

6 Sequential Logic

Student Group

First Name	Surname	Matrikel Nr.

Table of Contents

6 Sequential Logic	2
6.1 First Terminology	2
Exercise 6.1.3.1 Example of a State Machine	5
6.2 Classical State Machine Types	6
6.2.1 Moore Machine	6
Note!	6
6.2.2 Mealy Machine	7
Note!	7
Exercise 6.2.2.1 Invalid States of the Mealy Machine	7
6.2.3 Medvedev Machine	8
Note!	8
Note!	9
6.3 State Diagram, State Transition Diagram	9
6.3.1 Motivation	9
6.3.2 Simple logic Example	10
6.3.3 First Adaption for Up-Counter	12
Exercise 6.3.3.1. State Transition Diagram in Digital.exe	13
Exercise 6.3.3.2. State Transition Diagram II in Digital.exe	14
Exercise 6.3.3.3. State Transition Diagram III in Digital.exe	15
Exercise 6.3.3.4. State Transition Diagram III in Digital.exe	16
Exercises	17
Exercise 6.1.1. State Transition Diagram I - fill the gaps	17
Exercise 6.1.2. State Transition Diagram II - Sequence	19
Exercise 6.1.3. State Transition Diagram III - Milling Machine	20
Exercise 6.1.4. State Transition Diagram IV - Find a sequence	20
Exercise 6.1.5. State Transition Diagram V - LED Fun	21
Exercise 6.1.6. State Transition Diagram VI - Roll the dice	21

6 Sequential Logic

“I Know What You Did Last Cycle”

6.1 First Terminology

The most important term for the upcoming topics is the word **state**. But what is a state? It is a unique situation, where the possible next steps (= possible next states), the inner behavior, or the outputs are distinguishable from other situations. Here are some practical examples:

- Being happy or being sad, are two different states, since the inner behavior is different (this least often also to a different output).
- Similarly, an empty memory (or hard drive) is in a different state compared to a filled one.
- A traffic light showing green has an output distinguishable from red, or yellow.

Sequential logic is used to describe logic circuits that show internal states (“stateful logic”), and therefore have at least one memory element (= flip-flop).

The following terminology is used in the upcoming explanations:

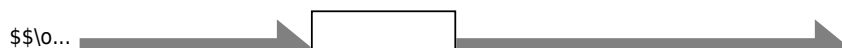
- The **input vector** $\vec{X}$ represents the k inputs $X_0 \dots X_{k-1}$.
- The **output vector** $\vec{Y}$ represents the l outputs $Y_0 \dots Y_{l-1}$.
- The **state vector** $\vec{Z}$ represents the m inputs $Z_0 \dots Z_{m-1}$.
- The sign (n) or n marking the current point in time and therefore e.g. the current state $Z_0(n)$.
- The sign $(n+1)$ or $n+1$ marking the next upcoming point in time and therefore e.g. the next state $Z_0(n+1)$.
- Sequential logic circuits are also called **Finite State Machines** (FSM) or sometimes also shortened to “state machines”.

The [figure 12](#) shows the different terms in an abstract diagram. The “current” output values are here the values, which are shown after the delay time of the gates (about some nanoseconds).

Fig. 1: Abstract View onto a Sequential Logic

The principle interior of the black box in [figure 12](#) was already shown in one practical application in the [previous chapter](#): We saw, that we need the combination of combinatorial logic and some storage components. Additionally, both have to be connected by feeding back some of the outputs back to the combinatorial logic. The output bits $\vec{Y}$ can result either from the combinatorial logic or the flip-flops. This is shown in [figure 13](#).

Fig. 2: One Step more into the Sequential Logic



Exercise 6.1.3.1 Example of a State Machine

[figure 1](#) depicts a state machine.

- What happens, when X is changed? (click onto the 0 on the left)
On which edge the change is triggered?
- Write down how many components each vector $\vec{X}$ and $\vec{Y}$ has.
- How many bits (= flip flops) might the state vector $\vec{Z}$ need?

Fig. 3: Example for a sequential logic

One simple example of sequential logic is shown in [figure 4](#). There, the combinatorial logic is explicitly shown. Depending on the input X the output $\vec{Y}$ shows an up-counting 2-bit value counting $0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 0 \rightarrow \dots$. This is a simple state machine that will be used in the net chapters

Fig. 4: State machine of an Up-Counter

6.2 Classical State Machine Types

The up-counter in the previous sub-chapter was able to count from 0 to 3 . But what can we do to count differently, like $7, 6, 1, 5$? To understand this, a simpler situation will be investigated. So, let's look at how one can create an up-counter counting $1, 2, 3, 4$.

6.2.1 Moore Machine

The first idea might be to use what we already have: an up-counter, which facilitates 2 flip-flops to result in 2-bit output. The wanted new state machine needs 3 bits for the output, since the binary representation of our outputs is $001_2, 010_2, 011_2, 100_2$.

