AALTO-YLIOPISTO
SÄHKÖTEKNIIKAN KORKEAKOULU
Mikro- ja nanotekniikan laitos

Mika Pulkkinen

Digitaalinen signaalinkäsittelylohko ja synkronoija anturimikropiiriin

Diplomityö, joka on jätetty opinnäytteenä tarkastettavaksi diplomi-insinöörin tutkintoa varten Espoossa 26.5.2014.

Työn valvoja:

Professori Kari Halonen

Työn ohjaaja:

TkT Lasse Aaltonen

AALTO-YLIOPISTO
SÄHKÖTEKNIIKAN KORKEAKOULU

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä: Mika Pulkkinen

Työn nimi: Digitaalinen signaalinkäsittelylohko ja synkronoija anturimikropiiriin

Päivämäärä: 26.5.2014 **Kieli:** Suomi **Sivumäärä:** 9+72

Laitos: Mikro- ja nanotekniikan laitos

Professuuri: Piiritekniikka **Koodi:** S-87

Työn valvoja: Professori Kari Halonen

Työn ohjaaja: TkT Lasse Aaltonen

Tässä diplomityössä suunniteltiin ja toteutettiin tutkimuskäyttöön tarkoitetulle, kaksiakseliselle kiihtyvyysanturipiirille digitaaliset lohkot, joihin kuului kaksi desimoivaa CIC-suodatinta, SPI-tiedonsiirtoväylä, muistirekisterit säätöbittejä varten, tasapoikkeaman- ja vahvistuksenkorjaimet kiihtyvyyslukemille sekä synkronoija.

Piirin AD-muuntimilta saadaan kahden akselin kiihtyvyyksien suuruus yksibittisenä digitaalisena datana. Desimointisuodattimet kasvattavat datan sananleveyttä, suodattavat sitä ja pienentävät sen näytteistystaajuutta. Vaatimuksena oli, ettei kiihtyvyysdatan signaali-kohinasuhde saa pienentyä merkittävästi laskostumisen takia. SPI-väylällä puolestaan mahdollistettiin analogisten ja digitaalisten lohkojen toimintaa säätävien bittien syöttäminen piirille ja digitaalisen kiihtyvyysdatan lukeminen ulkoisen lukijan, esimerkiksi mikrokontrollerin tai tietokoneen, avulla. Kiihtyvyyslukeman tasapoikkeaman ja vahvistuksen korjauksella saadaan pienennettyä piiriyksilöiden antamien kiihtyvyyslukemien välisiä eroja ja synkronoijalla vähennetään merkittävästi asynkronisesta kiihtyvyysdatan lukemisesta aiheutuvia lukemavirheitä.

AD-muuntimilta tulevan kiihtyvyysdatan näytteistystaajuus on 100 kHz ja CIC-suodattimet pudottavat sen 100 Hz:iin. Molempien CIC-suodattimien lähdössä sananleveys on 31 bittiä, joka lyhennetään 24-bittiseksi ennen poikkeaman ja vahvistuksen korjaamista. Digitaalilohkot toteutettiin 0,35 μm:n CMOS-teknologialla ja niiden käyttöjännite on 3,3 V. Digitaalisolujen käyttämä kokonaispinta-ala on 0,60 mm^2 ja lohkojen kokonaisvirrankulutus on 12 μA, kun molempien akselien 24-bittisiä kiihtyvyysarvoja luetaan SPI-väylän kautta nopeudella 100 näytettä per sekunti.

Avainsanat: integroitu piiri, mikropiiri, VHDL, synteesi, desimointi, CIC-suodatin, DSP, SPI, synkronointi, metastabiilisuus, ASIC, CMOS

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Mika Pulkkinen

Title: Digital signal processing block and synchronizer for a sensor microcircuit

Date: 26.5.2014 **Language:** Finnish **Number of pages:** 9+72

Department: Department of Micro- and Nanosciences

Professorship: Electronic Circuit Design **Code:** S-87

Supervisor: Professor Kari Halonen

Instructor: D.Sc. Lasse Aaltonen

In this Master's thesis, digital integrated circuit blocks were designed and processed for an integrated 2-axis accelerometer microcircuit for a research project. Implemented blocks consist of two decimating CIC filters, an SPI communication interface, memory registers for control bits, a synchronizer and adders and a shared multiplier for correcting the offset and gain of the acceleration outputs.

The analog-to-digital converters of the accelerometer system produce 1-bit output data streams. The CIC filters increase the word lengths of the data, filter the noise and decrease the sampling rates. The SPI interface enables writing and reading of control bits for the analog and digital blocks and reading of the acceleration data e.g. with a microcontroller. By using the offset and gain correction, variation of outputs of a set of multiple accelerometer circuits can be decreased. The synchronizer significantly decreases the probability of occurrence of erroneous acceleration values received by an SPI master device when acceleration data is read through the SPI interface asynchronously.

Sampling frequency of the 1-bit data from the analog-to-digital converters is 100 kHz. The CIC filters decrease the sampling frequency to 100 Hz and increase the word lengths to 31 bits. The word lengths are truncated to 24 bits and the offsets and gains are corrected by the adders and the shared Booth multiplier. The accelerometer system with the digital block was processed in a 0.35 μm CMOS process, whose nominal supply voltage is 3.3 V. The area of the digital core is 0.60 mm^2. Current consumption of the implemented digital blocks is 12 μA, when CIC filters are running and all 24 acceleration data bits of both axes are read at the speed of 100 samples per second.

Keywords: integrated circuit, VHDL, synthesis, decimation, CIC filter, DSP, SPI, synchronization, metastability, ASIC, CMOS

Esipuhe

Arvoisa lukija,

Edessäsi oleva opinnäytetyö lienee yksiä kauimmin valmisteilla olleita diplomitöitä. Aloitin työn suunnitteluosan tekemisen vuonna 2010, siis jo neljä vuotta sitten, jolloin myös tekstin ensimmäiset rivit näkivät päivänvalon. Suunnitelmissani oli saada työ valmiiksi vuodessa ja suorittaa loput DI-tutkintoon kuuluvat kurssit kesään 2012 mennessä. Kesän 2010 alussa perheenjäseneni kuitenkin sairastui vakavasti. Alkoi puolitoista vuotta kestänyt ajanjakso, jonka aikana se ja muutamat muut tapahtumat, joista kaikkine käänteineen riittäisi kerrottavaa kirjaksi asti, järisyttivät niin minun kuin koko pienen perhepiirini maailmaa ja elämämme muuttuivat kenties pysyvästi. Oman haasteensa noihin aikoihin toivat maantieteelliset etäisyydet perheeni jäsenten välillä. Olosuhteiden takia katsoin parhaaksi tehdä palkkatöitä opiskelemisen sijaan, kunnes elämä asettuisi taas raiteilleen, ja koulun penkille palaaminen sai odottaa lopulta liki 2,5 vuotta. Kaiken tuolloin kokemamme jälkeen ei ole lainkaan itsestäänselvyys, että olen päässyt tähän pisteeseen, valmistumisen kynnykselle, vaan siitä kuuluu kiitos elämässäni oleville lämminhenkisille ja hyväsydämisille ihmisille.

Haluan kiittää ensimmäisenä vanhempiani Kalervoa ja Kirstiä ja siskoani Katjaa siitä tavasta, jolla he ovat kohdanneet, ja auttaneet minua kohtaamaan, eteemme asetetut haasteet. Heidän ansiostaan on ollut mahdollista selvitä mitä vaikeimmistakin asioista. Mitä opiskeluihin tulee, haluan kiittää koko perhettäni kannustuksesta ja tukemisesta opiskeluaikoina ensimmäiseltä luokalta näihin päiviin asti. Lisäksi haluan kiittää isääni siitä, että hän taannoin vihjasi, että silloinen Teknillinen korkeakoulu, eli nykyinen Aalto-yliopisto, saattaisi olla otollinen opiskelupaikka. Täältä toden totta on löytynyt mielenkiintoista opiskeltavaa, ja kurssien käynti on johdattanut minut äärimmäisen mielenkiintoisten työtehtävien pariin mikropiirien kiehtovaan maailmaan.

Toiseksi haluan kiittää työni valvojaa prof. Kari Halosta ja työni ohjaajaa TkT Lasse Aaltosta loistavasta esimiestyöstä, joustavuudesta haastavina aikoina ja siitä, että he ovat sallineet tämän työn kypsyä tähän asti ja antaneet ykkösluokan palautetta vielä kaikkien näiden vuosien jälkeen. Prof. Haloselle suurkiitos siitä, että hän otti minut kesätöihin vuonna 2008 silloiseen piiritekniikan laboratorioon ja siitä, että olen saanut jatkaa työskentelyä ja uuden oppimista täällä näihin päiviin asti. Olen heille molemmille erityisen kiitollinen työtehtävistä, jotka he ovat osoittaneet minulle kuluneina vuosina. Ne ovat olleet erittäin mielenkiintoisia ja mukavia – jotkin tehtävät, esimerkiksi alkuaikojen Labview-ohjelmointitehtävät ja lukuisat suunnittelutehtävät, jopa niin mukavia, että välillä on melkein ihmetyttänyt, miksi minun ei tarvitse maksaa niiden tekemisestä mitään. Kiitokset heille myös asiallisen ja ammattimaisen, mutta siltikin kotoisan ja rennon työympäristön luomisesta.

Kolmanneksi haluan kiittää kaikkia piiritekniikan tutkimusryhmän tutkijoita, tutkijanalkuja ja muuta henkilöstöä positiivisen työilmapiirin luomisesta. On ollut ilo ja kunnia työskennellä näin hauskojen ja terävien ihmisten kanssa. Alallamme ei liene ongelmaa, jota tällä porukalla ei saataisi ratkaistua. Erityisen suuri megakiitos inspiroivan ja legendaarisen hauskan työilmapiirin luomisesta kuuluu opiskelu- ja työhuonetoverilleni DI Jarno Salomaalle, jonka huumorintaju on pelastanut monet päivät kuluneiden vuosien varrella, ja jonka vertaansa vailla oleva piirisuunnitteluntaju on puolestaan pelastanut lukuisat ongelmanratkaisutilanteet erityisesti analogiaelektroniikkasuunnitteluun liittyen. Haluan kiittää myös TkT Mikail Yücetaşia, DI Jakub Groniczia ja DI Antti Kalantia yhteistyöstä projekteissa, avusta analogiaelektroniikkasuunnitteluun perehtymisessä ja hetkistä tuopperoisten äärellä Gallows Birdissä, joista jälkimmäisistä kiitokset myös Lasselle ja Jarnolle. Suurille digitaaliguruille eli TkT Marko Kosuselle, TkT Lauri Koskiselle, DI Vesa Turuselle, DI Matthew Turnquistille ja DI Erkka Laulaiselle kiitokset vastauksista digitaalipiireihin ja VHDL-koodaamiseen liittyviin kysymyksiin. Lisäksi Tuomakselle ja Shivalle vielä kiitokset I313b:n hengen ylläpitämisestä, Artturi Kailalle tietokonepulmissa avustamisesta sekä Lea Södermanille, Arja Hjeltille, Helena Yllölle ja Anja Meuroselle toimistoasioissa avustamisesta.

Lopuksi haluan esittää kiitokseni ystävilleni. Elämän kolhiessa he ovat jaksaneet kysellä kuulumisia ja tarjonneet työ- ja muiden haasteiden vastapainoksi rattoisaa ajanvietettä. Ilman heitä tästä työstä ei olisi taatusti tullut valmista. Erityiskiitokset Henkalle, jonka kanssa aloitimme opintaipaleen samaan aikaan ja samassa paikassa 23 vuotta sitten. Mieltäni lämmittää suunnattomasti, että ystävyytemme on säilynyt kaikki nämä vuodet. Erityiskiitokset myös Hanna M:lle ja Minnalle tukikeskusteluista IRC:ssä silloin, kun elämä löi luun kurkkuun, ja kauniille ja upealle Hennalle siitä, että hän toi auringonpaisteen takaisin risukasaani. Heille ja kaikille muillekin tovereilleni haluan esittää valtaisat kiitokset kaikista vuosien aikana koetuista lukuisista seikkailuista ja yhteisistä hetkistä niin virtuaali- kuin oikeassakin maailmassa.

Tämä diplomityö on tehty osana anturiprojekteja, joita ovat rahoittaneet Murata Electronics Oy, aiemmin VTI Technologies Oy, ja teknologian ja innovaatioiden kehittämiskeskus Tekes. Niin näiden kuin muidenkin kuluneiden vuosien aikana toteutettujen mielenkiintoisten projektien parissa tehdystä yhteistyöstä haluan esittää kiitokseni Muratan työntekijöille DI Tero Sillanpäälle, TkT Teemu Salolle, DI Teemu Elolle, FT Pasi Kiviselle ja DI Risto Mourujärvelle.

Espoossa 22.5.2014

Mika Pulkkinen

Sisällysluettelo

Symbolit ja lyhenteet

Symbolit

τ	Ratkaisuaikakavakio (resolving time constant)
a	Kiihtyvyys
f_s	Näytteistystaajuus (sampling frequency)
g	Painovoimakiihtyvyys, 9,81 m/s^2
R	Taajuudenpudottamissuhde
T_0	Metastabiilisuusikkuna (metastability window)
t_{hold}	Pitoaika (hold time)
t_p	Etenemisviive (propagation delay)
t_{setup}	Asetusaika (setup time)

Lyhenteet

AD	Analogia-digitaali (analog to digital)
BIBO	Äärellinen tulo, äärellinen lähtö (bounded-input bounded-output)
CIC	Cascaded Integrator-Comb-suodatintyyppi
CMOS	Komplementaarinen metallioksidipuolijohde (complementary metal-oxide semiconductor)
DRC	Piirikuvion piirustussääntöjenmukaisuuden tarkastus (design rule check)
DSP	Digitaalinen signaalinkäsittely (digital signal processing)
FIR	Äärellinen impulssivaste (finite impulse response)
GPIB	General Purpose Interface Bus-rinnakkaisväylä
I2C	Inter-Integrated Circuit-standardiväylä
IC	Mikropiiri (integrated circuit)
IIR	Ääretön impulssivaste (infinite impulse response)
IO	Tulo-lähtö (input-output)
LPT	Line Print Terminal-rinnakkaisväylä
LSB	Vähiten merkitsevä bitti binääriluvussa (least significant bit)
LVS	Piirikuvion ja -kaavion yhdenmukaisuuden tarkastus (layout versus schematic)
MEMS	Mikroelektromekaaninen järjestelmä (micro-electro-mechanical system)
MIDI	Musical Instrument Digital Interface-väylä
MISO	Datalinja orjalta isännälle SPI-väylässä (master input, slave output)
MOMI	Kaksisuuntainen datalinja SPI-väylässä (master output, master input)
MOSI	Datalinja isännältä orjalle SPI-väylässä (master output, slave input)
MSB	Eniten merkitsevä bitti binääriluvussa (most significant bit)
MTBF	Virheiden keskimääräinen aikaväli (mean time between failures)

PCB	Piirilevy (printed circuit board)
SCK	Kellolinja SPI-väylässä (serial clock)
SCL	Kellolinja I2C-väylässä (serial clock)
SDA	Datalinja I2C-väylässä (serial data)
SNR	Signaali-kohinasuhde (signal-to-noise ratio)
SPI	Serial Peripheral Interface-standardiväylä
SPS	Näytettä per sekunti (samples per second)
SS	Orjanvalitsinlinja SPI-väylässä (slave select)
USB	Universal Serial Bus-väylä
VHDL	VHSIC-laitteistokuvauskieli (VHSIC hardware description language)
VHSIC	Erittäin nopeat integroidut piirit (very-high-speed integrated circuits)
WL	Sananleveys (word length)

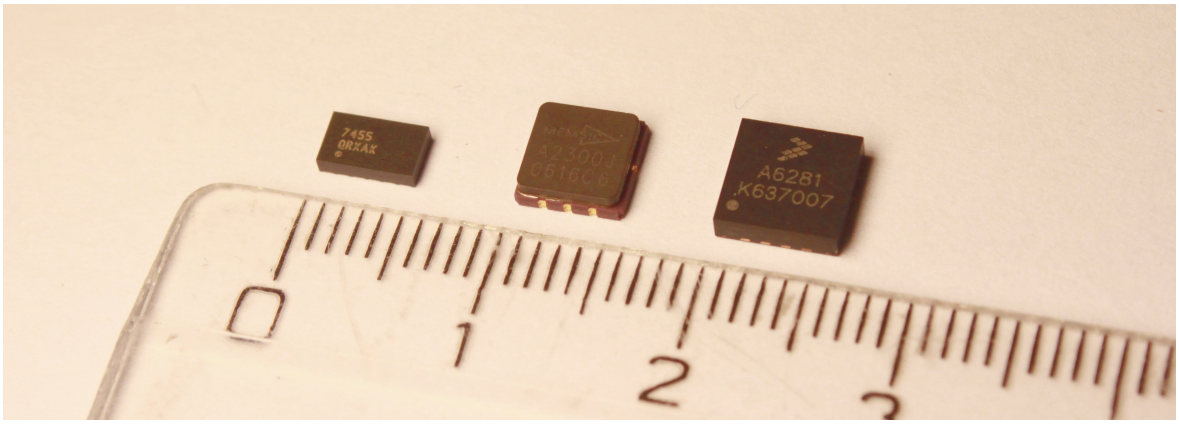
Luku 1

Johdanto

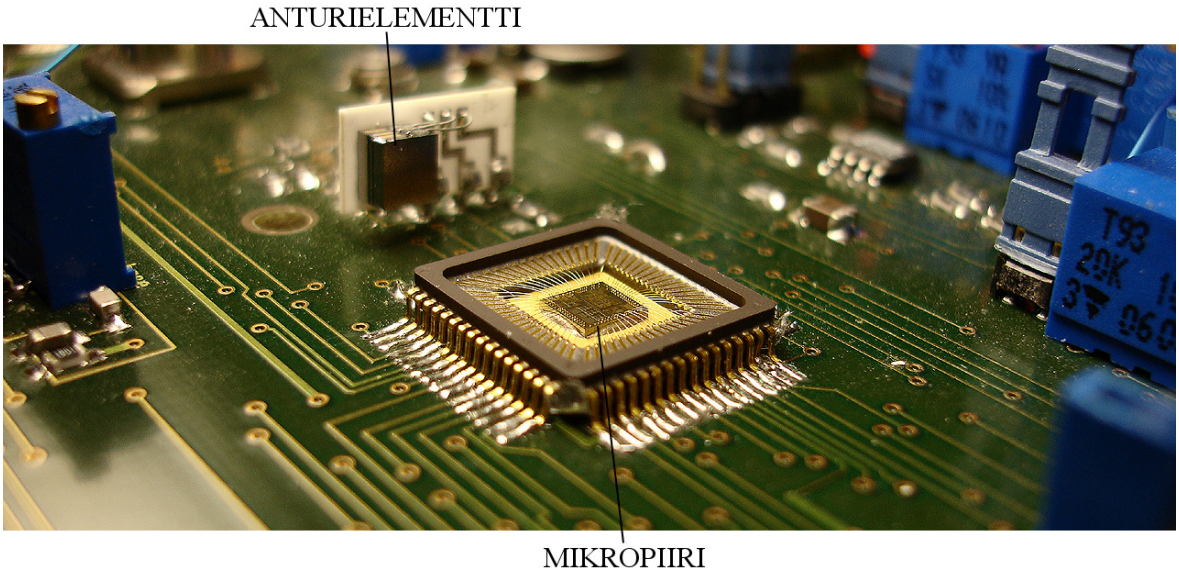
Anturipiirit ovat mikropiirejä, jotka antavat tarkkailtavan suureen, esimerkiksi kiihtyvyyden, kulmanopeuden, lämpötilan tai paineen, sähköisessä muodossa, esimerkiksi jännitteenä, pulssileveysmoduloituna signaalina tai digitaalisena lukuna. Tuotteistettuja anturimikropiirejä saa pieniin koteloihin pakattuna, joiden mitat voivat olla jopa vain muutamien millimetrien luokkaa. Kokonsa puolesta niitä saadaan mahdutettua esimerkiksi matkapuhelimiin, urheilukelloihin ja peliohjaimiin. Kuvassa 1.1(a) on esitetty muutamia kaupallisia kiihtyvyysantureita [1]. Antureiden monipuolisten käyttökohteiden takia niiden markkina-arvo on suuri. Esimerkiksi kiihtyvyys- ja kulmanopeusantureita valmistettiin 2,7 miljardia kappaletta vuonna 2012, jolloin niiden markkinat olivat 2,5 miljardin dollarin suuruiset, ja markkinoiden odotetaan kasvavan yli 5 miljardin dollarin vuoteen 2018 mennessä.[2]

Kiihtyvyys- ja kulmanopeusantureita on mm. kulkuneuvoissa, erilaisissa liikkeeseen perustuvissa peliohjaimissa ja kännyköissä. Kiihtyvyysanturin avulla voidaan esimerkiksi havaita auton törmäys ja laukaista sen perusteella turvatyynyt. Näiden liikeanturien avulla mahdollistuvat myös kännyköiden ja pelikonsoleiden pelien ohjaaminen peliohjaimen liikkeen avulla. Sen lisäksi esimerkiksi ihmisen kehon liikkeitä voidaan tallentaa sijoittamalla useita liikeantureita eri puolille vartaloa, mitä voidaan hyödyntää esimerkiksi elokuva- ja peliteollisuudessa erilaisten hahmojen realististen liikkeiden animoimisessa.[3] Paineantureita voidaan puolestaan käyttää mittaamaan esimerkiksi äänenpaine mikrofonisovelluksessa [4], urheilukellossa vuorikiipeilijän korkeus [5], renkaan ilmanpaine [6] tai lääketeollisuudessa verenpaine [7]. Magneettisen anturin avulla voidaan puolestaan toteuttaa digitaalinen kompassipiiri ja lämpötila-anturilla voidaan mitata esimerkiksi sisätilojen lämpötilat rakennuksissa.

Anturimikropiiri toimii usein rajapintana varsinaisen havainnoivan anturielementin ja jonkin elektronisen järjestelmän välissä. Kuvassa 1.1(b) on esimerkki piirilevyllä olevasta anturijärjestelmästä, jossa mikropiiri ja anturielementti on koteloitu erikseen. Esimerkiksi kapasitiivinen MEMS-anturielementti (MEMS, engl. *micro-electro-mechanical system*) ilmaisee havaittavan suureen määrän kapasitanssina. Mikropiirin kapasitanssi-jännite- tai kapasitanssi-virtamuunninlohkon avulla kapasitanssimuoto voidaan muuttaa jännitteeksi tai



(a) **Kaupallisia kiihtyvyysanturikomponentteja, joissa havainnoiva MEMS-anturielementti ja anturimikropiiri on pakattu samaan koteloon.[1]**

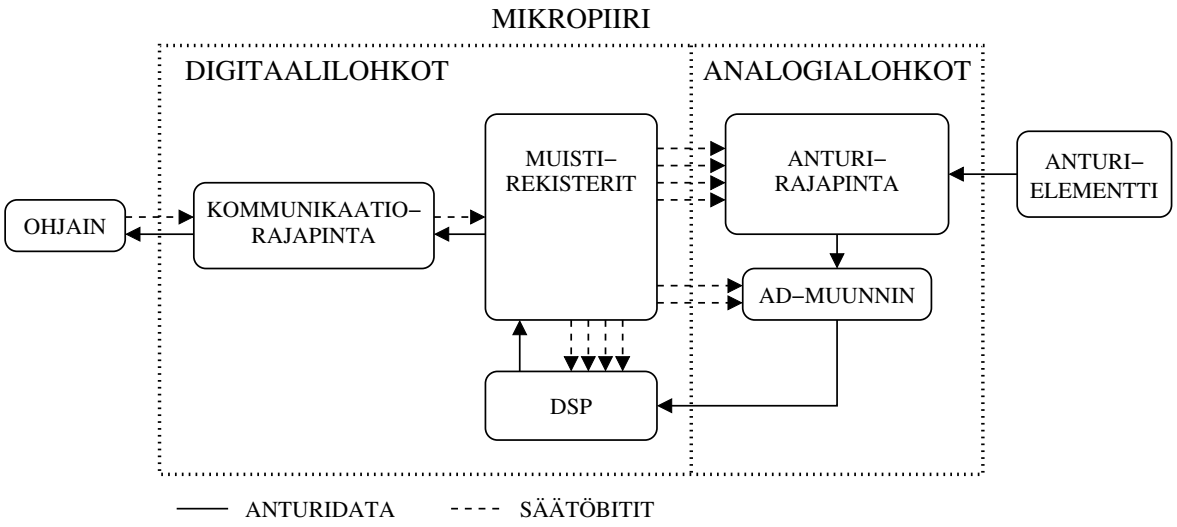


(b) **Anturimikropiiri, erillinen anturielementti ja testaukseen liittyviä elektroniikkakomponentteja piirilevyllä.**

Kuva 1.1: Anturimikropiirejä.

virraksi ja analogia-digitaalimuuntimen avulla edelleen digitaaliseen muotoon jatkokäsittelyä, esimerkiksi suodatusta, sananleveyden (engl. *word length*) muuttamista tai poikkeaman ja vahvistuksen korjaamista, varten. MEMS-elementti voidaan valmistaa suoraan piisirulle yhteen elektroniikkalohkojen kanssa, mikä pienentää järjestelmän tilavaatimuksia.

Osa signaalinkäsittelystä, eli suodatus sekä poikkeaman ja vahvistuksen korjaaminen, voitaisiin toteuttaa myös analogisesti. Havaintolukema muutetaan kuitenkin usein digitaaliseen muotoon mikropiirillä ja synteesipohjaisen digitaalilohkojensuunnitteluvuon ansiosta monimutkaisempienkin digitaalisten korjausfunktioiden ja muiden järjestelmien toteuttaminen AD-muuntimen ohelle on helpottunut ja nopeutunut huomattavasti. Synteesipohjainen suunnitteluvuo voi nopeuttaa piirin suunnittelu- ja toteutusvaihetta ja pienentää siten suunnittelukuluja. Digitaalinen signaalinkäsittelylohko voi myös viedä vähemmän pinta-alaa kuin analoginen, mikä puolestaan voi pienentää piirin valmistuskustannuksissa.



Kuva 1.2: Anturipiirin yksinkertaistettu lohkokaavio.

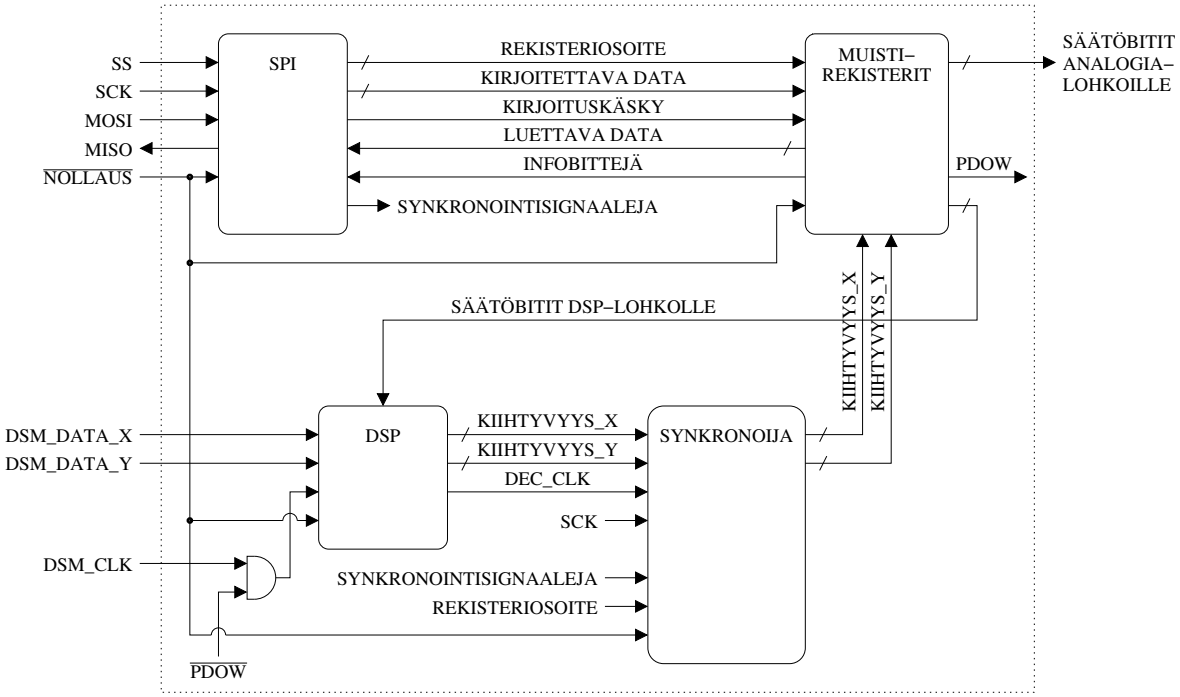
Kuvassa 1.2 on esitetty anturipiirin yksinkertaistettu lohkokaavio. Oikealla oleva anturirajapinta muuntaa anturielementillä olevan havaintotiedon, esimerkiksi kapasitanssin tai resistanssin, sopivaan elektroniseen muotoon. AD-muunnin muuntaa tiedon edelleen digitaaliseen muotoon. DSP-lohko käsittelee datan ja se tallennetaan muistirekistereihin, joista käyttäjä voi hakea sen kommunikaatiorajapinnan kautta. Kommunikaatiorajapinnan kautta voidaan myös tallentaa muistirekistereihin säätöbittejä muille lohkoille, joiden avulla voidaan muuttaa esimerkiksi DSP-lohkon vahvistus ja poikkeamankorjausvakio (engl. *offset correction value*) tai valita sopivat kellotaajuudet eri lohkoille. Ohjain voi olla esimerkiksi mikrokontrolleri, FPGA-piiri tai tietokone.

Tässä työssä toteutettiin digitaalilohkot tutkimuskäyttöön tarkoitetulle, kaksiakseliselle kiihtyvyysanturimikropiirille, joka on suunniteltu tarkkaa kallistuskulman mittaamista varten. Toteutetut lohkot sisältävät SPI-tiedonsiirtoväylän, kaksi identtistä desimointisuodatinta, muistirekisterit säätöbittejä varten, kiihtyvyysarvon poikkeaman ja vahvistuksen korjaimet ja synkronoijan. SPI-väylän vaatimuksena oli, että sen tulee toimia 8 MHz:n kellotaajuudella. CIC-suodattimen vaatimuksena oli, että taajuudenpudotussuhteen tulee olla noin 1000, koska molempien akseleiden AD-muuntimilta tulevan datan näytteistystaajuus on noin 100 kHz ja piirin lähdössä näytteistystaajuuden tulee olla 100 Hz. Muistirekisterit tuli toteuttaa 128:lle analogialohkojen säätöbitille ja lisäksi molempien akseleiden digitaalisille poikkeaman- ja vahvistuksenkorjausarvoille ja CIC-suodatinten kellonjakosuhteelle.

Työn digitaalilohkot tehtiin kiihtyvyysanturipiiriä varten, mutta vastaavanlaisia digitaalilohkoja käytetään myös muun tyyppisissä anturi- ja muissa piireissä, joiden toimintaa halutaan ohjata digitaalisesti tai joiden halutaan antavan digitaalista lähtödataa, esimerkiksi kulmanopeusanturipiireissä, AD-muuntimissa ja muistipiireissä. Toteutettu järjestelmä ylätasolla esitetään kappaleessa 2 ja alilohkojen teoria ja toiminta esitetään kappaleissa 3-7. Mittaustulokset esitetään kappaleessa 8 ja käytetyt menetelmät esitellään kappaleessa 9.

Luku 2

Toteutettu digitaalipiiri



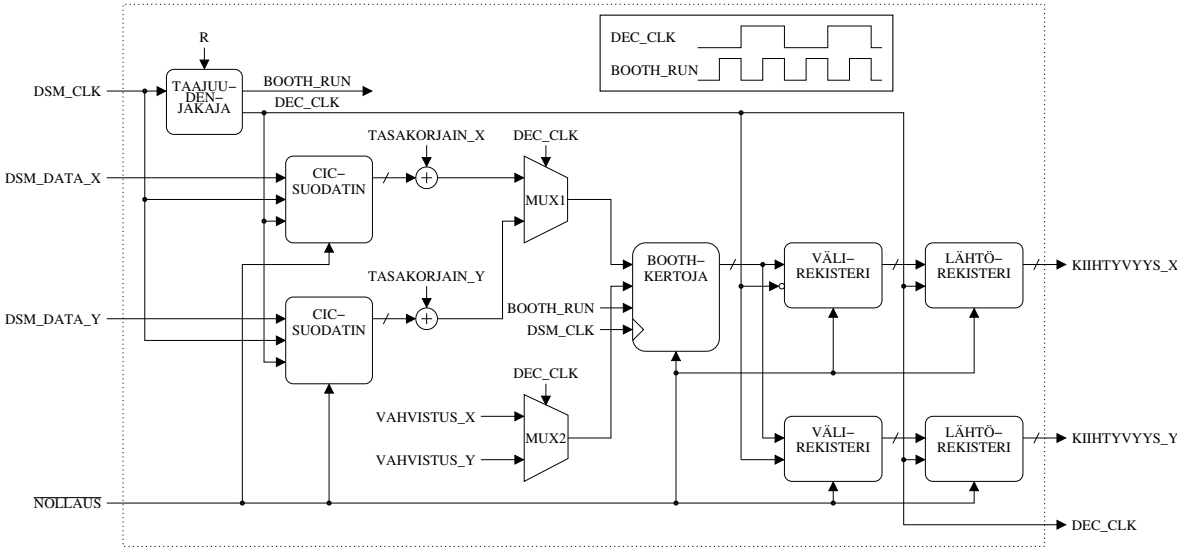
Kuva 2.1: Toteutetun digitaalipiirin lohkokaavio.

Toteutetun digitaalipiirin ylätason lohkokaavio on esitetty kuvassa 2.1. SPI-lohko muuttaa ulkoiselta ohjainlaitteelta MOSI-linjaa pitkin sarjamuodossa tulevan rekisteriosoitteen ja datan rinnakkaismuotoon ja ohjaa ne muistirekisterilohkolle. Jos kyseessä on kirjoituskäsky, muistirekisterilohkolle lähetetään kirjoituskäskylinjalla nouseva reuna, joka kellottaa kirjoitettavan datan muistirekisterilohkon D-kiikkuihin. Muistirekisterilohko limittää SPI-lohkolle menevälle luettavan datan väylälle sen rekisterin sisällön, jonka osoite SPI-väylältä tulee, joten muistirekisterilohkolle ei lähetetä erillistä lukukäskyä. SPI-lohko lähettää myös lukukäsky- ja SYNC_TRIG-signaalit synkronoijalohkolle. Niiden toiminta on esitetty kappaleessa 7.6.

AD-muuntimilta tulee yksibittistä x-akselin ja y-akselin kiihtyvyysdataa DSM_DATA_X- ja DSM_DATA_Y-linjoja pitkin kellon DSM_CLK kellottamana. Yksibittinen data ohja-

taan DSP-lohkolle, joka on esitetty tarkemmin kuvassa 2.2, jossa molempien akseleiden yksibittinen data suodatetaan CIC-suodattimilla ja lukujen sananleveys kasvaa. Suodatettuihin kiihtyvyyslukuihin lisätään tasapoikkeamankorjainarvot TASAKORJAIN_X ja TASAKORJAIN_Y. Tasakorjatut kiihtyvyysarvot ohjataan DEC_CLK-kellon eri vaiheissa Booth-kertojalle limittimellä MUX1 ja akselia vastaava vahvistusarvo VAHVISTUS_X tai VAHVISTUS_Y limittimellä MUX2. Kertojalle ohjataan x-akselin data ja vahvistusarvo, kun DEC_CLK-kellon arvo on 1, ja vastaavasti y-akselin data ja vahvistusarvo, kun DEC_CLK-kellon arvo on 0. BOOTH_RUN-signaalin nouseva reuna aloittaa kertolaskun suorittamisen, joka on kuvattu luvussa 6. DEC_CLK- ja BOOTH_RUN-signaalit on esitetty kuvan 2.2 yläreunassa. Tasapoikkeamankorjainarvot ja vahvistuksenkorjauskertoimet tulevat DSP-lohkolle muistirekisterilohkolta (signaali SÄÄTÖBITIT DSP-LOHKOLLE kuvassa 2.1).

DSP-lohkolta lähtevät kiihtyvyyslukemat menevät synkronoijalohkolle, jonka avulla pienennetään asynkronisesta kiihtyvyysdatan lukemisesta aiheutuvien virheellisten lukema-arvojen esiintymisen todennäköisyyttä. Kiihtyvyysdata ohjataan synkronoijalta muistirekisterilohkolle, joka limittää sen SPI-lohkolle aivan kuten säätöbittirekisterien arvotkin, kun MOSI-linjaa pitkin saatava rekisteriosoite niin määrää. Kiihtyvyysdataa ei talleteta muistirekisterilohkoon erikseen, vaan synkronoituja kiihtyvyyslukuja säilytetään vain synkronoijalohkon lähtörekistereissä.



