Projekat iz predmeta Funkcionalna Verifikacija

Tema : Detektovanje pravih linija na fotografiji I njihovo iscrtvanje koriscenjem Hough transformacije.

Mentor : Nikola Kovačević

Studenti :

Momir Carević EE3/2019

Dejan Pejić EE75/2019

Petar Stamenković EE18/2019

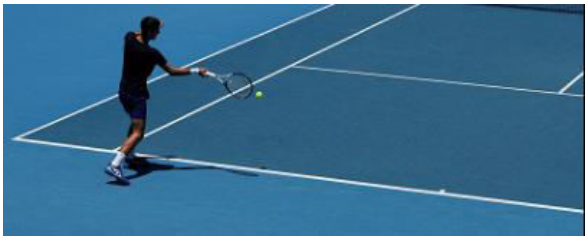
SADRŽAJ

1.Uvod

Ovaj dokument predstavlja specifikaciju i način na koji je realizovana verifikacija sistema koji izvršava Hough transformaciju i time detektuje prave linije 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.

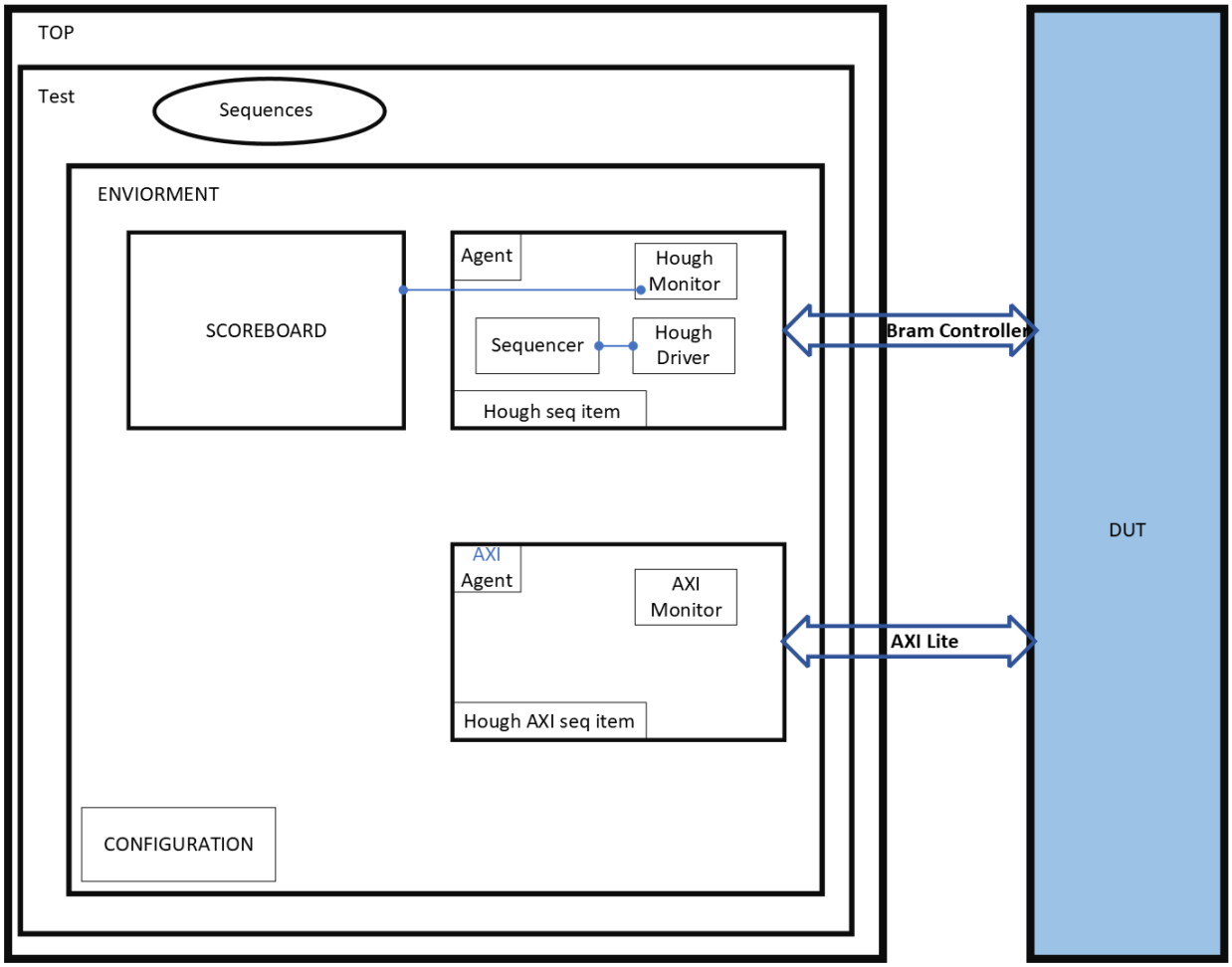
Hough transoformacija je linearna transformacija i ideja je da se svaki piksel iz jednog domena (naša fotografija : x,y osa) prebacuje u drugi domen (trigonometrijski oblik prave : rho i teta). Algoritmu se prosleđuje slika sa već izdvojenim ivicama, dok se tok informacija same Hough transformacija smešta u akumulatore. Maksimalna vrednost akumulatora predstavlja najuočljiviji pravac. Na kraju rada sistema, metodom zlatnih vektora, poredimo izlaze akumulatora sa očekivanim vrednostima dobijenih iz virtuelne platforme.

