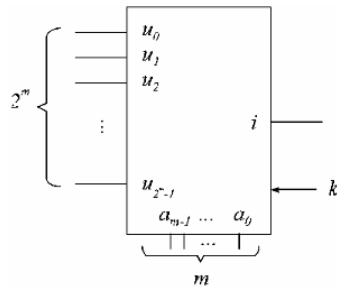


# 11. KOMBINACIJSKI SKLOPOVI SREDNJEG STUPNJA INTEGRACIJE

## 11.1. Selektor/multiplekser

- definirati funkciju selektora/multipleksera
- izvesti formulu za selektor/multiplekser
- opisati uporabu selektora/multipleksera

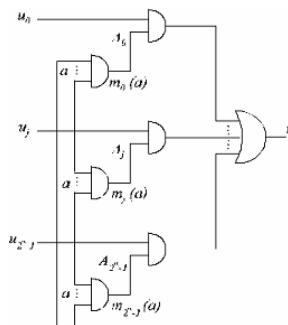


Multiplexser je logički sklop koji na informacijski izlaz  $i$  propušta vrijednost onog od  $n=2^m$  informacijskih ulaza  $u_0 \dots u_{n-1}$ , čiji je redni broj prisutan u prirodnom binarnom obliku na  $m$  adresnih

ulaza  $a_0 \dots a_{m-1}$ . Algebarski izraz multipleksera:  $i = \bigvee_{j=0}^{2^m-1} m_j \& u_j$ .

Multiplexser možemo koristiti kao selektor kada su vrijednosti adresnih varijabli stacionarne u odnosu na ulazne varijable. Dakle, funkcija multipleksera kao selektora je da ovisno o adresnim

ulazima propusti odgovarajući ulaz na izlaz.



Da bi neki ulaz bio proveden na izlaz treba prepoznati pripadnu kodnu riječ adresnih varijabli, te aktivnim signalom propustiti signal s odabranog ulaza. Kodna riječ se algebarski može prepoznati algebarskim izrazom, kojeg smo nazvali minterm, dakle konjunkcijom svih varijabli (sada adresnih). Minterm jedinicom prepozna pripadnu kodnu riječ pa možemo koristiti **I** vrata za propuštanje vrijednosti s izabranog ulaza. Za sve ostale kodne riječi vrijednost minterma je 0, **I** vrata su zatvorena, a na izlazu imamo 0. to omogućava povezivanje svih međurezultata na jedna **ILI** vrata.

Algebarski možemo pisati:

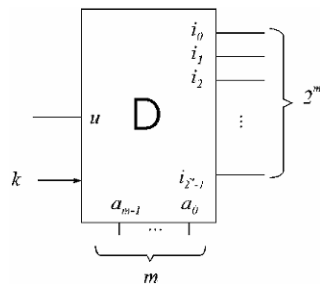
$$i = \bigvee_{j=0}^{2^m-1} m_j(a) u_j = \bigvee_{j=0}^{2^m-1} \overline{m_j(a)} u_j = \bigvee_{j=0}^{2^m-1} \overline{m_j(a)} u_j$$

Dvostrukom negacijom dobili smo mogućnost korištenja **NI** vrata.

Multiplexser se koristi kao selektor, za paralelno-serijsku konverziju i za realizaciju Booleovih funkcija. Ako su informacijski ulazi slobodno promjenljivi, a adresa stacionarna, izlaz će slijediti vrijednost sa odabranog ulaza. Obavili smo **selektiranje** ulaza na izlaz. Ako su informacijski ulazi stacionarni, a adrese mijenjamo u prirodnom binarnom nizu nekim ritmom, na izlazu će se pojaviti niz bita ulazne kodne riječi. Obavili smo **paralelno-serijsko konverziju**.

## 11.2. Dekoder/demultiplekser

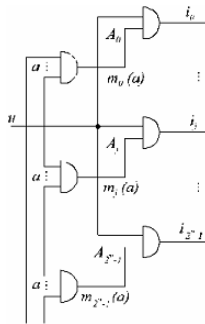
- definirati funkciju dekodera/demultipleksera
- izvesti formulu za dekodekser/demultiplekser
- opisati uporabu dekodera/demultipleksera



Demultiplekser je logički sklop koji vrijednost informacijskog ulaza **u** propušta na onaj od  $n=2^m$  informacijskih izlaza  $i_0 \dots i_{n-1}$ , čiji je redni broj prisutan u prirodnom binarnom obliku na  $m$  adresnih ulaza  $a_0 \dots a_{m-1}$ .

Algebarski izraz demultipleksera:  $i_j = m_j \& u$

Demultiplekser koristimo za razvođenje signala (kao deselektor), za serijsko-paralelnu pretvorbu, kao dekodekser i za realizaciju Booleovih funkcija.



Da bi ulaz bio proveden na neki izlaz, treba prepoznati pripadnu kodnu riječ adresnih varijabli, te aktivnim signalom propustiti signal na odabrani izlaz. Kodnu riječ algebarski prepoznamo algebarskim izrazom kojeg samo nazvali minterm, dakle konjunkcijom svih varijabli (sada adresnih). Minterm jedinicom prepozna pripadnu kodnu riječ pa možemo koristiti **I** vrata za propuštanje vrijednosti s ulaza na izabrani izlaz.

Algebarski se može pisati:

$$i_j = m_j(a) \cdot u$$

$$j = 0 \dots 2^m - 1$$

a negacijom izraza dobije se mogućnost korištenja NI vrata:

$$\overline{i_j} = \overline{m_j(a) \cdot u}$$

$$j = 0 \dots 2^m - 1$$

Ako je informacijski ulaz slobodno promjenjiv, a adresa stacionarna, odabrani izlaz će slijediti vrijednost sa ulaza. Obavili smo **razvođenje ulaza** na odabrani izlaz.

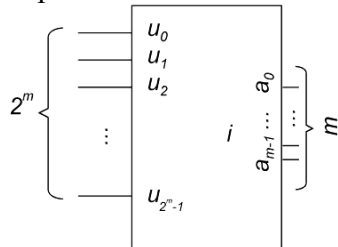
Ako se informacijski ulaz mijenja istovremeno (sinkrono) sa adresama, a adrese mijenjamo u prirodnom binarnom nizu, na izlazima će se pojaviti niz bita sa ulaza. Obavili smo **serijsko-paralelnu konverziju**.

Ako na ulaz trajno dovedemo 1, adresom biramo jedan od izlaza. Demultiplekser preuzima funkciju **dekodera**.

### 11.3. Enkoder s prioritetom

- definirati funkciju enkodera
- definirati funkciju enkodera s prioritetom
- opisati uporabu enkodera s prioritetom

**Enkoder** je uređaj koji se koristi za kodiranje signala ili podataka u oblik pogodan za prijenos ili pohranu.



Enkoder prioriteta na adresne izlaze  $a_{m-1}, a_{m-2}, \dots, a_1, a_0$  dovodi redni broj  $j$  u prirodnom binarnom obliku (kao kodnu riječ) onog informacijskog ulaza  $u_j$ , na kojem je prisutna jedinica.

Budući da nema jamstva da u nekom trenutku neće biti više informacijskih ulaza aktivirano istovremeno, sklop treba odlučiti kojem će od aktiviranih ulaza dati prednost (prioritet). Stoga se zove enkoder prioriteta.

