HYBRID TECHNIQUES APPLIED TO OPTIMIZATION PROBLEMS

Hans S. Witsenhausen
Electronic Associates, Inc.
Princeton, New Jersey

SUMMARY

A function dependent on the solution of a set of differential equations containing adjustable parameters, can be minimized by systematic search procedures in parameter space. Such procedures can be implemented by a hybrid system consisting of a general purpose analog computer and a digital expansion providing parallel logic building blocks. Programming of such a system is illustrated for a simple search procedure in n parameters.

INTRODUCTION

The solutions of many problems benefit from the use of a combination of both analog and digital computing devices. The required hybrid techniques of computation can be carried out by the linkage of an analog computer with a general purpose digital computer. For iterative solution of problems, in particular system optimization, a more desirable approach is the addition to the analog computer of an EAI HYDAC Series 350 Digital Operations System designed to provide primarily:

(a.) Flexible automatic logic control of an analog computation or a long sequence of analog computations.

(b.) Large capacity, fast-access storage for problem solutions requiring a sequence of computations with variable function storage.

The assembly of many analog and digital computing components in an integral system gives the programmer complete flexibility in his organization of a problem solution. To demonstrate what is possible with a system this paper considers a problem solution which makes good use of the flexible automatic logic provided by the proposed digital expansion system. For simplicity in explanation, this more basic feature of the hybrid computer is applied to the solution of a straightforward problem in optimization.

The Series 350 HYDAC D. O. S. provides "logic building blocks" which can be connected to form in this case the automatic control of the optimization program. These digital modules are electronic (solid-state) devices providing high computing speed. Any logical function could be implemented by relays and stepping switches, as has been done in many analog computer applications in the past. [6, 7, 8] However, higher speeds, flexibility and programming convenience are required in order to take full advantage of the speed of analog computers. This is not done in the repetitive operation mode where successive computations are simply repetitious or differ only by small perturbations. The full possibilities are realized by speeding up the usual real-time operation where the operator makes intelligent decisions between computations, on the basis of the fresh information just obtained. Successive computations can then be distinctly different.

The digital modules are interconnected on a removable patch panel. Programming is conveniently similar in principle to the approach familiar to analog computer users.

Series 350 HYDAC Digital Operations System

GENERAL FEATURES

The Digital Operations System building blocks use two voltage levels to represent 0 and 1. A connection carrying a 1 level is said to be energized. Logical functions can be mechanized by interconnection of gates operating on these levels. Sequential logic requires flip-flops. Changes in the state of a flip-flop are clocked by a central system clock. Frequency of the clock pulses is adjustable from several megacycles down to manual control, pulse by pulse, by a push-button. This enables the operator to check parts of the set-up at leisure.

Counters provided with the system can be used to generate signals at binary or decimal submultiples of the central clock frequency.

LOGIC BUILDING BLOCKS

For the present purpose, only four types of building blocks are required: (See Figure 1)

(a.) AND gates (2 inputs), the negated output is provided.

(b.) OR gates (up to 6 inputs), the negated output is provided. (AND and OR gates, although logically interchangeable through negation, are distinguished for ease of intuitive programming).

(c.) General purpose flip-flops (GPFF).

These are clocked RST flip-flops. The S (set) input commands the 1 state; the R (reset) input the 0 state; the T (trigger) input commands state reversal. Only one of these inputs may be energized at a time. The input levels are sampled by the clock pulse. Changes in state take place and the new state is established before the following pulse. Assembled in groups of four, these flip-flops have two group-input controls: a "clear" input which resets all four, and an "enable" input which can be used to inhibit the RST inputs.

(d.) Quadruple flip-flops in shift register connection. (QFF) These are sets of four flip-flops with internal connection into a shift register. Each set has eight outputs, the 0 and 1 states of each flip-flop.

The inputs are as follows:

Serial Inputs: The serial set (SS) and serial reset (SR) inputs are applied only to the first flip-flop and can be patched, in particular, from either the output of the last flip-flop of another QFF package for construction of long registers or the output of the last flip-flop in the register for closure of a loop.

Shift Input (SH): Transfers the state of each flip-flop to the next in order on one clock pulse and enables the serial inputs to control the first flip-flop.

Clear Input: (CLR) Resets all four flip-flops.

Set and Load Inputs: One set input(S) is provided for each flip-flop. The load input (L) is common to the building block. When the load input is energized, flip-flops for which the set input is energized will go into the 1 state.

PROGRAMMING OF THE SERIES 350 D. O. S.

The techniques for programming the Digital Operations System are presently unfamiliar to most analog and digital computer users. The program is not introduced as a list of successive instructions but as a set of patching connections. This implies a task of organization similar to that needed in the design of a digital computer. However, there is considerable difference in the emphasis placed on component economy. Economy is essential for a computer design but of little concern in the programming of the digital expansion system as long as enough components are available. Thus, an empirical approach to digital expansion system programming, by successive simplifications and rearrangements is entirely acceptable.

The logic building blocks provide entirely parallel operation though it is possible to perform digital operations serially when desired. The digital and analog programs are introduced by their respective patch panels and interconnected by trunk lines forming one single set of parallel analog-digital components.

Several signals from analog comparators can be accepted simultaneously and combined to form several logical functions, while simultaneously several commands to analog switches (electronic relays, analog memory, mode control) can be generated.

This paper shows some aspects of this programming technique, for consideration by analog computer users. To this effect, it considers the general optimization problem which is both significant and representative of the class of applications requiring logic operations rather than large capacity storage.

For clarity, no attempt is made to reduce the number of building blocks to a minimum.

USE OF SUBROUTINES

Operations in the analog computer have to follow a definite time sequence. Whenever a sequence of operations has to be repeated many times, it is defined as a subroutine. One way to implement a subroutine is to use a ring counter, acting effectively as an electronic stepping switch, where the sequence of states determines the sequence of operations. A ring counter is obtained by assembling QFF units into a shift register of the desired length and then connecting the outputs of the last stage to the serial inputs of the first QFF. Initially the first flip-flop in this loop is set to the 1 state and the others are in the zero state. When the shift input is energized each clock pulse will advance the sequence by moving the 1 state to the next flip-flop resetting the previous one. After all flip-flops have been energized successively the next shift returns the ring to the initial state. Each flip-flop can command in parallel a set of operations, including the initiation of other subroutines.

The effective use of subroutines depends on two features:

(a.) Appropriate control by other routines.

(b.) Provision for delays necessary for operation of analog components.

CONTROL OF INTERLOCKING SUBROUTINES

When a routine calls, at one stage, for the execution of a subroutine it is often necessary to halt the advance of this routine until the subroutine has gone through its entire cycle. This interlock can be obtained by the formation for a given subroutine A of 3 signals.

sA: this signal is the call for subroutine A; it is formed by an OR gate which receives signals from all the sources of calls, generally different positions in one or more other routines.

eA: this signal is generated at the time routine A returns to its initial state. It is produced by an AND gate supplied from the last state of A and the shift input of A. The eA signal is energized for one clock time and is called the end signal of subroutine A. If a subroutine should be repeated n times, a separate counter (analogous to index registers) can be used to produce the end pulse.

eA^X: this signal is produced by an OR gate supplied by eA and by the negation $\overline{sA}$ of sA. Called the interlock signal, eA^X is fanned out to all routines which call subroutine A.

