Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 681.324.06
DOI: https://doi.org/10.24867/28BE21Djokovic

IMPLEMENTACIJA I VERIFIKACIJA SISTEMA ZA PROCESIRANJE PAKETA PRIMENOM FORMALNIH METODA

IMPLEMENTATION AND VERIFICATION OF PACKET PROCESSING SYSTEM UTILIZING FORMAL METHODS

Aleksa Đoković, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U radu je predstavljena implementacija sistema za procesiranje paketa koja podrazumeva izgradnju paketa na osnovu ulaznih podataka i konfiguracije, i parsiranje dolaznih paketa kako bi se ekstrahovale potrebne informacije i utvrdilo postojanje grešaka prilikom prenosa. Sistem je verifikovan FPV formalnom tehnikom (eng. Formal property verification), dok je otpornost na greške testirana primenom FSV formalne tehnike (eng. Formal Safety Verification).*

Ključne reči: *procesiranje paketa, formalna verifikacija, FPV, FSV, Hemingov kod, otpornost na greške*

Abstract – *The paper presents the implementation of the packet processing system, which includes the construction of packets based on input data and configuration, and the parsing of incoming packets in order to determine the existence of errors during transmission. The system was verified using the FPV formal technique (Formal property verification), while the fault tolerance was tested using the FSV formal technique (Formal Safety Verification).*

Keywords: *packet processing, formal verification, FPV, FSV, Hamming code, fault tolerance*

1. UVOD

Kompleksni digitalni sistemi prožimaju skoro svaki aspekt modernog života. Ispravno funkcionisanje često se smatra podrazumevanim, iako predstavlja jedan od najtežih zadataka prilikom razvoja i fabrikacije. Posledice postojanja kvarova mogu biti katastrofalne, i zbog toga uspešna verifikacija digitalnih sistema ima ključnu ulogu u proizvodnji.

Tradicionalne metode verifikacije, poput simulacije, uključuju formiranje testbenč okruženja i pisanje specifičnih ili randomizovanih testova, a kasnije temeljnu analizu i popravku uočenih kvarova, kako bi se postigla funkcionalnost sistema. Međutim, ovakav pristup ima puno nedostataka. Pre svega, formiranje kvalitetnog okruženja iziskuje veliku količinu vremena, gde postoji verovatnoća da će se veliki broj kvarova nalaziti baš unutar testbenča. Formalne metode pružaju drugačiji pristup, verifikaciju na osnovu izlaza sistema. To podrazumeva specifikaciju određenih osobina, za koje formalni alat generiše test vektor kako bi dokazao ili opovrgnuo mogućnost sistema da ispuni definisane osobine. Formalni alat matematički analizira potpun prostor svih mogućih simulacionih puteva. Zbog toga, ima sposobnost da dostigne bilo koje dostižno stanje sistema, a zatim generiše dugačku i kompleksnu sekvencu na ulazima sistema, za koju bi inače bili potrebni sati razvoja jednako efikasnog testbenča u verifikaciji pomoću simulacije [1].

Osnovni principi formalne metodologije detaljnije su opisani u sekciji 2. U sekciji 3, predstavljena je specifikacija i struktura implementiranog sistema. Sekcija 4 posvećena je detaljima projektovanja sistema za procesiranje paketa, dok je u sekciji 5 predstavljen postupak formalne verifikacije, kao i još jedne formalne tehnike koja dokazuje otpornost sistema na greške. Zatim, u sekciji 6, iznete su ideje za unapređenje sistema. Zaključno sa sekcijom 7, dat je konačan pregled svega urađenog.

