POKAZNA VEŽBA 3
Sekvencijalne mreže

Potrebno predznanje

- Urađena pokazna vežba 2
- 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
- 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 da 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 karakteristati 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 prezenotava 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). Zbog svog karakterističnog ponašanja, odn. prelaženja između dva stabilna stanja, ova komponenta je nazvana **flip-flop**.

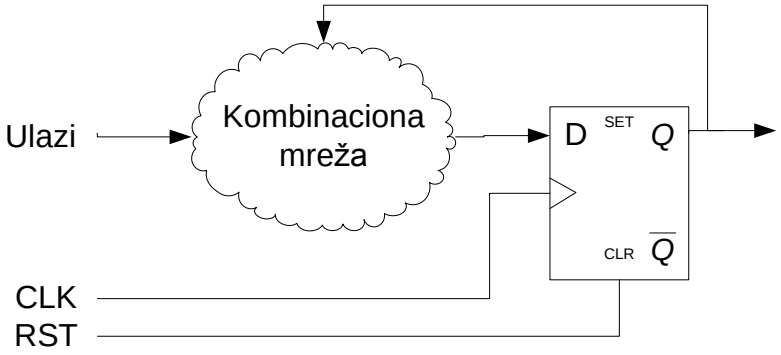
Flip-flop (FF) je osetljiv na specijalan signal koji nazivamo **takt** (clock). Ovaj signal govori flip-flopu u kom vremenskom trenutku treba da promeni svoje stanje u ono koje se u tom trenutku nalazi na ulazu flip-flopa. Flip-flop menja stanje isključivo na **ivici** takt signala, rastućoj ili opadajućoj. 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.

Drugi specijalni 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.

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 ponovno iskoristimo u nekoj drugoj Bulovoj funkciji. Flip-flop nam omogućuje da u naš 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.

U šemama, flip-flop se obeležava simbolom prikazanim na slici 1-1. Ova slika prikazuje tipičnu vezu kombinacione mreže i flip-flopa koji pamti njen izlaz. Ukoliko postoji povratna veza iz flip-flopa ka kombinacionoj

mreži, tada naredno stanje flip-flopa zavisi od prethodnog stanja, što lepo oslikava vremenski redosled rada našeg sistema.



Slika 1-1. Osnovna komponenta sekvencijalnih mreža

Sistem prikazan na slici 1-1 predstavlja osnovnu komponentu **sekvencijalnih mreža**. Svaka sekvencijalna komponenta u sistemu ima dva podrazumevana ulaza – takt i reset i nula ili više ostalih ulaza. 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.

Sistem prikazan na slici 1-1 se naziva **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).

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 ovih 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.

Razlikujemo dva tipa reset signala:

- **asinhroni reset** je reset koji nije sinhron 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 sinhron sa taktom, tako da se vraćanje flip-flopova u početno stanje ne dešava do prve ivice takt signala nakon što je reset postao aktivan.

Reset signal treba biti u listi osetljivosti samo ako je asinhron.

U zavisnosti od tipa reseta, sekvencijalni proces ima različitu strukturu. U opštem slučaju, svaki sekvencijalni proces ima jednu od naredne dve strukture. Listing 2-1 prikazuje strukturu sekvencijalnog procesa sa asinhronim resetom, a listing 2-2 strukturu sekvencijalnog procesa sa sinhronim resetom.

Listing 2-1. Primer sekvencijalnog procesa sa asinhronim resetom

```
process (iCLK, inRST) begin
    if (inRST = '0') then
        sREG <= <pocetna_vrednost>;
    elsif (iCLK'event and iCLK = '1') then
        <opis_kombinacione_mreze>
    end if;
end process;
```

Listing 2-2. Primer sekvencijalnog procesa sa sinhronim resetom

```
process (iCLK) begin
    if (iCLK'event and iCLK = '1') then
        if (inRST = '0') then
            sREG <= <pocetna_vrednost>;
        else
            <opis_kombinacione_mreze>
        end if;
    end if;
end process;
```

Svaka sekvencijalna komponenta u sistemu mora biti opisana kao interni signal. Ovaj signal predstavlja flip-flop i izlazni signal iz flip-flopa. Signal treba biti tipa *std_logic* ukoliko je u pitanju opis jednog flip-flopa, odn. *std_logic_vector* ukoliko je u pitanju opis višebitnog registra. 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.

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.

Primetite da sistemi u listinzima 2-1 i 2-2 koriste reset signal koji je **aktivan na niskom naponskom nivou** (logičkoj vrednosti 0). Slično, reset signal može biti i **aktivan na visokom naponskom nivou**, a izbor jedne od ovih opcija najčešće zavisi od načina na koji je realizovano reset dugme u datom sistemu. Na našoj platformi, reset taster je realizovan tako da generiše vrednost 0 kada je pritisnut, pa ćemo mi uvek koristiti reset aktivan na 0.

VHDL konvencija nalaže da se jednobitni signali koji su aktivni na niskoj logičkoj vrednosti nazovu imenom koje ima prefiks *n* nakon prefiksa tipa signala (ulaz, izlaz ili interni signal). Zbog toga je naš reset nazvan *inRST*. Ova konvencija treba da se prati ne samo za reset signal, nego i za bilo koji drugi jednobitni signal koji radu u ovakvoj, tzv. obrnutoj logici.

