5

Technical Embodiment of Finite Automata and Sequential Machines

5.1. TWO METHODS FOR TECHNICAL REALIZATION OF FINITE AUTOMATA AND SEQUENTIAL MACHINES

In the preceding chapters we have formally introduced the concepts of "finite automaton," "sequential machines," and "abstract structure." So far, these were presented only as equations or systems of equations, and we did not deal with the physical nature of the dynamic systems whose motion they describe. Now we shall show that the above concepts describe important technical systems, and we shall introduce techniques for determining the hardware needed for realizing any given finite automaton or s-machine.

We have shown in Chapter 4 that each finite automaton or s-machine may be represented by many abstract structures. But each abstract structure may be embodied by some practical device that functions just like this abstract structure. It follows that any finite automaton can have many technical embodiments. We shall also show that any given abstract structure of any given automaton may be embodied (realized) by many technical means.

In this chapter we shall consider only embodiments (realizations) of binary abstract structures; that is, it will be assumed that the finite automaton is given by a system of relations

$$x_i^{p+1} = F_i\left[x_1^p,\ x_2^p,\ \ldots,\ x_n^p,\ u_1^p,\ u_2^p,\ \ldots,\ u_s^p\right] \quad (i = 1,\ 2,\ \ldots,\ n), \tag{5.1}$$

where x_i $(i = 1, 2, \ldots, n)$ and u_j $(j = 1, 2, \ldots, s)$ are logical variables which can be only 0 or 1, and F_i $(i = 1, 2, \ldots, n)$ are logical functions, which also can be only 0 or 1. We also assume that the timing of the automaton is given, that is, we are given the conditions defining the occurrence of the discrete moments $0, 1, 2, \ldots, p$ on the continuous time scale.

To produce a technical device performing relation (5.1), one must have logical converters performing the functions F_i. We have already described such devices in Chapter 2. Now, however, we do not want to perform functions F_i themselves, but want to embody relations (5.1) of which such functions are a part. Thus, we are faced with the question: What modification must be introduced into the function converters of Chapter 2 (or with what must these converters be supplemented), in order to transform them into devices whose states shall vary in time so as to model the abstract structure (5.1)?

We shall now present two essentially different methods for solving the above problem.

5.2. AGGREGATIVE DESIGN OF FINITE AUTOMATA AND SEQUENTIAL MACHINES

We already know that an abstract structure such as (5.1) can be placed into correspondence with a structural diagram. Such a diagram (for $n = 3$, $s = 2$) is shown in Fig. 5.1. The diagram contains s input lines (input wires $u_1, u_2, \ldots, u_s$) and n output lines (their coordinates are states $x_1, x_2, \ldots, x_n$). Each of the n logical converters performing functions $F_1, F_2, \ldots, F_n$, respectively, receives signals from all the $n + s$ lines; the output of the ith converter feeds the line x_i via a one-instant delay element (denoted by a circle in Fig. 5.1), whose output and input are related by

$$x_{\text{out}}^{p+1} = x_{\text{in}}^{p}.$$

Direct examination shows that such a circuit models precisely the structure of relations (5.1). To construct a technical device according to this diagram, one must have one-instant delay elements in addition to the requisite logical converters. Thus in order to convert a set of elements sufficient for the embodiment of any logical function into a set of elements sufficient for the realization of a finite automaton, one needs only to supplement the first set with a single element—a one-instant delay element.

Such a set is also sufficient for the construction of any sequential machine, since the latter differs from an automaton only in having an output logical converter.

A one-instant delay element must have two inputs—the basic input x_{in} and an auxiliary (time) input x_t. It also must have an output x_{out}. The conventional notation for such an element is shown in Fig. 5.2.* The auxiliary (time) input receives the signals indicating

*One usually omits the input wire x_t whenever such an omission does not hinder the understanding of the operation of the circuit.

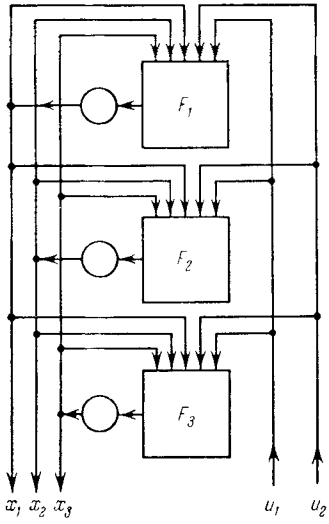

Fig. 5.1.

the occurrence of the next discrete moment, such signals being supplied to the automaton from an external signal-producing device (a "clock" or "synchronous source").

The delay element operates in the following manner: let x^*_{in} be the state of the input to the element at the first discrete moment. Then, after a short

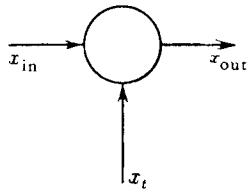

Fig. 5.2.

time interval τ (the specific delay of the delay element), the output shows $x_{out} = x^*_{in}$.

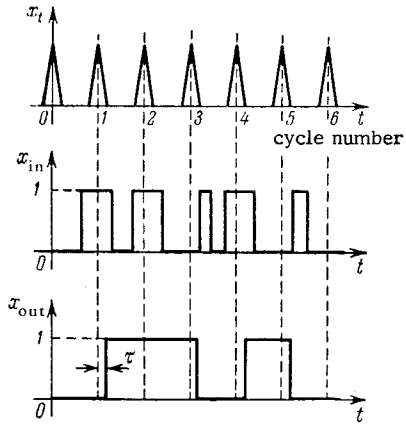

Fig. 5.3.

After this, regardless of what happens at the input, the output will retain its value until the next discrete moment, when the same procedure is repeated. The delay element does not react to any changes occurring at the input during the time interval between the discrete moments.

Figure 5.3 shows an example of changes occurring at the input and output of a one-instant delay element. In this example the synchronizing signals are short pulses. However, the synchronizing signal often is each change of state of the auxiliary input, which can also have only two values: it can be either 1 or 0 (Fig. 5.4) or, alternatively, it can only change from state 0 to state 1 (Fig. 5.5).

Consider the construction of a pneumatic one-instant delay element. Such an element is based on so-called memory cells. Schematic diagrams of two types of memory cell are shown in Figs. 5.6,a and b, respectively, where the change of state of the "time input" P_t from 0 to 1 serves as the synchronizing signal. A memory cell consists of two pneumatic relays (see Section 2.4). One of these (the output) is connected so as to perform a "repetition," maintaining the output pressure P of the cell equal to pressure P_h; the other relay (the input) performs the function of a pneumatic valve, opening or closing the connection between the chamber where the pressure P_h is established and the input line P_1. The operation of the pneumatic valve is governed by the pressure P_t; in the cell of the first type (Fig. 5.6,a) the valve is closed when $P_t = 1$ and open when $P_t = 0$, and, conversely, in the cell of the second type (Fig. 5.6,b) it is closed when $P_t = 0$ and open when $P_t = 1$. Because of this arrangement, either the cell output is equal to its input (for the first cell when $P_t = 0$, and for the second cell when $P_t = 1$), or the output is not connected with the input and is constant (in the first cell when $P_t = 1$, and in the second cell when $P_t = 0$), its value being determined by the magnitude of the pressure P_h in the dead-end chamber.

