TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Arvutitehnika instituut

IAY40LT

Keijo Lass 123823IASB

TARKVARA KOMPONENDI ENERGIATARBE HINDAMINE ERINEVATEL SAGEDUSTEL

Bakalaureuse töö

Priit Ruberg
MSc
Nooremteadur

Tallinn 2015

Autorideklaratsioon

Olen koostanud antud töö iseseisvalt. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud. Käesolevat tööd ei ole varem esitatud kaitsmisele kusagil mujal.

Autor: Keijo Lass

12.06.15

Annotatsioon

See töö on ajendatud jätkuna Mihkel Kiili magistritööle "Tarkvara komponendi energiatarbe hindamine“, kus on pakutud meetod protsessori energiatarbe hindamiseks C keele tasemel. Töö on koostatud eesmärgiga pakkuda täiendusi energiatarbe hindamiseks erinevatel sagedustel. Sagedus mõjutab energiatarvet läbi kahe teguri: voolutarve ja käsu täitmiseks kuluv aeg. Mõlema tarbeks on välja pakutud valemid, et arvutada energiatarbe hinnang uuele sagedusele ilma uusi mõõtmisi tegemata.

Töös kasutatakse mikrokontrollerit PIC32MX perekonnast, mis omab sisemist ostsillaatorit ja faasilukuga sagedussüntesaatorit. Neid kahte koos kasutades genereeritakse erinevaid taktsagedusi vahemikes 4-80MHz. Energeetiliselt kõige optimaalsem on kontrollerit kasutada maksimaalse lubatud taktsagedusega 80MHz.

Töös on teostatud energiatarbe mõõtmised ja programmi eeldatava energiatarbe hinnangute analüüs. Tulemuseks saavutati hinnang, mis ei erinenud mõõdetust rohkem kui 3.1%. Energiatarbe hinnangu erinevatele sagedustele ümberarvutamine eeldab täpset keskmise voolutarbe regressioonimudeli koostamist. Seda on võimalik teha sagedusvahemiku terves ulatuses kahe või enam mõõtmisega, või jaotada sagedusvahemik osadeks, millele vastavad igaühele oma võrrandid.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 30 leheküljel, 5 peatükki, 9 joonist, 7 tabelit.

Abstract

This thesis is a continuation of Mihkel Kiil's master's thesis „Energy Consumption Evaluation of Software Component" in which a method for evaluating software power consumption is proposed. The thesis bases on the fact that a program could be divided into base operations. The sum of the energy consumption of each base operation is a total energy consumption of the given program. In that study one of the suggested topic for further research is scalability for different frequencies. The main idea of this thesis is to propose a method to calculate power consumption evaluation for different clock frequencies without taking new measurements.

Selected clock frequency affects power consumption in two ways – through the average current draw and instruction execution time. Both of these factors are linearly related to the frequency. Given that the observable system could be frequency scaled, the average current draw is calculable through linear regression model. The instruction execution time and clock frequency have also linear relationship.

The evaluated system is PIC32 microcontroller, which implements internal fast RC oscillator and phase-locked loop. It is crucial to understand how both of these oscillator system modules affect the overall power consumption. The estimations for the power consumption are given in frequency range of 4-80MHz, with the supply voltage of 3,3V.

The results are confirmed by measuring a test program. The objective was to give energy consumption estimations at a 95% confidence level. The estimations and measurements did not differ for more than 3%. Accuracy of the estimations were also influenced by oscillator system itself. Results show that a PLL system which uses internal fast RC oscillator as a clock source creates a clock frequency which differs from the configured settings up to 1.13%. To overcome this margin of error and to improve on the estimations accuracy, it is suggested to use more stable external oscillator. It is also suggested to research different microcontrollers and processors to further confirm the validity of the method.

The thesis is in estonian and contains 30 pages of text, 5 chapters, 9 figures, 7 tables.

Lühendite ja mõistete sõnastik

BGA	*Ball grid array*, kiibikorpuse tüüp
FRC	*Fast RC Oscillator*, kiire sisemine RC ostsillaator
ICSP	*In-circuit Serial Programming*, sidus jadaprogrammaator
PLL	*Phase-locked loop*, faasilukuga sagedussüntesaator
RAM	*Random-Access Memory*, suvapöördusmälu
TQFP	*Thin Quad Flat Package*, õhuke ruudukujuline lamekorpus
USB	*Universal Serial Bus*, universaalne järjestiksiin

Sisukord

Jooniste nimekiri

Tabelite nimekiri

1. Sissejuhatus

Energiatarbe hindamine ja prognoosimine on kujunenud tähtsaks osaks uute süsteemide väljaarendamisel, seda eriti hajussüsteemide puhul, kus seadmed võivad voolu tarbida patareide või akude pealt. Süsteemi erinevad osad võivad asuda üksteisest kaugel ja seadmete hooldusvälp peaks olema mõõdetav kuudes, paremal juhul aastates. Täpne energiatarbe hindamine on abiks selliste süsteemide projekteerimisel ning komponentide valikul. Kuivõrd riistvaraline lahendus on sageli kindlaks määratud või rangete piirangutega, on võimalik läbi tarkvara seda mõjutada. Eesmärgiks on saavutada vastavale riistvarale optimaalne ning energiasäästlik lahendus just läbi tarkvara. Võttes arvesse kui palju mingi operatsioon konkreetse riistvara peal energia seisukohast maksab on võimalik projekteerida minimaalse energiakulukusega süsteem.

Käesolev töö baseerub Mihkel Kiili 2013. aasta magistritööl „Tarkvara komponendi energiatarbe hindamine“, kus on analüüsitud süsteemi tarkvara mõju energiatarbele. Töös on välja pakutud metoodika energiatarbe hindamiseks C keele tasemel. Meetodi täpsust kontrolliti 32-bitise mikroprotsessoriga PIC. Selgus, et väljapakutud meetodi täpsus on võrreldav assembleri tasemel teostatud hinnangutega - mõõdetud tulemuste ja antud hinnangute vahe ei ületanud 2.5% piiri. Küll aga on pakutud meetod teatud kasutuspiirangutega, millest üheks on sama taktsageduse kasutamine. Selle töö eesmärk on pakkuda energiatarbe hindamise meetodile täiendust erinevate taktsageduste testimise näol.