The required interlock is obtained by supplying the shift input of a routine by the AND combination of its own call signal and the interlock signals of all the routines for which it calls.

DELAYS

Some routines control electronic relays, analog memory and mode of integrators in the analog computer. When exercising such control the routine must incorporate delays to allow sufficient time for changes in computation to take place so that solutions do not include the transient response of amplifiers and the charging of capacitors necessary between consecutive operations.

These delays can be obtained by an AND gate at the shift input to the routine; this gate is fed by the shift command and timing signals of slow rate derived from the carries of the central clock counter. Different rates can be used in different routines.

If delays of different sizes are required in the same routine, the routine itself can switch control from timing signals of one rate to those of another. Alternatively general purpose delay units (one-shot multivibrators) can be used to inhibit shifting during the required time span.

The delay required while the analog computer is going through an operate cycle, is obtained by inhibiting shifts of the controlling subroutine in the operate mode; this takes care of the cases where the run time is variable, depending on some condition in the problem.

SIGNALS FROM DIGITAL TO ANALOG EQUIPMENT

Signals computed by digital components and transferred to the analog computer are required for the following purposes:

(a.) Control of the analog computer mode.

(b.) Control of the mode of individual integrators or groups of integrators.

(c.) Control of analog storage units.

(d.) Control electronic gates used for switching of operations.

For the control of the analog computer modes (OPERATE, HOLD, INITIAL CONDITION) input terminations are provided on the digital patchboard. A pulse applied to any input termination will trigger the corresponding mode.

The other control functions are effected by electronic gates which accept digital signals. The opening or closing of the gate is a function of the applied digital level. In many cases, a GPFF will buffer these commands so that the condition of the gate is maintained until new signals are fed to the GPFF. One GPFF can control several gates in synchronism.

SIGNALS FROM ANALOG TO DIGITAL EQUIPMENT

Particular signals produced within the analog computer are transferred to the digital equipment. These signals provide information of:

(a.) The analog computer mode, and

(b.) The state of comparators.

The computer mode signals are ICL, HL, and OPL which are energized whenever the analog computer is in the initial condition, hold, or operate mode, respectively.

Comparators will supply one of two levels depending on the sign of the algebraic sum of their analog voltage inputs. These levels are normalized for direct utilization in the digital system.

The "Optimization Problem"

A set of algebraic and differential equations, mechanized on the analog computer, uniquely determines in one computation (the run) a value E, the loss-function or error-function to be minimized by manipulation of n parameters $\lambda_1 \ldots\ldots, \lambda_n$.

This problem occurs in many situations;

--solution of a set of algebraic equations
--matching of boundary conditions
--search for eigenvalues
-- process optimization
--model building
--curve fitting etc.....

While special techniques exist for each class of problem, the general problem is considered here.

Analytical methods [1, 2, 5] for solving optimization problems, such as that of steepest descent, make use of partial derivatives determined either from the analytical problem statement or by parameter influence techniques[3]. The experimental approach considered is quite different conceptually, for the search for an optimum is entirely based on repeated tests with different parameter combinations. Experimental methods [4, 6] can use a varied degree of sophistication to accelerate convergence to an optimum and to overcome possible difficulties due to the features of the unknown hypersurface. $E(\lambda_1, \ldots\ldots, \lambda_n)$.

The price of refined search procedures increases very rapidly with the number n of parameters. For the purpose of illustrating the methods of hybrid programming of any search technique the following simple approach is selected.

SEARCH PROCEDURE

Assume normalization of all parameters so that a unique "exploration distance", h (say 1 volt on a $\pm$ 100 volt computer), is acceptable.

Given λ_{io} and E_o at an arbitrary starting point, the first partial derivatives are approximated by

$$\frac{\partial E}{\partial \lambda_i} \simeq \frac{\delta E^+_i - \delta E^-_i}{2h}$$

where

$$\delta E^{+}_{i} = E(\lambda_{1o}, \ldots, \lambda_{io}+h, \ldots, \lambda_{no}) - E_o$$

$$\delta E^{-}_{i} = E(\lambda_{1o}, \ldots, \lambda_{io}-h, \ldots, \lambda_{no}) - E_o$$

All n partial derivatives are thus approximated by means of 2n calculations of E (runs). In order to place most of the load in the digital part of the hybrid system, the values of the partial derivatives are stored in quantized form. Instead of storing $\partial E/\partial \lambda_i$, a quantity p_i, susceptible of only 3 values, is defined by comparison with a limit L selected by the computer operator. Extension to a higher number of values poses no special problem.

$$p_i = +1 \quad \text{if } \delta E^{+} - \delta E^{-} < -2hL$$

$$p_i = 0 \quad \text{if } -2hL < \delta E^{+} - \delta E^{-} < 2hL$$

$$p_i = -1 \quad \text{if } \delta E^{+} - \delta E^{-} > 2hL$$

The vector with components p_i defines a direction in n-space in which a decrease in E is probable. It is close to the steepest descent direction for which the components are $\partial E/\partial \lambda_i$. The program stops if all $p_i = 0$. With one of the 3^n-1 possible directions determined, the parameters are changed to $\lambda_{io} + p_i \Delta_1$, where Δ_1 is a fixed quantity (say 2 volts). If an improvement, indicated by a decrease in E, is obtained the values of all parameters λ_{io} are "updated" to give a new starting point. Additional changes by steps, Δ_1, in the same direction are carried out until no more improvement occurs. Then a smaller step-size Δ_2 (say .2 volt) is selected and when this also fails, the partial derivatives are redetermined and the process repeated. If, after a new determination of the quantities p_i, even a single small step leads to no improvement, the program stops. If the analog computer does not overload, the program will ultimately reach one of the stop conditions and this should occur in the vicinity of the optimum.

An advantage of this technique is that the equipment requirement, as demonstrated later, has a slow linear increase with n. It is recognized that many possible features of the E function (high curvatures, local minima, narrow winding "canyons", etc.) will defeat the search procedure; on the other hand, the very same features can also defeat more elaborate and equipment-consuming alternatives.

If a Euclidean metric based on voltage is used, the step length $\Delta\sqrt{\Sigma\ p_i^2}$ will vary between Δ and $\Delta\sqrt{n}$ depending on the direction selected. This is acceptable because there is no a priori reason for a voltage metric to have special significance.

PROGRAM REFINEMENTS:

The procedure outlined above can be improved at a small expense in equipment by addition of the following features:

(1) If, after determination of direction, at least one large step Δ_1 is successful, then no small steps are used before a new determination of direction. However, when a single large step does not decrease E, a smaller step Δ_2 is used. This program arrangement is a time-saving feature.

(2) For non-convex E functions it is necessary to redetermine direction from time to time even if improvement is constantly obtained. To this end an upper limit N (say 8) is put on the number of steps in a direction before redetermination of direction.

(3) The limit L for quantization of the slopes should be carefully chosen, If L is too large, the program will stop in any "flat" region of the E-function and for non-convex functions this final value of E may be far from a minimum. If L is too small, all quantities p_i will have values ± 1 most of the time. This has the disadvantage of reducing the effective set of possible directions from 3^n-1 to 2^n (for n=6 from 728 to 64) and therefore the efficiency of the search procedure.

Many refinements are possible to avoid this difficulty. The following is selected for its economy of equipment.

