Chapter 4 Algorithmic State Machines and Finite State Machines

In this Chapter, we will introduce Algorithmic state machines and consider their use for description of the behavior of control units. Next, we will use algorithmic state machines to design Finite State Machines (FSM) with hardly any constraints on the number of inputs, outputs and states.

4.1 Flowcharts and Algorithmic state machines

4.1.1 Example of ASM. An Algorithmic state machine (ASM) is the directed connected graph containing an initial vertex (Begin), a final vertex (End) and a finite set of operator and conditional vertices (Fig. 1). The final, operator and conditional vertices have only one input, the initial vertex has no input. Initial and operator vertices have only one output, a conditional vertex has two outputs marked by "1" and "0". A final vertex has no outputs.

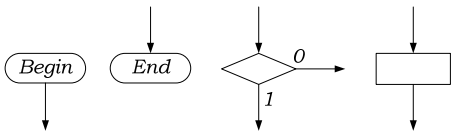

Figure 1. Vertices of Algorithmic state machine

As the first example, let us consider a very simple Traffic Light Controller (TLC) presented in the flowchart in Fig. 2. This controller is at the intersection of a main road and a secondary road. Immediately after vertex *Begin* we have a waiting vertex (one of the outputs of this vertex is connected to its input) with a logical condition *Start*. It means that the controller begins to work only when signal *Start = 1*. At this time, cars can move along the main road for two minutes. For that, the traffic light at the main road is green, the traffic light at the secondary road is red and the special timer that counts seconds is set to zero *(main_grn := 1; sec_red := 1; t := 0).*

Although our TLC is very simple it is also a little smart – it can recognize an ambulance on the road. When an ambulance is on the road the signal *amb* is equal to one (*amb = 1*), when there is no ambulance on the road this signal is equal to zero (*amb = 0*). First we will discuss the case when there are no ambulances on the road.

Thus, when *amb = 0* and *t = 120 sec* TLC transits into some intermediate state to allow cars to finish driving along the main road: *main_yel := 1; sec_red := 1; t := 0.* TLC is in this state only for three seconds (*t = 3 sec*), after which cars can move along the secondary road for 30 seconds: *main_red := 1; sec_grn := 1; t := 0.*

Thirty seconds later, if there are no ambulances on the road (*amb = 0; t = 30 sec*), there is one more intermediate state. Now cars must finish driving along the secondary road: *main_red := 1; sec_yel := 1; t := 0.* After three seconds, if, once again, there are no ambulances on the road, the process reaches vertex End, or, that is the same, it returns to the beginning vertex *Begin*.

When there is ambulance on the road (*amb = 1*) outputs of conditional vertices with logical condition *amb*, marked by “1” bring us to the intermediate state to let cars to finish their driving: *main_yel := 1; sec_yel := 1; t := 0.* One more logical condition *dmain* tells us where the ambulance is – whether it is on the main road or on the secondary one. If it is on the main road (*dmain = 1*), after three seconds the traffic

light will be green on the main road, otherwise (*dmain = 0*) the traffic light will be green on the secondary road.

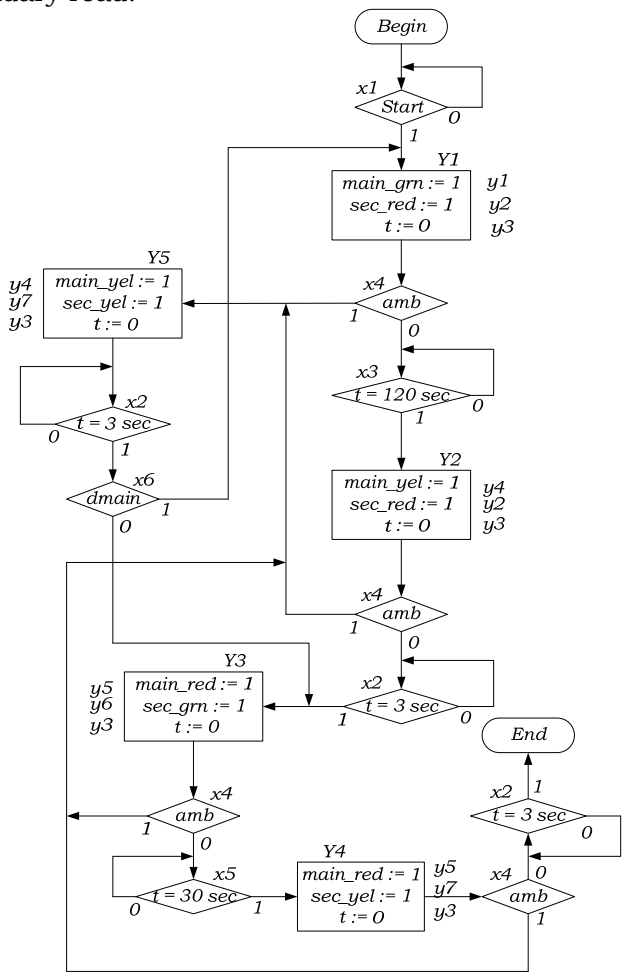

Figure 2. A simple Traffic Light Controller

In the flowchart, a logical condition is written in each conditional vertex. It is possible to write the same logical condition in different conditional vertices. A microinstruction (an operator), containing one, two, three or more microoperations, is written in each operator vertex of the flowchart. It is possible to write the same operator in different operator vertices.

If we replace logical conditions by $x_1, x_2, \ldots, x_L$, microoperations by $y_1, y_2, \ldots, y_N$ and operators by $Y_1, Y_2, \ldots, Y_T$ we will get Algorithmic State Machine (ASM). ASM for the flowchart in Fig. 2 is shown in Fig. 3.

ASM vertices are connected in such a way that:

1. Inputs and outputs of the vertices are connected by arcs directed from an output to an input, each output is connected with only one input;
2. Each input is connected with at least one output;
3. Each vertex is located on at least one of the paths from vertex “Begin” to vertex “End”. Hereinafter we will not consider ASMs with subgraphs, containing an infinite cycle. An example of such a subgraph with an infinite

loop between vertices with Y_1 and Y_3 is shown in Fig. 4. The dots in this ASM between vertex *"Begin"* and the conditional vertex with x_1 and between this vertex and vertex *"End"* mean that ASM has other vertices on the path from vertex *"Begin"* to vertex *"End"*. The vertices in the loop are not on the path from *"Begin"* to *"End"*.

4. One of the outputs of a conditional vertex can be connected with its input. We will call such conditional vertices the "*waiting vertices*", since they simulate the waiting process in the system behavior description.

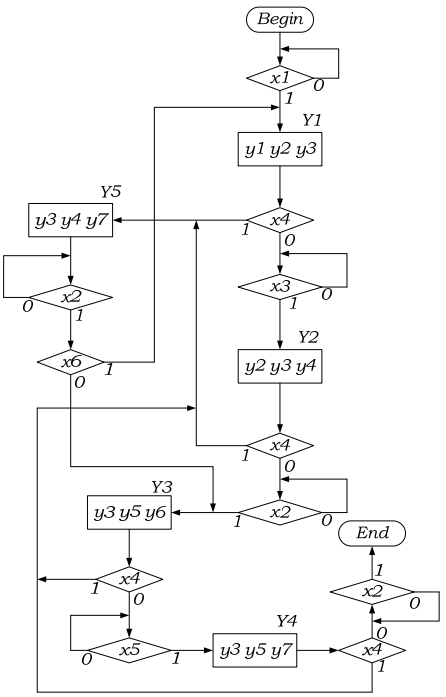

Figure 3. ASM for the flowchart in Fig. 2

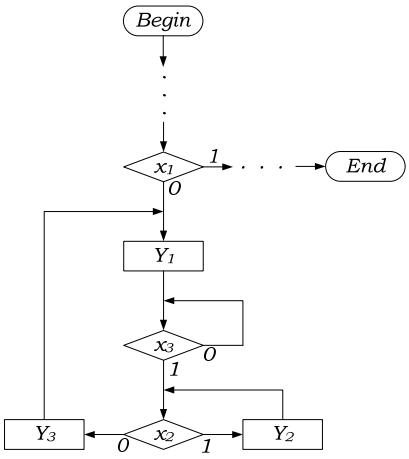

Figure 4. Subgraph with an infinite loop

One more example of ASM G_1 with logical conditions $X = \{x_1, \ldots, x_7\}$ and microoperations $Y = \{y_1, \ldots, y_{10}\}$ is shown in Fig. 5. This ASM has eight operators $Y_1, \ldots, Y_8$, they are written near operator vertices.

4.1.2 Transition functions. Let us discuss the paths between the vertex "Begin", the vertex "End" and operator vertices passing only through conditional vertices. We will write such paths as follows:

$$Y_i \tilde{x}_{i1} \ldots \tilde{x}_{iR} Y_j \qquad (1)$$

In such a path, $\tilde{x}_{ir}$ is equal to x_{ir} if the path proceeds from the conditional vertex with x_{ir} via output *'1'*, and $\tilde{x}_{ir}$ is equal to x_{ir}' if the path proceeds from the conditional vertex with x_{ir} via output *'0'*. For example, we have the following paths from Y_b (vertex *Begin*) in ASM G_1:

$Y_b\ x'_1\ Y_2;$
$Y_b\ x_1 x_2 x'_3\ Y_6;$
$Y_b\ x_1 x'_2\ Y_1;$
$Y_b\ x_1 x_2 x_3\ Y_5.$

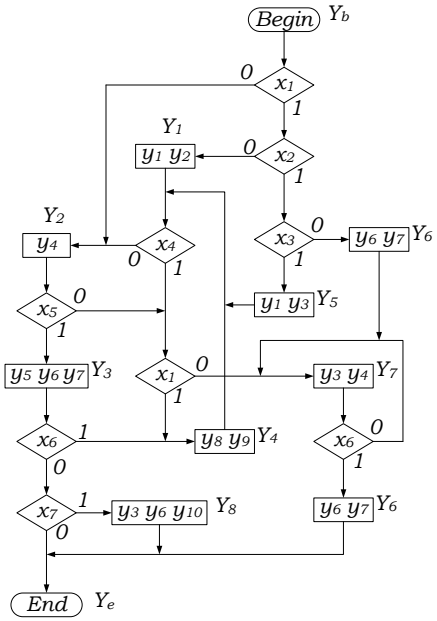

Figure 5. ASM G_1

Let us match a product of variables in the path (1) from operator vertex Y_i to operator vertex Y_j

$$\alpha_{ij} = \tilde{x}_{i1} \ldots \tilde{x}_{iR}$$

with this path from Y_i to Y_j. For example, for ASM G_1 in Fig. 5

$$a_{17} = x_4\, x'_1; \quad a_{12} = x'_4; \quad a_{14} = x_4\, x_1.$$

If there exist H paths between Y_i and Y_j through the conditional vertices, then

$$a_{ij} = a^1_{ij} + a^2_{ij} + \ldots + a^H_{ij}$$

where a^h_{ij} ($h = 1, \ldots, H$) is the product for the h-th path. Let us call a_{ij} a transition function from operator (microinstruction) Y_i to operator (microinstruction) Y_j.

