

**PROJEKAT IZ PROJEKTOVANJA SLOŽENIH**

**DIGITALNIH SISTEMA I FUNKCIONALNE**

**VERFIKACIJE HARDVERA**

Naziv projekta:

Drajver za “MP3 decoder” algoritam.

Tekst zadatka:

1. Koristeći alat Petalinux formirati SD karticu i generisati neophodne fajlove za podizanje Linux operativnog sistema na Zybo ploči..
2. Napraviti drajver koji softverski simulira pristup svim memorijski mapiranim hardverskim modulima neophodnim za akceleraciju odabranog algoritma.
3. Napisati aplikaciju koja komunicira sa drajverom, te proverava njegovu funkcionalnost.

Mentor projekta:

Nebojša Pilipović

Projekat izradili:

Miloš Nedeljković, EE 234/2018

Vasilije Batas, EE 190/2018

Almen Brenoli, EE 109/2018

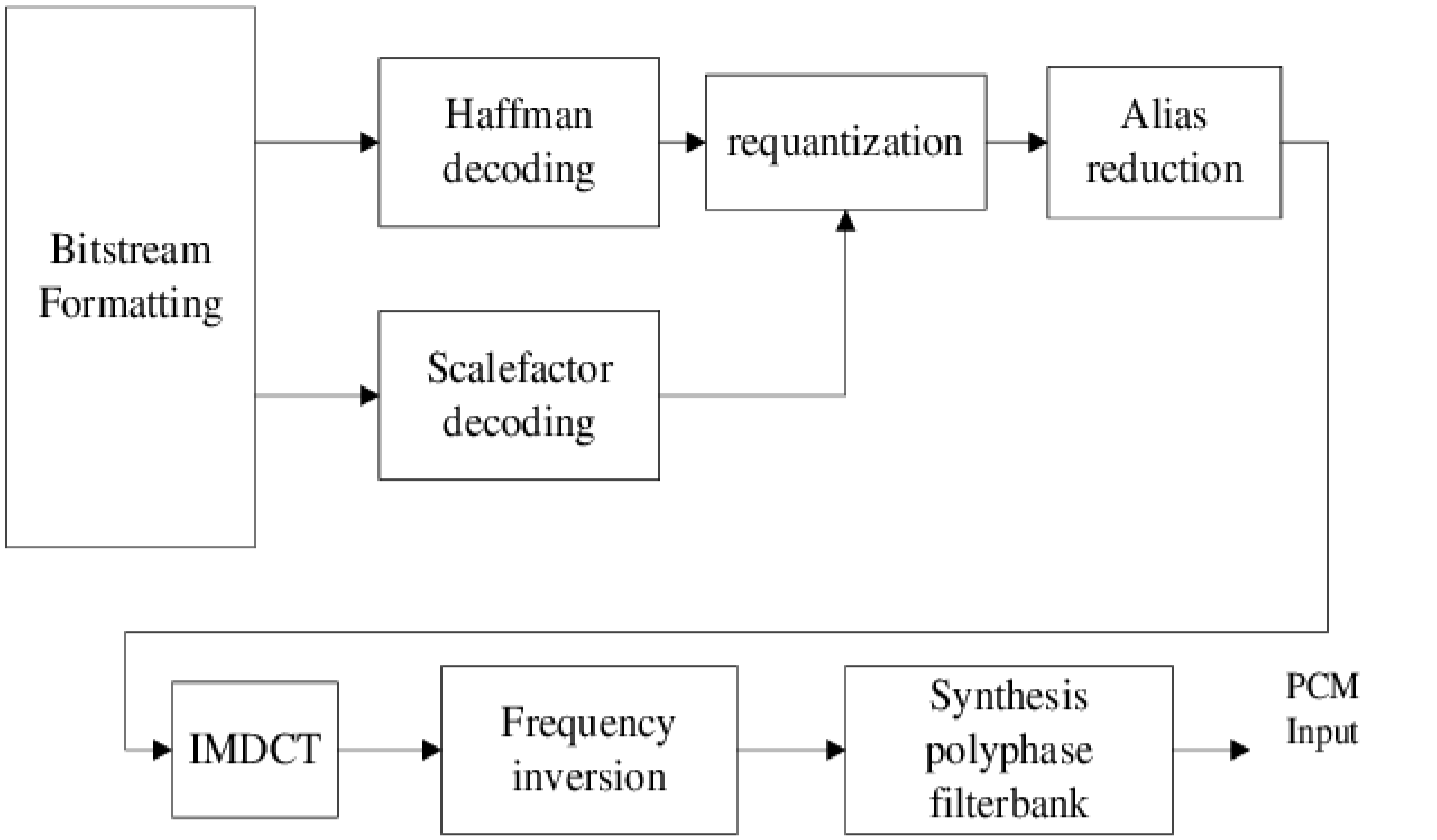
# Uvod

U današnjem digitalnom dobu, reprodukcija i kompresija zvuka postali su ključni aspekti u mnogim aplikacijama koje se kreću od muzike i filmske industrije do telekomunikacija i multimedijalnih uređaja. Jedan od najznačajnijih formata za kompresiju zvuka je MP3 (MPEG Audio Layer III), koji omogućava efikasno skladištenje i prenos audio sadržaja uz minimalan gubitak kvaliteta. MP3 je postao standardni format za digitalnu distribuciju muzike i drugih zvučnih sadržaja.

