



**УНИВЕРЗИТЕТ У НОВОМ САДУ  
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**



**УНИВЕРЗИТЕТ У НОВОМ САДУ  
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА  
НОВИ САД**

**Департаман за рачунарство и аутоматику**

**Одсек за рачунарску технику и рачунарске комуникације**

## **ЗАВРШНИ (BACHELOR) РАД**

**Кандидат:** Јоксимовић Богдан

**Број индекса:** RA-171/2017

**Тема рада:** Реализација *SPI* руковоаоца за контролу *FLASH* меморије на *NXP i.MX 8M Mini* платформи

**Ментор рада:** Доц. др. Ковачевић Јелена

Нови Сад, јул, 2021



КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, <b>РБР:</b>	
Идентификациони број, <b>ИБР:</b>	
Тип документације, <b>ТД:</b>	Монографска документација
Тип записа, <b>ТЗ:</b>	Текстуални штампани материјал
Врста рада, <b>ВР:</b>	Завршни (Bachelor) рад
Аутор, <b>АУ:</b>	<b>Јоксимовић Богдан</b>
Ментор, <b>МН:</b>	<b>Доц. Др. Ковачевић Јелена</b>
Наслов рада, <b>НР:</b>	<b>Реализација SPI руковаоца за контролу FLASH меморије на NXP i.MX 8M Mini платформи</b>
Језик публикације, <b>ЈП:</b>	Српски / латиница
Језик извода, <b>ЈИ:</b>	Српски
Земља публикавања, <b>ЗП:</b>	Република Србија
Уже географско подручје, <b>УГП:</b>	Војводина
Година, <b>ГО:</b>	<b>2021.</b>
Издавач, <b>ИЗ:</b>	Ауторски репринт
Место и адреса, <b>МА:</b>	Нови Сад; трг Доситеја Обрадовића 6
Физички опис рада, <b>ФО:</b> (поглавља/страна/ цитата/табела/слика/графика/прилога)	
Научна област, <b>НО:</b>	Електротехника и рачунарство
Научна дисциплина, <b>НД:</b>	Рачунарска техника
Предметна одредница/Кључне речи, <b>ПО:</b>	
<b>УДК</b>	
Чува се, <b>ЧУ:</b>	У библиотеци Факултета техничких наука, Нови Сад
Важна напомена, <b>ВН:</b>	
Извод, <b>ИЗ:</b>	Zadatak ovog rada bio je da se prikaže rešenje programske podrške rukovaoca SPI FLASH memorije u okviru sistema baziranog na NXP i.MX 8M Mini platformi. Prikazan je princip rada SPI FLASH memorije u okviru sistema potrošačke elektronike. Primenjeno rešenje izloženo je po slojevima podeljenim po nivoima apstrakcije. Predstavljen je način provere validnosti rešenja.
Датум прихватања теме, <b>ДП:</b>	
Датум одбране, <b>ДО:</b>	
Чланови комисије, <b>КО:</b>	Председник:
	Члан:
	Члан, ментор:
	Потпис ментора



UNIVERSITY OF NOVI SAD • FACULTY OF TECHNICAL SCIENCES  
21000 NOVI SAD, Trg Dositeja Obradovića 6

## KEY WORDS DOCUMENTATION

Accession number, <b>ANO</b> :	
Identification number, <b>INO</b> :	
Document type, <b>DT</b> :	Monographic publication
Type of record, <b>TR</b> :	Textual printed material
Contents code, <b>CC</b> :	Bachelor Thesis
Author, <b>AU</b> :	<b>Joksimović Bogdan</b>
Mentor, <b>MN</b> :	<b>Ph.D Kovačević Jelena</b>
Title, <b>TI</b> :	<b>Realization of SPI driver for controlling the FLASH memory on NXP i.MX 8M Mini platform</b>
Language of text, <b>LT</b> :	Serbian
Language of abstract, <b>LA</b> :	Serbian
Country of publication, <b>CP</b> :	Republic of Serbia
Locality of publication, <b>LP</b> :	Vojvodina
Publication year, <b>PY</b> :	
Publisher, <b>PB</b> :	Author's reprint
Publication place, <b>PP</b> :	Novi Sad, Dositeja Obradovica sq. 6
Physical description, <b>PD</b> : (chapters/pages/ref./tables/pictures/graphs/appendixes)	
Scientific field, <b>SF</b> :	Electrical Engineering
Scientific discipline, <b>SD</b> :	Computer Engineering, Engineering of Computer Based Systems
Subject/Key words, <b>S/KW</b> :	
<b>UC</b>	
Holding data, <b>HD</b> :	The Library of Faculty of Technical Sciences, Novi Sad, Serbia
Note, <b>N</b> :	
Abstract, <b>AB</b> :	The aim of this paper was to develop software for the SPI FLASH memory driver for a system based on NXP i.MX 8M Mini platform. This paper shows basic usage of SPI FLASH memory in consumer electronics. Code is presented by layers of abstraction. Way of validation of the software solution is also presented.
Accepted by the Scientific Board on, <b>ASB</b> :	
Defended on, <b>DE</b> :	
Defended Board, <b>DB</b> :	President:
	Member:
	Member, Mentor:
	Mentor's sign

## **Zahvalnost**

Zahvaljujem se institutu „RT-RK“ na stručnoj pomoći pri izradi završnog rada.



# УНИВЕРЗИТЕТ У НОВОМ САДУ ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА



## SADRŽAJ

1. Uvod .....	1
2. Namenski sistemi u okviru potrošačke elektronike .....	3
2.1 Trajna memorija u okviru namenskih sistema .....	4
2.2 FLASH memorija.....	7
2.3 SPI sprega .....	8
3. Analiza pristupa korišćenja SPI FLASH memorije na Cirrus Logic platofrmi.....	10
3.1 Inicijalizacija DSP-a .....	11
3.2 Uloga SPI FLASH memorije nakon inicijalizacije.....	12
4. Realizacija programske podrške za SPI FLASH rukovalac .....	16
4.1 Opis fizičke arhitekture na kojoj je izvršena implementacija.....	16
4.2 Programsko rešenje .....	18
4.2.1 Rukovalac SPI FLASH periferije .....	18
4.2.2 Aplikacija za prikaz rada .....	23
5. Verifikacija tačnosti rešenja .....	25
6. Zaključak .....	28
7. Literatura .....	30

## SPISAK SLIKA

<i>Slika 2-1 Uprošćena blok šema sistema prijemnika audio i video sadržaja [6] .....</i>	<i>4</i>
<i>2-2 Poređenje trajnih memorija zasnovanih na poluprovodničkim tehnologijama [10] .</i>	<i>6</i>
<i>Slika 2-3 Razlika u izradi NOR i NAND FLASH memorije [13] .....</i>	<i>7</i>
<i>Slika 2-4 Primer povezivanja jednog vodećeg SPI uređaja sa tri prateća [6] .....</i>	<i>8</i>
<i>Slika 2-5 Vremenski dijagram polariteta i faze takta [6] .....</i>	<i>9</i>
<i>Slika 3-1 Način rada DSP-a u master boot modu [15] .....</i>	<i>10</i>
<i>Slika 3-2 Algoritam inicijalizacije putem SPI FLASH memorije [15] .....</i>	<i>12</i>
<i>Slika 4-1 Blok dijagram i.MX 8M Mini sistema [1], crvenim kvadratima su označene celine korišćene u okviru razvijene programske podrške .....</i>	<i>17</i>
<i>Slika 5-1 PowerShell komandna linija i grafičko korisničko okruženje koji kreira python skripta za testiranje rukovaoca .....</i>	<i>25</i>
<i>Slika 5-2 Izgled grafika u okviru Trace32 programske podrške .....</i>	<i>26</i>
<i>Slika 5-3 RTAG fizička arhitektura .....</i>	<i>27</i>
<i>Slika 5-4 Programska podrška Data Center .....</i>	<i>27</i>

**SPISAK TABELA**

<i>Tabela 2-1 Uticaj CPOL i CPHA na logiku prenosa podataka</i>	9
<i>Tabela 3-1 Komandna reč za poruke pisanja ka DSP-u</i>	13
<i>Tabela 3-2 Opcije za bite 23:22 u komandnoj reči pisanja ka DSP-u</i>	13
<i>Tabela 3-3 Komandna reč za poruke zatraženog čitanja</i>	14
<i>Tabela 3-4 Komandna reč za odgovor na poruku zatraženog čitanja</i>	14
<i>Tabela 3-5 Reč za prenos podataka zatraženog čitanja</i>	14
<i>Tabela 3-6 Komandna reč za odgovor na poruku nezatraženog čitanja</i>	14
<i>Tabela 3-7 Reč za prenos podataka nezatraženog čitanja</i>	15
<i>Tabela 3-8 Komandna reč za odgovor na poruku nezatraženog čitanja</i>	15
<i>Tabela 3-9 Reč za prenos podataka nezatraženog čitanja</i>	15
<i>Tabela 4-1 LUT, struktura sekvence i struktura instrukcije</i>	21

## SKRAĆENICE

**SPI** – eng. *Serial Peripheral Interface*, protokol za serijsku komunikaciju

**AVR** – eng. *Audio-Video Receiver*, prijemnik audio i video sadržaja

**DSP** – eng. *Digital Signal Processing*, digitalna obrada signala

**SoC** – eng. *System on Chip*, sistem na čipu

**DVD** – eng. *Digital Video Disc*, digitalni video disk

**ADC** – eng. *Analog - Digital converter*, Analogno-digitalna konverzija

**DAC** – eng. *Digital-Analog converter*, Digitalno-analogna konverzija

**ROM** – eng. *Read only memory*, memorija iz koje se mogu samo čitati podaci

**PROM** – eng. *Programmable Read only memory*, ROM memorija koja se jednom može programirati

**EPROM** – eng. *Erasable Programmable Read-Only Memory*, ROM memorija koja se može obrisati i više puta programirati

**EEPROM** – eng. *Electrically Erasable Programmable Read-Only Memory*, ROM memorija koja se može elektronski obrisati i više puta programirati

