L-60

Digital Computer Switching Circuits

Basic operational requirements of digital computers and fundamentals of the means for obtaining them are set forth. For the most part familiar switching circuits can be used but they must meet the special requirements of positive action that are described here

By C. H. PAGE

Chief, Electronic Computers Section
National Bureau of Standards
Department of Commerce
Washington, D. C.

AUTOMATICALLY-SEQUENCED digital computers are machines that have no intelligence, yet carry out, without intervention, lengthy routines of mathematical calculation. An understanding of general design considerations requires a survey of the procedures followed by a human computer using desk calculator.

A human computer does more than arithmetic; he not only carries out the elementary processes of addition, subtraction, multiplication, and division, but he also decides what numbers to add, multiply, etc., and what to do with his results. These results of his arithmetic are only stepping stones to his final goal, just as the numbers upon which he performs his arithmetic were previous stepping stones. Some problems require millions of arithmetic operations to arrive at a relatively small set of numbers representing the final answer.

If we reduce the human computer to an automaton having only the ability to read, write, and do arithmetic, we need to give him a very detailed set of working instructions. These instructions include original numerical data from which he works, and an explicit program of operations to be performed. He must be told, for example, to read numbers in two specified places, add them, and write the result in a specified place. He must then be told where to find his next instruction, unless all instructions are serially listed and no variations in their order are to be made. Explicit instructions as to where to write partial results and when and where to refer back to them for further use comprise a sort of automatic memory. The sheets of paper, numbered for identification, form a storage for numbers; his whole program is stored on paper before he starts to work.

Even the power of decision can be mechanized. If a human computer is supposed to compute one intermediate result to a specified degree of accuracy by a method of successive approximations, he must continue until further steps make insignificant changes. He is therefore instructed to keep repeating the procedure until a tentative answer, taken to ten places, equals the previous tentative answer, and then to proceed with the main program.

We see that our automaton must be given instructions, or orders, incorporating the following information: (1) where to find operands; that is, the two numbers to be combined by addition, multiplication, subtraction, or division, (2) which arithmetic operation to perform, (3) where to write the result, either in a specified place for furture reference or on his final answer sheet, and (4) where to find his next set of similar instructions.

An electronic computer operates on a similar routine. Machines being designed and built will perform this cycle of operations in a millisecond or less, working with numbers having ten decimal places. Such speed means that these machines will make it practical to solve problems requiring so many millions of arithmetic operations as not to be considered at present. Directing such a machine is a major administrative problem. As Dr. von Neumann of the Institute for Advanced Study expressed it, "Programming a problem for such a machine is equivalent to writing a detailed set of instructions for twenty automatons with desk calculators sufficient to keep them busy for two years, working a forty-hour week." These automatons have no ability to think for themselves!

Leaving the mathematical and administrative problems to others, we can proceed to the basic electronic problems. We must first have (A) an electronic alphabet for writing numbers and orders, (B) a medium on which to write, (C) means of writing and reading, and (D) means for interpreting the written word. These words may be numerical, as 3721499825, or coded orders, as A0173Q75B6. When a number-word (number) is read, it must be translated into what the machine recognizes as numerical form. An order-word (order) must be interpreted by being converted to a set of voltages, to operate switches.

Reading a word consists in part of transmitting it to the organ which is to interpret and be affected by it. Thus numbers are transmitted from storage to arithmetic unit, or vice versa, and orders are sent from storage to the central control organ, or dispatcher. In ad-

1 of 8

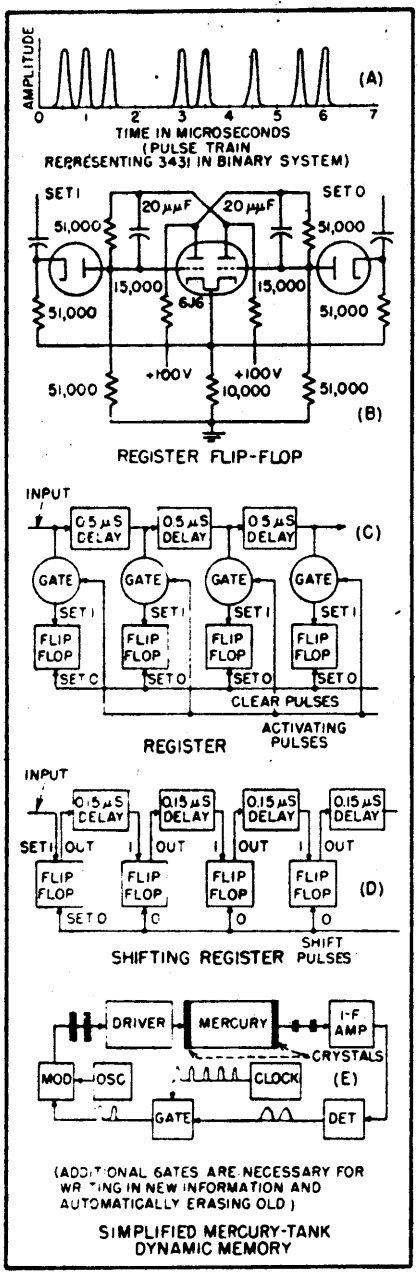


FIG. 1—Pulses are stored statically in flip-flops, dynamically in delay lines

dition, both kinds of words are transmitted to storage from the input as needed, and final answers or desired partial results are transmitted to the machine output.

