Fakultet elektrotehnike, strojarstva i brodogradnje

Digitalni sustavi i strukture
Teorija
Računarstvo 750/3
2005./2006.

Izradili:

Anđelić Ines
Beara Marina
Matijević Marko
Sommer Andrija
Tomašić Berislav
Zarić Tanja
Zrno Josipa

Split, veljača 2006.

Sadržaj

I. DIO - DIGITALNI SUSTAVI I STRUKTURE

I. TREĆINA GRADIVA

1. PRIKAZ INFORMACIJA U DIGITALNIM SUSTAVIMA

1.1. Analogni i digitalni sustavi

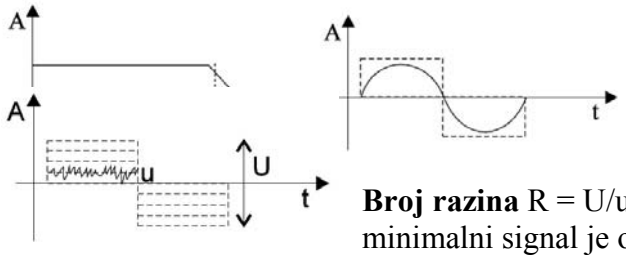
– činjenica, informacija, električni signal, modulacija
– definicija analognog sustava
– definicija digitalnog sustava

ČINJENICA – neka pojava postoji bez obzira da li je osjećamo.
INFORMACIJA – je saznanje čovjeka o biti neke stvari, događaja ili postupka.
ELEKTRIČNI SIGNAL – prijenos informacija električnom energijom.
MODULACIJA – Postupak mijenjanja neke od fizikalnih veličina električnog signala(napon, struja, frekvencija, faza, valni oblik) u skladu s promjenom informacije. Postupke modulacije dijelimo na digitalne i analogne.
Kod **ANALOGNIH** sustava informaciji se pridružuje neka veličina električnog signala prema izabranoj funkciji. To znači da će sva informacija biti sadržana u npr. naponu električnog signala.
Kod **DIGITALNIH** sustava informacija se najprije predstavi brojem, a onda taj broj električnim signalom. Za svaku znamenku broja koristi se po jedan električni signal.

1.2. Informacijski volumen i digitalni sustav

– sustav s niskim propustom, broj razina, kapacitet
– digitalni i analogni sustav
– paralelni i serijski prijenos

SUSTAV S NISKIM PROPUSTOM



Kod sustava s niskim propustom širina pojasa $B = f_g – f_d = f_g – 0 = f_g$
u jednom periodu signala f_g prenesemo dva signalna elementa. Odatle 2B signalnih elemenata u sekundi.
Broj razina R = U/u. Raspon signala ograničen dogovorom, a minimalni signal je ograničen smetnjama.
dogovor: D = log2(R) = ld(R) bita/sign. elementu
KAPACITET SUSTAVA izražava brzinu obrade, brzinu prijenosa, brzinu pristupa podacima, Kapacitet C = 2B·D [se/sek · bit/se = bit/sek]
Kod **ANALOGNIH** sustava informaciji se pridružuje neka veličina električnog signala prema izabranoj funkciji. To znači da će sva informacija biti sadržana u npr. naponu električnog signala.
Kod **DIGITALNIH** sustava informacija se najprije predstavi brojem, a onda taj broj električnim signalom. Za svaku znamenku broja koristi se po jedan električni signal.
Digitalni sustavi imaju niz prednosti i sve više u primjeni zamjenjuju analogne sustave. Međutim, analogna tehnologija zadržava svoju ulogu u području povezivanja digitalnih sustava s okolinom, koja je uglavnom analogna.
Serijski i paralelni prijenos:
Paralelni prijenos je prijenos informacije pomoću više kanala, dok je serijski prijenos, prijenos informacija pomoću jednog kanala. Istu količinu podataka možemo prenijeti serijski 1 kanalom k puta većom brzinom od paralelno: k kanala jediničnom brzinom.

I. DIO - DIGITALNI SUSTAVI I STRUKTURE

1.3. Kodovi i kodiranje

– definicija, jednoznačnost, razlučivost, kodiranje, dekodiranje
– podjela kodova, kodna riječ
– analogni i digitalni sustavi u odnosu na kodove

Kod je dogovoreno uspostavljen sustav simbola kojima označavamo pojmove(informacije) iz nekog skupa pojmova.
Kodiranje ima dva značenja: postupak konstrukcije nekog koda, primjena nekog koda kroz zamjenu pojma simbolom.
Jednoznačnost – jednom pojmu najmanje jedan simbol
Razlučivost – treba dovoljan broj simbola
Dekodiranje – postupak izdvajanja polazne informacije iz simbola.
Podjela kodova: posredni i neposredni.
Neposrednih kodovi – za svaki se pojam bira zasebni simbol.
Posredni kodovi – pojmu se najprije dodijeli kodna riječ ili kompleksija, a onda se kodna riječ prikaže pomoću konačnog skupa simbola koje zovemo slovima.
Kodna riječ ili kompleksija: kodnu riječ formiramo iz skupa elementarnih simbola, kompleksija: cjelina sastavljena od više dijelova
Analogni i digitalni sustavi u odnosu na kodove: analogni sustavi su sustavi neposrednog kodiranja. Kod njih vrijednost analognog signala ima značenje simbola. Digitalni sustavi su sustavi posrednog kodiranja. Kod njih vrijednost digitalnog signala ima značenje slova, a skup vrijednosti više signala značenje kodne riječi.

2. BROJEVNI SUSTAVI

2.1. Poliadski brojevni sustavi

– definicija poliadskog brojevnog sustava, svojstva
– zapis realnih brojeva
– odnos i pretvorba binarnog i heksadecimalnog
– analogni i digitalni sustavi prema brojevnom sustavu

Poliadski brojevni sustav: broj zapisujemo kompleksijom od n elemenata a_k, pomnoženih s potencijom baze s: $a = \sum_{k=0}^{n-1} a_k s^k = a_{n-1}s^{n-1} + a_{n-2}s^{n-2} + \ldots + a_1 s + a_0$

Svojstva: Znamenke a_k se samo napišu jedna iza druge, tako da je krajnja desna znamenka a_0. Svaka pozicija znamenke na lijevo od a_0 podrazumijeva množenje s potencijom baze s. Svaka pozicija ima svoju težinu, te je kod težinski.
Realni brojevi se zapisuju korištenjem zareza ili točke, kojima se označava pozicija znamenke a_0, iza koje se može dopisati potreban broj znamenki:

$$a = \sum_{k=-d}^{n-d-1} a_k s^k = a_{n-d-1}s^{n-d-1} + a_{n-d-2}s^{n-d-2} + \ldots + a_1 s + a_0 + a_{-1}s^{-1} + a_{-2}s^{-2} + \ldots + a_{-d}s^{-d}$$

Odnos binarnog i heksadecimalnog: $s_{16}=(s_2)^4$ – znači da se svaka znamenka heksadecimalnog broja može bez ostatka prikazati s četiri znamenke binarnog broja.
Pretvorba: $\frac{0001}{1}\frac{1101}{D}$ $1D_{16}=29_{10}$

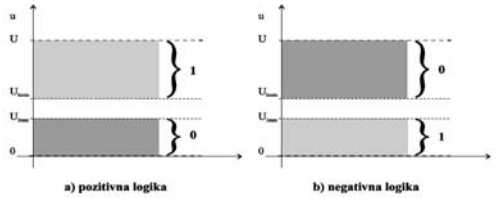
I. DIO - DIGITALNI SUSTAVI I STRUKTURE

Analogni i digitalni sustavi prema brojevnom sustavu
Analogni sustavi su sustavi neposrednog kodiranja gdje se za svaki pojam bira zasebni simbol. Dakle, povećanjem baze brojevnog sustava se sve veći skup brojeva može izraziti jednom znamenkom. Za dovoljno veliki s, bit će dovoljna jedna znamenka, a to nije ništa drugo nego analogni sustav s jednim električnim signalom.
Digitalni sustavi su sustavi posrednog kodiranja. Kod njih vrijednost digitalnog signala ima značenje slova, a skup vrijednosti više signala značenje kode riječi. Kodnom riječi predstavljamo informaciju.

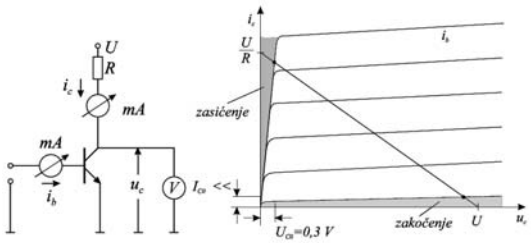
2.2. Izbor brojevnog sustava za digitalne sustave

– električni signal i dopuštena pogrješka
– pozitivna i negativna logika
– rad tranzistora kao sklopke
– kodne rijeci binarnog sustava

Električni signal – prijenos informacija električnom energijom.
Dopuštena pogrješka: električnom signalu podijelimo čitavo područje vrijednosti napona koje može poprimiti na s jednakih pojasa tako da je za svaku vrijednost napona vjerojatnost djelovanja smetnje jednaka. Različiti poremećaji utječu tako da signal koji je izvorno poslan unutar jednog pojasa prebace u neki od susjednih pojasa. Zbog toga nastojimo napon signala zadržati sredini pojasa. Na taj način je dopušteni poremećaj najveći.



Pozitivna logika je kada nižom naponskom razinom prikazujemo logičku 0, a višom naponskom razinom logičku 1.
Negativna logika je kada nižom naponskom razinom prikazujemo logičku 1, a višom naponskom razinom logičku 0.



Ako je struja baze zanemariva kažemo da je tranzistor zakočen, pa će izlaz biti spojen na izvor. Dakle, na izlazu sklopa će biti logička 1.
Ako je struja baze dovoljno velika s obzirom na β kažemo da je tranzistor u zasićenju, pa će izlaz biti spojen preko tranzistora na uzemljenje. Dakle, na izlazu sklopa će biti logička 0.

Kodne riječi binarnog sustava: u binarnom brojevnom sustavu ukupni broj mogućih kodnih riječi je $N=2^n$, a najveći broj koji se može zapisati pomoću n znamenki je $a_{max}=2^n-1$.

I. DIO - DIGITALNI SUSTAVI I STRUKTURE

2.3. Prikaz brojeva binarnim kodovima

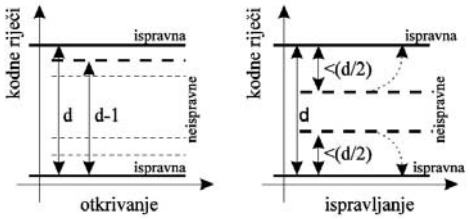
– prirodni binarni kod
– pretvorbe binarnog u dekadski i obrnuto
– BCD kodovi

Prirodni binarni kod: kodne riječi binarnih brojeva koristimo za kodiranje $\mathbb{N}_0$. Dakle, svaka kodna riječ predstavlja broj napisan u binarnom poliadskom brojevnom sustavu.
Pretvorba: Pretvorba dekadskog u binarni br. sustav se vrši algoritmom sukcesivnog dijeljenja. Pretvorba binarnog u dekadski br. sustav se vrši: $a_{10} = \sum_{k=0}^{n-1} a_k 2^k$ ili algoritmom sukcesivnog množenja.
Binarno kodirani dekadski brojevi (BCD) – svaka znamenka dekadskog broja zasebno je kodirana binarnom kodnom riječi. Za prikaz jedne dekadske znamenke potrebna je kodna riječ duljine 4 bita. BCD kodovi mogu biti komplementarni, težinski ili oboje. 8421BCD kod je težinski, a 2421BCD kod je komplementarno–težinski.

2.4. Primjene binarnih kodova

– kodna udaljenost, kodovi za otkrivanje i ispravljanje pogrješki
– osnovne aritmetičke operacije nad binarnim brojevima

Kodna udaljenost ili distanca je broj bita u kojima se razlikuju dvije kodne riječi istih varijabli i iste duljine.
Kodovi za otkrivanje pogreški:
POGREŠKA: prevodi 1→0, 0→1. Novo nastala kodna riječ razlikuje se u onoliko bita, koliko ih je promijenjeno djelovanjem smetnje. Pogrešku je moguće otkriti, ako smetnja prevodi korištenu (ispravnu) kodnu riječ u neku neiskorištenu (neispravnu). Višestruke pogrješke su manje vjerojatne. Kod konstruiramo tako da između bilo koje dvije ispravne kodne riječi distanca bude najmanje "d". Tada možemo **otkriti** d–1 struku pogrešku **ispraviti** (d/2) struku pogrešku.



Aritmetičke operacije nad binarnim brojevima: zbrajanje (obavlja se po znamenku vodeći računa o preteku), množenje (također se obavlja kao kod dekadskih brojeva), zbrajanje po modulu, inverz, oduzimanje po modulu, pomak u lijevo i pomaku desno.

I. DIO - DIGITALNI SUSTAVI I STRUKTURE

3. ARITMETIKA PO MODULU

3.1. Definicija sume po modulu kao grupe

– motivacija
– definicija grupe "suma po modulu"
– svojstva (postulati)

Motivacija: Sklopovi za zbrajanje u nekim slučajevima ne mogu prikazati rezultat s istim brojem bita kojim raspolažu pribrojnici zbog konačnosti stvarnih sklopova.
Definicija grupe "suma po modulu": grupa koja se sastoji od skupa F i operacije ⊕:

$$F = \{0, 2, ..., m-1\}; m \geq 2; m \in \mathrm{N}; \qquad a \oplus b = c = \mathrm{Re}\, z\left(\frac{a+b}{m}\right)$$

gdje je m modul, a operacija ⊕ zbrajanje po modulu m, definirana kao ostatak (Rez) cjelobrojnog dijeljenja zbroja a+b s modulom m.
Svojstva:

zatvorenost: ∀a, b ∈ F a ⊕ b = c : c ∈ F
asocijativnost: ∀a, b, c ∈ F a ⊕ b ⊕ c = (a ⊕ b) ⊕ c = a ⊕ (b ⊕ c)
komutativnost: ∀a, b ∈ F a ⊕ b = b ⊕ a
neutralni element: ∀a ∈ F ∃e ∈ F: a ⊕ e = e ⊕ a = a
inverz: ∀ a ∈ F ∃a′∈ F: a ⊕ a′ = a′⊕ a = e

3.2. Neutralni element i inverz za sumu po modulu

– definirati neutralni element, pokazati problem neodređenosti
– izračunati neutralni element
– definirati i izračunati inverz

Neutralni element: ∀a ∈ F ∃e ∈ F: a ⊕ e = e ⊕ a = a
Problem neodređenosti: Iz običnog zbroja jednoznačno dobijemo ostatak, ali iz ostatka ne možemo znati koliki je bio obični zbroj, pa trebamo uvrstiti faktore nesigurnosti k' i k":
a + e + k′·m = a + k″·m ⇒ e =(k″ − k′)·m ⇒ e = k·m
Faktore nesigurnosti k' i k" možemo zamijeniti jednim faktorom k=k'–k', kojeg biramo kako bi e zadovoljio svojstvo zatvorenosti. k = 0 ⇒ e = 0 ∈ F

Izračun neutralnog elementa: $a \oplus e = a \Rightarrow \mathrm{Rez}\left(\frac{a+e}{m}\right) = \mathrm{Rez}\left(\frac{a}{m}\right) \Rightarrow a + e = a \Rightarrow e = 0$

Inverz : ∀ a ∈ F ∃a′∈ F: a ⊕ a′ = a′⊕ a = e
Inverz ili dvojni komplement ili drugi komplement ili komplement po modulu dva, jednak je običnom komplementu uvećanom za jedan: $a' = \overline{a} + 1$
Izračun inverza: $a \oplus a' = 0 \Rightarrow a + a' = k \cdot m \Rightarrow a' = km - a$
Imamo dva slučaja: $a = 0 \Rightarrow k = 0 \Rightarrow a' = 0$
$a > 0 \Rightarrow k = 1 \Rightarrow a' = m - a$
Pa možemo pisati: $a' = \begin{cases} 0; & a = 0 \\ m - a; & a > 0 \end{cases}$

3.3. Binarni brojevni sustav i suma po modulu

– pokazati posljedice konačnosti sklopa
– izračunati inverz za binarni sustav
– definirati prvi i drugi komplement

Konačni sklop za obično zbrajanje:

$$\begin{array}{r} 1001 \equiv 9 \\ +\quad 1100 \equiv 12 \\ \hline 1\!\!\downarrow\!0101 \equiv 5 \end{array} \quad \rightarrow \quad 5 = 21_{\text{mod } 16} = rez\left(\frac{21}{16}\right) = 5$$

jer je ograničen na 4 bita: s=2; n=4; $a_{max}=2^n-1=15$; $N=2^n=16$
Sklop se ponaša kao da imamo sumu po modulu:

$$F = \{0,...,m-1\} = \{0,...,2^n-1\} = \{0,...,a_{max}\}$$

Povezali smo binarni brojevni sustav i sumu po modulu!

Izracun inverza za binarni sustav:

a'=m–a m=2^n $a_{max}=2^n-1 \rightarrow 2^n=m=a_{max}+1 \rightarrow a'= a_{max}+1-a = a_{max}-a+1$

$$a' = \sum_{k=0}^{n-1} \overline{a}_k 2^k + 1 \quad \rightarrow \quad a' = \left\{ \begin{array}{ll} a = 0; & a' = 0 \\ a > 0; & a' = \overline{a} + 1 \end{array} \right\} = \overline{a} \oplus 1$$

Prvi komplement ili obični komplement broja a: $\overline{a} = \sum_{k=0}^{n-1} (s - 1 - a_k) s^k = \sum_{k=0}^{n-1} \overline{a_k} s^k$

Inverz ili dvojni komplement ili **drugi komplement** ili komplement po modulu dva, jednak je običnom komplementu uvećanom za jedan: $a' = \overline{a} + 1$

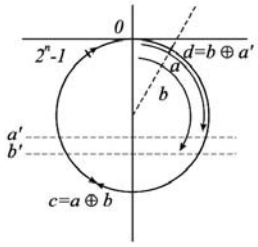
3.4. Primjena drugog komplementa

– prikazati oduzimanje po modulu
– komentirati kodiranje pozitivnih cijelih brojeva
– zbrajanje i oduzimanje pozitivnih cijelih brojeva
– komentirati kodiranje pozitivnih i negativnih brojeva
– zbrajanje i oduzimanje pozitivnih i negativnih brojeva

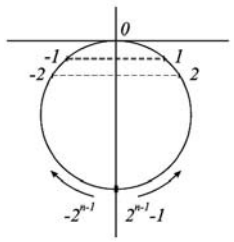
Oduzimanje po modulu:

$$a \ominus b = \mathrm{Rez}\left(\frac{a-b}{m}\right) = \mathrm{Rez}\left(\frac{a-b+km}{m}\right)^{k=1} = \mathrm{Rez}\left(\frac{a+m-b}{m}\right) = \mathrm{Rez}\left(\frac{a+b'}{m}\right) = a \oplus b'$$

odnosno, oduzimanje jednostavno ostvarimo pribrajanjem inverza.



Brojeve grafički prikazujemo na brojevnom pravcu, koji se prostire u beskonačnost. Zbog konačnosti sklopova i zbroja po modulu radije ih prikazujemo na kružnici. Smjer kazaljke na satu smatramo kretanjem rastućih pozitivnih brojeva.
Dva broja, a i b, možemo **zbrojiti**, i zbroj c dobiti grafičkim zbrajanjem lukova. Rezultat će biti točan dok je zbroj manji od $2^n - 1$ za n bitova. Ako je zbroj veći dolazi do preteka s najznačajnijeg bita.
Razliku d=b–a možemo dobiti oduzimanjem lukova. U praksi, razliku d računamo korištenjem inverza a i zbrajanjem s b: d=b+a'.



Brojevi s desne strane kružnice, koji slijede niz prirodnih binarnih brojeva 0, 1, ..., $2^{n-1}-1$ smatraju se pozitivnima. Njihovi inverzi s lijeve strane, koji idu smjerom obrnutim od kretanja kazaljke na satu, smatraju se negativnim brojevima, koji slijede niz –1, –2, ..., -2^{n-1}. Svim brojevima najznačajniji bit je bit predznaka(npr. 1 za negative).
Pri zbrajanju i oduzimanju sada vrijede ista pravila kao kad se brojevi koriste kao samo pozitivni. Jedino se kod zbrajanja ne smije prijeći donja točka, dok se kod oduzimanja mora prijeći.

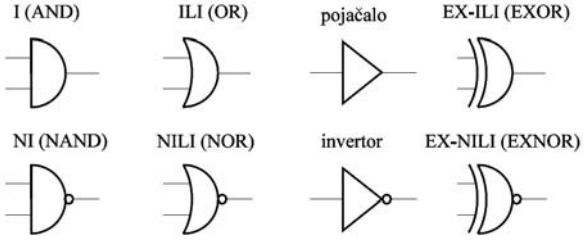
I. DIO - DIGITALNI SUSTAVI I STRUKTURE

4. NORMALNI ALGEBARSKI OBLICI

4.1. Koncept elementarnih logičkih sklopova

–motivacija, operatori algebre logike
–algebarski izraz, logički dijagram, shema sklopa
–elementarni logički sklopovi (logička vrata)
–elektromehanički logički krugovi

Motivacija: digitalni se sklopovi izrađuju korištenjem gotovih modula proizvedenih elektroničkom tehnologijom. To su elementarni logički sklopovi.