MP3 kompresija postiže se kroz eliminaciju suvišnih informacija iz audio signala, čime se smanjuje veličina datoteke bez narušavanja percepcije ljudskog sluha. Proces dekompresije MP3 datoteka je esencijalan kako bi se originalni audio sadržaj povratio u obliku razumljivom ljudskom uhu. Ovaj postupak zahteva izuzetno visoke računske performanse, posebno kada se radi o reprodukciji u realnom vremenu.

Hardverska implementacija *MP3 decoder* algoritma pruža rešenje za efikasno i brzo dekodiranje MP3 datoteka, omogućavajući kvalitetno slušanje muzike ili reprodukciju zvuka na raznim uređajima, uključujući mobilne telefone, audio plejere, televizore i druge multimedijalne platforme.

Na slici 1.1 se može videti blok šema *MP3 decoder* algoritma



*Slika 1.1 Blok dijagram MP3 decoder algoritma*

Nakon profajliranja početnog *MP3 decoder* algoritma napisanog u jeziku C++ dobijeni su rezultati da je funkcija koja troši najviše resursa tokom izvršavanja algoritma: *Inverese Modified Discrete Cosine Transformation* (IMDCT). Na osnovu dobijenih rezultata odlučeno je da treba taj deo koda hardverski ubrzati kako bi dobili bolje performanse sistema.

Za hardversku implementaciju sistema korišćen je razvojni sistem Zybo Z7-10 i na njegovoj programabilnoj logici se hardverski ubrzava funkcija IMDCT, dok se ostatak koda izvršava softverski na ARM-ovom Cortex A9 procesoru.

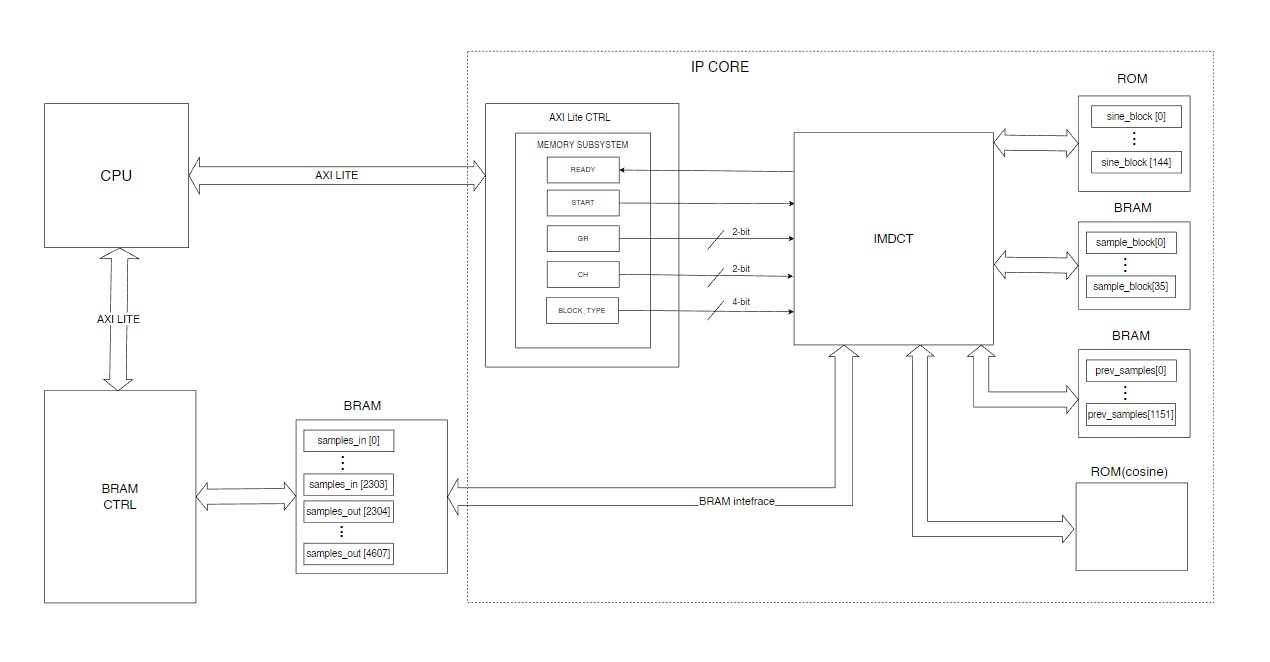
# Specifikacija sistema

Celokupan sistem sastoji se od AXI Lite kontrolera, memorijskog podsistema, IMDCT funkcije, tri BRAM ćelije, dve ROM ćelije i jednim BRAM kontrolerom.

