Sistemas Digitais I

LESI - 2º ano

Lesson 7 - Sequential Systems Principles

Prof. João Miguel Fernandes
(miguel@di.uminho.pt)

Dept. Informática

7. Sequential Circuits

- Combinational vs. Sequential Circuits -

- Logic circuits are classified as combinational or sequential.
- A combinational circuit is one whose outputs depend only on its current inputs. Example: TV channel selector.
- A sequential circuit is one whose outputs depend on its current inputs, but also on the past sequence of inputs. Example: TV channel selector with channel up/down buttons.
- It is impossible to describe the behaviour of a sequential circuit by means of a table that relates inputs with outputs.
- To know where to go next, we need to know where we are now.
- The state of the system must be memorised.

7. Sequential Circuits

- State (1) -

- The state of a sequential circuit is a collection of state variables whose values contain all the information about the past necessary to account for the circuit's future behaviour.
- In the TV channel example, the current channel number is the current state.
- Given the current state, we can always predict the next state as a function of the inputs.
- In a digital circuit, state variables are binary values.
- A circuit with n binary state variables has 2^n possible states.
- Sequential circuits are also called finite-state machines.

7. Sequential Circuits

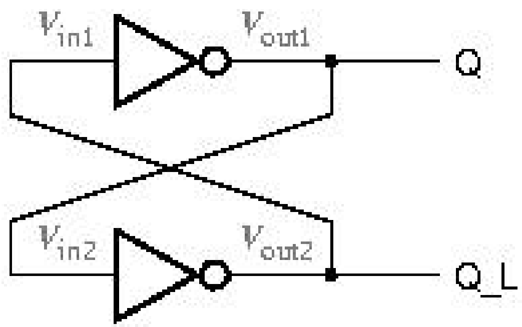
- State (2) -

- The state changes occur at times specified by a clock signal.
- A clock signal is active high if state changes occur at the clock's rising edge or when the clock is HIGH. Otherwise, it is active low.
- The clock period (T) is the time between successive transitions in the same direction.
- The clock frequency (f) is the reciprocal of the clock period (f=1/T).
- Two types of sequential circuits:
  - **Feedback sequential circuits use ordinary gates and feedback loops to obtain memory elements (latches and flip-flops).**
  - **Clocked synchronous state machines use latches and flip-flops to create circuits that are regulated by a controlling clock signal.**

7. Sequential Circuits

- Bistable Elements (1) -

- The simplest sequential circuit consists of a pair of inverters forming a feedback loop.
- The circuit is called a <u>bistable</u>, since a digital analysis shows that it has two stable states.

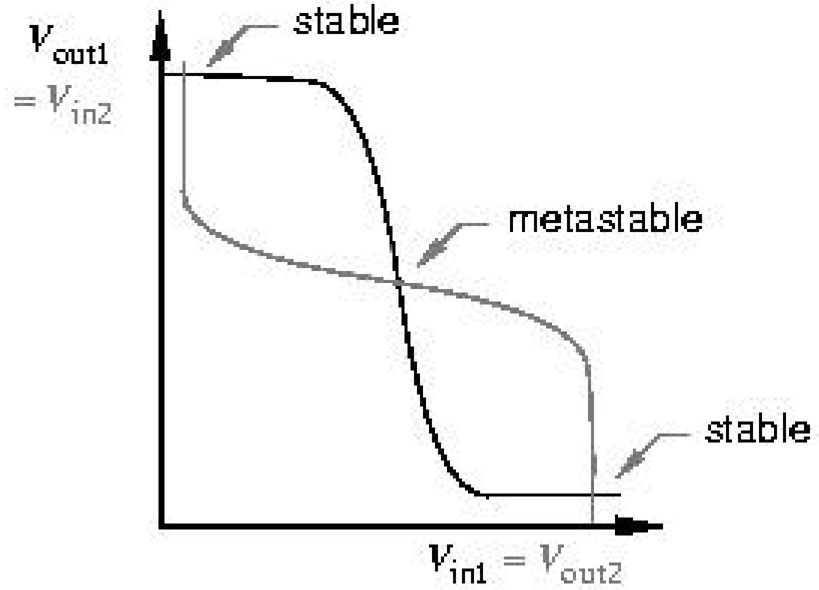


- If Q is HIGH, the bottom inverter has a LOW output, which forces the top inverter to produce a HIGH output (as assumed initially).
- If Q is LOW, the bottom inverter has a HIGH output, which forces the top inverter to produce a LOW output (as assumed initially).
- We can use a single state variable (signal Q) to describe the state of the circuit. There are 2 possible states, Q=0 and Q=1.

7. Sequential Circuits

- Bistable Elements (2) -

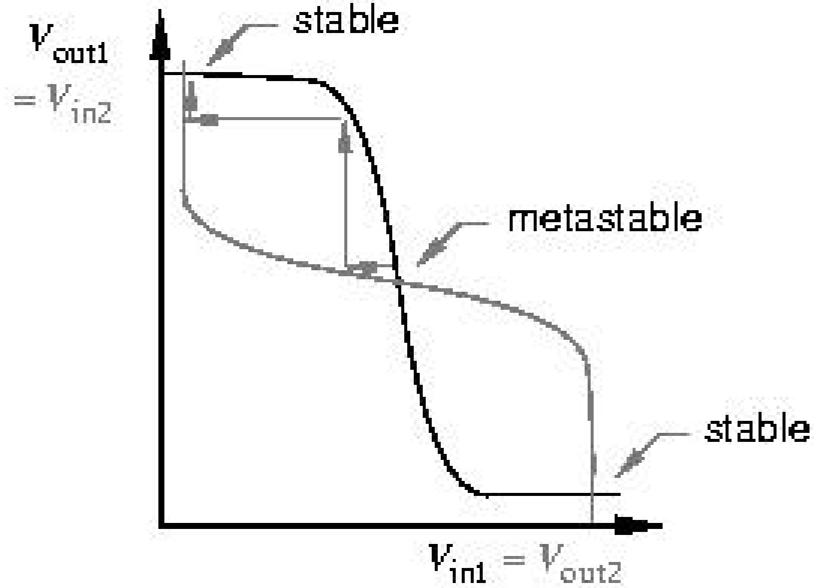
- The bistable element is so simple that it has no inputs, so its state cannot be controlled.
- When power is applied to the circuit, it randomly comes up in one state and stays there forever.
- The analysis of the bistable from an analog perspective shows more aspects.
- The bistable is in equilibrium if the input and output voltages of both inverters are constant values consistent with the loop connections and the transfer functions.

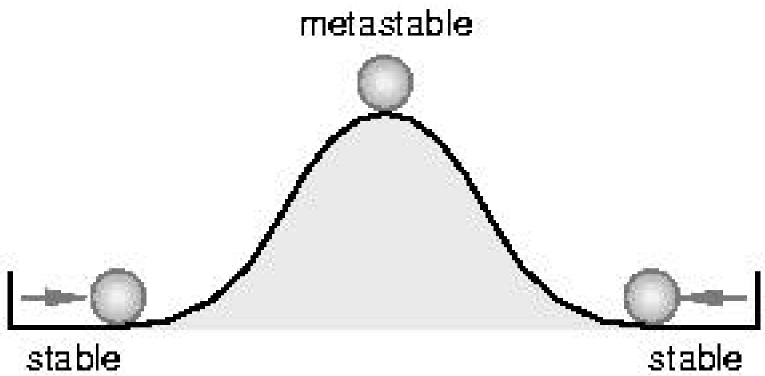


7. Sequential Circuits

- Bistable Elements (3) -

- The bistable is in equilibrium at the points marked "stable".
- The third equilibrium point, labelled "metastable", occurs when V_{out1} and V_{out2} have no valid logic values.
- If the circuit operates at the metastable point, it could stay there indefinitely.
- The point is METAstable, because random noise will tend to drive the circuit toward one of the stable points.
- Ball and hill analogy for metastable point.





7. Sequential Circuits

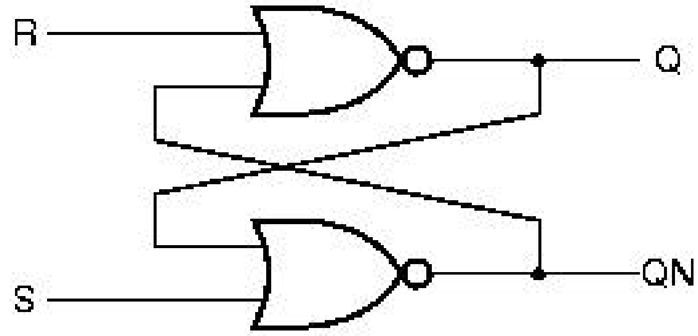
- Latches and Flip-flops (1) -

- Latches and flips-flops are the basic building blocks of most sequential circuits.
- A flip-flop is a sequential device that samples its inputs and changes its outputs only at times determined by a clocking signal.
- A latch is a sequential device that watches all of its inputs continuously and changes its outputs at any time.

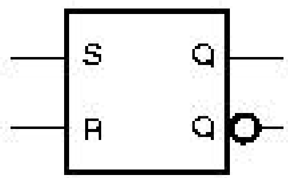
7. Sequential Circuits

- S-R Latches (1) -

- An S-R latch can be built with NOR gates.
- QN is usually the complement of Q.
- If S and R are both 0, the circuit behaves like the bistable element.
- Either S or R may be asserted to force the feedback loop to a desired state.
- S <u>sets</u> or presets the Q output to 1.
- R <u>resets</u> or clears the Q output to 0.



