

**VILNIAUS UNIVERSITETAS  
FIZIKOS FAKULTETAS  
KIETO KŪNO ELEKTRONIKOS KATEDRA**

Mindaugas Kurmauskas

**CORTEX R4 MIKROVALDIKLIO ARCHITEKTŪROS TYRIMAS**

Pagrindinių studijų kursinis darbas

(studijų programa – TAIKOMOJI FIZIKA)

Studentas

Darbo vadovas

Katedros vedėjas

Mindaugas Kurmauskas

dr. Mindaugas Vilūnas

dr.(HP) Kęstutis Arlauskas

Vilnius 2013

# Turinys

<b>Įvadas</b>	<b>3</b>
<b>1 Naudojamos įrangos ir programos</b>	<b>4</b>
1.1 xRM48L950 aprašas	4
1.1.1 Procesorius	4
1.1.2 Atmintys	5
1.1.3 Trūkių sistema	5
1.1.4 Prievadai	6
1.2 STM32F4 aprašas	6
1.2.1 Procesorius	6
1.2.2 Atmintys	6
1.2.3 Trūkių sistema	7
1.2.4 Prievadai	7
<b>2 Naudoti testavimo algoritmai ir jų aprašai</b>	<b>8</b>
2.1 Matematinių funkcijų skaičiavimo greičių įvertinimo algoritmai	8
2.1.1 Slankaus kabelio algoritmai	8
2.1.2 Fiksuoto tikslumo algoritmas	9
2.1.3 Loginių funkcijų testavimo algoritmas	10
2.1.4 Greitas diskretinis Furjė eilutės transformavimo testas	10
2.2 Įėjimo ir išėjimo į trūkį algoritmo aprašymas	12
2.3 Srovės matavimai	12
<b>3 Rezultatai</b>	<b>14</b>
<b>4 Išvados</b>	<b>17</b>
<b>Literatūros sąrašas</b>	<b>18</b>
<b>Santrauka</b>	<b>19</b>

# Įvadas

Egzistuoja daug užduočių kurioms atlikti reikia didelio patikimumo. Jų atlikimui paprasčiausiu atveju galima naudoti standartinius mikrovaldiklius, kombinuojant su programinėmis gudrybėmis. Tokiomis kaip WDT (watchdog timer), kurios prižiūri pagrindinę programą, kad ji tinkamai atliktų savo darbą. Nuolatiniais 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. Mum kilo klausimas koks yra šių mikrovaldiklių energetinis efektyvumas, ką šiame darbe ir bandoma išsiaiškinti.

Tolimesniame darbe būtų idomu patikrinti veikimą esant ekstremaliam poveikiui, galimybę panaudoti vieną iš Hercules šeimos atstovų RM48 kosmose, kur jo užduotis galėtų būti navigacija, kuriai atlikti reikia slankaus formato ir skaičių tikslumo.

# 1 Naudojamos įrangos ir programos

Naudojamas TMDXRM48USB maketas. Jame yra:

- Mikrovaldiklis xRM48L950AZWTT:
  - Du 32-bitu ARM Cortex-R4F procesoriai, veikiantys kartu<sup>1</sup>
  - 3MB flash, 256kB RAM
  - greitis iki 200MHz
  - neslankaus(32bitu) ir dvigubo tikslumo (64bitu) slankaus kablelio aritmetikos modulis
- Integruotas XDS100v2 emuliatorius programavimui per usb
- LED'ai, temperatūros sensorius, šviesos sensorius, akselerometras
- prožektoriukas

Palyginimui naudojamas STM32F4DISCOVERY maketas:

- mikrovaldiklis STM32F407VGT:
  - 32-bitu ARM Cortex-M4F procesorius
  - 1MB flash, 192 kB RAM
  - greitis iki 168MHz
  - neslankaus (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 klases garso stiprinimu