Preko AXI Lite interfejsa se konfigurišu registri memorijskog podsistema. U registre se upisuju vrednosti neophodne za ispravan radi IP jezgra kao i start signal kojim započinje izvršavanje IMDCT funkcije, dok se informacije o završetku izvršavanje funkcije dobija čitanjem iz statusnog registra READY.

Ulazni semplovi u sistem koji treba da budu obrađeni se šalju preko AXI Lite interfejsa, a kasnije preko BRAM kontrolera se upisuju u BRAM koji se nalazi izvan IP jezgra. Kada sistem završi obradu semplova on ih odmah zatim upisuje na nove lokacije u istom BRAM-u a zatim se oni pomoću istog AXI Lite interfejska čitaju kako bi softverski deo koda mogao nesmetano da nastavi sa izvršavanjem.

Ostale BRAM i ROM ćelije služe za skladištenje promenljivih koje koristi IMDCTfunkcija, detaljniji rad IP jezgra biće kasnije razmatran.



*Slika 2.1 Blok dizajn sistema*

# Opis funkcionalnosti komponenti sistema

## IMDCT modul

IMDCT (Inverse Modified Discrete Cosine Transform) je ključna komponenta u *MP3 dekoder* algoritmu, odgovorna za transformaciju frekvencijskih podataka nazad u vremenski domen. Ova transformacija omogućava rekonstrukciju audio signala u njegovu originalnu formu.

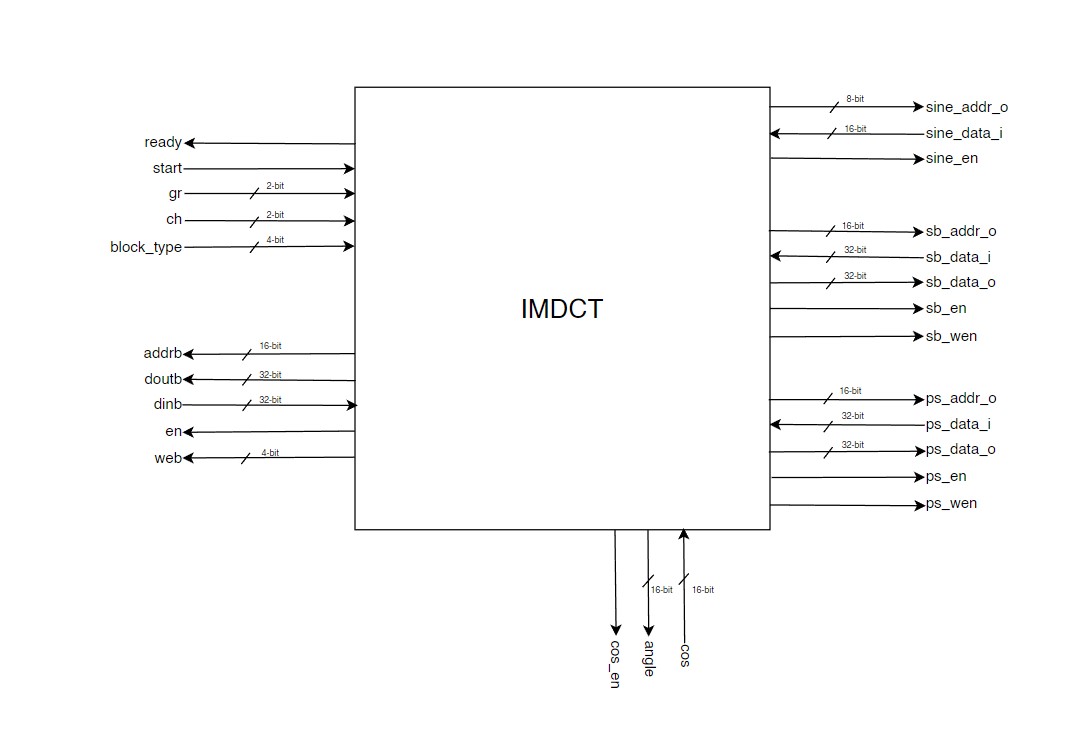
Koristi kako bi se obrnula operacija koju je izvršila MDCT (Modified Discrete Cosine Transform) na ulaznom audio signalu. MDCT je korišćena tokom kompresije u MP3 formatu kako bi se audio signal podelio u manje blokove i transformisao iz vremenskog domena u frekvencijski domen. Svaki blok se zatim kvantizuje i kodira.

Princip rada IMDCT-a je zasnovan na kosinusnoj transformaciji. Blokovi frekvencijskih podataka, dobijeni iz MDCT-a, se kombinuju sa odgovarajućim kosinusnim talasima kako bi se rekonstruisali vremenski signali. Ovaj proces se radi za svaki vremenski trenutak unutar bloka, čime se dobija vremenski signal u vremenskom domenu.

