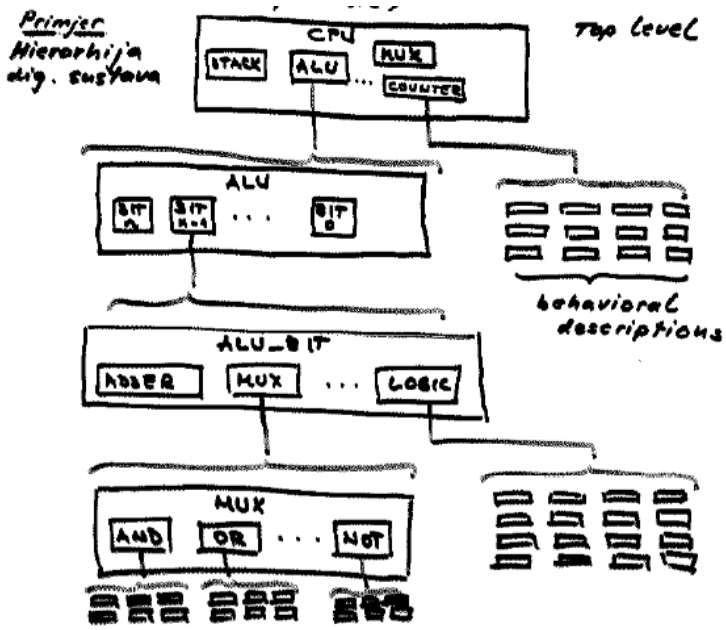
VHDL - VHSIC (Very High Speed Integrated Circuit) Hardware Description Language

- alat za standardno oblikovanje i dokumentiranje za VHSIC
- United States Department of Defense (DoD) sponzorirao radionicu u ljeto 1981. u Woods Hole, Massachusetts (Workshop on HDLs), Institute for Defense Analysis (IDA)
- 1983. godine Dod postavlja zahtjeve za standardnim VHSIC sklopovskim opisnim jezikom
- VHDL jezik -> ugovor IBM, Texas Instruments i Intermetrics
- VHDL 2.0 - šest mjeseci nakon početka projekta
- VHDL 6.0 - prosinac 1984.
- VHDL 7.2 - Language Reference Manual
- IEEE 1076 - svibanj 1987.
- VHDL 1076 - 1987 -> IEEE standard HDL (prosinac 1987.)
- IEEE DASC (Design Automation Standard Commiteee) -> nova verzija 1992. godine
- VHDL 1076 - 1993.: IEEE 1076 - 1987. i IEEE 1164 - 1993. godina -> kompletni VHDL standard

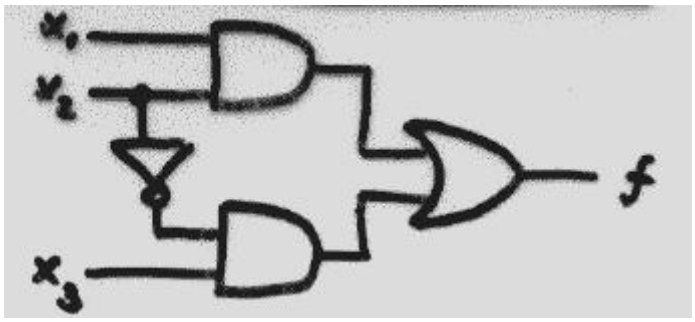
Zahtjevi DOD-a za VHSIC HDL:

1. VHSIC HDL (VHDL) mora biti JEZIK ZA OBLIKOVANJE I OPISIVANJE sklopovlja (dokumentiranje, oblikovanje visoke razine - high-level design, simulacija, sinteza, ispitivanje sklopovlja, te driver za uređaje za fizičku realizaciju sklopa); digitalni sustav -> komponente istodobno aktivne i (zadatak/e) sustav obavlja konkurentno. Zato VHDL mora imati značajke konkurentnosti i istodobnosti obavljanja zadataka. U HDL-u konkurentnost znači da se naredbe prijenosa, opisa komponenti, dodjeljivanje logičkih sklopova i jedinica obavlja tako da izgleda kao da se izvršavaju istodobno.
2. Potpora hijerarhijskom oblikovanju
   - hijerarhijska specifikacija sklopovlja pomoću jezika VHDL
   - djelovanje sustava može se specificirati na razini njegove FUNKCIJE ili pak
   - može biti specificirano STRUKTURNO pomoću manjih (pod)komponenti (na najnižoj razini komponente su opisane funkcijom i ne koriste se podkomponente)