**I2C** – eng. *Inter-Integrated Circuit*, protokol za serijsku komunikaciju

**FGMOS** – eng. *floating-gate MOSFET*, tehnologija izrade tranzistora

**NOR** – eng. *Negative OR*, negirano ili

**NAND** – eng. *Negative AND*, negirano i

**XIP** – eng. *eXecute In Place*, izvršavanje programske podrške pročitane direktno sa trajne memorije

**SCLK** – eng. *Serial Clock*, Serijski takt

**MOSI** – eng. *Master Output Slave Input*, Izlaz iz vodećeg/Ulaz u podređenog

**MISO** – eng. *Master Input Slave Output*, Vodeći ulaz, prateći izlaz

**SS** – eng. *Slave Select*, linija za odabir pratećeg uređaja

**CPOL** – eng. *Clock Polarity*, Polaritet takta

**CPHA** – eng. *Clock Phase*, Faza takta

**UART** – eng. *Universal Asynchronous Receiver/Transmitter*, protokol za asinhronu komunikaciju

**SAI** – eng. *Synchronous Audio Interface*, sprega za sinhroni prenos audio podataka

**LUT** – eng. *Look Up Table*, tabela sa sekvencama operacija koje formiraju željenu instrukciju

**RTAG** – eng. *RT-Audio Grabber*, fizička arhitektura za reprodukciju i snimanje audio signala

## 1. Uvod

Moderne tehnologije za reprodukovanje zvuka bazirane su na kompleksnoj digitalnoj obradi signala sa ciljem povećanja kvaliteta i pružanja osećaja prostornog zvuka. Za izvršavanje takve obrade u realnom vremenu neophodni su namenski sistemi kreirani za tu specifičnu svrhu. Primer jednog takvog namenskog sistema je sistem za obradu zvuka u okviru prijemnika audio i video sadržaja *eng. Audio Video Receiver - AVR*.

AVR je sastavni deo kućnog bioskopa. AVR ima ulogu prijema i obrade audio i video signala iz različitih izvora. Nakon prijema audio signala namenski sistem zadužen za obradu digitalnog signala vrši obradu signala na načine definisane u tehnologijama koje AVR podržava. Sistemi za obradu signala u okviru AVR uređaja mogu biti bazirani na procesorima za digitalnu obradu signala *eng. Digital signal processor – DSP* ili sistemima na čipu *eng. System on Chip – SoC*. Programaska podrška i konfiguracione informacije potrebne za pravilno izvršavanje te obrade se nalaze na trajnoj memoriji. U okviru namenskih sistema trajna memorija treba biti fizički kompaktna i niske potrošnje električne energije. Mogućnost pisanja i brisanja podataka trajne memorije za vreme rada sistema je takođe velika prednost. Karakteristike FLASH memorije je čine odličnim izborom za trajnu memoriju. Da bi se smanjio broj vodova potrebnih za povezivanje FLASH memorije sa ostatkom sistema koristi se serijska komunikacija putem SPI sprege.

U radu je realizovana programaska podrška sa svrhom upravljanja FLASH memorijom povezanom putem *eng. Serial Peripheral Interface - SPI* sprege na ploči baziranoj na i.MX 8M Mini fizičkoj arhitekturi. Fizička arhitektura i.MX 8M Mini zasnovana je na procesorima izgrađenim u ARM arhitekturi [1]. Prednost korišćenja ARM i.MX 8M fizičke arhitekture je pojednostavljenje razvoja i smanjenje cene u poređenju sa rešenjima zasnovanim na DSP-u

---

[2]. Očekivani rezultat je rukovalac koji je sposoban da izvrši operacije serijske FLASH memorije, a to su čitanje, pisanje i brisanje podataka.

U drugom poglavlju rada iznete su osnove o namenskim sistemima za obradu zvuka, trajnoj memoriji u okviru njih kao i spregama kojim se trajna memorija vezuje za ostatak sistema.

U trećem poglavlju rada prikazan je princip korišćenja SPI FLASH memorije u okviru jednog realnog rešenja baziranog na Cirrus Logic platformi.

U četvrtom poglavlju rada prikazana je platforma na kojoj je izrađen praktičan deo rada kao i sam praktičan deo.

U petom poglavlju rada iznet je način na koji je verifikovana funkcionalnost rešenja i automatizacija procesa verifikacije.

U šestom poglavlju rada dat je zaključak ovog rada.

U sedmom poglavlju data je literatura korišćena u radu.

## 2. Namenski sistemi u okviru potrošačke elektronike

Namenski sistemi su skup fizičke arhitekture i programske podrške projektovane tako da vrše neku namensku funkciju. U nekim slučajevima namenski sistemi su skup većeg sistema ili proizvoda [3].

Računarski sistem treba da radi dovoljno dobro, da bude što jeftiniji i da troši što manje električne energije [4]. Prednost namenskih sistema je to što njihova namenska funkcija pruža dobre granice koje definišu šta predstavlja dovoljno dobar rad sistema. Sa dobro definisanim granicama performansi dizajneri sistema mogu odabrati odgovarajuće fizičke komponente sistema koje ispunjavaju ekonomske zahteve.

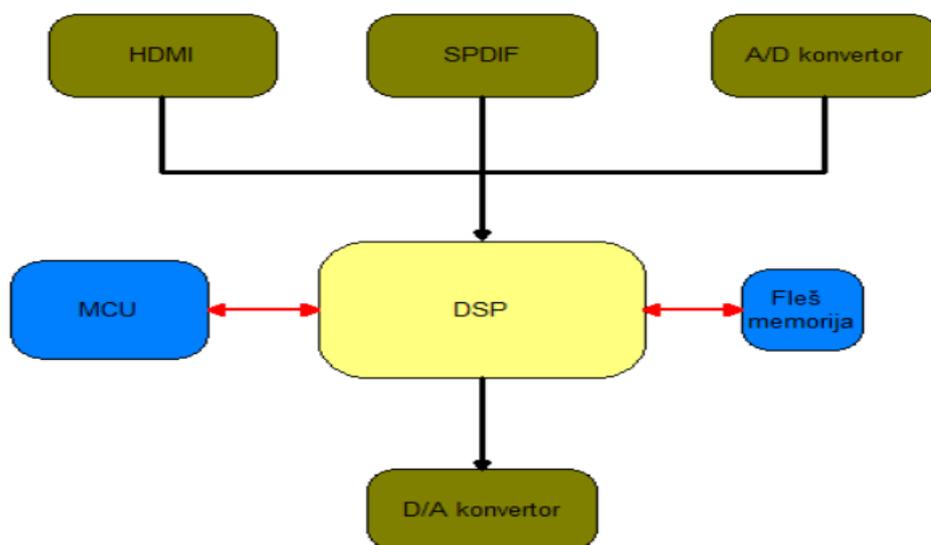
Namenski sistemi imaju široku primenu od potrošačke elektronike do raznih industrijskih i naučnih primena. Neki primeri namenskih sistema su ABS kočnice i sistemi za aktivaciju vazdušnih jastuka u automobilima, uređaji korišćeni za dijagnostiku pacijenta u medicini poput uređaja za merenje pritiska, elektrokardiograma i magnetne rezonantne tomografije i uređaji u okviru interneta stvari koji se mogu koristiti za prikupljanje podataka korišćenih u meteorologiji i seizmografiji. U okviru potrošačke elektronike namenski sistemi se mogu pronaći u jednostavnim sistemima kao što su digitalni časovnici ali i u kompleksnim sistemima kao na primer konzole za video igre koje se po procesorskoj snazi mogu porediti sa personalnim računarima [5].

Još jedan primer namenskog sistema u potrošačkoj elektronici je prijemnik audio i video sadržaja *eng. Audio Video Receiver - AVR*. Osnovna svrha AVR-a je prijem audio i video signala sa izvora kao što su TV, satelitski i radio prijemnici, DVD i Blue-Ray prijemnici,

konzole za video igre i sl. AVR pruža mogućnost obrade različitih vrsta digitalnih i analognih audio signala dok uporedno pušta video sadržaj na odgovarajući izlaz [6].

Pojednostavljena struktura AVR uređaja baziranog na procesoru za obradu digitalnog signala *eng. Digital signal processor – DSP*:

- Ulaz - digitalni ili analogni koji se prosleđuje na ADC konvertor
- Obrada - DSP - procesor koji vrši glavnu obradu
- Kontrola sistema - Mikrokontroler zadužen da kontroliše ponašanje DSP-a i ostatka sistema
- Izlaz - analogni nakon DAC konvertora



Slika 2-1 Uprošćena blok šema sistema prijemnika audio i video sadržaja[6]

Kao što je spomenuto obradu izvršava sistem za obradu zvuka. U slučaju da je sistem baziran na DSP-u on je taj koji vrši obradu. Programska podrška za obradu zvuka i konfiguracione informacije potrebne za izvršavanje obrade se nalaze na trajnoj memoriji kojoj pristupa sistem za obradu zvuka.

## 2.1 Trajna memorija u okviru namenskih sistema

U trajnu memoriju spadaju tipovi memorije koji čuvaju podatke upisane u njih i nakon isključenja izvora napajanja. Za trajnu memoriju koja čuva podatke koji se mogu promeniti u toku izvršenja, na primer konfiguracione informacije koje korisnik može promeniti, potrebno

je da ima mogućnosti brisanja i upisa podataka pored podrazumevane mogućnosti čitanja podataka.

U okviru računarskih sistema postoje različite tehnologije trajnog čuvanja podataka. U modernim računarskim sistemima trajna memorija se najčešće zasniva na dve tehnologije: magnetni diskovi i poluprovodničke memorije. Prednost memorija koje koriste magnetne diskove je niža cena prema kapacitetu memorije dok poluprovodničke memorije imaju manju potrošnju električne energije i veću brzinu [7]. Poluprovodničke memorije ne koriste pokretne delove za razliku od memorija zasnovanih na magnetnim diskovima. Ova karakteristika omogućava otpornost na mehaničku traumu kao i mogućnost proizvodnje na malim fizičkim veličinama. Karakteristike trajne memorije od značaja za namenske sisteme su fizička veličina i potrošnja električne energije [8]. Poluprovodnička memorija je bolji izbor za trajnu memoriju u okviru namenskog sistema. U okviru poluprovodničke memorije imamo više tipova memorije kao što su ROM – *eng. Read Only Memory*, PROM – *eng. Programmable ROM*, EPROM – *eng. Erasable PROM* i EEPROM – *eng. Electrically Erasable PROM*. U poluprovodničke memorije koje nemaju mogućnost brisanja spadaju ROM i PROM. Razlika između ROM i PROM memorije je to što sadržaj ROM memorije mora biti upisan pri proizvodnji čipa dok je PROM memorija proizvedena prazna i uz pomoć programatora se na nju upisuje željeni sadržaj. U poluprovodničke memorije koje imaju mogućnost brisanja spadaju EPROM i EEPROM. Sadržaj EPROM memorije je moguće obrisati putem jake ultraljubičaste svetlosti. EEPROM memoriju je moguće elektronski obrisati što omogućava brisanje i pisanje za vreme rada sistema. FLASH memorija je podtip EEPROM memorije koji uklanja nedostatke starijih načina izrade EEPROM memorije koji su ograničavali gustinu pakovanja memorije [9].

