states. Using the automata theory approach Glushkov would be out of the question for any modern digital system where the machines have billions of states. Thus even though the approach taken by Glushkov is significant from a theoretical point of view, the techniques he suggests are infeasible for all practical purposes.

It must also be pointed out that even if an incompletely specified CP is obtained a reduced machine need not be minimal. Lower and upper bounds on the number of states in the reduced machine can easily be found and merger graphs/tables and compatibility graphs can be used to obtain a systematic procedure [3]. However, there is no simple precise way to obtain a minimal machine from an incompletely specified one and a cartain amount of "trial and error" is unavoidable.

III. The Ad Hoc or Engineering Approach

Here we present some of the minimization techniques proposed by Casaglia e.t., al. [11], Mischenko [11,12], Flynn and Rosin [4]. The contol part can be defined by means of a microprogram which is a sequence of microinstructions. As pointed out in [1], this microprogram can be written in one of two languges which we will discuss here.

Let O_1, O_2, ..., O_n be the list of all possible elementary operations of the OP of a system. These are called simple micro operations. A complex micro operation a_j is a set of O_i's such that all the O_i's can be executed simultaneously. Thus the a_j's are the possible outputs of the CP. Recall that the inputs y_i to the CP are logical conditions $\{x_1, \ldots, x_m\}$ formed as a result of the execution of certain micro operations in the OP (figure 2). An unconditional microinstruction is an expression

$$|h|\ a_j, k \qquad (1)$$

where $|h|$ is the label of the micro instruction being considered, k is the

label of the next microinstruction. Thus a_j defines the complex microinstruction to be executed by the OP and k the transfer to be executed by the CP. A conditional microinstruction can be represented by the following general statement

|h| if logical condition X_1, then a_{j1}, k_1 else

if logical condition X_2, then a_{j2}, k_2 else

.....

if logical condition X_p, then a_{jp}, k_p.

which takes the symbolic form:

$$|h|\ (X_1)\ a_{j1},\ k_1;\ \ (X_2)\ a_{j2},\ k_2;\ .....;\ (X_p)\ a_{j_p}k_p \qquad (2)$$

(X_r), a conditional expression, stands for an expression such as $(X_1, X_2, X_7 = 001, 110)$ which indicates that a subset of the signals X_i must equal a specified binary combination. The pair a_j, k is called a phrase and the expression $(X_r)\ a_j$, k a conditional phrase. A phrase structured (ps) language is defined as one where every microinstruction is formed by a one phrase microinstruction (1) or by a set of conditional phrases where (a) all the conditional expressions and phrases are different and (b) one and only one of the logical conditions is satisfied. A microinstruction structured (ms) language is diectly derived from a ps language with the additional condition: "all phrases in any microinstruction differ only in the transfers." A microinstruction in an ms language can thus be written as:

$$|h|\ a_1\ \ (X_1)\ k_1;\ (X_2)\ k_2;\ ....\ ;\ (X_p)\ kp.$$

Assume that there are two microprograms one written in the ps language and the other in the ms. These microprograms can be viewed as definitions of the control part and from each definition a particular CP can be realized. Figures 7 and 8 show the CP's corresponding to the ps and ms languages respectively.

Let us now discuss the possible ways of reducing the CP's.

(1) If in the ms microinstructions the maximum number of alternate transfers is reduced, this will reduce the size of the ROM word in figure 8.

(2) If the alternate transfers of every ms microinstruction are all constrained to be coded by addresses that are relative to a "base address" and this address is stored in the ROM memory, then the ROM word length can be considerably reduced.

(3) Given an ms microprogram let there be a set of microinstructions such that (a) all microinstructions in the set differ only in the component a_j and (b) logical conditions exist which can distinguish the different a_j's. For example, let the set be:

$|s_1|\ a_1\ (X_1)\ \ell;\ (X_2)\ s;\ (X_3)\ p$

$|s_2|\ a_2\ (X_1)\ \ell;\ (X_2)\ s;\ (X_3)\ p$

.....

$|s_n|\ a_n\ (X_1)\ \ell;\ (X_2)\ s;\ (X_3)\ p.$

The following reductions are possible:

(i) Instead of having n words in memory we have only one $\boxed{a \mid \ell, s, p}$. A small combinatorial network introduced at point B in Figure 8 can be used to provide the correct output.

(ii) The n potential transfer addresses $s_1, \ldots, s_n$ are reduced to one, i.e. s. Therefore the complexity of CN should decrease.

(4) If in a ps microporgram there is a microinstruction such that all its phrases differ only in the component a_j, then this microinstruction can be written as

$|h|\ [(X_1)\ a_1,\ (X_2)\ \ a_2,\ \ldots,\ (X_p)\ a_p],\ k.$

The p memory cells corresponding to this instruction can be reduced to 1, i.e. $\boxed{a \mid k}$ and a combinatiorial network can be used to generate the desired output.

