

Elektronski fakultet Niš

Seminarski rad iz predmeta
PROJEKTOVANJE ELEKTRONSKIH KOLA

Tema rada:
**TROBITNI KOMPARATOR,
REALIZACIJA ZVUČNE I VERTIKALNE SIGNALIZACIJE
JEDNOG PRUŽNOG PRELAZA**

Mentor:
Prof. Dr Predrag Petković

Student:
Aleksa Randelović
broj indeksa 17378

Maj, 2020.

Sadržaj

Uvod u VHDL	3
VHDL kod	4
LIBRARY	5
ENTITY	5
ARCHITECTURE.....	5
Komparator.....	6
Trobitni komparator	7
Realizacija kola na FPGA čipu Cyclone II korišćenjem programa Quartus i Altera DE1 razvojne ploče	14
Arduino.....	19
Fritzing	21
Projektni zadatak	22
Literatura	27

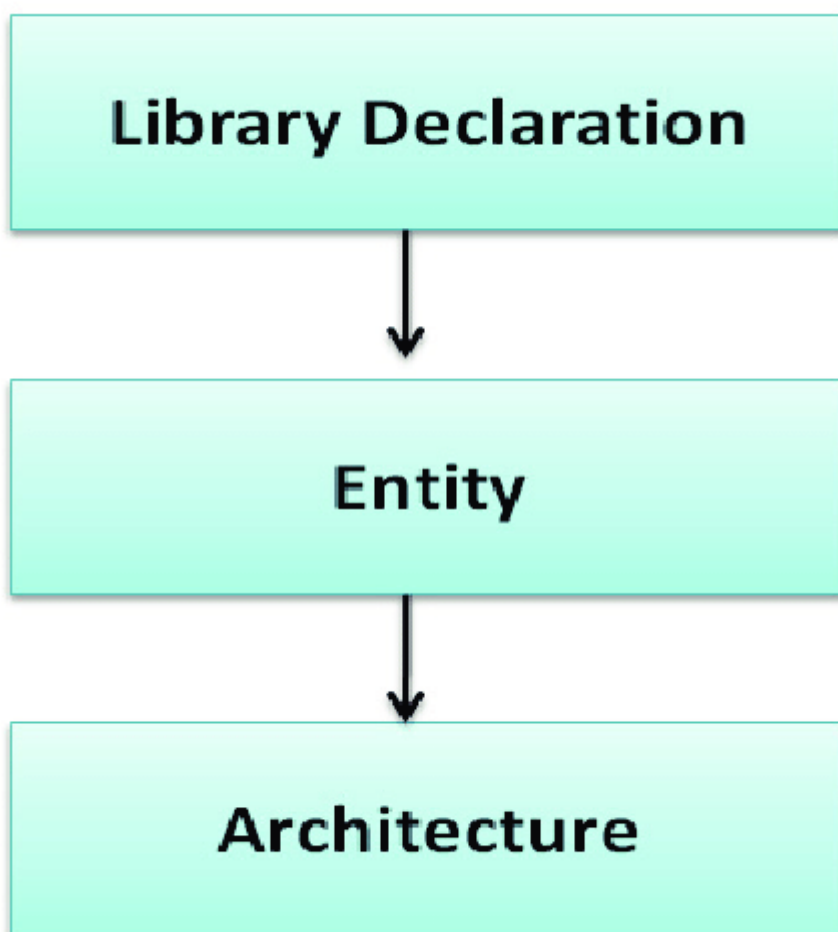
Uvod u VHDL

VHDL je skraćenica od “VHSIC Hardware Description Language“ (u prevodu, VHSIC jezik za opis hardvera) gde je VHSIC takođe skraćenica od “Very High Speed Integrated Circuits” (u prevodu, integrisana kola veoma velike brzine rada). Iako prevashodno namenjen za modeliranje hardvera, VHDL je po mnogim svojim karakteristikama sličan konvencionalnim, računarskim programskim jezicima, kao što je to npr: C. U VHDL-u, kao u C-u, postoje promenljive, programska grananja, petlje, funkcije itd. Međutim, između VHDL-a i programskih jezika postoje i značajne razlike, uslovljene njihovom specifičnom namenom. Program napisan na programskom jeziku se kompajlira (uz pomoć kompajlera) i tako prevodi u mašinski kod, koji se može direktno izvršiti na računaru opšte namene. S druge strane, VHDL kod je moguće sintetizovati, pomoću specijalizovanih softvera za hardversku sintezu, odnosno prevesti ga u oblik netliste na osnovu koje je moguće automatski fizički realizovati sistem. Takođe, VHDL opis nekog sistema se može funkcionalno i vremenski simulirati, uz pomoć HDL simulatora, a u cilju verifikacije projekta pre njegove fizičke realizacije

VHDL kod

Svaki kompletan VHDL opis sastoji se iz sledeće tri sekcije: Deklaracija LIBRARY: sadrži spisak biblioteka i/ili delova biblioteka koji se koriste u projektu, npr: ieee, std, work itd.

ENTITY: definiše ulazne i izlazne signale (pinove ili portove) kola.
ARCHITECTURE: sadrži VHDL kod koji opisuje ponašanje (tj. funkciju) ili unutrašnju organizaciju (tj. strukturu) kola.



Slika 1 : Struktura VHDL opisa

LIBRARY

Library je kolekcija često korišćenih delova VHDL koda. Biblioteka se sastoji iz jednog ili više paketa (package) koji sadrže funkcije (functions), procedure (procedures), komponente (components), konstante (constants) i definicije tipova podataka (types). Jednom kreirana, biblioteka se može koristiti u drugim projektima ili razmenjivati s drugim projektantima.

```
LIBRARY ime_biblioteke;
```

```
USE ime_biblioteke.ime_paketa.delovi_paketa;
```

ENTITY

Sekcija entity definiše ime kola koje se projektuje i imena i osnovne karakteristike njegovih ulaza i izlaza, tzv. pinova, odnosno portova (ports). Sintaksa deklaracije entiteta je sledećeg oblika:

```
ENTITY ime_entiteta IS  
PORT (  
ime_porta : smer_signala tip_signala;  
ime_porta : smer_signala tip_signala;  
... );  
END ime_entiteta;
```

ARCHITECTURE

Da bi VHDL opis nekog kola bio potpun, osim entiteta, koji definiše ulaze i izlaze kola, neophodno je napisati još jednu programsku celinu koja se zove arhitektura (architecture), a koja će sadržati opis funkcionisanja (ponašanja) ili opis unutrašnje strukture kola. Njena sintaksa je sledeća:

```
ARCHITECTURE           ime_arhitekture           OF           ime_entiteta           IS  
[deklaracije]  
BEGIN  
[kod]  
END ime_arhitekture
```

Komparator

Poređenje binarnih brojeva je operacija koja se često koristi u digitalnim sistemima. Kao što znamo, tri osnovna operatora poređenja su:

veće - $G(X > Y)$,

jednako - $E(X = Y)$ i

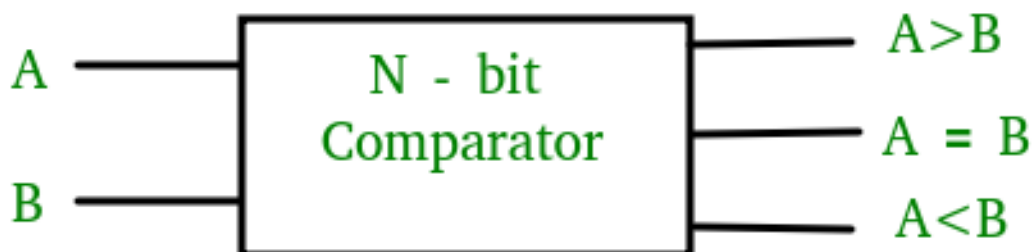
manje - $L(X < Y)$.

Takođe, za svaki od ovih operatora postoji komplementarni operator. Operator “veće ili jednako” ($X \geq Y$) je komplement od “manje”, “manje ili jednako” ($X \leq Y$) je komplement od “veće”, dok je “različito” ($X \neq Y$) komplement od “jednako”.

Rezultat bilo kog od navedenih operatora je logička promenljiva koja može imati vrednost 0 (netačno) ili 1 (tačno). Za digitalnu realizaciju operatora poređenja koriste se kombinaciona kola koja se zovu komparatori. U opštem slučaju, n-bitni univerzalni komparator poredi dva neoznačena binarna broja i generiše tri binarna rezultata G, E, i, L koji imaju sledeće značenje:

Ako je izlaz $G=1$, tada je $X > Y$; $E=1$ ukazuje na, $X=Y$, a $L=1$ na $X < Y$.