Na primer, zamislimo da imamo blok frekvencijskih podataka koji se odnose na određeni vremenski segment audio signala. IMDCT će kombinovati ove podatke sa odgovarajućim kosinusnim talasima različitih frekvencija kako bi se dobio rekonstruisani audio signal za taj segment. Ovaj rekonstruisani signal će zatim biti spojen sa ostalim segmentima da bi se dobio kompletni audio signal.

U ovom projektu IMDCT modul je projektovan kao jedna celina a promenljive koje se koriste unutar IMDCT funkcije su smestene u BRAM-ove sa kojima IMDCT modul komunicira preko BRAM interfejsa tokom izvršavanja.

Na slici 3.1.1 se mogu videti svi interfejsi koje IMDCT modul koristi kako bi komunicirao sa ostalim modulima unutar IP jezgra. Tu se nalazi interfejs za komunikaciju sa AXI Lite kontrolerom koji dalje komunicira sa softversim delom algoritma. A ostali intefejsi se koriste za komunkaciju sa BRAM i ROM memorijama, koje će biti detaljnije razmatrene u narednim poglavljima.



*Slika 3.1.1 imdct modul*

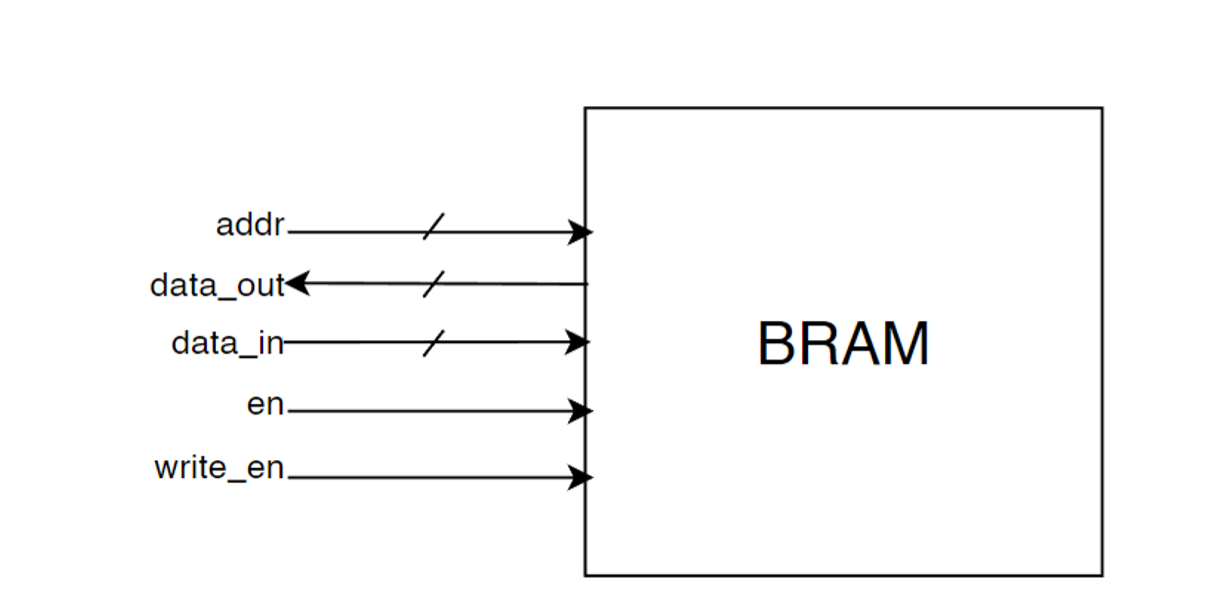
## Blok RAM

Blok RAM (BRAM) je važan resurs unutar programabilnih logičkih uređaja poput FPGA (Field-Programmable Gate Array) i SoC (System-on-Chip) čipova. To je vrsta brze i lokalne memorije koja omogućava efikasno skladištenje podataka u blizini logičkih blokova, što dovodi do ubrzanja operacija čitanja i pisanja u odnosu na udaljenije i sporije spoljne memorije.

BRAM se sastoji od niza memorijskih ćelija organizovanih u blokove. Svaki blok obično sadrži nekoliko hiljada memorijskih lokacija, a svaka lokacija može skladištiti višebitne podatke. BRAM se koristi za skladištenje različitih vrsta podataka kao što su konfiguracijske informacije, tablice za brzu obradu podataka ili privremeni rezultati izračunavanja.

Interfejsi za upis i čitanje su dva osnovna načina na koja se pristupa podacima unutar BRAM-a. Za operaciju upisa, korisnik šalje adresu memorijske lokacije koju želi da ažurira, kao i podatke koje želi da upiše. Ovi podaci se zatim beleže u odgovarajućoj memorijskoj ćeliji. Operacija čitanja zahteva slanje adrese željene lokacije, nakon čega BRAM vraća podatke sa te adrese. Bitno je napomenuti da signal *enable* treba postaviti na visok logički nivo kada se rade ove operacije, takođe postoji i signal *write enable* kako bi sistem znao kada da čita iz memorije a kada da upisuje. Primer jednog BRAM-a sa pomenutim interfejsom je dat na slici 3.2.1

U ovom projektu su iskorišćena tri modula BRAM-a, od kojih su dva ručno projektovanja za skladištenje promenljivih *sample\_block* i *prev\_samples* dok je jednan BRAM modul generisan pomoću Vivado IP integratora za skladištenje ulaznih i izlaznih semplova koje IP jezgro obrađuje. Što se može videti na slici 2.1.

