

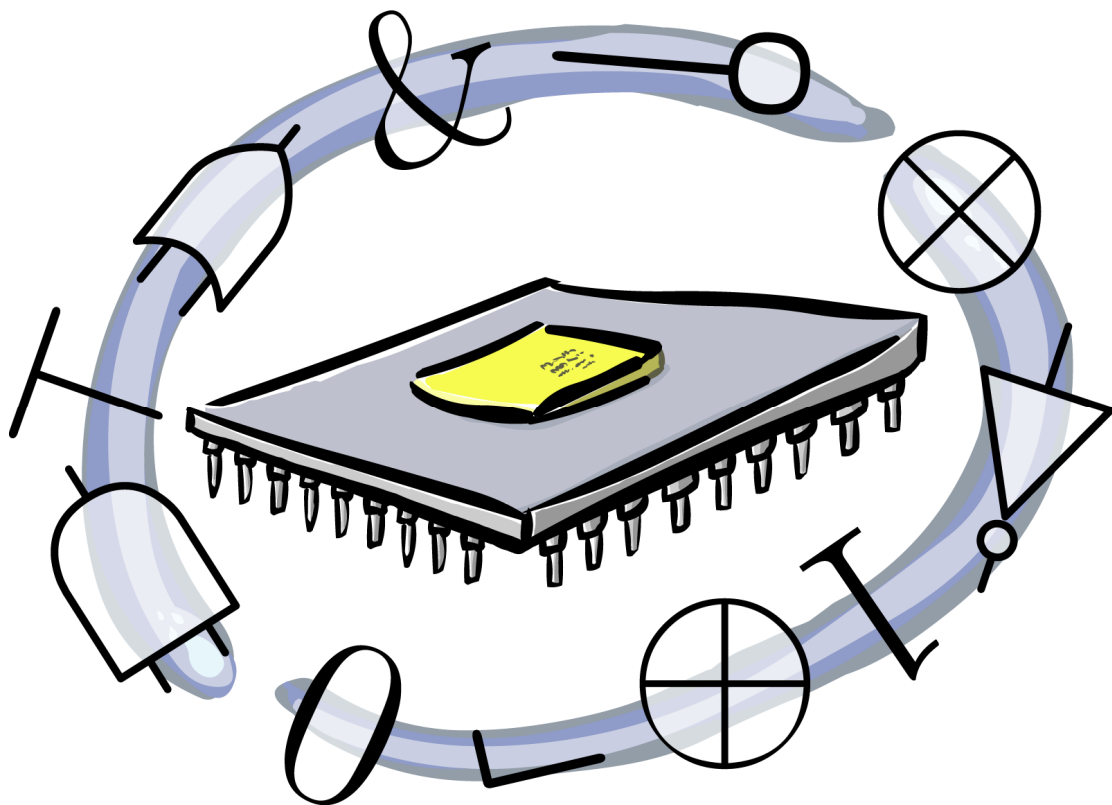


Univerzitet u Nišu
Elektronski fakultet

Seminarski rad

Alati za projektovanje

- upotreba alata za projektovanje na konkretnim primerima -



Goran Mišić, 12103

Niš, 24. 03. 2008.

1 Uvod

Savremena elektronska kola su dostigla stepen kompleksnosti koji prevazilazi mogućnosti njihovog efikasnog projektovanja bez upotrebe adekvatnog softvera i računara. Šta više, neka kola su proizvedena tako da se upotreba softvera za projektovanje podrazumeva pri njihovom "programiranju" odnosno formiranju veza u njima.

Programi koji se koriste za projektovanje nazivaju se "alatima". Često se ne radi o jednom programu već o skupu programa - programskom paketu. Takođe, pri projektovanju složenog elektronskog sistema, može se zahtevati i upotreba više različitih alata za isti projekat. Ovo ukazuje na značaj kompatibilnosti alata za projektovanje.

U ovom radu je na praktičnim primerima prikazana upotreba nekih od alata za projektovanje. Detaljno je opisan svaki postupak koji je preduzet u toku procesa projektovanja.

U poglavlju 2 je uradjena implementacija 32-bitnog generatora slučajnih brojeva u FPGA kolo. Najpre je opisan algoritam koji je korišćen za generisanje slučajnih brojeva. Algoritam je zatim preveden u VHDL opis kola koje obavlja funkciju generisanja slučajnih brojeva (RNG) . Zatim je pomoću alata Active-HDL uradjena simulacija rada kola. Na kraju je, pomoću alata Xilinx ISE, uradjena implementacija VHDL opisa kola u FPGA čip.

U poglavlju 3 je ukratko opisan postupak generisanja lejauta ASIC kola iz VHDL opisa. Korišćeni alat je ModelSim, Leonardo Spectrum, DesignArchitect i ICStation.

U poglavlju 4 je uradjeno projektovanje lejauta ASIC kola stereo koder. Najpre je dat funkcionalni opis stereo koder. Zatim je kolo koder opisano na nivou šeme i uradjena je simulacija rada kola. Korišćen je softverski paket PSpice. Zatim je dat skup MOSIS pravila koja su korišćena pri projektovanju lejauta. Na kraju je pomoću alata LAsI7 izvršeno projektovanje lejauta čipa stereo koder. Simulacija podkola čipa je radjena pomoću alata PSpice AD.

2 Implementacija 32-bitnog MWC RNG u FPGA

Algoritam MWC RNG (Multiply-With-Carry Random Number Generator) za generisanje 32-bitnih pseudoslučajnih brojeva je veoma jednostavan. Realizacija MWC generatora u digitalnom kolu zahteva veliki broj gejtova. Zato je u ovom slučaju odabrano FPGA kolo iz familije Spartan3E firme Xilinx. Implementacija je izvedena do trenutka kada je generisan fajl kojim se može programirati FPGA.

2.1 MWC algoritam

MWC algoritam se zasniva na sledećoj rekurziji [1]:

$$x_n = (a \cdot x_{n-1} + c_{n-1}) \bmod b$$
$$c_n = \text{int} \left\{ \frac{a \cdot x_{n-1} + c_{n-1}}{b} \right\}$$

odnosno:

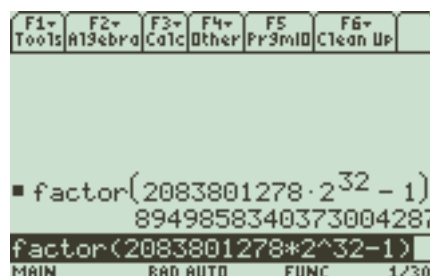
- novi x = ostatak od (a·x+c)/b
- novo c = celobrojni deo od (a·x+c)/b .

Za bilo koju inicijalnu vrednost x iz opsega $0 < x < (b-1)$ i inicijalnu vrednost za c iz opsega $0 < c < (a-1)$ rekurzija generiše niz x-ova: x_0, x_1, x_2, \dots koji je striktno periodičan. Dužina periode, a nama je potrebno da bude što veća, zavisi od izabranih vrednosti a i b. Ako je $m = a \cdot b - 1$ prost broj, tada perioda ima dužinu m.

Vrednost za b se može izabrati tako da bude stepen broja 2 što može značajno olakšati primenu MWC algoritma u nekom binarnom okruženju. Neka je $b = 2^{32}$. Vrednosti za a, takve da je $m = a \cdot b - 1$ prost broj, možemo uzeti iz sledeće tabele [2]:

a				
1075433238	1447497129	1517746329	1554115554	1557985959
1606218150	1631296680	1655692410	1675393560	1683268614
1781943330	1791398085	1873196400	1893513180	1929682203
1965537969	1967773755	2051013963	2083801278	2131995753

Neka je $a = 2083801278$. Tada je dužina periode $m = a \cdot b - 1 = 2083801278 \cdot 2^{32} - 1 = 8949858340373004287$!!! Uzmemo digitron (malo bolji digitron - TI-89) i proverimo da li je m prost broj:



Kao što se vidi, faktorizacijom broja m dobijamo opet m, što znači da je m prost broj.

[1] <http://www.stat.fsu.edu/pub/diehard/>

[2] <http://www.rkrupinski.ps.pl/langEn/random.php>

Neka je w binarna reč dužine 64 bita takva da je prva polovina (bitovi 63 do 31) načinjena od bitova binarne reprezentacije broja c , a druga polovina (bitovi 31 do 0) načinjena od binarne reprezentacije broja x . Sada možemo umesto gornje rekurzije koristiti jednostavniju:

$$w_n = a * (w_{n-1} \& (2^{32} - 1)) + (w_{n-1} \gg 32)$$

pri čemu se u svakoj novoj iteraciji dobija novi 32-bitni pseudo-slučajni broj:

$$w_{n-1} \& (2^{32} - 1)$$

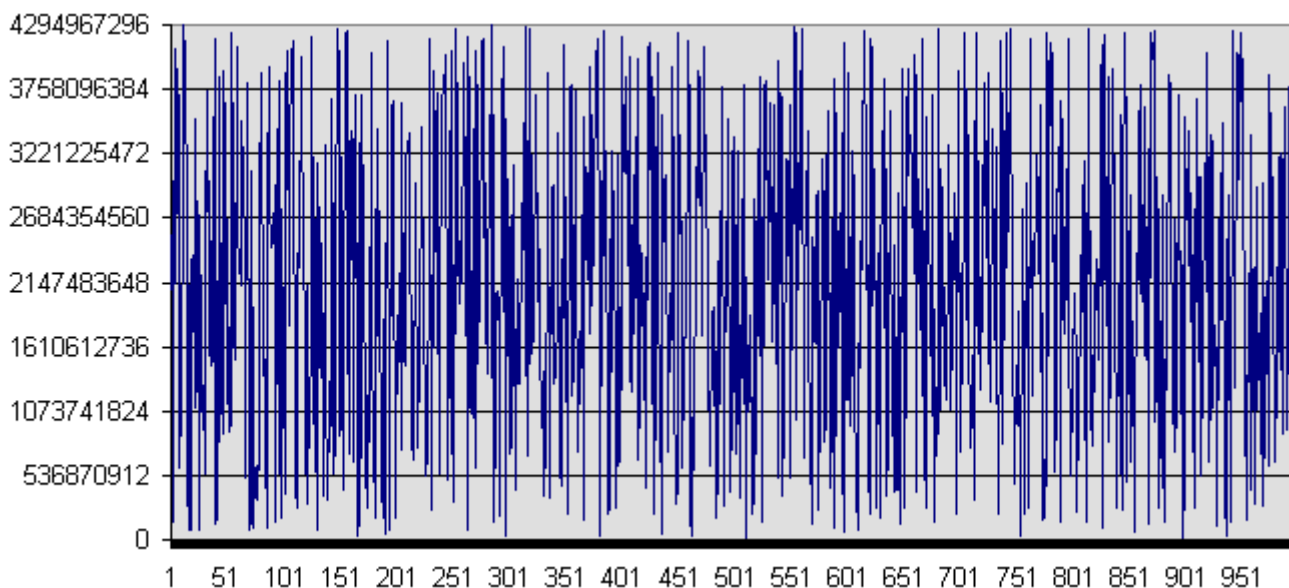
Program napisan u C-u koji generiše prvih 1000 pseudo-slučajnih brojeva koristeći ovakav oblik rekurzije MWC algoritma je:

```
#include <stdio.h>
void main()
{
    int i;
    long long w=1;
    for(i=0;i<1000;i++)
    {
        w=2083801278*(w&4294967295)+(w>>32); //a=2083801278
        printf("%lld\n",w&4294967295);
    }
}
```

Može se koristiti Microsoft Visual C++. Generisani niz brojeva je:

2083801278
2983947524
144095773
4100253040
2723449940
3923414890
...
2674864222

Ako taj niz kopiramo u Microsoft Office Excel i nacrtamo grafik, izgledaće veoma "slučajno":



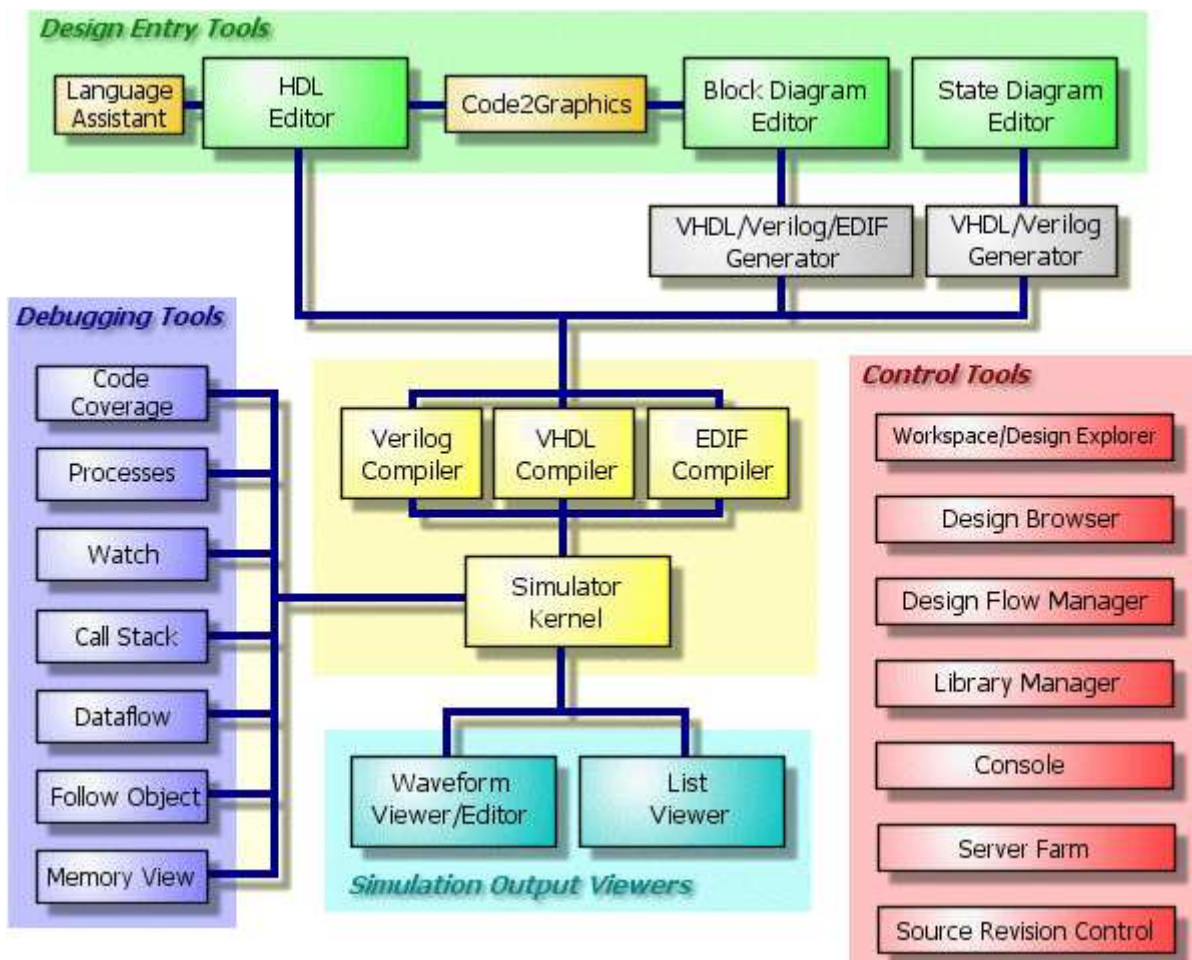
2.2 VHDL opis MWC RNG

Implementacija MWC RNG u FPGA kolo podrazumeva njegov opis u jeziku za opis hardvera (Hardware Description Language - HDL). Koristićemo jedan od mogućih jezika - VHDL (Very high speed Hardware Description Language). Korišćeni alat je Active-HDL firme Aldec.

2.2.1 Kratak opis alata Active-HDL

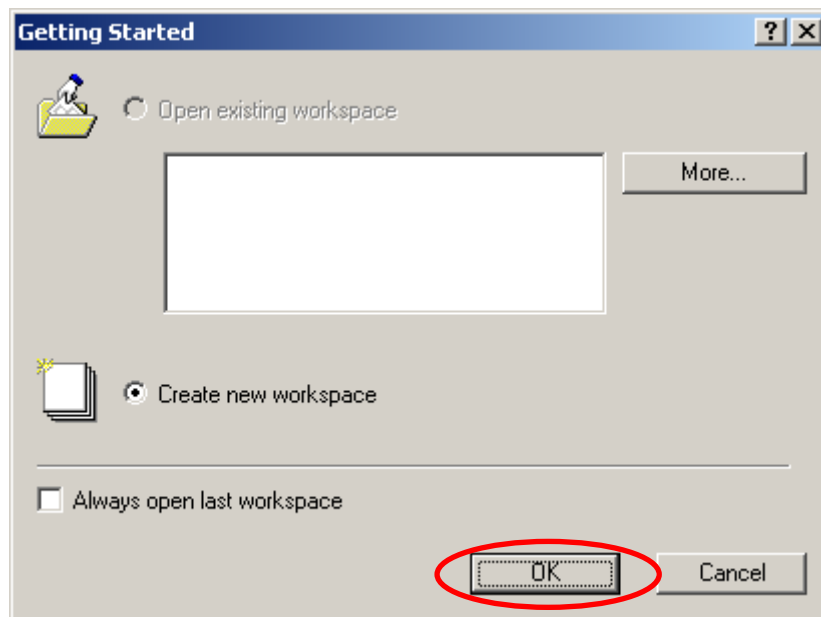
Active-HDL je integrisano okruženje za rad sa VHDL, Verilog, EDIF i kombinovanim VHDL-Verilog-EDIF projektima. Čini ga nekoliko alata za:

- unos projekta
- VHDL i Verilog kompajliranje
- simulaciju
- debugiranje
- grafički i tekstualni pregled rezultata simulacije
- pomoć u menadžmentu fajlova i biblioteka projekta:
 - Block Diagram Editor
 - Code2Graphics converter
 - Console
 - Design Browser
 - Design Flow Manager
 - HDL Editor
 - Language Assistant
 - Library Manager
 - Memory View
 - State Diagram Editor
 - Waveform Viewer/Editor
 - Workspace/Design Explorer
 - ...