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

## 1.1 xRM48L950 aprašas

### 1.1.1 Procesorius

xRM48L950 turi du integruotus 32-bitų RISC ARM Cortex-R4F procesorius, su slankaus kablelio aritmetikos moduliu. Procesoriai išdestyti skirtinga orientacija

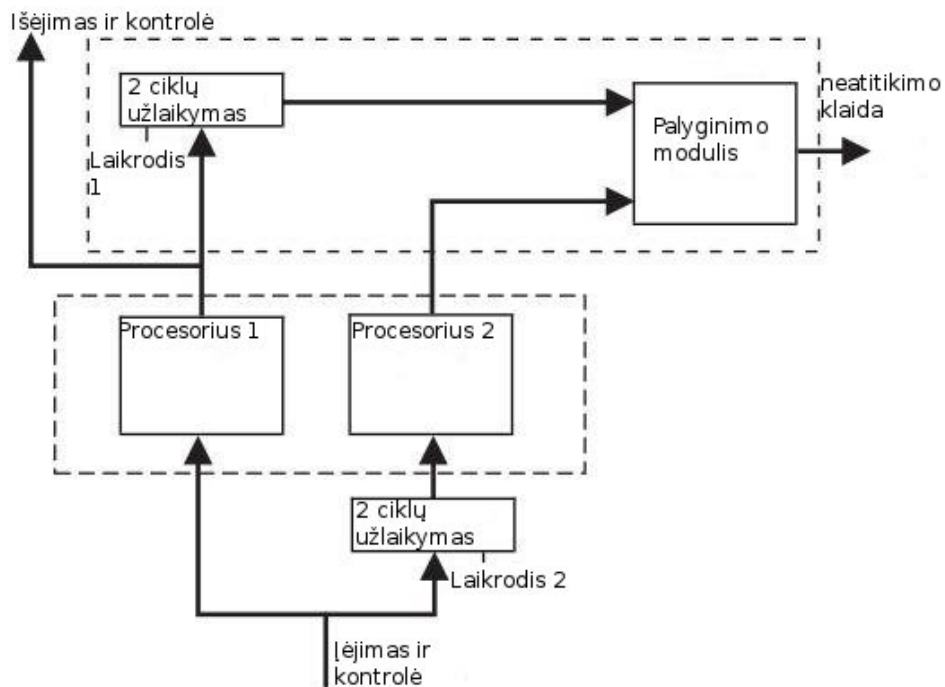
---

<sup>1</sup>lockstep - kartu atlieka tas pačias komandas



1 pav. Procesoriai pakreipti fiziškai vienas kito atžvilgiu korpuse

Taip pat taktinis dažnis procesoriams paduodamas su 2 taktų užlaikymo skirtumu. Procesorių išėjimo signalai palyginami atskirame modulyje.



2 pav. Dviejų branduolių įgyvendinimas

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

### 1.1.2 Atmintys

Įrenginys palaiko little-endian (LE32) formatą. Tai reiškia, kad atmintyje jauniausias bitas yra saugomas pirmas. SRAM ir Flash atminys turi ECC<sup>2</sup> apsaugą - galimybė 1 bito klaidos aptikimui ir pataisymui, bei 2 bitų klaidos aptikimui. 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.

### 1.1.3 Trūkių sistema

Vektorinis trūkių valdiklis įgalina veiksmų prioritizavimą ir kontrolę. Trūkis - tai programės iškvietimas, galimai nutraukiant esamą procesoriaus veiklą. Paprastai toks įvykis

<sup>2</sup>ECC - error corection code

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 trūčiai (IRQ<sup>3</sup>) ir greitieji trūčiai (FIQ<sup>4</sup>). Greitieji trūčiai turi didesnę prioritetą už normalius ir yra nemaskuojami t.y. juos įjungus jų neimanoma išjungti ar pakeisti, nebent atlikus sistemos perkrovimą, kai visi trūčiai yra automatiškai išjungiami. Greitieji trūčiai gali pertraukti normalius.

