Univerzitet Džemal Bijedić Mostar

Fakultet informacijskih tehnologija Mostar

Mostar, 13.06.2010

AKS – Arhitektura kompjuterskih sistema

Seminarski rad: VHDL

(i ne samo o tome ...)

verzija: 01.00

Student: **Husremović Ernad, DL 2792**

1. Uvod

1.1 Razlozi za pisanje seminarskog rada

Student je položio u toku ranijeg studija ETF-u Sarajevo imao niz predmeta koji pokrivaju tematiku ovog predmeta.

Zbog razlike u nastavnim planovima, student treba obraditi temu VHDL-a.

1.2 Hardware, nekada davno ...

25 godina[1] se bavim informatikom.

Moj prvi računar je bio "kućni" računar ZX Spectrum 48[2], baziran na 8-bitnim procesorima Zilog Z80[3]. Kasnije ću kod prijatelja sretati Commodore 64[4], baziran na MOS 6510[5].

To je bila revolucija. Možeš imati računar u svojoj kući i programirati. Za razliku "običnih" elektronskih kola, kod računara si mogao, i nakon što su svi djelovi ploči zalemljeni, praviti nove funkcije. Mogao si **programirati**. Oooou. Granice moje mašte su se preko noći promjenile ....

1.3 A onda i ne tako davno ...

Kasnije ću tu "magiju" programiranja oprobati na PC računarima, Palm handheld-ovima ... Svaki je neku novu genijalnu stvar ... i nanovo izazivao ushićenje: " **... Šta bi sa ovim mogao uraditi** ... "

1.4 Otvoreni software

Sljedeća stvar velika nova stvar u IT industriji (za mene) nije bio hardware. Bio je to koncept otvorenog software-a (FSS[6] & OSS[7]). Nadalje ću koristiti termin otvoreni software. Otvoreni software ponovo je pomjerio limite razvoja IT industrije.

Otvoreni software je obezbjedio sljedeće:

1. postao je ozbiljan (i prijeko potreban) konkurent zatvorenom software-u[8]
2. Otvorio mogućnosti da i male firme, ili čak pojedinci, učestvuju u razvoju velikih softverskih projekata koje u zatvorenom modelu razvoja software-a jednostavno nije bio moguć.
   Internet je postao medij za komunikaciju armije developera ozbiljnih projekata baziranih na otvorenom software-u.

1.5 Primjer: Linux kernel projekat

Sinonim za eru otvorenog software-a je svakako Linux kernel projekat. Šta razlikuje linux kernel od zatvorenih sistema ?

U poslovnom smislu, taj resurs koriste na hiljade različitih kompanija koje su također konkurencija. Takvo što u svijetu zatvorenog software-a nikada nije postignuto. Sinergija znanja globalnih razmjera.

---

1 http://github.com/hernad/blog/blob/master/articles/25-godina-hardware-software-20100606.markdown

2 http://en.wikipedia.org/wiki/ZX_Spectrum
3 http://en.wikipedia.org/wiki/Z80
4 http://en.wikipedia.org/wiki/Commodore_64

5 http://en.wikipedia.org/wiki/MOS_Technology_6510

6 http://en.wikipedia.org/wiki/Free_software
7 http://en.wikipedia.org/wiki/Open-source_software

8 http://en.wikipedia.org/wiki/Proprietary_software

Da je hipotetčki linux kernel razvijala neka kompanija, ona bi morala obezbjediti budžet reda veličine bilion dolara, a o potrebnom vremenu da ne govorimo.

Diskutabilno je da li bi takav projekat mogao napraviti neku konkurentsku prednost takav projekat mogao napraviti u odnosu na ustanovljene tržišne lidere.

1.5 A šta je sa hardware-om ?

Pretpostavljam da je strpljenje čitaoca već pri kraju. Šta ovaj čovjek priča ?! Kakve veze ima otvoreni software sa VHDL-om ?

E pa ima.

Postojeći hardware ima sljedeće atribute koji su potpuno analogni **zatvorenom** software-u:

1. radi se o proizvodima visoke tehnologije ("high level knowledge")
2. Da bi ušao u tržište razvoja hardware-a moraš imati budžet koji je veći od 10 budžeta BiH

Prva tačka je ista bez obzira da li je softverski projekat otvoren ili zatvoren.

Druga tačka je međutim za nas kao druđtvo interesantna. Ona kazuje da se u našoj zemlji ne može razvijati hardware.

Ako neko hoće da se time bavi, neka ide u inostranstvo. Tamo se dešava visoka tehnologija. Ovdje ne, niti je to moguće.

Odakle nam pare za to ?!

1.5 Razvoj hardware-a u BiH !?

Prije nego ću dobiti temu za seminarski, ja bih "kao iz topa" rekao: "To se ovdje ne može raditi !"[9]

Odmah bi se prisjetio svog kolege Admira koji se zaposlio u Austriju u "Infinion"-u. On je radio u dijelu firme koja se bavi dizajnom sistema zaštite (alarmi sistemi protuprovala itd). Prvu stvar koju je pomenuo je bilo to da je za najjednostavniji prototip uređaja potrebno desetine miliona EUR-a.

I tu je odmah država kao mi eliminisana iz igre.

1.6 Otvoreni hardware ?

Međutim, kada sam se počeo pripremati za seminarski počeo sam čitati o CPLD[10], ASIC[11], FPGA kolima[12]. I tu zastadoh.