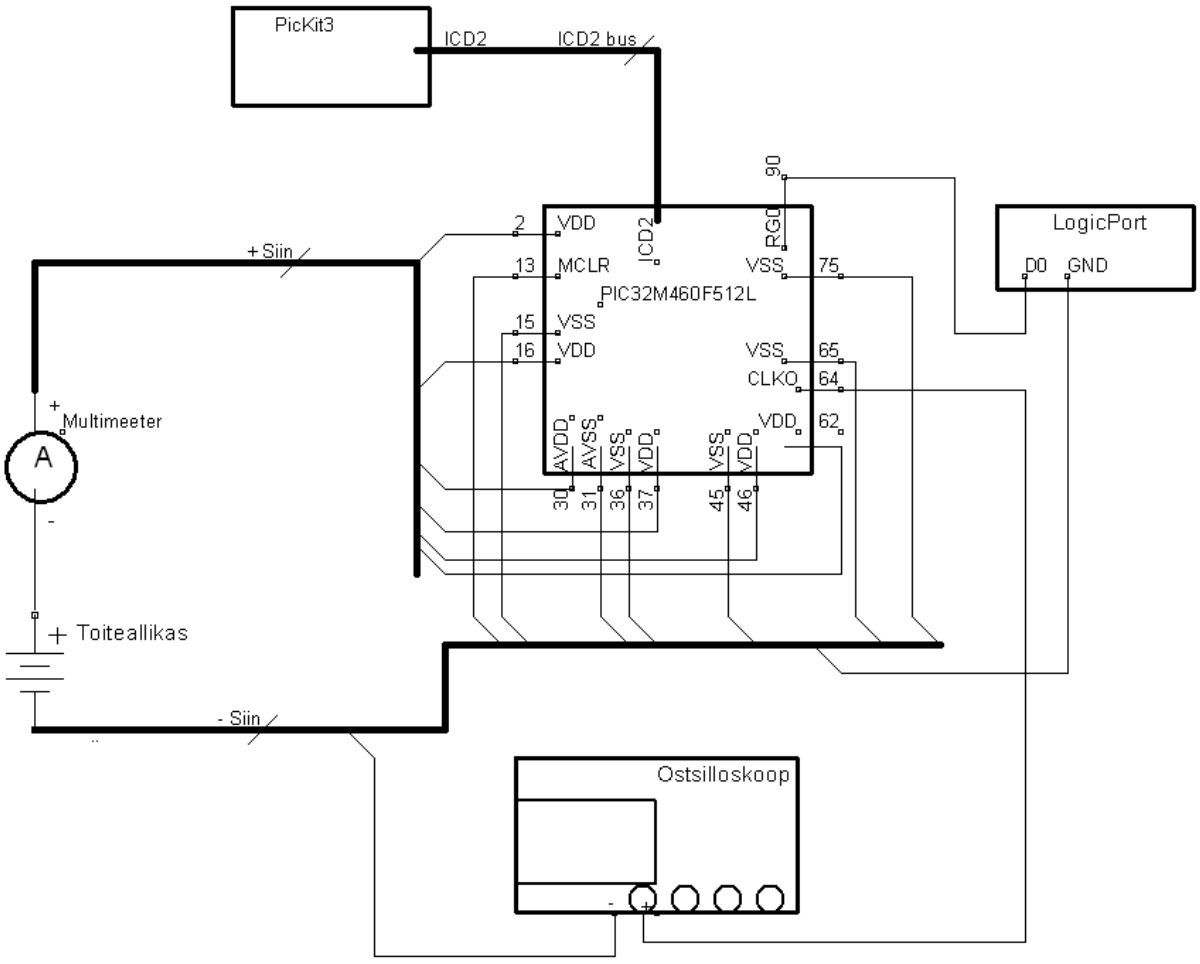
Õige taktsageduse valik aitab energiatarvet vähendada kahel põhjusel. Esiteks ajal, kui kontroller ei käita põhiprogrammi, vaid ootab näiteks sisendite järel, võib taktsageduse dünaamiliselt viia madalamaks. Sama võimalust võib kasutada ka programmi erinevate osade puhul, kus ajalised piirangud on suuremad. Teiseks tuleks valida taktsagedus, mille kiiruse ja energiatarbe suhe on optimaalne. Väiksem taktsagedus tähendab küll väiksemat voolutarvet, kuid ajaliselt kestab iga operatsioon kauem, seega on ka kokkuvõttes energiatarve suurem.

Töö põhieesmärk on tõestada, et energiatarbehinnangut on võimalik ümber arvutada erinevatele taktsagedustele ilma baasmõõtmisi uuesti teostamata. Sealjuures peavad tulemused vastama vähemalt 95% usaldusnivoole. Töös pakutud lahenduste teostamiseks on kasutatud mikrokontrollerit PIC32 tootjalt Microchip. Kogu töös kasutatav tarkvara peab olema vabavaraline. Töö alameesmärgiks on analüüsida kontrolleris olevat

ostsillaatorisüsteemi, milles sisalduvad faasilukuga sagedussüntesaator ning kiire sisemine RC ostsillaator. Eesmärgiks on mõista, kuidas taktgeneraatori komponendid mõjutavad lõplikku energiatarvet ja hinnangut. Konkreetse mikrokontrolleri tootja poolt avaldatud spetsifikatsioonis puudub analüüs taktgeneraatori seadistuse ja voolutarbe vahel. Seetõttu on vaja teha need mõõtmised ning analüüs töösiseselt. Testsüsteemi skeem peab olema minimaalne, et teostada energiatarbe mõõtmisi. Kõik mõõtmisteks mittevajalikud moodulid peavad olema kontrolleris välja lülitatud. Sealhulgas kõik energiasäästurežiimid ning vahemälu. See aitab isoleerida voolutarbijaid skeemis. Mõõtmistel ja arvutustel kasutatav taktsageduste vahemik on 4MHz kuni 80MHz. Kontrolleri toitepingeks on 3,3V.

2. Testsüsteem

Mõõtmistel kasutatava testsüsteemi aluseks on võetud M. Kiili magistritöös kasutatav testsüsteem, mis koosneb mikrokontrollerist PIC32MX460F512L, programmaatorist PicKit3, vooluallikast Hameg HMP2030, multimeetrist Agilent 34405A, loogikaanalüsaatorist Intronix LogicPort ja makettplaadist [2]. Lisaks on kasutatud ostsilloskoopi Agilent DSO-X 3034A. Programmeerimiseks ning silumiseks on kasutatud MPLAB X IDE versioon 2.35, mis kasutab vabavaralist C kompilaatori XC32 versiooni 1.34. Täielik ühendusskeem koos klemmide ja seadmetega on toodud joonisel 1.



Joonis 1. Testsüsteemi skeem.

Joonisel 1 on kontrolleri ühendusviigud märgitud vastavate viigu numbrite ja nimedega. Kontroller on vooluvõrku ühendatud läbi 12 juhtme. Kuna käesolev töö baseerub

eelpoolnimetatud magistritööl, oli eesmärgiks valida võimalikult lähedane testsüsteem. Järgnevalt lühiülevaade kasutatud seadmetest ja nende režiimidest.

PIC32MX460F512L on 32bitine kontroller, mis kuulub PIC32MX3/MX4 perekonda tootjalt Microchip. Töös on kasutatud 100-viiguga TQFP kontrollerit, kuid on olemas ka 121-viiguga BGA kiibikorpuses variant. Kontrolleril on 512kB välkmälu ning 32kB RAM, mis asuvad väljaspool tuuma. Välkmälu ja siiniliidese vahele on paigutatud vahemälu moodul käskude ja andmete jaoks. Maksimaalne taktsagedus on 80MHz ning toitepinge 3.6V [3]. MIPS32 M4K tuumas on kaks konveierit millest kahte esimest, käsuvõtu ja käsu töötluse, astet jagatakse. Kolm ülejäänud astet on iseseisvad. Seega ei ole konveieritel väga palju astmeid ning nad ei paku võimalust käskude spekulatiivseks eelvõtuks. Nii ei teki energiakulukaid konveieri ümberlaadimisi [2]. Tuuma ümber on moodulid täiendavate funktsionaalsuste nagu näiteks USB ja ICD kasutamiseks. Kontrolleri põhimõtteline skeem on nähtav peatükis Lisa1.

Microchipi PicKit3 (edaspidi PicKit) on kontrolleri programmeerimiseks ja silumiseks mõeldud seade, mis on vahelüli MPLAB X tarkvara ning kontrolleri vahel. Läbi PicKiti on võimalik programmeerida seadmeid mille toitepinge on 1.8V – 5V [5]. Seade ühendatakse arvutiga USB kaabliga, seega eraldi toidet ei vajata. PicKit võib ka kontrollerit varustada toitepingega, kuid maksimaalne lubatud voolutarve on sellisel juhul 30mA [5]. Võimalust kontrollerit toita läbi PicKiti töös ei kasutata, sest toitepinge peab olema mõõtmiste täpsuse saavutamiseks jälgitav, kuid programmaator ei paku võimalust toitepinget ilma lisamõõteseadmeteta mõõta. Lisaks ületab kontrolleri voolutarve kõrgemate sageduste juures lubatud maksimaalse 30mA piiri. Kontrolleri ja PicKiti ühendamiseks on 5 juhet, mille ülesanded on määratud ICSP protokolliga. Juhtmete ühendamise skeem on nähtav tabelist 1.

Tabel 1. Programmaatori klemmid ja nende tähendus [2].

Programmaatori klemmi number	**Klemmi nimetus**	**Klemmi funktsioon**
1	MCLR/Vpp	Kontrolleri reset, ühendatud kontrolleri MCLR klemmiga.
2	Vdd	Ühendatud seadme toitega.
3	Vss	Ühendatud seadme maaga.
4	PGD	ICSP programmeerimise andmed, ühendatud kontrolleri PGD klemmiga.
5	PGC	ICSP programmeerimise taktsagedus, ühendatud kontrolleri PGC klemmiga.

