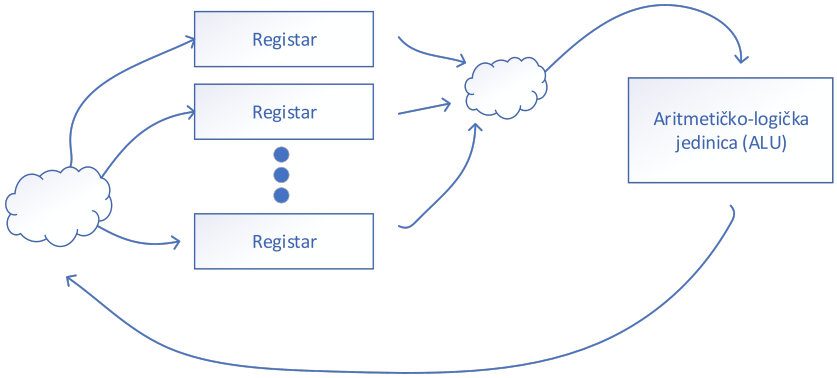
A3. Auditorna vežba 3: Sekvencijalne mreže

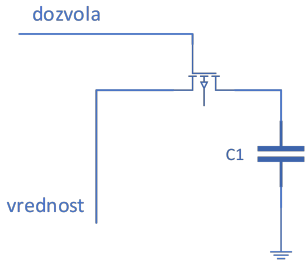


Memorija u digitalnim sistemima

Da bismo rešili problem projektovanja registra kao memorijskog elementa u digitalnim sistemima, razmotrićemo nekoliko ideja za formiranje memorijskih elemenata. Za svaku ideju ćemo dati opis ponašanja, prednosti i mane. Da bi memorijski element bio koristan, moramo da omogućimo pamćenje podatka (jednog bita informacije), operaciju čitanja iz memorijskog elementa i operaciju pisanja u memorijski element.

Prvi pokušaj: kondenzator

Analizirajući dostupne komponente iz elektrotehnike, mogli bismo doći do zaključka da nam kondenzator može pomoći prilikom pamćenja podataka, jer znamo da se kondenzator može napuniti na određeni napon i ukoliko kondenzator držimo u otvorenom kolu, ne bi trebalo doći do promene napona na kondenzatoru.



Sledi idejna analiza ponašanja ovog sistema:

- Ako je signal dozvole jednak 0, MOSFET je isključen i nema pristupa kondenzatoru od strane linije za vrednost. Ova situacija nam odgovara dok želimo da vrednost zapisana u kondenzatoru bude zapamćena.
- Ako je signal dozvole jednak 1, MOSFET je uključen i linija za vrednost je spojena na jednu stranu kondenzatora, dok je druga strana kondenzatora povezana na tačku nultog potencijala. Varijante ponašanja su sledeće:
  - Ukoliko na liniji za vrednost ne forsiramo neku vrednost napona, ona će imati potencijal jednak naponu na kondenzatoru. Ovo je operacija čitanja vrednosti sa kondenzatora.
  - Ukoliko liniju za vrednost spojimo na nulti potencijal, kondenzator je kratko spojen i isprazniće se. Ovo je upis vrednosti 0.
  - Ukoliko liniju za vrednost spojimo na visoki potencijal, kondenzator će se napuniti do tog napona. Ovo je upis vrednosti 1.

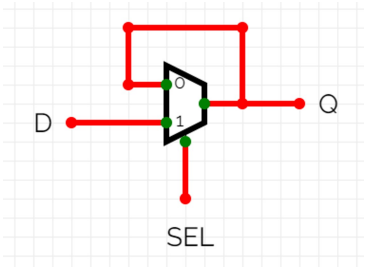
Ova idejna realizacija memorijske lokacije je jeftinija od ostalih pristupa koje ćemo opisati, odn. troši manje komponenti (tranzistora) po bitu zapamćenog podatka. U realnoj realizaciji, ne pravi se poseban kondenzator, već se sama lokacija projektuje tako da parazitna kapacitivnost samog MOSFET-a bude iskorišćena za pamćenje jednog bita podatka.