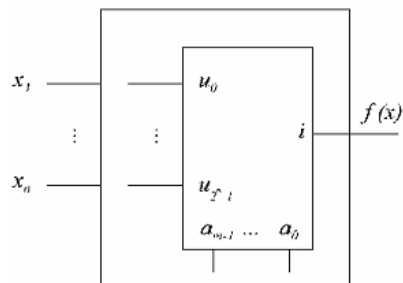
Enkoder prioriteta koristi se uvijek kada je potrebno neki kod tipa 1 od  $2^m$  prevesti u koncentrirani kod od  $m$  bita. Funkcija mu je suprotna funkciji dekodera. Njime možemo realizirati jednostavnu tastaturu. Masovno se primjenjuje u mikroprocesorima (za razlučivanje različitih prekidnih zahtjeva).

## 12. REALIZACIJA BF MULTIPLEKSEROM

### 12.1. Pristup realizaciji Booleove funkcije multiplekserom

- osnovni model realizacije
- osnovna jednadžba realizacije
- komentar veličine problema i mogućih rješenja

Osnovni model realizacije Booleove funkcije multiplekserom je:



Multiplexer ima jedan izlaz, pa samo na tom izlazu možemo dobiti vrijednost funkcije:

$$i = f(x_1, \dots, x_n)$$

Uvrštavanjem izraza za multiplekser i PDNO funkcije slijedi:

$$\bigvee_{j=0}^{2^m-1} m_j(a_{m-1}, \dots, a_0) \cdot u_j = \bigvee_{i=0}^{2^n-1} m_i(x_1, \dots, x_n) \cdot T_i$$

## 12.2. Realizacija BF multiplekserom za $n=m$

- jednačba realizacije za  $n=m$
- specijalno rješenje za  $n=m$
- konstrukcija sklopa za  $n=m$
- komentar rada na osnovi sheme multipleksera

Kada je broj varijabli jednak broju adresnih ulaza, tj.  $m=n$ , iz osnovne jednačbe realizacije

$$\text{dobijemo: } \bigvee_{j=0}^{2^m-1} m_j(a) \cdot u_j = \bigvee_{j=0}^{2^m-1} m_j(x) \cdot T_j$$

Lijeva i desna strana dobivene jednačbe su strukturno identične pa običnu jednakost možemo zamijeniti identitetom te izjednačiti po dijelovima. Dobijemo:

$$u_j = T_j; \quad j = 0 \dots 2^m - 1$$

$$m_j(a) = m_j(x); \quad j = 0 \dots 2^m - 1$$

Ako na adresne ulaze dovedemo redom varijable funkcije:

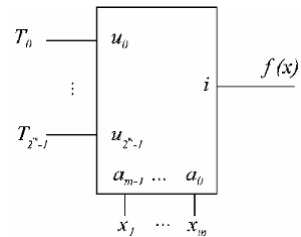
$$\begin{array}{ccccccc} a_{m-1} & a_{m-2} & \dots & a_1 & a_0 \\ \uparrow & \uparrow & & \uparrow & \uparrow \\ x_1 & x_2 & \dots & x_{m-1} & x_m \end{array}$$

Odnosno:

$$a_e = x_{m-e}; \quad e = m-1, \dots, 0$$

Booleovu funkciju  $m$  varijabli realiziramo multiplekserom s  $m$  adresnih ulaza tako da na adresne ulaze multipleksera dovedemo redom varijable funkcije (MSB na MSB, LSB na LSB), a na informacijske ulaze multipleksera redom vrijednost funkcije (0 ili 1). Redoslijed varijabli je važan da bi redoslijed ulaza multipleksera odgovarao redoslijedu redaka tablice istine, dakle redoslijedu vrijednosti funkcije.

Struktura realizacije Booleove funkcije multiplekserom za  $m=n$ :



## 12.3. Realizacija BF multiplekserom za $n>m$

- jednačba realizacije za  $n>m$
- algebarsko rješenje za  $n>m$
- konstrukcija sklopa za  $n>m$
- definirati potpuno multipleksero stablo

Za opći slučaj  $n>m$  gubi se strukturni identitet:

$$\bigvee_{j=0}^{2^m-1} m_j(a) u_j = \bigvee_{i=0}^{2^n-1} m_i(x) T_i$$

Da bismo postigli strukturni identitet, transformiramo desnu stranu. Zbog svojstva asocijativnosti konjunkcije, svaki minterm možemo rastaviti na:

$$m_i : (x_1 x_2 \dots x_m) \cdot (x_{m+1} \dots x_n)$$

To su zapravo mintermi prvih  $m$  i preostalih  $n-m$  varijabli:

$$m_i(x_1 \dots x_n) = m_j(x_1 \dots x_m) m_k(x_{m+1} \dots x_n)$$

U PDNO funkcije grupiramo minterm sa zajedničkim prvim dijelom, te korištenjem svojstva distributivnosti izlučimo zajednički dio:

$$f(x) = \bigvee_{j=0}^{2^m-1} m_j(x_1 \dots x_m) \left[ m_0(x_{m+1} \dots x_n) T_{j \cdot 2^{n-m} + 0} \vee \dots \vee m_{2^{n-m}-1}(x_{m+1} \dots x_n) T_{j \cdot 2^{n-m} + 2^{n-m}-1} \right]$$

Izraz u zagradi je PDNO preostale funkcije pa pišemo:

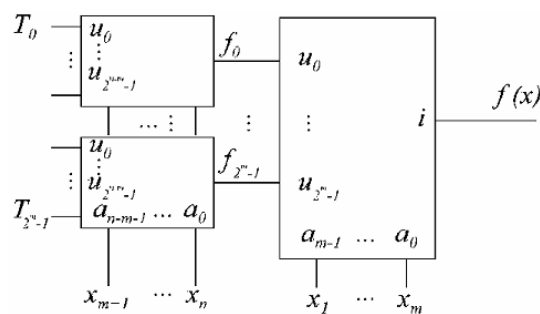
$$\bigvee_{j=0}^{2^m-1} m_j(a) u_j = \bigvee_{j=0}^{2^m-1} m_j(x_1 \dots x_m) \cdot f_j(x_{m+1} \dots x_n) = \bigvee_{j=0}^{2^m-1} m_j(x) f_j(x)$$

Opet je uspostavljen strukturni identitet budući da su sve preostale funkcije funkcije istih varijabli pa vrijedi:

$$\begin{aligned} u_j &= f_j(x_{m+1} \dots x_n) \\ m_j(a) &= m_j(x_1 \dots x_m) \\ j &= 0 \dots 2^m - 1 \\ a_e &= x_{m-e} \\ e &= m-1, \dots, 0 \end{aligned}$$

Booleovu funkciju **n** varijabli realiziramo multiplekserom s **m** adresnih ulaza tako da na adresne ulaze multipleksera dovedemo **m** varijabli funkcije (to su adresne varijable), a na informacijske ulaze multipleksera preostale funkcije preostalih **n-m** varijabli funkcije (to su preostale varijable). Redoslijed adresnih varijabli određuje redoslijed preostalih funkcija.

Preostale funkcije treba realizirati posebnim sklopovljem koji mogu biti logička vrata ali i multiplekseri. Ako smo ih realizirali multiplekserima, dobili smo strukturu koju nazivamo **multiplekstersko stablo**. Potpuno stablo ekvivalentno je jednom multiplekseru.



## 12.4. Minimizacija multipleksterskog stabla

- definirati mogućnost minimizacije multipleksterskog stabla
- kriterij minimizacije multipleksterskog stabla
- specijalni slučaj optimalnog sklopa s multiplekserom
- metodologija minimizacije multipleksterskog stabla

Minimizacija multipleksterskog stabla je moguća tako da eliminiramo čitavu granu stabla. Pojedinu granu možemo eliminirati ako pripadnu preostalu funkciju možemo realizirati bez sklopovlja, a takve funkcije su funkcije jedne varijable: konstante 0 i 1, jednakost i negacija. Za realizaciju preostalih funkcija multiplekserima adresne se varijable biraju tako da što veći broj preostalih funkcija bude funkcija jedne varijable.

