Laboratorij za načrtovanje integriranih vezij

Univerza *v Ljubljani*
Fakulteta *za elektrotehniko*

Digitalni Elektronski Sistemi

Osnove jezika VHDL

5. del: komponente in testne strukture

Komponente

- Model vezja je sestavljen iz para entity-architecture
- Modele vezja lahko povezujemo med seboj, tako da v opis vezja vključimo vezja kot komponente
  - takšen princip uporabljamo pri risanju sheme vezja
- Strukturno načrtovanje
  - struktura (zgradba) vezja, ki temelji na povezavi komponent
- Hierarhična vezja
  - celotno vezje je sestavljeno iz komponent, ki so spet lahko zgrajene iz komponent...

Deklaracija komponent v povezovalni strukturi

- Komponente deklariramo v strukturi na začetku arhitekturnega stavka (pred begin)
- Navedemo ime komponente in zunanje priključke (stavek port)

```
architecture struktura of test is
  component reg is
    port ( d : in std_logic_vector(7 downto 0);
           clk, en : in std_logic;
           q : out std_logic_vector(7 downto 0) );
  end component;
begin
```

Vključevanje komponente v strukturo

- Komponento vključimo v vezje s stavkom port map
  - vsaki komponenti damo enolično oznako
  - v oklepaju navedemo povezave

```
begin
   r1: reg port map (d=>data, clk=>clk, en=>en1, q=>data_reg);
   r2: reg port map (d=>data, clk=>clk, en=>en2, q=>op_reg);
   ...

end struktura;
```

Vključevanje komponent (2)

```
oznaka: ime_komp port map (sk1=>sv1, sk2=>sv2, sk3=>open);
```

- Oznaka predstavlja ime dela vezja
  - oznake uporabljamo za poimenovanje procesov, komponent in ostalih gradnikov vezja
- Ime komponente je ime iz entity stavka
- Povezovanje: najprej navedemo ime signala na komponenti, ki ga priredimo signalu v vezju
  - če kakšen signal ni povezan, uporabimo open

Strukture za testiranje vezij

- Testiranje s simulacijo, kjer simulator ročno nastavljamo
  - omejene možnosti, predvsem za preprosta vezja
  - simulator praktično nikoli ne poženemo le enkrat !
- Testiranje s testno strukturo
  - v testni strukturi določimo časovni potek vhodov v vezje
  - ko je testna struktura narejena, jo večkrat uporabimo za izvedbo simulacije vezja
  - v simulatorju se ni potrebno ukvarjati z nastavljanjem vrednosti

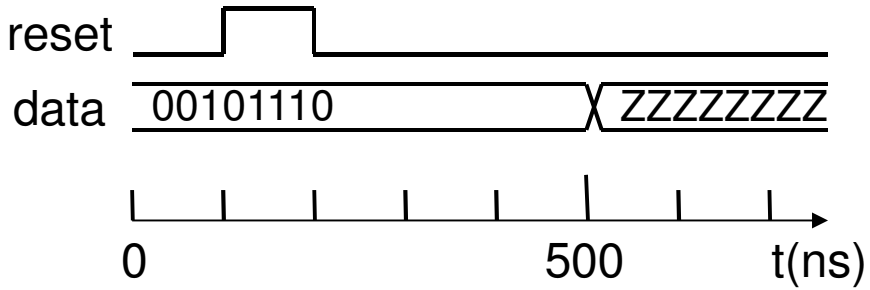
Testna struktura v jeziku VHDL

- Testna struktura - Test Bench
  - model vezja (entity – architecture), ki običajno nima zunanjih priključkov
- Testna struktura vključuje testirano vezje kot komponento
- Za generiranje stimulatorjev (vhodov v vezje) uporabljamo VHDL konstrukte brez omejitev
  - definiramo časovno spreminjanje signalov
  - uporabljamo stavke za izpis poročil med simulacijo
  - uporabljamo IO funkcije za branje in zapis datotek

Časovno spreminjanje signalov

- Signalu lahko priredimo več dogodkov, ki so vezani na čas:

```
reset  <= '0', '1' after 100 ns, '0' after 200 ns;
data <= "00101100", "ZZZZZZZZ" after 500 ns;
```



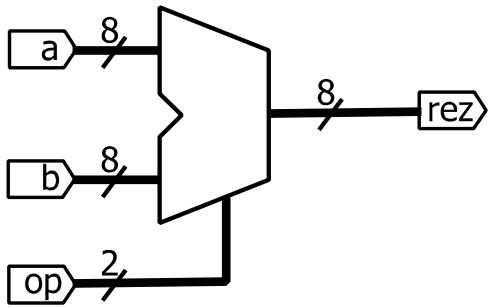
- Kadar je v testu veliko dogodkov uporabimo proces
  - Prirejanje dogodkov le v najbolj preprostih primerih, npr. za enkratne dogodke