Each slope is compared with two limit values L_1 and L_2 ($L_1 < L_2$). If no slope exceeds L_2 the quantities p_i are based on L_1. If any slopes exceed L_2 the quantities p_i are based on L_2. This procedure is poor whenever many slopes are clustered in a narrow range around either L_1 or L_2, but this is unlikely to happen frequently in a long optimization. The procedure requires just one more comparator in the analog computer.

(4) Whenever desired, half the exploration runs can be saved by replacing the slope evaluation $\frac{\delta E^{+} - \delta E^{-}}{2h}$ by $\frac{\delta E^{+}}{h}$, that is used forward

instead of central differences.

ANALOG PROGRAM

From the statement of the optimization procedure the analog circuit (Figure 2) can be derived immediately as follows.

The problem equations are mechanized by circuits which accept n input signals λ_i. The last updated values of the parameters λ_{io} are held on n analog memories. The size h, Δ_1, or Δ_2 of the increments $\delta\lambda$ is selected by 2 switches and distributed with both signs to n pairs of switches; for each parameter one switch determines the sign of $\delta\lambda_i$, the other determines the addition of $\delta\lambda_i$ to λ_{io}. This addition is performed by an analog memory so that the new value $\lambda_i = \lambda_{io} + \delta\lambda_i$ can be transferred to the λ_{io} memories when updating of the parameters is required.

The duration of the operate period (the run) is determined in general by the problem itself and can be dependent on the parameters. A comparator produces the hold signal when the appropriate condition is satisfied. At that time the value E becomes available.

The previous updated value E_o is held on a memory amplifier. The difference $E - E_o$ is fed to two memory amplifiers so that δE^+ and δE^- can be successively computed, stored and then compared. The value E itself is stored to permit updating of E_o when required. If the problem integrators can conveniently be used as storage for the value E the number of analog memories required at the output can be reduced from four to two.

For determination of the direction parameters p_i the sign of $\delta E^+ - \delta E^-$ is sensed by a comparator, and the absolute value of this difference is compared with two constants by two more comparators.

After a step has been carried out the change in the E function is stored in one of the δE memories, say δE^+, and the sign of this quantity is sensed by a comparator to determine whether improvement has been obtained. Table 1 shows the conditions for digital control of the circuit switches.

The comparators deliver information about the E-function to the digital equipment according to Table 2.

TABLE I

UNITS	DIGITAL LEVEL 0	1
Δ_1/Δ_2 relay	Δ_1	Δ_2
h/Δ relay	h	Δ
$\delta\lambda_i$ Off/On relay $i = 1, 2, \ldots n$	Off	On
$\delta\lambda_i$ + / - relay	+	-
Stores for E_o and λ_{io}	store	track
Stores for -E and $-\lambda_i$	track	store
δE^+ store	track	store
δE^- store	track	store

TABLE 2

ANALOG CONDITION		DIGITAL LEVEL	NAME	MEANING
$\operatorname{sgn} \delta E^+$	+	0	IMP	Improvement
	-	1		
$\operatorname{sgn} (\delta E^- - \delta E^+)$	+	0	N	Best direction is $\delta\lambda_i < 0$
	-	1		
$\operatorname{sgn} \left(\lvert \delta E^- - \delta E^+ \rvert \; . 2hL_j \right)$	+	0	L_j	Change in λ_i is advisable
	-	1		

DIGITAL OPERATIONS SYSTEM PROGRAM

The Digital Operations System program must insure that, once begun, the computation progresses automatically in its search for an optimum. This part of the program has inputs only from the IMP, N, L_1, L_2 comparator signals and the analog computer mode signals.

PROGRAM STRUCTURE

The first step in producing such a program is to specify a flow diagram of the required sequence of operations. (Figure 3) This flow diagram organizes the operations into the following subroutines:

Master Routine: selects the step size h, Δ_1 or Δ_2 tests the two stop conditions and calls alternately for the subroutines Explore, for determination of a direction, and Proceed, for carrying out steps in that direction.

Explore Routine: calls for a computation of E with a single parameter displaced by +h then by -h. It steps through all parameters storing the p_i as they are obtained. When all p_i have been obtained it sets the $\delta\lambda_i$ switches accordingly.

Run Routine: executes the computation of E for the prevailing λ_i values.

Proceed Routine: A step size (Δ_1 or Δ_2) being selected by the Master routine and a direction (a set of p_i) having been set by the Explore routine, the Proceed routine displaces the λ_i accordingly, calls for computation of E and tests for improvement. The routine repeats as long as improvement is obtained, unless a preset number of steps is exceeded, and calls for the updating routine after each improvement.

Update Routine: transfers the present values of λ_i and E into the λ_{io}, E_o memory units.

PROGRAM MECHANIZATION

The subroutines are instrumented in detail as follows:

Subroutine C (Figure 4) (RUN) will start with the analog computer in its IC mode. The call signal sC is obtained by an OR gate (not shown) which combines all sources of calls from the EXPLORE and PROCEED routines. The routine will be started after new commands have been given for the $\delta\lambda$; a delay is necessary for acquisition of the new λ_i by the analog computer (especially if some of them are initial conditions). Next the OP mode is triggered and the subroutine inhibited by the OP level. Upon automatic hold, the routine resumes with a delay to allow for acquisition of the final δE values, Then the δE^+ store and, if permitted by a signal ENHM (enable hold minus) generated in the EXPLORE routine, the δE^- store are switched to hold by setting the flip-flops controlling the memories. After another delay, the subroutine returns to its initial state, generating the end signal eC. The interlock signal eC^X is used to inhibit the calling routines.

Subroutine D (Figure 5) (UPDATE) will update the λ_{io}, E_o values to the last λ_i, E values. Here delays are provided to enable the tracking stores to acquire steady-state values. Since only two states are required, the ring counter is replaced by a GPFF with the trigger input acting as the shift control. Signals eD and eD^X are generated, eD^X inhibits the PROCEED routine.

The delays for subroutines C and D are obtained by the use of a slow clock rate (SLCL) derived from a carry of a clock counter.

Subroutine A (Figure 6) is the program for determination of approximate partial derivatives (EXPLORE). It is associated with shift registers δ^X, δ, N^X, N.

δ^X and N^X control the analog value of $\delta\lambda$: The data obtained during exploration is stored in registers δ and N.

The subroutine begins with the δ^X, N^X cleared. First $\delta_1{}^X$ is loaded and subroutine C called then $N_1{}^X$ is loaded (implementing $\delta\lambda = -h$) and subroutine C called with ENHM turned on. At this stage the correct value of N, L_1, L_2 appear on the comparators. A test is carried out to determine which comparison level should be used. The first occurence of a slope $> L_2$ will clear all previous δ's, set flip-flop L, which controls determination of future δ's on the basis of L_2 alone. If L_2 is not exceeded, δ's are selected according to L_1. The routine shifts the new data into the δ, N stores, sends the δE memories to the tracking mode and repeats, shifting to the next parameter. When the last parameter is reached the δ^X, N^X are cleared and the δ, N are transferred into δ^X, N^X registers in a parallel loading operation. The end signal and the interlock signal eA^X are generated.

Subroutine B (Figure 7) will carry out steps in the chosen direction (PROCEED). It calls upon subroutine C for an analog run, then senses the IMP comparator level. If no improvement is obtained the end pulse $eB^{(1)}$ is generated. If improvement is obtained, subroutine D is requested for updating and the S flip-flop associated with the master routine is set, to record success. The end signal eD of the UPDATE routine increments a counter (r) and subroutine B repeats unless r = N where the routine end pulse $eB^{(2)}$ is generated. The two types of end signals are combined to generate the interlock signal eB^X.

