
UNIVERZITET U BANJOJ LUCI
ELEKTROTEHNIČKI FAKULTET
RAČUNARSTVO I INFORMATIKA

# MANIPULISANJE 7-SEGMENTNIM DISPLEJIMA

IZVJEŠTAJ PROJEKTNOG ZADATKA IZ PREDMETA UGRAĐENI RAČUNARSKI SISTEMI

Studenti:
Katarina Antešević 1187/15
Tamara Lekić 11113/19
Jovana Arežina 1105/15

Mentori:
Profesor: prof. dr. Zlatko Bundalo
Asistent: Marko Mihajlović

Banja Luka, jun, 2022.

**SADRŽAJ**

# 1. UVOD

## 1.1. Tekst projektnog zadatka

Pri pokretanju aplikacije pali se prvih *n* led dioda i pomoću displeja ispisati „boje“ dioda. Broj *n* može biti bilo koji broj iz skupa N = {3, 4, 5, 6}. Koriste se displeji za prikaz „boje“ dioda jer su sve diode na ploči crvene. Implementirati da se broj *n* može definisati ručno (unos sa standardnog ulaza, kao argument komandne linije ili čitanje iz fajla) i generisan nasumično.

Potrebno je izabrati jedan od prvih *n* prekidača da bi se izvršavanje aplikacije podrazumijevalo uspješno. Ukoliko se naprave 3 pogrešna izbora, aplikacija završava sa radom i na svim displejima je potrebno prikazati slovo H te 5s nakon toga ispisati FAILED. Izbor prekidača se vrši prateći sledeća pravila (jedno pravilo će uvijek odgovarati testnom slučaju, ukoliko ima više pravila prati se ono na koje se prvo naiđe):

- ***3 diode***
    - o Ukoliko postoji samo jedna crvena dioda prebacite drugi prekidač.
    - o Ukoliko je zadnja dioda bijela prebacite zadnji prekidač
    - o Ukoliko postoji više od jedne plave diode, prebaciti prekidač koji odgovara zadnjoj plavoj diodi
    - o U suprotnom prebacite zadnji prekidač
- ***4 diode***
    - o Ukoliko postoji više od jedne crvene diode i ukoliko je zadnja cifra serijskog broja neparna prebacite prekidač koji odgovara zadnjoj crvenoj diodi
    - o Ukoliko je zadnja dioda žuta i ne postoji ni jedna crvena dioda, prebaciti prvi prekidač
    - o Ukoliko postoji tačno jedna plava dioda, prebaciti prvi prekidač
    - o Ukoliko postoji više od jedne žute diode, prebacite zadnji prekidač
    - o U suprotnom prebacite drugi prekidač

- ***5 dioda***
    - o Ukoliko je zadnja dioda crna i ukoliko je zadnja cifra serijskog broja neparna, prebacite četvrti prekidač
    - o Ukoliko postoji tačno jedna crvena dioda i ukoliko postoji više od jedne žute diode, prebacite prvi prekidač
    - o Ukoliko ne postoji ni jedna crna dioda, prebacite drugi prekidač
    - o U suprotnom prebacite prvi prekidač
- ***6 dioda***
    - o Ukoliko ne postoji ni jedna žuta dioda i ukoliko je zadnja cifra serijskog broja neparna, prebacite treći prekidač
    - o Ukoliko postoji tačno jedna žuta dioda i postoji više od jedne bijele diode, prebacite četvrti prekidač
    - o Ukoliko ne postoji ni jedna crvena dioda, prebacite zadnji prekidač
    - o U suprotnom prebacite četvrti prekidač

Poslednja 4 prekidača (ili tasteri) se koriste za izbor režima rada, samo jedan može biti uključen u bilo kojem trenutku. Ukoliko je prebačen sedmi prekidač „boje“ dioda u cikličnom redoslijedu prikazuju na displejima. Ukoliko je prebačen osmi prekidač potrebno je na displejima prikazati vrijednost tajmera (implementacija tajmera je opisana naknadno). Ukoliko je prebačen deveti prekidač potrebno je na displejima prikazati serijski broj. Ukoliko je prebačen deseti prekidač potrebno je na displejima prikazati broj grešaka (u slučaju jedne greške dva displeja prikazuju H dok su ostali prazni, u slučaju dve greške četiri displeja prikazuju H dok su ostali prazni, u slučaju tri greške svi displeji prikazuju H).

Prikaz „boja“ dioda realizovati kombinovanjem malih i velikih slova zbog ograničenja koja unose 7-segmentni displeji. Serijski broj predstavlja šestocifreni broj iz dekadnog brojnog sistema. Za izbor „boja“ dioda i serijskog broja je potrebno obezbijediti način ručnog definisanja i nasumičnog generisanja, slično kao i za broj dioda.

Potrebno je implementirati tajmer koji odbrojava unazad počevši od 2 minute odnosno 120 sekundi. Za implementaciju se mogu koristiti *interrupt* i *timer* ugrađene

biblioteke ili *usleep* funkcija i interni brojač. Drugi način neće dati najtačnije rješenje ali će ipak biti prihvaćeno.

Kada tajmer odbroji do nule potrebno je na displejima prikazati FAILED i završiti izvršavanje aplikacije. U slučaju uspješnog izvršavanja aplikacije na displejima prikazati SUCSES.

## 1.2. Proces pripreme ploče i okruženja za impementaciju aplikacije

Implementacija datog projektnog zadatka se svodi na pisanje aplikacije u *DS-5 Eclipse* razvojnom okruženju za *HPS* komponentu na *DE1-SoC* ploči – platformi (detaljniji opis ploče i okruženja slijedi u poglavlju 2 ovog rada). Međutim, prije početka samog pisanja koda korisničke aplikacije čija je specifikacija data tekstom projektnog zadatka bilo je neophodno izvršiti potrebne pripreme *DE1-SoC* ploče i *DS-5 Eclipse* okruženja koje su ovde ukratko opisane.

### *1.2.1. Priprema DE1-SoC ploče*

Priprema *DE1-SoC* ploče prije svega podrazumijeva ***pripremu potrebnih fajlova za narezivanje SD kartice*** koja služi za *boot*-anje ploče i obuhvata sljedeći niz koraka:

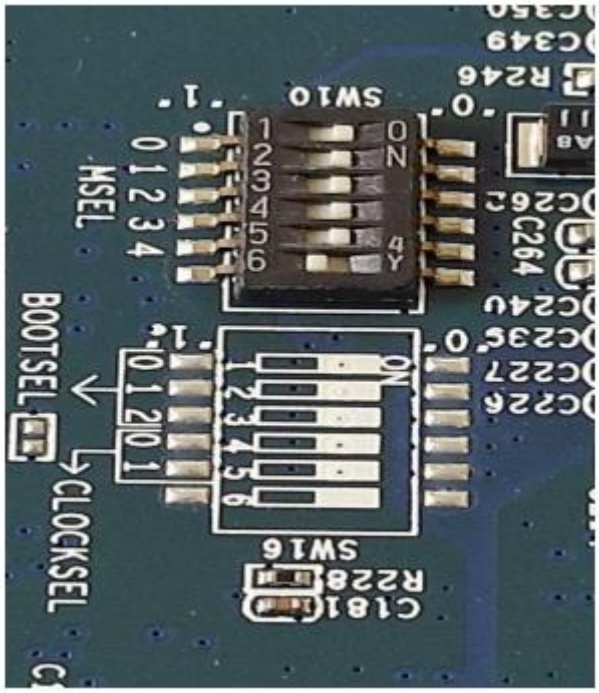
- Kreiranje potrebnih fajlova kao što su ***.dts*** – *device tree source* (koji se kompajlira u ***.dtb*** – *device tree blob* fajl koji opisuje hardversku platformu nezavisno od operativnog sistema i proslijeđuje se operativnom sistemu iz *bootloader*-a), ***.rbf*** fajl (binarna datoteka koja sadrži konfiguracione podatke za upotrebu izvan *Quartus Prime Sowtware*-a), *u-boot* (*bootloader* korišten od strane *embedded Linux* OS-a za pokretanje kernela, uz opciju da obavijesti kernel gdje se nalazi *root filesystem*; *u-boot* se pokreće prvi nakon ROM *bootloader-a*) i drugi.
- Pribavljanje ***zImage***-a (komprimovana verzija slike jezgra Linux OS-a koja se sama raspakuje) i kreiranje ***rootfs***-a (osnovni fajl sistem *Linux* operativnog sistema koji se nalazi na istoj particiji kao i *root* direktorijum; to je fajl sistem na koji se *mount*-uju svi ostali fajl sistemi nakon *boot*-anja; za kreiranje *rootfs*-a korišten je alat koji se zove *buildroot*).
- Generisanje *sdcard image*-a – *image* datoteka za SD karticu (koja objedinjuje kernel, *rootfs* i *u-boot* i moguće je izvesti njeno kopiranje na SD karticu komandom *dd* iz *Linux* terminala; nakon kopiranja se dobije FAT particija sa kernelom OS-a, *device tree*-om i EXT particija sa *rootfs*-om; *u-boot* je u neparticionisanoj oblasti prije FAT-a).

- Narezivanje SD karice (prepisivanje sadržaja SD karice odgovarajućim *sdcard image* fajlom). Ovaj korak je sproveden korištenjem *Win32DiskImager* programa korištenjem opcije *write* koju, između ostalih, nudi ovaj program.

