POKAZNA VEŽBA 4
Sekvencijalne mreže

Potrebno predznanje

- Urađene pokazne vežbe: 1, 2 i 3
- Standardne sekvencijalne mreže – registri, brojači, pomerački registri

Šta će biti naučeno tokom izrade vežbe?

Nakon urađene vežbe, bićete u mogućnosti da:

- Razumete primenu memorijskih elemenata u digitalnim sistemima
- Projektujete digitalne sisteme sa memorijskim elementima
- Opišete sekvencijalne mreže pomoću VHDL jezika za opis digitalnih sistema
- Napišete test bench za verifikaciju sekvencijalnih mreža
- Projektujete sisteme sa više sekvencijalnih i kombinacionih mreža

Apstrakt i motivacija

Kombinacione mreže su u mogućnosti da izračunaju skoro sve što možete da zamislite, ukoliko imaju dovoljno prostora i vremena u kojima mogu da vrše to računanje. Međutim, kombinacione mreže same po sebi nisu previše korisne zbog toga što rezultat njihovog računanja odlazi u zaborav čim se vrednosti ulaza promene. Izlaz kombinacione mreže je uvek jednak Bulovoj funkciji trenutnih vrednosti na ulazu i čim se neka od vrednosti na ulazu promeni, menja se i izlaz. U ovoj vežbi naučićemo koje komponente možemo koristiti kako bi zapamtili vrednost koju je kombinaciona mreža izračunala. Ovi, tzv. memorijski elementi u digitalnim sistemima, omogućavaju projektovanje digitalnih sistema koji vrše računanja u nekom redosledu, prolazeći kroz stanja. Sekvencijalne mreže znatno uvećavaju primenjivost digitalnih sistema i omogućavaju projektovanje inteligentnih sistema koji postoje ne samo u prostoru (sastavljeni od logičkih kola koji računaju Bulove funkcije) nego i u vremenu, odn. prolaze kroz stanja koja možemo karakterisati kao „prethodno“, „trenutno“ i „sledeće“.

TEORIJSKE OSNOVE

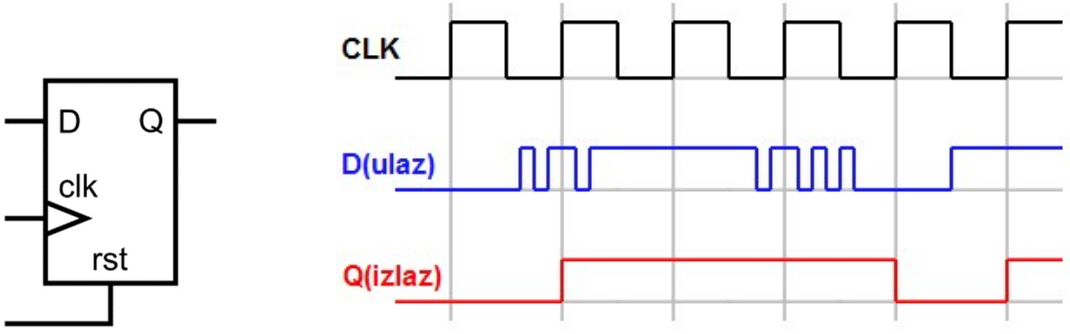
1. Memorijski elementi u digitalnim sistemima

Digitalni sistemi izvode računanja na prostoru Bulove algebre. Već smo utvrdili da sa svoje dve vrednosti i bar jednom operacijom, Bulova algebra predstavlja veoma moćan prostor u kome se mogu izvoditi veoma složena izračunavanja, pretpostavljajući da imamo dovoljno resursa (prostora i vremena) za projektovanje i implementaciju kombinacione mreže koja izvodi potrebno izračunavanje. Bez obzira na svoju snagu u izračunavanju, kombinacione mreže nisu u mogućnosti da iskoriste ono što su izračunale, tj. čim se ulaz kombinacione mreže promeni, menja se i izlaz, a ono što je prethodno bilo izračunato nepovratno se gubi. Bez mogućnosti pamćenja onoga što je prethodno bilo izračunato, nije moguće korišćenje vrednosti prethodne kalkulacije u narednoj, što znači da sistem nije u mogućnosti da vrši niz izračunavanja i nema vremensku komponentu – **sistem uvek ima trenutno stanje**, tj. izlaz je uvek funkcija trenutnih ulaza, i ne postoje „prethodno“ i „sledeće“ stanje sistema.