Master Routine M (Figure 8) first calls upon A for exploration, selecting step size h. Next it tests the stop condition $\delta_1 = \ldots = \delta_n = 0$. It then calls upon B (PROCEED) with the large step-size Δ_1. If, and only if, at the end of B, S = 0 (no improvement) B is repeated with the smaller step-size Δ_2. Next the stop condition S = 0 is tested. If S = 1 it is reset to 0, the δ^X, N^X are cleared and routine M repeats.

Initialization (not illustrated): It is convenient to initialize the digital system by clearing all its flip-flops from a central control. At this time the operator can introduce starting values for λ_{io}. The hybrid program will start as soon as the system clock is turned on. A non-repeating initialization routine is patched to perform the following operations:

(a.) Set all flip-flops which should be energized initially.

(b.) Call on the RUN subroutine to calculate and store the value E_0 corresponding to the initial parameter values.

(c.) Start the master routine of the problem.

EQUIPMENT REQUIREMENTS

If no further rearrangements are carried out, the following numbers of components are required, where n is the number of parameters.

On the analog side: $2n + 2$ switches
$2n + 4$ analog memory units
5 comparators

On the digital side: 11 GPFF
$6 + 4\left[\frac{n}{4}\right]$ QFF

50 AND/OR GATES
1 counter

CONCLUSIONS

The addition of the EAI Series 350 HYDAC Digital Operations System providing logic capability to the general purpose analog computer will enlarge its capabilities as an automatic computing device.

The less flexible relay and stepping-switch arrangements used in the past are eliminated and full use is made of the computing speed of the analog computer. This alone brings within reach problems, such as optimization, where computing times were previously excessive in most cases. The further addition of point storage in digital form on delay lines will permit the use of serial techniques for solution of integral and partial differential equations. Problems of a statistical nature can be handled with a maximum of automatic operation. Many simpler auxiliary functions for GPAC's can be easily carried out (automatic scale changes, editing of graphical plots which are not continuous in time, Fourier analysis).

ACKNOWLEDGEMENT

The author is indebted to Mr. J. Paul Landauer of Electronic Associates, Incorporated, for information and suggestions.

REFERENCES

1. Brooks, S. H., "A Comparison of Maximum-Seeking Methods," Operations Research, 7, No. 4, 1959.

2. Kinney, R., "A Model Adaptation Technique for Computer Controlled Batch Chemical Process," M. S. Thesis, Case Institute of Technology, 1960.

3. Brunner, W., "An Iteration Procedure for Parametric Model Building and Boundary Value Problems," Western Joint Computer Conference, 1961.

4. Box, G. E. P. and Wilson, K. B., Journal Royal Statistical Society, B13, 1, 1951.

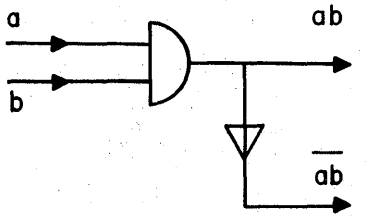
5. Lapidus, L., Shapiro, E., Shapiro, S. and Stillman, R. E., "Optimization of Process Performance", A. I. Ch. E. Journal, 7, 288, 1961.

6. Munson, J. and Rubin, A. I., "Optimization by Random Search on the Analog Computer," I. R. E. Trans. EC-8, No. 2, 200, 1959.

7. Follin, J. W., Emch, G. F. and Walters, F. M. "Modification and Additions to the REAC," Project Cyclone Symposium II, p. 173, 1952.

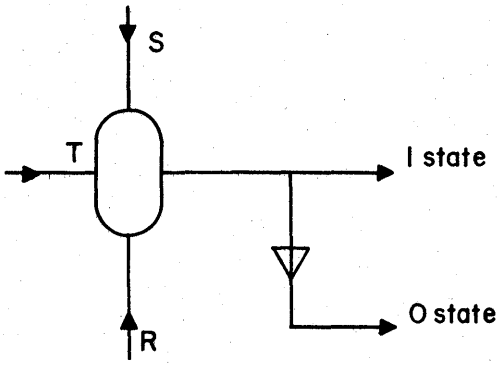
8. Hansen, G. R., "The Time-Sequence Controller for Automatic Operation of the Electronic Differential Analyzer," Project Typhoon Symposium III, p. 65, 1953.