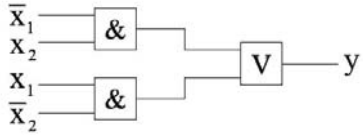
U algebri logike koriste se **operatori** konjunkcije (&), disjunkcije (V) i negacije (–).

x_1	x_2	$x_1 \& x_2$	$x_1 \vee x_2$	$\overline{x}_1$
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

Algebarski izrazi predstavljaju funkcijsku ovisnost o nezavisnim varijablama izraza. Na osnovi algebarskog izraza crtamo logički dijagram, te shemu sklopa.

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

$$y = \overline{x}_1 \cdot x_2 \vee x_1 \cdot \overline{x}_2$$



Operatore konjunkcije i disjunkcije možemo realizirati i elektromehaničkim relejima.
Relej je uređaj kod kojeg prolaskom struje kroz svitak nastaje magnetsko polje, a ono privuče kotvu na koju je pričvršćen kontakt. Mirni kontakt je trajno zatvoren, pa se aktiviranjem releja strujni krug prekida, dok je radni kontakt trajno otvoren, pa se aktiviranjem releja strujni krug zatvara.
Konjunkcija se ostvaruje serijskim, a disjunkcija paralelnim spojem radnih kontakata.
Negacija se ostvaruje mirnim kontaktom.

I. DIO - DIGITALNI SUSTAVI I STRUKTURE

4.2. Klasifikacija digitalnih tehnologija

–diskretne i integrirane tehnologije
–stupnjevi integracije
–vrste izlaza logičkih vrata

Diskretna izvedba odnosi se na izradu sklopova od pojedinačnih tranzistora i otpornika.
U diskretnu izvedbu spadaju: diodna, otporno tranzistorska i diodno tranzistorska tehnologija.
Integrirana izvedba odnosi se na izradu sklopa kao integriranog kruga, gdje su sve komponente integrirane na jednoj pločici silicija.
U integriranu izvedbu spadaju: tranzistorsko tranzistorska tehnologija, emiterski spregnuta, PMOS, NMOS i komplementarna unipolarna tehnologija.
Stupnjevi integracije:

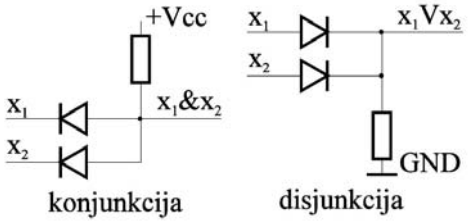
SSI (Small Scale Integration): niski stupanj integracije, do 100 tranzistora, do 10 logičkih vrata.
MSI (Medium Scale Integration): srednji stupanj integracije, do 1000 tranzistora, do 100 logičkih vrata.
LSI (Large Scale integration): visoki stupanj integracije, do 10000 tranzistora, do 1000 logičkih vrata.
VLSI (Very Large Scale Integration): vrlo visoki stupanj integracije preko 10000 tranzistora, više od 1000 logičkih vrata.

Vrste izlaza logičkih vrata:
Bipolarni izlazi(TP = totem–pole) generiraju vrijednosti 0 i 1. U logičkoj 1 služe kao izvorišta, a u logičkoj 0 kao potrošači struje.
Unipolarni izlazi(OC = open–collector) generiraju samo logičku 0, i mogu biti samo potrošači struje. Kod logičke 1 tranzistor se isključuje. Unipolarni izlazi se koriste kad je potrebno spojiti više izlaza na zajedničku točku.
Tri–state izlazi (TS, bipolarni sa trećim stanjem visoke impedancije) imaju karakteristike bipolarnih, a sposobnost isključenja omogućava spajanje na sabirnicu.

4.3. Diodna i diodno–tranzistorska logika

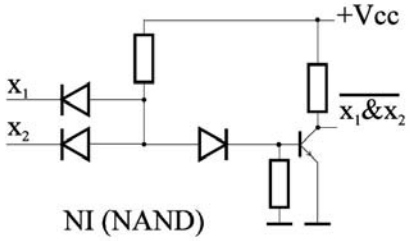
–karakteristike i osnovni sklopovi DL
–karakteristike i osnovni sklopovi DTL

Karakteristike i osnovni sklopovi DL:
Diodama se sprječava kratki spoj ulaznih varijabli. Mana diodne tehnike je u problemu generiranja nule i jedinice kod spajanja više I–ILI sklopova u seriju.



Karakteristike i osnovni sklopovi DTL

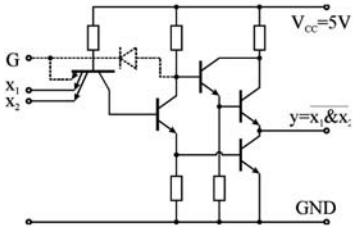
Promjenom vrijednosti otpornika manipulira se strujama baze, pa je moguće postići da sklop obavlja negaciju konjunkcije(NI) ili disjunkcije(NILI).



4.4. Tranzistorski–tranzistorska logika

– osnovni sklop TTL
– određivanje razina 0 i 1
– karakteristike familija TTL krugova

Osnovni sklop TTL:



Tranzistorsko tranzistorska tehnika nastala je integriranjem ulaznih dioda u višeemiterski tranzistor. Ako su svi ulazi u 1, PN spoj baza–kolektor višeemiterskog tranzistora vodi, te vodi srednji tranzistor koji se zove obrtač faze. Zbog toga teče struja baze donjeg izlaznog tranzistora, pa on vodi i na izlazu daje nulu. Gornji izlazni tranzistor ne vodi.

Ako je makar jedan ulaz u nuli, ulazni tranzistor vodi, te je obrtač faze zakočen. Stoga je i donji izlazni tranzistor zakočen. Gornji tranzistor dobiva sada struju baze s kolektorskog otpornika obrtača faze, te na izlazu generira jedinicu.

Razine: TTL tehnologija je postavila niz standarada u digitalnoj elektronici, od kojih su najvažniji napon napajanja 5 V i standardne razine za 0 i 1: 0V–0,8V za 0; 2,4V–5V za 1.

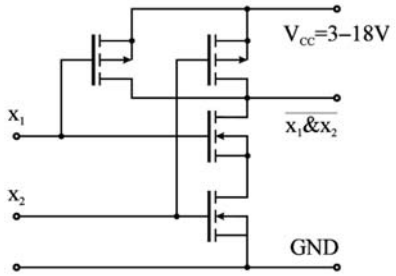
Karakteristike familija TTL krugova:

74xx = normalni,
74Lxx = niska potrošnja i brzina,
74Hxx = velika brzina i potrošnja,
74Sxx = normalni shottky,
74LSxx = shottky sa malom potrošnjom,
74ALSxx = novija familija sa manjom dimenzijom tranzistora
74Fxx = novija familija brzih TTL integriranih krugova

I. DIO - DIGITALNI SUSTAVI I STRUKTURE

4.5. Komplementarna MOS tehnologija

– osnovni sklop CMOS
– karakteristike CMOS logičkih vrata
– potrošnja CMOS vrata
– primjena CMOS u VLSI krugovima

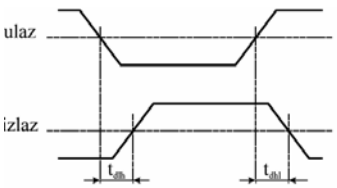


U **CMOS** tehnologiji koriste se tranzistori s osiromašenjem. Ako su oba ulaza u 1, oba serijski spojena donja tranzistora su otvorena, a oba gornja paralelno spojena tranzistora su zatvorena, pa je izlaz u 0. Ako je makar jedan ulaz u 0, jedan od gornjih tranzistora je otvoren, a jedan od donjih je zatvore, te je izlaz u 1.

Karakteristrike: upotreba komplementarnih (P i N kanalnih) MOS tranzistora koji rade u protufazi, velika brzina rada, te niska potrošnja energije.
Potrošnja energije CMOS integriranog kruga je niska, jer uvijek jedna od grana sklopa ne vodi. Potrošnja ovisi o broju preklapanja u jedinici vremena. Na visokim frekvencijama se približava potrošnji TTL i NMOS krugova.
Zbog svojih prednosti CMOS tehnologija je danas primarna tehnologija za proizvodnju svih vrsta digitalnih integriranih krugova, od pojedinačnih logičkih vrata niskog stupnja integracije do **VLSI** mikroprocesora i memorija s više desetaka milijuna tranzistora na jednoj pločici silicija.

4.6. Primjena elementarnih logičkih sklopova

– kašnjenje i brzina porasta
– standardne naponske razine 0 i 1
– ulazne i izlazne struje
– definicija faktora izlaznog grananja
– definicija faktora izlaznog grananja



Kašnjenje se definira kao vrijeme koje protekne između promjene na ulazu u sklop do promjene na izlazu sklopa. Kašnjenje mjerimo od sredine do sredine promjene. Posebno mjerimo vrijeme potrebno da sklop postigne jedinicu (t_{dlh}), a posebno vrijeme potrebno da postigne nulu (t_{dhl}).

Vrijeme porasta t_r i pada t_f je zapravo vrijeme u kojem se obavlja promjena, od 0 u 1 ili obratno. Ta vremena mjerimo od trenutka kad je dostignuto 10% početne do trenutka kad je dostignuto 90% konačne vrijednosti signala.

Standardne naponske razine 0 i 1: su drugi bitan čimbenik kompatibilnosti. Karakteristike ulaznog i izlaznog sklopa TTL logičkih vrata zahtijevaju nesimetrično postavljanje za 0 i 1, pa je određeno da su te granice za TTL i NMOS tehnologiju 0–0,8V max za logičku 0 i 2,4–5V min za logičku 1.
Ulazne i izlazne struje su struje koje se nalaze na ulazu i izlazu sklopa, a to su: I_{il} i I_{ih}, I_{ol} i I_{oh}. Važne su zbog toga jer utječu na faktore ulaznog i izlaznog grananja.

Faktora izlaznog grananja: nam kaže koliko standardnih ulaza možemo spojiti na jedan izlaz, a da pri tom naponi signala ostanu u propisanim granicama. Za TTL i NMOS tehnologiju te su granice 0,8V max za logičku 0 i 2,4V min za logičku 1.

$$FG_{izl} = \min (I_{ohm} / I_{ih0} ; I_{olm} / I_{il0})$$

Faktora ulaznog grananja: nam kaže koliko stvarni ulaz opterećuje izlaz na koji je spojen u odnosu na standardni ulaz za promatranu tehnologiju i varijantu izrade. Ovaj faktor je najčešće 1.

$$FG_{ul} = \max (I_{ih} / I_{ih0} ; I_{il} / I_{il0})$$

5. BOOLEOVA ALGEBRA

5.1. Booleova algebra i algebra logike

– definirati Booleovu algebru
– definirati logičke operatore
– definirati algebru logike
– definirati redoslijed operacija

Booleova algebra je matematička struktura definirana kao:

B.A. ={G, x, =, S}

Gdje je: G –skup operatora algebre;
= –operator jednakosti;
S ={0,1} – skup Booleovih konstanti 0 i 1;
x e S – Booleova varijabla koja uzima vrijednost iz S.

Logički operatori:
Konjunkcija dvaju sudova je istinita ako su oba suda istinita.
Disjunkcija dvaju sudova je istinita ako je istinit makar jedan od njih.
Negacija nekog suda je istinita ako je ulazni sud neistinit.

Algebra logike je Booleova algebra kod koje je skup G:
G = {&, V , –}
A.L. = {&, V, –, =, x, S={0,1}}.
Redoslijed operacija: kod računana algebarskih izraza prednost se daje negaciji, pa konjunkciji, i na kraju disjunkciji.

5.2. Postulati algebre logike

– navesti poimence sve postulate i njihove formule
– pokazati distributivnost tablicom istine i Vennovim dijagramom

1. Zatvorenost

a) $\forall x_1, x_2 \in S \Rightarrow x_1 \vee x_2 \in S$ (disjunkcija je zatvorena u S)

b) $\forall x_1, x_2 \in S \Rightarrow x_1 \,\&\, x_2 \in S$ (konjunkcija je zatvorena u S)

c) $\forall x_1 \in S \Rightarrow \overline{x}_1 \in S$ (negacija je zatvorena u S)

2. Neutralni element

a) $\forall x_1, 0 \in S \Rightarrow x_1 \vee 0 = x_1$ (0 je neutralni element za disjunkciju)

b) $\forall x_1, 1 \in S \Rightarrow x_1 \,\&\, 1 = x_1$ (1 je neutralni element za konjunkciju)

3. Komutativnost

a) $\forall x_1, x_2 \in S \Rightarrow x_1 \vee x_2 = x_2 \vee x_1 \in S$ (disjunkcija je komutativna)

b) $\forall x_1, x_2 \in S \Rightarrow x_1 \,\&\, x_2 = x_2 \,\&\, x_1 \in S$ (konjunkcija je komutativna)

4. Distributivnost

a) $\forall x_1, x_2, x_3 \in S \Rightarrow x_1 \vee (x_2 \,\&\, x_3) = (x_1 \vee x_2) \,\&\, (x_1 \vee x_3)$

(disjunkcija je distributivna s obzirom na konjunkciju)

b) $\forall x_1, x_2, x_3 \in S \Rightarrow x_1 \,\&\, (x_2 \vee x_3) = (x_1 \,\&\, x_2) \vee (x_1 \,\&\, x_3)$

(konjunkcija je distributivna s obzirom na disjunkciju)

5. Komplementiranje

a) $\forall x_1 \in S \Rightarrow x_1 \vee \overline{x}_1 = 1$

(disjunkcija nenegirane i negirane varijable jednaka je jedinici)

b) $\forall x_1 \in S \Rightarrow x_1 \,\&\, \overline{x}_1 = 0$

(konjunkcija nenegirane varijable jednaka je nuli)

6. Asocijativnost

a) $\forall x_1, x_2, x_3 \in S \Rightarrow x_1 \vee (x_2 \vee x_3) = (x_1 \vee x_2) \vee x_3$

(disjunkcija je asocijativna)

b) $\forall x_1, x_2, x_3 \in S \Rightarrow x_1 \,\&\, (x_2 \,\&\, x_3) = (x_1 \,\&\, x_2) \,\&\, x_3$

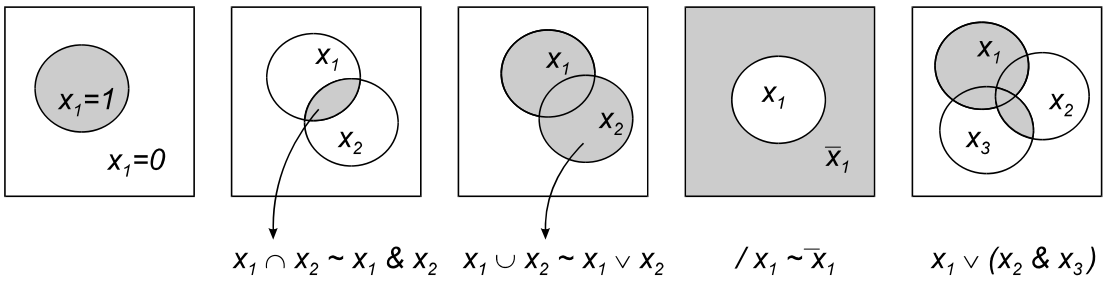
(konjunkcija je asocijativna)

Pokazati distributivnost tablicom istine i Vennovim dijagramom:

x_1	x_2	x_3	$x_2 \,\&\, x_3$	$x_1 \vee (x_2 \,\&\, x_3)$	$x_1 \vee x_2$	$x_1 \vee x_3$	$(x_1 \vee x_2) \,\&\, (x_1 \vee x_3)$
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

REDOSLIJED:

1. **negacija (i sve ispod)**
2. **konjunkcija**
3. **disjunkcija**



5.3. Teoremi algebre logike s jednom varijablom

– navesti poimence sve teoreme s jednom varijablom i njihove formule s dokazima

T1. Apsorpcija za disjunkciju $x_1 \vee 1 = 1$

Jedinica disjunktivno vezana s nekim izrazom apsorbira taj izraz. Dokaz:

$\equiv (x_1 \vee 1)\cdot 1 = (x_1 \vee 1)\cdot(x_1 \vee \overline{x}_1) = x_1 \vee (1\cdot \overline{x}_1) = x_1 \vee \overline{x}_1 = 1$

T2. Idepotentnost za disjunkciju $x_1 \vee x_1 = x_1$

Disjunkcija nekog izraza sa samim sobom jednaka je tom izrazu. Koristi se za sažimanje ili proširenje algebarskog izraza postojećim članom. Dokaz:

$\equiv (x_1 \vee x_1)\cdot 1 = (x_1 \vee x_1)\cdot(x_1 \vee \overline{x}_1) = x_1 \vee (x_1 \cdot \overline{x}_1) = x_1 \vee 0 = x_1$

T3. Idepotentnost za konjunkciju $x_1 \cdot x_1 = x_1$

Konjunkcija nekog izraza sa samim sobom jednaka je tom izrazu .Koristi se za sažimanje ili proširenje algebarskog izraza postojećim članom. Dokaz:

$\equiv x_1 \cdot x_1 \vee 0 = x_1 x_1 \vee x_1 \overline{x}_1 = x_1(x_1 \vee \overline{x}_1) = x_1 \cdot 1 = x_1$

T4. Dvostruka negacija $\overline{\overline{x}}_1 = x_1$

Dokazuje se tablicom:

x_1	$\overline{x}_1$	$\overline{(\overline{x}_1)} = \overline{\overline{x}}_1$
0	1	0
1	0	1

T5. Apsorpcija za konjunkciju $x_1 0 = 0$

Nula konjunktivno vezana s nekim izrazom apsorbira taj izraz. Dokaz:

$\equiv x_1 0 \vee 0 = x_1 0 \vee x_1 \overline{x_1} = x1\left(0 \vee \overline{x_1}\right) = x_1 \overline{x_1} = 0$

5.4. Teoremi algebre logike s dvije varijable

– navesti poimence sve teoreme s dvije varijable i njihove formule s dokazima

T1. DeMorganov teorem za disjunkciju $\overline{x_1 \vee x_2} = \overline{x}_1 \cdot \overline{x}_2$

Dokazuje se indukcijom: ako je lijeva strana jednaka desnoj (A=A), mora vrijediti postulat o komplementiranju u oba oblika. Pišemo:

$$A = \overline{x_1 \vee x_2}\ ;\quad \overline{A} = \overline{\overline{x_1 \vee x_2}} = x_1 \vee x_2\ ;\quad A = \overline{x}_1\,\overline{x}_2$$

$$A \vee \overline{A} = 1 \quad \Rightarrow \quad \overline{x}_1 \cdot \overline{x}_2 \vee (x_1 \vee x_2) = 1$$

$$\overline{x}_1 \cdot \overline{x}_2 \vee (x_1 \vee x_2) = (x_1 \vee x_2 \vee \overline{x}_1)\cdot(x_1 \vee x_2 \vee \overline{x}_2) = 1\cdot 1 = 1$$

$$A \cdot \overline{A} = 0 \quad \Rightarrow \quad \overline{x}_1 \cdot \overline{x}_2 \cdot (x_1 \vee x_2) = 0$$

$$\overline{x}_1 \cdot \overline{x}_2 \cdot (x_1 \vee x_2) = (\overline{x}_1 \cdot \overline{x}_2 \cdot x_1) \vee (\overline{x}_1 \cdot \overline{x}_2 \cdot x_2) = 0 \vee 0 = 0$$

T2.DeMorganov teorem za konjunkciju $\overline{x_1 \cdot x_2} = \overline{x}_1 \vee \overline{x}_2$

Negacija konjunkcije dviju varijabli jednaka je disjunkciji negiranih varijabli.

$$\overline{x_1 \cdot x_2} = \overline{x}_1 \vee \overline{x}_2 \quad /\overline{\ }$$

$$\overline{\overline{x_1 \cdot x_2}} = \overline{\overline{x}_1 \vee \overline{x}_2}$$

$$x_1 \cdot x_2 = \overline{\overline{x}}_1 \cdot \overline{\overline{x}}_2 = x_1 \cdot x_2$$

I. DIO - DIGITALNI SUSTAVI I STRUKTURE

6. BOOLEOVE FUNKCIJE

6.1. Booleova funkcija kao preslikavanje

– definirati skup kodnih rijeci nad skupom varijabli
– definirati Booleovu funkciju kao preslikavanje
– definirati jednostavni sklop i vezu sa Booleovom funkcijom

Definirati skup kodnih riječi nad skupom varijabli:
Ako je skup X svih n varijabli x: $X = \{x_1, x_2, \ldots, x_n\}$
tada je Pn(X) skup svih kodnih riječi varijabli x

$$P_n(X) = \{00\ldots0,\ 00\ldots1, \cdots\cdots, 11\ldots0,\ 11\ldots1\}$$

Definirati Booleovu funkciju kao preslikavanje:
Booleova funkcija y=f(x , x ,...,x) je preslikavanje nadskupa P (x) u skup S = {0,1}
Definirati jednostavni sklop i vezu sa Booleovom funkcijom:
Jednostavni sklop je digitalni sklop sa jednim izlazom, koji na osnovu ulaznih varijabli (Koje imaju konkretnu vrijednost 0 ili 1) daje izlaznu funkciju ulaznih varijabli y. Odnosno vidimo da jednostavni sklop obavlja preslikavanje Booleova funkcije.