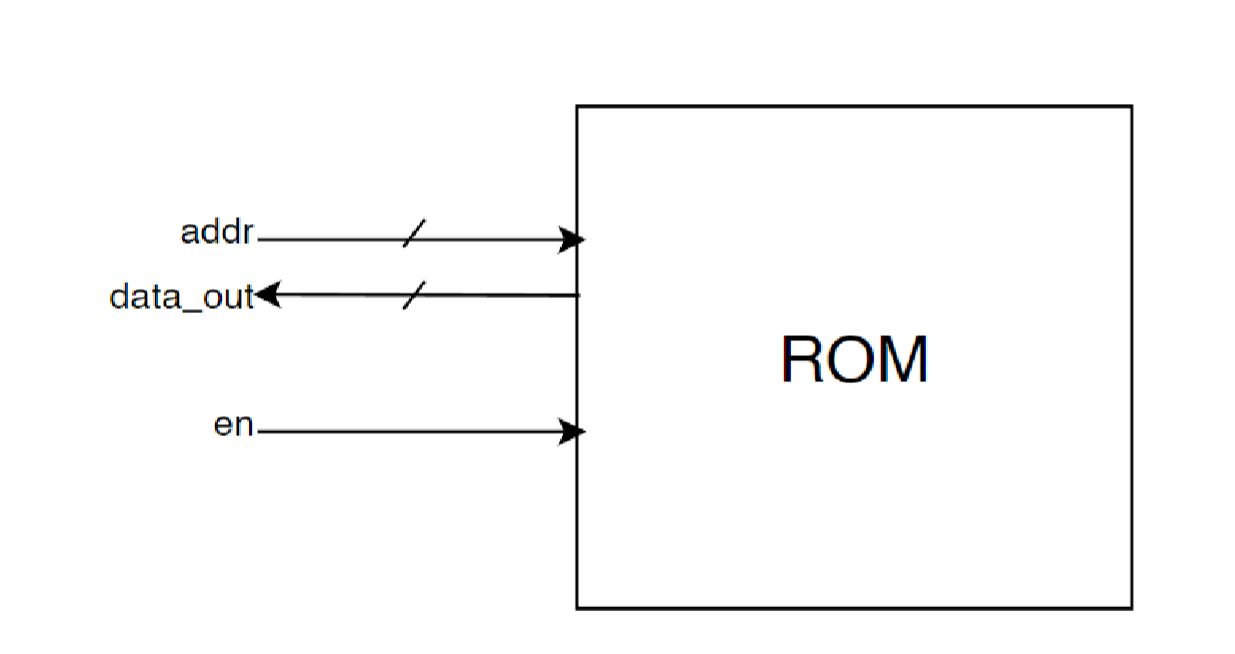


*Slika 3.2.1 Blok RAM interfejs*

## ROM

Za potrebe ovog projekta ROM je realizovan kao prethodno opisani BRAM moduli sa razlikom u tome što ne postoji interfejs za upis u memoriju već samo za čitanje, a lokacije u memoriji su popunjene inicijalnim vrednostima tako da odgovaraju potrebama IMDCT modula.

U sistemu imamo dva ROM-a koji su iskorišćeni za sladištenje *sine\_block* promenljive kao i za skladištenje vrednosti kosinus funckije za svih 360 stepeni sa rezolucijom od jednog stepena. Primer ROM-a je dat na slici 3.3.1.



*Slika 3.3.1 ROM interfejs*

# Opis Linux drajvera

Drajveri imaju specijalnu ulogu u Linuksovom jezgru (kernelu). Oni predstavljaju softverski sloj koji leži između aplikacije i hardvera. Drajveri predstavljaju spregu između hardvera i softvera (aplikacije), i olakšavaju komunikaciju između njih.

Osnovne funkcionalnosti funkcija implementiranih u drajveru :

• imdct\_probe () - Probe funkcija se poziva pri prepoznavanju uređaja koji sadrži isto compatible polje kao što je navedeno u modulu. U ovom slučaju ona će biti pozvana dva puta - jednom za BRAM i jednom za IP jezgro. Realizovana je tako da se prvo proverava za koji od dva uređaja se poziva (čita se compatible polje platformskog uređaja koji ju poziva), te pomoću switch petlje

se obavlja potrebna operacija za svaki od uređaja. Za oba uređaja se alocira prostor fizičke

memorije i vrši se mapiranje fizičke memorije na virtualne memorijske lokacije.