Note that for the path Y_6Y_7 (operator Y_7 follows operator Y_6 immediately without conditional vertices) $a_{67} = 1$, as the product of an empty set of variables is equal to one.

4.1.3 Value of ASM at the sequence of vectors. Denote all possible L-component vectors of the logical conditions $x_1, \ldots, x_L$ by $\Delta_1, \ldots, \Delta_2{}^L$ and define the execution of an ASM on any given sequence of vectors $\Delta_1, \ldots, \Delta_{mq}$ beginning from the initial operator Y_b. We will demonstrate this procedure by means of ASM G_1 in Fig. 5 and the sequence (2) containing eight vectors $\Delta_1, \ldots, \Delta_8$:

		x_1	x_2	x_3	x_4	x_5	x_6	x_7
Δ_1	=	1	0	1	0	1	1	1
Δ_2	=	0	1	1	0	1	0	0
Δ_3	=	1	0	1	0	0	1	0
Δ_4	=	0	1	0	0	0	0	1
Δ_5	=	1	1	0	1	1	1	0
Δ_6	=	1	1	0	0	1	0	1
Δ_7	=	0	1	1	1	0	0	0
Δ_8	=	0	1	0	1	0	0	1

(2)

ASM G_1 in Fig. 5 contains logical variables $x_1, \ldots, x_7$ and operators $Y_b, Y_1, \ldots, Y_8, Ye$. Now let us find the sequence of operators which would be implemented, if we consecutively, beginning from Y_b, give variables the values from these vectors. We suppose that the values of logical conditions can be changed only during an execution of operators.

Step 1. Write the initial operator

$$Y_b.$$

Step 2. Let logical variables $x_1, \ldots, x_7$ take their values from vector Δ_1. From the set of the transition functions $a_{b1}, \ldots, a_{b8}, a_{be}$ we choose such a function a_{bt} that $a_{bt}(\Delta_1) = 1$. In our example for the operator Y_b, the following transition functions are not identically equal to zero:

$$a_{b5} = x_1\, x_2\, x_3; \qquad a_{b6} = x_1\, x_2\, x'_3; \qquad a_{b1} = x_1\, x'_2; \qquad a_{b2} = x'_1.$$

We will call such functions *non-trivial transition functions* to distinguish them from the trivial functions, which are identically equal to zero. Function a_{ij} is *trivial* if there is no path from operator Y_i to operator Y_j. In the example at this step, we choose the function a_{b1}, since only a_{b1} is equal to one on the first vector Δ_1:

$$a_{b1}\,(\Delta_1) = 1.$$

Write Y_1 to the right of Y_b:

$$Y_bY_1.$$

Step 3. Let $x_1,\ldots,x_7$ take their values from vector Δ_2. From the set of the transition functions $a_{11},\ldots, a_{18}, a_{1e}$ we choose non-trivial functions

$$a_{14} = x_4 x_1; \qquad a_{17} = x_4 x'_1; \qquad a_{12} = x'_4$$

and among them – the only function $a_{12}(\Delta_2) = 1$. Write Y_2 to the right of Y_bY_1:

$$Y_bY_1Y_2.$$

The computational process for the given sequence of vectors may reach its end in two cases:

1. The final vertex "End" is reached. In this case, the last operator is Y_e. The number of operators in the operator row (without Y_b and Y_e) is less or equal (if we reached the final vertex with the last vector) to the number of vectors;
2. The vectors are exhausted but we have not yet reached the final vertex. In this case, the number of operators in the operator row is equal to the number of vectors.

In our example, we reached the final vertex "End" at the seventh vector

$$\Delta_7 = 0\ 1\ 1\ 1\ 0\ 0\ 0$$

and we get the row

$$Y_b\ Y_1\ Y_2\ Y_4\ Y_2\ Y_3\ Y_8\ Y_e. \tag{3}$$

The operator row thus obtained is *the value of the ASM G_1 for the given sequence of vectors (2).*

4.2 Synthesis of Mealy FSM

We will use Algorithmic state machines to describe the behavior of digital systems, mainly of their control units. But if we must construct a logic circuit of the control unit we should use a Finite state machine (FSM). We will consider methods of synthesis of FSM Mealy, Moore and their combined model implementing a given ASM, with hardly any constrains on the number of inputs, outputs and states.

4.2.1 Construction of a marked ASM. As an example we will use ASM G_1 in Fig. 6. A Mealy FSM implementing given ASM may be constructed in two stages:

Stage1. Construction of a marked ASM;

Stage 2. Construction of a state diagram (state graph).

At the first stage, the inputs of vertices following operator vertices are marked by symbols $a_1, a_2, \ldots, a_M$ as follows:

1. Symbol a_1 marks the input of the vertex following the initial vertex *"Begin"* and the input of the final vertex *"End"*;
2. Symbols $a_2, \ldots, a_M$ mark the inputs of all vertices following operator vertices;
3. Vertex inputs are marked only once;
4. Inputs of different vertices, except the final one, are marked by different symbols.

Marked ASM G_1 in Fig. 6 is a result of the first step. Symbols $a_1, \ldots, a_6$ are used to mark this ASM. Note, that we mark the inputs not only of conditional vertices but

of operator vertices as well (see mark a_3 at the input of the vertex with operator Y_7). It is important that each marked vertex follows an operator vertex.

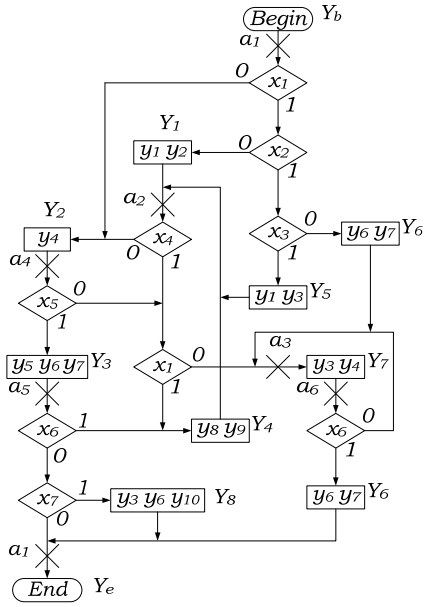

Figure 6. ASM G_1 marked for the Mealy FSM synthesis

4.2.2 Transition Paths. At the second stage, we will consider the following paths in the marked ASM:

$$a_m \tilde{x}_{m1} \ldots \tilde{x}_{mR_m} Y_g a_s \qquad (P1)$$

$$a_m \tilde{x}_{m1} \ldots \tilde{x}_{mR_m} a_1 \qquad (P2)$$

We call these paths *transition paths*. Thus, the path *P1* proceeds from a_m to a_s ($a_m = a_s$ is also allowed) and contains only one operator vertex at the end of this path. The path *P2* proceeds from a_m only to a_1 without operator vertex. Here, $\tilde{x}_{mr} = x_{mr}$, if on the transition path we leave the conditional vertex with x_{mr} via output '1' and $\tilde{x}_{mr} = x'_{mr}$ if we leave it via output '0'. If $R_m = 0$ on the path *P1*, two operator vertices follow one after another and this path turns into

$$a_m Y_g a_s .$$

There are sixteen transition paths in the marked ASM G_2 in Fig. 6:

$a_1\ x_1x_2x_3\ Y_5\ a_2$	$a_2\ x_4x_1\ Y_4\ a_2$	$a_4\ x_5\ Y_3\ a_5$	$a_5\ x'_6x_7\ Y_8\ a_1$
$a_1\ x_1x_2x'_3\ Y_6\ a_3$	$a_2\ x_4x'_1\ Y_7\ a_6$	$a_4\ x'_5x_1\ Y_4\ a_2$	$a_5\ x'_6x'_7\ a_1$
$a_1\ x_1x'_2\ Y_1\ a_2$	$a_2\ x'_4\ Y_2\ a_4$	$a_4\ x'_5x'_1\ Y_7\ a_6$	$a_6\ x_6\ Y_6\ a_1$
$a_1\ x'_1\ Y_2\ a_4$	$a_3\ Y_7\ a_6$	$a_5\ x_6\ Y_4\ a_2$	$a_6\ x'_6\ Y_7\ a_6$

Note, that the path $a_2\ x_4x'_1\ a_3$ doesn't correspond to the transition path *P1* (the operator vertex is absent on the path) and to transition path *P2* (it isn't a path to a_1). Thus, it isn't a transition path and we should go on to get the path $a_2\ x_4x'_1\ Y_7\ a_6$. For the same reason, paths $a_4\ x'_5x'_1\ a_3$ and $a_6\ x'_6\ a_3$ are not the transition paths either.

4.2.3 Graph of FSM. Next we construct a graph (state diagram) of FSM Mealy with states (marks) $a_1, \ldots, a_M$, obtained at the first stage. We have six such states $a_1, \ldots, a_6$ in our example. Thus, the FSM graph contains as many states as the number of marks we get at the previous stage. Now we should define transitions between these states.

FSM has a transition from state a_m to state a_s with input $X(a_m, a_s)$ and output Y_g (see the upper subgraph in Fig. 7) if, in ASM, there is transition path $P1$

$$a_m \tilde{x}_{m1} \ldots \tilde{x}_{mR_m} Y_g a_s .$$

Here $X(a_m, a_s)$ is the product of logical conditions written in this path:

$$X(a_m, a_s) = \tilde{x}_{m1} \ldots \tilde{x}_{mR_m} .$$

In exactly the same way, for the path $a_m Y_g a_s$ we have a transition from state a_m to state a_s with input $X(a_m, a_s) = 1$ and output Y_g, as the product of an empty set of variables is equal to zero. If, for a certain r $(r = 1, \ldots, R_m)$, symbol x_{mr} (or x'_{mr}) occurs several times on the transition path, all symbols x_{mr} (x'_{mr}) but one are deleted; if for a certain r $(r = 1, \ldots, R_m)$, both symbols x_{mr} and x'_{mr} occur on the transition path, this path is removed. In such a case $X(a_m, a_s) = 0$.

For the second transition path $P2$, FSM transits from state a_m to the initial state a_1 with input $X(a_m, a_1)$ and output Y_0 (see the lower subgraph in Fig. 7). Y_0 is the operator containing an empty set of microoperations.