Kada bi digitalni sistemi bili implementirani isključivo kao kombinacione mreže, njihova primenjivost bi bila veoma ograničena – oni bi bili u stanju da izvode veoma složena izračunavanja, ali ne bi bili u stanju da inteligentno iskoriste ono što su izračunali. Ovakvi sistemi bi bili kao **kalkulator bez memorije** – mogli bi biti iskorišćeni da nešto izračunaju, ali oni sami ne bi mogli da koriste rezultat svog izračunavanja i ne bi imali nikakav vid veštačke inteligencije.

Kako bi digitalni sistemi imali mogućnost pamćenja rezultata svojih računanja, inženjeri u oblasti digitalne elektronike su napravili elektronska kola koja imaju sposobnost „pamćenja“ poslednje vrednosti koja im je prezentovana na ulazu. Ovo pamćenje se ispoljava na taj način što dato elektronsko kolo ima dva stabilna stanja, pa ih zato zovemo **bistabilne komponente**. Dva stabilna stanja ovog kola odgovaraju dvema Bulovim vrednostima (0 i 1), a promene između stanja se mogu vršiti isključivo dejstvom **pobudnih signala** (**takta** i **reseta**) na njihovim ulazima. Jedan od osnovnih predstavnika bistabilnih komponenti je **Flip-flop.**

Takt signal govori flip-flopu u kom vremenskom trenutku treba da promeni svoje stanje u ono koje se u tom trenutku nalazi na ulazu flip-flopa. Flip-flop menja stanje isključivo na **ivici** takt signala, **rastućoj** ili **opadajućoj**. Između ivica takt signala flip-flop ostaje u jednom od stabilnih stanja, onom koje odgovara Bulovoj vrednosti koja je kod prethodne ivice takt signala bila na ulazu. Izlaz flip-flopa je uvek jednak trenutnom stanju flip-flopa. U šemama, flip-flop se obeležava simbolom prikazanim na Slici 1-1.



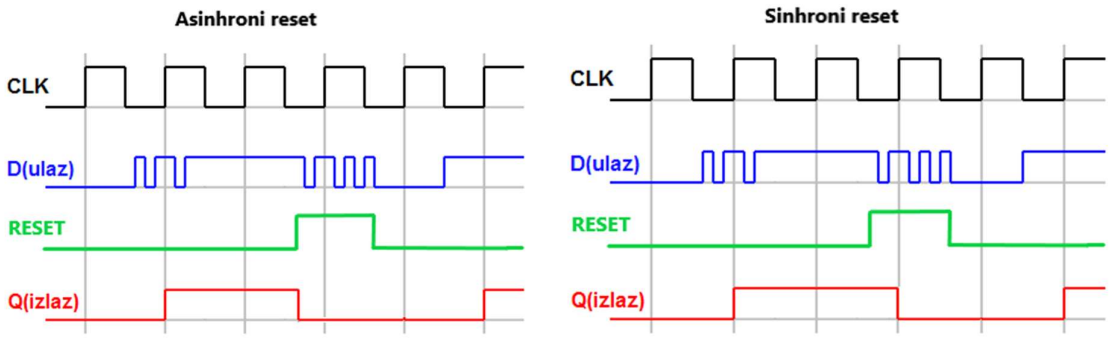
Slika 1-1. D flip-flop i prikaz izlaznog signala u odnosu na promenu takta i ulaza

Drugi pobudni signal karakterističan za flip-flop je **reset**. On se koristi za dovođenje flip-flopa u početno stanje. U zavisnosti od implementacije, bilo koje od dva stabilna stanja se može proglasiti za početno i nakon pritiska reseta, flip-flop se prebacuje u to izabrano početno stanje.

Razlikujemo dva tipa reset signala:

- **asinhroni reset** je reset koji nije sinhronizovan sa takt signalom, odn. kada god se postavi na aktivnu vrednost, flip-flopovi se vraćaju u početno stanje, bez obzira u kom vremenskom trenutku se to desilo,
- **sinhroni reset** je sinhronizovan sa taktom, tako da se vraćanje flip-flopova u početno stanje ne dešava do prve ivice takt signala nakon što je reset postao aktivan.

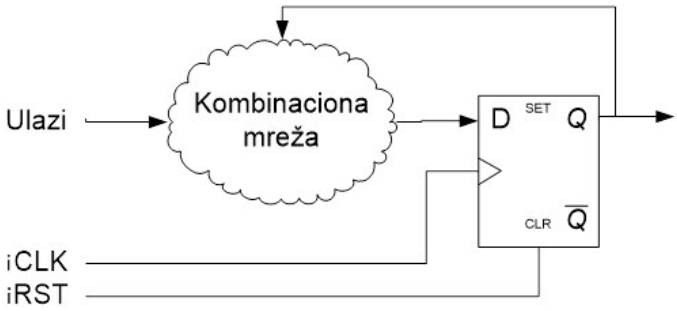
Na Slici 1-2 prikazani su vremenski dijagrami za asinhroni i sinhroni reset.



