Tomi Piirainen

PAINESÄÄTÖ LANGATTOMASSA PROSESSISOLMUSSA

Insinöörityö
Kajaanin ammattikorkeakoulu
Tekniikan ala
Tietotekniikan koulutusohjelma
Kevät 2006

Kajaanin ammattikorkeakoulu

OPINNÄYTETYÖ
TIIVISTELMÄ

Koulutusala	Koulutusohjelma
Tekniikka	Tietotekniikka
Tekijä(t) Tomi Piirainen	
Työn nimi Painesäätö langattomassa prosessisolmussa	
Vaihtoehtoiset ammattiopinnot Konenäkö ja mittaustekniikka	Ohjaaja(t) Veijo Sutinen, Arto Partanen
	Toimeksiantaja Oulun yliopisto, Mittalaitelaboratorio
Aika Kevät 2006	Sivumäärä ja liitteet 58 + 3

Digitaalinen säädin voidaan toteuttaa ohjelmallisesti prosessorilla tai ohjelmoitavilla logiikkapiireillä. Nykyään on saatavilla myös järjestelmäpiirejä, jotka sisältävät prosessorin, FPGA-lohkon ja muistin samalla piirillä. Tällaisten piirien käyttö säätötekniikassa ja signaalinkäsittelyssä on yleistymässä niiden hyvän suorituskyvyn takia.

Insinöörityössä toteutettiin säädin langattomaan prosessisolmuun, joka perustui Atmelin valmistamaan FPSLIC-järjestelmäpiiriin. Prosessisolmu on osa langatonta mittaus- ja ohjausjärjestelmää. PID-säätimen algoritmi toteutettiin aluksi kehitysalustalla piirin FPGA-lohkoon, jossa sen toiminta testattiin. Seuraavaksi säädin muokattiin sopivaksi prosessisolmuun, jossa säätimen ohjaus ja parametrien välitys toteutettiin langattoman verkon kautta. Säädin ja säädettävän suureen mittaus olivat molemmat samalla prosessisolmulla. Säätimen algoritmi toteutettiin piirin FPGA-lohkolle ja säädintä ohjattiin piirille integroidun prosessorin avulla. Säädin toteutettiin lohkopohjaisena, joten se oli siirrettävissä myös muihin sovelluksiin.

Lisäksi työssä tutkittiin mittaustiedon reititysmenetelmiä langattoman verkon kautta prosessisolmulta toiselle. Työssä tutkittiin kolmea eri toteutusvaihtoehtoa, jotka erosivat toisistaan tiedonvälitysreitin osalta. Tutkimuksessa arvioitiin viivettä, joka kului mittaushetkestä siihen, kun mittaustulos oli säätimellä. Lisäksi pohdittiin kuinka toteutuskelpoisia menetelmät olivat. Paras reititysmenetelmä olisi käyttää Jabber-palvelinta tiedon välittämiseen, koska viive on pieni ja se olisi helpoin toteuttaa. Tämä toteutus mahdollistaisi myös sen, että mittaustieto voitaisiin lähettää mille tahansa prosessisolmulle.

Kieli	Suomi
Asiasanat	PID, digitaalinen säätö, FPGA, FPSLIC
Säilytyspaikka	☒ Kajaanin ammattikorkeakoulun Kaktus-tietokanta ☒ Kajaanin ammattikorkeakoulun kirjasto

Kajaanin ammattikorkeakoulu
University of Applied Sciences

THESIS
ABSTRACT

School	Degree Programme
Engineering	Information Technology
Author(s) Tomi Piirainen	
Title A Pressure Control System in a Wireless Process Node	
Optional Professional Studies Measurement and Machine Vision	Instructor(s) Veijo Sutinen, Arto Partanen
	Commissioned by University of Oulu, Measurement and Sensor Laboratory
Date Spring 2006	Total Number of Pages and Appendices 58 + 3

Because of the fast speed and growing performance of new digital technologies, almost all controls are based on digital systems. A digital control system can be implemented, for example, with a processor or programmable logic. There are also System on Chip circuits available. They include a processor, an FPGA and the memory in one chip. These kinds of circuits are becoming more general in digital signal processing and also in control systems because of their good performance.

In this Bachelor's thesis a PID control system was implemented into a wireless process node based on the Atmel's FPSLIC circuit. The process node is part of a bigger wireless control and measurement system. First the algorithm of the PID control was implemented into the starter kit in which its operation was also tested. Next, the control system was implemented into the wireless process node and the parameters of the control system were sent by the wireless communication. Both the measurement of the controlled variable and the PID control were in the same process node. The control system was designed so that it could be used easily in other applications.

In this thesis it was also examined how measurement of the controlled variable could be sent to the control system using wireless communication. Three different solutions were compared with each other. The biggest differences between the solutions were how measurements were sent to the control system and how long the delay was. One of the major criteria was also that how easily the solutions could be implemented into the current system. Using Jabber server to send the measurement would be the best solution because the delay is quite short and the solution is the easiest to realize.

Language of Thesis	Finnish
Keywords	PID, digital control systems, FPGA, FPSLIC
Deposited at	☒ Kaktus Database at University of Applied Sciences Library ☒ Library of University of Applied Sciences

ALKUSANAT

Tämä insinöörityö on tehty Oulun yliopiston Kajaanin yliopistokeskuksen Mittalaitelaboratorion toimeksiannosta. Työn valvojana toimi Arto Partanen Kajaanin ammattikorkeakoulusta ja ohjaajana Veijo Sutinen Valtion teknillisestä tutkimuslaitoksesta (VTT). Haluan osoittaa heille kiitokset työn ohjaamisesta.

Haluan myös kiittää Jari Perttua Mittalaitelaboratoriosta sekä Marko Korkalaista VTT:ltä korvaamattomista neuvoista työn suoritusvaiheessa. Kiitän myös muita työn ohjaukseen liittyviä henkilöitä, kuten Eero Soinista kielellisestä ohjauksesta ja Kaisu Korhosta englanninkielisen abstraktin ohjaamisesta.

Kajaanissa 12.4.2006

_____

Tomi Piirainen

SISÄLLYSLUETTELO

KÄYTETYT LYHENTEET

KÄYTETYT LYHENTEET

ASIC	Application Specific Integrated Circuit
CPLD	Complex PLD
FPGA	Field Programmable Gate Array
FPSLIC	Field Programmable System Level Integrated Circuit
HDL	Hardware Description Language
LSB	Least Significant Bit
LUT	Look-Up Table
MSB	Most Significant Bit
PID	Proportional, Integral, Derivative
PLD	Programmable Logic Device
SoPC	System on a Programmable Chip
SPI	Serial Peripheral Interface
SPLD	Simple PLD
SRAM	Static Random Access Memory
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit

1 JOHDANTO

Ensimmäiset säätöjärjestelmät keksittiin Kreikassa jo ennen ajanlaskun alkua. Yleensä säätimet oli tarkoitettu pitämään veden, öljyn tai muun nesteen pinnankorkeus vakiona. Euroopassa kehitettiin 1600-luvun alkupuolella ensimmäinen takaisinkytketty säätöjärjestelmä, joka oli Cornelis Drebbelin lämpötilansäädin. Vähän myöhemmin, 1600-luvulla, Dennis Papin kehitti höyrykattiloihin ensimmäisen paineensäätimen, joka toimi mallina myös nykypäivän höyrykeittimien ja -kattiloiden suojaventtiileille. Ensimmäisen teollisuuden säätöjärjestelmän kehitti James Watt, joka oli merkittävimpiä höyrykoneen kehittäjiä. [1, s. 4.] Hän kehitti höyrykoneeseen 1700-luvun lopulla "Flyball Governorin". Se oli keskipakovoimalla toimiva höyrykoneen akseliin kytketty höyryventtiilisäädin, joka avasi koneen päähöyryventtiiliä pyörimisnopeuden ollessa liian alhainen ja sulki sen pyörimisnopeuden ollessa riittävän suuri. [2.]

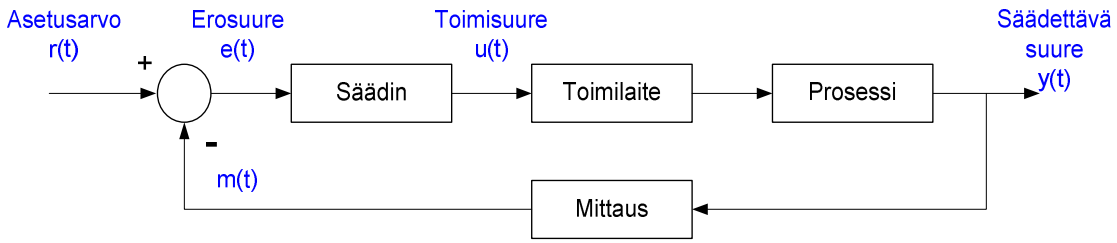
Tietotekniikan komponenttien halpeneminen ja suorituskyvyn huima kasvu ovat johtaneet siihen, että suurin osa nykyään toteutettavista säätöjärjestelmistä perustuu digitaaliseen säätöön. Digitaalinen säätö voidaan toteuttaa esimerkiksi mikroprosessorin tai ohjelmoitavan piirin avulla. Teollisuudessa eniten käytetyllä PID-säädöllä (Proportional Integral Derivative) saadaan ratkaistua useimmat säätöongelmat. [3.]

Työn tavoitteena oli suunnitella ja toteuttaa painesäätö langattomaan prosessisolmuun. Työ tehtiin Oulun yliopiston Kajaanin yliopistokeskuksen Mittalaitelaboratoriolle LAMITY-projektiin, joka on VTT:n kanssa toteutettu langattoman instrumentoinnin kehitysprojekti. Työssä suunniteltiin ja toteutettiin säädin langattomaan mittaus- ja ohjausjärjestelmään, jolla ohjataan paperimassan kierrätyslaitoksen toimintaa. Järjestelmän mittaus- ja ohjaussolmu perustuu Atmelin FPSLIC-piirin (Field Programmable System Level Integratated Circuits), jossa samalle piirille on integroitu mikrokontrolleri, FPGA-lohko ja muistia. Ohjelmoitavan piirin suunnitteluun käytettiin System Designer -ohjelmistoa. Säätimestä suurin osa toteutettiin piirin FPGA-lohkolla. Lisäksi työssä tutkittiin mittaustiedon reititysmenetelmiä prosessisolmulta toiselle.

2 SÄÄTIMET

Säätimen tarkoituksena on toimilaitteen avulla ohjata säädettävää suuretta siten, että anturilta saatava mittaussignaali on yhtä suuri käyttäjän säätimelle antaman asetusarvon kanssa. Toimilaitteena voi olla esimerkiksi venttiili tai moottori ja säädettävänä suureena paine tai lämpötila. [4, s. 2.] Esimerkiksi jos säädettävään systeemiin halutaan asettaa 1 barin paine, on säätimen tarkoituksena säätää venttiilin asentoa siten, että asetettu paine saavutetaan, vaikka prosessin olosuhteet muuttuisivat.

Yleensä säätöjärjestelmä (kuva 1) koostuu säätimestä, toimilaitteesta, prosessista ja mittauksesta, joista viimeisin toimii takaisinkytkentänä (feedback signal) säätimelle. Mittaustulosta verrataan säätimen asetusarvoon. Näin saadaan muodostettua erosuure, jolla ohjataan säätimen toimintaa. Säädin muodostaa ohjaussignaalin toimilaitteelle, jonka ohjauksen tuloksena säädettävä suure alkaa lähetä haluttua arvoa. Lopulta säädettävä suure saavuttaa asetusarvon, jos asetusarvo pidetään vakiona. Kun säädettävä suure ja asetusarvo ovat yhtä suuria, erosuure menee nollaan ja säätimen ulostulosignaali ei muuta toimilaitteen asentoa mihinkään suuntaan. [1, s. 3.]