3. Potpora bibliotekama
   - biblioteke za VHDL
     i. korisničko definirano primitivi
     ii. sistemsko definirani primitivi
   - jezik VHDL mora imati mehanizme za pristup različitim bibliotekama
4. Sekvencijske naredbe
   - iako je dat jaki naglasak na konkurentnom izvođenju procesa i naredbi u VHDL-u, postoji i zahtjev za "softwarelike" sekvencijsko izvođenje
5. Mogućnost generiranja generičkog opisa komponenti (veličina, fizičke karakteristike, timing, uvjeti okoline i sl.)
6. Potpora različitih tipova podataka
   - bit/boolean
   - integer
   - floating-point
   - enumerate tipovi
   - user-defined
   - array-type
   - composite-type
7. Uporaba potprograma - funkcije, procedure
8. Upravljanje vremenskim vođenjem (Timing Control)
   - jezik VHDL treba omogućiti dizajneru da dodjeljuje vrijednosti signalima i njihova kašnjenja (gate or line delay)
   - eksplicitno definiranje taktnih signala (konstrukti za detekciju brida signala, specifikaciju kašnjenja, provjeru širine impulsa i sl.)
9. Strukturna dekompozicija sklopovlja na svim razinama (npr. može se opisati generički 1-bitni design i zatim ga upotrijebiti za opis multibitne pravilne strukture u 1 ili više dimenzija)

JEDNOSTAVAN PRIMJER



- VHDL kod koji prikazuje gornji sklop

1. korak: Definirati ulazne i izlazne signale
   - rabi se **konstrukt ENTITY**

```
ENTITY example IS
PORT (x1, x2, x3: IN BIT;
        f: OUT BIT);
END example;
```

   - konstruktu ENTITY mora se dodijeliti ime
   - ulazni i izlazni signali nazivaju se portovima entiteta (Ports) - ključna riječ PORT

- svakom portu pridružen je NAČIN (mode) -> input (IN) ili output (OUT)
- entitet u našem primjeru ima ukupno 4 porta: x1, x2 i x3 su ulazni signali tipa BIT, a port s imenom f je izlaz također tipa BIT
- BIT kao tip podataka - objekt tog tipa predstavlja digitalni signal budući da BIT objekt može imati samo DVIJE vrijednosti (0 i 1)
- ime signala MORA započeti slovom (ime signala ne može biti ključna riječ VHDL-a)
- entitet određuje samo ulazne i izlazne signale sklopa ALI NE DAJE podrobnosti što sklop predstavlja

2. Funkcija sklopa mora se specificirati uporabom konstrukta VHDL koji se naziva **arhitektura (ARCHITECTURE)**

```
ARCHITECTURE LogFun OF example IS
BEGIN
        f <= (x1 AND x2) OR (NOT x2 AND x3);
END LogFun;
```

- arhitekturi mora biti dato ime
- VHDL ima ugrađene slijedeće Booleove operatore: AND, OR, NOT, NAND, NOR, XOR, XNOR
- BEGIN - ključna riječ
- operator dodjeljivanja <=
- VHDL ne pretpostavlja prioritete log. operatora unaprijed - ZATO se moraju rabiti zagrade
- za ovo će VHDL prevodilac generirati grešku tijekom prevođenja: f <= x1 AND x2 OR NOT x2 AND x3
- kompletni VHDL kod:

```
ENTITY example IS
        PORT (x1, x2, x3: IN BIT;
                f: OUT BIT);
END example;
ARCHITECTURE LogFun OF example IS
BEGIN
        f <= (x1 AND x2) OR (NOT x2 AND x3);
END LogFun;
```