Vooluallikas Rohde & Schwarz Hameg HMP2030 on kolme eraldiseisva kanaliga ja võimaldab maksimaalselt 32V toitepinget ja voolutugevust vahemikus 0.001-5A. Ühe kanali maksimaalne väljudvõimsus on 80W ning kolme kanali väljundvõimsus 188W. Seade võimaldab voolutugevuse mõõtmist täpsusega 100µA [6]. Vooluallikal on LCD näidik, mis kuvab reaalajas kasutatavad kanalid, voolutarbe ning toitepinge kanalis.

Agilent 34405A multimeeter oli testsüsteemis kasutusel voolutugevuse mõõtmiseks. Mõõtmistel kasutatud mõõtepiirkond 100mA lubas kasutada täpsust kuni 1µA. Sellises vahemikus on mõõteviga 0.05%+0.5µA [7]. Multimeeter võimaldab kasutada ka automaatset maksimaalse, minimaalse ja keskmise mõõtetulemuse talletamist.

Intronix LogicPort on 34 eraldiseisva andmesisendiga loogikaanalüsaator, mis võimaldab maksimaalset diskreetimissagedust 500MHz [8]. Analüsaator ühendatakse arvutisse USB kaabliga ning kasutatakse läbi samanimelise tarkvara LogicPort, mille abil saab tulemusi analüüsida ning seadet juhtida. Töös on kasutatud loogikaanalüsaatorit programmi ajaliseks mõõtmiseks. Kasutatavad seadistused on näha tabelist 2.

Tabel 2. Loogikaanalüsaatori seaded.

Parameeter	**Väärtus**
Diskreetmissagedus, MHz	200
Lävipinge, V	1.5
Enne trigerit salvestatud andmete osakaal näidatavas tulemuses, %	4

Testsüsteemis kasutatud ostsilloskoop Agilent DSO-X 3034A on 4 kanaliga, 350MHz ribalaiuse ning maksimaalse diskreetmissagedusega 4 GSa/s (miljonit diskreeti sekundis). Seadmel on 8.5 tolline WVGA ekraan [9]. Teostatud mõõtmistel oli kasutusel ka seadme mõõtmistulemuste funktsioon, mis kuvab mõõdetud sageduse minimaalset, maksimaalset ja keskmist suurust. Ostsilloskoop oli testsüsteemis ühendatud kontrolleri CLKO viigu külge, mis võimaldas monitoorida kontrolleri taktsagedust.

Kontrolleri programmeerimiseks kasutatav MPLAB X on vabavaraline arenduskeskond tootjalt Microchip. Lisana on kasutusel XC32 C kompilaatori tasuta versioon, mis võimaldab

programmeerida kontrollerit C keeles. Kasutatakse vaikimise sätet, mis lülitab välja kõik kompilaatori poolt teostatavad optimeeringud. Niiviisi tagatakse, et iga C keelne lause täidetakse, ning seejuures saab kindel olla, et mõõtetulemused kajastavad vastava operatsiooni energiatarvet. Koodikatte leidmisel oli kasutusel GCC hulka kuuluv profileerimisvahend gcov, mis väljastab iga koodirea kohta info, kas ja kui mitu korda seda täideti. GCC kompileeritud programmi ei ole võimalik PIC kontrolleril käivitada, selleks on vaja see kompileerida XC32 kompilaatoriga, kuid C keelne lähtekood ja koodikate on täpselt sama.

2.1. PIC32 ostsillaatorisüsteem

PIC32 taktgeneraator on ehitatud väga paindlikult ning võimaldab erinevaid seadeid ja režiime süsteemi taktsageduse määramiseks. Kasutajal on võimalik süsteemi taktsagedusi genereerida neljast erinevast allikast:

- POSC – primaarne ostsillaator, klemmidelt OSC1 ja OSC2
- SOSC – sekundaarne ostsillaator, klemmidelt SOSCI ja SOSCO
- FRC – sisemine kiire ostsillaator
- LPRC – madala energiatarbega sisemine ostsillaator [1].

Ostsillaatorisüsteemi kohandatakse läbi erifunktsioonregistrite, millest selles töös on olulisteks kaks:

- OSCCON - *Oscillator Control Register* – ostsillaatori juhtregister. Selle registri abil saab teostada taktgeneraatori vahetust käitluse ajal, samuti sätestatakse sealt FRC jagatisbitid. Registris on määratud PLL väljundjagaja (PLLODIV) ning sisendkordaja (PLLMULT). Lisaks saab registrist määrata käituse ajal uut tugisignaali allikat (NOSC). OSCON register käitub ka olekuregistrina, kust saab lugeda juhtbitte taktsignaali allika vahetuse kohta (OSWEN).
- OSCTUN - *FRC Tuning Register* – Kiire RC ostsillaatori häälestusregister. Register on kasutusel sisemise FRC ostsillaatori tarkvaraliseks peenhäälestamiseks. 32-bitisest registrist on kasutusel 6 bitti, mille abil saab taktsagedust muuta ±12.5%. Iga samm on tootjapoolne umbkaudne hinnang, mida ei ole testitud ega dokumenteeritud.

Registribittide paigutust illustreerib joonis 2.

Name	Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
OSCCON(1)	31:24	—	—	PLLODIV<2:0>			FRCDIV<2:0>		
	23:16	—	SOSCRDY	PBDIVRDY	PBDIV<1:0>		PLLMULT<2:0>		
	15:8	—	COSC<2:0>			—	NOSC<2:0>		
	7:0	CLKLOCK	ULOCK	SLOCK	SLPEN	CF	UFRCEN	SOSCEN	OSWEN
OSCTUN(1)	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	—	—	—	—	—	—	—	—
	7:0	—	—	TUN<5:0>					

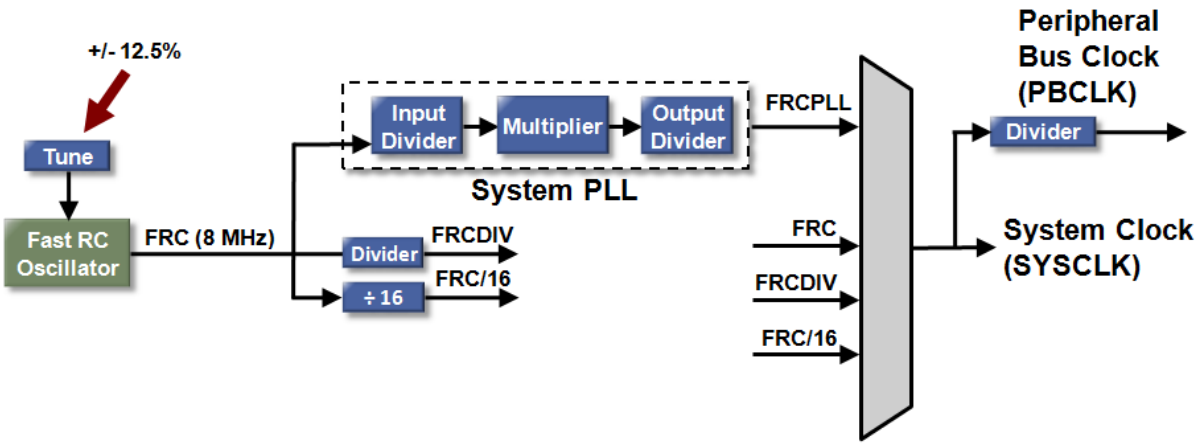
Joonis 2.OSCON ja OSCTUN registrid [1].

Käesoleva töö seisukohast on olulisteks osadeks süsteemi juures kiire RC ostsilaator (FRC) ning PLL erinevad komponendid.

2.1.1. Kiire RC ostsillaator

PIC32 mikrokontrolleri on sisse ehitatud kiire RC ostsillaator, mis genereerib taktsageduse 8 MHz. Sageduse kõikumine pingel 3.3V ja temperatuuril 25 °C on ±2% [1]. Ostsillaatori signaali saab anda edasi PLL sisendisse, jagada läbi 16ga või jagada läbi kasutaja poolt määratud arvuga, mida saab seadistada OSCON registris asuvate FRCDIV bittidega. Need bitid võimaldavad langetada sagedust 4MHz (jagatud 2) kuni 31 kHz (jagatud 256) ka programmi käituse ajal. Samuti on võimalus ostsillaatorit peenhäälestada läbi OSCTUN registri, kus sagedust saab tõsta ja langetada kuni 12.5%. Peenhäälestus register on mõeldud kompenseerimaks temperatuuri mõjusid FRC ostsillaatorile. FRC illustratsioon on nähtav joonisel 3. OSCTUN register on tähistatud joonisel punase noolega.

