. 9. BASIC ASYNCHRONOUS CIRCUIT DESIGN

. *9.1. Introduction*

As mentioned in Chapter 1XXXXXXXX, the basic sequential circuit model of CY7C361 is that of asynchronous machine. In order to be able to design reliable circuits with CY7C361 one has then to understand the asynchronous circuit theory and the methodology of designing asynchronous state machines. In the first part of this chapter we will briefly review the asynchronous circuits and machines from the point of view of using them for the 361-based design.

Since the CY7C361 chip has additionally some peculiarities of its own, like the signal differentiation, we will enhance the basic theory from the first part of this report, to include all those properties particular to CY7C361. This will be done in the second part of this chapter.

In asynchronous circuits, the input signals x_i directly affect the internal state of the circuit, and cause the change of internal state. The new state stabilizes after time τ from the moment of the input change: $\mathrm{A(t+}\tau) = \delta(A(t)), X(t))$. The numerical value of τ results from the natural delays that exist in the logic elements of the circuit, as well as in the connections.

Since as told, the internal states can be both stable and nonstable, we will distinguish and define the respective concepts:

The *stable internal states* $A(t+\tau) = A(t)$ are those that are held continuously when the state of inputs does not change. The *nonstable states* $A(t+\tau) \neq A(t)$ are those which occur in the moment of the change of input signal(s).

The entire theory of asynchronous circuits operation presented below is based on the following conditions. These conditions must be satisfied in order to be able to assume the proper behavior of an asynchronous circuit:

1. - only one input can change its state at each moment of time,
2. - the next change of inputs can occur not before them τ, required for the stabilization of the internal state of the circuit.

A standard realization of an asynchronous state machine, useful for the explanation of the phenomena occuring in such machines, is in the form of the combinational circuit with feedback loops (Figure ??a). Another model uses static flip-flops (Figure ??b). Realization of CY7C361 is another model, a more complex one, of the asynchronous state machine. We will discuss all those model, since all of them have applications to design asynchronous PLDs, either with the existing devices such as 361, 331 and 332, or new asynchronous devices that we are going to propose to Cypress this summer.

. *9.2. Creation of flow tables.*

The flow-table of an asynchronous state machine can be drawn from the natural language description or from the timing diagram of the circuit's behavior. Usually, a separate state of the machine (the so-called *complete state*) is subordinated to each combination of the input and output signals. The states, for which the input signals are identical, but the next (internal) states differ, must be distinguished when the initial flow table is created. This will be explained in the example.

Example 1

The Voltage Controlling Circuit will be used to explain several design phases in this and few following examples. The circuit is to control two pass transistor switches, y_1 and y_2, that control in turn the voltage level on the capacitor (Figure ??).

The voltage level is signalized with two sensors x_1 and x_2. The cycle of work of the switches is determined as follows:

1. 1. Voltage level below x_1 ($x_1 = x_2 = 0$) - none of the switches is ON.
2. 2. Voltage level between x_1 and x_2 ($x_1 = 1$, $x_2 = 0$). The switch is ON, which has not been ON recently, as a single switch being ON.

3. 3. Voltage level above x_2 ($x_1 = x_2 = 1$) - both switches are ON.

As one can see from the above description, there exist four different combinations of the input and output signals. The number of the complete states, however, will be greater, since when both switches are ON or both are OFF, the circuit has to remember which one of the switches has been ON recently.

The following complete states can be distinguished:

1. 1. none of the switches is ON, y_1 was on recently,
2. 2. none of the switches is ON, y_2 was on recently,
3. 3. switch y_1 is ON,
4. 4. switch y_2 is ON,
5. 5. both switches are ON, y_2 has not been ON recently,
6. 6. both switches are ON, y_2 has not worked recently.

Figure ?? presents the cycle of switches' work and the timing diagram of the machine. This diagram enumerates also the complete states of the machine. Let us point out the difference between the states 1,2 and the states 5,6 which both correspond to the same combinations of input and output signals. Although they both have the same input/output combinations, their dynamic behaviors are different. For instance, with the state of inputs $x_1 = 1$, $x_2 = 0$, the machine transits from state 1 to state 4, and with the same inputs it transits from state 2 to state 3. Let us observe, however, that states 3 and 4 are different, since they have different outputs.

After specification of the number of complete states one draws the so-called *initial flow table* which specifies for each complete state in moment t, the next state in moment $t + \tau$. Practically, at first one writes the stable states into the cells corresponding to the input signals respective to this state (Figure ??a). (Stable states are denoted with circles in the figures and they will be surrounded with parantheses in the text). Next, using the non-stable states, one marks the transitions among the stable states. For instance, if by $x_1 = x_2 = 0$ the machine was in state (1) (the switch y_1 has worked recently), then after occurrence of signal $x_1 = 1$ the machine will transit to state (4) (switch y_2 works). During the transition time τ, i.e. after changing of the state of inputs and before the change of the state of the feedback signals, the machine is in a *nonstable state* 4. This non-stable state is then written in the flow table at the intersection of the row in which stands state 1, and the column corresponding to the actual state of inputs (10), in which the stable state (4) stands (Figure ??b). All possible transitions among the states shall be written to the flow table, and the dashes shall fill up all the remaining cells of the table (particularly, the dashes can be directly written into column 01 of Figure ??a because such combination of input signals can never occur).

When the behavior of the machine is specified by a timing diagram, the creation of the flow table from it is even simpler.

Example 2

The Generator Gating Circuit will be designed. The design task is to create the circuit for gating the generator g (Figure ??a), which operator as follows:

a) When signal $b = 0$, the output signal $y = 0$.

b) When signal $b = 1$ occurs on the input, the pulses of the generator occur on the output y. These pulses must be full and not shortened. Their shape should not be affected by the moment of arising or disappearing of signal b.

c) The generator's frequency is much greater than the frequency of the gating signal: $f_g > f_b$.

Solution.

According to the above problem specification, four cases can be distinguished (Figure ??b).

1. Signal $b = 1$ occurs in the interrupt between the generator pulses,
2. Signal $b = 1$ occurs during the pulse. In such case, one more pulse should be created on the output.
3. Signal $b = 1$ disappears in the interrupt between the pulses,

4. Signal $b = 1$ disappears during the pulse, however, the full pulse should occur on the output.

The complete states of the circuit, created according to this specification, correspond in the timing diagram to the time intervals in which the input signals do not change. These states are enumerated arbitrarily, but a new number is assigned to each different combination of input and output signals.

The *complete states* (i.e. the states corresponding to the same combination of inputs and outputs), which transit under the same input signals to different next states, must be distinguished. Next, the table from Figure ??c) is created, analogously as before.

c) initial flow table

The output signals which correspond to the corresponding complete states are written at rightmost column of the table. It should be kept in mind that the timing diagram may include not all the possible transitions among the states enumerated in it. Therefore, after writing into the table all the transitions that result from the timing diagram, one cannot automatically put dashes into the remaining cells of the table, and treat such transitions as impossible to occur. Instead, one must consider whether these transitions are essentially impossible with respect to the assumptions and specifications of the problem. For instance, in our example there is no transition in the timing diagram from state 6 under input 11. Such transition is impossible from the problem's assumptions, because it would correspond to the occurrence of two pulses b with an interrupt shorter than pulse g, and it was assumed that $f_g >> f_b$. The reader is asked to use the same method to analyze also the remaining transitions, which are not shown in the timing diagram.