Slika 1-2. Vremenski dijagrami D flip-flopa sa asinhronim i sinhronim resetom

Dakle, flip-flop se može iskoristiti da zapamti **1 bit** informacije, tj. rezultat jedne Bulove funkcije. Pamćenje će trajati od jedne do druge ivice takt signala. Ovo konačno rešava problem koje su imale kombinacione mreže – sada smo u mogućnosti da rezultat izračunavanja iz kombinacione mreže zapamtimo neko vreme i ponovo iskoristimo u nekoj drugoj Bulovoj funkciji. Flip-flop nam omogućuje da u sistem uvedemo i vremensku komponentu, jer stanje flip-flopa možemo posmatrati kao stanje tog dela sistema i prelasci između stanja flip-flopova mogu da se posmatraju vremenski – jasno je šta je prethodno, trenutno i sledeće stanje.

Slika 1-3 prikazuje tipičnu vezu kombinacione mreže i flip-flopa koji pamti njen izlaz i predstavlja osnovnu komponentu **sekvencijalnih mreža**. Ukoliko postoji povratna veza iz flip-flopa ka kombinacionoj mreži, tada naredno stanje flip-flopa zavisi od prethodnog stanja. Na Slici 1-3 se vidi da pored standardnih ulaza (kojih može biti 0 ili više), svaka sekvencijalna komponenta mora da ima **dva podrazumevana ulaza**: **Takt** i **Reset**.



Slika 1-3. Osnovna komponenta sekvencijalnih mreža

Izlaz sekvencijalne komponente je **trenutno stanje flip-flopa**. Povratna sprega nije neophodna, tj. kombinaciona mreža ne mora da koristi rezultat prethodnog izračunavanja, ali je moguća. Kombinaciona mreža računa vrednost Bulove funkcije svih svojih ulaza i opciono prethodnog stanja flip-flopa.

Osnovne sekvencijalne komponente su:

1. **Registri**
2. **Brojači**
3. **Pomerački registri**

Sistem prikazan na Slici 1-3 se naziva i **jednobitni registar**. Ovakav registar se koristi unutar digitalnih sistema za pamćenje jednog bita informacije, odn. vrednosti jedne Bulove funkcije. Kako bi zapamtili više bita, koristi se više flip-flopova, npr. za pamćenje 64-bitne vrednosti koristimo 64-bitni registar koji se sastoji od 64 jednobitna registra (64 flip-flopa gde je svaki praćen kombinacionom logikom na ulazu). Višebitni registri se sastoje od n flip-flopova sa **zajedničkim takt signalom**.

Brojači su standardne sekvencijalne mreže koji se sastoje od jednog n-bitnog registra (koji se obično naziva brojački registar) i odgovarajućih aritmetičkih kombinacionih mreža koje služe za izračunavanje nove vrednosti brojačkog registra. Osnovni tip brojača je onaj koji u svakoj periodi takt signala **uvećava** ili **umanjuje** trenutni sadržaj brojačkog registra za jedan ili za neku drugu vrednost. Brojači koji prolaze kroz konačan broj jedinstvenih stanja i nakon poslednjeg stanja prelaze ponovo u prvo definisano stanje, nazivaju se **kružni brojači ili brojači po modulu n**. Brojači se najčešće koriste za brojanje dešavanja odgovarajućih događaja, generisanje periodičnih vremenskih intervala koji se mogu iskoristiti za kontrolu različitih zadataka u sistemu, merenje proteklog vremena između specifičnih događaja, itd.

PITANJE: Ako znamo da platforma MAX 1000 koristi radni takt na 12MHz, do koliko će brojač morati da izbroji da bi u sistemu prošla 1 sekunda?

Pomerački registri služe za pomeranje binarne informacije između susednih memorijskih elemenata ulevo ili udesno. Sastoje se od niza serijski povezanih flip-flopova, koji dele isti takt i gde je svaki izlaz jednog filp-flopa povezan sa D ulazom sledećeg flip-flopa.