(5) Let there be a set of ps microinstructions,

$|h_1|,\ |h_2|,\ \ldots,\ |h_m|,$

each of which has n conditional phrases. If when all occurrence of $h_1, h_2, \ldots, h_m$ are replaced by h, the m microinstructions

become identical then they can be replaced by a single ps microinstruction. Here again m potential transfer addresses have been replaced by one and mn memory cells by n.

(6) Consider the following m ps microinstructions

$|h_1|$ (X_1) a_1, k_1

$|h_2|$ (X_2) a_2, k_2

.....

$|h_m|$ (X_m) a_m, k_3

If one and only one of the conditions X_1, X_2, ... X_m can be satisfied at any time and if (a_1, k_1), (a_2, k_2), ..., (a_m, k_m) are all different then these can be combined into a single instruction

$|h|$ (X_1) a_1, k_1; (X_2) a_2, k_2; ..., (X_m) a_m, k_m.

Instructions with more than one conditional phrase can similarly be combined. Here we do not get a reduction in ROM but since the number of poetntial transfer points has been reduced, CN should become less complex. Note: In 3, 5, and 6 where a number of addresses are being replaced by one throughout the microprogram, care should be taken to establish that the entire program is still a valid ps or ms micrprogram as the case may be.

Finally, we would like to mention the very general scheme of "residual control" in dynamic microprogramming proposed by Flynn and Rosin [4]. The basic idea is as follows: much information specified in the microinstruction is static. The status remains unchanged during the execution of a numberof microinstructions. If this static information and specifications is filtered out of the microinstruction and placed in "set up" registers, the combination of a particular field of microinstruction with its corresponding set up register would completely define the control for resource. As Dr. Flynn has pointed out (private communication) this technique is closely related to the well known Huffman coding problem and its solution. In this latter problem one

is given an alphabet $\{a_1, a_2, \ldots, a_n\}$ for transmission with probability of occurrence of a_i equal to p_i. The problem is to encode the elements of the alphabet so that $\sum_{i=1}^{n} \ell_i \ p_i$ is minimized, where ℓ_i is the length of code assigned to a_i. Thus one is interested in minimizing the expected value of the length of transmitted code. The problem is solved, naturally, by assigning the codes in a manner so that the higher the probability of occurrence of an element the shorter the code assigned to it.

In residual control the problem is analogous. The fields in a microinstruction represent information to be transmitted. In a certain field in the microinstruction let the information represented by some of the bits (say n in number) have a high probability p of remaining static. These n bits take on a few definite values with high probabilities. Because certain configurations of these n bits have a high frequency of occurrence, the corresponding information should be minimally encoded. This is done by placing the information in set up registers and introducing k bits instead of n ($k \leq n$) which indicate how the information in the set up register changes. For this method to work, some appropriate fixed probability p_o has to be determined so that only if $p > p_o$ will the information be placed in a set up register.

In some of the techniques proposed by Casaglia et. al. the ROM is reduced at the cost of introducing special combinatorial networks. Thus there is a loss of flexibility and a change in the ROM necessitates a change in the special circuitry. In the residual control method, the loss of flexibility is minimal.

Comments:

Compared to the "Glushkov Approach" the "Engineering Approach" has obvious practical advantages. The techniques suggested in this section naturally do not guarantee a minimal CP. However, in practical use, these techniques should be quite effective. Also, complete minimization of an automaton as we find in the Glushkov approach is not necessarily realizable with minimal

apparatus cost. The techniques presented in this section are thus more down to earth, more what a CP designer would be interested in.

The Schwartz Approach;

Schwartz [13] is concerned with minimizing the bit dimension of ROM's employed in the control part of a microprogrammed digital computer and attempts to provide an algorithm to do so. The ROM is an array of storage elements consisting of W words of B bits each. Each word specifies one or more elementary operations of the control part which can be executed in parallel. The sequencing of ROM words is not of concern. It is therefore assumed that the words do not contain address fields and all B bits are used for specifying subcommands. Figure 9 gives an example of ROM specification.

There are several ways of coding the B bits to reduce size. One important consideration, flexibility, should be kept in mind: the control microprogram should be easily modifiable. There are two extreme prossibilities for encoding the ROM:

1) Each bit of W is used to encode one subcommand. Thus the number of bits in W is equal to the number of distinct subcommands. The latter will usually be very large compared to the actual number of subcommands in any word. Thus this method is extremely inefficient with regard to the bit dimension B. The advantage is maximum flexibility. Since no combinational circuit is required at the output of the ROM, the contents of the ROM can be arbitrarily changed.

2) The ROM words are minimally encoded, i.e. $B = \lceil \log W_u \rceil$ where W_u is the number of unique words in the ROM $W_u \leq W$. In this case all advantages of microprogramming are lost. The ROM is used only to sequence words in the microprograms since the ROM word address is already an encoding of the corresponding word. A large combinatorial network would be required at the output of the ROM. If this is to remain unchanged with changes in the

microprogram, then any such change would have to result in a word already in the system.