Primjer sinteze sklopa na temelju VHDL koda

- logički signal je u VHDL-u predstavljen kao objekt podataka (dana object), a svakom se takvom objektu pridružuje tip
- npr. svi objekti podataka u prethodnom primjeru imali su tip BIT
- VHDL nudi i drugi tip podataka nazvan STD_LOGIC – standardni tip podataka za prikaz logičkih signala. Da bi se koristio STD_LOGIC, VHDL mora sadržavati još dvije dodatne linije koda -> one su namijenjene kao direktive VHDL prevodiocu

/ VHDL standard IEEE 1076 nije uključivao tip STD_LOGIC /

- IEEE 1164 standard ima STD_LOGIC kao skup datoteka koje se mogu uključiti u VHDL kod tijekom prevođenja… (skup datoteka =LIBRARY)

1. ⟶ LIBRARY ieee;
2. ⟶ USE ieee.std_logic_1164.all;

```
ENTITY func2 IS
        PORT (x1,x2,x3 : IN STD_LOGIC;
                f      : OUT STD_LOGIC);
END funct2;
ARCHITECTURE  LogicFunc OF func2 IS
BEGIN
        f <= (NOT x1 AND NOT x2 AND x3) OR
             (x1 AND NOT x2 AND NOT x3) OR
             (x1 AND NOT x2 AND x3) OR
             (x1 AND x2 AND NOT x3);
END LogicFunc;
```

STD_LOGIC –> 0, 1, 2 i —

2 - stanje visoke impedancije (treće log. stanje)

- don't care —

std_logic_1164 – paket (engl. package)

all – govori da je cijeli paket za nas zanimljiv

Elementi VHDL-a

VHDL se koristi za opisivanje sklopovskih komponenti i sustava. Najjednostavniji oblik za opisivanje komponenti sastoji se od:

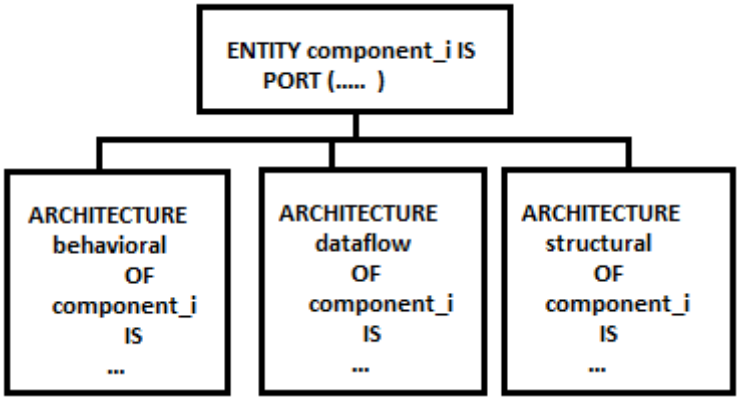
i) sučeljne specifikacije (engl. interface specification)

ii) arhitektonske specifikacije

```
ENTITIY component_name IS
        input and output parts
        physical and other parameters
END component_name;
```

```
ARCHITECTURE identifier OF
                    component_name IS
    declarations
BEGIN
    specification oft he functionality
    of the component in terms of
    its input lines and influenced
    by physical and other parameters
END identifier;
```

- Arhitektura specifikacija određuje i opisuje funkciju komponente.
- Različite specifikacije s različitim identifikacijama mogu postojati za jednu komponentu koja ima zadanu sučeljnu specifikaciju:



Zašto?
- za simulatore kojima se ispituje funkcija sustava upotrebljava se BEHAVIORAL specifikacija
- za simulaciju u vezi vremenskog vođenja i vremenskog odziva – STRUKTURNA specifikacija

behavioral
dataflow
structura

} identifikatori NISU dio jezika VHDL ali ih koristimo da bismo specificirali razine apstrakcije VHDL opisa

- višestruke arhitektonske specifikacije u VHDL-u dopuštaju da se u arhitektonski opis uključi i specifičnosti tehnologije –
Npr. strukturni opis component_i može biti tako oblikovan da se sklop konfigurira za CMOS ili NMOS tehnologiju bez mijenjanja njegovog funkcijskog opisa

Paketi (Packages)

U VHDL okolini postoji potreba za grupiranjem komponenti i korisnik programa (engl. utilities) koji se rabe za opis komponenti.