Prije nastavka opisivanja procesa pripreme ploče neophodno je napomenuti da se prva 3 koraka, navedena u tekstu iznad, odnose na "ručno" pripremanje *sdcard image* fajla za narezivanje na SD karticu. Konkretno za ovaj projektni zadatak, mi smo dobili gotov *sdcard image* fajl, a zatim smo samo sproveli korak 4 (narezivanje SD kartice).

Nakon pripreme fajlova za SD karticu i narezivanja iste slijedi niz koraka koji se odnosi na ***boot-anje DE1-SoC ploče***, a to su sljedeći koraci:

- Podešavanje FPGA konfiguracionog režima koji se sprovodi odgovarajućim podešavanjem konfiguracionih bita (smještenih na donjem dijelu ploče) prikazano je na Slici 1.1.




*Slika 1.1. Podešavanje konfiguracionih bita*

- Nakon ubacivanja SD kartice u odgovarajući port i povezivanja ploče sa napajanjem i *host* računarom pokreće se *Tera Term* (ili bilo koji drugi program za emulaciju serijskog terminala).
- Podešavanjem tipa konekcije na serijsku, izborom odgovarajućeg USB serijskog porta na *host* računaru, te podešavanjem odgovarajuće brzine prenosa na 115200 pokreće se process *boot*-anja ploče.

Nakon *boot*-anja ploča je spremna za pokretanje aplikacije.

### *1.2.2. Priprema Eclipse razvojnog okruženja za DS-5*

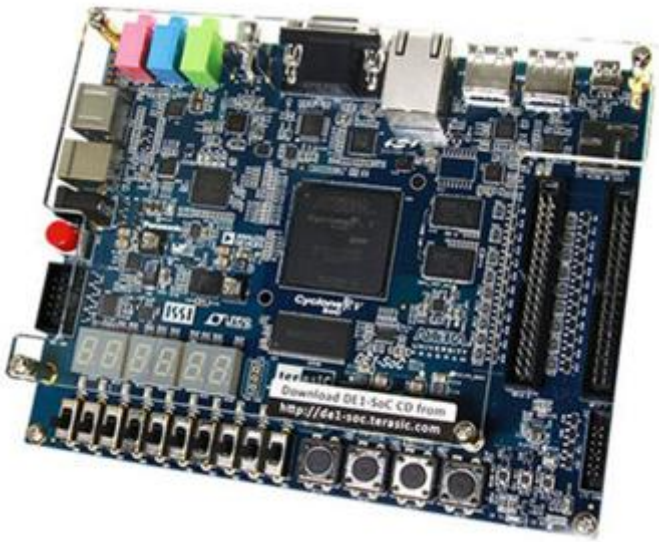
Proces ***pripreme Eclipse razvojnog okruženja za DS-5*** obuhvata sljedeće korake:

- Pošto se *DE1-SoC* ploča sa *host* računarom povezuje i preko *ethernet* kabla potrebno je za *Network Interface* u okruženju izabrati interfejs sa *host* računara koji koristi *Ethernet* port.
- Kako bi se mogla uspostaviti konekcija sa pločom potrebno je kreirati SSH konekciju gdje se kao *Host name* postavlja IP adresa koja je dodijeljena ploči (u našem slučaju 192.168.23.100), a za *Connection name* dozvoljeno je postaviti proizvoljno ime (to ime se kasnije koristi pri konfiguraciji *debugger*-a). U našem slučaju, ime konekcije je postavljeno na *DE1-SoC*.
- Za kompajliranje C koda potreban je odgovarajući *toolchain* (lanac alata). U našem slučaju, koristi se *linaro toolchain* koji je nakon preuzimanja potrebno dodati u *Eclipse* okruženje za DS-5.
- Prije svakog kreiranja projekta u *Eclipse* razvojnom okruženju za DS-5 neophodno je dodati putanje za potrebne biblioteke (iz *intelFPGA* direktorijuma sa fajl sistema), kao i simbole za pretprocesor. Pored toga, u projekat je potrebno uvesti i *source* fajl *alt_generalpurpose_io.c* u kojem je definisano manipulisanja ulazima i izlazima opšte namjene.
- Posljednji korak je konfigurisanje *debugger*-a i ovo je jedini korak od svih nabrojanih koji se sprovodi nakon izvršenog uspješnog kompajliranja koda, zato što je za njegovo podešavanje potreban binarni fajl koji se generiše nakon uspješno sprovedenog kompajliranja projekta. Konfiguraciji se prvo dodjeljuje proizvoljno ime koje se koristi za povezivanje sa ciljnom platformom (*DE1-SoC* ploča). Zatim se postavlja konekcija na odgovarajuću SSH konekciju čije je kreiranje već opisano u drugom koraku (i to je *DE1-SoC*), *Debug from symbol* opcija se podešava da bude *main* i ovo podrazumijeva da će process *debug*-ovanja početi od funkcije *main*, itd.

# 2. DE1-SOC PLOČA I ECLIPSE RAZVOJNO OKRUŽENJE ZA DS-5

## 2.1. DE1-SoC ploča

Na Slici 2.1. prikazana je DE1-SoC ploča na kojoj se izvršava tražena aplikacija. To je ciljna platforma.

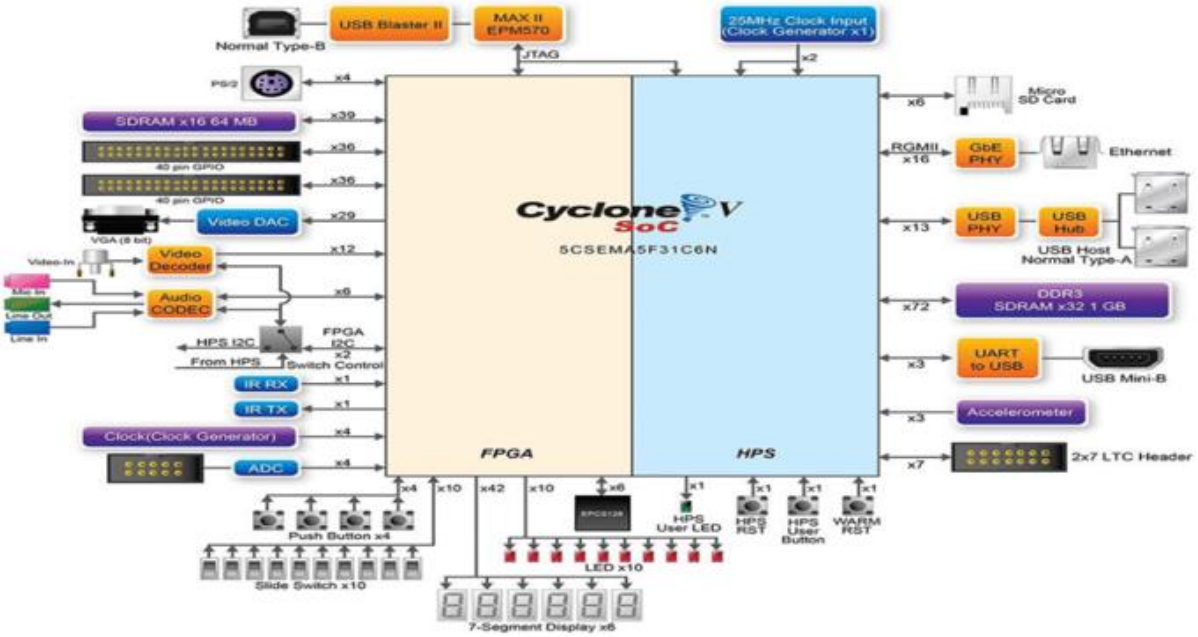



*Slika 2.1. Izgled DE1-SoC ploče*

Na Slici 2.2. prikazan je blok djagram *DE1-SoC* ploče. Ploča sadrži *Cyclone V* uređaj koji se sastoji od dvije različite komponente – FPGA i HPS (*Hard Processor System*). HPS je *hard logic* (soft procesor) mikroprocesorska jedinica (MPU) koja se sastoji od dvojezgarnog *ARM Cortex-A9* procesora, memorije na čipu, SDRAM-a, L3 interkonekcije i perifernih uređaja za podršku i interfejs. Mi koristimo HPS za izvršavanje softverskog dijela SoC[1] (*System on Chip - Sistem na čipu*) dizajna.

FPGA komponenta *Ciclone V* uređaja se sastoji od standardnih FPGA komponenti (LUT, CLB, PLL itd.), kontrolera zajedničke memorije i opštih perifernih uređaja koji se nalaze na standardnoj FPGA ploči. FPGA se koristi za prototip hardvera za *SoC* dizajn, primanje i slanje podataka do i od HPS-a koristeći AXI magistrale, mostove i *Avalon master-slave* uređaje.

[1] Sistem na čipu (*System-on-Chip - SoC*) je elektronski sistem ugrađen u jedinstveni čip. Svaki *SoC* sadrži najmanje jedan mikroprocesor. Da bi ispunili postavljene zahtjeve neki od *SoC*-ova imaju ugrađeno po dva ili tri mikroprocesora, a postoje i *SoC*-ovi koji koriste po više od desetinu procesora.




*Slika 2.2. Blok dijagram DE1-SoC ploče*

Na osnovu Slike 2.2. je očigledno da obe komponente (FPGA i HPS) imaju svoje pinove i periferne uređaje. Shodno tome, pinovi se ne dijele slobodno između HPS i FPGA komponenata i zato moraju biti dodijeljeni odgovarajućoj komponenti u različitim fazama projektovanja. Pored toga, važno je napomenuti da su HPS i FPGA povezani preko mostova (*bridge*-eva).

