

Synchronous sequential circuits

- 1 Transition and output functions
- 2 Description using graphs
- 3 Description using tables
- 4 Description using symbolic representations of transitions
- 5 Description in a programming language
- 6 Circuit realization of finite state machine
 - 6.1 States coding
 - 6.2 Flip-flops selection
 - 6.3 Design of excitation functions
 - 6. 4 Notes on codes assignment
- 7 Initial state setting
- 8 Timing of signals
- 9 Clock pulses
- 10 Delay of output signals
- 11 Connected sequential circuits

Synchronous sequential circuits

The basic feature of sequential circuits, which differs them from combinational circuits, is that the sequential circuits can react differently to one and the same combination of values of input signals. The output signals of the sequential circuits are thus dependent not only on the input signals at a given moment, but also on the **previous** input signals. In other words - the sequential circuit has an **internal memory**.

1 Transition and output functions

The abstract model of a sequential circuit is a **finite state machine** - **FSM**. It has a finite number of input states, output states and internal states. A state is a combination of signal values. The finite state machine as a mathematical construction is implemented by a sequential circuit. In technical practice we deal with dual-state signals - values 0 or 1, which is a finite number. Therefore, the number of possible combinations of values of these signals is also finite. We mark each of the combinations with a symbol - most frequently a letter for simple writing.

Fig. 1 shows a sequential circuit with input signals x_i , output signals y_i and internal signals Q_i . The input state is generally marked as I , the output state as O and the internal state as S . Individual specific combinations of values of input signals will be marked as I_0, I_1 , etc.. Similarly, combinations of values of output signals will be marked as O_0, O_1 , etc., and internal signals as S_0, S_1 , etc..

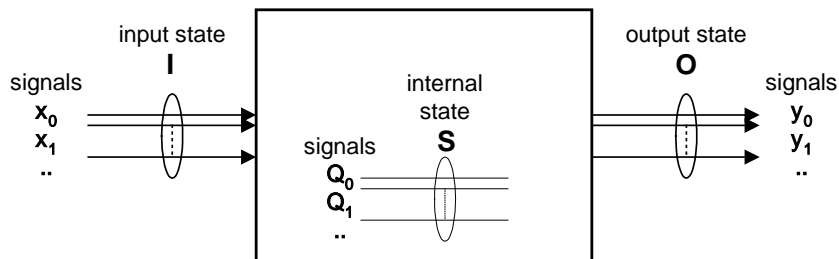


Fig. 1: Signals and states in a sequential circuit

We can imagine the function of the sequential circuit as follows in this **example**:

Suppose the present internal state is S_0 . With input state I_3 , the next internal state will change to S_5 . If the input state I_3 holds, the internal state changes to S_2 , ... etc.. Evidently, the sequence of internal states depends on the input sequence and also on the specific feature of the circuit called "**transition function**". The sequence of the output states will be discussed later.

Transition function is the dependence of the next internal state on the present internal state and on the input state. It is expressed as:

$$S^{t+1} = f(S^t, I^t),$$

where the symbol S^{t+1} denotes the next internal state, S^t denotes the present internal state, I^t denotes the present input state.

The output state of an FSM can be generated in several ways. Each of them is defined by a corresponding **output function**. In the most general case, the output state is a function of the input state and the internal state:

$$O^t = g(S^t, I^t)$$

Identical indices t for all variables mean that the output state is given by the internal state and the input state at the same time (i.e. present time), and thus it can be implemented by a combinational function. A FSM with an output function defined in this way is called a **Mealy's FSM**.

Another possibility of output state generation is defined by the output function, which depends only on the internal state:

$$O^t = g(S^t)$$

Apparently this is again a combinational function. A FSM with an output function defined in this way is called a **Moore's FSM**. At first glance, it might seem that Moore's FSM is simpler. In fact it is not, as we will see.

On the contrary, the dependence of the output state only on the current input state is not permissible, as it would no longer need an internal state, and therefore it will no more be a sequential system - it would be a combinational system.

A special case is an **autonomous FSM**. Its transition function is:

$$S^{t+1} = f(S^t)$$

The next internal state does not depend on the input state, just on the present internal state. The machine does without input signals completely. This is the case, for example, of the counters, which are very common digital systems.

It is obvious that there must be a time delay between the present and the next internal state. This may be due to delays in the internal circuits that generate status signals. The machine thus changes its internal state (with a small delay) after changing the input state. Sequential circuits that respond in this way are called **asynchronous**. The delay can also be introduced so that the status signals are generated by synchronous triggers. Their **synchronizing pulses** (clock pulses) then indicate the moment of change of the internal states. The sequential circuits that react in this way are called **synchronous**. For them, the timing of internal signals can be managed much easier.

For the sake of brevity, the word "state" meaning "internal state", the word "input" meaning "input state" and the word "output" meaning "output state" will be used in the following text.

To design a sequential circuit, it is first necessary to describe its operation in a suitable way. The most natural way it would seem to enumerate all the input sequences and their

corresponding output sequences. However, it must be noted that although the FSM has a finite number of input, output and internal states, it can easily handle sequences of **infinite length**. There are generally no restrictions on the length of the input sequence. Such a description is therefore completely impractical and applicable perhaps in a few exceptional cases. Applicable are the following methods of description:

- Description using graphs and flowcharts.
- Description using tables.
- Description by a set of symbolic representations of transitions.
- Description in some programming language.

2 Description using graphs

A **transition diagram** or also **state diagram** is a directed graph used to represent the transition and output function in a sequential system. Each state is represented by a **node** and each transition by an **arc**. An arc from node S_k to node S_j and labeled I_p specifies that, for a present state S_k and input I_p , the next state is S_j . The node corresponding to the initial state is frequently labeled S_0 and marked by a thick circle.

The outputs of the Mealy FSM are marked at the individual arcs (transitions), e.g. I_0/O_2 of the graph, because the output depends on the state **and** the input. In the Moore FSM, the outputs are marked at the nodes of the graph, because the output depends only on the state. Fig. 2 illustrates the case of the Moore FSM, Fig. 3 the case of the Mealy FSM.

