Technical Report TR-66-35
NsG-398 October 1966

Building Blocks for Large-Scale Integration of Logic Circuits

by

Yaohan Chu

Abstract

This paper describes a study which has led to a set of building blocks for large-scale integration of logic circuits. The study began by designing a very simple but non-trivial stored-program digital computer. It was possible to arrange the logic circuits of the entire computer into a large matrix (a diode matrix is used as an illustration in this paper) and register blocks. This large matrix was then divided according to the micro-operations of the computer into 22 small matrices, each of which can be obtained from one universal matrix by disabling unused matrix junctions with an attempt to have one matrix perform just one micro-operation. The idea of integration of both circuits and logic, not circuits alone, is thus shown.

From the above study, a set of two building blocks for large-scale integration of logic circuits is proposed: a matrix chip for performing micro-operations and a register chip for storing a binary word. Alternatively, matrix chips and combination chips (on the latter of which both the matrix and the register are fabricated) can be used, following the same design and basic layout. At present, bipolar transistor or MOS field-effect transistor matrix and register chips are probably the best choice for most applications. Each chip may have hundreds or thousands of devices, yet each is universal in permitting implementation of any logic functions by disabling unneeded devices. For practical reasons, availability of several sizes of chip is highly desirable.

Table of Contents

Building Blocks for Large-scale Integration of Logic Circuits

Yaohan Chu

1. Introduction

The rapid advance of monolithic technology in semiconductors has made integrated circuits a success in a span of only several years. Indeed, integrated circuits are widely accepted today, and digital computers built with integrated circuits are being offered by computer manufacturers. Most of these integrated circuit packages in a digital computer are individual gates or a group of several gates that are bult to replace packages each of which has one or several discrete-component circuits. The trend has been clear that the next generation of integrated logic circuits is a large-scale integration of circuits (1), because large-scale integration has the potential of further lowering the cost, further increasing the reliability of the system, and opening up a new approach in digital system design. Announcement has recently been made of a large number of gates in a single package, particularly those MOS circuits where there can be up to 100 circuits in one package. On the other hand, there exists an interesting problem as to how large-scale integration should be accomplished. If one merely puts more and more circuits on a silicon chip, one loses the flexibility that simple logic circuits have thus far enjoyed. Indeed, the loss of flexibility in large-scale integration of

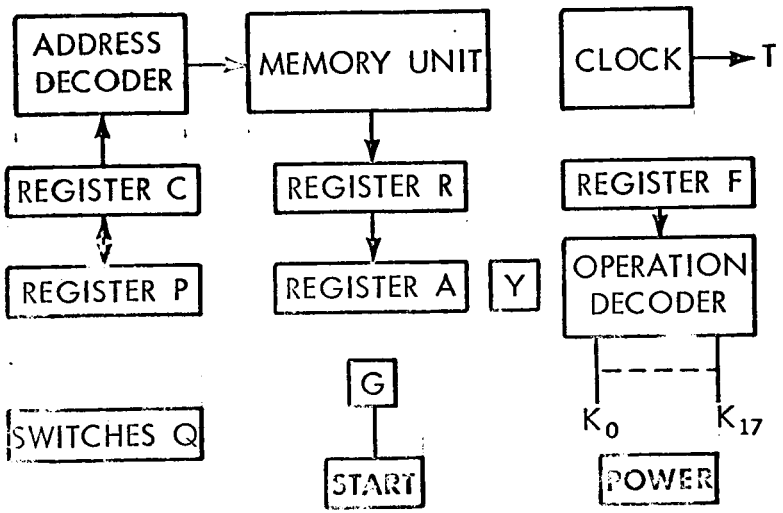

Figure 1, Configuration of the computer

circuits can offset the cost reduction that has been originally envisioned, and can create an inventory problem for computer maintenance.

This paper reports the result of a study of the logic of a simple digital computer which has led to a set of building blocks for large-scale integration of logic circuits. This set of building blocks integrates both circuits and logic. We now begin by describing the logic of a simple digital computer.

2. A Simple Stored-program Computer

A simple but nontrivial stored-program computer was designed for the purpose of studying the relation between the logic structure and the integrated circuit. It is a binary, parallel, synchronous computer. There are nine instructions of single-address format.

Each word in the computer consists of 4 bits. As a number, it is a 4-bit fixed-point number with the leftmost bit as the sign bit, and the binary point can be considered to be located at the right of the sign bit. Negative numbers are represented in two's complement format. As an instruction, a word consists of two parts: the operation code part and the address part. The former consists of the leftmost three bits and the latter the one remaining bit.

The major elements of the computer are shown in Figure 1. There are two 4-bit memory registers, one 4-bit buffer register R, one 4-bit accumulator (or register A), one 4-bit operation

counter F and an associated operation decoder, one 1-bit address register C, one 1-bit program register P, one start-stop flipflop G with an associated switch S, one carry flip-flop Y, a clock source, 8 Q-switches (one for each memory flipflop to insert the inputs into the memory), and pin lights to show the outputs of the flipflops. Notice that the memory unit is made of flipflops. This choice was for the sake of simplicity. On the other hand, such a small-capacity transistor memory can be and will be used in the computers of the near future, not to replace the usual magnetic memory but for other purposes.

There are nine instructions which include addition and subtraction, but for the sake of simplicity do not include multiplication and division. These instructions and their internal codes are shown in Table 1, where x represents the address and y a value of either 0 or 1. In addition, there are the two manual operations of turning the power on or off, and of starting or stopping the computer operation.

