SADRŽAJ

SADRŽAJ 1
Particionisanje 2
Podela sistema – funkcionalna dekompozicija. Izgled virtuelne platforme 2
Interconnect modul 5
DMA (Direct Memory Access) modul 6
IP modul 7
BRAM modul 9
Hardversko-softverski interfejs 9
Registarska mapa 11
Registar IP modula – IP_CFG_REG 11
Registri DMA modula 11
BRAM modul 12
Literatura 13

Particionisanje

Podela sistema – funkcionalna dekompozicija. Izgled virtuelne platforme

Korisnički interfejs softverskih modela predstavlja veb aplikacija koja pruža korisniku da izabere lokacije između kojih se pronalazi optimalna putanja. Unete lokacije mogu da se resetuju i ponovo iznova može da se pokrene upit o optimalnoj putanji. Kada korisnik završi sa korišćenjem uredjaja, klikom na dugme *izlaz* zatvara aplikaciju. U izvršnoj specifikaciji i softverskom modelu pred particionisanje geografske koordinate trenutno unetih lokacija se zapisuju u tekstualnu datoteku `input_file.txt`. Odatle se dalje koriste u `main.cpp` izvornim datotekama modela. Izvorne datoteke vrše upit optimalne putanje i potom upisuju geografske koordinate temena putanje u `output_file.txt`, odakle se one prikazuju na mapi u veb aplikaciji.

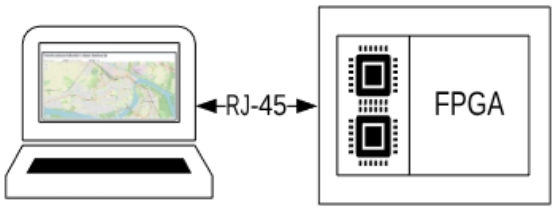
Slika 1 Korisnički interfejs

Veb aplikacija omogućava lakšu interakciju između korisnika i izvršnog fajla. Do sada, u dokumentima Specifikacija i Analiza pre particionisanja, posmatran je samo deo sistema koji je vršio relaksacije. Nakon kompajliranja izvornih C++ fajlova, izvršni fajl se pokreće unutar JavaScript aplikacije (veb aplikacije).

Po pokretanju aplikacije, učitava se mapa sa slike 1 na lokalnom serveru i čeka se unos lokacija. Validne unete lokacije se zapisuju u tekstualni datoteku i pokreće se izvršni fajl koji obavlja relaksacije. Izvršene relaksacije se smeštaju u tekstualnu datoteku, odakle ih JavaScript aplikacija koristi kako bi na mapi iscrtala optimalnu putanju.

Veb server aplikacije će se pokretati na jednom od dva jezgra Zybo procesora, dok će drugo obavljati čitanje iz tekstualne datoteke `input_file.txt`, pretragu najbližih temena grafa unetim lokacijama, sinhronizaciju akcija u sistemu i upis koordinata putanje u

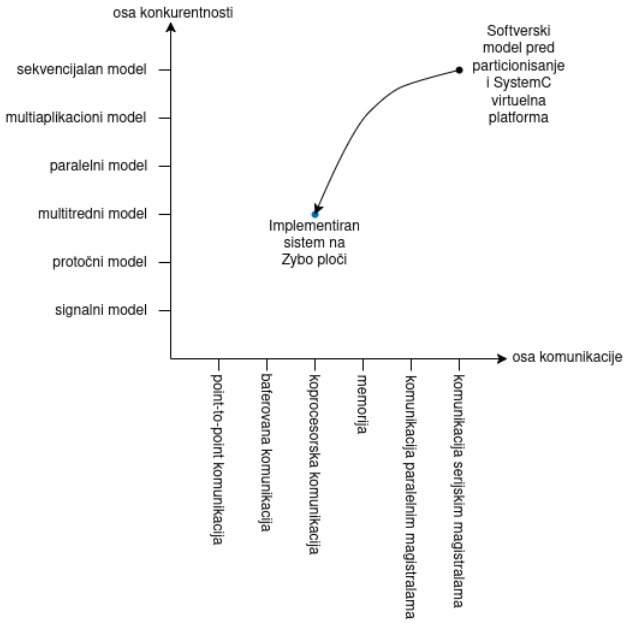
`output_file.txt`. Veb aplikacija je prevedena u C++ korišćenjem `httplib.h` zaglavlja. Korisnički interfejs biće prikazan u veb pretraživaču računara koji je povezan standardnim Ethernet kablom sa Ethernet PHY modulom Zybo Z7 10 pločice.



