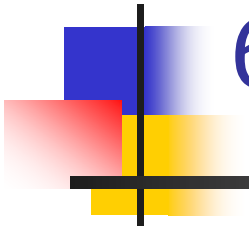
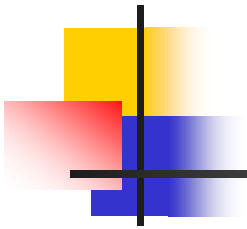


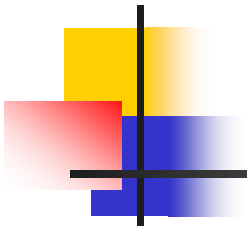
6. Modeliranje jezikom VHDL

- osnovni elementi jezika
- osnove sintakse jezika
- elementi opisa jezikom VHDL
- ponašajni opis
- strukturni opis
- istodobno pridruživanje
- slijedno pridruživanje
- proces simuliranja VHDL koda



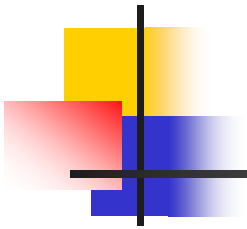
Osnovni elementi jezika VHDL

- opis sklopova ~ *specifikacija*:
  - koncizni i nedvosmisleni *zapis* za razmjenu podataka između projektanata
  - *modeliranje* sklopova
    ~ isticanje bitnih karakteristika uz uklanjanje detalja
  - "ulazni jezik" za:
    - *simuliranje* sklopova
    - CAD (engl. Computer-Aided Design) alate;
      npr. za *sintezu*



Osnovni elementi jezika VHDL

- *jezici za opis sklopovlja*,
  HDL (engl. Hardware Description Languages):
  - posebno razvijeni u tu svrhu
  - ugrađeni primjereni mehanizmi
  - prvi HDL:
    CDL (engl. Computer Design Language), Y. Chu, 1963.
  - *normirani* (i trenutno najšire korišteni) HDL:
    - VHDL
    - Verilog

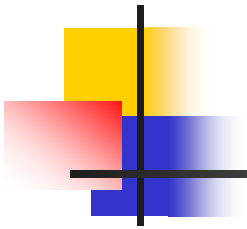


Osnovni elementi jezika VHDL

- VHDL (engl. Very high speed integrated circuits HDL):
  - popratni rezultat projekta DoD VHSIC
  - utvrđen dvama normama IEEE:

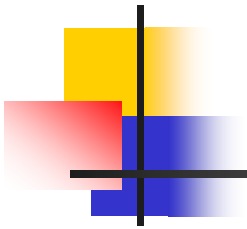
    IEEE Std 1076-1993: *IEEE Standard VHDL Language Reference Manual*, Revision of IEEE Std 1076-1987, June 6, 1994, IEEE, New York, NY, 1994.

    IEEE Std 1164-1993: *IEEE Standard Multivalue Logic System for VHDL Model Interoperability (Std_logic_1164)*, May 26, 1993, IEEE, New York, NY, 1993.



Osnovni elementi jezika VHDL

- osnovne značajke VHDL:
  - sintaksa slična onoj jezika Pascal
    ~ laka čitljivost
  - primjereni mehanizmi za opis *strukture* i *ponašanja* sklopovlja
  - mogućnost poslovanja *komponentama*
    ~ "gotovim" podsklopovima
    koji se tretiraju kao crne kutije

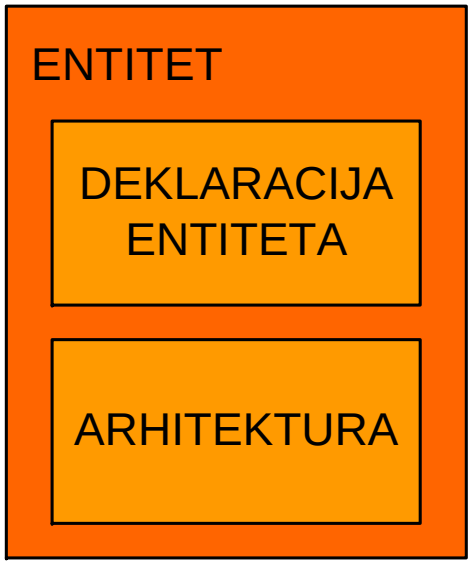


Osnovni elementi jezika VHDL

- osnovna jedinica modeliranja
  ~ digitalni (pod)sklop:
  - crna kutija sa *sučeljem* (ulazi i izlazi)
    ~ formalno *entitet*
  - sučelje
    ~ formalno *deklaracija entiteta*
    - popis ulaza + tip
      [+ parametri za utvrđivanje unutarnjih svojstava]
    - popis izlaza + tip
      [+ parametri za utvrđivanje unutarnjih svojstava]
  - unutarnje ostvarenje sklopa
    ~ formalno *arhitektura*