Schwartz takes a midway position. The bit dimension is partitioned into groups. Each group represents a number of subcommands, no two of which occur together in the same word. The essential feature however is that each group represents only one subcommand in any given word.

The problem now is to minimize B. Schwartz realizes this but is unable to solve the problem. He gives instead an alogrithm that partitions the subcommands into a minimal number of groups. The basic procedure is as follows: Find the word W_j with the largest number of subcommands S_j. Thus B must be partitioned into at least S_j groups. The remaining words are taken one at a time and the subcommands contained in each are assigned to groups. The following constraints have to be met at each stage: all subcommands in the same word must be placed in different groups. When words are encountered with previously assigned subcommands, it must be ascertained that no two of them have been assigned to the same group. Any new subcommand in such words must be assigned to groups not in prior use in the word. The algorithm is basically one of exhaustive evaluation. However, there is no guarantee that a solution of S_j groups exists. If the solution for S_j groups does not exist, the entire procedure is repeated for $S_j + 1$ groups.

As Grasselli and Montanari [9] point out, a minimimum group solution does not imply a minimum B solution. The minimum group solution of Schwartz for the ROM in Figure 9 is {a}, {b,g}, {c,j,k}, {d,i}. Number of groups = 5, B = 10 (Each group also contains the subcommand NO OP). The solution suggested by Grasselli and Montanari is: {a}, {b}, {c}, {d,g,j}, {e}, {f,i,k} {h}. Number of groups = 6, B = 9.

Thus one has to look beyond the minimum group solution. Schwartz now gives an upper bound on the number of groups that need be considered. He shows that if a minimum group solution has a bit dimension of size B_m and if ST is the total number of subcommands then the largest number of groups that could have a smaller bit dimension is C_{m+1} where Cm is given by

$$\log_2 (ST - C_m + 2) = B_m - C_m$$

Noting the inability of Schwartz to give an algorithm for minimizing B, the problem was reformulated by Grasselli and Montanari in the framework of switching theory. The main minimization problem was reduced to a set covering problem of the prime implicant type.

The ROM words are considered to be a set of subcommands:

$W_a = \{S_{a_1}, S_{a_2}, \ldots, S_{a_\alpha}\}$

$W_B = \{S_{b_1}, S_{b_2}, \ldots, S_{b_\beta}\}$

.....

$W_T = \{S_{t_1}, S_{t_2}, \ldots, S_{t_\zeta}\}$

A compatibility relation is defined among the subcommands. S_i and S_j are compatible. If

$$S_i \in W_H \Rightarrow S_j \notin W_H \quad \forall H$$

A compatibility class C_i of subcommands is a class whose members are pairwise compatible. (compatibility is not an equivalence relation) A maximal compatibility class is one to which no subcommand can be added without violating pairwise compatibility. Let the set of subcommands be A. Then a minimal solution is a set of compatibility classes: $\{C_{i_1}, C_{i_2}, \ldots, C_{i_h}\}$ such that $\bigcup_{n=1}^{h} C_{i_n} = A$ and $B = \sum_{n=1}^{h} \lceil \log(\#C_{i_n} + 1 \rceil$ is minimal. Here $\# C_{i_n}$ denotes the number of subcommands in C_{i_n}.

Grasselli and Montanari shows that the only classes that need be considered for a minimal solution are prime compatibility classes defined below:

1) C_i is nonmaximal and $\# C_i = 2^h - 1$. $(h = 1, 2, \ldots)$

2) C_i is maximal and $\# C_i \neq 2^k$ $(k = 1, 2, \ldots)$

The minimal cover is obtained by solving a covering table of the prime implicant type. The table has a column corresponding to every subcommand S_j and a row corresponding to every prime class. The cost of row i is the cost of the corresponding class C_i. The aim is to select rows so that the total cost is minimized and each column is covered.

The three basic reduction rules for solving a covering table are:

1) Row essentiality.

2) Row dominance.

3) Column dominance.

Rule 3 is never applicable because of the peculiar nature of the covering table and Rule 2 can be replaced by : 2') Any row i of cost W_i $(W_i > 1)$ in which the number of crosses is not greater than $2^{W_i - 1}$, can be erased from the table.

We will now present a well known technique of integer programming to solve this problem. Rule 1 and 2' can be very easily applied in any reduced cover table. Since the method to be presented involves solving reduced tables, it should be particularly suited for our problem.

