UNIVERSITY OF MARYLAND
COMPUTER SCIENCE CENTER

COLLEGE PARK, MARYLAND

Technical Report 69-86 March 1969

Design Automation by the Computer Design Language

by

Yaohan Chu
Professor

The computer time used was supported by the National Aeronautics and Space Administration under Grant NsG 398 to the Computer Science Center of the University of Maryland.

Abstract

A Computer Design Language (CDL) has been developed for facilitating design automation of digital computers. When the functional organization and sequential operation of a digital computer are conceived and specified by the CDL, this CDL description is called a Macro design. The macro design is highly descriptive in computer elements. It describes precisely and concisely what the computer is expected to do functionally step by step. It is then punched into a deck of cards. A CDL simulator accepts the deck, simulates the design, checks out the operations and outputs the contents of selected registers and memory words at every clock, every sequence or every instruction.

The macro design is then translated into a Micro design which is a set of boolean equations. This translation is called boolean translation. A boolean translator accepts the CDL deck and translates the macro design into a micro design. The micro design describes interconnections of gates, flip-flops, switches and the like as is done in a conventional logic design. The micro design is again punched on a deck of cards. The CDL simulator can also simulate and check out the micro design. The set of boolean equations can be translated by a digital computer into logic diagrams which may then be implemented by modules of logic circuits and memories. Alternatively, the set of boolean equations can be translated by a digital computer into a set of logic matrices which in turn may be implemented by large scale integration of logic circuits and memories. A change in the design as a result of evaluation of the implementation is conveniently made in the macro design with subsequent simulation and translation done automatically. Macro design, micro design, logic diagrams and logic matrices are illustrated by examples.

Design Automation by the Computer Design Language

Design Automation by the Computer Design Language

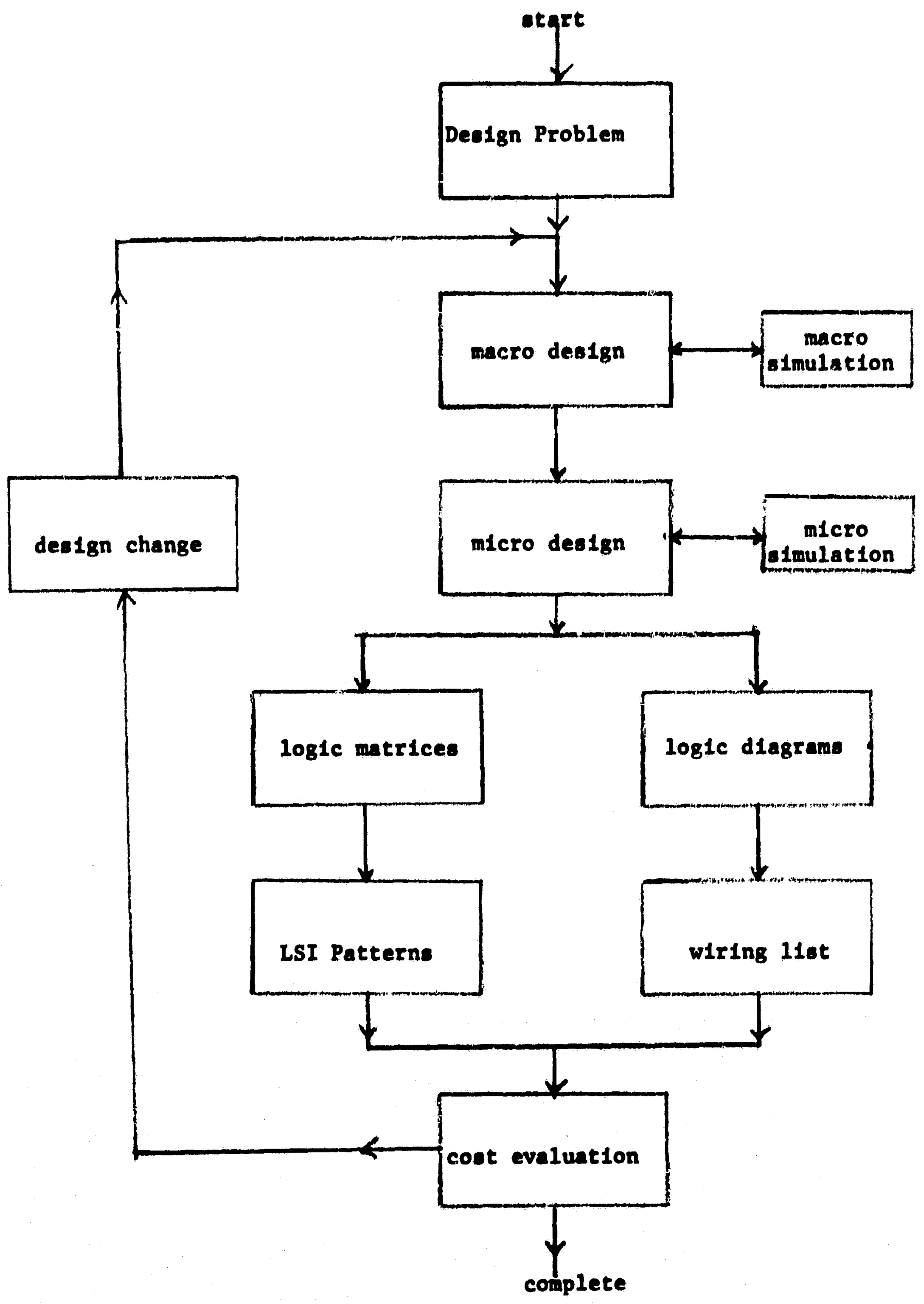
Automation, according to Webster's distionary, is the technique of making a process (or system) automatic. Design automation may simply be defined as the technique of making a design process (or a design system) automatic. The design process of a digital computer may include functional design, logic design, circuit design, wiring design, test design or LSI (large-scale integration) design and so forth. Since the advent of the digital computer, the pace of automation is quickened. It is no surprise that much effort (13) has been spent in automating various aspects of the design process of a digital computer. This paper presents the idea of design automation of a digital computer by a particular approach in using a digital computer. Indeed, it is an application of computer aided design.

1. Computer Design Automation

The approach of computer design automation presented in this paper is shown by the block diagram in Figure 1. It begins by describing and specifying the computer elements, micro-operations, control sequences and, if any, microprograms in a highly descriptive yet precise and concise language. Such a description and specification is called a macro design. When a macro design is ready, a computer program called the CDL simulator is employed to simulate the design and shows the operations of the designed computer, sequence by sequence and step by step, as a means of checking out the macro design.

Similar to the conventional logic design, the micro design is represented by a set of boolean equations. A micro design describes how the computer is interconnected from the components such as gates, flipflops and switches, but a macro design specifies what the computer is expected to do functionally. The micro design can be obtained from the macro design by translation. A computer program called the boolean translator translates the statements of a macro design into the equations of a micro design. When a micro design is ready, it can again be simulated by the above mentioned CDL simulator to check out the micro design.

Fig. 1 CDL Design Automation of Digital Computers



The boolean equations can be translated into a set of logic diagrams by a digital computer. The design may then be implemented by wiring the logic circuits, modules and memories. Alternatively, the boolean equations can be translated into a set of logic matrices, and the computer may then be implemented by large-scale integration of logic circuits and memories. In either approach, the cost of the implementation may then be evaluated.

Should a change in the design be needed as a result of cost and other evaluations, the change is made in the macro design with subsequent simulations and translations done automatically. This is the feedback which forms the design shown in Figure 1. The advantage of such an automatic design process, like the others, relieves the design engineer of many tedious and repetitive details and eliminates possible inconsistencies and errors.

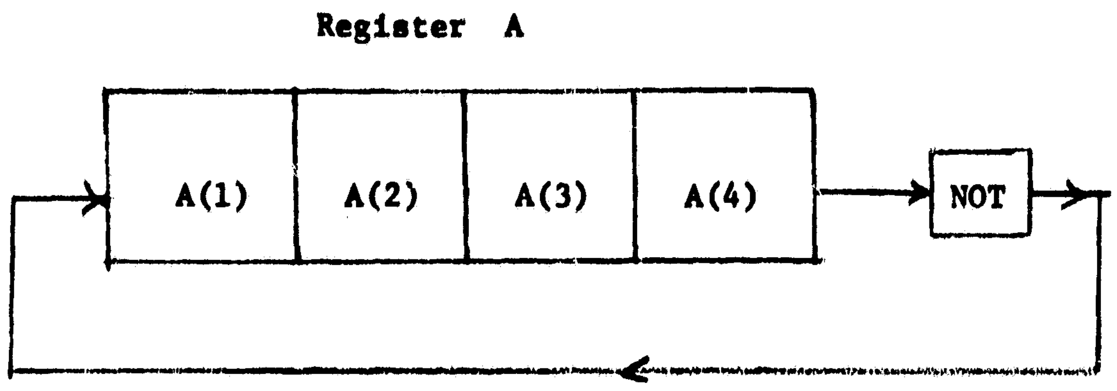
In the subsequent sections, a simple design problem is chosen. The details of various steps of the design automation are then shown in order to illustrate this particular approach of computer design automation.

2. A Design Problem

In order to show some details of macro design and micro design, a simple design problem is chosen. This problem is to design a serial complementer. This complementer serially complements every bit of a binary word stored in a shift register.

Figure 2 shows register A in which the binary word is stored. When register A is being shifted one bit to the right, the contents of bit A(4) at the right end of the register are complemented by a logical NOT block and then transferred to bit A(1) at the left end of the register. After complementing and right shifting in this manner four times, each bit of register A is complemented. An example is shown in Figure 2. Register A initially stores binary word 1111. After the first complement-shift operation, it becomes 0111. It next becomes 0011, then 0001, and finally 0000 which is the 1's complement of 1111.

Fig. 2 <u>Design Problem---A Serial Complementer</u>



Initial condition	1	1	1	1
step 1	0	1	1	1
2	0	0	1	1
3	0	0	0	1
4	0	0	0	0

3. Macro Design and Simulation

The designer of digital computers has long had a language problem. The minute details of the design, whether in diagrams or in equations, obscure the entire logic description of the computer. A similar situation once existed in programming. A program written in machine language or assembly language is too detailed to be clearly comprehended and, as a result, programming languages at higher levels have been developed. Similarly, the computer designer can profit from a design language of higher level.