6.2. Osnovno zapisivanje i vrste Booleovih funkcija

– zapis tablicom istine
– standardni oblik tablice istine, broj redaka i njihove oznake
– potpuno i nepotpuno specificirane funkcije
– univerzalna funkcija

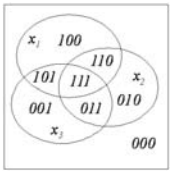
Zapis tablicom istine:
S lijeve strane zapišemo sve kodne riječi prirodnim binarnim nizom.
S desne strane napišemo vrijednosti funkcije (vrijednosti y 0, 1 i R).
Takvu strukturu zovemo tablicom isitne.
Standardni oblik tablice istine, broj redaka i njihove oznake:
S lijeve strane tablice se nalaze ulazne varijable (x_1,x_2...x_n), a s desne strane funkcije(f_1,f_2,...f_k). Svaki redak označavamo rednim brojem i od 0 do 2^n–1, koji odgovara vrijednosti pripadne kompleksije varijabli promatrane kao prirodni binarni broj.
Potpuno specifirana funkcija je funkcija, kod koje je preslikavanje definirano za sve kodne riječi ulaznih varijabli.
Nepotpuno specifirana funkcija je funkcija, kod koje je preslikavanje definirano samo za neke kodne riječi ulaznih varijabli.

I. DIO - DIGITALNI SUSTAVI I STRUKTURE

6.3. Grafički zapis Booleovih funkcija

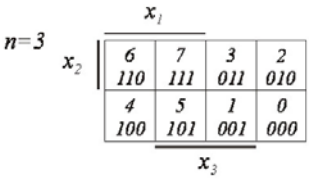
– Vennovi dijagrami
– Veitchevi dijagrami
– standardni oblik do n=6 varijabli

Vennovi dijagrami



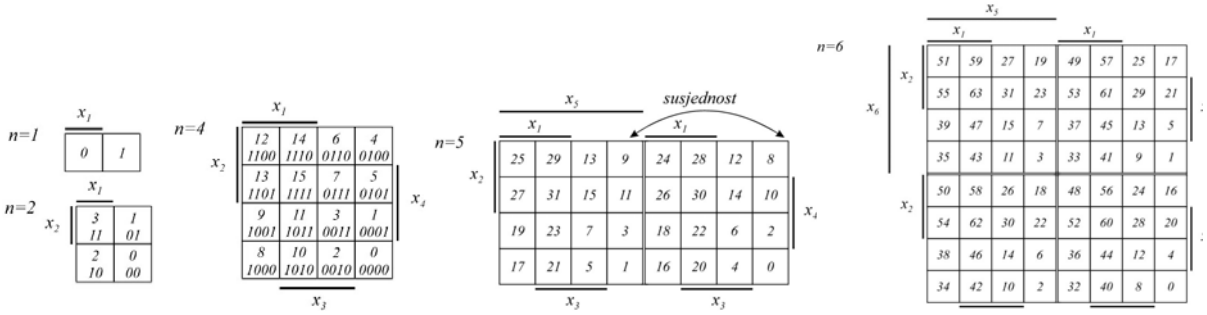
Kod Vennovih dijagrama definiramo područja u okviru univerzalnog skupa nad kojima se pojedina varijabla definira kao logička jedinica. Svako presjecište definirano je jednom kombinacijom vrijednosti varijabli koja ima značenje kompleksije varijabli, pa nad njim možemo definirati vrijednost funkcije, odnosno preslikavanje u skupu S={0,1}.

Veitchevi dijagrami



Veitchev dijagram je standardizirani oblik grafičkog prikaza svih kodnih riječi nekih varijabli, nastao stiliziranim crtanjem Vennovih dijagrama. Kod Veitchevog dijagrama područja su formirana u obliku kvadrata koji je podijeljen na polovine u vertikalnom i horizontalnom smjeru. S vanjske strane pišemo oznaku područja kao svojevrsnu koordinatu, a u svaki kvadrat se upisuje kodna riječ.

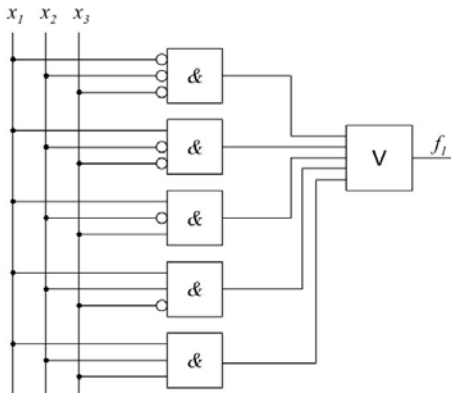
Standardni oblik do n=6 varijabli



6.4. Ostali načini zapisa Booleove funkcije

– Grayev kod i K–tablice
– logički dijagram
– shema sklopa na osnovi PDNO i PKNO

Grayev kod je kod kod kojeg su susjedni bitovi kodirani tako da se razlikuju u samo jednom bit–u. Koristi se kod optičkih senzora položaja ili kuta.
Karnaughove (K) tablice su dvodimenzionalni tablični zapis funkcije koji rezultira oblikom sličnim kao Veitchev dijagram. Mana K tablica je u otežanom očitavanju pripadne kodne riječi, pa se koriste rjeđe od Veitchevih dijagrama.
Logički dijagram: je grafički oblik zapisa, koji predstavlja blok shemu funkcije, i to na način da su varijable i međurezultati prikazani linijama, a operatori blokovima.

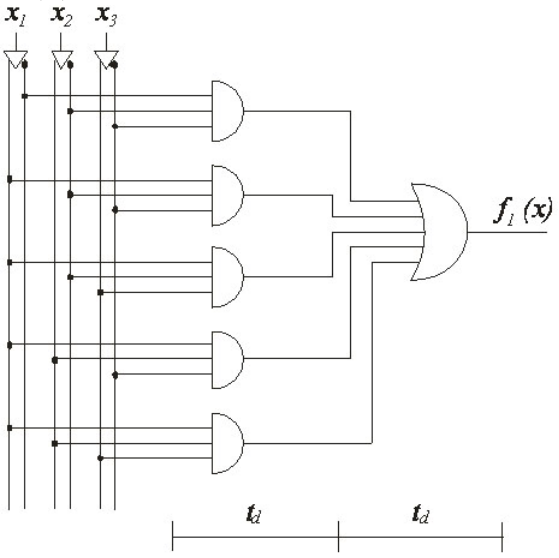


S obzirom da raspolažemo elementarnim sklopovima koji obavljaju konjunkciju, disjunkciju i negaciju, a varijable funkcije predstavljamo električnim signalima, možemo konstruirati sklop po strukturi sličan logičkom dijagramu, koji u stvarnosti realizira zadanu funkciju.

Shema sklopa na osnovi:

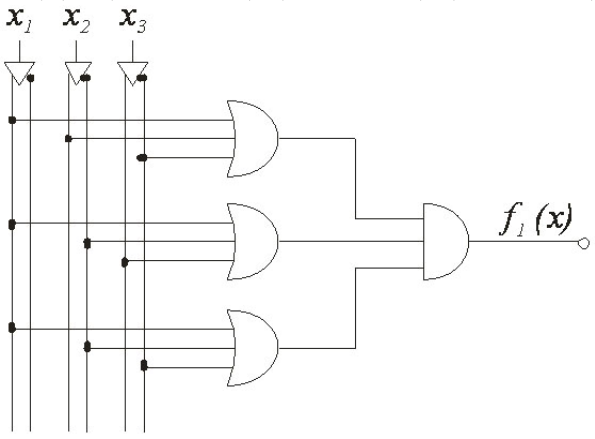
PDNO:

$$f_1(x) = \overline{x}_1\overline{x}_2\overline{x}_3 \vee x_1\overline{x}_2\overline{x}_3 \vee x_1\overline{x}_2x_3 \vee x_1x_2\overline{x}_3 \vee x_1x_2x_3$$



PKNO:

$$f_1(x) = (x_1 \vee x_2 \vee \overline{x}_3)\cdot(x_1 \vee \overline{x}_2 \vee x_3)\cdot(x_1 \vee \overline{x}_2 \vee \overline{x}_3)$$



I. DIO - DIGITALNI SUSTAVI I STRUKTURE

7. NORMALNI ALGEBARSKI OBLICI

7.1. Algebarski zapis potpunim normalnim oblicima

– motivacija
– definicija PDNO i minterma
– definicija PKNO i maksterma

Motivacija:
Normalni algebarski oblici omogućavaju:
– moguće ih je napisati neposredno iz tablice istine
– omogućavaju izradu sklopa s najmanjim kašnjenjem
– sklop ima jednoliko kašnjenje
– moguće ih je minimizirati egzaktnim postupcima
– garantiran je prijelaz na NI i NILI operatore

PDNO je disjunkcija svih onih MINTERMA mi za koje je vrijednost funkcije i–tog retka Ti jednaka jedinici: $f(x_1, x_2, \ldots, x_n) = \bigvee_{i=0}^{2^n-1} m_i \cdot T_i$

MINTERM m_i i–tog retka tablice istine je konjunkcija SVIH varijabli tako da su one koje u pripadnoj kodnoj riječi imaju vrijednost nula negirane, a one u jedinici nenegirane:

$$m_3(x_1, x_2, x_3)_{011} = \overline{x}_1 \cdot x_2 \cdot x_3$$

PKNO je konjunkcija svih onih MAKSTERMA M_i za koje je vrijednost funkcije i–tog retka Ti jednaka nuli: $f(x_1, x_2, \ldots, x_n) = \bigwedge_{i=0}^{2^n-1} (M_i \vee T_i)$

MAKSTERM M_i i–tog retka tablice istine je disjunkcija SVIH varijabli tako da su one koje u pripadnoj kodnoj riječi imaju vrijednost jedan negirane, a one u nuli nenegirane:

$$M_3(x_1, x_2, x_3)_{011} = x_1 \vee \overline{x}_2 \vee \overline{x}_3$$

7.2. Svojstva negirane funkcije

– svojstva potpunih normalnih oblika
– definicija negirane funkcije
– dokaz višestrukih DeMorganovih teorema

Svojstva potpunih normalnih oblika:
Disjunkcija svih minterma jednaka je jedinici, a konjunkcija svih maksterma jednaka je nuli:

$\bigvee_{i=0}^{2^n-1} m_i = 1$ $\qquad$ $\bigwedge_{i=0}^{2^n-1} M_i = 0$

Negirani minterm jednak je maksterm istog retka: $\overline{m}_i = M_i$ $\qquad$ $\overline{M}_i = m_i$

Disjunkcija minterma i maksterma istog retka je uvijek jednaka jedinici , a njihova je konjukcija jednaka nuli: $m_i \vee M_i = 1$ $\qquad$ $m_i M_i = 0$

Konjunkcija različitih minterma je uvijek jednaka nuli, a disjunkcija različitih maksterma jedinici: $m_i \; m_{j\, i \neq j} = 0$ $\qquad$ $M_i \vee M_{j\, i \neq j} = 1$

Negirana funkcija je funkcija koja ima vrijednost 1 tamo gdje izvorna funkcija ima vrijednost 0, a 0 tamo gdje izvorna funkcija ima vrijednost 1. Ako je izvorna funkcija nepotpuno specifirana, negirana je nepotpuno specifirana za iste retke tablice istine.

Dokaz negirane funkcije pomoću DeMorganovih teorema:

$$f(x)=\bigvee_{i=0}^{2^n-1} m_i T_i \Rightarrow$$

$$\bar{f}(x)=\overline{\bigvee_{i=0}^{2^n-1} m_i T_i}=\bigwedge_{i=0}^{2^n-1}\overline{m_i T_i}=\bigwedge_{i=0}^{2^n-1}(\bar{m}_i \vee \bar{T}_i)=\bigwedge_{i=0}^{2^n-1}(M_i \vee \bar{T}_i)$$

$$\bar{f}(x)=\overline{\bigwedge_{i=0}^{2^n-1}(M_i \vee T_i)}=\bigvee_{i=0}^{2^n-1}\overline{M_i \vee T_i}=\bigvee_{i=0}^{2^n-1}\overline{M}_i \cdot \bar{T}_i=\bigvee_{i=0}^{2^n-1} m_i \cdot \bar{T}_i$$

Dokaz dobijemo neposredno preko $f \vee \bar{f}=1$ i $f \wedge \bar{f}=0$

7.3. Minimalni normalni oblici

– definicija MDNO
– definicija MKNO

Minimalni disjunktivni normalni oblik (MDNO) je disjunkcija nužnih elementarnih članova tipa minterma. Član tipa minterma je konjunkcija nekih ili svih varijabli, negiranih prema pravilu pisanja minterma. Elementarni član je onaj koji nema susjeda. Nužni elementarni član je onaj, bez kojeg bi vrijednost funkcije bila poremećena.
Minimalni konjuktivni normalni oblik (MKNO) je konjunkcija nužnih elementarnih članova tipa maksterma. Član tipa maksterma je disjunkcija nekih ili svih varijabli, negiranih prema pravilu pisanja minterma. Elementarni član je onaj koji nema susjeda. Nužni elementarni član je onaj, bez kojeg bi vrijednost funkcije bila poremećena.

7.4. Razbijanje PDNO na preostale funkcije

– algebarski postupak
– postupak Veitchevog dijagrama
– primjena preostalih funkcija

Algebarski postupak:
Mintermi u raspisanom općem obliku PDNO funkcije:

$$\underset{\{x_1,x_2,x_3\}}{f(x)}=\bigvee_{i=0}^{2^n-1} m_i \cdot T_i=\bar{x}_1\bar{x}_2\bar{x}_3 T_0 \vee \bar{x}_1\bar{x}_2 x_3 T_1 \vee \bar{x}_1 x_2 \bar{x}_3 T_2 \vee \bar{x}_1 x_2 x_3 T_3 \vee$$

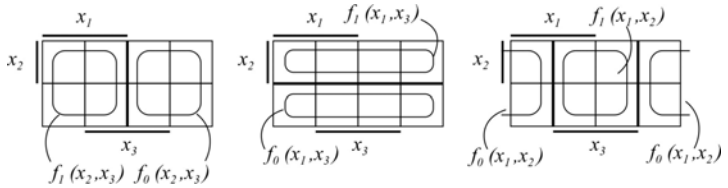
$$\vee x_1\bar{x}_2\bar{x}_3 T_4 \vee x_1\bar{x}_2 x_3 T_5 \vee x_1 x_2 \bar{x}_3 T_6 \vee x_1 x_2 x_3 T_7$$

imaju zajedničke članove, npr. za x_1, x_2 to su $\bar{x}_1\bar{x}_2$, $\bar{x}_1 x_2$, $x_1\bar{x}_2$, $x_1 x_2$, koje možemo izlučiti na osnovi svojstva distributivnosti. U zagradama su ostali neki izrazi koji su očito PDNO funkcija varijable x_3, a koje su preuzele vrijednosti izvorne funkcije T_1 do T_7. Možemo pisati: $f(x)=\bar{x}_1\cdot\bar{x}_2\cdot f_0(x_3)\vee\bar{x}_1\cdot x_2\cdot f_1(x_3)\vee x_1\cdot\bar{x}_2\cdot f_2(x_3)\vee x_1\cdot x_2\cdot f_3(x_3)$, gdje su funkcije f_0 do f_3 **parcijalne** ili **preostale funkcije**, a varijabla x_3 (u ovom primjeru) preostala varijabla:

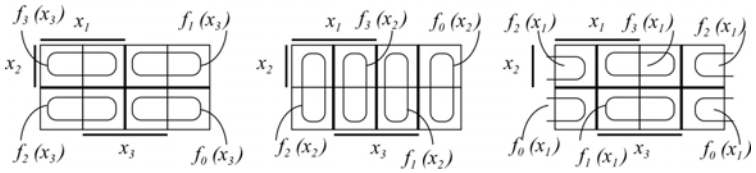
$$f(x)=m_0(x_1x_2)f_0(x_3)\vee m_1(x_1x_2)f_1(x_3)\vee m_2(x_1x_2)f_2(x_3)\vee m_3(x_1x_2)f_3(x_3)=$$

$$=\bigvee_{j=0}^{2^m-1} m_j(x_1\cdots x_m)f_j(x_{m+1}\cdots x_n);\quad f_j(x_{m+1}\cdots x_n)=\bigvee_{k=0}^{2^{n-m}-1} m_k(x_{m+1}\cdots x_n)T_{j\cdot 2^{n-m}+k}$$

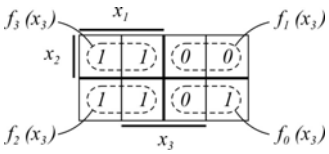
Postupak Veitchevog dijagrama:
Pogodan postupak izjednačavanja preostalih funkcija je postupak Veitchevog dijagrama. Veitchev dijagram se raspada na dijelove izdvajanjem **m** varijabli. Međutim, zbog svojevrsne simetričnosti dijagrama, ti dijelovi su više–manje suvisli, bez obzira kojih **m** varijabli izdvojili.



Rastavljanje na parcijalne funkcije Veitchevim dijagramom, n=3, m=1



Rastavljanje na parcijalne funkcije Veitchevim dijagramom, n=3, m=2



Rastavljanje na parcijalne funkcije za primjer y_1 po x_1 i x_2

Preostale funkcije se koriste kod realizacije multiplekserskog stabla.

8. POTPUNI SKUPOVI FUNKCIJA

8.1. Elementarne funkcije

– definirati broj mogućih Booleovih funkcija za n varijabli
– analizirati funkcije jedne varijable
– analizirati funkcije dvije varijable

Broj mogućih Booleovih funkcija za n varijabli:
Tablica istine za n varijabli ima 2^n redaka. Funkciju definiramo stupcem koji možemo smatrati kodnom riječi od $N=2^n$ bita. Takvih kodnih riječi ima $2^N = 2^{2^n}$ što znači da imamo upravo toliko različitih Booleovih funkcija za n varijabli.

Funkcije jedne varijable:
Za n=1 varijabli imamo $2^2=4$ funkcija, a to su prema tablici

x_1	f_0	f_1	f_2	f_3
0	0	1	0	1
1	0	0	1	1

$f_0(x_1)=0$ → konstanta 0, $f_1(x_1)=\overline{x}_1$ → negacija, $f_2(x_1)=x_1$ → identitet i $f_3(x_1)=1$ → konstanta 1.

Funkcije dviju varijabli:
Za n=2 varijabli imamo 2^4=16 funkcija. To su:

x_1	x_2	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Među ovim funkcijama prepoznajemo funkcije jedne varijable, a u drugu skupinu funkcija možemo svrstati konjunkciju, disjunkciju i njihove negacije:

$f_1=\overline{x_1 \vee x_2}$ → NILI (NOR, Pierce), $f_7=\overline{x_1 x_2}$ → NI(NAND, Shaeffer), $f_8= x_1 x_2$ → I (AND, konjunkcija) i $f_{14}= x_1 \vee x_2$ → ILI (OR, disjunkcija).
U treću skupinu svrstat ćemo sumu po modulu i njenu negaciju:

$f_6= x_1 \oplus x_2$ → ekskluzivno ILI(zbroj po modulu) i $f_9= x_1 \equiv x_2 = \overline{x_1 \oplus x_2}$ → ekvivalencija (negacija sume po modulu).
Preostalih šest funkcija spada u kategoriju implikacija i u praksi nemaju značenja.

8.2. Potpuni skup funkcija

– cilj razmatranja
– definicija potpunog skupa funkcija algebre logike
– dokazivanje potpunosti

Cilj razmatranja: Osnovni kriterij uspješnosti neke Booleove algebre je mogućnost zapisivanja proizvoljne Booleove funkcije konačnim algebarskim izrazom. Stoga trebamo izabrati skup operatora koji omogućava zapisivanje proizvoljne funkcije i takav se skup zove potpuni skup funkcija algebre logike.
Potpuni skup funkcija algebre logike je takav skup operatora s pomoću kojega se konačnim algebarskim izrazom može zapisati proizvoljna Booleova funkcija.
Skup konjunkcije, disjunkcije i negacije je potpuni skup funkcija algebre logike.
Potpunost bilo kojeg drugog skupa dokazujemo tako, da pokušamo izraziti konjunkciju, disjunkciju i negaciju. Za skup operatora kojim uspijemo izraziti konjunkciju, disjunkciju i negaciju zaključujemo da je potpun.