### *2.1.1. Resursi korišteni za projektni zadatak na DE1-SoC platformi*

Resursi koji su korišteni na *DE1-SoC* platformi prilikom izrade projektnog zadatka pored *Cyclone V* FPGA uređaja su i FPGA periferije: 10 prekidača (SW0 - SW9), 6 led dioda (LED0 - LED5), te šest 7-Segmentnih displeja (HEX0 - HEX5). Ove periferije su povezane sa HPS-om preko FPGA-HPS mosta.

U nastavku slijedi kratak opis svake od navedenih periferija:

- ***Prekidači*** predstavljaju ulaznu periferiju. U implementiranoj aplikaciji koristi se svih deset kliznih prekidača koji su dostupni na ploči i oni su predstavljeni periferijom SWITCHES_0. Prilikom čitanja vrijednosti prekidača najčešće se čita vrijednost svih deset prekidača odjednom, gdje je jedan prekidač predstavljen jednim bitom čije vrijednosti mogu biti 0 ili 1. Prvi prekidač sa desne strane predstavlja najniži bit, dok prvi prekidač sa lijeve strane predstavlja najviši bit. Vrijednost 0 predstavlja stanje kada je prekidač isključen (donja pozicija), a vrijednost 1 kada je prekidač uključen (gornja

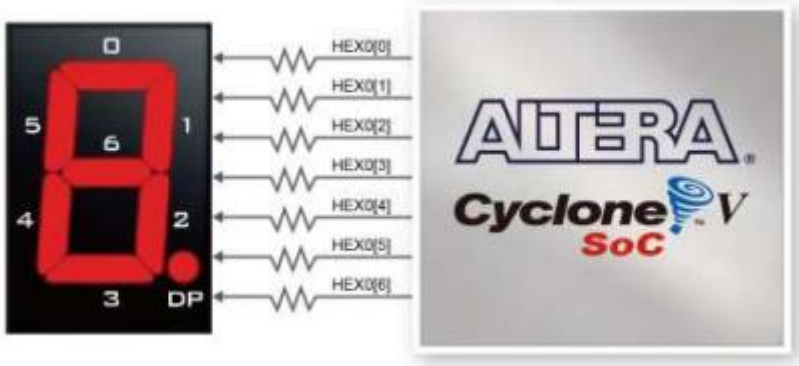
pozicija). Na Slici 2.3. prikazane su veze između 10 prekidača i *Cyclone V SoC* FPGA uređaja.


*Slika 2.3. Veze između prekidača i Cyclone V SoC FPGA uređaja*

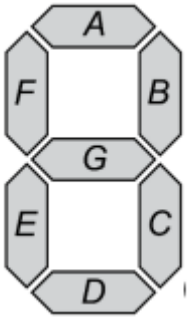
- ***Led diode*** predstavljaju izlazne periferije. U implementiranoj aplikaciji postoji deset led dioda koje su predstavljene periferijom LEDS_0, od kojih se koristi samo prvih (nižih) 6 za potrebe prikaza broja ”boja” koje se prikazuju na 7-segmentnim displejima. Prilikom upisa vrijednosti na led diode šalje se vrijednost za svih deset dioda gdje jedan bit predstavlja vrijednost za jednu diodu. Prva dioda sa desne strane je predstavljena najnižim bitom dok je prva dioda sa lijeve strane predstavljana najvišim. Ukoliko se upiše vrijednost 1 za jednu od dioda ta dioda će da se upali, dok upis vrijednosti 0 gasi diodu ako je već upaljena. Na Slici 2.4. prikazane su veze između 10 led dioda i *Cyclone V SoC* FPGA uređaja.
- ***7-Segmentni displeji*** predstavljaju drugu grupu izlaznih periferija i u implementiranoj aplikaciji se koristi svih šest displeja koji su dostupni na ploči. Koriste se za prikaz boja, vrijednosti tajmera, poruka o greškama itd. Imena displeja su u rasponu od HEX_0 do HEX_5. Na Slici 2.5. prikazane su veze između 7-Segmentnog displeja HEX0 i *Cyclone V SoC* FPGA uređaja. Za upis vrijednosti na displeje koriste se maske definisane u *header* fajlu

(jednobajtni broj u heksadecimalnom formatu) i to je način koji je najlakši za praćenje, jer su sve maske definisane na jednom mjestu. Za kreiranje maski bitno je znati raspored segmenata displeja u poređenju sa bitima 8-bitnog heksadecimalnog broja. Na Slici 2.6. je prikazan 7-Segmentni displej. Najniži bit (bit 0) 8-bitnog broja predstavlja segment A, bit do njega (bit 1) 8-bitnog broja predstavlja segment B i tako dalje do bita 6 koji predstavlja segment G. Bit 7, odnosno najviši bit je uvijek 0 jer ne predstavlja nijedan segment. Pojedinačni segmenti displeja se pobuđuju niskim naponskim nivoom, tj. oni su *active-low* tipa. To znači da kada se u neki od 7 nižih bita upiše 0 odgovarajući segment će biti upaljen, dok će za vrijednost 1 segment biti ugašen. Kako bi se na displeju npr. prikazao broj 0 svi segmenti sem segmenta G trebaju biti upaljeni, pa je vrijednost svih bita 0 sem za bit 6 koji treba biti 1.


*Slika 2.4. Veze između led dioda i Cyclone V SoC FPGA*



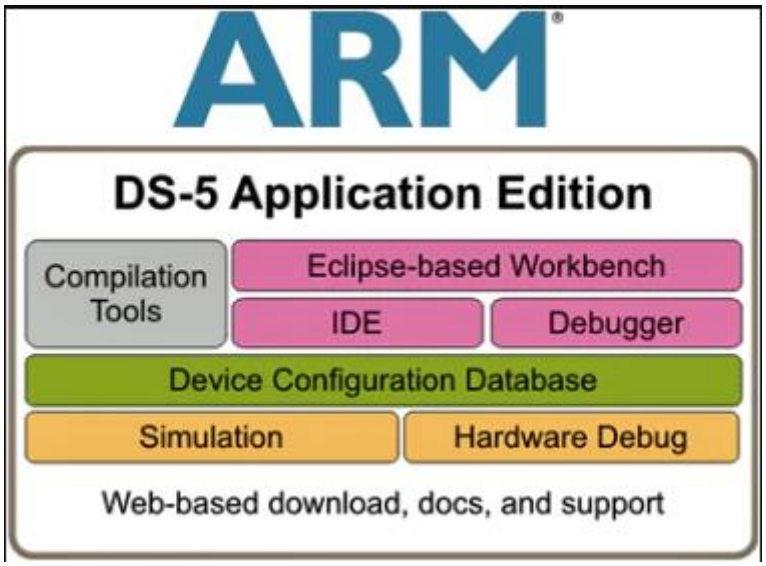

*Slika 2.5. Veze između 7-Segmentnog displeja HEX0 i Cyclone V SoC FPGA*




*Slika 2.6. Raspored segmenata 7-Segmentnog displeja*

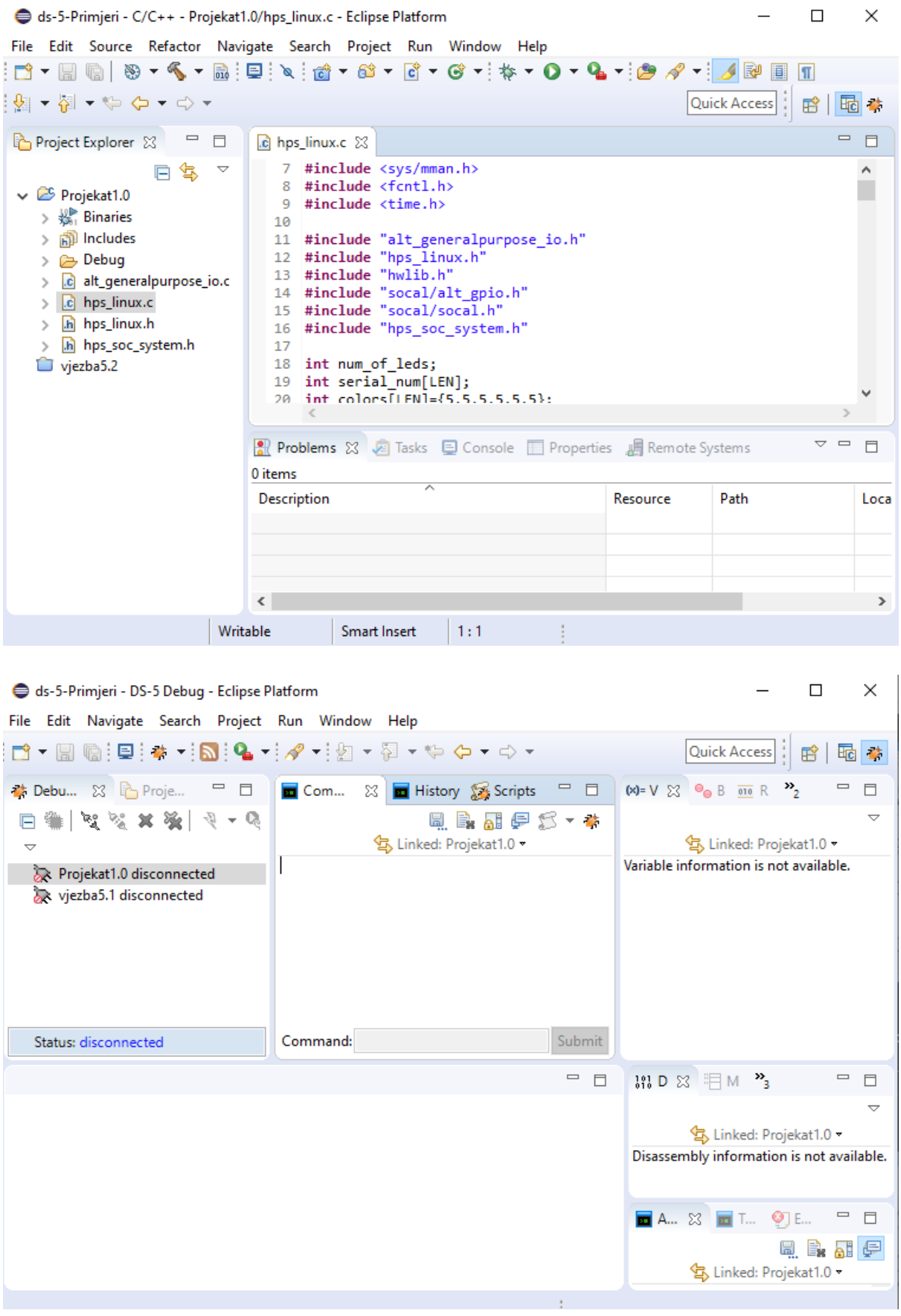
## 2.2. Eclipse razvojno okruženje za DS-5