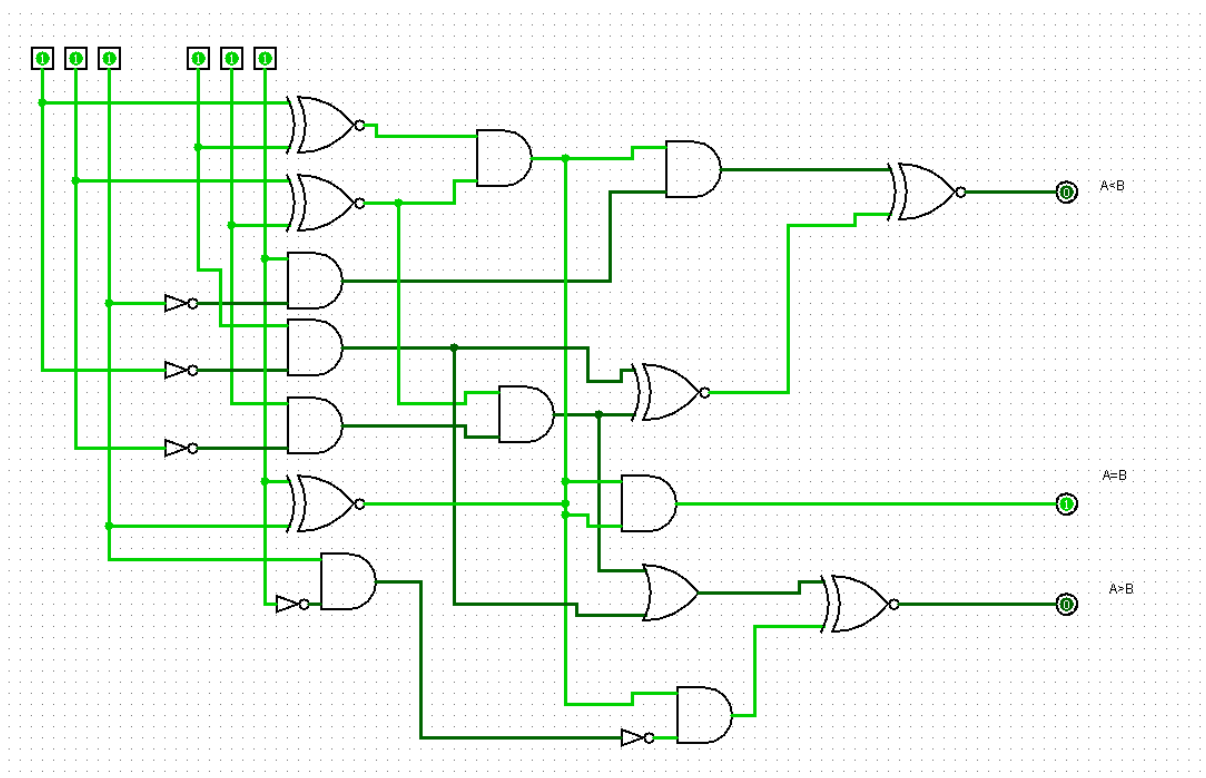
Kombinaciona mreža univerzalnog komparatora bi mogla da se sintetiše na osnovu tabele istinitosti u kojoj bi za svaku kombinaciju n-bitnih operanda X i Y bile navedene vrednosti izlaza G, E i L. Međutim, takva tabela bi bila previše velika, čak i za relativno malo n. Zato je bolji pristup da se logičke funkcije univerzalnog komparatora izvedu direktno, svođenjem poređenja više bitnih brojeva na poređenje njihovih bitova.



Slika 2 : Komparator u digitalnoj logici

Trobitni komparator

Trobitni komparator je kombinaciono kolo koje poredi dva neoznačena trobitna broja.



Slika 3 : Šema trobitnog komparatora u programu LOGISIM.

U VHDL-u trobitni komparator se može opisati ponašanjem i na nivou gejtova.

Pre projektovanja komparatora u VHDL-u prvo je potrebno nacrtati skicu s prikazom nekoliko glavnih delova kola i njihovih veza. Nakon toga, svaki blok je opisivan jednim segmentom VHDL koda. Nakon deklaracije komponente, ona može postati dostupna za korišćenje u drugim delovima VHDL projekta.

Deklaracija komponente sadrži informacije o interfejsu komponente, što podrazumeva ulazne i izlazne portove i druge relevantne parametre komponente. Informacije o komponenti sadržane u deklaraciji komponente su identične onim iz deklaracije entiteta.

```
66 library IEEE;
67 use IEEE.STD_LOGIC_1164.all;
68
69 entity ispit is
70 port(
71     a: in std_logic_vector (2 downto 0);
72     b: in std_logic_vector (2 downto 0);
73     g: out std_logic;
74     e: out std_logic;
75     l: out std_logic
76 );
77
78 end ispit;
79
80
81 architecture ispit of ispit is
82
83     COMPONENT inv IS
84         PORT(a: IN STD_LOGIC; b: OUT STD_LOGIC);
85     END COMPONENT;
86
87     COMPONENT and_kolo IS
88         PORT(a, b: IN STD_LOGIC; c: OUT STD_LOGIC);
89     END COMPONENT;
90
91     COMPONENT xnor_kolo IS
92         PORT(a,b: IN STD_LOGIC; c: OUT STD_LOGIC);
93     END COMPONENT;
94
95     COMPONENT or_kolo IS
96         PORT(a,b: IN STD_LOGIC; c: OUT STD_LOGIC);
97     END COMPONENT;
98
99
```

-- prvi broj koji se poredi
-- drugi broj koji se poredi
-- izlaz ukoliko je prvi broj veći od drugog
-- izlaz ukoliko su brojevi jednaki
-- izlaz ukoliko je prvi broj manji od drugog

-- deklaracija komponente NOT kola

-- deklaracija komponente AND kola

-- deklaracija komponente XNOR kola

-- deklaracija komponente OR kola

Slika 4 : Deklaracija komponenata potrebnih za projektovanje komparatora


```

1  -----
2  --
3  -- Title       : ispit
4  -- Design      : ispit
5  -- Author      : aleksa
6  -- Company     : Elektronski fakultet
7  --
8  -----
9  --
10 -- File        : c:\My_Designs\ispitni_projekat\ispit\src\ispit.vhd
11 -- Generated   : Mon Apr 27 18:47:10 2020
12 -- From       : interface description file
13 -- By        : Itf2Vhdl ver. 1.22
14 --
15 -----
16 --
17 -- Description :
18 --
19 -----
20
21 library IEEE;
22 use IEEE.STD_LOGIC_1164.all;
23
24 ENTITY inv IS
25     PORT (a: IN STD_LOGIC; b : OUT STD_LOGIC);
26 END inv;
27 ARCHITECTURE inv OF inv IS
28 BEGIN
29     b <= NOT a;
30 END inv;
31
32
33 library IEEE;
34 use IEEE.STD_LOGIC_1164.all;
35 ENTITY and_kolo IS
36     PORT (a, b: IN STD_LOGIC; c : OUT STD_LOGIC);
37 END and_kolo;
38
39 ARCHITECTURE and_kolo OF and_kolo IS
40 BEGIN
41     c <= a AND b;
42 END and_kolo;
43
44 library IEEE;
45 use IEEE.STD_LOGIC_1164.all;
46 ENTITY xnor_kolo IS
47     PORT (a, b: IN STD_LOGIC; c : OUT STD_LOGIC);
48 END xnor_kolo;
49
50 ARCHITECTURE xnor_kolo OF xnor_kolo IS
51 BEGIN
52     c <= a XNOR b;
53 END xnor_kolo;
54
55 library IEEE;
56 use IEEE.STD_LOGIC_1164.all;
57 ENTITY or_kolo IS
58     PORT (a, b: IN STD_LOGIC; c : OUT STD_LOGIC);
59 END or_kolo;
60
61 ARCHITECTURE or_kolo OF or_kolo IS
62 BEGIN
63     c <= a OR b;
64 END or_kolo;
65

```

Slika 5 i 6 : Instanciranje komponenata koje se koriste za projektovanje komparatora