Let there be m distinct subcommands and let $\{C_1, C_2, \ldots, C_n\}$ be the prime compatibility classes, and W_i the cost of C_i, $1 \leq i \leq n$. Let $\{aij\}_{nxm}$ denote the covering table where aij = 1 if the subcommand corresponding to column j belongs to class C_i. Let there be n variables $x_1, x_2, \ldots, x_n$. Then a solution is an assignment of values 0 or 1 to the variables x_i. Class C_i is selected if and only if $x_i = 1$. Let x be a solution. The cost of x is $z = \sum_{i=1}^{n} W_i x_i$. x is a feasible solution if $\sum_{i=1}^{n} a_{ij} x_i \geq 1$, $1 \leq j \leq m$.

x is a minimal solution if it is feasible and in addition $\sum_{i=1}^{n} W_i x_i$ is minimal. Let P_k be a vector that represents an assignment of 0 or 1 to some of the variables x_i. Thus if $P_k = (1, 3, \underline{4})$ then $x_1 = 1$, $x_3 = 1$ and $x_4 = 0$.

The above covering problem can be formulated as an integer programming problem. If the constrants $x_i = 1$ or 0, $1 \leq i \leq n$ are replaced by $x_i \geq 0$ $1 \leq i \leq n$, we get the following linear program:

$$\min \sum_{i=1}^{n} W_i x_i$$

subject to the constraints

$$\sum_{i=1}^{n} a_{ij} x_i \geq 1 \qquad 1 \leq j \leq m$$

$$x_i \geq 0 \qquad 1 \leq i \leq n$$

If x° is the solution of this linear program with cost z° then let $\langle x_i^\circ \rangle = \begin{cases} 0 & \text{if } x_i^\circ = 0 \\ 1 & \text{otherwise} \end{cases}$ and $\langle z^\circ \rangle = \sum_{i=1}^{n} W_i \langle x_i^\circ \rangle$.

Stage k:

Let x* be the best solution so far with cost z^* and let a particular P_k be given. Define $S_k^+ = \{i \mid i \in P_k\}$, $S_k^- = \{i \mid \underline{i} \in P_k\}$

$$F_k = \{i \mid i \notin S_k^+ \quad S_k^- \text{ and } 1 \leq i \leq n\}.$$

k · 1 Get a reduced cover table from the original one by deleting the set $\{C_i \mid i \in S_k^+ \cup S_k^-\}$, and deleting columns j for which $\sum_{i \in S^+} a_{ij} \geq 1$.

k ·2 Reduce the cover table further by applying rules 1 and 2'. Formulate a linear program for the final reduced table (In a manner similar to that described above for the complete table). Solve this linear program. P_k together with steps k·1 and k·2 gives a solution.

The following are possible:

a) An optimal integer solution x_k^* is obtained. Let its cost be Z_k^*. If $Z_k^* < Z^*$ let the new value of Z^* be Z_k^* and x* be x_k^*. If all entries of P_k are underlined, go to Quit. Else get P_{k+1} from P_k by underlining the rightmost nonunderlined entry in P_k and erasing all entries to the

right of it. Start stage k + 1.

b) There is no feasible solution. If all entries of P_k are underlined, go to Quit. Else, get P_{k+1} as in a) and start stage k+1.

c) An optimal noninteger solution x_k^* is obtained, with cost Z_k^*.

i) $\langle Z_k^* \rangle \geq Z^*$. If all entries of P_k are underlined, go to Quit. Else obtain P_{k_n} as in a) and start stage k+1.

ii) $\langle Z_k^* \rangle < Z^*$, then $\langle x_k^* \rangle$ defines a feasible solution better than x*. However, a much better solution, called a prime solution can be obtained by the following simple heuristic.

Consider the original cover table. Delete all rows except those corresponding to $\{C_i \mid i \in F_k \text{ and } x_i = 1\}$. Delete the columns corresponding to the subcommands covered by $\{C_i \mid i \in S_k^+\}$.

i) Reduce the table further according to rules 1 and 2'.

ii) Select the row C_i with the least cost | column. (Calculated by dividing the cost of C_i by the number of columns covered by C_i after step (i).) If there is more than one, select any. Delete the columns covered.

Repeat (i) and (ii) until the cover is complete. Let S_k be the set of rows selected by repeated application of (i) and (ii). S_k^+ and S_k define a solution x_k° with cost Z_k°. Let the new value of x* be x_k° and Z^k be Z_k°. Define P_{k+1} so that $S_{k+1}^+ = S_k^+ \quad S_k$ and $S_{k+1}^- = S_k^-$. Start stage k + 1.

Quit: Stop the computation. x* gives the minimal solution with cost z*.

Thus the procedure is as follows:

Start at stage 0 with $P_o = \emptyset$, $S_o^+ = S_o^- = \emptyset$ and $F_o = \{1, 2, \ldots, n\}$; $Z^* = \infty$ and x* undefined. If steps 0·1 and 0·2 yield an integer solution, we are done. Even if a noninteger solution is obtained the heuristic yields a result that is near minimal. (Let 0·1 and 0·2 yield a minimum cost Z_o^*, not necessarily integral). Then continue with stages 1,2,... till Quit is

reached or a solution with $Z_k^* = \lceil Z_o^* \rceil$ is obtained.