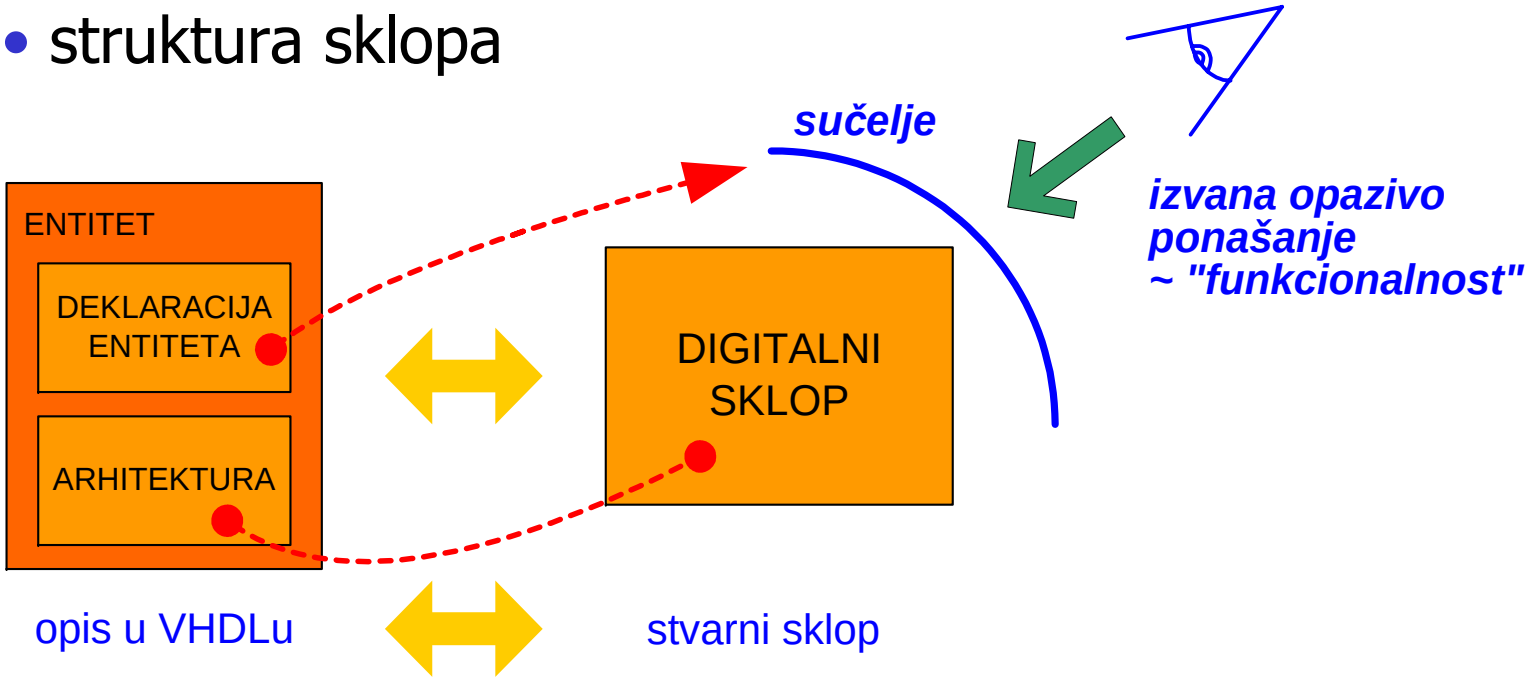
Osnovni elementi jezika VHDL

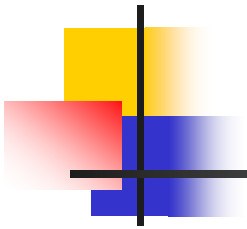
- općenita struktura modela digitalnog sklopa
  ~ entitet:
  - *deklaracija entiteta*
    ~ sučelje
  - *arhitektura*
    ~ unutarnje ostvarenje sklopa
    - deklarativni dio
      ~ unutarnji signali,
      korisnički tipovi, konstante
    - tijelo
      - funkcionalnost sklopa
      - struktura sklopa



Osnovni elementi jezika VHDL

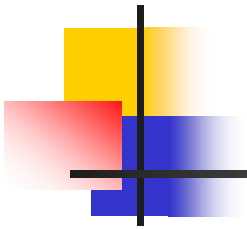
- entitet:
  - *deklaracija entiteta* ~ sučelje
  - *arhitektura* ~ unutarnje ostvarenje sklopa
    - funkcionalnost sklopa
    - struktura sklopa





Osnove sintakse jezika VHDL

- podatkovni objekti:
  - signali
    ~ logički signali (npr. vodovi digitalnog sklopa)
  - konstante
    ~ nepromjenjive vrijednosti kao *pokrate*,
    nisu vodovi digitalnog sklopa
  - varijable
    ~ "programska" pomagala:
    - pridržavanje parcijalnih rezultata izračunavanja
    - indeksi u programskim petljama opisa
    - *mogu* biti vodovi digitalnog sklopa

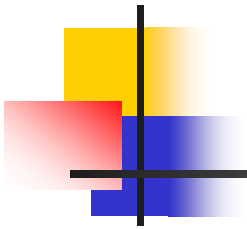


Osnove sintakse jezika VHDL

- leksika *imena* podatkovnih objekata:
  - slova, znamenke i podcrta _
  - *ne* razlikuju se velika i mala slova!
  - moraju započeti slovom
  - *ne smiju*:
    - biti ključne riječi VHDLa
    - završiti podcrtom _
    - sadržavati *dvije* podcrte _ _

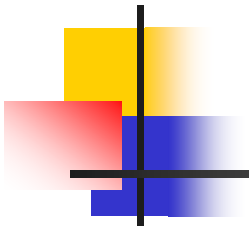
Primjer:

ispravno: `ulaz, ulaz2, UlaZ2, vazan_ulaz`
pogrešno: `ulaz 1, ulaz_, ulaz__2, integer, const, 2ulaz, process`



Osnove sintakse jezika VHDL

- signali
  ~ *osnovni* podatkovni objekti:
  - jednobitni logički signali
    npr. '1', '0'
  - višebitni logički signali
    npr. "1001", "01101001"
  - binarni (cijeli) brojevi



Osnove sintakse jezika VHDL

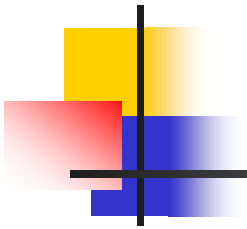
- tipovi VHDL signala *prethodno definiranih* u normama IEEE:
  - bit
  - vektor bitova
  - isto kao gore, ali iz *standardne biblioteke* `std_logic`
  - cijeli brojevi
  - Booleove konstante
  - nabrajanje

Osnove sintakse jezika VHDL

- *prethodno definirani* tipovi VHDL signala:

tip signala	sintaksa	semantika tipa
bit	**signal** a : bit;	0 ili 1
bit_vector	**signal** A : bit_vector (0 **to** 7); **signal** B : bit_vector (7 **downto** 0);	linearno polje ("vektor") bitova u rastućem ili opadajućem poretku
std_logic; std_logic_vector	**library** ieee; **use** ieee.std_logic_1164.all; **signal** I0, I1, S : std_logic; **signal** A : std_logic_vector (0 **to** 7); **signal** B : std_logic_vector (7 **downto** 0);	standardni prethodno definirani tipovi koji su uključeni u biblioteku ieee kao paket std_logic_1164; prethodno je potrebno deklarirati korištenje biblioteke i paketa klauzulama library i use
integer	**signal** A : integer **range** –128 **to** 127;	8-bitni cijeli broj u zapisu 2-komplementa
boolean	**signal** Status : boolean;	true (1) ili false (0)
nabrajanje	**type** stanje **is** (stanjeA, stanjeB, StanjeC, stanjeD); **signal** s : stanje;	korisnički specificirane vrijednosti tipa, primjerice stanja stroja s konačnim brojem stanja

- osnovne vrijednosti u `std_logic`:
  0, 1, U (nije zadano), – (nespecificirano)
- ostale vrijednosti u `std_logic`:
  Z (visoka impedancija), L i H ("slabi" 0 i 1) ,
  X i W (nepoznate vrijednosti)



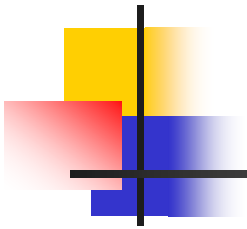
Osnove sintakse jezika VHDL

- dodjela vrijednosti
  ~ naredba pridruživanja:
  - pridruživanje vrijednosti signalu:
    <=
  - pridruživanje vrijednosti konstanti:
    :=
  - pridruživanje vrijednosti varijabli:
    :=