• imdct\_open () - funkcija za otvaranje fajla u kojeg se upisuju ili čitaju podaci

• imdct\_close () - funkcija za zatvaranje fajla u kojeg se upisuju ili čitaju podaci

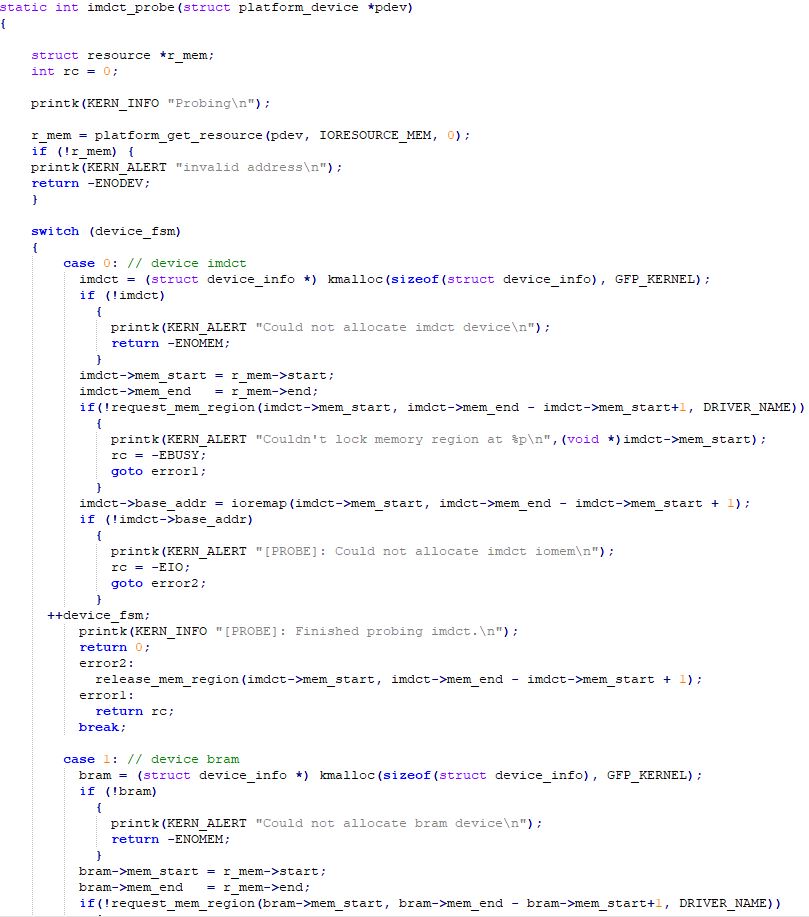
• imdct\_write () - funkcija koja omogućava upisivanje potrebnih vrednosti u registre i memorijske lokacije uređaja..

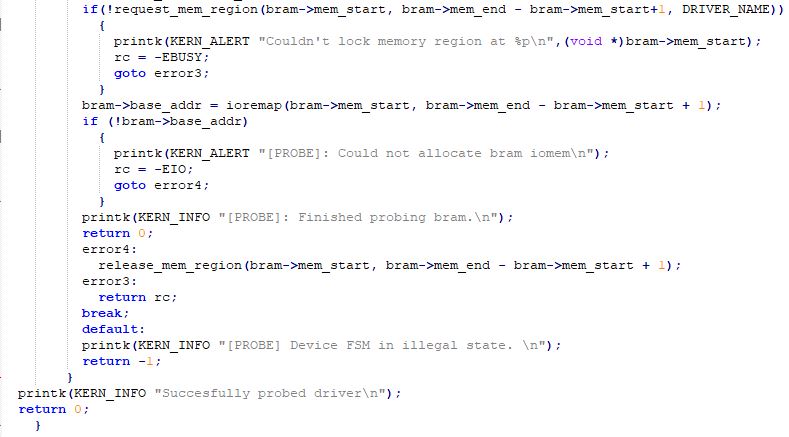
• imdct\_read () - služi za čitanje podataka.

• imdct\_exit - definiše funkciju koja se poziva pri pokretanju rmmod komande, odnosno isključivanje našeg programa iz kernela.

