POKAZNA VEŽBA 5

Složeni digitalni sistemi

Potrebno predznanje

- Urađena pokazna vežba 4
- Poznavanje principa modularnosti, apstrakcije i skrivanja informacija

Šta će biti naučeno tokom izrade vežbe?

Nakon urađene vežbe, bićete u mogućnosti da:

- Rukovodite različitim nivoima apstrakcije u vašem digitalnom sistemu – nakon implementiranja jednostavnije komponente, istu ćete koristiti kao crnu kutiju unutar složenijeg sistema
- Primenite osnovne principe projektovanja složenih sistema – modularnost, apstrakciju i skrivanje informacija
- Opišete složeni digitalni sistem u VHDL jeziku koristeći instanciranje modula
- Simulirate složeni digitalni sistem.

Apstrakt i motivacija

Projektovanje složenih digitalnih sistema bi bilo nemoguće kada bi ih morali projektovati u celini na nivou digitalnih logičkih kola i flip-flopova. Standardne kombinacione i sekvencijalne komponente nam pomažu da smanjimo broj komponenti sistema, a uradimo istu funkciju, npr. umesto da vodimo računa o N logičkih kola, vodimo računa o jednom multiplekseru. U ovoj vežbi naučićete da enkapsulirate vaš sistem kao crnu kutiju i iskoristite ga kao komponentu složenijeg sistema, opisanom u drugoj VHDL datoteci. Naučićete kako da rukovodite projektima sa više VHDL datoteka i kako da instancirate jedan sistem unutar drugog sa višim nivoom apstrakcije. Principi modularnosti, apstrakcije i skrivanja informacija, centralni u inženjerstvu složenih sistema, primenjivi su i na digitalne sisteme i omogućavaju projektovanje veoma složenih digitalnih sistema čiji dizajn bi trajao nekoliko redova veličine duže kako bi oni bili projektovani na najnižem nivou apstrakcije u celini.

TEORIJSKE OSNOVE

1. Opis složenih digitalnih sistema u VHDL jeziku

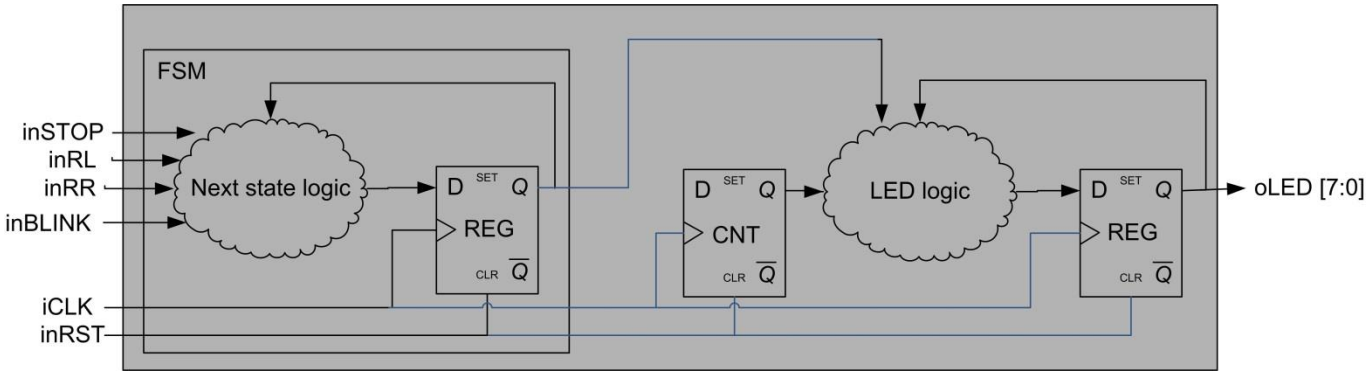
Osnovni principi prilikom projektovanja složenih sistema, ne samo u digitalnom svetu, su principi apstrakcije, modularnosti i skrivanja informacija. Principi modularnosti i apstrakcije znače da se neki deo sistema može „zatvoriti“ u crnu kutiju i koristiti samo posmatrajući njegove ulaze i izlaze (apstrakcija se odnosi na izdvajanje bitnih osobina datog dela sistema, odn. njegovog ponašanja). Kada koristimo deo sistema kao crnu kutiju, kažemo da se nalazimo na višem nivou apstrakcije, u odnosu na nivo u kome je projektovan sadržaj te crne kutije. Složeni sistemi se projektuju kombinovanjem ovih crnih kutija povezivanjem njihovih prolaza. Koristeći ovaj pristup, sistem se može posmatrati kroz nivoe apstrakcije – od najnižih (digitalna logička kola, flip-flopovi) do najviših (vrh hijerarhije sistema).

