VILNIAUS UNIVERSITETAS
FIZIKOS FAKULTETAS
KIETO KŪNO ELEKTRONIKOS KATEDRA

Mindaugas Kurmauskas

CORTEX R4 MIKROVALDIKLIO ARCHITEKTŪROS TYRIMAS

Pagrindinių studijų kursinis darbas

(studijų programa – TAIKOMOJI FIZIKA)

Studentas Mindaugas Kurmauskas
Darbo vadovas dr. Mindaugas Vilūnas
Katedros vedėjas dr.(HP) Kęstutis Arlauskas

Vilnius 2013

Turinys

Įvadas **3**

1 Naudojamos įrangos ir programos **4**

2 Naudoti testavimo algoritmai ir jų aprašai **5**
2.1 Matematinių funkcijų skaičiavimo greičių įvertinimo algoritmai 5
2.1.1 Slankaus kablelio algoritmai 5
2.1.2 Neslankaus kablelio algoritmas 6
2.1.3 Loginių funkciję testavimo algoritmas 7
2.1.4 Greitas disckretinis Furjė eilutės transformavimo testas 7
2.2 Įėjimo ir išėjimo į trūkį algoritmo aprašymas 9
2.3 Srovės matavimai 9

3 Rezultatai **11**

4 Išvados **13**

5 Priedai **14**

Literatūros sąrašas **15**

Santrauka **16**

Summary **17**

Įvadas

Čia yra įvado tekstas!

1. Pirmas
2. Antras

1 Naudojamos įrangos ir programos

Naudojamas TMDXRM48USB maketas. Jame yra:

- Mikrovaldiklis xRM48L950AZWTT:
  - Du 32-bitu ARM Cortex-R4F procesoriai, veikiantys kartu[1]
  - 3MB flash, 256kB RAM
  - greitis iki 220MHz
  - 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 kompileris. Darbo pradėjimo metu vienintelis palaikė abu maketus, be papildomos įrangos.

[1] lockstep - kartu atlieka tas pačias komandas

2 Naudoti testavimo algoritmai ir jų aprašai

2.1 Matematinių funkcijų skaičiavimo greičių įvertinimo algoritmai

Šiame paragrafe aprašytiems algoritmams buvo naudojamas laiko trūkis, kurio pagalba buvo skačiuojamas 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 kablelio algoritmai

Slankaus kablelio Gauss Legendre algoritmas π skaičiavimui:

$$a_0 = 1\ b_0 = \frac{1}{\sqrt{2}}\ t_0 = \frac{1}{4}\ p_0 = 1$$

$$a_{n+1} = \frac{a_n + b_n}{2},$$

$$b_{n+1} = \sqrt{a_n b_n},$$

$$t_{n+1} = t_n - p_n(a_n - a_{n+1})^2,$$

$$p_{n+1} = 2p_n.$$

$$\pi \approx \frac{(a_n + b_n)^2}{4t_n} \tag{1}$$

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 kablelio 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 Neslankaus kablelio algoritmas

Fiksuoto tikslumo algoritmas:

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

```
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 funkciję testavimo algoritmas

```
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 disckretinis Furjė eilutės transformavimo testas

Naudotos arm dsp[2] 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;
  }
```

[2]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 naudota funkcija [**?**]:

$$Q31(x) = ROUND(2^{31} * 10^{x/20}) \tag{4}$$

Masyvo užpildo funkcija:

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

$$Im(z) = 0 \tag{5}$$

```
#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)round((pow(2,31)) * pow(10,(0/20)));
```

```
  for(volatile int32_t i = 0; i < 2048; i+=2) {
    skaicius = 1000*sin(i*3) + 1000*cos(i*7);
    testInputQ[i] = (int)round((pow(2,31)) * pow(10,(skaicius/20)));;
    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 porto vienos kojos loginį lygį, bei buvo įjungtas nuo laiko priklausantis tuščias trūkis. Osciloscopu matuota portų lygmenų kitimo laikas.

