Implementacija JPEG kodera zasnovanog na FPGA

Valek, Igor

SVEUČILIŠTE JOSIPA JURIJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Diplomski sveučilišni studij

IMPLEMENTACIJA JPEG KODERA ZASNOVANOG NA FPGA

Diplomski Rad

Igor Valek

Osijek, 2018

Sadržaj

1. UVOD

Obrada digitalne slike zahtjeva veliku procesorsku snagu, a vrijeme obrade slike velike rezolucije nije zanemarivo. Jedna od korištenih normi za kompresiju i spremanje slike u digitalnom obliku je JPEG (engl. *Joint Photographic Expert Group*) norma. Kompresija slike u JPEG normi temelji se na matematičkim proračunima kako bi se postigla kompresija, a zadržala približno ista vizualna kvaliteta slike. Rješavanje matematičkih proračuna za JPEG normu je proces koji zahtjeva vrijeme i resurse za izvršavanje. Za rješavanje takvih matematičkih problema izrađuju se sustavi posebne namjene, poput FPGA sustava, koji ubrzavaju cjelokupni proces i ne zauzimaju procesorsku moć.

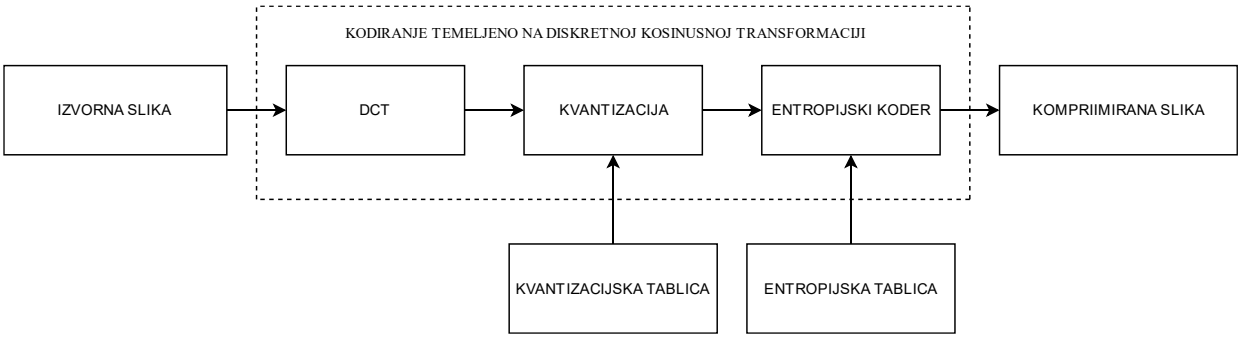
U sklopu diplomskog rada potrebno je realizirati JPEG koder na Zynq-7000 SoC (engl. *System on Chip*) platformi. Koder je potrebno realizirati programibilnom logikom korištenjem VHDL-a za opis fizičke arhitekture. Koder treba imati svojstvo podešavanja stupnja kompresije na skali od 1 do 100. Upravljanje koderom u vidu slanja i primanja podataka, podešavanja faktora kvalitete te mjerenje vremena obrade slike potrebno je realizirati na dvojezgrenom procesoru ARM Cortex-A9 u programskom jeziku C.

Postupak kompresije slike u slkadu s definiranom JPEG normom objašnjen je u drugom poglavlju ovog diplomskog rada. Tehnologije koje su korištene za izradu FPGA sustava koji vrši kompresiju slike prema JPEG normi objašnjene su u trećem poglavlju. U četvrtom poglavlju opisan je izrađeni FPGA sustav koji vrši kompresiju slike prema JPEG normi. U petom poglavlju ovog diplomskog rada prikazani su rezultati korištenja stvorenog FPGA sustava koji vrši kodiranje slike prema JPEG normi.

2. JPEG NORMA ZA KOMPRESIJU SLIKE

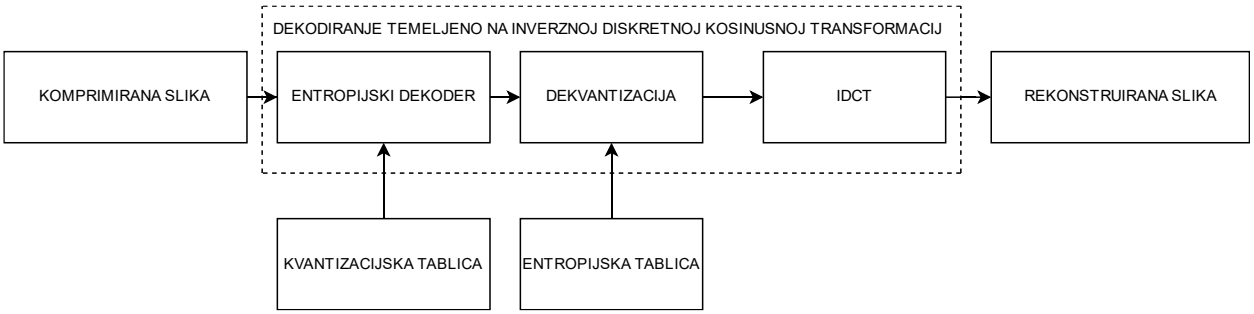
JPEG norma definira način na koji se slika kodira u niz podataka, i način na koji se iz kodiranog niza podataka slika ponovno dekodira. JPEG norma definira komprimiranje slike na dva načina, s gubitcima i bez gubitaka.

JPEG koder temelji se na diskretnoj kosinusnoj transformaciji (engl. *Discrete Cosine Transform*, DCT) koja vrši transformaciju iz prostorne domene u frekvencijsku domenu. Nakon što se izvrši DCT transformacija dobije se udio pojedinih prostornih frekvencija u slici. Kvantizacijskim postupkom uklanjaju se visoke prostorne frekvencije iz slike, koje se pojavljuju na oštrim prijelazima u slici, a zatim se entropijskim koderom vrši kompresija slike. Na slici 2.1. prikazan je blok dijagram JPEG kodera.



Slika 2.1. *Blok dijagram JPEG kodera [1]*

JPEG dekoder sastoji se od entropijskog dekodera, dekvantizatora i inverzne diskretne kosinusne transformacije. Blok dijagram JPEG dekodera prikazan je na slici 2.2. Dekoder iz kodiranog niza podataka slike, učitava entropijsku i kvantizacijsku tablicu. Nakon učitanih tablica, dekoder započinje dekodiranje slike.



Slika 2.2. *Blok dijagram JPEG dekodera [1]*

Entropijski dekoder prema specificiranoj entropijskoj tablici dekodira podatke koji opisuju komprimiranu sliku i podatke prosljeđuje sustavu za dekvantizaciju. Podaci se devkantiziraju

prema istoj kvantizacijskoj tablici prema kojoj je izvorna slika kvantizirana, te nakon izvršene dekvantizacije, inverzna diskretna kosinusna transformacija vrši transformaciju iz frekvencijske u prostornu domenu, te je tim postupkom slika rekonstruirana.

2.1. Diskretna kosinusna transformacija

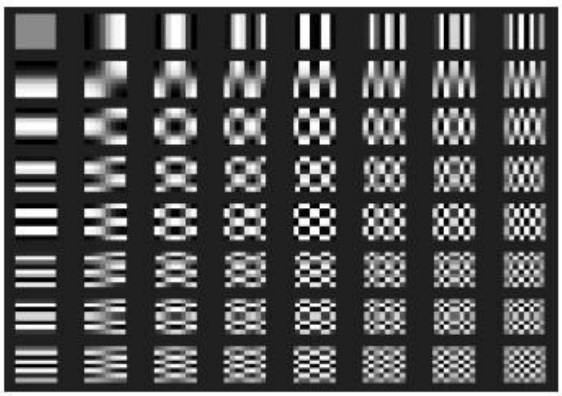
Ulazna slika dijeli se na blokove veličine 8x8 elemenata slike, nad kojima se vrši diskretna kosinusna transformacija koja se računa po formuli (2-1).

$$F(u,v) = \frac{1}{4}C(u)C(v)\left[\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y) * \cos\frac{(2x+1)u\pi}{16}\cos\frac{(2y+1)v\pi}{16}\right] \quad (2\text{-}1)$$

$$C(u) = \begin{cases} \frac{1}{\sqrt{2}}, & u = 0 \\ 1, & u \neq 0 \end{cases} \quad (2\text{-}2)$$

$$C(v) = \begin{cases} \frac{1}{\sqrt{2}}, & v = 0 \\ 1, & v \neq 0 \end{cases} \quad (2\text{-}3)$$

Svaki blok izvorne slike veličine 8x8 predstavlja 64 elementa slike diskretnog signala koji je funkcija prostornih koordinata *x* i *y*. Diskretna kosinusna transformacija razdvaja takav signal na 64 ortogonalna bazna signala. Svaki od tih signala prikazuje jedinstveni udio dvodimenzionalnih prostornih frekvencija *u* i *v* u slici. C(*u*) i C(*v*) su konstante definirane formulama (2-2) i (2-3). Na slici 2.3. prikazani su bazni signali diskretne kosinusne transformacije. Rezultat diskretne kosinusne transformacije je novih 64 vrijednosti koje predstavljaju amplitude baznih signala prostornih frekvencija odnosno DCT koeficijente.



Slika 2.3. *Bazne funkcije diskretne kosinusne transformacije [2]*

Vrijednosti DCT koeficijenata predstavljaju relativni udio dvodimenzionalnih prostornih frekvencija sadržanih u slici. Koeficijent čija je vrijednost prostornih frekvencija u obje osi jednaka nuli naziva se DC koeficijent i predstavlja istosmjernu komponentu, a preostalih 63 koeficijenta

nazivaju se AC koeficijenti i svaki od njih predstavlja udio pojedine prostorne frekvencije unutar slike.

Transformacija iz frekvencijske domene natrag u prostornu vrši se inverznom diskretnom kosinusnom transformacijom. Inverzna diskretna kosinusna transformacija dijeli kodiranu sliku na 8x8 blokove koji se sastoje od jedne istosmjerne i 63 izmjenične DCT komponente, te ih vraća u prostornu domenu prema formuli (2-4). Nakon izvršene inverzne diskretne kosinusne transformacije dobije se 8x8 blok čije vrijednosti odgovaraju vrijednostima elemenata slike na prostornim koordinatama *x* i *y*, koje opisuju izvornu sliku.

$$f(x,y) = \frac{1}{4}\left[\sum_{u=0}^{7}\sum_{v=0}^{7} C(u)C(v)F(u,v)\cos\frac{(2x+1)u\pi}{16}\cos\frac{(2y+1)v\pi}{16}\right] \tag{2-4}$$

$$C(u) = \begin{cases} \frac{1}{\sqrt{2}}, & u = 0 \\ 1, & u \neq 0 \end{cases} \tag{2-5}$$

$$C(v) = \begin{cases} \frac{1}{\sqrt{2}}, & v = 0 \\ 1, & v \neq 0 \end{cases} \tag{2-6}$$

2.2. Kvantizacija

Kako bi se postigla kompresija slike, za svaki od dobivena 64 DCT koeficijenta vrši se uniformna kvantizacija s odgovarajućom kvantizacijskom tablicom koja mora biti specificirana od strane korisnika kao ulazni podatak u kvantizator. Vrijednosti kvantizacijske tablice su cjelobrojni brojevi u rasponu od 1 do vrijednosti 255, a odgovaraju koraku kvantizacije za definirane DCT koeficijente. Cilj kvantizacije je izbaciti informacije iz slike koje korisniku nisu značajne, i ne utječu uvelike na kvalitetu slike. Postupkom kvantizacije postiže se kompresija slike s gubicima.

Kvantizacijski postupak je definiran kao dijeljenje pojedinog DCT koeficijenta *F*(*u*,*v*) s odgovarajućim kvantizacijskim korakom *Q*(*u*,*v*), zapisanim unutar kvantizacijske tablice i računa se prema formuli (2-7).

$$F^Q = round\left(\frac{F(u,v)}{Q(u,v)}\right) \tag{2-7}$$

Dekvantizacija je postupak koji kvantizirane vrijednosti DCT koeficijenata množi s kvantizacijskim korakom danim u kvantizacijskoj tablici. Za pravilnu dekvantizaciju mora biti korištena ista kvantizacijska tablica, odnosno kvantizacijski korak koji je bio korišten za samu kvantizaciju. Dekvantizacija se računa po formuli (2-8).

$$F^{Q'}(u,v) = F^Q(u,v) * Q(u,v) \tag{2-8}$$

Za postizanje najveće kompresije uz zadovoljavajuću kvalitetu bez vidljivih artefakata, kvantizacijski korak odabire se subjektivno. JPEG organizacija psiho-vizualnim eksperimentima donijela je standardiziranu kvantizacijsku tablicu koje bi se korisnici trebali pridržavati [3]. Tablica je prikazana na slici 2.4. Gornja polovina kvantizacijske tablice odnosi se na luminantnu komponentu, a donja polovina na krominantnu komponentu unutar slike.

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99
17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

Slika 2.4. *Kvantizacijska tablica [3]*

2.2.1. Koeficijent kvalitete

Kvaliteta slike će direktno ovisiti o razini kompresije. Što je kompresija veća to će kvaliteta slike biti manja. Koeficijent kvalitete direktno utječe na standardiziranu kvantizacijsku tablicu, kako bi postigao veću ili manju kompresiju slike. Faktor kvalitete *S* je broj u intervalu od 1 do 100. Što je faktor kvalitete manji to je kompresija slike veća. Ovisnost vrijednosti unutar kvantizacijske tablice o faktoru kvalitete S dan je formulom (2-9).

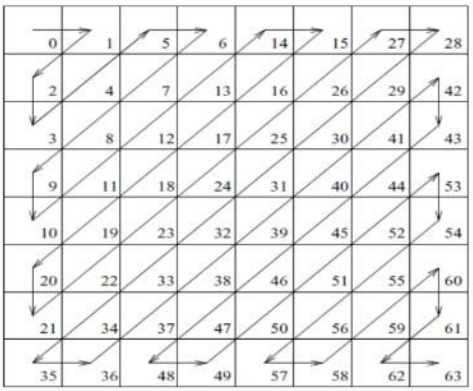
$$Q(u,v)' = \begin{cases} \frac{5000}{S} \cdot Q(u,v) & za\ S\ [1,50] \\ (200 - 2S) \cdot Q(u,v) & za\ \ S\ [50,100\rangle \\ Q(u,v) & za\ \ S = 100 \end{cases} \tag{2-9}$$

2.3. Entropijski koder

Entropijski koder je posljednji korak unutar okvira JPEG norme, i njegova zadaća je izvršiti kompresiju nad podacima koji su rezultat kvantizacije. Unutar slike postoji snažna korelacija između istosmjernih komponenti diskretne kosinusne transformacije dvaju susjednih 8x8 blokova, te se zbog toga istosmjerna komponenta pojedinog DCT bloka kodira zasebno od izmjeničnih komponenata. Kvantizirane DC vrijednosti kodiraju se tako što se DC vrijednost trenutnog bloka

prikazuje kao razlika kvantizirane DC komponente iz prethodnog bloka, odnosno kodira se diferencijalnom metodom za kodiranje.

Kodiranje izmjeničnih komponenti vrši se tako da se kvantizirani izmjenični koeficijenti poslože u „zig-zag“ formaciju koja je prikazana na slici 2.5., i nakon toga se izvrši kodiranje podataka. JPEG norma definira kompresiju koja se vrši u dva ciklusa. Prvi ciklus kompresije je „run-lenght“ metoda kodiranja, a drugi ciklus norma definira kodiranje ili Huffmanovim algoritmom ili aritmetičkim algoritmom za kodiranje podataka.



Slika 2.5. *Primjer zig-zag formacije[4]*

2.4. Algoritmi za kompresiju

2.4.1. Run-lenght algoritam

Run-length (RLE) algoritam za kodiranje podataka je algoritam za kompresiju podataka bez gubitaka. Algoritam vrši kompresiju tako što se kreira novi podatak koji sadrži ponavljajući podatak i broj koliko puta se podatak ponovio za redom. Kako bi kompresija bila učinkovita, statistika podataka koji se komprimiraju mora biti takva da se pojavljuje velik broj istih podataka u nizu. U ovisnosti o statistici podataka koji se trebaju kodirati, odabrat će se za koliko bitova će se proširiti novo stvoreni vektor od vektora koji opisuje kodirani podatak. Ukupni faktor kompresije ovisit će o izboru veličine vektora za brojač i statistici podataka.

2.4.2. Huffmanov algoritam

Huffmanov algoritam za kodiranje vrši kompresiju bez gubitaka. Kako bi se Huffmanovim algoritmom mogli komprimirati podaci, isti se prethodno sortiraju prema frekvenciji ponavljanja pojedinih vrijednosti. Huffmanov algoritam kodira učestalije vrijednosti s manjim brojem bitova,

a manje učestale vrijednosti kodira s većim brojem bitova. Takvim postupkom kodiranja osigurava se minimalni broj bitova za kodiranje pojedinog znaka s obzirom na frekvenciju ponavljanja. Prilikom kodiranja Huffanovim algoritmom kodovi se spremaju u rječnik koji se šalje dekoderu.

2.4.3. Aritmetičko kodiranje

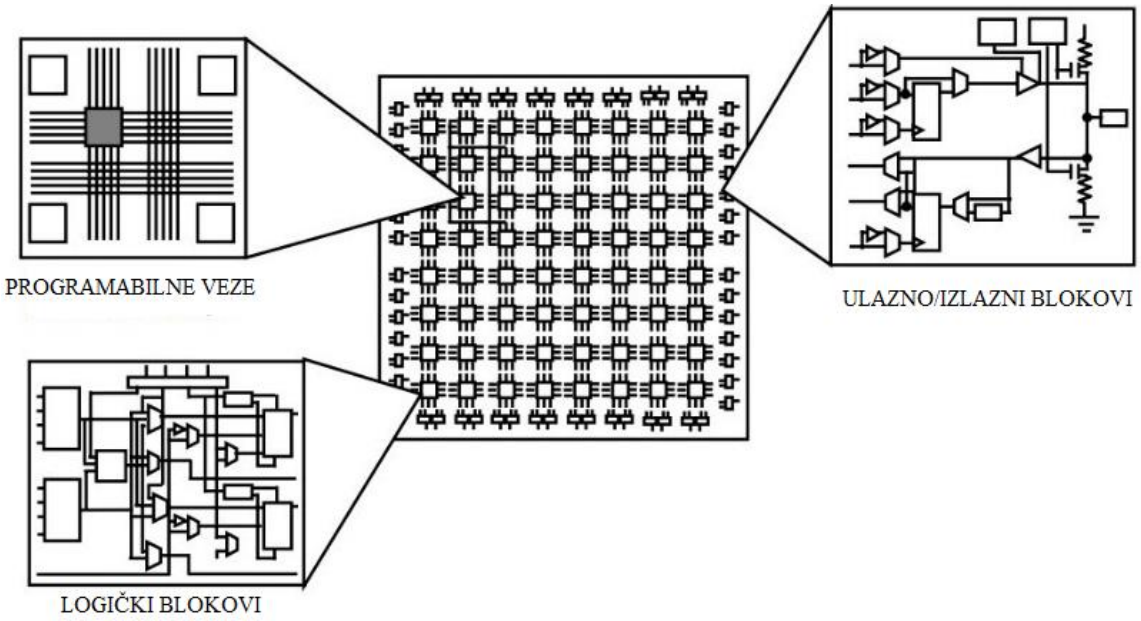
Aritmetičko kodiranje koristi se za kompresiju podataka bez gubitaka. Podaci se kodiraju prema statistici simbola, a rezultat kodiranja je jedinstveni decimalni broj. Informacija se kodira tako da se prvo napravi statistička analiza ulaznih znakova i na osnovu izvršene analize znakovi se poslože u interval od 0 do 1. Interval pojedinog znaka je područje vjerojatnosti njegovog pojavljivanja u informaciji. Nakon određenih intervala za pojedini znak koder započinje kodiranje sekvence. Sekvenca se kodira tako da svaki znak predstavlja jednu decimalu, pritom kodiranje započinje od prvog znaka koji ujedno označava i prvu decimalu a završava s zadnjim znakom sekvence koja određuje zadnju decimalu jedinstvenog decimalnog broja. Svaka decimala određena je područjem vjerojatnosti pojavljivanja znaka koji se kodira.

3. TEHNOLOGIJE KORIŠTENE ZA IZRADU FPGA SUSTAVA

3.1. FPGA

FPGA (engl. *Field Programmable Gate Array*) sastoji se od mreže programabilnih logičkih blokova (engl. *Configurable Logic Block*, CLB), od kojih se svaki pojedini blok može konfigurirati kako bi obavljao određenu logičku funkcionalnost. CLB blokovi su okruženi s perifernim ulazno/izlaznim krugovima i povezani su programabilnim vezama. Interna struktura FPGA prikazana je na slici 3.1.

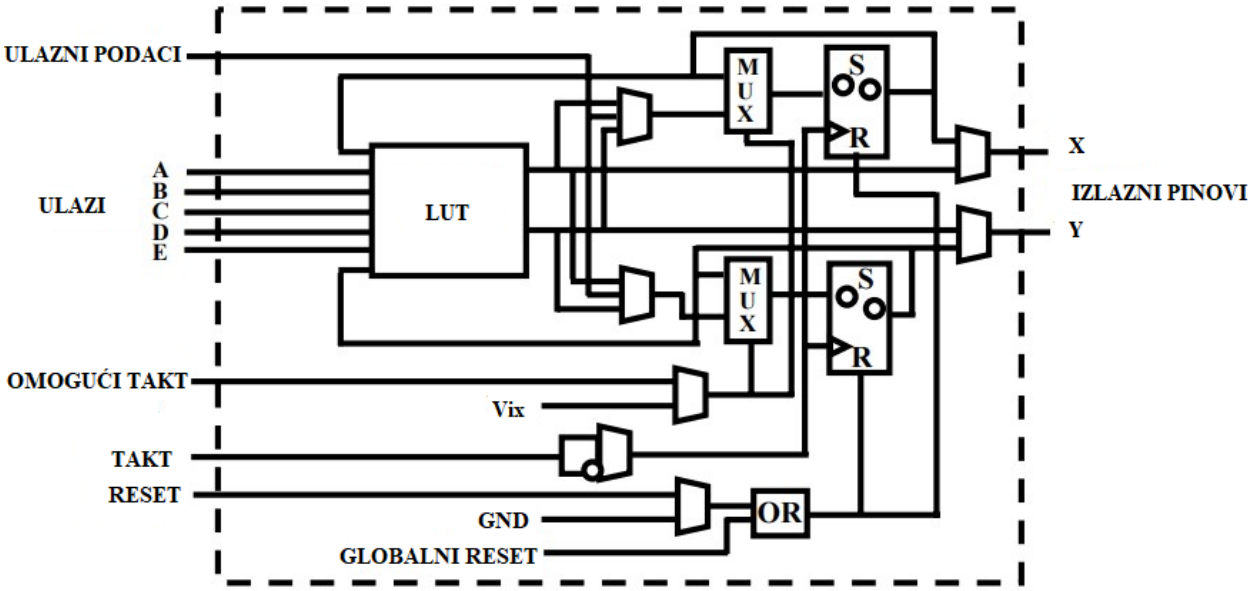
Kada se FPGA konfigurira, njegovi unutarnji vodovi se spajaju tako da kreiraju digitalnu hardversku logiku koju je korisnik opisao u jednom od jezika za opisivanje hardvera, poput VHDL-a ili Veriloga.



Slika 3.1. *Interna struktura FPGA [5]*

3.1.1. Logički blok

Logički blok FPGA sustava može biti konfiguriran tako da njegova funkcionalnost može predstavljati kompleksnu digitalnu logiku. Logički blok može implementirati kombinacijsku ili sekvencijalnu digitalnu logiku. Logički blokovi mogu biti: tranzistorski parovi, kombinacijska vrata poput „i“, „ili“, „ni“, „nili“, „isključivo ili“, „isključivo nili“, mogu biti LUT (engl. *Lookup Table*) , i multiplekser.



Slika 3.2. *Xilinx LUT temeljen na CPLD[5]*

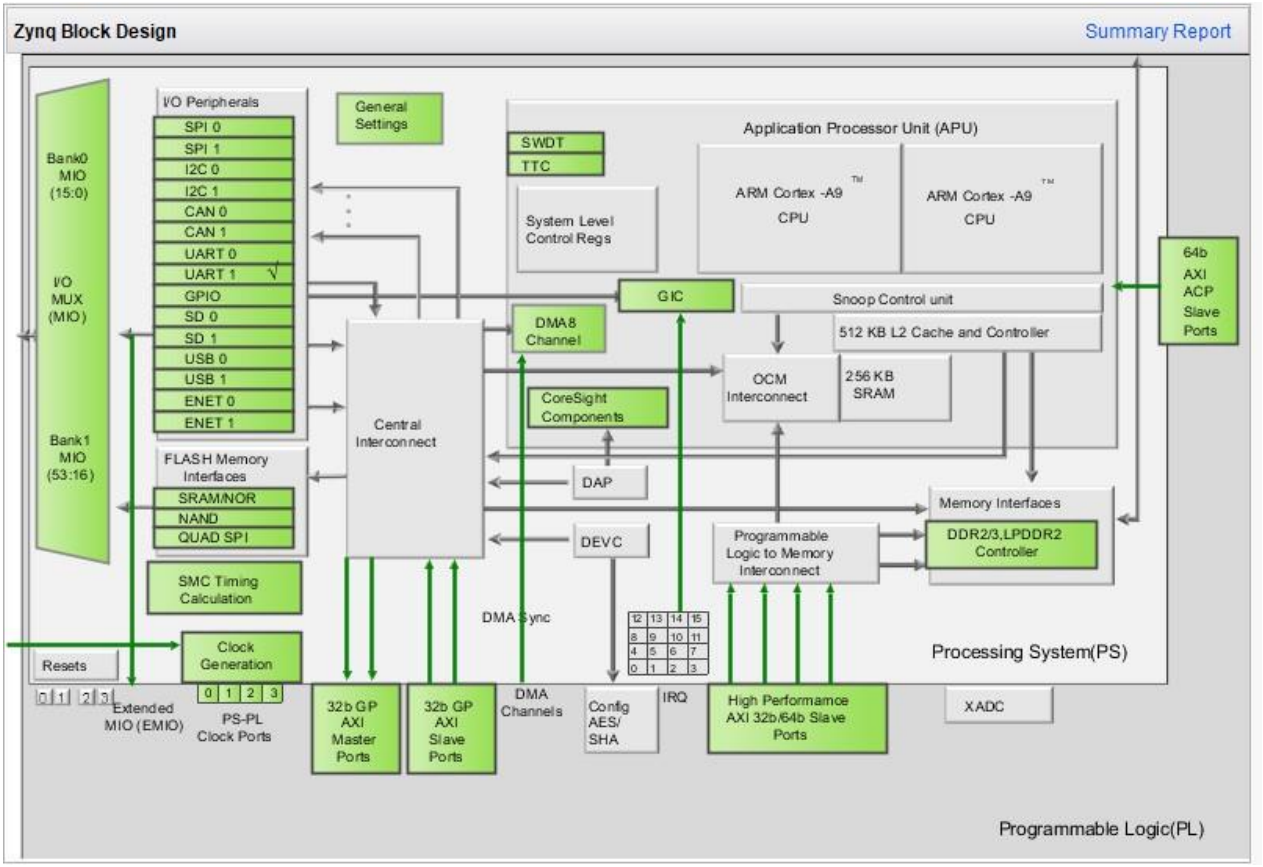
Slika 3.2. prikazuje arhitekturu CLB-a. LUT se koristi za odabiranje različitih funkcionalnosti CLB-a, a napravljene su od SRAM-a (engl. *Static Random Access Memory*) Različitim vrijednostima na ulazu u LUT on će na svom izlazu dati različite logičke funkcije.[5]

3.1.2. Zynq-7000

Zynq-7000 temelji se na „system on chip“ arhitekturi koja se sastoji od programskog sustava i programske logike. Zynq-7000 sastoji se od dvojezgrenog ARM Cortex A9 procesora. Na slici 3.3. prikazan je blok dijagram Zynq procesorskog sustava. Zynq-7000 sadrži sučelja koja povezuju programabilnu logiku i programski sustav. Sučelja koje povezuju procesorski sustav i programsku logiku dijele se na tri glavne grupe: proširene multipleksirane ulazno izlazne jedinice, ulazno izlazne jedinice programske logike, i ulazno izlaznih grupa proširivih naprednih sučelja (engl. *Advanced eXtensible interface,* AXI) .

Procesorski sustav sastoji se od jedinice za procesiranje aplikacija (engl. *Application Processor Unit,* APU), CACHE memorije, unutarnje memorije, vanjskih memorijskih sučelja i 8-kanalnog DMA sustava.

Programabilna logika sastoji se od ulazno izlaznih periferija i sučelja, konfiguracijskih blokova, blok memorija (engl. *Block Random Access Memory, BRAM*) veličine 36 Kb, Procesora za obradu digitalnih signala (engl. *Digital Signal Processor,* DSP), te programabilnih ulazno izlaznih blokova.



Slika 3.3. *Zynq blok dijagram [6]*

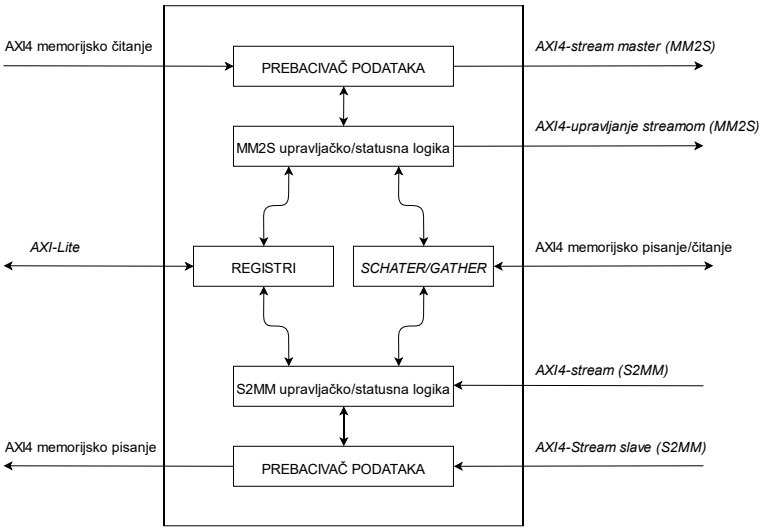
3.1.3. Blok memorija

Blok memorija je memorija koja ima mogućnost proizvoljnog odabira duljine vektora i veličine memorije. Podržava dvostruki pristup memoriji veličine 36 Kb. Također, BRAM podržava programabilnu FIFO (engl. *First Input First Output*) logiku, te ugrađene sustave za zaštitu od pogreške. Memorija je sinkrona što znači da se ispis i upis odvijaju sinkrono s taktom. Pristupanje ulazima u blok memoriju može se podijeliti na segmente tako da tvore više sučelja koja mogu biti veličine: 32K × 1, 16K × 2, 8K × 4, 4K × 9 (or 8), 2K × 18 (or 16), 1K × 36 (or 32), 512 × 72. Različita sučelja mogu raditi istovremeno na istom memorijskom bloku.

3.1.4. Izravni pristup memoriji

Sustav za izravni pristup memoriji (engl. *Direct Memory Access*, DMA) je modul koji omogućava brzi prijenos podataka iz procesorskog sustava u programsku logiku i obrnuto, i pritom ne koristi resurse procesora za obavljanje prijenosa podataka. Za prijenos podataka koristi „*memory maped to stream* (MM2S)“ i „*stream to memory map* (S2MM)“ protokole, a za konfiguraciju samog modula koristi se AXI protokol. AXI DMA omogućuje istovremeni rad do 16 kanala koji mogu biti ili MM2S ili S2MM. Podržava „*micro* DMA“, mod koji se koristi za

prijenose male količine podataka, „*scather gather*“ mod za samostalni prijenos podataka, i „*simple*“ mod rada. AXI DMA podržava mogućnost stavljanja prijenosa podataka u red čekanja. Također, AXI DMA omogućuje poravnanje prijenosa, odnosno čitanje ili pisanje podataka može započeti na bilo kojem bajtu. Na slici 3.4. prikazan je blok dijagram DMA modula.



Slika 3.4. *Blok dijagram DMA modula [7]*

3.1.5. Procesor za obradu digitalnih signala

Procesor za obradu digitalnih signala (DSP) je mikroprocesor posebne namjene, čija je arhitektura prilagođena za obradu digitalnih signala. DSP koristi posebnu memorijsku arhitekturu koja mu omogućava višestruko istovremeno dohvaćanje podataka i instrukcija. Za razliku od standardnih procesora DSP sadrži specifični skup instrukcija od kojih je jedna instrukcija DSP-a u mogućnosti zamijeniti višestruke instrukcije standardnih procesora. Instrukcije su optimizirane za računanje učestalih matematičkih operacija koje se pojavljuju u obradi digitalnog signala.

DSP blok sastoji se od 25x18 dvojno komplementarnih množitelja, i 48-bitnih akumulatora koji mogu raditi na frekvenciji od 748 MHz. Također, akumulatori mogu imati ulogu 48-bitnog uzlaznog i silaznog brojača. DSP uključuje dodatna zbrajala u ulaznom stupnju, koji poboljšavaju performanse u velikim sustavima.

3.2. Protokoli integriranog kruga

3.2.1. Napredna arhitektura mikroupravljačkog sustava

Napredna arhitektura mikroupravljačkog sustava (engl. *Advanced Microcontroller Bus Architecture,* AMBA) je standard za spajanje i upravljanje modulima koji se nalaze u istom

integriranom krugu. AMBA specifikacija definira naprednu visoko-učinkovitu sabirnicu (engl. *Advanced High-performance Bus,* AHB) koja podržava efikasne konekcije procesora i memorija koje se nalaze unutar i van čipa. Specificira naprednu periferijalnu sabirnicu (engl. *Advanced Peripheral Bus,* APB) koja se koristi kao sabirnica za pristup periferijalnim funkcijama. APB je optimiziran po potrošnji energije i kompleksnost sučelja je minimizirana. AMBA definira i naprednu sustavnu sabirnicu (engl. *Advanced System Bus,* ASB) koja se ponaša kao zamjena za AHB, odnosno može zamijeniti AHB gdje nisu potrebni visoko-učinkoviti zahtjevi.

3.2.2. Proširivo napredno sučelje

Proširivo napredno sučelje (AXI) je sučelje definirano u trećoj generaciji AMBA standarda. Cilj AXI sučelja je postizanje visokih performansi i visoke frekvencije takta pri prijenosu podataka. AXI omogućuje odvojene adrese, kontrolnu i podatkovnu fazu, podržava neporavnati prijenos podataka na razini bajta, i omogućuje kontinuirani prijenos podataka „*burst mode*“.

3.3. Jezici za opisivanje fizičke arhitekture

Hardverski opisni jezici (engl. *Hardware Description Language,* HDL) su jezici koji se koriste za opisivanje strukture i ponašanja digitalnih sklopova. Također, omogućavaju testiranje opisanih digitalnih sklopova kroz simulaciju.

Dizajniranje pomoću HDL-a dijeli se na četiri razine. Prva razina je sustavna razina (engl. *System level*), u ovoj razini se opisuje model koji predstavlja ponašanje opisanog sustava. Druga razina je blokovska razina, naziva se još i registarska razina (engl. *Register Transfer Level*, RTL). U ovoj razini sustav se sastoji od detaljnog opisa, i sagrađen je od različitih blokova poput zbrajala, multipleksera, množitelja i ostalih blokova. Treća razina je razina logičkih sklopova (engl. *Gate level*) u kojoj su blokovi predstavljeni mrežom međusobno povezanih osnovnih digitalnih logičkih elemenata. Zadnja razina je razina u kojoj se dobiveni dizajn prilagođava uvjetima za određenu arhitekturu.

3.3.1. VHDL

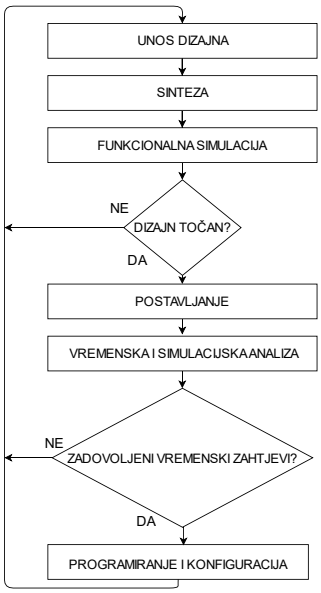
VHDL (engl. *Very High-Speed Integrated Circuit Hardware Description Language*) je jezik za opisivanje fizičke arhitekture. VHDL objašnjava strukturu i ponašanje digitalnih elektroničkih sustava. Sastoji se od entiteta koji opisuje sučelje modula, odnosno ulazno izlazne portove, i od arhitekture koja opisuje ponašanje entiteta. Primarno se koristi za izradu specifičnih integriranih krugova (engl. *Application Specific Integrated Circuits,* ASIC).

3.3.2. Verilog

Verilog je jezik za opisivanje hardvera koji se koristi za dizajniranje i verificiranje digitalnih sustava. Koristi sintaksu sličnu programskom jeziku C. Sastoji se od 4 apstrakcijske razine. Prva razina je opisna razina (engl. *Behavioral level*). Opisna razina je najviša razina te se u njoj opisuje dizajn na razini opisivanja algoritma. Druga razina je razina podatkovne putanje (engl. *Dataflow level*); na ovoj razini korisnik poznaje putanju podataka, odnosno tok podataka kroz sustav. Razina logičkih sklopova (engl. *Gate level*) je treća razina. Na toj razini korisnik dizajnira logiku od osnovnih digitalnih sklopova. Posljednja razina je najniža razina gdje korisnik logiku opisuje na temelju prekidača (*Switch level*) i tako kreira dizajn sustava. Verilog dozvoljava miješanje apstrakcijskih razina u istom dizajnu.

3.4. Dizajniranje FPGA sustava

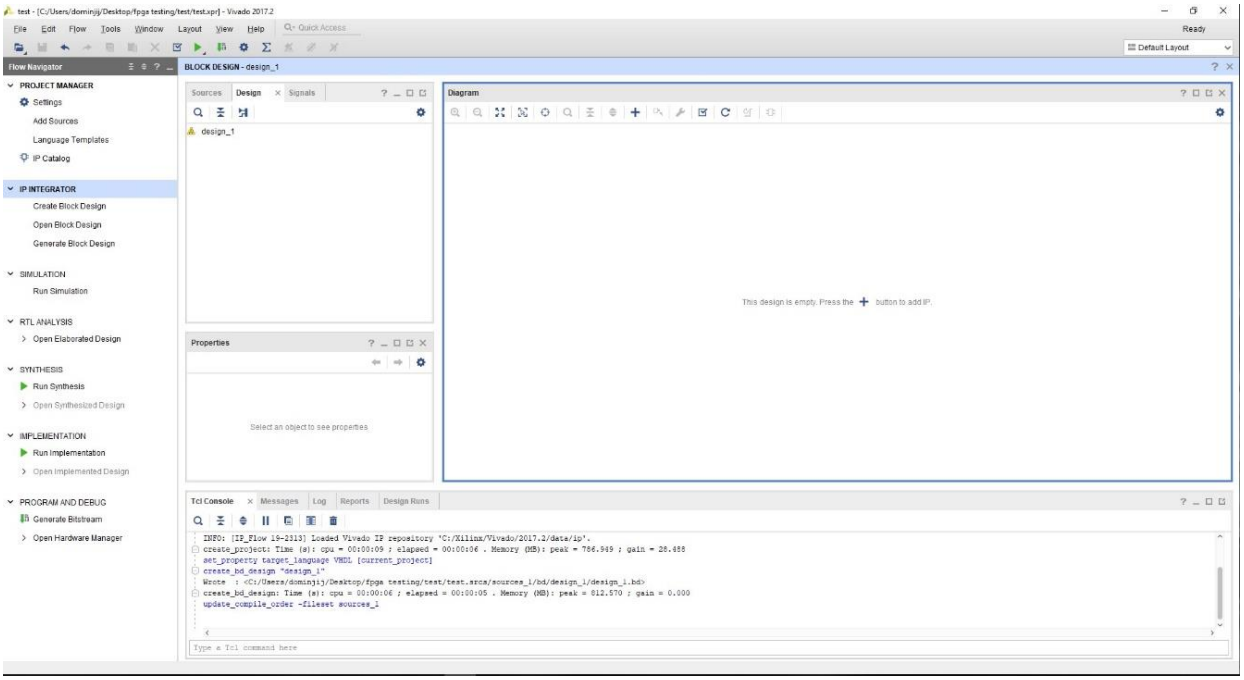
Sinteza je postupak izrade digitalnog sustava na temelju opisanog HDL dizajna. U postupku sinteze dizajn se gradi od standardnih modula poput registara, multipleksera, LUT-ova, i osnovnih digitalnih sklopova. Implementacija je proces koji prevodi dizajn dobiven sintezom u oblik dizajna koji odgovara ciljanoj arhitekturi. Zatim proces za mapiranje prevodi standardne module iz dizajna u module koji su dostupni na specifičnoj arhitekturi. Proces za postavljanje i spajanje modula određuje gdje će se nalaziti moduli na ciljanoj arhitekturi i spaja ih po danom dizajnu. Zadnji korak je stvaranje binarne datoteke na temelju izvršene implementacije, kojom se programira FPGA sustav. Na slici 3.5. prikazan je dijagram toka dizajniranja FPGA sustava.



Slika 3.5. *Dijagram toka dizajniranja FPGA sustava [8]*

3.5. VIVADO

Vivado je razvojno programsko okruženje za dizajniranje, sintetiziranje, analiziranje i simuliranje digitalnih sustava. Ima mogućnost razvoja sustava koji se temelje na „system on chip“ tehnologiji te pruža „*high-level*“ sintezu (HLS). HLS sintetizira kod napisan u programskom jeziku C. Sadrži informacije o hardverskim platformama koje se koriste za postavljanje i generiranje paketa podrške za ploče (engl. *Board Suport Package*, BSP), programiranje programabilne logike, postavke JTAG i ostale automatizirane operacije. Vivado podržava grafičko spajanje i podešavanje parametara modula, i perifernih sklopova. Na slici 3.6. prikazano je sučelje Vivado razvojnog okruženja.



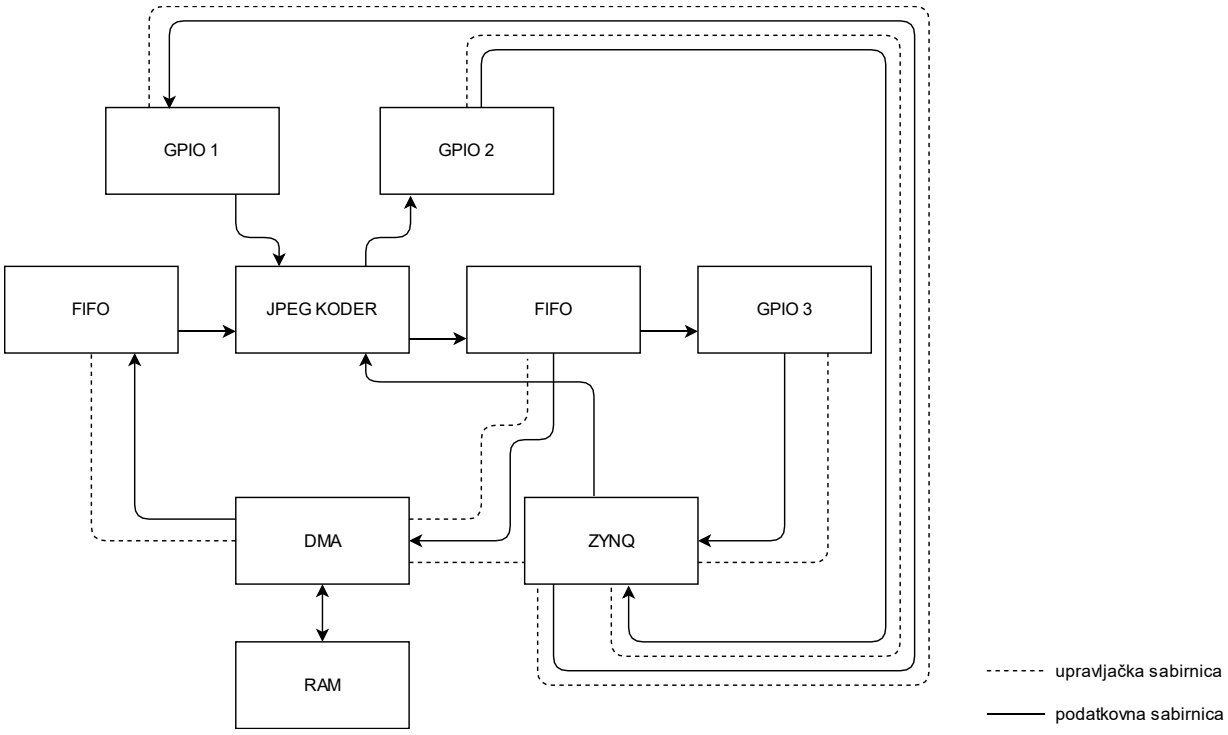
Slika 3.6. *Izgled sučelja Vivado razvojnog okruženja [6]*

4. FPGA IMPLEMENTACIJA JPEG KODERA

Sustav za kompresiju slike prema JPEG normi na FPGA tehnologiji realiziran je na Zybo razvojnoj pločici. Sustav se pokreće iz programskog razvojnog okruženja (engl. *Software Development Kit*, SDK). SDK omogućava unos podataka i čitanje podataka iz RAM-a Zybo razvojne pločice. SDK također pokreće upravljački program (engl. *Driver*) koji upravlja sustavom za kodiranje slike prema JPEG normi.

4.1. Sustav za kompresiju slike prema JPEG normi

Na slici 4.1. prikazan je blok dijagram sustava za kompresiju slike prema JPEG normi. Sustav se sastoji od Zynq procesora čija je uloga upravljanje cijelim sustavom za vršenje kompresije slike prema JPEG normi. Sastoji se od DMA modula čija je uloga dostavljati podatke za komprimiranje iz RAM-a u JPEG koder koji vrši kompresiju slike, i komprimirane podatke spremati natrag u RAM. Ulazi i izlazi općenite namjene (engl. *General Purpose Input Output*, GPIO) koriste se za povezivanje Zynq procesora s hardverskom implementacijom JPEG kodera.



Slika 4.1. *Sustav za kompresiju slike prema JPEG normi, JPEG koder napravljen je u sklopu ovog diplomskog rada*

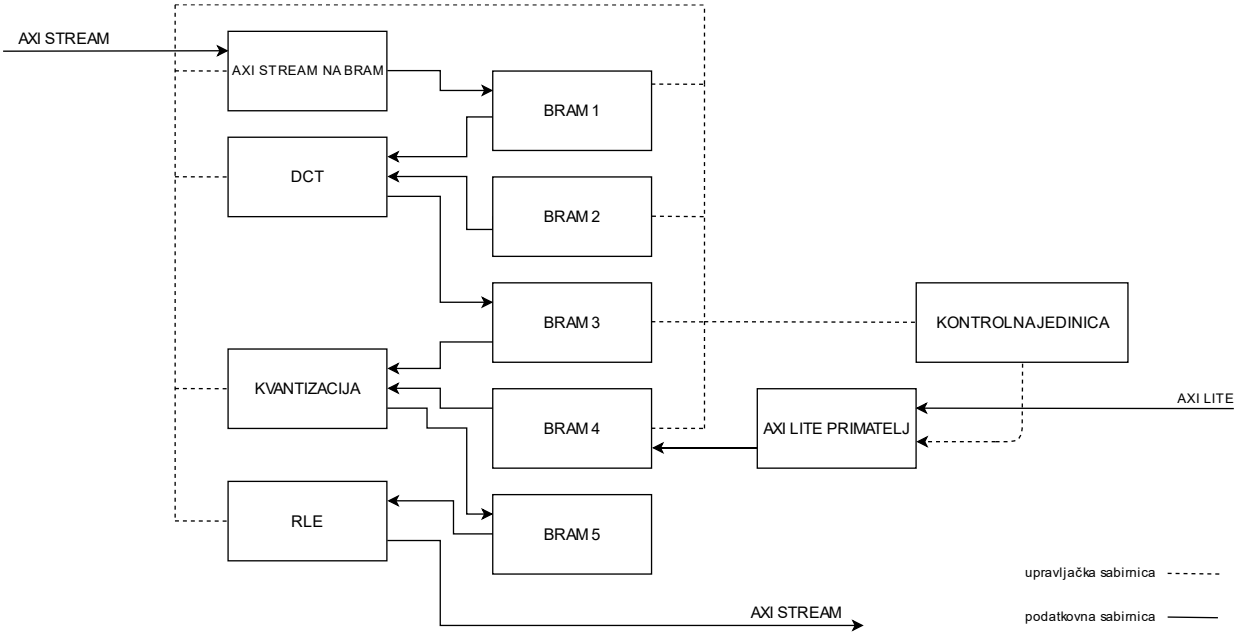
JPEG koder modul učitava podatke iz FIFO memorije u BRAM koristeći „AXI stream“ protokol. Kapacitet BRAM modula za ulazne podatke iznosi 128 podataka veličine 8 bita. U

inicijalnom stanju DCT pod-modul čeka da se napuni prvi blok podataka slike, odnosno 64 podatka koji se nalaze na prve 64 adrese u BRAM-u, te nakon toga započinje obradu. Kada DCT obradi blok podataka s prve 64 adrese, tada kreće obrađivati podatke s druge 64 adrese BRAM-a, dok se prve 64 adrese BRAM-a puni s novim podacima. Navedenim načinom DCT pod-modul obrađuje 8x8 blokove podataka bez prekida, odnosno izbjegava se čekanje na učitavanje podataka u BRAM. DCT pod-modul računa DCT koeficijente prema formuli (2-1), ali je formula preuređena i svedena na 4096 prethodno izračunatih konstanti koje predstavljaju DCT bazne funkcije. Konstante su spremljene u zasebni BRAM. Tim postupkom se cijela diskretna kosinusna transformacija svela na umnožak odgovarajuće konstante s odgovarajućim elementom slike, što zahtijeva manje hardverskih resursa, i manji broj taktova za obradu diskretne kosinusne transformacije. Svaki puta kada algoritam za računanje DCT-a izračuna DCT koeficijent, pod-modul za kvantizaciju isti DCT koeficijent kvantizira i spremi ga u zaseban BRAM. Kvantizacijski pod-modul kvantizirane vrijednosti sprema u BRAM na način da ih slaže na adrese u BRAM-u u „zig-zag“ poretku vidljivom na slici 2.5. Kada su svi DCT koeficijenti izračunati i nad njima izvršena kvantizacija, modul za komprimiranje izvrši *run-lenght* kodiranje i podatke šalje preko „*axi stream*“ i „*stream to memory maped*“ protokoa u RAM korištenjem modula za direktni pristup memoriji. U tablici 4.1. prikazano je zauzeće hardverskih resursa JPEG sustava.

Tablica 4.1. *Zauzeće hardverskih resursa sustava za vršenje kompresije slike prema JPEG normi*

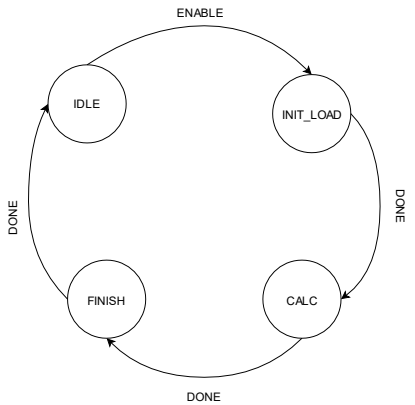
Resursi	Dostupno	Iskorišteno	Iskorišteno [%]
LUT	17600	3937	22.37
LUTRAM	6000	235	3.92
FF	35200	5630	15.99
BRAM	60	24.50	40.83
DSP	80	3	3.75

Na slici 4.2. prikazan je blok dijagram JPEG modula. U BRAM 2 modul prilikom izvršavanja sinteze, učitano je 4096 koeficijenata koji predstavljaju DCT bazne funkcije za obavljanje diskretne kosinusne transformacije. Prilikom inicijalizacije sustava u BRAM 4 se učitavaju vrijednosti kvantizacijske tablice.



Slika 4.2. *Modul JPEG kodera napravljen u sklopu ovog diplomskog rada*

Na slici 4.3. prikazan je automat s konačnim brojem stanja koji upravlja JPEG koder modulom. Sustav se početno nalazi u stanju „IDLE“ u kojemu čeka dozvolu za rad. Nakon što dobije dozvolu za rad prelazi u stanje „INIT_LOAD“ u kojemu učitava podatke o slici u BRAM. Po završetku učitavanja podataka u BRAM prelazi u stanje „CALC“ u kojemu obrađuje blokove slike. Nakon završetka obrade blokova sustav prelazi u stanje „FINISH“ u kojemu šalje preostale podatke u FIFO memoriju i nakon toga se ponovno vrača u stanje „IDLE“.



Slika 4.3. *Automat za upravljanje implementiranim JPEG koderom u ovom diplomskom radu*

4.2. Unos podataka u sustav

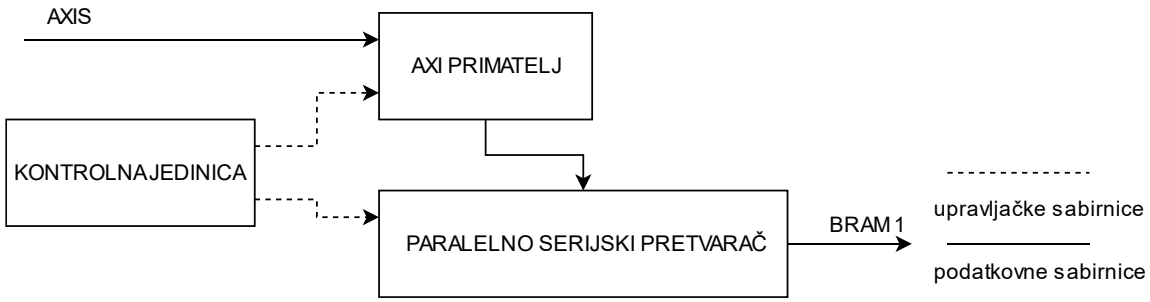
Prvi proces u izvršavanju JPEG kompresije podataka je odrediti faktor kvalitete i na osnovu njega definirati kvantizacijsku tablicu. Faktor kvalitete zadaje se u programskom dijelu izvršavanja

JPEG kompresije. Nakon što se program pokrene, on će na temelju odabranog faktora kvalitete definirati kvantizacijske koeficijente te ih preko AXI Lite protokola upisati u BRAM.

Unos podataka u sustav, s informacijama o slici, vrši se kao direktni prijenos podataka iz RAM u FIFO memoriju koja se nalazi unutar programabilne logike korištenjem „*memoy map to stream*“ i „AXI *stream*“ protokola. Prijenos podataka započinje procesor, gdje on preko DMA sustava šalje prvi podatak veličine 32 bita. Kada se prijenos dovrši aktivira se prekidna rutina. Prekidna rutina zatim šalje podatke sve dok se ne pošalje prvih osam redaka slice. Nakon završenog prijenosa prvih osam redaka slike, procesor čeka naredbu od kontrolne jedinice JPEG kodera, i započinje novi prijenos idućih 8 redaka slike, te se prethodno opisan proces ponavlja sve dok se ne prenesu svi podaci o slici.

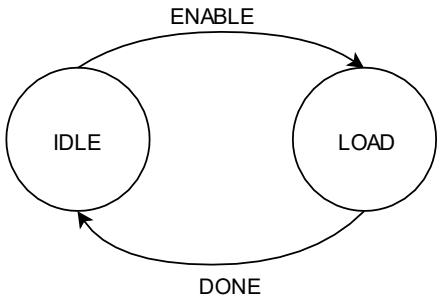
Modul za izvlačenje podataka pristiglih iz „AXI *stream“* protokola koje sprema u BRAM povlači 32-bitne podatke iz FIFO spremnika, te svaki 32-bitni vektor rastavlja na četiri osam bitna vektora koje sprema u BRAM. Blok shema modula za prijenos podataka AXI *stream-om* u BRAM prikazan je na slici 4.4.

Kontrolna jedinica ovog sustava upravlja modulom i brine se da „AXI *stream*“ protokol bude ispoštovan, odnosno podaci će se primati u trenutcima kada protokol postavi zastavicu da su podaci valjani i spremni za preuzimanje. Kontrolna jedinica dizajnirana je tako da kada dobije dozvolu za rad, u BRAM započinje spremati podatke jednog 8x8 bloka podataka.



Slika 4.4. *Modul za prijenos podataka iz AXI streama u BRAM*

Na slici 4.5. prikazan je automat s konačnim brojem stanja, koji opisuje tijek učitavanja podataka iz AXI stream protokola u BRAM. Automat se nalazi u početnom stanju nazvanom „IDLE“ te kada mu sustav omogući rad on prelazi u stanje „LOAD“. U stanju „LOAD“ učitavaju se podaci o slici i vrši se pretvorba podataka iz 32-bitnog vektora u četiri zasebna četvero bitna vektora koje sprema u BRAM. Iz stanja „LOAD“ automat izlazi kada uspješno u BRAM upiše jedan cijeli 8x8 blok slike.

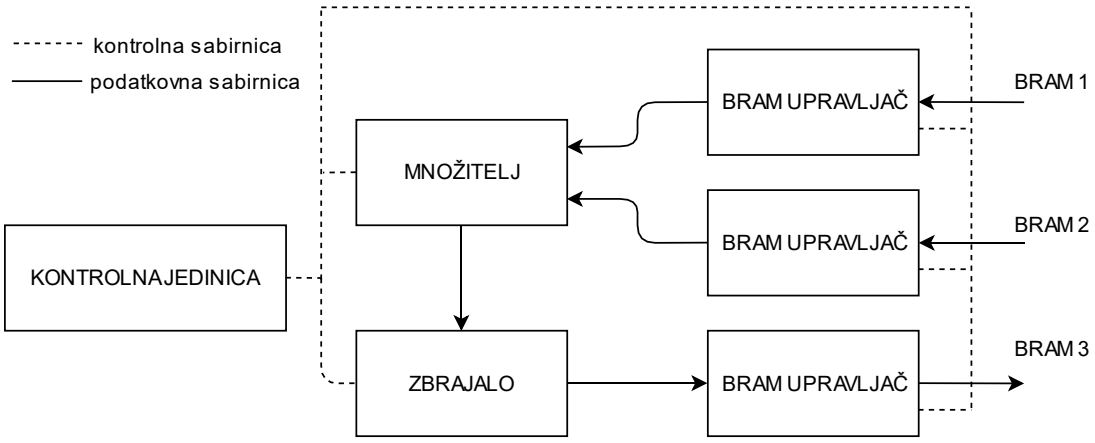


Slika 4.5. *Automat za učitavanje podataka u BRAM*

Sustavu za učitavanje podataka u BRAM potrebno je sedam taktova kako bi učitao 32-bitni podatak i pretvorio ga u četiri vektora od osam bita, te ih spremio u BRAM. Za učitavanje 8x8 bloka podataka slike, to iznosi sveukupno 112 taktova.

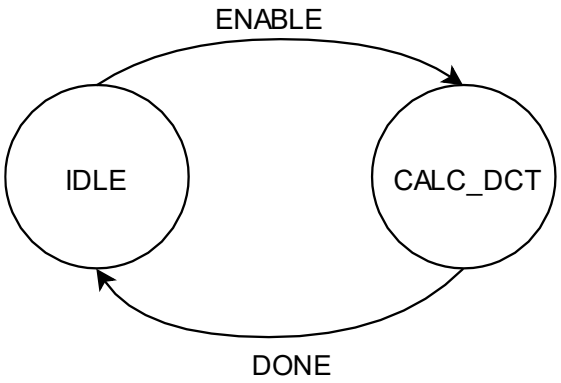
4.3. Diskretna kosinusna transformacija

Modul za provođenje diskretne kosinusne transformacije sadrži automat s konačnim brojem stanja koji je zaslužan za kontrolu izračuna diskretnih kosinusnih koeficijenata. Sustav računa diskretnu kosinusnu transformaciju po formuli (2-1). Kako bi se izbjeglo dvostruko računanje kosinusa i njihov umnožak formula je preuređena, te se svela na 4096 DCT baznih funkcija koje množe elemente slike. Konstante koje predstavljaju DCT bazne funkcije spremljene su u zaseban BRAM. Kada blok za obradu diskretnih kosinusnih transformacija dobije dozvolu za rad, učitavaju se bazne funkcije i elementi slike, svaki element slike množi se sa svojom odgovarajućom baznom funkcijom te nakon toga rezultat umnoška se sumira. Proces se ponavlja 4096 puta, sve dok se ne dobije odgovarajućih 64 DCT koeficijenata. Rezultirajući DCT koeficijenti spremaju se u zaseban BRAM, gdje postaju dostupni modulu za vršenje kvantizacije podataka. Na slici 4.6. prikazan je blok dijagram DCT modula.



Slika 4.6. *Blok dijagram modula za provođenje diskretne kosinusne transformacije*

Automat s konačnim brojem stanja, koji se nalazi u kontrolnoj jedinici za obradu diskretne kosinusne transformacije, sastoji se od dva stanja. Jedno stanje je „IDLE“ u kojemu automat čeka signal kako bi mogao započeti obradu diskretne kosinusne transformacije, a drugo stanje je stanje „CALC_DCT“ u kojemu se pokreće proces za računanje diskretne kosinusne transformacije. Automat će izaći iz stanja „CALC_DCT“ kada obradi svih 64 DCT koeficijenata. Na slici 4.7. prikazan je blok dijagram automata za računanje DCT koeficijenata.



Slika 4.7. *Automat za provođenje diskrretne kosinusne transformacije*

Proračun diskretne kosinusne transformacije traje 132 takta po DCT koeficijentu, odnosno za DCT transformaciju 8x8 bloka 8448 taktova.

DCT modul zauzima malu količinu hardverskih resursa, jer je sustav dizajniran tako da obrađuje jednu operaciju po taktu. Za matematičku operaciju množenje koristi jedan DSP modul. Hardversko zauzeće je prikazano u tablici 4.2.

Tablica 4.2. *Zauzeće hardverskih resursa DCT modula*

Resursi	Dostupno	Iskorišteno	Iskorišteno [%]
LUT	17600	157	0.89
FF	35200	219	0.62
DSP	80	1	1.25

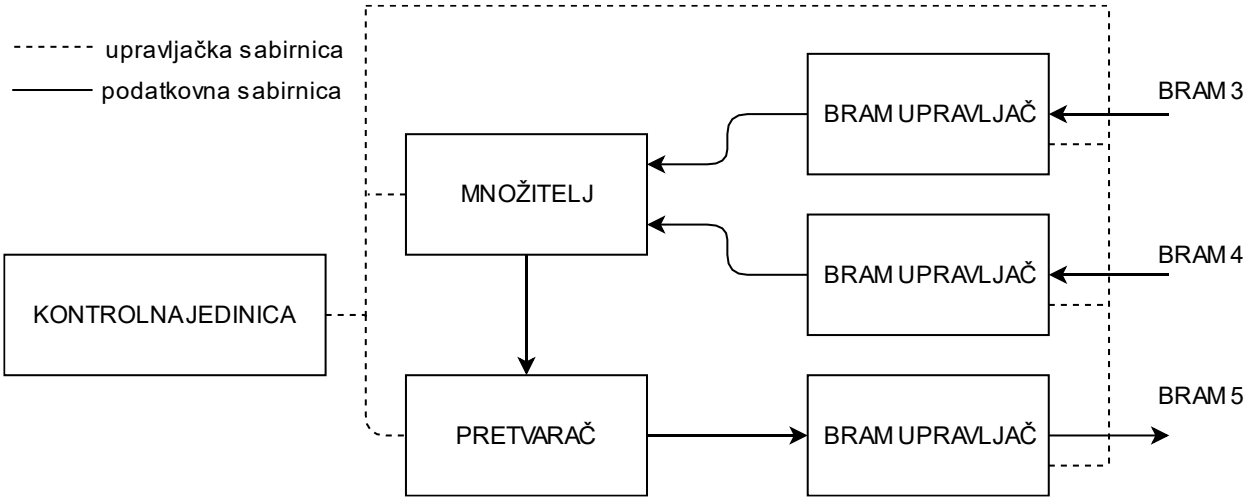
4.4. Kvantizacija

Zadaća modula za kvantizaciju je podijeliti izračunati DCT koeficijent s odgovarajućom konstantom iz kvantizacijske tablice. Hardversko dijeljenje je zahtjevnije od hardverskog množenja, i zahtijeva više vremena obrade, a kako je kvantizacijski korak unaprijed poznat

podatak, dijeljenje se može svesti na množenje s inverznom vrijednošću. Blok dijagram sustava za kvantizaciju prikazan je na slici 4.8.

Proces započinje kada kontrolna jedinica dobije dozvolu za rad, DCT podatak i podatak o kvantizacijskom koraku se učitavaju iz BRAM-a i prosljeđuju na množitelj. Nakon što množitelj izvrši množenje rezultirajući vektor je proslijeđen modulu čija je funkcija zaokružiti rezultantu vrijednost na 8 bitni cjelobrojni broj.

Rezultat pod modula za zaokruživanje sprema se u zasebni BRAM kojeg dijeli s modulom za vršenje kompresije podataka. Kontrolna jedinica određuje s koje memorijske adrese BRAM-a će se podaci učitavati i na koje memorijske adrese će se podaci spremati, stoga je logika za spremanje podataka u BRAM unutar kontrolne jedinice napravljena tako da se rezultirajući podaci spremaju na točno određene adrese čime tvore „zig-zag“ formaciju 8x8 bloka podataka.



Slika 4.8. *Blok dijagram modula za kvantizaciju napravljenog u sklopu ovog diplomskog rada*

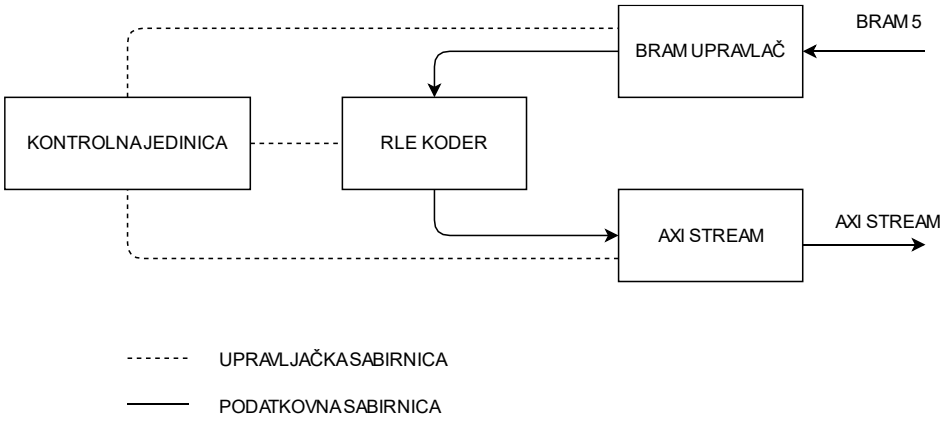
Modul za vršenje kvantizacije DCT koeficijenata, dobiva dozvolu za rad svaki puta kada modul za računanje diskretne kosinusne transformacije izračuna jedan koeficijent. Za operaciju množenja koristi dva DSP bloka, a potpuno zauzimanje hardverskih resursa se može vidjeti u tablici 4.3. Množenje DCT koeficijenta traje 4 takta.

Tablica 4.3. *Zauzeće hardverskih resursa modula za kvantizaciju*

Resursi	Dostupno	Iskorišteno	Iskorišteno [%]
LUT	17600	73	0.41
FF	35200	200	0.57
DSP	80	2	2.5

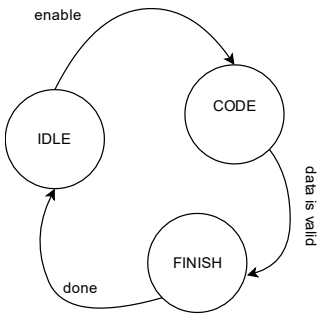
4.5. RLE koder

Run lenght koder iz BRAM-a učitava kvantizirane vrijednosti. Kvantizirane vrijednosti se zatim unose u pod modul koji se sastoji od jednog komparatora i jednog brojača. Nakon učitanog prvog podatka, svaki idući se uspoređuje s prethodnim. Ako su podaci isti, uvećava se brojač, a ako se pojavi podatak od prethodnog, sustav na izlaz daje podatak, te daje iznos koliko se puta podatak ponovio, a zatim započinje brojanje ispočetka. Kontrolna jedinica ima povratnu informaciju od komparatora te kada komparator naznači da je podatak spreman za slanje, kontrolna jedinica proslijedi podatak na pod modul koji taj podatak šalje u FIFO memoriju preko AXI stream protokola. Blok dijagram modula koji vrši „run lenght" kodiranje prikazan je na slici 4.9.



Slika 4.9. *Blok dijagram RLE kodera napravljenog u sklopu ovog diplomskog rada*

Na slici 4.10. prikazan je automat s konačnim brojem stanja koji se nalazi u kontrolnoj jedinici RLE kodera. RLE koder se nalazi u početnom stanju „IDLE", a kada dobije dozvolu za rad, prelazi u stanje „CODE". U stanju „CODE" učitava podatke iz BRAM-a, obrađuje ih i šalje u FIFO memoriju. Nakon obrade 63 od 64 podatka, RLE koder prelazi u stanje „FINISH". U stanju „FINISH" prekida se učitavanje novih podataka i odlučuje se gdje spada, odnosno kako kodirati preostali, posljednji podatak koji se nalazi u komparatoru.



Slika 4.10. *Automat za upravljanje RLE koderom*

RLE koderu potrebno je 325 ciklusa takta za kodiranje cijelog 8x8 bloka podataka, odnosno sveukupno 65 ciklusa po 5 taktova za svaki podatak. U tablici 4.4. prikazan je hardverski zahtjev ovog modula.

Tablica 4.4. *Zauzeće hardverskih resursa modula za kvantizaciju*

Resursi	Dostupno	Iskorišteno	Iskorišteno [%]
LUT	17600	46	0.26
FF	35200	81	0.23

4.6. Aplikacijski sustav

4.6.1. Opis programskih modula

Programski dio JPEG sustava sastoji se od modula za kontrolu direktnom pristupu memoriji, modula za JPEG, te glavnog modula za kontrolu.

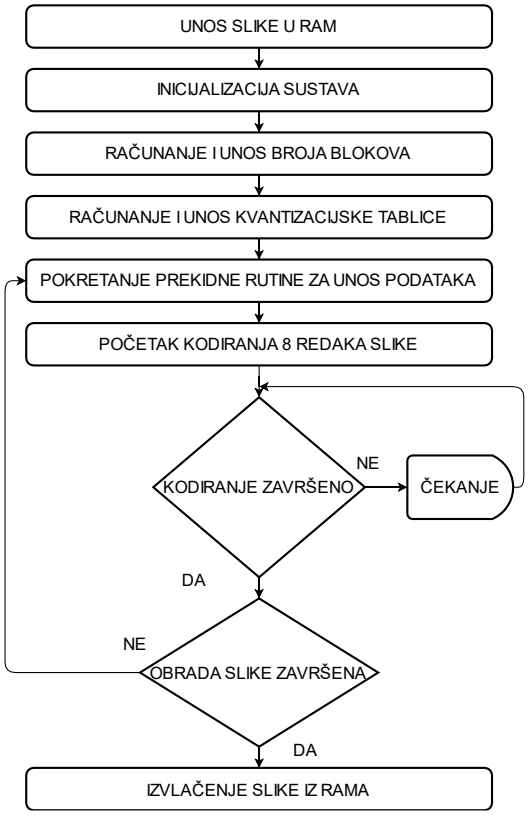
U modulu za direktni pristup memoriji nalaze se sve potrebne funkcije koje vrše inicijalnu konfiguraciju softverskog i hardverskog dijela za direktni pristup memoriji. Također, modul sadrži funkcije koje omogućuju prijenos podataka iz programskog sustava u programsku logiku. Modul za direktni pristup memoriji je postavljen tako da podatke šalje u programsku logiku preko prekidnih rutina, a podatke prima iz programske logike na zahtjev programskog sustava.

Modul za JPEG koder sastoji se od funkcija koje su potrebne za inicijalno postavljanje JPEG sustava. Modul JPEG kodera sastoji se od funkcija za postavljanje veličine slike koja će se obrađivati, odnosno broj 8x8 blokova koje koder treba obraditi. JPEG modul sadrži funkcije koje omogućuju da se na temelju unosa faktora kvalitete, vrši preskaliranje kvantizacijske tablice, te njihov unos u programsku logiku. Također JPEG koder sadrži skup funkcija koje služe za omogućavanje rada, resetiranje JPEG sustava te funkcije za provjeru kada je JPEG koder završio s radom.

Inicijalizacija sustava sastoji se od svih varijabli i struktura koje su potrebne ostalim modulima. Brine se o tome da memorija iz koje će se podaci slati ili primati preko modula za direktni pristup memoriji bude inicijalizirana i njemu dostupna. Modul za inicijalizaciju sustava poziva valjanim redoslijedom funkcije iz ostala dva modula kako bi JPEG koder mogao obaviti kodiranje.

4.6.1. Tijek programa

Nakon pokretanja rezvojne programske okoline (engl. *Software Development Kit*, SDK), najprije se učitavaju podatci o slici koja će se kodirati. slika se učitava u RAM Zybo razvojne ploče pomoću „*memory dump*“ alata koji prima datoteku ekstenzije „.bin“. Slika je pretvorena u taj oblik korištenjem Octave programskog alata. Nakon što je slika učitana u RAM, sustav može započeti s radom. Nakon učitane slike u RAM, pokrene se program koji prvo izvrši inicijalizaciju programske logike i procesora na Zybo ploči, zatim inicijalizira sustav za izravni pristup memoriji i odgovarajuće prekidne rutine koje su potrebne za rad sustava. Nakon toga pokreću se funkcije za računanje broja 8x8 blokova, koje JPEG sustav treba obraditi. Na temelju faktora kvalitete preskalira se zadana kvantizacijska tablica i preko AXI Lite protokola upiše u BRAM 4. Nakon svih inicijalnih postavki sustava, programski sustav započinje s upisom podataka iz RAM-a u BRAM-e korištenjem direktnog pristupa memoriji, na način da se upisuje 8 redaka, blok po blok veličine 8x8. Programski sustav zatim uključuje JPEG koder i čeka isti da obradi podatke. Kada JPEG sustav obradi podatke, programski sustav će preko izravnog pristupa memoriji iščitati kodirane podatke iz memorije te ih spremiti u RAM. Proces se ponavlja sve dok cijela slika ne bude komprimirana. Na slici 4.11. prikazan je dijagram toka programa JPEG kodera.



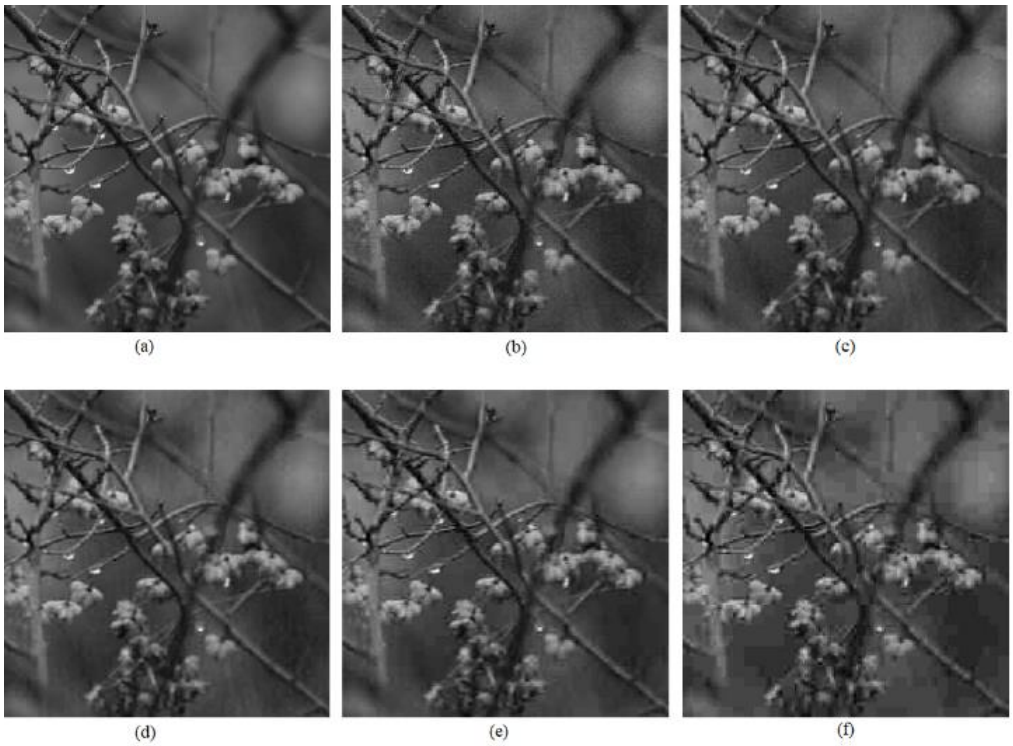
Slika 4.11. *Dijagram toka programa za upravljanje sustavom za kodiranje slike prema JPEG normi.*

5. REZULTATI MJERENJA IMPLEMENTIRANOG JPEG KODERA NA FPGA

Mjerenja su vršena nad slikama rezolucije 256x256, 512x512, i 1024x1024 elemenata slike. Za svaku rezoluciju mjerenja su se vršila za sliku koje sadrže puno i za sliku s malo detalja. Za svaku rezoluciju i za koeficijente kvalitete 100, 80, 60, 40, 20, mjerila se stvarna brzina obrade podataka, cjelokupna obrada podataka, te faktor kompresije odnosno omjer broja bitova potrebnih za pohranu izvorne slike i broja bitova potrebnih za pohranu komprimirane slike. Stvarna brzina obrade podataka je vrijeme koje je potrebno JPEG koderu za obradu slike. Ukupna brzina obrade podataka je vrijeme koje je potrebno da cijeli sustav obradi sliku. Ono uključuje stvarnu obradu podataka, prijenos podataka o slici iz RAM-a u BRAM, učitavanje kvantizacijske tablice u sustav, te postavljanje inicijalnih parametara i modula JPEG sustava.

5.1. Testiranje implementiranog JPEG kodera na slikama

Na slici 5.1. prikazana je slika „Grana“, rezolucije 256x256 elemenata slike, s odgovarajućim koeficijentima kvalitete. Slika sadrži malo detalja, te su zbog toga gubitci nastali kompresijom slike izrazito vidljivi. U tablici 5.1. prikazani su rezultati mjerenja za ovu sliku.

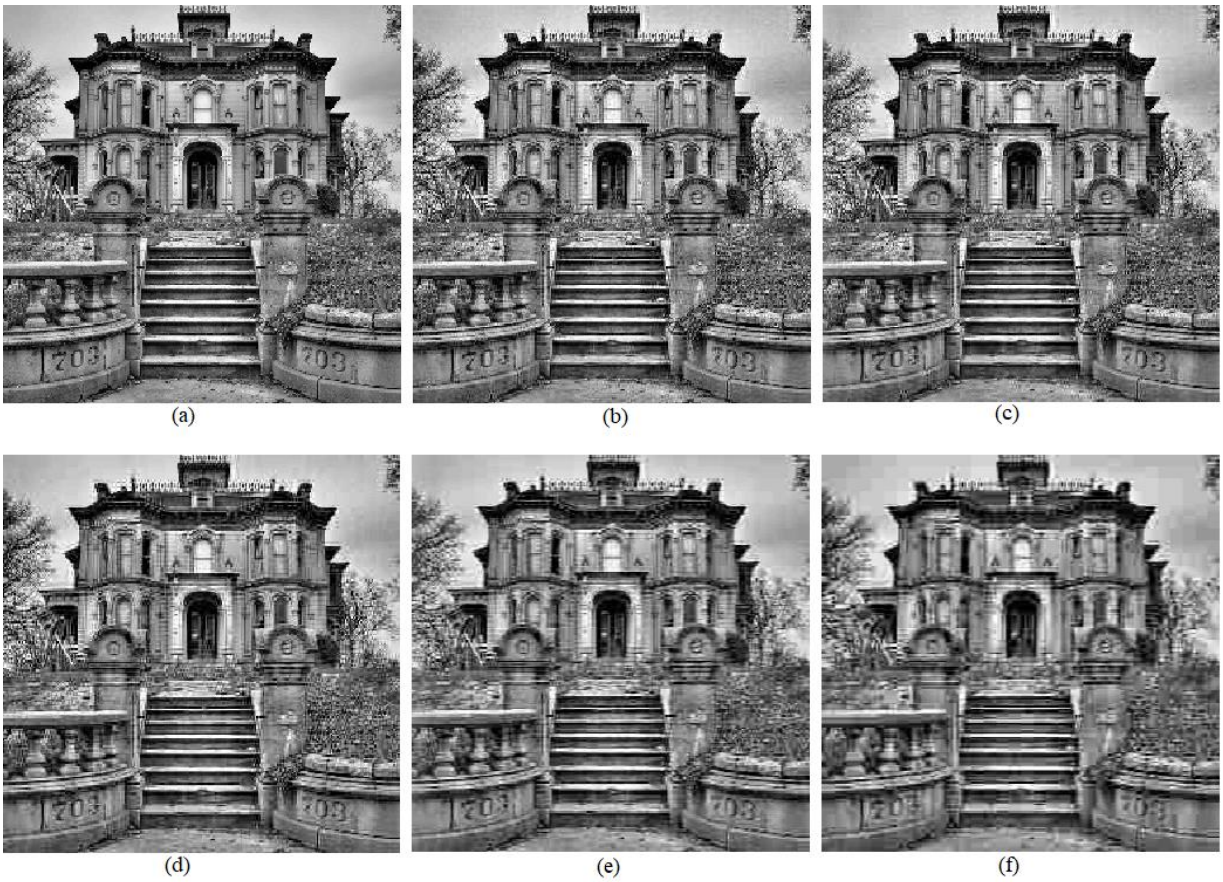


Slika 5.1. *Slika „Grana“ rezolucije 256x256 elemenata slike, (a) izvorna slika; te slike komprimirane s faktorom kvalitete: (b) 100, (c) 80, (d) 60, (e) 40, (f) 20.*

Tablica 5.1. *Rezultati mjerenja za sliku „Grana“, rezolucije 256x256 elemenata slike*

Kvaliteta	Stvarna brzina	Ukupna Brzina	Memorijsko zauzeće	Faktor kompresije
IZVORNA	-	-	524 288 bita	-
100	209.92 ms	2234 ms	187 464 bita	2.80
80	209.92 ms	2234 ms	168 696 bita	3.11
60	209.92 ms	2234 ms	148 236 bita	3.54
40	209.92 ms	2234 ms	125 388 bita	4.18
20	209.92 ms	2234 ms	99 588 bita	5.26

Slika „Dvorac“, prikazana na slici 5.2. je rezolucije 256x256 elemenata slike i sadrži puno detalja. Artefakti izobličenja nastali kompresijom slike su manje izraženi nego kod slika s malo detalja, ali su vidljiva izobličenja na prijelazima. Rezultati obrade podataka prikazani su u tablici 5.2.

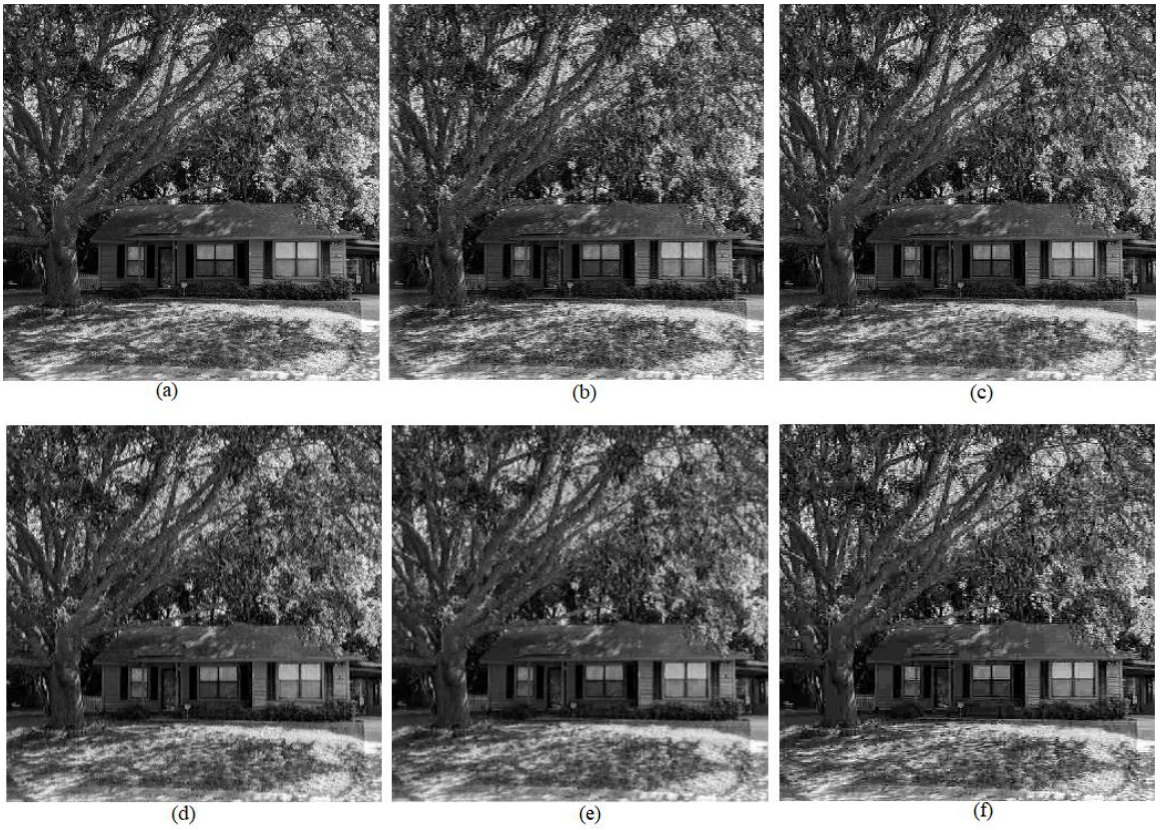


Slika 5.2. *Slika „Dvorac“ rezolucije 256x256 elemenata slike, (a) izvorna slika; te slike komprimirane s faktorom kvalitete: (b) 100, (c) 80, (d) 60, (e) 40, (f) 20.*

Tablica 5.2. *Rezultati mjerenja za slike „Dvorac“ rezolucije 256x256 elemenata slike*

Kvaliteta	Stvarna brzina	Ukupna Brzina	Memorijsko zauzeće	Faktor kompresije
IZVORNA	-	-	524 288 bita	-
100	209.92 ms	2235 ms	348 108 bita	1.51
80	209.92 ms	2234 ms	301 428 bita	1.74
60	209.92 ms	2234 ms	260 424 bita	2.01
40	209.92 ms	2234 ms	214 080 bita	2.45
20	209.92 ms	2234 ms	152 928 bita	3.43

Na slici 5.3. prikazana je slika „Kuća“ rezolucije 512x512 elemenata slike koja sadrži puno detalja. Slika sadrži toliko visokofrekventnih komponenti, da značajna izobličenja nastaju tek pri faktoru kvalitete 20, no zbog toga nije postignut veliki faktor kompresije slike. U tablici 5.3. mogu se vidjeti rezultati mjerenja.



Slika 5.3. *Slika „Kuća“ rezolucije 512x512 elemenata slike(a) izvorna slika; te slike komprimirane s faktorom kvalitete: (b) 100, (c) 80, (d) 60, (e) 40, (f) 20.*

Tablica 5.3. Rezultati mjerenja za sliku „Kuća“ rezolucije 512x512 elemenata slike

Kvaliteta	Stvarna brzina	Ukupna Brzina	Memorijsko zauzeće	Faktor kompresije
IZVORNA	-	-	2 097 152 bita	-
100	839.20 ms	8172 ms	1 470 156 bita	1.43
80	839.20 ms	8172 ms	1 090 548 bita	1.92
60	839.20 ms	8172 ms	1 026 660 bita	2.04
40	839.20 ms	8172 ms	812 700 bita	2.58
20	839.20 ms	8172 ms	572 928 bita	3.66

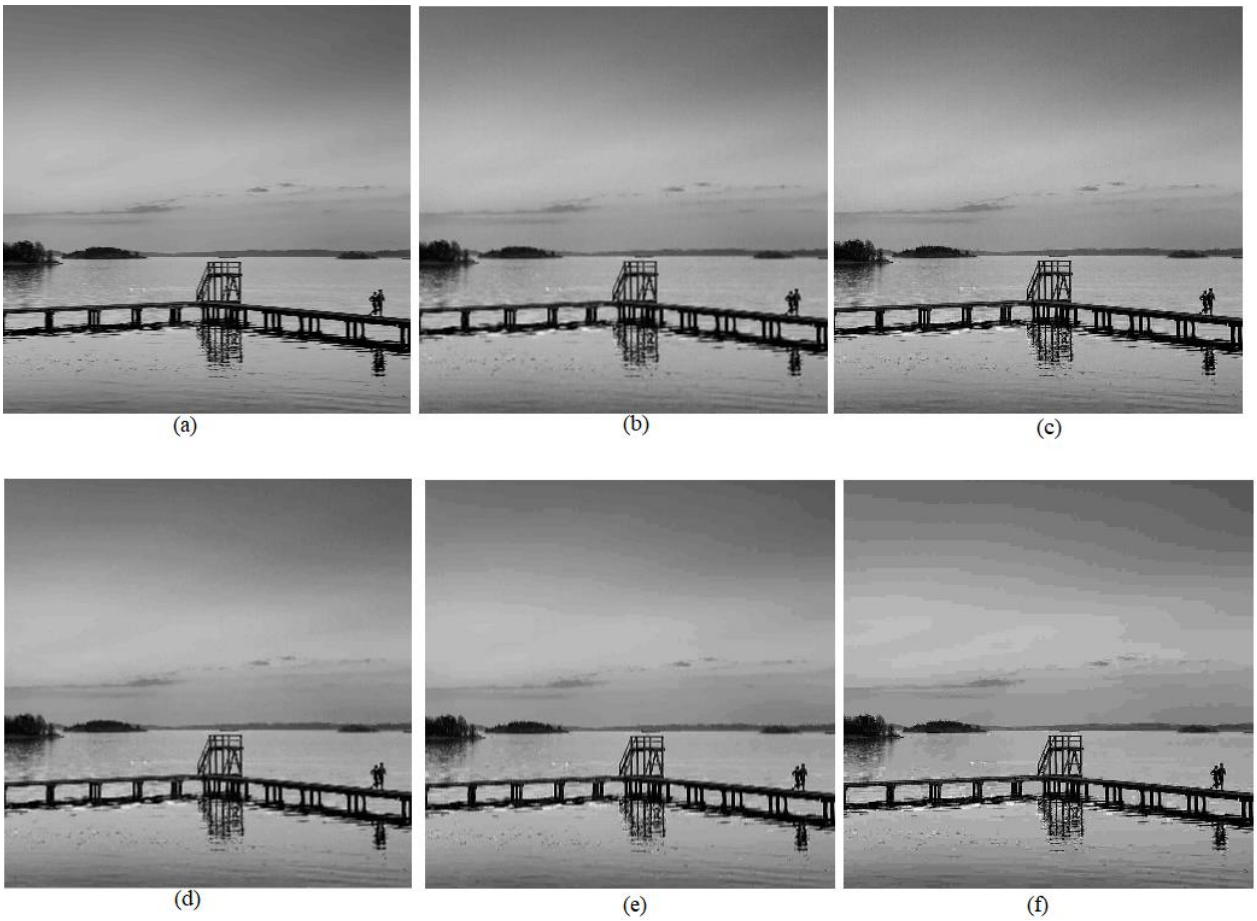
Na slici 5.4. prikazana je slika „Lena“ za različite faktore kvalitete. Vidljiva izobličenja nastala prilikom kompresije značajno su vidljiva tek pri faktoru kvalitete 60. u tablici 4. prikazani su rezultati mjerenja za danu sliku.

Slika 5.4. *Slika „Lena“ rezolucije 512x512 elemenata slike, (a) izvorna slika; te slike komprimirane s faktorom kvalitete: (b) 100, (c) 80, (d) 60, (e) 40, (f) 20.*

Tablica 5.4. *Rezultati mjerenja za sliku „Lena“ rezolucije 512x512 elemenata slike*

Kvaliteta	Stvarna brzina	Ukupna Brzina	Memorijsko zauzeće	Faktor kompresije
IZVORNA	-	-	2 097 152 bita	-
100	839.20 ms	8172 ms	684 840 bita	3.06
80	839.20 ms	8172 ms	603.288 bita	3.47
60	839.20 ms	8172 ms	508 464 bita	4.12
40	839.20 ms	8172 ms	418 836 bita	5.01
20	839.20 ms	8172 ms	339 216 bita	6.18

Na slici 5.5. prikazana je slika „Most“ rezolucije 1024x1024 elemenata slike. Slika sadrži malo detalja, iz tog razloga postiže se velika kompresija. Artefakti nastali kompresijom postaju izraženi tek pri kvaliteti 40. rezultati mjerenja prikazani su u tablici 5.5.

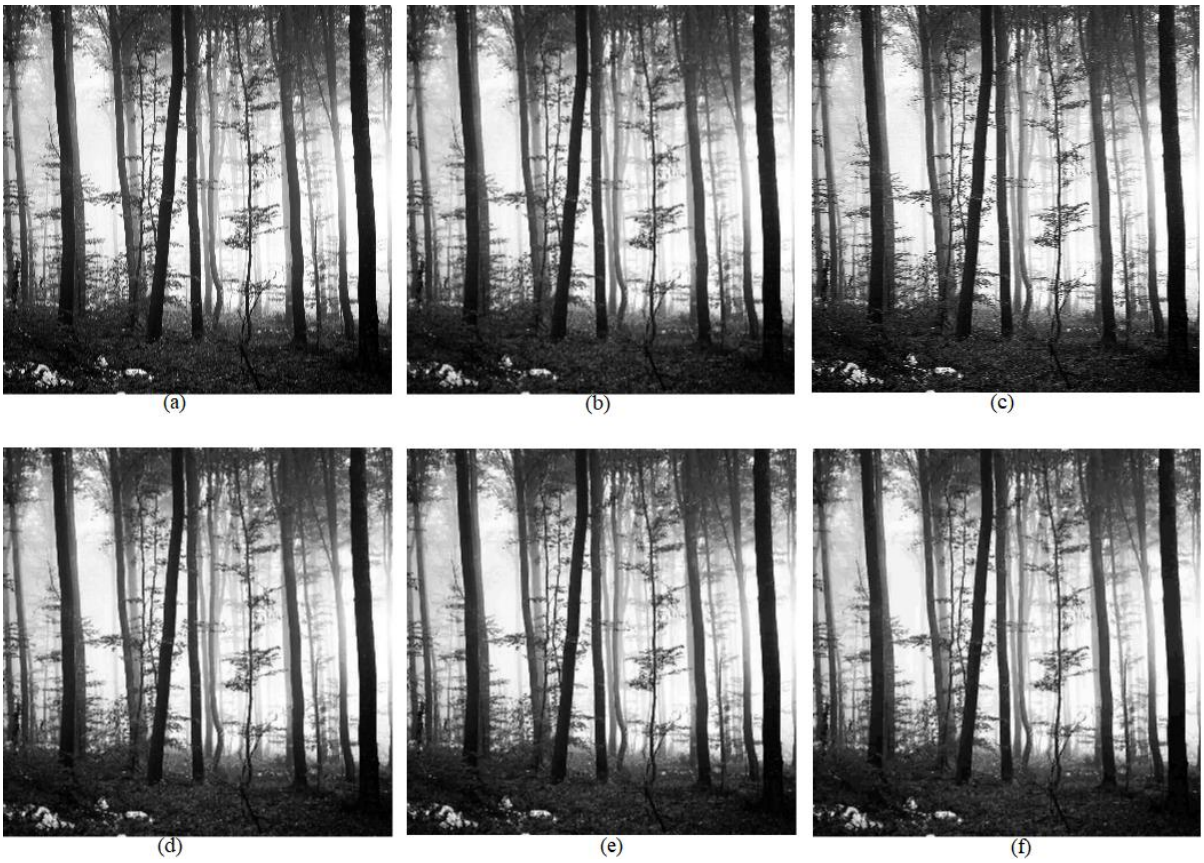


Slika 5.5. *Slika „Most“ rezolucije 1024x1024 elemenata slike, (a) izvorna slika; te slike komprimirane s faktorom kvalitete: (b) 100, (c) 80, (d) 60, (e) 40, (f) 20.*

Tablica 5.5. *Rezultati mjerenja za sliku „Most“ rezolucije 1024x1024 elemenata slike*

Kvaliteta	Stvarna brzina	Ukupna Brzina	Memorijsko zauzeće	Faktor kompresije
IZVORNA	-	-	8 388 608 bita	-
100	3357.44 ms	31268 ms	2 040 000 bita	4.11
80	3357.44 ms	31268 ms	1 861 884 bita	4.51
60	3357.44 ms	31268 ms	1 533 192 bita	5.47
40	3357.44 ms	31268 ms	1 331 208 bita	6.30
20	3357.44 ms	31268 ms	1 179 324 bita	7.11

Na slici 5.6. prikazana je slika „Šuma“ rezolucije 1024x1024 elemenata slike, koja sadrži puno detalja. Faktor kompresije je značajno manji od slika koje sadrže malo detalja, ali nema značajno vidljivih izobličenja nastalih kompresijom. Rezultati su prikazanu u tablici 5.6.



Slika 5.6. Slika „Šuma“ rezolucije 1024x1024 elemenata slike, (a) izvorna slika; te slike komprimirane s faktorom kvalitete: (b) 100, (c) 80, (d) 60, (e) 40, (f) 20.

Tablica 5.6. *Rezultati mjerenja za sliku „Šuma“ rezolucije 1024x1024 elemenata slike*

Kvaliteta	Stvarna brzina	Ukupna Brzina	Memorijsko zauzeće	Faktor kompresije
IZVORNA	-	-	8 388 608 bita	-
100	3357.44 ms	31268 ms	4 256 448 bita	1.97
80	3357.44 ms	31268 ms	3 355 272 bita	2.50
60	3357.44 ms	31268 ms	2 987 448 bita	2.80
40	3357.44 ms	31268 ms	2 406 624 bita	3.49
20	3357.44 ms	31268 ms	1 801 968 bita	4.66

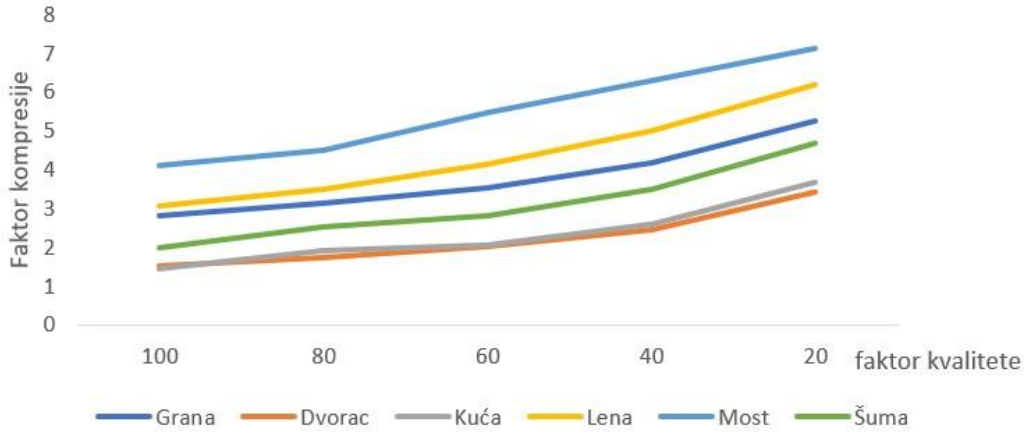
5.2. Analiza postignutih rezultata

Vrijeme potrebno za obradu slike ne mijenja se između obrade slika iste rezolucije i ne ovisi o stupnju kompresije, ali vrijeme obrade podataka raste eksponencijalno s povećanjem rezolucije. Grafikon ovisnosti vremena obrade podataka o rezoluciji, prikazan je na slici 5.7. Vrijeme obrade podataka odnosi se na vrijeme koje je potrebno koderu za kodiranje podatka, a ukupno vrijeme označava cjelokupnu obradu slike s uključenim vremenom za učitavanje podataka.

Slika 5.7. *Ovisnost vremenskog trajanja obrade podataka o rezoluciji slike*

Komprimiranje slike utječe i na kvalitetu slike. Što je kompresija slike veća to će kvaliteta slike biti manja. Na slici 5.8. prikazana je ovisnost faktora kompresije o faktoru kvalitete. Iz grafikona je vidljivo da slike koje sadrže puno detalja su za isti faktor kvalitete su postigle manji

omjer kompresije, dok su slike s manje detalja postigle veći omjer kompresije za istu kvalitetu slike.



Slika 5.8. *Prikaz faktora kompresije u ovisnosti o faktoru kvalitete*

ZAKLJUČAK

U radu je uspješno implementiran JPEG koder, kojemu se konfiguracijski parametri unose procesorskim sustavom, a obrada slike se vrši hardverskim sustavom. JPEG koder može obrađivati slike do veličine rezolucije 1024x1024 elemenata slike te sama obrada podataka pri toj rezoluciji traje 3357.44 ms, a ukupna obrada podataka koja uključuje i unos podataka traje 31.268 s.

Analizom i mjerenjima je ustvrđeno kako JPEG koder izvršava svoju funkcionalnost, jer se mjereni rezultati ponašaju prema očekivanju. Temeljem rezultata i analize ispunjen je zahtjev za dizajniranjem i implementacijom JPEG kodera na FPGA sustavu.

Implementirani JPEG koder na FPGA sustavu može se poboljšati u vidu brzine kodiranja slike i veličine faktora kompresije slike. Veća brzina kodiranja može se postići globalnom paralelizacijom sustava, odnosno paralelnom obradom više različitih 8x8 blokova slike ili ubrzanjem obrade pojedinog segmenta JPEG kodera, poput diskretne kosinusne transformacije. Vrijeme prijenosa podataka može se smanjiti implementacijom više-kanalnog DMA sustava ili postavkom DMA sustava u „*schater gather*“ načinu rada. Faktor kompresije može se poboljšati implementacijom naprednih algoritama za kompresiju slike.

LITERATURA

[1] G. K. Wallace, *The JPEG Still Picture Compression Standard*, IEEE Transactions on Consumer Electronics, Vol. 38, No. 1, FEBRUARY 1992.

[2] S. A. Khayam, *The Discrete Cosine Transform*, Department of Electrical & Computer Engineering Michigan State University March 10th 2003

[3] J. Kornblum, *Using JPEG Quantization Tables to Identify Imagery Processed by Software*, The Digital Forensic Research Conference DFRWS 2008 USA Baltimore, MD (Aug 11th - 13th).

[4] A. Malsha, D. G. Bailey, A. Punchihewa, *Exploring the Implementation of JPEG Compression on FPGA*, School of Engineering and Advanced Technology Massey University Palmerston North, New Zealand.

[5] *Design of Embeded Processors*, https://nptel.ac.in/courses/Webcourse-contents/IIT%20Kharagpur/Embedded%20systems/Pdf/Lesson-20.pdf, zadnji pristup 1.12.2018

[6] Vivado razvojno okruženje, https://www.xilinx.com/products/design-tools/vivado.html, zadnji pristup 1.12.2018

[7] AXI DMA, https://www.xilinx.com/support/documentation/ip_documentation/axi_dma/v7_1/pg021_axi_dma.pdf, zadnji pristup 1.12.2018

[8] Dijagram toka dizajna FPGA, https://www.researchgate.net/figure/FPGA-design-flow_fig2_228875713, zadnji pristup 1.12.2018

SAŽETAK

U radu je opisana izrada i implementacija JPEG kodera zasnovanog na FPGA tehnologiji. Objašnjene su sve tehnologije potrebne za rad sustava, te postupak dizajniranja i funkcionalnosti svakog pojedinog modula. Nad sustavom su izvršena mjerenja, u kojima su se mjerila brzina obrade slike, te postignuti omjer kompresije za različite faktore kvalitete slike. Sustav se dizajnirao u Vivado razvojnom okruženju na Zynq-7000 SoC-u, gdje je procesor unosio konfiguracijske parametre a hardverski sustav vršio obradu slike. Rad sustava testiran je na više različitih slika, gdje se pokazalo da sustav ispunjava sve zadatke propisane JPEG normom.

Ključne riječi : JPEG, FPGA, VHDL, Zybo, Zynq.

Implementation of JPEG coder based on FPGA

ABSTRACT

The paper describes the development and implementation of the JPEG coder based on the FPGA. All the technologies needed to operate the system, as well as the design and functionality of each module are explained. Measurements were performed over the system, measuring the speed of the image processing, and the compression ratio achieved for different image quality factors. The system was designed in the Vivado development environment on the Zynq-7000 SoC, where the processor introduced configuration parameters, and the hardware system processed the image. The system operation was tested on several different images, where it was shown that the system fulfilled all the tasks prescribed by the JPEG standard.

Key words: JPEG, FPGA, VHDL, Zybo, Zynq.

ŽIVOTOPIS

Igor Valek rođen je 01.03.1994 godine u Virovitici. Završio je osnovnu školu Ivana Nepomuka Jemeršića u Grubišnom Polju, nakon toga upisuje srednju Tehničku školu Daruvar, smjer elektrotehnika. Srednju školu završava s maturalnim radom „Stereo FM odašiljač“, a po završetku srednje škole upisuje preddiplomski studij Elektrotehnike na Elektrotehničkom fakultetu Osijek. Na istom fakultetu upisuje diplomski studij smjer Komunikacijske tehnologije. Kroz diplomski studij započinje suradnju s Institutom RT-RK Osijek, gdje se kroz diplomski rad usavršava u FPGA području.