#### 1.1.4 Prievadai

RM48L950 turi 2 laikmačių (N2HET<sup>5</sup>) koprosorius su tiesiogine magistralės prieiga (DMA) realaus laiko kontrolei. Jų pagalba galima generuoti skirtingo pločio impulsus (PWM), stebėti išvado loginį lygį, bei gali veikti kaip įvesties/išvesties periferija. Taipat yra du 12 bitų rezoliucijos analogas-skaitmeninis keitikliai su išviso 24 kanalais. Prievadai maitinami 3.3V įtampa. Taipat yra bendrosios paskirties įvesties išvesties prievadai. Palaiko šias periferijas:

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

## 1.2 STM32F4 aprašas

### 1.2.1 Procesorius

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

### 1.2.2 Atmintys

Adaptyvus realaus laiko atminties greitintuvas leidžia minimalų procesoriaus laukimą esant dideliems taktavimo dažniams nuskaitant ir įrašant į Flash atmintį. SRAM atmintis nuskaitom ir įrašoma 1 taktinio dažnio greičiu. 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.

---

<sup>3</sup>IRQ - interrupt request

<sup>4</sup>FIQ - fast interrupt request

<sup>5</sup>N2HET - next generation high end timer

### 1.2.3 Trūkių sistema

Palaiko įdėtinę vektorinę trūkių kontrolę (NVIC<sup>6</sup>). Yra 16 prioritetų lygių, 82 maskuojami trūkių kanalai. Leidžia aukštesnio prioriteto trūkiams pertraukti vykdomus žemesnio prioriteto trūkius, minimaliai apkraunant procesorių.

### 1.2.4 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 skaitmeninis keitikliai su išviso 24 kanalais. Palaikomos periferijos:

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

---

<sup>6</sup>NVIC - nested vectored interrupt controller

## 2 Naudoti testavimo algoritmai ir jų aprašai

### 2.1 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 paskaičiuotas laikas suvidurkintas.

#### 2.1.1 Slankaus kabelio Gauss Legendre algoritmai

Slankaus kabelio 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 sėkmingai
    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
}
```

### 2.1.2 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;
    }
}
```

### 2.1.3 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
}
```

### 2.1.4 Greitas diskretinis Furjė eilutės transformavimo testas

Naudotos ARM dsp<sup>7</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) {
    for(volatile int t = 0; t < 2048; t+=2) {
        testInput[i] = sin(t*3) + cos(t*7);
        testInput[i+1] = 0;
```

---

<sup>7</sup>Dsp - digital signal processing (skaitmeninis signalų apdorojimas)

```

    }
}

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;
}

```

## 2.2 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. Osciloskopu matuotas išvadų lygmenų kitimo laikas.

## 2.3 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 įterptos  $0.12\Omega$  varžos. Visų matavimų metu išvadai buvo nustatyti išvesties režime ir nustatyti nuliniame loginiame lygije. Buvo atlikti matavimai esant šioms konfigūracijoms:

- įjungtas stanby režimas:

- Cortex-R4:
  - \* išjungta Flash atmintis
  - \* procesoriai neatlieka jokių skaičiavimų
  - \* taktavimo dažnis 20kHz
- Cortex-M4:
  - \* vidinis 1.2V įtampos reguliatorius išjungtas
  - \* PLL<sup>8</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 negilus miego režimas esant įvairiems taktavimo greičiams:
  - Cortex-R4:
    - \* sistema veikia taktavimo dažniu
    - \* procesorius neatlieka skaičiavimų
  - Cortex-M4:
    - \* sistema veikia pll taktavimo dažniu
    - \* procesorius sustabdytas (CPU)
    - \* 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:

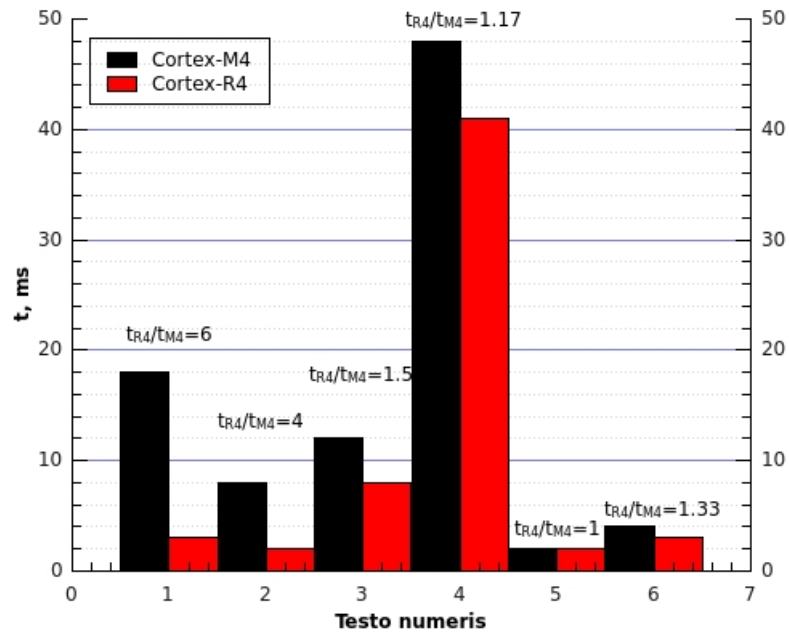
```
while(1);
```

---

<sup>8</sup>PLL - phase lock loop dažnių generatorius

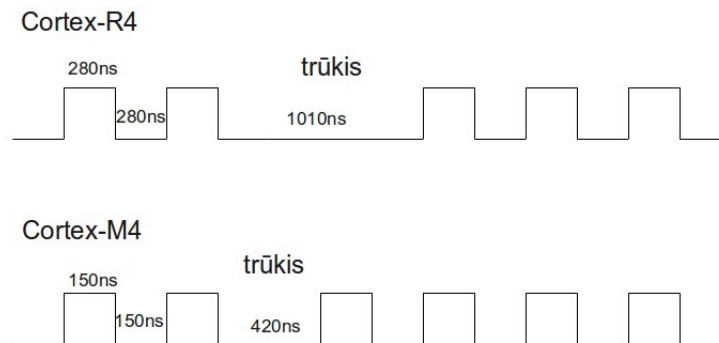
### 3 Rezultatai

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



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

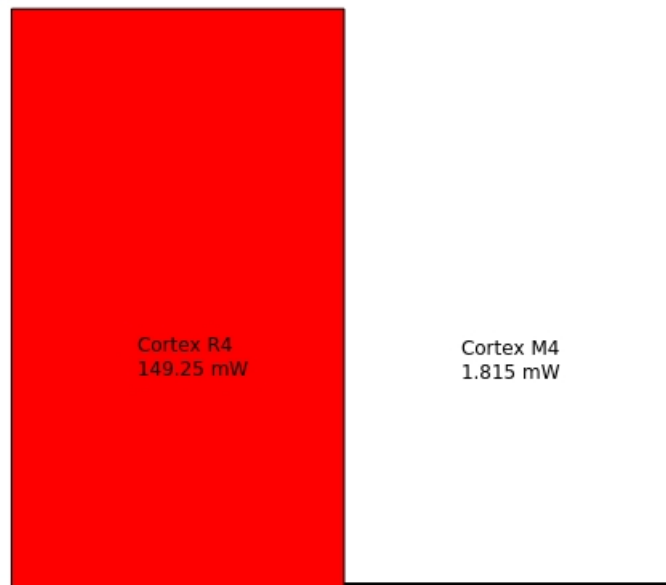
Tirtas periferijos perjungimo greitis ir įėjimo išėjimo į tuščią trūkio paprogramę laikas esant 100MHz taktavimo dažniui. Cortex-R4 periferijos išėjimo lygmenį keitė kas 280ns. Įėjimo į trūkį ir išėjimo laikas 730ns. Skirtumo tarp IRQ ir FIQ trūkių laiko nebuvo. Cortex-M4 periferikos išvado loginį lygmenį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