An order must not only tell the central control which numbers to dispatch to the arithmetic unit from storage, but must also tell central control which arithmetic operation is to be performed and where the result is to be sent.

In addition to the central control organ, there must be various local control stations. The arithmetic unit itself, for example, is primarily a traffic unit such that the arrival of two numbers causes the transmission of a third number. Whether this third number is the sum, difference, product, or quotient of the other two depends upon the dispatching system of the arithmetic unit. Separate arithmetic units can be built for the four cases, but it is also feasible to make a universal arithmetic unit which will perform any one of the four processes upon request of the central control. Hence the central control must not only dispatch number-words and orders, but must also interpret orders and actuate circuit changes.

Transmission and Representation

A NUMBER, say 43712, can be read and transmitted in two fundamentally different ways. If one transmission channel is used for each column, we can simultaneously transmit a 2 along the first channel, a 1 along the next, 7 along the next, etc. This simultaneous transmission of the digits of each position along their appropriate channels is a PARALLEL operation. Its characteristic feature is that it distinguishes between digits by a spatial relation, transmitting all digits at the same time.

Conversely, we could transmit all digits over a common channel, at successive times, in the order 2, 1, 7, 3, 4. The separate digits would be distinguished by their time of arrival on a common line. This is a SERIAL process, digits being distinguished by a temporal relation.

If ten pulses, made recognizable from each other by modulation, are available, any number can be transmitted either serially, over one line, or in parallel, over many lines, from one organ to another. We will consider only serial operation because it is more illustrative of traffic (switching) dispatching problems, as well as because it is the system employed in the machines that will first be constructed.

ORDERS to various parts of the machine must also be capable of transmission, hence they can be expressed conveniently as numbers in some arbitrary code. Thus numbers and orders are represented in the same way, being strings of digits. We know which is which when we put them into the machine, so that if our programmer dispatches only orders to central points and numbers to arithmetic points, it will not matter that the machine by itself cannot distinguish orders from numbers. In fact, this is a convenience, because by considering an order as a number we can modify an order by operating on it with the arithmetic unit.

REPRESENTING the ten digits by pulses of different amplitude would reduce machine reliability, making results depend upon tube constants and supply voltages. It is better to have only two amplitudes to distinguish. If these two amplitudes represent digits 0 and 1, we must find a way of representing numbers in terms of these two digits. In decimal notation, the number 352 means

$$2 \times 10^0 + 5 \times 10^1 + 3 \times 10^2 = 2 + 50 + 300$$

Each successive digit position to the left represents the coefficient of the next higher power of 10. We therefore need digits only to 9; a coefficient of 10 in any place is equivalent to a coefficient of unity in the next place. If we d[illegible] use of 10 as our base, and us[illegible] stead, we write a number [illegible] as 37 in the following binary manner, 100101, meaning

$$1 \times 2^0 + 0 \times 2^1 + 1 \times 2^2 + 0 \times 2^3 + 0 \times 2^4 + 1 \times 2^5 = 1 + 4 + 32 = 37$$

We pay for the simplicity of having only two different digits by needing approximately three times as many columns to write a number in the binary system as in the decimal system.

To represent 0 and 1 and the corresponding pulse trains, we choose a basic pulse repetition rate of 2 mc, and synchronize all parts of the machine so that successive pulses (representing 0 or 1) occur at these half-microsecond intervals. If all trains of pulses are locked to this reprate (repetition rate), we can use the presence of a pulse to represent 1, and the absence of a pulse to represent 0. Thus the six-microsecond pulse train shown graphically in Fig. 1A represents the binary word 110101100111 (read from right to left) which has the (decimal) value 3431. Voltage and tube parameters need only be held within the tolerance range to

248

keep the pulses within their amplitude range of reliable operation.

Now that we have a scheme for representing numbers as pulse trains, we are ready to analyze problems of storing numbers.

STORAGE — Typical machines operate with numbers of ten significant figures in the decimal system, so will require roughly 35 binary places. A 35 binary place number at 2-mc reprate will be represented by a pulse train having a duration of 17.5 microseconds. It is impractical to put information into a machine or to print results at such a rate, over 50,000 words per second. We need a speed changer, or device for storing the many words being written into it at one speed, and capable of being read at some other speed, either faster or slower. One scheme is magnetic recording of the pulse trains on either wire or tape. Magnetic pulses cannot be packed more closely than about 200 per inch if they are not to overlap and become incapable of resolution. The reprate of reading and writing magnetically for a given packing is proportional to the speed at which the wire is transported. Hence we can magnetically record pulse trains leisurely and run them into the machine rapidly or conversely, can record fast signals on a fast wire, and later read the wire at a speed which an electric typewriter can reliably be expected to follow.

