Laboratorij za načrtovanje integriranih vezij

Univerza *v Ljubljani*
Fakulteta *za elektrotehniko*

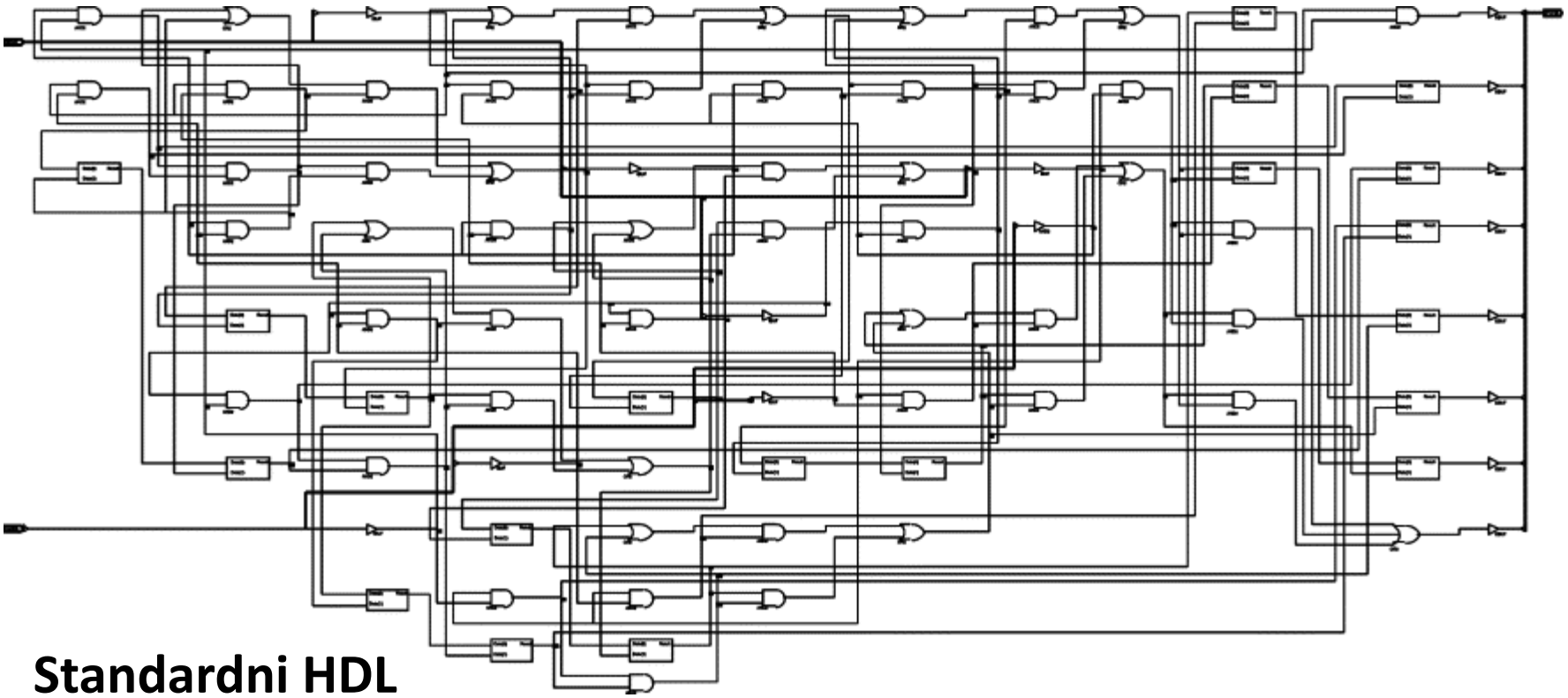
Digitalni elektronski sistemi

strojno-opisni jezik

ravni modeliranja VHDL, SHDL in orodja

Zakaj strojno-opisni jezik?

učinkovitost opisa > shema



Standardni HDL

- VHDL
- Verilog
- SystemVerilog

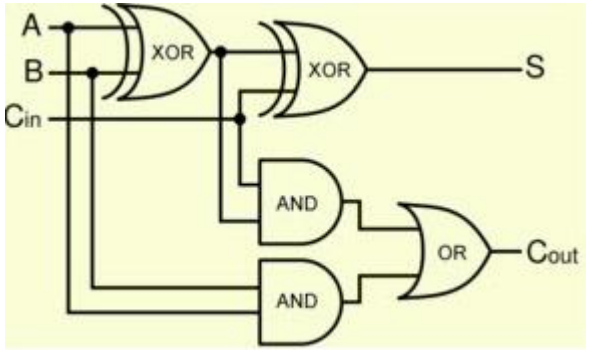
Ravni modeliranja v HDL

- postopkovna (behavioral) – opis algoritma
- funkcijska (dataflow, RTL) – opis zgradbe z operatorji
- logična – opis zgradbe z logičnimi operacijami

Opis vezja na ravni logičnih vrat

- Elementi opisa: signali (poimenovane povezave) in vrata
- Opis logične sheme v strojno-opisnem jeziku
- npr. polni seštevalnik v jeziku Verilog:

```
module full_adder(a,b,cin,s,cout);
 input a, b, cin;
 output s, cout;

 xor(x, a, b);
 xor(s, x, cin);
 and(c1, a, b);
 and(c2, cin, x);
 or(cout, c1, c2);

endmodule
```



Primer funkcijskega modela vezja

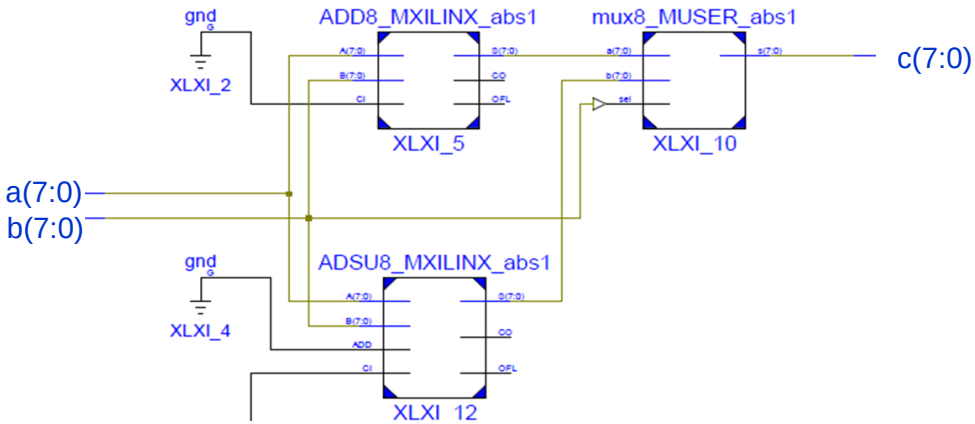
- Stavki opisujejo pretok in preoblikovanje podatkov
  - stavki, ki opisujejo gradnike vezja se izvajajo paralelno

```
entity fun is
  port ( a, b : in unsigned(7 downto 0);
           c : out unsigned(7 downto 0));
end fun;
```

```
architecture one of fun is
begin
  c <= a+b when b>0 else a-b;
end one;
```

- VHDL

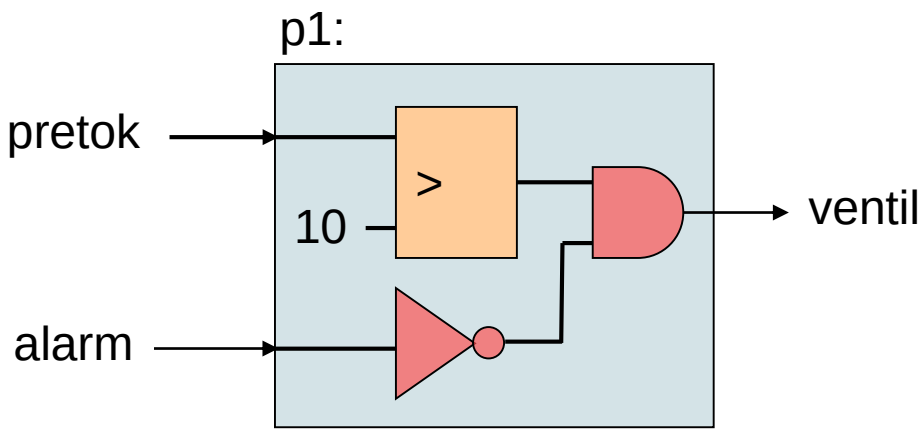
- operatorji +, - so kombinacijska vezja: seštevalnik, odštevalnik
- izbirni stavek when … else je izbiralnik



