

VILNIAUS UNIVERSITETAS  
FIZIKOS FAKULTETAS  
KIETO KŪNO ELEKTRONIKOS KATEDRA

Mindaugas Kurmauskas

CORTEX R4 ARCHITEKTŪROS MIKROVALDIKLIO SAVYBIŲ TYRIMAS

Pagrindinių studijų baigiamasis darbas

(studijų programa – TAIKOMOJI FIZIKA)

Studentas

Darbo vadovas

Recenzentas

Katedros vedėjas

Mindaugas Kurmauskas

dr. Mindaugas Vilūnas

dr. Recenzentas

dr.(HP) Kęstutis Arlauskas

Vilnius 2013

# Turinys

<b>Įvadas</b>	<b>3</b>
<b>1 ARM architektūra</b>	<b>4</b>
1.1 ARM mikrovaldiklių architektūros šeimos	5
<b>2 Naudojamos įrangos ir programos</b>	<b>6</b>
2.1 xRM48L950 aprašas	7
2.1.1 Procesorius	7
2.1.2 Atmintys	8
2.1.3 Trūkių sistema	8
2.1.4 Prievadai	9
2.2 STM32F4 aprašas	9
2.2.1 Procesorius	9
2.2.2 Atmintys	9
2.2.3 Trūkių sistema	9
2.2.4 Prievadai	10
<b>3 Ka kas ir kaip</b>	<b>11</b>
<b>4 Naudoti testavimo algoritmai ir jų aprašai</b>	<b>11</b>
4.1 Matematinų funkcijų skaičiavimo greičių įvertinimo algoritmai	11
4.1.1 Slankaus kablelio algoritmai	11
4.1.2 Fiksuoto tikslumo algoritmas	12
4.1.3 Loginių funkcijų testavimo algoritmas	13
4.1.4 Greitas diskretinis Furjė eilutės transformavimo testas	13
4.2 Įėjimo ir išėjimo į trūkį algoritmo aprašymas	15
4.3 Srovės matavimai	15
4.4 Radiacinis matavimas	17
<b>5 Rezultatai</b>	<b>19</b>
<b>6 Išvados</b>	<b>23</b>
<b>Literatūros sąrašas</b>	<b>24</b>
<b>Santrauka</b>	<b>25</b>

# Įvadas

Egzistuoja daug užduočių kurioms atlikti reikia didelio patikimumo. Jų atlikimui galima naudoti standartinius mikrovaldiklius, kombinuojant su programinėmis sprendimais. Tokiais kaip WDT (watchdog timer), kuris prižiūri pagrindinę programą, kad ji tinkamai atliktų savo darbą. Nuolatiniiais testais patikrinti duomenų skaičiavimo patikimumą, pavyzdžiui atlikti algoritmą du kartus ir tikrinti ar atsakymai sutampa. Daugeliu atveju visos papildomos priemonės užtikrinti patikimumą prailgina duomenų skaičiavimo laiką.

Sistemos kurioms reikalingas didelis patikimumas ir darbas realiaame laike labai praverčia sistemos dubliavimas su tikrinimo funkcijomis. Klaidos tikimybė ne visais atvejais yra tolydi laike, taigi klaidos gali tuo pačiu metu įvykti abiejose sistemose. Tokiu atveju naudinga, kai sistemos dirba pastumtos laike atžvilgiu viena kitos.

Yra daug aplinkų kur gali reikėti tokių sistemų. Pramoninės saugumo reikalaujančios aplinkos, tokios kaip:

- automobilių stabdymo sistemos, tokios kaip ratų anti blokavimo sistema (ABS)
- elektros gaminimo ir paskirstymo sistemos.
- liftai ir eskalatoriai

Medicinoje:

- defibriliatoriai
- radiacijos terapija
- robotizuotos operacijos

Atsižvelgiant į šį poreikį kompanija Teksas Instruments neseniai išleido ARM Cortex-R4 mikrovaldiklių šeimą kodiniu pavadinimu Hercules. Mums kilo klausimas koks yra šių mikrovaldiklių energetinis efektyvumas, bei mikrovaldiklių veikimas esant ekstramaliu poveikiui, ką šiame darba ir bandoma išsiaiškinti.

# 1 ARM architektūra

ARM architektūra paremta RISC<sup>1</sup> - paprastesnių komandų sistema:

- daug bendrosios paskirties registru
- paprastai tiesiogiai dirbama su registrais, o ne su atmintimi
- paprasti adresavimo režimai
- vienodo fiksuoto ilgio instrukcijos

Tai pat ARM architektūra papildomai turi:

- tiesioginę ALU<sup>2</sup> - aritmetinio loginio ir postūmio įrenginių kontrolę
- automatiškai didėjančius ir automatiškai mažėjančius adresavimo režimus ciklo optimizavimui
- iš karto kelių instrukcijų įkėlimo ir paėmimo komandas duomenų pralaidumo optimizavimui
- beveik visų komandų sąlyginį vykdymą vykdymo pralaidumui padidinti
- Thumb instrukcijų praplėtimą, kuris leidžia vykdyti 16bitų komandas vietoje 32bitų. Tai leidžia padidinti kodo tanki neilginant komandų atlikimo trukmės

Paprastai kiekviena instrukcija atliekama per vieną taktą, tad yra nesunku nuspėti programos vykdymą ir galima nesunkiai pritaikyti kaskadinį komandų vykdymą. Komandų atkodavimas reikalauja mažiau tranzistorių, nei mikrovaldikliams paremtiems sudėtingesnių komandų sistema (pvz. CISC<sup>3</sup>) - kur viena komanda yra kelių instrukcijų rinkinys. Dabartiniai ARM mikrovaldikliai yra 32bitų instrukcijos ilgio, bei 32bitų adresavimo. Yra paruoštos ir 64bitų virsijos.

ARM turi 31 bendrosios paskirties 32bitų ilgio registrus. Bet kuriuo metu galima matyti 16 jų. Kiti registrai yra naudojami išimčių apdorojimui pagreitinti. Visos instrukcijos skirtos registrams gali naudoti bet kurį iš 16 registru. Iš 16 visad matomų registru keli turi specializuotas funkcijas:

- Rietuvės rodyklė (ang. stack pointer) - paprastai 13 registras.
- Nuorodos registras (ang. link register) - 14 registras naudojamas kaip adresas į kurį einama funkcijos iškvietimui pasibaigus. 14 registras gali būti naudojamas kaip bendrosios paskirties registras.
- Programos skaitiklis (ang. program counter) - 15 registras. Naudojamas kaip nuoroda į adresą sekančios instrukcijos.

