UNIVERZITET U ZENICI

Politehnički fakultet

Softversko inženjerstvo

Operativni sistemi

Delaida Muminović

ANALIZA PERFORMANSI I VERIFIKACIJA REAL-TIME OPERATIVNOG SISTEMA NA RISC-V ARHITEKTURI

Diplomski rad - radna verzija 1

Mentor: Van. Prof. dr. Denis Čeke

Zenica, 2025.

Sadržaj

1. Uvod

Ugradbeni sistemi (embedded systems) predstavljaju ključnu komponentu moderne tehnologije, nalazeći primjenu u širokom spektru uređaja, od kućanskih aparata i automobilske industrije do industrijske automatizacije i medicinske opreme. Zajednička karakteristika mnogih od ovih sistema je potreba za predvidljivim i pravovremenim izvršavanjem zadataka, što ih svrstava u kategoriju sistema realnog vremena (real-time systems). U ovakvim sistemima, ispravnost rezultata ne zavisi samo od logičke korektnosti, već i od toga da li je rezultat dostavljen unutar strogo definisanog vremenskog roka.

Za efikasno upravljanje zadacima, resursima i interakcijama u kompleksnim ugradbenim sistemima često se koriste Real-Time Operativni Sistemi (RTOS). RTOS pruža mehanizme za multitasking, komunikaciju i sinhronizaciju zadataka, te upravljanje vremenom na način koji omogućava viši stepen predvidljivosti u poređenju sa jednostavnijim bare-metal pristupom ili opštim operativnim sistemima.

Posljednjih godina primjećuje se rastući trend u razvoju hardvera i softvera otvorenog koda u domenu ugradbenih sistema. RISC-V arhitektura se ističe kao otvorena i modularna arhitektura skupa instrukcija (ISA) koja dobija sve veću popularnost, nudeći fleksibilnost i prilagodljivost za različite aplikacije. Uporedo s tim, FreeRTOS, kao popularan RTOS otvorenog koda, često se bira za implementaciju u resursno ograničenim ugradbenim sistemima. Kombinacija RISC-V arhitekture i FreeRTOS-a otvara nove mogućnosti, ali istovremeno postavlja pitanja o performansama i predvidljivosti RTOS-a na ovoj novijoj hardverskoj platformi.

Cilj ovog diplomskog rada je da se provede analiza performansi i, gdje je moguće, izvrši verifikacija determinističkog ponašanja real-time operativnog sistema (konkretno, FreeRTOS kernela) na RISC-V arhitekturi. Rad će koristiti specifičnu implementaciju RISC-V soft-core procesora (PicoRV32) na FPGA platformi (TangNano 9K) kao ciljni hardver. Analiza će biti provedena kroz praktičnu implementaciju i eksperimentalnu evaluaciju, uz osvrt na primjenjivost formalnih metoda i matematičkih modela za procjenu performansi i predvidljivosti.

Predmet ovog rada je studija slučaja ponašanja FreeRTOS-a na navedenoj RISC-V platformi, sa fokusom na ključne mehanizme kernela kao što su upravljanje zadacima, međuprocesna komunikacija, sinhronizacija i upravljanje vremenom. Kao aplikativna osnova za

eksperimentalnu evaluaciju koristit će se portana verzija jednostavne aplikacije (Tetris igre) na RTOS taskove.

Metod rada obuhvata teoretsko proučavanje RTOS koncepata i relevantnih matematičkih modela, praktičnu implementaciju (portanje FreeRTOS-a, razvoj aplikacije na RTOS-u, implementacija potrebnih drivera), eksperimentalno testiranje na stvarnom hardveru i analizu prikupljenih podataka.

Kroz ovaj rad nastojat će se pružiti uvid u praktične aspekte rada sa RTOS-om na RISC-V arhitekturi, te procijeniti u kojoj mjeri se teoretski modeli predvidljivosti mogu primijeniti i verificirati na ovakvom sistemu.

2. Teorijski okvir

2.1. Ugradbeni sistemi i sistemi realnog vremena

Ugradbeni sistem se definiše kao računarski sistem dizajniran za obavljanje jedne ili više specifičnih funkcija, često u okviru većeg mehaničkog ili elektronskog sistema. Ključne karakteristike uključuju ograničenja u pogledu resursa (procesorska snaga, memorija, potrošnja energije) i, za značajan dio ugradbenih sistema, zahtjeve u pogledu vremena izvršavanja [1].