Postopkovni model vezja

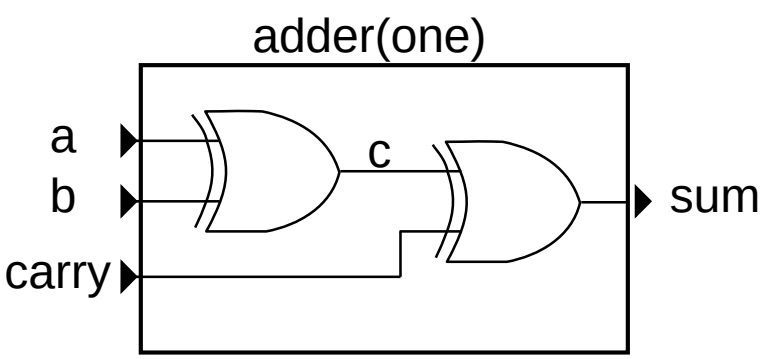
- Jezik VHDL: v procesu opišemo delovanje vezja
  - zgradbo vezja določi program za sintezo vezij

```
arhitektura
p1: process
ventil <= '0';
if pretok > 10 then
    ventil <= '1';
end if;
if alarm = '1' then
   ventil <= '0';
end if;
end process;
```

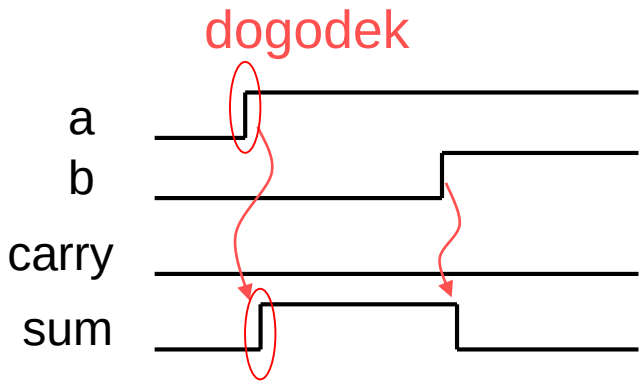


Obravnava modelov vezij

```
architecture logic of adder is
  signal c : std_logic;
begin
    sum <= c xor carry;
    c <= a xor b;
end logic;
```

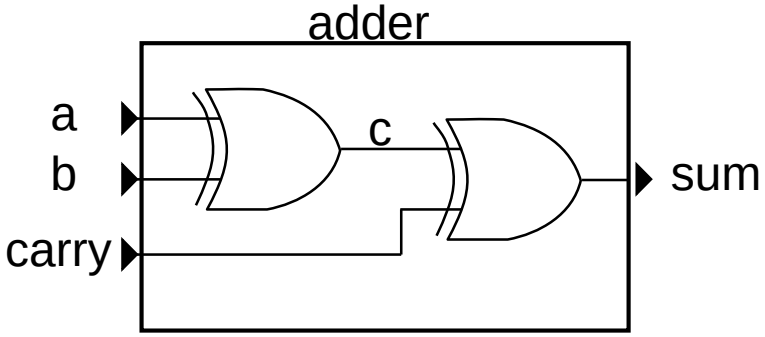


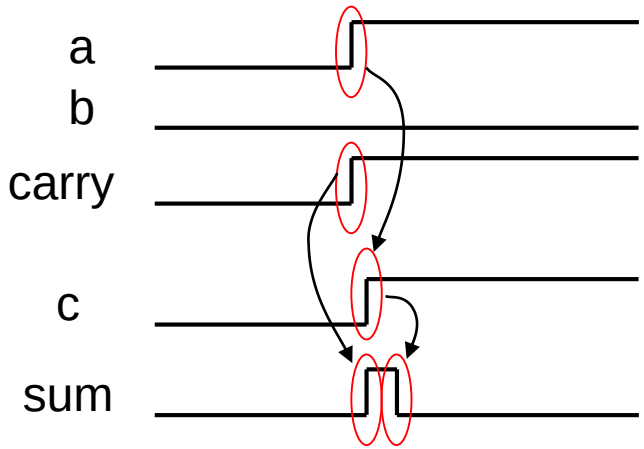
Sintetizirano vezje



Graf simulacije (waveform)

Potek simulacije





seznam dogodkov:
a: 0, 1 (T) carry: 0, 1 (T)

korak	**čas**	a	b	carry	sum	c
seznam	0	0, 1(T)	0	0, 1(T)	0	0
izvrši	T	1	0	1	0	0
izračunaj	T	1	0	1	1(T+Δ)	1(T+Δ)
izvrši	T+Δ	1	0	1	1	1
izračunaj	T+Δ	1	0	1	0(T+2Δ)	1
izvrši	T+2Δ	1	0	1	0	1

VHDL

Proces načrtovanja

1. razumeti nalogo
2. izdelati model
   - sintaksa
   - semantika
   - tehnološka implementacija
3. preveriti načrt
   - simulacija, prototip vezja

ključne besede:
"library","use","all","entity","port",
"in","out","is","begin","end"…
pravila: ločila, pretvorba tipov

sočasni stavki
več prireditev
procesni blok
fronta ure

poraba strojnih virov
cena operatorjev
zapahi / flip-flopi

Števec v jeziku VHDL

Osnova : Opis

```
cnt <= cnt+1
```

deklaracije:

Velikost cnt

ura…

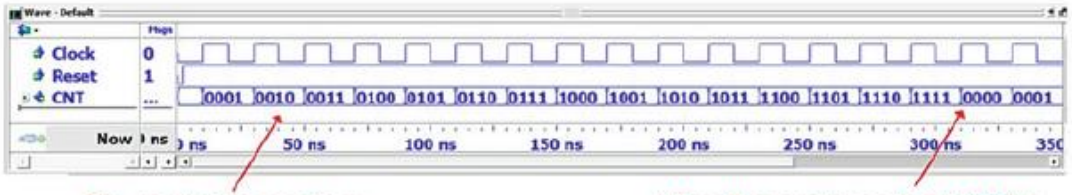
Example: 4-Bit Binary Up Counter in VHDL Using the Type UNSIGNED

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity Counter_4bit_Up is
 port (Clock, Reset : in  std_logic;
       CNT          : out unsigned(3 downto 0));
end entity;

architecture Counter_4bit_Up_arch of Counter_4bit_Up is

 signal CNT_tmp : unsigned(3 downto 0);

 begin

  COUNTER : process (Clock, Reset)
    begin
      if (Reset = '0') then
        CNT_tmp <= "0000";
      elsif (Clock'event and Clock='1') then
        CNT_tmp <= CNT_tmp + 1;
      end if;
  end process;

  CNT <= CNT_tmp;

end architecture;
```

The numeric_std package is needed to include the "+" operator. This operator only works on types signed/unsigned, so we will define the output CNT as type unsigned.

An internal signal is needed to support assignments in the form C <= C+1; because a port cannot be used as an argument in a signal assignment.

A concurrent signal assignment is used to continually assign CNT_tmp to CNT.



The counter increments on each rising edge of clock.

When the counter reaches "1111", it rolls over to "0000" and continues.

B.J. LaMeres: Quick Start Guide to VHDL, Springer, 2019

Small HDL

- Poenostavljena sintaksa
  - enostaven model vmesnika in deklaracije
  - prireditve (=, <=), **`when-else, if`**
  - ne zahteva pretvorb tipov (celoštevikske vrednosti)
  - ne zahteva procesa, sinhrona logika
- Obseg
  - signali: eno-bitni, signed, unsigned, naštevni, 1D zbirka
  - vrednosti: cela števila, binarne (`'1'`) in nizi (`"101"`)
  - operatorji: **`and, or, not, xor`**, `+, -, *, &,` **`sll, srl`**
    - vektorski: `a(0), b(3:0), c(7` **`downto`** `0)`

Primer: seštevalnik

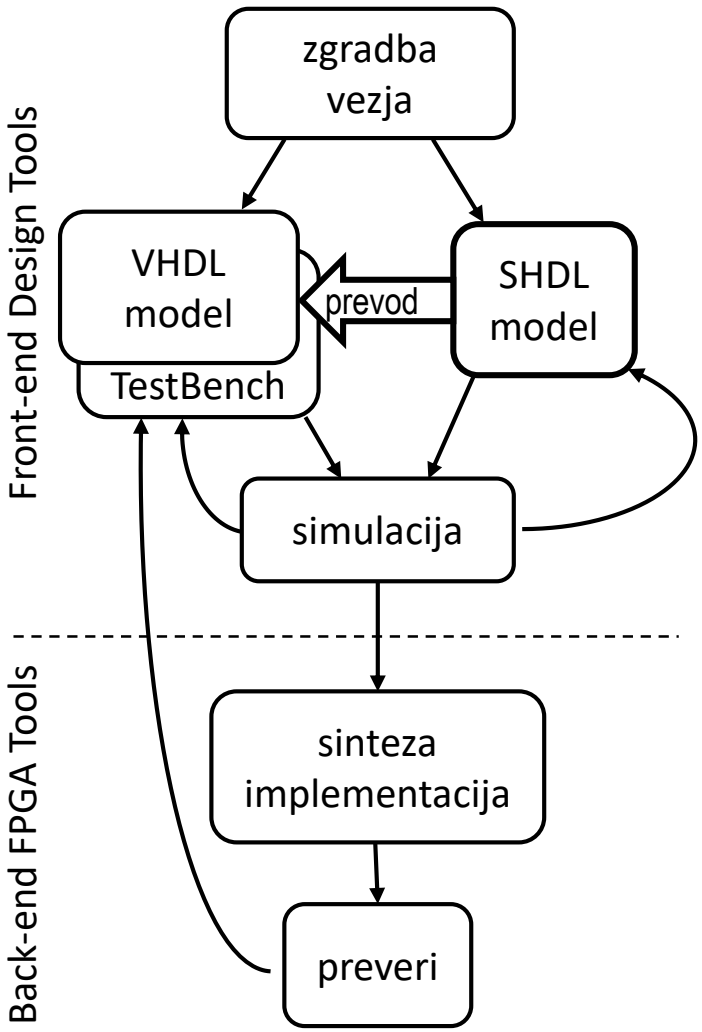
SHDL :