---

<sup>1</sup>RISC - reduced instruction set computing.

<sup>2</sup>ALU - arithmetic logic unit

<sup>3</sup>CISC - complex instruction set computing.

Likę 13 registų specialių funkcijų neatlieka.

ARM palaiko 7 rūšių išimtis ir privilegijuotą apdorojimo režimą kiekvienai iš jų:

- reset
- bandymas atlikti nežinomos instrukcijos
- programos pertraukties instrukcijos (SWI - software interrupt instruction). Gali būti naudojamos iškviesti operacinę sistemą
- išankstinio duomenų paėmimo atšaukimas (ang. prefetch abort)
- normalus trūkis (IRQ - interrupt request)
- greitas trūkis (FIQ - fast interrupt request)

Įvykus išimčiai kaikurie standartiniai registrai yra pakeičiami registrais specifiniais tai išimčiai. 14-tame registre laikomas adresas į kurį grįžtama po išimties įvykdymo. 13 registras yra išsaugomas, kad išimtis galėtų naudotis savo rietuve, netrigdydama tolimesnio programos darbo. Greito trūkio režime yra išsaugomi registrai nuo 8 iki 12, kad nereikėtų jų saugoti atskirai ir būtų galima jais greitai naudotis.

Įvykus išimčiai šerdis sustabdo programos vykdymą apibrėžtu būdu ir pereina prie išimties vykdymo į vieną iš fiksuotų adresų atmintyje, žinomų kaip išimties vektoriai (ang. exception vector). Yra atskiras vektorius kiekvienai išimčiai, įskaitant ir reset. Veikimas yra apibrėžtas normaliam programos darbui ir klaidų radimo režimui (ang. debug mode).

Visa branduolio būklė yra laikoma statuso registre vadinamu dabartinės programos statuso registru (CPSR - current program status register), kuriame yra:

- keturios aritmetinės vėlevėlės (neigiamas, nulis, perkelti ir perpildymo)
- 2 bitai trūkių išjungimui
- 5 bitai procesoriaus darbo režimui
- 2 bitai nurodantys kuris instrukcijų rinkinys naudojamas: ARM, Thumb ar Jazelle (naujesniuose procesoriuose palaikomas JAVA instrukcijų rinkinys)

## 2 Naudojamos įrangos ir programos

Naudojamas TMDXRM48USB maketas. Jame yra:

- Mikrovaldiklis xRM48L950AZWTT:
  - Du 32-bitu ARM Cortex-R4F procesoriai, veikiantys kartu<sup>4</sup>
  - 3MB flash, 256kB RAM

---

<sup>4</sup>lockstep - kartu atlieka tas pačias komandas

- Šerdies taktinis dažnis 200MHz
- fiksuoto tikslumo (32bitu) ir dvigubo tikslumo (64bitu) slankaus kablelio aritmetikos modulis
- Integruotas XDS100v2 emuliatorius programavimui per usb
- LED'ai, temperatūros sensorius, šviesos sensorius, akselerometras

Palyginimui pasirinktas STM32F4DISCOVERY maketas. Jis pasirinktas dėl

- mikrovaldiklis STM32F407VGT:
  - 32-bitu ARM Cortex-M4F procesorius
  - 1MB flash, 192 kB RAM
  - taktinis dažnis iki 168MHz
  - fiksuoto tikslumo (32bitu) ir slankaus (32bitu) kablelio aritmetikos modulis
- Integruotas ST-LINK/V2 emuliatorius programavimui per usb
- LED'ai, akselerometras, skaitmeninis mikrofonas
- CS43L22- SAK garsui su integruotu D klasės garso stiprinimu

Pasirinktas kompiliatorius - IAR 6.4. Darbo pradėjimo metu vienintelis palaikė abu maketus be papildomos įrangos.

## 2.1 xRM48L950 aprašas

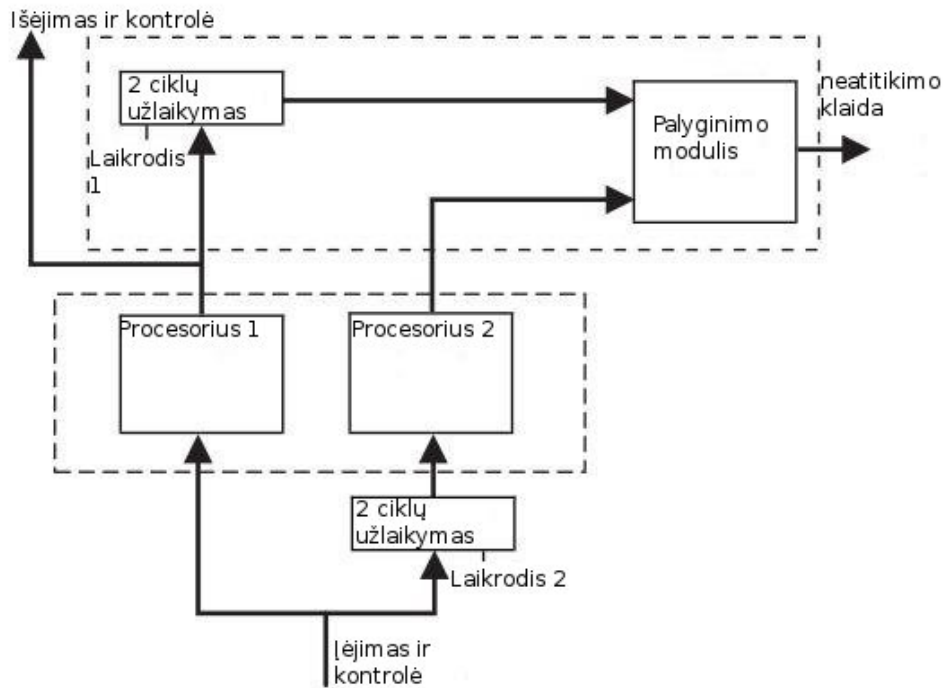
### 2.1.1 Procesorius

xRM48L950 turi dvi integruotas 32-bitų RISC ARM Cortex-R4F šerdis, su slankaus kablelio aritmetikos moduliu. Šerdys išdestyti skirtinga orientacija



1 pav. Šerdys pakreiptos fiziškai viena kitos atžvilgiu korpuse

Taip pat taktinis dažnis procesoriams paduodamas su 2 taktų užlaikymo skirtumu. Procesoriai fiziškai atskirti 100μm. atstumu. Procesorių išėjimo signalai palyginami atskirame modulyje.