Each instruction is implemented by one or more so-called micor-operations. For a synchronous computer, a micro-operation can be regarded as that can be built with hardware to be performed in one clock pulse. Therefore, the complexity of a micro-operation depends to a great extent on the hardware technology. The micro-operations that are required in this computer are shown in Table 2. There are nine types, in

Table 1. Instruction Repertoire

Instruction	Code
add	000x
subtract	001x
jump on minus	010x
store	011x
jump	100x
clear accumulator	101y
shift acc. one bit toward right	1100
circular shift acc. one bit toward left	1101
stop	111y

Table 2. Micro-operations

Type	Required in the design
set a register	G <-- 0
	G <-- 1
	Y <-- 0
	Y <-- 1
	P <-- 0
	C <-- 0
	A <-- 0
	F <-- "an octal constant"
transfer	P <-- C
	C <-- P
	F(OP) <-- R(OP)
	C <-- R(AD)
count	P count +1
complement	R <-- comp R
add	A <-- A add R
shift right	A <-- 1 shr A
circular shift left	A <-- 1 cirl A
load	R <-- M(C)
store	M(C) <-- A

addition to the generation of timing, control, and sequencing signals. The actual micro-operations that each type performs are also shown in the Table. Note that the micro-operations to set a register to a particular bit pattern and to transfer the contents of one (or part of one) register to another (or part of another) are the two most useful ones.

Next comes the problem of how to sequence these micro-operations so that the instructions of a program stored in the memory unit can be properly executed. This is done here, as is done conventionally, by forming two cycles: a fetch cycle and an execution cycle. An instruction is taken from the memory during the fetch cycle, and then executed during the execution cycle. These two cycles are sequenced alternately. Such a design can be done symbolically by using an Algol-like computer design language (2). Symbolic design of this simple computer is shown in the Appendix.

The symbolic design in the Appendix begins with a comment statement and is immediately followed by the statements designating the names of registers, memory, decoder, switches, and clock. Statements generally carry a label such as POWERON or START, which is the command signal for the statement. The statement is executed when the command signal is activated. For example, when the POWER switch is at the ON position, the command signal causes the micro-operations $F \leftarrow 17$ and $G \leftarrow 0$ to be executed simultaneously. Other command signals consist of a coincidence of a control signal from the operation decoder

K(i) and a clock pulse T. These control signals are activated by the contents of register F. During the execution of one statement, register F is normally set to a preassigned octal number for the purpose of specifying the next statement to be executed. In short, the micro-operations are grouped into statements and the statements are organized into sequences. Each instruction merely calls the execution of a particular sequence. There is only one sequence for the fetch cycle, but there are as many sequences as the number of instructions for the execution cycle. And when the execution of one instruction is completed, the next instruction is fetched from the memory. In this manner, the program in the memory is carried out.

3. Integration of logic and circuits

By integration of logic and circuits, we mean that the logic circuits are integrated on the basis of performing micro-operation. Since each micro-operation requires a significant amount of logic circuits, integrated logic circuits built on this basis give large-scale integration.

The design of the computer described in the Appendix can be expressed into a set of boolean equations, which represent the logic of the computer. However, the design can alternatively be expressed by boolean matrices (3), and the logic of the computer can now be represented by a large matrix. For convenience, diode logic circuits are used to illustrate the implementation. When these diode circuits are arranged into a matrix form, the logic of the computer is now implemented

by a large diode matrix, as is shown in Figure 2. In Figure 2 the squares along the sides are flipflops or switches. The vertical and horizontal straight lines of the large matrix are desposited conductors. An intersection where there is a circle represents the presence of a diode. The zigzag lines represent the resistors. The rectangles inside the diode matrix are binary amplifiers, as these diode circuits cannot be connected for more than two levels. The pattern of the diodes in the matrix indeed prescribes the logic of the simple computer.

The large matrix in Figure 2 is next rearranged into such a manner that it can be divided into smaller matrices as shown in Figure 3. There are 22 matrices in Figure 3 in addition to six blocks of registers and two blocks of binary amplifiers. The registers are located in blocks AV, AW, EV, EW, EX, and EY, while the binary amplifiers are in AX and EZ. In the interest of having only one matrix size, the logic for the operation register F to generate control signals is implemented by the two matrices in blocks BV and CV. The micro-operations to set the accumulator to zero, to shift the accumulator toward right and to circularly shift the accumulator toward left can all be put in one matrix in block BY. The micro-operation of complementing the R register takes only one half of the matrix in block DX. The add micro-operation involves eight matrices. In short, circuits are divided into matrices according to the micro-operations.