3. Opis VHDL test bencha za sekvencijalne mreže

Xilinx ISE alat će za vas automatski formirati test bench koji u sebi već sadrži generisan takt signal. Sećate se da smo do sada brisali sve delove test bench-a koji u sebi sadrže „clock“? Sada to više ne treba raditi.

Stimulus proces 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.

Listing 3-1 prikazuje primer početka stimulus procesa za proveru sekvencijalne mreže.

Listing 3-1. Sekvenca reseta u opisu VHDL test bencha za sekvencijalne mreže

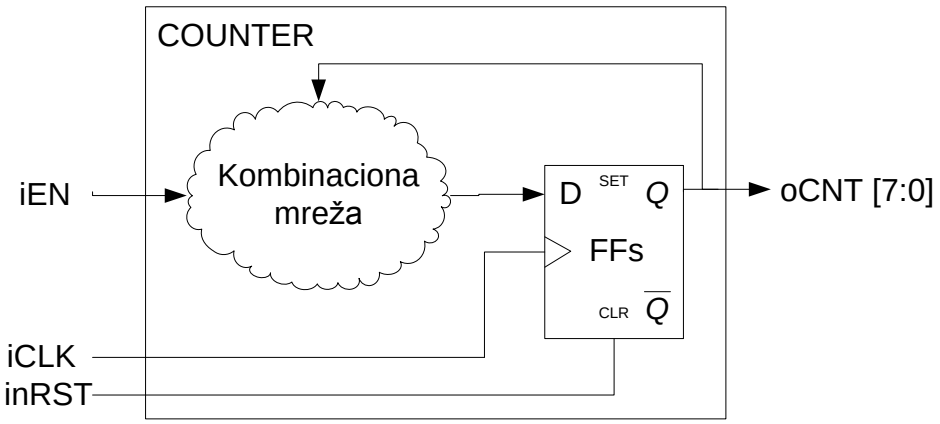
```
-- stimulus process --

process
begin

      inRST <= '0';

      wait for 5.25 * iCLK_period;    -- iCLK_period je konstanta

      inRST <= '1';

      --sekvenca ulaza ide ovde--

      wait;
end process;
```

Unutar stimulus procesa ne treba definisati takt signal. On je već generisan u svom procesu iznad ovog.

ZADACI

4. Brojač sa dozvolom brojanja

Vreme je da implementiramo naš prvi sekvencijalni sistem! Kao prvi primer, implementiraćemo 8-bitni brojač sa dozvolom brojanja, Slika 4-1. Brojač treba da uvećava vrednost za 1 na svakoj rastućoj ivici takt signala, ukoliko je signal dozvole brojanja aktivan. Uzeti da je aktivna vrednost dozvole 1.



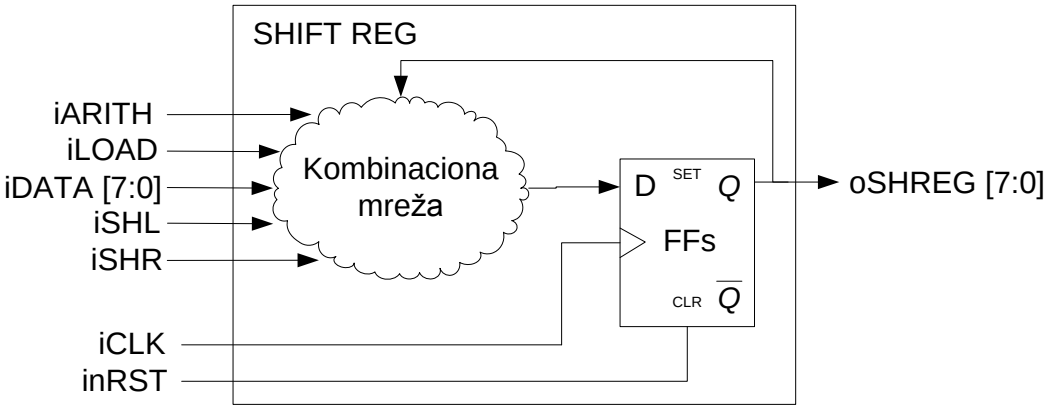
Slika 4-1. Brojač sa dozvolom brojanja

5. Pomerački registar

Sada ćemo implementirati 8-bitni pomerački registar, Slika 5-1. 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, registar treba da pomeri vrednost za 1 bit ulevo,
  - Ako je ulaz iSHR aktivan, 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 obliku drugog komplementa.



Slika 4-2. Pomerački registar

6. Nešto složeniji sekvencijalni sistem

Konačno, iskombinovaćemo prethodno opisan brojač i pomerački registar u složenijem sistemu. Implementirati 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 ili jednaka 128 pomerač treba svoj sadržaj da pomera aritmetički udesno,
  - Ako je ta vrednost manja od 128 pomerač treba svoj sadržaj da pomera logički ulevo.

Sistem ima sledeće prolaze:

- Ulazi: iCLK, inRST, iSHL, iSHR, iEN
- Izlazi: oSHREG

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.