```
entity add
 a: in u10;
 b: in u5;
 sum: out u11;
begin
 sum = a+b
end
```

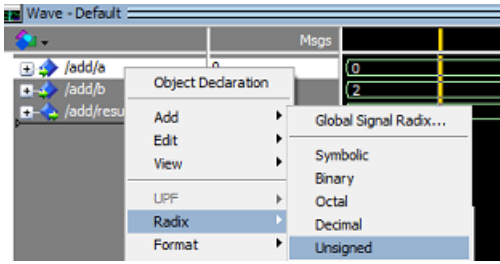
VHDL

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity add is
 port (
   a : in unsigned(9 downto 0);
   b : in unsigned(4 downto 0);
   sum : out unsigned(10 downto 0) );
end add;

architecture RTL of add is
begin
sum <= resize(a,11) + resize(b,11);

end RTL;
```

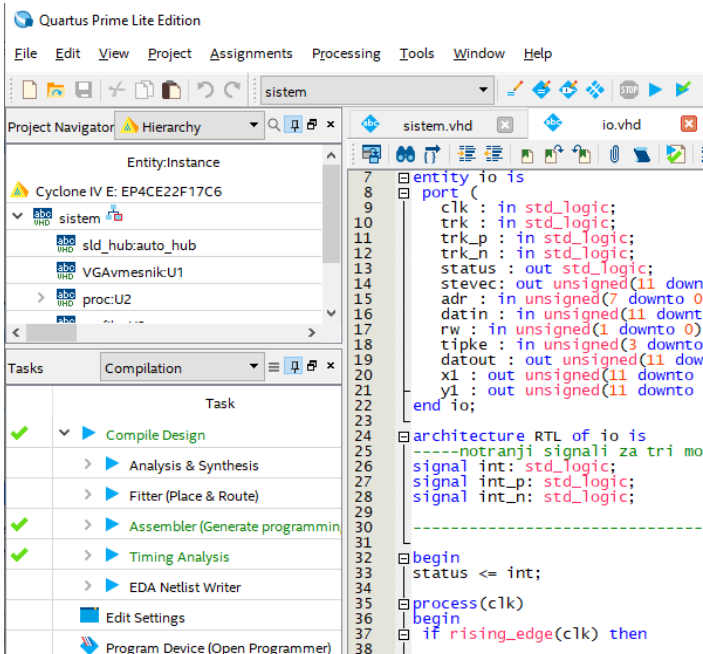
Potek načrtovanja digitalnih vezij



ModelSim

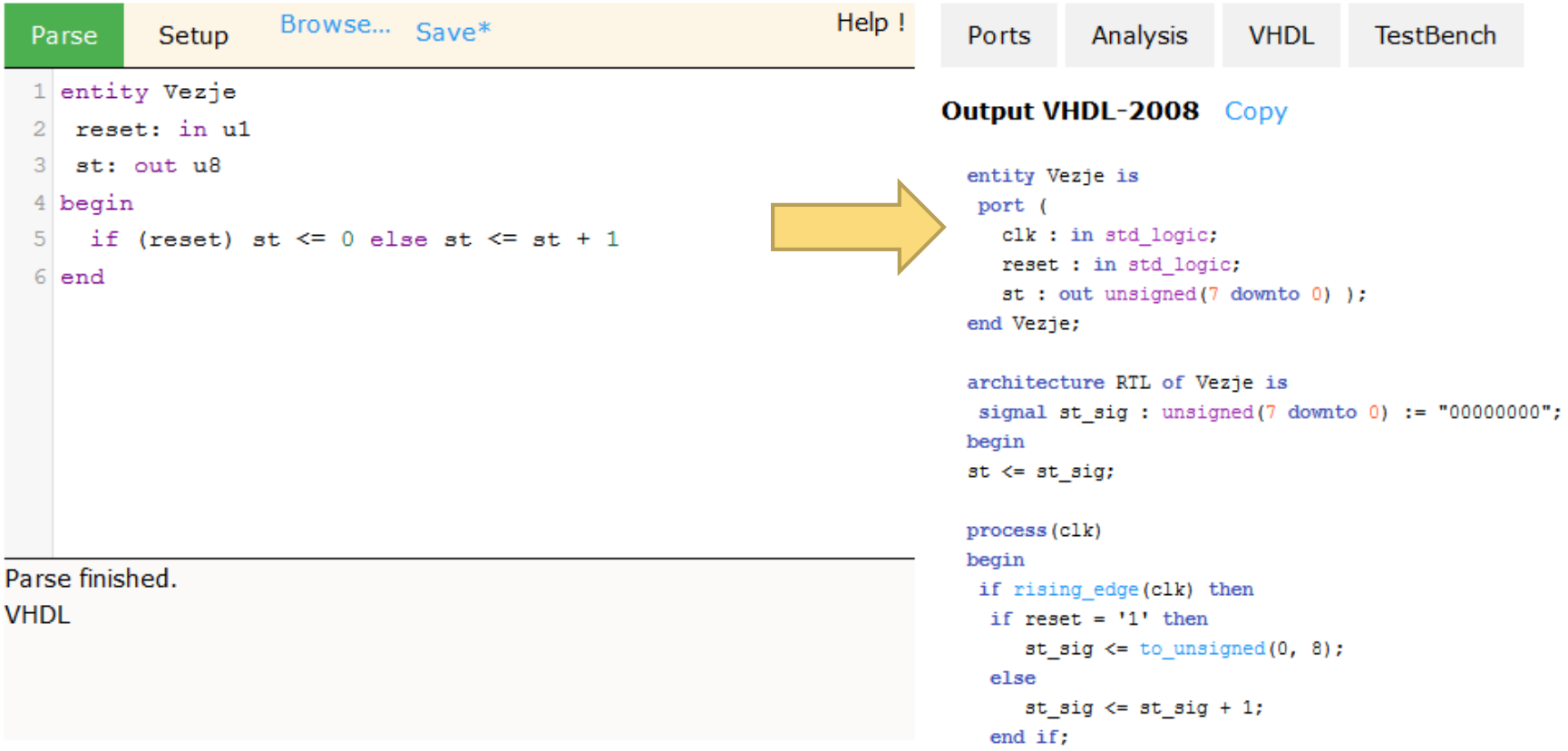


Intel Quartus



Spletno orodje SHDL

lniv.fe.uni-lj.si/shdl/

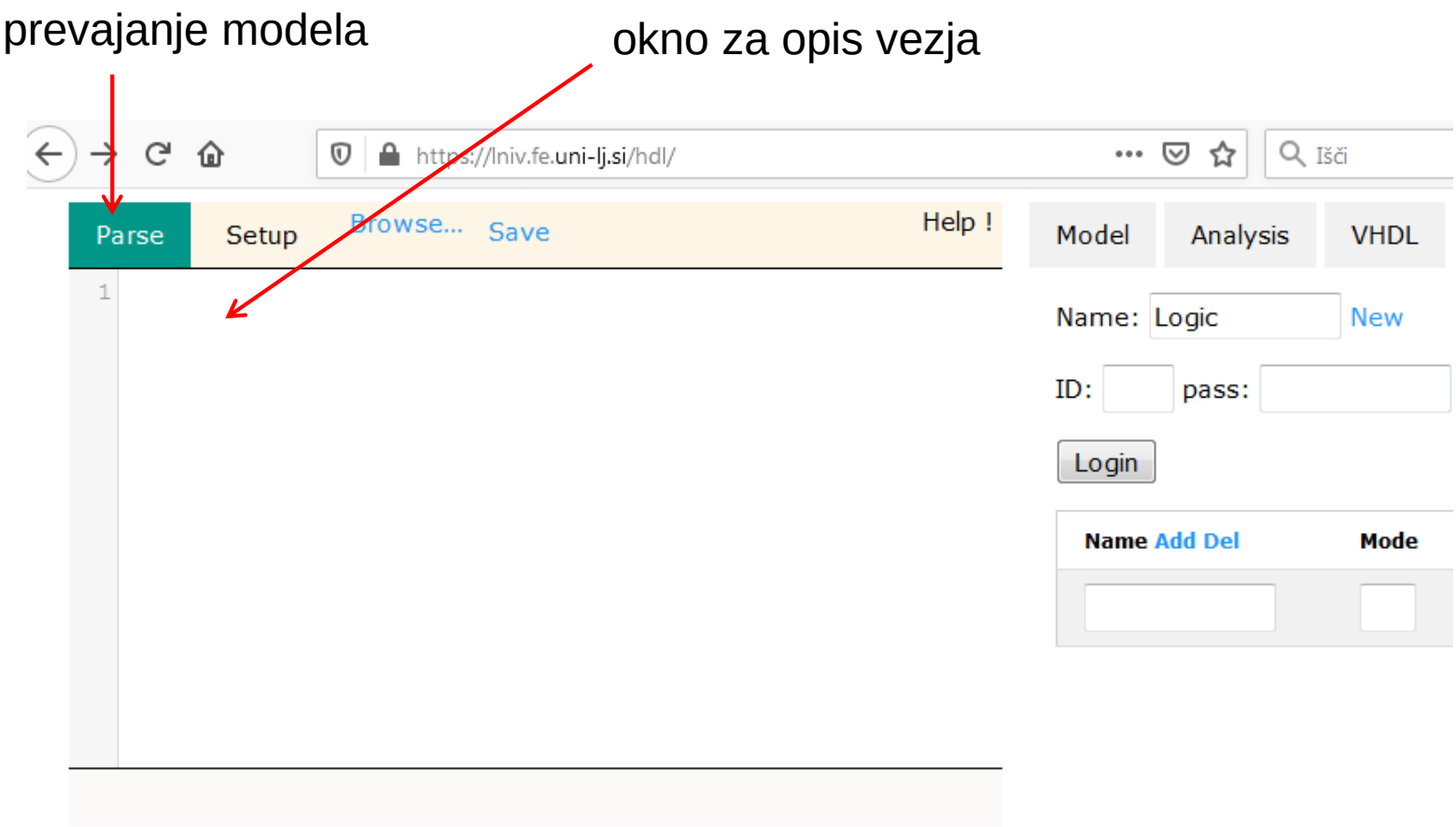


Jezik in orodje razvito v laboratoriju na FE za hitrejše učenje VHDL

- poenostavljena vhodna sintaksa (deklaracija, prireditev, if)
- podatkovni tipi: eno-bitni: u1 in vektorski u8, s8, …
- avtomatsko prevede v VHDL-1993 ali 2008
- načrtovanje in simulacija v brskalniku (Firefox, Chrome)

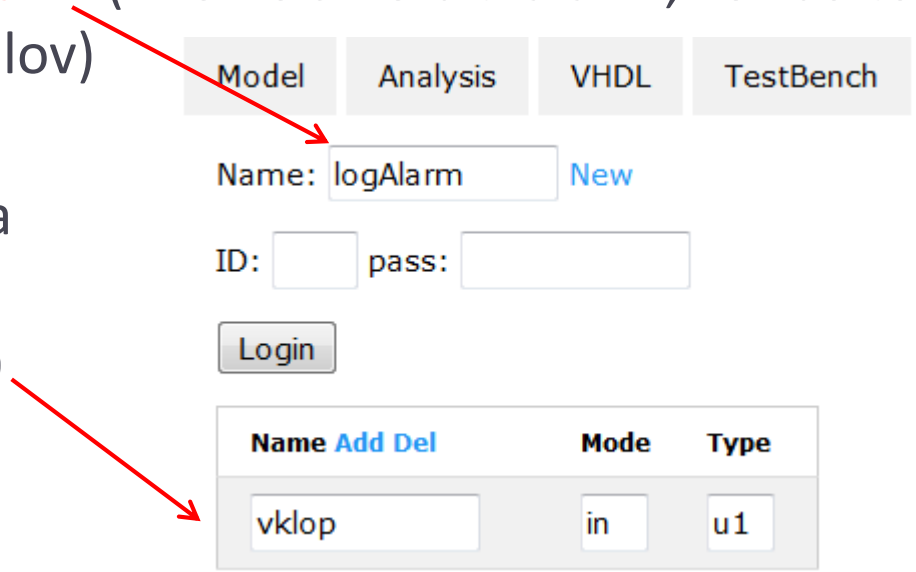
Kje dobim orodje ?

- v brskalniku Firefox ali Chrome odpri: https://lniv.fe.uni-lj.si/shdl/
- na levi strani je okno za opis vezja v učnem jeziku



Priprava

- Opis modela vezja pripravimo tako, da določimo ime vezja (ena beseda, brez šumnikov) in priključke
- Primer izdelave kombinacijskega vezja za alarm:
  - kadar je vhod vklop=1 in je na logični 1 tudi eden izmed vhodov v1-v4, naj bo izhod alarm=1