A simple idea is to take the 2-bit up-counter and add a combinatorial logic in behind it. This logic shall convert the 2-bit up-counter output 00_2 into 001_2 , the 01_2 into 010_2 , 10_2 into 011_2 and 11_2 into 101_2 . This can be logic can be created by:

- writing down the truth table
- putting the values into a Karnaugh map
- extracting the formula with a view onto the implicants
- generating the circuit with gates

When this is done, the result looks like [figure 5](#)

Fig. 5: Up-Counter 1..4 as a Moore Machine

This resembles a so-called **Moore Machine**.

Note!

A state machine is a **Moore Machine** when the output values $\vec{Y}$ depend only on the state values $\vec{Z}$. For this the Moore machine uses two combinatorial circuits:

- The input circuit, which processes the input values $\vec{X}(n+1)$ and the state values $\vec{Z}(n)$ (of the previous step) in such a way that the new states $\vec{Z}(n+1)$ are generated.
- The output circuit, which transform the state values $\vec{Z}(n)$ into the output values $\vec{Y}(n)$.

The properties of a Moore machine are:

- The number of flip-flops is only given by the number of states m .
- The output only changes when an edge on the clock input happens. The Moore machine is a **synchronous state machine**.
- The Moore machine usually needs fewer logic gates. But this comes with the cost of optimizing two combinatorial logic circuits.

6.2.2 Mealy Machine

When looking at [figure 5](#) a bit more in detail, one can see, that the outputs Y_0 and Y_1 just equals the output of the first combinatorial logic circuit. This is not surprising: the input logic circuit shows the $\vec{Z}(n+1)$ and this is for the counter always the stored value plus one, except when the maximum is reached.

With this information, the state machine in [figure 5](#) can be simplified by using the outputs of the input circuit for Y_0 and Y_1 . This is shown in [figure 6](#).

Fig. 6: Up-Counter 1..4 as a Mealy Machine

Note!

A state machine is a **Mealy Machine** when the output values $\vec{Y}$ depend not only on the state values $\vec{Z}$. For this, the mealy machine uses two combinatorial circuits:

- The input circuit, which processes the input values $\vec{X}(n+1)$ and the state values $\vec{Z}(n)$ (of the previous step) in such a way that the new states $\vec{Z}(n+1)$ are generated.
- The output circuit, which transforms the state values $\vec{Z}(n)$ and some inputs from ahead of the flip-flops into the output values $\vec{Y}(n)$.

The properties of a mealy machine are:

- The number of flip-flops is only given by the number of states m .
- The output **not** only changes when an edge on the clock input happens: It also depends on the input $\vec{X}(n)$. The mealy machine is an **asynchronous state machine**.
- The mealy machine usually needs fewer logic gates.
- The mealy machine must be designed properly not to get invalid outputs.

Exercise 6.2.2.1 Invalid States of the Mealy Machine

The mealy machine in [figure 6](#) can show invalid outputs. Try to find these by the correct timing of the input $X=1$ or $X=0$.

- Which outputs can be created?

6.2.3 Medvedev Machine

Fig. 7: Up-Counter 1..4 as a Medvedev Machine

Note!

A state machine is a **Medvedev Machine** when the output values $\vec{Y}$ are directly given by the state values $\vec{Z}$. For this, the Medvedev machine uses only one combinatorial circuit. This circuit processes the input values $X_{(n+1)}$ and the state values $Z_{(n)}$ (of the previous step) in such a way that the new states $Z_{(n+1)}$ and the output values $Y_{(n+1)} = Z_{(n+1)}$ are generated.

The properties of a Medvedev machine are:

- The number of flip-flops is given by the number of outputs l .
- The output only changes when an edge on the clock input happens: The mealy machine is an **synchronous state machine**.
- The Medvedev machine usually needs more logic gates.

The [figure 8](#) shows the principle differences in the architecture of the state machines.

Fig. 8: States of Water

Note!

This chapter is only focussing on Moore machines.

6.3 State Diagram, State Transition Diagram

6.3.1 Motivation

The diagrams of different states are well known from physics for example the state diagram (or better: phase diagram) of water, where its three states are: solid ice, liquid water, and gaseous steam. The possible state transitions are due to temperature increase or decrease.

In [figure 9](#) image (1) the states of water are shown on the temperature axis. When only the state transitions are relevant, the states are simplified to a circle, showing the state name and behavior. The transitions are depicted as arrows, where the needed condition is written onto (See [figure 9](#) image (2)). This diagram is called **state transition diagram**.

Fig. 9: States of Water

...

For matter not only the dimension “temperature” is important, but also the “pressure”. The full phase diagram is shown in [figure 10](#) image (1). By this, another variable is available and more transitions. These can be drawn into the state transition diagram ([figure 10](#) image (2)).

Fig. 10: States of Water

...

6.3.2 Simple logic Example

In Germany, often one has to pay for entering the toilet. An example of such an entrance control system is shown in [figure 11](#). In this (artificial) example, one can pay either 50ct or 1€. Once paid, the turnstile will release and one can enter. Once the turnstile was pushed the entrance is closed again.