Kuva 1. Takaisinkytketyn säätöjärjestelmän periaate [5, s. 2]

Säätimien algoritmit digitaalisissa säätöjärjestelmissä voidaan toteuttaa useilla eri tavoilla. Yleisin käytössä oleva säätöalgoritmi on PID-säätö ja sen eri variaatiot. PID-säädin koostuu kolmesta erillisestä termistä P, I ja D sekä kolmesta viritysparametrista K, T_I ja T_D. [6, s. 71.] Säätimiä voidaan toteuttaa myös käyttämällä vain yhtä tai kahta termiä, jolloin saadaan joko P-, PI- tai PD-säädin, joista viimeisin on melko harvoin käytetty.

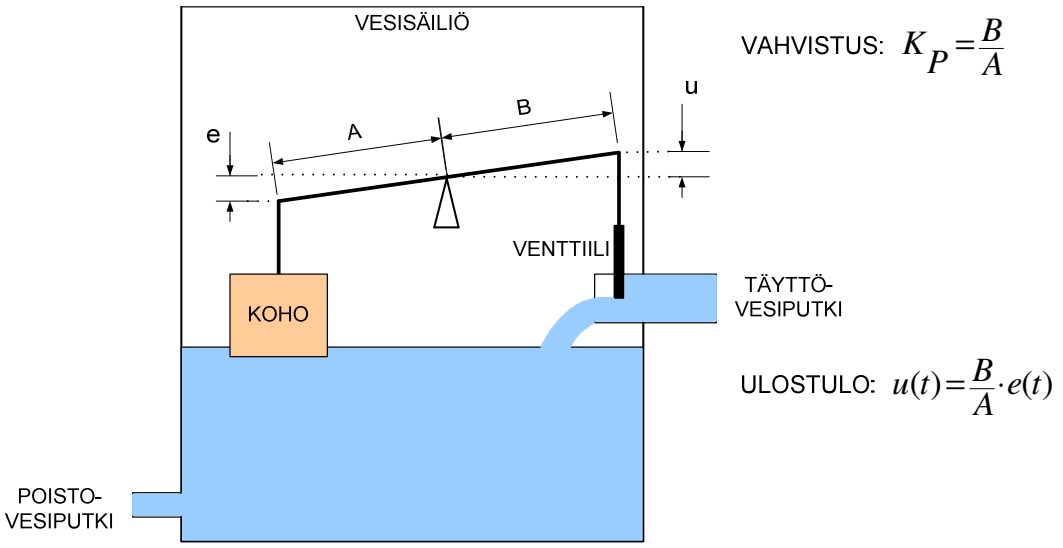
2.1 P-säätö (Proportional)

P-säätimelle on voimassa yhtälö

$$u(t) = K_P e(t), \tag{1}$$

missä $u(t)$ on säätimen ulostulo, $e(t)$ on erosuure ja K_P on vahvistus. Tässä algoritmissa ainoa viritettävä parametri on vahvistus (proportional gain) K_P. P-säädin ei sisällä mitään tietoa aiemmista erosuureen tai säätimen ulostulon arvoista, joten digitaalisissa säätimissä ei tarvita muistia P-säätimen toteuttamiseen. [4, s. 3.]

Tarkastellaan mekaanisen P-säädön toimintaa kuvan 2 mukaisen järjestelmän avulla, jossa vesisäiliön pinnankorkeus halutaan pitää vakiona. Järjestelmässä on varsi, joka on tuettu keskeltä. Varren toiseen päähän on asennettu koho ja toisessa päässä on venttiili.

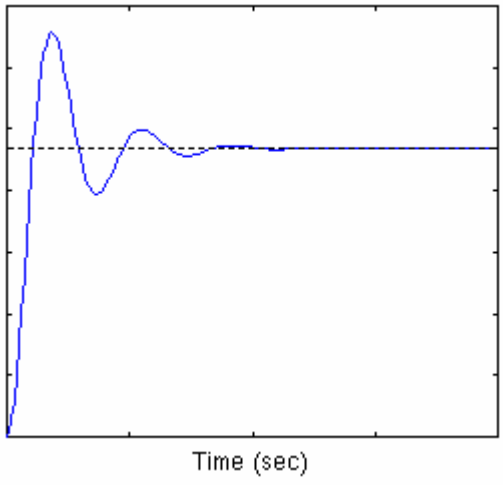


Kuva 2. Vesisäiliön pinnankorkeuden säätö

Kun pinnankorkeus vähenee, koho menee alaspäin ja venttiili aukeaa suhteessa pinnankorkeuden ja vakiopinnankorkeuden erotukseen e. Järjestelmän

vahvistus riippuu osien A ja B suhteista toisiinsa. Mitä pidempi varren B-pituus on verrattuna A-pituuteen, sitä suurempi on järjestelmän vahvistus. Kuvan mukaisessa järjestelmässä varren tuenta on keskikohdasta, jolloin vahvistus on yksi.

Kuvassa 3 on esimerkki P-säätimen askelvasteesta. P-säätimen askelvasteelle on ominaista korkea asetusarvon ylitys ja värähtely ennen kuin asetettu arvo saavutetaan. Säätimien askelvasteet riippuvat prosessista ja siitä, miten säädin on viritetty.



Kuva 3. Esimerkki P-säätimen askelvasteesta [7]

2.2 PI-säätö (Proportional, Integral)

PI-säätimeen on lisätty P-säädön lisäksi integrointitermi. Sen tehtävänä on vahvistuksen ja erosuureen lisäksi muodostaa ohjaussuureelle sellainen nollasta poikkeava arvo, että säätövirhe voisi pienentyä nollaan asetusarvon ollessa vakio. Teollisuudessa PID-säädintä käytetään usein PI-moodissa, koska kohinaisen signaalin derivoiminen ilman suodatusta ei ole järkevää. [4, s. 5.]

PI-säätimelle on voimassa yhtälö

$$u(t) = K\left[e(t) + \frac{1}{T_I}\int_0^t e(t)dt \right], \tag{2}$$

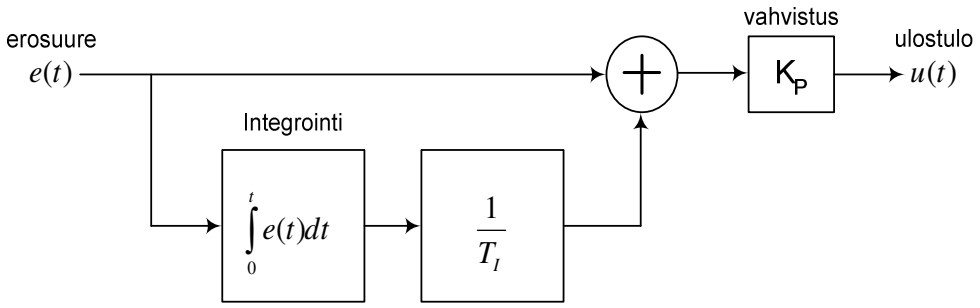
missä T_I on integrointiaika ja K on vahvistus. Integrointiajan yksikkö on joko sekunteja tai minuutteja [4, s. 4]. Määritellään:

$$K_I = \frac{K_P}{T_I}, \tag{3}$$

jolloin yhtälö (2) voidaan kirjoittaa muotoon:

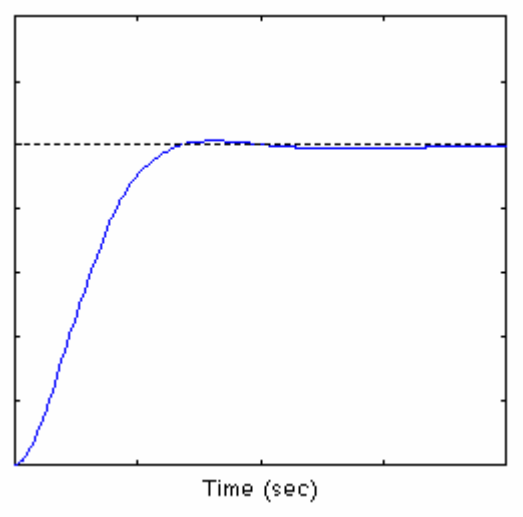
$$u(t) = K_P e(t) + K_I \int_0^t e(t)dt\ . \tag{4}$$

PI-säätimen toimintaa voidaan havainnoida kuvan 4 mukaisella lohkokaaviolla. Erosuuretta integroidaan ja kerrotaan integrointiajan käänteisarvolla $1/T_I$. Saatu tulos lasketaan yhteen erosuureen kanssa ja kerrotaan vahvistuksella K_P. Tällöin säätimen ulostulo on saatu laskettua.



Kuva 4. PI-säätimen lohkokaavio

Kuvassa 5 on esimerkki PI-säätimen askelvasteesta. PI-säätimen askelvasteeseen vaikuttavat prosessi ja se, miten parametrit on viritetty. PI-säätimen vaste ei värähtele niin paljon askelmaiseen asetusarvon muutokseen kuin P-säädin.



Kuva 5. Esimerkki PI-säätimen askelvasteesta [7]

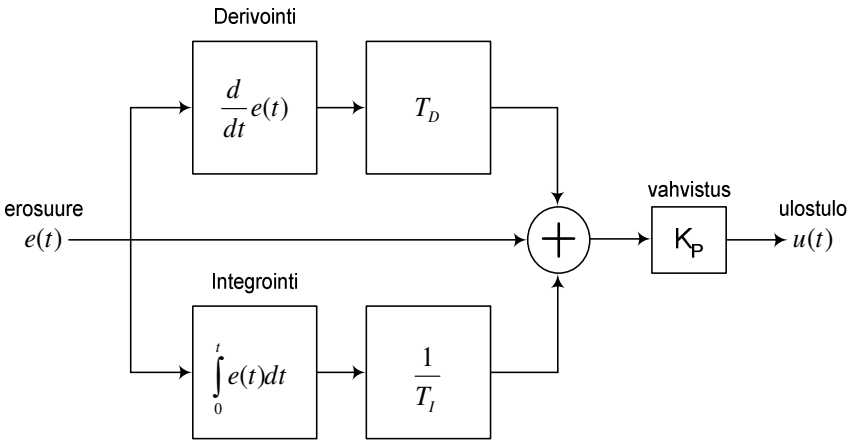
2.3 PID-säätö (Proportional, Integral, Derivative)

Säätimen nopeutta lisätään derivointitermillä, joka pyrkii ennakoinnin avulla korjaamaan ohjaussuuretta oikeaan suuntaan [4, s. 4]. Säätimelle tarvitaan edellä esiteltyjen parametrien lisäksi derivointiaika T_D. Analogiselle PID-säätimelle on voimassa yhtälö

$$u(t) = K\left[e(t) + \frac{1}{T_I}\int_0^t e(t)dt + T_D\frac{de(t)}{dt}\right], \tag{5}$$

missä $e(t)$ on erosuure, $u(t)$ on säätimen ulostulo, K on vahvistus, T_I on integrointiaika ja T_D on derivointiaika [8, *s. 115*].

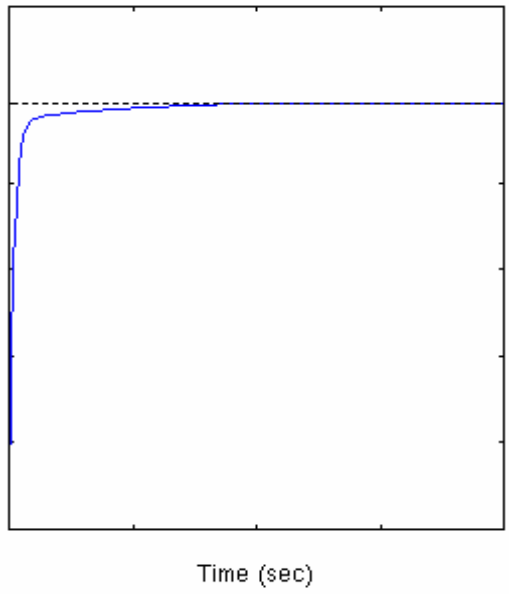
PID-säätimen toimintaa voidaan havainnoida kuvan 6 mukaisella lohkokaaviolla. Erosuuretta integroidaan ja kerrotaan integrointiajan käänteisarvolla kuten PI-säätimessäkin. PID-säätimessä on lisänä ”derivointihaara”, jossa erosuuretta derivoidaan ja kerrotaan derivointiajalla T_D. Erosuure sekä integrointi ja derivointihaarojen tulokset summataan yhteen ja kerrotaan vahvistuksella K.



