

Mašine stanja u VHDL-u

Vladimir Petrović, Strahinja Janković, Dragomir El Mezeni

Katedra za elektroniku
Elektrotehnički fakultet
Univerzitet u Beogradu

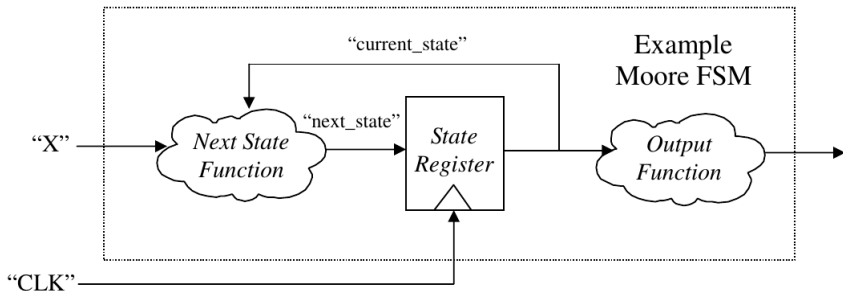
24. oktobar 2018.



- Kratak pregled
- What **not** to do
- Primeri



Mašine stanja



Tip podataka za stanja – *State Type*

```
type state_type is (init, active, wait);  
signal state_reg, next_state : state_type;
```



Registar stanja – *State Register*

```
state_transition: process (clk, reset)
begin
    if (reset = '1') then
        state_reg <= init;
    elsif (clk'event and clk = '1') then -- rising_edge(clk)
        state_reg <= next_state;
    end if;
end process;
```



Logika za sledeće stanje – *Next State Logic*

```
next_state_logic: process (state_reg)
begin
    case state_reg is
        when init =>
            ...
            next_state <= active;
        when active =>
            ...
            next_state <= wait;
        when wait =>
            ...
            next_state <= active;
    end case;
end process;
```



Logika za izlaze – *Output Logic*

```
output_logic: process (state_reg)
begin
    case state_reg is
        when init =>
            ...
        when active =>
            ...
        when wait =>
            ...
    end case;
end process;
```



Hint

Vrednosti signala se menjaju tek na kraju procesa.

```
sample_process: process (a)
begin
    s_out <= '0';
    case (a) is
        when a1 =>
        when a2 =>
        when a3 =>
            s_out <= '1';
        when a4 =>
    end case;
end process;
```



What **not** to do



- Ne stavljati sve moguće signale u listu osetljivosti već **samo one signale koji utiču na promene!**

```
wrong_process: process (signal1, signal2, ..., signaln)
```

```
...
```

```
right_process: process (signalImportant1, signalImportant2)
```

```
...
```



Komplikovani procesi

- Ne stavljati sve funkcionalnosti u jedan ogroman proces!
- Podeliti kod u više procesa.
- Time ćete poštedeti sebe raznih problema.



- Svaki **if** mora da ima svoj **else**.
- Svaki **case** mora da ima svoj **when others**.
- Izuzeci?
 - `if (clk'event and clk = '1')`



Gejtovanje takta

Ukoliko se ispituje stanje taktnog signala, onda samo to raditi u jednom if-u!

Neispravno:

```
if (clk'event and clk = '1' and signalA = '1') then
...

```

Ispravno:

```
if (clk'event and clk = '1') then
    if (signalA = '1') then
...