  - določimo ime vezja: logAlarm (ime ne sme biti alarm, ker bo to oznaka enega izmed signalov)
  - napišemo v tabelo ports: vklop in u1 (to pomeni, da imamo signal z imenom vklop, ki je enobitni vhod)



Priprava …

- nato kliknemo add in dodamo še nekaj vrstic
  - v1,v2,v3,v4 so štirje enobitni vhodi
  - alarm je enobitni izhod
  - vrata pa je enobitni vmesni signal
- kliknemo new in potrdimo vnos novega opisa vezja
  - v glavnem oknu se pojavi koda, ki predstavlja ogodje opisa vezja v jeziku SHDL

Name Add Del	Mode	Type
vklop	in	u1
v1,v2,v3,v4	in	u1
alarm	out	u1
vrata		u1

Parse | Setup | Browse...

```
1 entity logAlarm
2  vklop: in u1;
3  v1,v2,v3,v4: in u1;
4  alarm: out u1;
5  vrata:  u1;
6 begin
7
8 end
9
```

med begin in end dodamo stavke,
ki opisujejo zgradbo ali delovanje vezja

Opis vezja

- Vezje bomo opisali z dvema prireditvenima stavkoma
  - delovanje: kadar je vhod vklop=1 in je na logični 1 tudi eden izmed vhodov v1-v4, naj bo izhod alarm=1
  - eden predstavlja 4-vhodna OR vrata, drugi pa AND s katerim določimo končni izhod

Parse | Setup | Browse... | Save*

```
1  entity logAlarm
2   vklop: in u1;
3   v1,v2,v3,v4: in u1;
4   alarm: out u1;
5   vrata:  u1;
6  begin
7   vrata = v1 or v2 or v3 or v4
8   alarm = vrata and vklop
9  end
10
```

vrstni red stavkov, ki opisujeta vezje ni pomemben! (poskusi)

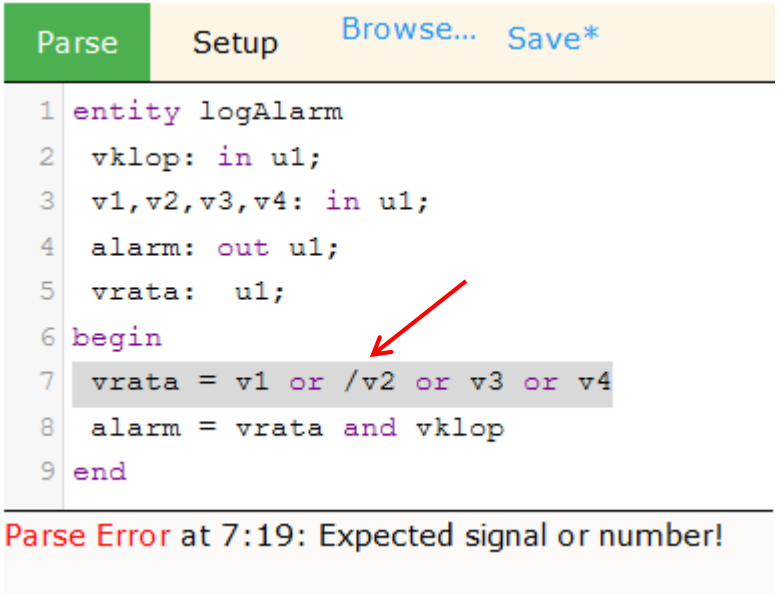
prireditvenim stavkom z operatorjem = pravimo sočasni stavki