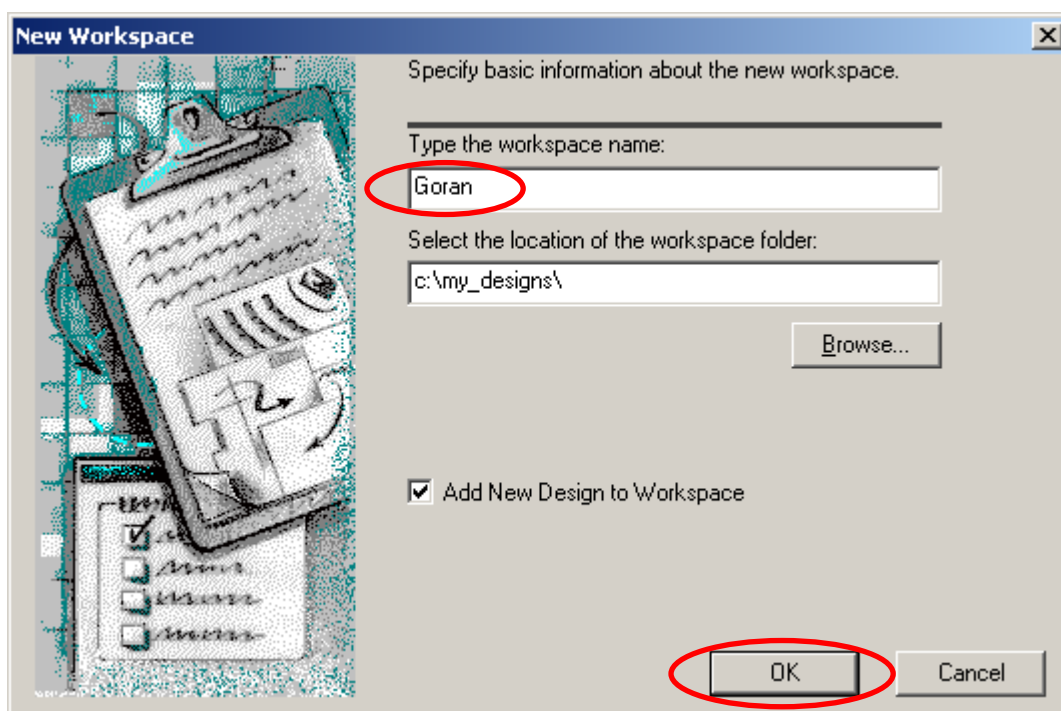


2.2.2 Priprema radnog prostora

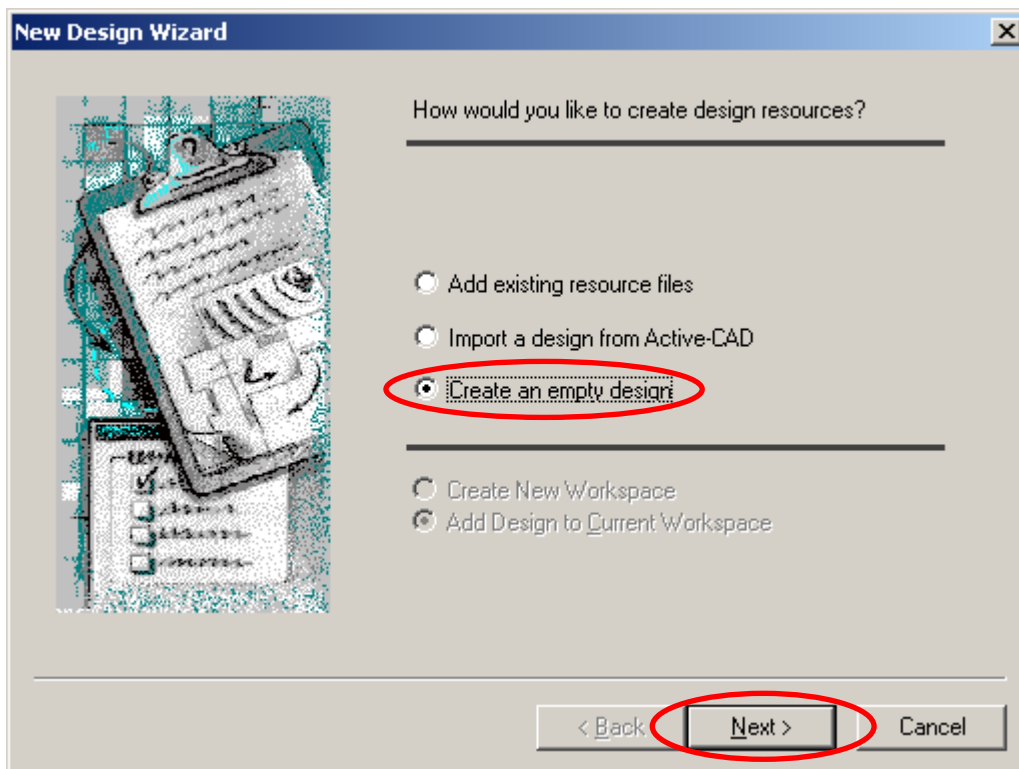
Nakon pokretanja programa Active-HDL otvara se sledeći prozor:



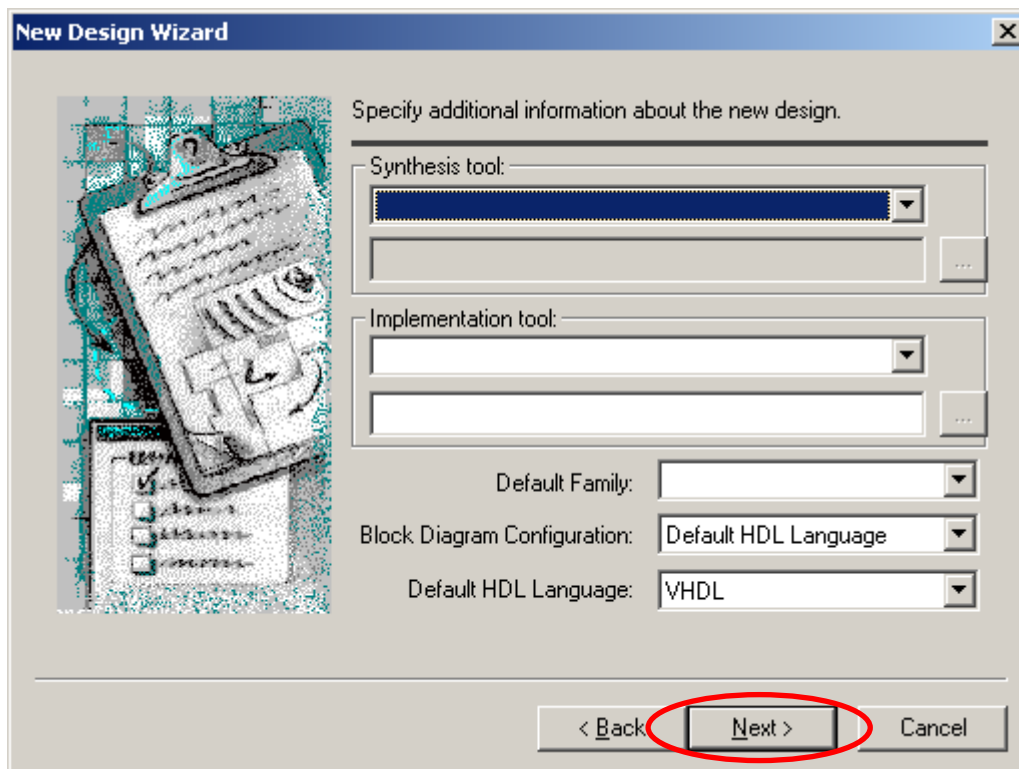
Kliknemo na OK. Otvoriće se novi prozor u kome definišemo ime radnog prostora (ovde je to Goran) i kliknemo na OK:



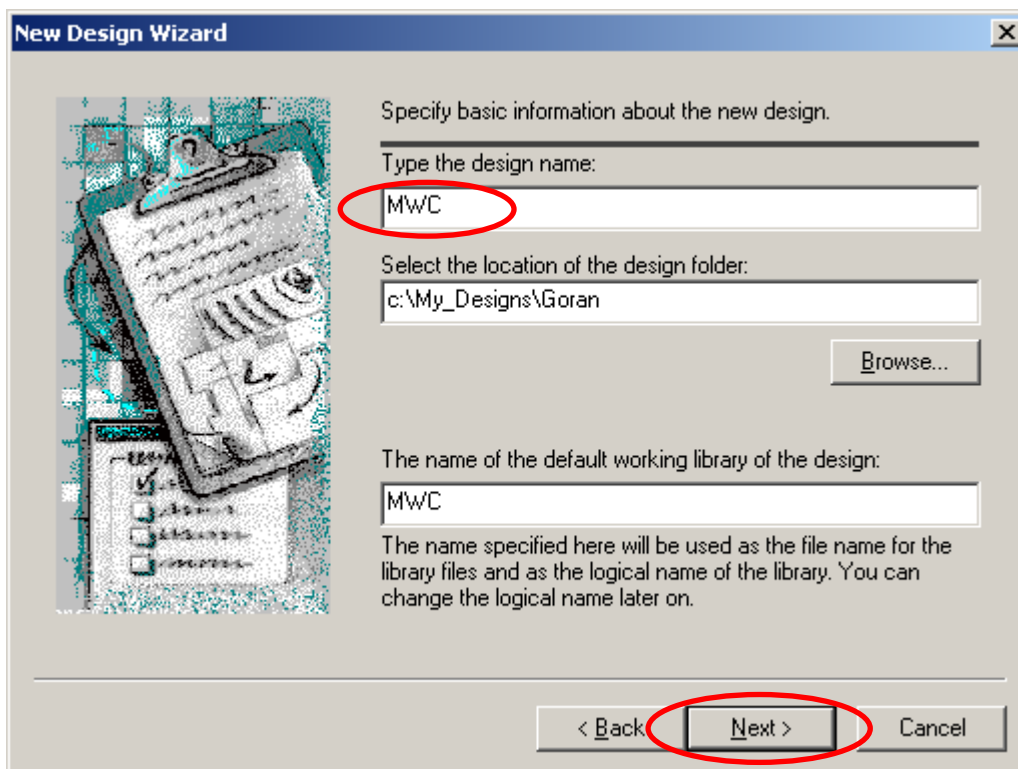
Označimo Create an empty Design i kliknemo na Next >:



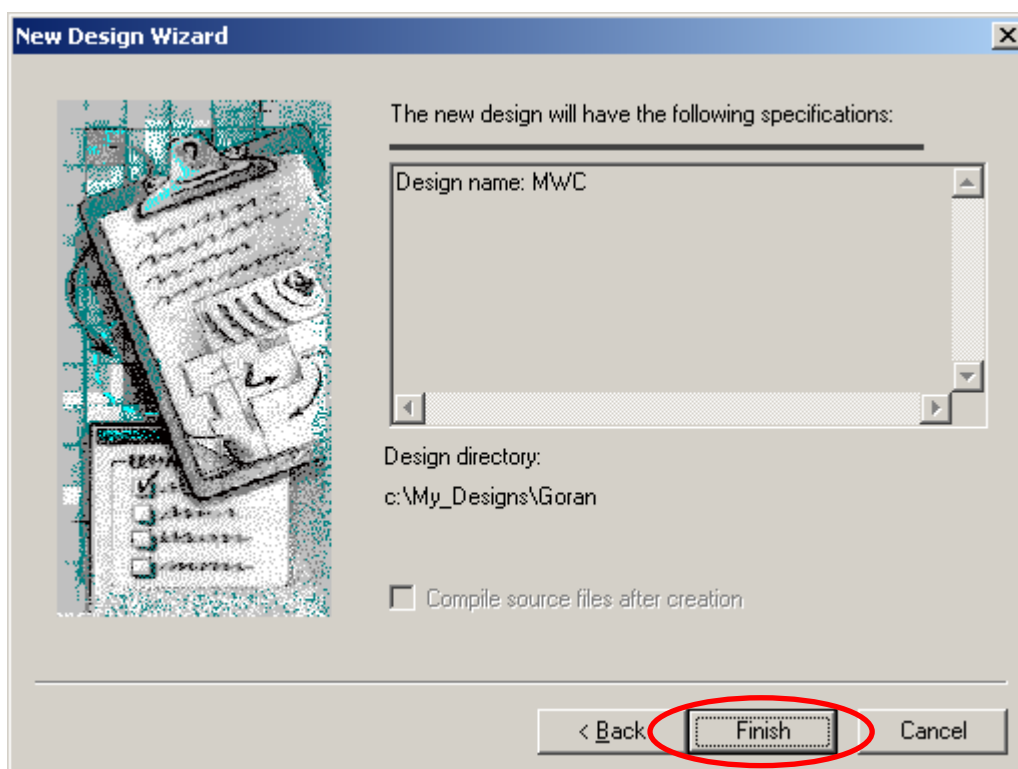
U novootvorenom prozoru samo kliknemo na Next >:



Sada ukucamo ime projekta (u ovom slučaju je to MWC) i kliknemo na Next >:



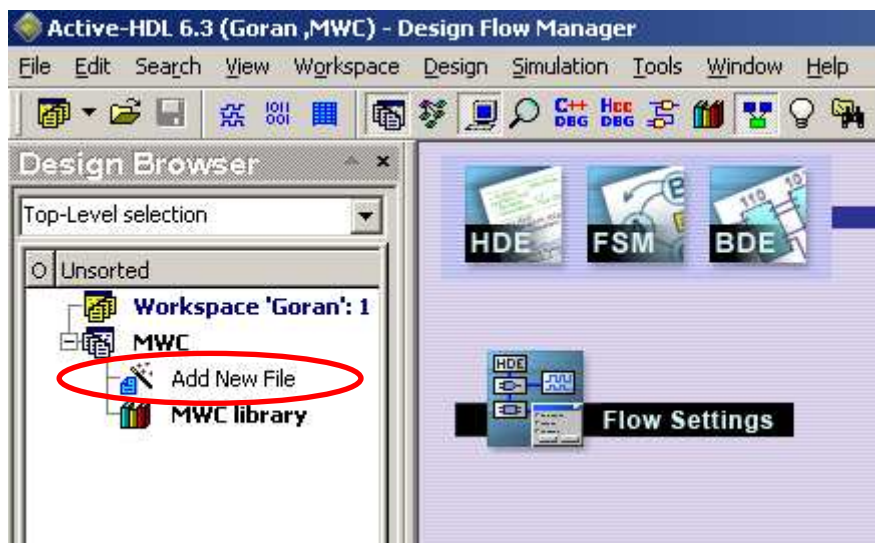
Kliknemo na Finish:



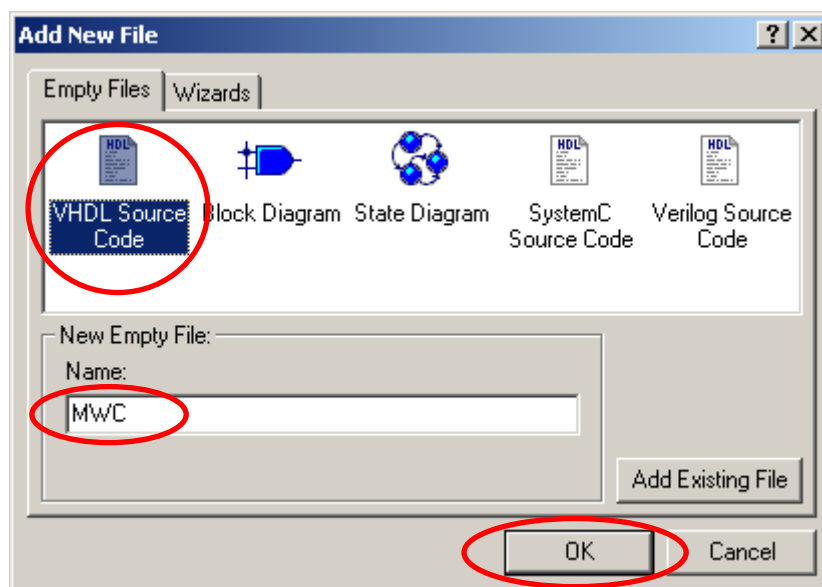
Ovim je priprema radnog prostora završena.

2.2.3 Unos VHDL koda

Nakon završene pripreme radnog prostora možemo uneti VHDL kod . Duplim klikom na Add New File unutar Design Browsera:



otvoriće se sledeći prozor:



Ukucamo ime novog fajla (ovde je to MWC), selektujemo VHDL Source Code i kliknemo na OK.

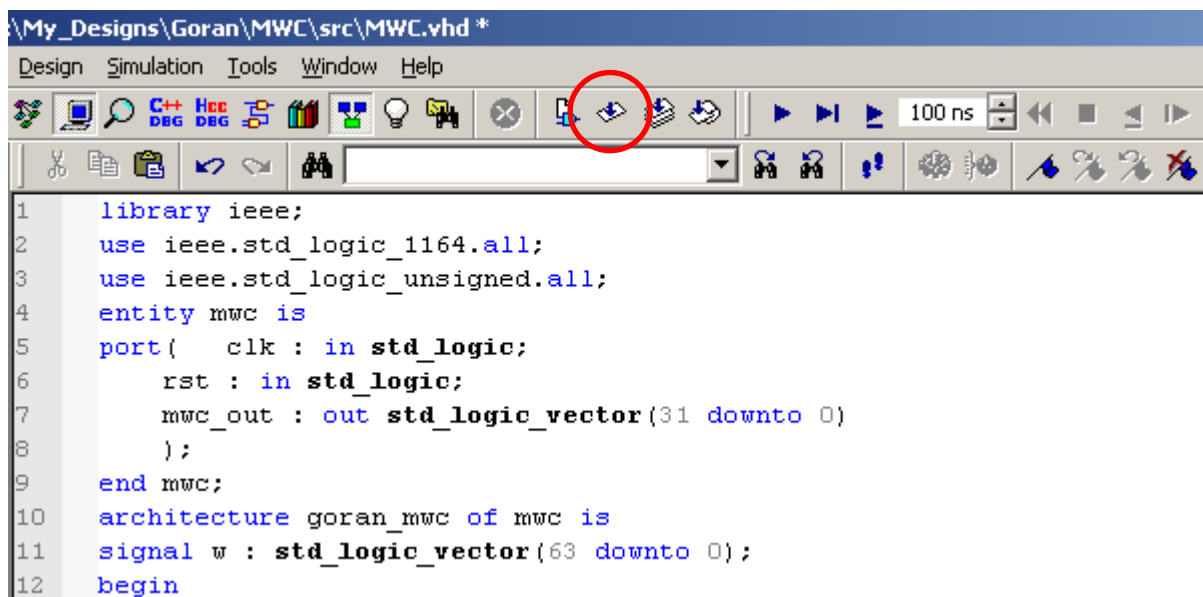
Otvoriće se prozor editora. U njemu pišemo nas VHDL kod, proveravamo njegovu sintaksnu ispravnost, editujemo ga ako je potrebno, i konačno, ako je kod ispravan, kompilujemo ga.

