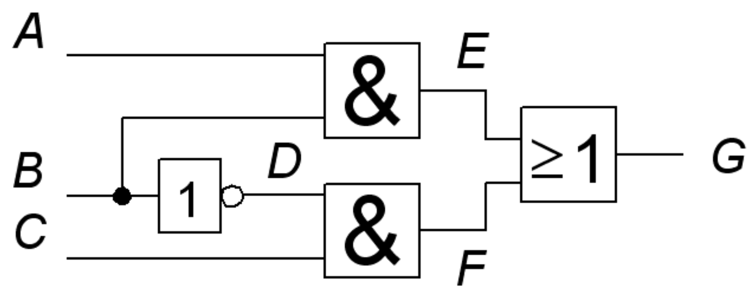
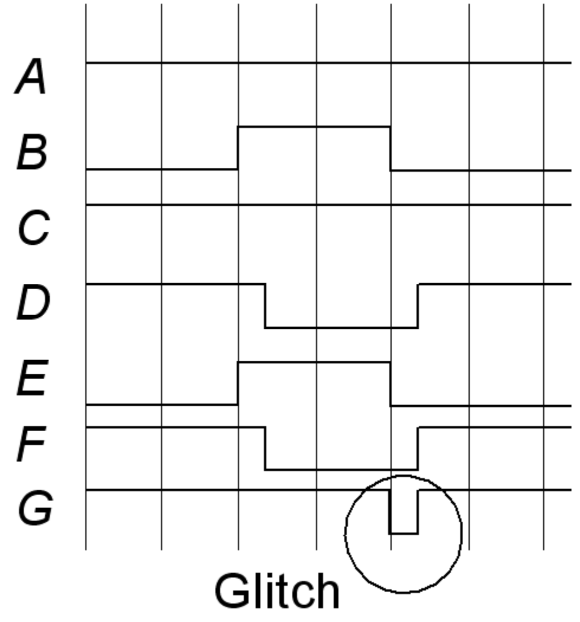
Ex **11.1** ”Glitches”

If the signals passes different amount of gate delays before they are combined at the output, then momentary unwanted deviations from the truth table can occur, so-called "glitches".

Show in Karnaugh map how to avoid them.





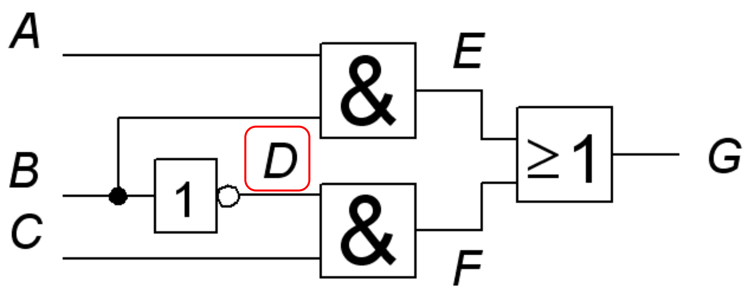
(*in the figure, only the delay in the inverter is included - the other gate delays that do not affect the "glitch" has not been included*)

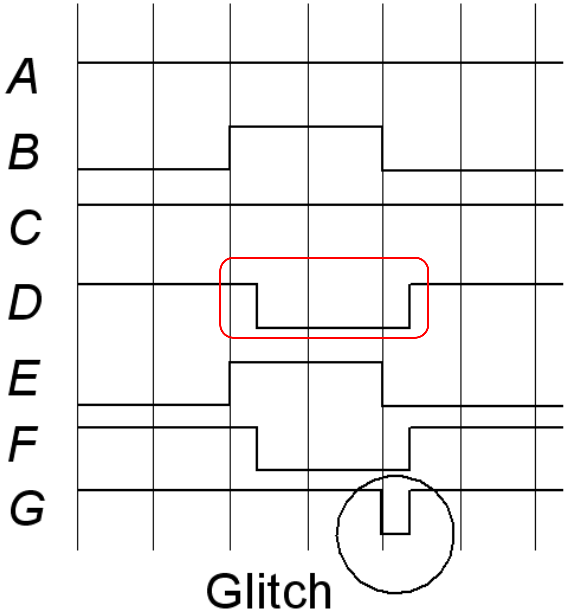
William Sandqvist william@kth.se

Ex **11.1** ”Glitches”

If the signals passes different amount of gate delays before they are combined at the output, then momentary unwanted deviations from the truth table can occur, so-called "glitches".

Show in Karnaugh map how to avoid them.

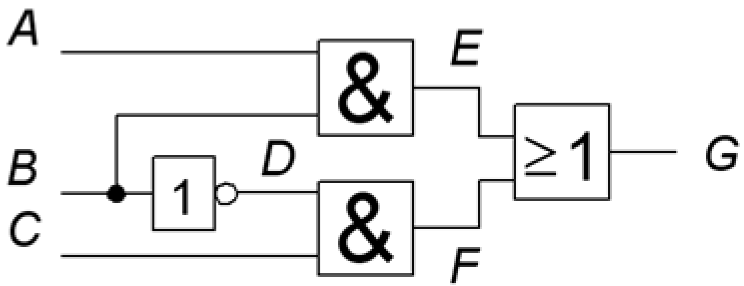


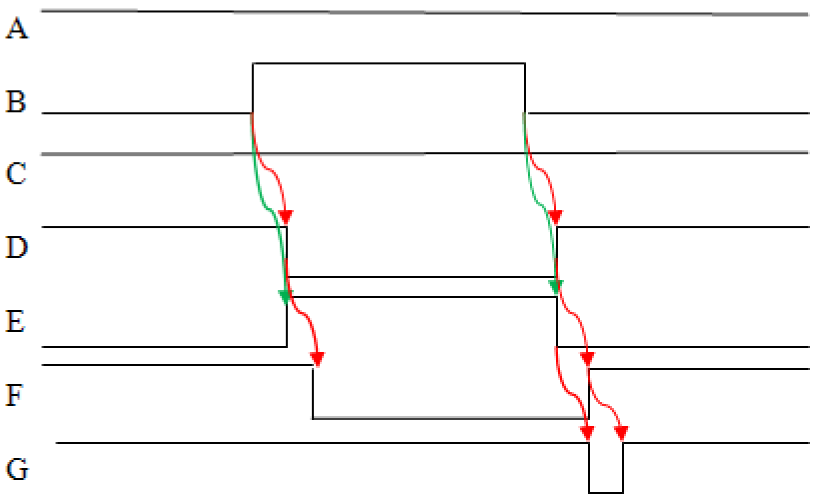


The signal D is delayed compared to A B C.

(*in the figure, only the delay in the inverter is included - the other gate delays that do not affect the "glitch" has not been included*)

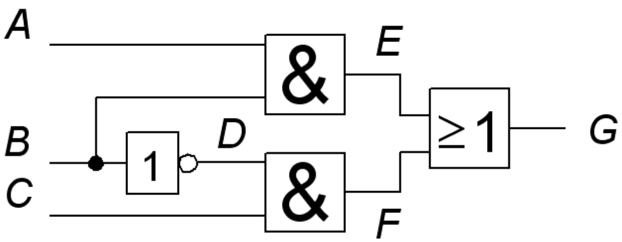
(with all gate delays included)





(Jan Andersson)

11.1

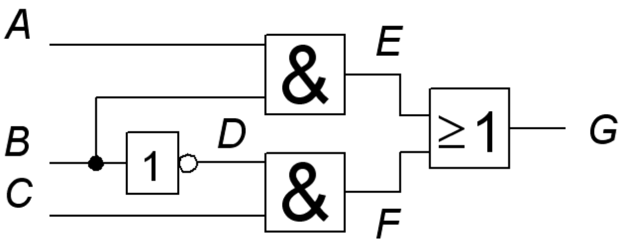


The circuit in a Karnaughmap:

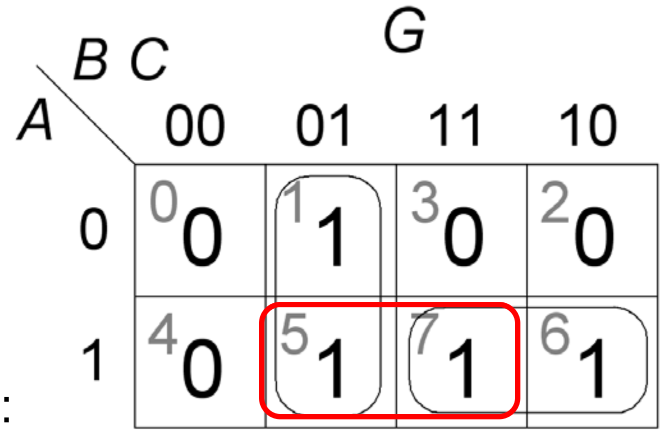
$A \backslash BC$	00	01	11	10
0	$^{0}0$	$^{1}1$	$^{3}0$	$^{2}0$
1	$^{4}0$	$^{5}1$	$^{7}1$	$^{6}1$

Output: G

11.1

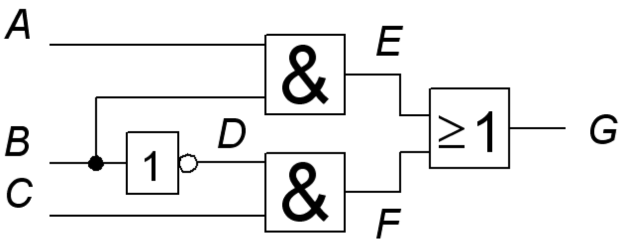


The circuit in a Karnaughmap:



Make sure the groupings in the Karnaugh map form a continuous "continent" - no islands! (You include the consensus terms to obtain the function in full prime implicator form).

11.1



The circuit in a Karnaughmap:

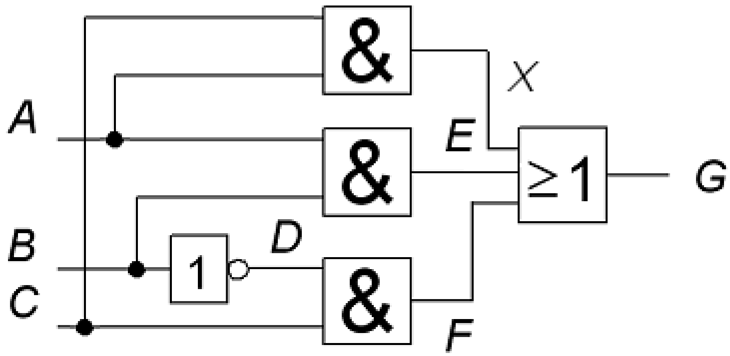
G

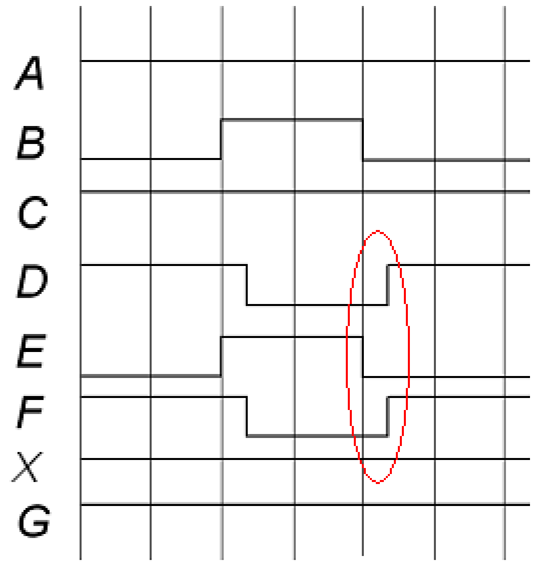
A \ BC	00	01	11	10
0	[0] 0	[1] 1	[3] 0	[2] 0
1	[4] 0	[5] 1	[7] 1	[6] 1

