RAČUNARI I PROGRAMIRANJE

1. DIGITALNI I ANALOGNI SISTEM ZA OBRADU PODATAKA

U ovom naslovu daje se pojam računara i uvodi se podjela za računare na digitalne i analogne. Računar je uređaj koji vrši automatsku obradu nad brojevima. Mi zadajemo pomoću programa aritmetičke operacije koje treba da budu izvedene, kao i njihov redosljed. Onda se obrada izvršava automatski, bez intervencije čovjeka. Brojevi odnosno podaci prikazuju se u dekadnom ili binarnom brojnom sistemu. Znamo da računar vrši obradu i nad drugim vrstama podataka, recimo nad slovima. Takvi podaci prikazuju se na isti način kao i brojevi. Lako se zapaža da je maločas uvedena definicija računara nedovoljna. Postoje sasvim precizne matematičke definicije tog pojma. Tako se govori o računaru von Neumannovog tipa. Ili o Tjuringovoj mašini. Ili o konačnom automatu sa memorijom.

Podatak može da bude analogni (kontinualni) ili digitalni (diskretni). Za podatak se kaže da je u analognom obliku ako se radi o fizičkoj veličini čije se vrijednosti mijenjaju na neprekidan način, kako vrijeme protiče. Recimo, količina tečnosti u posudi ili napon u strujnom kolu. Za podatak se kaže da je u digitalnom obliku ako se radi o fizičkoj veličini čije su vrijednosti pojedinačne, odnosno jasno su odvojene jedna od druge. Tako da mogućih vrijednosti ima konačno mnogo ili prebrojivo mnogo. Prosto rečeno, vrijednosti su cijeli brojevi. Recimo, koliko primjeraka knjige ili koliko aviona. Ako sistem za obradu podataka radi sa analognim podacima onda je to jedan analogni sistem. A ako uređaj za računanje radi sa podacima koji su predstavljeni digitalno onda je to jedan digitalni uređaj.

O analognom računaru. Električne komponente podese se kako nalažu ulazni podaci zadatka koji se rješava. One se povežu kako odgovara jednačini koja se rješava. Tako da se onda rezultati dobijaju jednostavnim mjerenjem ili čitanjem napona na određenim mjestima u kolu. Razvoj analognih računara počeo je tridesetih godina, a napušten je tokom šezdesetih godina.

Znamo da se danas koriste jedino digitalni računari, pa se u nastavku govori samo o njima. Temeljna pitanja o hardveru riješena su tokom četrdesetih godina, a tokom pedesetih godina riješena su i ona o softveru.

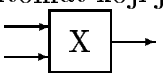
Znamo da su savremeni računari električni uređaji. Neka t označava vrijeme, a $V = V(t)$ napon. Za električni signal čija je amplituda $V = V(t)$ kontinualna po vremenu kaže se da je analogni signal. Nasuprot tome, ako amplituda signala može da se mijenja naglo ili prekidno onda je to impulsni signal. Posebno, ako impulsni signal ima samo nekoliko dozvoljenih nivoa amplitude onda je to digitalni signal. Najzad, kada su definisana samo dva različita naponska nivoa V_0 i V_1 onda je to binarni digitalni signal.

Vrijednosti bliske V_0 ili bliske V_1 tumače se kao logička nula, odnosno kao logička jedinica.

2. IDEJA O HARDVERU I SOFTVERU

Sredstva za obradu podataka mogu da se podijele u dvije klase: hardverska i softverska (programska). U osnovi hardvera je materijalni dio računara, dok osnovu softvera čini logika na osnovu koje su izgrađeni programi (koji upravljaju radom računara). Pod hardverom se podrazumijevaju čvrsti tj. fizički djelovi računara. Umjesto hardver ponekad se kaže tehnički sistem računara ili aparatni dio ili skup uređaja. Softver čini skup programa koji omogućavaju korišćenje hardvera. Umjesto softver ponekad se kaže programska podrška ili programski sistem računara ili matematičko obezbjeđenje. Savremeni računari su hardversko–softverski uređaji.

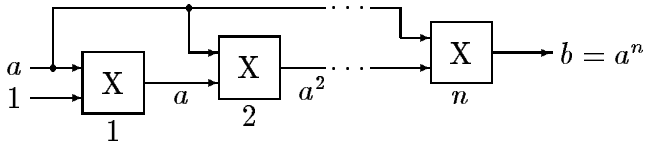
Pogledajmo primjer. Mali automat koji je u stanju da izvrši množenje dva broja prikazan je na slikama kao 

odnosno 

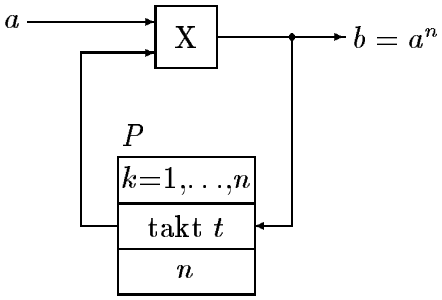
. A rješava se zadatak o računanju veličine $b = a^n$. Na Slici 1 predstavljeno je hardversko rješenje postavljenog zadatka. Na prost način je povezano n malih automata. Uočavamo odmah da mora da se konstruiše novi automat ako se n izmijeni. Drugim riječima, jasno je da su zadaci o računanju recimo $b = a^{10}$ i $b = a^{12}$ srodni, a mi ovako ne uspijevamo da jedan te isti automat služi za rješavanje dva ili više srodnih zadataka. U ovome se ogleda nedostatak hardverskog automata, odnosno prednost hardversko–softverskog.

Na Slici 2 je prikazano rješenje koje sadrži i softversku komponentu. Prisutan je i drugi mali automat P koji ima svoj program, pa može da broji od 1 do n. Takođe, može da taktuje, tako da u pravom trenutku prihvata izlaz iz 

odnosno predaje mu među–rezultat na njegov drugi ulaz. Takođe pretpostavljamo da P ima svoju memoriju, pa može da pamti n.



Slika 1



Slika 2

Hardver računara čine fizičke komponente i pridružena oprema, a računarski softver čine programi koji su napisani za računar.

3. ŠEMA RAČUNARA

Računar se sastoji iz pet djelova: ulazna, izlazna, memorijska, aritmetička i kontrolna jedinica.

Ulazna jedinica ima zadatak da prima spoljašnje informacije i da ih, u pogodnom obliku, prenese u memorijsku jedinicu. Tako da ona ima i ulogu translatora informacija sa spoljašnjeg jezika na jezik računara. Proces primanja i prenošenja informacija mora biti usaglašen sa drugim procesima u digitalnom uređaju, te je stoga ulazna jedinica povezana i sa kontrolnom jedinicom.

Zavisno od namjene digitalnog sistema, ulazna jedinica sadrži razne sklopove, pa i aparature, kao što su: tastatura, čitači informacija sa magnetnih traka ili diskova, zatim analogno–digitalni konvertori, kao i eventualni uređaj za daljinski prenos informacija.

Izlazna jedinica ima obrnutu ulogu od ulazne. Ona prenosi obrađene informacije, odnosno rezultate iz memorije računara na spoljašnji indikator. Prema tome, ovdje se radi o translatoru informacija sa mašinskog na spoljašnji jezik. I ova jedinica se aktivira samo po uputstvima koja prima iz kontrolne jedinice.

Izlazna jedinica najčešće sadrži sklopove, odnosno aparature kao što su: ekrani, štampači, magnetni rekorderi, digitalno–analogni konvertori, itd.

Memorijska jedinica ima zadatak da zapamti, odnosno sačuva informacije koje dolaze iz ulazne ili aritmetičke jedinice. Ove informacije odnose se ne samo na podatke obrađivanog

zadatka, već i na uputstva, odnosno programe u vezi načina obrade primljenih podataka. Otuda je poželjno da memorijska jedinica ima što veći kapacitet, kako bi mogla da primi veliki broj informacija. Pri tome treba imati u vidu ne samo mogućnost smještanja informacija u memoriju, već i brzo pronalaženje lokacije, odnosno adrese memorijskih ćelija kojima će informacija biti predata ili od kojih je treba uzeti. Oba ta procesa, upisivanje podataka u memoriju, kao i njihovo očitavanje iz memorije, takođe su dirigovani od strane kontrolne jedinice.

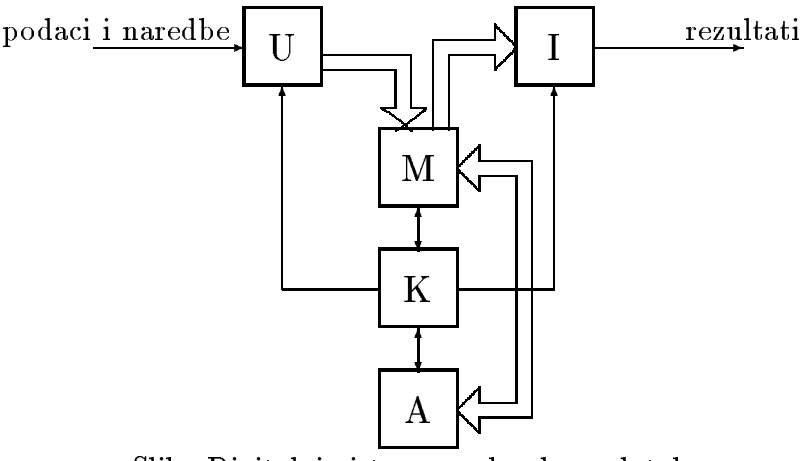
Postoji više vrsta memorijskih elemenata koji se koriste u memorijskoj jedinici. Za veće kapacitete memorije, u upotrebi su uglavnom magnetni elementi, recimo diskovi. Međutim, za kratkotrajno pamćenje podataka češće se koriste elektronski memorijski elementi, recimo registri.

Zadatak aritmetičke jedinice je da izvršava predviđene operacije prilikom obrade informacija. Potrebne podatke ova jedinica uzima iz memorije, u koju isto tako i vraća dobijene rezultate nakon izvršenih aritmetičkih operacija. Ove operacije obuhvataju četiri osnovne računske radnje, a zastupljeni su i neki drugi matematički postupci. Svi postupci u aritmetičkoj jedinici izvode se prema uputstvima dobijenim od kontrolne jedinice.

Aritmetička jedinica sadrži raznovrsna elektronska kola. Tu su prije svega logički elementi, koji čine sabirače, dekodere, itd. a zatim i memorijski elementi, koji sačinjavaju razne registre, brojače, itd.

Kontrolna jedinica je najsloženiji dio digitalnog uređaja. Ona je povezana sa svim drugim jedinicama. Ona im daje uputstva o tome šta, kada i kako treba da rade. Ona je pravo središte računara. Tumači uputstva i daje da se ona ostvare. Drugim riječima, zadatak kontrolne jedinice je da organizuje i koordinira djelatnost svih ostalih jedinica u digitalnom sistemu. Na primjer, kada neka informacija treba da bude primljena sa ulaza, tada kontrolna jedinica aktivira ulaz i stavlja na raspolaganje prenosni put od ulaza do određenog mjesta u memoriji. Kontrolna jedinica sve radnje izvodi saglasno datom programu, koji je takođe smješten u memoriji.

Kontrolna jedinica sadrži veliki broj raznovrsnih elektronskih kola, kako sa logičkim, tako isto i sa memorijskim elementima. Posebno treba istaći da se u ovoj jedinici nalazi i generator taktova. To je jedno astabilno kolo. Na primjer, to je obični oscilator, stabilisan pomoću kvarcnog kristala. Pored toga, tu se nalaze djelitelji učestanosti, uobličivači napona, itd.

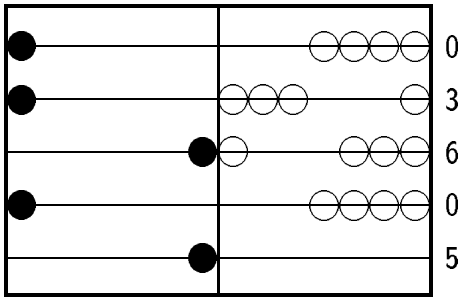


Slika Digitalni sistem za obradu podataka

Za ulaznu i izlaznu se kaže da su periferne jedinice. Za aritmetičku i kontrolnu zajedno kaže se da čine centralnu jedinicu. Budući da se sam uređaj brine o izvršavanju programa, kaže se da se radi o računaru fon Nojmanovog tipa.

4. RAZVOJ RAČUNARSKE TEHNIKE

Još prije dvije hiljade godina pojavio se abakus – prvi uređaj za računanje. Na slici je prikazan jedan mogući raspored njegovih kuglica. Pomoću jednog rasporeda zadaje se jedan broj. Upravo, računaju se kuglice blizu sredine, a tamne kuglice vrijede po 5. Tako da je na slici prikazan broj n = 3605. Pomoću ovog uređaja lako se obavlja sabiranje. Vidimo da abakus koristi digitalni princip.



Francuski matematičar Blaise Pascal je tvorac prvog uredđaja za računanje. Njegova mehanička mašina Pascaline mogla je da sabira brojeve od po 8 cifara. Pojedini točkić mašine imao je deset zubaca. Mašina je koristila tegove koji su mogli da se podižu i spuštaju. Izradu prvog primjerka mašine Paskal je završio 1642. godine. Ukupno je izradio oko 50 primjeraka. Neki od tih primjeraka sačuvani su do danas.

Njemački matematičar G. Leibniz izradio je 1670. godine mašinu koja je mogla da obavlja sabiranje, oduzimanje i množenje. Mašina je koristila pokretni prenosni mehamizam. Mogla je da pomnoži broj koji ima do 5 cifara sa brojem koji ima do 12 cifara. Za množenje, trebalo je da korisnik okrene ručicu jednom za svaku jedinicu svake cifre prvog množitelja, a nazubljeni valjak je pretvarao okretanja u sabiranja.

Dekan katedre za matematiku univerziteta u Kembridžu (Engleska) Charles Babbage počeo je 1834. godine rad na mehaničkoj mašini za računanje koju je nazvao Analitička mašina (Analytical Engine) i na kojoj je radio niz godina. Zamislio je da našina bude u stanju da obavlja dugačke serije izračunavanja bez intervencije čovjeka. Mašina je trebalo da radi nad brojevima od po 40 cifara. Mlin ili centralna jedinica trebalo je da ima dva glavna akumulatora i još nekoliko pomoćnih za posebne namjene. Ostava ili memorija trebalo je da čuva 100 brojeva. Bilo je predviđeno da ima nekoliko čitača kartica za unošenje kako ulaznih podataka tako isto i programa. Ipak, Bebidž nije postigao da njegova mašina zaista proradi (za njegovog života). On je uspio da konstruiše neke djelove mašine i prototip svoje mašine.

Teorijski su značajni rezultati engleskog matematičara A. M. Turinga koji su objavljeni 1936. godine. Tjuring je razmatrao mašinu koja izvršava program, pri čemu redosljed operacija zavisi od ulaznih podataka i od međurezultata. Ta mašina je kasnije nazvana Tjuringovom mašinom. Tjuringov rad je bio usmjeren ka matematici i ka teoriji, a ne da bi se po njegovom radu napravila stvarna mašina. Dakle, "Tjuringova mašina" nije fizički objekat, nego je samo nešto što je matematički zamišljeno. Ipak, Tjuringove ideje su značajne u razvoju računarske tehnike. Pokazalo se da Tjuringova mašina ima opšti karakter: obradu informacija koju može da izvrši Tjuringova mašina može da izvrši i računar, i obrnuto. Dakle, Tjuringova mašina predstavlja matematičku formalizaciju računara u opštem smislu.

1941. godine Konrad Zuse iz Berlina je završio mašinu Z3. Ona je radila nad brojevima u pokretnom zarezu, pojedini broj imao je 22 binarne cifre. Memorija mašine bila je sastavljena od elektromehaničkih tj. elektromagnetnih elemenata tj. od releja. Memorija je držala 64 takva broja, tako da je sadržala više od 1400 releja. Bilo je još 1200 releja u aritmetičkoj i kontrolnoj jedinici.

Program se čitao sa bušene papirne trake. Brojne vrijednosti unosile su se preko numeričke tastature. Mašina je mogla da obavi 3–4 sabiranja u sekundi, a trebalo joj je 3–5 sekundi za jedno množenje. Zuse je još 1938. godine bio napravio mašinu Z1.

1941. godine, na Iowa State University (SAD), John V. Atanasoff, profesor fizike, završio je mašinu ABC. Brzina časovnika iznosila je 60Hz. Za jedno sabiranje trošila je jednu sekundu.

Američki inžinjer Howard H. Aiken i njegova ekipa radili su na Harvard University, Cambridge, Mass., SAD, još od kraja tridesetih godina na izgradnji elektromehaničkog računara. Tako je 1943. godine završena mašina Mark I, takođe poznata pod nazivom Harvard Mark I. Kasnije su oni izgradili i bolje mašine, elektromehanički elementi (releji) bili su postepeno zamjenjivani elektronskim elementima, upravo vakuumskim cijevima.

Do sada smo govorili o mašinama Z3, ABC i Harvard Mark I. U nastavku ćemo govoriti o mašinama Colossus i Eniac. Za sve ove mašine može se reći da su kalkulatori ili da su kalkulatori koji mogu da se programiraju. A ne bi trebalo reći da su računari koji mogu da se programiraju. Zato što se program nalazio van memorije tj. program se nalazio u memoriji iz koje može samo da se čita, kako bi se reklo današnjim rječnikom, engl. read–only memory. Tako da program tokom rada mašine očito nije mogao da se mijenja. Zato ovakve mašine praktično nisu raspolagale mogućnošću uslovnog skoka (naredba if).

Nabrojane mašine koristile su se uglavnom za tabeliranje vrijednosti transcedentnih funkcija i za rješavanje sistema linearnih jednačina. Osim mašine Colossus, koja je služila za rješavanje kombinatornog logičkog zadatka, upravo zadatka o dekodiranju ili dešifrovanju. Naime, tokom Drugog svjetskog rata, njemačka vojska je koristila mašinu Enigma za kodiranje odnosno dekodiranje poruka koje je jedna vojna komanda slala drugoj. Englezi su lako dobijali kodirani oblik poruke, a pomoću računara Colossus uspijevali su da poruku dešifruju.

Prvi primjerak računara Colossus završen je 1943. godine, u Engleskoj, blizu Londona. Tjuring je učestvovao u projektu računara Colossus, ali ne u izradi samog računara, nego u sastavljanju programa za dekodiranje.

J. Presper Eckert i John Mauchly, sa University of Pensilvanya, SAD, izgradili su elektronski računar Eniac, Electronic Numerical Integrator and Calculator. Rad na mašini počeo je 1943. godine, mašina je postala operativna tokom 1945. godine, a pokazana je javnosti 1946. godine. Eniac je sadržao 18.000 vakuumskih cijevi i 1500 releja. Brzina časovnoka iznosila je 100KHz. Eniac se sastojao od otprilike 30 manjih mašina (djelova), koje ćemo sada nabrojati. Bilo je 20 akumulatora, svaki od po 10 dekadnih cifara, koji su služili za pamćenje brojeva i kao sabirači. Jedan množač. Jedna mašina za dijeljenje i računanje kvadratnog korijena. Jedna jedinica za ostvarivanje programske petlje tj. ciklusa, današnjim rječnikom kazano za naredbu for i = 1 to 25 ili slično. Tabla sa 104 tzv. konstantna registra, za definisanje vrijednosti konstanti. Tabla za umetanje utičnica, pomoću čega se definisala veza djelova računara tj. definisao se program rada mašine; pridruženi kablovi. Izvor napajanja. Generator taktova, zajedno sa pridruženim kablovima odnosno sa linijama za sinhronizovanje. Najzad, čitači bušenih kartica i štampači.

Tako da se unošenje programa za Eniac ustvari sastojalo od ručno vršenog povezivanja kablovima različitih jedinica mašine, na način da ona može da obavi svoje operacije u željenom redosljedu. Dakle, programiranje računara Eniac vršilo se umetanjem kablova i time povezivanjem različitih jedinica računara. Imali su okvire za kablove, što je davalo bar malo reda izgledu rasporeda kablova. Ipak je programiranje Eniac–a bilo težak posao i zadavanje jednog programa zahtijevalo je obično 24 časa. To se smatralo zadovoljavajućim, jer je onda izvršavanje tog programa trajalo nedjelju dana ili nekoliko nedjelja.

Da li je Eniac mogao da obavlja uslovne skokove (if)? Djelimično zahvaljujući i slučaju, up-

ravljački signali Eniac–a poklapali su se skoro sa njegovim signalima podataka. Inžinjeri i matematičari koji su radili na razvoju računara otkrili su jednog dana da izlazne linije podataka jednog akumulatora mogu da budu sprovedene na upravljačke ulazne linije drugog akumulatora. Funkcionisanje drugog akumulatora zavisi od primljenih njegovih upravljačkih signala. Dakle, zavisi od brojne vrijednosti sadržane u prvom akumulatoru. Prema tome, govoreći sa izvjesnom slobodom, Eniac je raspolagao sa mogućnošću uslovnih skokova (naročito kasnije izrađene verzije računara).

John von Neumann je bio član ekipe koja je radila na razvoju Eniac–a. U jednom svom izvještaju on je predložio da se sa programom manipuliše na isti način kao sa brojmim vrijednostima, predložio je da se i program upiše u memoriju računara, engl. stored program. Neka raspored kablova bude stalan, a neka program definiše kakve veze jedinica zaista važe. Logički koncept rada računara predložen od strane fon Nojmana ostao je da važi skoro bez ikakve izmjene do dana današnjeg! U spomenutom izvještaju predlaže se izrada računara nazvanog Edvac, Electronic Discrete Variable Automatic Calculator.

Približno 1946. godine fon Nojman je napustio Pensilvaniju, a napustili su je i Ekert i Maukli približno 1947. godine. Računar Edvac završen je u Pensilvaniji tek 1952. godine. Fon Nojman je zajedno sa svojim saradnicima na Institute for Advanced Study (IAS), Princeton, SAD, izgradio mašinu zvanu IAS mašina, zasnovanu na ideji iz svog već spomenutog izvještaja, približno 1950. godine. Pomenimo još dva slična računara, takođe naravno sa upisanim programom. Edsac iz 1949. godine, Cambridge, Engleska, izgrađen od strane Maurice Wilkes. Mark I, mašina ili mali prototip iz 1948. godine, Mančester, Engleska.

Krupan korak u razvoju računara predstavljala je pojava 1964. godine IBM–ove familije računara System 360. U odnosu na prethodnike, ovi računari imali su poboljšane mogućnosti, a uticali su i da se tržište za računare proširi. Računar je mogao da obavi 500.000 sabiranja u sekundi.

Firma DEC je 1965. godine pokazala PDP–8 – prvi komercijalni miniračunar. Ova mala mašina predstavljala je prodor prema projektovanju jeftinog. Miniračunari su bili prethodnici mikroprocesora.

Prve personalne računare proizvela je firma Apple 1977. godine. IBM–ov Personal Computer najavljen je 1981. godine. IBM PC postao je najbolje prodavani računar, od svih vrsta računara. Njegov uspjeh učinio je da Intel 80x86 bude najpopularniji mikroprocesor i učinio je da DOS firme Microsoft postane najpopularniji operativni sistem.

Da bi se vidjela perspektiva razvoja, industrija teži da grupiše računare u generacije. Obično se podjela vrši po tehnologiji izrade koju koristi određena generacija. Po dogovoru se uzima da se prva generacija odnosi na komercijalne elektronske računare. Sljedeća tabela prikazuje generacije računara.

Generacija	Godine	Tehnologija
I	1950–1959	Vakuumske cijevi
II	1960–1968	Tranzistori
III	1969–1977	Integrisana kola
IV	1978–...	Mikroprocesori

Najzad, nekoliko riječi o razvoju u posljednjih nekoliko godina. Razvoj paralelnih računara. Jedan paralelni računar sadrži veliki broj procesora, recimo sadrži njih 2^4 ili 2^{16}. Svi oni rade u saradnji na rješavanju jednog te istog zadatka. Mnogo je važniji razvoj mreža personalnih računara. Bilo da se radi o lokalnoj mreži koja se odnosi na jednu zgradu bilo o raširenoj mreži koja povezuje računare iz raznih gradova. Internet je najpoznatija mreža, to je globalna računarska mreža.

Rezime o razvoju računara (o istoriji računara)

1642. godine Pascal: mehanički kalkulator, nije automatizovan. Izvodi pojedinačne aritmetičke operacije.

1840. godine Babbage: mehanički računar Analytical Engine. Plan je bio dobar, ali je tadašnji nivo tehnologije bio nedovoljan. Nije proradilo, ostalo je kao zamisao.

1941. godine Zuse: elektromehanički kalkulator Z3. Izvršava pravolinijske programe. Program se tokom izvršavanja čita sa papirne trake.

1945. godine Eckert: električna mašina Eniac. Mješavina kalkulatora i računara. Program se definiše fizičkim rasporedom kablova (po tome je slabiji od Z3).

1952. godine von Neumann: elektronski računar Edvac! Program se nalazi u memoriji; stored–program computer. Zato su mogući uslovni skokovi (naredba if). Isto tako, lako se radi sa nizovima brojeva; pišemo recimo A[n]+B[n] u Paskalu. Prosto rečeno, Edvac je računar. Nema principijelne razlike između računara Edvac i savremenih računara.

5. TEHNOLOGIJE IZRADE, RAZVOJ MIKROPROCESORA

Tranzistor se pojavio 1948. godine. To je elektronski element koji može da se nalazi u jednom od dva jasno definisana stanja – zakočenja ili zasićenja (ne provodi, odnosno provodi), koja odgovaraju logičkoj nuli i logičkoj jedinici. Drugim riječima, tranzistor je nula–jedan prekidač. Tranzistor obavlja istu funkciju kao i vakuumska cijev, samo što ima bolje karakteristike nego vakuumska cijev, u pogledu pouzdanosti, veličine, potrošnje energije i cijene.

Mi možemo da povežemo nekoliko pojedinačnih (diskretno izrađenih) tranzistora u jednu cjelinu. A bilo bi bolje da je ta cjelina fizički izrađena kao jedan komad, da ona predstavlja jednu elektronsku komponentu. Tada se govori o integrisanom kolu. Integrisano kolo sadrži veliki broj tranzistora i tranzistoru srodnih elemenata. Prvo integrisano kolo pojavilo se 1959. godine. Integrisana kola (čipovi) izrađuju se na silicijumskoj pločici (na parčetu poluprovodničkog materijala). Korišćenje integrisanih kola doprinijelo je boljim performansama jedinice centralnog procesora (engl. central processor unit, CPU), a omogućilo je i da dotada korišćene memorije od magnetnih jezgara budu zamijenjene poluprovodničkim memorijama.

Integrisana kola se razlikuju po stepenu složenosti, po broju elemenata koje sadrže. Iz godine u godinu, dozvoljena gornja granica složenosti se pomijerala. Početkom osamdesetih godina bilo je moguće smjestiti sto hiljada tranzistora na jednom parčetu poluprovodnika (na jednom čipu), a početkom devedesetih godina već i preko milion. Umjesto stepen složenosti kaže se isto stepen integracije, a ponekad se kaže gustina pakovanja.

Napredak tehnologije omogućio je smještanje čitave jedinice centralnog procesora (CPU) na jednom čipu, tzv. mikroprocesoru. Za računar koji ima mikroprocesor (kao svoj CPU, kao svoj procesor) ponekad se kaže da je mikroračunar. Prvi mikroprocesor pojavio se 1971. godine – Intel 4004. Sadrži 2300 tranzistora, a ima dimenzije 0,3cm $\times$ 0,4cm.

Od prvih elektronskih računara pa do savremenih računara, principi organizacije računara (program se nalazi upisan u memoriji, itd), kao i osnovna struktura računara, ostali su dobrim dijelom neizmijenjeni. Revolucionarni razvoj koji su računari napravili od tada pa do danas vezan je, prije svega, za revoluciju u proizvodnji elektronskih komponenti.

Firma Intel (Integrated Logic) osnovana je 1968. godine i bavila se uglavnom proizvodnjom memorijskih čipova. Najveći proizvođači mikroprocesora su američke firme Intel i Motorola. Intel 4004 može da obrađuje podatke u grupama od po četiri bita (četiri binarne cifre). Godinu dana nakon toga Intel proizvodi i osmobitni mikroprocesor poznat pod nazivom 8008; u jednom snopu obrađuje podatke veličine osam bita (jednog bajta). Mikroprocesor Intel 8080 iz 1974.

godine takođe je osmobitni. Ispostavilo se da su osmobitni mikroprocesori obično dovoljni za razne slučajeve mašina (koje nisu računari) koje sadrže mikroprocesor, a služe recimo za upravljanje nekim tehnološkim procesom.

Prvi IBM–ov personalni računar baziran je na procesoru Intel 8088. Intel 8088 interno ima 16–bitnu strukturu, što znači da radi sa 16–bitnim informacijama kao osnovnim jedinicama informacija, ali sa okolnim čipovima komunicira preko 8 linija (preko 8-bitne magistrale). Intel 8086 pojavio se 1978. godine (moćniji je od 8088); to je 16–bitni mikroprocesor; ima mogućnost 16–bitne komunikacije sa okolnim čipovima. Računar IBM PC XT obično koristi mikroprocesor Intel 8086. Sa pojavom 16–bitnih mikroprocesora dostignute su mogućnosti onoga što se dotad smatralo velikim računarskim sistemom. Intel 80286 je takođe 16–bitni, a pojavio se 1983. godine. Ugrađuje se u IBM PC AT, koji se pojavio 1984. godine.

Personalni računari koji su slijedili, 386 i 486, bazirani su na 32–bitnim čipovima Intel 80386 (iz 1985. godine) i njegovoj bržoj verziji Intel 80486 (iz 1989. godine). Danas se najviše koristi Intelov mikroprocesor Pentium; 32–bitni.

Personalni računar (engl. personal computer, PC) znači IBM–ov računar koji ima mikroprocesor, a nestrogo ima smisao računar koji ima mikroprocesor.

Kao primjer, pogledajmo bar neke karakteristike mikroprocesora Intel 8080, za koga je već rečeno da je 8–bitni i da je proizveden 1974. godine. Izrađen je u tehnologiji MOS, sa kanalom tipa N (sa kanalom negativnog tipa); izrađen je u tehnologiji NMOS.

Ima 40 pinova (spoljašnjih izvoda).

Radi sa frekvencijom takta od 2MHz.

Njegov mašinski jezik ima 78 vrsta naredbi.

MOS je skraćenica za engl. metal oxide semiconductor, što se prevodi kao metal oksid poluprovodnik. To su tri materijala koji učestvuju u sastavu tranzistora.

1993. godine	Pentium broj 1	3.100.000 tranzistora	273 pina	takt 60MHz
1997. godine	Pentium broj 2	7.500.000 tranzistora	370 pinova	takt 300MHz
1999. godine	Pentium broj 3	28.000.000 tranzistora	370 pinova	takt 1130MHz

Za proizvodnju integrisanih kola najviše se koriste sljedeće dvije tehnologije:

bipolarna tehnologija i

MOS tehnologija.

Bipolarna tehnologija zasnovana je na upotrebi tranzistora koji koriste pokretna punjenja za oba polariteta. Bipolarni tranzistori su brzi, pa se koriste za veoma brza logička kola. MOS tehnologija koristi tranzistore u kojima aktivni tokovi prolaze neposredno ispod površine oksida (u kojima se provodni kanal obrazuje neposredno ispod površine oksida). Važne prednosti MOS tehnologije nad bipolarnom su: lakše se postiže veća gustina pakovanja, ne zahtijevaju se dodatne izolacije i manja je potrošnja energije.

Moore–ov zakon: broj tranzistora na čipu udvostruči se svake dvije godine (ili svake dvije godine i šest mjeseci).

6. BROJNI SISTEMI

Skup oznaka i pravila koja služe za prikazivanje brojeva čini brojni sistem. Brojne sisteme dijelimo na nepozicione i pozicione. U nepozicionom sistemu vrijednost cifre ne zavisi od mjesta

koje ona ima u nizu cifara (u nizu cifara koji čini zapis određenog broja). A ako zavisi onda je pozicioni. Nepozicioni su se u prošlosti prvi pojavili, a danas se koriste jedino pozicioni.

Pogledajmo tri primjera brojnih sistema.

Prvi pozicioni brojni sistem u prošlosti bio je sistem koji su koristili Vavilonci u staroj Mesopotamiji, od godine -2000 približno. Sistem je bio sa osnovom $N = 60$. Tako da su postojale cifre od $n = 1$ do $n = 59$. Cifru $n = 0$ nisu pisali eksplicitno. Oni su imali svega dva simbola koje ćemo mi prikazivati uslovno kao α i β. Dva simbola imali su vrijednost 1 odnosno 10 redom. Navedimo primjer. Zapis $\beta\beta\alpha\alpha$, $\beta\alpha\alpha\alpha$, $\alpha\alpha\alpha\alpha$ odgovara broju $22 \cdot 60^2 + 13 \cdot 60 + 4$.

Navedimo i najpoznatiji primjer nepozicionog brojnog sistema iz prošlosti. Rimski brojni sistem koristio se od godine -700 približno. Predstavljao je glavno sredstvo za prikazivanje brojeva u Evropi sve do sredine XIV vijeka. U rimskom sistemu imamo cifre I V X L C D i M čije su vrijednosti redom 1 5 10 50 100 500 i 1000. Recimo MMCCCXLVIII znači 2348.

Postoji najprostiji način za prikazivanje prirodnih brojeva gdje se koristi samo jedan znak, recimo znak 1. Da se prikaže broj n, taj znak se napiše n puta. Recimo 1111 znači 4. Ovaj brojni sistem je nepozicioni.

Ubuduće ćemo govoriti samo o pozicionim.

Znamo da se u matematici koristi jedino brojni sistem čija je osnova $N = 10$ tj. dekadni brojni sistem. Imamo cifre $0, 1, \ldots, 9$. Recimo $4623 = 4 \cdot 10^3 + 6 \cdot 10^2 + 2 \cdot 10 + 3 \cdot 1$. Za veličine 10^3, 10^2, 10 i 1 koje se pojavljuju kažemo da predstavljaju težine pojedinih mjesta u zapisu. Potpuno isti princip važi ako osnova brojnog sistema više nije $N = 10$ nego je sada neki prirodan broj $N \geq 2$. Sistem sa osnovom N ima skup cifara $\{0, 1, \ldots, N-1\}$. Niz cifara $c_k c_{k-1} \ldots c_2 c_1 c_0$ označava po definiciji broj $c_k \cdot N^k + c_{k-1} \cdot N^{k-1} + \ldots + c_2 \cdot N^2 + c_1 \cdot N + c_0$. Pogledajmo posebno slučajeve $N = 8$, $N = 16$ i $N = 2$.

U oktalnom sistemu ($N = 8$), cifre su od 0 do 7. Recimo, $(724)_8 = 7 \cdot 64 + 2 \cdot 8 + 4 = 468$.

U heksadekadnom sistemu ($N = 16$), osim svih dekadnih cifara, imamo još i cifre od A do F čije su vrijednosti redom od 10 do 15. Recimo, $(2\mathrm{E})_{16} = 2 \cdot 16 + 14 = 46$.

U binarnom sistemu ($N = 2$), imamo samo dvije cifre, to su 0 i 1. Recimo, $(1101110)_2 = 64 + 32 + 8 + 4 + 2 = 110$. Binarni sistem je značajan u računarskoj tehnici zato što njegove dvije cifre 0 i 1 imaju jednostavnu fizičku realizaciju. Na primjer, zatvoren prekidač može da predstavlja 1, a otvoren 0. Ili, visok napon može da znači 1, a nizak napon (ili napon uzemljenja) 0. Jedan smjer magnetnog polja može da se tumači kao 1, a drugi smjer kao 0.

Sljedeća tabela sadrži, za prirodne brojeve od 1 do 16, njihovo prikazivanje u sistemima sa osnovom redom $N = 10$, $N = 2$, $N = 8$ i $N = 16$.

$N=10$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$N=2$	1	10	11	100	101	110	111	1000	1001	1010	1011	1100	1101	1110	1111	10000
$N=8$	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17	20
$N=16$	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10

Pogledajmo kako se u brojnom sistemu čija je osnova $N \geq 2$ prikazuje broj x koji pored cijelog ima i razlomljeni dio: $x = c_k c_{k-1} \ldots c_1 c_0, c_{-1} \ldots c_{-l}$ znači $x = \sum_{j=-l}^{k} c_j \cdot N^j$, gdje je $0 \leq c_j \leq N-1$. Pogledajmo primjere koji se odnose na binarni brojni sistem: $(0{,}011)_2 = \frac{1}{4} + \frac{1}{8} = \frac{3}{8}$, $(10{,}011)_2 = 1 \cdot 2^1 + 0 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} = 2\frac{3}{8} = 2{,}375$.

Najzad, četiri osnovne aritmetičke operacije u sistemu sa osnovom N izvode se po analogiji sa slučajem dekadnog brojnog sistema.

Ako neki broj u dekadnom obliku ima k cifara onda taj isti broj prikazan u binarnom sistemu ima približno $k \cdot \log_2 10$ cifara.

7. PREVOĐENJE BROJA IZ JEDNOG SISTEMA U DRUGI

Razmotrimo sljedeći zadatak. Dat je prikaz jednog broja u sistemu čija je osnova N_1, a treba odrediti prikaz tog istog broja u sistemu čija je osnova N_2.

Razlikovaćemo tri slučaja.

Prvi slučaj: $N_2 = 10$. Ovaj slučaj ne predstavlja teškoću. Zadatak se rješava tako što se realizuje definicija prikazivanja broja u brojnom sistemu čija je osnova jednaka N_1. Mi umijemo da računamo u dekadnom brojnom sistemu.

Pogledajmo primjer, iz oktalnog u dekadni. Neka treba da rastumačimo oktalni zapis 7104. Mi prosto pišemo $7 \cdot 8^3 + 1 \cdot 8^2 + 0 \cdot 8 + 4 \cdot 1$ i računanjem ovog izraza izlazi 3652. Taj odgovor zapisujemo u obliku $(7104)_8 = 3652$. Slično se postupa kada je $N_1 = 16$ ili $N_1 = 2$.

Drugi slučaj: $N_1 = 10$, prevođenje iz dekadnog sistema. Formulišimo pravilo koje rješava razmatrani zadatak. Izrazićemo to pravilo u slučaju da je $N_2 = 8$, a slično pravilo važi i kada se cio broj prevodi iz dekadnog u bilo koju osnovu N_2.

Dati broj treba podijeliti sa 8, čime se dobijaju količnik i ostatak. Ostatak predstavlja zadnju cifru rezultata. Količnik opet podijeliti sa 8, čime se dobijaju novi količnik i novi ostatak. Novi ostatak predstavlja predzadnju cifru rezultata. Itd. Sa postupkom uzastopnog dijeljenja sa 8 prestaje se kada tekuća vrijednost količnika postane jednaka nuli.

Razmotrimo primjer. Neka broj $n = 939$ treba zapisati oktalno. V. tabelu.

$939 : 8 = 117$ ostatak 3 ↑
$117 : 8 = 14$ ostatak 5 |
$14 : 8 = 1$ ostatak 6 |
$1 : 8 = 0$ ostatak 1 |

Odgovor je $939 = (1653)_8$.

Sada hoćemo da broj 939 zapišemo heksadekadno. V. tabelu.

$939 : 16 = 58$ ostaje 11 tj. B ↑
$58 : 16 = 3$ ostaje 10 tj. A |
$3 : 16 = 0$ ostaje 3 |

Odgovor je $939 = (3AB)_{16}$.

Najzad, hoćemo da 939 zapišemo u binarnom brojnom sistemu. V. tabelu.

$939 : 2 = 469$ 1
$469 : 2 = 234$ 1
$234 : 2 = 117$ 0
$117 : 2 = 58$ 1
$58 : 2 = 29$ 0
$29 : 2 = 14$ 1
$14 : 2 = 7$ 0
$7 : 2 = 3$ 1
$3 : 2 = 1$ 1
$1 : 2 = 0$ 1

Odgovor je $939 = (1110101011)_2$.

Treba dokazati korektnost predloženog postupka za pretvaranje iz dekadnog u neki drugi sistem. Napišimo jednačine da potkrijepimo predlog:

$$\begin{aligned}
&x : N = x_0 \text{ sa ostatkom } c_0\\
&x_0 : N = x_1 \text{ sa ostatkom } c_1\\
&x_1 : N = x_2 \text{ sa ostatkom } c_2\\
&\dots\\
&x_{k-1} : N = x_k \text{ sa ostatkom } c_k\\
&x_k = 0
\end{aligned}$$

Znači da je $x = Nx_0 + c_0$, $x_0 = Nx_1 + c_1$, $x_1 = Nx_2 + c_2$, ..., $x_{k-1} = c_k$. I da je $0 \le c_j \le N-1$ za $j = 0, \dots, k$. Tako

$$x = Nx_0 + c_0 = N(Nx_1 + c_1) + c_0 = N^2x_1 + Nc_1 + c_0 = N^2(Nx_2 + c_2) + Nc_1 + c_0 =$$

$$N^3x_2 + N^2c_2 + Nc_1 + c_0 = \dots = N^kc_k + \dots + N^2c_2 + Nc_1 + c_0 = (c_k \dots c_2c_1c_0)_N,$$

po definiciji zapisa prirodnog broja u brojnom sistemu čija je osnova N.

Dokazana je korektnost postupka za prevođenje $10 \to N$.

Treći slučaj: neka $N_1, N_2 \in \{2, 8, 16\}$.

Ako se iz oktalnog prevodi u binarni onda svaku oktalnu cifru treba zamijeniti grupom od tri binarne cifre, prema Tabeli 1. Vidimo da Tabela 1 daje binarno predstavljanje svakog broja koji sam po sebi predstavlja pojedinačnu oktalnu cifru.

Tabela 1

oktalno	binarno
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Navedimo primjer. Neka kao primjer posluži opet broj $n = 939$ odranije. Dakle, dat je oktalni zapis $(1653)_8$, a treba odrediti ekvivalentni binarni zapis. Upravljajući se po Tabeli 1, imamo $(1653)_8 = (001\,110\,101\,011)_2 = (1110101011)_2$, što i predstavlja odgovor.

Jasno je da proces inverzan ovom procesu odgovara prevođenju binarni $\to$ oktalni. Recimo $(1110101011)_2 = (1\,110\,101\,011)_2 = (001\,110\,101\,011)_2 = (1653)_8$. Vidimo da treba grupisati tri po tri binarne cifre u smjeru zdesna ulijevo.

Utvrdimo ispravnost predloženog postupka grupisanja, barem u slučaju broja x koji se izražava pomoću šest binarnih cifara: $x = (c_5c_4c_3c_2c_1c_0)_2$, $0 \le c_i \le 1$,

$$x = c_5 \cdot 2^5 + c_4 \cdot 2^4 + c_3 \cdot 2^3 + c_2 \cdot 2^2 + c_1 \cdot 2 + c_0 =$$

$$(c_5 \cdot 2^2 + c_4 \cdot 2 + c_3) \cdot 8 + (c_2 \cdot 2^2 + c_1 \cdot 2 + c_0) = d_1 \cdot 8 + d_0 = (d_1d_0)_8$$

Zapažamo da je $0 \le d_j \le 7$. A takođe zapažamo da je $d_1 = (c_5c_4c_3)_2$ i $d_0 = (c_2c_1c_0)_2$.

Slično prethodnom, ako se iz heksadekadnog prevodi u binarni onda svaku heksadekadnu cifru u zapisu treba zamijeniti grupom od četiri binarne cifre, upravljajući se po Tabeli 2. Vidimo da Tabela 2 za svaki jednocifreni heksadekadni broj daje njegov binarni zapis, odnosno da daje njegov tzv. prirodni binarni kod.

Tabela 2

Hex	Bin	Hex	Bin
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

Recimo da je dato $(3AB)_{16}$ i da se traži $(\ldots)_2$. Imamo $(3AB)_{16} = (0011\ 1010\ 1011)_2 = (1110101011)_2$.

Slično, binarni $\rightarrow$ heksadekadni.

Na kraju, ako treba da pretvorimo iz oktalnog u heksadekadni ili obrnuto onda to možemo da uradimo posredstvom binarnog.

Sva tri slučaja su riješena.

U nastavku, kako da se pravi razlomak x zadat dekadno $(0 < x < 1)$ prikaže u binarnom obliku. Treba uraditi ovako. Na početku je tekuća vrijednost jednaka x. Tekuću vrijednost pomnožiti sa $N = 2$ i proizvod rastaviti na njegov cijeli dio (predstavlja cifru c_{-1} odgovora) i njegov ostatak (predstavlja novu tekuću vrijednost). Itd. da nađemo $c_{-2}, c_{-3}, \ldots$. Sve dok se tekuća vrijednost ne svede na nulu.

Sljedeći račun služi da pokaže ispravnost predloženog postupka: $0 < x < 1$, dekadni $\rightarrow$ binarni, $N = 2$, $0 < x \cdot N < 2$, $x \cdot N$ rastavimo na cijeli dio plus razlomljeni dio:

$$x \cdot N = c_{-1} + x_{-1}, \quad c_{-1} = 0 \text{ ili } 1, \quad 0 < x_{-1} < 1, \quad x = N^{-1}c_{-1} + N^{-1}x_{-1}$$

$$x_{-1} \cdot N = c_{-2} + x_{-2}, \quad c_{-2} = 0 \text{ ili } 1, \quad 0 < x_{-2} < 1$$

$$x_{-2} \cdot N = c_{-3} + x_{-3}, \quad c_{-3} = 0 \text{ ili } 1, \quad 0 < x_{-3} < 1, \quad \ldots$$

$$x_{-(l-1)} \cdot N = c_{-l} + x_{-l}, \quad c_{-l} = 0 \text{ ili } 1, \quad x_{-l} = 0$$

Znači da je $x = N^{-1}c_{-1} + N^{-1}x_{-1} = N^{-1}c_{-1} + N^{-1}(N^{-1}c_{-2} + N^{-1}x_{-2}) = N^{-1}c_{-1} + N^{-2}c_{-2} + N^{-2}x_{-2} = \ldots = N^{-1}c_{-1} + N^{-2}c_{-2} + \ldots + N^{-l}c_{-l}$ po definiciji $= (0{,}c_{-1}c_{-2}\ldots c_{-l})_2$.

8. UVOD O FLIPFLOPOVIMA, REGISTRIMA I MEMORIJI

U digitalnoj elektronici koriste se i elementi koji imaju sposobnost pamćenja tj. memorisanja svog stanja. Ti elementi imaju dva stabilna stanja pa se nazivaju bistabilnim kolima. Jedno stanje odgovara visokom naponu recimo U=5V, a drugo stanje niskom naponu recimo U=0V. Jedno stanje ima smisao binarne cifre 1, a drugo cifre 0. Element može izaći iz svog stanja samo pod dejstvom pobudnog signala. Ako nema pobude onda element zadržava svoje stanje neograničeno dugo. Pobudni impuls mora imati dovoljnu amplitudu i dovoljno trajanje Δt da

bi obezbijedio promjenu stanja. Kaže se pobudni signal ili okidni signal. Kaže se da kolo okida ili da se prebacuje ili da mijenja svoje stanje Q. Kaže se bistabilno kolo ili triger; engl. trigger. Običan primjer bistabilnog kola jeste SR triger. Ima dva ulaza S i R i ima jedan izlaz Q. Slika

Izlazni signal tj. vrijednost napona na izlaznoj liniji Q jeste funkcija od vremena t. Izlazni signal može da bude na visokom nivou (Q=1) ili na niskom (Q=0). Vrijednost Q predstavlja stanje trigera, predstavlja binarnu vrijednost koju triger pamti. Ako se po jednoj i po drugoj ulaznoj liniji S i R dovodi logička nula onda se Q ne mijenja odnosno triger pamti. Ako se ulaz S pobudi tokom vremenskog intervala Δt onda će poslije toga da postane Q(t)=1; t – vrijeme. Pobuđivanje ulaza R izaziva upisivanje nule u triger; postaće Q(t)=0. S – set, R – reset.

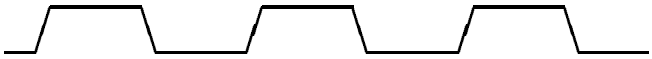
Postoje dvije klase bistabilnih kola. Kola prve klase nazivaju se leč kola; engl. latch. Kod tih kola izlaz stalno prati promjene na ulazima. Napominje se da neka leč kola imaju i tzv. ulazni signal dozvole E; engl. enable – dozvola. Ako je taj signal =0 onda se izlaz sigurno ne mijenja, bez obzira na ostale ulaze. Kola druge klase nazivaju se flipflopovi. Kod flipflopova, promjena stanja može se vršiti samo u tačno određenim trenucima vremena, ti trenuci determinisani su taktnim signalom sistema. Dakle, izlaz iz generatora taktova dovodi se kao jedan ulaz na flipflop. Taj ulaz se obično označava kao CLK; engl. clock – časovnik. Prema tome, u slučaju flipflopa, govori se o njegovom stanju Q_n tj. o vrijednosti njegovog izlaznog signala poslije n otkucaja časovnika, u n–tom taktu. Treba reći da se u literaturi često ne pravi razlika između ove dvije klase, pa se kola i jedne i druge klase nazivaju flipflopovima. U sastav jednog trigera ulazi obično desetak tranzistora. Tako se SR leč kolo iz TTL familije sastoji od 12 tranzistora i dvije diode, tj. sastoji se od 14 komponenti. TTL znači transistor–transistor logic. U tehnici niskog stepena integracije (SSI) proizvode se leč kola i flipflopovi iz TTL, ECL i CMOS familija, najčešće po dva u istom kućištu. Slična je situacija i u tehnici srednjeg stepena integracije (MSI), samo što je broj kola u kućištu veći i iznosi 4, 6 ili 8. NMOS bistabilna kola proizvode se samo kao elementi složenijih kola u visokom stepenu integracije (LSI) i u vrlo visokom (VLSI), zbog problema oko sprezanja. Familije ECL i TTL odnose se na bipolarnu tehnologiju, a CMOS i NMOS su podvrste MOS tehnologije.

Poznato je da se integrisana kola dijele u četiri klase, prema složenosti, koja se mjeri brojem komponenti, kako slijedi. 1. Kola niskog stepena integracije, Small scale integration, SSI. Sadrže do 100 komponenti na silicijumskoj podlozi. Na ovaj način realizuju se standardna logička kola i flipflopovi. Silicijumska podloga pakuje se u standardno kućište. 2. Kola srednjeg stepena integracije, Medium scale integration, MSI. Sadrže od 100 do 1000 komponenti na silicijumskoj podlozi. Ovakvom tehnologijom realizuju se složenija kombinaciona i sekvencijalna kola, koja se takođe pakuju u standardna kućišta. 3. Kola visokog stepena integracije, Large scale integration, LSI. Sadrže od 1000 do 10000 komponenti. U LSI tehnologiji realizuju se složeni specijalizovani digitalni sistemi. 4. Kola vrlo visokog stepena integracije, Very large scale integration, VLSI. Sadrže preko 10000 komponenti. Ovim kolima realizuju se vrlo složeni digitalni sistemi, kao što su mikroprocesori, memorije, itd. Prave se u MOS tehnologiji. Najsloženija savremena VLSI kola sadrže preko 1.000.000 komponenti na silicijumskoj pločici. Znamo da logička ili kombinaciona kola odgovaraju logičkim funkcijama (i, ili, ne i slično). A sekvencijalna kola uključuju i trigere.

Za sekvencijalnu mrežu koja se koristi za privremeno memorisanje digitalnih informacija kaže se da je registar ili da je stacionarni registar. Da bi dva ili više bistabilnih kola pred-

stavljali registar uslov je da imaju zajednički taktni impuls. Stacionarni registri se koriste i za privremeno memorisanje digitalnih podataka prilikom asinhrone razmjene informacija, odnosno razmjene između digitalnih uređaja čije su brzine različite. U takvoj ulozi, registar se naziva prihvatnim registrom ili baferom; engl. buffer. Ako svi memorijski elementi u nekoj digitalnoj mreži jednovremeno mijenjaju stanje nakon uzlazne ili silazne ivice zajedničkog taktnog impulsa onda je to jedna sinhrona mreža. Ako se u nekoj mreži koristi više od jednog taktnog impulsa ili ako neki od njenih memorijskih elemenata može da promijeni stanje (promjenom neke od ulaznih promjenljivih) nezavisno od taktnog impulsa onda je to jedna asinhrona mreža.

Generator taktova se takođe realizuje kao jedno integrisano kolo. Zamislimo da je tokom jedne sekunde izlaz tog generatora nizak napon, onda tokom jedne sekunde da je visok napon, itd. naizmjenično. Ivicu imamo kada dolazi do promjene napona, uzlaznu kada napon raste, a silaznu kada napon opada. Na sljedećoj maloj slici prikazani su tzv. vremenski dijagrami izlaznog signala iz izvora taktova, tj. signala CLK. Drugim riječima, prikazan je grafik funkcije U=U(t), gdje je t vrijeme, a U je vrijednost signala CLK, U je odgovarajući nivo napona.



