Projekat iz predmeta Funkcionalna Verifikacija

Tema : Ekstrahovanje karakterističnih tačaka na fotografiji pomoću SURF algoritma

Mentor : Ivan Čejić

Studenti :

Dejana Ristić EE56/2019

Nemanja Žarković EE69/2019

Aleksandar Vig EE142/2019

SADRŽAJ

Contents

1. Uvod

Ovaj dokument predstavlja specifikaciju i način na koji je realizovana verifikacija sistema koji izvršava SURF metodu i time ekstrahuje karakteristične tačke na ulaznoj slici. Detalji oko projektovanja sistema dati su dokumentaciji predmeta „Projektovanje složenih digitalnih sistema“ dok će u ovom uvodu biti prezentovana samo osnovna ideja.

SURF metoda (Speeded Up Robust Features) je brz i robustan algoritam za lokalno, invarijantno predstavljanje sličnosti i poređenje slika.

SURF algoritam je sam po sebi zasnovan na dva uzastopna koraka (detekcija karakteristika i opis). Slično mnogim drugim pristupima, kao što je SIFT metoda, detekcija karakteristika u SURF-u se oslanja na skalno-prostornu reprezentaciju, kombinovanu sa diferencijalnim operatorima prvog i drugog reda. Originalnost SURF algoritma je u tome što se ove operacije ubrzavaju upotrebom tehnika prozor (box) filtera.
Pristup za detekciju interesnih tačaka koristi veoma osnovnu aproksimaciju Hesijanove matrice. SURF koristi Hesijanovu matricu zbog njenih dobrih performansi u vremenu računanja i tačnosti. Umesto da koristi drugačiju meru za izbor lokacije i razmere (Hesijan-Laplasov detektor), SURF se oslanja na determinantu Hesijanove matrice za oba. Ako imamo piksel, njegov Hesijan je nešto poput:

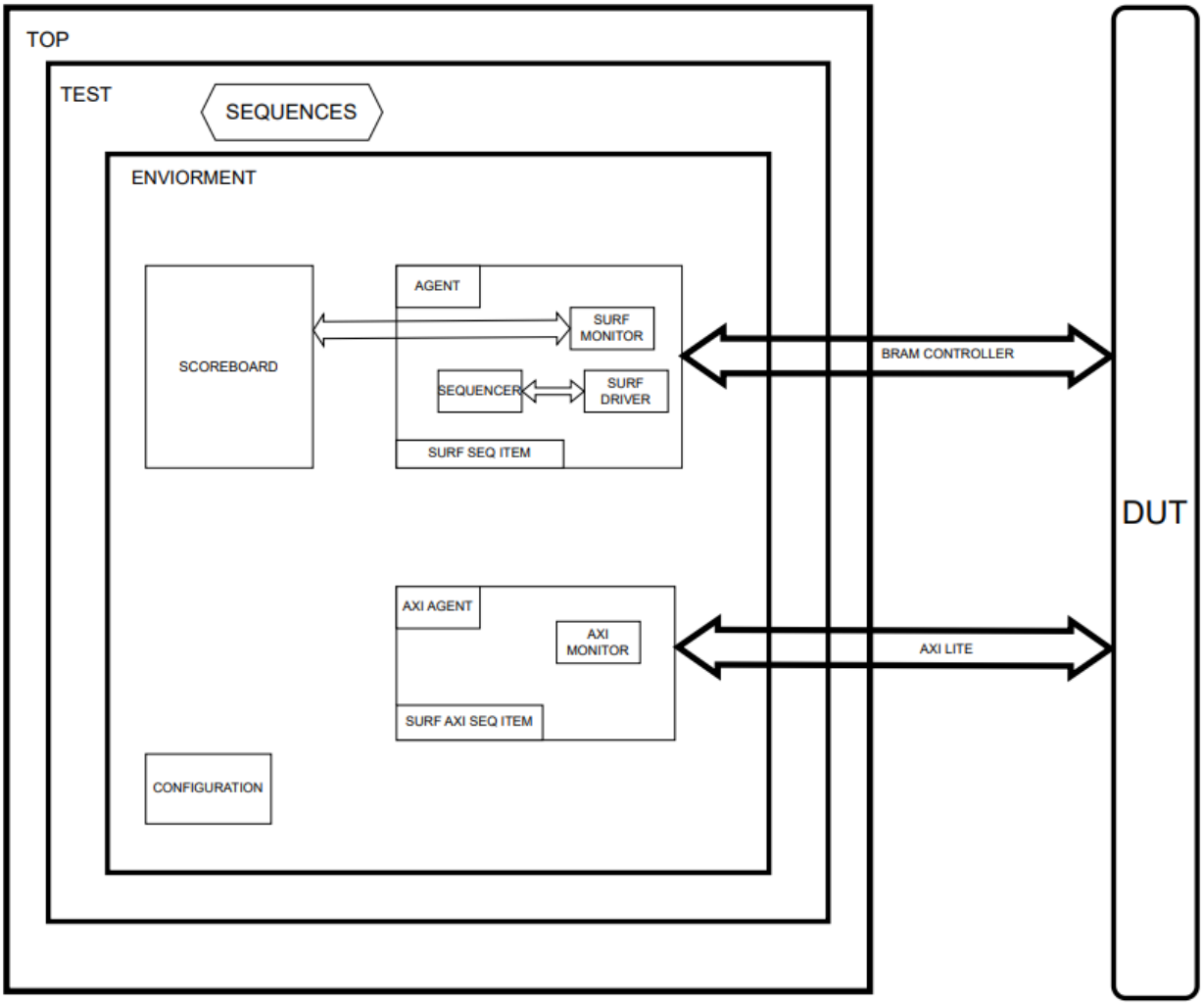
$$H(f(x,y)) = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix}$$

