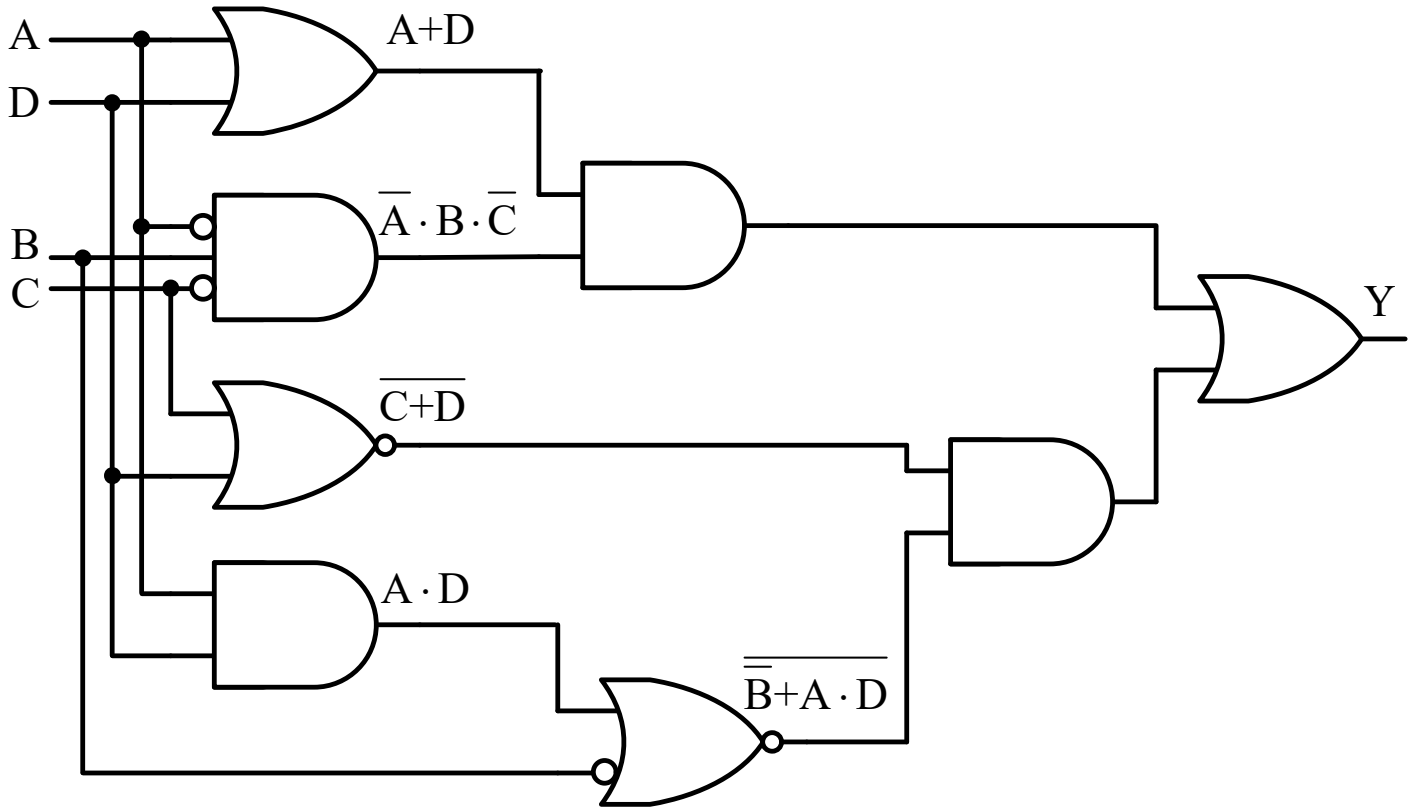
Osnovi računarstva I

Prekidačke mreže

- Posmatrajmo sljedeću složenu prekidačku funkciju:

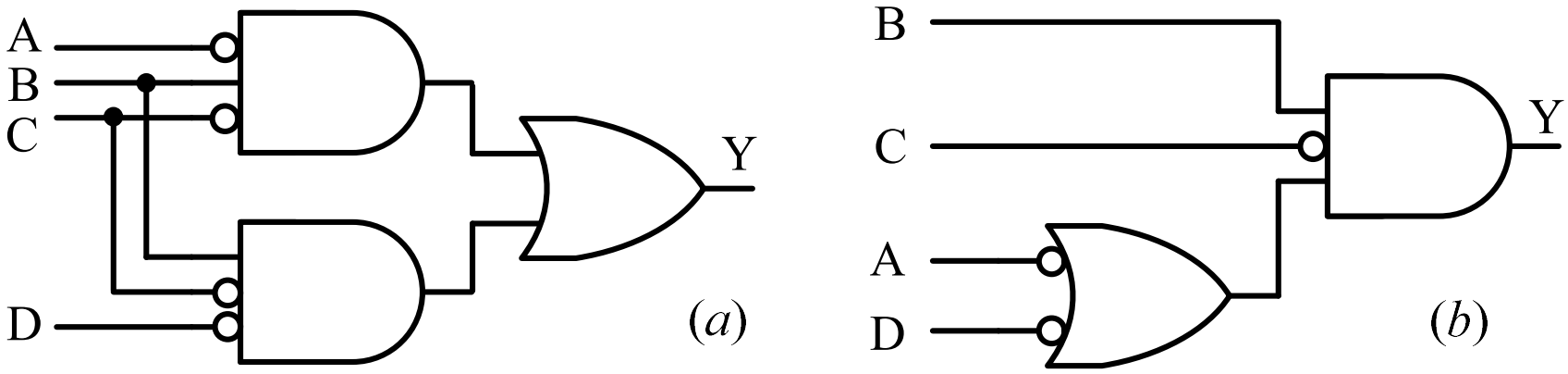
$$Y = (A + D) \cdot \overline{A} \cdot B \cdot \overline{C} + \overline{C + D} \cdot \overline{\overline{B} + A \cdot D}$$

- Logička šema prekidačke mreže može se dobiti **direktnim preslikavanjem** prekidačke funkcije u odgovarajući dijagram elementarnih logičkih kola:

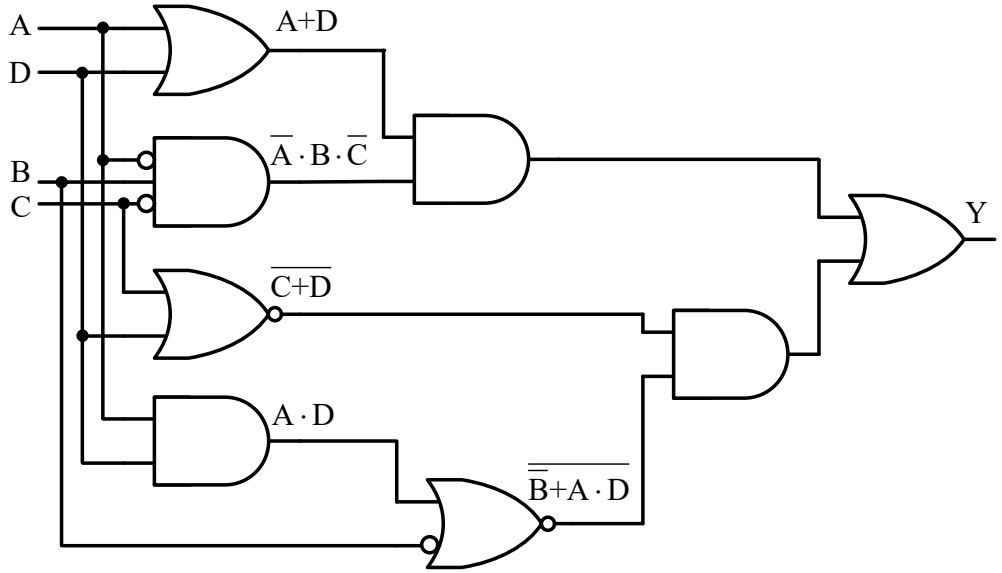


- Veoma često, izlaz Y iste logičke vrijednosti može se realizovati pomoću jednostavnije mreže logičkih kola (realizacijom ekvivalentne prekidačke f-je dobijene minimizacijom, algebarski ili pomoću Karnoovih mapa):

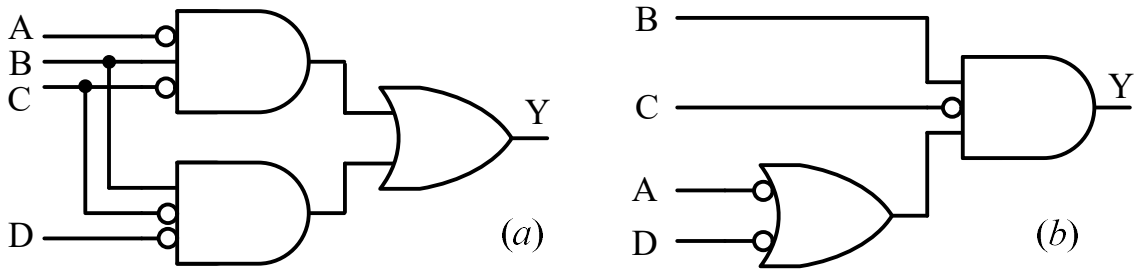
$$Y = \overline{A}B\overline{C} + B\overline{C}\overline{D} = B\overline{C} \cdot (\overline{A} + \overline{D})$$



- **Prekidačke mreže za implementaciju funkcije ne razlikuju se samo u pogledu broja i tipa upotrijebljenih logičkih elemenata, već i po broju nivoa (stepeni) u strukturi mreže**
- **Stepen prekidačke mreže određen je najvećim br. redno vezanih log. kola kroz koja neki od ulaznih signala treba da prođe do izlaza mreže**
- Četvorostepena:



- Dvostepene:



REALIZACIJA NORMALNIH FORMI PREKIDAČKIH FUNKCIJA UPOTREBOM LOGIČKIH NI I NILI KOLA

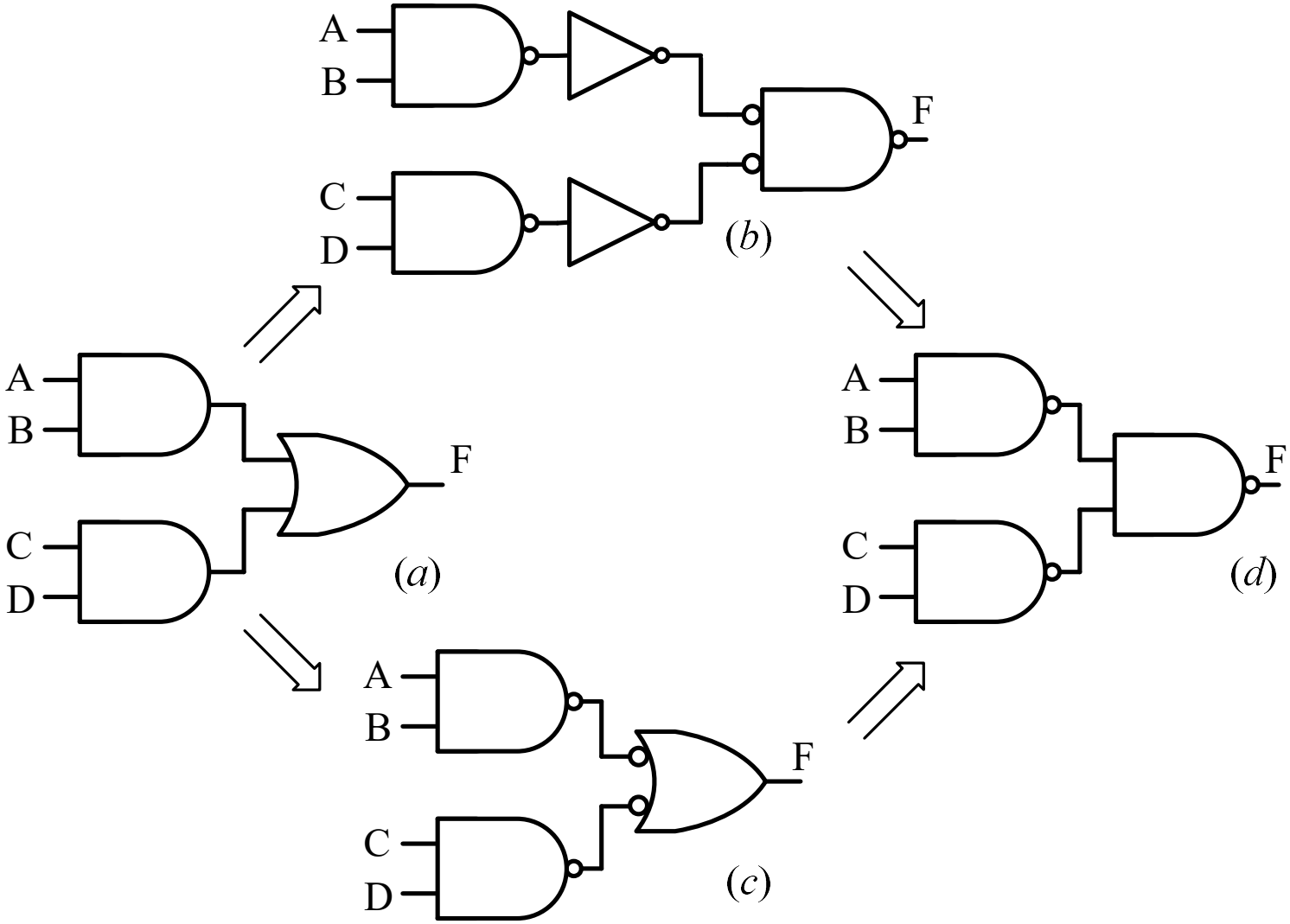
- U slučaju logičke funkcije oblika zbira logičkih proizvoda, implementacija **dvostepene** prekidačke mreže može se izvršiti **upotrebom isključivo logičkih NI kola**:

$$F = \mathrm{A} \cdot \mathrm{B} + \mathrm{C} \cdot \mathrm{D} = \overline{\overline{\mathrm{A} \cdot \mathrm{B}} + \overline{\mathrm{C} \cdot \mathrm{D}}} = \overline{\overline{\mathrm{A} \cdot \mathrm{B}} \cdot \overline{\mathrm{C} \cdot \mathrm{D}}}$$