Za poseban slučaj kada je **n=m+1**, na adresne ulaze multipleksera dovodimo **m** varijabli funkcije, a preostale funkcije su sve sigurno funkcije jedne preostale varijable. Stoga je optimalno multiplekserom s **m** adresnih ulaza realizirati funkciju s **n=m+1** varijablom.

Iz algebarskog oblika i sheme multipleksera vidi se da on neposredno realizira PDNO funkcije (ili u osnovnom obliku ili nakon razbijanja na preostale funkcije).

U realizaciji Booleovih funkcija multiplekserom za izračunavanje preostalih funkcija koristimo Veitcheve dijagrame. Izborom svake adresne varijable Veitchev dijagram se raspada na dva dijela koji predstavljaju Veitcheve dijagrame preostalih funkcija.

## 13. REALIZACIJA BF DEMULTIPLEKSEROM

### 13.1. Pristup realizaciji Boolove funkcije demultiplekserom

- osnovni model realizacije
- osnovna jednačba realizacije
- komentar veličine problema i mogućih rješenja

Demultiplekser ima  $2^m$  (informacijskih) izlaza, od kojih u sklopu dekodera (ako je informacijski ulaz  $u = 1$ ) svaki (izlaz) ostvaruje po jedan minterm:

$$u = 1$$

$$i_j = m_j(a) * u = m_j(a) * 1 = m_j(a)$$

pri čemu je  $j = 0 \dots 2^m - 1$

Zatim je dovoljno povezati potrebne izlaze (one za koje je vrijednost funkcije jednaka 1) demultipleksera na ILI logička vrata i tako neposredno realizirati PDNO Boolove funkcije.

**Mana:** realizira PDNO, nema minimizacije

**Prednost:** realizira više funkcija istih varijabli

### 13.2. Realizacija BF demultiplekserom za $n=m$

- jednačba realizacije za  $n=m$
- specijalno rješenje za  $n=m$
- konstrukcija sklopa za  $n=m$
- problem konstrukcije ILI vrata
- komentar rada na osnovi sheme demuxa

Za  $n=m$  vrijedi:

$$f(x) = \bigvee_{j=0}^{2^m-1} m_j(x) T_j \quad \Rightarrow \quad i_j = m_j(a)$$

Da bismo minterme funkcije mogli realizirati mintermima adresnih varijabli,

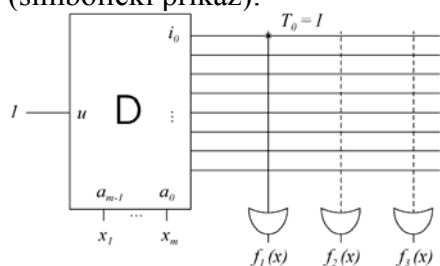
na adresne ulaze demux-a:  $a_{m-1}, a_{m-2}, \dots, a_1, a_0$

dovedemo varijable funkcije:  $x_1, x_2, \dots, x_{m-1}, x_m$  točno ovim redom

pa se dobije:

$$\left. \begin{array}{l} a_e = x_{m-e} \\ e = 0 \dots m-1 \end{array} \right\} m_j(a) = m_j(x) = i_j \quad \Rightarrow \quad f(x) = \bigvee_{j=0}^{2^m-1} i_j T_j$$

Na ILI vrata spojimo samo one izlaze, za koje je vrijednost funkcije jednaka jedinici (simbolički prikaz):



**Mana:** realizira PDNO, nema minimizacije

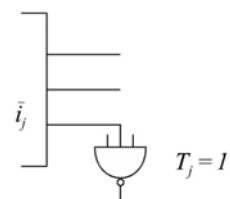
**Prednost:** realizira više funkcija istih varijabli

Javlja se problem konstrukcije ILI vrata. Tomu doskočimo ovako:

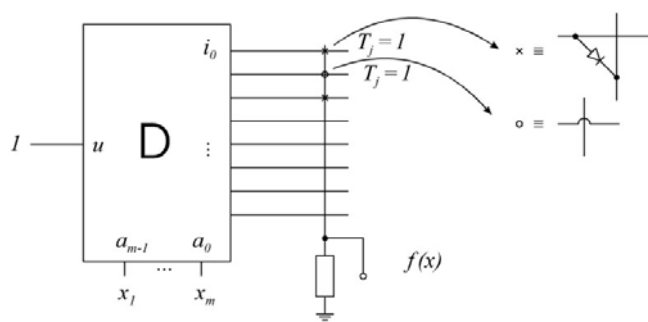
Dvostruko negiramo izraz (i primijenimo DeMorgana) :

$$f(x) = \bigvee_{j=0}^{2^m-1} i_j T_j = \overline{\bigwedge_{j=0}^{2^m-1} \overline{i_j T_j}} = \overline{\bigwedge_{j=0}^{2^m-1} \overline{i_j} \overline{T_j}}$$

pa sada vidimo da možemo koristiti demux s negiranim izlazima i NI vrata:



Umjesto ILI odnosno NI vrata koristimo diodnu logiku:



pa jednostavnim dodavanjem dioda lagano realiziramo ILI vrata s potrebnim brojem ulaza.  
Za više funkcija istih varijabli, dobijemo polje dioda ("diodna matrica")

### 13.3. Realizacija BF demultiplekserom za $n > m$

- jednažba realizacije za  $n > m$
- algebarsko rješenje za  $n > m$
- konstrukcija sklopa za  $n > m$
- definirati potpuno demultiplekstersko stablo

Za  $n > m$  pokušamo transformirati PDNO funkcije:

$$f(x) = \bigvee_{i=0}^{2^n-1} m_i(x) T_i = \bigvee_{j=0}^{2^m-1} m_j(x_1, \dots, x_m) f_j(x_{m+1}, \dots, x_n)$$

$$\Rightarrow f(x) = \bigvee_{i=0}^{2^m-1} i_j f_j(x)$$

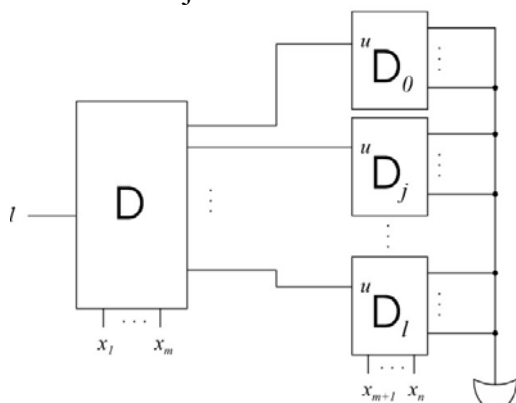
$$\Rightarrow f(x) = \bigvee_{i=0}^{2^m-1} i_j \bigvee_{k=0}^{2^{n-m}-1} m_k(x) T_{2^{n-m} \cdot j + k}$$

Rješenje rezultira nezgrapnim sklopom. Zato probamo drugačije:

Ako preostalu funkciju realiziramo demultiplekserom možemo koristiti ulaz, koji je konjunktivno vezan sa izlazima, a ranije smo ga isključili:

$$f(x) = \bigvee_{i=0}^{2^m-1} i_j \bigvee_{k=0}^{2^{n-m}-1} u \cdot i_k(x) T_{2^{n-m} \cdot j + k}$$

na način da izlaz glavnog demultipleksera dovedemo na ulaz demultipleksera preostale funkcije. Dobili smo DEMULTIPLEKSTERSKO STABLO! Kada je POTPUNO, stablo je ekvivalentno jednom većem demux-u. Sklop sada ima strukturu:



### 13.4. Minimizacija demultipleksterskog stabla

- definirati mogućnost minimizacije demux stabla
- kriterij minimizacije demux stabla
- metodologija minimizacije demux stabla
- specijalni slučaj optimalnog demux stabla

Struktura stabla nam omogućava minimizaciju demultipleksterskog stabla eliminacijom pojedine grane stabla. To je moguće potpunom eliminacijom preostale funkcije, tj. kad je preostala funkcija jednaka **KONSTANTI 1 ili 0**.