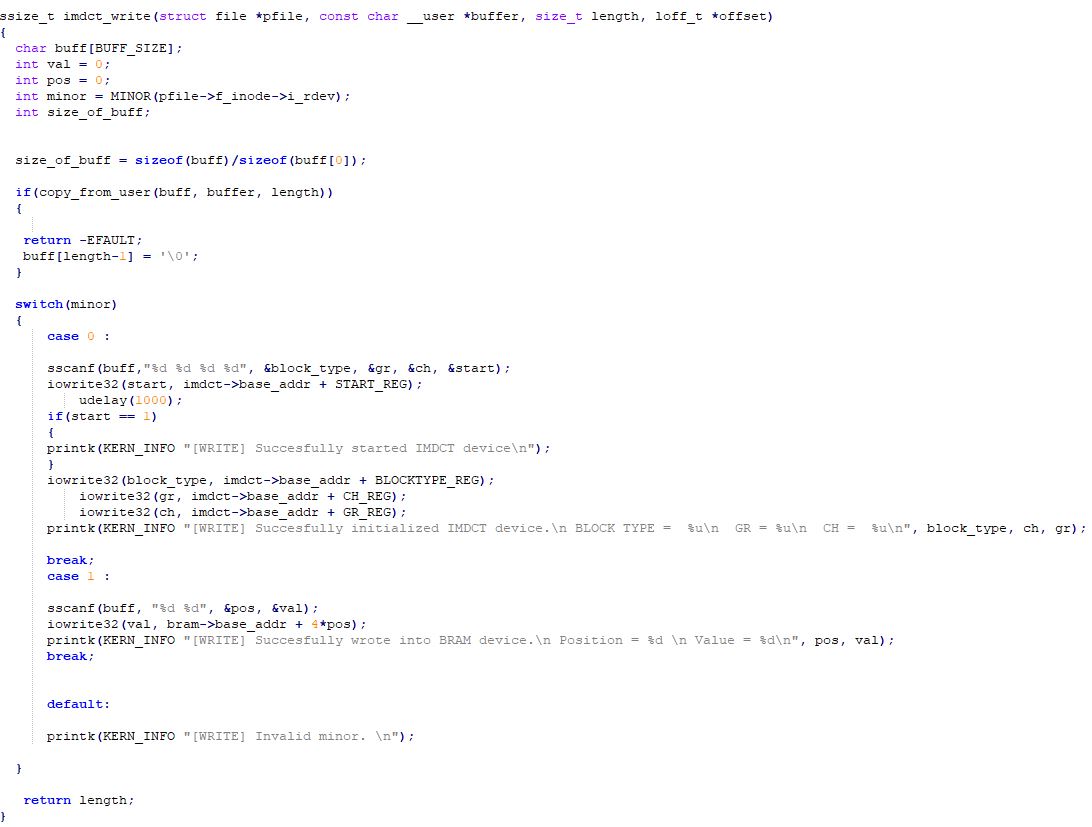
• imdct\_init () – definiše funkciju koja se poziva pri izvršavanju insmod komande, odnosno

izvršava povezivanje našeg programa sa kernelom.