3.1 Computer Design Language

A number of higher level languages have been reported during recent years (1-13, 16-19). One of them, called Computer design language (CDL), has been developed to describe the computer organization and operation (7). This language is highly descriptive. It identifies major computer elements, such as registers, decoders, switches, memories and terminals. It is precise, concise and highly expressive at the bit level, work level and bit-array level. It can express timing signals, control commands, and serial and parallel register transfers. It allows special operators and subcontrol sequences to be defined by the user. When a digital computer is specified by the CDL, the computer elements, the micro-operations, the sequences and the micro-program (if any) are all described.

3.2 Configuration

As an example of describing computer elements by the Computer Design Language, consider the configuration shown in Figure 3 for the serial complementer. Register A is the shift register where the binary word is stored. Counter C counts the number of times of shifting. Control Register T and clock P generate the control signals T(i)*P. Switch START actuates the necessary operations for initializing the sequence, and light FINI indicates the completion of the operation. This configuration can be described by the following statements of the Computer Design Language,

```
    Register,    A(1-4),          $shift register        (1)
                 T(1-3),          $control register
                 C(3-1),          $counter
    Switch       START(ON)        $start switch
    Light,       FINI(ON,OFF)     $completion indicator
    Clock,       P                $clock signal
```

The first statement is a register statement which declares the 4-bit register A, the 4-bit register T, and the 3-bit register C.

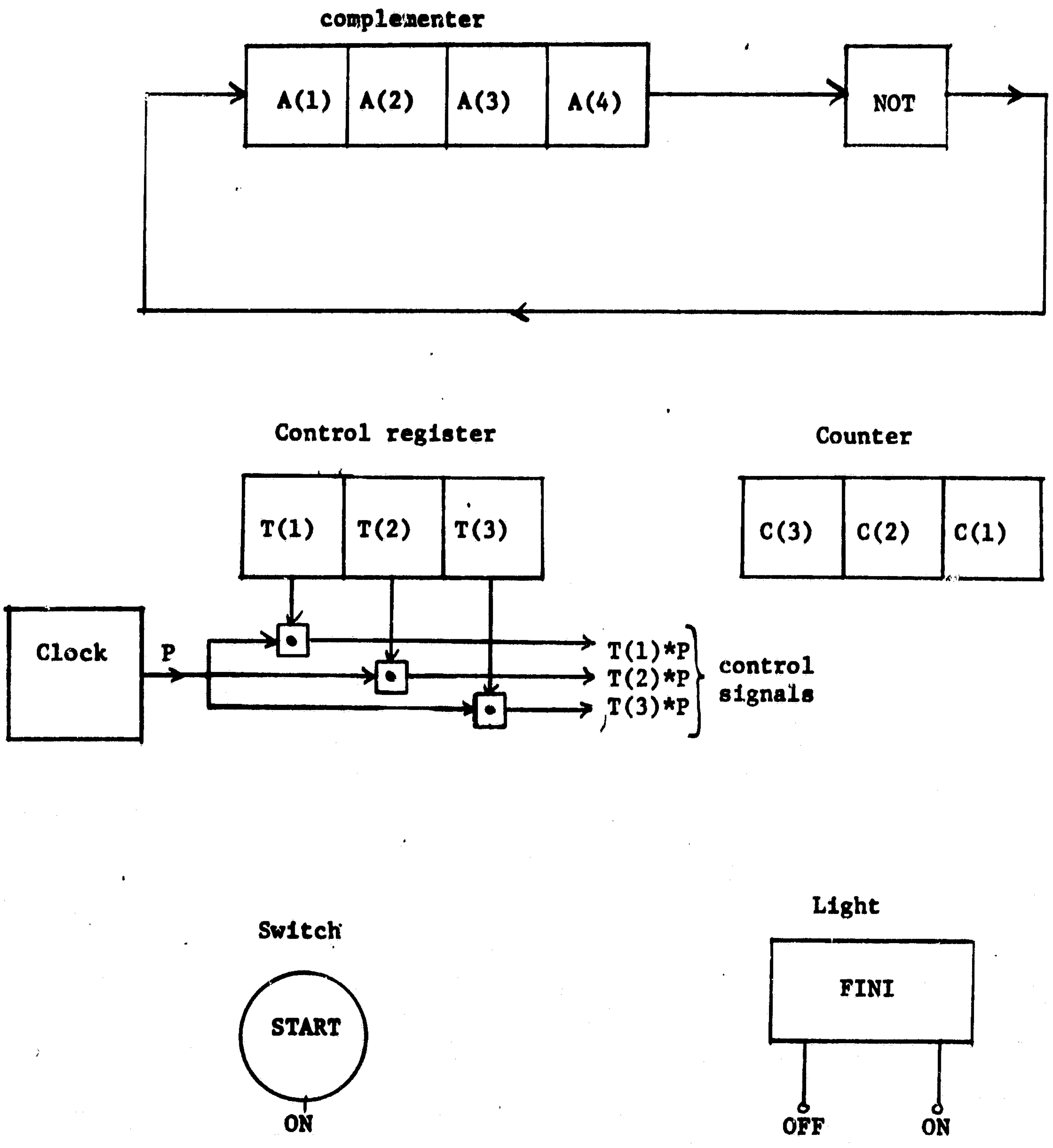


Fig. 3 Configuration of a Serial Complementer

The second statement is a switch statement which declares a single-position switch START. The third statement is a light statement which declares light FINI with two light conditions ON and OFF. The last statement declares a single-phase clock whose pulses are called P. Since these statements declare the computer elements, they are called <u>declaration statements</u>.

3.3 <u>Sequence Chart</u>

The sequential operations of the complementer are described by the <u>sequence chart</u> shown in Figure 4. As shown, when switch START is turned to the ON position, counter C is reset to 0 and light FINI is set to the OFF condition. Then, the complementing and right-shifting operation is performed and at the same time counter C is incremented by 1. Counter C is next tested for a value of 4. If the value is not 4, the complementing and right-shifting operation and counter-incrementing operation are repeated. Counter C is again tested. This process continues on until counter C reaches a value of 4; by then, each bit of register A is complemented. Light FINI is then turned to the ON condition and the sequence is terminated. Notice the loop in the sequence chart.

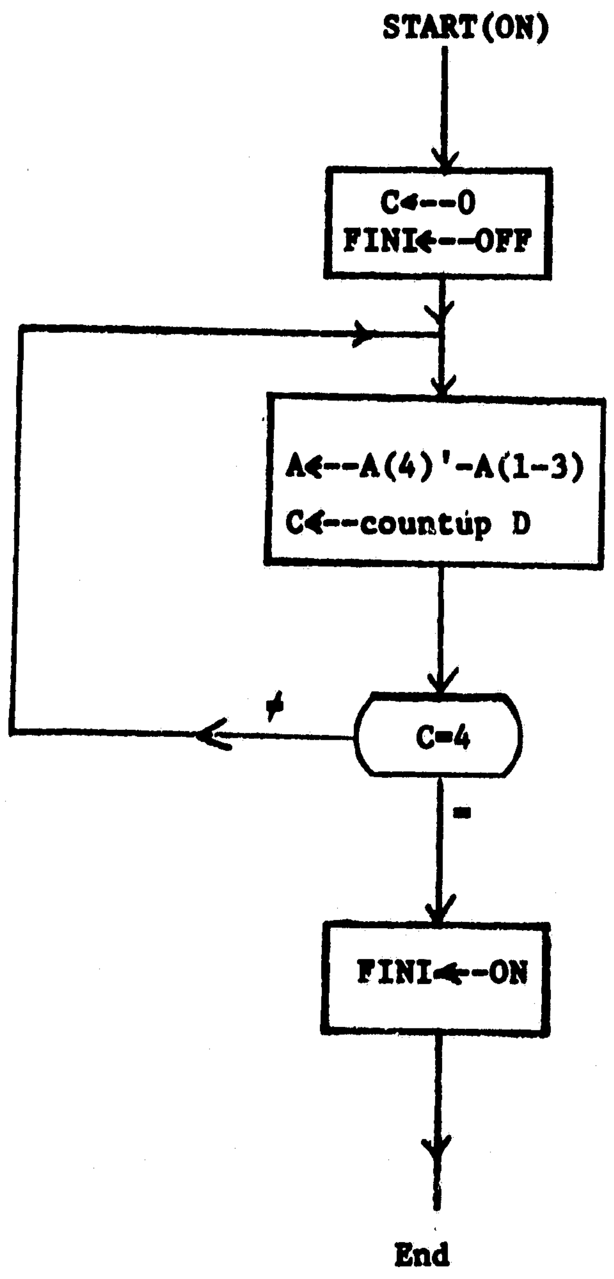


Fig. 4 Sequence chart of the Serial Complement Sequence

The following statements,

$$C \leftarrow\text{--}0 \qquad (2)$$

$$FINI \leftarrow\text{--}OFF$$

$$A \leftarrow\text{--}A(4)'\text{-}A(1\text{-}3)$$

$$C \leftarrow\text{--countup } C$$

$$FINI \leftarrow\text{--}ON$$

are taken from the chart in Figure 4. They are called _micro-statements_. Each micro-statement specifies a micro-operation. A _micro-operation_ is an elementary, functional operation that is physically built in a digital computer. Since a micro-operation performs only a simple function, a more complex function can be obtained by ordering a number of micro-operations into a sequence. The complementer is obtained by a sequence. The sequence chart describes the operations of the sequence. Similar to the flow chart, the sequence chart describes an algorithm. Likewise, similar to the software, the hardware also implements an algorithm though their restraints are different.

3.4 Statement Description

The sequencing in the sequence chart of Figure 4 is represented by lines. In the physical implementation, the sequencing is activated by a control sequence. An example is the control sequence generated by register T. The contents of register T are initially set to 100_2 by switch START; this makes the output terminal of bit T(1) as the command signal for the first step of operation. During the first step, the contents of register T are circulated one bit to the right; this makes the output terminal of bit T(2) as the command signal for the second step. If the contents of register T are again circulated during the second step, the output terminal of bit T(3) becomes the command signal for the third step. Thus, by circulating the contents of register T, the control sequence for a three-step sequence is generated.