Engl. binary digit – binarna cifra, skraćeno bit. Bit je informacija koju sadrži jedan flipflop (jedna ćelija). Upravo, bit je $\boxed{0}$ ili $\boxed{1}$. Engl. byte. Jedan bajt je informacija koju sadrži registar koji se sastoji od osam flipflopova. Na primjer $\boxed{1}\boxed{1}\boxed{1}\boxed{0}\boxed{1}\boxed{0}\boxed{1}\boxed{0}$. Jedan bajt jednako je osam bita, pišemo 1B=8b.

količina informacije	njen fizički nosilac
bit (binarna cifra)	flipflop ili memorijska ćelija
bajt (8 bita)	registar

Memorije u digitalnim sistemima predstavljaju sklopove u koje se može upisati i iz kojih se može pročitati informacija. Zavisno od medijuma na kome se informacija pamti, najčešće se koriste poluprovodničke i magnetne memorije. Magnetne memorije koriste se uglavnom za memorisanje velikog broja digitalnih informacija. Vrijeme upisivanja i čitanja informacija je relativno dugačko, zbog neophodnih mehaničkih pomijeranja diska ili trake. Informacije u tim memorijama ostaju zapamćene i kada je isključeno električno napajanje, tako da te memorije spadaju u klasu postojanih memorija. Memorija u koju informacija može da bude bilo upisana bilo pročitana iz nje u proizvoljnom trenutku (i sa proizvoljne adrese) naziva se RAM memorijom. RAM – Random access memory – memorija sa slučajnim pristupom. Poluprovodničke RAM memorije po pravilu gube sadržaj kada se isključi napon napajanja, tako da spadaju u klasu nepostojanih memorija. RAM memorija predstavlja skup stacionarnih registara sa zajedničkim ulaznim i izlaznim priključcima. Selekcija registra u koji će se informacija upisati ili iz koga će se informacija pročitati obavlja se adresnim dekoderom.

Poluprovodničke memorije mogu biti statičke ili dinamičke. Informacija upisana u statičkoj memoriji ostaje zapamćena sve dok je memorija priključena na napon napajanja. S druge strane, da bi informacija ostala zapamćena u dinamičkoj memoriji neophodno je periodično obavljati "osvježavanje" memorije, inače bi se informacija izgubila. Ovakva memorija ima svoj tzv. kontroler za osvježavanje koji automatski "potvrđuje" sadržaj svake ćelije, imajući u vidu karakteristike kondenzatora, odnosno tempo kojim se troši punjenje kondenzatora.

Svaka memorijska ćelija statičke RAM memorije (SRAM) sastoji se od najmanje četiri do šest tranzistora. Da bi se realizovala memorija sa većom gustinom pakovanja, konstruisana je memorija sa samo jednim tranzistorom i jednim kondenzatorom po memorijskoj ćeliji. Ovakva

memorija bazira pamćenje informacije na električnom punjenju kondenzatora. To je dinamička RAM memorija (DRAM).

Poluprovodničke RAM memorije su sastavni dio svakog računarskog sistema. Zavisno od veličine računara, potrebni kapacitet RAM memorije se kreće od nekoliko desetina KB za specijalizovane računarske sisteme pa do više GB za velike superračunare. Kapacitet RAM memorije personalnog računara je najčešće u granicama od 1MB do 16MB. Integrisana memorijska kola izrađuju se komercijalno sa kapacitetom primjera radi 64KB ako su u pitanju statičke memorije, odnosno 0,5MB ako su u pitanju dinamičke memorije. Tako da je za realizaciju RAM memorije računarskog sistema neophodno da se koristi više čipova. Statičke memorije su manjeg kapaciteta po čipu, a koriste se u sistemima gdje se zahtijeva veća brzina pristupa memoriji i manja potrošnja struje iz izvora za napajanje. Dinamičke memorije zahtijevaju manji broj integrisanih kola nego statičke, za isti kapacitet, zbog daleko veće gustine pakovanja. Zbog manjeg broja čipova i jednostavnije štampane ploče, cijena DRAM memorije je niža od SRAM istog kapaciteta.

Memorija sa konstantnim sadržajem jeste integrisano kolo u koje se posebnim postupkom upiše željeni sadržaj, a kada je sadržaj jednom upisan onda jedino mogu da se čitaju informacije iz memorije. Takva memorija naziva se ROM memorijom; Read only memory.

Tabela o memoriji: 1 dio računarskog sistema, 2 princip izrade tog dijela, 3 kakve memorijske komponente sadrži taj dio, 4 moguće operacije nad komponentama, 5 kako komponenta pamti, 6 količina komponenti, 7 brzina pristupa

1	CPU (centralni procesor)	memorija ili unutrašnja memorija ili RAM ili memorijski čip	spoljašnja memorija ili disk
2	elektronski	elektronski	magnetni
3	 registar	 registar ili lokacija ili riječ	 lokacija ili riječ
4	upisivanje i čitanje i druge operacije	upisivanje i čitanje	upisivanje i čitanje
5	privremeno	privremeno	trajno
6	mali broj registara	veliki broj registara	ogroman broj
7	vrlo brzo	brzo	sporo

9. NEPOKRETNI ZAREZ I SABIRANJE U NEPOKRETNOM ZAREZU

U ovom naslovu govori se o načinu kodiranja ili predstavljanja cijelih brojeva u memoriji računara, a zatim se još govori o postupku ili algoritmu za izvršavanje aritmetičkih operacija nad cijelim brojevima, odnosno nad kodovima cijelih brojeva (ograničićemo se samo na operaciju sabiranja). Način izvršavanja operacija očito je uslovljen memorijskim predstavljanjem.

Umjesto nepokretni zarez još se kaže i fiksirani zarez, zato što zarez ima fiksirano mjesto (iza zadnje cifre). Također se kaže i potpuni komplement, skraćeno PK.

Za predstavljanje ili reprezentaciju jednog cijelog broja n koristi se prostor veličine 2 bajta ili 4 bajta. Ponekad se (mada rijetko) za prikazivanje troši prostor od samo jednog bajta. Mi ćemo princip prikazivanja objasniti za taj slučaj jednog bajta, budući da je princip prikazivanja uvijek jedan te isti, a samo zapisivanje je kraće.

Prva ideja jeste da se broj prikaže preko svog znaka i svoje apsolutne vrijednosti. U osnovi, brojevi se prikazuju u binarnom sistemu. Jedan bit zapisa (upravo prvi bit zapisa) izdvoji se da razlikuje pozitivne brojeve od negativnih. Neka s (znak, engl. sign) bude vrijednost prvog bita zapisa. Ako je $s = 0$ onda je prikazani broj pozitivan ili je nula. Ako je $s = 1$ onda se prikazani broj smatra negativnim. Sedam bita odnosi se na apsolutnu vrijednost broja. Pogledajmo primjer. Znamo da je $5 = (101)_2$. Tako $\boxed{00000101}$ prikazuje broj $n = 5$. A $\boxed{10000101}$ bi prikazivalo broj $n = -5$.

Pogodno je da se prva ideja malo dogradi, što ćemo sada izložiti. Dogradnja se odnosi samo na negativne brojeve. Dakle, PK zapis broja $n = 5$ jeste upravo ono što je maločas navedeno, jer je $n \geq 0$, a PK zapis broja $n = -5 < 0$ tek će biti konstruisan. Dogradnja se sastoji iz dva koraka. U prvom koraku vrši se komplementiranje (0 se zamijeni sa 1, a 1 se zamijeni sa 0) svih bita koji se odnose na prikaz apsolutne vrijednosti broja, tj. svih bita zapisa osim prvog bita. Za rezultujući zapis ili kod kaže se da je tzv. prvi komplement ili da je nepotpuni komplement, skraćeno NK. U slučaju $n = -5$ ovome odgovara $\boxed{11111010}$.

U drugom koraku se na oblik NK doda 1. Drugim riječima, na NK oblik se sada gleda kao na običan binarni broj i taj broj se sabere sa 1. Treba očito znati kako se u binarnom sistemu računa recimo $11111010 + 1$. Izlazi 11111011. Kada se izvrši sabiranje onda se i dobija konačni oblik za koji se kaže da predstavlja zapis u obliku drugog komplementa (engl. 2's complement) ili u obliku potpunog komplementa (PK). Konkretno, za $n = -5$ zapis glasi $\boxed{11111011}$. Pravilo o prikazivanju cijelih brojeva u nepokretnom zarezu (u obliku PK) ovim je formulisano.

Mala slika:

a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8

ili:

s	...						

Primjer 1: zapisati broj $n = 28$ u obliku PK. Rješenje: uočimo da je $28 = (11100)_2$. Odgovor: $\boxed{00011100}$ ili

0	0	0	1	1	1	0	0

Primjer 2: zapisati broj $n = -28$ u obliku PK. Rješenje se sastoji iz tri koraka: oblik sa znakom i apsolutnom vrijednošću (ZA) $\boxed{10011100}$, oblik NK $\boxed{11100011}$ i najzad oblik PK koji i predstavlja odgovor $\boxed{11100100}$ (imamo da je u binarnom prikazu $11100011 + 1 = 11100100$). Ili

1	1	1	0	0	1	0	0

Može se sastaviti tabela koja sadrži sve brojeve n koji mogu da budu prikazani u obliku PK na prostoru veličine jednog bajta i u kojoj su još navedeni i odgovarajući PK prikazi tih brojeva n. U nastavku je dat dio te tabele.

broj n	PK prikaz
-128	10000000
-127	10000001
...	...
-5	11111011
-4	11111100
-3	11111101
-2	11111110
-1	11111111

0	00000000
1	00000001
2	00000010
3	00000011
4	00000100
5	00000101
...	...
126	01111110
127	01111111

Iz tabele vidimo da u memoriji mogu da se čuvaju brojevi od $n = \min = -128$ do $n = \max = 127$, tj. mogu da budu upisani brojevi u granicama ili u rasponu od $-2^7 \leq n \leq 2^7 - 1$. Ima $2^8 = 256$ različitih mogućih sadržaja u jednom bajtu.

Važi formula:

$$(n)_{\text{PK}} = \begin{cases} n \text{ ako je } n \geq 0 \\ n + 2^8 \text{ ako je } n < 0 \end{cases}$$

Uvedena je oznaka $(n)_{\text{PK}} = (a_1 a_2 \ldots a_8)_2$, gdje je $\boxed{a_1 a_2 \ldots a_8}$ – PK prikaz broja n.

Npr. $(-1)_{\text{PK}} = (11111111)_2 = 255$.

Razmotrimo puno realniji slučaj zapisivanja cijelih brojeva: neka se za zapisivanje jednog cijelog broja troše 2 bajta (16 bita). U tom slučaju se recimo broj $n = 5$ čuva kao

0000 0000 0000 0101

dok se recimo broj $n = -5$ čuva u obliku

1111 1111 1111 1011

Koji je najmanji a koji je najveći cio broj koji može da se prikaže u obliku PK na prostoru od 2 bajta? Najmanji mogući broj je $n = \min = -2^{15} = -32768$, a najveći je $n = \max = 2^{15} - 1 = 32767$.

Pogledajmo i slučaj kada se za zapisivanje cijelog broja troše 4 bajta (32 bita). Najmanji mogući cio broj koji može da bude prikazan jeste $n = \min = -2^{31} = -2{,}147 \cdot 10^9$, a najveći je $n = \max = 2^{31} - 1 = 2{,}147 \cdot 10^9$. A ako se u našem programu pojavljuju brojevi koji su veći od 2.000.000.000, kako onda, razmotriti za vježbu.

Svaki programer zna da se cijeli brojevi unose na običan dekadni način (za negativne se unosi znak minus). A rečeno je da se u memoriji broj čuva u binarnom obliku ili preciznije u obliku PK. Proces unošenja broja u memoriju podrazumijeva i njegovo pretvaranje (konverziju) iz spoljašnjeg oblika u unutrašnji oblik prikaza. Za ovo pretvaranje zadužena je jedna kombinaciona mreža (jedno digitalno kolo), jedna vrsta enkodera. Slično, prilikom izdavanja (saopštavanja, štampanja) broja vrši se pretvaranje u suprotnom smjeru, PK zapis se dekodira ili dešifruje ili tumači, odgovarajuća mreža ili kolo jeste jedna vrsta dekodera.

Dosad smo govorili o prikazivanju cijelih brojeva.

Odsad ćemo govoriti o postupku za sabiranje dva cijela broja. Sabiraju se prikazi jednog i drugog broja n_1 i n_2, rezultat $n_1 + n_2$ će takođe biti u obliku PK. Ako je recimo $n_1 = -3$ i $n_2 = -2$ onda treba da bude očito $n_1 + n_2 = -5$. Drugim riječima, ako se na ulazu zadaju dva zapisa $\boxed{11111101}$ i $\boxed{11111110}$ onda na izlazu mreže za sabiranje (na izlazu sabirača) treba da se dobije $\boxed{11111011}$, uporediti sa tabelom. I dalje koristimo model – jedan bajt za jedan broj, radi kraćeg pisanja.

Pravilo o sabiranju dva PK zapisa glasi: dva PK zapisa sabiraju se kao da su to obični binarni brojevi, s tim da se posljednji prenos zanemaruje. Posljednji prenos ili krajnji lijevi prenos ili prenos najveće težine p ne ulazi u rezultat, bilo da je $p = 0$ bilo da je $p = 1$. Zanemarivanje posljednjeg prenosa: kao kada bismo dekadno rekli da je $715 + 716 = 431$. Da ponovimo, rezultat sabiranja dva prikaza $(n_1)_{\text{PK}}$ i $(n_2)_{\text{PK}}$ jeste $((n_1)_{\text{PK}} + (n_2)_{\text{PK}}) \bmod 2^8$.

Navedeno pravilo o računanju $n_1 + n_2$ nećemo dokazivati, već ćemo ga samo ilustrovati na četiri karakteristična primjera. Za primjere, mi znamo kako se sabiraju dva obična binarna broja.

Prelazimo na primjere 1–4. 1 $n_1 > 0$ i $n_2 > 0$: ovdje ustvari nema šta da se provjerava, jer se PK zapisi poklapaju sa običnim binarnim zapisima. 2 $n_1 < 0$ i $n_2 < 0$, recimo da je $n_1 = -3$ i $n_2 = -2$. 3 Jedan sabirak je pozitivan a drugi je negativan, dok je zbir pozitivan. Recimo da je $n_1 = -1$ i $n_2 = 5$. 4 Jedan sabirak je pozitivan a drugi je negativan, dok je zbir negativan. Recimo da je $n_1 = -4$ i $n_2 = 2$. U nastavku su urađena posljednja tri primjera 2–4:

$$\begin{array}{r} 11111101 \\ +\ 11111110 \\ \hline 11111011 \end{array} \qquad \begin{array}{r} 11111111 \\ +\ 00000101 \\ \hline 00000100 \end{array} \qquad \begin{array}{r} 11111100 \\ +\ 00000010 \\ \hline 11111110 \end{array}$$

$$\left(\begin{array}{r} a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 \\ +\ b_1 b_2 b_3 b_4 b_5 b_6 b_7 b_8 \\ \hline c_1 c_2 c_3 c_4 c_5 c_6 c_7 c_8 \end{array} \right)$$

Šta će se desiti ako je recimo $n_1 = 81$ i $n_2 = 92$? Jasno je da zbir $n = n_1 + n_2$ ne pripada dozvoljenom intervalu $-128 \leq n \leq 127$. Tako da pravilna vrijednost n ne može da bude na pravilan način zapisana u memorijski prostor predviđene veličine jednog bajta (bez obzira na algoritam za izvođenje sabiranja). U takvom slučaju se kaže da je nastupilo prekoračenje ili engl. overflow. Po pravilu, računar nam saopšti da je došlo do prekoračenja (i zaustavi se). Za vježbu, na konkretnom softveru, utvrditi granice koje definišu dozvoljeni interval za cijele brojeve $[\min, \max]$ i utvrditi da li se o prekoračenju dobija obavještenje.

U zaključku, vidimo da pravilo o sabiranju ne traži da se razlikuje slučaj kada su oba sabirka jednog te istog znaka od slučaja kada je jedan sabirak pozitivan a drugi negativan, čime se i opravdava uvođenje prikazivanja cijelih brojeva na način PK: odgovarajuća mreža za sabiranje biće jednostavnija, na račun nerazlikovanja dva slučaja. Za odgovarajuću mrežu se kaže da predstavlja sabirač (radićemo kasnije). Na kraju, uz mrežu sabirača, može da se doda i jedna mala mreža za otkrivanje eventualnog prekoračenja. Ta mala mreža na osnovu binarnih promjenljivih a_1, b_1, c_1 i p zaključuje da li je do prekoračenja došlo ili nije došlo.

10. POKRETNI ZAREZ

U ovom naslovu govori se o memorijskom predstavljanju realnih (decimalnih) brojeva. Umjesto pokretni zarez kaže se i engl. floating point, a takođe se kaže i princip mantise i eksponenta. Za predstavljanje realnog broja x troši se prostor od 4 bajta (32 bita), po pravilu, pa ćemo izložiti taj osnovni slučaj.

Za pripremu, u slučaju dekadnog sistema, broj $x > 0$ može da se prikaže u obliku $x = m \cdot 10^e$, gdje je eksponent e cio broj a mantisa m je < 1. Recimo, $x = 34{,}56 = m \cdot 10^e$ sa $m = 0{,}3456$ i $e = 2$. A ako je $x < 0$ onda se rastavlja kao $x = -m \cdot 10^e$. Veličina e definiše pravo mjesto decimalnog zareza u mantisi m, zato se i kaže pokretni zarez. Povećanje e za recimo 1 izaziva množenje broja x sa 10. Kod računara, ulogu dekadnog sistema preuzima binarni sistem. Povećanje e za recimo 1 izazvalo bi sada da se x udvostruči, što se smatra nedovoljnim povećanjem broja x. Tako da se binarni sistem ustvari kombinuje sa heksadekadnim sistemom.

Formulišimo pravilo o predstavljanju realnog broja x. Broju x pridruže se tri veličine: s (sign, znak), m i e. Ako je $x \geq 0$ onda je $s = 1$ a ako je $x < 0$ onda je $s = -1$. Važi jednakost $x = s \cdot m \cdot 16^e$, gdje je e cio broj, a m zadovoljava $m < 1$. Dodatno, m zadovoljava uslov $\frac{1}{16} \leq m < 1$; kaže se da je mantisa normalizovana. Sada su sve tri veličine s, m i e jednoznačno određene, po datom x.

S druge strane, imamo na raspolaganju 32 bita. Raspoloživi memorijski prostor raspodjeljuje se na sljedeći način: 1 bit za prikazivanje s, 7 bita za e i 24 bita za m. Numerišimo bite slijeva udesno od 1 do 32, tako da za s bit 1, za e od 2 do 8 i za m od 9 do 32. Što se tiče prvog bita, ako je $x > 0$ onda je njegov sadržaj $s_1 = 0$ a ako je $x < 0$ onda je njegov sadržaj $s_1 = 1$. Što se tiče bita od drugog do osmog, eksponent e prikazuje se u obliku PK (potpuni komplement), na prostoru veličine sedam bita; $e = (e_1 \dots e_7)_{PK}$. Tako da su moguće vrijednosti e u granicama $-64 \leq e \leq 63$. Što se tiče mantise $m > 0$, na mjesta 9–32 upisuje se njen obični binarni oblik; pravi razlomak m predstavlja se sa svoje prve 24 binarne cifre iza zareza. Ako je $m = (0{,}m_1 m_2 \dots m_{24})_2$ onda se upisuju binarne cifre $m_1, m_2, \dots, m_{24}$. Ukupno, u memoriji se nalazi $\boxed{s_1|e_1|e_2|e_3|e_4|e_5|e_6|e_7|m_1|m_2|\ \dots\ |m_{24}}$. Posebno, broj $x = 0$ prikazuje se svim nulama $\boxed{0|0|\ \dots\ |0}$ (32 bita). Pravilo je formulisano.

Zapazimo sljedeće. Uslov $x = 0$ ekvivalentan je sa $m_1 = m_2 = m_3 = m_4 = 0$.

Navedimo primjere. U drugom primjeru krećemo se u smjeru od broja x ka njegovom zapisu ili kodu u obliku pokretnog zareza, a u prvom primjeru krećemo se u suprotnom smjeru.

Primjer 1. Dat je zapis (32 bita) $\boxed{1111110011111100\dots00}$, odrediti koji broj x je zapisan. Rješenje. Vidimo da je $s_1 = 1$, $e_1 \dots e_7 = 1111100$, $m_1 \dots m_{24} = 11111100 \dots 00$. Znači da je $s = -1$ tj. da je $x < 0$. Što se tiče eksponenta e, ravnamo se po sljedećoj tabeli:

$e =$	-64	-63	$\dots$	-2	-1	0	1	2	$\dots$	62	63
$PK =$	1000000	1000001	$\dots$	1111110	1111111	0000000	0000001	0000010	$\dots$	0111110	0111111

Tako nalazimo da je $e = -4$. Sljedeće, $m = (0{,}m_1 m_2 \dots m_{24})_2 = (0{,}11111100 \dots 00)_2 = (0{,}1111 11)_2 = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32} + \frac{1}{64} = \frac{63}{64}$. Na kraju, $x = s \cdot m \cdot 16^e = -\frac{63}{64} \cdot 16^{-4}$, što i predstavlja odgovor.

Primjer 2. Odrediti zapis broja $x = 48$ ili $x = 48.0$ u obliku pokretnog zareza. Rješenje. Nalazimo da je $x = s \cdot m \cdot 16^e$ sa $s = 1$, $m = \frac{3}{16}$ i $e = 2$, pa će biti $s_1 = 0$, $m = \frac{1}{8} + \frac{1}{16} = (0{,}0011)_2$, pa će biti $m_1 = m_2 = 0$, $m_3 = m_4 = 1$, $m_5 = m_6 = \dots = m_{24} = 0$ i $e = 2 = (0000010)_{PK} = (e_1 \dots e_7)_{PK}$. Zbirno, odgovor je $s_1 e_1 \dots e_7 m_1 \dots m_{24} = \boxed{0|0000010|00110\dots0}$ (32 bita).

Primjer 3. Odrediti zapis broja $x = 76{,}8$ u obliku pokretnog zareza. Rješenje. Budući da je $x > 0$ to će biti $s = 1$ tj. $s_1 = 0$. Treba odrediti m i e da bude $\frac{1}{16} \leq m < 1$ i $x = m \cdot 16^e$.

Nalazimo da je $76{,}8 = 0{,}3 \cdot 16^2$ tj. da je $m = 0{,}3$ i $e = 2$. Sljedeće, $e = 2 = (0000010)_{PK}$ tj. $e_1 \ldots e_7 = 0000010$. Ostaje da se mantisa prikaže binarno. Treba naći binarni oblik broja $m = 0{,}3$ tj. treba naći $m = (0, \ldots)_2 = (0{,}m_1 m_2 \ldots)_2$. Poznato je da treba uzastopno množiti sa 2 i da onda cijeli djelovi proizvoda daju redom upravo $m_1, m_2, \ldots \in \{0, 1\}$.

$0{,}3 \cdot 2 = 0{,}6 \qquad m_1 = 0$
$0{,}6 \cdot 2 = 1{,}2 \qquad m_2 = 1$
$0{,}2 \cdot 2 = 0{,}4 \qquad m_3 = 0$
$0{,}4 \cdot 2 = 0{,}8 \qquad m_4 = 0$
$0{,}8 \cdot 2 = 1{,}6 \qquad m_5 = 1$
$0{,}6 \cdot 2 = 1{,}2 \qquad m_6 = 1 \qquad$ (ponavlja se)
$0{,}2 \cdot 2 = \ldots$

$\ldots$
$m_{24} =$

$0{,}3 = (0.0\overline{1001})_2 = (0, 0\,1001\,1001\,1001\,1001\,1001\,100)_2$, iza zareza 24 cifre. Odgovor glasi $\boxed{0}\boxed{0000010}\boxed{0\,1001\,1001\,1001\,1001\,1001\,100}$.

Slijede razne dopune.

Zanimljivo je odrediti raspon, tj. odrediti koji je najmanji pozitivan broj $x = min$ i koji je najveći pozitivan broj $x = max$ koji ovako može da se zapiše. Najmanji mogući pozitivan broj min dobija se ako se uzme najmanji mogući eksponent e, a to je $e = -64$, i najmanja moguća mantisa m, a to je $m = \frac{1}{16}$. Tako da je $min = m \cdot 16^e = \frac{1}{16} \cdot 16^{-64}$. Za max, najveći mogući eksponent je $e = 63$, a najveća moguća mantisa je $m \approx 1$. Tako da je $max = m \cdot 16^e \approx 1 \cdot 16^{63}$. Broj min je reda veličine 10^{-80}, a broj max je reda veličine 10^{80}, grubo govoreći. Ako je x takav da je $|x| < min$ onda će x biti registrovan kao nula. Ako je realni broj x takav da je $|x| > max$ onda x ne može da bude upisan (u obliku pokretnog zareza, na prostoru od četiri bajta).

Ako na opisani način hoćemo da zapišemo (kodiramo) brojeve recimo $x_1 = 0{,}1236$ i $x_2 = 0{,}1237$ onda će oni imati različite zapise (kodove). Međutim, ako želimo da zapišemo dva broja x_1 i x_2 koji se razlikuju tek na osmoj (dekadnoj) decimali onda će se jedan i drugi zapis poklopiti, budući da se prve 24 binarne cifre iza zareza takvih brojeva x_1 i x_2 poklapaju. Kaže se da preciznost ili tačnost zapisivanja realnog broja x iznosi sedam dekadnih cifara (sedam tačnih ili sigurnih dekadnih cifara); oko sedam.

Mi smo izložili princip pokretnog zareza. Postoje dva ili tri standarda koji su opšte–prihvaćeni za prikazivanje realnog broja. Ti se standardi potpuno poklapaju sa izloženim principom ili se malo razlikuju od izloženog principa. Tako se u zapisu može izdvojiti jedan bit više za mantisu, odnosno samim tim i jedan bit manje za eksponent. Sada se preciznost ili tačnost malo poveća. Ali se raspon (dozvoljeni interval) smanji, jer se sada eksponent e kreće samo u granicama od $e = -32$ do $e = 31$. Tako da je sada min reda veličine 10^{-40}, a max je reda veličine 10^{40}, grubo govoreći.

11. BCD KODOVI I ASCII KOD

Svi podaci u računaru kodiraju se (predstavljaju se) pomoću binarnih znakova 0 i 1. Treba izabrati način predstavljanja brojeva, tekstova sastavljenih od riječi i programa na mašinskom jeziku. Što se tiče brojeva, već smo govorili o načinima nepokretni i pokretni zarez, a u nastavku će biti riječi i o tzv. BCD kodu, iako se BCD kodovi danas rijetko koriste. Što se tiče tekstova, odnosno riječi (odnosno alfa–numeričkih podataka), riječ je sastavljena od slova tj. od štamparskih znakova. Tako da treba izvršiti kodiranje pojedinačnih slova koja sastavljaju riječ.

Za slova se koristi tzv. ASCII kod, o tom kodu će biti riječi u nastavku. Što se tiče programa na mašinskom jeziku, taj program se sastoji od naredbi. Jedna naredba troši obično dva ili četiri bajta. Dakle, ako mašinska naredba zauzima dva bajta onda ona predstavlja jedan niz dužine 16 čije su komponente znaci 0 i 1. O ovome ćemo govoriti kasnije, kada se opredijelimo za jedan konkretan primjer mašinskog jezika.

Recimo, tzv. osnovni računar (basic computer) ima obični oblik naredbe tj. ima tzv. jedno–adresni oblik naredbe; u naredbi se pojavljuje jedna adresa; 16 bita po naredbi. Navedimo mali primjer: uvećaj akumulator za sadržaj memorijske lokacije m; $ac \leftarrow ac + c(m)$; $c(m)$ znači contents of m; radnja sabiranja. Ova vrsta naredbe drži se kao |0001|m|; m zauzima 12 bita. Recimo, naredba |0001 0000 0000 1110| ima smisao $ac \leftarrow ac + c(14)$.

BCD ili Binary coded decimal predstavlja jedan način za kodiranje prirodnih odnosno cijelih brojeva. Ulazna informacija koja treba da bude kodirana jeste prirodan broj zapisan u dekadnom sistemu. Ovdje se svaka dekadna cifra kodira pomoću jedne četvorke binarnih cifara. A kod čitavog broja dobija se kada se kodovi pojedinih njegovih dekadnih cifara napišu jedan za drugim. Još samo treba kazati kako se kodira pojedina dekadna cifra. U tabeli su prikazana dva načina za kodiranje (zapisivanje) pojedine dekadne cifre.

dekadna	njen kod	
cifra	8421	višak 3
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

Način "8421" predstavlja ustvari obični ili prirodni binarni zapis pojedine dekadne cifre. Predloženi način (predloženi kod) "8421" ima svojstvo da je težinski; težine pojedinih mjesta iznose upravo 8, 4, 2 i 1 redom. Drugi način "višak 3" dobija se kada se prvi način uveća za 3. Ovaj način (ovaj kod) "višak 3" je simetričan tj. posjeduje svojstvo komplementarnosti: dvije dekadne cifre koje se dopunjavaju do 9 imaju kodove koji se dopunjavaju do 1111.

Vidimo da jedna cifra troši pola bajta. Uzmimo da je dužina memorijskog prostora predviđenog za predstavljanje cijelog broja stalna (fiksirana), upravo uzmimo da je dužina jednaka dva bajta. Uzmimo da se za zapisivanje znaka broja potroši takođe pola bajta: prva polovina prvog bajta jednaka je 0000 za pozitivne cijele brojeve a jednaka je 0001 za negativne. Tako može da bude predstavljen bilo koji trocifreni cio broj. Navedimo jedan primjer. Kako se zapisuje broj -87? Upotrebiti "višak 3". Odgovor je |0001 0011|1011 1010|, u smislu |$-$ 0|8 7|.

Način kodiranja BCD upotrebljavao se kod računara prve i druge generacije, a danas se koristi uglavnom kod kalkulatora.

Za zapisivanje jednog štamparskog znaka tj. jednog karaktera troši se jedan bajt. Danas je za to zapisivanje opšte–prihvaćen ASCII kod. ASCII znači American standard code for information interchange. Dužina zapisa karaktera jednaka je 7 bita. Jednom karakteru (jednom znaku) odgovara jedna kombinacija znakova 0 i 1 u kojoj ima 7 članova. U sljedećoj tabeli

prikazan je sedmobitni ASCII kod. Za svaki znak ili original prikazan je njegov kod. Mi navodimo samo dio tabele.

original	kod		
znak	njiegov kod $b_6b_5b_4\,b_3b_2b_1b_0$	Hex	Dec
CR	000 1101	0D	13
ESC	001 1011	1B	27
SP	010 0000	20	32
>	011 1110	3E	62
?	011 1111	3F	63
A	100 0001	41	65
B	100 0010	42	66
a	110 0001	61	97
b	110 0010	62	98
DEL	111 1111	7F	127

Uz tabelu: CR carriage return (enter), ESC escape, SP space (blanko), DEL delete.

Prve 32 kombinacije i posljednja kombinacija (DEL) predstavljaju kontrolne kodove za upravljanje perifernim uređajima računarskog sistema i ne mogu da budu odštampane. Dekadne cifre predstavljaju se kombinacijama od 30_{16} do 39_{16}. Velika slova od 41_{16} do $5A_{16}$. Mala slova od 61_{16} do $7A_{16}$. Primjećujemo da se kod za malo slovo razlikuje od koda za veliko slovo samo po vrijednosti bita b_5 tj. za 20_{16}. To olakšava konverziju velikih slova u mala i obrnuto. Ostale kombinacije predstavljaju znake za interpunkciju i specijalne znake.

Na tastaturi, ako se na numeričkoj tj. maloj tastaturi otkuca recimo 65, i to se otkuca pod `Alt`, onda će se dobiti A.

Vrlo često se ASCII kodu dodaje osmi bit kojim se vrši kontrola parnosti (parity check). Upravo, u b_7 se upiše 0 ili 1 tako da ukupan broj znakova 1 u bajtu bude paran. Time se dobija osmobitni kod koji ima mogućnost detekcije jedne greške. Neka se podaci prepisuju iz jednog dijela digitalnog sistema u drugi. Ako u prepisanom obliku ima neparan broj znakova 1 u nekom bajtu onda to znači da je prilikom prepisivanja došlo do greške.

(Pojmu kontrola parnosti sličan je pojam zbir za provjeru (checksum). Prilikom zapisivanja fajla na disk, operativni sistem pridružuje fajlu njegov zbir za provjeru y. Definiše se kao $y = (x_1 + \ldots + x_n) \bmod 256$, gdje je veličina fajla jednaka n bajta, a sadržaj raznih bajta označen je kao x_k $(0 \leq x_k \leq 255)$. Kasnije će se taj fajl otvarati. Zapamćena vrijednost y biće upoređena sa vrijednošću koja odgovara otvorenom obliku fajla.)

Osmi bit može i drukčije da se iskoristi.

Da bi se omogućilo predstavljanje posebnih slova važnijih evropskih jezika, nekih grafičkih simbola i drugih simbola uveden je i proširenи osmobitni ASCII kod. U tom kodu, proširenje koda čine kodne riječi od 128_{10} do 255_{10} tj. kodne riječi za koje je $b_7 = 1$. Nažalost, ovo proširenje nije u potpunosti standardizovano u proširenom dijelu koda $b_7 = 1$. U pojedinim zemljama se proširenи dio tabele koristi po svome.

Od osmobitnih alfa–numeričkih kodova treba još spomenuti i EBCDIC koji se koristio u računarskim sistemima američke firme IBM, ali se sve manje upotrebljava.

12. MATEMATIČKI POJAM KODIRANJA

U ovom naslovu i u sljedećem naslovu govorimo o teoriji kodiranja koja je dio jedne matematičke discipline – diskretne matematike. Najšire gledano, svaka funkcija zadaje (definiše)

jedno kodiranje. Već smo imali više primjera pravila za kodiranje. Na primjer, pravilo "potpuni komplement": svakom cijelom broju od -128 do 127 pridružuje se po jednom propisu jedan tačno određeni kod, upravo jedan niz od 8 binarnih znakova. Taj kod odnosno to kodiranje "potpuni komplement" ima svojstvo uzajamne jednoznačnosti: ako je dat niz od 8 binarnih znakova onda se jednoznačno može odrediti polazni (originalni) cio broj.

Neka je $A = \{a_1, \ldots, a_m\}$. Za skup A se kaže da je azbuka a za njegove članove a_k se kaže da su slova. Kada se nekoliko slova napiše jedno za drugim onda tako nastaje jedna riječ u azbuci A. Skup svih riječi u azbuci A označava se kao $\Omega(A)$. Te riječi će biti predmet kodiranja. Moguće je da se neće vršiti kodiranje baš svake riječi. Označimo sa $\Omega'(A)$ jedan podskup skupa $\Omega(A)$. Članove skupa $\Omega'(A)$ zovemo podacima i želimo da ih kodiramo. Razmotrimo i drugu azbuku $B = \{b_1, \ldots, b_n\}$. Pojedini podatak biće kodiran pomoću jedne riječi u toj azbuci B tj. pomoću jednog člana skupa $\Omega(B)$.

Definicija: bilo koje preslikavanje $F: \Omega'(A) \to \Omega(B)$ naziva se kodiranjem.

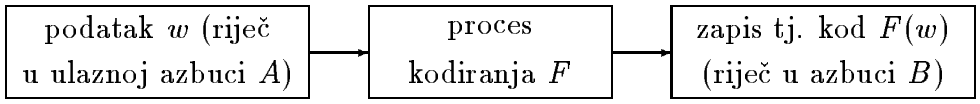
Uporediti sa ranijim primjerom PK (potpuni komplement): šta je A, $\Omega(A)$, $\Omega'(A)$, B i $\Omega(B)$ i kako glasi pravilo F. Zapaziti da kodiranje PK ima dvije osobine: da je binarno i da je ravnomjerno. Uopšte, za kodiranje se kaže da je binarno kada je $B = \{0, 1\}$. Uopšte, za kodiranje se kaže da je ravnomjerno kada svaka riječ iz skupa $\Omega(B)$ koja je nečiji kod (koja predstavlja zapis nekog objekta iz skupa svih podataka $\Omega'(A)$) ima jednu te istu dužinu. Ima dužinu 8, kada PK.

Dekodiranje je proces inverzan procesu kodiranja. Najšire gledano, dekodiranje ne mora da bude jednoznačno. A podrazunijeva se da procesi kodiranja koji se u praksi upotrebljavaju ispunjavaju uslov o jednoznačnom dekodiranju.

Dakle, za kodiranje $F: \Omega'(A) \to \Omega(B)$ kaže se da je uspješno ili da dopušta jednoznačno dekodiranje ako je funkcija F uzajamno jednoznačna.

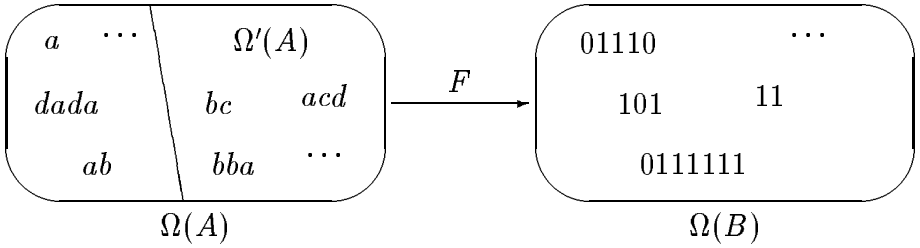
Tada se ne može desiti da dva različita objekta iz skupa $\Omega'(A)$ imaju jedan te isti kod. Na primjer, studentu odgovara broj indeksa. Ne treba da dođe do poklapanja brojeva indeksa.

Sljedeća šema prikazuje uopšteno pojam kodiranja:



Sljedeća slika prikazuje jedan primjer kodiranja F. Ovdje je $A = \{a, b, c, d\}$ i $B = \{0, 1\}$.

Može da bude recimo $F(bc) = 101$, $F(acd) = 11$ i $F(acdd) = 00000$. Vidimo da $F(dada)$ nije definisano.



U ovom primjeru i uopšte, rezultujući kodovi ne moraju da budu svi mogući tj. dopušteni kodovi. Drugim riječima, rezultujući kodovi ne moraju da pokriju čitav skup $\Omega(B)$ tj. funkcija F ne mora da bude "na" $\Omega(B)$.

Teorija kodiranja se kod računara koristi još i za tajno prenošenje podataka, tj. u tzv. kriptografiji. Može se desiti da nepoželjni učesnik u komunikaciji sazna rezultujući kod poruke. Ipak, on neće moći da razumije poslatu poruku, odnosno da dekodira (dešifruje).

13. AZBUČNO KODIRANJE, PREFIKSNO AZBUČNO KODIRANJE

Definicija: za kodiranje se kaže da je azbučno ako je svakom slovu a_k ulazne azbuke A pridružena jedna riječ w_k u izlaznoj azbuci B i ako se još zapis (kod) riječi iz azbuke A dobija tako što se odgovarajuće pridružene riječi prosto nadovežu tj. napišu se jedna za drugom.

Tako da se azbučno kodiranje zadaje (definiše) time što se zadaju sve tzv. kodne riječi $w_1, \ldots, w_m$ koje odgovaraju svim mogućim slovima $a_1, \ldots, a_m$ azbuke A. Jasno je da $w_k \in \Omega(B)$ tj. w_k je sastavljena od slova azbuke B. Pogodno je da se to definisanje izrazi u obliku tzv. šeme za kodiranje ili šablona kodiranja Σ, recimo u obliku spiska:

$$a_1 — w_1,\ a_2 — w_2,\ \ldots,\ a_m — w_m.$$

Umjesto azbučno kodiranje može se reći kodiranje slovo–po–slovo ili kodiranje znak–po–znak. Ako sve riječi $w_1, \ldots, w_m$ imaju jednu te istu dužinu (broj slova) onda se kaže da je kod ravnomjeran.

Prvi primjer azbučnog kodiranja. Neka je $A = \{a_1, a_2\}$, $B = \{0, 1\}$ i neka je definisan sljedeći šablon kodiranja: a_1 — 0, a_2 — 01. Vidimo da kod nije ravnomjeran budući da kodne riječi $w_1 = 0$ i $w_2 = 01$ imaju različite dužine (jedan odnosno dva). Odrediti kod riječi $a_2a_2a_1$. Odgovor je 01010. Rastumačiti kod 001010001. Odgovor je $a_1a_2a_2a_1a_1a_2$. Može se pokazati da je razmatrano kodiranje uspješno tj. da je tumačenje (dekodiranje) uvijek jednoznačno.

Drugi primjer azbučnog kodiranja. Neka bude $A = \{a_1, a_2, a_3\}$ i $B = \{b_1, b_2, b_3\}$. Definišimo šablon Σ: a_1 — b_1, a_2 — b_1b_2, a_3 — b_3b_1. Dakle, dato je $w_1 = b_1$, $w_2 = b_1b_2$ i $w_3 = b_3b_1$. Ovaj primjer ima iste karakteristike kao prethodni. Upravo, vidi se da kod nije ravnomjeran, a lako se pokazuje da je dekodiranje jednoznačno.

Kodiranje cijelih brojeva na način PK nije azbučno. Kodiranje cijelih brojeva na način "višak 3" jeste azbučno. Njegov šablon kodiranja Σ izražen je tabelom koju smo ranije prikazali. To kodiranje ima osobinu ravnomjernosti: četiri bita po dekadnoj cifri. Slično, ako se upotrebljava ASCII tabela onda se ustvari vrši jedno azbučno kodiranje.

Razmotrimo jednu klasu azbučnih kodiranja – klasu prefiksnih azbučnih kodiranja.

Neka je $r = c_1c_2 \ldots c_q$, tj. r je riječ, a prikazana su njena slova. Za svaku riječ oblika $c_1 \ldots c_p$, gdje je $1 \leq p \leq q$, kaže se da je početak ili prefiks riječi r. Na primjer, riječ PRED je prefiks riječi PREDMET.

Definicija: za šablon Σ tj. za azbučno kodiranje definisano tim šablonom (v. prethodnu definiciju) kažemo da je prefiksno ako nijedna kodna riječ w_i nije prefiks bilo koje druge kodne riječi w_j.

Dakle, kodne riječi se lako razlikuju jedna od druge, ne može se desiti da se dopisivanjem slova jednoj kodnoj riječi w_i dobije neka druga kodna riječ w_j. Dakle, neka pokušavamo da rastumačimo određeni rezultujući kod R, ovo R je riječ u azbuci B. Mi gledamo početna slova riječi R i upoređujemo sa šablonom Σ. Mi nađemo da se prvih nekoliko slova riječi R poklapa sa jednom kodnom riječi, recimo poklapa se sa w_2. Budući da dati šablon Σ ima osobinu da je prefiksan, to smo mi sada sigurni da je prvo slovo originalne riječi a_2. Drugim riječima, sigurni smo da nema početnog dijela riječi R koji bi se poklapao sa nekim $w_k \neq w_2$. Dakle, nešto malo smo već rastumačili, tj. otkrili smo prvo slovo originalne riječi. Sada posmatramo preostali dio riječi R. Na sličan način otkrivamo drugo slovo originalne riječi. Uzmimo da je drugo slovo a_4. Itd. Vidimo da se dekodiranje vrši po odvojenim koracima, da tokom dekodiranja stalno napredujemo. Kada se polazna data riječ R tako iscrpi onda je dekodiranje okončano tj. sastavili smo originalnu riječ $a_2a_4 \ldots$ (u primjeru).

Moguće je da data riječ R nije ničiji kod. Tada će se tokom primjene maločas opisanog postupka na određenom koraku pokazati (ustanoviti) da R nije ničiji kod.

Maločas izložena rasuđivanja čine ustvari dokaz sljedeće teoreme.

Teorema. Svako prefiksno azbučno kodiranje je uspješno tj. uzajamno jednoznačno tj. dekodiranje je jednoznačno.

Dakle, u slučaju prefiksnog šablona Σ sigurno se neće desiti da neki kod R može da bude rastumačen (rastavljen) na dva načina.

Ima primjera šablona koji su uspješni a da nisu prefiksni. Recimo, azbučno kodiranje iz prvog primjera je uspješno iako nije prefiksno: $w_1 = 0$ je prefiks od $w_2 = 01$. Isto tako, azbučno kodiranje iz drugog primjera je uspješno iako nije prefiksno: $w_1 = b_1$ je prefiks od $w_2 = b_1b_2$.

Kod razmatranja azbučnih kodiranja obično se uzima da svi članovi skupa $\Omega(A)$ podliježu kodiranju tj. predstavljaju podatke. Drugim riječima, uzima se da je $\Omega'(A) = \Omega(A)$.

Za $w_1, \ldots, w_m$ (iz definicije azbučnog kodiranja) kaže se da su kodne riječi ili se kaže da su elementarni kodovi.

Razmotrimo jedan primjer kodiranja. Neka se dekadni zapisi brojeva od 1 do 1000 kodiraju pomoću rimskog brojnog sistema. Da li je to kodiranje azbučno?

Zadatak za vježbu sami. Neka je $A = \{\alpha, \beta, \gamma, \delta, \varepsilon\}$ i B=$\{a, b, c\}$. Azbučno kodiranje zadato je svojim šablonom Σ:

$$\alpha - aa,\ \beta - ab,\ \gamma - cc,\ \delta - cca,\ \varepsilon - bcca.$$

(Kod riječi $\beta\delta\gamma\gamma$ (4 slova) je riječ $abccacccc$ (9 slova), i slično.) Ne zna se da li je ovo kodiranje uzajamno jednoznačno. Dato je više riječi R u azbuci B. Za svaku od tih riječi ispitati da li je ona kod nekog podatka i (ako jeste) da li je ona kod samo jednog podatka. 1) $R = ccabc\,cabcc\,abcc$, 2) $R = bccac\,cabcc\,abcca\,cabcc\,a$, 3) $R = abbcc\,accab\,ccaab\,ab$.

Ili možda sastaviti odgovarajući program za računar.

O jednoj primjeni prefiksnog azbučnog kodiranja kod računara.

Postoji više programa koji služe za tzv. kompresiju ili sabijanje podataka. Ti programi koriste se radi uštede prostora na spoljašnjoj memoriji. Takođe, ti programi obično se koriste prilikom prenošenja podataka kroz računarsku mrežu. Postigne se da se prenošenje brže obavi i pored toga što se u polasku vrši kodiranje a u dolasku dekodiranje. Poruka koja je predmet slanja zauzima u originalnom (nekodiranom) obliku nekoliko hiljada bajta. Zamislimo da je ta poruka – neki tekst zapisan na način ASCII. Sadržaj nekog bajta odgovara slovu `a`, nekog drugog slovu `e`, itd. Učestanost slova u tekstu nije ravnomjerna. Slovima koja se češće pojavljuju dodijele se kraći kodovi, tri ili četiri bita. Znacima koji se rijetko pojavljuju dodjeljuju se elementarni kodovi koji su i duži od osam bita. Vodi se računa da šablon bude prefiksan. Tako nastaje kodirani oblik teksta. Budući da je šablon prefiksan, to očito nisu potrebni nikakvi graničnici koji bi razdvajali slovo od slova. Elementarni kodovi se lijepo rasporede po bajtovima. Kodirani oblik teksta se iz jednog čvora računarske mreže proslijedi u drugi čvor. Moguće je da šablon tj. zaglavlje (header) bude unaprijed definisan. A moguće je da zaglavlje zavisi od originalnog teksta tj. od učestanosti pojedinih znakova u tekstu. Dakle, takav program izbroji učestanosti, pa na osnovu toga izvrši dodjeljivanje (raspodjelu) elementarnih kodova, odnosno izvrši kodiranje. U slučaju takvih programa za kompresiju, pored kodiranog oblika teksta očito treba poslati i izabrani šablon tj. zaglavlje, da bi u dolaznom čvoru mogla da se izvrši dekompresija, odnosno dekodiranje. Standardni komercijalni programi za kompresiju obično postignu koeficijent sažimanja otprilike 2. Huffmanov kod.

14. O ISKAZNOM RAČUNU

Iskazom se naziva rečenica koja ima smisla i za koju se može reći da je istinita ili da je lažna. Tako je rečenica "Zbir uglova trougla jednak je 180 stepeni" primjer istinitog iskaza. A "Trougao ima četiri stranice" je lažan iskaz. Dakle, jedan iskaz ima jednu od dvije moguće vrijednosti istinitosti – ili je istinit ili nije istinit.

Nad iskazima se mogu izvoditi operacije, čime se dobijaju novi (složeni) iskazi. Postoji pet osnovnih operacija, označavaju se sa $\&$, $\vee$, $\Rightarrow$, $\Leftrightarrow$ i $\neg$. Neka su p i q dva iskaza. Za iskaz $p\,\&\,q$ se kaže da je konjunkcija polaznih iskaza p i q. Kakva je njegova vrijednost istinitosti? To zavisi od vrijednosti istinitosti njegove dvije komponente. Ako su i jedna i druga komponenta (ako su i p i q) istiniti iskazi onda je i iskaz $p\,\&\,q$ istinit, dok je u svim ostalim slučajevima taj iskaz $p\,\&\,q$ neistinit. Ovim je konjunkcija definisana. Znak $\&$ čita se "i". Umjesto oznake $p\,\&\,q$ koriste se i oznake $p \wedge q$, $p \cdot q$ i $p\,q$.

Disjunkcija $p \vee q$ dva iskaza je istinita kada je bar jedna od komponenti istinita (komponente su p i q), a inače je neistinita. Čita se: p ili q.

Implikacija se označava kao $p \Rightarrow q$ ili kao $p \to q$, što se čita: p povlači q, ili: ako p onda q. Iz iskaza p može se pravilnim zaključivanjem (pravilnim izvođenjem) dobiti iskaz q. Tako da implikacija nije istinita jedino ako je p istinit a q neistinit.

Ekvivalencija ili ravnovaljanost dva iskaza označava se kao $p \Leftrightarrow q$ ili kao $p \sim q$, a čita se: p ekvivalentno q. Ovaj novi (složeni) iskaz $p \Leftrightarrow q$ je istinit kada njegove dvije komponente p i q imaju jednaku (imaju jednu te istu) vrijednost istinitosti.

Dosad nabrojane četiri operacije su binarne, jer se novi iskaz gradi od dva polazna iskaza, a preostala peta operacija je unarna, jer se novi iskaz gradi od jednog polaznog iskaza. Dakle, neka je p bilo koji iskaz. Razmotrimo novi iskaz, upravo razmotrimo negaciju iskaza p, u oznaci $\neg p$ ili $\overline{p}$, što se čita kao: nije p, ili kao: ne p. $\overline{p}$ ima različitu (suprotnu) vrijednost istinitosti od p. Drukčije rečeno, kada je p istinit tada je $\overline{p}$ lažan, a kada je p lažan tada je $\overline{p}$ istinit.

Ovako se o iskazima govori na elementarnom nivou, što se može nazvati naivnom ili nestrogom logikom. Prilikom izgradnje matematičke teorije ne možemo upotrebljavati nedovoljno precizne pojmove, kakav je recimo "rečenica koja ima smisla". Matematička teorija gradi se formalno, aksiomatski. Ipak, može se reći da naivna logika izražava zakone ljudskog mišljenja, kao recimo "$p \vee q$ je tačno u slučaju da ...". Tako da će formalna teorija biti izgrađena tako da bude usaglašena sa naivnom logikom. Zato će nam ono što znamo iz naivne logike sigurno koristiti kod proučavanja formalne teorije Bulovih funkcija.

Šta se mijenja prilikom prelaska sa naivne logike na formalnu teoriju? Prestaje da bude značajno da li se iskaz odnosi na trouglove ili se odnosi na pravila koja važe za recimo logaritme ili itd. Za iskaz je jedino važno – da li je on istinit ili nije. Tako da sada p prestaje da bude rečenica. Sada se p svodi na promjenljivu koja može da uzme jednu od samo dvije vrijednosti – istinito ili lažno. Za te dvije vrijednosti još se koriste i drugi nazivi – tačno i netačno, jeste i nije, istinito i neistinito. A koriste se i sljedeće oznake: $\top$ i $\bot$, 1 i 0 (još T i F, TRUE i FALSE).

Sada je recimo konjunkcija postala jednostavno jedna binarna operacija nad skupom $\{0,1\}$. Ona se definiše na formalan način. Budući da je skup $\{0,1\}$ konačan, to njena definicija može da bude izražena tablicom ili može prosto da bude izražena sljedećim spiskom: $0\,\&\,0 = 0$, $0\,\&\,1 = 0$, $1\,\&\,0 = 0$, $1\,\&\,1 = 1$.