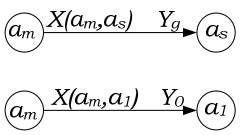

Figure 7. Subgraphs for transition paths *P1* and *P2*

As a result, we obtain a Mealy FSM with as many states as the number of marks we used to mark the ASM in Fig. 6. The state diagram of the Mealy FSM is shown in Fig. 8.

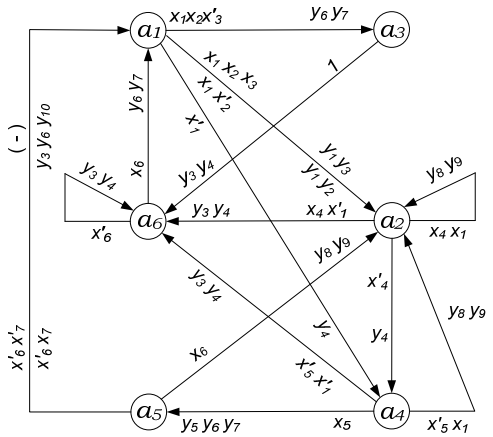

Figure 8. The state diagram of the Mealy FSM

4.2.4 How not to loose transition paths. Sometimes, if ASM contains many conditional vertices, it is difficult not to loose one or several transition paths. Here

we give a very simple algorithm to resolve this problem. This algorithm has only two steps.

1. Find the first transition path leaving each conditional vertex through output '*1*'. For subgraph of ASM in Fig. 9 we will get the following first path from state a_2:

$$a_2\ x_1\ x_2\ x_5\ Y_6\ a_3.$$

2. Invert the *last non-inverted* variable in the previous path, return to ASM and continue the path (if it is possible) leaving each conditional vertex through output '*1*'. To construct the second path, we should invert variable x_5. We cannot continue because we reached an operator vertex:

$$a_2\ x_1\ x_2\ x'_5\ Y_2\ a_3.$$

We should construct paths in the same manner until all variables in a transition path will be inverted. For our example, we will get the following paths:

$a_2\ x_1\ x'_2\ x_5\ x_6\ Y_3\ a_4;$
$a_2\ x_1\ x'_2\ x_5\ x'_6\ x_7\ x_4\ Y_5\ a_5;$
$a_2\ x_1\ x'_2\ x_5\ x'_6\ x_7\ x'_4\ Y_7\ a_4;$
$a_2\ x_1\ x'_2\ x_5\ x'_6\ x'_7\ Y_5\ a_5;$

$a_2\ x_1\ x'_2\ x'_5\ Y_2\ a_3;$
$a_2\ x'_1\ x_3\ x_7\ x_4\ Y_5\ a_5;$
$a_2\ x'_1\ x_3\ x_7\ x'_4\ Y_7\ a_4;$
$a_2\ x'_1\ x_3\ x'_7\ Y_5\ a_5;$

$a_2\ x'_1\ x'_3\ x_6\ Y_3\ a_4;$
$a_2\ x'_1\ x'_3\ x'_6\ x_7\ x_4\ Y_5\ a_5;$
$a_2\ x'_1\ x'_3\ x'_6\ x_7\ x'_4\ Y_7\ a_4;$
$a_2\ x'_1\ x'_3\ x'_6\ x'_7\ Y_5\ a_5.$

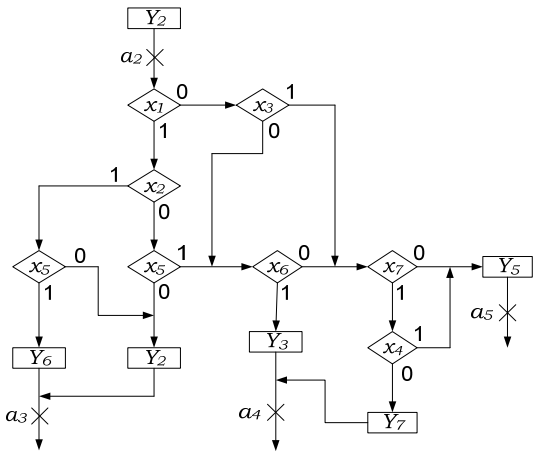

Figure 9. Subgraph of ASM

4.2.5 Transition tables of Mealy FSM. The graph of Mealy FSM in Fig. 8 has only 6 states and 16 arcs. Practically, however, we must construct FSMs with tens of states and more than one-two hundreds of transitions. In such a case, it is difficult to use a graph, so we will present it as a table. Table 1 for the same Mealy FSM has five columns:

- a_m – a current state;
- a_s – a next state;
- $X(a_m,a_s)$ – an input signal;
- $Y(a_m,a_s)$ – an output signal;
- H – a number of line.

Actually, immediately from ASM, we should write transition paths, one after another, into the transition table. In Table 1, $\sim x_t$ is used instead of x'_t for the inversion of x_t.

Now we will discuss what kind of FSM we have received. Our ASM G_1 in Fig. 6 which we used to construct FSM S_1 in Table 1, has seven logical conditions and ten microoperations. FSM S_1 has seven binary inputs in the column $X(a_m,a_s)$ and ten binary outputs in the column $Y(a_m,a_s)$. The input signal of this FSM (Fig. 10) is the 7-component vector, the output signal of this FSM is the 10-component vector.

Table 1. Direct transition table of Mealy FSM S_1

a_m	a_s	$X(a_m,a_s)$	$Y(a_m,a_s)$	H
a_1	a_2	$x_1x_2x_3$	y_1y_3	1
	a_3	$x_1x_2\sim x_3$	y_6y_7	2
	a_2	$x_1\sim x_2$	y_1y_2	3
	a_4	$\sim x_1$	y_4	4
a_2	a_2	x_4x_1	y_8y_9	5
	a_6	$x_4\sim x_1$	y_3y_4	6
	a_4	$\sim x_4$	y_4	7
a_3	a_6	1	y_3y_4	8
a_4	a_5	x_5	$y_5y_6y_7$	9
	a_2	$\sim x_5x_1$	y_8y_9	10
	a_6	$\sim x_5\sim x_1$	y_3y_4	11
a_5	a_2	x_6	y_8y_9	12
	a_1	$\sim x_6x_7$	$y_3y_6y_{10}$	13
	a_1	$\sim x_6\sim x_7$	-	14
a_6	a_1	x_6	y_6y_7	15
	a_6	$\sim x_6$	y_3y_4	16

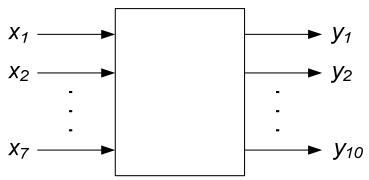

Figure 10. FSM as a black box

Let us take one of the rows from Table 1, for example row 3, and look at the behavior of FSM presented in this row. Our FSM transits from state a_1 into state a_2 when the product $x_1 x'_2 = 1$. It is clear that such a transition takes place for any input vector in which the first component is equal to 1, the second component is equal to 0. The values of other components are not important. Thus, we can say that the third row of Table 1 presents transitions from a_1 with any vector which is covered by cube $10xxxxx$. In other words, this row presents not one but $2^5 = 32$ transitions. In exactly the same way, the first and the second row present 16 transitions, the fourth row – 64 transitions and the eighth row – 128 transitions.

Two microoperations y_1, y_2, written in the third row of the output column, mean that two components y_1 and y_2 are equal to 1 and others are equal to 0 ($y_1 = y_2 = 1$; $y_3 = y_4 = \ldots = y_{10} = 0$) in the output vector. I remind you that if the operator, written in the operator vertex of some ASM, contains microoperations y_m, y_n, only these microoperations are equal to 1 and other microoperations are equal to 0 during implementation of this operator.

Let us compare Table 1 with a classical FSM representation in Table 3.7 from Chapter 3. If we would like to present our FSM with six states a_1, …, a_6 and seven inputs x_1, …, x_7 in the classical table, this table will have about $6x2^7$ rows, because each row of

this table describes only one FSM transition. In our Table 1 from this Chapter, we have only 16 rows because each row of such table presents lot of transitions.

The specific feature of such FSM is the multiplicity of inputs in the column $X(a_m,a_s)$, maybe several tens or even hundreds, but each product in one row contains only few variables from the whole set of input variables – as a rule, not more than 8 – 10 variables. It means that each time the values of the output variables depend only on the values of a small number of the input variables. Really, if, for example, FSM has 30 input variables, the total number of input vectors is equal to 2^{30}, and if each time the values of the output variables depended on the values of all the input variables, no designer could either describe or construct such an FSM.

Let us briefly discuss the correspondence between FSM S_1 (Table 1) and ASM G_1 (Fig. 6) which we used to construct FSM S_1. In Section 4.1.3 we got the value of ASM G_1

$$Y_b\, Y_1\, Y_2\, Y_4\, Y_2\, Y_3\, Y_8\, Y_e$$

for some random sequence of vectors (2) of logical conditions:

		x_1	x_2	x_3	x_4	x_5	x_6	x_7
Δ_1	=	1	0	1	0	1	1	1
Δ_2	=	0	1	1	0	1	0	0
Δ_3	=	1	0	1	0	0	1	0
Δ_4	=	0	1	0	0	0	0	1
Δ_5	=	1	1	0	1	1	1	0
Δ_6	=	1	1	0	0	1	0	1
Δ_7	=	0	1	1	1	0	0	0
Δ_8	=	0	1	0	1	0	0	1

Now we will find the response of FSM S_1 in the initial state a_1 to the same sequence of input vectors:

State sequence	a_1	a_2	a_4	a_2	a_4	a_5	a_1
Input sequence	Δ_1	Δ_2	Δ_3	Δ_4	Δ_5	Δ_6	
Response	y_1y_2	y_4	y_8y_9	y_4	$y_5y_6y_7$	$y_3y_6y_{10}$	(4)
Microinstructions	Y_1	Y_2	Y_4	Y_2	Y_3	Y_8	

Let FSM be in the initial state a_1 with the first vector $\Delta_1 = 1010111$ at its input. To determine the next state and the output we should find such a row in the array of transitions from a_1 (Table 1) that the product $X(a_m,a_s)$, written in this row, be equal to one at input vector Δ_1. Since $x_1x'_2(\Delta_1) = 1$ (the third row), FSM S_1 produces output signal $y_1y_2 = Y_1$ and transits into state a_2. Similarly, we find that $x'_4(\Delta_2)$ is equal to one at one of transitions from state a_2 and FSM transits to the state a_4 with the output signal $y_4 = Y_2$ (see row 7 in Table 1) etc. As a result, we get the response of FSM S_1 in the initial state a_1 to the input sequence Δ_1, …, Δ_6 in the fourth row of sequence (4).

