

## **Asynchronous sequential circuits**

- 1 Difference between synchronous and asynchronous automaton
- 2 Transients in asynchronous sequential circuits
- 3 Asynchronous sequential circuits operating in Burst Mode
- 4 Outputs of asynchronous sequential circuits
- 5 Internal code selection
- 6 Initial state setting
- 7 C-element

Attachment

## Asynchronous sequential circuits

### 1 Difference between synchronous and asynchronous sequential circuits

The basic structure of asynchronous sequential circuits is similar to that of synchronous circuits. The general diagram is shown in Fig.1. In the case of a synchronous sequential circuit (Fig. 1 on the left), the next state is prepared between two clock pulses by combinational circuits which process the current input and state (feedback) signals. The status signals are considered to be the output signals of the flip-flops, which remain stable between the clock pulses. The next state is determined by the inputs of flip-flops, but it will not become evident until the next tact.

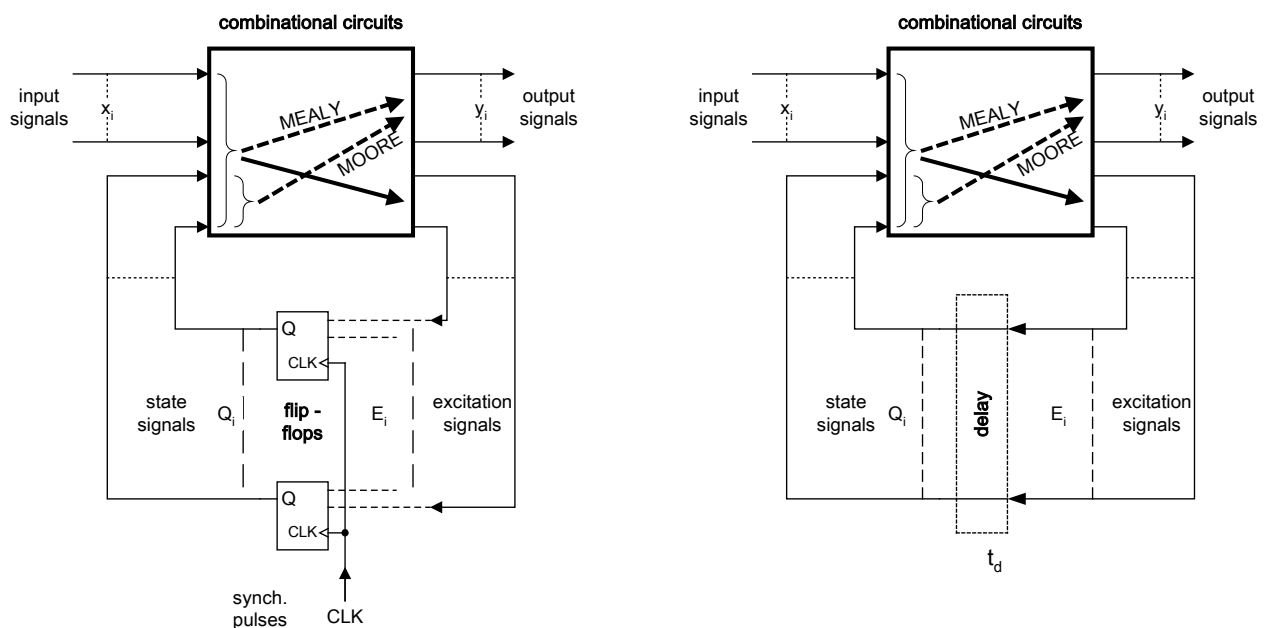


Fig. 1: Synchronous seq. circuit on the left  
Asynchronous seq. circuit on the right

For asynchronous circuit (Fig. 1 right), there are also input, output and internal states. However, synchronous flip-flops are not included in the feedback and the feedback signals thus act together with the input signals. Therefore an internal status can change immediately (exactly - after a small delay) **after changing the input**.

Since there are no circuits in the feedback of an asynchronous sequential circuit, the question arises as to how the present and next states actually differ. It should be noted that combinational circuits always have a certain delay, which, however, is not apparent from the logical expressions describing their function. Therefore, this delay is understood as a hypothetical **delay element**, shifted from the combinational circuits out to the feedback path. Ideal undelayed signal on the outputs of the combinational circuits are therefore theoretically identical to those which, as status signals, are delivered to the combinational circuits. The designation was chosen the same as for the synchronous circuit. For example, for the outputs of the combinational circuits, the designation "excitation signals" is retained, even if they are not applied to the inputs of the FF, as was the case for synchronous sequential circuits. This makes it possible to use a similar design procedure as for synchronous sequential circuits.

As a rule, status variables are considered unchanged (due to the feedback delay) shortly after the change of external inputs. Conversely, changes in the signals at the output of the combinational circuits are considered to be **immediate**.

As for the **output** function (see dashed arrow in Fig. 1), it is similar to synchronous circuits - it is also just a combination function and its specific shape is different for Moore and Mealy circuit. We will deal with it only marginally and we will pay attention mainly to the transition function.

The design of asynchronous sequential circuits is complicated, especially for larger circuits. The reason is the direct connection from the outputs of the combinational circuits to their inputs. Thus, **false impulses** on state signals, caused by hazards in combinational circuits and signal **races** on all their inputs, are here fully manifested.

However, asynchronous sequential circuits have a **number of advantages** for which they are considered a significant alternative in the design of large integration circuits. It is especially:

- **Fast response** of outputs to changes in inputs, as no synchronization pulse is expected.
- **Lower consumption** from the power source, since there is no circuitry associated with the generation and distribution of clock pulses.
- **Lower level of radiated interference**, as signal changes are distributed over time in contrast to synchronous circuits, where changes occur simultaneously in most internal circuits at the edges of the sync. pulses. The interference spectrum from synchronous systems typically shows distinct spectral lines at the clock frequency and its harmonics. For asynchronous systems, the spectrum is flat.

## 2 Transients in asynchronous sequential circuits

After the change of input state  $I$ , the internal state  $S$  also changes, but even if the input  $I$  still holds, then another change of the internal state can follow, etc.. Four possible cases exist:

1. The input state  $I$  does not change and in cooperation with the present internal state  $S$  leads to the **same state**  $S$  - it is therefore **stable**. Stable states are bold and underlined in the transition table.
2. The new (and then continued) input state  $I$  in cooperation with the present internal state  $S$  leads to the **new state**  $S$ , which is then stable – a **single transition**.
3. The new (and then continued) input state  $I$ , in cooperation with the present internal state  $S$ , leads to the new state  $S$ , from there to another state  $S$ , etc., until a stable state is reached after the **final number** of transitions. States through which the circuit "traveled", are temporary – a **multiple transition**.
4. The new (and then continued) input state  $I$  in cooperation with the internal state  $S$  leads to a new state  $S$ , hence to another state  $S$ , but a stable state is never reached. The circuit **oscillates**. This is an **undesirable** situation.