A memory cell of the first type connected in series with a cell of the second type constitutes a one-instant delay element (Fig. 5.6,c). This element operates in the following way: at t_n, when P_t is 1 (the beginning of the nth discrete moment), the first cell "memorizes" the value of the input, that is, $P^*(t_n) = P_1(t_n)$. In the same instant (more precisely, at time $t_n + \Delta t$, where the increment Δt is caused by the fact that the working membrane of the second memory cell must travel a longer path than that in the first one), the second memory cell transfers the value remembered by the first cell to the output of the system: the pressure $P(t_n) = P^*(t_n) = P_1(t_n)$ is thus established at the output of the delay element. After this, as long as $P_t = 1$, there can be no changes in the system, since its state is determined by the fact that throughout all this time the first cell "remembers" the input value $P_1(t_n)$. This means that $P(t) = P^*(t) = P_1(t_n)$ when $t_n \leqslant t \leqslant t'_n$, where t'_n is the instant at which P_t becomes 0.

At time t'_n (see Fig. 5.6,d) the input to the second memory cell is $P^*(t'_n) = P_1(t_n)$; the cell "memorizes" it, and there is thus no

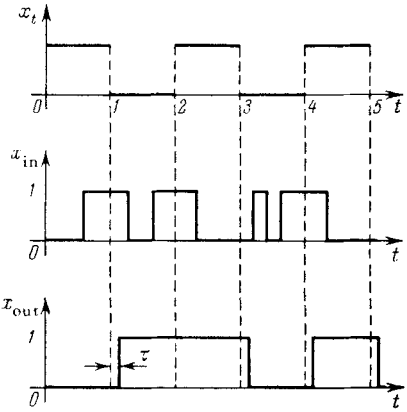

Fig. 5.4.

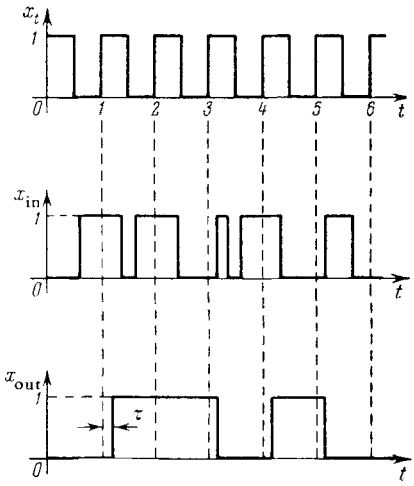

Fig. 5.5.

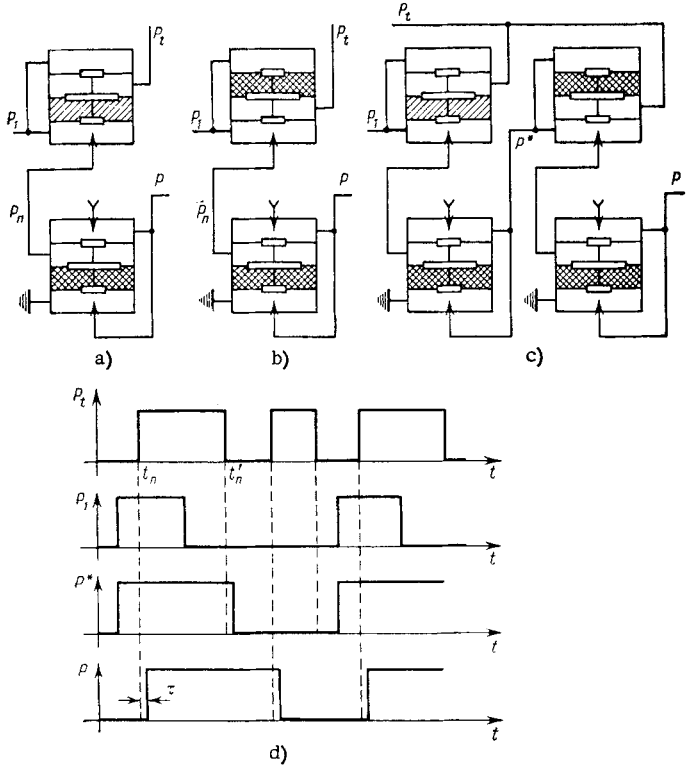

Fig. 5.6.

change in the output of the system, which is still at $P(t'_n) = P_1(t_n)$; at time $t'_n + \Delta t$, the first memory cell starts to operate as a repeater. Subsequently, as long as $P_t = 0$, the output of this system (that is, the delay element) will remain unchanged; it can assume a new value only if, at time t_{n+1}, the control P_t(the time input) becomes 1 again.

Thus this pneumatic device performs the function of a one-instant delay element: its output at the instant of the synchronizing signal (when $P_t = 1$) becomes equal to the input and then, no matter what happens at the input, remains unchanged until the following synchronizing signal (compare Fig. 5.6,d with Fig. 5.5).

Figure 5.7 shows an electromechanical embodiment of a one-instant delay element, which has many conceptual similarities to the above pneumatic delay element. Again, we have two inputs, X and X_t, where X_t is the time input—the change of X_t from 0 to 1 being the synchronizing signal for the delay element. Again the circuit consists of two series-connected memory cells (*1* and *2* in Fig. 5.7,a). The state of the relay coil Y is the output of the element.

The cells memorize by using blocking contacts (contact y^* in cell *1* and contact y in cell *2*). The time input X_t acts on the cell via its associated contacts x_t and $\bar{x}_t$ in such a way that when the first

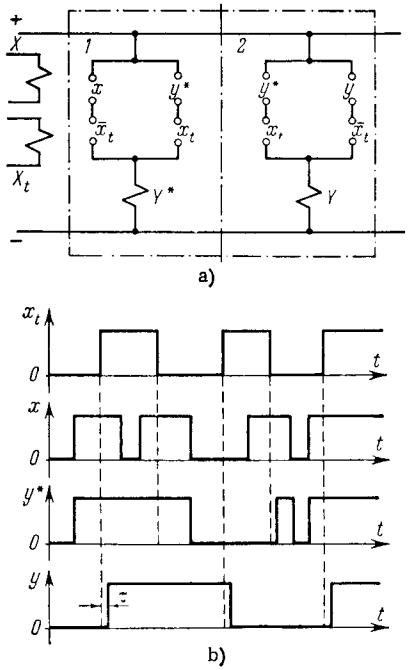

Fig. 5.7.

cell "memorizes" (this will occur at $X_t = 1$), the second cell operates as a repeater of the first one ($Y = Y^*$), and, conversely, when the second cell memorizes (at $X_t = 0$), the first cell repeats the input ($Y^* = X$). Figure 5.7,b shows that this system operates in precisely the same manner as the previously described pneumatic delay element.

As we stated before, a delay element consisting of two memory cells can operate correctly only if the theoretically simultaneous change of state of the cells actually takes place in a certain specified sequence: that is, both cells respond initially by remaining in a state of "memorizing," and only then does one of the cells transform its state into that of a repeater. In a pneumatic element this is achieved by applying differing back pressures P_{h1} and P_{h2}, whereas in the relay switching element this same function is filled by the specific delay τ of relays Y and Y^*.

Any finite automaton may be embodied by replacing the contacts x ($x_1, \ldots, x_s$) of the delay element circuits (such as that of Fig. 5.7) with chains of contacts $f_1, f_2, \ldots, f_n$. Such chains not only incorporate the input contacts $x_1, \ldots, x_s$, but also include the contacts $y_1, \ldots, y_n$ of the output relays of the delay elements. This is shown by the circuit diagram of the automaton (Fig. 5.8). Thus the $u_1, \ldots, u_s$ states of the input fibers of the automaton of Fig. 5.1 correspond to the $x_1, \ldots, x_s$ states of the input contacts of Fig. 5.8 and the $x_1, \ldots, x_n$ states of the automaton of Fig. 5.1 correspond to state of the relay coils $Y_1, \ldots, Y_n$ of Fig. 5.8, and, finally, the logical converters $F_1, \ldots, F_n$ of Fig. 5.1 correspond to the chains of contacts $f_1, \ldots, f_n$ in Fig. 5.8.