Kuva 2.2: Toteutetun DSP-lohkon lohkokaavio.

Luku 3

Signaali taajuustasossa, suodatus ja alinäytteistäminen

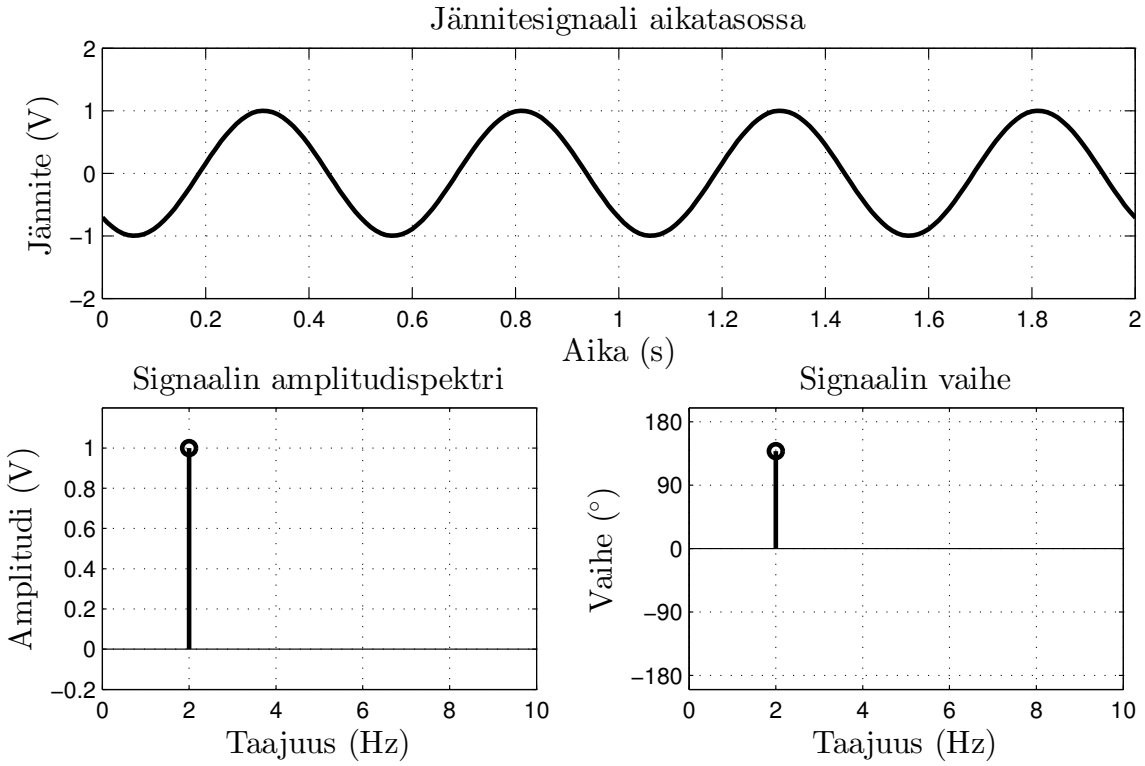
3.1 Signaali taajuustasossa

Mikä tahansa jaksollinen signaali voidaan esittää sini- ja kosinisignaalien ja vakiotermin summana, jotka muodostavat signaalin Fourier-sarjan.[8] Fourier-sarja voidaan laskea aikatason jaksollisesta signaalista Fourier-muunnoksella. Sarja voidaan esittää amplitudi-vaihemuodossa [9]

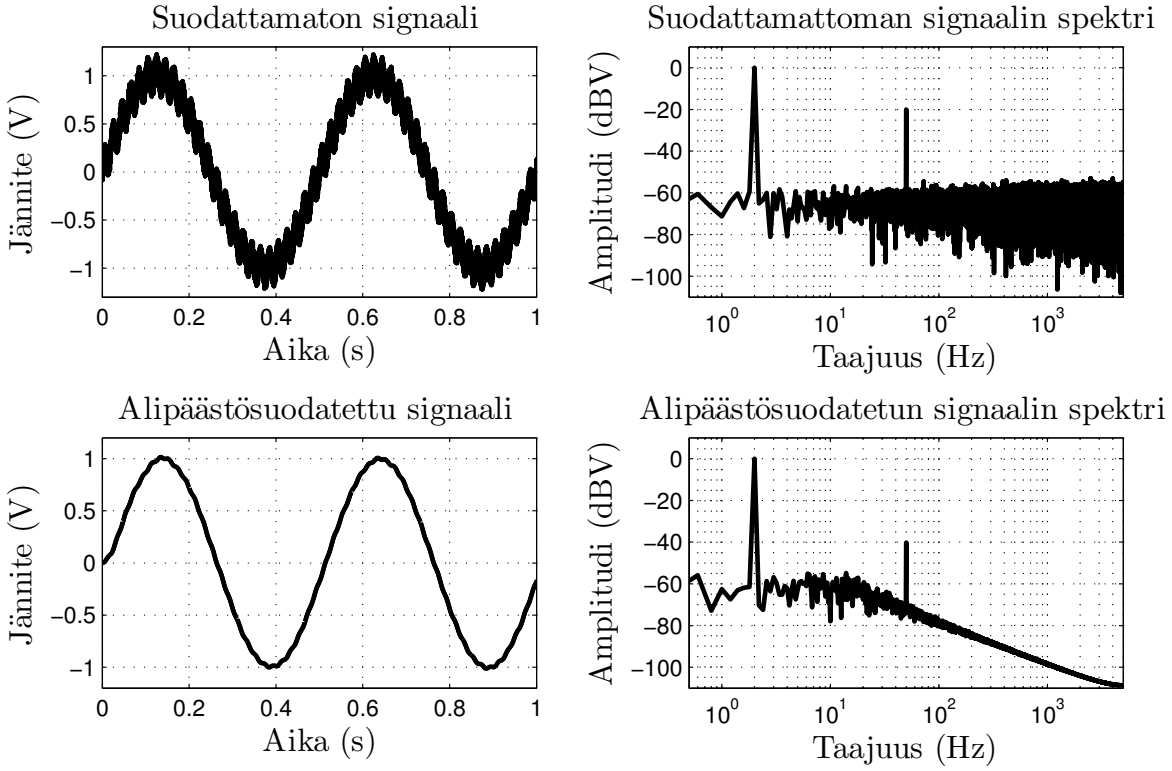
$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} A_n \cdot cos(n\omega t + \varphi_n), \tag{3.1}$$

jossa $a_0/2$ on signaalin vakiotermi, amplitudi A_n kunkin signaaliin kuuluvan kosinimuotoisen signaalikomponentin suuruus ja φ_n kunkin signaalikomponentin vaihe hetkellä t = 0. Fourier-sarja sisältää täydellisen informaation aikatason signaalista, ja suuruus- ja vaiheinformaatiosta voidaan muodostaa aikatason signaali ajan funktiona käänteisellä Fourier-muunnoksella. Amplitudit ja vaiheet voidaan esittää taajuuden funktiona, kuten kuvissa 3.1 ja 3.2, joista ensimmäisessä on yksittäinen kosiniaalto taajuusinformaatioineen ja jälkimmäisessä realistisempi kosinisignaali kohinan ja häiriösignaalien kanssa.

Signaalin esittäminen taajuustasossa helpottaa signaalin numeerista ja silmämääräistä analysointia. Esimerkiksi kuvan 3.2 spektristä nähdään välittömästi, että signaalissa olevat kaksi selvästi erottuvaa taajuutta ovat kaksi hertsiä ja 50 hertsiä. Signaalipiikkien taajuuksien perusteella voidaan esimerkiksi jäljittää mekaanisia häiriölähteitä kiihtyvyysanturimittauksissa ja toisaalta päätellä, mitkä ylimääräiset signaalit piirillä kytkeytyvät varsinaiseen signaaliin. Suuruustiedosta voidaan laskea esimerkiksi signaalipiikin teho ja toisaalta kohinan teho eri kaistoilla ja tästä tiedosta edelleen esimerkiksi signaali-kohinasuhde (engl. *signal-to-noise ratio*), joka on yksi suorituskykyä ilmaiseva parametri esimerkiksi anturipiireissä.



Kuva 3.1: Esimerkki signaalin esittämisestä aika- ja taajuustasoissa.



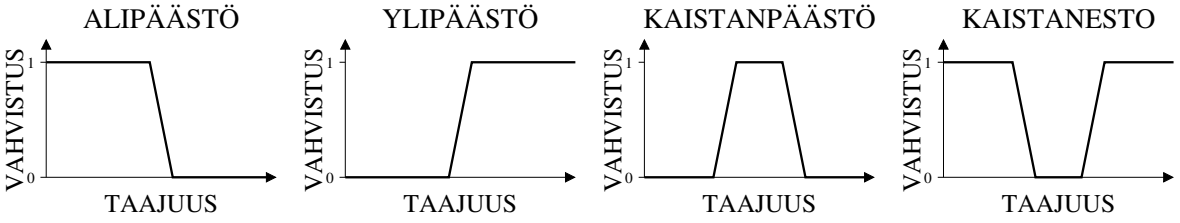
Kuva 3.2: Esimerkki signaalin alipäästösuodattamisen vaikutuksesta siihen aika- ja taajuustasoissa.

3.2 Signaalin suodattaminen

Signaaleja suodatetaan esimerkiksi elektronisten häiriösignaaleiden ja kohinan vaikutuksen vähentämiseksi, koska ne vääristävät signaalia. Kuvassa 3.2 vasemmalla ylhäällä on todellista kahden hertsin sinimuotoista jännitesignaalia muistuttava, Matlabilla luotu signaali, joka sisältää kohinaa ja johon on kytkeytynyt korkeampitaajuinen signaali. Kuvassa oikealla ylhäällä on signaalin amplitudispektri. Tässä tapauksessa häiriösignaali näkyy aikatasossa varsinaisen suurempiamplitudisen signaalin pieniamplitudisena heilahteluna. Kohinan taso on -60 dB, joten se näkyy aikatasossa satunnaisena, pieniamplitudisena vaihteluna. Kohinan ja häiriösignaalien takia signaalin arvo heittelee noin 0,1 volttia varsinaisen kahden hertsin sinisignaalin ympärillä. Spektrissä häiriösignaali näkyy selvästi erottuvana signaalipiikkinä taajuuden 50 Hz kohdalla. Kohina puolestaan muodostaa spektriin “lattian” tasolle -60 dBV. Jos signaali olisi kiihtyvyysanturin lähtöjännite, jonka jännitealue on -1 voltista +1 volttiin, häiriöt voisivat aiheuttaa noin 0,1 voltin eli noin viiden prosentin virheen signaalista satunnaisena ajanhetkenä näytteistettyyn arvoon. Alipäästösuodattamalla kuvan signaali se saadaan näyttämään siltä kuin kuvassa 3.2 alhaalla. Suodatettu signaali näyttää aikatasossa puhtaammalta sinisignaalilta. Taajuustasossa kohinan taso alkaa laskea noin 20 hertsin taajuudelta alkaen. 50 hertsin signaalipiikin amplitudi on pienentynyt 20 dBV, mikä vastaa amplitudin putoamista kymmenesosaan.

Kaikki analogiset, elektroniset signaalit sisältävät vähintään sähköjohtimissa aina esiintyvää termistä kohinaa. Häiriösignaaleja voi puolestaan tulla eri lähteistä niin elektronisen järjestelmän ulko- kuin sisäpuoleltakin. Esimerkiksi sähköverkkoon kytkettyyn järjestelmään voi kytkeytyä 50 hertsin häiriösignaali. Radiovastaanottimen antenni puolestaan vastaanottaa laajalta taajuuskaistalta radiosignaaleja, joista pitää suodattaa pois kaikki muut signaalit paitsi halutun kaistan sisältö. Mikropiirin sisällä tai piirilevyllä voi olla haitallista kytkeytymistä toisiaan lähellä sijaitsevien johtimien välillä, jolloin yhdessä johtimessa tapahtuva signaalin muutos vaikuttaa myös siihen kytkeytyneissä johtimissa kulkeviin signaaleihin.

3.2.1 Tavallisimmat suodatintyypit



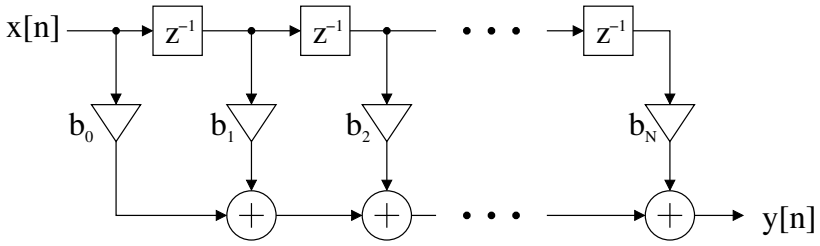
Kuva 3.3: Tavallisimpien suodatinten siirtofunktiot.

Tavallisimpien suodatinten siirtofunktiot on esitetty kuvassa 3.3. Alipäästösuodatin päästää läpi matalilla taajuuksilla olevat signaalikomponentit ja vaimentaa korkeilla taajuuksilla

olevat komponentit. Ylipäästösuodatin vastaavasti päästää läpi korkeilla taajuuksilla olevat signaalikomponentit ja vaimentaa matalilla taajuuksilla olevia. Kaistanpäästösuodatin päästää läpi ja kaistanestosuodatin vaimentaa jollain tietyllä kaistalla olevat signaalikomponentit. Lisäksi on olemassa kokopäästösuodatin. Sen vahvistus on kaikilla taajuuksilla yksi, mutta vaihevaste suunnitellaan esimerkiksi korjaamaan toisen suodattimen tai suodatinketjun vaihevaste, kun vahvistukseen ei haluta muutosta.

3.3 Äärellisen impulssivasteen suodattimet

Äärellinen impulssivaste (engl. *finite impulse response, FIR*) on signaalinkäsittelyjärjestelmän ominaisuus, joka tarkoittaa sitä, että kun järjestelmään syötetään impulssiheräte, saavuttaa vaste arvon 0 äärellisessä ajassa. Suodatinta, jolla on äärellinen impulssivaste, kutsutaan FIR-suodattimeksi. Digitaalisen FIR-suodattimen lohkokaavio on esitetty kuvassa 3.4. N-asteista FIR-suodatinta kuvaa N+1 kerrointa $b_0 ... b_N$, joten sen digitaaliseen implementointiin tarvitaan N+1 kertojaa, N kahden luvun summainta ja N viiverekisteriä.



Kuva 3.4: N-asteinen FIR-suodatin.

FIR-suodatin on aina BIBO-stabiili (engl. *bounded-input bounded-output*), eli sen lähtö on rajallinen kaikilla rajallisilla tulosignaaleilla. Lähtöarvo ei siis voi kasvaa äärettömäksi. Ominaisuus johtuu siitä, että FIR-tyyppisessä suodattimessa ei ole takaisinkytkentää, vaan vaste on aina summa äärellisestä määrästä herätenäytteitä kerrottuna suodattimen kertoimilla. Diskreetin FIR-suodattimen lähtöarvo riippuu siihen syötettävistä arvoista kaavan 3.2 mukaisesti. [10]

$$y[n] = b_0 x[n] + b_1 x[n-1] + b_2 x[n-2] + ... + b_N x[n-N] \tag{3.2}$$

FIR-suodatin voidaan suunnitella niin, että sen vaihevaste on lineaarinen signaalikaistalla. Jos suodattimen vaihevaste on lineaarinen signaalikaistalla, myös ryhmäviive (engl. *group delay*) on vakio signaalikaistalla ja hyötysignaalin muoto ei vääristy suodattimessa. Epälineaarinen vaihevaste vääristää lähtösignaalin muotoa. Sallitun epälineaarisuuden määrä riippuu sovelluksesta, jossa suodatinta käytetään. [10]

FIR-suodattimen lineaarinen vaihevaste saadaan aikaiseksi valitsemalla kertoimet symmetrisesti tai antisymmetrisesti. Symmetrisille kertoimille pätee kaava 3.3 ja antisymmetrisille kaava 3.4. Symmetriaa hyväksikäyttämällä saadaan pienennettyä kertoimien määrä pa-

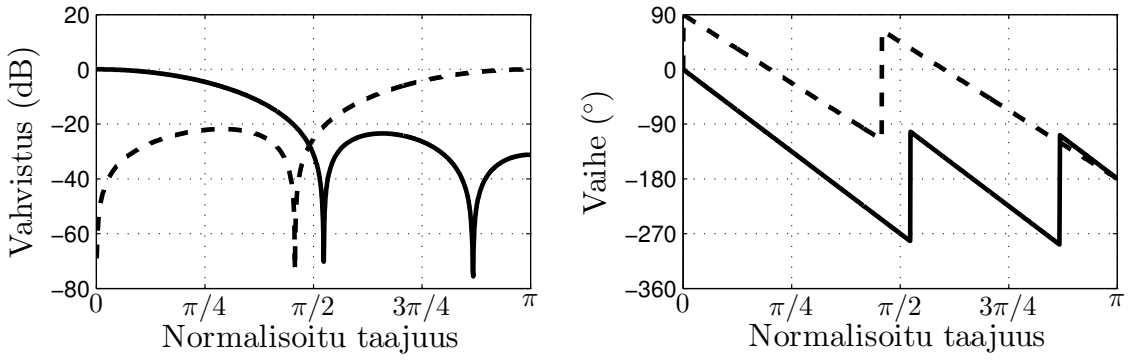
rittoman asteluvun tapauksessa puoleen ja parillisen tapauksessa lähes puoleen, koska kunkin symmetrisen tai antisymmetrisen kertojaparin voi korvata yhdellä kertojalla. [10] Se pienentää suodattimen tarvitsemaa pinta-alaa mikropiirillä huomattavasti, koska kertojat ovat isoja lohkoja esimerkiksi summaimiin verrattuna. Samalla myös tehonkulutus pienenee. Taulukossa 3.1 on esimerkit symmetrisistä ja antisymmetrisistä kertoimista ja kuvassa 3.5 niitä vastaavien suodattimien vahvistus- ja vaihevasteet. Esimerkin symmetriset kertoimet on valittu siten, että ne toteuttavat alipäästöfunktion, ja antisymmetriset siten, että ne toteuttavat ylipäästöfunktion.

$$b_n = b_{N-n} \tag{3.3}$$

$$b_n = -b_{N-n} \tag{3.4}$$

Taulukko 3.1: Esimerkit symmetrisistä ja antisymmetrisistä FIR-suodattimen kertoimista.

Kerroin	**Symmetriset kertoimet, N=6**	**Antisymmetriset kertoimet, N=5**
b_0	0,0096	-0,1
b_1	0,1314	0,5
b_2	0,2532	-0,7
b_3	0,2927	0,7
b_4	0,2532	-0,5
b_5	0,1314	0,1
b_6	0,0096	-

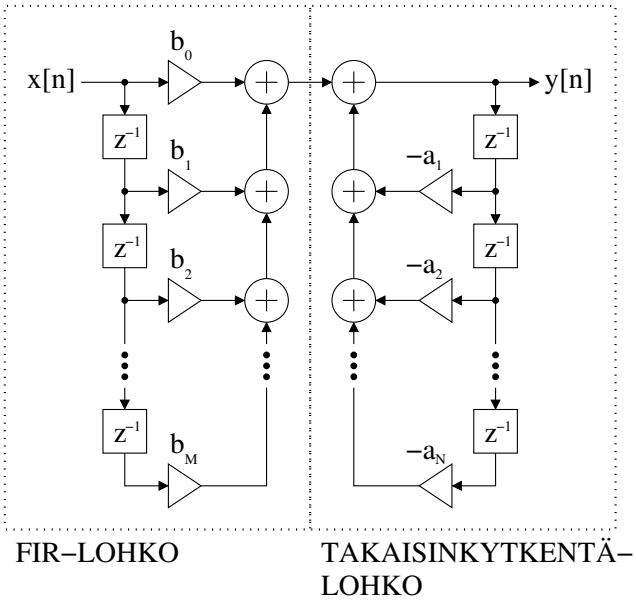


Kuva 3.5: Taulukon 3.1 kertoimia vastaavien suodatinten vahvistus- ja vaihevasteet. Taulukon symmetristen kertomien antama vaste on esitetty yhtenäisellä viivalla, antisymmetristen vaste katkonaisella.

3.4 Äärettömän impulssivasteen suodattimet

Ääretön impulssivaste (engl. *infinite impulse response, IIR*) on signaalinkäsittelyjärjestelmän ominaisuus, joka tarkoittaa sitä, että kun signaalinkäsittelyjärjestelmään syötetään impulssiheräte, vaste ei saavuta koskaan nollaa. Ominaisuus johtuu siitä, että IIR-järjestelmässä on takaisinkytkentää, eli lähtöarvo riippuu aina aiemmista lähtöarvoista. Oikein suunniteltuna suodattimen impulssivaste kuitenkin lähenee nollaa. Suodatinta, jolla on ääretön impulssivaste, sanotaan IIR-suodattimeksi. Suodatintoteutuksessa, jossa on äärelliset sananleveydet, impulssivaste voi kuitenkin pyöristyä nollaan, jos lähtöarvo pienenee pienemmäksi kuin vähiten merkitsevän bitin arvo suodattimen lähdössä.

Kuvassa 3.6 on esimerkki digitaalisen IIR-suodattimen lohkokaaviosta ja kaavassa 3.5 sen tuottaman vasteen yhtälö. Esimerkin IIR-suodattimen toteuttamiseen vaaditaan N+M+1 kertojaa, N+M summainta ja N+M viiverekisteriä, joissa M on vasemmanpuoleisen, FIR-tyyppisen lohkon asteluku ja N takaisinkytkentäpolun asteluku.



Kuva 3.6: IIR-suodatin.

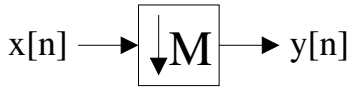
$$y[n] = b_0x[n]+b_1x[n-1]+...+b_Mx[n-M]-a_1y[n-1]-a_2y[n-2]-...-a_Ny[n-N] \quad (3.5)$$

IIR-suodattimen suunnittelu ei ole yhtä suoraviivaista kuin FIR-suodattimen. Takaisinkytkennästä aiheutuu siirtofunktioon napoja, joten se ei ole välttämättä stabiili. Jollain äärellisellä herätteellä huonosti suunnitellun IIR-suodattimen lähtö voi siis alkaa kasvaa rajattomasti, vaikka tuloon alettaisiinkin sen jälkeen syöttää nollaa. Näin tapahtuisi esimerkiksi jos olisi vain yksi takaisinkytkentäkerroin a_1 = -2, eli lähtöarvoon lisättäisiin aina edellinen lähtöarvo kaksinkertaisena.

Jotta suodatin on stabiili, on kertoimet valittava siten, että navat ovat yksikköympyrän sisällä.[10] Lisäksi pitää varmistaa, että navat ovat yksikköympyrän sisäpuolella myös sen jälkeen, kun kertoimet on kvantisoitu digitaalisissa kertojissa kertoimen äärelliselle sananleveydelle.

IIR-suodattimen etuna on se, että pienemmällä kerroinmäärällä voidaan saavuttaa kapeampi siirtokaista tai toisin sanoen suurempi vaimennus kuin FIR-suodattimella. Monien käytännöllisten suodatinvaatimusten toteuttamisessa FIR-suodatin vaatii kymmeniä kertoja enemmän kertoimia kuin IIR-suodatin.[10] Näin ollen IIR-toteutus voi viedä huomattavasti vähemmän pinta-alaa mikropiirillä ja kuluttaa vähemmän laskentatehoa. IIR-suodattimissa sananleveyksien pyöristämisestä aiheutuvat seuraukset pitää analysoida huolellisemmin, koska takaisinkytkentäpoluissa samaa näytettä "kierrätetään" loputtomiin, ja pyöristysvirheet kertautuvat kierros kierrokselta, mistä voi seurata esimerkiksi pieniamplitudista oskillointia. IIR-suodatinten haittapuolena on se, että niillä ei voida saavuttaa täysin lineaarista vaihevastetta. Suodattimen vaihevastetta voidaan kuitenkin korjata esimerkiksi kokopäästösuodattimella. [8]

3.5 Alinäytteistäminen

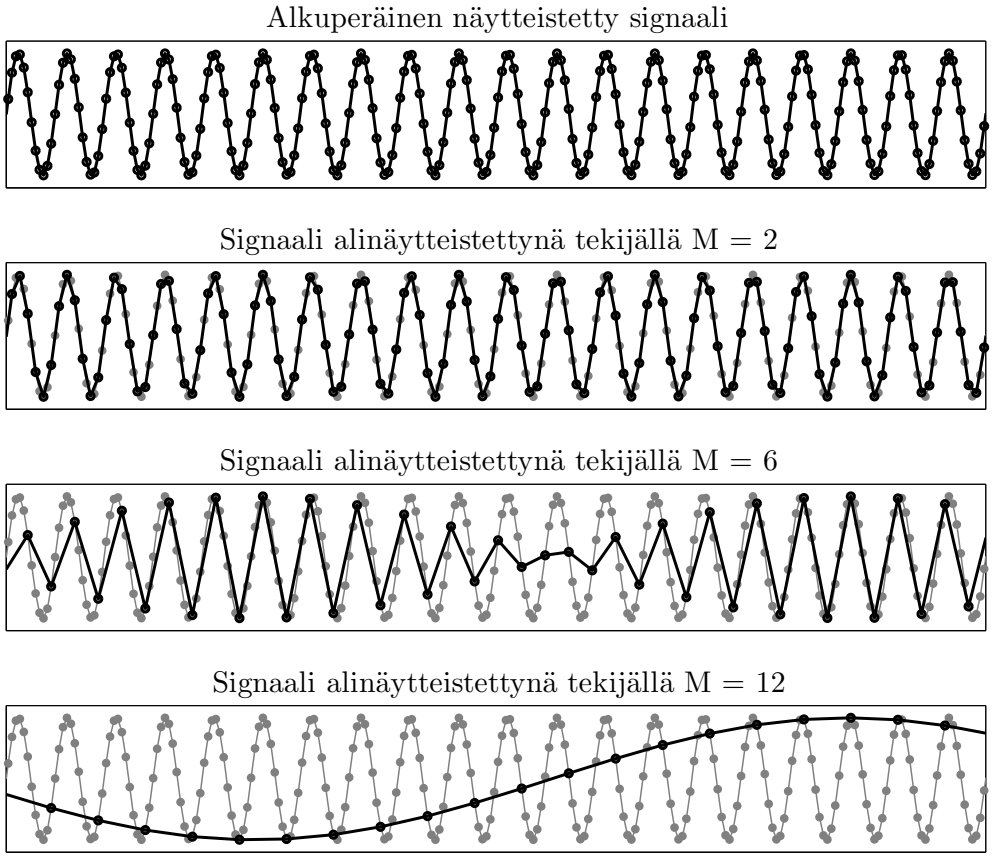


Kuva 3.7: Alinäytteistäjän piirrossymboli.

Alinäytteistäminen tarkoittaa näytteistystaajuuden pudottamista signaalinkäsittelyjärjestelmässä. Alinäytteistäjän piirrossymboli on kuvassa 3.7. Tekijällä M alinäytteistäminen tarkoittaa aikatasossa sitä, että alinäytteistäjän syötteestä x[n] poimitaan joka M. näyte, eli $y[n] = x[nM]$. Muut näytteet jätetään huomioimatta.

Kuvassa 3.8 on esimerkki signaalin alinäytteistämisestä eri tekijöillä M. Siitä nähdään, että tekijä M voi vaikuttaa huomattavasti siihen, miltä alinäytteistetty signaali näyttää. Kuvan tapauksessa tekijällä M = 2 alinäytteistetty signaali näyttää lähes samalta kuin alkuperäinen signaali. Sen sijaan tekijällä M = 6 alinäytteistetty signaali näyttää kahden kosinimuotoisen signaalin summalta ja tekijällä M = 12 alinäytteistetty puolestaan alkuperäistä matalataajuisemmalta kosinisignaalilta.

Alinäytteistystä voidaan hyödyntää anturipiireissä esimerkiksi kun käytetään ylinäytteistävää AD-muunninta. Silloin näytteistystaajuus on huomattavasti suurempi kuin signaalikaistan suurin taajuus, kuten esimerkiksi kuvassa 3.13. Esimerkiksi tämän työn piirin AD-muuntimien tuottamien yksibittisten signaalien näytteistystaajuus on 100 kHz. Niitä voidaan kuitenkin alinäytteistää, jotta suodattimien perässä olevat summaimet ja kertoja toimisivat pienemmällä taajuudella, jolloin niiden tehonkulutus pienenee.



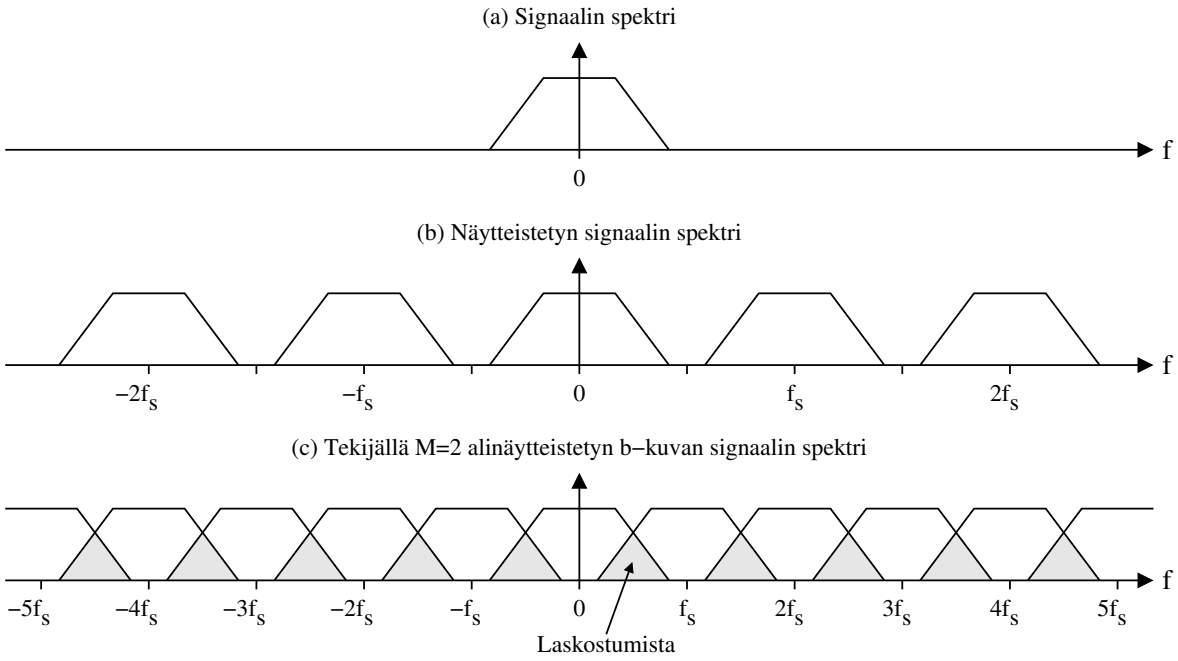
Kuva 3.8: Alinäytteistäminen aikatasossa.

3.6 Laskostuminen alinäytteistyksessä ja desimointisuodatus

Taajuuden muutokset alinäytteistettäessä johtuvat laskostumisesta (engl. *aliasing*). Laskostuminen taajuusalueessa on havainnollistettu kuvassa 3.9. Kuvassa 3.9(a) on signaalin spektri. Kuvassa 3.9(b) on saman signaalin spektri taajuudella f_s näytteistämisen jälkeen. Näytteistämisen jälkeen alkuperäisen signaalin spektrin kopio toistuu näytteistystaajuuden moninkertojen $N \cdot f_s$ kohdalla. [8][10] Kuvassa 3.9(c) on b-kuvan signaalin spektri tekijällä $M = 2$ alinäytteistämisen jälkeen. Alinäytteistämisen jälkeen spektrin kopioita esiintyy uuden näytteistystaajuuden moninkertojen kohdalla. Kuvaan on merkitty harmaalla alue, jossa spektrin kopio on tuonut alkuperäisen signaalin spektrin päälle signaalikomponentteja, eli on tapahtunut laskostumista. Laskostuminen voi tuoda alkuperäisen signaalin kaistalle esimerkiksi kohinaa tai ylimääräisiä signaalipiikkejä, mikä vääristää signaalia.

Jotta laskostuminen aiheuttaisi vähemmän vääristymistä signaaliin, pitää signaali alipäästösuodattaa ennen alinäytteistämistä. Tekijällä M alinäytteistettäessä pitää suodattaa taajuuden $f_s/2M$ yläpuolella olevia taajuuksia. Suodatinta, joka rajoittaa kaistaa edellä mainitun mukaisesti, sanotaan desimointisuodattimeksi (engl. *decimation filter*). [10] Tämän työn piirillä ylinäytteistävien delta-sigma-AD-muuntimien lähtösignaaleissa korkeilla taajuuksil-

la on lähinnä kohinaa, joka halutaan suodattaa pois, jotta se ei laskostu signaalikaistan päälle alinäytteistyksessä.



Kuva 3.9: Esimerkki signaalin, näytteistetyn signaalin ja tekijällä $M = 2$ alinäytteistetyn signaalin spektreistä.

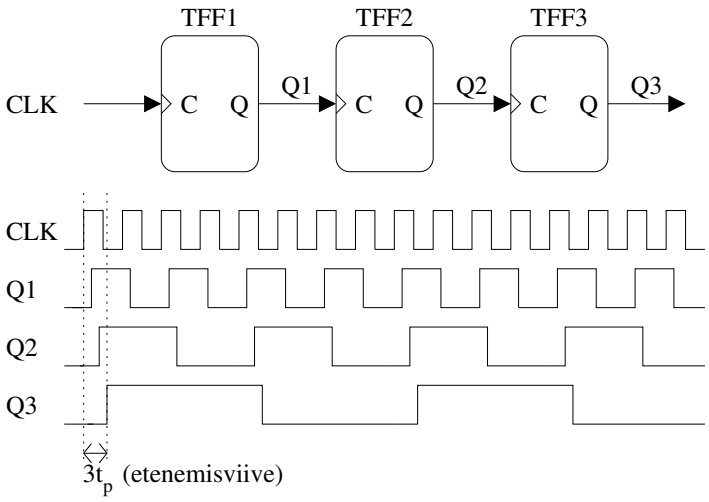
3.7 Taajuudenjakaja

Alinäytteistäjässä tarvitaan kellosignaali, jonka taajuus on pienempi kuin alinäytteistäjään tulevan datan näytteistystaajuus. Pienempi taajuus kannattaa luoda näytteistystaajuudesta taajuudenjakajalla (engl. *frequency divider*) [11], jotta taajuudenjakosuhde olisi tarkasti määritelty. Taajuudenjakajaan syötetään alkuperäinen kellosignaali ja se luo kellosignaalin, jolla on pienempi taajuus.

3.7.1 Taajuuden jakaminen kiikkuketjulla

Kellotaajuus voidaan puolittaa yksinkertaisimmillaan yhdellä kiikulla.[11] Esimerkiksi T-kiikun (engl. *toggle flip-flop*) lähtö vaihtaa tilaa kellotulonsa nousevalla reunalla. Asettamalla T-kiikkuja peräkkäin ja kellottamalla ensimmäistä alkuperäisellä kellolla ja seuraavia kiikkuja aina edellisen kiikun lähdöllä, saadaan kellotaajuus jaettua kahden potenssiluvuilla. T-kiikun asemesta voidaan myös käyttää D-kiikkua ja kytkeä sen lähtö tuloonsa invertterin kautta, jolloin se käyttäytyy T-kiikun tavoin.

Kuvassa 3.10 on esimerkki kolmen T-kiikun taajuudenjakajasta ja kiikkujen lähtösignaaleista. T-kiikun TFF1 lähdön Q1 taajuus on puolet kellon CLK taajuudesta. T-kiikun TFF2 lähdön Q2 taajuus on puolet signaalin Q1 taajuudesta, eli neljäsosa kellon CLK taajuudesta.



Kuva 3.10: T-kiikuilla toteutettu taajuudenjakaja.

