Fakultet elektrotehnike, strojarstva i brodogradnje

DIGITALNA ELEKTRONIKA
DIGITALNI SUSTAVI I STRUKTURE
DISKRETNI SUSTAVI I STRUKTURE

-

odgovori na teoretska pitanja

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

Preradio i nadopunio:
Vrvilo Mijo

Akademska godina 2010./2011.

SADRŽAJ

NAPOMENA: - Obratiti pozornost na pitanje 13.4., naime, profesor je rekao da nedostaje izvod pa pogledati u knjigu. Taj izvod postoji, (str. 108, Digitalna i mikroprocesorska tehnika, Julije Ožegović), ali je to izvod za demultiplekser sa m=1, a to se ne pita u natuknicama pa ga nisam pisao.

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 koja postoji bez obzira da li je osjećamo.

Informacija: -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 sa promjenom informacije. Postupke modulacije dijelimo na digitalne i analogne.

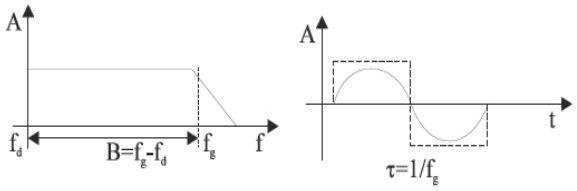
Analogni sustavi: -kod analognih sustava informaciji se pridružuje neka veličina električnog signala prema izabranoj funkciji. To znači da će sva informacija biti sadržana npr. u naponu električnog signala.

Digitalni sustavi: -kod digitalnih sustava informacija se najprije predstavi brojem, a onda taj broj električnim signalom. Za svaku znamenku broja koristi se po jedan električni signal.

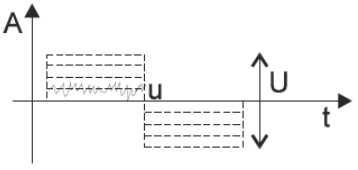
1.2. Informacijski volumen i digitalni sustav:

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

Sustav s niskim propustom: -kod tog sustava širina pojasa je $\mathbf{B=f_g-f_d= f_g-0= f_g}$. U jednom periodu signala f_g prenesemo dva signalna elementa.
Odatle **2B** signalnih elemenata u sekundi.



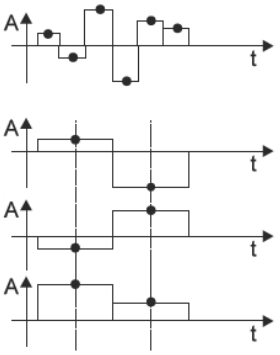
Broj razina: R=U/u. Raspon signala ograničen je dogovorom, a minimalni signal je ograničen smetnjama.
Dogovor: $D=\log_2(R)=ld(R)$ [bita/sign. elementu]



Kapacitet sustava: -izražava brzinu obrade, brzinu prijenosa, brzinu pristupa podacima. Kapacitet $\mathbf{C=2B\cdot D}$ [se/sek·bit/se=bit/sek].

Analogni sustavi: -kod analognih sustava informaciji se pridružuje neka veličina električnog signala prema izabranoj funkciji. To znači da će sva informacija biti sadržana npr. u naponu električnog signala.

Digitalni sustavi: -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, a serijski prijenos je prijenos informacija pomoću jednog kanala.
Istu količinu podataka možemo prenijeti serijski jednim kanalom **k** puta većom brzinom ili paralelno sa **k** kanala jediničnom brzinom.

Informacijski volumen: *V=2BDTK*
Gdje je: 2B – dvostruka širina pojasa, D – dinamika (broj bita po sign. elementu), T – period u kojem je sustav raspoloživ i K – broj paralelnih informacijskih sustava.

1.3. Kodovi i kodiranje:

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

Kôd: -dogovorno uspostavljen sustav simbola kojima označavamo pojmove (informacije) iz nekog skupa pojmova.

Kodiranje ima dva značenja: postupak konstrukcije nekog kôda, ili primjena nekog kôda kroz zamjenu pojma simbolom.

Jednoznačnost: -jednom pojmu najmanje jedan jedinstveni simbol.

Razlučivost: -treba dovoljan broj simbola

Dekodiranje: -postupak izdvajanja polazne informacije iz simbola.

Podjela kodova: -posredni i neposredni

Kodna riječ ili kompleksija: -kodnu riječ formiramo iz skupa elementarnih simbola. *Kompleksija:* cjelina sastavljena od više dijelova.

Analogni i digitalni sustav 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

Definicija poliadskog brojevnog sustava: -broj zapisujemo kompleksijom od **n** elemenata $\mathbf{a_k}$, pomnoženih s potencijom baze **s**:

$$a = \sum_{k=0}^{n-1} a_k s^k = a_{n-1}s^{n-1} + a_{n-2}s^{n-2} + ... + a_1 s + a_0$$

Svojstva: -znamenke $\mathbf{a_k}$ se samo pišu jedna iz druge, tako da je krajnja desna znamenka $\mathbf{a_0}$. Svaka pozicija znamenke na lijevo od $\mathbf{a_0}$ podrazumijeva množenje s potencijom baze **s**. Svaka pozicija ima svoju težinu, te je kôd težinski.

Zapis realnih brojeva: -zapisuju se korištenjem zareza ili točke, kojima se označava pozicija znamenke $\mathbf{a_0}$, iza koje se može dopisati potreban broj znamenki:

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

Odnos i pretvorba 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: $\frac{0001}{1}\frac{1101}{D}$ $1D_{16}=29_{10}$

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 sa manje znamenaka. Za dovoljno veliki **s**, biti će dovoljna jedna znamenka, a to nije ništa drugo nego analogni sustav s jednim električnim signalom. (Tako za brojevne sustave sa manjom bazom treba više znamenki za izraziti neki broj).
Digitalni sustavi su sustavi posrednog kodiranja. Kod njih vrijednost digitalnog signala ima značenje slova, a skup vrijednosti više signala značenje kodne riječi. Kodnom riječi predstavljamo informaciju.

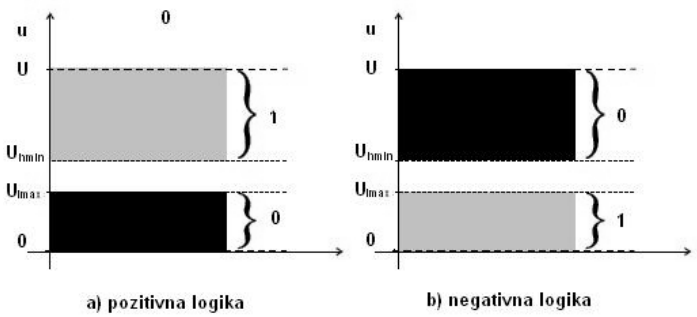
2.2. Izbor brojevnog sustava za digitalne sustave:

-električni signal i dopuštena pogreška
-pozitivna i negativna logika
-rad tranzistora kao sklopke
-kodne riječi binarnog sustava

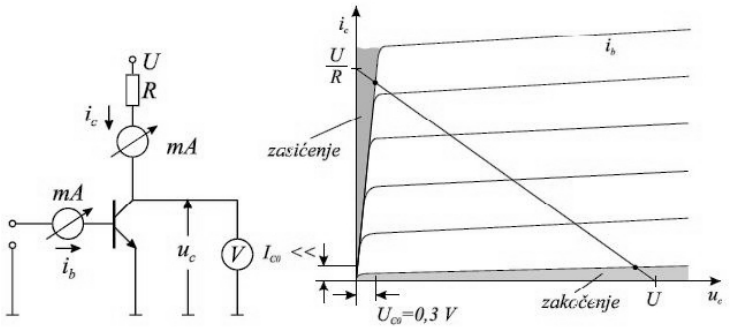
Električni signal: -prijenos informacija električnom energijom.

Dopuštena pogreška: -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 u 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.



Rad tranzistora kao sklopke: -ako je struja baze zanemariva kažemo da je tranzistor zakočen pa će izlaz biti spojen na izvor. Dakle na izlazu će biti logička 1. Ako je struja kolektora dovoljno velika s obzirom na faktor istosmjernog pojačanja β i struju baze kažemo da je tranzistor u zasićenju (uvjet zasićenja: $I_C > \beta * I_B$) 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 kodnih riječi je $N=2^n$, a najveći broj koji se može zapisati pomoću **n** znamenki je $a_{max}=2^n-1$.

2.3. Prikaz brojeva binarnim kodovima:

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

Prirodni binarni kôd: -kodne riječi binarnih brojeva koristimo za kodiranje $\mathbb{N}_0$. Dakle, svaka kodna riječ predstavlja broj napisan u poliadskom brojevnom sustavu.

Pretvorba: -pretvorba dekadskog u binarni brojevni sustav se vrši algoritmom sukcesivnog dijeljenja.
Pretvorba binarnog u dekadski se vrši algoritmom sukcesivnog množenja ili $a_{10} = \sum_{k=0}^{n-10} a_k 2^k$.

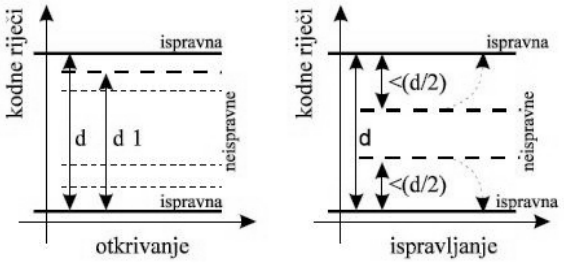
Binarno kodirani dekadski brojevi (BCD): -svaka znamenka dekadskog broja zasebno je kodirana binarnom kodnom riječi. Za prikaz jedne dekadske znamenke potrebna je kodna riječ duljine 4 bita. BCD kodovi mogu biti komplementarni, težinski ili oboje. 8241BCD kôd je težinski, a 2421BCD je komplementarno-težinski.

2.4. Primjene binarnih kodova:

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

Kodna udaljenost ili distanca: -to 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. Novonastala 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 neiskorištenu (neispravnu). Višestruke pogreške su manje vjerojatne. Kôd konstruiramo tako da između bilo koje dvije ispravne kodne riječi distanca bude najmanje „**d**“. Tada možemo **otkriti** (d-1) struku pogrešku i **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 ulijevo i pomak udesno.

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 $\oplus$:

$F=\{0,1,2,\ldots,m-1\}$; $m\geq 2$; $m\in\mathbb{N}$; $a \oplus b=c=\mathrm{Rez}\left(\frac{a+b}{m}\right)$ gdje je **m** modul, a operacija $\oplus$ zbrajanje po modulu **m** definirana kao ostatak (Rez) cjelobrojnog dijeljenja zbroja **a+b** s modulom **m**.

Svojstva (postulati):

-*zatvorenost:* $\forall a, b \in F: a \oplus b = c : c \in F$
-*asocijativnost:* $\forall a, b, c \in F = (a \oplus b) \oplus c = a \oplus (b \oplus c)$
-*komutativnost:* $\forall a, b \in F\ a \oplus b = b \oplus a$
-*neutralni element:* $\forall\ a \in F \quad \exists\ e \in F: a \oplus e = e \oplus a = a$
-*inverz:* $\forall a \in F \quad \exists\ a' \in F: a \oplus a' = a' \oplus a = e$

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: $\forall\ a \in F \quad \exists\ e \in F: a \oplus e = e \oplus a = a$
Problem neodređenosti: -iz običnog zbroja jednoznačno dobijemo ostatak, ali iz ostatka ne možemo znati koliki je bio obični zbroj, pa trebamo uvrstit faktore nesigurnost **k'** i **k''**: $a + e + k'\cdot m \Rightarrow e=(k''-k')\cdot m \Rightarrow e=k\cdot m$

Faktore nesigurnosti k' i k'' možemo zamijeniti jednim faktorom k=k''-k', kojeg biramo kako bi '**e**' zadovoljio svojstvo zatvorenosti: $k=0 \Rightarrow e=0 \in F$

Izračun neutralnog elementa:

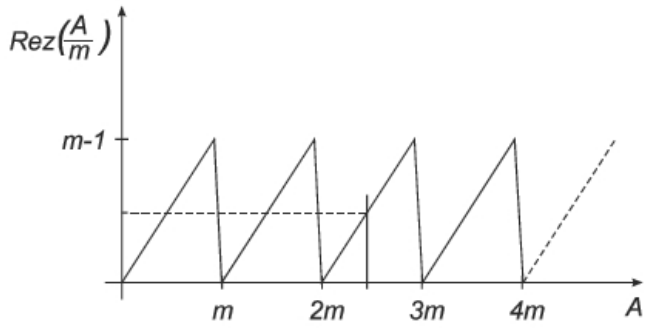
$$a \oplus e = a \Rightarrow \mathrm{Re}\,z\left(\frac{a+e}{m}\right) = \mathrm{Re}\,z\left(\frac{a}{m}\right) \Rightarrow a+e=a \Rightarrow e=0$$

Inverz: $\forall a \in F \quad \exists\ a' \in F: a \oplus a' = a' \oplus a = e$

Inverz (dvojni komplement ili drugi komplement ili komplement po modulu 2) 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: $\begin{matrix} a = 0 \Rightarrow k = 0 \Rightarrow a' = 0 \\ a > 0 \Rightarrow k = 1 \Rightarrow a' = m - a \end{matrix}$ Pa možemo pisati: $a' = \begin{Bmatrix} 0; & a = 0 \\ m - a; & a > 0 \end{Bmatrix}$



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:

$$\begin{array}{l} 1001 \equiv 9 \\ \underline{+1100 \equiv 12} \\ 1\hookleftarrow 0101 \equiv 5 \end{array} \rightarrow 5 = 21_{\mathrm{mod}16} = rez\left(\frac{21}{16}\right) = 5$$

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:
$F=\{0,...,m-1\}=\{0,...,\ 2^n-1\}=\{0,\ldots,\ a_{max}\}$
Povezali smo binarni brojevni sustav i sumu po modulu!

Izračun inverza za binarni sustav:
$a'=m-a$ $m=2^n$ $a_{max}=2^n-1 \rightarrow 2^n=m=a_{max}+1 \rightarrow a'=a_{max}+1-a=a_{max}-a+1$

$$a' = \sum_{k=0}^{n-1} \overline{a_k} 2^k + 1 \rightarrow a' = \begin{Bmatrix} a = 0; & a' = 0 \\ a > 0; & a' = \overline{a} + 1 \end{Bmatrix} = \overline{a} \oplus 1$$

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 (drugi komplement ili komplement po modulu 2) jednak je običnom komplementu uvećanom za jedan $a' = \overline{a} + 1$

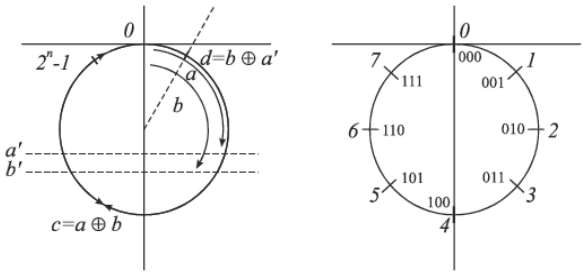
3.4. Primjena drugog komplementa:

-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

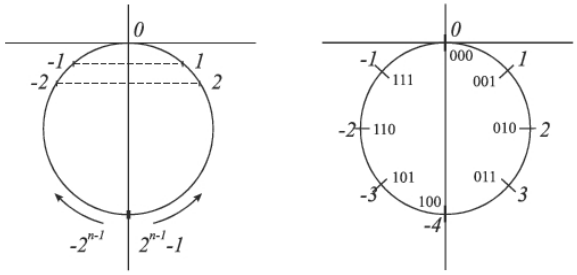
Oduzimanje po modulu:

$$a \Theta b = \mathrm{Re}z\left(\frac{a-b}{m}\right) = \mathrm{Re}\, z\left(\frac{a-b+km}{m}\right)^{k=1} = \mathrm{Re}\, z\left(\frac{a+m-b}{m}\right) = \mathrm{Re}\, z\left(\frac{a+b'}{m}\right) = a \oplus b'$$

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 i rezultat nije točan, tj. ako prijeđemo točku (0) tada ćemo dobiti rezultat koji je manji od oba pribrojnika. Npr. kada bi (gledajući po slici zbrajali) i zbrojili 5+4 dobili bi rezultat 1 jer prelazimo nulu, a to se ne smije desiti.
Razliku d=b-a možemo dobiti oduzimanjem lukova. U praksi, razliku '**d**' računamo korištenjem inverza '**a**' i zbrajanjem s '**b**': d=b+a'. Kod oduzimanja, tj. pribrajanja inverza, donja točka (0) se mora prijeći. Npr. (gledajući po slici), 6-5=6+3=1.



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 kazaljke na satu smatraju se negativnim brojevima koji slijede niz -1, -2,…,-2^{n-1}. Svim brojevima najznačajniji je bit predznaka (npr. 1 za negativne). Pri zbrajanju i oduzimanju sada vrijede ista pravila kao kad se brojevi koriste kao samo pozitivni.



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 sklopovi se izrađuju korištenjem gotovih elementarnih logičkih sklopova (modula). Iz tog razloga se proizvode elementarni logički sklopovi čijim se spajanjem dobije željeni digitalni sklop.

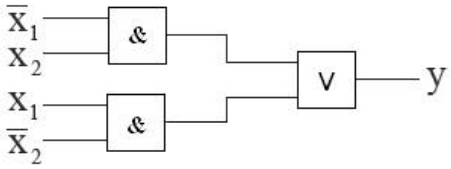
Operatori algebre logike: konjunkcija (&), disjunkcija (v) i negacija ($\bar{}$).

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

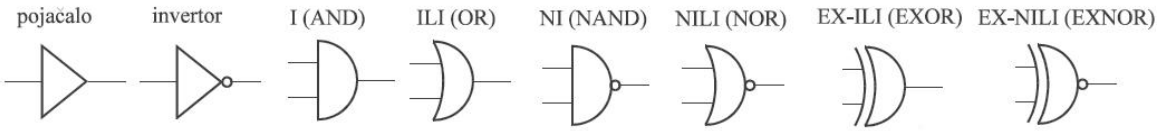
Algebarski izrazi predstavljaju funkcijsku ovisnost o nezavisnim varijablama izraza. Na osnovi algebarskog izraza crtamo logički dijagram, te shemu sklopa.

