

POKAZNA VEŽBA 5

Automati sa konačnim brojem stanja

Potrebno predznanje

- Urađena pokazna vežba 4
- Teorija automata sa konačnim brojem stanja

Šta će biti naučeno tokom izrade vežbe?

Nakon urađene vežbe, bićete u mogućnosti da:

- Projektujete digitalni sistem koji je zasnovan na automatu sa konačnim brojem stanja definisanim funkcijom prelaza i funkcijom izlaza
- Opišete digitalni sistem zasnovan na automatu sa konačnim brojem stanja u VHDL jeziku
- Opišete VHDL test bench za automat
- Kombinujete automate sa ostalim kombinacionim i sekvencijalnim komponentama u digitalnom sistemu.

Apstrakt i motivacija

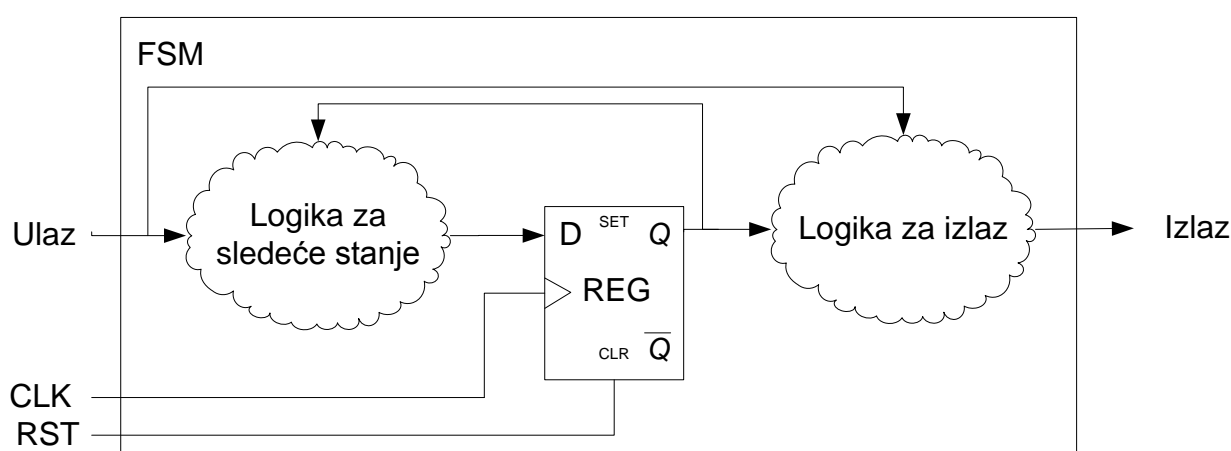
Veoma često digitalni sistem treba da izvršava sekvencu pre-definisanih zadataka. Da li je u pitanju žmigavac kod automobila, digitalni svetleći signali kao što je semafor ili sistem koji komunicira sa nekim drugim sistemom, sve ove operacije sadrže niz pre-definisanih koraka koji se trebaju izvršiti. Ovakvi sistemi imaju konačan broj stanja kroz koji treba da prolaze tokom svog životnog veka i u svakom stanju vrše neku operaciju. U teoriji digitalnih sistema, a i šire, ovim sistema je dato ime: automati sa konačnim brojem stanja (eng. *finite state machine* – *FSM*). FSM se nalazi u svakom nešto složenijem digitalnim sistemu. U ovoj vežbi naučićete da projektujete i implementirate sistem zasnovan na FSM-u, kao i da ga opišete u VHDL jeziku i simulirate.

TEORIJSKE OSNOVE

1. Projektovanje automata sa konačnim brojem stanja

U digitalnim sistema, sekvencijalne komponente su idealne za realizaciju automata sa konačnim brojem stanja. Vrednosti flip-floпова (registara) mogu predstavljati stanja automata, dok kombinaciona logika može da računa naredno stanje i izlaz automata. Ovo nas dovodi do jednog načina realizacije automata u digitalnom sistemu, koristeći sledeće komponente:

- Registar – memoriše **trenutno stanje** automata,
- Kombinaciona mreža koja računa **naredno stanje** automata na osnovu trenutnog stanja i ulaza,
- Kombinaciona mreža koja računa **izlaz** automata na osnovu trenutnog stanja i ulaza.



