

Анализа перформанси *Zephyr* и *FreeRTOS* оперативних система на *RISC-V* архитектури процесора

Performance Analysis of Zephyr and FreeRTOS Operating Systems on RISC-V Processor Architecture

Милица Ђорђевић, Факултет техничких наука, Нови Сад

**Студијски програм – ЕНЕРГЕТИКА,
ЕЛЕКТРОНИКА И ТЕЛЕКОМУНИКАЦИЈЕ**

Кратак садржај - У раду су приказани анализа и поређење перформанси *Zephyr* и *FreeRTOS* оперативних система на *RISC-V* софтверском процесору конфигурираном на *Xilinx Artix-7 FPGA* платформи. Помоћу осмишљених софтверских алгоритама мерена су, а затим и поређена, трајања извршавања критичних делова оперативних система у процесорским циклусима. Резултати су прокоментарисани и упоређени са очекиваним теоријским вредностима.

Кључне речи: оперативни системи, *RISC-V* процесорска архитектура, *FPGA* софтверски процесор, софтверски алгоритми, поређење перформанси

Abstract - This paper presents an analysis and comparative evaluation of the performance of the *Zephyr* and *FreeRTOS* operating systems on a *RISC-V* software processor implemented on a *Xilinx Artix-7 FPGA* platform. By using specifically designed software algorithms, the execution times of critical operating system parts were measured in processor cycles and subsequently compared. The obtained results were analyzed and discussed in relation to the expected theoretical values.

Keywords: operating systems, *RISC-V* processor architecture, *FPGA* software processor, software algorithms, performance measurement

НАПОМЕНА: Овај рад проистекао је из мастер рада чији ментор је био проф. др Владимир Рајс

1. УВОД

Уграђени, односно ембедед системи су компјутерски системи дизајнирани за специфичну намену, веома често ограничених ресурса и велике брзине одзива. Поред хардверских делова система као што су процесорска јединица, меморија, улазно- излазни подсистем, периферије и напајање, други једнако битан део је софтвер. Развојем технологија за израду

електронских компоненти, уграђени системи су постали хардверски комплекснији, уз високу компактност и енергетску ефикасност. Задаци које ови системи извршавају и захтеви које при томе треба да испуне су се временом проширили, што је у многоме узроковало и усложњавање софтвера. Од простих програма писаних у бесконачној петљи стигло се до потребе за интеграцијом оперативних система у софтвер уграђених система.

Главни критеријум за одабир оперативног система за ембедед уређај је жељена функционалност и брзина одзива, тј. да ли је уређај предвиђен да ради у реалном времену. Уколико јесте, користе се такозвани оперативни системи за рад у реалном времену (енгл. *RTOS- Real Time Operating System*).

Приликом одабира *RTOS* система за ембедед уређај, потребно је обратити пажњу на његову поузданост, подршку за дати хардвер, потребне ресурсе за извршавање и чување кода, као и перформансе. У данашње време постоји више бесплатних оперативних система за рад у реалном времену прилагођених уграђеним системима, од којих су два најпознатија *FreeRTOS* и *Zephyr*. Овај рад се бави анализом и поређењем перформанси ова два оперативна система на *RISC-V* архитектури процесора.

На почетку рада је описан хардвер коришћеног развојног система као и конфигурација *RISC-V* софтверског процесора. Описани су и одабрани софтверски алати коришћени при развоју система, те начини за инструментацију кода и отклањање грешака. Дат је преглед начина рада и поделе оперативних система, са посебним освртом на системе за рад у реалном времену и алгоритме распоређивања који се у њиховој имплементацији користе. Потом су анализирани структуре и начини функционисања поређених (*FreeRTOS* и *Zephyr*) *RTOS* система са нагласком на специфичностима приликом њихове интеграције у пројекат. Дато је поређење ова два система са становишта корисника и комплексности интеграције. Описани су софтверски алгоритми коришћени при мерењу перформанси система, као и разлози за одабир одређених софтверских решења. На крају рада су табеларно приказани добијени резултати, прокоментарисани и упоређени са очекиваним теоријским вредностима.