Fig. 2 shows a transition graph and a corresponding transition table.

Changes of input and internal states will be symbolically marked as follows:

$S_i \& I_r \rightarrow S_k \& I_s$  ..... the input  $I_r$  has changed to  $I_s$  and the state of  $S_i$  has changed to  $S_k$

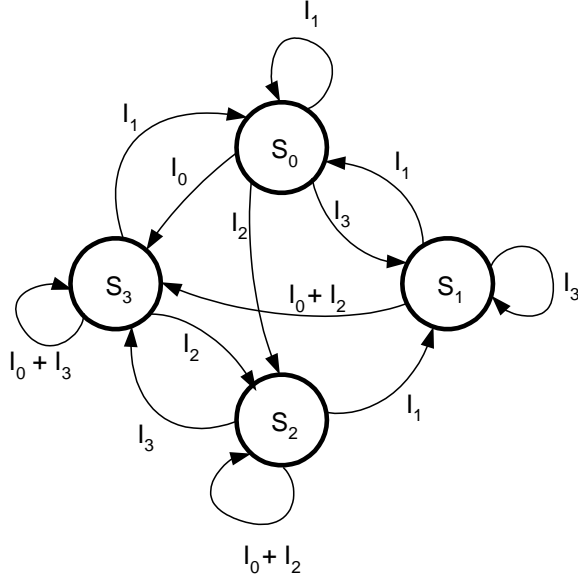


Fig. 2: Transition graph

	I <sub>0</sub>	I <sub>1</sub>	I <sub>2</sub>	I <sub>3</sub>
S <sub>0</sub>	S <sub>3</sub>	<b><u>S<sub>0</sub></u></b>	S <sub>2</sub>	S <sub>1</sub>
S <sub>1</sub>	S <sub>3</sub>	S <sub>0</sub>	S <sub>3</sub>	<b><u>S<sub>1</sub></u></b>
S <sub>2</sub>	<b><u>S<sub>2</sub></u></b>	S <sub>1</sub>	<b><u>S<sub>2</sub></u></b>	S <sub>3</sub>
S <sub>3</sub>	<b><u>S<sub>3</sub></u></b>	S <sub>0</sub>	S <sub>2</sub>	<b><u>S<sub>3</sub></u></b>

Tab 1: Transition table

In the transition table, the headings of the rows are **present** states, the cells in the table are **next** states. The state is **stable**, if the next state is equal to the present state, i.e. the state in the heading is the same as the state in the cell. The transition function of an asynchronous sequential circuit can be understood as moving from the cell with a present stable state  $S_j$  and input  $I_a$  horizontally in a row to a new cell under the input  $I_b$ . We get to the cell in which the new, temporary state is written. From there we proceed vertically to the row containing the same state. It is desirable that it be stable. The combination of the input and state is called "**total state**".

For example, at state  $S_0$  and input  $I_1$ , the next state is still **S<sub>0</sub>**, the state did not change and is stable. Now, when changing from  $I_1$  to  $I_3$ , the originally stable state **S<sub>0</sub>** changes (by a horizontal shift) to a temporary  $S_1$ . Now we shift down to the state  $S_1$ , which is evidently stable.

Now another example of a more complicated path through a temporary state. The stable state **S<sub>2</sub>** with the input  $I_2$ , after the input changed to  $I_1$ , passes to  $S_1$ , then to  $S_0$ , and finally to the stable state **S<sub>0</sub>**.

Symbolically, the transitions between total states can be written as follows (stable states are marked in bold):

$$\begin{aligned}
 \underline{S_0} \& I_1 &\rightarrow \underline{S_0} \& I_1 \\
 \underline{S_0} \& I_1 &\rightarrow S_1 \& I_3 \rightarrow \underline{S_1} \& I_3 \\
 \underline{S_2} \& I_2 &\rightarrow S_1 \& I_1 \rightarrow S_0 \& I_1 \rightarrow \underline{S_0} \& I_1
 \end{aligned}$$

Further steps in the design of our sequential circuit (Fig. 2) will be rewriting the transition table in the set of symbolic representations of transitions, coding of the state and inputs, and

substituting codes for the symbolic states – very similarly with the solution of synchronous sequential circuits. The input and internal codes are in Tab. 2 and Tab. 3.

	$x_1$	$x_0$
$I_0$	0	0
$I_1$	0	1
$I_2$	1	0
$I_3$	1	1

Tab. 2: Input code for Tab 1

	$Q_1$	$Q_0$
$S_0$	0	0
$S_1$	0	1
$S_2$	1	0
$S_3$	1	1

Tab. 3: Input code for Tab 1

We write the transitions between states in a form of a system of symbolic transitions and substitute input and internal codes:

symbolic transitions	codes substituted	excitation variables	
		$E_1$	$E_0$
$S_0 \& I_0 \rightarrow S_3$	$\bar{Q}_1 \cdot \bar{Q}_0 \cdot \bar{x}_1 \cdot \bar{x}_0 \rightarrow Q_1 \cdot Q_0$		1
.....	.....etc.....	.....	
$S_0 \& I_2 \rightarrow S_2$	$\bar{Q}_1 \cdot \bar{Q}_0 \cdot x_1 \cdot \bar{x}_0 \rightarrow Q_1 \cdot \bar{Q}_0$		1
.....	.....etc.....	.....	
$S_3 \& I_3 \rightarrow S_3$	$Q_1 \cdot Q_0 \cdot x_1 \cdot x_0 \rightarrow Q_1 \cdot Q_0$		1

At the outputs of the combinational circuit we get the excitation variables  $E_1$  and  $E_0$  as functions of  $Q_1$ ,  $Q_0$ ,  $x_1$ ,  $x_0$ . If we look for the function  $E_i = f_i(Q_1, Q_0, x_1, x_0)$  in the sum of products, the **rows** in which there is value **one** in column  $E$  will be important. From the fully filled table, the functions can be expressed in maps and minimized so:

$$E_1 = \bar{x}_0 + Q_1 \cdot x_1$$

$$E_0 = x_1 \cdot x_0 + Q_1 \cdot \bar{Q}_0 \cdot x_0 + Q_0 \cdot \bar{x}_1 \cdot \bar{x}_0 + \bar{Q}_1 \cdot Q_0 \cdot \bar{x}_0 + \bar{Q}_1 \cdot \bar{x}_1 \cdot \bar{x}_0$$

A circuit diagram can now be drawn (Fig. 3):

The output function, unspecified in the specification, will be realized by a combinational circuit on the basis of state variables  $Q_1$  and  $Q_0$ , or also input variables  $x_1$  and  $x_0$ , as well as in the case of a synchronous sequential circuit.