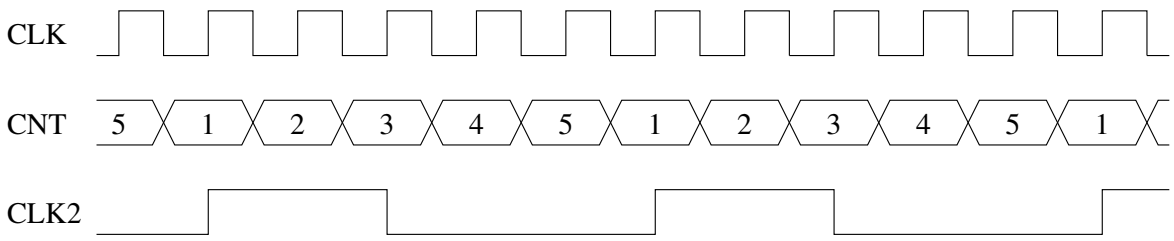
Viimeisen kiikun TFF3 lähdön Q3 taajuus on edelleen puolet signaalin Q2 taajuudesta, eli kahdeksasosa kellon CLK taajuudesta.

Jos jaettu kellotaajuus luodaan kiikkuketjulla, tulee suunnittelussa ottaa huomioon se, että jokainen ketjun kiikku aiheuttaa etenemisviiveensä (engl. *propagation delay*) suuruisen viiveen tulo- ja lähtökellojensa väliin. Jos kello jaetaan N kiikulla, on alkuperäisen ja jaetun kellon viive-ero $N \cdot t_p$, jossa t_p on yhden kiikun etenemisviive. Etenemisviiveen vaikutus on havainnollistettu kuvassa 3.10 vasemmalla alhaalla. Kiikkujen lähtöjen nousevat reunat tapahtuvat viiveellä suhteessa kellon CLK nouseviin reunoihin ja signaalin Q3 nouseva reuna viivästyy kolmen etenemisviiveen t_p verran.

3.7.2 Taajuuden jakaminen laskurilla

Toinen tapa luoda jaettu kellotaajuus on käyttää laskuria (engl. *counter*), jota kellotetaan alkuperäisellä kellolla ja joka laskee johonkin lukuun R asti. Luku R määrää suoraan taajuudenjakosuhteen. Laskuria käyttämällä kellotaajuus saadaan jaettua siis muillakin kokonaisluvuilla kuin kahden potensseilla.

Yksi tapa toteuttaa taajuudenjako laskurilla on havainnollistettu kuvassa 3.11, jossa taajuus jaetaan luvulla R = 5. Laskuri laskee toistuvassa silmukassa lukuja arvosta 1 arvoon 5 ja laskurin arvo CNT päivitetään kellotulon CLK laskevilla reunoilla. Kun laskurin arvo on CNT = 1, lähtökellon arvo CLK2 asetetaan ylös. Kun laskuri ylittää luvun $R/2$, eli CNT = $\lceil R/2 \rceil$ = 3, lähtökellon arvo CLK2 asetetaan alas. Kun laskuri saavuttaa arvon CNT = R = 5, laskurin arvo palautetaan arvoon 1 ja laskeminen alkaa alusta. Laskuritoteutuksessakin taajuudenjakajan lähtökello saattaa viivästyä suhteessa tulokelloon.



Kuva 3.11: Esimerkki laskurilla toteutetun taajuudenjakajan toiminnasta.

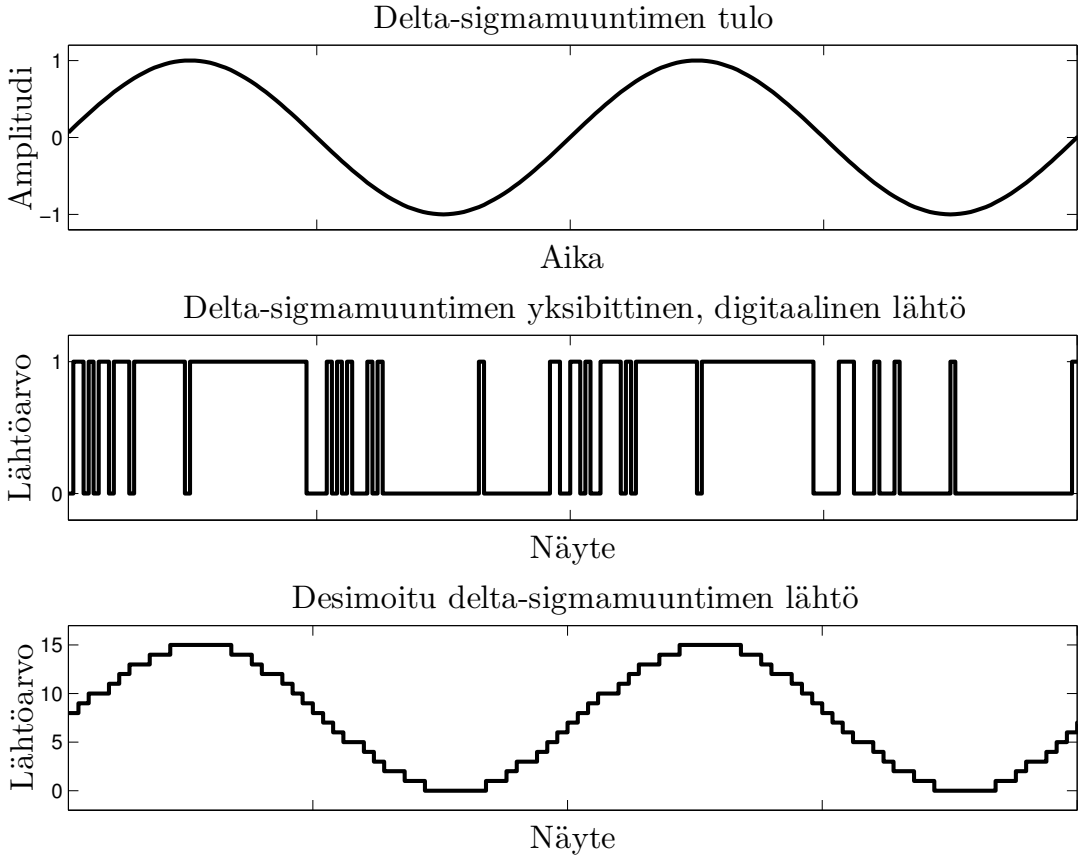
3.8 Yksibittinen AD-muunnos

Tämän työn mikropiirillä on molempia kiihtyvyysakseleita varten omat AD-muuntimensa, jotka on toteutettu yksibittisinä delta-sigmamuuntimina[12]. Yksibittinen delta-sigmamuunnin muuntaa jännitteen, joka tämän työn piirillä ilmaisee MEMS-anturin havaitseman kiihtyvyyden, suurella näytteistystaajuudella yksibittiseksi, digitaaliseksi dataksi. Sitä keskiarvoistamalla tai suodattamalla kiihtyvyys saadaan esitettyä monibittisenä lukuna. Jos esimerkiksi tarkkaillaan kuudentoista yksibittisen näytteen jaksoja ja delta-sigmamuuntimen lähdöstä saadaan lukujono 1010 1110 1110 1111, jossa on 12 kappaletta ykkösiä, saadaan sen keskiarvoksi 12 / 16 = 0,75. Näiden bittien näytteistyksen aikana kiihtyvyys olisi ollut siis 75% mitta-asteikon maksimiarvosta.

Kuvassa 3.12 on esitetty delta-sigma-AD-muuntimen sinimuotoinen tulosignaali ja sitä vastaavat yksibittinen lähtösignaali ja desimoitu lähtösignaali. Kuvasta voidaan nähdä, että kun siniaalto on huipussaan, muuntimen lähdössä on enemmän ykkösiä, ja vastaavasti siniaallon pohjalla lähdössä on enemmän nollia.

3.9 Suodatettava signaali

Tämän työn digitaalilohkon täytyy suodattaa yksibittiseltä delta-sigma-AD-muuntimelta tulevaa signaalia. Delta-sigmamuunnin on kohinaa muokkaava (engl. *noise shaping*) AD-muunnin. Kun käytetään kohinaa muokkaavaa AD-muunninta sopivan suodattimen kanssa, saavutetaan huomattava parannus signaali-kohinasuhteessa ja siten parempi havainnointitarkkuus.[13] Esimerkiksi yksibittinen alipäästötyyppinen delta-sigmamuunnin toimii siten, että kvantisointikohinaa on alhaisilla taajuuksilla vain vähän ja korkeilla taajuuksilla enemmän. Kuvassa 3.13 on esimerkki sellaisen signaalin spektristä.[14] Hyötysignaalipiikki on 90 hertsin kohdalla ja kohina alkaa kasvaa n. 500 hertsin kohdalta +40 dB/dec. Alipäästösuodattimella korkeilla taajuuksilla oleva kohina saadaan vaimennettua siten, että matalalla taajuudella olevat signaalit kuitenkin pääsevät läpi muuttumattomina.

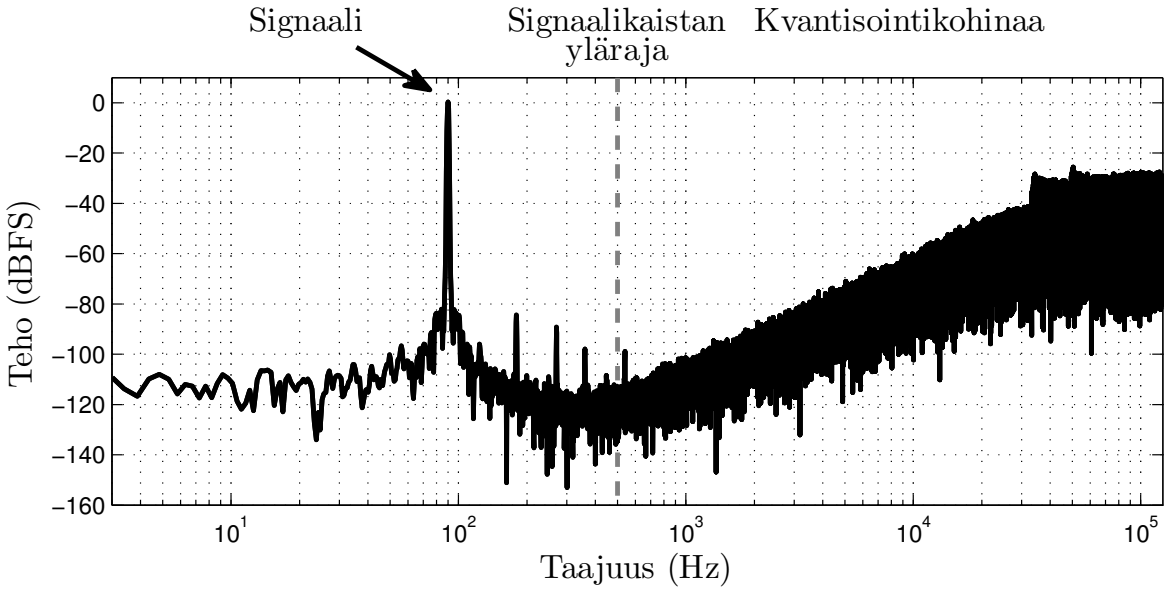


Kuva 3.12: Esimerkki yksibittisen delta-sigma-AD-muuntimen tulo- ja lähtösignaalista sekä desimointisuodatetusta lähtösignaalista.

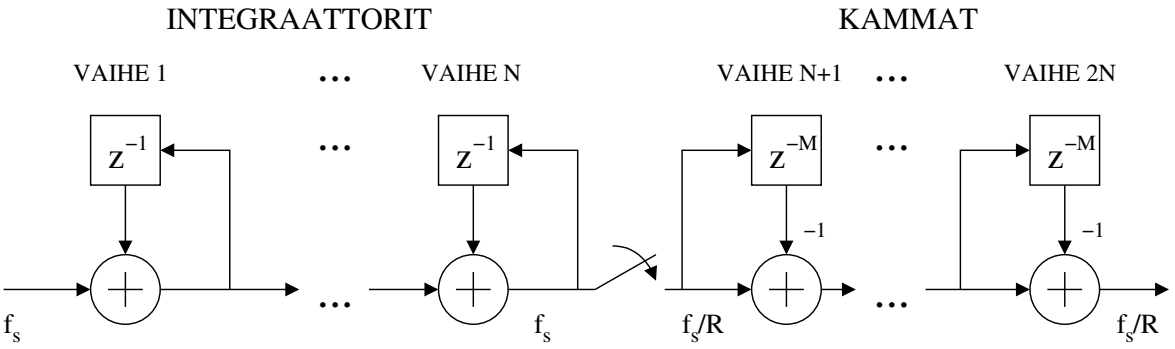
3.10 CIC-suodatin

CIC- eli Cascaded Integrator-Comb-suodattimen keksi Eugene B. Hogenauer ja hän esitteli sen vuonna 1981 julkaisussaan *An Economical Class of Digital Filters for Decimation and Interpolation* [15]. Se on FIR-tyyppinen suodatin, jonka kaikkien kertoimien arvo on yksi, minkä takia suodatin ei tarvitse lainkaan kertojalohkoja. Siksi CIC-suodatin vie paljon vähemmän pinta-alaa mikropiirillä kuin perinteiset, paljon kertoimia sisältävät FIR-suodattimet. Yksinkertaisen rakenteen takia sen taajuusvasteen muoto on kuitenkin rajallinen. CIC-suodatin voi olla desimoiva tai interpoloiva.

CIC-suodatin koostuu sarjaan kytketyistä integraattoreista ja kammoista. Ne molemmat ovat rakenteeltaan yksinkertaisia lohkoja, jotka sisältävät vain summaimen ja viiverekisterin. Kuvassa 3.14 on esitetty CIC-desimointisuodattimen rakenne. N-asteisessa, alinäytteistävässä CIC-desimointisuodatinkytkennässä on N kappaletta integraattoreita, alinäytteistäjä ja N kappaletta kampoja. Alinäytteistäjän sijoittaminen integraattori- ja kampalohkojen väliin pienentää kampojen tehonkulutusta, koska silloin ne toimivat pienemmällä taajuudella. Alinäytteistäjä voidaan myös jättää pois, jos halutaan, että lähtödatan näytteistystaajuus on sama kuin tulevan datan. Interpoloivan CIC-suodattimen lohkokaavio on samanlainen kuin



Kuva 3.13: Mitattu kaksiasteiselta delta-sigma-AD-muuntimelta tulevan 90 hertsin sinisignaalin tehospektri.[14]



Kuva 3.14: CIC-desimointisuodattimen rakenne.

kuvassa 3.14, mutta integraattoreiden ja kampojen paikat vaihdetaan keskenään ja niiden välissä on ylinäytteistäjä, eli tulodata menee kampavaiheelle ja integraattorit ovat lähdön puolella.

Integraattorin (kaava 3.6) ja kamman (kaava 3.7) siirtofunktioista saadaan johdettua CIC-desimaattorin z-tason siirtofunktio (kaava 3.8).

$$H_I(z) = \frac{1}{1 - z^{-1}} \tag{3.6}$$

$$H_C(z) = 1 - z^{-RM} \tag{3.7}$$

$$H(z) = H_I^N(z) \cdot H_C^N(z) = \frac{(1 - z^{-RM})^N}{(1 - z^{-1})^N} \tag{3.8}$$

CIC-desimaattorin z-tason siirtofunktiosta voidaan edelleen johtaa signaalitehon kaava taajuustasossa (kaava 3.9). Jos taajuudenpudotussuhde R on suuri, kaava saadaan muotoon 3.10, joka on sinc-funktion muotoinen.[15]

$$P(f) = \left[\frac{sin(\pi M f)}{sin\ \frac{\pi f}{R}}\right]^{2N} \tag{3.9}$$

$$P(f) = \left[RM\frac{sin(\pi M f)}{\pi M f}\right]^{2N} \tag{3.10}$$

Rakenteensa takia CIC-tyyppisen desimaattorin taajuusvasteen muoto on rajallinen. Kuten kaavasta 3.10 nähdään, on vain kolme muuttujaa, jotka voidaan valita: suodatinasteiden määrä N, taajuudenpudottamissuhde R ja kampojen viive M. Esimerkkikuvia muuttujien vaikutuksesta on esitetty liitteessä A. Suodatinasteiden määrä N vaikuttaa vaimennuksen määrään siten, että vaimennus kasvaa taajuuden kasvaessa $N \cdot 20dB/dec$, kuten on havainnollistettu kuvassa A.1. Taajuudenpudottamissuhde ja kampojen viive vaikuttavat nollakohtien määrään, mutta samalla myös vaimennukseen - mitä suurempi R tai M, sitä enemmän vaimennusta on suurilla taajuuksilla. Kuvassa A.2 on esitetty taajuudenpudottamissuhteen R vaikutus vasteeseen ja kuvassa A.3 kampojen viiveen M vaikutus.

Kuvassa 3.15 on esitetty sellaisen suodattimen taajuusvaste, jolla N = 3, R = 8 ja M = 1. Vasteen nollat tulevat tasaisin välein normalisoiduille taajuuksille

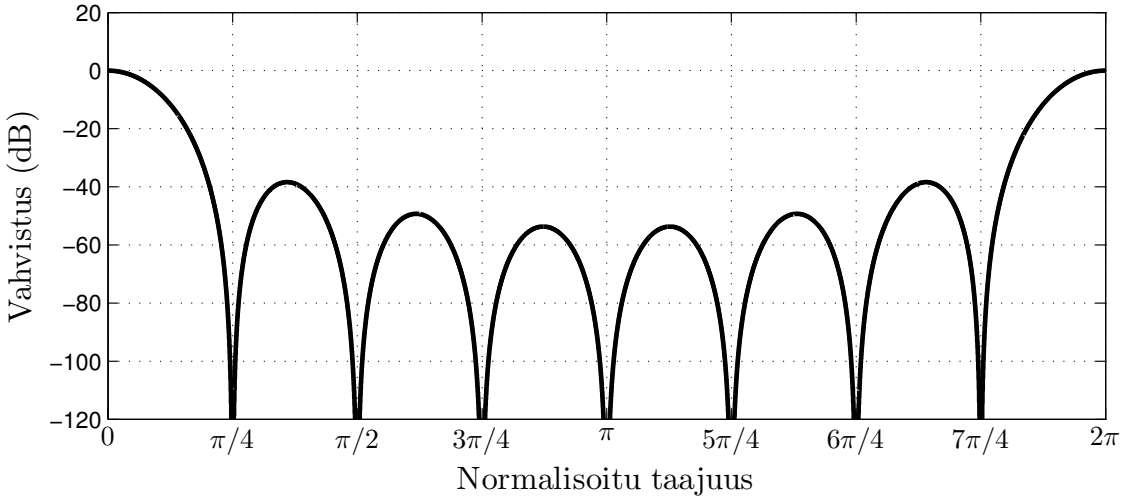
$$f_{z,i} = \frac{2\pi i}{RM}, \qquad i = 1,2,...,RM-1. \tag{3.11}$$

CIC-suodattimessa on tärkeää valita lohkojen sananleveys oikein, jotta tietoa ei menetetä rekisterien ylivuodon takia. Eniten merkitsevän bitin järjestysluku saadaan laskettua kaavalla 3.12 [15], ja bittien numerointi alkaa nollasta, joten sananleveydelle saadaan muodostettua kaava 3.13. Tyypillisesti kaavalla 3.13 laskettu sananleveys *WL* on kaikkien integraattoreiden ja kampojen sananleveys, mutta on myös mahdollista optimoida integraattoreiden ja kampojen sananleveydet pienemmiksi esimerkiksi jos suodattimen perässä oleville lohkoille riittää pienempi sananleveys [16]. Silloin suodatinlohkon pinta-ala pienenee, mutta bittimäärän pudottamisen vaikutus kohinaan tulee analysoida huolella.

$$B_{max} = Nlog_2(RM) + B_{in} - 1 \tag{3.12}$$

$$WL = B_{max} + 1 = Nlog_2(RM) + B_{in} \tag{3.13}$$

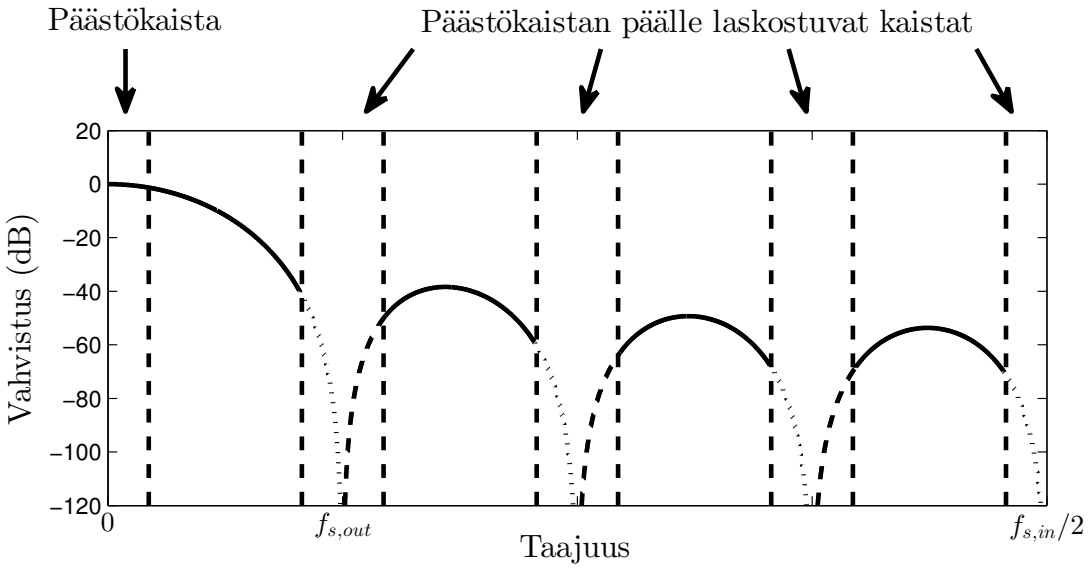
Vasteensa puolesta CIC-desimaattori sopii hyvin suodatukseen, kun vaaditaan suuri taajuudenpudottamissuhde ja pinta-alan ja tehonkulutuksen pitää olla pieniä, sillä CIC-desimaattorissa rajataajuus saadaan alhaiseksi pienellä määrällä digitaalisoluja. Kertojiin perustuva FIR-desimointisuodatin veisi paljon pinta-alaa, koska suuren taajuudenpudotuksen takia tarvittava alhainen rajataajuus vaatisi suuren määrän kertojia ja siten enemmän mikro-



Kuva 3.15: CIC-desimointisuodattimen taajuusvaste, kun asteluku on N = 3, taajuudenpudotussuhde on R = 8 ja kampojen viive on M = 1.

piirin pinta-alaa. Jos vaatimuksena on tasaisempi vahvistus päästökaistalla, voidaan CIC-suodattimen perään laittaa suodatin, joka tasoittaa vastetta päästökaistalla [17].

Kuvassa 3.16 on havainnollistettu sellaisen CIC-desimointisuodattimen päästökaistan päälle laskostuvat kaistat, jonka taajuudenpudotussuhde on R = 8. Päästökaistalle laskostuvat siirtofunktion nollakohtien vierestä eniten vaimennetut taajuudet. Katkoviivalla piirretyt vasteen osat laskostuvat päästökaistalle suoraan, kun taas pistekatkoviivalla piirretyt vasteen osat laskostuvat samalle alueelle pystyakselin suhteen peilautuen. Näin ollen kohinaa laskostuu eniten signaalikaistan 0 ... $f_{s,out}$ yläpäähän.



Kuva 3.16: Päästökaistalle laskostuvat kaistat CIC-desimointisuodattimessa (N = 3, R = 8 ja M = 1).

3.11 Toteutettu CIC-suodatin

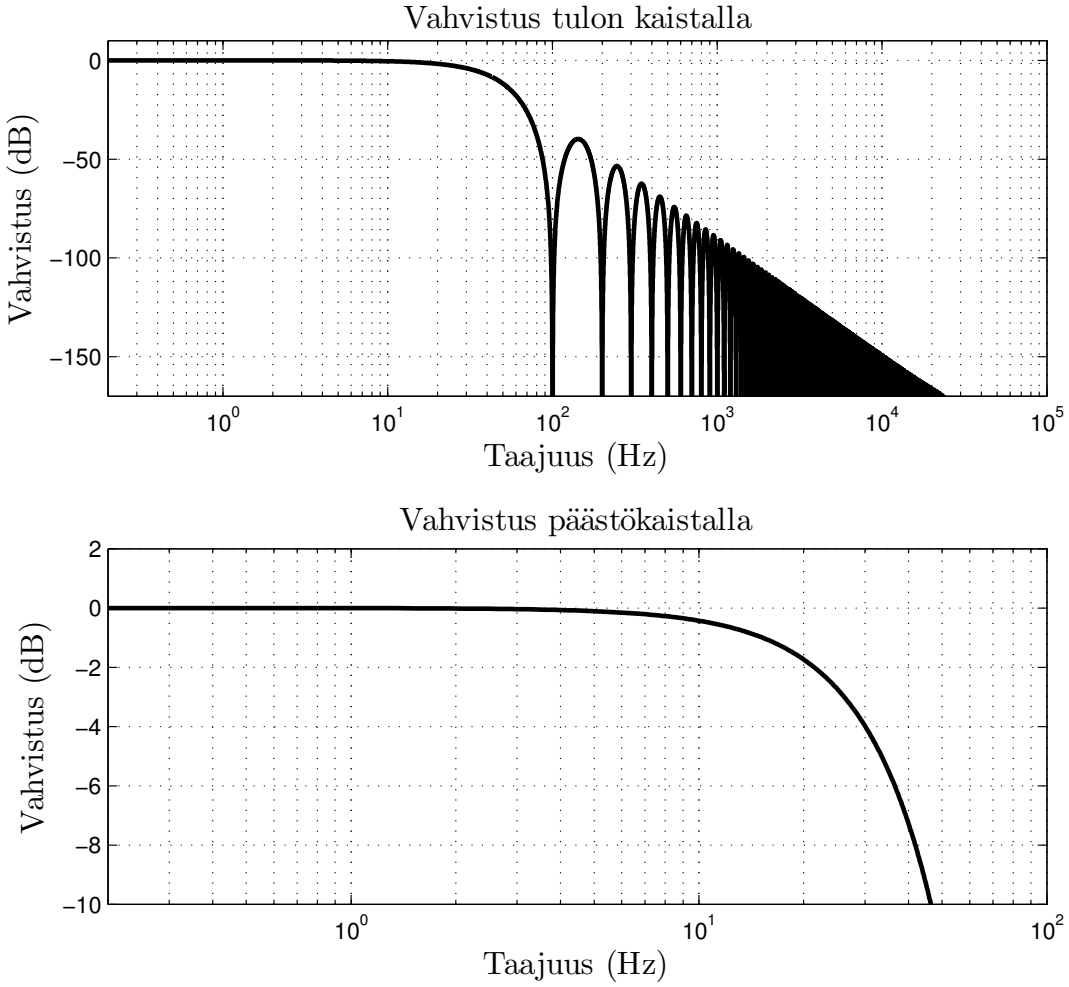
Molemmat työn kiihtyvyysanturipiirillä olevista AD-muuntimista muuntavat kiihtyvyyden ilmaisevan jännitteen yksibittiseksi dataksi näytteistystaajuudella 100 kHz, joten CIC-suodattimien tulojen näytteistystaajuus on $f_{s,in}$ = 100 kHz. Koko anturipiirin digitaalisten kiihtyvyysdatalähtöjen näytteistystaajuudeksi puolestaan on määritelty $f_{s,out}$ = 100 Hz. Näistä saadaan laskettua taajuudenpudotussuhteeksi $R = f_{s,in}/f_{s,out}$ = 1000. Piirillä olevan kellogeneraattorin tuottama taajuus saattaa poiketa muutamia prosentteja arvosta $f_{s,in}$, joten digitaalilohkon taajuudenjakaja toteutettiin laskurilla, jotta taajuudenpudotussuhteeksi R voidaan valita kokonaislukuja luvun 1000 ympäriltä. Toteutetussa piirissä R on valittavissa väliltä 64-1024. Huomattavasti tuhatta pienemmät taajuudenpudotussuhteet mahdollistettiin, jotta voidaan testata myös vaatimuksia pienempiä desimointisuhteita leveämmän kaistanleveyden saavuttamiseksi.

Delta-sigma-AD-muuntimen lähtödatan suodattamiseen riittää yleensä CIC-desimointisuodatin, jonka asteluku on

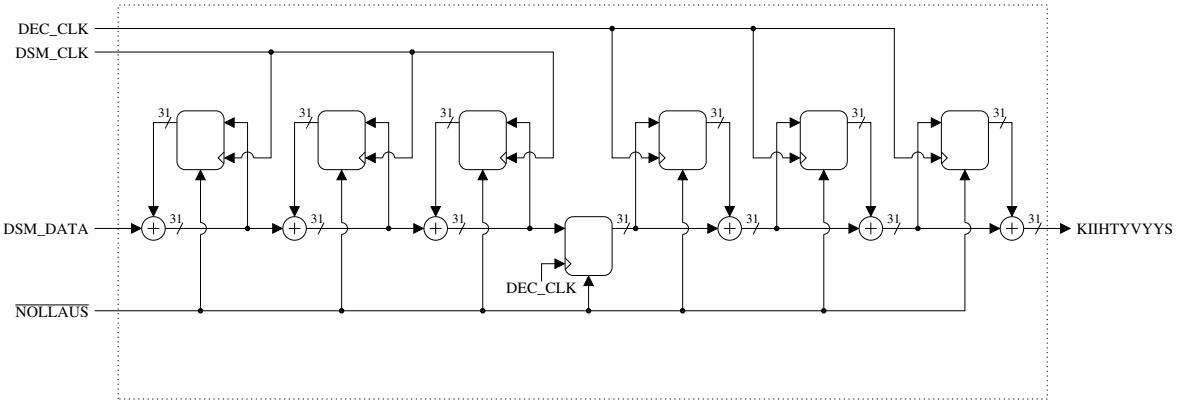
$$N = L + 1, \tag{3.14}$$

missä L on AD-muuntimen asteluku.[18] Kaava perustuu siihen, että L-asteisen delta-sigma-AD-muuntimen lähtösignaalin kohina kasvaa $L\cdot$ 20 dB/dec suurilla taajuuksilla, kun taas N-asteisen CIC-desimointisuodattimen vaimennus kasvaa $N\cdot$ 20 dB/dec. Kaavan 3.14 mukaisen suodattimen suodattaman signaalin kohina korkeilla taajuuksilla laskee 20 dB/dec, mikä ei aiheuta merkittävää signaali-kohinasuhteen laskemista alinäytteistyksestä aiheutuvan laskostumisen myötä. Työn piirin tapauksessa delta-sigmamuuntimille L = 2, josta saadaan laskettua CIC-suodattimien asteluvuksi $N = L + 1 = 3$. Näin ollen suodattimien vahvistus laskee estokaistalla 60 dB/dec ja suodattamisen jälkeen kohinan taso laskee estokaistalla siis 20 dB/dec. Kampojen viiveeksi valittiin M = 1. Suodattimien taajuusvaste on esitetty kuvassa 3.17.

Kaavalla 3.13 suodattimien sisäiseksi sananleveydeksi tulee 31. Käytössä olevalla 0,35 µm CMOS-teknologialla yhden toteutetun CIC-suodattimen standardisolut vievät pinta-alaa 0,126 mm^2, ja kahden suodattimen 0,252 mm^2, mikä on 42% kaikkien työssä toteutettujen digitaalilohkojen muodostaman ytimen (engl. *core*) kokonaispinta-alasta. Toteutetun suodattimen lohkokaavio on esitetty kuvassa 3.18.



Kuva 3.17: Piirille toteutetun CIC-desimointisuodattimen ($N = 3$, $R = 1000$, $M = 1$) vahvistusvaste tulon kaistalla ja päästökaistalla.



Kuva 3.18: Työssä toteutetun CIC-suodattimen lohkokaavio.

Luku 4

Standardiväylät

Monet kaupalliset mikropiirit mikrokontrollereista niiden ohjaamiin anturi- ja muistipiireihin sisältävät jonkin, usein sarjamuotoisen, standardiväylän datan lukemista ja kirjoittamista varten. Väylän avulla voidaan esimerkiksi tallentaa piirille sen toimintaa ohjaavia bittejä tuotetta käytettäessä tai tehdaskalibroinnin yhteydessä tai lukea dataa piiriltä. Esimerkiksi AD-muunninpiireille voidaan kirjoittaa FIR-suodattimien kertoimia, muistipiireille voidaan kirjoittaa mitä vain sisältöä ja anturipiireiltä voidaan lukea tietoa havaittavasta suureesta. Tietoa voitaisiin siirtää myös yksinkertaisen siirtorekisterin tai rinnakkaisväylän avulla. Suunniteltavan mikropiirin väylätyyppiä valitessa ratkaisevia tekijöitä voivat olla esimerkiksi tarvittava tiedonsiirtonopeus, kotelossa tarvittavien jalkojen määrä, piirillä tarvittavien digitaalisolujen määrä tai se, missä järjestelmässä piiriä käytetään ja minkälaisella piirillä tai laitteella piiriä on tarkoitus ohjata.

Sarjamuotoisen standardiväylän käyttämisellä mikropiireissä on useita etuja rinnakkaisväyliin verrattuna. Ensinnäkin monet sarjamuotoiset standardiväylät vaativat usein vain 2-4 signaalikanavaa, kun taas rinnakkaisväylissä voi olla jopa yli 20 kanavaa. Ne vaativat siis vähemmän nastoja piirikomponenteissa, mistä on etua, kun yritetään valmistaa aina vain pienempiä komponentteja. Suuremman jalkamäärän reititys vaatisi huomattavasti enemmän pinta-alaa myös piirilevyllä. Standardoiduissa sarjaväylissä niiden vähätkin signaalijohtimet voidaan usein jakaa useampien samaa väylätyyppiä käyttävien piirien kesken, mikä vähentää reititysten tarvetta entisestään.

Toisena etuna on se, että standardiväylät ovat yleisesti tunnettuja ja hyvin dokumentoituja. Suunnittelija tai harrastelija, joka joutuu käyttämään piiriä, on todennäköisesti joutunut käyttämään yleisesti käytettyjä standardiväyliä jo aikaisemmin jossain muussa yhteydessä. Monet mikrokontrollerit tukevat niitä ja on olemassa esimerkiksi USB-portin ja standardiväylien välisiä muuntimia, joiden avulla anturidata saadaan luettua helposti myös tietokoneelle. Sarjamuotoisten standardiväylien etuna siirtorekisteriin nähden on se, että ne ovat useimmiten pakettipohjaisia, joten niiden avulla voidaan kirjoittaa ja lukea yksittäinen, usein 8-bittinen muistirekisteri kerrallaan lyhyellä tiedonsiirtopurskeella. Anturipiireillä voi olla kymmeniä ja muistipiireillä jopa tuhansia rekistereitä, ja kaikkea tietoa ei yleensä ole järkeä

kirjoittaa tai lukea kerralla, kuten yksinkertaisella siirtorekisterillä pitäisi tehdä. Kolmantena etuna on laitteen liitettävyys muihin samaa väylätyyppiä käyttävien laitteiden kanssa.

Sarjamuotoisten väylien haittapuolena on se, että ne ovat hitaampia kuin rinnakkaisväylät. Kahdeksan bitin kirjoittaminen piirille vaatii vähintään kahdeksan kellopulssia itse dataa varten ja lisäksi usempia kellopulsseja, kun pitää erikseen lähettää kirjoitettavan muistirekisterin osoite ja joissain väylissä vielä kutsua piiriä sen piirikohtaisen osoitteen perusteella.

Liikeanturipiireissä usein käytettyjä väyliä ovat sarjamuotoiset SPI- ja I2C-väylät (engl. *serial peripheral interface, inter-integrated circuit*), joita suosivat isoimmista valmistajista esimerkiksi STMicroelectronics, Analog Devices, Murata ja InvenSense.[19] Näiden valmistajien monet kiihtyvyys- ja kulmanopeusanturipiirit tukevat molempia väyliä, mikä helpottaa piirien käyttöönottoa, sillä monet mikrokontrollerit tukevat juuri SPI- ja I2C-väyliä.

Muita sarjamuotoisia väyliä ovat esimerkiksi etenkin tietokoneista ja niiden oheislaitteista tutut USB, FireWire ja niitä vanhempi RS-232 sekä syntetisaattoreista tuttu MIDI. Rinnakkaisväyliä ovat puolestaan esimerkiksi useimmista elektronisista mittauslaitteista löytyvä GPIB-väylä (engl. *general purpose interface bus*) ja vanhemmista printtereistä tuttu LPT-väylä (engl. *line print terminal*), joka on väistymässä USB:n ja FireWiren tieltä. GPIB- ja LPT-väylissä tietoa siirretään 8-bittinen tavu kerrallaan.