8.3. Dokazati potpunost za (I, NE) i (NI)

– dokazati potpunost za I, NE
– dokazati potpunost za NI
– komentirati problem i način zapisa

Potpunost za I, NE:

Skup {&,–} je potpun. Dokaz: $x_1 \vee x_2 = \overline{\overline{x_1 \vee x_2}} = \overline{\overline{x_1} \cdot \overline{x_2}}$

Potpunost za NI:

Shaefferov operator ({↑}, Shaffer, NI, NAND, $\overline{x_1 \cdot x_2}$) je sam za sebe potpuni skup funkcija algebre logike. Dokaz:

$$x_1 \uparrow x_1 = \overline{x_1 \cdot x_1} = \overline{x_1} \quad \text{ili} \quad x_1 \uparrow 1 = \overline{x_1 \cdot 1} = \overline{x_1}$$
$$(x_1 \uparrow x_1) \uparrow (x_2 \uparrow x_2) = \overline{x_1} \uparrow \overline{x_2} = \overline{\overline{x_1} \cdot \overline{x_2}} = x_1 \vee x_2$$
$$(x_1 \uparrow x_2) \uparrow (x_1 \uparrow x_2) = \overline{\overline{x_1} \uparrow \overline{x_2}} = \overline{\overline{x_1 \cdot x_2}} = x_1 \cdot x_2$$

Problem i način zapisivanja:

Da bi se izbjegla nesigurnost u stvarno značenje algebarskih izraza, u praksi se ne koristi simbol ↑ već se za zapis NI operatora koristi sustav {&,–}. U takvom zapisu prepoznat ćemo negiranu konjunkciju dviju ili više varijabli kao NI operator.

8.4. Dokazati potpunost za(ILI, NE) i (NILI)

– dokazati potpunost za ILI, NE
– dokazati potpunost za NILI
– komentirati problem i način zapisa

Potpunost za ILI, NE:

Skup {∨,–} je potpun. Dokaz: $x_1 \cdot x_2 = \overline{\overline{x_1 \cdot x_2}} = \overline{\overline{x_1} \vee \overline{x_2}}$

Potpunost za NILI:

Pierce operator ({↓}, Pierce, NILI, NOR, $\overline{x_1 \vee x_2}$) je sam za sebe potpuni skup funkcija algebre logike. Dokaz:

$$x_1 \downarrow x_1 = \overline{x_1 \vee x_1} = \overline{x_1} \cdot \overline{x_1} = \overline{x_1} \quad \text{ili} \quad x_1 \downarrow 0 = \overline{x_1 \vee 0} = \overline{x_1}$$
$$(x_1 \downarrow x_1) \downarrow (x_2 \downarrow x_2) = \overline{\overline{x_1} \vee \overline{x_2}} = \overline{\overline{x_1}} \cdot \overline{\overline{x_2}} = x_1 \cdot x_2$$
$$(x_1 \downarrow x_2) \downarrow (x_1 \downarrow x_2) = \overline{x_1 \downarrow x_2} = \overline{\overline{x_1 \vee x_2}} = x_1 \vee x_2$$

Problem i način zapisivanja:

Da bi se izbjegla nesigurnost u stvarno značenje algebarskih izraza, u praksi se ne koristi simbol ↓ već se za zapis NILI operatora koristi sustav {∨,–}. U takvom zapisu prepoznat ćemo negiranu disjunkciju dviju ili više varijabli kao NILI operator.

I. DIO - DIGITALNI SUSTAVI I STRUKTURE

9. MINIMIZACIJA NORMALNIH OBLIKA

9.1. Kriteriji minimizacije

– definirati zahtjeve na sklop
– definirati minimalnost sklopa
– definirati zahtjeve na algebarski oblik
– navesti i pokazati svojstva normalnih oblika

Zahtjevi na sklop:
Cilj minimizacije je da konačan sklop bude ekonomičan u proizvodnji i primjeni (dakle minimalan), pouzdan, brz i takav da se njegova konstrukcija može izvesti **egzaktnim postupcima** da bi se mogućnost pogreške svela na najmanju moguću mjeru.
Minimalnost sklopa:
Minimalnost možemo definirati kao minimalan broj (diskretnih) komponenti, minimalan broj integriranih krugova, **minimalan broj logičkih vrata**, minimalna površina štampane pločice, minimalna potrošnja energije.
Zahtjevi na algebarski oblik:
Koristit ćemo kriterij minimalnog broja logičkih vrata i NI i NILI operatore. Uključimo li ostale zahtjeve, algebarski oblik funkcije mora biti napisan tako da bude minimalan, osigura minimalno kašnjenje i jednolikost kašnjenje, omogući primjenu postupaka minimizacije, omogući primjenu NI i NILI vrata. Sve ove kriterije zadovoljavaju minimalni normalni oblici.
Svojstva normalnih oblika:

$$\bigvee_{i=0}^{2^n-1} m_i = 1 \qquad \&_{i=0}^{2^n-1} M_i = 0$$

$$\overline{m}_i = M_i \qquad \overline{M}_i = m_i$$

$$m_i \vee M_i = 1 \qquad m_i M_i = 0$$

$$m_i m_j \underset{i\neq j}{=} 0 \qquad M_i \vee M_j \underset{i\neq j}{=} 1$$

9.2. Osnovni algebarski postupak minimizacije normalnih oblika

– definirati susjednost članova
– definirati osnovni postupak minimizacije
– primjer s komentarom primjene postulata i teorema
– komentirati uštedu i značaj postupka

Susjednost članova: kod osnovnog postupka minimizacije tražimo članove čije su pripadne kodne riječi susjedne(susjedni članovi).
Osnovni postupak minimizacije za PDNO:

$$x_1 \cdot x_2 \cdot x_3 \vee x_1 \cdot x_2 \overline{x_3} \vee .... = x_1 \cdot x_2 \left(x_3 \vee \overline{x_3}\right) \vee .... = x_1 \cdot x_2 \cdot 1 \vee .... = x_1 \cdot x_2 \vee .....$$

Osnovni postupak minimizacije za PKNO:

$$\left(x_1 \vee \overline{x_2} \vee x_3\right) \cdot \left(\overline{x_1} \vee \overline{x_2} \vee x_3\right) \cdot (...) = \left(\overline{x_2} \vee x_3 \vee \left(x_1 \cdot \overline{x_1}\right)\right) \cdot (...) = \left(\overline{x_2} \vee x_3 \vee 0\right) \cdot (.....) =$$
$$= \left(\overline{x_2} \vee x_3\right) \cdot (...)$$

Primjer s komentarom primjene postulata i teorema:
Osnovni postupak minimizacije normalnih oblika provodi se kroz korake:

1. traženje članova čije su pripadne kodne riječi susjedne
2. izdvajanje zajedničkog dijela na osnovi svojstava asocijativnosti
3. izlučivanje zajedničkog dijela na osnovi svojstva distributivnosti
4. u zagradama ostaje oblik $x \vee \bar{x}$ ili $x \& \bar{x}$ baš zbog udruživanja susjednih članova
5. zagrada se reducira u konstantu na osnovi svojstva komplementiranja
6. konstanta se reducira na osnovi svojstva neutralnog elementa

Uštedu i značaj postupka: Ušteda pojedinog ulaza je značajna, jer može rezultirati boljom optimizacijom broja integriranih krugova.

9.3. Pomoćni algebarski postupci (proširenja)

– definirati potrebu za proširenjem
– objasniti proširenje postojećim članom
– objasniti proširenje redundantnim članom
– komentirati uštedu i značaj postupka

Potreba za proširenjem:
Pomoćni postupci minimizacije normalnih oblika zasnivaju se na mogućnosti proširenja algebarskih oblika. Proširenje je moguće dopisivanjem postojećeg člana na osnovi teorema o idepotentnosti ($x \vee x = x$, $x \,\&\, x = x$), te dopisivanjem redundantnog člana za nepotpuno specificirane funkcije. Pomoćni postupak se provodi kroz korake:

1. pronalaženje mogućnosti i potrebe za proširenjem
2. proširenje postojećim ili redundantnim članom
3. provođenje osnovnog postupka minimizacije

Proširenje postojećim članom:
Za PDNO kad izlaz proširujemo postojećim članom, vrijedi:

$$x_1 x_2 x_3 \vee x_1 x_2 \bar{x}_3 \vee x_1 \bar{x}_2 x_3 \vee \ldots =$$
$$\underset{T_2}{=} x_1 x_2 x_3 \vee x_1 x_2 \bar{x}_3 \vee x_1 x_2 x_3 \vee x_1 \bar{x}_2 x_3 \vee \ldots =$$
$$= x_1 x_2 \vee x_1 x_3 \vee \ldots$$

Proširenje redundantnim članom:
Za PDNO kad izlaz proširujemo redundantnim članom, vrijedi:

$$x_1 x_2 x_3 \vee x_1 x_2 \bar{x}_3 \vee x_1 \bar{x}_2 x_3 \vee x_1 \bar{x}_2 \bar{x}_3 \cdot R^{=1} \vee \ldots =$$
$$= x_1 x_2 \vee x_1 \bar{x}_2 \vee \ldots = x_1 (x_2 \vee \bar{x}_2) \vee \ldots = x_1 \cdot 1 \vee \ldots = x_1 \vee \ldots$$

9.4. Postupak minimizacije PKNO

– objasniti motivaciju i probleme
– navesti svojstva negirane funkcije
– definirati proceduru minimizacije
– primjer s komentarom primjene postulata i teorema

Motivacija:
Cilj minimizacije je da konačan sklop bude ekonomičan u proizvodnji i primjeni, pouzdan, brz. Algebarska minimizacija normalnih oblika provodi se transformacija nad algebarskim izrazom. Primjenom postulata i teorema A.L., iz PKNO može se dobiti minimalni konjuktivni normalni oblik MKNO. U praksi, MKNO se dobije transformacijom negirane funkcije.

Navesti svojstva negirane funkcije:
Negirana funkcija je funkcija koja ima vrijednost 1 tamo gdje izvorna funkcija ima vrijednost 0, a 0 tamo gdje izvorna funkcija ima vrijednost 1. Ako je izvorna funkcija nepotpuno specifirana, negirana je nepotpuno specifirana za iste retke tablice istine.

Definirati proceduru minimizacije:
Postupak minimizacije negirane funkcije provodi se kroz korake:
1. traženje članova čije su pripadne kodne riječi susjedne
2. izdvajanje zajedničkog dijela na osnovi svojstava asocijativnosti
3. izlučivanje zajedničkog dijela na osnovi svojstva distributivnosti
4. u zagradama ostaje oblik $x \vee \overline{x}$ ili $x \& \overline{x}$ baš zbog udruživanja susjednih članova
5. zagrada se reducira u konstantu na osnovi svojstva komplementiranja
6. konstanta se reducira na osnovi svojstva neutralnog elementa

Rezultat ovog postupka je MDNO negirane funkcije koju korištenjem DeMorganovih teorema možemo transformirati u MKNO.

Primjer s komentarom primjene postulata i teorema:

$$\begin{aligned}
&(x_1 \vee \overline{x}_2 \vee x_3)\cdot(\overline{x}_1 \vee \overline{x}_2 \vee x_3)\cdot(\ldots) \underset{P_{3a}}{=} \\
&= (\overline{x}_2 \vee x_3 \vee x_1)\cdot(\overline{x}_2 \vee x_3 \vee \overline{x}_1)\cdot(\ldots) \underset{P_{4a}}{=} \\
&= (\overline{x}_2 \vee x_3 \vee (x_1 \cdot \overline{x}_1))\cdot(\ldots) \underset{P_{5b}}{=} \\
&= (\overline{x}_2 \vee x_3 \vee 0)\cdot(\ldots) \underset{P_{2a}}{=} \\
&= (\overline{x}_2 \vee x_3)\cdot(\ldots)
\end{aligned}$$

10. POSTUPCI MINIMIZACIJE I REALIZACIJE NI I NILI VRATIMA

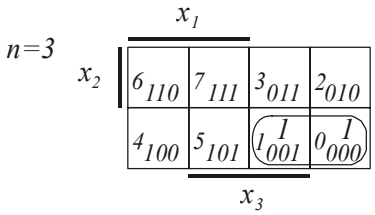
10.1. Postupak minimizacije Veitchevim dijagramom

– pokazati zapis funkcije s pomoću VD
– pokazati osnovni postupak minimizacije na VD
– pokazati pomoćne postupke minimizacije na VD
– pokazati ispis članova
– definirati pravila i postupak minimizacije s pomoću VD

Pokazati zapis funkcije s pomoću VD:
upisujemo funkciju u Veitchev dijagram:

- za PDNO upisujemo 1 i R
- za PKNO upisujemo 0 i R

npr: $\bar{x}_1 \bar{x}_2 \bar{x}_3 \vee \bar{x}_1 \bar{x}_2 x_3 = \bar{x}_1 \bar{x}_2$

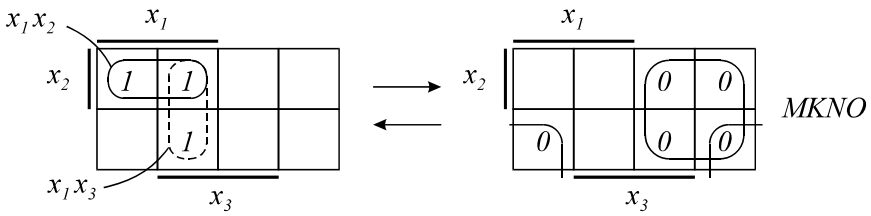


Susjednim mintermima odgovaraju susjedna područja!
Rezultat minimizacije je ekvivalentan ujedinjavanju područja!

Pokazati osnovni postupak minimizacije na VD:
Osnovni postupak minimizacije u postupku VD odgovara objedinjavanju susjednih površina
PRIMJER:

$$f(x) = x_1 x_2 \bar{x}_3 \vee x_1 x_2 x_3 \vee x_1 \bar{x}_2 x_3$$

PDNO: PKNO:

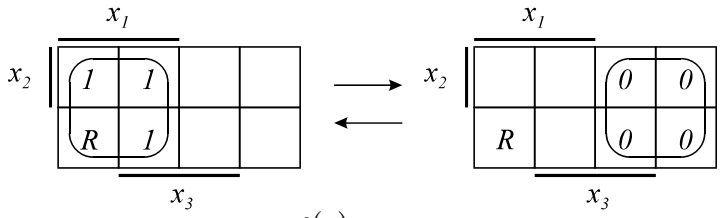


$$f(x) = x_1 x_2 \vee x_1 x_3 \qquad f(x) = x_1 \cdot (x_2 \vee x_3)$$

Pokazati pomoćne postupke minimizacije na VD:
Pomoćni postupak minimizacije dopisivanjem postojećeg člana u postupku VD odgovara višestrukom zaokruživanju pripadnog kvadrata VD. Pomoćni postupak minimizacije dopisivanjem redundantnog člana u postupku VD odgovara zaokruživanju kvadrata VD u kojem je upisana oznaka R. Za MDNO time je definirana vrijednost preslikavanja funkcije u 1, a za MKNO definirana je vrijednost preslikavanja funkcije u 0.
PRIMJER:

$$x_1 x_2 \overline{x}_3 \vee x_1 x_2 x_3 \vee x_1 \overline{x}_2 x_3 \vee x_1 \overline{x}_2 \overline{x}_3 \, R = x_1 x_2 \vee x_1 \overline{x}_2 = x_1$$

PDNO: PKNO:



$$f(x) = x_1$$

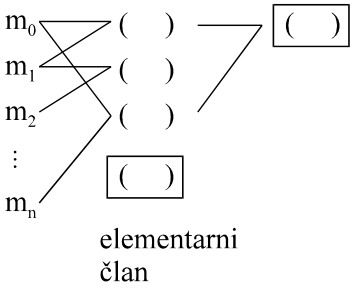
Pravila i postupak minimizacije s pomoću VD:
MINIMIZACIJA PDNO u MDNO:

- u Veitchev dijagram upišemo 1 i R
- zaokružimo **sve** jedinice **što manjim brojem što većih površina**
- ispišemo MDNO na osnovu "koordinata površina"
- **dvostruko zaokruživanje** jedinice znači **proširenje postojećim članom**
- **zaokruživanje redundantnog** člana znači **proširenje** izraza tim članom i time sklop obavlja preslikavanje u 1
- **ne–zaokruživanje redundantnog** člana znači **izostavljanje** tog člana i time sklop obavlja preslikavanje u 0

10.2. Quinn–McClusky postupak minimizacije

– pokazati zapis funkcije s pomoću QMC postupka
– pokazati osnovni postupak minimizacije na QMC
– pokazati pomoćne postupke minimizacije na QMC
– pokazati izbor nužnih elementarnih članova
– definirati pravila i postupak minimizacije s pomoću QMC postupka

Postupak:
ispitujemo susjednost minterma i formiramo elementarne članove:



elementarni član

odaberemo na kraju nužne članove, može za MDNO i MKNO

QMC je tablično–algebarski postupak koji se sastoji od slijedećih koraka:
- sve minterme PDNO ispišemo jedan ispod drugog
- provjeravamo susjednost svih parova od vrha prema dnu
- ako su dva člana susjedna, desno ispisujemo reducirani član
- na isti način provjeravamo susjednost novih članova te ispisujemo eventualne reducirane članove

Postupak je gotov kad više nema susjednih članova.
Postupak je definiran za PDNO, a MKNO dobijemo preko negirane funkcije

10.3. Harvardski postupak minimizacije

– pokazati zapis funkcije s pomoću HV postupka
– pokazati osnovni postupak minimizacije na HV
– pokazati pomoćne postupke minimizacije na HV
– pokazati izbor nužnih elementarnih članova
– definirati pravila i postupak minimizacije s pomoću HV postupka

HV postupak je nastao iz QMC postupka tako da su za određeni broj varijabli unaprijed izračunati svi reducirani članovi. To omogućuje formiranje standardnih tablica pomoću kojih se lako minimizira funkcija.
Postupak je također definiran za PDNO, a MKNO dobijemo preko negirane funkcije.
Značaj HV postupka je u mogućnosti programiranja na računalu.
Postupak minimizacije:
– upisati funkcije(kombinaciju varijabli i vrijednosti funkcije)
– precrtati retke za koje je funkcija jednaka nuli(redundantne NE precrtati)
– precrtati (horizontalnim crtama) brojeve po stupcima koji su precrtani u drugom koraku
– provjeravajući s desna na lijevo, precrtati po recima (kosim crtama) one brojeve, koji s lijeve strane imaju neprecrtan kraći član
– neprecrtane brojeve proglasiti elementarnim članovima
– izabrati nužne elementarne članove tako da u njima budu sadržani svi mintermi za koje je funkcija jednaka 1, a po potrebi i redundantni članovi

10.4. Minimizacija i realizacija NI vratima

– definirati postupak i transformaciju za NI vrata
– primjer s komentarom primjene postulata i teorema

Postupak i transformaciju za NI vrata:
Booleova funkcija realizira se NI vratima u sljedećim koracima:
– polazimo do PDNO originalne funkcije
– izračunamo MDNO originalne funkcije
– dvostruko negiramo MDNO (cijeli izraz)
– primjenom DeMorganovih teorema transformiramo za NI
  – dobijemo izraz za NI vrata zapisan sustavom {&, –}

Primjer s komentarom primjene postulata i teorema:

$$f_1(x) = x_1 x_2 \vee x_2 x_3 \qquad /^{=}$$

$$f_1(x) = \overline{\overline{x_1 x_2 \vee x_2 x_3}} \underset{\text{DeM.T.}}{=} \overline{\overline{x_1 x_2} \cdot \overline{x_2 x_3}}$$

10.5. Minimizacija i realizacija NILI vratima

– definirati postupak i transformaciju za NILI vrata
– primjer s komentarom primjene postulata i teorema

Postupak i transformaciju za NILI vrata:
Booleova funkcija realizira se NILI vratima u sljedećim koracima:
– polazimo do PDNO negirane (inverzne) funkcije
– izračunamo MDNO negirane funkcije
– negiramo obje strane izraza, to je već NILI
– dvostruko negiramo pojedine članove
– primjenom DeMorganovih teorema transformiramo za NILI
Primjer s komentarom primjene postulata i teorema:

$$\bar{f}_2(x) = \bar{x}_2 x_4 \vee \bar{x}_1 \bar{x}_3 \qquad /^{-}$$

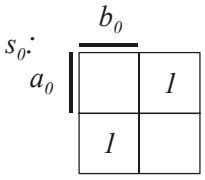
$$\bar{\bar{f}}_2(x) = \overline{\overline{\bar{x}_2 x_4} \vee \overline{\bar{x}_1 \bar{x}_3}} = \overline{\overline{x_2 \vee \bar{x}_4} \vee \overline{x_1 \vee x_3}}$$

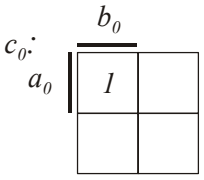
10.6. Sinteza sklopova za zbrajanje

– definirati problem
– zbrajanje na LSB – polusumator
– zbrajanje s pretekom – potpuni sumator
– komentar kašnjenja za n–bitni sumator

Definirati problem:
Problem se javlja kad je minimizacija otežana zbog dijagonalnog položaja jedinica

b_0	a_0	s_0	c_0
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



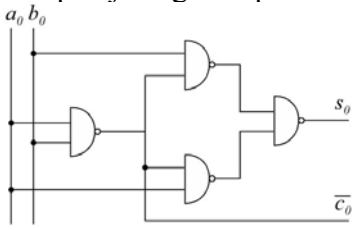


$$s_0 = a_0 \oplus b_0 = \overline{\overline{\bar{b}_0 a_0} \; \overline{b_0 \bar{a}_0}} \qquad c_0 = a_0 b_0 = \overline{\overline{a_0 b_0}}$$