Inside the machine we need two types of memory, one that stores a train of pulses statically and another that stores the high reprate trains of pulses.

STATIC REGISTER — The first of these, the static register, is needed, among other places, in the arithmetic unit, to set up central voltages in accordance with the 0's and 1's of a number. Basically a static register is a flip-flop such as that of Fig. 1B which has two stable states. High and low plate voltages can be taken to represent the storage of a 1 or a 0.

In a practical flip-flop, grid capacitors are used to speed transition from one state to the other. Minimum transition time depends upon mutual conductance of the tubes. A more rapid flip-flop than the one shown can be made by using such tubes as the 6AK5, connected either as pentodes or triodes. Provision is also made for setting the flip-flop in either state by applying a negative pulse to the appropriate tube. The diodes are isolation buffers to disconnect the pulse sources when pulses are not being applied. This not only reduces loading on the transfer pulse from one tube to the other, but also prevents this pulse from being transmitted to other flip-flops via the input circuit.

Tying the two input leads together provides a binary counter. The plate-grid coupling capacitances provide enough memory (time lag) for the flip-flop to remember in which state it was prior to the application of a pulse applied to both tubes. As a result, an input pulse changes the state of the flip-flop and provides a scale-of-two, or binary counter. Cascaded binary counters have many applications. For binary counter purposes, the grid input arrangements can be omitted and a positive pulse applied to the common cathode lead.

By using 35 flip-flops, one for each binary column, we can statically store a 35 place binary number. Writing a number into a register consists of setting its flip-flops in accordance with the succession of 0's and 1's in the binary number. Reading the register consists of causing it to generate the pulse train corresponding to its array of 0's and 1's.

FEEDING REGISTER — There are

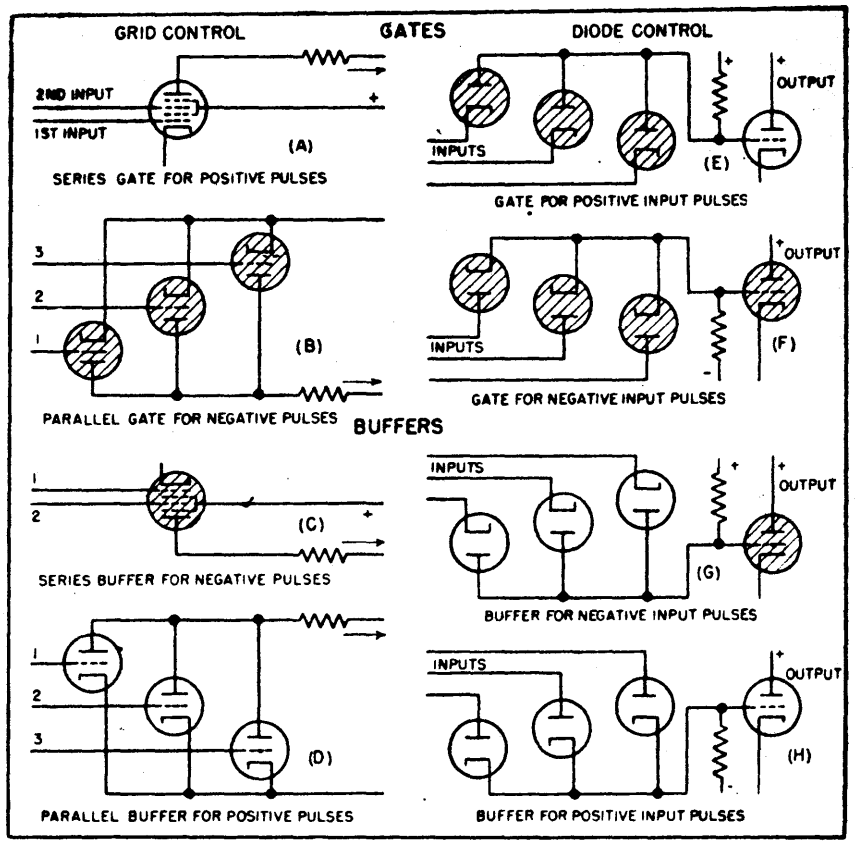


FIG. 2—Gates and buffers constitute the operating elements of the arithmetic units. Germanium diodes may be used for compactness

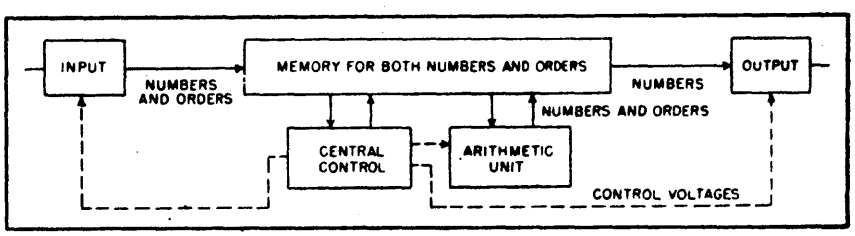


FIG. 3—Basic functional components of digital computer, and their interrelation

3 of 8

two ways of converting a serial train of pulses into the parallel form for storage in the static register. The pulses can either be fed into the register from the end or set up in parallel alongside it.