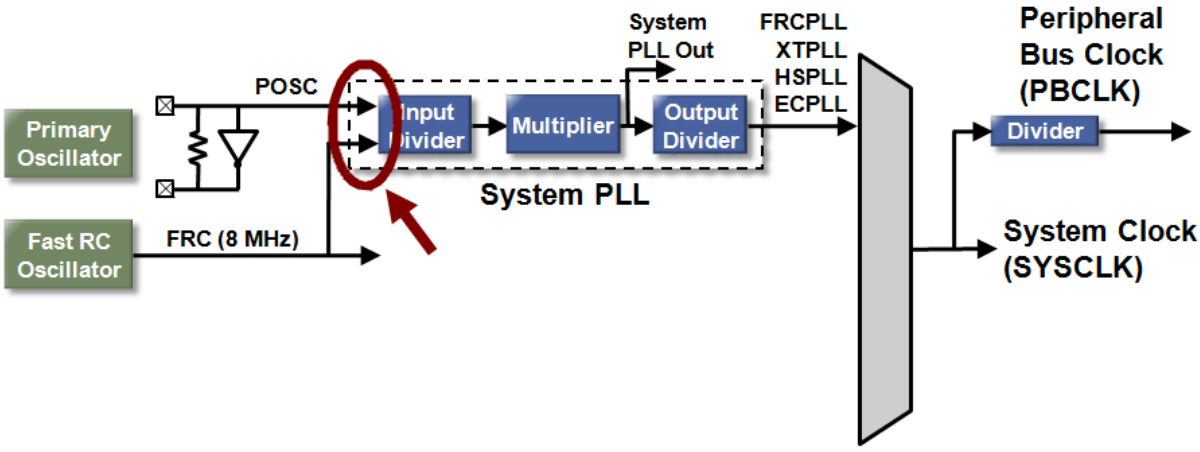
Seades OSCON registri COSC bitid väärtusele 001, ehk režiimi FRCPLL, saab anda FRC poolt genereeritava 8MHz taktsageduse süsteemi faasilukuga sagedussüntesaatorisse. Kuna sagedussüntesaator nõuab, et sisendsignaal oleks vahemikus 4-5MHz, siis FRCPLL režiimis on sisendjagaja väärtuseks vaikimisi 2, seega saab seadistada vaid sisendkordajat, ning väljundjagajat. Niisiis on võimalik näiteks FRC väljundsignaalist (8MHz) genereerida signaale vahemikus 23.4 KHz (muutujad vastavalt 2,15,256) kuni 80MHz (muutujad 2,20,1).



Joonis 3. PIC32 ostsillaatorsüsteemi FRC [4].

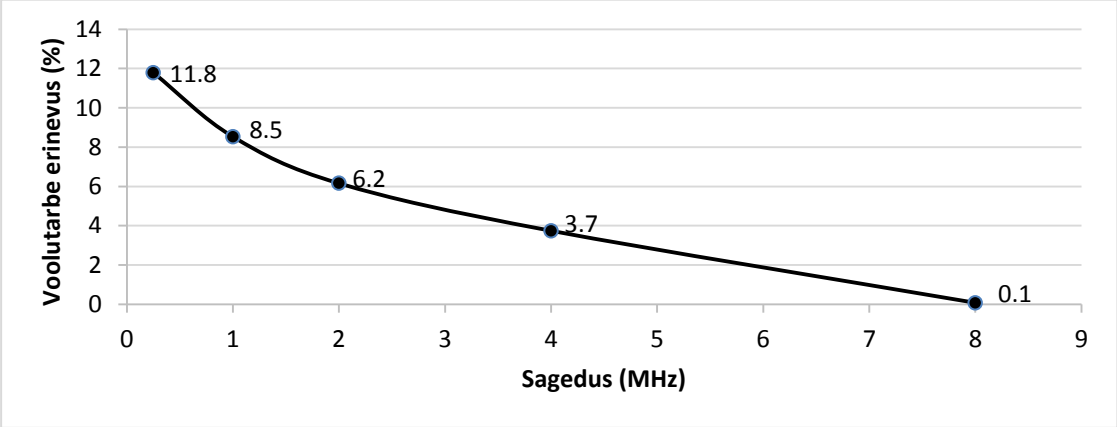
2.1.2. PIC32 faasilukuga sagedussüntesaator

PIC32 taktgeneraator sisaldab PLL süsteemi, mis võimaldab muuta taktsagedust, kasutades ühte valitud sisendsignaal allikat. Süsteemi saab kasutada nii sisemise RC ostsillaatori, kui ka primaarse ostsillaatoriga XT, HS või EC režiimis. Kasutaja saab valida sisend- ja väljundjagajat, ning sisendkordajat. Kõik eelnimetatud bitid on seadistatavad DEVCFG2 juhtregistri kaudu. Sisendsignaalikordaja ja väljundsignaalijagaja bitid kantakse kontrolleri taaskäivituse puhul üle ka OSCCON registrisse. Seadme käitusajal on võimalik muuta vaid sisendkordajat ja väljundjagajat, kuna PLL-i süsteemi sissetulev taktsagedus peab jääma 4-5MHz vahele. Seadme sisendsignaalijagaja on tähistatud joonisel 4 punase ringiga. Sagedussüntesaator kulutab teatud aja, et tagada stabiilne väljund. Selle jaoks on OSCCON registri 5. bitt SLOCK, mis seatakse seisu 0, kui PLL on saavutanud stabiilse väljundi või kui PLL käivitustaimer on lõpetanud. Taimer on seatud 2 ms peale.[3]



Joonis 4. PIC32MX faasilukuga sagedussüntesaator [4].

Sagedussüntesaatori mooduli kasutamine toob kaasa suurenenud voolutarbe, kuid tootjapoolne sellekohane dokumentatsioon puudub. Vool mida sagedussüntesaator vajab määratud väljundi tekitamiseks, sõltub sellest, kui palju erineb sisend signaaliallika sagedus soovitud väljundsagedusest. Allolev joonis 5 illustreerib, kui palju erineb voolutarve protsentuaalselt kahe režiimi, FRCDIV ja FRCPLL, vahel samadel taktsagedustel. FRCPLL režiimi kasutamine mõjutab voolutarbe kasvu mittelineaarselt. Mõõdetud programmiks on testprogrammi (joonis 6) tühikäik, toitepingel 3,3 V.



Joonis 5. FRCPLL ja FRCDIV režiimide keskmine voolutarbe vahe.

Töö tulemustes toodud hinnangud ja mõõtmised on tehtud kõik PLL režiimis, seega ei ole vaja tarvis sagedussüntesaatori lisa voolutarvet neis arvestada. Hinnangute arvutamist üle erinevate taktgeneraatori režiimidele käesolev töö ei käsitle.

3. Energiatarbe hindamise meetod

Energiatarbe hindamise metoodika põhineb eeldusel, et programmi energiatarve on leitav programmis täidetavate baasavaldiste ning baasoperatsioonide energiate summana. Kogu programmi energiatarbe võib seega arvutada valemiga (1).

$$E_{programm} = \sum_{i=1}^{n} m_i \times E_{avaldis\ i} \qquad (1)$$

Valemis 1 on C keelse i-nda avaldise energiatarve $E_{avaldis\ i}$ (ühikuks džaul, J) ja arv, mitu korda i-ndat avaldist programmis esineb m_i. Kõikide avaldiste energiatarbe summa on kogu programmi energiatarve $E_{programm}$. Tervikavaldiseks võiks näiteks lugeda a = b + c. Selles avaldises on sees kaks baasoperatsiooni: a = ja b+c. Kui igas tervikavaldises on 1..k baasoperatsiooni, kus k on lõplik positiivne täisarv, on iga tervikavaldise energiatarve $E_{avaldis\ i}$ avaldatav baasoperatsiooni energiate E_{baasop} kaudu (2).

$$E_{avaldis\ i} = \sum_{i=1}^{k} E_{baasop\ i} \qquad (2)$$

Baasoperatsiooni energiatarve E_{baasop} on mõõdetav suurus, mis mille mõõtühikuks on mikrodžaul (µJ). Energiahulk E leitakse valemi (3) abil.

$$E = V_{dd} \times I_{kesk} \times T_{op} \qquad (3)$$

Seega võib avaldada baasoperatsiooni korrutisena, mille tegurid on kontrolleri toitepinge V_{dd} (ühikuks volt, V), operatsiooni ajal keskmine tarbitav voolutugevus I_{kesk} (ühikuks amper, A) ning operatsiooni täitmiseks kulutatav aeg T_{op} (ühikuks sekund, s).