SYMBOLS

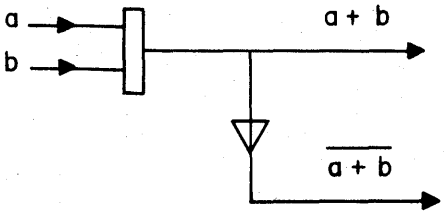


AND gate

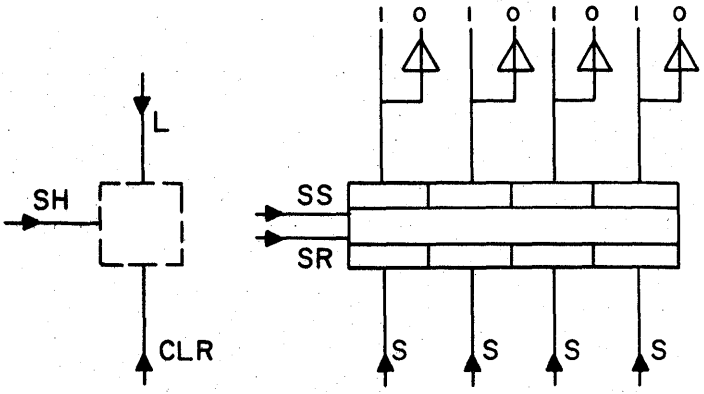
negation



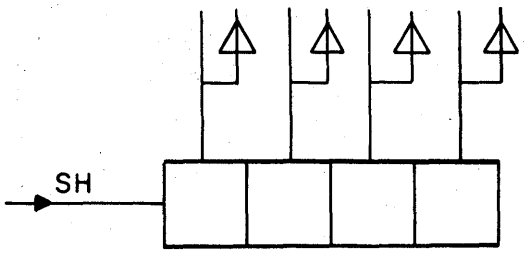
GPFF general purpose flip-flop



OR gate



QFF quadruple flip-flop