VHDL kod MWC generatora je veoma jednostavan:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity mwc is
port( clk : in std_logic;
      rst : in std_logic;
      mwc_out : out std_logic_vector(31 downto 0)
);
end mwc;
architecture goran_mwc of mwc is
signal w : std_logic_vector(63 downto 0);
begin
process (clk, rst)
begin
if rst='1' then
w <= "0000000000000000000000000000000000000000000000000000000000000001";
elsif clk'event and clk='1' then
w <= "01111100001101000100100010111110" * w(31 downto 0) +
      ("00000000000000000000000000000000" & w(63 downto 32));
end if;
end process;
mwc_out <= "00000000000000000000000000000000" when rst='1' else w(31 downto 0);
end goran_mwc;
```

Napomena: onaj egzotični broj u VHDL kodu ("01111100001101000100100010111110") je binarna reprezentacija našeg broja a = 2083801278.

Da bi smo proverili sintaksnu ispravnost koda i istovremeno izvršili njegovu kompilaciju kliknućemo na ikonu za kompilaciju:



U donjem delu prozora (Console) dobićemo izveštaj:

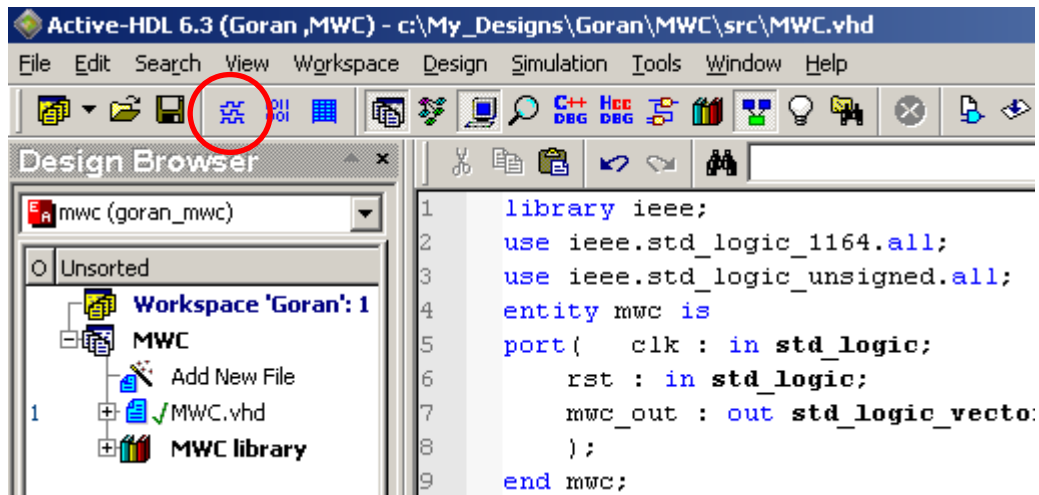
```
# File: c:\My_Designs\Goran\MWC\src\MWC.vhd
# Compile Entity "mwc"
# Compile Architecture "goran_mwc" of Entity "mwc"
# Top-Level unit detected
# Entity => mwc
# Compile success 0 Errors 0 Warnings Analysis time : 0.6 [s]
```

Ako postoje greške u kodu, ispravimo ih i ponovimo poslednju tačku.

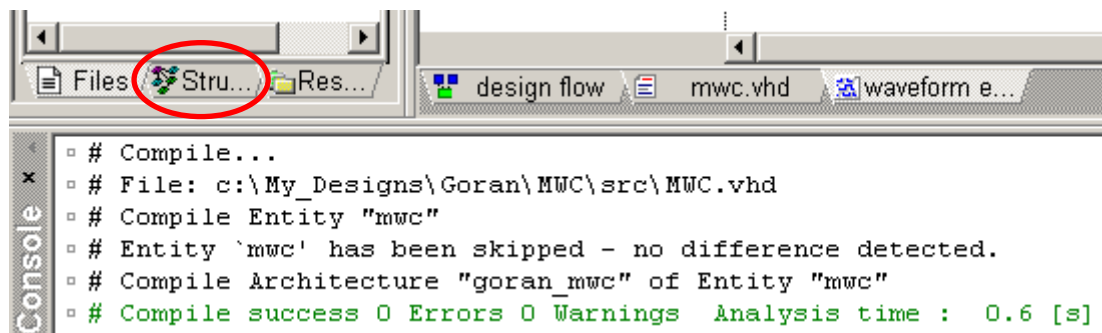
2.2.4 Simulacija VHDL koda

Funkcionalnu simulaciju VHDL koda MWC RNG izvršićemo posmatranjem talasnih oblika signala koje koristimo u kodu.

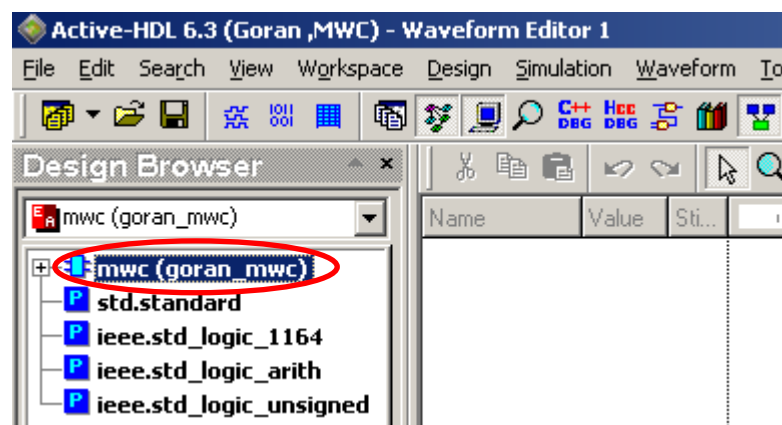
Da bi smo dobili talasne oblike najpre kliknemo na ikonu New Waveform:



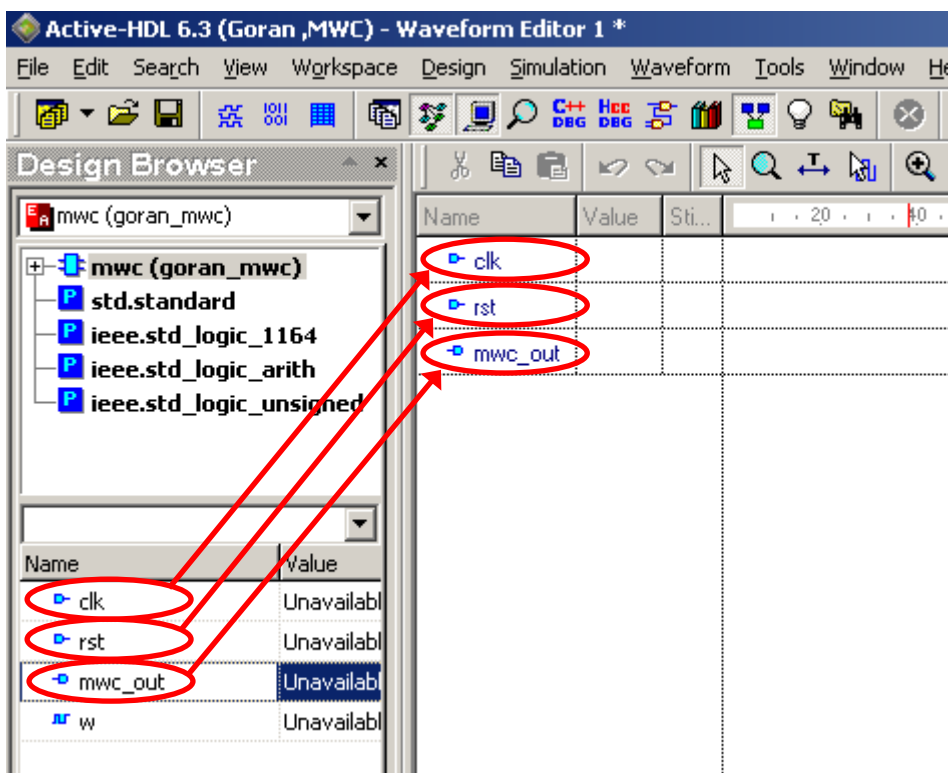
Zatim kliknemo na Structures:



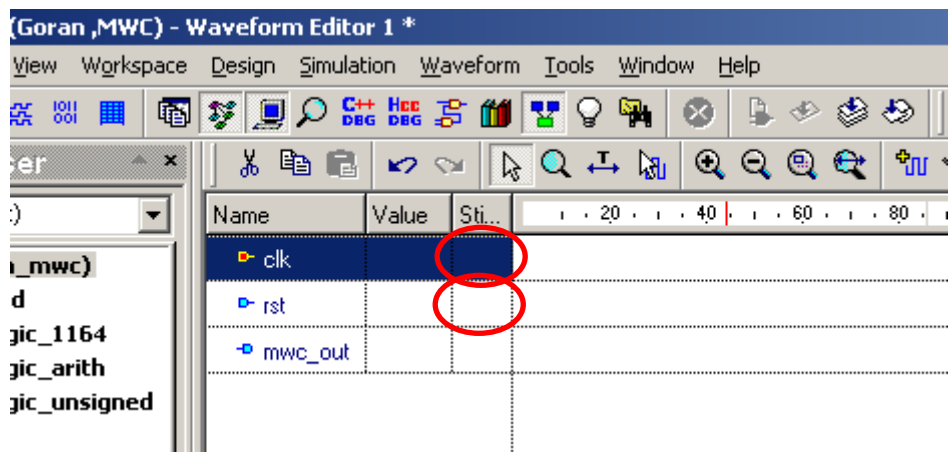
Kliknemo na entitet mwc (goran_mwc):



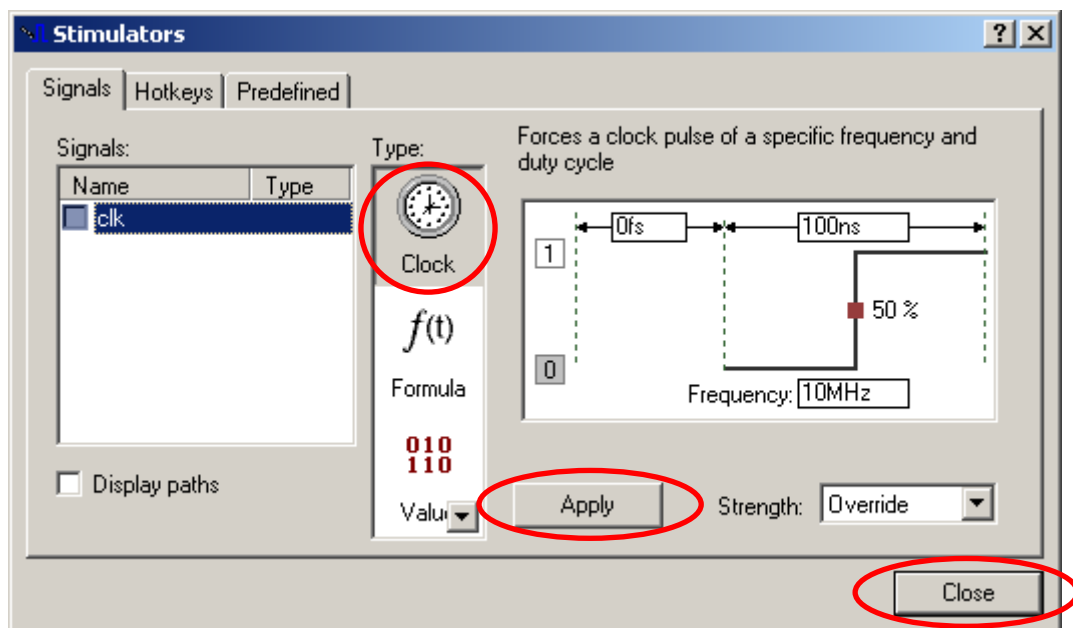
Pojaviće se signali koje koristimo u VHDL kodu. One signale čiji nas talasni oblici zanimaju prebacićemo u Waveform Editor tako što ih jednostavno "uhvatimo i prevučemo" u Waveform Editor:



Sada definišemo pobudu za ulazne signale, a to su clk i rst. Dvoklikom na Stimulator iza clk definišemo talasni oblik signala za clk, a dvoklikom na Stimulator iza rst - oblik signala za rst:

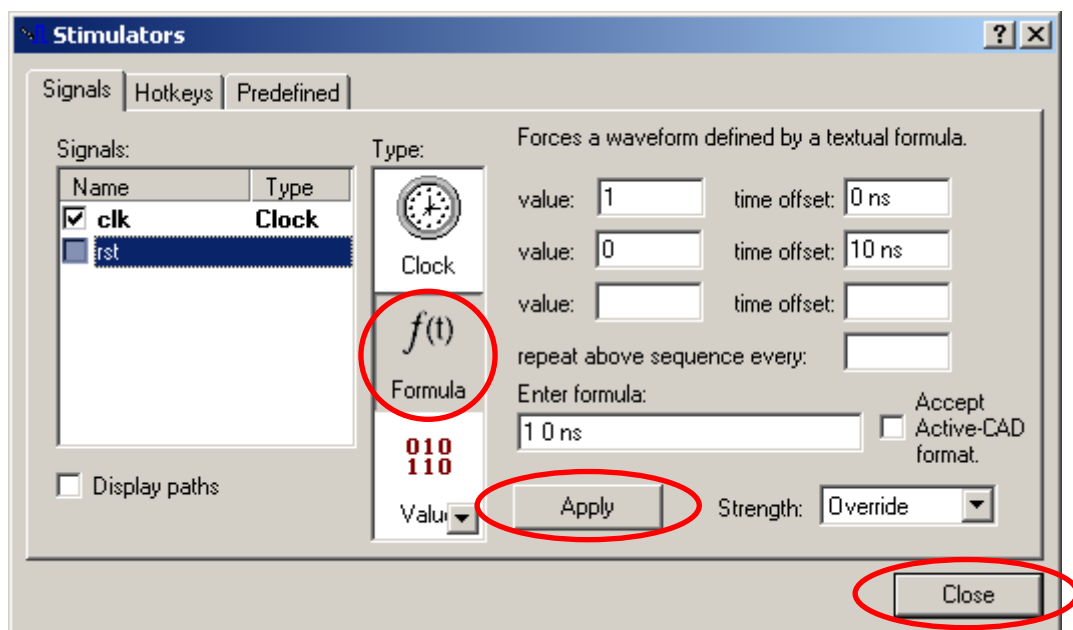


Nakon dvoklika na Stimulator za clk otvoriće se prozor:



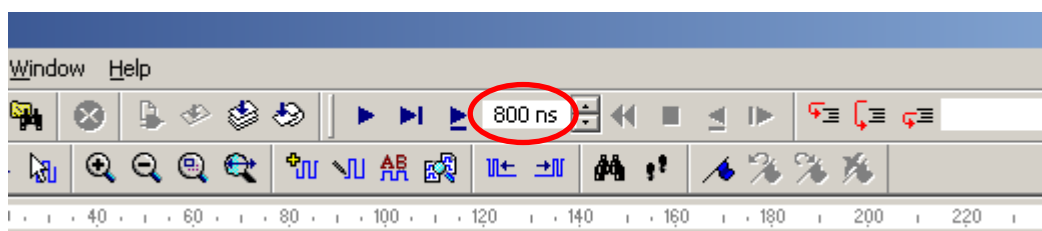
Kliknemo na ikonu Clock, podesimo parametre signala kao na slici i kliknemo na Apply pa na Close.

Analogno, nakon dvoklika na Stimulator za rst otvoriće se prozor:

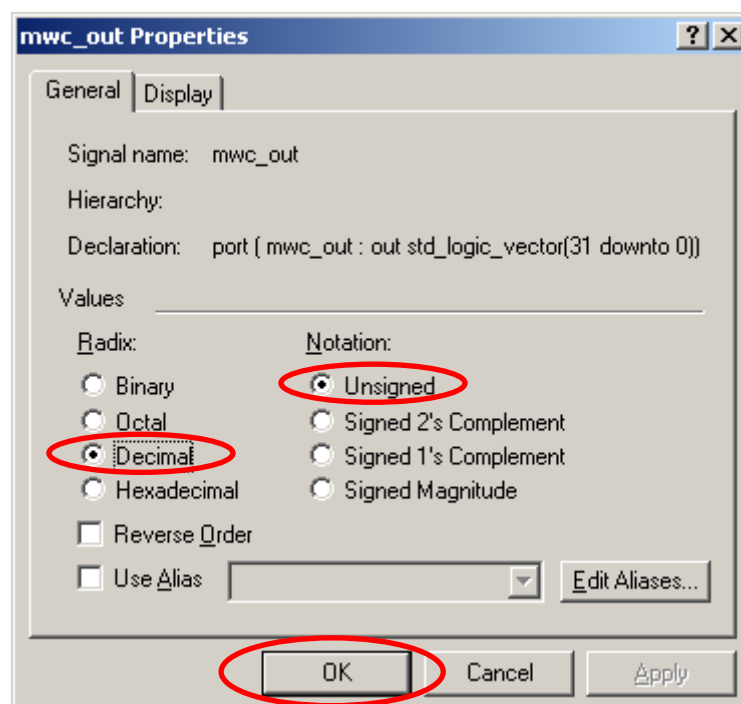
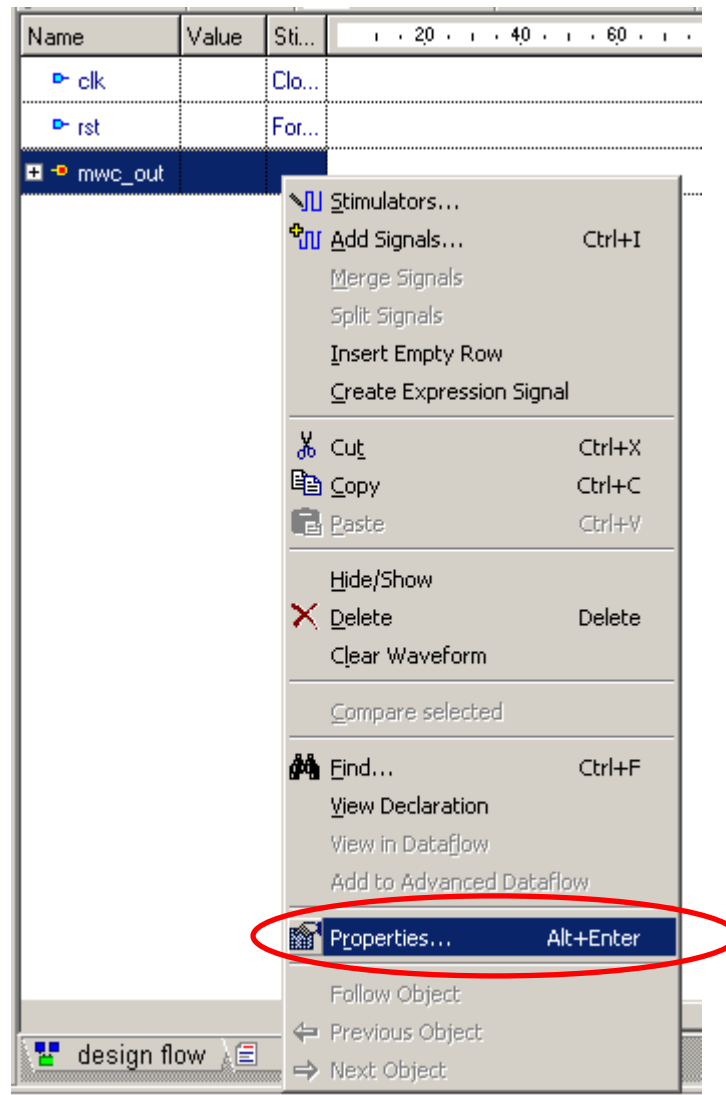


Kliknemo na ikonu Formula, unesemo vrednosti kao na slici i kliknemo na Apply pa na Close.

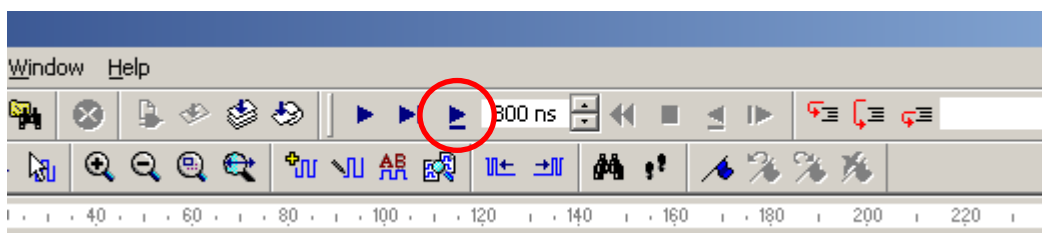
Sada podesimo vreme trajanja simulacije (ovde je to 800 ns):



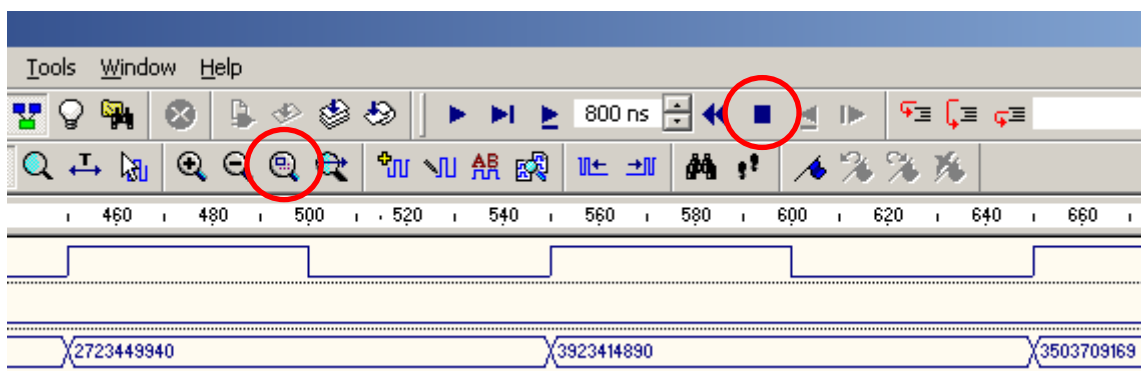
Konačno, podesimo osobine signala mwc_out, tako što desnim tasterom miša kliknemo na signal mwc_out a zatim kliknemo na Properties. U novootvorenom prozoru (mwc_out Properties) označimo Decimal i Unsigned i kliknemo na OK:



Sada možemo pokrenuti simulaciju klikom na ikonu Run For:



Nakon završetka simulacije, kliknemo na ikonu End simulation, a zatim na ikonu Zoom To Fit:

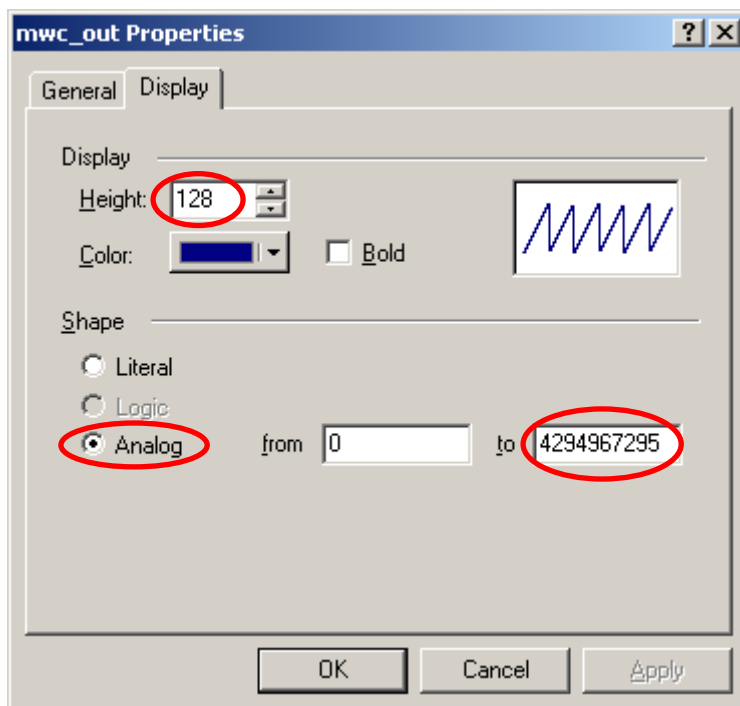


Simulacija pokazuje sledeće:

Name	V...	Sti...	50	100	150	200	250	300	350	400	450	5
clk	0	Clo...										
rst	0	For...										
mwc_out	59...		2083801278	2983947524	144095773	4100253040	2723449940					