The latter scheme is indicated in Fig. 1C; the train of pulses is fed into a delay line of 0.5 μs sections, so that just as the last pulse appears at the input the previous pulses appear at the various junctions. The delay line thus momentarily converts the serial pattern of voltage peaks versus time into a spatial pattern of voltage versus position; voltage appears at the junctions corresponding to the positions of the binary 1's in the number represented, no voltage appears at the positions corresponding to 0's. When this space pattern is obtained, all the gates are opened by an activating pulse, and the 1's are entered into the register via the set 1 input leads. The register can be cleared by applying a pulse to the set 0 inputs.

If the plate outputs of the flip-flops are connected to successive junctions of a duplicate delay line, clearing the register (by simultaneously setting all flip-flops to 0) will introduce pulses into the line at the 1 positions; these pulses will come out of the delay line as the desired train.

The other scheme for sending a train into a static register is somewhat similar to the operation of some desk computing machines that have only 10 keys, 0 through 9. Pushing 3 enters 0003 on the dials, then pushing 5 shifts the 3 along as the 5 is entered, showing 0035, etc. This sequential to parallel conversion can be accomplished by the shifting register of Fig. 1D.

The set 0 lines are all connected to a shift pulse bus. A shift pulse then clears all flip-flops, and any registering 1 generate output pulses. These pulses arrive at the set 1 leads of the next flip-flops, transferring the 1's one place to the right. Clearing a flip-flop registering 0 generates no pulse, so leaves the next flip-flop cleared to 0. Hence every time a shift pulse is sent in, the contents of the register shift to the right. If the shift pulses come at a 2-mc reprate, evenly interspersed between the 2-mc signal pulses sent into the left-hand flip-flop, every time the register is shifted it will find the next digit of the train in the left-hand flip-flop and 35 shifts will result in a static storage of the 35 pulses in the train. We now stop the shifting and have the number stored.

Reading the register (regenerating the train of pulses) is simple. The output of the right-hand flip-flop is connected to a transmission bus and 35 shifts are made, sending the successive 1's and 0's onto the line, and leaving the register cleared to all 0's, assuming that no signal is coming in from the left.

The static registers described above require two tubes per binary digit, or 70 tubes per word stored, so are uneconomical for the main storage. (A general purpose computer needs storage facilities for at least 1,000 words). However the static register is useful in the arithmetic unit for intermediate storage between two organs with different speeds, such as internal parts of the machine and the magnetic wire. One word at a time can be written at any speed, and then read at any other, permitting synchronizing input data pulses with the 2-mc clock, which would be impossible to do by trying to run the wire at an exact speed.

The other internal high-speed memory, or scratch paper, of the machine can either hold pulse trains as a static array, or remember them dynamically; that is, in the form of pulse trains available for retransmission on demand. Only the latter choice will be discussed here.

DYNAMIC MEMORY—The simplest way of achieving dynamic memory is to feed pulses into a delay line whose output is connected back to the input to keep the pulses circulating. An amplifier and pulse regenerator are needed at the delay line output to compensate losses. Distorted pulses from the line are used to control a gate feeding fresh pulses from the master pulser, or clock, back into the line. Such a gating combination in the recirculation system is referred to as a pulse reshaper.

The losses of an electric delay line are too great. Each word to be stored requires 17.5 μs of line to hold it; this implies a total of 17.5 milliseconds of electrical delay line, whether in one or several segments. To transmit the individual 0.2 μs pulses without excessive distortion requires a bandwidth of 10 mc. Even with the optimistic figure of 6 db per μs attenuation in lines having this bandwidth, attenuation would be 105,000 db, requiring 7,000 tubes such as the 6AK5 having a gain of 15 db per stage. This is excessive.

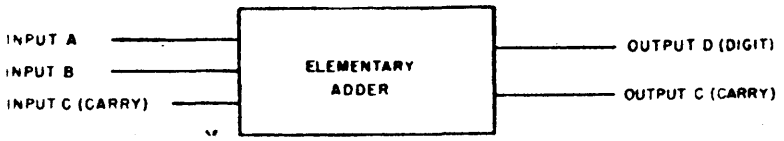
A practical way to simplify dynamic storage is to store pulses acoustically rather than electrically. We can convert the 0.2 μs pulses into 0.2 μs packets of h-f using a carrier frequency of 20 or 30 mc. These h-f pulses can then be used to drive a quartz crystal which in turn generates waves in a mercury column. A receiving crystal at the far end senses these waves giving a signal that is amplified and rectified to regenerate the pulses. Attenuation in mercury is approximately 0.06 db per μs at a carrier frequency of 30 mc, or one percent of that for the electrical. The pair of crystal transducers with the line introduces a loss of about 50 db.