One further heuristic may decrease computation by indicating directly that no feasible solution exists at a particular stage. Let p be the number of columns selected in the heuristic at the end of the 0^{th} stage. For the original cover table determine the minimum number of rows that must be selected, using the method proposed by Du [3]. Let the number be r. Note that if there is no feasible solution using $\leq$ s maximal compatibility classes, then there is no feasible solution using only s rows. Thus by examining the numbers between r and p (possibly by doing a binary search if p - r is large) and the maximal compatibility classes only, we can find very quickly, a number t, $r \leq t \leq p$ so that we are guaranteed that any solution of the covering table must have at least t rows. The closer that t is to p, the better the heuristic will work. At any stage k ($k > 0$) we run a quick check on P_k and count the number of underlined indices. Let the number be p_k. If $n - p_k < t$ we can immediately conclude that there is no feasible solution and go directly to stage k + 1.

Finally, Das et. al [2] start with the same basic formulation as Grasselli and Montanari. However, they start directly with the maximal compatibility classes, MCC's whose number is usually small. The basic procedure is as follows:

Given a set of microcommands $\{S_1, S_2, \ldots S_k\}$ and a set of MCC's $\{ C_{m_1}, C_{m_2}, \ldots C_{m_n}\}$ obtained from the set of microcommands, a table is constructed by writing $S_1, S_2, \ldots S_k$ in a row and by entering C_{m_j} below S_i if $S_i \in C_{m_j}$. This table is called a CM cover table. The MCC's that appear alone in some columns are called globally essential. The CM cover table can be reduced by

1. Selecting globally essential MCC's and deleting the columns in which these MCC's appear.
2. Deleting all but one of the columns having identical sets of MCC's.

3. Deleting dominated columns.

Figure 10 gives a CM cover table corresponding to Figure 9 and Figure 11 a reduced CM cover table. The MCC's are :

C_1 = {a,g,k}, C_2 = {b,g,k}, C_3 ={c,j,k}, C_4 = {d,g,j}, C_5 = {d,i},

C_6 = {e,g,j,k}, C_7 = {e,h}, C_8 = {e,i,k}, C_9 = {f,g,j,k}, C_{10} = {f,i,k}.

The reduced CM cover table is then solved. One thus obtains the following irredundant solutions:

1) $C_1C_2C_3C_4C_7C_{10}$,

2) $C_1C_2C_3C_4C_7C_8C_9$,

3) $C_1C_2C_3C_5C_7C_{10}$,

4) $C_1C_2C_3C_5C_7C_9$.

These solutions are irredundant because if any class is dropped from any solution we no longer have a feasible solution (i.e., at least one microcommand will not be covered).

The following procedure is now carried out for each irredundant solution I_j.

A table (solution CM table) is constructed similar to the CM cover table where we restrict ourselves only to classes belonging to I_j. The table corresponding to the solution $C_1C_2C_3C_4C_7C_{10}$ is given in Figure 12.

The solution CM table indicates which particular MCC (or its subclass) has to be retained in the solution in order that all microcommands are included in the MCC's (or their subclasses).

The table in figure 12, for example, tells us that a can be covered only by C_1 or a subclass of C_1 containing a. Also, microcommands g,j,k can be covered by more than one MCC. To find the different covers a reduced table is constructed (figure 13).

This cover table is solved and irredundant solutions C_1C_3, C_1C_4, C_2C_3,

C_2C_4 and C_4C_{10} are found. For each of these solutions the following procedures is adopted:

Suppose we decide to cover g, j, k by C_3 and C_4. Then, retaining g,j,k in C_3 and C_4 and deleting their appearance from all other MCC's, the following solutions are obtained: (Remember that the overall irredundant solution is $C_1C_2C_3C_4C_7C_{10}$.)

{a}, {b},{c,j,k}, {e,h}, {d,g},{f,i}

{a}, {b}, {c,k}, {e,h}, {d,g,j}, {f,i}.

After going through all the iterations, the minimal solution is obtained.

Das et. al. also give two theorems which supposedly reduce the overall effort.

Comments:

We see in the Schwartz approach that the problem was formulated in the framework of switching theory. Two methods for solving the problem in this framework were presented: the integer programming method and the method of Das et. al. Which of these methods is better practically can only be decided by using both for minimizing real systems. It appears that the integer programming method can be easily programmed for a computer. Maybe a combination of the two methods would be the best.

Conclusions:

In this report we have surveyed most of the important research to date on minimizing the contol part of a microprogrammed digital computer. We feel that the results are largely negative, i.e., the Glushkov approach does not seem feasible in any practical environment and we have our reservations regarding the Schwartz approach. If the requirements of minimal solution are removed so that one would be satisfied with a near minimal solution, the integer programming method can be used since a very good solution is usually obtained after the first iteration.