Princip skrivanja informacija znači da, nakon što deo sistema „zatvorimo“ u crnu kutiju, korišćenje tog dela sistema ne sme da zavisi od njegovog načina realizacije, odn. od onoga što se nalazi u crnoj kutiji. Razumevanje ovog principa zahteva veću pažnju – ovaj princip *ne zabranjuje* korisniku modula da poznaje arhitekturu sistema koji koristi. Naprotiv, princip olakšava rad korisniku, jer korisnik *ne treba da bude primoran* da poznaje arhitekturu sistema ako želi da ga iskoristi kao komponentu. Jedan od najčešćih razloga kašnjenja u projektima je potreba da se ulazi u tuđ sistem, razume njegov rad, modifikuje po potrebi i iskoristi u svom sistemu. **Ovo nikada ne bi trebao biti slučaj**! Nakon što je neki modul završen, niko ko ga koristi ne bi trebao biti primoran da razume njegovu unutrašnjost da bi ga koristio, a sve potrebne promene unutar njega treba da vrši autor modula. Ovaj princip obavezuje autora nekog modula da ga napravi potpunog, iskoristljivog isključivo putem svojih prolaza i specifikacije funkcionalnosti.

Pričajući u VHDL jeziku, kada završite projektovanje vašeg *entiteta (entity)*, on mora biti iskoristljiv kroz svoje *prolaze (ports)* i specifikaciju ponašanja za date ulaze. Korisnik modula, odn. projektant sistema na višem nivou apstrakcije, nikada ne treba biti primorana da razume *arhitekturu (architecture)* modula da bi ga koristila.

Ovi principi su potreban uslov da bi se projektovali složeni sistemi, pošto na visokim nivoima složenosti postaje nemoguće projektovati sistem vodeći računa o svakoj komponenti na najnižem nivou apstrakcije. Zamislite samo da se najnoviji procesori u celosti projektuju na nivou logičkih kola, koliko bi takvo projektovanje trajalo i da li bi uopšte bilo moguće? Mi smo već koristili ove principe u ranijim primerima, kada smo kombinacionu mrežu prikazali pomoću oblačića. Oblačić je u stvari crna kutija koja predstavlja skup logičkih kola koji realizuju funkciju koju mi želimo, no mi koristimo oblačić kao crnu kutiju ne vodeći računa o njegovom sadržaju, već samo posmatrajući njegove ulaze i izlaze.