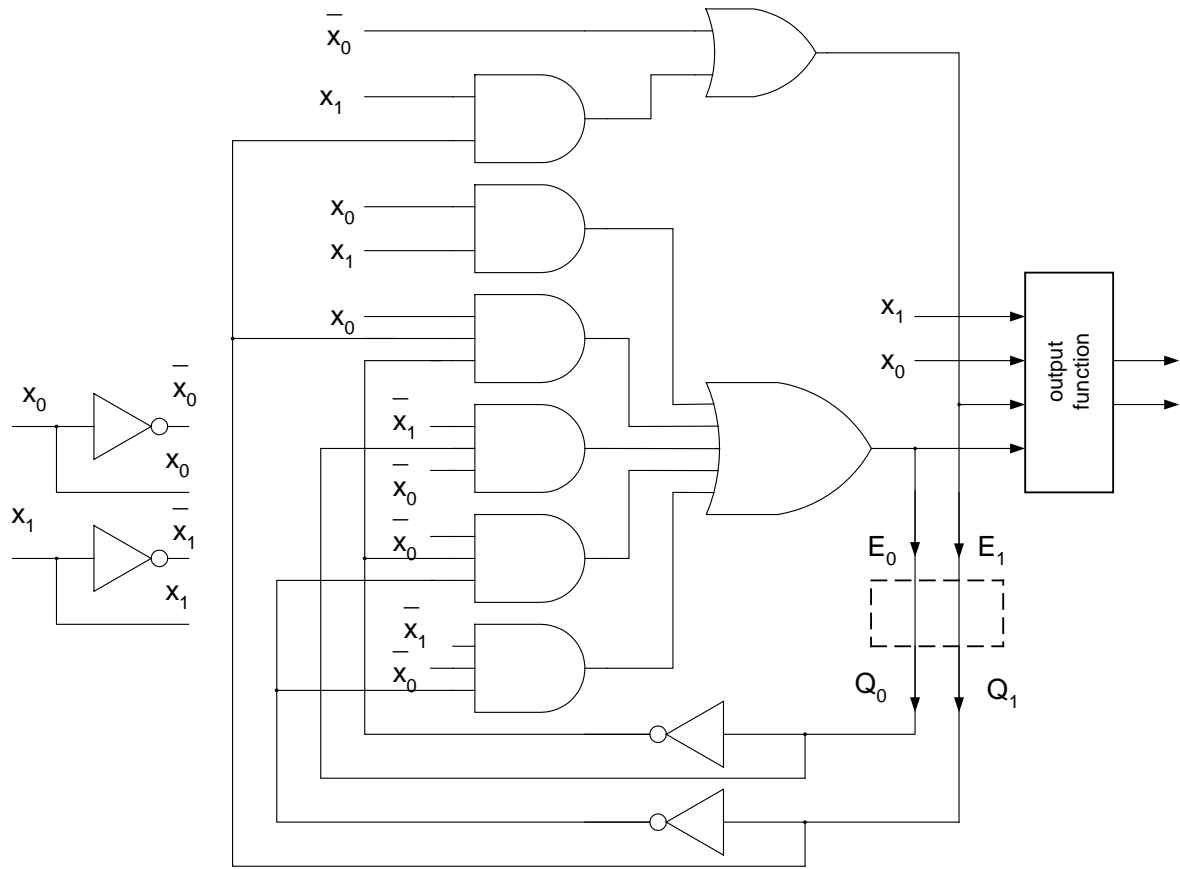


Fig. 3: Sequential circuit diagram according to Fig. 2 or Tab. 1

Even if the entire design were done perfectly, most probably the circuit will not work properly due to the signal races and hazards in the combinational circuits. Their effect could be completely eliminated by inserting a register in the feedback path (dashed in Fig. 3). Such a modification, however, would change the circuit from asynchronous to synchronous.

When changing input (and also internal) states, the **order of changes** of individual variables can be important - a phenomenon called "signal races". For example, the change of inputs from  $I_0$  (code 00) to  $I_3$  (code 11) from Tab. 1 will never happen like  $00 \rightarrow 11$ , but always as  $00 \rightarrow 01 \rightarrow 11$  or  $00 \rightarrow 10 \rightarrow 11$ . Both (or all) variables will never change simultaneously, but always with slight differences in their delays. The states between the present and the next states are called "**intermediate**". The question is how the sequential circuit responds to the intermediate states.

In an ideal scenario in our example the two signals change simultaneously, then:

$$\underline{S}_2 \& I_0 \rightarrow S_3 \& I_3 \rightarrow \underline{S}_3 \& I_3 \quad .$$

In the real case,  $x_0$  may change first, the input change is  $00 \rightarrow 01$ , and the state changes so:

$$\underline{S}_2 \& I_0 \rightarrow S_1 \& I_1 \rightarrow S_0 \& I_1 \rightarrow \underline{S}_0 \& I_1,$$

Next the slower  $x_1$  changes and the input change is 01→11. The states change so:

$$\underline{S}_0 \& I_1 \rightarrow S_1 \& I_3 \rightarrow \underline{S}_1 \& I_3$$

But also  $x_1$  could have changed first and thus the input change is 00 → 10:

$$\underline{S}_2 \& I_0 \rightarrow \underline{S}_2 \& I_2,$$

and then only to the next change of the slower  $x_0$ , and thus the input change is 10 → 11:

$$\underline{S}_2 \& I_2 \rightarrow S_3 \& I_3 \rightarrow \underline{S}_3 \& I_3$$

Apparently, in the order of input changes 00→01→11, state  $S_1$  is the result (which is an error against the ideal case), while in the order of changes 00→10→11, state  $S_3$  is the result (which agrees with the ideal case). Evidently the choice of the appropriate **input and internal codes** can have a big impact, although success may be far from guaranteed.

Another and much safer option is to limit the changes of the input states to only those when only one variable changes, and then to redesign the combinational circuits - introduction of redundancy can eliminate hazards. At the same time, this eliminates the races of input variables, as they cannot occur when only one of them changes. Worse problem are the so-called "**substantial hazards**" arising from changes of the **input and state** signals. It is possible that the delay in some path of status signals is short compared to the other path, and thus the change of some status signals can occur during changes in some input signals. This can result in a combinational circuit malfunction and an incorrect circuit transition. Returning to Table 1, we assume a steady state of  $\underline{S}_0 \& I_1$ . When changing the input to  $I_0$ , we would expect the circuit to pass to a steady state  $\underline{S}_3 \& I_0$ . But if the internal states were coded according to Tab. 3 ( $S_0 \dots 00$ ,  $S_3 \dots 11$ ), it may happen that the state variable  $Q_1$  changes before  $Q_0$ . Then there is an internal state  $\underline{S}_2 \dots 10$  in the circuit for a while, but according to the table of transitions it will be stable! That, of course, is a mistake.

Substantial hazards can only sometimes be eliminated by changing the internal code and redesigning the combinational circuits, but this is usually not possible. The situation can be theoretically solved by inserting additional, precisely set delays into selected signal paths. However, this has no practical application.