4.1 Väylien ominaisuudet

Väylän määrittelyyn kuuluu mm. siinä käytettävät signaalikanavat, yhteyskäytännöt, kellotaajuudet ja dupleksisuus. Useimmissa väylissä on ainakin kellokanava, yksi tai kaksi datakanavaa sekä mahdollisesti renginvalitsinkanava. Renginvalitsinta käytetään, kun useita piirejä kytketään jakamaan samat kello- ja datasignaalit ja halutaan siirtää tietoa vain osan renkikoneista kanssa. Vain ne rengit kuuntelevat käskyjä, joiden renginvalitsinlinja on aktiivitilassa. Renkikoneiden kutsumiseen voidaan käyttää myös laitekohtaista osoitetta, jonka isäntälaite lähettää kommunikaatiopurskeessa datalinjaa pitkin. Yhteyskäytännöissä määritellään tiedonsiirrossa käytettävä syntaksi, virheenhavainnointi- ja virheenkorjauskeinot sekä signaalien ajoitukset. Tyypillinen datapurske sisältää kutsuttavan laitteen tai piirin osoitteen, suoritettavan käskyn, joka voi olla esimerkiksi lukeminen tai kirjoittaminen, muistirekisterin osoitteen, josta luetaan tai johon kirjoitetaan, sekä luettavan tai kirjoitettavan datan. Virheenhavainnoinnissa voidaan käyttää esimerkiksi pariteettibittiä tai jotain monimutkaisempaa bittivirheentunnistusalgoritmia. Signaalien ajoitusmääritelmissä määritellään kellotaajuudet ja miten datan pitää muuttua suhteessa kelloreunoihin.

Dupleksisuudella tarkoitetaan dataliikenteen suuntaisuutta, joka voi olla yksi-, vuoro- tai kaksisuuntainen (engl. *simplex, half duplex, full duplex*). Yksisuuntaisessa liikenteessä tietoa siirretään vain toiseen suuntaan joko isännältä rengille tai rengiltä isännälle. Vuorosuuntaisessa liikenteessä tietoa siirretään molempiin suuntiin, mutta vuorotellen. Kaksisuuntaisessa

liikenteessä tietoa siirretään molempiin suuntiin samanaikaisesti. Taulukossa 4.1 on listattu SPI- ja I2C-väylien sekä siirtorekisterin ominaisuuksia.

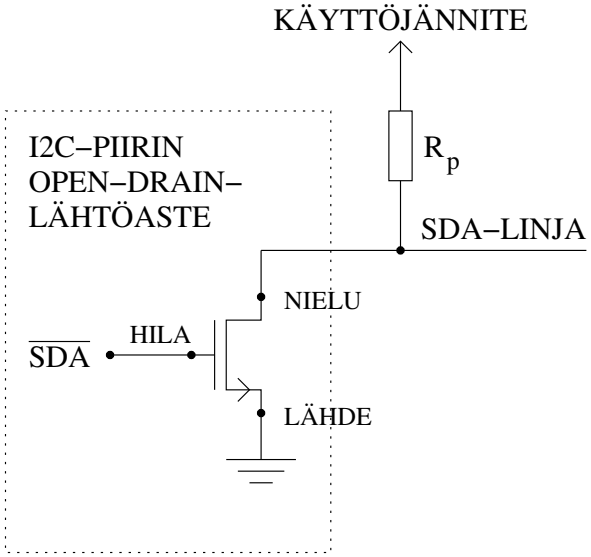
Taulukko 4.1: SPI- ja I2C-väylien ja siirtorekisterin ominaisuuksia.

	SPI	**I2C**	**Siirtorekisteri**
Suurin kellotaajuus	70 MHz	3,4 MHz[1]	Vapaasti valittavissa
Kanavien määrä	väh. 3	2	2-3
Sananleveys	8, 12, 16	8, 16	Vapaasti valittavissa
Dupleksisuus	Kaksisuunt.	Vuorosuunt.	Yksisuunt.

[1]I2C-väylän Hs-tilassa (*high-speed mode*) [20]

4.2 Suurimpedanssinen signaalilähtö

Jotta väylässä kiinni olevista laitteista useampi voisi vuorollaan määrätä laitteiden kesken jaetun linjan jännitteen, laitteet liitetään jaettuun linjaan lähtöasteella, jonka lähtösolmu voidaan asettaa suurimpedassitilaan. Sellaisia ovat esimerkiksi kolmitilapuskurit (engl. *tri-state buffer*) sekä *open-drain*- ja *open-collector*-tyyppiset piirit. Lähdön ollessa suurimpedanssitilassa lähtöaste ei yritä ajaa lähtösolmua loogiseen ykköseen tai nollaan, vaan piiri näyttää suurelta vastukselta lähdöstä päin katsottuna. Jos joku laitteista ajaisi linjaa ykköseksi ja toinen samaan aikaan nollaksi, ei voitaisi olla varmoja, mihin jännitteeseen linja asettuu ja tiedon siirtäminen olisi mahdotonta.



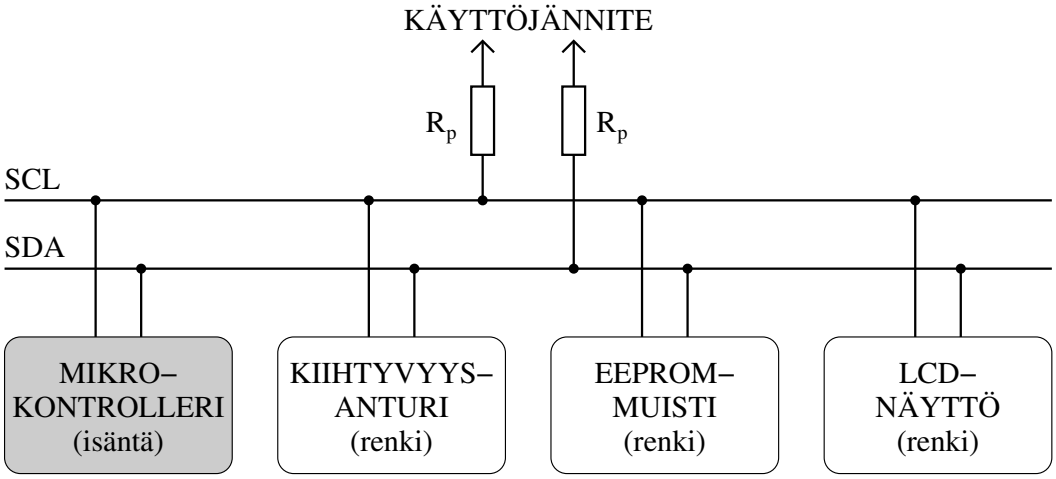
Kuva 4.1: I2C-väylää tukevan laitteen *open-drain*-tyyppinen lähtöaste.

Kuvassa 4.1 on esitetty I2C-väylän *open-drain*-tyyppinen datalähtöaste. Transistori toimii piirissä jänniteohjattuna vastuksena ja SDA-linjan jännite määräytyy jännitteenjaon mukaisesti. Kun SDA-datalinjan jännite pitää pudottaa nollaan, asetetaan NMOS-transistorin

hilajännite käyttöjännitteeseen, jolloin transistori toimii kuin pienenä vastuksena nielun ja lähteen välillä. Silloin jännitteenjaon mukaisesti käyttöjännitteen määräämästä jännitteestä suurin osa on ylösvetovastuksen R_p yli ja SDA-linjan jännite on lähellä nollaa volttia. Kun linja pitää nostaa takaisin ykköseksi, asetetaan hilajännite nollaan, jolloin transistori ei johda, vaan on kuin suuri vastus SDA-linjan ja maan välissä, jolloin suurin osa käyttöjännitteestä onkin transistorin nielun ja lähteen välissä. Koska transistori ei yritä silloin ajaa SDA-linjaa mihinkään jännitteeseen, vaan näyttää suurelta vastukselta signaalilinjan ja maan välissä, lähdön sanotaan olevan suurimpedanssitilassa.

Laitteen ollessa vastaanottavassa tilassa se asettaa lähtönsä suurimpedanssitilaan. Kaikkien SDA-linjassa kiinni olevien laitteiden lähtöjen ollessa suurimpedanssitilassa linja siirtyy käyttöjännitteeseen ja jokin isäntälaite voi alkaa kirjoittaa linjalle.

4.3 I2C-väylä



Kuva 4.2: Esimerkki I2C-väylästä, johon on liitetty yksi isäntä ja kolme renkipiiriä ja -laitetta.

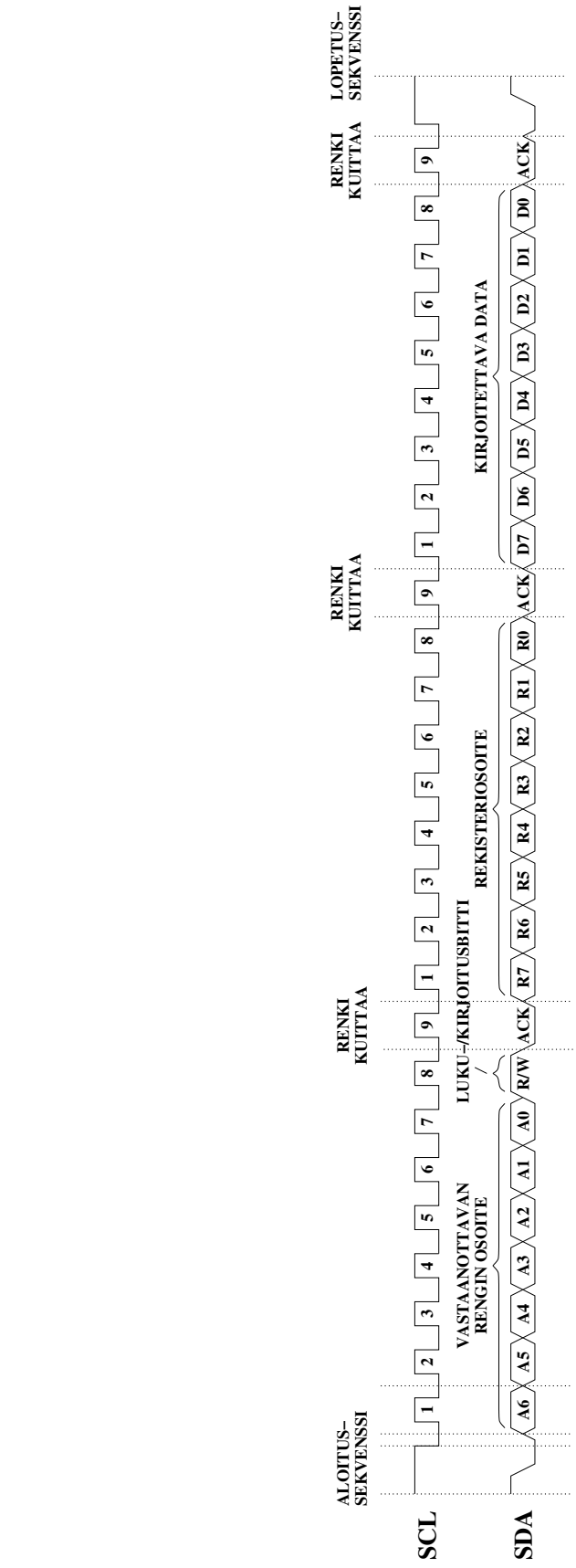
I2C-väylä on Philips Semiconductorsin (nykyinen NXP Semiconductors) kehittämä sarjamuotoinen väylästandardi, jossa käytetään kahta signaalikanavaa: SCL-kellolinjaa (*serial clock*) ja SDA-datalinjaa (*serial data*). Linjat voidaan jakaa useiden isäntä- ja renkilaitteiden kesken, sillä viestit osoitetaan kullekin rengille sen laitekohtaisen osoitteen avulla. Tiedonsiirron aloittaa aina isäntälaite, joka myös tuottaa kellosignaalin. Väylässä voi olla kiinni useampia isäntälaitteita, sillä standardiin kuuluu synkronointi- ja arbitraatiomenetelmät, joiden avulla vältytään tilanteelta, jossa kaksi isäntälaitetta alkaisi lähettää kelloa ja dataa samanaikaisesti jaetuilla linjoilla. Koska datakanavia on vain yksi, on tiedonsiirto joko yksi- tai vuorosuuntaista. Suurin SCL-kellon taajuus on standardin mukaan 3,4 megahertsiä sekunnissa, kun käytetään Hs-tilaa (*high-speed mode*). Linjojen kuormakapasitanssit voivat kuitenkin rajoittaa suurinta käytetävissä olevaa taajuutta. [20] Kuvassa 4.2 on esimerkki I2C-väylästä,

johon on liitetty mikrokontrolleri ja kolme komponenttia, joita se hallitsee: kiihtyvyysanturi, EEPROM-muisti ja LCD-näyttö.

Väylään kytkettyjen laitteiden osoitteet voivat olla seitsemän- tai kymmenenbittisiä, mutta kymmenenbittisiä osoitteita käytetään harvoin kaupallisissa tuotteissa. 7-bittinen osoiteavaruus sallii vain 128 eri osoitetta, joista kahdeksan on varattuja. NXP Semiconductors hallinnoi kaupallisten tuotteiden osoitteiden jakelua, jotta eri tyyppisille piireille ei tule samaa osoitetta. Osoitteiden päällekkäisyyksiä ehkäistään joissain piireissä myös siten, että osoitteen muutaman bitin arvon voi määrittää piirin tätä tarkoitusta varten tarkoitettuihin nastoihin asetettavilla jännitteillä.

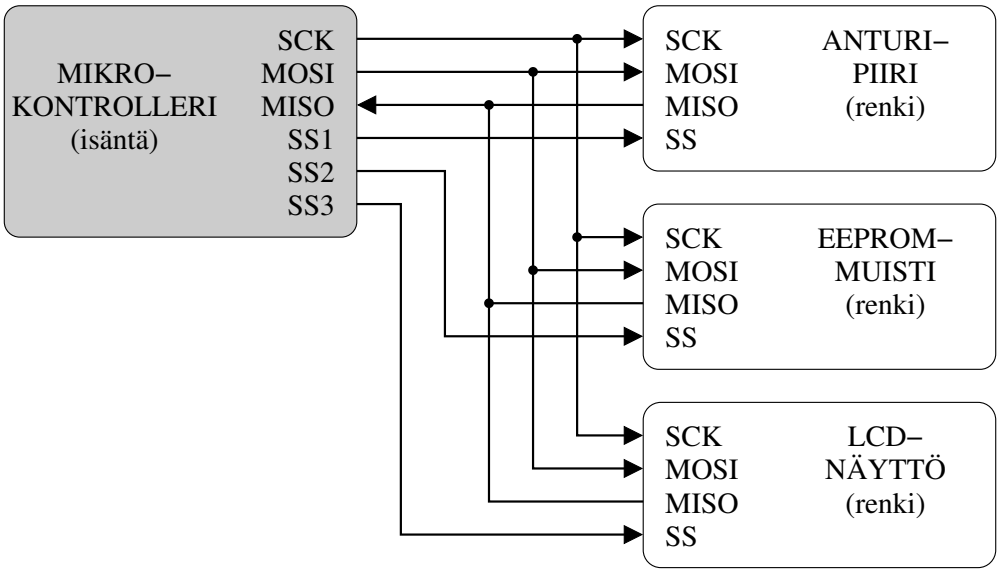
Kuvassa 4.3 on esitetty esimerkki I2C-väylän datapurskeesta, jolla kirjoitetaan dataa rengin muistirekisteriin. Alussa, kun liikennettä ei vielä ole, molemmat linjat ovat käyttöjännitteessä. Liikennöinti alkaa aloitussekvenssillä. Vain isännät voivat aloittaa tiedonsiirron ja ne ovat ainoat laitteet, jotka lähettävät kellosignaalia. Aloitussekvenssissä isäntä asettaa datalinjan loogiseen arvoon 0, kun kellolinja on vielä ylhäällä. Sen jälkeen isäntä lähettää kellolinjaa pitkin yhdeksän pulssia. Seitsemän ensimmäisen pulssin aikana se lähettää datalinjaa pitkin vastaanottavan rengin seitsemänbittisen osoitteen. Kahdeksas bitti on luku- ja kirjoitusbitti. Sen arvo 0 tarkoittaa, että seuraa lukukäsky, ja arvo 1, että seuraa kirjoituskäsky. Datalinjan arvojen tulee vaihtua kellon ollessa alhaalla, jotta muut laitteet eivät tulkitse liikennöintiä aloitus- tai lopetussekvenssiksi. Yhdeksännen kellopulssin aikana rengin tulee kuitata, että se on valmis vastaanottamaan käskyjä. Yhdeksännen pulssin nousevalla reunalla isäntä lakkaa ajamasta datalinjaa ja renkikoneen tulee ajaa se arvoon 0, jotta isäntä saa tiedon, että renki on valmis vastaanottamaan dataa. Jos mikään renki ei vastaa, SDA-linja pysyy oletusarvossa eli ylhäällä. Pulssin jälkeen renki vapauttaa datalinjan ja isäntä voi seuraavien kahdeksan kellopulssin aikana kirjoittaa siihen rekisteriosoitteen, johon dataa kirjoitetaan. Kun rekisteriosoite on kirjoitettu, renki kuittaa jälleen. Seuraavaksi isäntä lähettää kirjoitettavan datan, minkä jälkeen rengin tulee vielä kuitata. Liikennöinnin lopuksi tulee lopetussekvenssi: isäntä nostaa kellolinjan käyttöjännitteeseen ja sen jälkeen myös datalinjan.[20]

I2C-väylää käytetään tiedonsiirtoon monissa elektroniikkakomponenteissa, esimerkiksi antureissa, EEPROM-muisteissa, LCD-näytöissä ja DSP-piireissä. Sitä käytetään myös VGA-, DVI- ja HDMI-liitännöissä [21] erinäisten tietojen, kuten näyttöjen resoluutiotietojen sekä kirkkaus- ja kontrastiasetusten, siirtämiseen.



Kuva 4.3: Esimerkki I2C-väylän datankirjoituspurskeesta.

4.4 SPI-väylä



Kuva 4.4: Standardin mukaisesti kytketty SPI-väylä laitteineen.

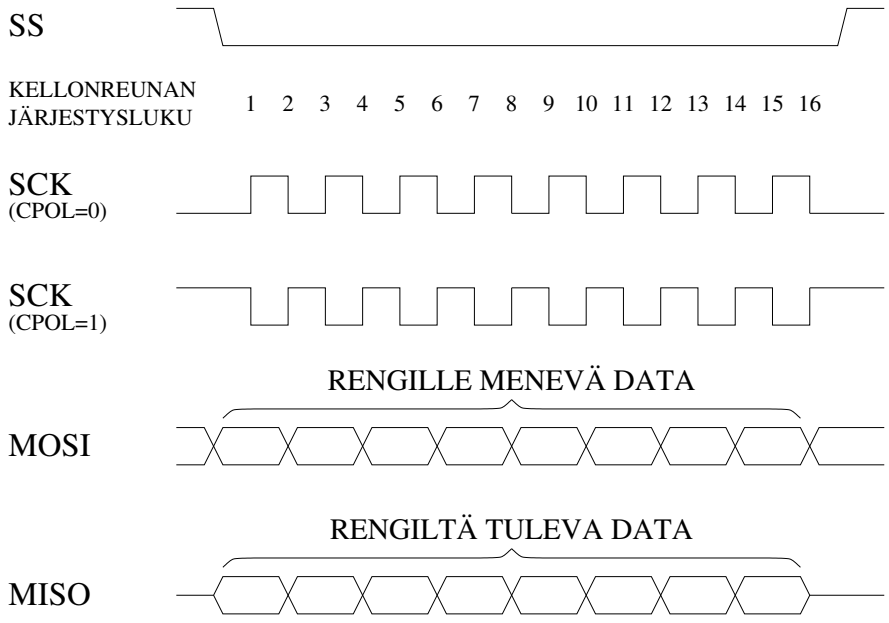
SPI-väylä on Motorolan kehittämä sarjamuotoinen väylä, joka käyttää neljää signaalikanavaa: SCK-kellolinjaa, MOSI- ja MISO-datalinjoja sekä alhaalla aktiivista SS-renginvalitsinlinjaa. Isäntä lähettää dataa rengille MOSI-linjaa pitkin ja renki isännälle MISO-linjaa pitkin. Jos dataa ei tarvitse vastaanottaa rengiltä, voidaan MISO-linja jättää käyttämättä. Kello- ja datalinjat voidaan jakaa useampien laitteiden kesken ja renginvalitsinlinjaa käytetään määrittämään se renki, jonka kanssa isäntä kommunikoi. [22]

Usein ei voida lähettää useille laitteille samoja käskyjä samanaikaisesti, koska käskyt ovat piirikohtaisia, joten jokainen renki tarvitsee oman valitsinlinjan SSi isännältä. Useiden renkien verkossa tarvitaan siis huomattavasti enemmän signaalilinjoja kuin I2C-väylässä. Poikkeuksena on ketjutettu kokoonpano, jossa renkilaitteet kytketään sarjaan, SS- ja kellolinja menevät joka rengille, ja rengit välittävät käskyt MISO-lähdöstä seuraavan rengin MOSI-tuloon. SPI-väylä tyypillisillä kytkennöillä on esitetty kuvassa 4.4. Väylästä on määritelty myös vain yhtä kaksisuuntaista datalinjaa käyttävä kolmen linjan kytkentä, jossa MOSI- ja MISO-kanavat korvataan MOMI-kanavalla.[22]

Kuvassa 4.5 on esitetty tyypillinen SPI-liikenteen datapurske. Dataliikenne purskeen sisällä on väljästi määritelty, mutta noudattaa muutamia perussääntöjä. Purske alkaa aina siten, että isäntä asettaa SS-linjan nollaan, jolloin renki valmistautuu vastaanottamaan komennon. Sen jälkeen isäntä lähettää N kappaletta kellopulsseja ja purskeen lopuksi nostaa SS-linjan takaisin ykköseksi. Renki voi lähettää dataa isännälle SS-linjan ollessa alhaalla, mutta purskeen ulkopuolella sen tulee asettaa MISO-linja suurimpedanssitilaan, koska linja jaetaan muiden renkien kanssa ja niiden pitää voida lähettää sitä pitkin dataa omalla vuorollaan. Purskeen sisältämien kellopulssien määrää ei ole rajattu väylän määrittelyssä, vaan se vaihtelee renkilaitekohtaisesti. Renkilaitteesta riippuu myös se, siirretäänkö dataa eniten vai vähi-

ten merkitsevä bitti ensin. Määrittely ei rajoita kellotaajuusaluetta, vaan sekin riippuu rengistä. Anturipiireissä käytettyjä kellotaajuuksia on ainakin välillä 2-25 MHz:iä [19]. Toisin kuin I2C-väylässä, SPI:ssä ei ole määritelty kuittausmenetelmää, joten tietojen kirjoittamisen onnistuminen pitää varmentaa jollain muulla keinolla. Isäntä voi esimerkiksi pyytää kirjoitettua dataa piiriltä kirjoittamisen jälkeen uudella datapurskeella ja verrata siihen dataan, joka piti kirjoittaa.

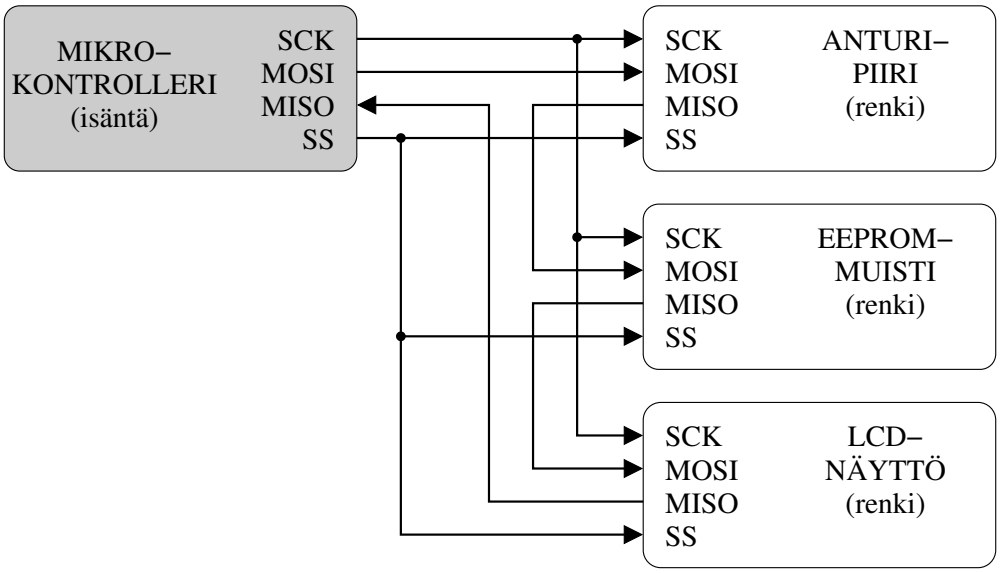
Datalinjan näytteistämiskohdan määrittää standardin mukaan kaksi muuttujaa: kellon polariteetin määrittävä CPOL ja kellon vaiheen määrittävä CPHA. Data näytteistetään nousevalla reunalla, jos CPOL on nolla, ja laskevalla reunalla, jos CPOL on yksi. Data näytteistetään parittomilla kellonreunoilla, jos CPHA on nolla. Silloin datalinjojen arvojen tulee päivittyä parillisilla kellonreunoilla. Vastaavasti jos CPHA on yksi, näytteistetään data parillisilla kellonreunoilla ja silloin datalinjojen arvot pitää päivittää parillisilla kellonreunoilla. Näytteistysajankohdan määrittely koskee sekä isäntää että renkiä. Numeroidut kellonreunat on esitetty kuvassa 4.5. [22]



Kuva 4.5: Kahdeksan kellopulssin mittainen SPI-tietoliikennepurske.

Jotkin SPI:tä tukevat laitteet tukevat ketjutusta (engl. *daisy-chaining*). Ketjutetussa kokoonpanossa rengit kytketään sarjaan, kuten kuvassa 4.6, ja yksi kello- ja yksi SS-linja yhdistetään useammalle rengille. Isännältä tuleva MOSI-linja yhdistetään ensimmäisen rengin MOSI-tuloon. Loput rengit ovat sarjassa tämän rengin perässä siten, että edellisen rengin MISO-lähtö yhdistetään seuraavan rengin MOSI-tuloon. Ketjutetussa kokoonpanossa olevilta rengeiltä vaaditaan, että ne pystyvät lähettämään MOSI-tuloon saamansa käskyn MISO-lähtönsä kautta eteenpäin, kun useampia komentosyklejä tulee peräkkäin ilman että SS-linja nousee ylös. Lähettäessään komennon jokaiselle ketjussa olevalle rengille isäntä ensin lähettää yhden, esimerkiksi 16 kellopulssin mittaisen komentosyklin, mutta ei nostakaan vielä SS-linjaa ylös. Seuraavaksi isäntä lähettää toisen 16 kellopulssin mittaisen komentosyklin,

minkä aikana ensimmäinen renki välittää ensimmäisen komennon toiselle rengille ja vastaanottaa itse toisen käskyn. Kolmannen 16 kellopulssin sarjan aikana toinen renki lähettää ensimmäisen komennon kolmannelle rengille ja ensimmäinen renki välittää toisen komennon toiselle rengille ja vastaanottaa itse tämän kolmannen komennon. Jos ketjussa on N laitetta, isäntä lähettää N komentosykliä, joiden jälkeen viimeinen renki on saanut ensimmäisenä lähetetyn käskyn ja ensimmäinen renki on saanut viimeisenä lähetetyn käskyn. Kun isäntä nostaa lopuksi SS-linjan ylös, rengit toteuttavat viimeisimpänä saamansa käskyn. Ketjutusta ei ole määritelty SPI-väylän alkuperäisessä ohjekirjassa [22], vaan se on joidenkin valmistajien piireihinsä lisäämä ominaisuus. [23]

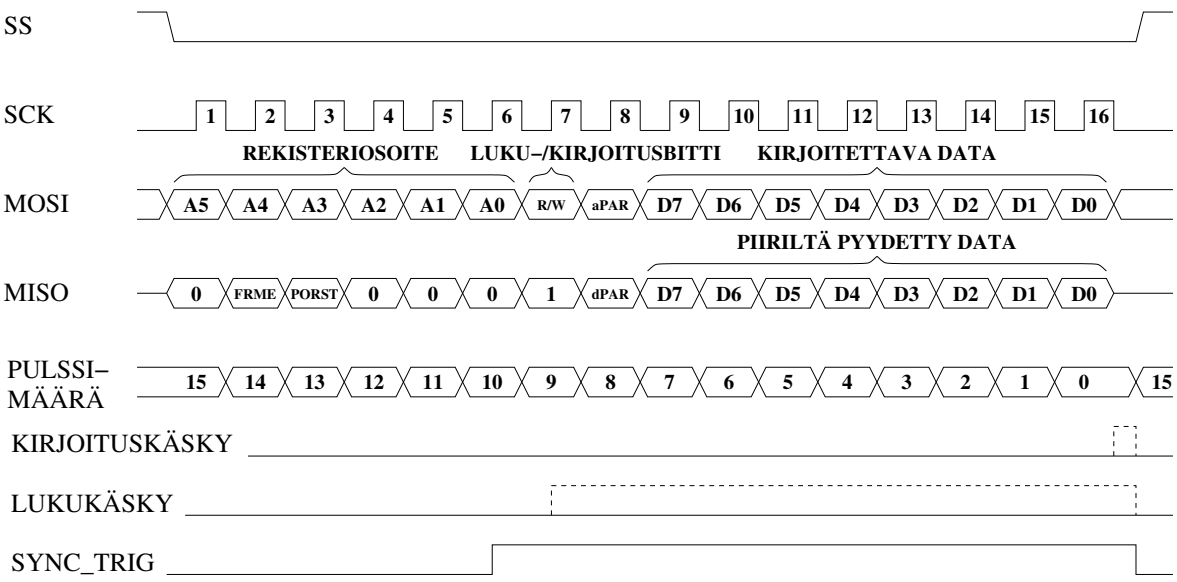


Kuva 4.6: Ketjutettu SPI-väylä laitteineen

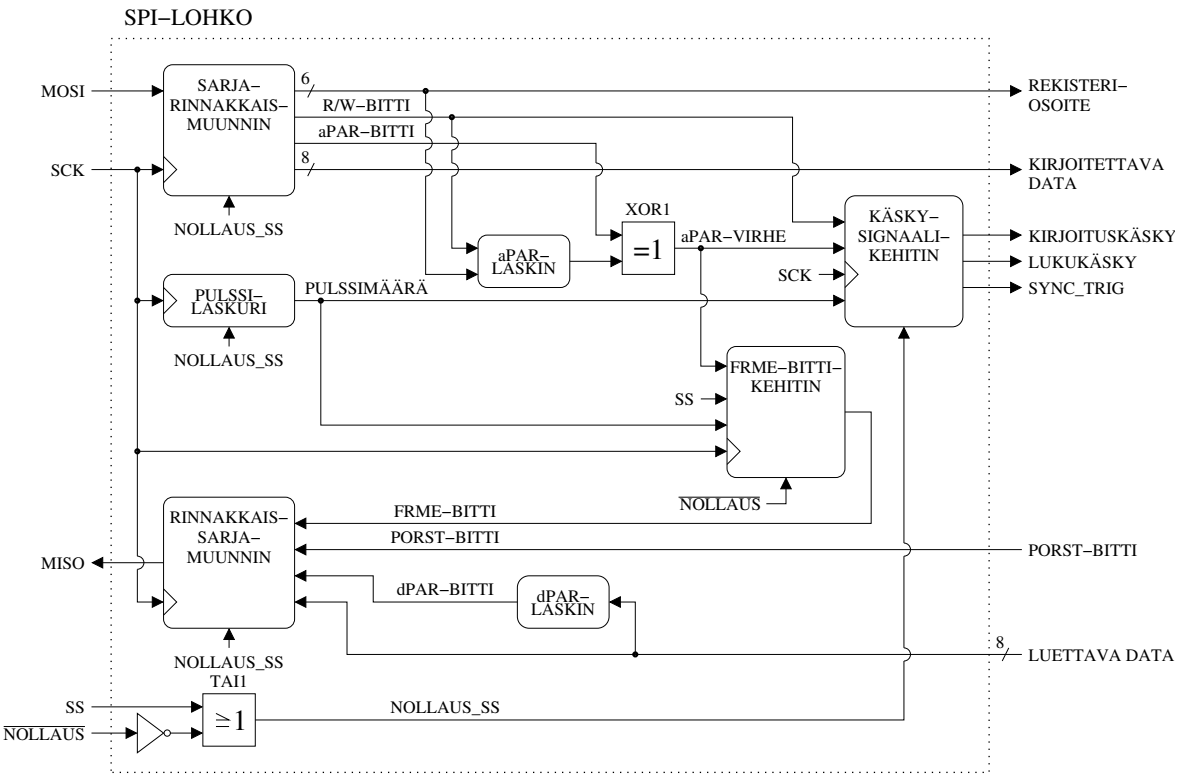
4.5 Toteutettu SPI-väylä

Tässä työssä toteutettiin karsittu SPI-väylä, joka noudattaa osittain Muratan kaupallisten kiihtyvyysantureiden SPI-väylämäärittelyjä [24]. Piirin kanssa kommunikoidaan 16 kellojakson tiedonsiirtopurskeilla ja R/W-, FRME-, PORST-, aPAR- ja dPAR-bittien toiminta on vastaavanlaista. Rekisteriosoitteet ovat 6-bittisiä ja datarekistereiden koko on kahdeksan bittiä. Niin rekisteriosoite- kuin databititkin sijaitsevat datapurskeessa väylämäärittelyjä vastaavilla paikoilla. Näytteistämisen polariteettiarvo CPOL on 0 ja vaihearvo CPHA on 0, eli kellosignaalin pitää olla arvossa 0 tiedonsiirtopurskeen alussa ja data näytteistetään nousevilla kellonreunoilla. SS-, SCK-, MOSI- ja MISO-signaalien ja muutamien lohkon sisäisten signaalien ajoitukset on esitetty kuvassa 4.7.

Toteutetun SPI-väylälohkon lohkokaavio on esitetty kuvassa 4.8. Data tulee SPI-isäntälaitteelta MOSI-linjaa pitkin SPI-väylän SCK-kellon kellottamana ja menee sarja-rinnakkaismuuntimelle, joka muuntaa sarjamuodossa tulevan datan rinnakkaismuotoiseksi. Kah-



Kuva 4.7: Toteutetun digitaalilohkon SPI-tietoliikennepurske.



Kuva 4.8: SPI-väylälohkon lohkokaavio.

deksan ensimmäistä MOSI-linjaa pitkin tulevaa bittiä muodostavat käskysanan. Sen kuusi ensimmäistä bittiä määrittävät kirjoitettavan tai luettavan rekisterin osoitteen, joka välitetään muille alilohkoille 6-bittisellä rekisteriosoiteväylällä. Seitsemäs bitti eli R/W-bitti määrittää, kirjoitetaanko rekisteriin dataa vai luetaanko sitä. Arvo 0 tarkoittaa lukukäskyä ja arvo 1 kirjoituskäskyä. Kahdeksas bitti, aPAR, on osoitteen ja R/W-bitin pariton pariteettibitti, jonka avulla voidaan havaita, onko käskysanassa pariton määrä virheellisiä bittejä. Virheen havain-

nointi tehdään siten, että käskysanan pariton pariteettibitti lasketaan myös SPI-lohkon sisällä aPAR-laskinlohkolla, ja sitä verrataan MOSI-linjaa pitkin tulleeseen aPAR-bitin arvoon poissulkeva tai-portilla XOR1. XOR1-portin lähdön eli aPAR-virhelinjan arvo 1 tarkoittaa, että virhe on tapahtunut, ja arvo 0 vastaavasti, että virhettä ei ole tapahtunut. Kahdeksan seuraavaa MOSI-linjaa pitkin tulevaa bittiä sisältävät piirille kirjoitettavan datan ja ne ohjataan 8-bittiselle kirjoitettavan datan väylälle.

aPAR-virheen ilmaiseva signaali menee R/W-bitin kanssa käskysignaalikehitinlohkolle, joka nostaa kirjoituskäskylinjan ylös datapurskeen lopuksi, jos kyseessä on kirjoituskäsky ja aPAR-virhelinja on alhaalla. Käskysignaalikehitinlohko luo pulssin puolestaan lukukäskylinjalla, jos kyseessä on lukukäsky. Synkronoijalohkolle menevä SYNC_TRIG-pulssi luodaan joka purskeen aikana huolimatta siitä, onko kyseessä luku- vai kirjoituskäsky. Sen merkitys selitetään tarkemmin kappaleessa 7. Käskysignaalikehitinlohko ajoittaa lähtevät käskyt SCK-kellon ja pulssilaskurilta tulevan pulssimäärän avulla.