Osnove sintakse jezika VHDL

- operatori kombiniranja unutar naredbe pridruživanja:
  - između pojedinih grupa postoje prioriteti!
  - *nema* prednosti operatora unutar pojedine grupe
    ~ koristiti zagrade!

prioritet	klasa operatora	operator
najviši	miješani operatori	`**`, `abs`, `not`
	operatori množenja	`*`, `/`, `mod`, `rem`
	operatori predznaka	`+`, `-`
	operatori zbrajanja	`+`, `-`, `&`
	relacijski operatori	`=`, `/=`, `<`, `<=`, `>`, `>=`
najniži	logički operatori	`and`, `or`, `nand`, `nor`, `xor`, `xnor`



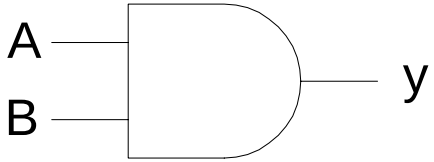
Elementi opisa jezikom VHDL

- struktura VHDL opisa:
  - navođenje korištenih biblioteka
  - definiranje sučelja sklopa
  - definiranje arhitekture sklopa

Elementi opisa jezikom VHDL

Primjer

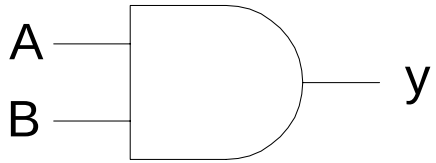
```
library ieee;
use ieee.std_logic_1164.all;

entity sklopAND2 is
  port (a, b: in std_logic;
        y:    out std_logic);
end sklopAND2;

architecture ponasajna of sklopAND2 is
begin
  y <= a and b after 10 ns;
end ponasajna;
```



Elementi opisa jezikom VHDL

- specifikacija *sučelja* sklopa:

```
...
entity sklopAND2 is
    port (a, b: in std_logic;
          y:    out std_logic);
end sklopAND2;
...
```



  - ključne riječi:
    - **entity**
    - **port**
    - **is**, **end**, **in**, **out**

Elementi opisa jezikom VHDL

- specifikacija sučelja sklopa:
  - tipovi (~"načini", modes) *pristupa* (engl. port):

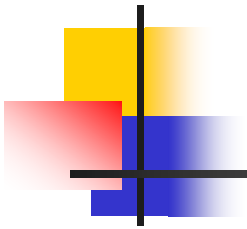
način	semantika
in	signal je ulazni za entitet
out	signal je izlazni za entitet; ovaj se signal ne može ponovno upotrijebiti unutar entiteta (ne smije se pojaviti s lijeve strane operatora pridruživanja <=)
inout	signal je dvosmjeran u odnosu na entitet
buffer	signal je izlazni za entitet, ali se može ponovno upotrijebiti unutar entiteta (smije se pojaviti s obje strane operatora pridruživanja <=)

  - u primjeru:
    - a, b: ulazni signali
    - y: izlazni signal

Elementi opisa jezikom VHDL

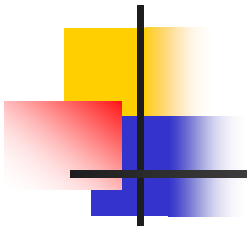
- specifikacija *arhitekture*:

```
...
architecture NazivArhitekture
   of sklopAND2 is
  -- deklariranje internih signala
  -- deklariranje vrsta korištenih komponenti
begin
  -- opis ponašanja ili strukture sklopa
end NazivArhitekture ;
...
```



Ponašajni opis

- opis rada sklopa
  ~ definicija *ponašanja* sklopa:
  - izražavanje logičke funkcije koja povezuje izlaz s ulazima
  - definicija vrijednosti izlaza za svaku ulaznu kombinaciju
  - algoritam dodjele vrijednosti izlazu



Ponašajni opis

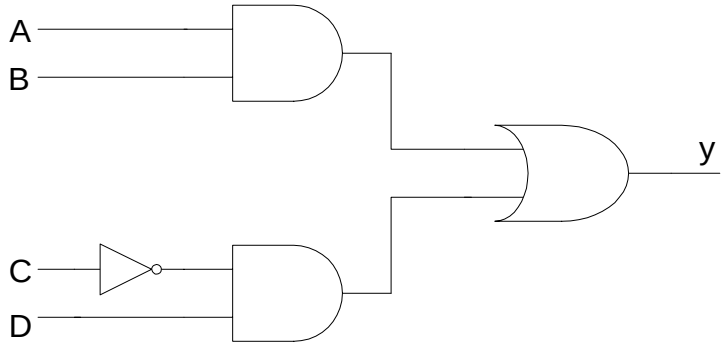
- opis rada sklopa
  ~ definicija *ponašanja* sklopa:

```
architecture ponasajna of sklopAND2 is
begin
   y <= a and b after 10 ns;
end ponasajna;
```

  - funkcija: y = a·b
  - kašnjenje primitivnih sklopova 10 ns

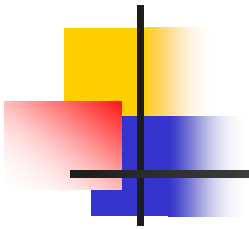
Ponašajni opis

Primjer: opis sklopa na slici



...

```
architecture sS of slozeniSklop is
begin
   y <= (a and b) or (not c and d)
        after 25 ns;
end sS;
```

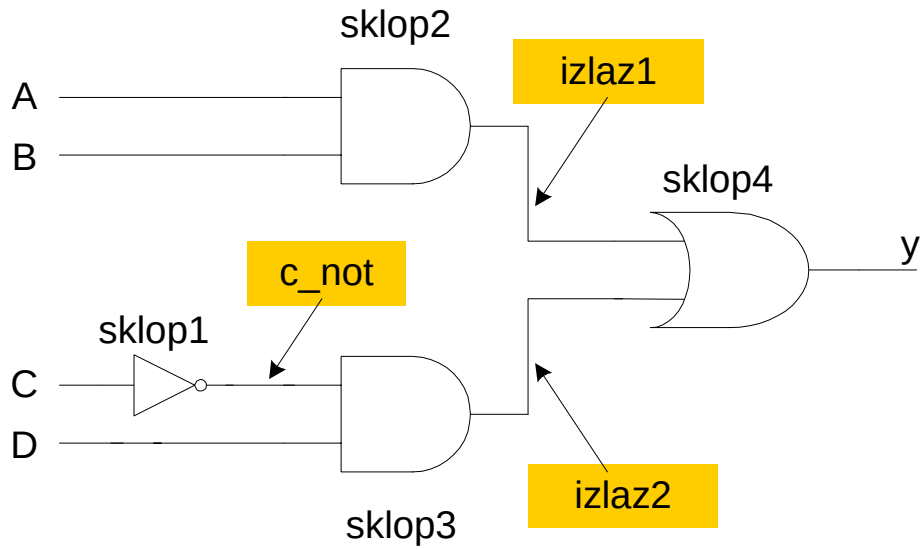


Strukturni opis

- opis rada sklopa
  ~ definicija *strukture* sklopa
  ostvarenog *jednostavnijim* komponentama
  - *hijerarhijski dizajn*, npr.:
    - računalo:
      procesor, memorija, ulazno/izlazne jedinice,...
    - procesor:
      ALU, upravljačka jedinica,...
  - najniža razina hijerarhije:
    - jednostavne komponente
    - ponašajni opis