Za prilagođavanje bilo kojoj skali, filtrira se slika pomoću Gausovog kernela, tako da je data tačka X = (x,y), Hesijanova matrica X(x, σ) u x na skali σ je definisana kao:

$$\mathcal{H}(x,\sigma) = \begin{bmatrix} L_{xx}(x,\sigma) & L_{xy}(x,\sigma) \\ L_{xy}(x,\sigma) & L_{yy}(x,\sigma) \end{bmatrix}$$

gde je Lxx(x, σ) konvolucija Gausovog izvoda drugog reda sa slikom I u tački x, i slično za Lxy(x, σ) i Lyy(x, σ).
Kreiranje SURF deskriptora odvija se u dva koraka. Prvi korak se sastoji od fiksiranja ponovljive orijentacije na osnovu informacija iz kružnog regiona oko ključne tačke. Zatim konstruišemo kvadratni region poravnat sa izabranom orijentacijom i iz njega izdvajamo deskriptor SURF.
Cilj deskriptora je da obezbedi jedinstven i robustan opis karakteristike slike, na primer, opisivanjem distribucije intenziteta piksela u okolini tačke interesovanja. Većina deskriptora se stoga izračunava na lokalni način, pa se dobija opis za svaku tačku od interesa koja je prethodno identifikovana.

2. Verifikaciono okruženje



Slika 1 : Verifikaciono okruženje

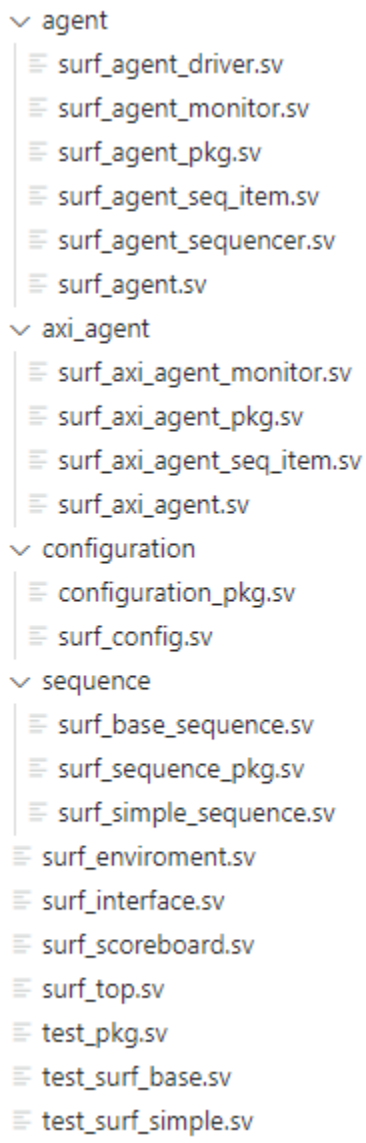
Verifikaciono okruženje se sastoji se od sledećih komponenti :

- Aktivni agent (Agent)
- Pasivni agent (AXI Agent)
- Scoreboard za proveru rezultata
- Konfiguracionog paketa

- Sequence paketa
- Test paketa
- Enviorment-a
- Top fajl

Pored navedenih komponenti, u dizajnu se nalazi i DUT (Design Under Test) koji se verifikuje. Glavni protokol korišćen za pokretanje sistema i unos parametara slike je AXI Lite, dok se BRAM memorije za sliku i izlazni INDEX niz pune putem AXI BRAM kontrolera.