Ring Counter

Figure 1: Digital System Symbols

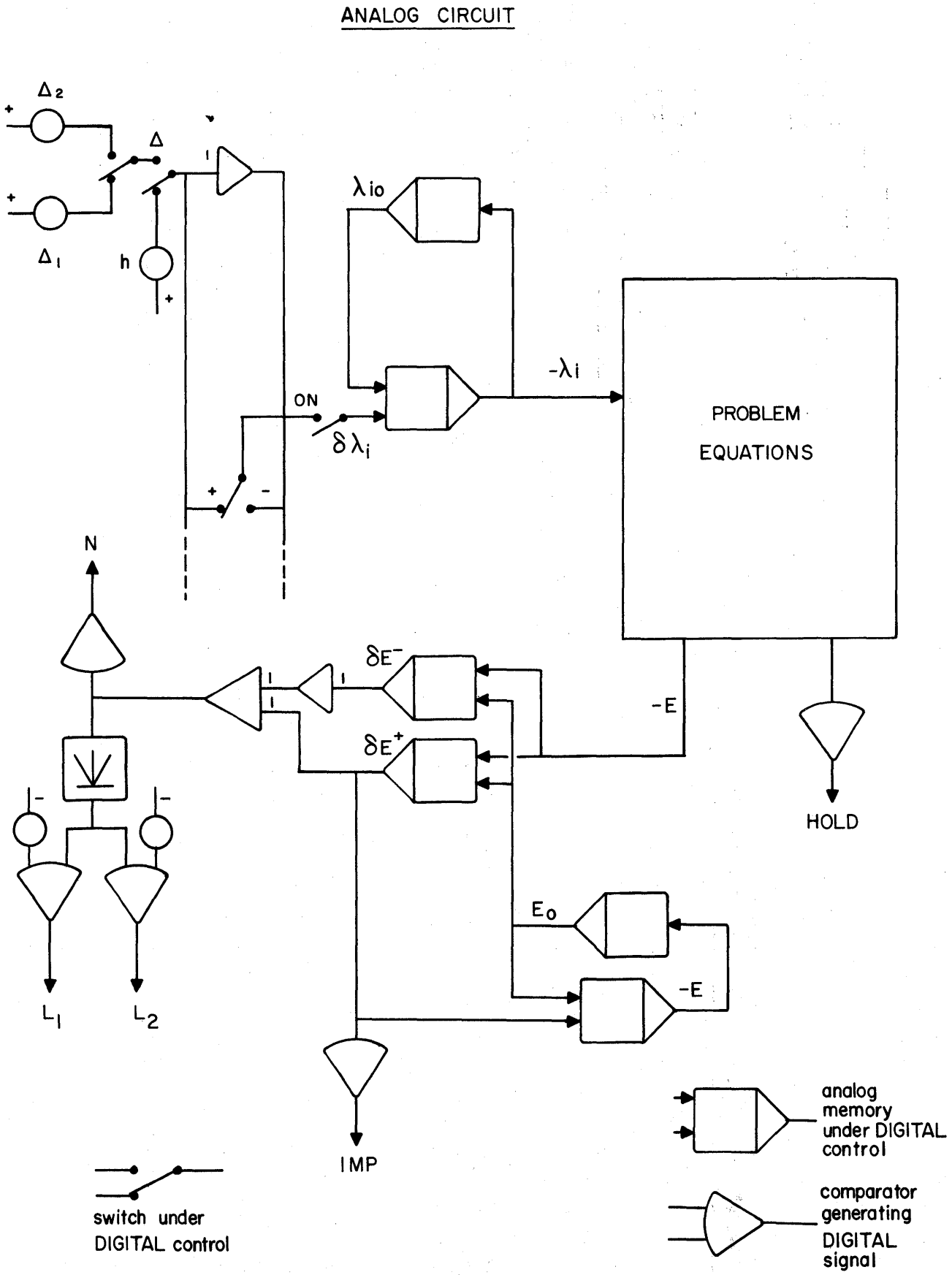


Figure 2: Analog Computer Circuit

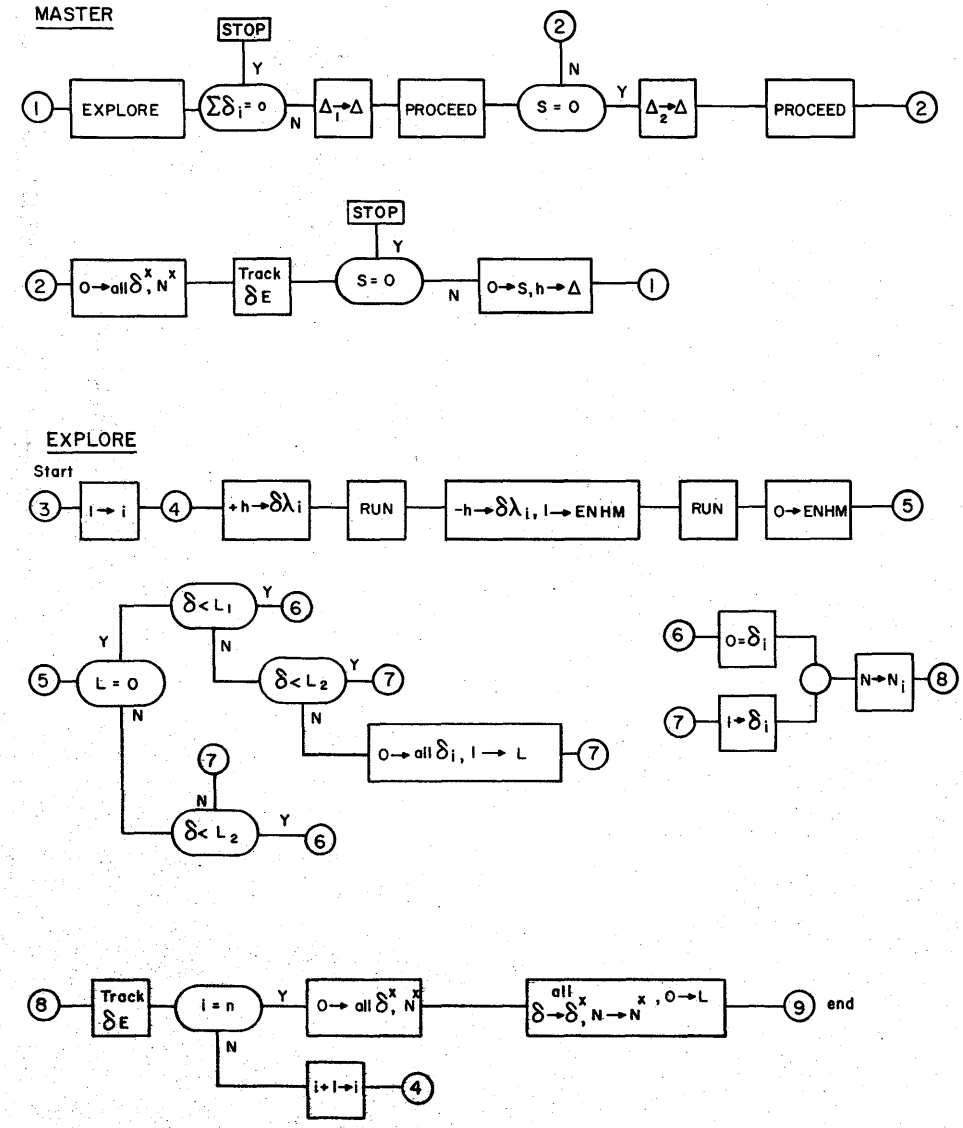
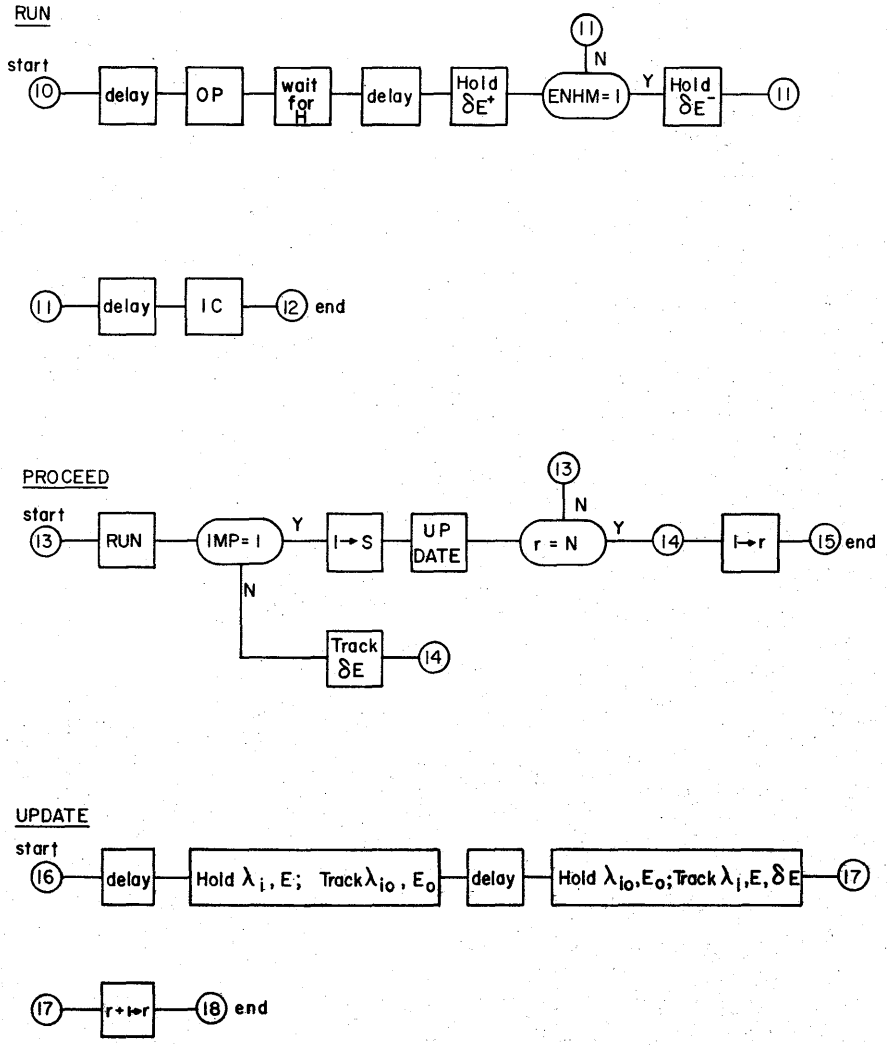