If one long delay line is used, coupling losses would be negligible, but a single delay line of 17.5 milliseconds would require on the average a waiting time of 9 milliseconds before the desired word would be available. This is too long. A practical compromise between equipment and speed is to subdivide the memory into lines, or tanks, of 20 word capacity, each having a delay of 350 μs. Thus 50 lines are needed involving 50 pairs of transducers having 2,500 db attenuation. Adding the attenuation of 1,050 db in the mercury, we have a total of 3,550 db attenuation (to be compared with the 105,000 db of electrical lines) and requiring only about 250 amplifier tubes. A typical recirculating tank circuit is shown in Fig. 1E.

We now have conceptually a source of input signals, a receiver for output signals, an arithmetic unit, static registers and dynamic memory tanks. Signals must be dispatched from one to another of

Table I—Operation of an Elementary Adder

Terminals of Elementary Adder



List of Binary Input-Output Combinations

In A	0	0	0	0	1	1	1	1
In B	0	0	1	1	0	0	1	1
In C	0	1	0	1	0	1	0	1
Out D	0	1	1	0	1	0	0	1
Out C	0	0	0	1	0	1	1	1

Rules of Arithmetic

Binary operations	Logical concepts
1) A single input 1 generates a 1 and no carry	(1) (A AND B) or (A AND C,) or (B AND C) generates a carry
2) Two input 1's generate a carry but no output	(2) A or B or C generates an output digit unless one of the above AND combinations occurs, which operates a gate to prevent the transmission of the digit
3) Three input 1's generate both and output and a carry	(3) A AND B AND C generates both digit and carry

Functions of Elementary Adder

Transmit a digit if A OR B OR C and not A AND B, A AND C, nor B AND C, or if A AND B AND C
Generate a carry if A AND B, A AND C, or B AND C

these organs. In general, any organ may be called upon to send signals to any other. The simplest way of doing this is to connect all tank inputs to a common point through switches (electronic gates) and to connect the arithmetic unit output to this point. Then opening the proper gate will allow the signal to proceed to the chosen tank, and to [illegible]ther. Conversely, if several [illegible]s are to be capable of sending [illegible]veral receivers, all sources can [illegible]e connected in parallel to a common transmission bus, and the receivers connected to this bus through gates. Then by opening a receiver gate, and instructing the proper source to transmit, the desired result should follow. In practice, this would not work, for with many sources in parallel, each source would be loaded by the parallel combination of the output impedances of all the others. We need, between each source and the common bus, a buffer which allows only one way traffic, so that a signal can come from a source through the buffer to the bus, but the other sources cannot load the bus. The use of a buffer between an oscillator and a modulated r-f amplifier is well known. In our case of passing pulses of only one polarity, we do not need a triode or pentode buffer, but can use a diode. This diode is normally biased with back voltage so that it presents a high impedance to the common bus. A pulse on the bus increases the back voltage on the diodes and is protected. A pulse from a source, however, reverses the polarity on that one diode and goes through with small loss. The advantage of such buffers is that germanium diodes can be used, greatly reducing shunt capacitance.

With gates and buffers we can perform circuit switching, or spatial selection for traffic control. If we stored our 1,000 words in 1,000 one-word tanks, there would be an exorbitant number of switches with their attendant losses and control problems. We could compromise on 50 tanks holding 20 words each. We can choose any one of these 50 tanks by spatial switching and any one of the 20 words in a tank by temporal selection. The temporal selection requires no switches aside from the timing gate.

The timing circuit can be operated by dividing the master clock rate. The 2-mc reprate drives a counter which counts up to 35 and then throws a flip-flop, giving an output which is on for 35 pulses, or one word time, and off for the next. By feeding these rectangular waves of word duration into a scale-of-20 counter, we can devise a circuit which will give an output (to control a gate) for the duration of any desired one of the twenty words.

Arithmetic Circuits

To understand how to combine gates and buffers to make a circuit that will do arithmetic, it is convenient to interpret gates and buffers in terms of their logical behavior.

A GATE is essentially a device having two inputs and one output. Either input can be considered as the signal, and the other as the control. Obtaining output from a gate is dependent upon stimulating both inputs; that is, it requires stimulation of one input AND the other input. Logically the gate detects the AND concept, one thing AND another.

BUFFERS, on the other hand, that feed two or more signals to a common point give an output signal if any one of the sources is excited; that is, if one OR another input of the row of buffers is stimulated. Hence two buffers connecting two inputs to one output constitute the logical concept of OR, one signal OR another.

Typical gate and buffer circuits using tubes are shown in Fig. 2. The series gate of Fig. 2A has both grids normally biased beyond cutoff; both must be driven above cutoff to produce an output. The parallel gate of Fig. 2B has all tubes normally conducting. If the load resistor is large compared to the conducting resistance of a single tube, the common plate voltage will remain low unless all tubes are cut off by signals.