Baasoperatsiooni energiatarbe leidmiseks on vaja koostada testprogramm ja seejärel panna see tööle koos ja ilma baasavaldiseta. Kahe programmi mõõdetud energiatarbe vahe ongi baasoperatsiooni energiatarve (4).

$$E_{baasop} = E_{kogu} - E_{ilma} \qquad (4)$$

E_{kogu} on programm koos uuritava baasoperatsiooniga ja E_{ilma} on programm ilma baasavaldiseta. Viimast võib nimetada ka testprogrammi „tühikäiguks“. Järgnevalt näide testprogrammist joonisel 6, kus täidetavaks baasoperatsiooniks on b = c.

```
Int main(void)
{
	TRISG = 0;   // Määrab PORTG väljundpordiks
	PORTG = 0;   // Algväärtustab PORTG väärtusele 0
	while(1)
	{
		LATVGINV = 0x0001; /*Pordi G biti 0
					inverteerimine */
		int i, b = 5 , c = 10;
		for(i = 0 ; i < 1000 ; i++) // Testtsükkel
		{
			b = c ; // Uuritav baasoperatsioon
		}
	}
	return 0;
}
```

Joonis 6. Testprogramm koos uuritava baasoperatsiooniga.

Peaprogrammi sees olev *while(1)* tsükkel tagab meile programmi lõputu käitamise, seega ka pideva muutujatele mälu eraldamise ja vabastamise. Käsuga *LATGINV = 0x0001* inverteeritakse väljundpordi G nullis bitt, mis on ühendatud loogikaanalüsaatoriga. Nii on võimalik leida *while* tsükkli sees oleva programmiosa ajaline kestus. Testtsükkli *for()* sees on uuritav baasoperatsioon. Operatsiooni korratakse 1000 korda, et ümardada väikseid kõikumisi mõõtetulemustes. Toitepingel 3,3V ning sagedusel 4MHz olid testprogrammi mõõdetud tulemused järgmised:

- I_{kesk} = 12,420 mA
- T_{op} = 17,8756 s

Valemi (3) abil võime arvutada baasoperatsiooni energiatarbe E_{kogu}.

$$E_{kogu} = 3{,}3 \times 12{,}420 \times 17{,}8756 = 732{,}64934\ [\mu J] \qquad (5)$$

Järgmiseks on vaja leida testprogrammi poolt tarbitav energiahulk ilma uuritava baasavaldiseta. Selleks kasutame joonisel 6 toodud programmi, toitepinget ja taktsagedust, kuid kustutame *for()* tsükkli sees oleva avaldise b = c. Mõõdetud testtsükkli mõõdetud suurused olid:

- I_{kesk} = 12,454 mA
- T_{op} = 13,9073 s

Testprogrammi energiatarve ilma uuritava baasavaldiseta arvutatakse valemi (3) abil E_{ilma}.

$$E_{ilma} = 3{,}3 \times 12{,}454 \times 13{,}9073 \ = 571.56500\ [\mu J] \qquad (6)$$

Kasutades valemit (4), leitakse baasoperatsiooni energiatarve. Tuleb veel meeles pidada, et kuna testprogrammis täideti käsku b = c täpselt 1000 korda, on vaja vahe läbi jagada 1000-ga.

$$E_{baasop} = \frac{(732{,}64934 - 571{,}56500)}{1000} = 0{,}161[\mu J] \qquad (7)$$

Järgmiseks on vaja leida mitu korda baasoperatsiooni programmis tehakse. Selleks kasutatakse koodikatte hindamise programme, millest üks on näiteks GCC tööriist gcov. Programm koostab pärast kompileerimist ning käivitamist faili laiendiga .c.gcov, kus on kirjas enne iga koodirida, mitu korda seda täideti. Illustreeriv näide asub joonisel 7.

```
    1:    1:int main()
    -:    2:{
    1:    3:    LATGINV();
    -:    4:    int i;
    1:    5:    int b = 100;
    1:    6:    int c = 2;
 1001:    7:    for(i = 0; i < 1000; i++){
 1000:    8:          b = c ;
    -:   9:    }
    1:   10:    return 0;
    -:   11:}
```

Joonis 7. Gcov väljund testprogrammile.

Avaldist b = c täideti programmi käitamisel 1000 korda, tsükklit *for()* täideti 1001 korda ja teisi lauseid 1 korra. Tasub ära märkimist, et GCC ei tule toime PIC spetsiifiliste C käskudega, näiteks *LATGINV* väärtustamisega. Lahendusena tuleks testitavas programmis asendada taolised käsud funktsioonide väljakutsetega (joonisel 7 on *LATGINV() void* tüüpi funktsioon).

3.1. Energiatarve hindamismeetodi kasutamine erinevatel taktsagedustel

Eelnevalt näiteks toodud mõõtmiste puhul eeldati, et taktsagedus on alati sama. Meetodi laiemaks kasutamiseks on oluline teha täiendusi, et hinnata programmi energiatarvet erinevatel taktsagedustel ilma baasmõõtmisi uuesti teostamata. Selle jaoks on vaja mõista, kuidas mõjutab taktsageduse muutus programmi energiatarvet. Võttes aluseks energiatarbe

arvutamise põhivalemi (3) selgub, et taktsageduse muutus mõjutab kahte tegurit: keskmist voolutarvet I_{kesk} ja operatsiooni täitmiseks kulutatavat aega T_{op}.

Käsu töötlusaega T_{op} võib vaadelda läbi valemi (8), mille komponentideks on käsule kuluv protsessortaktide arv CPI ning protsessori taktsageduse ajaline kestus t_{CY} (ühik s, sekund). Seejuures on huvipakkuv just teine tegur, kuna käsule kuluv taktide arv on määratud protsessori käsustiku ja mikroarhitektuuriga.

$$T_{op} = CPI \times t_{CY} \qquad (8)$$

Taktsageduse ajaline kestvus on lineaarses seoses taktsagedusega (9), kus f_{CY} on protsessori taktsagedus.

$$t_{CY} = (f_{CY})^{-1} \qquad (9)$$

Valemi (9) põhjal võib järeldada, et muutes taktsagedust muutub ka käsule kuluv aeg lineaarselt. Seda kinnitavad ka mõõtetulemused. Tõstes taktsagedust kaks korda, väheneb ka operatsioonile kuluv aeg kaks korda. Illustreeriv näide tabelis 3, kus on baasoperatsiooni b = c mõõdetud ajaline kestus erinevate taktsageduste juures.

Tabel 3. Baasoperatsiooni b = c töötlusaeg erinevatel sagedustel.