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

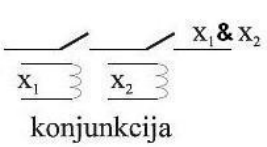
$$y = \overline{x_1} \cdot x_2 \vee x_1 \cdot \overline{x_2}$$

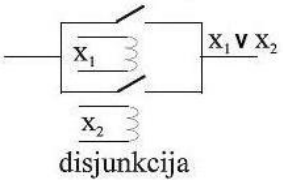


Elementarni logički sklopovi (logička vrata):



Elektromehanički logički krugovi: -operatore konjunkcije i disjunkcije možemo realizirati elektromehaničkim relejima. Relej je uređaj kod kojeg prolaskom struje kroz svitak nastaje magnetsko polje, a ono privuče kotvu na kojoj 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 paralelni spojem radnih kontakata. Negacija se ostvaruje mirnim kontaktom.
Logičke krugove možemo dobiti kombinacijom ovih triju osnovnih spojeva releja.





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, emiterski spregnuta, PMOS, NMOS i komplementarna unipolarna tehnologija.

Stupnjevi integracije:
SSI (small scale integration): -niski stupanj integracije,
do 10 logičkih vrata, do 100 tranzistora.

MSI (medium scale integration): -srednji stupanj integracije,
do 100 logičkih vrata, do 1.000 tranzistora.

LSI (large scale integration): -visoki stupanj integracije,
do 1.000 logičkih vrata, do 10.000 tranzistora.

VLSI (very large scale integration): -vrlo visoki stupanj integracije,
više od 1.000 logičkih vrata, preko 10.000 tranzistora.

Vrste izlaza logičkih vrata:
Bipolarni izlazi (TP – totem pole): -generiraju vrijednosti 0 i 1. U logičkoj 1 služe kao izvorišta, a u 0 kao potrošači.

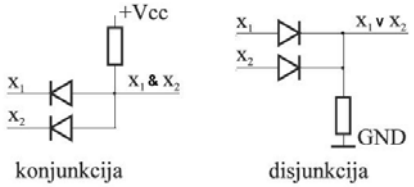
Unipolarni izlazi (OC – open collector): -generiraju samo logičku 0 i mogu biti samo potrošači struje. Kod logičke 1 tranzistor se isključuje. Unipolarni izlazi se koriste kada je potrebno spojiti više izlaza na zajedničku točku.

Tri-state izlazi (TS – bipolarni sa trećim stanjem visoke impedancije): -imaju karakteristike bipolarnih, a sposobnost isključenja omogućava spajanje na sabirnicu.

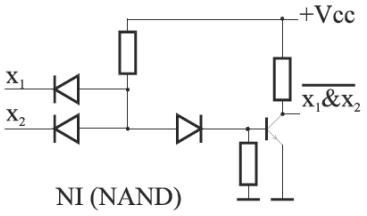
4.3. Diodna i diodno-tranzistorska logika:

-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, stoga se kombinira s pojačalima, bilo relejnim ili tranzistorskim.

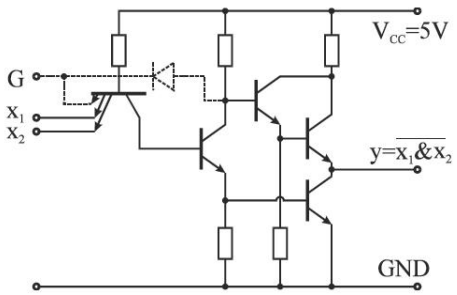


Karakteristike i osnovni sklopovi DTL: -objedinjuje dobra svojstva diodne i tranzistorske tehnike. To je ujedno i prva integrirana tehnika. Jedna od njenih prednosti je u mogućnosti dodavanja proizvoljnog broja ulaza jednostavnim dodavanjem dioda na središnju ulaznu točku.



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

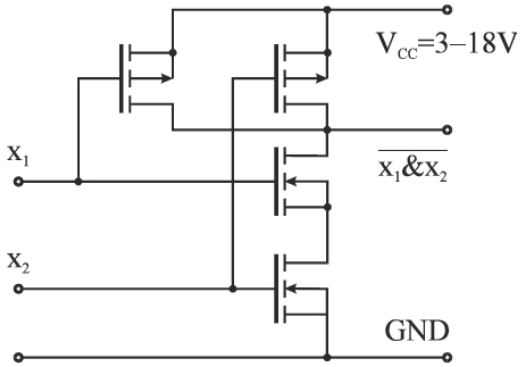
Određivanje razina 0 i 1: -TTL tehnologija je postavila niz standarda u digitalnoj elektronici, od kojih su najvažniji napon napajanja 5V i standardne razine:
-za logičku nulu *0V – 0,8V*, -za logičku jedinicu *2,4V – 5V*.

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 – nova familija brzih TTL integriranih krugova.

4.5. Komplementarna MOS tehnologija:

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

Osnovni sklop CMOS: -u CMOS tehnologiji se koriste 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 zatvoren, te je izlaz u 1.



Karakteristike: -upotreba komplementarnih (P i N kanalnih) MOS tranzistora koji rade u protufazi, velika brzina rada te niska potrošnja energije.

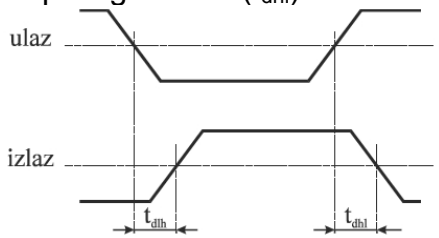
Potrošnja energije CMOS integriranog kruga je niska, jer uvijek jedna od grana sklopa ne vodi. Potrošnja ovisi o broju preklapanja u jedinici vremena. Na visokim frekvencijama se približava potrošnji TTL i NMOS krugova.

Primjena CMOS u VLSI krugovima: -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 ulaznog grananja

Kašnjenje: -definira se kao vrijeme koje protekne između promjene stanja na ulazu u sklop do promjene stanja na izlazu sklopa. Kašnjenje mjerimo od sredine do sredine promjene. Posebno mjerimo vrijeme potrebno da sklop postigne jedinicu (t_{dlh}), a posebno 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: -to su drugi bitan čimbenik kompatibilnosti. Karakteristike ulaznog i izlaznog sklopa TTL logičkih vrata zahtijevaju nesimetrično ponavljanje za 0 i 1, pa je određeno da su te granice za TTL i NMOS tehnologiju 0 – 0,8V max. za logičku nulu i min. 2,4-5V za logičku jedinicu.

Ulazne i izlazne struje su struje koje se nalaze na ulazu i izlazu sklopa, a to su: I_{il}, I_{ih}, I_{ol} i I_{oh}. Važne su zbog toga jer utječu na faktore ulaznog i izlaznog grananja.

Faktor izlaznog grananja: -govori nam koliko standardnih ulaza možemo spojiti na jedan standardni izlaz, a da pritom nisu narušene naponske razine 0 i 1.

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

Faktor ulaznog grananja: -govori nam koliko stvarni ulaz opterećuje izlaz na koji je spojen u odnosu na standardni ulaz za promatranu tehnologiju i varijantu izrade, odnosno koliko je standardnih ulaza integriranog kruga spojeno na istu nožicu kućišta. Ovaj faktor je najčešće 1.

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

5. BOOLEOVA ALGEBRA:

5.1. Booleova 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 logike, = - operator jednakosti, S={0,1} – skup Booleovih konstanti 0 i 1, x $\in$ S – Booleova varijabla koja uzima vrijednost iz S.

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

Algebra logike je Booleova algebra kod koje je skup G:

G={&,v, $\bar{}$ }

A.L.={&,v, $\bar{}$,=,x,S={0,1}}

x_1	x_2	$x_1 \& x_2$	$x_1 \vee x_2$	$\bar{x}_1$	$\bar{x}_2$
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	0	1
1	1	1	1	0	0

Redoslijed operacija: -kod računanja 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

Postulati:

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 i negirane varijable jednaka je nuli)

6.) Asocijativnost

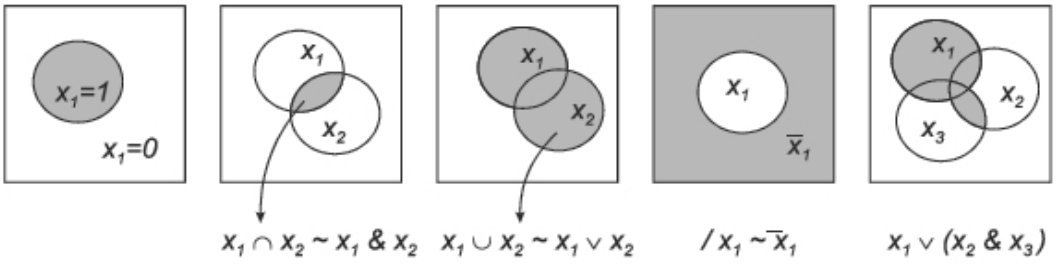
a) $\forall x_1, x_2, x_3 \in S \Rightarrow x_1 \vee (x_2 \vee x_3) = (x_1 \vee x_2) \vee x_3$ (disjunkcija je asocijativna)

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)

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, 2.) Konjunkcija, 3.) Disjunkcija



5.3. Teoremi algebre logike s jednom varijablom:

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

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

Jedinica disjunktivno vezana s nekim izrazom apsorbira taj izraz.

Dokaz: $\equiv (x_1 \vee 1)\cdot 1 = (x_1 \vee 1)\cdot (x_1 \vee \overline{x_1}) = x_1 \vee (1\cdot \overline{x_1}) = x_1 \vee \overline{x_1} = 1$

2.) Apsorpcija za konjunkciju $x_1 \cdot 0=0$

Nula konjunktivno vezana s nekim izrazom apsorbira taj izraz:

Dokaz: $\equiv x_1 \cdot 0 \vee 0 = x_1 \cdot 0 \vee x_1 \cdot \overline{x_1} = x_1 (0 \vee \overline{x_1}) = x_1 \overline{x_1} = 0$

3.) Idempotentnost za disjunkciju $x_1 \vee x_1=x_1$

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

Dokaz: $\equiv (x_1 \vee x_1)\cdot 1 = (x_1 \vee x_1)\cdot (x_1 \vee \overline{x_1}) = x_1 \vee (x_1 \cdot \overline{x_1}) = x_1 \vee 0 = x_1$

4.) Idempotentnost za konjunkciju $x_1 \cdot x_1=x_1$

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

Dokaz: $\equiv x_1 \cdot x_1 \vee 0 = x_1 x_1 \vee x_1 \overline{x_1} = x_1 \cdot (x_1 \vee \overline{x_1}) = x_1 \cdot 1 = x_1$

5.) 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

5.4. Teoremi algebre logike s dvije varijable:

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

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 pa pišemo:

$$A = \overline{x_1 \vee x_2}; \qquad \overline{A} = \overline{\overline{x_1 \vee x_2}} = x_1 \vee x_2; \qquad A = \overline{x_1} \cdot \overline{x_2}$$

$$A \vee \overline{A} = 1 \Rightarrow \overline{x_1} \cdot \overline{x_2} \vee (x_1 \vee x_2) = 1$$

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

$$A \cdot \overline{A} = 0 \Rightarrow \overline{x_1} \cdot \overline{x_2} \cdot (x_1 \vee x_2) = 0$$

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

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:

$$\overline{x_1 \cdot x_2} = \overline{x_1} \vee \overline{x_2} \;\Big/\,\overline{}$$

$$\overline{\overline{x_1 \cdot x_2}} = \overline{\overline{x_1} \vee \overline{x_2}}$$

$$x_1 \cdot x_2 = \overline{\overline{x_1}} \cdot \overline{\overline{x_2}} = x_1 \cdot x_2$$

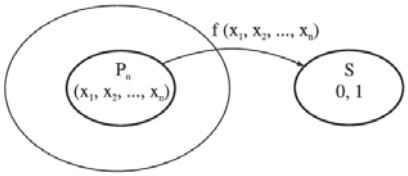
6. BOOLEOVE FUNKCIJE

6.1. Booleova funkcija kao preslikavanje:

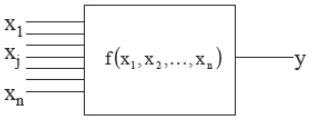
-definirati skup kodnih riječi nad skupom varijabli
-definirati Booleovu funkciju kao preslikavanje
-definirati jednostavni sklop i vezu sa Booleovom funkcijom

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 $P_n(X)$ skup svih kodnih riječi varijabli x. $P_n(X)=\{00\ldots0, 00\ldots1,..., 11\ldots0, 11\ldots1\}$

Definirati Booleovu funkciju kao preslikavanje:
Booleova funkcija $y=f(x_1, x_2, \ldots, x_n)$ 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 Booleove 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

i	x_1	x_2	x_3	y_1	y_2	T_i
0	0	0	0	1	1	T_0
1	0	0	1	0	R	T_1
2	0	1	0	0	0	T_2
3	0	1	1	0	0	T_3
4	1	0	0	1	1	T_4
5	1	0	1	1	1	T_5
6	1	1	0	1	1	T_6
7	1	1	1	1	1	T_7

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 istine.

Standardni oblik tablice istine, broj redaka i njihove oznake: -s lijeve strane tablice se nalaze ulazne varijable ($x_1, x_2, \ldots, x_n$), a s desne strane funkcije ($f_1, f_2, \ldots, f_k$). Svaki redak označavamo rednim brojem '**i**' od 0 do 2^n-1, koji odgovara vrijednosti pripadne kompleksije varijabli promatrane kao prirodni binarni broj.

Potpuno specificirana funkcija: -funkcija kod koje je preslikavanje definirano za sve kodne riječi ulaznih varijabli.

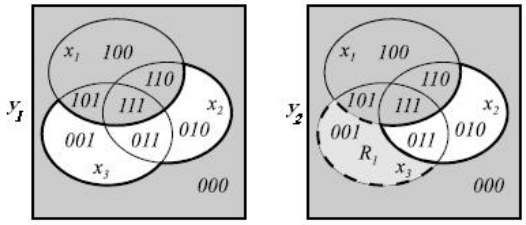
Nepotpuno specificirana funkcija: -funkcija kod koje je preslikavanje definirano samo za neke kodne riječi ulaznih varijabli.

Univerzalna funkcija: -funkcija kojoj još nisu definirane vrijednosti (0 ili 1).

6.3. Grafički zapis Booleovih funkcija:

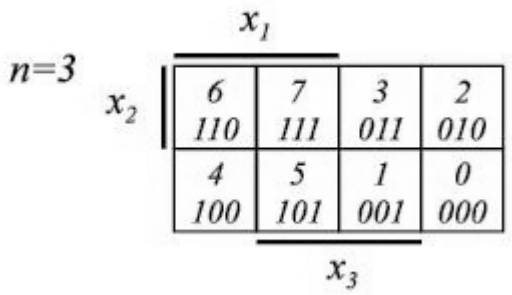
-Vennovi dijagrami
-Veitchevi dijagrami
-standardni oblik do n=6 varijabli

Vennovi dijagrami:



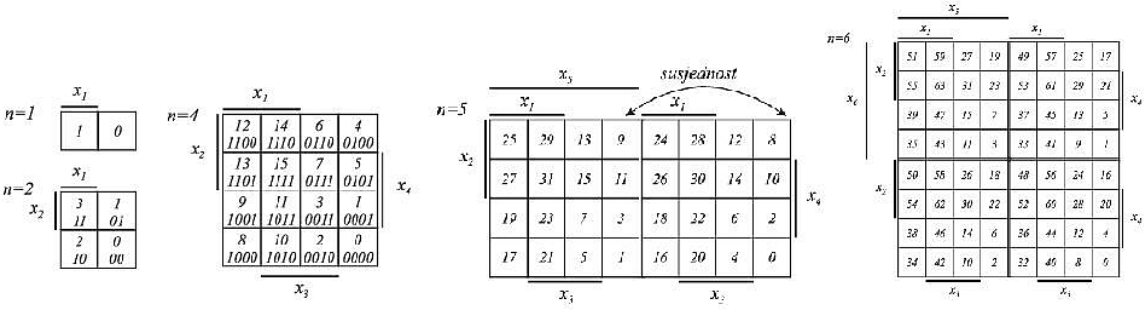
Kod Vennovih dijagrama definiramo područja u okviru univerzalnog skupa nad kojim se pojedina varijabla definira kao logička jedinica. Svako presjecište definirano je jednom kombinacijom vrijednosti varijabli koja ima značenje kompleksije varijabli, pa nad njim možemo definirati vrijednost funkcije, odnosno 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 Veitchevih 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:



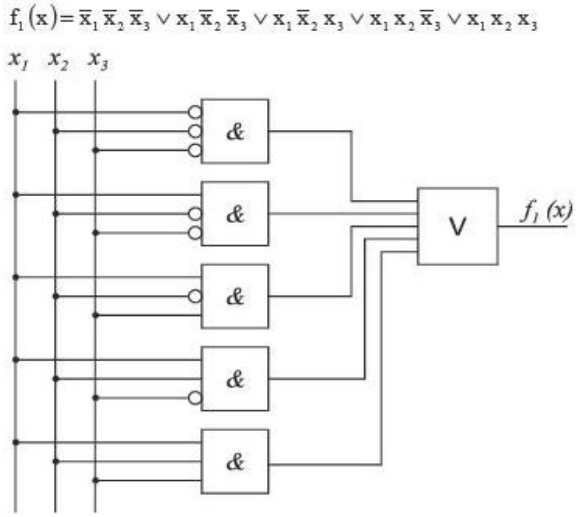
6.4. Ostali načini zapisa Booleove funkcije:

-Grayev kôd i K-tablice
-logički dijagram
-shema sklopa na osnovi PDNO i PKNO

Grayev kôd: -kôd kod kojeg su susjedne kodne riječi kodirane tako da se razlikuju u samo jednom bitu. Koristi se kod optičkih senzora položaja ili kuta.