By applying this control sequence to the sequence in Figure 4, the complement sequence can be described by the following statements of the Computer Design language.

```
/START(ON)/  C←--0,                                          (3)
             FINI←--OFF,
             T←--100,
/T(1)*P/     A←--A(4)'-A(1-3),
             C←--countup C,
             T(1-2)←--01
/T(2)*P/     IF (C=4) THEN (T(2,3)←--ol) ELSE (T(1,2)←--10)
/T(3)*P/     FINI←--ON
             END
```

The above statements are called execution statements except the last statement which is the end statement. Each execution statement consists of one or more micro-statements and a label. The label, the quantity enclosed by a pair of slashes, represents the control signal. The label is a logical quantity. When its value is true, the associated micro-statements are executed; otherwise, they are ignored. This is the manner that execution of parallel micro-operations is described. The following statements,

```
T←--100,            (4)
T(1,2)←--01,
T(2,3)←--01,
T(1,2)←--10
```

taken from the above statements (3), are also micro-statements which circulate the contents of register T or set the contents of register T to a particular value. The statement.

```
IF (C=4) THEN (T(2,3)←--01) ELSE (T(1,2)←--10)     (5)
```

is called a conditional micro-statement. The above conditional micro-statement is of the IF-THEN-ELSE type and can be replaced by the two conditional micro-statements of the IF-THEN type as shown below,

```
IF (C=4) THEN (T(2,3)<--01),          (6)
IF (C≠4) THEN (T(1,2)<--10)
```

the conditional micro-statement (5) represents the test block in the sequence chart in Figure 4.

The declaration statements in statements (1) and the executioh statements in statements (3) completely and precisely specify the complement sequence and thus constitute the macro design of the serial complementer.

3.5 Macro Simulation

A CDL simulator has been developed. It consists of two parts, a translator program and a simulator program. The translator program accepts a description in the CDL punched on a deck of cards, translates it into a program called "polish string", and establishes varous tables and a storage array. The simulator program consists of five parts; loader, output routine, switch routine, simulate routine and reset routine. The loader accepts test data from punched cards and stores them into the simulated memories and registers of the CDL described computer. The output routine handles the printout of the contents of the chosen registers and the memory words and the positions of the switches during the simulation. The switch routine simulates the operation of manual switches. The simulate routine executes the Polish String in an interpretive mode. The reset routine reinitializes the simulator program for a next simulation run.

Execution of the Polish String by the simulate routine is carried out in a control loop, called Label Cycle. During a Label Cycle, the following processing is performed. (a) If a manual switch operation has occurred, the micro-statements of the execution statement with the switch as the label or a part of are executed. (b) Labels of all execution statements are evaluated. Those labels which are activated are noted. Activated labels are those whose logical values are 1. (c) The micro-statements of those execution statements with the activated labels are executed. (d) Condition for simulation termination is checked. If the condition is fulfilled, the run is terminated; otherwise, it proceeds to the next label cycle.

Version 1 of the CDL simulator has been available since the Summer of 1967; this version allows a limited set of the Computer Design Language (15). Version 2 has been available since February, 1968; this version implements most of the features of the language (20). Version 2 was further improved to become version 3. Version 3 has been available since the Fall of 1968. All the three versions were written in Fortran IV with several routines in assembly language MAP for the IBM 7090 family of computers.

The simulation deck consists of three types of punched cards: system control cards, CDL statement cards, and simulation control cards. An example of a simulation deck is shown in Figure 5. The system control cards are those for the user to communicate with the operating system of a computer installation. The first ten and the last three cards in Figure 5 are the system control cards. The CDL statement cards constitute the description of a sequence or a computer to be simulated. The 13th through 30th cards in Figure 5 are the CDL statement cards. The simulation control cards are those for the user to communicate with the CDL Simulator. The 11th, 12th, 31st through 40th cards in Figure 5 are the simulation cards. There are eight types of simulation control cards: heading, load, output, switch, simulate, reset, data and call-simulator-program.

The statements (1) and (3) which describe the serial complement sequence are punched into a deck of cards shown in Figure 5 with the following exceptions. Light FINI and operator countup are replaced respectively by operator COUNT and register FINI. And Comment statements (those with letter C in column 1) are added for better readability. As mentioned, the 31st through 40th cards are simulation control cards. The 31st $SIMULATE card calls the simulator program. The 32nd through 36th cards are those simulation control cards for the first simulation run. The 32nd *OUTPUT card specifies that the contents of registers A, T, C and FINI be printed out at every clock cycle. The 33rd *SWITCH card simulates the ON position of the START switch. The 34th *LOAD card loads the octal number on the 35th data card into register A. The 36th *SIM card specifies that simulation run be terminated at the end of 30 Label cycles or when three consecutive Label Cycles with a group of repeatedly activated labels occur. The 37th through 40th cards are those for the second simulation run. The 37th *RESET card reinitializes the simulator program. The 38th through 40th cards are similar to those explained above.

The output of the second simulation run of the description in Figure 5 is shown in Figure 6 where the label cycle becomes the clock cycle. Notice that the initial value of register

```
$IBSYS
$*      MOUNT TAPE 1608 ON A9, RING OUT AND SAVE
$*      THANK YOU
$PAUSE
$ATTACH             A9
$AS                 SYSLB4
$REWIND             SYSLB4
$EXECUTE            USER
$ID         CHU   *001/01/125*2M*100P*T$
$CDL3
$TRANSLATE
$MAIN
C
COMMENT    ****A SERIAL COMPLEMENT SEQUENCE
C
 REGISTER,            A(1-4),
1                     T(1-3),
1                     C(3-1),
1                     FINI
 SWITCH,              START(ON)
 CLOCK,               P
 /START(ON)/        T=4,
                    FINI=0,
                    C=0
 /T(1)*P/           A(4-1)=A(1)'-A(4-2),
                    C=C.COUNT.,
                    T(1,2)=1
 /T(2)*P/           IF (C.EQ.4) THEN (T(2,3)=1) ELSE (T(1,2)=2)
 /T(3)*P/           FINI=1
                    END
$SIMULATE
*OUTPUT    CLOCK(1)=A,T,C,FINI
*SWITCH    1,START=ON
*LOAD
      A=16
*SIM       30,3
*RESET     CYCLE
*LOAD
      A=05
*SIM       30,3
'
$IBSYS
$RESTORE
```

Fig. 5, A listing of a CDL simulation deck for macro simulation

OUTPUT OF SIMULATION

```
SWITCH INTERRUPT
    START  = ON
        A = ....05          T = .....4          C = .....0      FINI = .....0
*******************************************************************
LABEL CYCLE     1                 TRUE LABELS                CLOCK TIME =     1
                                /T(1)*P/
        A = ....02          T = .....2          C = .....1      FINI = .....0
*******************************************************************
LABEL CYCLE     2                 TRUE LABELS                CLOCK TIME =     2
                                /T(2)*P/
        A = ....02          T = .....4          C = .....1      FINI = .....0
*******************************************************************
LABEL CYCLE     3                 TRUE LABELS                CLOCK TIME =     3
                                /T(1)*P/
        A = ....21          T = .....2          C = .....2      FINI = .....0
*******************************************************************
LABEL CYCLE     4                 TRUE LABELS                CLOCK TIME =     4
                                /T(2)*P/
        A = ....21          T = .....4          C = .....2      FINI = .....0
*******************************************************************
LABEL CYCLE     5                 TRUE LABELS                CLOCK TIME =     5
                                /T(1)*P/
        A = ....10          T = .....2          C = .....3      FINI = .....0
*******************************************************************
LABEL CYCLE     6                 TRUE LABELS                CLOCK TIME =     6
                                /T(2)*P/
        A = ....10          T = .....4          C = .....3      FINI = .....0
*******************************************************************
LABEL CYCLE     7                 TRUE LABELS                CLOCK TIME =     7
                                /T(1)*P/
        A = ....24          T = .....2          C = .....4      FINI = .....0
*******************************************************************
LABEL CYCLE     8                 TRUE LABELS                CLOCK TIME =     8
                                /T(2)*P/
        A = ....24          T = .....4          C = .....4      FINI = .....0
*******************************************************************
LABEL CYCLE     9                 TRUE LABELS                CLOCK TIME =     9
                                /T(1)*P/
        A = ....32          T = .....2          C = .....5      FINI = .....0
*******************************************************************
LABEL CYCLE    10                 TRUE LABELS                CLOCK TIME =    10
                                /T(2)*P/
        A = ....32          T = .....1          C = .....5      FINI = .....0
*******************************************************************
LABEL CYCLE    11                 TRUE LABELS                CLOCK TIME =    11
                                /T(3)*P/
        A = ....32          T = .....1          C = .....5      FINI = .....1
*******************************************************************
LABEL CYCLE    12                 TRUE LABELS                CLOCK TIME =    12
                                /T(3)*P/
        A = ....32          T = .....1          C = .....5      FINI = .....1
*******************************************************************
LABEL CYCLE    13                 TRUE LABELS                CLOCK TIME =    13
                                /T(3)*P/
        A = ....32          T = .....1          C = .....5      FINI = .....1
*******************************************************************
```

Fig. 6, Output of the simulation

($A=05_8$) is the value on the data card of the second simulation run, and the final value of register A ($A=32_8$) is shown in the last line in Figure 6. The simulated result is correct because octal number 32 is the 1's complement of octal number 05.

4. Micro Design and Simulation

Micro design is the conventional logic design described by a set of boolean equations. The micro design specifies how the gates, flipflops, switches and the like are interconnected. Because of its minute details, the micro design is difficult for anyone to understand and often fraught with errors.

Since the macro design completely and precisely specifies the functional organization and sequential operations, it would be most desirable to translate the macro design into a micro design by the digital computer. Such a translator is called a boolean translator. A boolean translator which accepts a design in the Computer Design Language is partially available (20). As an example, the macro design of the complement sequence described in statements (1) and (3) is now translated into a micro design. Assume that RS flipflops are chosen for the registers.

4.1 State Equations

The micro design consists of state equations and input equations. The state equations describe those storage elements that are translated from the register and light statements. The input equations describe the logic networks. The register and light statements of the complement sequence are, Register, A(1-4), T(1-3), C(3-1) (7)

Light FINI(ON,OFF)

The state equations for registers A, T, and C are,

$$A(1\text{-}4)=A_r(1\text{-}4)'*A(1\text{-}4)+A_s(1\text{-}4), \quad (8)$$
$$T(1\text{-}3)=T_r(1\text{-}3)'*T(1\text{-}3)+T_s(1\text{-}3),$$
$$C(1\text{-}2)=C_r(1\text{-}2)'*C(1\text{-}2)+C_s(1\text{-}2)$$

where A_r, T_r and C_r are the reset inputs and A_s, T_s and C_s are the set inputs of the flipflops of registers A, T and C. The state equation for light FINI is,

$$\text{FINI(ON)=FINIOFF'*FINI(ON)+FINION} \quad (9)$$

where FINIOFF and FINION are the inputs of light FINI by which the light is turned to the OFF and ON positions respectively.