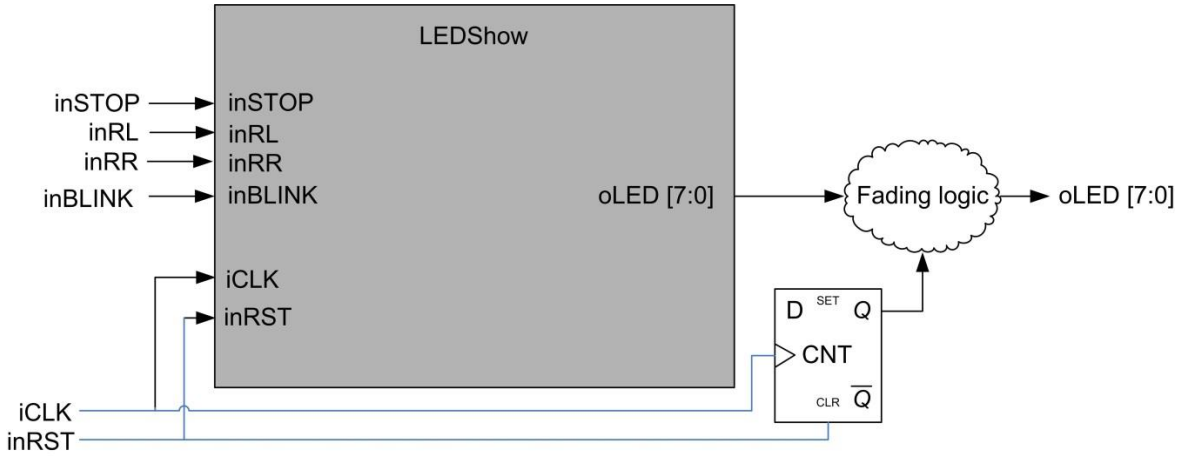
Slično možemo da uradimo i sa LED show sistemom iz prethodne vežbe. Kao što je prikazano na Slici 3-1, mi taj sistem možemo „zatvoriti“ u crnu kutiju i koristiti ga u složenijim sistemima samo preko njegovih ulaza/izlaza.



Slika 3-1. LED show sistem zatvoren u crnu kutiju

Nakon što pređemo na viši nivo apstrakcije, LED show sistem koristimo kao komponentu komunicirajući sa njom putem njenih ulaza/izlaza (odn. prolaza).

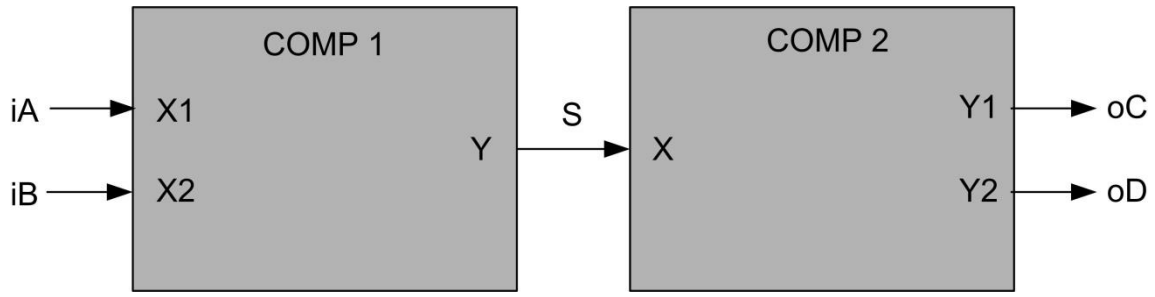
VHDL jezik i alat Xilinx ISE podržavaju ranije navedene principe prilikom projektovanja složenih digitalnih sistema omogućavajući definisanje više entiteta unutar jednog projekta i instanciranje jednog entiteta u drugom. Na primer, pretpostavimo da želimo da iskoristimo naš LED show sistem u složenijem sistemu u kome, sem LED show entiteta, želimo da postavimo još jedan brojač između izlaza LED show sistema i izlaza ka LED diodama. Na višem nivou apstrakcije, možemo da **instanciramo** komponentu LED show sistema i povežemo je na ostatak sistema, kompletirajući sistem kao na Slici 3-2. U terminologiji digitalnih sistema, opis sistema na najvišem nivou hijerarhije se naziva **top level**.



Slika 3-2. Top level složenog digitalnog sistema

Instanciranje komponenti se u VHDL jeziku vrši unutar opisa arhitekture sistema. Kako bi pokazali kako se vrši instanciranje komponenti, posmatrajmo za početak top level sistema na Slici 3-3 koji se sastoji od dve komponente COMP1 i COMP2. Neka komponenta COMP1 ima dva ulaza (X1, X2) i jedan izlaz (Y). Neka komponenta COMP2 ima jedan ulaz (X) i dva izlaza (Y1, Y2). Komponente treba povezati kao na Slici 3-3. Komponente COMP1 i COMP2 su definisane kao posebni entiteti, a njihovi opisi se nalaze u posebnim VHDL datotekama.