Ništa mi nije bilo jasno. Hardware koji je programibilan na nivou ožičavanja, strukture ? Hardware kod koga je logičko kolo gradivni element ?

Hardware koje možeš modelirati da bude mikroprocesor ili sound encoder ili ... ko zna šta ?!

Oooou.

Nisam očekivao da u drugoj polovini (ako ne i drugoj trećini :() svoje profensionalne karijere imam ovaj "ooooou" efekat.

Nakon otvorenog software mnoge super stvari bile su sjajne, ali ne i "ooou". Touch screen, smartphone, cloud computing. Sve je to super ali sve je to ipak bazirano na već viđenim stvarima i konceptima.

Završiću ovaj dugački uvod sa OpenCores[13], odnosno OpenRisc[14] projektima.

---

9 Za razliku od mog mišljenja po pitanju za razvoj software-a.

10 http://en.wikipedia.org/wiki/Complex_programmable_logic_device

11 http://en.wikipedia.org/wiki/Application-specific_integrated_circuit

12 http://en.wikipedia.org/wiki/Field-programmable_gate_array

13 http://en.wikipedia.org/wiki/OpenCores

14 http://en.wikipedia.org/wiki/OpenRISC

Ovi projekti na FPGA kolima rade kompletne IT sisteme. A svi ti projekti su opet otvoreni software.

1.6 Zašto sam napisao ovaj maratonski uvodni tekst ?

Ja sam, za razliku od većine mojih kolega, u srednjoj dobi. Kao informatičar, u poodmakloj dobi :(.

Za mene lično će biti sasvim dovoljno da o FPGA, VHDL-u naučim toliko da napišem ovaj seminarski rad za prolaznu ocjenu :). Moja karijera je najvjerovatnije određena onim što trenutno radim (poslovni software).

Želio sam međutim svoje ushićenje podijeliti sa kolegama koje su 20 godina mlađe od mene i logično razmišljaju o tome čime bi se mogli baviti.

Da sam 20 godina mlađi, meni bi ova oblast definitivno bila “na radaru” pri odabiru poslovne karijere.

1.7 Hardware biznis u BiH: utopija ili mogućnost

I ono što je još bitnije, treba uočiti da je čak i u Bosni moguće započeti hardware biznis. Jedina granica je znanje i trud, a ne neki ogromni kapital.

Starter kitovi koji sadrže sve da se počne eksperimentisati sa ovom tehnologijom koštaju 400-500 EUR-a. To i nije neka suma. Drage kolege, ako vas interesuje “hardware” biznis kupite neki FPGA starter kit[15] umjesto novog laptopa i “opletite” :)

Što se tiče pomenutog OpenCores projekta, interesantno je da sa su i software development toolchain maksimalno bazirali na otvorenom software-u.

Surfajući po internetu radi upoznavanja ove teme, naišao sam na niz firmi koje su specijalizirane za dizajn FPGA uređaja sa VHDL i Verilog programskim jezicima.

15 http://opencores.org/shop,items ; http://orsoc.se/fpgaboard,

2. Šta je VHDL ?

2.1 VHDL, Verilog

VHDL[16] je skraćenica od Very High Speed Integrated Circuits Hardware Description Language.

U istoj problemskoj domeni (hardware design) se koristi i Verilog[17].

Za razliku od VHDL-a, Verilog je nastao kao proprietary jezik firme “Gateway”. Nakon što je “Cadence” kupila “Gateway”, Verilog je otvoren kao public domain 1990. Kasnije verzije su izlazile pod okriljem IEEE-a[18], kao i VHDL.

Posljednja verzije verzije jezika su VHDL 1993 (IEEE standard 1076 1993), dok je posljednja verzija Verilog 2001 (IEE standard 1364 2001).

Ono što je najbitnije, danas su oba jezika otvoreni, te predstavljaju industrijski standard.

2.2 Usporedba VHDL-a i Verilog-a

Prema analizi “VHDL & Verilog Compared & Contrasted”[19] koju sam našao na internetu, oba jezika se mogu podjednako koristiti za modeliranje hardware-a.

Pri tome je VHDL jezik koji ima moderniju strukturu, što je i logično jer je kasnije koncipiran.

Konstrukti **visokog nivoa apstrakcije**: VHDL ima više takvih konstrukata nego Verilog:

- “package” izrazi koji mogućavaju korištenje istih modela na više projekata
- “configuration” izrazi koji omogućavaju posebno konfigurisanje dizajnerske strukture
- “generate” izrazi za ponavljanje iste strukture

S druge strane Verilog veoma zastupljen iz razloga što je duže prisutan u primjeni.

Meni je bilo interesantno da mnogi open hardware projekit koriste Verilog. Developeri (npr. ranije pomenuti OpenCores projekat) kažu da je podrška otvorenih alata za (opensource toolchain) znatno bolja za Verilog nego li za VHDL.

2.3. Glavni proizvođači programabilnog hardware-a (CPLD, ASIC, FPGA)

Firme **Xilinx**[20] and **Altera**[21] drže većinu svjetskog tržišnog udjela (> 80%) u oblasti proizvodnje ovog hardware-a.

Svaki od proizvođača ima svoj set softverskih alata za dizajn sopstveih hardverskih platformi:

- Altera: Quartus® II Software
- Xilinix: ISE Webpack[22] (sadrži ModelSim Xilinx Edition-III)

Oba proizvođača ravnopravno podržavaju programiranje Verilog i VHDL-u.