For reducing the overall control unit, the engineering approach (Casaglia, et. al., Flynn and Rosin and Mishenko) seems to be the only feasible one.

Acknowledgement:

I am grateful to Professor Michael J. Flynn for his guidance, encouragement and support and for introducing me to this area.

Bibliography:

1. Casaglia, G. F., Gerace, G. B. and Vanaeshi, M., Equivalent Models and Comparisons of Microprogrammed Schemes, Internal Report No. 3, Special Series Instituto di Elaborazione della Informazione, C.N.R., Pisa, August 1971.

2. Das, S. R., Banerji, D. K. and Chattopadhyay, A., "On Control Memory Minimization in Microprogrammed Digital Computers", IEEE Transactions on Computers, Vol C-22, No. 9, September 1973, pp. 845-848.

3. Du, Min-Wen, "A Way to Find a Lower Bound for the Minimal Solution of the Covering Problem", IEEE Transactions on Computers, March 1972, pp. 317-318.

4. Flynn, M. J. and Rosin, R. F., "Microprogramming: An Introduction and a Viewpoint", IEEE Transactions on Computers, C-20, July 1971, pp. 727-731.

5. Garfunkel, R. S. and Nemhauser, G. L., Integer Programming, John Wiley and Sons, Inc. 1972.

6. Glushkov, V. M., "Automata Theory and Structural Design Problems of Digital Machines", Kibernetika, Vol 1, No. 1, 1965, pp. 3 - 11.

7. Glushkov, V. M., "Automata Theory and Formal Microprogram Transformations", Kibernetika, Vol 1, No. 5, 1965, pp. 1-9.

8. Glushkov, V. M., "Minimization of Microprograms and Algorithm Schemes", Kibernetika, Vol 2, No. 5, 1966, pp. 1 - 3.

9. Grasselli, A. and Montanari, U., "On the Minimization of READ-ONLY Memories in Microprogrammed Digital Computers", IEEE Transactions on Computers, November 1970, pp. 1111 - 1114.

10. Kohavi, Zvi, Switching and Finite Automata Theory, McGraw-Hill, Inc. 1970.

11. Mishenko, A. T., "The Formal Synthesis of an Automaton by a Microprogram II", Kibernetika, Vol 4, No. 5, 1968, pp. 21 - 27.

Bibliography: (continued)

12. Mischenko, A. T., "The Formal Synthesis of an Automaton by a Microprogram I", Kibernetika, Bol 4, No. 3, 1968, pp. 24 -31.

13. Schwartz, S. J., "An Algorithm for Minimizing Read only memories for machine control", 1968 9th Annual Symposium on Switching and Automata Theory, pp. 28 - 33.

Additional Referens Not Cited In Text:

14. Astopas, F. F. and Plukas, K. I., "A Method for Minimization of Microprograms for Digital Computers, Autom. & Vychisl. Tech., (USSR) No. 4, p. 10 - 16, 1971 (in Russian).

15. Chatelin, P. and Picket, E., "Minimization of the Control Part of a Microprogrammed Memory", Workshop on Microprogramming, Grenoble, France, June 1970. (abstract only) Two new algorithms based on Boolean methods and written in PL/1 were developed to minimize the length of microprogram words.

