PROJEKTOVANJE VLSI SISTEMA

User-manual

IMPLEMENTACIJA TRAFFIC LIGHTS KONTROLERA NA PYNQ-Z2 PLOČI

Ivan Miljković 100/2018

Student treće godine Softverskog Inženjerstva

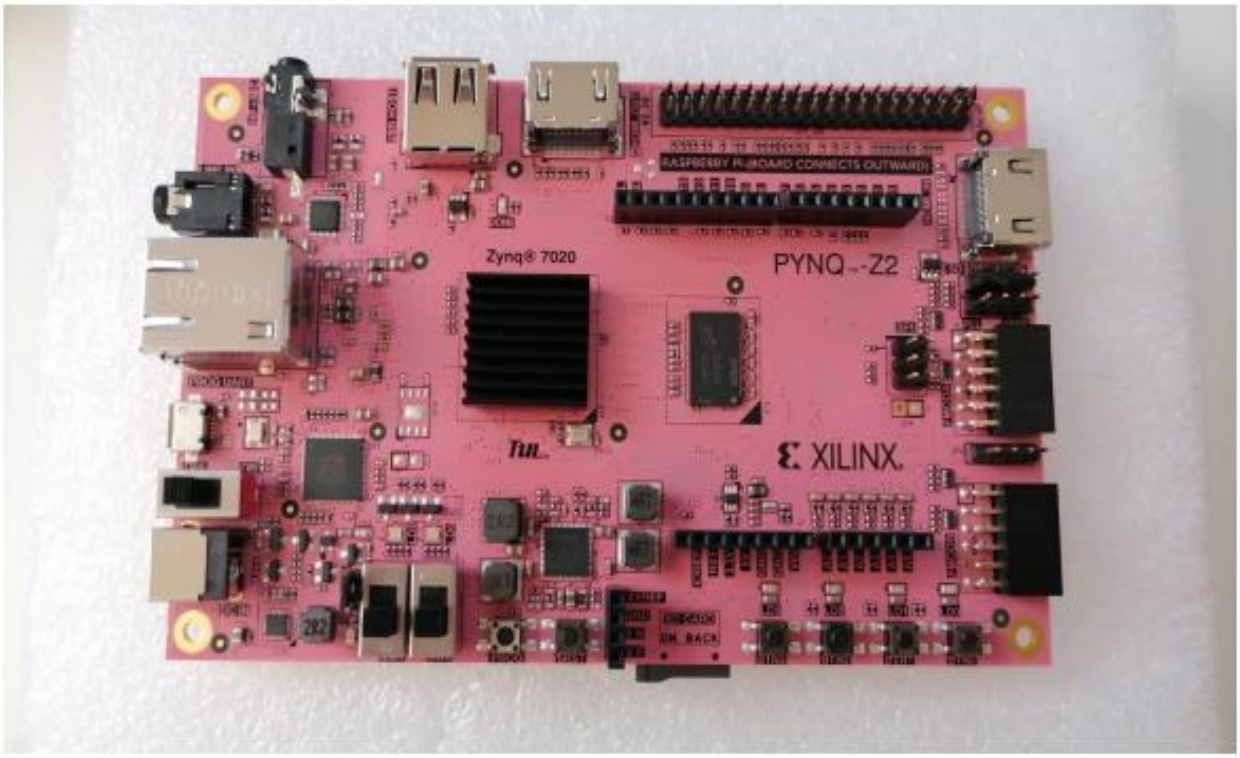
Prirodno-Matematičkog Fakulteta

Uvod

Sistemi za kontrolu semafora se široko koriste za praćenje i kontrolu protoka automobile, kroz raskrsnicu mnogih puteva i omogućuje bezbedniji prelazak pešaka.

Kodiranje je urađeno korišćenjem Verilog hardverskog opisnog jezika u programu Vivado.

Projekat je rađen na PYNQ-Z2 ploči.



Verilog kod

```
timescale 1ns / 1ps

module Traffic_Light(input clk,rst, output reg [2:0]light_M1, output reg [2:0]light_S, output reg [2:0]light_MT, output reg [2:0]light_M2);

    parameter    S1=0, S2=1, S3 =2, S4=3, S5=4,S6=5;

    reg [3:0]count;

    reg[2:0] ps;

    parameter    sec7=7,sec5=5,sec2=2,sec3=3;

    always@(posedge clk or posedge rst )

        begin

        if(rst==1)

        begin

        ps<=S1;
        count<=0;
        light_M1<=~light_M1;
        light_M2<=~light_M2;
        light_MT<=~light_MT;
        light_S<=~light_S;

        end

        else

            case(ps)

                S1: if(count<sec7)

                        begin

                        ps<=S1;
                        count<=count+1;

                        end

                    else

                        begin

                        ps<=S2;
                        count<=0;

                        end

                S2: if(count<sec2)

                        begin

                        ps<=S2;
                        count<=count+1;

                        end

                    else
```

```
            begin

            ps<=S3;
            count<=0;

            end

S3: if(count<sec5)

            begin

            ps<=S3;
            count<=count+1;

            end

      else

            begin

            ps<=S4;
            count<=0;

            end

S4:if(count<sec2)

            begin

            ps<=S4;
            count<=count+1;

            end

else

            begin

            ps<=S5;
            count<=0;

            end

S5:if(count<sec3)

            begin

            ps<=S5;
            count<=count+1;

            end

      else

            begin

            ps<=S6;
            count<=0;

            end

S6:if(count<sec2)

            begin

            ps<=S6;
            count<=count+1;
```

```
                    end

                else

                    begin

                    ps<=S1;
                    count<=0;

                    end

            default: ps<=S1l

            endcase

        end

/*      // za testiranje simulacije

        always@(ps)

        begin

            case(ps)

                S1:

                begin

                    light_M1<=3'b001;
                    light_M2<=3'b001;
                    light_MT<=3'b100;
                    light_S<=3'b100;

                end

                S2:

                begin

                    light_M1<=3'b001;
                    light_M2<=3'b010;
                    light_MT<=3'b100;
                    light_S<=3'b100;

                end

                S3:

                begin

                    light_M1<=3'b001;
                    light_M2<=3'b100;
                    light_MT<=3'b001;
                    light_S<=3'b100;

                end

                S4:

                begin

                    light_M1<=3'b010;
                    light_M2<=3'b100;
                    light_MT<=3'b010;
                    light_S<=3'b100;

                end
```

```
                S5:

                begin

                    light_M1<=3'b100;
                    light_M2<=3'b100;
                    light_MT<=3'b100;
                    light_S<=3'b001;

                end

                S6:

                begin

                    light_M1<=3'b100;
                    light_M2<=3'b100;
                    light_MT<=3'b100;
                    light_S<=3'b100;

                end

                default:

                begin

                    light_M1<=3'b000;
                    light_M2<=3'b000;
                    light_MT<=3'b000;
                    light_S<=3'b000;

                end

                endcase

        end      */
```