U tabeli 1-1 je dat pregled osnovnih osobina kombinacionih i sekvencijalnih mreža:

	KOMBINACIONE MREŽE	SEKVENCIJALNE MREŽE
Obavezni ulazi?	NEMA	Takt i reset signali
Od čega zavisi izlaz?	Od **trenutnog** stanja ulaza	Od **trenutnog** ulaza i **memorisanog** stanja
Memorijski elementi:	NEMA	IMA
Povratna sprega (izlaz -> ulaz)	NEMA	MOŽE DA IMA
Brzina:	Brzina obrnuto proprocionalna dubini mreže	Brzina ograničena periodom takta
Osnovna kola:	Logička kola	Flip-flop
Primena:	Aritmetičke i logičke operacije	Pamćenje stanja
Primeri:	Dekoder, koder, multiplekser, demultiplekser, sabirač, oduzimač, ALU, pomerač, ...	Flip-flop, registar, brojač, pomerački registar, ...

2. VHDL opis sekvencijalnih mreža

Za razliku od kombinacionih mreža, koje se u VHDL mogu opisati na više načina, sekvencijalne mreže se mogu opisati isključivo pomoću **sekvencijalnih procesa**.

U **listi osetljivosti** sekvencijalnih procesa mogu da se nađu **SAMO SIGNALI TAKTA I RESETA**, pošto su to jedini signali čijom promenom može da se promeni stanje unutar flip-flopa.

U zavisnosti od tipa reseta, sekvencijalni proces ima različitu strukturu. U opštem slučaju, **svaki sekvencijalni proces ima jednu od naredne dve strukture**: sa SINHRONIM ili ASINHRONIM resetom.

Listing 2-1. Primer sekvencijalnog procesa sa ASINHRONIM resetom

```
process (iCLK, iRST) begin
    if (iRST = '1') then
        sREG <= <početna_vrednost>;
    elsif (iCLK'event and iCLK = '1') then
        -- opis ponašanja sekvencijalne mreže
    end if;
end process;
```

Listing 2-2. Primer sekvencijalnog procesa sa SINHRONIM resetom

```
process (iCLK) begin
    if (iCLK'event and iCLK = '1') then
        if (iRST = '1') then
            sREG <= <početna_vrednost>;
        else
            -- opis ponašanja sekvencijalne mreže
        end if;
    end if;
end process;
```

Svaka sekvencijalna komponenta u sistemu mora biti opisana kao **interni signal**, koji predstavlja flip-flop i izlazni signal iz flip-flopa. U prethodnim listinzima, ovaj signal nazvan je ***sREG***.

Kombinaciona mreža koja definiše naredno stanje flip-flopa je glavni deo opisa sekvencijalne komponente, jer ona definiše **ponašanje komponente**, odn. koje će biti naredno stanje komponente. **Opis kombinacione mreže može biti u okviru sekvencijalnog procesa (umesto komentara u listinzima) ili u posebnom kombinacionom opisu izvan sekvencijalnog procesa.**

PITANJE: Zašto ***if*** struktura u prethodnim listinzima nema podrazumevanu ***else*** granu?

Podrazumevanu ***else*** granu smo morali definisati kod kombinacionih mreža kako ne bi imali nedefinisane izlaze u sistemu. Kod sekvencijalnih mreža, prlikom resetovanja sistema postavljamo signale na početne vrednosti, tako da u ovom slučaju ne možemo imati nedefinisane vrednosti i podrazumevana ***else*** grana nije potrebna.

NAPOMENA: Reset signal treba biti u listi osetljivosti samo ako je ASINHRON. Zašto?

Kada koristimo asinhroni reset, ceo sistem se može resetovati bez obzira da li je takt signal na rastućoj ili opadajućoj ivici, tačnije u bilo kom trenutku. Zbog toga je reset signal neophodan u listi osetljivosti, kako bi se izvršavanje procesa dešavalo odmah kada se promeni vrednost reseta.

Primetite da sistemi u Listinzima 2-1 i 2-2 koriste reset signal koji je **aktivan na visokom naponskom nivou** (logičkoj vrednosti 1). Slično, reset signal može biti i **aktivan na niskom naponskom nivou**, a izbor jedne od ovih opcija zavisi od načina na koji je realizovano reset dugme u datom sistemu. Na našoj platformi, reset taster je realizovan tako da generiše vrednost 1 kada je pritisnut, pa ćemo mi uvek koristiti reset aktivan na visokom naponskom nivou.

Predlažemo da pratite strukturu opisa iz prethodnih listinga kada god opisujete sekvencijalne komponente u vašem sistemu. Prateći ovu strukturu, uvek ćete biti sigurni da ste sistem opisali na pravi način.

NAPOMENA: Umesto uslova za ispitivanje stanja takta **if (iCLK 'event and iCLK ='1')**, mogu se koristiti i ugrađene funkcije **rising_edge**(iCLK) i **falling_edge**(iCLK).