Kuva 6. PID-säätimen lohkokaavio

Usein derivointitermin käyttö PID-säätimessä edellyttää voimakasta erosuureen suodatusta ennen derivointia. Suodatuksen ansiosta mitattavan signaalin kohina pienenee huomattavasti ja samalla sen vaikutus derivaattaan pienenee. [4, s. 5]

Kuvassa 7 on esitetty esimerkki PID-säätimen askelvasteesta. Säädin reagoi nopeasti askelmaisiin erosuureen muutoksiin ja saavuttaa asetusarvon nopeasti.



Kuva 7. Esimerkki PID-säätimen askelvasteesta [7]

Säätimissä vahvistuksen K_P kasvattaminen voi parantaa vasteaikaa, mutta säätimellä menee enemmän aikaa asettuakseen säätöarvoon ja pahimmassa tapauksessa se voi alkaa värähdellä. Ongelmaa voidaan parantaa virittämällä T_I ja T_D parametreja. [6, s. 74.]

2.4 Digitaalinen PID-säädin

Analogisen PID-säätimen yhtälöstä (5) voidaan johtaa myös digitaalisen säätimen yhtälöt. Pienellä näytteenottovälillä T, yhtälö (5) voidaan johtaa differenssiyhtälöksi diskretisoimalla. Derivointitermi korvataan ensimmäisen kertaluvun approksimaatiolla ja integrointitermiä approksimoidaan summalla, jolloin differenssiyhtälöksi saadaan [6, s. 71 - 72]

$$u[n] = K\left[e[n] + \frac{T}{T_I}\sum_{j=0}^{n} e[j] + \frac{T_D}{T}(e[n] - e[n-1]) \right]. \tag{6}$$

Merkitään

$$K_I = \frac{T}{T_I} \tag{7}$$

ja

$$K_D = \frac{T_D}{T}, \tag{8}$$

jolloin yhtälö (6) voidaan esittää muodossa

$$u[n] = K_P e[n] + K_I \sum_{j=0}^{n} e[j] + K_D (e[n] - e[n-1]). \tag{9}$$

Tätä PID-säätimen algoritmia kutsutaan asentoalgoritmiksi (positional algorithm). [6, s. 71 - 72.]

Yhtälöstä (9) havaitaan, että integrointitermistä johtuvaa summaa tulee päivittää aina uutta ohjaussuuretta laskettaessa. Tämä ei aina ole laskennallisesti tehokasta, vaan usein käytännön sovelluksissa käytetään inkrementaalista algoritmia, jota kutsutaan usein myös nopeusalgoritmiksi. [3.]

Säätimen ulostuloarvon muutokselle Δu on voimassa yhtälö [6, s. 72]

$$\Delta u[n] = u[n] - u[n-1]. \tag{10}$$

Sijoitetaan edelliseen yhtälöön asentoalgoritmin differenssiyhtälöt (yhtälö (6)) [3]

$$\begin{aligned}\Delta u[n] = {} & K_P\left[e[n] + \frac{T}{T_I}\sum_{j=0}^{n} e[j] + \frac{T_D}{T}(e[n] - e[n-1])\right] \\ & - K_P\left[e[n-1] + \frac{T}{T_I}\sum_{j=0}^{n-1} e[j] + \frac{T_D}{T}(e[n-1] - e[n-2])\right].\end{aligned} \tag{11}$$

Yhtälö (11) voidaan kirjoittaa muotoon [6, s. 72]

$$\Delta u[n] = K_0 e[n] + K_1 e[n-1] + K_2 e[n-2], \tag{12}$$

missä

$$K_0 = K_P + K_I + K_D,$$

$$K_1 = -K_P - 2K_D$$

ja

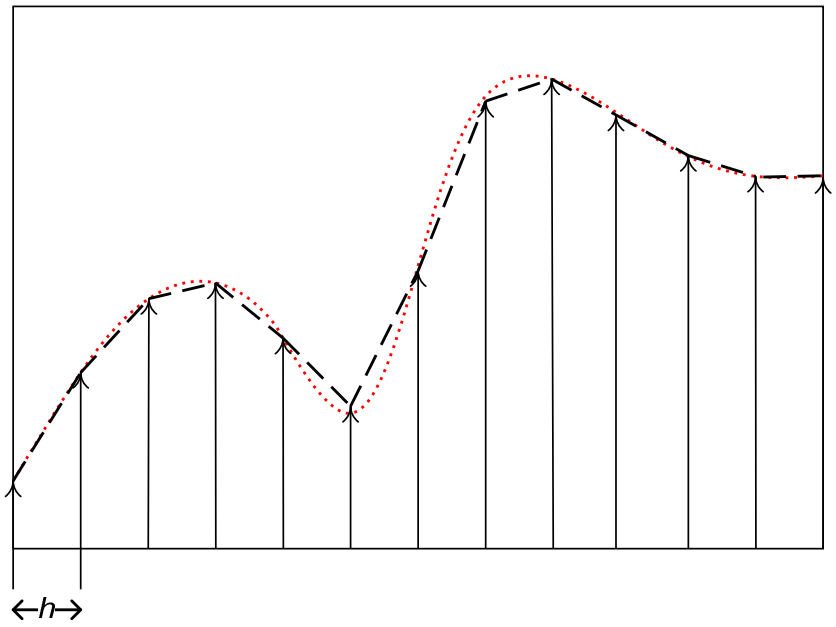
$$K_2 = K_D.$$

Inkrementaalisen algoritmin ulostulo saadaan laskettua sijoittamalla edellinen yhtälö (12) yhtälöön (10) [6, s. 72]

$$u[n] = u[n-1] + K_0 e[n] + K_1 e[n-1] + K_2 e[n-2]\,. \tag{13}$$

Derivaatan ja integraalin approksimaatiot

Digitaalisissa säätimissä derivaatan ja integraalin arvoa joudutaan approksimoimaan. Molempiin laskutoimituksiin on olemassa useita eri menetelmiä. Kuvassa 8 on esitetty taaksepäin derivoinnin periaate. [3.]



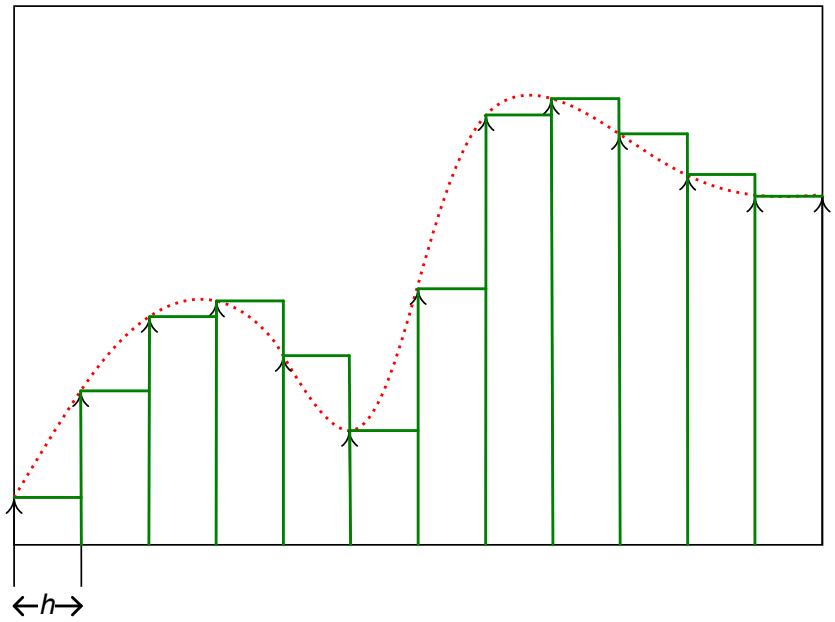
Kuva 8. Taaksepäin derivoinnin periaate

Taaksepäin derivoinnissa vähennetään kaksi eri näytettä toisistaan ja tulos jaetaan näytteenottovälillä. Taaksepäin derivoinnille on voimassa yhtälö

$$y[n] = \frac{x[n] - x[n-h]}{h}\,, \tag{14}$$

missä h on näytteenottoväli (merkitään usein myös T), $x[n]$ on näyte ajanhetkellä t ja $x[n-h]$ on näyte ajanhetkellä $t-h$. Mitä pienempi näytteenottoväli h on, sitä tarkemmin derivaatan arvo saadaan määritettyä. Eräs toinen derivaatan approksimaatio on eteenpäin derivointi eli Eulerin menetelmä. Kuten taaksepäin ja eteenpäin derivoinnissakin, useimmat derivaatan approksimaatiot on johdettu derivaatan määritelmistä. [3.]

Integraalia voidaan approksimoida taaksepäin suorakaidesäännöllä, eteenpäin suorakaidesäännöllä tai trapetsoidisäännöllä, jossa käytetään edellisten approksimaatioiden keskiarvoa. Trapetsoidisääntöä kutsutaan digitaalisen säädön yhteydessä Tustinin approksimaatioksi. Integraalin approksimointi eteenpäin suorakaidesäännöllä on esitetty kuvassa 9. [3.]



Kuva 9. Integraalin approksimointi eteenpäin suorakaidesäännöllä

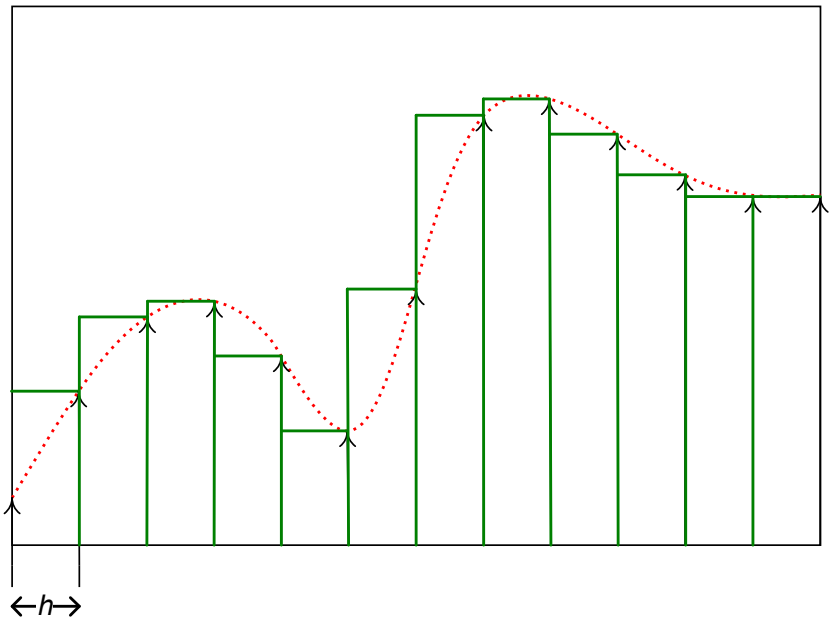
Näytteenottopisteistä piirretään suora seuraavaan näytteenottopisteeseen saakka. Menetelmää kutsutaan Eulerin menetelmäksi ja sille on voimassa yhtälö [3.]

$$y[n] = \sum_{i=0}^{n-1} x[i] \cdot h \,. \tag{15}$$

Toinen menetelmä integraalin approksimointiin on käyttää taaksepäin suorakaidesääntöä. Taaksepäin suorakaidesäännölle on voimassa yhtälö

$$y[n] = \sum_{i=0}^{n} x[i] \cdot h \,. \tag{16}$$

Kuten yhtälöstä (16) voidaan havaita, erona on vain, että näytteenottopisteestä piirretään suora taaksepäin edelliseen näytteenottohetkeen kuvan 10 mukaisesti. [3.]