Objašnjenje koda

Na početku smo definisali dve ulazne promenljive, clock(clk) i reset(rst), koje predstavljaju dugme za početak rada kontrolera, i dvanaest registarskih izlaznih promenljivi [2:0]light_M1, [2:0]light_S, [2:0]light_MT, [2:0]light_M2, koje predstavljaju led diode koje svetle na ploči. Prave se promenljive tipa parameter gde S1 ima vrednost 0, S2 ima vrednost 1, S3 ima vrednost 2, S4 ima vrednost 3, S5 ima vrednost 4 i S6 ima vrednost 5, sec7 koji ima vrednost 7, sec5 koji ima vrednost 5, sec2 koji ima vrednost 2 i sec3 koji ima vrednost 3 i još dva registra [3:0]count i [2:0] ps.
Na svaki pozitivni deo takta, postavlja se pitanje da li je pritisnuto dugme rst. U slučaju da jeste, na svaki takt se ispisuje ps vrednost S1, count dobija vrednost 0 I na output se šalje suprotno od onoga što je trenutno na output-u light_M1, light_M2, light_MT i light_S. U suprotnom counter se inkrementira na svaki parameter S(od 1 do 6). Što predstavlja kružno kretanje Traffic Lights, odnosno na svaki klik dugmeta menjaju se boje na led diodama.

Ciljevi

- Definisanje i analiza principa rada sistema kao i potrebnih komponenti.
- Projektovanje i razvoj sistema kao i testiranje njegove funkcionalnosti.
- Finalizacija celokupnog projekta i izrada prateće dokumentacije.

Definisanje i analiza principa rada sistema kao i potrebnih

Sa sajta fakulteta https://imi.pmf.kg.ac.rs/moodle/course/view.php?id=555 prikupljena je dokumentacija sa svim uputstvima:

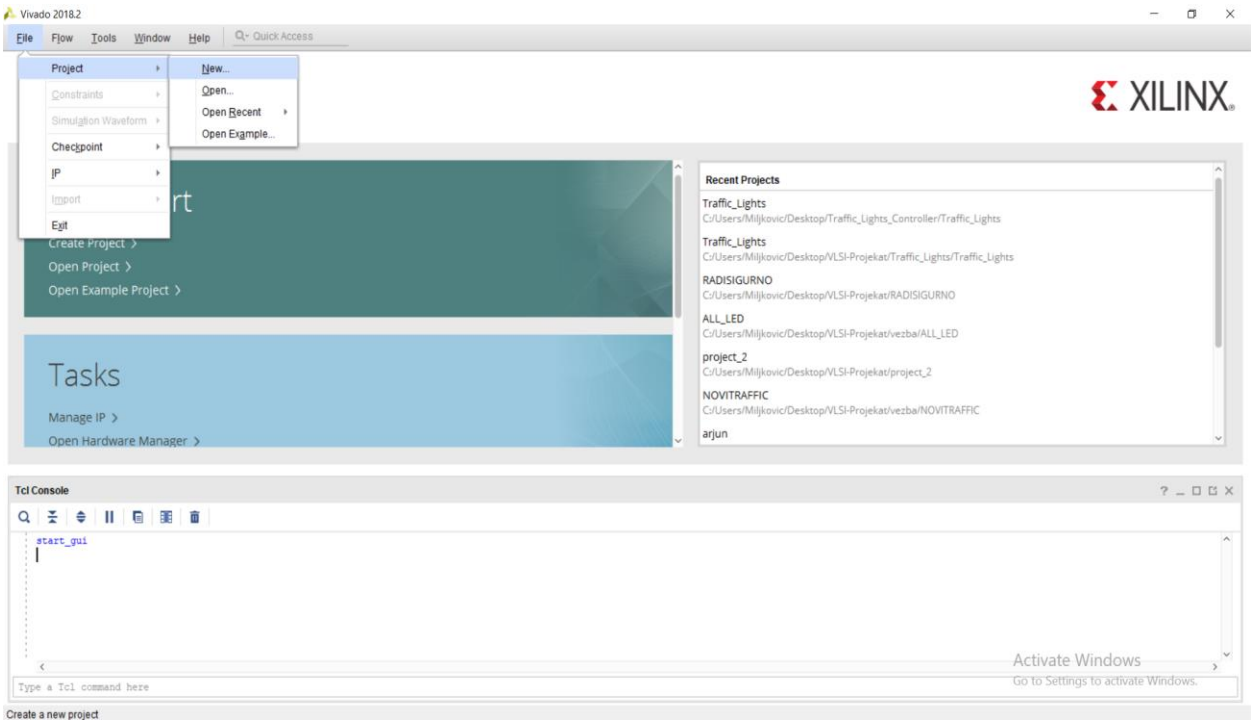
- Kako instalirati Vivado program i kako započeti projekat
- Odabir teme
- Razni sajtovi za pomoć korišćenja PYNQ-Z2 ploče i pronalaženja koda za odabranu temu
- Kako instalirati Gantogram i kako koristiti
- Kako napraviti klonirani repozitorijum sa Githuba-a i kako uploudovati
- Koja je sve dokumentacija potrebna za predaju projekta
- Koji je rok predaje