As seen from this row, the FSM response is equal to the value of ASM G_1 for the same input sequence. Note, that we consider here only the FSM response until its return to the initial state a_1 and this response $Y_1\, Y_2\, Y_4\, Y_2\, Y_3\, Y_8$ corresponds to the value of ASM G_1 between the operator Y_b (vertex "Begin") and the operator Y_e (vertex "End").

Let us define *FSM S as implementing ASM G* if the response of this FSM in the state a_1 to any input sequence (until its return to the state a_1) is equal to the value of ASM G

for the same input sequence. From the considered method of synthesis of Mealy FSM S_1 from ASM G_1 it follows that this FSM S_1 implements ASM G_1.

4.2.6 Synthesis of Mealy FSM logic circuit. As in Chapter 3, we will construct a Mealy FSM logic circuit with the structure presented in Fig. 11. To design this circuit we will use an FSM structure table (Table 2). This table was constructed from the direct transition table (Table 1) by adding three additional columns:

- $K(a_m)$ – a code of the current state;
- $K(a_s)$ – a code of the next state;
- $D(a_m,a_s)$ – an input memory function.

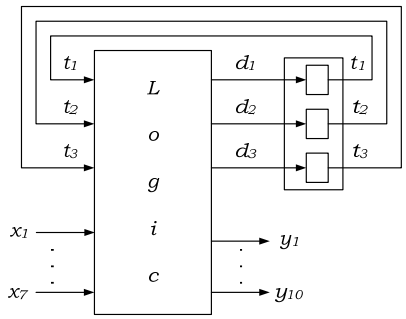

Figure 11. The structure for the Mealy FSM logic circuit

Table 2. Structure table of FSM S_1

a_m	$K(a_m)$	a_s	$K(a_s)$	$X(a_m,a_s)$	$Y(a_m,a_s)$	$D(a_m,a_s)$	H
a_1	001	a_2	000	$x_1x_2x_3$	y_1y_3	–	1
		a_3	101	$x_1x_2\sim x_3$	y_6y_7	d_1d_3	2
		a_2	000	$x_1\sim x_2$	y_1y_2	–	3
		a_4	010	$\sim x_1$	y_4	d_2	4
a_2	000	a_2	000	x_4x_1	y_8y_9	–	5
		a_6	100	$x_4\sim x_1$	y_3y_4	d_1	6
		a_4	010	$\sim x_4$	y_4	d_2	7
a_3	101	a_6	100	1	y_3y_4	d_1	8
a_4	010	a_5	110	x_5	$y_5y_6y_7$	d_1d_2	9
		a_2	000	$\sim x_5x_1$	y_8y_9	–	10
		a_6	100	$\sim x_5\sim x_1$	y_3y_4	d_1	11
a_5	110	a_2	000	x_6	y_8y_9	–	12
		a_1	001	$\sim x_6x_7$	$y_3y_6y_{10}$	d_3	13
		a_1	001	$\sim x_6\sim x_7$	-	d_3	14
a_6	100	a_1	001	x_6	y_6y_7	d_3	15
		a_6	100	$\sim x_6$	y_3y_4	d_1	16

To encode FSM states we constructed Table 3 where $p(a_s)$ is the number of appearances of each state in the next state column a_s in Table 2. The algorithm for state assignment is absolutely the same as in Chapter 3. First, we use the zero code for state a_2 with max $p(a_2) = 5$. Then codes with one '1' are used for states a_6, a_1, a_4 with the next max appearances and, finally, two codes with two 'ones' are used for the left states a_3 and a_5.

To fill column $D(a_m,a_s)$ it is sufficient to write there column $K(a_s)$ because the input of D flip-flop is equal to its next state. However, here we use the same notation as in column $Y(a_m,a_s)$ and write d_r in the column $D(a_m,a_s)$ if d_r is equal to 1 at the

corresponding transition (a_m,a_s) – equal to 1 in column $K(a_s)$. After that, the shaded part of Table 2 is something like a truth table with input variables $t_1, t_2, t_3, x_1, \ldots, x_7$ in the columns $K(a_m)$ and $X(a_m,a_s)$ and output variables (functions) $y_1, \ldots, y_{10}, d_1, d_2, d_3$ in the columns $Y(a_m,a_s)$ and $D(a_m,a_s)$.

Table 3. State assignment

a_s	$p(a_s)$	$t_1\ t_2\ t_3$
a_1	3	0 0 1
a_2	5	0 0 0
a_3	1	1 0 1
a_4	2	0 1 0
a_5	1	1 1 0
a_6	4	1 0 0

Let A_m be a product, corresponding to the state code $K(a_m)$, and X_h be the product of input variables, written in the column $X(a_m,a_s)$ in the h row. For example, from the column $K(a_m)$: $K(a_1) = 001$, then $A_1 = t'_1t'_2t_3$; $K(a_2) = 000$, then $A_2 = t'_1t'_2t'_3$; $K(a_3) = 101$, then $A_3 = t_1t'_2t_3$ etc. Immediately from the column $X(a_m,a_s)$ we get:

$$X_1 = x_1x_2x_3;\ X_2 = x_1x_2x'_3;\ X_6 = x_4x'_1;\ X_8 = 1;\ X_{16} = x'_6.$$

We call the term

$$e_h = A_m X_h$$

the product corresponding to the h row of the FSM structure table if a_m is the current state in this row. For example, from Table 2, we get:

$$\begin{aligned} e_1 &= t'_1t'_2t_3\ x_1x_2x_3; \\ e_2 &= t'_1t'_2t_3\ x_1x_2x'_3; \\ e_6 &= t'_1t'_2t'_3\ x_4x'_1; \\ e_8 &= t_1t'_2t_3; \\ e_{16} &= t_1t'_2t'_3\ x'_6. \end{aligned}$$

Let $H(y_n)$ is the set of rows with y_n in the column $Y(a_m,a_s)$. Then, as in the truth table:

$$y_n = \sum_{h \in H(y_n)} e_h.$$

For example, y_6 is written in rows 2, 9, 13, 15 in the column $Y(a_m,a_s)$. Then

$$y_6 = e_2 + e_9 + e_{13} + e_{15} = t'_1t'_2t_3x_1x_2x'_3 + t'_1t_2t'_3\ x_5 + t_1t_2t'_3\ x'_6x_7 + t_1t'_2t'_3\ x_6.$$

In exactly the same way, if $H(d_r)$ is the set of rows with d_r in the column $D(a_m,a_s)$, then

$$d_r = \sum_{h \in H(d_r)} e_h.$$

For example, d_2 is written in rows 4, 7, 9 in the column $D(a_m,a_s)$. Then

$$d_2 = e_4 + e_7 + e_9 = t'_1t'_2t_3\ x'_1 + t'_1t'_2t'_3\ x'_4 + t'_1t_2t'_3\ x_5.$$

Thus, immediately from Table 1 we can get expressions for outputs of circuit *"Logic"* in Fig. 11:

$$y_1 = e_1 + e_3 = t'_1t'_2t_3\ x_1x_2x_3 + t'_1t'_2t_3\ x_1x'_2;$$

$y_2 = e_3 = t'_1t'_2t_3\, x_1x'_2;$

$y_3 = e_1 + e_6 + e_8 + e_{11} + e_{13} + e_{16} = t'_1t'_2t_3\, x_1x_2x_3 + t'_1t'_2t'_3\, x_4x'_1 + t_1t'_2t_3 +$
$+ t'_1t_2t'_3\, x'_5x'_1 + t_1t_2t'_3\, x'_6x_7 + t_1t'_2t'_3\, x'_6;$
. . .
$y_{10} = e_{13} = t_1t_2t'_3\, x'_6x_7;$

$d_1 = e_2 + e_6 + e_8 + e_9 + e_{11} + e_{16} = t'_1t'_2t_3\, x_1x_2x'_3 + t'_1t'_2t'_3\, x_4x'_1 + t_1t'_2t_3 +$
$+ t'_1t_2t'_3\, x_5 + t'_1t_2t'_3\, x'_5x'_1 + t_1t'_2t'_3\, x'_6;$
$d_2 = e_4 + e_7 + e_9 = t'_1t'_2t_3\, x'_1 + t'_1t'_2t'_3\, x'_4 + t'_1t_2t'_3\, x_5;$
$d_3 = e_2 + e_{13} + e_{14} + e_{15} = t'_1t'_2t_3\, x_1x_2x'_3 + t_1t_2t'_3\, x'_6x_7 + t_1t_2t'_3\, x'_6x'_7 + t_1t'_2t'_3\, x_6.$

How many *different products* are there in these expressions? The answer is very simple – only sixteen, because we have 16 rows in Table 2 and only one product corresponds to one row. Thus, we should not write any expressions but can design the logic circuit immediately from the structure table. For that, it is sufficient to construct H AND-gates, one for each row, and $N+R$ OR-gates, one for each output variable y_n ($n = 1, \ldots, 10$ in our example) and one for each input memory function d_r ($r = 1, 2, 3$ in our example). The logic circuit of Mealy FSM is shown in Fig. 12. We have constructed 16 AND-gates, as there are 16 rows in its structure table. The number of OR-gates in this circuit is less than the number of input memory functions and output functions. Really, if y_n or d_r (y_2 and y_{10} in our example) are written only in one row of the structure table, it is not necessary to construct OR-gate for such y_n or d_r, we can get these signals from the corresponding AND-gates. Moreover, we have constructed one OR-gate for y_8 and y_9 since these outputs are always together in the structure table of Mealy FSM S_1.

4.2.7 ASM with waiting vertices. In this section, we will show that the algorithm for FSM synthesis does not change if ASM contains waiting vertices. In a waiting vertex, one of its outputs is connected with its input (see the ASM subgraph in Fig. 13). Let us find all transition paths from the state a_8. The first two are trivial – see the first two rows in Table 4.

To find the next path we should invert the variable x_7. The output *'0'* for x_7 brings us to the input of this conditional vertex. So, the next paths will be:

$$a_8 \sim x_7\, x_7\, x_{12}\, (y_{11})\, a_{13};$$
$$a_8 \sim x_7\, x_7\, \sim x_{12}\, (y_{23}, y_{29})\, a_{17}.$$

The products of input variables for both of these paths are equal to zero ($x'_7\, x_7 = 0$), so FSM cannot transit from the state a_8 to any other state when $x_7 = 0$. If FSM cannot transit into any other state, it remains in the same state a_8 or, we can say, it transits from a_8 to a_8 with $X(a_8, a_8) = x'_7$. No output variables are equal to *'1'* at this transition, so we have '–' in the column $Y(a_m, a_s)$ in the third row.