Fig. 11: Entrance Control for Toilets

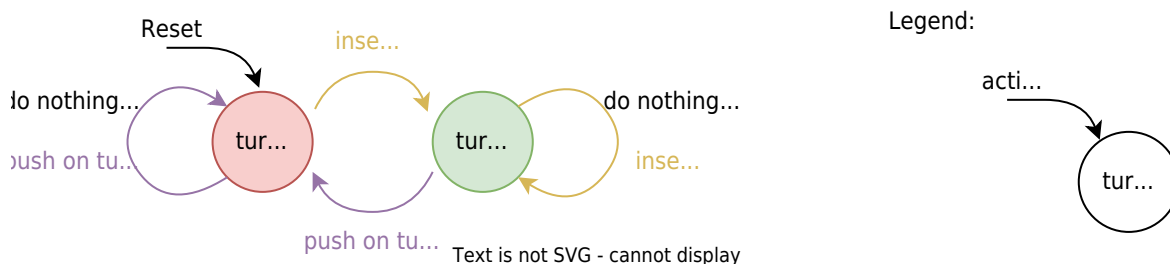
The [figure 12](#) the state transition diagram is drawn.

- The two states are that (1) the turnstile is opened and one can go through and (2) the turnstile is closed and one cannot enter anymore.
- The transitions are given by the done actions: one can either insert a coin or push on the turnstile.

Important here are some additional considerations:

- For the state transition diagram one has to **look for all possible transitions**. So, also pushing a closed turnstile or inserting more coins have to take into account.
- A state transition diagram is not complete without a **legend** and without an **beginning/reset point**. The reset point is given by an arrow with "reset" written on it

Fig. 12: State Transition Diagram of the Entrance Control for Toilets



Out of this state transition diagram, one can create a table-like representation, see [figure 13](#).

Fig. 13: "non-binary" State Transition Table of the Entrance Control for Toilets

Toilet Entrance Control			
current state	input / event	next state	output / action
turnstile closed	push turnstile	turnstile closed	disallow entrance
turnstile closed	insert coin	turnstile opened	allow entrance
turnstile opened	push turnstile	turnstile closed	disallow entrance
turnstile opened	insert coin	turnstile opened	allow entrance

the inputs, outputs, and states have to be encoded into binary, to investigate this table a bit more. How the binary value is connected to the outputs does not matter. We will choose the following coding:

- Encoding of the states: turnstile closed $\triangleq \$Z=0\$, turnstile opened $\triangleq \$Z=1\$,$$
- Encoding of the inputs: no coin inserted $\triangleq \$Xc=0\$, coin inserted $\triangleq \$Xc=1\$, turnstile not pushed $\triangleq \$Xp=0\$, turnstile pushed $\triangleq \$Xp=1\$,$$$$
- Encoding of the outputs: disallow entrance $\triangleq \$Y=0\$, allow entrance $\triangleq \$Y=1\$,$$

This table is shown in [figure 14](#) and is called **state transition table**.

Fig. 14: State Transition Table of the Entrance Control for Toilets

Toilet Entrance Control				
current state	input	next state	output	
0	0	0	0	
0	1	1	1	
1	0	0	0	
1	1	1	1	

Interestingly, the logic circuit for this state transition table was already part of the course: it is the RS flip-flop! When looking deeper into the table in [figure 14](#) one can substitute $\$Xc\$$ with $\$S\$$ (as in Set) and $\$Xp\$$ with $\$R\$$ (as in Reset) to directly get the truth table of the RS flip flop.

6.3.3 First Adaption for Up-Counter

In the previous sub-chapter (6.2) we had a look at different implementations of an up counter and in this chapter the way to represent a state machine via a state transition diagram. So one question is: what do these up-counters look like in the state transition diagram?

The answer is quite simple: there are 4 states, so 4 “circles” are needed. These states need to have circular connections to get the wanted increase from $\$00_2=0_{\{10\}}\$, to $\$01_2=1_{\{10\}}\$, to $\$10_2=2_{\{10\}}\$, to $\$11_2=3_{\{10\}}\$ and then back to $\$00_2=0_{\{10\}}\$. The first state shall be $\$00_2=0_{\{10\}}\$, so after the reset, the state machine will get entered there. Finally, the output vector is similar to the state vector. With a deeper look at it: This is therefore in particular a Medvedev machine!$$$$$$

In [figure 15](#) both types of state machines are shown.

- In the Moore machine the already seen “halving of the circle” is used to show the distinct state vector in the upper half and the output vector in the lower half.
- In the Medvedev machine only one value is written in the circle, since here the state vector is also the output vector.

Fig. 15: State Transition Diagram of an up-counter 0..3 as Moore Machine

As Moore Machine

Exercise 6.3.3.1. State Transition Diagram in Digital.exe

The shown state transition diagram can easily be created in the tool Digital:

1. Click on the menu Analysis » Finite State Machine
2. Here one either can add new states with a right click, or - easier - go to the menu Create » Create Counter » 4 States
3. A (up) counter will get created

Tasks:

1. Make the state machine get alive:
 1. Click to the menu Create » State Transition Table in order to see the state transition table
 2. Here, click on the menu Create » Circuit
 3. Now, the circuit can be started via the start button. The present state in the state transition diagram will also get highlighted.