	ROM	EPROM	Single Gate Flash	Split Gate Flash	EEPROM
<b>Density</b>	+++	++	+	-	---
<b>Electrically Prog.</b>	---	+	+	+	+
<b>Electrically Erase.</b>	---	---	+	+	+
<b>Byte Erasable</b>	-	-	-	-	+
<b>Program Disturb</b>	+	+	--	-	++
<b>Over Erase/Program</b>	+	+++	-	+	++
<b>Process Complexity</b>	+++	++	--	+	-
<b>Manufacturability</b>	+++	++	-	+	+
<b>Cost</b>	+++	++	+	+	--

Worst --- -- - + ++ +++ Best

Source: Motorola/ICE, "Memory 1997"

20810

## 2-2 Poređenje trajnih memorija zasnovanih na poluprovodničkim tehnologijama [10]

Uz zahtev za mogućnost izmene podataka na memoriji u toku rada sistema EEPROM memorija postaje jedini izbor.

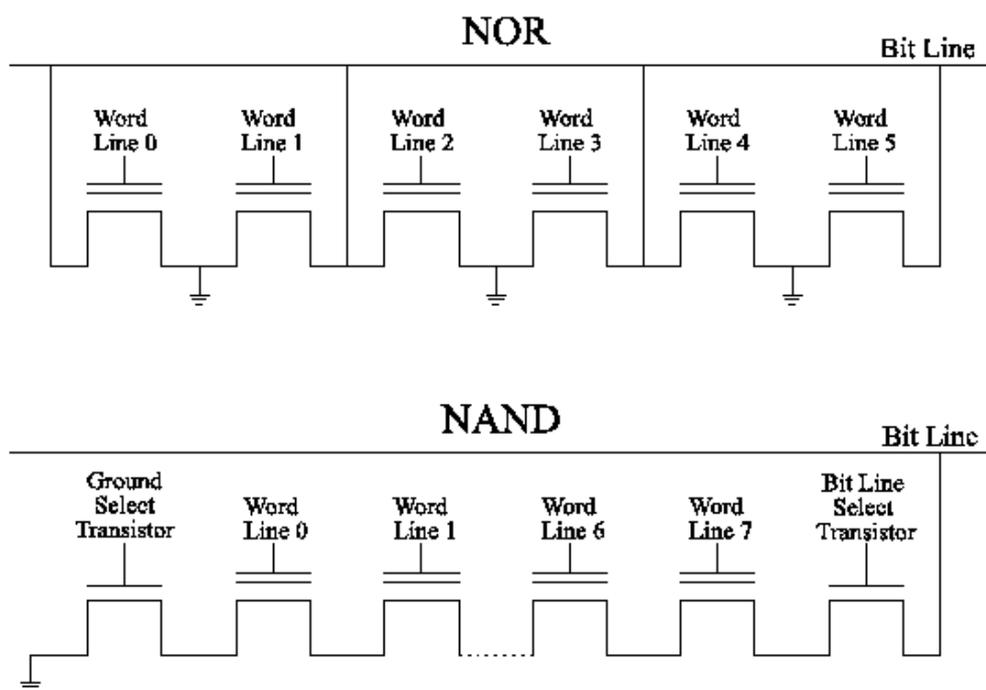
Trajnu memoriju je potrebno povezati odgovarajućom spregom. Unutar namenskih sistema za svrhu povezivanja periferija i različitih komponenti koriste se sprege kao što su I2C i SPI. Obe sprege su napravljene za serijsku komunikaciju između komponenti u okviru namenskih sistema.

I2C – eng. *Inter-Integrated Circuit* je komunikaciona sprega koja podržava sprezanje više vodećih i više pratećih uređaja. Cilj I2C sprege je povezivanje perifernih uređaja sa centralnim procesorima niskom brzinom komunikacije - tipično 100Kb/s. Uređaji mogu komunicirati u jednom smeru [11]. Brzina koju pruža I2C sprega može praviti usko grlo u brzini sistema pa je ova sprega lošiji izbor.

SPI – eng. *Serial Peripheral Interface* je komunikaciona sprega koja podržava sprezanje jednog vodećeg i više pratećih uređaja. Moguća je simultana dvosmerna komunikacija između vodećeg i pratećeg uređaja. Brzina prenosa je veća od I2C sprege zbog veće brzine takta, zbog toga što se ceo propusni opseg koristi za podatke jer nema metapodataka i zbog toga što je moguće ostvariti simultanu dvosmernu komunikaciju [11]. U okviru ovog rada FLASH memorija se vezuje putem SPI sprege pa će dalji fokus biti na njoj.

## 2.2 FLASH memorija

FLASH memorija je trajna računarska memorija koju je moguće elektronski obrisati i programirati. Kreirana je 1980. godine od strane Dr. Fujio Masuoka. FLASH memorija omogućava brisanje delova memorije, podizanjem potencijala na jednomvodu povezanom na celu grupu memorijskih ćelija. Ovaj dizajn je kompromis između fleksibilnosti i cene. Prethodne memorije takođe bazirane na FGMOS – *eng. floating-gate MOSFET* tehnologiji su bile u stanju da brišu memorijske ćelije jednu po jednu što ih je činilo kompleksnijim, skupljim i većim potrošačima električne energije [12]. FLASH memorija se može podeliti na NOR i NAND FLASH [13]. Razlika između ova dva tipa FLASH memorije je u povezivanju između Source i Bit vodova.



Slika 2-3 Razlika u izradi NOR i NAND FLASH memorije [13]

Razlike u implementaciji dovode do razlika u sposobnosti ova dva tipa FLASH memorije.

NOR FLASH ima visoke brzine čitanja i omogućava adresiranje na nivou najmanje veličine koja može biti adresirana. Zbog tih karakteristika NOR FLASH podržava *eng. eExecution in place* – XIP. XIP dozvoljava izvršavanje programske podrške direktno sa FLASH memorije čime se zaobilazi korak upisa u sistemski RAM. Nedostaci NOR FLASH-a su spori ciklusi pisanja i brisanja i fizički veće memoriske ćelije od NAND FLASH-a. NAND FLASH pruža visoku brzinu čitanja i pisanja ali mu je nedostatak komplikovaniji pristup

memoriji. Podacima upisanim na NAND FLASH-u se mora pristupiti serijski u blokovima. NAND FLASH ne može pristupiti memoriji na nivou manjem od veličine bloka.

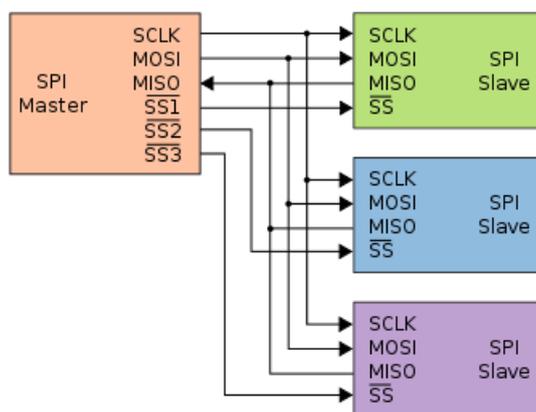
Serijski FLASH je mala FLASH memorija niske potrošnje koja dozvoljava serijski pristup podacima umesto adresiranja individualnih adresnih lokacija. Prednost ovakve implementacije je to što magistrale za povezivanje, poput često korišćene SPI magistrale, koriste manji broj vodova na štampanoj ploči nego sama paralelna FLASH memorija. Serijska FLASH memorija se često koristi za čuvanje firmvera.

## 2.3 SPI sprega

SPI sprežni sistem razvijen je u Motoroli početkom 2000. godine sa krajnjom revizijom napisanom 2004. godine [14]. Osim ovog standarda postoji još adhoc izvedbi SPI sprega [11]. Sve izvedbe imaju zajedničko to što imaju vod za:

- Takt serijskog uređaja *eng. Serial Clock - SCLK*.
- Izlaz iz vodećeg/Ulaz u prateći *eng. Master Out/Slave In - MOSI*.
- Ulaz u vodećeg/Izlaz iz pratećeg *eng. Master In/Slave Out - MISO*.
- Odabir pratećeg uređaja *eng. Slave Select – SS*.

Vodovi SCLK, MOSI, MISO se povezuju od vodećeg ka svim pratećih uređajima, prateći ih dele. Vodovi SS se povezuju od vodećeg ka pojedinačnim pratećim, svaki prateći ima jedan SS vod.

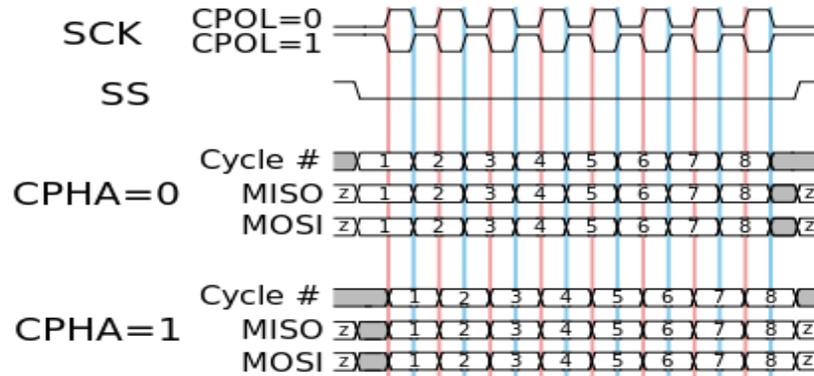


Slika 2-4 Primer povezivanja jednog vodećeg SPI uređaja sa tri prateća [6]

SCLK je vod koji prenosi takt korišćen u komunikaciji između serijskih uređaja. MOSI i MISO vodovi se koriste za slanje podataka između vodećeg i odabranog pratećeg uređaja, smer prenosa je naznačen u imenu vodova. SS vodovi su korišćeni za odabir pratećeg uređaja sa kojim vodeći uređaj izvršava komunikaciju u određenom trenutku.