U svim oblastima matematike značajne su formule koje su stalno istinite. Za takve formule se kaže da izražavaju jednu teoremu ili da izražavaju jedno tvrđenje. U slučaju da se takva formula odnosi na Bulove funkcije, onda se kaže da je to jedna tautologija. Na primjer, u

matematičkoj analizi imamo da važi jednakost $\sin(\alpha+\beta)=\sin\alpha\cos\beta+\cos\alpha\sin\beta$. Ovome se mogu dodati riječi: za ma kakve α i β. A misli se: za ma kakve realne brojeve α i β. Evo sada primjera iz matematičke logike: $\overline{p\,\&\,q}=\overline{p}\vee\overline{q}$. Ovdje se mogu dodati riječi: za ma kakve p i q. A misli se: za ma kakve p i q iz skupa $\{0,1\}$.

Ilustrujmo na navedrnoj formuli $\overline{p\,\&\,q}=\overline{p}\vee\overline{q}$ njeno interpretiranje uz pomoć naivne logike. Ili njeno provjeravanje uz pomoć naivne logike. Zamislimo da je p sljedeći iskaz: Danas je srijeda, a q: Zbir uglova trougla jednak je 180 stepeni. Posmatrajmo iskaz $p\,\&\,q$. Posmatrajmo dalje negaciju tog iskaza, tj. posmatrajmo iskaz $\overline{p\,\&\,q}$. Ako ja tvrdim da je iskaz $p\,\&\,q$ istinit, a neko drugi tvrdi da to što ja govorim nije tačno, kada će biti da je taj drugi u pravu? Ako danas nije srijeda ili ako je zbir uglova trougla različit od 180 stepeni.

Kada želimo da dokažemo da je formula oblika $\varphi=\psi$ tautologija onda je očito dovoljno da se dokaže sljedeće: lijeva strana je jednaka jedinici ako i samo ako je desna strana jednaka jedinici. Zato što lijeva strana φ uzima samo dvije moguće vrijednosti (nula i jedan), a naravno također i desna strana ψ.

Posebnu analizu zahtijeva implikacija $p\Rightarrow q$. Budući da p i q više nisu rečenice nego su samo binarne promjenljive, to se kontrola – da li je implikacija ispravna, tj. da li je zaključivanje (da iz p slijedi q) pravilno izvedeno, može izvršiti samo formalno.

Implikacija je značajna zato što većina teorema u matematici ima oblik $p\Rightarrow q$. Navedimo jedan primjer teoreme: ako je neki niz monoton i ograničen onda je taj niz konvergentan. Za p se kaže da je pretpostavka, hipoteza. A za q se kaže da je zaključak, tvrđenje. Šta smo mi dužni da uradimo ako želimo da dokažemo da je implikacija ispravna? Drugim riječima, šta smo mi dužni da uradimo kada dokazujemo neku teoremu? U toj situaciji, mi iskaz p uzimamo "zdravo za gotovo", mi uzimamo da je on istinit, mi se uopšte ne pitamo zašto je on istinit, mi prilikom dokazivanja od njega polazimo. Sada, mi smo dužni da dokažemo da je q istinito.

Tako da u slučaju da je $p=0$ imamo sigurno da je $p\Rightarrow q$ ispravno. Kaže se da iz netačnog slijedi bilo šta. Slično, ako je $q=1$ onda je implikacija $p\Rightarrow q$ ispravna. Kaže se da istina slijedi iz bilo čega.

Već je rečeno da je implikacija $p\Rightarrow q$ lažna samo u slučaju da je p istinito a q lažno. Što odgovara sljedećem. Za neko izvođenje se može reći da je neispravno (neko zaključivanje može da se osudi) samo ako se polazeći od pravilnih pretpostavki stiglo do nepravilnog zaključka. Ne može se pravilnim zaključivanjem od pravilne pretpostavke dobiti nepravilan zaključak.

[Pravilo "modus ponens" glasi $(p\ \&\ (p\Rightarrow q))\Rightarrow q$.]

Razmotrimo tvrđenje $p\Rightarrow q$. Kaže se da je p dovoljan (ili siguran) uslov za q. Ili se kaže da je q potreban (ili neophodan ili nužan) uslov za p. Razmotrimo $p\Leftrightarrow q$; uslov p je potreban i dovoljan za uslov q; isto, q je potrebno i dovoljno da bi bilo p (da bi važilo p). Znamo da je $(p\Rightarrow q)\ \&\ (q\Rightarrow p)=p\Leftrightarrow q$.

Na kraju, navedimo 4 primjera formula, odnosno tautologija u kojima se pojavljuje implikacija. Kod analize formula, korisna je šema od maločas: pretpostavka $\Rightarrow$ zaključak.

1. $(p\,\&\,q)\Rightarrow p$. Očito je da je pretpostavka "jača" od zaključka. Drugim riječima, uzimamo "zdravo za gotovo" da je iskaz $p\,\&\,q$ istinit. Iz toga slijedi da je p (kao komponenta tog iskaza) jedan istinit iskaz; znamo definiciju konjunkcije. A to je i trebalo da se dokaže (da je $p=1$).

2. $p\Rightarrow(q\Rightarrow(p\,\&\,q))$. Na račun prve implikacije uzimamo da je njena pretpostavka p istinita a dužni smo da dokažemo da je istinit njen zaključak $q\Rightarrow(p\,\&\,q)$. Ono što smo dužni da dokažemo je opet jedna implikacija, pa smatramo da je njena pretpostavka q istinita a obavezni smo da dokažemo njen zaključak koji glasi $p\,\&\,q$. Dakle, već imamo da je p istinito i da je q istinito, a to dvoje daje očito da je i $p\,\&\,q$ istinito, čime je dokaz završen.

3. $(p \Rightarrow q) \Leftrightarrow (\neg q \Rightarrow \neg p)$ ili svejedno $(p \Rightarrow q) = (\neg q \Rightarrow \neg p)$, zakon kontrapozicije, koji se često koristi prilikom dokazivanja teorema.

Na primjer, p – zadnja cifra nekog broja je 0, q – taj broj je paran. Implikacija $p \Rightarrow q$ je istinita.

4. $(p \Rightarrow q) \Leftrightarrow ((p \,\&\, \neg q) \Rightarrow \bot)$ ili $(p \Rightarrow q) \Leftrightarrow ((p \,\&\, \neg q) \Rightarrow (r \,\&\, \neg r))$, dokazivanje svođenjem na apsurd.

[Osnovna jednakost za implikaciju glasi $p \Rightarrow q = \neg p \vee q$.]

Zaključak. Mi ćemo ubuduće teoriju Bulovih funkcija da izgradimo aksiomatski.

U matematičkoj logici se dalje umjesto iskaza $p \in \{0,1\}$ posmatra predikat $p(n) \in \{0,1\}$, gdje promjenljiva n prolazi kroz neki skup, recimo kroz skup N. A posmatraju se onda i izrazi oblika $(\forall n)\, p(n)$ ili oblika $(\exists n)\, p(n)$, recimo $(\forall n \in N)\, n + 1 > n$ ili $(\exists n)\, n + n = 4$. Primjer teoreme iz predikatskog računa prvog reda je $\neg((\forall n)\, p(n)) = (\exists n)\, \neg p(n)$. Znamo da $\forall n$ znači za svako n. Znamo da $\exists n$ znači postoji n.

15. KARAKTERISTIKE IDEALNOG LOGIČKOG ELEMENTA

Biće riječi o osnovnim karakteristikama tzv. idealnog logičkog elementa, u digitalnoj elektronici.

Logički element je fizički uređaj čija jednačina odgovara jednoj Bulovoj funkciji. To je elementarno kolo, od takvih kola grade se složenija kola.

Na primjer, i–kolo ima dva ulaza a i b i ima izlaz y, s tim da je $y = a \,\&\, b$. Na primjer, invertor ima jedan ulaz a i ima izlaz y, s tim da je $y = \overline{a}$.

Logički element je digitalno kolo koje odražava jednu logičku (Bulovu) funkciju, na primjer odražava konjunkciju. Logički elementi su sastavni djelovi složenijih digitalnih kola, oni su sastavni djelovi kombinacionih mreža; primjer kombinacione mreže je sabirač. Može se govoriti o idealnom i o realnom logičkom elementu. Zamišljeni ili idealni logički element se skoro ne razlikuje od odgovarajuće logičke funkcije, jedino što ima svoj grafički simbol. Fizički ili realni logički elementi proučavaju se u okviru digitalne elektronike.

O idealnom logičkom elementu

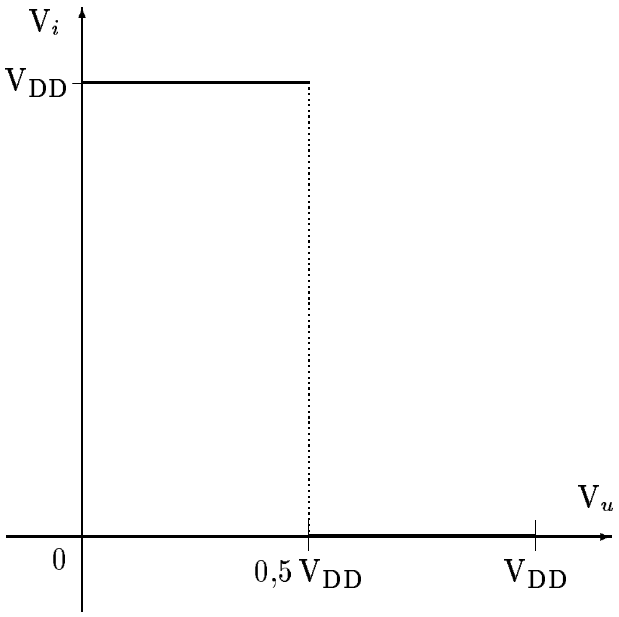
Idealni logički element realizuje jednu unaprijed zadatu logičku funkciju. Logičkoj nuli odgovara napon nula. Logičkoj jedinici odgovara napon napajanja. Izlazni napon zavisi naravno od ulaznog napona (od ulaznih napona), a pod karakteristikom prenosa podrazumijeva se oblik te zavisnosti. Prelazak izlaznog napona sa jednog na drugi nivo izvodi se naglo i to kada je ulazni napon jednak polovini napona napajanja. Pogledajmo o ovome na primjeru invertora. Invertor odgovara Bulovoj funkciji negaciji. Kod invertora: ako je ulazni signal visok onda izlazni treba da bude nizak, i obrnuto. Na slici 1 prikazana je idealna karakteristika prenosa invertora. Na slici su sa V_u i V_i označeni ulazni odnosno izlazni napon. Sa V_{DD} označen je napon napajanja, obično 5V (pet volti). Kao $0 < x < 1/2 \Rightarrow y = y(x) = 1$ i $1/2 < x < 1 \Rightarrow y = y(x) = 0$. Ulazni i izlazni napon zavise od vremena t. Dinamičke karakteristike odnose se na oblik tih zavisnosti. Izlaz se odaziva ulazu tek poslije određenog vremena kašnjenja. Vrijeme potrebno za prelazak iz jednog u drugo logičko stanje je beskonačno kratko; kašnjenje je jednako nula sekundi. Navedimo i druga svojstva idealnog logičkog elementa. Njegova izlazna impedansa je jednaka nuli, a njegova ulazna impedansa je beskonačno velika; impedansa – otpornost; lako saopštava izlaznu vrijednost, a dešavanja unutar elementa ne odražavaju se na ulazne vrijednosti. On nema nikakvu potrošnju. Njegova cijena je nula.

Naravno da idealne karakteristike ne mogu da budu zadovoljene u praksi, bez obzira o

kakvoj tehnologiji se radi. Stoga će biti govora o karakteristikama realnih elemenata koje mogu da budu ostvarene u praksi.

Logičkoj jedinici dosad je odgovarao napon napajanja. Logičkoj jedinici odsad odgovara jedan interval od v_3 do v_4 ili jedna naponska zona (jedan naponski opseg). Ako je naponski nivo (ako je amplitudski nivo) između v_3 i v_4 onda se ta vrijednost napona tumači kao logička jedinica. Slično, i logičkoj nuli (umjesto idealne vrijednosti nula) odsad odgovara jedan interval napona od v_1 do v_2. Naravno da je $v_1 < v_2 < v_3 < v_4$. Za interval od v_2 do v_3 kaže se da obrazuje prelaznu zonu. Signali iz prelazne zone ne predstavljaju ni logičku nulu ni logičku jedinicu, pa prema tome nisu dozvoljeni kao rezultat u normalnom radu digitalnog kola.

Zavisno od familije logičkih kola, neutralni naponski nivo (napon nula, uzemljenje) ne mora da se odnosi ni na logičku nulu ni na logičku jedinicu.



Slika 1. Idealna karakteristika prenosa invertora

Od idealnog prema želji. Svako digitalno logičko kolo treba da zadovoljava bar neke osobine idealnih logičkih kola. Navedimo glavne uslove ili želje:

1. Izlazni signal mora da bude unaprijed definisana jednoznačna funkcija ulaznih signala. Ta funkcija i predstavlja logičku funkciju kola.

2. Karakteristika prenosa ulaz–izlaz treba da bude jako nelinearna. Drugim riječima. karakteristika prenosa u prelaznoj zoni treba da bude strma što je više moguće. Tako da su onda normalni nivoi izlaznog napona skoncentrisani u dvije uske oblasti, u dva logička nivoa.

3. Prolaskom kroz logičko kolo nastaje regeneracija amplitudskih nivoa. Drugim riječima, nove ulazne vrijednosti svaki put će izazvati postavljanje izlaza na odgovarajući nivo, kada prođe vrijeme odziva kola.

4. Logičko kolo treba da ima osobine jednostranosti i usmjerenosti. Promjena na izlazu kola ne treba da izaziva nikakvu naknadnu promjenu na ulazima kola. Signali se prostiru od ulaza prema izlazu.

5. Logičko kolo može da ima više ulaznih priključaka. Izlazni signal logičkog kola može da bude razveden, odnosno može da bude poslat kao ulaz na više narednih kola.

Od želje prema stvarnom. Polazeći od osobina idealnog logičkog elementa i poželjnih karakteristika realnih logičkih elemenata izvedeni su neki pokazatelji mjere vrijednosti stvarnih logičkih elemenata, odnosno definisane su njihove glavne ili osnovne karakteristike na koje treba obratiti pažnju. O tome će biti riječi u nastavku.

16. KARAKTERISTIKE REALNOG LOGIČKOG ELEMENTA

Biće riječi o osnovnim karakteristikama realnog (stvarnog) logičkog elementa, u digitalnoj elektronici. Biće nabrojani parametri (pokazatelji) koji mjere kvalitet (vrijednost) takvog elementa.

O realnim logičkim elementima

Navedimo osnovne karakteristike realnih logičkih elemenata.

1. Karakteristike prenosa

Karakteristike prenosa realnih logičkih elemenata samo aproksimiraju idealnu karakteristiku sa slike 1. Na slici 2 prikazana je tipična karakteristika prenosa realnog invertorskog kola. Uočavaju se dvije bitne razlike između idealne i realne karakteristike prenosa. Prvo, prelazak sa jednog na drugi logički nivo nije jasno definisan, već postoji prelazna zona između stanja logičke nule i logičke jedinice. Drugo, nivo logičke nule nije jednak nula volti a nivo logičke jedinice nije jednak naponu napajanja.

Ponovimo da je kod invertora $y(x) = \overline{x}$ kao $y(0) = 1$ i $y(1) = 0$.

Ponovimo da se vrijednost signala može nalaziti u prelaznoj zoni samo tokom promjene logičkog stanja signala (tokom normalnog rada kola).

Značajne su četiri naponske vrijednosti V_{IL}, V_{IH}, V_{OL} i V_{OH} od I, O, L, H inpput, output, low, high ulaz, izlaz, nizak, visok. V_{IL} predstavlja najveći dozvoljeni napon na ulazu kola koji će se smatrati logičkom nulom (najveći dozvoljeni nivo logičke nule na ulazu). V_{IH} predstavlja najmanji dozvoljeni nivo logičke jedinice na ulazu kola. Naponski nivo V_{OL} predstavlja najveći dozvoljeni nivo logičke nule na izlazu kola (garantovana gornja granica za nulu na izlazu). Slično, napon V_{OH} predstavlja najmanji dopušteni nivo logičke jedinice na izlazu kola (garantovana donja granica za 1 na izlazu), izlazni signali iznad V_{OH} smatraju se da su = 1.

Na primjer, može da bude $V_{IL} = 0{,}6V$ $V_{IH} = 1{,}5V$ $V_{OL} = 0{,}1V$ $V_{OH} = 3{,}6V$ gdje je napajanje od 5V. Ove brojne vrijednosti odnose se na kola u bipolarnoj tehnologiji (TTL familija).

Naime, uobičajeno je da se u složenim digitalnim kolima pomoću izlaza jednog logičkog kola pobuđuju ulazi narednih logičkih kola (izlaz iz kola K_1 dovodi se kao ulaz na kolo K_2). Da bi cio niz kola ispravno funkcionisao neophodno je da budu zadovoljeni uslovi $V_{OL} < V_{IL}$ i $V_{OH} > V_{IH}$.

2. Margine šuma

Neželjena promjena naponskog nivoa u čvoru odnosno u tački gdje su logički nivoi bitni (u tzv. logičkim čvorovima kola) naziva se šumom. Ako je amplituda šuma na ulazu logičkog kola mala onda će izlaz biti ispravan. Ako je pak amplituda neželjene promjene na ulazu logičkog kola velika onda ona može da izazove logičku grešku. Pod pojmom margine šuma podrazumijeva se dozvoljena promjena naponskih nivoa na ulazu kola koja neće izazvati nepoželjnu promjenu na izlazu. Amplituda dozvoljene promjene zavisi i od logičkog stanja na ulazu, tako da postoje dvije margine šuma, NM_1 za logičku jedinicu i NM_0 za logičku nulu. Na osnovu ranije rečenog, margina šuma za logičku jedinicu iznosi $NM_1 = V_{OH} - V_{IH}$ a za logičku nulu $NM_0 = V_{IL} - V_{OL}$.

3. Faktor grananja na izlazu (engl. fan–out) N i faktor grananja na ulazu

Ulazna impedansa realnog logičkog kola nije beskonačno velika, a njegova izlazna impedansa nije nula. Zbog toga se prilikom sprezanja logičkih kola (radi obrazovanja složenih digitalnih mreža) pojavljuje problem opterećivanja izlaza.

N pokazuje na koliko mjesta može da se razvede izlazni signal logičkog kola.

Faktor grananja na izlazu kola N jeste broj nezavisnih ulaznih priključaka koji mogu da budu priključeni na izlaz tog kola, a da tako ne budu narušene dozvoljene varijacije logičkih nivoa. Recimo, može da bude $N = 10$ (TTL familija). Kada se računa faktor grananja na izlazu onda treba uočiti da sva kola ne opterećuju jednako prethodno kolo. Zato se u okviru jedne familije logičkih kola definiše tzv. standardno opterećenje pomoću koga se određuje uticaj svakog ulaza na izlaz prethodnog kola.

Faktor grananja na ulazu predstavlja dozvoljeni broj nezavisnih ulaznih priključaka u kolo. U većini slučajeva ograničen je samo praktičnim razlozima kao što su broj nožica na kućištu ili male potrebe za kolima sa velikim brojem ulaza i slično. Ipak, kod nekih familija logičkih kola broj ulaza ograničen je i zbog degradacije električnih karakteristika.

4. Dinamičke karakteristike (vremenske karakteristike, pokazatelji koji se odnose na vrijeme t)

Prelazak iz jednog u drugo logičko stanje kod realnog logičkog kola ne može se obaviti beskonačno brzo. Razlozi za to su višestruki. Prije svega, u svakom kolu postoje kapaciteti. Kao što je poznato iz teorije električnih kola, napon na kapacitetu ne može da se mijenja trenutno već se takve promjene vrše po eksponencijalnom zakonu. Osim toga, struje kroz elemente su konačne. Jačina struje je ograničena zahtjevima da kolo ima što manju disipaciju (potrošnju); da se kolo ne pregrije. Iz tih razloga, promjena nivoa na izlazu kola obavlja se za konačno vrijeme i kasni za promjenom nivoa na ulazu. Posmatrajmo slučaj kada je pobudni signal logičkog invertora idealizovan i predstavlja povorku pravougaonih impulsa. V. sliku 3. Izlazni signal realnog invertora imaće tipičan oblik koji je takođe prikazan na slici 3. Na vremenskim dijagramima izlaznog signala mogu se uočiti karakteristični intervali koji definišu kašnjenje odziva za pobudom. Vrijeme kašnjenja opadajuće ivice t_{pHL} predstavlja vrijeme za koje opadajuća ivica izlaznog signala kasni za pobudom koja ju je izazvala. Definiše se kao vrijeme između trenutka promjene ulaznog signala i trenutka kada izlazni signal opadne do srednje vrijednosti $V_{sr} = (V_{OL} + V_{OH}) / 2$. Vrijeme kašnjenja rastuće ivice t_{pLH} predstavlja vrijeme između trenutka promjene ulaznog signala i trenutka kada izlazni signal dostigne srednju vrijednost napona V_{sr}. Perioda ili ciklično vrijeme signala T na ulazu i izlazu je jedna te ista. Vrijeme kašnjenja mjeri se jedinicama nano–sekundama i može da bude jednako recimo 10 ns. Postoje i druge dinamičke karakteristike.

5. Disipacija

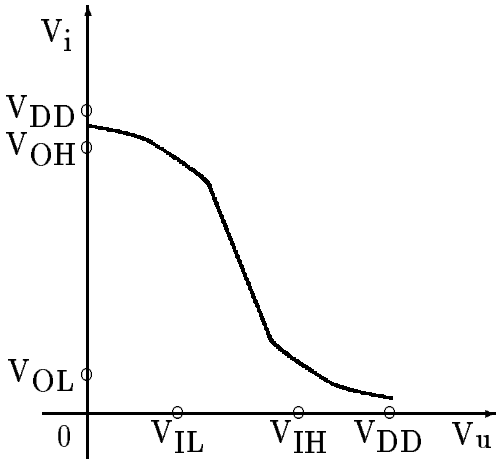
Svako realno logičko kolo mora imati neku potrošnju. Međutim, disipaciju kola nije uvijek lako odrediti jer će se kolo (zavisno od logičkog stanja) nalaziti u različitim uslovima rada. Stoga se pri definiciji disipacije obično uzima da se kolo pobuđuje povorkom pravougaonih impulsa takvih da je trajanje impulsa i pauze jednako. Tada je struja izvora za napajanje jednaka aritmetičkoj sredini struja u jednom i drugom logičkom stanju. Označimo sa V_{DD} napon napajanja i sa I_{DD} struju kroz izvor napajanja. Tako da je $P_D = V_{DD} \cdot (I_{DDmin} + I_{DDmax}) / 2$. Sa P_D označena je prosječna snaga disipacije.

6. Proizvod snage i kašnjenja

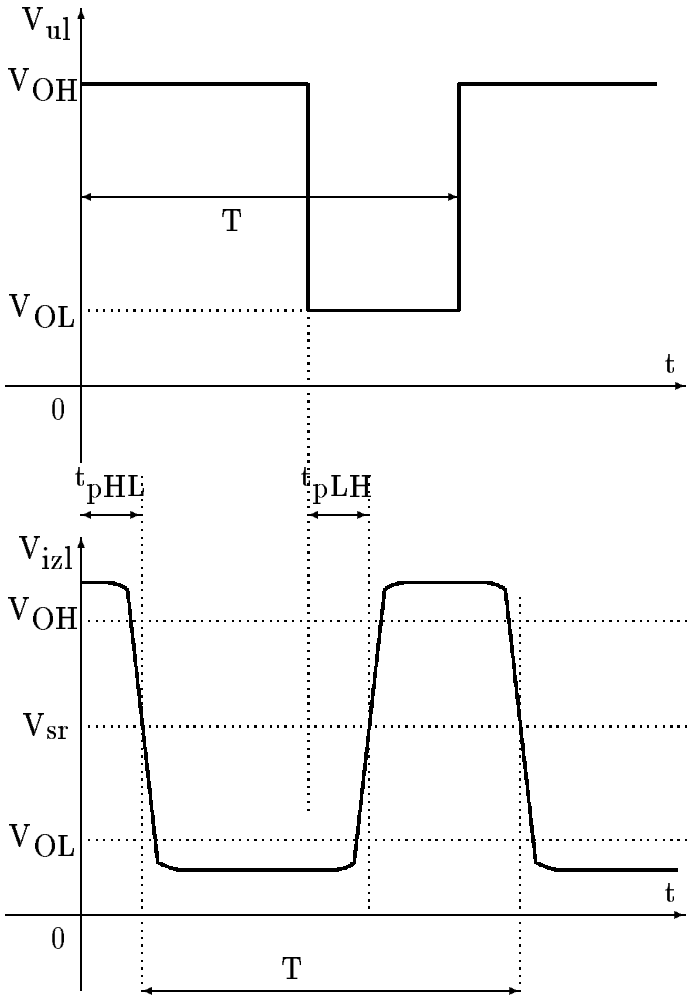
Snaga disipacije P_D logičkog kola obično je povezana sa njegovom najvećom mogućom brzinom rada. Naime, kola čija je brzina veća rade sa većim strujama. Zato se kod njih

nepoželjni kapaciteti brže pune i prazne ili radni režim tranzistora je takav da je disipacija veća. Imajući to u vidu, prilikom projektovanja logičkih kola uvijek se pravi izvjesan kompromis između potrošnje i brzine. Neka t_p označava vrijeme kašnjenja. Da se ocijeni mjera vrijednosti tog kompromisa, definiše se pokazatelj PDP, engl. power delay product ili prevedeno proizvod snage i kašnjenja. Kao što i sama riječ kaže, stavlja se $\mathrm{PDP} = P_D \cdot t_p$. Kompromis je bolji ako je PDP manji. Veličina PDP izražava se očito u jedinicama $\mathrm{W} \cdot \mathrm{sec} = \mathrm{J}$. Savremena logička kola imaju PDP reda veličine pJ, jer su tipične vrijednosti snage disipacije reda mW a tipične vrijednosti vremena kašnjenja su reda ns.

[Vidjećemo da je vrijeme kašnjenja ili odziva logičkog elementa po pravilu znatno manje od taktnog vremena; taktno vrijeme ili perioda takta jednaka je primjera radi 1μs, kada je učestanost takta jednaka 1MHz.]



Slika 2. Realna karakteristika prenosa invertora



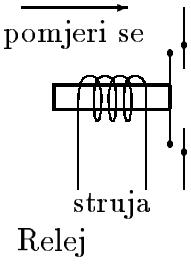
Slika 3. Odziv realnog invertora na idealizovanu pobudu

17. PRIMJERI LOGIČKIH ELEMENATA U DIGITALNOJ ELEKTRONICI

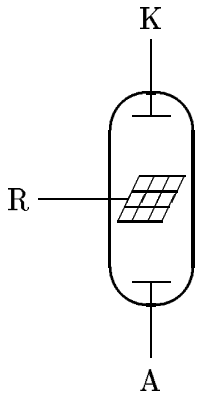
Ovaj naslov sastoji se iz tri dijela: Relej, vakuumska cijev i tranzistor, Logički elementi sastavljeni od tranzistora i Grafički simboli za logičke elemente.

Relej, vakuumska cijev i tranzistor

Kod starih računara kao 0–1 prekidač koristio se relej. Relej ili kalem sastoji se od metalne šipke i oko nje namotaja žice. Kada se kroz namotaje propusti struja onda se stvori elektromagnetno polje i zato se šipka pomjeri, čime se uključi jedan ili više prekidača. S druge strane, kada struja prestane onda se prekidač isključi, jer se šipka vrati u polazni položaj (za ovo se koristi jedna opruga). Prekidač omogućava prolazak struje kroz druge releje, itd. Tako se pružaju mogućnosti za složene kombinacije. Te kombinacije mogu da se upotrebe u automatizaciji ili kod uređaja za računanje. Ulogu releja preuzela je vakuumska cijev. Vakuumsku cijev ili elektronsku lampu čini jedna zatopljena staklena cijev u kojoj je vakuum zajedno sa tri elektrode (kontakta): katoda, anoda i rešetka. Ako se na katodu i anodu primijeni naponska razlika onda struja teče između katode i anode (od katode prema anodi). Neka je sada između katode i anode umetnuta rešetka. Tada se može upravljati strujnim tokom između katode i anode, mijenjajući napon na rešetki između visokog i niskog napona. Dakle, ako za napon na rešetki postoje dvije mogućnosti onda vakuumska cijev djeluje kao prekidač u odnosu na katodu i anodu. Očito, prekidačem se upravlja elektronski. Vakuumska cijev je sa svoje strane zatim zamijenjena tranzistorom, koji se upotrebljava u savremenim digitalnim uređajima (u integrisanim kolima). U nastavku govorimo o principu rada tranzistora.



Relej



Vakuumska cijev

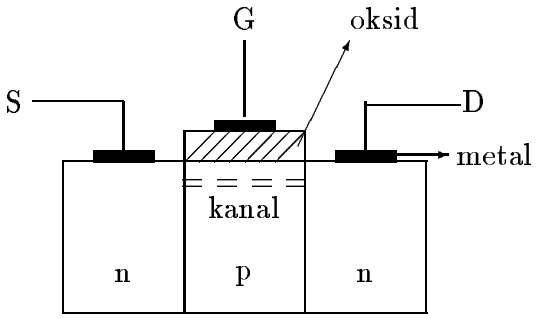
Primjeri provodnika su bakar i aluminijum. Primjer izolatora je plastika. Poluprovodnik je materijal koji se malo razlikuje od provodnika. Sa malim uticajem postiže se da on provodi elektricitet. Primjeri poluprovodnika su germanijum i silikon.

Kako izgleda tranzistor u tehnologiji MOS? Između dva pojasa tipa n stavljen je jedan pojas tipa p. Prilikom izrade tranzistora, poluprovodničkoj podlozi dodaju se primjese. Dodavanjem jedne vrste primjesa stvara se tzv. područje tipa n. Područje tipa n može da sadrži višak elektrona, tj. višak negativnog punjenja, tokom rada tranzistora. Dodavanjem druge vrste primjese obrazuje se područje tipa p, koje ima suprotna svojstva. U njemu elektroni mogu da budu deficitarni ili se kaže da su prisutne tzv. šupljine; kao da ima višak pozitivnog punjenja.

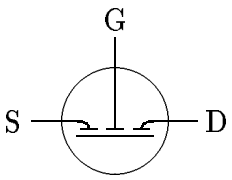
Svaki pojas ima spoljašnji metalni priključak, kontakt, elektrodu. To su sors, gejt i drejn; engl. source gate drain. Samo što je p pojas razdvojen od svoje elektrode (od gejta) jednim slojem SiO_2 (predstavlja izolator).

Pretpostavimo da postoji potencijalna razlika (naponska razlika) između sorsa i drejna. Šta se dešava u tranzistoru? Prva mogućnost: nije primijenjen napon na gejt. Tada gejt djeluje kao prepreka, odnosno srednji pojas sprečava protok elektrona od sorsa ka drejnu. Kroz tranzistor skoro i da ne teče nikakva struja. Druga mogućnost: na gejt je doveden pozitivan napon veći od tzv. prekidnog napona (praga napona); prekidni napon može da bude jednak recimo 1 V. Sada se elektronska vrata otvaraju! Upravo, pod uticajem električnog polja, elektroni u srednjem sloju bivaju privučeni u oblast neposredno ispod oksida, čime se u toj oblasti stvara kanal ili provodni put. Tako da kroz kanal teče struja od sorsa ka drejnu. Znamo da riječ gejt znači vrata.

Prema tome, tranzistor može da se opiše na sljedeći način. Ako je gejt = 0 onda su sors i drejn razdvojeni, a ako je gejt = 1 onda su sors i drejn kratko spojeni.



MOS tranzistor



Simbol MOS tranzistora

Mi smo ustvari govorili o MOS tranzistoru čiji je raspored n–p–n, tako da se obrazuje kanal tipa N; NMOS. Slično, PMOS sa rasporedom p–n–p. Samo se napominje da logička kola iz familije CMOS (C – complementary) imaju bolje karakteristike: brži odziv i manju disipaciju.

Slične okolnosti važe i za tranzistor izrađen u bipolarnoj tehnologiji. Jedino se sada umjesto s, g i d kaže redom emiter, baza i kolektor; engl. emitter base collector. Kada se baza optereti onda struja teče od emitera ka kolektoru.

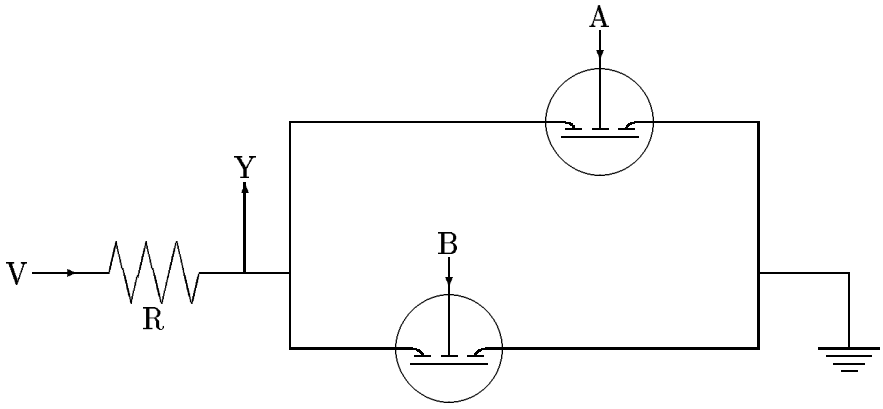
Logički elementi sastavljeni od tranzistora

Sada ćemo vidjeti kako se od tranzistora može sastaviti električno kolo koje odgovara nekoj unaprijed datoj Bulovoj funkciji. Za takvo kolo kaže se da predstavlja logički element. Slika 1 prikazuje kako kombinovanjem dva MOS tranzistora može da bude sastavljeno NILI kolo. Zaista, izvor napajanja V može bilo preko ulaza A bilo preko ulaza B da bude spojen sa uzemljenjem, čime bi se izlaz Y ostavio bez napona; $Y = \overline{A \vee B}$. Izvor napajanja je jedna baterija od 5 V. Zapaziti da otpornik R služi da se izbjegne kratak spoj napajanja V i uzemljenja 
.

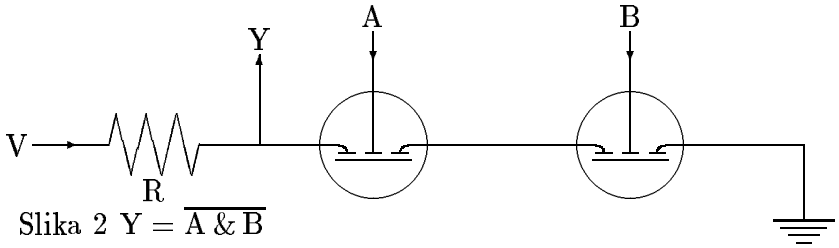
Na sljedećoj slici dva MOS tranzistora više neće biti stavljeni paralelno nego će oni sada biti stavljeni serijski. Slika 2 prikazuje model NI kola; $Y = Y(A, B) = \overline{A \,\&\, B}$.

Posmatrajmo sada slučaj sličnog kola sa samo jednim MOS tranzistorom. Na slici 3 prikazan je invertor, kome odgovara jednačina $Y = \overline{A}$ (negacija). Pogledajmo kako se izlazni napon

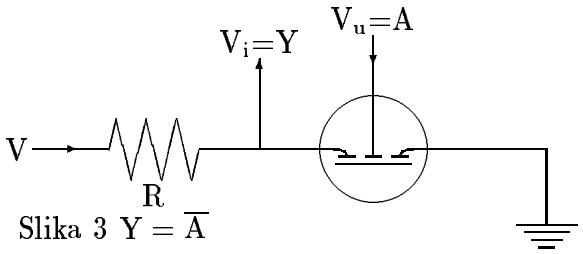
$V_i = Y$ obrazuje u zavisnosti od ulaznog napona $V_u = A$. Ako je ulazni napon visok ($A = 1$) onda tranzistor provodi (tranzistor je uključen), tako da je napajanje spojeno sa zemljom, pa će u tački Y biti nizak napon ($Y = 0$). A ako je $A = 0$ onda tranzistor ne provodi, tako da izlaz postepeno biva preko otpornika opterećen od strane napajanja, tako da će postati $Y = 1$.



Slika 1 $Y = \overline{A \vee B}$



Slika 2 $Y = \overline{A \,\&\, B}$

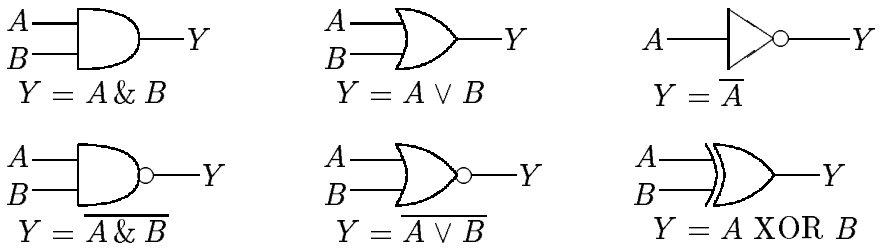


Slika 3 $Y = \overline{A}$

Grafički simboli za logičke elemente

Ubuduće ćemo manje gledati na tehnologiju i familiju u kojoj je logičko kolo izrađeno, a više na njegovu jednačinu. Logički element odgovara nekoj osnovnoj logičkoj operaciji, kao što je već rečeno. Na primjer, konjunkciji odgovara i–kolo ili svejedno i–element.

Na slici 4 prikazani su grafički simboli logičkih elemenata, ima ih šest.



Slika 4

Naziv elementa praktično se poklapa sa imenom operacije, što je prikazano u tabeli:

Y = A & B	I, AND
Y = A ∨ B	ILI, OR
Y = $\overline{\text{A}}$	NE, NOT
Y = $\overline{\text{A \& B}}$	NI, NAND
Y = $\overline{\text{A} \vee \text{B}}$	NILI, NOR
Y = A XOR B	ili A ili B

U posljednjem redu tabele treba dodati: isključivo ili, exclusive or, EX–OR.
Posljednji red tabele može drukčije da se prikaže kako slijedi:

Y = A ⊕ B	sabiranje po modulu 2

ZADACI ZA VJEŽBU,
BULOVE FUNKCIJE,
POSTAVKE ZADATAKA I RJEŠENJA

POSTAVKE ZADATAKA

1. Na osnovu funkcija $f(x_1, x_2)$ i $g(x_3, x_4)$ koje su date vektorski, odrediti vektorski zapis funkcije h: a) $f = 1011$, $g = 1001$, $h(x_2, x_3, x_4) = f(g(x_3, x_4), x_2)$. b) $f = 1011$, $g = 1001$, $h(x_1, x_2, x_3, x_4) = f(x_1, x_2) \vee g(x_3, x_4)$. c) $f = 1000$, $g = 0111$, $h(x_1, x_2, x_3, x_4) = f(x_1, x_2) \,\&\, g(x_3, x_4)$.

2. a) Funkcija $f(x_1, x_2, x_3)$ zadata je na sljedeći način: ona je jednaka 1 bilo za $x_1 = 1$, bilo ako promjenljive x_2 i x_3 imaju istu vrijednost a vrijednost promjenljive x_1 je manja od vrijednosti promjenljive x_3 (u ostalim slučajevima ova funkcija je jednaka nuli). Sastaviti tablicu funkcije f. b) Sastaviti tablicu funkcije f koja je definisana na sljedeći način: ova funkcija $f(x_1, x_2, x_3, x_4)$ je jednaka nuli samo za one (x_1, x_2, x_3, x_4) za koje važi nejednakost $x_1 + x_2 > x_3 + 2x_4$.

3. Funkcija je definisana pomoću formule, a treba sastaviti tablicu te funkcije: a) $(x \to y) \oplus ((y \to z) \oplus (z \to x))$. b) $\overline{(x \vee y) \vee x\overline{z}} \uparrow (x \sim y)$. c) $\overline{x} \to (\overline{z} \sim (y \oplus xz))$. $\sim$ znači ekvivalenciju tj. znači 1001.

4. Da li je sljedeća formula možda identički istinita (uvijek je $= 1$) ili je možda identički lažna (uvijek je $= 0$): a) $(x \to y) \to ((x \vee z) \to (y \vee z))$. b) $((x \oplus y) \sim z)(x \to yz)$. c) $((x \vee \overline{y})z \to ((x \sim z) \oplus y))(x(yz))$.

5. Ispitati da li važe sljedeće jednakosti: a) $x \vee (y \sim z) = (x \vee y) \sim (x \vee z)$. b) $x \to (y \sim z) = (x \to y) \sim (x \to z)$. c) $x \,\&\, (y \sim z) = (x \,\&\, y) \sim (x \,\&\, z)$. d) $x \to (y \vee z) = (x \to y) \vee (x \to z)$. e) $x \to (y \,\&\, z) = (x \to y) \,\&\, (x \to z)$. f) $x \oplus (y \to z) = (x \oplus y) \to (x \oplus z)$. g) $x \to (y \to z) = (x \to y) \to (x \to z)$.

6. Realizovati funkciju f pomoću formule nad skupom simbola funkcija S: a) $f = x \to y$, $S = \{\neg, \vee\}$. b) $f(x, y) = x \vee y$, $S = \{\to\}$. c) $f(x, y) = x \sim y$, $S = \{\,\&\,, \to\}$.

7. Uz korišćenje spiska poznatih elementarnih ekvivalentnosti, dokazati da su formule φ i ψ ekvivalentne: a) $\varphi = (\overline{x} \,\&\, \overline{z}) \vee (x \,\&\, y) \vee (x \,\&\, \overline{z})$, $\psi = x \,\&\, \overline{\overline{y} \,\&\, z} \vee \overline{x} \,\&\, \overline{z}$. b) $\varphi = (x \to y) \to ((x \,\&\, \overline{y}) \oplus (x \sim \overline{y}))$, $\psi = (x \vee y) \,\&\, (\overline{x} \vee \overline{y})$.

8. Neka funkcija $f(x, y)$ iz P_2 zadovoljava relaciju $f(f(x, f(y, z)), f(f(x, y), f(x, z))) = 1$. Dokazati da tada važe sljedeće dvije jednakosti: i) $f(x, x) = 1$, ii) $f(x, f(y, x)) = 1$.

9. Nabrojati suštinske promjenljive funkcije: a) $f(x_1, x_2, x_3) = (x_1 \to (x_1 \vee x_2)) \to x_3$. b) $f(x_1, x_2) = (x_1 \vee x_2) \to \overline{x_2}$. c) $f(x_1, x_2, x_3, x_4) = (x_1 \vee x_2 \vee \overline{x_2}x_3 \vee \overline{x_1}\,\overline{x_2}\,\overline{x_3})x_4$.

10. Pokazati da je x_1 fiktivna promjenljiva za funkciju f (izraziti f formulom u kojoj se x_1 ne pojavljuje): a) $f(x_1, x_2, x_3) = (((x_3 \to x_2) \vee x_1)(x_2 \to x_1)x_3\overline{x_1}) \oplus x_3$. b) $f(x_1, x_2, x_3) = ((x_1 \vee x_2)(x_1 \vee \overline{x_3}) \to (\overline{x_1} \to x_2\overline{x_3}))x_2$.

11. Odrediti fiktivne promjenljive funkcije f. A zatim konstruisati njoj jednaku funkciju čije su sve promjenljive suštinske. a) $f(x_1, x_2, x_3) = 11110000$. b) $f = 00110011$. c) $f = 00111100$.

12. Da li je funkcija g dualna funkciji f: a) $f = x \oplus y$, $g = x \sim y$. b) $f = x \to y$, $g = y \to x$. c) $f = xy \vee xz \vee yz$, $g = xy \oplus xz \oplus yz$. d) $f = x \oplus y \oplus z$, $g = x \oplus y \oplus z$. e) $f = \overline{x}\,\overline{y}\,z \vee x(y \sim z)$, $g = 01101101$.

13. Koristeći princip dualnosti, konstruisati formulu koja realizuje funkciju dualnu funkciji f. Dobijeni izraz uprostiti (zapisati u DNF ili u obliku polonoma Žegalkina). a) $f = xy \vee yz \vee xt \vee zt$. b) $f = x \cdot 1 \vee y(zt \vee 0) \vee \overline{x}yz$. c) $f = (x \vee y \vee (y\overline{z} \oplus 1)) \to 1$.

14. Da li je funkcija f samodualna: a) $f = \overline{(x \to y) \to xz} \to (y \to z)$. b) $f = (\overline{x} \vee y \vee \overline{z})t \vee \overline{x}y\overline{z}$. c) f=0001 0010 0110 0111.

15. Dokazati da ne postoji samodualna funkcija koja suštinski zavisi od dvije promjenljive.

16. Pomoću ekvivalentnih transformacija prevesti formulu u DNF: $F = (x_1 \vee x_2\overline{x_2})(x_1 \vee x_3)$.

17. Predstaviti funkciju u obliku SDNF: a) $f(x_1, x_2, x_3) = (x_1 \oplus x_2) \to x_2x_3$. b) $f(x_1, x_2, x_3)$ = 01101100. c) $f(x_1, x_2, x_3) = 10001110$. d) $f(x_1, x_2) = x_1 \uparrow x_2$.

18. Pomoću tramsformacija oblika $A = Ax \vee A\overline{x}$ i $A \vee A = A$ preći na SDNF: a) $f(x_1, x_2, x_3) = x_1 \vee \overline{x_2}x_3$. b) $f(x_1, x_2, x_3) = x_1\overline{x_2} \vee \overline{x_1}x_3$. c) $f(x_1, x_2, x_3) = x_1 \vee \overline{x_1}x_2 \vee \overline{x_2}x_3$.

19. Pomoću relacija oblika $x \vee yz = (x \vee y)(x \vee z)$ prevesti funkcije iz prethodnog zadatka u KNF.

20. Dokazati formulu: a) $\overline{x} = x \oplus 1$. b) $(x \oplus y)z = xz \oplus yz$.

21. Koliko puta data funkcija ima vrijednost 1? a) $f(x_1, x_2, \ldots, x_n) = x_1 \ldots x_k \oplus x_{k+1} \ldots x_n$. b) $f(x_1, x_2, \ldots, x_n) = 1 \oplus x_1 \oplus x_1x_2 \oplus \ldots \oplus x_1x_2 \ldots x_n$.

22. Dokazati: $x_1 \oplus x_2 \oplus \ldots \oplus x_n = 1$ ako i samo ako neparan broj promjenljivih x_1, x_2, ..., x_n ima vrijednost 1. Zato se i kaže "po modulu 2".

23. Metodom neodređenih koeficijenata, odrediti polinom Žegalkina za funkciju: a) f = 1001. Uputstvo: Napisati opšti oblik za $n = 2$, pa prvo odrediti a, zatim b i c, na kraju d. b) f = 01101000. Uputstvo: Prvo odrediti a, zatim b, c i d, onda e, f i g, na kraju h. Za određivanje a služi $f(0, 0, 0)$. Za određivanje b služi $f(1, 0, 0)$, već je poznato a. Za određivanje c služi $f(0, 1, 0)$, već su poznati a i b. Itd.

24. Dokazati da je funkcija $f(x_1, x_2, x_3) = x_1x_2 \vee x_1x_3 \vee x_2x_3$ samodualna.

25. Neka $f(x_1, x_2, x_3) \in L \cap S$, $f(0, 0, 0) = f(0, 0, 1)$ i $f(\alpha_1, \alpha_2, \alpha_3) = 0$. Čemu je jednako $f(\overline{\alpha_1}, \overline{\alpha_2}, \alpha_3)$?

26. Dokazati da svaka funkcija iz klase L pripada bar jednoj od klasa T_0, T_1 i S.

27. Svodeći neki sistem funkcija za koji je odranije poznato da je kompletan na zadati sistem funkcija A, dokazati da je zadati sistem funkcija A kompletan. Drugim riječima, izraziti svaku funkciju iz ranijeg skupa kao kompoziciju funkcija iz skupa A (kao superpoziciju u kojoj učestvuju samo članovi skupa A). a) $A = \{xy \oplus z, (x \sim y) \oplus z\}$. b) $A = \{x \to y, \overline{x \oplus y \oplus z}\}$.

RJEŠENJA

1. a) V. tablicu. Odgovor je $h = 11111001$.

2. a) $x_1 = 1$ četiri reda tablice $f = 1$. $0 = x_1 < x_2 = x_3 = 1$ jedan red tablice $f = 1$. Odgovor – v. tablicu.

1. a)

x_2	x_3	x_4	h
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

2. a)

x_1	x_2	x_3	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

3. a)

x	y	z	f
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

3. a) Znamo da je $x \to y = 0$ (netačno) samo u slučaju $x = 1$, $y = 0$, a u ostala tri slučaja je $= 1$ (tačno). Znamo da je $\oplus$ asocijativna, pa umjesto $(x \oplus y) \oplus z$ možemo pisati $x \oplus y \oplus z$. Takođe je komutativna, pa redosljed nije bitan, recimo $x \oplus y \oplus z = y \oplus z \oplus x$. Izraz od više $\oplus$ je $= 1$ kada je neparan broj članova $= 1$. Odgovor – v. tablicu.

4. a) Da li je moguće da ovaj izraz bude $= 0$? Jedino ako je $x \to y = 1$ i $(x \vee z) \to (y \vee z) = 0$. Ovo posljednje je opet implikacija, pa bi moralo $x \vee z = 1$ i $y \vee z = 0$. Posljednje znači $y = 0$ i $z = 0$, a tada je (zbog $x \vee z = 1$) $x = 1$. A tada ne važi ranije $x \to y = 1$. Nije moguće. Odgovor – ova formula je identički istinita.

Drugi način da se ovaj zadatak uradi – pomoću tablice. Treći način – pomoću transformacija kao što je $x \to y = \overline{x} \vee y$. Četvrti način. Analizom po z. U slučaju $z = 0$ ovdje piše $(x \to y) \to (x \to y)$ (što je istinito), a u slučaju $z = 1$ ovdje piše $(x \to y) \to (1 \to 1)$ (što je takođe istinito) ("istina slijedi iz bilo čega").

5. a) Analizom po x. Za $x = 0$ ovo je $y \sim z = y \sim z$, odnosno 1 (tačno). A za $x = 1$ data jednakost se svodi na $1 = 1 \sim 1$, odnosno $1 = 1$. Odgovor – važi, jer se analiza sastoji od dva slučaja koji uzeti zajedno pokrivaju sve moguće slučajeve.

6. a) Odgovor $x \to y = \overline{x} \vee y$. Na lijevoj strani znaka jednakosti pojavljuje se funkcija f koja treba da bude realizovanaa. A na desnoj funkcije iz (dozvoljenog) skupa S. Imamo na raspolaganju funkcije iz skupa S i proizvoljne promjenljive.

b) Odgovor je $x \vee y = (x \to y) \to y$.

c) Odgovor je $x \sim y = (x \to y)\,\&\,(y \to x)$, ekvivalencija je isto što i implikacija u jednom i u drugom smjeru.

7. a) Želimo da i formula ψ izgleda kao $\ldots \vee \ldots \vee \ldots$. Pri kraju formule ψ dio $\overline{x}\,\&\,\overline{z}$ može da bude u zagradi, jer & ima veći prioritet. Na dio $\overline{\overline{y}\,\&\,z}$ primijenimo formulu za negaciju konjunkcije:

$\psi = x\,\&\,\overline{\overline{y}\,\&\,z} \vee \overline{x}\,\&\,\overline{z} = (x\,\&\,(y \vee \overline{z})) \vee \overline{x}\,\&\,\overline{z} = (x\,\&\,y) \vee (x\,\&\,\overline{z}) \vee (\overline{x}\,\&\,\overline{z})$

Primijenili smo i poznati obrazac koji govori da je $a\,\&\,(b \vee c)$ isto što i $(a\,\&\,b) \vee (a\,\&\,c)$. Dokazano je da su dvije navedene formule ekvivalentne, jer smo izraz za ψ doveli do poklapanja sa izrazom za φ.

8. U datoj relaciji stavimo $(x, y, z) = (0, 0, 0)$. Ako dopustimo da je $f(0, 0) = 0$ onda ćemo dobiti kontradikciju. Znači da je $f(0, 0) = 1$. Zatim se usmjeravamo na $f(0, 1)$ uz pomoć $(x, y, z) = (0, 0, 0)$ i uz pomoć $(x, y, z) = (0, 1, 1)$. Dalje sami nastavite rješavanje ovog zadatka.