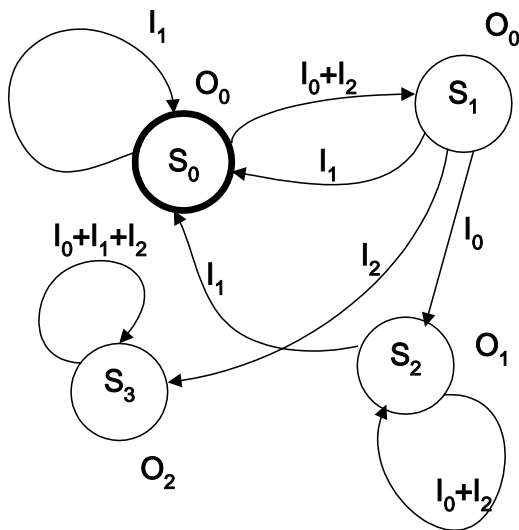


Fig. 2 : State diagram - Moore FSM

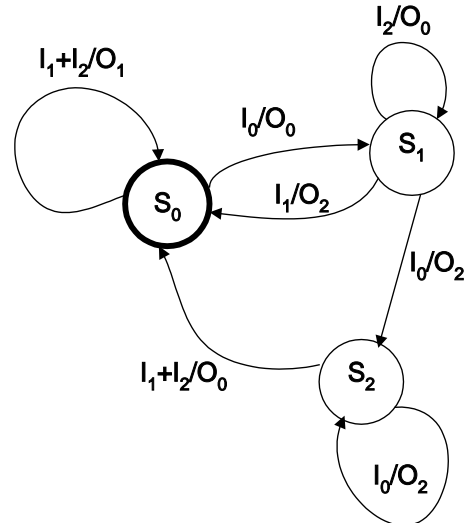


Fig. 3: State diagram - Mealy FSM

The figures illustrate various variants for the shape of graphs. Inputs joined by a "+" sign on some arcs mean that two or more input states cause the same transition to the given internal state. For example, in Fig. 2, the output O_0 is output in the state S_1 , and with the input I_2 the state switches to S_3 , in which the output O_2 is valid. In Fig. 3, in the state S_1 , with the input I_0 is the output O_2 valid, with the input I_1 is the output O_2 valid, and with the input I_2 is the output O_0 valid. From the state S_3 in Fig. 2 there is no transition to another state and, therefore,

S_3 is a **terminal** state. The FSM can be moved from it only by an external intervention - in real systems by the action of other signals or by switching off the power supply.

3 Description using tables

The **transition table** is constructed so that in the first column there are present states, in the next columns under the respective inputs there are the next states. The transition table is **equally** constructed for the Moore FSM as for the Mealy FSM.

The **output table** of the Moore FSM has states in the first column and outputs in the second. For Mealy's FSM, it has states in the first column and outputs in the other columns below the respective inputs.

Tables 1 and 2 belong to the graphs of Fig. 2, Tables 3 and 4 belong to the graphs of Fig. 3.

	I_0	I_1	I_2
S_0	S_1	S_0	S_1
S_1	S_2	S_0	S_3
S_2	S_2	S_0	S_2
S_3	S_3	S_3	S_3

S_0	O_0
S_1	O_0
S_2	O_1
S_3	O_2

Tab. 1: Transition table of Moore FSM

Tab. 2: Output table of Moore FSM

	I_0	I_1	I_2
S_0	S_1	S_0	S_0
S_1	S_2	S_0	S_1
S_2	S_2	S_0	S_0

	I_0	I_1	I_2
S_0	O_0	O_1	O_1
S_1	O_2	O_2	O_0
S_2	O_2	O_0	O_0

Tab. 3: Transition table of Mealy FSM

Tab. 4: Output table of Mealy FSM

Very often is used the concentrated shape of both tables - a **state table**. The entries of the state table contain information about the states and outputs - e.g. S/O or S,O - compare the tables 3 and 4.

4 Description using symbolic representations of transitions

The symbol " \rightarrow " will be used for the symbolic representation of the transition - hereinafter abbreviated **SRT** - from the state at time t to the state at time $t+1$, the symbol " α " will be used for the simultaneous occurrence of state S and input I . The arrow " \rightarrow " symbolizes the effect of the CLK pulse in a synchronous sequential system.

The **transition function** is then described by SRT in the form:

$$S_i \alpha I_j \rightarrow S_k$$

Specifically for the Mealy FSM according to the graph of Fig. 3 or according to Tab. 3:

$$\begin{aligned} S_0 \alpha I_0 &\rightarrow S_1 \\ S_0 \alpha (I_1 + I_2) &\rightarrow S_0 \\ S_1 \alpha I_0 &\rightarrow S_2 \\ S_1 \alpha I_1 &\rightarrow S_0 \\ S_1 \alpha I_2 &\rightarrow S_1 \\ S_2 \alpha I_0 &\rightarrow S_2 \\ S_2 \alpha (I_1 + I_2) &\rightarrow S_0 \end{aligned}$$

The same formal notation applies to Moore's FSM.

The **output function** is a combinational function and for Mealy's FSM it is written by the SRT system in the form:

$$S_i \alpha I_j : O_k$$

The ":" symbol here indicates the immediate creation of the output. For a Moore FSM, the output function has the form:

$$S_i : O_k$$

Specifically for the Mealy FSM of Fig. 3, the output function is described by the SRT system:

$$\begin{aligned} S_0 \alpha I_0 : O_0 \\ S_0 \alpha (I_1 + I_2) : O_1 \\ S_1 \alpha I_0 : O_2 \\ S_1 \alpha I_1 : O_2 \\ \dots\dots\dots \\ \dots\dots\dots \\ \text{etc.} \end{aligned}$$

For the Moore FSM of Fig. 2, the output function is:

$$\begin{aligned} S_0 : O_0 \\ S_1 : O_0 \\ S_2 : O_1 \\ S_3 : O_2 \end{aligned}$$

5 Description in a programming language

This method of description depends on the language in which the task is to be programmed. "C" is a suitable programming language for tasks performed by computers, microcontrollers,

etc., while **VHDL**, **VERILOG**, or other languages are suitable for implementation by programmable logic circuits FPGA, CPLD.

6 Circuit implementation of finite state machines

The FSM as a mathematical construction is implemented by a sequential circuit. Fig. 4 shows a synchronous sequential circuit. The internal state is determined by combining the values of signals Q_i , obtained from the memory elements - triggers. In case of sequential circuits, these must be **edge controlled** (see text below), thus we will further denote them **FF** (flip-flops). If the type of flip-flops is known (D, T, JK, ..., synchronous, edge controlled), then it is possible for each transition from the present state to the next state to determine the required values of **excitation signals** of FFs. These are generally denoted as E_i - in a specific case it is possible to substitute D_i , T_i , J_i , K_i , etc. Since the next state is determined by the present state and the input at time t , the following will apply:

$$E_i = e_i(Q_0, Q_1, \dots, x_0, x_1, \dots)$$

where functions e_i are called **excitation functions**. All the variables of the excitation functions act simultaneously, i.e. combinational circuits can be used to generate them. The path of the excitation signals is symbolically indicated in the Fig. 4 by a thick solid arrow.