Od profesora Aleksandra Peulića je dobijena PYNQ-Z2 ploča na kojoj se radi projekat. Odabran je tema **Implementacija algoritama veštačke inteligencije na fpga (Traffic Lights).**

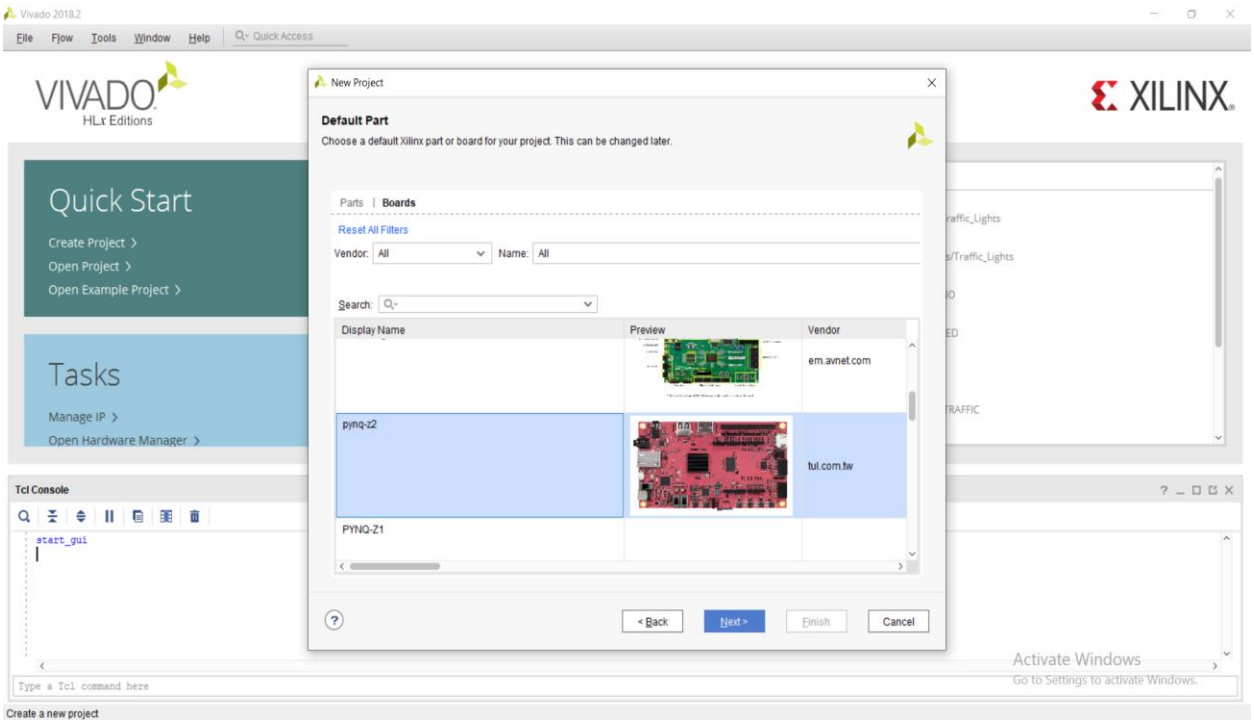
Projektovanje i razvoj sistema kao i testiranje njegove funkcionalnosti.

Nakon prikupljenih uputstava, opreme i odabira teme započinje se rad.

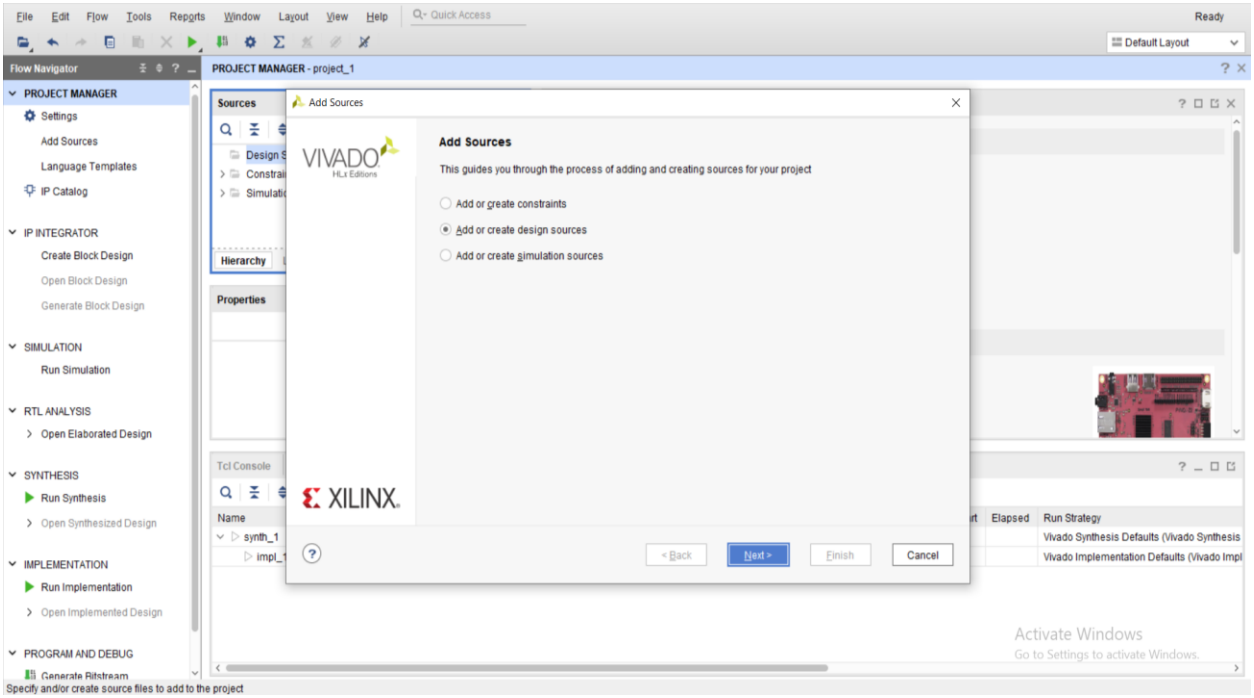
Praćenjem uputstva za instalaciju Vivada instaliran je program. Kada otvorimo program prvo moramo da napravimo novi projekat.