Nedostatak ove realizacije je sporost u izvršenju operacija čitanja i pisanja, pošto je punjenje i pražnjenje kapacitivnosti mnogo sporija operacija od procesorskih operacija. Takođe, nijedan MOSFET nije idealan, pa je kapacitivnost uvek kroz neku konačnu otpornost spojena u kolo, zbog čega dolazi do blagog pražnjenja upamćenog napona. Zbog toga ove memorije zahtevaju periodično osvežavanje upamćenih vrednosti, što dodatno usporava rad ovih memorija u odnosu na procesor.

Na ovom idejnom rešenju su zasnovane dinamičke radne memorije sa slučajnim pristupom (DRAM) i sva kasnija unapređenja ove tehnologije (DDR, DDR2, DDR3, ...). Međutim, mi ćemo pokušati registar da rešimo na drugačiji način, jer nam je cilj da napravimo komponentu koja će raditi na istoj brzini kao ostatak procesora.

Drugi pokušaj: multiplekser

Posmatrajmo ideju pamćenja na sledeći način: uvedimo neki signal koji označava da li želimo da se zapamćena vrednost pamti, odn. zadržava ili želimo da je promenimo; ako je taj signal na jednoj vrednosti (npr. na vrednosti 0), želimo da se zapamćena vrednost zadržava, a u suprotnom želimo da je promenimo. Ovaj način uslovnog ponašanja nam govori da nam možda multiplekser može pomoći u implementaciji memorijske lokacije.



Sledi idejna analiza ponašanja ovog sistema:

- Ako je signal selekcije multipleksera na vrednosti 0, izlazna vrednost Q se ne menja (pamćenje vrednosti);
- Ako je signal selekcije multipleksera na vrednosti 1, izlazna vrednost Q se menja na vrednost ulaza D (upis vrednosti 0 ili 1);
- U svakom trenutku se na izlazu Q vidi koja vrednost je "zapamćena" (čitanje vrednosti).

Ovo je primer komponente koja se zove **leč (**engl. ***latch*)**. Prema analizi navedenoj iznad, zaključujemo da se ova komponenta ponaša kao memorijski element koji pamti 1 bit podatka. Ostale načine implementacije leča ste učili na predavanjima.

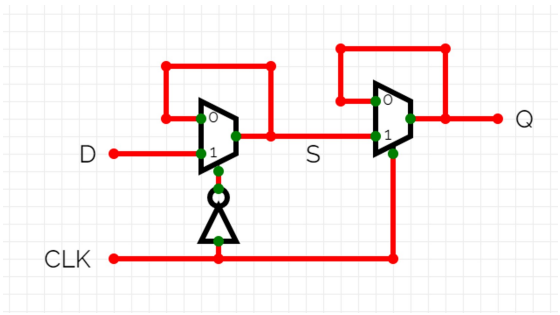
Međutim, leč ima svoje nedostatke.

- Promena vrednosti leča se dešava kad god je signal selekcije na jedinici. Ukoliko signal selekcije realizujemo kao periodični signal, kako bismo periodično imali intervale vremena u kojima se vrednost menja i intervale vremena u kojima se vrednost pamti, promena vrednosti u leču će se dešavati bilo kada tokom intervala vremena u kome je selekcija na vrednosti 1. Ovo može predstaviti ozbiljan problem sinhronizacije sistema ukoliko u njemu imamo više lečeva.
- Ukoliko je selekcija na vrednosti 1, izlaz Q je kombinaciona funkcija ulaza D, i to Q = D, što znači da će se ova komponenta u tom slučaju ponašati kao kombinaciona mreža. Zbog toga ona ne sme ući u kombinacionu petlju sa ostalim kombinacionim mrežama u sistemu, jer bi takva kombinaciona petlja poremetila funkciju ostalih kombinacionih mreža koje su deo petlje.