The next step is to transform this into an up counter from 0..3. For this simply the output has to be changed. This means the numbers in the “lower half of the circles” have to be adapted (see in [pic16](#)).

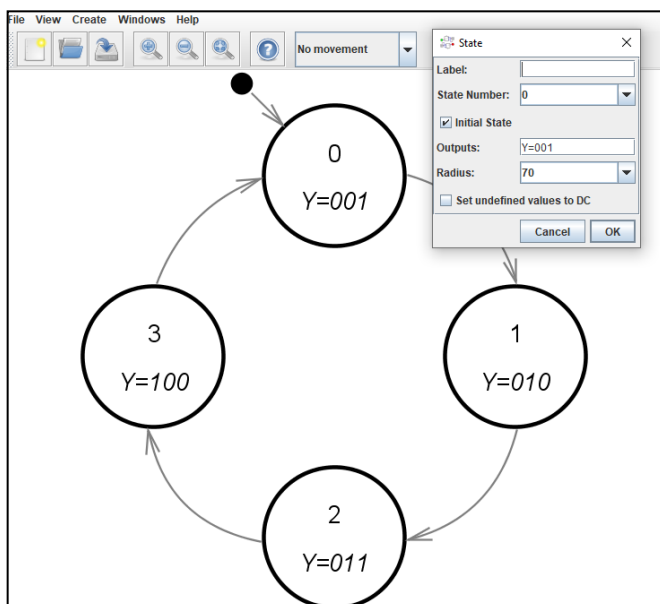
Fig. 16: State Transition Diagram of an up-counter 0..3 as Moore Machine

SSRe...

Exercise 6.3.3.2. State Transition Diagram II in Digital.exe

This exercise is directly attached to exercise 6.3.3.2

The counter shall now be changed in such a way, that it counts \$1 - 2 - 3 - 4\$. For this: right-click on each present state beginning with state 0 and add for Outputs \$Y=001\$ and up-counting (see image).



Tasks:

1. Make this state machine again get alive
2. Look at the circuit: Is it a Moore, Mealy, or a Medvedev machine?

Looking at what we got, there is one part missing: the state machine does not have any input... So the input vectors have to be added to the transition (arrows). For an activatable counter, there only has to be one input \$X\$, which acts as an enable input.

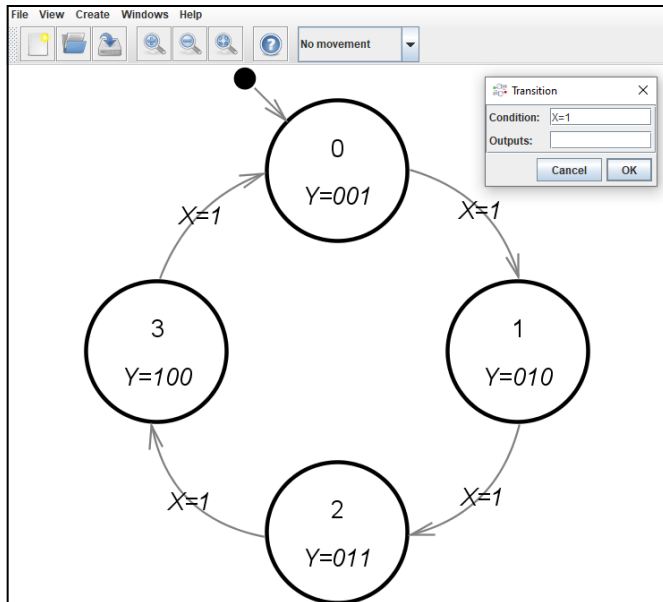
Fig. 17: State Transition Diagram of an up-counter 1..4 as Moore Machine with enable input

\$\$Re...

Exercise 6.3.3.3. State Transition Diagram III in Digital.exe

This exercise is directly attached to exercise 6.3.3.2

The last step is to add the transitions: right-click on all transitions and add $X=1$ as a Condition.



Tasks:

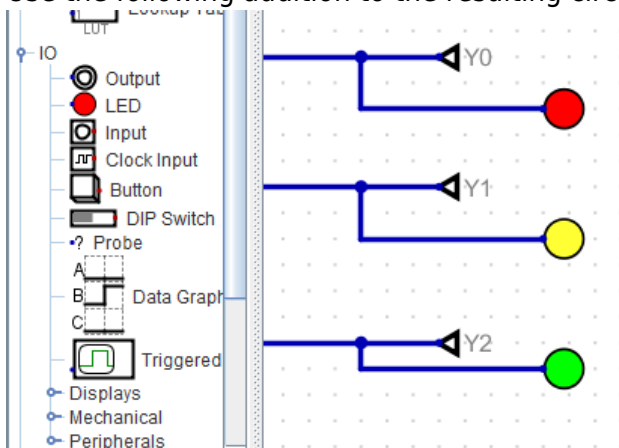
1. Make this state machine again get alive

Exercise 6.3.3.4. State Transition Diagram III in Digital.exe

This exercise is directly attached to exercise 6.3.3.3

With the state machine in 6.3.3.3, it is also possible to create a state machine that can resemble a traffic light state machine. This shall transit from red to red-yellow, to green, to yellow, and then back to red.