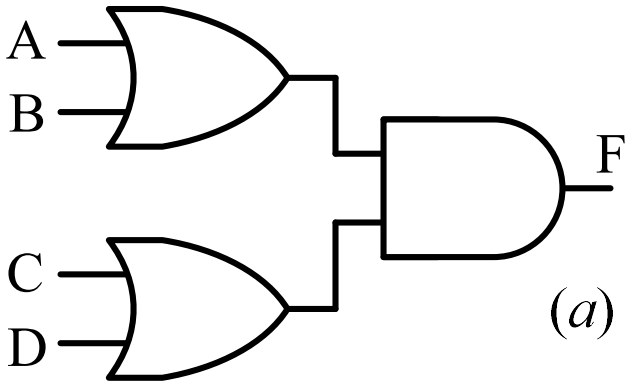
- Analogno tome, funkcije oblika proizvoda logičkih zbirova mogu se implementirati **dvostepenom** prekidačkom mrežom, **koristeći isključivo logička NILI kola**:

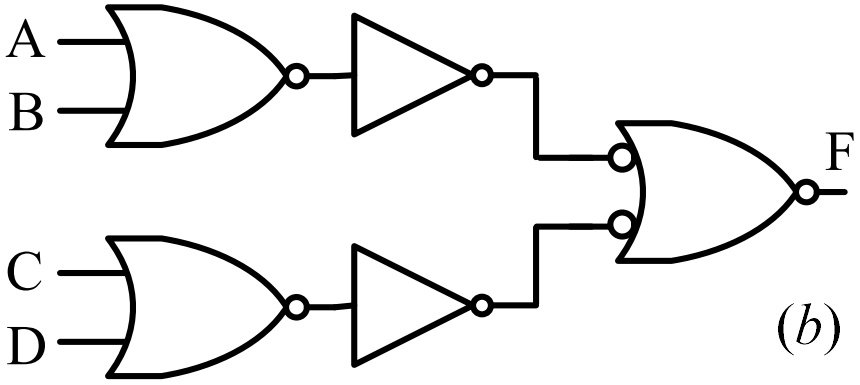
$$F = (\mathrm{A} + \mathrm{B}) \cdot (\mathrm{C} + \mathrm{D}) = \overline{\overline{\mathrm{A} + \mathrm{B}} \cdot \overline{\mathrm{C} + \mathrm{D}}} = \overline{\overline{\mathrm{A} + \mathrm{B}} + \overline{\mathrm{C} + \mathrm{D}}}$$