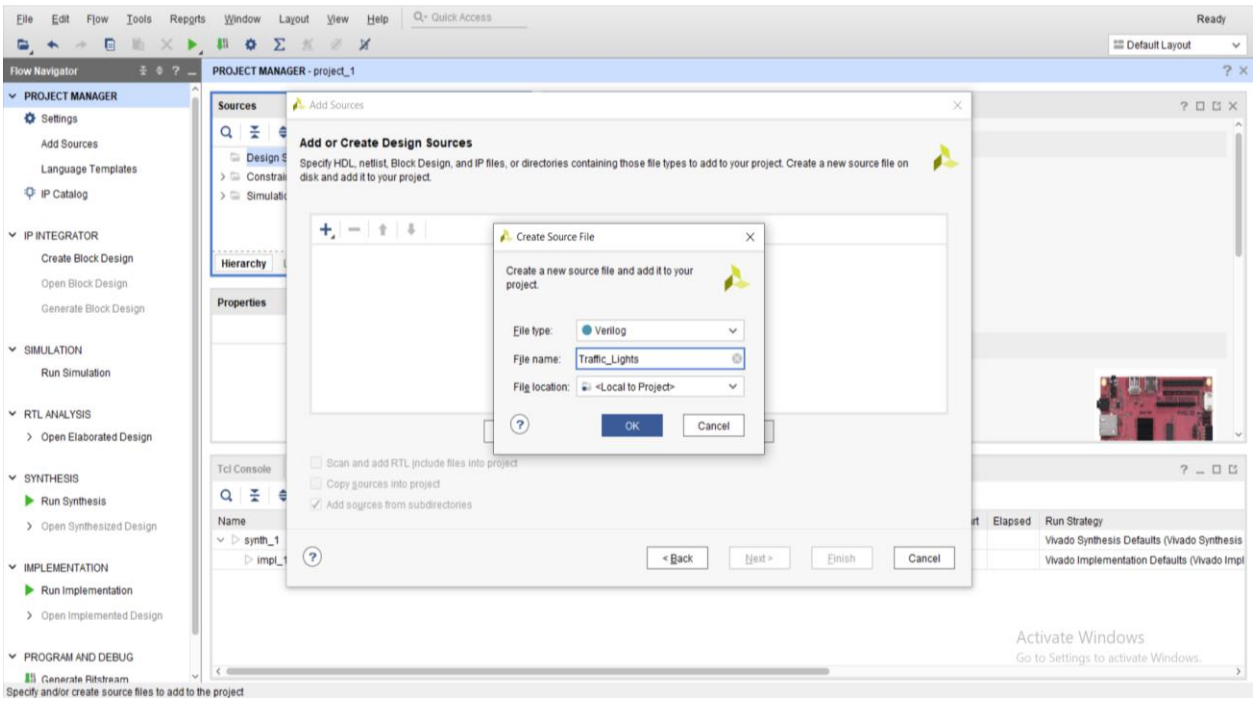


Prilikom kreiranja projekta najvažnije je odabir ploče. Kako bi Vivado prepoznao PYNQ-Z2 ploču moramo otpakovati zip file koji smo dobili sa sajta fakulteta i smestiti folder namenjen za PYNQ-Z2 ploču u folder „boards“ koji se nalazi u direktorijumu instaliranog Vivada.