Use the following addition to the resulting circuit to get the light running:



Tasks:

1. Think about the right way to change the outputs in the state machine.
2. Implement the needed correction and test the state machine.

Exercises

Exercise 6.1.1. State Transition Diagram I - fill the gaps

The following state transition diagram shall be given:

1586...

- There are two transitions marked with \$A\$ and \$B\$.
What values do the inputs need to have to show all transitions explicitly?

Strategy

- Find the transitions wanted
- Look at which state these transitions start.
- Which other transitions start there?
- Which transition conditions are missing?

Solution

- transition A
 - Starts at state \$000\$
 - Also transition with \$11\$, \$00\$ starts here
 - \$01\$, \$10\$ are missing
- transition B
 - Starts at state \$010\$
 - Also transition with \$0-\$ starts here
 - \$1-\$ are missing

Result

- A: \$01\$, \$10\$
- B: \$1-\$

- How many flip-flops are necessary for such a Moore Machine?

Strategy

Each flipflop can store one Bit. Each stored bit can be used to address states. So, check the number of bits n of states Z is ("size of the state vector").

Be aware, that one bit can address a maximum of 2 states, two bits a maximum of 4 states, three bits a maximum of 8 states, and so on.

Solution

- Number of bits i of states Z_i is given in the legend. The number of bits i has also to fit the number of states.
- Here the legend shows Z_2, Z_1, Z_0 , so there are 3 bits.
- Also the number of states in the diagram is 5. This can only be numbered with at least 3 bits.

Result

3

- Fill in the missing cells in the following state transition table:

$ss2_2(n)ss2_1(n)ss2_0(n)$	$ss1_1ss1x_0ss1ss2_2(n+1)ss1ss2_1(n+1)ss1ss2_0(n+1)ss1$	$ssy_3(n)ss1y_2(n)ss1y_1(n)ss1y_0(n)$
0 1 0	0 1	
1 1 0	1 1	
0 0 1	1 1	
0 0 1	0 1	

Strategy

Check for each line:

- What is the start/present state (given by the bits $\text{\color{red}\{Z_2(n), Z_1(n), Z_0(n)\}}$ in the line)?
- Which transition is shown as start/present state (given by $\text{\color{brown}\{X_1, X_0\}}$)?

Out of this orientation, the next columns can be derived:

- At the end of the transition, the next state can be found (necessary for columns $\{\textcolor{green}{Z}_{2(n+1)}, Z_{1(n+1)}, Z_{0(n+1)}\}$)

- Within the state symbol of the present state, the output values can be found (necessary for columns $\{Y_2(n), Y_1(n), Y_0(n)\}$)