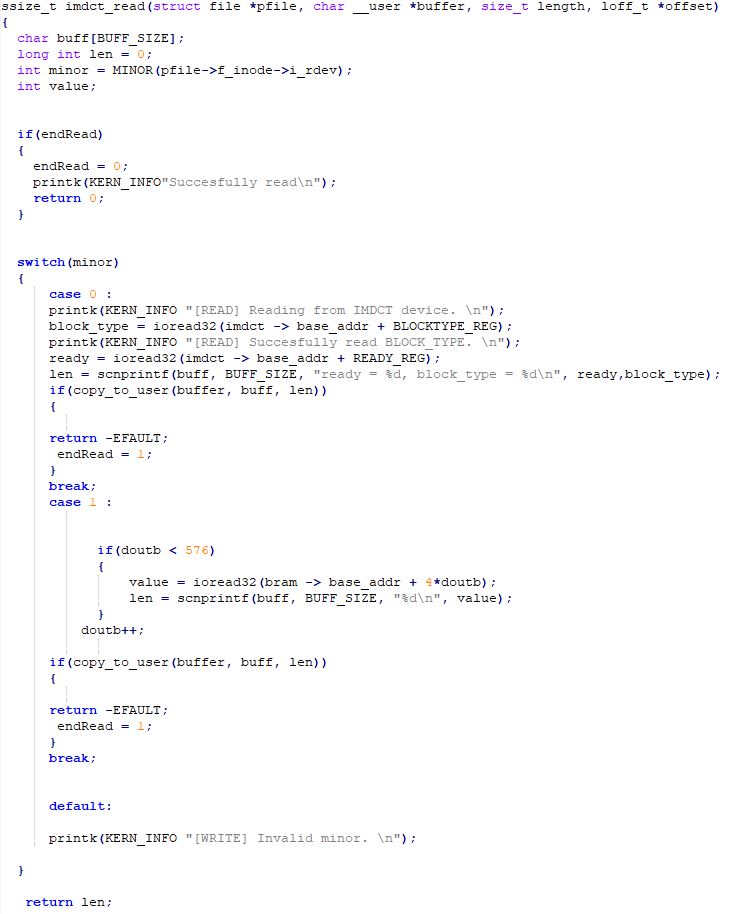




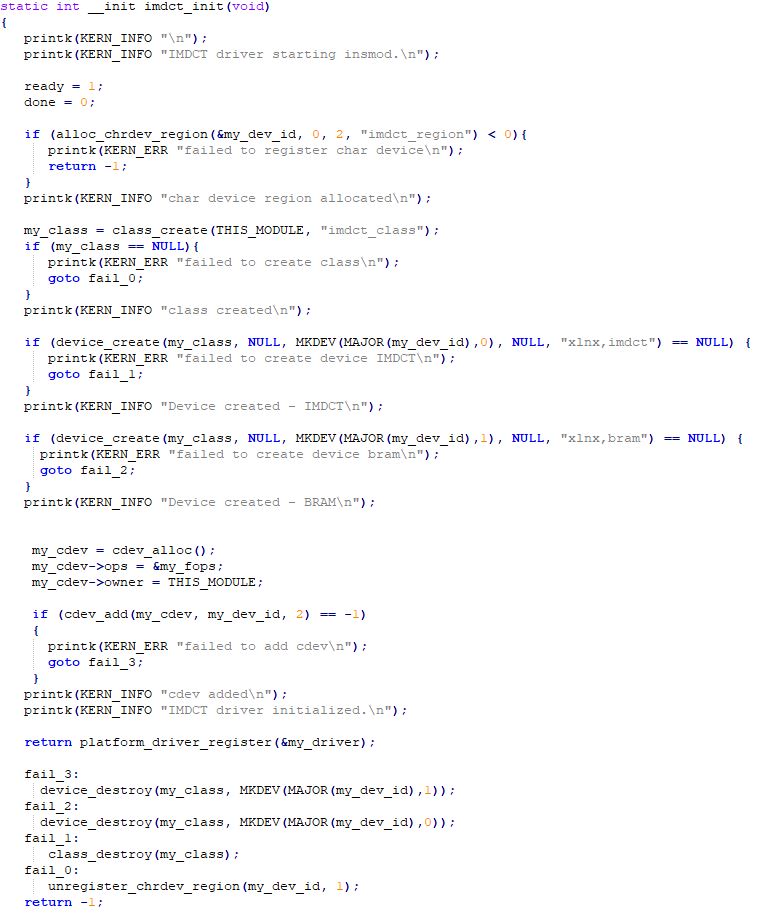
*slika 4.1 imdct\_probe funkcija*



*slika 4.2 imdct\_write funkcija*



*slika 4.3 imdct\_read funkcija*



*slika 4.4 imdct\_init funkcija*

# Aplikacija za testiranje drajvera

Osnovne funkcionalnosti funkcija implementiranih u aplikaciji :

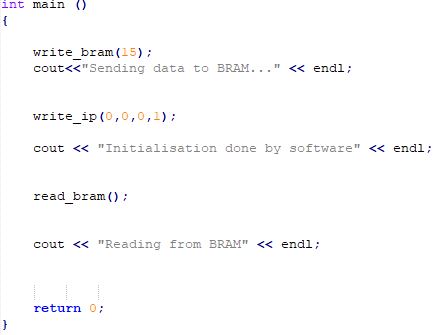
• write\_ip () – funkcija kojoj se prosleđuju 4 veličine. Kada start veličina ima vrednost 1

znači da kreće imdct, odnosno to označava da je sve spremno za početak rada funkcije IMDCT. Veličina block\_type,gr i ch označavaju register gde će se određeni podaci upisivati.

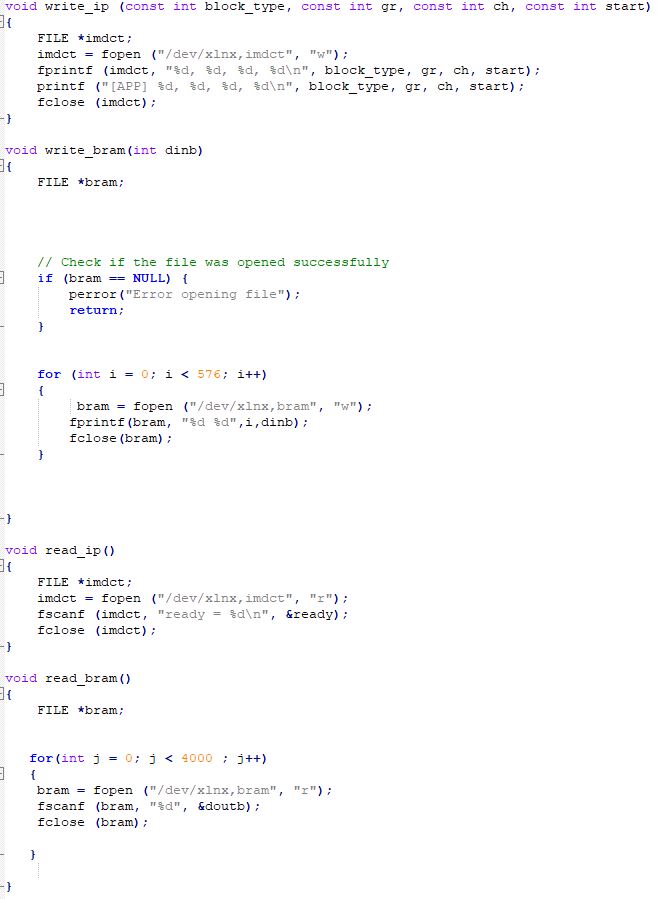
• write\_bram () – označava upisivanje podataka u BRAM

• read\_ip () – označava čitanje ready veličine koja predstavlja kraj funkcije

• read\_bram () – označava čitanje podataka iz BRAM



*slika 5.1 main funkcija aplikacije*



*slika 5.2 write\_ip, write\_bram, read\_ip i read\_bram funkcije aplikacije*