Strukturni opis

Primjer:



- 3 vrste sklopova: NE, I, ILI
- 3 interna signala: `c_not`, `izlaz1`, `izlaz2`

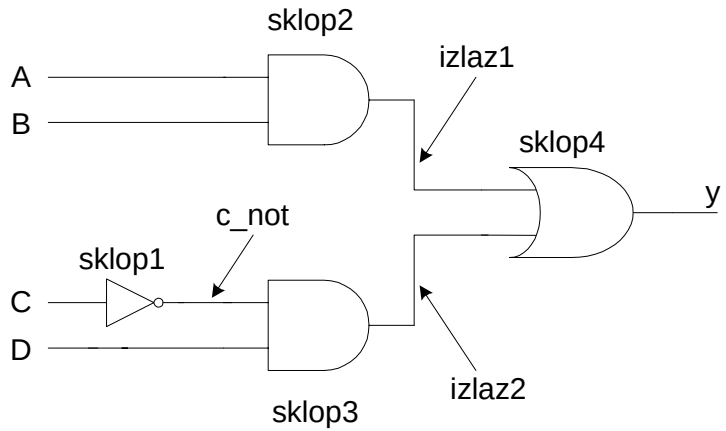
Strukturni opis

- *komponente* sklopa
  ~ *ponašajni* opis, svaka u *svojoj* datoteci

```
...
architecture ponasajna of sklopNOT is
begin
  y <= not a after 5 ns;
end ponasajna;
---------------------
...
architecture ponasajna of sklopAND2 is
begin
  y <= a and b after 10 ns;
end ponasajna;
---------------------
...
architecture ponasajna of sklopOR2 is
begin
  y <= a or b after 10 ns;
end ponasajna;
```

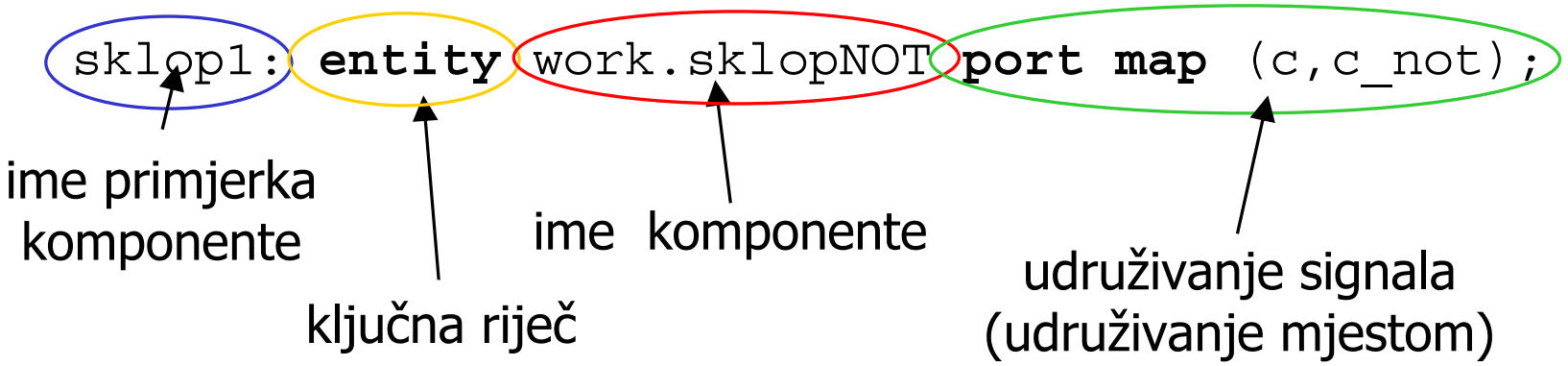
Strukturni opis

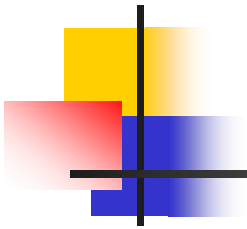
```
...
architecture strukturna of slozeniSklop is
  signal c_not, izlaz1, izlaz2: std_logic;
begin
  sklop1: entity work.sklopNOT  port map (c,c_not);
  sklop2: entity work.sklopAND2 port map (a,b,izlaz1);
  sklop3: entity work.sklopAND2 port map (c_not,d,izlaz2);
  sklop4: entity work.sklopOR2  port map (izlaz1,izlaz2,y);
end strukturna;
```



Strukturni opis

- *stvaranje primjerka* komponente (engl. instantiation):





Strukturni opis

- udruživanje:
  - udruživanje mjestom (*engl. positional association*)
    ~ redoslijed deklariranja pristupa
    u deklaraciji komponenti

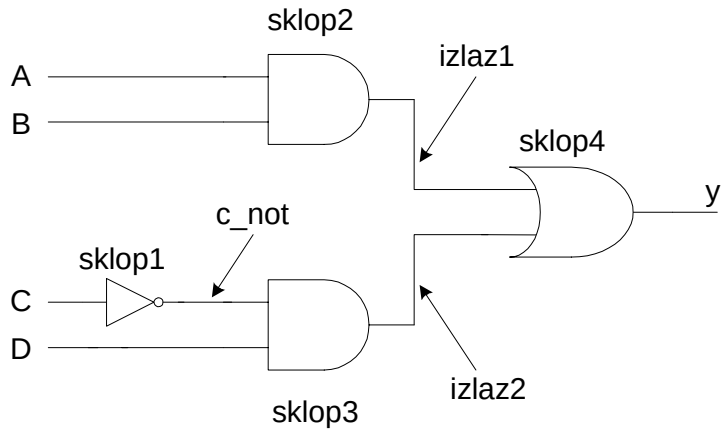
```
... port map (c_not,d,izlaz2);
```

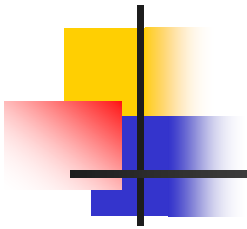
  - udruživanje imenima (*engl. named association*)
    ~ redoslijed navođenja signala *nije bitan*

```
... port map (a => c_not,
              y => izlaz2,
              b => d);
```