Result

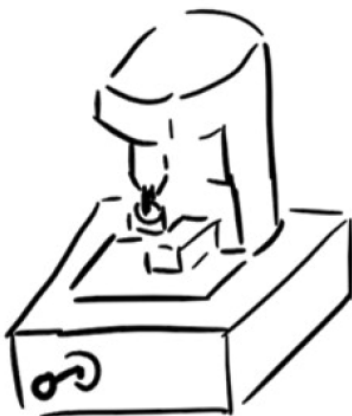
ss1	ss2	ss3	ss4	ss5	ss6	ss7	ss8	ss9	ss10	ss11	ss12	ss13	ss14	ss15	ss16	ss17	ss18	ss19	ss20	ss21	ss22	ss23	ss24	ss25	ss26	ss27	ss28	ss29	ss30	ss31	ss32	ss33	ss34	ss35	ss36	ss37	ss38	ss39	ss40	ss41	ss42	ss43	ss44	ss45	ss46	ss47	ss48	ss49	ss50	ss51	ss52	ss53	ss54	ss55	ss56	ss57	ss58	ss59	ss60	ss61	ss62	ss63	ss64	ss65	ss66	ss67	ss68	ss69	ss70	ss71	ss72	ss73	ss74	ss75	ss76	ss77	ss78	ss79	ss80	ss81	ss82	ss83	ss84	ss85	ss86	ss87	ss88	ss89	ss90	ss91	ss92	ss93	ss94	ss95	ss96	ss97	ss98	ss99	ss100	ss101	ss102	ss103	ss104	ss105	ss106	ss107	ss108	ss109	ss110	ss111	ss112	ss113	ss114	ss115	ss116	ss117	ss118	ss119	ss120	ss121	ss122	ss123	ss124	ss125	ss126	ss127	ss128	ss129	ss130	ss131	ss132	ss133	ss134	ss135	ss136	ss137	ss138	ss139	ss140	ss141	ss142	ss143	ss144	ss145	ss146	ss147	ss148	ss149	ss150	ss151	ss152	ss153	ss154	ss155	ss156	ss157	ss158	ss159	ss160	ss161	ss162	ss163	ss164	ss165	ss166	ss167	ss168	ss169	ss170	ss171	ss172	ss173	ss174	ss175	ss176	ss177	ss178	ss179	ss180	ss181	ss182	ss183	ss184	ss185	ss186	ss187	ss188	ss189	ss190	ss191	ss192	ss193	ss194	ss195	ss196	ss197	ss198	ss199	ss200	ss201	ss202	ss203	ss204	ss205	ss206	ss207	ss208	ss209	ss210	ss211	ss212	ss213	ss214	ss215	ss216	ss217	ss218	ss219	ss220	ss221	ss222	ss223	ss224	ss225	ss226	ss227	ss228	ss229	ss230	ss231	ss232	ss233	ss234	ss235	ss236	ss237	ss238	ss239	ss240	ss241	ss242	ss243	ss244	ss245	ss246	ss247	ss248	ss249	ss250	ss251	ss252	ss253	ss254	ss255	ss256	ss257	ss258	ss259	ss260	ss261	ss262	ss263	ss264	ss265	ss266	ss267	ss268	ss269	ss270	ss271	ss272	ss273	ss274	ss275	ss276	ss277	ss278	ss279	ss280	ss281	ss282	ss283	ss284	ss285	ss286	ss287	ss288	ss289	ss290	ss291	ss292	ss293	ss294	ss295	ss296	ss297	ss298	ss299	ss300	ss301	ss302	ss303	ss304	ss305	ss306	ss307	ss308	ss309	ss310	ss311	ss312	ss313	ss314	ss315	ss316	ss317	ss318	ss319	ss320	ss321	ss322	ss323	ss324	ss325	ss326	ss327	ss328	ss329	ss330	ss331	ss332	ss333	ss334	ss335	ss336	ss337	ss338	ss339	ss340	ss341	ss342	ss343	ss344	ss345	ss346	ss347	ss348	ss349	ss350	ss351	ss352	ss353	ss354	ss355	ss356	ss357	ss358	ss359	ss360	ss361	ss362	ss363	ss364	ss365	ss366	ss367	ss368	ss369	ss370	ss371	ss372	ss373	ss374	ss375	ss376	ss377	ss378	ss379	ss380	ss381	ss382	ss383	ss384	ss385	ss386	ss387	ss388	ss389	ss390	ss391	ss392	ss393	ss394	ss395	ss396	ss397	ss398	ss399	ss400	ss401	ss402	ss403	ss404	ss405	ss406	ss407	ss408	ss409	ss410	ss411	ss412	ss413	ss414	ss415	ss416	ss417	ss418	ss419	ss420	ss421	ss422	ss423	ss424	ss425	ss426	ss427	ss428	ss429	ss430	ss431	ss432	ss433	ss434	ss435	ss436	ss437	ss438	ss439	ss440	ss441	ss442	ss443	ss444	ss445	ss446	ss447	ss448	ss449	ss450	ss451	ss452	ss453	ss454	ss455	ss456	ss457	ss458	ss459	ss460	ss461	ss462	ss463	ss464	ss465	ss466	ss467	ss468	ss469	ss470	ss471	ss472	ss473	ss474	ss475	ss476	ss477	ss478	ss479	ss480	ss481	ss482	ss483	ss484	ss485	ss486	ss487	ss488	ss489	ss490	ss491	ss492	ss493	ss494	ss495	ss496	ss497	ss498	ss499	ss500	ss501	ss502	ss503	ss504	ss505	ss506	ss507	ss508	ss509	ss510	ss511	ss512	ss513	ss514	ss515	ss516	ss517	ss518	ss519	ss520	ss521	ss522	ss523	ss524	ss525	ss526	ss527	ss528	ss529	ss530	ss531	ss532	ss533	ss534	ss535	ss536	ss537	ss538	ss539	ss540	ss541	ss542	ss543	ss544	ss545	ss546	ss547	ss548	ss549	ss550	ss551	ss552	ss553	ss554	ss555	ss556	ss557	ss558	ss559	ss560	ss561	ss562	ss563	ss564	ss565	ss566	ss567	ss568	ss569	ss570	ss571	ss572	ss573	ss574	ss575	ss576	ss577	ss578	ss579	ss580	ss581	ss582	ss583	ss584	ss585	ss586	ss587	ss588	ss589	ss590	ss591	ss592	ss593	ss594	ss595	ss596	ss597	ss598	ss599	ss600	ss601	ss602	ss603	ss604	ss605	ss606	ss607	ss608	ss609	ss610	ss611	ss612	ss613	ss614	ss615	ss616	ss617	ss618	ss619	ss620	ss621	ss622	ss623	ss624	ss625	ss626	ss627	ss628	ss629	ss630	ss631	ss632	ss633	ss634	ss635	ss636	ss637	ss638	ss639	ss640	ss641	ss642	ss643	ss644	ss645	ss646	ss647	ss648	ss649	ss650	ss651	ss652	ss653	ss654	ss655	ss656	ss657	ss658	ss659	ss660	ss661	ss662	ss663	ss664	ss665	ss666	ss667	ss668	ss669	ss670	ss671	ss672	ss673	ss674	ss675	ss676	ss677	ss678	ss679	ss680	ss681	ss682	ss683	ss684	ss685	ss686	ss687	ss688	ss689	ss690	ss691	ss692	ss693	