Postoji više načina na koji se može sinhronizovati slanje i prijem podataka putem takta. Način koji će se koristiti mora biti isti na vodećem i pratećim uređajima. Logiku koju će učesnici u komunikaciji koristiti definišu dva parametra:

- Faza Takta *eng. Clock Phase - CPHA*
- Polaritet takta *eng. Clock Polarity – CPOL*



Slika 2-5 Vremenski dijagram polariteta i faze takta [6]

CPOL bira da li je bazična vrednost takta 0 ili 1. CPHA bira da li se podatak čita na rastućoj a menja na opadajućoj ivici takta ili obratno.

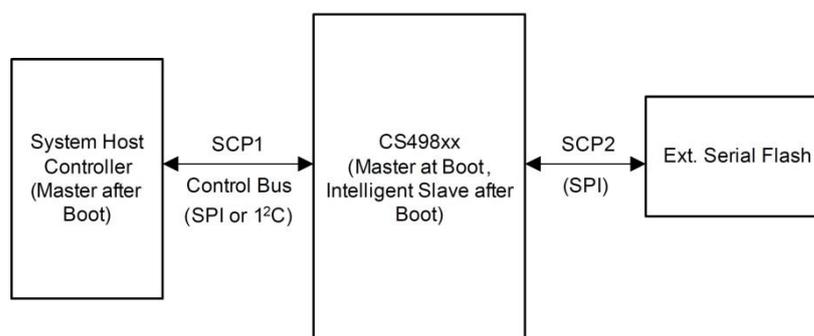
CPOL:\nCPHA:	0	1
0	Čitanje na rastućoj ivici Menjanje na opadajućoj	Čitanje na opadajuću ivicu Menjanje na rastuću
1	Čitanje na opadajuću ivicu Menjanje na rastuću	Čitanje na rastućoj ivici Menjanje na opadajućoj

Tabela 2-1 Uticaj CPOL i CPHA na logiku prenosa podataka

### 3. Analiza pristupa korišćenja SPI FLASH memorije na Cirrus Logic platformi

Cilj ovog rada je razvoj rukovaoca za SPI FLASH memoriju na platformi zasnovanoj na procesoru izrađenom u ARM arhitekturi. Uvod ARM arhitekture u tržište u kom se većinski nalaze rešenja zasnovana na DSP-u nalaže istraživanje rešenja proverenih u realnom vremenu na platformi zasnovanoj na DSP-u. Primer ovakve platforme je platforma zasnovana na DSP procesoru CS49844 firme Cirrus Logic.

U okviru Cirrus Logic platforme zasnovane na CS49844 DSP-u eksterna SPI FLASH memorija je povezana kao sekundarni SPI uređaj DSP-a, dok je primarni mikrokontroler koji upravlja celokupnim sistemom [15]. Za vreme inicijalizacije DSP se nalazi u glavnoj ulozi (*eng. master-boot mode*) dok se nakon inicijalizacije nalazi u ulozi inteligentnog pratećeg uređaja (*eng. Intelligent slave*) a glavnu ulogu preuzima mikrokontroler. SPI FLASH memorija je uvek prateći uređaj DSP-u.



Slika 3-1 Način rada DSP-a u master boot modu [15]

U SPI FLASH memoriju se upisuju podaci i programska podrška koji vrše inicijalizaciju i različite obrade. Za kreiranje podataka koje upisujemo na SPI FLASH (flash

slike) koristi se DSP Condenser Wizard skup alata. Najbitnije izlazne datoteke DSP Condenser Wizard-a su sledeće:

egg.img - Datoteka koja sadrži programsku podršku za inicijalizaciju DSP-a. Neophodno je postaviti je na početnu lokaciju u SPI FLASH memoriji da bi proces inicijalizacije tačno radio. Ova slika se kreira tako što generičku sliku izmenimo tako da sadrži konkretne vrednosti za tip FLASH-a i početnu lokaciju flash.img slike.

flash.img - Glavni izlazni fajl DSP Condenser Wizard-a. Sadrži odabranu programsku podršku za vršenje obrade (ULD) i sve datoteke sa konfiguracionim informacijama određene kao modove u DSP Condenser projektu. Takođe može sadržati unapred popunjene datoteke sa konfiguracionim informacijama koje se upisuju u memorijski prostor zauzet za konfiguracione datoteke koje mikrokontroler kreira za vreme izvođenja.

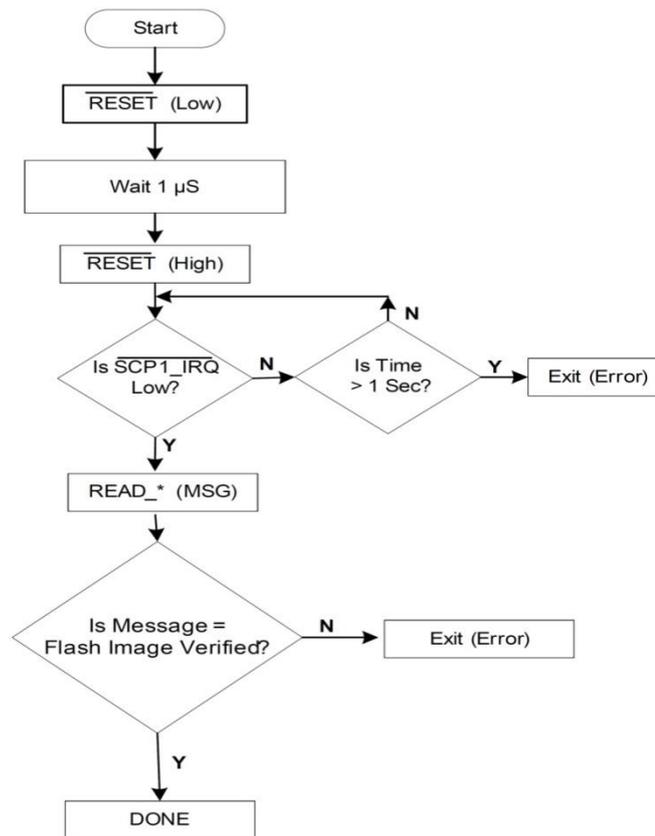
flash.h - C-ovska datoteka u .h formatu. Sadrži informacije o flash.img datoteci koje mikrokontroler koristi. Konkretno informacije služe za mapiranje simboličkih imena za razne komponente programske podrške i modova definisanim u projektu DSP Condenser Wizard-a na celobrojne vrednosti korišćene u komunikaciji od mikrokontrolera ka DSP-u

Ostale datoteke se koriste za ažuriranje i kao međukoraci u kreiranju prethodno spomenutih datoteka.

### **3.1 Inicijalizacija DSP-a**

Proces inicijalizacije DSP-a zahteva čitanje inicijalnih podataka sa SPI FLASH memorije. Procedura za inicijalizaciju je sledeća:

1. Podesiti RESET na nisku vrednost. Reset pin je aktivan na niskoj vrednosti signala.
2. Sačekati 1 mikrosekundu.
3. Podesiti RESET na visoku vrednost. Visoka vrednost RESET-a započinje prebacivanje podataka sa SPI memorije.
4. Sačekati da SCP1\_IRQ opadne na nisku vrednost. Označava da je prenos podataka završen
5. Izčitati poruku od DSP-a
6. Ako je poruka "Flash image verified" to nam signalizira uspešno završenu inicijalizaciju i verifikovane podatke.



Slika 3-2 Algoritam inicijalizacije putem SPI FLASH memorije [15]

Nakon inicijalizacije DSP uvek šalje poruku vezanu za stanje DSP SPI FLASH memorije. Ako poruka signalizira oštećenje na podacima očekuje se ažuriranje. Ako je memorija ispravna sistem nastavlja sa podešavanjem audio izvora.

### 3.2 Uloga SPI FLASH memorije nakon inicijalizacije

Nakon inicijalizacije SPI FLASH memorija se koristi za učitavanje modula koji se aktivno koriste. Na primer moduli za dekodovanje se menjaju pri izmeni izvornog audio signala. Nakon što DSP prepozna izmenu audio signala on učitava podrazumevanu programsku podršku korišćenu pri dekodovanju tog tipa audio signala. Da bi mikrokontroler znao da se desila izmena DSP ga obaveštava putem poruka.

Poruke su 32 bitne vrednosti koje se sastoje od operacionog koda i indeksa. Sa strane mikrokontrolera firmver DSP-a se može gledati kao blokovi više 32 bitnih vrednosti. Indeksi u porukama se odnose na ove 32 bitne vrednosti tj. registre. Operacioni kod nosi informaciju o operaciji koja se izvršava. Na osnovu operacionog koda poruke koje se razmenjuju između mikrokontrolera i DSP-a mogu da spadaju u komunikaciju jednog od sledećih tipova:

- Pisanje ka DSP-u
- Zatraženo čitanje
- Nezatraženo čitanje

U slučaju operacija pisanja i zatraženog čitanja operacioni kod identifikuje modul nad kojim se operacija izvršava, količinu podataka i tip prenosa. U svrhu identifikovanja modula koriste se vrednosti koje je DSP Condenser Wizard smestio u flash.h fajl. U slučaju nezatraženog čitanja operacioni kod sadrži vrednost koja označava događaj koji je prouzrokovao nezatraženo čitanje. Indeks predstavlja registar modula nad kojim se operacija izvršava.