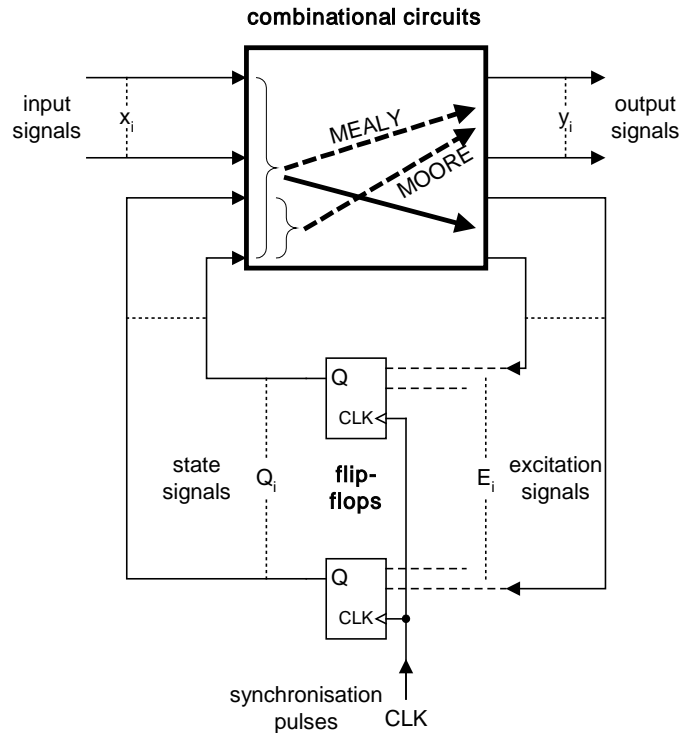


Fig. 4: Signal paths in a synchronous sequential circuit

The FFs must be **edge-controlled**. In case of level-controlled triggers, for the duration of the CLK pulse the following loop would be created:

$$\text{outputs } E_i \rightarrow \text{triggers} \rightarrow \text{comb. circuits}$$

During this time, the loop behaves as an **asynchronous** sequential circuit which, however, must be solved completely differently than the intended synchronous circuit.

The outputs of the Moore sequential circuit depend only on its states, i.e. the output variables y_i will be combinational functions only of the variables Q_i , as indicated by a thick dashed arrow in the Fig. 4. The outputs of the Mealy sequential circuit depend on both its states and inputs, and thus the output variables y_i will be combinational functions of the variables Q_i and x_i , as indicated by the second thick dashed arrow in the figure.

The whole complex combinational circuits can be divided into two sub-components: one for creating excitation signals, the other for producing output signals. Dividing into two parts simplifies the solution of each of them.

Fig. 5 shows such a division of the combinational circuits. The output circuits can take three different forms. Case 1 (in a circle) shows the Mealy version, case 2 concerns the Moore version, and case 3 is again a variant in which the combinational circuits can be left out in special cases and the output signals are delivered directly from the flip-flop outputs. For example, synchronous counters are this case. There are almost always **hazards** in the output circuits and false pulses can then occur at the outputs. If the combinational circuits are missing, there are no false impulses. However, a solution without output combinational circuits may require an **increased number** of flip-flops. It should be noted that variant 1, 2, or 3 always requires a different transition and output function and therefore it is not possible to use two or three of them at the same time - Figure 5 is just a summary of the options.

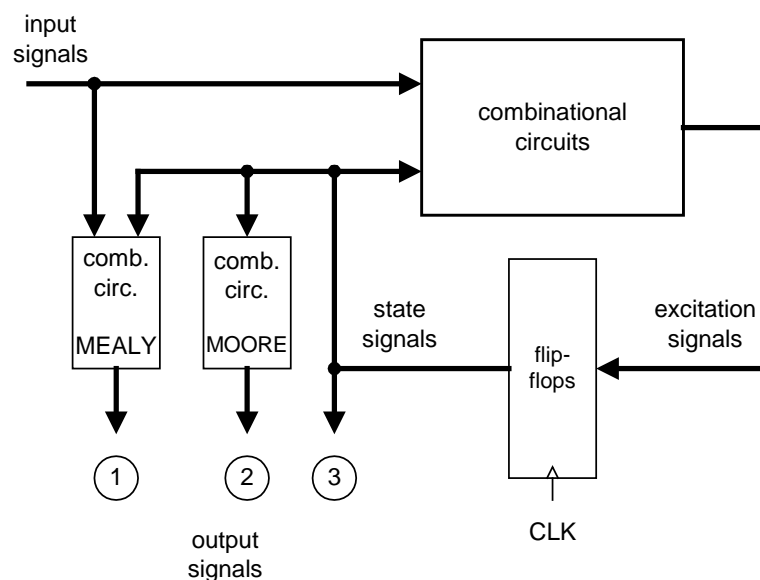


Fig. 5: Output circuits variants

6.1 States coding

Each state corresponds to a single combination of signal values. Assignment of states to combinations of signal values must be performed for input states, internal states and output states. We get the **input code**, **internal code** and **output code**. For synchronous sequential circuits, the circuits can be properly designed to achieve the desired function with any coding. Code assignment, however, typically has an influence on the **complexity** of the circuits. The minimization of the entire sequential circuit therefore requires that the choice of codes be taken into account. It is a very complex task, and with computers at least partially solvable.

For further design, it is necessary to determine the **number of signals** needed to encode states. The following applies to n states and k signals:

$$2^{k-1} < n \leq 2^k$$

Here, k denotes the minimum number of signals. Sometimes it is appropriate to introduce more than the minimum number of signals and then there will be unused combinations of signal values and a corresponding number of **unused states**. Unused states exist even if n is not a power of two. For example, with the number of states 5, the minimum number of signals is 3, but three signals can encode $2^3 = 8$ states. Therefore, there will be 3 unused states. The unused states need not be considered in the design, but they can cause complications - see text below.

For correct **code assignment**, two conditions must be met: only one combination of values may be assigned to each state, and only one state may be assigned to each combination of values. The task is seemingly trivial, but with a large number of conditions, error is not ruled out. A simple method, eliminating errors, is to convert the index of the status symbol to a binary number and assign the weights of the individual bits to the individual signals. The signals order must be selected and kept.