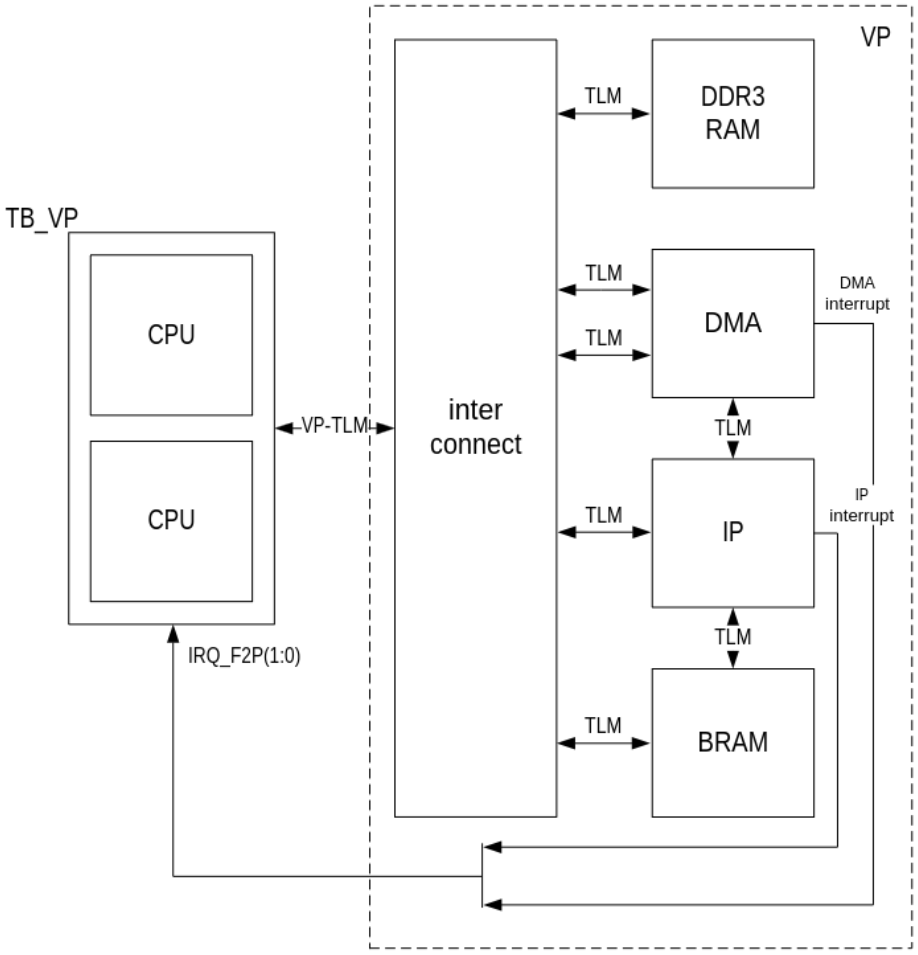
Slika 2 Ilustracija sistema nakon particionisanja

Vremenski najkritičniju funkciju, `relaxationProcess`, je potrebno mapirati na hardverske resurse Zybo Z7 10 pločice. Parametri ove funkcije, podaci o mreži Novog Sada, se smeštaju u BRAM ćelije. Ostatak programabilne logike može pristupiti BRAM ćelijama dosta brže nego što procesor pristupa DRAM memoriji. Ovim se smanjuje broj ciklusa po pristupu podacima mreže.

U ovom dokumentu se objašnjava virtuelna platforma za razvoj softvera na procesoru ZYNQ-7000 SoC koji se nalazi na pločici Zybo Z7 10. Prilikom modelovanja ovog sistema, korišćen je TLM 2.0 standard za povezivanje. Unutar virtuelne platforme, kasnije i implementacije, `relaxationProcess` se modeluje IP modulom. Ovim će se od klasičnog softvera razviti **multinitan sistem sa koprocesorskom komunikacijom**.