9. a) Dio izraza (date funkcije) $x_1 \to (x_1 \vee x_2)$ je uvijek istinit. Zato što pretpostavka x_1

uvijek povlači zaključak $x_1 \vee x_2$; ovom drugom $x_1 \vee x_2$ je "lakše" da bude istinit od ovog prvog x_1. U ovo možemo da se uvjerimo i pomoću transformacija: $x_1 \to (x_1 \vee x_2) = \overline{x_1} \vee (x_1 \vee x_2) = 1 \vee x_2 = 1$; $x_1 \vee \overline{x_1} = 1$ disjunkcija između nekog iskaza i negacije tog istog iskaza je uvijek istinita; $1 \vee x_2 = 1$ disjunkcija između istine i bilo kog iskaza je sigurno istina.

Tako da se dati izraz svodi na $1 \to x_3$, što se dalje svodi na prosto x_3. Dobili smo da je $f = x_3$. Vidimo da x_3 utiče na vrijednost funkcije. Odgovor – samo x_3 je suštinska promjenljiva funkcije f.

Drugi način da se ovaj zadatak uradi. Sastaviti tablicu date funkcije. Zatim provjeriti da li uvijek važi da je $f(0, x_2, x_3) = f(1, x_2, x_3)$, da li sva četiri puta važi. Ako jeste onda je x_1 fiktivna (a ako nije onda je x_1 suštinska). Slično ustanoviti da li je x_2 fiktivna ili suštinska. A slično naravno i za x_3.

10. a) $f(x_1, x_2, x_3) = (((x_3 \to x_2) \vee x_1)(x_2 \to x_1)x_3\overline{x_1}) \oplus x_3$
eliminišemo jednu i drugu implikaciju po poznatom obrascu $x \to y = \overline{x} \vee y$ (osnovni obrazac za implikaciju)

$f = ((\overline{x_3} \vee x_2 \vee x_1)(\overline{x_2} \vee x_1)x_3\overline{x_1}) \oplus x_3$
izvršimo "množenje" sa $x_3\overline{x_1}$, pri čemu znamo da je nekorisno da se piše član koji sadrži dio oblika $x\overline{x}$

$f = (\overline{x_1}x_3x_2(\overline{x_2} \vee x_1)) \oplus x_3$
dio ispred znaka $\oplus$ jednak je nuli, jer da bi on bio jednak jedinici trebalo bi da je istovremeno $\overline{x_1} = 1$ i $x_3 = 1$ i $x_2 = 1$ i $\overline{x_2} \vee x_1 = 1$

$f = 0 \oplus x_3 = x_3$. Odgovor je $f = x_3$.

11. a) Napisati prvo tablicu date funkcije. Odatle je očito $f = \overline{x_1}$. Odgovor – fiktivne za f su x_2 i x_3, $f = g$ gdje je $g(x_1) = \overline{x_1}$.

12. a)

x	y	f
0	0	0
0	1	1
1	0	1
1	1	0

x	y	g
0	0	1
0	1	0
1	0	0
1	1	1

15.

x	y	f
0	0	p
0	1	q
1	0	$\overline{q}$
1	1	$\overline{p}$

I	II	III	IV
0	0	1	1
0	1	0	1
1	0	1	0
1	1	0	0

12. a) Načinimo tablice za f i g, v. dvije tablice. Da li se, kada se vektor vrijednosti za f preokrene i komplementira, dobije vektor g? Odgovor – da.

b) Odgovor je ne.

13. a) $f = xy \vee yz \vee xt \vee zt = (x \vee z)y \vee (x \vee z)t = (x \vee z)(y \vee t)$. Znamo da je $(a \vee b)^* = ab$, $(ab)^* = a \vee b$, kao i princip dualnosti: $(f(f_1, f_2))^* = f^*(f_1^*, f_2^*)$; f_1 mi je $x \vee z$. Tako, $f^* = xz \vee yt$, što i predstavlja odgovor.

14. a) Prvo dva puta koristimo osnovni obrazac za implikaciju. Koriste se i obrasci za negaciju konjunkcije i za negaciju disjunkcije (De Morganovi obrasci):

$\overline{(x \to y) \to xz} = \overline{(\overline{x} \vee y) \to xz} = \overline{\overline{\overline{x}} \vee y} \vee xz = \overline{x\overline{y} \vee xz} = (\overline{x} \vee y)(\overline{x} \vee \overline{z}) = \overline{x} \vee \overline{x}\,\overline{z} \vee y\overline{x} \vee y\overline{z} = \overline{x} \vee y\overline{z}$

Zato što pored $\overline{x}$ nema koristi da se u disjunkciji piše član $\overline{x}\,\overline{z}$ (sadrži dodatni faktor $\overline{z}$), a takođe ni $y\overline{x}$ (dodato je y). Drugim riječima, važi formula $a \vee ab = a$, koja se lako dokazuje. Tako:

$f = (\overline{x} \vee y\overline{z}) \to (y \to z) = \overline{\overline{x}} \vee y\overline{z} \vee (y \to z) = x\overline{y} \vee xz \vee \overline{y} \vee z = \overline{y} \vee z$.

Iz ovoga što smo dobili da je $f(x, y, z) = \overline{y} \vee z$ lako slijedi da funkcija f nije samodualna, jer ne ispunjava uslov $f(\overline{x}, \overline{y}, \overline{z}) = \overline{f}(x, y, z)$. Odgovor: nije.

b) Odgovor je – jeste.

c) Odgovor je – nije.

15. Ako je f samodualna onda je dovoljno da se odaberu samo dvije njene vrijednosti i to $f(0,0)$ i $f(0,1)$ pa da time čitava funkcija f bude definisana, odnosno da budu definisane sve četiri njene vrijednosti. Kako odabrati $p = f(0,0)$ i $q = f(0,1)$. Za ovo postoje četiri mogućnosti. V. sliku. Prva mogućnost (da i p i q budu nula) ustvari daje funkciju $f = x$, a za tu funkciju je očito da suštinski zavisi od samo jedne promjenljive. Slično i dalje: II $f = y$, III $f = \overline{y}$, IV $f = \overline{x}$. Dokaz je završen.

16. $F = (x_1 \vee x_2\overline{x_2})(x_1 \vee x_3) = (x_1 \vee 0)(x_1 \vee x_3) = x_1(x_1 \vee x_3) = x_1$, poznat je obrazac $a(a \vee b) = a$, koji se uostalom lako dokazuje (provjerava).

17. b) $f = 01101100 = \overline{x_1}\,\overline{x_2}x_3 \vee \overline{x_1}x_2\overline{x_3} \vee x_1\overline{x_2}\,\overline{x_3} \vee x_1\overline{x_2}x_3$. Prvo se po datom vektoru vrijednosti napiše tablica date funkcije (v. tablicu), a u drugom koraku se po tablici izvede ono što se traži, SDNF. Upravo, na račun svake jedinice kod f doći će po jedan član. Prva jedinica je za $(x_1, x_2, x_3) = (0, 0, 1)$ pa njoj sljeduje $\overline{x_1}\,\overline{x_2}x_3$. Itd.

18. a) $f(x_1, x_2, x_3) = x_1 \vee \overline{x_2}x_3$. Već imamo DNF, a treba da dobijemo SDNF. Treba svaka e. k. (elementarna konjunkcija) da ima puni rang, tj. da ima rang tri, budući da se radi o funkciji od tri promjenljive. Ispod x_1 kriju se $x_1\overline{x_2}\,\overline{x_3}$, $x_1\overline{x_2}x_3$, $x_1x_2\overline{x_3}$, $x_1x_2x_3$, četiri. Ispod $\overline{x_2}x_3$ kriju se $\overline{x_1}\,\overline{x_2}x_3$, $x_1\overline{x_2}x_3$, dva. Ovih četiri plus dva treba sastaviti pomoću $\vee$, a jedan se ponavlja i nećemo ga pisati dvaput. Odgovor je: $f(x_1, x_2, x_3) = x_1\overline{x_2}\,\overline{x_3} \vee x_1\overline{x_2}x_3 \vee x_1x_2\overline{x_3} \vee x_1x_2x_3 \vee \overline{x_1}\,\overline{x_2}x_3$.

19. a) $f(x_1, x_2, x_3) = x_1 \vee \overline{x_2}x_3 = (x_1 \vee \overline{x_2})(x_1 \vee x_3)$. Znamo da se KNF razlikuje od DNF po tome što & i $\vee$ zamijene uloge.

17. b)

x_1	x_2	x_3	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

23.

x_1	x_2	f
0	0	1
0	1	0
1	0	0
1	1	1

20. a), b) Provjerom.

21. a) Posmatrajmo izraz $x_1 \ldots x_k \oplus x_{k+1} \ldots x_n$ ili recimo $x_1x_2x_3 \oplus x_4x_5x_6x_7x_8x_9x_{10}$ i analizirajmo kada je taj izraz jednak 1. Budući da izraz ima oblik $a \oplus b$ trebalo bi da bude kao $1 \oplus 0$ ili kao $0 \oplus 1$. Gledamo prvo za $1 \oplus 0$. Važi da je $x_1 \,\&\, x_2 \,\&\, x_3 = 1$ samo u jednom slučaju (ako su sva tri učesnika jednaka 1), a inače je naravno $= 0$. Dok $x_4x_5x_6x_7x_8x_9x_{10} = 0$ važi u svim slučajevima osim slučaja kada su svi $x_4, \ldots, x_{10}$ jednaki 1, tj. važi u $2^7 - 1$ slučajeva. Zatim gledamo za $0 \oplus 1$. Na konkretnom primjeru $n = 10$, $k = 3$ uočili smo zakonitost. Odgovor je $1 \cdot (2^{n-k} - 1) + (2^k - 1) \cdot 1 = 2^{n-k} + 2^k - 2$.

22. Dokaz se izvodi pomoću principa matematičke indukcije. Za $n = 1$ ovdje piše: $x_1 = 1$ ako i samo ako je jedna promjenljiva x_1 vrijednosti 1. Za $n = 2$ tvrđenje koje treba da bude dokazano glasi: važi $x_1 \oplus x_2 = 1$ tada i samo tada kada je među promjenljivima x_1, x_2 njih neparno jednako 1. I jedno i drugo (i $n = 1$ i $n = 2$) je tačno. Npr. za $n = 2$ vidi tablicu funkcije $\oplus$. Da bismo dokazali za $n > 2$, izvršimo indukcijski korak (prelazak sa n na $n + 1$).

Indukcijski korak se lako sprovede. To se u osnovi svodi na sljedeće: paran broj + 1 = neparan broj, neparan broj + 1 = paran broj.

Na kraju, znači da kada se gleda kolika je vrijednost izraza oblika $a \oplus b \oplus \ldots \oplus c$ samo treba prebrojati koliko je od učesnika $a, b. \ldots, c$ vrijednosti 1. Ako neparno mnogo onda je izraz $= 1$, ako parna količina onda je čitav izraz $= 0$.

23. a) V. tablicu funkcije f. Opšti oblik polinoma po modulu dva (polinoma Žegalkina) za $n = 2$ glasi u opštim oznakama $f(x_1, x_2) = a \oplus a_1 x_1 \oplus a_2 x_2 \oplus a_{12} x_1 x_2$ a prostija nam je oznaka $f(x_1, x_2) = a \oplus b x_1 \oplus c x_2 \oplus d x_1 x_2$.

$1 = f(0,0) = a$; iz tablice se vidi da je $f(0,0) = 1$; iz ranijeg predstavljanja se vidi da je $f(0,0) = a$; ostale sabirke iz tog predstavljanja ne pišemo jer konjunkcija u kojoj je bar nešto jednako nuli "propada"; pa $a = 1$

$0 = f(1,0) = 1 \oplus b$ pa je zato $b = 1$

$0 = f(0,1) = 1 \oplus c$ tako da nalazimo $c = 1$ (iskoristili smo to što su a i b već određeni)

$1 = a \oplus b x_1 \oplus c x_2 \oplus d x_1 x_2 = 1 \oplus 1 \oplus 1 \oplus d$ pa $d = 0$ (gledamo da li ima paran ili neparan broj jedinica, vidi prethodni zadatak)

Odgovor: $f(x_1, x_2) = 1 \oplus x_1 \oplus x_2$.

b) Opšti oblik polinoma Žegalkina za $n = 3$ glasi $f(x_1, x_2, x_3) = a \oplus b x_1 \oplus c x_2 \oplus d x_3 \oplus e x_1 x_2 \oplus f x_1 x_3 \oplus g x_2 x_3 \oplus h x_1 x_2 x_3$. Bitan je redosljed određivanja koeficijenata $a, b, \ldots, h$, taj redosljed se ne poklapa sa redosljedom čitanja tablice. Prvo se određuje koeficijent a, drugo oni uz koje stoji jedan faktor, zatim oni uz koje stoje dva faktora, na kraju h. Iz konstrukcije se vidi da polinom postoji i da je jedinstven. Dalje sami riješite ovaj zadatak.

24. Dokazuje se bilo po definiciji samodualnosti, bilo uz korišćenje principa dualnosti (naći izraz za dualnu funkciju f^*).

25. f je linearna $\Rightarrow f = a \oplus a_1 x_1 \oplus a_2 x_2 \oplus a_3 x_3$ (nema nelinearnih članova)

$f(0,0,0) = f(0,0,1) \Rightarrow a_3 = 0$ tj. $f = a \oplus a_1 x_1 \oplus a_2 x_2$ tj. f ne zavisi od x_3

f je samodualna $\Rightarrow f(\overline{\alpha_1}, \overline{\alpha_2}, \overline{\alpha_3}) = \overline{f}(\alpha_1, \alpha_2, \alpha_3) = \overline{0} = 1$

Kako nije bitno čemu je jednako x_3 to je i $f(\overline{\alpha_1}, \overline{\alpha_2}, \alpha_3) = 1$, što i jeste odgovor.

26. Definicije: $f \in T_0 \Leftrightarrow f(0, \ldots, 0) = 0$, $f \in T_1 \Leftrightarrow f(1, \ldots, 1) = 1$.

Dopustimo da postoji funkcija $f = f(x_1, \ldots, x_n)$ takva da $f \in L$, $f \notin T_0$, $f \notin T_1$ i $f \notin S$. $f \notin T_0$ pa $f(0, \ldots, 0) = 1$. $f \notin T_1$ pa $f(1, \ldots, 1) = 0$. $f \in L$ pa $f = a \oplus y_1 \oplus \ldots \oplus y_k$, gdje $y_1, \ldots, y_k \in \{x_1, x_2, \ldots, x_n\}$, $a \in \{0,1\}$, $y_i \neq y_j$ za $i \neq j$, $0 \leq k \leq n$. U vezi $f(0, \ldots, 0) = 1$ imamo $a = 1$, tako da je $f = 1 \oplus y_1 \oplus \ldots \oplus y_k$. U vezi $f(1, \ldots, 1) = 0$ imamo da je broj k neparan. Recimo, $f = 1 \oplus x_6$ ili $f = 1 \oplus x_1 \oplus x_3 \oplus x_7$ ili slično. Dalje:

$\overline{f^*(x_1, \ldots, x_n) = \overline{f}(\overline{x_1}, \ldots, \overline{x_n})} = \overline{1 \oplus \overline{y_1} \oplus \ldots \oplus \overline{y_k}} =$ $\{\ \overline{u} = u \oplus 1\ \}$

$\overline{1 \oplus (y_1 \oplus 1) \oplus \ldots \oplus (y_k \oplus 1)} =$ $\{$ paran broj jedinica se skrati $\}$

$\overline{y_1 \oplus \ldots \oplus y_k} = 1 \oplus y_1 \oplus \ldots \oplus y_k = f(x_1, \ldots, x_n)$

Ovo govori da je $f^* = f$, tj. da $f \in S$. Dobili smo kontradikciju. Dokaz je završen.

27. a) Od dvije date funkcije $xy \oplus z$ i $(x \sim y) \oplus z$ mi ćemo superpozicijama izgraditi funkcije $\overline{x}$ i $x \,\&\, y$, čime će dokaz biti završen, jer znamo da je sistem $\{\overline{x}, x \,\&\, y\}$ kompletan. Prvo, $(x \sim y) \oplus z$ daje $(x \sim x) \oplus z = 1 \oplus z = \overline{z}$, izgradili smo negaciju. Drugo, iz $xy \oplus z$ imamo $x\overline{y} \oplus x$ (imamo pravo da pišemo $\overline{y}$ jer smo maločas izgradili negaciju) $= x\overline{y} \oplus x1 = x(\overline{y} \oplus 1) = xy$, i konjunkciju smo izgradili.

b) Sada se svođenje vrši na kompletni sistem $\{x \vee y, \overline{x}\}$. Sljedeće dvije formule rješavaju postavljeni zadatak: $(x \to y) \to y = x \vee y$, $\overline{x \oplus x \oplus x} = \overline{x}$. Naravno da je korisno da se osvjedočimo da su te dvije formule ispravne. Zapaziti šablon koji je prisutan i u jednoj i u drugoj formuli: na lijevoj strani imamo izraz koji je dozvoljenog oblika (koriste se date funkcije, koriste se proizvoljno imena promjenljivih), a na desnoj strani imamo članove ranijeg kompletnog sistema.

18. POJAM BULOVE FUNKCIJE, FIKTIVNE PROMJENLJIVE

U prvom dijelu ovog naslova navode se primjeri osnovnih ili elementarnih Bulovih ili logičkih funkcija. Onda se još u drugom dijelu govori o suštinskoj i (suprotno) fiktivnoj promjenljivoj Bulove funkcije.

Neka je $E_2 = \{0, 1\}$. Bulovom funkcijom ili funkcijom algebre logike nazivamo funkciju od jedne promjenljive ili od više promjenljivih, čije promjenljive uzimaju vrijednosti iz skupa E_2, a i sama vrijednost funkcije f je iz E_2. Dakle, f: $E_2^n \to E_2$, gdje je n prirodan broj, tj. $n \geq 1$. Još se dopušta i da bude $n = 0$. Funkcija od nula promjenljivih je očito konstanta. Tako da postoje dvije Bulove funkcije koje zavise od nula promjenljivih. Skup svih mogućih Bulovih funkcija označava se sa P_2.

Oblast definisanosti (domen) Bulove funkcije je konačan skup. Ako Bulova funkcija zavisi od n promjenljivih onda njen domen ima 2^n članova. Pojedini član domena očito je jedna uređena n–torka čije su komponente 0 i 1. Kako je domen konačan, funkcija može da bude definisana svojom tablicom. Uvodi se tzv. prirodni redosljed među članovima domena. Ako se redom napišu vrijednosti promjenljivih onda se tako dobija broj $x_1x_2 \ldots x_n$ koji ima n binarnih cifara. Kako se po tablici napreduje, tako ti brojevi $x_1x_2 \ldots x_n$ rastu za po 1, od 0 do $2^n - 1$. U slučaju $n = 3$ tablica ima 8 redova. Redosljed smo definisali samo da bismo mogli da tablicu izrazimo na jedan sasvim sažet način. Naime, dovoljno je da se redom napiše 8 vrijednosti iz skupa E_2, a podrazumijeva se da te vrijednosti znače redom $f(0,0,0)$, $f(0,0,1)$, ..., $f(1,1,1)$. Ovo kada je $n = 3$, a slično za bilo koje n. Pogledajmo jedan primjer (odnosi se na slučaj $n = 3$). Tablica 1 definiše jednu funkciju $f = f(x_1, x_2, x_3)$ od tri promjenljive.

x_1	x_2	x_3	f
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

$f(0,0,0) = 1$
$f(0,0,1) = 1$
$f(0,1,0) = 0$
...

Tablica 1

Na sažet način se ova ista funkcija definiše pomoću svog tzv. vektora ili vektora vrijednosti $f = 11000011$; dužina vektora vrijednosti jednaka je 8.

Sada ćemo navesti devet primjera Bulovih funkcija. Za svaku od tih funkcija kaže se da je elementarna. Definicije svih ovih devet funkcija date su tablicama. Vidimo da prva i druga funkcija imaju po nula promjenljivih, treća i četvrta po jednu promjenljivu, a sve ostale po dvije promjenljive.

a) $f = 0$, konstanta 0
b) $f = 1$, konstanta 1
c) $f(x) = x$, identičko preslikavanje
d) $f(x) = \overline{x}$, negacija od x, "ne x"
e) $f(x_1, x_2) = x_1 \,\&\, x_2$, konjunkcija od x_1 i x_2, "x_1 i x_2"; neki pišu $x_1 \cdot x_2$ ili x_1x_2

f) $f(x_1, x_2) = x_1 \vee x_2$, disjunkcija od x_1 i x_2, "x_1 ili x_2"
g) $f(x_1, x_2) = x_1 \to x_2$, implikacija, "$x_1$ povlači x_2"
h) $f(x_1, x_2) = x_1 \oplus x_2$, ekskluzivno ili, sabiranje x_1 i x_2 po modulu 2
i) $f(x_1, x_2) = x_1 \uparrow x_2$, Šeferova funkcija

f
0

f=0

a)

f
1

f=1

b)

x	f
0	0
1	1

$f(x)$=x

c)

x	f
0	1
1	0

$f(x)$=$\overline{x}$

d)

x_1	x_2	f
0	0	0
0	1	0
1	0	0
1	1	1

$f(x_1, x_2)$=$x_1 \& x_2$

e)

x_1	x_2	f
0	0	0
0	1	1
1	0	1
1	1	1

$f(x_1, x_2)$=$x_1 \vee x_2$

f)

x_1	x_2	f
0	0	1
0	1	1
1	0	0
1	1	1

$f(x_1, x_2)$=$x_1 \to x_2$

g)

x_1	x_2	f
0	0	0
0	1	1
1	0	1
1	1	0

$f(x_1, x_2)$=$x_1 \oplus x_2$

h)

x_1	x_2	f
0	0	1
0	1	1
1	0	1
1	1	0

$f(x_1, x_2)$=$x_1 \uparrow x_2$

i)

Lako se vidi da ima 2^{2^n} Bulovih funkcija koje zavise od n promjenljivih ($n \geq 0$).

Drugi dio. Razmotrimo jednu Bulovu funkciju f od n promjenljivih, $f = f(x_1, \ldots, x_n)$. Neka je k jedan broj, $1 \leq k \leq n$. Za x_k se može postaviti sljedeće pitanje: da li je promjenljiva x_k suštinska (ili svejedno bitna) za funkciju f. Mogući odgovori na postavljeno pitanje su očito da i ne (ili svejedno jeste i nije). Ako je odgovor "ne" onda se još kaže i da je x_k fiktivna (ili svejedno prividna) promjenljiva funkcije f.

Govoreći opisno, neka promjenljiva je suštinska ako ona svojom vrijednošću utiče (makar jednom utiče) na obrazovanje vrijednosti funkcije.

Pogledajmo kao primjer funkciju f definisanu tablicom 1. Lako se vidi sljedeće: kada se izaberu vrijednosti za x_1 i x_2 onda je time već određena vrijednost $f(x_1, x_2, x_3)$, tj. nije ni potrebno da se bira vrijednost za x_3. Zato kažemo da je x_3 fiktivna promjenljiva razmatrane funkcije f. Pogledajmo x_1. U tablici se mogu uočiti dva reda da se razlikuju samo po x_1 (po x_2 i x_3 se ne razlikuju), a da vrijednost funkcije nije istovjetna; dvije vrijednosti funkcije nisu istovjetne zato što dvije vrijednosti x_1 nisu istovjetne. Uočena dva reda su $x_1x_2x_3 = 000$ i $x_1x_2x_3 = 100$; zaista je $f(0,0,0) \neq f(1,0,0)$. Dakle, x_1 je suštinska. A za x_2? I promjenljiva x_2 je suštinska, jer se bar jednom dešava da ona utiče na obrazovanje vrijednosti f.

Dakle, funkcija f ima dvije suštinske promjenljive x_1 i x_2 i ima jednu fiktivnu promjenljivu x_3. Možemo reći da f ustvari zavisi samo od dvije promjenljive. Kako funkcija f ima fiktivnih promjenljivih, to ona može da bude zamijenjena drugom jednom funkcijom g čije će sve promjenljive biti suštinske; g ima dvije promjenljive. Kaže se da je kod funkcije f izvršeno uklanjanje

njene fiktivne promjenljive x_3. Ili se svejedno kaže da je funkcija g nastala od funkcije f postupkom uklanjanja fiktivne promjenljive. Ili da se dodavanjem fiktivne promjenljive funkciji g dobija funkcija f. Funkcija g definisana je tablicom 2.

x_1	x_2	g
0	0	1
0	1	0
1	0	0
1	1	1

$g(0,0) = 1$
$g(0,1) = 0$
$g(1,0) = 0$
$g(1,1) = 1$

Tablica 2

O pojmovima suštinske odnosno fiktivne promjenljive dosad smo govorili pomoću primjera, a slične okolnosti važe i u opštem slučaju, odnosno u sljedećoj definiciji.

Definicija. Za funkciju $f = f(x_1, \ldots, x_n)$ iz P_2 kažemo da na suštinski način zavisi od svoje promjenljive x_k (odnosno da je x_k njena suštinska promjenljiva) ako postoje vrijednosti $\alpha_1, \ldots, \alpha_{k-1}, \alpha_{k+1}, \ldots, \alpha_n \in \{0,1\}$ za promjenljive $x_1, \ldots, x_{k-1}, x_{k+1}, \ldots, x_n$ takve da je $f(\alpha_1, \ldots, \alpha_{k-1}, 0, \alpha_{k+1}, \ldots, \alpha_n) \neq f(\alpha_1, \ldots, \alpha_{k-1}, 1, \alpha_{k+1}, \ldots, \alpha_n)$. U suprotnom slučaju (kada takve vrijednosti ne postoje) kaže se da je x_k fiktivna promjenljiva za f.

Neka su f_1 i f_2 dvije funkcije iz P_2 (dvije Bulove funkcije). Neka se postupkom uklanjanja ili uvođenja fiktivnih promjenljivih funkciji f_1 može dobiti f_2. Drugim riječima, neka se tim postupkom f_1 može dovesti do poklapanja sa f_2. Tada su f_1 i f_2 jednake u nekom smislu. Drukčije rečeno, izraz za f_1 jednak je sa izrazom za f_2, do na oznake promjenljivih. Recimo, za funkcije f i g iz tablice 1 odnosno tablice 2 važi $f(x,y,z) = xy \vee \overline{x}\,\overline{y}$ odnosno $g(a,b) = ab \vee \overline{a}\,\overline{b}$.

19. SUPERPOZICIJA FUNKCIJA, FORMULE

Pojam superpozicije (ili kompozicije) za Bulove funkcije uvodi se na način koji je u matematici običan. Neka imamo funkcije f_1 (ima i_1 argumenata), ..., f_k (ima i_k argumenata) i f_0 (ima k argumenata). Mi definišemo njihovu superpoziciju, odnosno jednu novu funkciju f kao $f(x_1, x_2, \ldots, x_n) = f_0(f_1(a, b, \ldots, c), \ldots, f_k(d, \ldots, e))$ (od a do c ima i_1 argumenata, ..., od d do e ima i_k argumenata), gdje su umjesto $a, \ldots, e$ napisani razni x_j (nisu obavezno različiti). Ili: $f_0(f_1(x_{11}, x_{12}, \ldots, x_{1i_1}), \ldots, f_k(x_{k1}, x_{k2}, \ldots, x_{ki_k}))$.

Primjer: Razmotrimo funkcije f_1, f_2, f_3 i f_4:

x	y	f_1
0	0	1
0	1	0
1	0	1
1	1	0

,

x	y	f_2
0	0	0
0	1	1
1	0	1
1	1	1

,

x	f_3
0	1
1	0

i $f_4 = 11000011$. Tada relacija $f_5(x, y) = f_4(f_1(x, x), f_2(x, y), f_3(y))$ definiše jednu novu funkciju od dvije promjenljive f_5. Recimo, $f_5(0, 0) = f_4(f_1(0, 0), f_2(0, 0), f_3(0)) = f_4(1, 0, 1) = 0$.

Tablica funkcije f_5:

x	y	f_5
0	0	0
0	1	1
1	0	0
1	1	0

Polazimo od jednog datog konačnog ili beskonačnog skupa funkcija $A = \{f_1, f_2, \ldots\}$. Pomoću superpozicija dobijamo nove funkcije. Nastojimo da dobijemo što više novih funkcija; izvršimo sve moguće superpozicije; ako smo jednu funkciju "dobili" onda ona može da učestvuje u idućim superpozicijama. Koje sve funkcije mogu da budu dobijene? Da li mogu da budu dobijene sve funkcije iz P_2? Drugim riječima, da li određena funkcija f može da se izrazi preko funkcija iz polaznog skupa A? Da li svaka Bulova funkcija može da bude izražena preko polaznog sistema funkcija? O ovome će više biti riječi kasnije. Za označavanje argumenata (promjenljivih) obično koristimo članove skupa $X = \{x_1, x_2, x_3, \ldots\}$.

Ako imamo jedan skup funkcija i jednu operaciju nad tim funkcijama onda se kaže da taj skup i ta operacija čine (da ta struktura čini) jednu algebru. Za skup svih Bulovih funkcija P_2 zajedno sa maločas uvedenom operacijom superpozicije kaže se da predstavlja Bulovu algebru ili algebru logike.

Za ranije napisano $f_0(f_1(a, b, \ldots, c), \ldots, f_k(d, \ldots, e))$ kaže se da predstavlja formulu ili izraz. Možda je pogodnije da se umjesto o svim mogućim superpozicijama govori o svim mogućim formulama. Suština ostaje ista, samo se jezik mijenja. Iskažimo ovo precizno.

Definicija. Neka je A jedan skup Bulovih funkcija. a) Ako $f \in A$ onda je $f(y_1, y_2, \ldots, y_n)$ formula, ovdje je f očito funkcija od n promjenljivih, gdje $y_1, y_2, \ldots, y_n \in X$. Ako $f \in A$, gdje je f funkcija od nula promjenljivih, onda je f formula. b) Ako $f \in A$ (f je funkcija od n promjenljivih) i ako su $\varphi_1, \ldots, \varphi_n$ bilo članovi skupa X bilo formule onda je i $f(\varphi_1, \ldots, \varphi_n)$ formula. c) Formule, tj. formule nad A mogu da budu dobijene jedino konačnom i uzastopnom primjenom pravila a) i b).

Svakoj formuli, tj. svakom izrazu se na prirodan način pridružuje jedna funkcija. Recimo, raniji izraz $f_4(f_1(x, x), f_2(x, y), f_3(y))$ definiše jednu funkciju, kako je već opisano. Primjer izraza: $x \oplus f(y, z)$.

Kada se u izrazu pojavljuje funkcija od dvije promjenljive onda se umjesto $f(x, y)$ obično

piše $x\,f\,y$, recimo $x \vee y$. Funkcije koje imaju manje od dvije promjenljive obično se zapisuju kao 0, 1, x i $\overline{x}$. Smatra se da znak & ima veći prioritet od znakova $\vee$, $\oplus$, $\to$ i $\uparrow$, tako da na račun toga u zapisu može da bude jedan par zagrada manje.

Primjer: sabiranje binarnih brojeva. Ako su x_1, x_2 i x_3 Bulove promjenljive onda je $x_3x_2x_1$ trocifren binarni broj i obrnuto. Dva takva broja $x_3x_2x_1$ i $y_3y_2y_1$ jednoznačno definišu zbir $z_4z_3z_2z_1$. Svaka veličina z_k jeste očito funkcija od promjenljivih $x_3, \ldots, y_1$. Sastaviti četiri odgovarajuća izraza.

Da bismo riješili postavljeni zadatak, uvedimo i pomoćne promjenljive p_1, p_2 i p_3 koje označavaju prenose. Upravo, prvo se sabiraju x_1 i y_1, čime se dobijaju z_1 i p_1. Zatim vrijednost p_1 interveniše prilikom sabiranja x_2 i y_2. Itd.

Recimo:
$$\begin{array}{cccc} & 1 & 0 & 1 \\ + & 1 & 1 & 1 \\ \hline 1 & 1 & 0 & 0 \end{array}$$
s tim da je ovdje $p_3 = z_4 = 1$, $p_2 = 1$, $p_1 = 1$, $p_0 = 0$,

dekadno: $7 + 5 = 12$. U opštim oznakama:
$$\begin{array}{ccccc} & & x_3 & x_2 & x_1 \\ + & & y_3 & y_2 & y_1 \\ \hline & z_4 & z_3 & z_2 & z_1 \end{array}$$

Rješenje:

$$z_1 = x_1 \oplus y_1$$

$$p_1 = x_1 \,\&\, y_1$$

$$z_2 = (x_2 \oplus y_2) \oplus p_1$$

$$p_2 = (x_2 \,\&\, y_2) \vee (x_2 \,\&\, p_1) \vee (y_2 \,\&\, p_1)$$

$$z_3 = (x_3 \oplus y_3) \oplus p_2$$

$$z_4 = (x_3 \,\&\, y_3) \vee (x_3 \,\&\, p_2) \vee (y_3 \,\&\, p_2)$$

Kako su dobijene formule za z_k? Veličina z_k zavisi od x_k, y_k i p_{k-1}. Ako je tačno jedna od tri veličine x_k, y_k, p_{k-1} $= 1$ onda je i $z_k = 1$; ili ako su sve tri veličine $= 1$. Veličina p_k takođe je određena veličinama x_k, y_k i p_{k-1}, ali po drugom zakonu. Među x_k, y_k, p_{k-1} barem dva treba da budu $= 1$ da bi bilo $p_k = 1$.

Da bi se rješenje razumjelo, treba znati sljedeće o funkciji $\oplus$: $x_1 \oplus \ldots \oplus x_n = 1$ ako i samo ako među veličinama $x_1, \ldots, x_n$ neparan broj ima vrijednost 1.

Iz formula $z_1 = \ldots$, $\ldots$, $z_4 = \ldots$ mogu da se eliminišu p_1 i p_2, tj. z_1 do z_4 mogu da se izraze preko samo $x_1, x_2, x_3, y_1, y_2, y_3$.

20. GLAVNI IDENTITETI, DUALNA FUNKCIJA

Navedimo glavne jednakosti (identitete) koji važe u Bulovoj algebri. Svaka jednakost može da bude dokazana putem neposredne provjere. Upravo, izračunati vrijednost lijeve odnosno desne strane za sve kombinacije vrijednosti promjenljivih koje se pojavljuju.

1. $0 \,\&\, x = 0 \qquad 0 \vee x = x$

   $1 \,\&\, x = x \qquad 1 \vee x = 1$

   $x \,\&\, x = x \qquad x \vee x = x$

   $x \,\&\, \overline{x} = 0 \qquad x \vee \overline{x} = 1$

2. $x \vee y = y \vee x$ komutativnost disjunkcije slično za & i za $\oplus$

   $(x \vee y) \vee z = x \vee (y \vee z)$ asocijativnost disjunkcije slično za & i za $\oplus$

   distributivni zakoni:

   $(x \vee y) \,\&\, z = (x \,\&\, z) \vee (y \,\&\, z) \qquad (x \,\&\, y) \vee z = (x \vee z) \,\&\, (y \vee z)$

   zakoni apsorpcije:

   $x \vee (x \,\&\, y) = x \qquad x \,\&\, (x \vee y) = x \qquad x \vee (\overline{x} \,\&\, y) = x \vee y \qquad x \,\&\, (\overline{x} \vee y) = x \,\&\, y$

3. $\overline{\overline{x}} = x \qquad x \to y = \overline{x} \vee y$

4. De Morganovi obrasci: $\overline{x \vee y} = \overline{x} \,\&\, \overline{y} \qquad \overline{x \,\&\, y} = \overline{x} \vee \overline{y}$

Definicija. Za dvije formule φ i ψ kažemo da su ekvivalentne (i pišemo $\varphi = \psi$) ako one definišu jednu te istu funkciju, tj. ako važi $f = g$, gdje je f funkcija koju definiše formula φ a g je funkcija koju definiše formula ψ.

Primjeri $0 = x \,\&\, \overline{x} \qquad x \to y = \overline{y} \to \overline{x} \qquad \overline{x_1}(x_2 \oplus x_3) = \overline{x_1 \vee (x_2 \to x_3)(x_3 \to x_2)}$

Kako je konjunkcija asocijativna to ima smisla pisati $x_1 \& x_2 \& x_3$. A takođe $x_1 \& x_2 \& \ldots \& x_n$ što ćemo odsad zapisivati i kao $\&_{i=1}^{n} x_i$. Slično, $\vee_{i=1}^{n} x_i$ znači $x_1 \vee x_2 \vee \ldots \vee x_n$, gdje je $n \geq 1$.

Lako se utvrđuju sljedeća pravila. Ako je u konjunkciji (koja ima više faktora) bar jedan faktor $= 0$ onda je i cijela ta konjunkcija $= 0$. Kao $0 \cdot x \cdot y = 0$. Ako je u konjunkciji neki faktor $= 1$ onda taj faktor može da se izostavi. Kao $x \cdot 1 \cdot x \cdot \overline{y} = x \cdot x \cdot \overline{y}$. Slično za $\vee$. Ako je u disjunkciji bar jedan sabirak $= 1$ onda je cijela ta disjunkcija $= 1$. Kao $1 \vee x = 1$. Iz disjunkcije mogu da se izostave sabirci koji su jednaki nuli (misli se da se time dobija ekvivalentna formula). Kao $0 \vee x = x$.

———————————————

Definicija. Razmotrimo funkciju $f = f(x_1, \ldots, x_n)$. Definišimo jednu novu funkciju $g = g(x_1, \ldots, x_n)$ sljedećom relacijom: $g(x_1, \ldots, x_n) = \overline{f}(\overline{x_1}, \ldots, \overline{x_n})$. Za funkciju g kažemo da je dualna funkciji f. Umjesto g pisaćemo f^*.

Ako je funkcija f data tablicom onda se lako dobija tablica njoj dualne funkcije f^*. Kolona koja definiše vrijednosti funkcije f treba da bude preokrenuta i još treba da bude izvršeno komplementiranje vrijednosti koje čine tu kolonu (0 se pretvori u 1, a 1 se pretvori u 0). Recimo, ako je $f = 11000101$ onda je $f^* = 01011100$, v. dvije tablice. U okviru toga, recimo:

$$f^*(0,0,0) = \overline{f}(\overline{0},\overline{0},\overline{0}) = \overline{f}(1,1,1) = \overline{1} = 0.$$

Možda bi bilo preglednije ako bi se umjesto $\overline{f}(1,1,1)$ pisalo $\overline{f(1,1,1)}$.

x_1	x_2	x_3	f
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

x_1	x_2	x_3	f^*
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Data je funkcija f, odrediti njenu dualnu funkciju. Za neke jednostavne funkcije f, razmatrani zadatak se lako rješava. Ako je $f = 0$ onda je $f^* = 1$. Ako je $f = 1$ onda je $f^* = 0$. Ako je $f = x$ onda je $f^* = x$. Ako je $f = \overline{x}$ onda je $f^* = \overline{x}$. Ako je $f = x_1 \,\&\, x_2$ onda je $f^* = x_1 \vee x_2$. Ako je $f = x_1 \vee x_2$ onda je $f^* = x_1 \,\&\, x_2$. Zaista:

$$f(x_1, x_2) = x_1 \vee x_2$$

$$f^*(x_1, x_2) = \overline{f}(\overline{x_1}, \overline{x_2}) = \overline{\overline{x_1} \vee \overline{x_2}} = x_1 \,\&\, x_2$$

Možemo reći da je formuli $x_1 \vee x_2$ dualna formula $x_1 \,\&\, x_2$.

Koja funkcija je dualna funkciji f^*? Iz definicije se lako dobija da je $(f^*)^* = f$. Zaista:

$$(f^*(x_1, \ldots, x_n))^* = (\overline{f}(\overline{x_1}, \ldots, \overline{x_n}))^* = \overline{\overline{f}}(\overline{\overline{x_1}}, \ldots, \overline{\overline{x_n}}) = f(x_1, \ldots, x_n),$$

što je i trebalo.

Odnos superpozicije i dualnosti. Pođimo od jednog primjera. Neka je

$$f_4(x, y) = f_3(f_1(x, x, y), f_2(x, y))$$

i neka treba odrediti dualnu funkciju f_4^*. Imamo:

$$f_4^*(x, y) = \overline{f_4}(\overline{x}, \overline{y}) = \overline{f_3}(f_1(\overline{x}, \overline{x}, \overline{y}), f_2(\overline{x}, \overline{y})) = \overline{f_3}(\overline{\overline{f_1}}(\overline{x}, \overline{x}, \overline{y}), \overline{\overline{f_2}}(\overline{x}, \overline{y})) =$$

$$\overline{f_3}(\overline{f_1^*}(x, x, y), \overline{f_2^*}(x, y)) = f_3^*(f_1^*(x, x, y), f_2^*(x, y))$$

tj.

$$(f_3(f_1, f_2))^* = f_3^*(f_1^*, f_2^*).$$

U slučaju opšte superpozicije imamo slične okolnosti. Drugim riječima, sredstvima koja su po svemu slična onima upotrebljenim u primjeru može da bude dokazano sljedeće tvrđenje. Princip dualnosti:

$$(f(f_1,\ldots,f_k))^* = f^*(f_1^*,\ldots,f_k^*).$$

21. SDNF BULOVE FUNKCIJE

SDNF znači savršena disjunktivna normalna forma. Vidjećemo da je SDNF jedno kanonsko predstavljanje Bulove funkcije. Uvedimo potrebne oznake i termine. Neka je $n \geq 1$ i razmotrimo skup promjenljivih $\{x_1,\ldots,x_n\}$. Za promjenljivu x iz skupa promjenljivih i za konstantu $\sigma \in \{0,1\}$ definišimo tzv. atom u oznaci x^σ relacijom

$$x^\sigma = \begin{cases} x, \text{ ako je } \sigma = 1, \\ \overline{x}, \text{ ako je } \sigma = 0. \end{cases}$$

Razmotrimo izraz koji nastaje kada se više atoma poveže znakom konjunkcije. Za izraz oblika $K = y_1^{\sigma_1} \,\&\, y_2^{\sigma_2} \,\&\, \ldots \,\&\, y_r^{\sigma_r}$ kažemo da je konjunkcija, gdje su $y_1,\ldots,y_r$ Bulove promjenljive, a $\sigma_1,\ldots,\sigma_r \in \{0,1\}$. Primjer konjunkcije je $K = x_1^0 \cdot x_1^0 \cdot x_1^1 \cdot x_2^1 = \overline{x_1} \cdot \overline{x_1} \cdot x_1 \cdot x_2$. Broj atoma u konjunkciji nazivamo njenim rangom. U primjeru je očito rang $= 4$. Konstantu 1 smatramo konjunkcijom nultog ranga. Za neku konjunkciju kažemo da je elementarna konjunkcija (e. k.) ako je $y_i \neq y_j$ za $i \neq j$. U primjeru ovo očito nije ispunjeno jer je $y_1 = y_2 = y_3$. Dalje, za izraz oblika $\mathcal{D} = \mathcal{D}(x_1,\ldots,x_n) = \vee_{i=1}^s K_i = K_1 \vee K_2 \vee \ldots \vee K_s$ kažemo da je DNF, gdje su $K_1,\ldots,K_s$ konjunkcije. Najzad, za DNF nad razmatranim skupom promjenljivih $\{x_1,\ldots,x_n\}$ kažemo da je SDNF ako je izraz $\mathcal{D}$ sastavljen od e. k. koje su međusobno različite i koja svaka ima maksimalni rang (to znači da je rang $= n$).

Neka je Bulova funkcija od n promjenljivih data svojom tablicom. Da li ta funkcija $f = f(x_1,\ldots,x_n)$ može da bude prikazana u obliku SDNF?

Primjer ($n = 3$). Tablica prikazuje funkciju $f = f(x_1,x_2,x_3)$. Sastaviti odgovarajuću SDNF $\mathcal{D} = \mathcal{D}(x_1,x_2,x_3)$.

x_1	x_2	x_3	f
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Rješenje. Izložimo postupak za konstrukciju SDNF. Iz tablice vidimo da je $f = 1$ na 4 mjesta (4 puta) i zato će odgovor biti $\mathcal{D} = K_1 \vee K_2 \vee K_3 \vee K_4$. Izraz K_1 odgovara slučaju $f(0,0,0) = 1$ i zato $K_1 = \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3}$. Slično se formiraju K_2, K_3 i K_4. Oni odgovaraju za $f(0,0,1) = 1$, $f(1,0,1) = 1$ i $f(1,1,1) = 1$. Postupa se po šablonu: $f(\sigma_1,\sigma_2,\sigma_3) = 1 \Rightarrow K = x_1^{\sigma_1} \cdot x_2^{\sigma_2} \cdot x_3^{\sigma_3}$. Tako da odgovor glasi $\mathcal{D} = \mathcal{D}(x_1,x_2,x_3) = \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \vee \overline{x_1} \cdot \overline{x_2} \cdot x_3 \vee x_1 \cdot \overline{x_2} \cdot x_3 \vee x_1 \cdot x_2 \cdot x_3$. Dokaz da je zaista $f(x_1,x_2,x_3) = \mathcal{D}(x_1,x_2,x_3)$: $K_1 = \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} = 1 \Leftrightarrow \overline{x_1} = 1$ i $\overline{x_2} = 1$ i $\overline{x_3} = 1 \Leftrightarrow$

$x_1 = 0$ i $x_2 = 0$ i $x_3 = 0$, napiši slično i za $K_2 = 1$, $K_3 = 1$ i $K_4 = 1$; plus $K_1 \vee K_2 \vee K_3 \vee K_4 = 1 \Leftrightarrow K_1 = 1$ ili $K_2 = 1$ ili $K_3 = 1$ ili $K_4 = 1$; slijedi da je $f = K_1 \vee K_2 \vee K_3 \vee K_4$. Da bi se dokaz razumio dovoljno je znati elementarna svojstva konjunkcije xy i disjunkcije $x \vee y$. Tek sada imamo pravo da pišemo $f = \overline{x_1}\,\overline{x_2}\,\overline{x_3} \vee \overline{x_1}\,\overline{x_2}x_3 \vee x_1\overline{x_2}x_3 \vee x_1x_2x_3$.

Slično se postupa prilikom sastavljanja SDNF funkcije od ma koliko promjenljivih n koja je data svojom tablicom. Upravo, treba uzeti u obzir one redove tablice u kojima je $f = 1$. Takvom redu pridružuje se e. k. Ta e. k. glasi $x_1x_2 \dots x_n$, gdje još x_1 treba da bude nadvučeno ako je u tom redu tablice $x_1 = 0$, a slično naravno i za $x_2, \dots, x_n$. Onda još sve e. k. povežemo pomoću znakova $\vee$, nastala je tražena SDNF.

Primjer ($n = 2$). Naći SDNF funkcije $f = f(x_1, x_2)$ definisane tablicom.

x_1	x_2	f
0	0	1
0	1	1
1	0	0
1	1	1

Odgovor je: $f(x_1, x_2) = \overline{x_1}\,\overline{x_2} \vee \overline{x_1}x_2 \vee x_1x_2$.

Možemo reći da se na račun $\sigma = (\sigma_1, \sigma_2) \in \{0,1\} \times \{0,1\}$ takvog da je $f(\sigma_1, \sigma_2) = 1$ pojavljuje e. k. $x_1^{\sigma_1}x_2^{\sigma_2}$. Recimo, $f(0,1) = 1$ pa se pojavljuje $\overline{x_1}x_2$, srednji sabirak. Iz dokaza: $\overline{x_1}x_2 = 1 \Leftrightarrow (x_1, x_2) = (0,1)$.

Smatramo da dosadašnji primjeri mogu da posluže kao dokaz sljedeće teoreme.

Teorema. Svaka Bulova funkcija $f = f(x_1, \dots, x_n)$ može da bude prikazana u obliku SDNF, tj. za svaku funkciju $f \in P_2$ važi relacija

$$f(x_1, \dots, x_n) = \bigvee_{\sigma:\, f(\sigma)=1} x_1^{\sigma_1} \,\&\, x_2^{\sigma_2} \,\&\, \dots \,\&\, x_n^{\sigma_n}.$$

Napisano je da se disjunkcija uzima po svim mogućim uređenim n–torkama $\sigma = (\sigma_1, \dots, \sigma_n) \in \{0,1\}^n$ koje zadovoljavaju $f(\sigma_1, \dots, \sigma_n) = 1$.

Dalje, lako se može pokazati da je predstavljanje Bulove funkcije u obliku SDNF – jedinstveno, do na redosljed faktora unutar jednog sabirka i na redosljed sabiraka.

Na kraju, dualno se definiše SKNF (... konjunktivna ...).

22. PRIMJERI KOMPLETNIH SISTEMA

Za sistem Bulovih funkcija $A \subset P_2$ kaže se da je kompletan ako bilo koja Bulova funkcija može da se izrazi kao superpozicija u kojoj učestvuju samo članovi tog sistema. Ako je sistem $A = \{f_1, \ldots, f_k\}$ kompletan onda su logički elementi $f_1, \ldots, f_k$ dovoljni da se sastavi bilo koja logička mreža. U nastavku će biti navedeno sedam primjera kompletnih sistema.

1. Sistem $A = P_2$ je očito kompletan.

2. Sistem $A = \{\,\&\,, \vee, \neg\}$ je kompletan, kao što je pokazano u prethodnom naslovu (SDNF – savršena disjunktivna normalna forma). Jedino treba dodati da se $f(x) = 0$ izražava kao $f(x) = x\,\&\,\overline{x}$.

3. Sistem $A = \{\,\&\,, \neg\}$ je kompletan. Zaista, $\vee$ može da se izrazi preko & i $\neg$; znamo da je $x \vee y = \overline{\overline{x}\,\&\,\overline{y}}$. Tako da smo sveli na prethodni slučaj. Drugim riječima, I, ILI i NE čine kompletan sistem i ILI može da se izrazi preko I i NE $\Rightarrow$ I i NE čine kompletan sistem.

4. Sistem $A = \{\vee, \neg\}$ je kompletan. Takođe se dokazuje svođenjem na slučaj SDNF. Naime, poznata je jednakost $x\,\&\,y = \overline{\overline{x} \vee \overline{y}}$.

5. Sistem $A = \{\uparrow\}$ je kompletan, $\uparrow$ je tzv. Šeferova funkcija, $\uparrow = 1110$. Dovoljno je vidjeti da & i $\neg$ mogu da se izraze preko $\uparrow$. Uvjeriti se da su sljedeće dvije jednakosti istinite: $\overline{x} = x \uparrow x$ i $x\,\&\,y = (x \uparrow y) \uparrow (x \uparrow y)$.

6. Sistem $A = \{0, 1, x\,\&\,y, x \oplus y\}$ je kompletan. Bilo koji sistem koji može da proizvede konjunkciju i negaciju je kompletan, u vezi primjera 3. Zaista, $f \in P_2$ prvo se izrazi preko konjunkcije i negacije, a onda se konjunkcija i negacija izraze preko članova tog sistema. Dati sistem A već sadrži konjunkciju. Negacija može da bude prikazana kao $\overline{x} = 1 \oplus x$.

7. Polinom po modulu 2 ili polinom Žegalkina je izraz oblika $a \oplus b \oplus \ldots \oplus l$ gdje su učesnici $a, b, \ldots, l$ jednaki 0 ili 1 ili x_i ili $x_i x_j$ ili itd. Primjeri polinoma po modulu 2 su $x_1 \oplus x_3 \oplus x_2 x_3 \oplus x_1 x_2 x_3$ i $1 \oplus x_1 x_2 \oplus x_1 x_3$. Najprostiji mogući primjeri polinoma po modulu 2 su Bulove funkcije 0, 1, x i $1 \oplus x$.

Skup svih polinoma po modulu 2 je kompletan sistem. Zaista, taj skup je nadskup kompletnog sistema $\{0, 1, x\,\&\,y, x \oplus y\}$ iz prethodnog primjera 6.

Kako ispitati da li je dati sistem kompletan? Sljedeća teorema daje opšti odgovor.

Postov kriterijum kompletnosti. Skup $A \subset P_2$ je kompletan ako i samo ako je ispunjeno sljedećih pet relacija: $A \setminus S \neq \emptyset$, $A \setminus T_0 \neq \emptyset$, $A \setminus T_1 \neq \emptyset$, $A \setminus L \neq \emptyset$ i $A \setminus M \neq \emptyset$.

Samo ćemo ukratko objasniti oznake. Kaže se da je S klasa samodualnih funkcija, T_0 klasa funkcija koje čuvaju nulu, T_1 klasa funkcija koje čuvaju jedinicu, L klasa linearnih funkcija i M klasa monotonih funkcija.