Another difficult to solve problem occurs in the cases when input signals can change at any time - even at a time when the internal state of the circuit is **not yet stabilized** after the previous change. Therefore, a limitation on the moments of input changes is usually introduced. They are allowed only after the internal state of the circuit is stabilized. The signals thus change in two phases - first, the input signals change with constant state signals. False pulses at the outputs of the combinational circuits must be avoided, mainly through redundancy. Then the status signals change, but now with the input signals stable. Solving two partial problems is easier than solving a whole complex problem. The above mode of operation of an asynchronous circuit is called "**fundamental mode**".

However, the question naturally arises as to how to ensure this regime. Apparently, input changes can be tied to some regular rhythm, determined on the basis of knowledge of individual delays - but this brings us to the field of synchronous systems. The second option is to establish the link between two circuits - **source** of data and destination or **target** of data. Via this link, the sequential circuits inform each other. The target confirms to the source that the new data (signals) can be accepted, and the source confirms to the target that the new data is available. This principle is called "**handshake**" - abbreviated as **HS** - and is very widely applied. Chapter "Design of large-scale systems" is devoted to this issue.

The same applies to the internal state signals - hazards and races can cause incorrect transitions between states. A suitable internal code, in which only one status signal changes at a time, can help.

The above problems indicate the difficulty with implementation of asynchronous sequential circuits, especially in case of large-scale circuitry. The advantages of asynchronous systems are, however, so significant that the development of methods and computer programs proceeds fast and has achieved significant progress. Huge problems with the design of complex asynchronous systems can be overcome by the fact that the entire large system can be composed of **smaller** asynchronous units that cooperate with each other. The small units can be solved relatively easily. A typical structure, suitable for this purpose, is a **chained system**, also called "**pipeline**". Chapter "Design of large-scale systems" is devoted to this issue.

The practical use of small asynchronous units in a **standard structure** is usually well possible even with drastic restrictions on the shape of the transition table, i.e. on the transition function. This helps to eliminate the effects resulting from races, hazards, etc..

### 3 Asynchronous sequential circuits operating in Burst Mode

Sequential circuits operating in the **Burst Mode** – **BM** – have limited variety of changes at the input and, therefore, limited freedom in constructing the transition table. The benefit of these restrictions is the absence of problems with the signal races and hazards. Admitted are changes on **any input** or group of inputs with the exception of multiple changes during transitions between states – no "ringing" on input signals. The changes **do not** have to be simultaneous, which is a significant advantage. In the transition table, each temporary state must lead **directly** to a stable state, i.e. paths through several states and state oscillations are prohibited.

The result is a sequential circuit that in each stable state always "waits" for a single input state, which can be achieved by either simultaneous or sequential changes of input signals. Only when this input state is reached will the circuit pass to a new internal state. This mode, although it may be limiting for a general sequential circuit, is on the other hand suitable for the implementation of many practical circuits, in particular circuits for controlling **communication** between mutually cooperating asynchronous units. It is important that the circuit must be designed in such a way that the achievement of a new state **must not** depend on the order of changes of inputs – i.e. the state variables must not be "spoiled" by false impulses caused by hazards or races of input signals. These are very harsh conditions requiring harsh restrictions on the shape of the table of transitions. On the other hand, if for



some applications these limitations are not a problem, we have a design method that can lead to a **quick and reliable** design of asynchronous circuits.

BM mode uses **different symbolic** to describe changes in input signals. Instead of describing the old and new input states, e.g.  $x_2.\bar{x}_1.\bar{x}_0 \rightarrow \bar{x}_2.\bar{x}_1.x_0$ , we will mark only the **changes** of the input signals and ignore those signals where no changes have occurred - for the given case we will write  $x_2-$ ,  $x_0+$ . The "-" symbol indicates a change from 1 to 0 and the "+" symbol indicates a change from 0 to 1; the missing  $x_1$  means no change. In the same way will be described the transitions between states in the transition graph.

An example is shown in Fig. 4. a). In the transitions graph are transitions between states marked by changes in input signals. It is assumed that if there will be no change or only a partial change in input signals, there will be no change in internal states. Since only changes - and not whole input states - are marked in the graph, it is important to determine the **initial state** of the circuit - we will mark it with thicker and underlined letters. Next, in Fig. 4. b), is the corresponding transition table. For comparison the Fig. 4. c) shows a corresponding graph of transitions constructed by the classical method. **Unspecified states** can be filled appropriately. Their existence is quite common and depends on the technical limitations of the input signal source, for which some combinations of values are impossible. Fig. 4 shows a very simple variant of the graph and table, where the paths between the states do not branch. In real circuits, branching is common in at least some states. A more complex case will be shown at the end of the text.

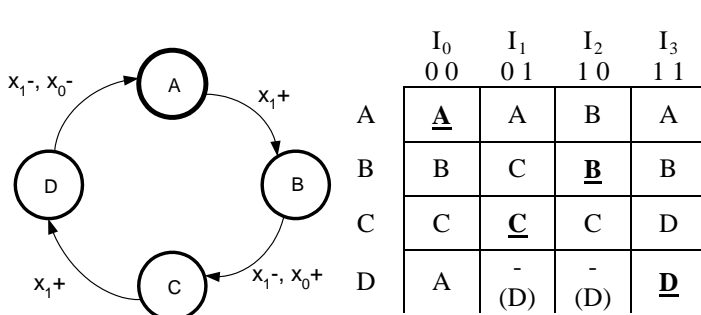


Fig.4.a

	I <sub>0</sub> 0 0	I <sub>1</sub> 0 1	I <sub>2</sub> 1 0	I <sub>3</sub> 1 1
A	<u><b>A</b></u>	A	B	A
B	B	C	<u><b>B</b></u>	B
C	C	<u><b>C</b></u>	C	D
D	A	(D)	(D)	<u><b>D</b></u>

Fig.4.b

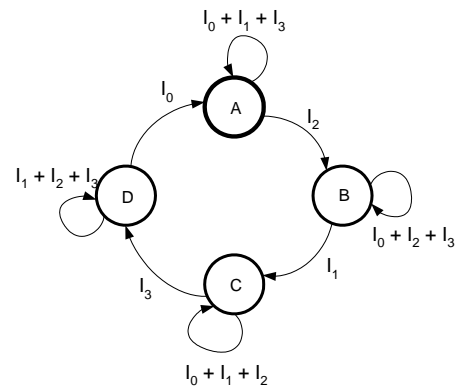


Fig.4.c

Fig. 4: A simple example of the BM method

When constructing a state diagram and a transition table, it is necessary to respect a number of **restrictions** necessary for the removal of false impulses on state variables in BM sequential circuits. Some of these limitations are common to all asynchronous circuits and some (in addition) are specific to BM solutions. Individual restrictions will be explained.

*1. Input changes are only permitted after all internal signals have stabilized after a previous internal state change.*

The solution of hazards and races in combinational circuits with possible changes of all signals is extremely difficult and rather impossible. As a standard, therefore, in the case of asynchronous sequential circuits, a change of the inputs at constant state signals is assumed first and only then a change of the state signals, now with constant input signals. This division

of activity is possible due to the delay of the combinational circuits - see also Fig. 1 and the relevant text. A favorable consequence is a much simpler design of combinational circuits.