Obviously, the one-instant delay element is itself the simplest finite automaton. If one desires to assemble not merely logical converters but also automata then the set of constituent elements must include either a one-instant delay element or some other elementary

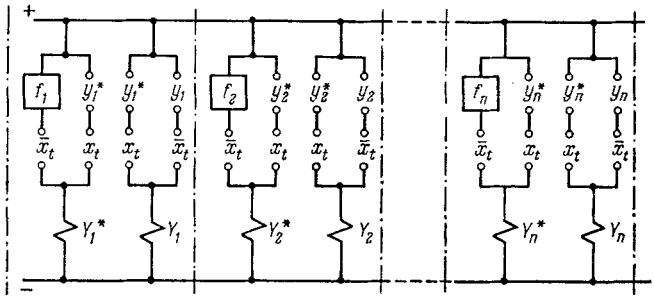

Fig. 5.8.

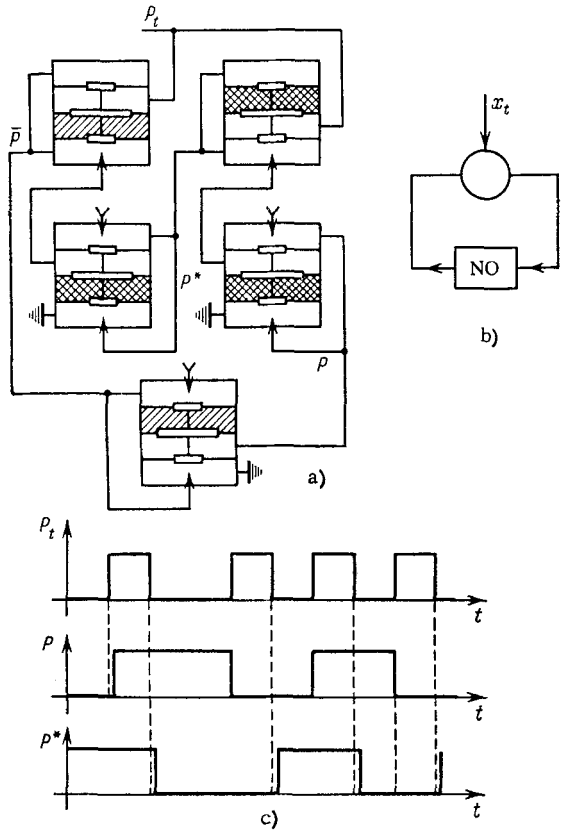

Fig. 5.9.

(nonautonomous) finite automaton. In another widely used method, one supplements the logical elements with an elementary automaton which, although it does not permit construction of all conceivable finite automata, does give many finite automata of practical value. One such elementary automaton is the complementary flip-flop (an autonomous automaton). Figure 5.9 shows a gas-operated flip-flop based on a pneumatic delay element. This flip-flop (Fig. 5.9,a) is obtained from a delay element by switching its output into its own input via a negation element (Fig. 5.9,b). Such a circuit is an autonomous finite automaton operating according to $x^{p+1} = \bar{x}^p$ (here, pressure P substitutes for x), an operation shown in Fig. 5.9,c. Figure 5.10 shows an electromechanical flip-flop, also made from a delay element by switching its output into its own input via a negation element.

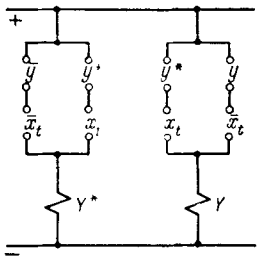

Fig. 5.10.

This technique for the synthesis of automata involves supplementing the set of instantaneously acting logical elements with some very simple automaton (for example, a one-instant delay element, a flip-flop, and so on). In addition, in using delay elements, this technique assumes the availability of a synchronous source whose output becomes the time input of the delay elements. In many cases, however, there is no need for supplementing the logical set with new elements: one merely utilizes the fact that any real element has an inherent time lag τ; that is, any real element is an elementary automaton operating in a discrete time scale devised by dividing the time axis into uniform intervals of length τ. The realization of this fact leads to the most popular (although somewhat limited) technique for synthesizing automata. This technique is applicable when the synchronizing signal, defining the division of the continuous time into discrete moments, is the change of the input state of the system.

5.3. SYNTHESIS OF FINITE AUTOMATA AND SEQUENTIAL MACHINES BY UTILIZING INHERENT DELAYS AS WELL AS FEEDBACK

Consider again the simplest electromechanical relay, which in Chapter 2 was assumed to be acting instantaneously. In fact, however, relay actuation involves a short time lag τ. This means that even though the output (the state of contacts x) and the input (flow of current in coil X), are both logical variables (that is, can only be 0 or 1), their relationship involves a time element. Thus

$$x^{t+\tau} = X^t.$$

If time is uniformly divided into a succession of discrete moments $0, \tau, 2\tau, 3\tau, \ldots$ and if changes of the input as well as all outputs occur only at these moments, we get

$$x^{p+1} = X^p,$$

that is, the relay* is an elementary automaton of the "one-instant time delay" type, operating at intervals τ.

*We are referring here to a relay with normally open contacts. If the actuation time is also taken into account, then a relay with normally closed contacts may be considered as a circuit consisting of a one-instant time delay element and an instantaneous negation element.

Further, a real contact network synthesized by the methods of Chapter 2 will not, in fact, perform the "instantaneous" function

$$x = F(u_1, u_2, \ldots, u_s),$$

but will be an automaton

$$x^{p+1} = F[u_1^p, u_2^p, \ldots, u_s^p],$$

operating at times $0, \tau, 2\tau, 3\tau, \ldots$.

Consider now a relay network such that normally open contacts of one relay close the circuit of the coil of the succeeding relay (Fig. 5.11). Then the input of the whole network is the current in the coil of the first relay, while its output is the closing of the contact x_n of the last relay. Such a network can be described by

$$x_n^{p+1} = X_n^p = x_{n-1}^p, \quad x_{n-1}^{p+1} = X_{n-1}^p = x_{n-2}^p, \ldots,$$
$$x_2^{p+1} = X_2^p = x_1^p, \quad x_1^{p+1} = X_1^p$$

or

$$x_n^{p+n} = X_1^p,$$

forming a typical loop-free automaton—an n-instants time delay line.

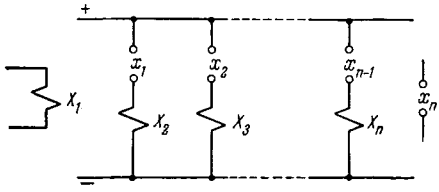

Fig. 5.11.

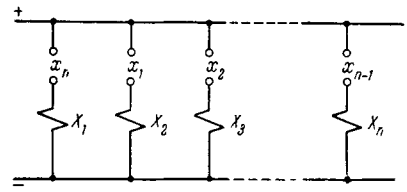

Fig. 5.12.

Let us now construct an automaton with a loop, connecting the coil of the first relay of this delay line with the contacts of the last relay; that is, we shall close the delay circuit by means of feedback (Fig. 5.12). Again, all contacts are normally open. Then, following some initial state of contacts, this automaton, operating at intervals of τ, will assume and stay in one of two possible stable states (all contacts closed or all open). If, however, the first relay was normally closed, while the others were normally open, then we would have continuous cyclic switching of contacts. The diagram of

this automaton would show all its states connected into a closed cycle. In particular, this is how the flip-flop circuit of Fig. 5.13 operates. Considered in this way, any relay switching circuit is an automaton operating at intervals τ. As we have seen, both loop-free automata (for example, the delay line shown in Fig. 5.11) and automata with loops (for example, those of Figs. 5.12 and 5.13) may be synthesized by this method. However, the only automaton of this type which makes sense is the autonomous one, since the assumption that the input also changes at intervals τ would be unrealistic.