$f \in S$ ako i samo ako $f^* = f$

$f = f(x_1, \ldots, x_n) \in T_0$ ako i samo ako $f(0, \ldots, 0) = 0$

$f = f(x_1, \ldots, x_n) \in T_1$ ako i samo ako $f(1, \ldots, 1) = 1$

$f = f(x_1, \ldots, x_n) \in L$ ako se u prikazivanju funkcije f u obliku polinoma po modulu 2 kao učesnici pojavljuju jedino (mogu da se pojave) $0, 1, x_1, \ldots, x_n$

$f = f(x_1, \ldots, x_n) \in M$ ako važi $\alpha_1 \leq \beta_1, \ldots, \alpha_n \leq \beta_n \;\Rightarrow\; f(\alpha_1, \ldots, \alpha_n) \leq f(\beta_1, \ldots, \beta_n)$

Na kraju oblasti o Bulovim funkcijama navedimo neke relacije koje nisu uspjele da ranije nađu sebi svoje pravo mjesto: $x \uparrow y = \overline{x\,\&\,y}$, $x \oplus x = 0$, $x \oplus \overline{x} = 1$, $1 \oplus 1 \oplus \varphi = \varphi$ i $ab \oplus ac = a(b \oplus c)$.

23. SABIRAČ

Kao prvi primjer logičke mreže ili kombinacione mreže navodimo sabirač. Logičke mreže su tipični primjeri hardverskih komponenti. Jedna logička mreža održava jednu ili više logičkih (Bulovih) funkcija. Logičke mreže sastavljene su od logičkih elemenata koji odražavaju pojedine elementarne Bulove funkcije, kao što su recimo I, ILI, NE i isključivo–ILI.

Za jednu određenu logičku mrežu imamo sljedeći šablon ili redosljed razmatranja:

1) definicija, 2) tablica, 3) izraz, 4) sinteza i 5) simbol.

Definicijom mreže određuju se njene ulazne promjenljive i njena izlazna ili izlazne promjenljive a takođe se definiše i zavisnost izlaznih promjenljivih od ulaznih. Na osnovu definisane zavisnosti neposredno se sastavi tablica za odgovarajuće Bulove funkcije (za izlazne promjenljive). Zatim se na osnovu tablice sastavi funkcionalni izraz (izrazi) u kojima se pojavljuju elementarne funkcije recimo I, ILI, NE i isključivo–ILI. Uopšte uzev, mogući su razni izrazi koji odražavaju jednu te istu zavisnost izlazne veličine od vrijednosti ulaznih veličina. Nastoji se da izraz bude što je moguće jednostavniji, jer će tada i rezultujuća mreža biti jednostavnija. Znamo da zadatak sastaviti izraz na osnovu date tablice sigurno može da bude riješen pomoću SDNF Bulove funkcije. Sljedeće, da bi se ostvarila sinteza logičke mreže, treba da se doslovno slijedi sastavljeni izraz (ako mreža ima jedan izlaz) odnosno sastavljeni izrazi (kada mreža ima više izlaza). Povezati odgovarajuće logičke elemente onako kako to izraz predviđa. Sada je mreža sastavljena, čime je i završeno rješavanje postavljenog zadatka. Na kraju, treba da bude uveden i grafički znak (simbol) koji odgovara mreži koja je maločas sastavljena. Zato što će kasnije ova mreža da predstavlja sastavni dio neke složenije mreže.

Pogledajmo mrežu koja omogućava da se izvrši sabiranje dva broja prikazana u binarnom brojnom sistemu. Mi ćemo ustvari razmotriti tri mreže: a) polusabirač (semi–adder), b) potpuni sabirač (full adder) i c) paralelni sabirač (parallel adder).

Što se tiče polusabirača, postoje dvije ulazne promjenljive x i y i dvije izlazne promjenljive z i q a njihov smisao je redom: x – prvi sabirak, y – drugi sabirak, z – cifra zbira i q – prenos koji se obrazuje. Oblast definisanosti svake promjenljive je skup $\{0,1\}$. Već je rečeno da z i q zavise od x i y tako da se može pisati $z = z(x,y)$ i $q = q(x,y)$. Da bi se razumjelo, ako bi se umjesto binarnog brojnog sistema koristio dekadni onda bi ulaznim vrijednostima $x = 7$ i $y = 8$ odgovarale izlazne vrijednosti $z = 5$ i $q = 1$. U tabeli 1 prikazana je tablica koja odgovara mreži polusabirača. Lako je sastaviti dva odgovarajuća izraza $z = z(x,y)$ i $q = q(x,y)$. Upravo, važe sljedeće dvije jednakosti: $z = x \oplus y$, $q = x \cdot y$. Mreža polusabirača prikazana je na slici 1, tako da je i sinteza razmatrane logičke mreže urađena.

Što se tiče potpunog sabirača, takođe se razmatra zadatak o sabiranju na jednoj koordinati, samo što će sada biti uzeta u obzir i treća ulazna veličina – prenos koji je došao sa prethodne koordinate. Slično kao što u dekadnom prenos koji dolazi recimo sa mjesta stotina utiče na cifru zbira na mjestu hiljada. Za potpuni sabirač, ulazne promjenljive su x, y i p – cifra prvog sabirka, cifra drugog sabirka i stari prenos a izlazne promjenljive su z i q – cifra zbira i novi prenos. Možda bi bilo pogodnije da se koriste druge oznake, tj. da se umjesto x, y, p, z i q piše redom x_i, y_i, p_{i-1}, z_i i p_i. U tabeli 2 prikazana je tablica za potpuni sabirač. Uvjeriti se da mogu da posluže sljedeći izrazi: $z = x \oplus y \oplus p$, $q = (x \oplus y) \cdot p \vee x \cdot y$. Pridržavajući se napisanih izraza, lako može da bude sastavljena šema potpunog sabirača, v. sliku 2. Samo se napominje da bi mogla da posluže i sljedeća tri izraza u kojima se pojavljuje i pomoćna veličina f: $f = \overline{x}y \vee x\overline{y}$, $z = \overline{f}p \vee f\overline{p}$, $q = fp \vee xy$; naravno da bi šema potpunog sabirača izgledala drukčije ako bismo se poslužili posljednjim jednačinama. Na slici 3 prikazan je grafički znak.

Što se tiče paralelnog sabirača, treba da bude konstruisana mreža koja ostvaruje operaciju sabiranja dva broja. Uzima se da su sabirci $x_4x_3x_2x_1$ i $y_4y_3y_2y_1$ prikazani u binarnom brojnom sistemu i da imaju po četiri bita. Zbir tj. rezultat označava se kao $z_4z_3z_2z_1$, tako da važi jednakost $(x_4x_3x_2x_1)_2 + (y_4y_3y_2y_1)_2 = (z_4z_3z_2z_1)_2$. Na primjer: $(0100)_2 + (0101)_2 = (1001)_2$. Da bi se riješio postavljeni zadatak, dovoljno je da se povežu četiri potpuna sabirača Σ, onako kako to prikazuje slika 4. Umjesto krajnjeg desnog Σ može da se stavi jedan polusabirač.

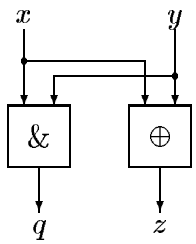
Vidimo da se signal za prenos p_i kaskadno prenosi od bita najmanje težine prema bitu najveće težine. Kaže se da se prenos obavlja na redni (serijski) način.

Pogledajmo i vremenske karakteristike mreže na slici. Označimo sa t vrijeme propagacije signala kroz jedan stepen paralelnog sabirača (kroz jednu komponentu Σ). Drugim riječima, za svako Σ, tek kada prođe t mili–sekundi od trenutka kada su ulazni signali postavljeni, tek tada će izlazni signali zauzeti pravilne vrijednosti. Tako da je ukupno vrijeme kašnjenja razmatrane mreže jednako $4t$ očito.

Na kraju, ako je $p_4 = 1$ onda se time saopštava da je došlo do prekoračenja, tj. da zbir ima više od četiri cifre.

x	y	q	z
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

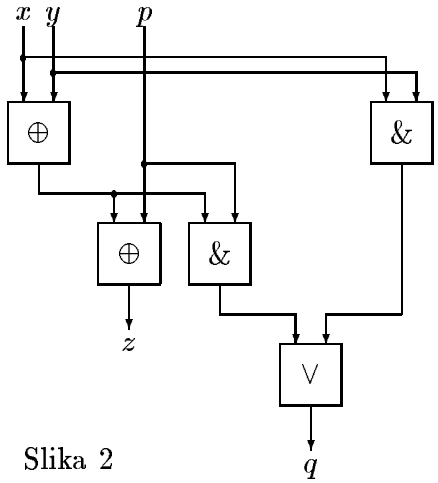
Tabela 1



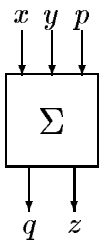
Slika 1

x	y	p	q	z
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

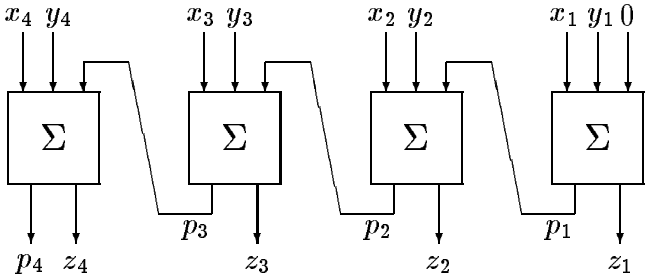
Tabela 2



Slika 2



Slika 3



Slika 4

24. DEKODER

Kao drugi primjer logičke (kombinacione) mreže razmotrimo dekoder. Na početku se daje definicija mreže: nabrojati ulazne i izlazne veličine i definisati kakva je funkcionalna zavisnost izlaza od ulaza. Neka je $n \geq 1$. Dekoder ima n ulaznih veličina u oznaci $x_1, \ldots, x_n$ i ima $N = 2^n$ izlaznih veličina u oznaci $y_1, y_2, \ldots, y_N$, gdje su sve nabrojane veličine bulovske, što znači da uzimaju vrijednosti iz skupa $\{0, 1\}$. Što se tiče zavisnosti $y_j = y_j(x_1, \ldots, x_n)$, za svako $j \in \{1, 2, \ldots, N\}$ osim za jedno j treba da bude $y_j = 0$, a znači samo za jednu odabranu vrijednost j treba da bude $y_j = 1$. Kombinacija vrijednosti ulaznih veličina $x_1, \ldots, x_n$ treba da determiniše vrijednost indeksa j za koju je $y_j = 1$. Za različite kombinacije, različit indeks j treba da bude odabran.

Navedimo jedan običan primjer za upotrebu dekodera. Obrada koju računar vrši definisana je programom na mašinskom jeziku. Jedna po jedna naredba dolazi na red za izvršavanje. Aktuelna naredba donosi se iz memorije u procesor. Procesor treba da razumije smisao naredbe koja je pristigla (da dekodira naredbu). Onda će procesor da generiše jedan upravljački signal, zavisno od vrste naredbe. Upravljački signal je upućen i zato počinje da radi odabrana hardverska komponenta. Aktivirana komponenta će izvršiti predviđenu obradu: sabraće dva broja ili će upisati nulu u određeni bajt u memoriji ili će broj iz akumulatora poslati na izlazni port ili slično. Dakle, naredba koja je pristigla prvo se propušta kroz dekoder da bi bio generisan tačan upravljački signal. Govoreći preciznije, kroz dekoder se propuštaju samo oni bitovi naredbe koji definišu vrstu naredbe, vrstu radnje (+ ili 0 ili izlaz ili slično).

Umjesto dekoder sa n ulaza ponekad se kaže dekoder oblika n/N ili dekoder 1 od N. Najviše se koriste dekoderi kod kojih je $n = 2$, $n = 3$ ili $n = 4$. Riješićemo postavljeni zadatak (sastaviti jednačinu i šemu dekodera) u slučajevima $n = 2$ i $n = 3$. Prilikom rješavanja polazimo od tablice koja je naravno sastavljena saglasno prethodnoj definiciji; v. jednu i drugu tablicu.

U slučaju $n = 2$ jednačine dekodera glase

$$y_1 = x_1 \cdot x_2, \quad y_2 = x_1 \cdot \overline{x_2}, \quad y_3 = \overline{x_1} \cdot x_2, \quad y_4 = \overline{x_1} \cdot \overline{x_2}.$$

Jednačine smo lako sastavili zato što su nam dobro poznata svojstva tri glavne Bulove funkcije &, $\vee$ i $\overline{x}$. Sada nemamo nikakvih teškoća da sastavimo šemu dekodera, v. sliku. Na redu je odgovarajući grafički simbol, v. sliku: grafički simbol dekodera 1 od 4.

Razmotrimo slučaj kada je $n = 3$ tj. razmotrimo dekoder oblika 3/8. Ulaze označimo kao A_0, A_1 i A_2 a izlaze kao $Y_0, Y_1, \ldots, Y_7$. Jednačine mogu da budu zapisane kao

$$Y_0 = \overline{A_2} \cdot \overline{A_1} \cdot \overline{A_0}, \quad Y_1 = \overline{A_2} \cdot \overline{A_1} \cdot A_0, \quad Y_2 = \overline{A_2} \cdot A_1 \cdot \overline{A_0}, \quad Y_3 = \overline{A_2} \cdot A_1 \cdot A_0,$$

$$Y_4 = A_2 \cdot \overline{A_1} \cdot \overline{A_0}, \quad Y_5 = A_2 \cdot \overline{A_1} \cdot A_0, \quad Y_6 = A_2 \cdot A_1 \cdot \overline{A_0}, \quad Y_7 = A_2 \cdot A_1 \cdot A_0.$$

U nastavku, definicija dekodera se donekle modifikuje (dograđuje). Izlazi dekodera mogu da budu uslovljeni postojanjem sinhronizacionog signala E. Za E se drukčije kaže da je signal dozvole, engl. enable. Ovakav dekoder funkcioniše na sljedeći način. Ako je $E = 1$ (signal E je aktivan, signal E je na visokom nivou) onda zavisnost izlaza od ulaza ostaje po starom. Međutim, ako je $E = 0$ (dozvola nije data) onda svi izlazi treba da budu jednaki nuli, tj. treba da bude $Y_0 = Y_1 = \ldots = Y_7 = 0$, bez obzira na kombinaciju vrijednosti ulaznih promjenljivih A_0, A_1 i A_2. Sa ovim uslovljavanjem, ranije jednačine se nešto modifikuju i one postaju

$$Y_0 = \overline{A_2} \cdot \overline{A_1} \cdot \overline{A_0} \cdot E, \quad Y_1 = \overline{A_2} \cdot \overline{A_1} \cdot A_0 \cdot E, \quad \ldots, \quad Y_7 = A_2 \cdot A_1 \cdot A_0 \cdot E.$$

Za signal E kaže se da je kontrolni signal. Signal E treba držati na niskom nivou $E = 0$ za vrijeme dok ulazne promjenljive mijenjaju vrijednosti a treba ga postaviti na visoki nivo $E = 1$ kada su ulazne promjenljive postale stabilne, tj. kada su ulazne promjenljive zauzele predviđene vrijednosti.

Dalje, izdavanje signala dozvole može da zavisi od nekoliko okolnosti (događaja). Na primjer: dati dozvolu jedino ako su prvi i drugi događaj nastupili a treći događaj nije nastupio. Dakle, neka se signal E sa svoje strane generiše kombinacijom drugih kontrolnih signala, za koje se kaže da su signali tipa CS, što je skraćenica od engl. chip select, a što znači selekcija čipa. Primjera radi, uzmimo konkretno da su prisutna tri takva signala u oznaci $CS1$, $CS2$ i $CS3$ i uzmimo da su oni vezani tako da djeluju po sljedećem propisu:

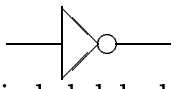
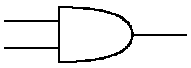
$$E = CS1 \cdot CS2 \cdot \overline{CS3}.$$

Na slici a) prikazana je šema dekodera oblika 3/8 sa signalom dozvole E za koji važi maločas napisana formula.

U šemama digitalnih kola, dekoder se prikazuje kao pravougaonik, gdje su još naznačeni njegovi ulazni, izlazni i kontrolni signali, a još je unutar pravougaonika naznačena i vrsta dekodera. Tako je na slici b) prikazan simbol koji odgovara dekoderu sa slike a).

Na kraju, enkoder je mreža koja je inverzna mreži dekodera. Dakle, ulazne promjenljive enkodera su $y_1, y_2, \ldots, y_N$ a njegove izlazne promjenljive su $x_1, \ldots, x_n$, pri čemu ostaju da važe one iste zavisnosti između promjenljivih koje su bile ranije specifikovane na početku ovog naslova.

Tabela: Tablica za dekoder sa dva ulaza
Tabela: Dio tablice za dekoder sa tri ulaza
Slika: Šema dekodera 1 od 4
Slika: Grafički simbol dekodera 1 od 4
Slika a): Šema dekodera oblika 3/8 sa signalom dozvole $E = CS1 \cdot CS2 \cdot \overline{CS3}$

uz ovu sliku:  = invertor,  = i–kolo

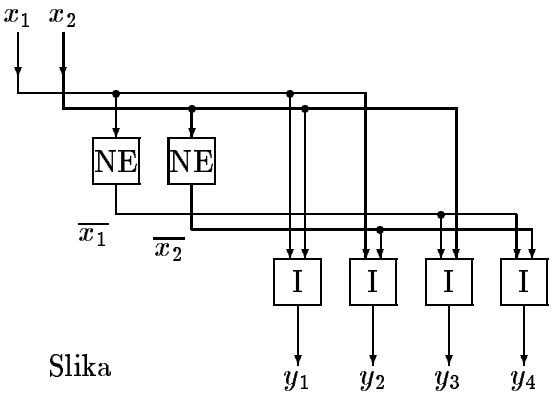
Slika b): Grafički simbol dekodera sa slike a)

x_1	x_2	y_1	y_2	y_3	y_4
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

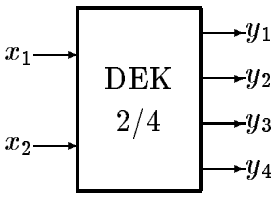
Tabela

A_2	A_1	A_0	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
...			.....							
1	1	1	0	0	0	0	0	0	0	1

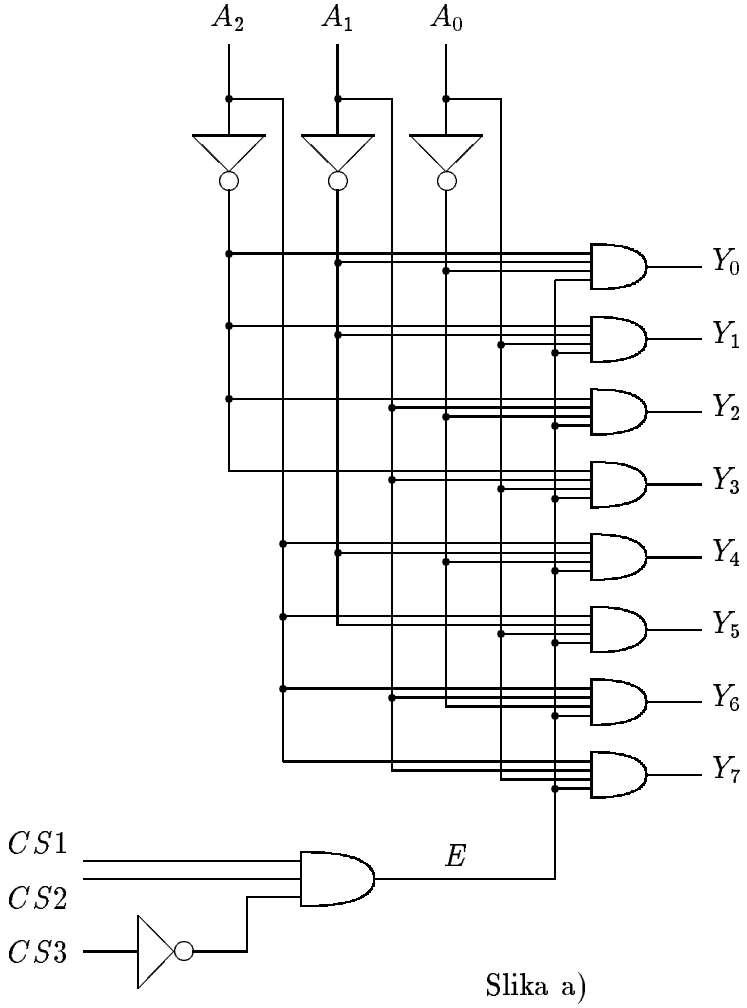
Tabela



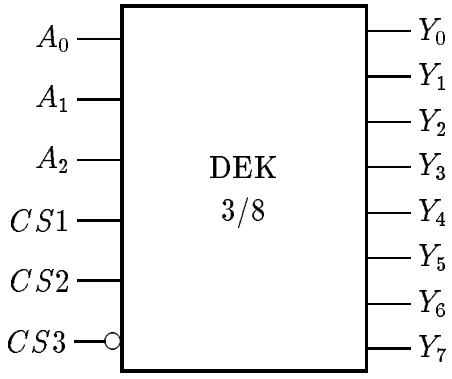
Slika



Slika



Slika a)



Slika b)

25. MULTIPLEKSER

Slika 1 Funkcionalna šema osmoulaznog multipleksera. Signal SEL sastoji se od tri binarna signala S_0, S_1 i S_2

Slika 2 Logička šema osmoulaznog multipleksera. Ova šema vjerno odražava formulu $(*)$. Ulazi: D_0, D_1, D_2, D_3, D_4, D_5, D_6, D_7, S_0, S_1, S_2 i E, izlaz: Y. Koriste se logički elementi NE, I i ILI

Slika 3 Simbol osmoulaznog multipleksera

Slika 4 Simbol četvoroulaznog multipleksera

Slika 5 Simbol dvoulaznog multipleksera. Ili: simbol multipleksera za dvije informacije. Ili: simbol multipleksera za dvije informacije čije su veličine po jedan bit

Slika 6 Multiplekser za dvije informacije veličine po četiri bita

$$Y = (D_0\,\overline{S_2}\,\overline{S_1}\,\overline{S_0} \vee D_1\,\overline{S_2}\,\overline{S_1}\,S_0 \vee D_2\,\overline{S_2}\,S_1\,\overline{S_0} \vee D_3\,\overline{S_2}\,S_1\,S_0 \vee$$

$$D_4\,S_2\,\overline{S_1}\,\overline{S_0} \vee D_5\,S_2\,\overline{S_1}\,S_0 \vee D_6\,S_2\,S_1\,\overline{S_0} \vee D_7\,S_2\,S_1\,S_0) \cdot E \qquad (*)$$

Uz sliku 1. Ako je $E = 0$ (prekidač je otvoren) onda će biti $Y = 0$. Ako je $E = 1$ (prekidač je zatvoren) onda će nastupiti jedan od sljedećih mogućih slučajeva: $Y = D_0$ ili $Y = D_1$ ili ... ili $Y = D_7$. Veličina SEL odlučuje koji od osam mogućih slučajeva nastupa. Napominje se da promjenljiva SEL nije binarna promjenljiva.

Uz sliku 3. Ako je $E = 0$ onda $Y \leftarrow 0$. Ako je $E = 1$ onda $Y \leftarrow D_0$ ili $Y \leftarrow D_1$ ili ... ili $Y \leftarrow D_7$, u zavisnosti od S_0, S_1 i S_2. Recimo, ako je $(S_2, S_1, S_0) = (0, 0, 0)$ onda $Y \leftarrow D_0$. Recimo, ako je $(S_2, S_1, S_0) = (0, 0, 1)$ onda $Y \leftarrow D_1$. I slično.

Uz sliku 4. Ako je $E = 0$ onda $Y \leftarrow 0$. Ako je $E = 1$ onda $Y \leftarrow D_i$ za neko $i \in \{0, 1, 2, 3\}$. Postoje četiri mogućnosti za i i postoje četiri mogućnosti za vrijednost para (S_0, S_1). Ima sedam ulaznih promjenljivih D_0, D_1, D_2, D_3, S_0, S_1 i E i ima jedna izlazna promjenljiva Y. Sve promjenljive su binarne, tj. mogu biti 0 ili 1.

Uz sliku 5. Ako je $E = 0$ onda $Y \leftarrow 0$. Ako je $E = 1$ i $S = 0$ onda $Y \leftarrow D_0$. Ako je $E = 1$ i $S = 1$ onda $Y \leftarrow D_1$. E – enable – dozvola. S – selekcija. Dvije ulazne linije D_0 i D_1 konkurišu da budu propuštene na izlaz.

Uz sliku 6. Dva snopa $(1D_0, 2D_0, 3D_0, 4D_0)$ i $(1D_1, 2D_1, 3D_1, 4D_1)$ konkurišu. Jedan će biti propušten na izlaz, S odlučuje. Izlazni snop $(1Y, 2Y, 3Y, 4Y)$ dobija vrijednost jednog od ulaznih snopova, ako $E = 1$.

Multiplekser je kombinaciona mreža koja obavlja funkciju digitalnog višepoložajnog prekidača. Signal selekcije SEL sa slike 1 bira jednog od osam ulaznih signala. Izabrani ulazni signal prenosi se na izlaz Y. Drugim riječima, željeni ulazni signal biva priključen na izlazni priključak Y; ukoliko je aktiviran prekidač dozvole E. Ako je E neaktivan onda će izlaz biti logička nula, bez obzira na položaj prekidača SEL i vrijednosti ulaznih promjenljivih D_0 do D_7. Na osnovu ove definicije multipleksera, sada treba sastaviti njegovu logičku funkciju, tj. izraz za Y. Selekcija ulaza kodira se binarno signalima S_0, S_1 i S_2. Analizirajmo ukratko izraz za Y iz gotovog rješenja $(*)$. Ako je $(S_2, S_1, S_0) = (0, 0, 1)$ onda će biti $Y = D_1$, i slično; svaki put: pod uslovom da je $E = 1$. Na osnovu $(*)$ lako se izvrši sinteza multipleksera, v. sliku 2. Na slici 3 prikazan je simbol digitalnog multipleksera sa osam ulaza.

Kao integrisane komponente, multiplekseri se izrađuju sa dva, četiri, osam ili šesnaest ulaza. U slučaju četiri ulaza (četiri ulaza podataka) D_0 do D_3 imamo naravno dva selekciona ulaza S_0 i S_1, a izlazni signal Y dat je jednačinom:

$$Y = (D_0\,\overline{S_1}\,\overline{S_0} \vee D_1\,\overline{S_1}\,S_0 \vee D_2\,S_1\,\overline{S_0} \vee D_3\,S_1\,S_0)\cdot E,$$

v. sliku 4. Slično, u slučaju dva ulaza:

$$Y = (D_0\,\overline{S} \vee D_1\,S)\cdot E,$$

v. sliku 5. Dva ulaza: $Y = D_0\,\overline{S} \vee D_1\,S$, $S = 0 \Rightarrow Y = D_0$, $S = 1 \Rightarrow Y = D_1$.

Multiplekseri se (radi univerzalnosti) obično izrađuju da sadrže i direktni izlaz Y i komplementirani izlaz $\overline{Y}$. Signal dozvole izlaza E istovremeno aktivira ili deaktivira oba izlaza.

U digitalnim sistemima, gdje se koriste višebitne informacije, po jedan multiplekser se koristi za svaki bit, dok su selekcioni ulazi i signal dozvole izlaza zajednički za sve multipleksere. Neka se vrši selekcija između n informacija D_0 do D_{n-1} i neka je veličina jedne informacije k bita. Npr. može biti $n = 2$ i $k = 4$. Ili $n = 2$ i $k = 8$. S druge strane, jasno je da n određuje količinu binarnih komponenti S_0 do S_{m-1} signala selekcije SEL. Upravo, ta količina mora biti jednaka $m = \log_2 n$. Vrlo često korišćena komponenta u digitalnim sistemima jeste dvoulazni multiplekser. Znači da će se birati između dvije mogućnosti. A kolika je širina? Dvoulazni multiplekser često se izrađuje kao četvorobitni, v. sliku 6.

U literaturi i u katalozima integrisanih komponenti, multiplekser se često naziva selektorom podataka, engl. Data Selector. Selekcioni ulazi nazivaju se adresnim ulazima i označavaju se sa A, B, C, itd. umjesto sa S_0, S_1, S_2, itd.

Multiplekser može da koristi i za realizaciju (za sintezu) logičkih funkcija. Pogledajmo na primjeru funkcije

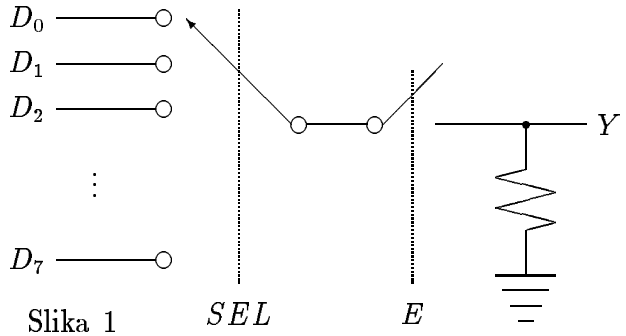
$$F = \overline{C}\,\overline{B}\,A \vee C\,\overline{B} \vee C\,B\,\overline{A} =$$

$$\overline{C}\,\overline{B}\,A \vee C\,\overline{B}\,\overline{A} \vee C\,\overline{B}\,A \vee C\,B\,\overline{A}.$$

V. sliku 7 i sliku 8. Prema slici 8, multiplekser sa dva selekciona ulaza poslužio je za realizaciju funkcije od tri promjenljive. Uopšte, funkcija od $m+1$ promjenljivih može da bude realizovana pomoću multipleksera sa $n = 2^m$ ulaza (ulaza za podatke).

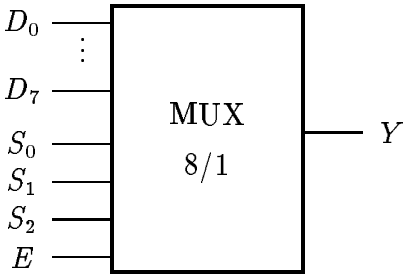
Dokažimo posljednje tvrđenje u slučaju funkcije od tri promjenljive $F = F(A, B, C)$. Na ulaze S_0 i S_1 dovode se redom B i C. Na ulaze D_0, D_1, D_2 i D_3 dovode se kako gdje 0, 1, $\overline{A}$ i A, u zavisnosti od izgleda SDNF funkcije F. Na primjer, za D_0 se gleda da li su u SDNF prisutni $\overline{A}\,\overline{B}\,\overline{C}$ i $A\,\overline{B}\,\overline{C}$. Ako su oba prisutna onda na D_0 dovedi 1. Ako su oba odsutna onda dovedi 0. Ako je samo jedan prisutan onda dovedi $\overline{A}$ ili A. Da bi upravo izložena analiza bila razumljiva, treba imati u vidu već ranije navedenu jednačinu multipleksera 4/1:

$$F = D_0\,\overline{S_1}\,\overline{S_0} \vee D_1\,\overline{S_1}\,S_0 \vee D_2\,S_1\,\overline{S_0} \vee D_3\,S_1\,S_0.$$

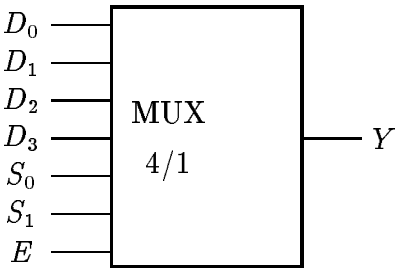


Slika 1

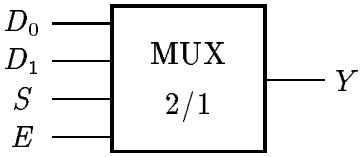
Slika 2



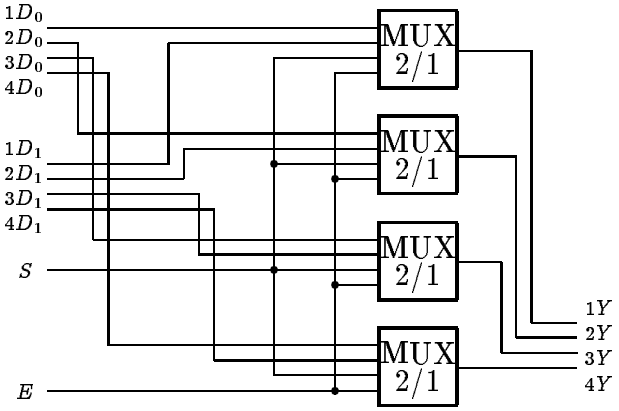
Slika 3



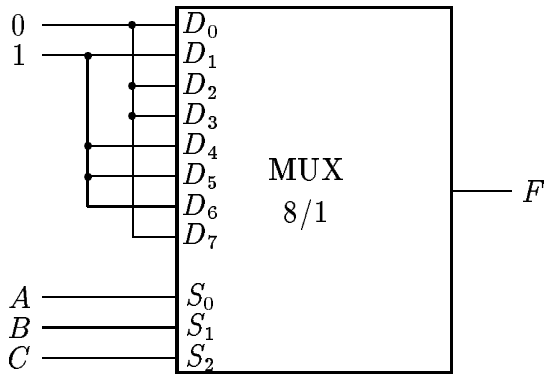
Slika 4



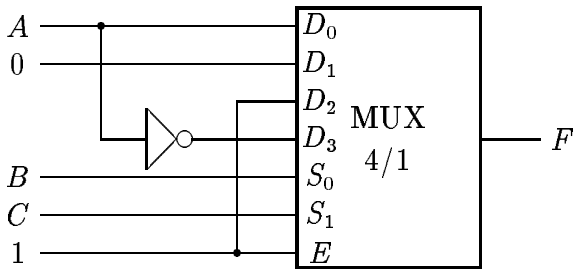
Slika 5



Slika 6

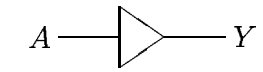


Slika 7 $F = F(A, B, C)$

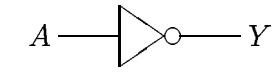


Slika 8 $F = F(A, B, C)$

26. TROSTATIČKI BAFERI

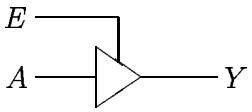
a) 

$Y = A$ (bafer)

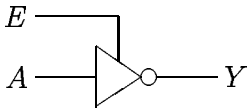
b) 

$Y = \overline{A}$ (invertor)

Logička kola sa slika a) i b) su poznata odranije. Za kolo sa slike a) kaže se da je neinvertujući pojačivač ili neinvertujući bafer ili bafer. Za kolo sa slike b) kaže se da je invertujući bafer ili invertor. Znamo da postoje razne tehnologije izrade logičkih kola, a primjeri tehnologija su TTL i CMOS. Jedno logičko kolo odražava jednu logičku funkciju (Bulovu funkciju). Osim takvih kola, u mnogim tehnologijama izrađuju se i modifikovana kola. Za modifikovano kolo kaže se da je trostatičko kolo (trostatičko logičko kolo). Izlaz trostatičkog kola nalazi se u jednom od tri moguća stanja (položaja). Za vrijednost izlaznog signala trostatičkog kola postoje tri mogućnosti. Upravo, izlazni signal može da bude logička nula ili logička jedinica ili izlaz može da bude u stanju visoke impedanse (visokog otpora). Trostatička kola mogu da budu logička I, ILI, NI, NILI, invertori i neinvertujući pojačivači. Ukoliko trostatički invertor odnosno neinvertujući pojačivač ima povećani izlazni faktor grananja u odnosu na standardna kola date familije onda se takvo kolo naziva trostatičkim baferom ili trostatičkim drajverom.

c) 

$\begin{cases} \text{ako je } E = 1 \text{ onda je } Y = A \\ \text{ako je } E = 0 \text{ onda je } Y \text{ u stanju visoke impedanse} \end{cases}$

d) 

$\begin{cases} \text{ako je } E = 1 \text{ onda je } Y = \overline{A} \\ \text{ako je } E = 0 \text{ onda je } Y \text{ u stanju visoke impedanse} \end{cases}$

Na slikama c) i d) prikazani su standardni simboli za trostatičke bafere. Signal koji prebacuje kolo u stanje visoke impedanse naziva se signalom dozvole ili signalom aktiviranja. Najčešće se obilježava slovom E, po engleskoj riječi enable = dozvola. Na slici c) je neinvertujući trostatički bafer, a na slici d) je invertujući trostatički bafer. I jedan i drugi bafer aktivni su kada je signal dozvole E jednak jedinici. Dok su za $E = 0$ oni u stanju visoke impedanse. Bafer c) za $E = 1$ predstavlja neinvertujući pojačivač. Bafer d) za $E = 1$ predstavlja invertor.

Trebalo bi nacrtati šemu sa tranzistorima za c) i d). Sa šeme se vidi da izlazu Y prethode gornji tranzistor T_1 (kome prethodi izvor napajanja) i donji tranzistor T_2 (kome prethodi uzemljenje). Neka bude $E = 0$. Tada i T_1 i T_2 postaju neprovodni ili zakočeni. Struja kroz jedan i drugi tranzistor je izuzetno mala; recimo, ona ima red veličine $10^{-15}A$. Izlaz Y je razdvojen i od izvora za napajanje i od zemlje (od mase). Izlazna impedansa je reda veličine recimo $10^9\Omega$. Zato se kaže da je tada izlaz Y u stanju visoke impedanse ili u trećem stanju ili u neopterećenom stanju. Izlaz Y svakako se dovodi na ulaz nekog drugog kola (nekog narednog kola). Ako je $E = 0$ onda Y ne opterećuje naredno kolo. Kao da je linija Y odspojena (ili otkačena) od narednog kola. Matematičar bi kazao da je vrijednost Y nedefinisana, kada je $E = 0$.

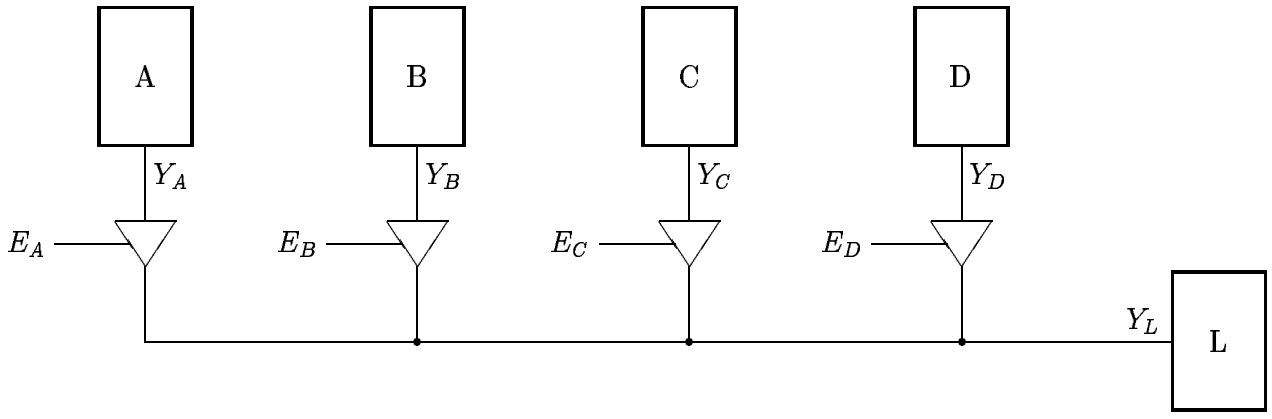
Prosto rečeno, vrijednost napona na izlaznoj liniji Y praktično je jednaka nula volti kad

dozvola nije data. Znamo da nije obavezno da se u svakoj familiji logička nula poklapa sa nula volti, odnosno sa fizičkom nulom. Ovo je zaključak o kolima sa slika c) i d). Ponovimo još jednom, u slučaju kola sa slike c). Ako je E = logički 1 i A = logički 0 onda je Y = logički 0. Ako je E = logički 1 i A = logički 1 onda je Y = logički 1. Ako je E = logički 0 onda je (bez obzira na A) Y = fizički 0, tj. $Y = 0$ volti.

Neka se izlazi iz više digitalnih uređaja šalju posredstvom zajedničke linije prema jednom digitalnom uređaju (recimo prema digitalnom uređaju L), naravno ne više od jednog izlaza u isto vrijeme. Ovakvo slanje ostvaruje se po pravilu uz pomoć trostatičkih bafera. Na slici 1 prikazan je digitalni sistem u kome se sa logičkih mreža A, B, C i D signali Y_A, Y_B, Y_C i Y_D šalju logičkoj mreži L, preko zajedničke linije. Signali dozvole E_A do E_D moraju biti generisani u različitim vremenskim intervalima, kako ne bi dolazilo do "sudaranja" na zajedničkoj izlaznoj liniji. Aktiviranje više od jednog trostatičkog bafera priključenog na istu liniju može da dovede do trajnog oštećenja bafera u slučaju da neki od aktiviranih bafera šalje jedinicu a neki drugi šalje nulu.

Još jednom, svaka izlazna mreža Y_A do Y_D ima svoj upravljački signal. Upravljački signal određuje da li je njena izlazna impedansa niska ili visoka. Uvijek samo jedna od izlaznih mreža, spojenih na zajedničku liniju, ima nisku izlaznu impedansu (nije zatvorena). Zadatak je onoga koji planira mrežu da bude osigurano da se izvrši izlaz samo iz jednog člana. U protivnom se mogu uništiti izlazni krugovi tropoložajnog člana velikim strujama u trenutku podijeljenog logičkog nivoa na njihovim izlazima.

Za mrežu oblika slike 1 kaže se da ostvaruje funkciju žičano ILI (engl. wired OR). U prethodnom naslovu smo vidjeli da multiplekser takođe može da posluži kada treba da se vrši izbor jednog izlaznog signala (među više izlaznih signala) koji će biti poslat. Samo se napominje da slične okolnosti važe kada umjesto jedne prijemne mreže (kakva je mreža L na slici 1) postoji nekoliko prijemnih mreža i treba naravno da upućeni signal dospije samo do jedne prijemne mreže (i to do one prave – izabrane).



$$\begin{cases} \text{ako je } E_A = 1,\ E_B = 0,\ E_C = 0,\ E_D = 0 \text{ onda je } Y_L = Y_A \\ \text{ako je } E_A = 0,\ E_B = 1,\ E_C = 0,\ E_D = 0 \text{ onda je } Y_L = Y_B \\ = 0 \quad = 0 \quad = 1 \quad = 0 \quad Y_L = Y_C \\ = 0 \quad = 0 \quad = 0 \quad = 1 \quad Y_L = Y_D \end{cases}$$

Slika 1 Prosleđivanje signala zajedničkom linijom

27. MAGISTRALE DIGITALNIH SIGNALA

Šta je to magistrala? Magistrala je prenosni put ili spojni put između dvije mreže ili između dva podsistema u digitalnom sistemu (od jednog podsistema ka drugom podsistemu) ili između dva registra. Sastoji se od određenog broja paralelnih vodova (paralelnih linija). Broj tih linija naziva se njenom širinom. Recimo, širina magistrale može da bude četiri ili osam linija. I jednu samu liniju možemo smatrati magistralom. Magistrala ili bus ili sabirnica. Engl. bus ili high–way. Dopunimo našu definiciju. Jedan snop paralelnih linija koristi se po pravilu za komunikaciju između nekoliko podsistema. Recimo, na ulaz tog snopa mogu da konkurišu četiri mreže, znači četiri predajne mreže. Ima mnogo podsistema i nije ekonomično da od bilo kog podsistema ka bilo kom drugom podsistemu postoji posebna veza (postoji neposredni put). Recimo, kod nekih mikroprocesora, adresna magistrala nema svojih posebnih izvoda (posebnih pinova) iz kućišta mikroprocesora već upotrebljava izvode multipleksnim načinom, zajedno sa magistralom podataka.

Mikroprocesorski sistem može da ima adresnu magistralu od recimo 20 linija, magistralu podataka od 16 linija i upravljačku magistralu od 8 linija. Sve tri magistrale zajedno u tom slučaju predstavljaju sistemsku magistralu. Na sistemsku magistralu su priključeni podsistemi koji međusobno razmjenjuju informacije. Na primjer, procesor P treba da šalje podatke u memoriju M i da prima podatke iz memorije; P upisuje podatke u M i P čita podatke iz M. Te dvije operacije se u računarskom sistemu u principu ne obavljaju u isto vrijeme. Zato ne bi bilo ekonomično da se koriste dva odvojena prenosna puta. Trostatički baferi omogućavaju realizaciju dvosmjerne magistrale (dvosmjerne magistrale podataka). Može se nacrtati šema koja prikazuje priključivanje digitalnih podsistema na magistralu. Upravljački signali E_M i E_P definišu da li će se jedna riječ (veličine 16 bita) prenijeti iz memorije u procesor ili će pak ona biti prenesena u suprotnom smjeru (iz procesora u memoriju). Prenos se ostvaruje preko magistrale podataka čija je širina jednaka šesnaest bita tj. šesnaest linija, linije od D_0 do D_{15}.

Ako je $E_M = 0$ i $E_P = 0$ onda se ne vrši nikakav prenos. Ako je $E_M = 1$ i $E_P = 0$ onda M šalje jednu riječ u P. Ako je $E_M = 0$ i $E_P = 1$ onda P šalje jednu riječ u M. Ako je $E_M = 1$ i $E_P = 1$ onda rad mreže nije definisan i ona može da pregori.

Dvosmjerna magistrala podataka upotrebljava se za prenos podataka između mikroprocesora P, radne memorije M i spoljašnjih uređaja (ulaznih i izlaznih uređaja).

Na slici 1 koriste se skraćenice DIR za DIRECTION (smjer) i OE za OUTPUT ENABLE (dozvola izlaza). Analizom mreže vidi se da je signal za dozvolu OE postavljen "obrnuto" od uobičajenog načina: dozvola je data ako i samo ako je OE $= 0$. Trostatički baferi vezani tako da propuštaju logičke signale u oba smjera nazivaju se bidirekcionim trostatičkim baferima ili trostatičkim baferima koji su primopredajnici. Trostatički baferi se proizvode kao integrisane komponente u više verzija. Mogu biti pakovani po 4–6 invertujućih ili neinvertujućih bafera sa odvojenim signalima dozvole, po 8 bafera sa zajedničkim signalom dozvole ili po 8 bidirekcionih bafera sa zajedničkim signalom dozvole. Magistrale računarskih sistema najčešće sadrže osam linija ili multiple po osam linija, tako da su integrisani trostatički baferi najčešće pakovani sa osam bidirekcionih bafera u čipu. Baferi mogu da budu invertujući ili neinvertujući, a izlazni faktor grananja je najčešće između 25 i 50. Na slici 1 prikazana je logička šema integrisanog kola 74HC640. Mreža sadrži osam bidirekcionih bafera i logička kola za generisanje zajedničkih signala dozvole. Signal OE, kada je aktivan tj. kada je na niskom logičkom nivou, aktivira trostatičke bafere u smjeru od A prema B u slučaju da je signal DIR na visokom logičkom nivou,

a u smjeru od B prema A kada je DIR $= 0$. Drugim riječima, pod uslovom OE, biće izvršeno $B \leftarrow \overline{A}$ ili $A \leftarrow \overline{B}$ u zavisnosti od DIR. Pošto se ovakvi baferi najčešće koriste za spajanje dvije magistrale, dobili su naziv bidirekcioni baferi magistrale odnosno primopredajnici magistrale.

Sredinom osamdesetih godina pojavila se jedna nova familija CMOS logičkih kola poznata kao 74 HC familija.

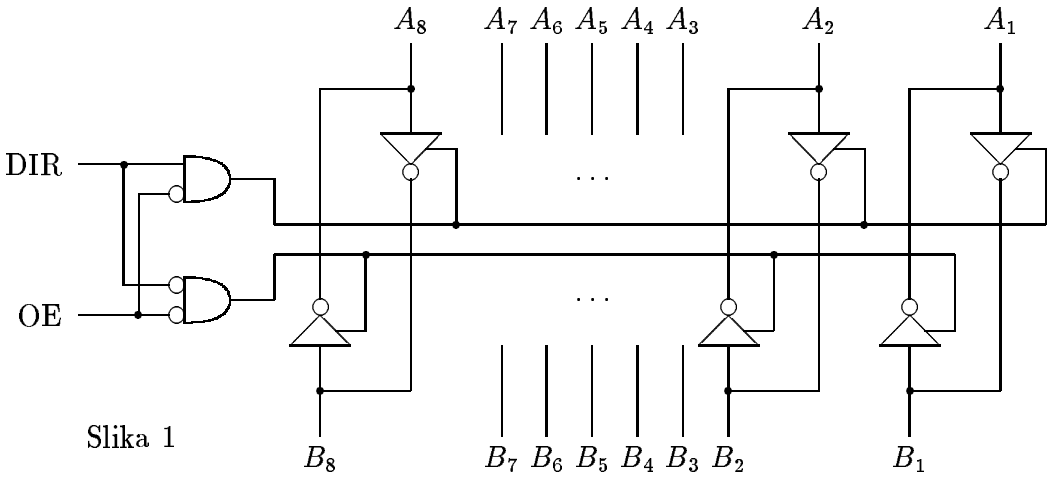
Slika 1 Logička šema komponente 74 HC 640:
Ako je OE $= 1$ onda se ništa ne prenosi. Ako je OE $= 0$ i DIR $= 0$ onda

$$(A_8, A_7, A_6, A_5, A_4, A_3, A_2, A_1) \leftarrow (\overline{B_8}, \overline{B_7}, \overline{B_6}, \overline{B_5}, \overline{B_4}, \overline{B_3}, \overline{B_2}, \overline{B_1}).$$

Ako je OE $= 0$ i DIR $= 1$ onda $(B_8, \ldots, B_1) \leftarrow (\overline{A_8}, \ldots, \overline{A_1})$.

Uz sliku 1. Smatrati da izlazna linija iz bafera uopšte nije ni nacrtana, kada baferu nije dat njegov signal dozvole (na kosu liniju dolazi).

Uz sliku 1. U šemama, kružić uvijek znači negaciju (recimo, na ulazu i–elementa). Po dvije dugačke horizontalne linije sprovodi se po gornjoj DIR & $\overline{\text{OE}}$, odnosno po donjoj $\overline{\text{DIR}}$ & $\overline{\text{OE}}$.



Slika 1

28. POJAM DIGITALNOG KOLA

Uobičajeni termin za signal koji je kontinualan po vremenu i po amplitudi je **analogni** signal. Kola koja operišu sa analognim električnim signalima nazivaju se analognim kolima.

Jednu važnu klasu analognih signala predstavljaju **impulsni** signali. Oni su kontinualni po vremenu, ali im se amplituda može naglo mijenjati, pa signal u nekim slučajevima ne može imati bilo koju vrijednost iz dozvoljenog intervala. Primjeri impulsnih signala su periodične ili aperiodične povorke pravougaonih impulsa. Kola koja generišu ili obrađuju impulsne signale nazivaju se impulsnim kolima. Primjeri impulsnih kola su generatori impulsa i povorki impulsa, fliplopovi i tajmeri.

Digitalni signali su jedna uža klasa impulsnih signala koji imaju samo nekoliko dozvoljenih amplitudskih nivoa. Najčešće se koriste **binarni** digitalni signali, gdje su definisana samo dva različita naponska nivoa. Neizbježna je tolerancija komponenti i napona napajanja. Zato se obično umjesto dva naponska nivoa definišu dva naponska opsega koji se interpretiraju kao logička jedinica i logička nula.

Električna kola koja obrađuju binarne digitalne signale nazivaju se digitalnim kolima. Sastavljena su, kao i analogna kola, od aktivnih elemenata (tranzistori) i pasivnih elemenata (otpornici i vrlo rijetko kondenzatori). Analogna kola se vrlo često izrađuju i u diskretnoj tehnologiji, dok se digitalna kola danas prave isključivo u tehnologiji integrisanih kola. Treba reći da su digitalna kola korišćena dosta prije integrisane tehnologije, pa i tranzistorske tehnologije. Prvi električni element koji je korišćen za realizaciju digitalnih kola bio je kontrolisani prekidač (relej). Sa pojavom elektronskih cijevi napravljena su impulsna i digitalna kola koja su omogućila veću brzinu rada. Prvi digitalni računar, napravljen početkom pedesetih godina, imao je sve digitalne elemente realizovane pomoću elektronskih cijevi. Sa pojavom tranzistora, digitalna kola se minijaturizuju i postaju brža. Glavni napredak u razvoju digitalnih kola došao je poslije pronalaska tehnologije integrisanih kola koja je omogućila smanjenje dimenzija i cijene, uz istovremeno povećanje brzine i kompleksnosti digitalnih kola.