Figure 3: Program Flow Diagram



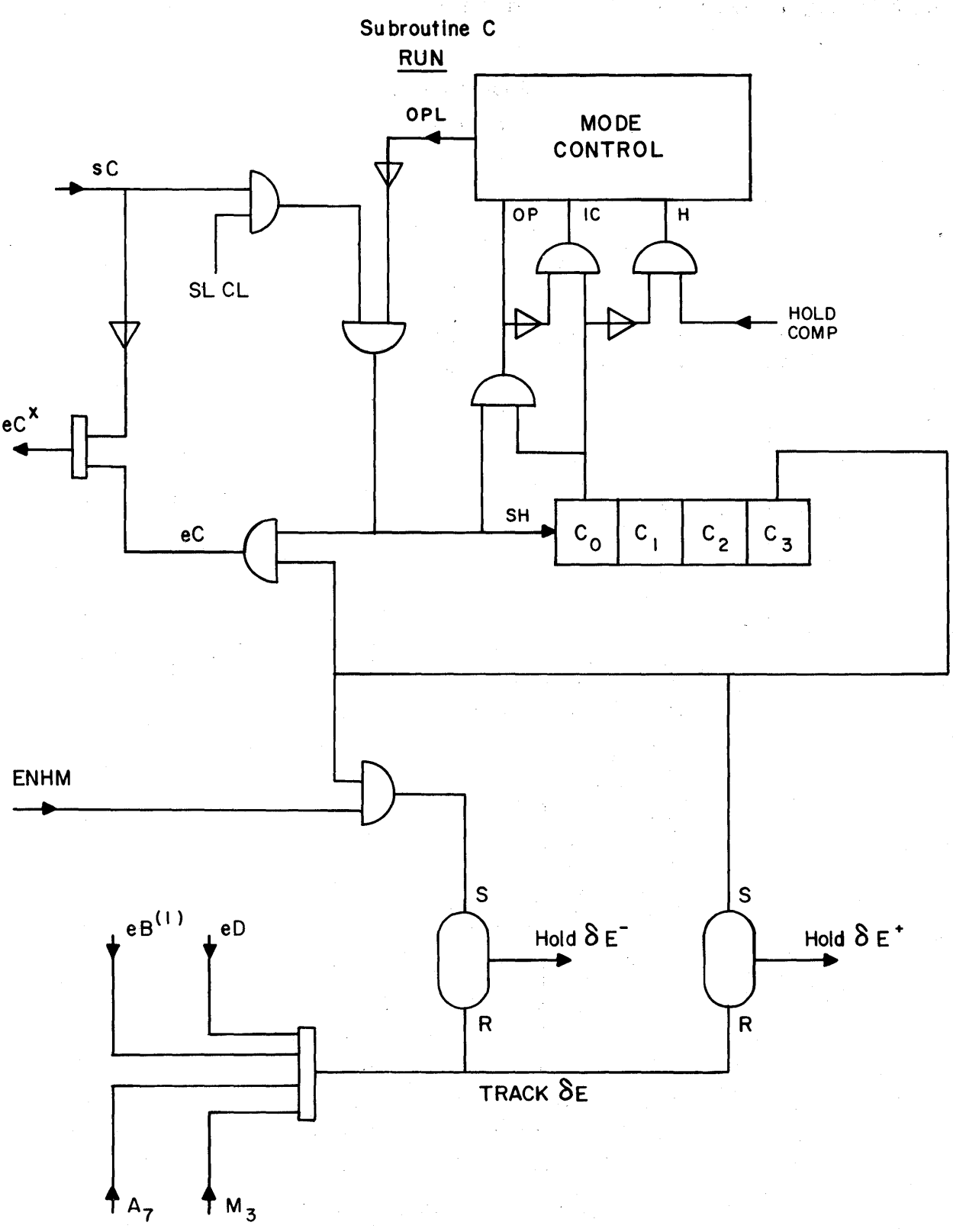


Figure 4: Subroutine C. RUN

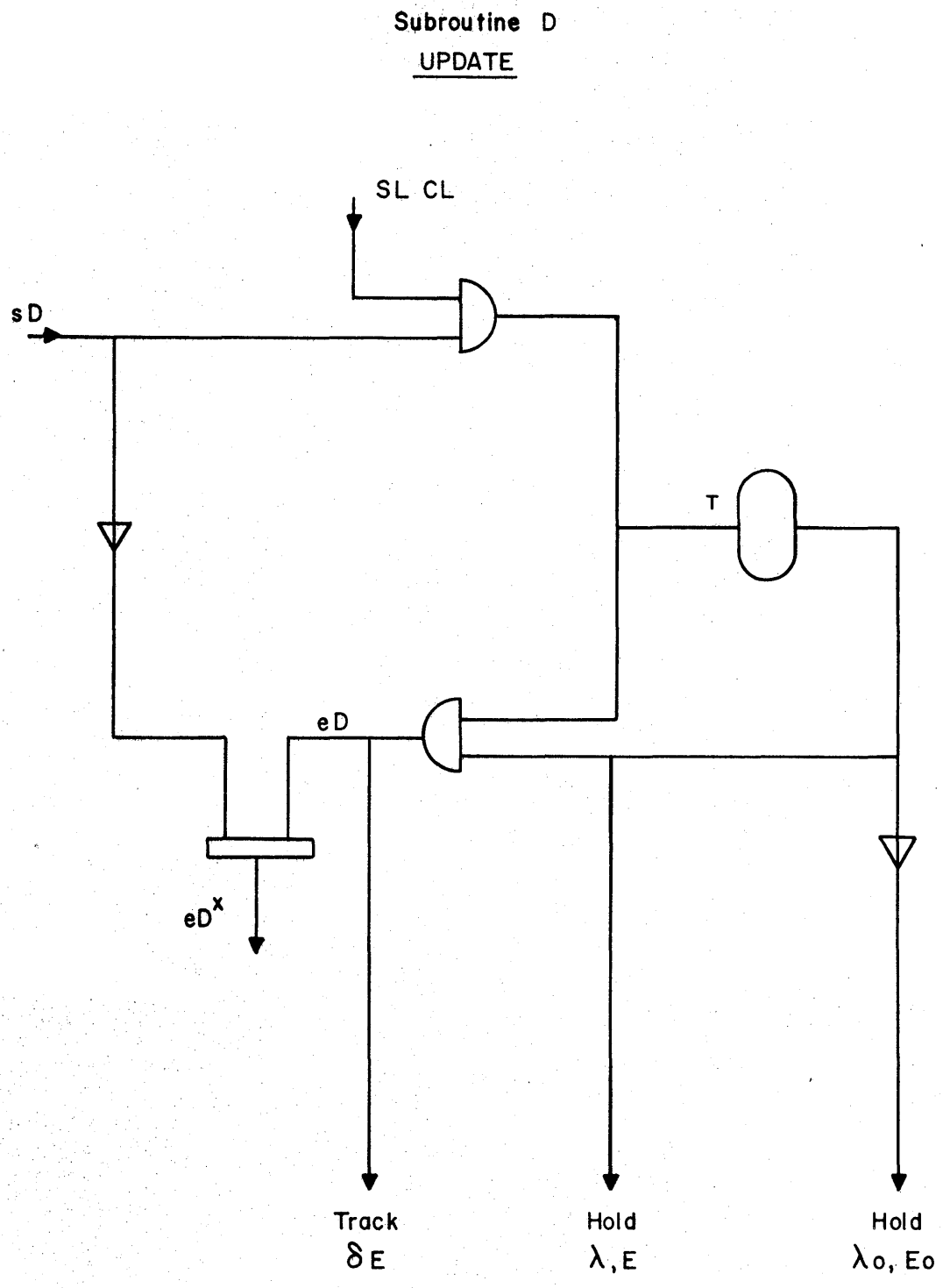


Figure 5: Subroutine D. UPDATE

Subroutine A
EXPLORE AND STORE