Mješoviti opis

```
...
architecture strukturna of slozeniSklop is
  signal c_not, izlaz1, izlaz2: std_logic;
begin
   -- sklop1: entity work.sklopNOT port map (c,c_not);
   c_not <= not c after 5 ns; -- ponašajno
   sklop2: entity work.sklopAND2 port map (a,b,izlaz1);
   sklop3: entity work.sklopAND2 port map (c_not,d,izlaz2);
   sklop4: entity work.sklopOR2  port map (izlaz1,izlaz2,y);
end strukturna;
```



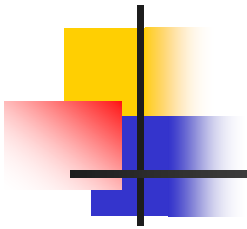


Paketi i biblioteke

- *paketi* (engl. packages)
  ~ skladišta VHDL deklaracija opće namjene:
  - tipova i podtipova
  - konstanti
  - komponenti
  - signala
  - funkcija i procedura

Primjer :

```
ieee.std_logic_1164
```



Paketi i biblioteke

- *biblioteke* (engl. libraries):
  - dodatno grupiranje jedinica VHDL koda
  - organizirane u podkazala datotečnog sustava
  - 2 vrste biblioteka
    - sustavske (npr. `ieee`)
    - korisničke (npr. `work`)
- deklaracija korištenja biblioteke:

```
library ieee;
```

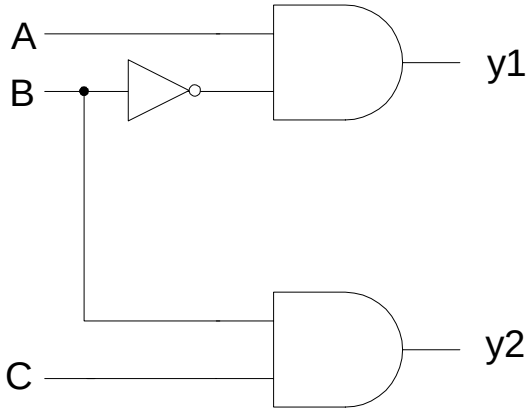
- uključivanje paketa:

```
use ieee.std_logic_1164.all;
```

Istodobno pridruživanje

- koncept istodobnog pridruživanja:
  - stvarni sklop
    ~ *istodobna* (engl. concurrent) aktivnost komponenti
  - VHDL podržava modeliranje istodobnih aktivnosti

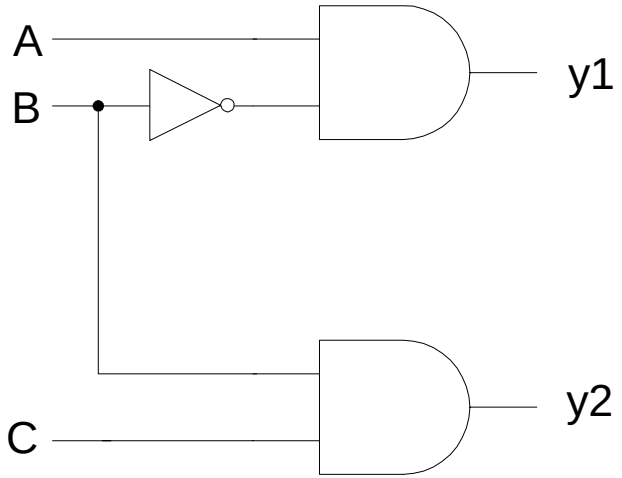
```
library ieee;
use ieee.std_logic_1164.all;

entity sklop3 is
  port (a, b, c: in std_logic;
        y1, y2:  out std_logic);
end sklop3;
...
```



Istodobno pridruživanje

- naredba *jednostavnog* pridruživanja: <=

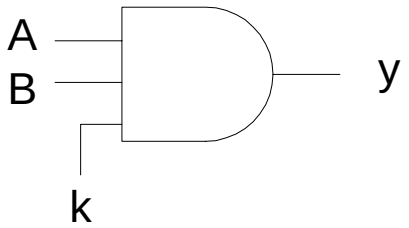
```
architecture ponasajna of sklop3 is
begin
  y1 <= a and not b after 15 ns;
  y2 <= b and c after 10 ns;
end ponasajna;
```



Istodobno pridruživanje

- naredba *izbornog* pridruživanja (engl. selected assignment):

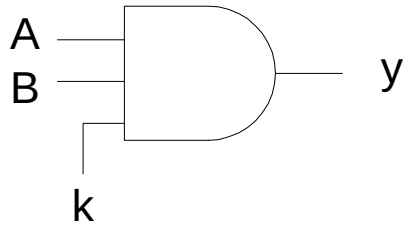
```
architecture ponasajna of sklop4 is
begin
  with k select
     y <= a and b when '1',
     '0'          when others;
end ponasajna;
```



Istodobno pridruživanje

- naredba *uvjetnog* pridruživanja (engl. conditional assignment):

```
architecture ponasajna of sklop4 is
begin
  y <= a and b when k = '1'
               else '0';
end ponasajna;
```



Slijedno pridruživanje

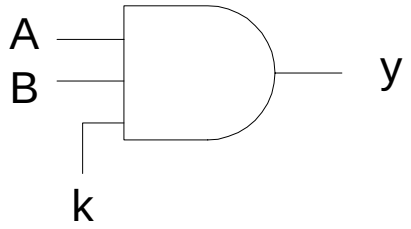
- koncept *slijednog* (engl. sequential) pridruživanja:
  - algoritamski pristup
    ~ bitan redoslijed izvršavanja
  - obuhvaćanje blokom **`process`**

```
architecture Naziv of sklop is
begin
  -- naredba istodobnog izvršavanja
  ...
  process
  begin
    -- slijedno izvršavanje
    ...
  end process;
  ...
  -- naredba istodobnog izvršavanja
  ...
end Naziv;
```

Slijedno pridruživanje

- naredbe *grananja*:

```
architecture ponasajna of sklop4 is
begin
  process(a,b,k) -- lista osjetljivosti
  begin
     if k = '1'
        then y <= a and b;
        else y <= '0';
     end if;
  end process;
end ponasajna;
```

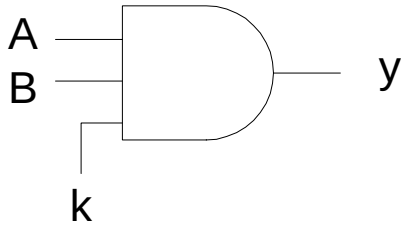


- *lista osjetljivosti*
  ~ popis varijabli koje "okidaju" izvršavanje procesa

Slijedno pridruživanje

- naredbe grananja:

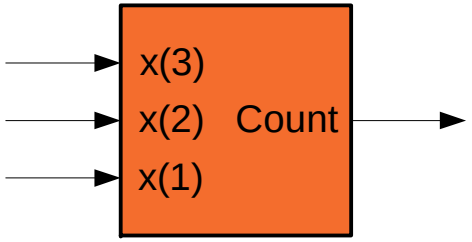
```
architecture ponasajna of sklop4 is
begin
  process(a,b,k)
  begin
     case k is
       when '1'      => y <= a and b;
       when others   => y <= '0';
     end case;
  end process;
end ponasajna;
```

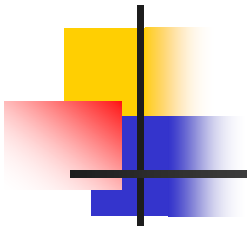


Slijedno pridruživanje

- naredbe *petlje*:

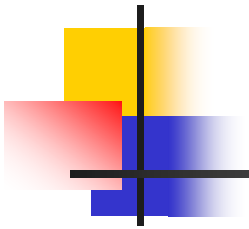
```
entity brojilo_jedinica is
  port(x:     in std_logic_vector(1 to 3);
       Count: buffer integer range 0 to 3);
end brojilo_jedinica;

architecture ponasajna of brojilo_jedinica is
begin
  process(x)
  begin
    Count <= 0;
    for i in 1 to 3 loop
      if x(i)='1'
         then Count <= Count+1;
      end if;
    end loop;
  end process;
end ponasajna;
```





Slijedno pridruživanje

- *lista osjetljivosti* (engl. sensitivity list):
  - popis signala kod čije će se promjene odnosni izraz (ponovno) izračunati
    ~ "okinuti izvršavanje"
  - uz *svaki* izraz *jedna od* alternativa:
    - *eksplicitna* lista osjetljivosti
      ~ nazivi signala odvojeni zarezima (blok **`process`**)
    - *implicitna* lista osjetljivosti
      ~ naredbe istodobnog pridruživanja



Slijedno pridruživanje

Primjer: implicitna lista osjetljivosti

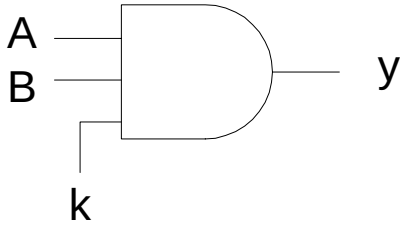
```
architecture ponasajna of sklop3 is
begin
  izraz1: y1 <= a and not b after 15 ns;
  izraz2: y2 <= b and c after 10 ns;
end ponasajna;
```

- lista osjetljivosti za `izraz1:` `(a,b)`
- lista osjetljivosti za `izraz2:` `(b,c)`
- promjena signala `a` ne "okida"
  ponovno izračunavanje `izraz2`

Slijedno pridruživanje

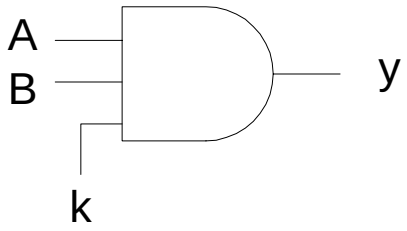
Primjer: eksplicitna lista osjetljivosti

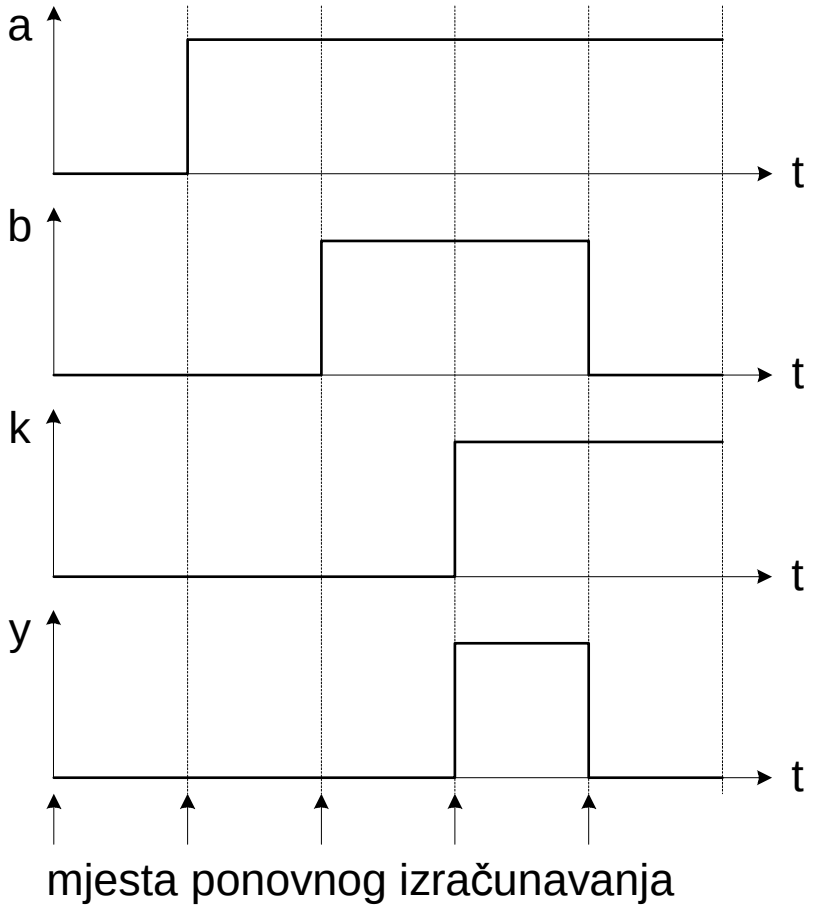
```
architecture ponasajna of sklop4 is
begin
  process(a,b,k) -- (a,b,k)= lista osjetljivosti
  begin
     if k = '1' then
        y <= a and b;
     else
        y <= '0';
     end if;
  end process;
end ponasajna;
```



Slijedno pridruživanje

- djelovanje liste osjetljivosti:

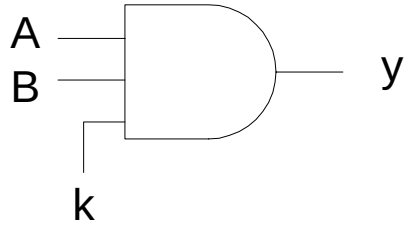




Slijedno pridruživanje

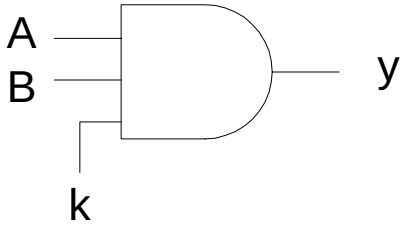
Primjer: eksplicitna lista osjetljivosti 2