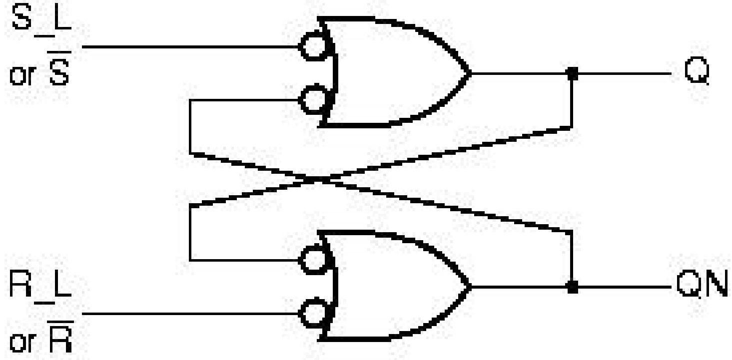
S	R	Q	QN
0	0	last Q	last QN
0	1	0	1
1	0	1	0
1	1	0	0



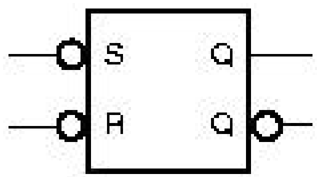
7. Sequential Circuits

- S-R Latches (2) -

- An S-R latch with active-low set and reset inputs may be built with NAND gates.
- The operation of this latch is similar to the previous one, with two major differences.
- First, S_L and R_L are active low, so the latch remembers its state, when S=R=1.
- Second, when S_L and R_L are both asserted, both outputs go to 1 (not 0).



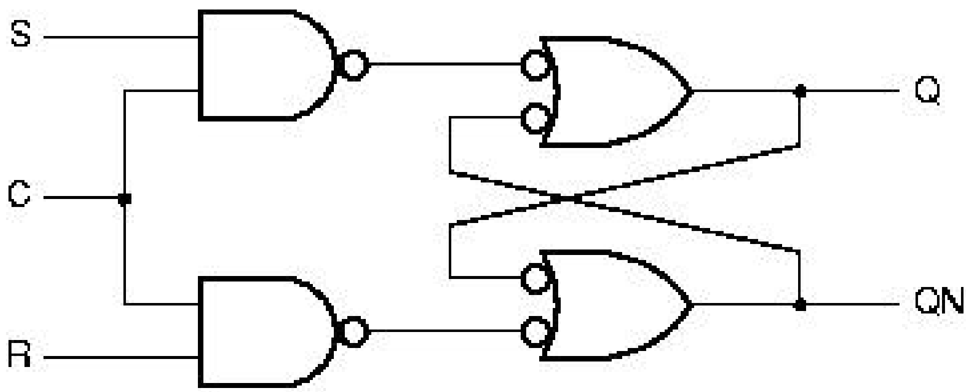
S_L	R_L	Q	QN
0	0	1	1
0	1	1	0
1	0	0	1
1	1	last Q	last QN



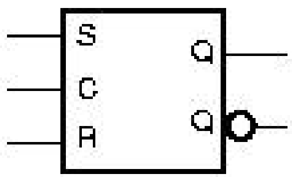
7. Sequential Circuits

- S-R Latches (3) -

- An S-R latch is sensitive to its inputs at all times.
- It may be modified to be sensitive to these inputs only when an enabling input C is asserted.
- The circuit behaves like an S-R latch when C=1.
- It retains its state when C=0.



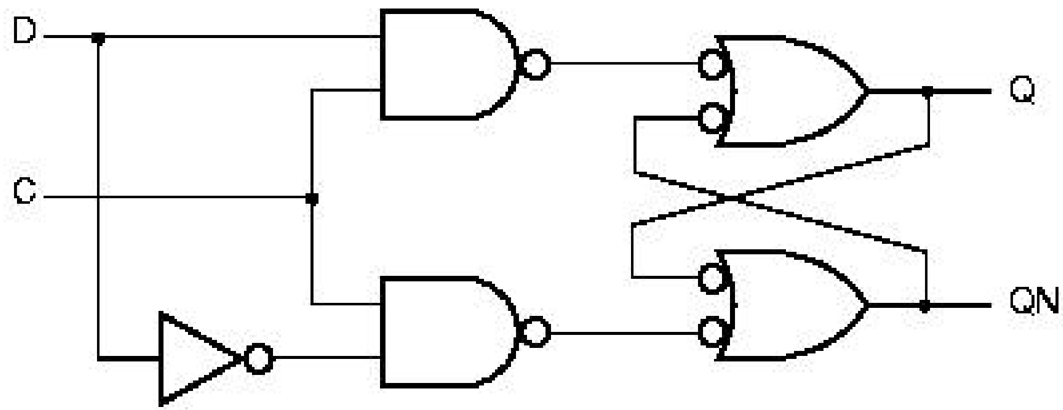
S	R	C	Q	QN
0	0	1	last Q	last QN
0	1	1	0	1
1	0	1	1	0
1	1	1	1	1
x	x	0	last Q	last QN



7. Sequential Circuits

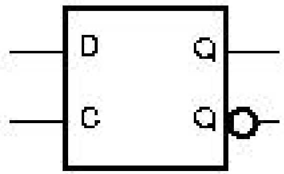
- D Latches (1) -

- Latches are needed to store bits of information.
- A D latch can be used for that purpose.
- The D latch can be built from an S-R latch.



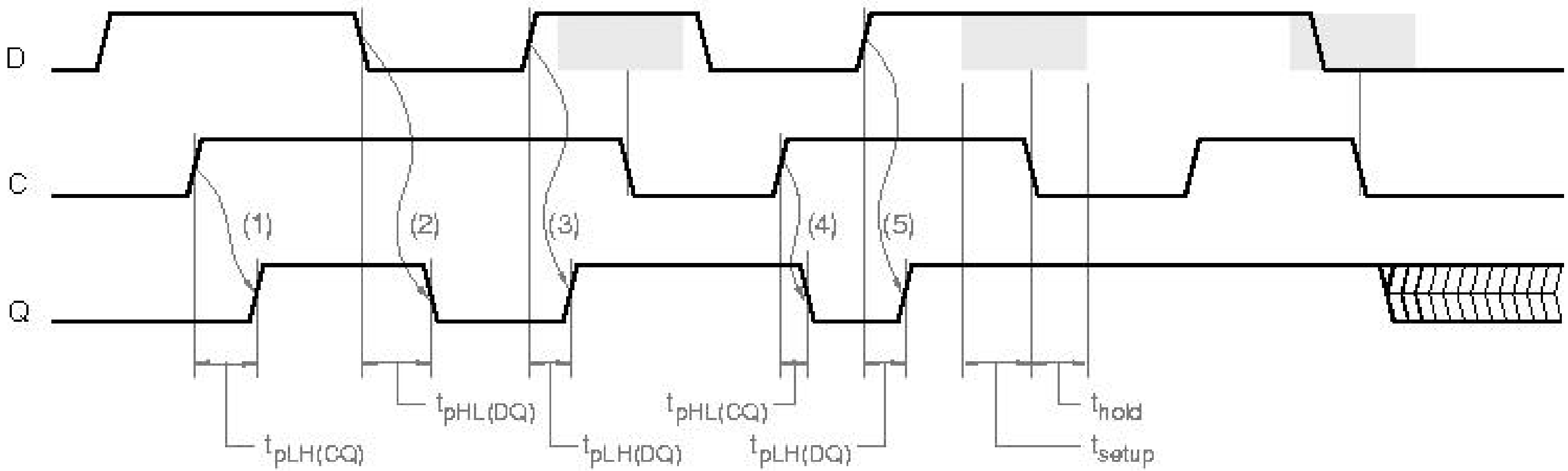
C	D	Q	QN
1	0	0	1
1	1	1	0
0	x	last Q	last QN

- This latch eliminates the troublesome situation in S-R latches, where S and R may be asserted simultaneously.
- When C=1, the latch is <u>open</u> and the Q output follows the D input. When C=0, the latch is <u>closed</u>.



7. Sequential Circuits

- D Latches (2) -

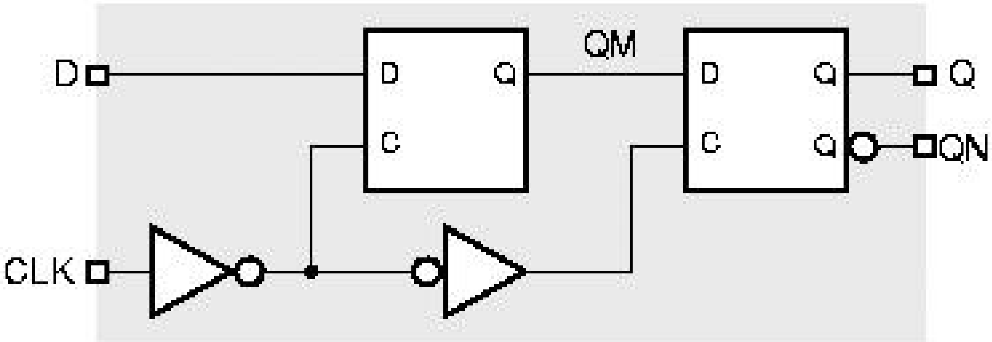


- Delays exist for signals that propagate from the inputs to the Q output.
- There is a window of time (<u>setup time</u> and <u>hold time</u>) around the falling edge of C when the D input must not change.
- The latch's output is unpredictable, if those times are not respected.

