# Jedan pristup verifikaciji simulatora digitalnog signal procesora

Aleksandar Zorić, Miodrag Đukić, Nebojša Zorić, Jelena Kovačević, Stanislav Očovaj

***Sadržaj* — U ovom radu je opisan problem verifikacije simulatora jezgra Coyote 32-bitnog digitalnog signal procesora. Problem verifikacije se zasniva na definisanju skupa ispitnih slučajeva koje simulator mora da zadovolji da bi se rad simulatora smatrao tačnim. U ovom radu su definisane tri vrste ispitivanja: namensko ispitivanje, ispitivanje na principu crne kutije i ispitivanje na konkretnoj aplikaciji. Simulator je realizovan u C programskom jeziku.**

***Ključne reči* — Simulator, verifikacija.**

## I. UVOD

SIMULATORI su računarski programi koji imitiraju rad nekog konkretnog uređaja. Imitiranje podrazumeva obrađivanje istih vrsta podataka koje obrađuje i sam uređaj i davanje identičnih rezultata. Simulatori koji imitiraju tačnost obavljanja naredbi koje izvršava konkretan uređaj, stavljajući u prvi plan brzinu i kontrolu podataka koji se obrađuju, nazivaju se simulatorima koji imitiraju tačnost izvršavanja komandi (*Instruction set simulator*). U ovom radu je opisana je programska podrška koja imitira tačnost izvršavanja komandi. Konkretan uređaj koji se simulira je Coyote 32 bit digitalni signal procesor (DSP).

Ispitivanje tačnosti rada takvog simulatora zasniva se na poređenju rezultata obrade simulatora i DSP-a, za identične ulazne podatke. Problem se ogleda u nepoznavanju skupa podataka pomoću kojih bi se jednoznačno ispitala tačnost rada simulatora.

## II. OPIS OKRUŽENJA

### A. Opis digitalnog signal procesora

Coyote 32-bitni procesor je DSP zasnovan na arhitekturi nepokretnog zareza. Pored više vrsta arhitektura kojom može biti realizovan jedan DSP, ovaj DSP je realizovan u Harvardskoj arhitekturi. Poseduje protočnu strukturu sa dva jezgra [2]. Coyote 32-bitni procesor pripada grupi procesora sa dugačkom instrukcijskom reči.

Jezgro Coyote 32 sadrži dve vrste memorije:

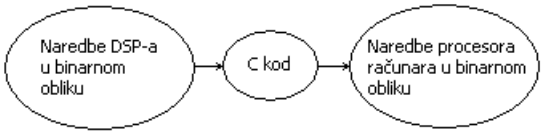
- Programska memorija (P memorija) – sadrži niz naredbi koje treba da se izvrše. Ova vrsta memorije može da čuva do $2^{16}$ reči širine 32-bita.
- Memorijski prostor (X i Y memorijski prostor) – čuva podatke koji se koriste u operacijama. Vrsta operacije je definisana u opisu naredbe. Oba memorijska prostora mogu da čuvaju po $2^{16}$ reči širine 32-bita.

Registri koji se koriste u radu Coyote 32 DSP-a se dele na 5 grupa:

- Osam 72-bitnih registara akumulatora
- Osam 32-bitnih registara podataka
- Dvanaest 16-bitnih adresnih registara
- Četiri 12-bitnih moduo registrara
- Registri koji čuvaju informacije o stanju DSP-a u kojem se procesor trenutno nalazi

### B. Opis simulatora

Simulator ima zadatak da imitira rad DSP-a na računaru na kome se izvršava. Ako se naredbe DSP-a razlikuju od naredbi procesora na kojem se simulator izvršava, naredbe DSP-a je potrebno prevesti u naredbe procesora računara. Naredbe DSP-a se ne prevode neposredno. Za realizaciju simulatora opisanog u ovom radu, korišćen je C programski jezik i C programski prevodilac, koji prevodi C naredbe u naredbe procesora na kome se simulator izvršava (sl. 1)




Sl. 1. Tok prevođenja naredbi.

Centralni deo simulatora predstavlja model DSP-a, realizovan u C jeziku. Registri DSP-a su modelovani kao strukture podataka, koje se razlikuju za svaki od tipova registara. Memorija je modelovana da omogući čuvanje $2^{16}$ (16-bita) 32-bitnih vrednosti. Sve matematičke operacije – sabiranja, oduzimanja, množenja itd. nad registrima i nad memorijom DSP modela, realizovani su u obliku makroa i funkcija.

Ovaj rad delimično je finansiran od Ministarstva za nauku Republike Srbije, projekat 12004, od 2008. God.