Nakon instanciranja i deklaracije vršimo sintezu svih komponenata u kombinaciono kolo koje će uporediti dva trobitna broja. Na sledećoj slici možemo videti kod koji se nalazi u arhitekturi.

```

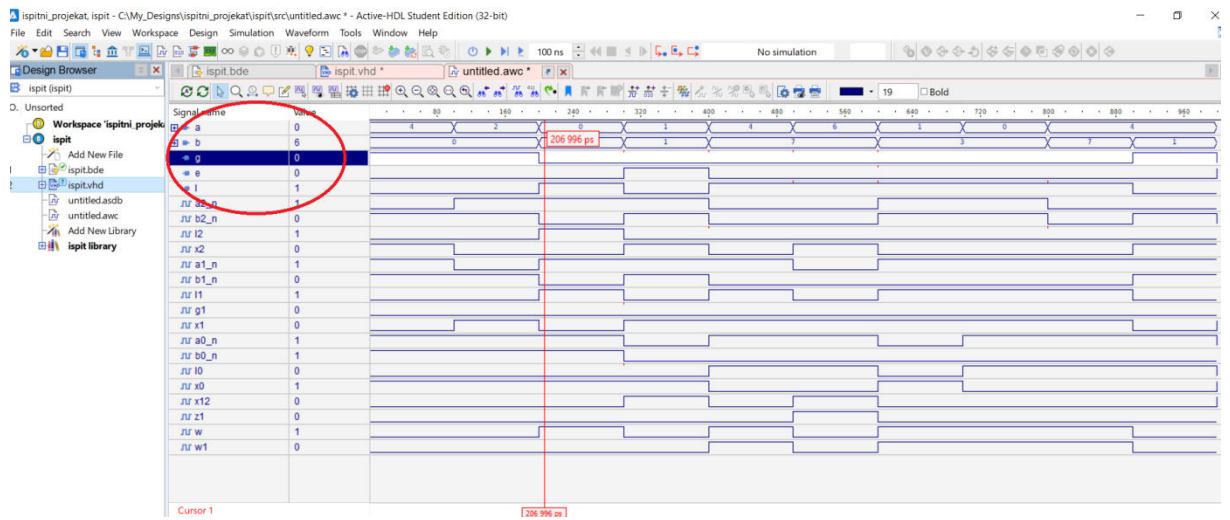
123 begin
124     INV1: inv PORT MAP(a(2), a2_n);
125     INV2: inv PORT MAP(b(2), b2_n);
126     INV3: inv PORT MAP(a(1), a1_n);
127     INV4: inv PORT MAP(b(1), b1_n);
128     INV5: inv PORT MAP(a(0), a0_n);
129     INV6: inv PORT MAP(b(0), b0_n);
130
131
132     XNOR1: xnor_kolo PORT MAP(a(2), b(2), x1);
133     XNOR2: xnor_kolo PORT MAP(a(1), b(1), x2);
134
135     AND1: and_kolo PORT MAP(x2, x1, x12);
136     AND2: and_kolo PORT MAP(a0_n, b(0), l0);
137     AND3: and_kolo PORT MAP(a2_n, b(2), l2);
138     AND4: and_kolo PORT MAP(a1_n, b(1), l1);
139
140     XNOR3: xnor_kolo PORT MAP(a(0), b(0), x0);
141     AND5: and_kolo PORT MAP(a(0), b0_n, g1);
142     AND6: and_kolo PORT MAP(x1, l1, w1);
143     AND9: and_kolo PORT MAP(x12, l0, z1);
144     XNOR4: xnor_kolo PORT MAP(w1, l2, c1);
145     XNOR5: xnor_kolo PORT MAP(c1, z1, l);
146
147     AND7: and_kolo PORT MAP(x12, x0, e);
148
149     INV7: inv PORT MAP(g1, g1_n);
150
151     AND8: and_kolo PORT MAP(x12, g1_n, c2);
152
153     OR1: or_kolo PORT MAP(w1, l2, w);
154
155     XNOR6: xnor_kolo PORT MAP(w, c2, g);
156 end ispit;

```

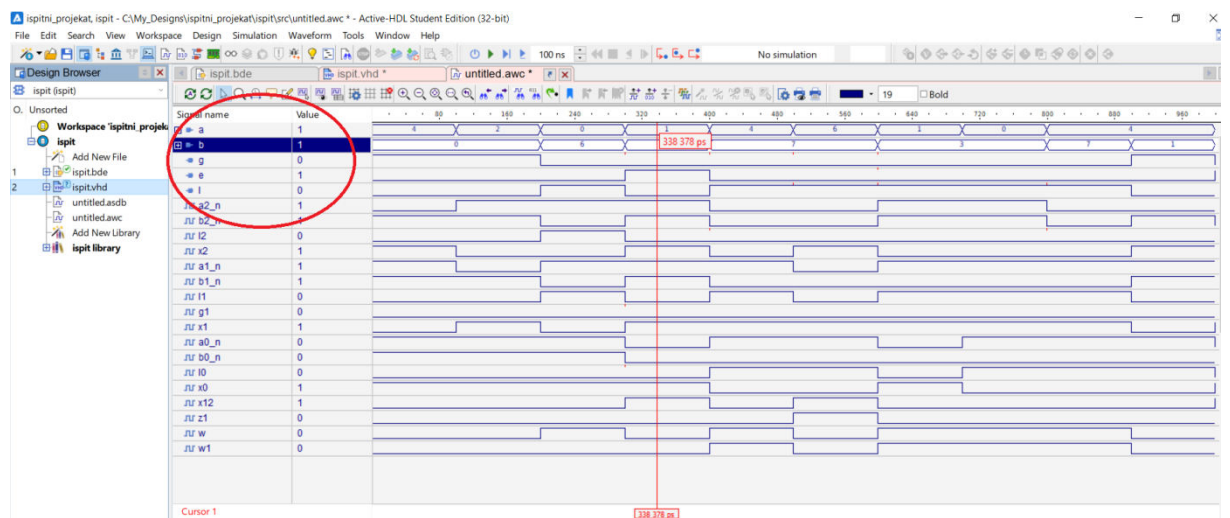
-- na osnovu priložene šeme vršimo sintezu komponenata u kombinaciono kolo

Slika 7 : VHDL opis trobitnog komparatora na nivou gejtova

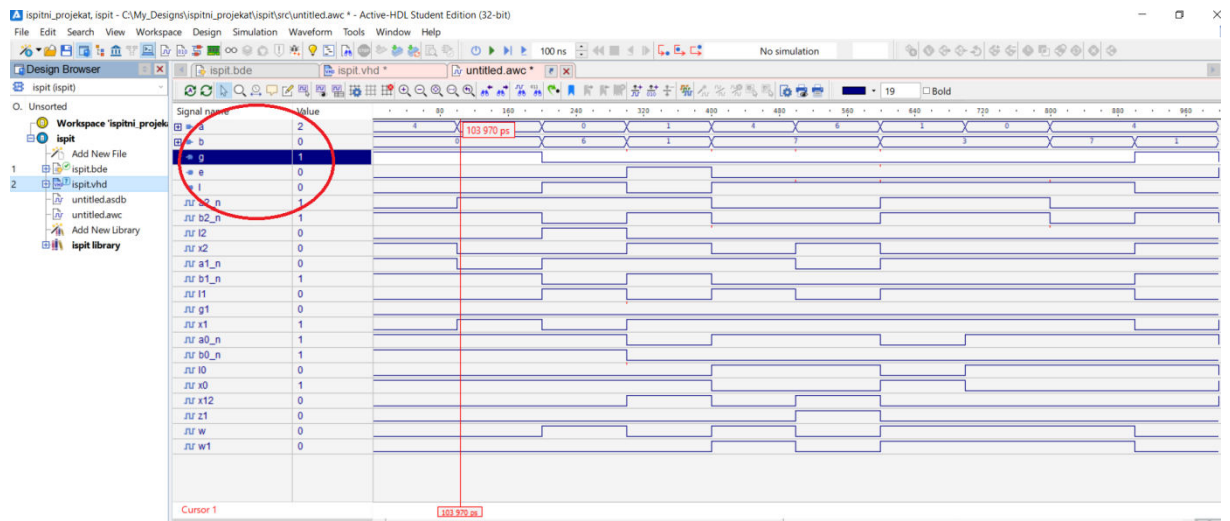
Sada, kada smo završili pisanje koda, sledi provera i ispitivanje grešaka. Ukoliko one ne postoje, verifikujemo ispravnost opisa kola simulacijom. Na sledećim slikama prikazani su rezultati simulacije poređenja dva trobitna broja, kada je jedan veći od drugog i kada su oni jednaki.



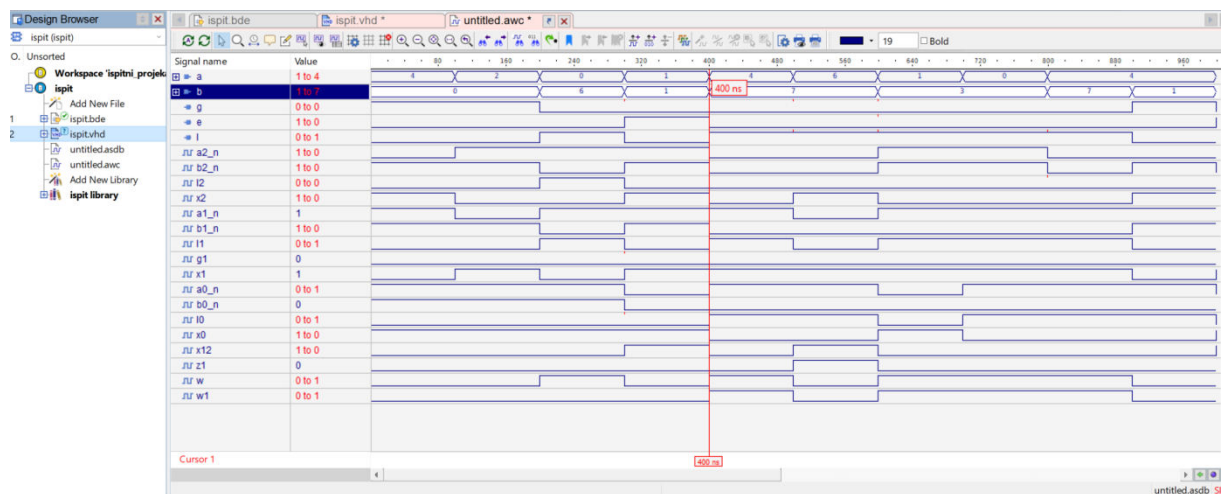
Slika 8 : Za vrednosti brojeva $a = 000$ i $b = 110$ signal na izlazu je $l=1$.