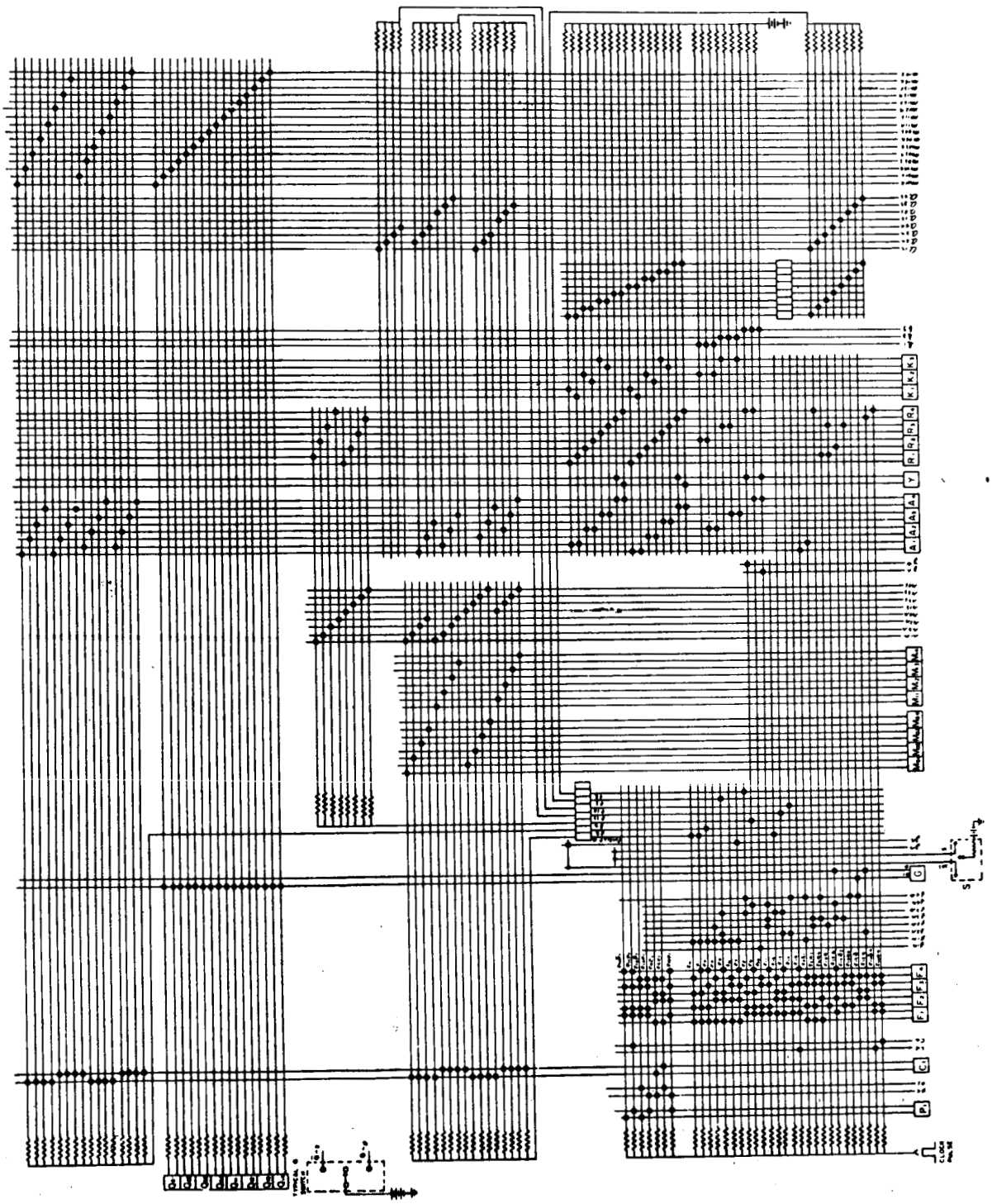

Figure 2, A diode matrix for the computer logic

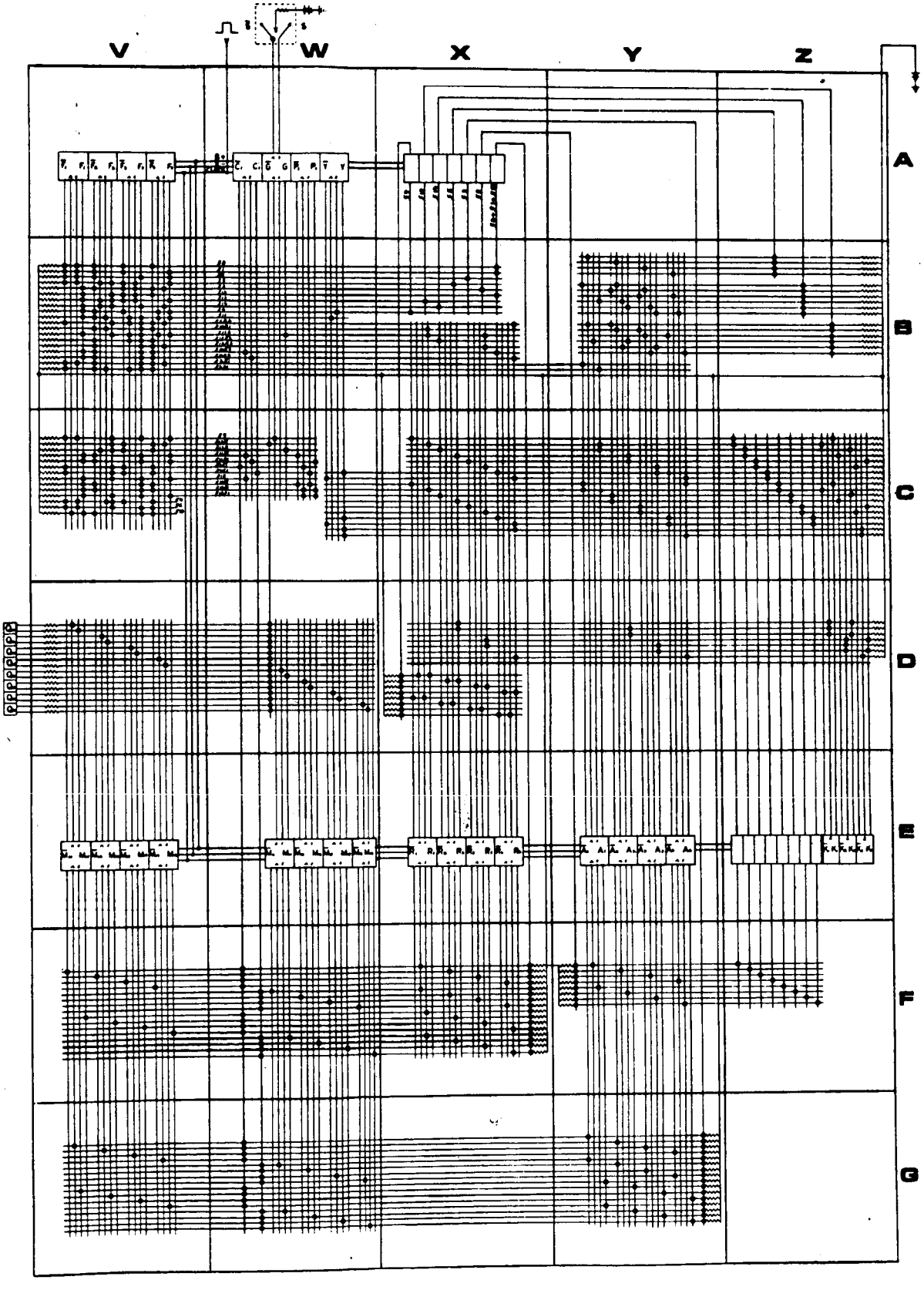

Figure 3, Partition of the large matrix into smaller matrices

4. Building Blocks

The basic idea of implementing the micro-operations of the computer is illustrated in Figure 4, where each matrix performs one micro-operation. Each matrix is an integrated circuit with a large number of devices. This design can be accomplished by considering two building blocks: a matrix chip and a register chip. The register chip stores a word (or a partial word) of a number of bits, and the matrix chip performs a micro-operation. The matrix chip can be a diode matrix as shown in Figure 5, on which the undesired diodes have been disabled by selective burnout or other suitable destructive means. Alternatively, if the scale of production permits, a