4.2 Input Equations

Input equations describe the logic networks specified by the terminal and decoder statements, by the basic operators such as countup, by the micro-statements and by the labels. Since the clock and the switches in the clock and switch statements are input devices, no translation is needed.

Input equations are grouped according to each register or each light. There is only one execution statement that changes the contents of register A. This statement is,

$$/T(1)*P/ \qquad A \leftarrow\!\!-\!- A(4)'\text{-}A(1\text{-}3) \qquad (10)$$

The input equations for register A translated from the above statement are shown below,

$$A1_r = A(4)*T(1)*P \qquad (11)$$
$$A1_s = A(4)'*T(1)*P$$
$$A2_r = A(1)*T(1)*P$$
$$A2_s = A(1)'*T(1)*P$$
$$A3_r = A(2)*T(1)*P$$
$$A3_s = A(2)'*T(1)*P$$
$$A4_r = A(3)*T(1)*P$$
$$A4_s = A(3)*T(1)*P$$

where $A1_r$, $A2_r$, $A3_r$ and $A4_r$ are the reset inputs and $A1_s$, $A2_s$, $A3_s$ and $A4_s$ are the set inputs of the flipflops of register A. The above 1st, 3rd, 5th and 7th input equations are obtained from the conditions that bits A(4), A(1), A(2) and A(3) are reset to 0 respectively, while the 2nd, 4th, 6th and 8th input equations from the conditions that these bits are set to 1 respectively.

There are two execution statements which change the contents of register C(3-1),

$$/\mathrm{START(ON)}/ \quad C \leftarrow\!\!-\!- 0, \qquad (12)$$

$$/\mathrm{T(1)*P}/ \quad C \leftarrow\!\!-\!- \mathrm{countup}\ C$$

The input equations for register C translated from these two execution statements are,

$$\begin{aligned} C1_r &= C(1)*T(1)*P+START(ON) \qquad (13)\\ C1_s &= C(1)'*T(1)*P\\ C2_r &= C(2)*C(1)*T(1)*1+START(ON)\\ C2_s &= C(2)'*C(1)*T(1)*P\\ C3_r &= C(3)*C(2)*C(1)*T(1)*P+START(ON)\\ C3_s &= C(3)'C(2)*C(1)*T(1)*P \end{aligned}$$

where $C1_r$, $C2_r$ and $C3_r$ are the reset inputs and $C1_s$, $C2_s$ and $C3_s$ are the set inputs of the flipflops of register C. The above 1st, 3rd and 5th input equations contain term START(ON) which reflects the turning of switch START to the ON position; the other terms of these and the remaining input equations specify the count micro-operation.

There are four execution statements which change the contents of register T(1-3),

```
/START(ON)/   T<--100,                              (14)
/T(1)*P/      T(1,2)<--01,
/T(2)*P/      IF (C=4) THEN (T(2,3)<--01),
/T(2)*P/      IF (C≠4) THEN (T(1,2)<--10),
```

For ease of translation, the above third and fourth statements are decomposed from the IF-THEN-ELSE conditional micro-statement in statements (3). The input equations for register T translated from these four statements are,

$$T1_r = T(1)*P \qquad (15)$$

$$T1_s = (C(3)*C(2)'*C(1)')'*T(2)*P+START(ON),$$

$$T2_r = T(2)*P+START(ON),$$

$$T2_s = T(1)*P,$$

$$T3_r = START(ON),$$

$$T3_s = C(3)*C(2)'*C(1)'*T(2)*P,$$

where $T1_r$, $T2_r$ and $T3_r$ are the reset inputs and $T1_s$, $T2_s$ and $T3_s$ are the set inputs of the flipflops of register T respectively. Factor C(3)*C(2)'*C(1)' represents the condition that register C contains 4.

There are two execution statements that change the condition of light FINI,

```
/START(ON)    FINI<--OFF,          (16)
/T(3)*P/      FINI<--ON,
```

The input equations for light FINI are,

```
FINIOFF = START(ON),               (17)
FINION = T(3)*P,
```

where FINIOFF and FINION are the inputs to turn the light to the OFF and ON conditions.

Equations (8), (9), (11), (13), (15) and (17) constitute the set of the boolean equations for the complement sequence.

4.3 Micro Simulation

The set of the boolean equations for the complement sequence can be simulated by the previously described CDL Simulator. A listing which lists the deck of cards for the micro simulation is shown in Figure 7. As shown in Figure 7, the boolean equations become the terminal statements, and the state equations become the execution statement. It is possible to have the simulation partially in macro simulation and partially in micro simulation. To illustrate this type of

```
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
C
[illegible]
C
 [illegible]  A(1-4),
1             T(1-3),
1             C(3-1),
1             FINI
 [illegible]  [illegible]
 [illegible]  P
C
COMMENT***HERE BEGIN THE INPUT [illegible]
C
 TERMINAL, A1R=A(4) *T(1),
1             A1S=A(4)'*T(1),
1             A2R=A(1) *T(1),
1             A2S=A(1)'*T(1),
1             A3R=A(2) *T(1),
1             A3S=A(2)'*T(1),
1             A4R=A(3) *T(1),
1             A4S=A(3)'*T(1)
C
 TERMINAL, C1R=C(1) *T(1),
1             C1S=C(1)'*T(1),
1             C2R=C(2) *C(1)*T(1),
1             C2S=C(2)'*C(1)*T(1),
1              C3R=C(3) *C(2)*C(1)*T(1),
1             C3S=C(3)'*C(2)*C(1)*T(1)
C
 TERMINAL, T1R=T(1),
1             T1S=T(2)*(C(3)*C(2)'*C(1)'))',
1             T2R=T(2),
1             T2S=T(1),
1             T3R=0,
1             T3S=T(2)*C(3)*C(2)'*C(1)'
C
 TERMINAL, FINION=T(3)*P
C
 TERMINAL, TR(1-3)=T1R-T2R-T3R,
1             TS(1-3)=T1S-T2S-T3S,
1             CR(3-1)=C3R-C2R-C1R,
1             CS(3-1)=C3S-C2S-C1S,
1             AR(1-4)=A1R-A2R-A3R-A4R,
1             AS(1-4)=A1S-A2S-A3S-A4S
C
COMMENT***HERE BEGINS MANUAL OPERATION
C
 /START(ON)/  T=4,
                C=0,
                FINI=0
```

Fig. 7, A listing of a CDL simulation deck for micro simulation

```
[illegible] WITH THE STATE EQUATIONS

[illegible]        [illegible]
                    [illegible]
                    [illegible]
                    [illegible]
                 [illegible]
[illegible]
[illegible]              (1)=[illegible] INT
[illegible]        [illegible]
[illegible]
      [illegible]
[illegible]        [illegible]
[illegible]        STEEL
[illegible]
      [illegible]
[illegible]        [illegible]

[illegible]
[illegible]
```

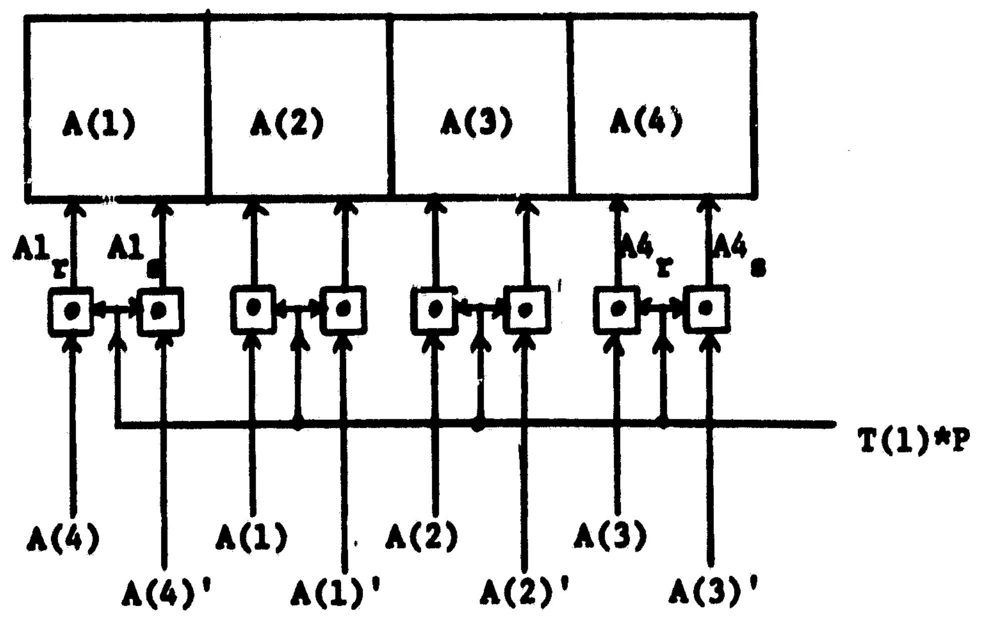
Fig. 7 (continued)

simulation, the simulation of the micro-operations actuated by the START switch in Figure 7 is macro simulation, while the remaining simulation is micro simulation.