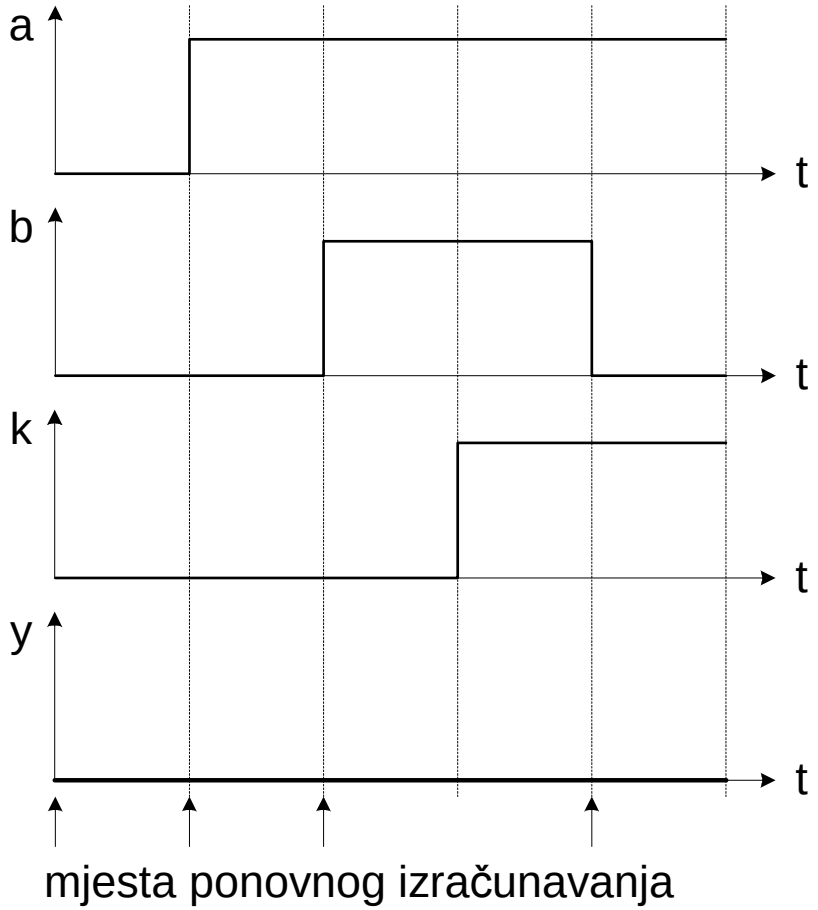
```
architecture ponasajna of sklop4 is
begin
  process(a,b) -- u listi osjetljivosti nema k!!!
  begin
     if k = '1' then
        y <= a and b;
     else
        y <= '0';
     end if;
  end process;
end ponasajna;
```

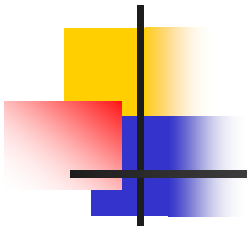


Slijedno pridruživanje

- djelovanje liste osjetljivosti 2:
  - u listi nema `k`







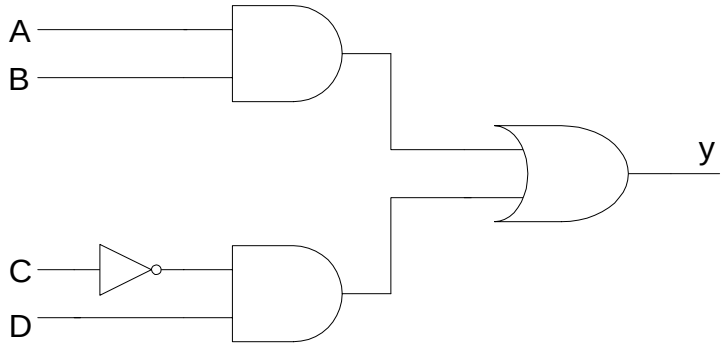
Proces simuliranja VHDL koda

- osnovni pojmovi:
  - model
    ~ simulirani sklop
  - realno vrijeme, vrijeme simulacije
    ~ vrijeme "izvan simulacije":
    - od trenutka pokretanja
    - do trenutka završetka simulacije
  - simulirano vrijeme
    ~ vrijeme koje protječe simuliranom sklopu
  - realno vrijeme i simulirano vrijeme
    nisu u međusobnom odnosu

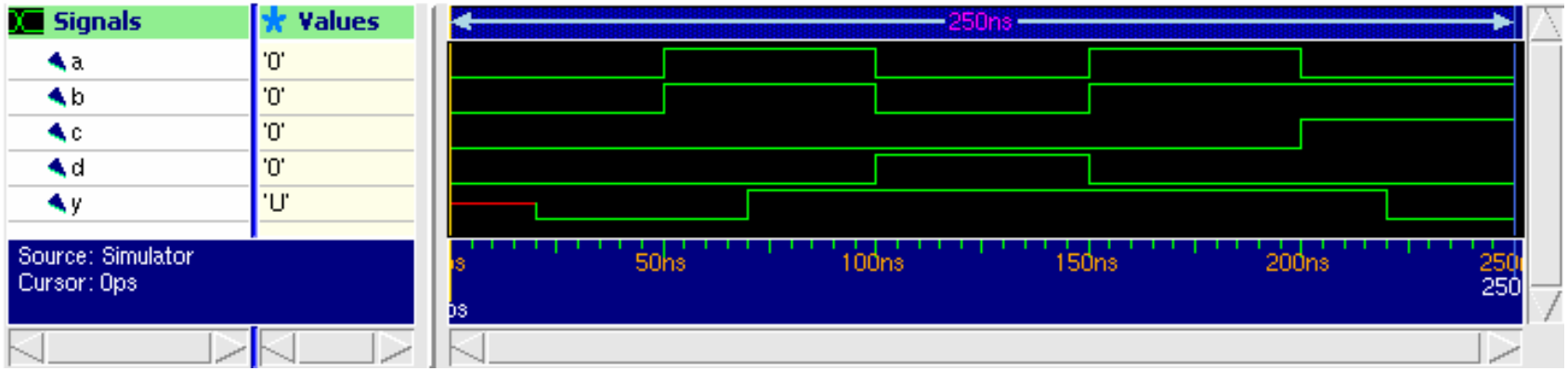
Proces simuliranja VHDL koda

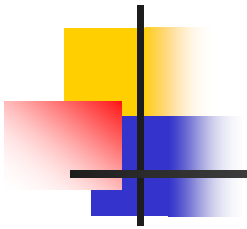
Primjer:

kašnjenja sklopova: $t_{dI} = t_{dILI} = 2 \cdot t_{dNE} = 10$ ns



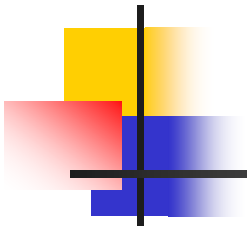
VHDL model sklopa





Proces simuliranja VHDL koda

- koncept izvršavanja simulacije:
  - provjera da se opis sklopa (model) ponaša kako se očekivalo
  - cikličko izračunavanje izraza s "napretkom" simuliranog vremena:
    - u nekom trenutku simuliranog vremena simulacija se obavlja kroz *niz* "delta ciklusa"
    - *delta ciklus*
      ~ jedan ciklus izračunavanja paralelnih izraza, beskonačno kratkog (simuliranog) trajanja