Такође је вршено и поређење са резултатима пронађеним за другу хардверску платформу. Овај рад има за циљ да укаже на битне практичне ставке приликом одабира одговарајућег оперативног система за уграђен систем, како са становишта корисника тако и са становишта перформанси, као и да прикаже практично поређење два тренутно најпознатија *RTOS* система за ову намену, *Zephyr*-а и *FreeRTOS*-а.

2. КОРИШЋЕНИ ХАРДВЕР

2.1. *Arty A7-100T* развојна плоча

Током израде овог рада, коришћена је развојна плоча *Digilent Arty A7100T*, која у себи садржи програмабилни део хардвера, *FPGA Artix-7* компаније *Xilinx*. Специјално је дизајнирана са намером да буде коришћена као систем са програмабилним софтверским процесором *RISC-V* архитектуре. Плоча поседује порт за *UART* комуникацију који дели са *USB-JTAG*-ом, који је кориштен приликом развоја система. На располагању су и групе прекидача, светлосних диода и улазно/излазних конектора. Разлог за одабир овог развојног система лежи у једноставном повезивању са рачунаром, великој подршци софтверских алата и најбитније, могућношћу програмирања и модификовања жељеног софтверског процесора.

2.2. *Xilinx Artix-7 FPGA*

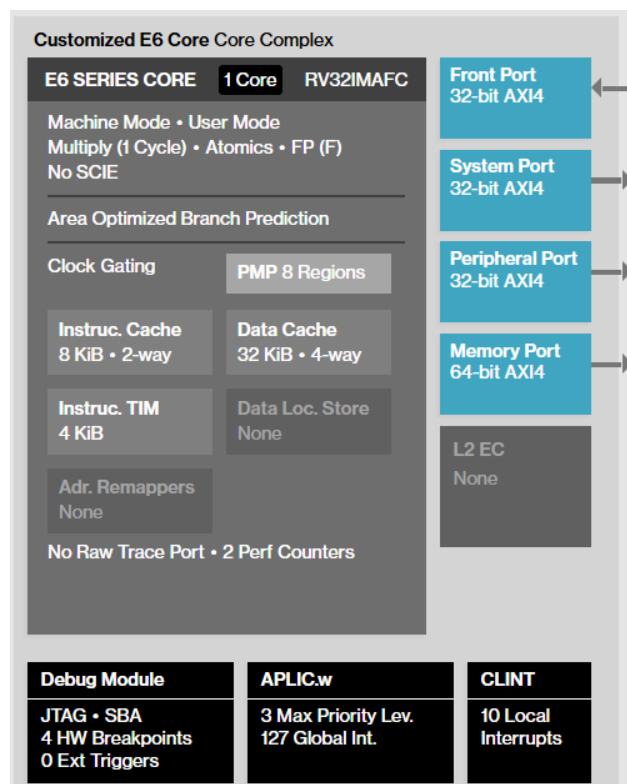
Поред конфигурабилних блокова, основног програмабилног дела, *Artix-7* поседује интегрисану брзу блок *RAM* меморију на чипу. Сваки блок има капацитет од *36Kb* са подршком за двоструко улазни приступ меморији. На чипу се налази интегрисан хардверски блок за конфигурацију, управљање и генерисање сигнала такта. Доступан је и улазно-излазни подсистем који је груписан у банке, као и *PCI* интерфејс. Чип има могућност конфигурисања сигурносне аутентификације и енкрипције *FPGA* конфигурације која се на њега програмира.

2.3. Конфигурација *RISC-V* софтверског процесора на *FPGA*

Овај рад користи *Artix-7* као конфигурисани 32-битни софтверски процесор *RISC-V* архитектуре. Ову архитектуру карактериш висока скалабилност и модуларност.

Софтверски процесор је дизајниран конфигурацијом *E6* плоче компаније *Sifive* помоћу алата *Sifive Core Designer*. Комплетна конфигурација процесорског језгра приказана је на слици 1.