Make sure the groupings in the Karnaugh map form a continuous "continent" - no islands! (You include the consensus terms to obtain the function in full prime implicator form).

$$G = \overline{B}C + AB \quad \{No\ Hazards\} \quad G = \overline{B}C + AB + \boxed{AC}$$

11.1

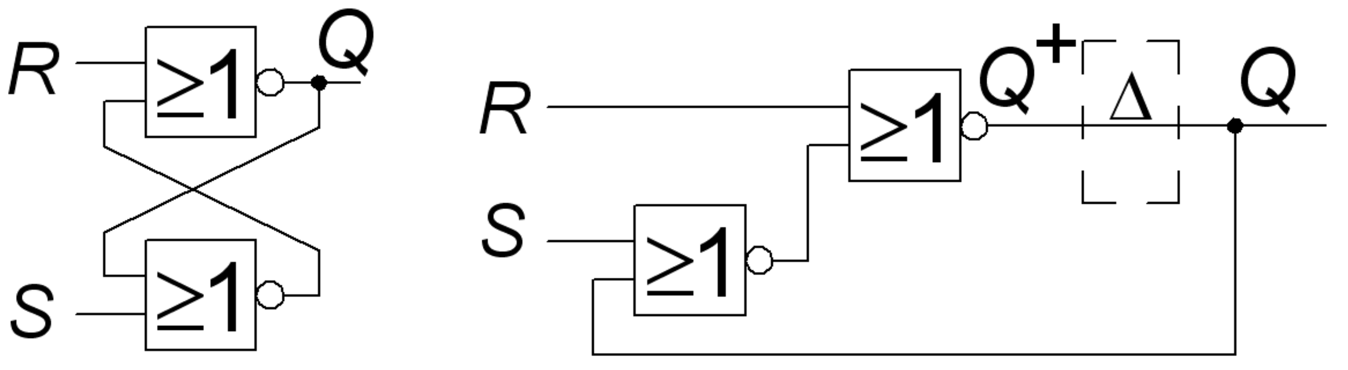




We see that the signal *X* is "covering up" when there is a risk of a "glitch", to the price of a more complex network!

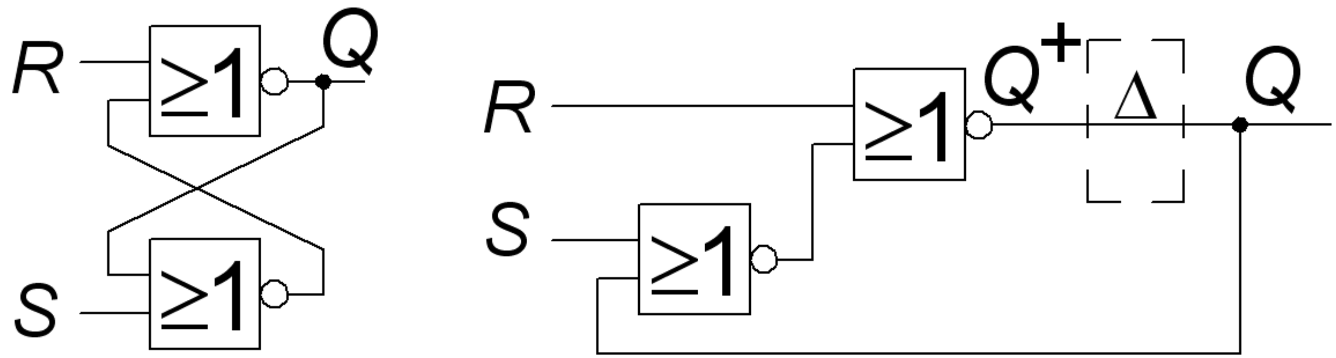
Ex **11.2** SR asynchronous sequential circuit

SR-latch is an asynchronous sequential circuit.

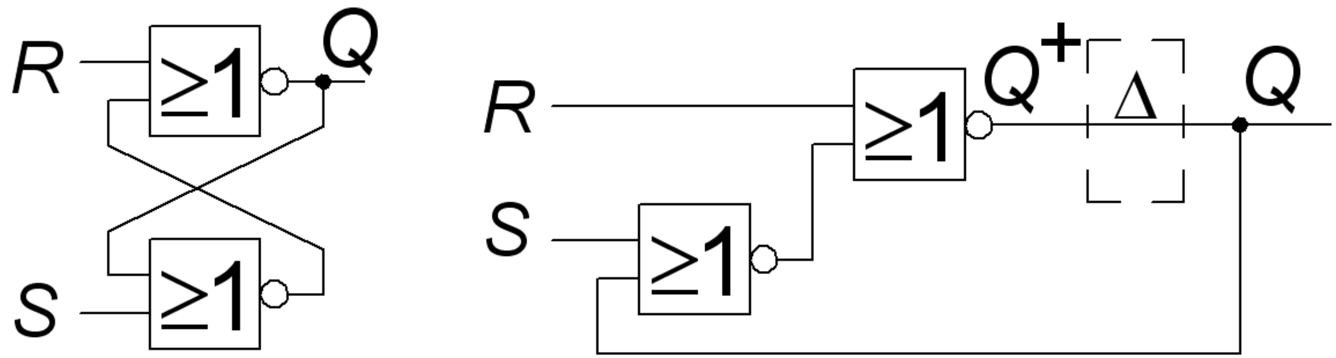


All gate delays present in the network is thought placed in the symbol Δ which has a similar function to the D-flip-flop in a synchronous sequential circuit.

SR Analyses:

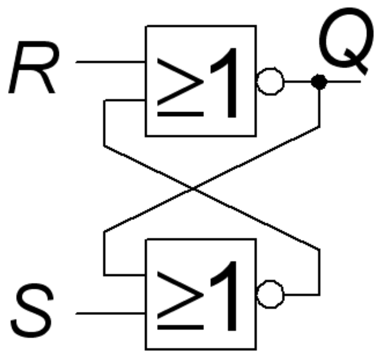


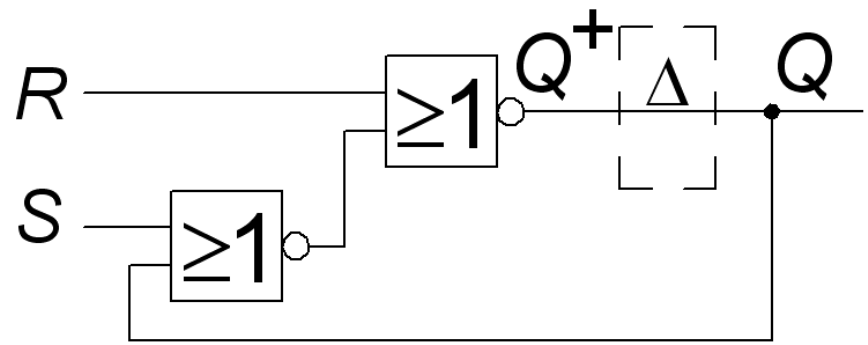
SR Analyses:



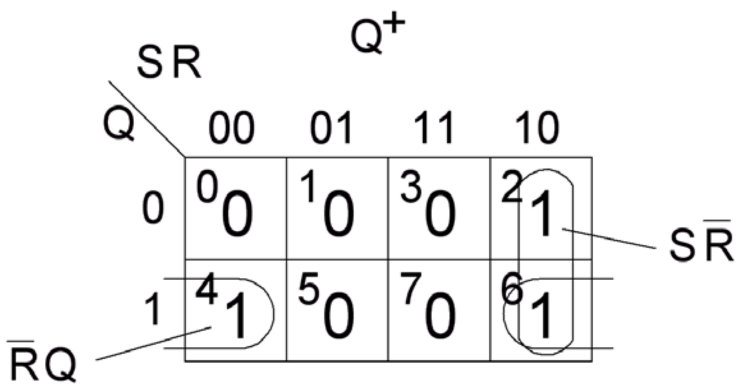
$$Q^{+} = \overline{R + \overline{S + Q}} = \overline{R} \cdot \overline{\overline{(S + Q)}} = \overline{R} \cdot (S + Q) = S\overline{R} + \overline{R}Q$$

SR Analyses:



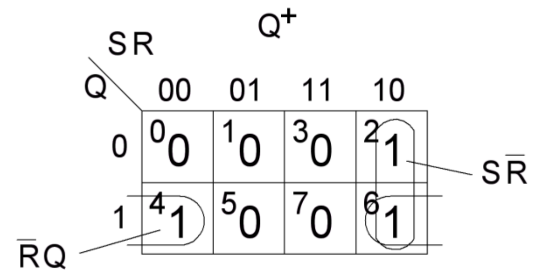


$$Q^{+} = \overline{R + \overline{S + Q}} = \overline{R} \cdot \overline{\overline{(S + Q)}} = \overline{R} \cdot (S + Q) = S\overline{R} + \overline{R}Q$$



SR Coded state table

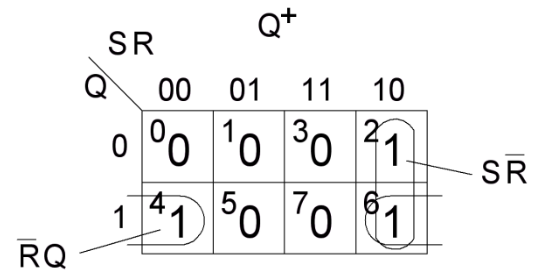
The encoded state table is usually called **excitationstable** when working with asynchronous state machines.



Present state Q	Next state Q^+			
	Input signals SR			
	00	01	11	10
0	0	0	0	1
1	1	0	0	1

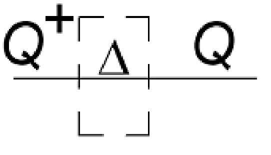
SR Coded state table

The encoded state table is usually called **excitationstable** when working with asynchronous state machines.



Present state Q	Next state Q^+			
	Input signals SR			
	00	01	11	10
0	0	0	0	1
1	1	0	0	1

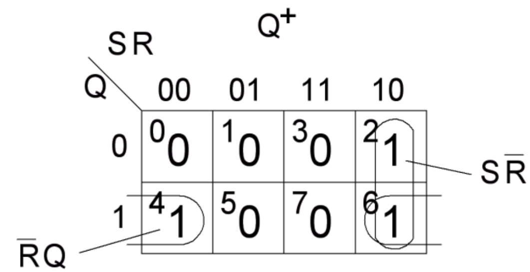
For each input (column), there must be at least one state where Q = Q^+. Such conditions are stable and they are usually marked by a circle.



William Sandqvist william@kth.se

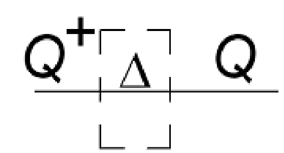
SR Coded state table

The encoded state table is usually called **excitationstable** when working with asynchronous state machines.



Present state Q	**Next state Q^+**			
	Input signals SR			
	00	01	11	10
0	0	0	0	1
1	1	0	0	1

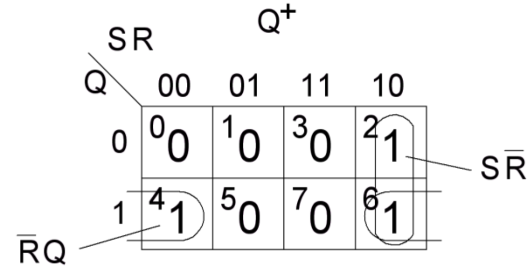
For each input (column), there must be at least one state where $Q = Q^+$. Such conditions are stable and they are usually marked by a circle.



William Sandqvist william@kth.se

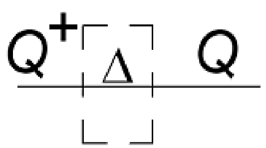
SR Coded state table

The encoded state table is usually called **excitationstable** when working with asynchronous state machines.