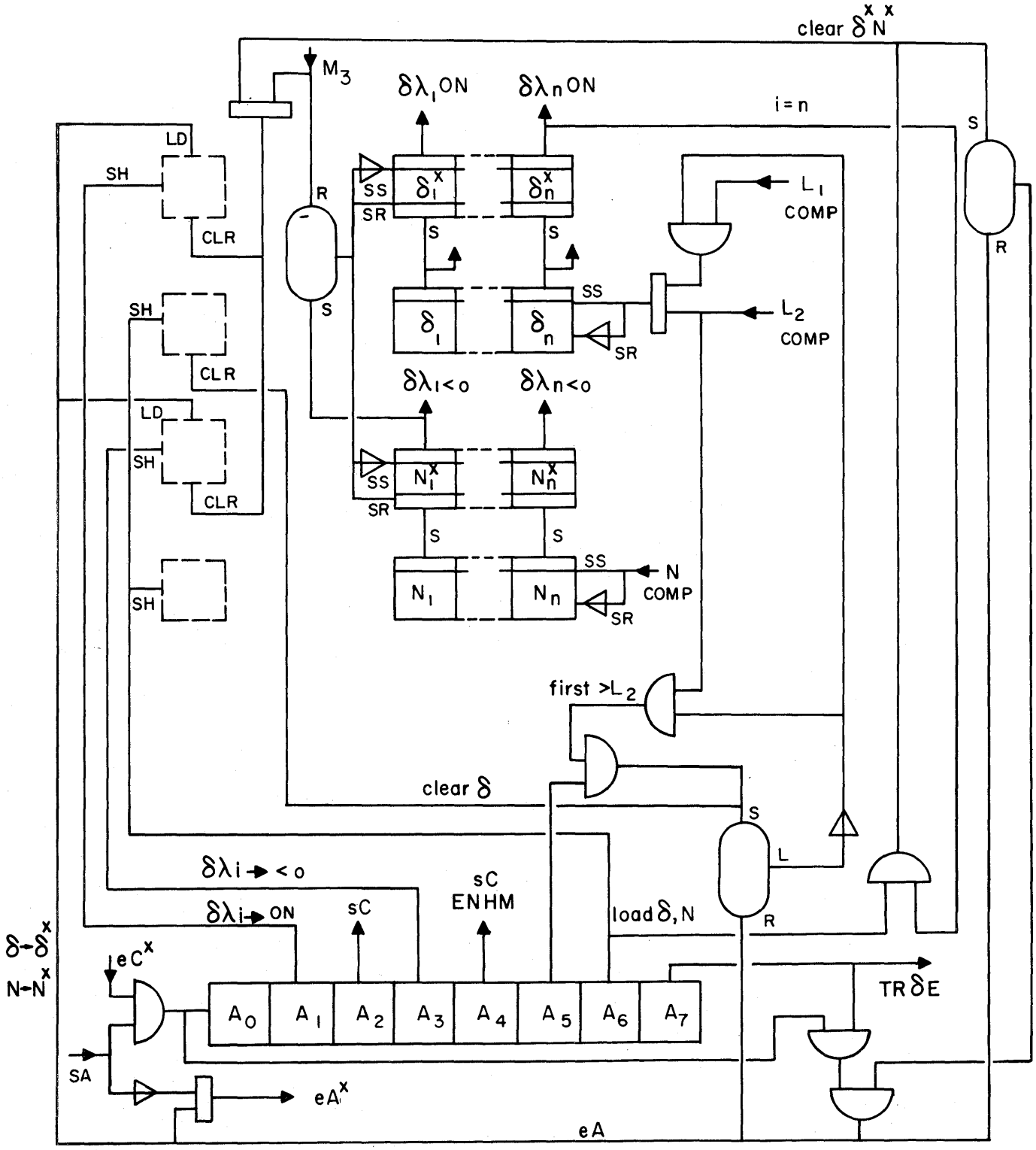


Figure 6: Subroutine A. EXPLORE

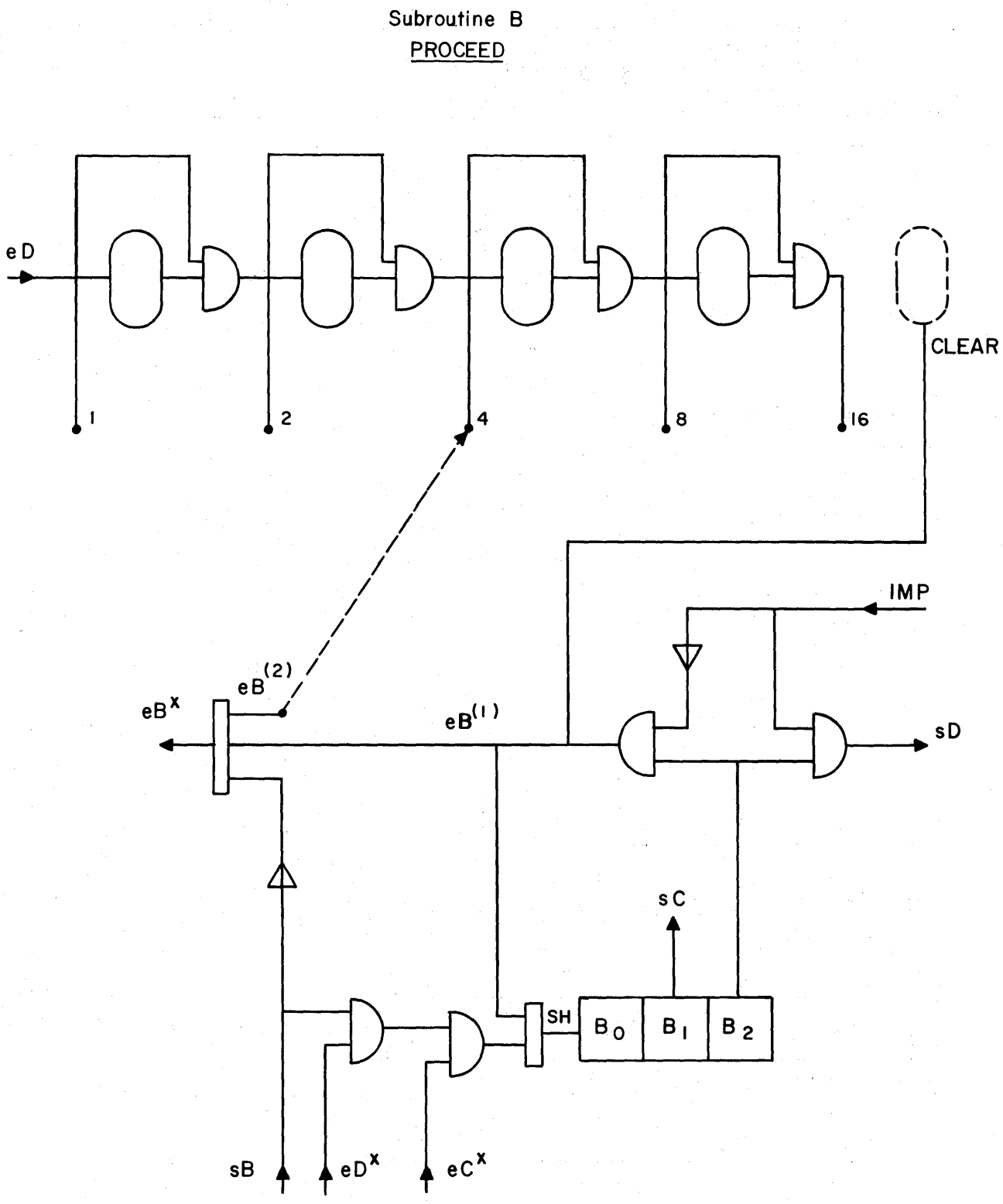


Figure 7: Subroutine B. PROCEED

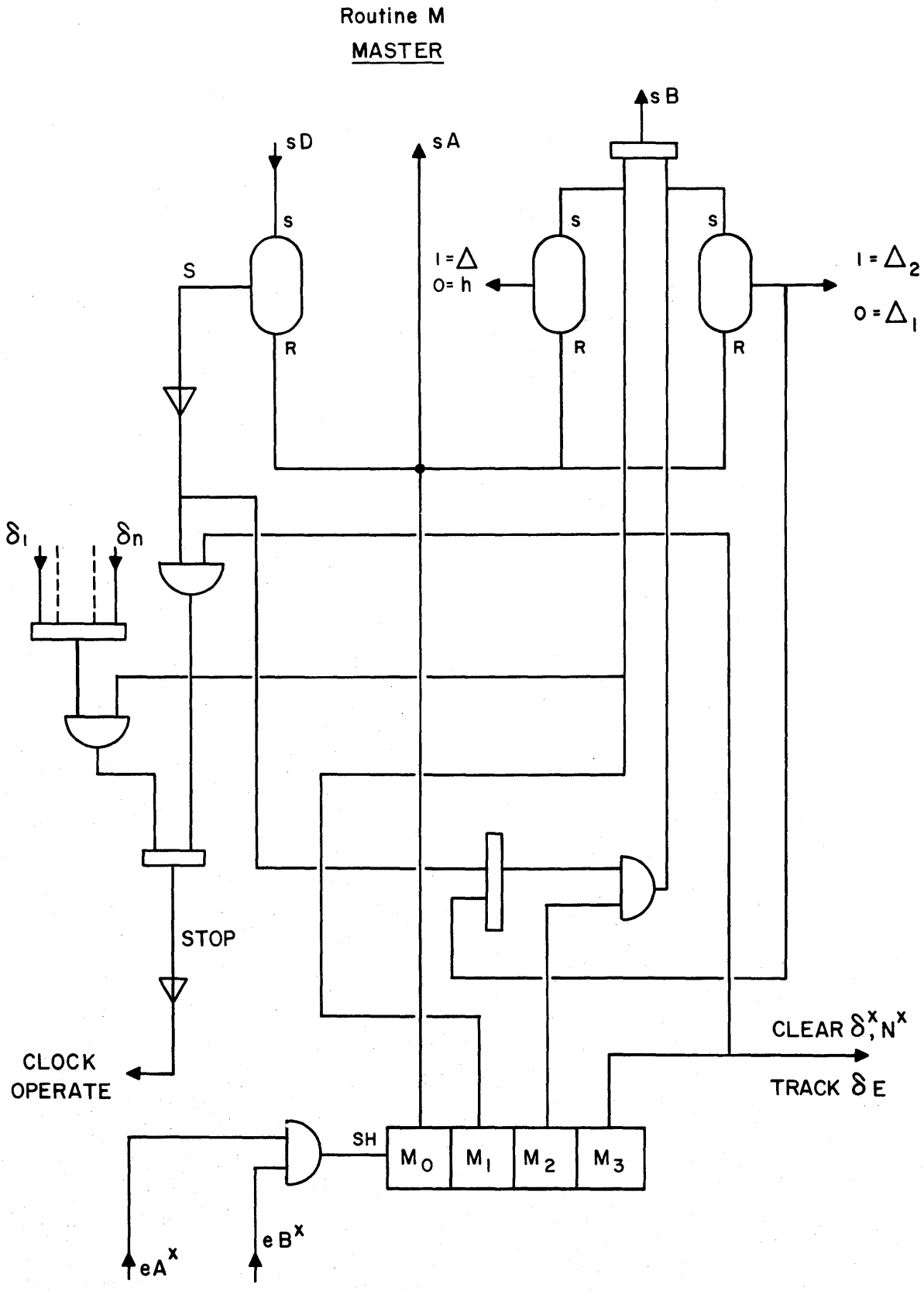


Figure 8: Routine M. MASTER