ss694	ss695	ss696	ss697	ss698	ss699	ss700	ss701	ss702	ss703	ss704	ss705	ss706	ss707	ss708	ss709	ss710	ss711	ss712	ss713	ss714	ss715	ss716	ss717	ss718	ss719	ss720	ss721	ss722	ss723	ss724	ss725	ss726	ss727	ss728	ss729	ss730	ss731	ss732	ss733	ss734	ss735	ss736	ss737	ss738	ss739	ss740	ss741	ss742	ss743	ss744	ss745	ss746	ss747	ss748	ss749	ss750	ss751	ss752	ss753	ss754	ss755	ss756	ss757	ss758	ss759	ss760	ss761	ss762	ss763	ss764	ss765	ss766	ss767	ss768	ss769	ss770	ss771	ss772	ss773	ss774	ss775	ss776	ss777	ss778	ss779	ss780	ss781	ss782	ss783	ss784	ss785	ss786	ss787	ss788	ss789	ss790	ss791	ss792	ss793	ss794	ss795	ss796	ss797	ss798	ss799	ss800	ss801	ss802	ss803	ss804	ss805	ss806	ss807	ss808	ss809	ss810	ss811	ss812	ss813	ss814	ss815	ss816	ss817	ss818	ss819	ss820	ss821	ss822	ss823	ss824	ss825	ss826	ss827	ss828	ss829	ss830	ss831	ss832	ss833	ss834	ss835	ss836	ss837	ss838	ss839	ss840	ss841	ss842	ss843	ss844	ss845	ss846	ss847	ss848	ss849	ss850	ss851	ss852	ss853	ss854	ss855	ss856	ss857	ss858	ss859	ss860	ss861	ss862	ss863	ss864	ss865	ss866	ss867	ss868	ss869	ss870	ss871	ss872	ss873	ss874	ss875	ss876	ss877	ss878	ss879	ss880	ss881	ss882	ss883	ss884	ss885	ss886	ss887	ss888	ss889	ss890	ss891	ss892	ss893	ss894	ss895	ss896	ss897	ss898	ss899	ss900	ss901	ss902	ss903	ss904	ss905	ss906	ss907	ss908	ss909	ss910	ss911	ss912	ss913	ss914	ss915	ss916	ss917	ss918	ss919	ss920	ss921	ss922	ss923	ss924	ss925	ss926	ss927	ss928	ss929	ss930	ss931	ss932	ss933	ss934	ss935	ss936	ss937	ss938	ss939	ss940	ss941	ss942	ss943	ss944	ss945	ss946	ss947	ss948	ss949	ss950	ss951	ss952	ss953	ss954	ss955	ss956	ss957	ss958	ss959	ss960	ss961	ss962	ss963	ss964	ss965	ss966	ss967	ss968	ss969	ss970	ss971	ss972	ss973	ss974	ss975	ss976	ss977	ss978	ss979	ss980	ss981	ss982	ss983	ss984	ss985	ss986	ss987	ss988	ss989	ss990	ss991	ss992	ss993	ss994	ss995	ss996	ss997	ss998	ss999	ss1000	ss1001	ss1002	ss1003	ss1004	ss1005	ss1006	ss1007	ss1008	ss1009	ss1010	ss1011	ss1012	ss1013	ss1014	ss1015	ss1016	ss1017	ss1018	ss1019	ss1020	ss1021	ss1022	ss1023	ss1024	ss1025	ss1026	ss1027	ss1028	ss1029	ss1030	ss1031	ss1032	ss1033	ss1034	ss1035	ss1036	ss1037	ss1038	ss1039	ss1040	ss1041	ss1042	ss1043	ss1044	ss1045	ss1046	ss1047	ss1048	ss1049	ss1050	ss1051	ss1052	ss1053	ss1054	ss1055	ss1056	ss1057	ss1058	ss1059	ss1060	ss1061	ss1062	ss1063	ss1064	ss1065	ss1066	ss1067	ss1068	ss1069	ss1070	ss1071	ss1072	ss1073	ss1074	ss1075	ss1076	ss1077	ss1078	ss1079	ss1080	ss1081	ss1082	ss1083	ss1084	ss1085	ss1086	ss1087	ss1088	ss1089	ss1090	ss1091	ss1092	ss1093	ss1094	ss1095	ss1096	ss1097	ss1098	ss1099	ss1100	ss1101	ss1102	ss1103	ss1104	ss1105	ss1106	ss1107	ss1108	ss1109	ss1110	ss1111	ss1112	ss1113	ss1114	ss1115	ss1116	ss1117	ss1118	ss1119	ss1120	ss1121	ss1122	ss1123	ss1124	ss1125	ss1126	ss1127	ss1128	ss1129	ss1130	ss1131	ss1132	ss1133	ss1134	ss1135	ss1136	ss1137	ss1138	ss1139	ss1140	ss1141	ss1142	ss1143	ss1144	ss1145	ss1146	ss1147	ss1148	ss1149	ss1150	ss1151	ss1152	ss1153	ss1154	ss1155	ss1156	ss1157	ss1158	ss1159	ss1160	ss1161	ss1162	ss1163	ss1164	ss1165	ss1166	ss1167	ss1168	ss1169	ss1170	ss1171	ss1172	ss1173	ss1174	ss1175	ss1176	ss1177	ss1178	ss1179	ss1180	ss1181	ss1182	ss1183	ss1184	ss1185	ss1186	ss1187	ss1188	ss1189	ss1190	ss1191	ss1192	ss1193	ss1194	ss1195	ss1196	ss1197	ss1198	ss1199	ss1200	ss1201	ss1202	ss1203	ss1204	ss1205	ss1206	ss1207	ss1208	ss1209	ss1210	ss1211	ss1212	ss1213	ss1214	ss1215	ss1216	ss1217	ss1218	ss1219	ss1220	ss1221	ss1222	ss1223	ss1224	ss1225	ss1226	ss1227	ss1228	ss1229	ss1230	ss1231	ss1232	ss1233	ss1234	ss1235	ss1236	ss1237	ss1238	ss1239	ss1240	ss1241	ss1242	ss1243	ss1244	ss1245	ss1246	ss1247	ss1248	ss1249	ss1250	ss1251	ss1252	ss1253	ss1254	ss1255	ss1256	ss1257	ss1258	ss1259	ss1260	ss1261	ss1262	ss1263	ss1264	ss1265	ss1266	ss1267	ss1268	ss1269	ss1270	ss1271	ss1272	ss1273	ss1274	ss1275	ss1276	ss1277	ss1278	ss1279	ss1280	ss1281	ss1282	ss1283	ss1284	ss1285	ss1286	ss1287	ss1288	ss1289	ss1290	ss1291	ss1292	ss1293	ss1294	ss1295	ss1296	ss1297	ss1298	ss1299	ss1300	ss1301	ss1302	ss1303	ss1304	ss1305	ss1306	ss1307	ss1308	ss1309	ss1310	ss1311	ss1312	ss1313
-----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------