PAKETI (definicije tipova podataka i potprogrami) PAKETI VIDLJIVI DRUGIM DESIGN JEDINICAMA

BIBLIOTEKE (Design Libraries)

VHDL 1076 standard definira biblioteku kao „implementacijski zavisno mjesto za pohranu prethodno analiziranih oblikovanih jedinica (engl. design units)“.

BIBLIOTEKA obično predočena kao zbirka općenito rabljenih oblikovanih jedinica (npr. paketa i tijela pakete) određena jedinstvenim imenom te se može referencirati iz višestrukih izvornih podataka u postupku oblikovanja.

Oblikovana jedinica (VHDL 1076/1993.) /Design unit / -> konstrukt koji može biti nezavisno analiziran i pohranjen u biblioteci. Oblikovana jedinica može biti deklaracija entiteta, tijelo arhitekture, deklaracija konfiguracije, deklaracija paketa ili pak deklaracija tijela paketa.
- default library -> work

```
LIBRARY library_name;
USE library_name.package_name.all;
```

Primjer:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
```

std_logic_1164 PAKET
devetvrijednosna logika

0
1
2
—

} za sintezu

L - weak 0
H - weak 1
U - uninitialized
X - unkown
W - weak unknown

} za simulaciju

KONFIGURACIJA (engl. configuration)
- lista dijelova našeg dizajna
CONFIGURATION – ključna (rezervirana) riječ – određuje koje se arhitekture povezuje s kojim entitetima

VHDL – „VHSIC“ – very high-speed integrated circuit – jezik za opisivanje sklopovlja

- IEEE standard 1076

1987. godina VHDL 87

1993. godina VHDL 93

<u>1-bitni komparator</u>

ulaz		izlaz
i0	i1	eq
0	0	1
0	1	0
1	0	0
1	1	1

Slika 1: Tablica istinitosti

$$eq = i0 \cdot i1 + i0' \cdot i1'$$

- Realizirati 1-bitni komparator uporabom osnovnih logičkih sklopova NOT; AND; OR; XOR

VHDL program:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY eq1 IS
        PART (
                i0, i1: IN std_logic;
                eq: OUT std_logic
              );
END eq1;
ARCHITECTURE sop_arch OF eq1 IS
        SIGNAL p0, p1 : std_logic;
BEGIN
--      sum of two product terms
        eq <= p0 OR p1;
--      product terms
        p0 <= (NOT i0) AND (NOT i1);
        p1 <= i0 AND i1;
END sop_arch;
```

- VHDL – neosjetljiv na velika i mala slova
- identifikatori (engl. identifier) su imena objekata / 26 slova, brojke , (_) / npr. i0, i1, data_bus1_enable; mora započinjati slovom

- komentar: -- teskt; npr. -- ovo je komentar
- ključne (rezervirane) riječi: **entity, signal, end**, …
- biblioteka i paket

Linije

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

pozivaju std_logic_1164 paket iz biblioteke ieee.

- paket i biblioteka omogućuju nam dodavanje dodatnih tipova, operatora, funkcija itd.

```
entity deklaracija
        entity eq1 is
                port (
                        i0, i1 : in std_logic;
                        eq : out std_logic
                      );
        end eq;
```

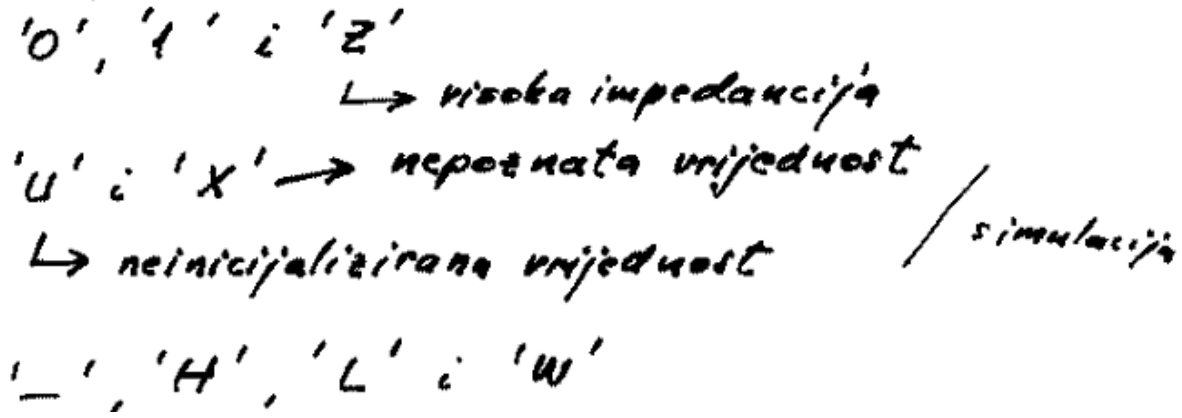
- specificira ulazno-izlazne (I/O) linije/signale (priključke) sklopa predočenog kao „crna kutija“.