Example 3

Design of a "D type flip-flop" Circuit. Design an asynchronous circuit, which input/output behavior will be the same as of the synchronized, D type flip-flop. The circuit has the "c clock input" and the "D signal input". The timing diagram of this circuits is shown in Figure ??a. After enumeration of states, the flow table (Figure ??b) is created. Dashes are put into the cells of the table, corresponding to transitions that would result from simultaneous changes of two inputs (this is done to satisfy one of the basic assumptions of asynchronous circuts). The remaining not filled cells (the transitions not specified in the timing diagram) shall be now analyzed. The easy way to perform such analysis consists in drawing of the additional segments of the timing diagram, which would illustrate the not previously considered situations.

This additional diagram is shown in Figure ??c. The arrows point to the transitions. From state (3) under signal cD = 10 the machine must transit to state 2, because short pulses D (not containing back edge of signal c) should be ignored. Figure ??d presents the complete initial flow-table.

. *9.3. Minimization of flow-tables.*

The flow tables, created as in the previous section, should be next *minimized.* State Minimization of asynchronous machines is simplier than the minimization of synchronous machines, because the initial flow tables of asynchronous machines have the following special properties:

1. - the initial flow-tables are of Moore machines,
2. - in each state row of a table there exists a single stable state cell,
3. - a non-stable state can stand only in the column in which stands the same state number but stable (it can then occur only in a single column of an initial flow table).

The concept of the *pseudoequivalent states* is introduced for minimization of the asynchronous machines. The *psuedoequivalent states* are the compatible states, which have stable states in the same column, i.e.

$$A_i \ = p = \ A_j \ \Leftrightarrow \ [(A_i \ = i \ A_j) \ \ (\forall \ \ X_k)[(\delta(A_i \ , \ X_k) \ = \ A_i \ <-- > \ \delta(A_j \ , \ X_k) \ = \ A_j \ ]] \tag{1}$$

where $A_i \ = i =$

$$A_k \ \Leftrightarrow \ (\forall \ \ X_k) \ [\ (\ \delta(A_i \ , \ X_k \ ) \ \ = n = \ \ \delta(A_j \ , \ X_k \ ) \ \ (lambda(A_i) \ \ = n = \ \ lambda(A_j)) \ ] \tag{2}$$

The state minimization process which uses the concept of pseudoequivalent states, should consider as well the concept of *conditional pseudoequivalency.*

Determining the conditionally pseudoequivalent states can be executed, as for the synchronous machines, by using the triangle table. However, it is often not necessary, because the number of pseudoequivalent states is usually small.

For example, let us consider the table from Figure ??a. The pseudoequivalent states are searched among the states that have stable states in the same column.

States 1 and 6 are not pseudoequivalent, because they have inconsistent output signals. States 4,5 are pseudoequivalent under the condition of pseudoequivalence of states 7 and 9. State 10 is not pseudoequivalent with states 4 and 5, because of inconsistent outputs. States 7 and 9 are pseudoequivalent under conditions 4,5 and 3,8. States 3,8 are pseudoequivalent. All conditions are then fulfilled and the set 4, 5; 7, 9; 3,8 of pseudoequivalent states is obtained. The compatible states of each group of this set are next joined together (Figure ??b).

After joining the pseudoequivalent states, the conditional compatibility cannot exist in the table of an asynchronous machine. Two states are then considered as compatible, if their respective successor states are equal, or at least one of the successors is not specified. A merger graph is useful by searching for the maximal groups of compatible states. In this graph, the continuous lines join the states which are compatible and have consistent outputs. The interrupted line is used to link two compatible states with inconsistent outputs. By joining the states having consistent outputs (continuous lines), one obtains the Moore machine. By joining the states with inconsistent output signals (interrupted lines), the Mealy machine is created. Figure ??c presents the merger graph of the machine considered.

The maximal set of compatible states for Moore machine is:

$\phi_1 = \{1{,}7 \; ; \; 1{,}4 \; ; \; 1{,}3 \; ; \; 2{,}3 \; ; \; 6 \; ; \; 10\}$

The maximal set of compatible states for Mealy machine is:

$\phi_2 = \{1{,}4{,}7 \; ; \; 1{,}3 \; ; \; 2{,}3 \; ; \; 6{,}7 \; ; \; 2{,}10\}$

Let us observe, that in the case of an asynchronous machine the closure condition is satisfied for each set of groups of compatible states. In other words, there is no conditional compatibility for asynchronous machines. Therefore, by searching for the minimal set of compatible states, only the covering condition must be considered. Let us notice, that the group {1,3} and the state 1 from one of the groups {1,7}, and {1,4}, can be deleted from the set ϕ_1. The group {1,3} and the state 2 from the groups { 2,3} or {2,10} can be deleted from the set ϕ_2. In such a way, the following minimal sets of compatible states are obtained: { 1,4 ; 2,3 ; 6 ; 7 ; 10, or 1,7 ; 2,3 ; 4 ; 6 ; 10 for a Moore machine; and the sets 1,4,7 ; 2,3 ; 6 ; 10 or 1,4,7 ; 2,10 ; 3 ; 6 are obtained for a Mealy machine. Of course, these sets can be also obtained directly from the merger graph.

Any of these sets is selected in order to create the minimal flow table. The minimal table is created identically as for the synchronous machines; by joining the states from each group of the above set to a single state. In each column of the table, all compatible next internal states are merged together. Stable and nonstable states can be merged. After merging rows, there can be more than one stable state in one row. Figure ??d presents the minimized flow table of the Moore machine for the example considered. Figure ??e presents the respective minimized flow table of the Mealy machine. In the Moore machine the output signal corresponds to each internal state, creating of the output table is then straightforward from the flow table.

Creating of the output table for the Mealy machine is more complicated, because the output signals depend also on the states of inputs. The table is created according to the following algorithm:

Algorithm 9.3.1.

1. One creates outputs for the cells that correspond to stable states from the initial table. And so, in Figure ??e the stable state (1) in column 00 corresponds to the stable state (1) from the non-minimal table (Figure ??b) to which output 0- corresponds. One writes this output value to the respective cell of the output table. Therefore, the cell on the intersection of row 1 and column 00 obtains value 0- (Figure ??f). Analogously, state 1 in column 01 of the minimal table of Figure —refFig. 9.3.1f corresponds to state (4) of the non-minimal table, so that the respective output 00 is written to the output table. Dashes are written into the cells of the output table which correspond to the dashes of the initial flow-table (Figure ??f).
2. Output signals corresponding to the non-stable states are specified as follows:

(a) - if in the row with the considered non-stable state, at least one stable state has the same output as the stable state to which the machine transits, one writes this output to the respective cell of the flow-table. For instance, let us specify the output signal corresponding to state 2 in row 1 of the table from Figure ??e. State (2), to which the machine transits, has the output 01. In row 1 there exists three stable states, one of them (in column 11) has also output 01. Output 01 should be then subordinated to state 2.

(b) - if all stable states in a row have different values of the outputs than the final state, then a nonspecified output will be subordinated to a non-stable state. As an example, let us specify the output for state 4 in row 2 (Figure ??e). The final state (4) has output 11, the stable states in row 2 have output 01, i.e., we will subordinate output -1 to state 4, because in the first bit the outputs are different, and they are identical in the second bit. We will subordinate both nonspecified outputs for state 2 in row 3, because state (3) has output 10, and state (2) has output 01.

The resultant Mealy machine's output table is shown in Figure ??g.