For example, 3 signals are required to encode five input states I_0, I_1, I_2, I_3, I_4 . Let us denote them x_0, x_1, x_2 and arrange them in the order (MSB) $x_2 x_1 x_0$. Then the index at I_0 , i.e. the decimal number 0, corresponds to the binary number 000, i.e. the input state I_0 will be coded by a combination of values $x_2 = 0, x_1 = 0, x_0 = 0$, shortly $\overline{x_2} \overline{x_1} \overline{x_0}$. We proceed similarly for other conditions. Overall, this code can be summarized in Table 5.

I_0	$\overline{x_2} \overline{x_1} \overline{x_0}$
I_1	$\overline{x_2} \overline{x_1} x_0$
I_2	$\overline{x_2} x_1 \overline{x_0}$
I_3	$\overline{x_2} x_1 x_0$
I_4	$x_2 \overline{x_1} \overline{x_0}$

Tab. 5: Example of input code

The remaining states I_5, I_6, I_7 will not be used and therefore do not have to be coded.

Although the internal code is very rarely specified in the task, the input and output codes often are specified, e.g. as follows: "... if the combination of input signals $A=1$, $B=0$, $C=0$ is followed by a combination $A=0$, $B=0$, $C=1$, then the output combination will be $Y=0$, $Z=1$, and further if

After setting the codes, it is possible to overwrite the SRT set (symbolic representation of transitions), or alternatively the transition and output tables, so that the symbols for inputs, states and outputs are replaced by combinations of signal values according to the selected codes. Let us continue with the solution of the Mealy FSM, described by the graph in Fig. 3. It has three input states I_0 , I_1 , I_2 and therefore must have two input signals, which we denote as x_1 , x_0 . Furthermore, it has three internal states S_0 , S_1 , S_2 , and thus there will be two state signals Q_1 , Q_0 . Input and internal codes are in Tab. 6 and 7.

input code	
state	combination x values
I_0	$\overline{x_1} \overline{x_0}$
I_1	$\overline{x_1} x_0$
I_2	$x_1 \overline{x_0}$

Tab 6: Input code table

internal code	
state	combination Q values
S_0	$\overline{Q_1} \overline{Q_0}$
S_1	$\overline{Q_1} Q_0$
S_2	$Q_1 \overline{Q_0}$

Tab. 7: Internal code table

The following are the SRTs that have already been derived in the previous text. Status and input symbols in Fig. 6 are replaced by status and input codes in Fig. 7. The symbol " α ", expressing the simultaneous existence of certain state and input, is replaced by a logic product (AND). This way we get the products of status and input signals. The alternative occurrence of inputs is expressed by a logical sum (OR).

$$\begin{aligned}
S_0 &\alpha I_0 \rightarrow S_1 \\
S_0 &\alpha (I_1 + I_2) \rightarrow S_0 \\
S_1 &\alpha I_0 \rightarrow S_2 \\
S_1 &\alpha I_1 \rightarrow S_0 \\
S_1 &\alpha I_2 \rightarrow S_1 \\
S_2 &\alpha I_0 \rightarrow S_2 \\
S_2 &\alpha (I_1 + I_2) \rightarrow S_0
\end{aligned}$$

Fig. 6: Symbolic representations of transitions

present state and input	next state
$\overline{Q_1} \overline{Q_0} \cdot \overline{x_1} \overline{x_0}$	$\rightarrow \overline{Q_1} \overline{Q_0}$
$\overline{Q_1} \overline{Q_0} \cdot (\overline{x_1} x_0 + x_1 \overline{x_0})$	$\rightarrow \overline{Q_1} \overline{Q_0}$
$\overline{Q_1} Q_0 \cdot \overline{x_1} \overline{x_0}$	$\rightarrow Q_1 \overline{Q_0}$
$\overline{Q_1} Q_0 \cdot \overline{x_1} x_0$	$\rightarrow \overline{Q_1} \overline{Q_0}$
$\overline{Q_1} Q_0 \cdot x_1 \overline{x_0}$	$\rightarrow \overline{Q_1} Q_0$
$Q_1 \overline{Q_0} \cdot \overline{x_1} \overline{x_0}$	$\rightarrow Q_1 \overline{Q_0}$
$Q_1 \overline{Q_0} \cdot (\overline{x_1} x_0 + x_1 \overline{x_0})$	$\rightarrow \overline{Q_1} \overline{Q_0}$

Fig. 7: SRT in coded form

For example, the second line in Fig. 6 indicates that the concurrent state of S_0 and inputs I_1 or I_2 passes to state S_0 . The state S_0 is coded as $\overline{Q_1} \overline{Q_0}$, input I_1 as $\overline{x_1} x_0$ and input I_2 as $x_1 \overline{x_0}$. The next state S_0 is coded as $\overline{Q_1} \overline{Q_0}$.

6.2 Flip-flops selection

It is also necessary to design the type of flip-flops. Again, the desired functions of a synchronous sequential circuit can be achieved with any **synchronous, edge-controlled** flip-flops. It is even possible to select a different type of FF for each internal signal, but all with the same clock control and timing. However, the choice of flip-flops can affect the complexity of combinational circuits. In general, it is possible to follow this rough guide:

- with FF type **D**, the derivation of excitation functions is the easiest,
- with FF type **T**, the sequential circuits, whose function is close to the function of the counter, come out simpler than with D-FFs.
- with FF type **JK**, the excitation functions are the simplest, but they are two per each FF.

After selecting the FFs, it is now possible to derive the excitation functions that are combinational functions. Each of the excitation functions depends in general on all input signals of the sequential circuit and output signals of flip-flops.

6.3 Design of excitation functions

For each transition between two internal states, it is possible to determine the changes at the output of each FF (there are only four possibilities: $0 \rightarrow 0$, $0 \rightarrow 1$, $1 \rightarrow 0$, $1 \rightarrow 1$). The necessary condition is the knowledge of the input and internal code. Each required change at the output of the FF determines the required signal(s) at the FF's input(s). The rules for each type of FF, i.e. the requirements for the excitation functions, can be easily inferred from the FF behavior. In Tab. 8 are the requirements summarized.

$Q^t \rightarrow Q^{t+1}$	D^t	T^t	J^t	K^t
$0 \rightarrow 0$	0	0	0	-
$0 \rightarrow 1$	1	1	1	-
$1 \rightarrow 0$	0	1	-	1
$1 \rightarrow 1$	1	0	-	0

Tab. 8: States at FF inputs for required transitions

For FF type D, the column Q^{t+1} can be simply rewritten into the excitation function D^t . For FF type T, input T^t has a value of 1 whenever Q is to change. For FF type JK, the excitation functions J and K have half of the values undetermined, allowing greater simplification of circuits.