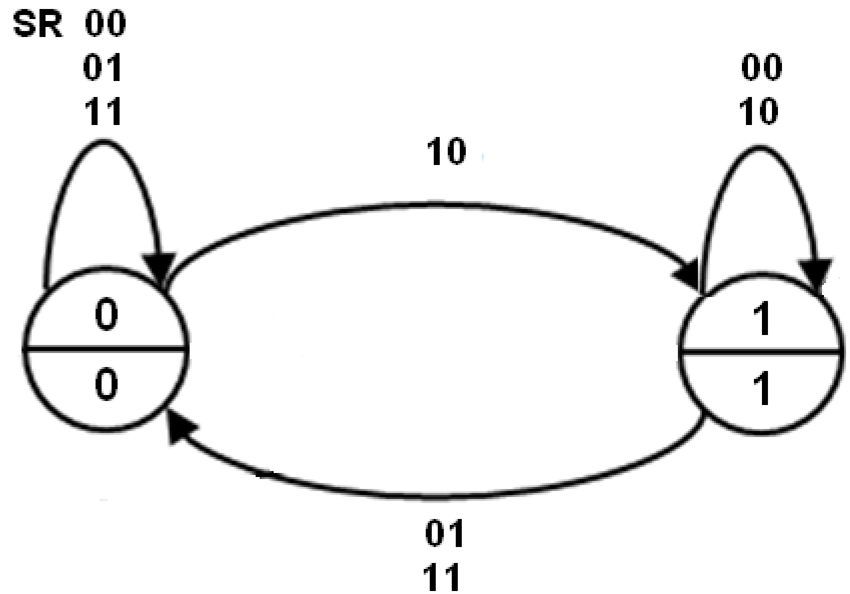
Present state Q	Next state Q^+			
	Input signals SR			
	00	01	11	10
0	0	0	0	1
1	1	0	0	1

For each input (column), there must be at least one state where $Q = Q^+$. Such conditions are stable and they are usually marked by a circle.



SR State diagram

Present state Q	Next state Q^+			
	Input signals SR			
	00	01	11	10
0	0	0	0	1
1	1	0	0	1

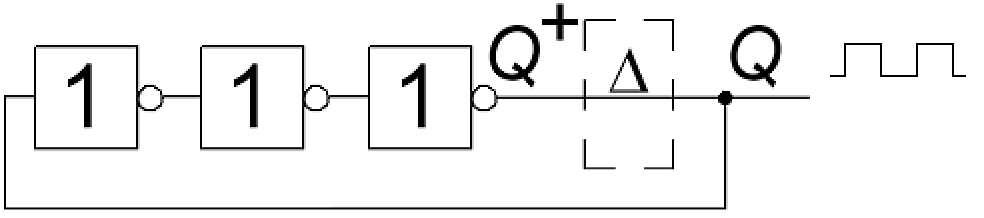


SR State table

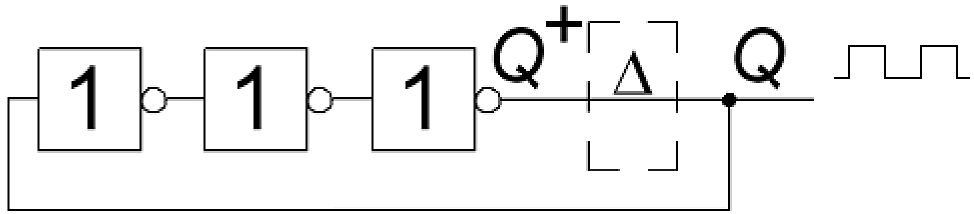
The state table is named **flow table** when working with asynchronous state machines.

Present state Q	**Next state Q^+**			
	Input signals SR			
	00	01	11	10
A	A	A	A	B
B	B	A	A	B

Ex **11.3** Oscillator?

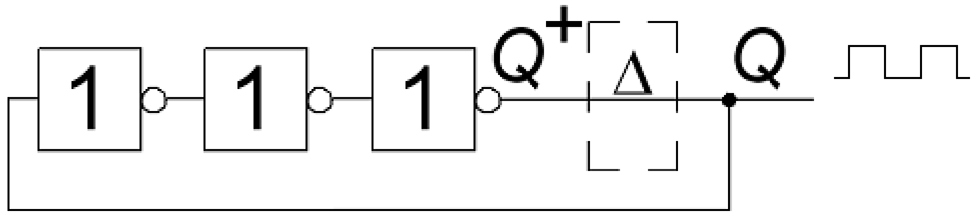


Ex **11.3** Oscillator?



$$Q^{+} = \overline{Q}$$

Ex **11.3** Oscillator?

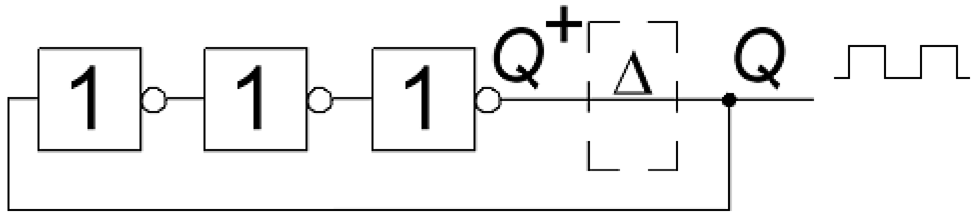


$$Q^+ = \overline{Q}$$

Q	Q^+
0	1
1	0

No stable states!

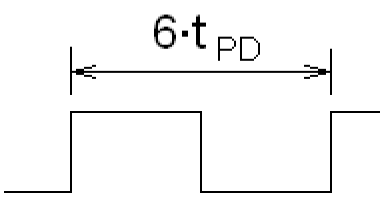
Ex **11.3** Oscillator?



$$Q^+ = \overline{Q}$$

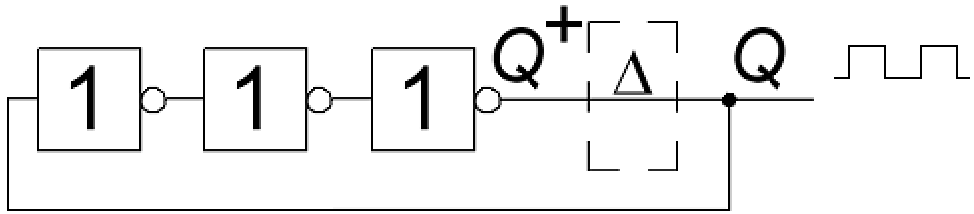
Q	Q^+
0	1
1	0

No stable states!



$$T = 6 \cdot t_{PD} \quad \Rightarrow \quad f = \frac{1}{6 \cdot t_{PD}}$$

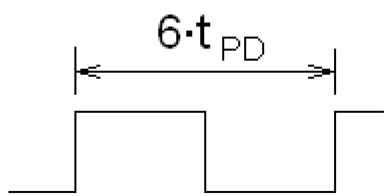
Ex **11.3** Oscillator?



$$Q^{+} = \overline{Q}$$

Q	Q^{+}
0	1
1	0

No stable states!

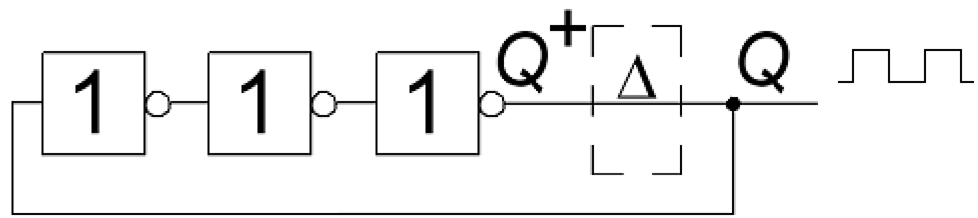


$$T = 6 \cdot t_{PD} \quad \Rightarrow \quad f = \frac{1}{6 \cdot t_{PD}}$$

Numerical Example: $t_{pd} = 5 \cdot 10^{-9} \quad f = \frac{1}{6 \cdot 5 \cdot 10^{-9}} = 33\,\text{MHz}$

William Sandqvist william@kth.se

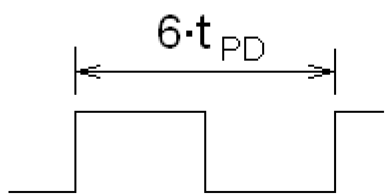
Ex **11.3** Oscillator?



$$Q^{+} = \overline{Q}$$

Q	Q^{+}
0	1
1	0

No stable states!



$$T = 6 \cdot t_{PD} \quad \Rightarrow \quad f = \frac{1}{6 \cdot t_{PD}}$$

Numerical Example: $t_{pd} = 5 \cdot 10^{-9} \quad f = \dfrac{1}{6 \cdot 5 \cdot 10^{-9}} = 33\,\text{MHz}$

Can be used to indirectly measure the gate delay of logic circuits.

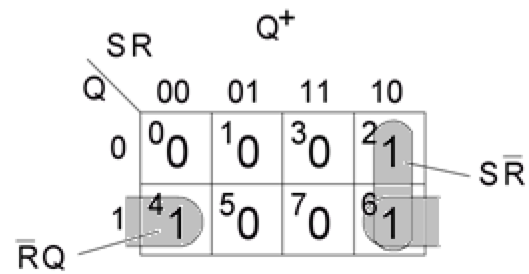
Especially for asynchronous circuits

- The states must be encoded Race-free (eg. Gray code).

SR latch is race free because there is only one state signal, which of course can not run races with itself.

- Next state decoder must be glitch free / Hazard free (with the consensus terms included).

SR-latch circuit groupings are contiguous in the Karnaugh map, there are no more consensus terms that need to be included.



Especially for asynchronous circuits

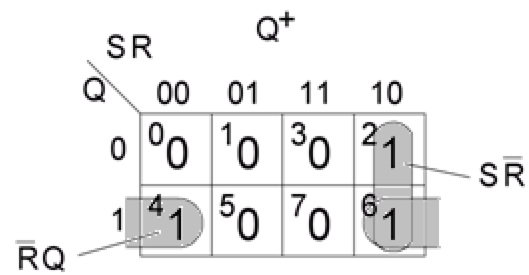
- The states must be encoded Race-free (eg. Gray code).

SR latch is race free because there is only one state signal, which of course can not run races with itself.

- Next state decoder must be glitch free / Hazard free (with the consensus terms included).

SR-latch circuit groupings are contiguous in the Karnaugh map, there are no more consensus terms that need to be included.

The SR-latch is thus an "goof-proof" design. Larger asynchronous sequential circuits are significantly more complex to construct!