Slika 2 : Slika nakon transformacije



Slika 1 : Slika pre transformacije

2. Verifikaciono okruženje



Slika 3 : Verifikaciono okruženje

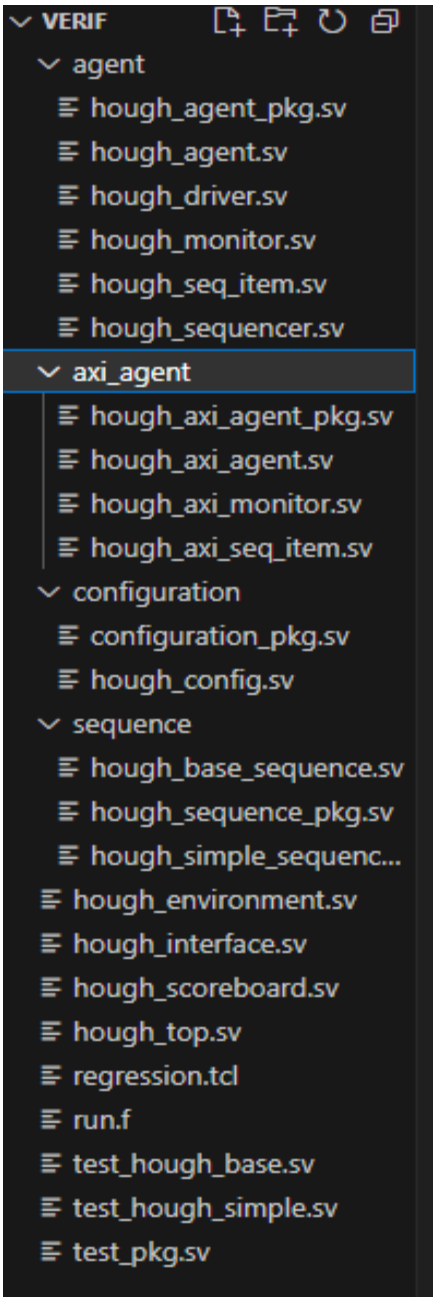
Verifikaciono okruženje se sastoji se od sledećih komponenti :

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

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

Pored navedenih komponenti tu se naravno nalazi i sam dizajn koji se verifikuje (DUT). Glavni protokol koji se koristi za pokretanje sistema i unos parametara slike je AXI Lite, dok se bram memorije za sliku i akumulatore popunjavaju preko AXI Bram kontrolera.

