

Logic Design – Lab 3: “Design of asynchronous sequential logic circuits using transition tables and output tables”

For the report, please complete the tasks marked in green boxes only!

Main rules

For **combinatorial circuits** we had no memory contained:

$$Y^t = f(X^t). \quad (1)$$

where: Y – output vector, X – input vector. The index (superscript) t was not important and thus was omitted.

In turn, for **sequential circuits** the memory is included:

$$Y^t = f(X^t, X^{t-1}, X^{t-2}, \dots), \quad (2)$$

where: $X = (x_1, x_2, \dots, x_n)$, $Y = (y_1, y_2, \dots, y_m)$ sets of input and output signals.

The memory can be implemented:

- using feed-back connections in the circuit composed of logic gates
- using the own contacts on the way of supplying the coils
- using flip-flops which are elementary elements of memory.

The formula (2) can be substituted by a set of two equations:

$$\begin{aligned} A^t &= \delta(X^{t-1}, X^{t-2}, \dots) = \delta(X^{t-1}, A^{t-1}), \\ Y^t &= \lambda(X^t, A^t), \end{aligned} \quad (3)$$

where:

A^t – internal state at time t ,

δ – transition function

λ – output function.

Introducing a single step delay one obtains:

$$\begin{aligned} A^{t+1} &= \delta(X^t, A^t), \\ Y^t &= \lambda(X^t, A^t). \end{aligned} \quad (4)$$

and this is a description of the Mealy type automata (Fig. 1a).

It is also possible to have:

$$\begin{aligned} A^{t+1} &= \delta(X^t, A^t), \\ Y^t &= \lambda_1(A^t). \end{aligned} \quad (5)$$

and this is a description of the Moore type automata (Fig. 1b).

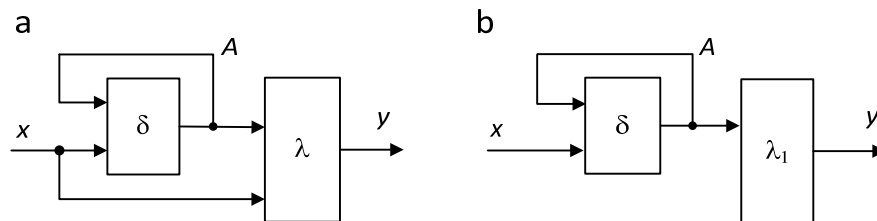


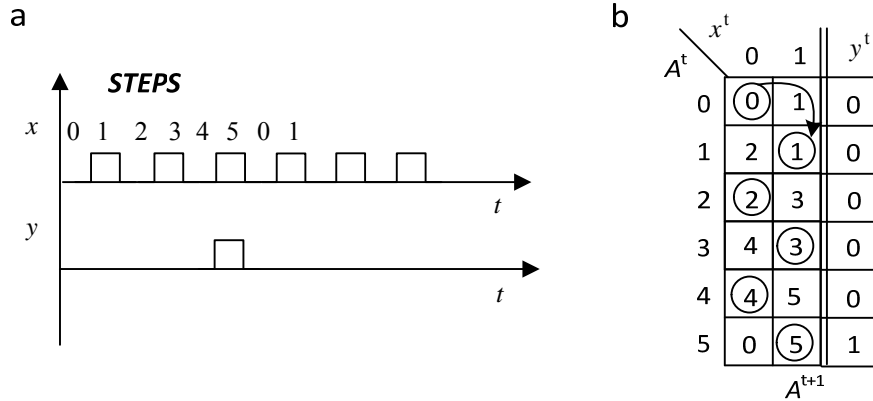
Fig. 1. Structure of sequential automata: a) Mealy type, b) Moore type

Automata's description

1. String of numbers (0 and 1)

$X = x$	0	1	0	1	0	1	0	1
$Y = y$	0	1	1	0	0	1	1	0

2. Time diagram \rightarrow transition table and output map

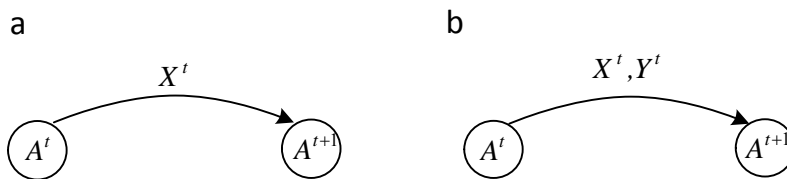


Internal state A_i under inputs X_j is **stable** if:

$$\delta(A_i^t, X_j^t) = A_i^{t+1}. \quad (6)$$

The stable internal states are marked in transition tables with the circle (for example: ⑤).

Transition between two states: graph for Moore type (a) and Mealy type (b) automata:



Minimization of initial transition tables

Minimization is performed to simplify the circuit.

By minimizing internal states one gets that two or more rows are substituted by a resultant single row.

Internal states A_i i A_j are **not contradictory**: 2 and 2, 7 and –, – and –.

Output states Y_m i Y_k [as for example (y_1, y_2, y_3)] are **not contradictory** if they are identical or one of them is a don't care, as for example: (010) and (–1–) are not contradictory.

Internal states A_i i A_j are **in accordance** if they can be substituted by a resultant single state without influencing the circuit operation and this takes place if

- under any X they transit to states which are not contradictory or are in accordance,
- under any X the corresponding output states Y are not contradictory.