Taktsagedus, MHz	**Operatsiooni ajaline kestus, µs**
4	3.9726
40	0.3967
80	0.1981

Sellest lineaarsest sõltuvusest lähtuvalt võib valemis (3) oleva töötlusaja T_{op} korrutada teguriga *k*, mille moodustavad teostatud baasmõõtmiste taktsagedus f_B ja hinnatava süsteemi taktsagedus f_N.

$$k = \frac{f_B}{f_N} \qquad (10)$$

Olgu baasmõõtmised teostatud sagedusel 4MHz. Vaja on hinnata baasoperatsiooni b = c töötlusaega sagedusel 80MHz. Mõõtmiste tulemusena oli operatsiooni ajaline kestus 4MHz taktsageduse juures 3,9726 µs. Ajaline kestus 80MHz juures on arvutatud valemis (11). Mõõtmise tulemusena saadud operatsiooni ajakulu on näha tabelis 3.

$$T_{op80MHz} = 3{,}9726 \times \frac{4}{80} = 0{,}19863\ [\mu s] \qquad (11)$$

Taktsageduse muutmine toob kaasa muutuse ka keskmises voolutarbimises. Käesolevas töös eeldatakse Ohm'i seadusele (12) tuginedes, et protsessori võimsustarve on ligikaudu lineaarses sõltuvuses sagedusega. Selline lähendus töötab süsteemides, kus protsessori energiatarvet on võimalik muuta läbi taktsageduse muutmise [10].

$$P = C \times V^2 \times f \qquad (12)$$

Valemis on arvutatud protsessori võimsustarve *P* läbi mahtuvuse *C*, toitepinge ruudu V^2 ning sageduse *f*.

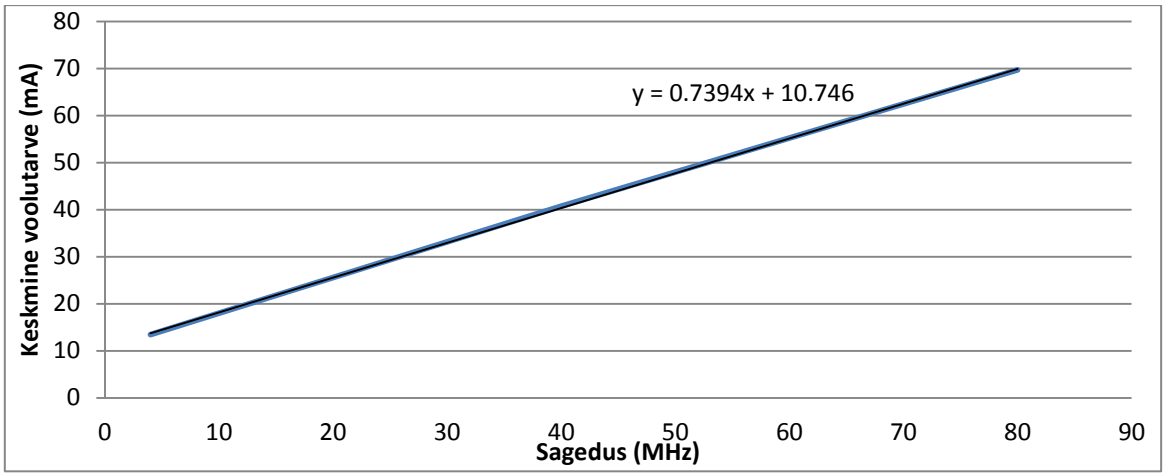
Selline lähenemine võimaldab kasutada minimaalselt kahe baasmõõtmise tulemust lineaarse regressiooni arvutamiseks. Baasmõõtmistel kasutatud sagedusi nimetatakse edasipidi baassagedusteks. Baassagedusi võib olla 2..n, kus n on lõplik positiivne täisarv. Valemi (13) abil arvutatakse uuele sagedusele keskmine voolutarve I_{Kesk}.

$$I_{Kesk} = a + \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} f_N \qquad (13)$$

Millest

- a - regressiooni vabaliige
- x_i - i-ndal sagedusel saadud operatsiooni keskmine voolutarve
- $\bar{x}$ - uuritava operatsiooni keskmine voolutarve baassagedustel
- y_i - baassagedus järjekorranumbriga i
- $\bar{y}$ - baassageduste keskmine
- f_N - sagedus, millele antakse voolutarbe hinnang.

Seega kui on teada näiteks avaldise b = c keskmine voolutarve sagedustel 4MHz, 40MHz ja 80 MHz, on võimalik luua sirge võrrand, mille põhjal tehakse voolutarbe hinnang soovitud sagedusele. Lineaarset regressiooni illustreeriv pilt koos regressioonisirge võrrandiga on nähtav joonisel 8.



Joonis 8. Baasoperatsiooniga b = c testtsükkli voolutarbe lineaarne regressioon.

Oluline on märkida, et joonisel 8 toodud keskmine voolutarbe hulk on mõõdetud ajal, kui programm käitas tervet testprogrammi (joonis 6). Seega mõõdetakse tegelikkuses testprogrammi keskmist voolutarvet ning ka hinnang teistele sagedustele antakse kogu testprogrammi kohta. Baasoperatsiooni osa kogu programmi energiatarbest leitakse valemi (4) abil. See tähendab omakorda, et ka testprogrammi tühikäigule on vaja oma regressiooni võrrandit.

Saadud võrrandi (joonis 8) abil on võimalik arvutada testtsükkli, mis sisaldab baasavaldist b = c, keskmine voolutarve erineva taktsageduste korral. Sama meetodi järgi leitakse ka testtsükkli tühikäigu keskmine voolutarve. Valemis 14 on arvutatud baasoperatsiooniga testtsükkli hinnang sagedusele 40MHz.

$$I_{40MHz} = 0{,}7394 \times 40 + 10{,}746 = 40{,}318\ [mA] \qquad (14)$$

Kombineerides valemid (3,4,10 ja 13) võib leida valemis 15 b = c energiatarbe hinnangu uuele sagedusele 40MHz. Eelnevalt on teostatud kõik mõõtmised toitepingel 3,3V.

$$E_{b=c} = \frac{(3{,}3 \times 40{,}001 \times 1{,}7883) - (3{,}3 \times 40{,}318 \times 1{,}3473)}{1000} = 0{,}0557\ [\mu J] \qquad (15)$$

Seega baasoperatsiooni b = c energiatarbe hinnanguks sagedusel 40MHz on 0,0557µJ. Teostades mõõtmise samal sagedusel on tulemuseks 0,0550µJ. Hinnangu viga on 1.27%. Seega on täidetud eesmärk anda hinnang vähemalt usaldusnivool 0,95.

4. Tulemused ja analüüs

Selles töös on hinnangud antud testprogrammile, mis tegeleb PGM formaadis pilditöötlusega. Algoritmi valik langes kokku varasemalt tehtud magistritööga mitmel põhjusel: esiteks oli testitud algoritmi erinevaid versioone, millest sobis kõige paremini versioon, kus miinimum ja maksimum hinnang olid kõige täpsemad. Seda põhjusel, et hinnata täpsemalt meetodi täiendusest tekkivat viga. Teiseks ei kasuta algoritm väliseid mäluseadmeid ega mooduleid, et pilti tekitada. Selle asemel tekitatakse genereeritakse pilt testprogrammi sees tarkvaraliselt, seejärel sooritatakse pilditöötlus meetodid: Gaussi hägustamine ning servade leidmine.

Baasmõõtmised on tehtud kontrolleri PLL režiimis. Selline otsus võeti vastu seetõttu, et PLL režiimis on kontrolleril kasutatav sageduspiirkond laiem. Lisaks on energeetiliselt mõttekam kasutada kiiremaid taktsagedusi, mida on võimalik saavutada sisemist ostsillaatorit kasutades ainult sagedussüntesaatoriga. Baasmõõtmised on teostatud kolmel taktsagedusel: 4 ,40 ja 80 MHz. Mõõtetulemused ja hinnangud koos vigadega on tabelis 4.

Tabel 4. Testprogrammi baasmõõtmiste hinnatud ja mõõdetud energiatarve.

Sagedus, MHz	**Mõõdetud tegelik, mJ**	**Hinnang, mJ**	**Kõrvalekalle, %**
4	102.95	103.55	-0.58
40	31.43	31.21	0.68
80	27.89	27.23	2.38

Saadud baasmõõtmiste tulemustest võib järeldada, et meetod jääb 2,5% täpsuspiiresse ka faasilukuga sagedussüntesaatorit kasutades. Maksimaalsel lubatud sagedusel on veaks 2,38%. Suurematel sagedustel on oluliseks sagedussüntesaatori poolt väljastatava taktsageduse täpsus ning stabiilsus.