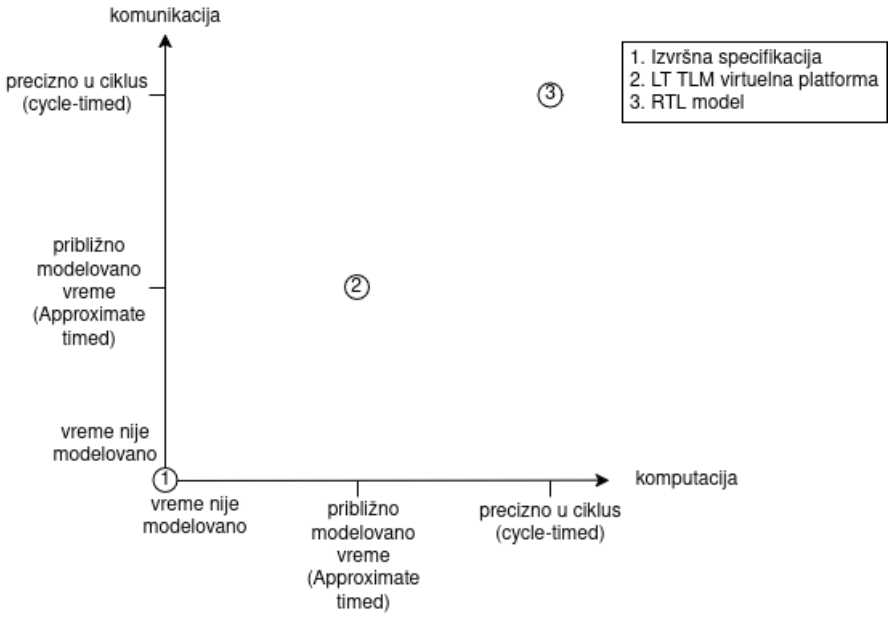


Slika 3 Particionisan sistem nakon implementacije na osama taksonomije



Slika 4 Virtuelna platforma

Ipak, za sada se još samo u softveru modeluje virtuelna platforma. Virtuelna platforma je modelovana LT stilom pomoću SystemC biblioteke. Ova C++ biblioteka nam omogućava da simuliramo hardver na nivou diskretnih događaja. Koristi se zajednički model hardvera i softvera u SystemC simulatoru. **Prilikom klasifikacije ovog modela po vremenskoj osi, može se govoriti o događajima koji su uzročno - posledično povezani. Na osi podataka, model govori o vrednostima.**