The next example (Fig.14) presents a general case. The only difference from the previous example – the waiting vertex is in the middle of the path. After the third path in Table 5 we should invert variable x_{11} and again return to the input of the conditional vertex with x_{11}. We can construct the following transitions paths:

$$a_{10} \sim x_4\, \sim x_{11}\, x_{11}\, x_{27}\, (y_{33})\, a_{22};$$
$$a_{10} \sim x_4\, \sim x_{11}\, x_{11}\, \sim x_{27}\, (y_7, y_{31})\, a_{17}.$$

The products for both of these paths are equal to zero. So, when $x_4 = 0$, we reached a waiting vertex with condition x_{11}. If $x_{11} = 0$ (return to the input), FSM transits from state a_{10} to state a_{10} (remains in this state) with input $x'_4 x'_{11}$ and each output variable is equal to zero (the forth row in Table 5).

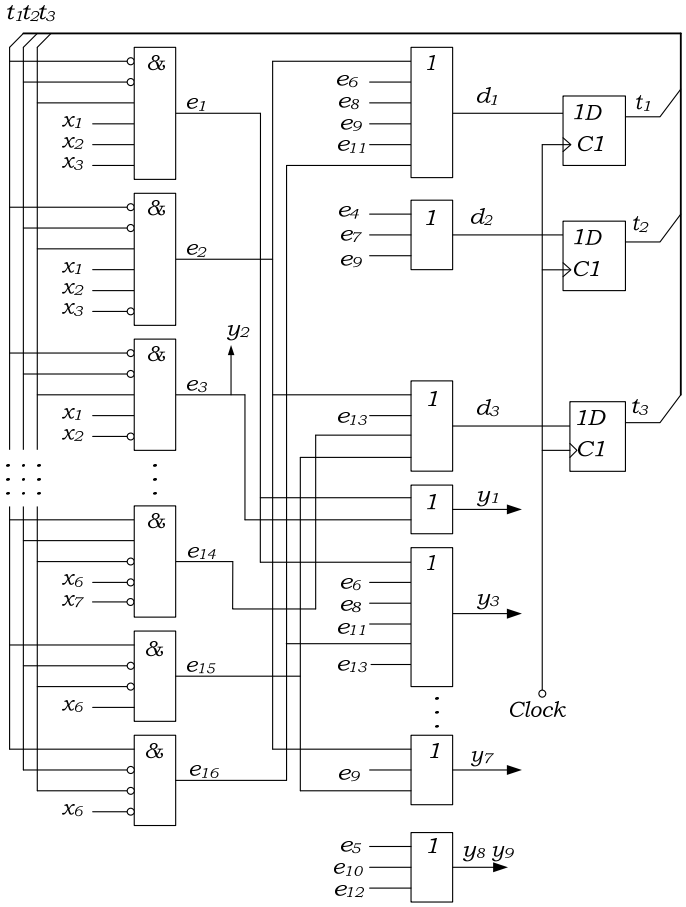

Figure 12. The logic circuit for Mealy FSM S_1

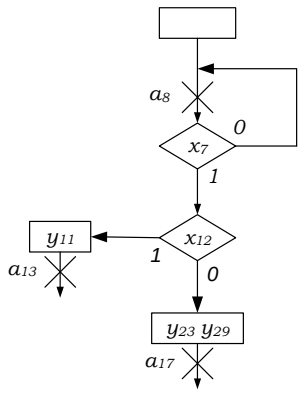

Table 4. Transitions for subgraph G_1

a_m	a_s	$X(a_m,a_s)$	$Y(a_m,a_s)$	H
		. . .		
a_8	a_{13} a_{17} a_8	x_7x_{12} $x_7 \sim x_{12}$ $\sim x_7$	y_{11} $y_{23}y_{29}$ –	

Figure 13. Subgraph G_1 with waiting vertex

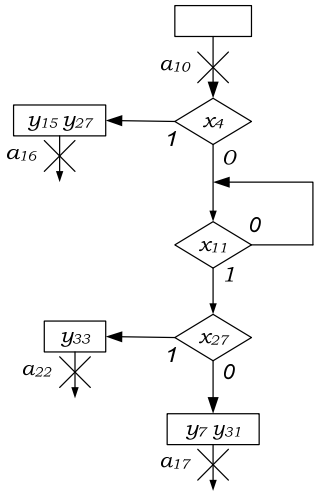

Table 5. Transitions for subgraph G_2

a_m	a_s	$X(a_m,a_s)$	$Y(a_m,a_s)$	H
		. . .		
a_{10}	a_{16}	x_4	$y_{15}y_{27}$	
	a_{22}	$\sim x_4 x_{11} x_{27}$	y_{33}	
	a_{17}	$\sim x_4 x_{11} \sim x_{27}$	$y_7 y_{31}$	
	a_{10}	$\sim x_4 \sim x_{11}$	–	

Figure 14. Subgraph G_2 with a waiting vertex

4.3 Synthesis of Moore FSM

As an example, we will use ASM G_1 in Fig. 15. A Moore FSM, implementing given ASM, can be constructed in two stages:

Stage 1. Construction of a marked ASM;
Stage 2. Construction of an FSM transition table.

At the first stage, the vertices "Begin", "End" and operator vertices are marked by symbols $a_1, a_2, \ldots, a_M$ as follows:

1. Vertices "Begin" and "End" are marked by the same symbol a_1;
2. Operator vertices are marked by different symbols $a_2, \ldots, a_M$;
3. All operator vertices should be marked.

Thus, while synthesizing a Moore FSM, symbols of states do not mark inputs of vertices following the operator vertices (as in the Mealy FSM) but operator vertices themselves. The number of marks is $T+1$, where T is the number of operator vertices in the marked ASM. In our example (Fig. 15), we need marks $a_1, \ldots, a_{10}$ for ASM G_1.

We will find the following transition paths in the marked ASM:

$$a_m \tilde{x}_{m1} \ldots \tilde{x}_{mR_m} a_s .$$

Thus, the transition path is the path between two operator vertices, containing R_m conditional vertices. Here, as above in the case of Mealy FSM, $\tilde{x}_{mr} = x_{mr}$, if in the transition path, we leave the conditional vertex with x_{mr} via output '1' and $\tilde{x}_{mr} = x'_{mr}$ if we leave the vertex with x_{mr} via output '0'. If $R_m = 0$ in such a path, there are no conditional vertices between two operator vertices, and this path turns into $a_m a_s$.

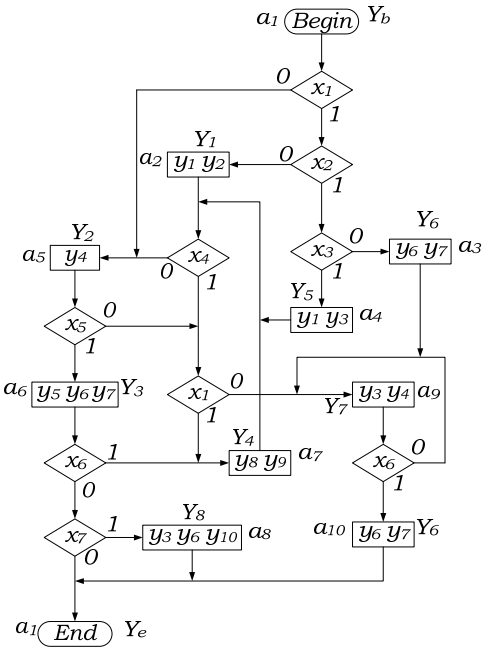

Figure 15. ASM G_1 marked for the Moore FSM synthesis

At the second stage we construct a transition table (or the state diagram) of the Moore FSM with states (marks) $a_1, \ldots, a_M$, obtained at the first stage. We have ten such states $a_1, \ldots, a_{10}$ in our example. Thus, the FSM contains as many states as the number of marks we get at the previous stage. Now we should define transitions between these states.

Thus, a Moore FSM has a transition from state a_m to state a_s with input $X(a_m, a_s)$ (see the upper subgraph in Fig. 16) if, in ASM, there is a transition path $a_m \tilde{x}_{m1} \ldots \tilde{x}_{mR_m} a_s$. Here $X(a_m, a_s)$ is a product of logical conditions written in this path: $X(a_m, a_s) = \tilde{x}_{m1} \ldots \tilde{x}_{mR_m}$. In exactly the same way, for the path $a_m a_s$ (see the lower subgraph in Fig. 16) we have a transition from state a_m to state a_s with input $X(a_m, a_s) = 1$, because the product of an empty set of variables is equal to zero. If a_m marks the operator vertex with operator Y_t, then $\lambda(a_m) = Y_t$, i.e. we identify the operator Y_t written in the operator vertex with this state a_m.

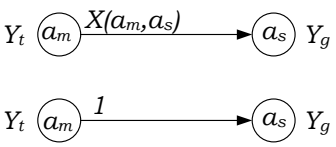

Figure 16. Subgraphs to illustrate transitions in the Moore FSM

The transition table for Moore FSM S_2, thus constructed, is presented in Table 6. The outputs are written in column $Y(a_m)$ immediately after the column with the current states. To design the logic circuit for this FSM we will use the structure presented in Fig. 17. It consists of two logic blocks (*Logic1* and *Logic2*) and memory block with four *D* flip-flops. *Logic1* implements input memory functions, depending on flip-flop

outputs $t_1, \ldots, t_4$ (feedback) and input variables $x_1, \ldots, x_7$. *Logic2* implements output functions, depending only on flip-flop outputs $t_1, \ldots, t_4$.

Table 6. The transition table of Moore FSM S_2

a_m	$Y(a_m)$	a_s	$X(a_m, a_s)$	h
a_1	–	a_4	$x_1x_2x_3$	1
		a_3	$x_1x_2{\sim}x_3$	2
		a_2	$x_1{\sim}x_2$	3
		a_5	$\sim x_1$	4
a_2	y_1y_2	a_7	x_4x_1	5
		a_9	$x_4{\sim}x_1$	6
		a_5	$\sim x_4$	7
a_3	y_6y_7	a_9	1	8
a_4	y_1y_3	a_7	x_4x_1	9
		a_9	$x_4{\sim}x_1$	10
		a_5	$\sim x_4$	11
a_5	y_4	a_6	x_5	12
		a_7	$\sim x_5x_1$	13
		a_9	$\sim x_5{\sim}x_1$	14
a_6	$y_5y_6y_7$	a_7	x_6	15
		a_8	$\sim x_6x_7$	16
		a_1	$\sim x_6{\sim}x_7$	17
a_7	y_8y_9	a_7	x_4x_1	18
		a_9	$x_4{\sim}x_1$	19
		a_5	$\sim x_4$	20
a_8	$y_3y_6y_{10}$	a_1	1	21
a_9	y_3y_4	a_{10}	x_6	22
		a_9	$\sim x_6$	23
a_{10}	y_6y_7	a_1	1	24

To encode FSM states we constructed Table 7 where $p(a_s)$, as before, is the number of appearances of each state in the next state column a_s in Table 6. The algorithm for state assignment is absolutely the same as in the case of Mealy FSM. First, we use the zero code for state a_9 with max $p(a_9) = 6$. Then codes with one '1' are used for states a_7, a_5, a_1 and a_2 with the next max appearences and, finally, five codes with two 'ones' are used for the left five states a_3, a_4, a_6, a_8 and a_{10}.

