Predlog diplomskog rada

Implementacija podrške za eksterno debagovanje RISC-V mikroprocesora

Implementing external debug support for a RISC-V microprocessor

Student: Lazar Premović 2019/0091
Mentor: prof. dr Zaharije Radivojević

Svrha ovog dokumenta je kratak opis motivacije i pregled rada na visokom nivou, kao i prikazivanje koraka po kojima će implementacija biti vršena (zajedno sa projektnim odlukama i otvorenim pitanjima u vezi sa tim korakom) i na kraju je takođe navedena i literatura korišćena za pripremu ovog dokumenta (i ta koju se očekuje da će ostati primarna literatura za sam rad).

Motivacija i pregled rada

Vremenom softver postaje sve kompleksniji i samim time i podložniji greškama, jedan od najčešćih alata koji programeri koriste pri pronalaženju grešaka je debager koji im omogućava da uvid u stanje programa i kontrolu njegovog izvršavanja. Na današnjim desktop sistemima podršku za debagovanje obezbeđuje operativni sistem, međutim embedded sistemi često imaju vrlo proste operativne sisteme ili ih uopšte nemaju. Zato se kod embedded sistema podrška za debagovanje realizuje u hardveru tako što se programeru obezbeđuje poseban interfejs kojim se ciljni sistem povezuje na softver koji će upravljati debagovanjem a koji se nalazi na njegovom desktop računaru.

Ovaj rad je osmišljen kao nadogradnja na *arilla-vlsi* računarski sistem koji po svojoj prirodi ne izvršava operativni sistem i samim tim njegovo debagovanje do sada je bilo izuzetno komplikovano.

Arilla-vlsi[1] je i sam po sebi nadogradnja *arilla* računarskog sistema realizovana za potrebe projekta na predmetu *Računarski VLSI sistemi*, *arilla-vlsi* proširuje originalni *arilla* sistem dodavanjem dve nove periferije:

- Kontroler PS/2 tastature
- Primopredajnik morzeove azbuke (sa opcionom komponentom audio kodek-a)

Za potrebe demonstracije ovih novih periferija projekat je portovan na *DE1* i *DE10-Standard* FPGA razvojne ploče i izvršavao je prost *chat* program.

Arilla[2] je računarski sistem realizovan za potrebe projekta na predmetu *Arhitektura Računara* i sastoji se od sledećih komponenti:

- **RISC-V** jezgro (**RV32I** set funkcionalnosti, bez podrške za prekide)
- VGA kontroler (800x600@72Hz)
- Kontroler PS/2 Miša
- SRAM memorija sa 8K reči

Sve komponente su povezane prostom sinhronom magistralom sa atomičnim ciklusima.
Sistem je implementiran na *DE0* FPGA razvojnoj ploči i za potrebe demonstracije je izvršavao prost *paint* program.

Rad će biti implementiran u *Verilog/SystemVerilog* jeziku za opis hardvera i pokrenut na *DE-10 Standard* FPGA razvojnoj ploči.

[1] Projekat je realizovan u timu koji su činili: Luka Simić, Aleksa Marković, Aleksandar Ivanović i Lazar Premović.
[2] Projekat je realizovan u timu koji su činili: Luka Simić, Aleksa Marković i Lazar Premović.

Koraci implementacije

1. Implementacija **RISC-V** jezgra
    1. Implementacija **RV32I** osnovnog instrukcijskog seta
    2. Implementacija **M** ekstenzije
    3. Implementacija **Zicsr** ekstenzije
    4. Implementacija **M** moda izvršavanja i pratećeg izvršnog okuženja
2. Implementacija hardverske podrške za eksterno debagovanje
    1. Modifikacija jezgra podrškom za eksterno debagovanje
    2. Implementacija debug modula (**Debug Module**)
    3. Implementacija modula za transport debug naredbi (**Debug Transport Module**)
    4. Implementacija direktnog pristupa magistrali od strane debagera (**Bus Access**)
3. Osmišljavanje softverskog steka za debagovanje i verifikacija kompatibilnosti
4. Implementacija dodatnih periferija za **RISC-V** jezgro
    1. Adaptacija VGA kontrolera iz *arille*
    2. Implementacija kontrolera PS/2 tastature i misa uz pomoć apstraktnog PS/2 kontrolera
    3. Adaptacija primopredajnika morzeove azbuke
    4. Implementacija GPIO kontrolera i kontrolera sedmosegmentnih displeja
5. Implementacija hardvera za generisanje traga izvršavanja
    1. Implementacija interfejsa jezgra ka enkoderu traga izvršavanja
    2. Implementacija enkodera za trag izvršavanja
    3. Implementacija memorije za skladištenje traga izvršavanja
    4. Implementacija interfejsa za prenos traga izvršavanja
    5. Implementacija traga pristupa memoriji
    6. Implementacija instrumentacionog traga
6. Implementacija hardversko-softverskog rešenja za prikupljanje i prikazivanje traga izvršavanja
    1. Implementacija hardvera za prijem traga izvršavanja
    2. Implementacija softvera koji će upravljati hardverom za prijem traga izvršavanja
    3. Implementacija ili upotreba postojećeg softvera za dekompresiju i prikaz traga izvršavanja
    4. Proširenje softvera za debagovanje sa mogućnošću konfiguracije traga izvršavanja