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



Slika 4 : Hijerarhija okruženja

3. Komponente verifikacionog okruženja

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

3.1. Top

U top file-u su ubačeni uvm paketi, uvm makroi i naš test paket. Instanciran je interfejs kao i DUT čiji su signali poverzani sa signalima iz pomenutog interfejsa.

U inital petljama prosleđen je interfejs enviorment-u preko *uvm_config_db,* pokrenut test i obavljen je inicijalni reset sistema kao i generisanje clk signala.

3.2. Enviorment

Deklaracija i dodela virtuelnog interface-a agentima, konfiguraciji i scoreboard-u kao i samo kreiranje pomenutih komponenti. Ovde takođe povezujemo monitor i scoreboard komponentu.

3.3. Interface

U interface fajlu dati su AXI paramteri za širinu podatka i adrese (32 i 5) i deklarisani su signali za axi lite protokol i bram memorije.

```
// Accumulators
logic [17:0]    acc0_addr_cont_i;
logic [31:0]    acc0_data_cont_i;
logic [3:0]     acc0_we_cont;
logic [31:0]    acc0_data_cont_o;
logic [17:0]    acc1_addr_cont_i;
logic [31:0]    acc1_data_cont_i;
logic [3:0]     acc1_we_cont;
logic [31:0]    acc1_data_cont_o;

// Image
logic [18:0]    img_addr_cont_i;
logic [31:0]    img_data_cont_i;
logic [31:0]    img_data_cont_o;
logic [3:0]     img_we_cont;
```