With selected types of flip-flops, the excitation functions of FFs can already be read directly from the SRT set in coded form. The excitation functions will be expressed as a **sum of products**, therefore substantial will only be those combinations of variables in which the excitation function has a **value of 1**. It is a similar task as the derivation of the expression from the truth table (see chapter Combinational circuits).

Let's continue with the solution of the sequential circuit from the previous text. Assume that the flip-flops are of type D. From the SRT in coded form (Fig. 7), select the rows where Q_1 **changes to 1 or remains 1**, i.e. the transitions $\bar{Q} \rightarrow Q_1$ or $Q_1 \rightarrow Q_1$. These transitions from present to next state are caused by $D_1=1$. This happens in the row 3 and 6 of the Fig. 7. In the row 3, the combination of internal variables and input variables is $\bar{Q}_1 Q_0 \cdot \bar{x}_1 \bar{x}_0$, in the row 6 it is $Q_1 \bar{Q}_0 \cdot \bar{x}_1 \bar{x}_0$. We get:

$$D_1 = \bar{Q}_1 Q_0 \cdot \bar{x}_1 \bar{x}_0 + Q_1 \bar{Q}_0 \cdot \bar{x}_1 \bar{x}_0$$

The same procedure is applied to D_0 . We get:

$$D_0 = \bar{Q}_1 \bar{Q}_0 \cdot \bar{x}_1 \bar{x}_0 + \bar{Q}_1 Q_0 \cdot \bar{x}_1 \bar{x}_0$$

If we select T-type flip-flops, it will be necessary to select from the SRT set those rows in which the Q output **changes**. This happens in the row 3 and 7 of the Fig. 7. We get:

$$\begin{aligned} T_1 &= \bar{Q}_1 Q_0 \cdot \bar{x}_1 \bar{x}_0 + Q_1 \bar{Q}_0 \cdot (\bar{x}_1 \bar{x}_0 + x_1 \bar{x}_0) \\ T_0 &= \bar{Q}_1 \bar{Q}_0 \cdot \bar{x}_1 \bar{x}_0 + \bar{Q}_1 Q_0 \cdot \bar{x}_1 \bar{x}_0 + \bar{Q}_1 Q_0 \cdot \bar{x}_1 x_0 = \dots = \bar{Q}_1 \bar{x}_1 \bar{x}_0 + \bar{Q}_1 Q_0 \bar{x}_1 \end{aligned}$$

For the case of JK flip-flops it is necessary to find two excitation functions. Function J includes the rows in which there is a change $\bar{Q} \rightarrow Q$, function K includes the rows in which there is a change $Q \rightarrow \bar{Q}$. Due to the **unspecified states** in Tab. 8, other rows can (and need not be) included in J and K functions. Optional terms are enclosed in square brackets.

$$\begin{aligned}
J_1 &= \overline{Q_1} Q_0 \cdot \overline{x_1} \overline{x_0} + [Q_1 \overline{Q_0} \cdot \overline{x_1} \overline{x_0} + Q_1 \overline{Q_0} \cdot (\overline{x_1} x_0 + x_1 \overline{x_0})] = \dots = \overline{Q_1} Q_0 \cdot \overline{x_1} \overline{x_0} \\
K_1 &= Q_1 \overline{Q_0} \cdot (\overline{x_1} x_0 + x_1 \overline{x_0}) + [\overline{Q_1} \overline{Q_0} \cdot \overline{x_1} \overline{x_0} + \overline{Q_1} \overline{Q_0} \cdot (\overline{x_1} x_0 + x_1 \overline{x_0}) + \overline{Q_1} Q_0 \cdot \overline{x_1} \overline{x_0} + \\
&\quad + \overline{Q_1} Q_0 \cdot \overline{x_1} x_0 + \overline{Q_1} Q_0 \cdot x_1 \overline{x_0}] = \dots = \overline{Q_0} x_1 \overline{x_0} + \overline{Q_0} \overline{x_1} x_0
\end{aligned}$$

Unspecified states apparently contributed to the simplification of expressions and, therefore, also circuits. It would now be possible to select the most advantageous excitation functions and thus the flip-flop assembly. Probably JK would be preferred.

6. 4 Notes on codes assignment

As mentioned above, the choice of codes is important for minimizing sequential circuits. Input and output codes are usually given in the task description and it is not necessary to deal with their choice. However, internal states are not part of the assignment, as they are not important from the client's point of view and are not visible. Their number and coding - "internal code" - are determined only during the design. Although any internal code can be used in terms of the function of the synchronous sequential circuit, the choice of code is essential in terms of the complexity of the combinational circuits in the sequential circuit. Next, we will deal with three typical approaches to assigning the internal code:

- random assignment
- quasi-optimal assignment
- assignment "1 of N"

Random assignment was used in paragraph 6.1, although this was not explicitly stated. Randomness lies in how the states have been named at the beginning of the design, i.e. what indexes have been assigned to them. This could be done in a variety of ways and just one was chosen. Random assignment is simple and functional, but the resulting complexity of circuits is just as random.

Quasi-optimal assignment is one that increases the **probability** of circuit simplification. Though it does not guarantee the best result, it very probably leads to a better result than the random assignment. There are three basic rules that assign **adjacent codes** to two or more states. Codes differing in exactly **one bit** are considered adjacent. Adjacent codes should be assigned to the following states:

- States that have the same next state with the same input.
- States following one and the same state with different inputs.
- States in which the outputs are the same with the same input.

For **assignment 1 of N**, the internal code is determined so that only one internal variable has a value of 1 at a time and all others have a value of 0. The code is known as "**one-hot**". For coding n states, however, we now need exactly n state signals and n flip-flops, and thus n excitation functions. This obvious disadvantage is balanced by much simpler expressions for excitation functions. The method is sometimes suitable for **programmable logic circuits** (FPGA circuits), where a large number of flip-flops are available (typically 2 in one programmable cell) and at the same time limited means for realizing the excitation functions of the respective flip-flops. Design according to this method is very fast.

Optimal code is considered to be such that provides the simplest combinational circuits. This task has no other solution than to try all the variants, which is practically impossible for an enormous number of different codes even with a relatively small number of states. For example, with 4 flip-flops and 16 states (still a simple task), there are approximately 2.1×10^{12} different codes (!). It is certainly not possible to try them. Thus, quasi-optimal assignment makes sense.