Detaljan opis komponenti i fajlova će biti dat u narednim poglavljima, a hijerarhija fajlova izgleda ovako :



Slika 2 : Hijerarhija okruženja

3. Komponente verifikacionog okruženja

U ovom poglavlju će biti dat detaljniji uvid u komponente samog okruženja.

3.1. Top

U top fajlu su uključeni UVM paketi, UVM makroi i naš test paket. Instanciran je interfejs, kao i DUT, čiji su signali povezani sa signalima iz tog interfejsa. U okviru initial blokova, interfejs je prosleđen enviorment-u putem uvm_config_db, pokrenut je test, izvršen inicijalni reset sistema i generisanje clk signala.

3.2. Enviorment

Deklaracija i dodela virtuelnog interfejsa agentima, konfiguraciji i scoreboard-u, kao i samo kreiranje tih komponenti. Takođe, ovde vršimo povezivanje monitor komponente sa scoreboard komponentom

3.3. Interface

U interface fajlu navedeni su AXI parametri za širinu podataka i adresa (24 i 17), a takođe su deklarisani signali za AXI Lite protokol i BRAM memorije.

```
// Memory - Output
logic [7:0] ip_addrc;
logic [23:0] ip_doutc;
logic ip_enc;
logic [23:0] img_doutc;

logic [7:0] ip_addrd;
logic [23:0] ip_doutd;
logic ip_end;
logic [23:0] img_doutd;
```

Slika 3 : BRAM signali za INDEX niz

```
// AXI Lite - Main registers
 logic [C_S00_AXI_ADDR_WIDTH -1:0] s00_axi_awaddr;
 logic [2:0] s00_axi_awprot;
 logic s00_axi_awvalid;
 logic s00_axi_awready;
 logic [C_S00_AXI_DATA_WIDTH -1:0] s00_axi_wdata;
 logic [(C_S00_AXI_DATA_WIDTH/8) -1:0] s00_axi_wstrb;
 logic s00_axi_wvalid;
 logic s00_axi_wready;
 logic [1:0] s00_axi_bresp;
 logic s00_axi_bvalid;
 logic s00_axi_bready;
 logic [C_S00_AXI_ADDR_WIDTH -1:0] s00_axi_araddr;
 logic [2:0] s00_axi_arprot;
 logic s00_axi_arvalid;
 logic s00_axi_arready;
 logic [C_S00_AXI_DATA_WIDTH - 1:0] s00_axi_rdata;
 logic [1:0] s00_axi_rresp;
 logic s00_axi_rvalid;
 logic s00_axi_rready;
```

Slika 4 : AXI Lite signali

3.4. Test Paket

Pored UVM paketa i makroa, uključuju se i svi naši ostali paketi (agent, AXI agent, sequence, configuration), kao i ostale komponente (scoreboard, environment, test base i test simple) koje nisu deo nekog paketa, uključujući i interface.

3.4.1 Base Test

U test base-u se kreiraju environment i konfiguracija, unutar koje se randomizuje slika koja će biti obrađena. Takođe se poziva funkcija koja će izvući podatke iz tekstualnih fajlova.

3.4.2 Test Simple

Test simple nasleđuje test base i pokreće simple sequence ka agentu i sekvenceru.

3.5. Sequence

U ovom folderu se nalaze sequence paket, base sequence i simple sequence.

3.5.1. Base Sequence

U ovom fajlu su se iz tekstualnih fajlova učitavale parametri za sliku dok to nije premešteno u konfiguracioni fajl.