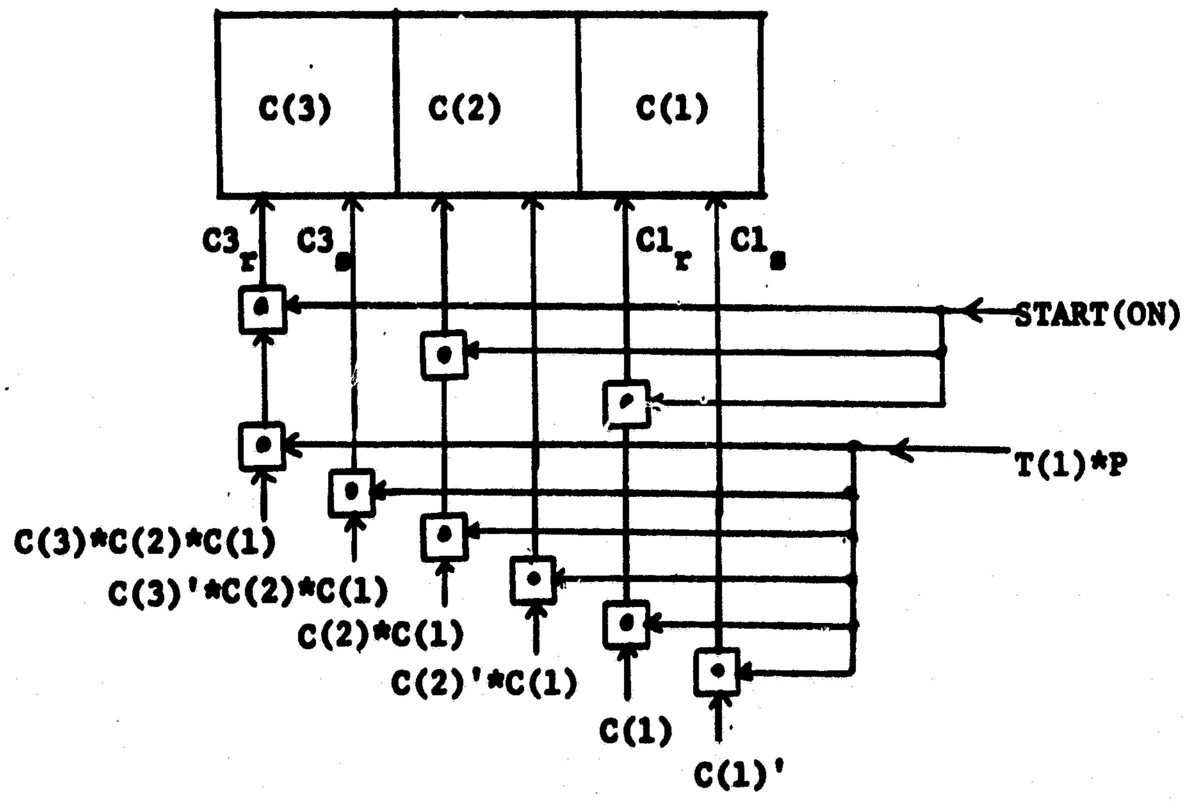
Both the system control cards and simulation control cards in Figure 7 are identical to those in Figure 5. The output of the simulation from the deck in Figure 7 is identical to the output shown in Figure 6 as it should be.

4.4 Logic Diagrams

The set of the boolean equations can be translated into logic diagrams by a digital computer. The logic diagrams may then be implemented by logic circuits and memories. The logic diagrams for the complement sequence are shown in Figure 8. In Figure 8, large squares represent the flipflops; each small square with a dot inside the square represents a logical AND circuit and each small square with a plus sign represents a logical OR circuit. The inputs of the flipflops are shown, while their outputs are not as they are apparent. The execution statements implemented by each logic diagram are also shown in Figure 8.

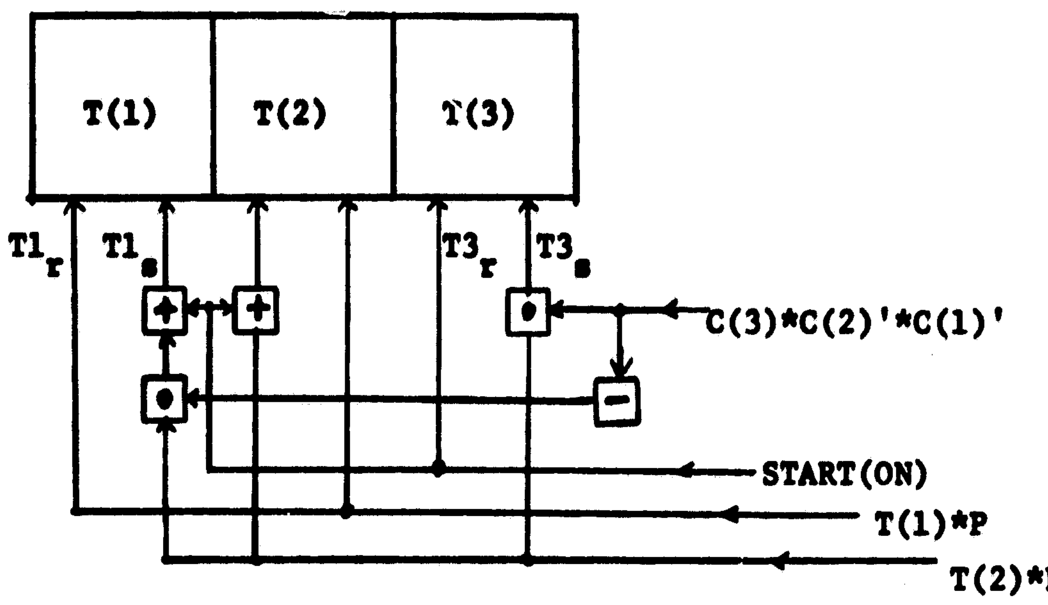


/T(1)*P/ A←-A(4)'-A(1-3)



/START(ON)/ C←-0

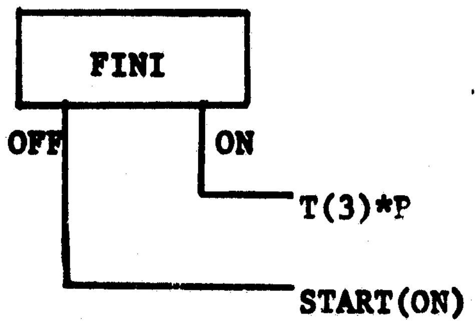
/T(1)*P/ C←-countup C



```
/START(ON)/    T<--100

/T(1)*P/       T(1,2)<--01

/T(2)*P/       IF (C=4) THEN (T(2,3)<--01) ELSE (T(1,2)<--10)
```



```
/START(ON)/  FINI<--OFF

/T(3)*P/     FINI<--ON
```

5. Logic Matrices

Boolean equations may alternatively be represented by boolean matrices. This approach gives a functional representation in the matrix form which can readily be implemented by logic matrices. Therefore, description of a micro design by boolean matrices gives a simultaneous realization of logic design and circuit implementation. As will be indicated, such a description is amenable to implementation by large scale integration.

5.1 Boolean Matrices

A boolean matrix can represent one micro-statement together with the associated control signal. Therefore, representation of a micro design by boolean matrices is a functional representation. The boolean matrices for the micro-statements of the complement sequence are now presented.

The execution statement which describes the count-C micro-operation is,

$$/T(1)*P/ \quad C \leftarrow \text{countup } C \qquad (18)$$

This statement can be expressed by the following matrix equation,

$$\begin{vmatrix} C3_r \\ C3_s \\ C2_r \\ C2_s \\ C1_r \\ C1_s \end{vmatrix} = \begin{vmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 \end{vmatrix} * \begin{vmatrix} T(1)*P \\ C(3)' \\ C(3) \\ C(2)' \\ C(2) \\ C(1)' \\ C(1) \end{vmatrix} \qquad (19)$$

The above matrix equation represents a logic network. The outputs of the network are described by the vector on the left side of the equation; this vector represents the inputs of register C. The inputs of the network are described by the vector on the right side of the equation; this vector represents the control signal T(1)*P and the outputs of the flip-flops of register C. The matrix, having only 1's and 0's, is operated by the operator * and the input vector. The operation is performed according to the rule as illustrated below for the first row,

$$C3_r=(0+T(1)*P)*(1+C(3)')*(0+C(3))*(1+C(2)')*(0+C(2))*(1+C(1)')*C\,(0+C(1)) \qquad (20)$$

which is the product of the first element of the first row by the first element of the input vector and of the second element of the first row by the second element of the input vector and so forth. Equation (20) can be simplified into,

$$C3_r = T(1)*P*C(3)*C(2)*C(1) \qquad (21)$$

Equation (21) is identical to the input equation for $C3_r$ in statement (13). By expanding the other rows of the boolean matrix in the similar manner, one obtains the input equations (13) for register C (except the terms START(ON)).

The execution statement which describes the complement-shift micro-operation is,

$$/T(1)*P/ \quad A \leftarrow A(4)'-A(1\text{-}3) \qquad (22)$$

This statement can be expressed by the following matrix equation,

$$\begin{vmatrix} A1_r \\ A1_s \\ A2_r \\ A2_s \\ A3_r \\ A3_s \\ A4_r \\ A4_s \end{vmatrix} = \begin{vmatrix} 0&1&1&1&1&1&1&1&0 \\ 0&1&1&1&1&1&1&0&1 \\ 0&1&0&1&1&1&1&1&1 \\ 0&0&1&1&1&1&1&1&1 \\ 0&1&1&1&0&1&1&1&1 \\ 0&1&1&0&1&1&1&1&1 \\ 0&1&1&1&1&1&0&1&1 \\ 0&1&1&1&1&0&1&1&1 \end{vmatrix} * \begin{vmatrix} T(1)*P \\ A(1)' \\ A(1) \\ A(2)' \\ A(2) \\ A(3)' \\ A(3) \\ A(4)' \\ A(4) \end{vmatrix} \qquad (23)$$

Again, this matrix equation describes a logic network, the inputs of which are the control signal T(1)*P and the outputs of the flipflops of register A and the outputs of which are the inputs of the flipflops of register A. When the above boolean matrix is expanded according to the manner illustrated by equation (20), one obtains the input equations (11).

The execution statements which describe the control sequence are,

```
/START(ON)/  T<--100,
/T(1)*P/     T(1,2)<--01,                                          (24)
/T(2)*P/     IF (C=4) THEN (T(2,3)<--01) ELSE (T(1,2)<--10)
```

These statements can be expressed by the following matrix equation,

$$\begin{vmatrix} T1_r \\ T1_s \\ T2_r \\ T2_s \\ T3_r \\ T3_s \end{vmatrix} = \begin{vmatrix} 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \end{vmatrix} * \begin{vmatrix} P \\ T(1)' \\ T(1) \\ T(2)' \\ T(2) \\ T(3)' \\ T(3) \\ T4' \\ T4 \end{vmatrix} \qquad (25)$$

where $T4 = C(3)*C(2)'*C(1)'$

Expansion of the above matrix gives the input equations (15) for register T (except the START(ON) terms).