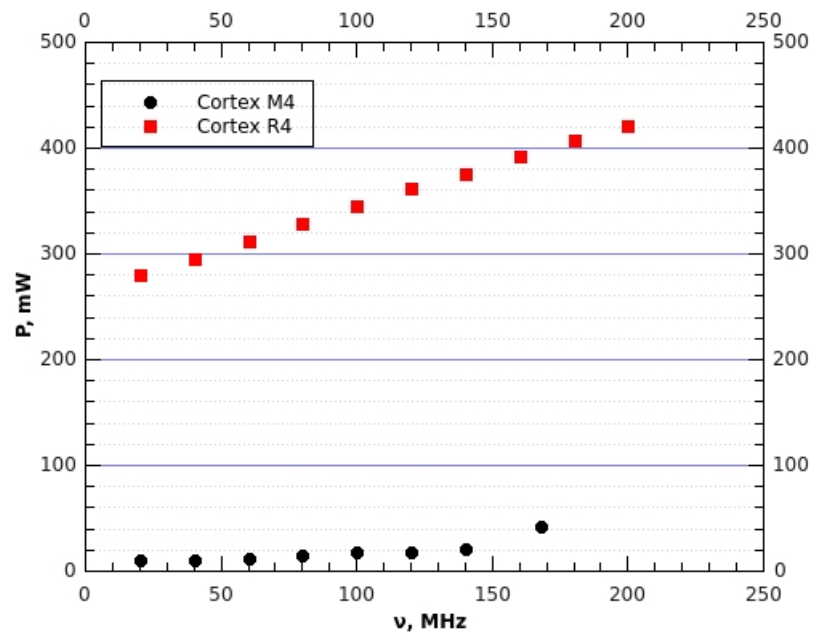
4 pav. Generuotų impulsų laikinė diagrama.

Mikrovaldiklio Cortex-R4 šerdis maitinama 1.2V, periferijos 3.3V. Kadangi 3.3V maitinama ir kiti įrenginiai esantys makete (tokie kaip šviesos sesnosrius, akselerometras) naudojama galia buvo skaičiuojama pagal formulę  $P = I_{serdies} * 1.2V + (I_{bendras} - I_{ivisoriniuiurenginiu}) * 3.3V$ ,

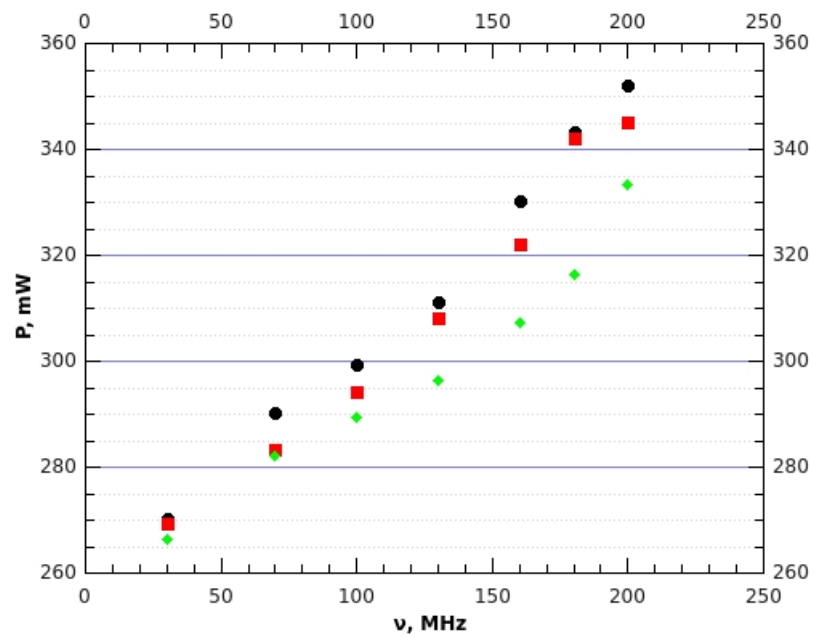
kur  $I_{\text{svoriniu irenginiu}}$  rastas visiškai atjungus mikrovaldiklio periferiją. Cortex-M4 maitinamas 3.3V, galia  $P = I * 3.3V$ .



