**L04**

***Sloj digitalne logike - Sekvencijalna kola***

**CLOCK SIGNAL**

Na logickom nivou digitalna kola se dele na dve klase: Ona koja ne poseduju memoriju nazivamo kombinacionim kolima, I ona koja poseduju memoriju nazivamo sekvencijalna kola.

Ponasanje kombinacionog kola u potpunosti je odredjeno istinitosnom tablciom ili skupom jednacina koje odredjuju koje ce se vrednosti naci na ilzazu na osnovu svake ulazne kombinacije.

U praksi postoji malo kasnjenje prilikom racunanja u ovim kolima ali je ono veoma malo I ono se meri u nanosekundama tako da se smatra da je vreme odziva nula.

***POJAM SEKVENCIJALNIH KOLA***

Digitalna kola koja izvrsavaju zadatak kao sekvencu koraka nazivamo sekvencijalna kola. Ova kola sadrze memoriju koja pamti vrednost logickih signala. Sadrzaj svih memorijskih elemenata u jednom sekvencijalnom kolu predstavlja **stanje** kola. Vremenom kada se promene vrednosti na ulazu, kolo prolazi kroz sekvencu stanja. Kola koja se ponasaju ovako nazivaju se sekvencijalna kola.

Prednost sekvencijalnog rada je sto je za izvrsenje svakogh koraka potrebno ugraditi jednostavan hardver, dok sa druge strane treba da protekne znatno duzi period dok se ne dobije rezultat.

***STRUKTURA I PODELA SEKVENCIJALNIH KOLA***

Sekvencijalna logicka kola se mogu podeliti na sinhrona I asinhrona.

Kod sinhronih kola interna stanja se menjaju u diskretnim vremenskim trenucima pod kontrolom impulsa za sinhronizaciju koji nazivamo signal ili clock signal.

**ON** vreme se definise kao period dok se kolo nalazi u stanju 1, a **OFF** dok se nalazi u stanju 0.

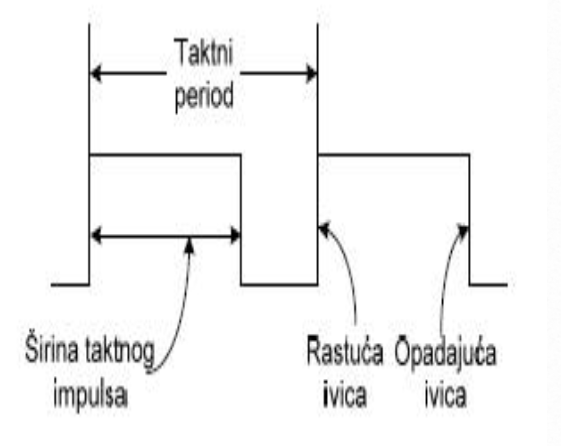
***TAKTNI PERIOD***

Prelaz sa 0 na 1 se naziva pozitivna ivica ili usponska ivica dok prelaz sa sa 1 na 0 – negativna ili opadajuca ivica

Sinhrona sekvencijalna kola su poznata I kao taktovana sekvencijalna kola.

**Taktni perio** je vremenski interval izmedju dve uzastopne promene taktnog signala u istom smeru, tj. Izmedju dve rastuce ili dve opadajuce ivice.

Reciprocna vrednost taktnog perioda je **taktna frekvencija. Sirina taktnog impulsa** je vreme u toku koga je taktni signal jednak 1.



**MEMORIJSKA KOLA**

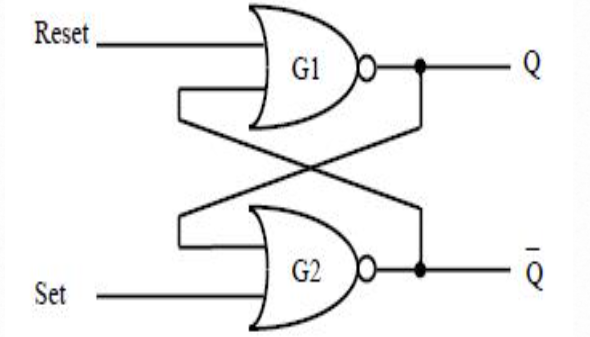
***Bistabilna kola***

Element koji zadrzava tj pamti trenutno stanje po pri prestanku signala, naziva se **memorijski element.**

U digitalnoj tehnici kao memorisjki elementi se koriste bistabilna kola koja imaju dva stabilna stanja. Bistabilno kolo moze da memorise informaciju od 1 bita. Dva osnovna tipa bistabilnih kola su **latch I fli-flops.**

***SR latch kolo***

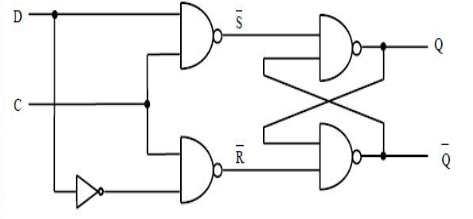
SR latch je najednostavniji memorijski element koji se koristi za projektovanje digitalnih sistema. SR latch cine dva unakrsno vezana NOR kola.