The algorithm given in step 2 ensures obtaining of a proper output table, but not necessarily the minimum one. It results from the fact, that by specifying outputs of nonstable states, only those stable states in row may be considered, from which there is essentially a transition through the nonstable state, for which the outputs are specified. For instance, by specifying the output of state 4 in row 2, one can take into account in column 11 only stable state (2) (corresponding to state (2) of the non-minimized table). It is not necessary to consider the state (2) in column 10 (corresponding to state (3) of the non-minimized table), because there is no transition from this state, through the considered by us unstable state 4 (see again the non-minimized table).

The above procedure can produce more don't cares in the output table which is advantageous, because having more don't cares permits to better minimize the output functions. It is, however, more time consuming. Application of the non-minimized table is required at this step, which is inconvenient because of the need to reenumerate the states.

The complete sequence of steps to minimize a flow table of an asynchronous machine is given in the following algorithm.

Algorithm 9.3.2.

1. Find the groups of pseudoequivalent states and join them,
2. Draw the merger graph and find the minimal sets of compatible states for Moore and Mealy machines. Remember that the conditional compatibility does not exist here.
3. Create the minimal flow table and the output table of the Moore machine and the flow table of the Mealy machine. Keep in mind that a stable state in the initial table remains also stable in the minimal table.
4. Create the output table of the Mealy machine using Algorithm 9.3.1.
5. The better of the two machines is selected by inspection.

Example 4

We will minimize the flow table of the Voltage Controlling Machine from ?? (the initial table is repeated in Figure ??a):

caption*Fig. 9.3.2. Minimization of the voltage controlling machine: a) the initial flow-table, b) the merger graph, c) the table of the minimal Mealy machine, d) the step in creating the output table, e) the complete output table.*

1. Search for pseudoequivalent states. These can be the states 1 and 2 if the states {3,4} and {5,6} are pseudoequivalent under the same condition. However, the states {3,4} have inconsistent outputs and hence are not pseudoequivalent. States {1,2} and {5,6} are then not pseudoequivalent. Hence, the states {1,2} and {5,6} are not pseudoequivalent as well. Therefore, there are no pseudoequivalent states in the table.
2. The merger graph is presented in Figure ??b. In our case this graph is not indispensable, because it can be seen directly from the table that only states {1,5} and { 2,6 } can be joined to obtain a Mealy machine. The initial table is also the minimal table of a Moore machine.
3. The table of the minimal Mealy machine is presented in Figure ??c (the parentheses include enumeration of the states from the initial table).

4. Figure ??d presents the first stage of creating the output table of the machine - writing down the outputs which corresponds to the stable states. And so, to state (1) in column 00 corresponds to state (1) in the initial table, then the respective output in Figure ??d is 00. State (1) in column 11 is the state (5) from the initial table, then output 11 corresponds to it in Figure ??d. After specification of outputs which correspond to the stable states the outputs for the non-stable states should be specified (Figure ??e). State 4 stands in the first row, in which the stable states have outputs 00 and 11 (Figure ??d). State (4) to which the machine transits has output 01. State (1) from column 00 has identical output Y_1 with state (4), and state (1) from column 11 has identical output Y_2 with state (4). Therefore, one subordinates to the non-stable state 4 the same output as the state 4 has, i.e., 01. Following the Algorithm 9.3.1, the table from Figure ??e is finally created. The process of the table minimization is terminated.

Example 5

The flow table of the circuit from Example ?? will be minimized. Figure ??a repeats the flow table. As it can be observed, there are no pseudoequivalent states. The compatibility graph is presented in Figure ??b. The minimal machine is then the four-state Moore machine from Figure ??c.
caption*Fig. 9.3.3. Minimization of the flow table of the D type flip-flop: a) the initial flow table, b) the compatibility graph, c) the minimal table.*
Figure ??a presents the minimal flow table of the Moore machine for gating the generator (Example ??). Figure —refFig. 9.3.4b,c presents the flow-tables and the output table of the respective minimal Mealy machine. Checking of these solutions is left to the reader as an exercise.
caption*Fig. 9.3.4. Minimal table for the generator gating machine: a) the flow table of the Moore machine, b) the flow table of the Mealy machine, c) the output table of the Mealy machine.*

. *9.4. Assignment of flow tables. Races.*

The state assignment process of an asynchronous machine consists, as in the synchronous circuits, in subordination of sequences of values of memory elements Q_1, Q_2, ..., Q_k to the internal states of the machine. The memory elements can be realized as flip-flops or feedback loops of combinational gates. Therefore, for encoding the table with K rows (internal states) one assumes at least k signals where:

$$2^n - 1 < K < 2^k$$

Machine of K internal states must have then at least k memory elements.
The assignment of asynchronous machines is a very important process. Because of the so called *races,* a wrong assignment can produce a circuit, which would not work correctly.
9.4.1. The races.
The phenomenon of races can occur in an asynchronous machine, because of not equal delays in the feedback loops. We will explain the essence of this phenomenon on the example.
Figure ??a presents the block diagram of an asynchronous machine with two feedback loops. Let us assume that this machine, which realizes the flow table of Figure ??b, was encoded according to Figure ??c. For clarity it is assumed that all delays from the combinational unit γ have been collected in the feedback loops so that there are not any delays inside the logic network (Figure ??a).
caption*Fig. 9.4.1. The phenomena of races: a) the model of an asynchronous machine with two feedback loops, b) the flow-table of this machine, c) the encoded flow-table, d) the timing diagram, which assumes $\tau_1 = \tau_2$ (no races), e) the timing diagram, which assumes τ_1 ¡ τ_2 (critical races).*
Let us assume, that the machine is in the stable state (1), i.e.: X = 0, $Q_1 = 0$, $Q_2 = 0$. Let in some moment t_1 the input signal changes its state from 0 to 1. According to the flow table, the machine should then transit to the state 3 (encoded as 11), i.e., both feedback signals should change from 0 to 1. If delays τ_1 and τ_2 are equal, then the machine operates correctly. In time t_2 simultaneously Q_1 and Q_2 change to state zero. Such case is in real life not possible - one of the delays would always be greater. Let us assume now, that τ_2 ¿ τ_1. In such a case there will exist some time interval in which Q_1 will bf already change its value to 1 and Q_2 will bf yet not change the value. On the input of circuit γ (Figure ??e), there will arise a combination of $x = 1$, $Q_1 = 1$, $Q_2 = 0$ in moment t_3. According to the flow table, the circuit γ will create the output signals $q_1 = 1$, $q_2 = 0$, which will make the change of the signal Q_2 to 1 impossible. The value $Q_1 = 1$, $Q_2 = 0$ will be then stabilized, i.e. the machine will transit to state (4) rather than to state (3). If one assumes τ_1 ¿ τ_2 then, analyzing the table in an analogous way, one can see that the machine will transit to state (2) (the values $Q_1 = 0$, $Q_2 = 1$ will stabilize). The phenomenon described here is called the *critical hazard.*