Сама *FPGA* конфигурација чипа се чува у екстерној флеш меморији у облику *MCS* датотеке. Остале меморије су повезане са процесорским језгром путем *TileLink* магистрале. Меморија унутар *FPGA* је реализована као низ меморијских *BRAM* (енгл. *Block RAM*) блокова. Конфигурацијом ових блокова добијена је *SRAM* меморија од *128KB*. Систем садржи и емулирану *ROM* меморију величине *256KB*, која је реализована као *FPGA BRAM*, другим речима њен садржај се губи престанком напајања, али приступи и брзине су моделоване према реалној *ROM* меморији.



Слика 1. Конфигурација процесорског језгра

Од брзих меморија, процесор у својој конфигурацији поседује инструкциони кеш, кеш за податке као и интегрисану брзу инструкциону меморију од *4KB*. *ITIM* меморија је корисна приликом интеграције оперативних система, јер за разлику од кеша на који корисник не утиче директно, у *ITIM* је могуће сместити временски критичан део кода и на тај начин убрзати систем. У оквиру конфигурације процесорског језгра додат је *PMP* уређај, који представља механизам за контролу приступа меморијским регијама, посебно важан у имплементацији корисничког простора оперативног система. Конфигурисани систем подржава прекиде кроз два модула: *CLINT* и *APLIC*. Периферни контролери као што су *GPIO*, *SPI* и *UART* повезују се на процесорско језгро преко стандардизованих системских магистрала. Приликом мерења пребацивања контекста конфигурисани су *GPIO* прекиди, а за иницијалну инструментацију система *UART*. За потребе развоја и анализе система, плоча садржи модул за инструментацију кода са подршком за *JTAG* интерфејс. Сви наведени делови су интегрисани у коначан *FPGA* дизајн и спуштени на систем.

3. ПОВЕЗИВАЊЕ И ПРОГРАМИРАЊЕ ЦЕЛОГ СИСТЕМА

Систем је повезан са рачунаром путем два улаза. Један је преко *USB* интерфејса типа *A*, који осим напајања система од *5V*, поседује контролер који омогућава и контролише комуникацију преко *UART*-а и *JTAG* програмирање *FPGA* слике на *Artix-7*.

У сврху спуштања и инструментације кода, систем је повезан на други улаз рачунара преко *Olimex ARM-*

USB-Tiny-H хардверског модула који представља интерфејс између рачунара и *FPGA* дела плоче, са програматором базираном на *FTDI* чипу, који обезбеђује конверзију између *USB* протокола и *JTAG/SWD* сигнала. Овај улаз служи за програмирање и инструментацију кода на самом софтверском процесору.

3.1. Коришћени софтверски алати за отклањање грешака у систему

У процесу развоја за потребе анализе система и отклањања грешака коришћени су *OpenOCD* и *GDB*. Ови софтверски алати представљају стандардне и веома популарне механизме за отклањање грешака у уграђеним системима, омогућавајући повезивање персоналног рачунара са плочом и комуникацију између њих.

OpenOCD је алат намењен инструментализацији кода и система преко *JTAG/SWD* интерфејса. Његова основна улога је да преведе команде примљене са рачунара у сигнале разумљиве хардверу и на тај начин омогући комуникацију са плочом.

GDB је званични алат за отклањање грешака *GNU* пројекта, који не може директно да управља хардвером, већ је првенствено намењен за рад у симулацијама или на програмима који се извршавају у корисничком простору.

OpenOCD функционише као сервер: прима команде од *GDB*-а преко *TCP/IP* конекције и преводи их у *JTAG/SWD* сигнале за процесор. У комбинацији са *GDB*-ом, *OpenOCD* добија виши степен апстракције: *GDB* омогућава рад са симболима, функцијама и променљивим у програму.

Хардверски специфични делови система и иницијализација комуникације се налазе у *OpenOCD* конфигурационој датотеци.

4. ОПЕРАТИВНИ СИСТЕМИ

Оперативни систем је по дефиницији слој софтвера који управља ресурсима рачунара за своје кориснике и њихове апликације. На основу сврхе, величине (односно комплексности) и захтева за брзином одзива, оперативни системи се класификују у више категорија. Тако постоје оперативни системи опште намене, који су намењени широком спектру апликација и корисничких захтева, затим ембедед оперативни системи оптимизовани за рад на системима са ограниченим ресурсима, као и оперативни системи за рад у реалном времену (*RTOS*) који гарантују одзив у веома уском, ограниченом временском интервалу. Фокус рада је управо на анализи последње врсте система.