Veličina preostalih funkcija (pa i cijelog sklopovlja) ovisi o izboru adresnih varijabli demux-a.

#### Specijalan slučaj je za $n=m+1$

Tada su sve preostale funkcije = funkcije jedne varijable.

Na adresne ulaze dvaju demultipleksera dovodimo  $m$  varijabli funkcije, a na njihove informacijske ulaze dovodimo izvornu, odnosno negiranu, vrijednost preostale varijable.

## 14. MULTIPLEKSKO-DEMULTIPLEKSKA (MD) STRUKTURA

### 14.1. Multiplekstersko-demultipleksterska struktura

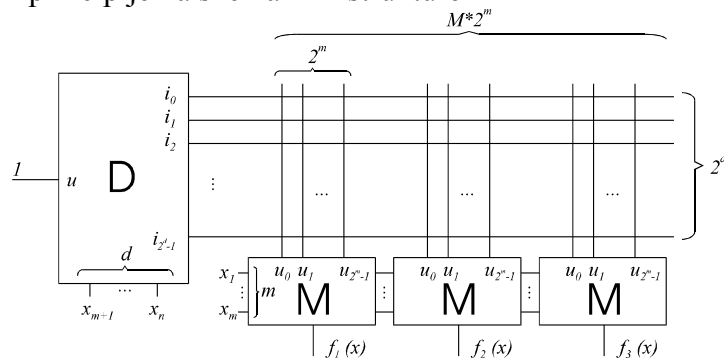
- motivacija, koncept
- principijelna shema MD strukture
- broj redaka i stupaca matrice

Motivacija za uvođenje MD strukture bila je želja da realiziramo tehnologije visoke i vrlo visoke razine integracije.

Koncept: MD struktura služi za realizaciju Boolovih funkcija istih varijabli.

Broj adresnih ulaza u strukturu je jednak zbroju adresnih ulaza u multiplekser  $m$  i zbroju adresnih ulaza u demultiplekser  $d$ .

- principijelna shema MD strukture



- broj redaka i stupaca matrice

Broj redaka  $R$  matrice jednak je broju izlaza demultipleksera. Broj stupaca  $S$  matrice jednak je broju multipleksera pomnoženom s brojem ulaza u pojedini multiplekser.



## 14.2. Optimalna veličina MD strukture

- koncept obodnih vrata
- optimalni broj logičkih vrata
- određivanje optimalne MD strukture za n varijabli i M funkcija

Zbog potrebe da broj logičkih vrata strukture bude minimalan, tražimo da broj redaka R i stupaca S diodne matrice bude minimalan, a minimalnost postignemo približno jednakim brojem stupaca i redaka. Broj logičkih vrata  $L=R+S$

Za f-ju od n varijabli broj članova matrice je uvijek  $2^n$ , jer za svaku od  $2^n$  kodnih riječi varijabli x upisujemo vrijednost f-je 0 ili 1. Broj logičkih vrata jednak je zbroju redaka i stupaca matrice i može se pokazati da je minimalan kada je  $R=S$ .

## 14.3. Memorije sa samom očitanjem

- MD struktura kao ROM
- kompatibilnost s radom računala
- tehnološke osobine ROM komponenti
- vremenski dijagram čitanja podatka

### MD struktura kao ROM:

Rom je MD struktura sa programibilnom ILI i fiksnom I matricom. Mana je nemogućnost minimizacije minterma funkcije u PDNO tj. nemogućnost korištenja MDNO.

### Kompatibilnost s radom računala:

Imaju podatke trajno pohranjene u svoju strukturu, ne gube ih isključenjem napona napajanja. Sadržaj se upisuje u trenutku izrade.

### Tehnološke osobine ROM komponenti:

ROM – sadržaj upisan kod izrade i ne može se kasnije mijenjati

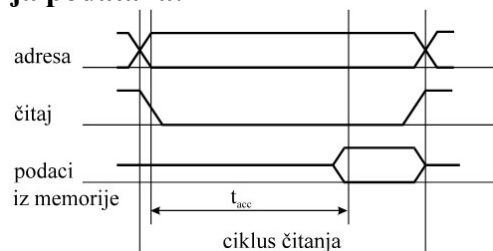
PROM – sadržaj se mijenja pregaranjem osigurača

EPROM – koristi tehnologiju lebdećih vrata, briše se UV svjetlom

EEPROM – lebdeća vrata, briše se elektronički byte po byte

FEPRM – lebdeća vrata, brisanje električko blok po blok

### Vremenski dijagram čitanja podatka:



**Vrijeme čitanja podatka** – vrijeme koje protekne od trenutka postavljanja adrese i signala čitanja do trenutka kada na izlazu dobijemo očitane podatke.

## 15. PROGRAMABILNE LOGICKE STRUKTURE

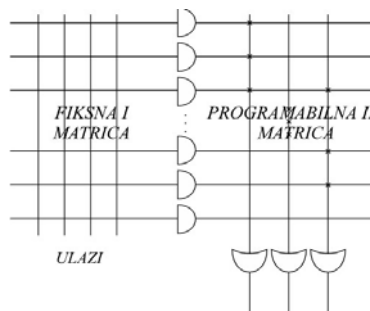
### 15.1. Definicija programibilne logičke strukture

- koncept I i ILI matrice
- vrste PLS ovisno o programibilnosti matrice
- ROM struktura

Struktura memorije se može prikazati pomoću I matrice i ILI matrice. Demultiplekser i multiplekser iz MD strukture prikazujemo sad simbolički ILI vratima i I vratima.

Vrste PLS ovisno o programibilnosti matrice

- **FPLA** - programibilna I i ILI matrica
- **GAL, PAL** - programibilna I i fiksna ILI matrica



**ROM** je memorijska struktura kod koje je sadržaj upisan kod izrade. Struktura ROM memorije nije optimalna.

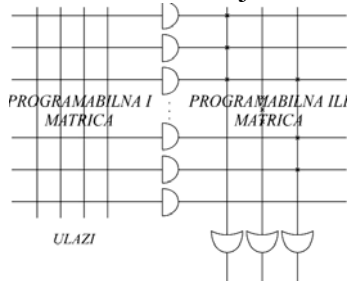
Programibilna ILI matrica samo učinkovito povezuje minterme funkcije u PDNO.

Mintermi su realizirani fiksnom I matricom, što onemogućuje minimizaciju i korištenje MDNO.

### 15.2. FPLA (Field Programmable Logic Array)

- FPLA struktura
- prednosti i mane FPLA strukture

**FPLA struktura** je struktura s programabilnim I i ILI matricama.



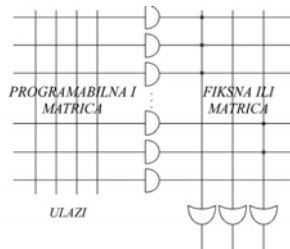
**Prednosti:** omogućena minimizacija pojedine funkcije i izbor elementarnih članova. Izračunamo MDNO funkcije, el. članove programiramo u I matricu, te ih povežemo u konačni oblik ILI matricom.

**Mane:** dvije programibilne matrice zahtijevaju dvostruko više pomoćnog sklopovlja, skuplji integrirani krugovi, troši se previše energije. Programibilna ILI matrica se programira: prvih nekoliko el. članova za prvu funkciju, slijedećih nekoliko za drugu itd.