2. FORMALNE METODE

Formalne metode predstavljaju efikasniji i sveobuhvatniji pristup, koji omogućava da se određeni aspekti sistema matematički istraže i dokažu. Neke od glavnih prednosti su [1]:

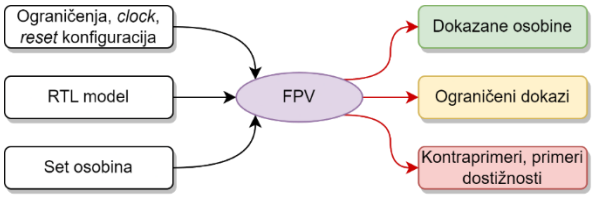
- Formalni alati analiziraju mogući prostor stanja sitema, odnosno uzimaju u obzir sve simulacione puteve odjednom.
- Pružaju potpunu pokrivenost, bez potrebe za kreiranjem kompleksnih verifikacionih okruženja i test vektora.
- Obezbeđuju kratke primere ili kontraprimere ponašanja sistema, koji u velikoj meri olakšavaju proces debagovanja.
- Bilo koje stanje sistema, koje direktno nije zabranjeno ograničenjima, smatra se dostupnim i stoga alat može vrlo lako otkriti interesantne ugaone slučajeve.
- Suštinska razlika u odnosu na simulaciju predstavlja analizu na osnovu internih stanja i izlaza sistema, gde nije potrebno specificirati ulaze kako bi sistem prešao u stanje od interesa. Uz postavljena ograničenja na ulazima alat će istražiti prostor stanja i pronaći najkraći simulacioni put koji pokazuje zahtevano ponašanje ili dokazati da to nije moguće.
- Analiza osobina sistema koje ne moraju biti vremenski ograničene, tj postavljanje pitanja da li je određeno ponašanje moguće bez specificiranja vremenskog ograničenja pojavljivanja.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Vuk Vranjković, vanr. prof.

2.1. FPV TEHNIKA

FPV (eng. *Formal Property Verification*) je tehnika pomoću koje se dokazuje da određeni set osobina (eng. *Property*) važi za neki digitalni sistem. Suštinski to je potpuno drugačiji pristup od simulacije jer omogućava specifikaciju određenih vrednosti na izlazu sistema gde alat praktično trenutno generiše test vektor koji zadovoljava te uslove [1]. Ulazi i izlazi FPV modela prikazani su na slici 1. FPV matematički analizira sve moguće puteve izvršavanja za sistem koji se verifikuje unutar dozvoljenog prostora stanja. U procesu dokazivanja, nad osobinama se izvršavaju verifikacione radnje koje se dele na [1]:

- **tvrdnje** (eng. *assert*) – očekuje se da uvek važi, definišu određena stanja sistema kao nelegalna
- **tačke pokrivenosti** (eng. *cover*) – neophodno je da postoji bar jedan primer, definiše stanje sistema koje mora biti dostižno
- **pretpostavke** (eng. *assume*) – ograničenje koje isključuje određena stanja iz skupa svih dostupnih stanja