Slika 1-1. Automat

U zavisnosti od čega zavisi izlaz automata, razlikujemo:

- **Mealy**-eve automate, kod kojih izlaz zavisi od ulaza i trenutnog stanja automata,
- **Moore**-ove automate, kod kojih izlaz zavisi samo od trenutnog stanja automata.

U teoriji, automati se definišu pomoću sledećih šest veličina:

- **Skup vrednosti ulaza**, predstavljen svim vrednostima koje mogu imati ulazi automata,
- **Skup vrednosti izlaza**, predstavljen svim vrednostima koje mogu imati izlazi automata,
- **Skup vrednosti stanja**, predstavljen svim vrednostima koje može imati registar,
- **Početno stanje**, predstavljeno vrednošću registra u resetu,
- **Funkcija prelaza**, koja definiše kako se računa naredno stanje na osnovu trenutnog stanja i ulaza,
- **Funkcija izlaza**, koja definiše kako se računa izlaz na osnovu trenutnog stanja i, eventualno, ulaza.

Funkcija prelaza automata se najčešće definiše pomoću **grafa prelaza stanja**. Primer će biti dat u narednom delu vežbe.

U VHDL jeziku, automati se mogu opisati iz dva dela:

- Sekvencijalni proces koji opisuje registar i prateću logiku računanja narednog stanja,
- Kombinatorna dodela ili proces koji opisuje računanje izlaza automata.

Listing 1-1 na narednoj strani daje opšti oblik opisa automata u VHDL jeziku. Za opis logike za računanje narednog stanja najpogodnije je koristiti CASE strukturu.

Listing 1-1. Automat u VHDL-u

```
process (iCLK, inRST) begin
    if (inRST = '0') then
        sSTATE <= <initialState>;
    elsif (iCLK'event and iCLK = '1') then
        <logika_za_racunanje_narednog_stanja>
    end if;
end process;

<uslovna_dodela_ili_proces_za_racunanje_izlaza>
```

Registar za memorisanje trenutnog stanja automata može da se realizuje kao interni signal tipa STD_LOGIC_VECTOR. No, tada stanja u kodu postaju nečitljiva, pošto imaju samo brojne vrednosti. Kao u programskim jezicima, VHDL za ovu svrhu omogućava da se definiše tip **enumeracije**. Vrednosti koje enumeracija dobije određuje alat za simulaciju/sintezu i nisu bitne za opis sistema. Listing 1-2 prikazuje kako se u VHDL-u opisuje tip enumeracije i definiše signal koji je tog tipa.

Listing 1-2. Deklaracija enumeracije u VHDL-u

```
architecture Behavioral of MySystem is

    type <typeName> is (<enumerationNames>);
    signal <signalName> : <typeName>;

begin
    ...
end architecture;
```

Tip enumeracije **ne treba koristiti za definisanje ulaza i izlaza automata** jer je enumeracija lokalno definisani tip, a ulazi i izlazi treba uvek da budu standardnog tipa kako bi sistemi koji komuniciraju sa našim sistemom preko tih ulaza i izlaza bili tipski kompatibilni.

Provera sistema sa automatima se može opisati VHDL testbenchom po sličnom principu kao i za opšte sekvencijalne mreže. U početku treba resetovati sistem, a nakon toga menjati vrednosti ulaza prema željenom redosledu provere automata. Idealno, provera bi trebala da obuhvati sve moguće prelaze stanja automata.

ZADACI

2. LED Show

Vreme je da implementiramo naš automat! Projektovaćemo sistem koji kontroliše LED diode na E2LP platformi zasnovan na automatu sa konačnim brojem stanja.