Let us assume that the circuit is in state (3) (encoded as (11)) with x = 1, and the change of x to 0 occurs. The machine should transit to state (1) of code 00. Not depending on which of the signals Q_1 or Q_2 will first change its value to 0 the circuit will attain the state 1. Let for instance $\tau_1 < \tau_2$, i.e. at first it will be $Q_1 = 0$. The circuit will transit to state 2 (code 01), from which under x = 0 it should also transit to (1) (the circuit γ still generates signals $q_1 = 0$, $q_2 = 0$). If τ_2 ¡ τ_1 then the situation is similar and the transition (3) $\rightarrow$ 4 $\rightarrow$ (1) will occur. The above phenomenon is called the *noncritical race.*
A race can occur in the circuit only if some transition in the table requires change of at least two feedback signals. If the column to which the machine transits includes two or more stable states, then the critical race becomes possible, i.e., the race which can lead to the improper stable state. If there is only one stable state in such a column then only the noncritical race is possible in it.
It results from the above that the sufficient condition for avoiding the race is such an assignment of the machine only one feedback signal is changed in each transition among stable states. However, it should be stressed that this is not the necessary condition to avoid critical races.
Considering now a machine with the flow table of Figure ??a, it can be easily noticed that a danger of critical races exist in columns X_1, X_2, X_3. It can be checked that the encoding of the machine that would satisfy the sufficient condition of avoiding critical races is not possible. It is, however, possible to remove races from this table by introducing the so called *cyclical transitions* (Figure ??).
caption*Fig. 9.4.2. Elimination of races: a), c) the flow tables with races, b) the flow table with race removed by introducing the cyclic transitions, e), f), g) the simplified flow-graphs, d) the table in which the race was removed by adding a new state.*
Let us try to encode the states as in the table from Figure ??a. The danger of races exists now in column X_1 by transition (3) $\rightarrow$ 1 $\rightarrow$ (1) and in column X_2 by transition (1) $\rightarrow$ 3 $\rightarrow$ (3), because codes of states 3 and 1 differ in two positions. The transition (3) $\rightarrow$ 1 $\rightarrow$ (1), i.e. 11 $\rightarrow$ 00, can be practically executed in two ways: (11) $\rightarrow$ 01 $\rightarrow$ (01)((3) $\rightarrow$ 2 $\rightarrow$ (2)), which gives a bad final state (the critical race), or the transition (11) $\rightarrow$ 10 $\rightarrow$ 00 $\rightarrow$ (00) (which is, (3) $\rightarrow$ 4 $\rightarrow$ 1 $\rightarrow$ (1)), that gives the proper final state. Both transitions are marked with arrows in the table from Figure ??a. The introduction of the cyclical transition consists in forcing of this "proper" transition by writing state 4 in row 3, column X_1 (Figure ??b). This concludes how the critical race has been removed from this column. Analogously, the race from column X_2 can be deleted by introducing the cyclical transition (1) $\rightarrow$ 2 $\rightarrow$ 3 $\rightarrow$ (3) by writing state 2 in row 1 of this column.
Let us encode now the same machine according to Figure ??c. The danger of race exists now in column X_3 by transition (1) $\rightarrow$ 2 $\rightarrow$ $(2) and (3)$ $\rightarrow$ 4 $\rightarrow$ (4), because codes of states 1,2 and 3,4 differ in two positions. Transition (1) $\rightarrow$ 2 $\rightarrow$ (2), i.e., 00 $\rightarrow$ 11 can be now executed in two ways (1) $\rightarrow$ 4 $\rightarrow$ (4)((00) $\rightarrow$ 10 $\rightarrow$ (10)), or (1) $\rightarrow$ 3 $\rightarrow$ 4 $\rightarrow$ (4). Both ways lead to the wrong final state. This is an example of the critical race which cannot be removed by introducing the *cyclical transitions.* This race can be eliminated by *adding a new state.* The transition (1) $\rightarrow$ 5 $\rightarrow$ 2 $\rightarrow$ (2) is introduced instead of (1) $\rightarrow$ 2 $\rightarrow$ (2), and the transition (3) $\rightarrow$ 6 $\rightarrow$ 4 $\rightarrow$ (4) instead of (3) $\rightarrow$ 4 $\rightarrow$ (4). This way, one obtains the table from Figure ??d. This table can be now encoded according to the sufficient condition for a race-free table. Introduction of additional states causes often the necessity to increase the number of feedbacks, which usually increases the circuit's complexity. The additional states for avoiding races should be then introduced only in cases of extreme necessity.
By removing races, the attention must be also paid to the columns in which exist the noncritical races. It is possible that there exist in such columns some transitions through cells, which are unspecified in the states, through which the machine may transit during the race. These cells should be, therefore, completed (specified). In this example, the column X_4 of the table from Figure ??a. has such property. In state 3 of this column there is a dash, and if the machine transits from state (2) to state (4) then one of the possible transitions will lead through state 3. Because a nonspecified transition exist in this state then, in some particular case it may happen that the further creation of the realization logic will specify involuntarily this state as a stable state (3), which would lead to a wrong transition. This transition should be then completed by writing the nonstable state 4 into the cell on the intersection of row 3 and column X_4 (Figure ??b). A similar situation exists in row 2 of the table from Figure ??c.
9.4.2. The State Assignment.
It is required that the code taken for the assignment of an asynchronous machine removes the critical races. This is achieved by applying the sufficient condition for races elemination, by which the machine is encoded in such a way that only one memory element changes its state by each transition.
The simplest assignment method that applies this condition is the *method of hypercubes.* The simplified transition graph of the machine should be transformed to the form in which the nodes of the graph become the vertices, and the arrows become the edges of the k-dimensional hypercube (k is the number of memory elements). If the

adjacent codes of Gray code are now subordinated to the adjacent vertices of the hypercube, then the sufficient condition for eliminating races is satisfied.
Let us now consider the machine from Figure ??a. (There may be some confusion because of three unstable states. For instance in row 5. We can, however, assume that there exist some additional columns including only transient, unstable states, and that for simplification those columns have not been drawn in the table). The simplified transition graph is shown in Figure ??b. Three signals are necessary for the assignment of the table - in this case the hypercube is a standard three-dimensional cube. Transition graph in the form of the cube is shown in Figure ??c, and the encoded transition table of this machine is presented in Figure ??d.
caption*Fig. 9.4.3. State-assignment using the method of hypercubes: a) the flow table, b) the simplified state-graph, c) the simplified state-graph, presented in the form of a hypercube, d) the encoded flow-table.*
For most transition graphs it is impossible to present them in the form where all arrows are aligned with the edges of the hypercube. Some arrows being the diagonals of the hypercube can also occur, and in such case it is impossible to encode the machine in such a way that only a single feedback signal changes for each transition. In these cases, all the transitions to which correspond arrows being the diagonals of the hypercube should be inspected one by one. If a transition occurs only in a column which can include a noncritical race, then this transition can remain unmodified. If a critical race is possible in the column, this race can be removed by introducing a cyclical transition. If, in turn, the introduction of a cyclical transition were not possible - a new state would be added to the graph. Figure ??e presents the simplified transition graph of the machine from Figure ??a. The hypercube reduces here to the square. The diagonal transitions are $2 \to 4$, $1 \to 3$, and $3 \to 1$. It results from the considerations of the previous section that the cyclical transitions can be introduced for transitions $1 \to 3$ and $3 \to 1$, and that transition $2 \to 4$ does not cause a critical race. The transition graph of this machine after introducing of the cyclical transitions is shown in Figure ??f.