7. Sequential Circuits

- D Flip-flops (1) -

- A positive-edge-triggered D flip-flop combines a pair of D latches to create a circuit that samples its D input and changes its outputs only at the rising edge of the CLK signal.



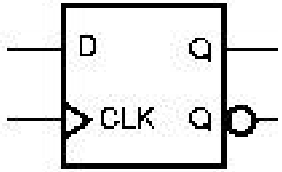
D	CLK	Q	QN
0	↑	0	1
1	↑	1	0
x	0	last Q	last QN
x	1	last Q	last QN

- The first latch is called the master and it is open when CLK=0.
- When CLK goes to 1, the master latch is closed.
- The second latch, the slave, is open while CLK=1, but changes only at the begin of the interval, because the master is closed.

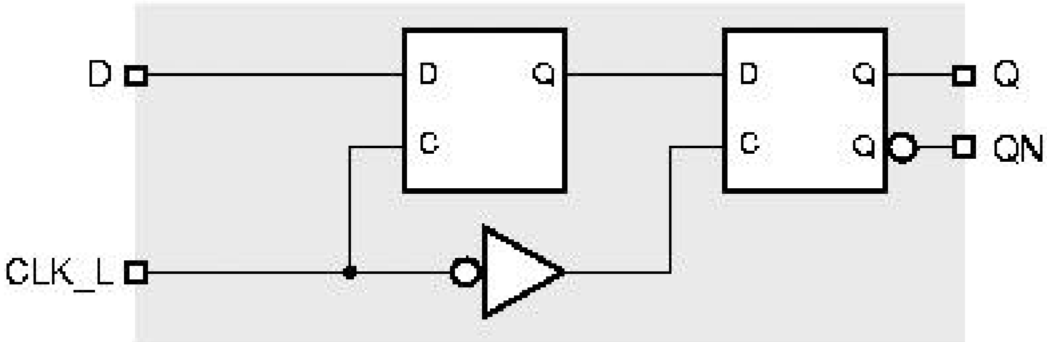
7. Sequential Circuits

- D Flip-flops (2) -

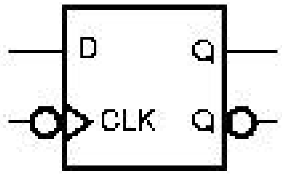
- The triangle on the CLK input is a <u>dynamic-input indicator</u> and indicates edge-triggered behaviour.



- A <u>negative-edge-triggered D flip-flop</u> simply inverts the clock input and actions occur on the falling edge of the clock signal.

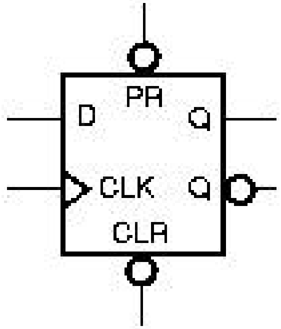


D	CLK_L	Q	QN
0	↓	0	1
1	↓	1	0
x	0	last Q	last QN
x	1	last Q	last QN

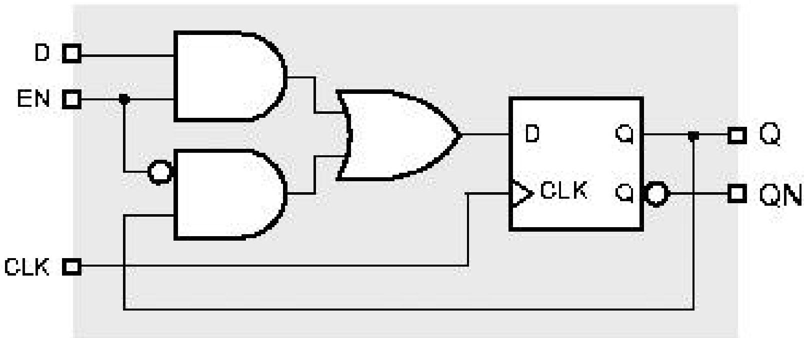


7. Sequential Circuits

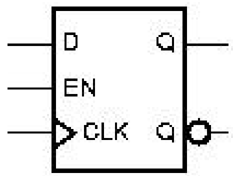
- D Flip-flop (3) -



- Some D flip-flops have asynchronous inputs that are used to force its state, independent of the CLK and D inputs.
- These inputs (PR and CLR) behave like the set and reset inputs on an S-R latch.
- They should be used for initialisation and testing purposes.
- Some D flip-flops have the possibility to hold the last value stored. This is accomplished by adding an <u>enable</u> input.



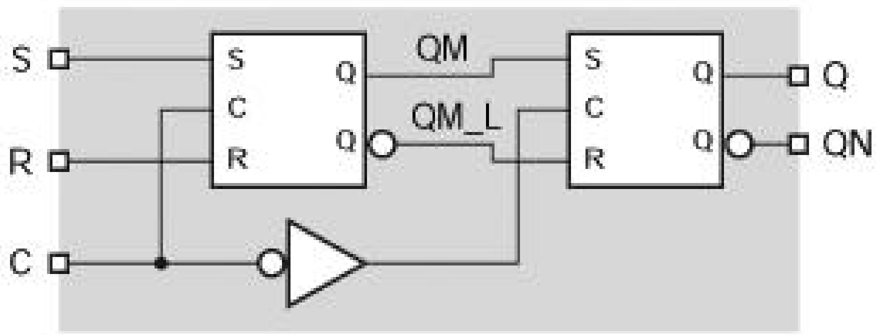
D	EN	CLK	Q	QN
0	1	↑	0	1
1	1	↑	1	0
x	0	↑	last Q	last QN
x	x	0	last Q	last QN
x	x	1	last Q	last QN



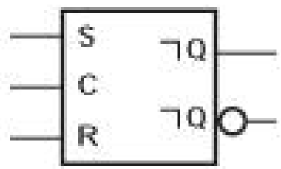
7. Sequential Circuits

- S-R Flip-flops (1) -

- S-R latches are useful for in control applications, where we may have independent conditions for setting/resetting a control bit.
- If the control bit is supposed to change only at certain times with respect to a clock signal, we need an S-R flip-flop.



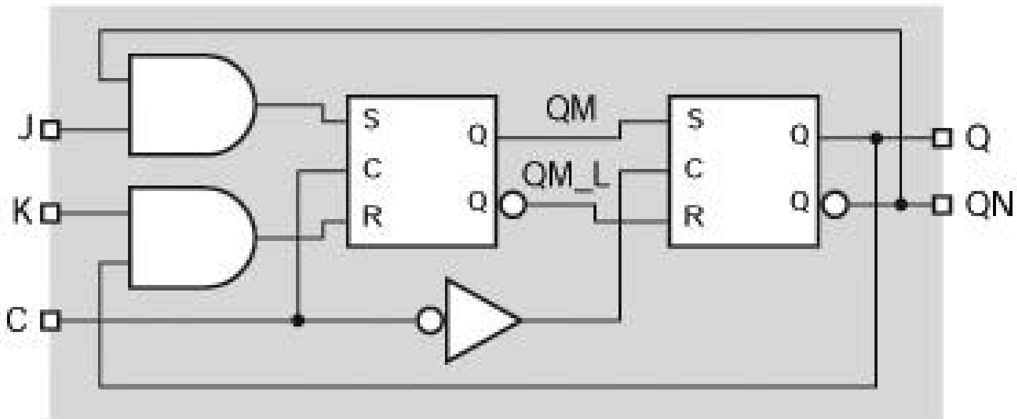
S	R	C	Q	QN
x	x	0	last Q	last QN
0	0	⎍	last Q	last QN
0	1	⎍	0	1
1	0	⎍	1	0
1	1	⎍	undef.	undef.



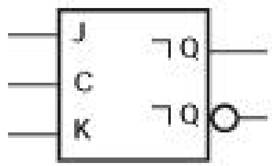
7. Sequential Circuits

- J-K Flip-flops (1) -

- The problem of what to do when S and R are both asserted is solved in a master/slave J-K flip-flop.
- The J and K inputs are analogous to S and R.
- However, asserting J asserts the master’s S input only if Q=0.
- Asserting K asserts the master’s R input only if Q=1.
- Thus, if J and K are asserted simultaneously, the flip-flop goes to the opposite of its current state.