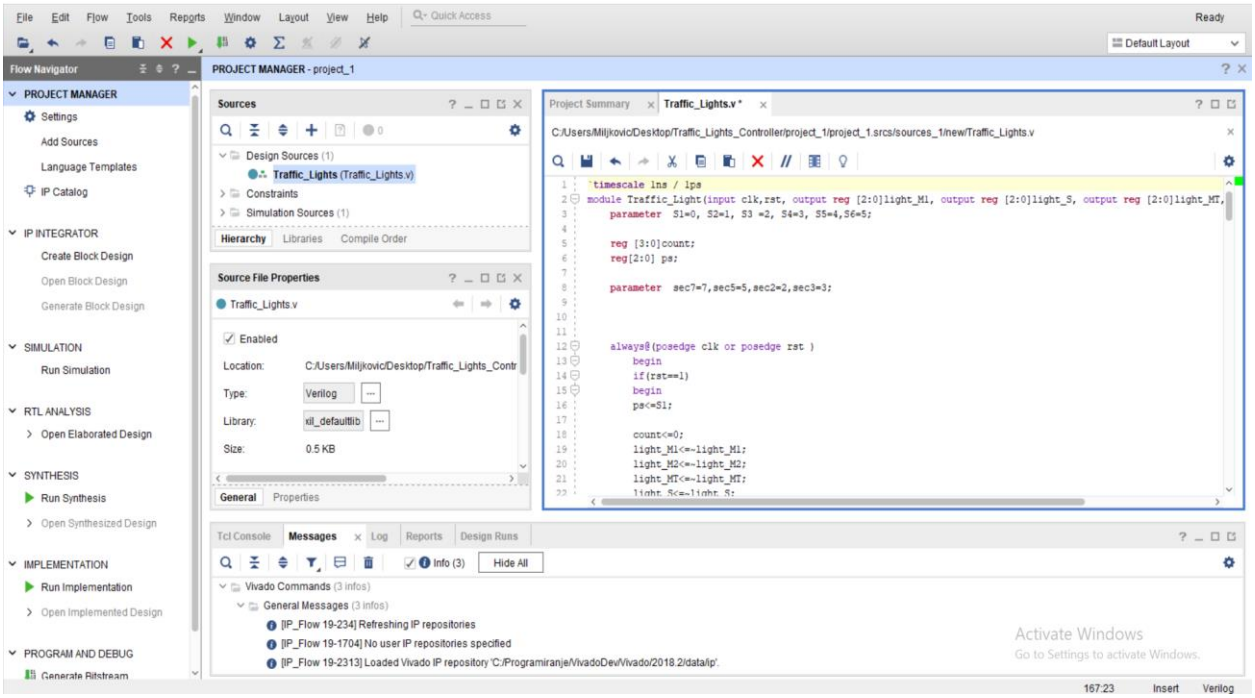


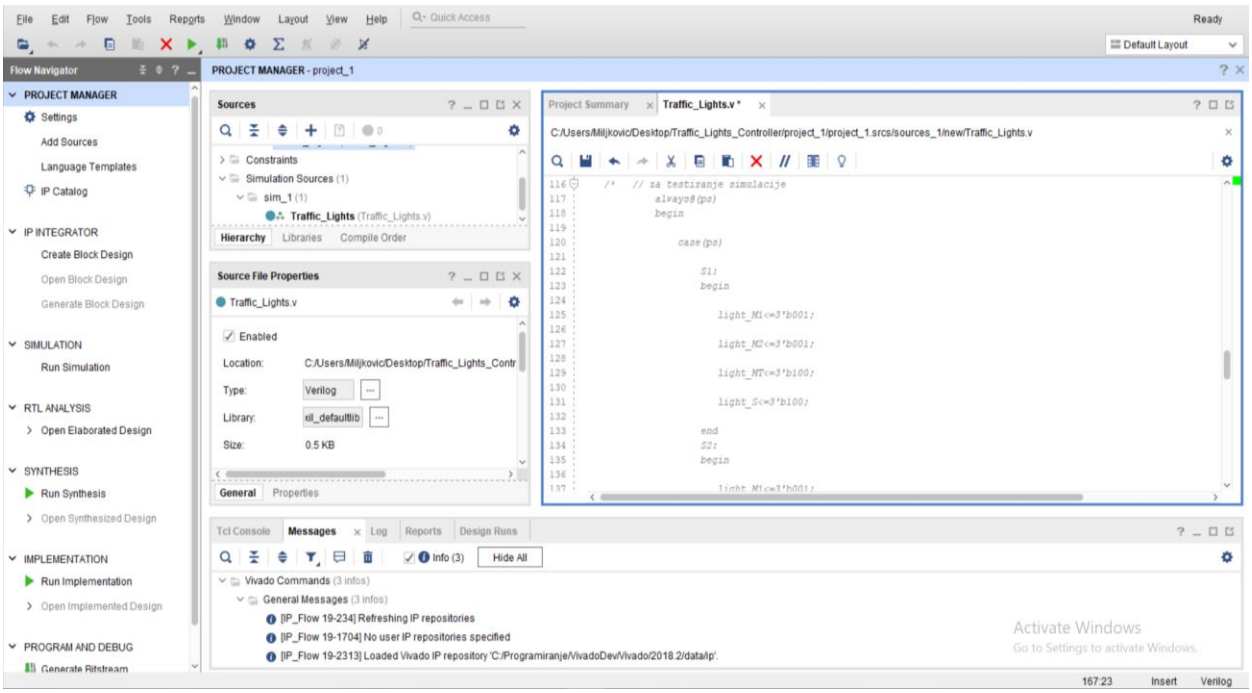
Kada je napravljen projekat vidimo u sources „Designe Sources“, „Constraints“ i „Simulation Sources“. U designe sources pravimo nov file u koji ubacujemo naš Verilog kod.



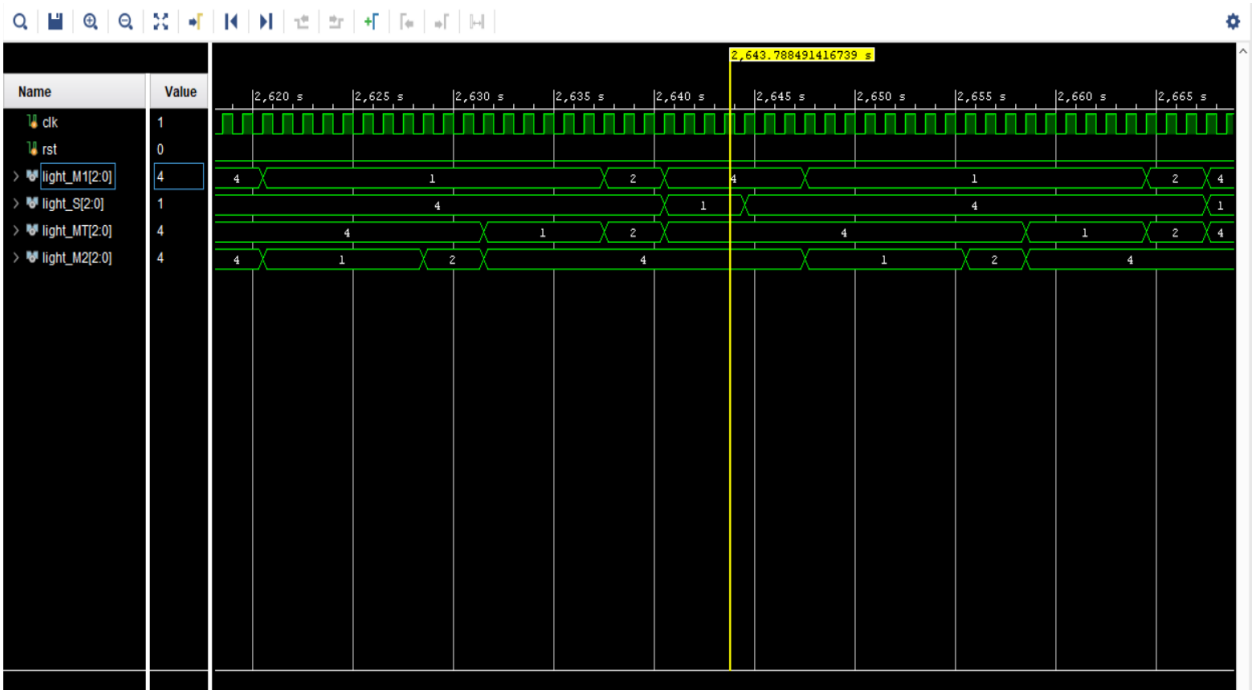


U kodu imamo isprogramiran rad kontrolera za Traffic Lights i testni kod koji se koristi za simulaciju.