Example 6

The machine from Figure ??a will be assigned.
caption*Fig. 9.4.4. Example of state-assignment with hypercubes: a) the flow-table, b), c), d) the placement of the transition graph in the hypercube, e) the flow-table with races removed, f) the encoded transition table.*
The machine has 6 states so that at least 3 memory elements are required. The transition graph is drawn as a hypercube - one of the possibilities is presented in Figure ??b. The transitions on diagonals of the hypercube occur: $4 \to 1$, $6 \to 1$, $4 \to 5$, and $3 \to 6$. While $4 \to 1$, and $6 \to 1$ can produce only noncritical races, the transitions $4 \to 5$ and $3 \to 6$ can result in critical races.
The race in transition $3 \to 6$ can be eliminated (assuming this form of the graph) only by increasing the hypercube's dimension. The cyclical transition cannot be introduced, while in column X_4 of the flow table the nonstable state 6 occurs only in row 3. The new state cannot be also added without increasing the hypercube's dimension, while all the nodes of the hypercube adjacent to node 3 are already occupied. The transition graph is, therefore, drawn in a slightly different form (Figure ??c). Now the transitions on the diagonals are: $2 \to 3$, $6 \to 4$, $4 \to 5$, and $4 \to 1$ (the last one does not cause a critical race). The races on transitions $4 \to 5$ and $6 \to 4$ can be easily eliminated by introducing additional states (in both cases it is impossible to introduce the cyclical transitions). The race on transition $2 \to 3$ is more difficult to delete. It can be eliminated by introducing an additional state and a cyclical transition. The transition $(2) \to 8 \to 5 \to 3 \to (3)$ is obtained. The transition graph after elimination of races is shown in Figure ??d, the flow-table without critical races is shown in Figure ??e. Figure ??f presents the encoded flow-table of this machine.
The assignment method presented above not always leads to minimal solutions. This is because the principle of changes of single signals Q_i in transitions assumed in this method is not a necessary condition to eliminate critical races. This method does not also yield circuits of minimal complexity.
Better assignment results can be obtained by applying *partitions* of the set of internal states of the machine. The partition-based method specified below takes into account both: the races and the circuit's complexity. It makes use of the partition theory [], as well as the new concepts of internal and output partitions.
The method makes use of Theorem 9.1.
Theorem 9.1.
If for each $X_i \in X$ and each pair of states A_r, $A_j \in A$ such that

$$\delta(A_r\ ,\ X_i)\ =\ \delta(A_j\ ,\ X_i)\ =\ A_j$$

the codes of states A_r and A_j have a common part which do not stand in code of any A_p such that $\delta(A^p\ ,\ X^i)\ =\ A^n\ \neq\ A^j$ then the state assignment does not cause the critical races.
Proof.

By change of state of the machine from A_r to A_j the feedback signals Q_j corresponding to the common part of the code do not change the state. The undesired change of state from A_r because of not equal delays in the remaining feedback signals would be critical if there were a path from state A_k to the nonstable state $A_n \neq A_j$. However, according to the assumption of the theorem, no state A_k has the common part of code with A_r and A_j, which excludes the critical race.
It results from the above theorem that a proper encoding can be obtained by selecting the partitions in such a way that a partition always exists in which pairs of states (A_r , A_j) and (A_n , A_p) belong to different blocks. The expressions of the type $\overline{A_r, A_j}$ - $\overline{A_n, A_p}$, which order placing the two different pairs of states in the different blocks of a partition will be called the *elementary conditions.*

Example 7

In the table from Figure ??a (the generator gating circuit of Example ??) there is:

$$\delta(1,\ X_3)\ =\ 3,\ \ \delta(3, X_3)\ =\ 3,$$

and

$$\delta(2,\ X_3)\ =\ 2.$$

Then the codes of states 1 and 3 must have the common part, not occurring in the code of state 2. The partition $\tau_1 = \{\overline{1, 3},\ \bar{2}\}$ is thus found.
Next: $\delta(1,\ X_4)\ =\ 2,\ \delta(2,\ X_4)\ =\ 2,\ \delta(3,\ X_4)\ =\ 3$, determine the partition $\tau_2 = \{\overline{1, 2},\ \bar{3}\}$. The flow-table encoded with respect to these partitions is presented in Figure ??b. It can be easily checked that this table is free from all critical races. Several useful definitions will be now introduced.
The *final family* T_F is the set of proper partitions $T_F = \{\tau_1,\ \tau_2, ...,\ \tau_k\}$, which fulfills the conditions:

1. $\tau_1\ \cdot\ \tau_2\ \cdot\ ...\ \cdot\ \tau_k\ =\ \bar{0}$,

(a) 2. Assignment according to partitions from T_F does not cause critical races.

caption*Fig. 9.4.5. The State Assignment of the Generator Gating Machine.*
The optimal final family $T_{F_{opt}}$ is such a final family, that the assignment according to its partitions gives:

1. the simplest expressions for transition function of the machine:
   $Q_i = \delta_i(Q_1,\ Q_2, ...,\ Q_k,\ x_1,\ x_2,\ x_n)$; i = 1,2,..., k;
2. the simplest expressions for the output function:
   $y_j = \lambda_j(Q_1,\ Q_2, ...,\ Q_k,\ x_1,\ x_2,\ x_n)$; i = 1,2,..., m;

Determination of Optimal Final Family from different final families is similar to the case of synchronous machines. Attention will be now devoted to the generation of final families.
We will introduce first an additional partition that can be found from the flow-table.
The *internal partition* $\pi(X_i)$ is the partition, the blocks B_j of which include only such states A_r that transit into a single state A_j under input state X_i:

$$B_j = \{A_r\ \mid\ \delta(A_r,\ X_i)\ =\ A_j\}.$$

For instance, for the table from Figure ??a the following internal partitions exist:
$\pi(X_1) = \{\overline{123},\ \pi(X_2)\ =\ \pi(X_2)\ ,\ \pi(X_3) = \overline{13},\ \bar{2},\ \pi(X_4) = \overline{12},\ \bar{3}\}$.
The assignment with respect to the internal partitions leads to the simplification of the transition functions of the machine (this takes also place in the case of synchronous machines).
Theorem 9.2.
If each of the arbitrary two blocks of every internal partition $\pi(X_i)$ is included in two different blocks of some proper partition from set T with k elements then the set T is the final family T_F and can be selected for the state assignment.
For instance, for the machine from Figure ??a the partitions $\tau_1 = \pi(X_3) = \{\overline{13},\ \bar{2}$ and $\tau_2 = \pi(X_4) = \{\overline{12},\ \bar{3}\}$ are selected.
The state assignment with respect to these partitions is identical to one found in Example ??. Let us notice that Theorem 9.2 determines stronger condition than those from Theorem 9.1. Let us illustrate this fact on an example.

Example 8

The following partitions exist for the machine from Table in Figure ??a:
$\pi(X_2) = \{\ \overline{134},\ \bar{2}\}$, $\pi(X_3) = \{\ \overline{13},\ \overline{24}\} = \tau_1$, $\pi(X_4) = \{\ \overline{14},\ \overline{23}\} = \tau_2$.
caption*Fig. 9.4.6. Flow table and transition table for the machine to Example 9.8.*
When the partitions τ_1 and τ_2 are considered for state assignment then the condition of Theorem 9.2 would not be satisfied. These partitions do not satisfy the separation of the blocks of partition $\pi(X_2)$. Let us however notice that the satisfaction of this condition is not necessary to avoid critical races. According to Theorem 9.1, the pairs A_r, A_j and A_n, A_p must be separated; which means that $\delta(A_r,\ X_i) = A_j$ and $\delta(A_p,\ X_i) = A_n \neq A_j$. Because $\delta(3, X_2) = 1$ and $\delta(4, X_2) = 1$, the pair of states $\overline{3, 4}$ cannot be separated from the state 2. The condition of separation of blocks $\overline{134}$ - $\bar{2}$ can be then split into two conditions of separation: $\overline{13}$ - $\bar{2}$, and $\overline{14}$ - $\bar{2}$. These conditions are fulfilled by partitions τ_1, and τ_2. These two partitions are then selected for assignment and the assigned table is shown in Figure ??b.
The *reduced family* T_R is the family of proper partitions τ_i which fulfill the conditions:

1. there exist only one partition τ_i such that $\tau_i \geq \pi(X_j)$,
2. τ_i is the sum of two or more internal partition.