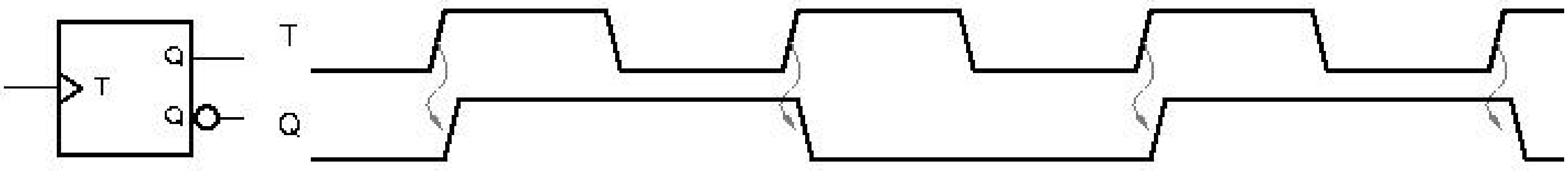
J	K	C	Q	QN
x	x	0	last Q	last QN
0	0	⎍	last Q	last QN
0	1	⎍	0	1
1	0	⎍	1	0
1	1	⎍	last QN	last Q



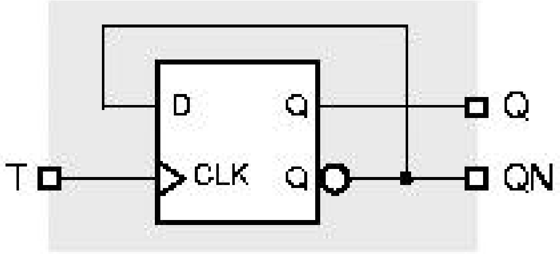
7. Sequential Circuits

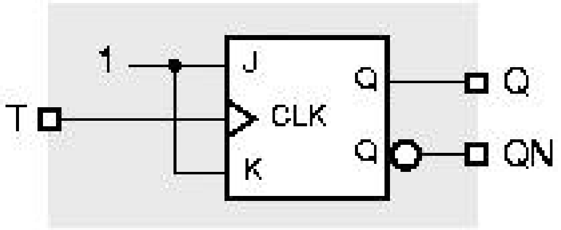
- T Flip-flops (1) -

- A T flip-flop changes state every tick of the clock.



- The signal on the flip-flop's Q output has half the frequency of the T input.
- D and J-K flip-flops can be used to build a T flip-flop.

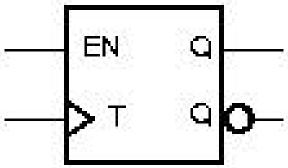


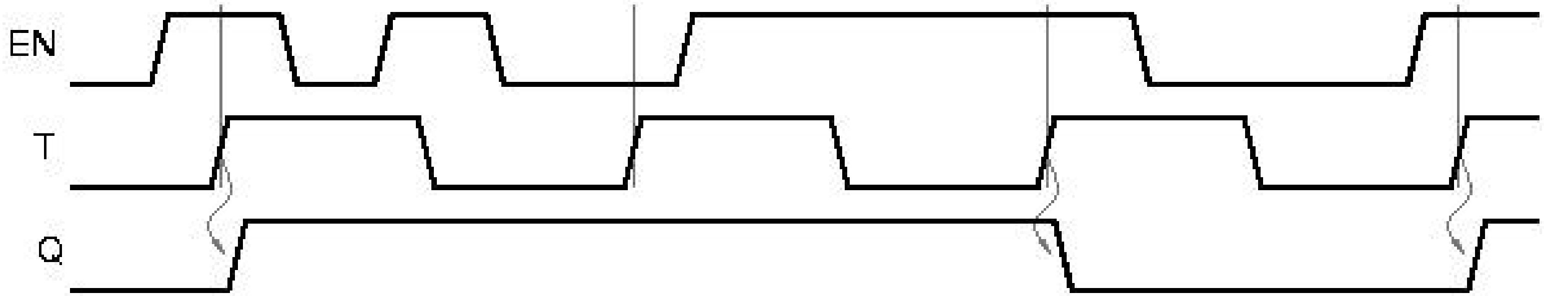


7. Sequential Circuits

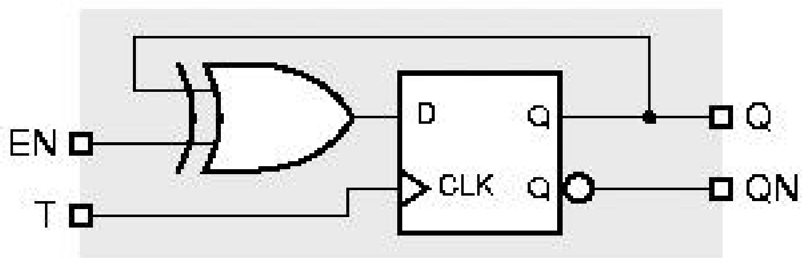
- T Flip-flops (2) -

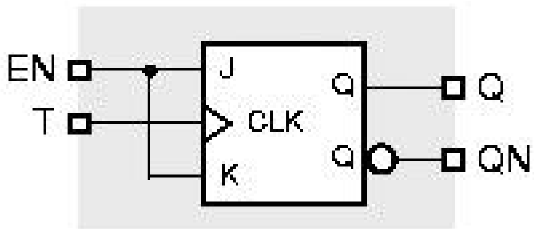
- A T flip-flop can have an enable input.
- The flip-flop changes state at the triggering edge of the clock, only if the enable signal EN is asserted.





- D and J-K flip-flops can be used to build a T flip-flop with enable.





7. Sequential Circuits

- State Machine Design (1) -

- A <u>finite state machine</u> (FSM) can be formally defined as the quintuple `<S,I,O,F,G>`, where:
  - `S` **represents the set of states.**
  - `I` **represents the set of inputs.**
  - `O` **represents the set of outputs.**
  - `F` **represents the next-state function.**
  - `G` **represents the output function.**
- The F function assigns to every pair of state and input combination another state ($F:S \times I \rightarrow S$).
- The G function determines the output values in the present state.

7. Sequential Circuits

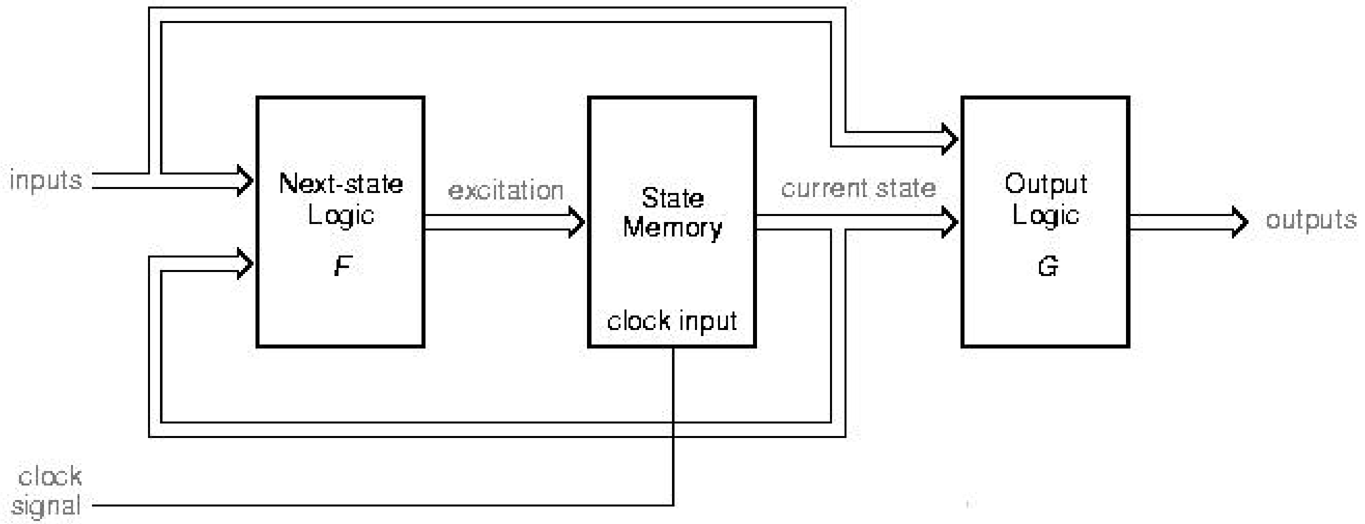
- State Machine Design (2) -

- There are 2 types of FSMs, which correspond to 2 different definitions of the output function G.
- For the Moore type, the G function is state-based ($G:S \rightarrow O$).
- An output symbol is assigned to each state of the FSM.
- For the Mealy type, the G function is input-based ($G:S \times I \rightarrow O$).
- An output symbol is defined by a pair of state and input symbol.
- The FSM model assumes that time is divided into uniform intervals and that transitions occur only at the beginning of each time interval.
- There is a clock signal that defines the time intervals, called <u>clock cycles</u>.
- Each FSM model can be implemented with flip-flops and logic gates.

7. Sequential Circuits

- State Machine Design (3) -

- General structure of a clocked synchronous Mealy State Machine:

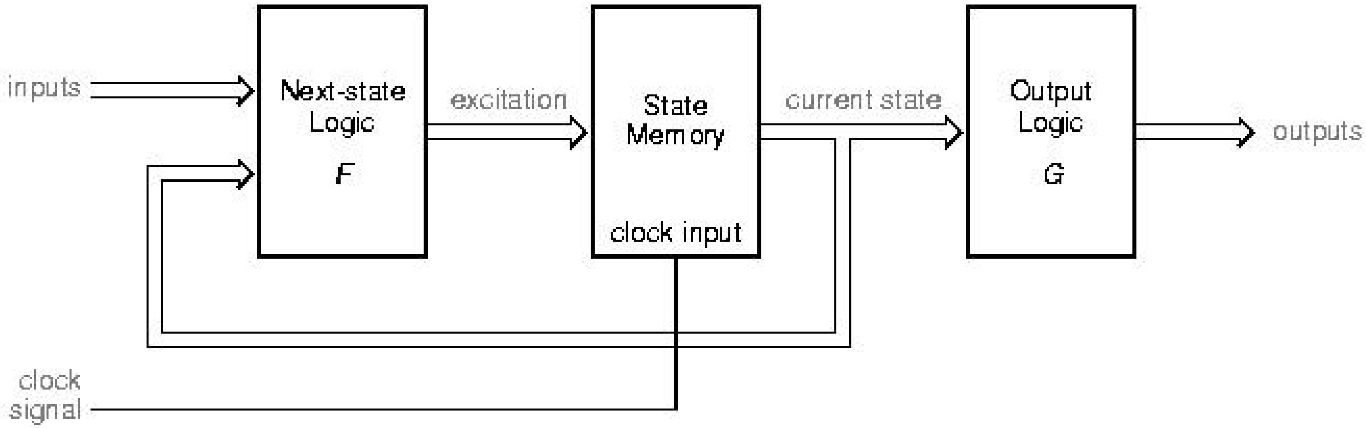


- The State Memory is a set of flip-flops that store the current state of the machine. The flip-flops are connected to a common clock signal.
- Both F and G are strictly combinational circuits.