The series and parallel buffers of Fig. 2C and 2D represent inverse operating conditions on the corresponding gate circuits. The nor-

mal-abnormal conduction states are interchanged, and the circuits are stimulated by pulses of sign opposite to those required by the corresponding gates. A signal on any input produces a change in the output.

The diode circuits of Fig. 2 are all parallel circuits. Gates, requiring the AND or multiple coincidence, have all their diodes normally conducting, while buffers have all their diodes normally nonconducting. Diodes are generally of the germanium type.

Adder Is Basic Element

To add two digits, the basic operation of arithmetic, we need two inputs and one output. If the sum of the two digits is greater than 9 in the decimal system, or greater than 1 in the binary system, a carry will be produced to add in the next digit position. Hence we need three inputs, one for each digit in the given position, plus one for the possible carry from the previous position. We also need two outputs, one for the output digit, and one for the carry. Thus each digit position requires a device as shown in Table I. Operating characteristics of this elementary adder can be deduced from the laws of arithmetic. The desired outputs for the eight possible input combinations of 0 and 1 on the three inputs are listed in the table.

There are two types of adders: parallel and serial.

A PARALLEL ADDER is made of 35 elementary adders, one for each digit position. Various digits are set up in a static register, as previously discussed, and the steady register output voltages representing 0's and 1's activate static elementary adders. The carry output lead of each place can be permanently connected to the carry input lead of the next, requiring one type of elementary adder to satisfy the rules of arithmetic. Alternatively the sum and carry digits can be formed statically in each place, and the carrier transmitted to their neighboring adders an instant later. Part of the difference in the circuitry is involved with the fact that a carry may generate a carry, as in adding 7774 to 2226. Propagation of the carry down the line can be handled in various ways.

THE SERIAL ADDER uses a single complicated elementary adder for successive digit places in sequence. Pulse trains are not set up in static form, but are fed in dynamically, the two numbers arriving simultaneously. If an output 1 pulse is generated, it is transmitted immediately as one digit of the sum. If a carry pulse is generated, it is delayed 0.5 μs and returned to the carry input, arriving there coincident with the input digits of the next place.

An elementary adder can be made of gates and buffers. Rules of arithmetic shown by the list of input digit combinations are stated in Table I. The preventing operation in case (2) implies a negative gate, or logical AND NOT, which is easy to devise from diodes by using several bias levels. With this terminology, the functions of an elementary adder can be described logically as at the bottom of the table. The complicated combinations of AND and OR are straightforward logically and electronically, but lead to a practical circuit employing (in one design) nine pentodes and 36 diodes! Some of these elements are incorporated to reshape pulses, and several diodes are used as limiters and d-c level restorers.

Any adder can be considered as a problem in traffic control where the signals (numbers) that are put in control the transmission of pulses throughout the adder. This local control is one step more complicated than the central control, or traffic dispatch between organs. In the central control problem, control voltages set up the paths to be taken by signal pulses. In the local control, pulse paths, and times (clock beats) at which pulses occur are set by the signals themselves, so that there is no longer a clear-cut distinction between signal and control pulses.

MULTIPLICATION is a more complex problem. Ordinary longhand multiplication consists essentially of adding the multiplicand (574) as many times as the right-hand digit of the multiplier (31) shifting columns, adding on the multiplicand [illegible] times as the next digit, [illegible] the example:

	574
	31
=	
	574
1722	574
	574
17794	574
	17794

Because in the binary system, only 1's and 0's occur, we have for the partial products either the multiplicand itself, or zero.

Decimal		Binary
23		10111
5		101
115	=	10111
		00000
		10111
		1110011

This allows us to use a shifting register (previously described) together with a basic adder, to perform multiplication. We do or do not add in the multiplicand according to whether the right-hand digit of the multiplier is 1 or 0, shift the number in the register, and repeat. Thus a basic arithmetic unit consisting of registers, which can be shifted when desired, and buffers, can either add according to whether it gets a simple signal to add, or whether it gets also a signal to shift and repeat. Other modifications permit subtraction and division. Which operation is to be performed is controlled by signals from central control, usually quasi-static voltages to keep certain gates open until the operation is completed.

Before examining means for converting pulse trains representing arbitrarily coded orders into gate control voltages, let us glance at the overall organization of the computer.

The input portion of the machine sends all its words, both numbers and orders, to the high speed memory storage. From storage, orders go to the central control, logically through a decoder, but this decoder is the main part of the central control and so is not usually considered separately. Central control must dispatch operating instructions to all machine units, including the input, for it must tell the input when there is room in the memory for more data and orders

to continue the problem. The general scheme is shown in Fig. 3. The only feature of the diagram that is unnecessary is the transmission of orders (not control voltages) to and from the arithmetic unit. This is a useful way of pyramiding the hierarchy of control to achieve versatility of operation. Because orders themselves are coded to appear as numbers, orders can be modified by performing arithmetic upon them. This feature simplifies programming the mathematical problem in terms of dispatching orders, but need not concern the electronic circuit designer.