For instance, the internal partition $\pi(X_1) = \{\ \bar{1},\ \overline{23},\ \bar{4}\}$ specifies to T_R the partition $\tau = \{\ \overline{14},\ \overline{23}\}$, since $\tau\ ge\ \pi(X_1)$. The partitions $pi(X_2) = \{\bar{1},\ \overline{23},\ \overline{45}\}$ and $\pi(X_3) = \{\overline{12},\ \bar{3},\ \bar{4},\ \bar{5}\}$ determine to T_R the partition $\tau = \{\ \overline{123},\ \overline{45}\} = \pi(X_3)\ +\ \pi(X_2)$.
The partitions from T_R not always satisfy the separation of all blocks of internal partitions. The pairs of blocks shall be then written out after the creation of T_R which have to be separated by some additional partitions, the so-called *auxiliary conditions.* For the above examples, these will be the pairs $\bar{1}$ - $\bar{4}$ in the first case, and $\bar{1}$ - $\overline{23}$, $\bar{3}$ - $bar12$, $\bar{4}$ - $\bar{5}$ in the second case. The pairs consisting of single states (here $\bar{1}$ - $\bar{4}$ and $\bar{4}$ - $\bar{5}$ are such pairs) can be omitted by the determination of the auxiliary conditions.
The next partitions for assignment can be determined from the additional conditions so that the final family T_F is found.
It can happen that more than one final family can be created from the T_R and the auxiliary conditions. Then the optimal family $T_{F_{opt}}$ shall be selected among all the T_F families, using methods analogical to those of Chapter 5.
It can also happen that no T_F family can be created from the auxiliary conditions. If possible, the auxiliary conditions of blocks separation shall be split in such a case into the elementary conditions (as done in Example ??). If this is not possible, the cyclical transition shall be introduced. If this is still impossible, then it is necessary to increase the number of memory elements by introducing an additional partition. It cannot be determined a priori which of the methods will lead to better solutions. Plenty of examples can be shown which show the advantages of cyclical transitions, there are also examples for which the introduction of additional partition give better results. In the worst case the user has to introduce several cyclical transitions and several additional partitions.

Example 9

For the table from Figure ?? one obtains the following partitions:
$\pi(X_{00}) = \{\ \bar{1},\ \overline{34},\ (2) \leq \{\ \overline{12},\ \overline{34}\} = \tau_1$, $\pi(X_{01}) = \{\ \bar{1},\ \overline{24},\ (3)\} \leq \{\ \overline{13},\ \overline{24}\} = \tau_2$, $\pi(X_{11}) = \{\ \overline{13},\ \bar{4},\ (2)\} \leq \{\ \overline{13},\ \overline{24}\} = \tau_2$, $\pi(X_{10}) = \{\ \bar{4},\ \overline{123}\}$.
The family of partitions $T_R = \{\ \tau_1,\ \tau_2\}$ and the auxiliary condition $\overline{123}$ - $\bar{4}$ is created.
caption*Fig. 9.4.7. Flow table to Example 9.9.*
This condition cannot be satisfied by the partitions from T_R. It is then split into elementary conditions. State 1 is stable in column X_{10} so that the elementary conditions $\overline{12}$ - $\bar{4}$ and $\overline{13}$ - $\bar{4}$ exist. These conditions are fulfilled by τ_1 and τ_2 so that these partitions create the final family $T_F = \{\tau_1,\ \tau_2\}$.
Because it is the only final family obtained from the T_R set and the auxiliary conditions, then this is the $T_{F_{opt}}$ family as well. Then $T_{F_{opt}} = \{\tau_1,\ \tau_2\}$.

Example 10

For the table from Figure ??a the following partitions are obtained: $\pi(X_{00}) = \{\ \overline{14},\ \overline{235}\} = \tau_1$, $\pi(X_{01}) = \{\ \bar{1},\ \overline{25}, \overline{34}\}$, $\pi(X_{11}) = \{\ \overline{124},\ \overline{35}\} = \tau_2$, $\pi(X_{10}) = \{\ \overline{12},\ \overline{34}, \bar{5}\}$.
Then, $\pi(X_{01}) + \pi(X_{10}) = \{\ \overline{125},\ \overline{34}\}$.
Therefore, the family T_R, being as well the T_{F_opt} family, is the family $\{\ \tau_1,\ \tau_2,\ \tau_3\}$. The encoded flow table is presented in Figure ??b.
caption*Fig. 9.4.8. The Flow Table and the Assigned Table to Example 9.10.*
In the columns of this table in which the noncritical races can occur, some transitions exist that show the states

to which the machine can transit during the race. For instance, if the machine is in the state 3 (111) and state $X = 01$ occurs on input, then the machine should transit to state (4) (001). Two memory signals Q_1 and Q_2 are changed in such transition. If Q_1 changes as the first then the machine will at first transit to state 001. If there were an *unspecified transition* (a dash) in the state (3) for input state 01, then the further behavior of the machine would not be known. Therefore, to induce the transition to the proper state one has to *specify* this cell of the table, by writing state 001 in it.
Also, in row 101 of this column, the state 001 should be written in order to make the machine's behavior safe from the nonspecified result of a race that might occur in case of a quicker change of signal Q_2. After the assignment of the table, it must be always verified whether two or more feedback signals change in some transition. If yes, then one has to consider all the possible paths resulting from any race that may be produced by this change. If there exists an unspecified cell in such a path, then this cell should be specified by writing to it the state, to which the machine should transit.