Implementacija RISC-V jezgra

Arhitektura Za osnovni instrukcijski set je izabran **RV32I** (Širina reči od 32bita i 32 registra) jer za edukacionu implementaciju nema benefita od prelaska na širinu reči od 64bita a ušteda FPGA resursa nije zanemarljiva. Od opcionih ekstenzija odabrane su **M** (množenje i deljenje celih brojeva) i **Zicsr** (operacije nad kontrolnim i statusnim registrima). Ekstenzija **M** je odabrana jer ima dobar odnos kompleksnosti implementacije i poboljšanja performansi dok su M mod izvršavanja i podrška za debagovanje uslovljeni postojanjem **Zicsr** ekstenzije. Od privilegovanih modova biće implementiran samo najprivilegovaniji **Machine** mod jer je neophodno implementirati bar neko izvršno okruženje kako bi bilo moguće realizovati sistem prekida (koji značajno poboljšava fleksibilnost samog jezgra a i potencijalno olakšava implementaciju podrške za debagovanje).

Organizacija Odlučeno je da procesor bude implementiran sa izvršavanjem instrukcija u više ciklusa i mikroprogramskom upravljačkom jedinicom, ova implementacija je izabrana jer arhitektura FPGA onemogućava izvršavanje instrukcija u jednom ciklusu (SRAM memorije u FPGA obavezno imaju jedan ciklus kašnjenja) a implementacija protočne obrade bi nepotrebno komplikovala deo sistema koji nije direktan fokus rada. Radni takt procesora će biti fiksnih 50MHz (primarna frekvencija signala takta FPGA ploče) što olakšava kompatibilnost sa VGA adapterom (čija bi promena signala takta bila poprilično komplikovana) a takođe i predstavlja lep kompromis između brzine procesora i protoka potrebnog za trag izvršavanja.

Odlučeno je da se jezgro ponovo implementira jer originalno jezgro iz *arille* nema podršku za prekide i implementirano je logičkim dijagramima umesto jezikom za opis hardvera.

Rezultat Rezultat ovog koraka je funkcionalno **RISC-V** jezgro koje se može testirati izvršavanjem test programa koji pokriva sve instrukcije odabranog instrukcijskog seta.

Implementacija hardverske podrške za eksterno debagovanje

Odlučeno je da implementacija poštuje [4], takođe postoji i novija verzije iste specifikacije nazvana **1.0-STABLE** ali ona još uvek nije ratifikovana i zbog toga je podržana od strane samo jednog proizvođača debug adaptera[3], dok je verzija 0.13 podržana od strane svih značajnih proizvođača debug adaptera[4].

Debug podrška se sastoji od **Debug Transport Module**, modula koji pruža interfejs kojim debug adapter pristupa **Debug Module**-u, Kao preporučeni transportni protokol je izabran **JTAG** koji je IEEE standard originalno namenjen za *boundary scan testing* ali je ubrzo usvojen kao standardni transport za eksterno debagovanje od strane svih značajnih proizvođača mikroprocesora. **Debug Module** je jezgro podrške za eksterno debagovanje i omogućava upravljanje izvršavanjem **RISC-V** jezgra, pristup registrima jezgra kao i opcioni direktan pristup magistrali sistema (koja će biti implementirana u ovoj implementaciji). Debug standard takođe specifikuje i opcioni programski bafer koji debager može da koristi za izvršavanje proizvoljnog koda, kao i opcionu hardversku podršku za tačke prekida na izvršavanje instrukcije ili pristup memoriji. Ove opcione komponente će obe biti realizovane u ovoj implementaciji.

Rezultat Rezultat ovo koraka je kompletna hardverska podrška za eksterno debagovanje **RISC-V** jezgra iz prethodnog koraka, ovo će biti testirano izvršavanjem prostih debug operacija direktnom interakcijom sa **Debug Module**-om kao i kroz **Debug Transport Module** koji implementira **JTAG** standard.