2 pav. Dviejų branduolių įgyvendinimas

Įrenginys palaiko savęs testavimą. Jo metu galima priversti branduolių išėjimo signalų nesutapimą, taip patikrinama ar veikia branduolių palyginimo modulis ir ar įrenginys sugeba atpažinti klaidas. Šerdys maitinama 1.2V įtampa.

### 2.1.2 Atmintys

Įrenginys palaiko little-endian (LE32) formatą. Tai reiškia, kad atmintyje jauniausias bitas yra saugomas pirmas. SRAM ir Flash atminys turi ECC<sup>5</sup> apsaugą - galimybė 1 bito klaidos aptikimui ir pataisymui, bei 2 bitų klaidos aptikimui 8bitų bloke. Flash atmintis maitinama 3.3V įtampa. Esant kaskadiniui režimui flash atmintis gali veikti iki 200MHz taktiniu dažniu. SRAM atmintį galima nuskaityti arba įrašyti 1 ciklu nepriklausomai nuo režimo.

### 2.1.3 Trūkių sistema

Vektorinis trūkių valdiklis įgalina veiksmų prioritizavimą ir kontrolę. Trūkis - tai paprogramės iškviestas, galimai nutraukiant esamą procesoriaus veiklą. Paprastai toks įvykis reikalauja greito procesoriaus atsako. Procesorius iš normalios programos persōka į trūkio aptarnavimo paprogramę. RM48 palaiko 96 skirtingus programiškai reguliuojamo prioriteto trūkius. Yra 2 trūkio vektoriai - normalūs trūkiai (IRQ<sup>6</sup>) ir greitieji trūkiai (FIQ<sup>7</sup>). Greitieji trūkiai turi didesnę prioriteta už normalius ir yra nemaskuojami t.y. juos įjungus jų neimanoma išjungti ar pakeisti, nebent atlikus sistemos perkrovimą, kai visi trūkiai yra automatiškai išjungiami. Greitieji trūkiai gali pertraukti normalius.

<sup>5</sup>ECC - error corection code

<sup>6</sup>IRQ - interrupt request

<sup>7</sup>FIQ - fast interrupt request

#### 2.1.4 Prievadai

RM48L950 turi 2 laikmačių (N2HET<sup>8</sup>) koprosorius su tiesiogine magistralės prieiga (DMA) realaus laiko kontrolei. Jų pagalba galima atlikti įtampos impulso pločio moduliaciją (PWM), stebėti išvado loginį lygį, bei gali veikti kaip įvesties/išvesties periferija. Taip pat yra du integruoti 12 bitų rezoliucijos analogas kodas keitikliai. Prievadai maitinami 3.3V įtampa. Taip pat yra bendrosios paskirties įvesties išvesties prievadai. Palaiko šias periferijas:

- USB
- Ethernet
- UART
- MibSPI
- $I^2C$
- SCI
- DCAN

## 2.2 STM32F4 aprašas

### 2.2.1 Procesorius

ARM Cortex-M4F 32-bitų RISC procesorius, su slankaus kablelio aritmetikos moduliu. Palaiko DSP instrukcijas ir 32bitų slankaus kablelio duomenis.

### 2.2.2 Atmintys

Adaptivus realaus laiko atminties greitintuvas leidžia minimalų procesoriaus laukimą esant dideliems taktavimo dažniams nuskaitant ir įrašant į Flash atmintį. SRAM atmintis nuskaitom ir įrašoma per 1 taktą. Atminties apsaugos modulis riboja procesoriaus priejimą prie atminties, kad būtų galima išvengti netyčinių atminties sugadinimų. Tai ypač praverčia kai turima kritiškai svarbių duomenų ir juos reikia apsaugoti nuo kitų veikiančių procesų. Pavyzdžiui, jeigu yra veikianti operacinė sistema procesoriuje.

### 2.2.3 Trūkių sistema

Palaiko įdėtinę vektorinę trūkių kontrolę (NVIC<sup>9</sup>). Yra 16 prioritetų lygių, 82 maskuojami trūkių kanalai. Leidžia aukštesnio prioriteto trūkiams pertraukti vykdomus žemesnio prioriteto trūkius, minimaliai apkraunant procesorių.

---

<sup>8</sup>N2HET - next generation high end timer

<sup>9</sup>NVIC - nested vectored interrupt controller



#### 2.2.4 Prievadai

140 įėjimo išėjimo bendrosios paskirties prievadų, su trūkio generavimo galimybe. Periferijos greitis - iki 84MHz. Du 12 bitų skaitmeninis analogas keitikliai. Trys 12 bitų analogas kodas keitikliai. Palaikomos periferijos:

- $I^2C$
- SPI
- UART
- $I^2S$
- CAN
- USB 2.0
- Ethernet

### 3 Ka kas ir kaip

## 4 Naudoti testavimo algoritmai ir jų aprašai

### 4.1 Matematinų funkcijų skaičiavimo greičių įvertinimo algoritmai

Šiame paragrafe aprašytiems algoritmams buvo naudojamas laiko trūkis, kurio pagalba buvo skaičiuojamas algoritmo atlikimo greitis 1ms tikslumu. Abu maketai buvo nustatyti veikti 100MHz greičiu. Algoritmai buvo leisti 100 kartų ir fiksuoti užduoties atlikimo laikai suvidurkinti.

#### 4.1.1 Slankaus kablelio Gauss Legendre algoritmai

Slankaus kablelio Gauss Legendre algoritmas  $\pi$  skaičiavimui:

$$\begin{aligned}a_0 &= 1 \quad b_0 = \frac{1}{\sqrt{2}} \quad t_0 = \frac{1}{4} \quad p_0 = 1 \\a_{n+1} &= \frac{a_n + b_n}{2}, \\b_{n+1} &= \sqrt{a_n b_n}, \\t_{n+1} &= t_n - p_n(a_n - a_{n+1})^2, \\p_{n+1} &= 2p_n. \\\pi &\approx \frac{(a_n + b_n)^2}{4t_n}\end{aligned}\tag{1}$$

Naudojant dvigubo tikslumo kintamųjų testą buvo ieškomas 1000 narys, suskaičiuota konstanta nuo pasirinktosios skiriasi  $3.55271 * 10^{-15}$ .

```
int doubleTest() {
    volatile int i;
    volatile double an,bn,tn,pi;
    volatile double a,b,t,p;
    a = 1.0; b = 1/sqrt(2);
    t = 1/4; p = 1.0;
    for (i = 0; i < 1000; i++) {
        an = (a+b)/2;
        bn = sqrt(a*b);
        tn = t - p*(a-an)*(a-an);
        p *= 2;
        pi = (an+bn)*(an+bn)/(4*tn);
        a = an; b = bn; t = tn;
    }
}
```

```

    if ((pi - 3.14159265358979) <= 3.55271e-15)
        return 1; //testas atliktas sekmingai
    return 0; //teste ivyko klaida
}