Karnaughove (K) tablice: -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.

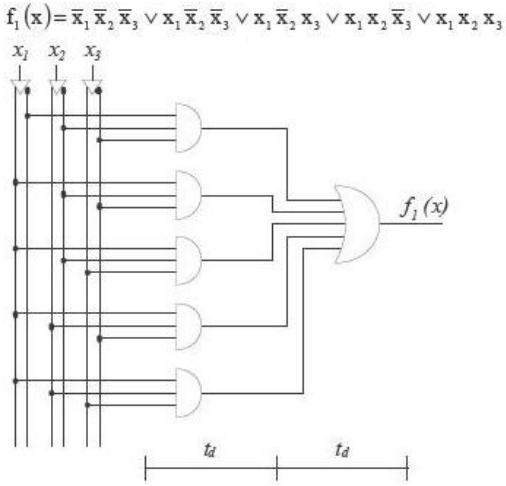
Logički dijagram: -grafički oblik zapisa koji predstavlja blok shemu funkcije i to na način da su varijable i međurezultati prikazani linijama, a operatori blokovima.



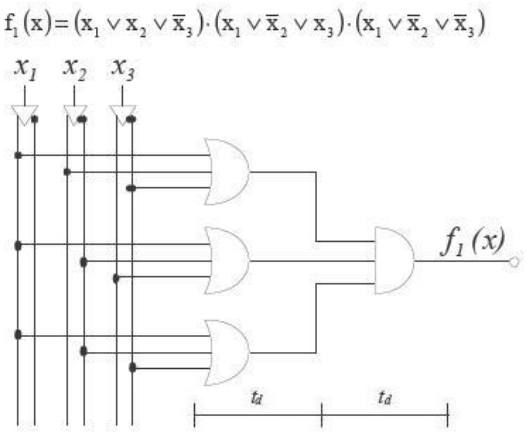
S obzirom da raspolažemo elementarnim sklopovima koji obavljaju konjunkciju, disjunkciju i negaciju, a varijable funkcije predstavljamo električnim signalima, možemo konstruirati sklop po strukturi sličan logičkom dijagramu koji u stvarnosti realizira zadanu funkciju.

Shema sklopa na osnovi

PDNO:



PKNO:



7. NORMALNI ALGEBARSKI OBLICI

7.1. Algebarski zapis potpunim normalnim oblicima:

-motivacija
-definicija PDNO (potpuni disjunktivni normalni oblik) i minterma
-definicija PKNO (potpuni konjunktivni normalni oblik) i maksterma

Motivacija je ta što normalni algebarski izrazi imaju sljedeće karakteristike:
-moguće ih je napisati izravno iz tablice istine
-omogućavaju izradu sklopa s najmanjim kašnjenjem
-sklop ima jednoliko kašnjenje
-moguće ih je minimizirati egzaktnim postupcima
-garantiran je prijelaz na NI ili NILI operatore

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

Minterm m_i i-tog retka tablice istine: -konjunkcija SVIH varijabli tako da su one koje u pripadnoj kodnoj riječi imaju vrijednost nula negirane, a one koje imaju vrijednost jedan nenegirane: $m_3(x_1, x_2, x_3)_{011} = \overline{x_1} \cdot x_2 \cdot x_3$

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

Maksterm M_i i-tog retka tablice istine: -disjunkcija SVIH varijabli tako da su one koje u pripadnoj kodnoj riječi imaju vrijednost jedan negirane, a one koje imaju vrijednost nula 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 nuli: $\bigvee_{i=0}^{2^n-1} m_i = 1 \qquad \&_{i=0}^{2^n-1} M_i = 0$

-Negirani minterm jednak je makstermu istog retka i obrnuto: $\overline{m_i} = M_i;\ \overline{M_i} = m_i$

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

-Konjunkcija različitih minterma uvijek je jednaka nuli, a disjunkcija različitih maksterma uvijek je jednaka jedinici: $m_i \,\&\, m_j \underset{i\neq j}{=} 0 \qquad M_i \vee M_j \underset{i\neq j}{=} 1$

Negirana funkcija: -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 specificirana, negirana je također nepotpuno specificirana za iste retke tablice istine.

Dokaz negirane funkcije pomoću DeMorganovih teorema:

$$f(x) = \bigvee_{i=0}^{2^n-1} m_i T_i \quad \Rightarrow \quad \overline{f}(x) = \overline{\bigvee_{i=0}^{2^n-1} m_i T_i} = \&_{i=0}^{2^n-1} \overline{m_i T_i} = \&_{i=0}^{2^n-1} \left(\overline{m_i} \vee \overline{T_i}\right) = \&_{i=0}^{2^n-1} \left(M_i \vee \overline{T_i}\right)$$

$$\overline{f}(x) = \overline{\&_{i=0}^{2^n-1} \left(M_i \vee T_i\right)} = \bigvee_{i=0}^{2^n-1} \overline{M_i \vee T_i} = \bigvee_{i=0}^{2^n-1} \overline{M_i} \cdot \overline{T_i} = \bigvee_{i=0}^{2^n-1} m_i \cdot \overline{T_i}$$

Dokaz dobijemo neposredno preko $f \vee \overline{f} = 1 \quad i \quad f \cdot \overline{f} = 0$

7.3. Minimalni normalni oblici:

-definicija MDNO
-definicija MKNO

MDNO (Minimalni disjunktivni normalni oblik): -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.

MKNO (Minimalni konjunktivni normalni oblik): -konjunkcija nužnih elementarnih članova tipa maksterma. Član tipa maksterma je disjunkcija nekih ili svih varijabli, negiranih prema pravilu pisanja maksterma. 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:

$$f(x)_{\{x_1,x_2,x_3\}} = \bigvee_{i=0}^{2^n-1} m_i \cdot T_i = \bar{x}_1\bar{x}_2\bar{x}_3T_0 \vee \bar{x}_1\bar{x}_2x_3T_1 \vee \bar{x}_1x_2\bar{x}_3T_2 \vee \bar{x}_1x_2x_3T_3 \vee x_1\bar{x}_2\bar{x}_3T_4 \vee x_1\bar{x}_2x_3T_5 \vee x_1x_2\bar{x}_3T_6 \vee x_1x_2x_3T_7 =$$
$$= \bar{x}_1\bar{x}_2\left(\bar{x}_3T_0 \vee x_3T_1\right) \vee \bar{x}_1x_2\left(\bar{x}_3T_2 \vee x_3T_3\right) \vee x_1\bar{x}_2\left(\bar{x}_3T_4 \vee x_3T_5\right) \vee x_1x_2\left(\bar{x}_3T_6 \vee x_3T_7\right) =$$
$$= \bar{x}_1\bar{x}_2 \cdot f_0(x_3) \vee \bar{x}_1x_2 \cdot f_1(x_3) \vee x_1\bar{x}_2 \cdot f_2(x_3) \vee x_1x_2 \cdot f_3(x_3)$$

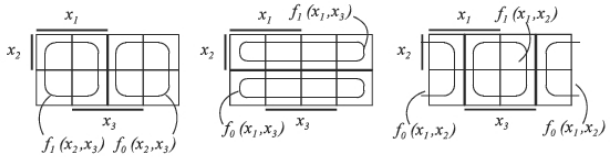
- Mintermi u raspisanom općem obliku PDNO funkcije imaju zajedničke članove, npr za x_1,x_2 to su $\bar{x}_1\bar{x}_2, \bar{x}_1x_2, x_1\bar{x}_2, x_1x_2$, koje možemo izlučiti na osnovi svojstva distributivnosti. U zagradama su ostali neki izrazi koji su očito PDNO funkcija varijable x_3, a koje su preuzele vrijednosti izvorne funkcije T_1 do T_7.

- Funkcije f_0 do f_3 su **parcijalne ili preostale funkcije**, a varijabla x_3 (u ovom primjeru) preostala varijabla:

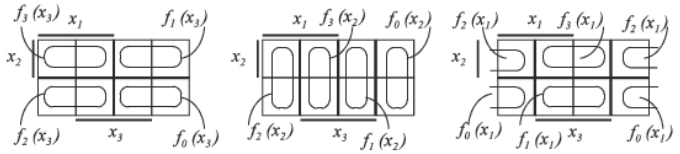
$$f(x)_{\{x_1,x_2,x_3\}} = m_0(x_1x_2) \cdot f_0(x_3) \vee m_1(x_1x_2) \cdot f_1(x_3) \vee m_2(x_1x_2) \cdot f_2(x_3) \vee m_3(x_1x_2) \cdot f_3(x_3) =$$
$$= \bigvee_{j=0}^{2^m-1} m_j(x_1 \cdots x_m) f_j(x_{m+1} \cdots x_n); \qquad f_j(x_{m+1} \cdots x_n) = \bigvee_{k=0}^{2^{n-m}-1} m_k(x_{m+1} \cdots x_n) T_{j \cdot 2^{n-m}+k}$$

Postupak Veitchevog dijagrama:

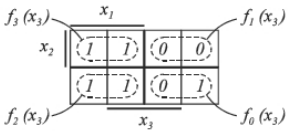
Pogodan postupak izjednačavanja preostalih funkcija je postupak Veitchevog dijagrama. Veitchev dijagram se raspada na dijelove izdvajanjem **m** varijabli. Međutim, zbog svojevrsne simetričnosti dijagrama, ti dijelovi su više-manje suvisli, bez obzira kojih **m** varijabli izdvojili.



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 funkcije, 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

$f_0(x_1) = 0 \rightarrow$ konstanta 0, $f_1(x_1) = \overline{x_1} \rightarrow$ negacija,
$f_2(x_1) = x_1 \rightarrow$ identitet, $f_3(x_1) = 1 \rightarrow$ 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 <u>*funkcije jedne varijable*</u>:
$f_0 = 0, \quad f_3 = \overline{x_1}, \quad f_5 = \overline{x_2}, \quad f_{10} = x_2, \quad f_{12} = x_1, \quad f_{15} = 1$

U <u>*drugu skupinu*</u> funkcija možemo svrstati konjunkciju, disjunkciju i njihove negacije:
$f_1 = \overline{x_1 \vee x_2} \rightarrow$ NILI (NOR, Pierce), $f_7 = \overline{x_1 \cdot x_2} \rightarrow$ NI (NAND, Shaeffer),
$f_8 = x_1 \cdot x_2 \rightarrow$ I (AND, konjunkcija) i $f_{14} = x_1 \vee x_2 \rightarrow$ ILI (OR, disjunkcija)

U <u>*treću skupinu*</u> svrstat ćemo sumu po modulu i njenu negaciju:
$f_6 = x_1 \oplus x_2 \rightarrow$ ekskluzivno ILI (zbroj po modulu) i
$f_9 = \overline{x_1 \oplus x_2} \rightarrow$ ekvivalencija (negacija sume po modulu).

<u>*Preostale četiri funkcije*</u> spadaju 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: -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 ({↑}, Shaeffer, NI, NAND, $\overline{x_1 \cdot x_2}$) je sam za sebe potpun skup funkcija algebre logike.

Dokaz:

$$x_1 \uparrow x_1 = \overline{x_1 \cdot x_1} = \overline{x_1} \vee \overline{x_1} = \overline{x_1} \quad ili \quad x_1 \uparrow 1 = \overline{x_1 \cdot 1} = \overline{x_1} \vee \overline{1} = \overline{x_1}$$

$$(x_1 \uparrow x_1) \uparrow (x_2 \uparrow x_2) = \overline{x_1} \uparrow \overline{x_2} = \overline{\overline{x_1} \cdot \overline{x_2}} = \overline{\overline{x_1}} \vee \overline{\overline{x_2}} = x_1 \vee x_2$$

$$(x_1 \uparrow x_2) \uparrow (x_1 \uparrow x_2) = \overline{x_1 \uparrow x_2} = \overline{\overline{x_1 \cdot x_2}} = x_1 \cdot x_2$$

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 {v, ¯} 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 potpun skup funkcija algebre logike.

Dokaz:

$x_1 \downarrow x_1 = \overline{x_1 \vee x_1} = \overline{x_1} \cdot \overline{x_1} = \overline{x_1}$ *ili* $x_1 \downarrow 0 = \overline{x_1 \vee 0} = \overline{x_1} \cdot \overline{0} = \overline{x_1}$

$(x_1 \downarrow x_1) \downarrow (x_2 \downarrow x_2) = \overline{x_1} \downarrow \overline{x_2} = \overline{\overline{x_1} \vee \overline{x_2}} = \overline{\overline{x_1}} \cdot \overline{\overline{x_2}} = x_1 \cdot x_2$

$(x_1 \downarrow x_2) \downarrow (x_1 \downarrow x_2) = \overline{x_1 \downarrow x_2} = \overline{\overline{x_1 \vee x_2}} = x_1 \vee x_2$

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 {v, ¯}. U takvom zapisu prepoznat ćemo negiranu disjunkciju dviju ili više varijabli kao NILI operator.

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 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 kašnjenja, omogući primjenu postupaka minimizacije, omogući primjenu NI i NILI vrata. Sve ove kriterije zadovoljavaju minimalni normalni oblici.

Svojstva normalnih oblika: -uz uvjete za minimalnost sklopa (da sklop bude minimalan, sa najmanjim brojem logičkih vrata i sa najmanjim kašnjenjem), sklop mora imati najmanji broj logičkih razina za veliki broj ulaznih vrata jer je vrijeme kašnjenja proporcionalno broju logičkih razina (serijski spojenih logičkih vrata).

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:

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

Primjer s komentarom primjene postulata i teorema:
- za PDNO:

- polazimo od PDNO: $x_1x_2x_3 \vee x_1x_2\overline{x_3} \vee ... =$
- asocijativnost za konjunkciju: $= (x_1x_2)(x_3) \vee (x_1x_2)(\overline{x_3}) \vee ... =$
- distributivnost: $= x_1x_2(x_3 \vee \overline{x_3}) \vee ... =$
- komplementiranje: $= x_1x_2 \cdot 1 \vee ... =$
- neutralni element: $= x_1x_2 \vee ...$

- za PKNO:

- polazimo od PKNO: $(x_1 \vee \overline{x_2} \vee x_3) \cdot (\overline{x_1} \vee \overline{x_2} \vee x_3) \cdot (...) =$
- komutativnost: $= (\overline{x_2} \vee x_3 \vee x_1) \cdot (\overline{x_2} \vee x_3 \vee \overline{x_1}) \cdot (...) =$
- distributivnost: $= (\overline{x_2} \vee x_3 \vee (x_1 \cdot \overline{x_1})) \cdot (...) =$
- komplementiranje: $= (\overline{x_2} \vee x_3 \vee 0) \cdot (...) =$
- neutralni element: $= (\overline{x_2} \vee x_3) \cdot (...)$

Ušteda 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 idempotentnosti (x v 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 izraz proširujemo postojećim članom, vrijedi:

$$x_1x_2x_3 \vee x_1x_2\overline{x_3} \vee x_1\overline{x_2}x_3 \vee ... = x_1x_2x_3 \vee x_1x_2\overline{x_3} \vee \underline{x_1x_2x_3} \vee x_1\overline{x_2}x_3 \vee ... = x_1x_2 \vee x_1x_3 \vee ...$$

(Proširenje (dodani član) je podcrtan). Vidimo da je član koji smo dodali susjedan sa oba člana u izrazu pa ga nakon proširivanja možemo udružiti sa oba susjeda, što prije proširenja nije bilo moguće. Nakon minimizacije smo od četiri člana (3 izvorna i 1 dodani) sa n (3) varijabli došli na dva člana sa n-1 (2) varijablom.

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

$$x_1 \cdot x_2 \cdot x_3 \vee x_1 \cdot x_2 \cdot \overline{x_3} \vee x_1 \cdot \overline{x_2} \cdot x_3 \vee x_1 \cdot \overline{x_2} \cdot \overline{x_3} \cdot R^{=1} \vee \ldots =$$

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

Vidimo da je u osnovnom algebarskom izrazu je jedan član redundantan, tj pripadna kodna riječ nije iskorištena. Kako ta kodna riječ nikada neće doći na ulaz sklopa, možemo zaključiti da je svejedno hoće li redundantni član biti uključen u PDNO ili ne.

Komentirati uštedu i značaj postupka: -ušteda pomoćnim algebarskim postupcima se krije u tome što dodavanjem članova proširimo funkciju ali je istovremeno minimiziramo za jednu ulaznu varijablu ili za cijelu logičku funkciju, tako da uštedimo na broju ulaznih logičkih vrata i/ili na broju funkcija.

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 i brz. Algebarska minimizacija normalnih oblika provodi se transformacijama nad algebarskim izrazom. Primjenom postulata i teorema A.L., iz PDNO dobije se MDNO, a iz PKNO dobije se MKNO. U praksi, MKNO se dobije transformacijom negirane funkcije, a negiranu koristimo zato jer tada zbog svojstva $\overline{M_i} = m_i$ umjesto MKNO imamo negirani MDNO, a njim se zbog jednostavnosti zapisa lakše raditi i manja je mogućnost greške.

Svojstva negirane funkcije: -funkcija koja ima vrijednost 1 tamo gdje izvorna funkcija ima vrijednost 0, a 0 gdje izvorna funkcija ima vrijednost 1. Ako je izvorna funkcija nepotpuno specificirana tada je i negirana funkcija također nepotpuno specificirana za iste retke tablice istine. Negirana funkcija u V. D. ima isti raspored nula kao originalna jedinica pa su zaokruživanje i susjednost identični.

Definirati proceduru minimizacije:

1.) Definirati negiranu funkciju i upisati je u Veitchev dijagram
2.) Traženje članova čije su pripadne kodne riječi susjedne
3.) Minimiziramo dobivenu funkciju kao da minimiziramo PDNO, a ne PKNO jer je $\overline{M_i} = m_i$ (radom sa PDNO je manja mogućnost greške, iz tog razloga je i negirana)
4.) Dobijemo negirani MDNO
5.) Dobivenu funkciju negiramo jednom jer nas ne zanima negirana funkcija nego originalna, dobivamo minterme vezane disjunkcijom, a sve je negirano pa već imamo NILI sklop
6.) Pojedine minterme koji imaju konjunkciju u sebi dvaput negirati, riješiti pomoću DeMorganovih teorema te dobivamo konačnu minimizaciju na NILI sklopove.