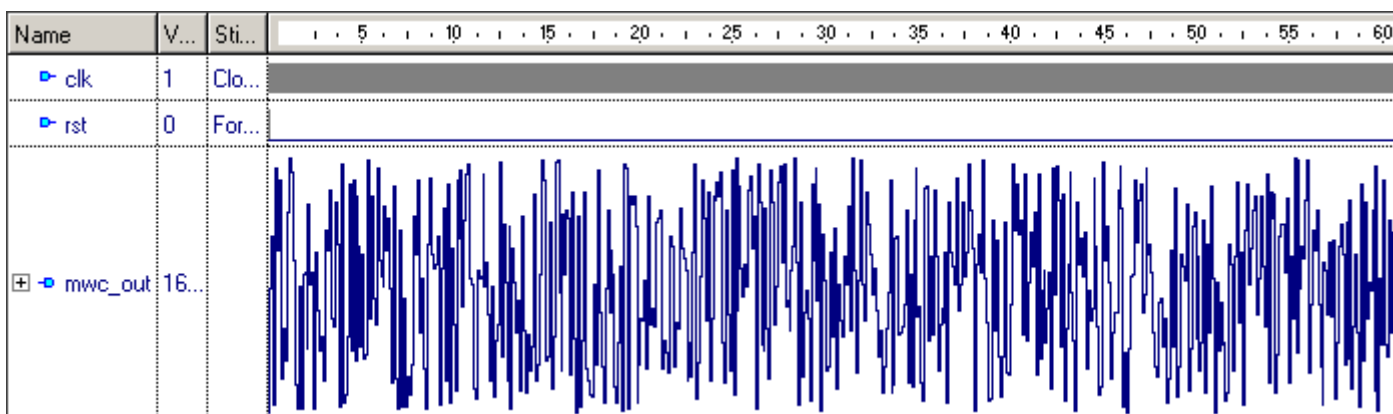
Signal mwc_out je `std_logic_vector (31 downto 0)`, a njegove vrednosti prevedene u decimalne brojeve se vide na prethodnoj slici. Upoređivanjem vrednosti koje smo dobili simulacijom VHDL koda, sa vrednostima koje je generisao program u C-u, konstatovaćemo da su identične, ☺.

Oblik signala mwc_out možemo prikazati i na drugaciji - analogni način. Podesimo vreme trajanja simulacije na 100 us. Zatim podesimo osobine signala mwc_out, tako što desnim tasterom miša kliknemo na signal mwc_out a zatim kliknemo na Properties. U novootvorenom prozoru kliknemo na zaglavlje Display i podesimo Height 128, Shape Analog from 0 to 4294967295 i kliknemo na OK.



Ponovo pokrećemo simulaciju klikom na ikonu Run For. Nakon završetka simulacije, kliknemo na ikonu End simulation, a zatim na ikonu Zoom To Fit.

Ovoga puta simulacija pokazuje analogni oblik signala mwc_out:



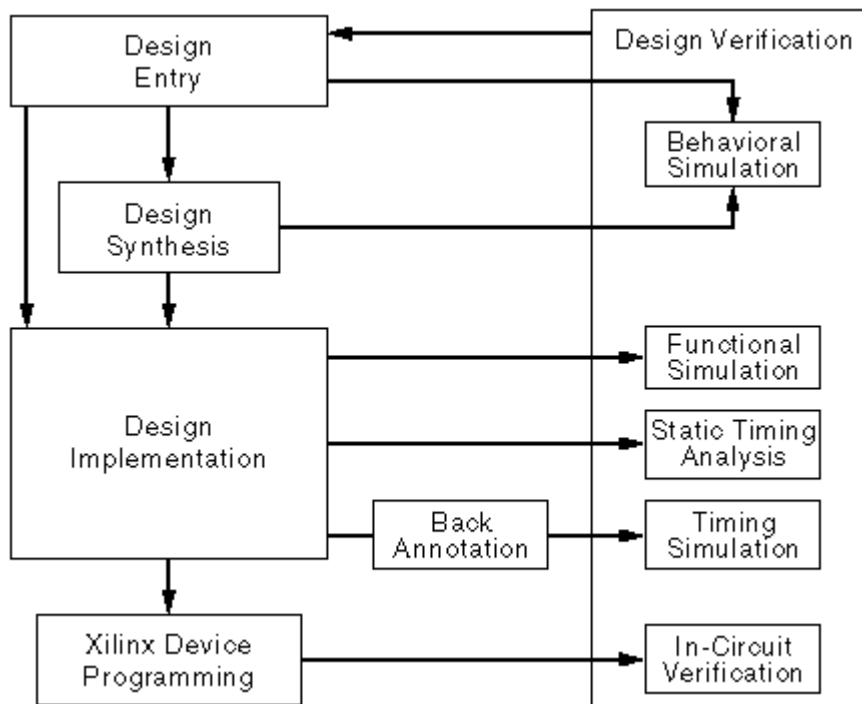
Ovim je proverena funkcionalna ispravnost VHDL opisa MWC RNG i sada se može preći na implementaciju koda u FPGA kolo.

2.3 Implementacija VHDL opisa MWC RNG u FPGA

Iako je VHDL opis MWC generatora pseudo-slučajnih brojeva veoma jednostavan, njegova implementacija u digitalno kolo zahteva veliki broj gejtova. Zato je u ovom slučaju odabrano FPGA kolo iz familije Spartan3E firme Xilinx, u koje je VHDL opis MWC generatora "stao" bez ikakvih problema. Ciljna tačka u ovom poglavlju je stići do trenutka kada se generiše fajl kojim se može programirati FPGA kolo. Korišćeni alat je Xilinx ISE.

2.3.1 Kratak opis alata Xilinx ISE

Ime alata potiče od Integrated Software Environment (ISE). Alat predstavlja potpuno okruženje za projektovanje programabilnih kola firme Xilinx, počev od unosa HDL opisa kola pa do verifikacije već programiranog i ugrađenog kola. Tok projektovanja je dat na sledećoj slici:



Project Navigator je program za upravljanje i procesiranje toka projektovanja kroz sledeće korake:

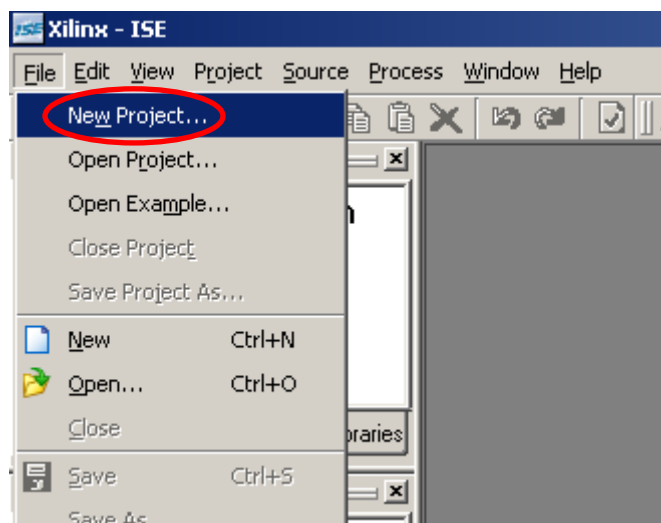
- Unošenje projekta
 - prvi korak u ISE toku projektovanja
 - kreiraju se izvorni fajlovi projekta po principu top-down korišćenjem jezika za opis hardvera (VHDL, Verilog, ABEL) ili korišćenjem šematskog unosa
- Sinteza
 - pokreće se nakon unosa i eventualne simulacije projekta
 - iz izvornih fajlova projekta kreiraju fajlovi netlisti neophodnih za implementaciju
- Implementacija
 - pokreće se nakon sinteze
 - projekat opisan na RTL nivou konvertuje se u fajl za programiranje izabranog logičkog kola
 - proces implementacije zavisi od tipa izabranog logičkog kola (FPGA ili CPLD)
- Verifikacija
 - projekat se može više puta proveravati u toku procesa projektovanja
 - verifikuje se funkcionalnost i tajming projekta ili njegovog dela
- Konfiguracija kola
 - nakon konvertovanja RTL opisa u fajl za programiranje, kolo se može programirati
 - generišu se konfiguracioni fajlovi i upišuju se pomoću kompjutera u izabrano Xilinx kolo

Pomoćni alati iz Xilinx ISE paketa su:

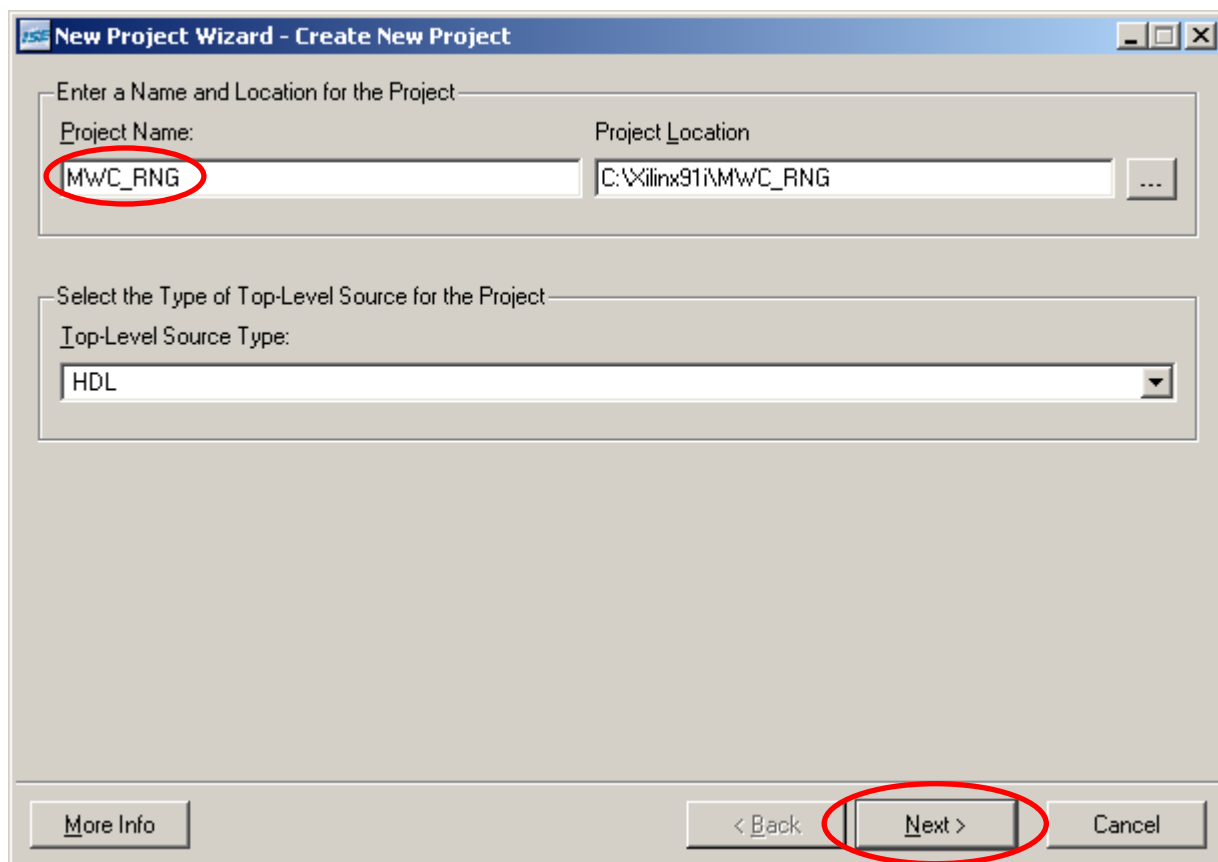
- Architecture Wizard
 - pomoć u kreiranju novog projekta
- Constraints Editor
 - Constraints su instrukcije ograničenja plasirane pri unosu opisa projekta
 - specifikuju razmeštaj pinova, stanja memorije flip-flopova i lečeva, globalni i tajming grupa portova, imena i smerove signala
 - pišu se u kotisničkim fajlovima ograničenja (User Constraints File - UCF)
- CORE Generator
 - generisanje fajlova korova koji štite intelektualnu svojinu (Intellectual Property - IP)