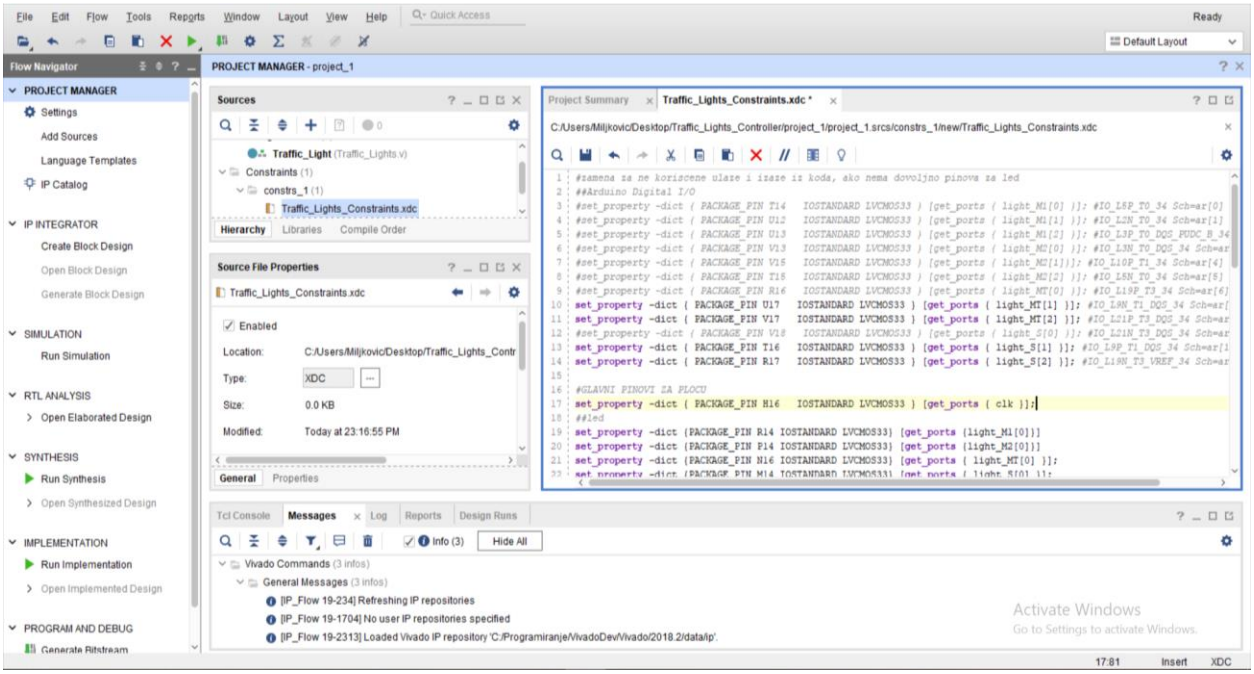



File Edit Flow Tools Reports Window Layout View Help
Ready
Default Layout
Flow Navigator
PROJECT MANAGER - project_1
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Run Simulation
RTL ANALYSIS
Open Elaborated Design
SYNTHESIS
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Run Implementation
Open Implemented Design
PROGRAM AND DEBUG
Generate Bitstream
Sources
Constraints
Simulation Sources (1)
sim_1 (1)
Traffic_Lights (Traffic_Lights.v)
Hierarchy Libraries Compile Order
Source File Properties
Traffic_Lights.v
Enabled
Location: C:/Users/Miljkovic/Desktop/Traffic_Lights_Contr
Type: Verilog
Library: xil_defaultlib
Size: 0.5 KB
General Properties
Project Summary
Traffic_Lights.v *
C:/Users/Miljkovic/Desktop/Traffic_Lights_Controller/project_1/project_1.srcs/sources_1/new/Traffic_Lights.v
/* // sa testiranje simulacije
always@(ps)
begin
case(ps)
S1:
begin
light_M1<=3'b001;
light_M2<=3'b001;
light_MT<=3'b100;
light_S<=3'b100;
end
S2:
begin
Tcl Console Messages Log Reports Design Runs
Info (3) Hide All
Vivado Commands (3 infos)
General Messages (3 infos)
[IP_Flow 19-234] Refreshing IP repositories
[IP_Flow 19-1704] No user IP repositories specified
[IP_Flow 19-2313] Loaded Vivado IP repository 'C:/Programiranje/VivadoDev/Vivado/2018.2/data/ip'.
Activate Windows
Go to Settings to activate Windows.
167:23 Insert Verilog