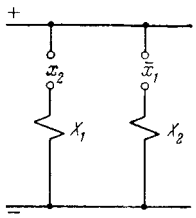

Fig. 5.13.

It should be pointed out that, in the case of loop-free autonomous automata, the diagram can have only one stable point (equilibrium) toward which the automaton tends whatever the initial state. In the case of automata with loops, however (that is, feedback circuits), the diagrams may show closed cycles, several equilibria, and so on (see Chapter 6). Even although such automata are sometimes used, they are not of great value since their cycle timing, that is, the intervals between successive discrete moments, is predetermined by the delay inherent in the relay, and so is usually very fast.

The mostly widely used automata are those in which the cycle timing is governed only by the change of the state of the input, such changes being infrequent and spaced over longer intervals of time than those required for the actuation time τ of the relay. We shall call such a cycle timing *slow*, while the cycle timing in which the time is divided into uniform intervals τ shall be known as *fast*.

Automata with slow timing governed by a change at the input may be synthesized from automata with fast timing, in which case we have a *transformation of timing* (see Chapter 10). To achieve this, one takes advantage of the fact that it is possible to synthesize fast, relay-based autonomous automata whose diagrams show several stable states. Consider, for example, the simplest relay circuit (Fig. 5.14). This circuit contains two relays, whose coils Y_1 and Y_2 are connected in subcircuits which also contain the contacts belonging to these relays. Consequently, we have a feedback circuit or an automaton with loops. In addition, the circuit also includes the contacts x_1 and x_2 of two auxiliary relays X_1 and X_2. These contacts supply the input signals.

Let the input contacts be fixed in some position. Then, if the initial state of the remaining contacts is given, the circuit operates as an autonomous automaton with fast cycle timing, conforming to the diagram of this automaton. If the diagram does not show any closed cycles but has several possible equilibria, the system can

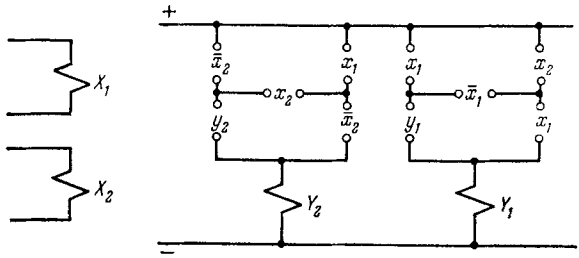

Fig. 5.14.

only tend toward one of these. Precisely which equilibrium state will be attained is determined by the initial state of the system.

Assume that the equilibrium state achieved is A. Then, some time after A has been established, let us change the state of the input contacts; after this, in accordance with previous discussion, the new state of the input contacts remains fixed. Now, with this new state of input contacts, the circuit is a new autonomous automaton with a new diagram. This new diagram may also have several possible equilibria, but the previous equilibrium A need not be one of them. If this is the case, we have a new "transient process"; that is, the automaton begins operating in fast cycle timing, tending toward a new equilibrium B, whose position is governed both by the diagram of the new automaton and by the position of the state A on this diagram.

This process is repeated whenever the state of the input contacts is changed. However, if the input contacts revert to their first state sometime later, the system need not necessarily return to equilibrium A. Indeed, in this case we recreate the initial autonomous automaton with the initial diagrams, but now the point B may be positioned on some branch of the diagram other than that on which the system was initially (prior to establishment of A). The result is that the new state of equilibrium will be other than A; it may, for example, be C, in conformance with our assumption that the diagram of our autonomous automaton shows more than one state of equilibrium.

Let us now imagine that we are recording the states of the inputs and outputs of our relay system α seconds after each change of state of the input contacts. The value of α shall be made so large that all "transient processes" occurring with fast cycle timing will have ended and a state of equilibrium attained by the time the reading is taken. However, α will not be so large that a change in the state of the input will occur during it. Then, at instants α, we shall

observe only equilibrium states; whether some state will occur will depend on the preceding equilibrium state and the state of the inputs; that is, the finite automaton now embodied by the circuit no longer operates with fast cycle timing but with a timing which is governed by the changing of the state of the input.

If the output and perhaps even the input of this automaton, are fed to a logical output converter, we have a sequential machine with slow cycle timing.

The circuit really operates with fast timing, but this is immaterial since we are interested only in the states occurring after the transient processes have terminated, and so we simply neglect these transient processes. We have thus transformed a fast-cycling automaton into a slow-cycling one. This technique of synthesizing finite automata and s-machines is, in reality, the one used for systems based on electromechanical relays, vacuum tubes, semiconductor triodes or diodes.

We come now to the following problem: can all *a priori* defined automata (or s-machines) operating with a timing governed by the change of its input states be synthesized via the above technique? The answer is yes. One method utilizing this technique is described in Section 5.4.

A related problem is that of the most economical network, that is, the network utilizing the above transformation technique to embody a given automaton and, at the same time, containing the least number of relays, contacts and states (or minimizing some other parameters affecting the cost of the circuit). A general solution for one such problem is given in Chapter 10.

The above transformation technique is based on the assumption that the diagrams of the corresponding autonomous automata show several states of equilibrium. However, this is possible only in the case of automata with loops. It follows that a fast automaton must of necessity be one with a loop, which in practice is achieved by means of feedback, that is, by connecting the relay coils to their own contacts. In this sense the resulting networks become slow automata only because of feedback. Relays connected into feedback circuits are sometimes called *intermediate relays*, as distinct from relays that are employed for the control of input contacts *(input relays)* or for picking up the signal resulting in the circuit *(output relays)*.

Comparing the aggregate method of synthesis of automata with that based on multiple stable states, we see that the aggregate method is based on a special element—the one-instant delay element—whereas the technique of multiple stable states requires no

other devices than the very same relays that are used in the logical converters, while the spacing of the operation in time is achieved by means of feedback loops and the special construction of contact networks.

It is quite obvious that all the elements of an aggregate set, in particular, its one-instant time delay elements, can themselves be based on the multiple stable states concept (see the circuit of the relay-based delay element, Fig. 5.7). However, such elements can be utilized in the aggregate systems only in conjunction with output power amplifiers; that is, they must be active.

Relay circuits are frequently designed in such a manner that the diagrams of the autonomous automata, resulting at all possible states of the input contacts, are of the specific form shown in Fig. 5.15; such diagrams show several equilibrium states (where the arrows issuing from these states lead back to the same states), while all the remaining, nonequilibirum, states are directly connected by arrows with equilibrium states. Given such a circuit, only one automaton cycle is required for attaining equilibrium; that is, equilibrium is achieved in time τ. Therefore the time α needs to be only slightly longer than τ. In practice, this means that the state of a slow automaton can be observed almost immediately after a change of the input. It is, of course, understood, that several relays may operate simultaneously during this single cycle.