 - FIFO i memorije, Reed-Solomon Decoder i Encoder, FIR filtri, brza Furijerova transformacija (FFT), standardni bus interfejsi (PCI i PCI-X), konekcije i mreže (Ethernet, SPI-4.2, RapidIO, CAN i PCI Express)
- Floorplanner
 - planiranje detaljnog razmeštaja pinova
 - pregled i editovanje ograničenja razmeštaja
 - pronalaženje logičkih komponenti ili mreža na čipu
 - automatski razmeštaj portova
- FPGA Editor
 - prikaz i konfigurisanje FPGA
 - zahteva Native Circuit Description (.ncd) fajl koji sadrži logiku projekta mapiranu u odgovarajuće blokove kao što su CLB i IOB
 - razmeštaj i povezivanje kritičnih komponenti pre automatskog razmeštanja i povezivanja
 - završni razmeštaj i povezivanje ukoliko program za rutiranje nije sam kompletirao razmeštaj
 - pomoć pri analizi stanja signala i debugiranju kola
 - analiza tajminga
- iMPACT
 - pomoć pri konfigurisanju i realizaciji procesa programiranja kola
- PACE
 - pregled i editovanje ograničenja za lokacije I/O i globalne logike
 - kreiranje ograničenja za površinu koju zauzima logika
 - utvrđivanje zahteva za resursima izabranog kola koje zahteva projekat
- StateCAD
 - grafički način unošenja projekta pomoću dijagrama stanja
- Timing Analyzer
 - analiza statičkog tajminga FPGA i CPLD projekata
 - izveštaj o kašnjenju na putanjama i pomoć u analizi kritičnih putanja u kolu
 - set up i hold provere tajminga
 - analiza clock signala kod sinhronih sistema
 - kreiranje izveštaja o tajmingu
- XPower
 - analiza potrošnje FPGA i CPLD kola
 - analiza temperature na spojevima
 - provera pravila projektovanja (DRC)

2.3.2 Priprema projekta

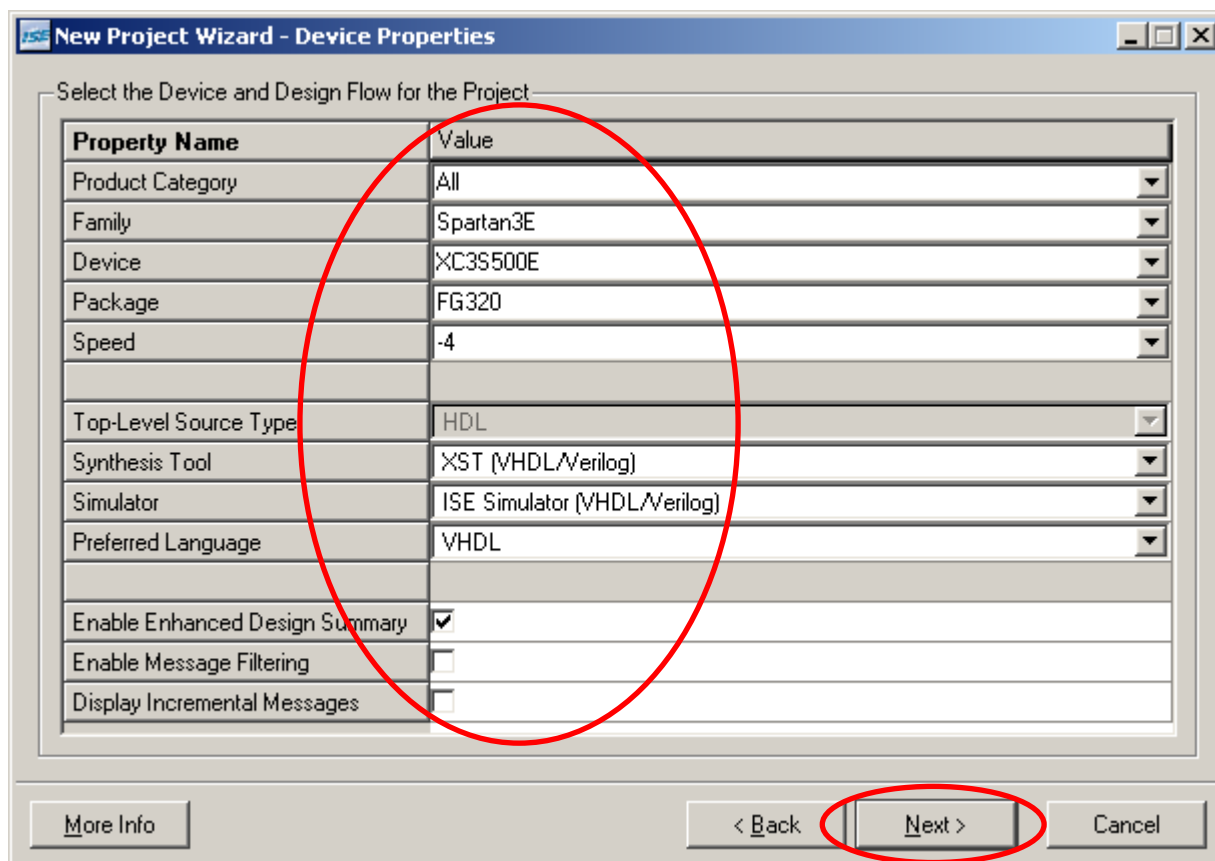
Nakon pokretanja Project Navigator-a kliknemo na File pa na New Project:



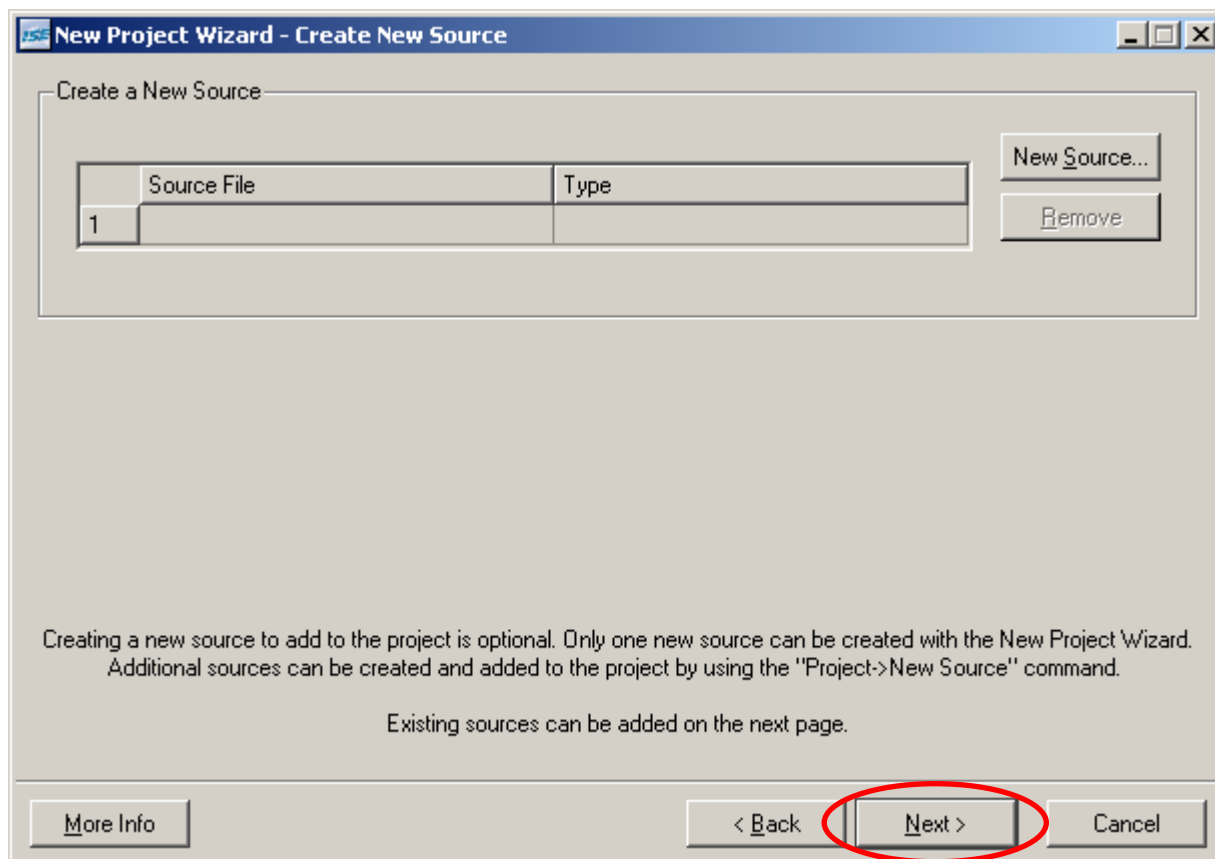
Time se pokreće New Project Wizard. U polju Project Name upišemo ime projekta (ovde je to MWC_RNG) i kliknemo na Next >:



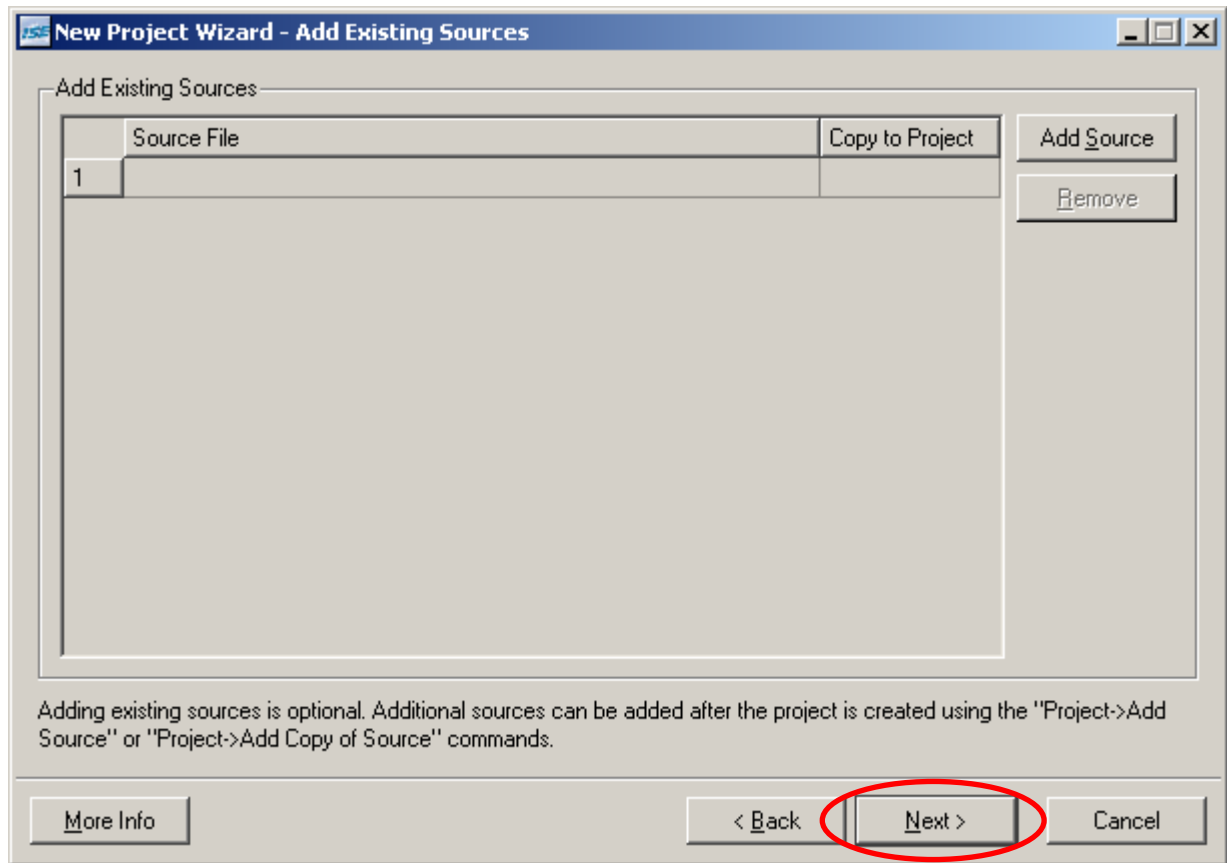
Sada treba definisati parametre izabranog FPGA kola kao na slici pa kliknuti na Next >:



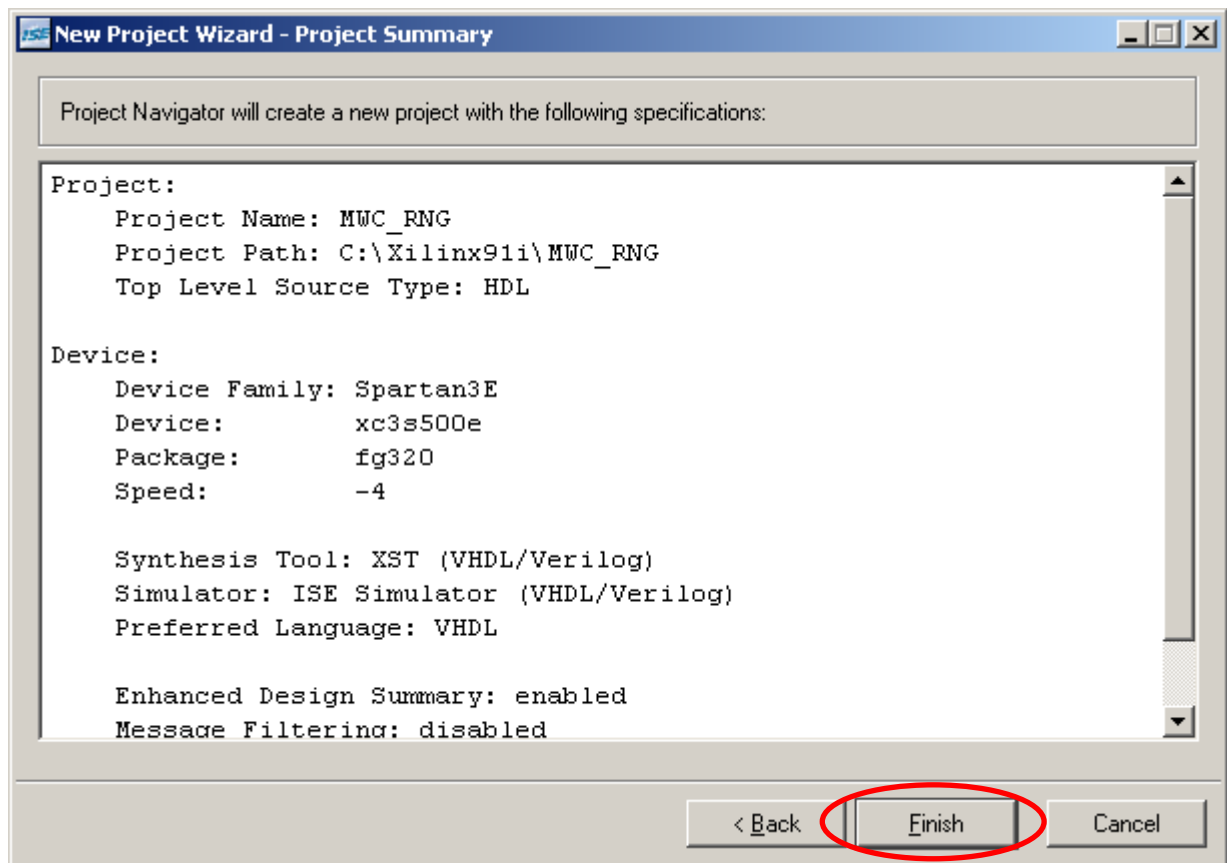
Samo kliknuti na Next >:



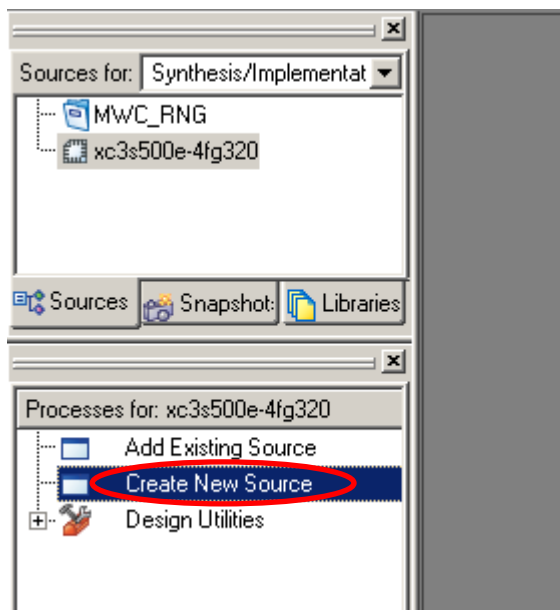
Kliknuti na Next >:



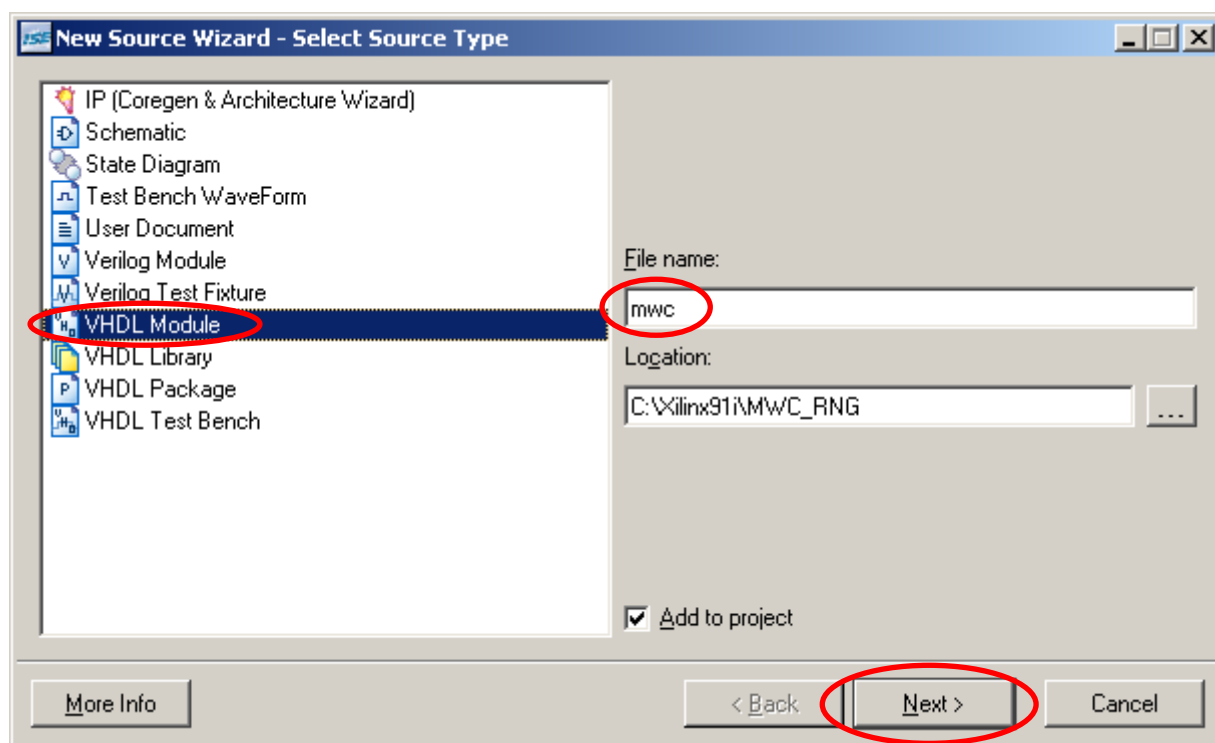
Kliknuti na Finish:



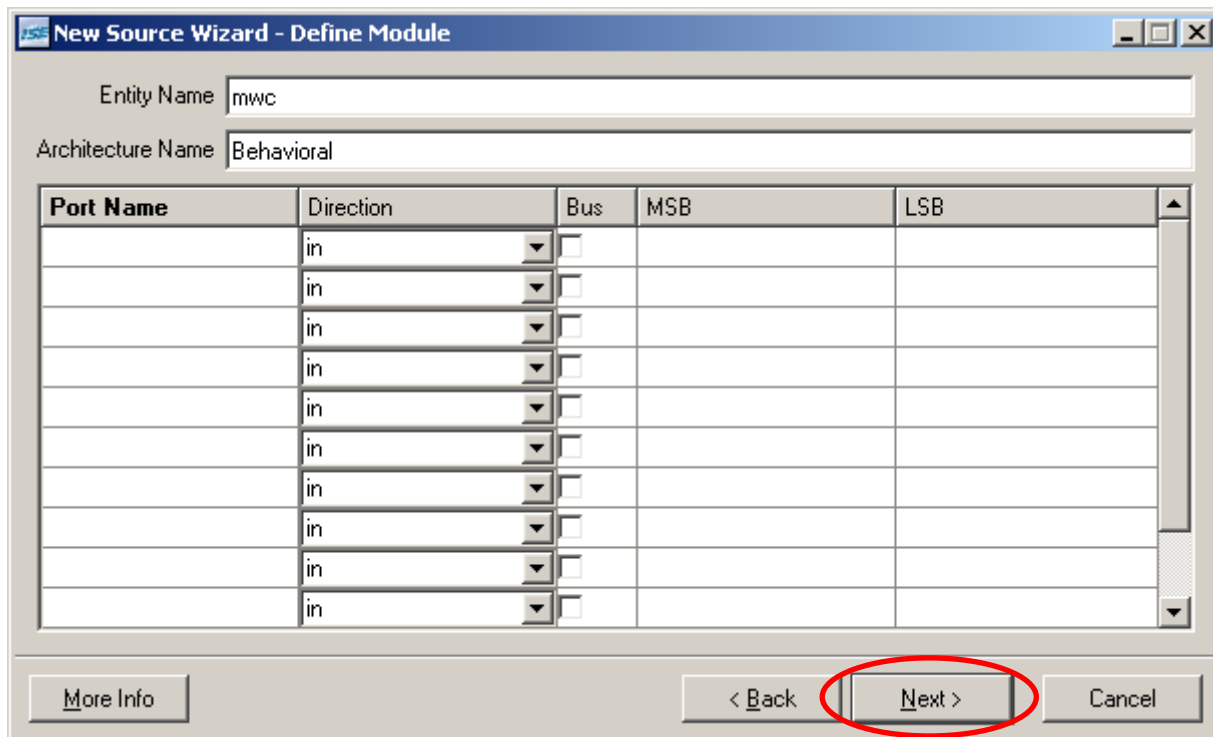
Kliknemo dva puta na Create New Source:



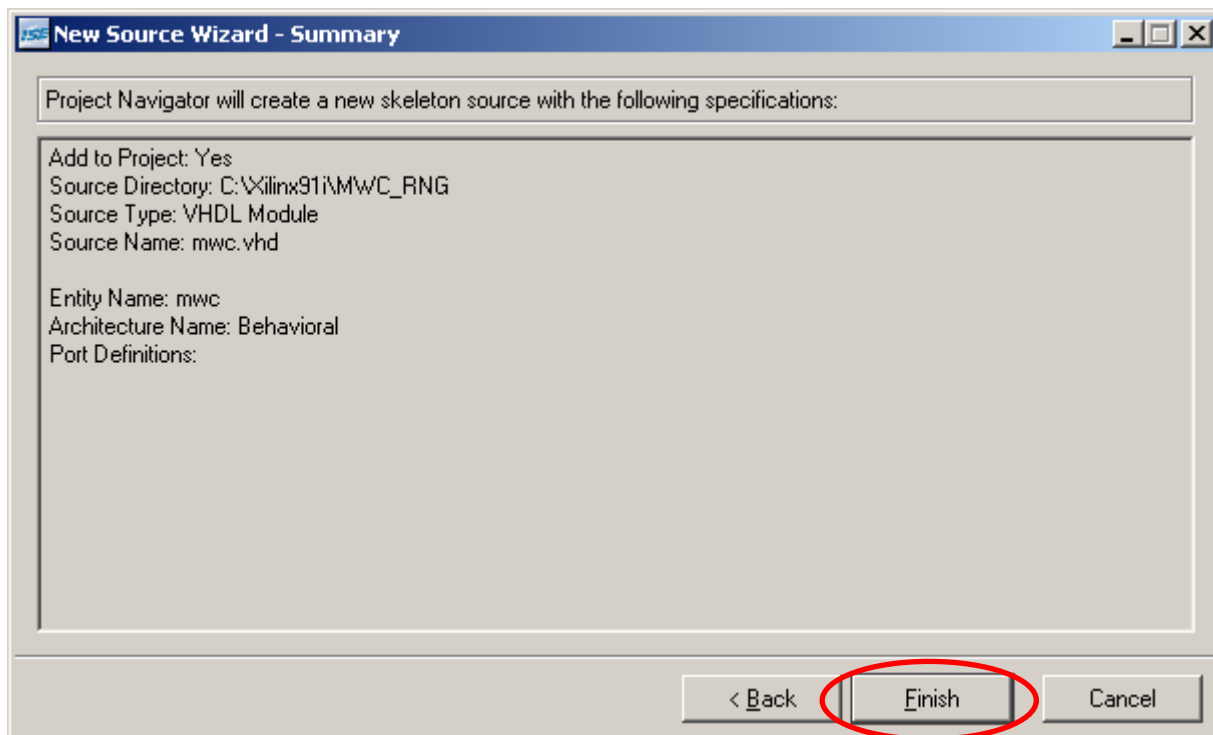
Upišemo ime fajla (ovde je to mwc), zatim selektujemo VHDL Module i kliknemo na Next >:



Samo kliknemo na Next >:



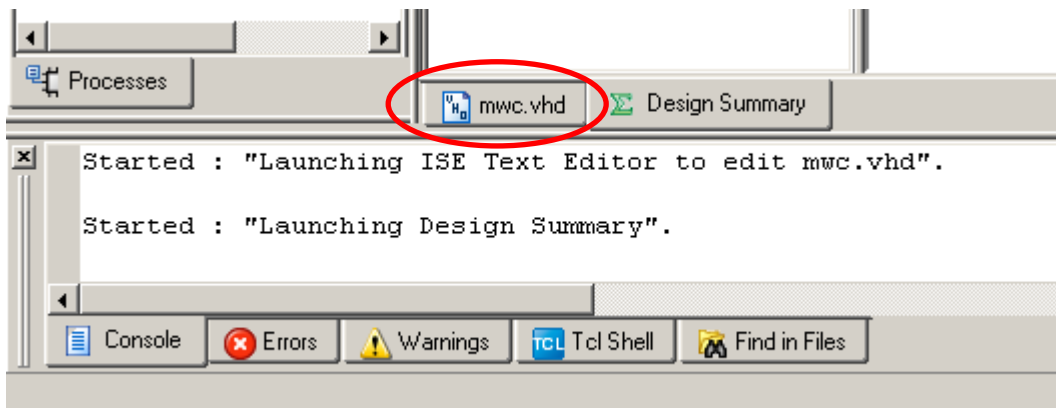
Kliknemo na Finish:



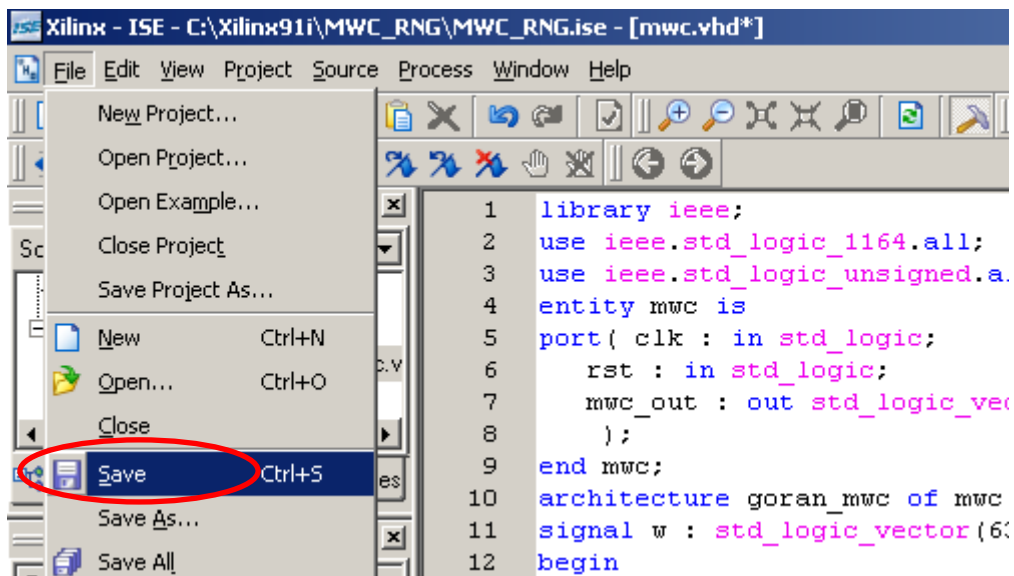
Ovim je priprema projekta završena.

2.3.3 Unos VHDL koda

Nakon završene pripreme projekta potrebno je da unesemo VHDL kod . Najpre kliknemo na zaglavlje editora, mwc.vhd:



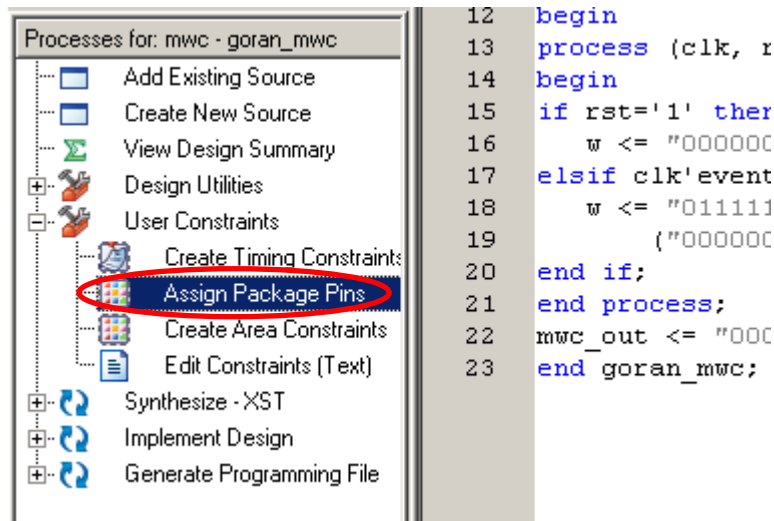
Sada u editoru selektujemo i izbrisemo (Delete na tastaturi) kod koji je program automatski kreirao. Zatim unesemo naš kod za MWC generator. Uneti VHDL kod treba sačuvati. Kliknemo na File pa na Save:



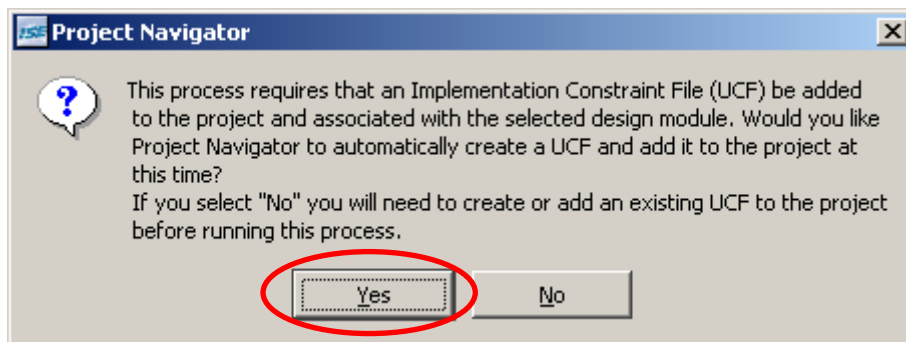
2.3.4 Unos ograničenja

Unećemo samo ograničenja koja se tiču fizičkog razmeštaja pinova izabranog FPGA kola.

Najpre, kliknemo na + ispred User Constraints, zatim kliknemo dva puta na Assign Package Pins:



Kliknemo na Yes:

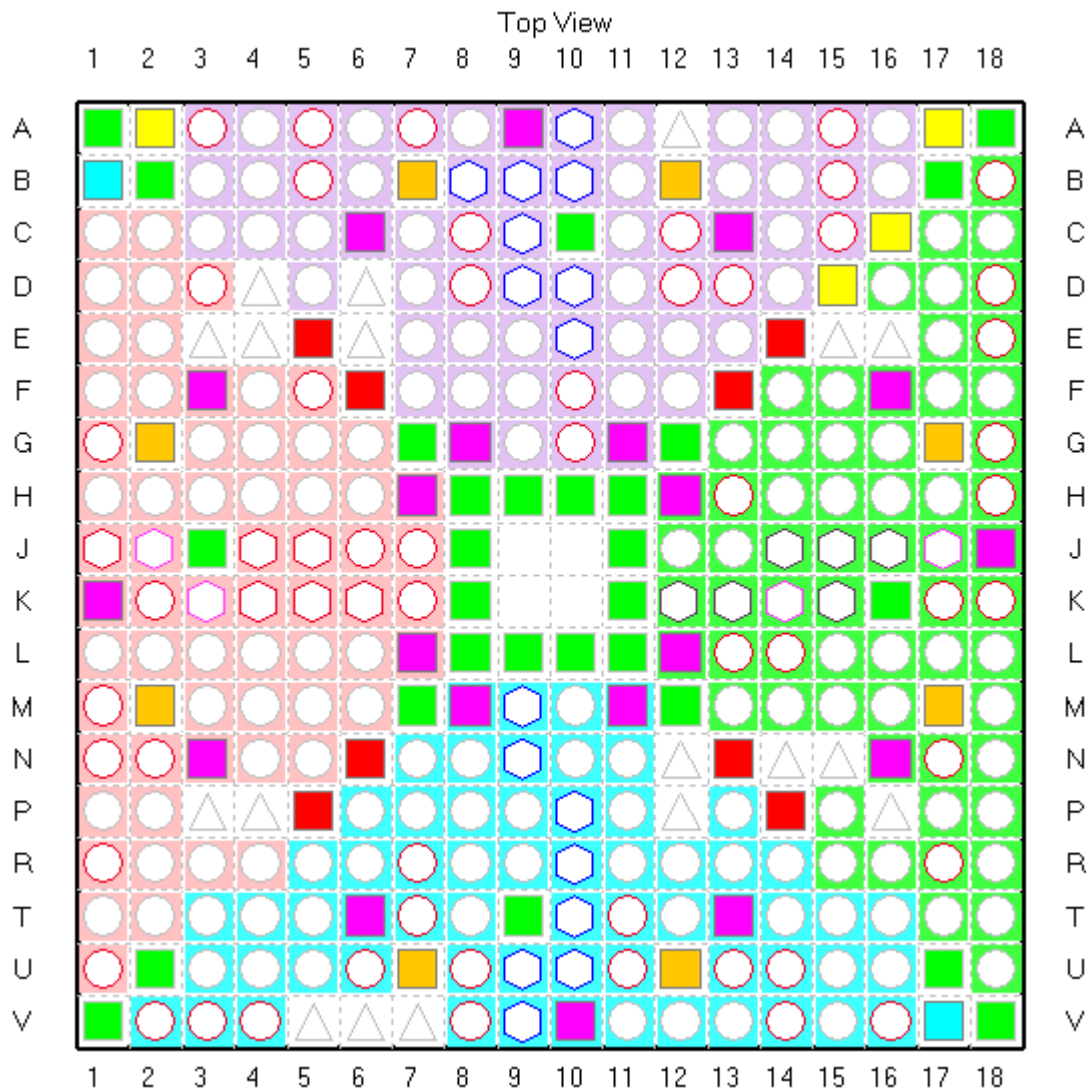


Pokrenuće se jedan od pomoćnih alata - Xilinx PACE.

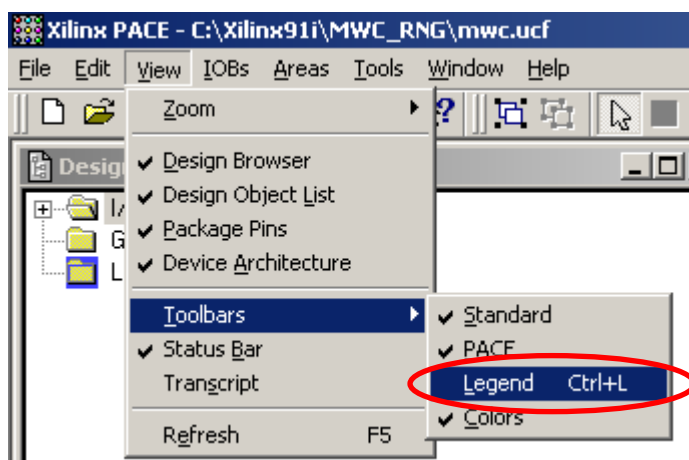
U donjem delu prozora treba kliknuti na zaglavlje Package View:



Pojaviće se XC3S500E FPGA sa svih svojih 320 (!) pinova:



Da bi smo se malo snašli u tom polju pinova kliknemo na View, pa na Toolbars i zatim na Legend:

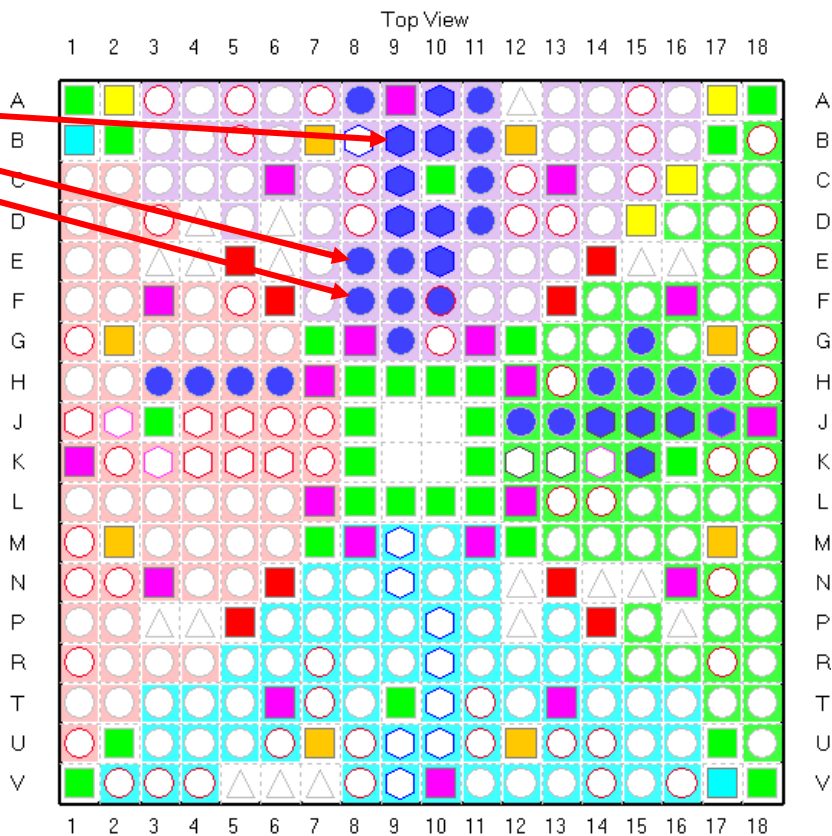


Symbol	Pin Type
	User IO
	Input Only
	User Prohibit
	GND
	VCCINT
	VCCAUX
	VCCO
	CONFIG
	JTAG
	GCLK / GCK
	Left Hand Clock
	Right Hand Clock
	IRDY / TRDY / LHCLK / RHCLK
	Power Management
	Not Connected
	Bank0
	Bank1
	Bank2
	Bank3

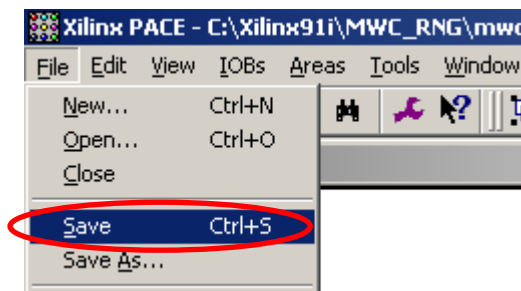
Sada, pošto nam je legenda pomogla oko snalaženja u polju pinova, ☺, možemo izvršiti njihov fizički razmeštaj u smislu dodele signala pojedinim pinovima.

Kliknemo na polje Loc za neki od signala i upišemo lokaciju pina na kojoj će biti taj signal. Svaka dodela signala nekom pinu odmah bude označena plavom bojom u prozoru koji pokazuje njihov razmeštaj:

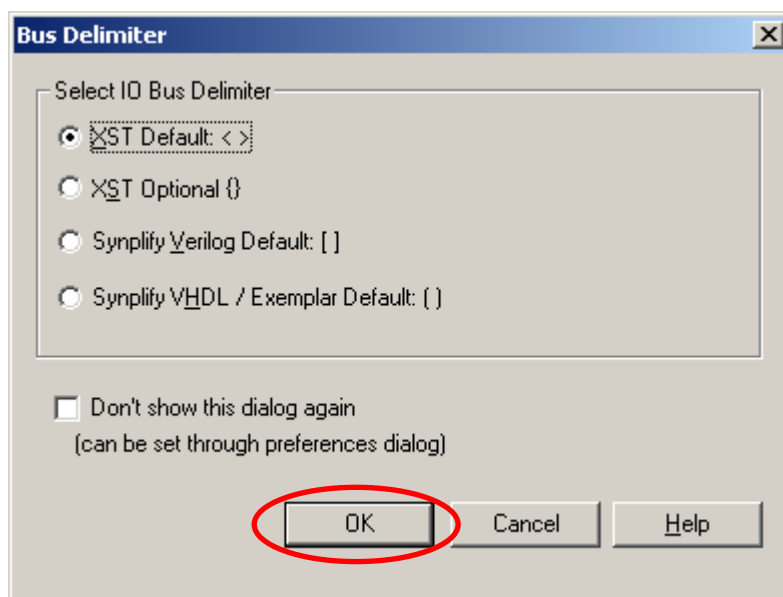
I/O Name	I/O Direction	Loc	Bank
clk	Input	B9	BANK
mwc_out<0>	Output	E8	BANK
mwc_out<1>	Output	F8	BANK
mwc_out<2>	Output	J17	BANK
mwc_out<3>	Output	J16	BANK
mwc_out<4>	Output	D10	BANK
mwc_out<5>	Output	H3	BANK
mwc_out<6>	Output	H5	BANK
mwc_out<7>	Output	H4	BANK
mwc_out<8>	Output	H15	BANK
mwc_out<9>	Output	H6	BANK
mwc_out<10>	Output	A8	BANK
mwc_out<11>	Output	F9	BANK
mwc_out<12>	Output	E10	BANK
mwc_out<13>	Output	A11	BANK
mwc_out<14>	Output	B10	BANK
mwc_out<15>	Output	J12	BANK
mwc_out<16>	Output	D9	BANK
mwc_out<17>	Output	G9	BANK
mwc_out<18>	Output	H14	BANK
mwc_out<19>	Output	H17	BANK
mwc_out<20>	Output	D11	BANK
mwc_out<21>	Output	B11	BANK
mwc_out<22>	Output	J13	BANK
mwc_out<23>	Output	C9	BANK
mwc_out<24>	Output	G15	BANK
mwc_out<25>	Output	E9	BANK
mwc_out<26>	Output	C11	BANK
mwc_out<27>	Output	J15	BANK
mwc_out<28>	Output	J14	BANK
mwc_out<29>	Output	K15	BANK
mwc_out<30>	Output	A10	BANK
mwc_out<31>	Output	H16	BANK
rst	Input	F10	BANK



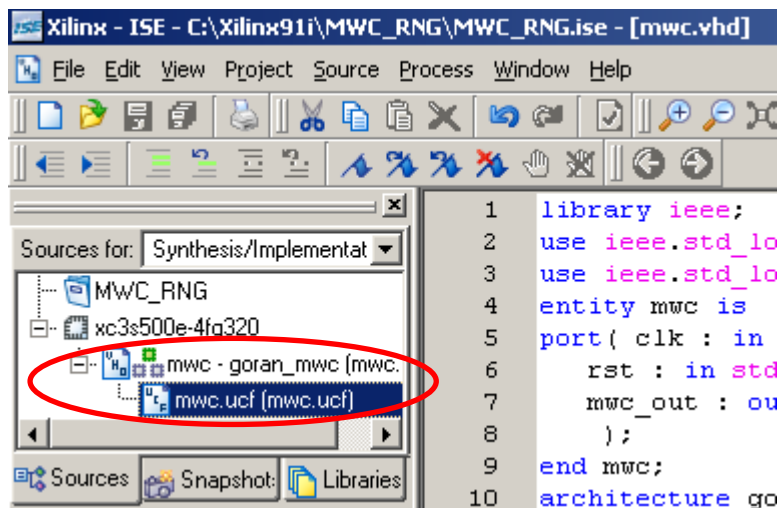
Nakon izvršenog dodeljivanja signala pinovima, potrebno je to i sačuvati. Kliknemo na File, zatim na Save:



Pojaviće se Dialog Box u kome samo treba kliknuti na OK:



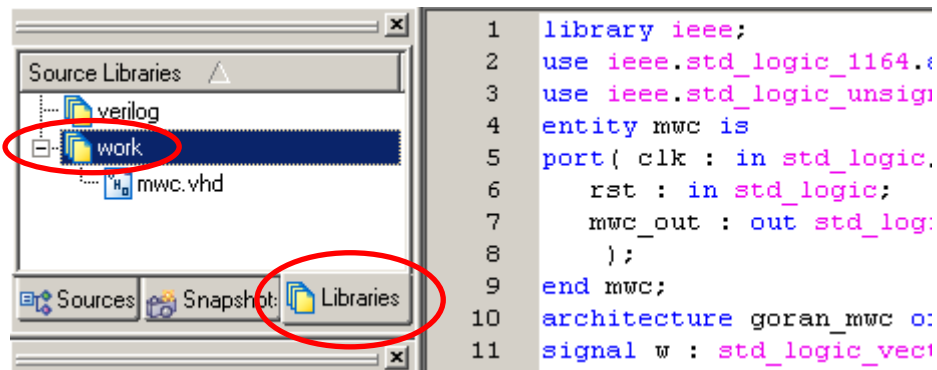
Dodeljivanje signala pinovima koje smo upravo uradili predstavlja ograničenje za softver, kog se on mora pridržavati u daljem toku implementacije. Fajl koji sadrži ograničenja je User Constraints File - UCF. Prethodnom naredbom (Save), softver je upravo generisao UCF fajl i dodao ga našem projektu. Vratimo se nazad na naš projekat, kliknemo na + ispred mwc - goran_mwc i proverimo: mwc.ucf fajl je zaista dodat našem projektu:



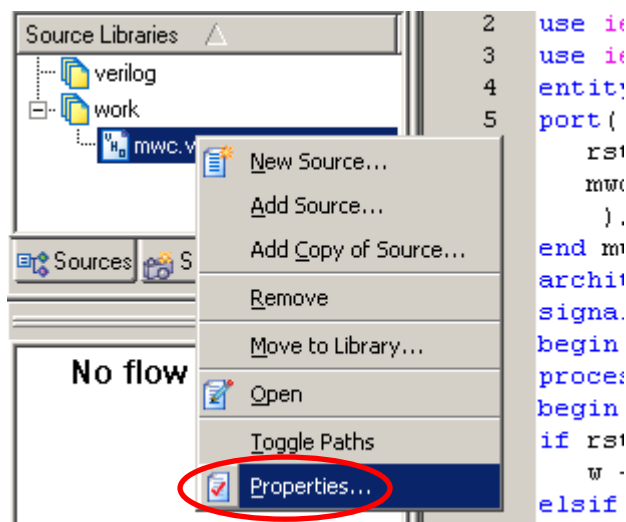
2.3.5 Implementacija

Xilinx ISE je veoma automatizovani softwer. Nakon unošenja VHDL opisa i ograničenja, potrebno je još par puta kliknuti mišem da bi se stiglo do krajnje tačke imlementacije, koju smo (u ovom projektu) definisali kao generisanje fajla za programiranje FPGA kola. Sve međufaze u toku projektovanja softwer može obaviti samostalno i po automatizmu. Nećemo postavljati nikakva ograničenja vezana za tajming. Funkcionalnu simulaciju koda smo već obavili u programu Active-HDL i sada to nećemo ponavljati.

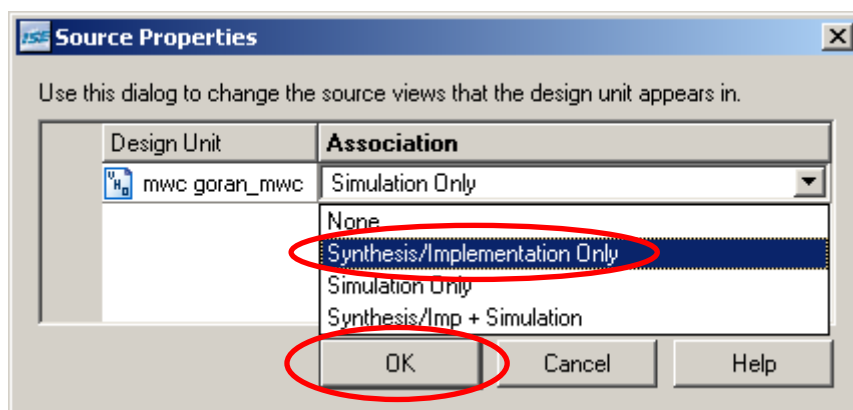
Najpre treba izvršiti nekoliko podešavanja. Kliknemo na zaglavlje Libraries a zatim na + ispred biblioteke work:



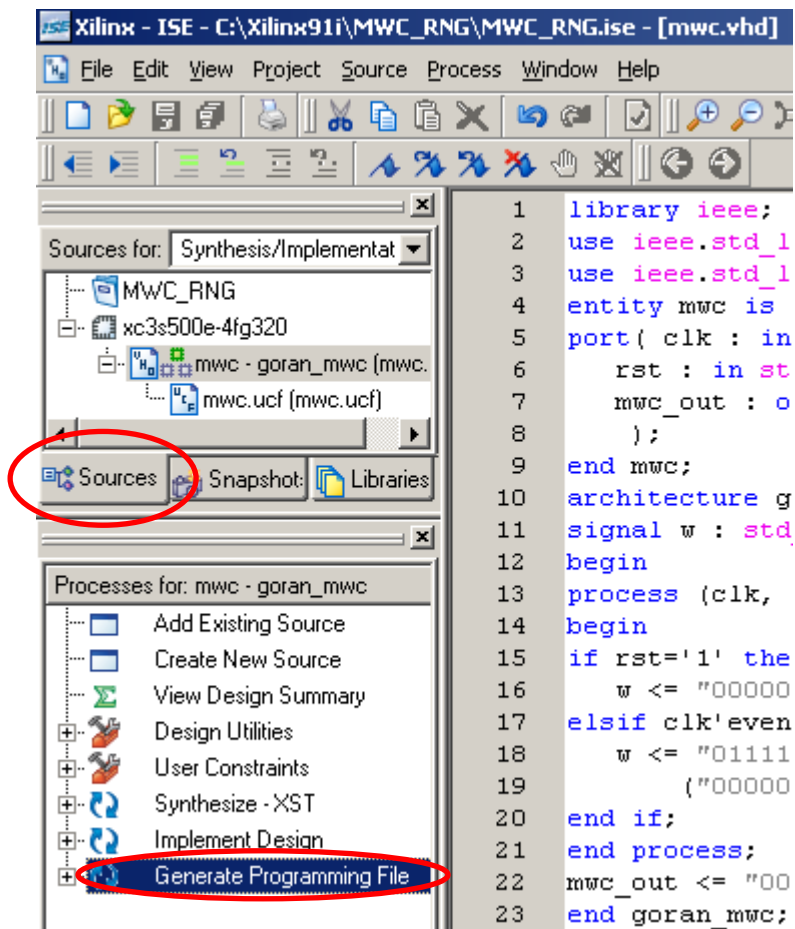
Desnim tasterom kliknemo na fajl mwc.vhd a zatim kliknemo na Properties...:



U prozoru Source Properties kliknemo na Synthesis/Implementation Only, a zatim na OK:

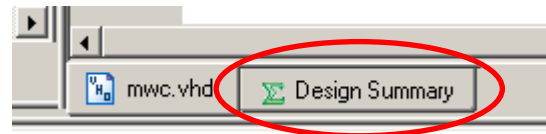