Sistemi realnog vremena su tip ugradbenih sistema gdje ispravnost rada ne zavisi samo od logički tačnog rezultata računarskog procesa, već i od trenutka kada se taj rezultat isporuči. Razlikuju se hard real-time sistemi (gdje propuštanje vremenskog roka dovodi do katastrofalnog kvara sistema) i soft real-time sistemi (gdje propuštanje roka dovodi do degradacije performansi, ali ne i kvara) [2]. Za postizanje predvidljivosti u ovim sistemima neophodno je precizno upravljanje izvršavanjem zadataka i interakcijom sa hardverom.

2.2. Real-Time Operativni Sistemi (RTOS)

Real-Time Operativni Sistem (RTOS) je operativni sistem dizajniran da podrži aplikacije sa zahtjevima realnog vremena. Za razliku od opštih operativnih sistema (kao što su Windows ili Linux), RTOS-i se fokusiraju na determinističko ponašanje – garantuju da će se određeni zadaci izvršiti unutar predvidljivih i definisanih vremenskih rokova [2].

Ključna komponenta RTOS-a je njegov kernel, koji obavlja osnovne funkcije:

- Upravljanje zadacima (Task Management): Kreiranje, raspoređivanje (scheduling), brisanje i upravljanje stanjima zadataka (running, ready, blocked, suspended). RTOS koristi algoritme raspoređivanja u realnom vremenu (npr. Fixed-Priority Scheduling, Earliest Deadline First) kako bi osigurao da zadaci sa višim prioritetom ili bližim rokovima dobiju procesorsko vrijeme kada im je potrebno. FreeRTOS primarno koristi preemptivni scheduling sa fiksnim prioritetima [3].
- Međuprocesna komunikacija (IPC - Inter-Process Communication): Mehanizmi koji omogućavaju zadacima da razmjenjuju podatke. U FreeRTOS-u to uključuje Queue-ove, Stream i Message Buffere, te Direktne Notifikacije Taskovima [3].
- Sinhronizacija: Mehanizmi za koordinaciju pristupa dijeljenim resursima ili signalizaciju događaja između zadataka. FreeRTOS pruža Semafore (binarne i

brojačke) i Mutexe (sa podrškom za prioritetno nasljeđivanje radi izbjegavanja problema prioritetne inverzije) [3].

- Upravljanje Vremenom: Korištenje hardverskog tajmera za generisanje periodičnih RTOS tickova, implementacija funkcija za odgođeno izvršavanje zadataka (`vTaskDelay`) i Softverskih Tajmera za zakazane pozive korisničkih funkcija [3].
- Upravljanje Memorijom: Strategije alokacije memorije za stekove zadataka, heap i druge potrebe kernela i aplikacije [3].

RTOS kernel komunicira direktno sa hardverom (često nazivano "bare-metal") putem specifičnog portabilnog sloja (portable layer) i drivera za periferije.

2.3. RISC-V Arhitektura

RISC-V je otvorena, besplatna (open-source) arhitektura skupa instrukcija (ISA) razvijena na Univerzitetu Berkeley [4]. Njena modularnost i jednostavnost čine je atraktivnom za ugradbene sisteme. Osnovni 32-bitni skup instrukcija je RV32I, koji se može proširiti standardnim ekstenzijama kao što su M (množenje/dijeljenje), A (atomske operacije), C (kompresija instrukcija), itd. [4].

Za rad RTOS-a na RISC-V arhitekturi ključni su Control and Status Registers (CSR) i funkcionalnost prekida (često implementirana kao CLINT - Core Local Interruptor). Specifični CSR registri (`mtime`, `mtimecmp`, `mstatus`, `mie`, `mip`, `mcause`, `mepc`) su neophodni za upravljanje tajmerima i prekidima od strane RTOS kernela [5].

2.4. Hardverska platforma: PicoRV32 na TangNano 9K FPGA

Kao hardverska platforma za ovaj rad koristi se PicoRV32, jednostavna open-source implementacija 32-bitne RISC-V procesorske jezgre, implementirana na TangNano 9K FPGA ploči. Za razliku od fiksnih mikrokontrolera, na FPGA čipu se hardver (uključujući i PicoRV32 procesor i potrebne periferije) programira koristeći hardverske opisne jezike poput Veriloga.

Sistem implementiran na FPGA za potrebe ovog projekta uključuje PicoRV32 jezgru, memoriju (RAM/ROM), tajmer (baziran na CLINT-u), UART interfejs za komunikaciju i debugging, te GPIO i prilagođenu logiku za interakciju sa tasterima i displayom Tetris igre.