Osnovni format za I/O deklaraciju priključaka (I/O port declaration):

signal.name1, signal_name2, … : mode
dana_type;

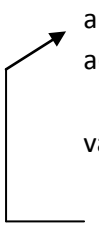
mode ε { **in, out, inout** } .

- Tipovi podataka i operatori:
  objekt mora imati specificirani tip podataka te se mogu rabiti samo deklarirane vrijednosti i operacije koje se mogu primijetiti na objektima
- predefinirani tip podataka pogodan za sintezu std_logic tip
  std_logic tip definiran je u paketu std_logic_1164 i ima devet vrijednosti:



- signal često sadrži veći broj bitova
  std_logic_vector tip podataka

Npr. 8-bitni ulaz a



a: ins td_logic_vector (7 downto 0);
a(1) -> jedan element u 1-dim. polju

varijanta:
a: in std_logic_vector (0 to 7);
! MSB je krajnji lijevi bit

Logički operatori: **not**, **and**, **or** i **xor** definirani za std_logic_vector i std_logic tip podataka

POZOR: and, or i xor imaju jednak prioritet u izvođenju – NUŽNO RABITI ZAGRADE !

Npr. (a **and** b) or (c **and** d)

Architecture body (Arhitektonsko tijelo)

```
architecture   sop_arch of eq1 is
        signal p0, p1, : std_logic;
begin
        -- sum of two product terms
        eq<p0 or p1;
        -- product terms
        p0 <= (not i0) and (not i1);
        p1 <= i0 and i1;
end sop_arch;
```

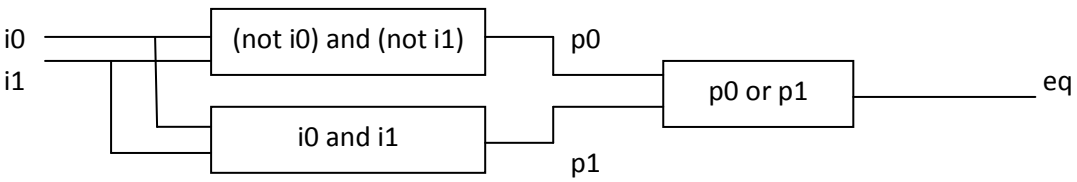
Opisuje djelovanje sklopa.

VHDL dopušta pridruživanje više arhitektura jednom entitetu (entity):

- arhitektonsko tijelo je identificirano imenom (sop_arch) /“sum-of-products architecture“)
- arhitektonsko tijelo može (opcija) sadržavati deklaracijski dio koji specificira konstante, interne signale i sl.

Npr. signal p0, p1 : std_logic;

Glavni opis smješten između **begin** i **end** sadrži **KONKURENTNE NAREDBE** koje se izvode **paralelno**!
Analogija sklopovlju:



- redoslijed konkurentskih naredbi je nevažan!

2-bitni komparator

a i b ulazi u 2-bitni komparator
aeqb - izlaz

VHDL program:

```
library ieee;
use ieee.std_logic_1164.all;
entity eq2 is
   port (
        a, b : in std_logic_vector
                                (1 downto 0);
        aeqb : out std_logic
        );
end eq1;

architecture sop_arch of eq1 is
        signal p0, p1, p2, p3 : std_logic;
begin
   -- sum of product terms
aeqb <= p0 or p1 or p2 or p3;
  p0 <= ((not a(1)) and (not b(1))) and ((not a(0)) and (not b(0));
  p1 <= ((not a(1)) and (not b(1))) and (a(0)) and b(0));
  p2 <= ( a(1) and  b(1)) and ((not a(0)) and (not b(0));
  p3 <= ( a(1) and  b(1)) and ( a(0)) and  b(0));
end sop_arch;
```