16 http://en.wikipedia.org/wiki/VHDL

17 http://en.wikipedia.org/wiki/Verilog

18 Skraćenica od **Institute of Electrical and Electronics Engineers**, izgovara se kao eng. “I-triple-E”

19 http://www.angelfire.com/in/rajesh52/verilogvhdl.html

20 http://www.xilinx.com/

21 http://www.altera.com/

22 http://www.xilinx.com/tools/webpack.htm

3. Opensource toolset za učenje VHDL-a

3.1 Zašto ?

Iako je u nastavnim materijalima navedeno da se za učenje koristi software “ModelSIM Student edition”, odlučio sam da potražim opensource alate.

Razlozi su sljedeći:

1. ja[23] sam veliki pobornik korištenja otvorenih rješenja, gdje god je to moguće. Smatram da bi posebno **obrazovne institucije** trebale forsirati primjenu otvorenog i slobodnog software-a
2. nemam i ne koristim windows operativni sistem, pa sam se pobojao da bi podešenje linux verzija ModelSIM-a moglo postalo “noćna mora”, što bi me skrenulo za glavnog zadatka.

3.2 GHDL

U daljnjim radnjama sa VHDL-om koristio sam ghdl[24] (*http://ghdl.free.fr/features.html*)

```
bringout@nvostro-hernad:~$ ghdl -v
GHDL 0.29 (20100109) [Sokcho edition]
 Compiled with GNAT Version: 4.4.3
 GCC back-end code generator
Written by Tristan Gingold.

Copyright (C) 2003 - 2010 Tristan Gingold.
GHDL is free software, covered by the GNU General Public License.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

Na stranicama ghdl-a piše:

> *“GHDL je VHDL compiler koji može (otprilike) izvršiti bilo koji VHDL program. GHDL nije syntesis alat. Sa GHDL-om ne možete napraviti netlistu.”*

Znači, za naše potrebe ghdl bi trebao biti sasvim dostatan.

3.3 GtkWave

http://gtkwave.sourceforge.net/

Na stranici gtkwave-a piše:

> *GTKWave je GTK+ bazirani preglednik signala za Unix i Win32 koji čita LXT, LXT2, VZT, FST, i GHW fajlove, kao i standarne Verilog VCD/EVCD fajlove.*

Mi ćemo koristiti funkciju pregleda vcd fajlova, koje ghdl može generisati, što ćemo kasnije vidjeti.

23 http://www.bring.out.ba

24 na ubuntu/debian se instalira sa apt-get install **freehdl**

3. VHDL Hello world

3.1 hello world.vhdl

http://gitorious.org/fit-mostar/aks/blobs/master/vhdl/hello/hello_world.vhdl

```
-- hello_world.vhdl  izlaz na ekran
--                   primjer bi trebao biti portabilan -
--                   trebao bi se moci pokrenuti kod svih VHDL vendor-implementacija
--                   koji zadovoljavaju IEEE Std. 1076-1993


entity hello_world is  -- test bench (top level program kao što je "main" u C-u)
end entity hello_world;

library STD;                            -- STD je inace automatic, ali neka ga
library IEEE;                           -- standardne IEEE biblioteke
use IEEE.std_logic_1164.all;            -- osnovni logicki tipovi
use STD.textio.all;                     -- osnovni I/O

-- I/O za logicke tipove - za ovo sam morao dati ghdl-u parametre --ieee=synopsys jer
-- std_logic_textio nije dio standardne IEEE implementacije nego je dodatak firme synopsys
-- http://www.mail-archive.com/ghdl-discuss@gna.org/msg00713.html
-- use IEEE.std_logic_textio.all;

architecture test of hello_world is
  -- deklaracija varijabli
  subtype word_32 is std_logic_vector(31 downto 0);
  -- 4 kao heksadecimalni broj
  -- signal four_32 : word_32 := x"00000004";
  signal counter : integer := 55;

begin
  -- parallel kod se ovdje izvrsava
  my_print : process is
             -- process je  parallel
             -- tip line dolazi iz textio
               variable my_line : line;
             begin


               -- zaglavlje

               write(my_line, string'("-------------------------------------------"));
               writeline(output, my_line);

               write(my_line, string'("Hello World from FreeHDL"));
               writeline(output, my_line);

               write(my_line, string'("-------------------------------------------"));
               writeline(output, my_line);



               -- linija 4
               -- write(my_line, string'("four_32 = "));

               -- format type std_logic_vector as hex
               -- hwrite(my_line, four_32)

               write(my_line, string'("counter= "));
               -- format counter kao integer varijablu
               write(my_line, counter);
```

```
                writeline(output, my_line);

                -- linija 5 - vrijeme
                write(my_line, string'("trenutno vrijeme "));
                write(my_line, now);
                writeline(output, my_line);

                wait for 10 ns;

                -- linija 6 - vrijeme nakon 10 ns
                write(my_line, string'("vrijeme nakon wait-a "));
                write(my_line, now);
                writeline(output, my_line);


                wait;
            end process my_print;
end architecture test;
```

3.2 bash skripta za kompilaciju i pokretanje primjera

http://gitorious.org/fit-mostar/aks/blobs/master/vhdl/hello/analyze_examine_run_hello_world.sh

```
#!/bin/bash

#OPTIONS="--ieee=synopsys --workdir=work"
OPTIONS=" --workdir=work"
mkdir work

CMD="ghdl -a $OPTIONS hello_world.vhdl"

echo 1. analize: $CMD
$CMD

CMD="ghdl -e $OPTIONS hello_world"

echo 2. examine: $CMD
$CMD

CMD="ghdl -r $OPTIONS hello_world"

echo 3. run: $CMD
$CMD
```

3.3 output

bringout@nvostro-hernad:~/fit/aks/vhdl/hello$ ./analyze_examine_run_hello_world.sh