*2. During the transition between states, each input signal may change at most once.*

Oscillations at the inputs are prohibited. If an input signal changes, its further change is possible only after reaching a new internal state and stabilizing all internal signals.

*3. If the inputs do not change, the internal state must not change either.*

Automatic changing of status signals without prior input change would indicate circuit instability and oscillation.

*4. Each temporary state leads directly to a stable state.*

Paths through several temporary states should not exist. But sometimes this possibility is allowed.

*5. If two or more input signals change, then each input state that is temporarily arisen during successive input signal changes must lead to the stable internal state.*

This ensures that any transient combinations of input signal values - "intermediate states" - cannot cause an undesirable change in the present state. This eliminates the effect of races and hazards.

*6. If a path from a current state branches, then a group of changes leading to one next state must not be a subset of changes leading to another next state.*

This condition concerns alternative paths between internal states. For one, a certain group of input changes is prescribed, after which the state changes. If for reaching the other state it is sufficient to change only some of these inputs, then this state can be reached before changes leading to the first state are completed. E.g. if  $S_i \rightarrow S_j$  when  $x_2-$ ,  $x_1+$ ,  $x_0-$ , and  $S_i \rightarrow S_k$  when  $x_2-$ ,  $x_0-$ , then if  $x_0-$  occurs first and then  $x_2-$ , or  $x_2-$  first and then  $x_0-$  (the order must not matter), the condition for the transition to  $S_k$  is completely fulfilled and the sequential circuit is no longer "waiting" for a possible further change  $x_1+$ . However, if the change  $x_1+$  comes first, then the path to  $S_k$  is no longer possible and the circuit after the changes  $x_2-$  and  $x_0-$  goes to  $S_i$ . That would contradict to the main characteristics BM circuits, i.e. the independence on the order of input changes.

*7. Each steady state must be achieved by only one combination of input signal values.*

If more paths lead to a certain state, they must always be enabled by the same combination of input signals.

*8. Each field in the transition table that is not reachable by any sequence of input changes is marked as an unused state. Restrictions of input changes depend on the circuits which deliver input signals.*

The occurrence of a large number of unspecified states is common in practice. These states can advantageously be filled so that the above conditions 1 to 7 are met.

We will further deal with the circuit according to Fig. 4. The table shows compliance with condition No. 5, except for unspecified conditions, which for some reason cannot occur at the input. In the line D we can fill all "D". Note in particular the lines B in which the steady state B &  $I_2$ , which should change to the stable state C when the input is changed from  $I_2$  to  $I_1$ . However, this means that both input variables change and input combinations  $I_0$  and  $I_3$  can also exist for a short time. Since in the row B are under these inputs states B, these temporary input states cause no change in the present internal state. Rows A and C also meet the above condition, and in the row D it would be possible to fill D in all unspecified cells (but if they cannot occur, they can be filled in other ways).

We select the input and internal code as follows:

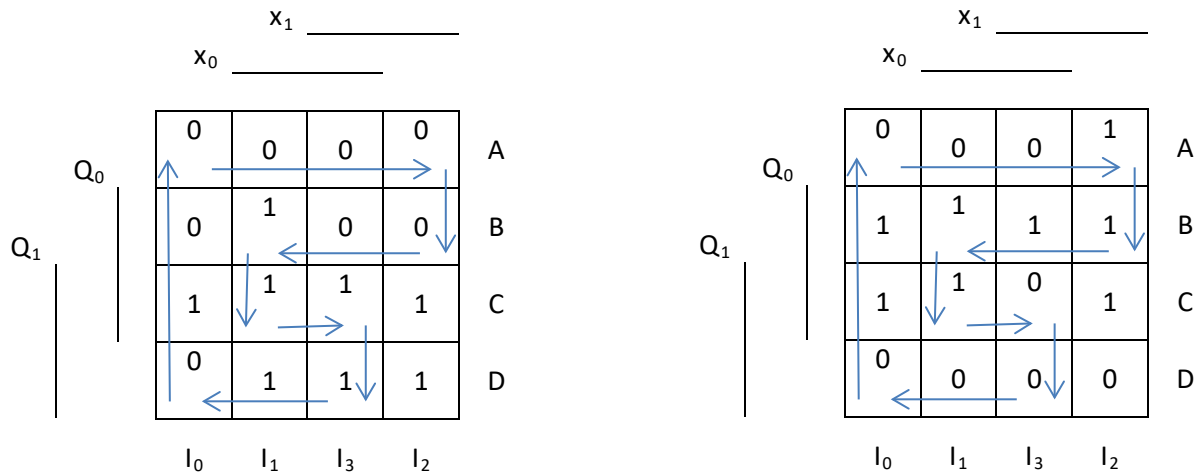
	$x_1 \ x_0$		$Q_1 \ Q_0$
$I_0$	0 0	A	0 0
$I_1$	0 1	B	0 1
$I_2$	1 0	C	1 1
$I_3$	1 1	D	1 0

Transition tables of Fig. 4.b) we can now convert to the system of symbolic representations of transitions and substitute selected codes for symbols - see Tab. 4:

	transitions symbolically	transitions in codes $E_1 \ E_0$	state variables
1	$A \ \& \ (I_0+I_1+I_3) \rightarrow A$	$\bar{Q}_1 \cdot \bar{Q}_0 \cdot (\bar{x}_1 \cdot \bar{x}_0 + \bar{x}_1 \cdot x_0 + x_1 \cdot x_0)$	$\rightarrow \bar{Q}_1 \cdot \bar{Q}_0 \quad 0 \quad 0$
2	$A \ \& \ I_2 \rightarrow B$	$\bar{Q}_1 \cdot \bar{Q}_0 \cdot x_1 \cdot \bar{x}_0$	$\rightarrow \bar{Q}_1 \cdot Q_0 \quad 0 \quad 1$
3	$B \ \& \ (I_0+I_2+I_3) \rightarrow B$	$\bar{Q}_1 \cdot Q_0 \cdot (\bar{x}_1 \cdot \bar{x}_0 + x_1 \cdot \bar{x}_0 + x_1 \cdot x_0)$	$\rightarrow \bar{Q}_1 \cdot Q_0 \quad 0 \quad 1$
4	$B \ \& \ I_1 \rightarrow C$	$\bar{Q}_1 \cdot Q_0 \cdot \bar{x}_1 \cdot x_0$	$\rightarrow Q_1 \cdot Q_0 \quad 1 \quad 1$
5	$C \ \& \ (I_0+I_1+I_2) \rightarrow C$	$Q_1 \cdot Q_0 \cdot (\bar{x}_1 \cdot \bar{x}_0 + \bar{x}_1 \cdot x_0 + x_1 \cdot \bar{x}_0)$	$\rightarrow Q_1 \cdot Q_0 \quad 1 \quad 1$
6	$C \ \& \ I_3 \rightarrow D$	$Q_1 \cdot Q_0 \cdot x_1 \cdot x_0$	$\rightarrow Q_1 \cdot \bar{Q}_0 \quad 1 \quad 0$
7	$D \ \& \ (I_1+I_2+I_3) \rightarrow D$	$Q_1 \cdot \bar{Q}_0 \cdot (\bar{x}_1 \cdot x_0 + x_1 \cdot \bar{x}_0 + x_1 \cdot x_0)$	$\rightarrow Q_1 \cdot \bar{Q}_0 \quad 1 \quad 0$
8	$D \ \& \ I_0 \rightarrow A$	$Q_1 \cdot \bar{Q}_0 \cdot \bar{x}_1 \cdot \bar{x}_0$	$\rightarrow \bar{Q}_1 \cdot \bar{Q}_0 \quad 0 \quad 0$