Table 8 is the structure table of the Moore FSM S_2. Its logic circuit is constructed in Fig. 18. In this circuit, A_m is a product of state variables for the state a_m ($m = 1, \ldots, 10$). As above we construct one AND-gate for one row of the structure table, but we need not construct the gates for rows 6, 8, 10, 14, 19 and 23, as all input memory functions are equal to zero in these rows (see the column $D(a_m,a_s)$ in Table 8). Neither

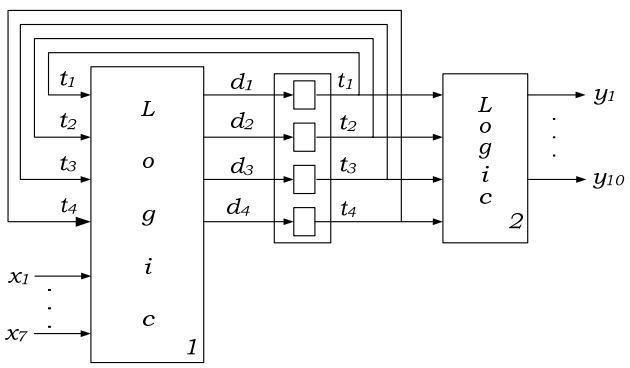

Figure 17. Moore FSM structure

Table 7. State assignment

a_s	$p(a_s)$	$t_1t_2t_3t_4$
a_1	3	0100
a_2	1	0010
a_3	1	1001
a_4	1	0110
a_5	4	0001
a_6	1	1100
a_7	5	1000
a_8	1	0011
a_9	6	0000
a_{10}	1	0101

Table 8. The structure table of the Moore FSM S_2

a_m	$Y(a_m)$	$K(a_m)$	a_s	$K(a_s)$	$X(a_m,a_s)$	$D(a_m,a_s)$	h
a_1	–	0100	a_4	0110	$x_1x_2x_3$	d_2d_3	1
			a_3	1001	$x_1x_2\sim x_3$	d_1d_4	2
			a_2	0010	$x_1\sim x_2$	d_3	3
			a_5	0001	$\sim x_1$	d_4	4
a_2	y_1y_2	0010	a_7	1000	x_4x_1	d_1	5
			a_9	0000	$x_4\sim x_1$	–	6
			a_5	0001	$\sim x_4$	d_4	7
a_3	y_6y_7	1001	a_9	0000	1	–	8
a_4	y_1y_3	0110	a_7	1000	x_4x_1	d_1	9
			a_9	0000	$x_4\sim x_1$	–	10
			a_5	0001	$\sim x_4$	d_4	11
a_5	y_4	0001	a_6	1100	x_5	d_1d_2	12
			a_7	1000	$\sim x_5x_1$	d_1	13
			a_9	0000	$\sim x_5\sim x_1$	–	14
a_6	$y_5y_6y_7$	1100	a_7	1000	x_6	d_1	15
			a_8	0100	$\sim x_6x_7$	d_2	16
			a_1	0011	$\sim x_6\sim x_7$	d_3d_4	17
a_7	y_8y_9	1000	a_7	1000	x_4x_1	d_1	18
			a_9	0000	$x_4\sim x_1$	–	19
			a_5	0001	$\sim x_4$	d_4	20
a_8	$y_3y_6y_{10}$	0011	a_1	0100	1	d_2	21
a_9	y_3y_4	0000	a_{10}	0101	x_6	d_2d_4	22
			a_9	0000	$\sim x_6$	–	23
a_{10}	y_6y_7	0101	a_1	0100	1	d_2	24

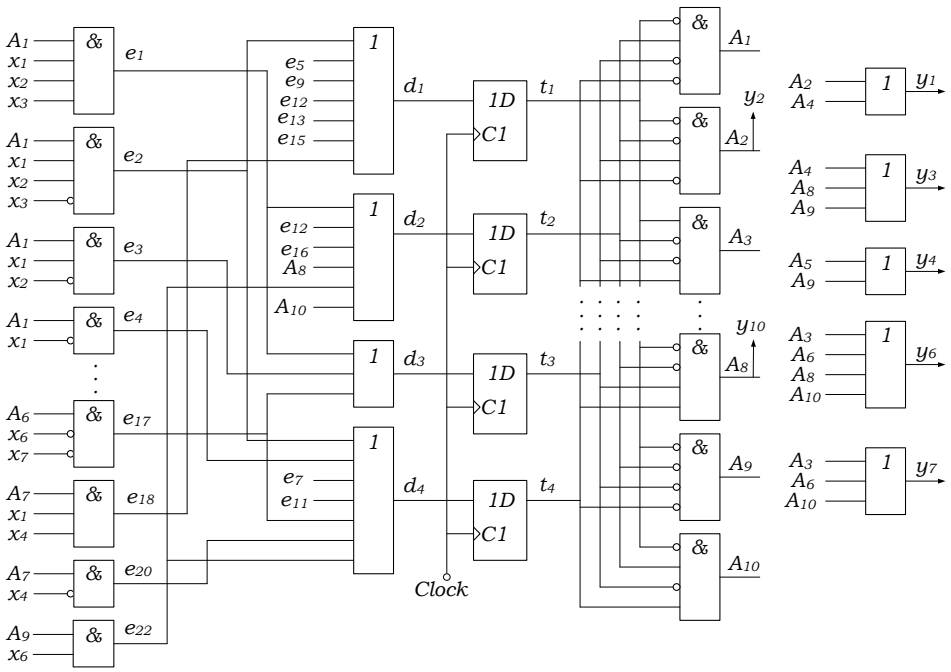

Figure 18. The logical circuit for the Moore FSM S_2

do we construct the gates for rows 21 and 24, since there are no input variables in the corresponding terms e_{21} and e_{24}: $e_{21} = A_8$ and $e_{24} = A_{10}$ and we use A_8 and A_{10} directly as inputs in OR-gate for d_2.

4.4. Synthesis of Combined FSM model

In this book we will use two kinds of transition tables – direct and reverse. In a *direct table* (Table 9), transitions are ordered according to the current state (the first column in this table) – first we write all transitions *from* the state a_1, then *from* the state a_2 , etc. In a *reverse table* (Table 10), transitions are ordered according to the next state (the second column in this table) – first we write all transitions *to* the state a_1, then *to* the state a_2 , etc.

Table 9. Direct transition table of Mealy FSM S_3

a_m	a_s	$X(a_m,a_s)$	$Y(a_m,a_s)$	h
a1	a2	x6	y8y9	1
a1	a5	~x6*x7	y6	2
a1	a5	~x6*~x7	y3y6y10	3
a2	a2	x4*x1	y1y2	4
a2	a6	x4*~x1	y3y4	5
a2	a4	~x4	y4	6
a3	a6	1	y3y5	7
a4	a1	x5	--	8
a4	a2	~x5*x1	y8y9	9
a4	a6	~x5*~x1	y3y4	10
a5	a2	x1*x2*x3	y1y3	11
a5	a3	x1*x2*~x3	y1y4	12
a5	a2	x1*~x2	y1y2	13
a5	a4	~x1	y4	14
a6	a5	x6	y6y7	15
a6	a6	~x6	y3y5	16

Now we will discuss the transformation of Mealy FSM into Combined FSM and synthesis of its logic circuit. I remind here that Combined FSM has two kinds of output signals:

1. Signals depending on the current state and the current input (as in the Mealy model);
2. Signals depending only on the current state (as in the Moore model);

As an example, we use the transition table of Mealy FSM in Table 9. Our first step is to construct a reverse table for this FSM (Table 10).

Fig. 19,a illustrates all transitions into state a_5 of Mealy FSM from Table 10. Here we have three transitions with different outputs but all of them contain the same output variable y_6. So, we can identify this output variable y_6 with the state a_5 as a Moore signal (see Fig. 19,b).

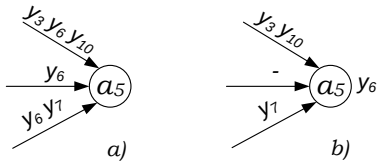

Figure 19. Transformation from Mealy FSM to Combined FSM

Table 10. Reverse transition table of Mealy FSM S_3

a_m	a_s	$X(a_m,a_s)$	$Y(a_m,a_s)$	H
a4	a1	x5	--	1
a2	a2	x4*x1	y1y2	2
a1	a2	x6	y8y9	3
a4	a2	~x5*x1	y8y9	4
a5	a2	x1*x2*x3	y1y3	5
a5	a2	x1*~x2	y1y2	6
a5	a3	x1*x2*~x3	y1y4	7
a2	a4	~x4	y4	8
a5	a4	~x1	y4	9
a1	a5	~x6*x7	y6	10
a1	a5	~x6*~x7	y3y6y10	11
a6	a5	x6	y6y7	12
a4	a6	~x5*~x1	y3y4	13
a2	a6	x4*~x1	y3y4	14
a3	a6	1	y3y5	15
a6	a6	~x6	y3y5	16

After this, the transformation of Mealy FSM into Combined model is trivial. Let us return to the reverse Table 10 and begin to construct the reverse transition table of Combined FSM S_4 (Table 11 In Table10, we look at the transitions to each state, beginning from transitions to state a_1. Let Y_s be the set of output variables at the transitions into state a_s ($Y_5 = \{y_3, y_6, y_7, y_{10}\}$ in Fig19,a or in Table 10) and Y_s^{Moore} be the subset of common output variables at all transitions into a_s ($Y_5^{Moore} = \{y_6\}$ in Fig19,a or in Table 10). Then, in Table 11, we delete Y_s^{Moore} from the column $Y(a_m,a_s)$ at each row with transition to a_s and write Y_s^{Moore} next to a_s in the column $Y(a_s)$. In our example:

$Y_1^{Moore} = Y_2^{Moore} = \varnothing$; $Y_3^{Moore} = \{y_1, y_4\}$; $Y_4^{Moore} = \{y_4\}$;
$Y_5^{Moore} = \{y_6\}$; $Y_6^{Moore} = \{y_3\}$.