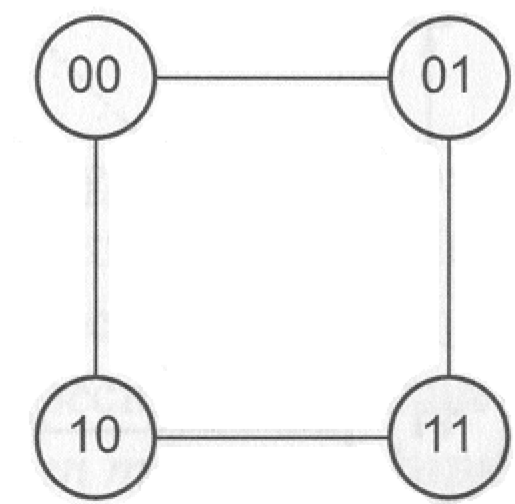


William Sandqvist william@kth.se

State Diagram as hypercubes

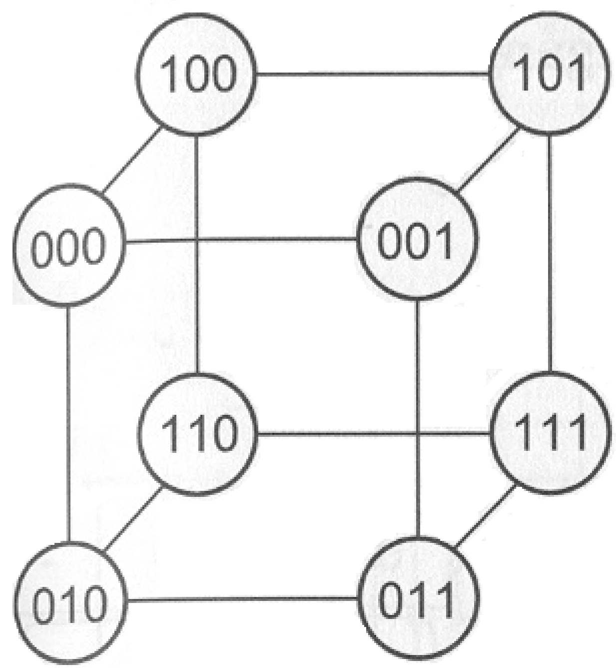
The state diagram is placed on a hypercube with Gray-coded corners.

With two state variables, it becomes a square.



State Diagram as hypercubes

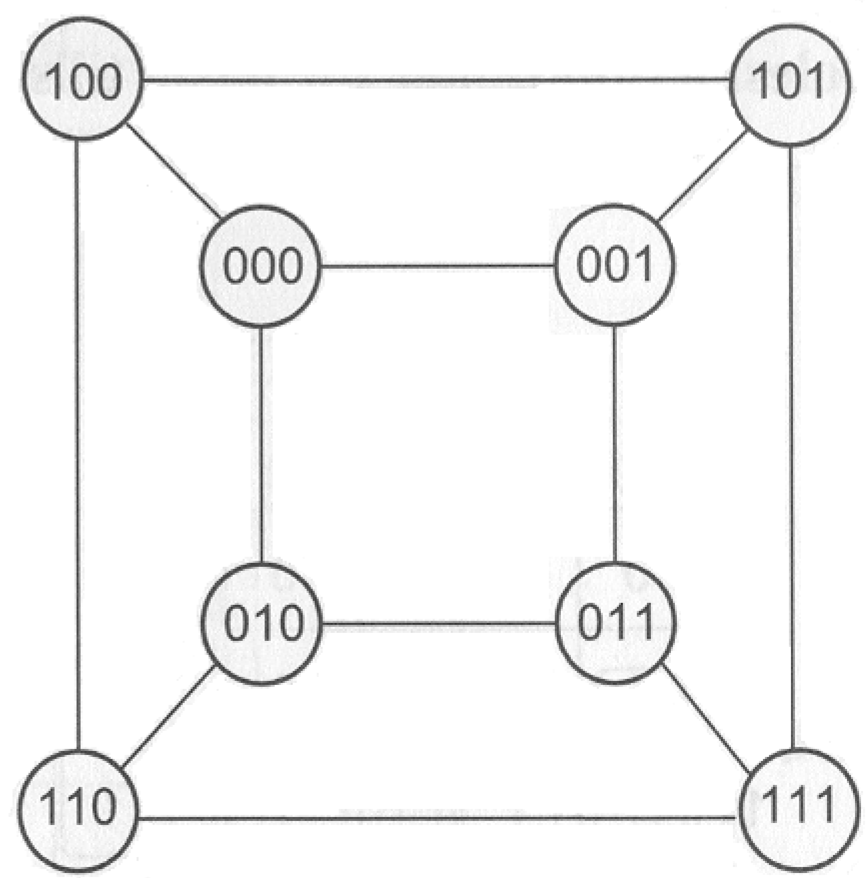
With three state variables, it becomes a cube



State Diagram as hypercubes

With three state variables, it becomes a cube

It is becoming clearer if one "flattens" the cube.

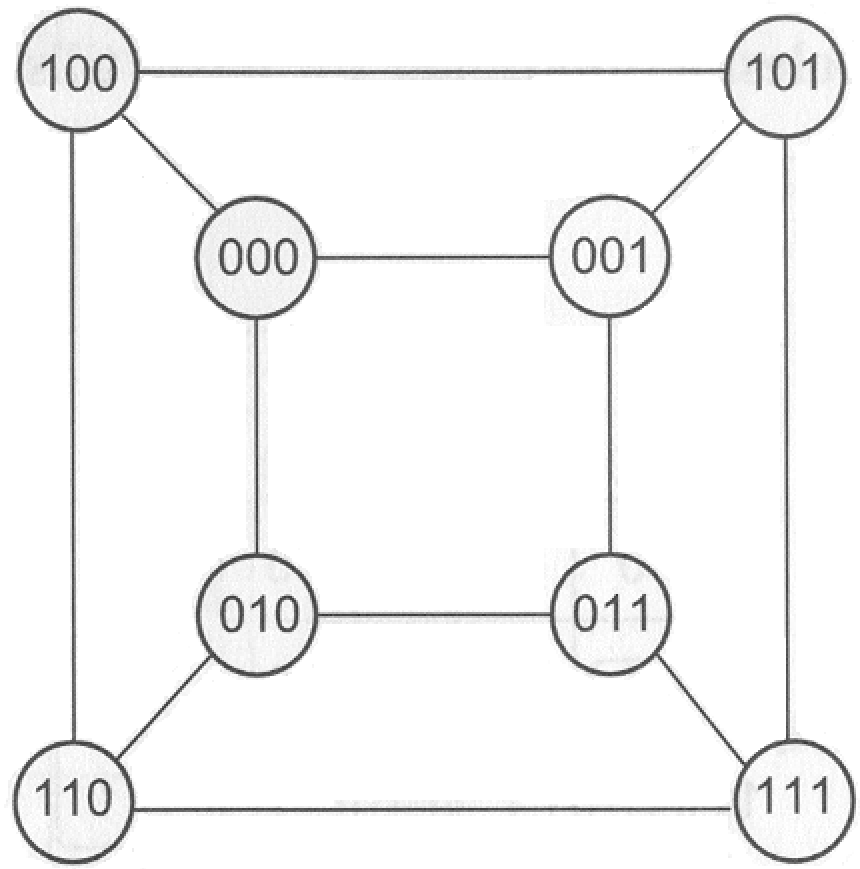


State Diagram as hypercubes

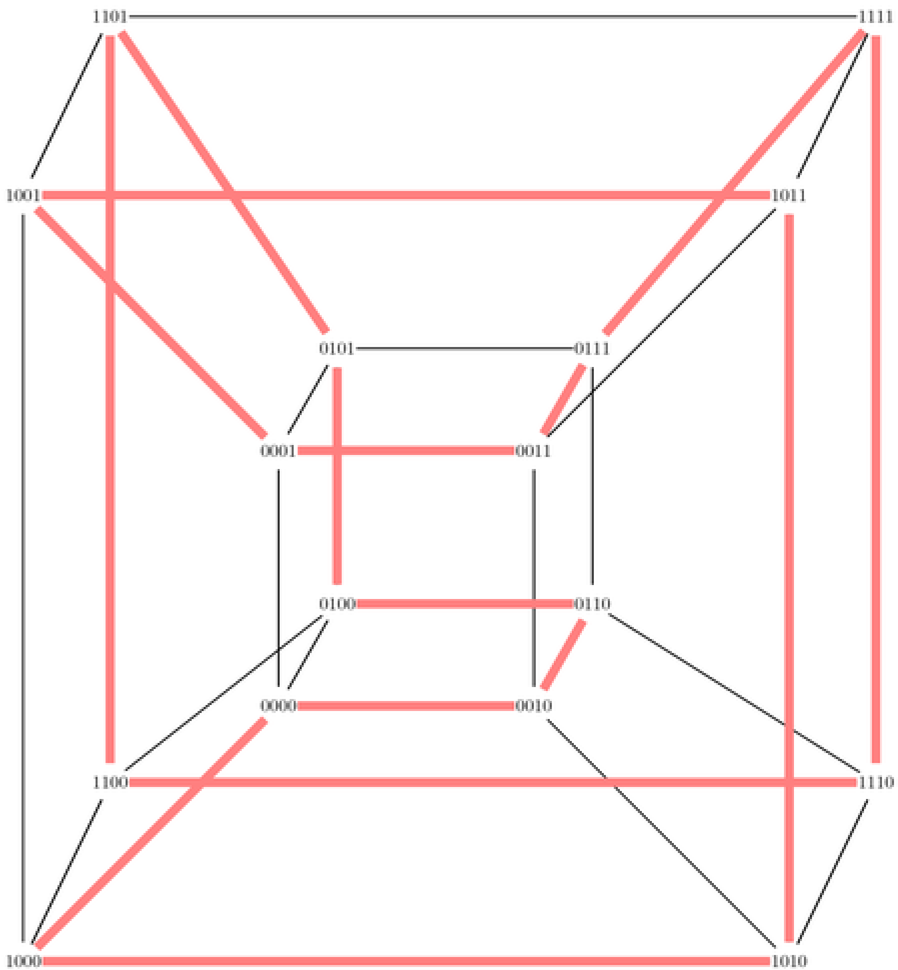
With three state variables, it becomes a cube

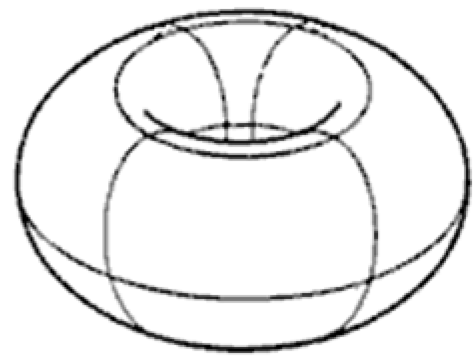
It is becoming clearer if one "flattens" the cube.

For more variables, the principle is the same, but the states are placed in the corners of hypercubes and it becomes harder to draw.