```

Naudojant viengubo tikslumo slankaus kabelio kintamuosius buvo ieškomas 120 narys. Konstanta nuo pasirinktosios skiriasi  $8.74228 * 10^{-8}$

```

int floatTest() {
    volatile int i;
    volatile float an,bn,tn,pi;
    volatile float a,b,t,p;
    for(volatile int j = 0; j <9; j++) {
        a = 1.0; b = sqrt(0.5);
        t = 0.25; p = 1.0;
        for (i = 0; i < 120; i++) {
            an = (a+b)/2;
            bn = sqrt(a*b);
            tn = t - (p*(a-an)*(a-an));
            p *= 2;
            pi = (an+bn)*(an+bn)/(4*tn);
            a = an; b = bn; t = tn;
        }
    }
    if ((pi - 3.14159265358979) <= 8.74228e-8)
        return 1; //testas atliktas sekmingai
    return 0; //teste ivyko klaida
}

```

#### 4.1.2 Fiksuoto tikslumo algoritmas

Fiksuoto tikslumo algoritmas:

$$b = \sum_{i=0}^{100000} (i * (-1)^{i+1}) \quad (2)$$

```

void intTest() {
    volatile int a,b;
    a = 1;
    b = 0;
    for(volatile int i = 0; i < 100000; i++) {
        b += i * a;
        a *= -1;
    }
}

```

```

    }
}

```

#### 4.1.3 Loginių funkcijų testavimo algoritmas

Mikrovaldikliai atlieka logines AND, NOT, OR, XOR, bei bitų stumdymo funkcijas.

```

short testBits() {
    volatile unsigned sum = 0x55555555;
    // 0101 0101 0101 0101 0101 0101 0101 0101
    for (volatile int i = 0; i < 0x020000; i++) {
        sum = sum << 1;          /// sum = 0xAAAAAAAA
        sum &= 0x0000FFFF;       /// sum = 0x0000AAAA
        sum ^= 0xFFFFFFFF;       /// sum = 0xFFFF5555
        sum = sum << 16;         /// sum = 0x55550000
        sum = sum | (sum >> 16); /// sum = 0x55555555
        sum = ~sum;              /// sum = 0xAAAAAAAA
        sum = sum >> 1;          /// sum = 0x55555555
    }
    if (0x55555555 == sum)
        return 1; //testas ivykditas sekmingai
    return 0; // teste ivyko klaida
}

```

#### 4.1.4 Greitas diskretinis Furjė eilutės transformavimo testas

Naudotos ARM dsp<sup>10</sup> bibliotekos, atskirai optimizuotos R4 bei M4 mikrovaldikliams. Naudota 1024 kompleksinių taškų, kurie masyve išdėstyti vienas po kito ir bendras masyvo ilgis 2048. Slankaus kablelio testui masyvas užpildomas funkcija:

$$Re(z) = (\sin(v * i) + \cos(v * 7)), t = 0, 2, 4...2046$$

$$Im(z) = 0 \tag{3}$$

```

#define TEST_LENGTH_SAMPLES 2048
static float32_t testInput[TEST_LENGTH_SAMPLES];
static float32_t testOutput[TEST_LENGTH_SAMPLES/2];
uint32_t fftSize = 1024;
uint32_t ifftFlag = 0;
uint32_t doBitReverse = 1;

void prepareTestFloat(void) {

```

---

<sup>10</sup>Dsp - digital signal processing (skaitmeninis signalų apdorojimas)

```

for(volatile int t = 0; t < 2048; t+=2) {
    testInput[i] = sin(t*3) + cos(t*7);
    testInput[i+1] = 0;
}
}

int32_t testCfftFloat(void) {
    arm_status status;
    arm_cfft_radix4_instance_f32 S;
    float32_t maxValue;

    arm_cfft_radix4_init_f32(&S, fftSize, ifftFlag, doBitReverse);

    arm_cfft_radix4_f32(&S, testInput);

    arm_cmplx_mag_f32(testInput, testOutput, fftSize);

    arm_max_f32(testOutput, fftSize, &maxValue, &testIndex);

    if(testIndex == refIndex) {
        return 1;
    }
    return 0;
}

```

Fiksuoto tikslumo Furijė transformacijai naudoti ARM Q31 fiksuoto kablelio kintamieji. Iš slankaus kablelio paversti į Q31 formato kintamuosius naudota funkcija:

$$Q31(x) = ((int) ((x) * (float)(1 << 31))) \quad (4)$$

Masyvo užpildo funkcija:

$$Re(z) = Q31(1000 * (\sin(v * i) + \cos(v * 7))), \quad t = 0, 2, 4 \dots 2046$$

$$Im(z) = 0 \quad (5)$$

```

#define TEST_LENGTH_SAMPLES 2048
uint32_t fftSize = 1024;
uint32_t ifftFlag = 0;
uint32_t doBitReverse = 1;
static q31_t testInputQ[TEST_LENGTH_SAMPLES];
static q31_t testOutputQ[TEST_LENGTH_SAMPLES/2];
uint32_t refQIndex = 0;

```

```

void prepareTestQ31(void) {
    double skaicius=0;
    int nulis = (int) ((0 * (float)(1<<31)));
    for(volatile int32_t i = 0; i < 2048; i+=2) {
        skaicius = 1000*sin(i*3) + 1000*cos(i*7);
        testInputQ[i] = ((int)(skaicius * (float)(1<<31)));
        testInputQ[i+1] = nulis;
    }
}

int32_t testCfftQ31(void) {
    arm_status status;
    arm_cfft_radix4_instance_q31 S;
    q31_t maxValue;

    status = arm_cfft_radix4_init_q31(&S, fftSize, ifftFlag, doBitReverse);

    arm_cfft_radix4_q31(&S, testInputQ);

    arm_cmplx_mag_q31(testInputQ, testOutputQ, fftSize);

    arm_max_q31(testOutputQ, fftSize, &maxValue, &testIndex);

    if(testIndex == refIndex) {
        return 1;
    }
    return 0;
}