The execution statement which describes the reset-C micro-operation is,

$$/START(ON)/ \quad C \leftarrow\!\!-\!-0, \qquad (26)$$

This statement can be expressed by the following matrix equation,

$$\begin{vmatrix} C3_r \\ C3_s \\ C2_r \\ C2_s \\ C1_r \\ C1_s \end{vmatrix} = \begin{vmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{vmatrix} * \left| START(ON) \right| \qquad (27)$$

The execution statement which describes the set-T micro-operation is,

$$/START(ON)/ \quad T \leftarrow\!\!-\!-100 \qquad (28)$$

This statement can be expressed by the following matrix equation,

$$\begin{vmatrix} T1_r \\ T1_s \\ T2_r \\ T2_s \\ T3_r \\ T3_s \end{vmatrix} = \begin{vmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{vmatrix} * \begin{vmatrix} START(ON) \end{vmatrix} \qquad (29)$$

The execution statements which describe the reset-FINI and set-FINI micro-operations are,

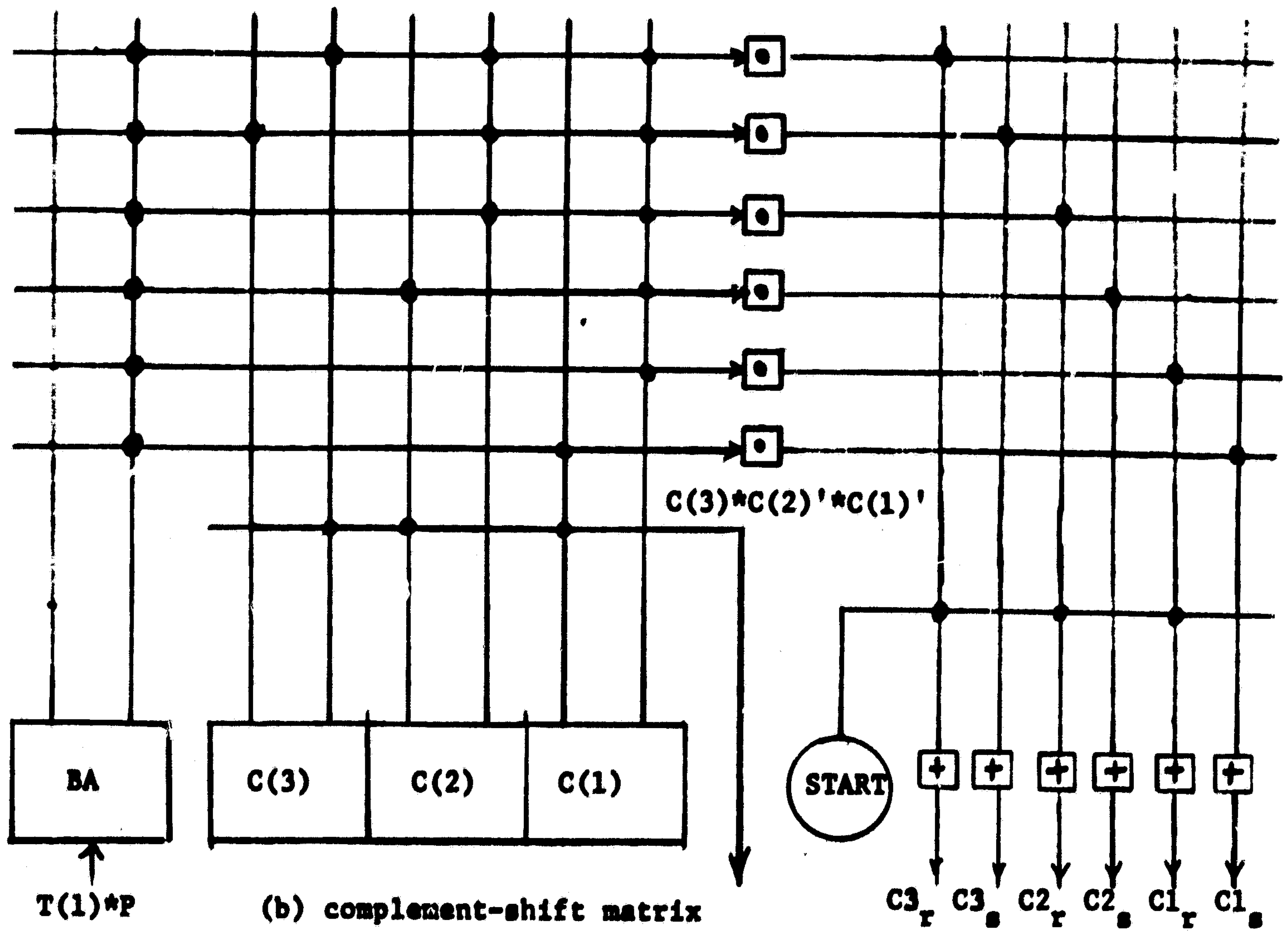
$$\begin{array}{ll} /START(ON)/ & FINI\leftarrow\text{---}OFF, \\ /T(3)*P/ & FINI\leftarrow\text{---}ON, \end{array} \qquad (30)$$

These statements can be expressed by the following matrix equation,

$$\begin{vmatrix} FINIOFF \\ FINION \end{vmatrix} = \begin{vmatrix} 0 & 1 \\ 1 & 0 \end{vmatrix} * \begin{vmatrix} START(ON) \\ T(3)*P \end{vmatrix} \qquad (31)$$

Matrix Eqautions (19), (23), (25), (27), (29) and (31) give another form of the micro design for the complement sequence.

(a) count-C matrix and reset-C matrix



(b) complement-shift matrix

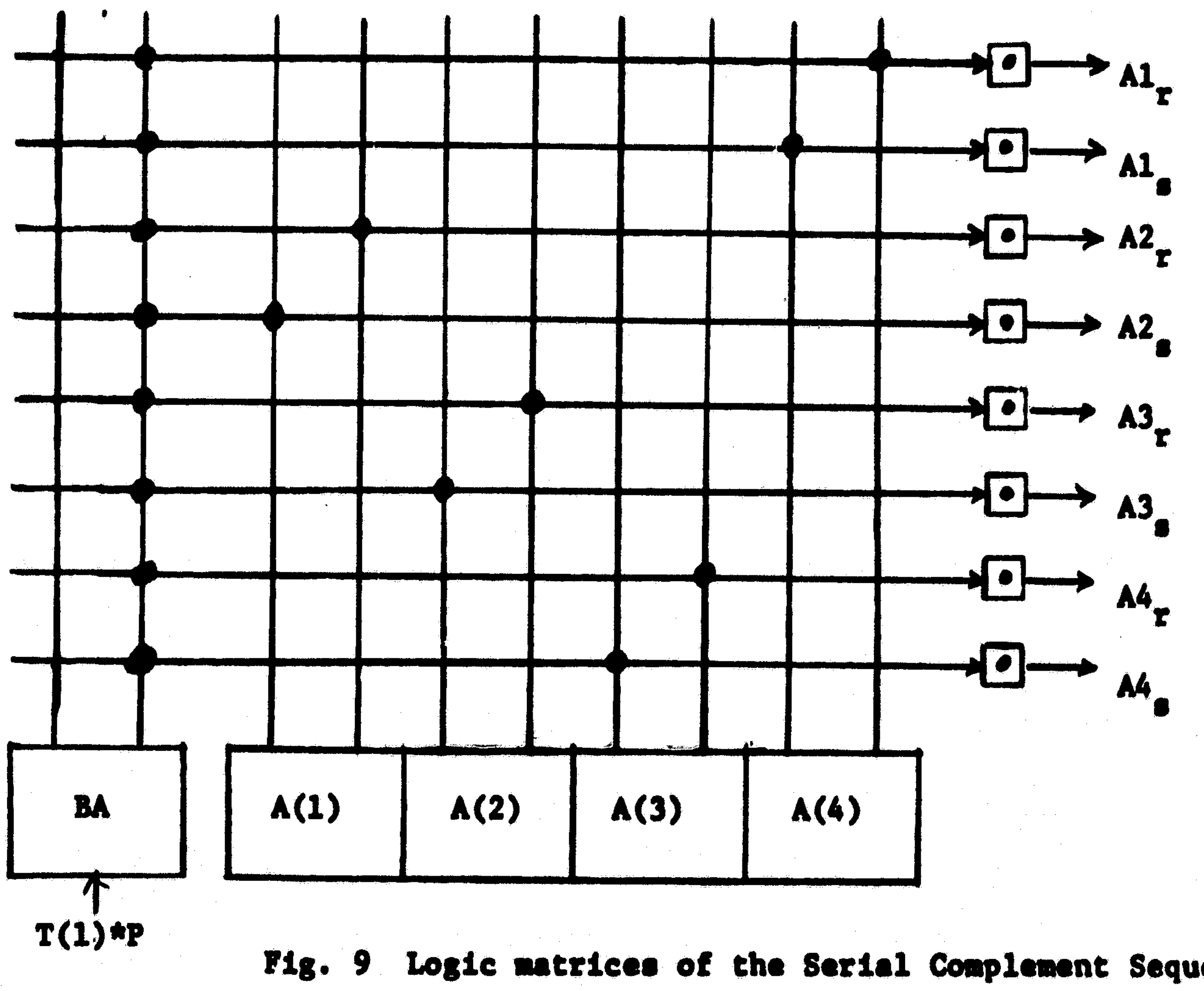
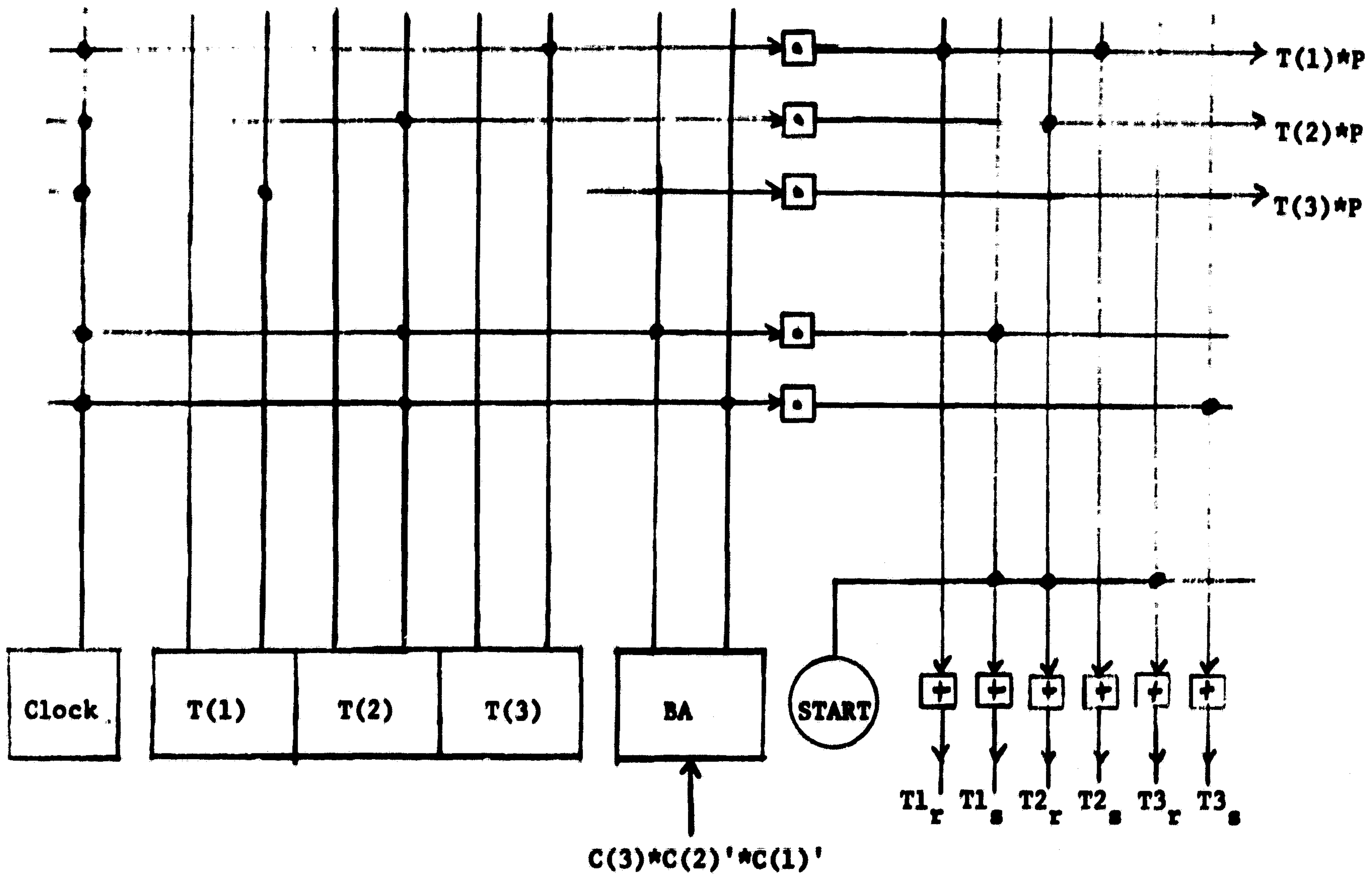