special matrix can be fabricated by changing slightly in the set of masks. Note that such a special matrix could be replaced by a properly prepared universal matrix if maintenance so required.

The combination of a four bit register chip and an 8 by 8 diode matrix chip is shown in Figure 6. Alternatively, the register and matrix could be on one chip. If the diodes on the universal matrix of Figure 5 are so chosen that the effective diodes have the pattern as those shown in Figure 7 (with the unused diodes disabled), the diode matrix is a count matrix, and the register together with this matrix can carry out a count micro-operation. Thus, the combination of the count matrix and the register becomes a counter. If the effective diodes have

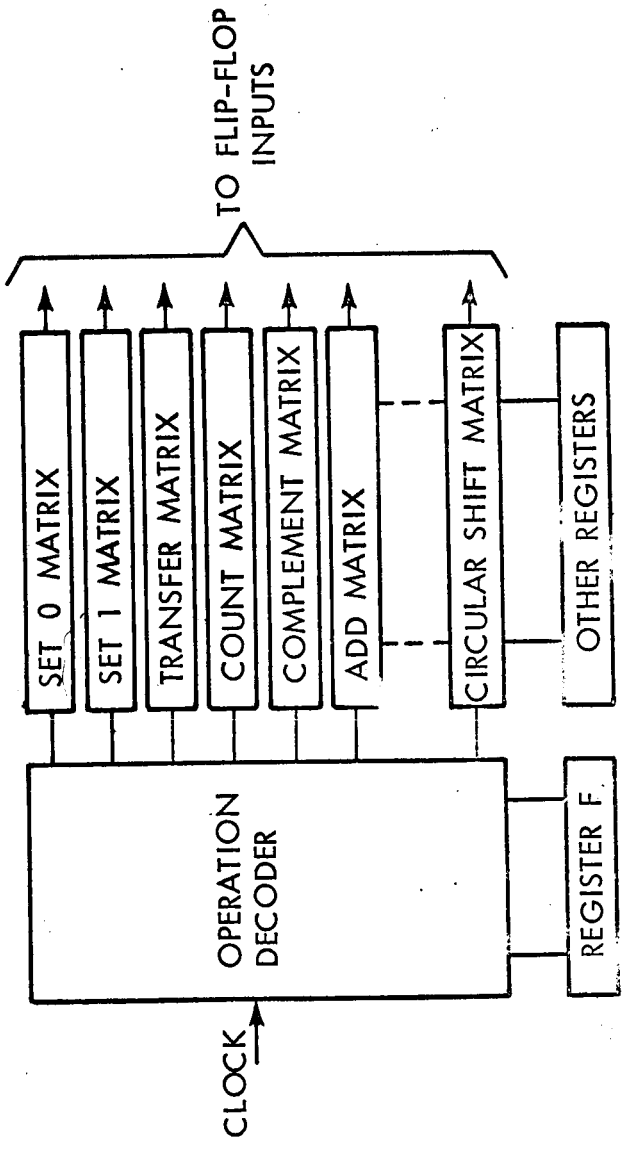

Figure 4. Block diagram showing the use of matrices to implement micro-operations