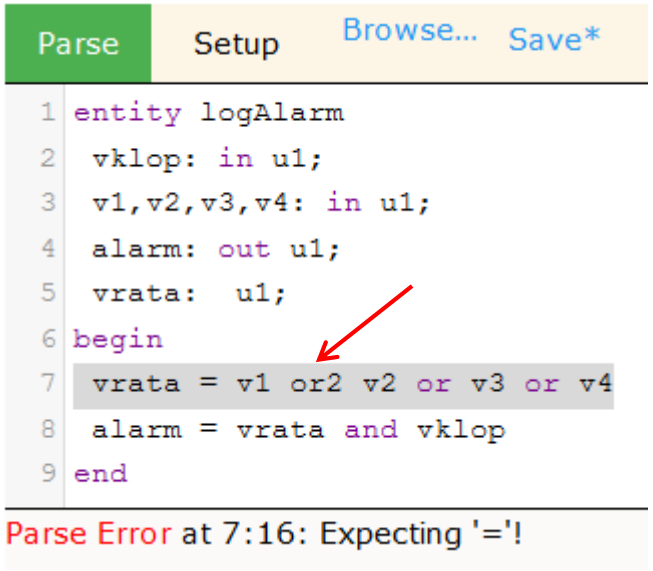
Nekaj osnovnega o sintaksi učnega jezika

- Sintaksa SDHL je v primerjavi z VHDL zelo poenostavljena
  - sintaksa jezika VHDL je gostobesedna
  - jezik SHDL pozna le osnovne konstrukte za modeliranje digitalnih kombinacijskih in sinhronih sekvenčnih vezij
- Ločila med stavki so presledki, konec vrstice ali podpičja
  - podpičje na koncu stavkov ni obvezno
- ime vezja in ime signalov ne sme biti rezervirana beseda v SHDL ali VHDL
  - urejevalnik že sam obarva rezervirane besede jezika
  - če je beseda obarvana je lahko tudi ena izmed rezerviran besed v jeziku VHDL, ki jo sicer SHDL ne zna prevesti!

Prevajanje in popravljanje napak

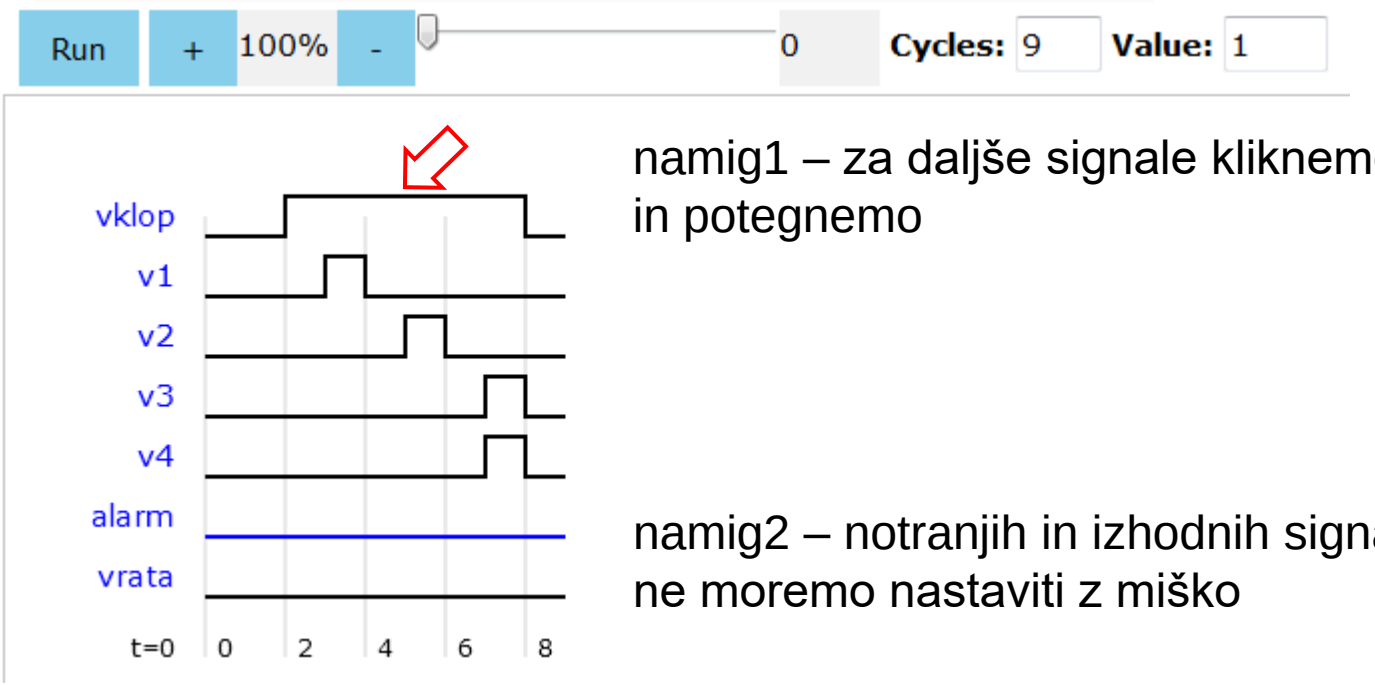
- Model prevedemo s klikom na Parse
- Če naredimo napako, dobimo opozorilo in osenčeno vrstico z napako
  - napako popravimo in ponovno prevedemo





Simulacija

- Preveden model vezja preizkusimo s simulacijo
  - signali za simulacijsko polje se berejo iz tabele Ports & Signals, kamor jih poljubno dodajamo oz. brišemo
  - ob spremembi modela je potrebno ponovno prevajanje (Parse)
- V simulatorju s klikanjem na graf nastavimo vhodne vrednosti (enobitna vrednost se ob vsakem kliku zamenja)

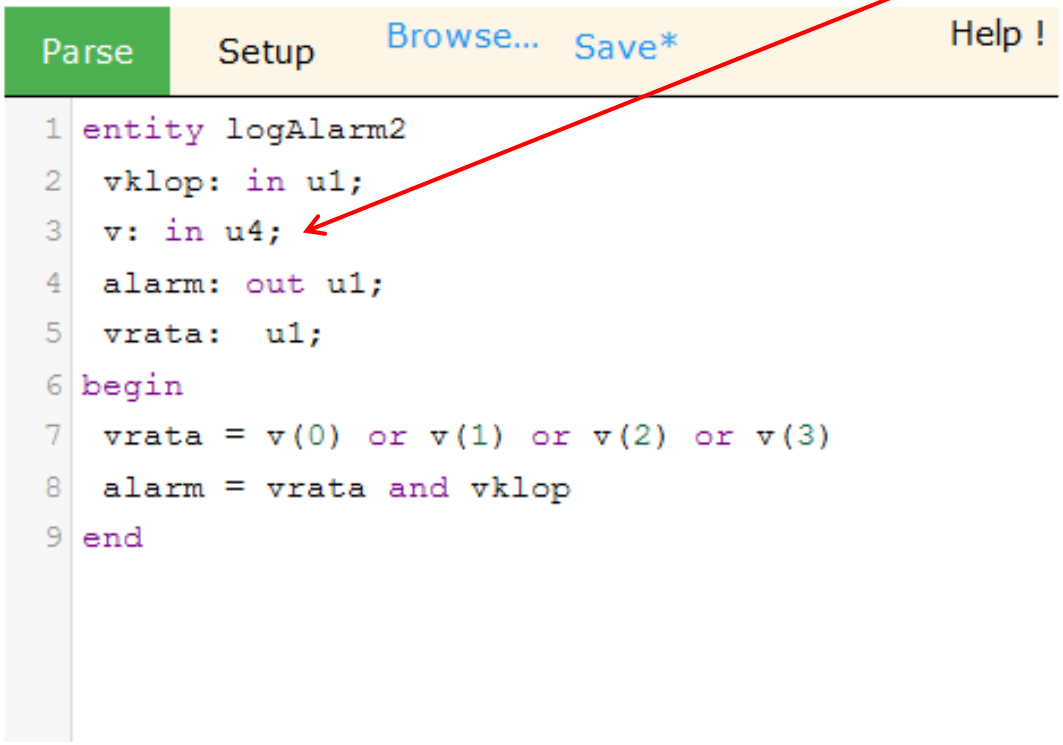


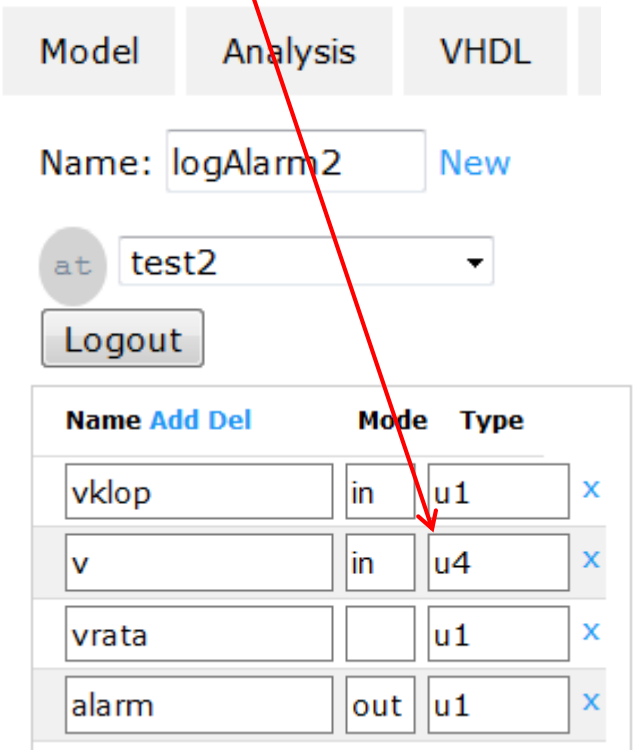
namig1 – za daljše signale kliknemo in potegnemo