Zbrajanje na LSB – polusumator:
Moguća je sljedeća transformacija:

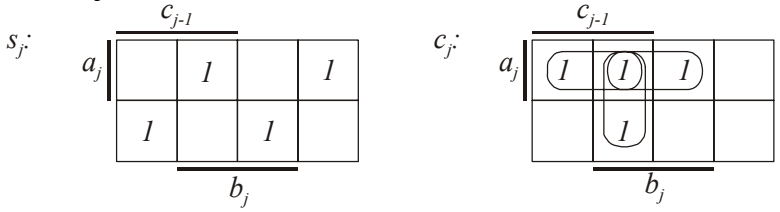
$$s_0 = b_0\,\overline{a}_0 \vee \overline{b}_0 a_0 \vee 0 \vee 0 =$$
$$= b_0\,\overline{a}_0 \vee \overline{b}_0 a_0 \vee b_0\,\overline{b}_0 \vee a_0\,\overline{a}_0 =$$
$$= \overline{\overline{b_0\left(\overline{\overline{a}_0 \vee \overline{b}_0}\right) \vee a_0\left(\overline{\overline{b}_0 \vee \overline{a}_0}\right)}} =$$
$$= \overline{\overline{b_0\left(\overline{a_0\,b_0}\right)} \cdot \overline{a_0\left(\overline{a_0\,b_0}\right)}}$$

Uštedjeli smo ulazne invertore, a sklop daje negirani pretek



Zbrajanje s pretekom – potpuni sumator:

Pokušajmo minimizirati:



Transformiramo:

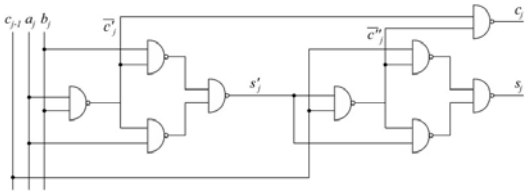
$$s_j = a_j\,\overline{b}_j\,\overline{c}_{j-1} \vee a_j\,b_j\,c_{j-1} \vee \overline{a}_j\,\overline{b}_j c_{j-1} \vee \overline{a}_j\,b_j\,\overline{c}_{j-1}$$
$$s_j = c_{j-1}\left(a_j\,b_j \vee \overline{a}_j\,\overline{b}_j\right) \vee \overline{c}_{j-1}\left(a_j\,\overline{b}_j \vee \overline{a}_j\,b_j\right)$$
$$s_j = c_{j-1}\,\overline{a_j \oplus b_j} \vee \overline{c}_{j-1}\left(a_j \oplus b_j\right)$$
$$s_j = c_{j-1} \oplus \left(a_j \oplus b_j\right)$$

Potrebno je generirati pretek:

$$c_j = a_j\,b_j \vee c_{j-1}\left(a_j\,\overline{b}_j \vee \overline{a}_j\,b_j\right)$$

I konačno:

$$c_j = a_j\,b_j \vee c_{j-1}\left(a_j \oplus b_j\right) = a_j\,b_j \vee c_{j-1}\,s'_j = \overline{\overline{c'_j \vee c''_j}} = \overline{\overline{c}'_j\,\overline{c}''_j}$$



Komentar kašnjenja za n–bitni sumator:
Mana potpunog sumatora je u kašnjenju za 6 t_d i preteka za 5 t_d. Sam pretek kasni prema prethodnom za 2 t_d. Sklop koji bi trebao zbrojiti n znamenki potrošio bi prvo 3 t_d za generiranje svih poluzbrojeva s'_j, a nakon toga bi se pretek prostirao s desna na lijevo s kašnjenjem od 2 t_d po bitu. Zbroj na najznačajnijem bitu kasni još dodatnih 3–2 t_d. Dakle, ukupno kašnjenje je približno izrazu: $T(n)=(4+2n)t_d$

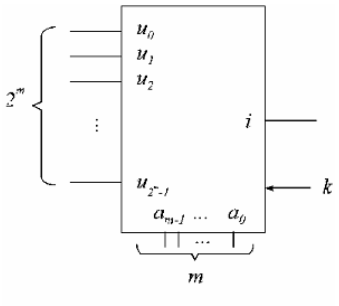
II. DIO - DIGITALNI SUSTAVI I STRUKTURE

II. TREĆINA GRADIVA

11. KOMBINACIJSKI SKLOPOVI SREDNJEG STUPNJA INTEGRACIJE

11.1. Selektor/multiplekser

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



Multiplekser je logički sklop koji na informacijski izlaz **i** propušta vrijednost onog od $n=2^m$ informacijskih ulaza $u_0 \ldots u_{n-1}$, čiji je redni broj prisutan u prirodnom binarnom obliku na m adresnih ulaza $a_0 \ldots a_{m-1}$, pod uvjetom da je f–ja sklopa omogućena aktivnim signalima na kontrolnim ulazima. Algebarski izraz multipleksera: $i = \bigvee_{j=0}^{2^m-1} m_j \,\&\, u_j$.

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 mintermom, koji 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:

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

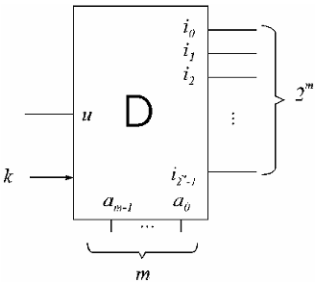
Dvostrukom negacijom dobili smo mogućnost korištenja NI vrata.

Multiplekser 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 **paralelnu–serijsko konverziju**.

II. DIO - DIGITALNI SUSTAVI I STRUKTURE

11.2. Dekoder/demultiplekser

– definirati funkciju dekodera/demultipleksera
– izvesti formulu za dekoder/demultiplekser
– opisati uporabu dekodera/demultipleksera



Demultiplekser je logički sklop koji vrijednost informacijskog ulaza **u** propušta na onaj od n=$2^{\mathbf{m}}$ informacijskih izlaza $\mathbf{i_0}\ldots\mathbf{i_{n-1}}$, čiji je redni broj prisutan u prirodnom binarnom obliku na m adresnih ulaza $\mathbf{a_0}\ldots\mathbf{a_{m-1}}$, pod uvjetom da je f–ja sklopa omogućena aktivnim signalima na kontrolnim ulazima**.**
Algebarski izraz demultipleksera: $i_j=m_j\&u$

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 mintermom,koji jedinicom prepozna pripadnu kodnu riječ pa možemo koristiti **I** vrata za propuštanje vrijednosti s ulaza na izabrani izlaz.

Algebarski se može pisati:

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

$$j = 0...2^m - 1$$

a negacijom izraza dobije se mogućnost korištenja NI vrata:

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

$$j = 0...2^m - 1$$

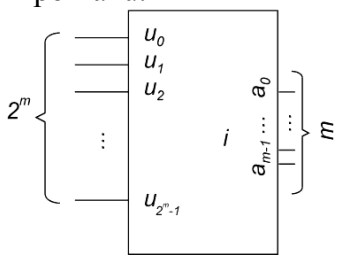
Ako je informacijski ulaz slobodno promjenjiv, a adresa stacionarna, odabrani izlaz će slijediti vrijednost sa ulaza. Obavili smo **razvođenje ulaza** na odabrani izlaz.
Ako se informacijski ulaz mijenja istovremeno (sinkrono) sa adresama, a adrese mijenjamo u prirodnom binarnom nizu, na izlazima će 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.**

II. DIO - DIGITALNI SUSTAVI I STRUKTURE

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 $\mathbf{a_{m-1},a_{m-2},\ldots,a_1,a_0}$ dovodi redni broj **j** u prirodnom binarnom obliku (kao kodnu riječ) onog informacijskog ulaza $\mathbf{u_j}$, 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.

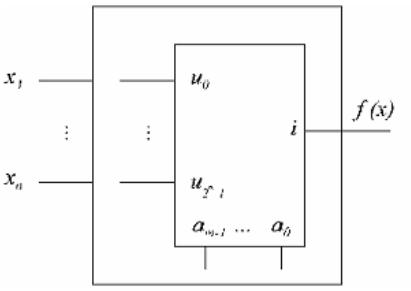
Enkoder prioriteta koristi se uvijek kada je potrebno neki kod tipa 1 od 2^m prevesti u koncentrirani kod od **m** bita. Funkcija mu je suprotna funkciji dekodera. Njime možemo realizirati jednostavnu tastaturu. Masovno se primjenjuje u mikroprocesorima (za razlučivanje različitih prekidnih zahtjeva).

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:



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

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

Uvrštavanjem izraza za multiplekser i PDNO funkcije slijedi:

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

Imamo jednu jednadžbu, a $\mathbf{m} + 2^m$ nepoznanica. Potrebno je spojiti **m** adresnih i 2^m informacijskih ulaza MUX–a za n varijabli funkcija.

II. DIO - DIGITALNI SUSTAVI I STRUKTURE

12.2. Realizacija BF multiplekserom za n=m

– jednadžba 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 jednadžbe realizacije 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 jednadžbe su strukturno identične pa običnu jednakost možemo zamijeniti identitetom te izjednačiti po dijelovima. Dobijemo:

$$u_j = T_j; \qquad j = 0...2^m - 1$$

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

Ako na adresne ulaze dovedemo redom varijable funkcije:

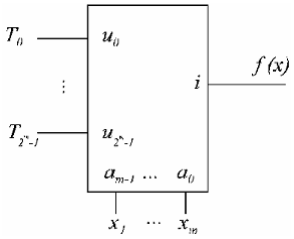
$$\begin{array}{cccccc} a_{m-1} & a_{m-2} & \dots & a_1 & a_0 \\ \uparrow & \uparrow & & \uparrow & \uparrow \\ x_1 & x_2 & \dots & x_{m-1} & x_m \end{array}$$

Odnosno:

$a_e = x_{m-e}; \qquad e = m-1,...,0$

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 ili1). 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:



Komentar rada:Multiplekser neposredno realizira PDNO funkcije

12.3. Realizacija BF multiplekserom za n>m

– jednadžba realizacije za n>m
– algebarsko rješenje za n>m
– konstrukcija sklopa za n>m
– definirati potpuno multiplekersko 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.
Koristimo razbijanje na parcijalne funkcije.

Izraz u zagradi je PDNO preostale funkcije pa pišemo:

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

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

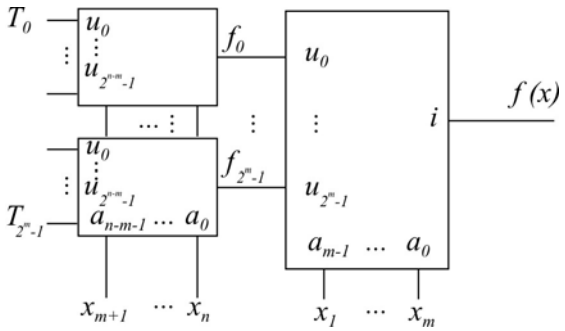
$u_j = f_j(x_{m+1}...x_n)$

$m_j(a) = m_j(x_1...x_m)$

$j = 0...2^m - 1$

Booleovu funkciju **n** varijabli realiziramo multiplekserom s **m** adresnih ulaza tako da na adresne ulaze multipleksera dovedemo **m** varijabli funkcije, 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 **multipleksersko stablo**. Potpuno stablo ekvivalentno je jednom multiplekseru.



12.4. Minimizacija multiplekserskog stabla

– definirati mogućnost minimizacije multiplekserskog stabla
– kriterij minimizacije multiplekserskog stabla
– specijalni slučaj optimalnog sklopa s multiplekserom
– metodologija minimizacije multiplekserskog stabla

Minimizacija multiplekserskog 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** varijabli.

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.

II. DIO - DIGITALNI SUSTAVI I STRUKTURE

13. REALIZACIJA BF DEMULTIPLEKSEROM

13.1. Pristup realizaciji Boolove funkcije demultiplekserom

– osnovni model realizacije
– osnovna jednadžba 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
$i_j = m_j(a) \cdot u = m_j(a) \cdot 1 = m_j(a)$
pri čemu je $j = 0...2^m-1$
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

– jednadžba realizacije za n=m
– specijalno rješenje za n=m
– konstrukcija sklopa za n=m
– problem konstrukcije ILI vrata
– komentar rada na osnovi sheme demuxa

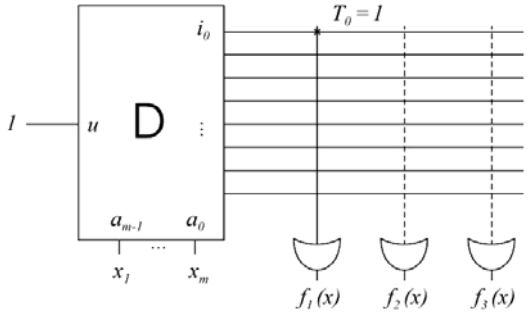
Za n=m vrijedi**:**
Koristimo DEMUX kao dekoder (u=1)
Pa vrijedi:
$i_j = m_j(a) \cdot u = m_j(a) \cdot 1 = m_j(a)$

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

Da bismo minterme funkcije mogli realizirati mintermima adresnih varijabli,
na adresne ulaze demux–a: $a_{m-1}, a_{m-2}, \ldots, a_1, a_0$
dovedemo varijable funkcije: $x_1, x_2, \ldots, x_{m-1}, x_m$ točno ovim redom
pa se dobije:

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

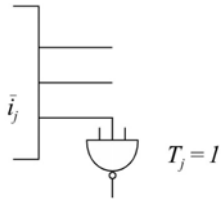
Na ILI vrata spojimo samo one izlaze, za koje je vrijednost funkcije jednaka jedinici (simbolički prikaz):



Mana: realizira PDNO, nema minimizacije
Prednost: realizira više funkcija istih varijabli

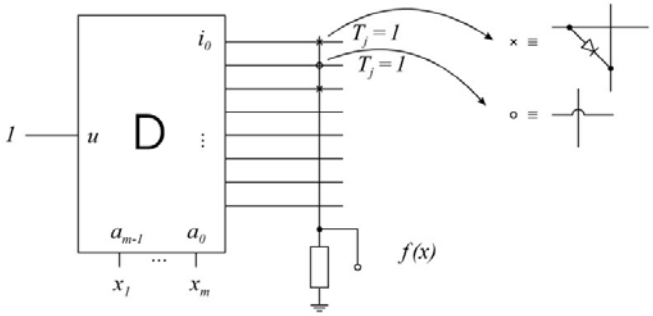
Javlja se problem konstrukcije ILI vrata,pa dvostruko negiramo izraz (i primijenimo DeMorgana) **:**

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



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

– jednadžba realizacije za **n>m**
– algebarsko rješenje za **n>m**
– konstrukcija sklopa za **n>m**
– definirati potpuno demultipleksersko stablo

Za **n>m** pokušamo transformirati PDNO funkcije:

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

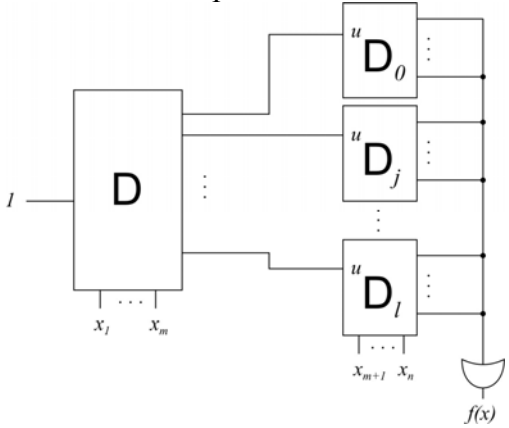
$$\Rightarrow f(x)=\bigvee_{j=0}^{2^m-1} i_j f_j(x)$$

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

Rješenje rezultira nezgrapnim sklopom.
Ako preostalu funkciju realiziramo demultiplekserom možemo koristiti njegov informacijski ulaz koji je konjunktivno vezan sa svim mintermima:

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

Izlaz glavnog demultipleksera dovedemo na ulaz demultipleksera preostale funkcije,čime ga aktiviramo . Dobili smo DEMULTIPLEKSERSKO STABLO! Kada je POTPUNO, stablo je ekvivalentno jednom većem demux–u. Sklop sada ima strukturu:



II. DIO - DIGITALNI SUSTAVI I STRUKTURE

13.4. Minimizacija demultiplekserskog stabla

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

Struktura stabla nam omogućava minimizaciju demultiplekserskog 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.

Metodologija minimizacije demux stabla: Adresne varijable se biraju tako da što veći broj preostalih f–ja bude konstanta.

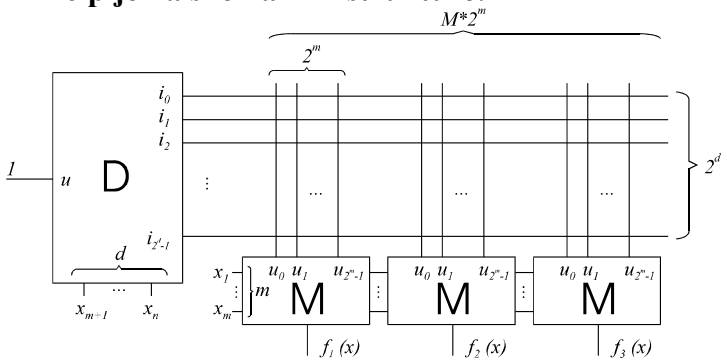
Specijalan slučaj je za n=m+1
Tada su sve preostale funkcije = funkcije jedne varijable.
Na adresne ulaze dvaju demultipleksera dovodimo m varijabli funkcije, a na njihove informacijske ulaze dovodimo izvornu, odnosno negiranu, vrijednost preostale varijable.

14. MULTIPLEKSERSKO–DEMULTIPLEKSERSKA (MD) STRUKTURA

14.1. Multipleksersko–demultiplekserska 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

Obodna vrata: Vrata na rubovima (obodu) MD strukture

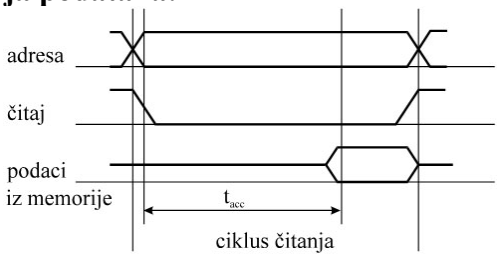
Zbog potrebe da broj logičkih vrata strukture bude minimalan, tražimo da broj redaka R i stupaca S diodne matrice bude minimalan, a minimalnost postižemo približno jednakim brojem stupaca i redaka. Broj logičkih vrata L=R+S

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:
Integracijom MD strukture dobivamo memoriju. 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:
ROM–ovi 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
FEPROM – lebdeća vrata, brisanje električko blok po blok
Vremenski dijagram čitanja podataka:



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

II. DIO - DIGITALNI SUSTAVI I STRUKTURE

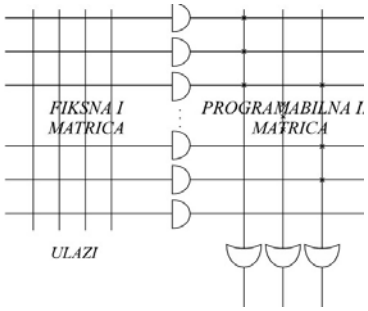
15. PROGRAMABILNE LOGICKE STRUKTURE

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

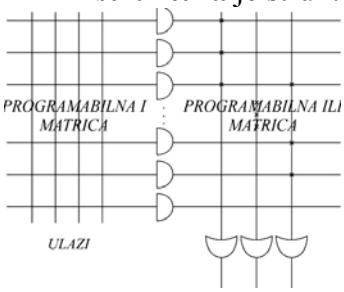


ROM je memorijska struktura kod koje je sadržaj upisan kod izrade. Struktura ROM memorije nije optimalna.
Programibilna ILI matrica samo učinkovito povezuje minterme funkcije u PDNO.
Mintermi su realizirani fiksnom I matricom, što onemogućuje minimizaciju i korištenje MDNO.

15.2. FPLA (Field Programmable Logic Array)

– FPLA struktura
– prednosti i mane FPLA strukture

FPLA struktura je struktura s programibilnim I i ILI matricama.



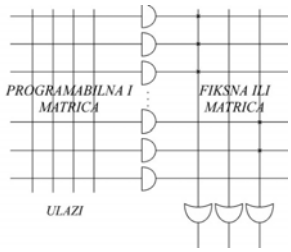
Prednosti: omogućena minimizacija pojedine funkcije i izbor elementarnih članova. Izračunamo MDNO funkcije, el. članove programiramo u I matricu, te ih povežemo u konačni oblik ILI matricom.
Mane: dvije programibilne matrice zahtijevaju dvostruko više pomoćnog sklopovlja, skuplji integrirani krugovi, troši se previše energije.

II. DIO - DIGITALNI SUSTAVI I STRUKTURE

15.3. GAL (Generic Array Logic)

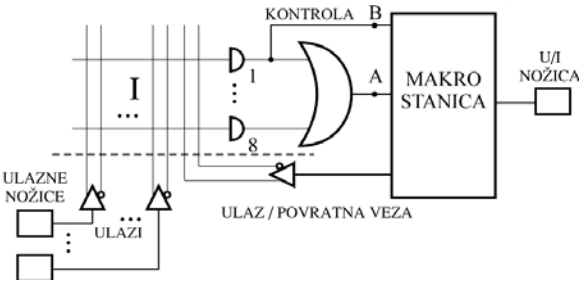
– PAL struktura
– GAL struktura
– koncept makro ćelije

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.