Sarja-rinnakkaismuunnin muuntaa piiriltä SPI-isännälle MISO-linjaa pitkin lähetettävät tiedot sarjamuotoon. Sille menee FRME-bittikehittimeltä FRME-bitti, joka nostetaan ylös kommunikaatiovirheen merkiksi, jos datapurskeen SCK-pulssimäärä ei ole ollut 16 tai jos aPAR-virheen arvo on ollut 1. SPI-purskeessa MISO-linjaa pitkin lähetettävä FRME-bitin arvo kertoo aina edellisen SPI-purskeen aikana tapahtuneesta virheestä, koska arvo päivittyy vasta SS-linjan noustessa ylös. Rinnakkais-sarjamuuntimelle menee myös PORST-bitti, joka tulee muistirekisterilohkolta kontrollirekisteristä. PORST-bitin arvo on 1, kun piirin käyttöjännite nostetaan ylös ja sen arvo voidaan vaihtaa nollaksi kirjoittamalla uusi arvo kontrollirekisteriin. Sen jälkeen tiedetään, että piiri on nollautunut, jos bitin arvo muuttuu takaisin ykköseksi, vaikka kontrollirekisterin arvoa ei ole muutettu manuaalisesti. PORST-bitti lähetetään SPI-isännälle joka tiedonsiirtopurskeessa. FRME- ja PORST-bitin lisäksi muunninlohkolle menee muistirekisterilohkolta saatava luettavan datarekisterin sisältö ja siitä dPAR-laskimen laskema pariton dPAR-pariteettibitti.

Kaikki lohkot saadaan nollattua alkutilaan alhaalla aktiivisella, asynkronisella nollaustulolla. Osa lohkoista nollataan myös, kun SS-linja nostetaan ylös. Toteutetun SPI-lohkon sisältämien digitaalisolujen viemä pinta-ala on 0,017 mm^2. Lohko toimii simulaatioiden mukaan hitaimmissakin toimintaolosuhteissa SCK-kellon taajuudella f_{SCK} = 8 MHz.

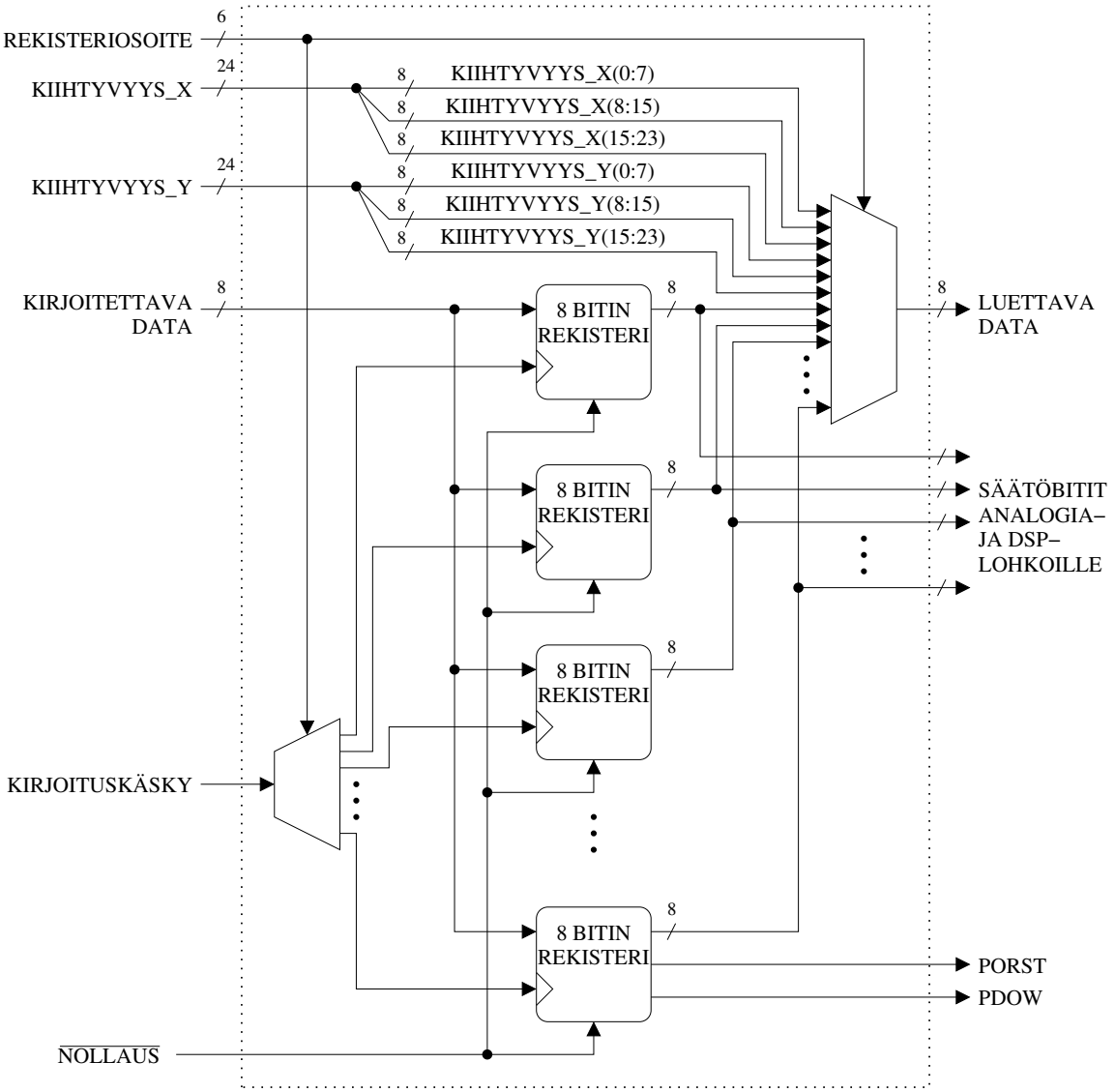
Luku 5

Muistirekisterit

Muistirekisterilohkossa säilötään 29:n ja käsitellään 35:n kahdeksanbittisen rekisterin tietoja. Lohkon sisäisissä rekistereissä pidetään tallessa DSP- ja analogialohkojen säätöbittejä sekä PORST- ja PDOW-bittiä. Lohko ohjaa myös synkronoijalta tulevan kiihtyvyysdatan SPI-lohkolle sitä kysyttäessä, mutta sitä ei säilytetä muistirekisterilohkossa. Muistirekisterilohko itsessään sisältää siis vain 29 kahdeksan bitin D-kiikkurekisteriä. Jokaisella kahdeksan kiikun rekisterillä on oma kuusibittinen osoitteensa. Rekisterien sisältö on esitetty taulukossa 5.1. Koska muistirekistereinä toimivat D-kiikut, data kadotetaan, kun anturipiirin käyttöjännite sammutetaan. Varsinaisten muistirekisterien lisäksi lohko sisältää kontrollilogiikkaa kirjoitettavan datan ohjaamiseksi oikeisiin rekistereihin ja SPI-väylän kautta kysyttävän datan ohjaamiseksi SPI-rajapintalohkolle. Osassa rekistereitä säilytetään alle kahdeksaa bittiä, mutta synteesiohjelma optimoi piirin siten, ettei tyhjiä muistibittejä varten luoda logiikkaportteja lopulliselle piirille.

Kuvassa 5.1 on esitetty muistirekisterilohkon lohkokaavio. Nollaussignaalin tulee olla ylhäällä, jotta rekistereihin voidaan kirjoittaa dataa. SPI-lohkolta tuleva kirjoitettava data välitetään suoraan jokaiselle kahdeksan bitin kiikkulohkolle. Kirjoituskäskysignaali demultiplexataan sille rekisterilohkolle, jonka osoite tulee kuusibittistä rekisteriosoiteväylää pitkin SPI-lohkolta. SPI-lohkolle menevään kahdeksan bitin lähtödataväylään multiplexataan sen rekisterin sisältö, jonka rekisteriosoiteväylä määrittää. Lähtöväylässä on siis oikea data heti viimeisen SPI-väylältä tulevan osoitebitin saapumisen jälkeen.

Lohkon lähtöinä on SPI-lohkolle menevän lähtödatan lisäksi analogia- ja DSP-lohkojen säätöbitit. Lisäksi kontrollirekisteristä ohjataan PDOW-bitti DSP-lohkolle ja PORST-bitti SPI-lohkolle. Toteutetun muistirekisterilohkon digitaalisolut vievät pinta-alaa 0,093 mm^2 käytössä olevalla 0,35 μm CMOS-teknologialla.



Kuva 5.1: Muistirekisterilohkon lohkokaavio.

Taulukko 5.1: Muistirekisterilohkon käsittelemät tiedot sekä niille varatut rekisteri- ja bittimäärät.

Data	**Varattuja rekisterejä**	**Bittimäärä**
PORST- ja PDOW-bitit	1	2
Kiihtyvyysdata / x-akseli[1]	(3)	(24)
Kiihtyvyysdata / y-akseli[1]	(3)	(24)
Säätöbitit analogialohkoille	16	128
Poikkeamankorjain / x-akseli	3	24
Poikkeamankorjain / y-akseli	3	24
Vahvistus / x-akseli	2	12
Vahvistus / y-akseli	2	12
Taajuudenjakosuhde	2	10
Yhteensä	**29 (35)**	**212 (260)**

[1]Kiihtyvyysdataa ei tallenneta muistirekisterilohkoon, vaan se tulee synkronoijalohkolta.

Luku 6

Lähtöarvon poikkeaman ja vahvistuksen korjaus

Tämän työn DSP-lohkolle tuleva digitaalinen kiihtyvyysarvo kasvaa lineaarisesti suhteessa kiihtyvyyteen ja noudattaa ensimmäisen asteen yhtälöä $y = k \cdot a + c$, jossa y on lähtöarvo, a on kiihtyvyys, k on kulmakerroin ja c on poikkeama. Kaavasta on jätetty pois todellisuudessa mukana olevat toisen ja kolmannen asteen yms. epälineaarisuustermit ja kohina. Tavoitteena on, että poikkeama olisi $c = 0$, ja että jokaisella yksittäisellä anturipiirillä kulmakerroin k olisi samanarvoinen.

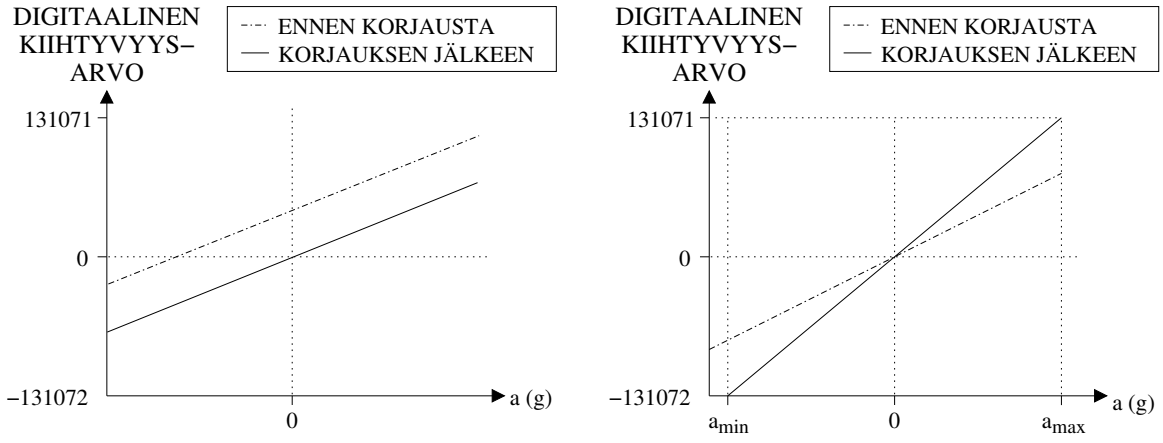
Kulmakerroin ja poikkeama vaihtelevat piirikohtaisesti mikropiirin ja MEMS-elementin prosessivariaatiosta (engl. *process variation*) johtuen. Prosessivariaation takia mikropiirien sisäisten komponenttien mitat ja muut ominaisuudet, kuten esimerkiksi neliöalan vastusarvot (engl. *sheet resistance*) ja transistorien kynnysjännitteet, vaihtelevat.[25] Sen seurauksena kapasitanssi- ja vastusarvot, transistorien vahvistukset yms. vaihtelevat, minkä takia eri piireissä toisiaan vastaavissa virroissa ja jänniteissä on eroja. On kuitenkin toivottavaa, että anturipiirisarjassa kaikkien piirien lähdöt vastaisivat toisiaan halutulla tarkkuudella.

Anturipiirien lähtölukemat saadaan vastaamaan toisiaan esimerkiksi prosentin tarkkuudella korjaamalla poikkeama ja kulmakerroin. Korjaus voidaan tehdä analogisesti, digitaalisesti tai molempia menetelmiä hyödyntäen. Vaihtelevista kapasitansseista johtuvia ongelmia voidaan korjata analogisesti esimerkiksi korvaamalla yksittäiset kondensaattorit kondensaattorimatriiseilla, jolloin kapasitansseja voidaan säätää käyttämällä vain osaa matriisin pienistä kondensaattoreista. Poikkeamaa voitaisiin korjata esimerkiksi analogisella ylipäästösuodattimella, vaihtelevat virrat voitaisiin korjata käyttämällä rinnakkaisia, ohjattavia virtalähteitä jne.

Jos piirillä muutetaan analoginen havaintoarvo digitaaliseen muotoon, voidaan piirille lisätä digitaalisia korjausfunktioita. Automatisoitavien synteesi- ja sijoittelu- ja reititysohjelmien avulla toteutettavan digitaalisuunnitteluvuon ansiosta digitaalisten signaalinkäsittelylohkojen toteuttaminen voi olla huomattavasti nopeampaa kuin analogialohkojen toteuttaminen. Yksinkertaisen DSP-lohkon, esimerkiksi summaimen, vähentimen tai kertojan, toimin-

ta voidaan saada mahtumaan jopa vain muutamaan riviin laitteistonkuvauskielistä koodia, esimerkiksi VHDL-koodia, josta synteesiohjelmalla saadaan luotua piirikaavio ja sijoitteluja reititystyökalulla edelleen mikropiirilohkon piirikuvio (engl. *layout*). Mikropiiriteknologioiden pienentyessä digitaalilohkojen vaatima pinta-ala mikropiirillä pienenee, minkä takia digitaalinen korjaus on pinta-alan suhteen aina vain edullisempaa.

Poikkeaman korjauksen vaikutus kiihtyvyysanturipiirin digitaaliseen lähtöön on havainnollistettu kuvassa 6.1 vasemmalla ja vahvistuksen korjauksen vaikutus oikealla. Poikkeama on korjattu siten, että kiihtyvyyden ollessa a = 0, on digitaalinen kiihtyvyysarvo 0. Vahvistus on korjattu siten, että suurin digitaalinen lähtöarvo vastaa suurinta kiihtyvyyttä a_{max}, joka piirillä on tarkoitus voida ilmaista. Pienin digitaalinen arvo vastaavasti vastaa pienintä mitattavaa kiihtyvyyttä a_{min}. Kaupallisissa kiihtyvyysanturipiireissä mitattavissa olevat arvoalueet ovat esimerkiksi $\pm 1g$, $\pm 2g$, $\pm 6g$ jne.



Kuva 6.1: Vasemmalla poikkeaman korjauksen ja oikealla vahvistuksen korjauksen vaikutus digitaaliseen kiihtyvyysdataan.

6.1 Booth-kertoja

Booth-kertoja on Andrew D. Boothin kehittämä algoritmi [26], jolla voidaan kertoa keskenään kaksi kahden komplementtimuodossa olevaa binäärilukua. Se perustuu silmukassa suoritettaviin yhteen- ja vähennyslaskuihin ja aritmeettiseen siirtoon (engl. *arithmetic shift*). Booth-kertojan vuokaavio on esitetty kuvassa 6.2. Olkoon binääriluku m kerrottava, jossa on x bittiä, ja binääriluku r kertoja, jossa on y bittiä. Silmukassa joko lisätään luku m lukuun r tai vähennetään se siitä tiettyjen ehtojen mukaisesti. Käytännön VHDL-toteutuksessa sitä varten voidaan muodostaa kaksi uutta binäärilukua A ja P, joiden kummankin pituus on $x + y + 1$ bittiä. Binäärilukuun lukuun A sijoitetaan kertoja m ja lukuun P sijoitetaan kerrottava r seuraavasti, missä symboli $\&$ tarkoittaa bittijonojen liittämistä ketjuksi:

$A = m \ \& \ (y+1) \ kappaletta \ nollia$
$P = x \ kappaletta \ nollia \ \& \ r \ \& \ 0$

Silmukka suoritetaan y kertaa ja siinä tarkastellaan luvun P kahta vähiten merkitsevää bittiä. Olkoon q näistä kahdesta bitistä muodostettu luku. Luvun q arvon perusteella suoritetaan seuraavat toimenpiteet:

- Jos $q = 00_2$, suoritetaan vain P:n aritmeettinen siirto oikealle.
- Jos $q = 01_2$, on $P = P + A$ ja sen jälkeen suoritetaan P:n aritmeettinen siirto oikealle.
- Jos $q = 10_2$, on $P = P - A$ ja sen jälkeen suoritetaan P:n aritmeettinen siirto oikealle.
- Jos $q = 11_2$, suoritetaan vain P:n aritmeettinen siirto oikealle.

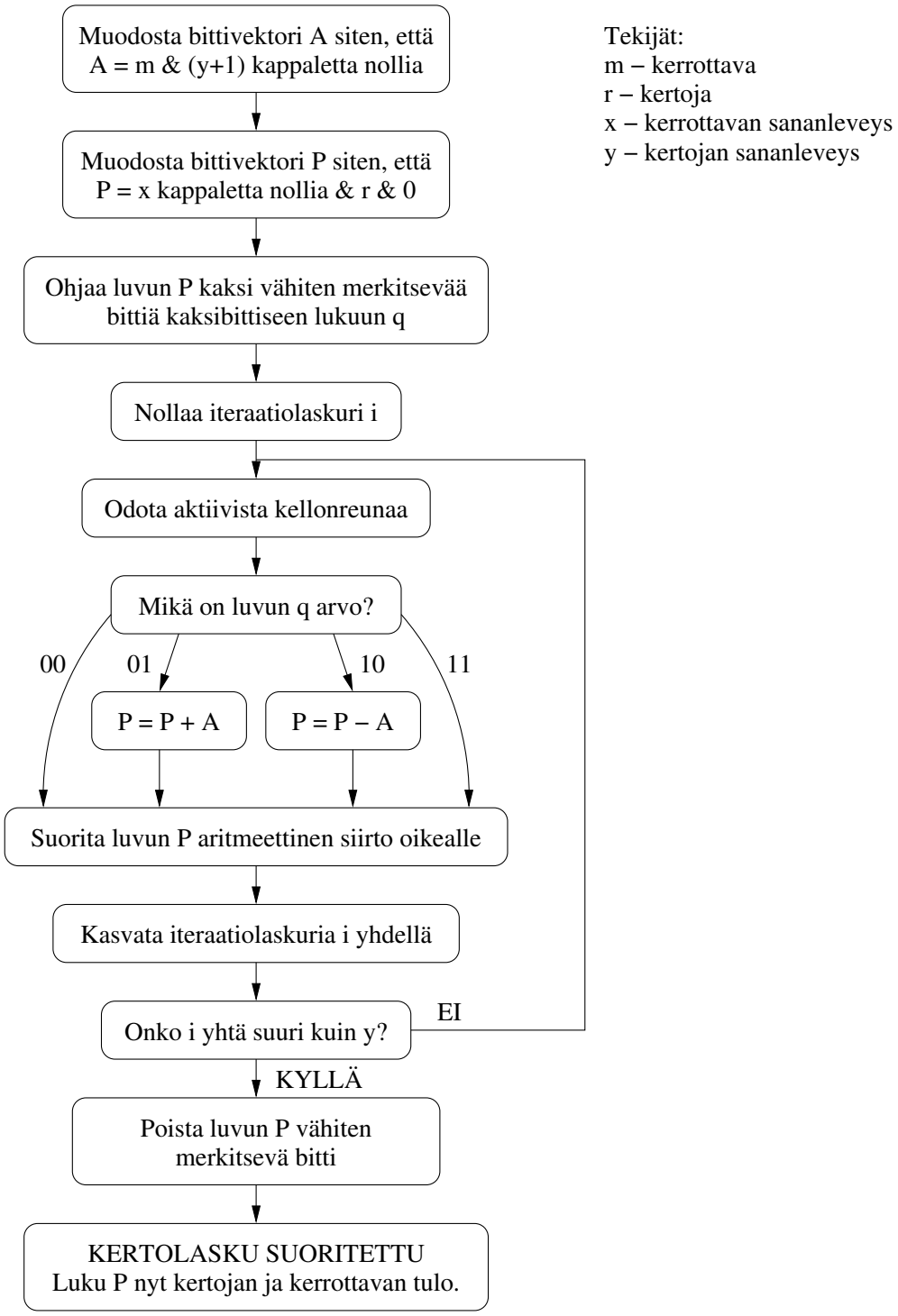
Kun silmukka on suoritettu y kertaa, pudotetaan P:n vähiten merkitsevä bitti pois, jolloin jäljelle jäävä luku on lukujen m ja r tulo. [26] [27]

6.1.1 Esimerkki Booth-kertolaskusta

Olkoon kerrottavat luvut $m = 3_{10} = 0011_2$ ja $r = -4_{10} = 1100_2$, joille $x = y = 4$. Binääriluvut ovat kahden komplementtimuodossa ja niistä saadaan edelleen luvut A ja P:

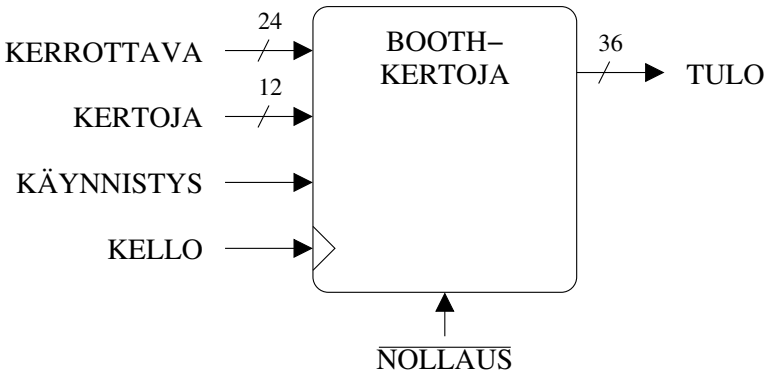
$A = 0011\ 0000\ 0_2$
$P = 0000\ 1100\ 0_2$

- Alussa on $q = 00_2$. Silmukan ensimmäisellä kierroksella suoritetaan siis vain P:n aritmeettinen siirto oikealle, jolloin saadaan $P = 0000\ 0110\ 0_2$.
- Suoritetaan silmukka toisen kerran. Nyt on edelleen $q = 00_2$, joten suoritetaan P:n aritmeettinen siirto oikealle ja saadaan $P = 0000\ 0011\ 0_2$.
- Suoritetaan silmukka kolmannen kerran. Nyt on $q = 10_2$, eli suoritetaan vähennyslasku $P = P - A$ ja sen jälkeen P:n aritmeettinen siirto oikealle. Saadaan $P = 1110\ 1001\ 1_2$.
- Silmukan neljännellä kierroksella on nyt $q = 11_2$. Suoritetaan P:n aritmeettinen siirto oikealle. Saadaan $P = 1111\ 0100\ 1_2$.
- Silmukka on suoritettu $y = 4$ kertaa, joten pudotetaan P:n vähiten merkitsevä bitti pois, jolloin jäljelle jää lukujen m ja r tulo: $m \cdot r = 1111\ 0100_2 = -12_{10}$.



Kuva 6.2: Booth-kertojan vuokaavio.

6.2 Toteutettu Booth-kertoja



Kuva 6.3: Toteutetun Booth-kertojan tulot ja lähdöt.

Kuvassa 6.3 on esitetty toteutetun Booth-kertojalohkon tulot ja lähdöt. Lohkon ylemmän tason kytkennät on esitetty aiemmin kuvassa 2.2. Nollaussignaalin pitää olla ylhäällä, jotta kertoja voi toimia. Kertojatuloon tulee vahvistuskorjain muistirekisterilohkolta ja kerrottavatuloon CIC-suodattimen lähdöstä kiihtyvyysarvo, joka on typistetty 24-bittiseksi. Käynnistystuloon tulee BOOTH_RUN-signaali ja kellotuloon AD-muuntimen lähtöä kellottava DSM_CLK-signaali.

Kun käynnistyslinja nousee ylös, kertojasta ja kerrottavasta muodostetaan bittivektorit A ja P kuten kappaleessa 6.1 on esitetty. Sen jälkeen suoritetaan aritmeettiset siirrot, summaukset ja vähennykset kellosignaalin tahdittamana. Silmukkaa suoritetaan niin kauan, että lasku on suoritettu tai käynnistyslinja laskee alas. Lohko ei anna signaalia kertolaskun valmistumisen merkiksi, vaan järjestelmä on suunniteltu niin, että laskutoimenpiteet saadaan suoritetuksi siihen mennessä, kun lohkon lähtöarvo liipaistaan toiseen DSP-lohkon välirekistereistä (kts. kuva 2.2). Nollaussignaalin ollessa alhaalla lohko ei tee aktiivisesti mitään, vaan ainoastaan syöttää lähtöön kerrottavan luvun luvulla 12 loogisesti vasemmalle siirrettynä (engl. *logical shift*).

Booth-kertojan tuottama kertolaskun tulo, eli vahvistuskorjattu kiihtyvyyslukema, on määritelty VHDL-koodissa 36-bittiseksi, joista vain 24 bittiä on lopulta luettavissa SPI-isäntälaitteella. Synteesiohjelma poistaa tarpeettomien 12 bitin tuottamiseksi tarvittavan logiikan, joten ylimääräistä pinta-alaa ei kuitenkaan kulu määrittelyn takia.

Suunnitellun Booth-kertojan tarvitsemien digitaalisolujen pinta-ala on yhteensä 0,061 mm^2 käytössä olevalla 0,35 µm CMOS-teknologialla. Se on 10% kaikkien toteutettujen lohkojen pinta-alasta.

Luku 7

Metastabiilisuus ja kelloalueiden välinen synkronointi

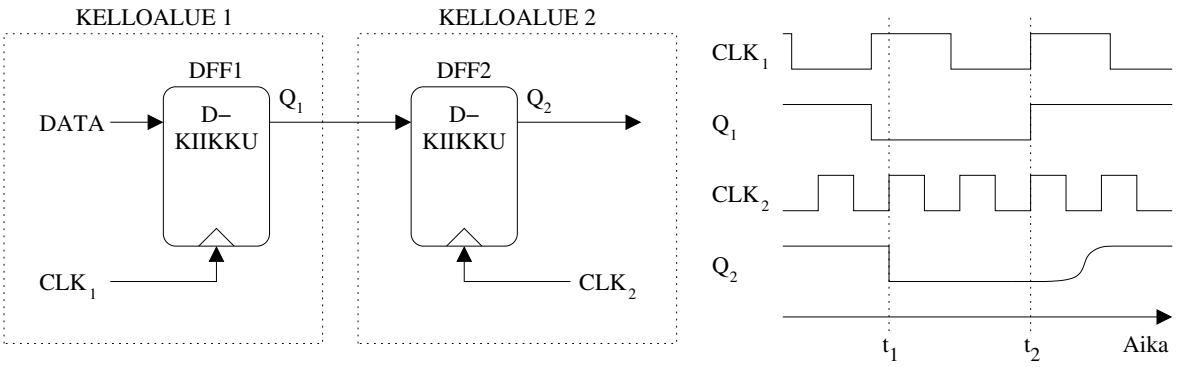
Elektroniikkajärjestelmissä voi olla useita eri piirejä, jotka toimivat eri kellotaajuuksilla, ja joiden täytyy lähettää signaaleja toisilleen. Esimerkiksi tietokoneessa prosessori toimii yhdellä kellotaajuudella ja näytönohjain toisella ja niiden täytyy siirtää tietoa keskenään. Toisaalta järjestelmässä voi olla I/O-lohko, jonka kautta siihen voidaan tuoda herätteitä satunnaisina ajanhetkenä. Esimerkiksi kun käytetään tietokonetta ja painetaan näppäimistöllä yksittäistä kirjainta, tapahtuu napin painaminen asynkronisesti suhteessa prosessorin kellosignaaliin.

Eri kelloilla toimivien piirien rajapinnassa voi tapahtua virheitä, kun tietoa siirretään yhdellä kellolla liipaistusta kiikusta toiseen, eri kellolla liipaistuun kiikkuun. Dataa vastaanottava kiikku joutuu metastabiiliin tilaan, kun sen tulodatalinjan arvo vaihtuu liian lähellä aktiivisen kellonreunan saapumista, jolloin sen lähtölinjan arvon vaihtuminen tapahtuu tyypillistä viivettä myöhemmin. Monibittisen väylän tapauksessa taas lähettävän puolen lähtökiikut eivät usein päivity täysin samaan aikaan, vaan kellotaajuusvaihevirheen (engl. *clock skew*) takia jonkin aikaikkunan sisällä. Jos vastaanottavalla puolella näytteistetään kiikkujen lähdöt tuon aikaikkunan sisällä, osaan vastaanottavista kiikuista näytteistyy vanhan näytteen bitti ja osaan uuden. Lisäksi silloinkin yksi tai useampi vastaanottavista kiikuista voi mennä metastabiiliin tilaan.

Väylällä vastaanotettu näyte, joka sisältää virheellisesti kääntyneitä bittejä, voi aiheuttaa järjestelmässä vakavan virheen. Tilanne voisi olla esimerkiksi se, että auton törmäyksenhavainnointijärjestelmässä siirrettäisiin väylää pitkin kahden komplementtimuodossa olevaa, kahdeksanbittistä kiihtyvyysdataa kiihtyvyysanturilta dataa käsittelevälle mikrokontrollerille. Jos auto olisi pysähtynyt esimerkiksi valoissa, kiihtyvyys olisi nolla, mutta kohinan takia kahdeksanbittisen datan arvo voisi vaihdella lukujen 0_{10} ja -1_{10} välillä. Jos lukema vaihtuisi luvusta 0_{10} lukuun -1_{10}, eli luvusta 00000000_2 lukuun 11111111_2, ja mikrokontrolleri vastaanottaisikin synkronointivirheen takia vain eniten merkitsevän bitin uudesta arvosta ja loput vanhasta, olisi vastaanotettu luku $10000000_2 = -128_{10}$. Luku on lukema-alueen pienin

negatiivinen arvo, joka tarkoittaisi suurta kiihtyvyyttä taaksepäin, mikä voisi laukaista turvatyynyn, vaikka auto on paikoillaan.[1] Metastabiilisuusvirheestä aiheutunut viivästynyt bitin vaihtuminen arvosta 0 arvoon 1 puolestaan voisi aiheuttaa nopeassa prosessorissa virheen, jos kiikun lähdössä oleva logiikkapolku ei ehtisikään laskea uutta arvoa vastaavaa tulosta, ennen kuin se näytteistetään logiikkapolun perässä olevaan kiikkuun. Vaihtoehtoisesti logiikkapolkuja voisi olla useampia, joiden tulokset voitaisiin näytteistää samalla kellonreunalla eri kiikkuihin. Kellotaajuusvaihevirheen takia osaan logiikkapolkujen perässä olevista kiikuista voisi näytteistyä bitillä 0 laskettu tulos ja osaan bitillä 1 laskettu, mikä voisi aiheuttaa järjestelmässä virheitä. Tämän kaltaisten tilanteiden välttämiseksi on syytä analysoida virheiden mahdollisuudet eri kelloilla toimivien piirien rajapinnoissa ja tarvittaessa lisätä piirien väliin synkronoija.

7.1 Metastabiilisuus

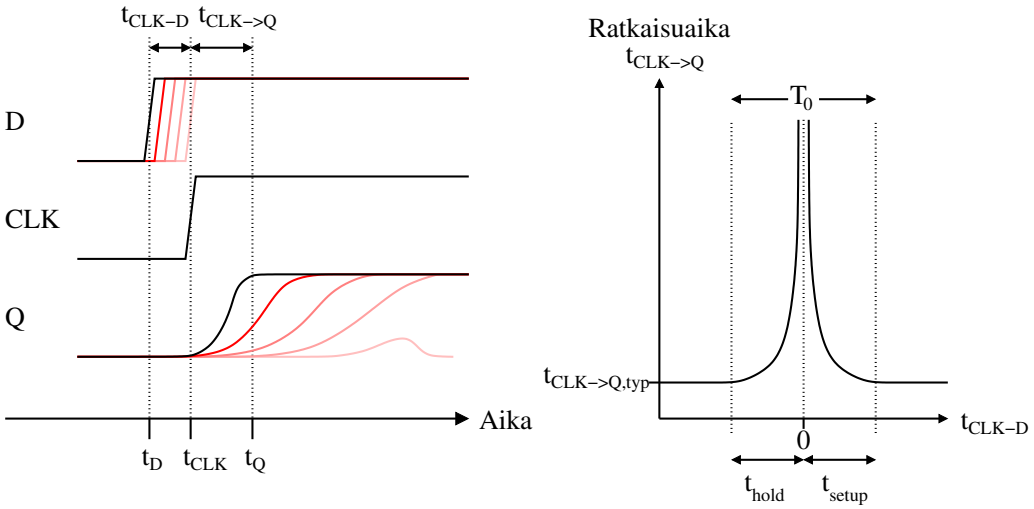


Kuva 7.1: Kahden kelloalueen piiri.

Eri kelloilla toimivien digitaalipiirien välisen tiedonsiirron ongelma on havainnollistettu kuvassa 7.1, jossa vasemmalla on kahden eri kelloilla toimivan digitaalilohkon välinen, yhden bitin siirtämiseen tarkoitettu rajapinta. Vasemmanpuoleiselle kiikulle DFF1 tulee kello CLK_1 ja kiikulle DFF2 kello CLK_2, joilla on eri taajuudet. Kellosignaalit ja kiikkujen lähtösignaalit on esitetty kuvassa oikealla. Tiedonsiirto toimisi virheettömästi luultavasti suurimman osan ajasta. Esimerkiksi kuvassa hetkellä t_1 DFF2 saa aktiivisen kellonreunan ja sille tulevan datalinjan Q_1 arvo on muuttunut hyvissä ajoin sitä ennen, joten kiikun lähtö Q_2 vaihtaa tilaa nopeasti nollaan. Jos järjestelmän annetaan toimia tarpeeksi kauan, oikeanpuoleinen kiikku saa kuitenkin väistämättä jossain vaiheessa aktiivisen kellonreunan lähellä ajankohtaa, jolloin vasemmanpuoleisen kiikun lähtö Q_1 on vaihtamassa tilaa. Näin tapahtuu ajanhetkellä t_2. Tapahtuman seurauksena kiikku DFF2 joutuu metastabiiliin tilaan ja sen lähtö vaihtaa tilaa nollasta ykköseen viiveellä.

[1] Tämänkaltaisen tilanteen välttämiseksi voitaisiin tosin esimerkiksi vertailla kahta peräkkäistä kiihtyvyysnäytettä tai kahdelta eri kiihtyvyysanturilta saatua kiihtyvyysnäytettä, jolloin synkronoimattomasta lukemisesta aiheutuva lukemavirhe voitaisiin havaita.

Kiikuilla on niiden rakenteesta ja transistorien mitoituksesta sekä toimintaolosuhteista, kuten esimerkiksi lämpötilasta ja käyttöjännitteestä, riippuvat asetus- ja pitoaika (engl. *setup time* ja *hold time*), jotka määrittävät, milloin datatulo saa vaihtaa tilaansa suhteessa aktiiviseen kellonreunaan. Asetusaika määrittää, miten paljon ennen aktiivista kellonreunaa datatulon pitää olla asettunut loogiseksi ykköseksi tai nollaksi. Pitoaika määrittää, miten kauan aktiivisen kellonreunan jälkeen datatulon pitää säilyttää looginen arvonsa. Kiikku joutuu metastabiiliin tilaan, jos sen asetus- tai pitoaikaa rikotaan, ja silloin sen datalähtölinjan arvon päivittyminen kestää normaalia kauemmin, eli kiikun ratkaisuaika (engl. *resolving time*) on tavallista suurempi.



Kuva 7.2: D-kiikun lähtödatalinjan Q käyttäytyminen, kun datalinjan jännite nousee lähellä aktiivista kellonreunaa, ja ratkaisuajan $t_{CLK->Q}$ piteneminen aktiivisen kellonreunan saapumisen ja tulevan datalinjan vaihtumisen ajankohtien erotuksen t_{CLK-D} funktiona. [28]

Kuvassa 7.2 vasemmalla on esimerkki siitä, miten kiikun datalähtö Q käyttäytyy, kun datatulo D vaihtaa tilaansa eri hetkinä ennen aktiivisen kellonreunan saapumista [28], eli rikotaan asetusaikaa. Tummimmat signaalit D ja Q esittävät tilannetta, jossa D vaihtuu nopeasti. Vaaleimpien viivojen tapauksessa aktiivinen kellonreuna on tullut D-linjan ollessa jännitealueiden rajalla ja arvo tulkitaan pitkän viiveen jälkeen nollaksi. Tapausten välissä on viiveellä ykköseksi tulkittuja lähtöarvoja.