namig2 – notranjih in izhodnih signalov ne moremo nastaviti z miško

Vektorski signali

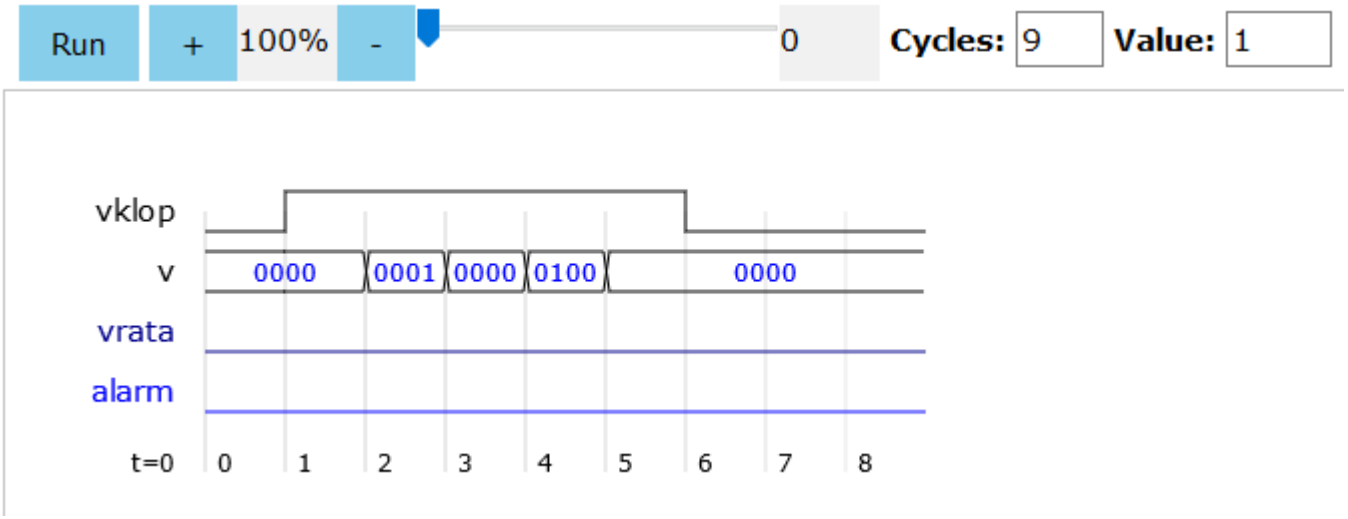
- Večbitne signale predstavimo z vektorji, npr. namesto v1,v2,v3,v4 deklariramo 4-bitni vektor: v in u4
  - elementi vektorja so v(0), v(1), v(2) in v(3)





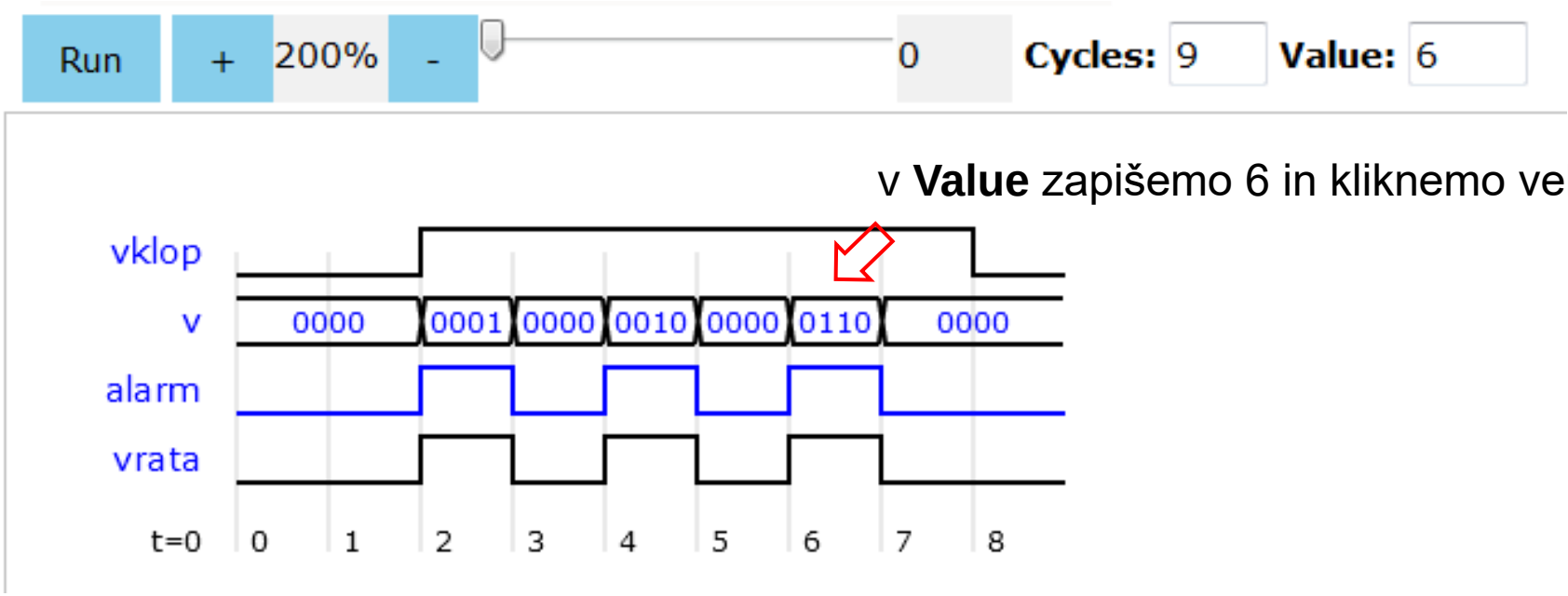
Simulacija…

- Dolžino simulacije nastavimo v polju **Cycles**, max=999
- Nastavimo vhode
  - ob kliku na signal se vrednost spreminja med 0 in nastavljeno
  - če želimo nastaviti vrednost za celoten čas simulacije, kliknemo na ime signala
- Za pomoč pri pregledu so gumbi za zoomiranje + in – ter premikanje



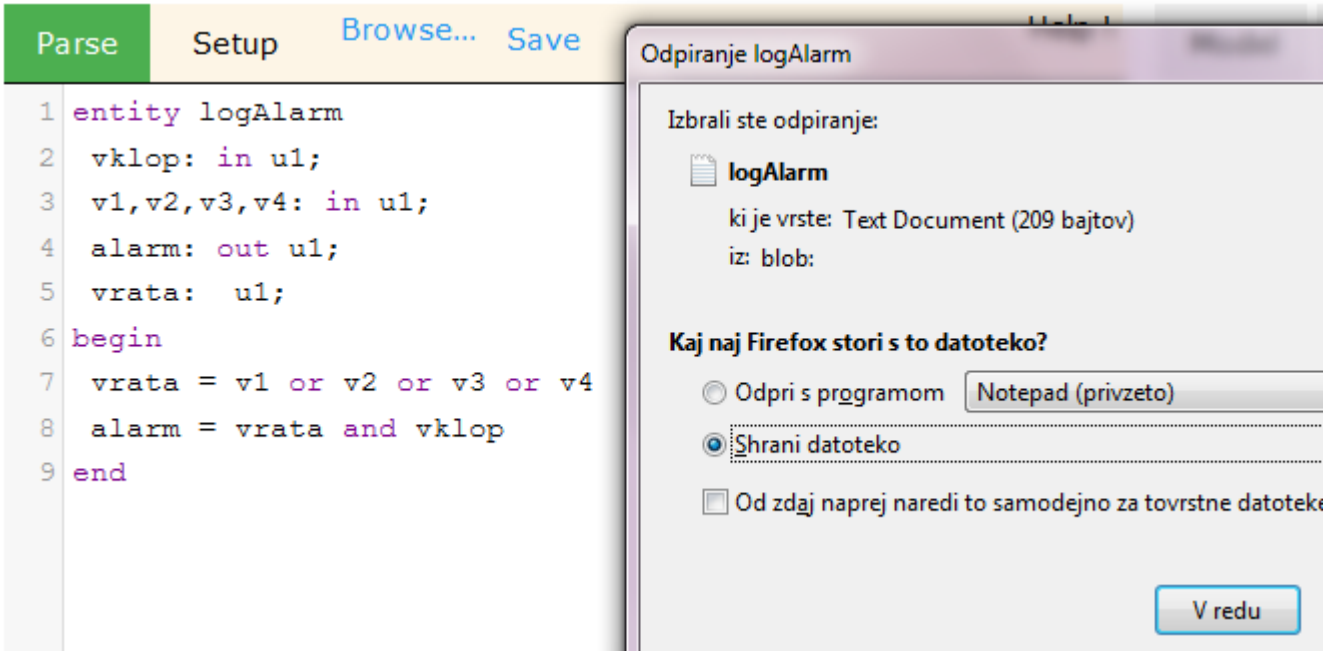
Simulacija z vektorji

- ob nastavljanju vhodov določa vrednost vektorja polje **Value**
- z desnim klikom na vektor spreminjamo prikaz med: desetiškim / dvojiškim / analognim
  - namig: za pregled dvojiškega prikaza uporabi zoom +