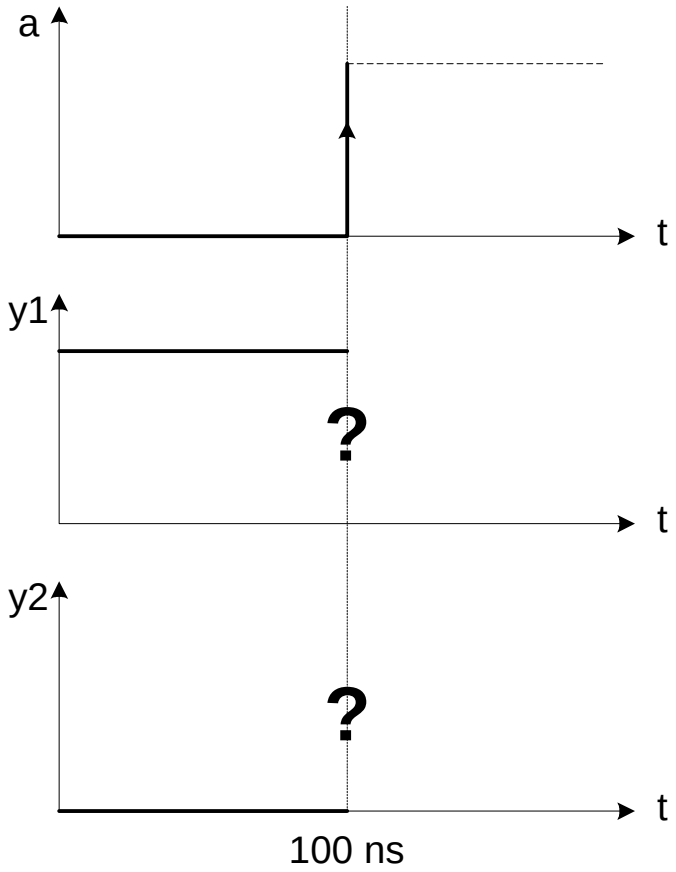
Proces simuliranja VHDL koda

- ponavljanje ciklusa simulacije
  ~ postoji signal (ili više njih)
  promijenjen u tom delta ciklusu:
  - promjena signala u listi osjetljivosti nekog od izraza:
    - da: novi delta ciklus (simulirano vrijeme stoji!)
    - ne: simulirano vrijeme napreduje
  - provjera lista osjetljivosti *na kraju* izvršavanja svakog delta ciklusa

Proces simuliranja VHDL koda

Primjer: rezultat simulacije u t = 100 ns ?

```
architecture ponasajna of sklop5 is
begin
   izraz1: y1 <= not y2;
   izraz2: y2 <= a;
end ponasajna;
```



Proces simuliranja VHDL koda

Primjer: rezultat simulacije u t = 100 ns ?

- t = 100 ns : promjena `a`
- simulator zamrzava vrijeme
- postoje li izrazi koji u listi osjetljivosti imaju `a`?
  → `izraz2`
- izračunavanje `izraz2, y2 <= '1'`
- promjena `y2`
- postoje li izrazi koji u listi osjetljivosti imaju `y2`?
  → `izraz1`

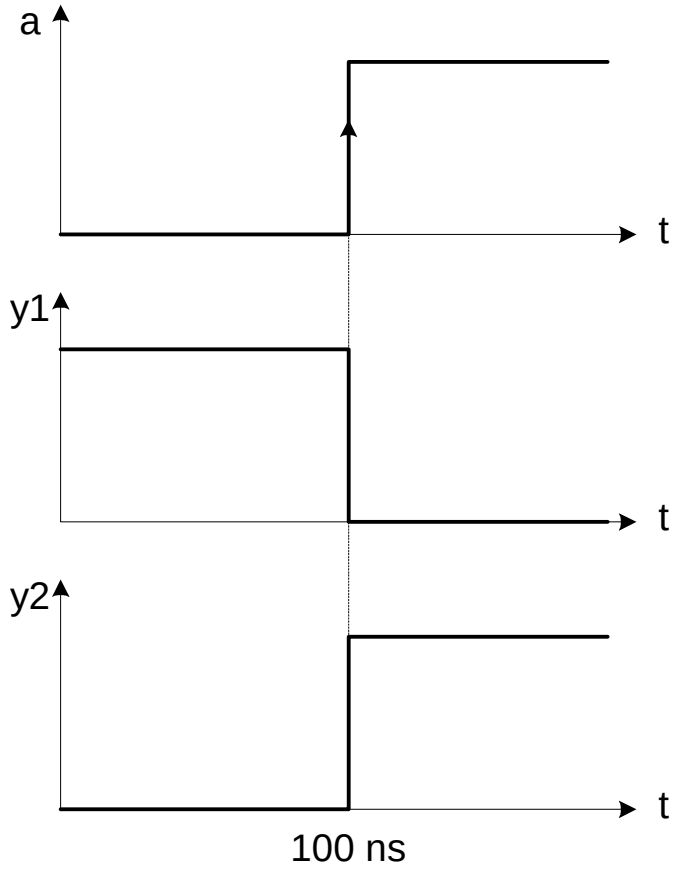
$\Delta 1$

- izračunavanje `izraz1, y1 <= '0'`
- promjena `y1`
- postoje li izrazi koji u listi osjetljivosti imaju `y1`?
  → NE! (napredak simuliranog vremena)

$\Delta 2$

Proces simuliranja VHDL koda

Primjer: rezultat simulacije u t = 100 ns ?



Proces simuliranja VHDL koda

Primjer za vježbu:
(svi su signali tipa `std_logic`)

```
architecture ponasajna of
    sklop6 is
begin
     izraz1: A <= not(W and D);
     izraz2: B <= not(W and C);
     izraz3: C <= not(A and D);
     izraz4: D <= not(B and C);
end ponasajna;
```

Proces simuliranja VHDL koda

Primjer za vježbu:

Koji je rezultat simulacije uz pobudu sa slike?