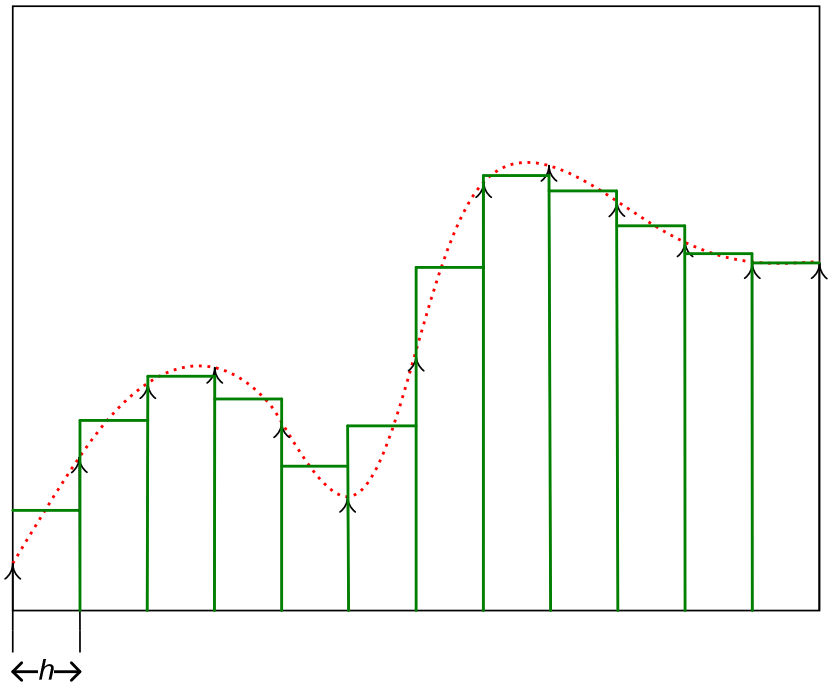


Kuva 10. Integraalin approksimointi taaksepäin suorakaidesäännöllä

Tustinin approksimaatiossa otetaan edellisten menetelmien approksimaatioista keskiarvo. Trapetsoidisäännölle eli Tustinin approksimaatiolle on voimassa yhtälö

$$y[n] = \frac{1}{2} \cdot \sum_{i=0}^{n-1} x[i] \cdot h + \frac{1}{2} \cdot \sum_{i=0}^{n} x[i] \cdot h = \frac{1}{2} \cdot \sum_{i=0}^{n} ((x[i] + x[i-1]) \cdot h) \,. \tag{17}$$

Käytännössä yhtälö (17) tarkoittaa, että lasketaan nykyisen ja edellisen näytteenottohetken keskiarvo, kerrotaan näytteenottovälillä h ja lisätään summaan. Tustinin approksimaatio on esitetty kuvassa 11. [3.]



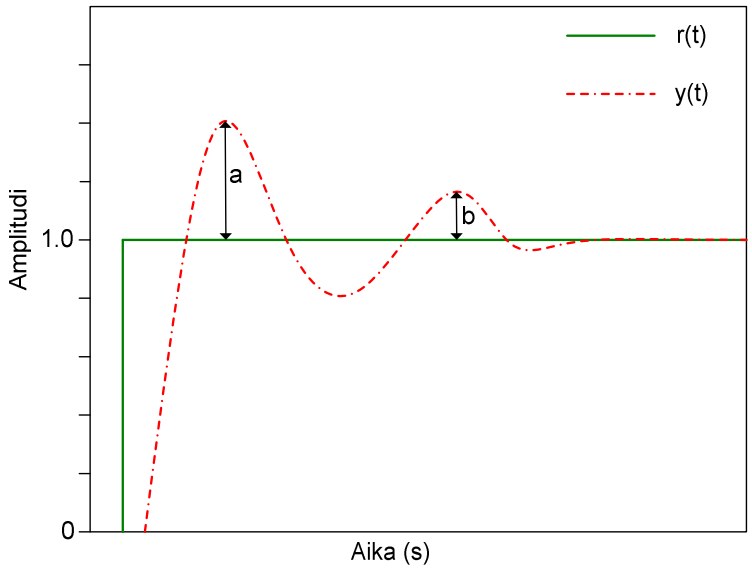
Kuva 11. Tustinin approksimaatio

2.5 Säätimien virittäminen

Säätimen oikeanlaisen toiminnan kannalta on tärkeää, että parametrit viritetään oikein prosessin ja toimintatavan mukaan. Tavoitteena virittämisessä on saada säätöpoikkeama mahdollisimman pieneksi. Virittäminen voi tapahtua joko laskennallisesti tai kokeellisesti. Laskennallisessa virittämisessä prosessin mallista lasketaan sopivat parametrit siirtofunktion tai taajuusvasteen perusteella. Kokeellisesti parametrit valitaan määriteltyjen taajuusvasteiden perusteella, askelvastekokeella, tutkimalla prosessin ominaisvärähtelyjä tai kokeilemalla arvoja parametreille, kunnes säätö toimii oikein. [4, s. 5.]

Säätöpiirin virittämisen keskeisinä tavoitteina on saada riittävä säädön tarkkuus, nopeus ja/tai kuormitushäiriöiden kompensointikyky, ohjaussignaalin rauhallinen käyttäytyminen sekä epäherkkyys mittauskohinaa ja parametrimuutoksia vastaan. Vaikka säädöt ovat prosessiohjauksen perusta, niistä noin kolmasosa on väärin viritettyjä ja vain noin 20 prosenttia toimii hyvin. Muita ongelmia väärän virityksen lisäksi ovat huono toimilaitteen toimintakyky sekä väärä prosessisuunnittelu tai säätöstrategia. [9, s. 12.]

Säätöparametrien optimiarvojen määrittämiseksi on olemassa useita eri kriteereitä, joista hyvin käytäntöön soveltuva on Ziegler-Nicholsin kriteeri (ZN). Siinä kriteerinä on, että lopullisessa suljetussa järjestelmässä säädin reagoi prosessiin kuvan 12 mukaisesti. ZN-kriteerissä a:n ja b:n optimaalinen suhde on 4.

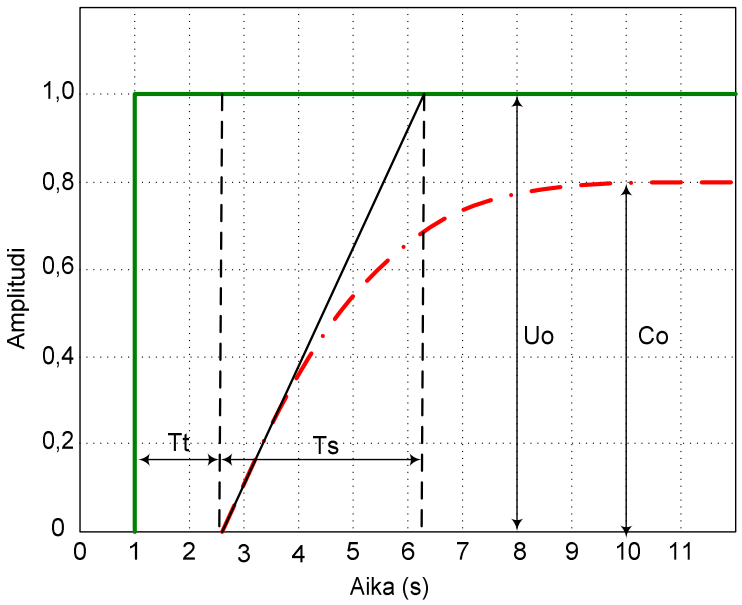


Kuva 12. ZN-kriteerin mukaan viritetyn säätöjärjestelmän askelvaste

Askelvastekoe

Askelvastekokeessa aiheutetaan askelmainen muutos prosessiin säätimen ollessa käsiajolla. Samalla prosessin toiminnasta kerätään ohjaus- ja mittaussignaalia niin kauan, kunnes prosessi on asettunut uudelle tasolle. [9, s. 16.] Mitta-

ustuloksista muodostetaan kuvan 13 mukaiset kuvaajat. Seuraavaksi säädetyn suureen kuvaajaan piirretään jyrkimmän kohdan suuntainen tangentti. Tämän avulla lasketaan nousuaika T_S ja viive T_T. Lisäksi määritellään askelvasteen (sisäänmenevän signaalin) ja säädetyn suureen amplitudit. Amplitudien määrittämisessä on huomioitava, että molemmat signaalit ovat samassa skaalassa. [10, s. 8.]



Kuva 13. Säätimen viritys askelvastemenetelmällä [10, s. 8]

Kuvan suureiden avulla voidaan laskea parametrien arvot eri kriteerien perusteella. ZN-kriteerin perusteella kuvan suureista saadaan seuraavat säännöt säätimen parametrien virittämiseksi:

P-säädin: $$K_P = \frac{T_S U_O}{T_T C_O} \tag{14}$$

PI-säädin: $$K_P = 0{,}9\frac{T_S U_O}{T_T C_O} \text{ ja } T_I = 3T_T \tag{15}$$

PID-säädin: $$K_P = 1{,}2\frac{T_S U_O}{T_T C_O},\ T_I = 2T_T \text{ ja } T_D = 0{,}5T_T. \tag{16}$$

2.6 Digitaalisen säätimen toteutusvaihtoehdot

Digitaalinen säätöjärjestelmä voidaan toteuttaa esimerkiksi prosessorilla tai ohjelmoitavalla logiikalla. Prosessorilla säätöjärjestelmä toteutetaan ohjelmallisesti käyttämällä esimerkiksi C- tai Assembly-kieliä. Tällaisessa säätöjärjestelmässä tarvitaan paljon prosessorin suoritusaikaa, koska prosessorilla menee yhden laskutoimituksen suorittamiseen monta kellojaksoa. PC-pohjaisissa järjestelmissä tarvitaan myös reaaliaikainen käyttöjärjestelmä. Järjestelmissä, kuten esimerkiksi robotin ohjaus, prosessorin suoritustehoa tarvitaan myös muihin laitteen toimintoihin. [6, s. 70].

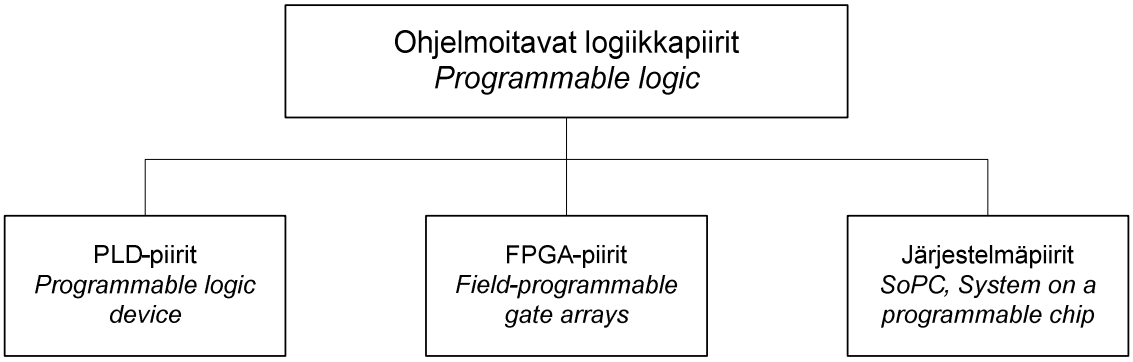
Nykyisin ohjelmoitavilla piireillä voidaan toteuttaa useita monimutkaisia laskutoimituksia rinnakkaisesti saman kellojakson aikana. Se miten laskutoimitukset suoritetaan, riippuu suunnittelijasta ja käytettävistä ohjelmistoista. FPGA-piirien suunnitteluun tarvitaan usein paljon laajempia ja monimutkaisempia ohjelmistoja kuin prosessorien ohjelmointiin. Lisäksi FPGA-piirin suunnittelu on yleensä vaikeampaa kuin prosessorin ohjelmointi. FPGA-toteutuksessa suunnittelijan tulee kiinnittää huomiota laskuoperaatioiden suorittamiseen ja lukujärjestelmien käsittelyyn, kun taas prosessori ja kääntäjä huolehtivat näistä itse. Säätimen algoritmi voidaan toteuttaa FPGA:lla, mutta usein lisäksi tarvitaan prosessoria muiden toimintojen suorittamiseen, kuten säätimen ohjaamiseen ja parametrien välittämiseen käyttäjältä.