Primjer s komentarom primjene postulata i teorema:

- polazimo od MDNO negirane funkcije (iz VD):

$$\overline{f}(x) = x_2 x_3 \vee \overline{x_1} x_2 \vee \overline{x_1} x_3$$

- cijelu funkciju negiramo:

$$f(x) = \overline{x_2 x_3 \vee \overline{x_1} x_2 \vee \overline{x_1} x_3}$$

- dvaput negiramo minterme koji u sebi imaju konjunkciju:

$$f(x) = \overline{\overline{x_2 x_3} \vee \overline{\overline{x_1} x_2} \vee \overline{\overline{x_1} x_3}}$$

- koristeći DeMorganove teoreme dobijemo konačan izraz za NILI sklop:

$$f(x) = \overline{\overline{x_2} \vee \overline{x_3} \vee x_1 \vee \overline{x_2} \vee x_1 \vee \overline{x_3}}$$

10. POSTUPCI MINIMIZACIJE I REALIZACIJE NI I NILI VRATIMA

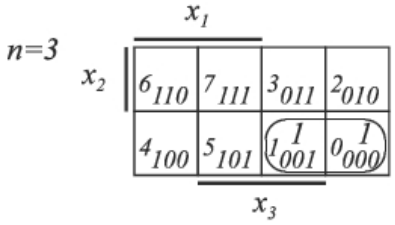
10.1. Postupak minimizacije Veitchevim dijagramom:

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

Pokazati zapis funkcije pomoću VD:

U VD funkciju upisujemo tako da za PDNO upisujemo 1 i R, a za PKNO 0 i R na odgovarajuća mjesta zadana funkcijom.

Npr: $\overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \vee \overline{x_1} \cdot \overline{x_2} \cdot x_3 = \overline{x_1} \cdot \overline{x_2}$



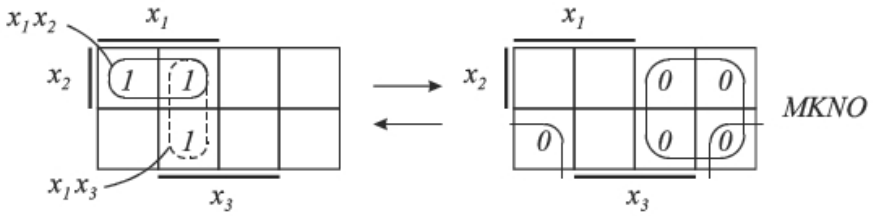
Susjednim mintermima odgovaraju susjedna područja!
Rezultat minimizacije je ekvivalentan ujedinjavanju područja!

Pokazati osnovni postupak minimizacije na VD:

Osnovni postupak minimizacije u postupku VD odgovara objedinjavanju susjednih površina, tj. zaokruživanjem što manjim brojem što većih površina.

Primjer: $f(x) = x_1 \cdot x_2 \cdot \overline{x_3} \vee x_1 \cdot x_2 \cdot x_3 \vee x_1 \cdot \overline{x_2} \cdot x_3$

PDNO: *PKNO:*

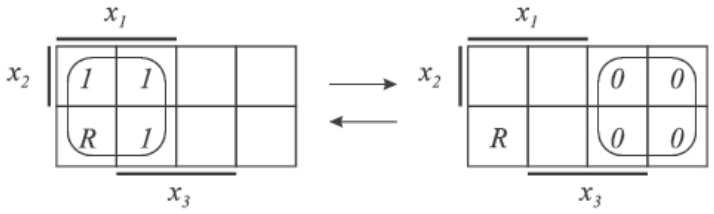


$f(x) = x_1 \cdot x_2 \vee x_1 \cdot x_3$ $\qquad$ $f(x) = x_1 \cdot (x_2 \vee x_3)$

Pokazati pomoćne postupke minimizacije na VD:
Pomoćni postupak minimizacije dopisivanjem redundantnog člana u algebarskom obliku odgovara zaokruživanju kvadrata VD u kojem je upisana oznaka R. Višestruko zaokruživanje nekog člana znači proširenje postojećim članom (u algebarskom izrazu to predstavlja dopisivanje postojećeg člana).
Za MDNO time je definirana vrijednost preslikavanja funkcije u 1, a za MKNO je definirana vrijednost preslikavanja funkcije u 0.

Primjer: $x_1 \cdot x_2 \cdot \overline{x_3} \vee x_1 \cdot x_2 \cdot x_3 \vee x_1 \cdot \overline{x_2} \cdot x_3 \vee x_1 \cdot \overline{x_2} \cdot \overline{x_3} \cdot R = x_1 \cdot x_2 \vee x_1 \cdot \overline{x_2} = x_1$

PDNO: PKNO:



$f(x) = x_1$

Pravila i postupak minimizacije pomoću VD:

- U VD upisujemo samo jedinice i R za PDNO, samo nule i R za PKNO

- Svakom mintermu (makstermu) za koji je funkcija jednaka jedinici (nuli) tražimo susjeda. Ukoliko promatrani minterm (maksterm) nema susjednog člana, proglašavamo ga elementarnim članom.

- Zaokružimo sve jedinice **što manjim brojem što većih površina**

- Površine susjednih minterma (maksterma) udružujemo u veće pravokutnike koji predstavljaju izraze dužine n-1, gdje je n broj varijabli. Eliminirana je ona varijabla po kojoj se dva udružena područja razlikuju. Članovima dužine n-1 ponovo tražimo susjede koji također moraju imati dužinu n-1. Ukoliko ih pronađemo, eliminiramo još jednu varijablu i dobijemo član dužine n-2. Članovi koji nemaju susjede, smatraju se elementarnima. Višestruko zaokruživanje neke površine ekvivalentno je dopisivanju postojećeg člana normalnog oblika. **Višestruko zaokruživanje** neke površine ekvivalentno je dopisivanju postojećeg člana normalnog oblika. **Zaokruživanje redundantnog člana R** ekvivalentno je njegovom dopisivanju u algebarski oblik.

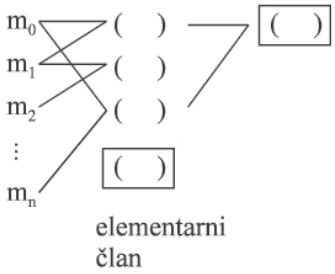
-Postupak ponavljamo sve dok ne ostanu samo elementarni članovi. Cilj je sve jedinice (nule) obuhvatiti što manjim brojem što većih površina.

-Ispisujemo minimalni algebarski oblik funkcije (MDNO ili MKNO), tako da za svaku površinu ispišemo pripadni elementarni član. Uzimamo samo nužne elementarne članove.

10.2. Quinne-McClusky postupak minimizacije:

-pokazati zapis funkcije s pomocu 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 pomocu QMC postupka

Zapis funkcije s pomoću QMC postupka:
QMC postupak je tablično-algebarski postupak kod kojeg sve minterme PDNO ispišemo jedna ispod drugog, a zatim provjeravamo susjednost svih parova redom od gore prema dolje.



Osnovni postupak minimizacije na QMC:
Ako su dva člana susjedna, desno ispisujemo reducirani član. Ako smo dobili nove članove, na isti način provjeravamo njihovu susjednost, te ispisujemo eventualne reducirane članove. Postupak je gotov kada više nema članova koji bi bili susjedni. Postupak je definiran za PDNO, a MKNO dobijemo pomoću negirane funkcije. Dobiveni članovi koji s desne strane nemaju kraći, reducirani član su elementarni.

Pomoćni postupci minimizacije na QMC:
Provjerimo sve moguće susjednosti i ispišemo reducirane članove, te konačno elementarne članove. Pri tome redundantni član uzimamo u razmatranje kao da je uključenu PDNO. Uočimo da se reducirani članovi nakon druge iteracije pojavljuju po dva puta, što samo pokazuje dva načina na koji ih možemo (algebarski) dobiti.

Izbor nužnih elementarnih članova:
Nužan član je onaj koji sadrži neki od početnih članova (minterma), osim redundantnog. Ako za neki minterm postoje dva elementarna člana biramo jednog po volji.

Pravila i postupak minimizacije s pomocu QMC postupka:
- 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 kada više nema susjednih članova
- postupak je definiran za PDNO, a MKNO dobijemo preko negirane funkcije

10.3. Harvardski postupak minimizacije:

-pokazati zapis funkcije s pomocu 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

Zapis funkcije s pomoću Harvardskog postupka:
Harvardski postupak je nastao iz Quinne-McClusky postupka, tako da su za određeni broj varijabli unaprijed izračunati svi reducirani članovi. To omogućava formiranje standardnih tablica, kojima se primjenom čvrstih pravila jednostavno minimizira proizvoljna funkcija.
Postupak je također definiran za PDNO, a MKNO dobijemo preko negirane funkcije.
Značaj HV postupka je u mogućnosti programiranja na računalu.

i	x_1	x_2	x_3	$x_1 x_2$	$x_1 x_3$	$x_2 x_3$	$x_1 x_2 x_3$	$f(x)$
0	$\overline{x}_1$	$\overline{x}_2$	$\overline{x}_3$	$\overline{x}_1\overline{x}_2$	$\overline{x}_1\overline{x}_3$	$\overline{x}_2\overline{x}_3$	$\overline{x}_1\overline{x}_2\overline{x}_3$	0
1	$\overline{x}_1$	$\overline{x}_2$	x_3	$\overline{x}_1\overline{x}_2$	$\overline{x}_1 x_3$	$\overline{x}_2 x_3$	$\overline{x}_1\overline{x}_2 x_3$	R
2	$\overline{x}_1$	x_2	$\overline{x}_3$	$\overline{x}_1 x_2$	$\overline{x}_1\overline{x}_3$	$x_2\overline{x}_3$	$\overline{x}_1 x_2\overline{x}_3$	0
3	$\overline{x}_1$	x_2	x_3	$\overline{x}_1 x_2$	$\overline{x}_1 x_3$	$x_2 x_3$	$\overline{x}_1 x_2 x_3$	0
4	x_1	$\overline{x}_2$	$\overline{x}_3$	$x_1\overline{x}_2$	$x_1\overline{x}_3$	$\overline{x}_2\overline{x}_3$	$x_1\overline{x}_2\overline{x}_3$	1
5	x_1	$\overline{x}_2$	x_3	$x_1\overline{x}_2$	$x_1 x_3$	$\overline{x}_2 x_3$	$x_1\overline{x}_2 x_3$	1
6	x_1	x_2	$\overline{x}_3$	$x_1 x_2$	$x_1\overline{x}_3$	$x_2\overline{x}_3$	$x_1 x_2\overline{x}_3$	1
7	x_1	x_2	x_3	$x_1 x_2$	$x_1 x_3$	$x_2 x_3$	$x_1 x_2 x_3$	1

Osnovni postupak minimizacije HV postupkom:
Pomoćni postupci minimizacije HV postupkom:

?????? U KNJIZI NEMA!!!

Izbor nužnih elementarnih članova:
Obavljamo ga po već poznatom kriteriju: najprije odredimo eksplicitno nužne članove, dakle one koji jedini sadrže neki od početnih minterma. Nakon toga među ostalima izaberemo članove po volji, tako da funkcija bude kompletna, a dobiveni izraz minimalan. Po potrebi, uključujemo i članove koji proizlaze iz redundantnih članova.

Pravila i postupak minimizacije s pomoću HV postupka:
- upisati funkciju (kombinacije varijabli i vrijednost funkcije)
- precrtati retke za koje je funkcija jednaka nuli (redundantne NE precrtavati)
- precrtati (horizontalnim crtama) brojeve po stupcima koji su precrtani u prvom koraku
- provjeravajući s desna na lijevo, precrtati po redcima (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

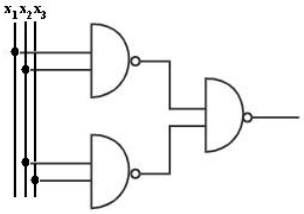
Postupak i transformacija za NI vrata:

- polazimo od PDNO
- izračunamo MDNO
- dvostruko negiramo algebarski izraz
- primjenom DeMorganovih teorema transformiramo izraz
- dobijemo izraz za NI vrata zapisan sustavom {&, ¯}

Primjer s komentarom primjene postulata i teorema:

- MDNO: $f(x) = x_1 \cdot x_2 \vee x_2 \cdot x_3$
- Dvostruka negacija: $f(x) = \overline{\overline{x_1 \cdot x_2 \vee x_2 \cdot x_3}}$
- DeMorganovim teoremima dobijemo konačan izraz:

$$f(x) = \overline{\overline{x_1 \cdot x_2} \,\&\, \overline{x_2 \cdot x_3}}$$



10.5. Minimizacija i realizacija NILI vratima:

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

Postupak i transformacija za NILI vrata:

- polazimo od PDNO negirane funkcije
- izračunamo MDNO negirane funkcije
- negiramo lijevu i desnu stranu jednadžbe
- u gornjoj negaciji s disjunkcijama prepoznamo NILI operator
- dvostruko negiramo svaki pojedini član tipa minterma
- primjenom DeMorganovih teorema transformiramo članove tipa minterma u članove tipa maksterma, tj. konjunkcije u disjunkcije
- uočimo da dobivene disjunkcije s gornjom negacijom čine NILI vrata
- dobijemo izraz za NILI vrata zapisan sustavom {v, ¯}

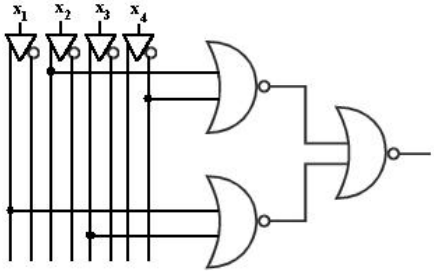
Primjer s komentarom primjene postulata i teorema:

- MDNO: $\overline{f}(x) = \overline{x_2} \cdot x_4 \vee \overline{x_1} \cdot \overline{x_3}$
- Negacija: $f(x) = \overline{\overline{x_2} \cdot x_4 \vee \overline{x_1} \cdot \overline{x_3}}$
- Dvostruko negiramo svaki pojedini član tipa minterma:

$$f(x) = \overline{\overline{\overline{x_2} \cdot x_4} \vee \overline{\overline{x_1} \cdot \overline{x_3}}}$$

- DeMorganovim teoremima za svaki pojedini član dobijemo konačan izraz:

$$f(x) = \overline{\overline{\overline{x_2} \vee \overline{x_4}} \vee \overline{\overline{x_1} \vee x_3}}$$

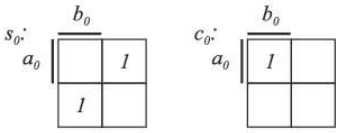


10.6. Sinteza sklopova za zbrajanje:

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

Problem se javlja kada je minimizacija otežana zbog dijagonalno postavljenih jedinica u Veitchevu dijagramu.

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

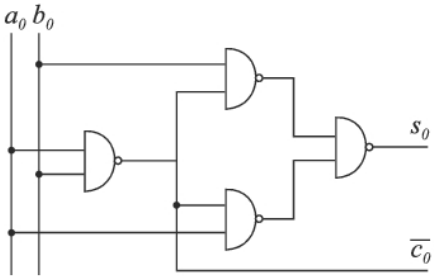


$$s_0 = a_0 \oplus b_0 = \overline{\overline{b_0}\, a_0}\ \overline{b_0\, \overline{a_0}}$$

$$c_0 = a_0 b_0 = \overline{\overline{a_0 b_0}}$$

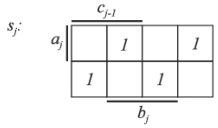
Zbrajanje na LSB – polusumator:
Moguća je sljedeća transformacija:

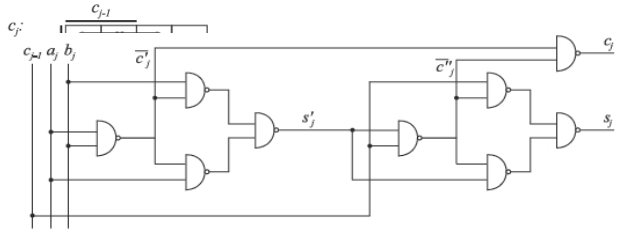
$$s_0 = b_0\overline{a_0} \vee \overline{b_0}a_0 \vee 0 \vee 0 =$$
$$= b_0\overline{a_0} \vee \overline{b_0}a_0 \vee b_0\overline{b_0} \vee a_0\overline{a_0} =$$
$$= b_0\overline{\left(\overline{a_0} \vee \overline{b_0}\right)} \vee a_0\overline{\left(\overline{b_0} \vee \overline{a_0}\right)} =$$
$$=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



Dobiveni sklop nazivamo polusumator. Uštedjeli smo ulazne invertore, a sklop daje negirani pretek:

Zbrajanje s pretekom:
Pokušajmo minimizirati:





Transformirajmo:

$$s_j = a_j\overline{b_j}\,\overline{c_{j-1}} \vee a_j b_j c_{j-1} \vee \overline{a_j}\,\overline{b_j} c_{j-1} \vee \overline{a_j} b_j \overline{c_{j-1}} =$$
$$= c_{j-1}\left(a_j b_j \vee \overline{a_j}\,\overline{b_j}\right) \vee \overline{c_{j-1}}\left(a_j\overline{b_j} \vee \overline{a_j} b_j\right) =$$
$$= c_{j-1} \cdot \overline{\left(a_j \oplus b_j\right)} \vee \overline{c_{j-1}} \cdot \left(a_j \oplus b_j\right) =$$
$$= c_{j-1} \oplus \left(a_j \oplus b_j\right)$$

Sklop generira pretek, a izraz u zagradama je suma po modulu pa možemo pisati:

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

Komentar kašnjenja za n-bitni sumator:
Mana potpunog sumatora je u kašnjenju zbroja za 6 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.
Iz toga je ukupno kašnjenje: $T(n)=(4+2n)t_d$

II. TREĆINA GRADIVA

11. KOMBINACIJSKI SKLOPOVI SREDNJEG STUPNJA INTEGRACIJE:

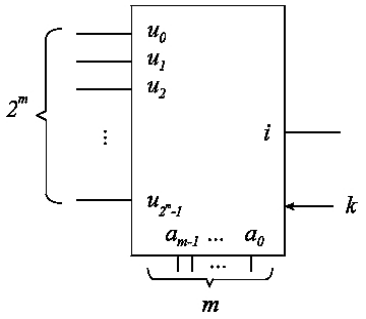
11.1. Selektor/multiplekser:

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

Definirati funkciju selektora/multipleksera:

Multiplekser je logički sklop koji na informacijski izlaz **i** propušta vrijednost onog od $n=2^m$ informacijskih ulaza **u** ($u_0,...,u_{n-1}$) čiji je redni broj prisutan u prirodnom binarnom obliku na **m** adresnih ulaza ($a_0,...,a_{m-1}$), pod uvjetom da je funkcija sklopa omogućena aktivnim signalima na kontrolnim ulazima.