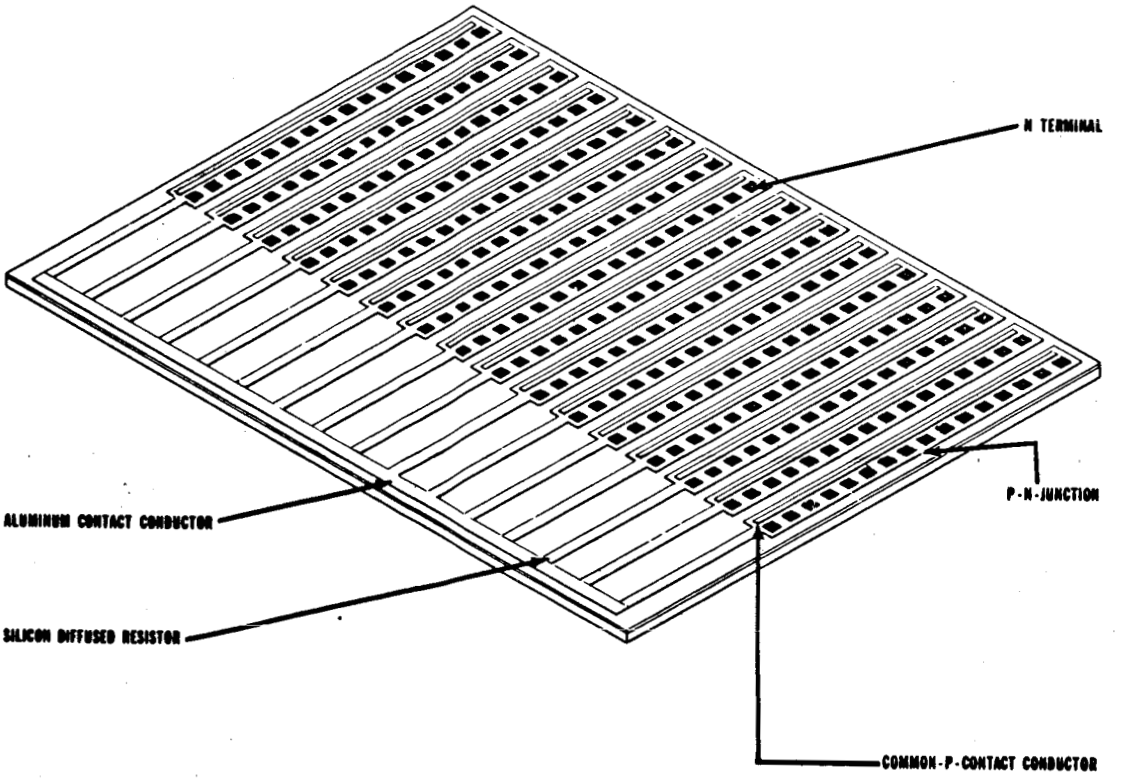

Figure 5. Diode matrix chip.