*Eclipse* za *DS-5* je integrisano razvojno okruženje (IDE) koje kombinuje *Eclipse IDE* sa tehnologijom kompajliranja i otklanjanja grešaka (*debug*-ovanja) ARM alata. Takođe kombinuje GNU lanac alata (engl. *toolchain*) za ARM Linux ciljne platforme. Na Slici 2.7. prikazana je struktura razvojnog okruženja.




*Slika 2.7. Struktura DS-5 razvojnog okruženja*

Na Slici 2.8. prikazan je izgled *Eclipse* razvojnog okruženja za *DS-5*: C/C++ perspektiva (gore) i *Debug* perspektiva (dole).




*Slika 2.8. Izgled Eclipse razvojnog okruženja za DS-5*

# 3. IMPLEMENTACIJA APLIKACIJE

Implementacija aplikacije podrazumijeva pisanje programskog koda u C programskom jeziku u *Eclipse* razvojnom okruženju za DS-5. Prilikom predaje projektnog zadatka biće dostavljena 2 *header* fajla i 1 *source* fajl:

- ***hps_linux.h*** - u ovom fajlu su definisane globalne promjenljive za aplikaciju kao i pretprocesorske direktive koje predstavljaju konstante koje se koriste u aplikaciji i određene maske za čitanje podataka sa periferija ili upisivanje podataka na periferije (korištene maske će biti opisane u nastavku). Pored toga u ovom fajlu se nalaze deklaracije funkcija iz *source* fajla *hps_linux.c* i razni pokazivači.
- ***hps_linux.c*** – u ovom fajlu su definisane funkcije deklarisane u pomenutom *header* fajlu kao i *main* funkcija. Funkcije se mogu podijeliti na funkcije za mapiranje/oslobađanje memorije i funkcije koje predstavljaju algoritam aplikacije - implementacija manipulisanja prekidačima, implementacija tajmera, paljenje led diode, prikaz sadržaja na 7-segmentne displeje itd.
- ***hps_soc_system.h*** - fajl koji opisuje projektovani hardver na ploči. U ovom fajlu su opisane FPGA periferije koje su povezane sa HPS-om preko FPGA-HPS mosta. A te periferije su: prekidači, led diode, 7-segmentni displeji i tasteri.

## 3.1. Maske korištene za paljenje led dioda i postavljanje sadržaja 7-Segmentnih displeja

Maske za paljenje odgovarajućeg broja led dioda prikazane su u Tabeli 3.1.

Tabela 3.1. Maske za paljenje led dioda

| Maska | Značenje |
|---|---|
| 0b0000000111 | Maska za paljenje prve 3 led diode |
| 0b0000001111 | Maska za paljenje prve 4 led diode |
| 0b0000011111 | Maska za paljenje prvih 5 led dioda |
| 0b0000111111 | Maska za paljenje prvih 6 led dioda |
| 0b1111111111 | Maska za paljenje svih led dioda |

Maske za postavljanje sadržaja 7-Segmentnih displeja (u heksadecimalnom formatu) prikazane su u tabeli 3.2.

Tabela 3.2. Maske za postavljanje sadržaja 7-Segmentnih displeja

| HEX5 | HEX4 | HEX3 | HEX2 | HEX1 | HEX0 | ASCII reprezentacija | Značenje |
|---|---|---|---|---|---|---|---|
| 0x7F | 0x7F | 0x7F | 0x7F | 0x7F | 0x7F | XXXXXX[2] | Svi displeji ugašeni |
| 0x03 | 0x4F | 0x70 | 0x06 | 0x47 | 0x08 | bIJELA | Bijela boja |
| 0x24 | 0x41 | 0x07 | 0x08 | 0x7F | 0x7F | 2UtAXX | Zuta boja |
| 0x0C | 0x47 | 0x08 | 0x63 | 0x08 | 0x7F | PLAvAX | Plava boja |
| 0x46 | 0x2F | 0x63 | 0x06 | 0x2B | 0x08 | CrvEnA | Crvena boja |
| 0x46 | 0x2F | 0x2B | 0x08 | 0x7F | 0x7F | CrnAXX | Crna boja |
| 0x09 | 0x09 | 0x7F | 0x7F | 0x7F | 0x7F | HHXXXX | Jedna greška |
| 0x09 | 0x09 | 0x09 | 0x09 | 0x7F | 0x7F | HHHHXX | Dvije greške |
| 0x09 | 0x09 | 0x09 | 0x09 | 0x09 | 0x09 | HHHHHH | Tri greške |
| 0x0E | 0x08 | 0x4F | 0x47 | 0x06 | 0x21 | FAILEd | Nepravilan kraj aplikacije |
| 0x12 | 0x41 | 0x46 | 0x12 | 0x06 | 0x12 | 5UC5E5 | Uspješan kraj aplikacije |
| 0x40 | 0x40 | 0x40 | 0x40 | 0x18 | 0x18 | 000099 | Vrijednost tajmera |
| 0x78 | 0x24 | 0x12 | 0x18 | 0x19 | 0x79 | 725941 | Serijski broj |

Za svaki od potrebnih ispisa na 7-Segmentne displeje (navedenih u gornjoj tabeli) kreira se niz od 6 elemenata u koji se smiještaju odgovarajuće maske za taj ispis, na primjer za prikaz crne boje taj niz izgleda kao na Slici 3.1.

```
uint32_t hex_display_black[6] = {
        HEX_DISPLAY_C, HEX_DISPLAY_R, HEX_DISPLAY_N, HEX_DISPLAY_A,
        HEX_DISPLAY_CLEAR, HEX_DISPLAY_CLEAR
};
```

*Slika 3.1. Niz sa maskama za prikaz crne boje*

[2] X označava ugašen displej

## 3.2. Generisanje slučajnog broja dioda

Generisanje slučajnog broja dioda implementirano je u funkciji *generate_num_in_range()* (Slika 3.2.)

```
int generate_num_in_range(int lower, int upper)
{
    int number;
    srand(time(NULL));
    number = (rand() % (upper - lower +1)) + lower;
    return number;
}
```

*Slika 3.2. Implementacija funkcije generate_num_in_range()*

## 3.3. Implementacija algoritma manipulisanja prekicačima

Algoritam manipulisanja prekidačima implementiran je u funkciji *set_diode_cases().*

### *3.3.1. Slučaj 3 diode*

```
void set_diode_cases()
{
    switch (num_of_leds){
        case 3:
        {
            int red_counter=0;
            int blue_counter=0;
            int last_blue_position=6;
            int i;
            for(i=0;i<num_of_leds;i++)
            {
                if(colors[i]==3)
                    red_counter++;
                if(colors[i]==2)
                {
                    blue_counter++;
                    if(blue_counter>1)
                    {
                        last_blue_position=i;
                    }
                }
            }

            if(red_counter==1)
            {
                swich=1;
            }
            else if(colors[num_of_leds-1]==0)
            {
                swich=2;
            }
            else if(blue_counter>1)
            {
                swich=last_blue_position;
            }
            else
            {
                swich=2;
            }

            break;
        }
```

*Slika 3.3. Dio implementacije funkcije set_diode_cases() u slučaju kada su upaljene 3 diode*

### 3.3.2. *Slučaj 4 diode*

```
case 4:
{
    int red_counter=0;
    int blue_counter=0;
    int yellow_counter=0;
    int last_red_position=6;
    int i;
    for(i=0;i<num_of_leds;i++)
    {
       if(colors[i]==3)
       {
           red_counter++;
           if(red_counter>1)
               last_red_position=i;
       }
       if(colors[i]==2)
           blue_counter++;
       if(colors[i]==1)
           yellow_counter++;
    }

   if(red_counter>1 && (serial_num[LEN-1] % 2 != 0))
   {
       swich=last_red_position;
   }
   else if(red_counter==0 && colors[num_of_leds-1]==1)
   {
       swich=0;
   }
   else if(blue_counter==1)
   {
       swich=0;
   }
   else if(yellow_counter>1)
   {
       swich=3;
   }
   else
   {
       swich=1;
   }
       break;
}
```

*Slika 3.4. Dio implementacije funkcije set_diode_cases() u slučaju kada su upaljene 4 diode*