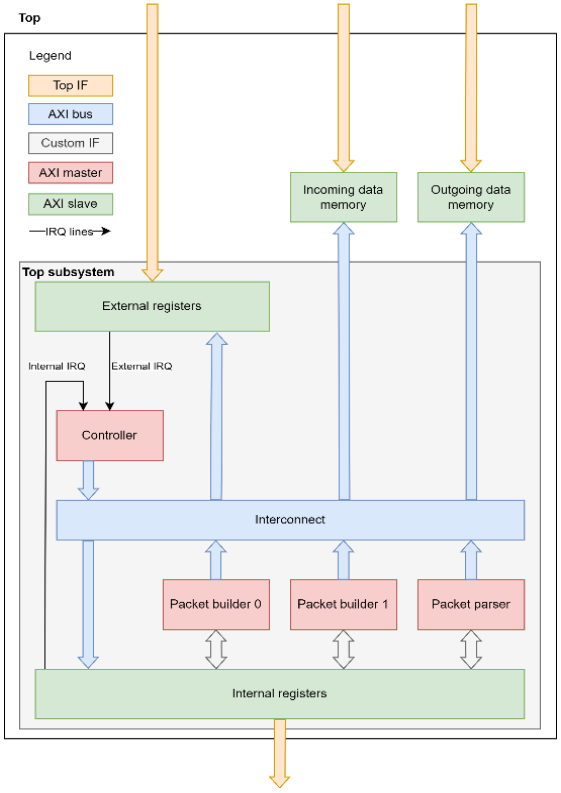


Slika 1. *Ulazi i izlazi FPV alata*

3 SPECIFIKACIJA SISTEMA

Osnovna namena sistema je izgradnja paketa na osnovu podataka koji pristižu u memoriju ili parsiranje paketa kako bi se izvukle potrebne informacije i eventualno detektovale greške tokom prenosa. Komunikacija svih komponenti u sistemu se bazira na AXI4 protokolu. Memorije su standardne komponente sa dva porta, širina podataka je 32 bita. Podržavaju *burst* (prenos podataka u blokovima) upis i čitanje, kao i bajt pristup. Glavni interfejs sistema čine interfejsi prema eksternim i internim (samo za čitanje) registrima i memorijama. Struktura podataka u sistemu prati *little-endian* šemu. Sistem se sastoji iz sledećih blokova (slika 2):

- Kontroler – obavlja konfiguraciju sistema, pokreće obradu paketa. AXI4 *Lite* interfejs.
- AXI4 interkonekt – povezuje memorijski mapirane master i slejv komponente.
- Memorija iz koje se čitaju podaci za obradu. (eng. *Incomming data memory*)
- Memorija u koju se upisuju formirani paketi. (eng. *Outgoing data memory*)
- Dva bloka za izgradnju paketa (eng. *Packet builder*)
- Blok za parsiranje paketa (eng. *Packet parser*)
- Interni registri – sadrže konfiguraciju trenutnog procesa obrade paketa.
- Eksterni registri – sadrže konfiguraciju trenutnog procesa obrade, koju kontroler treba da prepiše u interne registre.



Slika 2. *Struktura sistema*

3.1. MODOVI RADA

Sistem poseduje dva moda rada: izgradnja paketa (PB, eng. *Packet Building*) i parsiranje paketa (PP, eng. *Packet Parsing*). Za svaki od modova postoje eksterni prekidi (slika 2) koji pokreću izgradnju ili parsiranje. Signal *ext_irq[0]* indikuje da su podaci spremni u memoriji za izgradnju paketa, dok signal *ext_irq[1]* indikuje da je paket spreman za parsiranje. Ako se zahtevani zadatak ne može ispuniti, jer su blokovi zauzeti, kontroler šalje signal ka eksternim registrima, koji inkrementuje brojač o otkazanim procesima obrade, i završava zadatak.

Nakon dobijanja zahteva za pokretanjem određenog zadatka kontroler čita konfiguraciju iz eksternih registara, zatim upisuje u interne registre, pokreće zadatak i uklanja eskterni prekid. Odgovarajući blok za obradu započinje svoj proces u skladu sa definisanom konfiguracijom unutar internih registara. Izgrađeni paket se upisuje u memoriju u slučaju procesa za izgradnju paketa, dok u slučaju procesa za parsiranjem, ekstrahovane informacije paketa (podaci iz zaglavlja paketa, ECC i CRC greške) se upisuju u odgovarajuće interne registre. Svaki blok za obradu, na kraju, podiže signal prekida (*int_irq[2:0]* signal), kako bi označio kraj obrade. Kontroler uklanja interni prekid pre pokretanja novog procesa, čime se ciklus rada sistema završava.

3.2. STRUKTURA PAKETA

Svaki paket u sistemu se sastoji iz tri polja (slika 4), zaglavlje paketa, podaci i CRC8 kod. Zaglavlje paketa (slika 3) se sastoji od polja: **SOP** [15:13] – Indikator početka paketa, **P bit** [12:12] – dodatni MSB bit ECC koda, neophodan za detektovanje dvostrukih grešaka, **Tip paketa** [11:8] – može uzeti bilo koju vrednost, **Broj bajtova** [7:4]– broj bajtova podataka u paketu, **ECC** [3:0]

– Hemingov kod računa se na osnovu bita tipa paketa i broja bajtova. Može da detektuje dvostruke i ispravi jednostruke greške.