Tab. 4: Symbolic transitions for example from Fig. 4

Due to the small total number of variables, maps can now be produced for the state variables  $E_1$  and  $E_0$ , which gives the clear view - see Fig. 5. In the map for  $E_1$  will be ones in the cells corresponding to the rows 4, 5, 6, 7 in Tab. 4, and in the map for  $E_0$  in the cells corresponding to the rows 2, 3, 4, 5. Minimum expressions are under the maps.



$$E_1 = Q_1.Q_0 + Q_1.x_0 + Q_0.\bar{x}_1.x_0 + Q_1.x_1$$

$$Q_0.\bar{x}_0$$

$$E_0 = \bar{Q}_1.Q_0 + Q_0.\bar{x}_1 + Q_1.x_1.\bar{x}_0 +$$

Fig. 5: Solution of state variables

Transitions between inputs (see Fig. 5) are marked by horizontal arrows, transitions between internal states by vertical arrows. The consequence of condition No. 1 (fundamental mode) is clear - first the inputs change at a constant internal state (shift in a row) and then only the state changes at a constant input (shift in a column). It can be seen that if the jumps along the horizontal arrows take place between the fields with the written "1", it is possible to cover them with a loop and therefore hazards can be eliminated. This favorable situation is the result of the condition No. 5. Suppose that a given state variable has the value "1" in the present and in the next internal state. The state change was triggered by an input change, but in all input temporary states the given state variable also has the value "1".

False pulses on outputs of combinational circuits can result from changes in both input and status signals. In the maps of Fig. 5, changes in state variables are indicated by vertical arrows, and in this particular case it will not be difficult to assemble a set of loops covering all these transitions. In other cases, it may be necessary to select a proper internal code such that only one state variable changes.

Although internal code selection is critical for asynchronous sequential circuits, the BM mode provides simplification. The transitions between states only occur when a unique combination of values appears on the input; this combination is not allowed further to vary (condition no. 2). Only then do the state variables change, but the inputs must stay unchanged till all internal signals have stabilized (condition 1). This eliminates any possible simultaneous change of input and status signals, which might lead to substantial hazards.

#### 4 Outputs of asynchronous sequential circuits

The output signals of the asynchronous sequential circuit are generated in the same manner as in the case of the synchronous sequential circuits. There is Mealy and Moore - outputs for Moore machine are dependent only on the internal states and for Mealy machine on states and inputs. In both cases, these are combinational functions and their design does not impose any specific problems. As an example, the output function  $y = f(Q_1, Q_2)$  for the sequential circuit of Fig. 4 is shown in Fig. 6 a). Apparently it is a Moore sequential circuit and  $y = \bar{Q}_1 Q_0$ .

	$Q_1$	$Q_0$	$y$
A	0	0	0
B	0	1	1
C	1	1	0
D	1	0	0

Fig. 6 a)

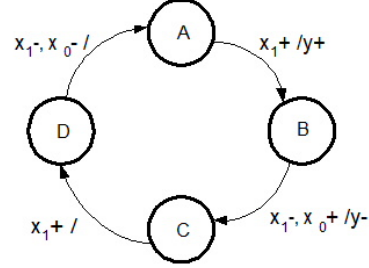


Fig. 6 b)

Fig. 6: Marking of outputs in MB symbolic

In the case of sequential BM circuits, the output signals can be written in a transition graph similarly to the input states, i.e. by **signal changes** (+ or -), after the slash. If there is no change, the space after the slash is empty - see Fig. 6 b).

This notation allows an easy check - when proceeding in the graph through a closed loop, the number of + and - signs must be the same.

## 5 Internal code selection

Changes in the internal states of BM circuits do not occur until the input state required to trigger the transition is reached. Thus, changes of internal states take place when input signals are already stable. This is a significant simplification of the issue of races and hazard, but not their complete elimination. The best situation is with internal code with changes only in **one variable**. This was achieved, for example, in Fig. 6. Note, however, that if the graph of transitions contained also a transition  $D \rightarrow B$ , it would no longer be possible to realize this transition when changing only one variable.

If transitions between stable states are allowed to take place over one or more temporary states (e.g. in BM mode this is not normally allowed, but it is not excluded), the states can be coded with an excessive number of variables. Fig. 7 a) shows the coding of 4 states by three variables. For example, two variables change from steady state **A** to steady state **C**, but only one variable changes during the transition to the transient state **C**, and only one variable changes again during the subsequent transition to the steady state **C**. Similarly, the transition between **B** and **D**. Apparently, this expands the table of transitions by temporary states and the time for the transition between stable states is then extended. On the other hand, transitions with only one variable change are possible between all states. The coding for 6 states is shown in Fig. 7 b) and for the 8 states of Fig. 7 c). In general,  $2n-1$  state variables are required for

coding  $2^n$  states with possible transitions between all states, and for coding max.  $(3/4) \cdot 2^n$  states are sufficient  $2n-2$

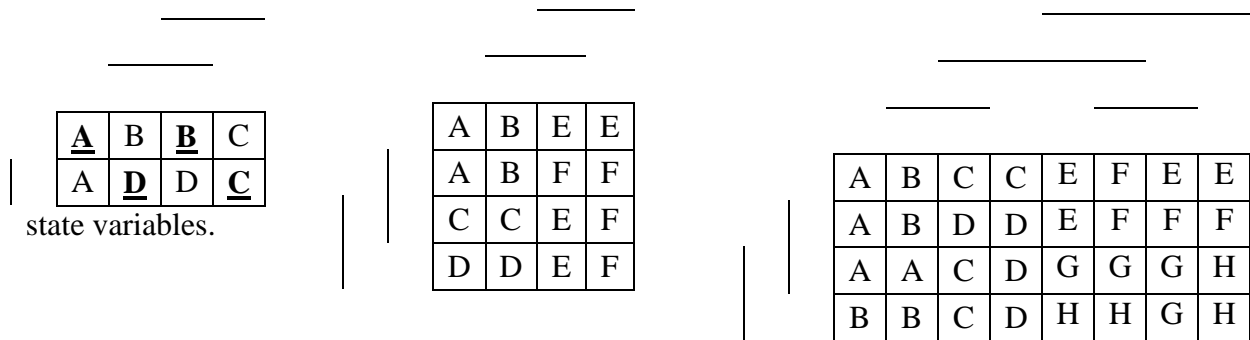


Fig. 7 a)

Fig. 7 b)

Fig. 7.c)

Fig 7: Coding with change in one variable,  
a): for 4 states, b): for 6 states, c): for 8 states

In most cases, it is not necessary to take into account the transitions between all states, so it is not necessary to introduce the full number of temporary states. Fig. 8 a) shows a graph of transitions of the circuit in Fig. 4 c), but with one transition in addition ( $C \rightarrow A$ ). States  $C1$  and  $D1$  were added and the table of transitions in Fig. 8 b) was extended. The new internal code  $Q_2, Q_1, Q_0$  guarantees changes in only one variable at a time. Fig. 8 shows only one option how to modify the transition table and the internal code - in fact there are more.