2.5. Alati za razvoj

Razvojni proces uključuje korištenje GCC (GNU Compiler Collection) toolchaina za RISC-V arhitekturu za kompajliranje C i asemblerskog koda. Make se koristi za automatizaciju build procesa (kompajliranje, asembliranje, linkovanje). Linker script definiše raspored koda i podataka u memoriji ciljnog hardvera. FPGA alati (npr. Gowin EDA) koriste se za sintezu hardverskog dizajna (Verilog koda) i generisanje bitstream fajla za programiranje FPGA čipa.

3. Analiza dosadašnjih istraživanja

U ovoj cjelini će se predstaviti pregled relevantne literature i postojećih radova koji se odnose na temu diplomskog rada.

Proučavani izvori obuhvataju fundamentalnu literaturu o sistemima realnog vremena i teoriji schedulinga [2], kao i specifičnu dokumentaciju i knjige o FreeRTOS-u i njegovom korištenju u ugradbenim sistemima [1, 3, 4].

Značajan dio dosadašnjih istraživanja odnosi se na implementaciju i portanje FreeRTOS-a na različite procesorske arhitekture, dok je posebna pažnja posvećena istraživanju radova koji se bave implementacijom FreeRTOS-a na RISC-V arhitekturi.

Istraživanja su također obuhvatila materijale o metodama analize performansi ugradbenih sistema i sistema realnog vremena, uključujući koncept Worst-Case Execution Time (WCET) i primjenu teorijskih modela poput Liu & Layland modela za analizu raspoređivanja zadataka i teorije redova čekanja za modeliranje komunikacije.

Na osnovu pregledane literature, može se konstatovati da, iako postoje standardni portovi FreeRTOS-a za RISC-V i primjeri implementacije (često na simulatorima ili specifičnim development kitovima), detaljna eksperimentalna analiza performansi ključnih FreeRTOS mehanizama (kao što su latencije prebacivanja taskova, kašnjenja u IPC mehanizmima, ili uticaj obrade prekida) na konkretnoj, resursno ograničenoj soft core implementaciji na FPGA (poput PicoRV32 na TangNano 9K), uz primjenu metodologije koja povezuje praktična mjerenja sa konceptima teoretskih modela predvidljivosti, nije široko dokumentovana. Ovaj rad nastoji popuniti tu prazninu pružajući i praktičan i analitički doprinos.

4. Skica metodologije

Metodologija rada obuhvata kombinaciju praktične implementacije softverskog sistema i eksperimentalne evaluacije njegovih performansi, uz analizu prikupljenih rezultata u kontekstu teoretskih modela predvidljivosti sistema realnog vremena.

4.1. Hardverska platforma i softverski stek

Rad se zasniva na hardverskoj platformi koju čini PicoRV32 procesor implementiran kao soft core na TangNano 9K FPGA ploči. Hardverski dizajn sistema (uključujući PicoRV32 jezgru, memorijsku mapu, tajmere, UART i periferije za Tetris) je već definisan u Verilog kodu.

Na ovu hardversku platformu će se postaviti softverski stek koji se sastoji od sljedećih slojeva:

- Hardverski sloj: PicoRV32 jezgra i periferije na FPGA.
- Low-level sloj: CRT startup kod (`crt0.S`) za inicijalizaciju sistema pri bootu, i portable layer FreeRTOS-a specifičan za RISC-V/GCC (u `FreeRTOS/Source/portable/GCC/RISC-V`), koji upravlja prebacivanjem konteksta, prekidima i tickovima na najnižem nivou.
- RTOS Kernel i Driveri: FreeRTOS kernel (generički C kod u `FreeRTOS/Source`), implementacija upravljanja memorijom (npr. `heap_1.c`) i driveri za periferije specifične za platformu i aplikaciju (UART, GPIO za tastere, display logika) koji komuniciraju direktno sa hardverskim registrima periferija.
- Aplikativni sloj: Glavna aplikacija (adaptirana Tetris igra) portana na set RTOS taskova.

4.2. Portanje i konfiguracija FreeRTOS-a

Proces portanja obuhvata integraciju standardnog FreeRTOS izvornog koda u postojeći projekat. Ključni koraci uključuju:

- Konfiguracija kernela: Prilagođavanje fajla `FreeRTOSConfig.h` prema specifikacijama hardvera (frekvencija tajmera, adrese tajmerskih registara, veličine memorije, broj prioriteta, omogućavanje potrebnih featurea poput task notifikacija, mutexa, softverskih tajmera).
- Konfiguracija specifična za RISC-V: Podešavanje fajla `freertos_risc_v_chip_specific_extensions.h` i, ako je potrebno, prilagodba koda u portable

layeru, mada se očekuje da će standardni port za RISC-V biti funkcionalan uz ispravnu konfiguraciju.