Slika 3. *Struktura zaglavlja paketa*

Polje	Zaglavlje paketa	Podaci	CRC8
Veličina u bajtovima	2	1-16	1

Slika 4. *Struktura paketa*

3.3. HEMINGOV KOD

U osnovi Hemingov (ECC) kod se sastoji od 4 bita koji se prenose zajedno sa bitima poruke koja sadrži potrebne informacije (broj bajtova u paketu i tip paketa). Svaki bit ECC koda formira se primenom XOR operacije između odgovarajućih bita poruke koji nose potrebne informacije. Da bi se greška detektovala neophodno je formirati korekcioni kod C, na osnovu čije vrednosti se zaključuje koji bit poruke je pogrešan. Shodno tom, pogrešan bit se invertuje, kako bi se dobila ispravna poruka [2].
Za detekciju dvostrukih grešaka neophodno je uvesti dodatan bit P, koji se formira primenom XOR operacije između svih bita koji se nalaze u poruci. U odnosu na kombinaciju vrednosti bita P i korekcionog koda C razlikuju se slučajevi prikazani na slici 5.

C	P	Zaključak
0	0	Nema greške
0	1	Greška se nalazi u P bitu.
≠ 0	1	Greška na jednom bitu.
≠ 0	0	Greška na dva ili više bita.

Slika 5. *ECC greške [2]*

4. IMPLEMENTACIJA SISTEMA

Implementacija sistema podeljena je u tri faze: Faza komunikacije, faza konfiguracije i faza obrade. Prva faza uključuje razvoj interkonekt modula i kontrolera za AXI4 protokol. Ove komponente predstavljaju ključnu ulogu za komunikaciju ostalih blokova u sistemu. Osnovni cilj je implementacija funkcionalnog interkonekta, koji omogućava komunikaciju master-slejv komponenti uz primenu arbitracije po *Round-robin* principu, i master i slejv kontrolera koji poštuju specifikaciju AXI4 standarda.
Druga faza uključuje razvoj glavnog kontrolera u sistemu kao i eksternih i internih registarskih blokova. Ove komponente neophodne su za konfiguraciju blokova za izgradnju i parsiranje paketa i shodno tome osnovni cilj je postići ispravnu konfiguraciju u trenutku kada blokovi započinju zahtevanu obradu podataka.
Treća faza uključuje razvoj blokova za izgradnju i parsiranje paketa. Ove komponente predstavljaju suštinu rada sistema jer proizvode rezultat obrade podataka. Osnovni cilj jeste implementacija blokova koji su sposobni da izgrade paket u skladu sa definisanom konfiguracijom i zaključe postojanje eventualnih kvarova u toku prenosa paketa.

5. VERIFIKACIJA SISTEMA

Verifikacija sistema prati faze razvoja sistema. Pre svega, neophodno je osigurati da komponente mogu međusobno da komuniciraju poštujući AXI protokol. Zatim sledi verifikacija rada kontrolera koji treba da obezbedi ispravnu konfiguraciju komponenti za obradu podataka. Na kraju, potrebno je verifikovati integritet podataka prilikom izgradnje paketa u odnosu na svaki tip operacije, kao i sposobnost sistema za detekciju i korekciju grešaka.
Tokom svake faze primenjen je verifikacioni postupak koji se sastoji iz tri koraka. U prvom koraku neophodno je proveriti da li sistem uopšte ispunjava osnovne funkcionalnosti predviđene specifikacijom dizajna (eng. *Design Exercise*). To se postiže definicijom jednostavnih tački pokrivenosti koje daju izvesnu dozu samopouzdanja da sistem prati glavnu ideju i namenu. Očekivano je da se u ovoj fazi pronađe najveći broj kvarova u sistemu. Kako broj kvarova opada, verifikacija prelazi na drugi korak a to je traženje specijalnih ugaonih slučajeva u kojima se kriju neki neočekivani problemi (eng. *Bug Hunting)*. U ovom trenutku postavljena su stroga ograničenja sa fokusom na specijalne slučajeve. Postepenim popuštanjem ograničenja prelazi se na treći korak, gde formalni alat uzima u obzir potpunu kompleksnost sistema kako bi postigao potpun dokaz (eng. *Full Proof*). Ovo predstavlja vrlo zahtevan zadatak, koji ako se ispuni, daje veliku dozu sigurnosti da sistem radi ispravno [1].