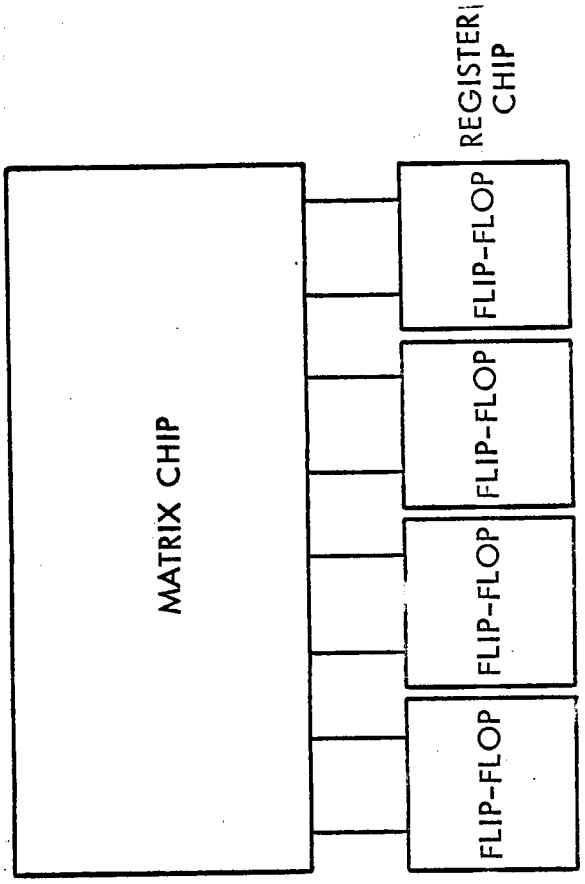

Figure 6. Two building blocks: matrix chip and register chip

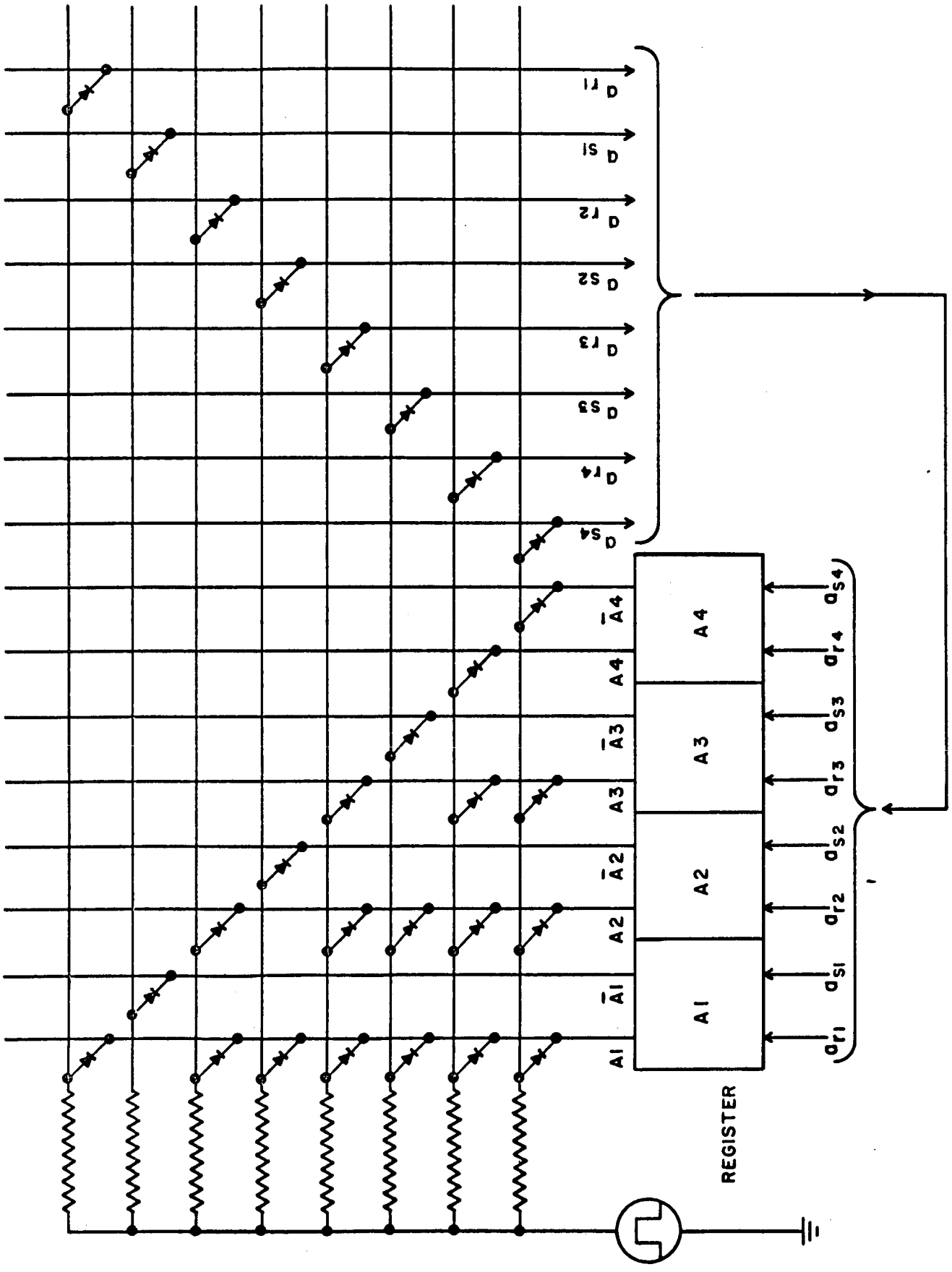

Figure 7. Diode-Counting Matrix

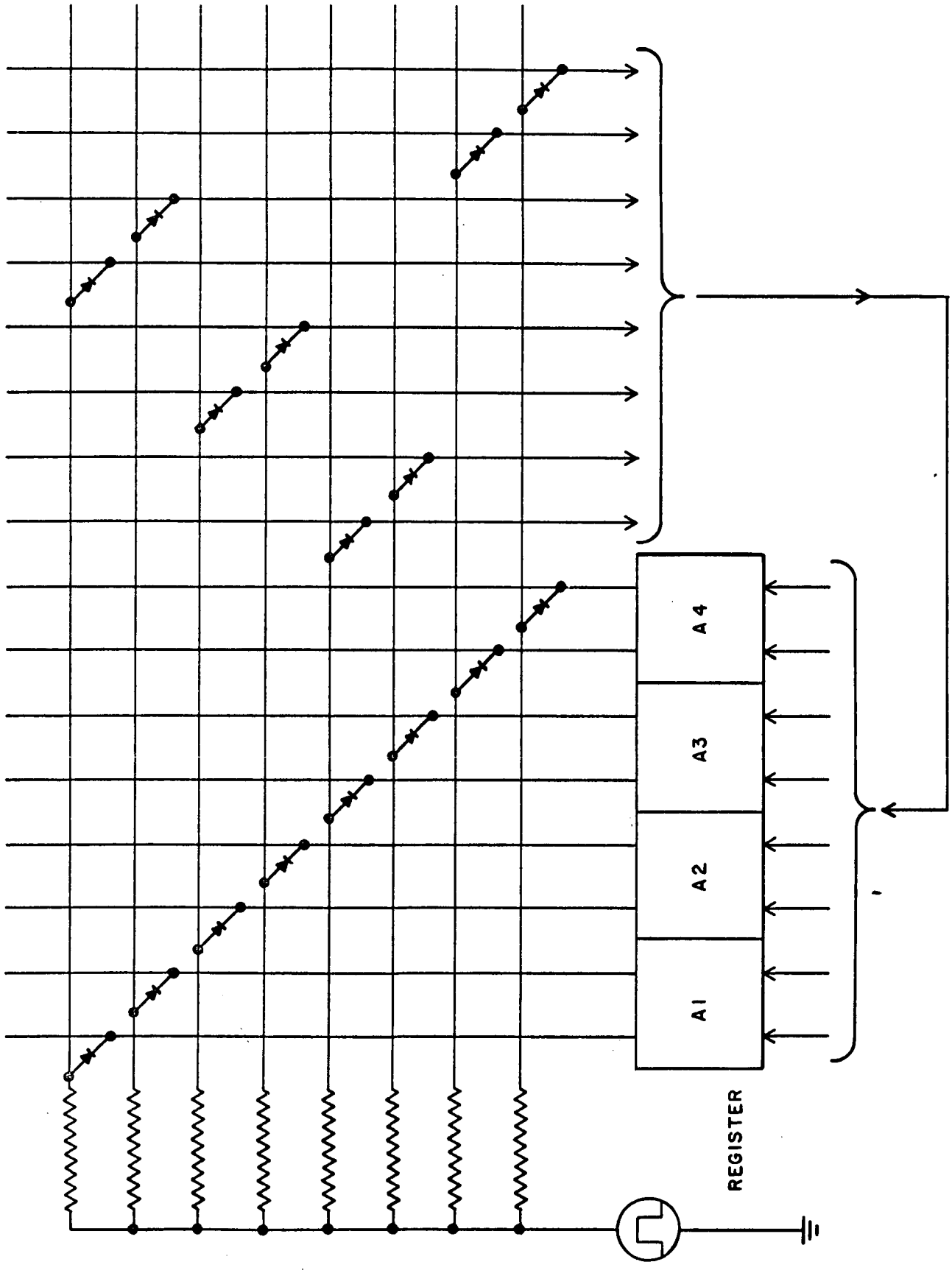

Figure 8. Diode-Shifting Matrix

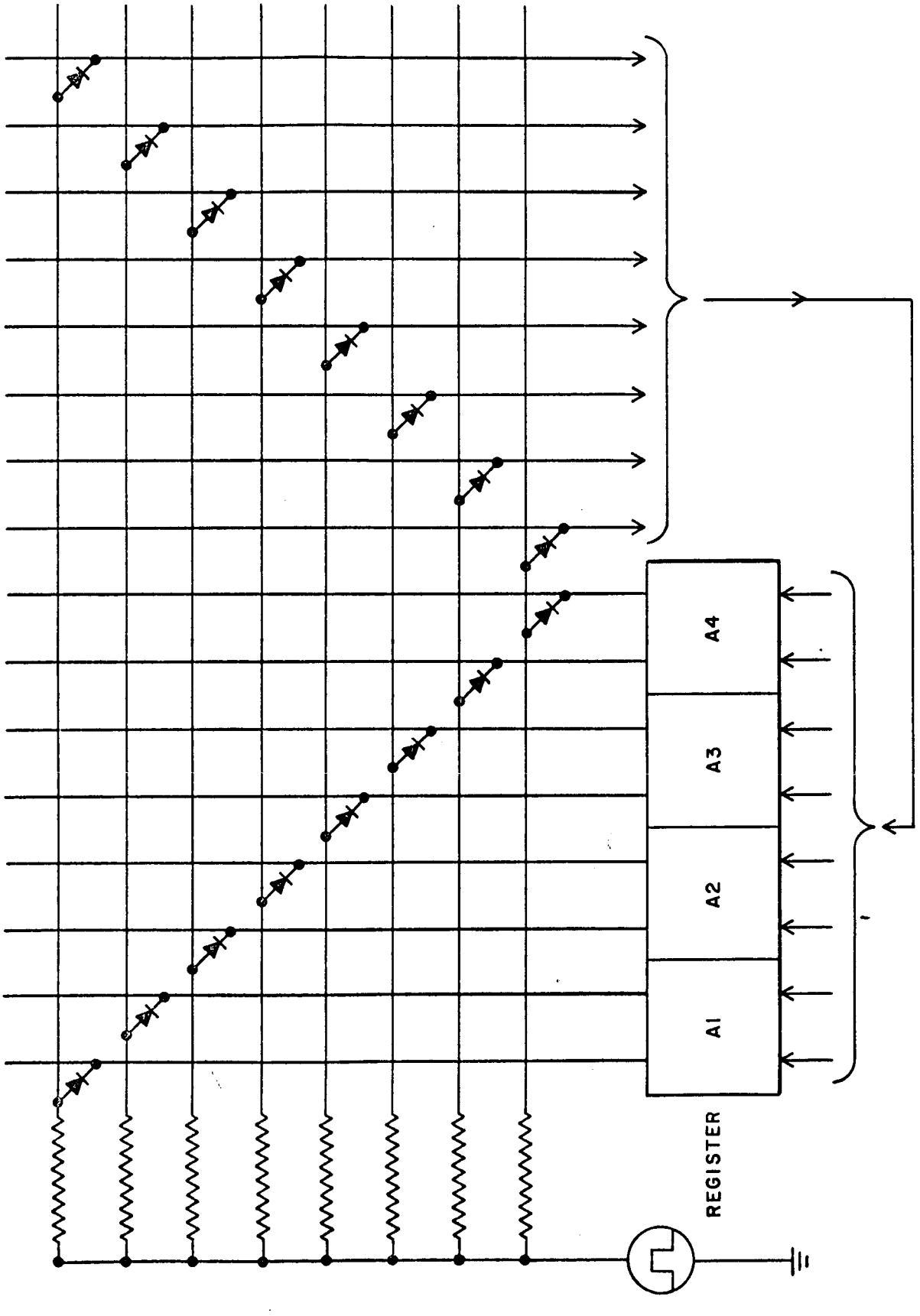

Figure 9. Diode-Complementing Matrix

the pattern as those shown in Figure 9, the diode matrix complementing matrix, and the register can now carry out a complement micro-operation. In other words, it is the pattern of the diodes in the matrix that determines what micro-operation the matrix performs.

For micro-operations where only one register is involved, one matrix is sufficient to implement these micro-operations. For micro-operations where two registers are involved such as the logical-AND operation on the corresponding bits of the two registers, two matrices are required for each micro-operation. And for more complex micro-operations involving two registers such as add or subtract, more than two matrices are required and these matrices may have to be connected on a multiple level basis. In the case of diode matrix requiring more than two levels, binary amplifiers are required. On the other hand, the use of NOR matrices or NAND matrices using diode-transistor circuits or transistor-only circuits eliminates the need for binary amplifiers between the levels.