```
1. analize: ghdl -a --workdir=work hello_world.vhdl
2. examine: ghdl -e --workdir=work hello_world
3. run: ghdl -r --workdir=work hello_world
-----------------------------------------
Hello World from FreeHDL
-----------------------------------------
counter= 55
trenutno vrijeme 0 ns
vrijeme nakon wait-a 10 ns
```

3.4 Komentari

Originalni source kod sam pronašao, tražeći resource za rad sa VHDL/ghdl. Dosta sam vremena izgubio da shvatim kako da pozovem library “std_logic_textio.all” (unutar koje se nalazi funkcija za ispis heksadecimalnog broja hwrite).

Na kraju sam to uspio (ghdl opcija --ieee=synopsys) ali sam utvrdio da se radi o nestandardnoj biblioteci pa sam kao pobornik standarda :) taj dio koda isključio u formi komentara.

Sam kod demonstrira osnovnu strukturu VHDL programa, ali ne radi ništa korisno.

Uočimo sljedeće segmente našeg prvog VHDL programa:

- entity hello_world koji u sebi ništa ne sadrži
- use library-ja (STD, IEEE)
    - architecture test
        - deklaracija korisničkih tipova, varijabli. signala: counter
        - architecture izvršenje (begin blok)
        - my_print proces
            - deklaracija lokalne varijable my_line
            - proces izvršenje (begin blok)
                - komande unutar procesa

U nastavnim materijalima piše:

Entity se koristi za deklaraciju I/O portova kola. Naš hello world nema nikakve inpute niti outpute, zato je prazan.

Architecture je dio u kome se dešavaju procesi - što u stvari reprezentuje funkcionisanje sklopa.

Naš hello_world, trebamo uočiti, ne može se sintetizirati - na osnovu njega nema smisla praviti nekakvu fizičku implementacija na konkretan programabilni hardware (FPGA, CPLD ...).

4. VHDL hello_2 - or_gate

Naš sljedeći primjer će sadržati nešto što zna uraditi nekakav posao. Uzećemo najjednostavniji mogući primjer: rad jednog or kola.

4.1 definicija or kola: or_gate - or_gate.vhdl

http://gitorious.org/fit-mostar/aks/blobs/master/vhdl/or_gate/or_gate.vhdl

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

entity or_gate is
port(    x: in bit;
         y: in bit;
         F: out bit
);
end or_gate;

architecture rtl of or_gate is
begin

    F <= x or y;

end rtl;
```

4.2 test or kola

http://gitorious.org/fit-mostar/aks/blobs/master/vhdl/or_gate/or_test.vhdl

```
entity or_test is
end or_test;

architecture behav of or_test is
    component or_gate
        port (x, y : in bit; F : out bit);
    end component;

    for or_gate_0: or_gate use entity work.or_gate;
    signal x, y, F: bit;

begin

    or_gate_0: or_gate port map (x => x, y => y, F => F);

    process

    type pattern_type is record
        x, y : bit;
        F: bit;
    end record;

    type pattern_array is array (natural range <>) of pattern_type;

    constant patterns: pattern_array :=
      (('0', '0', '0'),
       ('0', '1', '1'),
       ('1', '0', '1'),
       ('1', '0', '0'),
```

```
      ('1', '1', '1'));

   begin
       for i in patterns'range loop
           x <= patterns(i).x;
           y <= patterns(i).y;

           wait for 1 ns;

           assert F = patterns(i).F
              report "bad F value" severity error;

       end loop;

   assert false report "end of test" severity note;
   wait;

  end process;

end behav;
```

4.3 compile, run - kreiraj vcd fajl za gtkwave

http://gitorious.org/fit-mostar/aks/blobs/master/vhdl/or_gate/analyze_examine_run_or.sh

```
#!/bin/bash
OPTIONS=" --workdir=work"

mkdir work
CMD="ghdl -a $OPTIONS or_gate.vhdl"

echo 1. analize: $CMD
$CMD

CMD="ghdl -a $OPTIONS or_test.vhdl"

echo 1b. analize: $CMD
$CMD

CMD="ghdl -e $OPTIONS or_test"

echo 2. examine: $CMD
$CMD
CMD="ghdl -r $OPTIONS or_test --vcd=or_test.vcd"

echo 3. run - vcd output: $CMD

$CMD
```