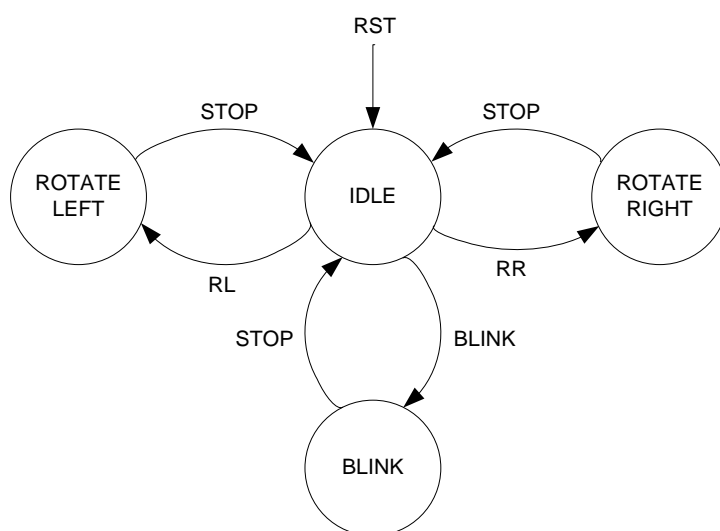
Stanja automata su: IDLE, ROTATE_LEFT, ROTATE_RIGHT, BLINK.

Početno stanje je IDLE.

Ulazi automata su: inSTOP, inRL, inRR, inBLINK.

Izlaz automata je: oLED.

Funkcija prelaza je data grafom na slici 2-1.

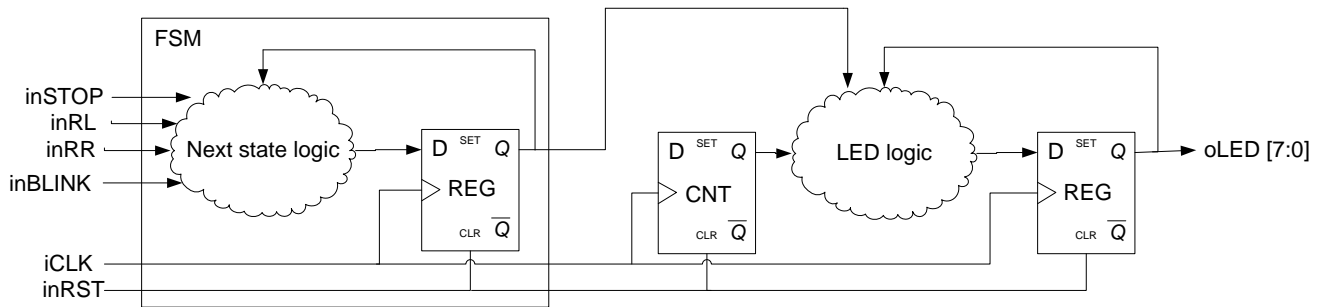


Slika 2-1. Graf prelaza stanja automata za LED show

Funkcija izlaza je definisana na sledeći način:

- U stanju IDLE, upaljena je samo dioda LED0.
- U stanju ROTATE_LEFT, jedna upaljena dioda rotira ulevo,
- U stanju ROTATE_RIGHT, jedna upaljena dioda rotira udesno,
- U stanju BLINK, diode se naizmenično pale/gase.

Blok dijagram sistema je dat na slici 2-2. Rotiranje i naizmenično paljenje i gašenje dioda treba da se dešavaju svaki sekund.



Slika 2-2. LED show sistem

3. Dodatak

Dopunićemo logiku na izlazu prema LED diodama gušenjem signala, 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.

Intenzitet se definiše tako što se dioda periodično pali/gasi sa faktorom ispune jednakom potrebnom procentu intenziteta osvetljenja.

ZAKLJUČAK

Upravo ste projektovali vaš prvi digitalni sistem zasnovan na automatu sa konačnim brojem stanja. Pored toga što ste naučili da u VHDL jeziku opišete i implementirate automat koji je unapred definisan, naučili ste kako da koristite automat kao komponentu unutar složenijeg digitalnog sistema. Automati su često korišteni u digitalnim sistemima zato što su korisni za izvršenje sekvence operacija u sistemu. U ostatku ovog predmeta, automate ćemo koristiti kao gradivni element procesora, u njegovom upravljačkom delu.