Ulazi u latch se oznacavaju sa Set I Reset. 

***D latch***

Da bi se eliminisalo nezeljeno nedefinisano stanje kod SR latcha je da se obezbedi da ulazi Set I Reset nikada ne budu istovremeno jednaki nuli. Ovo se izvodi kod D latcha.

D latch ima dva ulaza Data I Control.



***Flip flopovi***

Flip flopovi se definisu kao bistabilna kola koje koristi specijalni upravljacki signal C radi odredjivanja trenutaka u kojima se memorija odziva na promene.

Da bi ogranicili memorijski element na samoo jednu promenu stanja kada je latchu dozvoljen rad, neophodno je koristiti transparentne memorijske lemente kao sto su fli flopovi.

Postoje Ivicno okidani D flip flop.

Postoje I 4 tipa flip flopova sa asinhronim ulazima: SR Flip flop, JK flip flpo, D flip flop, T flip flop.

***Registri***

Registar je memorijska komponenta koja se sastoji od n broja flip flopova sa zajednickim taktnim signalom.

U svom osnovnom obliku registar pored taktnog signala ima n ulaza I n izlaza.

Sinhronizovano sa aktivnom ivicom taktnog signala, u svaki flip flop se upisuje jedan bit informacije.

Ako registar treba da se resetuje ili setuje u zavisnoti od taktnog signala u sled neke promene, mogu se dodati signali za asinhrono resetovanje ili setovanje.

***Registar sa dozvolom***

Kod obicnog registra novi podatak se automatski upisuje u registar sa svakom rastucom ivicom takta. Medjutim kod mnogih sistema podataka koji je upisan u registar ostaje upisan tokom nekoliko taktnih ciklusa dok se ne upise novi podatak. Zbog ovog mogucnost kontrole upisa predstavlja korisnu funkciju registra. Ova kontrola se postize koriscnejem upravljackog signala Enable **(En)** koji dozvoljava novi upis podataka kada je signal na jednici.

***Pomeracki registar***

Pomeracki registar omogucava pomeranje upisanog sardzaj za jednu bit poziciju.

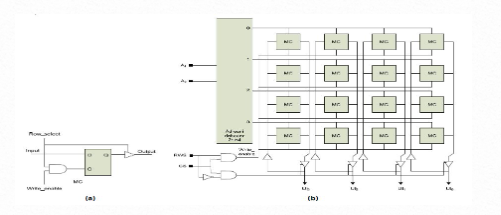
**RAM**

Memorija sa proizvoljnim pristupom ili (**Random access memory**) predstavlja spoljnu memoriju velikog kapaciteta za dugotrajno smestanje podataka koji se koriste prilikom racunanja. RAM je organizovan u vidu polja od 2 na n vrsta sa m bita u svakoj vrsti.

U opstem slucaju n se krece izmedju 16 I 32 dok je m obicno 1, 4, 8, 16 I 32.

RAM memoriju cine: polje memorijskih celija, adresni dekoder I ulazno izlazni baferi.

Memorijsku celiju mozgu simoblicki da prikazem kao strukturu koju cine taktovani D latch, jedno AND kolo I izlazni trostaticki bafer.



***Staticki I dinamicki RAM***

Staticki RAM se konstruise na bazi memorisjih celija sa 4 do 6 tranzistora kod kojih se D latch realizuje uz pomoc unakrsno povezanih tranzistora, dok se za AND kolo I trostaticki bafer koristi jos po jedan tranzistor. SRAM memorija cuva upisani sadrzaj sve dok se ne upise novi ili iskljuci napajanje.

Dinamcki RAM za realizaciju memorijske celije koristi samo jedan tranzistor. Ovakva memorisjka celija gubi upisan sadrzaj pri svakom citanju.

SRAM I DRAM su tzv nepostojece memorije ili **volatile memory** s obzirom da se njihov sadrzaj gubi prilikom prestanka napajanja.

**STACK I QUEUE memorija**

***Stack memorija***

Stek je po definicji memorija sa ogranicenim pristupom. Podacima se u steku pristupa iskljucivo preko jedne lokacije tj vrha steka.

Kad se podatak upise u stek svi podaci koji su vec upisani se spustaju za jedno mesto nadole. (**push operacija**)

Kada se podatak cita iz steka uzima se samo zadnji dodati podatak a svi ostali ispod njega se pomeraju za jedno mesto na gore. (**operacija pop**)