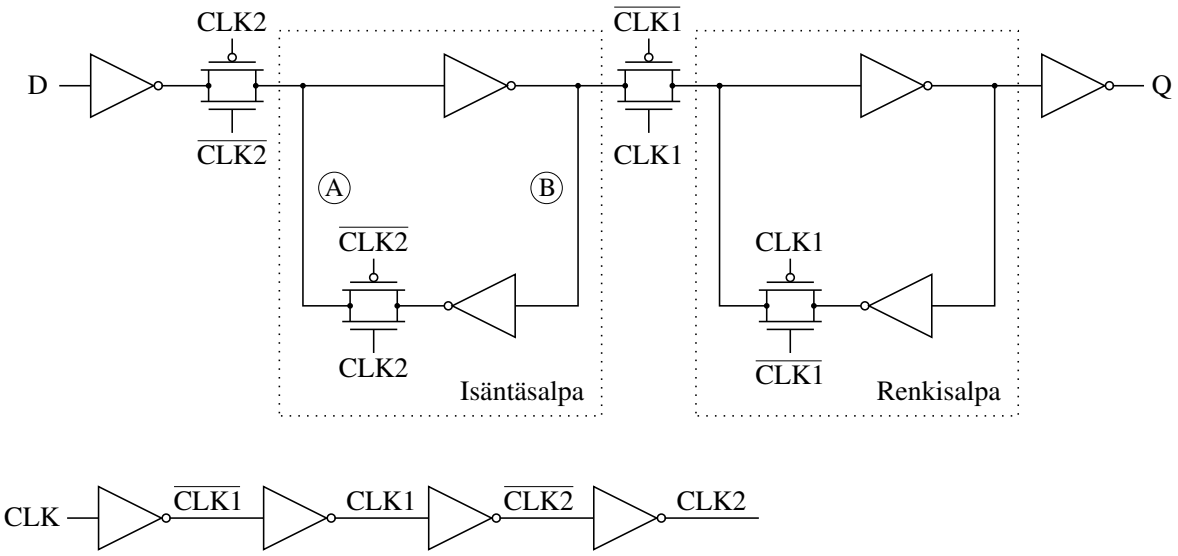
Kuvassa 7.2 oikealla on esitetty D-kiikun ratkaisuaika $t_{CLK->Q}$ aktiivisen kellonreunan ja datalinjan tilanvaihtamisen ajankohtien erotuksen $t_{CLK-D} = t_{CLK} - t_D$ funktiona.[29] Ratkaisuaika $t_{CLK->Q}$ on tyypillistä arvoa $t_{CLK->Q,typ}$ suurempi, jos datalinjan arvo vaihtuu lähellä aktiivista kellonreunaa metastabiilisuusikkunan T_0 sisällä, joka muodostuu asetusajasta, t_{setup}, ja pitoajasta, t_{hold}.

Kun on tiedossa kellojen CLK_1 ja CLK_2 taajuudet ja metastabiilisuusikkunan T_0 pituus, voidaan kaavalla 7.1 [30] laskea aikaväli, kuinka usein keskimäärin vastaanottava kiikku joutuu metastabiiliin tilaan. Erään tuotteistetun D-kiikun asetus- ja pitoajat ovat tyypillisesti t_{setup} = 2 ns ja pitoaika t_{hold} = 0 ns [31], joista saadaan T_0 = 2 ns. Jos sitä käytettäisiin

tiedonsiirrossa kahden eri kelloilla toimivan lohkon rajapinnassa, joiden kellotaajuudet ovat f_{CLK1} = 100 MHz ja f_{CLK2} = 10 MHz, joutuisi vastaanottava kiikku metastabiiliin tilaan ajan $T_{metastab}$ = 500 ns välein.

$$T_{metastab} = \frac{1}{f_{CLK1} \cdot f_{CLK2} \cdot T_0} \tag{7.1}$$

7.2 Metastabiilin tilan ratkeaminen



Kuva 7.3: Inverttereillä ja siirtoporteilla toteutetun D-kiikun lohkokaavio [30].

Kuvassa 7.3 on inverttereillä ja siirtoporteilla (engl. *transmission gate*) toteutetun D-kiikun lohkokaavio [30]. Kiikku joutuu metastabiiliin tilaan, kun siirtoportti datatulon D ja solmun A välissä avautuu ja siirtoportti solmujen A ja B välissä sulkeutuu sellaiseen aikaan, jolloin solmujen A ja B jännitteiden välillä on pieni ero, esimerkiksi yksi mikrovoltti. Mitä lähempänä jännitteet ovat toisiaan, sitä kauemmin kestää, ennen kuin invertterit ajavat toisen jännitteistä maahan ja toisen käyttöjännitteeseen. Jos jännite-ero on metastabiiliin tilaan jouduttaessa alussa V_0 ja metastabiilin tilan katsotaan ratkenneen, kun jännite-ero on V_1, metastabiilisuustilanteen ratkeamiseen kuluva aika t_m saadaan laskettua kaavalla 7.2, missä τ on kiikun ratkaisuaikakavakio (engl. *resolving time constant*). [30] Alussa jännite voisi olla V_0 = 0 V, jolloin saataisiin $t_m = \infty$, mutta käytännössä invertterisilmukka alkaa ajautua pois metastabiilista tilasta siinäkin tapauksessa lopulta kohinan takia.[28]

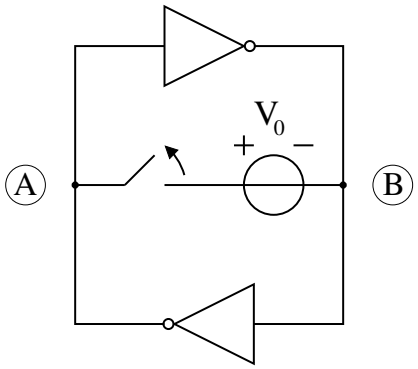
$$t_m = \tau \cdot ln\Big(\frac{V_1}{V_0}\Big), \tag{7.2}$$

Ratkaisuaikakavakion τ arvon voi ratkaista kuvan 7.3 D-kiikun tapauksessa simuloimalla kiikussa olevan isäntäsalvan (engl. *latch*) invertterisilmukkaa. Simuloitava piiri on esitetty kuvassa 7.4. Simulaation alussa kytkin on suljettu, jolloin solmujen A ja B välille asettuu

jännitelähteen V_0 määräämä jännite-ero. Kytkin avataan esimerkiksi yhden nanosekunnin jälkeen, jolloin invertterit pakottavat solmujen jännitteet käyttöjännitteeseen ja maahan, kuten on havainnollistettu kuvassa 7.5 vasemmalla. Kuvassa oikealla on solmujen A ja B jännite-eron kymmenkantainen logaritmi. Kuvan mukaisesti jännite-eron kasvu on eksponentiaalista alussa, jos alussa asetettu jännite-ero V_0 on pieni suhteessa transistorien kynnysjännitteeseen, esimerkiksi muutamia mikrovoltteja tai nanovoltteja. Kuvan 7.5 tapauksessa V_0 on ollut 1 μV. Ratkaisuaikavakio τ saadaan laskettua valitsemalla eksponentiaalisen jännite-eron kasvun alueelta pisteet x ja y ja sijoittamalla näiden pisteiden ajat ja jännitteet kaavaan 7.3. [30]

$$\tau = \frac{t_x - t_y}{ln\frac{V_x}{V_y}} \tag{7.3}$$

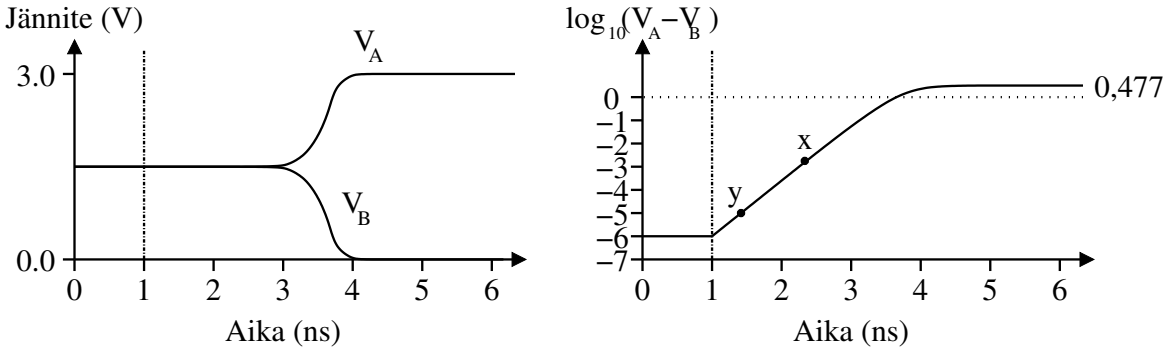
Invertterisilmukkaa simuloitiin käytössä olleella 0,35 μm teknologialla ja ratkaisuaikavakion arvoksi saatiin τ = 53,2 ps. Kuvassa 7.6 on muutamilla teknologioilla mitattuja ja simuloituja ratkaisuaikavakioita [28][30][32] ja simuloitu arvo on lähellä datan perusteella laskettua lineaarista sovitusta, jonka mukaan vakion arvo olisi n. 45 ps. Simuloidessa käytettiin hitaimpien toimintaolosuhteiden lämpötila- ja käyttöjännitearvoja. Simuloidun tuloksen paikkansapitävyyttä tukee se, että kokeellisten tulosten perusteella [30] τ on aina lähellä käytössä olevan prosessin tyypillistä etenemisviivettä, joka on tässä työssä käytössä olevalla prosessilla muutamien kymmenien pikosekuntien kokoluokkaa[33].



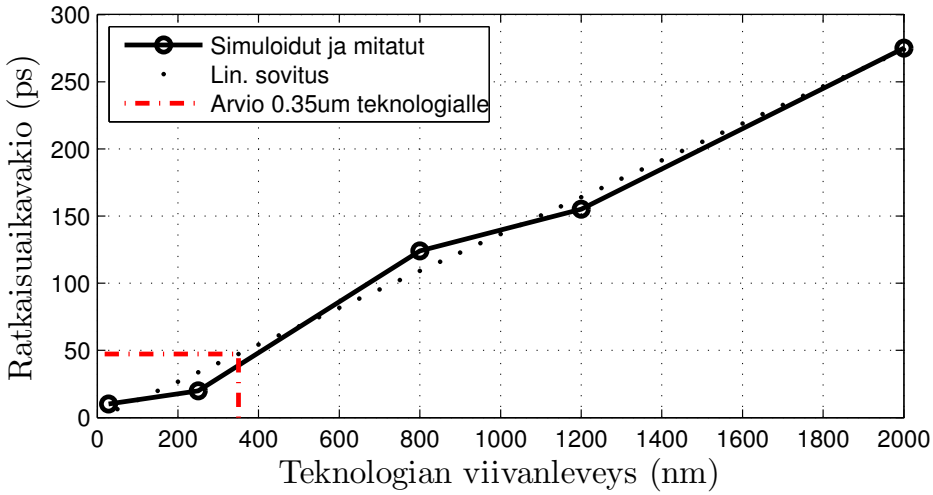
Kuva 7.4: Piiri, jota simuloimalla voidaan selvittää ratkaisuaikavakion τ arvo.[30]

7.3 Synkronointiongelma monibittisen väylän tapauksessa

Rinnakkaisväylillä dataa kelloalueesta toiseen siirrettäessä ongelmana on se, että kelloalueiden rajapinnassa mikä tahansa vastaanottavan puolen kiikuista voi mennä metastabiiliin tilaan. Lisäksi ongelmana on se, että kellotaajuusvaihevirheen takia rinnakkaiset kiikut liipaistaan usein eri aikoihin jonkin aikaikkunan sisällä, joka riippuu kellopuun viiveistä. Kellopuun eriävät viiveet johtuvat esimerkiksi kellolinjojen eri pituisista signaalivedoista mikropiirillä.



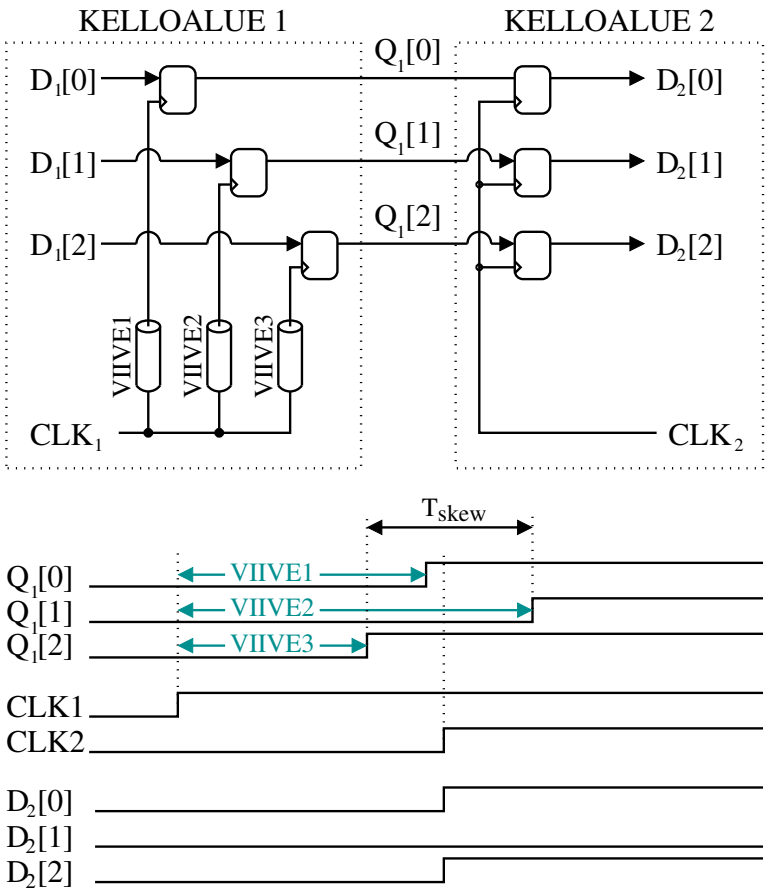
Kuva 7.5: Kuvan 7.4 piirin solmujen A ja B jännitteet, kun kytkin avataan hetkellä t = 1 ns.



Kuva 7.6: Julkaistuja simuloituja ja mitattuja ratkaisuaikavakioita muutamilla teknologioilla [28][30][32] ja tietojen pohjalta laskettu arvio 0,35 µm teknologialle.

Kellotaajuusvaihevirheestä aiheutuva ongelma on havainnollistettu kuvassa 7.7. Tulevan datan D_1 arvo vaihtuu arvosta 000_2 arvoon 111_2. Viiveet 1-3 ovat kellon CLK_1 kellopuun viiveet, joiden takia vasemmanpuoleisessa kelloalueessa kiikkujen lähdöt $Q_1[0]$-$Q_1[2]$ päivittyvät eri aikaan, kuten kuvassa on esitetty alhaalla. Ensimmäisen ja viimeisen lähettävän puolen kiikun lähtöarvon vaihtumisen ajankohtien erotus on merkitty symbolilla T_{skew}. Jos vastaanottavan puolen kellon CLK_2 nouseva reuna osuu aikaikkunan sisälle, jossa lähettävän puolen kiikkujen lähtöarvot päivittyvät, vastaanottavan puolen kiikkuihin näytteistyy väärä arvo. Kuvan esimerkissä lähettävän puolen lähtölinjat $Q_1[0]$ ja $Q_1[2]$ ovat ehtineet vaihtua arvoon 1 arvosta 0, mutta $Q_1[1]$ ei, kun kellon CLK_2 nouseva reuna saapuu. Sen seurauksena vastaanottavalla puolella väylän D_2 arvoksi tuleekin 101_2, vaikka lähettävällä puolella uusi arvo on 111_2. Todellisuudessa myös kellon CLK_2 kellopuussa on omat viiveensä ja vastaanottavan puolen kiikut saavat kellopulssin eri aikoihin, mutta esimerkin yksinkertaistamiseksi vain lähettävän puolen kellotaajuusvaihevirhe on huomioitu.

Kellotaajuusvaihevirheen määrä vaihtelee esimerkiksi käytettävän teknologian ja digitaalisolujen sijoittelun ja siitä aiheutuvien eri pituisten signaalivetojen takia. Automaattisissa



Kuva 7.7: Synkronointiongelma kolmen bitin väylän tapauksessa.

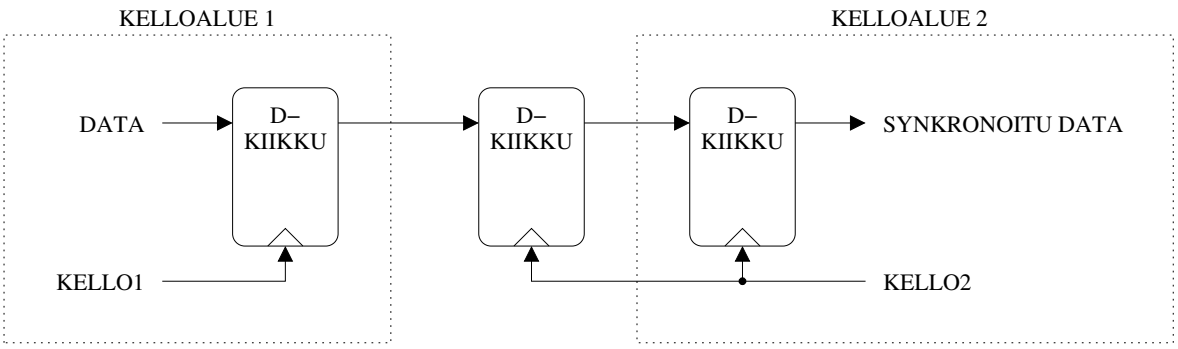
sijoittelu- ja reititysohjelmissa voi usein antaa määrittelyn suurimmalle sallitulle kellotaajuusvaihevirheelle ja ohjelma yrittää sijoittaa ja reitittää solut siten, että määrittely täytetään. Jotkin ohjelmat osaavat myös esimerkiksi sijoittaa viivepuskureita kellopuuhun siten, että vaihevirhe pienenee. Kuvan 7.7 tapauksessa esimerkiksi viiveitä 1 ja 3 pidentämällä $Q_1[0]$:n ja $Q_1[2]$:n arvojen liipaisemista voisi viivästyttää siten, että ne liipaistaan lähempänä linjan $Q_1[1]$ päivittymistä, jolloin T_{skew} pienenisi. Ideaalisesti T_{skew} olisi 0 s, mutta todellisuudessa arvot voivat olla esimerkiksi kymmenien tai satojen pikosekuntien luokkaa. Esimerkiksi 0,18 µm:n teknologialla toteutetun Pentium 4-prosessorin 42 miljoonan transistorin piirillä mitatuksi pahimpien toimintaolosuhteiden kellotaajuusvaihevirheen arvoksi on saatu 16 ps [34].

Vaikka alle nanosekunnin arvot kuulostavat pieniltä, kellotaajuusvaihevirheen aiheuttamien (lukema-)virheiden keskimääräinen aikaväli (MTBF, engl. *mean time between failures*) voi olla suurilla kellotaajuuksilla jopa alle sekunnin, jos väylän dataa luetaan asynkronisesti. Virheiden keskimääräinen aikaväli saadaan laskettua kaavalla 7.4, joka on muuten identtinen kaavan 7.1 kanssa, mutta muuttujat on nimetty vastaamaan tilannetta. Kaavassa f_{sender} on lähettävän puolen kellotaajuus, $f_{receiver}$ vastaanottavan puolen kellotaajuus ja T_{skew} kellotaajuusvaihevirhe lähettävän puolen kiikkujen kellopuussa. Jos esimerkiksi

f_{sender} = $f_{receiver}$ = 1 GHz ja T_{skew} = 10 ps, tulee virheiden keskimääräiseksi aikaväliksi $MTBF_{skew}$ = 100 ns.

$$MTBF_{skew} = \frac{1}{f_{sender} \cdot f_{receiver} \cdot T_{skew}} \tag{7.4}$$

7.4 Kahden kiikun synkronoija



Kuva 7.8: Kahden kiikun synkronoija.

Metastabiilisuustilanteiden harventamiseksi käytetään kelloalueiden välissä synkronointipiiriä. Yhden bitin synkronoinnissa voidaan käyttää esimerkiksi kahden kiikun synkronoijaa. Kahden kiikun synkronoijassa kelloalueiden rajapintaan laitetaan ylimääräinen kiikku, jota liipaistaan vastaanottavan puolen kellolla, kuten kuvassa 7.8 on esitetty. Ensimmäisen vastaanottavan kiikun lähtö voi mennä metastabiiliin tilaan normaalilla todennäköisyydellä. Sillä on kuitenkin toisen kelloalueen kellojakson verran aikaa päätyä jompaan kumpaan loogiseen tilaan ennen kuin sen lähtöarvo liipaistaan seuraavaan kiikkuun. Siksi toisen kiikun lähdöllä on huomattavasti pienempi epätodennäköisyys päätyä metastabiiliin tilaan, jolloin sen lähtöarvon käyttäminen aiheuttaa vähemmän virhetilanteita. Vastaanottavia kiikkuja voidaan lisätä peräkkäin vielä enemmän, jolloin epästabiilisuuden todennäköisyys pienenee eksponentiaalisesti. Haittapuolena on se, että datan siirtymiseen kuluva aika kasvaa synkronoivien kiikkujen aiheuttaman viiveen takia. [35] Lisäksi synkronoinnin takia lisätyt digitaalisolut tai muut piirit esimerkiksi kuluttavat virtaa ja kasvattavat mikropiirin pinta-alaa.

Synkronoijan hyvyyttä kuvaa sen MTBF, jonka kaava on kahden kiikun synkronoijalle

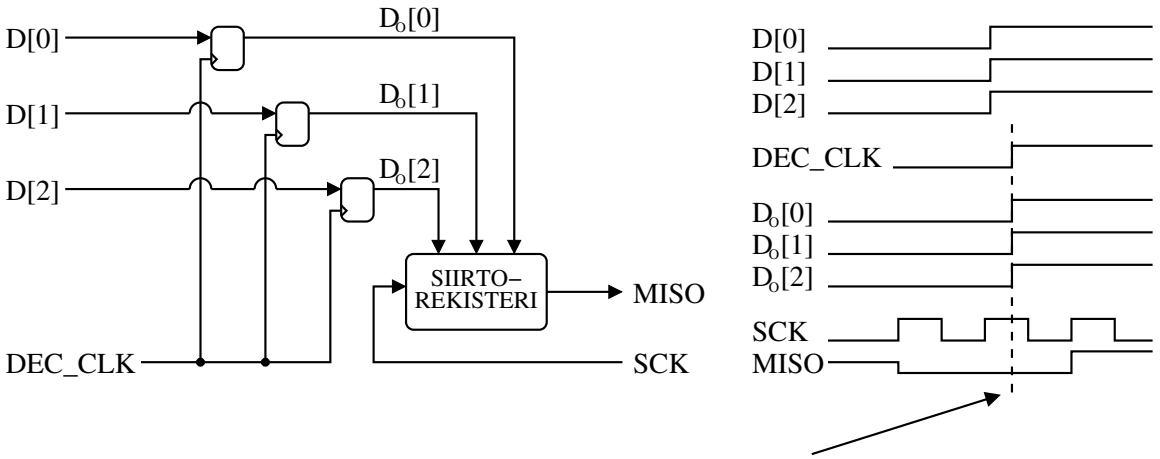
$$MTBF = \frac{1}{f_c f_{data} T_0 e^{-t_r/\tau}}, \tag{7.5}$$

jossa f_c on vastaanottavan puolen kellotaajuus, f_{data} tulevan datan kellotaajuus, T_0 kiikun metastabiilisuusikkuna (kts. kuva 7.2), t_r aika, joka kiikun lähdöllä on selvitä jompaan kumpaan loogiseen arvoon ja τ kiikun ratkaisuaikavakio.[28][30][35] Kahden tai useammankaan kiikun synkronoijaa ei voida tehdä täysin varmasti toimivaksi, vaan se tekee virheen aina jollain todennäköisyydellä, mutta virheiden keskimääräinen aikaväli voidaan tehdä jopa yli satojen miljoonien vuosien suuruiseksi. [28][35] Kaavasta 7.5 nähdään, että MTBF:ää voidaan

kasvattaa esimerkiksi lisäämällä kiikun ratkaisuun käytössä olevaa aikaa t_r, mikä kasvattaa MTBF:ää eksponentiaalisesti.

Useampibittisen väylän datan siirtämistä kelloalueesta toiseen ei yleensä voida toteuttaa sijoittamalla kahden kiikun synkronoijaa jokaisen bitin polulle [30], vaan datasanan siirtämiseen tarvitaan esimerkiksi jokin kättelyprotokolla, jossa kahden kiikun synkronoijaa voidaan kuitenkin hyödyntää.

7.5 Synkronointiongelma työn piirillä



Kuva 7.9: Synkronointiongelma työn piirillä yksinkertaistettuna.

Työn kiihtyvyysanturipiirillä ongelmana on se, että rinnakkaismuodossa olevaa kiihtyvyysdataa luetaan sarjamuodossa SPI-väylän kautta kahdeksan bittiä kerrallaan kolmen SPI-purskeen aikana. Luettu data voi sisältää virheitä, jos data päivittyy lukemisen aikana. Ongelma on havainnollistettu yksinkertaistaen kuvassa 7.9. Kuvassa vasemmalla on kolme DSP-lohkon lähdössä olevaa lähtökiikkua, joita liipaistaan DEC_CLK-kellolla, ja joiden data luetaan SPI-purskeella. Siirtorekisteri kuvaa SPI-väylän MISO-lähdössä olevaa siirtorekisteriä, jota kellotetaan väylän SCK-kellolla, ja jolle kiihtyvyysdata limitetään muistirekisterilohkossa. Siirtorekisterille tulevat kiikkujen lähdöt päivittyvät kesken SPI-tiedonsiirtopurskeen ja SPI-isäntä vastaanottaa MISO-linjaa pitkin bittijonon 001_2, vaikka datan D arvo on ensin 000_2 ja sitten 111_2.

Työn piirin tapauksessa 24-bittinen kiihtyvyysdata luetaan kolmen SPI-purskeen aikana. Data saa päivittyä vielä seitsemän ensimmäisen kellopulssin aikana ensimmäisen purskeen alussa, minkä jälkeen dPAR-bitin pitää kuitenkin olla laskettuna virheettömästä datasta. Sen jälkeen kiihtyvyysdata ei saisi päivittyä, ennen kuin kaikki 24 bittiä on luettu, eli (16-7) + 16 + 16 = 41 SCK-kellopulssin aikana. Jos SPI-väylän kellotaajuus on f_{SCK} = 8 MHz, saadaan ajaksi, jona data ei saa päivittyä, T_{luku} = 41 · T_{SCK} = 5,125 µs.

DSP-lohkon lähdössä näytteistystaajuus on f_{data} = 100 Hz ja oletetaan, että dataa luetaan taajuudella f_c = f_{data} = 100 Hz. Näistä saadaan laskettua virheiden keskimääräinen aikaväli sellaisessa tapauksessa, jossa ei käytetä synkronointia:

$$MTBF_{ei-synk} = \frac{1}{f_c f_{data} T_{luku}} = \frac{1}{100Hz \cdot 100Hz \cdot 5{,}125\mu s} \approx 19{,}5s. \quad (7.6)$$

Pahimmassa tapauksessa SPI-isäntä voisi siis vastaanottaa kolme virheellistä lukemaa per minuutti. Hitaammalla väylän kellotaajuudella datan lukemiseen kuluva aika pitenisi ja virheiden keskimääräinen aikaväli olisi vielä pienempi, esimerkiksi 0,61 s taajuudella f_{SCK} = 250 kHz, jolloin virheellisiä lukemia voitaisiin vastaanottaa 96 per minuutti.

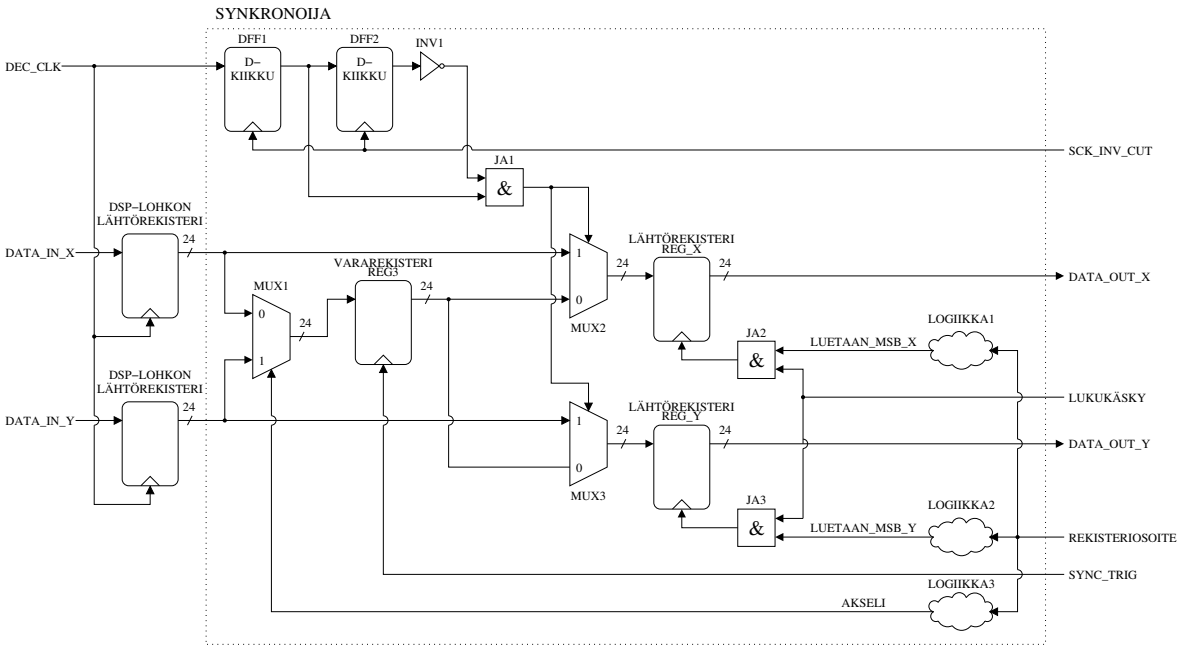
Kiihtyvyysdatan päivittyminen kolmen lukemiseen tarvittavan purskeen aikana voitaisiin estää esimerkiksi tallentamalla 24-bittinen data välirekisteriin, jonka päivittymistä ei sallittaisi, ennen kuin kaikki bitit on saatu luettua SPI-väylän kautta. Data voitaisiin liipaista välirekisteriin esimerkiksi sen SPI-purskeen alussa, jolla luetaan datan eniten merkitsevät kahdeksan bittiä, minkä jälkeen loputkin 16 bittiä voitaisiin lukea välirekisteristä suurellakin viiveellä. Siinä tapauksessa ongelma muuttuisi väylän synkronoimisen ongelmaksi, jollainen on kuvattu kappaleessa 7.3, koska välirekisterin kiikkuihin voisi kuitenkin näytteistyä osaan vanhan ja osaan uuden kiihtyvyyslukeman bittejä, jos välirekisterin liipaiseva kelloreuna tulisi kesken kiihtyvyysdatan päivittymisen. MTBF voidaan laskea siinä tapauksessa seuraavasti. Kiihtyvyysdata päivittyy taajuudella f_{data} = 100 Hz ja 24 rinnakkaista kiihtyvyysdatakiikkua voivat vaihtaa tilaa kellotaajuusvaihevirheen rajoittaman aikaikkunan T_{skew} sisällä. Aikaikkunaksi voidaan määritellä sijoittelu- ja reititysohjelmassa esimerkiksi T_{skew} = 1 ns. Oletetaan, että dataa myös luetaan taajuudella f_c = f_{data} = 100 Hz. Sijoittamalla arvot kaavaan 7.4, saadaan

$$MTBF = \frac{1}{f_c f_{data} T_{skew}} = \frac{1}{100Hz \cdot 100Hz \cdot 1ns} = 100\,000s \approx 27{,}8h. \quad (7.7)$$

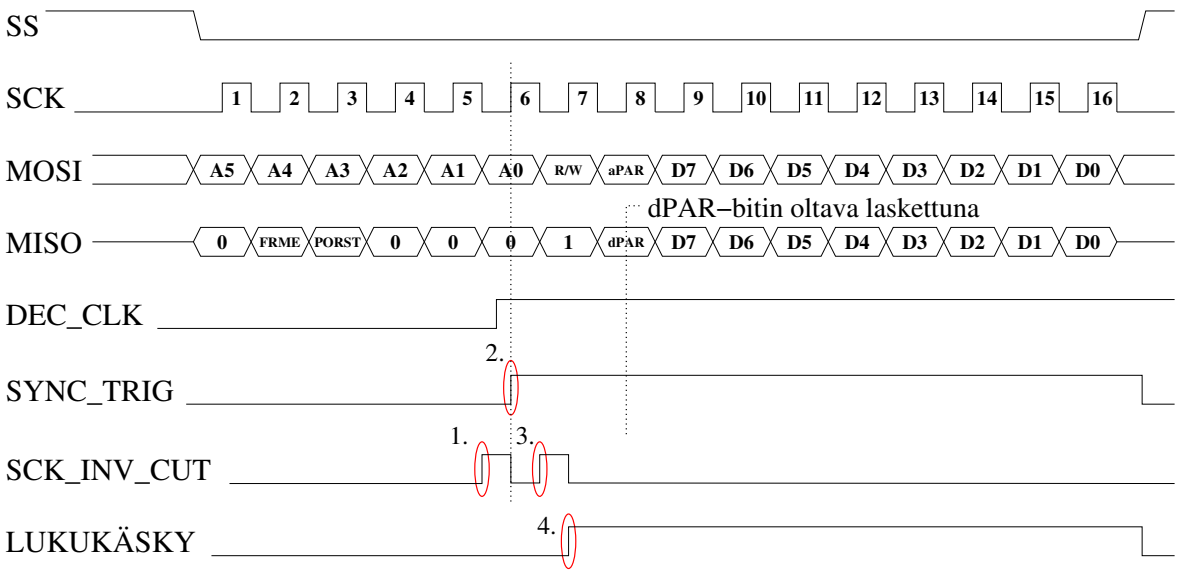
Välirekisterin avulla MTBF kasvaa, mutta voi olla silti edelleen liian pieni joihinkin sovelluksiin. Voi siis olla eduksi käyttää synkronoijaa, jonka avulla MTBF olisi vielä suurempi.

7.6 Toteutettu synkronoija

Toteutetun synkronoijan lohkokaavio on esitetty kuvassa 7.10 ja sen olennaisimmat signaalit kuvassa 7.11. Tulevat kiihtyvyyslukemat DATA_IN_X ja DATA_IN_Y tulevat DSP-lohkolta, jonka lähtörekisterit liipaistaan DEC_CLK-kellon nousevalla reunalla. Synkronoijalohkolta lähtevät, synkronoidut kiihtyvyyslukemat DATA_OUT_X ja DATA_OUT_Y ohjataan muistirekisterilohkolle, joka ohjaa ne edelleen SPI-lohkolle, kun kiihtyvyyslukuja kysytään SPI-väylän kautta. MISO-linjaa pitkin lähetettävän dPAR-bitin, joka lasketaan luettavasta datasta, pitää olla laskettuna jo kahdeksannen SCK-kellopulssin nousevan reunan koh-



Kuva 7.10: Toteutetun synkronoijalohkon yksinkertaistettu lohkokaavio.



1. Näytteistetään CIC–suodattimen lähtökellosignaali DEC_CLK.
2. CIC–suodattimelta tuleva data näytteistetään vararekisteriin REG3.
3. Näytteistetään uudelleen DEC_CLK.
4. Päivitetään synkronoijan DATA_OUT_X–lähtö, jos luetaan x–akselin eniten merkitsevä tavu, tai DATA_OUT_Y–lähtö, jos luetaan y–akselin eniten merkitsevä tavu

Kuva 7.11: Synkronointiin liittyvät olennaiset signaalit piirillä.

dalla, joten synkronoijalohkon lähdössä täytyy olla synkronointivirheetön kiihtyvyysdata jo ennen sitä.