An important problem in integrated circuits is that of interconnecting them. In the proposed design, there are two types of connections: (A) those between matrix chips and a register chip, and (B) those between matrix chips. For type A connections, there are four leads on each flipflop; two input and two output leads. These connections can be accomplished in the following manner: the output leads of the

register chip form a "micro-bus" on which are connected the input leads of those matrices for the micro-operations that the register is required to perform. The spacing of the vertical conductors on the matrix chip is uniform and should match the spacing of the input-output leads of the register chip. The connections need not be permanent (such as a row of micro-pins), so that each matrix chip could be easily removable. For micro-operations involving a single register, this is the only type of connection required. For micro-operations involving two registers, type B connections are also required. For type B connections, the leads of the uniformly-spaced horizontal conductors of one matrix are connected to those of other matrices. Again, the connections depend on the proximity of the two registers; longer connections are often encountered if the micro-operation is conditional upon some bits in another register. Ultimately, the density of the devices on the chip and the size of each chip will be limited by the interconnection technology.

It thus appears that one can use the matrix chip and the register chip as the basic building blocks. The design parameters for the register chip are the number of bits, the kind of devices for flipflops, and the type of circuit and gating; those for the matrix chip are the kind of devices for switching, the type of circuits, and the number of rows and columns.

For a properly chosen set of parameters, one matrix chip and one register chip can be adequate. But for better utility of these chips, several sizes of register chip and matrix chip are highly desirable. Although a diode matrix has been used to illustrate the idea, other devices can be similarly applied to accomplish large-scale integration.

An alternative choice of a set of building blocks can be simply a matrix chip and a combination chip on which both the register and the matrix (as those shown in Figure 6) are fabricated. This choice gives the advantage of reducing type A connections. From practical consideration, there can be a number of different sizes of such combination chips with different numbers of rows of the matrix and/or different numbers of bits of the register.

5. Conclusion

The proposed set of building blocks of a matrix chip and a register chip or an alternative set of a combination chip and a matrix chip is amenable to large-scale integration of logic circuits on a single chip of semiconductor. The minimum set can be fabricated of two building blocks, yet these building blocks are universal in their logic applications when unwanted matrix junctions are disabled.

When the matrix chips are organized on micro-operation basis, a malfunctioning matrix can be located quickly by

detecting a corresponding faulty micro-operation, and can be serviced simply by merely replacing that matrix. The building blocks based on integration of both logic and circuits can be applied to large-scale integration of various kinds of integrated devices such as diode chips, transistor chips, field-effect transistor chip, thin-film transistor chip, or cryotron chip. At present bipolar transistor chips and MOS field-effect transistor chips are probably the best choice.

6. References

(1) "Large Scale Integration", J. S. Kirby and J. C. Logue, Digest of Technical Papers, International Solid-state Circuits Conference, 1966, pp. 30-31.

(2) "An Algol-Like Computer Design Language", Yaohan Chu, Communications of the ACM, Volume 8, Number 10, October, 1965, pp. 607-615 U.S.A.

(3) "Digital Computer Design Fundamentals", Yaohan Chu, McGraw-Hill Book Company, U.S.A., 1962.

Appendix. Symbolic Design of the Simple Computer

```
        comment begin this is the description of a design of a
        simple stored program digital computer using chu's computer
        design language. The four-bit instruction format consists
        of a three-bit op-code field and one-bit address field.
        There are nine instructions: add, subtract, jump, jump-on-
        minus, store accumulator, shift left, shift right, clear
        accumulator, and stop. The memory has two four-bit words
        which are registers. The sequencing is controlled by
        operation counter F. start-stop is controlled by flipflop
        G. end
        register R(1-4); F(1-4); A(1-4); C(1); G(1); P(1); Y(1).
        subregister R(OP)=R(1-3); R(AD)=R(4); F(OP)=F(244).
        memory M(C)=M(0-1, 1-4).
        decoder K(0-17)=F.
        clock T.
        switch POWER(ON, OFF); START(ON, OFF).
        comment begin the following statement with label poweron
        indicates what happens when power switch is turned on
        and that with label start indicates what happens when
        the operator next turns the start switch on. end
POWER(ON): F ← 17; G ← 0.
START(ON): G ← 1.
        comment begin the following is the wait sequence which
        is broken off when G is equal to 1. end
K(17)*T: if G=0 then begin C ← 0; P ← 0 end; if G=1 then F ← 14.
        comment begin the following is the fetch sequence. end
K(13)*T: C ← P; Y ← 0; F ← 14.
K(14)*T: R ← M(C);
        if G=0 then F ← 17;
        if G=1 then F ← 12.
```

```
K(12)*T:   F(OP) ← R(OP); C ← R(AD); P count +1; F(0) ← 0.

           comment begin the following is the add sequence. end
K(00)*T:   R ← M(C); F ← 10.
K(10)*T:   A ← A add R; F ← 13.

           comment begin the following is the subtract sequence. end
K(01)*T:   R ← M(C); F ← 11.
K(11)*T:   R ← comp R; Y ← 1; F ← 10.

           comment begin the following is the jump sequence. end
K(04)*T:   P ← C; F ← 14.

           comment begin the following is the jump-on-minus
           sequence. end
K(02)*T:   if A(1)=0 then F ← 13 end;
           if A(1)=1 then F ← 4 end.

           comment begin the following is the store-accumulator
           sequence. end
K(03)*T:   M(C) ← A; F ← 13.

           comment begin the following is the clear-accumulator
           sequence. end
K(05)*T:   A ← 0; F ← 13.

           comment begin the following are the two shift sequences. end
K(06)*T:   if R(4)=0 then F ← 16;
           if R(4)=1 then F ← 15.
K(15)*T:   A ← 1 shr A; P count +1; F ← 13.
K(16)*T:   A ← 1 cirl A; P count +1; F ← 13.

           comment begin the following is the stop sequence. end
K(07)*T:   G ← 0; F ← 17.
           end
```