7 Initial state setting

When the power supply is switched on, the individual flip-flops get into **random states**. This is undesirable in most cases. For some sequential circuits, there are undefined internal states (when the number of states used is not equal to the power of two). These can form a closed group, which is not connected to the used states by any transition. After the supply voltage rises, the sequential circuit may enter one of these states, preventing its proper operation. This problem is solved by setting all FFs to a suitable initial state immediately after the supply voltages are switched ON. Very often it is sufficient to **reset** all FFs, but the initial state may be in many cases other than 00...0. In the graphic representation, the initial state is indicated by a thicker line (circle). The setting of the initial state of the FF was described in the chapter Triggers and metastability.

The possibility of setting arbitrary state is also valuable for **debugging and diagnostics** of the sequential circuit. Suppose we want to check all transitions between all states. To do this, it is necessary to set the selected state, e.g. S_k and test the transition with the input I_0 - this will get the circuit to a new internal state. In order to check the other transitions from S_k , it is necessary to transfer the circuit back to this state. In this way, all transitions can be checked gradually. Without the possibility of setting arbitrary state, a complete check would not be practically possible.

The pulse for setting the initial state can be generated **automatically** when the supply voltage starts. A simple solution is shown in Figure 8.

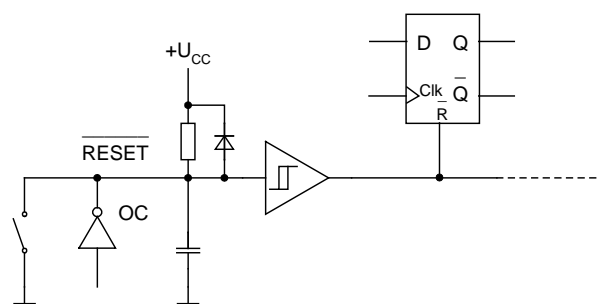


Fig. 8: Automatic resetting after the start of supply voltage

After the start of the power supply, the capacitor is slowly charged and thus the state 0 lasts at the output of the hysteresis gate for some time. After charging the capacitor to the upper threshold level, the hysteresis gate switches to the state 1 and resetting ends. The diode allows the capacitor to be discharged quickly even during short-term power failures, so that the circuit can immediately generate a new reset pulse. The reset signal can also be initiated by a

manual reset button, an open collector circuit, etc. Manufacturers offer various integrated circuits for "**power monitoring**", some of them very sophisticated.

8 Timing of signals

The first task will be to determine the **maximum frequency of clock pulses**. To derive it, we assume that the **input** signals do not change. Fig. 9 on the left shows the situation; figure on the right shows the time diagrams. The flip-flops (FF) toggle on the rising edge of the *CLK* pulse and the status signals *Q* change with a delay t_{pCQ} . After passing through the combinational circuits and after the end of transients (due to the delays and hazards), the excitation signals E_i are generated with the delay t_{pKmax} . After the FF setup time t_s , another clock pulse can be applied. The period and frequency of the *CLK* pulses is thus:

$$T_{CLK} \geq t_{pCQ} + t_{pKmax} + t_s, \quad f_{CLK} = 1/T_{CLK}$$

With a given circuit technology, the *CLK* period can be reduced only by shortening the signal path through the combinational circuits. The availability of *Q* signals on the FF outputs in both **direct** and **inverted** form contributes to this, so that the combinational circuits can have only **two stages**.

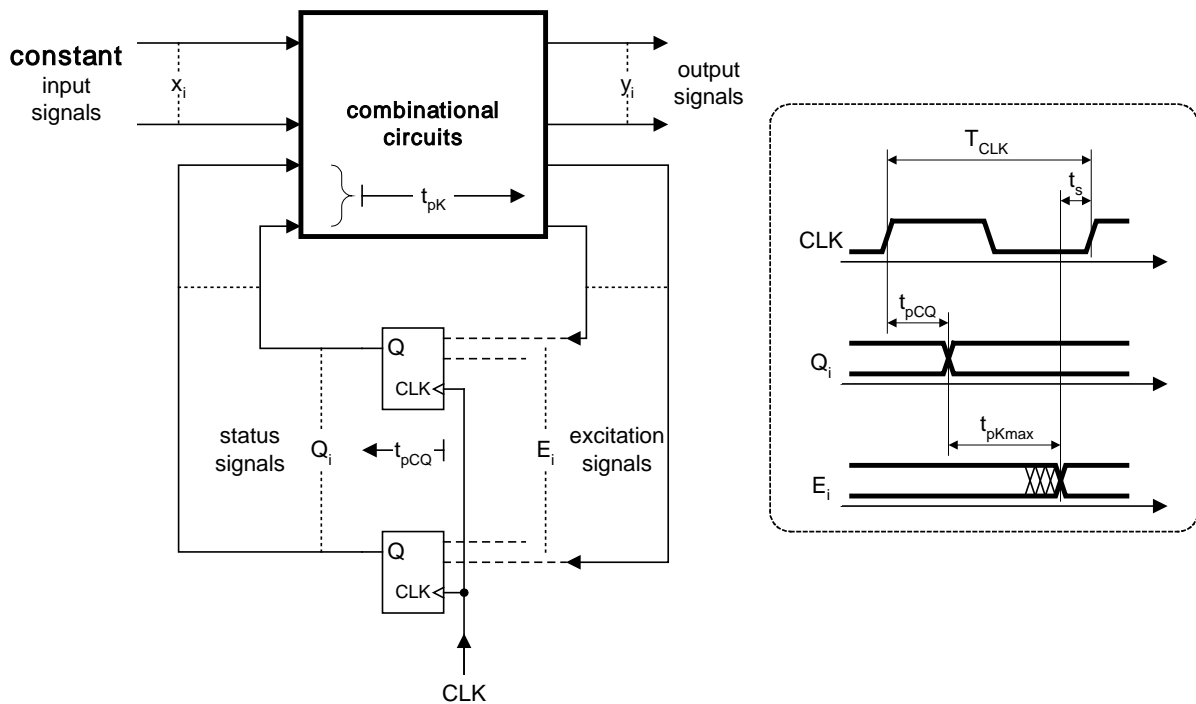


Fig. 9: Derivation of maximum clock frequency

The second task will be to determine the conditions under which the **input signals** may change. We get necessary setup time and hold time of input signals *X* before and after the *CLK* edge. Unlike the setup and hold times of the FFs themselves, we will call them the setup and hold times of the (whole) **sequential circuit**.