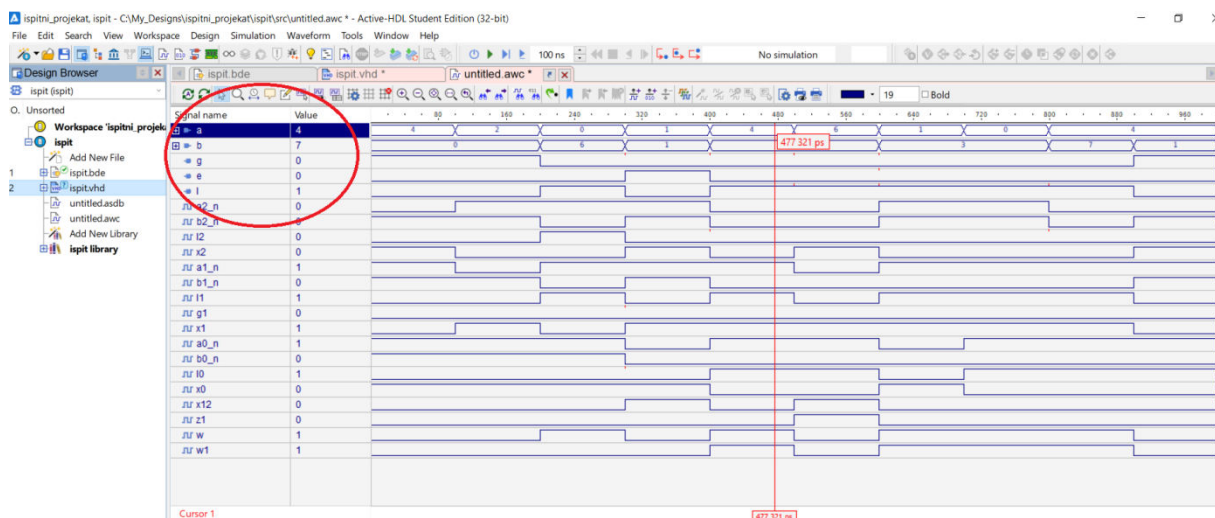
Slika 9 : Za vrednosti brojeva $a = 001$ i $b = 001$ signal na izlazu je $e = 1$.



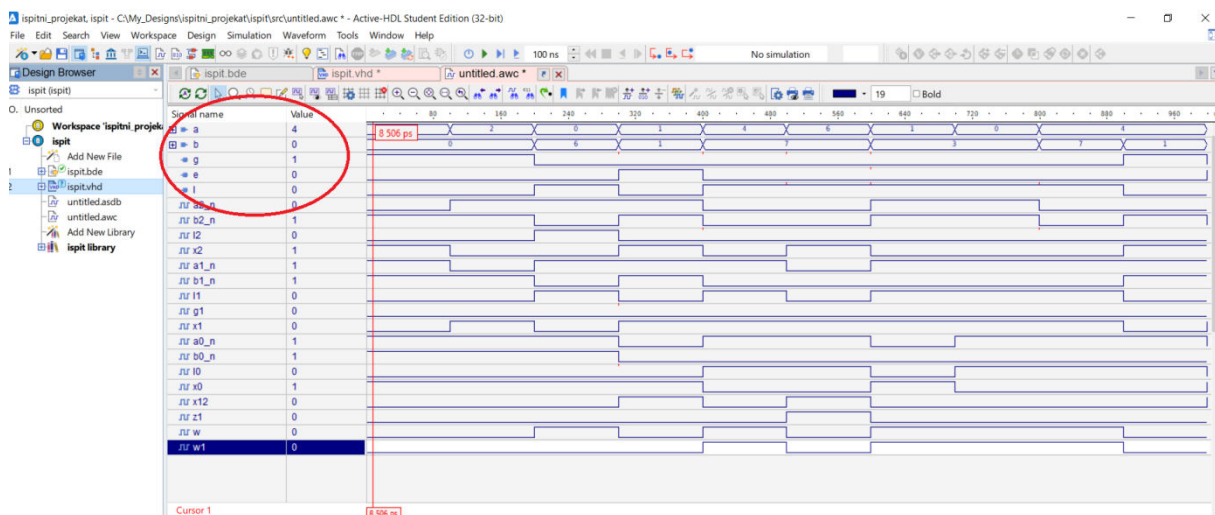
Slika 10 : Za vrednosti brojeva $a = 010$ i $b = 000$ signal na izlazu je $g = 1$.



Slika 11 : Promena vrednosti brojeva u 400 ns.



Slika 12 : Za vrednosti brojeva $a = 100$ i $b = 111$ signal na izlazu je $l = 1$.



Slika 13 : Za vrednosti brojeva $a = 100$ i $b = 000$ signal na izlazu je $g = 1$.

Simulacija će se završiti nakon 1000ns, nakon čega možemo videti ponašanje našeg kola za različite vrednosti brojeva A i B.

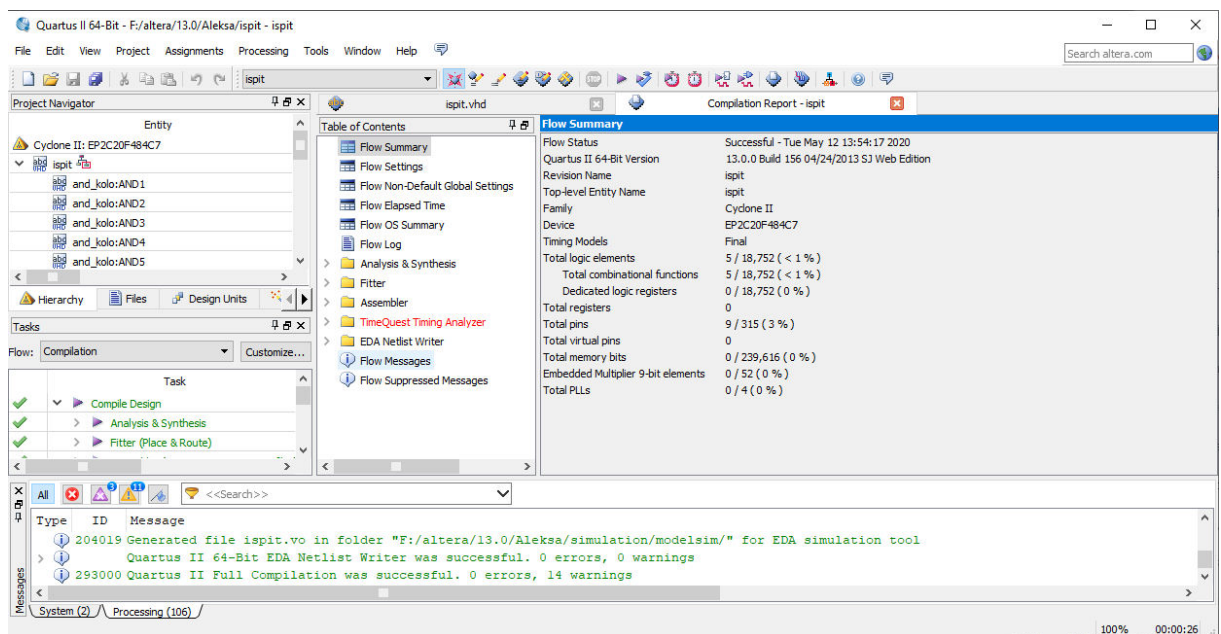
Realizacija kola na FPGA čipu Cyclone II korišćenjem programa Quartus i Altera DE1 razvojne ploče

Da bi se programabilne logičke komponente programirale, odnosno da bi im se ugradila željena funkcija, neophodno je korišćenje odgovarajućih softverskih alata. Postoje dva ključna koraka u projektovanju PLD, a to su strukturna faza projektovanja i fizičko projektovanje.

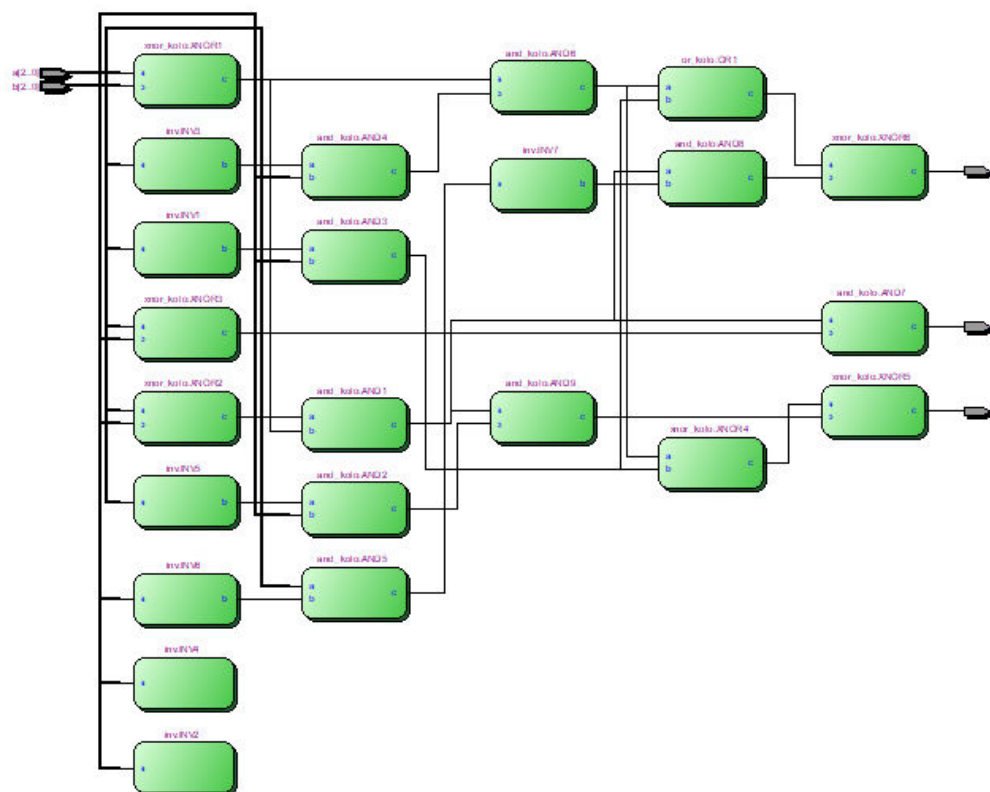
Strukturna faza projektovanja je faza u kojoj se opisuje šta kolo radi i na osnovu toga softver sintetizuje logičku strukturu.