### 15.3. GAL (Generic Array Logic)

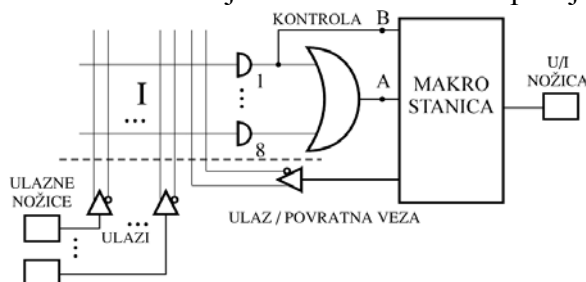
- PAL struktura
- GAL struktura
- koncept makro ćelije

**PAL struktura** je struktura s fiksnom ILI matricom i programibilnom I matricom.



**PAL** su komponente s PROM matricom u TTL tehnologiji. Potreba za različitim odnosom veličine funkcije (broj element. članova) i broja funkcija (broj izlaza) za istu veličinu matrice dovodi do pojave čitave familije različitih komponenti.

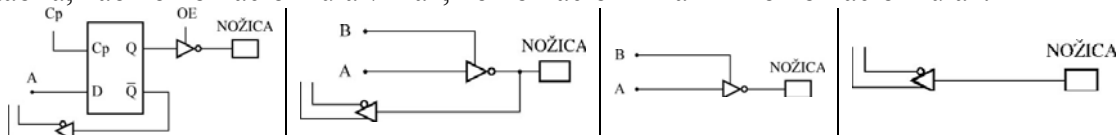
**GAL struktura** je struktura ostvarena primjenom CMOS tehnologije i EEPROM



upravljačkih bitova. Kod GAL komponenti moguće je brisanje i ponovo upisivanje I matrice.

Svega dvije komponente zamjenjuju ranije PAL-ove, te donose nove mogućnosti kroz programibilne izlazne sklopove. Fleksibilnost GAL-a proizlazi iz koncepta makro ćelija i povratnih veza.

**Makro ćelije**(stanice) su programibilni izlazni sklopovi korištene kod GAL strukture. Svaka makro stanica može biti konfigurirana na četiri načina kao: sekvencijalni izlaz preko D bistabila, kao kombinacioni ulaz/izlaz, kombinacioni izlaz ili kombinacioni ulaz.



### 15.4. CPLD (Complex Programmable Logic Device)

- struktura CPLD
- jezici za definiranje sklopovlja (HDL)

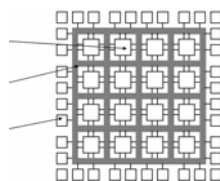
CPLD ili kompleksne programibilne logičke strukture su integrirani krugovi ili čipovi koji se koriste za implementiranje digitalnog hardvera.

**Struktura CPLD:**

CPLD sadrži više logičkih blokova, od kojih svaki uključuje 8 do 16 makro-ćelija. Budući da svaki logički blok izvršava određenu funkciju, sve makro-ćelije unutar logičkog bloka su potpuno spojene. Međutim, ovisno o primjeni, logički blokovi mogu ili ne moraju biti spojeni jedan s drugim.

Struktura CPLD

Logički blokovi  
Spojne veze  
U/I blokovi



Za definiranje sklopovlja koristimo VHDL.

## **16. SEKVENCIJALNI SKLOPOVI**

### **16.1. Kombinajski sklopovi**

- definirati kombinajski sklop
- obrazložiti potrebe za pamćenjem kombinajskog sklopa

Kombinajski sklopovi su sklopovi koji izlaz generiraju isključivo na osnovi trenutnih vrijednosti ulaza. Nemaju memoriju jer ne trebaju pamtit prethodne događaje tj. ne pamte stanja (stateless), iako stvarni sklopovi imaju neko kašnjenje (nepoželjno zadržavanje informacije) uzrokovano kašnjenjem na pojedinim logičkim vratima.

### **16.2. Sekvencijalni sklopovi**

- definirati sekvencijalni sklopovi
- obrazložiti potrebe za pamćenjem sekvencijalnog sklopa

Za razliku od kombinajskih, sekvencijalni sklopovi rade u vremenu tako da njihov izlaz ovisi o trenutnim i o prošlim vrijednostima ulaza. Stoga sekvencijalni sklopovi raspolažu memorijom kako bi pamtili prethodne događaje.

Napomena : Ako dva niza događaja (ulaznih sekvenci) dovedu sklop u isto stanje, sklop ih (u svom budućem radu) više ne može razlikovati.

### **16.3. Kašnjenje i pamćenje**

- definirati kašnjenje
- definirati pamćenje
- komentirati tehničko ostvarenje pamćenja

Kašnjenje je nepoželjno (štetno) zadržavanje informacije u vremenu (doduše jako kratko)

Pamćenje je također zadržavanje informacije u vremenu, ali ovaj put poželjno.

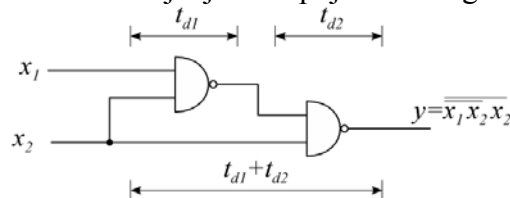
Mehanizmi koji dovode do kašnjenja mogu biti upotrijebljeni za realizaciju pamćenja, a osim toga, za pamćenje se mogu iskoristiti druge modifikacije energije (npr. EPROM) i modifikacije materije (npr. ROM)

## 17. RAD SKLOPA U DISKRETNOM VREMENU

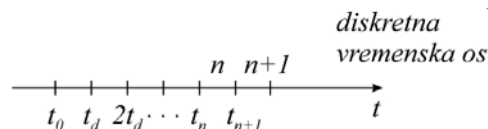
### 17.1. Diskretno vrijeme

- utjecaj kašnjenja na logičkim vratima
- definicija diskretnog vremena
- komentar diskretnog vremena obzirom na informacijski volumen
- teorem uzorkovanja

**Kašnjenje** je nepoželjno zadržavanje informacije u vremenu, odnosno ulazi u sklop djeluju na izlaz u trenucima koji su određeni kašnjenjem na pojedinim logičkim vratima.



Promjene se događaju u **diskretnim** trenucima  $t$  dok u periodima između trenutaka nema promjena.  $t_n$  = sadašnji trenutak,  $t_{n+1}$  slijedeći trenutak,  $n$  = sadašnji period,  $n+1$  slijedeći period.

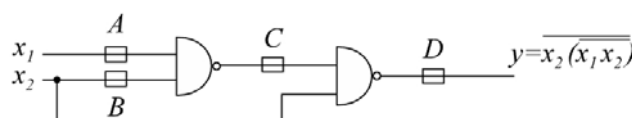


Diskretno vrijeme definira da moramo pročitati 2B signalnih elemenata u sekundi da bi pročitali informaciju.

### 17.2. Rad sklopa u diskretnom vremenu

- stabilno i nestabilno stanje sklopa
- prikaz ponašanja sklopa

Mjerimo signale u točkama A, B, C i D i u trenutku  $t_n$  kodna riječ od ABCD opisuje trenutno stanje sklopa. U nizu trenutaka sklop prolazi kroz niz stanja. Neka stanja su **stabilna** (iz njih izlazimo samo na vanjski poticaj), a neka su **nestabilna** (iz njih izlazimo samo na osnovi protoka vremena).