```

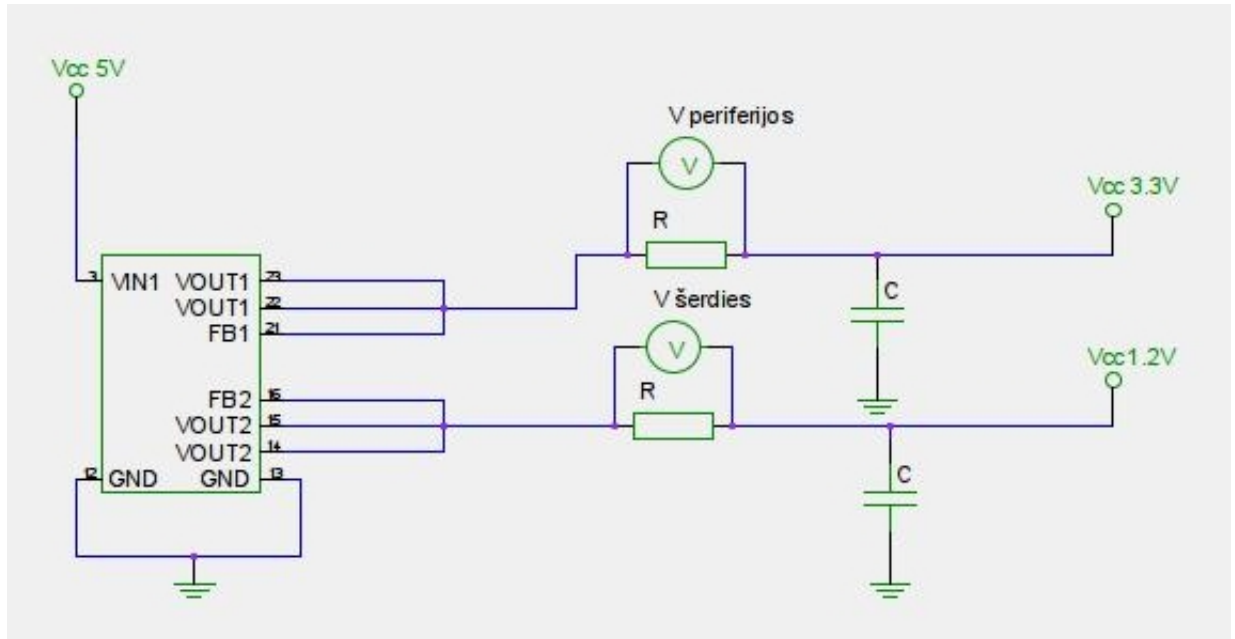
## 4.2 Įėjimo ir išėjimo į trūkį algoritmo aprašymas

Naudojami maketai buvo paleisti veikti 100MHz greičiu. Abu maketai keisdavo išėjimo prievado vieno iš išvadų loginį lygį, bei buvo įjungtas taimeris su tuščia paprograme. Osciloscopu matuotas išvadų lygmenų kitimo laikas.

## 4.3 Srovės matavimai

STM32F4DISCOVERY makete srovė matuota tiesiogiai. TMDXRM48USB maketas neturėjo trumpiklių tarp maitinimo ir mikrovaldiklio. Tarp maitinimo 3.3V išėjimo ir mikrovaldiklio išvadų maitinimo bei maitinimo 1.2V išėjimo ir mikrovaldiklio logikos maitinimo buvo įterp-

tos  $0.12\Omega$  varžos.



3 pav. Cortex-R4 srovės matavimo schema.  $R=0.12\Omega$ .

Visų matavimų metu išvadai buvo nustatyti išvesties režime ir nustatyti nuliniame loginiame lygijė. Buvo atlikti matavimai esant šioms konfigūracijoms:

- įjungtas stanby režimas. Siekta išmatuoti minimalų srovės sunaudojimą kai procesoriui nereikia atlikti darbo.
  - Cortex-R4:
    - \* Flash atmintis automatinio miego režime
    - \* procesoriai laukimo režime
    - \* taktuojama iš mažos galios išorinio kvarco
    - \* periferijos išjungtos
  - Cortex-M4:
    - \* vidinis 1.2V įtampos reguliatorius išjungtas
    - \* PLL<sup>11</sup>, vidinis 16MHz ir išorinis 8MHz taktiniai osciliatoriai atjungti. Veikia vidinis 32kHz taktinis osciliatorius
    - \* procesorius sustabdytas
    - \* mikrovaldiklis minimalioje srovės suvartojimo būsenoje.
    - \* režimą galima išeiti:
      - gavus išorinį impulsą į NRST arba WKUP išvadus
      - realaus laiko laikrodžiui (RTC) sugeneravus trūkį
- įjungtas negilaus miego režimas esant įvairiems taktavimo greičiams:

<sup>11</sup>PLL - phase lock loop dažnių generatorius

- Cortex-R4:
  - \* Flash atmintis automatinio miego režime
  - \* sistema veikia taktavimo dažniu
  - \* procesoriai laukimo režime
  - \* periferija atjungta
- Cortex-M4:
  - \* sistema veikia pll taktavimo dažniu
  - \* procesorius sustabdytas
  - \* iš režimo galima išeiti betkokio trūkio pagalba
- esant skirtingiems taktavimo dažniams mikrovaldikliai atlikinėjo slankaus kablelio ir fiksuoto tikslumo aritmetikos testus, bei buvo paleisti veikti cikle:

```
while(1);
```

