Verifikacija hardvera. Primeri. Jezici Verilog i VHDL.

Seminarski rad u okviru kursa
Verifikacija softvera
Matematički fakultet

Tomislav Milovanović 1091/2018
tomislav.milovannovic@gmail.com

11. decembar 2018

Sažetak

U ovom radu će biti predstavljene metode i značaj verifikacije hardvera. Kako sam proces verifikacije vremenski najzahtevniji prilikom proizvodnje hardvera, neophodno je na vreme se odlučiti za tehnike i metode verifikacije kako bi se postigli najbolji rezultati.

Sadržaj

1 Uvod

Greške predstavljaju sastavni deo naših života i pojavljuju se svuda gde ljudi sprovode akcije i donose odluke. Dok se greške u softveru mogu otkloniti prepravkom koda i ažuriranjem softvera, greške u hardveru zahtevaju ponovnu proizvodnju i isporuku proizvoda. Upravo zbog težine i cene otklanjanja grešaka kada je u pitanju proizvodnja hardvera neophodno je utrošiti više vremena na detekciju i prevenciju istih, nego što se utroši na dizajn same komponente. Statistička istraživanja pokazuju da je u proseku 60-70% celokupnog vremena koje je potrebno za proizvodnju nekog hardvera rezervisano za proces verifikacije. Pronalaženjem efikasnih i pouzdanih mera verifikacije može se značajno smanjiti vreme potrebno za razvoj samog proizvoda, što je za kompanije veoma značajno.

2 Metodi

Postoji dosta metoda i tehnika koji se primenjuju prilikom verifikacije hardvera, a najčešće se koriste emulacija, simulacija i fabričko testiranje.

2.1 Emulacija

Proces emulacije podrazumeva imitaciju ponašanja nekog dela hardvera korišćenjem generičkog hardvera koji se naziva emulator (eng. emulator). Kako bi se postiglo traženo ponašanje emulator se mora konfigurisati na određeni način. Tako konfigurisan biva podvrgnut intenzivnom testiranju kako bi se proverilo da li proizvedeno ponašanje odgovara specifikaciji.

2.2 Simulacija

Proces simulacije podrazumeva korišćenje jezika za opis hardvera kako bi se opisao model koji se razmatra. Uz pomoć tako kreiranog modela se može proveriti ponašanje nekog elektronskog sistem dok još uvek nije fizički kreiran. Jezici koji se najčešće koriste su VHDL i Verilog.

2.3 Fabričko testiranje

Fabričko testiranje se sprovodi prilikom samog procesa proizvodnje, pre nego što proizvod napusti fabriku. Ovaj proces je uglavnom automatizovan, i sprovodi se kroz više iteracija kako bi se pokrili svi test slučajevi. Često se, pored toga da li proizvod zadovoljava specifikaciju, proverava i njegova funckionalnost.

3 Jezici za opis hardvera

Jezici za opis hardvera spadaju u posebnu vrstu programskih jezika koja se koristi za opis strukture i ponašanja elektronskih i digitalnih kola. Na taj način dobijamo precizan i formalan opis kola što omogućava automatsku analizu i simulaciju. Dva jezika koja se najčešće koriste od strane digitalnih dizajnera su **VHDL** i **Verilog**.

3.1 VHDL

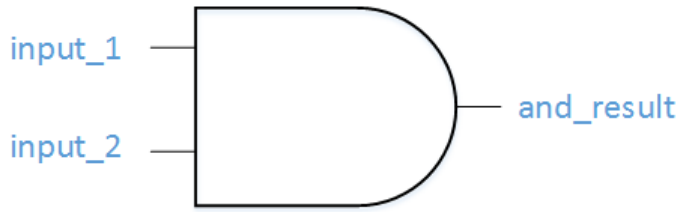
VHDL (Very High Speed Integrated Circuit Description Language) je jezik za opis hardvera koji se koristi u automatizaciji elektronskog dizajna za opisivanje digitalniih sistema.

VHDL je originalno nastao sa ciljem da se dokumentuje ponašanje integrisanih kola dizajniranih tako da budu konfigurisana od strane potrošača nakon proizvodnje (ASIC - Application-specific integrated circuit). To je dalje uticalo na razvoj logičkih simulatora koji su čitali VHDL fajlove kao i alata za sintezu logičkih kola, koji će kao ulaz imati VHDL fajlove a izlaz fizičku definiciju implementacije strujnih kola.

VHDL je strogo tipiziran jezik i nije osetljiv na mala i velika slova. Osnovni koncepti i sintaksta su preuzeti iz programskog jezika Ada. Kako poseduje ulazno - izlazne mogućnosti može se koristiti i kao jezik opšte namene. Međutim, postoji dosta karakteristika koje ovaj jezik poseduje kako bi mogao direktno da predstavi operacije uobičajene za hardver - negacija konjukcije (**nand**), negacija disjunkcije (**xor**), negacija ekskluzivne disjunkcije (**xnor**), itd.