First we notice the time of sequential circuit's setup time. Figure 10 on the left shows the situation, the upper figure on the right shows the time diagrams. After changing the input with

the delay t_{pKmax} , the excitation signals E stabilize. After the FF setup time t_s , a clock pulse can be applied. Thus we get **the setup time of the sequential circuit** t_{ss} as:

$$t_{ss} \geq t_{pKmax} + t_s$$

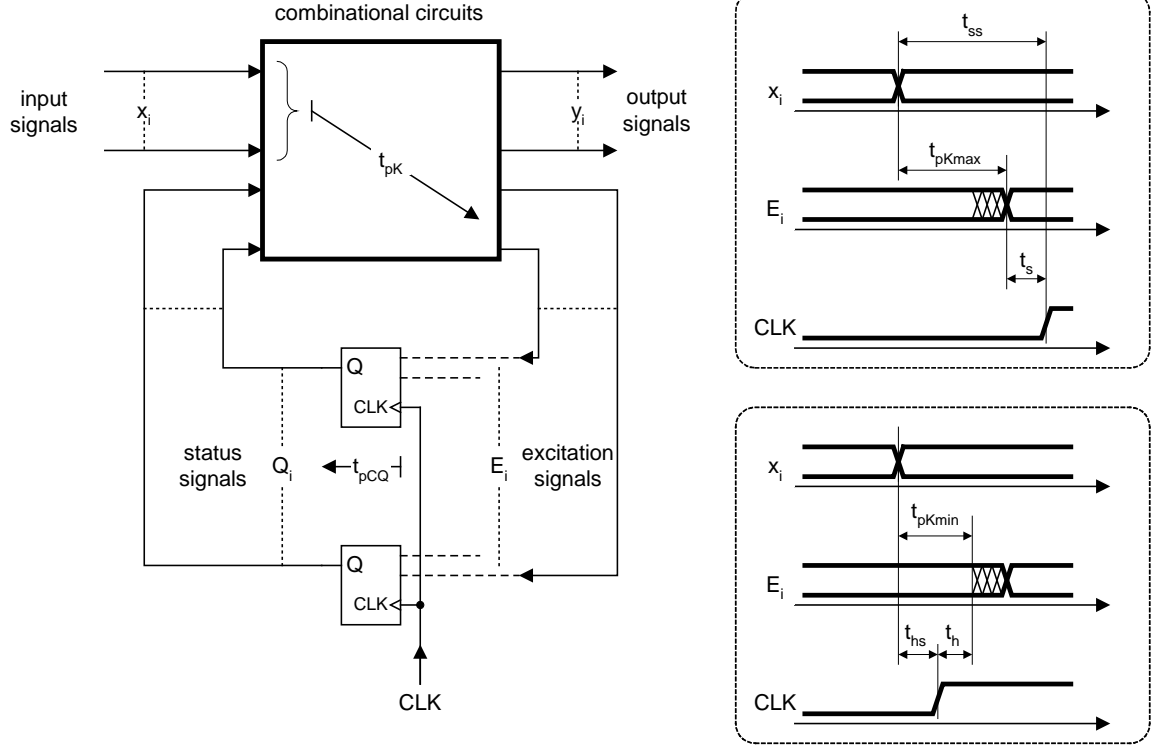


Fig. 10: Derivation of setup time and hold time of inputs with respect to clock pulses

The excitation signals E_i must remain constant for the FF's hold time t_h after the edge of CLK . The minimum time when E_i does not change after the input change is t_{pKmin} . The lower time diagram on the right shows the **hold time of the sequential circuit** t_{hs} :

$$t_{hs} \geq t_h - t_{pKmin}$$

where t_{pKmin} indicates the **minimum** delay time of the combinational circuits. Possible negative value of t_{hs} means that the necessary time t_{hs} starts **before the edge** of CLK .

Thus, the **maximum and minimum** delay of the combinational circuits t_{pKmax} , t_{pKmin} is used in the formulas. The minimum delay can be zero, as the output of one flip-flop can be directly connected to the input of another FF (which is not an exceptional case).

If we allow input changes at any time during the CLK period, then it may happen that the inputs change just in the "metastable window" of the length $t_{ss} + t_{hs}$, which results in unreliable function of flip-flops. In this case, the transitions between states do not always correspond to the desired ones. The resulting unreliability is occasional and **random**. Which is worse, these failures may not be detected during system validation and testing.

9 Clock pulses

The relations above were derived assuming ideal clock waveforms. However, the reality is not such and the clock impulses suffer from two **imperfections**:

1. Shift of clock impulses - **clock skew** - is measurable between two points in the pulse distribution network. Due to different delays in this network, there will be time shifts at the inputs of two or more flip-flops. This delay is permanent in nature, although it may vary slightly with temperature or long-term fluctuations in supply voltage. However, it does not change within a few impulses. The time shift can be calculated in advance if the delay in components and connections is known.

2. Instability of the edges of clock impulses - **jitter** - is defined as the difference of the positions of the edges between two consecutive clock pulses. In the range of certain extreme positions, the current position of the edges is random. Jitter is the result of noise in the clock source (oscillator), parasitic coupling from other circuits, and short-term fluctuations in the supply voltage.

The situation is shown in Fig. 11, where the nominal positions of the CLK pulse edges are indicated by a thick line. The time difference between the edges of CLK_1 and CLK_2 , denoted by t_{sk} , is apparent. Also visible is the extent of changes in edge position - the time difference between extreme positions is indicated as t_j . Both effects add up, so the time difference between the edges of CLK_1 and CLK_2 can be up to $t_{sk} + t_j$.

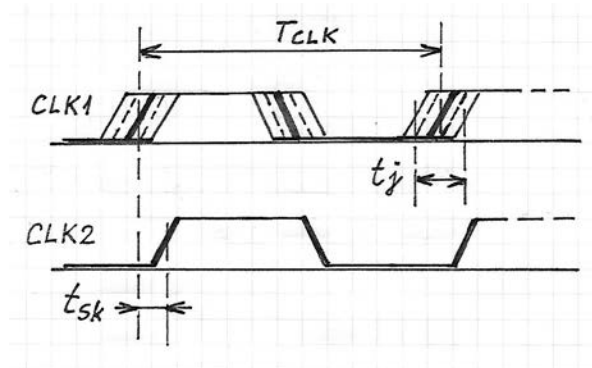


Fig. 11: Illustration of skew and jitter

In this situation it is necessary to modify the above calculations of the time T_{CLK} , t_{ss} , and t_{hs} . Both clock skew and clock jitter must be considered and the worst case must be calculated.