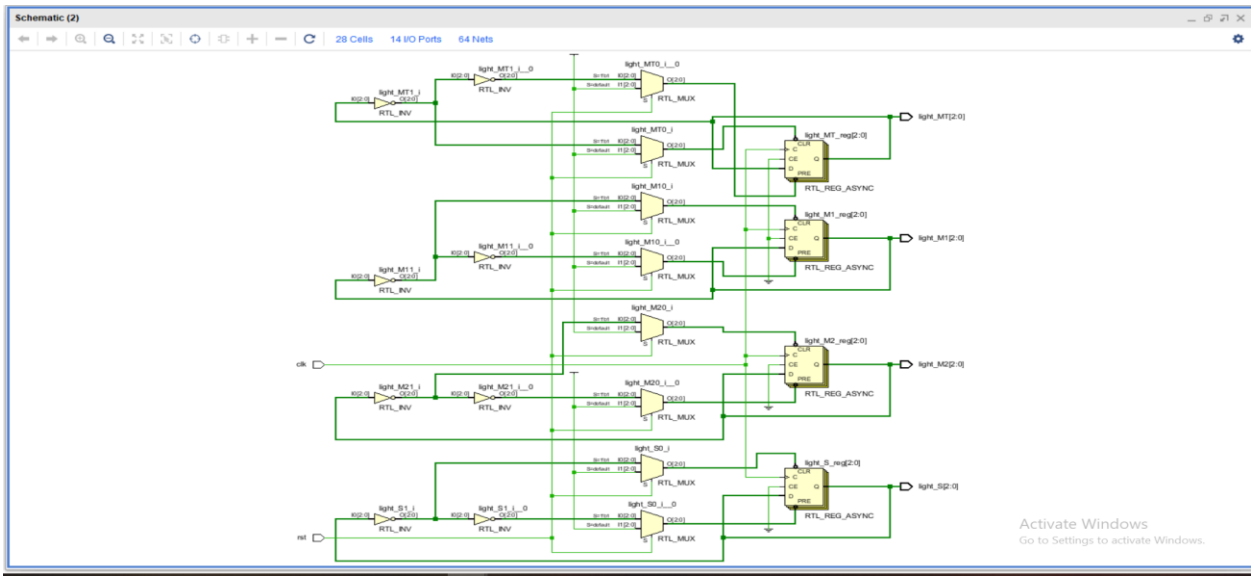
Simulacija


2,643.788491416739 s
Name
Value
clk
1
rst
0
light_M1[2:0]
4
light_S[2:0]
1
light_MT[2:0]
4
light_M2[2:0]
4
2,620 s
2,625 s
2,630 s
2,635 s
2,640 s
2,645 s
2,650 s
2,655 s
2,660 s
2,665 s

Nakon što je simulacija prošla i kod radi. Povezujemo ploču sa računarom, koristeći USB kabal. Kada uključimo ploču upali se crvena led dioda, što indikuje da je ploča uspešno povezana. Da bi naš kod bio implementirali na ploču u „Constraints“ ubacujemo constraints kodove, koje smo dobili sa sajta fakulteta i koristimo ih kako bi implementirali naš kod na ploču. Koristimo led, ledRGB, button i clock constraints za naš kod. Pošto imamo previše outputa, ne možemo svaki slučaj koda prikazati na ploči. Zato ćemo dodeliti bilo koji drugi constraint kod, jer program zahteva da svaki input i output ima svoj constraints kod.



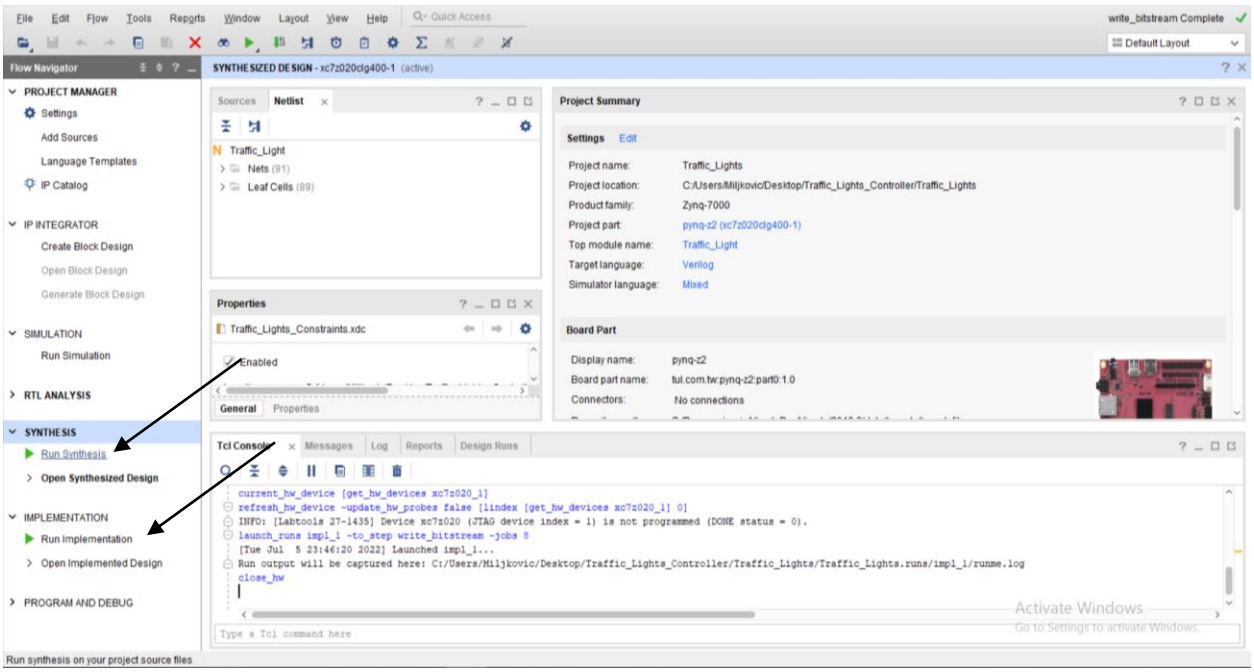
Možemo otvoriti RTL schematic i učitati RLT netlistu za interaktivnu analizu. Možemo proveriti RTL strukturu, sintaksu i definicije logike.



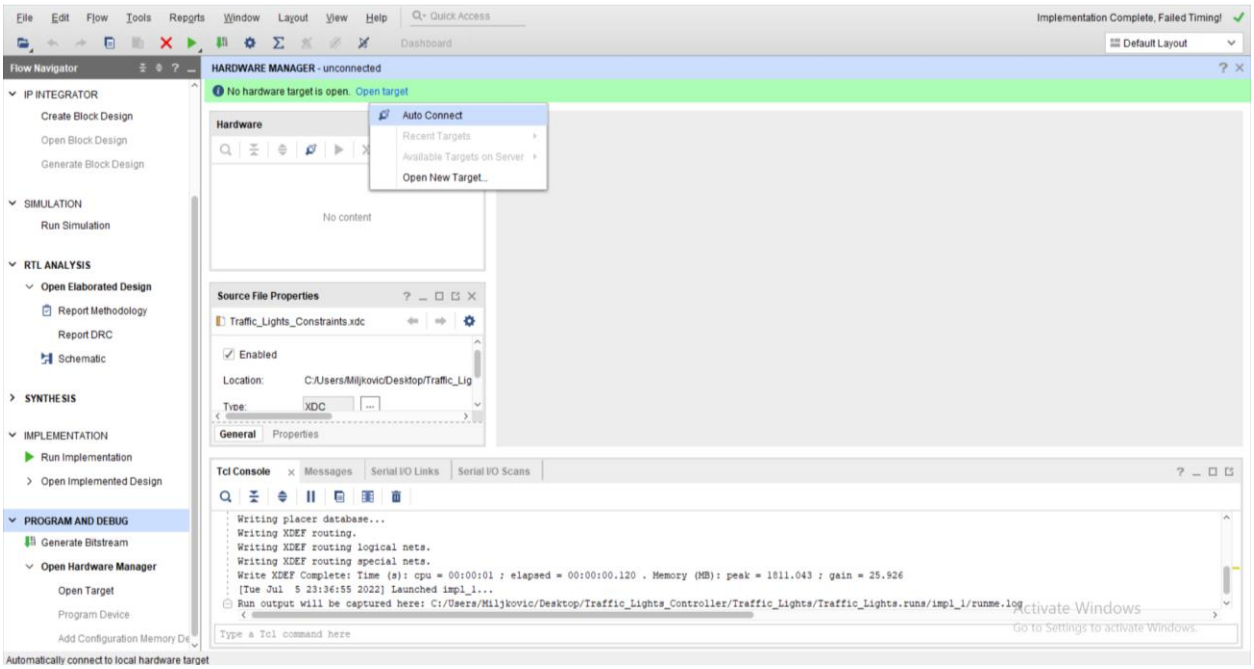
Sada možemo da pokrenemo „Synthesis“ i „Implementation“.