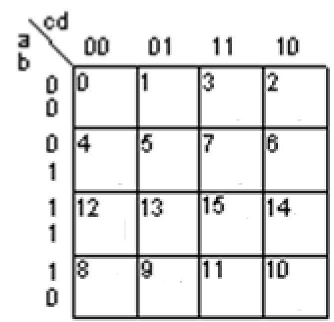


(Four variables)

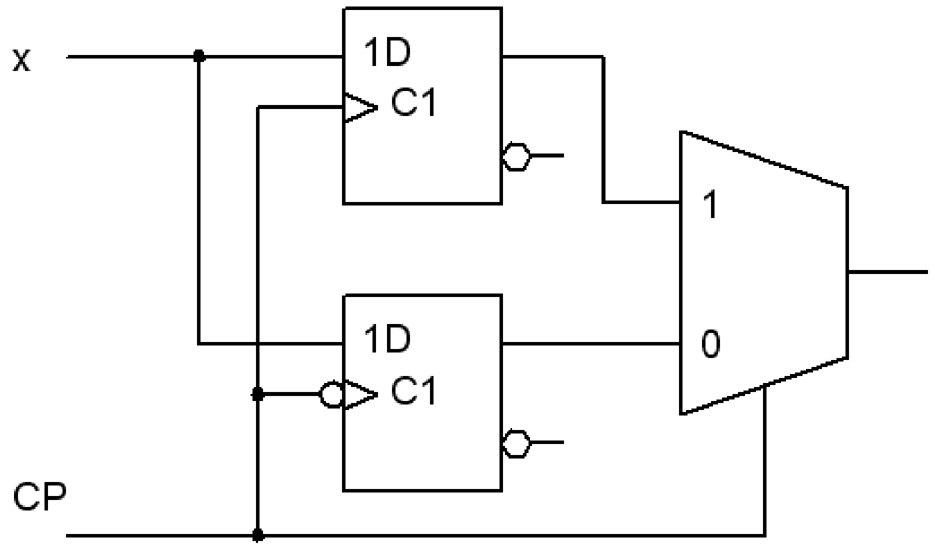




(Compare with the Karnaugh map)



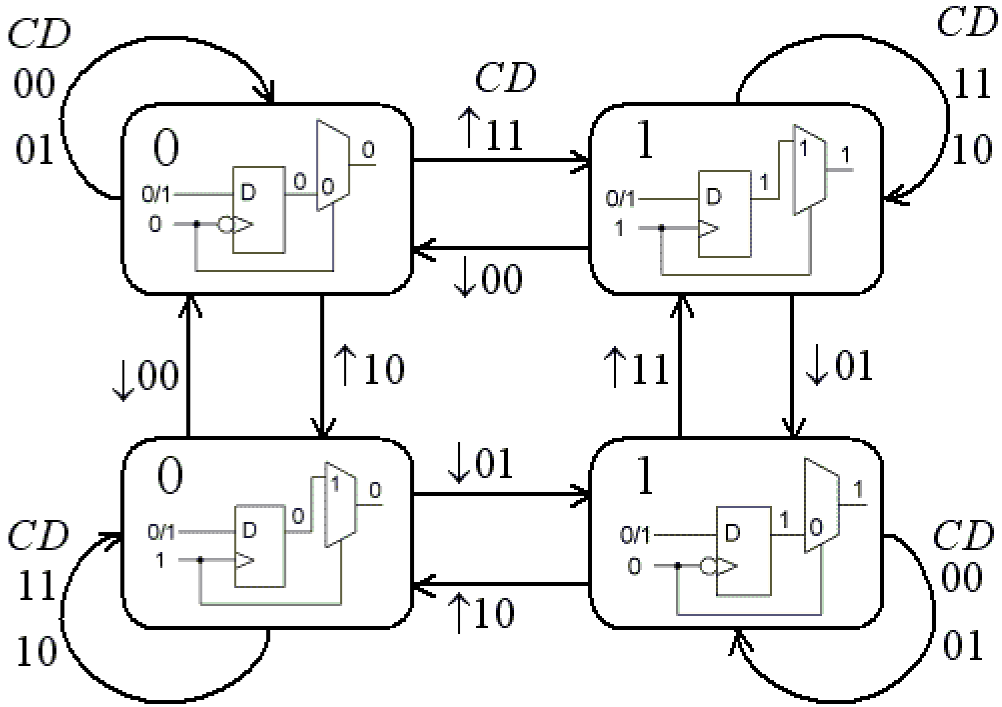
Ex **11.4**

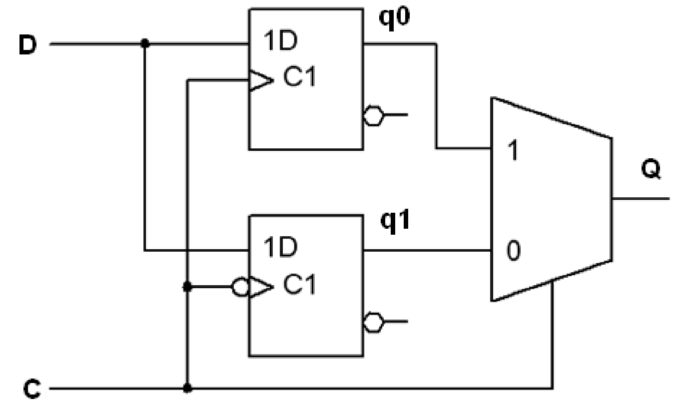


Analyze the following circuit. Draw a State Diagram.

Consider the circuit as an asynchronous sequential circuit which clock pulse input is one of the asynchronous inputs. What is the function of the circuit?

11.4 Positive edge and negative edge

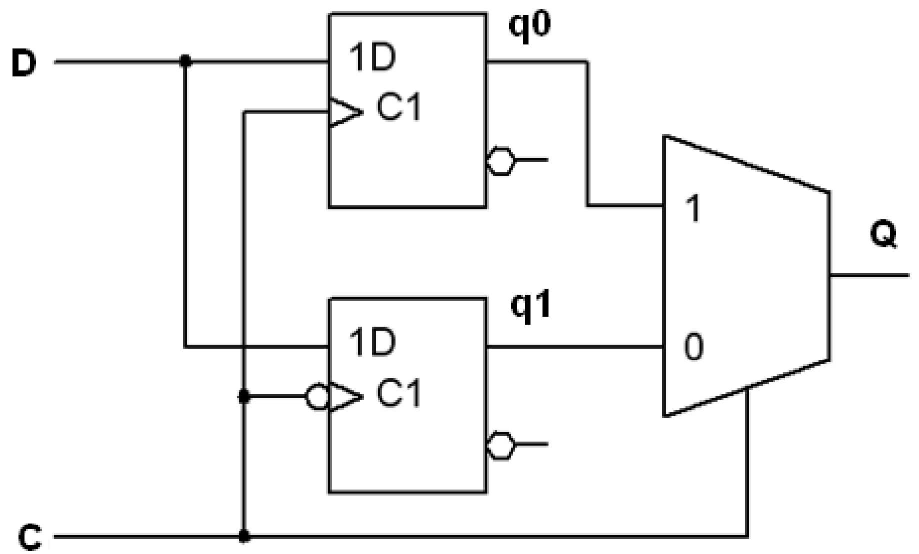


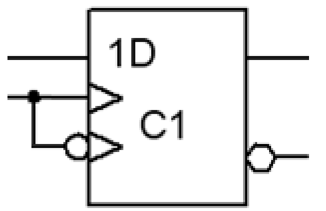


- At a positive edge ↑ **C** changes from 0 to 1 and when **C**=1 the MUX connects the upper flip-flop **q0** to the output.
- At a negative edge ↓ **C** changes from 1 to 0 and when **C**=0 the MUX connects the lower flip-flop **q1** to the output.

The result is a **D**-flip-flop that reacts on *both* edges of the clock.

DETFF-flip-flop



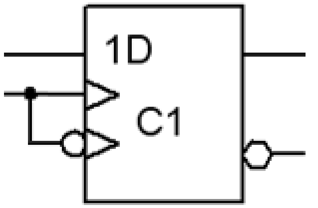


DETFF symbol

Double Edge Trigered Flip Flop (DETFF) has advantages in speed and power consumption. It can in principle provide twice as fast sequential circuits!
(Introduction of DETFF-flip-flops would require rethinking and redesigning of the other logic).

In order to benefit from the advantages of DETFF-flip-flop it must be designed as a separate component - ie as an asynchronous sequential circuit.

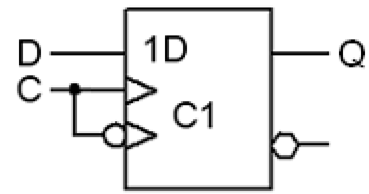
Ex **11.5** DETFF



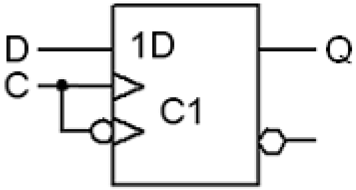
Construct an asynchronous state machine that functions as a dubble edge triggered D flip-flop (DETFF), the flip-flop will change value at both the positive and the negative edge of the clock.

a) Derive the FSM.
b) Construct the flow table and minimize it.
c) Assign states, transfer to Karnaugh maps and derive the Boolean expressions.
d) Draw the schematic for the circuit.

11.5 Possible in/out combinations



11.5 Possible in/out combinations

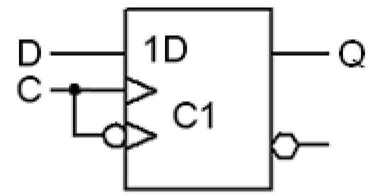


DETFF
Characteristic table

C	D	Q^+
0	-	Q
1	-	Q
↑	0	0
↑	1	1
↓	0	0
↓	1	1

11.5 Possible in/out combinations

There are four input combinations (CD) and two output combinations (Q). A total of 8 possible states (CD Q).



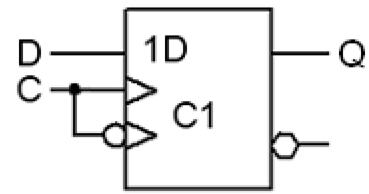
DETFF
Characteristic table

C D	Q^+
0 -	Q
1 -	Q
↑ 0	0
↑ 1	1
↓ 0	0
↓ 1	1

William Sandqvist william@kth.se

11.5 Possible in/out combinations

There are four input combinations (CD) and two output combinations (Q). A total of 8 possible states (CD Q).



DETFF
Characteristic table

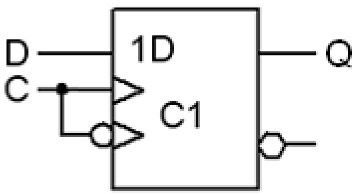
C D	Q^+
0 -	Q
1 -	Q
↑ 0	0
↑ 1	1
↓ 0	0
↓ 1	1

A new next state we get by changing either C or D. When C is changed, we get a positive edge (↑) or negative edge (↓). For *both* edges comes that D are copied to Q^+. (according to the characteristic table)

Present state		Next state
Name:	(CD Q)	$(CD\ Q)^+$
A	00 0	
B	00 1	
C	01 0	
D	01 1	
E	10 0	
F	10 1	
G	11 0	
H	11 1	

11.5 Possible in/out combinations

There are four input combinations (CD) and two output combinations (Q). A total of 8 possible states (CD Q).



DETFF
Characteristic table

C D	Q^+
0 -	Q
1 -	Q
↑ 0	0
↑ 1	1
↓ 0	0
↓ 1	1

A new next state we get by changing either C or D. When C is changed, we get a positive edge (↑) or negative edge (↓). For *both* edges comes that D are copied to Q^+. (according to the characteristic table)

Present state		Next state
Name:	(CD Q)	$(CD\ Q)^+$
A	00 0	01 0 C ↑ 10 0 E
B	00 1	01 1 D ↑ 10 0 E
C	01 0	00 0 A ↑ 11 1 H
D	01 1	00 1 B ↑ 11 1 H
E	10 0	↓ 00 0 A 11 0 G
F	10 1	↓ 00 0 A 11 1 H
G	11 0	↓ 01 1 D 10 0 E
H	11 1	↓ 01 1 D 10 1 F

11.5 Flow table

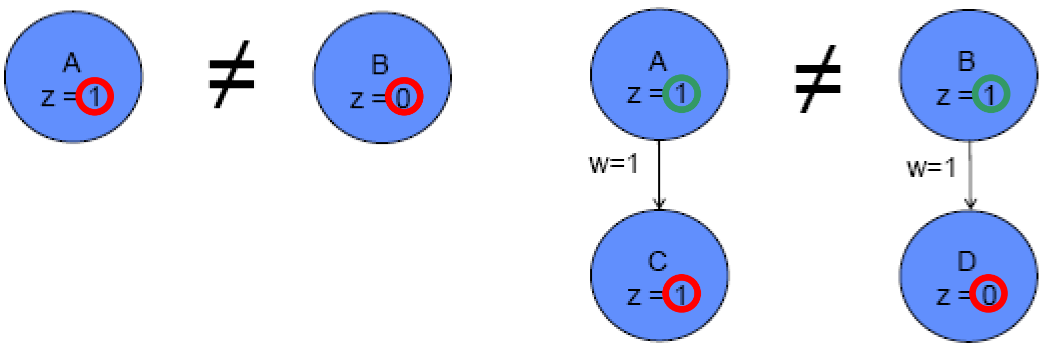
Present state		Next state
Name:	(CD Q)	$(CD\ Q)^{+}$
A	00 0	01 0 C ↑ 10 0 E
B	00 1	01 1 D ↑ 10 0 E
C	01 0	00 0 A ↑ 11 1 H
D	01 1	00 1 B ↑ 11 1 H
E	10 0	↓ 00 0 A 11 0 G
F	10 1	↓ 00 0 A 11 1 H
G	11 0	↓ 01 1 D 10 0 E
H	11 1	↓ 01 1 D 10 1 F

Present state	Next state CD= 00	01	11	10	Output Q
A	A	C	-	E	0
B	B	D	-	E	1
C	A	C	H	-	0
D	B	D	H	-	1
E	A	-	G	E	0
F	A	-	H	F	1
G	-	D	G	E	0
H	-	D	H	F	1

Stable states are marked by the ring. Make sure that each column "CD" contains at least one stable state, otherwise you get an "oscillating" network for that input signal. Don't-care "-" is introduced where the input "CD" contains more than change in one input variable from the steady state for the line.