```



Sinhronizacija sa signalom takta

Izbegavati korišćenje 'event atributa ukoliko signal nije izveden iz signala takta.

```
if (signalX'event and signalX = '1') then
    ...
```

Ispravnije:

```
if (clk'event and clk = '1') then
    if (signalX_prev = '0' and signalX = '1') then
        ...
    else
        ...
    end if;
    signalX_prev <= signalX;
end if;
```



Korišćenje signala takta

- Ne stavljati u listu osetljivosti signal takta ako se on ne koristi u procesu!
- Ovo je slučaj kada se može desiti da se razlikuju simulacija i realizacija.

```
architecture Behavioral of counter_entity is
    signal counter: integer range 0 to N:=0;
begin
    process (clk, reset)
    begin
        if (reset='1') then
            counter<=0;
        else
            counter<=counter+1; --NEISPRAVNO
        end if;
    end process;
    ...
end Behavioral;
```



Korišćenje signala takta

- Signal takta koristiti samo na uzlaznu ili silaznu ivicu.

```
architecture Behavioral of counter_entity is
    signal counter: integer range 0 to N:=0;
begin
    process (clk, reset)
    begin
        if (reset='1') then
            counter<=0;
        else
            if (clk'event and clk = '1') then
                counter<=counter+1;--ISPRAVNO
            end if;
        end if;
    end process;
    ...
end Behavioral;
```



Pitanja za razmišljanje

- Diferencijator ivice
- Debaunsiranje signala
- Konverzija iz binarnog u BCD kod (LVE2)
- Blinkanje diode sa različitom periodom (perioda se bira pritiskom na tastere `btn_up` i `btn_down`)
- Neki *real life* primeri
 - Kuvanje
 - Termostat za električni kotao
 - Bilo kakve mašine stanja koje vam padnu na pamet...

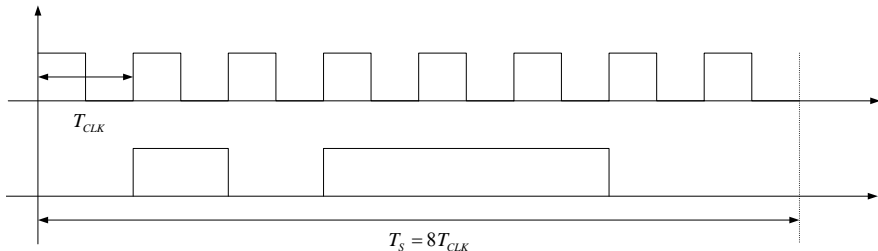


Primeri



Primer 1

Na slici je prikazan talasni oblik signala koji se periodično ponavlja sa periodom od $T_S = 8T_{CLK}$. Potrebno je opisati u VHDL-u mašinu stanja koji generiše posmatrani talasni oblik na svom izlazu. Na ulaz uređaja se dovodi signal takta periode T_{CLK} .



Primer 2

Mašina za prodavanje vode u flašama od 0,5l ima slotove za metalne novčiće u vrednostima od 5, 10 i 20 dinara. Mašina može da izda samo jednu flašu vode u jednom ciklusu i kusur od ubačenog novca do vrednosti 30 dinara koliko košta flaša vode. Za svaki slot u koji se ubacuju novčići postoji po jedan interni signal (n_5 , n_{10} i n_{20}), koji se postavlja na prvu nailazeću ivicu takta (uzlaznu ili silaznu) od momenta ubacivanja odgovarajućeg novčića. Signal ima vrednost logičke jedinice u trajanju periode signala takta i nakon toga se resetuje (sve to obavlja poseban kontroler koji nije potrebno realizovati). Smatrati da novčić trenutno prolazi kroz odgovarajući slot i završava u kasi mašine. Mašina izdaje vodu, odnosno kusur tako što postavlja signale $voda_out$, odnosno n_{5_out} i n_{10_out} na logičku jedinicu sa uzlaznom ivicom signala takta, pri čemu trajanje logičke jedinice iznosi jednu periodu signala takta. Mašina treba da ima signal reseta koji kada je aktivan $reset = 1$, zabranjuje rad mašine. Unutar mašine se generiše i signal takta clk . Zadatak: Nacrtati blok šemu sistema sa komponentama i FSM preko kojih će mašina biti realizovana. Napisati VHDL kod kojim se opisuje ova mašina. Simulacijom potvrditi rad kreiranog dizajna.