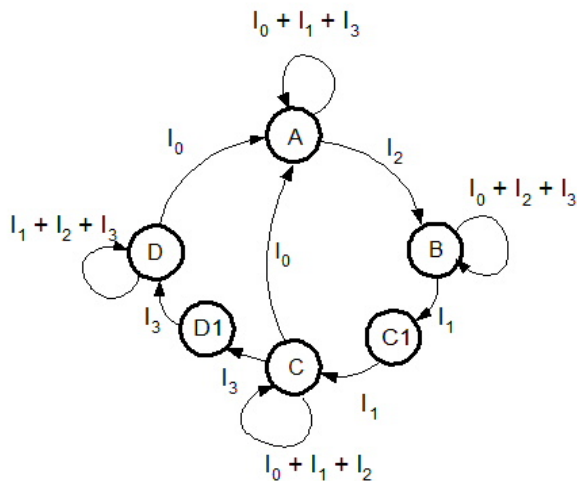


Fig. 8 a)

		x <sub>1</sub> x <sub>0</sub>			
		I <sub>0</sub>	I <sub>1</sub>	I <sub>2</sub>	I <sub>3</sub>
Q <sub>2</sub> Q <sub>1</sub> Q <sub>0</sub>		00	01	10	11
0 0 0	A	<u>A</u>	A	B	A
0 0 1	B	B	C <sub>1</sub>	<u>B</u>	B
0 1 1	C <sub>1</sub>	C <sub>1</sub>	C	C <sub>1</sub>	C <sub>1</sub>
0 1 0	C	A	<u>C</u>	C	D1
1 1 0	D <sub>1</sub>	D <sub>1</sub>	D <sub>1</sub>	D <sub>1</sub>	D
1 0 0	D	A	D	D	<u>D</u>

Fig. 8 b)

Fig. 8: Internal code with change in one variable

The importance of the appropriate choice of internal code also follows from the fact that internal codes with a change in only one variable allow the construction of output circuits that generate outputs without **false pulses**.

6 Initial state setting

After switching the power supply ON, the sequential circuit is in an unknown state and it is always necessary to ensure its initial state. This does not have to be the state 00 ... 0, so it is not literally "zeroing", but "**resetting**". The signal for invoking the initial state is commonly referred to as "Reset". While in the synchronous sequential circuits the priority inputs (R, S) are commonly used for setting or resetting individual flip-flops, in an asynchronous circuit the status signals must be influenced directly. Fig. 9 shows one solution. With a combination of X=0 and Y=0, 0 is forced to the output, with X=arbitrary and Y=1, 1 is forced to the output, and with X=1 and Y=0, the circuit operates normally. The initial state is usually set **automatically** after the power is turned ON.

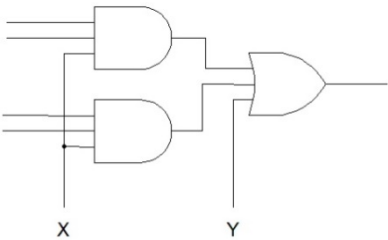


Fig. 9: Arrangement for setting and resetting

7 C-element

The function of a type C asynchronous trigger - also "**C-element**" - is defined as follows:

*If both inputs have value '0', the output is also '0'. If both inputs have value '1', the output is also '1'. If the values on the inputs are different, the output will not change.*

The circuit is evidently **asynchronous**, because there of the circuit is shown by a in the BM symbolism in Fig.

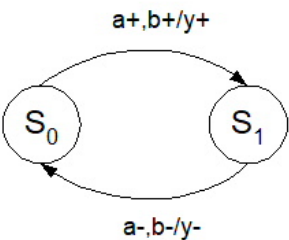


Fig. 10 a)

	$\bar{a}.\bar{b}$	$\bar{a}.b$	$a.\bar{b}$	$a.b$
$S_0$	<u><math>S_0</math></u>	$S_0$	$S_0$	$S_1$
$S_1$	$S_0$	$S_1$	$S_1$	<u><math>S_1</math></u>

Fig. 10.b)

**sequential**, because with the previous state. It is is no clock input. The function graph and a table of transitions 10 a) and 10 b). The input variables are *a*, *b* and the output variable *y*.

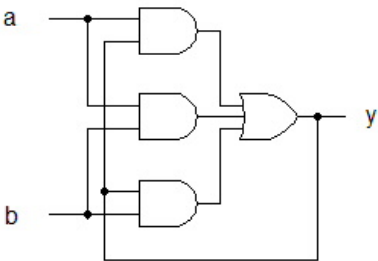


Fig. 10.c)

Fig. 10: C-element

a) Graph of transitions in BM notation,      b) Table of transitions,      c) Resulting diagram

We encode two internal states with one variable  $S$ . The output variable  $y$  is identical to the state variable, which is common in triggers. Thus,  $S_0$  is coded as  $\bar{y}$  and  $S_1$  as  $y$ . The coding of the input states is shown in table 10 b). After substituting the codes into the relationships for the transitions, we get:

transitions	output $y$	
$\bar{y} \cdot (\bar{a} \cdot \bar{b} + \bar{a} \cdot b + a \cdot \bar{b}) \rightarrow \bar{y}$	$\rightarrow \bar{y}$	0
$\bar{y} \cdot a \cdot b \rightarrow y$	$\rightarrow y$	1
$y \cdot (\bar{a} \cdot b + a \cdot \bar{b} + a \cdot b) \rightarrow y$	$\rightarrow y$	1
$y \cdot \bar{a} \cdot \bar{b} \rightarrow \bar{y}$	$\rightarrow \bar{y}$	0

After selecting the rows for  $y = 1$  we get:

$$y = \bar{y} \cdot a \cdot b + y \cdot (\bar{a} \cdot b + a \cdot \bar{b} + a \cdot b) = a \cdot b + y \cdot a + y \cdot b$$

The resulting circuit of a **C-element** is shown in Fig. 10 c).

The C-trigger or C-element is the basis of many asynchronous sequential circuits. Therefore, it is important to minimize its price (expressed in the number of transistors). The circuit in Fig. 10.c) is fully functional, but its price corresponds to the number 18. A much more economical solution with a price of **only 8** is shown in Fig. 11 a), the schematic symbol is in Fig. 11 b).