Slika 5 : BRAM signali za sliku i akumulatore

```
// AXI4 Lite interface
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 6 : AXI Lite signali

3.4. Test Paket

Pored uvm paketa i makroa ubacuju se i svi ostali naši paketi (agent,axi agent,sequence,configration) kao i ostale komponente (scoreboard,enviorment,test base i test simple) koje nisu unutar nekog paketa pa i interface.

3.4.1 Base Test

U test base-u se kreiraju enviorment i konfiguracija unutar koje se randomizuje slika koja ce biti obradjena. Takodje se poziva i funkcija koja ce izvuci podatke iz tekstualnih fajlova.

3.4.2 Test Simple

Test simple nasleđuje test base i pušta 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 vrednosti za sliku i akumulatore dok to nije premešteno u konfiguracioni fajl.

3.5.2. Simple Sequence

Ovaj fajl je dosta bitan jer se pomoću njega prosleđuje korektna sekvenca za pokretanje i rad samog sistema. Ovde su definisana početna adresa AXI protokola kao i ofseti kojima gađamo potrebne registre (Start, Reset, Width, Height, Rho, Threshold i Ready). Nakon uspešne inicjalizacije sistema, preko pomenutih ofseta se zadaju parametri za sliku, a zatim se bram memorije za akumulatore paralelno popunjavaju nulama (preload) dok se bram memorija za sliku popunjava samom slikom (vrednosti piksela). Na kraju, kad je sve spremno, start signal se podiže na 1 i sistem čeka da se vrednost ready signala podigne na 1 (kraj obrade).

U tom momentu poslednjom for petljom prolazimo kroz oba akumulatora i imamo pristup tim vrednostima preko *acc0_data_cont_o* i *acc1_data_cont_o* koje ćemo koristiti kasnije u scoreboard komponenti.

3.6. Agent (aktivni)

U aktivnom agentu uključujemo naš driver, monitor za nadgledanje izlaza sistema, sekvencer i sequence item. Povezujemo sequence item i driver.

3.6.1. Sequence Item

Sequence item poseduje u sebi sve signale interfejsa kao I dodatni *bram_axi* signal koji će driver-u dati informaciju da li radi sa bram memorijama ili axi protokolom.

3.6.2. Driver

Kao što je već rečeno, režim rada driver-a se odlučuje na osnovu *bram_axi* signala koji je jednobitan. Ukoliko 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 preko 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 fajlom, metodom zlatnih vektora, utvrdi validnost rada dizajna. U *main* fazi, je pomoću wait naredbi , omogućeno da monitor nadgleda od momenta kada počinje isčitavanje vrednosti iz akumulatora. Hvatamo adrese I vrednosti na njima I pomoću write funckije komuniciramo sa scoreboardom. Ovde je takođe implementiran deo coverage-a ali to će biti objašnjeno u posebnom poglavlju.

3.7. AXI Agent

Ovo je pasivni agent koji sarži samo monitor koji nadgleda axi lite transakcije na ulazu sistema. Ukoliko su određeni signali (*axi_awvalid I axi_awready*) aktivni, skupljamo informacije o toj transakciji upisa u *curr_it*, sequence itemu koji se kreira u monitoru. Analogno ovome, ako su signali *axi_rvalid* I *axi_arready* aktivni onda skupljamo informacije o transakciji čitanja. Takođe I ovde je implementiran coverage.

3.8. Scoreboard

Kao što je više puta napomenuto, za potvrdu rezultata je korišćena metoda zlatnih vektora I ona je ovde implementirana. Scoreboard je povezan sa monitorom I jednostavno poredi vrednosti koje je monitor uhvatio sa vrednostima iz golden vector fajlova koji su dobijeni iz virtuelne platforme. Koriste se assert naredbe za poređenje I ako je primećena greška, pomoću *uvm_error* poruke se ona prijavljuje u tcl konzoli.

3.9. Konfiguracija

U konfiguarionom fajlu se radi više stvari.

1. Pomoću *uvm_active_passive_enum* makroa određujemo da li su agenti pasivni ili aktivni.
2. Date su relativne putanje za svaki tekstualni fajl kako bi se test mogao pokrenuti na bilo kojem računaru bez modifikacija.

3. Randomizuje se koja slika se obrađuje u nizu više slika koje su u files folderima.
4. Implementirane su funkcije za učitavanje parametara slika (takođe iz tekstualnog fajla), popunjavanje memorija za akumulatore I sliku kao I čitanje iz fajlova za zlatne vektore.

4. Coverage analiza

Coverage (pokrivenost) prikupljamo na par različitih mesta od kojih su najbitniji sami monitori. Na slici 5, prikazan je coverage za izlazni monitor gde prikupljamo informacije o opsegu adresa koje se čitaju. Na slici 6 je prikazan coverage za axi montor gde je zadat tačan goal jer znamo da je uvek potreno da se desi 9 transakcija za upis I 3 za čitanje kako bi sistem radio kako treba. Takođe imamo I coverpoint za vrednosti registara poput start I ready koji kontolišu sam rad sistema.

```
covergroup acumulator_cover (int coverage_goal);
    option.per_instance = 1;
    option.goal = coverage_goal;
    acc0_address : coverpoint h_vif.acc0_addr_cont_i{
        bins b1 = {[0:6400]};
        bins b2 = {[6401:12800]};
        bins b3 = {[12801:19200]};
        bins b4 = {[19201:25600]};
        bins b5 = {[25601:32000]};
        bins b6 = {[32001:38400]};
        bins b7 = {[38401:44800]};
        bins b8 = {[44801:51200]};
        bins b9 = {[51201:57600]};
        bins b10 = {[57601:64000]};
    }
    acc1_address : coverpoint h_vif.acc1_addr_cont_i{
```

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	7	0	7	100

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

Covered bins

Name	Hit Count	At least
BASE_ADDRESS	60	1
START_REG_INPUT	60	1
RESET_REG_INPUT	40	1
WIDTH_REG_INPUT	20	1
HEIGHT_REG_INPUT	20	1
RHO_REG_INPUT	20	1
TRESHOLD_REG_INPUT	20	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	3	0	3	100

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

Covered bins

Name	Hit Count	At least
AXI_WDATA_LOW	60	1
AXI_WDATA_HIGH	40	1

Slika 9 : AXI Write coverage