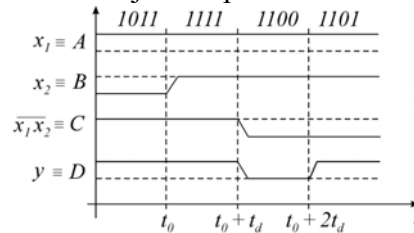
Stabilno stanje 1011, promjena na ulazu iz 10 u 11 uzrokuje nestabilno stanje 1111. Nakon  $t_d$  dolazi stanje 1100, a nakon  $2t_d$  novo stabilno stanje 1101.

$x_1$	$x_2$	A	B	C	D	
1	0	1	0	1	1	
1	1	1	1	1	1	$t_0$
1	1	1	1	0	0	$t_0 + t_d$
1	1	1	1	0	1	$t_0 + 2t_d$

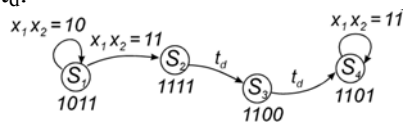
### 17.3. Sinkroni sklopovi

- proizvoljni period diskretnog vremena i sklopovi za pamćenje
- nestabilni i stabilni period rada sklopa
- vremenski dijagram rada sinkronog sklopa

Iz **vremenskog dijagrama** vidimo neželjeni impuls na izlazu.



Čvorovi usmjerenog grafa (krugovi) predstavljaju stanja. Usmjerene duljine predstavljaju prijelaze. Sklop ostaje u stabilnom stanju sve dok se ulaz ne promijeni, a u nestabilnim stanjima dok ne istekne period  $t_d$ .



Informaciju želimo pamtit **proizvoljno vrijeme**, odnosno ne želimo ovisiti o kašnjenju na logičkim vratima, pa trajanje diskretnog perioda povećamo na željenu vrijednost. Da bi se postiglo proizvoljno diskretno vrijeme, potreban je sklop koji može pamtit informaciju potreban period vremena.

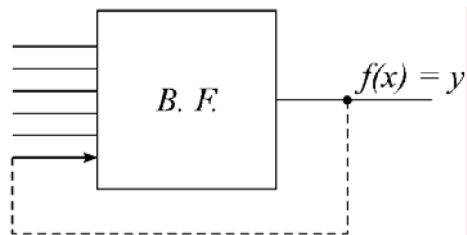
## 18. BISTABIL KAO SKLOP

### 18.1. Osnovni sklop za pamćenje - elementarni RS bistabil

- definirati funkciju bistabila
- povratna veza i njen značaj
- elementarni RS bistabil

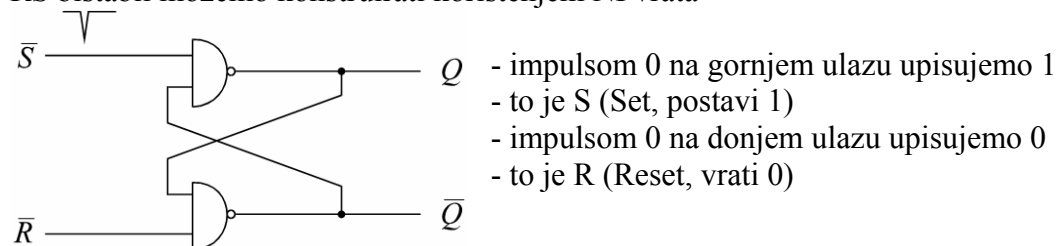
Bistabil je osnovni memorijski element za pamćenje vrijednosti Boole-ove varijable i ima dva stabilna unutrašnja stanja (0 ili 1). Promjena stanja memorijskog elementa ovisi o trenutnoj vrijednosti na njegovim ulazima  $q_1, q_2, \dots, q_n$ , kao i o njegovom unutrašnjem stanju.

Unutrašnje stanje očitavamo na izlazima  $Q$  i  $\bar{Q}$ .



Bistabil je simetričan sklop i kod realizacije TTL tehnologijom sastoji se od dva tranzistora, koji sačinjavaju dvostruko pojačalo s jakom pozitivnom povratnom vezom. Veliko pojačanje u petlji povratne veze ne dozvoljava da sklop oscilira, nego dovodi do toga da jedan tranzistor vodi (u zasićenju) dok drugi ne vodi.

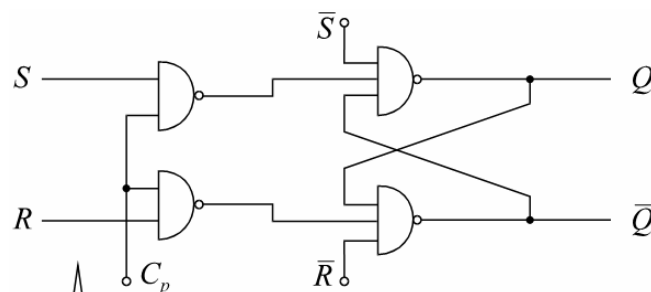
RS bistabil možemo konstruirati korištenjem NI vrata



### 18.2. Sinkronizacija bistabila s diskretnim vremenom

- informacija i trenutak prijelaza
- RS bistabil sinkroniziran impulsom
- sinkroni i asinkroni RS ulazi

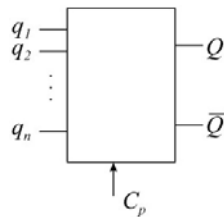
Trenuci u kojima djeluju impulsi su trenuci promjene u diskretnom vremenu. Diskretno vrijeme možemo definirati nekim tankim signalom, u obliku niza impulsa. Kontroliramo trenutak djelovanja ulaznih impulsa taktim signalom.



Ulazi R i S djeluju na sklop samo u trenutku pozitivnog taktog impulsa. To su sinkroni ulazi. S osnovnog bistabila izvedeni su dodatni asinkroni ulazi,  $\bar{R}$  i  $\bar{S}$ . Ovi ulazi služe prilikom uključanja napajanja za postavljanje bistabila u poznato početno stanje.

### 18.3. Bistabil kao funkcionalni blok

- model bistabila
- definirati tablice prijelaza
- definirati funkciju prijelaza



Bistabil možemo prikazati kao funkcionalni blok koji se sastoji od ulaz  $q_1, q_2, \dots, q_n$  i izlaza  $Q$  i  $\bar{Q}$ . Također ima ulaz  $C_p$  zbog kojeg u trenutku nastupa taktnog signala, bistabil mijenja stanje na osnovu vlastitog trenutnog stanja i vrijednosti na ulazima.

$(q_1 \ q_2 \ \dots \ q_n \ Q)^n$	$Q^{n+1}$	$Q^{n+1}$
0 0 ... 0 0	0	$Q^n$
0 0 ... 0 1	1	$\bar{Q}^n$
...	...	...
1 1 ... 1 0	1	0
1 1 ... 1 1	1	1

Bistabil se može zapisati tablicom prijelaza, potpunom i skraćenom. Tablica prijelaza ima vremenski odnos, pa se vrijednosti s lijeve strane odnose na sadašnji trenutak, a vrijednosti s desne strane tablice na sljedeći trenutak. Oznaka  $Q^n$  ovdje znači stanje bistabila u sadašnjem trenutku, a oznaka  $Q^{n+1}$  znači stanje bistabila u sljedećem trenutku.