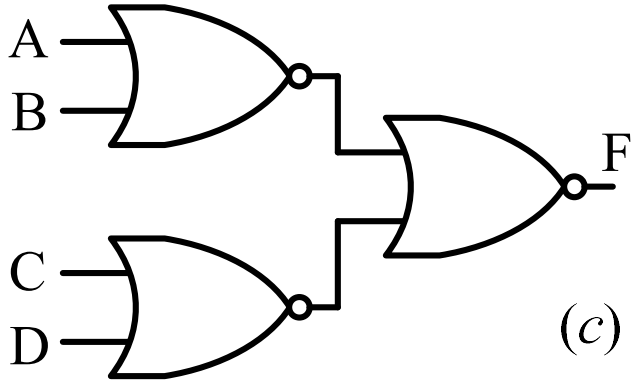
$$F = A \cdot B + C \cdot D$$



$$F = (A + B) \cdot (C + D)$$



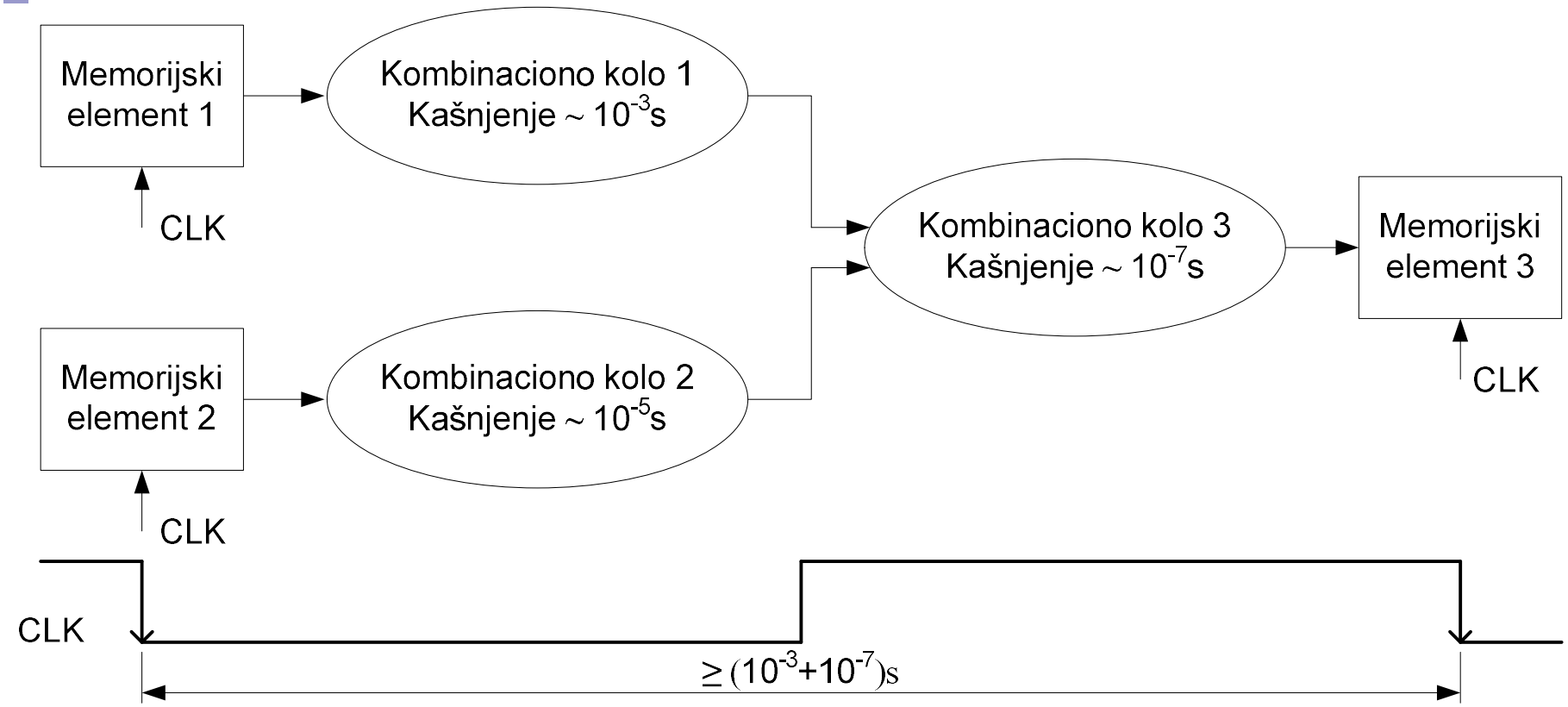




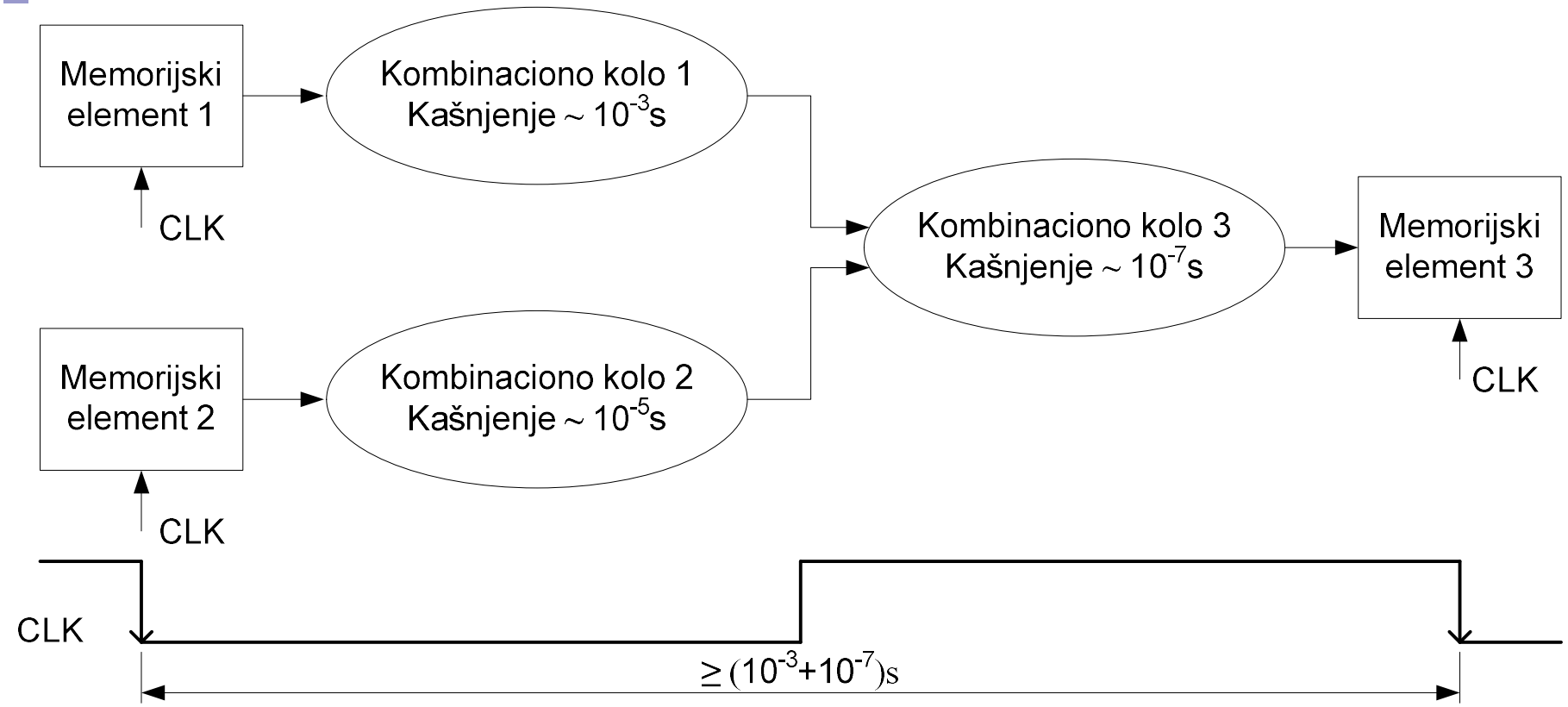
Osnovi računarstva I

Osnovni digitalni sistemi (leč i flip-flop)

- Do sada smo posmatrali tzv. ***idealni slučaj*: Kolo reaguje u istom trenutku kada se promijeni signal na njegovom ulazu**
- **Medjutim, prolaskom ulaznog signala kroz jedno ili veći broj logičkih kola unosi se određeno vremensko kašnjenje !!**
- Kašnjenja po jednom elektronskom elementu obično su veoma, veoma mala, reda veličine nekoliko nanosekundi (10^{-9} s)
- Prolaskom signala kroz veliki broj elektronskih elemenata ukupna kašnjenja mogu postati značajna
- **Da bi sistem ispravno funkcionisao nekada je potrebno uvesti *kontrolni* signal !!**
- Često se u mrežama koriste memorijski elementi – kod njih je naročito važna kontrola upisa
- Osnovni memorijski elementi su ***leč*** *(latch)* i ***flip-flop***
- Pomoću osnovnih memorijskih elemenata grade se složeniji memorijski elementi, npr. ***registri***



- Pomoću kontrolnih signala vrši se **sinhronizacija** između mreža različitog stepena složenosti, odnosno mreža koje unose različita kašnjenja
- Sinhronizacija se najčešće vrši pomoću kontrolnog signala u obliku periodične povorke pravougaonih impulsa, koju generiše vremenski nezavisni uređaj, a koja se naziva ***takt*** (eng. Clock – CLK)
- ***Uzlazna*** i ***silazna*** ivica taktnog impulsa

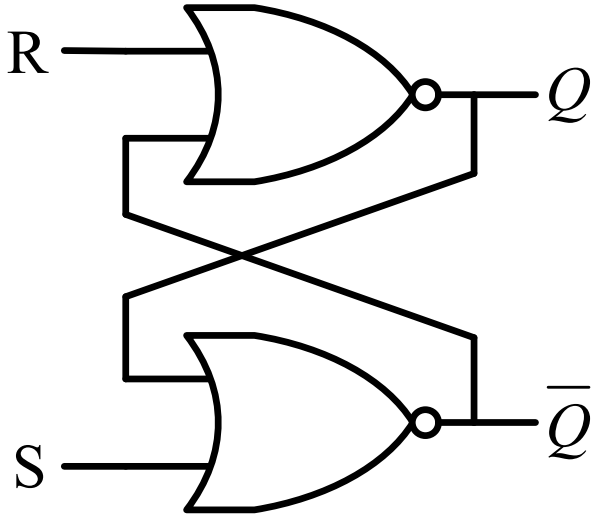


- Takt se kreira tako da omogući izvršavanje najduže akcije u sistemu (akcija koja unosi najveće kašnjenje) za vrijeme trajanja jednog taktnog intervala
- Zbirno kašnjenje kombinacionih kola 1 i 3 (kao veće od zbirnog kašnjenja kombinacionih kola 2 i 3) određuje vremensko ponašanje sistema, odnosno trajanje taktnog intervala