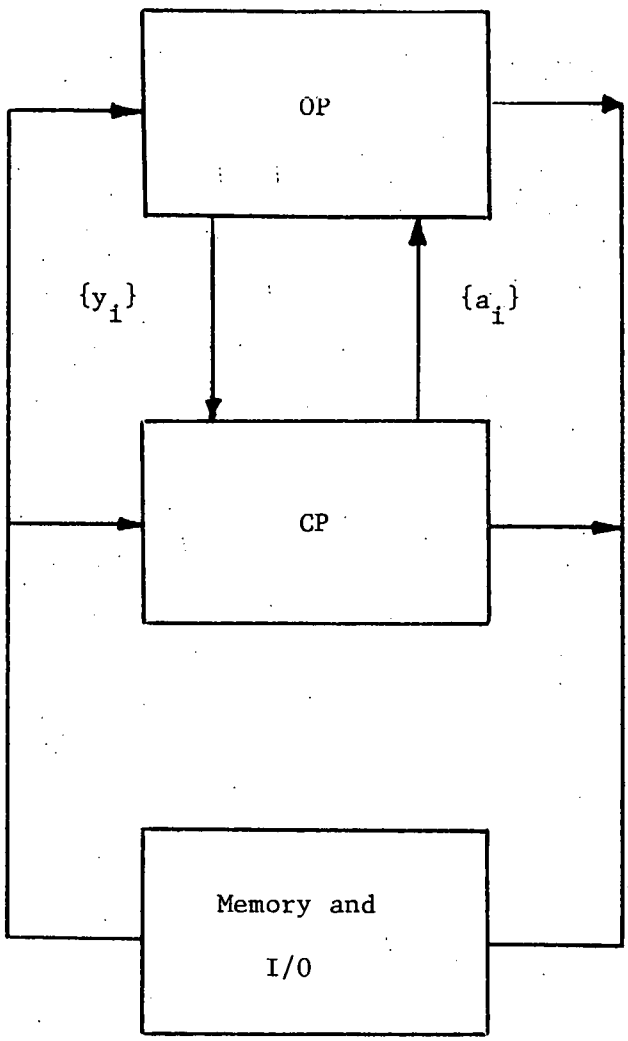


Figure 1

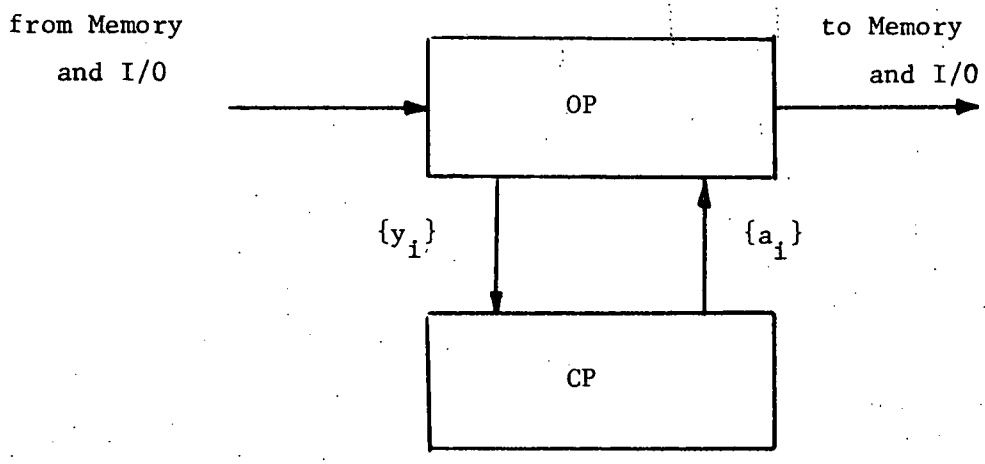


Figure 2

inputs / states	a_1	a_2	
p_1	p_2	p_3	y_1
p_2	p_2	p_3	y_2
p_3	p_2	p_4	y_1
p_4	p_2	p_1	y_2
			outputs