$(q_1 \ q_2 \ \dots \ q_n)^n$	$Q^{n+1}$
0 0 ... 0 0	$Q^n$
0 0 ... 0 1	$\bar{Q}^n$
...	...
1 1 ... 1 0	0
1 1 ... 1 1	1

Skraćena tablica prijelaza s lijeve strane ima kodne riječi ulaza  $q^n$  poredane prirodnim binarnim nizom, sve u sadašnjem,  $n$ -tom trenutku, a s desne strane stanje  $Q^{n+1}$ , stanje u sljedećem trenutku, kao funkciju sadašnjeg stanja  $Q^n$ . Situacija je slična razbijanju na parcijalne funkcije

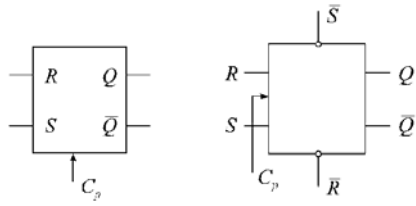
Bistabil možemo zapisati i funkcijom prijelaza:  $Q^{n+1} = f(Q, q_1, \dots, q_m)^n = (G_1 Q \vee G_2 \bar{Q})^n$ , gdje  $G_1$  opisuje rad bistabila kad je sadašnje stanje 1, a  $G_2$  rad bistabila kad je sadašnje stanje 0.

Sljedeće stanje je funkcija sadašnjeg stanja i ulaza. To je algebarski oblik sličan Booleovoj funkciji, ali za razliku od nje ima vremenski odnos između lijeve i desne strane tako da se s obje strane može pojaviti ista varijabla.



## 18.4. Standardni bistabili

- RS bistabil: definicija, tablice, funkcije
- JK bistabil: definicija, tablice, funkcije
- T bistabil: definicija, tablice, funkcije
- D bistabil: definicija, tablice, funkcije



**RS bistabil** je osnovni bistabil, jer se nalazi u osnovi realizacije svih drugih bistabila. Varijante RS bistabila, s asinkronim ulazima i bez njih, prikazane su na slici.

RS bistabil je zadan skraćenom i potpunom tablicom prijelaza.

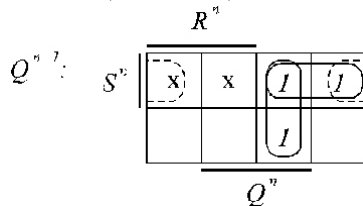
$(R \ S)^n$	$Q^{n+1}$	$(R \ S \ Q)^n$	$Q^{n+1}$
0 0	$Q^n$	0 0 0	0
0 1	1	0 0 1	1
1 0	0	0 1 0	1
1 1	X	0 1 1	1
		1 0 0	0
		1 0 1	0
		1 1 0	X
		1 1 1	X

X – neodređeno sljedeće stanje (prijelaz)

Funkcija prijelaza:

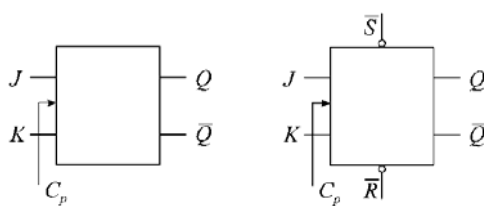
$$Q^{n+1} = (\bar{R} Q \vee S \bar{Q})^n \quad G_1 = \bar{R} \quad G_2 = S \quad Q^{n+1} = (\bar{R} Q \vee \bar{R} S)^n$$

$$Q^{n+1} = (\bar{R} (Q \vee S))^n \quad Q^{n+1} = (\bar{R} Q \vee S)^n$$



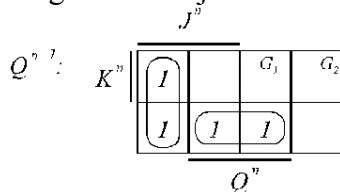
Uvjet  $RS=0$  osiguravamo izvana

**JK bistabil** je univerzalni bistabil koji je zadan skraćenom i potpunom tablicom prijelaza a simbol mu je:



$(J \ K)^n$	$Q^{n+1}$	$(J \ K \ Q)^n$	$Q^{n+1}$
0 0	$Q^n$	0 0 0	0
0 1	0	0 0 1	1
1 0	1	0 1 0	0
1 1	$\bar{Q}^n$	0 1 1	0
		1 0 0	1
		1 0 1	1
		1 1 0	1
		1 1 1	0

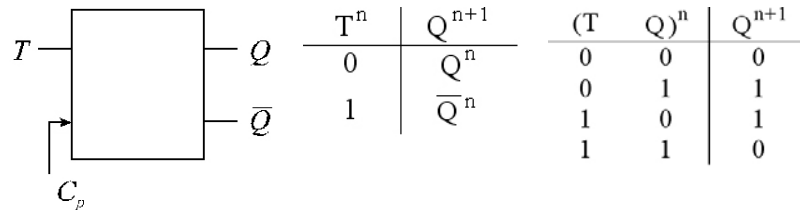
Za JK bistabil kaže se da je upravljiv s ulaza. Jer u skraćenoj tablici prijelaza ima sve četiri moguće funkcije ovisnosti o prethodnom stanju.



$$Q^{n+1} = (\bar{K} Q \vee J \bar{Q})^n$$

$$G_1 = \bar{K} \quad G_2 = J$$

**T bistabil** mijenja ili zadržava stanje. Zadan je skraćenom i potpunom tablicom a simbol mu je:

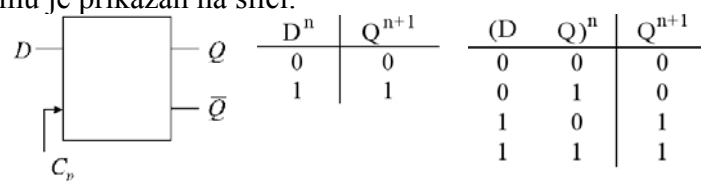


Za T bistabil kaže se da je upravljiv taktom, jer na osnovi ulaza T može samo zadržati ili mijenjati sadašnje stanje. Funkcija prijelaza:

$$Q^{n+1}: \quad Q \begin{array}{|c|c|} \hline \overline{G_1} & 1 \\ \hline G_2 & 1 \\ \hline \end{array} \quad Q^{n+1} = (\overline{T}Q \vee T\overline{Q})^n \quad G_1 = \overline{T} \quad G_2 = T \quad G_1 = \overline{G_2}$$

$G_1$  i  $G_2$  su komplementarni, što ograničava mogućnosti rada s funkcijom prijelaza.

**D bistabil** pamti 0 ili 1 neposredno s ulaza. Zadan je skraćenom i potpunom tablicom prijelaza a simbol mu je prikazan na slici.



Za D bistabil kažemo da pamti podatak te da realizira kašnjenje, jer na osnovi ulaza D neposredno pamti 0 ili 1 s ulaza. Funkcija prijelaza:

$$Q^{n+1}: \quad Q \begin{array}{|c|c|} \hline \overline{G_1} & 1 \\ \hline G_2 & 1 \\ \hline \end{array} \quad Q^{n+1} = (DQ \vee D\overline{Q})^n \quad Q^{n+1} = D^n$$

$$G_1 = D \quad G_2 = D$$

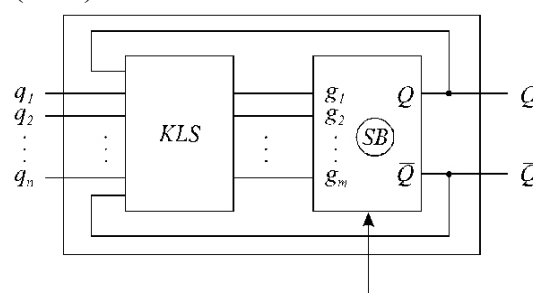
$$G_1 = G_2$$

## 19. SINTEZA OPĆIH BISTABILA

### 19.1. Model realizacije općih bistabila

- blok shema modela realizacije općeg bistabila
- značajke modela realizacije općeg bistabila
- osnovna formula realizacije općeg bistabila
- navesti metode poimence

**Opći bistabil** realiziramo korištenjem standardnog bistabila i kombinacijske logičke strukture (KLS)



Svojstvo je modela da su izlazi standardnog ujedno izlazi općeg bistabila. Stoga standardni bistabil treba raditi one prijelaze koji su tablicom prijelaza zadani za opći bistabil. KLS transformira ulaze u opći bistabil, na osnovi stanja bistabila, u signale potrebne da bi standardni bistabil obavio potrebne prijelaze.

$$Q_{OB}^n = Q_{SB}^n \quad Q_{OB}^{n+1} = Q_{SB}^{n+1}$$

Nakon izbora standardnog bistabila sinteza općeg bistabila se svodi na sintezu njegove KLS. U tu se svrhu koriste sljedeća tri postupka: metoda rekonstrukcije, metoda izjednačavanja, metoda za D bistabil.

