H.264-Bringer:

Dokumentacija

# Predgovor

U ovoj dokumentaciji opisani su postupci programiranja kodeka H.264 isključivo s tehničkog stajališta. Prethodno čitanju ove dokumentacije, kao teorijsku pozadinu preporuča se proučiti (ne nužno pročitati) diplomske radove koji opisuju ovaj kodek te u njima spomenutu literaturu.

# Važne globalne varijable

typedef struct {

int Lwidth,Lheight;

int Cwidth,Cheight;

unsigned char \*L, \*C[2];

} frame\_type;

**frame\_type**: Sadrži najvažnije podatke o slici. Članovi strukture su:

* **\*L** – niz luminantnih uzoraka slike (Y). Kardinalnost niza jednaka je broju piksela u slici, odnosno W\*H, gdje su W i H redom širina i visina slike u pikselima.
* **\*C[2]**–dva niza krominantnih uzoraka slike (Cb i Cr). C[0] odgovara komponenti Cb, a C[1] komponenti Cr. Kardinalnost svakog niza odgovara kardinalnosti krominantnih uzoraka u slici prema poduzorkovanju 4:2:0, odnosno (W/2)\*(H/2) , gdje su W i H redom širina i visina slike u pikselima.
* **Lwidth, Lheight, Cwidht, Cheight** – redom broj luminantnih uzoraka po širini i visini slike te broj krominantnih uzoraka po širini i visini slike. Vrijede jednakosti Cwidth = Lwidth / 2 i Cheight = Lheight / 2. Lwidth i Lheight su uvijek višekratnici broja 16, a Cwidth i Cheight su uvijek višekratnici broja 8. Ukoliko ulazne slike u koder nisu ovakvih dimenzija, one se obrezuju pri učitavanju.

**frame\_type frame**:Sadrži najvažnije podatke o slici koja se trenutno obrađuje.

**int CurrMbAddr** – indeks makrobloka koji se trenutno obrađuje. Raspon vrijednosti koje ova varijabla može poprimiti je [0, N–1], gdje je N broj makroblokova u slici.

# Komponente programa

## fer\_h264.cpp

Ova datoteka sadrži ulaznu točku u program. U njoj postoje samo dvije funkcije – decode() i encode() koje pokreću redom dekoder odnosno koder.

### void decode()

Upravlja dekodiranjem ulaznog toka sintakse H.264. Otvara ulazni i izlazni tok. Pokreće funkcije za upravljanje dinamičkim kadrovima (resursima), što uglavnom uključuje inicijaliziranje i oslobađanje dinamički alocirane memorije. Najvažnija zadaća ove funkcije je dohvaćanje po jedne NAL-jedinice i njezino predavanje funkciji RBSP\_decode u kojoj se odvija dekodiranje same sintakse H.264.

## nal.cpp

Upravljanje NAL-jedinicama u koderu i dekoderu. Komunikacija s tokom podataka koji sadrži sintaksu H.264. U jednoj se NAL-jedinici nalazi jedan skup parametara ili jedna čitava slika – nije podržana podjela slika na odsječke.

### Globalne varijable i strukture podataka

typedef struct

{

bool forbidden\_zero\_bit;

unsigned int nal\_ref\_idc, nal\_unit\_type, NumBytesInRBSP;

unsigned char \*rbsp\_byte;

}

NALunit;

**NALunit**: Struktura koja sadrži sve značajne podatke o NAL-jedinici. Značenja i nazivi varijabli jednaki su onima u Normi (vidi poglavlje 7.4 u Normi).

**NALunit \*NALbytes**: niz koji sadrži oktete trenutne NAL-jedinice.

### void getNAL(unsigned long \*fPtr, NALunit &nu)

**void fPtr**: kazaljka na trenutni oktet unutar ulaznog toka. Ova se kazaljka koristi kao parametar offset u pozivu STL funkcije fseek. fPtr se predaje kao referenca.

**NALunit &nu**: primjerak strukture NALunit, sadrži izdvojene podatke iz NAL jedinice koja će se sljedeća dekodirati. nu se predaje kao referenca.

Funkcija pronalazi veličinu trenutne NAL-jedinice u ulaznome toku pozivom funkcija findNALstart odnosno findNALend. Učitava NAL-jedinicu u niz NALbytes te ju predaje funkciji parseNAL koja podatke razvrstava u strukturu nu. Po završetku funkcije getNAL u strukturi nu se nalaze značajni podatci iz trenutne NAL-jedinice. Ova se funkcija poziva isključivo u dekokderu.

### void parseNAL(unsigned int NumBytesInNALunit,

### NALunit &nal\_unit);

**unsigned int NumBytesInNALunit**: broj učitanih okteta sadržanih u globalnome nizu NALbytes.

**NALunit nal\_unit**: struktura u koju se popunjavaju interpretirani podatci u kontekstu NAL-jedinice pročitani iz niza NALbytes. nal\_unit se predaje kao referenca.

Odvaja podatke o NAL-jedinici od podataka RBSP-a i upisuje ih u strukturu nal\_unit. Pronađeni kodovi za izbjegavanje emulacije (0x000003) se izbacuju prije prijepisa podataka u niz nal\_unit.rbsp\_byte. Ova se funkcija poziva isključivo u dekokderu.

### void writeNAL(NALunit nu)

**NALunit nu**: struktura koja predstavlja kodiranu NAL-jedinicu koja će se zapisati u izlazni tok.

nu je ulazna struktura te su svi podatci već u njoj zapisani. Ova funkcija ispisuje te podatke u odgovarajućemu formatu u izlazni tok. Kako bi se smanjila učestalost pristupanja datoteci, NAL-jedinice se privremeno zapisuju u međuspremnik streamBuffer veličine streamBufferSize pozivom funkcije writeToBuffer. Za inicijalizaciju međuspremnika streamBuffer potrebno je jednom, pri pokretanju programa pozvati funkciju InitNAL. Ova se funkcija poziva isključivo u koderu.

## rbsp\_decoding.cpp

Upravljanje najvažnijim postupcima dekodera: entropijskim dekodiranjem, predviđanjem, inverznom transformacijom i dekvantizacijom, učitavanjem ostataka, itd. Ova se komponenta koristi samo u dekoderu.

### void RBSP\_decode(NALunit nal\_unit)

**NALunit nal\_unit**: struktura u koju se popunjavaju interpretirani podatci u kontekstu NAL-jedinice pročitani iz niza NALbytes. nal\_unit se predaje kao referenca.

U ovisnosti o tipu NAL-jedinice (varijabla nal\_unit.nal\_unit\_type) dekodira podatke iz RBSP-a. Podržani tipovi NAL-jedinica su 1, 5, 7 i 8, koji redom odgovaraju tipovima ne-IDR, IDR, skup parametara niza i skup parametara slike. Nakon učitavanja skupova parametara inicijaliziraju se potrebni podatci poput dimenzija slike, potrebne memorije i sličnoga. Izvedeno dekodiranje za svaki od navedenih tipova izravno odgovara pseudokodovima opisanima u poglavlju 7.3 Norme. Svako odstupanje od Norme komentirano je u kôdu. U većini slučajeva odstupanja se pojavljuju zbog ograničenja profila *baseline*.

Osim učitavanja sintaksnih elemenata opisanog u poglavlju 7.3, u ovoj se funkciji pozivaju i funkcije za interpretaciju učitanih podataka. Prvo se izvodi predviđanje (intra – funkcija intraPrediction ili inter – funkcije DeriveMVs i Decode, ovisno o tipu makrobloka), zatim inverzna transformacija i dekvantizacija te rekonstrukcija slike (funkcije transformDecodingIntra\_16x16Luma, transformDecoding4x4Luma-Residual i transformDecodingChroma) i njen ispis u izlazni, nekomprimirani format Y4M[[1]](#footnote-1) (funkcija writeToY4M). Funkcije initialisationProcess i modificationProcess zaslužne su za upravljanje nizom referentnih slika kao što je opisano u poglavlju 8.2.4 Norme. Trenutno je podržano samo predviđanje u odnosu na neposredno prethodnu sliku pa se ovim funkcijama samo sprema posljednje dekodirana slika kao referentna za sljedeću P-sliku.

## rbsp\_encoding.cpp

Upravljanje najvažnijim postupcima kodera: entropijskim dekodiranjem, predviđanjem, transformacijom i kvantizacijom, upisivanjem ostataka, itd. Ova se komponenta koristi samo u koderu.

### void RBSP\_encode(NALunit &nal\_unit)

**NALunit &nal\_unit**: struktura u koju se upisuje (u član nal\_unit.rbsp\_byte) RBSP jedinica koja će se kodirati.

Kodiranje pojedine RBSP jedinice. Funkcija razlikuje tri načina kodiranja za tri vrste NAL-jedinica: skup parametara niza (SPS), skup parametara slike (PPS) i odsječak slike – I ili P. Svaka slika se kodira kao jedan odsječak. Odluka o tipu NAL-jedinice donosi se izvan ove funkcije. Nakon kodiranja skupa parametara niza (funkcija sps\_write), inicijaliziraju se potrebne varijable i alocira memorija.

Svaka IDR NAL-jedinica je I-slika i svaka ne-IDR NAL-jedinica je P-slika i sastoji se isključivo od P-makroblokova (iako norma dopušta i I-makroblokove unutar P-odsječaka). Ukoliko je trenutna NAL-jedinica I-slika, prije kodiranja zaglavlja poziva se funkcija IntraCL, koja pokreće intra kodiranje na GPU, ukoliko je u sustavu pronađena GPU koja podržava OpenCL.

Kodiranje slika odvija se po makroblokovima, stoga je glavna petlja unutar kodiranja odsječaka petlja po svim makroblokovima slike. U svakom se koraku petlje provodi predviđanje intra (funkcija intraPredictionEncoding) ili inter (funkcija interEncoding), transformacija i kvantizacija, entropijsko kodiranje parametara predviđanja i ostataka. Nakon transformacije i kvantizacije, poziva se funkcija setCodedBlockPattern koja postavlja varijable CodedBlockPatternLuma i CodedBlockPatternChroma. Ove dvije varijable govore u kojoj četvrtini makrobloka postoje ne-nul transformacijski koeficijenti. Za detaljnije objašnjenje ovih varijabli, vidi poglavlje 7.4 Norme.

Funkcije initialisationProcess i modificationProcess zaslužne su za upravljanje nizom referentnih slika kao što je opisano u poglavlju 8.2.4 Norme. Trenutno je podržano samo predviđanje u odnosu na neposredno prethodnu sliku pa se ovim funkcijama samo posljednje dekodirana slika sprema kao referentna za sljedeću P-sliku.

Na kraju kodiranja pojedine RBSP jedinice, poziva se funkcija RBSP\_trailing\_bits, kako je opisano u pseudokodu u poglavlju 7.3.2.11 Norme.

### unsigned int coded\_mb\_size(int intra16x16PredMode, int predL[16][16], int predCb[8][8], int predCr[8][8])

**int intra16x16PredMode**: odabrani način unutarslikovnog predviđanja 16×16, ako je vrijednost ove varijable -1, odabrano je predviđanje 4×4 te su načini za pojedine podblokove zapisani u globalnome nizu Intra4x4PredMode.

**int predL[16][16]**: predviđeni uzorci luminantne komponente.

**int predCb[8][8], int predCr[8][8]**: predviđeni uzorci krominantnih komponenata.

**povratna vrijednost**: broj bitova potrebnih za zapis trenutnog makrobloka uz odabrani način predviđanja.

Funkcija emulira dio funkcije RBSP\_encode, no umjesto zapisa bitova nakon entropijskog kodiranja, ona samo broji koliko je bitova potrebno za zapis te ukupan zbroj vraća kao povratnu vrijednost. Ova se funkcija koristi pri odabiru tipa makrobloka (4×4 ili 16×16) u unutarslikovnome predviđanju.

## intra.cpp

Skup svih postupaka korištenih za unutarslikovno predviđanje. Uključuje funkcije dekodera i kodera.

### void intraPrediction(int predL[16][16], int predCr[8][8], int predCb[8][8])

**int predL[16][16]**: predviđeni uzorci luminantne komponente.

**int predCb[8][8], int predCr[8][8]**: predviđeni uzorci krominantnih komponenata.

Funkcija obavlja unutarslikovno predviđanje za trenutni makroblok na osnovi učitanih parametara predviđanja i dosada dekodiranih makroblokova. Funkcija strogo poštuje postupke opisane u poglavlju 8.3 Norme, te su u kodu naznačena odgovarajuća potpoglavlja za svaki postupak.

Za predviđanje intra 4×4 potrebni su načini predviđanja iz susjednih podmakroblokova. Zbog toga se oni spremaju u globalno polje Intra4x4PredMode koje je kardinalnosti jednakoj ukupnom broju mogućih podmakroblokova 4×4 u slici. Indeks pojedinog načina predviđanja unutar tog polja jednak je CurrMbAddr \* 16 + luma4x4BlkIdx. Za makroblokove tipa intra 16×16 te vrijednosti nisu postavljene. Tipovi makroblokova nalaze se u globalnome polju mb\_type\_array, indeks tipa pojedinog makrobloka jednak je indeksu toga makrobloka u slici (npr. za trenutni makroblok to je CurrMbAddr). Budući da makroblokovi tipa intra 16×16 ne postavljaju vrijednosti u polju Intra4x4PredMode, za dohvat načina predviđanja susjednih podblokova u predviđanju načina predviđanja intra 4×4 prvo se provjerava je li makroblok kojemu pripada susjedni podblok tipa intra 4×4.

Kod izvedbe predviđanja intra 4×4, svaki se podblok mora odmah rekonstruirati (unutar postupka predviđanja za trenutni makroblok) jer rekonstruirani uzorci moraju biti dostupni kao uzorci za predviđanje u narednim podmakroblokovima. Zbog toga se na kraju izvedbe predviđanja za svaki podblok 4×4 poziva funkcija transformDecoding4x4LumaResidual.

Ako je trenutni makroblok tipa intra 16×16, u ovoj se funkciji izvodi samo izračun predviđenih uzoraka koji se kasnije predaju funkciji za transformaciju i kvantizaciju, koji inverzno transformirane i dekvantizirane ostatke pribrajaju ovim uzorcima i time rekonstruiraju makroblok.

Ova se funkcija poziva isključivo u dekoderu.

### int intraPredictionEncoding(int predL[16][16],

### int predCr[8][8], int predCb[8][8])

**int predL[16][16]**: predviđeni uzorci luminantne komponente.

**int predCb[8][8], int predCr[8][8]**: predviđeni uzorci krominantnih komponenata.

Funkcija za kodiranje i izvedbu načina predviđanja za trenutni makroblok. Razlozi odabranih postupaka opisani su u radu „Izvedba i optimiranje intra predikcije za normu MPEG-4 part 10“. Za svaki se makroblok odmah provodi i dekodiranje te se rezultantnim makroblokom prepisuje izvorni makroblok u izvornoj slici (varijabla frame). U funkciji postoje dva rješenja kodiranja u ovisnosti o tome je li u sustavu na kojemu se program izvodi pronađena GPU sukladna s OpenCL-om.

**Ako se ne koristi OpenCL**, provodi se svaki mogući način predviđanja 16×16 za luminantne uzorke. Najbolji način odabire se funkcijom SATD. Ukoliko za neki način nisu dostupni svi uzorci za predviđanje način se preskače. Funkcijom coded\_mb\_size provjerava se broj bitova potrebnih za zapis makrobloka za odabrano predviđanje intra 16×16.

Nakon odabira najboljeg načina intra 16×16, odabiru se najbolji načini predviđanja intra 4×4 za 16 podblokova trenutnoga makrobloka. Zatim se postavljaju parametri za predviđanje 4×4 funkcijom setIntra4x4PredMode i provodi se samo predviđanje. Budući da su za predviđanje podmakroblokova potrebni rekonstruirani uzorci iz susjednih podmakroblokova, u svakome se podmakrobloku za najbolji pronađeni način odmah provodi i rekonstrukcija podmakrobloka. Za dobivene se uzorke predviđanja ponovno poziva funkcija coded\_mb\_size te ukoliko je za predviđanje intra 4×4 potrebno manje bitova nego za predviđanje 16×16, odabire se ovo predviđanje te nije potrebno više provoditi rekonstrukciju kodiranog makrobloka, budući da je ona sastavni dio provedbe predviđanja 4×4 kako je napomenuto prethodno. S druge strane, ako je manje bitova potrebno za predviđanje 16×16, ponovno se mora izvesti predviđanje za odabrani način i rekonstruirati makroblok. Međutim, provedba predviđanja 4×4 zbog nužne rekonstrukcije prepisala je uzorke izvornoga makrobloka dekodiranima. Zbog toga se prije provedbe predviđanja 4×4 izvorni makroblok sprema u privremeno polje original[16][16], prema kojemu se ponovno provodi predviđanje 16×16 ukoliko je odabrano kao bolje.

Za krominantne komponente odabire se način predviđanja ekvivalentan najboljem pronađenom načinu 16×16 (ne izvodi se posebna pretraga najboljeg načina).

**Ako se koristi OpenCL**, u ovoj se funkciji preskače postupak pronalaska najboljeg načina predviđanja, budući da se on provodi na GPU. Odabrani načini predviđanja zapisani su u nizovima predModes16x16 odnosno predModes4x4. Za svaki se makroblok uspoređuje samo broj bitova potrebnih za zapis makrobloka najboljim načinom 16×16 i najboljim načinom 4×4.

Ova se funkcija poziva isključivo u koderu.

## openCL\_functions.cpp

Sadrži sve funkcije za upravljanje OpenCL-om. Ako nije pronađena GPU koja podržava OpenCL, globalna varijabla OpenCLEnabled postavlja se u false i program ne koristi OpenCL (ne koristi se emulacija na CPU, iako je moguća).

### void InitCL()

Funkcija za inicijalizaciju sučelja OpenCL. Potrebno ju je pokrenuti samo jednom pri pokretanju programa. Funkcija pronalazi platformu OpenCL na sustavu, traži GPU koja podržava OpenCL te ako ju ne pronalazi, varijabla OpenCLEnabled se postavlja u false, funkcija završava i program ne koristi OpenCL. Ako je pronađena adekvatna GPU, stvara se kontekst i učitavaju se jezgrice iz datoteka h264\_kernels.cl i intra\_kernels.cl te se prevode za pronađenu karticu. Za svaku jezgricu postavlja se globalna kazaljka kojom se pokreće pojedina jezgrica kada je to potrebno.

### void IntraCL()

Pokreće odabir načina predviđanja za svaki makroblok u trenutnoj slici na GPU. Luminantna komponenta trenutne slike (varijabla frame.L) preslikava se u memoriju GPU. Postavljaju se ulazni parametri jezgrica i pokreću se jezgrice. Pokreće se i učitavanje rezultata (načini predviđanja 16×16 i 4×4), ali u neblokirajućem načinu, što znači da po povratku iz funkcije podatci možda još nisu učitani u izlazne nizove (predModes4x4 i predModes16x16), dapače, u ovome slučaju sigurno nisu. Pri pozivu funkcije za učitavanje rezultata, predaju se parametri eventReadPredModes4x4 i eventReadPredModes16x16, koji predstavljaju događaje učitavanja. Rezultati stoga unutar ove funkcije (IntraCL) nisu učitani, već se neposredno prije njihovog korištenja (u funkciji intraPredictionEncoding) poziva funkcija clWaitForEvents[[2]](#footnote-2) koja blokira izvođenje programa dok rezultati nisu u potpunosti učitani s GPU.

## intra\_kernels.cl

Funkcije u ovoj datoteci većinom su identične onima u datoteci intra.cpp, uz neke nužne preinake zbog SIMD arhitekture GPGPU.

Komunikacija s globalnom memorijom na GPU vrlo je skupa, stoga ju treba što je više moguće smanjiti. Iz toga se razloga, na početku svake od dviju jezgrica (GetIntra4x4PredModes i GetIntra16x16PredModes) uzorci izvornog makrobloka odnosno podbloka dohvaćaju u lokalno polje original[16][16], odnosno original[4][4].

Budući da su računske jedinice na GPU SIMD strojevi, preskakanje izračuna predviđanja i SATD-a za makroblokove ili podblokove za načine za koje nisu dostupni svi potrebni uzorci za predviđanje nema smisla, jer se ionako vrijeme troši na dretvama u kojima su mogući svi načini predviđanja (vidi rad „Izvedba i optimiranje intra predikcije za normu MPEG-4 part 10“). Dapače, zato što se u tokovnim jezgrama mora izvesti svako grananje, ovo znatno usporava prevođenje jezgrica, pa se grananja moraju što više izbjegavati. Stoga se za nemoguće načine predviđanja ocjena funkcije SATD jednostavno postavlja na INT\_MAX.

Najznačajnija promjena u odnosu na funkcije korištene u datoteci intra.cpp jest vektorizacija transformacije i kvantizacije. Umjesto polja 4x4, matrice korištene za ove funkcije predstavljene su kao nizovi od četiri vektora int4. Odluka o tome predstavljaju li vektori retke ili stupce matrica donesena je u ovisnosti o smjeru paralelnosti operacija, te je komentirana u kodu.

1. Više o formatu Y4M na <http://wiki.multimedia.cx/index.php?title=YUV4MPEG2> [↑](#footnote-ref-1)
2. Točnije, it funkcije intraPredictionEncoding poziva se funkcija WaitIntraCL koja je omotač za funkciju clWaitForEvents. [↑](#footnote-ref-2)