

PARALELIZACIJA RAČUNANJA PROSTIRANJA TOPLOTE

1 Opis posla

Potrebno je realizovati višeprocorski sistem za izračunavanje i prikaz prostiranja toplote u 2D prostoru. Sa računara se zadaje se veličina prostora u vidu dimenzija matrice `int16_t` vrednosti, koeficijenti prostiranja toplote po x i y osi i do 5 izvora toplote. Vrednosti toplote u pojedinim tačkama prostora predstavljeni su u Q2.14 formatu. Toplota se predstavlja vrednostima od 0.0 do 1.0, gde 0.0 predstavlja najmanju toplotu (inicijalna vrednost cele matrice izuzev izvora toplote) a 1.0 najveću toplotu u sistemu. Izvori toplote se definišu svojom pozicijom (x i y koordinata) i toplotom koju generišu (vrednost između 0.0 i 1.0).

Koeficijenti prostiranja toplot po x i y osi i izvori toplote zadaju se kroz binarni fajl `sources.bin` koji se generiše pomoću `create_sources.m` skripte.

U SDRAM memoriji se inicijalizuju dve matrice i izračunavanje se naizmenično vrši u tim matricama. Podaci za izračunavanje (trenutna toplota u određenim tačkama) uzimaju se iz jedne matrice, a rezultati izvršavanja se smeštaju u drugu matricu. Izračunavanje se vrši iterativno, prema sledećoj formuli:

$$U_{x,y} = U_{x,y} + C_x (U_{x+1,y} + U_{x-1,y} - 2U_{x,y}) + C_y (U_{x,y+1} + U_{x,y-1} - 2U_{x,y}), \quad (1)$$

gde je $U_{x,y}$ toplota na poziciji (x, y) , C_x koeficijent prostiranja toplote po x osi i C_y koeficijent prostiranja toplote po y osi.

Toplota koju generišu izvori je konstanta i nikada se ne menja. Uzeti da je toplota izvan predviđenog prostora uvek jednaka 0.

Kada se završi izračunavanje odgovarajuća matrica se iz SDRAM memorije prebacuje na računar korišćenjem *host file system* mehanizma. Po uključivanju `hostfs` opcije u okviru BSP, moguće je koristiti standardnu C funkciju `fwrite` za pristup binarnim fajlovima koji se nalaze na host računaru. Voditi računa da ova opcija radi samo u *Debug* modu.

Procesiranje organizovati u okviru beskonačne `while` petlje, gde se na početku procesiranja zahteva unos dimenzija matrice, koeficijenata prostiranja toplote po x i y osi, imena binarnog fajla u kojem su smešteni podaci o koeficijentima i izvorima toplote, imena fajla u kojem se smešta izlazna matrica i broj iteracija za izračunavanje.

Po završetku procesiranja dobija se izlazna matrica koja se šalje na host računar i upisuje u izlazni binarni fajl. Prva 4 bajta izlaznog fajla sadrže širinu matrice, druga 4 bajta visinu matrice, a nakon toga slede elementi matrice u `Int16` formatu (Q2.14 reprezentacija). Učitavanje binarnog fajla izlazne matrice realizovano je skriptom `read_output_heat.m`.

2 Realizacija sistema

Sistem je potrebno organizovati tako da postoji jedan *manager* procesor i 1 do 4 *worker* procesora.

Uloga *manager* procesora je da:

- učitati podatke sa računara o matrici i izvorima toplote, i da u SDRAM memoriji inicijalizuje dve matrice za izračunavanje prostiranja toplote
- organizuje podelu posla po *worker* procesorima za obradu trenutno učitano delu slike u deljenu *On-chip* memoriju. Pored informacije o regionu za obradu potrebno je poslati i podatak iz koje matrice se čitaju podaci, odnosno u koju matricu se upisuju rezultati

- nakon završetka izračunavanja da pošalje rezultat na računar

Uloga *worker* procesora je da:

- po pojavi dostupnog posla od *manager* procesora krene procesiranje regiona matrice koji mu je dodeljen. Uzeti u obzir da su prilikom izračunavanja potrebne vrednosti susednih polja, pa je prilikom obrade određenog regiona potrebno pristupati i tačkama koje se ne nalaze u tom regionu
- nakon završetka obrade da rezultat smesti u predviđeni prostor u SDRAM memoriji

Zbog ograničenih resursa (posebno dostupne količine *On-Chip* memorije) DE0-Nano ploče, svi procesori treba da se izvršavaju iz SDRAM memorije i da koriste keš memorije i za instrukcije i podatke. Voditi računa o veličinama keš memorije, pošto je količina dostupne *On-chip* ograničena.

U linker skripti za sve procesore predvideti prostore u memorijskom prostoru SDRAM memorije gde će se smeštati matrice za obradu.

Takođe, u linker skripti za sve procesore predvideti prostor u deljenoj *On-Chip* memoriji gde će se smeštati opisi poslova i dodatne informacije koje trebaju da budu deljene između *manager* i *worker* procesora.

2.1 Zadatak (60 poena)

2.1.1 Osnovni zahtevi (40 poena)

Cilj projekta je da se nakon realizacije hardvera i softvera sistema prema zadatom opisu potvrdi funkcionalnost sistema za različit broj *worker* procesora (od 1 do 4 *worker* procesora). Demonstrirati funkcionalnost sistema za matricu dimenzija 128×128 , broj izvršavanja 100, koeficijente $C_x = C_y = 0.2$ i sledeće izvore toplote:

- pozicija (5, 5), intenzitet 1.0
- pozicija (50, 10), intenzitet 0.7
- pozicija (55, 64), intenzitet 0.6
- pozicija (64, 64), intenzitet 1.0
- pozicija (100, 120), intenzitet 0.8

Pri evaluaciji rezultata izvršiti poređenje sa procesiranjem u Matlabu pomoću priložene skripte `compare_output.m`.

2.1.2 Performanse sistema (20 poena)

Pored funkcionalnog sistema, potrebno je da dizajn ispuni određene zahteve u pogledu performansi izvršavanja. Od interesa je vreme obrade, tako da je pomoću performance counter-a potrebno meriti vreme od kada *manager* procesor izvrši generisanje poslova pa dok se ne završi sa obradom u poslednjoj iteraciji. Za merenje koristiti matricu dimenzija 64×64 , broj izvršavanja 1000, koeficijente $C_x = C_y = 0.2$ i sledeće izvore toplote:

- pozicija (25, 10), intenzitet 0.7
- pozicija (32, 32), intenzitet 1.0
- pozicija (50, 60), intenzitet 0.8

U tabeli 1 prikazani su zahtevi u pogledu vremena izvršavanja za različit broj *worker* procesora. Da bi se ostvario maksimalan broj poena na projektu, potrebno je da budu ispunjeni zahtevi po pitanju performansi za sve četiri konfiguracije *worker* procesora.

Tabela 1: Tabela performansi

broj <i>worker</i> -a	maksimalno vreme izvršavanja (s)
1	40
2	20
3	14
4	11

2.1.3 Dodatni načini za optimizaciju

Ostavljena vam je sloboda u izboru dodatnih načina za optimizaciju iskorišćenosti resursa FPGA i vremena izvršavanja. Očekuje se da sami odaberete načine na koje ćete da izvršite hijerarhijsku organizaciju sistema (sistem mora biti hijerarhijski organizovan), sinhronizaciju i komunikaciju između *manager*-a i *worker*-a. Što se tiče hijerarhijske organizacije, možete je implementirati na više načina: *worker*-e da stavite u pojedinačne podsisteme, ili sve *worker*-e u isti podsistem, ili SDRAM u podsistem za sebe, ili na neki drugi način, dosta smislenih kombinacija može da se napravi. Takođe, možete probati i sa različitim veličinama keš memorije, različitom organizacijom regiona koji se obrađuju, statičkom ili dinamičkom dodelom taskova, pajplajnovanjem obrade tako da iako nije sve završeno u prethodnoj iteraciji da se započne sa sledećom ako se ne koriste isti delovi matrice, itd.

3 Napomene

1. Svi radite isti projektni zadatak, tako da očekujemo da ćete biti kreativni i da će svaka grupa na različit način rešiti zadati problem, posebno u segmentu dodatnih načina za optimizaciju (2.1.3)
2. Na raspolaganju vam je primer implementacije softvera za jednoprocesorski sistem (`single.c` i `common.h`). U zavisnosti od toga kako budete dizajnirali vaš hardver, možda ćete morati da napravite neke izmene u primeru da bi radio.
3. Pogledati Nios II Multiprocessor Design Example za detalje oko implementacije hijerarhijskog sistema kada se više procesora izvršava iz iste memorije
4. Prilikom kreiranja softverske podrške za *manager*-a, krenuti od *Hello World* aplikacije (ne *Hello World Small*) da bi bila dostupna podrška za *hostfs*. Prilikom kreiranja softverske podrške za *worker*-e, preporuka je da se kao polazna tačka koristi *Hello World Small*, da bi se uštedelo na potrebnoj *On-Chip* memoriji za keš memorije i omogućilo instanciranje većeg broja procesora.

Na kraju projekta je potrebno da kreirati izveštaj koji predstavlja dokumentaciju projekta i opisuje arhitekturu sistema, realizaciju hardvera i softvera, i rezultate koji su dobijeni prilikom merenja vremena izvršavanja. Izveštaj predstavlja dokumentaciju projektovanog sistema i potrebno je da bude dovoljno detaljan kako bi neko uz izveštaj i kod mogao potpuno da reprodukuje rezultate i iskoristi delove projekta u većem sistemu.

Za pun broj poena potrebno je da pored potpune funkcionalnosti sistem bude smisleno particionisan i kod uredno napisan sa dovoljnim brojem komentara, kao i da su kriterijumi performansi zadovoljeni.