Digitalna kola se prema načinu formiranja izlaznog signala dijele na **kombinaciona** i **sekvencijalna**. Kod kombinacionih digitalnih kola, signal na izlazu kola zavisi samo od trenutnih vrijednosti ulaznih signala. Kod sekvencijalnih kola, stanje na izlazu zavisi od trenutnog stanja na ulazima, ali i od prethodnih stanja na ulazima. Sekvencijalna kola se dalje dijele na **sinhrona** i **asinhrona**. Kod sinhronih kola, sve promjene dešavaju se istovremeno pod dejstvom kontrolnog signala takta. Kod asinhronih kola promjene se mogu dešavati u proizvoljnom trenutku i određene su samo osobinama upotrebljenih elemenata i vremenom pojavljivanja pobude.

29. POJAM KOMBINACIONE MREŽE

Glavni primjeri kombinacionih mreža su sabirači, dekoderi, multiplekseri i magistrale.

Za kombinacione mreže je karakteristično da ne sadrže povratnu spregu, odnosno izlazni signal sa bilo kog kola ne smije da bude doveden na ulaz mreže, kako ne bi uticao na ulaz tog istog kola. Digitalne mreže sa povratnom spregom najčešće se ponašaju kao sekvencijalne mreže.

Za praktičnu realizaciju kombinacione mreže neophodno je koristiti realne karakteristike logičkih kola, kako bi realizovana mreža pouzdano generisala predviđenu funkciju. Karakteristike koje imaju najveći uticaj za ispravan i pouzdan rad logičke mreže su kako slijedi. A)

familija logičkih kola, b) ulazno opterećenje, c) izlazni faktor grananja, d) vrijeme propagacije uzlazne i silazne ivice signala i e) margine šuma logičke nule i jedinice.

Izbor familije logičkih kola uslovljen je u prvom redu zahtijevanom brzinom rada mreže, a zatim otpornošću na smetnje, potrošnjom električne energije i cijenom.

30. POJAM SEKVENCIJALNE MREŽE

Sekvencijalne mreže (često nazivane i sekvencijalne mašine ili sekvencijalni automati) razlikuju se od kombinacionih mreža po tome što izlazni signali iz sekvencijalnih mreža zavise ne samo od tekućih vrijednosti ulaznih promjenljivih već i od redosljeda (sekvence) generisanja ulaznih signala. Činjenica da je potrebno pamtiti redosljed aktiviranja ulaznih signala uslovljava da se u sekvencijalnim mrežama moraju koristiti memorijski elementi.

Kao memorijski elementi u sekvencijalnim mrežama mogu da se koriste sve vrste flipflopova ili leč kola. Ako svi flipflopovi u mreži istovremeno mijenjaju stanje nakon uzlazne ili silazne ivice zajedničkog taktnog impulsa onda je mreža sinhrona. Ulazni signali u sinhronu mrežu, odnosno memorijski elementi sinhrone mreže ne smiju da mijenjaju stanje za vrijeme dok je taktni impuls aktivan.

Ako se u mreži koristi više od jednog taktnog impulsa ili ako neki od memorijskih elemenata može da promijeni stanje (promjenom neke od ulaznih promjenljivih) nezavisno od taktnog impulsa onda je mreža asinhrona.

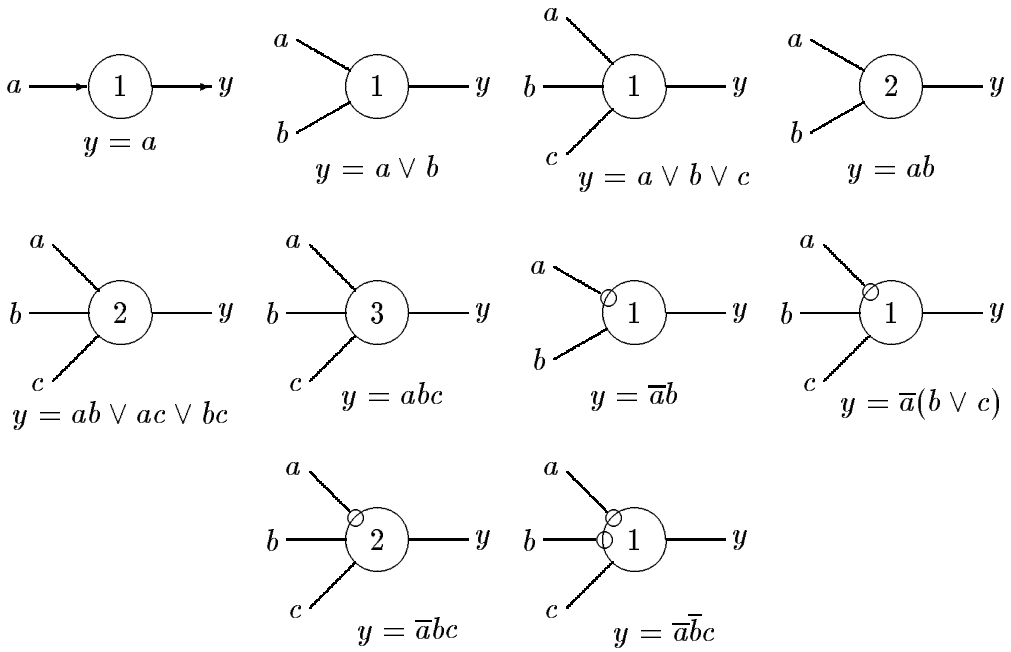
Glavni primjeri sekvencijalnih mreža su stacionarni registri, pomerački registri i brojači.

31. POJAM BISTABILNOG KOLA

U ovom naslovu biće u prvom dijelu izložen princip rada memorijskog elementa, a zatim će u drugom dijelu biti nabrojane vrste bistabilnih elemenata.

Prvi dio. Želimo da izložimo princip čuvanja podataka u računaru. Poslužimo se primjerom. Upravo, kao primjer pogledajmo predlog koji je još davno dao fon Nojman u svom izvještaju za računar Edvac. Kako se postiže da podatak bude zapamćen? U izvještaju se kao prvo razmatraju tzv. E–elementi ili kratko elementi. Šta je to element? Element ima više ulaza i ima jedan izlaz y. Element se crta kao kružić. Ka kružiću vode ulazi $a, b, \ldots$, a iz kružića vodi izlaz. Za element se definiše njegov tzv. prag (engl. threshold), u oznaci t. Broj t upisan je u kružiću. Moguće vrijednosti za t su $t = 1$, $t = 2$ i $t = 3$. Po kakvom zakonu y zavisi od ulaza? Postoje dvije vrste ulaza. Ulazi prve vrste crtaju se tako da vode pravo prema kružiću. Oni doprinose da postane $y = 1$: ako manje od t tih ulaza ima vrijednost 1 (prag nije dostignut) onda će svakako biti $y = 0$. Ulazi druge vrste crtaju se kroz mali kružić. Za ulaz druge vrste kaže se da predstavlja prepreku (engl. inhibitory), on čini prepreku da postane $y = 1$. Ako bar jedna prepreka ima vrijednost $i = 1$ onda će sigurno biti $y = 0$, bez obzira da li je prag dostignut ili nije dostignut. Vidimo da uključena prepreka ($i = 1$) utiče na apsolutni način na obrazovanje izlazne vrijednosti: sigurno će postati $y = 0$, bez obzira na bilo kakve druge okolnosti. S druge strane, isključena prepreka ($i = 0$) ne utiče na rad elementa. Dakle, ako su sve prepreke isključene i ako je (pomoću ulaza prve vrste) dostignut ili premašen prag onda je $y = 1$.

Na slici 1 prikazan je niz vrsta elemenata. Za svaku vrstu izvedena je odgovarajuća jednačina, tj. Bulov izraz u kome učestvuju ulazne veličine. Element je očito jedna fizička ili hardverska komponenta računara. Da se element izradi, treba povezati nekoliko releja. O tome nećemo govoriti.



Slika 1 E–elementi i njihove jednačine

Za vježbu, izraziti elemente preko uobičajenih logičkih elemenata I, ILI, NI, NILI i NE.

Slika 2 b) služi da ilustruje princip rada memorijskog elementa, a slika 2 a) služi za pripremu. Na slici 2 a) prikazan je element čiji je prag $t = 1$. Ima jednu prepreku a i još ima dva ulaza koji doprinose b i c. Malom modifikacijom nastaje drugi jedan element koji je prikazan na slici 2 b) i koji je za nas najviše interesantan. Upravo, vidi se da je signal sa izlazne linije y doveden na ulaznu liniju c. Vidimo da je sada krug zatvoren. Kaže se da postoji petlja ili da postoji povratna sprega, engl. feedback. Kako funkcioniše razmatrana mreža? Naponski nivoi u petlji brzo se ustale. Naponski nivo na izlaznoj liniji ustaliće se na vrijednosti $y' = 0$ ili na vrijednosti $y' = 1$, što zavisi od okolnosti. Upravo, zavisi od tekućih vrijednosti ulaza a i b i od ranije vrijednosti y. Očito y znači staru vrijednost na izlazu, a y' znači novu vrijednost na izlazu. Drugim riječima, y se odnosi na $(n-1)$–vi takt, dok se a, b i y' odnose na n–ti takt. Dakle, novo stanje na izlazu zavisi i od prethodnog stanja na izlazu, što je svojstvo svake sekvencijalne mreže, kao što znamo. Samo se napominje da postoje slučajevi povratne sprege gdje stalna (stabilna) vrijednost izlaza y neće biti obrazovana; slučaj nestabilne ili nedefinisane izlazne vrijednosti. Ili će vrijednost izlaza biti obrazovana, ali od sitnih slučajnih okolnosti zavisi da li će postati $y = 0$ ili $y = 1$; konfiguracija mreže je takva da ne obezbjeđuje pouzdan odziv.

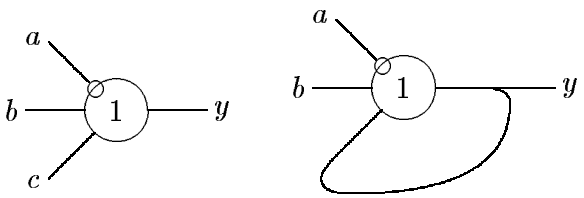
Mi govorimo o funkcionisanju razmatrane mreže, pa je najkorisnije da napišemo njenu jednačinu $y \leftarrow \overline{a} \cdot (b \vee y)$ ili bolje $y' = \overline{a} \cdot (b \vee y)$ ili eventualno $y_n = \overline{a} \cdot (b \vee y_{n-1})$.

Na osnovu jednačine, lako se vidi da kolo sa slike 2 b) ima sljedeća svojstva. Ako nema pobude, tj. ako nema ni jednog ni drugog ulaznog signala ($a = 0$ i $b = 0$) onda važi $y' = y$; izlazna vrijednost ostaje po starom. Ako je postavljen samo ulazni signal $b = 1$ onda će postati $y' = 1$, bez obzira čemu je bilo jednako y; upisivanje jedinice. $a = 1 \Rightarrow y' = 0$; upisivanje 0. Prikažimo kao tabelu:

a	b	y_n
0	0	y_{n-1}
0	1	1
1	0	0
1	1	0

U zaključku, zašto se za razmatrano kolo kaže da ono predstavlja jedan memorijski element, odnosno da ono pamti jedan bit (jednu binarnu cifru). Zato što se izlazna vrijednost neće mijenjati (vrijednost y koja se pamti ostaje po starom) koliko god taktova prošlo, samo ako nema pobude. I zato što se kolom može upravljati: ako se na ulaze dovedu odgovarajuće vrijednosti onda će se izlaz postaviti na željenu vrijednost ili izlaz će se promijeniti na željeni način.

Priča o savremenim memorijskim elementima samo po obliku se razlikuje od dosadašnje priče.

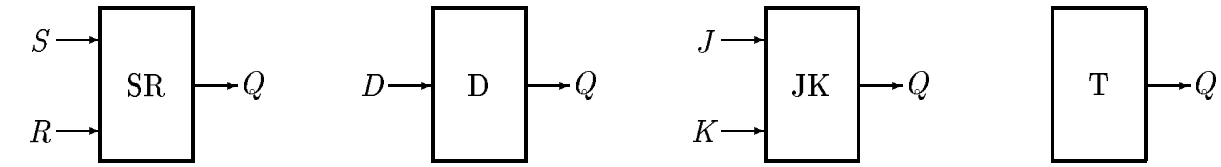


a) $y = \overline{a}(b \vee c)$ b) $y' = \overline{a}(b \vee y)$

Slika 2 Kako da se od E–elementa dobije memorijski element

Drugi dio. U savremenim računarima koriste se četiri vrste trigera, tj. bistabilnih kola i to: SR kolo, D kolo, JK kolo i T kolo, v. sliku 3. Q = stanje kola = izlaz kola = vrijednost koju kolo pamti. O tim kolima biće detaljno riječi u nastavku (u sljedećim naslovima).

1 Leč kolo, engl. latch – izlaz se jednostavno postavlja na određenu vrijednost (prebacuje) pod dejstvom ulaza. 2 Flipflop – prebacivanje je uslovljeno signalom takta.

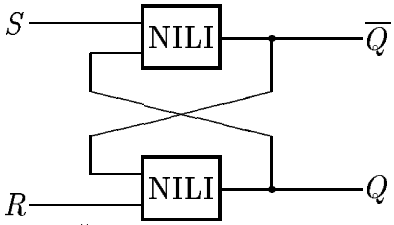


$$\begin{cases} S{=}0, R{=}0 & Q{\leftarrow}Q \\ S{=}0, R{=}1 & Q{\leftarrow}0 \\ S{=}1, R{=}0 & Q{\leftarrow}1 \\ S{=}1, R{=}1 & Q{\leftarrow}? \end{cases} \qquad \begin{cases} D{=}0 & Q{\leftarrow}0 \\ D{=}1 & Q{\leftarrow}1 \end{cases} \qquad \begin{cases} J{=}0, K{=}0 & Q{\leftarrow}Q \\ J{=}0, K{=}1 & Q{\leftarrow}0 \\ J{=}1, K{=}0 & Q{\leftarrow}1 \\ J{=}1, K{=}1 & Q{\leftarrow}\overline{Q} \end{cases} \qquad Q{\leftarrow}\overline{Q}$$

Slika 3 Grafički simboli za memorijske elemente i njihovo funkcionisanje

32. SR LEČ KOLO

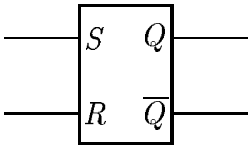
Treba izvršiti analizu mreže prikazane na slici 1 a). Pratimo logičke vrijednosti (nizak napon za nulu i visok napon za jedinicu) po njenim ulaznim i izlaznim linijama i u njenim karakterističnim tačkama, pratimo kako se te vrijednosti mijenjaju po vremenskoj osi. Vidimo da je u mreži prisutna povratna sprega. Znamo da se tada (uopšte uzev) može desiti da se stabilne naponske vrijednosti uopšte ne uspostave ili da se one uspostave ali na "slučajan" način, koji zavisi od okolnosti koje ne mogu da se predvide, tzv. nepravilne situacije. Za analizu, treba znati definiciju NILI–kola (nije ili): $y = \text{NILI}(a, b)$ znači $y = \overline{a \vee b}$. Prethodno, tokom pravilnog rada kola, po dvije izlazne linije uvijek se saopštavaju dvije međusobno komplementarne vrijednosti; zato su te dvije izlazne linije i označene kao Q i $\overline{Q}$. Analiza se sastoji od četiri slučaja. Prvi slučaj: $S = 0$ i $R = 0$, tj. po jednoj i drugoj ulaznoj liniji dovodi se logička nula. Tada na ulaz gornjeg NILI–kola pristižu $S = 0$ i Q tj. Q zatečeno, tako da će izlaz iz tog NILI–kola biti svakako jednak $y = \overline{0 \vee Q}$. A na ulaz donjeg NILI–kola tada pristižu $R = 0$ i $\overline{Q}$ zatečeno. Analiza se ovdje može razdvojiti na dva podslučaja: prvi: Q zatečeno $= 0$ i samim tim $\overline{Q}$ zatečeno $= 1$ i drugi: Q zatečeno $= 1$ i samim tim $\overline{Q}$ dosadašnje $= 0$. Drugi, treći i četvrti slučaj odnose se na vrijednosti $(S, R) = (0, 1)$, $(S, R) = (1, 0)$ i $(S, R) = (1, 1)$. U zaključku, jedno leč kolo pamti jedan bit. Ono po liniji Q saopštava vrijednost koju pamti (a po liniji $\overline{Q}$ saopštava još i komplementarnu vrijednost). Kolom se upravlja preko dvije linije S i R: dovođenjem odgovarajućih signala po tim linijama utiče se na bit koji se pamti. Ako se po obe linije dovodi nula onda se izlaz Q ne mijenja. To je tzv. obični slučaj. U običnom slučaju kolo očito pamti svoju upisanu vrijednost.



1a) Šema SR leč kola

S	R	Q_{n+1}	$\overline{Q_{n+1}}$
0	0	Q_n	$\overline{Q_n}$
0	1	0	1
1	0	1	0
1	1	0	0

1b) Tabela



1c) Simbol

Na slici 1 a) prikazano je bistabilno kolo realizovano sa NILI logičkim kolima koje se naziva SR leč kolo. Slobodni ulazi logičkih kola označeni su sa S i R, S znači set – postavi – upiši jedinicu, a R znači reset – vrati se – upiši nulu. Izlazi su označeni sa Q i $\overline{Q}$, oni moraju da budu komplementarni. Kada su izlazni nivoi $Q = 1$ i $\overline{Q} = 0$ kaže se da je leč kolo setovano, dok se za slučaj kada je $Q = 0$ i $\overline{Q} = 1$ kaže da je leč kolo resetovano.

Znajući definiciju NILI kola, vidi se da se dovođenjem kombinacije $S = 1$, $R = 0$ na ulaze kola dešava da se izlazi postave u stanje $Q = 1$, $\overline{Q} = 0$, kolo je sada setovano. Dovođenjem kombinacije $S = 0$, $R = 1$ izlazi će se postaviti u novo stanje $Q = 0$, $\overline{Q} = 1$, odnosno kolo će sada biti resetovano. Vidimo da se postavljanje kola u željeno stanje vrši dovođenjem logičke jedinice na odgovarajući ulaz. Kada se na ulazu nalazi kombinacija $S = 0$, $R = 0$ (oba ulazna signala su na neaktivnom nivou) onda se na izlazu ne dešava nikakva promjena.

Pogledajmo slučaj $S = R = 1$. Ako se na ulazu pojavi $S = 1$, $R = 1$ onda će se oba izlaza dovesti u stanje logičke nule i (znači) izlazi će prestati da budu komplementarni. U okviru ovog, poslije prelaska pobude iz stanja $S = 1$, $R = 1$ u stanje $S = 0$, $R = 0$ ne može da bude predviđeno stanje na izlazima; stanje na izlazima zavisi od toga koji se ulazni signal (S ili R) prvi promijenio. Zato se kombinacija $S = R = 1$ naziva zabranjenim stanjem ili nedozvoljenim stanjem na ulazu. Ako se SR leč kolo pojavljuje kao dio u nekoj većoj mreži onda je zadatak

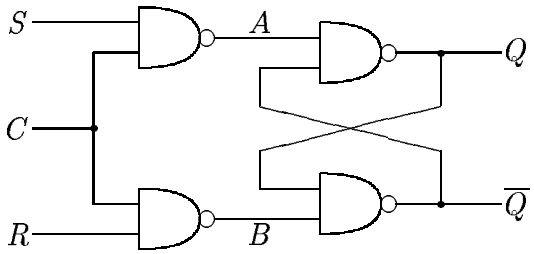
onoga koji projektuje mrežu da osigura da na ulazima kola nikad neće biti zabranjeno stanje.

Na osnovu dosadašnjeg razmatranja rada SR leč kola formirana je njegova funkcionalna tabela 1 b). Tabela daje stanja na izlazima za sve moguće kombinacije stanja na ulazima. Takva tabela naziva se funkcionalnom tabelom ili karakterističnom tabelom kola. U tabeli, Q_n označava trenutno stanje izlaza Q, dok Q_{n+1} označava naredno stanje izlaza Q, odnosno stanje izlaza Q poslije promjene ulaznih signala.

Na slici 1 c) prikazan je grafički simbol za SR leč kolo.

Prilikom sinteze složenih sekvencijalnih mreža, osim funkcionalne tabele, često se koristi i eksitaciona tabela ili tabela pobude. Eksitaciona tabela se može izvesti iz funkcionalne tabele, a definiše vrijednosti ulaznih signala pomoću kojih se kolo prevodi u željeno stanje. Tako eksitaciona tabela SR leč kola govori sljedeće: ako se kolo želi prebaciti iz stanja $Q_n = \ldots$ u stanje $Q_{n+1} = \ldots$ onda treba namjestiti $S = \ldots$ i $R = \ldots$.

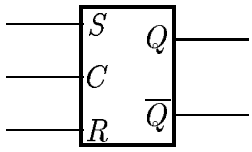
Rad leč kola može da bude opisan i pomoću njegove tzv. logičke jednačine (njegove formule); isto se kaže i funkcionalna jednačina ili karakteristična jednačina. Logička jednačina neposredno se dobija iz funkcionalne tabele. Logička jednačina (formula) izražava novo stanje izlaza Q_{n+1} kao funkciju od dosadašnjeg stanja izlaza Q_n i dva ulaza S i R, $Q_{n+1} = Q_{n+1}(Q_n, S, R)$. Logička jednačina glasi $Q_{n+1} = S\overline{R} \vee \overline{S}\,\overline{R}Q_n$. Ako se koristi činjenica da za dozvoljena stanja na izlazima nije istovremeno $S = 1$ i $R = 1$ onda se ova logička jednačina svodi na jednostavniji oblik. Upravo, ona se svodi na $Q_{n+1} = S \vee \overline{R}Q_n$; koristili smo zakon apsorpcije. Ponovimo da posljednja jednačina odgovara za slučajeve u kojima je ispunjeno $S = 0$ ili $R = 0$.



2a) SR leč sa signalom dozvole

S	R	C	Q_{n+1}	$\overline{Q_{n+1}}$	
0	0	1	Q_n	$\overline{Q_n}$	$A = 1, \quad B = 1$
0	1	1	0	1	$A = 1, \quad B = 0$
1	0	1	1	0	$A = 0, \quad B = 1$
1	1	1	1	1	$A = 0, \quad B = 0$
x	x	0	Q_n	$\overline{Q_n}$	$A = 1, \quad B = 1$

2b) Tabela. Na osnovu S, R i C formiraju se A i B, na osnovu A, B, Q staro i $\overline{Q}$ staro formiraju se Q novo i $\overline{Q}$ novo



2c) Simbol

Rad SR leč kola može da bude opisan i pomoću njegovih tzv. vremenskih dijagrama. U katalozima integrisanih kola, rad kola često se opisuje na takav način. Vremenski dijagrami izražavaju $Q = Q(t)$ i $\overline{Q} = \overline{Q}(t)$ u zavisnosti od $S = S(t)$ i $R = R(t)$; t – vrijeme. Drugim riječima, na vremenskim dijagramima je prikazano kako se tokom vremena mijenjaju četiri funkcije, gdje su očito prve dvije $Q = Q(t)$ i $\overline{Q} = \overline{Q}(t)$ uslovljene sa druge dvije $S = S(t)$ i $R = R(t)$. Obuhvaćeni su razni slučajevi, kada te funkcije tokom vremena mijenjaju vrijednosti.

Na slici 1 a) prikazan je jedan mogući način da se sastavi SR leč kolo. Napominje se da postoje i drugi načini (druge šeme) koje takođe realizuju SR leč kolo. U šemama se koriste logički elementi ILI, NILI, I, NI i NE. Tada ostaju da važe dosadašnja funkcionalna tabela, eksitaciona tabela, jednačina i vremenski dijagrami, a takođe naravno i grafički simbol kola.

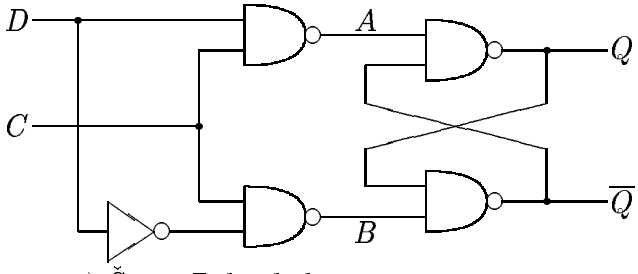
Prelazimo na drugo kolo (sa dozvolom). Dosad razmatrano SR leč kolo reaguje na promjenu ulaznih signala u bilo kom trenutku. Međutim, često se ukazuje potreba za leč kolom koje može da mijenja svoje stanje samo u određenim vremenskim intervalima, samo u vremenskim intervalima kada je aktivan taktni signal C. U literaturi se za C sreću i oznake CLK, EN, $ENABLE$ i G. Takvo leč kolo se naziva SR leč kolo sa signalom dozvole. Prikazano je na slici 2 a). Vidi se da je realizovano pomoću NI kola. Funkcionalna tabela ovog leč kola prikazana je na slici 2 b). Za tabelu, na osnovu S, R i C obrazuju se među–vrijednosti A i B. A i B su ulazi u petlju povratne sprege u kojoj učestvuju još i Q i $\overline{Q}$. Za razne slučajeve koji mogu da nastupe, proučiti kako će se formirati naponski nivoi u raznim čvorovima koji su obuhvaćeni petljom (da li će se stabilni naponski nivoi formirati). Slika 2 c) – grafički simbol za SR leč kolo sa signalom dozvole. Logička jednačina SR leč kola sa dozvolom glasi $Q_{n+1} = CS \vee \overline{CR}Q_n$.

Ako je kontrolni signal C periodični signal takta onda se dobija taktovano SR kolo ili svejedno sinhrono SR kolo.

33. D LEČ KOLO

Vidjeli smo da SR leč kolo ima razdvojene ulaze za setovanje odnosno za resetovanje. Razdvojeni ulazi pogodni su za primjene u kontrolnim sistemima. Međutim, za primjene u sistemima za pamćenje informacija pogodnije je imati samo jedan ulaz u leč kolo. Taj ulaz će određivati stanje na izlazu. Još ima i kontrolni ulaz, po pravilu. Takvu funkciju obavlja D leč kolo.

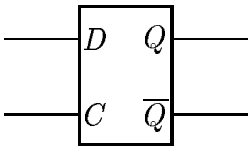
Šema D leč kola prikazana je na slici 1 a). Sa slike se vidi da osnovu za šemu D leč kola čini SR leč kolo sa dozvolom. Jedina razlika je dodatni invertor na ulazu koji uklanja mogućnost da na ulaz bude dovedena nedozvoljena kombinacija signala. Pogledajmo šemu i objasnimo funkcionisanje kola. Neka je $C = 1$. Kada je na ulazu $D = 1$ tada je $A = 0$ i $B = 1$, pa se kolo setuje ($Q \leftarrow 1$). Suprotno tome, kada je na ulazu $D = 0$ tada je $A = 1$ i $B = 0$, pa će se kolo resetovati. Dakle, na izlazu se uvijek pojavljuje isti signal koji je i na ulazu, naravno poslije vremena kašnjenja kroz logičke elemente i uspostavljanja ravnotežnog stanja u petlji povratne sprege. Kada se kontrolni signal C vrati na stanje logičke nule onda stanje na izlazu biva zamrznuto.



1a) Šema D leč kola

D	C	Q_{n+1}	$\overline{Q_{n+1}}$
0	1	0	1
1	1	1	0
x	0	Q_n	$\overline{Q_n}$

1b) Tabela



1c) Simbol

U našoj konfiguraciji D leč kola, ulazni signal dozvole C je aktivan kada je na visokom logičkom nivou (kada je jednak jedan).

Na slici 1 b) prikazana je funkcionalna tabela D leč kola. U tabeli se na jednom mjestu pojavljuje znak x. Znači da vrijednost ulaza D u tom redu tabele ne utiče na rad kola. Svejedno je da li je ta vrijednost nula ili jedan; isto će se desiti na izlazima kola.

Na slici 1 c) prikazan je grafički simbol D leč kola.

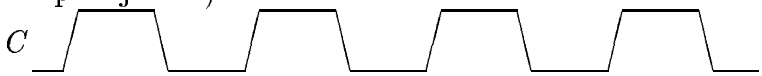
Osnovni vremenski parametri D leč kola su t_{su} – setup time, vrijeme postavljanja i t_h – hold time, vrijeme držanja. Za pravilan rad kola, signal na ulazu D treba da bude stabilan tokom

najmanje t_{su} ns prije opadanja signala dozvole sa $C = 1$ na $C = 0$. Isto tako, zahtijeva se da signal na ulazu D bude stabilan za vrijeme najmanje t_h poslije opadanja signala dozvole sa $C = 1$ na $C = 0$. Zašto treba t_{su}? Potrebno je određeno vrijeme da struja prođe kroz logičke elemente, tj. da se uspostave vrijednosti na izlazima kola. Zašto treba t_h? Signal D ne dolazi u istom trenutku na element čiji je izlaz A i na element čiji je izlaz B, jer je prisutan invertor, koji ima svoje vrijeme kašnjenja. Ako bi se signal D mijenjao još dok je $C = 1$ onda bi moglo da se desi da postane i $A = 0$ i $B = 0$, nedozvoljena kombinacija.

34. SINHRONI FLIPFLOPOVI SR I D

Često se koriste bistabilni elementi kod kojih se promjena stanja može vršiti samo u tačno određenim trenucima vremena koji su određeni taktom sistema. Takvi bistabilni elementi nazivaju se flipflopovima. Promjena stanja može se vršiti prilikom promjene logičke vrijednosti ulaza na koji se dovodi takt.

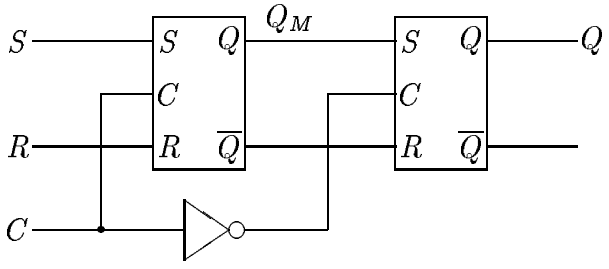
U praksi se sreću dva načina okidanja flipflopa: impulsni način gdje se okidanje vrši cijelim jednim takt impulsom i ivični način gdje se okidanje vrši sinhrono sa ivicom takta (rastućom ili opadajućom).



Slika Vremenski dijagrami signala takta C

Kaže se flipflop sa impulsnim okidanjem ili se kaže MS flipflop, engl. master–slave. Realizacija ove ideje može se izvršiti korišćenjem dva leč kola, v. sliku 1 a). U prvo leč kolo (master kolo) informacija sa ulaza upisuje se poslije uzlazne ivice takta, a u drugo leč kolo (slave kolo) informacija sa izlaza prvog leč kola upisuje se poslije silazne ivice takta. Na osnovu poznavanja rada SR leč kola i na osnovu maločas opisanog načina rada MS SR flipflopa sa slike 1 a), lako se izvodi njegova funkcionalna tabela, v. sliku 1 b). Na slici 1 c) prikazan je odgovarajući grafički simbol. Mogu se prikazati i vremenski dijagrami razmatranog flipflopa. Pobudni signal (S ili R) treba da postane aktivan odmah poslije uzlazne ivice takta C. Ne traži se da on ostane aktivan sve do silazne ivice, već samo dok se uspostave izlazi master kola, jer onda ti izlazi pobuđuju slave kolo. Ukupno, tokom jedne periode takta, može jednom da dođe do promjene stanja izlaznog signala flipflopa (do promjene vrijednosti koju flipflop pamti).

Za pouzdan rad kola treba dobro usaglasiti ulazne signale sa oblikom signala dozvole i sa vremenima kašnjenja logičkih elemenata kola. Za pojedini ulazni signal propisuje se: kada će stabilan nivo signala biti dostignut i koliko će onda takav nivo trajati.



Slika 1a) SR flipflop sa impulsnim okidanjem

S	R	C	Q_{n+1}	$\overline{Q_{n+1}}$
0	0		Q_n	$\overline{Q_n}$
0	1		0	1
1	0		1	0
1	1		?	?

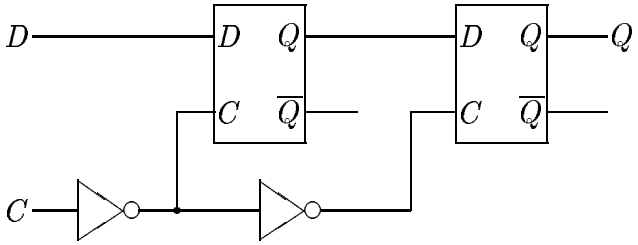
1b) Tabela

1c) Simbol

Na slici 2 a) prikazana je šema D flipflopa sa ivičnim okidanjem, sa okidanjem (prebacivanjem) na rastuću ivicu takta. Samo se napominje da se slično definiše i D flipflop koji se okida opadajućom ivicom takta. Posmatrajmo kolo D flipflopa prikazano na slici 2 a). Služi nam kao

primjer flipflopa sa ivičnim okidanjem. Vidimo da u sastavu flipflopa učestvuju dva leč kola. Za prvo D leč kolo kaže se da je master kolo, a za drugo D leč kolo kaže se da je izvršno kolo.

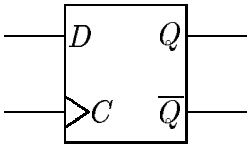
Kada je signal takta na niskom nivou onda je moguće upisivati informaciju u master kolo. Dovođenjem rastuće ivice takta, signal dozvole master kola ide na nivo nule i tako se stanje master kola zamrzava. Istovremeno, ako se zanemari kašnjenje kroz invertor, signal dozvole izvršnog kola ide na nivo jedinice. Drugim riječima, sada se aktivira drugo D leč kolo (izvršno). Sada se stanje prvog D leč kola upisuje u drugo D leč kolo. Time se završava proces upisivanja u flipflop. Na slici 2 b) prikazana je odgovarajuća funkcionalna tabela, funkcionalna tabela D flipflopa sa ivičnim okidanjem. Na slici 2 c) prikazan je odgovarajući grafički simbol. Uočava se trougao na CLK ulazu; taj ulaz se često označava i kao C ili CP. Taj trougao govori da je okidanje ivično ili dinamičko. Ivično okidanje razlikuje se od impulsnog okidanja po tome što u slučaju ivičnig okidanja pobuda D nastupa u "okolini" ivice takta (uzlazne ivice takta); pobuda počinje malo prije ivice a traje do malo poslije nje.



2a) D flipflop sa ivičnim okidanjem

D	C	Q_{n+1}	$\overline{Q_{n+1}}$
0	↓	0	1
1	↓	1	0

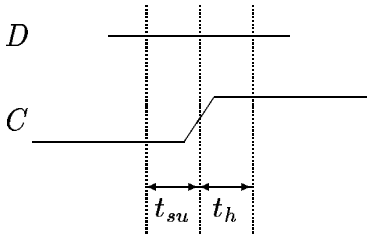
2b) Tabela



2c) Simbol

Propisuju se parametri t_{su} i t_h. U trenutku $t = t_1$ ulaz D dostigao je stabilan nivo i neće se mijenjati do trenutka $t = t_3$. Trenutak $t = t_2$ odnosi se na rastuću ivicu takta (ulaz C). Ovdje je $t_1 < t_2 < t_3$. Treba da bude $t_2 - t_1 > t_{su}$ i $t_3 - t_2 > t_h$. V. sliku – vremenski dijagrani. Na vremenskim dijagramima nacrtani su jedan iznad drugog grafici dvije funkcije $D = D(t)$ i $C = C(t)$.

Još nešto. Sinhroni flipflopovi mogu dodatno imati i asinhrone, odnosno direktne priključke oblika S i R koji služe za direktno setovanje i resetovanje. Dovođenjem signala na asinhrone ulaze, mijenja se stanje flipflopa, nezavisno od stanja na C i D ulazima. Znači, ti signali će djelovati na stanje flipflopa nezavisno od toga u kom trenutku oni nastupaju (nezavisno od stanja ostalih ulaznih signala). Ulaz za asinhrono setovanje obično se obilježava kao S_D ili PR, engl. preset. Ulaz za asinhrono resetovanje obično se obilježava kao R_D ili CLR, engl. clear. D – direct. Asinhroni ulazi obično se koriste za jednostavne operacije. Na primjer, koriste se za postavljanje flipflopa u određeno stanje prilikom inicijalizacije sistema.



Slika – vremenski dijagrami

35. FLIPFLOPOVI JK I T

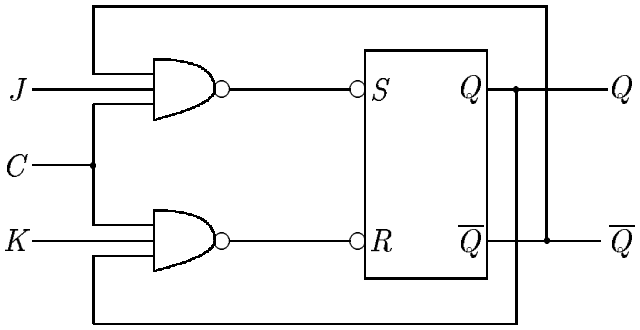
Kada su S i R ulazi SR leč kola istovremeno aktivni onda je stanje na izlazu kola nedefinisano. Ovaj problem rješava se uvođenjem novog tipa bistabilnog kola, uvođenjem JK flipflopa. Postoje brojne realizacije JK flipflopova u TTL i CMOS tehnologijama. Postoje razna kola za koja se može reći da predstavljaju JK flipflop. Jedno takvo kolo prikazano je na slici 1 a). Vidimo da je formirano od SR leč kola i dva NI kola. Preko NI kola dovode se nove povratne sprege sa izlaza na ulaze. Sa slike se vidi da se promjena stanja flipflopa može vršiti samo kada je takt C na visokom nivou. Takođe se vidi da ulaz J služi za setovanje flipflopa, a ulaz K služi za njegovo resetovanje. Zanimljivo je razmotriti šta će se desiti kada se na ulaze dovede stanje $J = K = 1$. U tom slučaju se komplementira stanje flipflopa, zbog dejstva povratnih sprega. Recimo, ako je $Q_n = 1$, $\overline{Q_n} = 0$ onda NI kolo na koje se dovodi ulaz K daje na izlazu logičku nulu. Zato se leč kolo resetuje, odnosno flipflop dolazi u stanje $Q_{n+1} = 0$, $\overline{Q_{n+1}} = 1$. Slično ako je bilo $Q_n = 0$ (umjesto $Q_n = 1$). Za ostale kombinacije ulaznih signala lako se utvrđuje stanje izlaza. To je prikazano u funkcionalnoj tabeli 1 b). Grafički simbol flipflopa prikazan je na slici 1 c). Karakteristična jednačina JK flipflopa glasi $Q_{n+1} = J\,\overline{Q_n} \vee \overline{K}\,Q_n$.

Posmatrajmo opet rad kola sa slike 1 a) u slučaju da je $C = 1$ i da je $J = K = 1$. Tada se komplementira izlaz Q, a i $\overline{Q}$ naravno. Izlazi se dovode na ulaze. Ako je takt impuls još uvijek aktivan, tj. ako je još uvijek $C = 1$ onda će se izlazi još jednom komplementirati, zato što su ulazi u stanju $J = K = 1$. Mi naravno ne želimo da se još jednom komplementira. Kako da se prevaziđe ovaj nedostatak kola? Kolo će ispravno raditi samo ako je takt impuls vrlo kratak. Bolje rečeno, takt treba da bude kraći od vremena kašnjenja kroz dva NI kola i SR leč kolo. Vrijeme kašnjenja podložno je velikim varijacijama usljed proizvodnih tolerancija i promjena temperature ambijenta. Nije pogodno da se oslonimo na to da će vrijeme kašnjenja biti veće ili jednako od neke vrijednosti. Rad opisanog flipflopa može da bude nepouzdan. Zato u njegovoj konfiguraciji treba izvršiti neke izmjene i dogradnje. JK flipflopovi uvijek se realizuju pomoću složenijih konfiguracija sa MS okidanjem (⎍) ili sa okidanjem na rastuću ili opadajuću ivicu (⎍ ili ⎍) .

T flipflop je ivični flipflop koji mijenja stanje na svaku rastuću ivicu takta. Može da bude realizovan recimo pomoću D flipflopa ili pomoću JK flipflopa. Karakteristična jednačina T flipflopa je vrlo prosta i glasi $Q_{n+1} = \overline{Q_n}$. Izlazni signal iz T flipflopa ima tačno dva puta manju učestanost od takta, pa je zato glavna primjena T flipflopa u djeliteljima učestanosti.

U nekim primjenama potrebno je zabraniti okidanje T flipflopa. Takva funkcija se ostvaruje pomoću T flipflopa sa dozvolom, koji takođe može da bude realizovan recimo pomoću D flipflopa ili pomoću JK flipflopa. Za ulaz dozvole se propisuje vrijeme postavljanja t_{su} i vrijeme držanja t_h u odnosu na okidnu ivicu takta. Karakteristična jednačina T flipflopa sa dozvolom glasi $Q_{n+1} = \overline{EN} \cdot Q_n \vee EN \cdot \overline{Q_n}$.

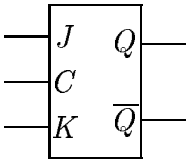
Slika 1 a) Šema JK flipflopa realizovanog sa NI kolima
Slika 1 b) Funkcionalna tabela JK flipflopa
Slika 1 c) Grafički simbol JK flipflopa
Slika 2 a) Grafički simbol T flipflopa
Slika 2 b) Vremenski dijagrami T flipflopa
Slika 2 c) Grafički simbol T flipflopa sa dozvolom



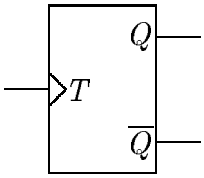
Slika 1 a)

J	K	C	Q_{n+1}	$\overline{Q_{n+1}}$
0	0	1	Q_n	$\overline{Q_n}$
0	1	1	0	1
1	0	1	1	0
1	1	1	$\overline{Q_n}$	Q_n
x	x	0	Q_n	$\overline{Q_n}$

Slika 1 b)



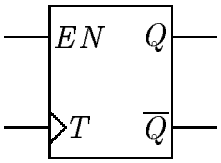
Slika 1 c)



Slika 2 a)

t od–do	do 0	0–1	1–2	2–5	5–6	6–10	10–11	11–12	12–15	15–16	16–20	20–21	$\cdots$
T	0	↗	1	1	↘	0	↗	1	1	↘	0		
Q	0	0	↗	1	1	1	1	↘	0	0	0		

Slika 2 b) Kada je $0 < t < 2$ onda T utiče na Q. Kada je $10 < t < 12$ onda T utiče na Q. Ponavlja se nakon $t = 20$



Slika 2 c)

36. STACIONARNI REGISTRI

Registar ili stacionarni registar je sekvencijalna mreža koja se koristi za privremeno memorisanje digitalnih informacija. Kao memorijski elementi obično se koriste D flipflopovi, a takođe se koriste D ili SR leč kola. Da bi dva ili više bistabilnih kola predstavljalo registar, uslov je da imaju zajednički taktni impuls.

Stacionarni registar može da se koristi i za privremeno memorisanje digitalnih podataka prilikom asinhrone razmjene informacija, odnosno između digitalnih uređaja različitih brzina. U takvoj ulozi registar se naziva bafer, po engleskoj riječi buffer.

Stacionarni registri koji se najčešće upotrebljavaju koriste ivične D flipflopove kao svoje memorijske elemente. Na slici 1 prikazana je logička šema stacionarnog registra u kome se flipflopovi aktiviraju uzlaznom ivicom taktnog impulsa, tj. ulazna informacija upisuje se u registar na uzlaznu ivicu taktnog impulsa. Registar sa slike 1 ima i priključak za istovremeno resetovanje svih flipflopova. Signal za reset dovodi se na asinhrone R_D ulaze flipflopova.

Kao integrisane komponente, registri se obično prave sa $n = 2$, $n = 4$ ili $n = 8$ flipflopova. Mogućnost reseta nije uvijek neophodna, tako da se proizvode i stacionarni registri bez priključka za reset.

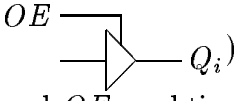
Za međusobni vremenski odnos i trajanje ulaznih signala registra važi sve što je ranije rečeno za flipflopove. Izlazi flipflopova zauzeće stabilno stanje nakon vremena propagacije t_d od aktivne ivice taktnog impulsa. Za vrijeme dok je signal za reset CLR aktivan u registar se ne može upisati nikakav sadržaj. U praksi je signal za reset CLR kod većine registara aktivan na niskom nivou.

Smisao stacionarnog registra: ako takt onda $Q_i \leftarrow D_i$ za $i = 0, 1, \ldots, 7$, tj. pod uslovom CLK biće urađeno $Q_0Q_1 \ldots Q_7 \leftarrow D_0D_1 \ldots D_7$. Drugim riječima, ulazni podaci D_i upisuju se u flipflopove koji čine registar. A upisane binarne vrijednosti saopštavaju se po linijama Q_i.

Uobičajeno je da se ulazne promjenljive obilježavaju sa D_i ili DAT_i (Data) ili slovima $A, B, C, \ldots$. Taktni impuls obilježava se sa CLK ili C (Clock) ili sa PE (Parallel Entry). Signal koji resetuje sve flipflopove u registru obilježava se obično sa CLR (Clear) ili MR (Master Reset). Najzad, izlazi iz registra obilježavaju se sa Q_i; to su stanja flipflopova, odnosno to su binarne vrijednosti koje se memorišu.

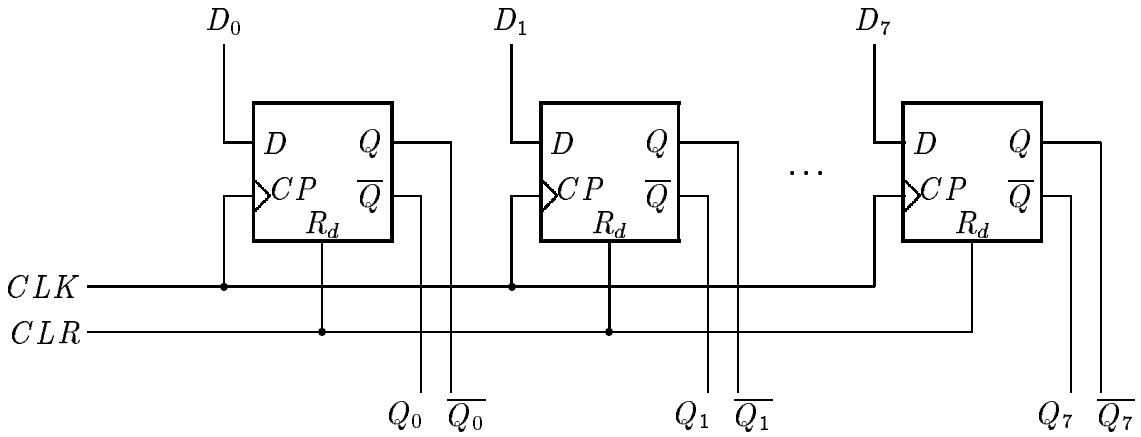
Na slikama od 2 do 4 prikazani su standardni simboli za stacionarne registre. Tako je na slici 2 prikazan simbol za registar sa slike 1. Na slici 3 prikazan je simbol za registar koji memoriše ulaznu informaciju nakon uzlazne ivice taktnog impulsa. Na tom registru, komplementarni izlazi flipflopova nisu izvedeni na nožice integrisanog kola. Najzad, registar sa slike 4 je takođe aktivan na uzlaznu ivicu takta, nema mogućnosti asinhronog reseta i nema komplementarnih izlaza flipflopova.

Postoje brojne vrste stacionarnih registara:

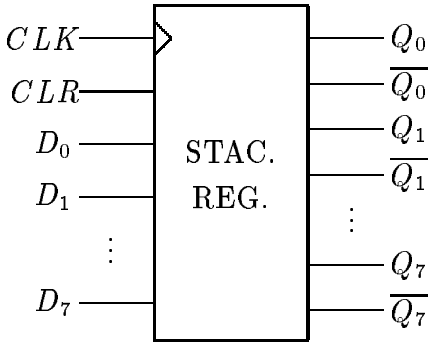
1 Stacionarni registar čiji su izlazi Q_i trostatički () poznaje se po prisustvu ulaznog priključka OE (Output Enable). Kada je signal OE neaktivan onda su izlazi registra u stanju visoke impedanse (praktično, izlazne vrijednosti se ne saopštavaju). Takvi registri primjenjuju se u sistemima gdje više digitalnih uređaja razmjenjuje informacije preko zajedničke magistrale. (Kada je $OE = 1$ radi kako je i uobičajeno.)

2 Bidirekcioni stacionarni registri. U slučaju kad digitalni uređaj treba i da prima informacije sa magistrale i da šalje informacije na magistralu (da se ne bi koristila dva registra) koristi se samo jedan bidirekcioni stacionarni registar.

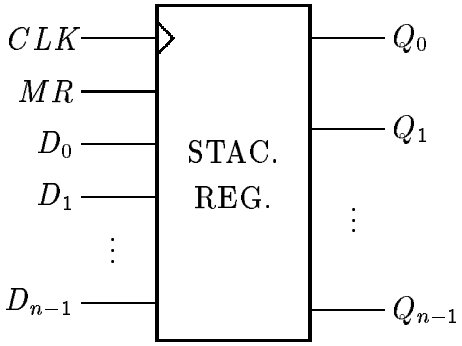
3 Bit–adresibilni stacionarni registar ili skraćeno adresibilni registar je registar kod koga informacija biva upisana u samo jedan njegov bit (u adresirani bit). Ulaz DAT donosi informaciju koja će biti upisana. Uzmimo da je $n = 8$, tj. uzmimo da ima $n = 8$ izlaza $Q_0, Q_1, \ldots, Q_7$. Ulazi A_0, A_1 i A_2 prolaze kroz dekoder oblika 3/8. Tri ulazne vrijednosti A_0, A_1 i A_2 određuju jedan broj i u granicama $0 \leq i \leq 7$. Koju funkciju obavlja takav registar? Pod uslovom ulaza za takt CLK, izvršiće se pridruživanje $Q_i \leftarrow DAT$. Dakle, ulazna informacija DAT upisaće se u adresirani flipflop (u odabrani flipflop).



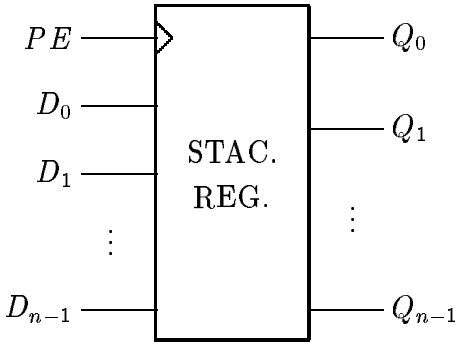
Slika 1 Logička šema stacionarnog registra ($n = 8$)



Slika 2 Standardni simbol za stacionarni registar sa slike 1



Slika 3 Standardni simbol za stacionarni registar



Slika 4 Standardni simbol za stacionarni registar

37. POMERAČKI REGISTRI

Pomerački registar ili šift registar (engl. Shift Register) je registar u kome se zapamćena informacija pomijera za jedno mjesto; pomijera se ako taktni impuls. U pomeračkim registrima mogu se koristiti D ili JK flipflopovi ivičnog ili MS tipa. Na slici 1 prikazana je logička šema pomeračkog registra čija je veličina dva bita, a koji ima još i mogućnost paralelnog upisivanja. Realizovan je pomoću ivičnih D flipflopova. Pogledajmo šta može u registru da se dešava. Informacija se upisuje paralelno u prikazani pomerački registar, preko asinhronih ulaza S_D i R_D, na način kako je to objašnjeno za stacionarne registre. Tako da smo vidjeli jednu funkciju razmatranog registra (Load, upis), a sada ćemo vidjeti i drugu njegovu funkciju (Shift, pomijeranje). Ako su ulazni signali A i B bili u stanju $A = 1$ i $B = 0$ za vrijeme dok je PE bio aktivan onda se u flipflopove Q_A i Q_B upisalo redom 1 i 0. Prostije rečeno, sada je $Q_A = 1$ i $Q_B = 0$. Pogledajmo šta se dešava tokom prve periode vremena (tokom jedne periode takta). Za vrijeme aktivne (uzlazne) ivice prvog CLK impulsa ulaz D_A je na logičkom nivou SI, a D_B je na logičkom nivou Q_A. Nakon vremena propagacije kroz flipflopove postaće $Q_A = 0$ i $Q_B = 1$; pretpostavili smo da je $SI = 0$. Tako da se sadržaj flipflopa A pomjerio u flipflop B. Možemo pogledati i šta će se onda dalje desiti tokom druge periode vremena. Sljedeća aktivna ivica taktnog impulsa ponovo pomjeri sadržaj u registru, tako da onda u registru ostaju nule na oba mjesta. SI je skraćenica za Serial–in, u prevodu: serijski ulaz ili redni ulaz. Prema tome, za registar sa slike 1 važe sljedeće dvije formule:

$$PE\colon Q_A \leftarrow A,\ Q_B \leftarrow B \ \text{ i } \ CLK\colon Q_B \leftarrow Q_A,\ Q_A \leftarrow SI.$$

U slučaju kada se za pomerački registar koriste JK flipflopovi, Q izlaz prethodnog flipflopa vezuje se na J ulaz narednog, a $\overline{Q}$ izlaz prethodnog na K ulaz narednog. Paralelni upis može se obavljati preko asinhronih ulaza S_D i R_D na isti način kao što je to prikazano na slici 1.