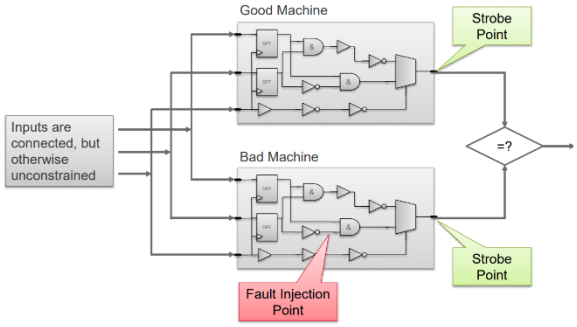
5.1. INTEGRITET PODATAKA

Verifikacija integriteta podataka bazira se na primeni slobodnih promenljivi (eng. Non-Deterministic variables). Vrednosti koje uzima slobodna promenljiva potpuno su randomizovane uz ograničenje da ostaju stabilne za jedan simulacioni put (test slučaj). Kada model sadrži određeni nivo simetrije, ove promenljive mogu značajno povećati efikasnost formalne verifikacije [1].
Strukutra paketa, kao što je već pominjano u specifikaciji, sastoji se od zaglavlja paketa, podataka i CRC koda. Organizacija bajtova u paketu zavisi od broja bajtova podataka i operacije koja se primenjuje prilikom formiranja paketa.
Vrednost slobodne promenljive odnosi se na poziciju određenog bajta ulaznih podataka. U izgrađenom paketu, zbog dodavanja dodatnih bajtova zaglavlja paketa i tipa operacije koji se primenjuje, pozicija selektovanog bajta će imati drugu vrednost. Vrednosti izabranog bajta se registruju na ulazu i izlazu, a zatim porede kako bi se potvrdio integritet. Na ovaj način slobodna promenljiva omogućava generalizaciju formalne analize, jer će svi bajtovi podataka u paketu biti provereni odjednom.

5.2. FSV TEHNIKA

FSV (eng. Formal Safety Verification) aplikacija služi za dokazivanje sigurnosti sistema koji poseduju određenu redundantnu logiku sa ciljem detektovanja i korekcije kvarova. FSV funkcioniše tako što se definišu interni signali sa kvarom i tačke opservacije (eng. *strobe points*) koje mogu biti FO (eng. *functional output*) ili CO (eng. *checker output*) tipa, gde alat generiše tvrdnje u skladu sa definisanim tačkama kako bi dokazao sigurnost ili pronašao kontra-primer. U odnosu na FO sistem se smatra sigurnim ako se greška ne može propagirati do FO, zbog redundantne logike koja to sprečava, ili ako se greška uvek može detektovati na CO. Da bi se to utvrdilo FSV formira formalni model ispravne i mašine sa kvarom (slika 6), gde se porede definisani izlazi kako bi se utvrdila sposobnost sistema za detekciju i propagaciju kvara.

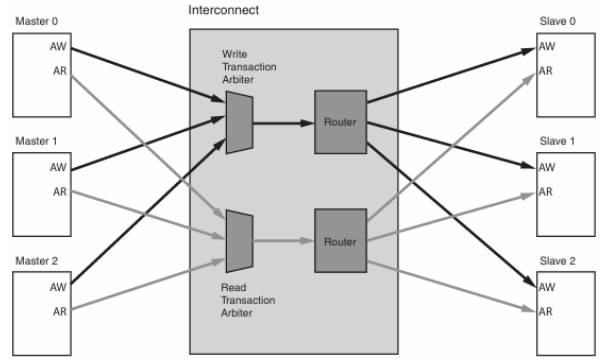
Sistem za procesiranje paketa poseduje logiku u zaglavlju paketa koja služi za detekciju i otklanjanje kvarova koji se odnose na informaciju o tipu i broju bajtova u paketu. ECC kod treba da bude sposoban da detektuje dvostruke i ispravlja jednostruke greške, što je dokazano pomoću FSV aplikacije [4].