Algebarski izraz multipleksera: $i = \bigvee_{j=0}^{2^m-1} m_j \,\&\, u_j$



Izvesti formulu za selektor/multiplekser:

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:

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

Dvostrukom negacijom dobili smo mogućnost korištenja NI vrata.

Opisati uporabu selektora/multipleksera:

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

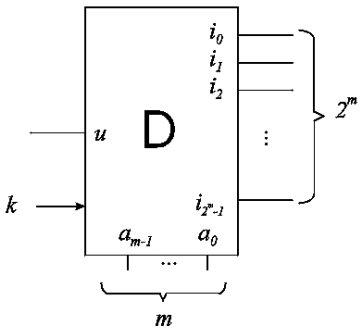
11.2. Dekoder/demultiplekser:

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

Definirati funkciju dekodera/demultipleksera:

Demultiplekser je logički sklop koji vrijednost informacijskog ulaza **u** propušta na onaj od $n=2^m$ informacijskih izlaza **i** ($i_0,...,i_{n-1}$) čiji je redni broj prisutan u prirodnom binarnom obliku na **m** adresnih ulaza ($a_0,...,a_{m-1}$), pod uvjetom da je funkcija sklopa omogućena aktivnim signalima na kontrolnim ulazima.

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



Izvesti formulu za dekoder/demultiplekser:

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 se dobije mogućnost korištenja NI vrata:

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

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

Opisati uporabu dekodera/demultipleksera:

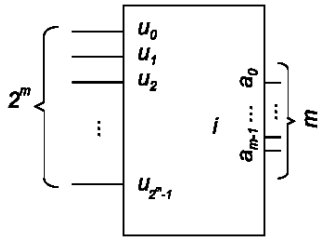
Demultiplekser se koristi za razvođenje ulaza na izlaz, serijsko-paralelnu konverziju, kao dekoder i za realizaciju Booleovih funkcija.
- Ako je informacijski ulaz slobodno promjenjiv, a adresa stacionarna, odabrani izlaz će slijediti vrijednost sa ulaza. Obavili smo razvođenje ulaza na odabrani izlaz.
- Ako se informacijski ulaz mijenja istovremeno (sinkrono) sa adresama, a adrese mijenjamo u prirodnom binarnom nizu, na izlazima će se pojaviti niz bitova sa ulaza. Obavili smo serijsko-paralelnu konverziju.
- Ako na ulaz trajno dovedemo 1, a adresom biramo jedan od izlaza, demultiplekser preuzima funkciju dekodera.

11.3. Enkoder s prioritetom:

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

Definirati funkciju enkodera:
Enkoder je uređaj koji se koristi za kodiranje signala ili podataka u oblik pogodan za prijenos ili pohranu.

Definirati funkciju enkodera s prioritetom:
Enkoder prioriteta na adresne izlaze $\mathbf{a_{m-1}}$, $\mathbf{a_{m-2}}$,..., $\mathbf{a_0}$ 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.



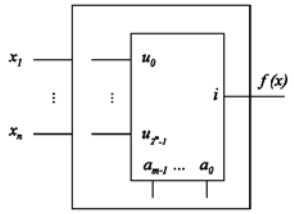
Opisati uporabu enkodera s prioritetom:
Enkoder prioriteta se koristi uvijek kada je potrebno neki kôd tipa jedan od 2^m prevesti u koncentrirani kôd od **m** bita. Funkcija mu je suprotna funkciji dekodera. Njime možemo realizirati jednostavnu tastaturu. Masovno se primjenjuje u mikroprocesorima (za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

12. REALIZACIJA BOOLEOVIH FUNKCIJA 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:



Osnovna jednadžba realizacije:

Multiplekser ima jedan izlaz pa samo na tom izlazu možemo dobiti vrijednost funkcije: $i = f(x_1,...,x_n)$

Uvrštavanjem izraza za multiplekser i PDNO funkcije slijedi:

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

Komentar veličine problema i mogućih rješenja:

Imamo jednu jednadžbu, a **m+2^m** nepoznanica. Potrebno je spojiti **m** adresnih i **2^m** informacijskih ulaza multipleksera za **n** varijabli funkcija.

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

Jednadžba realizacije za n=m:

Kada je broj varijabli jednak broju adresnih ulaza, tj. **n=m**, iz osnovne jednadžbe realizacije dobijemo:

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

Lijeva i desna strana dobivene jednadžbe su strukturno identične pa običnu jednakost možemo zamijeniti identitetom te izjednačiti po dijelovima pa dobijemo:

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

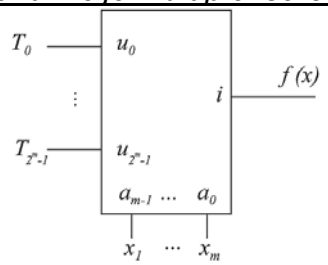
$$m_j(a) = m_j(x) \qquad j = 0,...,2^m-1$$

Specijalno rješenje za n=m:
Ako na adresne ulaze multipleksera dovedemo varijable funkcije redom:

$$\begin{matrix} a_{m-1} & a_{m-2} & & a_1 & a_0 \\ \uparrow & \uparrow & \dots & \uparrow & \uparrow \\ x_1 & x_2 & & x_{m-1} & x_m \end{matrix} \qquad \underline{\text{Dobijemo}}: \quad a_e = x_{m-e}; \qquad e = m-1,\dots,0$$

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

Struktura realizacije Booleove funkcije multiplekserom za m=n:



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 multipleksersko stablo

Jednadžba realizacije za n>m:
Za opći slučaj n>m gubi se strukturni identitet:

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

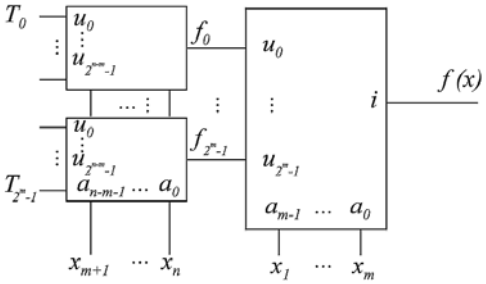
Algebarsko rješenje za n>m:
Da bismo postigli strukturni identitet, transformiramo desnu stranu. Koristimo razbijanje na preostale funkcije.
Izraz u zagradi je PDNO preostale funkcije pa pišemo:

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

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

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

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

Potpuno multiplekserske stablo:


Preostale funkcije treba realizirati posebnim sklopovljem koji mogu biti logička vrata ali i multiplekseri. Ako smo ih realizirali multiplekserima, dobili smo strukturu koju nazivamo **multiplekserske stablo**. Ako se svaka preostala funkcija realizira jednim multiplekserom, tj. ako korijenski multiplekser na svim informacijskim ulazima ima jedan multiplekser tada imamo potpuno multiplekserske stablo (vidi sliku!). Potpuno stablo je ekvivalentno jednom većem 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

Mogućnost minimizacije multiplekserskog stabla:
Minimizacija multiplekserskog stabla je moguća tako da eliminiramo čitavu granu stabla.

Kriterij minimizacije multiplekserskog 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.

Specijalni slučaj optimalnog sklopa s multiplekserom:
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 sa **m** adresnih ulaza realizirati funkciju sa **n=m+1** varijabli.

Metodologija minimizacije multiplekserskog stabla:
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.

13. REALIZACIJA BOOLEOVIH FUNKCIJA DEMULTIPLEKSEROM:

13.1. Pristup realizaciji Booleove funkcije demultiplekserom:

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

Demultiplekser ima **m** adresnih ulaza $\mathbf{a_{m-1}, a_{m-2},...,a_1, a_0}$, **jedan** informacijski ulaz **u** i $\mathbf{2^m}$ informacijskih izlaza **i**, 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 logička ILI vrata i tako neposredno realizirati PDNO Booleove funkcije.

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

13.2. Realizacija Booleovih funkcija 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 demultipleksera

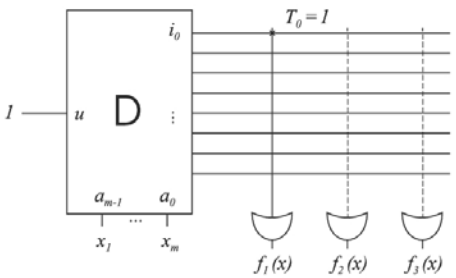
Jednadžba realizacije za n=m:
Koristimo demultiplekser kao dekoder (u=1), pa vrijedi:
$i_j = m_j(a) \cdot u = m_j(a) \cdot 1 = m_j(a)$

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

Specijalno rješenje: - Ako na adresne ulaze demultipleksera dovedemo varijable funkcije redom:

$$\begin{matrix} a_{m-1} & a_{m-2} & a_1 & a_0 \\ \uparrow & \uparrow & \uparrow & \uparrow \\ x_1 & x_2 & x_{m-1} & x_m \end{matrix} \quad \underline{Dobijemo}: \; m_j(a) = m_j(x) = i_j \quad \Rightarrow \quad f(x) = \bigvee_{j=0}^{2^m-1} i_j \cdot T_j$$

Konstrukcija sklopa za n=m: -na ILI vrata spojimo samo one izlaze za koje vrijednost funkcije jednaka jedinici (simbolički prikaz):

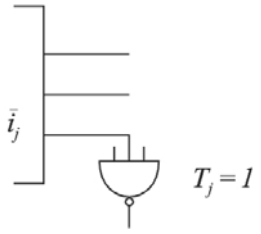


Mana: realizira PDNO, nema minimizacije.
Prednost: realizira više funkcija istih varijabli.

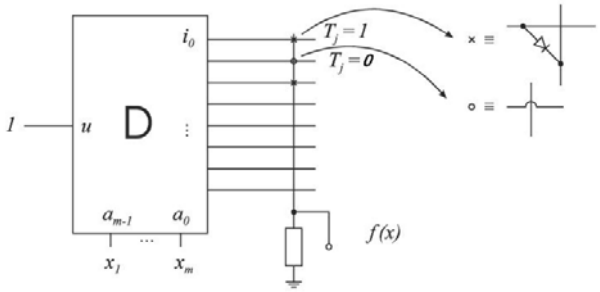
Problem kod konstrukcije ILI vrata: -kod konstrukcije tih vrata javlja se problem pa dvostruko negiramo izraz i primijenimo DeMorganove teoreme:

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

Sada vidimo da možemo koristiti demultiplekser sa negiranim izlazima i NI vrata:



Komentar rada: -umjesto ILI odnosno NI vrata koristimo diodnu logiku, tj. jednostavnim dodavanjem dioda lagano realiziramo ILI vrata s potrebnim brojem ulaza. Za više funkcija istih varijabli dobijemo polje dioda ("diodna matrica").



13.3. Realizacija Booleovih funkcija demultiplekserom za n>m:

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

Jednadžba realizacije za n>m:
Za n>m pokušajmo transformirati PDNO funkcije:

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

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

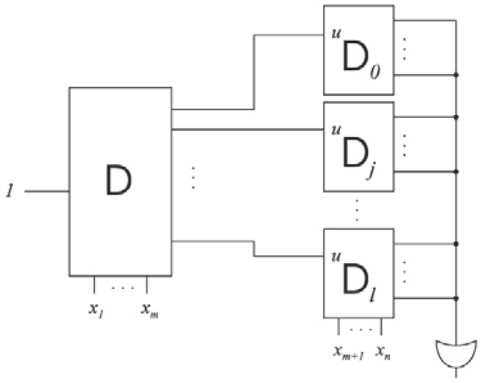
$$\Rightarrow f(x)=\bigvee_{j=0}^{2^m-1} i_j \bigvee_{k=0}^{2^{n-m}-1} m_k(x) T_{2^{n-m}\cdot j+k}$$

Rješenje rezultira nezgrapnim sklopom.

Algebarsko rješenje za n>m:
Ako preostalu funkciju realiziramo demultiplekserom možemo koristiti njegov informacijski ulaz koji je konjunktivno vezan sa izlazima, a ranije smo ga isključili:

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

Konstrukcija sklopa za n>m:
Izlaz glavnog demultipleksera dovedemo na ulaz demultipleksera preostale funkcije, čime ga aktiviramo.
Dobili smo ***demultiplekserskо stablo***!
Potpuno demultipleksersko stablo se dobije kada na svim izlazima korijenskog demultipleksera ima po još jedan demultiplekser. Kada je potpuno, stablo je ekvivalentno jednom većem demultiplekseru. Sklop sada ima strukturu:



13.4. Minimizacija demultiplekserskog stabla:

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

Mogućnost minimizacije demultiplekserskog stabla: -struktura stabla nam omogućava minimizaciju demultiplekserskog stabla eliminacijom pojedine grane stabla. To je moguće potpunom eliminacijom preostale funkcije, tj. kada je preostala funkcija jednaka KONSTANTI 1 ili 0.
Veličina preostalih funkcija (pa i cijelog sklopovlja) ovisi o izboru adresnih varijabli demultipleksera.

Metodologija minimizacije demultiplekserskog stabla: -adresne varijable se biraju tako da što veći broj preostalih funkcija bude konstanta.

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

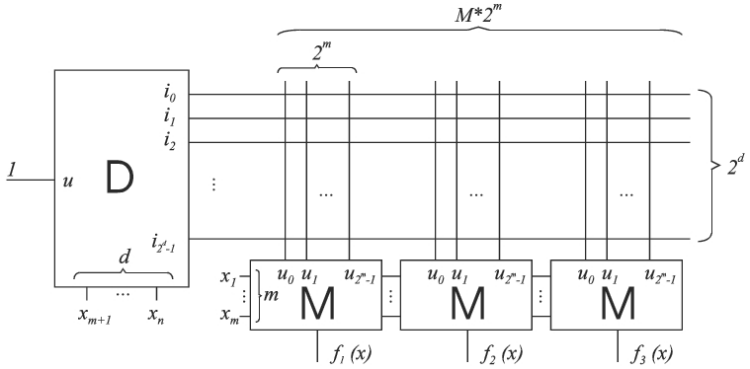
14. MULTIPLESKERSKO – DEMULTIPLEKSERSKA (MD) STRUKTURA:

14.1. Multipleksersko – demultiplekserska struktura:

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

Motivacija: -želja za realizacijom tehnologije visoke i vrlo visoke razine integracije.
Koncept: MD struktura služi za realizaciju Booleovih funkcija istih varijabli. Broj adresnih ulaza u strukturu je jednak zbroju adresnih ulaza **m** u multiplekser i zbroju adresnih ulaza **d** u demultiplekser.

Principijelna shema MD strukture:



Broj redaka i stupaca matrice:
Broj redaka R matrice jednak je broju izlaza demultipleksera (2^d). Broj stupaca S matrice jednak je broju multipleksera pomnoženi s brojem ulaza u pojedini multiplekser ($M \cdot 2^m$).

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

Optimalni broj logičkih vrata: -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 (kvadratna struktura → $2^d \approx M \cdot 2^m$). Broj logičkih vrata: $L=R+S=2^d+M \cdot 2^m$

Određivanje optimalne MD strukture: -za funkciju od **n** varijabli broj članova matrice je uvijek 2^n, jer za svaku od 2^n kodnih riječi varijabli **x** upisujemo vrijednost funkcije 0 ili 1. Broj logičkih vrata jednak je zbroju redaka i stupaca matrice i može se pokazati da je minimalan kada je R=S ($2^d = M \cdot 2^m$).

14.3. Memorije sa samim očitavanjem:

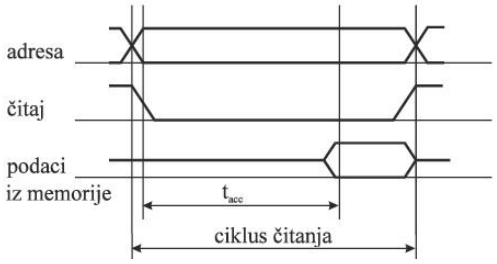
-MD struktura kao ROM
-kompatibilnost s radom računala
-tehnološke osobine ROM komponenti
-vremenski dijagram čitanja podataka

MD struktura kao ROM:
Integracijom MD strukture dobivamo memoriju. ROM je MD struktura sa programabilnom 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 isključivanjem 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 električno byte po byte
FEPROM – lebdeća vrata, brisanje električno blok po blok

Vremenski dijagram čitanja podataka:



Vrijeme čitanja podataka: -vrijeme koje protekne od trenutka postavljene adrese i signala čitanja do trenutka kada na izlazu dobijemo očitane podatke.