## 19.2. Metoda rekonstrukcije

- objasniti metodu rekonstrukcije
- tablica rekonstrukcije za standardne bistabile
- primjena metode rekonstrukcije

Postupak rekonstrukcije pogodan je za sve bistabile, a jedini je upotrebljiv za RS (zbog uvjeta  $RS=0$ ) i T (zbog komplementarnih  $G_1$  i  $G_2$ ) bistabile

$(q_1 \quad q_2 \dots q_n \quad Q)^n$	$Q^{n+1}$	$(g_1 \quad g_2 \dots g_m)^n$
0 → 0	0	...
0 → 1	1	...
1 → 0	0	...
1 → 1	1	...

- potpunu tablicu prijelaza nadopunimo potrebnim ulazima u standardni bistabil da bi radio iste prijelaze
- lijeva i dodana desna strana čine TABLICU ISTINE za KLS
- poznatim postupcima minimizacije i realizacije Booleovih funkcija obavimo sintezu KLS, čime smo završili sintezu općeg bistabila

Rekonstrukciju vršimo prema slici:

$Q^n \rightarrow Q^{n+1}$	R	S	J	K	T	D
0 → 0	r	0	0	r	0	0
0 → 1	0	1	1	r	1	1
1 → 0	1	0	r	1	1	0
1 → 1	0	r	r	0	0	1

Tablicu rekonstruiranih vrijednosti lako popunimo prema potpunim tablicama prijelaza standardnih bistabila. Oznaka r predstavlja redundantnu (neodređenu) vrijednost, a koristi se malo slovo da bi se razlikovalo od R ulaza RS bistabila.

Metoda rekonstrukcije je pogodna za sve standardne bistabile, a naročito za RS i T bistabil.

## 19.3. Metoda izjednačavanja

- objasniti metodu izjednačavanja
- specifičnost metode izjednačavanja za NI i NILI vrata
- primjena metode rekonstrukcije

Postupak izjednačavanja pogodan je za JK bistabil. Ovaj postupak se provodi tako da izjednačimo funkcije prijelaza općeg (aplikativna jednadžba) i standardnog bistabila (karakteristična jednadžba):

$$Q_{SB}^{n+1} = Q_{OB}^{n+1} \quad (H_1 Q \vee H_2 \bar{Q})^n = (G_1 Q \vee G_2 \bar{Q})^n$$

Na osnovi strukturne sličnosti slijedi:

$$H_1 = G_1 \quad H_2 = G_2$$

Za JK bistabil vrijedi:

$$\bar{K} Q \vee J \bar{Q} = G_1 Q \vee G_2 \bar{Q} = f(q_1, q_2, \dots, q_m)^n$$

Odakle slijedi:

$$K = \bar{G}_1 \quad J = G_2$$

U Veitchev dijagram dovoljno upisati vrijednosti  $Q^{n+1}$  općeg bistabila, te minimizirati funkcije  $\bar{G}_1$  i  $G_2$ .

## 19.4. Metoda za D bistabil

- objasniti metodu za D bistabil
- specifičnost metode za D bistabil kod MD struktura
- primjena metode za D bistabil

Postupak za D bistabil je istovremeno postupak rekonstrukcije i postupak izjednačavanja. Temelji se na funkciji prijelaza  $Q^{n+1} = D^n$ , tako da je potpuna tablica općeg bistabila numerički identična tablici istine za KLS. Postupak se provodi tako da prema potrebi minimiziramo funkciju za  $D^n$ , ili negiranu funkciju  $\overline{D}^n$ .

## 20. SLOŽENI SKLOPOVI S BISTABILIMA

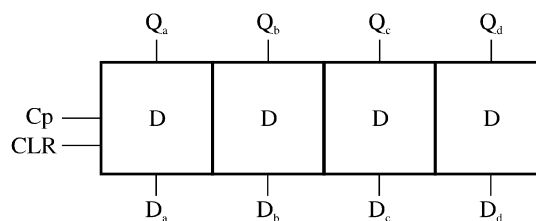
### 20.1. Registar

- definirati registar kao sklop
- principijelna shema registra
- primjena registra

#### Registar kao sklop:

Sklop koji se sastoji od više D bistabila, koji svi imaju zajednički takti ulaz Cp. Izvode se od 4 i 8 bistabila i najčešće raspolažu asinkronim CLR ulazom kojim se dovode u početno stanje 0.

#### Principijelna shema registra:



**Primjena registra:** Služi za pamćenje cjelovitih kodnih riječi.

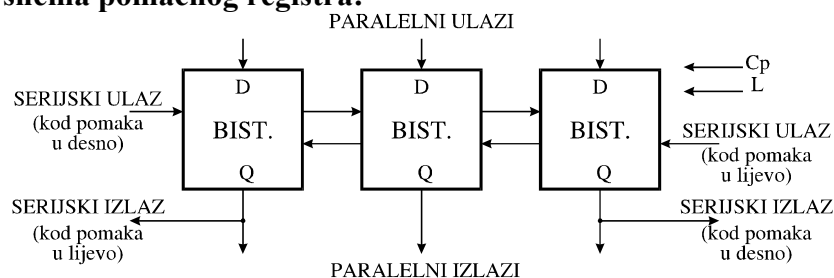
### 20.2. Pomačni registar

- definirati pomačni registar kao sklop
- principijelna shema pomačnog registra
- primjena pomačnog registra

#### Pomačni registar kao sklop:

Sklop koji se sastoji od više D bistabila, koji imaju zajednički takti ulaz, a spojeni su tako da je izlaz jednog doveden na ulaz drugog.

#### Principijelna shema pomačnog registra:



**Primjena pomačnog registra:** Osim pamćenja kodne riječi, osnovna funkcija je pomak bitova u lijevo ili desno čime ostvarujemo množenje ili dijeljenje, te paralelno – serijska pretvorba.

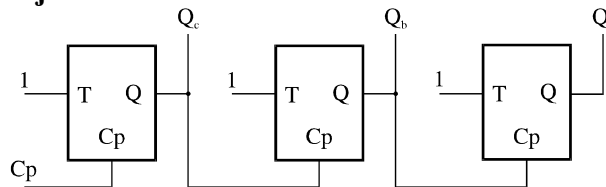
### 20.3. Brojilo

- definirati brojilo kao sklop
- principijelna shema brojila
- primjena brojila

#### Brojilo kao sklop:

Sklop koji se sastoji od više T ili JK bistabila koji su povezani tako da njihovo sljedeće stanje ovisi samo o prethodnom (prijelaz u sljedeće stanje nastaje u trenutku nastupanja taktnog signala).

#### Principijelna shema brojila:



**Primjena brojila:** Vrsta generatora sekvence koji prolaskom kroz sva stanja na izlazu generira konačnu sekvencu kodnih kompleksija. Ako na izlazu generira prirodni binarni niz tada govorimo o binarnom brojilu, a ako je riječ o BCD kodu tada imamo dekadsko brojilo.