### *3.3.3. Slučaj 5 dioda*

```
case 5:
{
    int red_counter=0;
    int black_counter=0;
    int yellow_counter=0;
    int i;
    for(i=0;i<num_of_leds;i++)
    {
       if(colors[i]==3)
          red_counter++;
       if(colors[i]==4)
          black_counter++;
       if(colors[i]==1)
          yellow_counter++;
    }

    if(colors[num_of_leds-1]==4 && (serial_num[LEN-1] % 2 != 0))
    {
        swich=3;
    }
    else if(red_counter==1 && yellow_counter>1)
    {
        swich=0;
    }
    else if(black_counter==0)
    {
        swich=1;
    }
    else
    {
        swich=0;
    }
        break;
 }
```

*Slika 3.5. Dio implementacije funkcije set_diode_cases() u slučaju kada je upaljeno 5 dioda*

### *3.3.4. Slučaj 6 dioda*

```
case 6:
{
    int red_counter=0;
    int white_counter=0;
    int yellow_counter=0;
    int i;
    for(i=0;i<num_of_leds;i++)
    {
        if(colors[i]==3)
          red_counter++;
        if(colors[i]==0)
          white_counter++;
        if(colors[i]==1)
          yellow_counter++;
    }
    if(yellow_counter==0 && (serial_num[LEN-1] % 2 != 0))
    {
        swich=2;
    }
    else if(yellow_counter==1 && white_counter>1)
    {
        swich=3;
    }
    else if(red_counter==0)
    {
        swich=5;
    }
    else
    {
        swich=3;
    }
        break;
}
```

*Slika 3.6. Dio implementacije funkcije set_diode_cases() u slučaju kada je upaljeno 6 dioda*

## 3.4. Implementacija tajmera

Tajmer je implementiran korištenjem *usleep()* funkcije i internog brojača.

```
void handle_timer()
{
    if(timer_counter==0)
    {
        display_failed();
        end_of_application = 1;
    }
    else
    {
        timer_counter--;
    }
}
```

*Slika 3.7. Funkcija kojom je implementiran tajmer*

### 3.5. Paljenje led dioda

Paljenje odgovarajućeg broja led dioda pri *start*-anju aplikacije na ploči postignuto je funkcijom *setup_leds()* (Slika 3.8.)

```
void setup_leds()
{

    switch (num_of_leds){
        case 3:
            alt_write_word(fpga_leds, THREE_LEDS);
            break;

        case 4:
            alt_write_word(fpga_leds, FOUR_LEDS);
            break;

        case 5:
            alt_write_word(fpga_leds, FIVE_LEDS);
            break;

        case 6:
            alt_write_word(fpga_leds, SIX_LEDS);
            break;

        default:
            alt_write_word(fpga_leds, LEDS_CLEAR);
    }
}
```

*Slika 3.8. Implementacija funkcije setup_leds()*

### 3.6. Provjera ispravnosti prebačenog prekidača

Provjera ispravnosti prebačenog prekidača vrši se u finkciji *check_switches()*.

```
void check_switches()
{
    switch_value_new = fpga_swiches_position() & 0b111111;

    if(switch_value_old == switch_value_new)
    {
        return;
    }

    switch_value_old = switch_value_new;

    if(switch_value_new == 0)
    {
        return;
    }

    switch(num_of_leds)
    {
    case 3:
    {
        if(swich==1)
        {
            if(switch_value_new!=2)
                num_of_errors++;

        }

        else if(swich==2)
        {
            if(switch_value_new!=4)
                num_of_errors++;
        }
            break;
      }
```

*Slika 3.9. Implementacija funkcije check_switches() – prvi dio*

```
case 4:
{
    if(swich==0)
    {
        if(switch_value_new!=1)
            num_of_errors++;
    }
    if(swich==1)
    {
        if(switch_value_new!=2)
            num_of_errors++;
    }
    else if(swich==2)
    {
        if(switch_value_new!=4)
            num_of_errors++;
    }
    else if(swich==3)
    {
        if(switch_value_new!=8)
            num_of_errors++;
    }
        break;
  }
case 5:
{
    if(swich==0)
    {
        if(switch_value_new!=1)
            num_of_errors++;
    }
    if(swich==1)
    {
        if(switch_value_new!=2)
            num_of_errors++;
    }
    else if(swich==3)
    {
        if(switch_value_new!=8)
            num_of_errors++;
    }
        break;
  }
```

*Slika 3.10. Implementacija funkcije check_switches() – drugi dio*

```
            case 6:
            {
                if(swich==2)
                {
                    if(switch_value_new!=4)
                        num_of_errors++;
                }
                else if(swich==3)
                {
                    if(switch_value_new!=8)
                        num_of_errors++;
                }
                else if(swich==5)
                {
                    if(switch_value_new!=32)
                        num_of_errors++;
                }
                    break;
              }
            }
        }
```

*Slika 3.11. Implementacija funkcije check_switches() – treći dio*

### 3.7. Postavljanje sadržaja 7-Segmentnih displeja

Kao primjer postavljanja sadržaja na 7-Segmentne displeje, navode se dvije funkcije:

- ***Prikaz boje*** – boje koje se mogu prikazati su : bijela, crna, žuta, crvena i plava. Na Slici 3.12. prikazana je fukcija za ispis crvene boje na displeju, a pored te funkcije postoje fukcije za ispis svake pomenute boje.

```
void display_red()
{
    uint32_t hex_display_index;

    for (hex_display_index = 0; hex_display_index < HEX_DISPLAY_COUNT; hex_display_index++)
    {
        uint32_t hex_value_to_write = hex_display_red[HEX_DISPLAY_COUNT - hex_display_index-1];

        alt_write_word(fpga_hex_displays[hex_display_index], hex_value_to_write);
    }
}
```

*Slika 3.12. Funkcija za prikaz crvene boje na 7-segmentnom displeju*

- ***Prikaz vrijednosti tajmera***

```
void display_timer()
{
    char current_char[2] = " \0";
    char hex_counter_hex_string[HEX_DISPLAY_COUNT + 1];

    /* Get hex string representation of input value on HEX_DISPLAY_COUNT 7-segment displays */
    snprintf(hex_counter_hex_string, HEX_DISPLAY_COUNT + 1, "%0*d", HEX_DISPLAY_COUNT, (unsigned int)timer_counter);

    uint32_t hex_display_index;
    for (hex_display_index = 0; hex_display_index < HEX_DISPLAY_COUNT; hex_display_index++)
    {
        current_char[0] = hex_counter_hex_string[HEX_DISPLAY_COUNT - hex_display_index - 1];

        /* Get decimal representation for this 7-segment display */
        uint32_t number = (uint32_t)strtol(current_char, NULL, 16);

        /* Use lookup table to find active-low value representing number on the 7-segment display */
        uint32_t hex_value_to_write = hex_display_table[number];

        alt_write_word(fpga_hex_displays[hex_display_index], hex_value_to_write);
    }
}
```

*Slika 3.13. Funkcija za prikaz vrijednosti tajmera na 7-segmentnom displeju*

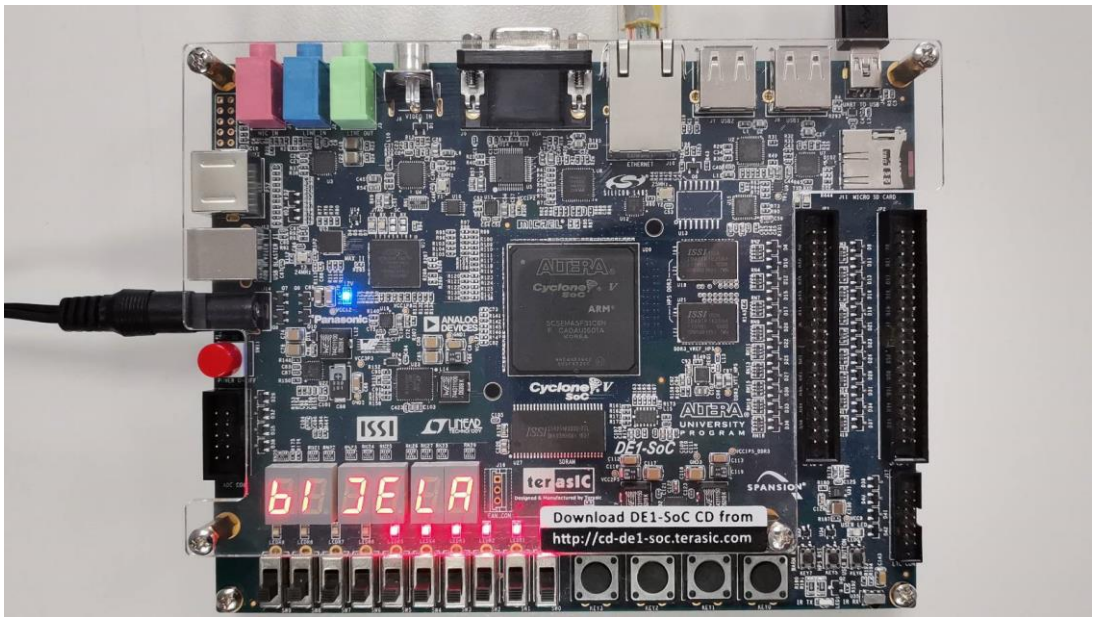
Ostali ispisi se rade analogno.