Shranjevanje modela na lokalni računalnik

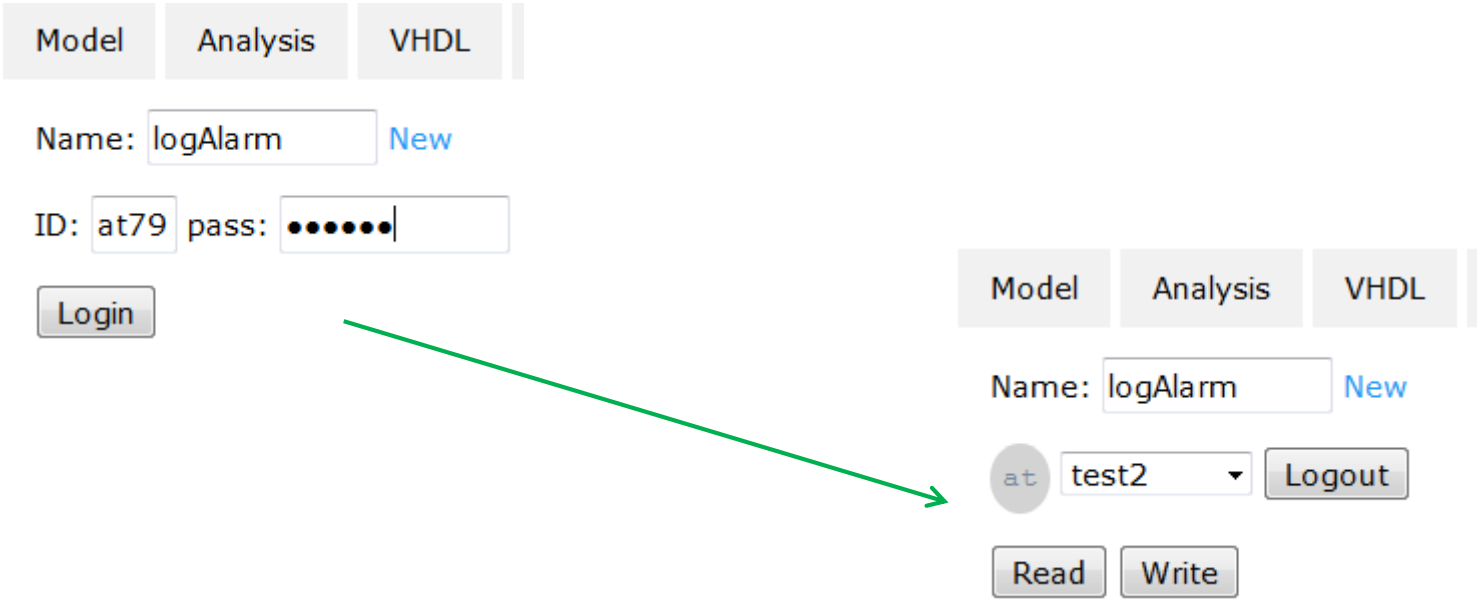
- Save zapiše tekstovni opis vezja v datoteko, ki jo brskalnik naloži na lokalni računalnik (shrani v Download)



- Browse... uporabi za nalaganje shranjene datoteke
  - namig: datoteka je berljiva z urejevalnikom kode, npr. Notepad++, Windows Notepad pa bo prikazal vsebino brez zaključkov vrstic

Shranjevanje na strežnik

- SHDL omogoča prijavo in shranjevanje v spletno bazo, kar bomo uporabili za pregled vaših izdelkov
- Prijavite se z uporabniškim imenom in geslom
  - ob uspešni prijavi dobite možnost branja iz baze in shranjevanja vaših modelov v bazo



Spletna prijava…

- Branje datotek
  - v okencu za izbiro datoteke boste imeli dva seznama:

  1. Vaje seznam datotek vaše skupine (npr. des, pds), ki vam jih pripravi asistent

  2. Saved je seznam datotek, ki ste jih sami shranili

- Shranjevanje opisa vezja
  - model vezja in nastavitev vhodnih signalov se shrani v zapisu z imenom vezja (Name:)
  - če delate nov model, uporabite drugo ime, da ne boste prepisali že obstoječ model
  - V3.7, če se ime pri zapisu novega modela ne spremeni, se odjavite in ponovno prijavite

VHDL

- V zavihku VHDL se po prevajanju modela pojavi VHDL opis modela v standardnem strojno-opisnem jeziku VHDL
  - uporabimo ga npr. za tehnološko preslikavo in nalaganje vezja na FPGA razvojno ploščo
- Zavihek TestBench pa predstavlja testno strukturo za simulacijo v jeziku VHDL

Parse | Setup | Browse... | Save | Help !

```
1 entity logAlarm
2  vklop: in u1;
3  v1,v2,v3,v4: in u1;
4  alarm: out u1;
5  vrata:  u1;
6 begin
7  vrata = v1 or v2 or v3 or v4
8  alarm = vrata and vklop
9 end
```

Model | Analysis | VHDL

Output VHDL-2008 Copy

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity logAlarm is
 port (
   vklop : in std_logic;
   v1, v2, v3, v4 : in std_logic;
   alarm : out std_logic );
end logAlarm;
```

Opis sekvenčnih vezij

- Dodajmo v alarm odštevalnik časa, da bo izhod aktiven le nekaj ciklov ure