Poruke tipa pisanje ka DSP-u se sastoje od jedne 32 bitne komandne reči praćene sa rečima koje prenose podatke kojih može biti od jedne do trideset i dve. Komandna reč je u formatu:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID Modula nad čijim registrima vršimo pisanje [30:24]								R/W		Broj reči za prenos podataka [20:16]					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Indeks [15:0]															

Tabela 3-1 Komandna reč za poruke pisanja ka DSP-u

23-i bit	22-i bit	Komanda
0	0	Pisanje preko trenutne vrednosti
0	1	Pisanje rezultata operacije logičkog ili između trenutne vrednosti i nove vrednosti
1	0	Pisanje rezultata operacije logičkog i između trenutne vrednosti i nove vrednosti

Tabela 3-2 Opcije za bite 23:22 u komandnoj reči pisanja ka DSP-u

Poruke tipa zatraženog pisanja se mogu smatrati zahtevom za čitanje sadržaja uzastopnih registra. Nakon slanja 32 bitne komandne reči, DSP odgovara sa 32 bitnom komandnom reči odgovora na čitanje praćenom sa rečima koje prenose podatke zatražene u zahtevu kojih ima onoliko koliko je zatraženo, najmanje jedna a najviše trideset i dve. Formatu poruka koje spadaju u tip zatraženog čitanja su:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
1	ID Modula nad čijim registrima vršimo čitanje[30:24]							1	1	0	Broj reči za prenos podataka [20:16]					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Indeks [15:0]																

*Tabela 3-3 Komandna reč za poruke zatraženog čitanja*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
0	ID Modula nad čijim registrima vršimo čitanje [30:24]							1	1	0	Broj reči za prenos podataka [20:16]					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Indeks [15:0]																

*Tabela 3-4 Komandna reč za odgovor na poruku zatraženog čitanja*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Podaci [31:0]															

*Tabela 3-5 Reč za prenos podataka zatraženog čitanja*

Poruke tipa nezatraženog čitanja se koriste u slučajevima kada DSP mora da obavesti mikrokontroler da su se desile promene u sistemu na koje mikrokontroler mora da odreaguje. Svaki put kad mikrokontroler detektuje postojanje nezatražene poruke on treba pročitati dve 32 bitne vrednosti koje imaju sledeći format:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Operacioni kod [31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Indeks [15:0]															

*Tabela 3-6 Komandna reč za odgovor na poruku nezatraženog čitanja*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Podaci [31:0]															

*Tabela 3-7 Reč za prenos podataka nezatraženog čitanja*

Poruke tipa nezatraženog čitanja se koriste u slučajevima kada DSP mora da obavesti mikrokontroler da su se desile promene u sistemu na koje mikrokontroler mora da odreaguje. Svaki put kad mikrokontroler detektuje postojanje nezatražene poruke on treba pročitati dve 32 bitne vrednosti koje imaju sledeći format:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Operacioni kod [31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Indeks [15:0]															

*Tabela 3-8 Komandna reč za odgovor na poruku nezatraženog čitanja*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Podaci [31:0]															

*Tabela 3-9 Reč za prenos podataka nezatraženog čitanja*

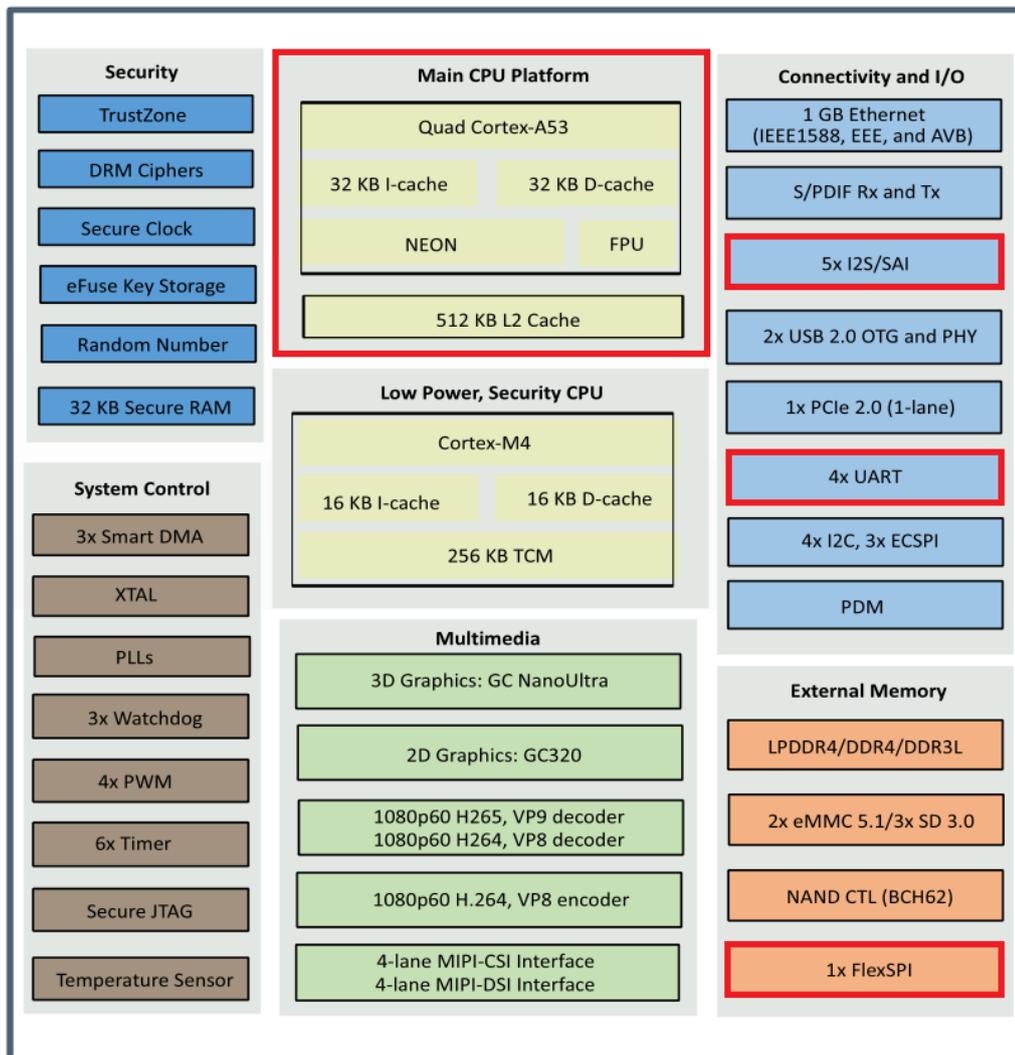
Na osnovu analize Cirrus Logic platforme vidimo da se SPI FLASH koristi u svrhu podizanja sistema i učitavanja programske podrške zadužene za željenu obradu i konfiguracionih podataka potrebnih za tu obradu. Za vreme rada sistema podsistem za kontrolisanje celokupnog sistema, u ovom slučaju mikrokontroler, treba imati mehanizam kojim bi kontrolisao programsku podršku za obradu koja je učitana iz trajne memorije. U okviru platforme zasnovane na CS49844 to je sistem komunikacije putem poruka.

## **4. Realizacija programske podrške za SPI FLASH rukovalac**

Na osnovu analize iz prethodnog poglavlja vidimo primer korišćenja SPI FLASH memorije. Za pravilan rad sistema rukovalac je zadužen da pruži osnovne operacije SPI FLASH memorije, a to su čitanje, pisanje i brisanje, dok ostatak sistema vrši kontrolisanje toga koje operacije se pozivaju, kada i nad kojim delovima memorije.

### **4.1 Opis fizičke arhitekture na kojoj je izvršena implementacija**

Platforma na kojoj je izvršen praktičan deo bazirana je na i.MX 8M Mini fizičkoj arhitekturi. Fizička arhitektura i.MX 8M Mini sadrži dva kompleksa procesorskih jezgra Quad Cortex-A53 i Cortex-M4, mnoštvo periferija za ulaz i izlaz podataka od kojih su korišćene UART i SAI kao i različite eksterne memorije od kojih smo koristili FlexSPI.



Slika 4-1 Blok dijagram i.MX 8M Mini sistema [1], crvenim kvadratima su označene celine korišćene u okviru razvijene programske podrške

Quad Cortex-A53 je četvororojezgarni procesor izrađen u ARM v8-A arhitekturi koji radi na brzinama do 1.6 GHz. Cortex-M4 je jednojezgarni procesor izrađen u ARM v7 arhitekturi koji radi na brzini od 400 Mhz. Quad Cortex-A53 je glavna procesorska platforma dok se Cortex-M4 koristi kao platforma manje energetske potrošnje i u bezbednosne svrhe. Programska podrška rukovaoca za SPI FLASH je pisana za Quad Cortex-A53 procesor.

Model fizičke arhitekture SPI FLASH je Micron Serial NOR Flash Memory MT25QU256ABA. Podržane su brzine takta do 166 MHz za protokole u režimu *eng. Single Transfer Rate* i brzine do 90 MHz za protokole u režimu *eng. Double Transfer Rate*. Ovaj čip ima 256 Mb memorije koja se može podeliti u sektore veličine 64KB ili podsektore veličine 32KB i 4KB. Čip pruža mogućnosti *eng. eExecute In Place - XIP*. Pružena je mogućnost zaštite od pisanja i brisanja na nivou sektora veličine 64KB.

## 4.2 Programsko rešenje

U okviru programske podrške za rukovalac SPI FLASH memorije implementirane su funkcionalnosti inicijalizacije SPI FLASH periferije, upis podataka veličine jedne stranice na memoriju, čitanje podataka veličine jedne stranice sa memorije, brisanje sektora, brisanje svih podataka u memoriji. Osim ovih funkcionalnosti implementirani su i rukovaoci UART i SAI periferijama čije funkcionalnosti koje nam pružaju mogućnost demonstracije rada SPI FLASH periferije.

Programsko rešenje se može podeliti u dva dela:

- Rukovalac SPI FLASH periferije – Ovaj deo se sastoji od deklaracija adresa, registra i struktura koje predstavljaju periferije, funkcija koje izvršavaju operacije SPI FLASH čipa kao što su na primer funkcija za omogućavanje pisanja i funkcija za čitanje vrednosti iz memorije.
- Aplikacija za prikaz rada – Ovaj deo je zaslužan za prikazivanje funkcionalnosti rukovaoca SPI FLASH-a. Sastoji se od funkcija koje pozivaju funkcije rukovaoca na određeni unos od strane korisnika kao i rukovaoca UART sprege za komunikaciju sa korisnikom i rukovaoca SAI periferije za reprodukciju i prijem zvučnog signala.

### 4.2.1 Rukovalac SPI FLASH periferije

Rukovalac SPI FLASH-a realizovan je uz pomoć SDK pruženog od strane kompanije NXP. U njemu se nalaze fajlovi sa deklaracijama struktura koje predstavljaju periferije, deklaracije adresa periferija i njihovih registra i fajlove sa funkcijama koji pružaju funkcije koje vrše kontrolu SPI FLASH periferije na niskom nivou.