Table 11. Reverse transition table of Combined FSM S_4

a_m	a_s	$Y(a_s)$	$X(a_m,a_s)$	$Y(a_m,a_s)$	H
a4	a1	--	x5	--	1
a1	a2	--	x6	y8y9	2
a2	a2	--	x4*x1	y1y2	3
a4	a2	--	~x5*x1	y8y9	4
a5	a2	--	x1*x2*x3	y1y3	5
a5	a2	--	x1*~x2	y1y2	6
a5	a3	y1y4	x1*x2*~x3	--	7
a2	a4	y4	~x4	--	8
a5	a4	y4	~x1	--	9
a1	a5	y6	~x6*~x7	y3y10	10
a1	a5	y6	~x6*x7	--	11
a6	a5	y6	x6	y7	12
a2	a6	y3	x4*~x1	y4	13
a3	a6	y3	1	y5	14
a4	a6	y3	~x5*~x1	y4	15
a6	a6	y3	~x6	y5	16

Now we consider the design of the logic circuit of Combined FSM. For this, let us return to the Mealy FSM S_1 with direct transition Table 1. Its reverse transition table is presented in Table 12. Immediately from this table we construct the direct transition table of Combined FSM S_1 (Table 13). To construct the logic circuit for this FSM we should encode the states and construct FSM structure table. But before state assignment we will make one more step.

Table 12. Reverse transition table of Mealy FSM S_1

a_m	a_s	$X(a_m,a_s)$	$Y(a_m,a_s)$	H
a_5	a_1	$\sim x_6x_7$	$y_3y_6y_{10}$	1
a_5		$\sim x_6\sim x_7$	-	2
a_6		x_6	y_6y_7	3
a_1	a_2	$x_1x_2x_3$	y_1y_3	4
a_1		$x_1\sim x_2$	y_1y_2	5
a_2		x_4x_1	y_8y_9	6
a_4		$\sim x_5x_1$	y_8y_9	7
a_5		x_6	y_8y_9	8
a_1	a_3	$x_1x_2\sim x_3$	y_6y_7	9
a_1	a_4	$\sim x_1$	y_4	10
a_2		$\sim x_4$	y_4	11
a_4	a_5	x_5	$y_5y_6y_7$	12
a_2	a_6	$x_4\sim x_1$	y_3y_4	13
a_3		1	y_3y_4	14
a_4		$\sim x_5\sim x_1$	y_3y_4	15
a_6		$\sim x_6$	y_3y_4	16

Unlike the transition table of the Mealy FSM, Table 13 contains many empty entries in the column $Y(a_m,a_s)$. It means that all output variables are equal to zero in these rows. If, after state assignment, we get an empty entry in column $D(a_m,a_s)$ for such a row, we shouldn't construct a product for this row, because all output variables and input memory functions are equal to zero in this row. Now we will try to maximize the number of such rows in the structure table of S_5.

Table 13. Direct transition table of Combined FSM S_5

a_m	$Y(a_m)$	a_s	$X(a_m,a_s)$	$Y(a_m,a_s)$	H
a_1	--	a_2	$x_1x_2x_3$	y_1y_3	1
	--	a_3	$x_1x_2\sim x_3$	-	2
	--	a_2	$x_1\sim x_2$	y_1y_2	3
	--	a_4	$\sim x_1$	-	4
a_2	--	a_2	x_4x_1	y_8y_9	5
	--	a_6	$x_4\sim x_1$	-	6
	--	a_4	$\sim x_4$	-	7
a_3	y_6y_7	a_6	1	-	8
a_4	y_4	a_5	x_5	-	9
		a_2	$\sim x_5x_1$	y_8y_9	10
		a_6	$\sim x_5\sim x_1$	--	11
a_5	$y_5y_6y_7$	a_2	x_6	y_8y_9	12
		a_1	$\sim x_6x_7$	$y_3y_6y_{10}$	13
		a_1	$\sim x_6\sim x_7$	-	14
a_6	y_3y_4	a_1	x_6	y_6y_7	15
		a_6	$\sim x_6$	-	16

Table 13 contains one row with empty entry in the column $Y(a_m,a_s)$ for the next states a_1 (row 14) and a_3 (row 2), two rows for a_4 (rows 4 and 7), one row for a_5 (row 9) and

four rows for a_6 (rows 6, 8, 11 and 16). This information is presented in the first two columns of Table 14, $z(a_s)$ is the number of empty entries in column $Y(a_m,a_s)$ for the next state a_s in Table 13. So, if we use zero code for states a_1 or a_3 or a_5, we shouldn't construct a product for one row ($z(a_1) = z(a_3) = z(a_5) = 1$), if we use zero code for state a_4 – for two rows ($z(a_4) = 2$); but if we use zero code for state a_6, we will construct four product less ($z(a_6) = 4$). Thus, we use code *000* for state a_6 with max $z(a_s)$. State assignment for other states is presented in Table 15. We have used here the same algorithm as we have used previously for Mealy and Moore models.

Table 14. Next states with zero outputs

a_s	$z(a_s)$	$t_1\ t_2\ t_3$
a_1	1	
a_3	1	
a_4	2	
a_5	1	
a_6	4	0 0 0

Table 15. State assignment

a_s	$p(a_s)$	$t_1\ t_2\ t_3$
a_1	3	0 1 0
a_2	5	0 0 1
a_3	1	1 0 1
a_4	2	1 0 0
a_5	1	1 1 0
a_6	4	0 0 0

Table 16. Structure table of Combined FSM S_5

a_m	$Y(a_m)$	$K(a_m)$	a_s	$K(a_s)$	$X(a_m,a_s)$	$Y(a_m,a_s)$	$D(a_m,a_s)$	H
a_1	--	010	a_2	001	$x_1x_2x_3$	y_1y_3	d_3	1
	--		a_3	101	$x_1x_2\sim x_3$	-	d_1d_3	2
	--		a_2	001	$x_1\sim x_2$	y_1y_2	d_3	3
	--		a_4	100	$\sim x_1$	-	d_1	4
a_2	--	001	a_2	001	x_4x_1	y_8y_9	d_3	5
	--		a_6	000	$x_4\sim x_1$	-	-	6
	--		a_4	100	$\sim x_4$	-	d_1	7
a_3	y_6y_7	101	a_6	000	1	-	-	8
a_4	y_4	100	a_5	110	x_5	-	d_1d_2	9
			a_2	001	$\sim x_5x_1$	y_8y_9	d_3	10
			a_6	000	$\sim x_5\sim x_1$	-	-	11
a_5	$y_5y_6y_7$	110	a_2	001	x_6	y_8y_9	d_3	12
			a_1	010	$\sim x_6x_7$	$y_3y_6y_{10}$	d_2	13
			a_1	010	$\sim x_6\sim x_7$	-	d_2	14
a_6	y_3y_4	000	a_1	010	x_6	y_6y_7	d_2	15
			a_6	000	$\sim x_6$	-	-	16

Table 16 is the structure table of Combined FSM S_5. We have three kinds of output variables here:

1. *Only Mealy signals:* y_1, y_2, y_8, y_9, y_{10}. They are written in column $Y(a_m,a_s)$ and are not written in column $Y(a_m)$ in Table 16;
2. *Only Moore signals:* y_4, y_5. They are written in the column $Y(a_m)$ and are not written in column $Y(a_m,a_s)$ in Table 16;
3. *Combined signals:* (both Mealy and Moore type) y_3, y_6, y_7. They are written in both columns $Y(a_m,a_s)$ and $Y(a_m)$ in Table 16.

The logic circuit of FSM S_5 is constructed in Fig. 20. In this circuit, A_m is a product of state variables for the state a_m ($m = 1, \ldots, 6$). The left part of this circuit, exactly as in the synthesis of the Mealy FSM logic circuit, implements input memory functions d_1, d_2, d_3 and *Mealy signals* y_1, y_2, y_8, y_9, y_{10}. As above, we construct one AND-gate for one row of the structure table, but we need not construct the gates for rows 6, 8, 11,

16 because all output variables and input memory functions are equal to zero in these rows in the columns $Y(a_m,a_s)$ and $D(a_m,a_s)$ in Table 16. As in the Mealy case, we do not construct OR gates for y_2 and y_{10} since they appear only once in the column $Y(a_m,a_s)$.

Moore signals y_4, y_5 are constructed as in the synthesis of Moore FSM logic circuit. Signal y_4 appears twice near the states a_4 and a_6 in the column $Y(a_m)$, so $y_4 = A_4 + A_6$ and we construct OR gate for this signal. Output signal y_5 appears only once in the column $Y(a_m)$ for the state a_5, so we get it straight from A_5: $y_5 = A_5$.

Combined signal y_6 is written in rows 13 and 15 in the column $Y(a_m,a_s)$ and near the states a_3 and a_5 in the column $Y(a_m)$, so

$$y_6 = e_{13} + e_{15} + A_3 + A_5.$$

Exactly in the same way

$$y_3 = e_1 + e_{13} + A_6; \quad y_7 = e_{15} + A_3 + A_5.$$

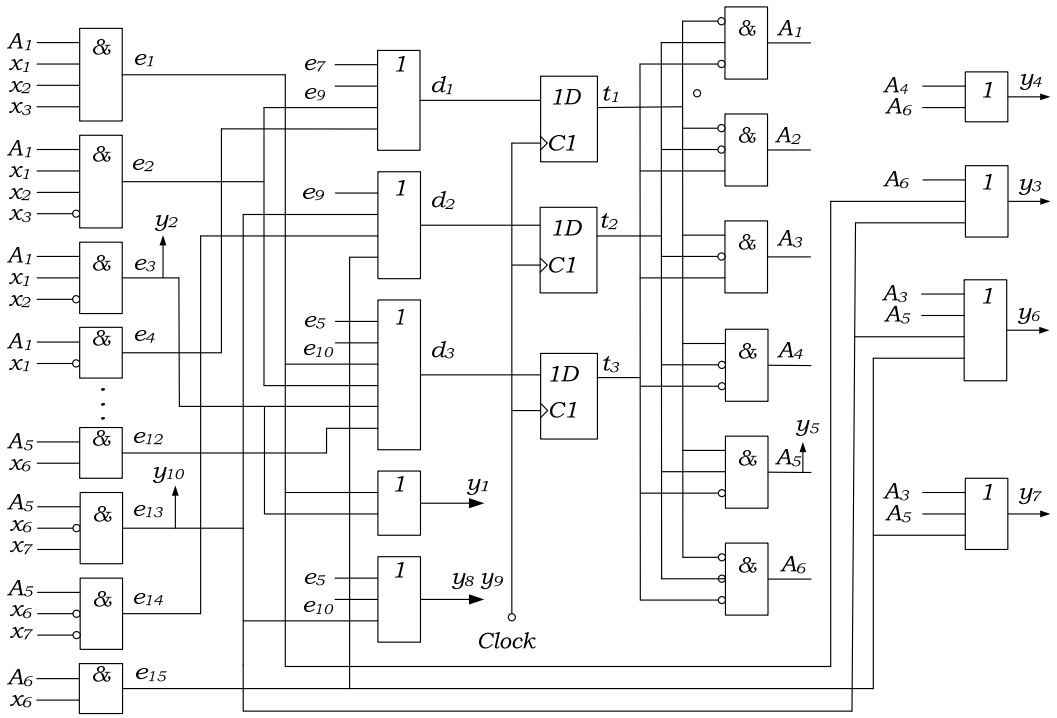

Figure 20. Logic circuit of Combined FSM S_5

4.5. FSM decomposition

In this section, we will discuss a very simple model for FSM decomposition. As an example, we use Mealy FSM S_6 (Table 17) and a partition π on the set of its states:

$$\pi = \{A_1, A_2, A_3\};$$
$$A_1 = \{a_2, a_3, a_9\}; A_2 = \{a_4, a_7, a_8\}; A_3 = \{a_1, a_5, a_6\}.$$

The number of component FSMs in the FSM network is equal to the number of blocks in partition π. Thus, in our example, we have three component FSMs S^1, S^2, S^3.