11.5 State minimization

A and B are *not* **equivalent** if …



Equivalence means that the states should be stable for the same input signals, and to have their ”do not care” for the same inputs - not to lose the flexibility for the continued minimization.

Kompatibility will be different for Moore or Mealy. For Moore-compatible machines it applies that the outputs must be equal, and the outputs of the follower states (all, if several) must also be equal. Otherwise, the two conditions are not compatible!

State minimization

Present state	Next state CD= 00	01	11	10	Output Q
A	A	C	-	E	0
B	B	D	-	E	1
C	A	C	H	-	0
D	B	D	H	-	1
E	A	-	G	E	0
F	A	-	H	F	1
G	-	D	G	E	0
H	-	D	H	F	1

We start with a block of all state
P_1 = (ABCDEFGH)

Thera are no **Equvivalent** states**,**
we then look at **Kompatibility**

The states are first divided in two blocks by output value. ACEG has output 0, BDFH has output 1.
P_2 = [ACEG][BDFH]

A and C has *same* follower state
(as don't-care can be utilized as H or E)

```
AC-E
ACH-
```

P_3 = [(AC)…][BDFH]

(For **compatibility** it's enough that *output* from the follower states are same, it need not be exactly the same state as it happens to be in this example.)

State minimization

Present state	Next state CD= 00	01	11	10	Output Q
A	A	C	-	E	0
B	B	D	-	E	1
C	A	C	H	-	0
D	B	D	H	-	1
E	A	-	G	E	0
F	A	-	H	F	1
G	-	D	G	E	0
H	-	D	H	F	1

E and G has *same* follower state
(as don't-care can be utilized as A or D)

A-G**E**

-D**G**E

P_3 = [(AC)(EG)][BDFH]

B and D has *same* follower state
(as don't-care can be utilized as H or E)

BD-E

B**D**H-

P_3 = [(AC)(EG)][(BD)…]

F and H has *same* follower state
(as don't-care can be utilized as A or D)

A-H**F**

-D**H**F

P_3 = (AC)(EG)(BD)(FH)

Four states are enough!

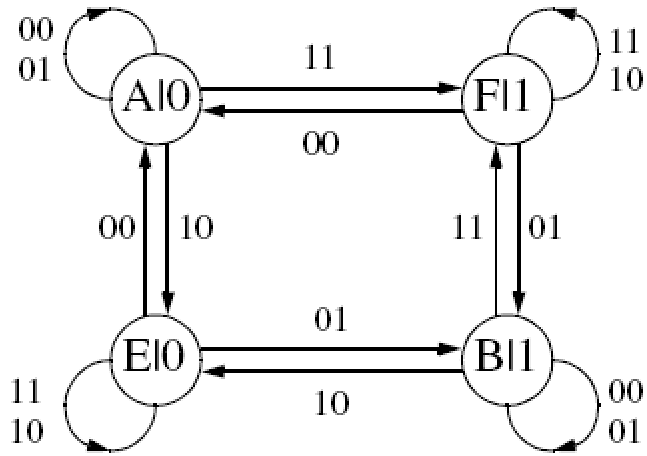
11.5 New Flow table

Nuvarande tillstånd	Nästa tillstånd CD= 00	01	11	10	Output Q
A	A	C	-	E	0
B	B	D	-	E	1
C	A	C	H	-	0
D	B	D	H	-	1
E	A	-	G	E	0
F	A	-	H	F	1
G	-	D	G	E	0
H	-	D	H	F	1

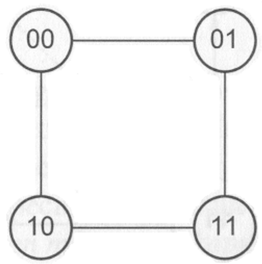
The new states are designated:
AC→**A**, EG→**E**, BD→**B**, FH→**F**.

Present state	Next state CD= 00	01	11	10	Output Q
A	A	A	F	E	0
B	B	B	F	E	1
E	A	B	E	E	0
F	A	B	F	F	1

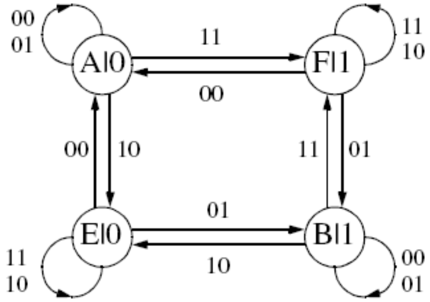
State diagram



11.5 State encoding

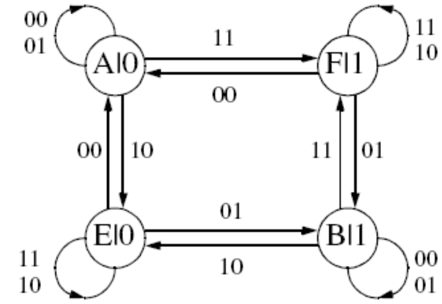


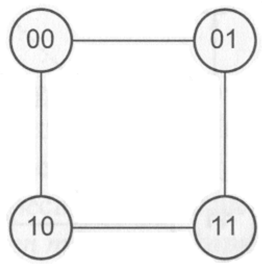
The states (q_1q_0), are placed in the corners of a Gray-coded square.
Eg. A=00, F=01, B=11, E=10.



William Sandqvist william@kth.se

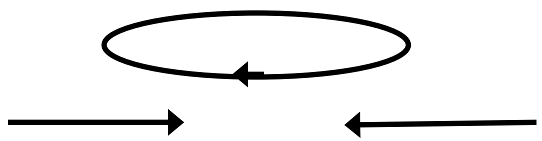
11.5 State encoding





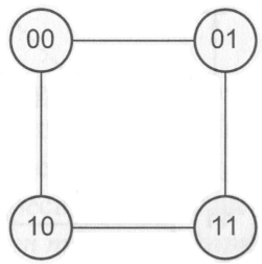
The states (q_1q_0), are placed in the corners of a Gray-coded square.
Eg. A=00, F=01, B=11, E=10.

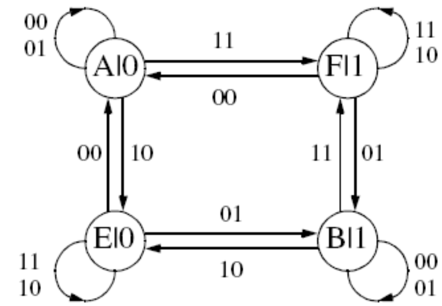
Although all "rotations" and "reflections" of the code is valid state encodings.



William Sandqvist william@kth.se

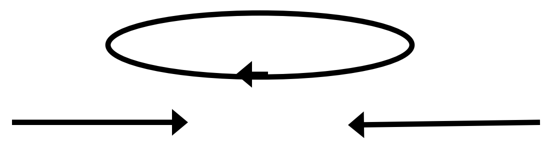
11.5 State encoding





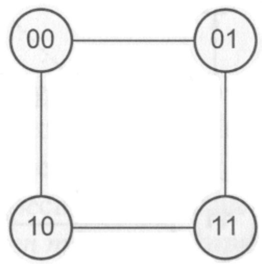
The states (q_1q_0), are placed in the corners of a Gray-coded square.
Eg. A=00, F=01, B=11, E=10.

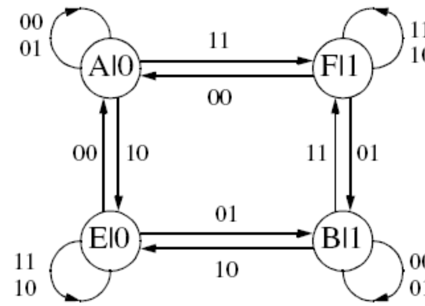
Although all "rotations" and "reflections" of the code is valid state encodings.



A	**F**	**B**	**E**	**A**	**F**	**B**	**E**
00	01	11	10	10	11	01	00
01	11	10	00	00	10	11	01
11	10	00	01	01	00	10	11
10	00	01	11	11	01	00	10

11.5 State encoding





The states (q_1q_0), are placed in the corners of a Gray-coded square.
Eg. A=00, F=01, B=11, E=10.

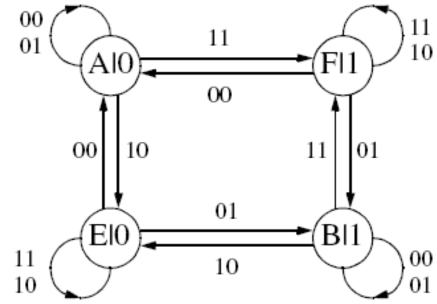
Although all "rotations" and "reflections" of the code is valid state encodings.

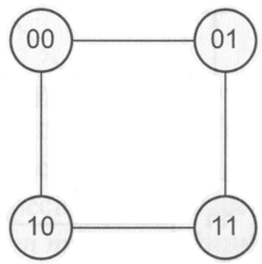
A	F	B	E
00	01	11	10
01	11	10	00
11	10	00	01
10	00	01	11

A	F	B	E
10	11	01	00
00	10	11	01
01	00	10	11
11	01	00	10

This will be our chosen arbitrarily state encoding.

11.5 State encoding





The states (q_1q_0), are placed in the corners of a Gray-coded square. Eg. A=00, F=01, B=11, E=10.

Although all "rotations" and "reflections" of the code is valid state encodings.

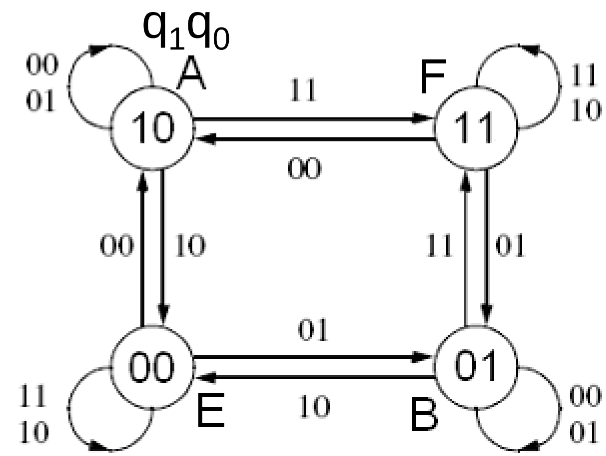
A	**F**	**B**	**E**
00	01	11	10
01	11	10	00
11	10	00	01
10	00	01	11

A	**F**	**B**	**E**
10	11	01	00
00	10	11	01
01	00	10	11
11	01	00	10

This will be our chosen arbitrarily state encoding.

Is this the best state encoding? Extensive search (= try all) is often the only solution for those who want to know!