4.4 pokrenimo naš test

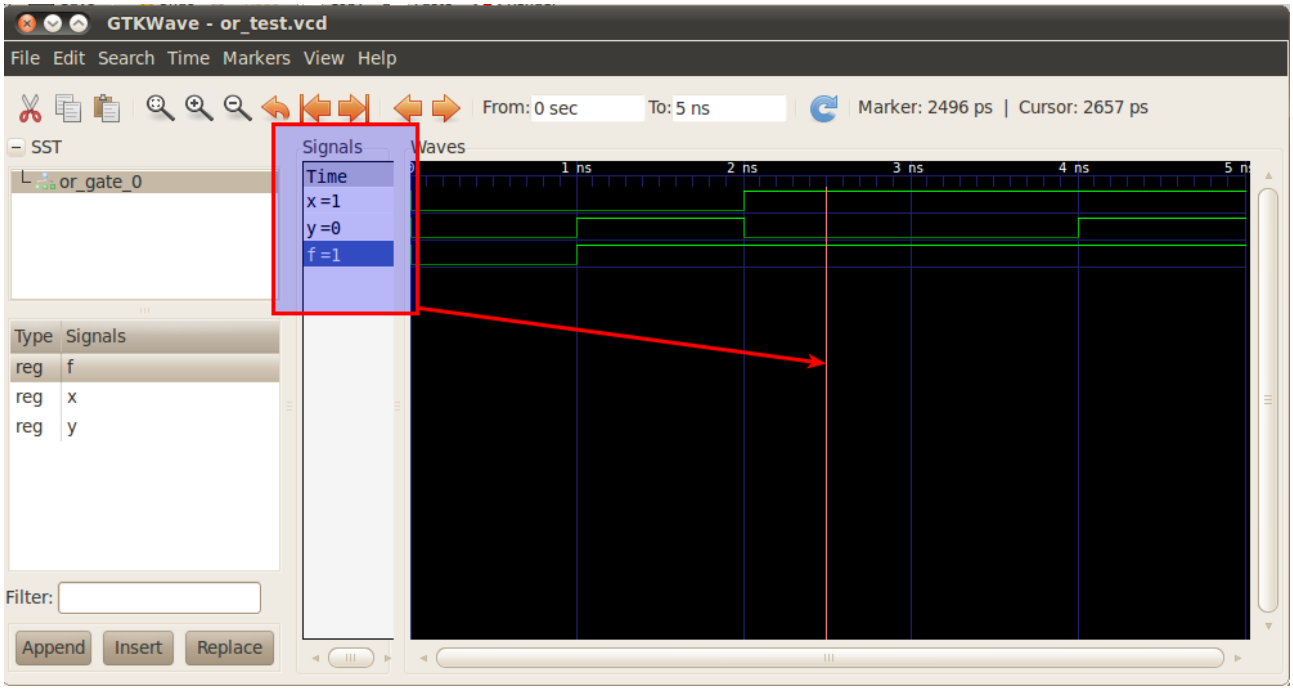
```
bringout@nvostro-hernad:~/fit/aks/vhdl/or_gate$ ./analyze_examine_run_or.sh
mkdir: ne mogu napraviti direktorijum „work“: File exists
1. analize: ghdl -a --workdir=work or_gate.vhdl
1b. analize: ghdl -a --workdir=work or_test.vhdl
2. examine: ghdl -e --workdir=work or_test
3. run - vcd output: ghdl -r --workdir=work or_test --vcd=or_test.vcd
or_test.vhdl:39:13:@4ns:(assertion error): bad F value
or_test.vhdl:44:5:@5ns:(assertion note): end of test
```

4.5 Value change dump file: pregled sa gtkwave

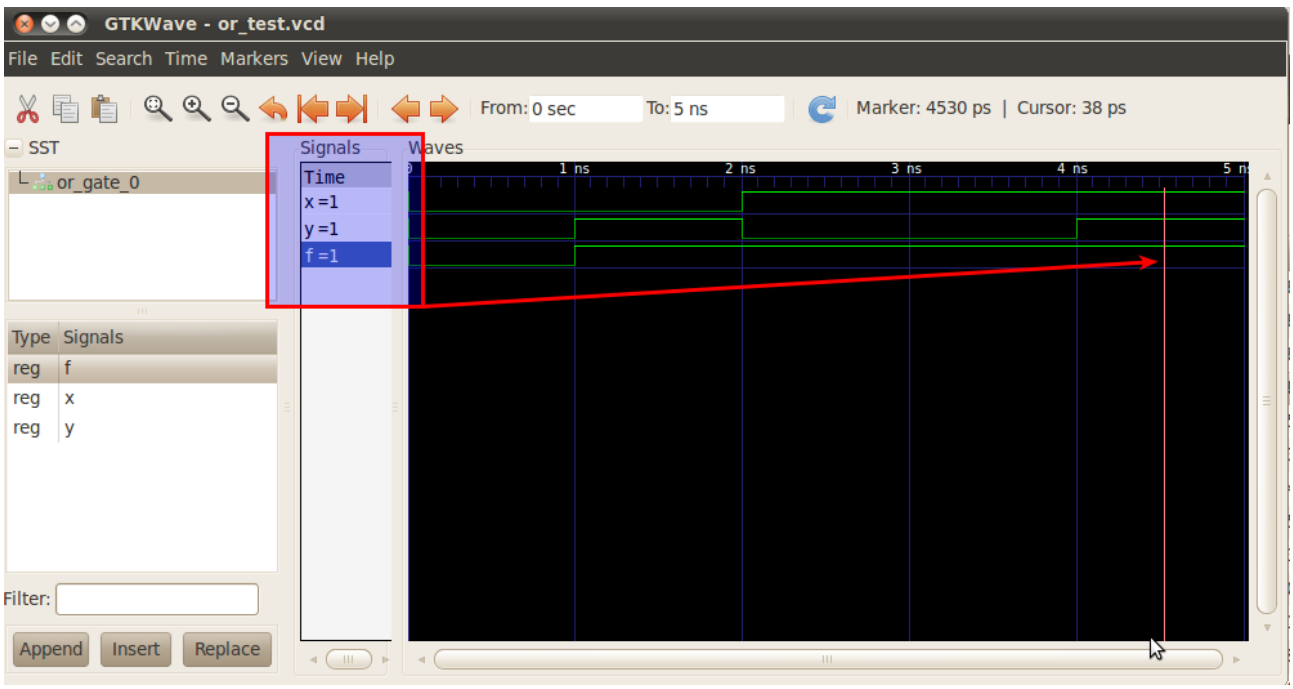
rezultat pokretanja “ghdl -r --workdir=work or_test --vcd=or_test.vcd ” je

vcd fajl koji nam omogućava pregled rada sklopa - pregled signala u vremenskoj osi:

Stanje signala nakon 2 ns (treći prolaz kroz for petlju):



Stanje signala nakon 4 ns (peti prolaz kroz for petlju):



4.6 Komentari

Or gate je prvi konkretan sklop koji smo modelirali u vhdl-u.