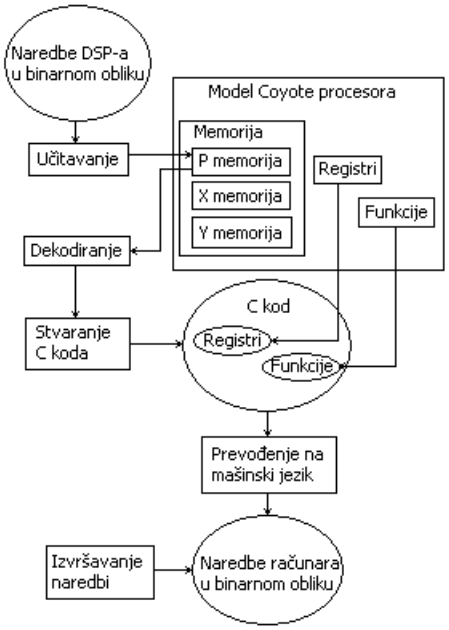
Aleksandar Zorić, Fakultet Tehničkih Nauka u Novom Sadu, Srbija, (e-mail: aleksandar.zoric@rt-sp.com).

Miodrag Đukić, Fakultet Tehničkih Nauka u Novom Sadu, Srbija, (e-mail: miodrag.djukic@rt-sp.com).

Nebojša Zorić, Fakultet Tehničkih Nauka u Novom Sadu, Srbija, (e-mail: nebojsa.zoric@rt-sp.com).

Jelena Kovačević, Fakultet Tehničkih Nauka u Novom Sadu, Srbija (e-mail: jelena.kovacevic@rt-sp.com).

Stanislav Očovaj, Fakultet Tehničkih Nauka u Novom Sadu, Srbija (e-mail: stanislav.ocovaj@rt-sp.com )




Sl. 2. Dizajn simulatora.

Prvi korak u simulaciji DSP-a predstavlja učitavanje binarnih naredbi DSP-a u P memoriju DSP modela. Naredbe iz P memorije se dekodiraju i prevode na C jezik, gde svakoj naredbi DSP-a odgovara jedna funkcija C jezika. U svakoj funkciji mogu da se obavljaju operacije nad registrima i nad memorijom DSP modela. Niz C funkcija se dobija prevođenjem DSP naredbi i one čine C programski kod. C programski prevodilac prevodi C kod na mašinski jezik procesora. Izvršavanjem mašinskih naredbi izvršava se simulacija DSP-a. Više informacija o načinu rada simulatora se može naći u [5].

## III. REŠENJE PROBLEMA

Usled nedostatka tačno definisanog skupa ispitnih slučajeva koje simulator mora da zadovolji, da bi se smatrao tačnim, u ovom radu definisana su sledeća tri skupa ispitnih slučajeva:

- Namensko ispitivanje
- Ispitivanje na principu crne kutije
- Ispitivanje na konkretnoj aplikaciji

### *A. Namensko ispitivanje*

Namensko ispitivanje ispituje tačnost makroa i funkcija realizovanih sa namenom da podrže operacije nad registrima i nad memorijom DSP-a.

Namenska ispitivanja se zasnivaju na određivanju skupa ulaznih i izlaznih vrednosti markoa i funkcija. Ulazne vrednosti predstavljaju vrednosti koje će se koristiti u radu makroa i funkcija, a izlazne vrednosti predstavljaju rezultat njihovog rada. Svaka grupa makroa i funkcija poseduje svoje jedinstvene ulazne i izlazne vrednosti. Grafički prikaz namenskog ispitivanja dato je na sl. 3.


Sl. 3. Namensko ispitivanje.

Skup ulaznih vrednosti se formira na osnovu sopstvenog iskustva projektanta koji određuje ulazne vrednosti ključne za ispitivanje rada makroa i funkcija. Ovaj način ispitivanja predstavlja namensko ispitivanje funkcionalnosti makroa i funkcija u cilju realizacije što tačnijeg modela registra i njegovog ponašanja. Ulazne vrednosti koje su ključne za ispitivanje rada makroa i funkcija se određuju na osnovu:

- sopstvenog iskustva projektanta koji realizuje simulator DSP-a
- opisa rada instrukcija koje su relizovane pomoću makroa i funkcija
- priloženih primera rada nad registrima u dokumentaciji DSP-a

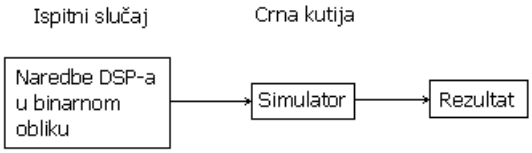
Skup izlaznih vrednosti koje odgovaraju ulaznim vrednostima se određuju na osnovu:

- Rezultujućih vrednosti registara dobijenih operacijom nad registima DSP-a
- Priloženih primera rada nad registrima u dokumentaciji DSP-a

Unapred definisanim skupom ulaznih i izlaznih vrednosti, poredi se izlazna vrednost makroa i funkcija sa očekivanom izlaznom vrednošću. Rezultat ispitivanja se smatra tačnim ukoliko su izlazne vrednosti jednake očekivanim.