11.5 Exitation table



Present state	Next state CD=				Output Q
	00	01	11	10	
A	A	A	F	E	0
B	B	B	F	E	1
E	A	B	E	E	0
F	A	B	F	F	1

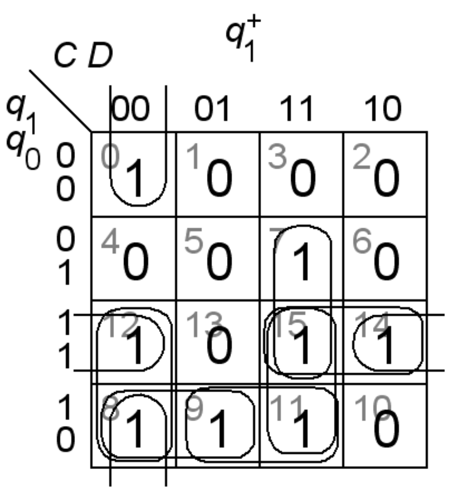
Present state q_1q_0	Next state CD=				Output Q
	00	01	11	10	
10	$\boxed{10}$	$\boxed{10}$	11	00	0
01	$\boxed{01}$	$\boxed{01}$	11	00	1
00	10	01	$\boxed{00}$	$\boxed{00}$	0
11	10	01	$\boxed{11}$	$\boxed{11}$	1

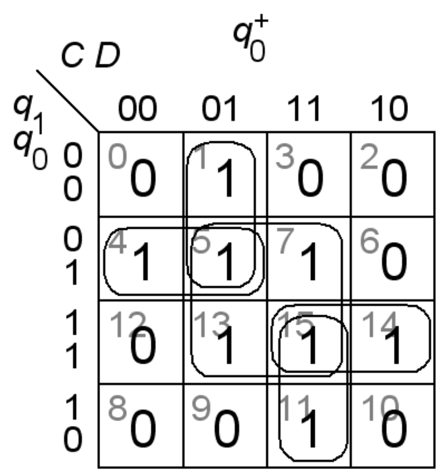
11.5 Karnaugh maps

Present state q_1q_0	Next state CD= 00	01	11	10	Output Q
10	10	10	11	00	0
01	01	01	11	00	1
00	10	01	00	00	0
11	10	01	11	11	1

On K-map-form:

Present state q_1q_0	Next state CD= 00	01	11	10	Output Q
00	10	01	00	00	0
01	01	01	11	00	1
11	10	01	11	11	1
10	10	10	11	00	0



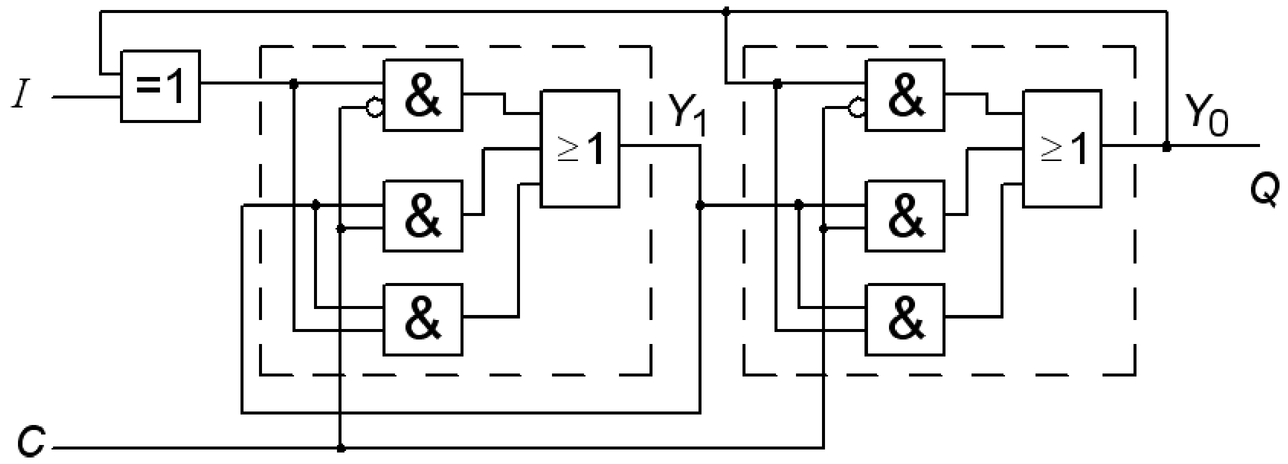


$$Q = q_0$$

$$q_1^+ = CDq_1 + CDq_0 + \overline{C}\,\overline{D}\,q_1 + \overline{C}\,\overline{D}\,\overline{q}_0 + q_1\overline{q}_0\overline{C} + q_1\overline{q}_0 D + q_1q_0C + q_1q_0\overline{D}$$

$$q_0^+ = q_0D + \overline{q}_1q_0\overline{C} + \overline{q}_1CD + q_1CD + q_1q_0C$$

Ex 11.6 Analyze



Analyze the above circuit.

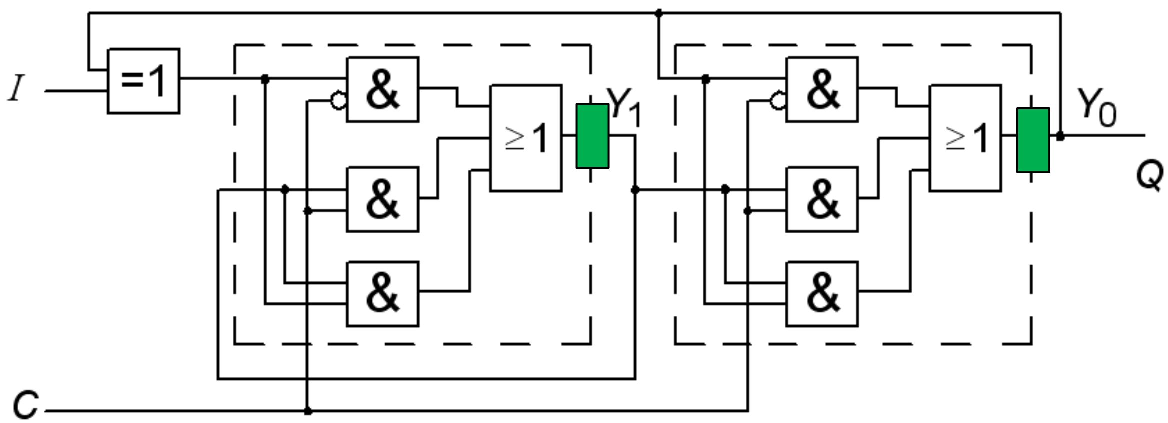
a) Derive the Boolean expressions for the state variables Y_1 and Y_0.

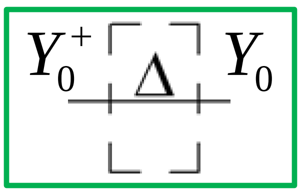
b) Derive the exitations table. Which function (dashed) are in the inner loops.

c) Derive the flow table, assign symbolic states and draw FSM.

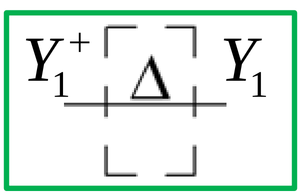
d) Which flip-flop does this correspond to?

11.6 Boolean equations



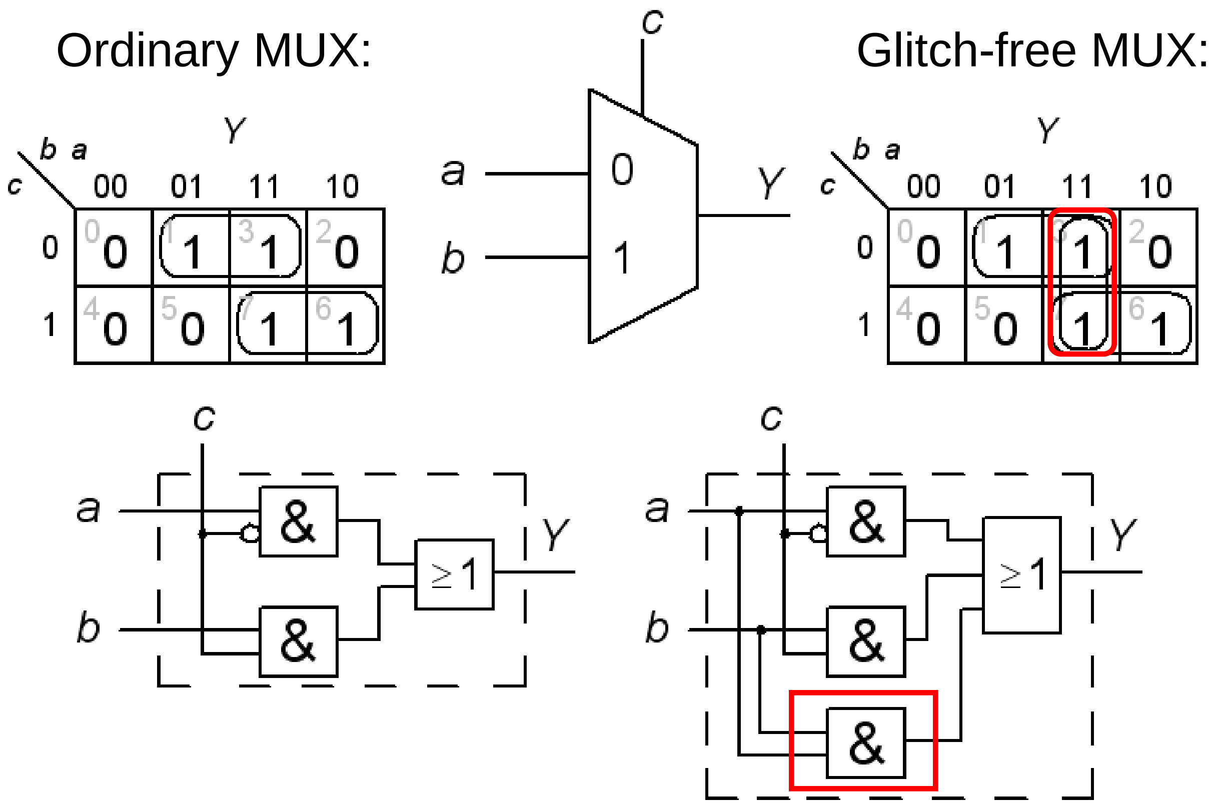


$$Y_0^+ = Y_0 Y_1 + Y_0 \overline{C} + Y_1 C$$



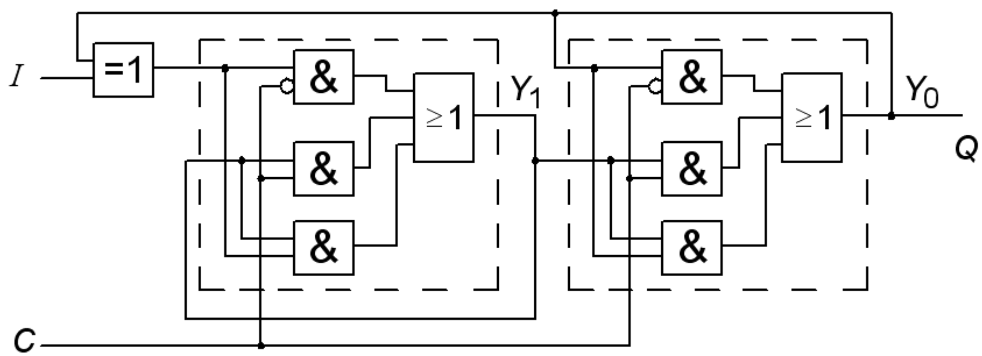
$$Y_1^+ = Y_1(Y_0 \oplus I) + (Y_0 \oplus I)\overline{C} + Y_1 C$$