Primer: test ALE

- Deklaracija komponente in notranjih signalov

```
architecture behavior of test is
  component ale is
    port ( a, b : in std_logic_vector(7 downto 0);
           op : in std_logic_vector(1 downto 0);
           rez : out std_logic_vector(7 downto 0) );
  end component;

  signal a,b: std_logic_vector(7 downto 0);
  signal op: std_logic_vector(1 downto 0);
  signal rez: std_logic_vector(7 downto 0);
begin
```

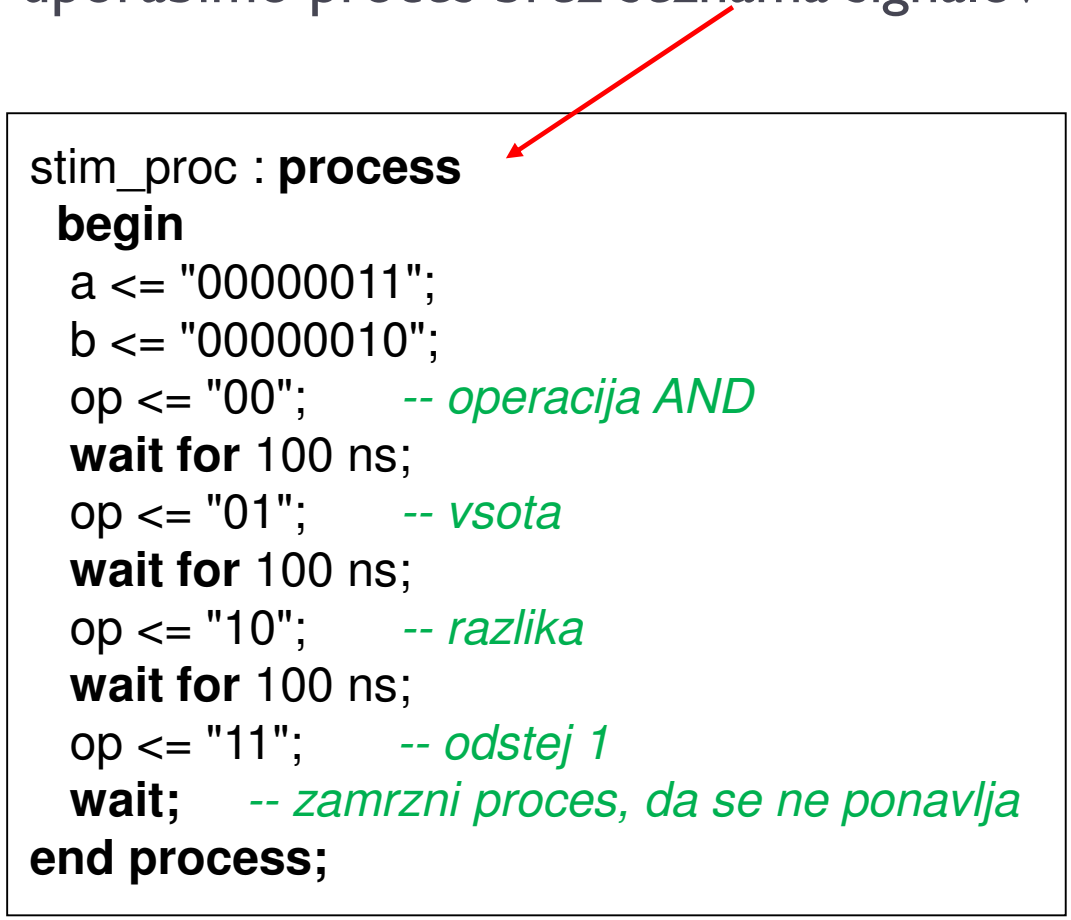


- Vključimo komponento v test

```
UUT: ALE port map (a=>a, b=>b, op=>op, rez=>rez);
```