We have mentioned that orders are coded in numerical form. Suppose for example that eight different orders are desired; that is, eight different lines are to be energized. Any eight things can be represented in code form by the binary numbers 0 to 7; that is, 000, 001, 010, 011, 100, 101, 110, 111. These are the eight combinations of three places, each having either of two values. Electrically, we can have three wires, each of which may have voltage applied. If orders are pulse trains they can be converted to the static three wire combination by setting up a static register of three flip-flops. We then have three wires, any one or more of which may be hot, representing eight different possibilities, and we wish to excite any one of eight leads in accordance with these choices. In general, we have N wires of two possible states each (hot or cold) giving 2^N combinations, and wish to excite only one of 2^N outputs. In practice, instead of using N wires from N flip-flops, having some hot and some cold, it is better to bring two wires from each flip-flop, one from each side. We then have N pairs of wires, each of which has only one side hot. All input pairs are thus excited one way or the other, avoiding complications of zero-voltage input signals.

The simplest case of a decoder is where $N = 2$, so that there are two input pairs and four output leads. The circuit of Fig. 4A shows this case. The horizontal and vertical lines are connected through diodes, so that the diodes in any column form a gate, or AND circuit. If upper and lower lines of the top pair are excited positively, output from the left-hand lead is excited, and so on for the four possible combinations of input.

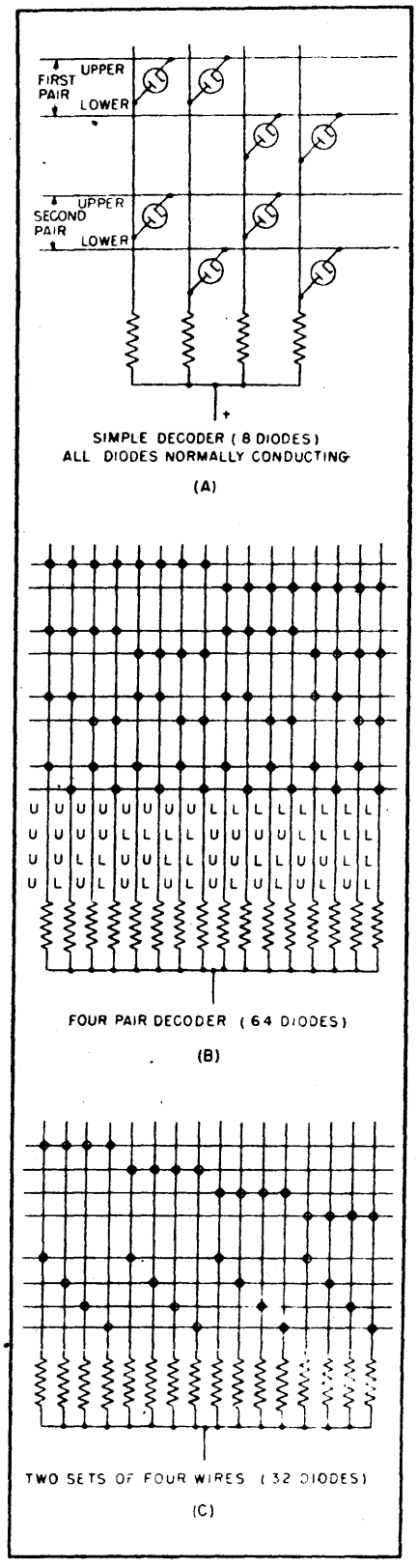


FIG. 4—Switching circuits use unidirectional conductance of diodes

For larger decoders, it will be convenient to indicate the presence of a diode connection between two lines by a circle at the crossover. There are no direct connections. Figure 4B shows a simple decoder for four input pairs, yielding 16 possible output excitations. Combinations of upper and lower pair excitations that result in excitation of each of the 16 lines are indicated on the figure.

This direct check of the possible combinations can be called a one-stage decoder. Fewer diodes are required if we decode in two stages, namely, by mixing two pairs as in Fig. 4A to get one line out of four, and doing the same with the other two pairs to get one line out of another set of four. We then have two sets of four lines each, in which only one line of each set is excited. These two sets can be fed into the circuit of Fig. 4C. Thus in using Fig. 4B, each output line requires a quadruple coincidence for excitation, and 64 diodes are needed. By using two circuits of Fig. 4A and one of Fig. 4C, making successive simple coincidences, we need $8 + 8 + 32 = 48$ decoders, or a saving of 25 percent.

Multistage decoding exhibits even greater savings as N increases. For $N = 8$, allowing selection of any one of 256 memory tanks by virtue of the $2^8 = 256$ different gates that may be opened by an 8-pulse signal, a three-stage decoding requires only 608 diodes as against 2,048 for single-stage decoding.