Sada se vratimo nazad klikom na zaglavlje Sources, zatim kliknemo dva puta na Generate Programming File i odemo da gledamo našu omiljenu seriju o Džordžu, dok softver sam dovršava projekat, ☺.



2.3.6 Izveštaji implementacije

Nakon komande Generate Programming File, softver je od VHDL opisa MWC generatora prošao sam kroz celokupan tok projektovanja sve do generisanja fajla kojim možemo programirati izabrano FPGA kolo. Usput je, o svakom od preduzetih koraka generisao iscrpan izveštaj. Rezime izveštaja implementacije dobijamo klikom na zaglavlje Design Summary:



MWC_RNG Project Status			
Project File:	MWC_RNG.isc	Current State:	Programming File Generated
Module Name:	mwc	• Errors:	No Errors
Target Device:	xc3s500e-4fg320	• Warnings:	1 Warning
Product Version:	ISE 9.1.01i	• Updated:	Sat 9. Feb 01:40:51 2008


MWC_RNG Partition Summary
No partition information was found.

Device Utilization Summary				
Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	64	9,312	1%	
Number of 4 input LUTs	153	9,312	1%	
Logic Distribution				
Number of occupied Slices	101	4,656	2%	
Number of Slices containing only related logic	101	101	100%	
Number of Slices containing unrelated logic	0	101	0%	
Total Number of 4 input LUTs	201	9,312	2%	
Number used as logic	153			
Number used as a route-thru	48			
Number of bonded IOBs	34	232	14%	
Number of GCLKs	1	24	4%	
Number of MULT18X18SIOs	4	20	20%	
Total equivalent gate count for design	2,426			
Additional JTAG gate count for IOBs	1,632			

Performance Summary			
Final Timing Score:	0	Pinout Data:	Pinout Report
Routing Results:	All Signals Completely Routed	Clock Data:	Clock Report
Timing Constraints:	All Constraints Met		

Detailed Reports					
Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Current	Sat 9. Feb 01:39:53 2008	0	1 Warning	0
Translation Report	Current	Sat 9. Feb 01:40:01 2008	0	0	0
Map Report	Current	Sat 9. Feb 01:40:12 2008	0	0	3 Infos
Place and Route Report	Current	Sat 9. Feb 01:40:34 2008	0	0	2 Infos
Static Timing Report	Current	Sat 9. Feb 01:40:40 2008	0	0	3 Infos
Bitgen Report	Current	Sat 9. Feb 01:40:51 2008	0	0	0

Odmah se može videti da je procenat korišćenja resursa (Utilization) FPGA kola XC3S500E izrazito mali, što znači da je 32-bitni MWC RNG bez problema stao u FPGA kolo. Ovo se sigurno ne bi dogodilo da je na početku projektovanja bilo izabrano bilo koje od Xilinx CPLD kola. Naime, autor teksta je probao da implementira 16-to bitnu verziju MWC generatora u CPLD kola serije XC9500 i nije mu uspelo. Softver je redovno prijavljivao da izabrana CPLD kola nemaju dovoljno resursa za realizaciju VHDL opisa. Ono što je uzimalo najviše resursa je množač.

Ostalim izveštajima implementacije jednostavno se pristupa klikom na ime izveštaja u Design Summary. Iz tih izveštaja se na "početnu stranu" vraćamo klikom na ikonu .

[Synthesys Report](#) je tekstualni fajl generisan u toku same implementacije. Podeljen je na više sekcija. Jedna od njih se odnosi na to kako je softver video naš VHDL kod MWC generatora. Naime, softver je u kodu "prepoznao":

# Multipliers	: 1
32x32-bit multiplier	: 1
# Adders/Subtractors	: 1
64-bit adder	: 1
# Registers	: 64
Flip-Flops	: 64

A evo sa čime je VHDL kod realizovao:

Cell Usage	
# IOs	: 34
# BELS	: 533
# GND	: 1
# LUT1	: 47
# LUT2	: 151
# LUT3	: 1
# LUT4	: 1
# MUXCY	: 166
# VCC	: 1
# XORCY	: 165
# FlipFlops/Latches	: 64
# FDC	: 63
# FDP	: 1
# Clock Buffers	: 1
# BUFGP	: 1
# IO Buffers	: 33
# IBUF	: 1
# OBUF	: 32
# MULTs	: 4
# MULT18X18SIO	: 4

Korišćenje resursa:

Number of Slices:	103	out of	4656	2%
Number of Slice Flip Flops:	64	out of	9312	0%
Number of 4 input LUTs:	200	out of	9312	2%
Number of bonded IOBs:	34	out of	232	14%
Number of MULT18X18SIOs:	4	out of	20	20%
Number of GCLKs:	1	out of	24	4%

Tajming:

Minimum period: 16.376ns (14.225ns logic, 2.151ns route) (86.9% logic, 13.1% route)
Maximum Frequency: 61.065MHz
Maximum output required time after clock: 5.597ns
Maximum combinational path delay: 7.071ns

[Pinout Report](#) je tabelarni prikaz pinova, njihovih imena, signala koji su im dodeljeni, smeru (O/I) i niza drugih parametara.

[Static Timing Report](#) je izveštaj koji sadrži statičke tajminge kola:

- Clock to Pad
- Clock to Setup on destination
- Pad to Pad

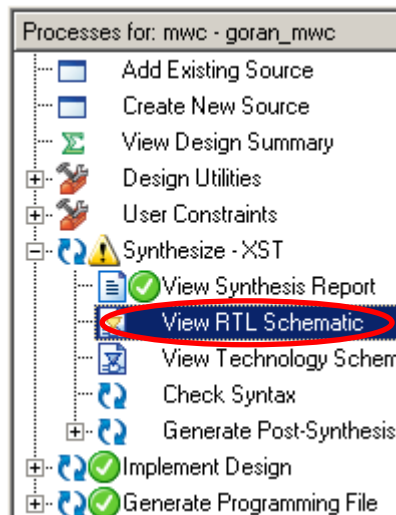
[Clock Report](#) sadrži izveštaj o tome koji bafer je upotrebljen za clock signal, koliki je fanout, kosina (skew) i maksimalno kašnjenje.

Izveštaji implementacije su veoma opširni. Njih projektant može koristiti u proceni performansi i funkcionalnosti projektovanog sistema. Analizom izveštaja projektant je u mogućnosti da uoči kritična mesta, eventualne nedostatke ili greške, ili da koriguje svoj projekat u smislu poboljšanja performansi.

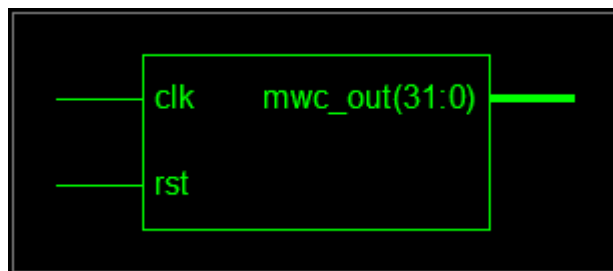
2.3.7 RTL i tehnološka šema

Interesantno je pogledati kako je softver prepoznao VHDL kod MWC RNG i konvertovao ga u šeme na RTL i tehnološkom nivou.

Za RTL šemu treba kliknuti na + ispred Synthesize i dva puta kliknuti na View RTL Schematic:



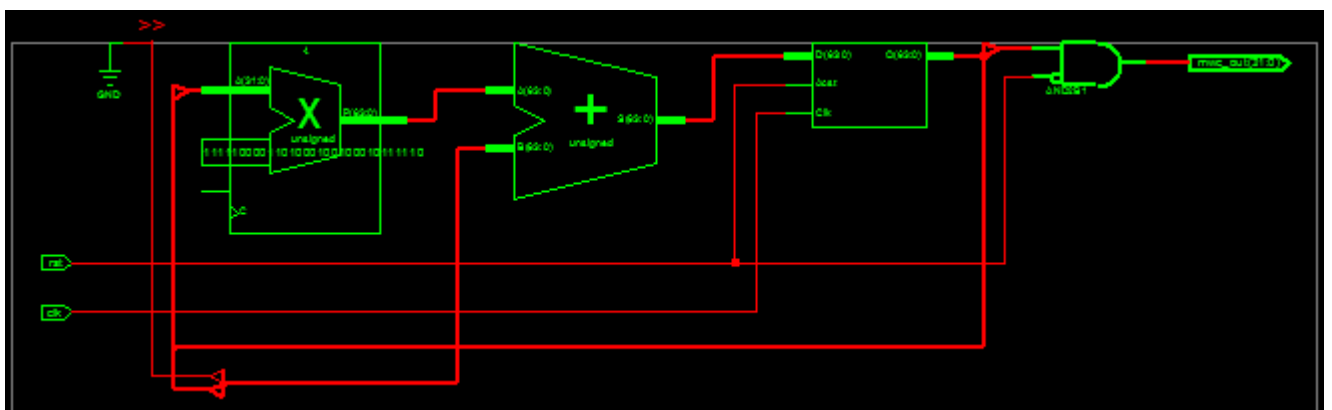
Dobiće se top-level dijagram MWC generatora. Vide se ulazi clk i rst i 32-bitni izlazni bus mwc_out:



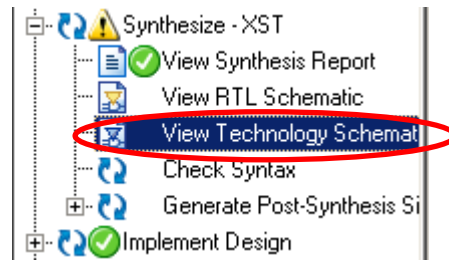
Da bi se u dijagramu spustili jedan nivo ispod, treba kliknuti na ikonu Push:



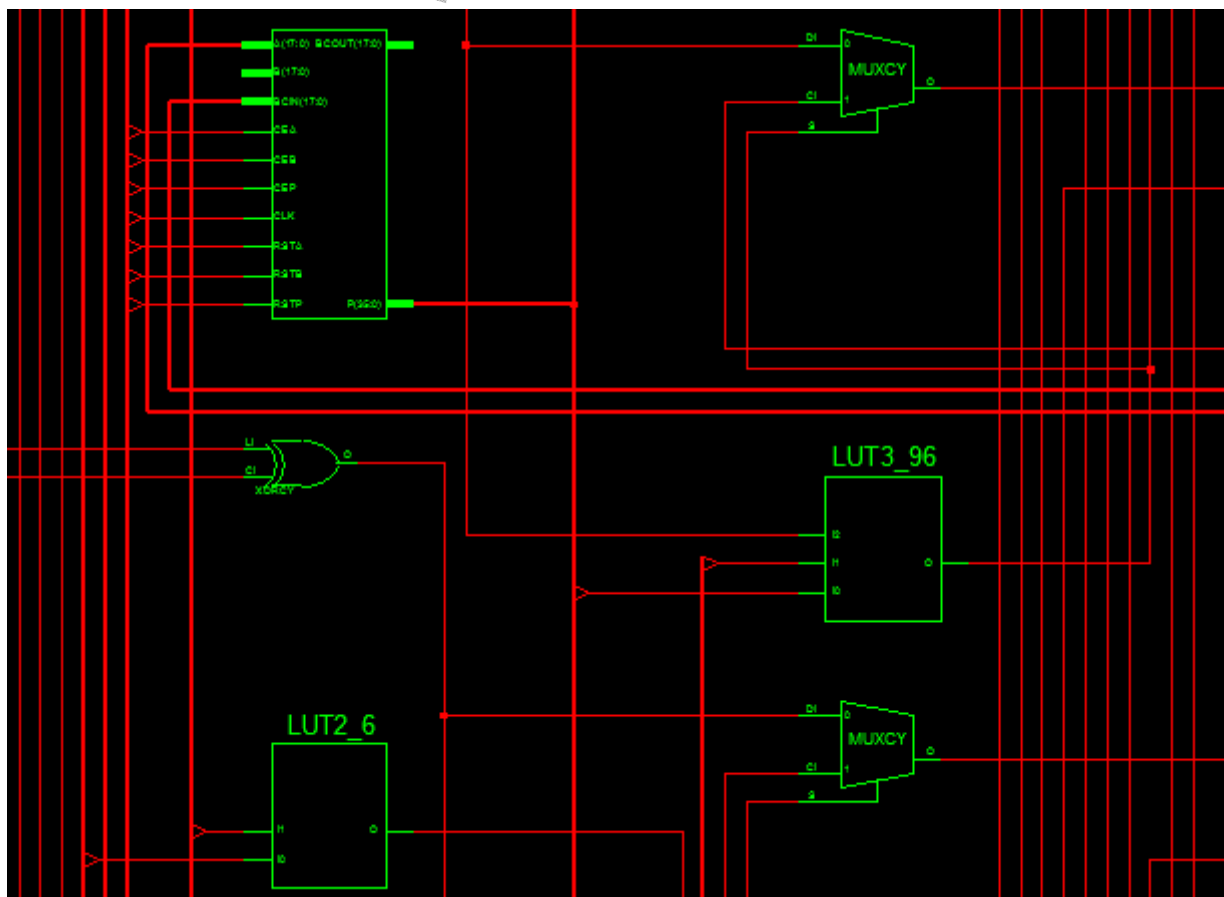
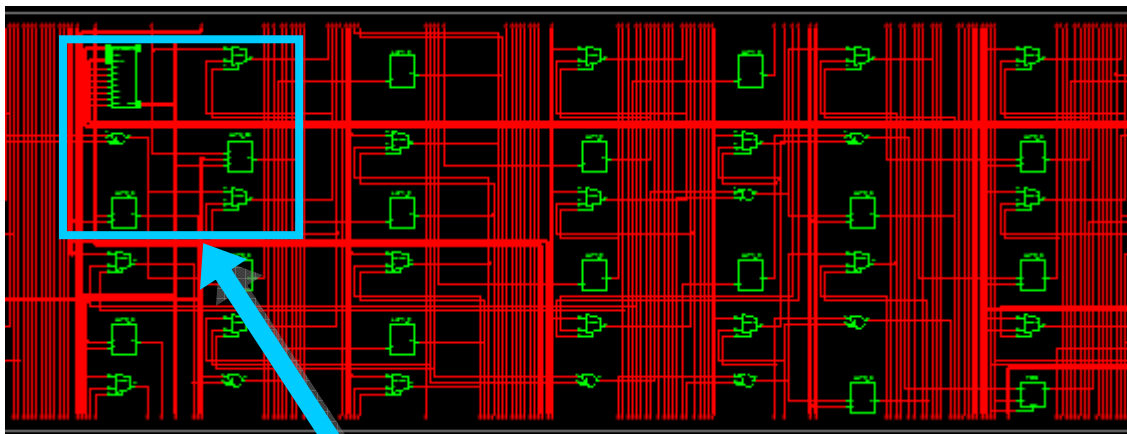
Vide se 32x32-bitni množač neoznačenih brojeva, 32-bitni potpuni sabirač neoznačenih brojeva i 64-bitni shift registar:



Za tehnološku šemu treba kliknuti na View Technology Schematic:



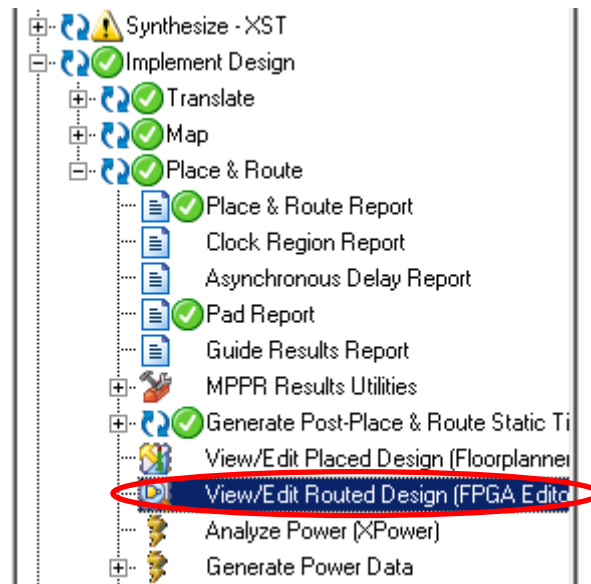
Dobija se šema koju je softver generisao za konkretno FPGA kolo i njegove resurse. Ovde je dat samo deo jedne od 11 stranica koliko sadrži šema. Vide se upotrebljeni LUT blokovi, množači, multiplekseri, gejtovi.




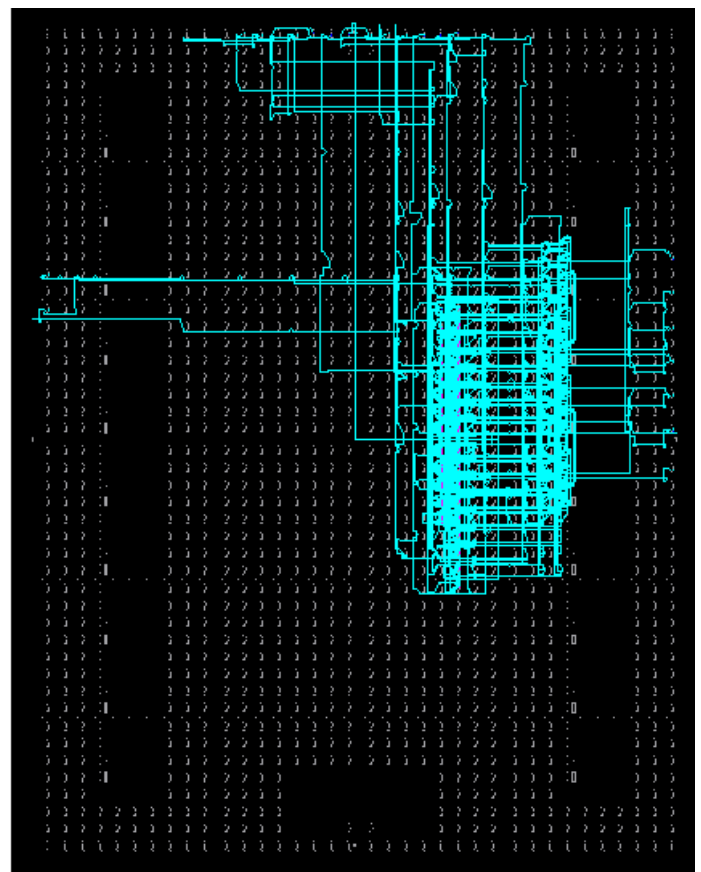
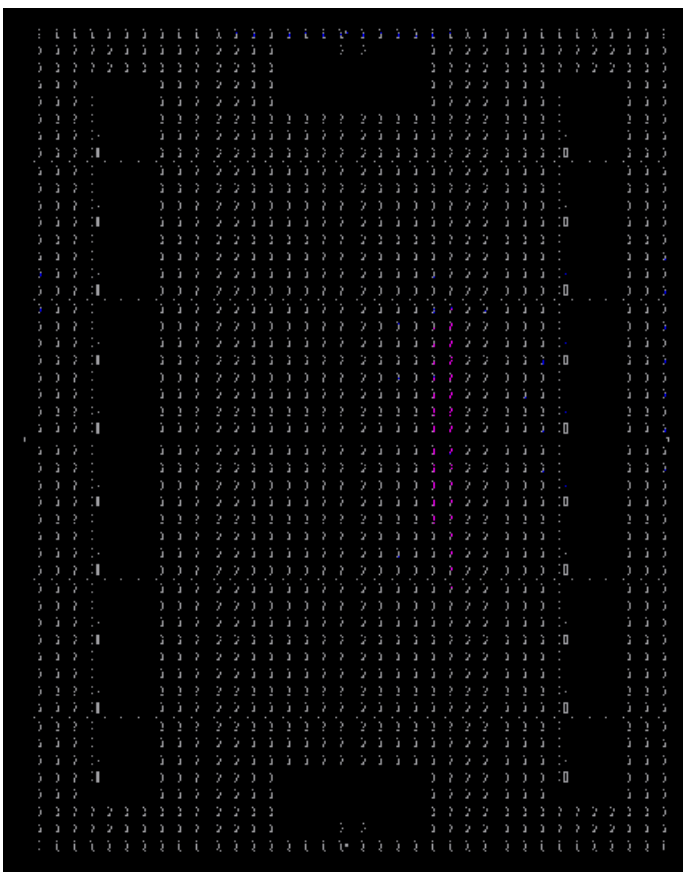
2.3.8 Pregled rasporeda i veza

Za grafički prikaz razmeštanja i povezivanja komponenti i blokova u FPGA kolu i eventualno editovanje, ukoliko softver nije mogao sam to obaviti, koristi se FPGA Editor.

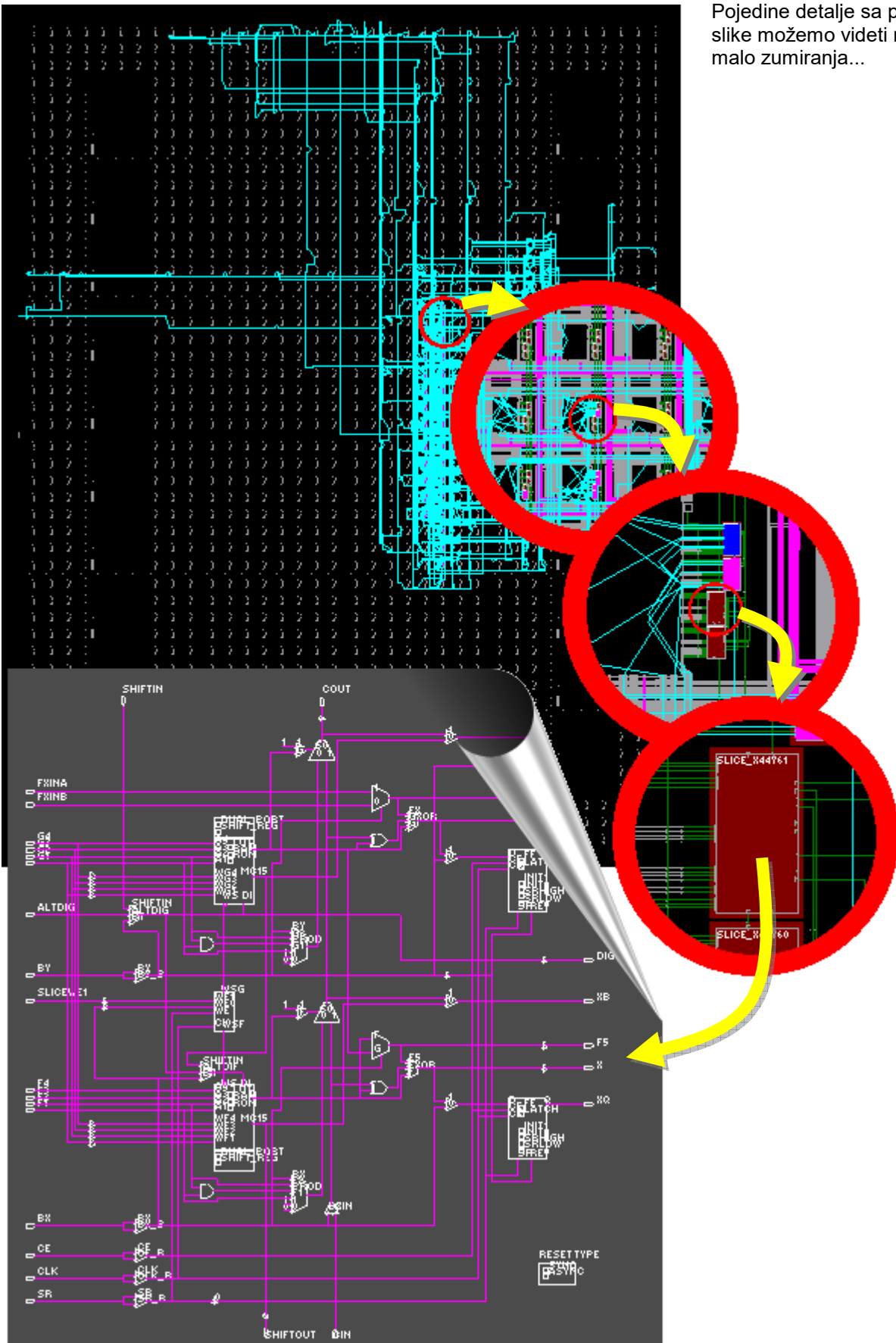
Da bi pokrenuli FPGA Editor, treba kliknuti na + ispred Implement Design, zatim na + ispred Place & Route i onda dva puta kliknuti na View/Edit Routed Design (FPGA Editor):



Ovim je pokrenut FPGA Editor i možemo videti kako je softver planirao razmeštaj komponenti i veza u izabranom FPGA kolu (da vidimo veze potrebno je kliknuti na ikonu Routes  u meniju):



Pojedine detalje sa prethodne slike možemo videti nakon malo zumiranja...



Programiranje FPGA se obavlja pomoću softverskog alata iMPACT iz paketa Xilinx ISE. Potreban je fajl za programiranje, PC, JTAG kabal i sistem sa ugradjenim FPGA kolom, i ceo proces traje par sekundi. Medjutim, pošto je naša ciljna tačka projekta već ostvarena tokom implementacije (generisanjem fajla za programiranje izabranog FPGA kola), ovde se projektovanje 32-bitnog MWC RNG završava.