### *B. Ispitivanja na principu crne kutije*

Da bi ispitivanje simulatora bilo potpuno potrebno je ispitati rad simulatora od učitavanja binarnih DSP naredbi do izvršavanja naredbi na procesoru računara na kojem se simulator izvršava. Simulator je potrebno gledati kao crnu kutiju na čiji se ulaz daju ispitni slučajevi, a na izlazu se posmatraju rezultati izvršavanja naredbi. Grafički prikaz rada na principu crne kutije, dat je na sl. 4.




Sl. 4. Korišćenje simulatora na principu crne kutije.

Skup ispitnih slučajeva kojima se proverava tačnost simulatora, čine dva podskupa:

- ispitni slučajevi koji proveravaju GNU C programski prevodilac (GCC).
- ispitni slučajevi napisani na osnovu sopstvenog iskustva projektanta u korišćenju simulatora

GCC je programski prevodilac koji prevodi kod napisan u C jeziku na jezik namenjen GNU operativnom sistemu. Skup GNU C ispitnih slučajeva sadrži ispitne slučajeve

koji su namenjeni ispitivanju rada GCC-a. Za potrebe ispitivanja simulatora, iz ove grupe ispitivanja isključena su ispitivanja koja koriste arhitekturu pokretnog zareza i ispitivanja čije instrukcije nisu podržane od strane DSP-a koji se simulira.

Svi ispitni slučajevi napisani su u C programskom jeziku. Svaki ispitni slučaj proverava funkcionalnost određenog dela simulatora. Ispitni slučaj definiše vrednost ulaza u simulator i očekivani rezultat koji simulator treba da proizvede da bi se smatrao tačnim. Na sl. 5 je prikazan primer ispitnog slučaja napisanog u C programskom jeziku i očekivani rezultat rada simulatora.

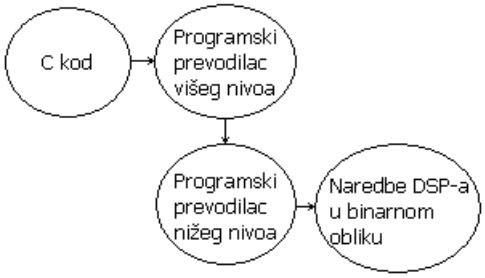
```
f (x)
{
  if (x != 0 || x == 0)
    return 0;
  return 1;
}

main ()
{
  if (f (3))
    abort ();  <-- Greška u simuliranju
  exit (0);  <-- Očekivani rezultat
}
```

Sl. 5. Ispitni slučaj napisan u C kodu.

Ukoliko bi se ispitivanje završilo pozivom funkcije *abort*, rezultat ispitivanja bi ukazivao na grešku u radu simulatora. Problem bi se u ovom slučaju odnosio na netačno izvršavanje funkcije *f(x)*.

Ispitni slučaj napisan u C programskom kodu, potrebno je prevesti u niz binarnih naredbi DSP-a. Prevođenje se vrši u dve faze. Prva faza predstavlja prevođenje C programskog koda na niži programski jezik pomoću programskog prevodioca višeg nivoa. U drugoj fazi niži programski jezik se prevodi pomoću programskog prevodioca nižeg nivoa u niz binarnih naredbi DSP-a, kao što je prikazano na sl. 6




Sl. 6. Faze prevođenja C koda.

Kao programski prevodilac višeg nivoa korišćen je *Cirrus C compiler* (skr. CCC), razvijen na Katedri za računarsku tehniku i računarske komunikacije na Fakultetu tehničkih nauka u Novom Sadu. Dobijene binarne naredbe DSP-a, predstavljaju ulaz u simulator, kao što je prikazano na sl. 4. Izvršavanjem naredbi DSP-a, dobijaju se rezultati rada simulatora. Ukoliko su rezultati jednaki očekivanim, rad simulatora se smatra tačnim.