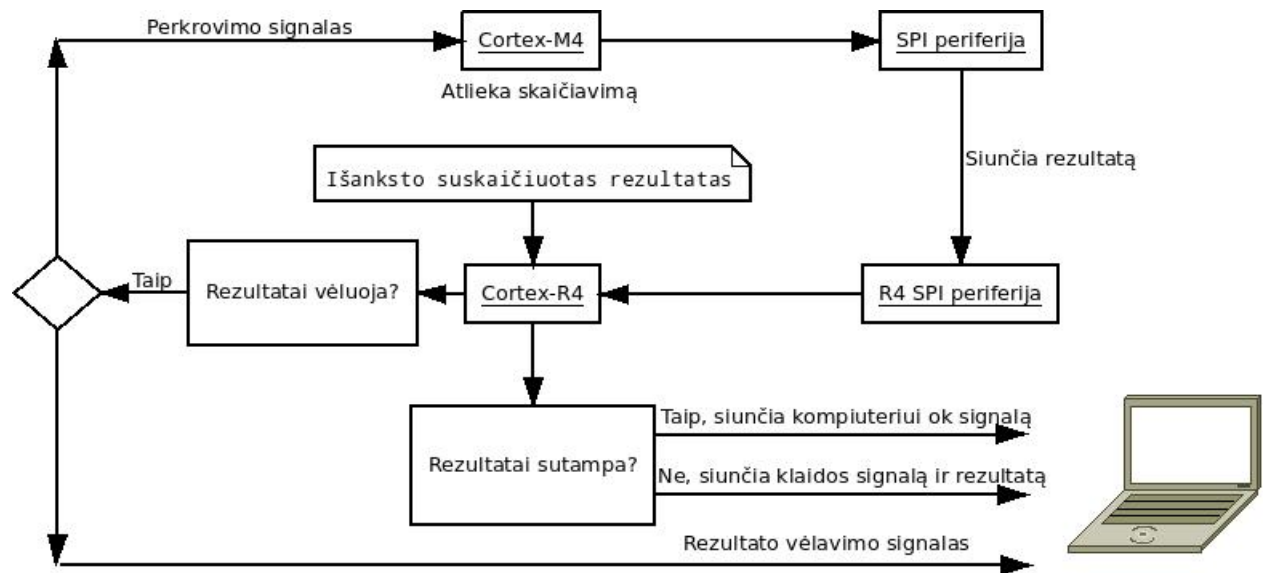
## 4.4 Radiacinis matavimas

Siekiant patikrinti mikrovaldiklio veikimą ekstremaliam poveikiui. Buvo iškelta hipotezė, kad apšviečiant jonizuojančiąją spinduliuotę gali būti sutrigdoma normali mikrovaldiklio veikla:

- pažeidžiant flash atmintį sutrigdoma programa, dėl ko mikrovaldiklis pradeda vykdyti neteisingas užduotis arba jų išvis nevykdyti
- pažeidžiant RAM atmintį gali būti iškraipomi skaičiavimų duomenys
- sutrigdant šerdies atliekamą darbą gali būti generuojamos skaičiavimų, logikos ir valdymo klaidos.

Cortex-M4 ir Cortex-R4 mikrovaldikliai sujungti 4 laidu + žemė spi protokolu. Cortex-M4 nustatytas kaip master, Cortex-R4 kaip slave. Cortex-M4 atlikinėjo matematinius skaičiavimus ir siuntė rezultatus į Cortex-R4 mikrovaldiklį patikrinimui, kuris per UART jungtį juos siuntė į kompiuterio terminalą.

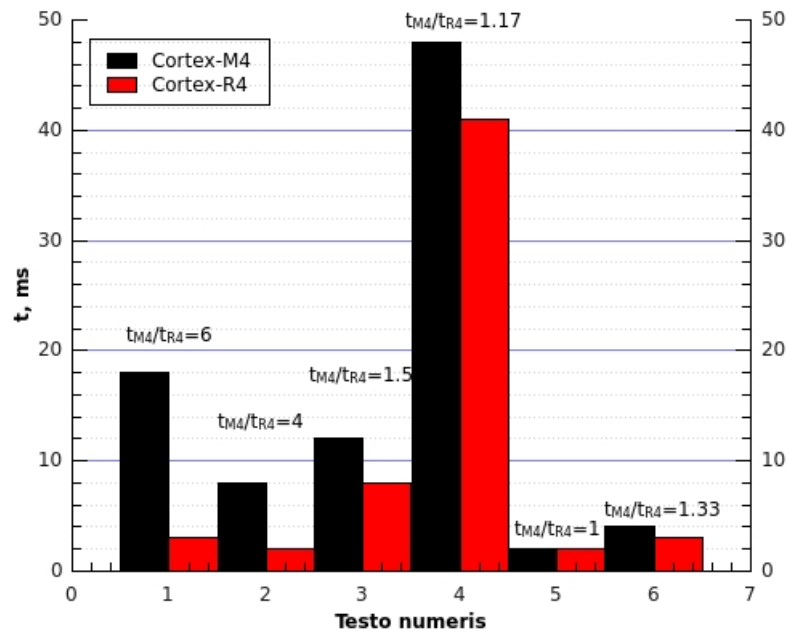




4 pav. Radiacinio matavimo logikos diagrama.

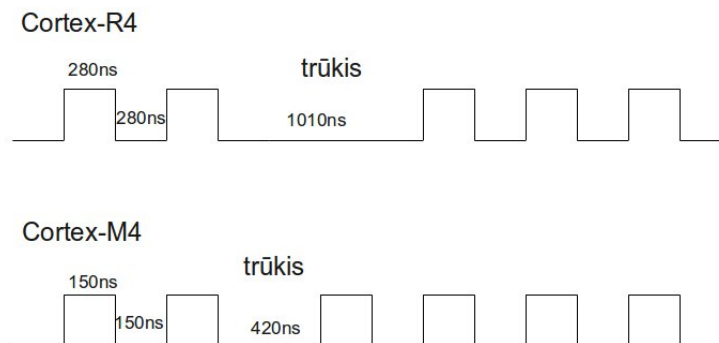
## 5 Rezultatai

Cortex-R4 mikrovaldiklis beveik visus skaičiavimo testus atliko greičiau nei cortex-M4 mikrovaldiklis esant 100MHz taktiniui dažniui.



5 pav. Skaičiavimo algoritmų laikai. 1 - dvigubo tikslumo slankaus kabelio testas ((2.1.1) algoritmas). 2 - slankaus kabelio testas ((2.1.1) antras algoritmas). 3 - fiksuoto tikslumo kintamųjų testas ((2.1.2) algoritmas). 4 - loginių funkcijų testas ((2.1.3) algoritmas). 5 - Furijė slankaus kabelio ((2.1.4) algoritmas). 6 - Furijė fiksuoto tikslumo kintamųjų testas((2.1.4) antras algoritmas).

Tirtas periferijos perjungimo greitis ir įėjimo išėjimo į tuščią trūkio paprogramę laikas esant 100MHz taktavimo dažniui. Cortex-R4 periferijos išėjimo lygmenį keitė kas 280ns. Įėjimo į trūkį ir išėjimo laikas 730ns. Skirtumo tarp IRQ ir FIQ trūkių laiko nebuvo. Cortex-M4 periferikos išvado loginį lygmenį keitė kas 150ns. Įėjimo į trūkį ir išėjimo laikas 200ns.