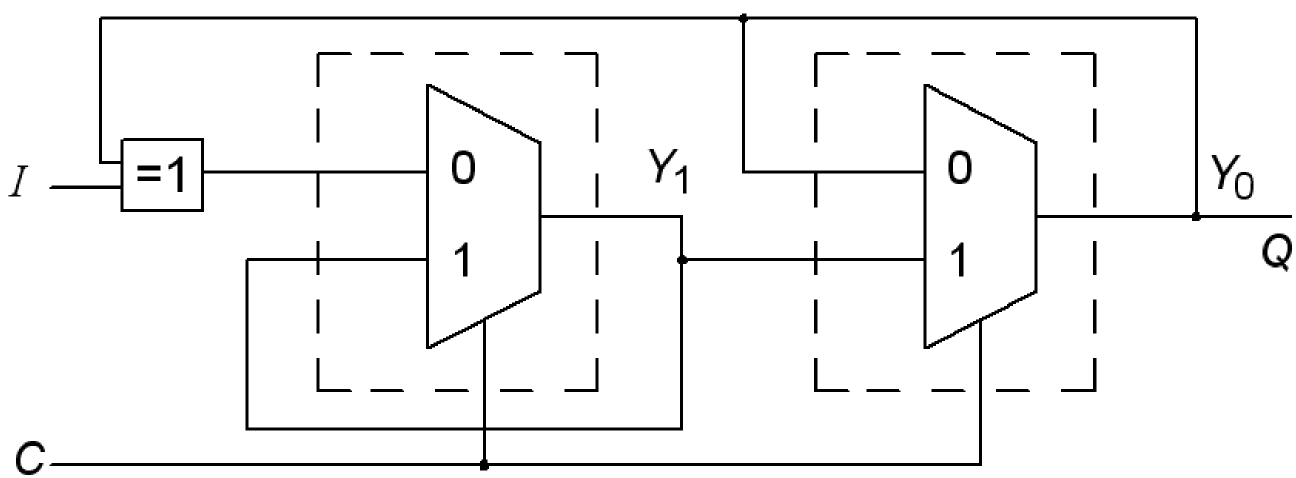
11.6 Glitch-free MUX?



11.6 Two Glitch-free MUXes

The network may be seen as composed of two glitch-free MUXes. This fact can be used if one wants to reason about the circuit's function.





11.6 Boolean equations

We use the Boolean functions to derive the function.

$$Q = Y_0$$

$$\boxed{Y_1^+} = Y_1(Y_0 \oplus I) + (Y_0 \oplus I)\overline{C} + Y_1 C =$$

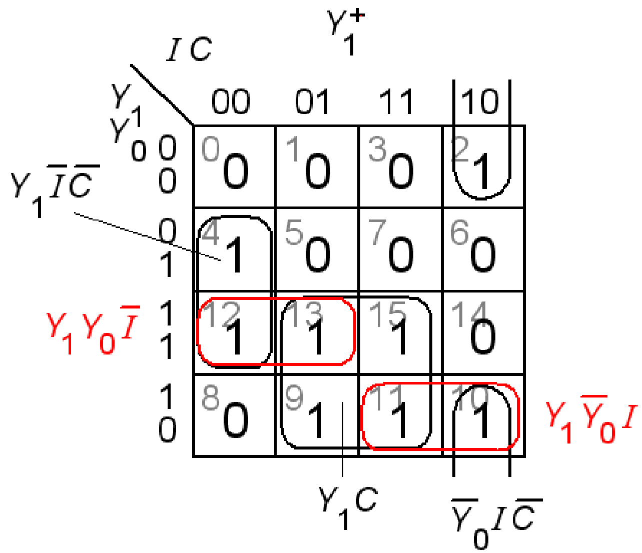
$$= Y_1(Y_0 \overline{I} + \overline{Y}_0 I) + (Y_0 \overline{I} + \overline{Y}_0 I)\overline{C} + Y_1 C =$$

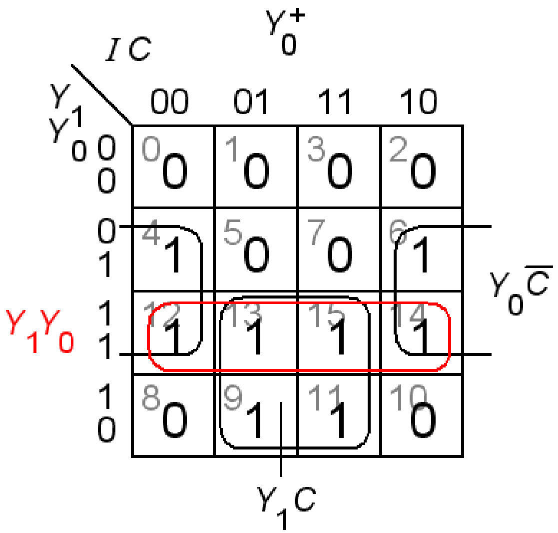
$$= \boxed{Y_1 Y_0 \overline{I} + Y_1 \overline{Y}_0 I + Y_0 \overline{I}\,\overline{C} + \overline{Y}_0 I \overline{C} + Y_1 C}$$

$$\boxed{Y_0^+} = \boxed{Y_0 Y_1 + Y_0 \overline{C} + Y_1 C}$$

11.6 Excitation table

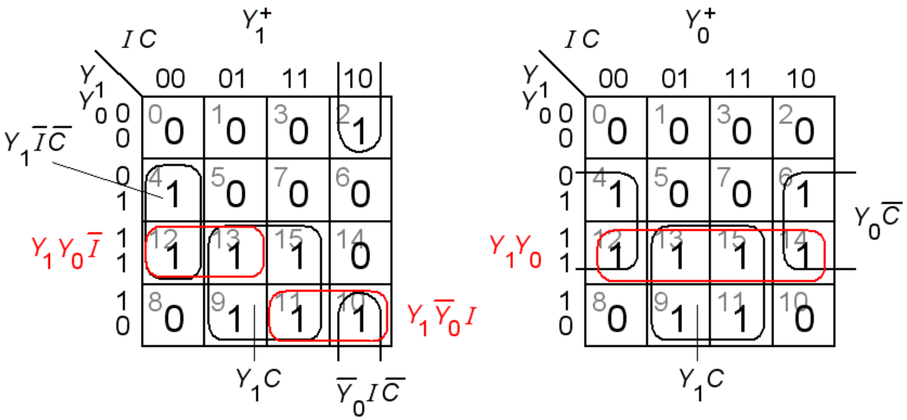
$$Y_1^+ = Y_1 Y_0 \overline{I} + Y_1 \overline{Y}_0 I + Y_0 \overline{I}\,\overline{C} + \overline{Y}_0 I \overline{C} + Y_1 C \qquad Y_0^+ = Y_0 Y_1 + Y_0 \overline{C} + Y_1 C$$





Red marked groupings are circuit Hazard Cover

11.6 Exitation table



Impossible states are denoted by ~~strikethrough~~. These are states that, as to be reached, would require two changes of input the signals from the stable state of the current row.

Present state Y_1Y_0	Next state IC= 00	01	11	10	Output Q
00	$\boxed{00}$	$\boxed{00}$	$\boxed{00}$	10	0
01	11	~~00~~	00	$\boxed{01}$	1
11	$\boxed{11}$	$\boxed{11}$	$\boxed{11}$	01	1
10	00	~~11~~	11	$\boxed{10}$	0

$$Q = Y_0$$

IC $10 \rightarrow 01$ is an impossible simultaneous change of the input signals.

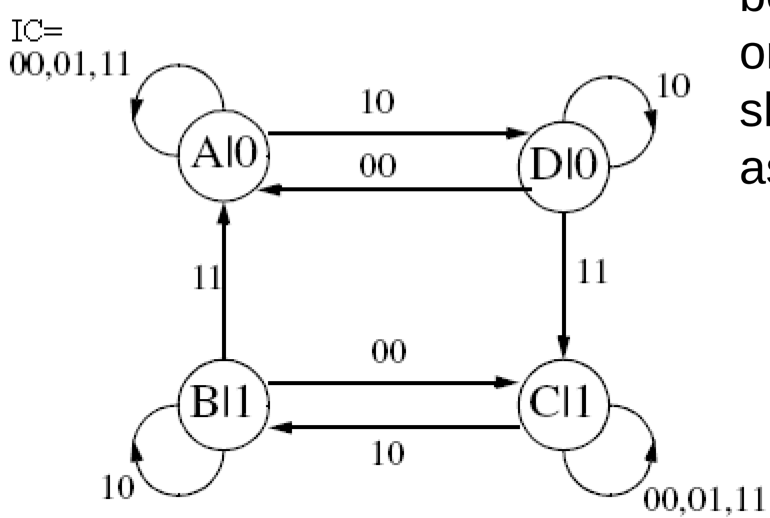
11.6 Flow table

Present state Y_1Y_0	Next state IC= 00	01	11	10	Output Q
A	A	A	A	D	0
B	C	~~A~~	A	B	1
C	C	C	C	B	1
D	A	~~C~~	C	D	0

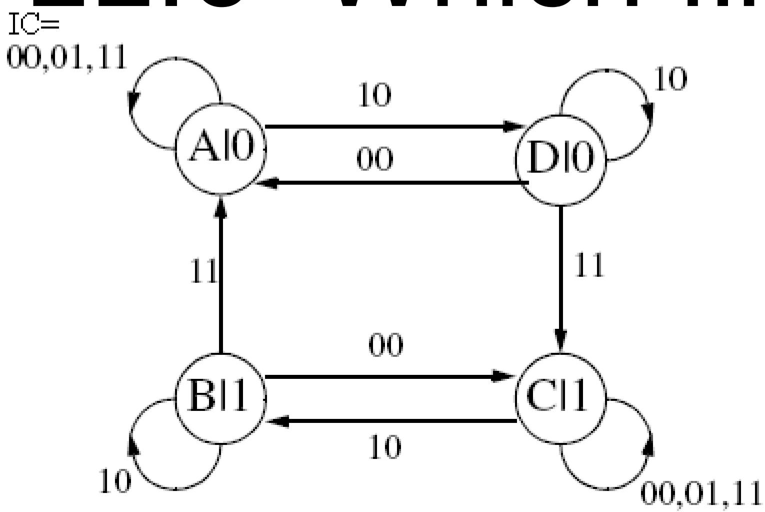
Present state Y_1Y_0	Next state IC= 00	01	11	10	Output Q
00	00	00	00	10	0
01	11	~~00~~	00	01	1
11	11	11	11	01	1
10	00	~~11~~	11	10	0

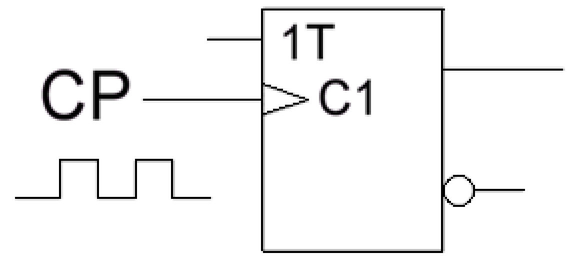
The impossible states (strikethrough text) could be used as don't-care if one at another time should change the state assignement.

State diagram:



11.6 Which flip-flop?





If I = **1** and C are clockpulses 1,0,1,0… the sequence is:

IC: **1**0 **1**1 **1**0 **1**1, D-C-B-A-D-C-B-A Q: 0-1-1-0-0-1-1-0
The flip-flop toggles on positive edge (↑) from C.

If I = 0 it becomes instead ”the same output”
A→A and D→A Q = 0
C →C and B→C Q = 1

The flip-flop changes state at the transitions from C = 0 to C = 1, so it is positive edgetriggered (↑) T-flip-flop (I = *T*).