If the actuating time τ were exactly the same for all the relays, then the fact that several relays are actuated simultaneously would cause no complications. However, in real systems τ is not exactly the same for all relays. For this reason, a system operating with fast cycling time may change states in a sequence different from the one that it would have followed given exactly synchronized relays. In this case, the type of resulting change of state would depend on which relay is the first actuated, that is, on factors that are random and usually not controllable. An example of this phenomenon, known as *critical race* of relays, is given in Section 5.4. This term emphasizes that the operation of the circuit is governed by the relay that operates fastest. Since one should not permit the operation of a circuit to depend on random factors, critical competition of relays must be prevented. To avoid this competition, the circuit embodying a given finite automaton or a sequential machine must satisfy

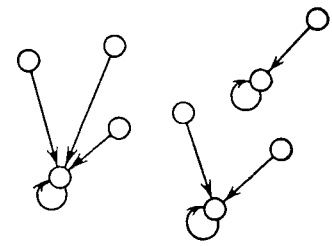

Fig. 5.15.

some additional requirements: for example, one requirement may be that the system shall be transformed from one state to another during a single "fast" cycle via the operation of a single relay. Such additional conditions necessitate more complex circuits, and thus a larger number of constituent elements (relays, contacts). Circuits satisfying these conditions are called *realizations*. There are a number of standard realizations. One of them, proposed by Huffman, will be described in the next section.

Naturally the competition problem does not apply in cases where the relays are strictly synchronized. Such a situation exists with some systems synthesized from magnetic amplifiers and tube elements, since in such systems this time τ is externally imposed on all the elements by the frequency of the alternating current feeding the system.

5.4. HUFFMAN'S METHOD AND REALIZATION

The early Huffman paper [170] on relay switching circuits still does not contain the concept of a sequential machine or a finite automaton, or their equivalents. While citing a number of ways in which the problem of synthesis of a relay switching network may be specified Huffman showed that one method is to start from a special table, which he calls the *flow table*. Assuming thereafter that the flow table is given, Huffman shows how it can be simplified (but does not show the limits of such a simplification), and then develops a general method for synthesis of relay switching circuits embodying this flow table. Huffman's circuit realizes the given table in its equilibrium states. But since his paper was not based on the concepts of a finite automaton and a sequential machine, Hoffman obviously could not specify that his method actually involves an s-machine with fast cycle timing which, in its stable states, realizes the given s-machine. The latter already has slow cycle timing, governed by changes in the states of the input.

We shall now develop Huffman's method, making use of the concepts of finite automaton, sequential machine, and cycle timing transformation. Assume we are given an s-machine, that is, two tables: the base table of the finite automaton involved, and the output converter table. We also assume that the cycle timing of the automaton is governed by change of the states of the input. We want to synthesize a relay network which, in its stable states will realize the given s-machine in accordance with the principles stated in Section 5.3. This problem is solved by Huffman's method on a simple

example, but the method is general and may be applied to other cases in exactly similar manner.

a) The Type of Automaton or Sequential Machine

Recall that any sequential machine (including any finite automaton) may be defined by the system of relations (see Section 3.4)

$$\left.\begin{aligned} \mu &= F(\varkappa, \rho), \\ \varkappa^{p+1} &= \mu^p, \\ \lambda &= \Phi(\varkappa, \rho), \end{aligned}\right\} \tag{5.2}$$

where

$$\mu, \varkappa = \{\varkappa_1, \ldots, \varkappa_k\}, \quad \rho = \{\rho_1, \ldots, \rho_r\}, \quad \lambda = \{\lambda_1, \ldots, \lambda_l\}.$$

This follows from the fact that sequential machines and automata defined by

$$\left.\begin{aligned} \mu &= F(\varkappa, \rho), \\ \varkappa^{p+1} &= \mu^p, \\ \lambda &= \Phi(\mu, \rho), \end{aligned}\right\} \tag{5.3}$$

can always be reduced to the form (5.2) by introducing the function

$$\Phi(\mu, \rho) = \Phi[F(\varkappa, \rho), \rho].$$

The converse is not true; that is, a system defined in the form (5.2), can only occasionally be reduced to (5.3) without a change in the number of states. In particular, an automaton defined by

$$\left.\begin{aligned} \mu &= F(\varkappa, \rho), \\ \varkappa^{p+1} &= \mu^p, \\ \lambda &= \mu, \end{aligned}\right\} \tag{5.4}$$

can always be represented by

$$\left.\begin{aligned} \mu &= F(\varkappa, \rho), \\ \varkappa^{p+1} &= \mu^p, \\ \lambda &= F(\varkappa, \rho), \end{aligned}\right\} \tag{5.5}$$

whereas an automaton given by

$$\left.\begin{aligned} \mu &= F(\varkappa, \rho), \\ \varkappa^{p+1} &= \mu^p, \\ \lambda &= \varkappa, \end{aligned}\right\} \tag{5.6}$$

cannot be reduced to form (5.3).

In Section 3.4, s-machines defined by (5.2) were called P -P machines, and s-machines given by (5.3) were named P - Pr machines. We shall again use this terminology here.

Since the equation system (5.2) is universal, it can be used as a canonical method for defining finite automata and sequential machines. System (5.2) finds correspondence in two tables, each with the arguments $\varkappa$ and ρ. In order to emphasize that the arguments are common to both tables, and also for the sake of conciseness, we shall treat these two tables as one. Thus the data required for the synthesis of the relay circuit will be presented as a combined table of the automaton and the output converter. The columns of this table correspond to various input states, and the rows to the various states of the automaton; in accordance with (5.2), the squares of this table shall contain two symbols: that denoting the state of the automaton, and that describing the state of the output of the s-machine. The combined table may be interpreted in the following manner. It may be assumed that the headings of the rows and columns correspond to the current states of the automaton and the input (the states at time p); then the symbols in the squares define the next state of the automaton (state at time $p + 1$) and the current state of the output of the s-machine. This is in accord with the representation of system (5.2) in the form

$$\begin{aligned} \varkappa^{p+1} &= F(\varkappa^p, \rho^p), \\ \lambda^p &= \Phi(\varkappa^p, \rho^p). \end{aligned}$$

Tables 5.1 - 5.5 show how the starting data may be given by means of combined tables.

Table 5.1 defines an automaton that may be interpreted as being either of the P - P or the P - Pr type (there is a one-to-one correspondence between λ and $\varkappa$ in all the squares of the table).

Table 5.2 gives an automaton that cannot be of the P - Pr type (different symbols λ may correspond to the same symbol $\varkappa$ within a single column; for example, we have pairs $x_3\lambda_2$ and $x_3\lambda_3$ in column ρ_1).

Table 5.1

$\varkappa$ \ ρ	ρ_1	ρ_2	ρ_3	ρ_4
$\varkappa_1$	$\varkappa_3\lambda_2$	$\varkappa_4\lambda_4$	$\varkappa_3\lambda_2$	$\varkappa_2\lambda_1$
$\varkappa_2$	$\varkappa_3\lambda_2$	$\varkappa_4\lambda_4$	$\varkappa_1\lambda_3$	$\varkappa_4\lambda_4$
$\varkappa_3$	$\varkappa_2\lambda_1$	$\varkappa_4\lambda_4$	$\varkappa_2\lambda_1$	$\varkappa_4\lambda_4$
$\varkappa_4$	$\varkappa_1\lambda_3$	$\varkappa_1\lambda_3$	$\varkappa_3\lambda_2$	$\varkappa_2\lambda_1$

Table 5.2

$\varkappa$ \ ρ	ρ_1	ρ_2	ρ_3	ρ_4
$\varkappa_1$	$\varkappa_3\lambda_2$	$\varkappa_4\lambda_2$	$\varkappa_3\lambda_2$	$\varkappa_2\lambda_2$
$\varkappa_2$	$\varkappa_3\lambda_3$	$\varkappa_4\lambda_3$	$\varkappa_1\lambda_3$	$\varkappa_4\lambda_3$
$\varkappa_3$	$\varkappa_2\lambda_4$	$\varkappa_4\lambda_4$	$\varkappa_2\lambda_4$	$\varkappa_4\lambda_4$
$\varkappa_4$	$\varkappa_1\lambda_1$	$\varkappa_1\lambda_1$	$\varkappa_3\lambda_1$	$\varkappa_2\lambda_1$

Table 5.3 defines a sequential machine that can be reduced to a P - Pr machine (there is a unique relationship between each $\varkappa$ and λ within each column; in this particular case there is also a one-to-one correspondence between $\varkappa$ and λ in each column).