- Prilagodba startup koda i linker scripta: Integracija poziva za pokretanje FreeRTOS schedulera u `crt0.S` i prilagođavanje linker scripta (`.ld` fajl) kako bi se pravilno rasporedili sekcije koda i podataka FreeRTOS-a i aplikacije u memorijsku mapu hardvera.
- Postavljanje build sistema: Modifikacija Makefile-a kako bi uključio sve FreeRTOS, portable, driver i aplikativne izvorne fajlove u proces kompajliranja i linkovanja koristeći RISC-V GCC toolchain.

4.3. Razvoj aplikacije na RTOS-u (Portanje Tetris Logike)

Bare-metal logika Tetris igre će biti portana na RTOS taskove:

- Identifikacija taskova: Podjela logike na zasebne RTOS taskove (npr. Task_Input, Task_GameLogic, Task_Drawing, Task_FallingTimer).
- Implementacija taskova: Pisanje C koda za funkciju svakog taska, koristeći FreeRTOS API unutar beskonačne petlje taska.
- Međuprocesna komunikacija (IPC): Implementacija razmjene podataka između taskova koristeći Queue-ove (npr. slanje komandi s tastera od Task_Input do Task_GameLogic) i Direktne Notifikacije Taskovima (npr. Task_GameLogic signalizira Task_Drawing da se tabla promijenila, ili ISR tajmera signalizira Task_GameLogic da spusti figuru).
- Sinhronizacija: Korištenje Mutexa za siguran pristup dijeljenim resursima kao što je struktura podataka koja predstavlja stanje Tetris table, kako bi se spriječili konflikti kada joj pristupaju Task_GameLogic i Task_Drawing.
- Upravljanje Tajmingom: Korištenje `vTaskDelay` za jednostavna kašnjenja i Softverskih Tajmera za periodične akcije (npr. zakazivanje spuštanja figure).
- Integracija sa Driverima i Prekidima: Adaptacija drivera za periferije da rade u RTOS okruženju i implementacija User ISR funkcija za obradu hardverskih prekida (tastera, tajmera), koristeći `FromISR` verzije RTOS API funkcija za komunikaciju sa taskovima iz prekidnih rutina. Implementacija potrebnih Hook funkcija (Idle, Tick, greške).

4.4. Eksperimentalna evaluacija

Performanse implementiranog sistema će se mjeriti na TangNano 9K hardveru. Planirani eksperimenti uključuju:

- Mjerenje latencije odziva: Vrijeme od hardverskog događaja (pritisak tastera, timer prekid) do izvršavanja odgovarajućeg dijela koda u tasku.
- Mjerenje vremena izvršavanja taskova: Procjena stvarnog vremena izvršavanja ključnih segmenata koda unutar taskova.
- Mjerenje kašnjenja u IPC mehanizmima: Procjena vremena potrebnog za slanje/primanje podataka kroz Queue-ove ili notifikacije.
- Analiza prebacivanja taskova: Mjerenje režijskog vremena (overhead) koje RTOS kernel troši na prebacivanje s jednog taska na drugi.
- Praćenje korištenja CPU-a: Procjena koliko vremena CPU provodi izvršavajući Idle task (indikator neiskorištenog CPU vremena).
- Testiranje pod opterećenjem: Evaluacija ponašanja sistema kada se brzina igre povećava ili se simuliraju česti događaji tastera/prekida. Korištene tehnike mjerenja uključivat će softverske pristupe (npr. RTOS tracing ako je dostupan i omogućen, korištenje GPIO pinova za signalizaciju početka/kraja sekcija koda i mjerenje eksternim instrumentima, ispis vremenskih oznaka preko UART-a).

4.5. Matematička analiza

Prikupljeni eksperimentalni podaci će se analizirati u kontekstu teoretskih modela predvidljivosti sistema realnog vremena:

- WCET (Worst-Case Execution Time) analiza: Diskutovaće se o faktorima koji utiču na najgore vrijeme izvršavanja ključnih dijelova Tetris taskova (npr. funkcija za provjeru kolizija, funkcija za spuštanje figure). Pokušaće se dati procjene WCET-a na osnovu mjerenja i analize koda. [7].
- Liu & Layland model: Primijeniće se osnovni koncepti ovog modela za diskusiju o raspoređivanju taskova sa fiksnim prioritetima (Rate Monotonic Scheduling, kojeg FreeRTOS primarno koristi). Analiziraće se iskoristivost CPU-a od strane vašeg skupa Tetris taskova i diskutovati da li je (teoretski, pod idealnim uslovima) skup schedulabilan na jednoj jezgri na osnovu njihove iskoristivosti i periodičnosti (gdje je primjenjivo). [8].