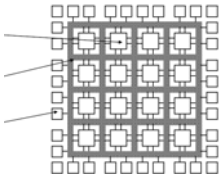
Makro ćelije(stanice) su programibilni izlazni sklopovi korištene kod GAL strukture. Svaka makro stanica može biti konfigurirana na četiri načina kao: sekvencijalni izlaz preko D bistabila, kao kombinacioni ulaz/izlaz, kombinacioni izlaz ili kombinacioni ulaz.

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.

II. DIO - DIGITALNI SUSTAVI I STRUKTURE

16. SEKVENCIJALNI SKLOPOVI

16.1. Kombinacijski sklopovi

– definirati kombinacijski sklop
– obrazložiti potrebe za pamćenjem kombinacijskog sklopa

Kombinacijski sklopovi su sklopovi koji izlaz generiraju isključivo na osnovi trenutnih vrijednosti ulaza. Nemaju memoriju jer ne trebaju pamtiti 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 kombinacijskih, 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. Stanje memorije je ujedno i stanje sekvencijalnog sklopa. Stanje memorije je jednia informacija koju sekvencijalni sklop ima o svim proteklim događajima. Ako dva niza događaja (ulaznih sekvenci) dovedu sklop u isto stanje, sklop ih (u svom budućem radu) više ne može razlikovati. Sekvencijalni sklop mora imati najmanje onoliko stanja, koliko različitih skupina ulaznih sekvenci mora razlikovati da bi obavljao zadanu funkciju.

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)

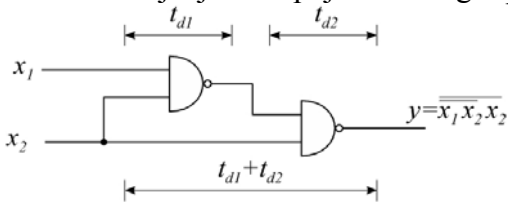
II. DIO - DIGITALNI SUSTAVI I STRUKTURE

17. RAD SKLOPA U DISKRETNOM VREMENU

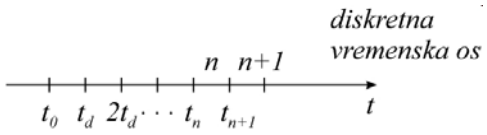
17.1. Diskretno vrijeme

– 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 trenutcima koji su određeni kašnjenjem na pojedinim logičkim vratima.



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

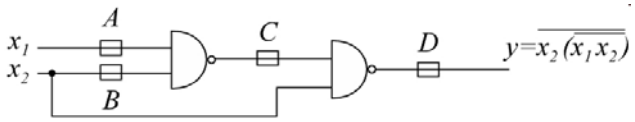


Teorem uzorkovanja:
Diskretno vrijeme definira da moramo pročitati 2B signalnih elemenata u sekundi da bi pročitali informaciju, odnosno predstavlja minimalnu duljinu trajanja signala koja uopće može proći kroz naš sklop, a to je 2B signalna elementa=$1/t_d$ signalna elementa.

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.

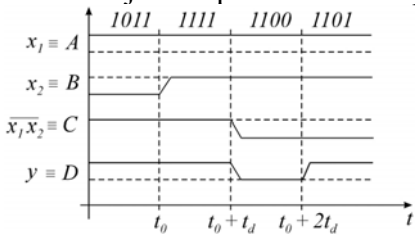
x_1	x_2	A	B	C	D	
1	0	1	0	1	1	
1	1	1	1	1	1	t_0
1	1	1	1	0	0	t_0+t_d
1	1	1	1	0	1	t_0+2t_d

II. DIO - DIGITALNI SUSTAVI I STRUKTURE

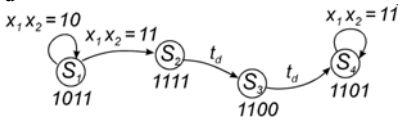
17.3. Sinkroni sklopovi

– 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.



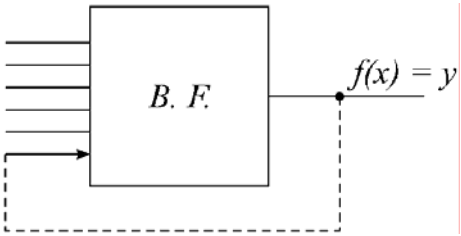
Informaciju želimo pamtiti **proizvoljno vrijeme** , odnosno ne želimo ovisiti o kašnjenju na logičkim vratima, pa trajanje diskretnog perioda povećamo na željenu vrijednost. Da bi se postiglo proizvoljno diskretno vrijeme, potreban je sklop koji može pamtiti informaciju potreban period vremena.

18. BISTABIL KAO SKLOP

18.1. Osnovni sklop za pamćenje – elementarni RS bistabil

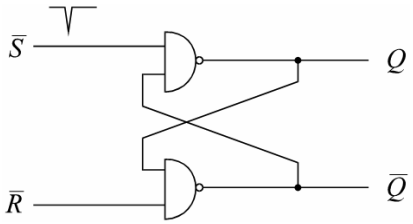
– definirati funkciju bistabila
– povratna veza i njen značaj
– elementarni RS bistabil

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,…, q_n, kao i o njegovom unutrašnjem stanju. Unutrašnje stanje očitavamo na izlazima Q i $\overline{Q}$.



Bistabil je sklop koji je ostvaren preko **povratne veze**, tj. kod njega se vrijednost izlaza dovodi na ulaz. Rad je bitno određen kašnjenjem na sklopu, tako da ce trenutna vrijednost izlaza djelovati nakon vremena kašnjenja sklopa.

RS bistabil je elementarni bistabil i svi drugi bistabili u svojoj osnovici koriste RS bistabil.Možemo ga konstruirati korištenjem NI vrata i NILI vratima.



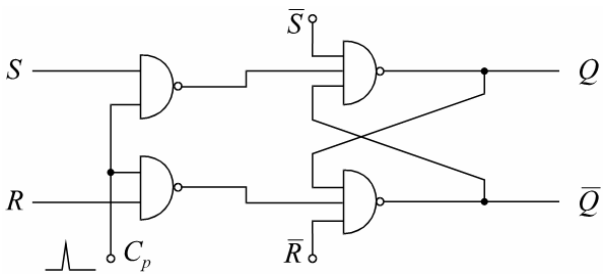
– impulsom 0 na gornjem ulazu upisujemo 1
– to je S (Set, postavi 1)
– impulsom 0 na donjem ulazu upisujemo 0
– to je R (Reset, vrati 0)

II. DIO - DIGITALNI SUSTAVI I STRUKTURE

18.2. Sinkronizacija bistabila s diskretnim vremenom

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

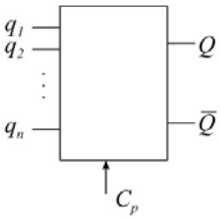
Trenuci u kojima djeluju impulsi su trenuci promjene u diskretnom vremenu. Diskretno vrijeme možemo definirati nekim taktnim signalom, u obliku niza impulsa. Kontroliramo trenutak djelovanja ulaznih impulsa taktnim signalom.



Ulazi R i S djeluju na sklop samo u trenutku pozitivnog taktnog impulsa. To su sinkroni ulazi. S osnovnog bistabila izvedeni su dodatni asinkroni ulazi, $\overline{R}$ i $\overline{S}$. Ovi ulazi služe za postavljanje bistabila u 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,…, q_n i izlaza Q i $\overline{Q}$. Također ima ulaz C_p zbog kojeg u trenutku nastupa taktnog signala, bistabil mijenja stanje na osnovu vlastitog trenutnog stanja i vrijednosti na ulazima.

$(q_1\ q_2\ \ldots\ q_n\ Q)^n$	Q^{n+1}	Q^{n+1}
0 0 … 0 0	0	Q^n
0 0 … 0 1	1	
0	1	$\overline{Q}^n$
⋮ 1	0	
0	0	0
1	0	
1 1 … 1 0	1	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.

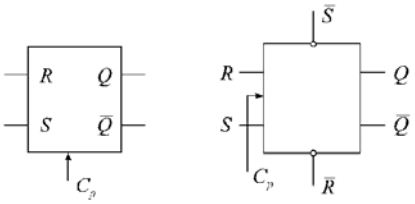
Skraćena tablica prijelaza s lijeve strane ima kodne riječi ulaza q^n poredane prirodnim binarnim nizom, sve u sadšnjem, n–tom trenutku, a s desne strane stanje Q^{n+1}, stanje u slijedećem trenutku, kao funkciju sadašnjeg stanja Q^n.

Bistabil možemo zapisati i funkcijom prijelaza: $Q^{n+1} = f(Q, q_1, \ldots, q_m)^n = (G_1 Q \vee G_2 \overline{Q})^n$, 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.

II. DIO - DIGITALNI SUSTAVI I STRUKTURE

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}
0	0	Q^n
0	1	1
1	0	0
1	1	X

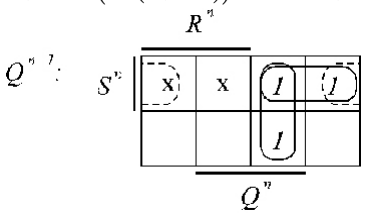
X – neodređeno sljedeće stanje (prijelaz)

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

Funkcija prijelaza:

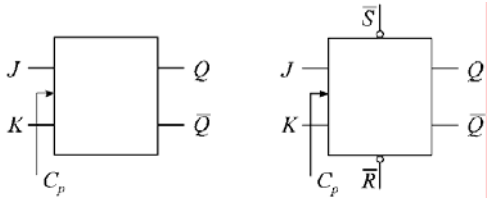
$$Q^{n+1} = \left(\overline{R}\,Q \vee S\,\overline{Q}\right)^n \qquad G_1 = \overline{R} \qquad G_2 = S \qquad Q^{n+1} = \left(\overline{R}\,Q \vee \overline{R}\,S\right)^n$$

$$Q^{n+1} = \left(\overline{R}\,(Q \vee S)\right)^n \qquad Q^{n+1} = \left(\overline{R}\,Q \vee S\right)^n$$



Uvjet RS=0 osiguravamo izvana

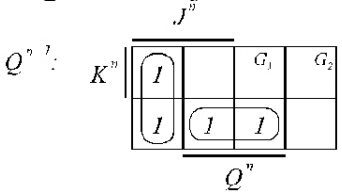
JK bistabil je univerzalni bistabil koji je zadan skraćenom i potpunom tablicom prijelaza a simbol mu je:



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

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

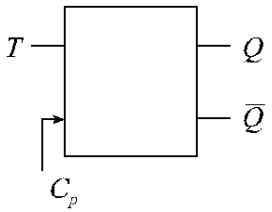
Za JK bistabil kaže se da je upravljiv s ulaza. Jer u skraćenoj tablici prijelaza ima sve četiri moguće funkcije ovisnosti o prethodnom stanju.



$$Q^{n+1} = \left(\overline{K}\,Q \vee J\,\overline{Q}\right)^n$$

$$G_1 = \overline{K} \qquad G_2 = J$$

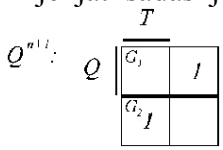
T bistabil mijenja ili zadržava stanje. Zadan je skraćenom i potpunom tablicom a simbol mu je:



T^n	Q^{n+1}
0	Q^n
1	$\overline{Q}^n$

(T	Q)^n	Q^{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

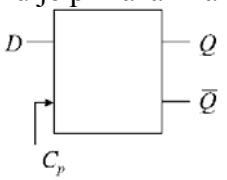
Za T bistabil kaže se da je upravljiv taktom, jer na osnovi ulaza T može samo zadržati ili mijenjati sadašnje stanje. Funkcija prijelaza:



$$Q^{n+1} = \left(\overline{T}\,Q \vee T\,\overline{Q}\right)^n \qquad G_1 = \overline{T} \qquad G_2 = T \qquad G_1 = \overline{G}_2$$

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

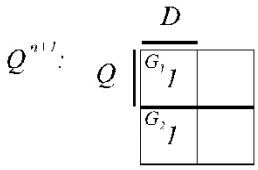
D bistabil pamti 0 ili 1 neposredno s ulaza. Zadan je skraćenom i potpunom tablicom prijelaza a simbol mu je prikazan na slici.



D^n	Q^{n+1}
0	0
1	1

(D	Q)^n	Q^{n+1}
0	0	0
0	1	0
1	0	1
1	1	1

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:



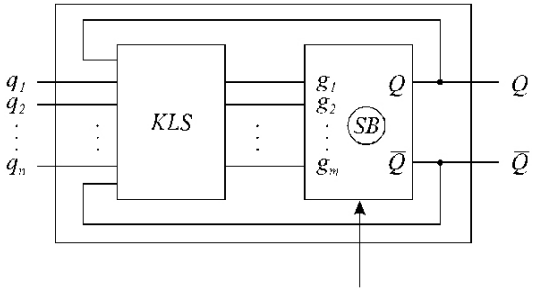
$$Q^{n+1} = \left(D\,Q \vee D\,\overline{Q}\right)^n \qquad Q^{n+1} = D^n$$
$$G_1 = D \qquad G_2 = D$$
$$G_1 = G_2$$

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.

$$Q^n_{OB} = Q^n_{SB} \qquad Q^{n+1}_{OB} = Q^{n+1}_{SB}$$

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

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

Q^n	→	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):

$$Q_{SB}^{n+1} = Q_{OB}^{n+1} \qquad \left(H_1 Q \vee H_2 \overline{Q}\right)^n = \left(G_1 Q \vee G_2 \overline{Q}\right)^n$$

Na osnovi strukturne sličnosti slijedi:

$$H_1 = G_1 \qquad H_2 = G_2$$

Za JK bistabil vrijedi:

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

Odakle slijedi:

$$K = \overline{G_1} \qquad J = G_2$$

U Veitchev dijagram dovoljno upisati vrijednosti Q^{n+1} općeg bistabila, te minimizirati funkcije $\overline{G_1}$ i G_2, tako da ovisno o tome da li raspolažemo s NI ili NILI vratima zaokružujemo nule ili jedinice.

II. DIO - DIGITALNI SUSTAVI I STRUKTURE

19.4. Metoda za D bistabil

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

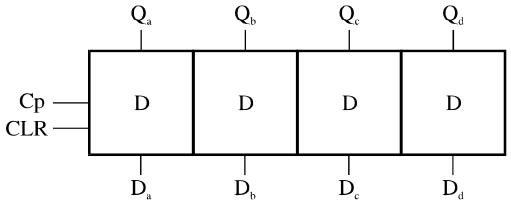
Postupak za D bistabil je istovremeno postupak rekonstrukcije i postupak izjednačavanja. Temelji se na funkciji prijelaza $Q^{n+1} = D^n$, tako da je potpuna tablica općeg bistabila numerički identična tablici istine za KLS. Postupak se provodi tako da prema potrebi minimiziramo funkciju za D^n, ili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

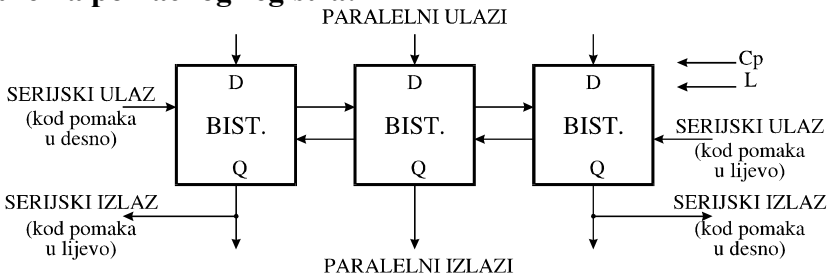


Primjena registra: Služi za pamćenje cjelovitih kodnih riječi.

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 taktni ulaz, a spojeni su tako da je izlaz jednog doveden na ulaz drugog.
Principijelna shema pomačnog registra:



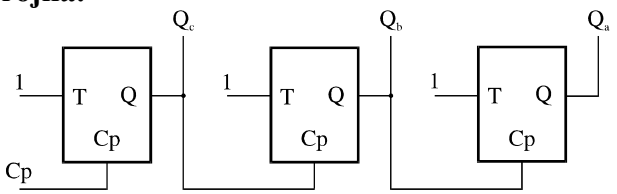
Primjena pomačnog registra: Osim pamćenja kodne riječi, osnovna funkcija je pomak bitova u lijevo ili desno čime ostvarujemo množenje ili dijeljenje, te paralelno – serijska pretvorba.

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.

III. DIO - DIGITALNI SUSTAVI I STRUKTURE

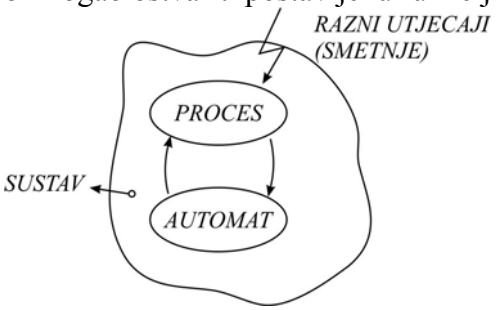
III. TREĆINA GRADIVA

21. DIGITALNI AUTOMAT

21.1. Sustav s upravljanjem

- definirati sustav s upravljanjem
- skica strukture sustava s upravljanjem
- funkcija cilja
- uvjeti uspješnosti sustava

Sustav s upravljanjem je sustav koji čine proces i automat. Na proces djeluje okolina remeteći njegov rad, dok automat pokušava održati proces u optimalnim uvjetima rada, kako bi mogao ostvariti postavljenu funkciju cilja.



Uvjeti uspješnosti sustava:
Automat mjeri stanje procesa i razlučuje sva njegova bitna stanja (mjerljivost) i posjeduje ugrađeno znanje o procesu (poznaje svojstva procesa). Na osnovi sadašnjih i prethodnih događaja u procesu, može se odrediti optimalni niz akcija kojima treba dovesti proces u optimalni režim rada. Da bi to bilo moguće, mora raspolagati i s dovoljnim skupom akcija da bi mogao kompenzirati bilo koji predvidljivi utjecaj okoline. Da bi upravljanje moglo funkcionirati proces mora biti upravljiv.

21.2. Svojstva automata 1. dio

- definirati konačnost
- definirati diskretnost
- definirati digitalnost

Konačnost – ima konačan broj stanja, konačnu memoriju
Diskretnost – radi u diskretnom vremenu
Digitalnost – raspolaže digitalnim ulazima i izlazima

21.3. Svojstva automata 2. dio

- definirati determiniranost
- definirati specificiranost
- definirati sinkronost

Determiniranost – jednoznačno obavlja svoju funkciju
Specificiranost
- potpuno: mogući svi nizovi ulaznih događaja (očekuje proizvoljni niz ulaza)
- nepotpuno: mogući su samo neki nizovi ulaznih događaja

Sinkronost – diskretno vrijeme definirano taktnim signalom

III. DIO - DIGITALNI SUSTAVI I STRUKTURE

22. APSTRAKTNI MODEL DIGITALNOG AUTOMATA

22.1. Automat 1. dio

- automat kao petorka
- definirati skupove U, I i S

Automat kao petorka:

Automat se opisuje petorkom: $A = \langle U, I, S, \delta, \lambda \rangle$ i naziva se apstraktni automat jer je definiran matematičkim elementima, skupovima i funkcijama.

Definirati skupove U, I i S:

U je skup ulaznih simbola ili ulazni alfabet koji su u stvarnom automatu kodirani kodnim riječima ulaznih varijabli **X**:

$$U = \{u_1, u_2, ..., u_p\}$$
$$X = \{x_1, x_2, ..., x_e\}$$
$$2^e \geq p$$

I je skup izlaznih simbola ili izlazni alfabet koji su u stvarnom automatu kodirani kodnim riječima izlaznih varijabli **Y**:

$$I = \{i_1, i_2, ..., i_q\}$$
$$Y = \{y_1, y_2, ..., y_m\}$$
$$2^m \geq q$$

S je skup unutarnjih stanja automata koja su u stvarnom automatu kodirana kodnim riječima varijabli **Z** (a to su kodne riječi stanja bistabila memorije):

$$S = \{s_1, s_2, ..., s_n\}$$
$$Z = \{z_1, z_2, ..., z_k\}$$
$$2^k \geq n$$

22.2. Automat 2. dio

- definirati funkcije δ i λ
- zapisivanje automata

Funkcija prijelaza δ određuje slijedeća stanja automata na osnovi sadašnjeg stanja i sadašnjeg ulaza:

$$\delta: \quad s^{n+1} = \delta(s, u)^n \qquad S \times U \rightarrow S$$

Funkcija izlaza λ određuje sadašnji izlaz automata. Razlikujemo Mealy i Moore model automata:

$$\lambda: \quad i^n = \begin{cases} \lambda(s, u)^n & \text{Mealy} \quad S \times U \rightarrow I \\ \lambda(s)^n & \text{Moore} \quad S \rightarrow I \end{cases}$$

III. DIO - DIGITALNI SUSTAVI I STRUKTURE

Zapisivanje automata:
Automat se zapisuje tablicom prijelaza i izlaza, i usmjerenim grafom.