2.3 Srovės matavimai

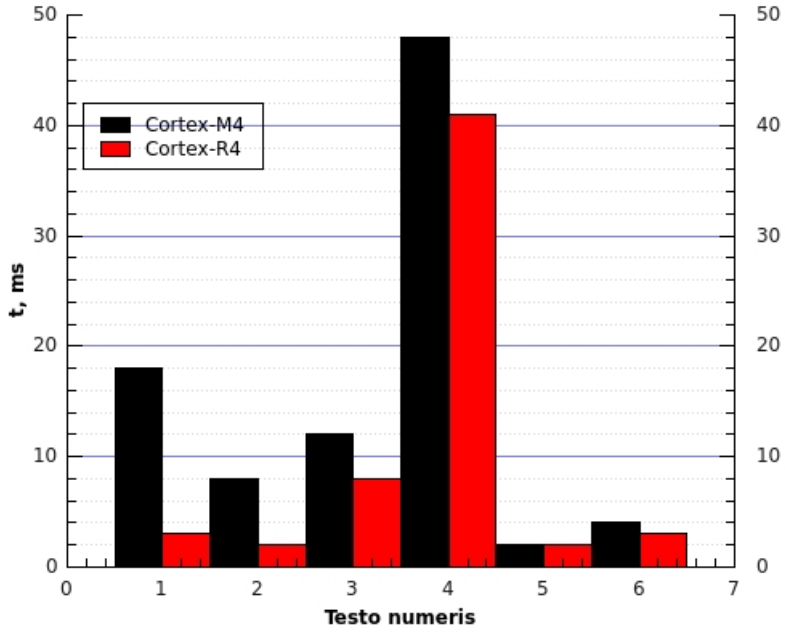
STM32F4DISCOVERY makete srovė matuota tiesiogiai multimetru tarp šerdies ir maitinimo. TMDXRM48USB maketas neturėjo trumpiklių, tarp maitinimo ir mikrovaldiklio buvo į lituotos dvi 0.12 Ω varžos prie šerdies ir portų maitinimo. Srovė matuota kaip įtampos kritimas ant žinomų varžų. Visų matavimų metu išoriniai portai buvo nustatyti kaip išėjimai ir nustatyti nuliniame loginiame lygije. Buvo atlikti matavimai esant šiom konfigūracijomis:

- įjungtas stanby rėžimas
- įjungtas negilaus miego rėžimas esant įvairiems taktavimo greičiams

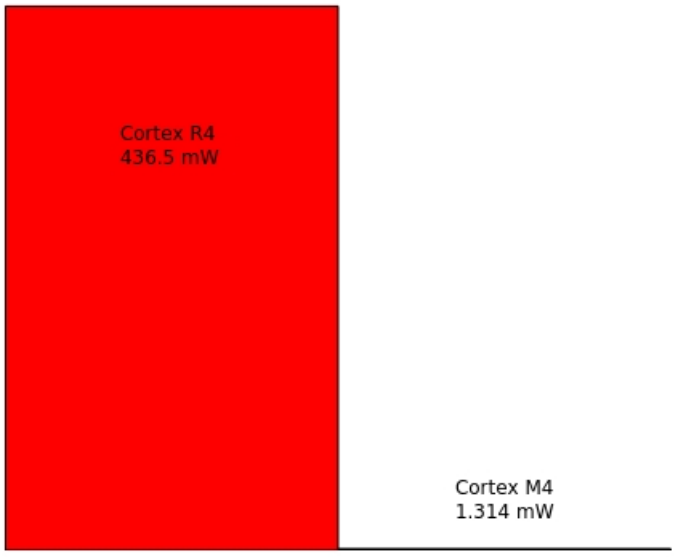
- mikrovaldikliai tuščiame cikle esant įvairiems taktavimo greičiams
- mikrovaldikliai atlikinėjo loginį testavimo algoritmą be trūkio esant įvairiems taktavimo greičiams

3 Rezultatai

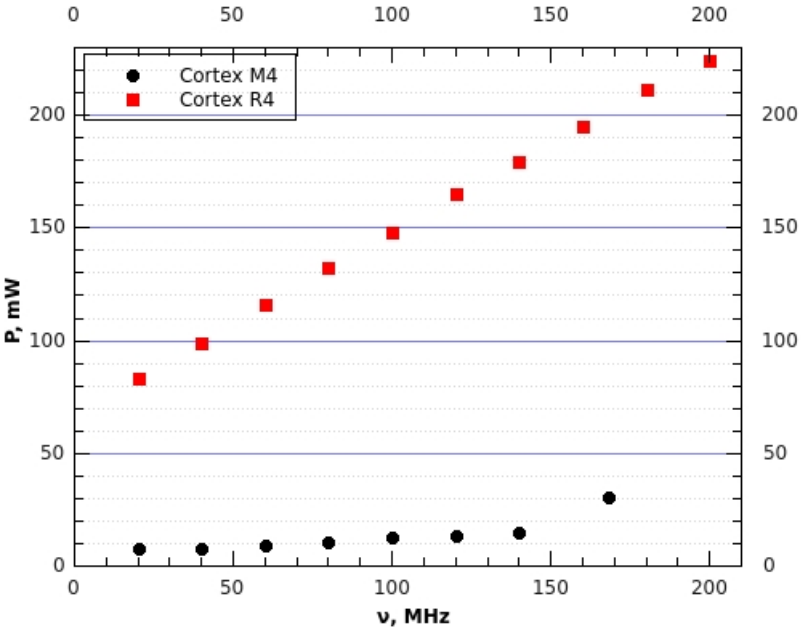
Cortex-R4 mikrovaldiklis visais skaičiavimo testais lenkė cortex-M4 mikrovaldiklĭ.