```
covergroup axi_write_transactions;
    option.per_instance = 1;
    option.goal = 9;
    write_address : coverpoint h_vif.s00_axi_awaddr{
        bins BASE_ADDRESS = {AXI_BASE};
        bins START_REG_INPUT = {AXI_BASE+START_REG_OFFSET};
        bins RESET_REG_INPUT = {AXI_BASE+RESET_REG_OFFSET};
        bins WIDTH_REG_INPUT = {AXI_BASE+WIDTH_REG_OFFSET};
        bins HEIGHT_REG_INPUT = {AXI_BASE+HEIGHT_REG_OFFSET};
        bins RHO_REG_INPUT = {AXI_BASE+RHO_REG_OFFSET};
        bins TRESHOLD_REG_INPUT = {AXI_BASE+TRESHOLD_REG_OFFSET};
    }
    write_data : coverpoint h_vif.s00_axi_wdata{
        bins AXI_WDATA_LOW = {0};
        bins AXI_WDATA_HIGH = {1};
    }
endgroup

roup axi_read_transactions;
tion.per_instance = 1;
tion.goal = 3;
ad_address : coverpoint h_vif.s00_axi_araddr{
  bins READY_ADDRESS = {AXI_BASE+READY_REG_OFFSET};

ad_data : coverpoint h_vif.s00_axi_rdata{
  bins READY_RDATA_LOW = {0};
  bins READY_RDATA_HIGH = {1};

up
```

8: Coverage u axi monitoru

Slika 7 : Coverage u izlaznom monitoru

Total Groups Coverage Summary

Score	Inst Score
100	100

Slika 10: Ukupni Coverage

User Generated Bins for Instance Cover Point - obj.axi_read_transactions

Covered bins

Name	Hit Count	At least
READY_RDATA_HIGH	2	1

Slika 11: AXI Read coverage

Takođe, u simple sequnce-u je ubačena jedna manja covergrupa koja prikuplja informacije u vrednostima piksela slike i smešta ih u jedan od 3 opsega.

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

Summary of Cover Point - img_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_pix_value

Covered bins

Name	Hit Count	At least
low_value	1376236	1
medium_value	81444	1
high_value	69890	1

Slika 12 : Image Cover po vrednostima piksela

5. Regresija

Regresija je trivijalna i svodi se na puštanje više simulacija uz promenu slike u svakoj iteraciji.

regression.tcl

```
for {set i 0} {$i < 5} {incr i} {
    set db_name "covdb_$i" ;
    set xsim_command "set_property -name \{xsim.simulate.xsim.more_options\} -value \{-testplusarg UVM_TESTNAME=test_hough_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 < 5} {
        close_sim
    }
}
```

Slika 13 : Regresija

6. Verifikacioni plan

Verifikacioni plan se sastoji iz sledećih koraka :

1. Provera funkcionalnosti reseta sistema. Reset je aktivan na 0 i ovo je pokazano na samom početku simulacije.

2. Provera funkcionalnosti AXI Lite protokola. Ovo proverava axi monitor i korektnost se vidi pomoću coverage analize na istom.
   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 se desio upis u sve registre axi lite protokola. (Covergroup za axi i waveform)
   2.4. Provera handshake-a start i ready signala.

3. Provera validnog upisa u bram memorije.
   3.1. Provera upisa nula u bram memorije akumulatora pre samog početka rada sistema. Ovo je utvrđeno preko waveforma
   3.2. Provera upisa vrednosti piksela slike u bram memoriju slike. Takođe utvrđeno preko waveforma.

4. Provera funkcionalnosti sistema. Ovo je utvrđeno preko scoreboard komponente i poruka koje se šalju preko nje.