Zbog gore navedenih nedostataka, a pre svega zbog drugog nedostatka, leč ne može biti rešenje koje ćemo iskoristiti u procesoru jer nam je petlja neophodna da bismo dobili podršku za izvršavanje više instrukcija koristeći iste registre.

Treći pokušaj: bistabilna komponenta

Posmatrajmo sada ideju povezivanja dva leča u tzv. vodeći-prateći konfiguraciju kao na šemi ispod.



Ukoliko su signali selekcije međusobno inverzni, nikada se neće desiti slučaj da je izlaz direktno funkcionalno zavistan od ulaza, odn. $Q \neq f(D)$ zbog čega ova komponenta može da se stavlja u petlju sa kombinacionim mrežama, ne praveći kombinacionu petlju.

Vremensko ponašanje ove veze lečeva je opisano u nastavku. Izlaz Q se menja isključivo u **vremenskim trenucima u kojima je signal takta (CLK) na rastućoj ivici**, odn. kada se taj signal menja sa vrednosti 0 na vrednost 1. Ovo nam uvodi i vremensku diskretizaciju promena u digitalnom sistemu sa više memorijskih elemenata, što nam olakšava sinhronizaciju promena u sistemu.

Da li ova komponenta implementira sve potrebne operacije memorijskog elementa?

Da. Izlaz Q je u svakom trenutku jednak zapamćenoj vrednosti bita (čitanje), vrednost se pamti između dve rastuće ivice selekcionog signala, a upis vrednosti se vrši na rastućoj ivici selekcionog signala, i to na ulaznu vrednost D (dakle, imamo i pisanje nule i pisanje jedinice).

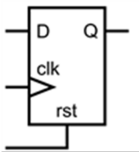
Ovo je primer komponente koja se zove **flip-flop,** odn. **registar** (engl. *flip-flop, register*). Ona spada u tzv. bistabilne komponente, odn. komponente koje se mogu zadržati u dva stabilna stanja. Jedno stanje je zapamćena vrednost 0, a drugo stanje je zapamćena vrednost 1. Prema analizi navedenoj iznad, zaključujemo da se ova komponenta ponaša kao memorijski element koji pamti 1 bit podatka. Ostale načine implementacije flip-flopa ste učili na predavanjima.

Selekcioni signal, odn. onaj koji definiše rastuće ivice na kojima se vrši promena vrednosti flip-flopa, zovemo **signal takta** i obično obeležavamo sa **iCLK**.

Flip-flop se dodatno obogaćuje i **signalom reseta** čiji je zadatak da komponentu vrati u početno stanje, što je najčešće upis vrednosti 0 bez obzira šta je na ulazu D. Osnovni tipovi reset signala:

- **Asinhroni reset** - implementira se kao sastavni deo flip-flopa; resetuje flip-flop na početnu vrednost čim se pojavi.
- **Sinhroni reset** - implementira se kao deo logike na ulazu flip-flopa; resetuje flip-flop tek na prvoj sledećoj rastućoj ivici signala takta.

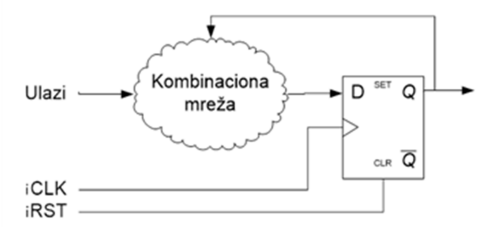
Simbol ove komponente je sledeći:



Sekvencijalna mreža

Registar zajedno sa kombinacionom mrežom na njegovom ulazu spada u tim digitalnih sistema koji se zove **sekvencijalna mreža**. Zbog načina na koji smo realizovali registar, ova komponenta sadrži interno stanje, koje je jednako sačuvanoj vrednosti u registru. Kombinaciona mreža računa narednu vrednost koja će se upisati u registar na narednoj rastućoj ivici takta, a ta vrednost se računa kao funkcija trenutne vrednosti u registru i ulaza u sekvencijalnu mrežu.