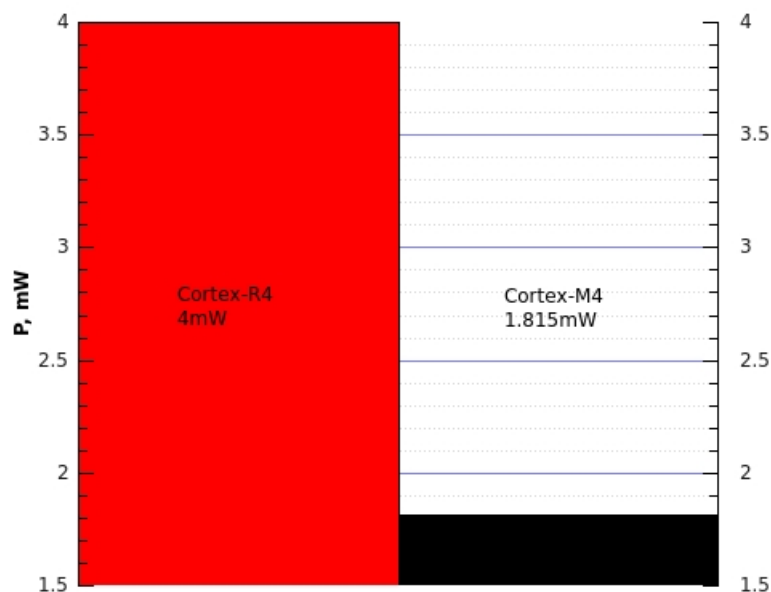
6 pav. Generuotų impulsų laikinė diagrama.

Mikrovaldiklio Cortex-R4 periferija tirta atskirai. Maitinama 3.3V.

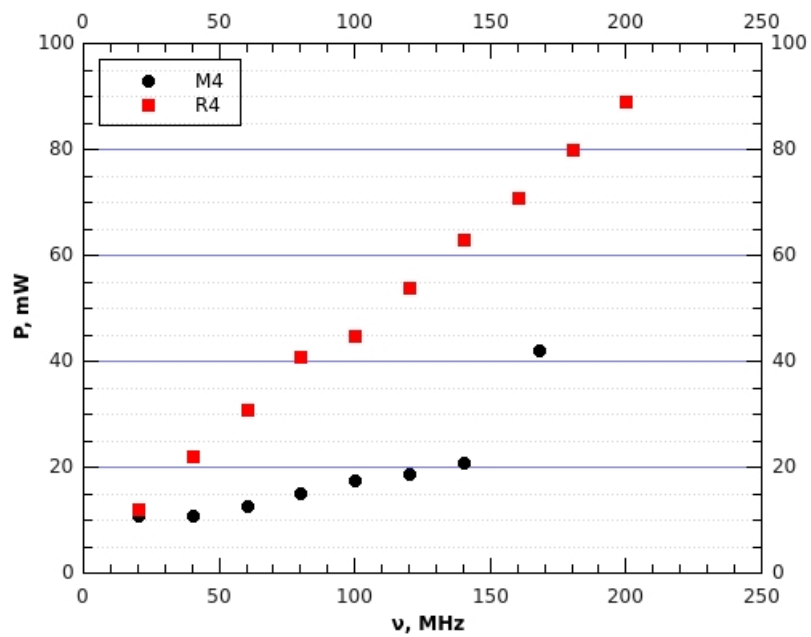
- Prievadai nustatyti žemame loginiame lygyje:

- Naudajama galia 371.5mW
- Periferija išjungta:
  - Periferijos išvaduose atsiranda loginis aukštas lygis:
    - \* SPI4nCS, SPI4nENA
    - \* DMMnENA
    - \* CAN3TX, CAN3RX
  - Naudojama galia 341mW
- Nurodomas mažos galios osciliatorius periferijų taktavimui. Periferija išjungiamą
  - Išjungiant periferiją turėtų būti automatiškai atjungiami ir osciliatoriai
  - Naudojama periferijos galia 283.25mW
  - (Procesoriaus naudojama galia minimali (4mW))
- Išjungiamos periferijos bei procesoriaus loginė dalis atsakinga už periferijos valdymą
  - Mikrovaldiklis nustoja vykdyti kitas komandas
  - Naudojama periferijos galia 374mW
  - Padidėja procesoriaus naudojama galia (16mW) lyginant su prieš tai atliktu matavimu.

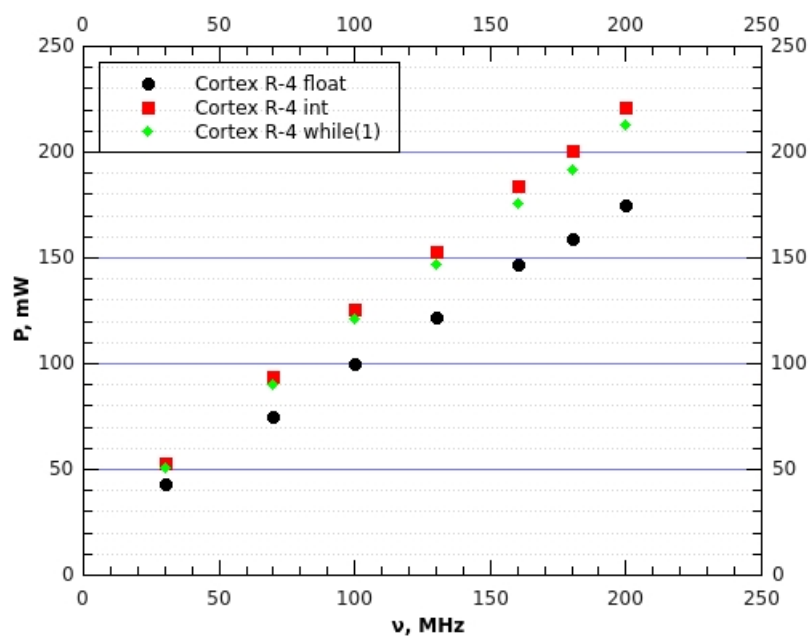
Mikrovaldiklio Cortex-R4 šerdis maitinama 1.2V. Kadangi 3.3V maitinama ir kiti įrenginiai esantys makete (tokie kaip šviesos sesnosrius, akselerometras), bei dėl galimos JTAG įtakos rezultatams naudojama galia buvo skaičiuojama pagal formulę  $P = I_{serdies} * 1.2V$ . Cortex-M4 maitinamas 3.3V. Skaičiuojant galią daryta prielaida, kad išjungus periferiją kitų įrenginių naudojama srovė yra apie 0mA. Cortex-M4 naudojama galia  $P = I * 3.3V$ .