Pod fizičkim projektovanjem se podrazumeva prenos projekta na hardver. Za ovaj zadatak upotrebićemo Quartus.

Da bi se počelo sa programiranjem FPGA, neophodno je opisati šta projekat treba da radi. Sledeći korak predstavlja prevođenje VHDL koda u logičke strukture koje fizički postoje u konkretnom FPGA kolu, odnosno automatska sinteza.



Slika 14 : Rezultat nakon prevođenja



Slika 15 : Rezultat sinteze kola

Aktivacijom prozora chip planner možemo videti fizički raspored blokova na FPGA. Sledeći korak u programiranju PLD-a, trobitnog komparatora, je da uz pomoć pin planner-a odredimo pinove koji će se koristiti u realizaciji.

Pin Planner - F:/altera/13.0/Aleksa/ispit - ispit

File Edit View Processing Tools Window Help

Search altera.com

Report

Report not available

Groups Report

Tasks

Run Anal

Early Pin

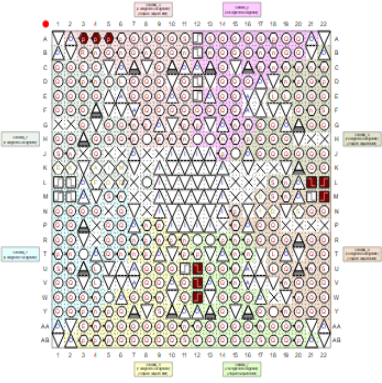
Early

Run

Expo

Change

Top View - Wire Bond
Cyclone II - EP2C20F484C7



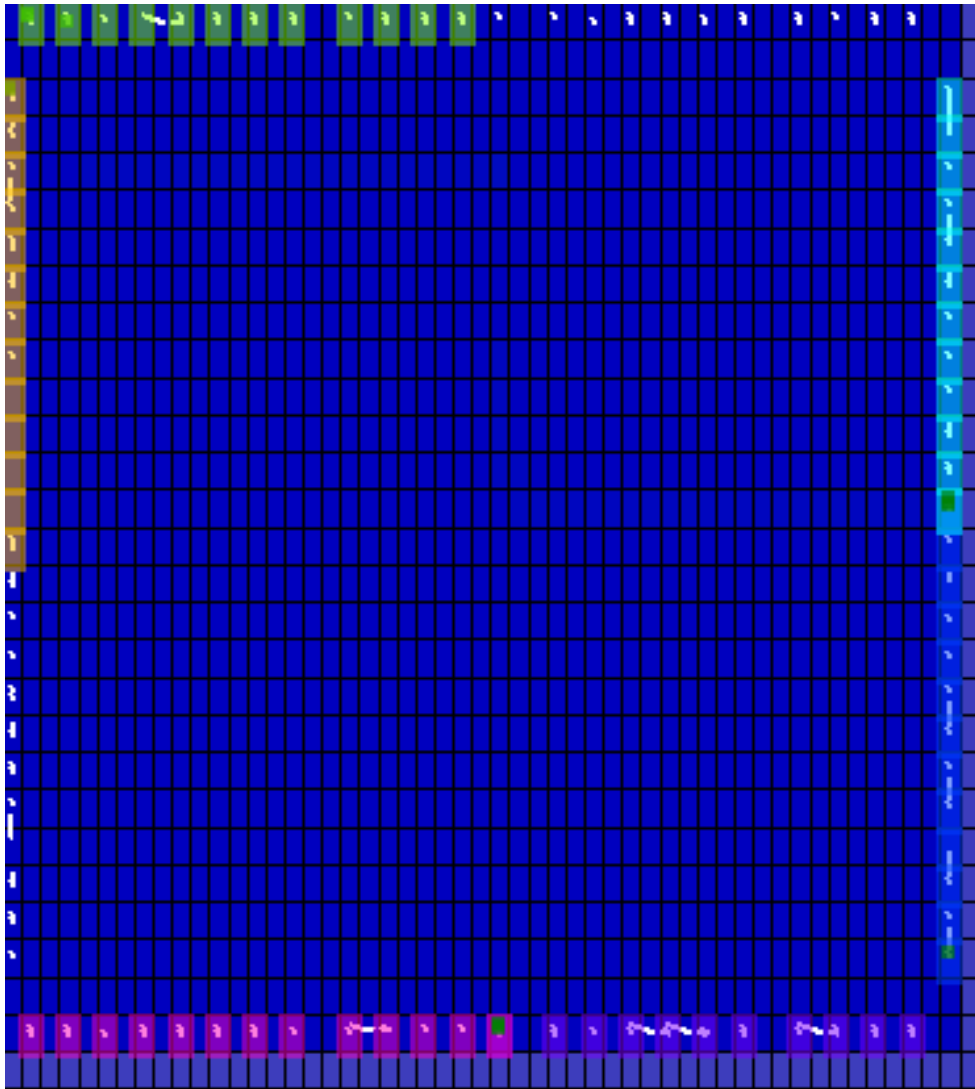
Named: * Edit: [Icons]

Filter: Pins: all

Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength	Differential Pair
in a[2]	Input	PIN_W12	7	B7_N1	PIN_L22	3.3-V LV...default		24mA (default)	
in a[1]	Input	PIN_U12	8	B8_N0	PIN_M5	3.3-V LV...default		24mA (default)	
in a[0]	Input	PIN_V12	7	B7_N1	PIN_P2	3.3-V LV...default		24mA (default)	
in b[2]	Input	PIN_L22	5	B5_N1	PIN_M6	3.3-V LV...default		24mA (default)	
in b[1]	Input	PIN_M22	6	B6_N0	PIN_W2	3.3-V LV...default		24mA (default)	
in b[0]	Input	PIN_L21	5	B5_N1	PIN_N3	3.3-V LV...default		24mA (default)	
out e	Output	PIN_A3	3	B3_N1	PIN_AB8	3.3-V LV...default		24mA (default)	
out g	Output	PIN_A4	3	B3_N1	PIN_N6	3.3-V LV...default		24mA (default)	
out l	Output	PIN_A5	3	B3_N1	PIN_P1	3.3-V LV...default		24mA (default)	
<<new node>>									