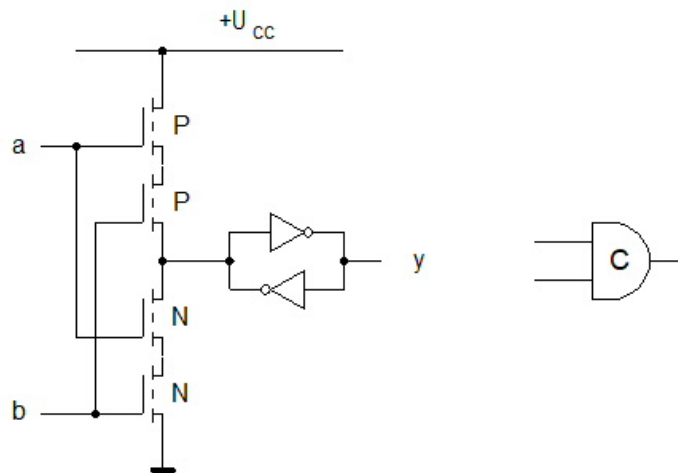


Fig. 11 a): Practical connection of C-element      Fig. 11 b): Schematic symbol

The core is the simplest trigger consisting of two inverters, which is controlled by switches with NMOS and PMOS transistors. If there are L levels on both inputs, both PMOS transistors lead and then  $y=0$ . At level H at both inputs, both NMOS transistors lead and then



$y=1$ . At different levels, always one of the PMOS and NMOS transistors does not lead and the state of the output is not affected. The C-element in this form is **very often** used and is the main part of asynchronous sequential circuits.

The C-element can be very easily **modified** by adding more transistors P or N. For example, by adding one NMOS transistor with input  $c$  in series with two already existing, we achieve that  $a=b=1$  is not enough to toggle  $y$  to 1, but also it must be  $c=1$ . However,  $c$  has no influence on toggling  $y$  to 0. In MB symbolic we can write  $a+$ ,  $b+$ ,  $c+$  /  $y+$ , but  $a-$ ,  $b-$  /  $y-$ . Figure 12 a) shows the connection and Fig. 12 b) the corresponding symbol. If, on the other hand, instead of an NMOS, a PMOS transistor were added in series with two already existing PMOS, the transition of  $y$  to 0 would be conditioned by the state 0 at input  $c$ . For a transition to 1, input  $c$  would have no effect. We can write  $a-$ ,  $b-$ ,  $c-$  /  $y-$ ,  $a+$ ,  $b+$  /  $y+$ . Fig. 12.c shows the corresponding schematic symbol.

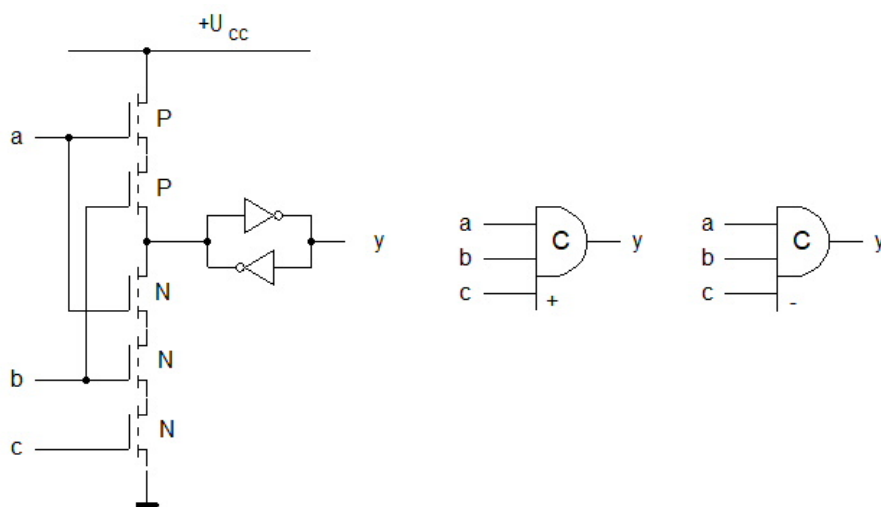


Fig. 12 a): Connection

Fig. 12.b): Input  $c$  conditioning  $y+$ Fig. 12.c): Input  $c$  conditioning  $y-$ 

Fig. 12: Additional input for C-element

There are **many other modifications**, some of which are shown in Figure 13. As a practical example, derive the BM notation of their functions:

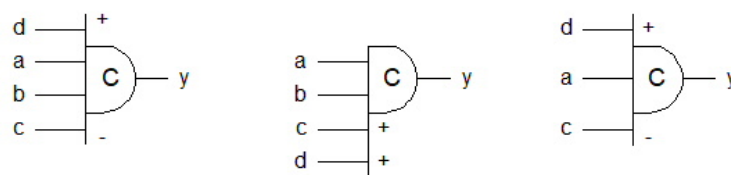
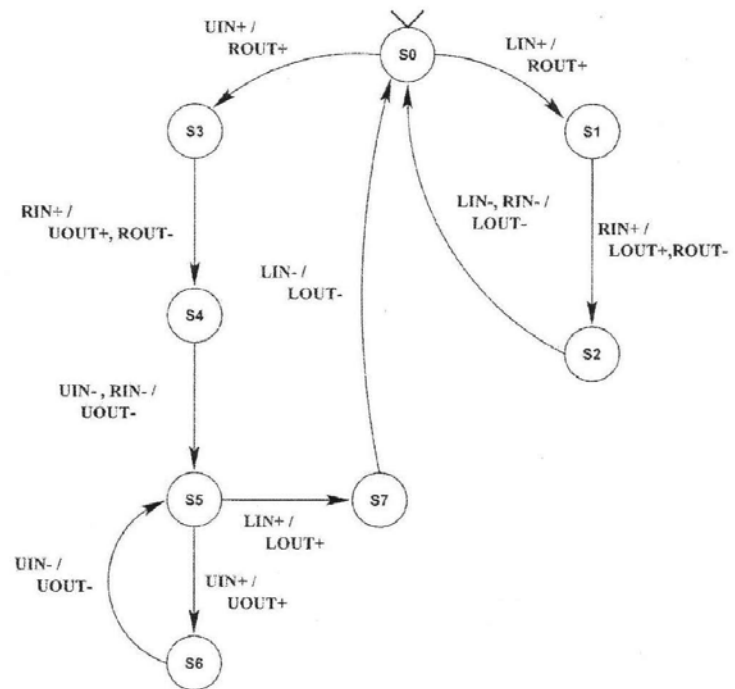


Fig. 13: Further modifications of C-element

**Attachment**

An example of a more complex BM sequential circuit. A large number of unspecified states makes it easy to fill in the transition table. In practice, this is common. Stable states are indicated by a circle in the table.



Inputs: LIN, RIN, UIN;

Outputs: LOUT, ROUT, UOUT;

[illegible]