Tablica prijelaza i izlaza za Mealy automat:

	s^{n+1}	i^n
	$u_1, u_2, \ldots, u_p$	$u_1, u_2, \ldots, u_p$
s_1		
s_2	s_j	i_k
$\vdots$		
s_n		

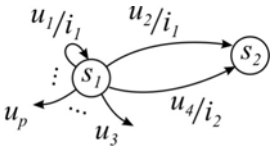
svako mjesto u tablici odgovara jednom paru s, u.

Tablica prijelaza i izlaza za Moore automat:

	s^{n+1}	i^n
	$u_1, u_2, \ldots, u_p$	
s_1		
s_2	s_j	i_k
$\vdots$		
s_n		

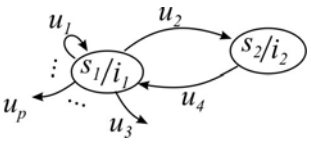
izlazi ovise samo o stanju s.

Usmjereni graf za Mealyev model automata:



Čvorovi su stanja, usmjerene duljine su prijelazi. Uz duljine pišemo izlaze jer ovise o stanju i ulazu.

Usmjereni graf za Mooreov model automata:



Čvorovi su stanja, usmjerene duljine su prijelazi. Uz duljine pišemo izlaze jer ovise samo o stanju.

22.3. Sinteza automata

- definirati faze sinteze
- definirati korake za svaku fazu sinteze

Sinteza automata se obavlja u dvije faze, kroz apstraktnu i strukturnu sintezu. Apstraktna se odnosi na definiranje automata kao matematičkog modela, a strukturna se odnosi na sintezu konkretnog digitalnog sklopa.

Koraci sinteze su:

- apstraktna sinteza
- zadavanje automata
- minimizacija automata

- strukturna sinteza
- kodiranje stanja, ulaza i izlaza
- uvrštavanje kodova, prepoznavanje
  - tablica prijelaza za pojedine bistabile
  - tablica istine za izlazne varijable
- realizacija automata
  - općim bistabilima i logičkim vratima
  - MD strukturom i D bistabilima (MDD)

III. DIO - DIGITALNI SUSTAVI I STRUKTURE

23. ZADAVANJE AUTOMATA

23.1. Pristupi zadavanju automata

- definirati tri transformacije
- navesti konkretne postupke zadavanja

Definirati tri transformacije:

- Zadavanje automata kao **transformatora sekvence** provodi se kroz postavljanje pravila o transformaciji ulazne na izlaznu sekvencu. (matematičke gramatike)
- Zadavanje automata kao **akceptora sekvence** provodi se kroz prepoznavanje sekvenci koje izazivaju pojavu nekog simbola na izlazu. Govorimo o transformaciji ulazne sekvence na izlazni simbol. (jezik regularnih izraza)
- Zadavanje automata postupkom **korak po korak** provodi se kroz analizu svakog mogućeg para stanje-ulazni simbol. Govorimo o transformaciji ulaznog simbola, uz stanje, na izlazni simbol. (metoda potpunog stabla)

Navesti konkretne postupke zadavanja:
Automat možemo zadati na tri načina:

- potpunim stablom
- tablicom prijelaza i izlaza
- regularnim izrazom

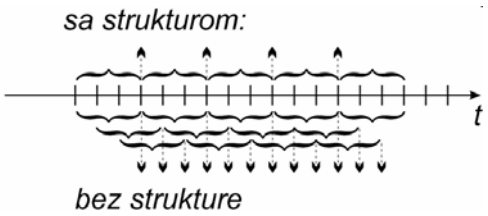
23.2. Vrste ulazne sekvence

- definirati vrste ulazne sekvence
- skicirati vremenski dijagram odlučivanja

Definirati vrste ulazne sekvence:
Sekvenca bez strukture je beskonačna sekvenca ulaznih simbola, kod koje se tražena sekvenca može pojaviti u proizvoljnom trenutku. Nekad su čak moguća preklapanja sekvenci na koje automat reagira, pa moramo odlučiti hoće li preklopljena sekvenca djelovati kao sekvenca koja nije preklopljena. Automat mora u svakom trenutku biti u stanju voditi računa o bilo kojem početnom dijelu sekvence koji se upravo dogodio.

Sekvenca sa strukturom je beskonačna sekvenca ulaznih simbola, koja se sastoji od beskonačnog niza konačnih sekvenci. Tražena sekvenca se ovdje može dogoditi nakon što se dogodi prethodna. Automat mora analizirati ulazni niz sinkrono s pojavom konačnih sekvenci. Konačne sekvence mogu biti iste duljine, ali ne moraju. Ako se koriste sekvence različite duljine, održavanje je moguće samo ako kraća sekvenca nije sadržana na početku niti jedne od dužih sekvenci.

Vremenski dijagram ulaznih sekvenci sa i bez strukture:



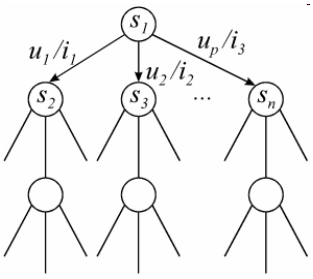
III. DIO - DIGITALNI SUSTAVI I STRUKTURE

23.3. Postupak zadavanja korak po korak

- definirati postupak
- definirati stablo i potpuno stablo
- svojstva automata s grafom potpunog stabla

Definirati postupak:
Zadavanje automata postupkom **korak po korak** provodi se kroz analizu svakog mogućeg para stanje-ulazni simbol. Govorimo o transformaciji ulaznog simbola, uz stanje, na izlazni simbol. (metoda potpunog stabla)

Potpuno stablo je karakterizirano zasebnim prijelazima u nova stanja za svaki par stanja i ulaznog simbola. Karakteristika potpunog stabla je što za svaki niz događaja (za svaku ulaznu sekvencu) generira zasebno stanje.



Svojstva automata s grafom potpunog stabla:
Velik broj sekvenci za automat nema značenje, a neke sekvence imaju isto značenje kao i druge. Stoga je nepotrebno realizirati automat koji će razlučivati svaku ulaznu sekvencu za sebe. Naprotiv, nastoji se koristiti automat koji ima najmanju razlučivost, dovoljnu za izvršavanje funkcije. Takav automat ima najmanji broj stanja.

23.4. Primjena postupka korak po korak

- svojstva stabla za različite modele automata i sekvence
- tretiranje preklapanja sekvence

Svojstva stabla za različite modele automata i sekvence:
MOORE automat, sekvenca SA strukturom: Od početka rada automat analizira konačnu ulaznu sekvencu i donosi odluku u jednom od stanja posljednje razine. Ta stanja se nazivaju akceptorska stanja, jer je sekvenca prepoznata i simbol generiran. Prvi sljedeći simbol je već prvi simbol sljedeće konačne sekvence.
MOORE automat, sekvenca BEZ strukture: Struktura potpunog stabla i odlučivanje automata od početka rada su jednaki kao za sekvencu sa strukturom. Međutim, nakon donošenja odluke automat prelazi u stanja posljednje razine (n+1), zato jer u obzir uzima i prethodnih n-1 simbola, kao da je počeo iz početka.
MEALY automat, sekvenca SA strukturom: Ovaj automat izlaz generira na osnovi stanja i ulaznog simbola (graf ima svega n razina, jednu manje nego Moore).
Nakon što je primljen posljednji simbol donosi se odluka bez prijelaza u novo stanje.
MEALY automat, sekvenca BEZ strukture: Graf je sličan kao i sa strukturom, osim što prijelazi posljednje razine stanja idu u istu, posljednju razinu. Na isti način kontroliramo prijelaze akceptorskih stanja.

Tretiranje preklapanja sekvence:
Posebno se kontroliraju prijelazi iz akceptorskih stanja. Ako želimo spriječiti preklapanje, za ta stanja odredimo prijelaze u stanja s_2 i s_3 kao za sekvencu sa strukturom.

III. DIO - DIGITALNI SUSTAVI I STRUKTURE

24. EKVIVALENTNOST AUTOMATA

24.1. Odnosi jednakosti medu automatima

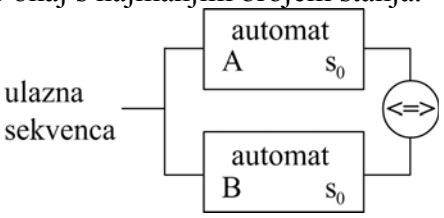
- vrste jednakosti medu automatima
- definicija minimalnosti automata
- objašnjenje razlike broja stanja

Dva automata koji obavljaju istu funkciju (mogu biti različiti po strukturi) su ekvivalentna. Postojanje ekvivalentnog automata objašnjava se sposobnošću razlučivanja ulazne sekvence. Ako dva automata nisu jednaka, a rade istu funkciju, jedan od njih nepotrebno razlučuje ulazne sekvence, iako na kraju za njih donosi istu odluku kao da ih nije razlučio.
Minimizacija se provodi da se pronađe optimalni apstraktni automat, koji će rezultirati minimalnim sekvencijalnim sklopom. Budući da automat s najmanjom razlučivošću ima najmanji **broj stanja**, proglašavamo ga minimalnim. Cilj minimizacije je pronaći minimalni automat ekvivalentan zadanom, odnosno ako je zadani već minimalan, dokazati njegovu minimalnost.

24.2. Definicija ekvivalentnosti automata

- blok shema testa
- definicija ekvivalentnosti automata

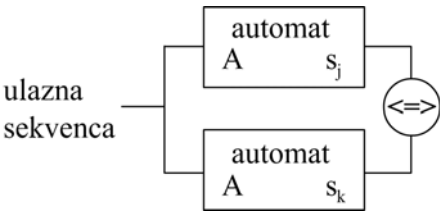
Dva automata (A i B) **su ekvivalentna** ako počnu raditi iz početnog stanja s_0, te za istu **proizvoljnu** sekvencu na ulazu daju **istu** sekvencu na izlazu.
Ekvivalentni automati se ne mogu razlikovati u skupovima ulaznih i izlaznih simbola, jer neće biti zadovoljen zahtjev istovjetnosti ulazne i izlaznih sekvenci. Oni se mogu razlikovati samo u skupu stanja, a minimalan je onaj s najmanjim brojem stanja.



24.3. Definicija ekvivalentnosti stanja

- blok shema testa
- definicija ekvivalentnosti stanja

Stanja koja nepotrebno razlučuju ulazne sekvence, jer za njih automat donosi istu odluku, nazivaju se **ekvivalentna stanja**.
Dva stanja automata A (s_1 i s_2) **su ekvivalentna** ako automat počne raditi iz jednog pa iz drugog stanja, te za istu **proizvoljnu** sekvencu na ulazu da u oba testa **istu** sekvencu na izlazu.



24.4. Nužan i dovoljan uvjet

- definirati nužan uvjet ekvivalencije
- definirati dovoljan uvjet ekvivalencije

Nužan uvjet ekvivalencije kaže da su dva stanja potencijalno ekvivalentna ako su im reci u tablici izlaza automata identični. Ovaj uvjet osigurava da u svakom paru testova prvi simbol izlazne sekvence bude isti.
Dovoljan uvjet ekvivalencije kaže da su dva stanja ekvivalentna ako je zadovoljen nužan uvjet, te ako su im reci u tablici prijelaza automata isti. Ovaj uvjet osigurava da u svakom paru testova, iz promatranih stanja automat prijeđe u isto vrijeme. Time će automat u nastavku testa proći kroz istu trajektoriju stanja, pa je time osigurano da i ostali simboli izlazne sekvence budu isti.

24.5. Minimizacija primitivne tablice

- definirati postupak minimizacije
- definirati minimalizaciju primitivne tablice
- mane jednostavnog postupka minimizacije primitivne tablice

Postupak minimizacije primitivne tablice provodi se nad primitivnom tablicom neposrednom primjenom nužnog i dovoljnog uvjeta ekvivalentnosti stanja. Postupak polazi od pretpostavke da su sva stanja neekvivalentna, te primjenom uvjeta traži moguća ekvivalentna stanja.
Neposredna primjena nužnog i dovoljnog uvjeta ima dvije mane. Prva je u tome što je dovoljan uvjet prestrog, jer ne vodi računa o mogućim neotkrivenim ekvivalentnostima sljedećih stanja. Ovo je moguće djelomično kompenzirati iterativnom primjenom uvjeta ekvivalentnosti. Druga mana proizlazi iz prve, a to je da primjena uvjeta ne može uvijek dati minimalni automat.
Postupak primitivne tablice provodi se u koracima:

1. traže se stanja s istim recima u tablici prijelaza i izlaza;
2. precrtaju se svi redci ekvivalentnih stanja osim jednoga;
3. sve oznake precrtanih stanja zamijenimo oznakom onog neprecrtanog;
4. nastavi se postupak dok ima ekvivalentnih stanja.

25. NAPREDNI POSTUPCI MINIMIZACIJE AUTOMATA

25.1. HM algoritam

- definicija klasa i zatvorenosti
- postupak minimizacije HM algoritmom

Hufmann-Mealy algoritam pretpostavlja da su stanja za koja je zadovoljen **nužan uvjet** ekvivalentna. **Klasa** je podskup stanja iz skupa **S** za koja pretpostavljamo da su ekvivalentna. Klasa je **zatvorena** ako sadrži samo jedno stanje ili ako sadrži više stanja koja sva imaju iste prijelaze u klase.

Postupak minimizacije H-M algoritmom:
1. Definiramo primarne klase na osnovu nužnog uvjeta ekvivalentnosti
2. Za sva stanja unutar klase odredimo prijelaze u klase
3. Kontroliramo zatvorenost klasa (isti prijelazi u klase ili samo jedno stanje)
4. Razbijamo otvorene klase i ponavljamo (2) (prema istim prijelazima u klase)
5. Kada su sve klase zatvorene, svaku zamijenimo s jednim stanjem, te ispišemo tablicu minimalnog automata.

25.2. PU algoritam

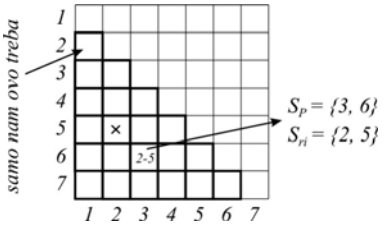
- definicija implikacije
- definicija ekvivalentnosti
- tablica implikanata
- postupak minimizacije tablicom implikanata

Paul-Unger algoritam:
Implikacija među skupovima:
Skup $\mathbf{S_p}$ impliciran je skupom $\mathbf{S_{ri}}$, ako $\mathbf{S_{ri}}$ sadrži sva stanja u koja prelaze stanja iz $\mathbf{S_p}$ za promatrani ulaz $\mathbf{u_i}$.
Pri tom skupova $\mathbf{S_{ri}}$ ima "**p**", koliko ima ulaznih simbola.
Ekvivalentnost: $\mathbf{S_p}$ sadrži ekvivalentna stanja:

- ako je zadovoljen nužan uvjet ekvivalencije
- ako su svi njemu pripadni $\mathbf{S_{ri}}$ ekvivalentni
  (svodi se na zadovoljavanje nužnog uvjeta)

Skupove $\mathbf{S_p}$ formiramo sistematski 2 po 2 stanja (ispitujemo ekvivalentnost svakog sa svakim)
Tablica implikanata je trokutasta matrica bez dijagonale jer vrijedi
$s_i \Leftrightarrow s_i$...svako stanje ekvivalentno samo sebi pa ne trebamo gledat dijagonalu
$s_i \Leftrightarrow s_j \Rightarrow s_j \Leftrightarrow s_i$ ...zbog komutativnosti ekvivalencije ne trebamo cijelu matricu nego samo jednu njenu trokutastu polovicu. Tablica ima **n-1** redaka i stupaca. Svako mjesto odgovara jednom paru s_i, s_j. U mjesta tablice upisujemo implikante, a to su skupovi $\mathbf{S_{ri}}$

Postupak minimizacije provodimo:



1. Formiramo trokutastu matricu (n-1) × (n-1)
2. Upišemo implikante
$S_{ri} \Rightarrow$ zadovoljen nužan, ne i dovoljan uvjet
X $\Rightarrow$ nužan uvjet nije zadovoljen (neekv.)
$\vee \Rightarrow$ zadovoljeni nužan i dovoljan uvjet (ekv.)
3. Ispitujemo kontradikcije, upisujemo X za otkrivene neekvivalentnosti
4. Ponavljamo (3) ako ima novih neekvivalentnosti
5. Ispisujemo tablicu minimalnog automata. Neprekrižena polja znače ekvivalentna stanja.

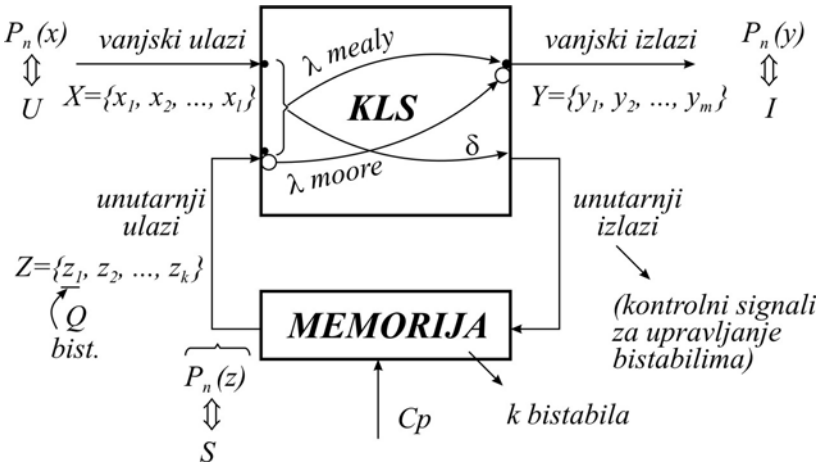
III. DIO - DIGITALNI SUSTAVI I STRUKTURE

26. STRUKTURNA SINTEZA AUTOMATA

26.1. Model realizacije automata

- blok shema modela
- namjena signala
- veza s apstraktnim automatom

Strukturnu sintezu automata vršimo prema **modelu automata** koji se sastoji od kombinacijsko logičke strukture (KLS) i memorije.



Kontrolni signali su unutrašnji izlazi koje KLS generira na osnovu vanjskih i unutrašnjih ulaza, s kojima upravlja prijelazima pojedinih bistabila memorije. Oni određuju sljedeće stanje memorije. KLS na taj način realizira funkciju prijelaza automata δ i funkciju izlaza automata λ.

Veza s **apstraktnim automatom** ostvaruje se kroz kodiranje ulaznih i izlaznih simbola, te kroz kodiranje automata.

26.2. Kodiranje automata

- kodiranje ulaza i izlaza
- problem kodiranja stanja
- strategija kodiranja stanja

Kodiranje ulaza i izlaza ovisi o okolini automata (kompatibilnosti izvorišta i odredištu informacije), pa često zadani kodovi ulaza i izlaza moraju biti usklađeni.

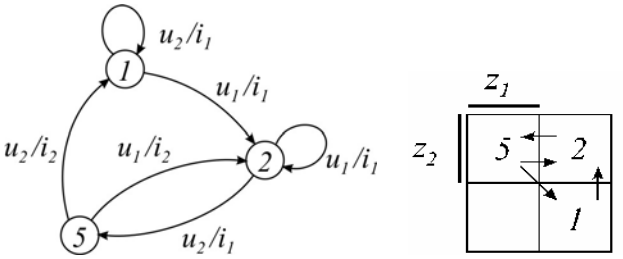
	s^{n+1}		i^n	
	u_1	u_2	u_1	u_2
1	2	1	1	1
2	2	5	1	1
5	2	1	2	2

U	x_1
u_1	0
u_2	1

I	y_1
i_1	0
i_2	1

Problem kodiranja stanja – neposredno utječe na veličinu KLS jer raspored nula i jedinica određuju mogućnost minimizacije Booleovih funkcija koje definiraju KLS. Ne postoji egzaktan postupak kodiranja koji daje minimalan sklop.

Strategija kodiranja stanja – obavlja se po **kriteriju susjednosti.** Stanjima između kojih postoji mogućnost prijelaza dodjeljujemo susjedne kompleksije ili kompleksije sa što manjom distancom. U postupku kodiranja koristimo Veitchev dijagram. Početno stanje obično se kodira kodnom riječi 0.



26.3. Tablica automata s kodovima

- transformiranje tablice
- tablica kao cjelina
- prepoznati dijelovi tablice

Transformiranje tablice

Koristimo jednodimenzionalnu tablicu prijelaza i izlaza apstraktnog automata kao osnovicu za upis kodova, nabrajanjem parova stanje-ulazni simbol.