Opšta struktura sekvencijalne mreže je data na sledećoj šemi.



Ponašanje sekvencijalne mreže je sledeće:

- Aktiviranjem reset signala, sekvencijalna mreža, odn. njen registar, se postavlja na početnu vrednost, što je najčešće 0.
- Na svakoj rastućoj ivici takta, registar menja vrednost na onu koju računa kombinaciona mreža, a ta vrednost je funkcija trenutne vrednosti registra i ulaza.
- Izlaz sekvencijalne mreže je trenutna vrednost registra.

Standardne sekvencijalne mreže

U nastavku sledi lista najvažnijih standardnih sekvencijalnih mreža koje ćemo koristiti kao komponente u složenijim sistemima. Za svaku sekvencijalnu mrežu je dat opis ponašanja, kao i VHDL opis te komponente pomoću sekvencijalnog procesa. U svim primerima je korišten asinhroni reset. Ponašanje koje je opisano kod svake komponente predstavlja opis promene vrednosti te komponente na svakoj rastućoj ivici signala takta.

Registar sa dozvolom upisa

Ponašanje: registar preuzima vrednost sa ulaza ukoliko je signal dozvole aktivan (na vrednosti 1), u suprotnom ne menja vrednost.

Ulazi: signal vrednosti koja se upisuje (iD) i jednobitni signal dozvole (iEN).

VHDL opis:

```
process (iCLK, iRST) begin
    if (iRST = '1') then                        -- opis reseta registra (asinhroni reset)
        sREG <= (others => '0');                    -- način da opišemo nulu koliko god bita imao signal
    elsif (iCLK'event and iCLK = '1') then      -- opis promene registra na svakoj rastućoj ivici takta
        if (iEN = '1') then                         -- signal dozvole
            sREG <= iD;
        end if;
    end if;
end process;
```

Pomerački registar

Ponašanje: registar se pomera prema nekom pravilu pomeranja na svakoj rastućoj ivici takta; može se implementirati bilo koje pomeranje za proizvoljan broj mesta.

Ulazi: nema ulaza (u osnovnoj realizaciji).

Varijante ponašanja (omogućene posebnim ulazima; ovde su navedene neke moguće varijante):

- **Dozvola pomeranja** (iEN): ukoliko je aktiviran signal dozvole, registar se pomera, u suprotnom se ne pomera.
- **Paralelni upis vrednosti** (iLOAD - dozvola upisa, iD - vrednost koja se upisuje): ukoliko je aktiviran signal iLOAD, u registar se upisuje vrednost iD, a u suprotnom se ne upisuje.
- **Smer pomeranja** (iDIR): može da definiše da li se registar pomera u levo ili u desno.
- **Tip pomeranja** (iTYPE): može da definiše da li se registar pomera logički ili aritmetički.

Pomerački registar može da podrži i više varijanti, a prioritet upravljačkih signala definišemo ugnježdenom IF strukturom u procesu - najprioritetniji upravljački signala se stavlja u spoljni IF, manje prioritetan u unutrašnji.

VHDL opis - primer 8-bitnog pomeračkog registra (logičko pomeranje u levo za 1 mesto) sa dozvolom pomeranja i paralelnim upisom vrednosti; dozvola je prioritetnija (što znači da, ako je dozvola isključena, registar se ne menja bez obzira da li je signal iLOAD uključen ili ne):

```
process (iCLK, iRST) begin
    if (iRST = '1') then                        -- opis reseta registra (asinhroni reset)
        sREG <= (others => '0');                    -- način da opišemo nulu koliko god bita imao signal
    elsif (iCLK'event and iCLK = '1') then      -- opis promene registra na svakoj rastućoj ivici takta
        if (iEN = '1') then                         -- ako je dozvola uključena, promena registra je dozvoljena
            if (iLOAD = '1') then                       -- ako je iLOAD uključen
                sREG <= iD;                                 -- upisuje se vrednost sa ulaza
            else                                        -- u suprotnom
                sREG <= sREG(6 downto 0) & '0';             -- registar se pomera logički za 1 mesto u levo
            end if;
        end if;
    end if;
end process;
```

Brojač

Ponašanje: registar se uvećava za konstantnu vrednost (na primer, +1) na svakoj rastućoj ivici takta.