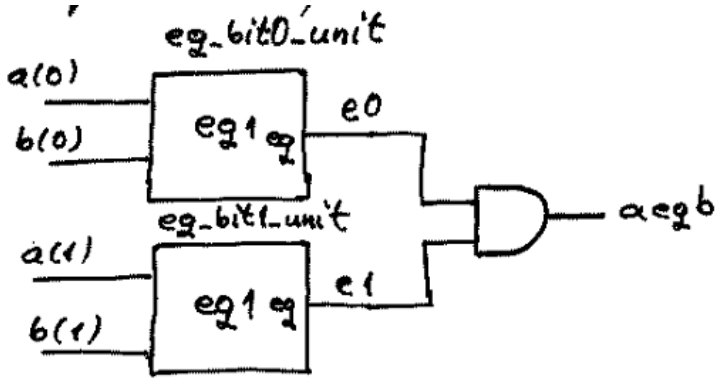
Strukturni opis

- digitalni sustav se vrlo često sastavlja od nekoliko manjih (pod)sustava

VHDL nudi mehanizam poznat pod nazivom **component instantiation** (prijedlog naziva „primjerkovanje“ / kojim podržava

Tip koda -> strukturni opis

Alternativni dizajn 2-bitnog komparatora -> upotreba prethodno oblikovanog sklopa 1-bitnog komparatora



VHDL program:

/ arhitekturno tijelo /

```
architecture struc_arch of eq2 is
        signal e0, e1 : std_logic;
begin
  -- primjerkovanje dva 1-bitna komparatora
  eq_bit0_unit : entity work.eq1 (sop_arch)
        port map (i0 => a(0), i1 => b(0), eq => e0);
  eq_bit1_unit: entity work.eq1 (sop_arch)
        port map (i0 => a(1), i1 => b(1), eq => e1);
  -- a  and  b are equal if individual bits are equal
  aeqb <= e0 and e1;
end struct_arch;
```

Primjerkovanje dviju komponenti pomoću koda sa sintaksom:

```
unit_label: entity lib_name.entity_name (arch_name)
        port map (
              formal_signal => actual_signal,
              formal_signal => actual_signal,
                 …
        );
```

Npr.

```
eq_bit0_unit: entity work.eq1 (sop_arch)
        port map (
              i0 => a(0), i1 => b(0), eq => e0
        );
```

biblioteka work (default)

eq1 ime entiteta

sop_arch ime arhitekture

Strukturni opis za VHDL – 87

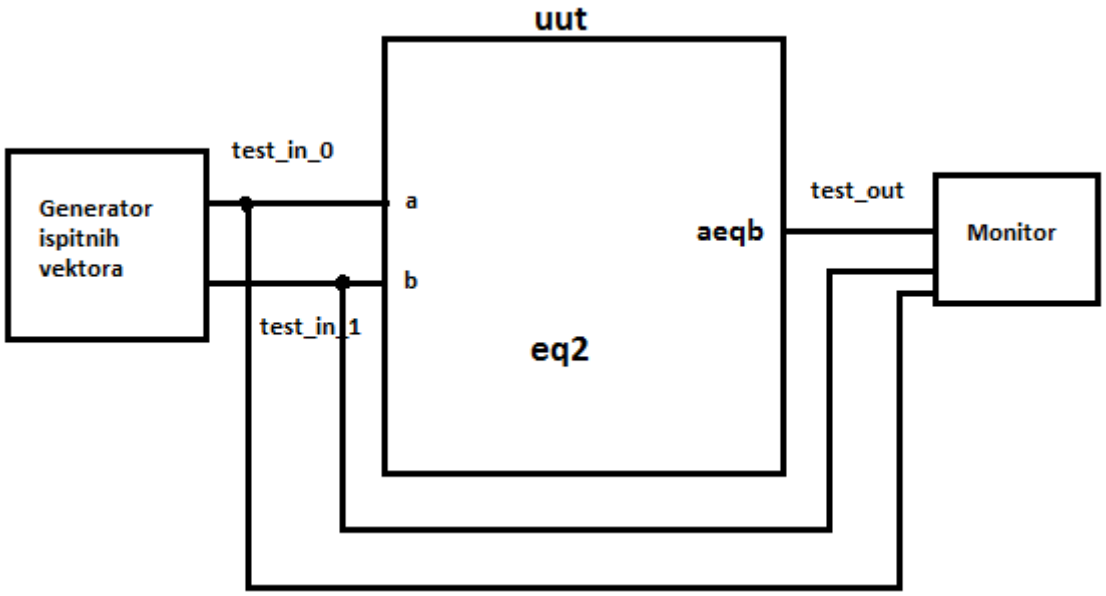
```
architecture vhd_87_arch of eq2 is
    -- deklaracija komponente
    component eq1
          port (
                    i0, i1 : in std_logic;
                    eq : out std_logic
                );
     end component;
     signal e0, e1: std_logic;
begin
    -- primjerkovanje dvaju 1-bit komp.
    eq_bit1_unit: eq1    -- use the declared name eq1
          port map (i0 => a(0), i1 => b(0), eq => e0);
    eq_bit1_unit: eq1
          port map (i0 => a(1), i1 => b(1), eq => e1);
   -- a  and  b  are equal if individual bits are equal
     aeqb <= e0  and  e1;
end vhd_87_arch;
```