3.5.2. Simple Sequence

Ovaj fajl je veoma važan jer omogućava prosleđivanje ispravne sekvence za pokretanje i rad celog sistema. U njemu su definisani početna adresa AXI protokola i ofseti pomoću kojih ciljamo potrebne registre (start, iradius, fracr, fracc, spacing, iy, ix, step, i_cose, i_sine, scale, ready). Nakon uspešne inicijalizacije sistema, preko tih ofseta se unose parametri za sliku, dok se BRAM memorija za sliku puni slikom (vrednostima piksela). Na kraju, kada je sve spremno, signal start se podiže na 1, a sistem čeka podizanje signala ready na 1, što označava kraj obrade. U tom trenutku, poslednjom for petljom prolazimo kroz oba index niza, a pristup njihovim vrednostima imamo preko ip_doutc i ip_doutd, koje ćemo kasnije koristiti u scoreboard komponenti.

3.6. Agent (aktivni)

U aktivnom agentu uključujemo naš driver, monitor za nadgledanje izlaza sistema, sekvencer i sequence item. Zatim povezujemo sequence item sa driver-om.

3.6.1. Sequence Item

Sequence item sadrži sve signale interfejsa, kao i dodatni signal bram_axi, koji će driver-u dati informaciju o tome da li radi sa BRAM memorijama ili AXI protokolom.

3.6.2. Driver

Kao što je već rečeno, režim rada driver-a se određuje na osnovu signala bram_axi. Ako je vrednost signala 0, driver će vrednosti iz prosleđenog sequence item-a proslediti ka BRAM memorijama, dok je u suprotnom aktiviran AXI režim. U AXI režimu (bram_axi = 1), napisan je kod za transakciju upisa putem Lite protokola i transakciju čitanja kada je to potrebno (čekanje ready signala).

3.6.3. Monitor

Ovaj monitor je zadužen da prati i nadgleda signale na izlazu sistema i, zajedno sa scoreboard komponentom, metodom zlatnih vektora, utvrdi validnost rada dizajna. U main fazi, putem wait naredbi, omogućeno je da monitor počne sa nadgledanjem od trenutka kada započne isčitavanje vrednosti. Hvatamo adrese i vrednosti na njima, a pomoću funkcije write komuniciramo sa scoreboard-om. U scoreboard-u se provera vrši na kraju izvršavanja, gde se validiraju rezultati na osnovu zlatnih vektora. Takođe je implementiran deo za pokrivenost (coverage), ali to će biti objašnjeno u posebnom poglavlju.

3.7. AXI Agent

Ovo je pasivni agent koji sadrži samo monitor, koji nadgleda AXI Lite transakcije na ulazu sistema. Ukoliko su određeni signali (axi_awvalid i axi_awready) aktivni, prikupljamo informacije o toj transakciji upisa u curr_it, tj. sequence item koji se kreira u monitoru. Analogno tome, ako su signali axi_rvalid i axi_arready aktivni, prikupljamo informacije o transakciji čitanja. Takođe, u ovom monitoru je implementiran coverage.

3.8. Scoreboard

Ovaj surf_scoreboard implementira metod zlatnih vektora za proveru tačnosti rezultata. Kroz prikupljanje podataka tokom izvršavanja sistema, scoreboard skladišti dolazne vrednosti u privremene promenljive, ali ne upisuje odmah u "observed" nizove. Tek u report_phase, na kraju procesa, ove prikupljene vrednosti se porede sa očekivanim vrednostima iz zlatnih vektora, čime se omogućava provera tačnosti sistema.

Ako se neslaganja detektuju, koristi se uvm_error za prijavu grešaka u TCL konzoli. Ovaj metod omogućava postepeno prikupljanje podataka tokom izvršavanja, a provere i izveštavanje o eventualnim greškama obavljaju se na kraju. Takođe, postoji i mehanizam za ignorisanje razlika u poslednjem bitu kada je to prikladno, kako bi se izbegla nepotrebna prijava grešaka zbog trivijalnih razlika.

3.9. Konfiguracija

U konfiguracionoj klasi surf_config obavljaju se različiti zadaci neophodni za postavljanje i pokretanje testova. Ključne funkcionalnosti su sledeće:

1. **Određivanje aktivnih/pasivnih agenata**:
   Pomoću makroa uvm_active_passive_enum, agenti se definišu kao pasivni ili aktivni, u zavisnosti od potrebe testa.
2. **Relativne putanje za fajlove**:
   Definišu se relativne putanje za svaki tekstualni fajl, omogućavajući da test može da se pokrene na bilo kom računaru bez potrebe za izmenama putanja.