Alat za sintezu je računarski program, koji uzima instrukcije u obliku jezika za opis hardvera kao ulaz i generiše sintetizovanu listu mreža kao izlaz.

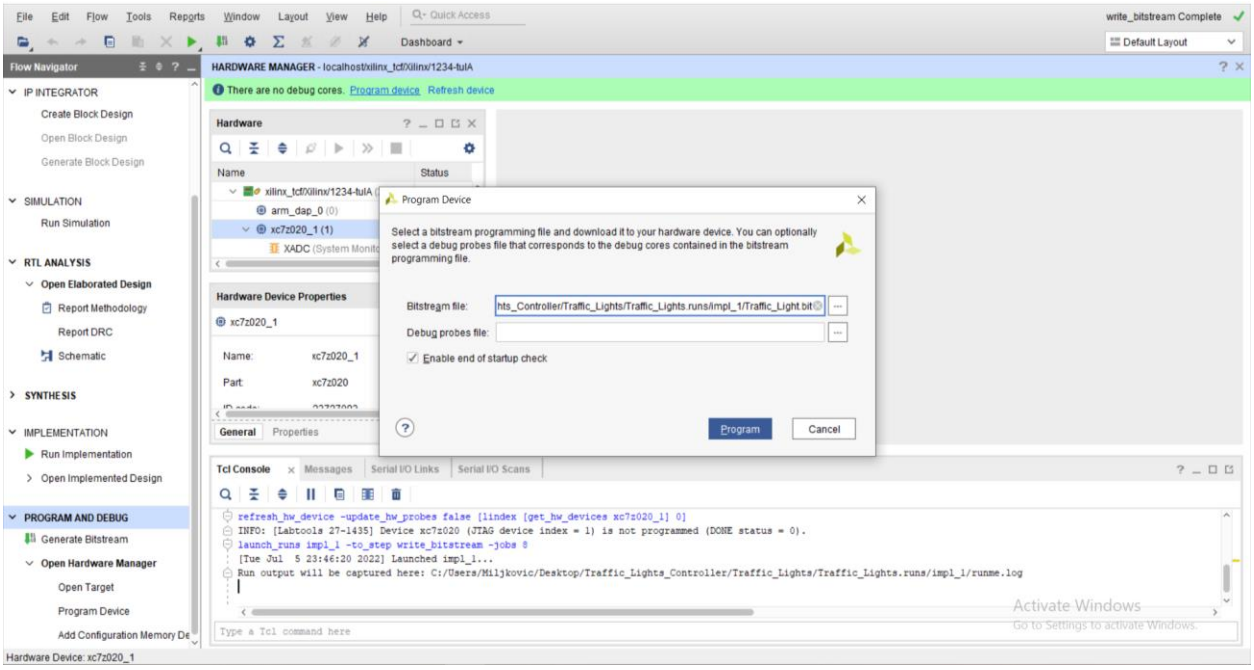
Alatka za implementaciju služi za postavljanje i rutiranje za Xilinx uređaje, generisanje tokova bitova i slika uređaja iz sintetizovane liste mreža.



Ako nema grešaka sinteza i implementacija je pokrenuta. Onda možemo da pokrenemo „Bitstream“, pre pokretanja prvo moramo da prepoznamo ploču pomoću „Open Hardware Manager“ i kliknemo na opciju „Open Target“. Ploča se automatski prepoznaje, ako je adekvatno povezana.



Sada može da se pokrene „Bitstream“. Bitstream je binarna sekvenca, koja se sastoji od niza bitova. Oni se koriste u FPGA aplikacijama za svrhe programiranja i za uspostavljanje komunikacionih kanala. Nakon završetka, program kreira bitstream datoteku koja sadrži programske podatke povezane sa našim FPGA čipom.



Ako smo sve povezali i nema grešaka u programu. Bitstream datoteka je uspešno pokrenuta. Sada je implementiran naš kod na ploči i možemo fizički da proverimo.

Finalizacija celokupnog projekta i izrada prateće dokumentacije

Nakon što je projekat završen, sve nepotrebne linije koda brišemo i u programu svaki deo koda je pod komentarom, gde je napomenuto čemu služi i za šta je namenjen kod.
Zatim napravimo Project_Charter. Gde pišemo:

- Naziv projekta
- Namenu projekta
- Ciljeve projekta
- Stejkholdere
- Ključne događaje
- Pretpostavke, ograničenja i rizike
- Kontakt osobu

Takođe instaliramo program „GanttProject“ gde pravimo gantogram sa svim ciljevima i zadacima projekta. Za svaki zadatak je određen datum, kad ga treba započeti i završiti, sa kratkim objašnjenjem šta je rađeno.

Projekat zajedno sa dokumentacijom čuvamo u klonirani repozitorijum Github-a.