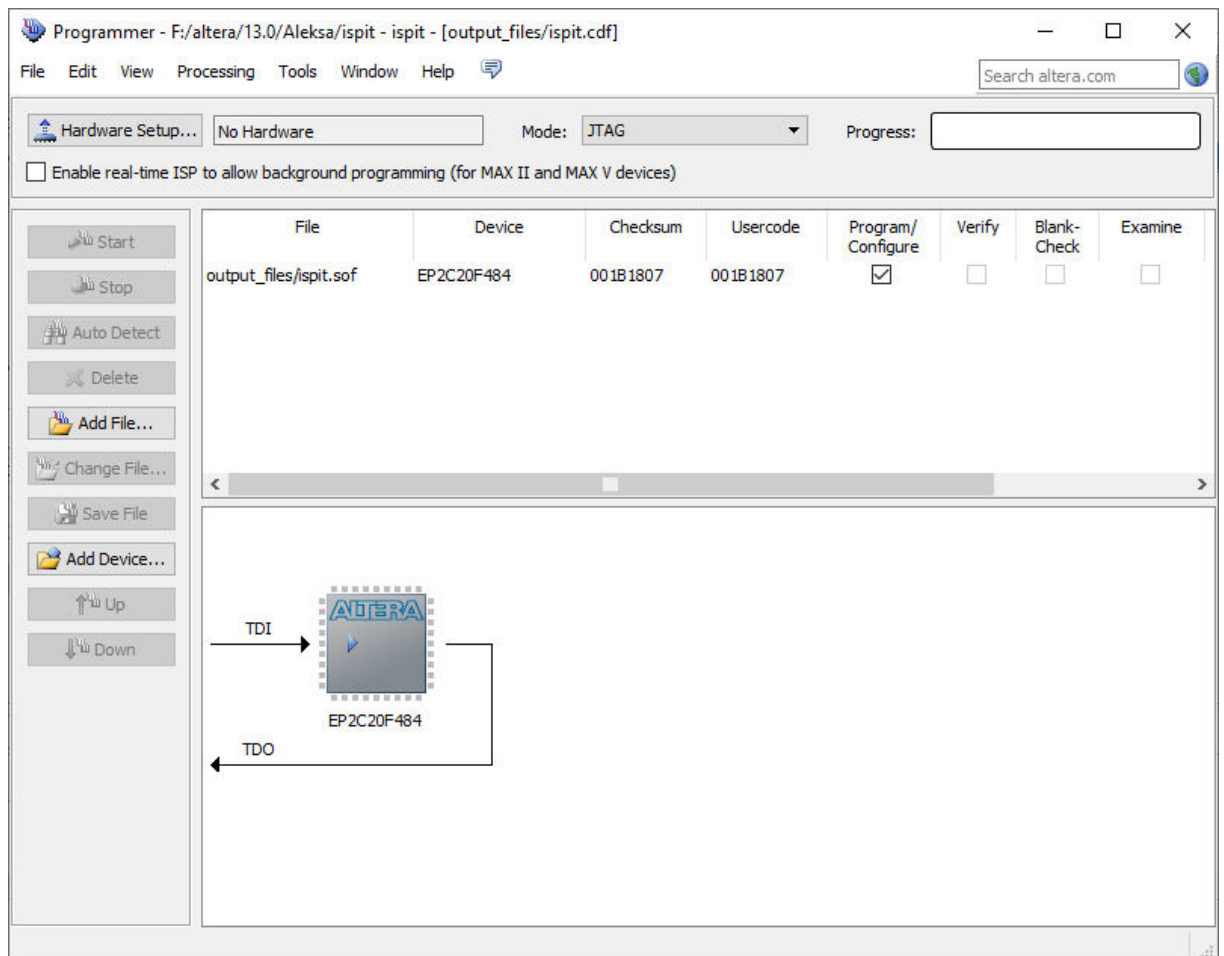
100% 00:00:26

Slika 16 : Pin Planner



Slika 17 : Chip planner

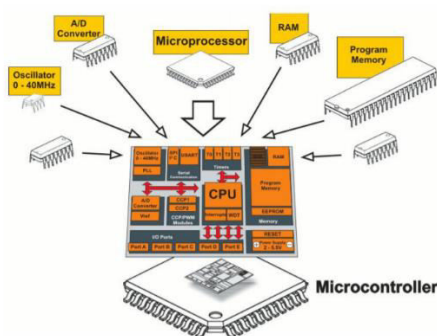
I na kraju, poslednji korak je programiranje FPGA koje se postiže korišćenjem alata Programmer. U prozoru treba selektovati opciju Add File. U novootvorenom prozoru za pretraživanje, sa nazivom Select Programming, prikazuju se svi raspoloživi fajlovi sa nastavkom .sof. Oni sadrže sve potrebne informacije o projektu, a FPGA se programira aktiviranjem dugmeta Start.



Slika 18 : Programiranje FPGA

Arduino

Arduino je open-source platforma za kreiranje elektronskih projekata. Arduino sadrži i fizičku ploču koja se može programirati (koja se često naziva mikrokontrolerom), ali i softver, odnosno IDE, koji pokreće računar, a koristi se i za pisanje i unošenje kompjuterskog koda na fizičku ploču. Arduino platforma je postala prilično popularna među onima koji tek ulaze u svet elektronike, i to s dobrim razlogom. Za razliku od prethodnih ploča koje se mogu programirati, Arduino ne zahteva poseban hardver - možemo prosto koristiti USB kabl. Pored toga, Arduino koristi pojednostavljenu verziju C++ jezika što ga čini još lakšim za naučiti. Na kraju, Arduino je sastavljen na takav način da čini funkcije mikrokontrolera još pritupačnijim. Arduino hardver i softver je namenjen umetnicima, dizajnerima, laicima, hakerima, novajlijama i svima ostalima koji su zainteresovani za stvaranje interaktivnih obejaka i okoline. Arduino se može povezati sa dugmićima, LED diodama, pa čak i sa vašim pametnim telefonom ili televizorom. Ovakva praktičnost, uz činjenicu da je Arduino softver besplatan, dok su hardverske ploče prilično jeftine, a pored toga je i lako naučiti kako se koriste softver i hardver je dovelo do stvaranja velike zajednice korisnika koji su doprineli kodu i objavili instrukcije za veliki broj Arduino projekata. Srce Arduino pločice je mikrokontroler. Skoro sve ostalo što postoji na pločici služi za njeno napajanje i omogućavanje njene komunikacije s računarom. Mikrokontroler je u suštini mali računar na čipu. On ima procesor, jedan ili dva kilobajta RAM memorije za skladištenje podataka, nekoliko kilobajta fleš memorije i ima ulazne i izlazne pinove. Ti U/I pinovi povezuju mikrokontroler s ostatkom elektronike. Ulazi mogu biti digitalni i analogni. To otvara mogućnost povezivanja mnogih vrsta senzora - za svetlost, temperaturu, zvuk itd. Izlazi takođe mogu biti analogni i digitalni, odnosno, možemo podesiti da određeni pin bude uključen ili isključen (5 volti odnosno 0 volti) i tako uključivati ili isključivati LED diode, a izlaz možemo koristiti i za upravljanje uređajima veće snage - kao što su motori.

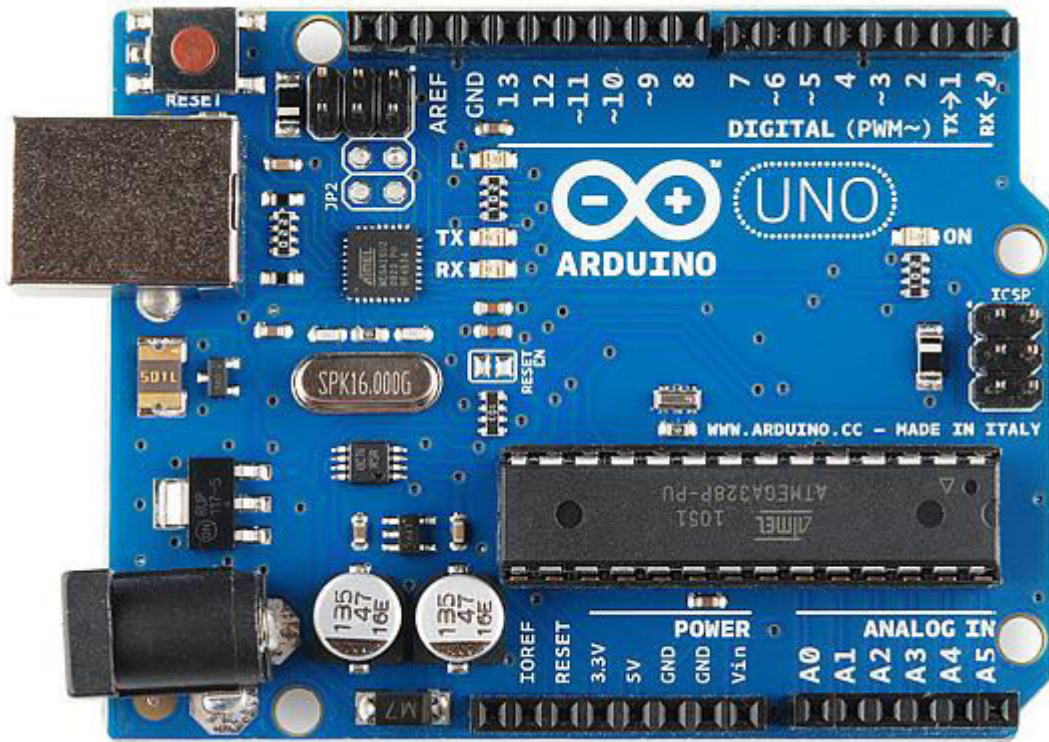


Slika 19 : Mikrokontroler

Arduino Uno je pločica za razvoj mikrokontrolera. Kod Arduino Uno pločice pinovi su grupisani u tri grupe, i to:

- 14 digitalnih pinova

- 6 analognih pinova
- Napajanje



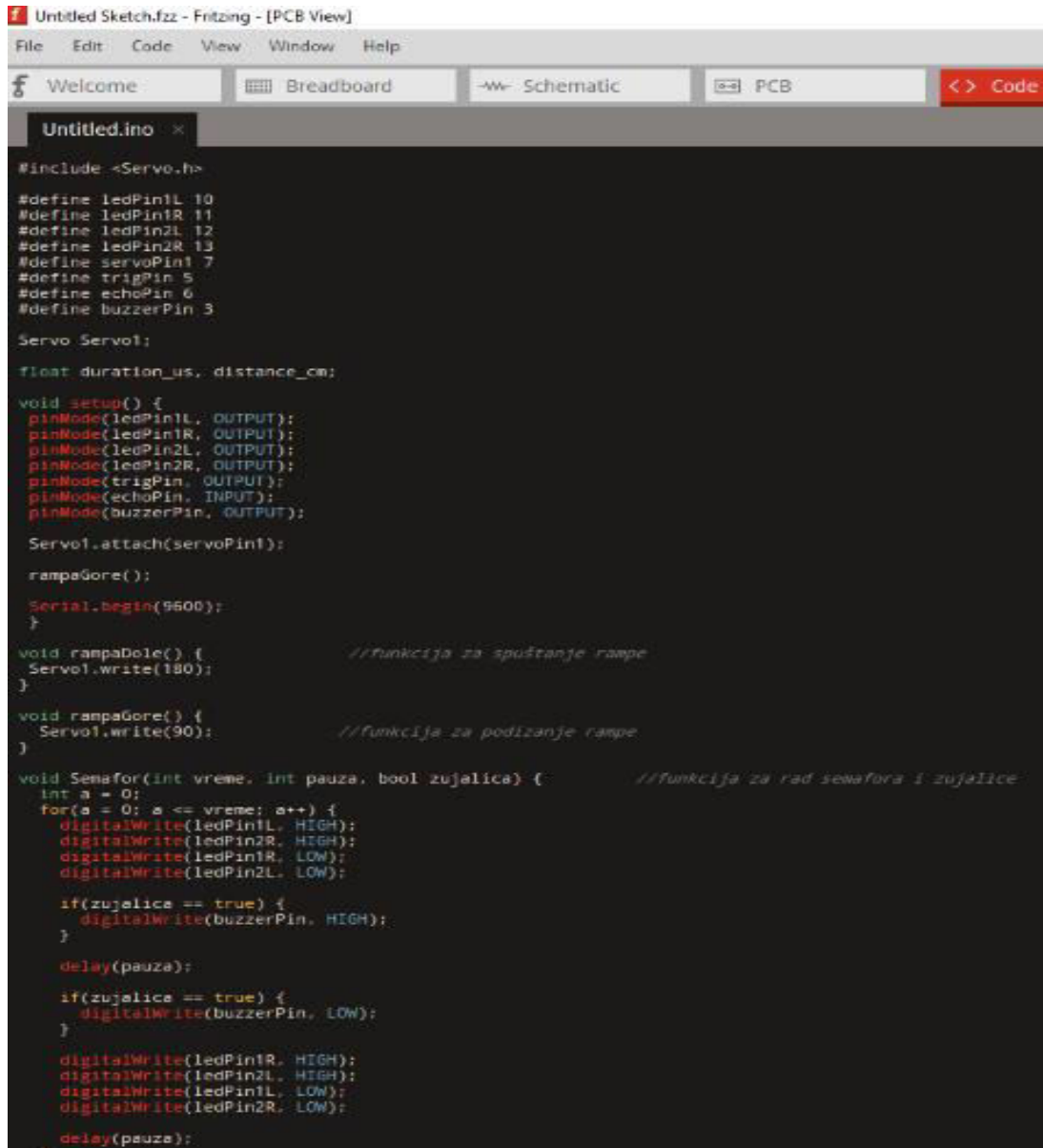
Slika 20 : Arduino Uno platforma

Naziv pina		Broj pina
(PCINT14/RESET) PC6	1	28 PC5 (ADC5/SCL/PCINT13)
(PCINT16/RXD) PD0	2	27 PC4 (ADC4/SDA/PCINT12)
(PCINT17/TXD) PD1	3	26 PC3 (ADC3/PCINT11)
(PCINT18/INT0) PD2	4	25 PC2 (ADC2/PCINT10)
(PCINT19/OC2B/INT1) PD3	5	24 PC1 (ADC1/PCINT9)
(PCINT20/XCK/T0) PD4	6	23 PC0 (ADC0/PCINT8)
VCC	7	22 GND
GND	8	21 AREF
(PCINT6/XTAL1/TOSC1) PB6	9	20 AVCC
(PCINT7/XTAL2/TOSC2) PB7	10	19 PB5 (SCK/PCINT5)
(PCINT21/OC0B/T1) PD5	11	18 PB4 (MISO/PCINT4)
(PCINT22/OC0A/AIN0) PD6	12	17 PB3 (MOSI/OC2A/PCINT3)
(PCINT23/AIN1) PD7	13	16 PB2 (SS/OC1B/PCINT2)
(PCINT0/CLKO/ICP1) PB0	14	15 PB1 (OC1A/PCINT1)

Slika 21 : ATmega328 Mikrokontroler

Projektni zadatak

Tema projektnog zadatka je - realizovati zvučnu i vertikalnu signalizaciju jednog pružnog prelaza. Komponente koje koristimo za uspešnu realizaciju ovog zadatka su LED diode, senzor, servo motor i zujalica.



```
#include <Servo.h>

#define ledPin1L 10
#define ledPin1R 11
#define ledPin2L 12
#define ledPin2R 13
#define servoPin1 7
#define trigPin 5
#define echoPin 6
#define buzzerPin 3

Servo Servo1;

float duration_us, distance_cm;

void setup() {
  pinMode(ledPin1L, OUTPUT);
  pinMode(ledPin1R, OUTPUT);
  pinMode(ledPin2L, OUTPUT);
  pinMode(ledPin2R, OUTPUT);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(buzzerPin, OUTPUT);

  Servo1.attach(servoPin1);

  rampaGore();

  Serial.begin(9600);
}

void rampaDole() { //funkcija za spuštanje rampe
  Servo1.write(180);
}

void rampaGore() { //funkcija za podizanje rampe
  Servo1.write(90);
}

void Senafor(int vreme, int pauza, bool zujalica) { //funkcija za rad senafora i zujalice
  int a = 0;
  for(a = 0; a <= vreme; a++) {
    digitalWrite(ledPin1L, HIGH);
    digitalWrite(ledPin2R, HIGH);
    digitalWrite(ledPin1R, LOW);
    digitalWrite(ledPin2L, LOW);

    if(zujalica == true) {
      digitalWrite(buzzerPin, HIGH);
    }

    delay(pauza);

    if(zujalica == true) {
      digitalWrite(buzzerPin, LOW);
    }

    digitalWrite(ledPin1R, HIGH);
    digitalWrite(ledPin2L, HIGH);
    digitalWrite(ledPin1L, LOW);
    digitalWrite(ledPin2R, LOW);

    delay(pauza);
  }
}
```

Slika 23 : Kod

Na samom početku koda, uključujemo biblioteku za servo. Nakon uključivanja biblioteke, definišemo nazive pinova, pa će se tako npr. pin pod rednim brojem 10 zvati ledPin1L, pin broj 5 trigPin itd. Kreiranjem klase Servo1, možemo pristupati svim funkcijama ove biblioteke. U delu koda koji počinje sa void setup() postavljamo početne vrednosti, odnosno definišemo koji će nam pinovi biti ulazni, a koji izlazni (npr. ledPin1L će biti izlazni pin, dok će echoPin biti ulazni). Linijom koda Servo1.attach(servoPin1) smo definisali gde ćemo povezati servo motor i na početku našeg programa postavljamo rampu tako da je ona podignuta. Uz pomoć funkcije voidrampaDole() vršimo spuštanje rampe, tako što će servo motor zauzeti određeni položaj. Suprotno od ove funkcije imamo funkciju voidrampaGore() koja podiže rampu , a servo motor će zauzeti drugi ugao.