S	U	s^{n+1}	i^n
s_1	u_1		
	u_2		
	⋮		
	u_p		
s_2	u_1		
	u_2		
	⋮		
	u_p		
⋮			

Nakon upisa kodova (umjesto simbola i stanja) kodne riječi su poslagane u prirodnom binarnom nizu.

s^n				u^n				s^{n+1}				i^n			
z_1	z_2	…	z_k	x_1	x_2	…	x_ℓ	z_1	z_2	…	z_k	y_1	y_2	…	y_m
0	0	…	0	0	0	…	0								
0	0	…	0	0	0	…	1								
		⋮				⋮		0	0	…	0	0	0	…	0
1	1	…	1	1	1	…	0								
1	1	…	1	1	1	…	1								

Ukupna kodna riječ zz...xx čini **lijevu stranu tablice**. **Desnu stranu tablice** prvo čine prijelazi (kodne riječi stanja bistabila u sljedećem diskretnom trenutku), zatim kodne riječi izlaza u sadašnjem trenutku. Svaka kodna riječ stanja sljedećeg trenutka ovisi o kodnoj riječi stanja, te ulaza i izlaza u sadašnjem trenutku. O lijevoj strani ovisi i svaki pojedini bit desne strane.

Prepoznati dijelovi tablice su tablice prijelaza bistabila z i tablice istine izlaznih varijabli y.

	z_1	z_2	x_1	z_1	z_2	y
s_1	0	0	0	0	1	0
			1	0	0	0
s_2	0	1	0	0	1	0
			1	1	1	0
R	1	0	0	R		R
			1			
s_5	1	1	0	0	1	1
			1	0	0	1

III. DIO - DIGITALNI SUSTAVI I STRUKTURE

26.4. Sinteza konkretnog automata

- moguća struktura sklopovlja
- kriteriji sinteze konvencionalnog automata
- kriteriji sinteze MDD strukture

Moguća struktura sklopovlja – integrirani krugovi niske razine integracije(bistabili i logička vrata), integrirani krugovi srednje razine integracije(multiplekseri, demultiplekseri i regristri) i programibilne logičke strukture s ugrađenim D bistabilom u izlaznoj makro ćeliji (GAL).

Sinteza memorije se provodi izborom vrste i izračunavanjem broja bistabila **po kriteriju jednoznačnosti kodiranja stanja automata**, a sinteza KLS se provodi korištenjem prepoznatih dijelova tablice prijelaza i izlaza automata.
Veličina multipleksera i demultipleksera se određuje **po kriteriju kvadratičnosti matrice**.
Broj multipleksera jednak je broju bistabila **k** uvećanom za broj izlaznih varijabli **p**:

$$\mathbf{M = k + p}$$

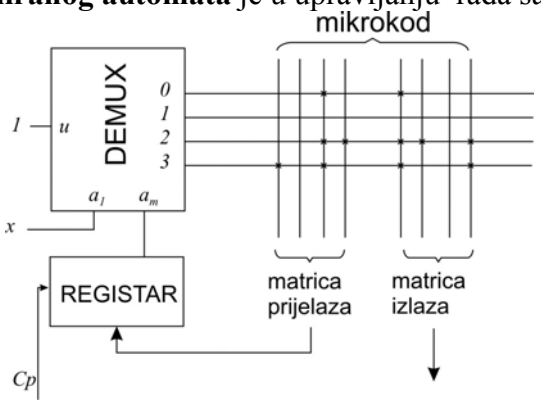
Zbog jednostavnosti postupka i smanjenja mogućnosti pogreške, redoslijed varijabli treba biti usklađen s lijevom stranom kodirane tablice prijelaza i izlaza automata.

27. AUTOMATI I ALGORITMI

27.1. Programabilni automat

- Wilkiesov model automata
- mikroprogramirani automat
- primjena mikroprogramiranog automata

Wilkiesov model automata je model mikroprogramiranog automata realiziranog MDD strukturom.
Mikroprogramirani automat je automat realiziran MDD strukturom. Sadržaj jednog retka matrice naziva se **mikroinstrukcija**, a sadržaj matrice je **mikroprogram.**
Automat aktivira jedan redak matrice i određuje sljedeće stanje (matrica prijelaza) i izlazne signale(matrica izlaza) ovisno o ulazima i stanju**.** Matricom prijelaza može se slijedno izvršavati niz uzastopnih redaka ili skakanje naprijed ili natrag u neki drugi dio matrice.
Primjena mikroprogramiranog automata je u upravljanju rada sabirnice i ALU jedinicom.



III. DIO - DIGITALNI SUSTAVI I STRUKTURE

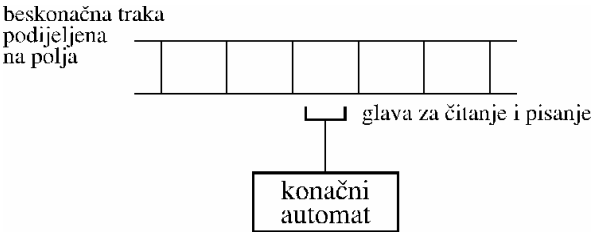
27.2. Algoritam

- definirati algoritam
- primjena algoritma
- istovjetnost algoritma i automata

Algoritam je skup transformacija ili instrukcija čijom se primjenom, u konačnom broju redaka, može doći do rješenja bilo kojeg problema iz promatranog skupa (klase) problema.
Primjena algoritma u rješavanju načina sinteze automata i mogućnost njegove primjene.
Istovjetnost algoritma i automata: Svaki se algoritam može zamisliti kao stroj(automat) koji izvodi niz zadanih instrukcija. Svakom algoritmu pripada stroj, a svakom stroju algoritam.

27.3. Turingov stroj

- definirati Turingov stroj
- definirati jezik petorki
- primjena Turingovog stroja



Turingov stroj je algoritam, odnosno model koji raspolaže beskonačnom memorijom, glavom za čitanje/upisivanje (R/W), mogućnošću pomaka glave lijevo ili desno i konačnim digitalnim automatom koji izvršava program.

Rad Turingovog stroja zadan je petorkom: $\langle S_i, T_i, T_j, \begin{vmatrix} S \\ L \\ R \end{vmatrix}, S_j \rangle$

Prva dva elementa ($\mathbf{S_i}$, $\mathbf{T_i}$) daju sadašnje stanje i simbol glave. Ostali elementi opisuju akciju koja će se izvršiti. $\mathbf{T_j}$ je novi simbol, $\mathbf{S_j}$ je sljedeće stanje automata, dok **L** i **R** specificiraju lijevo odnosno desno kretanje, a **S** zaustavljanje. Nizom ovakvih petorki opisujemo rad Turing-ovog automata. Uočimo da jedna petorka odgovara retku tablice prijelaza i izlaza automata kojem su ulazni i izlazni alfabet identični.
Primjena Turingovog stroja je u pretpostavci mogućnosti realiziranja "bilo kojeg automata"**,** u izračunu svih parcijalno-rekurzivnih funkcija.

III. DIO - DIGITALNI SUSTAVI I STRUKTURE

28. AUTOMATI I JEZICI

28.1. Značaj analize jezika

- ekvivalentnost programibilnih strojeva (algoritama)
- modeliranje procesa postizanja rješenja
- prepoznavanje pripadnosti rijeci jeziku L

Ekvivalentnost programibilnih strojeva (algoritama):
Dva računala (programabilna stroja) su ekvivalentna ako mogu riješiti isti skup (klasu) problema, bez obzira koliko vremena trebalo pojedinačnom računalu.

Modeliranje procesa postizanja rješenja:
Rješenje problema je preslikavanje problema u rezultat.
Problem i rezultat **su opisani** ulaznim i izlaznim nizovima znakova (riječima), stoga se rješenje problema može **modelirati** ispitivanjem da li riječ koja opisuje problem pripada skupu riječi-jeziku L (Language).

Prepoznavanje pripadnosti riječi jeziku L:
Za prepoznavanje pripadnosti riječi jeziku L koristimo strojeve:
– automate određenih sposobnosti (vrste) za konkretan skup (vrstu) jezika
– određene strukture za konkretan jezik.

28.2. Kompleksnost algoritama

- mjera kompleksnosti
- vrste kompleksnosti

Mjera kompleksnosti:
Mjera kompleksnosti je vrijeme postizanja rješenja u smislu izvršenja jednog koraka Turingovog stroja (TM). Pkušavamo izračunati odnos između broja koraka i duljine ulaznog niza znakova, a taj odnos označavamo s O(**g(n)**), gdje je **n** = duljina ulaznog niza.

Vrste kompleksnosti:
Ako postoji rješenje:
– pitanje je da li je ostvarivo u stvarnom vremenu
– raspoloživom tehnologijom
Problem kompleksnosti jednostavno rješavamo modelima na bazi Turingovog stroja.

III. DIO - DIGITALNI SUSTAVI I STRUKTURE

28.3. Izračunljivost

- vrste izračunljivosti
- mogućnost donošenja odluke

Vrste izračunljivosti:

Prihvatljivost rješenja ovisi o prognozi vremena izvršenja:

ovisnost	formula	prognoza
beskonačno	$O(\infty)$	nema algoritamskog rješenja
eksponencijalna:	$O(k^n)$	loša
polinomska	$O(n^k)$	dobro
linearna	O(kn)	vrlo dobro
logaritamska	O(log(n))	izvrsno

Mogućnost donošenja odluke:

Pitamo se:

– da li je niz član jezika L (TR, Turing Recognizable)

– da li niz ne pripada jeziku L (co-TR, Complementary Turing Recognizable)

Nekad u konačnom O() vremenu možemo odgovoriti samo na jedno ili ni na jedno pitanje. Pitamo da li je niz $x \in L$ TR i pritom nije co-TR:

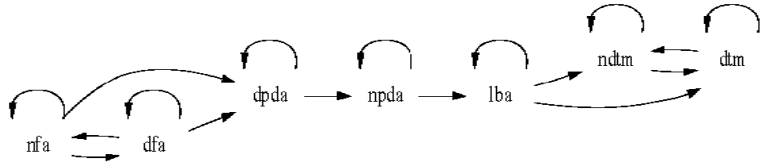
TR	Co-TR	odluka
0	0	ne postoji
0	1	ne postoji
1	0	ne postoji
1	1	postoji, eksponencijalno

III. DIO - DIGITALNI SUSTAVI I STRUKTURE

28.4. Taksonomija automata i jezika

- odnos snage vrsta automata
- ekvivalentnost vrsta automata
- odnos automata i jezika

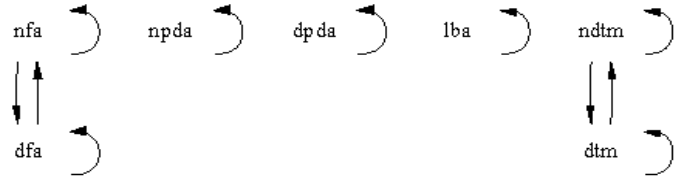
Odnos snage vrsta automata:



DFA (Deterministic Finite Automata)
NFA (Non Deterministic Finite Automata)
DPDA (Deterministic Push Down Automata)
NPDA (Non Deterministic Push Down Automata)
LBA (Linear Bounded Automata)
DTM (Deterministic Turing Machine)
NDTM (Non Deterministic Turing Machine)

Ekvivalentnost vrsta automata:

Ekvivalentnost raznih vrsta automata:



Odnos automata i jezika:

Jezik L je skup sekvenci izgrađenih od članova skupa simbola Σ.
Za neki skup simbola Σ jezika može biti beskonačno, jer karakteristična sekvenca može biti beskonačna. Nad jezicima su definirane operacije: unije, presjeka, razlike, simetrične razlike, pripajanja, eksponenciranja, iteracije i negacije. Ideja o funkciji automata zabilježava se nekim od standardnih jezika. Automat na osnovi ulazne riječi (sekvence slova ulaznog alfabeta) generira izlaznu riječ (sekvenca slova izlaznog alfabeta).
Jezici za pojedine automate:

– DFA, NFA - regularni jezici
– DPDA – determinirani CFL
– NPDA, PDA – CFL itd.

III. DIO - DIGITALNI SUSTAVI I STRUKTURE

29. ALGEBRA DOGAĐAJA

29.1. Elementarni i složeni događaji

- definicija elementarnog događaja
- elementarni događaj i diskretno vrijeme
- složeni događaji
- ulaz automata i algebra događaja

Definicija elementarnog događaja: nedjeljivi događaj, dogodi se kao cjelina.

Elementarni događaj i diskretno vrijeme: dogodi se samo JEDAN u trenutku diskretnog vremena.

Složeni događaji:
- nastaje kao kombinacija elementarnih događaja
- u suštini je vremenski niz - sekvenca - elementarnih

ULAZ AUTOMATA:
- ulazi u automat daju mu informaciju o okolini
- ulazni simboli dolaze u vremenskim nizovima
- to su sekvence u diskretnom vremenu
- te smo sekvence nazivali nizovima DOGAĐAJA

ALGEBRA DOGAĐAJA:
- algebra koja uzima u obzir vremenske odnose
- bavi se nizovima DOGAĐAJA
- uvodi ALGEBARSKE OPERACIJE među događajima

29.2. Operatori algebre događaja

- definicija i svojstva operatora algebre događaja
- definicija regularnog izraza i događaja

Definicija i svojstva operatora algebre događaja:
DISJUNKCIJA: $R = R_1 \vee R_2$
- složeni događaj od dva ili više (elementarnih) događaja
- dogodi se kad se dogodi JEDAN OD disjunktivno vezanih članova (događaja)
- traje koliko traje disjunktivni član koji se dogodio
- vrijedi svojstvo komutativnosti: $R_1 \vee R_2 = R_2 \vee R_1$

KONJUNKCIJA: $R = R_1 \,\&\, R_2$
- složeni događaj od dva ili više (elementarnih) događaja
- dogodi se kad se dogode SVI konjunktivno vezani članovi onim redom kojim su zapisani
- traje koliko traju SVI članovi
- opisuje SEKVENCU događaja
- ne vrijedi svojstvo komutativnosti: $R_1 \,\&\, R_2 \neq R_2 \,\&\, R_1$

NEGACIJA: $R = \overline{R_1}$
- događaj koji se dogodi kad se NE dogodi negirani događaj
- to je dakle skup SVIH događaja OSIM negiranog događaja
- problem je u određivanju POTPUNOG SKUPA događaja (može biti beskonačan)
- stoga se u praksi ne koristi
- definicija regularnog izraza i događaja

III. DIO - DIGITALNI SUSTAVI I STRUKTURE

REGULARNI IZRAZI:
- to su KONAČNI algebarski izrazi u okvirima algebre događaja

REGULARNI DOGAĐAJ:
- svaki složeni događaj koji možemo izraziti konačnim algebarskim izrazom - regularnim izrazom

30. ZADAVANJE AUTOMATA REGULARNIM IZRAZOM

30.1. Zadavanje automata s pomoću RI

- pristup zadavanju automata s RI
- tehnika pisanja RI za sekvencu sa strukturom
- tehnika pisanja RI za sekvencu bez strukture

Ulazni simbol automata je elementarni događaj. Sekvenca ulaznih simbola je složeni događaj
Promatramo automat kao akceptor sekvence. Stoga trebamo opisati traženu sekvencu,
ali i sekvence koje nisu tražene jer ih automat mora odbaciti.
Regularnim izrazom opisujemo ulaznu sekvencu tako da izdvojimo:
- dio sekvence koji nije tražena sekvenca
- dio sekvence koji jest tražena sekvenca

Ako ima više akceptorskih izlaznih simbola pišemo zasebni regularni izraz za svaki simbol.

Tehnika pisanja RI za sekvencu sa strukturom:
– iteracijskom zagradom obuhvatimo sekvence koje NISU tražene
– običnom zagradom obuhvatimo sekvence koje su tražene za promatrani akceptorski (aktivni) izlazni simbol
– pišemo više izraza ako je više akceptorskih izlaznih simbola
– automat u svim diskretnim trenucima, u kojima nije donio odluku, na izlazu daje neutralni (neaktivni) izlazni simbol

Tehnika pisanja RI za sekvencu bez strukture:
– analiziramo dijelove ulazne sekvence
– iteracijskom zagradom obuhvatimo sekvence do duljine tražene koje NISU početak tražene
– upišemo prvi simbol tražene sekvence
– iteracijskom zagradom obuhvatimo ostatke sekvenci nakon prvog simbola koje nisu početak tražene, ali čiji je zadnji simbol prvi simbol tražene sekvence (ako takvih ima)
– upišemo drugi simbol tražene sekvence
– iteracijskom zagradom obuhvatimo ostatke sekvenci nakon drugog simbola koje nisu početak tražene, ali čija su zadnja dva simbola prva dva simbola tražene sekvence (ako takvih ima)
– upišemo treći simbol tražene sekvence
– iteracijskom zagradom obuhvatimo ostatke sekvenci nakon trećeg simbola koje nisu početak tražene, ali čija su zadnja tri simbola prva tri simbola tražene sekvence (ako takvih ima)
– itd. do ispisa tražene sekvence

III. DIO - DIGITALNI SUSTAVI I STRUKTURE

30.2. Indeksiranje RI

- definicija mjesta i vrste mjesta
- analogija mjesta i rada automata
- osnovno indeksiranje
- pravila za rasprostiranje indeksa

Mjesta unutar regularnog izraza mogu biti:
Osnovna - ona koja se nalaze s lijeve strane obične i iteracijske zagrade i s desne strane slova. **Predosnovna** - ona koja se nalaze s desne strane obične i iteracijske zagrade i s lijeve strane slova. Između dva slova (simbola) mjesto je istovremeno osnovno i predosnovno, smatramo ga **osnovnim.** Između dvije zagrade mjesto je istovremeno osnovno i predosnovno, smatramo ga **predosnovnim.**

Analogija mjesta i rada automata:
Osnovna mjesta odgovaraju **stanjima** automata. To su stanja u koja dolazimo nakon određene sekvence ulaznih događaja. **Predosnovna** mjesta odgovaraju **prelazima** automata, a to su stanja iz kojih idemo u prelaze. **Završno** mjesto u RI odgovara akceptorskom stanju digitalnog automata.

Osnovno indeksiranje:
Osnovna mjesta označimo kratkim okomitim crtama i indeksiramo u prvom redu ispod izraza. Predosnovna mjesta označimo dugim okomitim crtama i indeksiramo u drugom redu ispod izraza. Osnovnim mjestima dodijelimo vlastite indekse, npr. redom 1, 2 … Na predosnovna mjesta rasprostiremo indekse osnovnih mjesta koristeći 5 pravila.

Pravila za rasprostiranje indeksa su:

1. indeks mjesta ispred **obične ili iteracijske** zagrade rasprostiremo na početna predosnovna mjesta disjunktivno vezanih članova unutar zagrade (zato jer se zagrada - disjunkcija dogodi, kad se dogodi jedan od disjunktivno vezanih članova)
2. indeks mjesta ispred iteracijske zagrade rasprostiremo na predosnovno mjesto **iza** zagrade(zato jer iteracija sadrži i praznu sekvencu)
3. indeks završnih mjesta disjunktivno vezanih članova unutar **obične ili iteracijske** zagrade rasprostiremo na predosnovno mjesto IZA zagrade (zato jer se zagrada - disjunkcija dogodi, kad se dogodi jedan od disjunktivno vezanih članova)
4. indekse svih mjesta koja smo rasprostirali na predosnovno mjesto iza **iteracijske** zagrade rasprostiremo na početna predosnovna mjesta disjunktivno vezanih članova unutar zagrade (zato jer iteracija dozvoljava proizvoljno ponavljanje)
5. indeks završnog mjesta regularnog izraza rasprostiremo na ona predosnovna mjesta, na koja se je rasprostirao indeks početnog mjesta regularnog izraza. Iznimno, ako za sekvencu bez strukture dozvoljavamo preklapanje, indeks završnog mjesta rasprostiremo na mjesta na koja se rasprostirao indeks mjesta u koje dolazimo nakon dijela sekvence koji se preklapa

30.3. Dobivanje strukture automata iz RI

- razrješenje izlaznog simbola
- razrješenje nastavka rada automata
- redukcija indeksa
- ispis primitivne tablice automata

Razrješenje izlaznog simbola (ovisno o vrsti automata):
Za **MOORE** akceptorski izlazni simbol dodijelimo završnom **osnovnom** mjestu regularnog izraza. Za **MEALY** akceptorski izlazni simbol dodijelimo posljednjem **predosnovnom** mjestu regularnog izraza, a to je uvijek mjesto **ispred** posljednjeg simbola tražene sekvence. Za sva ostala mjesta **podrazumijevamo neutralni** izlazni simbol.

Razrješenje nastavka rada automata:
Indeks završnog mjesta regularnog izraza rasprostiremo na ona predosnovna mjesta, na koja se je rasprostirao indeks početnog mjesta regularnog izraza. Iznimno, ako za sekvencu bez strukture dozvoljavamo preklapanje, indeks završnog mjesta rasprostiremo na mjesta na koja se rasprostirao indeks mjesta u koje dolazimo nakon dijela sekvence koji se preklapa.

Pravila za redukcija indeksa:
Indekse **svih** mjesta koji su se rasprostirali na **isto** predosnovno mjesto zamijenimo jednim indeksom, pod uvjetom da su im dodijeljeni **isti** izlazni simboli. Indekse **svih** mjesta u koja dolazimo iz **istog** predosnovnog mjesta s **istim** ulaznim simbolom zamijenimo jednim indeksom.

Ispis primitivne tablice automata:
Primitivnu tablicu ispišemo tako da preostale indekse osnovnih mjesta uzmemo za stanja automata, a njihove prelaze odredimo preko ulaznih simbola.