# 4. REZULTAT IZVRŠAVANJA

## 4.1. Prikaz boja na 7 – Segmentnim displejima

### *4.1.1. Prikaz bijele boje*

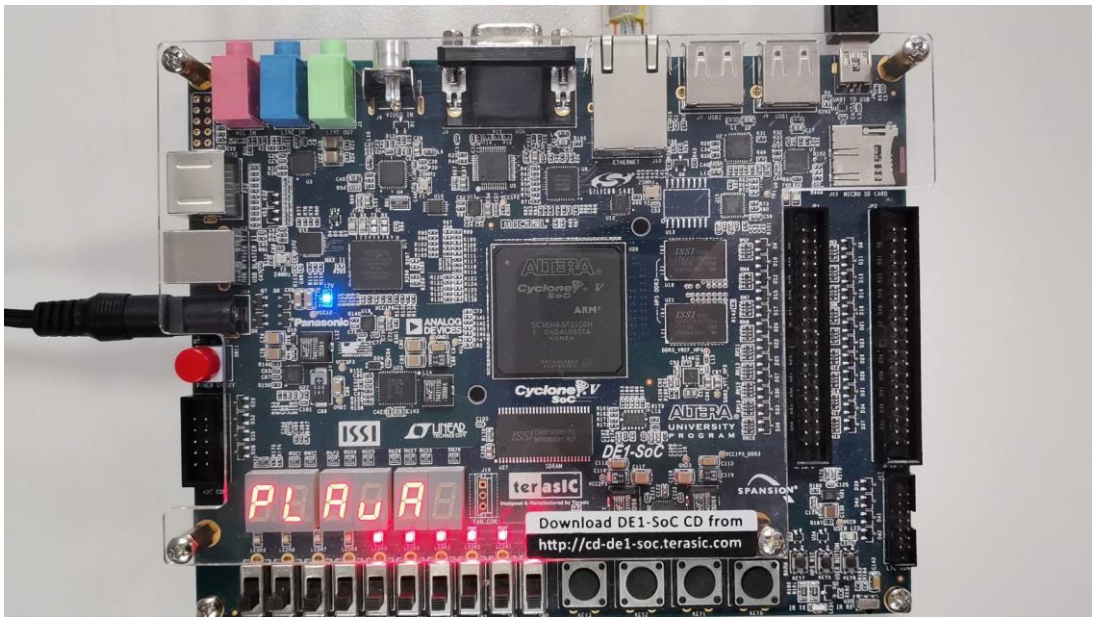
Na Slici 4.1. je prikazan ispis bijele boje na 7 – Segmentnim displejima.




*Slika 4.1. Prikaz bijele boje*

### *4.1.2. Prikaz plave boje*

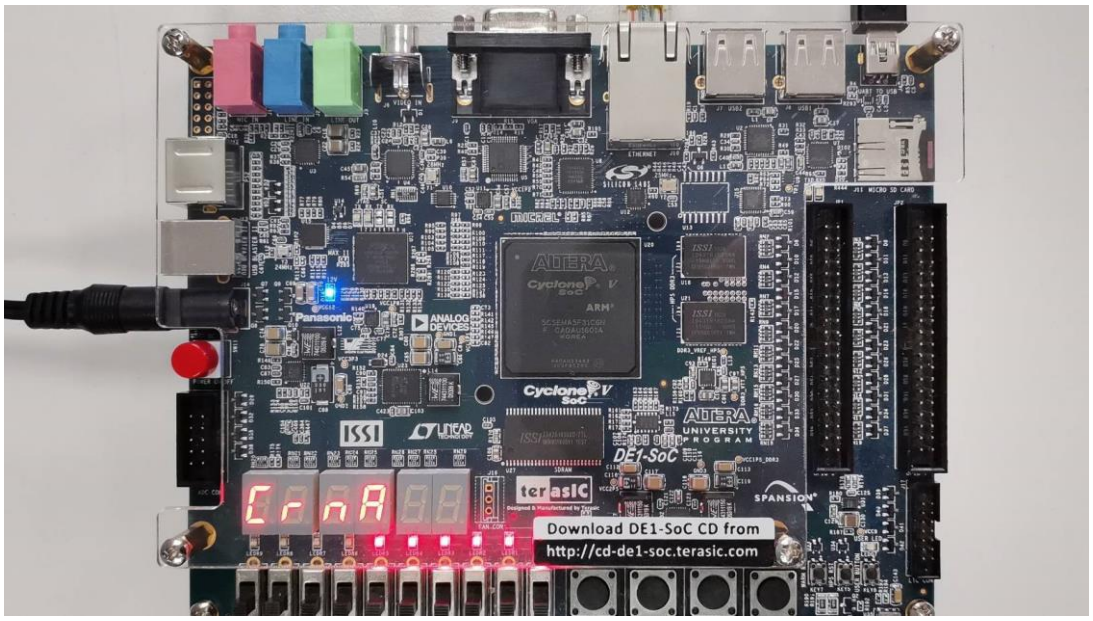
Na Slici 4.2. je prikazan ispis plave boje na 7 – Segementnim displejima.




Slika 4.2. Prikaz plave boje

### *4.1.3. Prikaz crne boje*

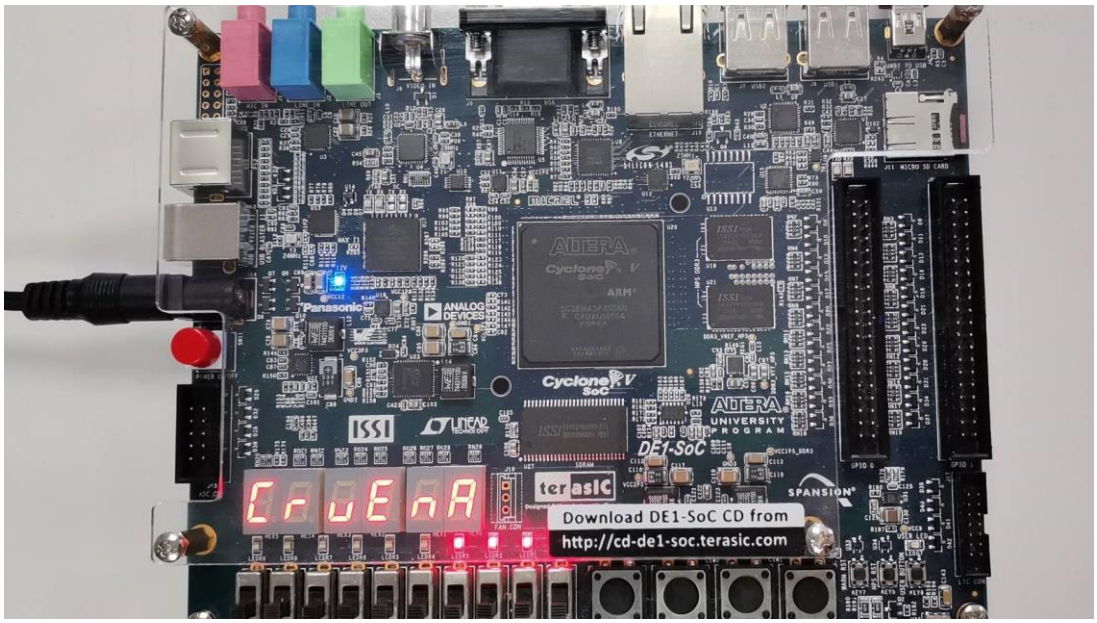
Na Slici 4.3. je prikazan ispis crne boje na 7 – Segmentnim displejima.




*Slika 4.3. Prikaz crne boje*

### *4.1.4. Prikaz crvene boje*

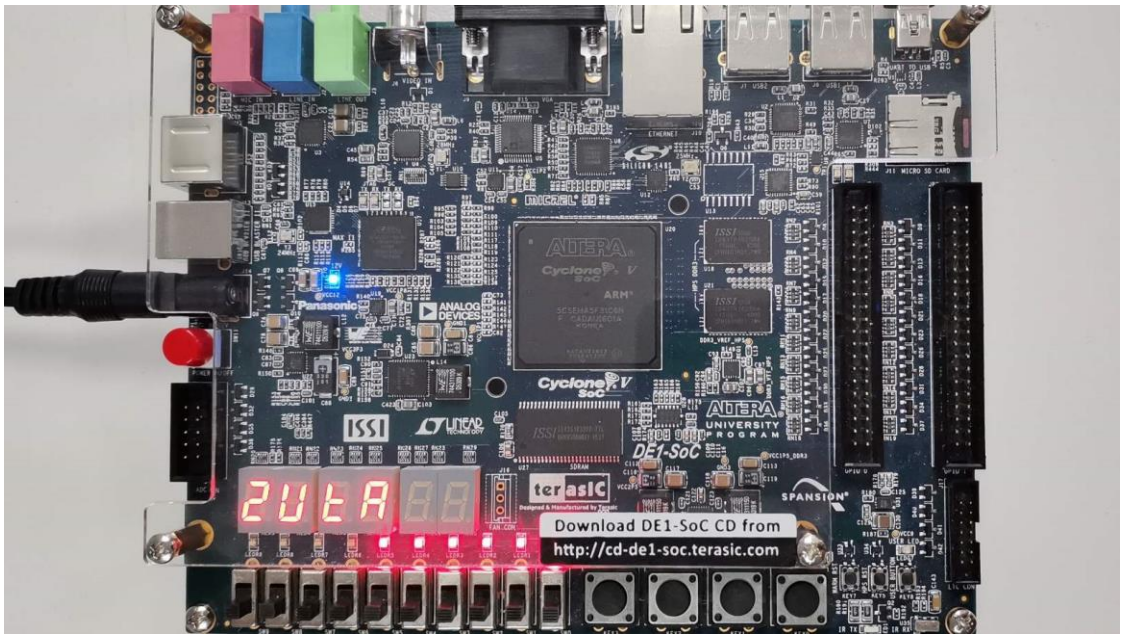
Na Slici 4.4. je prikazan ispis crvene boje na 7 – Segmentnim displejima.