4.1. Оперативни системи за рад у реалном времену

Оперативни системи за рад у реалном времену су системи дизајнирани да контролишу извршавање временски критичних задатака, гарантујући предвидивост и стабилност у раду. Ови системи се најчешће користе при имплементацији софтвера за управљачке јединице у аутомобилској индустрији,

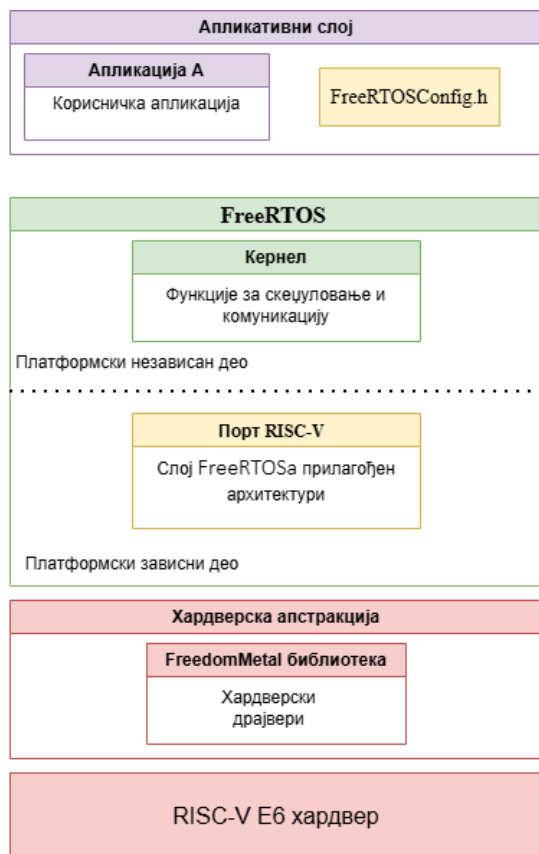
авио индустрији, роботизи, медицинским уређајима и свим осталим ембедед системима у којима кашњења и грешке у извршавању могу имати значајне последице. На основу брзине одзива *RTOS* системи се деле на *Soft RTOS*-е код којих мања кашњења одзива немају веће последице на систем и *Hard RTOS*-е код којих минимална кашњења могу довести до отказа система. У зависности од конфигурације, *FreeRTOS* и *Zephyr* је могуће користити и као *Hard* и као *Soft RTOS*-е. Друга подела *RTOS* система заснива се на начину на који се врши пребацивање извршавања (контекста) са једног задатка на други. У случају кооперативних система покретање наредног задатка врше унутар задатка који се тренутно извршава, те је сваки задатак одговоран да довољно често позове функцију распоређивања како не би дошло до блокирања система. Код преемптивних система покретање наредног задатка врше се форсирано, позивом функције за распоређивање од стране оперативног система, изван функције задатка који се тренутно извршава. У оквиру овог рада, мерена су трајања пребацивања контекста у оба случаја, тј. када су системи конфигурисани и као преемптивни и као кооперативни.

4.2. FreeRTOS оперативни систем

FreeRTOS је бесплатан, минималистички оперативни систем под *MIT* лиценцом, за чију је основну функционалност потребно укључити само три основне датотеке кернела. У пројекат се интегрише на исти начин као библиотека, а не дефинише систем изградње и развојно окружење. Током израде овог рада *FreeRTOS* је интегрисан у неколико корака. Прво је преузет изворни код са *Git* репозиторијума, затим је одабрано развојно окружење *Freedom Studio* компаније *SiFive* прилагођено раду на њиховим плочама, а затим је интегрисана хардверска библиотека произвођача. Систем изградње је дефинисан преко система *Make* у оквиру развојног окружења, као и систем за инструментацију кода. Хардверски део је зависан од библиотеке произвођача, те саме имплементације зависи сложеност укључивања модула. Дијаграм основних слојева *FreeRTOS*-а неопходних за интеграцију у пројекат приказан је на слици 2.