Tootja ei ole kirjeldanud seadme spetsifikatsioonis sagedussüntesaatori täpsust ega stabiilsust. Sellega seoses on vaja mõõta, milliseid sagedusvahemikke sagedussüntesaator väljastab. Selle jaoks on olemas kiibil väljaviik nr 64 nimega CLKO. Ühendades ostsilloskoobi selle klemmiga, on võimalik vaadelda protsessori taktsagedust. CLKO viigu signaali värelemine on spetsifikatsiooni järgi ±0,25%, FRC kõikumine toitepingel 3,3V ±2%. Testsüsteemi ostsilloskoop võimaldas teha mõõtmiste analüüsi, kuvades maksimaalse, minimaalse ning

keskmise taktsageduse väärtuse. Tabelis 5 on teostatud mõõtmised toitepingel 3,3V, PLL režiimis ning mõõdetud taktide arv vähemalt 10000.

Tabel 5. Taktsageduse kõikumine PLL režiimis.

Määratud Sagedus, MHz	**Miinimum, MHz**	**Maksimum, MHz**	**Keskmine, MHz**	**Keskmise viga, %**
4	4.03	4.04	4.04	0.88
10	10.08	10.13	10.10	0.99
20	20.06	20.32	20.20	1.00
30	30.10	30.50	30.28	0.93
40	40.10	40.60	40.41	1.02
48	48.10	48.90	48.50	1.04
60	59.80	61.20	60.58	0.95
72	71.90	74.00	72.81	1.12
80	78.00	84.20	80.65	0.80

Tabelist 5 võib järeldada, et sagedussüntesaatori väljastatav keskmine sagedus ei erine määratud taktsagedustest rohkem kui 1,13%. 80MHz sageduse juures tähendab 0,81% viga 648kHz. See selgitab ka tabelis 4 toodud suurema baasmõõtmise vea sagedusel 80MHz võrreldes 4 ja 40MHz-iga. Seega ei genereeri sisemine RC ostsillaator koos sagedsussüntesaatoriga väga stabiilset taktsagedust ja ei sobi kasutamiseks suurt täpsust nõudvates süsteemides.

Hinnangu andmisel teistele sagedustele on aluseks võetud mõõtetulemused, mis on saadud 4MHz juures. Baasmõõtmistel kasutatud baassageduste vahele on arvutatud hinnangud teistele taktsagedustele. Arvutatud hinnangus on sees olev keskmine voolutarbe regressioonivõrrand, millest lähemalt on juttu peatükis 3.1.2 „Keskmise voolutarbe ümberarvutamine“, on samuti koostatud nende kolme baasmõõtmise põhjal. Arvutatud hinnangud on toodud tabelis 6. Võrdluseks on toodud tabel 7, kus on antud hinnang sagedustele 8 ja 10 MHz nii, et keskmise voolutarbe regressioonivõrrand on saadud kasutades baassagedusi 4 ja 40MHz.

Tabel 6. Testprogrammi energiatarbe hinnangud teistele sagedustele.

Taktsagedus, MHz	**Mõõdetud tegelik, mJ**	**Hinnang, mJ**	**Kõrvalekalle, %**
8	67.60	65.60	-3.04
10	58.73	57.29	-2.51
20	40.87	40.66	-0.52
30	34.47	33.97	1.44
60	29.37	28.61	2.57
72	27.32	27.72	-1.48

Tabel 7. Testprogrammi energiatarbe hinnangud kahe regressioonipunktiga.

Taktsagedus, MHz	**Mõõdetud tegelik, mJ**	**Hinnang, mJ**	**Kõrvalekalle, %**
8	67.60	65.55	-3.12
10	58.73	57.26	-2.57

Üldjoontes võib hinnangute täpsusega jääda rahule, arvestades kasutatud ostsillaatori seadistuse stabiilsust. Püstitatud eesmärk jääda usaldusnivoole 95% sai täidetud. Suurimat mõju hinnangule avaldab keskmise voolutarbe arvutus. Tabelist 6 võib näha, et suurimad erinevused hinnatu ja mõõdetu vahel on regressioonivõrrandi otspunktides. Tuleb märkida, et antud töös on jäetud kõrvale tingimuslausete analüüs, valitud on kõige väiksema eksimusega hinnangud. Valitud testprogrammi juures ei ole tingimuslausetel väga suurt mõju, kõikudes 1% piires [2]. Algoritmides, kus tingimuslauseid täidetakse palju, on vajalik täiendav tingimuslausete analüüs.

Hinnangu erinevused on minimaalsed, kui rakendada mitut erinevat regressioonivõrrandit erinevatele sagedusvahemikele. Tabelis 7 olevad vead ei ületa tabelis 6 samadel sagedustel tekkivaid rohkem kui 0.08%. Kahe regressioonimudeli kasutamine tõi kaasa hoopis hinnanguvea suurenemise. Selline tulemus oli mitmeti oodatav, sest taktsageduse ning voolutarve vahel oli tugev positiivne korrelatsioon. Kolme baasmõõtmiste põhjal tehtud voolutarbe regressioonimudelite determinatsioonikordajad R^2 jäid vahemikke 0,9998 - 0,9999. Antud tulemuste pärast ei tasuks aga veel kõrvale heita võtet teha erinevatele sagedusvahemikele oma regressioonivõrrand. Testsüsteemis kasutatav sagedusvahemik ning protsessor vastasid kokkusattumusel lineaarsele mudelile väga hästi.

4.1. Meetodi edasiarendused

Uuritav meetod energiatarbe hindamiseks on alles väljatöötamisel, ning lahendamist vajavad mitmed probleemid.

Töös pakutud hinnangute ümberarvutamine erinevatele sagedustele võiks saada täiendusi, kasutades stabiilsemat protsessorkiibivälist ostsillaatorit. Sisemine FRC on kergesti mõjutatav temperatuuri poolt, lisaks on sagedussüntesaatori väljastatav sagedus väga kõikuv. Väline, täpsem ning stabiilsem ostsillaator võib hinnangu täpsust märgatavalt parandada.

Lineaarse regressioonimudeli kasutamine keskmise voolutarbe arvutamiseks vajab edasist tõestamist. Selles töös kasutatud PIC32 mikrokontroller lubab dünaamilist sagedusskaleerimist (DFS). Protsessorid, mis kasutavad dünaamilist pingeskaleerimist (DVS) ei allu lineaarsele sagedus-võimsus mudelile. Samuti tuleks teha kindlaks, millistes sagedusvahemikes lineaarse regressiooni kasutamine on lubatav. Peale selle võiks veelgi teostada baasmõõtmisi mille järgi regressiooni arvutatakse, et näha, kas hinnangutäpsus paraneb.

Meetodi täpsust võivad mõjutada erinevad protsessori energiasäästurežiimid. Käesolevas töös olid väljalülitatud valvetaimerid ning kasutatav taktsagedus püsis testprogrammi käitamise ajal konstantne. Lisaks ei olnud kontroller seadistatud programmi käitamise ajal minema ooterežiimi. Paljud reaalselt rakendatavad programmid aga kasutavad eelnimetatud seadistusi energiatarbe vähendamiseks.

Meetodi paikapidavust tuleks kontrollida erinevatel seadmetel. Praeguseks on tõestatud, et meetod annab üsnagi häid tulemusi PIC32 seeria mikrokontrolleritel. Sarnased mõõtmised tuleks teostada ka teiste tootjate, näiteks Texas Instrumentsi, Inteli või Silicon Labsi kontrolleritega, millede protsessorid on teistsuguste tuumade ja ülesehitusega. Üheks hinnangumeetodi kasutusvaldkonnaks võiksid olla ka üheplaadi arvutid, nagu näiteks Raspberry Pi ning Arduino erinevad mudelid.

Baasmõõtmiste teostamine tuleks kindlasti automatiseerida. Kasutatud testprogrammis oli 56 C keele avaldist. Need jagunesid 142 eraldiseisvaks baasoperatsiooniks. Iga baasoperatsiooni energiatarbe hindamine nõuab kahte testprogrammi, koos mõõdetava operatsiooniga ja ilma. Seega tuli kontrollerit ühe seadistusega baasmõõtmiste saamiseks 284 korda programmeerida. Sealjuures tuleb igal korral kood kompileerida, seade programmeerida, mõõta keskmine

energiatarve ja ajaline kestus, ning saadud tulemused üles märkida. Keskmiselt läheb sellise mahuga programmi energiatarbe hindamiseks operaatoril aega umbes 7-8 tundi. Suuremahuliste andmebaaside koostamiseks erinevate protsessorite ja seadistuste kohta, mis võib osutuda selle meetodi laiendamisel üliloluliseks, eeldab seega mõõtmiste programmset teostust.