3. **Randomizacija izbora fajla**:
   Randomizacija se koristi za biranje fajla koji će biti otvoren, a na osnovu toga se učitavaju ulazni parametri potrebni za testiranje.
4. **Učitavanje podataka iz fajlova**:
   Implementirane su funkcije za učitavanje različitih podataka iz fajlova, kao što su:
   - Parametri ulaza (fracr_upper, fracc_lower, spacing_upper, itd.),
   - Zlatni vektori za index_upper_gv i index_lower_gv,
   - Podaci za index_upper i index_lower.

4. Coverage analiza

Coverage (pokrivenost) prikupljamo na nekoliko ključnih mesta, a najvažniji su monitori. Na slici 5 prikazan je coverage za izlazni monitor, gde prikupljamo informacije o opsegu adresa koje se čitaju. Na slici 6 prikazan je coverage za AXI monitor, gde je zadat tačan cilj (goal) jer znamo da je potrebno da se desi 15 transakcija za upis i 4 za čitanje kako bi sistem pravilno funkcionisao. Takođe, imamo coverpoint za vrednosti registara poput start (cmd) i ready (status), koji kontrolišu rad samog sistema.

```
covergroup surf_cover (int coverage_goal);
    option.per_instance = 1;
    option.goal = 8;
// Pokrivanje adresa u koracima od 4 za imgUPPERaddress
imgUPPERaddress : coverpoint s_vif.ip_addrd {
    bins b1 = {[0:28]};
    bins b2 = {[32:60]};
    bins b3 = {[64:92]};
    bins b4 = {[96:124]};
    bins b5 = {[128:156]};
    bins b6 = {[160:188]};
    bins b7 = {[192:220]};
    bins b8 = {[224:252]};
}

// Pokrivanje adresa u koracima od 4 za imgLOWERaddress
imgLOWERaddress : coverpoint s_vif.ip_addrc {
    bins b1 = {[0:28]};
    bins b2 = {[32:60]};
    bins b3 = {[64:92]};
    bins b4 = {[96:124]};
    bins b5 = {[128:156]};
    bins b6 = {[160:188]};
    bins b7 = {[192:220]};
    bins b8 = {[224:252]};
}
```

Slika 5 : BRAM adrese za INDEX niz

Details of Instance Cover Point - obj.surf_cover :: imgUPPERaddress

Summary of Cover Point - imgUPPERaddress

Category	Expected	Uncovered	Covered	Percent
User Defined Bins	8	0	8	100

User Generated Bins for Instance Cover Point - obj.surf_cover :: imgUPPERaddress

Covered bins

Name	Hit Count	At least
b1	1072	1
b2	656	1
b3	536	1
b4	356	1
b5	532	1
b6	356	1
b7	512	1
b8	184	1