Obično se koristi za pisanje tekstualnih modela kojima opisuje logička kola, koji se zatim obrađuju sinteznim programom ukoliko pripadaju logičkom dizajnu. Grupa tih modela se obično naziva *testbench*. Svaka transakcija se dodaje u red za tačno određeno vreme. Na primer, ukoliko prenos određenog signala treba da se desi nakon 2 nanosekunde, dodaje se u red za vreme +2ns. Takođe, dozvoljeno je i nulto odlaganje, ali se i ono mora zakazati. Prilikom simulacije se menjaju dva režima: izvršenje izvođenja i obrada događaja.

Primer 3.1 *Na slici 1 prikazano je I - kolo, i u nastavku kod kojim se opisuje to kolo.*



Slika 1: I - kolo

```
library ieee;
use ieee.std_logic_1164.all;

entity example_and is
port (
input_1    : in  std_logic;
input_2    : in  std_logic;
and_result : out std_logic
);
end example_and;

architecture rtl of example_and is
signal and_gate : std_logic;
```

```
begin
and_gate   <= input_1 and input_2;
and_result <= and_gate;
end rtl;
```

Glavna prednost VHDL-a je to što omogućava modeliranje ponašanja sistema i njegovu verifikaciju pre kreiranja samog hardvera. Takođe, za razliku od proceduralnih programskih jezika koji se izvršavaju sekvencijalno, VHDL ima konstruktore koji se bave paralelizmom koji je svojstven za hardverske projekte.[?]

3.2 Verilog

Verilog je drugi jezik za opis hardvera koji se koristi za modelovanje elektronskih sistema. Pored toga, koristi se za verifikaciju analognih kola i kola sa mešovitim signalima.

Poseduje dosta sličnosti sa ostalim softverskim programskim jezicima jer sadrži načine opisivanja širenja vremena i jačine signala. U pitanju je jezik koji je osetljiv na velika i mala slova i sintaksno ima dosta sličnosti za programskim jezikom C. Ono što je drugačije je postojanje neblokirajućeg operatora dodele ($<=$) koja omogućava ažuriranje stanja mašine bez potrebe za deklaracijom i privremenim skladištenjem varijabli. Na taj način se mogu napisati opisi velikih kola u relativno kompaktnom obliku što je dosta uticalo na produktivnost dizajnera kola.[?]

Osnovni koncept u Verilog-u je koncept modula. Modul predstavlja osnovnu jedinicu dizajna, najčešće jednu hardversku komponentu. U jednom fajlu se može definisati više modula, mada se preporučuje da se za svaki modul kreira novi fajl koji se zove isto kao i modul. Svaki modul može imati ulaze i izlaze (portove) koji predstavljaju signale. U okviru većih modula se mogu istancirati manji, što odgovara korišćenju jednostavnijih logičkih kola u okviru složenijih. Postoje i moduli koji se koriste isključivo za testiranje drugih modula i oni uglavnom nemaju portove.

Izvršavanje Verilog simulatora se sastoji od većeg broja procesa koji se izvršavaju paralelno, u proizvoljnom redosledu. Pod procesom podrazumevamo svaki *gejt* (eng. gate), svaku *assign* naredbu kontinuirane dodele, *initial i always*. Za svaki proces se vezuju dva tipa događaja: događaj evaluacije (izračunavanje koje se obavlja u tom procesu) i događaj ažuriranja vrednosti signala (promena vrednosti nekog signala). Događaji ažuriranja kreiraju nove događaje evaluacije. Događaji evaluacije kreiraju nove događaje ažuriranja ako i samo ako se evaluacijom dobije vrednost koja je različita od tekuće vrednosti signala kome se ta vrednost dodeljuje.

Kako bi se raspoređivanje nekog događaja odložilo za neku vremensku jedinicu, neophodno je definisati kašnjenje. Kašnjenja je moguće definisati na nivou gejtova, za *assign* naredbu, kao i za naredbe *initial i always*. Takođe, kod initial i always naredbi mogu se navesti *@() i wait()* konstrukcije kojima se odlaže evaluacija naredbi do ispunjena određenih uslova.

Primer 3.2 *U nastavku je kod koji opisuje kolo prikazano na slici 1.*

```
module andgate (input_1, input_2, and_result);
input input_1, input_2;
output and_result;
assign and_result = input_1 & input_2;
endmodule
```

Svi događaji se raspoređuju u redove:

- **Qa** - red aktivnih događaja, raspoređeni za obradu u tekućojvremenskoj jedinici
- **Qi** - red neaktivnih događaja, raspoređeni za obradu u tekućoj vremenskoj jedinici sa kašnjenjem #0
- **Qn** - red neblokirajućih događaja, događaji ažuriranja nastali iz neblokirajućih dodela raspoređenih za izvršavanje u tekućoj vremenskoj jedinici
- **Qm** - red monitorskih događaja, $monitor događaji koji se kreiraju u svakoj vremenskoj jedinici za svaki od monitora koji postoje u kolu
- **Qf** - red budućih događaja, raspoređeni za izvršavanje u nekoj od sledećih vremenskih jedinica

4 Zaključak

U ovom radu predstavljen je značaj verifikacije hardvera prilikom procesa proizvodnje. Istaknute su metode koje se najviše koriste, kao i dva najpoznatija jezika za opis hardvera VHDL i Verilog. Predstavljeni su osnovni koncepti jezika kao i karakteristike koje ih izdvajaju u odnosu na druge programske jezike.

Literatura