Exercise 6.1.3. State Transition Diagram III - Milling Machine

Design a state transition diagram for an automatic milling machine as a Moore machine, under the following conditions:

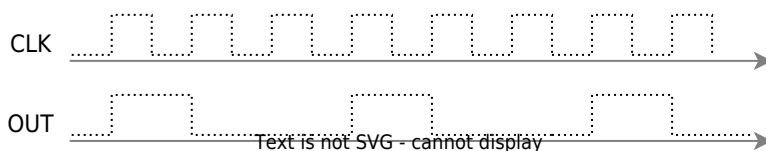
1. The milling machine has 4 states: Initialisation I , Component Change C , Running R , Error E .
2. The milling machine starts at the state I .
3. Once a limit switch L is read as activated, the state E shall be entered, independent of which state the machine was in.
4. Only in the state E an alarm shall ring $A=1$.
5. E can only be exited with a reset.
6. In order to change from Component Change C to Running R , the user has to activate a Key $K=1$.
7. Only when the machine is running R and the Fixed Condition is reached $F=1$, the state Component Change C shall be entered.
8. Initialisation I only changes into Component Change C , when no limit switch L is activated and the Key is deactivated (input $K=0$) and the Fixed Condition is reached $F=1$.

Explicitly draw all possible transitions.



Exercise 6.1.4. State Transition Diagram IV - Find a sequence

Develop a sequential circuit, which creates the following output



- Draw the state transition diagram of the Moore machine of the synchronous sequential circuit.
- Create the digital circuit.

Exercise 6.1.5. State Transition Diagram V - LED Fun

Develop a sequential circuit, which allows driving the following LED sequence

Electric Setup

- Draw the state transition diagram of the Moore machine of the synchronous sequential circuit.
- Create the digital circuit.

Exercise 6.1.6. State Transition Diagram VI - Roll the dice

Develop a sequential circuit, which generates a clock-driven up-counter from \$1..6\$. The not required states shall lead after one clock cycle to the state of number \$1\$.

- Draw the state transition diagram of the Moore machine of the synchronous sequential circuit.
- Create the digital circuit.

From:

<https://wiki.mexle.te.hs-heilbronn.de/> - MEXLE Wiki

Permanent link:

https://wiki.mexle.te.hs-heilbronn.de/introduction_to_digital_systems/sequential_logic?rev=1733829266

Last update: 2024/12/10 12:14