Puno zahtjevnije dio posla je bilo testiranje samog sklop-a - formiranje test bench-a.

Test bench je napravljen tako što je formirana matrica sa 5 uzoraka ulaza (prva dva elementa matrice) i željenih izlaza (posljednji element).

Samo testiranje je obavljeno tako što su svakih 1 ns na ulaze dovođeni signali iz testne matrice.

Unutar for petlje je stavljen "**assert** F signal jednak F očekivano ? ... **report** "xyz" **severity error**" iskaz koji je u slučaju da stanje izlaza ne bude kao predviđeno kod izvršenja programa prikazivao grešku.

Iako smo u primjeru dotakli mali subset VHDL-a, već sa ovim "arsenalom" izraza (statement-a) možemo konstruisati praviti i složenije sklopove.

Što se tiče same sintakse, moram reći da sam većinu stvari prekucavao iz postojećih primjera (vježbe, internet resursi). **Fokus** sam stavio na upoznavanje sa principima jezika i korištenim alatima, a ne na ovladavanje programskom sintaksom.

Da budem iskren, dok nisam shvatio kako da napravim analizu i vidim vizuelnu reprezentaciju rada sklopa sa gtkwave-om, malo šta mi je bilo jasno :(.

4.7 Video prezentacija: VHDL + ghdl + gtkwave

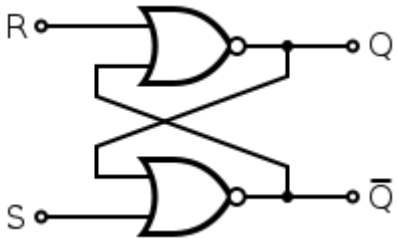
Video prezentacija demonstrira or_gate sa posebnim osvrtom na tim dijagram sklopa koji smo dobili sa gtkwave-om:
http://www.youtube.com/watch?v=vLQqhN8cBo0

5. Primjer 3 - S-R flip-flop

Napravimo sada nešto još “pamentije” : napravićemo S-R flip-flop.

5.1 shema

http://en.wikipedia.org/wiki/Flip-flop_%28electronics%29:



Za ovaj sklop nam trebaju dva nor kola, pa ćemo napraviti prvo nor_gate.vhdl, a onda flip_flop.vhdl.

5.2 internet resursi

http://www.edaboard.com/ftopic200442.html (iako netačan kod, ovaj mi je primjer najviše pomogao da napravim sklop)

http://www.edaboard.com/ftopic298408.html

http://esd.cs.ucr.edu/labs/tutorial/#flip-flops

5.3 nor_gate.vhdl

http://gitorious.org/fit-mostar/aks/blobs/master/vhdl/flip_flop/nor_gate.vhdl

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


entity nor_gate is
port(    x: in std_logic;
         y: in std_logic;
         f: out std_logic
);
end nor_gate;

architecture rtl of nor_gate is
begin

    f <= not (x or y);

end rtl;
```

5.4 flip_flop.vhdl

http://gitorious.org/fit-mostar/aks/blobs/master/vhdl/flip_flop/flip_flop.vhdl

```
entity flip_flop is
end flip_flop;
```

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

library std;
use std.textio.all;

-- ovo trebam radi stampe std_logic varijabli
use ieee.std_logic_textio.all;

architecture behav of flip_flop is
    -- koristimo nor_gate komponentu u nasem sklopu
    component nor_gate
        port (x, y : in std_logic; f : out std_logic);
    end component;

    for nor_gate_0: nor_gate use entity work.nor_gate;
    for nor_gate_1: nor_gate use entity work.nor_gate;
    signal r, s, y0, x1, q, q_invert: std_logic;

begin


    nor_gate_0: nor_gate port map (x => r, y => y0, f => q);
    nor_gate_1: nor_gate port map (x => x1, y => s, f => q_invert);

    -- spoji q i x1
    x1 <= q;

    -- spoji signale q_invert i x0
    y0 <= q_invert;



    process

    variable my_line: line;

    type pattern_type is record
        s, r : std_logic;
        q: std_logic;
    end record;

    type pattern_array is array (natural range <>) of pattern_type;

    constant patterns: pattern_array :=
      (('0', '0', '0'),
       ('0', '1', '0'),
       ('0', '0', '0'),
       ('1', '0', '1'),
       ('0', '0', '1'),
       ('0', '1', '0'));

    begin

        for i in patterns'range loop

            s <= patterns(i).s;
            r <= patterns(i).r;
            wait for 1 ns;

            write(my_line, string'("s="));
            write(my_line, s);

            write(my_line, string'(" r="));
```

```
            write(my_line, r);


            write(my_line, string'(" q="));
            write(my_line, q);

            writeline(output, my_line);



            assert q = patterns(i).q
               report "bad q value" severity error;

            assert q_invert = not patterns(i).q
               report "bad q_invert value" severity error;


        end loop;

    assert false report "end of flip_flop" severity note;
    wait;

  end process;

end behav;
```