3. VHDL opis test bench-a za sekvencijalne mreže

Do sada smo na vežbama koristili samo **stimulus** proces u okviru test bench datoteke za zadavanje sekvence ulaznih vrednosti. Kada testiramo sekvencijalne mreže, neophodno je napraviti **poseban proces za takt**, kako bi promena 0 i 1 kod takt signala radila nezavisno od ostatka sistema**.** U nastavku je prikazan Listing 3-1 sa primerom procesa za proveru sekvencijalne mreže.

Listing 3-1. Takt proces i sekvenca reseta u opisu VHDL test bencha za sekvencijalne mreže

```
constant iCLK_period : time := 10 ns; -- iCLK_period je konstanta

begin  -- begin od architecture

iCLK_process: process
begin
      sCLK <= '0';
      wait for iCLK_period / 2;        -- iCLK_period je konstanta
      sCLK <= '1';
      wait for iCLK_period / 2;
end process;

stimulus: process
begin
      sRST <= '1';
      wait for <x.y> * iCLK_period;  -- x.y je necelobrojna vrednost
      sRST <= '0';

      -- sve dalje promene za n*iCLK_period, n = {1, 2, ..., N}

      wait;
end process;
```

Između ključnih reči ***architecture*** i ***begin*** potrebno je definisati konstantu za period takta. Za potrebe simulacije to može biti bilo koja vrednost, a u primeru smo period postavili na 10 ns. Nakon toga sledi proces za takt, u kome se naizmenično smenjuju vrednosti 0 i 1.

TAKT SIGNAL TREBA MENJATI ISKLJUČIVO U POSEBNOM PROCESU ZA TAKT!

Stimulus proces u test bench datoteci **prvo treba da resetuje sistem**, a nakon toga definiše niz ulaza kojim želite da proverite vaš sistem. **Reset signal treba biti zadržan duže od jednog perioda takta** kako bi se resetovale sve sinhrone komponente. Preporučljivo je da reset signal prekinete negde između ivica takt signala, jer promene ulaza koje se dešavaju u istom trenutku kao i ivice takt signala mogu da dovedu do nedefinisanog ponašanja sistema. Iz tog razloga ćemo koristiti **NECELOBROJNU** vrednost za period nakon reseta.

U Tabeli 3-1 je dat pregled načina opisivanja kombinacionih i sekvencijalnih mreža:

	KOMBINACIONE MREŽE	SEKVENCIJALNE MREŽE
Kako se opisuju u VHDL-u?	• Logička kola • Dodela vrednosti • Uslovna dodela vrednosti • Kombinacioni procesi: **IF** i **CASE**	• Sekvencijalni procesi
Lista osetljivosti procesa	**Svi ulazi** koji utiču na promenu izlaza u procesu	**TAKT** i/ili **RESET** signali
Takt signal:	-	UVEK
Reset signal:	-	Samo kada je **asinhroni** reset
Test Bench	Stimulus process	TAKT process + Stimulus process

Tabela 3-1. Pregled načina opisivanja kombinacionih i sekvencijalnih mreža

ZADACI

1. 8-bitni registar sa dozvolom upisa

Vreme je da implementiramo naš prvi sekvencijalni sistem! Kao prvi primer, opisaćemo **8-bitni registar** sa dozvolom upisa. Registar treba da čuva vrednost sa svog ulaza ukoliko je signal dozvole **iWE** aktivan (na vrednosti 1), u suprotnom čuva staru vrednost. U tabeli su navedeni svi prolazi sistema:

Tabela 1-1. Prolazi registra

Prolaz	Smer	Funkcija
iCLK	in	signal takta
iRST	in	**sinhroni reset**, aktivan na visokom naponskom nivou
iD [7:0]	in	ulazni podatak
iWE	in	signal dozvole upisa
oQ [7:0]	out	vrednost registra

Sistem testirati i simulirati u ModelSim-Altera alatu sa sledećim testnim slučajevima, a zatim i na platformi:

1. Resetovati sistem 2.25 perioda takta
2. Postaviti ulazni signal iD na vrednost 5 na 2 perioda takta, ali tako da je signal dozvole upisa neaktivan
3. Aktivirati signal dozvole upisa ukupno 2 perioda takta, a zatim promeniti vrednost ulaznog signala iD na -1 i deaktivirati signal dozvole upisa na 1 period takta.
4. Ponovo aktivirati signal dozvole upisa ukupno 2 perioda takta, a zatim deaktivirati 1 period takta.
5. Postaviti vrednost ulaznog signala iD na vrednost 15 ukupno 2 perioda takta.
6. Resetovati ceo sistem 2 perioda takta i ponovo pokrenuti sistem nakon toga.
7. Aktivirati signal dozvole upisa 2 perioda takta, a zatim deaktivirati.