7. Sequential Circuits

- State Machine Design (4) -

- General structure of a clocked synchronous <u>Moore state machine</u>:



- The only difference between Mealy and Moore machines is in how the outputs are generated.
- To simplify the G block to just wires, we can use the <u>output-coded state assignment</u>, where the state variables serve as outputs.

7. Sequential Circuits

- State Machine Design (5) -

- The steps to design a clocked synchronous state machine are:
  - **Read the natural language description or specification of the system.**
  - **Draw a state diagram, using mnemonic names for the states.**
  - **Construct a state/output table.**
  - **(Optional) Minimise the number of states in the table.**
  - **Choose a set of state variables and assign state combinations to each state.**
  - **Substitute the state names for the corresponding state combinations in the table.**
  - **Choose a flip-flop type for the state memory.**
  - **Construct an excitation table that shows the excitation values required to obtain the desired next state for each state/input combination.**
  - **Derive excitation equations.**
  - **Derive output equations.**

7. Sequential Circuits

- State Machine Design (6) -

- Example of a state machine problem:
  Design a state machine with inputs A and B, and output Z that is 1 if:
  - **A had the same value at each of the two previous clock ticks, or**
  - **B has been 1 since the last time that the first condition was true.**

  Otherwise, the Z output is 0.
- Right now, the meaning of the specification may not be clear.
- The designer has to transform an ambiguous specifications written in natural language into an unambiguous <u>state table</u>.
- The machine is of Moore type, since the output depends only on the current state, that is, what happened in previous clock periods.

7. Sequential Circuits

- State Machine Design (7) -

Meaning	S	AB = 00	01	11	10	Z
Initial state	INIT					0
	.....					
	.....					
	.....					
		S*				

Meaning	S	AB = 00	01	11	10	Z
Initial state	INIT	A0	A0	A1	A1	0
Got a 0 on A	A0					0
Got a 1 on A	A1					0
		S*				

Meaning	S	AB = 00	01	11	10	Z
Initial state	INIT	A0	A0	A1	A1	0
Got a 0 on A	A0	OK	OK	A1	A1	0
Got a 1 on A	A1					0
Got two equal A inputs	OK					1
		S*				

Meaning	S	AB = 00	01	11	10	Z
Initial state	INIT	A0	A0	A1	A1	0
Got a 0 on A	A0	OK	OK	A1	A1	0
Got a 1 on A	A1	A0	A0	OK	OK	0
Got two equal A inputs	OK					1
		S*				

7. Sequential Circuits

- State Machine Design (8) -

Meaning	S	AB 00	01	11	10	Z
Initial state	INIT	A0	A0	A1	A1	0
Got a 0 on A	A0	OK	OK	A1	A1	0
Got a 1 on A	A1	A0	A0	OK	OK	0
Got two equal A inputs	OK	?	OK	OK	?	1
		S*				

Meaning	S	AB 00	01	11	10	Z
Initial state	INIT	A0	A0	A1	A1	0
Got a 0 on A	A0	OK0	OK0	A1	A1	0
Got a 1 on A	A1	A0	A0	OK1	OK1	0
Two equal, A=0 last	OK0					1
Two equal, A=1 last	OK1					1
		S*				

Meaning	S	AB 00	01	11	10	Z
Initial state	INIT	A0	A0	A1	A1	0
Got a 0 on A	A0	OK0	OK0	A1	A1	0
Got a 1 on A	A1	A0	A0	OK1	OK1	0
Two equal, A=0 last	OK0	OK0	OK0	OK1	A1	1
Two equal, A=1 last	OK1					1
		S*				

Meaning	S	AB 00	01	11	10	Z
Initial state	INIT	A0	A0	A1	A1	0
Got a 0 on A	A0	OK0	OK0	A1	A1	0
Got a 1 on A	A1	A0	A0	OK1	OK1	0
Two equal, A=0 last	OK0	OK0	OK0	OK1	A1	1
Two equal, A=1 last	OK1	A0	OK0	OK1	OK1	1
		S*				

7. Sequential Circuits

- State Machine Design (9) -

- The next step is to determine how many binary variables are needed to represent the states in the state table.
- After that, specific combinations are assigned to each state.
- The binary combination of state variables assigned to a particular state is a coded state.
- With n flip-flops, 2^n states can be coded.
- The number of flip-flops needed to code s states is $\lceil \log_2 s \rceil$.
- In our problem, there are 5 states, so 3 flip-flops are required.

	AB				
S	00	01	11	10	Z
INIT	A0	A0	A1	A1	0
A0	OK0	OK0	A1	A1	0
A1	A0	A0	OK1	OK1	0
OK0	OK0	OK0	OK1	A1	1
OK1	A0	OK0	OK1	OK1	1
	S*				

7. Sequential Circuits

- State Machine Design (10) -

- There are several alternatives to code the 5 states.

State Name	Assignment Simplest Q1–Q3	Decomposed Q1–Q3	One-hot Q1–Q5	Almost One-hot Q1–Q4
INIT	000	000	00001	0000
A0	001	100	00010	0001
A1	010	101	00100	0010
OK0	011	110	01000	0100
OK1	100	111	10000	1000

- The simplest assignment of s coded states is to use the first s binary integers in binary counting order.
- This assignment does not always lead to the simplest excitation equations, output equations and resulting logic circuit.

7. Sequential Circuits

- State Machine Design (11) -

- The state assignment has a major impact on circuit cost.
- It may interact with other factors, such as the choice of storage elements and the realisation approach for excitation and output logic.
- How to choose the best state assignment for a given problem?
- In general, the only formal way to find the “BEST” assignment is to try ALL the assignments.
- That is not possible to do by hand!!! For our example, there are 6.720 different ways to assign the 3-bit combinations to the 5 states.
- Designers must rely on practical guidelines to achieve reasonable state assignments.

7. Sequential Circuits

- State Machine Design (12) -

- Guidelines for state assignment:
  - **Choose an initial coded state into which the machine can easily be forced at reset (typically, 000...0 or 111...1).**
  - **Minimise the number of state variables that change on each transition.**
  - **Maximise the number of state variables that don't change in a group of related states.**
  - **Exploit symmetries in the problem specification and the corresponding symmetries in the state table. If one state or group means almost the same thing as another, once an assignment is established for the first, a similar assignment (differing in one bit) should be used for the second.**
  - **Decompose the set of state variables into individual bits or fields, where each one has a well defined meaning with respect to the input effects or the output behaviour.**
  - **Consider using more than the minimum number of state variables to make possible a decomposed assignment.**

7. Sequential Circuits

- State Machine Design (13) -

- Some of the previous guidelines were used in the decomposed state assignment.
- INIT is 000, which is easy to force with the asynchronous CLR input of the flip-flops.
- INIT is never re-entered, once the machine is working. Q1 is used to indicate whether or not the actual state is INIT.
- Q2,Q3 are used to distinguished among the other 4 states.
- Q3 gives the previous value of A.
- Q2 indicates that the condition for a 1 output are satisfied in the current state.

State Name	***Decomposed Q1–Q3***
INIT	000
A0	100
A1	101
OK0	110
OK1	111

7. Sequential Circuits

- State Machine Design (14) -

- The <u>one-hot state assignment</u> can be adapted to any state machine.
- This assignment uses more than the minimum number of state variables: it uses 1 bit per state.
- This usually leads to small excitation equations, since each flip-flop must be set to 1 for transitions into only one state.

State Name	*One-hot Q1–Q5*	*Almost One-hot Q1–Q4*
INIT	00001	0000
A0	00010	0001
A1	00100	0010
OK0	01000	0100
OK1	10000	1000

- The <u>almost one hot assignment</u> uses the no-hot combination for the initial state.
- This eases the reset of the machine, since the initial state is 00...0.

7. Sequential Circuits

- State Machine Design (15) -

- There are unused state codes when the number of states is less than the number of state variable combinations.
- How to consider those unused states?
- In a minimal risk approach, it is assumed that the machine may go to an unused state, due to a hardware failure, for example.
- For all the unused states, an explicit transition to a safe state is made.
- In a minimal cost approach, it is assumed that the machine will never enter an unused state.
- The next state entries of the unused states can be marked as “don’t cares”.

7. Sequential Circuits

- State Machine Design (16) -

- A transition table is obtained by substituting the state names by the corresponding code states.
- The transition table shows the next coded state for each combination of current coded state and input.
- For the state machine example, the transition table is obtained by using the decomposed state assignment.
- The next step is to write an excitation table that shows the flip-flop excitation values needed to make the machine go to the desired next coded state.