Uusissa järjestelmäpiireissä on samaan piiriin integroitu prosessori, FPGA-piiri, monipuoliset I/O-ominaisuudet sekä data- ja ohjelmamuistia. Tällaiset piirit yhdistävät molempien tekniikoiden edut ja usein laitteen koko digitaaliosa voidaan hoitaa yhdellä mikropiirillä. Järjestelmäpiirit ovat tehokkaita signaalinkäsittelyssä, jolloin ne sopivat hyvin myös säätimen toteutukseen.

3 OHJELMOITAVAN LOGIIKKAPIIRIN OHJELMOINTI

3.1 Ohjelmoitavat piirit

Ohjelmoitava logiikkaverkko on piiri, joka sisältää suuren määrän piirielementtejä ja johtimia niiden välillä. Piirielementtien välisiä kytkentöjä ei ole tehty vaan ne ovat ohjelmoitavissa tiettyyn rajaan asti. Ohjelmoitavat logiikkapiirit jaetaan kolmeen pääryhmään kuvan 14 mukaisesti. [11, s. 306, 308.]



Kuva 14. Ohjelmoitavien logiikkapiirien ryhmittely [11, s.308]

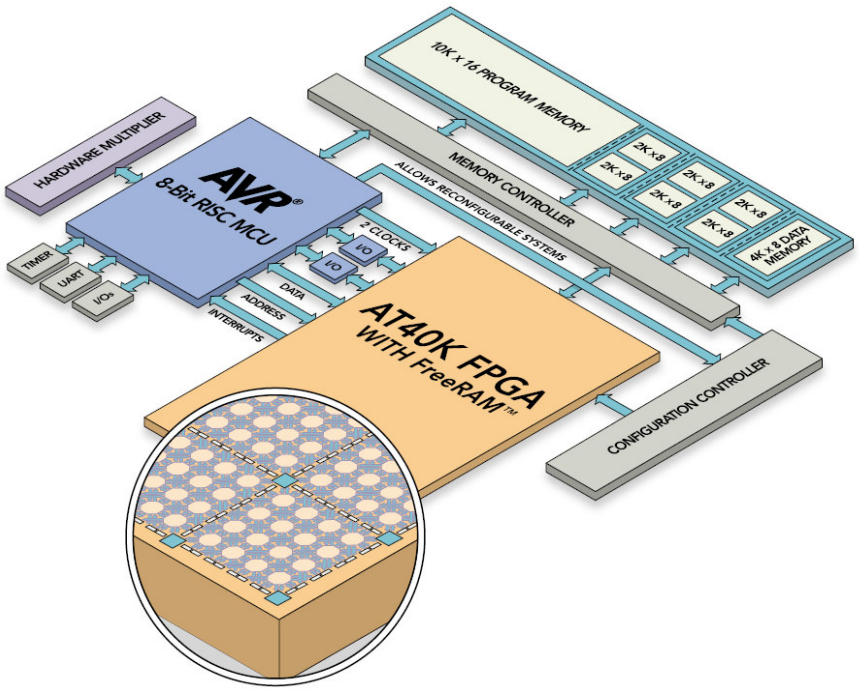
PLD-piirit ovat vanhimpia ohjelmoitavia logiikkapiirejä. Ne olivat aluksi pieniä ja yksinkertaisia, mutta nykyään niitä on saatavana paljon suurempina ja monimutkaisempina. PLD-piirit jaetaan usein vielä kahteen pienempään ryhmään niiden rakenteen mukaan: SPLD- (Simple PLD) ja CPLD (Complex PLD) -piireihin. [11, s. 308.]

FPGA-piirit ovat PLD-piirejä uudempia [11, s. 308]. Ne poikkeavat CPLD-piireistä ainoastaan logiikkasolujen toiminnan osalta. CPLD-piirit luokitellaankin usein FPGA-piireiksi tai toisinpäin. FPGA-piirit muodostuvat ohjelmoitavien logiikkasolujen verkosta CPLD-piirien tapaan. Solut ovat yleensä pienempiä verrattuna CPLD-piireihin, jolloin pinta-ala saadaan käytettyä tarkemmin. Solujen toimintaperiaate vaihtelee valmistaja- ja piirikohtaisesti, mutta ne voivat koostua esimerkiksi D-kiikuista, joiden edessä on staattiseen RAM-muistiin (SRAM) perustuvia taulukoita (LUT, look-up table). Niihin voidaan ohjelmoida solulle tulevi-

en signaalien logiikkafunktioita. Myös solujen hienorakeisuus vaihtelee valmistajakohtaisesti. Joillakin soluilla voidaan toteuttaa hyvin monimutkaisia funktioita, kun taas toiset solut sisältävät vain muutaman logiikkaportin. [12, s. 13.]

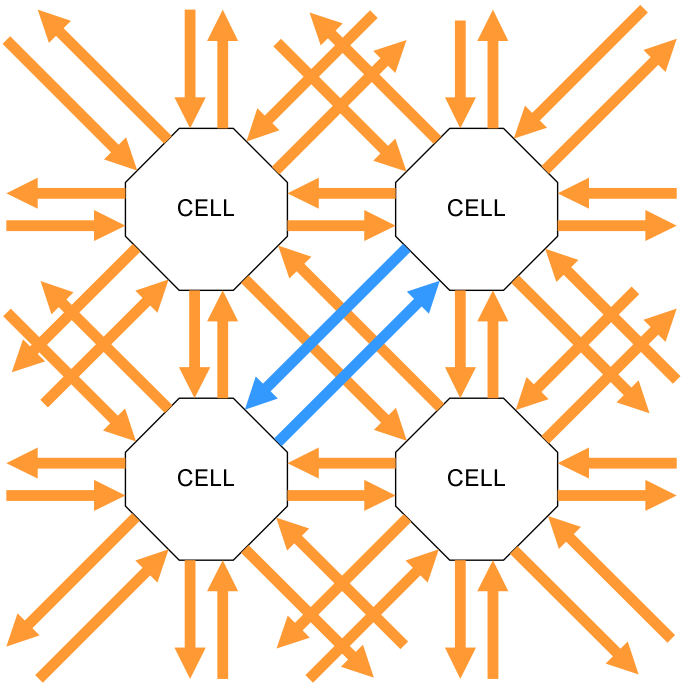
Ohjelmoitavat järjestelmäpiirit ovat suuria ja monimutkaisia piirejä, joissa on suuren FPGA-lohkon lisäksi prosessori, muistia, liitäntäpiirejä ja muita erikoislohkoja. Järjestelmäpiirin prosessori voi olla kiinteä, kuten FPSLIC-piirissä, tai se voi olla FPGA:lla toteutettu ns. softcore-prosessori, joita voi olla useampia yhdellä järjestelmäpiirillä. Järjestelmäpiiri mahdollistaa usein koko digitaaliosan toteutuksen yhdellä piirillä. [11, s. 306]

Atmel valmisti ensimmäisen kenttäohjelmoitavan yhden sirun järjestelmäpiirin, joka on ohjelmoitavissa dynaamisesti järjestelmän sisällä [13]. Atmelin FPSLIC-piirit (Field Programmable System Level Integrated Circuit) sisältävät mallista riippuen 5 k - 40 k ohjelmoitavaa porttia, 8-bittisen AVR-suorittimen ja 36 kilotavua FPGA:n ja suorittimen yhteistä data- ja ohjelmamuistia. Lisäksi piiri sisältää FPGA:n kanssa yhteisiä kello-, data-, osoite-, keskeytys- ja I/O-linjoja. Piirin rakenne on esitetty kuvassa 15. [14.]



Kuva 15. Atmelin valmistaman FPSLIC-piiriperheen rakenne [14]

Piirin tehokulutus on pieni, joten se soveltuu hyvin käytettäväksi langattomissa ja paristokäyttöisissä sovelluksissa. Atmel käyttää piirisarjassa logiikkasolujen yhdistämiseen AT40K-arkkitehtuuria, jossa jokainen solu on yhteydessä kaikkiin kahdeksaan naapurisoluun (kuva 16). Arkkitehtuurin etuna on nopea tiedonvälitys solulta toiselle, jolloin viiveet ovat pieniä ja piirin tehonkulutus saadaan määritettyä tarkasti. [14.]

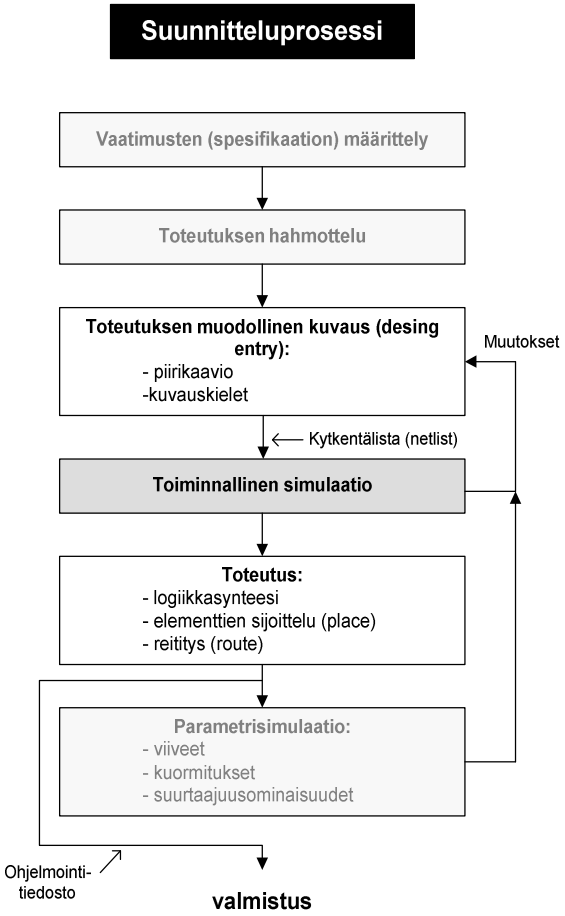


Kuva 16. Atmelin AT40K-arkkitehtuurin rakenne [14]

FPGA-piirien käyttö on järkevää vain pienissä tuotantoerissä. Suurin osa FPGA-piirin pinta-alasta on käytetty ohjelmoitavuuden aikaansaamiseksi. Jos laitetta valmistetaan yli 10 000 kappaletta vuodessa, on edullisempaa toteuttaa logiikka sovelluskohtaisella integroidulla piirillä eli ASIC:lla (Application Specific Integrated Circuit). Siinä koko pinta-ala käytetään pelkästään sovelluskohtaisen logiikan toteuttamiseen. ASIC-piirin kertaluonteiset suunnittelukustannukset ovat kymmenien tai satojentuhansien eurojen luokkaa, mutta piirin valmistuskustannukset ovat suurissa erissä edullisemmat kuin FPGA-piirien. ASIC-piirin huonona puolena on sen kiinteä ratkaisu tiettyyn tiedonkäsittelytehtävään. [12, s. 13.]