Deklaracije struktura, adresa i registra periferija nalaze se u okviru MIMX8MM6\_cm4.h fajla.

Deklaracija strukture koja predstavlja FlexSPI periferiju:

```
typedef struct {  
    __IO uint32_t MCR0;  
    __IO uint32_t MCR1;  
    __IO uint32_t MCR2;  
    __IO uint32_t AHBCR;
```

```
__IO uint32_t INTEN;  
__IO uint32_t INTR;  
__IO uint32_t LUTKEY;  
__IO uint32_t LUTCR;  
__IO uint32_t AHB RXBUFCR0[8];  
uint8_t RESERVED_0[32];  
__IO uint32_t FLSHCR0[4];  
__IO uint32_t FLSHCR1[4];  
__IO uint32_t FLSHCR2[4];  
uint8_t RESERVED_1[4];  
__IO uint32_t FLSHCR4;  
uint8_t RESERVED_2[8];  
__IO uint32_t IPCR0;  
__IO uint32_t IPCR1;  
uint8_t RESERVED_3[8];  
__IO uint32_t IPCMD;  
__IO uint32_t DLPR;  
__IO uint32_t IPRXFCR;  
__IO uint32_t IPTXFCR;  
__IO uint32_t DLLCR[2];  
uint8_t RESERVED_4[24];  
__I uint32_t STS0;  
__I uint32_t STS1;  
__I uint32_t STS2;  
__I uint32_t AHBSPNDSTS;  
__I uint32_t IPRXFSTS;  
__I uint32_t IPTXFSTS;
```

```

uint8_t RESERVED_5[8];

__I uint32_t RFDR[32];

__O uint32_t TFDR[32];

__IO uint32_t LUT[128];
} FlexSPI_Type;

```

Putem polja ove strukture pristupa se vrednostima registra FlexSPI periferije čime je olakšano kontrolisanje periferije na niskom nivou.

Funkcije za kontrolu FlexSPI periferije na niskom nivou nalaze se u `fsl_flexspi.c` fajlu. Ove funkcije kontrolišu vrednosti kontrolnih, komandnih i registra za prijem i slanje podataka da bi izvršile operacije SPI FLASH čipa. U nastavku prikazan je primer dela funkcije za inicijalizaciju koji inicijalizuje vrednost nultog kontrolnog registra modula:

```

void FlexSPI_Init(FlexSPI_Type *base, const flexspi_config_t *config) {

    uint32_t configValue = 0;

    ...

    /* Configure MCR0 configuration items. */

    configValue = FlexSPI_MCR0_RXCLKSRC(config->rxSampleClock) |

        FlexSPI_MCR0_DOZEEN(config->enableDoze) |

        FlexSPI_MCR0_IPGRANTWAIT(config->ipGrantTimeoutCycle) |

        FlexSPI_MCR0_AHBGRANTWAIT(config->ahbConfig.ahbGrantTimeoutCycle) |

        FlexSPI_MCR0_SCKFREERUNEN(config->enableSckFreeRunning) |

        FlexSPI_MCR0_HSEN(config->enableHalfSpeedAccess) |

        FlexSPI_MCR0_COMBINATIONEN(config->enableCombination) |

        FlexSPI_MCR0_ATDFEN(config->ahbConfig.enableAHBWriteIpTxFifo) |

        FlexSPI_MCR0_ARDFEN(config->ahbConfig.enableAHBWriteIpRxFifo) |

        FlexSPI_MCR0_MDIS_MASK;

    base->MCR0 = configValue;

    ...

};

```

Pored inicijalizacije konfiguracionih registra potrebna je i inicijalizacija *eng. Look Up Table* - LUT. Ovu operaciju izvršava *FlexSPI\_UpdateLUT* funkcija. LUT sadrži do trideset dve sekvence koje čip može izvršiti. Sekvence LUT se sastoje od osam šesnestobitnih instrukcija.

LUT	Udaljenost od početne adrese	Sekvenca		Udaljenost od početne adrese
Sekvenca 0	0x0	Instrukcija 1	Instrukcija 0	0x0
Sekvenca 1	0x10	Instrukcija 3	Instrukcija 2	0x4
Sekvenca 2	0x20	Instrukcija 5	Instrukcija 4	0x8
...		Instrukcija 7	Instrukcija 6	0xC
Sekvenca N	0x10 * N	[31:16]	[15:0]	

Instrukcija	opcode	num_pads	operand
	[15:10]	[9:8]	[7:0]

Tabela 4-1LUT, struktura sekvence i struktura instrukcije

Pored funkcije za inicijalizaciju od velikog značaja su i funkcije koje prenose podatke i instrukcije između SPI FLASH-a i ostatka sistema. Postoje dve funkcije koje vrše prenos to su:

<i>status_t FlexSPI_TransferBlocking(FlexSPI_Type *base, flexspi_transfer_t *xfer);</i>
<i>status_t FlexSPI_TransferNonBlocking(FlexSPI_Type *base, flexspi_handle_t *handle, flexspi_transfer_t *xfer);</i>

Razlika u ovim funkcijama je priroda njihovog izvršavanja. Funkcija *FlexSPI\_TransferBlocking* će blokirati procesor dok se ne obavi prenos dok funkcija *FlexSPI\_TransferNonBlocking* prenos obavlja uz pomoć rukovaoca prekidima samim tim ne blokira već vraća nakon što započne prenos.

Obe funkcije rade tako što prime podatke o transferu putem *flexspi\_transfer\_t* strukture. Nakon unosa podatka o transferu iz strukture u odgovarajuće registre u registar *IPCMD* se unosi vrednost za okidanje komande čime se započinje transfer. Nakon toga u zavisnosti od toga kojom funkcijom vršimo prenos ili se omogućuju odgovarajući prekidi

FlexSPI perifrije ili se pozivaju funkcije koje će blokirajućom metodom primiti podatke sa perifrije.

Kod koji definiše strukturu podataka *flexspi\_transfer\_t*:

```
typedef struct _flexspi_transfer
{
    uint32_t deviceAddress;    /*!< Operation device address. */
    flexspi_port_t port;      /*!< Operation port. */
    flexspi_command_type_t cmdType; /*!< Execution command type. */
    uint8_t seqIndex;        /*!< Sequence ID for command. */
    uint8_t SeqNumber;       /*!< Sequence number for command. */
    uint32_t *data;          /*!< Data buffer. */
    size_t dataSize;        /*!< Data size in bytes. */
} flexspi_transfer_t;
```

Osim ovih funkcija u okviru *fsl\_flexspi.c* postoje funkcije za prosleđivanje konfiguracionih informacija čipu, čitanje podrazumevanih vrednosti podešavanja, proveru grešaka i funkcije za rad sa prekidima.

Koristeći funkcije za kontrolu SPI FLASH-a na niskom nivou napravljene su funkcije rukovaoca iz fajla *diy\_flexspi.c* koje pružaju operacije višeg nivoa. Pružene su operacije za:

- Inicijalizaciju - *flexspi\_diy\_flash\_init* – vrši inicijalizaciju FlexSPI perifrije, konfigurisanje FLASH čipa i inicijalizaciju LUT.
- Upis jedne stranice podataka - *flexspi\_diy\_flash\_page\_program* - vrši upis vrednosti sa prosleđene adrese na specificiranu stranicu u SPI FLASH memoriji.
- Čitanje putem jedne linije - *flexspi\_diy\_flash\_read\_normal* - vrši čitanje putem jedne linije za podatke SPI FLASH čipa.
- Čitanje putem četiri linije - *flexspi\_diy\_flash\_read\_fast\_quad* – vrši čitanje putem četiri linije za podatke SPI FLASH čipa.
- Ulazak u quad mod - *flexspi\_diy\_enable\_quad\_mode* - quad mod omogućuje prenos putem četiri linije.
- Brisanje podsektora od 4KB - *flexspi\_diy\_flash\_erase\_subsector* – vrši brisanje specificiranog podsektora.
- Brisanje svih podataka sa čipa - *flexspi\_diy\_erase\_chip* – vrši brisanje svih podataka na čipu. Zahteva podešavanja sigurnosnih registara koji su programabilni samo jednom pa je na višim nivoima korišćenje ove funkcije zamenjeno korišćenjem više funkcija za brisanje podsektora.

- Čitanje identifikacije sa čipa - *flexspi\_diy\_get\_vendor\_id* – vrši čitanje prvog bajta identifikacije SPI FLASH periferije. Koristi se za proveru inicijalizacije jer ako povratna vrednost ne bude tačna to označava pogrešno inicijalizovanu periferiju.

Osim funkcije za inicijalizaciju sve funkcije se oslanjaju na podešavanje *flexspi\_transfer\_t* strukture i pozivanje neke od funkcija za prenos. U zavisnosti od toga koja operacija se izvršava potrebno je omogućiti pisanje što se vrši *flexspi\_diy\_write\_enable* funkcijom pre pozivanja funkcije za prenos. Nalik pozivanju funkcije *flexspi\_diy\_write\_enable* pre prenosa, nakon pozivanja funkcije za prenos potrebno je pozvati funkciju *flexspi\_diy\_wait\_bus\_busy* koja čeka na oslobađanje magistrale.

#### 4.2.2 Aplikacija za prikaz rada

Aplikacija za prikaz rada okuplja rukovaoce UART spregom, SAI i SPI FLASH periferijama i pruža mogućnost demonstriranja funkcionalnosti SPI FLASH rukovaoca. Koristeći funkcije iz rukovaoca SPI FLASH periferije napravljen je skup omotačkih funkcija koje se pozivaju nakon unosa komandi od strane korisnika putem UART sprege. Ove funkcije se nalaze u *demo.c* fajlu.

Pružene operacije su:

- Pisanje i čitanje ispitnih podataka.
- Brisanje sektora u kom se vrši upis testnih podataka.
- Upis koeficijenate filtera korišćenih u obradi.
- Vršenje jednostavne obrade, filtriranja.
- Podešavanje filtera korišćenih u obradi.
- Brisanje svih podataka sa čipa.