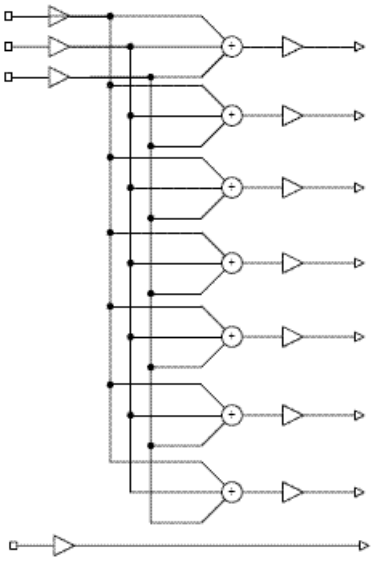
## IV. Ispitivanje na konkretnoj aplikaciji

U poslednjoj fazi ispitivanja simulatora, simulator je ispitivan u radu nad konkretnom aplikacijom – *Onkyo OB*. Aplikacija je razvijena od strane *Onkyo*-a, a implementiran je na Katedri za računarsku tehniku i računarske komunikacije na Fakultetu tehničkih nauka u Novom Sadu. *Onkyo OB* je aplikacija koja izvršava logičko povećanje broja izlaznih signala u odnosu na broj ulaznih signala. Simulacijom ove aplikacije ispituje se funkcionisanje simulatora u konkretnoj aplikaciji.

Aplikacija se ispituje poređenjem izlaznih rezultata rada DSP-a i simulatora, kao što je prikazano na sl. 7.


Sl. 7. Diagram ispitivanja *Onkyo OB* aplikacije.

Ulazni podaci predstavljaju četvorokanalni ulaz, a izlaz predstavlja logički proširen, osmokanalni izlaz, kao što je prikazano na sl.8.



Sl. 8. *Onkyo OB* aplikacija.

Uz svaki ulazni i izlazni kanal pridružen je koeficijent koji utiče na vrednost ulaznog ili izlaznog podatka. Izlazni podaci dobijaju se matematičkim operacijama nad predviđenim ulaznim podacima. Ukoliko su podaci nakon obrade DSP-a i simulatora istovetni, za iste ulazne podatke i koeficiente, rezultat rada simulatora se smatra tačnim.

Ova vrsta ispitivanja nije detaljna kao ispitivanje na principu crne kutije, ali ona daje informaciju o tačnosti rada simulatora u konkretnoj aplikaciji.

## V. Zaključak

Problem ispitivanja tačnosti simulatora proizilazi iz nedostataka tačno definisanog skupa ispitivanja koje mora da zadovolji simulator da bi odgovarao funkcionalnosti DSP-a. Usled nedostatka definisanog skupa ispitnih slučajeva, koristi se kombinacija više različitih vrsta ispitivanja.

Simulator DSP-a, opisan u ovom radu, imitira 68% naredbi od ukupnog broja naredbi DSP-a. Grupa

ispitivanja, opisana u ovom radu, ispituju tačnost realizacije tih instrukcija. Naredbe koje nisu pokrivene opisanim grupama ispitivanja, čine skup manje korišćenih naredbi izdvojenih na osnovu sopstvenog iskustva projektanta simulatora.

Pravac daljeg razvoja bi bio bolje definisanje skupa ispitnih slučajeva koje treba da zadovolji simulator DSP-a. Definisanjem skupa ispitivanja povećava se pouzdanost ispitivanja simulatora DSP-a.

## LITERATURA


[1] Stefan Kraemer, Lei Gao, Jan Weinstock, Rainer Leupers, Gerd Ascheid and Heinrich Meyr; HySim: A Fast Simulation Framework for Embedded Software Development, Institute for Intergrated Signal Processing Systems RWTH Aachen Universitz, GermanyCirrus Clogic, 2005W.K. Chen, Book style. Belmont, CA: Wadsworth, 1993, pp. 1–15.

[2] Zarić Zoran, Miodrag Đukić, Marko Gajić, Miroslav Popović, Jedan koncept optimizacione tehnike za iskorišćenje adresne jedinice u C kompajleru, Univerzitet u Novom Sadu, Fakultet tehničkih nauka, Odsek za računarsku tehniku i računarske komunikacijeB. Smith, "Unpublished work style," unpublished.

[3] Coyote 32 bit DSP: Instruction Set and Arhitecture Reference Manual DS641ISA3, Cirrus Clogic, 2005G.

[4] Rob savoye, DejaGnu – The GNU testing framework, Free software Foundation

[5] Miodrag Đukic, Nenad Četić, Radovan Obradović, Miroslav Popović - An Approach to Instruction Set Compiled Simulator Development Based on a Target Processor C Compiler Back-End Design, ECBS EERC 2009



## ABSTRACT

This paper describes the problem of verification a simulator of Coyote 32 bit digital signal processor core. The problem of verification is based on defining a group of tests which a simulator have to pass to consider it is verified. In this paper there are defined tree types op tests: purpose tests, tests based on a priciple of a black box and tests on an a working aplication. Simulator is implemented in C programing language.


## AN APPROACH TO THE VERIFICATION OF DIGITAL SIGNAL PROCESOR SIMULATOR


Aleksandar Zorić, Miodrag Đukić, Nebojša Zorić, Jelena Kovačević, Stanslav Očovaj