3.2 Ohjelmoitavien piirien suunnitteluprosessi

Ohjelmoitavia logiikkapiirejä ei voida käyttää ilman asianmukaisia suunnitteluohjelmistoja. Ohjelmistoja piirien suunnitteluun tarjoavat piirien valmistajat sekä niihin erikoistuneet suunnitteluohjelmistojen tuottajat. Ohjelmoitavien piirien suunnitteluprosessi muistuttaa hyvin paljon ASIC-piireissä käytettyä prosessia, joten myös suunnittelutyökalut muistuttavat hyvin paljon toisiaan. [15, s. 49.] FPGA-piirin suunnitteluprosessi on kuvan 17 mukainen.



Kuva 17. FPGA-piirin suunnitteluprosessi [15, s. 51]

FPGA-piirin suunnittelu eroaa ASIC-piirin suunnittelusta vasta aivan loppuvaiheessa. FPGA-piiri on valmistettu ja ainoastaan piirin solujen väliset yhteydet ovat määrittämättä. FPGA-piirin suunnittelutyön tuloksena saadaan ohjelmointitiedosto, jota käytetään piirin ohjelmointiin. SRAM-tyyppiset FPGA-piirit voidaan

ohjelmoida useita kertoja, mutta tiedot on syötettävä uudelleen aina piirin käynnistyksen yhteydessä. Antisulaketyyppiset FPGA-piirit voidaan ohjelmoida vain kerran ja niiden ohjelmointiin tarvitaan erillinen ohjelmointilaite. Niissä tieto säilyy virtojen katketessakin. Antisulaketyyppisten piirien kohdalla suunnittelun verifiointi onkin paljon tärkeämpää, koska piirit ovat kertakäyttöisiä. [15, s. 49.]

Ensimmäinen vaihe suunnittelutyössä on suunniteltavan piirin kuvaaminen. Kuvaus tapahtuu joko graafisesti piirikaavioesityksenä tai kuvauskielillä. Kuvauskielien käyttö piirien kuvauksessa on uudempi menetelmä ja muistuttaa paljon normaaleja ohjelmointikieliä. Formaalisten kuvauskielien käytöllä on helppo kuvata toimintoja, mutta signaalit näkyvät paremmin piirikaavioesityksissä. Piirikaaviokuvien ongelmana kuitenkin on, että usein ne eivät ole siirrettävissä toiseen suunnitteluohjelmistoon tai piirille. Standardoiduilla VHDL- ja Verilog-kielillä laaditut kuvaukset sen sijaan ovat usein siirrettävissä pienin muutoksin piiristä ja suunnitteluohjelmistosta toiseen. [15, s. 50.]

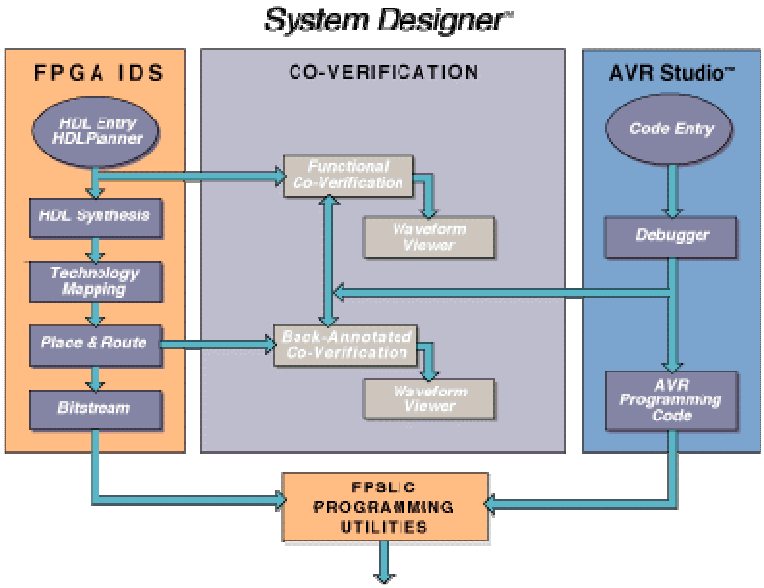
Toisessa suunnitteluvaiheessa simuloidaan ja varmistetaan piirin vaatimusten mukainen toiminta. Piirin kuvauksessa tulee helposti virheitä, joten varsinkin ASIC-suunnittelussa piirien verifiointi on hyvin tärkeää niiden prototyyppikerrosten kalliin hinnan vuoksi. FPGA-piireissä ennakkoverifiointi ei ole yhtä tärkeää ja ne voidaan testata suoraan lopullisessa sovelluksessa, koska piirit voidaan ohjelmoida uudelleen ilman lisäkustannuksia. Poikkeuksena tietenkin ovat antisulake-tyyppiset FPGA-piirit, jotka voidaan ohjelmoida vain kerran. [15, s. 50.]

Kolmantena vaiheena piirin kuvauksen ja toiminnallisen tarkistuksen jälkeen on logiikkasynteesi. Sen tarkoituksena on luoda piirin ohjelmointia varten tarvittava tiedosto. Synteesi on automaattinen toiminto, johon määritellään parametrit ennen sen käynnistämistä. Logiikkasynteesissä suunnitteluohjelmien välisenä erona on tehokkuus, joka pääasiassa tarkoittaa sitä, miten pienellä logiikkaporttien määrällä haluttu toiminto saadaan suoritettua. Tässä kohtaa piirikaavioesityksestä yleensä saadaan parempi tehokkuus kuin laitteistonkuvauskielillä, koska piirikaavioesitys perustuu valmistajan omiin, optimoituihin makroihin. [15, s. 50 - 51.]

Lisäksi suunnitteluprosessissa voidaan tehdä ajoitussimulaatio. Sen tarkoituksena on testata, toimiiko piiri riittävän nopeasti eri tilanteissa. Ajoitussimulaatio on välttämätöntä ASIC-piireille, mutta FPGA-piireillä se voidaan jättää pois pieniä järjestelmiä suunniteltaessa. FPGA-piiri voidaan testata lopullisessa sovelluksessa ja ohjelmoida uudelleen, mikäli ongelmia esiintyy. Ajoitussimulaation käyttö suuremmissa järjestelmissä kannattaa, koska virheellisen toiminnan syyn selvittäminen on helpompaa ajoitussimulaatiosta kuin piirilevyltä. [15, s. 51.]

System Designer 3.0

FPSLIC-piirien suunnittelutyökaluna käytetään System Designer -ohjelmistoa (kuva 18). Ohjelmisto sisältää tarvittavat työkalut piirin FPGA-lohkon suunnitteluun ja ohjelmointiin. Ohjelman mukana ei toimiteta kääntäjää piirillä sijaitsevalle prosessorille. Suunnitteluprosessi etenee aiemmin kuvatulla tavalla ja logiikkasynteesin tuloksena saadaan tiedosto, jolla piirin FPGA-lohko voidaan ohjelmoida. Laitteen ohjelmointivaiheessa voidaan lisätä prosessorille käännetty ohjelma ja piirille käännetty logiikka, jolloin molemmat ohjelmoidaan piirille yhtä aikaa. [14.]



Kuva 18. System Designer -ohjelmisto [14]

3.3 Laitteistonkuvauskieli, HDL

Laitteistonkuvauskieli eli HDL (Hardware Description Language) on kehitetty kuvaamaan digitaalipiiriä tai -laitetta tekstimuotoisesti. Se muistuttaa rakenteeltaan ja esitystavaltaan hyvin paljon tavanomaisia ohjelmointikieliä. Erona on, että laitteistonkuvauskielellä kuvataan laitteiston samanaikaista ja peräkkäistä toimintaa, kun ohjelmointikieli kuvaa pelkästään prosessorin ajallisesti peräkkäin suoritettavia toimintoja. Laitteistonkuvauskielellä tehtyä kuvausta voidaan sellaisenaan käyttää laitteiston simulointiin tai siitä voidaan laatia suunnitteluohjelmiston avulla uusi kuvaus. [11, s.356.]

Laitteistonkuvauskieliä on kehitetty useita, koska aluksi ohjelmoitavien piirien valmistajat kehittivät omia kuvauskieliä, joita voitiin käyttää vain kyseisen valmistajan piireille. Tällaisia kieliä ovat ABEL, PALASM, CUPL ja AHDL. Laitteistonkuvauskielien valmistajariippuvuuden vuoksi kaksi laitteistonkuvauskieltä standardisoitiin: VHDL (VHSIC Hardware Description Language) ja Verilog. Standardoinnilla saavutettiin se, että millä tahansa tekstieditorilla muodostettu laitteistonkuvauskieli oli siirrettävissä suunnitteluohjelmistosta toiseen ja usein myös toiselle piirille. [11, s. 365 - 366.]

Verilog HDL

Verilogin syntaksi muistuttaa hyvin paljon C-kielen syntaksia. Tämän takia sen oppiminen on yleensä helpompaa kuin VHDL:n, koska C-kieltä käytetään paljon laitteiden ohjelmoinnissa. Verilog on esitelty ensimmäisen kerran vuonna 1985 ja se standardisoitiin 1995. [16, s.4.]

Verilog-kielessä digitaalinen järjestelmä kuvataan moduuleina. Moduulien välisissä liitynnöissä on kuvattu, miten eri moduulit ovat yhteyksissä toisiinsa. Moduulille määritellään moduulin nimi, sisään- ja ulostuloportit sekä kaksisuuntaiset inout-portit. Moduuli ja sen toiminta kuvataan aina module- ja endmodule-lauseiden väliin. Module-lauseessa annetaan moduulille nimi ja sulkeiden sisään määritellään porttien nimet. Moduulin sisällä oleva rakenne voidaan suo-

rittaa kerran (*initial-rakenne*) tai se voidaan suorittaa jokaisella kello- tai sisääntulosignaalin laskevalla tai nousevalla reunalla (*always-rakenne*). Moduulin rakenne on seuraava:

```
module <moduulin nimi> (<porttien listaus>)
<määrittelyt>
<moduulin toiminnot>
endmodule
```

Moduulin nimeämisen jälkeen kirjoitetaan kunkin portin suunta (in, out tai inout) ja leveys bitteinä. Samalla tavalla määritellään myös moduulissa käytettävät rekisterit. Alla on esimerkki kolmen eri rekisterin määrittämisestä.

```
reg [7:0] A, B;
reg [0:7] C;
```

Ylempi lause määrittelee rekisterit A ja B, joiden molempien leveys on 8 bittiä ja eniten merkitsevä bitti on kahdeksas bitti. Myös rekisteri C on 8 bittiä leveä, mutta sen eniten merkitsevä bitti on alin bitti (bitti 0). [16, s. 8 - 9.]

Verilog sisältää samoja rakenteita kuin C-kieli, mutta niiden suoritus ei välttämättä tapahdu samoin rinnakkaisten operaatioiden vuoksi. Esimerkiksi for-lause suoritetaan Verilogissa yhden kellojakson aikana. Alla on esimerkki Kertolaskumoduulista, joka kertoo kaksi 4-bittistä rekisteriä ja päivittää ulostulon arvon jokaisella laskevalla kelloreunalla.