Details of Instance Cover Point - obj.surf_cover :: imgLOWERaddress

Summary of Cover Point - imgLOWERaddress

Category	Expected	Uncovered	Covered	Percent
User Defined Bins	8	0	8	100

User Generated Bins for Instance Cover Point - obj.surf_cover :: imgLOWERaddress

Covered bins

Name	Hit Count	At least
b1	1072	1
b2	656	1
b3	536	1
b4	356	1
b5	532	1
b6	356	1
b7	512	1
b8	184	1

```
covergroup axi_write_transactions;
    option.per_instance = 1;
    option.goal = 15;
    write_address : coverpoint s_vif.s00_axi_awaddr{
        bins BASE_ADDRESS = {AXI_BASE};
        bins FRACR_UPPER_REG_INPUT = {AXI_BASE + FRACR_UPPER_REG_OFFSET};
        bins FRACR_LOWER_REG_INPUT = {AXI_BASE + FRACR_LOWER_REG_OFFSET};
        bins FRACC_UPPER_REG_INPUT = {AXI_BASE + FRACC_UPPER_REG_OFFSET};
        bins FRACC_LOWER_REG_INPUT = {AXI_BASE + FRACC_LOWER_REG_OFFSET};
        bins SPACING_UPPER_REG_INPUT = {AXI_BASE + SPACING_UPPER_REG_OFFSET};
        bins SPACING_LOWER_REG_INPUT = {AXI_BASE + SPACING_LOWER_REG_OFFSET};
        bins I_COSE_UPPER_REG_INPUT = {AXI_BASE + I_COSE_UPPER_REG_OFFSET};
        bins I_COSE_LOWER_REG_INPUT = {AXI_BASE + I_COSE_LOWER_REG_OFFSET};
        bins I_SINE_UPPER_REG_INPUT = {AXI_BASE + I_SINE_UPPER_REG_OFFSET};
        bins I_SINE_LOWER_REG_INPUT = {AXI_BASE + I_SINE_LOWER_REG_OFFSET};
        bins IRADIUS_REG_INPUT = {AXI_BASE + IRADIUS_REG_OFFSET};
        bins IY_REG_INPUT = {AXI_BASE + IY_REG_REG_OFFSET};
        bins IX_REG_INPUT = {AXI_BASE + IX_REG_REG_OFFSET};
        bins STEP_REG_INPUT = {AXI_BASE + STEP_REG_OFFSET};
        bins SCALE_REG_INPUT = {AXI_BASE + SCALE_REG_OFFSET};
        bins CMD_REG_INPUT = {AXI_BASE + CMD_REG_OFFSET};

    }

    write_data : coverpoint s_vif.s00_axi_wdata {
          bins AXI_WDATA_LOW = {0};
          bins AXI_WDATA_HIGH = {1};
          bins AXI_WDATA_PARAMETERS_1 = {[2:80799]};
          bins AXI_WDATA_PARAMETERS_2 = {[80800:1200000]};
          bins AXI_WDATA_PARAMETERS_3 = {[1200001:16000000]};
          bins AXI_WDATA_PARAMETERS_4 = {[16000001:16777215]};
    }

endgroup

covergroup axi_read_transactions;
    option.per_instance = 1;
    option.goal = 3;
    read_address : coverpoint s_vif.s00_axi_araddr{
        bins READY_ADDRESS = {AXI_BASE + STATUS_REG_OFFSET};
    }

    read_data : coverpoint s_vif.s00_axi_rdata{
        bins READY_RDATA_HIGH = {0};
    }
endgroup
```

Slika 6 :Coverage u AXI monitoru

Details of Instance Cover Point - obj.axi_write_transactions :: write_address

Summary of Cover Point - write_address

Category	Expected	Uncovered	Covered	Percent
User Defined Bins	17	0	17	100

User Generated Bins for Instance Cover Point - obj.axi_write_transactions :: write_address

Covered bins

Name	Hit Count	At least
BASE_ADDRESS	1	1
FRACR_UPPER_REG_INPUT	1	1
FRACR_LOWER_REG_INPUT	1	1
FRACC_UPPER_REG_INPUT	1	1
FRACC_LOWER_REG_INPUT	1	1
SPACING_UPPER_REG_INPUT	1	1
SPACING_LOWER_REG_INPUT	1	1
I_COSE_UPPER_REG_INPUT	1	1
I_COSE_LOWER_REG_INPUT	1	1
I_SINE_UPPER_REG_INPUT	1	1
I_SINE_LOWER_REG_INPUT	1	1
IRADIUS_REG_INPUT	1	1
IY_REG_INPUT	1	1
IX_REG_INPUT	1	1
STEP_REG_INPUT	1	1
SCALE_REG_INPUT	1	1
CMD_REG_INPUT	3	1

Details of Instance Cover Point - obj.axi_write_transactions :: write_data

Summary of Cover Point - write_data

Category	Expected	Uncovered	Covered	Percent
User Defined Bins	5	0	5	100

User Generated Bins for Instance Cover Point - obj.axi_write_transactions :: write_data

Covered bins

Name	Hit Count	At least
AXI_WDATA_LOW	5	1
AXI_WDATA_HIGH	1	1
AXI_WDATA_PARAMETERS_1	6	1
AXI_WDATA_PARAMETERS_2	2	1
AXI_WDATA_PARAMETERS_3	4	1

Details of Instance Cover Point - obj.axi_read_transactions :: read_address

Summary of Cover Point - read_address

Category	Expected	Uncovered	Covered	Percent
User Defined Bins	1	0	1	100

User Generated Bins for Instance Cover Point - obj.axi_read_transactions :: read_address