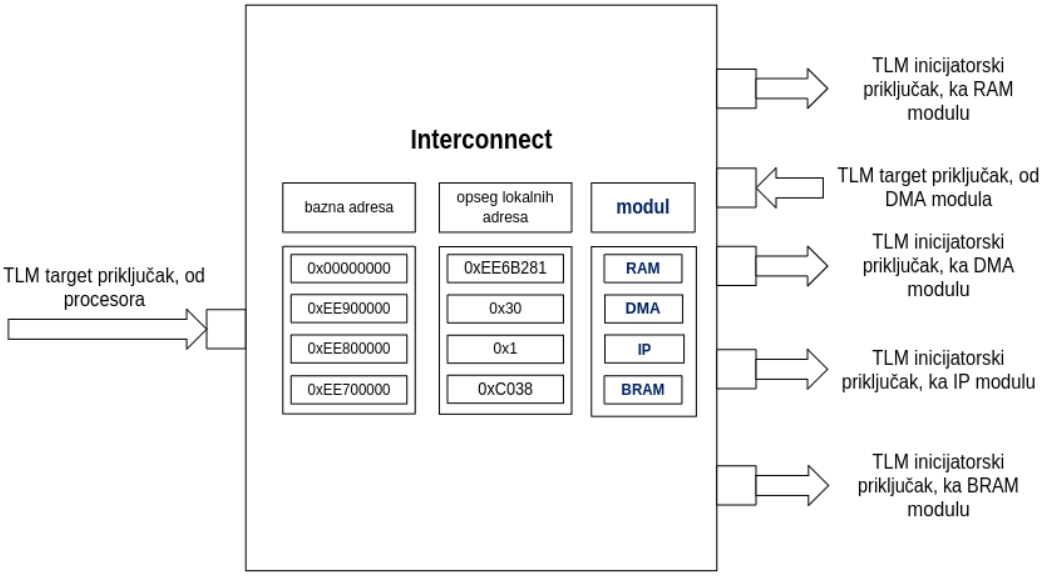


Slika 5 Evolucija sistema na osama komunikacije i proračuna

Model se nije menjao po osi konkurentnosti i kofigurabilnosti u odnosu na model pred particionisanje (slika 3)**. Zbog korišćenja LT stila, sistem je sekvencijalan** [2][3].

Interconnect modul

Interconnect modul je komponenta koja povezuje DDR3L RAM, IP, DMA i BRAM modul i omogućava procesoru da im pristupa preko jednog standardnog TLM priključka. Moduli su memorijski mapirani, odnosno postoji jedinstveni adresni opseg unutar kojeg svaki modul dobija jedinstvenu baznu adresu i opseg lokacija koji pokriva [1]. Ovim moduli, sa stanovišta procesora predstavljaju samo lokacije u toj jedinstvenoj memoriji. Na lokacijama dobijenim sabiranjem bazne adrese modula i lokalne adrese se nalaze SAHE elementi (eng. Software Accessible Hardware Elements) modula koji su interconnect modulom objedinjeni u virtuelnu platformu [2][3]. U trenutku kada procesor šalje TLM paket ka SAHE elementu određenog modula, on šalje paket ka interconnect modulu na adresu jedinstvenog adresnog opsega. Uloga interconnect modula je da na osnovu adrese paket usmeri ka ciljnom modulu, tako što pretvori adresu jedinstvenog adresnog opsega u lokalnu adresu ciljnog modula. Za svaki modul koji povezuje, interconnect poznaje baznu adresu i njegov opseg lokalnih adresa (slika 6) [1]. Takođe, upis podataka u RAM pomoću DMA modula obavlja se kroz interconnect modul.



Slika 6 Interconnect modul

DMA (Direct Memory Access) modul

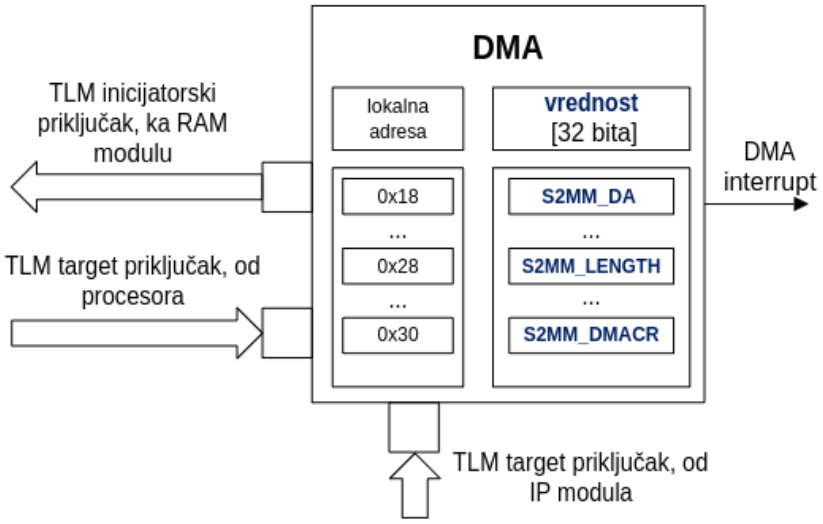
DMA modul modeluje Direct Memory Access kontroler. On omogućava IP modulu da šalje ili prima podatke iz RAM memorije bez direktnog posredstva procesora. Procesor konfiguriše DMA modul upisom u odgovarajuće memorijski mapirane SAHE. SAHE elemente u ovom slučaju procesor konfiguriše tako da DMA modul bude u režimu direktnog adresiranja, jer su podaci kojima se pristupa kontinualni u memoriji. Procesor će saznati da je prenos podataka gotov tako što će mu DMA modul poslati prekid.

U ovom sistemu, DMA modul će se koristiti da izvršene relaksacije upiše iz bafera u IP modulu direktno u RAM memoriju. Pošto DMA modul prenosi podatke od IP modula do RAM memorije, procesor konfiguriše 32-bitne SAHE S2MM_DMACR, S2MM_DA i S2MM_LENGTH. SAHE se moraju podesiti sledećim redosledom:

1. DMA modul se resetuje upisom logičke jedinice na drugi bit SAHE-a S2MM_DMACR,
2. Pokreće se S2MM kanal upisom logičke jedinice na prvi bit SAHE-a S2MM_DMACR,
3. Omogućavaju se prekidi upisom logičkih jedinica na 12. i 14. bit SAHE-a S2MM_DMACR,
4. U SAHE S2MM_DA se upisuje validna početna adresa RAM memorije od koje DMA modul upisuje podatke koje mu IP modul šalje,

5. U SAHE S2MM_LENGTH se upisuje broj bajtova koji se DMA modulom šalje u RAM.

SAHE-i DMA modula su prikazani na slici 7.



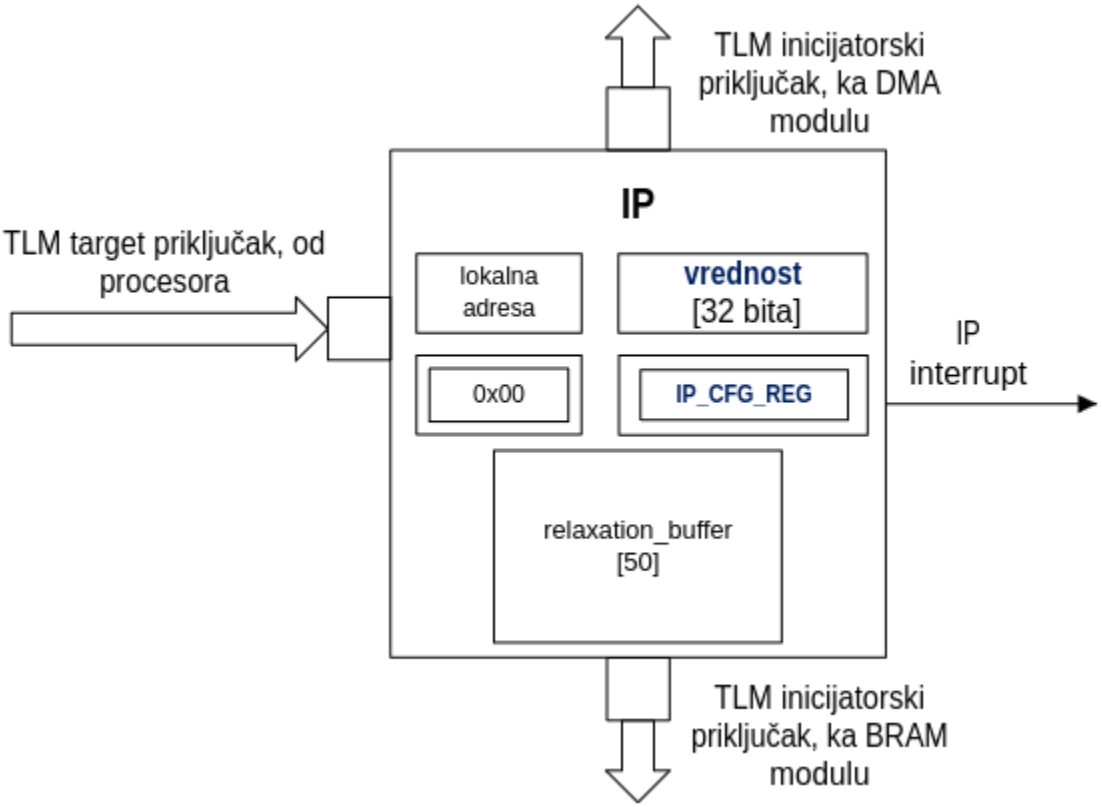
Slika 7 DMA modul

IP modul

Preko IP_CFG_REG SAHE elementa procesor upravlja IP modulom. U trenutku kada procesor pronađe najbliža temena grafa trenutno unetim lokacijama i shodno njima inicijalizuje BRAM memoriju, proces relaksacije u IP modulu može da počne. Najpre se modul resetuje upisom jedinice na nulti bit SAHE-a. Resetom se svi biti SAHE-a postavljaju na nulu. Sa procesom relaksacija se započinje upisom jedinice na prvi bit IP_CFG_REG SAHE-a.

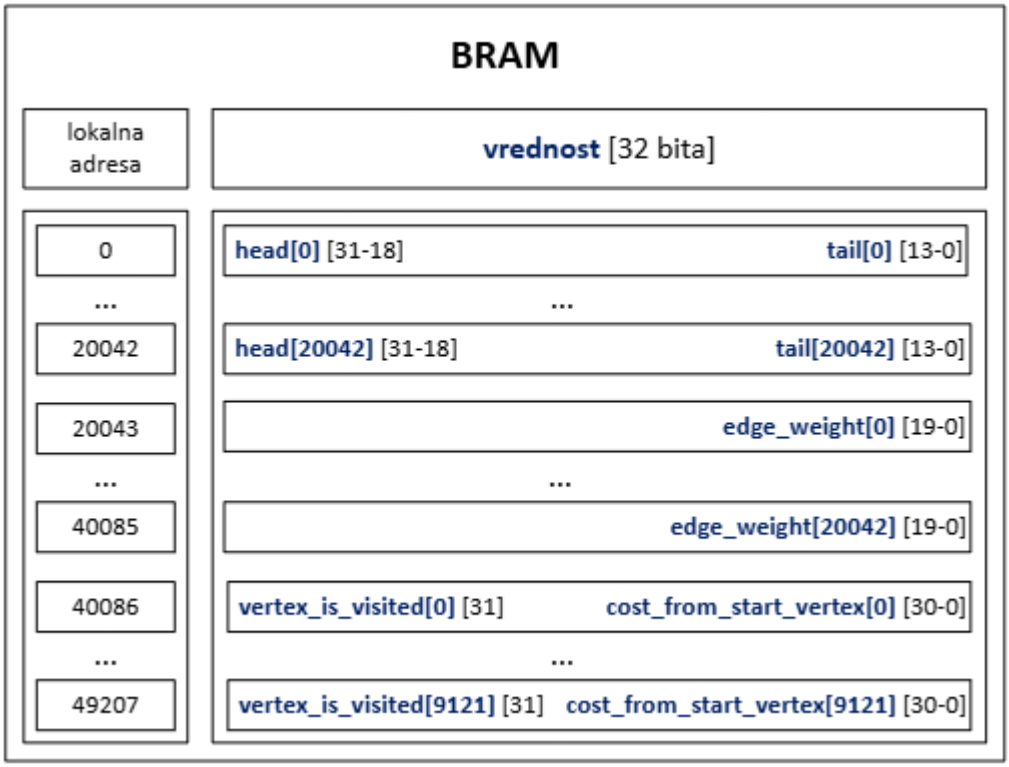
Modul vrši relaksacije i ažurira BRAM, a izvršene relaksacije skladišti u privremeni "relaksacioni bafer" koji je u stanju da primi 50 32-bitnih vrednosti. Format u kome se relaksacije skladište su 32 bitne vrednosti u čijih gornjih 14 bita se smešta identifikacioni broj temena koje je izvršilo relaksaciju, a u donjih 14 bita identifikacioni broj temena nad kojim je relaksacija izvršena. Kada se relaksacioni bafer napuni, IP modul postavlja drugi bit IP_CFG_REG SAHE-a na jedinicu, a treći na nulu. Drugi bit, IP_DMA_REQ_BIT, se postavlja na jedinicu u slučaju da se traži zahtev od procesora za pražnjenje relaksacionog bafera. Treći bit, IP_DMA_STREAM_BIT, signalizira da je IP modul u stanju slanja relaksacija DMA modulu i on je na nuli dok procesor ne postavi DMA modul u režim diretnog adresiranja. Na interrupt izlaz se potom postavlja visoka vrednost. Procesor u prekidnoj rutini u ovom slučaju konfiguriše DMA modul i postavlja IP_DMA_STREAM_BIT na jedinicu tako da IP modul može da počne sa procesom upisa relaksacija iz relaksacionog bafera u DMA modul.

Po završetku prenosa, IP modul postavlja IP_DMA_STREAM_BIT na nulu i nastavlja sa procesom relaksacija. Kada se dostigne kraj procesa, ukoliko bafer nije prazan on se najpre isprazni po opisanom postupku. Potom se četvrti bit, IP_END_BIT, postavlja na jedinicu i interrupt izlaz se postavlja na visoku vrednost.



Slika 8 IP modul

BRAM modul



Slika 9 Konfiguracija BRAM memorije

BRAM memorijski modul skladišti podatke o grafu. Sačinjen je od 49207 32-bitnih lokacija. Postoji segment memorije (od lokalne adrese 0 do 40086) koji predstavlja nepromenljive podatke – izvore, ponore ivica grafa i težine ivica. U ovom segmentu memorije je moguće samo čitanje. Nakon ovog segmenta, u BRAM-u se nalaze podaci koje procesor i IP modul ažuriraju pri svakom upitu optimalne putanje. Ovi podaci čine segment BRAM-a koji je u dokumentu Specifikacija, pri opisu algoritma, nazvan tabela neposećenih temena. Na slici 9 je prikazana konfiguracija BRAM modula i način na koji su podaci raspoređeni u 32-bitnim lokacijama.

Hardversko-softverski interfejs

Hardversko-softverski interfejs u ovom projektu se odvija preko prekida (eng. interrupts), zajedničke memorije i registara. DMA i IP prekidi su objašnjeni u prethodnim odeljcima. Processor prima dvobitni IRQ_F2P signal i na osnovu njegove vrednosti preko General Interrupt kontrolera (GIC) određuje koju prekidnu rutinu će procesor da izvrši. U odeljku Interconnect modul navedeni su moduli koji čine jedinstveni adresni opseg. Adrese jedinstvenog adresnog opsega su date u kod listingu 1.

```
//Virtuelna platforma:

//bazne adrese
const uint64 VP_ADDR_RAM = 0x00000000;
const uint64 VP_ADDR_BRAM = 0xEE700000;
const uint64 VP_ADDR_IP = 0xEE800000;
const uint64 VP_ADDR_DMA = 0xEE900000;

//adrese RAM memorije
const uint64 VP_ADDR_LAST_GRAPH_LATITUDE_ADDR = VP_ADDR_RAM +
LAST_GRAPH_LATITUDE_ADDR;
const uint64 VP_ADDR_LAST_GRAPH_LONGITUDE_ADDR = VP_ADDR_RAM +
LAST_GRAPH_LONGITUDE_ADDR;
const uint64 VP_ADDR_RAM_H = RAM_ADDR_H;

//adrese pojedinacnih opsega BRAM memorije
const uint64 VP_ADDR_BRAM_HEAD_AND_TAIL = VP_ADDR_BRAM +
HEAD_AND_TAIL_BASE_ADDR;
const uint64 VP_ADDR_BRAM_WEIGHT = VP_ADDR_BRAM + WEIGHT_BASE_ADDR;
const uint64 VP_ADDR_BRAM_COST_AND_VISITED_VERTEX = VP_ADDR_BRAM +
COST_AND_VISITED_VERTEX_BASE_ADDR;

// lokalne bram adrese: od 0-49207
// prva slobodna je 49208(0xC038)
const uint64 VP_ADDR_BRAM_H = 0xEE70C038;

//adresa registra IP modula
const uint64 VP_ADDR_IP_CFG_REG = VP_ADDR_IP + IP_CFG_REG;

//adrese registara DMA modula
const uint64 VP_ADDR_DMA_S2MM_DA = VP_ADDR_DMA + DMA_S2MM_DA;
const uint64 VP_ADDR_DMA_S2MM_LENGTH = VP_ADDR_DMA + DMA_S2MM_LENGTH;
const uint64 VP_ADDR_DMA_S2MM_DMACR = VP_ADDR_DMA + DMA_S2MM_DMACR;
```