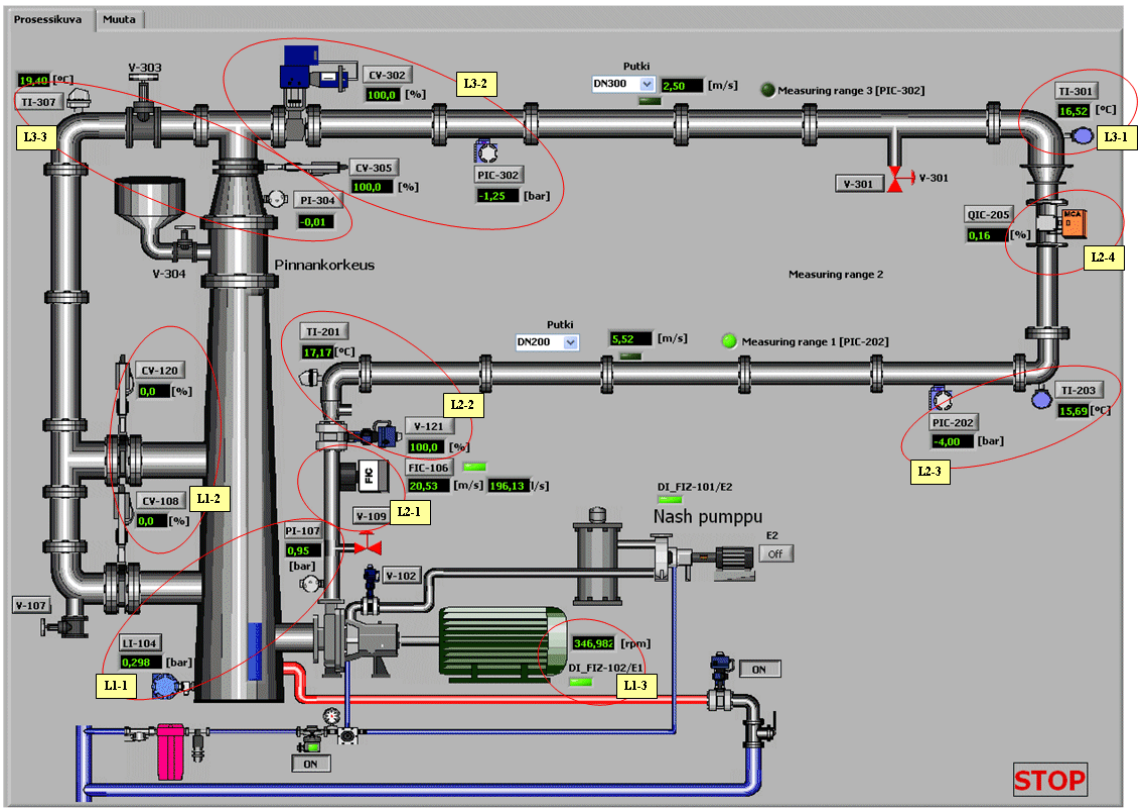
```
module Kertolasku (lukuA, lukuB, Tulos, clk);
 input clk;
 input [3:0] lukuA, lukuB;
 output [7:0] Tulos;
 reg [7:0] Tulos;

  always @(negedge clk)
   begin
    Tulos <= lukuA * lukuB;
   end
```

4 JÄRJESTELMÄ

4.1 Paperimassan kierrätyslaitteisto

Paperimassan kierrätyslaitteisto koostuu massaputkistosta, toimilaitteista, antureista ja mittaus- ja ohjausjärjestelmästä. Kuvassa 19 on esitetty paperimassan kierrätyslaitteisto. Lisäksi kuvassa on esitetty mittaukset ja ohjaukset sekä mittausantureiden ja toimilaitteiden jako langattomille lähettimille.



Kuva 19. Paperimassan kierrätyslaitteisto (kuva kierrätyslaitoksen käyttöliittymästä)

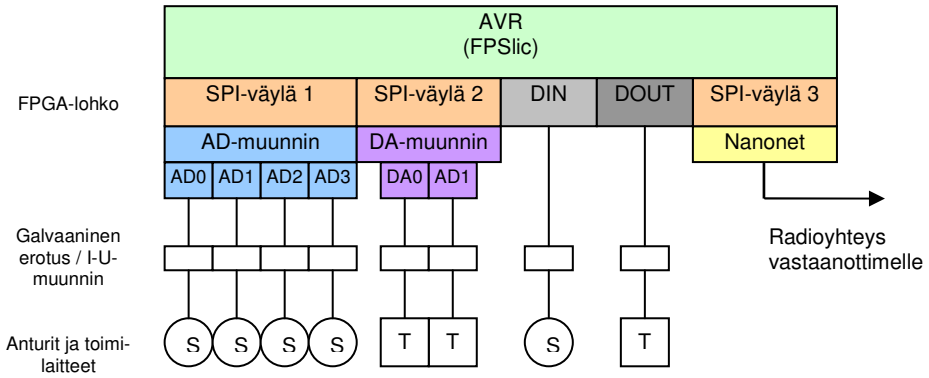
Kierrätyslaitteisto luo mahdollisuudet korkeatasoiseen paperimassan ja prosessimittalaitteiden tutkimiseen. Langallinen mittaus- ja ohjausjärjestelmä perustui National Instrumentsin compactFieldPoint-tuotteeseen. Langaton mittaus- ja ohjausjärjestelmä on rakennettu langallisen rinnalle. Kierrätyslaitos oli rakennettu kolmeen eri kerrokseen, joissa jokaisessa oli oma järjestelmäsolmunsa. Yhden järjestelmäsolmun alla oli useita eri prosessisolmuja.

4.2 Langaton mittaus- ja ohjausjärjestelmä

Langattoman mittaus- ja ohjausjärjestelmän rakenne on esitetty liitteenä (liite A). Prosessisolmu on osa suurempaa langatonta mittaus- ja ohjausjärjestelmää. Prosessisolmut mittaavat järjestelmässä antureilta saatavan signaalin, muokkaavat mittaustiedon mitattavaa suuretta vastaavaksi ja lähettävät tiedon langattomasti järjestelmäsolmuille. Järjestelmäsolmuilta tieto siirretään tietokantaan. Ohjaustieto välitetään käyttäjältä järjestelmäsolmulle ja siitä edelleen langattomasti prosessisolmulle.

4.3 Prosessisolmun rakenne

Prosessisolmussa käytettiin myös Atmelin FPSLIC-piiriä. Käytetyn piirin tyyppi oli AT94S40 SecureFPSLIC. Piiri voidaan suojata asettamalla suojausbitti, jolloin ohjelmointi tai ohjelman lukeminen piiriltä ei onnistu. Piiri reagoi ainoastaan alustuskäskyyn, jolloin piiri saadaan tyhjennettyä ja ohjelmoitua uudelleen. Prosessisolmuun liitetään anturit sekä ohjattavat toimilaitteet. Prosessisolmun rakenne on esitetty kuvassa 20.



Kuva 20. Langaton mittaus- ja ohjausmoduuli (prosessisolmu)

5 SÄÄTÖJÄRJESTELMÄN TOTEUTUS KEHITYSALUSTALLA

5.1 Säätimen osittelu piirin eri osiin

Ennen säädön toteutusta tutkittiin, miten FPGA-lohkoa voitaisiin käyttää hyväksi säätimen toteutuksessa. Prosessorin suoritusaikaa oli käytetty mittaus- ja ohjausjärjestelmässä jo muiden toimintojen suorittamiseen. Tämän takia ohjelmallisen toteutuksen vaarana oli, että säätöjärjestelmän reaaliaikaisuus menetettäisiin. AT94K40-piireissä on 40 000 ohjelmoitavaa porttia, joista osa oli käytetty SPI-väylien tiedonsiirtoon. Mahdollisina esteinä algoritmin toteutukseen FPGA-lohkolla olivat laskutoimituksien suorittaminen logiikalla sekä sopisiko algoritmi kokonaan FPGA-lohkolle.

PID-algoritmin toteutuksessa tarvitaan paljon laskuoperaatioita. Siksi oli tärkeää tutkia, kuinka eri laskutoimitukset tehdään piirin logiikalla ja Verilog HDL-kielellä. Vähennyslaskuissa tuli ottaa huomioon, että vähennettäessä pienemmästä luvusta suurempi tulos piti käsitellä kahden komplementin lukujärjestelmässä. Esimerkiksi jos vähennetään nelibittiset luvut 0011_B (luku 3 desimaalijärjestelmässä) ja 0100_B (luku 4 desimaalijärjestelmässä) keskenään, saadaan tuloksena 1111_B. Normaalissa binaarijärjestelmässä luku olisi 16, mutta käsiteltäessä lukua kahden komplementtina edellä saatu tulos on oikein (luku -1 desimaalijärjestelmässä).

Verilogilla kahden komplementin lukua ei voida suoraan kertoa toisella luvulla. Tämä johtuu siitä, että laskutoimituksissa luvut oletetaan positiivisiksi, jolloin tulokseksi tulee väärä (vrt. edellinen esimerkki). Ennen kertolaskun suorittamista luku pitää muuttaa positiiviseksi ja laittaa merkkibitti muistiin myöhempiä käsittelyjä varten. Kahden komplementti -esitystavassa luvun etumerkki saadaan muutettua komplementoimalla sen bitit ja lisäämällä tulokseen 1. Näin luvusta saadaan positiivinen ja se voidaan kertoa. Kertolaskun jälkeen luku täytyy muuttaa takaisin kahden komplementiksi, että tulos olisi oikein tai etumerkki tulee huomioida myöhemmissä yhteen- ja vähennyslaskuissa.

Kahden komplementin -esitystapaa voidaan kuvata taulukon 1 esimerkillä. Siinä nelibittisen kahden komplementin lukujärjestelmää on verrattu desimaalijärjestelmän lukuihin. Kuten taulukosta huomataan, nelibittisessä kahden komplementin luvussa 0000_B on nolla desimaalijärjestelmässäkin, ja luvut nollasta seitsemään on esitetty normaalin binäärijärjestelmän tavoin. Negatiivisilla luvuilla eniten merkitsevä bitti (MSB) on aina 1, ja suurin negatiivisesti esitettävä luku nelibittisellä luvulla on 1000_B, joka vastaa lukua -8 desimaalijärjestelmässä. Näin ollen kahden komplementin esitystavassa negatiivisia lukuja on aina yksi enemmän kuin positiivisia.

Taulukko 1. Nelibittisen kahden komplementin -esitystavan luku desimaalijärjestelmässä

kahden komplementti	desimaalijärjestelmän luku
0111	7
…	…
0010	2
0001	1
0000	0
1111	-1
1110	-2
…	…
1000	-8

Kahden komplementin binaariluvusta saadaan muodostettua desimaaliluku normaalin binaarijärjestelmän tapaan, mutta ensimmäisen bitin arvo vähennetään muiden bittien arvosta. Esimerkiksi luku 1011_B muutetaan kahden komplementin järjestelmästä desimaaliksi seuraavasti:

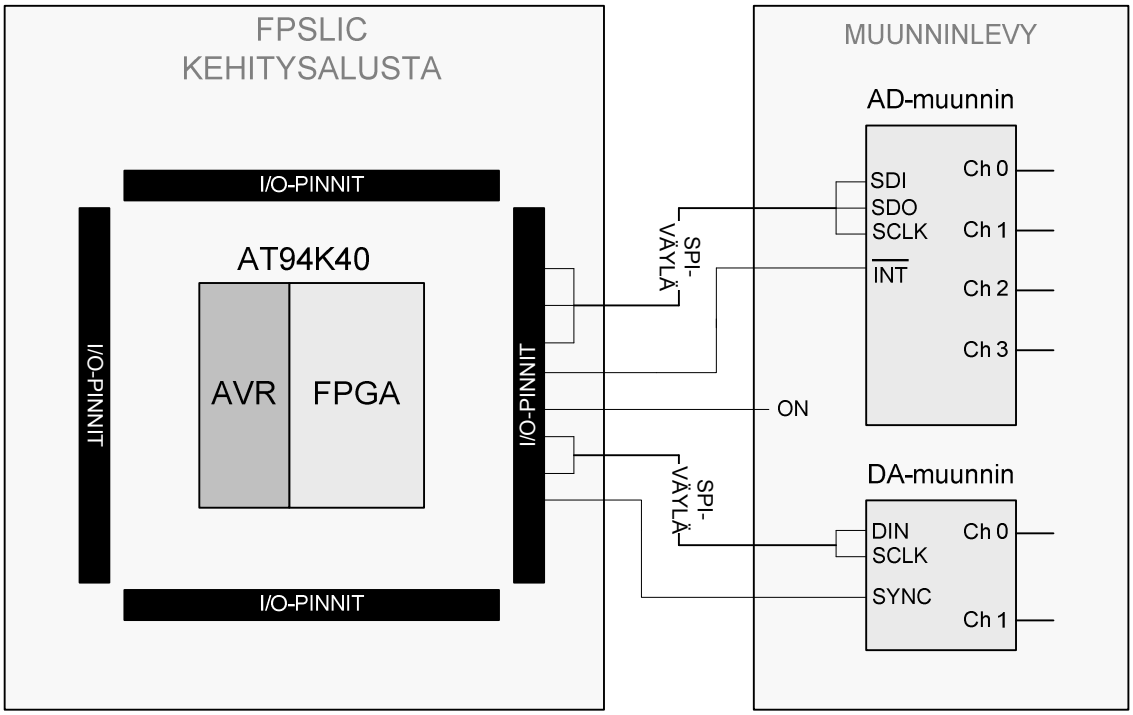
$$1011_B = -1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = -8 + 0 + 2 + 1 = -5_{10}.$$

Laskutoimitukset onnistuivat, kunhan negatiiviset luvut vaihdettiin positiivisiksi ennen kertolaskua ja se huomioitiin myöhemmissä yhteen- ja vähennyslaskuissa. Lisäksi PID-säätimen arveltiin sopivan kokonaan piirin FPGA-lohkoon. Näin

ollen säätimen toteutus jaoteltiin piirin eri osiin siten, että PID-algoritmi toteutettiin kokonaan FPGA:lla ja säätimen toimintaa ohjattiin prosessorin avulla.