*Slika 4.4. Prikaz crvene boje*

### *4.1.5. Prikaz žute boje*

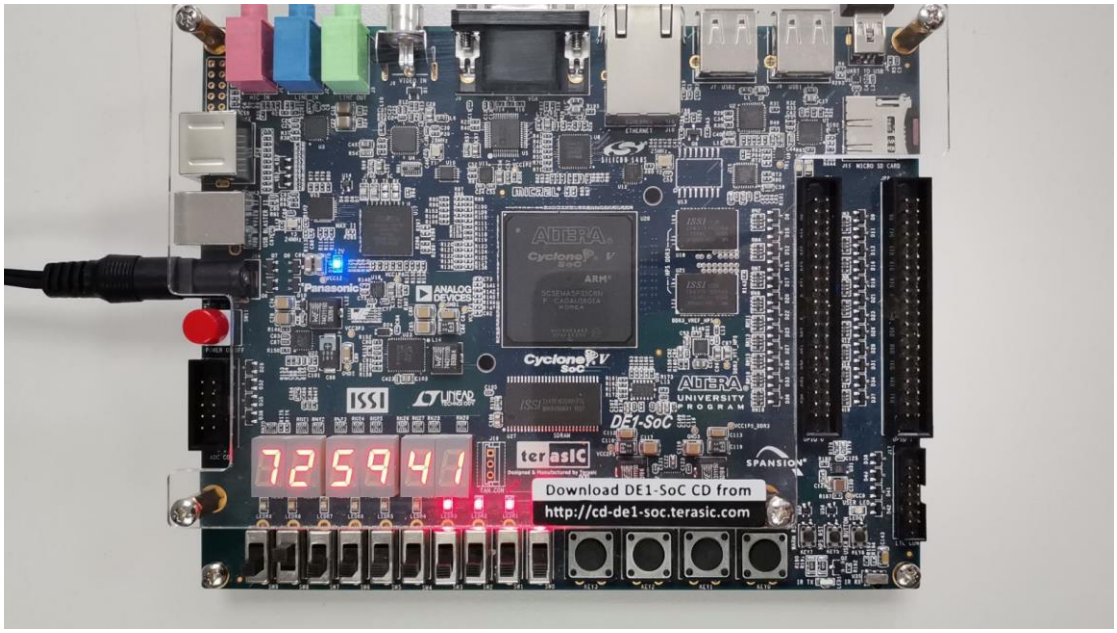
Na Slici 4.5. je prikazan ispis žute boje na 7 – Segmentnim displejima.




*Slika 4.5. Prikaz žute boje*

## 4.2. Prikaz serijskog broja

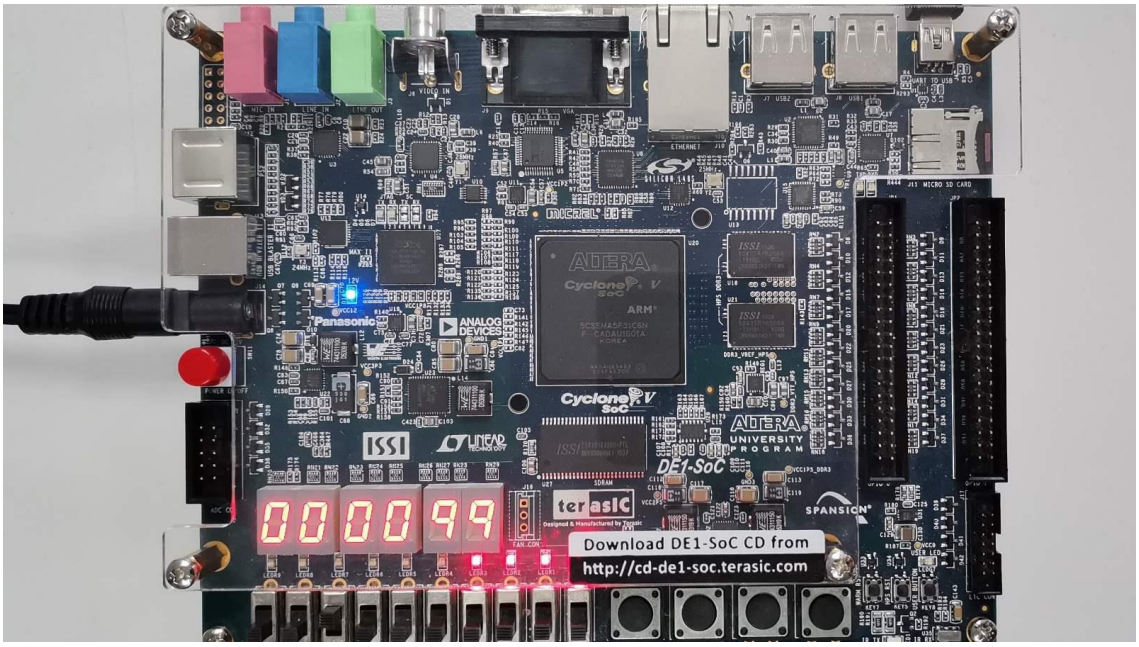
Na Slici 4.6. prikazan je ispis serijskog broja na 7 – Segmentnim displejima.




*Slika 4.6. Prikaz serijskog broja*

## 4.3. Prikaz vrijednosti tajmera

Na Slici 4.7. prikazan je ispis vrijednosti tajmera kada je odbrojavanje tajmera stiglo do 99.

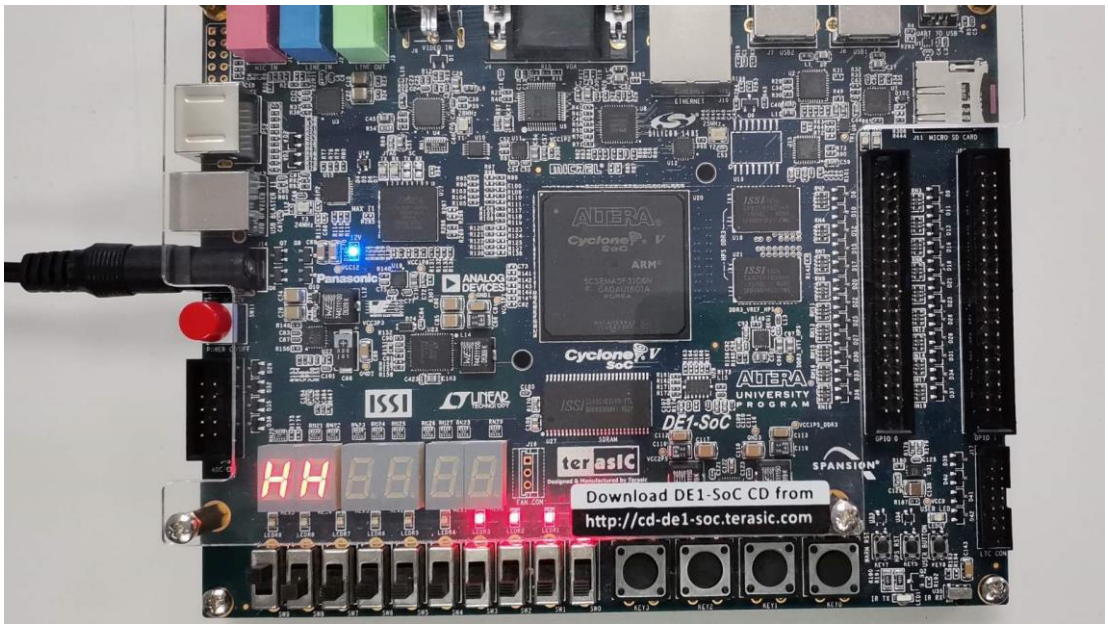



*Slika 4.7. Prikaz vrijednosti tajmera*

## 4.4. Prikaz poruka o greškama

### *4.4.1. Prikaz poruke u slučaju jedne greške*

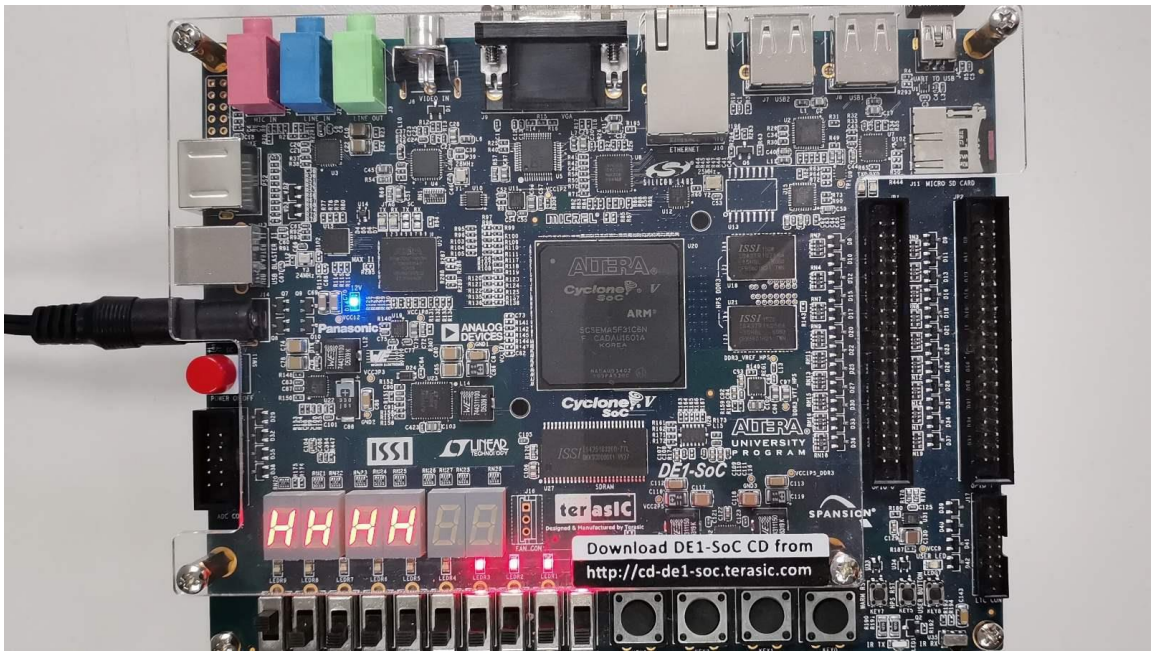
Na Slici 4.8. prikazan je ispis poruke o jednoj grešci.