5.5 bash skripta

http://gitorious.org/fit-mostar/aks/blobs/master/vhdl/flip_flop/analyze_examine_run.sh

```
#!/bin/bash

OPTIONS="  --ieee=synopsys --workdir=work"

mkdir work

CMD="ghdl -a $OPTIONS nor_gate.vhdl"

echo 1. analize: $CMD
$CMD

CMD="ghdl -a $OPTIONS flip_flop.vhdl"

echo 1b. analize: $CMD
$CMD

CMD="ghdl -e $OPTIONS flip_flop"

echo 2. examine: $CMD

$CMD

CMD="ghdl -r $OPTIONS flip_flop --vcd=flip_flop.vcd"

echo 3. run - vcd output: $CMD

$CMD
```

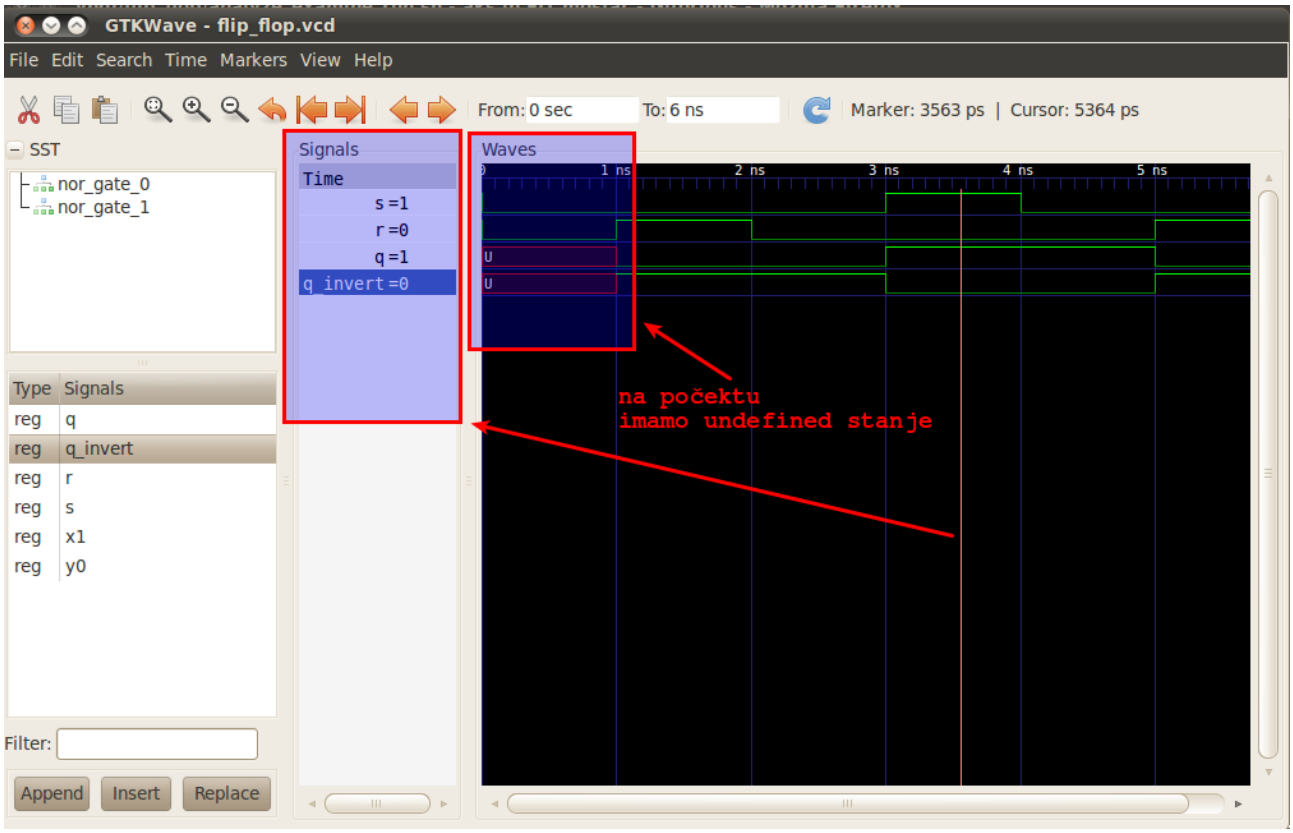
5.6 console output

```
bringout@nvostro-hernad:~/fit/aks/vhdl/flip_flop$ ./analyze_examine_run.sh
1. analize: ghdl -a --ieee=synopsys --workdir=work nor_gate.vhdl
1b. analize: ghdl -a --ieee=synopsys --workdir=work flip_flop.vhdl
```

```
2. examine: ghdl -e --ieee=synopsys --workdir=work flip_flop
3. run - vcd output: ghdl -r --ieee=synopsys --workdir=work flip_flop --vcd=flip_flop.vcd
s=0 r=0 q=U
flip_flop.vhdl:78:13:@1ns:(assertion error): bad q value
flip_flop.vhdl:81:13:@1ns:(assertion error): bad q_invert value
s=0 r=1 q=0
s=0 r=0 q=0
s=1 r=0 q=1
s=0 r=0 q=1
s=0 r=1 q=0
flip_flop.vhdl:87:5:@6ns:(assertion note): end of flip_flop
```

5.7 gtkwave sklopa

analiza pokazuje da se sklop dobro ponaša: kada setujemo “s” ulaz, “q” izlaz se setuje na “1”, “q_invert” ide na “0”:



5.7 Komentari

Za razliku od ranijih primjera koji su bili rađeni uglavnom tako što sam našao source kod koji sam analizirao i minorno prepravljao, ovaj sklop sam pokušao sam napraviti.