OSNOVNI MEMORIJSKI ELEMENTI – LEČ I FLIP-FLOP –

- **Osnovni memorijski element predstavlja jedinicu memorije koja može čuvati jedan bit informacije**
- Memorijski element čiji sadržaj se mijenja sa promjenom nivoa signala na njegovim ulazima, bez uticaja bilo kakvog kontrolnog signala, u literaturi se obično naziva ***leč*** (eng. *latch*)
- Ako postoji kontrolni signal koji će upravljati procesom upisivanja u leč, onda se takav leč naziva ***upravljivi* leč**.
- Obično se u upravljivi leč može upisivati kada je kontrolni signal na nivou logičke jedinice
- U praksi je često potrebno da se upis u memorijski element izvršava u tačno određenim, diskretnim vremenskim trenucima koji su odredjeni promjenom stanja kontrolnog signala (uzlazna ili silazna ivica takta)
- Memorijski elementi koji reaguju na ivicu kontrolnog signala nazivaju se ***edge-trigerred flip-flopovi***

RS LEČ



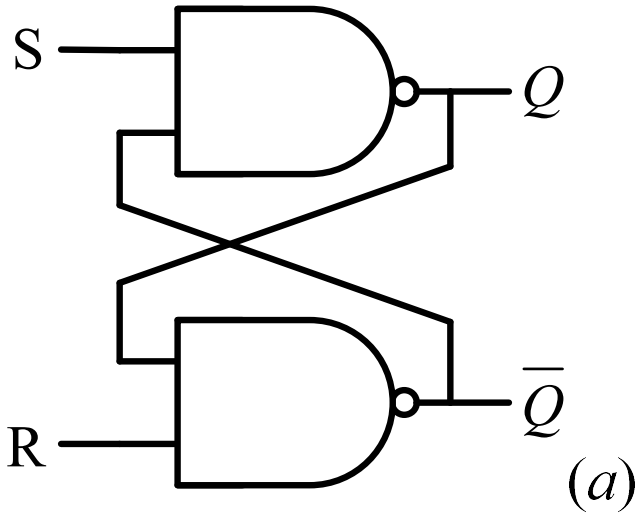
(*a*)

R	S	Q_{t+1}	$\overline{Q}_{t+1}$	
0	0	Q_t	$\overline{Q}_t$	← Zatečeno stanje
0	1	1	0	← Set
1	0	0	1	← Reset
1	1	0	0	← Nedef. stanje

(*b*)

- Varijanta 1: upotrebom logičkih **NILI** kola

RS LEČ



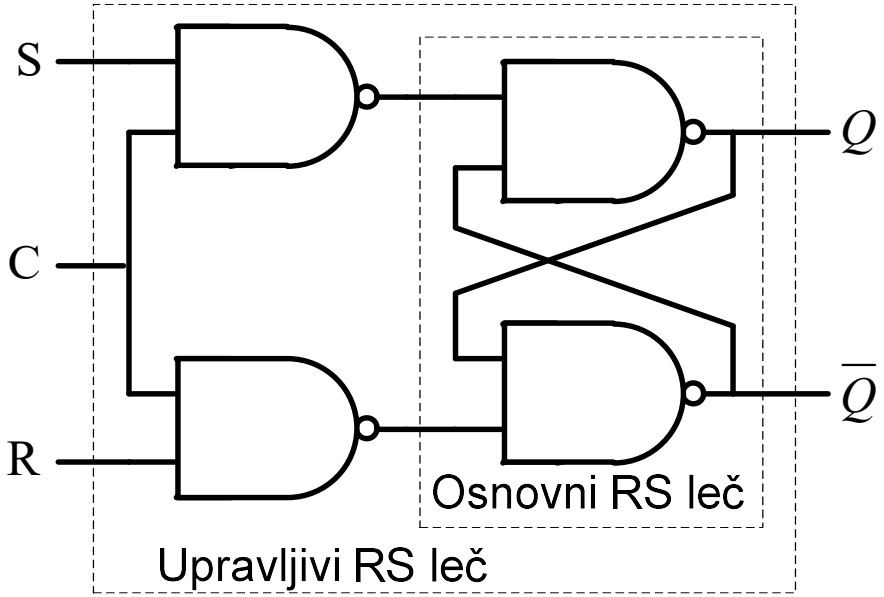
(a)

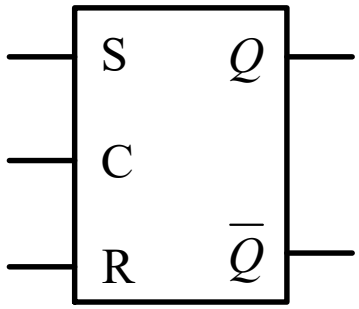
R	S	Q_{t+1}	$\bar{Q}_{t+1}$	
0	0	1	1	← Nedef.stanje
0	1	0	1	← Reset
1	0	1	0	← Set
1	1	Q_t	$\bar{Q}_t$	← Zatečeno stanje

(b)

- Varijanta 2: upotrebom **NI** kola
- Inverzna (negativna) logika na ulazima:
  Signali R i S su aktivni kada su na nivou logičke nule
- Iz gore navedenog razloga, RS leč sa međusobno povezanim NI kolima u literaturi se često naziva inv(R)inv(S) leč