- Teorija redova čekanja (Queuing theory): Koncepti ove teorije će se koristiti za modeliranje ponašanja RTOS Queue-ova ili Message Buffera koji se koriste za IPC. Analiziraće se faktori koji utiču na vrijeme čekanja poruka u redu (npr. frekvencija slanja, kapacitet reda, prioriteti taskova). [9].

Ova kombinacija praktične implementacije, eksperimentalne evaluacije i analize kroz prizmu teoretskih modela čini srž metodologije ovog diplomskog rada.

5. Očekivani doprinos

Očekivani rezultati i doprinosi ovog diplomskog rada su višestruki, obuhvatajući kako praktične, tako i teoretske aspekte:

- Praktični doprinos:
    - Uspješno portanje i konfiguracija FreeRTOS kernela za specifičnu RISC-V soft core platformu (PicoRV32 na TangNano 9K FPGA).
    - Razvoj funkcionalne aplikacije (Tetris igre) portane na FreeRTOS taskove, demonstrirajući upotrebu ključnih RTOS featurea (tasking, IPC, sinhronizacija, tajming, prekidi) u realnom (iako jednostavnom) scenariju.
    - Implementacija potrebnih drivera i low-level integracije softvera i hardvera na specifičnoj FPGA platformi.
- Eksperimentalni doprinos:
    - Prikupljanje konkretnih podataka o performansama FreeRTOS mehanizama na ciljnoj hardverskoj platformi (npr. latencija odziva, vrijeme prebacivanja taskova, performanse IPC-a) kroz sistematična eksperimentalna mjerenja.
    - Analiza uticaja različitih faktora (npr. brzina igre, konfiguracija RTOS-a) na performanse i ponašanje sistema.
- Teoretski doprinos:
    - Primjena i diskusija o relevanciji i izazovima korištenja formalnih metoda (WCET) i matematičkih modela (Liu & Layland, Teorija redova čekanja) za analizu predvidljivosti RTOS-baziranog sistema na RISC-V hardveru, na osnovu praktičnog iskustva i eksperimentalnih rezultata.
    - Pružanje uvida u predvidljivost i ponašanje FreeRTOS-a na resursno ograničenom RISC-V soft core-u, doprinoseći razumijevanju performansi RTOS-a na novijim, otvorenim hardverskim arhitekturama.

Ovaj rad će objediniti praktičnu implementaciju kompleksnog softverskog sistema (RTOS + aplikacija) na prilagođenom hardveru sa analitičkim pristupom procjene njegovih karakteristika u realnom vremenu, povezujući teoriju i praksu u oblasti ugradbenih sistema.

6. Literatura

[1] J. J. Labrosse, *MicroC/OS-II: The Real-Time Kernel*, CMP Books, 2002.

[2] G. C. Buttazzo, *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*, 3rd ed., Springer, 2011.

[3] R. Barry, *Mastering the FreeRTOS Real-Time Kernel*, Real Time Engineers Ltd, 2021.

[4] D. Patterson and A. Waterman, *The RISC-V Reader: An Open Architecture Atlas*, Strawberry Canyon LLC, 2017.

[5] RISC-V Foundation, *The RISC-V Instruction Set Manual, Volume II: Privileged Architecture, Version 1.10*, 2019. [Online]. Dostupno: https://riscv.org/technical/specifications/

[6] FreeRTOS, "FreeRTOS Official Documentation," [Online]. Dostupno: https://www.freertos.org/

[7] C. Ferdinand and R. Wilhelm, "The Worst-Case Execution Time Problem—Overview of Methods and Survey of Tools," *ACM Transactions on Embedded Computing Systems*, vol. 7, no. 3, pp. 1–53, 2008. [Online]. Dostupno: https://dl.acm.org/doi/10.1145/1347375.1347389

[8] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment," *Journal of the ACM*, vol. 20, no. 1, pp. 46–61, 1973. [Online]. Dostupno: https://dl.acm.org/doi/10.1145/321738.321743

[9] N. Chong and B. Jacobs, "Formally Verifying FreeRTOS' Interprocess Communication Mechanism," *Proceedings of the Workshop on Embedded and Cyber-Physical Systems Education (EWC)*, 2021. [Online]. Dostupno: https://nchong.github.io/papers/ewc21.pdf