The significance of delays in a signal distribution network is obvious. It is mainly the distribution of clock pulses which in the synchronous system are delivered to the flip-flops. In order to avoid overloading of the CLK source, the network is designed in the form of a **tree** with inserted signal **repeaters** - see Fig. 12. With proper design of signal paths, it is also possible to compensate for delay lengths in individual branches, thus minimizing signal shifts.

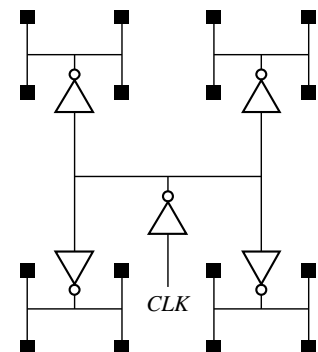


Fig. 12: Recommended shape of the clock pulse distribution network

In the integrated version, each component has a delay of several tens of ps ($1 \text{ ps} = 10^{-12} \text{ s}$). Connections can have a much longer delay - for example, a few mm long connections can have a delay up to 1000 ps. The delay of the connections on the chip is longer than that of a common metallic line. The reason is the high resistance of the connection due to its small cross-section and the relatively large capacity against the ground due to very small distances. The link then acts as an RC circuit with a relatively long time constant. A certain help was the transition from aluminum as a material to copper. However, the fact is that the delay in the connections is **substantial** and can easily exceed the delay of the semiconductor components. It must therefore be taken into account in the design. However, the connection pattern is not known at the beginning of the design, but **at the end**. The importance of **computer design systems** that can perform time analysis of signals in early stages of development is evident.

10 Delays of output signals

The last important parameters are related to the **output signals**. In the case of both **Moore** and Mealy FSM, the outputs are functions of the internal state and are delayed after the *CLK* edge by the delay of the flip-flops t_{pCQ} and the output combinational circuits t_{pK} , i.e. by a total of $t_{pCQ} + t_{pK}$. In the Mealy FSM, the outputs are in addition also the functions of inputs; the output signals can thus vary between the clock pulses with a delay t_{pK} . The timing of the **Mealy** FSM is shown in Fig. 13.

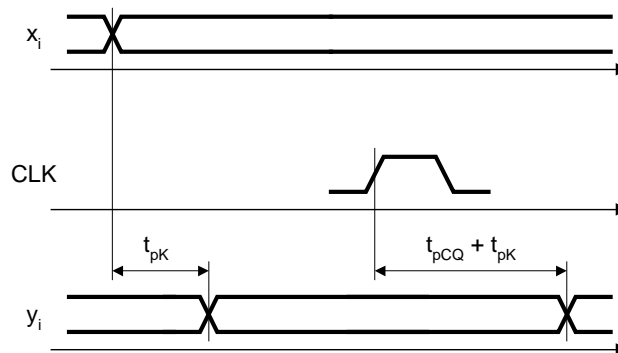


Fig. 13: Delay between output and input of the sequential circuit

Combinational circuits usually suffer from hazards, so false pulses can also appear at the output of the sequential circuit. They can be eliminated by the methods discussed in the chapter "Transients in combinational circuits". Most often, an **output register** will be included in the path of output signals - see Fig. 14. The output register can be controlled by the same *CLK* pulses, which are used in the state register of the sequential circuit. Mealy circuit will, however, change one its important characteristic - output signals are now constant between *CLK* pulses. For both Mealy and Moore circuit, the output signals will be **delayed by one CLK** period after the internal states. This must be taken into account when designing the system.

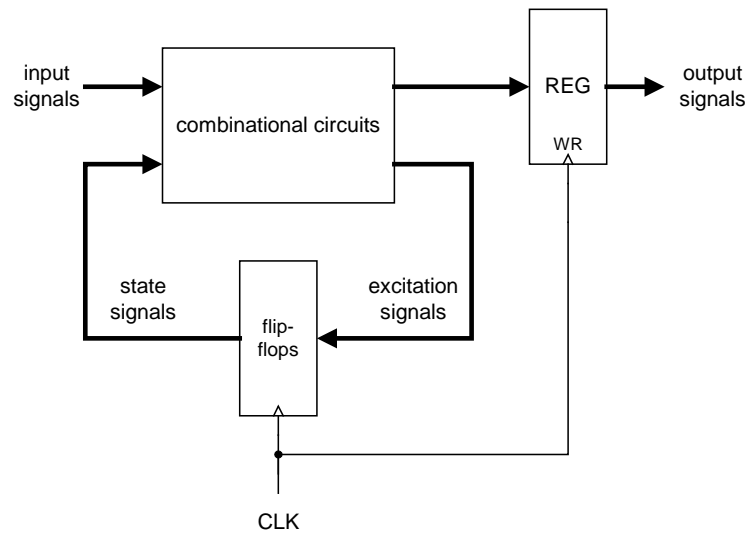


Fig. 14: Inclusion of an output register for hazards elimination

11 Connected sequential circuits

As shown, the timing of the signals at the input of the sequential system respect the time parameters t_s and t_h of the flip-flops. Incorrect timing leads to unreliable operation of the entire system. However, the input signals are supplied by a preceding system, which may or may not use the same clock pulses. In other words, it may or may not be in synchronism with a given sequential system. If it is **synchronous**, then it is not a problem to adjust the timing - it is sufficient to insert appropriate delays between the CLK inputs.

The whole situation becomes complicated if both systems are **not synchronous** with each other. Then, for the signals transmitted from the output of the preceding system to the inputs of the following system, the required critical time parameters cannot be guaranteed. This situation is briefly labeled as **Clock Domain Crossing**, abbreviated as **CDC**. Data generated in the domain controlled by CLK_1 is transmitted to the domain controlled by CLK_2 . The issue of CDC is very important in the design of mutually connected sequential systems.

In addition to metastability, other phenomena can occur due to improper timing if the number of input signals is greater than one. When the input status changes, it is expected that all signals change at the same moment. However, due to various delays in individual signal paths, the signals change in different moments. As a consequence, during the transition from the past state to next state, several combinations of values ("intermediate states") exist that belong **neither to the past nor to the next steady state**. If these false input states occur during the active edge of the clock pulse, they will be processed by the sequential circuit and may cause a completely erroneous function. This phenomenon is called **signal races**. As well as in the case of metastability, a simple solution is possible in the case of systems with a common distribution of synchronization pulses. However, in the case of asynchronous systems, a significant problem arises - the **CDC problem**. Its solution is discussed in the following chapters.