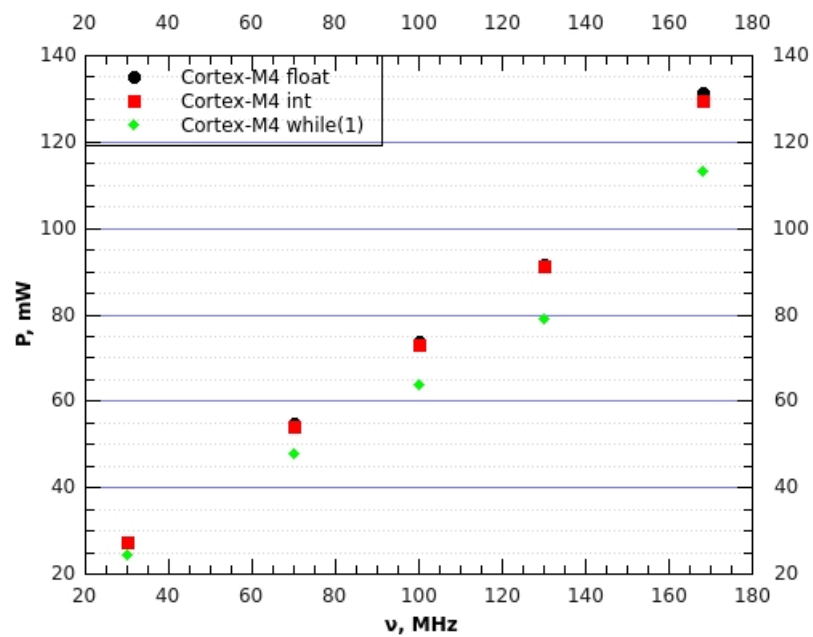
5 pav. Mikrovaldikliai stanby režime.



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



7 pav. RM48 mikrovaldiklio naudojamos galios priklausomybė nuo taktinio dažnio atliekant slankaus kabelio, neslankaus kabelio ir būnant tuščiame cikle.



8 pav. STM32F4 mikrovaldiklio naudojamos galios priklausomybė nuo taktinio dažnio atliekant slankaus kabelio, neslankaus kabelio ir būnant tuščiame cikle.



## 4 Išvados

1. RM48, esant 100MHz taktiniui dažniui, yra žymiai greitesnis atliekant dvigubo tikslumo slankaus kablelio skaičiavimus nei stm32f4 mikrovaldiklis.
2. Du procesoriai esantys R4 stipriai padidina mikrovaldiklio naudojamą galią
3. RM48 mikrovaldiklio periferija dirba lėčiau nei stm32f4 esant 100MHz taktiniam dažniui.
4. RM48 trūkių sistema yra labai lėta ir neefektyvi.
5. Mažiausia naudojama galia, kurią pavyko pasiekti su RM48 yra 82 kartus didesnė nei mažiausia vartojama galia, kurią pavyko pasiekti su stm32f4 maketu.
6. Mikrovaldikliams atliekant skirtingus skaičiavimus naudojama galia labiausiai priklauso nuo taktinio dažnio ir tik nežymiai nuo procesoriaus apkrovos skaičiavimais.

# 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. 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)

7. 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)

8. 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)

9.  $\pi$  skaičiavimo algoritmas.

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

# Santrauka

## „CORTEX R4 MIKROVALDIKLIO ARCHITEKTŪROS TYRIMAS”

Užduotys kurioms reikia didelio patikimumo reikalauja skaičiavimo dubliacijos. Mikrovaldiklis RM48 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 ir bandyta palyginti su kitu įprastesniu mikrovaldikliu. Taip pat susipažinta su 32 bitų Cortex šeimos programavimo ypatybėmis.