Listing 3-1 prikazuje arhitekturu najvišeg nivoa apstrakcije sistema gde se vidi način instanciranja komponenti u VHDL jeziku. Sistem ima dva ulaza (iA, iB) i dva izlaza (oC, oD) koji treba da se povežu da ulaze komponente COMP1 i izlaze komponente COMP2, respektivno. Izlaz komponente COMP1 povezati sa ulazom komponente COMP2. Komponente se međusobno vezuju žicama, odn. signalima u VHDL-u. U ovom primeru, neka to bude signal S.



Slika 3-3. Primer sistema sa dve komponente

Listing 3-1. Arhitektura sistema sa Slike 3-3

```
architecture Behavioral of MyTopLevel is

    component COMP1 is
        port ( X1 : in std_logic;
               X2 : in std_logic;
               Y : out std_logic );
    end component;

    component COMP2 is
        port ( X : in std_logic;
               Y1 : out std_logic;
               Y2 : out std_logic );
    end component;

    signal S : std_logic;

begin

    iCOMP1 : COMP1 port map (
        X1 => iA,
        X2 => iB,
        Y => S
    );

    iCOMP2 : COMP2 port map (
        X => S,
        Y1 => oC,
        Y2 => oD
    );

end Behavioral;
```

Instanciranje komponente zahteva sledeće tri akcije:

- treba dodati VHDL datoteku u kojoj je opisana komponenta u projekat (izborom **Project --> Add Source...**)
- unutar arhitekture opisa vrha hijerarhije, deklaracija komponente treba biti napisana bre ključne reči begin, ona je identična opisu entiteta u svemu sem u ključnoj reči na početku i kraju opisa
- Nakon ključne reči begin, komponenta se treba instancirati koristeći sintaksu iz Listinga 1-1. Prilikom navođenja veza, ime pre operatora obrnute dodele (**=>)** je naziv prolaza komponente, a ime nakon njega je naziv signala u vrhu hijerarhije koji se povezuje na odgovarajući prolaz.

Sve žice unutar vrha hijerarhije koje služe da bi se povezale dve komponente treba deklarisati kao interne signale.

ZADACI

2. Složeni LED Show

Opisati u VHDL-u top level sa Slike 3-2. Logika na izlazu prema LED diodama treba da guši signal, tako da:

- Prvih 200 ms, diode koje treba da budu upaljene, to budu sa četvrtinom intenziteta,
- Od 200 ms do 400 ms, diode budu upaljene polovinom intenziteta,
- Od 400 ms do 600 ms, diode budu upaljene celim intenzitetom,
- Od 600 ms do 800 ms, diode budu upaljene polovinom intenziteta,
- Od 800 ms 1000 ms, diode budu upaljene četvrtinom intenziteta.

Izlazni signal iz LEDShow modula definiše koje diode treba da budu upaljene, a logika na izlazu, pomoću brojača jedne sekunde, definiše intenzitet. Intenzitet se definiše tako što se dioda periodično pali/gasi sa faktorom ispune jednakom potrebnom procentu intenziteta osvetljenja.

ZAKLJUČAK

Ova vežba vam je pokazala kako da opišete složeni digitalni sistem u VHDL jeziku koristeći module i njihovo instanciranje. Projektovanje složenih sistema je nemoguće bez jasno definisanih nivoa apstrakcije i mogućnosti ponovnog iskorišćenja komponenti. U nastavku predmeta, projektovaćete sve složenije sisteme i uskoro ćete primetiti da bi bilo veoma mukotrpno razmišljati o svakom logičkom kolu unutar jednog takvog složenog sistema – broj informacija koje ljudski mozak može da procesira u jednom trenutku je veoma konačan. Apstrakcioni nivoi nam omogućavaju da problem podelimo i osvojimo deo po deo, taman na onom nivou na kom naš mozak može da ga obrađuje.