Naredbu:

eq_bit0_unit : enitity work.eq1 (sop_arch) zamjenili deklariranim imenom komponente

eq_bit0_unit: eq1

Ispitivanje sklopa – ispitni program (Testbench)



- Nakon oblikovanja koda -> simuliramo sklopovlje da verificiramo ISPRAVNOST DJELOVANJA SKLOPOVLJA
- pišemo poseban ispitni program (engl. testbench)

Komponente:

uut – unit under test

test vector generator -> generator ispitnih vektora

monitor – ispituje izlazne odgovore sklopovlja

ispitni program za 2-bitni komparator:

```
library ieee;
use ieee.std_logic_1164.all;
entity eq2_testbench is
end eq2_testbench;

architecture tb_arch of eq2_testbench is
    signal test_in0, test_in1: std_logic_vector (1 downto 0);
    signal test_out: std_logic;
begin
    -- primjerkovanje sklopa koji se ispituje
    uut : entity work.eq2 (struc_arch)
        port map (a =>test_in0, b=>test_in1, aeqb =>test_out);
    -- generator ispitnih vektora
process
begin
    -- test vektor 1
    test_in0 <= „00“;
    test_in1 <= „00“;
    wait for 200ns;
    -- test vektor 2
    test_in0 <= „01“;
    test_in1 <= „00“;
    wait for 200ns;
    -- test vektor 3
    test_in0 <= „10“;
    test_in0 <= „11“;
    wait for 200ns;
    -- test vektor 4
    test_in0 <= „10“;
    test_in0 <= „10“;
    wait for 200ns;
    -- test vektor 5
    test_in0 <= „10“;
    test_in0 <= „00“;
    wait for 200ns;
    -- test vektor 6
    test_in0 <= „11“;
    test_in0 <= „11“;
```

```
        wait for 200ns;
        -- test vektor 7
        test_in0 <= „11“;
        test_in0 <= „01“;
        wait for 200ns;
    end process;
    end tb_arch;
```

Programski kod sastoji se od naredbe za primjerkovanje (engl. instantiation statement) koja kreira primjerak 2-bitnog komparatora i naredbe process kojom se generira sekvenca ispitnih uzoraka (test vektora).

Naredba **process** je VHDL konstrukt u kojoj se naredbe izvode SLJEDNO (sekvencijalno)

- Svaki se ispitni uzorak generira trima naredbama;

npr.

```
-- test vektor  2
test_in0 <= „01“;  }   specificiraju vrijednosti za test_in0 i test_in1 signale
 test_in0 <= „00“; }
wait for 200ns;    }   pokazuje da će vrijednosti „trajati“ 200 ns
```

Pozor:

- monitor nema programski kód.

↓

Možemo promatrati uzlazne i izlazne valne oblike na prikazanoj jedinici simulatora („virtualni logički analizator“)

- VHDL omogućava precizno definiranje generatora ispitnih uzoraka te oblikovanje monitora