Funkcionalnost SPI FLASH-a je prikazana kroz osnovne funkcije čitanja, pisanja i brisanja. Čitanje i pisanje se vrši nad jednom odabranom stranicom funkcijama:

```
status_t DataWrite(FlexSPI_Type *base, uint8_t *buffer, uint32_t bufferSize, uint32_t address, flexspi_handle_t *handle);
```

```
status_t DataRead(FlexSPI_Type *base, uint8_t *buffer, uint32_t bufferSize, uint32_t address, flexspi_handle_t *handle);
```

Filtriranje se izvršava nad dva bafera čije se vrednosti nalaze u okviru programske podrške i nad audio podacima primljenim putem ulazne SAI sprege. Zbog razlike u prirodi podataka i načinima obrade postoji više funkcija za obradu.

Obrada bafera koji se prikazuje grafikom:

```
status_t DoGraphProcessing(FlexSPI_Type *base, double *buffer, uint32_t bufferSize,
uint32_t coefAddress, uint32_t coefSize, flexspi_handle_t *handle);
```

Ova funkcija za obradu vrši obradu nad vrednostima u pokretnom zarezu. Podaci su zapisani kao jednokanalni pa nema premeštanja članova bafera nad kojim se vrši obrada.

Obrada bafera koji se reprodukuje na SAI:

```
status_t DoProcessing(FlexSPI_Type *flexspiBase, I2S_Type *saiBase, int16_t *buffer,
uint32_t bufferSize, uint32_t coefAddress, uint32_t coefSize, sai_transfer_t *xfer,
flexspi_handle_t *flexspiHandle, sai_handle_t *saiHandle);
```

Ova funkcija prima bafer sa šesnaestobitnim označenim celobrojnim podacima koji su zapisani u formatu nalik dvokanalnom. Obrada se vrši nad 32 bitnim označenim celobrojnim vrednostima pa se u okviru funkcije vrši konverzija.

Obrada audio podataka koji su primljeni putem SAI sprege:

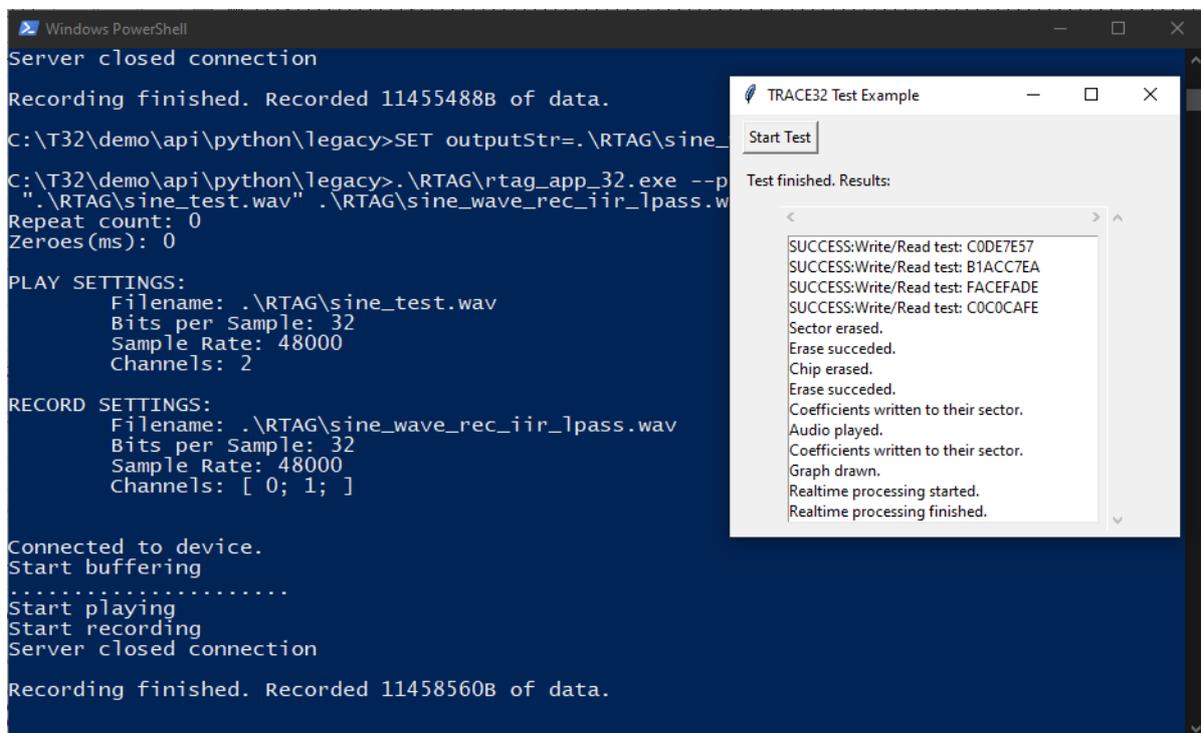
```
status_t DoFIRProcessing(int32_t *buffer, uint32_t bufferSize, uint32_t coefSize,
uint8_t bufferToUse);
```

```
status_t DoIIRProcessing(int32_t *buffer, uint32_t bufferSize, uint32_t coefSize, uint8_t
bufferToUse);
```

Pružene su dve funkcije za dva tipa filtera. Obe funkcije primaju 32 bitne označene celobrojne vrednosti. Funkcije vrše obradu nad jednim kanalom pa je pre poziva ovih funkcija potrebno je audio podatke primljene putem SAI sprege preurediti u dva bafera koji sadrže samo jedan kanal.

## 5. Verifikacija tačnosti rešenja

Verifikacija je urađena putem LAUTERBACH Trace32 PowerView programske podrške i PowerDebug USB 3.0 fizičke arhitekture. Trace32 nam pruža kontrolisano izvršavanje programske podrške na i.MX 8M Mini. Automatizacija procesa verifikacije je izvršena putem python skripte korišćenjem Trace32 python API-ja. Python skripta pokreće jednostavan program u kom se nalazi dugme Start Test na čiji klik se pokreću testovi pisanja i čitanja, brisanja i jednostavna obrada koja je kreirana za demonstraciju korišćenja SPI FLASH memorije.

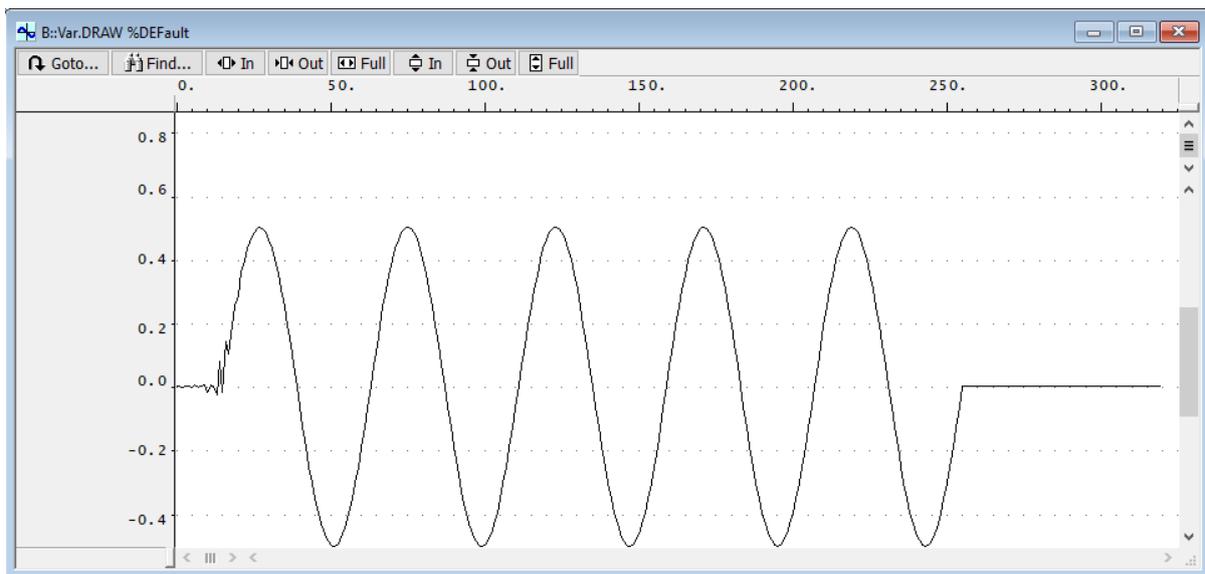


Slika 5-1 PowerShell komandna linija i grafičko korisničko okruženje koji kreira python skripta za testiranje rukovaoca

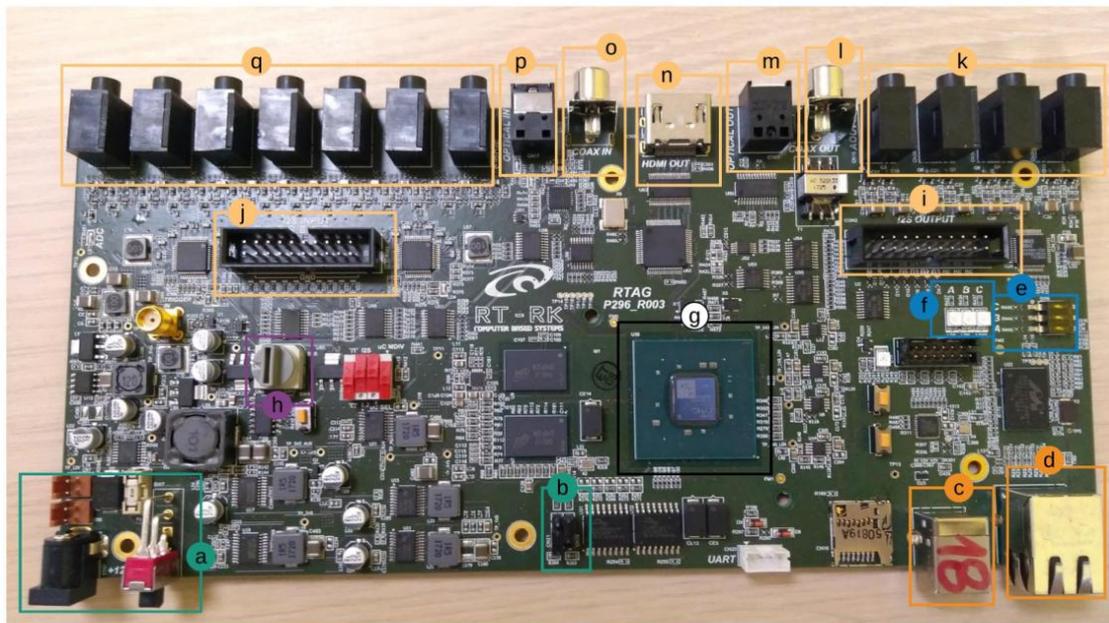
Verifikacija operacija pisanja i čitanja je izvršena tako što se programska podrška zaustavlja pre upisa vrednosti u SPI FLASH memoriju i nakon čitanja sa SPI FLASH memorije. Pre upisa na SPI FLASH pristupa se radnoj memoriji radi menjanja vrednosti bafera u kom se nalaze podaci koji će biti upisani. Ovo nam omogućava da testiramo upis sa različitim vrednostima. Nakon čitanja pristupa se radnoj memoriji radi čitanja vrednosti pročitane sa SPI FLASH memorije. Pročitana vrednost se poredi sa vrednosti koja je ranije upisana u radnu memoriju. Podudaranje te dve vrednosti označava uspešno obavljen par operacija pisanja i čitanja.