Table 5.3

$\varkappa$ \ ρ	ρ_1	ρ_2	ρ_3	ρ_4
$\varkappa_1$	$\varkappa_3\lambda_1$	$\varkappa_4\lambda_2$	$\varkappa_3\lambda_2$	$\varkappa_2\lambda_4$
$\varkappa_2$	$\varkappa_3\lambda_1$	$\varkappa_4\lambda_2$	$\varkappa_1\lambda_4$	$\varkappa_4\lambda_3$
$\varkappa_3$	$\varkappa_2\lambda_3$	$\varkappa_4\lambda_2$	$\varkappa_2\lambda_1$	$\varkappa_4\lambda_3$
$\varkappa_4$	$\varkappa_1\lambda_2$	$\varkappa_1\lambda_1$	$\varkappa_3\lambda_2$	$\varkappa_2\lambda_4$

Table 5.4

$\varkappa$ \ ρ	ρ_1	ρ_2	ρ_3	ρ_4
$\varkappa_1$	$\varkappa_3\lambda_1$	$\varkappa_4\lambda_2$	$\varkappa_3\lambda_2$	$\varkappa_2\lambda_4$
$\varkappa_2$	$\varkappa_3\lambda_1$	$\varkappa_4\lambda_2$	$\varkappa_1\lambda_4$	$\varkappa_4\lambda_3$
$\varkappa_3$	$\varkappa_2\lambda_3$	$\varkappa_4\lambda_3$	$\varkappa_2\lambda_1$	$\varkappa_4\lambda_3$
$\varkappa_4$	$\varkappa_1\lambda_2$	$\varkappa_1\lambda_1$	$\varkappa_3\lambda_2$	$\varkappa_2\lambda_4$
a_i	3	3	3	2

Tables 5.4 and 5.5 define sequential machines that cannot be reduced to the P - Pr type. Table 5.4 differs from Table 5.5 in that in each of its columns there is a unique relationship between λ and $\varkappa$; that is, each λ is always paired with the same $\varkappa$ but there is no overall one-to-one correspondence between $\varkappa$ and λ.

Tables 5.1 - 5.4 have one common property: in each of them, the next state of the automaton (symbol $\varkappa$ in a square) is uniquely

Table 5.5

x \ p	p_1	p_2	p_3	p_4
x_1	$x_3\lambda_1$	$x_4\lambda_2$	$x_3\lambda_2$	$x_2\lambda_4$
x_2	$x_3\lambda_1$	$x_4\lambda_2$	$x_1\lambda_4$	$x_4\lambda_3$
x_3	$x_2\lambda_3$	$x_4\lambda_1$	$x_2\lambda_1$	$x_4\lambda_3$
x_4	$x_1\lambda_2$	$x_1\lambda_1$	$x_3\lambda_2$	$x_2\lambda_4$

defined by the current states of the input and output of the s-machine. Table 5.5 does not follow this rule.

Huffman's method may be used directly where the given circuit is not of the form of Table 5.5. In other words, this method realizes all automata (both those that can and cannot be reduced to the P - Pr type), all sequential machines that can be reduced to the P - Pr type, and some sequential machines that cannot be reduced to the P - Pr type but have the properties specified in Table 5.4.

b) Development of a Flow Table from a Given Table for an s-Machine

It is required to develop a sequential machine defined by its stable states. The machine is given, in a P - P form, by a *basic table*, which is the combined table of the automaton and the converter (Table 5.6). We assume that the next (discrete) sampling instant, that is, the cycle timing of this machine, is governed by the instant of change of its input states. We also assume that the next state of the automaton is uniquely defined by the current states of the input and output of the s-machine, that is, that there are no table columns where one λ is paired with different x. Table 5.6 satisfies this requirement. To solve this problem by Huffman's method, we must convert this table into a *flow table*. Do this as follows:

Table 5.6

x \ p	p_1	p_2	p_3	p_4
x_1	$x_2\lambda_1$	$x_2\lambda_1$	$x_1\lambda_3$	$x_1\lambda_1$
x_2	$x_1\lambda_3$	$x_3\lambda_2$	$x_3\lambda_1$	$x_1\lambda_1$
x_3	$x_1\lambda_2$	$x_2\lambda_1$	$x_1\lambda_3$	$x_2\lambda_2$
α	3	2	2	2

We add to machine Table 5.6 a bottom row where we enter α_i, the number of different output states (different λ) the machine can have at any given ith input (in our case $\alpha_1 = 3$, $\alpha_2 = \alpha_3 = \alpha_4 = 2$. We then compute $\sigma = \sum_{i=1}^{r} \alpha_i$, where r is the number of different input states. In our case $r = 4$ and $\sigma = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 = 9$. After this, we develop Table 5.7 which has the same

number of columns as the basic Table 5.6, but has σ rows. The input (top) row again contains ρ, while the extreme left column carries a sequence of new variables $x' = \{x'_1, \ldots, x'_\sigma\}$; in our case $x' = \{x'_1, \ldots, x'_9\}$. Now we fill out the table. We copy into column ρ_1 (beginning with its first row) the various λ (total number α_1) from column ρ_1 of the basic table (here these are λ_1, λ_3, and λ_2). The sequence in which these are entered into Table 5.7 is immaterial. Next to these symbols λ we copy the symbols x' denoting the given rows of Table 5.7. These squares correspond to equilibrium states; let us mark them with rectangles. We fill in the α_2 squares of column ρ_2 in the same manner, but here the entries do not start from the first row but from the first blank row (here, row x'_4). We thus enter pairs $x'_4\lambda_1$ and $x'_5\lambda_2$. In a similar manner we complete the α_i squares of each ith column.

Now, we find the symbol λ in the top row of column ρ_1 of Table 5.7 (in our case, this is λ_1); we find the square containing the same

Table 5.7

x' \ ρ	ρ_1	ρ_2	ρ_3	ρ_4	x''
x'_1	$\boxed{x'_1}\lambda_1$	$x'_5\lambda_2$	$x'_7\lambda_1$	$x'_9\lambda_1$	x''_1
x'_2	$\boxed{x'_2}\lambda_3$	$x'_4\lambda_1$	$x'_6\lambda_3$	$x'_9\lambda_1$	x''_2
x'_3	$\boxed{x'_3}\lambda_2$	$x'_4\lambda_1$	$x'_6\lambda_3$	$x'_9\lambda_1$	x''_3
x'_4	$x'_2\lambda_3$	$\boxed{x'_4}\lambda_1$	$x'_7\lambda_1$	$x'_9\lambda_1$	x''_4
x'_5	$x'_3\lambda_2$	$\boxed{x'_5}\lambda_2$	$x'_6\lambda_3$	$x'_8\lambda_2$	x''_5
x'_6	$x'_1\lambda_1$	$x'_4\lambda_1$	$\boxed{x'_6}\lambda_3$	$x'_9\lambda_1$	x''_6
x'_7	$x'_3\lambda_2$	$x'_4\lambda_1$	$\boxed{x'_7}\lambda_1$	$x'_8\lambda_2$	x''_7
x'_8	$x'_2\lambda_3$	$x'_5\lambda_2$	$x'_7\lambda_1$	$\boxed{x'_8}\lambda_2$	x''_8
x'_9	$x'_1\lambda_1$	$x'_4\lambda_1$	$x'_6\lambda_3$	$\boxed{x'_9}\lambda_1$	x''_6