	A B				
Q1 Q2 Q3	00	01	11	10	Z
000	100	100	101	101	0
100	110	110	101	101	0
101	100	100	111	111	0
110	110	110	111	101	1
111	100	110	111	111	1
	Q1* Q2* Q3*				

7. Sequential Circuits

- State Machine Design (17) -

- The structure and content of the excitation table depend on the type of flip-flop (D, J-K, T, etc.) being used.
- Nowadays, most state-machine designs use D flip-flops, because of their availability in both discrete packages and PLDs, and their ease of use.

- The characteristic equation of a D flip-flop is: $Q^* = D$.
- For D flip-flops, the excitation table is identical to the transition table, except for the labels.

	A B				
Q1 Q2 Q3	00	01	11	10	Z
000	100	100	101	101	0
100	110	110	101	101	0
101	100	100	111	111	0
110	110	110	111	101	1
111	100	110	111	111	1
	D1 D2 D3				

7. Sequential Circuits

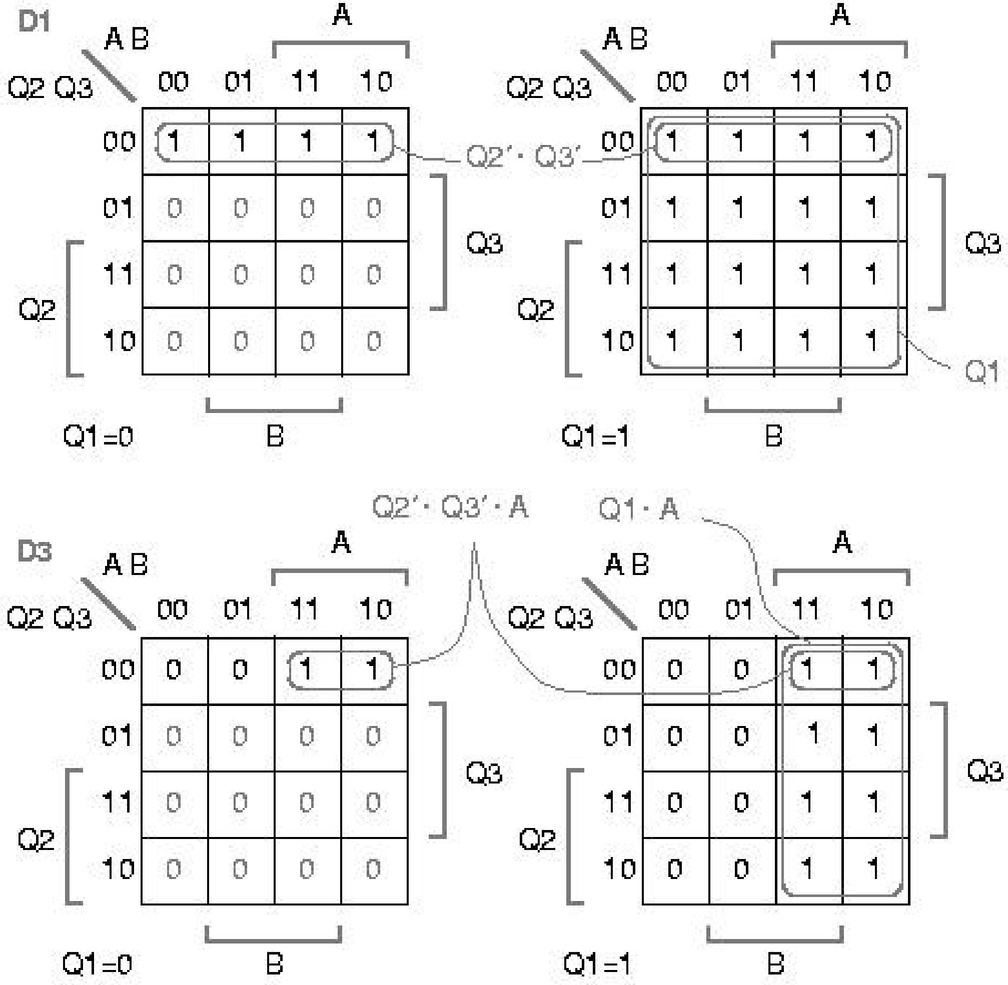
- State Machine Design (18) -

- The excitation table is like a truth table for 3 combinational functions (D1,D2,D3) of 5 variables (A,B,Q1,Q2,Q3).
- The information in the excitation table can be transferred to Karnaugh maps, to find minimal expressions for each function.
- The excitation table does not specify functional values for all input combinations, since the information for the unused states is not specified.
- For our example, we will take the two approaches previously referred: minimal risk and minimal cost.

7. Sequential Circuits

- State Machine Design (19) -

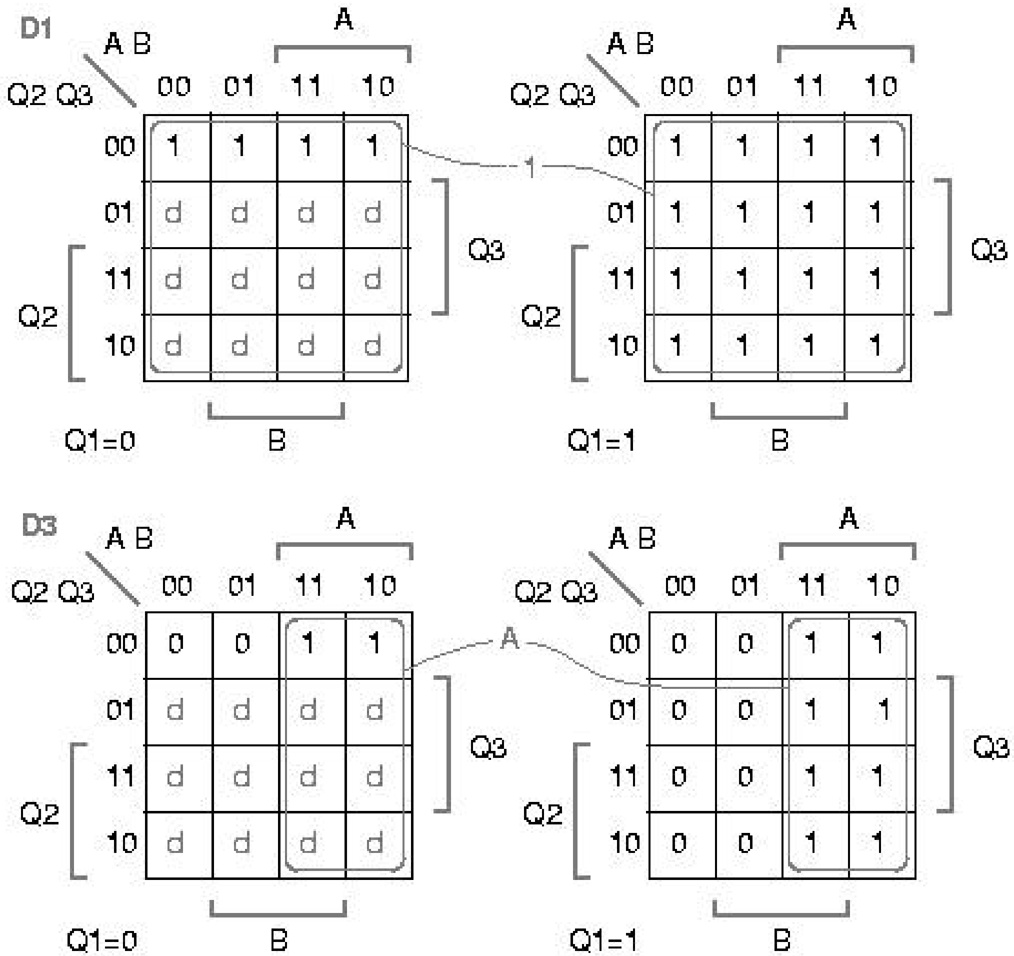
- In a <u>minimal-risk</u> approach, the next state for each unused state is 000.
- From the maps the following expressions can be obtained:
  D1 = Q1+Q2’·Q3’
  D2 = Q1·Q3’·A’+Q1·Q3·A+Q1·Q2·B
  D3 = Q1·A+Q2’·Q3’·A
- Z is active for states 110 and **111**
  Z = Q1·Q2·Q3’ + Q1·Q2·Q3
  = Q1·Q2