4.3. Zephyr оперативни систем

Zephyr оперативни систем за рад у реалном времену, намењен првенствено ембедед уређајима, представља један од тренутно најразвијенијих *RTOS* система те сврхе, познат по својој високој модуларности и великом броју подсистема. *Zephyr* је имплементиран као комплетна платформа са сопственом структуром у коју корисник може да интегрише своју апликацију. Подсистеме је могуће једноставно укључити *Kconfig* системом. За апстракцију хардвера користи принцип хардверског стабла, тзв. *Devicetree*. Хардверско стабло је организовано као хирерархија чворова у оквиру које сваки чвор представља један хардверски елемент.



Слика 2. Основни слојеви FreeRTOS-a

Хардверски елемент може бити уређај (*UART* периферија), подсистем (*GPIO* контролер) или елемент (*LED*). Сваки чвор садржи атрибуте који дефинишу параметре тог хардверског дела, као што су адреса у меморијском простору, број прекида, конфигурација пинова или брзина комуникације. Оваква структура омогућава јасно раздвајање хардверског описа од изворног кода. За потребе овог рада, генерисано је хардверско стабло помоћу раније поменутог алата *Sifive Core Designer*. За разлику од *FreeRTOS-a*, где није постојао стандардизован систем за изградњу, *Zephyr* систем је заснован на *CMake-y*. На слици 3 је приказана структура слојева *Zephyr* оперативног система.

4.4. Поређење FreeRTOS и Zephyr оперативних система са становишта корисника

FreeRTOS је дуго година био најпопуларнији *RTOS* систем за микроконтролере, док је *Zephyr* новије решење које га је временом сустигло у популарности. Једна од кључних предности *FreeRTOS-a* је његова једноставна интеграција у већ постојећи пројекат. *Zephyr* је сам по себи екосистем, долази са својим дефинисаним окружењем за развој и изградњу у које је потребно уклопити структуру пројекта. Иницијална интеграција *Zephyr-a* је комплекснија, али је систем знатно модуларнији и конфигурабилнији, те подржава развој исте апликације за више хардверских платформи истовремено специфицирањем плоче приликом покретања изградње. *Zephyr* у оквиру *West* алата нуди подршку за генерацију извештаја о потрошњи *RAM* и *ROM* меморије након изградње. Још једна битна разлика је имплементација корисничког

простора, који је у *Zephyr-y* потпуно одвојен од кернелског и не захтева хардверски систем заштите меморије, док *FreeRTOS* подржава корнички простор само у оквиру привилегованих задатака када се интегрише хардверска заштита меморије. Са друге стране, кориснички простор *Zephyr-a* додаје време односно оптерећење при сваком системском позиву, те успорава рад система.



Слика 3. Основни слојеви Zephyr-a

У случају мање захтевних пројеката и хардвера веома ограничених ресурса боље је користити *FreeRTOS*, а у случају комплекснијих система или озбиљнијих пројеката које је потребно прилагођавати различитим платформама и одржавати, *Zephyr* је боље решење.

4.5. Поређење перформанси FreeRTOS и Zephyr оперативних система

Приликом мерења перформанси и поређења *RTOS* система, фокус је на мерењу временског одзива и додатног оптерећења процесора које оперативни систем уноси својом интеграцијом. Значајан критеријум при одређивању квалитета система је и потребна радна меморија за извршавање и меморија за складиштење кода. Сва временска мерења су вршена у процесорским циклусима како би резултати били независни од фреквенције рада те упоредиви са резултатима других платформи.

4.6. Пребацивање контекста између задатака

За мерење додатног оптерећења које оперативни систем уноси при пребацивању контекста са једног на други задатак, кориштени су алгоритми засновани на

контроли тока извршавања задатака бинарним семафорима. Разлог за одабир овог приступа је могућност примене истог механизма на сва вршена мерења, те могућност поређења резултата.