*Slika 4.8. Prikaz poruke o jednoj grešci*

### 4.4.2. *Prikaz poruke u slučaju dvije greške*

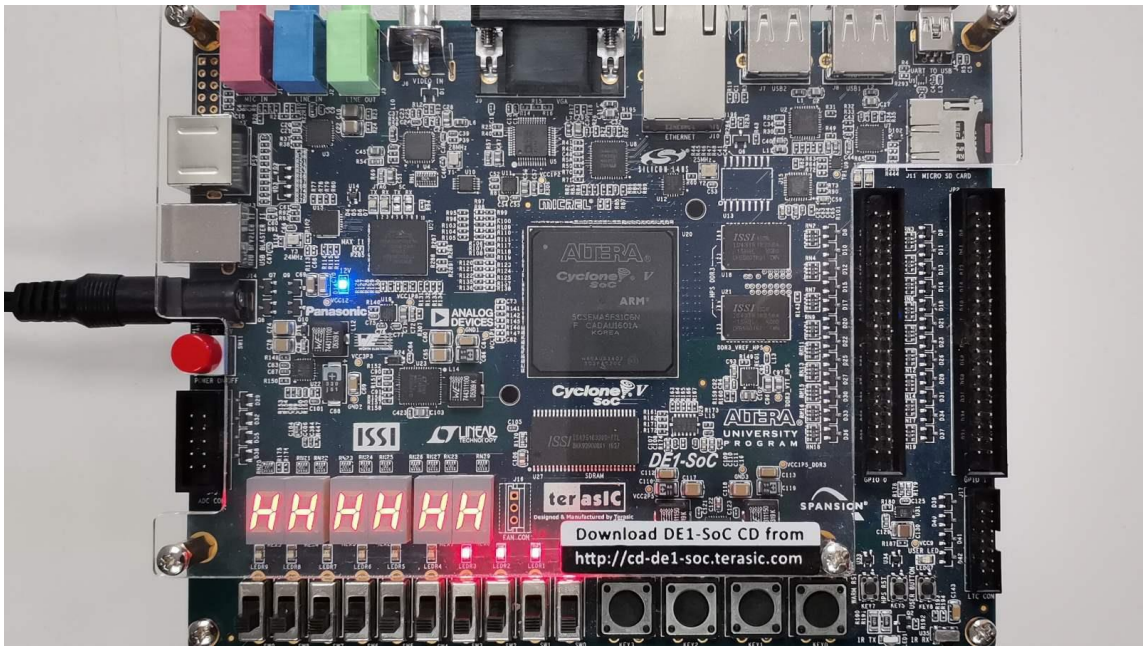
Na Slici 4.9. prikazan je ispis poruke o dvije greške.




*Slika 4.9. Prikaz poruke o dvije greške*

### 4.4.3. *Prikaz poruke u slučaju tri greške*

Na Slici 4.10. prikazan je ispis poruke o tri greške.

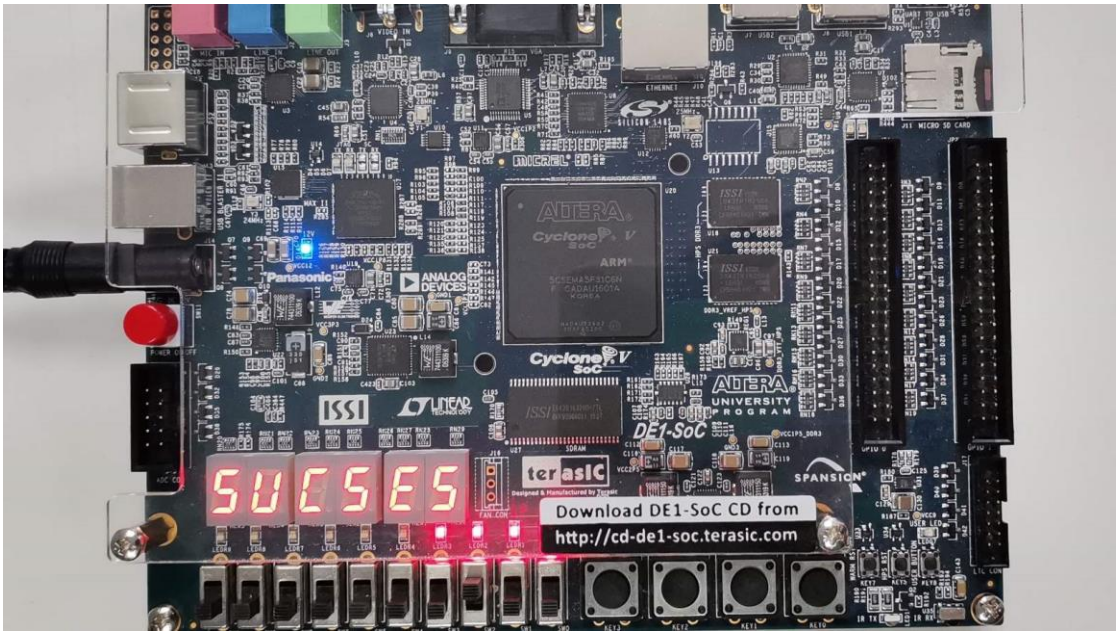



*Slika 4.10. Prikaz poruke o tri greške*

## 4.5. Prikaz poruka o statusu završetka aplikacije

### *4.5.1. Prikaz poruke u slučaju uspješnog završetka aplikacije*

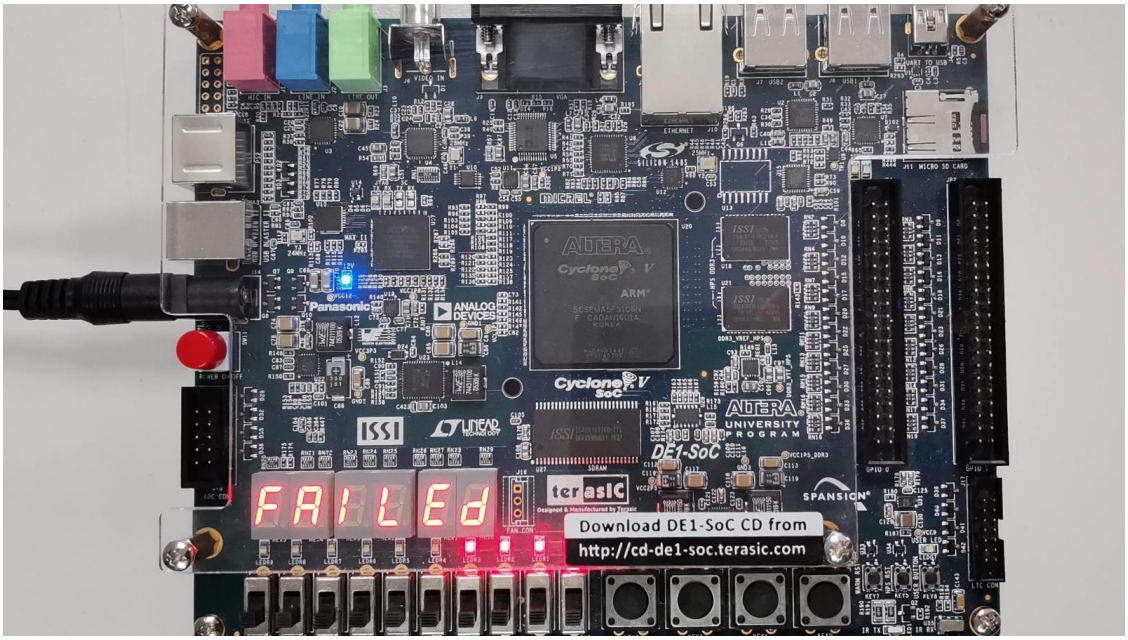
Na Slici 4.11. je prikazan ispis poruke o uspješnom završetku aplikacije.




*Slika 4.11. Prikaz poruke o uspješnom završetku aplikacije*

### *4.5.2. Prikaz poruke u slučaju neželjenog terminiranja aplikacije*

Na Slici 4.12. je prikazan ispis poruke u slučaju neželjenog terminiranja aplikacije.




*Slika 4.12. Prikaz poruke o neželjenom terminiranju aplikacije*

## 5. LITERATURA


1. *Zlatko Bundalo, Sistem na čipu – SoC*
2. *DE1-SOC User Manual:* http://www.ee.ic.ac.uk/pcheung/teaching/ee2_digital/DE1-SoC_User_manual.pdf
3. *Eclipse razvojno okruženje za DS*-5: https://developer.arm.com/documentation/dui0478/a/ds-5-product-overview/eclipse-for-ds-5