5. Kokkuvõte

Töös on välja pakutud meetodid energiatarbe hinnangu ümberarvutamiseks erinevatele protsessori taktsagedustele. Viimaseid genereeriti kontrolleri sees oleva RC ostsillaatori ning faasilukuga sagedussüntesaatori vahendusel. Vaatluse all olid põhiliselt kaks sagedusega seotud muutujat energiatarbe hindamise valemis: käsu ajaline kestus ning keskmine voolutarve käsu täitmisel. Selgus, et mõlemad tegurid on taktsagedusega seotud teatud tingimustel lineaarselt. Hinnangud anti sagedusvahemikes 4-80 MHz, seadme toitepingeks oli 3,3V.

Tootjapoolset sagedus-voolutarbe karakteristikut ei olnud antud, seega tuli leida töös meetod selle leidmiseks. Keskmise voolutarve suhe erinevate sagedustega osutus töös kasutatud testsüsteemil järgivat lineaarset regressioonivõrrandit. Teostades kaks või enam baasmõõtmist, on võimalik arvutada oletatav keskmine voolutarve ka teistele sagedustele. Käsu ajaline kestus on lineaarses seoses taktsagedusega ning baasmõõtmise olemasolul on lihtne leida sama käsu ajalist kestust ka teistel sagedustel.

Töö eesmärgiks seatud hinnangute arvutamine usaldusnivool 95% sai täidetud. Mõõdetud tulemuste ning hinnangute vahe ei ületanud 3%. Hinnangute täpsust mõjutasid ka taktgeneraatori komponendid. Selgus, et faasilukuga sagedussüntesaator, saades oma sisendallika kiibisiseselt RC ostsillaatorilt, väljastab taktsignaali, mis keskmiselt erineb soovitud seadistusest kuni 1,13%. Hinnangu täpsuse parandamiseks võiks seega kasutada välist stabiilsemat ning täpsemat ostsillaatorit.

Meetodi täiendamiseks tuleks mõõtmisi teostada erinevate käsustike ning mikroarhitektuuridega protsessorite peal. Töös saadud hinnangud on mõõdetud ühe taktgeneraatori seadistusega, seega peaks eraldi kontrollima hinnangute täpsust seadistuse vahetamisel. Samuti võiks lülitada süsteemi juurde ühekaupa erinevaid mooduleid – energiasäästurežiimid, vahemälu, erinevad sisend/väljund seadmed. Meetodi laiemaks kasutuselevõtuks peaks mõõtmiste teostamise automatiseerima, sealjuures peaks programm leidma koodikatte ning analüüsima tingimuslausete täitmist.

Kasutatud kirjandus

[1] PIC32 Family Reference Manual, Sect. 06 Oscillators. 2011.[Online] http://ww1.microchip.com/downloads/en/DeviceDoc/61112H.pdf (02.06.2015)

[2] Kiil, M. Tarkvara komponendi energiatarbe hindamine : magistritöö. Tallinna Tehnikaülikool, Tallinn. 2013.

[3] PIC32MX3XX/4XX Data Sheet. 2011.[Online] http://ww1.microchip.com/downloads/en/DeviceDoc/61143H.pdf (02.06.2015)

[4] PIC32MX Oscillator – Developer Help. 2015. [WWW] http://microchip.wikidot.com/32bit:mx-osc (02.06.2015)

[5] PICkit 3 In-Circuit Debugger/Programmer User's Guide. 2013. [Online] http://ww1.microchip.com/downloads/en/DeviceDoc/52116A.pdf (02.06.2015)

[6] Hameg HMP Datasheet. [Online] http://www.hameg.com/datasheets.0.html?&no_cache=1&tx_hmdownloads_pi1[mode]=download&tx_hmdownloads_pi1[uid]=7294 (02.06.2015)

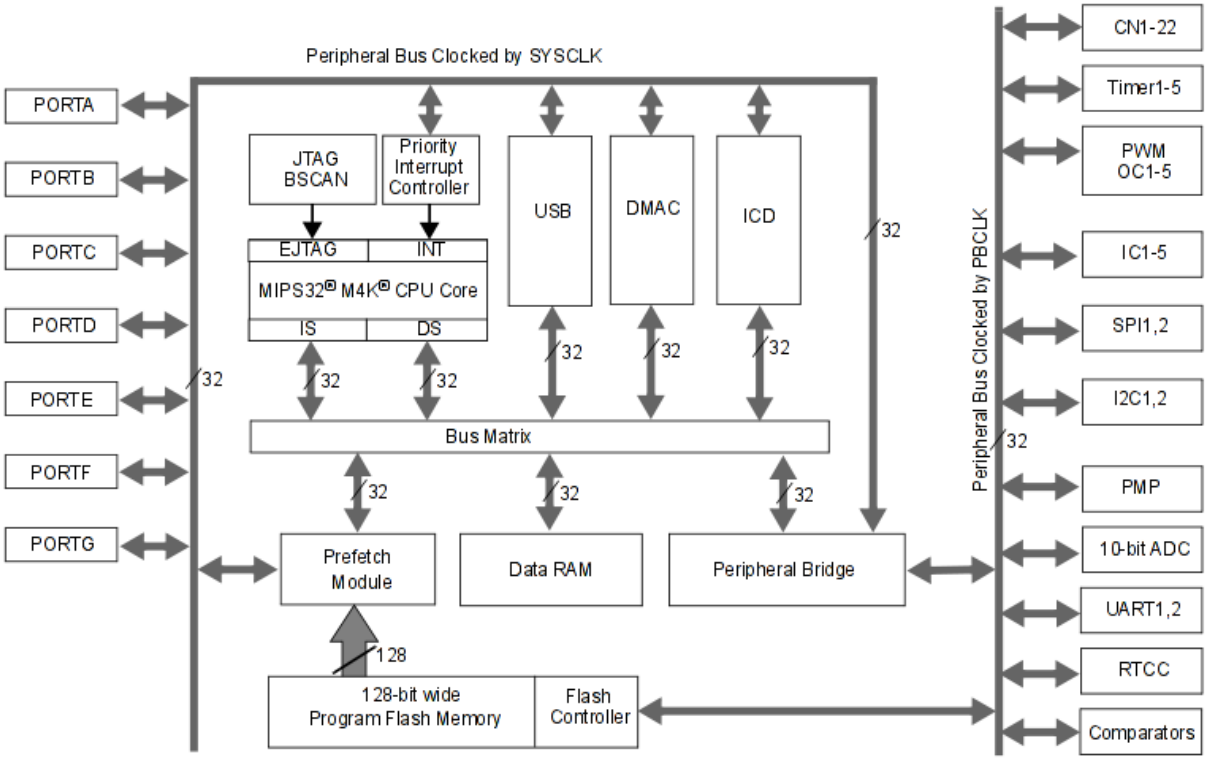
[7] Agilent 34405A 5 ½ Digit Multimeter User's and Service Guide. [Online] http://cp.literature.agilent.com/litweb/pdf/34405-91000.pdf (02.06.2015)

[8] Intronix LogicPort PC-Based Logic Analyzer with USB Interface. [WWW] http://www.pctestinstruments.com/logicport/specifications.htm (02.06.2015)

[9] Agilent InfiniiVision 3000 X-Series Oscilloscopes User's Guide. [Online] http://web.mit.edu/6.115/www/document/agilent_mso-x_manual.pdf (02.06.2015)

[10] A. Miyoshi, C. Lefurgy, E.V. Hensbergen, R. Rajamony, R. Rajkumar. (2002). Critical power slope: understanding the runtime effects of frequency scaling. - *ICS '02 Proceedings of the 16th international conference on Supercomputing*, 35-44

Lisa 1



PIC32 üldskeem. [1]