UPRAVLJIVI RS LEČ





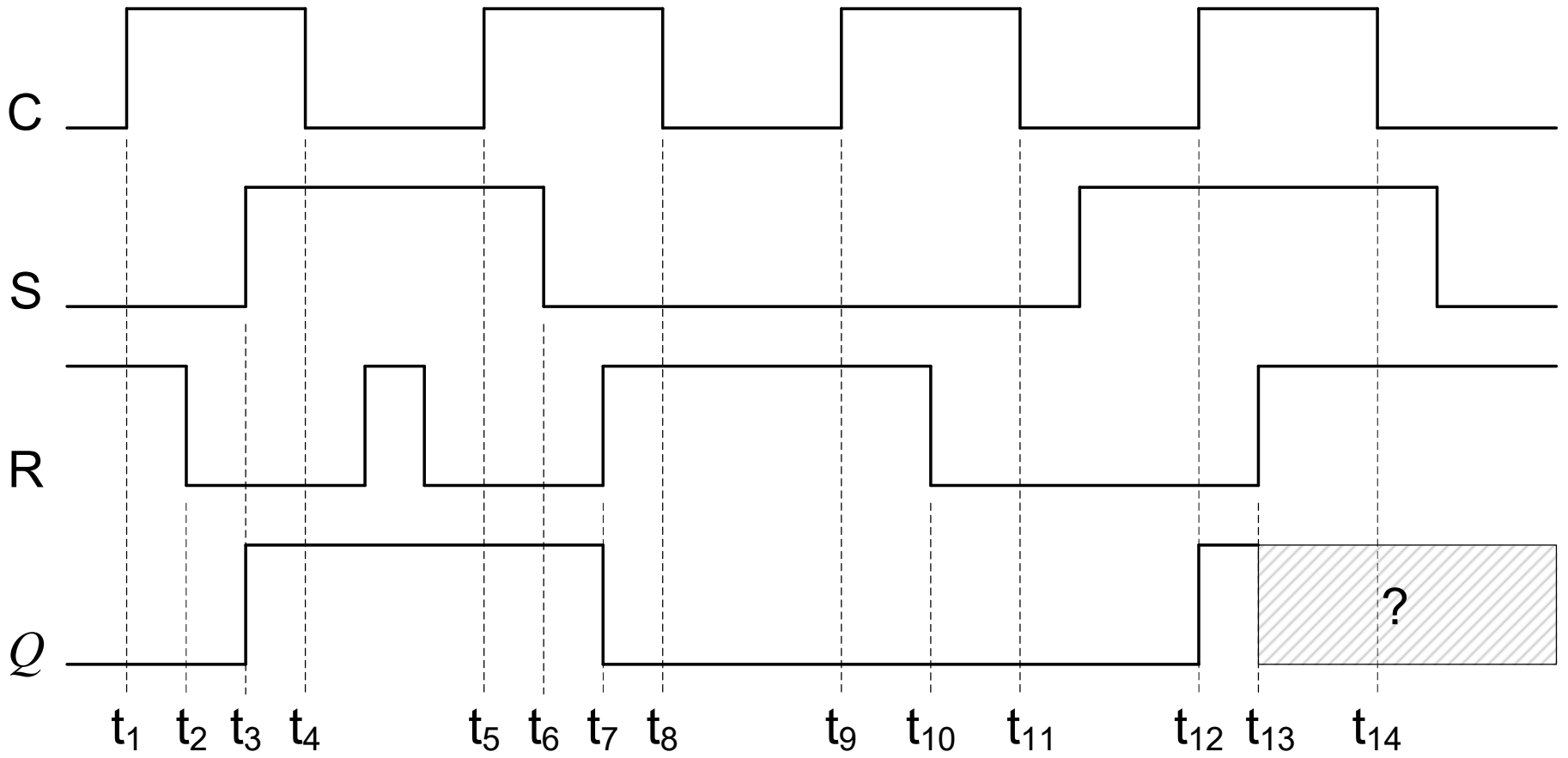
Osnovni:

R	S	Q_{t+1}	$\overline{Q}_{t+1}$	
0	0	1	1	← Nedef. stanje
0	1	0	1	← Reset
1	0	1	0	← Set
1	1	Q_t	$\overline{Q}_t$	← Zatečeno stanje

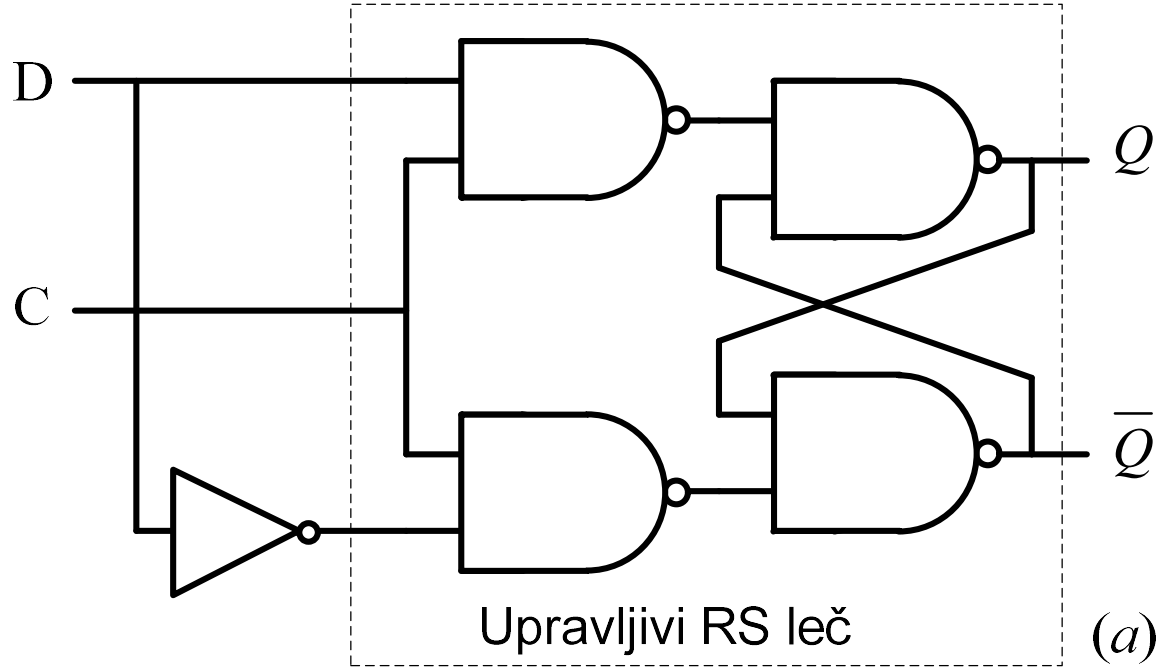
Upravljivi:

C	R	S	Q_{t+1}	
0	×	×	Q_t	← Zatečeno stanje
1	0	0	Q_t	← Zatečeno stanje
1	0	1	1	← Set
1	1	0	0	← Reset
1	1	1	Nedef.	← Nedef. stanje

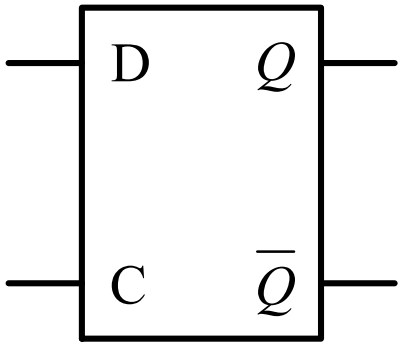
- Primjer:



UPRAVLJIVI D LEČ



(a)