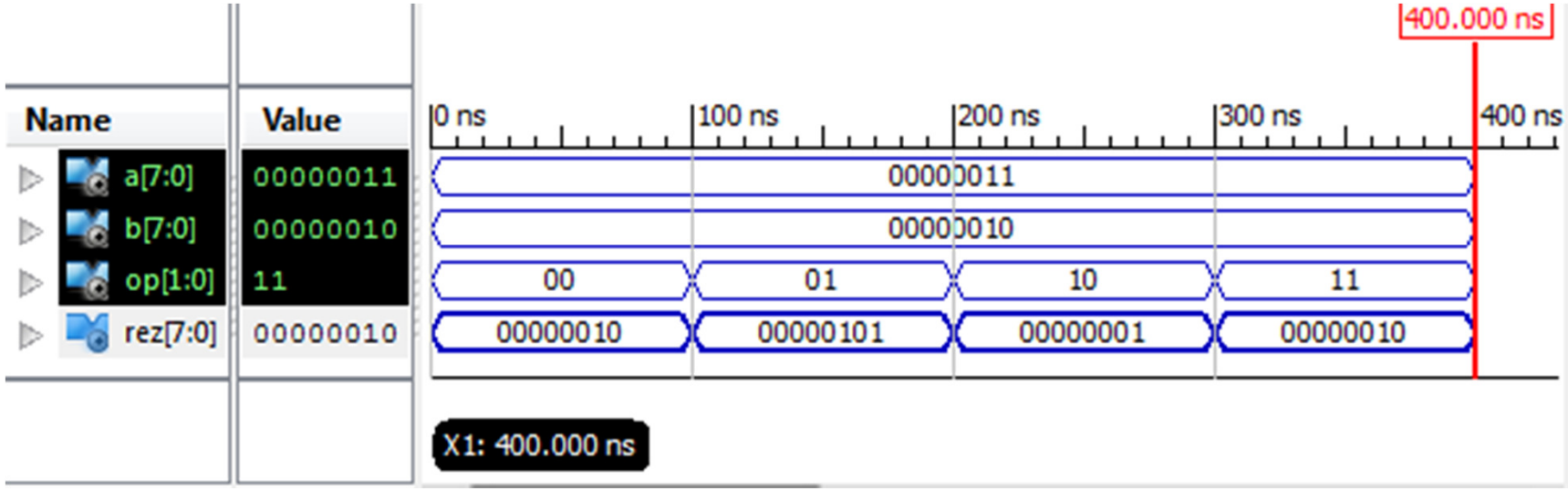
Test ALE (2): nastavljanje vhodov

- Napišemo proces v katerem spreminjamo vhode
  - časovni potek signalov določajo stavki wait for
  - uporabimo proces brez seznama signalov



```
stim_proc : process
  begin
   a <= "00000011";
   b <= "00000010";
   op <= "00";       -- operacija AND
   wait for 100 ns;
   op <= "01";       -- vsota
   wait for 100 ns;
   op <= "10";       -- razlika
   wait for 100 ns;
   op <= "11";       -- odstej 1
   wait;     -- zamrzni proces, da se ne ponavlja
end process;
```

Test ALE(3): rezultat simulacije



- Ali lahko testna struktura samodejno preverja vezje?
  - Da, npr. preverimo ali je vsota takšna, kot jo pričakujemo:

```
op <= "01";       -- vsota
wait for 100 ns;
assert rez = 5 report "Napaka, vsota ni enaka 5 !";
…
```

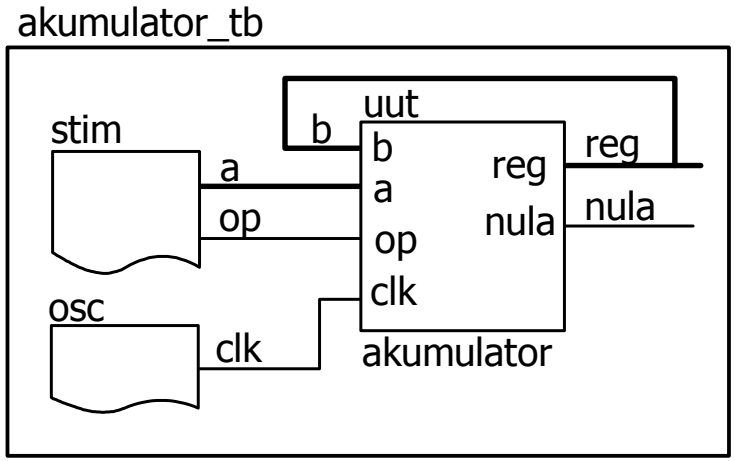
Testiranje sekvenčnih vezij

- Za generiranje ure uporabimo procesno okolje brez seznama signalov in stavke wait for
  - proces se izvaja, dokler ga ne ustavimo z wait

```
GEN_URE: process
  begin
    clk <= '0';
    wait for 50 ns;
    clk <= '1';
    wait for 50 ns;
end process;
```

```
GEN_URE: process
  begin
   if endsim=false then
      clk <= '0';
      wait for 50 ns;
      clk <= '1';
      wait for 50 ns;
   else wait;
end if;
end process;
```

Primer: testiranje akumulatorja



- V testni strukturi naredimo zunanje povezave v vezju
  - b <= reg
- V procesu osc naredimo generator ure
- V procesu stim pa določimo časovni potek ostalih vhodov