15. PROGRAMABILNE LOGIČKE STRUKTURE:

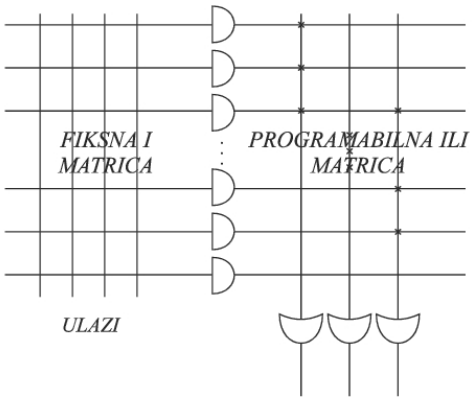
15.1. Definicija programabilne logičke strukture:

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

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

Vrste PLS ovisno o programabilnosti matrice:
-FPLA – programabilna I i ILI matrica
-GAL, PAL – programabilna I i fiksna ILI matrica

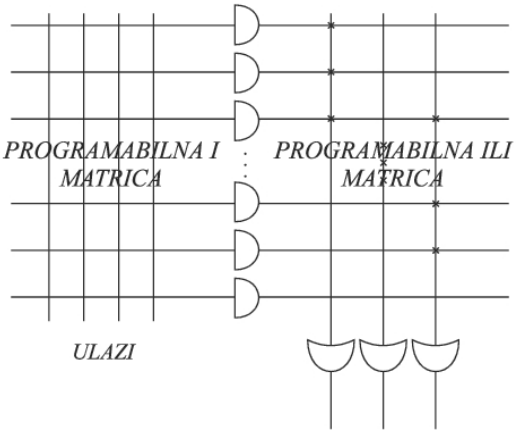
ROM struktura: -memorijska struktura kod koje je sadržaj upisan kod izrade. Struktura ROM memorije nije optimalna. Programabilna ILI matrica samo učinkovito povezuje minterme funkcije u PDNO. Mintermi su realizirani fiksnom I matricom, što onemogućuje minimizaciju i korištenje MDNO.



15.2. FPLA (Field Programmable Logic Array):

-FPLA struktura
-prednosti i mane FPLA strukture

FPLA struktura: -struktura sa programabilnim I i ILI matricama.



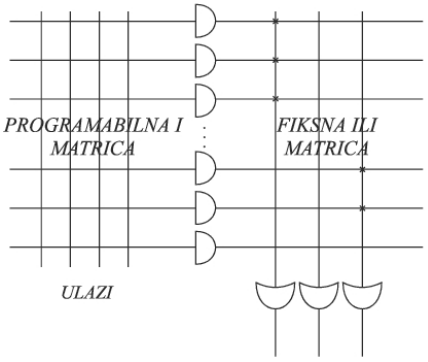
Prednosti: -omogućena minimizacija pojedine funkcije i izbor elementarnih članova. Izračunamo MDNO funkcije, elementarne članove programiramo u I matricu, te ih povežemo u konačni oblik ILI matricom.

Mane: -dvije programabilne matrice zahtijevaju dvostruko više pomoćnog sklopovlja, skuplji integrirani krugovi, troši se previše energije.

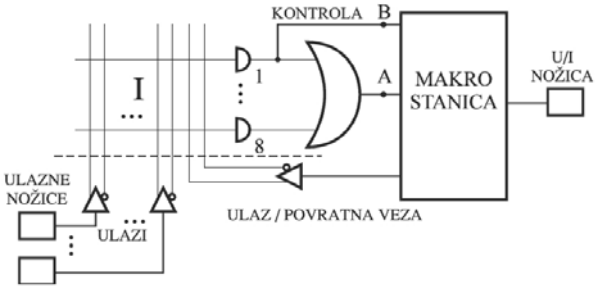
15.3. GAL (Generic Array Logic):

-PAL struktura
-GAL struktura
-koncept makro ćelije

PAL struktura: -struktura sa fiksnom ILI i programabilnom I matricom. PAL su komponente s PROM matricom u TTL tehnologiji. Potreba za različitim odnosom veličine funkcije (broj elementarnih članova) i broja funkcija (broj izlaza) za istu veličinu matrice dovodi do pojave čitave familije različitih komponenti.



GAL struktura: -struktura ostvarena primjenom CMOS tehnologije i EPROM upravljačkih bitova. Kod GAL komponenti moguće je brisanje i ponovno upisivanje I matrice. Svega dvije komponente zamjenjuju ranije PAL-ove, te donose nove mogućnosti kroz programabilne izlazne sklopove.



Fleksibilnost GAL-a proizlazi iz koncepta makro ćelija i povratnih veza.

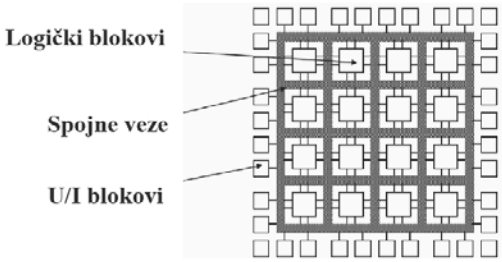
Makro ćelije (stanice): -programabilni izlazni sklopovi korišteni kod GAL strukture. Svaka makro stanica može biti konfigurirana na četiri načina: -kao sekvencijalni izlaz preko D bistabila, -kao kombinacijski ulaz/izlaz, -kombinacijski izlaz i -kombinacijski ulaz.

15.4. CPLD (Complex Programmable Logic Device):

-struktura CPLD
-jezici za definiranje sklopovlja (HDL)

Struktura CPLD:

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



Jezici za definiranje sklopovlja:
Za definiranje sklopovlja koristimo *VHDL* i *Verilog* (industrijski standard).

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 sekvencijalne sklopove
-obrazložiti potrebe za pamćenjem sekvencijalnog sklopa

Sekvencijalni sklopovi:
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 jedina 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.

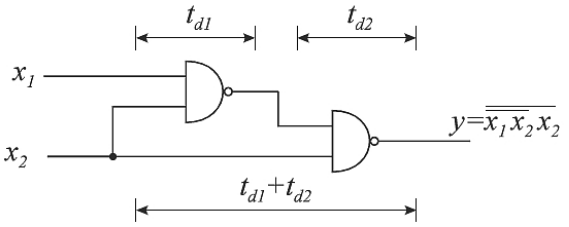
Tehničko ostvarenje pamćenja: -mehanizmi koji dovode do kašnjenja mogu biti upotrijebljeni za realizaciju pamćenja, a osim toga, za pamćenje se mogu iskoristiti druge modifikacije energije (npr. EPROM) i modifikacije materije (npr. ROM).

17. RAD SKLOPA U DISKRETNOM VREMENU:

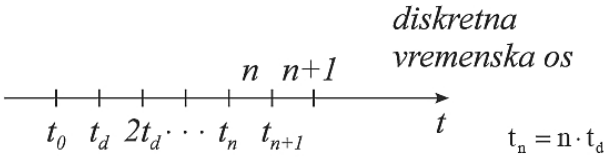
17.1. Diskretno vrijeme:

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

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



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

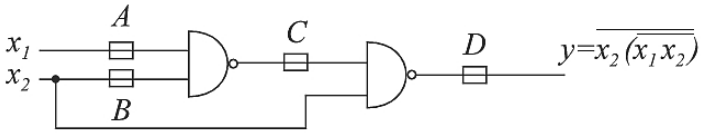


Komentar diskretnog vremena s obzirom na informacijski volumen, teorem uzorkovanja:
Diskretno vrijeme definira da moramo pročitati 2B signalnih elemenata u sekundi da bi pročitali informaciju, odnosno predstavlja minimalnu duljinu trajanja signala koja uopće može proći kroz naš sklop, a to je 2B signalna elementa = $1/t_d$ signalnih elemenata. Konačnost perioda diskretnog vremena je ekvivalentna konačnosti gornje granične frekvencije sustava s odnosom $1/f_g=2t_g$

17.2. Rad sklopa u diskretnom vremenu:

-stabilno i nestabilno stanje sklopa
-prikaz ponašanja sklopa

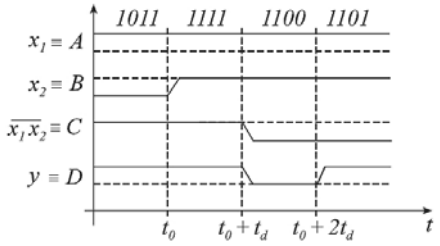
Stabilno i nestabilno stanje sklopa:
Mjerimo signale u točkama A, B, C, D, a u trenutku t_n kodna riječ od ABCD opisuje trenutno stanje sklopa. U nizu trenutaka sklop prolazi kroz niz stanja. Neka stanja su stabilna (iz njih izlazimo samo na vanjski poticaj), a neka su nestabilna (iz njih izlazimo samo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

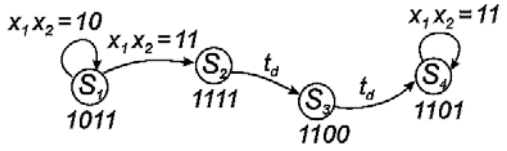


Prikaz ponašanja sklopa:

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	$t_0 + 2t_d$

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.
Posljedica prolaska kroz nestabilna stanja je u generiranju nepoželjnog impulsa 0 na izlazu sklopa u trenutku t_d. Ovaj impuls može dalje izazvati nepravilan rad čitavog digitalnog sustava.



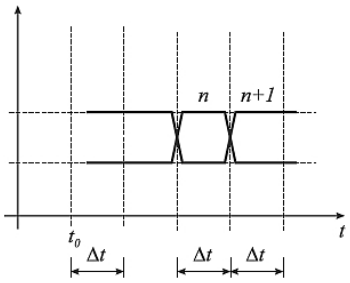


17.3. Sinkroni sklopovi:

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

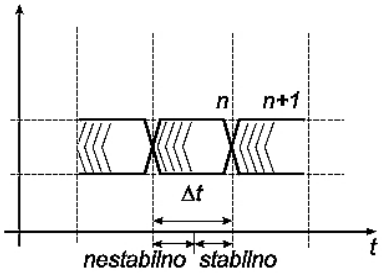
Proizvoljni period diskretnog vremena i sklopovi za pamćenje:

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. Postupak pisanja i čitanja podatka treba biti zanemarivo kratak u odnosu na period diskretnog vremena.



Nestabilni i stabilni period rada sklopa: -vrijednost neke varijable crtamo tako da naznačimo da se promjena događa u diskretnom trenutku. Unutar perioda nema promjene bez obzira da li je vrijednost varijable 0 ili 1. Nestabilni period je onaj u kojem promjena još nije završila, a stabilni je onaj u kojem su promjene završile i više nisu moguće.

Vremenski dijagram rada sinkronog sklopa:



18. BISTABIL KAO SKLOP

18.1. Osnovni sklop za pamćenje, elementarni RS bistabil:

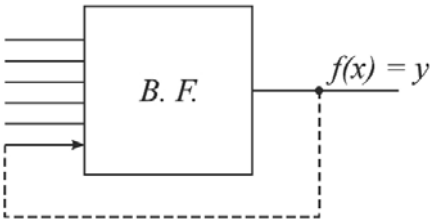
-definirati funkciju bistabila
-povratna veza i njen značaj
-elementarni RS bistabil

Definirati funkciju bistabila:

Bistabil je osnovni memorijski element za pamćenje vrijednosti Booleove 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 $\overline{Q}$.

Povratna veza i njen značaj:

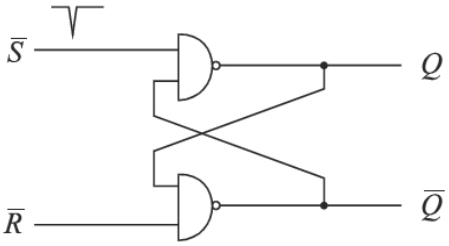
Bistabil je sklop koji je ostvaren preko povratne veze, tj. kod njega se vrijednost izlaza dovodi na ulaz. Rad je bitno određen kašnjenjem na sklopu, tako da će trenutna vrijednost izlaza djelovati nakon vremena kašnjenja sklopa.



Elementarni RS bistabil:

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)



18.2. Sinkronizacija bistabila s diskretnim vremenom:

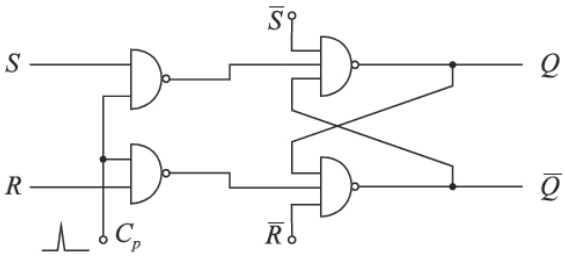
-informacija i trenutak prijelaza
-RS bistabil sinkroniziran impulsom
-sinkroni i asinkroni RS ulazi

Informacija i trenutak prijelaza, RS bistabil sinkroniziran impulsom:

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.

Sinkroni i asinkroni RS ulazi:

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 $\overline{S}$ koji služe za postavljanje bistabila u početno stanje i neovisni su o taktnom ulazu C_p.

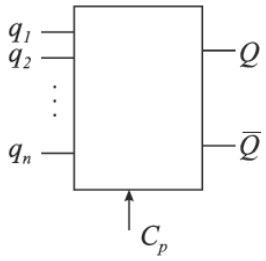


18.3. Bistabil kao funkcionalni blok:

-model bistabila
-definirati tablice prijelaza
-definirati funkciju prijelaza

Model bistabila:

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



Tablica prijelaza:

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} stanje bistabila u sljedećem trenutku.

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

Skraćena tablica prijelaza s lijeve strane ima kodne riječi ulaza **q^n** poredane prirodnim binarnim nizom, sve u sadašnjem, n-tom trenutku, s desne strane Q^{n+1} (stanje u sljedećem trenutku), kao funkciju sadašnjeg stanja Q^n.

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

Funkcija prijelaza:

Bistabil možemo zapisati i funkcijom prijelaza:

$Q^{n+1} = f(Q, q_1, ..., q_m)^n = \left(G_1 Q \vee G_2 \overline{Q}\right)^n$, gdje G_1 opisuje rad bistabila kad je sadašnje stanje 1, a G_2 opisuje rad bistabila kad je sadašnje stanje 0.

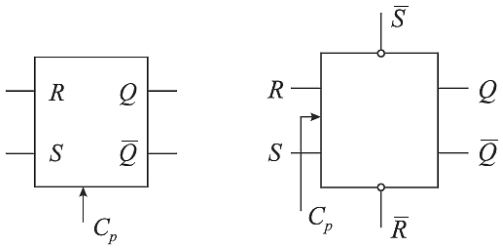
Sljedeće stanje je funkcija sadašnjeg stanja i ulaza. To je algebarski oblik sličan Booleovoj funkciji, ali za razliku od nje ima vremenski odnos između lijeve i desne strane tako da se s obje strane može pojaviti ista varijabla.

18.4. Standardni bistabili:

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

RS bistabil: 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 zadan je skraćenom i potpunom tablicom prijelaza:

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

$(R\ S\ Q)^n$	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

Funkcije prijelaza:

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

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

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

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

Q^{n+1}:

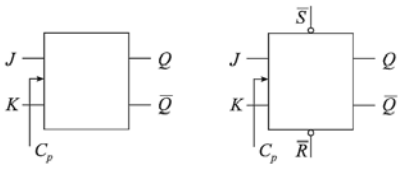
S^n \ R^n	X	X	1	1
			1	

Q^n

Uvjet RS=0 osiguravamo izvana!

JK bistabil: definicija, tablice, funkcije:

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 sa ulaza jer u skraćenoj tablici prijelaza ima sve četiri moguće funkcije ovisnosti o prethodnom stanju.

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

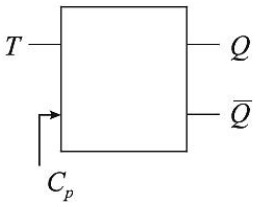
$$G_1 = \overline{K} \qquad G_2 = J$$

Q^{n+1}:

K^n \ J^n	1		G_1	G_2
	1		1	

Q^n

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

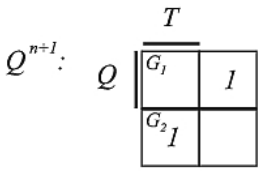


T^n	Q^{n+1}
0	Q^n
1	$\overline{Q}^n$

(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.

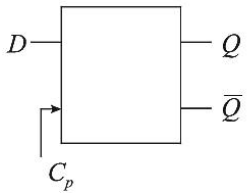
Funkcija prijelaza:



$$Q^{n+1} = \left(\overline{T} \cdot Q \vee T \cdot \overline{Q}\right)^n \qquad \left[G_1 = \overline{T} \qquad G_2 = T \qquad G_1 = \overline{G_2}\right]$$

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

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

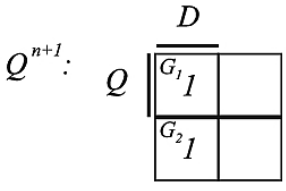


D^n	Q^{n+1}
0	0
1	1

(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:



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

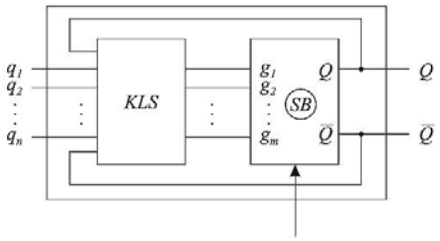
$$\left[G_1 = D \qquad G_2 = D \qquad G_1 = G_2\right]$$

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

Blok shema:



Značajke modela realizacije općeg bistabila:
Svojstvo je modela da su izlazi standardnog ujedno i 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, i signale potrebne da bi standardni bistabil obavio potrebne prijelaze.

Osnovna formula realizacije općeg bistabila:

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

Navesti metode poimence:
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 i metoda za D bistabil.

19.2. Metoda rekonstrukcije:

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

Metoda rekonstrukcije: - pogodna je za sve bistabile, a jedina je upotrebljiva za RS bistabile (zbog uvjeta RS=0) i T bistabile (zbog komplementarnih G_1 i G_2).

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

-potpunu tablicu prijelaza općeg bistabila 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.

Tablica rekonstrukcije za standardne bistabile:

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.

Primjena metode rekonstrukcije: -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 izjednačavanja

Metoda izjednačavanja: - pogodna je za JK bistabil. Ova metoda se provodi tako da izjednačimo funkcije prijelaza općeg (aplikativna jednadžba) i standardnog bistabila *(karakteristična jednadžba):*

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

Na osnovi strukturalne sličnosti slijedi:

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

Za JK bistabil vrijedi:

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

Odakle slijedi:

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

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

Primjena metode izjednačavanja: -koristi se za JK bistabil.