7. Sequential Circuits

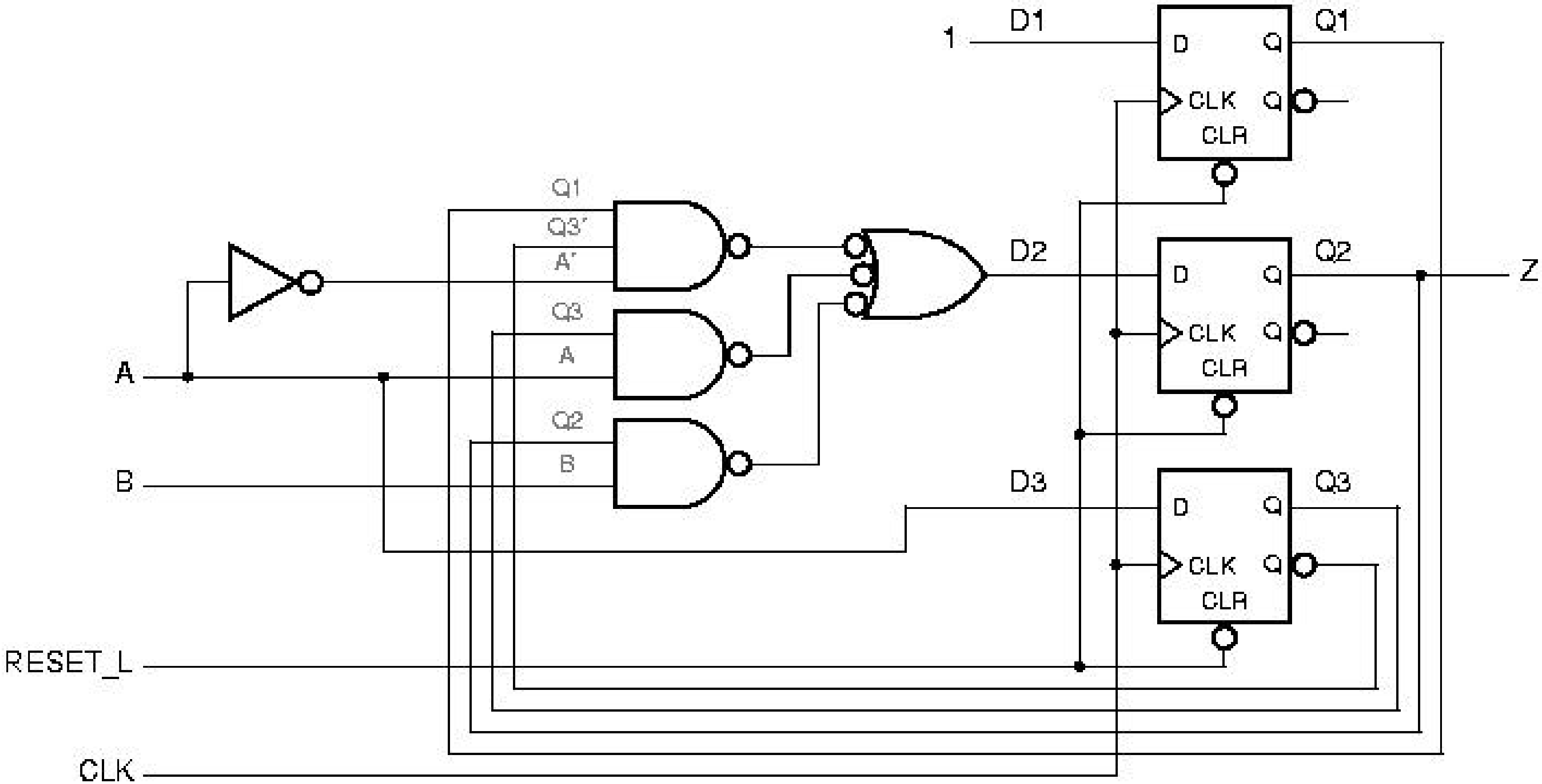
- State Machine Design (20) -

- In a <u>minimal-cost</u> approach, the next state for each unused state is a “don´t-care”.
- From the maps the following expressions can be obtained:
  D1 = 1
  D2 = Q1·Q3’·A’+Q3·A+Q2·B
  D3 = A
- Z is active for states 110 and 111 and don’t-care for the unused states.
  Z = Q2



7. Sequential Circuits

- State Machine Design (21) -



7. Sequential Circuits

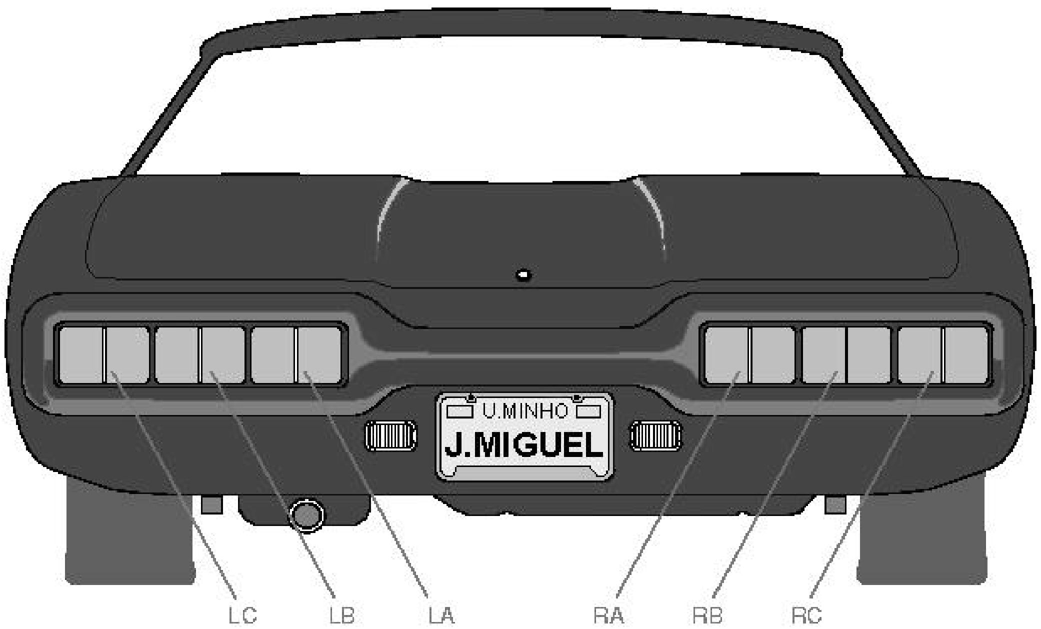
- State Machine Design (22) -

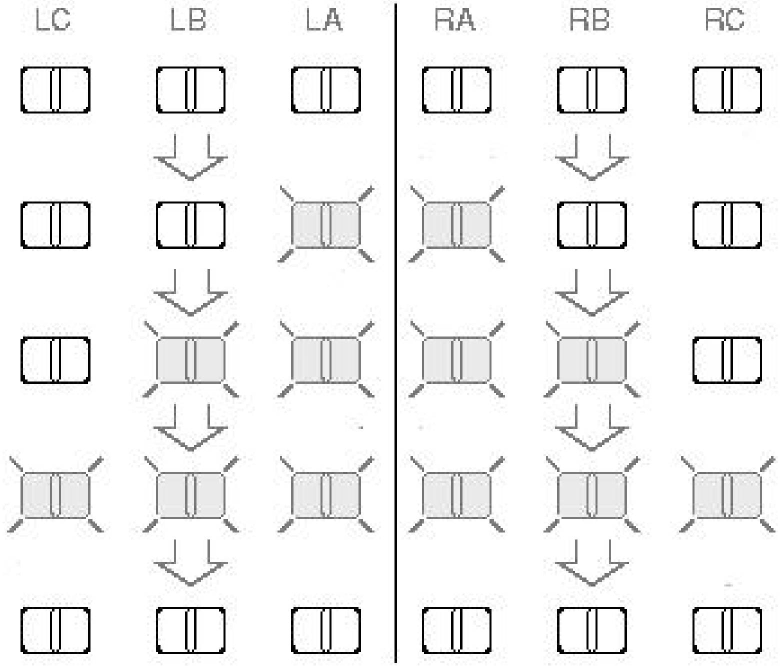
- State diagrams are often used to design state machines.
- Designing a state diagram is similar, but simpler, to design a state table.
- A state diagram can contain some ambiguities, which is not possible in a state table.
- In an improperly constructed state diagram, the next state for some input combinations may be unspecified, which is undesirable.
- It is also possible that multiple next states exist for the same input combination.

7. Sequential Circuits

- State Machine Design (23) -

- The next example is a state machine that control the tail lights of a car.
- The machine has 2 input signals, LEFT and RIGHT.
- It also has an emergency HAZ input that makes the 6 lights to flash.





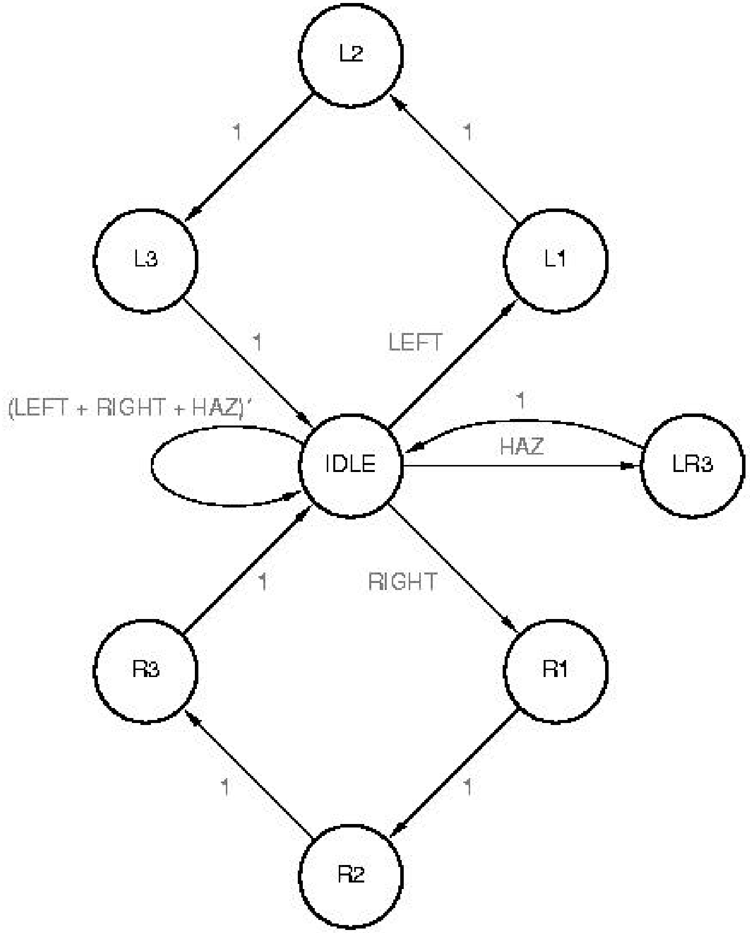
7. Sequential Circuits

- State Machine Design (24) -

- State diagram and Output table for the car lights controller.