Dalje kroz kod, dolazimo do funkcije koja nam služi za rad semafora i zujalice, kojom preko for petlje vršimo paljenje i gašenje led diode (semafor) i zujalice. Ako korisnik zada vreme=5, pauza 1000 i true za zujalicu, važi sledeće: pali se po jedna dioda za svaki semafor kao i zujalica, nakon pauze od 1000ms, zujalica se gasi i menjaju se diode koje svetle, odnosno one diode koje su bile ugašene sada se pale i obrnuto, sve to prolazi kroz petlju, broj prolazaka zavisi od vremena koje je korisnik zadao.

```
Serial.begin(9600);
}

void rampaDole() {           //funkcija za spuštanje rampe
  Servo1.write(180);
}

void rampaGore() {           //funkcija za podizanje rampe
  Servo1.write(90);
}

void Senafor(int vreme, int pauza, bool zujalica) {           //funkcija za rad semafora i zujalice
  int a = 0;
  for(a = 0; a <= vreme; a++) {
    digitalWrite(ledPin1L, HIGH);
    digitalWrite(ledPin2R, HIGH);
    digitalWrite(ledPin1R, LOW);
    digitalWrite(ledPin2L, LOW);

    if(zujalica == true) {
      digitalWrite(buzzerPin, HIGH);
    }

    delay(pauza);

    if(zujalica == true) {
      digitalWrite(buzzerPin, LOW);
    }

    digitalWrite(ledPin1R, HIGH);
    digitalWrite(ledPin2L, HIGH);
    digitalWrite(ledPin1L, LOW);
    digitalWrite(ledPin2R, LOW);

    delay(pauza);
  }

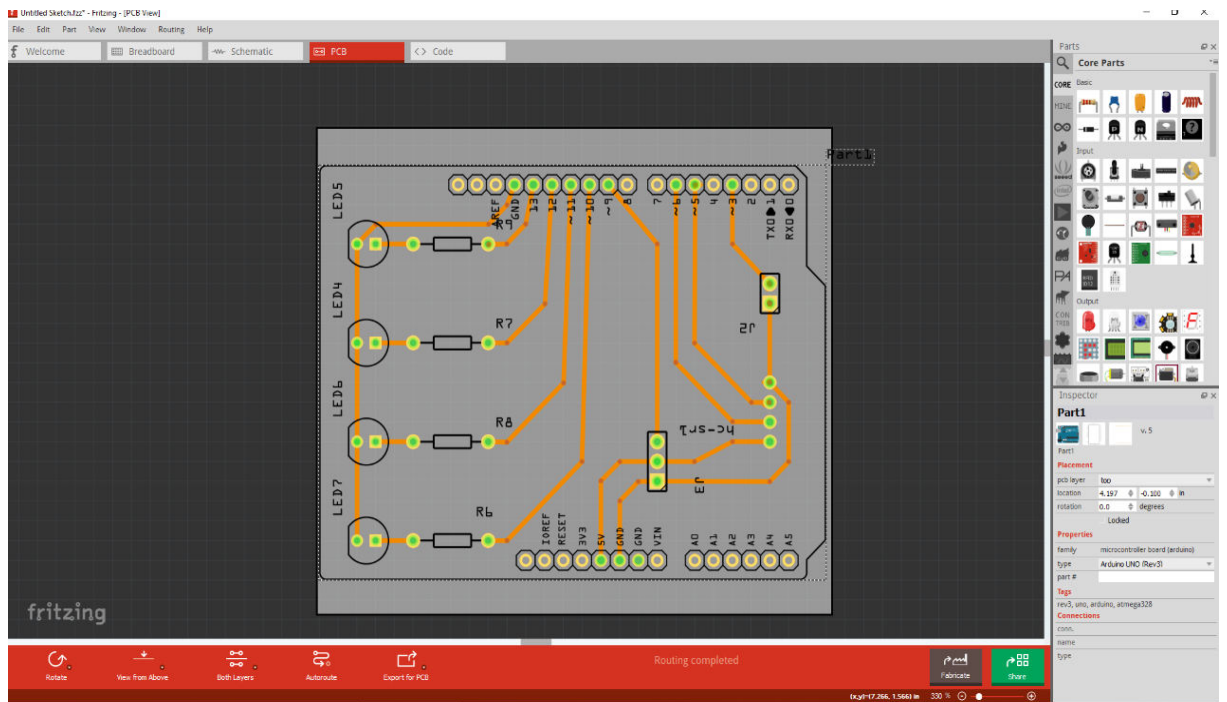
  digitalWrite(ledPin1R, LOW);
  digitalWrite(ledPin2L, LOW);
  digitalWrite(ledPin1L, LOW);
  digitalWrite(ledPin2R, LOW);
}

void loop() {
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration_us = pulseIn(echoPin, HIGH);
  distance_cm = 0.017 * duration_us;

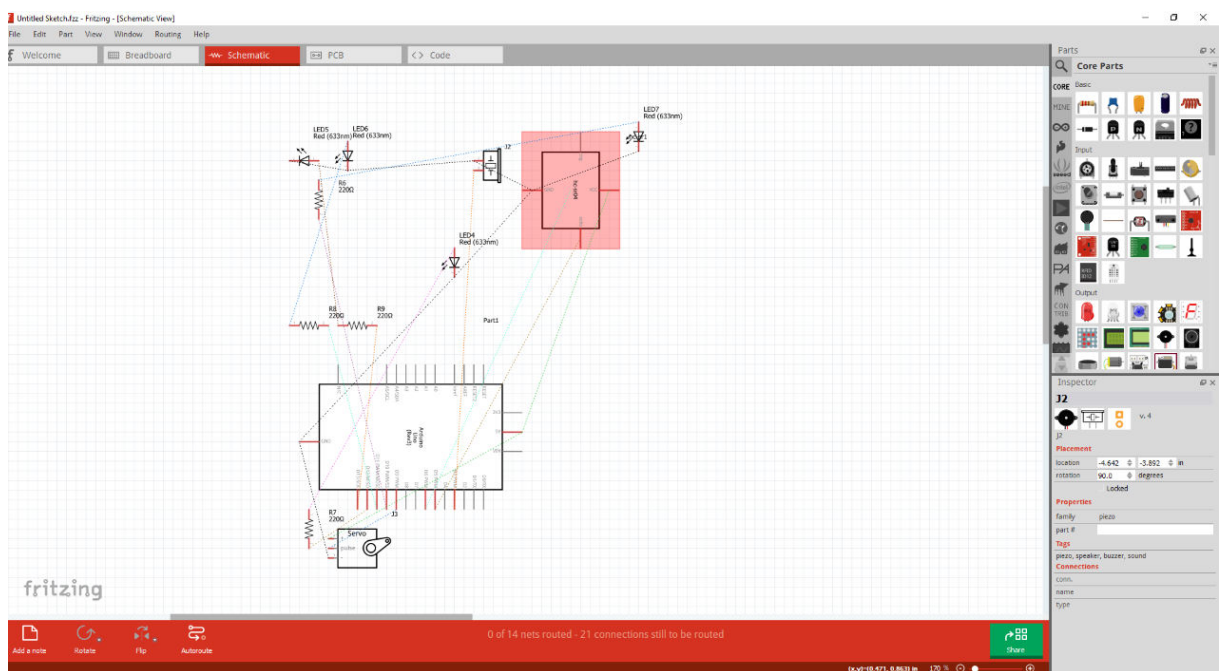
  if (distance_cm < 7.0) {
    Senafor(5, 500, true);
    rampaDole();
    Senafor(12, 500, true);
    rampaGore();
    Senafor(3, 500, false);
  }
  else{
    delay(500);
  }
}
```

Slika 24 : Kod

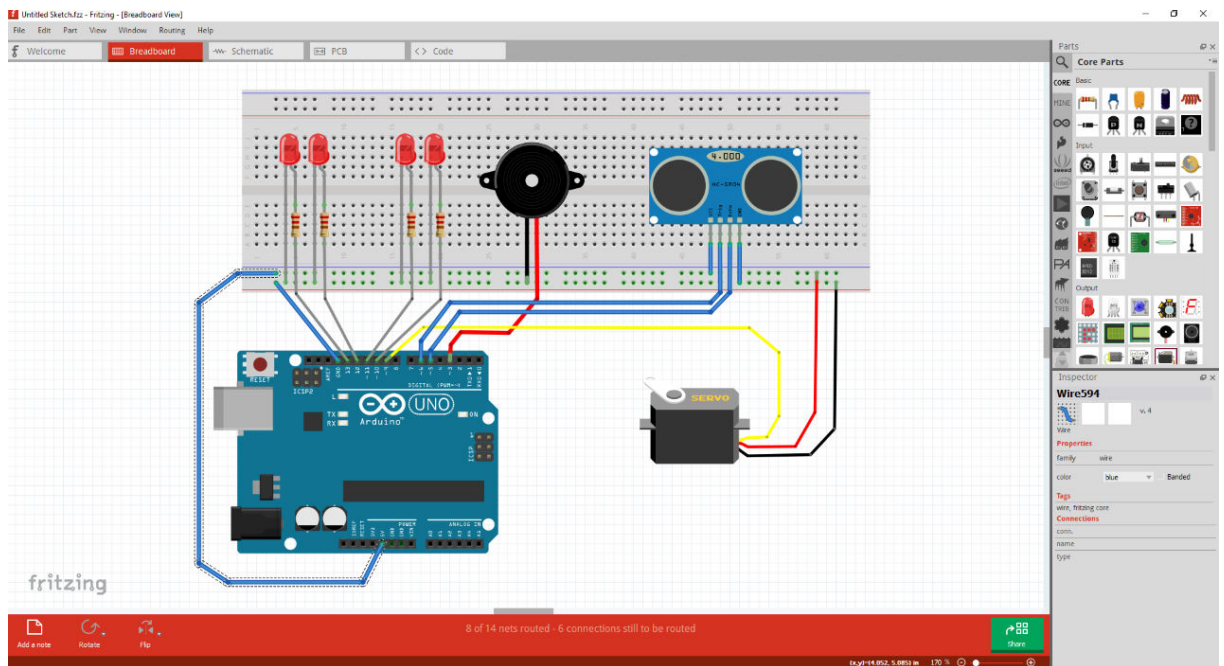
Void loop predstavlja glavni program i on ne prestaje sa izvršavanjem. Uz pomoć formule $distance_cm = 0.017 * duration_us$ senzor određuje daljinu, pa uz pomoć if petlje ispitujemo da li je ta distanca manja od 7cm, u slučaju da jeste, dolazimo do paljenja semafora i spuštanja rampe, u slučaju da je distanca veća od 7cm, semafori ne rade, rampa je podignuta i omogućeno je normalno odvijanje saobraćaja preko pružnog prelaza.



Slika 25 : PCB



Slika 26 : Šema



Slika 27 : Ploča

Povezivanje dioda, senzora i servo motora sa pinovima na Arduino platformi vršimo na osnovu toga kako je definisano u našem kodu.

Literatura

1. http://leda.elfak.ni.ac.rs/education/PEK_Elektronika/PEK_EKS_main.html
2. http://leda.elfak.ni.ac.rs/education/PEK_Elektronika/literatura/predavanja%20EKS/2019-20/05/VHDL%20primena%20u%20sintezi.pdf
3. [http://leda.elfak.ni.ac.rs/education/PEK_Elektronika/literatura/predavanja%20ELE%20\(EKS\)/vezbe_VLSI_1.pdf](http://leda.elfak.ni.ac.rs/education/PEK_Elektronika/literatura/predavanja%20ELE%20(EKS)/vezbe_VLSI_1.pdf)
4. [http://leda.elfak.ni.ac.rs/education/PEK_Elektronika/literatura/predavanja%20ELE%20\(EKS\)/vezbe_VLSI_2.pdf](http://leda.elfak.ni.ac.rs/education/PEK_Elektronika/literatura/predavanja%20ELE%20(EKS)/vezbe_VLSI_2.pdf)
5. P.Petković, M. Andrejević Stošović, M. Milić, D. Mirković - Praktikum laboratorijskih vežbi iz predmeta Projektovanje elektronskih kola i Projektovanje digitalnih elektronskih kola, Poglavlje Projektovanje elektronskih kola