19.4. Metoda za D bistabil:

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

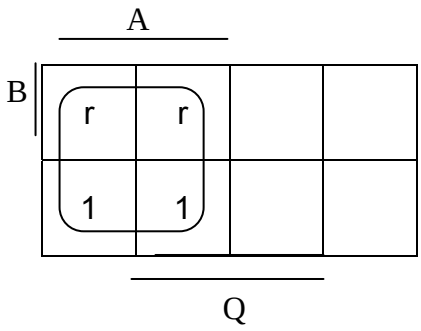
Metoda za D bistabil je istovremeno metoda rekonstrukcije i metoda izjednačavanja. Temelji se na funkciji prijelaza $Q^{n+1}=D^n$, tako da je potpuna tablica općeg bistabila numerički identična tablici istine za KLS. Postupak se provodi tako da prema potrebi minimiziramo funkciju za D^n ako radimo sa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

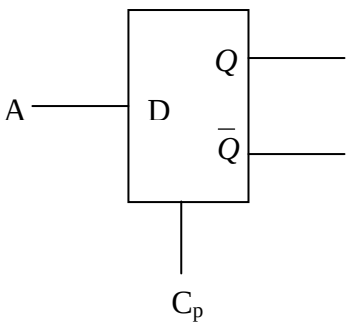
Primjer:

A	B	Q^{n+1}
0	0	0
0	1	0
1	0	1
1	1	r

$(A$	B	$Q)^n$	Q^{n+1}
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	r
1	1	1	r

D:





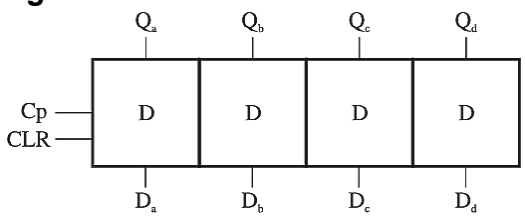
20. SLOŽENI SKLOPOVI S BISTABILIMA

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 ulaz C_p. Izvode se od 4 i 8 bistabila, i najčešće raspolažu asinkronim CLR ulazom kojim se dovode u početno stanje 0.
Dok je taktni signal C_p u 1 sklop prati kodne riječi na ulazu, kada taktni signal C_p dosegne 0 sklop pamti dosegnutu vrijednost.

Principijelna shema registra:



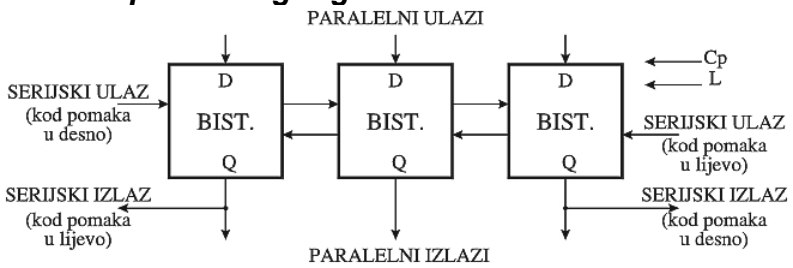
Primjena registra: -služi za pamćenje cjelovitih kodnih riječi

20.2. Pomačni registar:

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

Pomačni registar kao sklop:
Sklop 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.

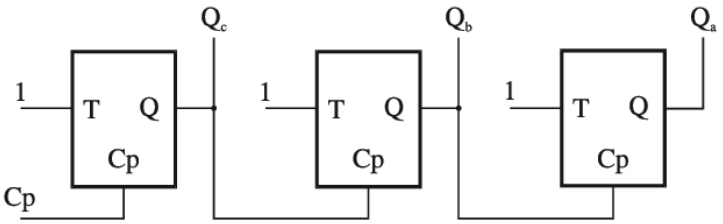
20.3. Brojilo:

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

Brojilo kao sklop:

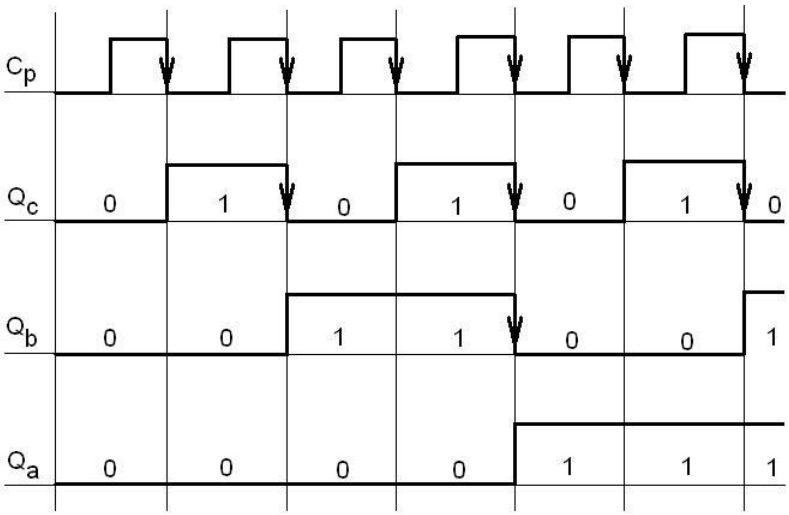
Brojilo je sklop koji se sastoji od više T ili JK bistabila spojenih tako da niz uzastopnih taktnih impulsa uzrokuje generiranje stanja brojila u prirodnom binarnom nizu, tj. prijelaz u sljedeće stanje nastaje u trenutku nastupanja taktnog signala. Mogu biti *sinkrona* (kada su svi taktni ulazi zajednički, a prijelazi ovise o stanjima ostalih bistabila), ili *asinkrona* (kada je izlaz jednog spojen na ulaz sljedećeg bistabila).

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. Koristi se još za mjerenje vremena i mjerenje frekvencije.



III. TREĆINA GRADIVA

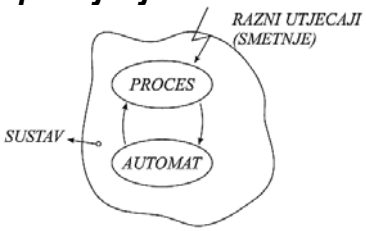
21. DIGITALNI AUTOMAT:

21.1. Sustav s upravljanjem:

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

Sustav s upravljanjem: -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.

Skica strukture sustava s upravljanjem:



Funkcija cilja: -ostvariti funkcionalnost procesa na ekonomičan način. To je dakle, funkcija ekonomičnosti.

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 1. dio:

-definirati determiniranost
-definirati specificiranost
-definirati sinkronost

Determiniranost: -jednoznačno obavlja svoju funkciju.

Specificiranost: -*potpuna*: mogući su svi nizovi ulaznih događaja (očekuje proizvoljni niz ulaza).
-*nepotpuna*: mogući su samo neki nizovi ulaznih događaja.

Sinkronost: -diskretno vrijeme definirano taktnim signalom.

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 definiram matematičkim elementima, skupovima i funkcijama.

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

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

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

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

Skup S:
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 δ: -funkcija prijelaza koja određuje sljedeća stanja automata na osnovi sadašnjeg stanja i sadašnjeg ulaza:

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

Funkcija λ: -funkcija izlaza koja određuje sadašnji izlaz automata na osnovu sadašnjeg stanja (Moore) ili sadašnjeg stanja i sadašnjeg ulaza (Mealy). Razlikujemo Mealy i Moore model automata:

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

Definicija kartezijevog produkta: -Produkt dvaju skupova, tj. skup koji sadrži sve uređene parove članova tih skupova.
Kažemo da funkcija prijelaza obavlja preslikavanje iz skupa svih parova SxU u skup stanja S.

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

Tablica prijelaza i izlaza za Mealy-ev model automata:

	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		

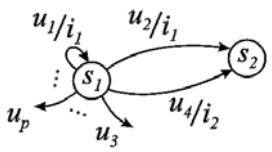
Svako mjesto u tablici odgovara jednom paru **s**, **u**.

Tablica prijelaza i izlaza za Moore-ov model automata:

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

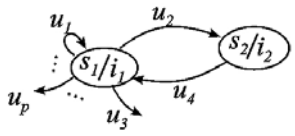
Izlazi ovise samo o stanju **s**.

Usmjereni graf za Mealy-ev model automata:



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

Usmjereni graf za Moore-ov model automata:



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

22.3. Sinteza automata:

-definirati faze sinteze
-definirati korake za svaku fazu sinteze

Faze 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:

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)

23. ZADAVANJE AUTOMATA:

23.1. Pristupi zadavanju automata:

-definirati tri transformacije
-navesti konkretne postupke zadavanja

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 ulaznih sekvenci koje izazivaju pojavu nekog simbola na izlazu. Govorimo o transformaciji ulazne sekvence na izlazni simbol.
- 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).

Konkretni postupci zadavanja:

Automat možemo zadati na tri načina:
- potpunim stablom
- tablicom prijelaza i izlaza
- regularnim izrazom

23.2. Vrste ulazne sekvence:

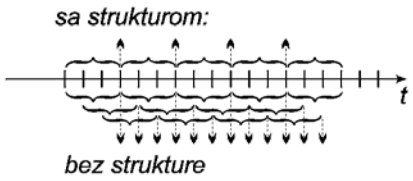
-definirati vrste ulazne sekvence
-skicirati vremenski dijagram odlučivanja

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 odučiti hoće li preklopljen sekvenca djelovati kao sekvenca koja nije preklopljena. Automat mora u svakom trenutku 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.

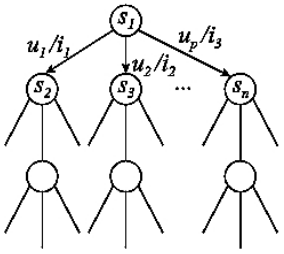
Vremenski dijagram odlučivanja:



23.3. Postupak zadavanja korak po korak:

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

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).



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 svaki 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 kao za sekvencu sa strukturom.

24. EKVIVALENTNOST AUTOMATA:

24.1. Odnosi jednakosti među automatima:

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

Vrste jednakosti među automatima:

Automati mogu biti:

ISTI: - rade istu funkciju, isti su po strukturi
RAZLIČITI: - rade različitu funkciju, različiti su po strukturi
EKVIVALENTNI: - rade istu funkciju, a različiti su po strukturi

Definicija minimalnosti automata:

Minimizacija se provodi tako 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.

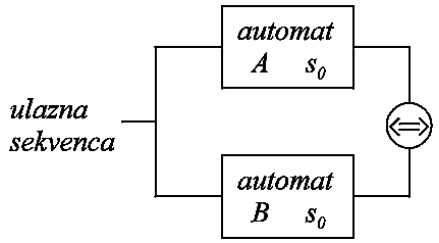
Objašnjenje razlike broja stanja:

Ekvivalentni automati se ne mogu razlikovati u skupovima ulaznih i izlaznih simbola, jer neće biti zadovoljen zahtjev istovjetnosti ulaznih i izlaznih sekvenci. Oni se mogu razlikovati samo u skupu stanja, a minimalan je onaj s najmanjim brojem stanja. Stanja koja nepotrebno razlučuju ulazne sekvence, jer za njih automat donosi istu odluku, nazivaju se **ekvivalentna stanja**.

24.2. Definicija ekvivalentnosti automata:

-blok shema testa
-definicija ekvivalentnosti automata

Blok shema testa:



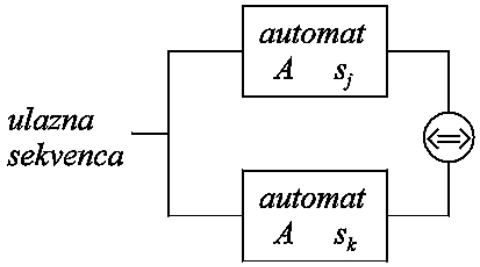
Definicija ekvivalentnosti automata:

- **Dva automata** (A i B) **su ekvivalentna** ako počnu raditi iz početnog stanja s_0 te za istu **proizvoljnu** sekvencu na ulazu dadu **istu** sekvencu na izlazu.

24.3. Definicija ekvivalentnosti stanja:

-blok shema testa
-definicija ekvivalentnosti stanja

Blok shema testa:



Definicija ekvivalentnosti stanja:
- **Dva stanja automata A** (s_j i s_k) **su ekvivalentna** ako automat počne raditi iz jednog pa iz drugog stanja te za istu **proizvoljnu** sekvencu na ulazu dade 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 redci 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 redci u tablici prijelaza automata isti.
Ovaj uvjet osigurava da u svakom paru testova, iz promatranih stanja automat prijeđe u isto stanje. 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 minimizaciju primitivne tablice
-mane jednostavnog postupka minimizacije primitivne tablice

Postupak minimizacije:
Da bi se dobio minimalan automat, dovoljno je otkriti podskupove ekvivalentnih stanja, te sva stanja nekog podskupa zamijeniti s jednim stanjem. Postupci za pronalaženje podskupova ekvivalentnih stanja su: postupak primitivne tablice, Hufmann-Mealy algoritam i Paul-Unger algoritam (tablica implikanata).

Postupak minimizacije primitivne tablice se provodi 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.

Postupak primitivne tablice se provodi u sljedećim koracima:
1.) Traže se stanja s istim redcima 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 sve dok ima ekvivalentnih stanja

Mane jednostavnog postupka minimizacije primitivne tablice: -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.

25. NAPREDNI POSTUPCI MINIMIZACIJE AUTOMATA:

25.1. HM algoritam:

-definicija klasa i zatvorenosti
-postupak minimizacije HM algoritmom

Definicija klasa i zatvorenosti: -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 sve imaju iste prijelaze u klase.

Postupak minimizacije HM 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 sa jednim stanjem iz pojedine klase te ispišemo tablicu minimalnog automata.

25.2. PU algoritam:

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

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 i 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)

Tablica implikanata:

Skupove $\mathbf{s_p}$ formiramo sistematski 2 po 2 stanja (ispitujemo ekvivalentnost svakog sa svakim).

Tablica implikanata je trokutasta matrica bez dijagonale jer vrijedi:

$s_i \Leftrightarrow s_i \rightarrow$ svako stanje je ekvivalentno samo sebi pa ne trebamo gledati dijagonalu

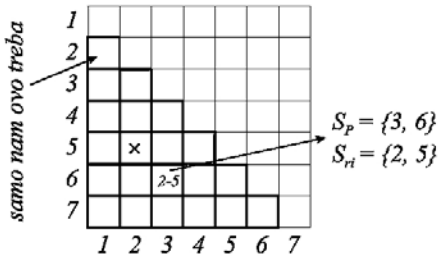
$s_i \Leftrightarrow s_j \Rightarrow s_j \Leftrightarrow s_i \rightarrow$ zbog komutativnosti ekvivalencije ne trebamo cijelu matricu nego samo jednu njenu trokutastu polovicu.

Tablica ima **n-1** redaka i stupaca. Svako mjesto odgovara jednom paru s_i, s_j.

U mjesta tablice upisujemo implikante, a to su svi skupovi $\mathbf{s_{ri}}$.

Postupak minimizacije provodimo:

1.) Formiramo trokutastu matricu (n-1)x(n-1)
2.) Upišemo implikante:
$s_{ri} \Rightarrow$ zadovoljen nužan, ne i dovoljan uvjet
X $\Rightarrow$ nužan uvjet nije zadovoljen (neekv.)
V $\Rightarrow$ zadovoljen nužan i dovoljan uvjet (ekv.)

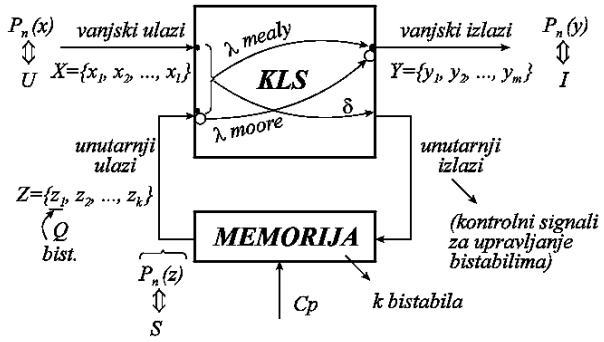


26. STRUKTURNA SINTEZA AUTOMATA:

26.1. Model realizacije automata:

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

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



Memorija automata se sastoji od k bistabila. Izlazi memorije, k varijabli iz skupa Z, su izlazi pojedinih bistabila memorije. Svaka kodna riječ varijabli Z predstavlja jedno stanje, dakle, svu informaciju koju memorija može dati o svim prethodnim događajima na ulazu u automat. Memorija ima k ili 2k ulaza, ovisno o vrsti bistabila. To su signali koji dolaze sa KLS i koji određuju sljedeće stanje memorije. Stanje memorije se mijenja u trenutku nastupa aktivnog dijela taktnog signala, koji je zajednički za sve bistabile. Memorija automata je zapravo registar.

KLS ima vanjske ulaze, električne varijable iz skupa X na koje dolaze kodne riječi kojima su kodirani ulazni simboli, te vanjske izlaze, električne varijable iz skupa Y na kojima se generiraju kodne riječi kojima su kodirani izlazni simboli.

KLS također raspolaže unutarnjim ulazima na koje dolaze varijable iz skupa Z, a to su spomenuti izlazi bistabila memorije, te unutarnjim izlazima preko kojih upravlja prijelazima memorije.

Namjena signala:
Kontrolni signali su unutarnji izlazi koje KLS generira na osnovu vanjskih i unutarnjih 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 sa apstraktnim automatom se ostvaruje kroz kodiranje ulaznih i izlaznih simbola, te kroz kodiranje stanja automata. Stoga je prvi korak strukturne sinteze, na osnovi poznavanja modela realizacije, kodiranje skupova ulaznih i izlaznih simbola.