Osmišljavanje softverskog steka za debagovanje i verifikacija kompatibilnosti

Benefit upotrebe ratifikovanog standarda za eksterno debagovanje je postojanje softverske podrške od strane debagera i drajvera za debug adaptere. Tako da će softverski stek za debagovanje koristiti **RISC-V** kompatibilnu verziju **GDB**-a i **RISC-V** kompatibilnu verziju **OpenOCD** drajvera debug adaptera. **OpenOCD** podrzava veliki broj debug adaptera, od njih, meni su dostupni *ST-Link* i *Raspberry Pi Zero W*. Ukoliko fakultet poseduje još neke debug/**JTAG** adaptere podržane od strane **OpenOCD**-a ili sa direktnom podrškom za **RISC-V** od strane proizvođača, smatram da bi bilo korisno demonstrirati funkcionisanje sistema i sa njima.

Rezultat

Rezultat ovog koraka će biti demonstracija debagovanja kompleksnijeg programa koji se izvršava na jezgru korišćenjem alata koje bi programeri i inače koristili za debagovanje drugih embedded sistema sa komforom kompletnog integrisanog razvojnog okruženja kao što je *Eclipse* ili *Visual Studio Code*.

Realizacijom ovog koraka rad je ispunio minimalne zahteve da bi se mogao smatrati uspešnim i realizacija ostalih koraka zavisi od raspoloživog vremena.

Implementacija dodatnih periferija za RISC-V jezgro

Iako ovaj korak nije direktno povezan sa temom samog rada, smatram da bi postojanje ovih periferija značajno obogatilo finalnu demonstraciju celog sistema, omogućavanjem izvršavanja i debagovanja kompleksnih programa nalik na one korišćene za demonstraciju *arilla-vlsi* i *arilla* sistema.

VGA kontroler je potrebno portovati iz logičkih dijagrama u jezik za opis hardvera ali se ne očekuju veće izmene samog dizajna.

Kontroleri PS/2 periferija će biti ponovo implementirani jer originalni kontroler miša ima probleme sa pouzdanošću a kontroler tastature trenutno koristi poprilično naivnu implementaciju PS/2 protokola, kako ovi kontroleri nisu preterano komplikovani i međusobno dele veliki deo implementacije ne očekuje se da će njihova reimplementacija zahtevati previše vremena.

Primopredajnik morzeove azbuke je već realizovan upotrebom jezika za opis hardvera i pokazao se kao poprilično robustan tako da se ne očekuju veće izmene njegove implementacije, osim dodavanja podrške za prekide.

Kontroleri GPIO pinova (koji se takođe može koristiti za upravljanje LED diodama, prekidačima i tasterima na FPGA ploči) i sedmosegmentnih displeja su takođe poprilično prosti za implementaciju

[3]Lauterbach je trenutno jedina kompanija koja podržava 1.0 verziju debug standarda.
[4]Lauterbach, Segger, OpenOCD.

a mogu dodatno da obogate raznovrsnost programa koji se mogu izvršavati na sistemu i samim tim i koristiti za demonstraciju.

Kako originalno jezgro korišćeno u *arilli* nije podržavalo prekide, sve periferije će biti obogaćene podrškom za prekide.

Implementacija hardvera za generisanje traga izvršavanja

Trenutno postoje dva standarda za generisanje traga izvršavanja **E-Trace** [5] i **N-Trace** [6], u ovom trenutku nije odlučeno koji od njih će biti implementiran tako da će ovde biti izneti njihove sličnosti i razlike koje će uticati na konačni izbor standarda za implementaciju.

Sličnosti

- Oba standarda specifikuju sličnu arhitekturu sa interfejsom između jezgra i enkodera, enkoderom traga izvršavanja, formatom poruka koje generiše enkoder i mogućnošću za čuvanje traga u memoriji ili njegovo emitovanje van čipa.
- Oba standarda koriste isti interfejs između jezgra i enkodera specifikovan u [5].
- Oba standarda koriste isti interfejs prema debageru koji se koristi za konfiguraciju [7].
- Oba standarda podržavaju trag izvršavanja instrukcija.
- Za oba standarda postoje implementacije softvera za dekodiranje traga izvršavanja.

Razlike:

- **E-Trace** dodano podržava trag pristupa memoriji.
- **E-Trace** predlaže implementaciju filtera traga izvršavanja.
- **E-Trace** je manje zastupljen u komercijalnim proizvodima te samim tim ima i slabiju softversku podršku.
- **N-Trace** je nastao iz vlasničke implementacije rešenja za generisanje traga izvršavanja i kao takav ima značajniju podršku kompanija, a samim tim i bolju softversku podršku.
- **N-Trace** je baziran na *IEEE Nexus 5001* standardu što dodatno poboljšava softversku podršku.
- Specifikacija **N-Trace**-a je dizajnirana sa transportnim protokolom u vidu, što rezultuje nešto jednostavnijom specifikacijom i potencijalno implementacijom.
- Vlasnička implementacija **N-Trace**-a podržava instrumentacioni trag izvršavanja.
- **N-Trace** za sada ne podržava trag pristupa memoriji.