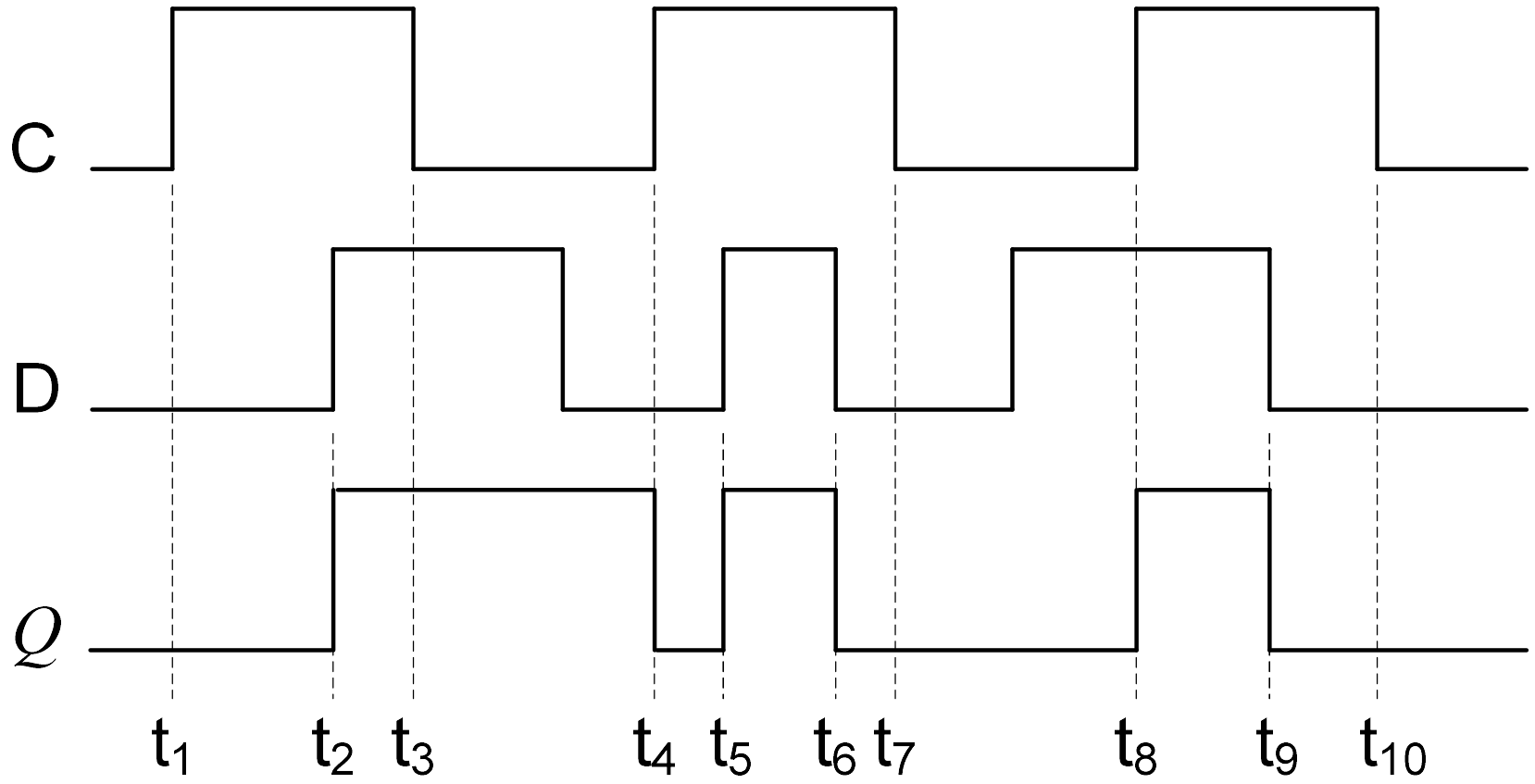


(b)

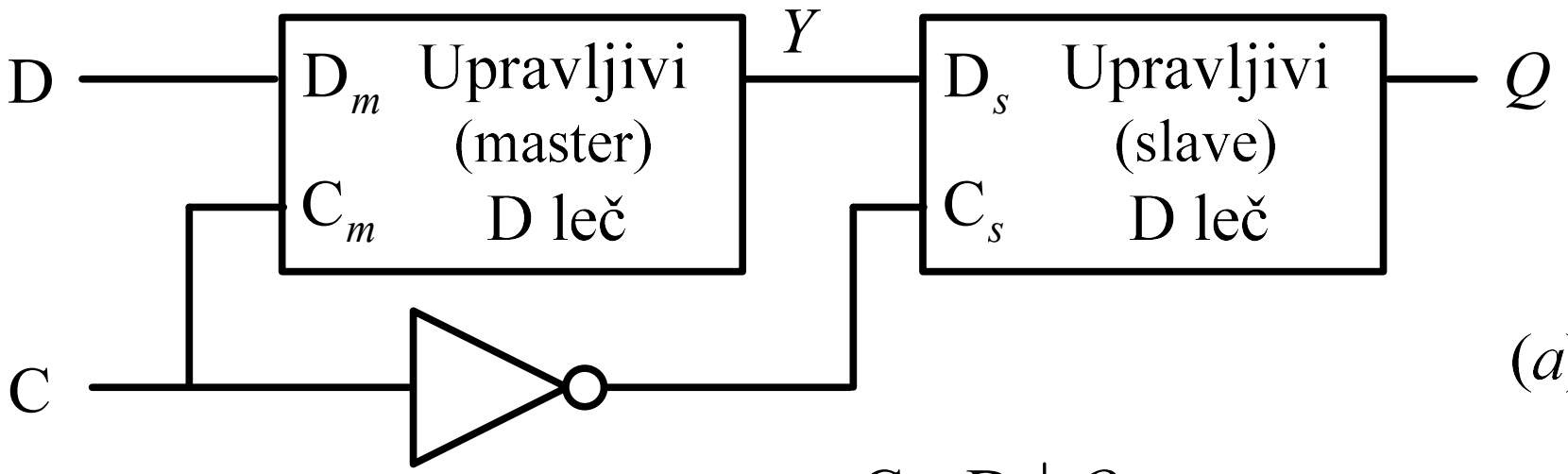
C	D	Q_{t+1}	
0	×	Q_t	← zatečeno stanje
1	0	0	← Reset
1	1	1	← Set

(c)

- Primjer:



D FLIP-FLOP



(a)