Let B^m is the set of states in the component FSM S^m. B^m contains the corresponding block of the partition π plus one additional state b_m. So, in our example:

S^1 has the set of states $B^1 = \{a_2, a_3, a_9, b_1\}$;
S^2 has the set of states $B^2 = \{a_4, a_7, a_8, b_2\}$;
S^3 has the set of states $B^3 = \{a_1, a_5, a_6, b_3\}$.

Table 17. Mealy FSM S_6

a_m	a_s	$X(a_m,a_s)$	$Y(a_m,a_s)$	H
a1	a3	x1*x2*x3	y1y2	1
a1	a6	x1*x2*~x3	y2y12	2
a1	a1	x1*~x2	y1y2	3
a1	a5	~x1	y1y2y12	4
a2	a2	x6	--	5
a2	a3	~x6	y3y5	6
a3	a3	x10	y3y5	7
a3	a9	~x10*x4	y10y15	8
a3	a8	~x10*~x4	y5y8y9	9
a4	a6	x7	y13	10
a4	a4	~x7*x9	y13y18	11
a4	a8	~x7*~x9	y13y14	12
a5	a6	x1	y16y17	13
a5	a5	~x1	y7y11	14
a6	a1	x5	y1y2	15
a6	a1	~x5	y16y17	16
a7	a2	x8	y14y18	17
a7	a4	~x8	y13y18	18
a8	a7	x9	y4y6	19
a8	a4	~x9	y6	20
a9	a9	x11*x6	y10y15	21
a9	a2	x11*~x6	y5y8y9	22
a9	a3	~x11	y3y8y9	23

To construct a transition table for each component FSM we should define the transitions between the states of these FSMs. For this, each transition between two states a_i and a_j of Mealy FSM S_6 from Table 17 should be implemented one after another as one or two transitions in component FSMs. There are two possible cases:

1. In Mealy FSM S_6, there is a transition between a_i and a_j (Fig. 21, left) and both of these states are in the same component FSM S^m. In such a case, we will have the same transition in this component FSM S^m (Fig. 21, right). It means that we must rewrite the corresponding row from the table of FSM S_6 into the table of component FSM S^m.

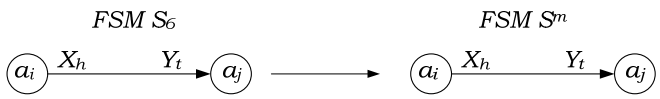

Figure 21. Two states a_i and a_j are in the same component FSM

2. Two states a_i and a_j are in different component FSMs (Fig. 22). Let a_i be in the component FSM S^m ($a_i \in B^m$) and a_j be in the component FSM S^p ($a_j \in B^p$). In such a case, one transition of FSM S_6 should be presented as two transitions – one in the component FSM S^m and one in the component FSM S^p:
   - FSM S^m transits from a_i into its additional state b_m with the same input X_h. At its output, we have the same output variables from set Y_t plus one additional output variable z_j, where index j is the index of state a_j in the component FSM S^p.
   - FSM S^p is in its additional state b_p. It transits from this state into state a_j with input signal z_j, that is an additional output variable in the component FSM S^m. The output at this transition is Y_0 – the signal with all output variables being equal to zero.

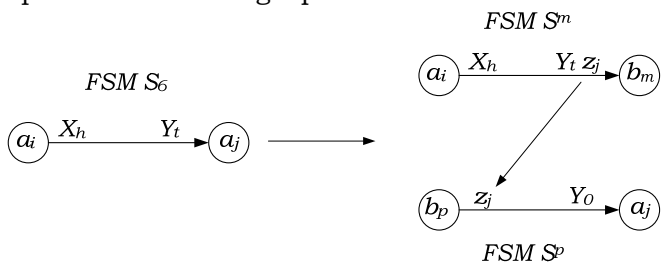

Figure 22. Two states a_i and a_j are in the different component FSMs

Thus, the procedure for FSM decomposition is reduced to:

a) Copying the row

$$a_i \quad a_j \quad X(a_i, a_j) \quad Y(a_i, a_j)$$

from the table of the decomposed FSM S to the table of the component FSM S^m if both states a_i and a_j are the states of S^m;

b) Replacing the row

$$a_i \quad a_j \quad X(a_i, a_j) \quad Y(a_i, a_j)$$

in the table of the decomposed FSM S by the row

$$a_i \quad b_m \quad X(a_i, a_j) \quad Y(a_i, a_j)\ z_j$$

in the table of the component FSM S^m, and by the row

$$b_p \quad a_j \quad z_j \quad --$$

in the table of the component FSM S^p, if a_i is the state of S^m and a_j is the state of S^p.

As a result of decomposition of FSM S_6, we obtain the network with three component FSMs in Fig. 23. Their transition tables are presented in Tables 18 – 20.

Now we will illustrate some examples of transitions for cases (a) and (b):

- In FSM S_6, there is a transition from state *a2* to state *a3* with input *~x6* and output *y3y5* (row 6 in Table 17). As both these states *a2* and *a3* are in the same component FSM S_1, in this FSM there is a transition from *a2* to *a3* with the same input *~x6* and the same output *y3y5* (row 2 in Table 18). Exactly in the same way, we rewrite row 12 of Table 17 into row 3 of Table 19 and row 2

of Table 17 into row 2 of Table 20 because the current states and the next states are in the same component FSMs.

- In FSM S_6, there is a transition from state *a3* to state *a8* with input *~x10*~x4* and the output *y5y8y9* (row 9 in Table 17). Since *a3* is the state of component FSM S^1 and *a8* is the state of another component FSM S^2, in FSM S^1 there is a transition from *a3* to *b1* with the same input *~x10*~x4* and output *y5y8y9z8* (row 5 in Table 18). The last output *z8* is the input of FSM S^2 that wakes this FSM up and transits it from state *b2* to state *a8* (row 8 in Table 19). Similarly, we convert row 1 of Table 17 into two rows – the first in Table 20 and the tenth in Table 18 etc. Note that we add the last row in each FSM table to remain component FSMs in the state b_m when each z_j is equal to zero.

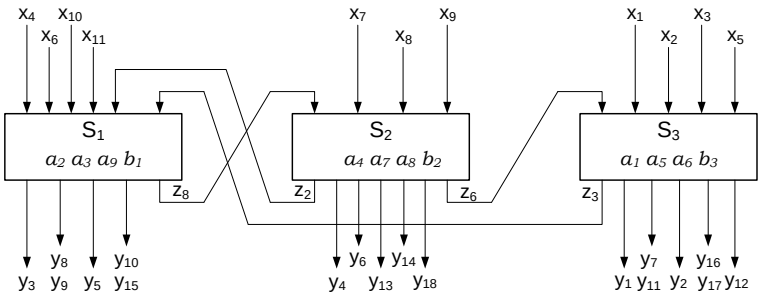

Figure 23. Network with three component FSMs

Table 18. Component FSM S^1

a_m	a_s	$X(a_m,a_s)$	$Y(a_m,a_s)$	H
a2	a2	x6	--	1
a2	a3	~x6	y3y5	2
a3	a3	x10	y3y5	3
a3	a9	~x10*x4	y10y15	4
a3	b1	~x10*~x4	y5y8y9z8	5
a9	a9	x11*x6	y10y15	6
a9	a2	x11*~x6	y5y8y9	7
a9	a3	~x11	y3y8y9	8
b1	a2	z2	--	9
b1	a3	z3	--	10
b1	b1	~z2*~z3	--	11

Table 19. Component FSM S^2

a_m	a_s	$X(a_m,a_s)$	$Y(a_m,a_s)$	H
a4	b2	x7	y13z6	1
a4	a4	~x7*x9	y13y18	2
a4	a8	~x7*~x9	y13y14	3
a7	b2	x8	y14y18z2	4
a7	a4	~x8	y13y18	5
a8	a7	x9	y4y6	6
a8	a4	~x9	y6	7
b2	a8	z8	--	8
b2	b2	~z8	--	9

Let us discuss how this network works. Let *a1* be an initial state in FSM S_6. After decomposition, state a_1 is in FSM S^3, so, at the beginning, just FSM S^3 is in state *a1*. Other FSMs are in states *b1* and *b2* correspondingly. It is possible to say that they "are sleeping" in these states. FSM S^3 transits from the state to the state until *x1*x2*x3 = 1* in state *a1* (see row 1 in Table 20). Only at this transition FSM S^3 produces output signal *z3* and transits into state *b3* (sleeping state). This signal *z3* is the input signal of FSM S^1. It wakes FSM S^1 up and transits it from the sleeping state *b1* to state *a3* (see row 10 in Table 18). Now FSM S^1 transits from the state to the state until, in state *a3,* it transits into state *b1* with input signal *~x10*~x4 = 1* and wakes FSM S^2 up by signal *z8* (see row 5 in Table 18 and row 8 in Table 19).

Table 20. Component FSM S^3

a_m	a_s	$X(a_m,a_s)$	$Y(a_m,a_s)$	H
a1	b3	x1*x2*x3	y1y2z3	1
a1	a6	x1*x2*~x3	y2y12	2
a1	a1	x1*~x2	y1y2	3
a1	a5	~x1	y1y2y12	4
a5	a6	x1	y16y17	5
a5	a5	~x1	y7y11	6
a6	a1	x5	y1y2	7
a6	a1	~x5	y16y17	8
b3	a6	z6	--	9
b3	b3	~z6	--	10

Unlike FSMs S^1 and S^3, the component FSM S^2 has two possibilities to wake other component FSMs up – in state *a4* with input signal *x7 = 1* (row 1 of Table 19) and in state *a7* with input signal *x8 = 1* (row 4 in the same Table), etc. Thus, each time all component FSMs, except one, are in the states of type b_m and only one of them is in the state of type a_i.