Toteutettu synkronoija toimii siten, että kiihtyvyysdata tuodaan DEC_CLK-kelloalueelta SCK-kelloalueelle liipaisemalla se vararekisteriin REG3 signaalin SYNC_TRIG nousevalla reunalla eli SCK-kellon kuudennella nousevalla reunalla (ajankohta 2. kuvassa 7.11). Synkronoijan lähtörekisteri REG_X tai REG_Y liipaistaan kuitenkin vasta yksi SCK-kellonjakso myöhemmin signaalin LUKUKÄSKY nousevalla reunalla (ajankohta 4. kuvassa 7.11). On epätodennäköistä, että rekisterissä REG3 olisi virheellistä dataa, joten lähtörekistereihin limitetään tyypillisesti rekisterin REG3 lähtö. Jos tuon kellonjakson aikana kuitenkin havaitaan, että REG3 on liipaistu DSP-lohkon lähtödatan päivittyessä, lähtörekistereille limitetäänkin kiihtyvyysdata suoraan DSP-lohkon lähtörekisteristä. Tiedetään, että DSP-lohkon lähtörekisterit eivät ole siinä tapauksessa päivittymässä signaalin LUKUKÄSKY nousevalla reunalla, koska ne ovat päivittyneet noin yhden SCK-kellonjakson verran eli 0,125 ... 8 µs aiemmin[2] ja DSP-lohkon lähtödata päivittyy 10000 µs välein eli taajuudella 100 Hz.

Vararekisterille REG3 limitetään x- tai y-akselin data sen mukaan, kumman akselin dataa luetaan. Synkronoijan lähtörekisteri REG_X tai REG_Y liipaistaan vain, jos luetaan kyseisen akselin eniten merkitsevät kahdeksan bittiä. Näin varmistetaan se, että kiihtyvyysdatan vähiten merkitsevät 16 bittiä voidaan lukea ilman, että data synkronoijan lähtörekisterissä päivittyy kesken lukemisen. Vararekisterin REG3 lähtödata voi olla virheellistä, jos rekisteri REG3 liipaistaan DSP-lohkon lähtörekisterien päivittyessä eli jos signaalin SYNC_TRIG nouseva reuna osuu lähelle DEC_CLK-kellon nousevaa reunaa. Tämä tilanne havaitaan näytteistämällä DEC_CLK-kellon arvo ennen signaalin SYNC_TRIG nousevaa reunaa ja sen jälkeen (ajankohdat 1. ja 3. kuvassa 7.11) kuvan 7.10 yläreunassa oleviin D-kiikkuihin DFF1 ja DFF2 signaalin SCK_INV_CUT nousevilla reunoilla. JA1-portin lähtö on 1, mikäli ensimmäinen DEC_CLK:in näytteistetty arvo on 0 ja jälkimmäinen 1, mistä tiedetään, että DEC_CLK-kellon nouseva reuna tullut lähellä SYNC_TRIG:in nousevaa reunaa. Silloin limittäjät MUX2 ja MUX3 ohjaavat synkronoijan lähtörekistereille kiihtyvyysdatan DSP-lohkon lähtörekistereiltä.

Suunnitellun synkronoijalohkon lähtödataan voi tulla virheitä, jos D-kiikuissa DFF1 ja DFF2 tapahtuu metastabiilisuustilanne ja limittäjille MUX2 ja MUX3 menevä valintasignaali päivittyy vasta juuri samaan aikaan, kun synkronoijan lähtörekisterit liipaistaan. Kiikut DFF1 ja DFF2 voivat mennä metastabiiliin tilaan, jos DEC_CLK-kellon nouseva reuna tulee samaan aikaan kiikut liipaisevan SCK_INV_CUT-kellon nousevalla reunalla. Silloin limittäjien MUX2 ja MUX3 valintasignaalin virheellisyys voisi aiheuttaa sen, että osa synkronoijalohkon lähtörekistereiden tuloista tulee limittäjän tulosta 0 ja osa tulosta 1. Limittäjien valintasignaalin täytyy siis päätyä metastabiilista tilasta toiseen loogiseen arvoon ennen kuin lähtörekisterit saavat signaalin LUKUKÄSKY nousevan reunan. Koska DFF2:n tulo tulee DFF1:n lähdöstä, ei DFF2 päädy metastabiiliin tilaan, ellei DFF1:n lähtö ole viivästynyt me-

[2]Oletetaan, että SPI-kellotaajuus on välillä f_{SCK} = 125 kHz ... 8 MHz

tastabiilisuuden takia. Näin ollen virheellisten lähtölukemien keskimääräisen aikavälin alarajan määrittää kiikkujen DFF1 ja DFF2 muodostaman kahden kiikun synkronoijan MTBF.

Jos DFF1 päätyy metastabiiliin tilaan, sillä on yhden SCK-kellojakson verran aikaa päätyä ratkaisuun, ennen kuin sen lähtöarvo liipaistaan DFF2:een. Aika on pienin suurimmalla vaaditulla SCK-kellotaajuudella f_{SCK}, jolla saadaan t_r = 1/f_{SCK} = 125 ns. D-kiikun metastabiilisuusikkunan arvona voidaan käyttää asetus- ja pitoaikojen hitaimpien toimintaolosuhteiden arvojen summaa, jotka saadaan digitaalisolukirjaston ajoitustiedoista, jolloin saadaan T_0 = 0,270 ns. Kappaleen 7.2 simulaatioiden perusteella ratkaisuaikavakion arvo on τ = 53,2 ps. Tulevan datan kellotaajuus on DSP-lohkon DEC_CLK-kellon taajuus f_{data} = 100 Hz ja vastaanottavan puolen kellotaajuutena voidaan käyttää samaa kellotaajuutta, eli f_c = 100 Hz. Sijoittamalla arvot kaavaan 7.5 saadaan virheiden keskimääräiseksi aikaväliksi

$$MTBF = \frac{1}{f_c f_{data} T_0 e^{-t_r/\tau}} \approx \frac{1}{2{,}7 \cdot 10^{-6} \cdot e^{-2349{,}6}} s \approx 9{,}7 \cdot 10^{1025} s. \tag{7.8}$$

MTBF:n suuri arvo johtuu siitä, että kiikkujen käytössä oleva ratkaisuaika t_r = 125 ns on erittäin pitkä suhteessa ratkaisuaikavakioon τ = 53,2 ps, mistä johtuen kaavan nimittäjässä oleva Neperin luvun eksponentti on erittäin suuri negatiivinen luku. Jos SPI-kellotaajuus olisi esimerkiksi 1 GHz ja vastaavasti t_r = 1 ns, olisi MTBF enää vain 1,36 $\cdot$ 10^{15} s.

Luku 8

Mittaukset

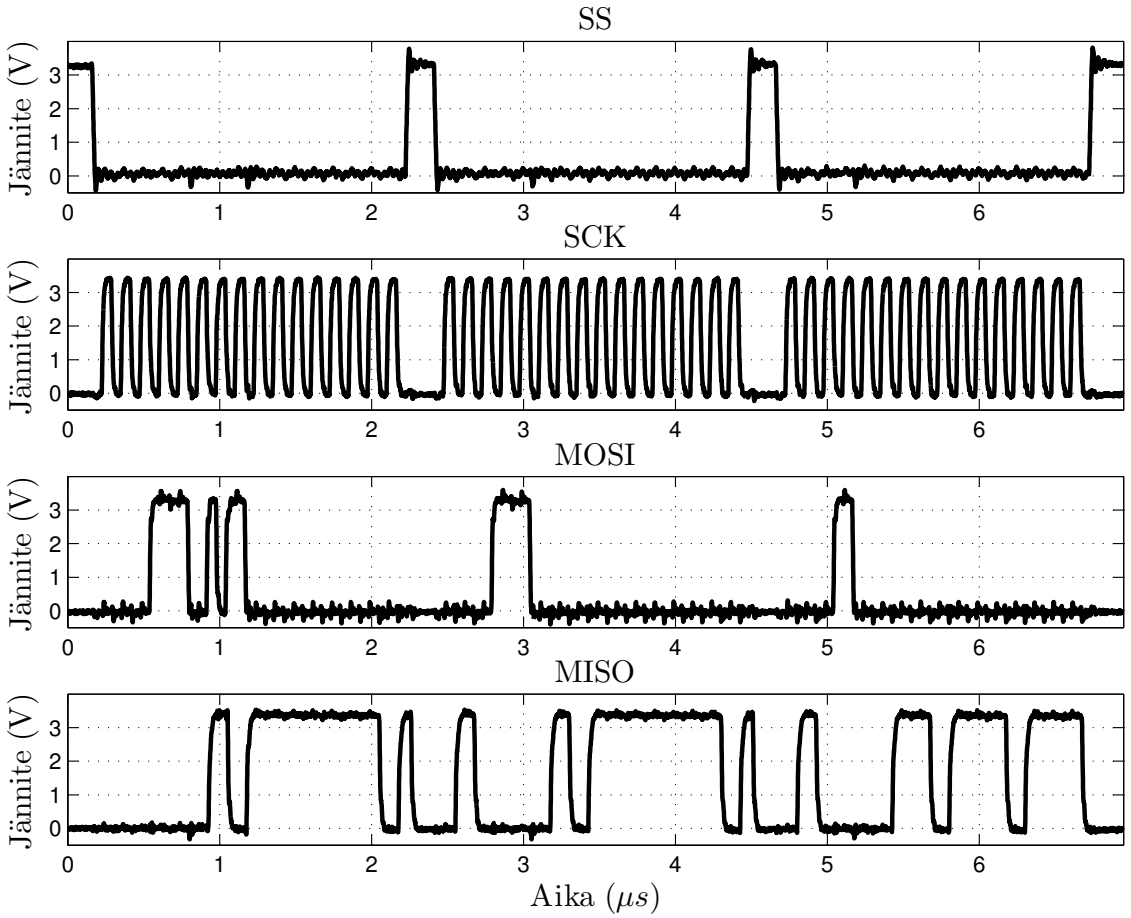
Työssä toteutettujen lohkojen mittauksissa testattiin seuraavat toiminnot ja ominaisuudet:

- SPI-väylän toiminta eri kellotaajuuksilla
- säätöbittirekisterien kirjoittaminen ja lukeminen
- kiihtyvyyden lukeminen
- CIC-suodattimen vahvistusvaste
- CIC-suodattimen aiheuttama signaali-kohinasuhteen muutos
- virrankulutuksen kasvu kiihtyvyyttä luettaessa
- kaikkien toteutettujen lohkojen dynaaminen virrankulutus

Anturipiirille ei toteutettu mahdollisuutta syöttää 1-bittistä dataa ulkoisesta signaaligeneraattorista tämän työn digitaalilohkoille, joten CIC-suodattimen vahvistusvastetta mitattaessa syötettiin sinimuotoista jännitettä AD-muuntimelle, joka edelleen tuotti 1-bittisen datan CIC-suodattimelle. Tässä työssä toteutettujen digitaalilohkojen virrankulutusta ei voitu mitata suoraan, koska samasta teholähteestä menee virta myös kahdelle muulle suurempia kellotaajuuksia käyttävälle digitaalilohkolle, joiden virrankulutusta ei saada katkaistua. Digitaalilohkojen virta menee myös kellogeneraattoripiirilohkolle, kahdeksalle juotospistesolulle (engl. *pad cell*) ja 22:lle erilliselle digitaalisolulle. Siksi dynaaminen virrankulutus piti selvittää mittaamalla kokonaisvirrankulutus ensin, kun työssä toteutetun digitaalipiirin kellonjako on päällä, ja sen jälkeen uudelleen, kun kellonjako on kytketty pois päältä. Lohkon kellonjaon voi katkaista PDOW-bitin avulla. Työn lohkojen vuotovirtaa ei saada mitattua, koska niiden osuutta kokonaisvuotovirrasta ei voida määritellä käytössä olevilla mittausmenetelmillä. Synkronoijalle ei tehty testauspiiriä, joten sen suorituskykyä ei voitu testata mittauksin.

8.1 SPI-väylä

SPI-väylää käytettiin säätöbittien kirjoittamiseen ja lukemiseen ja se toimi moitteettomasti väylän SCK-kellon taajuuksilla 250 kHz, 500 kHz, 1 MHz, 2 MHz, 4 MHz ja 8 MHz. Kuvassa 8.1 on esimerkki oskilloskoopilla mitatuista SPI-väylän SS-, SCK-, MOSI- ja MISO-linjajännitteistä, kun piiriltä luetaan x-akselin kiihtyvyyden kaikki kolme 8-bittistä rekisteriä kolmella tiedonsiirtopurskeella. Kuvassa SCK-kellon taajuus on 8 MHz, eli yksi kellojakso on 125 ns.



Kuva 8.1: SPI-väylän linjajännitteet SPI-purskeen aikana oskilloskoopilla mitattuna.

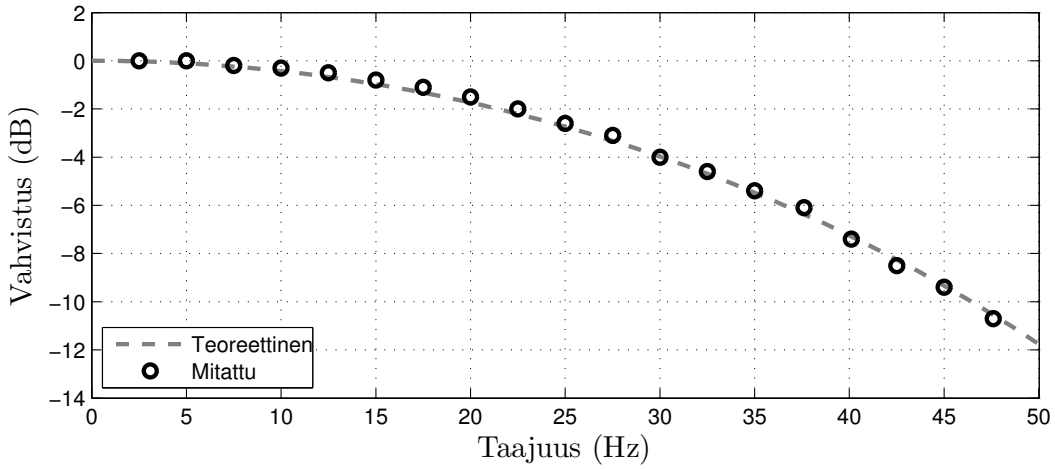
SPI-väylän kautta tapahtuvan kiihtyvyyden lukemisen vaikutus virrankulutukseen selvitettiin mittaamalla kokonaisvirrankulutus ensin ilman SPI-dataliikennettä ja sen jälkeen kiihtyvyyden lukemisen aikana. Mittauksessa sekä x- että y-akselin kaikki kolme tavua luettiin nopeudella 100 SPS eli 10 millisekunnin välein. SPI-väylän SCK-kellon taajuutena käytettiin arvoa 8 MHz. Kiihtyvyyksien lukeminen kasvattaa virrankulutusta mittausten perusteella 0,32 µA. Pelkän toteutetun digitaalipiirin virtasimulaatioissa virta kasvaa kiihtyvyyden lukemisen takia 0,18 µA. Simuloidun ja mitatun virrankulutuksen ero selittyy osittain prosessivariaatiolla ja sillä, että mittauksissa osa virrasta menee myös SPI-väylän neljälle juotospistesolulle. Erityisesti virtaa kasvattaa lähtevän datan eli MISO-linjan juotospistesolu,

joka ajaa 32 pikofaradin kuormaa piirilevyllä. Lisäksi virrankulutusta kasvattaa mikropiirillä jokaisella SPI-linjalla olevat ylös- tai alasvetovastukset, jotka ovat polypiikerroksella toteutettuja vastuksia, joiden tyypillinen suuruus on 200 $k\Omega$. Yksittäisen ylösvetovastuksen läpi menevä tyypillinen virta on 16,5 µA, kun SPI-isäntälaite ajaa linjaa alas, ja alasvetovastuksen läpi menee yhtä suuri virta, kun isäntälaite ajaa linjaa ylös. Juotospistesolujen virrankulutukseksi saatiin simuloitua Spectre-analogiapiirisimulaattorilla tällä lukutahdilla 0,2 µA, eli digitaalilohkojen ja juotospistesolujen simuloitujen virtojen summa on 0,38 µA, joka on lähellä mitattua arvoa.

8.2 CIC-suodattimen vahvistusvaste

CIC-suodattimen mitattu vahvistusvaste on esitetty kuvassa 8.2. Mittaustulokset vastaavat hyvin teoreettista vahvistusvastetta, joka on laskettu kaavalla 3.9. Vahvistusvaste mitattiin asettamalla AD-muuntimen herätesignaaliksi sinimuotoinen jännite ja vaihtamalla sen taajuutta välillä 2,5 ... 47,5 Hz. CIC-suodattimen lähtödataa mitattiin 20 sekunnin ajan jokaista mittaustaajuutta kohden. Jokaisesta mitatusta lähtödatavektorista laskettiin spektrit, joiden signaalipiikeistä signaaliteho laskettiin.

CIC-suodattimelle tulevan signaalin teho riippui mittauksessa myös AD-muuntimesta, mikä olisi voinut vääristää tuloksia. Siksi myös AD-muuntimen aiheuttama vaikutus signaalitehoon mitattiin taajuuden funktiona. AD-muunnin ei aiheuttanut merkittävää muutosta signaalitehoon taajuusvälillä 2,5 ... 47,5 Hz. CIC-suodattimelle tuleva signaaliteho oli vahvistusvastemittauksessa siis yhtä suuri jokaisella mittaustaajuudella ja AD-muuntimen käyttö ei vääristänyt tuloksia.



Kuva 8.2: CIC-suodattimen (N = 3, R = 562, M = 1, $f_{s,in}$ = 56 220 Hz) teoreettinen vahvistusvaste ja mitattu vahvistusvaste.

8.3 Signaali-kohinasuhde

Mittausten perusteella kiihtyvyysdatan SNR paranee CIC-suodattimella 4,6 dB, jos signaalitaajuus on alle 10 Hz, koska signaaliteho ei vaimene silloin merkittävästi (kts. kuva 8.2). Jos signaalitaajuus on lähellä taajuutta 50 Hz, SNR pienenee 7 dB, koska suodatin vaimentaa silloin signaalitehoakin. Mitattu SNR oli delta-sigma-AD-muuntimen lähdössä 29,6 dB kaistalla 0-50 Hz ja CIC-suodattimen lähdössä 34,2 dB kaistalla 0 ... 50 Hz. Delta-sigmamuuntimen lähdön näytteistystaajuus oli 56220 Hz ja CIC-suodattimen lähdössä 100 Hz, koska taajuudenpudotussuhteena käytettiin arvoa R = 562. Mittauksissa saadut SNR-arvot ovat pieniä, koska piirin delta-sigmamuuntimelle voidaan syöttää ulkoisesta lähteestä vain pieniamplitudista signaalia ja herätesignaalin amplitudi oli vain 9,5 mV.

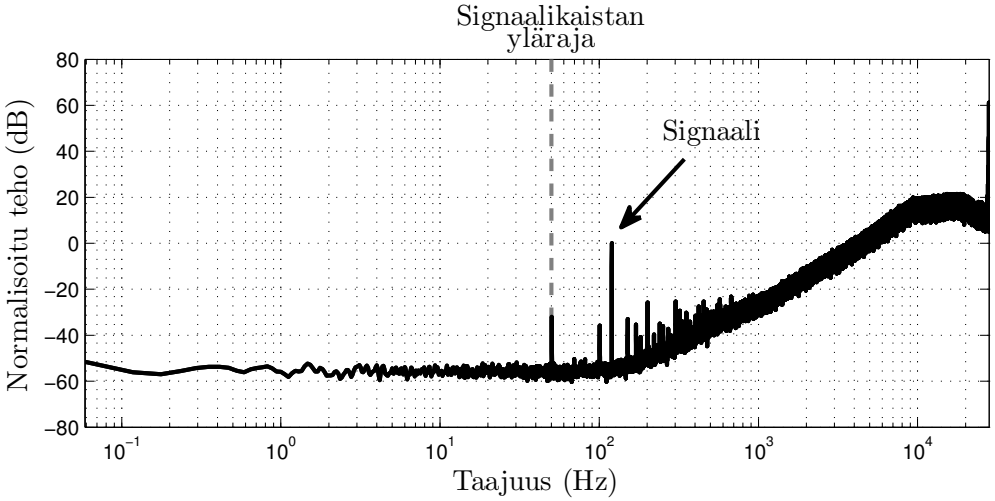
Delta-sigmamuuntimen SNR on laskettu kuvan 8.3 spektristä. Signaalitaajuudeksi valittiin 120 Hz, jotta signaalipiikki ja harmoniset särökomponentit eivät osu signaalikaistalle, jolta kohinan teho laskettiin. Signaaligeneraattorista aiheutuvaa, signaalikaistan ylärajalla olevaa 50 Hz:n häiriösignaalipiikkiä ei laskettu mukaan kohinatehoon.

CIC-suodattimen SNR laskettiin kuvan 8.4 spektreistä. Signaaliteho laskettiin kuvan 8.4(a) spektristä, jossa signaalitaajuus on 5 Hz. Signaaligeneraattori tuotti häiriöpiikkejä spektriin, joten kohinan tehoa ei laskettu samasta spektristä, vaan kuvan 8.4(b) spektristä, jota mitattaessa herätteenä oli tasajännitesignaali.

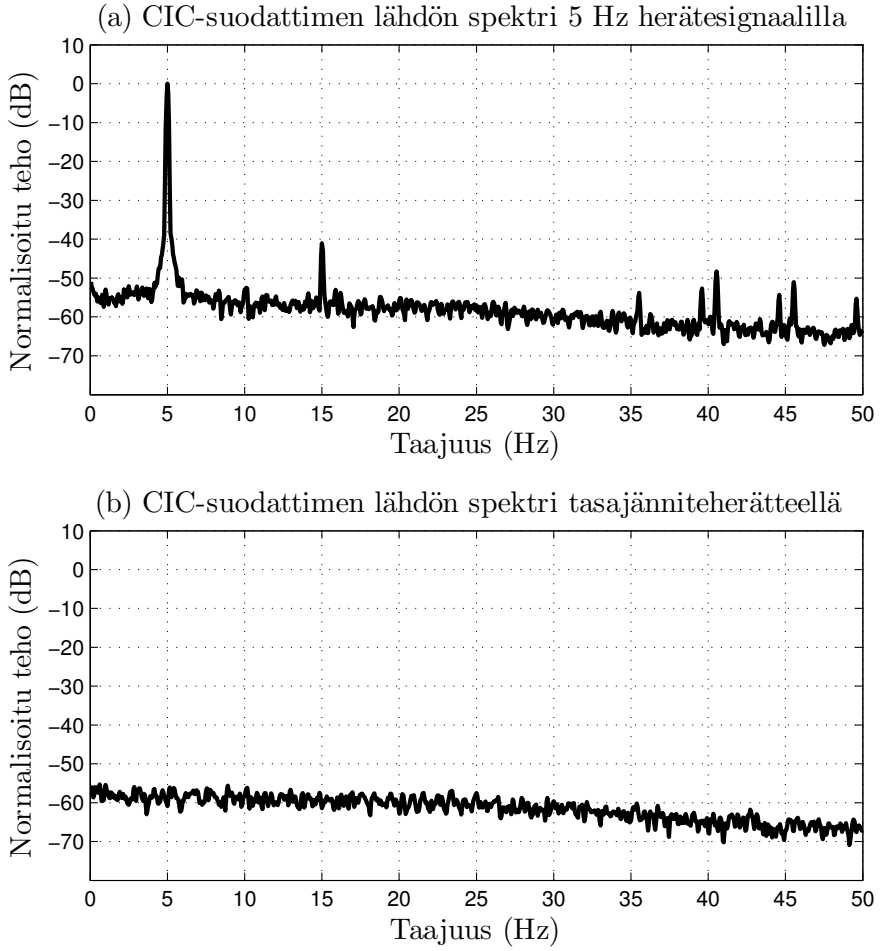
Pienillä signaalitaajuuksilla ilmenevä SNR:n paraneminen selittyy sillä, että CIC-suodatin vaimentaa kohinaa jo kaistalla 0 ... 50 Hz ja merkittävästi sitä korkeammilla taajuuksilla. Kohinan vaimeneminen on havainnollistettu liitteessä B kuvassa B.1. Kuvassa B.1(a) on SNR-mittauksessa käytetyn CIC-suodattimen Matlabissa kaavalla 3.9 laskettu vahvistusvaste. Kuvassa B.1(b) on piiriltä mitattu AD-muuntimen lähdön spektri kohinaherätteellä. Suodattamaton kohina on esitetty harmaalla viivalla ja kuvan B.1(a) siirtofunktiolla suodatettu kohina mustalla viivalla. Kohinatehon määrä suodattamattomassa spektrissä kaistalla 0 ... 50 Hz on $P_{n,50Hz}$ = 3,72·10^{-9} LSB^2. Kohinatehon määrä suodatetussa spektrissä koko kaistalla 0 ... 56220 Hz on $P_{n,suodatettu}$ = 2,05·10^{-9} LSB^2, josta taajuuden 50 Hz yläpuolella oleva kohina laskostuu kaistalle 0 ... 50 Hz. Kohinatehon muutos on

$$P_{diff} = 10dB \cdot \frac{P_{n,suodatettu}}{P_{n,50Hz}} \approx 10dB \cdot \frac{2{,}05 \cdot 10^{-9} LSB^2}{3{,}72 \cdot 10^{-9} LSB^2} = -2{,}6\ dB \qquad (8.1)$$

Koska kohinateho laskee 2,6 dB, SNR paranee 2,6 dB, jos signaalin teho ei vaimene merkittävästi, eli tässä tapauksessa silloin kun signaalitaajuus on alle 10 Hz (kts. kuva 8.2). Laskettu arvio on lähellä mitattua arvoa, joka on 4,6 dB.



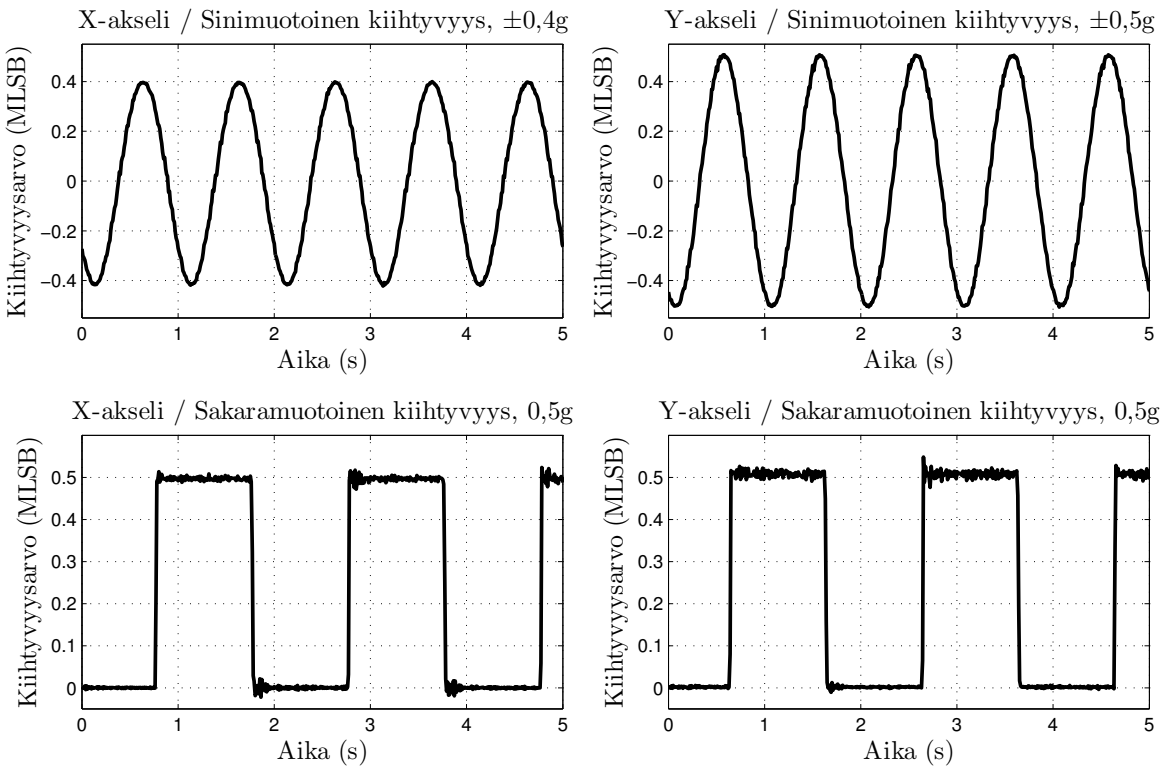
Kuva 8.3: Piirillä olevan AD-muuntimen lähtösignaalin mitattu spektri, kun signaalitaajuus on 120 Hz.



Kuva 8.4: CIC-suodattimen lähtösignaalin mitattu spektri, kun herätesignaaleina on 5 Hz sinimuotoinen jännite (ylempi kuva) ja tasajännite (alempi kuva).

8.4 Tasapoikkeaman ja vahvistuksen korjaus

Tasapoikkeamat korjaavien summainten ja vahvistukset korjaavan Booth-kertojan toiminta testattiin korjaamalla muutamien anturipiirien lähtöarvot. Kuvassa 8.5 on yhdeltä piiriltä mitatut kiihtyvyydet, jotka on korjattu näillä lohkoilla. Tasapoikkeamankorjausarvot on asetettu siten, että lähtöarvo on 0 LSB, kun kiihtyvyys on 0 g. Vahvistuskorjaimet on asetettu siten, että vahvistus on 1 000 000 LSB/g. Piirilevy on asetettu Acutronic AC1120S-pyörittimeen ja pyörimisnopeutta on ohjattu vaihtojännitelähteellä siten, että on saatu aikaiseksi sinimuotoinen ja sakaramuotoinen kiihtyvyys.



Kuva 8.5: Mitatut, SPI-väylän kautta luetut x-akselin ja y-akselin kiihtyvyydet sinimuotoisilla ja sakaramuotoisilla kiihtyvyyksillä.

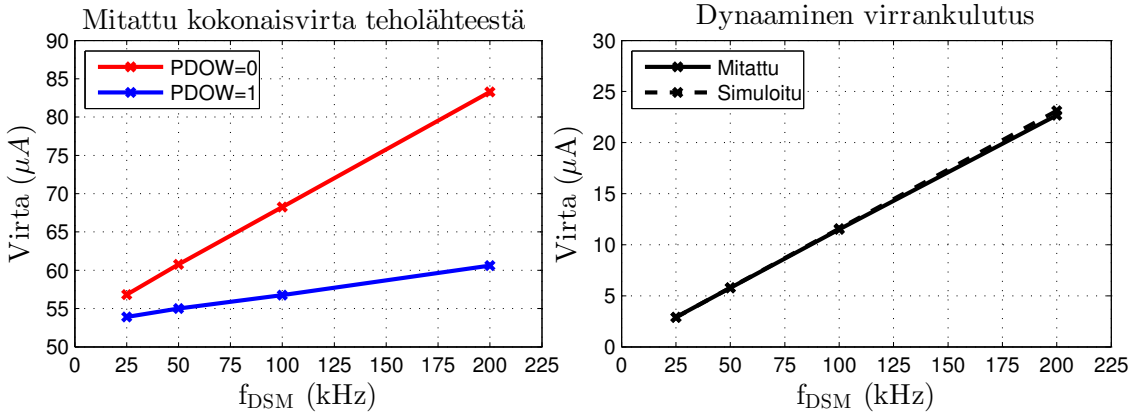
8.5 Dynaamisen virrankulutuksen selvittäminen

Tämän työn digitaalilohkojen virrankulutusta ei voida mitata suoraan yksittäisellä virtamittauksella, koska lohkoihin kytketystä käyttöjännitenastasta menee virta myös osalle piirin muita lohkoja ja osalle juotospistesoluja. Siksi dynaaminen virrankulutus selvitettiin mittaamalla teholähteestä otettu virta, kun DSP-lohkolle menevä kello DSM_CLK on päällä ja sen jälkeen, kun se on katkaistu. Kellonjako voidaan katkaista PDOW-säätöbitillä, joka menee muistirekisterilohkolta JA-portille, joka on kuvassa 2.1 vasemmassa alakulmassa. PDOW-bitin kytkeminen vaikuttaa vain tämän työn digitaalilohkojen virrankulutukseen. Mittauk-

sen aikana ei luettu kiihtyvyyksiä SPI-väylän kautta, ja työn lohkoille ei mene DSM_CLK-kellon ja SPI-väylän SCK-kellon lisäksi muita kellosignaaleja, joten lohkot eivät kuluttaneet dynaamista virtaa, kun kellonjako katkaistiin, vaan ainoastaan vuotovirtaa.

Kuvassa 8.6 vasemmalla on mitattu, teholähteestä otettu kokonaisvirta CIC-suodattimien tulojen kellotaajuuden f_{DSM} funktiona. Sininen suora on mitattu, kun kellonjako oli pysäytetty, ja punainen suora silloin, kun kellonjako oli päällä. Näiden suorien erotus on työssä toteutetun digitaalipiirin dynaaminen virrankulutus, joka on esitetty oikeanpuoleisessa kuvassa yhtenäisellä viivalla. Katkoviivalla on esitetty toteutettujen digitaalilohkojen simuloitu dynaaminen virrankulutus tyypillisessä prosessikulmassa. Mitattujen ja simuloitujen arvojen suurin ero on 200 kHz kellotaajuudella saaduissa virta-arvoissa, jonka kohdalla erotus on 0,1 µA. Se on alle 1% virran arvosta, eli simuloidut ja mitatut luvut täsmäävät hyvin toisiaan.

Kaikkien toteutettujen lohkojen tyypillinen vuotovirta huoneenlämmössä on simulaatioiden mukaan yhteensä 0,179 µA, kun taas dynaaminen virrankulutus on 10,53 µA, kun CIC-suodattimille tulee yksibittistä dataa kellotaajuudella 100 kHz. Simulaatioiden perusteella dynaaminen virrankulutus on siis n. 98% toteutetun digitaalipiirin kokonaisvirrankulutuksesta huoneenlämmössä. Simuloidut ja mitatut virrat on esitetty taulukossa 8.1.



Kuva 8.6: Vasemmanpuoleisessa kuvassa on teholähteestä otettu kokonaisvirta, kun kellonjaon katkaiseva PDOW-bitti on 1 ja 0. Oikeanpuoleisessa kuvassa on näiden virtojen erotus sekä toteutettujen digitaalilohkojen simuloitu virrankulutus.

Taulukko 8.1: Simuloidut ja mitatut virrankulutukset.

	Simuloitu (µA)	**Mitattu (µA)**
Vuotovirta	0,18	-
Dynaaminen virta[1]	11,58	11,50
Dynaamisen virran kasvu kiihtyvyyksiä luettaessa[2]	0,18	0,32
Yhteensä	**11,94**	**11,82**

[1]DSM_CLK-kellotaajuus 100 kHz, DEC_CLK-kellotaajuus 100 Hz.

[2]Luetaan 100 kertaa sekunnissa kuusi 8-bittistä rekisteriä SPI-väylän kautta.