Example 11

Example 12 *Let us consider now the machine from Figure ??a, assigned previously using the hypercubes method. The following internal partitions are found from it:*
$\pi(X_2) = \{ 1\bar{2},\ 3\bar{4},\ \bar{6}\ (5)\}$, $\pi(X_3) = \{ 2\bar{3}5,\ 4\bar{6},\ (1)\}$, $\pi(X_4) = \{ 1\bar{4}5,\ 3\bar{6},\ (2)\}$.
The T_R family cannot be determined from these internal partitions, because there exist no single proper partition greater than any of the internal partitions, and $\pi(X_2) + \pi(X_3) = \pi(X_2) + \pi(X_4) = \pi(X_3) + \pi(X_4) = 1$.
In such a case one looks for partitions among the proper partitions which are greater than the internal partitions.
For $\pi(X_2)$ the following partitions exist:
$\{1\bar{2}34,\ 5\bar{6}\} = \tau_1$, $\{1\bar{2}6,\ 3\bar{4}5\} = \tau_2$, $\{1\bar{2}56,\ 3\bar{4}\} = \tau_3$, $\{1\bar{2}5,\ 3\bar{4}6\} = \tau_4$, $\{1\bar{2},\ 3\bar{4}56\} = \tau_5$.
For $\pi(X_3)$:
$\{1\bar{2}35,\ 4\bar{6}\} = \tau_6$, $\{1\bar{4}6,\ 2\bar{3}5\} = \tau_7$,
For $\pi(X_4)$:
$\{1\bar{2}45,\ 3\bar{6}\} = \tau_8$. $\{2\bar{3}6,\ 1\bar{4}5\} = \tau_9$.
It results from the obtained Partition Pairs (Figure ??) that there is no possibility of simplifying function δ_i. We will check then the possibility of simplifying function λ.
$\{ 1\bar{4}6,\ 2\bar{3}5 \} \cdot \{ 1\bar{2}6,\ 3\bar{4}5 \} = \{ 1\bar{6},\ \bar{4},\ \bar{2},\ 3\bar{5} \}$. *The states 1 and 6 and 3 and 5 must be then separated. This can be done using both τ_8 and τ_9.*
*caption*Fig.9.4.9. Partition Pairs of the machine from Fig. 9.4.4.
The output partitions: $\pi_y 1 = \{ 1\bar{4}6,\ 2\bar{3}5\}$, $\pi_y 2 = \{ 1\bar{2},\ 4\bar{5},\ (36)\}$.
We have $\pi_y 1 = \tau_7$ *and* $\pi_y 2 \leq \tau_2$. *Then* $T_{F_2} = T_{F_{opt}} = \{\tau_2,\ \tau_7,\ \tau_8\}$.
The encoded transition table is presented in Figure ??. The same machine was previously encoded in Example ??. Cyclical transitions have been introduced there in order to avoid the critical races. Now, the table without cyclic transitions has been assigned, which simplifies the realization of the circuit.
We must remember, that the introduction of cyclic transitions usually increases the circuit's complexity.
*caption*Figure 9.4.10. The Encoded Transition Table of the machine from Fig. 9.4.4a.
*caption*Figure 9.4.11. The State Assignment for the Voltage Controlling Machine.
We will encode the Voltage Controlling Machine (Example ??). The minimal flow table is repeated in Figure ??.
$\pi(00) = \pi(11) = \{1\bar{2},\ 3\bar{4}\} = \tau_1$, $\pi(10) = \{1\bar{4},\ 2\bar{3}\} = \tau_2$, $T_R = \{\tau_1,\ \tau_2\}$. $T_{F_{opt}} = T_R$.
The encoded table is presented in Figure ??b. Figure ?? presents the encoded table of the D flip-flop (Example ??, Figure ??c). Checking of this solution is left to the reader.
*caption*Fig. 9.4.12. The Encoded Table of the D flip-flop.

Example 13

For the table from Figure ??a one gets the partitions:
$\pi(X_1) = \{ 1\bar{3}4,\ \bar{2}\}$, $\pi(X_2) = \{ 1\bar{2}3,\ \bar{4}\}$, $\pi(X_3) = \{ 1\bar{2},\ 3\bar{4}\} = \tau_1$.
We have $T_R = \{\tau_1\}$ *and additional conditions bar*$1\bar{3}$ - $\bar{2}$, $1\bar{4}$ - $\bar{2}$ *(from partition pi*(X_1)*)*, $1\bar{3}$ - $\bar{4}$, $2\bar{3}$ - $\bar{4}$ *(from partition pi*(X_2)*).*
*caption*Fig. 9.4.13. The Encoded Table from Fig. 9.4.2a.
None of these conditions is fulfilled by partition τ_1. It is also not possible to fit any proper partition fulfilling all remaining conditions. Therefore, two solutions exist:
- *introducing cyclic transitions,*
- *taking more memory elements than their minimal number.*

The table of this machine with the cyclical transitions introduced in columns X_1 and X_2 is shown in Figure ??b. The assignment of this table is left to our reader.

Leaving back the assumption of the minimal number of memory elements one can encode the table according to the internal partitions. Figure ?? presents the flow table encoded in such a way.
It can be easily checked that the circuit realized according to the table of Figure ?? is simpler. Therefore, in this case such assignment gives better results. However, in the general case one cannot decide which of the encoding methods gives better results, therefore both methods must be applied and their results compared to obtain the optimal results.
It should be also mentioned that although the assignment method based on partitions gives usually good results this does not happen always. An example of a machine which cannot be optimally encoded with use of this method is that of Figure ??a. This machine can be easily assigned with the method of hypercubes, while application of the partitions method will lead to the introduction of cyclical transitions and additional partitions. The reader is asked to verify these statements.

. *9.5.* The Realization of the Flow-Tables.

As it was mentioned at the beginning of this chapter, the asynchronous static machine can be realized in two variants: as the combinational network with feedbacks or as a circuit with static flip-flops. The circuits designed with the first method are usually less complex and will be considered at first.
From the encoded flow table one creates the excitation table, which describes the functions $q_i = f(Q_1, ..., Q_k, x_1, ..., x_n)$ *(see Figure ??a). In the case of the circuit with feedbacks the table is created by replacing the numbers of the states with respective codes.*
The excitation tables and the output tables (see Section 9.3) describe the combinational circuit, usually a multi-output one. This circuit can be realized with use of the method from chapter 13, or any synthesis method based on factorization or two-level logic minimization. One has to remember, however, that an active element must stand inside each feedback loop (this condition is automatically fulfilled in the case of design with NORs or NANDS). In asynchronous circuits the phenomenon of hazard is particularly dangerous (see Section 9.7 - not yet done). This is caused by the fact that each improper signal on the output can be stabilized by the feedback.

Example 14

Let us consider the realization of the machine from Figure ??a. The excitation table of this machine is presented in Figure ??b. The minimal NAND network will be obtained realizing the SOP:
$q = Q\bar{a} + ab,\ y = q.$
In such realization of function q there exists hazard of type HS1 (which means: Hazard, Static, in true minterms) in the transition marked by an arrow in Figure ??b.
*caption*Fig. 9.5.1. Hazard in an Asynchronous Circuit.
The effect of this hazard will be the creation of a zero on the output of gate 4 (Figure ??). This zero is given to the input of gate 2 (feedback) and can cause 1 on the output of this gate. This would mean not transmitting the change of signal a through this gate. In turn, this would cause supporting the zero on the circuit's output, i.e., transition of the machine to an improper state.
Let us observe that designing with 361 gives here very good new oportunities. We will also observe that methodology 2 gives sometimes better results, especially for small asynchronous machines, such as one from our example. The graph of the machine from Figure ??a is shown in Figure ??d. Its corresponding AONG requires four cells (Figure ??e). Let us , however, observe that when one calculates the excitation function on T type flip-flops (Figure ??f) from the excitation table of Figure ??b, the excitation function is simple and hazard-free. It could be reaized as in the left part of Figure ??g, but because of 361 technology constraints is transformed to the form of the right part of Figure ??g. The solution requires then two, not four cells.
*caption*Fig. 9.5.2. Realization of the Generator Gating Circuit: a), b) as a Mealy machine, c), d), e) as a Moore machine.
The question marks are written in Figure ??c above those arrows that denote changes which, as a result of hazard, can not be achieved. In order to eliminate the hazard, the implicants should be always added to the minimal realization of the asynchronous circuit (see Section 9.7). The hazard-free function q in the considered example is as follows: $q = Q\bar{a} + ab + Qb$.

Example 15

We will realize now the Generator Gating Circuit. Figure ??a presents the excitation function and the output function for the Mealy machine. After elimination of the hazard there is: $q = Qg + \bar{g}b + Qb,\ y = Qg$
The corresponding circuit with NANDs is presented in Figure ??b. Figure ??c-e presents the realization of the excitation function of this machine realized as a Moore machine. The SOPs of q_1 *and* q_2 *functions are hazard-*

free. Let us observe that it was especially resigned from the minimal SOP because the above functions are more convenient for factorization. As we see in this case the Mealy realization is slightly less expensive.