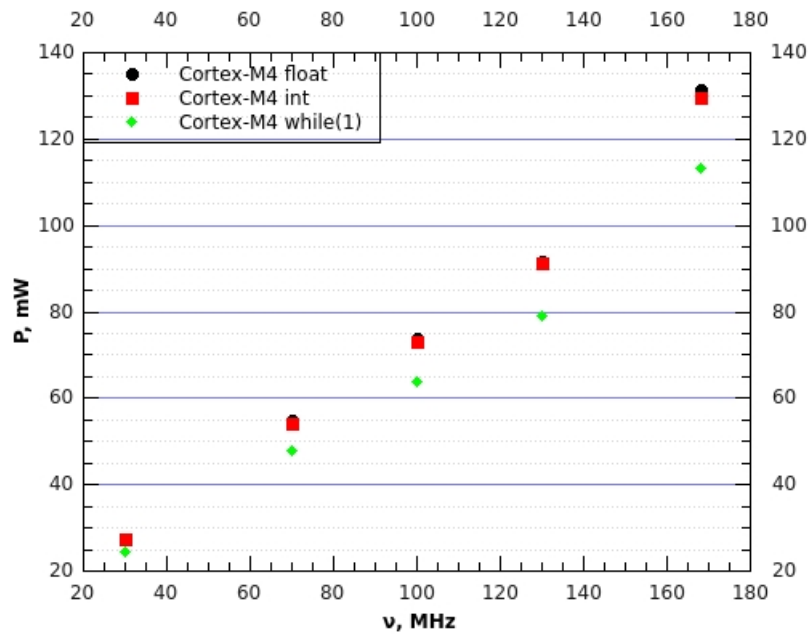
7 pav. Mikrovaldikliai stanby režime.



8 pav. Mikrovaldiklių galios vartojimo priklausomybė nuo taktinio greičio sleep režime.



9 pav. RM48 mikrovaldiklio naudojamos galios priklausomybė nuo taktinio dažnio atliekant slankaus kabelio, fiksuoto tikslumo ir būnant tuščiam cikle.



10 pav. *STM32F4* mikrovaldiklio naudojamos galios priklausomybė nuo taktinio dažnio atliekant slankaus kabelio, neslankaus kabelio ir būnant tuščiam cikle.

Mikrovaldiklis Cortex-M4 buvo 1h švitinamas 1.176MeV gama spinduliais ( $^{137}\text{Cs}$ ). Jokių nukrypimų nuo normalaus mikrovaldiklio darbo neužfiksuota. Pereita prie švitinimo rentgeno spinduliuote. Mikrovaldiklis buvo padėtas po rentgeno lempa kurios įtampa bei srovė didinta laipsniškai. Prie 19kV įtampa ir 19mA per 15min mikrovaldiklis nustojo veikti, tačiau iki nustojimo veikti visus skaičiavimus atliko gerai. Mikrovaldiklis per 3h atsistatė ir vėl pradėjo atlikinėti skaičiavimus.

## 6 Išvados

1. Cortex-R4, esant 100MHz taktiniui dažniui, yra žymiai greitesnis atliekant dvigubo tikslumo slankaus kablelio skaičiavimus nei cortex-M4 mikrovaldiklis.
2. Cortex-R4 trūkių sistema lyginant su Cortex-M4 yra labai lėta ir neefektyvi.
3. Mikrovaldikliams atliekant skirtingus skaičiavimus naudojama galia labiausiai priklauso nuo taktinio dažnio ir tik nežymiai nuo procesoriaus apkrovos skaičiavimais.
4. RM48 mikrovaldiklio periferija taisiklyngai neišsijungia.
5. Su prieinamais jonizuojančios spinduliuotės šaltiniais atliekant matavimus net su vienguba šerdimi nepavyko pasiekti duomenų pažeidimų programos vykdymo metu. Keliant intensyvumą prasidėjo išliekantys procesoriaus funkcionalumo pažeidimai.

# Literatūros sąrašas

1. Q formatai ir konvertavimas.

[http://infocenter.arm.com/help/topic/com.arm.doc.dai0033a/DAI0033A\\_fixedpoint\\_appsnote.pdf](http://infocenter.arm.com/help/topic/com.arm.doc.dai0033a/DAI0033A_fixedpoint_appsnote.pdf)

2. TMDXRM48USB trumpas aprašas.

[http://processors.wiki.ti.com/images/5/5c/RM48\\_USB\\_QUICK\\_START.pdf](http://processors.wiki.ti.com/images/5/5c/RM48_USB_QUICK_START.pdf)

3. Detalus Cortex-R4 aprašas.

<http://www.ti.com/lit/ds/spns177a/spns177a.pdf>

4. Cortex-R4 energijos vartojimo optimizavimo aprašas.

<http://www.ti.com/lit/an/spna172a/spna172a.pdf>

5. Cortex-R4 slankaus kabelio modulio savybių aprašas.

<http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dai0240a/index.html>

6. Cortex-R4 silicon errata.

<http://www.ti.com/lit/er/spnz194b/spnz194b.pdf>

7. STM32F4-Discovery trumpas aprašas.

[http://www.st.com/internet/com/TECHNICAL\\_RESOURCES/TECHNICAL\\_LITERATURE/DATA\\_BRIEF/DM00037955.pdf](http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/DATA_BRIEF/DM00037955.pdf)

8. Detalus Cortex-M4 mikrovaldiklio aprašas.

[http://www.st.com/internet/com/TECHNICAL\\_RESOURCES/TECHNICAL\\_LITERATURE/DATASHEET/DM00037051.pdf](http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/DATASHEET/DM00037051.pdf)

9. Cortex-M4 slankaus kabelio modulio savybių aprašas.

[http://www.st.com/internet/com/TECHNICAL\\_RESOURCES/TECHNICAL\\_LITERATURE/DATASHEET/DM00037051.pdf](http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/DATASHEET/DM00037051.pdf)

10.  $\pi$  skaičiavimo algoritmas.

[http://en.wikipedia.org/wiki/Gauss-Legendre\\_algorithm](http://en.wikipedia.org/wiki/Gauss-Legendre_algorithm)

# Santrauka

## „CORTEX R4 ARCHITEKTŪROS MIKROVALDIKLIO SAVYBIŲ TYRIMAS”

Užduotys kurioms reikia didelio patikimumo reikalauja skaičiavimų patikrinimo. Cortex-R4 turi integruotą skaičiavimo ir kontrolės dubliavimą, tačiau neaišku kaip tai yra energetiškai naudinga.

Šiame darbe buvo pabandyta išsiaiškinti Hercules šeimos atstovo RM48 mikrovaldiklio energetinį efektyvumą, bei veikimo ypatybes esant jonizuojančiai spinduliutei.