Tabela 1-2. Pinovi za povezivanje komponente registar

Prolazi	Smer	Komponenta	Location	I/O standard
iCLK	input	I_CLK	PIN_H6	3.3-V LVTTL
iRST	input	I_RST	PIN_F1	3.3 V Schmitt Trigger
iD [8:0]	input	I_SW[7:0]	K1, K2, N2, N3, M1, M, L3, M3	3.3 V Schmitt Trigger
iWE	input	I_PB_CENTER	PIN_C2	3.3 V Schmitt Trigger
oQ [8:0]	output	oLED[7:0]	D8, C10, C9, B10, A10, A11, A9, A8	3.3-V LVTTL

U nastavku je opisan entitet registra i njegova arhitektura, kao i test bench sistema.

```
entity Registar is
     port (
          iCLK  : in  std_logic;
          iRST  : in  std_logic;
          iD    : in  std_logic_vector (7 downto 0);
          iWE   : in  std_logic;
          oQ    : out std_logic_vector (7 downto 0)
     );
end Registar;
```

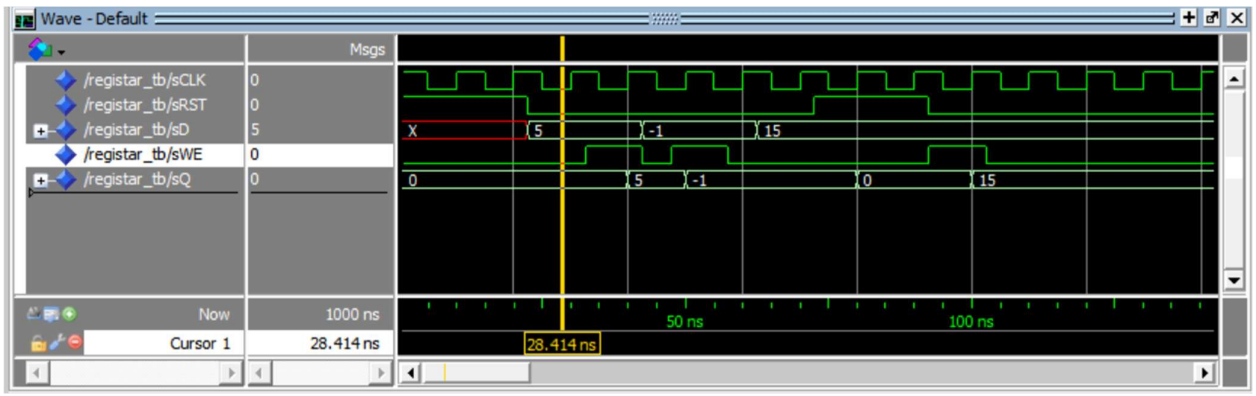
```
architecture Behavioral of Registar is
    signal sREG : std_logic_vector(7 downto 0);
begin
    process (iCLK) begin      -- sinhroni reset
        if (iCLK'event and iCLK = '1') then
            if (iRST = '1') then
                  sREG <= (others => '0');      -- početna vrednost je 0
            else
                  if (iWE = '1') then     -- ako je aktivan signal dozvole upisa
                        sREG <= iD;       -- upis u registar
                  end if;
            end if;
        end if;
    end process;

    oQ <= sREG;

end Behavioral;
```

U nastavku je opisan test bench tj. stimulus proces sistema:

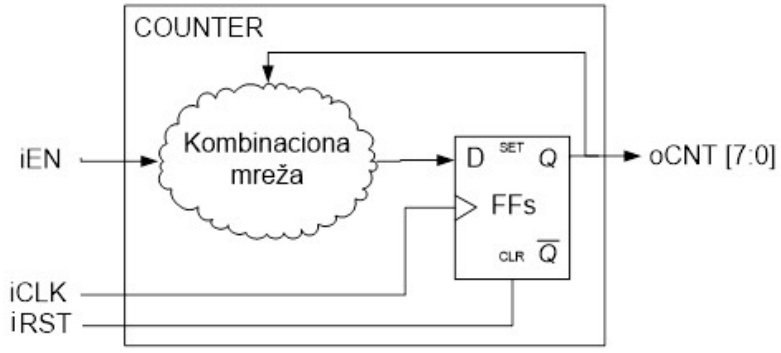
```
   stimulus : process
      begin
            -- 1. resetovanje sistema 2.25 perioda takta (NECELOBROJNA VREDNOST!)
            sRST <= '1';
            wait for 2.25* iCLK_PERIOD;
            sRST <= '0';
            -- 2. Signal iD = 5, 2 perioda + iWE = 0
            sD <= "00000101";
            wait for 2 * iCLK_PERIOD;
            -- 3. Aktivirati iWE 2 perioda, iD na -1 i deaktivirati iWE.
            sWE <= '1';
            wait for 2 * iCLK_PERIOD;
            sD <= "11111111";
            sWE <= '0';
            wait for iCLK_PERIOD;
            -- 4. Ponovo aktivirati iWE 2 perioda, a zatim deaktivirati.
            sWE <= '1';
            wait for 2 * iCLK_PERIOD;
            sWE <= '0';
            wait for 1 * iCLK_PERIOD;
            -- 5. Postaviti iD na vrednost 15 ukupno 2 perioda.
            sD <= "00001111";
            wait for 2 * iCLK_PERIOD;
            -- 6. Resetovati sistem 2 perioda takta i ponovo pokrenuti sistem
            sRST <= '1';
            wait for 2* iCLK_PERIOD;
            sRST <= '0';
            -- 7. Aktivirati signal dozvole upisa 2 perioda, a zatim deaktivirati
            sWE <= '1';
            wait for 2 * iCLK_PERIOD;
            sWE <= '0';
            wait;
```



Slika 1-1. Registar sa dozvolom upisa

2. 8-bitni brojač sa dozvolom brojanja - simulacija

U VHDL-u opisati 8-bitni brojač sa dozvolom brojanja, koji treba da uvećava vrednost za 1 na svakoj rastućoj ivici takt signala ukoliko je signal dozvole brojanja aktivan (Slika 2-1). Uzeti da je aktivna vrednost dozvole brojanja 1. Kada je brojač izbrojao do svoje maksimalne vrednosti, potrebno je postaviti interni **TERMINALNI SIGNAL** sTC na vrednost 1, dok za sve ostale vrednosti brojača ovaj signal treba da ima vrednost 0.



Slika 2-1. Brojač sa dozvolom brojanja

Sistem ima sledeće prolaze:

- Ulazi: iCLK, iRST (asinhron), iEN
- Izlazi: oCNT

Simulirati sledeće ponašanje sistema u ModelSim alatu:

- Resetovati sistem ukupno 5.25 perioda takta
- Brojač ne broji 5 perioda takta
- Brojač broji do 15
- Brojač zadržava svoju vrednost narednih 5 perioda takta
- Brojač nastavlja da broji do 50, a zatim se resetuje sistem 2 perioda takta i nastavlja brojanje do kraja trajanja simulacije

PITANJE: Koja je maksimalna vrednost do koje može da broji ovaj brojač i zašto?

DODATNI ZADATAK: Izmeniti brojač tako da radi **po modulu** vašeg broja indeksa. Npr. ako vam je broj indeksa 19, vaš brojač mora da broji od 0 do 18 i da nakon toga počinje opet da broji od 0.

3. Pomerački registar

Sada ćemo implementirati 8-bitni pomerački registar, Slika 3-1.

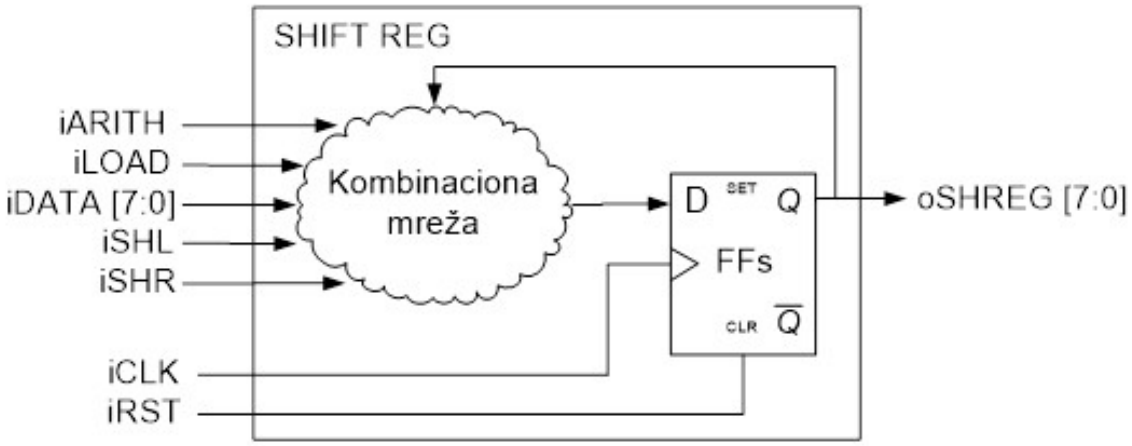
Sistem ima sledeće prolaze:

- Ulazi: iCLK, iRST (asinhron), iDATA[7:0], iLOAD, iARITH, iSHL, iSHR
- Izlazi: oSHREG

Pomerački registar treba da se ponaša na sledeći način:

- Ako je ulaz **iLOAD** aktivan (vrednost 1), registar treba dobiti vrednost sa ulaza iDATA,
- U suprotnom:
  - Ako je ulaz iSHL aktivan, a iSHR neaktivan – registar treba da pomeri vrednost za 1 bit ulevo,
  - Ako je ulaz iSHR aktivan, a iSHL neaktivan – registar treba da pomeri vrednost za 1 bit udesno,
  - Ako su ova dva ulaza na istoj vrednosti, registar ne menja vrednost.

Ulaz **iARITH** definiše tip pomeranja. Ukoliko je aktivan (vrednost 1), pomeranje treba biti aritmetičko, a ukoliko nije aktivan treba biti logičko. Uzeti da je predstava brojeva u formatu drugog komplementa.



Slika 3-1. Pomerački registar

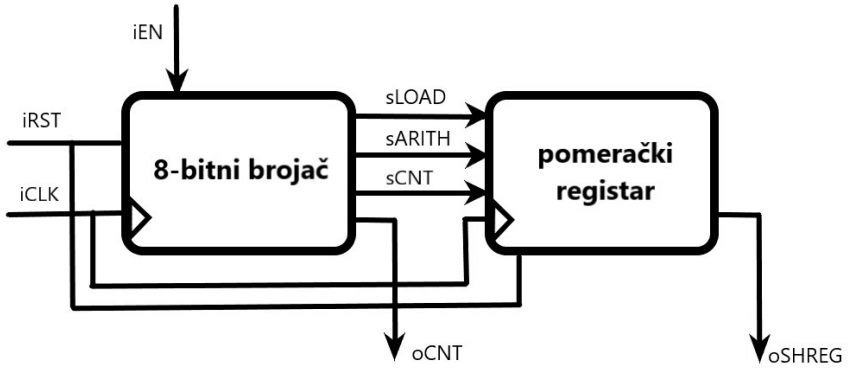
Simulirati ponašanje sistema u ModelSim alatu tako da se u simulaciji vide sve vrste pomeranja: logičko ulevo i udesno, kao i aritmetičko ulevo i udesno.

NAPOMENA: Voditi računa da ulazne vrednosti ne pomerite previše puta, kako svih 8 bita ne bi postale **NULE**!

4. Složeni sekvencijalni sistem

Konačno, iskombinovaćemo prethodno opisan brojač i pomerački registar u složenijem sistemu.
Sistem ima sledeće prolaze:

- Ulazi: iCLK, iRST(asinhron), iEN
- Izlazi: oCNT, oSHREG



Slika 4-1. Složeni sekvencijalni sistem

Potrebno je simulirati sistem koji radi na sledeći način:

- Ukoliko brojač ima vrednost 8 ili 128 (decimalno), pomerački registar treba da učita vrednost iz brojača,
- Ukoliko brojač ima ostale vrednosti:
  - Ako je ta vrednost veća od 128 pomerač treba svoj sadržaj da pomera aritmetički udesno,
  - Ako je ta vrednost manja od 128 pomerač treba svoj sadržaj da pomera logički ulevo.
- Simulirati digitalni sistem i proveriti rezultate u simulacionom dijagramu.

ZAKLJUČAK

Ova vežba je uvela pojam sekvencijalnih digitalnih sistema i pokazala osnovne sekvencijalne komponente i njihov opis u VHDL jeziku. Sada poznajete oba glavna tipa digitalnih sistema – kombinacione i sekvencijalne mreže. Njihovom kombinacijom se projektuju svi složeniji digitalni sistemi. U nastavku predmeta neće biti naučeno ništa konceptualno novo, svaki digitalni sistem je sastavljen od ova dva tipa komponenti. U ostatku vežbi naučićete kako da primenite ova dva osnovna tipa u projektovanju sistema koji rade određenu funkciju, kao što su automati, aritmetička kola, upravljačka kola i kako da povezujete više jednostavnijih sistema u jedan složen koristeći modularnost u opisu. No, osnovno znanje ste upravo stekli i koristeći do sada naučeno možete napraviti proizvoljno složen digitalni sistem.