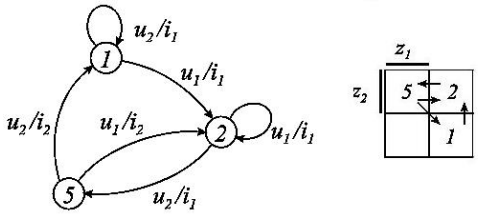
26.2. Kodiranje automata:

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

Kodiranje ulaza i izlaza ovisi o okolini automata. Preko ovih kodnih riječi automat komunicira s izvorištem i odredištem informacije pa kodovi moraju biti usklađeni. Stoga često nemamo slobodu kodiranja već su nam ulazni i izlazni kôd zadani.

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 se stanje obično kodira kodnom riječi 0.



26.3. Tablica automata s kodovima:

-transformacija 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 tj. članove Kartezijevog produkta S×U.

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

Tablica kao cjelina:
Nakon upisa kodova (umjesto simbola i stanja) kodne riječi su poslagane u prirodnom binarnom nizu. Ukupna kodna riječ zz...xx čini lijevu stranu tablice. Desnu stranu tablice prvo čine prijelazi (kodne riječi bistabila u sljedećem diskretnom trenutku), zatim kodne riječi izlaza u sadašnjem trenutku. O lijevoj strani ovisi i svaki pojedini bit desne strane.

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								

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

	n			n+1		n
	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

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, logička vrata), integrirani krugovi srednje razine integracije (multiplekseri, demultiplekseri, registri) i programabilne logičke strukture s ugrađenim D bistabilom u izlaznoj makro ćeliji (GAL).

Kriteriji sinteze konvencionalnog automata:
Sinteza memorije se provodi izborom vrste i izračunavanjem broja bistabila po kriteriju jednoznačnosti kodiranja stanja automata. Sinteza kombinacijske logičke strukture (KLS) provodi se korištenjem prepoznatih dijelova kodirane tablice prijelaza i izlaza automata: - potpunih tablica prijelaza bistabila memorije i tablica istine za izlazne varijable.
KLS automata je zapravo sastavljena od KLS pojedinih općih bistabila i KLS kojima realiziramo Booleove funkcije izlaznih varijabli. Sintezu svakog pojedinog dijela obavljamo korištenjem ranije razrađenih postupaka sinteze općih bistabila i realizacije Booleovih funkcija.

Kriteriji sinteze MDD strukture:
Kod realizacije automata multipleksersko-demultiplekserskom strukturom i registrom D bistabila (MDD) biramo veličinu multipleksera i demultipleksera, te registar s D bistabilima.
Veličina multipleksera i demultipleksera određuje se po kriteriju kvadratičnosti matrice. Broj multipleksera jednak je broju bistabila **k** uvećanom za broj izlaznih varijabli **p**:

$$M = k + p$$

Ukupni broj adresnih ulaza u strukturu **n** jednak je broju adresnih ulaza **d** demultipleksera i broju adresnih ulaza **m** multipleksera, a treba biti jednak broju bistabila **k** uvećanom za broj ulaznih varijabli ℓ:

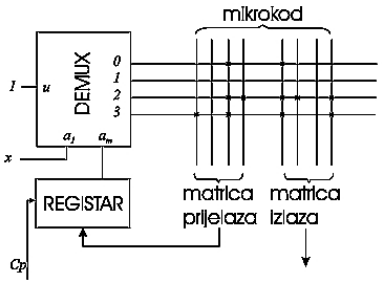
$$n = d + m = k + \ell$$

27. AUTOMATI I ALGORITMI:

27.1. Programabilni automat:

-Wilkiesov model automata
-mikroprogramirani automat
-primjena mikroprogramiranog automata

Wilkiesov model automata: -model mikroprogramiranog automata realiziranog MDD strukturom.



Mikroprogramirani automat: -automat realiziran MDD strukturom. Sadržaj jednog retka matrice naziva se mikroinstrukcija, a sadržaj matrice mikroprogram, a tako realizirani automat mikroprogramirani automat.
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: - upravlja radom računala (upravlja radom sabirnice i ALU jedinicom).

27.2. Algoritam:

-definirati algoritam
-primjena algoritma
-istovjetnost algoritma i automata

Algoritam: -skup transformacija ili instrukcija čijom se primjenom u konačnom broju koraka, može doći do rješenja bilo kojeg problema iz promatranog skupa (klase) problema.

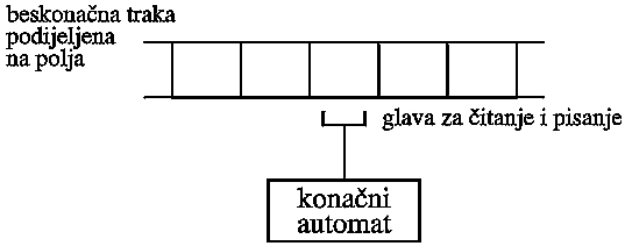
Primjena algoritma: -koristi se u rješavanju problema za koji se može dobiti točno rješenje u konačnom broju koraka.

Istovjetnost algoritma i automata:
Svaki se algoritam može zamisliti kao stroj (automat) koji izvodi zadani niz instrukcija. Kažemo da svakom algoritmu pripada stroj, a svakom stroju algoritam. Stroj i algoritam su samo dva načina promatranja istog problema: *kako doći do rješenja u konačnom broju koraka*.

27.3. Turingov stroj:

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

Turingov stroj je najsnažniji algoritam i stroj koji je do sada poznat u teoriji algoritama. On je model koji raspolaže beskonačnom trakom (s lijeva i desna), glavom za čitanje/upisivanje na traku (R/W), mogućnošću pomaka glave lijevo ili desno, te konačnim digitalnim automatom koji upravlja radom Turingovog stroja.



Jezik petorki:
Turingov stroj se zadaje kroz automat kojemu se rad definira petorkama:

$$\langle S_i, T_i, T_j, \begin{vmatrix} S \\ L \\ R \end{vmatrix}, S_j \rangle$$

Alfabet T koristimo kao skup ulaznih i izlaznih simbola. T_i je simbol očitan s trake (ulazni), T_j je simbol upisan na traku (izlazni). S_i je sadašnje, a S_j buduće stanje. Svaka petorka definira izlaz i prijelaz automata za jedan par sadašnjeg stanja i ulaza, tj. jedan redak tablice prijelaza i izlaza automata, odnosno jedan redak matrice mikroprogramiranog automata.
Jedina razlika je u dodavanju zapovjedi glave za čitanje i pisanje, koja može bili lijevo L (Left), desno R (Right) ili stani S (Stop).

Primjena Turingovog stroja je u pretpostavci mogućnosti realiziranja “bilo kojeg automata“ u izračunu svih parcijalno-rekurzivnih funkcija.

28. AUTOMATI I JEZICI:

28.1. Značaj analize jezika:

-ekvivalentnost programabilnih strojeva (algoritama)
-modeliranje procesa postizanja rješenja
-prepoznavanje pripadnosti riječi jeziku L

Ekvivalentnost programabilnih 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)
- određene strukture za konkretan jezik

28.2. Kompleksnost algoritama:

-mjera kompleksnosti
-vrste kompleksnosti

Mjera kompleksnosti: -vrijeme postizanja rješenja u smislu izvršenja jednog koraka Turingovog stroja (TM). Pokušavamo 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 i da li je ostvarivo raspoloživom tehnologijom.
Problem kompleksnosti jednostavno rješavamo modelima na bazi Turingovog stroja.

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
Eksponencijal na:	$O(k^n)$	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)

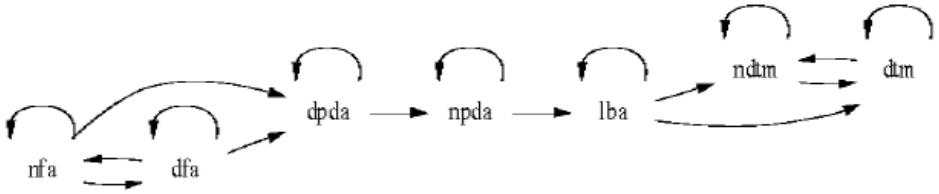
Nekad u konačnom O() vremenu možemo odgovoriti samo na jedno ili ni na jedno pitanje. Formalno, pitamo da li niz $x \in L$ je TR i pritom nije co-TR.

TR	co-TR	ODLUKA
0	0	Ne postoji
0	1	Ne postoji
1	0	Ne postoji
1	1	Postoji, eksponencijalno

28.4. Taksonomija automata i jezika:

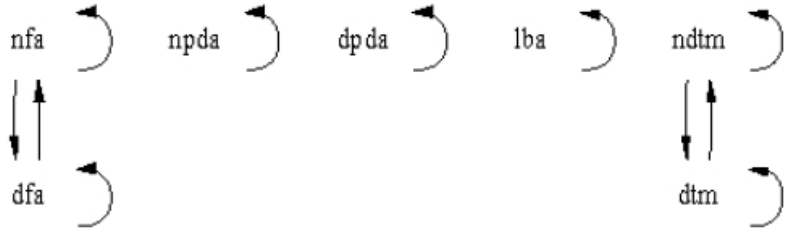
-odnos snage vrsta automata
-ekvivalentnost vrsta automata
-odnos automata i jezika

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:



Odnos automata i jezika:

Jezik L je skup sekvenci izgrađenih od članova skupa simbola $\sum$.
Za neki skup simbola $\sum$ jezika može biti beskonačno, jer karakteristična sekvenca može biti beskonačna.
Nad jezicima su definirane operacije: unija, presjek, razlika, simetrična razlika, pripajanje, eksponenciranje, iteracija i negacija.
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.

29. ALGEBRA DOGAĐAJA:

29.1. Elementarni složeni događaji:

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

Elementarni događaj: -nedjeljiv događaj, dogodi se kao cjelina.

Elementarni događaj i diskretno vrijeme: -dogodi se samo JEDAN događaj u diskretnom trenutku.

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

Ulaz automata: - ulazi u automat mu daju informaciju o okolini
- ulazni simboli dolaze u vremenskim nizovima
- to su sekvence u diskretnom vremenu
- te smo sekvence nazvali 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 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)
- zato se u praksi ne koristi

ITERACIJA: (Kleenee*)

- događaj koji se desi kad se događaj pod iteracijom desi PROIZVOLJAN BROJ puta, čak i NI JEDNOM
- to je skup svih sekvenci koje nastaju proizvoljnim ponavljanjem događaja pod iteracijom
- iteracija uključuje i praznu sekvencu Λ:

$$(R_1)^* = (\Lambda \vee R_1 \vee R_1R_1 \vee R_1R_1R_1 \vee R_1R_1R_1R_1 \vee ...)$$

$$(R_1 \vee R_2)^* = (\Lambda \vee R_1 \vee R_2 \vee R_1R_1 \vee R_1R_2 \vee R_2R_1 \vee R_2R_2 \vee R_1R_1R_1..).$$

Regularni izrazi: - 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

Pristup zadavanju automata s RI:

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 trenutcima, 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 dva zadnja 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 tri zadnja simbola prva tri simbola tražene sekvence (ako takvih ima)
- itd. do ispisa tražene sekvence

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 **prijelazima** automata, a to su stanja iz kojih idemo u prijelaze.
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 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-ov automat akceptorski izlazni simbol dodijelimo završnom **osnovnom** mjestu regularnog izraza.
Za MEALY-ev automat 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 redukciju 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 prijelaze odredimo preko ulaznih simbola.

OBVEZNI POJMOVI (NULTA PITANJA)

1.) Suma po modulu i primjer po želji:

Grupa koja se sastoji od skupa **F** i operacije $\oplus$:

F={0,1,2,…,m-1}; m≥2; m$\in\mathbb{N}$; a $\oplus$ b=c=Rez$\left(\frac{a+b}{m}\right)$ gdje je **m** modul, a operacija $\oplus$ zbrajanje po modulu **m** definirana kao ostatak (Rez) cjelobrojnog dijeljenja zbroja **a+b** s modulom **m**.

Primjer: -suma po modulu m=2

6 $\oplus$ 7= Rez$\left(\frac{6+7}{2}\right)$=Rez$\left(\frac{13}{2}\right)$=1

2.) Faktor izlaznog i ulaznog grananja:

Faktor izlaznog grananja: -govori nam koliko standardnih ulaza možemo spojiti na jedan standardni izlaz, a da pritom nisu narušene naponske razine 0 i 1.

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

Faktor ulaznog grananja: -govori nam koliko stvarni ulaz opterećuje izlaz na koji je spojen u odnosu na standardni ulaz za promatranu tehnologiju i varijantu izrade, odnosno koliko je standardnih ulaza integriranog kruga spojeno na istu nožicu kućišta. Ovaj faktor je najčešće 1.

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

3.) Postulati algebre logike i formule:

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 i negirane 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)

4.) Teoremi algebre logike i formule (bez dokaza):

1.) Apsorpcija za disjunkciju $x_1 \vee 1=1$
Jedinica disjunktivno vezana s nekim izrazom apsorbira taj izraz.

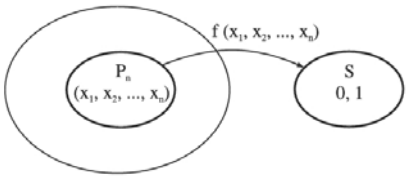
2.) Apsorpcija za konjunkciju $x_1 \cdot 0=0$
Nula konjunktivno vezana s nekim izrazom apsorbira taj izraz:

3.) Idempotentnost za disjunkciju $x_1 \vee x_1=x_1$
Disjunkcija nekog izraza sa samim sobom jednaka je tom izrazu. Koristi se sažimanje ili proširenje algebarskog izraza postojećim članom.

4.) Idempotentnost za konjunkciju $x_1 \cdot x_1=x_1$
Konjunkcija nekog izraza sa samim sobom jednaka je tom izrazu. Koristi se sažimanje ili proširenje algebarskog izraza postojećim članom.

5.) Definicija Booleove funkcije kao preslikavanja:

Definirati Booleovu funkciju kao preslikavanje:
Booleova funkcija $y=f(x_1, x_2, \ldots, x_n)$ je preslikavanje nadskupa P(x) u skup $S=\{0,1\}$.



6.) Definicija PDNO i formula:

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

7.) Definicija minterma i primjer po želji:

Minterm m_i i-tog retka tablice istine: -konjunkcija SVIH varijabli tako da su one koje u pripadnoj kodnoj riječi imaju vrijednost nula negirane, a one koje imaju vrijednost jedan nenegirane: $m_3(x_1, x_2, x_3)_{011} = \overline{x_1} \cdot x_2 \cdot x_3$

8.) Definicija PKNO i formula:

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

9.) Definicija maksterma i primjer po želji:

Maksterm M_i i-tog retka tablice istine: -disjunkcija SVIH varijabli tako da su one koje u pripadnoj kodnoj riječi imaju vrijednost jedan negirane, a one koje imaju vrijednost nula nenegirane: $m_3(x_1, x_2, x_3)_{011} = x_1 \vee \overline{x_2} \vee \overline{x_3}$

10.) Definicija MDNO:

MDNO (Minimalni disjunktivni normalni oblik): -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.

11.) Definicija MKNO:

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

12.) Definicija multipleksera i formula:

Definirati funkciju selektora/multipleksera:
Multiplekser je logički sklop koji na informacijski izlaz **i** propušta vrijednost onog od $n=2^m$ informacijskih ulaza **u** ($u_0,...,u_{n-1}$) čiji je redni broj prisutan u prirodnom binarnom obliku na **m** adresnih ulaza ($a_0,...,a_{m-1}$), pod uvjetom da je funkcija sklopa omogućena aktivnim signalima na kontrolnim ulazima.

$$i = \underset{j=0}{\overset{2^m-1}{\vee}} m_j \,\&\, u_j$$

13.) Definicija demultipleksera i formula:

Definirati funkciju dekodera/demultipleksera:
Demultiplekser je logički sklop koji vrijednost informacijskog ulaza **u** propušta na onaj od $n=2^m$ informacijskih izlaza **i** ($i_0,...,i_{n-1}$) čiji je redni broj prisutan u prirodnom binarnom obliku na **m** adresnih ulaza ($a_0,...,a_{m-1}$), pod uvjetom da je funkcija sklopa omogućena aktivnim signalima na kontrolnim ulazima.

$$i_j = m_j \,\&\, u$$

14.) Definicija diskretnog vremena:

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

15.) Definicija kartezijevog produkta:

-Produkt dvaju skupova, tj. skup koji sadrži sve uređene parove članova tih skupova.

16.) 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 dadu **istu** sekvencu na izlazu.

17.) Definicija ekvivalentnosti stanja:

Dva stanja automata A (s_j i s_k) **su ekvivalentna** ako automat počne raditi iz jednog pa iz drugog stanja te za istu **proizvoljnu** sekvencu na ulazu dade u oba testa **istu** sekvencu na izlazu.

18.) Nužan i dovoljan uvjet ekvivalencije:

Nužan uvjet ekvivalencije kaže da su dva stanja potencijalno ekvivalentna ako su im redci 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 redci u tablici prijelaza automata isti.
Ovaj uvjet osigurava da u svakom paru testova, iz promatranih stanja automat prijeđe u isto stanje. Time će automat u nastavku testa proći kroz istu trajektoriju stanja, pa je time osigurano da i ostali simboli izlazne sekvence budu isti.

19.) Definicija algoritma:

Algoritam je skup transformacija ili instrukcija čijom se primjenom u konačnom broju koraka, može doći do rješenja bilo kojeg problema iz promatranog skupa (klase) problema.

20.) Definicija iteracije (Kleenee*):

- događaj koji se desi kad se događaj pod iteracijom desi PROIZVOLJAN BROJ puta, čak i NI JEDNOM
- to je skup svih sekvenci koje nastaju proizvoljnim ponavljanjem događaja pod iteracijom
- iteracija uključuje i praznu sekvencu Λ:

$$(R_1)^* = (\Lambda \vee R_1 \vee R_1R_1 \vee R_1R_1R_1 \vee R_1R_1R_1R_1 \vee ...)$$

$$(R_1 \vee R_2)^* = (\Lambda \vee R_1 \vee R_2 \vee R_1R_1 \vee R_1R_2 \vee R_2R_1 \vee R_2R_2 \vee R_1R_1R_1...)$$