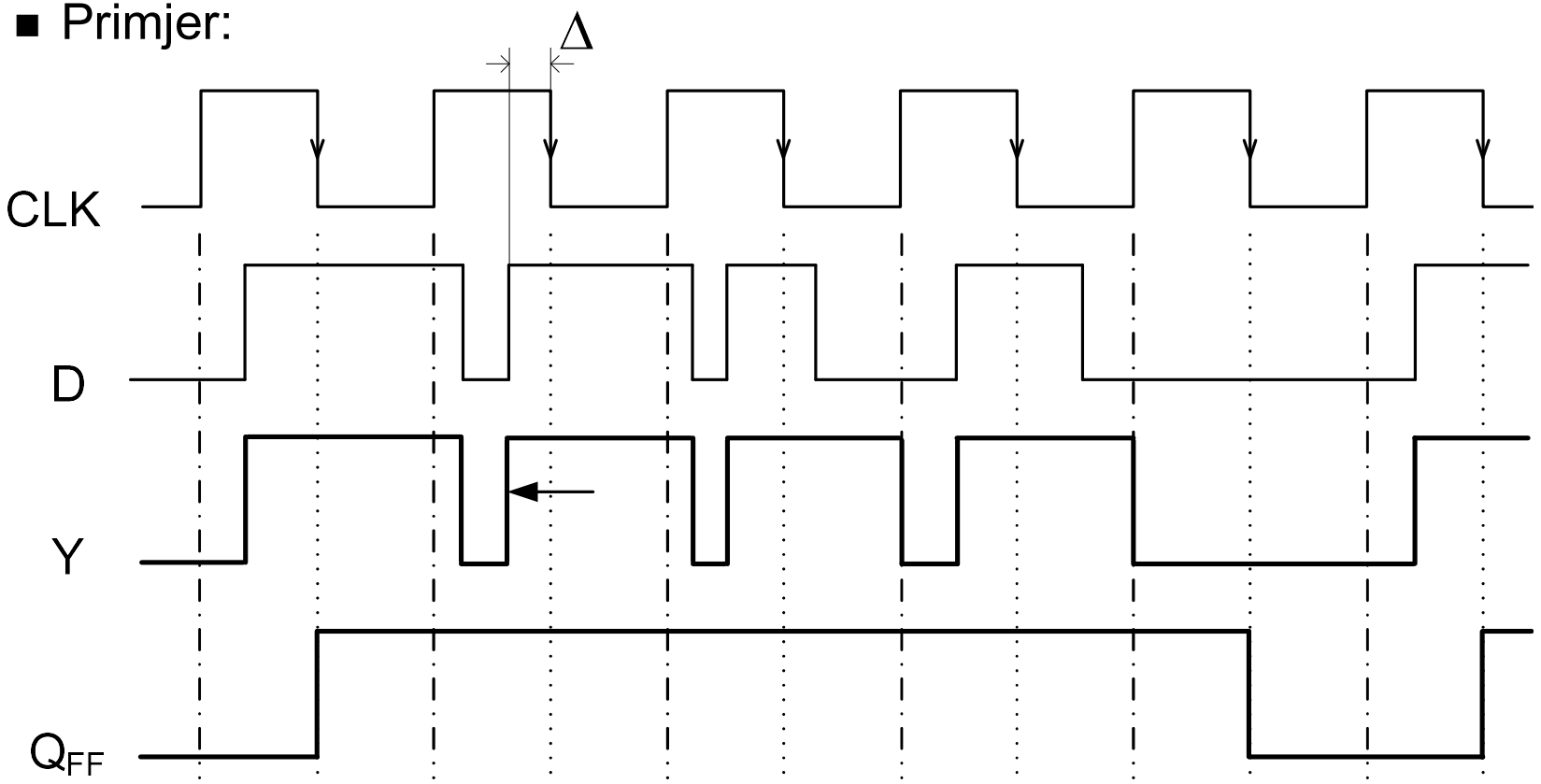
C	D	Q_{t+1}	
↓	0	0	← Reset
↓	1	1	← Set

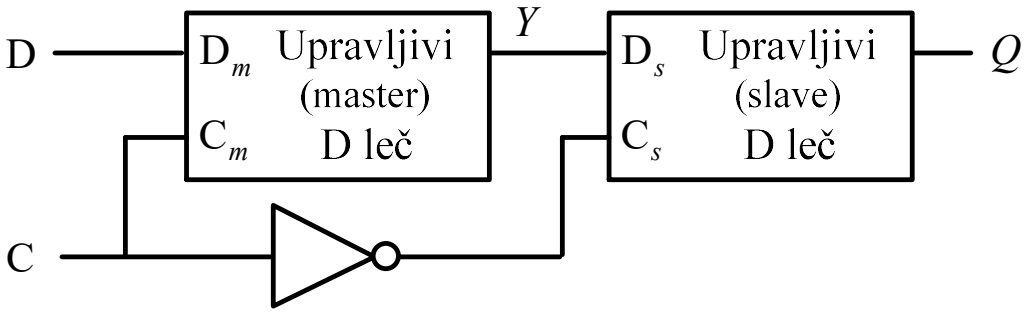
(b)

Silazna ivica

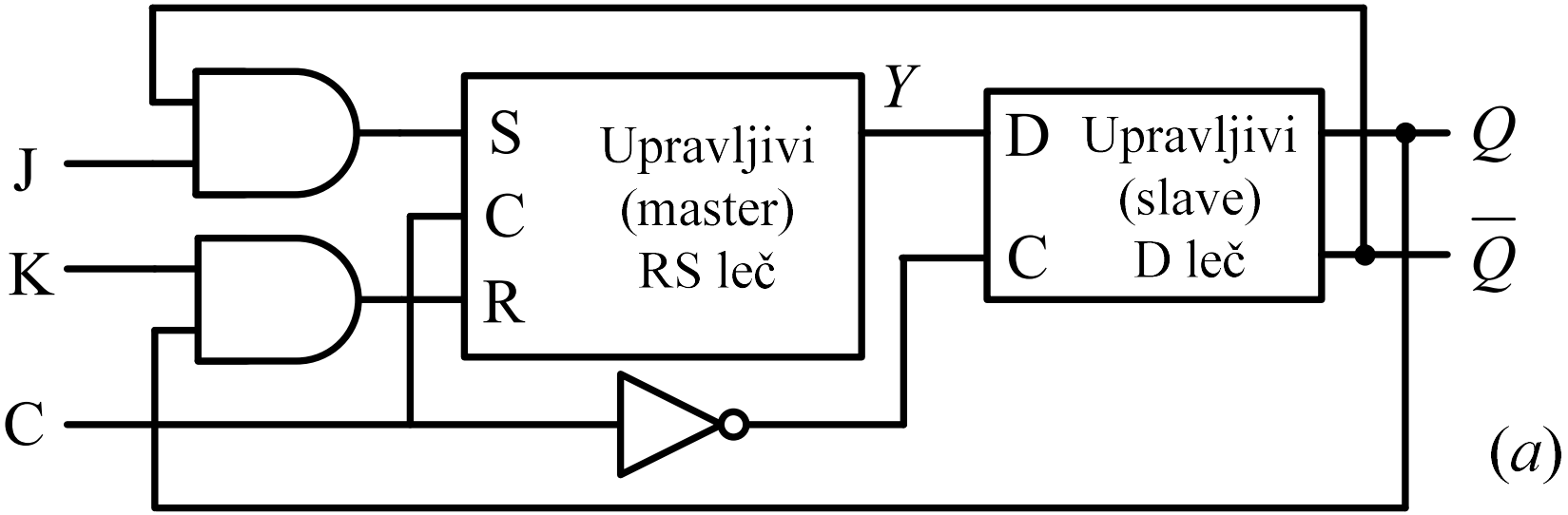
Uzlazna ivica

- Primjer:


Δ
CLK
D
Y
Q_{FF}


D
D_m Upravljivi (master) D leč
C_m
Y
D_s Upravljivi (slave) D leč
C_s
Q
C

JK FLIP-FLOP



(a)

C	J	K	Q_{t+1}	
↓	0	0	Q_t	← Zatečeno stanje
↓	0	1	0	← Reset
↓	1	0	1	← Set
↓	1	1	$\overline{Q}_t$	← Komplement

(b)

- Primjer: