Funkcionalna verifikacija IP jezgra za skeletizaciju slika

Studentski rad

Jana Janković
student drugog ciklusa studija
Fakultet tehničkih nauka
Novi Sad, Srbija

Sažetak—**U radu je predstavljena funkcionalna verifikacija IP jezgra za skeletizaciju slika. Kreiran je verifikacioni plan i razvijeno je verifikaciono okruženje bazirano na UVM metodologiji, korišćenjem SystemVerilog jezika. Za provjeru funkcionalnosti modula implementirani su testovi koji koriste slučajno generisanje stimulusa i referentni model. Kako bi se odredila uspješnost verifikacije tokom simulacije prikupljene su informacije o pokrivenosti.**

Ključne riječi - funkcionalna verifikacija, skeletizacija, UVM, SystemVerilog

I. UVOD

Funkcionalna verifikacija predstavlja neophodan korak u razvoju savremenih, složenih digitalnih sistema. To je proces koji ima za cilj da utvrdi ispunjava li implementirani dizajn zahtjeve definisane specifikacijom. Zbog ubrzanog porasta složenosti sistema, koji se sastoje iz ogromnog broja komponenti, korak funkcionalne verifikacije, u kreiranju savremenih čipova, troši sve više vremena i zahtjeva sve više napora dizajnerskih timova. [1] U osnovi, predstavlja veoma važan proces jer je cijena greške, otkrivene prilikom testiranja fabrikovanog uređaja, daleko veća od cijene greške otkrivene tokom verifikacije. Stoga su neophodne adekvatne verifikacione tehnike kako bi se sproveo kvalitetan proces verifikacije. Proces funkcionalne verifikacije sačinjen je od sljedećih koraka, koji zajedno predstavljaju verifikacioni ciklus: [2]

1) **Funkcionalna specifikacija** – predstavlja formalni dokument, koji sastavlja sistemski arhitekta. Sadrži opis funkcionalnosti koje dizajn mora da ispuni, kao i opis interfejsa preko kojih dizajn komunicira sa ostatkom sistema.
2) **Kreiranje verifikacionog plana** – čini krucijalni dio verifikacionog ciklusa koji opisuje šta i na koji način treba da bude verifikovano.
3) **Razvoj verifikacionog okruženja** – Verifikaciono okruženje predstavlja softverski kod - koji kreira stimuluse specifične za dizajn i provjerava ispravnost rezultata dizajna, i alate - koji olakšavaju provjeru ispravnosti dizajna.
4) **Debagovanje dizajna i verifikacionog okruženja** – predstavlja korak u kome se pokreću testovi, definisani u verifikacionom planu, i tokom njihovog izvršavanja pronalaze se i proučavaju razlike, između očekivanih rezultata verifikacionog okruženja i stvarnih rezultata dizajna.
5) **Izvršavanje regresionih testova** – predstavlja ponovno puštanje testova, kako bi se provjerilo da li nova verzija dizajna, nakon ispravljanja otkrivenih grešaka, pravilno funkcioniše i kako bi se pustilo što više različitih scenarija u slučaju okruženja sa randomizovanim elementima.
6) **Debagovanje proizvedenog hardvera** – predstavlja provjeru ispravnosti rada fabrikovanog dizajna. Postoji mogućnost pronalaženja grešaka čiji je uzrok sada teže pronaći, zbog nemogućnosti praćenja unutrašnjih signala fabrikovanog dizajna.
7) **Izvršavanje eskejp analize**(eng. *escape analysis*) – Ako su, prilikom testiranja fabrikovanog hardvera, uočene greške, vrše se eskejp analize, kako bi se otkrilo zbog čega greške nisu detektovane od strane verifikacionog okruženja. Na taj način, može se spriječiti ponavljanje sličnih grešaka u budućnosti.

Rad predstavlja opis funkcionalne verifikacije IP jezgra koje je sastavni dio sistema za skeletizaciju slika[3]. Implementacija sistema zasnovana je na jednom od najpoznatijih algoritama za skeletizaciju, Zhang-Suen algoritmu. Procesor učitava sliku u boji i pretvara je u binarnu sliku. Dobijena binarna slika se smješta u memoriju. Kada završi sa pripremom slike za obradu, procesor kofiguriše DMA kontroler i IP jezgro, upisivanjem odgovarajućih vrijednosti u njihove registre. DMA kontroler obavlja prenos podataka između memorije i IP jezgra. Kada se konfiguriše, DMA kontroler kreće da šalje IP jezgru podatke sa adrese na kojoj je smještena slika za obradu. Nakon što pošalje zadati broj podataka, čeka da mu jezgro pošalje podatke, koje je sada potrebno upisati u memoriju. IP jezgro predstavlja hardverski blok koji izvršava proces skeletizacije.[3]

II. KREIRANJE VERIFIKACIONOG PLANA

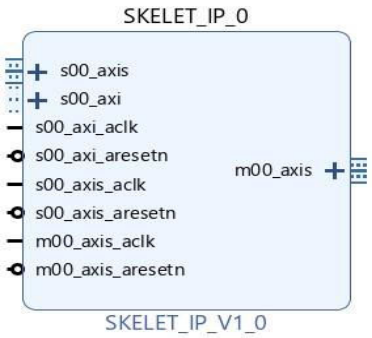
A. Opis rada dizajna

U ovom projektu bilo je potrebno verifikovati funkcionalnost IP jezgra za skeletizaciju slike[4]. IP jezgro ima parametrizovan broj workera, funkcionalnih jedinica koji vrše

obradu piksela, (NUM_WORKERS) i dimenzije BRAM-a (WIDTH, DEPTH). Slike koje jezgro obrađuje su binarne slike maksimalne rezolucije WIDTHxDEPTH piksela. Ukoliko se obrađuje slika manjih dimenzija, sliku je potrebno dopuniti nulama do širine WIDTH.

Interfejs IP jezgra (Sl. 1) je sačinjen od AXI-*Lite* interfejsa (s00_axi), AXI-*Stream master* (m00_axis) i *slave* (s00_axis) interfejsa. Preko AXI-*Lite* interfejsa se može pristupiti untrašnjim registrima jezgra. U registar *height_i,* na adresi 0x04 upisuje se vrijednost visine slike umanjena za jedan. U registar *num_px,* na adresi 0x08 upisuje se pozicija workera koji će obrađivati granični piksel. U registar *last_px,* na adresi 0x0C potrebno je upisati broj piksela koji worker, koji ne obrađuje granični piksel, treba da obradi. Za početak obrade potrebno je u *start* registar na adresi 0x00 upisati vrijdnost jedan. Provjera aktivnosti jezgra obavlja se čitanjem stanja *ready* registra na adresi 0x10.

Preko AXI-*Stream slave* interfejsa jezgro prima sliku koju treba da obradi. Podaci koji se šalju su 64-bitni i predstavljaju 64 piksela slike. Slika se šalje po redovima i kod prvog podatka koji se šalje MSB je gornji, lijevi piksel slike. Kada završi sa obradom, jezgro na isti način šalje sliku preko AXI-*Stream master* interfejsa.



Slika 1. Interfejs IP jezgra za skeletizaciju

B. *Opis nivoa verifikacije*

Osnovna funkcionalna jedinica dizajna je worker. Unutar workera je realizovana osnovna funkcionalnost, odnosno obrada jednog piksela. Worker ima jasno definisan interfejs i specifikaciju, i zbog toga ga je moguće verifikovati odvojeno na posebnom nivou hijerarhije.

Verifikaciona hijerarhija u ovom projektu će imati dva nivoa. Na prvom verifikacionom nivou će se verifikovati worker, dok će se na drugom verifikacionom nivou verifikovati čitav dizajn.

Na prvom nivou potrebno je provjeriti osnovnu funkcionalnost workera. Worker ima 10 jednobitnih ulaza i jedan ulaz čija se širina mijenja u zavisnosti od broja workera. Ukoliko je vrijednost tog ulaza nula, piksel se ne obrađuje, u suprotnom piksel se obrađuje. Na ovom nivou verifikacije verifikovaće se worker kod kog je taj ulaz jednobitan. Workeri u sebi ne sadrže registre, što znači da se u jednom simulacionom ciklusu može provjeriti jedna kombinacija ulaza. Koristiće se deterministički test koji će na ulazu workera generisati sve moguće kombinacije ulaza.

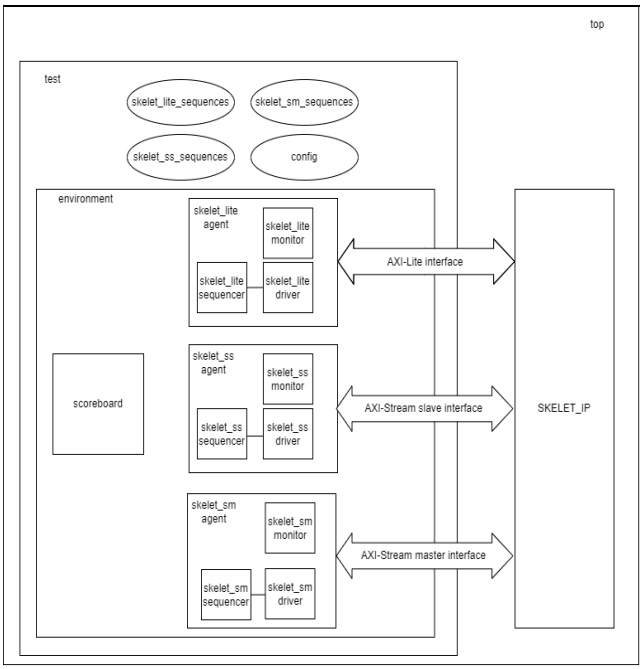
Na drugom verifikacionom nivou koristiće se testovi sa slučajnim generisanjem stimulusa koji će provjeravati sljedeće funkcionalnosti:

1. Ispravno reagovanje na reset signal.
2. Poštovanje AXI-*Lite* i AXI-*Stream* protokola.
3. Željena funkcionalnost u slučaju da se na ulaz dovode slike koje su iste širine kao i širina BRAM-a.
4. Željena funkcionalnost u slučaju da se na ulaz dovode slike koje su iste širine kao i širina BRAM-a i kod kojih su svi granični pikseli crni.
5. Željena funkcionalnost u slučaju da se na ulaz dovode slike koje su širine manje od širine BRAM-a.
6. Željena funkcionalnost u slučaju da se na ulaz dovode slike koje su širine manje od širine BRAM-a i kod kojih su svi granični pikseli crni.
7. Željena funkcionalnost u slučaju da se na ulaz dovode naizmjenično slike kod kojih su širine iste i manje od širine BRAM-a.

Dizajn može imati 16 mogućih širina BRAM-a. Za svaku širinu BRAM-a postoje tri arhitekture. Koja od arhitektura će se koristiti zavisi od broja workera koji se izabere za implementaciju dizajna. Potrebno je na isti način verifikovati sve tri arhitekture za različite širine BRAM-ova.

III. RAZVOJ VERIFIKACIONOG OKRUŽENJA

A. *Verifikaciono okruženje IP jezgra*



Slika 2. Šematski prikaz verifikacionog okruženja IP jezgra

Verifikaciono okruženje u ovom projektu je zasnovano na UVM (eng. *Universal Verification Methodology*) [5]. UVM je standardizovana metodologija za funkcionalnu verifikaciju koja koristi bibilioteku u SystemVerilog jeziku za kreiranje komponenti i TLM standard za komunikaciju komponenti. Blok dijagram kreiranog verifikacionog okruženja prikazan je na Sl. 2. Sastoji se iz sljedećih verifikacionih komponenti: testa, okruženja, sekvenci, sekvencera, paketa, monitora, drajvera, agenata, *scoreboard*-a, referentnog modela i konfiguracije.

1) *Sequence item, sekvenca i sekvencer*

Jedan od prvih koraka u kreiranju verifikacionog okruženja je kreiranje objekta *sequence item* koji predstavlja transakciju koja se koristi kao stimulus. Na osnovu interfejsa modula kreirana su tri *sequence item-a: skelet_lite_seq_item, skelet_ss_seq_item, skelet_sm_seq_item*. *Skelet_lite_seq_item* služi da prosleđuje visinu slike, broj piksela koji se obrađuju, broj workera koji obrađuje posljednji piksel, *start* i *ready* signale, dok *skelet_ss_seq_item* i *skelet_sm_seq_item* prosljeđuju vrijednosti podataka koji se šalju i dobijaju od modula preko AXI-*Stream* interfejsa.

Sekvenca sadrži logiku generisanja stimulusa [5]. U okviru ovog projekta postoji nekoliko različitih sekvenci pomoću kojih se testiraju sve funkcionalnosti definisane verifikacionim planom. Za AXI-*Lite* interfejs u jednoj sekvenci se prvo vrši čitanje vrijednosti registra sa nasumično izabrane adrese, zatim se čita vrijednost *ready* registra sve dok ona ne bude jedan, i potom se generišu vrijednosti visine slike, broja piksela koji se obrađuju, broj workera koji obrađuje posljednji piksel i start. Ostale sekvence za AXI-*Lite* interfejs, generišu nasumično odabrane vrijednosti za više slika, u nekim slučajevima za slike iste širine kao BRAM, u nekim za slike manje širine. Za AXI-*Stream master* interfejs sekvenca generiše vrijednost *ready* signala. Za AXI-*Stram slave* interfejs sekvence generišu proizvoljne vrijednosti piksela; za slike manje i iste širine kao BRAM i za slike sa graničnim crnim pikselima. Generišu, takođe, i poslije koliko taktova će *valid* signal imati vrijednost 1.

Sekvencer koristi sekvence kako bi poslao podatke drajveru, ali i primio odgovor od drajvera. U ovom verifikacionom okruženju postoje tri sekvencera, po jedan za svaki od interfejsa.

2) *Drajver*

Drajver je direktno povezan na interfejs od dizajna i njegova uloga je da emulira signale koji će biti poslati uređaju. *Skelet_lite* drajver vodi računa da AXI-*Lite* protokol bude ispoštovan, emulira upisivanje i čitanje podataka iz registara memorijskog podsistema jezgra. *Skelet_ss* drajver, poštujući AXI-*Stream* protokol, emulira slanje slike jezgru. Nakon pojave *ready* signala čeka određen broj taktova i zatim podiže *valid* signal i postavlja odgovarajući podatak na *data* magistralu. *Skelet_sm* drajver emulira slanje *ready* signala jezgru kako bi jezgro moglo da pošalje obrađenu sliku.

3) *Monitor*

Uloga monitor komponente je da nadgleda interfejs. Monitor *skelet_lite* nadgleda AXI-*Lite* interfejs. Kada se desi upis u registre memorijskog podsistema, šalje upisani podatak *scoreboard* komponenti, preko TLM interfejsa. *Skelet_ss* i *skelet_sm* nadgledaju AXI-*Stream slave* AXI-*Stream* master interfejse i kada su *valid* i *ready* signali na jedinici, šalju podatak sa *data* magistrale *scoreboard* komponenti.

4) *Agent i konfiguracija*

Agent je klasa koja enkapsulira drajver, monitor i sekvencer kako bi olakšala ponovno korišćenje datih komponenti. Konfiguracioni objekat sadrži sve podatke neophodne da bi se konfigurisao testbenč, kao što je informacija o tome koji će agenti biti pasivni (instancira se samo monitor) ili aktivni (sve tri komponente se instanciraju). U ovom projektu su realizovana tri aktivna agenta: *skelet_lite*, *skelet_sm* i *skelet_ss*.

5) *Scoreboard*

Glavna funkcionalnost *scoreboard* komponente je da provjeri funkcionalnost dizajna [5]. Unutar nje implementiran je referentni model koji predstavlja dizajn koji se verifikuje, napisan na visokom nivou apstrakcije. *Scoreboard* dobija podatke od *skelet_lite* i *skelet_ss* monitora na osnovu kojih se u referentnom modelu vrše proračuni. Podaci koji se dobijaju od *skelet_ss* monitora smještaju se u niz koji predstavlja ulaznu sliku. Referentni model realizovan je kao funkcija koje se poziva kada se od *skelet_ss* dobije informacija da je modul preko AXI-*Stream* interfejsa primio posljednji podatak. Referentni model obrađuje ulaznu sliku i obrađenu sliku smješta u niz. Preko *skelet_sm* monitora *scoreboard* dobija podatke koje jezgro šalje kao rezultat obrade i smješta ih u niz. Kada *skelet_sm* monitor pošalje informaciju da je modul poslao posljednji podatak vrši se poređenje. Porede se slika obrađena od strane referentnog modela i slika obrađena od strane modula; ako postoji razlika prijavljuje se greška putem *uvm_error* metode.

6) Okruženje i testovi

Klasa *environment* instancira i enkapsulira sve prethodno opisane komponente, dok test objekat instancira i enkapsulira *environment* komponentu, konfiguriše verifikaciono okruženje i pokreće sekvence na određenom sekvenceru i na taj način primjenjuje stimulus na dizajn [5]. U osnovnom testu *skelet_test_base* instancira se verifikaciono okruženje i vrši se konfiguracija. Ovaj osnovni test nasleđuje šest testova koji pokreću različite sekvence. *Skelet_test_simple* služi da se pokrenu sekvence pomoću kojih će se poslati jedna slika širine BRAM-a modulu. Namjenjen je za osnovnu provjeru funkcionalnosti i provjeru poštovanja protokola. Unutar *skelet_test_simple_2* šalje se više različitih slika širine BRAM-a, dok se u testu *skelet_test_simple_3* šalje više različitih slika širine manje od BRAM-a. U testovima *skelet_test_simple_4* i *skelet_test_simple_5* modulu se šalju slike sa graničnim crnim pikselima. Test *skelet_test_simple_6* služi za provjeru ponašanja sistema kada se modulu naizmjenično šalju slike širine iste i manje od širine BRAM-a.

Kako bi se verifikovalo parametrizovano IP jezgro testbenč mora posjedovati informacije o vrijednostima tih parametara. S obzirom da je UVM hijerarhijska kolekcija objekata, vrijednost parametara se može proslijediti i dizajnu i verifikacionom okruženju. U top modulu se moraju definisati parametri, instancirati dizajn i proslijediti mu parametre, i započeti test kom su takođe prosleđeni parametri. Testovi u tom slučaju takođe moraju biti parametrizovani. U slučaju da se za *factory* registraciju koristi makro *uvm_component_param_utils()* kod neće raditi jer se on registruje samo sa *type-based* UVM *factory*. Međutim, *task run_test()* zahtjeva da test bude registrovan pomoću *string-based* UVM *factory.* Zbog toga je *uvm_component_param_utils()* makro potrebno proširiti.

B. Verifikaciono okruženje workera

Verifikaciono okruženje workera slično je verifikacionom okruženju modula za skeletizaciju. Sastoji se iz testa, okruženja, sekvence, sekvencera, paketa, monitora, drajvera, agenta, *scoreboard*-a, referentnog modela i konfiguracije. Sekvenca redom šalje sve moguće kombinacije piksela koje drajver postavlja na ulaze workera. Monitor nadgleda ulaze i izlaze workera i šalje njihove vrijednosti *scoreboard* komponenti. Unutar *scoreboard* komponente se u referentnom modelu računaju očekivane vrijednosti, koje se zatim porede sa izlazima workera.

IV. PRIKUPLJANJE POKRIVENOSTI I REGRESIJA

Povećanjem složenosti dizajna dolazi do pojave ogromnog broja potencijalnih stanja dizajna, koje, funkcionalna verifikacija obično nije u mogućnosti sve detaljno da provjeri, zbog čega je teško ustanoviti kada je proces verifikacije završen. Zbog toga se uvodi nova mjera - funkcionalna pokrivenost, koja definiše procenat verifikacionih ciljeva, definisanih verifikacionim planom u odnosu na funkcionalne zahtjeve dizajna, koji su ispunjeni. Kod testova sa slučajnim generisanjem stimulusa, gdje ne znamo koji slučaj se tačno verifikuje, praćenje pokrivenosti pruža informaciju o tome da li su se desili slučajevi od interesa i time uklanja potrebu za analizom samih signala.

Grupe za praćenje pokrivenosti su definisane i kreirane u monitorima i *scoreboard* komponeni. U *skelet_lite* monitoru skupljaju se podaci o tome da li su se čitale i upisivale vrijednosti u registre memorijskog podsistema, pri čemu je izostavljena provjera upisa u registar *ready*. Takođe se skuplja pokrivenost vrijednosti *ready* signala. U *skelet_ss* i *skelet_sm* monitorima skupljaju se podaci o tome da li su se desile sve moguće kombinacije *ready* i *valid* signala. U *scoreboard* komponenti skupljaju se pokrivenost širine BRAM-a, da li su se desile širine slike iste i manje od širine BRAM-a, kao i njihov *cross coverage*.

Prilikom obavljanja testova potrebno je voditi računa o pokrivenosti. Kako bi se verifikovale moguće arhitekture modula i postigla maksimalna pokrivenost potrebno je pustiti različite testove. Za tu svrhu napravljena je tcl skripta koja će prolaziti kroz različite širine BRAM-a i randomizovan broj workera i puštati sve testove. U tabeli 1. prikazana je funkcionalna pokrivenost nakon obavljenih testova.

TABELA I. FUNKCIONALNA POKRIVENOST

Naziv grupe	**Pokrivenost[%]**
skelet_test_pkg::skelet_scoreboard::cg_width	100
skelet_lite_agent_pkg::skelet_lite_monitor::cg_addr	100
skelet_lite_agent_pkg::skelet_lite_monitor::cg_ready	100
skelet_ss_agent_pkg::skelet_ss_monitor::cg_srv	100
skelet_sm_agent_pkg::skelet_sm_monitor::cg_mrv	100

V. ZAKLJUČAK

Cilj ovog rada bio je da predstavi proces funkcionalne verifikacije modula za skeletizaciju slike. Postupak verifikacije započet je kreiranjem verifikacionog plana opisanog u drugom poglavlju. Zatim je implementirano verifikaciono okruženje i pušteni su testovi koji provjeravaju funkcionalnost sistema. Kreirane su i grupe za praćenje pokrivenosti da bi se dobila informacija o uspješnosti procesa verifikacije.

ZAHVALNICA

Rad je dio teorijskog istraživanja i praktične realizacije u sklopu izrade diplomskog rada na Fakultetu tehničkih nauka Univerziteta u Novom Sadu, pod mentorstvom dr Vuka Vranjkovića.

LITERATURA

[1] Laung-Terng Wang, Yao-Wen Chang, Kwang-Ting (Tim) Cheng „Electronic Design Automation: Synthesis, Verification, and Test”, Morgan Kaufmann, 2009.

[2] Bruce Wile, John C. Goss, Wolfgang Roesner, „Comprehensive Functional Verification”, Morgan Kaufmann, 2005.

[3] J. Janković, "Projektovanje sistema za skeletizaciju slika zasnovanog na Zhang-Suen algoritmu," *2022 30th Telecommunications Forum (TELFOR)*, Belgrade, Serbia, 2022, pp. 1-4, doi: 10.1109/TELFOR56187.2022.9983743.

[4] Jana Janković, Implementation of the hardware module for image skeletonization system, International Scientific Conference UNITECH 2022.Gabrovo, prihvaćen za objavljivanje

[5] Initiative, A. S. „Universal Verification Methodology (UVM) 1.2 User's Guide" Accellera Systems Initiative: Elk Grove, CA, USA, 2015.

ABSTRACT

Functional verification of the IP core for image skeletonization module is presented in this paper. A verification plan was created and a verification environment based on the UVM methodology was developed, using the SystemVerilog language. To check the functionality of the module, tests using random stimulus generation and a reference model were implemented. In order to determine the success of the verification, during the simulation, coverage was collected.

FUNCTIONAL VERIFICATION OF THE IP CORE FOR IMAGE SKELETONIZATION

Jana Janković