Traffic Handling Systems

Having seen how a coded order can be converted to the selection of a gate opening voltage, it is of interest to consider briefly the general traffic handling plan. The mathematician prepares his instructions to the machine in terms of numerical data, coded orders to select which basic operation the arithmetic unit is to perform, for sequencing the machine or for expressing the routine to be followed. In general two kinds of words are put into the machine memory: numbers and orders.

Assume that the memory is capable of storing 1,000 words and, for

simplicity, that the two kinds of words are of equal duration, or number of pulse positions. These 1,000 words occupy definite positions in two dimensional space-time. Hence we can consider their positions as pigeonholes numbered from 1 to 1,000 and call for transmission of a word to or from any pigeonhole. The simplest way of entering the input data is to take the first thousand words from a magnetic wire and store them sequentially in the thousand cells. This can be done by using a counter to measure off a word, and cause unity to be added to the address to which the next word is to be sent.

High speed reading of the memory can also be done sequentially by giving the address 1 as the instruction for the cell to be read and by having a built in arrangement for automatically adding unity to the address of the cell to be read. It will then automatically read cell 2 as soon as it has finished with cell 1 and is ready to read again.

A procedure that may be more flexible for repeating subsequences and setting up branch operations (choice of next order depending upon present results) and also more convenient in practical programming is the four address code. In this system each order is composed of four addresses (or memory cell locations): the address of the first operand (number to be arithmetically operated upon), the address of the second operand, the code for the operation to be performed, and the address of the next order to be read after completion of the present instructions. This system is more efficient if memory reference is slow compared to other operations; that is, if waiting time for a word to be reached in the sequential reading of a dynamic memory is relatively large because it allows the essentially simultaneous look-up of both operands.

A variation of the four address system is the use of a fifth address in the words on the input wire, to designate the cell into which that word is to be stored. The fifth address is automatically deleted as the word is entered into the machine.

In electronic digital computers, the tubes, for example, are called upon to develop a pulse of usable level, or not called upon at all. Variations between tubes, aging, or tolerances of resistors do not affect accuracy, until they become so extreme that the signal falls out of usable range. A ten to twenty per cent variation of signal strength has no effort on a series of pulses. Ideally a computing machine works perfectly or not at all. Actually, as tubes deteriorate, there is a threshold at which operation may be erratic. By setting a limit checking circuit for a safe level margin, this otherwise possible operation can be put in the class with complete breakdown.

Errors can occur due to noise generating a false pulse at an allowed pulse time when the word transmitted has a zero in that position. This noise pulse may be indistinguishable from a proper pulse. Occurrence of errors due to such random causes can be guarded against by one of several checking schemes.

One of the most elaborate checking schemes that has been proposed is to check the arithmetic and the transmission. The arithmetic can be checked in a fashion similar to the ancient system of casting out 9's, where each number is expressed as its excess over a multiple of 9; that is, it has a value of 0—8. This is done by adding sideways. The 9's excess of a sum of numbers is equal to the sum of their individual excesses, (expressed as an excess if larger than 9). The 9's excess of the product of two numbers equals the (excess of the) product of their excesses. A simple auxiliary addition or multiplication on the excesses has often been used for checking arithmetic. For example, multiplying 371 by 24 gives 8904. The 9's excess of 371 is found by adding the digits $3 + 7 + 1 = 11$, $1 + 1 = 2$. Similarly, the 9's excess of 24 is 6. The product of these two excesses is 12, having itself an excess of 3, which agrees with the excess of 8904, $8 + 4 = 12$, $1 + 2 = 3$. A corresponding procedure of casting out $(2^r - 1)$ can be set up for binary computation, and a small auxiliary arithmetic unit operated simultaneously with the main unit.

This type of checking lends itself to verifying correct transmission of a number. The excess count of a number can be stored with it in the memory for performing the parallel arithmetic check. It can be used as a transmission check by taking the excess count of a number received by the arithmetic unit and comparing it with the received check count. Very peculiar transmission errors are required to make the new count of an incorrectly transmitted number agree with either its original count or an incorrectly transmitted count. This type of checking is based on arithmetic.

Checking the address selection exercised by central control can be done by storing with each word its address. When the word and accompanying address is read, the read address is checked against the called-for address. This checks both the spatial and temporal phases of word selection in the machine.

Electronic design of machines is fast progressing to the point where they will be more perfect than the mathematics set up for them. I refer to such varied factors as round-off error, inevitably introduced by working to a fixed number of significant figures. If a machine performs 1,000 arithmetic operations a second for days on end, what relationship does the final answer have to the original hypotheses? Some mathematical research is being done on this point. A more vital question is the design of mathematics suited for machines. Many procedures use machines for replacing human computers, using numerical computational schemes developed for the human brain. Characteristics of an electronic machine are different from those of a human brain, and it is reasonable to suppose that computational procedures can be devised which, although unsuited for hand computing, are well adapted to machine routines. Such procedures have been developed for a few special problems.

The writer thanks the Raytheon Manufacturing Company and the Eckert-Mauchly Computer Corporation for supplying some of the circuit details shown in the figures.