Example 16

The Generator Gating example on 361. *The T type flip-flop excitation function for the machine from Figure ??a,b is shown in Figure ??g. As shown in Figure ??h, it requires two cells and one AND-type Mealy output.*
Another option to realize this circuit with 361 is to use a Moore machine. The T type flip-flop excitation functions calculated from transition functions from Figure ??d,e, are shown in Figure ??i,j. The output function, found from Figure ??c, is $y = Q_1$. One needs then four cells to realize the Moore machine, versus two cells required to realize the Mealy machine, which is also faster.

Example 17

We realize the Voltage Controlling Circuit. The excitation functions of this circuit are presented in Figure ??a-c. These functions are minimized as follows:
$q_1 = Q_1\,\bar{x_1} + Q_1\,x_2 + Q_1\,\bar{Q_2} + \bar{Q_2}\,x_1\,\bar{x_2}$, $q_2 = Q_1\,\bar{x_1} + Q_1\,x_2 + Q_1\,Q_2 + Q_2\,x_1\,\bar{x_2}$, $y_1 = x_2 + Q_1\,x_1\,\bar{x_2}$, $y_2 = x_2 + \bar{Q_2}\,x_1\,\bar{x_2}$.
By realization of excitation functions q_1 and q_2 the hazard was eliminated. The elimination of hazard in the output functions is not necessary, while the output signals are given (through power amplifiers) to the motors of high (according to the speed of logic element operation) inertia. Groups $Q_2\,x_1\bar{x_2}$ and $\bar{Q_2}\,x_1\,\bar{x_2}$ are common for the excitation and output functions.
The expressions for excitation functions can be further factorized: $q_1 = Q_1\,(\bar{x_1} + x_2 + \bar{Q_2}) + \bar{Q_2}\,x_1\,\bar{x_2}$, $q_2 = Q_1\,(\bar{x_1} + x_2 + Q_2) + Q_2\,x_1\,\bar{x_2}$.
*caption*Fig. 9.5.3. The excitation functions and output functions for the Voltage Controlling Machine.
To these expressions corresponds the circuit with NAND gates from Figure ??a. This circuit can be further simplified by replacing $\bar{Q_2}$ with a bunch of wires coming to the gate realizing q_2 (Figure ??b,c).
*caption*Fig. 9.5.4. The schematics for the Voltage Controlling Machine.
In the case when the asynchronous static circuit is realized with static rs flip-flops (latches), the excitations of these latches are calculated with use of the excitation array of Figure ??.
*caption*Fig. 9.5.5. Array of static *rs* flip-flops.
This array is identified as the array for synchronous RS flip-flop. The flow-table shows what signals must be given to inputs r and s to obtain the expected change of its state.
After encoding and thickening the changing values of Q the flow table specifies the required changes of the flip-flop states. From this table one calculates the excitation functions of these flip-flops, as follows:

(a) - *function s has ones for bold ones, and zeros for zeros of $Q^{t+\tau}$.*

(b) - *function r has ones for bold zeros ones, and zeros for ones of $Q^{t+\tau}$.*

*caption*Fig. 9.5.6. Diagram of the Generator Getting Machine with *rs* flip-flops.
The method of finding the excitation function maps is similar as for the synchronous circuits. Finding the realization of the static circuit with flip-flop is usually less laborious than the synthesis of the circuit with feedbacks. The circuit obtained is however usually not minimal (Figure ??b).

Example 18

Realization with 361. *From Figure ??b,c one obtains the T type excitation functions from Figure ??a,b. They are realized as shown in Figure ??. One can observe the usage of CDEC-cells. The Mealy outputs are realized by delaying outputs by one clock pulse using D type flip-flop realized from T type flip-flop. For instance, implicant $Q_1\,x_1$ being the input to the D flip-flop is realized as follows:*
$T = \bar{D}\;Q + \bar{Q}\;D =$
$T = \bar{Q_1\;x_1}\;Q + \bar{Q}\;Q_1\;x_1\;Q$. *This is realized with START extension and Mealy outputs, as shown in Figure ??. Similarly the realization of the second output is realized (see Figure ??) The CDEC-cells have been used again.*

Example 19

The diagram of D flip-flop (encoded flow-table of Figure ??a) is presented in Figure ??. Checking of this solution is left to the reader.
*caaption*Fig. 9.5.9. a) The diagram of the type D flip-flop, b),c) The excitation functions of T type flip-flops. d) The realization of logic with CY7C361 device.

. *9.6. Asynchronous dynamical machines.*

The ST cell of the CY7C361 producing short pulses, can be used to simplify the realization of asynchronous machines. We will call such machines - the dynamical machines, to distinguish it from the previously described static (level mode) machines. We can call them also the pulse machines.
*caption*Fig. 9.6.1. The creation of the dynamical flow table of the Generator Gating Circuit: a) the static table, b) the dynamic table, c) minimization, d) the assignment.
It is convenient to use a dynamic flow table while describing the dynamic machine. It specifies for each internal state and for each change of input states $X^{t-\tau}$ */* X^t *the next internal state:* $A(t+\tau) = \delta[A(t),\ X(t-\tau\ /\ X(t)]$. *Let us draw the dynamic flow table for the Generator Gating Machine, which corresponds to the static table of Figure ??a - this table is repeated in Figure ??a. By creating the dynamic flow table one must remember that only one input signal can change at any moment in an asynchronous machine. Therefore, for instance, there is a transition from state (1) to state (3) with the inputs changing from 01 to 11, and from state (1) to state (2) with the inputs changing from 00 to 10. However, during the changes of the input states from 00 to 01 and from 01 to 00, the circuit remains in state 1 (Figure ??b).*
The dynamic flow table is similar to the flow table of a synchronous machine and is minimized in a similar way. As we see, the states 1 and 2 in the Table from Figure ??b are compatible and can be joined, which leads to the table of Figure ??c. After the state assignment, this table takes the form presented in Figure ??d.
Figure ??a presents the operation of the differentiating element. It is very close to START:
$d(\bar{g}) = START(g), \quad d(\bar{g}) = START(\bar{g})$.
The dynamic circuit can be realized using many methods but we will use here a combination of ST, TO and TE cells of CY7C361. For these elements we shall find the excitation function T_1 *of T type flip-flop (Figure ??b). These excitation function is as follows:*
$T_1 \ = \ b\ d(\bar{g}) \ + \ Q\ d(g)$,
The realization using START cells operating as differentiating elements is shown in Figure ??c.
*caption*Fig. 9.6.2. The Generator Gating Circuit: a) Explanation of the concept of differentiating element. b) the dynamic excitation table, c) the realization using START cells and a TOGGLE cell.

. *9.7. Hazards and Hazard Free Logic Synthesis.*

This section will be added.

. 9.7. Literature to chapter 9.

General problems of asynchronous automation: [?, ?, ?]
Creating of initial flow tables: [?, ?, ?, ?].
Minimization: [?, ?, ?, ?, ?, ?].
State assignment: [?, ?, ?, ?, ?, ?, ?, ?].
Elementary machines: [?].
Description of machines and automata: [?].
Transition from timing diagram to flow table: [?].
Different types of asynchronous circuits: [?].
Counters: [?].
Minimization of number of columns in flow table: [?].
Hazards and races: [?, ?, ?, ?, ?, ?, ?, ?, ?].
Sources of good design problems: [?, ?, ?].

. 9.8. Review Questions.

9.8.1. What is asynchronous state machine?
2. What are the conditions of proper behavior of asynchronous machine?
3. What are the stable and unstable states?
4. What are the pseudoequivalent states?
5. Describe the process of state minimization of asynchronous state machines.
6. Describe creating of the output table of the minimal Mealy machine.