symbol in the first column of Table 5.6 and note the $\varkappa$ appearing in that pair (in our case this is $\varkappa_2$).* We find the row headed by this $\varkappa$ in the basic table and, retaining the same sequence, copy the symbols λ from this row into the corresponding blank squares of the first row of Table 5.7 (here, λ_2, λ_1 and λ_1, respectively, are copied below ρ_2, ρ_3 and ρ_4). We proceed in the same manner with all rows of Table 5.7, and we enter symbols λ in all its squares. Now we match each lone λ of Table 5.7 with a symbol $\varkappa'$ in such a manner that in each column the λ's are uniquely paired with $\varkappa'$; in other words, we pair the λ's of each column, with the symbols $\varkappa'$ enclosed by the equilibrium state rectangles. Now we have a full Table 5.7, which is equivalent to Huffman's flow table, and represents the table of a "fast" sequential machine defined in the P - P form. If the ρ and λ variables are sampled only in equilibrium states, this "fast" machine will operate exactly as the starting one (Table 5.6).

c) Abbreviation (Compression) of the Flow Table

If the flow table has one or more identical rows, then this automaton may be replaced by another "fast" automaton with a smaller number of states. We shall illustrate this with Table 5.7.

We introduce the new variable $\varkappa''$, and place it into correspondence with $\varkappa'$, making sure that wherever there are one or more identical rows in Table 5.7, they correspond to the same $\varkappa''$. This matching is best done by adding a right-hand side column of $\varkappa''$ to the table of the first "fast" automaton. In our particular case, this column (see Table 5.6) has two identical $\varkappa''$ values (in the sixth and ninth rows), corresponding to identical rows. We then develop the compressed flow Table 5.8; it contains the same number of columns as Table 5.7, but only as many rows as there are different symbols $\varkappa''$, and $\varkappa''$ replaces $\varkappa'$ throughout. The compressed table thus obtained defines (in the P - P form) a "fast machine," which, when sampled only at its states of equilibrium, operates in the same way as the starting "slow" machine, but has a minimum number of states (this last statement will be clarified in Chapter 10). This, in essence, concludes the synthesis of the sequential machine. All that remains is to realize the machine by means of a relay circuit.

---

*In view of the restrictions imposed on the basic table, there is always one matching $\varkappa$ for this λ, although the same $\varkappa$ may appear in more than one row of each column.

Table 5.8

x'' \ ρ	ρ_1	ρ_2	ρ_3	ρ_4
x''_1	$\boxed{x''_1}\lambda_1$	$x''_5\lambda_2$	$x''_7\lambda_1$	$x''_6\lambda_1$
x''_2	$\boxed{x''_2}\lambda_3$	$x''_4\lambda_1$	$x''_6\lambda_3$	$x''_6\lambda_1$
x''_3	$\boxed{x''_3}\lambda_2$	$x''_4\lambda_1$	$x''_6\lambda_3$	$x''_6\lambda_1$
x''_4	$x''_2\lambda_3$	$\boxed{x''_4}\lambda_1$	$x''_7\lambda_1$	$x''_6\lambda_1$
x''_5	$x''_3\lambda_2$	$\boxed{x''_5}\lambda_2$	$x''_6\lambda_3$	$x''_8\lambda_2$
x''_6	$x''_1\lambda_1$	$x''_4\lambda_1$	$\boxed{x''_6}\lambda_3$	$\boxed{x''_6}\lambda_1$
x''_7	$x''_3\lambda_2$	$x''_4\lambda_1$	$\boxed{x''_7}\lambda_1$	$x''_8\lambda_2$
x''_8	$x''_2\lambda_3$	$x''_5\lambda_2$	$x''_7\lambda_1$	$\boxed{x''_8}\lambda_2$

d) Compilation of the Table of the Relay Network

We encode the states of the "fast" automaton in binary notation (Table 5.9), and do the same for its inputs ρ (Table 5.10) and outputs λ (Table 5.11).

We now introduce relays in the same number as there are digits in the binary coding of a given state. We use two digits to code the inputs ρ (Table 5.10) and thus have two input relays. We also have two output relays, from coding of Table 5.11. The states of the automaton (Table 5.9) are matched to intermediate relays, of which there are thus three.

We then rewrite Table 5.8 (utilizing the notations of Tables 5.9 - 5.11) to obtain Table 5.12 (the automaton table) and Table 5.13, the output converter table.

Now we have logical functions which can be realized by actual circuits: the codes employed as headings of the rows and the columns of Tables 5.12 and 5.13 give the values of logical variables

Table 5.9

$\varkappa_1''$	000
$\varkappa_2''$	001
$\varkappa_3''$	010
$\varkappa_4''$	011
$\varkappa_5''$	100
$\varkappa_6''$	101
$\varkappa_7''$	110
$\varkappa_8''$	111

(the states of the contacts of input and intermediate relays), whereas the digits in the matrices themselves are the values of the logical functions (currents in the coils of the intermediate and output relays).

To develop an expanded table of logical functions for our example, let us use x_1 and x_2 for the

Table 5.10

ρ_1	00
ρ_2	01
ρ_3	10
ρ_4	11

Table 5.11

λ_1	00
λ_2	01
λ_3	10
λ_4	11

the states of the contacts of the input relays, y_1, y_2, and y_3 for the states of the contacts of the intermediate relays, Y_1, Y_2, and Y_3 for the states of the coils of the intermediate relays, and Z_1 and Z_2 for the states of the coils of the output relays. We now obtain Table 5.14.

Table 5.12

States of contacts of the intermediate relays	States of contacts of the input relays			
	x_2x_1 0 0	x_2x_1 0 1	x_2x_1 1 0	x_2x_1 1 1
0 0 0	0 0 0	1 0 0	1 1 0	1 0 1
0 0 1	0 0 1	0 1 1	1 0 1	1 0 1
0 1 0	0 1 0	0 1 1	1 0 1	1 0 1
0 1 1	0 0 1	0 1 1	1 1 0	1 0 1
1 0 0	0 1 0	1 0 0	1 0 1	1 1 1
1 0 1	0 0 0	0 1 1	1 0 1	1 0 1
1 1 0	0 1 0	0 1 1	1 1 0	1 1 1
1 1 1	0 0 1	1 0 0	1 1 0	1 1 1
$y_3y_2y_1$	$Y_3Y_2Y_1$	$Y_3Y_2Y_1$	$Y_3Y_2Y_1$	$Y_3Y_2Y_1$

The top part of Table 5.14 shows all the possible combinations of states of contacts x_1, x_2, y_1, y_2, and y_3. For each column of this table, there is an entry in Tables 5.12 and 5.13. We then copy into each of the rows Y_1, Y_2, Y_3, Z_1, and Z_2 of the lower part of Table 5.14 the

Table 5.13

States of contacts of the intermediate relays	States of contacts of the input relays			
	x_2x_1 0 0	x_2x_1 0 1	x_2x_1 1 0	x_2x_1 1 1
0 0 0	0 0	0 1	0 0	0 0
0 0 1	1 0	0 0	1 0	0 0
0 1 0	0 1	0 0	1 0	0 0
0 1 1	1 0	0 0	0 0	0 0
1 0 0	0 1	0 1	1 0	0 1
1 0 1	0 0	0 0	1 0	0 0
1 1 0	0 1	0 0	0 0	0 1
1 1 1	1 0	0 1	0 0	0 1
$y_3y_2y_1$	Z_2Z_1	Z_2Z_1	Z_2Z_1	Z_2Z_1

numbers contained in the corresponding positions of Tables 5.12 and 5.13, and we thus complete Table 5.14.