Covered bins

Name	Hit Count	At least
READY_ADDRESS	84514	1

Details of Instance Cover Point - obj.axi_read_transactions :: read_data

Summary of Cover Point - read_data

Category	Expected	Uncovered	Covered	Percent
User Defined Bins	1	0	1	100

User Generated Bins for Instance Cover Point - obj.axi_read_transactions :: read_data

Covered bins

Name	Hit Count	At least
READY_RDATA_HIGH	84514	1

Takođe, u simple sequnce-u je dodata manja covergrupa koja prikuplja informacije u vrednostima piksela slike i smešta ih u jedan od 3 opsega. Na ovaj način se osigurava da su pokriveni različiti rasponi vrednosti piksela, što doprinosi potpunijem testiranju sistema.

```
covergroup img_data_cover();
        option.per_instance = 2;

       img_upper_pix_value : coverpoint surf_item.img_douta {
            bins low_value    = {[0:61]};          // 0 to 33.33%
            bins mid_value    = {[62:122]};        // 33.33% to 66.66%
            bins high_value   = {[123:184]};       // 66.66% to 100%
        }



        img_lower_pix_value : coverpoint surf_item.img_doutb {
            bins low_value    = {[0:5592405]};         // 0 to 33.33% of the full range
            bins mid_value    = {[5592406:11184810]};  // 33.33% to 66.66% of the full range
            bins high_value   = {[11184811:16777215]}; // 66.66% to 100% of the full range
        }


    endgroup
```

Details of Instance Cover Point - obj.img_data_cover :: img_upper_pix_value

Summary of Cover Point - img_upper_pix_value

Category	Expected	Uncovered	Covered	Percent
User Defined Bins	3	0	3	100

User Generated Bins for Instance Cover Point - obj.img_data_cover :: img_upper_pix_value

Covered bins

Name	Hit Count	At least
low_value	11120	1
mid_value	4298	1
high_value	1223	1

Details of Instance Cover Point - obj.img_data_cover :: img_lower_pix_value

Summary of Cover Point - img_lower_pix_value

Category	Expected	Uncovered	Covered	Percent
User Defined Bins	3	0	3	100

User Generated Bins for Instance Cover Point - obj.img_data_cover :: img_lower_pix_value

Covered bins

Name	Hit Count	At least
low_value	5735	1
mid_value	5530	1
high_value	5376	1

5. Regresija

Regresija je jednostavna i sastoji se od pokretanja više simulacija, pri čemu se u svakoj iteraciji menja set ulaznih parametara slike. Na taj način se obezbeđuje testiranje sistema na različitim podacima, što doprinosi pouzdanosti verifikacije.

```
for {set i 0} {$i < 49} {incr i} {
    set db_name "covdb_$i" ;
    set xsim_command "set_property -name \{xsim.simulate.xsim.more_options\} -value \{-testplusarg UVM_TESTNAME=test_surf_simple -testplusarg UVM_VERBOSITY=UVM_LOW
    -sv_seed random -runall -cov_db_name $db_name\} -objects \[get_filesets sim_1\]"
    eval $xsim_command
    launch_simulation
    run all
    if {$i+1 < 49} {
        close_sim
        puts "Test is over !!!!"
    }
}

puts "Regression is over !!!!"
```

Slika 8 : Regresija

6. Verifikacioni plan

Verifikacioni plan se sastoji iz sledećih koraka:

1. **Provera funkcionalnosti reseta sistema**:
   Reset sistema je aktivan na 0 i ovo se prikazuje na samom početku simulacije.
2. **Provera funkcionalnosti AXI Lite protokola**:
   Ovu proveru obavlja AXI monitor, a korektnost se može videti pomoću analize pokrivenosti (coverage).
   2.1. Provera AXI Lite transakcija upisa (Covergroup za AXI write).
   2.2. Provera AXI Lite transakcija čitanja (Covergroup za AXI read).
   2.3. Provera da li su svi registri AXI Lite protokola uspešno zapisani (Covergroup za AXI i waveform).
   2.4. Provera handshake-a između start i ready signala.
3. **Provera validnog upisa u BRAM memorije**:
   3.1. Provera upisa vrednosti piksela slike u BRAM memoriju slike, što se takođe utvrđuje preko waveform-a.
4. **Provera funkcionalnosti sistema**:
   Ova provera se obavlja putem scoreboard komponente, a ispravnost se potvrđuje kroz poruke koje se šalju iz nje.