Postoje brojne realizacije integrisanih pomeračkih registara. Uglavnom se koriste ivični D flipflopovi. Paralelni upis podataka može se obavljati na način kako je to prikazano na slici 1 (asinhrono). Međutim, većina registara koristi taktovani, odnosno sinhroni paralelni upis, kao što bi se vidjelo sa šeme registra čiji je simbol dat na slici 2 a).

Na slici 2 a) prikazan je simbol pomeračkog registra koji ima i mogućnost paralelnog upisa, a pored toga još ima i mogućnost paralelnog čitanja. Kao što bi se vidjelo sa šeme, kada je signal L/S u stanju $L/S = 1$ onda se na D ulazima flipflopova nalazi prisutan logički nivo ulazne informacije DAT_i. Tako da se onda (generisanjem takta) u flipflopove paralelno upisuje informacija sa DAT_i ulaza. Ako je međutim $L/S = 0$ onda je na D ulazima flipflopova prisutan logički nivo Q izlaza prethodnog flipflopa, tako da se (generisanjem takta) informacija upisana u registru pomijera za jedno mjesto udesno, a u Q_0 se (aktivnom ivicom takta) upisuje logički nivo ulazne promjenljive SIN. Flipflopovi se prebacuju silaznom ivicom takta CLK (). L/S znači Load/Shift ili u prevodu unesi ili pomjeri. Vrijednost signala L/S razvodi se na n dvoulaznih multipleksera (realizovanih pomoću NI kola), čime se i postižu dvije različite funkcije, jedna kada je $L/S = 1$ i druga kada je $L/S = 0$.

Registar može da se resetuje asinhrono pomoću signala CLR, čiji je aktivni nivo nula. Na izlaznim linijama registra prisutna su trostatička kola koja se aktiviraju pomoću signala OE.

U sljedećoj tabeli prikazana su stanja pomeračkog registra koji se razmatra, u zavisnosti od ulaznih promjenljivih. Neka bude $n = 8$. U tabeli se koriste sljedeće oznake. Neka $Q_0, Q_1, \ldots, Q_7$ znače stare vrijednosti izlaznih promjenljivih. Neka $Q'_0, Q'_1, \ldots, Q'_7$ znače nove

vrijednosti izlaza. U tabeli se pomoću slova Z označava da su izlazi registra u stanju visoke impedanse.

Tabela: Stanja pomeračkog registra sa slike 2 a)

L/S	CLR	OE	Q_0'	Q_1'	Q_2'	...	Q_7'
X	X	0	Z	Z	Z		Z
X	0	1	0	0	0		0
1	1	1	DAT_0	DAT_1	DAT_2		DAT_7
0	1	1	SIN	Q_0	Q_1		Q_6

U posljednjem redu tabele može se dodati da je $SOUT = Q_7$. $SOUT$ (Serial–out) je serijski izlazni priključak.

Pomerački registar čiji je simbol prikazan na slici 2 a) proizvodi se kao integrisana komponenta sa četiri ili osam flipflopova.

Integrisani pomerački registri nemaju uvijek sve mogućnosti koje sadrži registar sa slike 2 a). Postoje registri bez mogućnosti reseta ili sa izlazima koji nisu trostatički. Takođe postoje integrisani pomerački registri kod kojih izlazi flipflopova nisu svi izvedeni na priključke, već je izveden samo izlaz posljednjeg flipflopa. Takav registar naziva se registar sa paralelnim ulazom i serijskim izlazom (Parallel–in, serial–out), a njegov grafički simbol dat je na slici 2 b).

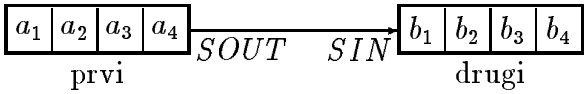
Slika 2 c) odgovara pomeračkom registru sa serijskim ulazom i paralelnim izlazom. U takav registar informacija može da se upiše samo serijski, ne sadrži multipleksere na D ulazima flipflopova. Registar ima mogućnost reseta, a izlazi nisu trostatički.

Govoreći uprošćeno, za registar sa slike 2 c) važe sljedeće okolnosti. Ako je njegovo staro stanje bilo $q_0 q_1 \ldots q_7$ onda će novo stanje biti $q_0' q_1' \ldots q_7' = 0 q_0 \ldots q_6$, misli se: ako je $SIN = 0$. Recimo, ako je staro stanje bilo 11111000 onda će novo stanje biti 01111100. Kao da je binarni broj podijeljen sa dva.

Slika 2 d) odgovara pomeračkom registru sa serijskim ulazom i serijskim izlazom, bez mogućnosti reseta.

Ako se izlazni priključak $SOUT$ jednog pomeračkog registra (čija je veličina n bita) veže na ulazni priključak SIN drugog (iste veličine) onda će sadržaj prvog da bude upisan u drugi, kada prođe vrijeme od n taktova. Zato što pomeračkom registru odgovara jednačina:

$$CLK\colon\ SOUT \leftarrow Q_7,\ Q_7 \leftarrow Q_6,\ Q_6 \leftarrow Q_5,\ \ldots,\ Q_1 \leftarrow Q_0,\ Q_0 \leftarrow SIN$$

$t = 0$ 

$t = 1$	$0\, a_1\, a_2\, a_3$	$a_4\, b_1\, b_2\, b_3$
$t = 2$	$0\, 0\, a_1\, a_2$	$a_3\, a_4\, b_1\, b_2$
$t = 3$	$0\, 0\, 0\, a_1$	$a_2\, a_3\, a_4\, b_1$
$t = 4$	$0\, 0\, 0\, 0$	$a_1\, a_2\, a_3\, a_4$

Postoje i brojne druge vrste pomeračkih registara:

1. Bidirekcioni pomerački registar. Kod ovakvog registra, upisana informacija pomijera se udesno ili ulijevo, zavisno od logičkog nivoa kontrolnog signala L/R (Left/Right – lijevo ili desno).

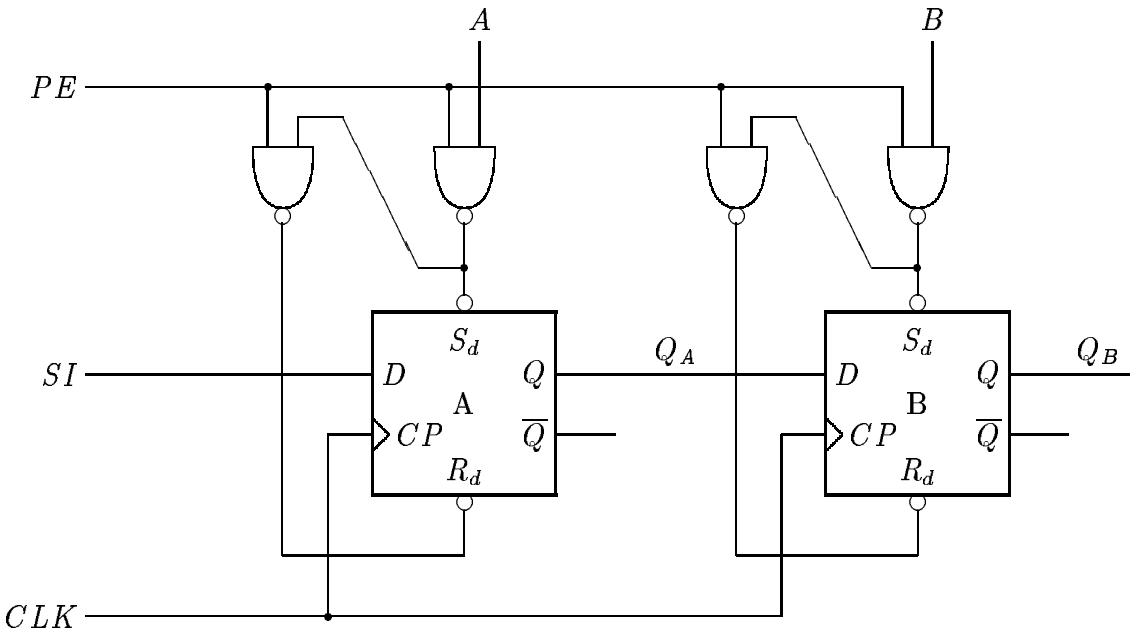
2. Kružni brojač (Ring counter). Ako je staro stanje osmobitnog kružnog brojača bilo

a_1	...	a_8

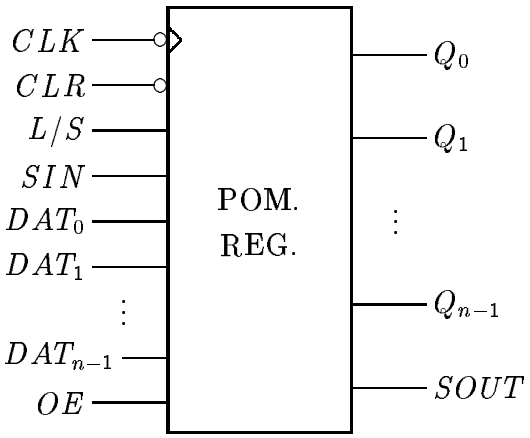
onda će njegovo novo stanje (poslije jednog takta) biti

a_8	a_1	a_2	a_3	a_4	a_5	a_6	a_7

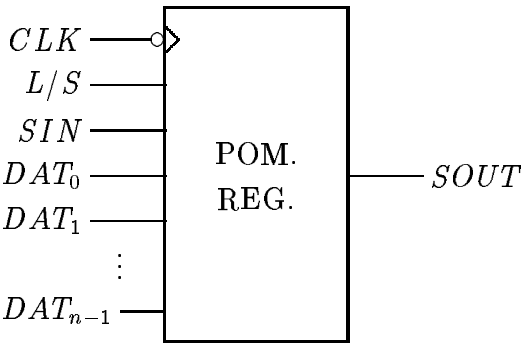
. Često se u kružni brojač kao njegov inicijalni sadržaj upiše sedam nula i samo na jednom mjestu jedna jedinica.



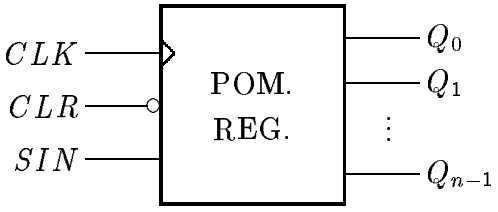
Slika 1 Logička šema dvo–bitnog pomeračkog registra



Slika 2 a)



Slika 2 b)



Slika 2 c)

Slika 2 d)

Slika 2 Simboli integrisanih pomeračkih registara

38. BROJAČI

Govoreći uprošćeno, brojač se može opisati na sljedeći način. Uzmimo da brojač ima $n = 8$ bita. Neka je njegovo prethodno stanje označeno kao $q = q_7 q_6 \ldots q_0$, a novo stanje kao $q' = q'_7 q'_6 \ldots q'_0$. Tada važi jednakost $q' = q + 1$. Na primjer, ako je staro stanje bilo $q = 1000\,0011$ onda će sadašnje stanje biti $q' = 1000\,0100$.

Brojač je sekvencijalna mreža kod koje se upisana vrijednost iz takta u takt povećava za jedan. Upisane vrijednosti tokom vremena obrazuju ciklus koji se ponavlja. Broj različitih stanja u ciklusu naziva se osnovom brojača (modulom brojača).

Kao memorijski elementi u brojačima se koriste flipflopovi. Razmotrimo brojač koji se sastoji od n flipflopova. Uzmimo da brojač ima $m = 2^n$ stanja i da se stanja smjenjuju u sekvenci binarnih brojeva. Tada se kaže da je to jedan n–bitni binarni brojač ili da je to binarni brojač čiji je moduo brojanja jednak $m = 2^n$.

Ako se svi flipflopovi u brojaču taktuju zajedničkim taktnim impulsom onda se brojač naziva sinhronim. Ako taktni impuls nije zajednički za sve flipflopove, brojač je asinhroni.

Kada se na prvi u lancu ivičnih flipflopova vrste T prikazanih na slici 1 priključi povorka taktnih impulsa CLK onda će se na izlazima flipflopova dobiti talasni oblici prikazani na slici 2, umjesto $\ldots 010 \ldots$ nacrtaj $\ldots \_/\overline{\ }\backslash\_ \ldots$ i slično. Vidimo da na slici 1 učestvuju T flipflopovi koji se prebacuju silaznom ivicom impulsa priključenog na T ulaz i da se svaki naredni flipflop prebacuje silaznom ivicom izlaznog signala prethodnog flipflopa. Prebacivanje se može obavljati kada su R_D ulazi na neaktivnom logičkom nivou. Razmatrani brojač može da bude postavljen u stanje $Q_3 Q_2 Q_1 Q_0 = 0000$ dovođenjem signala za reset CLR na aktivni logički nivo (na niski nivo).

Sa slike 2 vidi se da stanja četiri flipflopa Q_0 do Q_3 u vremenskim trenucima označenim kao $k = 0$, ..., $k = 15$ kodiraju binarni broj koji izražava broj taktnih impulsa koji su do posmatranog trenutka djelovali na ulaz flipflopa Q_0. Prema tome, mreža prikazana na slici 1 obavlja funkciju brojanja taktnih impulsa, gdje se broj prikazuje u prirodnom binarnom kodu. Flipflopovi u mreži ne prebacuju se zajedničkim taktnim impulsom. Za takvu mrežu kaže se da predstavlja asinhroni binarni brojač. Vidimo da se prebacivanje flipflopova izvodi serijski. Zato se takav brojač često naziva serijskim brojačem (rednim brojačem), a u engleskoj literaturi ripple counter, što se prevodi kao brojač sa prenošenjem talasa.

Ista mreža može da bude realizovana i korišćenjem JK flipflopova. J i K ulazi svih flipflopova u toj mreži vezani su za logički nivo 1. Q izlaz svakog flipflopa u lancu priključen je na taktni ulaz narednog flipflopa (isto kao na slici 1).

Asinhroni brojači imaju ograničenu primjenu upravo zbog asinhronog načina prebacivanja flipflopova u lancu. Uzimajući u obzir vrijeme propagacije kroz pojedini flipflop i uzimajući u obzir da se svaki naredni flipflop prebacuje tek nakon isteka vremena propagacije prethodnog, onda tokom trajanja prelaznih stanja brojač kodira pogrešan sadržaj. Tako da navedeni nedostatak asinhronog brojača dovodi do pojave neregularnih stanja. Pored toga, taj nedostatak ograničava maksimalnu učestanost takta na kojoj brojač može ispravno da radi. Naime, sljedeća aktivna ivica taktnog signala ne smije da se pojavi prije nego što se završi prethodno prelazno stanje. Asinhroni brojači upotrebljavaju se uglavnom kao djelitelji učestanosti. Oni se koriste u mrežama gdje nije potrebno dekodiranje stanja brojača (čitanje upisane vrijednosti). Već se koristi jedino izlaz posljednjeg flipflopa u lancu, to je $Q_{n-1} = Q_3$ na prethodnoj slici; koristi se kao signal čija je učestanost 2^n puta niža od učestanosti ulaznog takta.

Postoje razne vrste sekvencijalnih mreža za koje se kaže da predstavljaju brojače:

1. Asinhroni brojači. O njima je bilo govora maločas.

2. Sinhroni brojači. Za razliku od asinhronih brojača, kod sinhronih brojača svi flipflopovi prebacuju se sinhrono zajedničkim taktnim impulsom. Uslov brojanja može da se prenosi serijski ili paralelno.

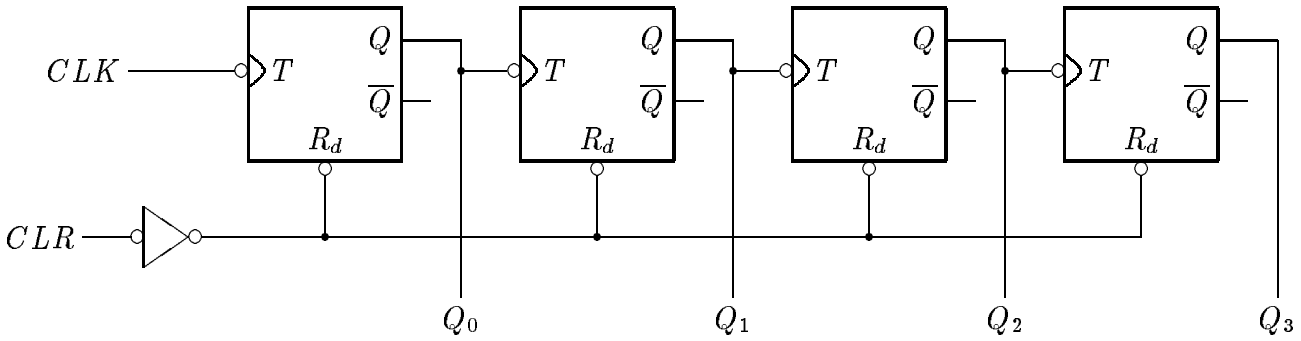
3. Brojači unazad.

4. Obostrani brojači.

5. Brojač sa mogućnošću paralelnog upisa sastavlja se od flipflopova koji posjeduju asinhrone S_D i R_D ulaze. Prije početka brojanja, upiše se inicijalni sadržaj.

6. Brojač čiji je moduo brojanja jednak $m = 10$ može da bude sastavljen od 4 flipflopa.

Zaključak o brojačima. Po liniji *count* šalje se uslov brojanja, tako da važi relacija *count*: $q \leftarrow q + 1$. Po liniji *reset* ostvaruje se svođenje na nulu, tako da važi *reset*: $q \leftarrow 0$ ili riječima ako *reset* onda q dobija vrijednost nula. Ovdje je $q = (q_7 q_6 \ldots q_0)_2$, v. ilustraciju.

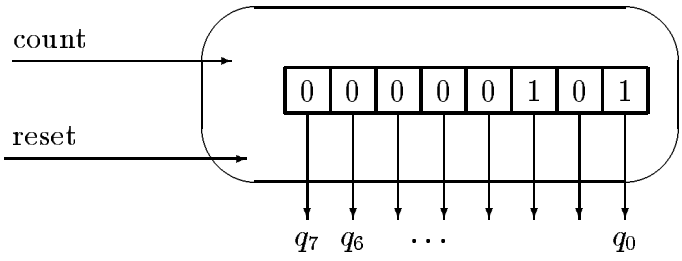


Slika 1 Kaskadna veza T flipflopova

Nakon k-te silazne ivice signala CLK imamo (za $k = 0$, $k = 1$, ...):

$k =$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	...
$Q_0 =$	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
$Q_1 =$	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	
$Q_2 =$	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	
$Q_3 =$	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	

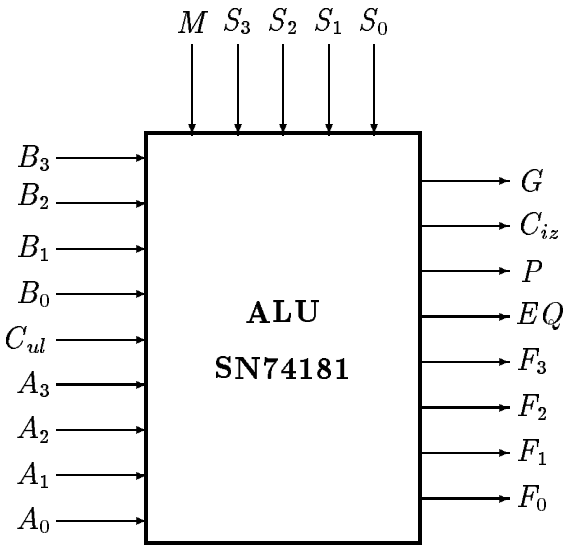
Slika 2 Vremenski dijagrami mreže brojača sa slike 1. U tabeli su prikazane vrijednosti Q_0, Q_1, Q_2, Q_3. Za T flipflop koji se prebacuje na opadajuću ivicu važi: na ulazu T je opadajuća ivica, tj. vrijednost signala spušta se sa visokog nivoa na niski nivo $\Rightarrow$ izlaz Q se komplementira



Brojač. Trenutni sadržaj brojača je
$q = (q_7 q_6 \ldots q_0)_2 = (00000101)_2 = 5$

39. PRIMJER ARITMETIČKO–LOGIČKE JEDINICE – ALU SN74181

Aritmetička kola u digitalnim sistemima obavljaju osnovne aritmetičke operacije: sabiranje, oduzimanje, množenje i dijeljenje. U klasu aritmetičkih kola spadaju i mreže za komplementiranje brojeva, mreže za poređenje brojeva i aritmetičko–logičke jedinice (ALU) koje mogu da obavljaju aritmetičke i logičke operacije. Osnovna aritmetička kola spadaju u klasu kombinacionih mreža. Međutim, aritmetička kola se najčešće koriste u mrežama koje pored kombinacionih elemenata koriste i sekvencijalne (memorijske) elemente, kako bi se pamtili rezultati složenijih aritmetičkih operacija. Kao integrisane komponente, aritmetička kola se izrađuju za operacije sa brojevima izraženim u binarnom brojnom sistemu i (rijetko) sa brojevima izraženim u BCD kodu.



Slika: Logička šema ALU SN74181

Aritmetičko–logičke jedinice, engl. arithmetic–logic unit, ALU, su višefunkcionalne kombinacione mreže koje (zavisno od kontrolnih ulaza) mogu da obavljaju različite aritmetičke ili logičke operacije nad dva broja od po n cifara. Tipične integrisane aritmetičko–logičke jedinice izrađuju se za operacije nad četvorobitnim brojevima i imaju do 5 kontrolnih ulaza, tako da mogu da obavljaju do $2^5 = 32$ različitih funkcija.

Razmotrimo jedan primjer četvorobitne ALU. Logički simbol komponente SN74181 prikazan je na slici. Komponenta je sastavljena od velikog broja logičkih elemenata i izostavlja se detaljna priča o njenoj unutrašnjoj strukturi. Pogledajmo smisao spoljašnjih priključaka komponente. Kontrolnim ulazom M bira se skup logičkih operacija ($M = 1$) ili skup aritmetičkih operacija ($M = 0$). Selekcionim signalima S_3, S_2, S_1 i S_0 bira se određena funkcija iz skupa funkcija pridruženog datoj vrijednosti M. Vidimo da ukupno ima 5 kontrolnih pinova. Pogledajmo sada linije za podatke. Po ulazima A_3 do A_0 i B_3 do B_0 upućuju se dva broja nad kojima će biti izvedena operacija, a rezultat operacije će biti saopšten preko izlaznih linija F_3 do F_0. U slučaju aritmetičkih funkcija, koriste se i signali C_{ul} i C_{iz} – ulazni i izlazni signal prenosa.

U tabeli su prikazane sve logičke i aritmetičke funkcije koje komponenta obavlja, zavisno od kombinacije kontrolnih promjenljivih. Ponešto o oznakama. $A = A_3A_2A_1A_0$. $F = \overline{A}$ ima smisao $F_i \leftarrow \overline{A_i}$ za $i = 3, 2, 1, 0$. $F = A \,\&\, B$ ima smisao $F_i \leftarrow A_i \,\&\, B_i$ za $i = 3, 2, 1, 0$. I slično.

Pogledajmo mali primjer. Pogledajmo deseti red tabele, gdje je $S_3S_2S_1S_0 = 1001$, i to kada je $M = 0$ (slučaj aritnetičke funkcije). U slučaju te kombinacije upravljačkih signala, razmatrana komponenta obavlja funkciju sabiranja dva četvorobitna broja, uz uračunavanje u zbir i prenosa C_{ul} i uz još i saopštavanje prenosa C_{iz} koji nastaje na mjestu najveće težine (na bitu broj 3). Izraženo formulom: $F \leftarrow A + B + C_{ul}$ i $C_{iz} \leftarrow 1$ ako je došlo do prekoračenja prilikom sabiranja. Navedimo mali primjer sa konkretnim brojevima: ako se postave ulazi $A_3A_2A_1A_0 = 0011$, $B_3B_2B_1B_0 = 0011$ i $C_{ul} = 1$ onda će se izlazne vrijednosti obrazovati kako slijedi: $F_3F_2F_1F_0 = 0111$ i $C_{iz} = 0$. Prevedeno u dekadni oblik: $3 + 3 + 1 = 7$. Vidimo da nema prenosa (prekoračenja), jer je saopšteno $C_{iz} = 0$. Važi ekvivalencija: $C_{iz} = 1$ (prenos se generiše) $\Leftrightarrow A + B + C_{ul} \geq 16$.

Za izlaznu liniju EQ važi: $EQ = 1$ ako i samo ako $A_0 = B_0$, $A_1 = B_1$, $A_2 = B_2$ i $A_3 = B_3$.

O izlaznim linijama G i P biće riječi kasnije. Treba reći da komponenta ima još dva pina (još dva izvoda, još dvije nožice) koji nisu prikazani na slici. To su V_{CC} preko koga se uvodi napon napajanja i GND – uzemljenje. Tako da razmatrana komponenta ukupno ima tačno 24 pina.

Tabela: Funkcije ALU SN74181

Selekcija				Logičke funkcije	Aritmetičke funkcije
S_3	S_2	S_1	S_0	$M=1$	$M=0$
0	0	0	0	$F=\overline{A}$	$F=A+C_{ul}$
0	0	0	1	$F=\overline{A\vee B}$	$F=(A\vee B)+C_{ul}$
0	0	1	0	$F=\overline{A}\,\&\,B$	$F=(A\vee\overline{B})+C_{ul}$
0	0	1	1	$F=0000$	$F=-\overline{C_{ul}}$
0	1	0	0	$F=\overline{A\,\&\,B}$	$F=A+(A\,\&\,\overline{B})+C_{ul}$
0	1	0	1	$F=\overline{B}$	$F=(A\vee B)+(A\,\&\,\overline{B})+C_{ul}$
0	1	1	0	$F=A\oplus B$	$F=A-B-\overline{C_{ul}}$
0	1	1	1	$F=A\,\&\,\overline{B}$	$F=(A\,\&\,\overline{B})-\overline{C_{ul}}$
1	0	0	0	$F=\overline{A}\vee B$	$F=A+(A\,\&\,B)+C_{ul}$
1	0	0	1	$F=\overline{A\oplus B}$	$F=A+B+C_{ul}$
1	0	1	0	$F=B$	$F=(A\vee\overline{B})+(A\,\&\,B)+C_{ul}$
1	0	1	1	$F=A\,\&\,B$	$F=(A\,\&\,B)-\overline{C_{ul}}$
1	1	0	0	$F=1111$	$F=A+A+C_{ul}$
1	1	0	1	$F=A\vee\overline{B}$	$F=(A\vee B)+A+C_{ul}$
1	1	1	0	$F=A\vee B$	$F=(A\vee\overline{B})+A+C_{ul}$
1	1	1	1	$F=A$	$F=A-\overline{C_{ul}}$

Mreža za izvođenje aritmetičkih i logičkih operacija nad dva broja od po 16 bita dobija se jednostavnim kaskadnim povezivanjem četiri razmatrane komponente $K(1)$, $K(2)$, $K(3)$ i $K(4)$. Cifre brojeva čija je težina slaba uvode se u $K(1)$, itd. a četiri najjače cifre jednog i drugog broja uvode se u $K(4)$. Izlaz C_{iz} iz $K(1)$ sprovodi se na ulaz C_{ul} komponente $K(2)$ i slično $C_{ul}(3)\leftarrow C_{iz}(2)$ i $C_{ul}(4)\leftarrow C_{iz}(3)$.

Još samo o jednom načinu da se aritmetička operacija (primjera radi sabiranje) nad dva 16–bitna broja ubrza (obavi za kraće vrijeme). Vidjećemo da taj način predviđa izvjesnu hardversku nadogradnju. Upravo, lako se zapaža da ima li ili nema prenosa za jednu komponentu $K(j)$ može da se izračuna brže nego rezultat $F=F_3F_2F_1F_0$ (rezultat može da bude recimo zbir). Izlazi G i P služe da omoguće ubrzano ili unaprijed saznavanje i predaju prenosa, engl. fast look–ahead carry, prilikom kaskadnog vezivanja. Definišimo G i P, mada nećemo sastavljati njihove izraze. $G=1$ kada je $A+B\geq 16$ (prenos se generiše). $P=1$ kada je $A+B\geq 15$ (prenos se generiše ili ne generiše zavisno od C_{ul}). Recimo, $A_3=1$ tj. $A\geq 8$ i $B_3=1$ tj. $B\geq 8$ slijedi $A+B\geq 16$, i slično.

Dakle, prilikom kaskadnog vezivanja četiri razmatrane komponente SN74181 u praksi se po pravilu doda još i jedna komponenta SN74182, u čemu se i ogleda hardverska nadogradnja. Time se smanji ukupno kašnjenje.

Pomoćna mreža SN74182 je prostija od mreže SN74181. Govoreći uprošćeno, ulazi u mrežu SN74182 su $C_{ul}(1)$, $G(1)$, $P(1)$, $G(2)$, $P(2)$, $G(3)$ i $P(3)$, a njeni izlazi sprovode se na $C_{ul}(2)$, $C_{ul}(3)$ i $C_{ul}(4)$. Oznake: $C_{ul}(1)$ je C_{ul} prve mreže SN74181, tj. mreže $K(1)$ i slično.

40. RAM MEMORIJE

1 Pojam memorije

Memorije u digitalnim sistemima predstavljaju sklopove u koje se može upisati i iz kojih se može pročitati informacija. Zavisno od medijuma na kome se informacija pamti, najčešće se koriste poluprovodničke, magnetne i optičke memorije. Magnetne i optičke memorije se uglavnom koriste za memorisanje velikog broja digitalnih informacija. Vrijeme upisivanja i čitanja informacija je relativno dugačko, zbog neophodnih mehaničkih pomijeranja diska ili trake. Informacija u ovim memorijama ostaje zapamćena i kada je isključeno električno napajanje, tako da ove memorije spadaju u klasu postojanih memorija, engl. nonvolatile memory.

Poluprovodnička memorija u koju se može upisati ili pročitati informacija u proizvoljnom trenutku naziva se RAM memorijom, za razliku od ROM memorije kod koje je fizički i vremenski proces upisivanja različit od procesa čitanja sadržaja. Naziv RAM dolazi od engleskog naziva random access memory – memorija sa slučajnim pristupom, što na neki način znači da je vrijeme za upisivanje ili čitanje nezavisno od adrese sa koje se čitanje ili upisivanje obavlja. Poluprovodničke RAM memorije po pravilu gube sadržaj kada se isključi napon napajanja, tako da spadaju u klasu nepostojanih memorija, engl. volatile memory. Samo se napominje da su se do početka sedamdesetih godina kao RAM memorije koristile i memorije od magnetnih jezgara. Danas se magnetne RAM memorije više ne koriste, zbog velike potrošnje i visoke cijene. Magnetne memorije su postojane memorije.

Poluprovodničke memorije mogu biti statičke i dinamičke. Informacija upisana u statičkoj memoriji ostaje zapamćena sve dok je memorija priključena na napon napajanja. Da bi informacija ostala zapamćena u dinamičkoj memoriji neophodno je periodično obavljati "osvježavanje" memorije, inače se informacija gubi.

2 Statičke poluprovodničke memorije

Statička RAM memorija (SRAM) predstavlja skup stacionarnih registara sa zajedničkim ulaznim i izlaznim priključcima. Selekcija registra u koji će se informacija upisati ili iz koga će se informacija pročitati obavlja se adresnim dekoderom. Logička šema statičke RAM memorije sa jednodimenzionim dekodiranjem prikazana je na slici 1.

Na slici, pojedini stacionarni registar sastoji se od m flipflopova vrste D sa ivičnim prebacivanjem. DIN znači data in, a $DOUT$ znači data out. DIN_0 do DIN_{m-1} su ulazne linije podataka, a $DOUT_0$ do $DOUT_{m-1}$ su izlazne linije podataka.

U svaki od 2^n stacionarnih registara može se upisati po jedna digitalna riječ od m bita. Adresnim ulazima $A_0, A_1, \ldots, A_{n-1}$ kodira se lokacija k–tog memorijskog registra. Time se k–ta izlazna linija dekodera (k–ta adresna linija) postavlja na logički nivo jedan. Postavljanjem $CS = 1$ selektuje se taj memorijski čip i onda se dovođenjem impulsa na kontrolni ulaz WE (write enable) u selektovani registar upisuje sadržaj prisutan na ulaznim linijama podataka $DIN_0, DIN_1, \ldots, DIN_{m-1}$. Sadržaj ostaje upisan (memorisan) sve dok se istim postupkom ne promijeni ili dok se ne isključi napon napajanja.

Čitanje upisane digitalne informacije obavlja se adresiranjem k–tog registra, čime se izlazne linije selektovanog registra priključuju na izlazne linije podataka. Postavljanjem $CS = 1$ i $OE = 1$, podaci upisani u k–tom memorijskom registru postaju pristupačni na izlaznim priključcima $DOUT_0, DOUT_1, \ldots, DOUT_{m-1}$.

U cilju smanjenja broja dekoderskih kola i u cilju formiranja kvadratne matrice memorijskih ćelija, RAM memorija se po pravilu izrađuje sa dvodimenzionim dekodiranjem. O ovome će biti više riječi kasnije (ne u ovom naslovu). Govoreći uprošćeno, kod jednodimenzionog dekodiranja,

u jednoj vrsti memorijske matrice nalazi se jedan registar, tako da je m dužina vrste, a 2^n je broj vrsta. A kod dvodimenzionog dekodiranja, u jednoj vrsti nalazi se više registara. Napominje se da iste okolnosti važe i za dekodiranje u slučaju ROM memorije.

Statičke memorije se u CMOS tehnologiji izrađuju sa jednim, četiri ili osam bita podataka ($nK \times 1$, $nK \times 4$ ili $nK \times 8$). Obično se prave sa kapacitetom od $64K \times 8$, a vrijeme pristupa se kreće u granicama od 12ns do 150ns (zavisno od tipa). Bipolarne memorije su po pravilu brže, ali znatno manjeg kapaciteta, a najbrže se izrađuju u ECL tehnologiji gdje vrijeme pristupa može biti i manje od 10ns.

3 Dinamičke poluprovodničke memorije

Svaka memorijska ćelija u statičkoj RAM memoriji sastoji se od najmanje četiri do šest tranzistora. Da bi se realizovala memorija sa većom gustinom pakovanja, konstruisana je memorija sa samo jednim tranzistorom i jednim kondenzatorom po memorijskoj ćeliji. Ovakva memorija bazira pamćenje informacije na električnom punjenju kondenzatora.

Da bi se postigla velika gustina pakovanja, kondenzator u memorijskoj ćeliji je veoma malih dimenzija, pa je i kapacitivnost kondenzatora veoma mala. Kada bi otpornost MOS tranzistora (za vrijeme dok je neprovodan) bila beskonačno velika onda bi napon na kondenzatoru ostao nepromijenjen. Zbog konačne otpornosti neprovodnog MOS tranzistora, a i zbog male kapacitivnosti kondenzatora, zapamćeni napon na kondenzatoru (kada je zapamćena logička jedinica) postepeno opada i nakon nekoliko ms zapamćena informacija bi se izgubila. Da se ovo ne bi dogodilo, svakih 2 do 4ms treba ponovo upisivati informaciju u memorijsku ćeliju. Repetitivni upis (ponovni upis) naziva se osvježavanjem sadržaja, a RAM memorija koja sadrži ćelije kojima je neophodno periodično osvježavanje naziva se dinamičkom RAM memorijom ili skraćeno DRAM.

Savremene dinamičke memorije su kapaciteta recimo $64K \times 1$ ili $1M \times 4$ (ili $4M \times 1$). U okviru integrisanog kola nalazi se i kontroler osvježavanja, tako da korisnik ne mora da vodi računa o redosljedu generisanja adresa za osvježavanje (za razliku od starijih tipova dinamičkih memorija). Da bi memorija bila raspoloživa za upisivanje i čitanje sadržaja, proces osvježavanja memorije treba da se obavi u što kraćem vremenu. U tom cilju, dinamičke memorije uvijek koriste dvodimenziono dekodiranje, a osvježavanje se obavlja istovremeno u svim ćelijama u jednoj vrsti. Takođe, u cilju smanjenja cijene memorijskog čipa, da bi broj spoljnih priključaka bio što manji, memorije se izrađuju sa zajedničkim (preklopljenim) adresnim ulazima za selekciju vrste i selekciju kolone. Po tim ulazima prvo se memoriji pošalje adresa vrste, a nakon toga se pošalje i adresa kolone.

4 Primjena RAM memorija

Poluprovodničke RAM memorije su sastavni dio svakog računarskog sistema. Zavisno od veličine računara, potrebni kapacitet RAM memorije kreće se od nekoliko desetina KB (kilo–bajta) za specijalizovane mikro–računarske sisteme pa do više GB za velike super–računare. Kapacitet RAM memorije personalnih računara je najčešće u granicama od 1 do 8MB, pa i više. Ako se integrisana memorijska kola komercijalno izrađuju do kapaciteta 64KB ($64K \times 8$) kada su u pitanju statičke memorije, odnosno do 4Mb (četiri mega–bita) ($4M \times 1$) kada su memorije dinamičke onda je za realizaciju RAM memorije računarskog sistema neophodno koristiti više čipova.

Statičke memorije su manjeg kapaciteta po čipu, a koriste se u sistemima gdje se zahtijeva veća brzina pristupa memoriji i manja potrošnja struje iz izvora za napajanje. Takođe je vjerovatnoća greške kod statičkih memorija manja nego kod dinamičkih, tako da se koriste u sistemima gdje se zahtijeva visoka pouzdanost.

Dinamičke memorije, zbog daleko veće gustine pakovanja, zahtijevaju manji broj integrisanih kola nego statičke, za isti kapacitet. Potrošnja struje dinamičkih memorija je znatno veća nego statičkih, a prilikom svake selekcije vrste struja napajanja se impulsno poveća za nekoliko desetina mA po čipu. Ove impulsne promjene struje mogu da generišu električne smetnje, tako da prilikom korišćenja DRAM memorija treba posebno voditi računa o filtraciji napona napajanja. Takođe, dinamička memorija zahtijeva periodični impuls za osvježavanje svakih nekoliko ms ako je kontroler osvježavanja ugrađen u čipu, odnosno za starije ali još uvijek aktuelne DRAM komponente koje nemaju ugrađen kontroler za osvježavanje kontroler se mora realizovati MSI kolima. I pored ovih nedostataka, kada je potreban veliki kapacitet, DRAM memorije se češće koriste, s obzirom da imaju veću gustinu pakovanja a time zauzimaju manje prostora na štampanoj ploči. Zbog manjeg broja čipova i jednostavnije štampane ploče, cijena DRAM memorije je niža od SRAM memorije istog kapaciteta.

5 Organizacija statičke memorije većeg kapaciteta

U računarskim sistemima se razmjena informacija sa memorijom uglavnom obavlja preko sistemske magistrale koja sadrži skup adresnih linija, skup linija podataka i skup kontrolnih linija. Na slici 2 prikazana je logička šema kontrolne mreže memorijskog čipa; slika 2: logička šema kontrolne mreže statičke RAM memorije. Prisutni su signali WE (engl. write enable) i OE (engl. output enable). Obezbijeđen je i ulaz za selekciju čipa CS (engl. chip select). Treba reći da su u praksi po pravilu prisutna dva ulaza za selekciju čipa, od kojih je jedan aktivan na visokom logičkom nivou a drugi na nultom nivou; čip je selektovan ako su oba aktivna. Isto tako, treba reći da su u praksi dva signala OE i WE najčešće aktivni na niskom logičkom nivou, zato što su na magistralama kontrolni signali po pravilu aktivni na niskom nivou.

Pošto se upis u memoriju i čitanje iz memorije nikad ne obavlja istovremeno, to su izrađeni zajednički bidirekcioni priključci za podatke.

Na slici 3 prikazan je simbol statičke RAM memorije oblka $8K \times 8$.

A znači address, DIO znači data input output. Vidimo da je $2^n = 2^{13} = 8K$ i da je $m = 8$. WE: $(DIO_0, \ldots, DIO_{m-1}) \rightarrow$ stac. reg. broj k. OE: stac. reg. broj $k \rightarrow (DIO_0, \ldots, DIO_{m-1})$. $0 \leq k \leq 2^n - 1$, broj k definiše se od strane $A_0, A_1, \ldots, A_{n-1}$.

Vidimo da memorija ima $2^{13} \times 2^3 = 2^{16}$ bita. Ako se vrši dvodimenziono dekodiranje, tj. ako se želi napraviti kvadratna matrica memorijskih ćelija onda očito treba da bude 2^8 vrsta, po 2^8 ćelija u jednoj vrsti.

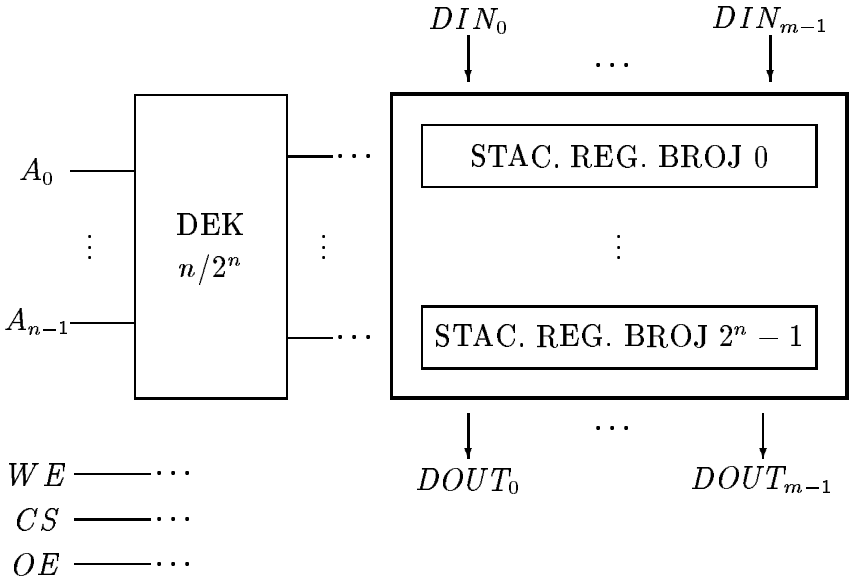
Ako je potrebno na primjer da se realizuje memorija od 64KB ($64K \times 8$) a na raspolaganju su memorijski čipovi kapaciteta $8K \times 8$ onda je za formiranje takve memorije potrebno 8 čipova.

6 Organizacija dinamičke memorije većeg kapaciteta

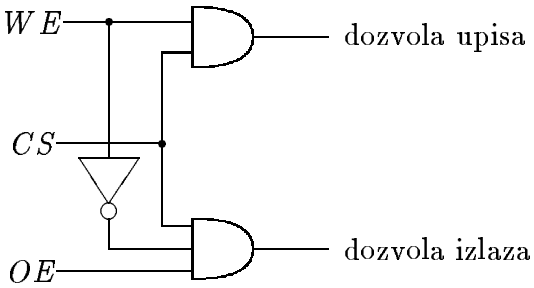
(...)

7 Memorijske komponente za specifične primjene

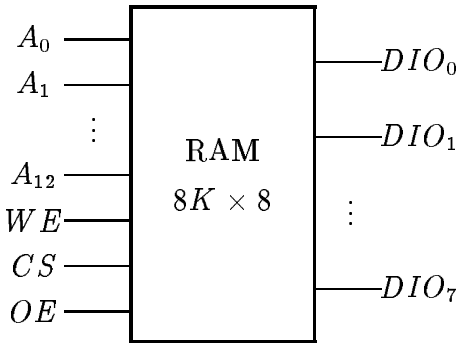
Baterijski podržana memorija ili RAM memorija sa rezervnim baterijskim napajanjem ili engl. batery backup RAM. Poluprovodničke RAM memorije spadaju u klasu nepostojanih memorija, s obzirom da gube sadržaj prilikom nestanka napona napajanja. Jedan od načina da se sačuvaju podaci u RAM memoriji i nakon nestanka napona napajanja je priključivanje suve, odnosno akunulatorske baterije koja će da napaja memoriju i kada nema mrežnog napajanja.



Slika 1 Uprošćena logička šema statičke $2^n \times m$ RAM memorije



Slika 2 Logička šema kontrolne mreže

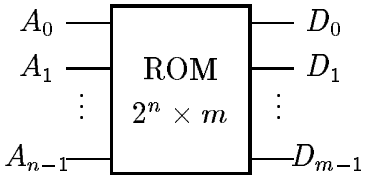


Slika 3 Simbol za statičku $8K \times 8$ RAM memoriju

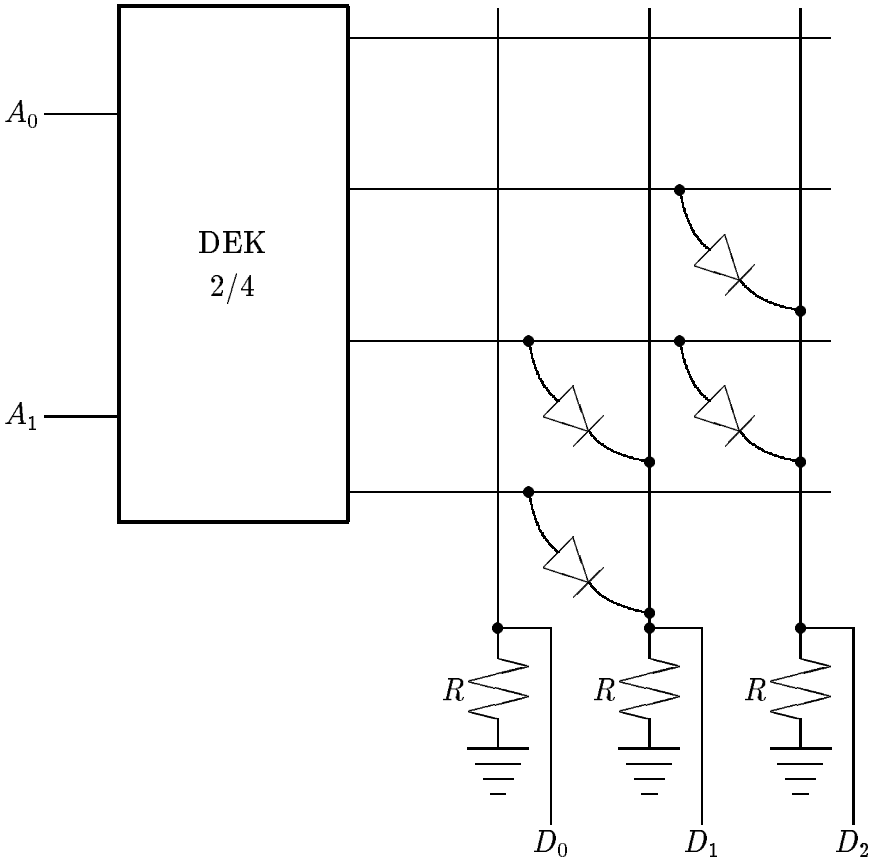
41. ROM MEMORIJE

1 Pojam ROM memorije

Memorija sa konstantnim sadržajem je integrisano kolo u koje se posebnim postupkom upisuje željeni sadržaj, a kada je sadržaj upisan onda memorija može samo da se čita. Takva memorija naziva se engl. read only memory, odnosno ROM memorija. Na slici 1 prikazan je logički simbol ROM memorije oblika $2^n \times m$. To znači da sadrži 2^n riječi (memorijskih riječi) i da pojedina riječ ima dužinu m bita. Ulazni signali obilježeni su kao $A_0, A_1, \ldots, A_{n-1}$ i nazivaju se adresnim ulazima. Izlazni signali obilježeni su kao $D_0, D_1, \ldots, D_{m-1}$ i nazivaju se izlazima podataka. Koju funkciju obavlja prikazani memorijski čip? Veličine A_i definišu broj k u granicama $0 \leq k \leq 2^n - 1$ na način koji je za dekoder uobičajen. Po izlaznim linijama podataka D_0 do D_{m-1} saopštava se sadržaj k–te memorijske riječi.



1 Simbol ROM memorije



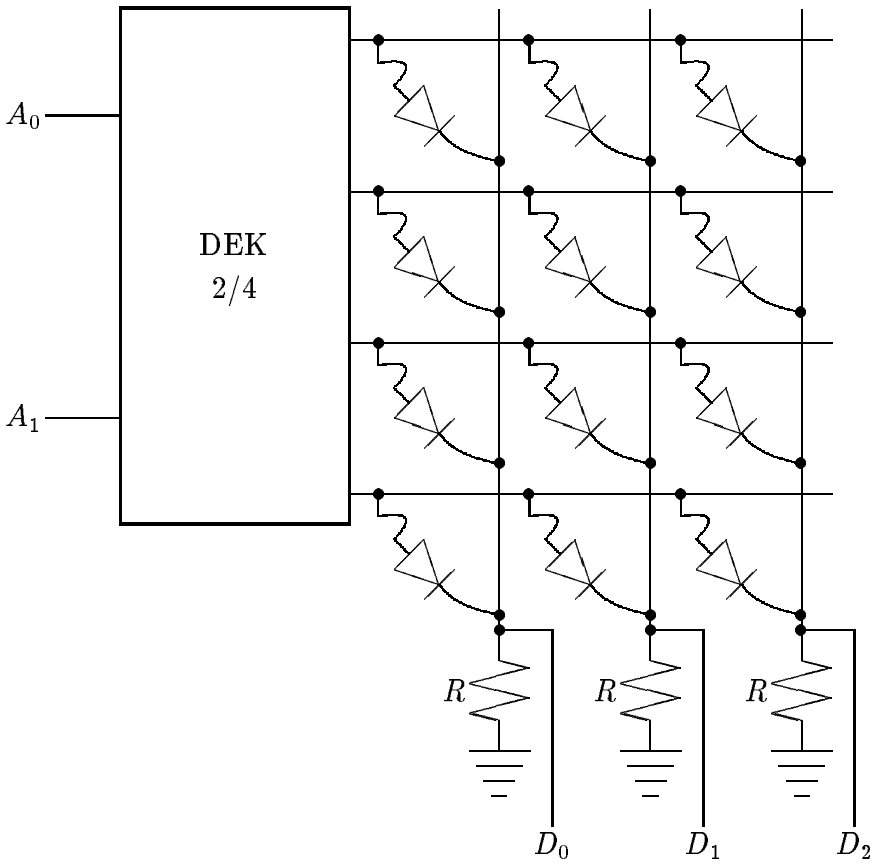
Slika 2 Diodna ROM memorija

2 ROM memorija sa fiksiranim sadržajem (mask–ROM)

Logička šema integrisane ROM memorije prikazana je na slici 2. Memorija je realizovana pomoću dioda. Da bi slika bila manja, stavljeno je $n = 2$ i $m = 3$. Ako je na jednom mjestu dioda prisutna onda se na tom mjestu (na tom bitu) pamti logička jedinica (logički nivo je visok), a ako je dioda izostavljena onda je na tom mjestu upisana logička nula. Dakle, kapacitet memorije je 4 riječi, a pojedina riječ je tro–bitna. Označimo izlaze dekodera kao W_0, W_1, W_2 i W_3 (izlazne adresne linije). Ako je $(A_0, A_1) = (0, 0)$ onda je $W_0 = 1$ a ostali su $W_j = 0$. Ako je $(A_0, A_1) = (0, 1)$ onda je $W_1 = 1$. I slično. Vidi se da na slici učestvuju četiri diode od dvanaest mogućih, tako da se pamte četiri jedinice. Prva memorijska riječ ($W_0 = 1$)

glasi $(D_0, D_1, D_2) = (0, 0, 0)$, jer na prvoj adresnoj liniji W_0 nije priključena nijedna dioda. Za $W_1 = 1$ izlazni kod je 001. Za $W_2 = 1$ izlazni kod ROM–a će biti 011. I četvrto, ako je $W_3 = 1$ onda se saopštava tro–bitna riječ 010. Umjesto dioda, za definisanje sadržaja ROM memorije mogu se koristiti tranzistori.