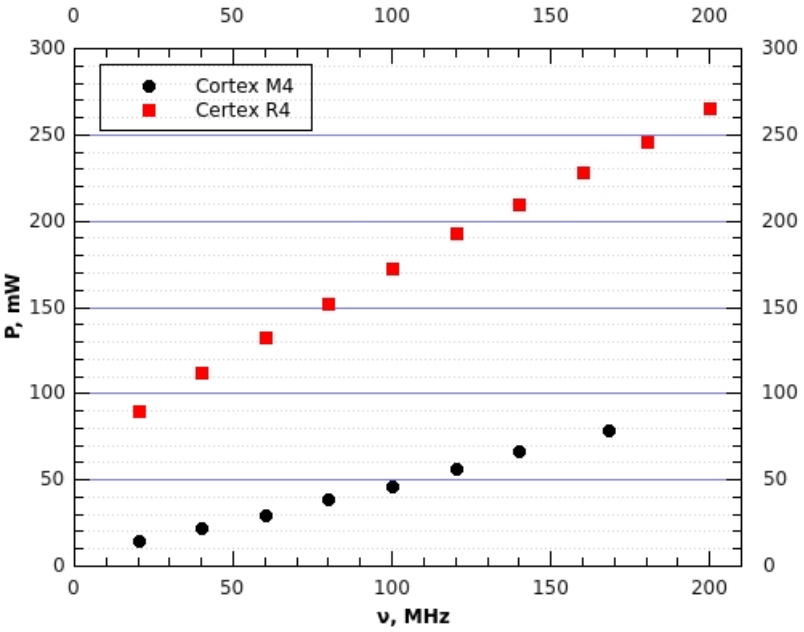
1 pav. *Skaičiavimo algoritmų laikai. 1 - dvigubo tikslumo slankaus kablelio testas. 2 - slankaus kablelio testas. 3 - neslankaus kablelio testas. 4 - loginių funkcijų tastas. 5 - Furijė slankaus kablelio. 6 - Furijė neslankaus kablelio.*



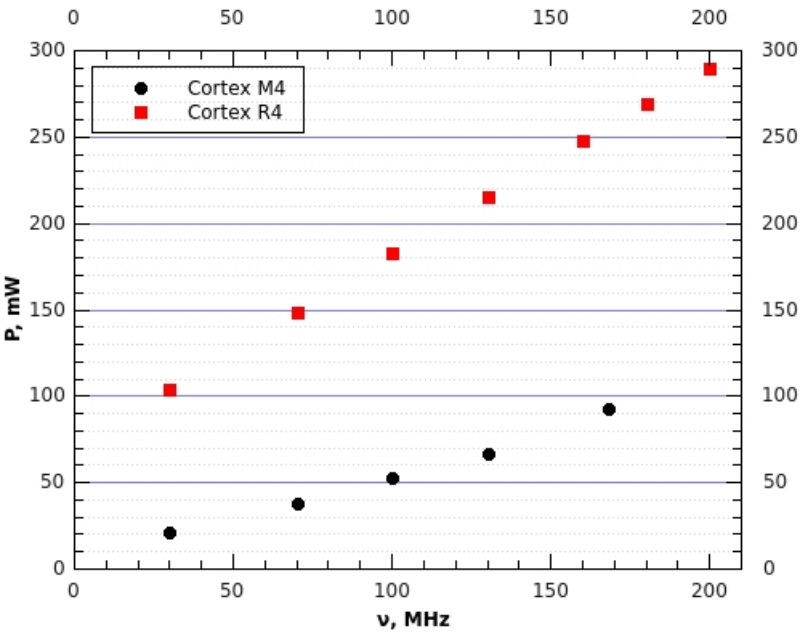
2 pav. *Mikrovaldikliai stanby rėžime*



3 pav. *Mikrovaldikliai stanby rėžime*



4 pav. *tuscias*



5 pav. *binary*

4 Išvados

1. Pirmas
2. Antras
3. Trečias
4. Ketvirtas

5 Priedai

Bet kokia reikalinga papildoma informacija: paveikslėliai, grafikai ir t.t.

@ONLINEDoe:2009:Online, author = Works, Wintergreen, title = This is a test entry of type @ONLINE, month = jun, year = 2009, url = http://wintergreenworks.wordpress.com/2011/12/2 of-float/

Santrauka

Mindaugas Kurmauskas

„CORTEX R4 MIKROVALDIKLIO ARCHITEKTŪROS TYRIMAS”

Tekstas

Summary

Mindaugas Kurmauskas

„TITLE”

Text