Ulazi: nema ulaza (u osnovnoj realizaciji).

Varijante ponašanja (omogućene posebnim ulazima ili dodatnim ograničenjima u opisu; ovde su navedene neke moguće varijante):

- **Dozvola brojanja** (iEN): ukoliko je aktiviran signal dozvole, registar se uvećava, u suprotnom se ne menja.
- **Paralelni upis vrednosti** (iLOAD - dozvola upisa, iD - vrednost koja se upisuje): ukoliko je aktiviran signal iLOAD, u registar se upisuje vrednost iD, a u suprotnom se ne upisuje.
- **Moduo brojanja**: registru se definiše najveća moguća vrednost, posle koje se više ne uvećava, nego se vraća na nulu.
- **Smer brojanja** (iDIR): može da definiše da li se registar uvećava ili umanjuje za datu konstantnu vrednost.
- **Korak brojanja** (iSTEP): može da definiše promenljiv korak uvećavanja ili umanjivanja, kao ulaznu vrednost.

Veoma često se brojač realizuje **zajedno sa komparatorom** koji proverava u kom se stanju nalazi brojač i signalizira kada se brojač pojavi u nekom stanju, najčešće kada brojač ima vrednost 0. Ovakav signal nam može biti koristan u sistemu kada brojač služi da broji neke događaje u sistemu (npr. protok vremena, prijem nekih podataka i sl.) i kada želimo da dobijemo signal kada se desi određeni broj tih događaja (npr. proteklo je određeno vreme ili smo primili određen broj bita podataka). Takav signal se zove ***terminal count*** i često naziva sTC.

VHDL opis - primer 8-bit brojača sa signalom dozvole i mogućnošću paralelnog upisa. Brojač se uvećava za 1 po modulu 200 (dakle, ima ukupno 200 stanja), ali pomoću signala iLOAD imamo način da neku spoljnu vrednost upišemo u registar, odakle će brojanje da se nastavi. Dozvola je prioritetnija od paralelnog upisa. Uz brojač realizovan je i komparator koji signalizira svaki put kada se brojač vrati na 0, što znači da je prošao ciklus od 200 uvećanja.

```
process (iCLK, iRST) begin
    if (iRST = '1') then                        -- opis reseta registra (asinhroni reset)
        sREG <= (others => '0');                    -- način da opišemo nulu koliko god bita imao signal
    elsif (iCLK'event and iCLK = '1') then      -- opis promene registra na svakoj rastućoj ivici takta
        if (iEN = '1') then                         -- ako je dozvola uključena, promena registra je dozvoljena
            if (iLOAD = '1') then                       -- ako je iLOAD uključen
                sREG <= iD;                                 -- upisuje se vrednost sa ulaza
            else                                        -- u suprotnom
                if (sREG = 199) then                        -- registar se uvećava za 1 po modulu 200
                    sREG <= (others => '0');                    -- ako je trenutna vrednost 199, naredna treba biti 0
                else                                        -- u suprotnom
                    sREG <= sREG + 1;                           -- naredna vrednost je trenutna vrednost plus 1
                end if;
            end if;
        end if;
    end if;
end process;

sTC <= '1' when sREG = 0 else '0';              -- komparator (terminal count)
```

VAŽNO: U VHDL opisu sekvencijalne komponente treba uvek opisati **kako se vrednost registra menja na svakoj rastućoj ivici takta kao <u>funkcija trenutne vrednosti registra i ulaza</u> te sekvencijalne komponente**. Moguća je, naravno, samo jedna promena, tako da taj deo procesa treba da **jednoznačno** opiše kako se računa nova vrednost registra. Različite varijante promene mogu da se pojave samo u međusobno isključivim granama nekih uslova.