Operacije brisanja sektora i celog čipa se verifikuje brisanjem i čitanjem nakon toga. Ako je pročitana vrednost jednaka nizu binarnih jedinica to znači da je brisanje uspešno.

Operacije obrade izvršavaju filtriranje dva bafera iz programske podrške kao i filtriranje podataka primljenih putem SAI periferije. Filtrirana vrednost iz prvog bafera iz programske podrške se može snimiti putem *eng. RT-Audio Grabber - RTAG-a*. Filtrirana vrednost iz drugog bafera iz programske podrške se može videti na grafiku u okviru Trace32. Python skripta vrši pokretanje programske podrške koja započinje reprodukciju i snimanje zvuka putem RTAG-a tako da se može snimiti obrađen izlaz sa SAI periferije.



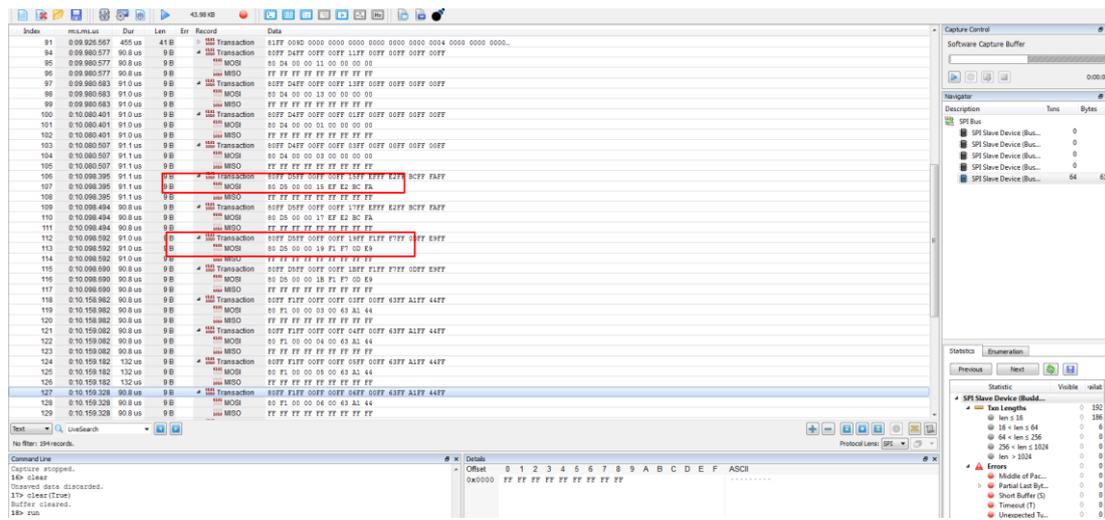
Slika 5-2 Izgled grafika u okviru Trace32 programske podrške



- a Power connector + switch
- e User switches
- i I2S output connector
- m S/PDIF optical output
- q Analog inputs
- b Boot jumpers
- f User LEDs
- j I2S input connector
- n HDMI output
- o S/PDIF coax input
- c USB connector
- g Zynq
- k Analog outputs
- p S/PDIF optical input
- d Ethernet connector
- h Audio input switch
- l S/PDIF coax output

Slika 5-3 RTAG fizička arhitektura

Pre izrade rukovaoca za SPI FLASH napravljen je rukovalac za ECSPi periferiju koji je korišćen kao upoznavanje sa platformom i SPI spregom. ECSPi periferija omogućava povezivanje SPI uređaja sa pločom putem GPIO pinova. Pri verifikaciji tog rukovaoca korišćene su Beagle fizička arhitektura za čitanje podataka i Data Center programska podrška kompanije Total Phase.



Slika 5-4 Programska podrška Data Center

## 6. Zaključak

Porast kompleksnosti modernih tehnologija za obradu zvuka radi dobijanja boljeg kvaliteta reprodukcije diktira porast u performansama fizičkih arhitektura koje su zadužene za tu obradu. Kao i svi ostali delovi fizičke arhitekture neophodno je koristiti trajnu memoriju visokih performansi. FLASH memorija pruža visoke brzine čitanja i pisanja uz malu potrošnju električne energije i nisku cenu izgradnje. Ovo je čini savršenim kandidatom za trajnu memoriju u okviru potrošačke elektronike za reprodukciju audio signala. Cilj ovog rada je bio prikaz rešenja programske podrške za upravljanje FLASH memorijom povezanom na ostatak sistema putem SPI sprege.

Programska podrška realizovana u radu pruža osnovne funkcije jedne trajne memorije a to su čitanje, pisanje i brisanje podataka. Predstavljeno rešenje je funkcionalan rukovalac SPI FLASH memorijom.

U radu su izložene teorijske osnove o trajnoj memoriji kao i spregama kojima se trajna memorija u okviru namenskih sistema povezuje sa ostatkom sistema. Objasnjeno je pristup korišćenja SPI FLASH memorije u okviru jednog realnog rešenja. Prikazana je platforma na kojoj je izvršen praktičan deo rada kao i delovi rešenja. Predstavljen je način verifikacije rešenja.

Za kontrolisano izvršavanje programske podrške korišćena je LAUTERBACH PowerDebug USB 3.0 fizička arhitektura u kombinaciji sa LAUTERBACH Trace32 programskom podrškom. Za reprodukciju i snimanje zvuka koristila se RT-AG platforma.

Potrebno je napomenuti da trenutno rešenje ne koristi prekide već blokira pri komunikaciji sa SPI FLASH memorijom. To je moguće je ispraviti ali zbog demonstracije koja je zahtevala korišćenje SAI periferije platforma pokrenuta sa *U-Boot bootloader*-om. *U-*

*boot* je rešio problem koji je SAI periferija pravila pri *eng. baremetal* implementaciji ali je unela problem loše sinhronizacije između obrađivača prekida i SPI FLASH-a.

Postojeće rešenje programske podrške za rukovalac SPI FLASH memorijom moguće je proširiti dodavanjem drugih relevantnih operacija u LUT koje bi omogućile još funkcionalnosti čipa.

## 7. Literatura

- [1] *i.MX 8M Mini Applications Processor Datasheet for Consumer Products*, Rev. 1, NXP Semiconductors, 07. 2020.
- [2] "NXP Brings Dolby Atmos® and DTS:X® to Living Rooms Everywhere with Immersiv3D Audio Solution." nxp.com <https://www.nxp.com/video/nxp-brings-dolby-atmos-and-dtsx-to-living-rooms-everywhere-with-immersiv3d-audio-solution:iMX-IMMERSIVE> (pristupljeno Jul. 2021)
- [3] M. Barr, "Embedded Systems Glossary." barrgroup.com <https://barrgroup.com/embedded-systems/glossary-e> (pristupljeno Jul. 2021)
- [4] M. Đukić. (2020). Programska podrška u realnom vremenu 1 – Uvod [Pdf slajdovi]. Dostupno na: [https://www.rtrk.uns.ac.rs/sites/default/files/materijali/predavanja/PPuRV\\_Predavanja\\_1\\_Uvod.pdf](https://www.rtrk.uns.ac.rs/sites/default/files/materijali/predavanja/PPuRV_Predavanja_1_Uvod.pdf)
- [5] M. Barr, A. Massa, "Introduction" u *Programming Embedded Systems With C and GNU Development Tools*, 2nd Edition, O'Reilly Media, 2006. pp 9-10.
- [6] M. Antičić, „Jedno rešenje softverske arhitekture prijelnika audio i video sadržaja zasnovano na digitalnom signal procesoru“, Novi Sad, FTN, 2018
- [7] M. Hajduković, Ž. Živanov, "Evolucioni period arhitekture računara oko 1990. godine," u *Arhitektura računara (pregled principa i evolucije)*, Novi Sad, Srbija, FTN, 2019. pp 261-263.
- [8] "Embedded memory systems 101." embeddedcomputing.com <https://www.embeddedcomputing.com/technology/storage/embedded-memory-systems-101> (pristupljeno Jul. 2021)
- [9] F. Masuoka i H. Iizuka, „Semiconductor Memory Device and Method for Manufacturing the Same“, U.S. Patent 4 531 203 (A) , Jul. 23. 1985.

- 
- [10] B. Matas, C. de Suberbasaux, "Embedded memory," u *Memory 1997*, 1997. Integrated Circuit Engineering Corporation, Dostupno na: <http://smithsonianchips.si.edu/ice/cd/MEMORY97/SEC11.PDF>
- [11] *Audio sprežni podsistemi*, RT-RK, Novi Sad, Srbija, 2012. pp 19-25.
- [12] "Not just a flash in the pan." economist.com <https://www.economist.com/technology-quarterly/2006/03/11/not-just-a-flash-in-the-pan> (pristupljeno Jul. 2021)
- [13] T. Windbacher, "Engineering Gate Stacks for Field-Effect Transistors," Ph.D. dissertation, Tech. Univ. of Vienna, Vienna., Austria, 2010. [Online]. Dostupno na: <https://www.iue.tuwien.ac.at/phd/windbacher/node14.html>
- [14] *SPI Block Guide*, V04.01, Motorola Inc, 2000
- [15] *CS498xx Family System Designers Guide*, Cirrus Logic, Inc, 2016