Luku 9

Käytetyt menetelmät

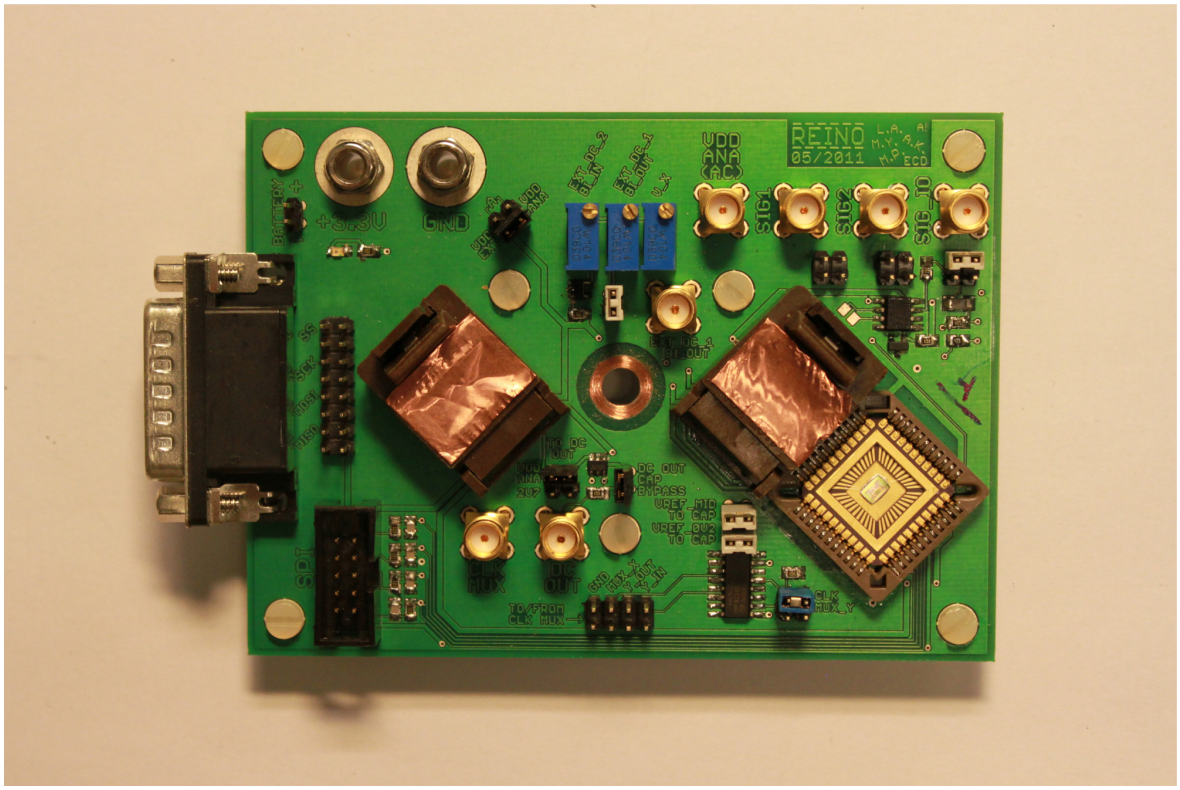
Tämän työn mikropiirilohkot toteutettiin logiikkasynteesiin pohjautuvaa digitaalisten mikropiirilohkojen suunnitteluvuota käyttäen. Lohkojen toiminta kirjoitettiin VHDL-laitteistonkuvauskielellä ja niiden toiminta simuloitiin Mentor Graphicsin QuestaSimillä. Desimointisuodattimen toiminta testattiin Alteran EP1C20-FPGA-piirillä Quartus-ohjelmistoa ja Nios Development Kit Cyclone Edition-pakettia käyttäen. Synteesi tehtiin Synopsyksen Design Visionilla ja digitaalisolujen sijoittelu ja reititys Cadencen Encounterilla. Lopulliseen piirikuvioon perustuvat (engl. *post-layout*), ajoituksen toimivuuden varmistavat simuloinnit tehtiin Mentor Graphicsin QuestaSimillä. Lopuksi tehtiin vielä DRC-, LVS- ja antennitarkastukset Cadencen IC Design Toolsissa. Näin saatiin varmuus, että digitaalilohko ei aiheuta virheitä, kun vastaavat testit ajetaan lopulliselle piirille, jossa digitaalilohkojen piirikuvio on yhdistettynä analogialohkojen piirikuvioiden kanssa.

Metastabiilisuuteen ja synkronointiin perehdyttäessä hyödynnettiin Spectre-analogiapiirisimulaattoria ratkaisuaikavakion selvittämisessä. Spectrellä simuloitiin myös juotospistesolujen ja joidenkin analogialohkojen virrankulutukset, kun selvitettiin digitaalilohkon virrankulutusta ja sen osuutta kokonaisvirrankulutuksesta. Digitaalilohkojen dynaamista virrankulutusta ja vuotovirtaa simuloitiin Synopsyksen PrimeTime-ohjelmalla. Näissä virtasimulaatioissa hyödynnettiin lopulliseen piirikuvioon perustuneista QuestaSim-simulaatioista saatua signaalien kytkemisdataa.

Työn osana piirrettiin anturipiiriä varten testipiirilevy Eagle-ohjelmalla, mutta analogialohkojen testaukseen tarvittavien PCB-kytkentäkaavioiden suunnittelusta vastasivat analogialohkojen suunnittelijat. Piirilevy on esitetty kuvassa 9.1. Piirien mittauksissa käytettiin Aardvark USB-SPI Host Adapteria datan lukemiseen ja kirjoittamiseen piirille. Pienillä taajuudenpudotussuhteilla kaikkea lähtödataa ei ehditä lukemaan käytetyn USB-SPI-laitteen suurimmallakaan kellotaajuudella, joten dataa luettiin myös Tektronixin TLA 720-logiikka-analysaattorilla.

Mittauksia varten tehtiin Labview'lla mittausohjelmat, jotka ohjaavat USB-SPI-laitetta ja jotka räätälöitiin juuri työn piirien toimintojen ohjaamiseen. Ohjelmilla voitiin tallentaa piirille ohjausbittejä, lukea niitä ja seurata kahdelta kiihtyvyysakselilta saatavaa dataa reaalia-

jassa. Lisäksi aiemmin kandidaatintyössä [36] toteutettu mittausohjelma, joka testaa piirin antaman kiihtyvyystiedon lineaarisuuden eri lämpötiloissa, laajennettiin kommunikoimaan SPI-väylää käyttävien piirien kanssa.



Kuva 9.1: Mittauksia varten suunniteltu ja tilattu piirilevy.

Luku 10

Yhteenveto ja pohdinnat

Työssä toteutettiin 0,35 μm:n CMOS-teknologialla SPI-väylä, muistirekisterit, CIC-desimointisuodattimet sekä lähtöarvojen poikkeamien ja vahvistusten digitaalinen korjaus kaksiakseliseen, tutkimuskäyttöön tarkoitettuun kiihtyvyysanturipiiriin. Lisäksi toteutettiin synkronoija kiihtyvyysdatan asynkronisesta lukemisesta aiheutuvien lukuvirheiden vähentämiseksi. Koko kiihtyvyysanturipiirin mikrovalokuva on esitetty liitteessä C.

Standardiväylät helpottavat niin anturipiirien kuin muidenkin mikropiirien käyttöönottoa, sillä esimerkiksi mikrokontrollerit ja muun tyyppiset ohjainlaitteet sisältävät usein tuen niille. Erityisesti etua on sarjamuotoisen väylän käyttämisestä, koska sellainen vähentää tarvittavien mikropiirin jalkojen määrää ja mikropiiri saadaan mahdutettua pienempään koteloon, mikä edelleen pienentää tilavaatimuksia järjestelmässä, johon piiri liitetään. Työssä toteutettiin kaksisuuntainen SPI-väylä, jonka avulla piirille voidaan kirjoittaa säätöbittejä ja lukea sekä niitä että kiihtyvyysdataa erillistä SPI-isäntälaitetta käyttämällä. Tiedonsiirtopurskeet ovat 16 kellopulssin pituisia ja suurin testattu väylän kellotaajuus on 8 MHz. Piirille toteutettiin yhteensä 212 bitin muistirekisterilohko analogia- ja digitaalilohkojen säätöbittejä varten. SPI-väylä- ja muistirekisterilohkot toimivat mittausten perusteella suunnitellun mukaisesti.

Toteutetut desimoivat CIC-suodattimet suodattavat anturipiirin kahdelta ylinäytteistävältä deltasigma-AD-muuntimelta tulevia yksibittisiä signaaleja. Niissä näytteistystaajuus on suuri varsinaiseen signaalikaistaan verrattuna ja suurilla taajuuksilla on paljon kvantisointikohinaa. Signaalin suodattamiseen tarvitaan siis suodatin, jolla on pieni päästökaista. CIC-suodattimella sellainen saadaan aikaiseksi pelkkiä summaimia ja viiverekistereitä käyttämällä. Kertojia sisältävä FIR-suodatin veisi enemmän pinta-alaa. CIC-suodattimet pudottavat signaalien näytteistystaajuudet 100 kHz:stä 100 Hz:iin ja kasvattavat sananleveydet 31:een bittiin. Suodattimen lisäksi signaalinkäsittelylohkoon toteutettiin summaimet ja kertoja, joiden avulla anturipiiriyksilöiden digitaaliset lähtöarvot saadaan vastaamaan paremmin toisiaan.

Synkronoijalla voidaan vähentää huomattavasti lukuvirheiden määrää, kun dataa luetaan piiriltä asynkronisesti. Toteutetussa synkronoijassa DSP-lohkolta tuleva kiihtyvyysdata näyt-

teistetään vararekisteriin SPI-purskeen aikana SCK-kellon kuudennella nousevalla reunalla. Näin ollen kiihtyvyysdata on tallessa kahdessa eri rekisterissä, joista vähintään toisessa on aina synkronointivirheetöntä kiihtyvyysdataa. Synkronoijalohkon tuloa, eli DSP-lohkon lähtöä, liipaisevan kellosignaalin arvoja vertailemalla saadaan selville, kummasta lähteestä virheetön data saadaan. Laskelmien perusteella tällä menetelmällä saadaan virheiden keskimääräiseksi aikaväliksi $3{,}7 \cdot 10^{1025{,}4}$ sekuntia. Synkronoijan suorituskykyä ei testattu mittauksin.

Toteutetussa digitaalilohkossa on 2500 logiikkaporttia. Ytimen pinta-ala on 0,60 mm^2 ja käyttöjännite- ja maavetojen kanssa kokonaispinta-ala on 0,67 mm^2. Alilohkojen pinta-alat on vielä taulukkoon 10.1. Koko digitaalilohkon virrankulutus on 12 μA, kun lohkolle tulevan yksibittisen kiihtyvyysdatan kellotaajuus on 100 kHz, lähdön kellotaajuus on 100 Hz ja molempien akseleiden 24-bittiset kiihtyvyysarvot luetaan SPI-väylän kautta nopeudella 100 SPS. SPI-väylä toimi täysin suunnitellun mukaisesti. CIC-suodattimien vahvistusvaste saatiin mitattua antamalla sille sinimuotoisia herätteitä AD-muuntimen kautta. Piirille olisi kuitenkin ollut syytä toteuttaa mahdollisuus syöttää suodattimille suoraan digitaalista dataa ulkoisesta lähteestä täsmällisempien mittausten, eli esimerkiksi impulssivasteen mittaamisen, mahdollistamiseksi. Lisäksi olisi ollut eduksi toteuttaa tehonsyöttö työn digitaalilohkoille erillisen käyttöjännitenastan kautta, jolta olisi mennyt virtaa ainoastaan tämän työn lohkoille, jolloin vuotovirtakin olisi saatu mitattua. Piirille olisi myös voitu tehdä jokin menetelmä synkronoijan suorituskyvyn mittaamisen mahdollistamiseksi.

Taulukko 10.1: Toteutettujen lohkojen sisältämien standardisolujen viemät pinta-alat.

Lohko	**Solujen pinta-ala (mm^2)**	**Solujen pinta-ala (%)**
SPI-väylä	0,017	2,9
Muistirekisterit	0,093	15,6
CIC-suodatin 1	0,126	21,1
CIC-suodatin 2	0,126	21,1
Booth-kertoja	0,061	10,2
Taajuudenjakaja	0,013	2,2
Synkronoija	0,036	6,0
Muut[1]	0,124	20,8
Yhteensä	**0,596**	**100%**

[1]Ylimääräistä pinta-alaa tarvitaan esimerkiksi sijoittelu- ja reititystyökalun lisäämiä kellopuskureita varten ja reititystä varten, jos signaalivetoja on paljon. Reitityksen vaatimaa ylimääräistä pinta-alaa kasvattaa digitaalilohkon päällä olevat, käyttöjännitteelle ja maalle tarkoitut metallivedot, jotka vievät tilaa reitityksiltä. Erityisesti se vaikuttaa pinta-alan tarpeeseen, kun käytössä on vain vähän metallikerroksia.

Viitteet

[1] Anturiohjelehdet: Freescale Semiconductor MMA7455, MEMSIC MXA2300J, Freescale Semiconductor MMA6281

[2] McGrath, D., *Growth of motion sensing IC market slowing*, artikkeli EE Times-sivustolla, 10.5.2013. Viitattu 17.5.2014. Saatavissa: http://www.electronics-eetimes.com/en/growth-of-motion-sensing-ic-market-slowing.html?cmp_id=7&news_id=222916831.

[3] Roetenberg, D., Luinge, H., Slycke, P., *Xsens MVN: Full 6DOF Human Motion Tracking Using Miniature Inertial Sensors*, Xsens-yhtiön julkaisu, 3.4.2013.

[4] Khoshnoud, F., de Silva, C. W., *Recent Advances in MEMS Sensor Technology - Biomedical Applications*, Instrumentation & Measurement Magazine, IEEE, Vol. 15, Issue 1, February 2012.

[5] *STMicroelectronics Takes Pressure Sensors to New Heights*, uutinen EDN Network-sivustolla, 7.12.2010. Viitattu 17.5.2014. Saatavissa: http://www.edn.com/electronics-news/4388582/STM-takes-pressure-sensors-to-new-heights.

[6] *VTI measures aircraft tire pressure*, uutinen Murata Electronics Oy:n verkkosivuilla, 11.6.2007. Viitattu 17.5.2014. Saatavissa: http://www.muratamems.fi/en/news/press-releases/vti-measures-aircraft-tire-pressure.

[7] Lopez, S., *Application Note AN4328: Blood Pressure Monitor Fundamentals and Design*, Freescale Semiconductor Inc., 2011.

[8] Ifeachor, E. C., Jervis, B. W., *Digital Signal Processing: A Practical Approach*, Addison-Wesley Publishers Ltd., Yhdysvallat, 1995.

[9] Kories, R., Schmidt-Walter, H., *Electrical Engineering: A Pocket Reference*, Springer-Verlag Berling Heidelberg, Saksa, 2003.

[10] Mitra, S. K., *Digital Signal Processing: A Computer Based Approach*, 3. painos, McGraw-Hill, Yhdysvallat, 2006.

[11] Floyd, T. L., *Digital Fundamentals With VHDL*, Prentice Hall Inc., Yhdysvallat, 2003.

[12] Yucetas, M. et al., *A Chopper Stabilized Low-Power Differential $\Delta\Sigma$ ADC*, Proc. Conf. Ph. D. Research in Microelectronics and Electronics, Madonna di Campiglio, Italy, heinäkuu 2011, ss. 121-124.

[13] Tsuzuki, T., Fisher, C., *Application Note AN-1063: Oversampling Technique to Improve ADXL345 Output Resolution*, Analog Devices Inc., 2010.

[14] Salomaa, J., Yucetas, M., Kalanti, A., Aaltonen, L., Halonen, K., *A $\Delta\Sigma$ ADC for Low Power Sensor Applications*, Proc. of IEEE International Symposium on Circuits and Systems 2010, ss. 3100-3103.

[15] Hogenauer, E. B., *An Economical Class of Digital Filters for Decimation and Interpolation*, IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-29, No. 2, huhtikuu 1981.

[16] Lehtinen, V., Renfors, M., *Truncation Noise Analysis of Noise Shaping DSP Systems with Application to CIC Decimators*, European Signal Processing Conference, 3.-6.9.2002, Toulouse, Ranska, ss. 207-210.

[17] Dolecek, G. J., Mitra, S. K., *Simple method for compensation of CIC decimation filter*, Electronics letters, Vol. 44, No. 19, 11.9.2008.

[18] Schreier, R., Temes, G. C., *Understanding Delta-Sigma Data Converters*, Yhdysvallat, Wiley-Interscience, 2005.

[19] Anturiohjelehdet: Sensonor SAR10 / SAR100 / SAR 150, STMicroelectronics LIS302DL / LIS331DL, Analog Devices ADIS16003 / ADIS16204 / ADXL345, Invensense MPU-6000 / MPU-6050 / MPU-6100 / MPU-6500 / IXZ-2020 / IDG-2020

[20] *I^2C-bus specification and user manual Rev. 03*, NXP Semiconductors, 19.6.2007.

[21] *High-Definition Multimedia Interface Specification*, Version 1.3

[22] *SPI Block Guide V04.01*, Motorola Inc., 2004.

[23] *Daisy-Chaining SPI Devices*, Maxim Application Note AN3947, 15.11.2006.

[24] *Murata Automotive Digital Accelerometer Platform SCA8X0/21X0/31X0 Product Family Specification*, Murata Electronics Oy.

[25] Rabaey, J. M., *Digital Integrated Circuits: A Design Perspective*, Prentice-Hall, Yhdysvallat, 1996.

[26] Booth, A. D., *A Signed Binary Multiplication Technique*, Quarterly Journal of Mechanics and Applied Mathematics, Vol. IV, pt. 2, 1951.

[27] *Booth's multiplication algorithm*, Wikipedia-artikkeli, 2.3.2011. Viitattu: 19.5.2014. Saatavissa: http://en.wikipedia.org/wiki/Booth's_multiplication_algorithm.

[28] Portmann, C. L., *Characterization and reduction of metastability errors in CMOS interface circuits*, tekninen raportti, Stanfordin yliopisto, Stanford Computer Systems Laboratory, Stanford, Kalifornia, Yhdysvallat, 1995.

[29] Foley, C., *Characterizing Metastability: Practical Measurement Techniques to accurately determine "device dependent coefficients" used to predict synchronizer MTBF.*, Second International Symposium on Advanced Research in Asynchronous Circuits and Systems, Proceedings, 1996.

[30] Ginosar, R., *Metastability and Synchronizers: A Tutorial*, IEEE Design and Test of Computers, syys-lokakuu 2011.

[31] *74HC574, 74HCT574 Octal D-type flip-flop Rev. 5*, ohjelehti, NXP Semiconductors, 2012.

[32] Dike, C., Burton, E., *Miller And Noise Effects in a Synchronizing Flip-Flop*, IEEE Journal of Solid-State Circuits, vol. 34, no. 6, kesäkuu 1999.

[33] *0.35 μm CMOS Digital Standard Cell Databook*, työssä käytetyn CMOS-teknologian standardisolujen ohjelehti, Austriamicrosystems, 2012.

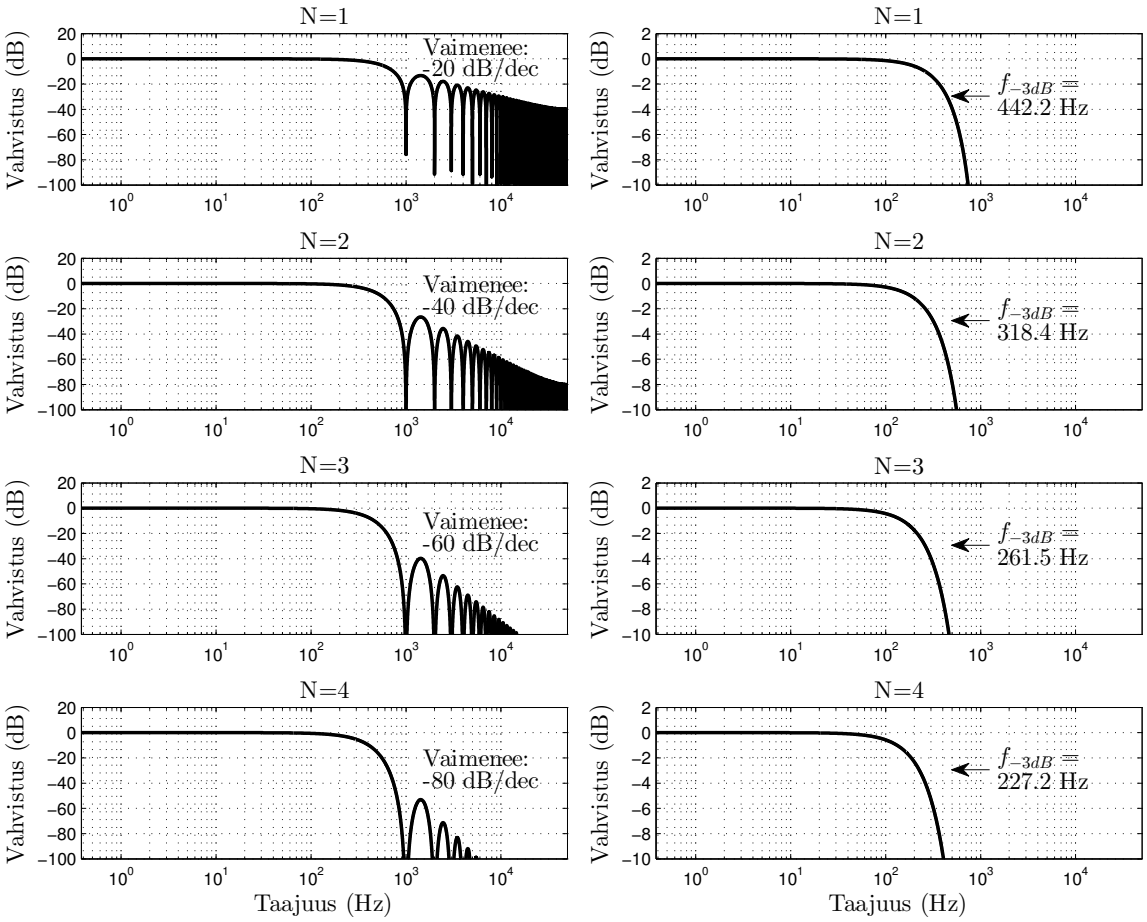
[34] Kurd, N. A., Barkatullah, J. V., Dizon, R. O., Fletcher, T. D., Madland, P. D., *A Multi-gigahertz Clocking Scheme for the Pentium 4 Microprocessor*, IEEE Journal of Solid-State Circuits, vol. 36, no. 11, marraskuu 2001.

[35] Weste, N. H. E., Eshraghian, K., *Principles of CMOS VLSI Design: A Systems Perspective*, 2. painos, Addison-Wesley Publishing Company, Yhdysvallat, 1993.

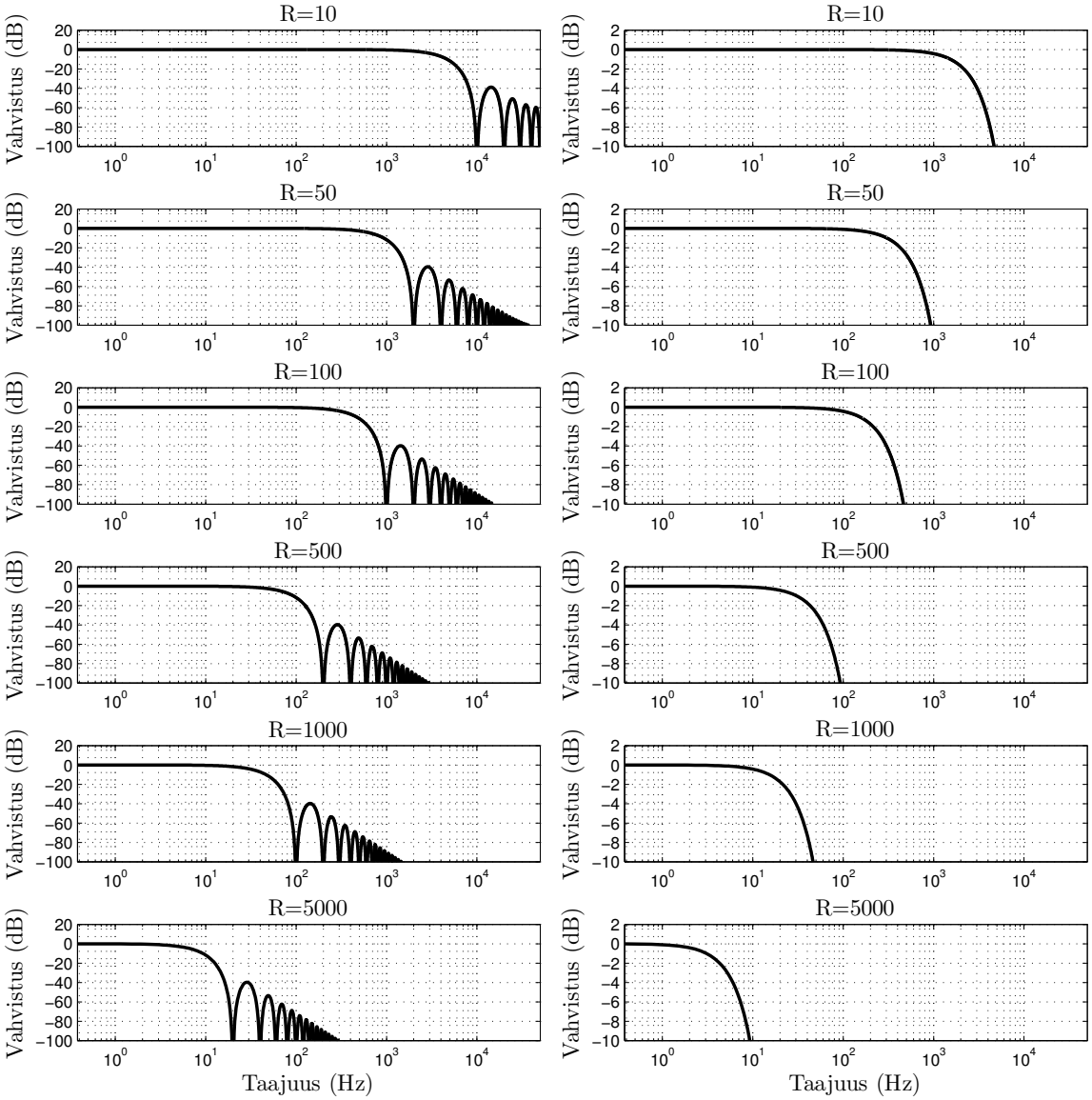
[36] Pulkkinen, M., *Inertiaaliantureiden testausautomatisointi Labview-ohjelmalla*, kandidaatintyö, Teknillinen korkeakoulu, Sähkö- ja tietoliikennetekniikan osasto, Espoo, 2008.

Liite A

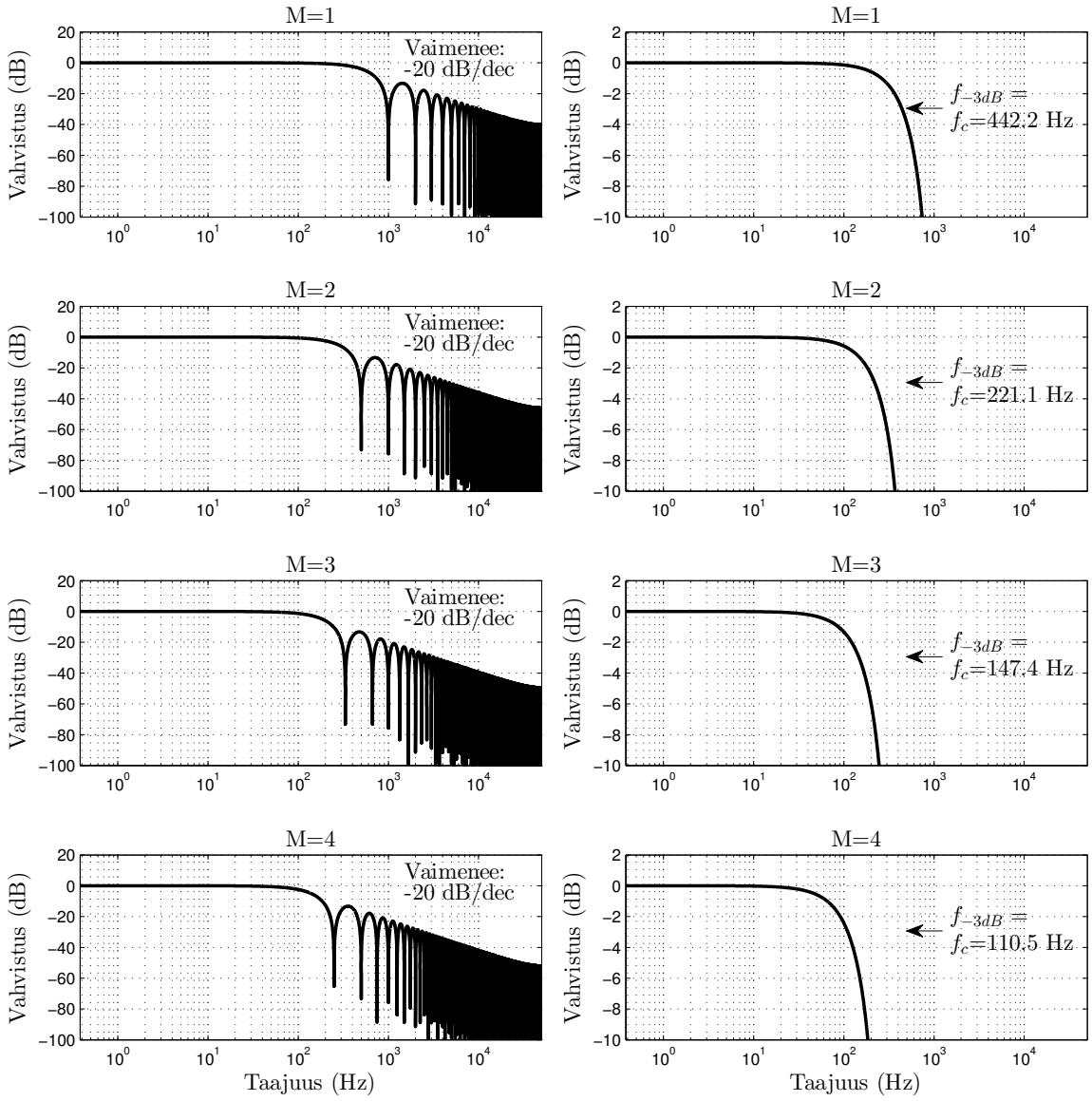
CIC-suodattimen vahvistusvasteita



Kuva A.1: CIC-suodattimen vahvistusvaste suodattimen asteluvun N eri arvoilla, kun tulon näytteistystaajuus on f_{in} = 100 kHz, taajuudenpudotussuhde on R = 100 ja kampojen viive on M = 1. Oikealla puolella vahvistusakseli on tarkennettu välille -10...+2 dB.



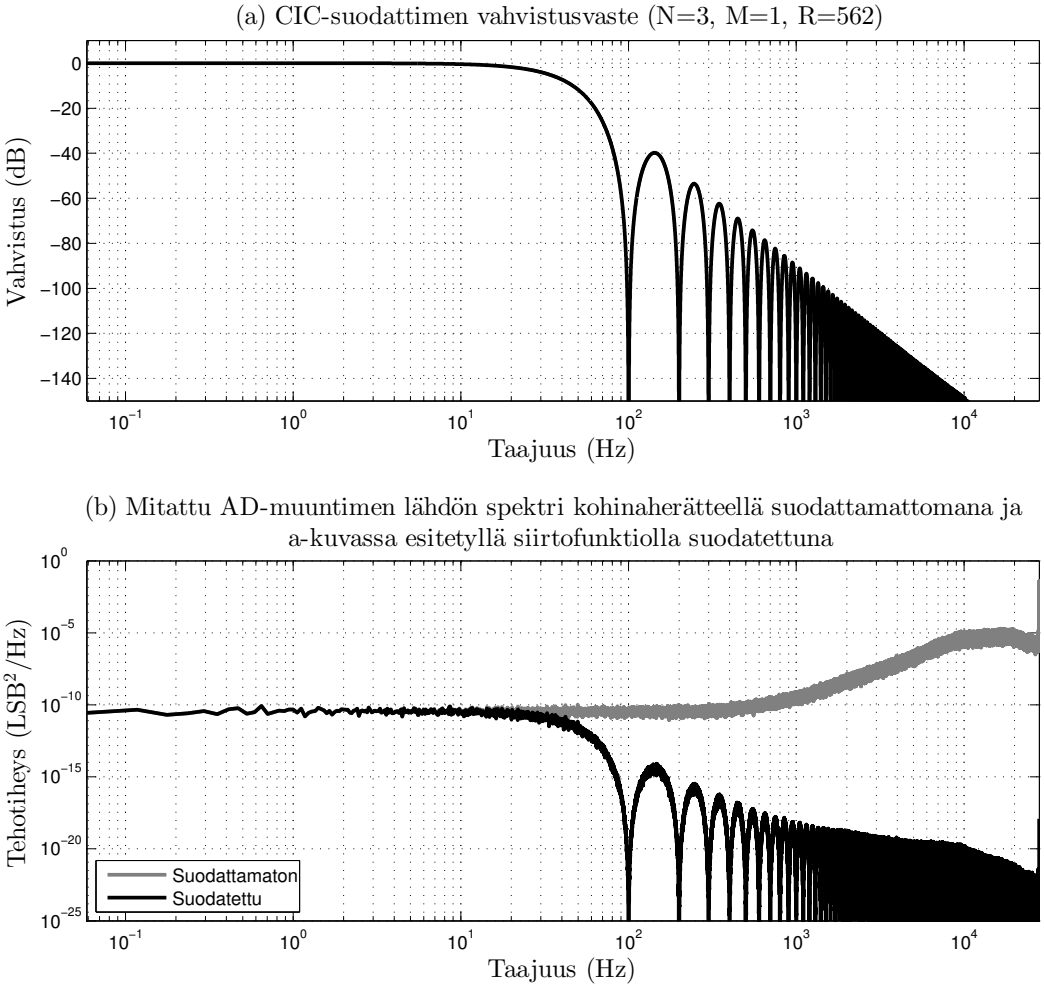
Kuva A.2: CIC-suodattimen vahvistusvaste taajuudenpudotussuhteen R eri arvoilla, kun tulon näytteistystaajuus on f_{in} = 100 kHz, suodattimen asteluku on N = 3 ja kampojen viive on M = 1. Oikealla puolella vahvistusakseli on tarkennettu välille -10...+2 dB.



Kuva A.3: CIC-suodattimen vahvistusvaste kampojen viiveen M eri arvoilla, kun tulon näytteistystaajuus on f_{in} = 100 kHz, suodattimen asteluku on N = 1 ja taajuudenpudotussuhde on R = 100. Oikealla puolella vahvistusakseli on tarkennettu välille -10...+2 dB.

Liite B

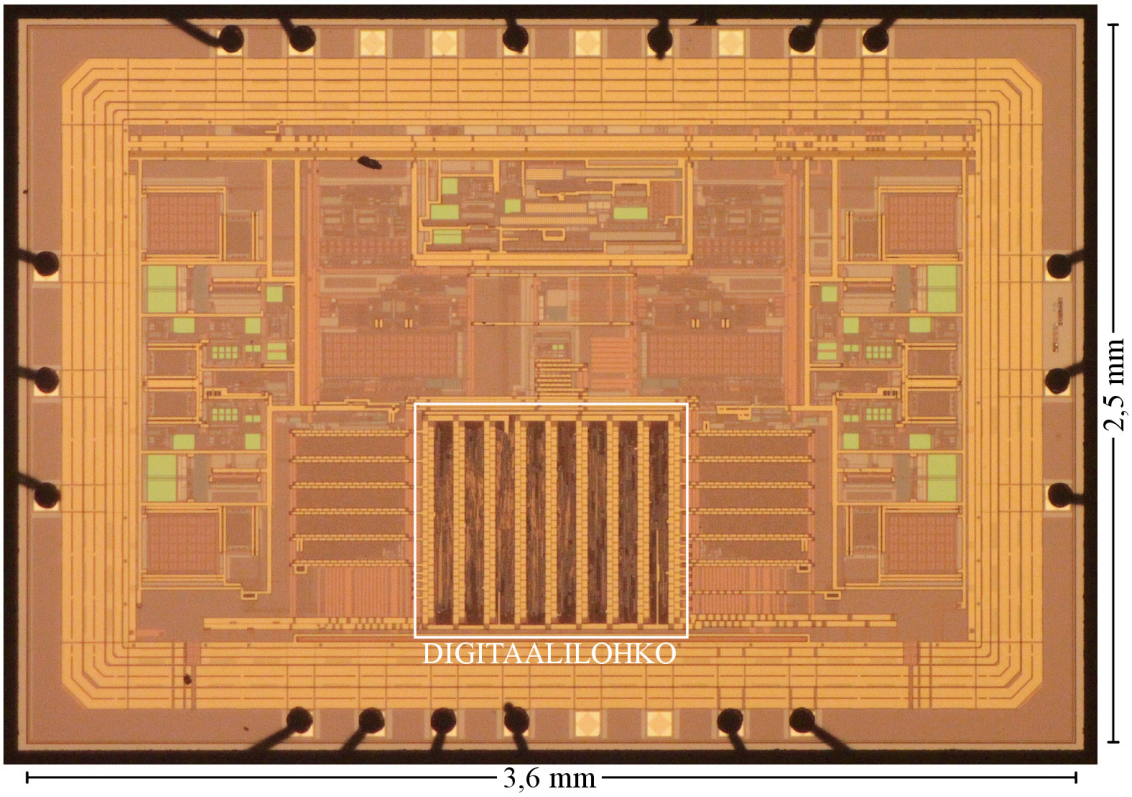
Kohinan suodattuminen CIC-suodattimessa



Kuva B.1: Ylemmässä kuvassa on CIC-suodattimen ($N = 3$, $M = 1$ ja $R = 562$) vahvistusvaste. Alemmassa kuvassa on mitatun AD-muuntimen lähtösignaalin kohinan spektri suodattamattomana ja ylemmän kuvan siirtofunktiolla suodatettuna. Suodatetun signaalin spektri on tuotettu kertomalla suodattamattoman kohinan spektri ylemmän kuvan siirtofunktiolla.

Liite C

Mikrovalokuva prosessoidusta piiristä



Kuva C.1: Prosessoitu kiihtyvyysanturipiiri, johon työn digitaalilohkot suunniteltiin. Toteutettu digitaalilohko on rajattu valkoisella viivalla.