Na osnovu svega navedenog iznad prednosti **N-Trace**-a su potencijalno nešto lakša implementacija i softverska podrška, dok su prednosti **E-Trace**-a podrška za trag pristupa memoriji.

Kako **IEEE Nexus 5001** (na kojem je **N-Trace** baziran) podržava i trag pristupa memoriji i instrumentacioni trag, moguće implementirati te funkcionalnosti na **IEEE Nexus 5001** kompatibilan način (šta više takva implementacija je i predložena od strane **N-Trace** radne grupe za standardizaciju u sklopu **N-Trace** standarda u nekom kasnijem trenutku).

Implementacija hardversko-softverskog rešenja za prikupljanje i prikazivanje traga izvršavanja

Kako trag izvršavanja može da generiše znatno više podataka nego što su debug/**JTAG** adapteri dizajnirani da prenesu (trenutni predlog implementacije teoretski može da generiše do 800Mbit/s traga izvršavanja), za prenos tragova izvršavanja se koriste posebni adapteri, kako si ti adapteri poprilično skupi za prenos traga će ovom slučaju biti napravljen adapter sposoban da prenese tu količinu podataka (ukoliko fakultet poseduje adapter za trag izvršavanja i to je jedno od potencijalnih rešenja). Zbog potrebe za velikim protokom jedino *USB2.0+* ili *1Gbit Ethernet* dolaze u obzir kao veza između desktop računara i adaptera za trag izvršavanja, što automatski eliminiše mogućnost upotrebe *STM32* baziranih rešenja (ni jedan *STM32* mikrokontroler koji posedujem ne podržava gorenavedene interfejse), ono što je preostalo je upotreba *Raspberry Pi Zero W*-a (korišćenjem *USB2.0* interfejsa) ili upotreba *Hard Processor System Arm* jezgara integrisanih na FPGA ploči (korišćenjem *1Gbit Ethernet* interfejsa). Odluka koje od tih rešenja će biti realizovano još nije doneta. Nakon realizacije hardvera i firmvera, potrebno je realizovati i softver koji će trag izvršavanja preneti na desktop i kojim će se upravljati konfiguracijom sistema za generisanje tragova izvršavanja.

Literatura

[1] Patterson, D. A., & Hennessy, J. L. (2021). Computer organization and design RISC-V edition: The hardware software interface (2nd ed.). Morgan Kaufmann.

Knjiga koja sadrži smernice i primere za implementaciju RISC-V jezgra.

[2] RISC-V Foundation. (2019). The RISC-V Instruction Set Manual, Volume I: User-Level ISA, Document Version 20191213 (A. Waterman & K. Asanović, Eds.).

Specifikacija korisnički vidljivog dela RISC-V arhitekture.

[3] RISC-V International. (2021). The RISC-V Instruction Set Manual, Volume II: Privileged Architecture, Document Version 20211203 (A. Waterman, K. Asanović, & J. Hauser, Eds.).

Specifikacija privilegovanih modova izvršavanja zajedno sa izvršnim okruženjem, od interesa zbog specifikacije izvršnog okruženja (koje obuhvata i specifikaciju obrade prekida) u M modu (M je najprivilegovaniji mod i samim tim jedini mod koji će biti implementiran).

[4] RISC-V Foundation. (2019). RISC-V External Debug Support, Document Version 0.13.2.

Specifikacija interfejsa neophodnih za realizaciju podrške za eksterno debagovanje.

[5] RISC-V International. (2022). Efficient Trace for RISC-V, Document Version 2.0.

Specifikacija interfejsa RISC-V jezgra sa enkoderom traga izvršavanja i formata traga izvršavanja.

[6] RISC-V International. (2023). RISC-V N-Trace (Nexus-based Trace) Specification, Document Version 1.0.0_rc8.

Specifikacija alternativnog formata traga izvršavanja.

[7] RISC-V International. (2023). RISC-V Trace Control Interface Specification, Document Version 1.0.0_rc8.

Specifikacija interfejsa kojim debager upravlja generisanjem traga izvršavanja.

[8] RISC-V International. (2023). RISC-V Trace Connectors Specification, Document Version 1.0.0_rc8.

Specifikacija fizičkih konektora preporučenih za upotrebu pri debagovanju.

[9] Velguenkar, N. (2018). The Design of a Debugger Unit for a RISC Processor Core (Master's thesis). Rochester Institute of Technology.

Master rad na sličnu temu, rad je objavljen pre ratifikacije zvanične specifikacije interfejsa za eksterno debagovanje te softverski stek nije postojao pa nije bilo moguće izvesti neku realističnu demonstraciju. Mogućnosti generisanja traga izvršavanja su takođe poprilično rudimentarne u odnosu na dve specifikacije koje danas postoje.