Output Table

State	LC	LB	LA	RA	RB	RC
IDLE	0	0	0	0	0	0
L1	0	0	1	0	0	0
L2	0	1	1	0	0	0
L3	1	1	1	0	0	0
R1	0	0	0	1	0	0
R2	0	0	0	1	1	0
R3	0	0	0	1	1	1
LR3	1	1	1	1	1	1

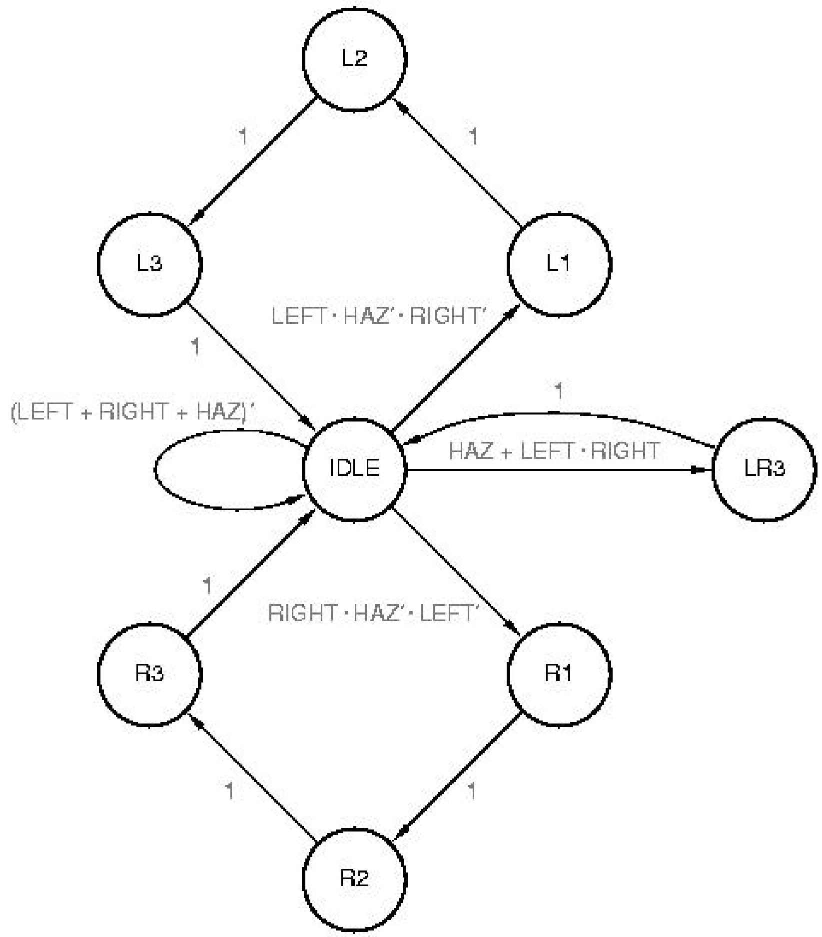


- Multiple inputs asserted simultaneously (LEFT and HAZ at IDLE) are not handled.

7. Sequential Circuits

- State Machine Design (25) -

- Priority was given to the HAZ input.
- When LEFT and RIGHT are asserted simultaneously, it is assumed that an emergency is requested.
- The new state diagram is unambiguous.
- The transition expressions on the arcs leaving the same state are mutually exclusive and all-inclusive.
  - **No two expressions are 1 for the same input combination.**
  - **Some expression is 1 for every input combination**

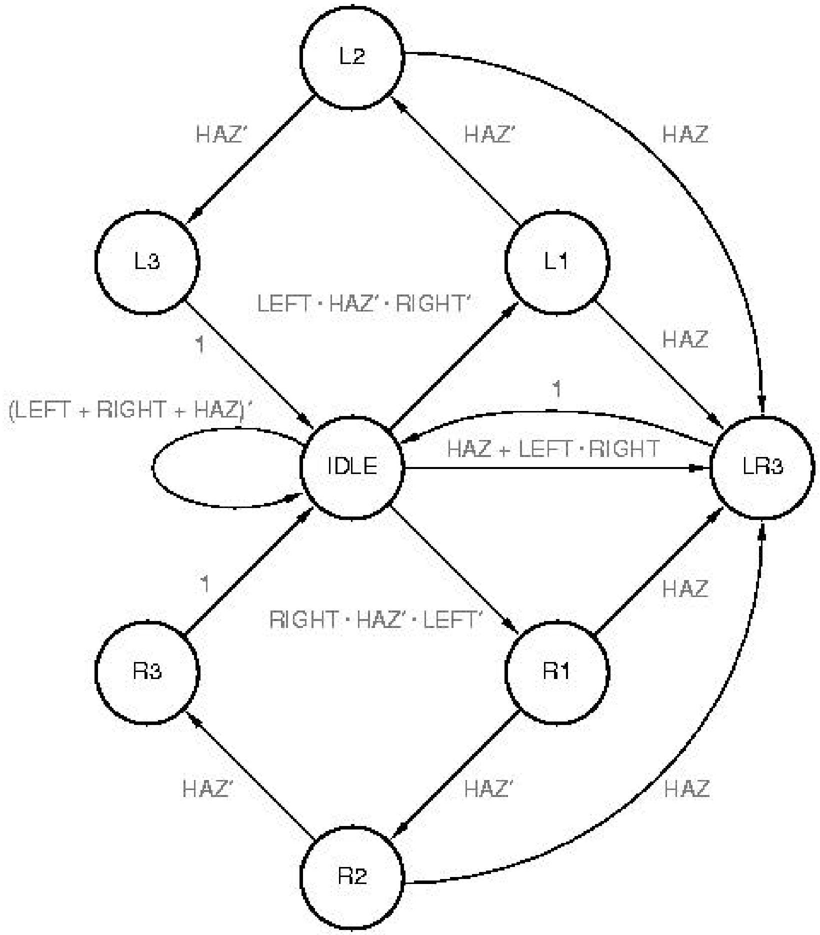


7. Sequential Circuits

- State Machine Design (26) -

- Once a left- or right-turn cycle has begun, if must be finished even if an emergency is requested.
- It is safer to have the machine go into LR3 state as soon as possible.
- There are 8 states, so 3 flip-flops are needed to synthesise the circuit.

State	Q2	Q1	Q0
IDLE	0	0	0
L1	0	0	1
L2	0	1	1
L3	0	1	0
R1	1	0	1
R2	1	1	1
R3	1	1	0
LR3	1	0	0



7. Sequential Circuits

- State Machine Design (27) -

S	*Q2*	*Q1*	*Q0*	*Transition Expression*	*S**	*Q2**	*Q1**	*Q0**
IDLE	0	0	0	(LEFT + RIGHT + HAZ)'	IDLE	0	0	0
IDLE	0	0	0	LEFT · HAZ' · RIGHT'	L1	0	0	1
IDLE	0	0	0	HAZ + LEFT · RIGHT	LR3	1	0	0
IDLE	0	0	0	RIGHT · HAZ' · LEFT'	R1	1	0	1
L1	0	0	1	HAZ'	L2	0	1	1
L1	0	0	1	HAZ	LR3	1	0	0
L2	0	1	1	HAZ'	L3	0	1	0
L2	0	1	1	HAZ	LR3	1	0	0
L3	0	1	0	1	IDLE	0	0	0
R1	1	0	1	HAZ'	R2	1	1	1
R1	1	0	1	HAZ	LR3	1	0	0
R2	1	1	1	HAZ'	R3	1	1	0
R2	1	1	1	HAZ	LR3	1	0	0
R3	1	1	0	1	IDLE	0	0	0
LR3	1	0	0	1	IDLE	0	0	0

7. Sequential Circuits

- State Machine Design (28) -

- Most of the VHDL features that are needed to support clocked synchronous state machines were already introduced.
- A VHDL process and the simulator's mechanism for tracking signal changes form the basis for handling sequential circuits in VHDL.
- The `event` attribute can be attached to a signal name to yield a value that is true if the signal has changed value.
- This allows edge-trigger behaviour to be modelled.

7. Sequential Circuits

- State Machine Design (29) -

- Positive-edge-triggered D flip-flop with asynchronous clear.
- The CLR input overrides any behaviour on the CLK input.
- `CLK'event` is true for any change on CLK.
- Two other ways to construct processes or statements with edge-triggered behaviour.

```
library IEEE;
use IEEE.std_logic_1164.all;

entity VposDff is
  port (CLK, CLR, D: in STD_LOGIC;
        Q, QN: out STD_LOGIC );
end VposDff;

architecture VposDff_arch of VposDff is
begin
  process (CLK, CLR)
  begin
    if CLR='1' then Q <= '0' ; QN <= '1' ;
    elsif CLK'event and CLK='1' then Q <= D; QN <= not D;
    end if;
  end process;
end VposDff_arch;
```

```
process
  wait until CLK'event and CLK='1' ;
  Q <= D;
end process;

Q <= D when CLK'event and CLK='1' else Q;
```