OP

Figure 3

inputs / states	y_1	y_2
1	2, a_1	1, a_1
2	2, a_1	3, a_2
3	3, a_2	1, a_2

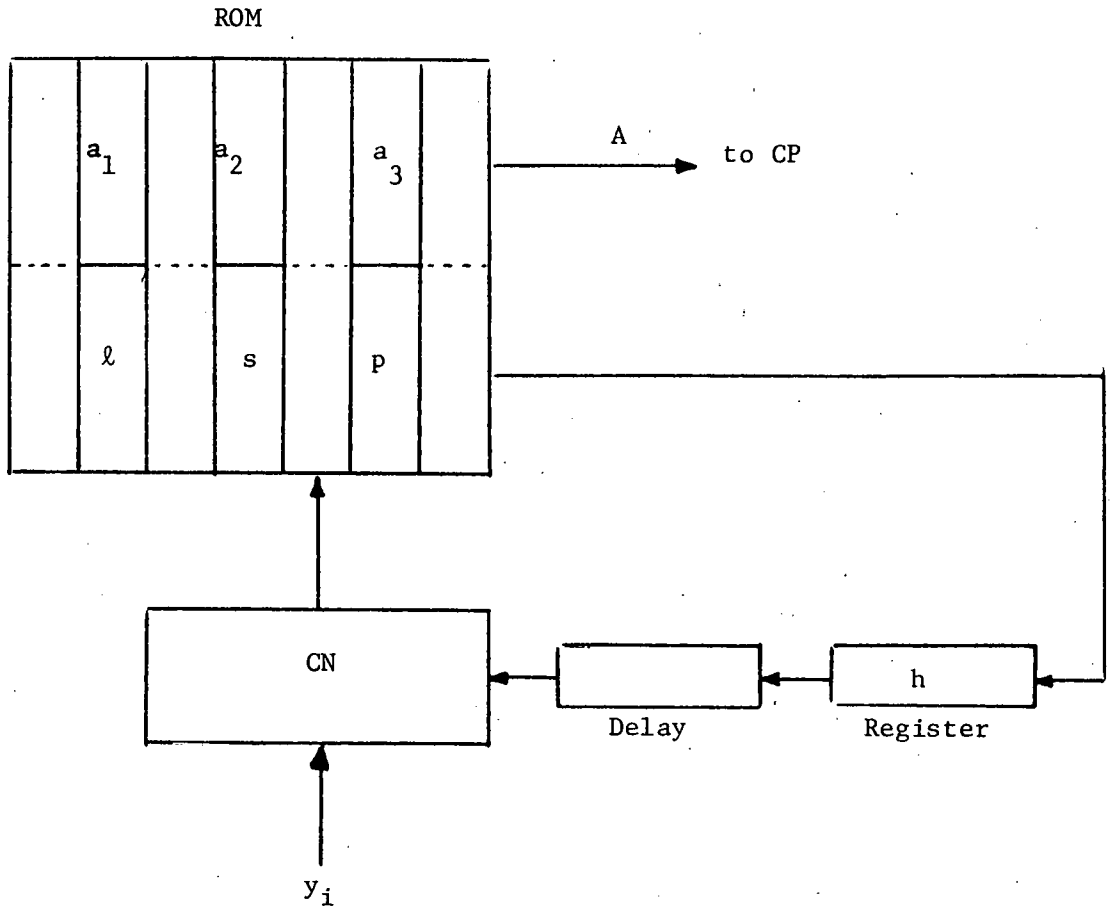
Figure 4

inputs / states	y_1	y_2
1	$2,a_1$	-
2	-	$3,a_2$
3	$3,a_2$	$1,a_2$

Figure 5

inputs / states	y_1	y_2
1	$1,a_1$	$2,a_2$
2	$2,a_2$	$1,a_2$

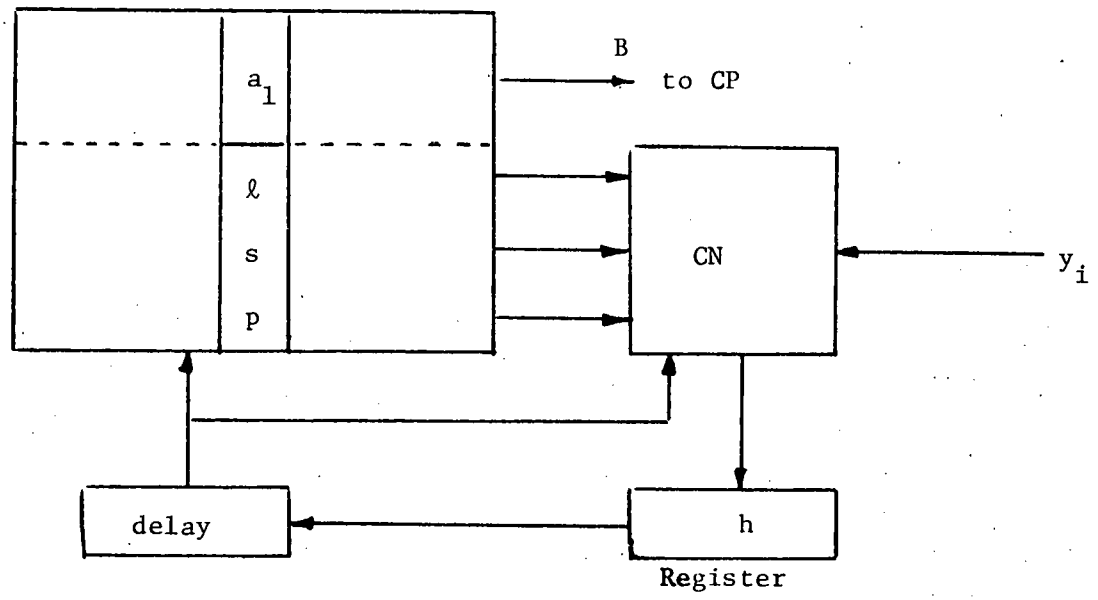
Figure 6



Mealy CP executing ps microinstruction

$|h|$ (x_i) a_i, ℓ; (x_2) a_2, s; (x_3) a_3, p

Figure 7



Moore CP executing ms microinstruction

$|h|$ a_1 (x_1) ℓ, (x_2) p, (x_3) s

Figure 8

WORD	SUBCOMMANDS
1	a,b,c,d,e,f
2	c,g,h,i
3	a,b,h,i,j
4	d,h,k
5	f,h

Figure 9

a	b	c	d	e	f	g	h	i	j	k
c_1	c_2	c_3	c_4 c_5	c_6 c_7 c_8	c_9 c_{10}	c_1 c_2 c_4 c_6 c_9	c_7	c_5 c_8 c_{10}	c_3 c_4 c_6 c_9	c_1 c_2 c_3 c_4 c_6 c_8 c_9 c_{10}

Figure 10

d	f	i
c_4 c_5	c_9 c_{10}	c_5 c_8 c_{10}

Figure 11

a	b	c	d	e	f	g	h	i	j	k
c_1	c_2	c_3	c_4	c_7	c_{10}	c_1 c_2 c_4	c_7	c_{10}	c_3 c_4	c_1 c_2 c_3 c_{10}

Figure 12

g	j	k
c_1 c_2 c_4	c_3 c_4	c_1 c_2 c_4 c_{10}

Figure 13