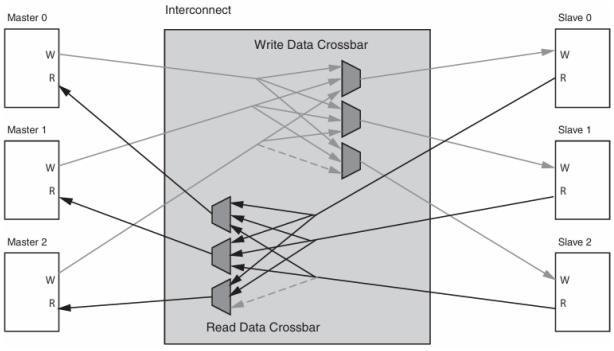
Slika 6. *FSV model [4]*

6. UNAPREĐENJE SISTEMA

Usko grlo u sistemu takođe predstavlja interkonekt koji podržava samo jednu transakciju u jednom trenutku. Deljeni pristup interkonekt (eng. *Shared Access Mode*) modulu, iako zahteva malo hardverskih resursa za implementaciju ne obezbeđuje dobre performanse sistema. Bolje, ali i skuplje, rešenje po pitanju resursa predstavlja interkonekt u *crossbar* režimu (slike 7 i 8). Struktura prati SAMD topologiju (eng. *Shared-Address Multiple-Data*). Nezavisni adresni kanali poseduju zasebne arbitre za transakcije upisa i čitanja. Master koji dobije *grant* može da zahteva određenu liniju za podatke prema slejv modulu. Kanali za prenos podataka organizovani su u *crossbar* topologiji koja omogućava da nezavisni parovi mogu da razmenjuju podatke u paraleli [3].



Slika 7. *Deljeni pristup adresnih kanala za upis i čitanje [3]*



Slika 8. *Crossbar organizacija kanala za podatke [3]*

7. ZAKLJUČAK

Implementacija sistema poseduje sve funkcionalnosti predviđene specifikacijom. Sistem je sposoban da na osnovu eksternog prekida, pokrene različite operacije izgradnje paketa, kao i da detektuje i ispravi eventualne greške nastale u procesu prenosa podataka. Svaka faza implementacije, kao i sistem u celosti, verifikovani su primenom formalne metodologije uz upotrebu efikasnih tehnika. Na kraju, razmatran je pravac unapređenja koji omogućava efikasniju komunikaciju komponentni u sistemu. Na osnovu svega navedenog u radu može se zaključiti da je sistem upešno projektovan i verifikovan primenom isključivo formalnih metoda.

8. LITERATURA

[1] E. Seligman, E. T. Schubert и K. M. V. A. Kiran, Formal verification: An essential toolkit for modern VLSI Design, Elsevier/Morgan Kaufmann, 2015.

[2] N. Raut, „tutorialspoint,“ [На мрежи]. Available: https://www.tutorialspoint.com/hamming-code-for-single-error-correction-double-error-detection. [Последњи приступ 22 1 2024].

[3] „LogiCORE IP AXI Interconnect,“ [На мрежи]. Available: https://www.xilinx.com/support.

[4] „JasperGold Functional Safety Verification,“ Cadence, [На мрежи]. Available: https://support.cadence.com.

Kratka biografija:

Aleksa Đoković rođen je u Prijepolju 1999. god. Diplomirao je na Fakultetu tehničkih nauka u Novom Sadu, na smeru Energetika, elektronika i telekomunikacije 2022. godine. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Energetska, elektronika i telekomunikacije odbranio je 2024. god.
Kontakt: djokovic64@gmail.com