Fig. 9 Logic matrices of the Serial Complement Sequence

(c) control matrix and reset-T matrix



(d) set-FINI and reset-FINI matrix

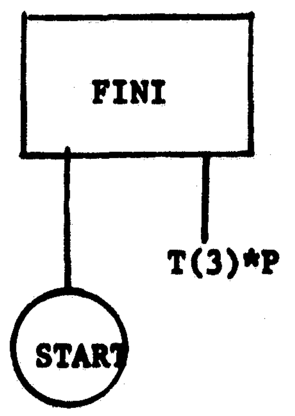


Fig. 9 (continued)

5.2 Logic Matrices

Each of the above boolean matrices describes a micro-operation together with the associated control signal. Each of these matrices can be diagrammed in a matrix form, and they are called logic matrices. Figure 9 shows several logic matrices.

Figure 9(a) shows the count-C matrix described by the boolean matrix in the matrix equation (19). Each 0 of the boolean matrix in the matrix equation (19) represents an interconnection which is represented by a dot in Figure 9(a). There is an exact correspondence between the 0's of the boolean matrix and the dots in the logic matrix. The reset-C matrix is also shown in Figure 9(a); this is the line connected to the START switch. Figure 9(b) shows the complement-shift matrix described by the boolean matrix in the matrix equation (23). Again, there shows the correspondence between the 0's of the boolean matrix and the dots of the logic matrix. Figure 9(c) shows the control matrix described by the boolean matrix in the matrix equation (25). Again, there shows the correspondence. The set-T matrix is also shown in Figure 9(c); this is the line connected to the START switch. Figure 9(d) shows the set-FINI and reset-FINI matrix which is degenerated into two lines.

In short, the patterns of 1's and 0's in the boolean matrices specify the patterns of interconnections if the micro design is implemented by logic matrices. This approach realizes logic design and circuit implementation simultaneously.

5.3 Implementation by Logic Matrices

The approach of implementing the complement sequence by means of logic matrices is illustrated by the block diagram in Figure 10. As indicated, each logic matrix performs one micro-operation. Thus, the functional nature of the implementation by logic matrices is apparent.

If the blocks in Figure 10 are replaced by the logic matrices in Figure 9, the diagram in Figure 10 becomes the one in Figure 11. This diagram in Figure 11 may be regarded as a large logic matrix which can be partitioned into smaller matrices on a functional basis. Therefore, the approach of using the logic matrices is amenable to implementation by large-scale integration of logic circuits. Building blocks of such an implementation has been presented elsewhere (14).

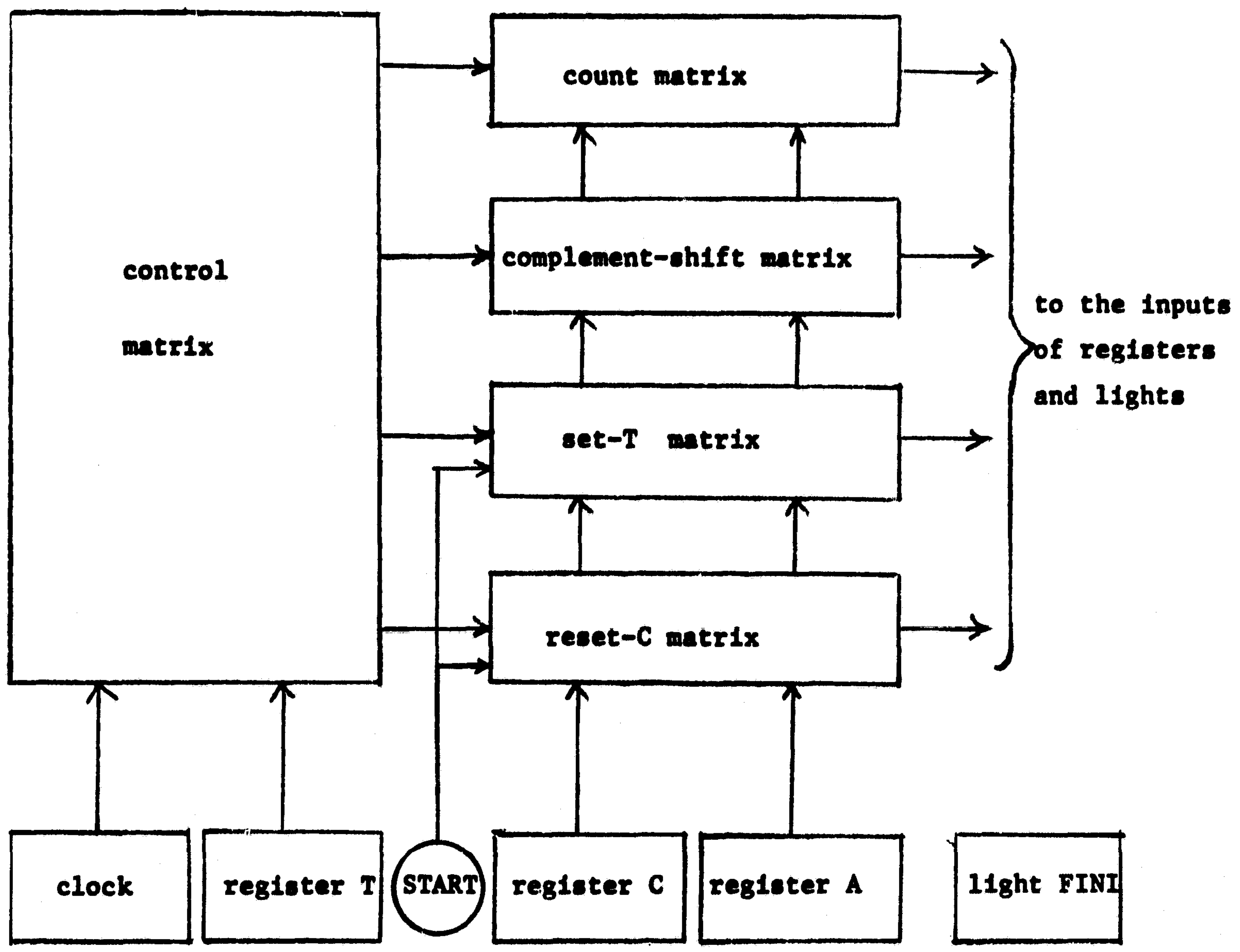
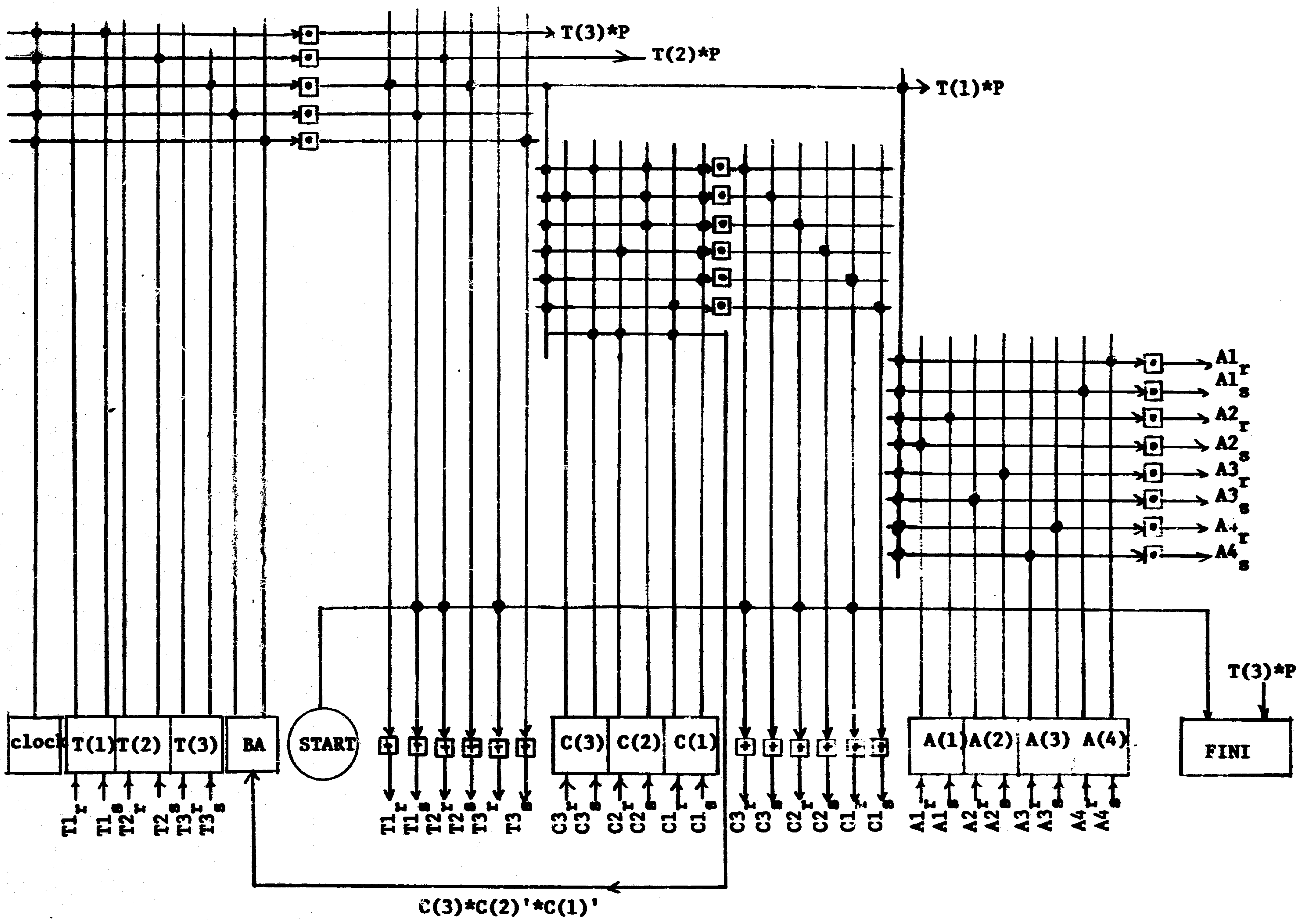