Sahat-dva sam pokušavao samostalno, ali nisam dobio ispravan rezultat :(.

Na kraju sam posegnuo za sličnim primjerima, i ispravio svoj VHDL kod.

Ono što je bitno, radi se ipak o mom kodu :)

6. Zaključci

Kako se iz uvodnog teksta da primjetiti, programabilni hardware (FPGA, CPDL itd) me je jednostavno oduševio.

U toku izrade seminarskog sam mnogo vremena utrošio na čitanje o (opensource) projektima koji su se bazirali na ovoj tehnologiji.

Jedan od prvih projekata na koje sam naletio pretražujući internet je bio projekat ***java on procesor***[25].

Tu su sam počeo shvatati o čemu se ovdje uopšte radi.

Kasnije sam pregledao hardware kitove koji se nude za razvoj sopstvenih projekata. To tržište u posljednjih 5-6 godina ima veliki razvoj.

Ubjeđen sam da će FPGA ili neki nasljednjih tehnologije u budućnosti bitno pomjeriti granice korištenja ove tehnologije. Međutim i sadašnje korištenje (koje je sada primarno orjentisano na izradu **hardware prototipova** ili specifičnih sklopova za koje se ne isplati veća serijska proizvodnja) ima ogroman potencijal i rast.

Pregledajući OpenCores projekte naletio sam recimo na implementaciju USB 3.0 protokola. FPGA je praktično poligon za uvođenje i testiranje novih tehnologija.

Što se VHDL-a tiče, moram reći da je tu ostalo puno neobrađenog i nejasnog. Sintetizaciju i različite nivoe apstrakcije sam više puta iščitao, ali tu imam dosta nejasnoća.

Siguran sam da bi eksperimentisanje sa konkretnim hardware-om te stvari postpuno razjasnila.

Što se tiče svih konstrukata jezika, mnogo toga sam samo pročitao u nastavnim materijalima. Išao sam principom da je bolje shvatiti napraviti par “hello world” primjera **čestito**, nego da naštrebam sve konstrukte k'o lektiru.

Na kraju krajeva, ovo je informatika. Kada nam nešto zatreba, tu je računar.

25 http://www.jopdesign.com/

7. Pojmovi

Navešći ostale pojmove koje sam tokom obrade teme upoznao.

7.1 IP cores

IP core (intellectual property core) je reusable (iskoristiv na više mjesta) logički blok, ćelija ili chip layout (organizacija čip-a) koji je vlasništvo jedne strane.

Ti IP core-ovi mogu biti licencirani na različite načine. Proizvođači programabilnog hardware-a često te core-ove daju kao dodatke na ponudu svog hardware-a.

Međutim, postoji i niz neovisnih proizvođača ovih core-ova, kao i niz IP core-ova čija je licenca otvoreni/slobodni software.

U principu IP core je za dizajn čipa ono što je u standardnom programiranju softverska biblioteka ili software framework.

Postoje soft IP cores i hard IP cores. Soft IP cores su isporučeni u jezicima za opisa hardware-a Verilog ili VHDL.

Hard IP cores su obično low level apstrakcija – nivo layout-a chip-ova.

Analogijom se može reći da su soft IP cores artifakti pisani u high level jezicima, a hardcores u asembler nivo jezicima za opis hardware-a.

7.2 VHDL, sinteza

- **Synthesis**[26] is a general term that describes the **process of transformation** of the model of a design, usually described in a hardware description language (HDL), from one level of behavioral abstraction to a lower, more detailed behavioral level.
- Nivoi sinteze:
  - **Behavioral** Synthesis - synthesis of abstract behavior or control-flow behavior from a high level algorithm description Levels of Synthesis
  - **RTL Synthesis** - synthesis of register-transfer structure from abstract, control-flow, or register-transfer behavior
  - **Logic Synthesis** - synthesis of gate-level logic (an implementation) from register-transfer structure or Boolean equations
- **RTL** - Register-Transfer Level
- **Technology consideration**
  - The developer of a synthesizable behavioral VHDL description MUST consider the implementation technology whenwriting the code
  - Some FPGA families do not have internal tri-state devices - behavioral descriptions that are based on internal tri-state busses will fail in the technology mapping phase ; Many PLDs have flip flops only on the I/O pins - state machines with too many I/O ports and state variables simply wont fit

26 http://www.scribd.com/doc/7216566/Synthesizable-Vhdl-Codes

8. Radno okruženje

8.1 Radna stanica OS: Ubuntu desktop 10.04 amd64

8.2 Tekst procesor korišten za pripremu seminarskog:

1. openoffice.org 3.2
2. openoffice.org-coooder plugin za syntax highlight

8.3 Rad sa VHDL-om

1. freehdl 0.0.7-1 - VHDL simulator for Linux
2. gtkwave 3.3.7
3. vi editor 7.2, gnome gedit 2.30

8.4 Video priprema:

1. kdenlive 0.7.7.1
2. ffmpeg SVN-r23019-4:0.6~svn20100505-1ubuntu2

9. artifakti na gitorious.org

Na sljedećoj lokaciji nalaze se svi artifakti ovog: http://gitorious.org/fit-mostar/aks/trees/master/seminarski