У случају добровољног пребацивања контекста са једног задатка на други, задаци су истог приоритета, а кориштен је „пинг-понг“ алгоритам где се задаци наизменично блокирају и препуштају извршавање другом инкрементовањем бинарног семафора. У овом случају, *Zephyr* уноси мање оптерећење око трећину. У случају принудног пребацивања контекста, где задатак нижег приоритета периодично одблокира извршавање задатка вишег приоритета семафором, резултати су приближно једнаки.

Ако се упореде додатна оптерећења између ових тестова, у оквиру истог система, може се приметити да је у случају *Zephyr-a* добровољно пребацивање контекста брже од принудног, а код *FreeRTOS-a* је обрнуто. На резултате мерења утиче имплементација како распоређивача тако и функција бинарних семафора на оперативном систему.

4.7. Пребацивање контекста између задатка и прекидне рутине

За мерење додатог оптерећења при пребацивању контекста са задатка на прекидну рутину, из задатка је мењано стање излазног *GPIO* пина, који је кратко спојен на други улазни *GPIO* пин и на тај начин је у петљи контролисано изазиван прекид. Добијени су слични резултати додатог оптерећења за оба оперативна система. Ово мерење не укључује синхронизационе објекте, те је једино оптерећење које је додато у систем осим промене контекста извршавање функције за промену стања пина, што не зависи од имплементације оперативног система. У случају додатног оптерећења при пребацивању контекста са прекидне рутине на задатак, кориштен је прекид на тастеру који затим одблокира задатак семафором. У овом случају је *Zephyr* скоро два пута бржи.

4.8. Анализа потребне меморије за код

Што се тиче потребне меморије за складиштење кода, *FreeRTOS* заузима 4.1 kB за кернел, док *Zephyr* узима 6,7kB. Али, укупна меморија потребна да се интегрише оперативни систем обухвата и хардверски део кода, који је у случају *Zephyr-a* доста модуларан и могу се избацити сви некористени модули, те је укупно минимално заузеће меморије у случају ове платформе за *Zephyr* 12kB, а за *FreeRTOS* 22kB.

5. ЗАКЉУЧАК

Приликом анализе добијених резултата, утврђено је да *Zephyr* има више детерминистичко понашање у односу на *FreeRTOS*, односно да су времена потребна за промене контекста мање зависна од конкретних услова у систему и разлога пребацивања. Рад је показао да је *Zephyr* боље користити у већим системима који захтевају одржавање и стабилност, а *FreeRTOS* за прототипе и системе са мање ригорозним захтевима.

6. ЛИТЕРАТУРА

- [1] Факултет техничких наука, Нови Сад, „Развој софтвера за дигиталне микроконтролере“, наставни материјал. (приступљено- март 2026.)
- [2] Mouser, „Arty A7-100T Development Board - Digilent – Datasheet“ [Arty A7-100T Development Board - Digilent | Mouser](https://www.mouser.com/pdfdocs/Arty%20A7-100T%20Development%20Board%20-%20Digilent.pdf) (приступљено - април 2026.)
- [3] FPGAKey, „XC7A100T-2FGG484I – Datasheet“. https://www.fpgakey.com/xilinx-parts/xc7a100t-2fgg484i?srsId=AfmBOorm5mtVZb24aNd1EStN4mF_puN2YSOJB-lsNBccLrBa6mD_k2aI (приступљено - мај 2026.)
- [6] A. Elias, „*Zephyr RTOS Embedded C Programming*“, New York, 2024.
- [7] R. Johnson, „*Real-Time Applications with FreeRTOS: Definitive Reference for Developers and Engineers*“, New York, 2025.
- [8] E. White, „*Making Embedded Systems*“, Sebastopol, CA, 2011.
- [9] Званични сајт са пројектном документацијом и описом рада *FreeRTOS* оперативног система <https://www.freertos.org/> (приступљено - април 2026.)

Кратка биографија:

Милица Ђорђевић је рођена у Сремској Митровици 1997. год. Дипломирала на Факултету техничких наука у Новом Саду. Мастер рад на Факултету техничких наука из области Електротехнике и рачунарства – Примењена електроника одбранила је 2026.год.

Контакт:
micadjordjevic6@gmail.com