Once we have this table of logical functions, we can use any desired method to derive the circuit corresponding to it (see Section 2.3).

e) Huffman's Realization of the Circuit

So far we have attempted to design a circuit that would substitute for a given sequential machine but we have not required a realization, that is, we did not ensure hazard-free operation. For example, we did not prevent simultaneous actuation of several relays. Thus, in the example of the preceding section there are conditions under which transition from one equilibrium to another will be accompanied by simultaneous actuation of several relays. For example, assume that the automaton is in state $Y_1 = 0$, $Y_2 = 1$, $Y_3 = 0$, with inputs at $x_1 = 0, x_2 = 0$ (first column, third row, Table 5.12). Then at a new input $x_1 = 1$, $x_2 = 1$ it will go to state $Y_1 = 1$, $Y_2 = 0$, $Y_3 = 1$. This transition is accompanied by simultaneous actuation of all three relays Y_1, Y_2 and Y_3. If some of the relays are faster than others, for example, if Y_1 and Y_3 are already set to 1 when Y_2 is still in the process of being set to 0, the circuit may assume a stable state $Y_1 = 1$, $Y_2 = 1$, $Y_3 = 1$; that is, the automaton may work in an undesirable fashion because its relays have inherent delays or operate in an improper sequence.

An automaton based on delay elements gives completely hazard-free operation. It is, however, rather difficult to design the automaton from delay elements when it is required to operate with a

Table 5.14

y_1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
y_2	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
y_3	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
x_1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
x_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Y_1	0	1	0	1	0	0	0	1	0	1	1	1	0	1	1	0	0	1	1	0	1	1	0	0	1	1	1	1	1	1	1	1
Y_2	0	0	1	0	1	0	1	0	0	1	1	1	0	1	1	0	1	0	0	1	0	0	1	1	0	0	0	0	1	0	1	1
Y_3	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Z_1	0	0	1	0	1	0	1	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1
Z_2	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0

cycle timing governed by the change of the input. In this case, we would also need a synchronous source which would respond to all change at the input, in order to provide a timing signal for the delay elements.

However, the flow table is actually the basic table of a "fast" automaton which corresponds to the initial automaton in the sense that sampling of its stable states gives a pattern describing the operation of that initial automaton (which works in a timing governed by change of the input). We can therefore design from it a "fast" automaton, corresponding to the initial one, using delay elements, and thus ensure hazard-free operation. This is easier to accomplish, since in this case the construction of the synchronous timing source is simpler.

The Huffman realization is, in reality, such a procedure. We design a "fast" automaton network from the flow table, using delay elements based on relays, and we organize a relay switching synchronous source. Thus the circuit of Huffman's realization contains an automaton based on delay elements (see Fig. 5.8) with contact networks f_i defined by the flow table.

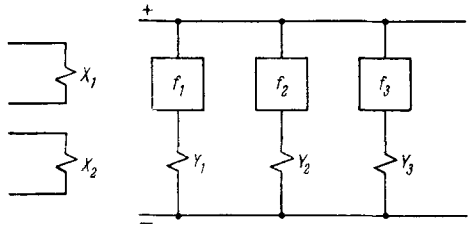

Fig. 5.16.

If the flow table contains m rows, then this part of the circuit will contain $2s_0$ relays (two relays in each delay element), where the number s_0 of delay elements is equal to the smallest integer satisfying the inequality $m \leqslant 2^{s_0}$.

The contact networks f_i are defined by the same logical functions of the flow table that define the states of the intermediate relays Y_i during the design of the switching network while ignoring hazards.

Figures 5.16 and 5.17 show block diagrams of relay switching networks corresponding to the automaton synthesized in the example of the preceding paragraph. The circuit of Fig. 5.16 ignores the possibility of hazards whereas that of Fig. 5.17 is a Huffman hazard-free realization. In these diagrams f_1, f_2, and f_3 are the contact networks (same for both tables) defined by Table 5.14. These contact networks may be synthesized from Table 5.14 by any desired method (for example, Bloch's method described in Section 2.3).

The contacts x_t and $\bar{x}_t$ of Fig. 5.17 govern the delay elements and belong to a special relay X_t at the output of the synchronous source (clock). Huffman has presented a generalized circuit for such a source. It is based on the following considerations.

The flow table is so constructed that after a change of its input and at the end of one "fast" cycle, the corresponding automaton is in equilibrium; that is, its state remains stable during subsequent

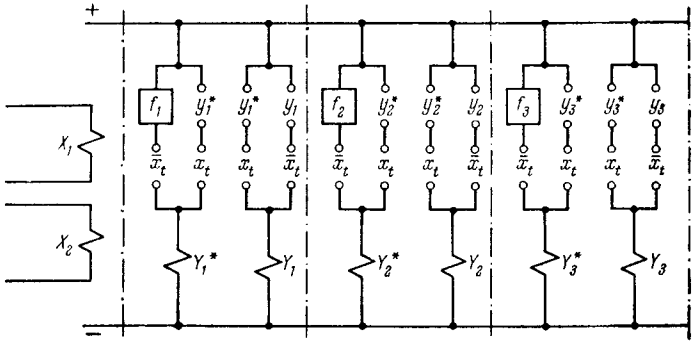

Fig. 5.17.

"fast" cycles. If delay elements are used in the circuit, the state of equilibrium means the equivalence of inputs and outputs in each delay element. Thus the beginning of a new cycle may be associated with the instant at which there is an inequality (nonequivalence) between the input and output in some delay element (that is, when for any delay element $Y_i^* \nabla Y_1 = 1$), provided that such a nonequivalence results from a change at the input and occurs in all those delay elements where it should occur. This condition, with the additional restriction that the synchronous source must return to its initial state only when equivalence between inputs and outputs has been

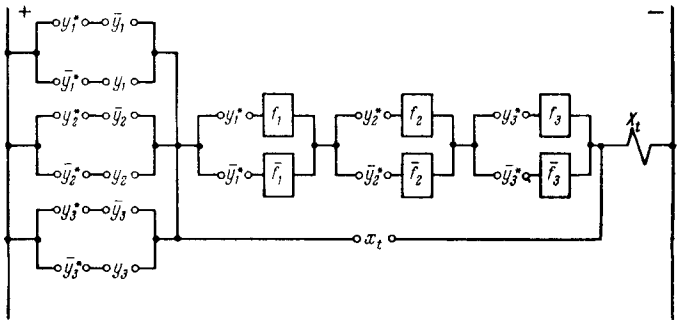

Fig. 5.18.

reestablished in the delay elements, leads to the following logic for the clock:

$$X_t = [(y_1^* \nabla y_1) \vee (y_2^* \nabla y_2) \vee \ldots \\ \ldots \vee (y_n^* \nabla y_n)] \,\&\, \{[(y_1^* \sim f_1) \,\&\, (y_2^* \sim f_2) \,\&\, (y_n^* \sim f_n)] \vee x_t\}.$$

A relay switching circuit of such a clock at $n = 3$ (to correspond with our example), is shown in Fig. 5.18. Since the clock adds one more relay to the $2s_0$ relays already employed in the delay elements, this Huffman realization is called the $(2s_0 + 1)$ realization.