Fig. 10 Implementation of the serial complementer by logic matrices



T(3)*P
T(2)*P
T(1)*P
clock
T(1)
T(2)
T(3)
BA
START
C(3)
C(2)
C(1)
A(1)
A(2)
A(3)
A(4)
FINI
T(3)*P
$T1_r$ $T1_s$ $T2_r$ $T2_s$ $T3_r$ $T3_s$
$C3_r$ $C3_s$ $C2_r$ $C2_s$ $C1_r$ $C1_s$
$A1_r$ $A1_s$ $A2_r$ $A2_s$ $A3_r$ $A3_s$ $A4_r$ $A4_s$
C(3)*C(2)'*C(1)'

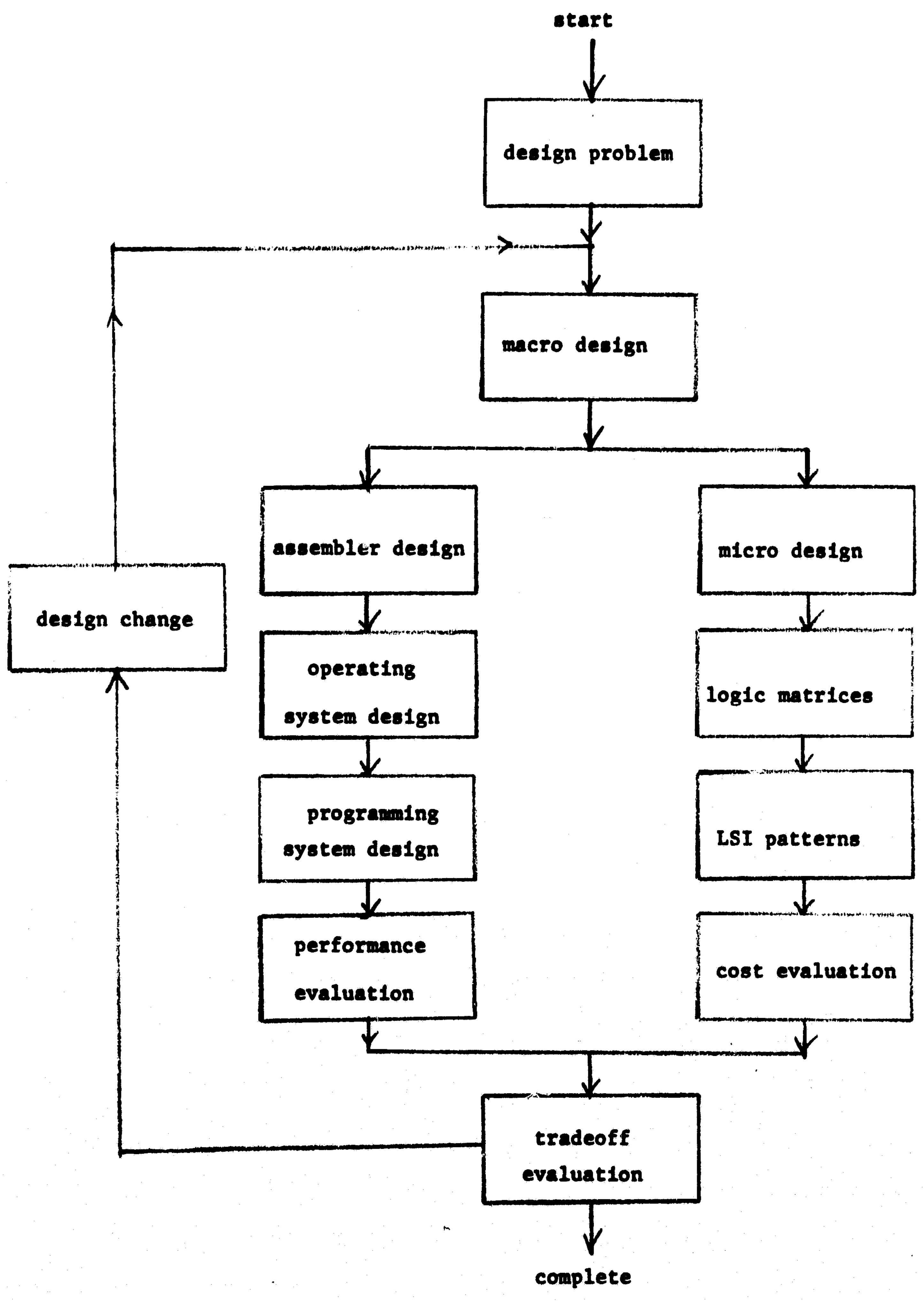
Fig. 11 A large logic matrix for large-scale integration

6. Unified Hardware-Software Design

An important objective of the development of the Computer Design language is to bridge the communication gap between the hardware and software designers and to realize a unified hardware-software design. This is possible because the description of a CDL design is readily understood by the software designer without knowledge of electronics. On the other hand, the CDL design specifies every bit and every step of operation required by the hardware designer.

Figure 12 shows a block diagram to illustrate the approach for a unified hardware-software design. When the design expressed in the CDL is completed, the hardware designer simulates the design, checks out the operations, translates the design for implementation, and in the meantime selects and develops devices, circuits, memories, packaging and interconnecting techniques, while the software designer builds and tests the assembler and designs and implements the operating system and programming systems. The results of the cost evaluation from the hardware design and the performance evaluation from the software design offer a basis for tradeoff. Should a change in the design be needed as a result of the tradeoff consideration, the change is made in the macro design; this brings

Fig. 12 Unified hardware-software design automation



about another design iteration. The simultaneous and unified design may need a shorter design period and may produce a better computer system design.

References

1. K. E. Iverson, "A Programming Language", John Wiley and Sons, New York, 1962.

2. A. D. Falkoff and K. E. Iverson, "A Formal Description of System/360", IBM Systems Journal, Vol. 3, No. 3, 1964, pp. 198-263.

3. H. P. Schlaeppi, "A Formal Language for Describing Machine Logic, Timing and Sequencing (LOTIS)", IEEE Trans. on Electronic Computers, August 1964, pp. 439-448.

4. A. P. Mullery, "A Procedure-Oriented Machine Language", IEEE Trans. on Electronic Computers, August 1964, pp. 449-455.

5. R. M. Proctor, "A Logic Design Translator Experiment Demonstrating Relationships of Language to Systems and Logic Design", IEEE Trans. on Electronic Computers, August 1964, pp. 422-430.

6. H. Schorr, "Computer-aided Digital System Design and Analysis Using a Resiter Transfer Language", IEEE Trans. on Electronic Computers, December 1964, pp. 730-737.

7. Y. Chu, "An Algol-like Computer Design Language", Comm. of ACM, October 1965, pp. 607-615.

8. M. S. Zucker, "LOCS: An EDP Machine Logic and Control Simulator", International Convention Record, Part 3, 1965, pp. 28-50.

9. D. L. Parnas, "A Language for Describing the Function of Synchronous Systems", Comm. of ACM, February 1966, pp. 72-76.

10. J. A. Wilber, "A Language for Describing Digital Computers", Report No. 197, Dept. of Comp. Science, U. of Illinois, February 15, 1966.

11. A. Giese, "Hargol - A Hardware Oriented Algol Language", Internal Report No. VA5, August 1966, A/S Regnecentralen, Copenhagen, Denmark.

12. G. Metze and S. Seshu, "A Proposal for a Computer Compiler", Proc. of the SFCC Conference, 1966, pp. 253-263.

13. M. A. Breuer, "General Survey of Design Automation of Digital Computer , Proc. of the IEEE, Vol. 54, No. 12, December 1966, pp. 1708-1721.

14. Y. Chu, "Building Blocks for Large-scale Integration of Logic Circuits", Proc. of Second International Congress on Microelectronics", Munich, 1966, Springer-Verlag.

15. C. K. Mesztenyi, "Translator and Simulator for the Computer Design and Simulation Program, Version 1", Technical Report 67-48, Computer Science Center, University of Maryland, May 1967.

16. T. D. Friedman, "ALERT: A Program to Compile Logic Designs of New Computers", Digest of the First Annual IEEE Computer Conference, September 6, 1967, pp. 128-130.

17. D. F. Gorman, Systems Level Design Automation: A Progress Report of the System Descriptive Language (SDL II)", Digest of the First Annual IEEE Computer Conference, September 6, 1967, pp. 131-134.

18. J. R. Duley and D. L. Dietmeyer, "A Digital System Design Language (DDL)", IEEE Transactions on Computers, September 1968, pp. 850-861.

19. J. A. Darringer, "A Language for the Description of Digital Computer Processors", Proceedings of the Design Automation Workshop, July 1968, (IEEE Cat. 68C36CPR).

20. C. K. Mesztenyi, "Computer Design Language: Simulation and Boolean Translation", Technical Report 68-72, Computer Science Center, University of Maryland, June 1968.