```
entity logAlarm2
 vklop: in u1;
 v: in u4;
 alarm: out u1;
 vrata:  u1;
 cnt: u3
begin
 vrata = v(0) or v(1) or v(2) or v(3)
 if cnt>0 then
   cnt <= cnt - 1
   alarm = 1
 else
   alarm = 0
 end
 if vrata and vklop then
   cnt <= 5
 end
end
```

3-bitni števec cnt

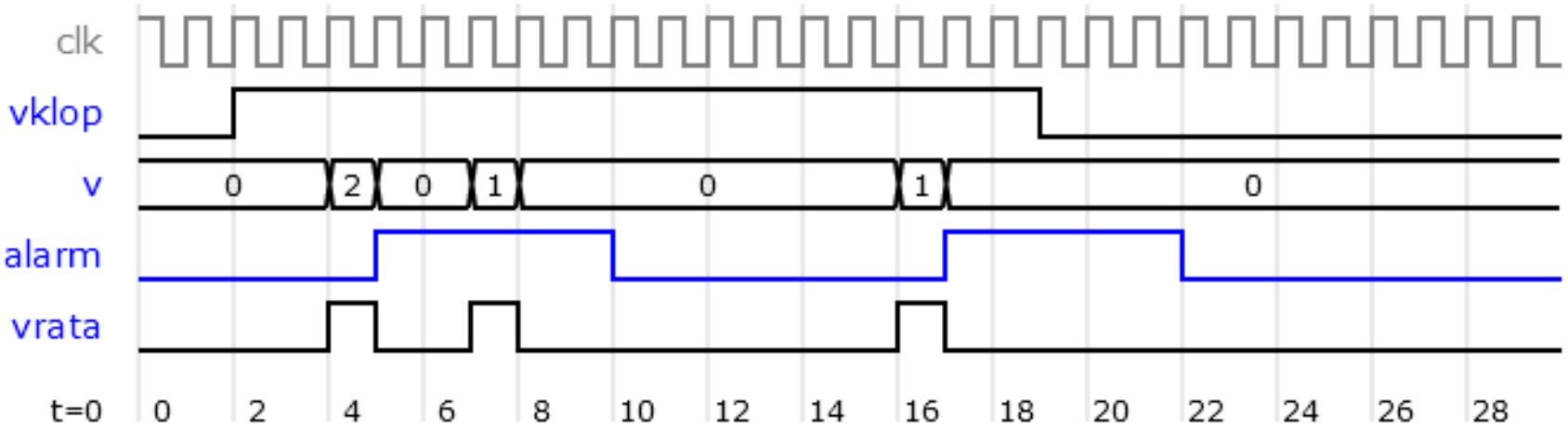
aktiviran alarm postavi števec na 5, nato pa se zmanjšuje do 0

števec naj se spreminja ob uri (clk), zato uporabimo prireditev z <=

prireditvenim stavkom z operatorjem <= pravimo sekvenčni stavki

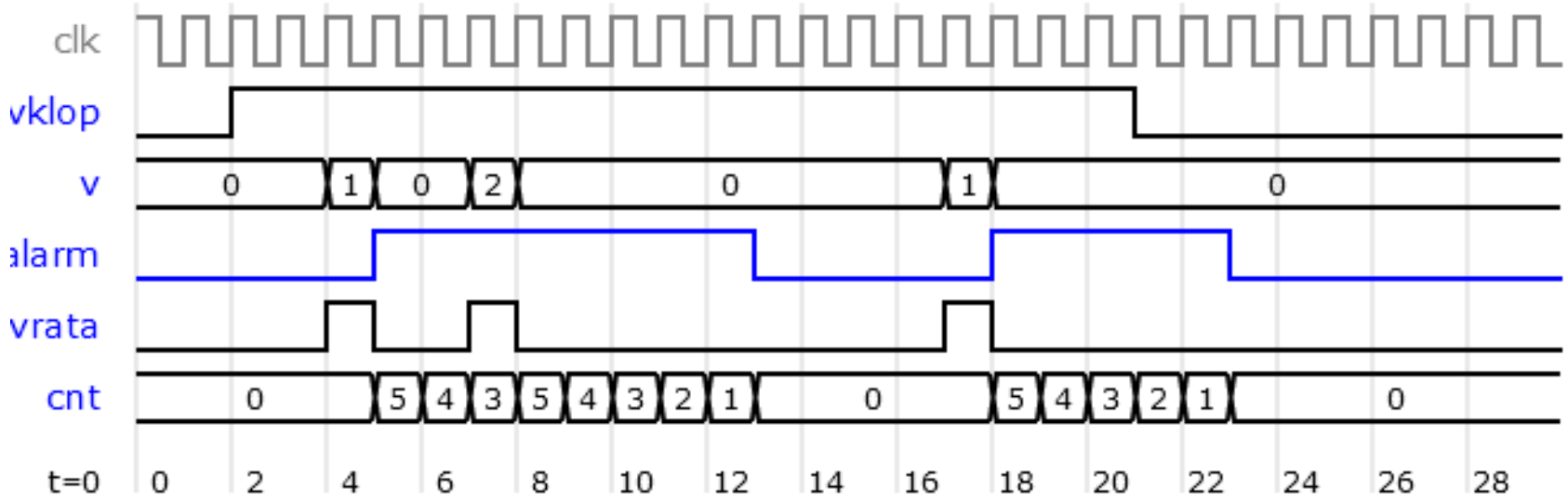
Simulacija

- na simulaciji se pojavi ura (clk)
- ob aktiviranju alarma bo šel izhod na 1 za 5 ciklov ure
  - nov signal cnt je potrebno dodati v tabelo signalov
  - ob večkratnem kliku na Run se simulacija nadaljuje od zadnjega shranjenega stanja vezja
  - za ponovno izvedbo simulacije od začetka najprej prevedi vezje



V razmislek

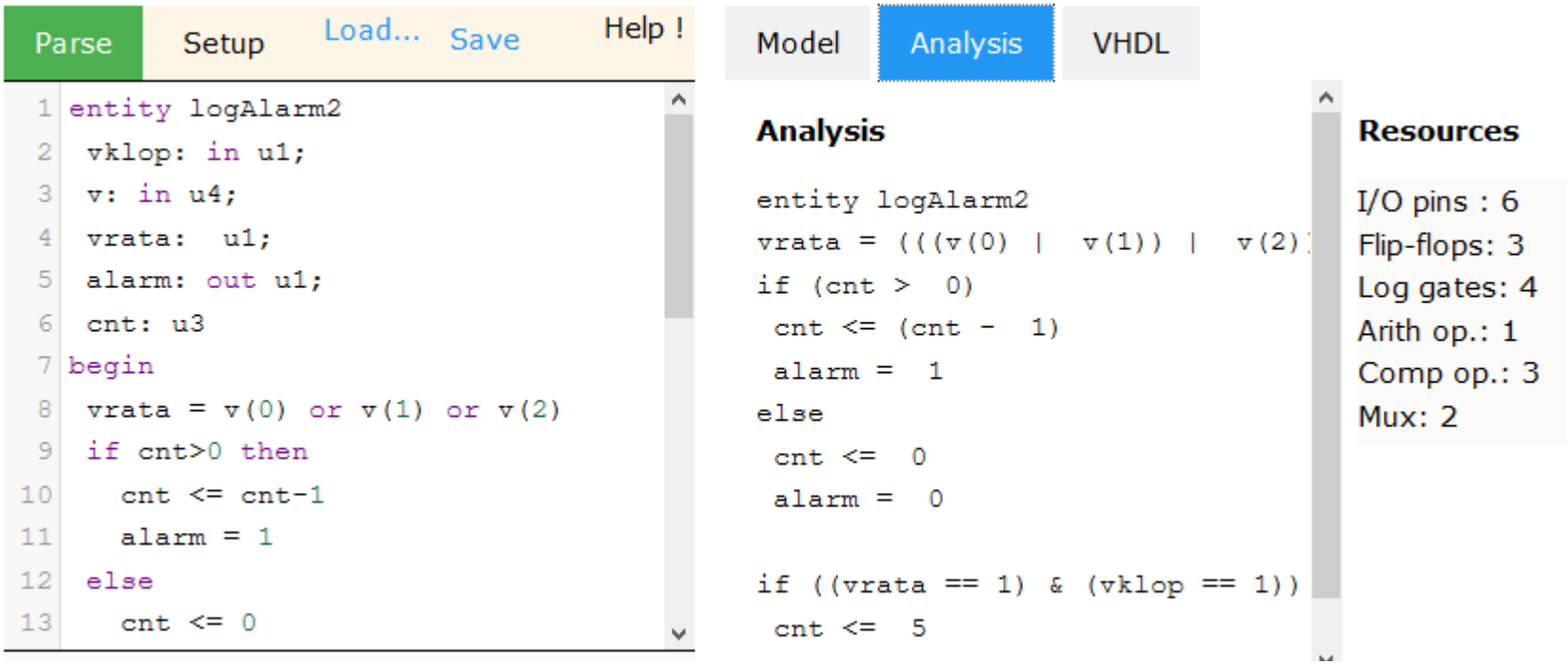
- Kaj se zgodi, če zamenjamo vrstni red if stavkov?
- poskusi razložiti...



Pomni: vrstni red sekvenčnih stavkov je pomemben.
Če so izpolnjeni pogoji za izvedbo več sekvenčnih prireditev istemu signalu, se dejansko izvede le zadnja prireditev!

Poročilo analize modela vezja

- zavihek Analysis: prikaz uporabljenih virov
  - število priključkov (I/O), flip-flopov, log. vrat, operacij in izbiralnikov (mux)



Avtomatsko generiran VHDL model

Parse | Setup | Load... | Save | Help !

```
1  entity logAlarm2
2   vklop: in u1;
3   v: in u4;
4   vrata:  u1;
5   alarm: out u1;
6   cnt: u3
7  begin
8   vrata = v(0) or v(1) or v(2) or v(3)
9   if cnt>0 then
10    cnt <= cnt-1
11    alarm = 1
12  else
13    cnt <= 0
```

Parse finished.
Generate VHDL.

Model | Analysis | VHDL

VHDL | TestBench | VCD | Copy

```
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity logAlarm2 is
 port (
   clk : in std_logic;
   vklop : in std_logic;
   v : in unsigned(3 downto 0);
   alarm : out std_logic );
end logAlarm2;

architecture RTL of logAlarm2 is
 signal vrata : std_logic;
 signal cnt : unsigned(2 downto 0) := "000";
begin
vrata <= ((v(0) or v(1)) or v(2)) or v(3);

process(cnt)
begin
  if cnt > 0 then
    alarm <= '1';
  else
     alarm <= '0';
  end if;
end process;
```

- TestBench: testna struktura za simulacijo
- VCD: zapis oblike signalov (Value Change Dump)

Povzetek

- Razloži ravni modeliranja digitalnih vezij.
  - Navedi primer v jeziku SHDL ali VHDL
- Kaj je strojno-opisni jezik?
  - Kaj lahko naredimo z modelom vezja?
  - Razloži delovanje simulatorja.
  - Opiši potek načrtovanja vezij s strojno-opisnim jezikom.