Under checking accordance of the particular internal states it can happen that the **accordance** depends upon the **accordance of the other pair of states**.

$$A_i \cap A_i = A_i$$

$$A_i \cap - = A_i$$

$$A_i \cap A_i = A_i$$

$$A_i \cap - = A_i$$

$$- \cap - = -$$

Flip-flops

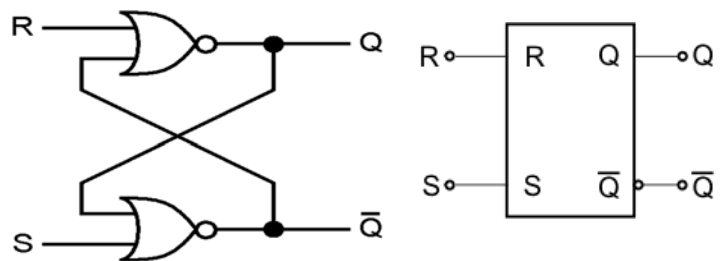
Flip-flops can be divided into common types: the **SR** ("set-reset"), **D** ("data" or "delay"), **T** ("toggle"), and **JK** types are the common ones. The behavior of a particular type can be described by what is termed the characteristic equation, which derives the "next" (i.e., after the next clock pulse) output, $Q_{\text{next}}(Q^{t+1})$ in terms of the input signal(s) and/or the current output, $Q(Q^t)$.

Set-Reset latches - SR NOR latch

When using static gates as building blocks, the most fundamental latch is the simple *SR latch*, where S and R stand for *set* and *reset*. It can be constructed from a pair of cross-coupled **NOR** logic gates. The stored bit is present on the output marked Q.

While the R and S inputs are both low, feedback maintains the Q and \bar{Q} outputs in a constant state, with \bar{Q} the complement of Q . If S (*Set*) is pulsed high while R (*Reset*) is held low, then the Q output is forced high, and stays high when S returns to low; similarly, if R is pulsed high while S is held low, then the Q output is forced low, and stays low when R returns to low.

SR latch operation							
Characteristic table				Excitation table			
S	R	Q^{t+1}	Action	Q^t	Q^{t+1}	S	R
0	0	Q	hold state	0	0	0	X
0	1	0	reset	0	1	1	0
1	0	1	set	1	0	0	1
1	1	X	not allowed	1	1	X	0



Note: X means don't care, that is, either 0 or 1 is a valid value.

The $R = S = 1$ combination is called a **restricted combination** or a **forbidden state** because, as both NOR gates then output zeros, it breaks the logical equation $Q = \text{not } Q'$. The combination is also inappropriate in circuits where *both* inputs may go low *simultaneously* (i.e. a transition from *restricted* to *keep*). The output would lock at either 1 or 0 depending on the propagation time relations between the gates (a race condition).

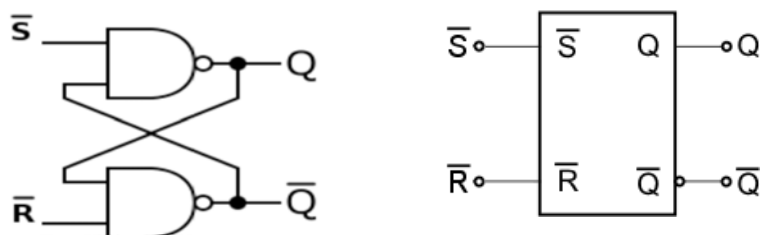
To overcome the restricted combination, one can add gates to the inputs that would convert $(S, R) = (1, 1)$ to one of the non-restricted combinations. That can be:

- $Q = 1$ (1,0) – referred to as an *S (dominated)-latch*
- $Q = 0$ (0,1) – referred to as an *R (dominated)-latch*

Set-Reset latches - SR NOR latch

This is an alternate model of the simple SR latch which is built with **NAND** logic gates. *Set* and *reset* now become active low signals, denoted \bar{S} and \bar{R} respectively. Otherwise, operation is identical to that of the SR latch. Historically, SR-latches have been predominant despite the notational inconvenience of active-low inputs.

SR latch operation		
\bar{S}	\bar{R}	Action
0	0	Not allowed
0	1	$Q = 1$
1	0	$Q = 0$
1	1	No change



The design of sequential automata:

- a) initial transition table with output map
- b) reduction triangle
- c) reduction graph
- Moore automata:
 - d) minimization effect:
 - left: reduced transition table with output map
 - right: as above but after renumbering
- Mealy automata:
 - e) reduced transition table
 - f) as above but after renumbering
 - g) output table.

Task:

A asynchronous sequential logic circuit of two input signals (x_1, x_2) and two output signals (y_1, y_2) is defined with the following transition table and output map (table):

x_1x_2							
A^t		00	01	11	10	y_1y_2	
0	0	0	–	–	3	00	
1	1	1	–	–	2	00	
2	2	0	–	4	2	10	
3	3	1	–	5	3	01	
4	4	–	–	4	3	11	
5	5	–	–	5	2	11	
		A^{t+1}					

Determine the equations suitable for implementation with NOR gates only and simulate them using Electronic Workbench to verify that design is working correctly (according to the given transition table and output map).

Secondly, please make implementation with with S-R flip-flops (as memory elements) and test it, too.

Tip: see example 4.1.3, page 73 in Handbook “Układy Logiczne – ćwiczenia laboratoryjne”, Mirosław Łukowicz, PWr, 2002, and “Lab 3a supplement....doc”.