Programiranje opisanih ROM memorija obavlja se prilikom izrade integrisanog kola. Korisnik od proizvođača naruči sadržaj memorije, proizvođač na osnovu zahtijevanog sadžaja formira masku za izradu integrisanog kola sa priključenim diodama na mjestima gdje sadržaj memorije treba da bude 1. Ili: masku sa priključenim tranzistorima na mjestima gdje sadržaj treba da bude 0. Zbog načina programiranja (načina upisivanja sadržaja) ove memorije se nazivaju mask–ROM. Izrada posebne maske za svaki različit sadržaj ROM memorije je veoma skup proces. Izrada maske košta nekoliko hiljada dolara, a cijena integrisanog kola ROM memorije je nekoliko dolara. Tako da se mask–ROM koristi tamo gdje su potrebe za memorijom sa istim sadržajem najmanje nekoliko hiljada komada.



Slika 3 Programabilna diodna ROM memorija

3 Programabilna ROM memorija (PROM)

Za digitalne uređaje gdje nisu potrebne velike serije ROM memorija sa jednakim sadržajem koristi se programabilna ROM memorija ili skraćeno PROM memorija. Diodne PROM memorije se proizvode sa ugrađenim svim diodama, a redno sa svakom diodom je ugrađen topljivi osigurač, kako je to prikazano na slici 3.

Kada memorija nije programirana, s obzirom da su sve diode priključene, na svim adresama memorije su "upisane" sve jedinice. Korisnik sam programira memoriju time što izazove pregorijevanje osigurača na mjestima gdje želi da sadržaj memorije bude 0. Pregorijevanje osigurača

izaziva se tako što se adresira riječ po riječ, a na linije podataka (gdje treba odstraniti diodu) dovodi se negativni impuls. Za vrijeme trajanja impulsa teći će struja kroz sve diode na toj liniji podataka. Međutim, kroz diodu koja je priključena na adresiranu adresnu liniju struja će biti većeg intenziteta, s obzirom da je anoda adresirane diode na visokom logičkom, odnosno naponskom nivou, dok su anode svih ostalih dioda na naponu logičke nule. Struja kroz adresiranu diodu izaziva pregorijevanje osigurača, čime će na izabranoj adresi i izabranoj poziciji biti upisana logička nula.

Za unikatne ili manje serije digitalnih uređaja, programabilne ROM memorije su daleko ekonomičnije od mask–ROM memorija. Programiranje se obavlja korišćenjem specijalnog uređaja – PROM programatora. U PROM programator se pomoću računara upiše željeni sadržaj memorije. Programator sukcesivno adresira sve adrese priključene PROM memorije i dovodi odgovarajuće impulse na linije podataka na mjestima gdje treba promijeniti inicijalni sadržaj.

Glavni nedostatak PROM memorije je što se jedanput upisani sadržaj ne može promijeniti. Potreba za promjenom sadržaja je veoma česta, naročito prilikom razvoja novih uređaja ili promjene načina funkcionisanja već razvijenih. Za takve primjene, pogodnije je da se koriste programabilne ROM memorije sa mogućnošću brisanja (EPROM memorije).

4 Programabilna ROM memorija sa mogućnošću brisanja (EPROM)

EPROM memorije (engl. erasable programmable read only memory) kao memorijske elemente koriste MOS tranzistore sa izolovanim gejtom. Svaki tranzistor u memorijskoj matrici ima dva gejta. Izolovani gejt je okružen praktično idealnim izolacionim materijalom (SiO_2) i sa neizolovanim gejtom predstavlja kapacitivni razdjelnik napona. Kada EPROM nije programiran, napon logičke jedinice na adresnoj liniji je dovoljan da preko kapacitivnog razdjelnika formira kanal MOS tranzistora, tako da je sadržaj svih lokacija u memoriji nula.

Da bi se na određenoj lokaciji upisala logička jedinica, željena linija podataka i adresna linija priključuju se na visok napon (oko $25V$). U tako selektovanom MOS tranzistoru dolazi do relativno velike struje drejna, elektroni dobijaju veliko ubrzanje, tako da dio elektrona nedestruktivno probija izolaciju i akumulira se na izolovanom gejtu. Kada se isključi visoki napon, izolovani gejt ostaje negativno naelektrisan (na oko $-5V$), tako da pri radnom naponu (tipično $5V$) napon logičke jedinice na neizolovanom gejtu nije dovoljan da formira kanal. I kada je adresiran, MOS tranzistor ostaje neprovodan, odnosno na tom mjestu je upisana logička jedinica, ekvivalentno pregorijevanju osigurača kod PROM memorije. Kvalitet izolacije izolovanog gejta obezbjeđuje da elektronski tovar ostaje na gejtu više od deset godina, čak i kada se EPROM nalazi na temperaturi od 125^oC. Međutim, ako se izolacioni materijal izloži ultraljubičastoj svjetlosti onda SiO_2 postaje slabo provodan i elektroni napuštaju izolovani gejt, odnosno sadržaj svih lokacija ponovo postaje nula. Iz tog razloga, EPROM memorije se izrađuju u kućištima sa providnim prozorom od kvarcnog stakla, kako bi mogle da bude izložene dejstvu ultraljubičastog svijetla u cilju brisanja.

5 EEPROM

Programabilna ROM memorija sa mogućnošću elektronskog brisanja (engl. electrically erasable programmable read only memory, EEPROM) takođe kao memorijske ćelije koristi MOS tranzistore sa izolovanim gejtom. Šema memorije je ista kao za EPROM, s tim da je izolacija između izolovanog gejta i kanala svedena na svega 100nm. Upis logičke jedinice u ćeliju EEPROM–a obavlja se slično kao kod EPROM–a, samo što je sada napon proboja izolacije sveden na oko $10V$. Brisanje EEPROM–a obavlja se električno, tako što se za brisanje na gejt ptikljućuje napon suprotnog polariteta od napona za upis.

Osnovna razlika između EPROM–a i EEPROM–a sastoji se u tome što se prilikom brisanja EPROM–a briše kompletan sadržaj cijele memorije, brisanje traje 10 do 20 minuta i to u komori sa ultraljubičastom svjetlošću. Brisanje EEPROM memorije obavlja se selektivno, samo adresirana riječ, priključivanjem negativnog napona na selektovane memorijske ćelije za vrijeme od oko 10ms po riječi. Savremene EEPROM memorije imaju u istom integrisanom kolu ugrađen pretvarač $5V$ u $\pm 10V$, tako da se upis i brisanje EEPROM–a može obavljati bez isključivanja digitalnog uređaja u kome je ugrađena EEPROM komponenta. Dovoljno je dovesti odgovarajući kontrolni signal na priključak za upis, odnosno za brisanje.

42. DVODIMENZIONO DEKODIRANJE ADRESA MEMORIJE

Znamo da izraz ROM memorija oblika $2^n \times m$ znači da memorija (ili memorijska matrica) ima n adresnih ulaza i onda naravno da sadrži 2^n podataka (memorijskih riječi). Veličina pojedinog podatka jednaka je m bita. Šema ROM memorije malog kapaciteta je: dekoder – matrica. Dakle, u slučaju jednodimenzionog dekodiranja, svi adresni ulazi usmjeravaju se pravo na dekoder. Međutim, ako se radi o memoriji većeg kapaciteta onda je pogodno da (prilikom izbora podatka koji se čita) dekoderu pomaže multiplekser (ili pomažu multiplekseri). Vidjećemo da se jedan dio adresnih linija usmjerava prema dekoderu, a drugi dio prema MUX. Tako da sada šema postaje: dekoder – matrica – multiplekser. Multipleksera ima m, tj. broj multipleksera u mreži zavisi od veličine podatka. Dakle, u ovom naslovu govori se o dvodimenzionom dekodiranju (o dvodimenzionom načinu da se definiše koji podatak treba da bude pročitan). Napominje se da važe iste okolnosti, u pogledu mogućnosti jednodimenzionog ili dvodimenzionog dekodiranja adrese podatka, za RAM memorije i za ROM memorije.

Dekodiranje memorijskih adresa na način kako je to dosad bilo prikazano (na jednodimenzioni način) primjenjuje se samo za memorije manjeg kapaciteta. Realizacija memorije većeg kapaciteta na takav način zahtijevala bi veliki broj logičkih elemenata (za dekoder), a i sama memorijska matrica bi bila nepogodna za izradu u integrisanoj tehnologiji, s obzirom da bi matrica zauzimala prostor veoma izduženog pravougaonika sa velikim brojem vrsta (redova) i malim brojem kolona. Na primjer, memorijska matrica ROM memorije oblika $1K \times 4$ imala bi 1024 reda i samo 4 kolone. Za izradu integrisanih kola je najekonomičnije, odnosno postiže se najveća gustina pakovanja kada su komponente fizički raspoređene unutar kvadrata. Iz tog razloga, za izradu memorije većeg kapaciteta primjenjuje se dvodimenziono dekodiranje, tako da se smanjuje broj logičkih elemenata i istovremeno se fizički raspored komponenti memorije približava kvadratnom obliku.

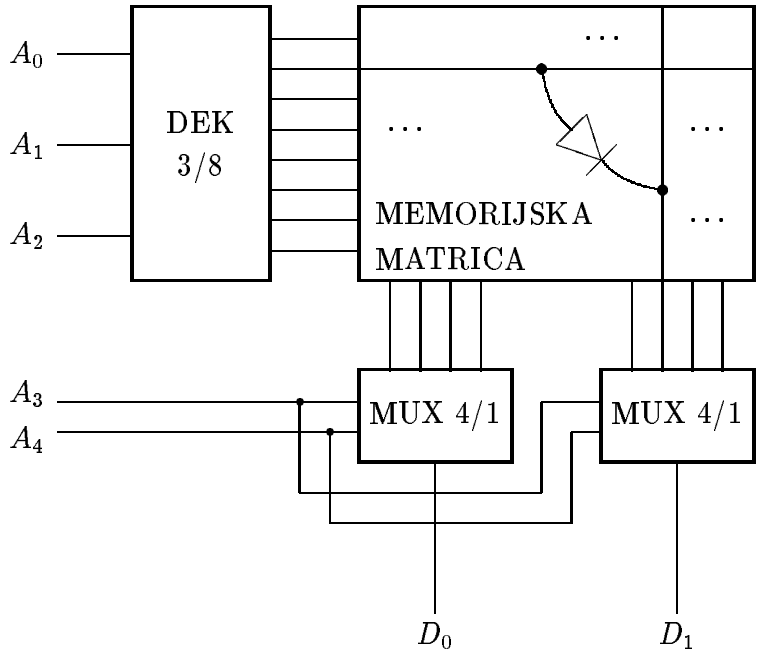
Princip dvodimenzionog dekodiranja prikazaćemo na primjeru sinteze ROM nemorije 32×2. Memorija se realizuje korišćenjem jednog dekodera 3/8, dva multipleksera 4/1 i kvadratne matrice memorijskih ćelija sa 8 redova i 8 kolona. Kada bi se koristilo jednodimenziono dekodiranje, memorija bi sadržala dekoder 5/32 i memorijsku matricu sa 32 reda i dvije kolone. Blok–šema memorije oblika 32×2 sa dvodimenzionim dekodiranjem prikazana je na slici. Dioda je prisutna – znači da je upisana vrijednost 1, a dioda je odsutna – znači 0.

Adresnim ulaznim signalima A_0, A_1 i A_2 selektuje se (preko dekodera) jedna od osam adresnih linija (obično se označavaju kao W_0 do W_7). Na svakoj adresnoj liniji nalaze se priključene memorijske ćelije četiri susjedne memorijske riječi. U mreži su prisutna dva multipleksera. Adresni ulazni signali A_3 i A_4 sprovode se na ulaze za selekciju (obično se označavaju kao S_0 i S_1) jednog i drugog multipleksera. Memorijska matrica ima svojih osam izlaznih linija podataka (vertikalne linije), s tim da se prve četiri linije uvode u prvi multiplekser, a druge četiri linije uvode se u drugi multiplekser; uvode se na ulaze za podatke MUX–a. Izlaz za podatke

prvog MUX–a jeste D_0, a izlaz za podatke drugog MUX–a jeste D_1. Da bi se slika razumjela, svakako da treba znati definiciju MUX–a oblika 4/1: na osnovu selekcionih signala S_0 i S_1 biva izabran jedan od četiri ulazna signala za podatke i izabrani signal propušta se (prenosi se) na izlaz za podatke, izlazni signal MUX–a obično se označava kao D.

Koju funkciju obavlja prikazano imtegrisano kolo? U memorijskoj matrici upisane su 64 binarne vrijednosti, organizovane kao 32 riječi dvo–bitne. Pomoću ulaza A_0 do A_4 definiše se jedan broj k u granicama $0 \leq k \leq 31$. Na izlazu iz mreže (izlazni priključci D_0 i D_1) saopštava se memorijska riječ čiji je redni broj k.

Navedimo još jedan primjer. Razmotrimo ROM memoriju 256×1 sa dvodimenzionim dekodiranjem. Tada imamo sljedeću šemu: dekoder 4/16 – memorijska matrica sa 16 redova i 16 kolona – jedan multiplekser 16/1.



ROM memorija 32×2 sa dvodimenzionim dekodiranjem

43. UKUPNI OPIS RAČUNARA SSEM (MANCHESTER MARK I)

1. Od uređaja za računanje do pravog računara

Gledano po vremenskoj osi, prvi računar konstruisan je 1948. godine u Engleskoj. To je pravi računar. Po svojim karakteristikama poklapa se sa savremenim računarima. Ima program upisan u memoriji. Konstruisan je na Univerzitetu u Manchesteru. Naziva se Small Scale Experimental Machine ili Manchester Mark I. U ovom tekstu biće riječi o tom računaru i o njegovom prvom programu. Prvi program propušten je kroz računar 21. juna 1948. godine. Program je računao najveći činilac učitanog broja. Tvorci računara bili su Fredderic C. Williams, inženjer i Tom Kilburn, matematičar. Williams je radio na izgradnji mašine, a Kilburn na programiranju. Značajno je što je Williams primijenio jednu novu tehnologiju za memorijsku jedinicu. Kilburn je autor prvog programa, za najveći činilac. Za memoriju se koristila jedna katodna cijev (Cathode Ray Tube – CRT), kakvu imaju televizori. Ta nova tehnologija predstavljala je napredak u odnosu na dotadašnje. Patentirana je pod nazivom Williams tube. Dakle, katodna cijev je činila memoriju, dok su ostali djelovi računara (znači procesor) bili izgrađeni od elektronskih lampi (vakuumskih cijevi). Napominje se da je računar sadržao još dvije manje katodne cijevi. Jedna od njih činila je akumulator, a druga je služila za programski brojač (CI) i tekuću naredbu (PI).

Dakle, računar je bio elektronski u cjelini. Kaže se da je njegova arhitektura bila sekvencijalna ili serijska (nije bila paralelna). To znači da se u pojedinom taktu izvodila operacija nad pojedinim bitovima (ne nad čitavim riječima). Učestanost takta iznosi 100KHz.

Eksperimentalna mašina je izgrađena najviše da bi se provjerila vrijednost memorijske tehnologije. Kasnije je izgrađeno i niz novih marki, poboljšanih.

Imajući u vidu eksperimentalnu prirodu računara razumjećemo okolnosti kao što su. Mali kapacitet memorije – svega 32 riječi. Mali skup narebi – svega 7 vrsta naredbi. Neobičan skup naredbi – ima oduzimanje, a nema sabiranje. Odsustvo ulaznih i izlaznih uređaja – prije puštanja programa memorija se popunjavala pomoću svičeva, a kada se izvršavanje programa okonča onda su se rezultati neposredno čitali iz memorije (sa ekrana katodne cijevi).

2. Ukupni opis računara SSEM (Manchester Mark I)

Memorija se sastoji od 32 riječi. Pojedina riječ sastoji se od 32 bita, tako da riječ može da bude prikazana kao $a_{31}a_{30}\ldots a_2a_1a_0$, gdje $a_k \in \{0,1\}$.

Brojevi se prikazuju u obliku potpunog komplementa (PK). Jedan broj zauzima jednu riječ. Tako da $a_{31}a_{30}\ldots a_2a_1a_0$ prikazuje broj $a_{30}\cdot 2^{30}+\ldots+a_2\cdot 4+a_1\cdot 2+a_0$, ako je $a_{31}=0$. Dok recimo riječ $a_{31}a_{30}\ldots a_2a_1a_0 = 1\ldots 100 = 1_{30}00$ ima smisao broja $n=-4$; 1_{30} znači trideset puta napisano 1. Dakle, mogu da budu prikazani brojevi u razmaku $-2^{31}\le n\le 2^{31}-1$. Tačno kao vrsta podatka longint (dugački cijeli) u programskom jeziku paskal.

Pogledajmo tri registra u procesoru. Svaki od tri je veličine 32 bita (po jedne riječi). Samo prva dva su programski dostupni. Registar akumulatora acc služi prilikom izvođenja aritmetičkih naredbi. Registar CI (current instruction) ili svejedno druga oznaka counter sadrži adresu tekuće naredbe. Registar PI (present instruction) ili svejedno order sadrži tekuću naredbu.

Jedna naredba takođe zauzima u memoriji jednu riječ (jednu liniju). Vidjećemo da se u naredbi razlikuju dva polja: vrsta radnje i adresa argumenta. Vidjećemo da je više od polovine bitova u memorijskoj riječi neiskorišćeno, ako riječ tumačimo kao naredbu. Prvo polje $a_{15}a_{14}a_{13}$ obično se naziva opkod (opcode, operation code) ili funkcija (function). Tako da postoji osam mogućnosti za vrstu radnje. Drugo polje $a_4a_3a_2a_1a_0$ obično se naziva adresa ili linija n. Tako da je $0\le n\le 31$, što se slaže sa iskazom: memorija se sastoji od 32 riječi.

Pogledajmo sada spisak sedam vrsta naredbi, odnosno pogledajmo koji je smisao pojedine naredbe, bolje reći koji je učinak izvršnog ciklusa naredbe. Ako piše naredba 001 onda to znači da je $a_{15} = 0$, $a_{14} = 0$ i $a_{13} = 1$ i slično.

vrsta	**učinak naredbe**	**oznaka**	**za ovu naredbu se kaže**
000	counter←store[n]	jump	bezuslovni skok
001	counter←counter+store[n]	jumprel	bezuslovni skok
010	acc←−store[n]	loadneg	donesi u ak., uz okretanje predznaka
011	store[n]←acc	store	odnesi iz akumulatora
100	acc←acc−store[n]	subtract	oduzimanje
110	if acc<0 then counter←counter+1	skipif	uslovni skok
111	stop	halt	računar se zaustavlja

U tabeli, store[n] znači memorijsku riječ n. Vidimo da jedna naredba za bezuslovni skok definiše tačku doskoka na apsolutni način, a druga na relativni način. Vidimo da u obe te naredbe važi indirektni način za tačku doskoka. Znamo da se kod današnjih računara više koristi direktno adresiranje nego indirektno adresiranje, kada su u pitanju bezuslovni skokovi.

Tvorci računara gledali su da bude što manje hardvera. Ugradili su samo oduzimanje, jer preko njega može da bude izraženo sabiranje.

Smisao naredbe za uslovni skok riječima se izražava ovako: ako je u akumulatoru upisan negativan broj onda preskoči jednu naredbu.

Tek sada će da bude kompletirana definicija naredbi. Ovo kompletiranje je značajno za dvije naredbe za bezuslovne skokove. Kod današnjh računara, prilikom izvršavanja jedne naredbe, uobičajeno je da se odvijaju redom sljedeća dešavanja: fetch instruction, increment program counter, execute the instruction. Kod Manchester Mark I dešavalo se redom ovako:

increment program counter, fetch instruction, execute the instruction.

Praktično, ako želimo da pomoću naredbe za bezuslovni skok izvršimo skok na naredbu koja se nalazi na poziciji 9 onda treba zapisati vrijednost 8 negdje u memoriji. Uzmimo da je vrijednost 8 zapisana na poziciji n. Tada naredba za bezuslovni skok u programu glasi jump n.

Sada je završena definicija računara. Naredba jumprel n pročitana kao broj jednaka je $2^{13} + n$. Naredbi loadneg n odgovara broj $2 \cdot 2^{13} + n$. I slično.

Naredbe i podaci unosili su se u računar u čisto binarnom obliku. Prilikom pisanja programa za računar mi ćemo ipak koristiti pogodniji način – simbolički oblik. To znači da se podrazumijeva da svakoj brojnoj vrijednosti treba pridružiti njen odgovarajući potpuni komplement prikaz. A naredbe treba prevesti u mašinski oblik saglasno maločas izloženom formatu instrukcije.

Koliko se vremena potroši za izvršavanje jedne naredbe? Potrebno vrijeme ne zavisi od naredbe, ne zavisi od vrste naredbe, ono je uvijek jedno te isto. Izvršavanje naredbe sastoji se od četiri periode vremena, a jedna perioda vremena jednaka je $w+4 = 32+4 = 36$ taktova, gdje je w dužina riječi izražena u bitovima. Tako da je vrijeme potrebno za jednu naredbu jednako $4 \cdot 36 = 144$ takta. Rekli smo da je 1 takt = 10 μ sec. Dakle, definitivno, jedna naredba = 1,44 m sec vremena. Drukčije rečeno, računar je izvršavao oko 700 naredbi u sekundi.

Pogledajmo mali primjer programa. Neka treba da se saberu brojevi x i y koji se drže na adresama 29 i 30 redom i još treba zatim rezultat sum da bude plasiran na mjesto 31. Program glasi kako slijedi:

1	loadneg 29	acc←−x
2	subtract 30	acc←acc−y

3	store 31	sum←acc
4	loadneg 31	acc←−sum
5	store 31	sum←acc

U prvoj koloni piše na kom mjestu u memoriji se drži ta naredba, a znamo da su od 0 do 31 moguća mjesta. U drugoj koloni pišu same naredbe. Treća kolona služi samo kao naš komentar.

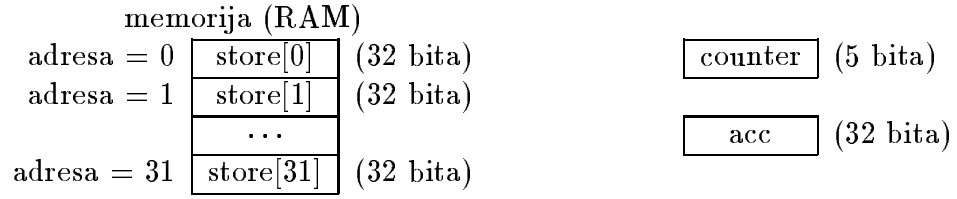
3. Ukupni opis računara SSEM (Manchester Mark I) – rekapitulacija

Tokom rada računara izvršava se program, odnosno izvršavaju se naredbe, korak po korak. Jedan korak znači izvršavanje jedne naredbe. Izvršavanje jedne naredbe ima tri faze. 1) Vrijednost counter poveća se za jedan, increment. 2) Iz memorije se donese sama naredba, fetch. Ona se donosi u procesor. Vrijednost counter determiniše iz koje lokacije se donosi. 3) Izvršavanje naredbe u užem smislu, execute. Prvo se ta naredba dekodira u procesoru. To znači da se izdvoje njeni opkod i adresa. Kaže se da procesor prvo rastumači smisao naredbe. Zatim se izvrše radnje kako naredba traži. Opkod definiše vrstu radnje, a adresa definiše iz koje lokacije treba uzeti argument za tu radnju. Znamo da se naredba sastoji od dva polja: opkod polje $\boxed{a_{15}\,|\,a_{14}\,|\,a_{13}}$ (3 bita) i adresno polje $\boxed{a_4\,|\,a_3\,|\,a_2\,|\,a_1\,|\,a_0}$ (5 bita). Kao što i sama riječ kaže, opkod polje sadrži opkod, a adresno polje sadrži adresu. Navedimo primjer. Ako je $\boxed{a_{15}\,|\,a_{14}\,|\,a_{13}} = \boxed{1\,|\,0\,|\,0}$ onda se radi o naredbi za oduzimanje, umanji akumulator. Konkretno, ako je $\boxed{a_{15}\,|\,a_{14}\,|\,a_{13}} = \boxed{1\,|\,0\,|\,0}$ i $\boxed{a_4\,|\,a_3\,|\,a_2\,|\,a_1\,|\,a_0} = \boxed{1\,|\,1\,|\,1\,|\,0\,|\,0}$ onda se radi o naredbi koja traži da se uradi sljedeće: umanji akumulator za sadržaj lokacije broj n = 28, acc ← acc − store[28]. Memorijska riječ ima oblik $\boxed{a_{31}\,|\,a_{30}\,|\,\ldots\,|\,a_1\,|\,a_0}$.

Mi možemo da snimimo stanje računara na početku nekog koraka. Čemu je jednak counter, čemu je jednak acc i šta piše u memoriji ukupno, tj. čemu je jednak sadržaj njenih lokacija (čemu su jednaki store[0], store[1], ..., store[31]). Posebno, kako glasi naredba koja će u tom trenutku biti izvršena. Dakle, stanje counter, acc, store[0], store[1], ..., store[31] u potpunosti determiniše šta će se desiti tokom izvršavanja tog koraka, odnosno kakvo će biti novo stanje, kakvo će biti stanje na kraju tog koraka. Prema tome, stanja se smjenjuju iz koraka u korak.

Naredba $a_{15}a_{14}a_{13} = 100$, $a_4a_3a_2a_1a_0 = 11100$ nalazi se na nekoj lokaciji u memoriji. U toj lokaciji piše bukvalno $a_{31}a_{30}\ldots a_1a_0 = 00000000000000001000000000011100$. Ili $0_{16}1000_8 11100$. To znači 16 puta 0, zatim 100, zatim 8 puta 0, na kraju 11100. Ako ovako govorimo o programu (o naredbama) onda se mi služimo mašinskim jezikom. Bolje nam je da za opkod uvedemo neku oznaku, a adresu da prikažemo dekadno. Drugim riječima, bolje je da se služimo simboličkim jezikom. Vraćajući se na primjer, simbolički zapis te naredbe glasi subtract 28. Dakle, računar može da razumije samo mašinski jezik, a za programera je mašinski jezik dosta teško razumljiv. Programeru odgovara simbolički jezik (jezik asemblera).

Na slici su prikazani glavni registri i memorija računara SSEM:



Tri vremenska ciklusa prilikom jednog koraka (prilikom izvršavanja jedne naredbe):

1) Increment (povećavanje brojača za jedan),

counter ← counter + 1 (jedino, counter ← 0 ako je bilo counter = 31),

2) Fetch (donošenje naredbe),

memorijska riječ čiji je redni broj counter (gdje je $0 \leq \text{counter} \leq 31$) donosi se u procesor,

3) Execute (izvršavanje naredbe u užem smislu),
izdvoje se opkod i adresa donesene naredbe i zatim se još izvrši radnja kako piše u nastavku.

Uzmimo da je u datom trenutku adresa = 23 i još da je
store[23] = 16, counter = 12 i acc = 3000,
izvršiće se radnja kako piše u nastavku.

Ako je opkod = $a_{15}a_{14}a_{13}$ = 000 (jump, ako govorimo simbolički) onda će postati counter = 16 (a ostalo ostaje po starom, bez promjene).

Ako je opkod = 001 (jumprel) onda će postati counter = 28.

Ako je opkod = 010 (loadneg) onda će postati acc = −16.

Ako je opkod = 011 (store) onda će postati store[23] = 3000.

Ako je opkod = 100 (subtract) onda će postati acc = 2984.

Ako je opkod = 110 (skipif) onda se ništa neće promijeniti, a da je acc < 0 onda bi postalo counter = 13.

Ako je opkod = 111 (halt) onda će se računar zaustaviti.

44. JEDAN PRIMJER PROGRAMA ZA RAČUNAR SSEM (MANCHESTER MARK I)

1. Program za najvećeg činioca

Primjer koji slijedi je značajan. Smatra se da je to prvi pravi program koji je ikad sastavljen i propušten, **prvi program** za računara u pravom smislu riječi, bilo gdje u svijetu.

Zadatak glasi. Dat je prirodan broj n. Među svim njegovim svojstvenim činiocima odrediti najvećeg. Sastaviti odgovarajući program.

U rješenju koje slijedi postupa se ovako. Ispituje se da li je n djeljivo sa k bez ostatka za k = n − 1, k = n − 2, itd. tim redom. Sve dok se ne naiđe na broj k koji je činilac ulaznog podatka n. A kako se gleda da li k dijeli n? Prvo se tekućoj vrijednosti x dodijeli vrijednost ulaznog podatka n (x ← n). Zatim se x u petlji umanjuje za po k svaki put (x ← x − k). Iz petlje se izlazi kada se vidi da je x < 0. Sada se jedno k vrati (x ← x + k). Ako je sada x > 0 onda to znači da k nije činilac (odbacujemo k), a ako je pak suprorotno x = 0 onda smo pronašli rezultat. Zaustavljamo se.

Oni su bili uzeli n = 2^{18} = 262144 kao ulazni podatak. Tako da su naravno kao odgovor dobili da je njegov najveći činilac k = 2^{17} = 131072.

Slijedi rješenje, odnosno tekst programa za Manchester Mark I. Kada se računar Mark I zaustavio onda je sa linije 27 (jer tako predviđa program) mogao da se pročita rezultat 131072 (čisto binarno je bio prikazan). e – enter. `Naredba`, njen učinak, komentar:

e → 1	`loadneg 24`	acc ← −c(24)	acc ← −(n − 1)
2	`store 26`	c(26) ← acc	ℓ ← −(n − 1)
3	`loadneg 26`	acc ← −c(26)	acc ← n − 1
4	`store 27`	c(27) ← acc	k ← n − 1
19 → 5	`loadneg 23`	acc ← −c(23)	acc ← n
8 → 6	`subtract 27`	acc ← acc − c(27)	acc ← acc − k
7	`skipif`	skip one instruction	if acc. is negative
8	`jumprel 20`	forward' c(20)	forward' −3 {+1}
7 → 9	`subtract 26`	acc ← acc − c(26)	acc ← acc − ℓ ≡ acc ← acc + k
10	`store 25`	c(25) ← acc	aux ← acc
11	`loadneg 25`	acc ← −c(25)	acc ← −aux
12	`skipif`	skip one instruction	if acc. is negative

	13	`halt`	halt	
12 →	14	`loadneg 26`	acc ← −c(26)	acc ← $-\ell \equiv$ acc ← k
	15	`subtract 21`	acc ← acc − c(21)	acc ← k − 1
	16	`store 27`	c(27) ← acc	k ← acc
	17	`loadneg 27`	acc ← −c(27)	acc ← −k
	18	`store 26`	c(26) ← acc	ℓ ← acc
	19	`jump 22`	goto' c(22)	goto' 4 {+1}
	20 ‖	`1...101`	−3 (serves for jumprel)	$1_{30}01$
	21 ‖	`0...01`	1	$0_{31}1$
	22 ‖	`0...0100`	4 (serves for jump)	$0_{29}100$
	23 ‖	`1 14 0 18`	$-n = -2^{18} = -262144$	$1_{14}0_{18}$
	24 ‖	`0 14 1 18`	$n - 1 = 2^{18} - 1 = 262143$	$0_{14}1_{18}$
	25 ‖	`0`	auxiliary	0_{32}
	26 ‖	`0`	$\ell = -k$	0_{32}
	27 ‖	`0`	k [it will become the result]	0_{32}

Obična notacija: c(n) ili svejedno contents(n) znači sadržaj od n, gdje je $0 \leq n \leq 31$.

Što se tiče naredbe za bezuslovni skok jump. Naša notacija goto' mjesto znači goto mjesto+1 (skoči na mjesto + 1). Što se tiče naredbe za bezuslovni skok jumprel. Naša notacija forward' koliko znači forward koliko + 1 (skoči unaprijed koliko + 1 mjesta).

Strelice na početku reda služe da se bolje snađemo sa skokovima.

Zadatak ondašnjeg programera bio je i da svaku naredbu ili podatak predviđen za smještanje u memoriju pripremi u čisto binarnom izdanju. Pogledajmo samo u slučaju jedne naredbe šta to znači. Na mjestu 1 treba da bude naredba loadneg 24. Znamo da je format instrukcije:

0...0 (16 bita) zatim opkod (3 bita) zatim 0...0 (8 bita) i na kraju adresa (5 bita).
Tako da je binarni oblik ove naredbe 0000000000000000 010 00000000 11000. Ovo se pripremi u dugačkom redu svičeva. A u kratkom redu svičeva pripremi se 00001, jer ova naredba dođe na mjesto 1. Zatim se pritisne dugme. Jedna naredba je upisana.

Oni su zapisali da je izvršavanje ovog istorijskog programa trajalo 52 minuta. Pažljivim brojanjem po tekstu rješenja nalazimo da treba tačno 16 naredbi da se ispita za jedno k, jer je već $n - 2k < 0$ uvijek. Tako da je ukupan broj izvršenih naredbi jednak približno $16 \cdot 2^{17}$, jer se ispituje za svako k od $k = 2^{18} - 1$ naniže do $k = 2^{17}$. Znamo da jedna naredba troši 1,44 m sec.

2. Program za najvećeg činioca – rekapitulacija

Neka nam kao primjer posluži zadatak o nalaženju najvećeg činioca datog broja, koji glasi. Dat je prirodan broj $n = 2^{18} = 262144$, sastaviti program koji će među svim njegovim svojstvenim činiocima odrediti najvećeg. Jasno je da kao rezultat treba da bude dobijeno da je najveći činilac razmatranog broja broj $k = 2^{17} = 131072$.

Na slici je prikazan plan upotrebe memorije za ovaj program. Drugim riječima, prikazano je koje memorijske riječi sadrže program (sadrže naredbe), a koje sadrže podatke (sadrže brojne vrijednosti). Strelica → pokazuje na kom mjestu počinje izvršavanje programa.

Da bi se rješenje na simboličkom jeziku nekako razumjelo, u nastavku je dat grubi algoritam. Grubi algoritam izražen je na jeziku "nestrogi paskal".

Da bi se primjer nekako razumio, data je i jedna tabela. U tabeli je prikazano kako se tokom rada računara mijenjaju vrijednosti promjenljivih. Smisao promjenljivih koje učestvuju je sljedeći. n – dati broj. k – ispituje se da li je k činilac. Tokom rada programa je $k = n - 1$, $k = n - 2$, ... x – tekuća vrijednost. Tokom ispitivanja za jedno k je $x = n$, $x = n - k$, $x = n - 2k$, ..., sve dok ne postane $x < 0$.

Mamorija (RAM):

0	ne koristi se
→ 1	program
2	
⋮	
18	
19	
20	podaci
21	
⋮	
26	
27	
28	ne koristi se
29	
30	
31	

Grubi algoritam:

```
        n ← 262144;
        k ← n − 1;
one:    x ← n;
two:    x ← x − k;
        if x < 0 then three else two;
three:  x ← x + k;
        if x = 0 then stop;
        k ← k − 1;
        goto one;
```

Tabela: n ← 262144,
k ← 262143, x ← n, x ← x − k, x ← x − k < 0, x ← x + k > 0,
k ← 262142, x ← n, x ← x − k, x ← x − k < 0, x ← x + k > 0,
k ← 262141, x ← n, x ← x − k, x ← x − k < 0, x ← x + k > 0,
...
k ← 131073, x ← n, x ← x − k, x ← x − k < 0, x ← x + k > 0,
k ← 131072, x ← n, x ← x − k, x ← x − k, x ← x − k < 0,
x ← x + k = 0 (the result = k)

3. Primjer: program za množenje dva broja

Množenje se svodi na uzastopno sabiranje: broj x sabira se sa samim sobom y puta. Dva množitelja x i y treba prije početka unijeti na mjesta 29 i 30. Nakon "halt", rezultat product = $x \cdot y$ očitavaće se na mjestu 31. U primjeru je izabrano x = 16 i y = 16 tako da je njihov proizvod jednak product = 256. Dakle, za mjesto 29 programer treba da pripremi 0...010000 (32 bita). Isto za mjesto 30. Na kraju će na mjestu 31 pisati 0...0100000000 (32 bita). Slijedi rješenje:

```
        0 ||  0              0 (serves for jump)
e, 13 → 1     loadneg 30     acc ← −c(30)          acc ← −y
        2     skipif         skip one instruction if acc. is negative
        3     jump 27        goto' c(27)           goto' 13 {+1}
    2 → 4     subtract 28    acc ← acc − c(28)     acc ← acc + 1
        5     store 30       c(30) ← acc           y ← acc
        6     loadneg 30     acc ← −c(30)          acc ← −y
        7     store 30       c(30) ← acc           y ← acc
        8     loadneg 31     acc ← −c(31)          acc ← −product
        9     subtract 29    acc ← acc − c(29)     acc ← acc − x
       10     store 31       c(31) ← acc           product ← acc
       11     loadneg 31     acc ← −c(31)          acc ← −product
       12     store 31       c(31) ← acc           product ← acc
       13     jump 0         goto' c(0)            goto' 0 {+1}
    3 → 14    halt           halt
       27 ||  1101           13 (serves for jump) 00000000 00000000 00000000 00001101
       28 ||  1...1          −1
       29 ||  10000          x = 16
       30 ||  10000          y = 16
       31 ||  0              product = 0 [initially]
```

Na kraju, ponovimo definiciju u asemblerskoj notaciji. Znamo da c znači "contents" i da važi $0 \leq n \leq 31$, $0 \leq CI \leq 31$. Opšti sistem rada računara: increment CI, execute instruction, increment CI, execute instruction, increment CI, execute instruction, etc. Prilikom "increment CI" ostvaruje se: $CI \leftarrow CI + 1$. Zavisno od slučaja, prilikom "execute instruction" ostvaruje se: `STORE n` $c(n) \leftarrow AC$, `LOADNEG n` $AC \leftarrow -c(n)$, `SUBTRACT n` $AC \leftarrow AC - c(n)$, `HALT` stop the computer, `SKIPIF` if $AC < 0$ then $CI \leftarrow CI + 1$, `JUMP n` $CI \leftarrow c(n)$, `JUMPREL n` $CI \leftarrow CI + c(n)$. Najzad, ako je na startu bilo (recimo) $CI = 0$ onda će se prva izvršiti instrukcija sa adrese $CI = 1$, naravno. Kaže se računar Manchester Mark I ili SSEM ili Baby.

45. SERIJSKI SABIRAČ

U ovom naslovu biće konstruisana mreža za sabiranje dva binarno prikazana broja $x = (x_n \dots x_2 x_1)_2$ i $y = (y_n \dots y_2 y_1)_2$ od po n bita. Za tu mrežu kaže se da predstavlja serijski sabirač budući da je potrebno n taktova da se izračuna rezultat $z = x + y = (z_n \dots z_2 z_1)_2$. Na ulaz mreže redom pristižu cifre jednog i drugog sabirka x_k i y_k a isto tako se postepeno obrazuju i cifre zbira z_k, gdje je $k = 1, 2, \dots, n$. Dakle, tri broja x, y i z drže se u tri pomeračka registra.

Neka je $x = 2^{n-1}x_n + \dots + 2x_2 + x_1$ gdje $x_k \in \{0, 1\}$ za $1 \leq k \leq n$, i slično za y i z. Ovdje je $n \geq 1$, a u primjeru kasnije biće izabrano $n = 8$. Dobro su poznate sljedeće relacije koje odražavaju zakon sabiranja u binarnom brojnom sistemu:

$$\begin{cases} z_k = x_k \oplus y_k \oplus q_{k-1} \\ q_k = x_k y_k \vee x_k q_{k-1} \vee y_k q_{k-1} \end{cases}$$

Pomoćna promjenljiva q_{k-1} ima smisao prenosa koji se formira prilikom sabiranja na $(k-1)$–voj koordinati a zatim utiče na formiranje k–te cifre zbira. Upravo, stari prenos q_{k-1} utiče na formiranje tekuće cifre zbira z_k i novog prenosa q_k. Treba staviti $q_0 = 0$.

Šema serijskog sabirača prikazana je na slici. Serijski sabirač je jedna sekvencijalna mreža: sastavljen je od logičkih elemenata (čine logičku mrežu na slici) i jednog memorijskog elementa, upravo jednog D flipflopa. Znamo da je izlazni signal Q tog flipflopa jednak njegovom ulaznom signalu D ali iz prethodnog takta, formulom $Q(t) = D(t-1)$, gdje je t broj takta ili vrijeme (koliko μsec). Za $Q(t)$ se kaže da je tekuće stanje flipflopa. Vidimo da je na slici povezano tako da važi $q(t) = D(t)$ za svako t i $Q(t) = q'(t)$ za svako t. Sliku bi trebalo upotpuniti: prikazati i upravljačke signale koji ostvaruju start i stop za cirkulaciju podataka u ukupnoj mreži i još ostvaruju da se na samom početku flipflop resetuje $Q(0) \leftarrow 0$.

Na ulaze serijskog sabirača pristižu iz takta u takt po jedna cifra prvog i drugog sabirka, a na izlazu se saopštava jedna cifra rezultata. Takođe, iz takta u takt mijenja se stanje flipflopa. Upravljajući se po šemi, analizirati rad predložene mreže, odnosno pratiti kako se tokom vremena mijenjaju vrijednosti u karakterističnim tačkama mreže. Pogodno je da se zapiše u obliku tabele:

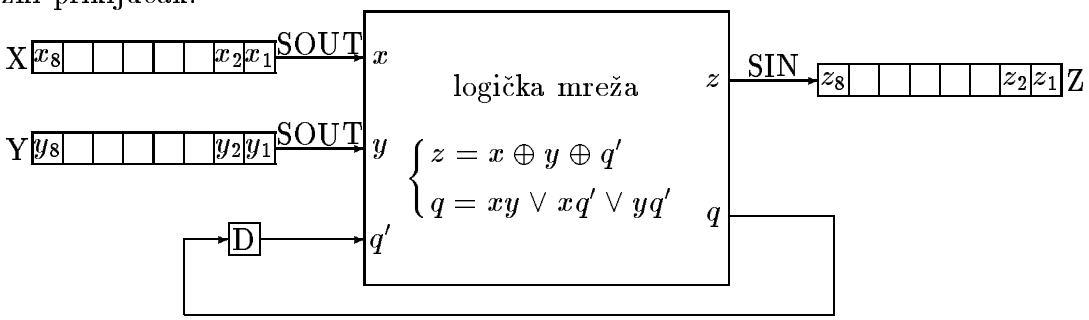
$t = \dots$ (μsec)	0	1	2		n
ulaz $x =$		x_1	x_2		x_n
ulaz $y =$		y_1	y_2		y_n
stanje $q =$	0	q_1	q_2		q_n
izlaz $z =$		z_1	z_2		z_n

Pogledajmo primjer 00000110 + 00000011 = 00001001 (binarno) ili u prevodu 6 + 3 = 9 (dekadno). U razmatranoj mreži važi $q'(t) = q(t-1)$:

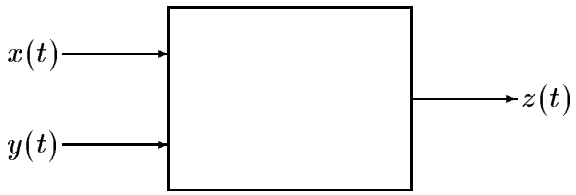
$t =$	0	1	2	3	4	5	6	7	8
$x =$		0	1	1	0	0	0	0	0
$y =$		1	1	0	0	0	0	0	0
$q =$	0	0	1	1	0	0	0	0	0
$z =$		1	0	0	1	0	0	0	0

Svaka sekvencijalna mreža može da bude realizovana ako su na raspolaganje stavljeni logički elementi iz nekog kompletnog sistema plus elementi jediničnog kašnjenja D čija jednačina glasi izlaz(t) = ulaz$(t-1)$.

SIN je serijski ulazni priključak pomeračkog registra (šift registra), a SOUT je njegov serijski izlazni priključak.



Slika: Serijski sabirač. Za flipflop D: izlaz(t) = ulaz$(t-1)$



Slika: Serijski sabirač kao cjelina

46. SERIJSKI I PARALELNI PRENOS DIGITALNIH INFORMACIJA

Slika 1 odnosi se na serijski prenos digitalnih informacija. Mreža se sastoji od dva četvorobitna pomeračka registra A i B. Predajni registar A ima serijski izlaz SOUT a prijemni registar B ima serijski ulaz SIN. SOUT se veže na SIN. Signal takta CLK je zajednički a kod A i B označen je kao C odnosno C', a uvodi se naravno na ulaze za dozvolu flipflopova. Vidi se da će nakon generisanja četiri taktna impulsa biti postignuto B $\leftarrow$ A.

Registar A može biti lociran u digitalnom uređaju koji šalje podatke a registar B može biti fizički dislociran kao sastavni dio uređaja kome se podaci šalju. Vidimo da se informacija prenosi na fizički udaljeno mjesto korišćenjem samo dvije električne veze: informacione linije SOUT – SIN i linije takta CLK – C – C'. Kaže se da se vrši sinhroni serijski prenos.

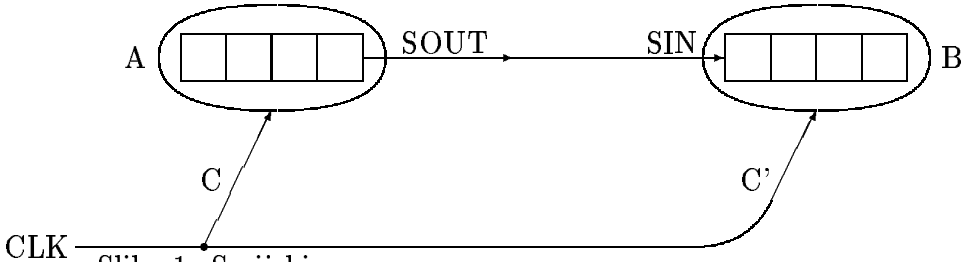
Kod asinhronog serijskog prenosa šalje se samo informacija, dok se takt prijemnog pomeračkog registra generiše lokalno. To omogućava da se informacije prenose samo po jednoj liniji, što je naročito pogodno u sistemima gdje se za prenos koriste standardni telekomunikacioni sistemi.

Slika 2 odnosi se na paralelni prenos digitalnih informacija. Posmatrajmo predajni registar A centralnog procesora i prijemni registar B štampača. Štampač radi puno sporije, pa želimo da centralni procesor šalje podatke po paketima. Između A i B postavi se mreža M za koju se kaže da predstavlja bafer. Mreža M sastoji se od $m+1$ pomeračkih registara R_0, R_1, ..., R_{m-1},

R_m. R_0, R_1, ..., R_{m-1} su informacioni registri. Registar R_m ima kontrolnu ulogu i naziva se flag registar. Mi smo uzeli u našem primjeru da je $m = 4$. Svi pomerački registri R_0, R_1, ..., R_{m-1}, R_m su jedne te iste veličine. Oni su veličine po n bita. Mi smo uzeli u našem primjeru da je $n = 8$. S druge strane, dva registra A i B su veličine po m bita. Ukupna mreža A – M – B funkcioniše na sljedeći način. Tokom prvog takta $t = 1$ iz A se pošalje po jedan bit informacije svakom informacionom pomeračkom registru R_0, R_1, ..., R_{m-1}. Tokom $t = 2$ pošalje im se po još jedan bit. Itd. Poslije $t = n$ čitav paket je poslat. Sada je bafer pun. Sljedeći koraci vršiće se po drugom taktu (po znatno sporijem taktu) koji odgovara registru B (odgovara štampaču). Informacije koje su tokom $t = 1$ bile poslate iz A u M sada se nalaze u R_0, R_1, ..., R_{m-1} u krajnjim desnim flipflopovima i one će tokom sljedećeg takta biti poslate registru B. Dakle, u nastavku se (tokom n koraka) informacije iz M postepeno šalju prijemnom digitalnom uređaju.

U ukupnoj mreži A – M – B, za regulisanje komunikacije, pored flag registra, služe i dodatne linije veze.

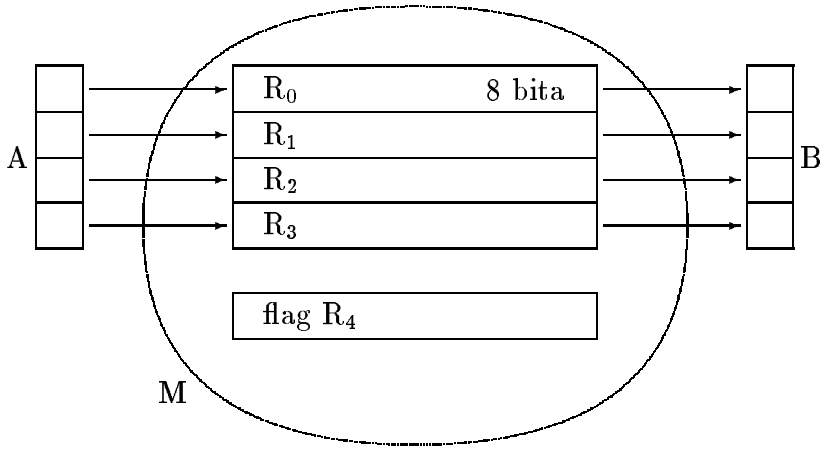
Dok se sadržaj bafera bude štampao, centralni procesor će obavljati druge radnje. Kada se štampanje jednog paketa podataka završi, uređaj A dobija obavještenje o tome, tako da uređaj A sada može da pošalje novi paket podataka za štampanje.



Slika 1: Serijski prenos

Uz sliku 1: Nakon t taktova u dva registra A i B piše (lijevo A, desno B):

$t = 0$	a_1	a_2	a_3	a_4	b_1	b_2	b_3	b_4
$t = 1$	0	a_1	a_2	a_3	a_4	b_1	b_2	b_3
$t = 2$	0	0	a_1	a_2	a_3	a_4	b_1	b_2
$t = 3$	0	0	0	a_1	a_2	a_3	a_4	b_1
$t = 4$	0	0	0	0	a_1	a_2	a_3	a_4



Slika 2: Paralelni prenos (sa baferom M)

Računari i programiranje
Sadržaj predavanja:

1 Digitalni i analogni sistem za obradu podataka prvab.tex
2 Ideja o hardveru i softveru prvab.tex
3 Šema računara prvab.tex
4 Razvoj računarske tehnike prvab.tex
5 Tehnologije izrade, razvoj mikroprocesora prvab.tex
6 Brojni sistemi prvab.tex
7 Prevođenje broja iz jednog sistema u drugi prvab.tex
8 Uvod o flipflopovima, registrima i memoriji prvab.tex
9 Nepokretni zarez i sabiranje u nepokretnom zarezu prvax.tex
10 Pokretni zarez prvax.tex
11 BCD kodovi i ASCII kod prvax.tex
12 Matematički pojam kodiranja prvax.tex
13 Azbučno kodiranje, prefiksno azbučno kodiranje prvax.tex
14 O iskaznom računu prvac.tex
15 Karakteristike idealnog logičkog elementa prvac.tex
16 Karakteristike realnog logičkog elementa prvac.tex
17 Primjeri logičkih elemenata u digitalnoj elektronici prvad.tex
x Zadaci za vježbu, Bulove funkcije, postavke zadataka i rješenja prvaf.tex
18 Pojam Bulove funkcije, fiktivne promjenljive prvag.tex
19 Superpozicija funkcija, formule prvag.tex
20 Glavni identiteti, dualna funkcija prvag.tex
21 SDNF Bulove funkcije prvag.tex
22 Primjeri kompletnih sistema prvag.tex
23 Sabirač prvag.tex
24 Dekoder prvag.tex
25 Multiplekser prvah.tex
26 Trostatički baferi prvah.tex
27 Magistrale digitalnih signala prvah.tex
28 Pojam digitalnog kola prvai.tex
29 Pojam kombinacione mreže prvai.tex
30 Pojam sekvencijalne mreže prvai.tex
31 Pojam bistabilnog kola prvai.tex
32 SR leč kolo prvaj.tex
33 D leč kolo prvaj.tex
34 Sinhroni flipflopovi SR i D prvaj.tex
35 Flipflopovi JK i T prvak.tex
36 Stacionarni registri prvam.tex
37 Pomerački registri prvam.tex
38 Brojači prvam.tex
39 Primjer aritmetičko–logičke jedinice – ALU SN74181 prvaq.tex
40 RAM memorije prvaq.tex
41 ROM memorije prvas.tex
42 Dvodimenziono dekodiranje adresa memorije prvas.tex
43 Ukupni opis računara SSEM (Manchester Mark I) prvat.tex
44 Jedan primjer programa za računar SSEM (Manchester Mark I) prvat.tex
45 Serijski sabirač prvay.tex
46 Serijski i paralelni prenos digitalnih informacija prvay.tex

Fajlovi (gsview): prvab.tex 15 pages prvax.tex 10 pages prvac.tex 7 pages prvad.tex 4 pages prvaf.tex 6 pages prvag.tex 15 pages prvah.tex 7 pages prvai.tex 4 pages prvaj.tex 5 pages prvak.tex 2 pages prvam.tex 7 pages prvaq.tex 6 pages prvas.tex 5 pages prvat.tex 6 pages prvay.tex 3 pages. $\Sigma = 15$ files & 102 pages