Kod listing 1 Jedinstveni adresni opseg

Unutar jedinstvenog adresnog opsega prikazanog kod listingom 1, svaki modul ima lokalne adrese. One su navedene kod listingom 2.

```
//lokalne adrese modula:
//bram
const uint32_t HEAD_AND_TAIL_BASE_ADDR = 0;
const uint32_t WEIGHT_BASE_ADDR = 20043;
const uint32_t COST_AND_VISITED_VERTEX_BASE_ADDR = 40086;

//registar u IP-u const
uint32_t IP_CFG_REG = 0;

//DMA
const uint32_t DMA_S2MM_DMACR = 0x30;
const uint32_t DMA_S2MM_DA = 0x18;
const uint32_t DMA_S2MM_LENGTH = 0x28;
```

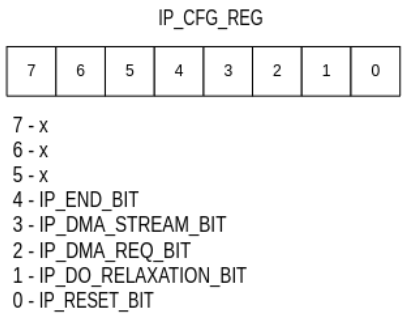
```
//DDR3
const uint64 RAM_BASE_ADDR = 0x00000000;
const uint64 LAST_GRAPH_LATITUDE_ADDR = RAM_BASE_ADDR + (VERTEX_NUM -
1);
const uint64 LAST_GRAPH_LONGITUDE_ADDR = RAM_BASE_ADDR +
(LAST_GRAPH_LATITUDE_ADDR + VERTEX_NUM);
const uint64 RAM_ADDR_H = 0xEE6B281; //1 GB RAM
```

Kod listing 2 Lokalne adrese modula

Registarska mapa

Registar IP modula – IP_CFG_REG

IP_CFG_REG SAHE biće mapiran na 8-bitni registar istog imena. Softver je modelovan tako da je najmanji podatak koji može biti adresiran veličine jednog bajta (eng. Least Adressable Unit - LAU). Raspored pojedinačnih bita i njihovo značenje je dato slikom 10.



Slika 10 IP_CFG_REG

Registar se koristi tako što procesor pročita vrednost registra (prva transakcija). Na osnovu trenutne vrednosti pomoću bit maske procesor podešava odgovarajuće bite registra (druga transakcija) i tako navodi proces relaksacije i povremenog pražnjenja pomoćnog bafera. Softverski API je vrlo jednostavan i čini ga prost upis/čitanje odgovarajuće vrednosti iz registra (ne postoje posebne funkcije za tu svrhu). Sporedni efekti (eng. side effects) nisu prisutni.

Registri DMA modula

Kao i za IP modul, SAHE elementi opisani u odeljku DMA modul predstavljaju registre DMA modula koje softver direktno vidi i sa kojima komunicira u ovom sistemu. Model DMA komponente je pisan u skladu sa Xilinx-ovim AXI DMA kontrolerom [4].

BRAM modul

Procesor BRAM modulu pristupa jedino kada je potrebno podesiti inicijalne podatke o neposećenim temenima uokviru segmenta memorije koji predstavlja tabelu neposećenih temena. Ostali detalji o rasporedu vrednosti unutar lokacija u BRAM-u su dati u odeljku o BRAM modulu.

Iako je sistem SystemC simulacija (još uvek softverski model, bez HLS ili RTL implementacije), na ovaj način je modelovana **koprocesorska i memorijska komunikacija, koja će profiniti dosadašnje odsustvo komunikacije čisto softverskog modela** [2][3].

Literatura

[1] Vranjković, V. : Projektovanje elektronskih uređaja na sistemskom nivou, Novi Sad: Fakultet tehničkih nauka, 2020.

[2] Grant Martin, Brian Bailey, Andrew Piziali: ESL Design and Verification: A Prescription for Electronic System Level Methodology (Systems on Silicon)

[3] Brian Bailey, Grant Martin: ESL Models and their Application: Electronic System Level Design and Verification in Practice

[4] Više o Xilinx AXI DMA kontroleru: link [jun 2025]