5.2 Muunninkortin kytkeminen kehitysalustaan

Säätöalgoritmin kehitykseen käytettiin Atmelin ATSTK94-kehitysalustaa, joka sisälsi FPSLIC AT94K40 -piirin, I/O-liitännät piirille, painokytkimiä ja ledejä sekä tarvittavat välineet piirin ohjelmointiin. Kehitysalustan I/O-pinneihin kytkettiin Mittalaitelaboratorion valmistama piirilevy, joka sisälsi AD- ja DA-muuntimet sekä digitaalisia I/O-liitäntöjä. Säätimen toteutuksessa tarvittiin ainoastaan piirilevyn AD- ja DA-muuntimia. AD-muunnin oli 14-bittinen Texas Instrumentsin valmistama TLC3544 ja DA-muunnin 12-bittinen Analog Devicesin AD5322. Antureiden kytkemistä varten AD-muuntimessa oli käytössä 4 kanavaa ja toimilaitteita vasten DA-muuntimella 2 kanavaa. Lisäkortin kytkeminen kehitysalustalle on esitetty kuvassa 21.



Kuva 21. Kehitysalustan ja muunninlevyn väliset kytkennät

Pinnien kytkentäjärjestyksellä kehitysalustalle ei ollut väliä, kunhan käytettiin FPGA-piirin I/O-pinnejä. Porttien signaalit määriteltiin myöhemmin suunnitteluvaiheessa.

5.3 Logiikan suunnittelu

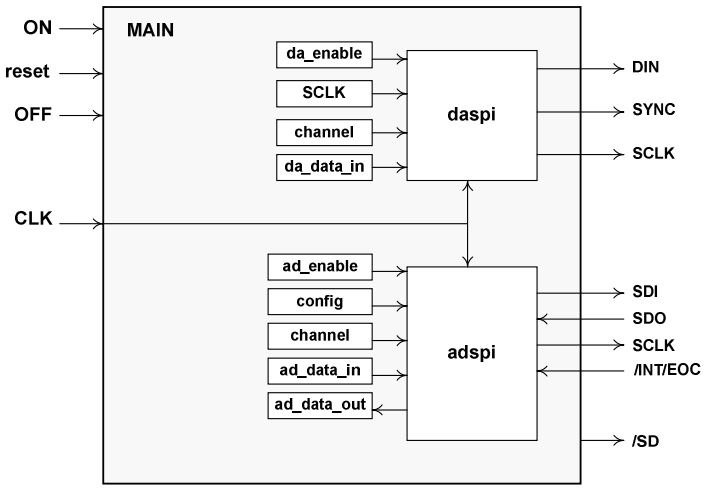
Työssä oli käytössä kehitysalustan ja lisäkortin väliseen tiedonsiirtoon valmiit rajapinnat, joita käytettiin apuna myös säätimen toteutuksessa. Tiedonsiirtorajapintojen kuvaukseen oli käytetty Verilog HDL -kieltä. FPSLIC-piirien suunnitteluun kehitetty System Designer -ohjelmisto tukee VHDL- ja Verilog-kieliä, muttei molempia yhtä aikaa. Tämän takia myös säätimen toteutuksessa käytettiin Verilogia logiikan suunnittelussa. Logiikan suunnittelu eteni teoriassa esitetyllä tavalla.

System Designerissa logiikan kuvaaminen aloitettiin luomalla uusi projekti, johon lisättiin SPI-väylien tiedonsiirtomoduulit sisältävät tiedostot. Tiedonsiirtorajapintoja ja järjestelmän toimintaa ohjattiin Main-moduulista. Projektiin luotiin yksi moduuli lisää, johon suunniteltiin säätimen toiminta. Tiedostoista tehtiin synteesi Leonardo Spectrum -ohjelmalla. Synteesin jälkeen määriteltiin prosessorin ja FPGA-lohkon väliset signaalit, kuten esimerkiksi yhteinen kellosignaali. Synteesin tuloksena saatu tiedosto avattiin Figaro IDS -ohjelmalla, jossa aluksi määriteltiin, mihin I/O-portteihin piirin ulkoiset signaalit kytketään. Ohjelman tuloksena saatiin tiedosto, jolla piirin FPGA-lohko voitiin ohjelmoida. Jos piirin prosessoria olisi käytetty järjestelmässä, prosessorille käännetty tiedosto olisi lisätty ohjelmoinnin yhteydessä.

SPI-väylien rajapinnat

AD- ja DA-muuntimien sekä FPGA-piirin väliseen tiedonsiirtoon käytettiin SPI-väyliä. Muuntimet tarvitsivat toimiakseen SPI-väylän lisäksi joitakin ohjauslinjoja. Kuvauksessa oli käytetty kolmea eri moduulia, joista main-moduulissa oli määritelty piirin ulkoiset signaalit kuvan 21 mukaisesti. Main-moduulista ohjattiin kahden alemman moduulin, adspi ja daspi, toimintaa. Piirin sisääntuloihin oli

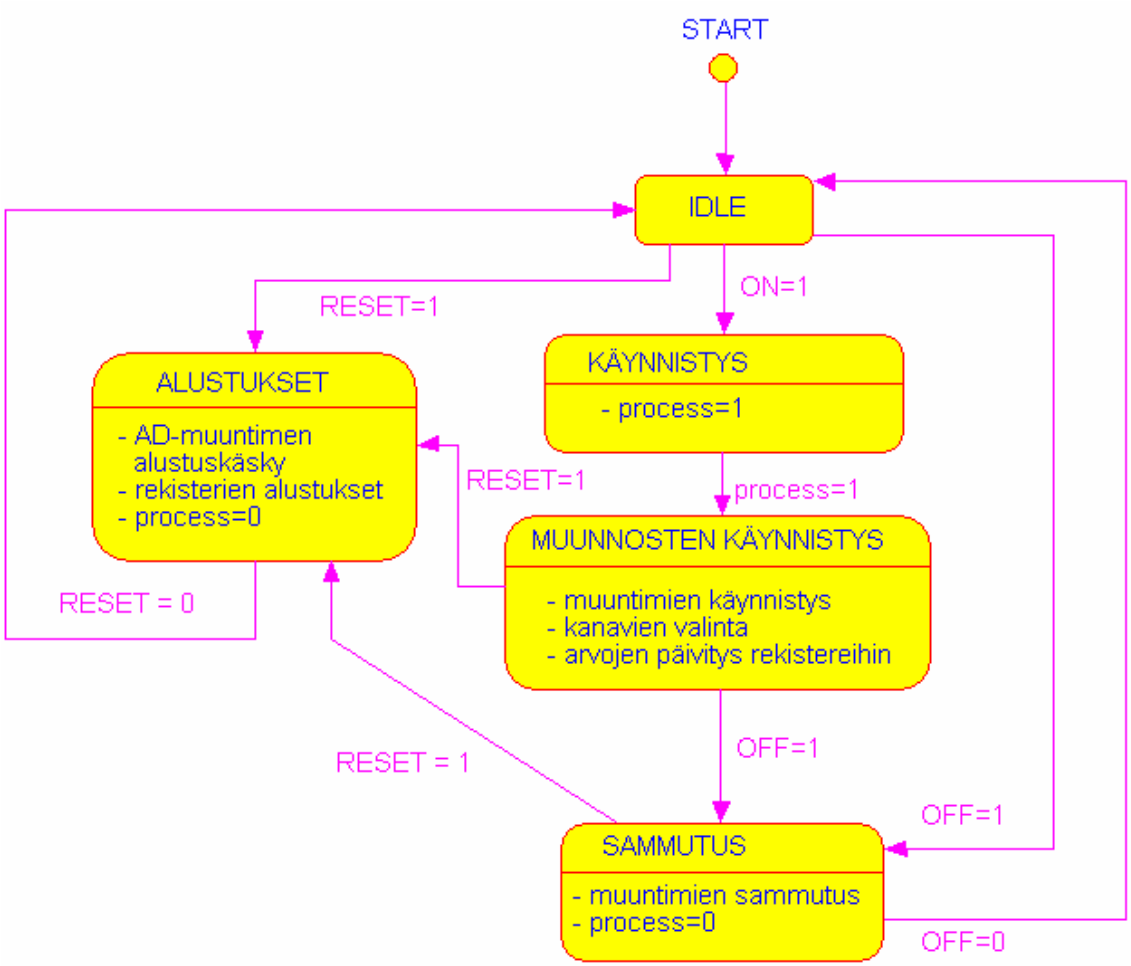
määritelty kolme kytkintä (ON, OFF ja reset), joilla voitiin käynnistää, alustaa ja sammuttaa muuntimet. Moduulien sisäiset ja ulkoiset signaalit on esitetty kuvassa 22.



Kuva 22. Moduulien sisäiset ja ulkoiset liitynnät

AD-muuntimen käynnistämiseen tarvittiin alustuskäsky, joka lähetettiin piirille alustettaessa järjestelmä reset-kytkimellä. Muunnos käynnistettiin adspi-moduulissa automaattisesti edellisen valmistuttua. Muunnoksen käynnistäminen tapahtui 16-bittisellä datasanalla, jossa valittiin mitattava kanava sekä muita muunnokseen liittyviä parametreja. Samalla kun 16-bittinen datasana lähetettiin SDI-nastan kautta muuntimelle, saatiin edellisen muunnoksen tulos yhtä aikaa SDO-nastasta. Muunnoksen valmistumisesta muunnin ilmoitti asettamalla keskeytyslinjan nollatilaan, jonka jälkeen kaksisuuntainen sarjamuotoinen tiedonsiirto voitiin aloittaa. SPI-väylän kello saatiin järjestelmän kellosta, joka yhdistettiin prosessorilta FPGA:lle. Kun tiedonsiirto oli valmis, päivitettiin da_data_out-rekisteriin mitattu arvo. AD-muuntimen kanava valittiin main-moduulista 2-bittisellä ad_channel-rekisterillä. Jos kanavaa ei muutettu muunnosten välissä, luettiin sama kanava uudelleen. AD-muuntimen alustus tapahtui asettamalla main-moduulissa config-rekisterin arvoksi 1. Tällöin muuntimelle lähetetään alustuskäskyä niin kauan, kunnes config-rekisteri asetetaan takaisin nollaan.

DA-muunninta ei tarvinnut alustaa erillisellä alustuskäskyllä, vaan kaikki tarvittava tieto siirrettiin 16-bittisessä datasanassa, joka sisälsi kanavan valinnan (bitti 15), referenssimoodibitin (bitti 14) ja bitit käytettävälle moodille (bitit 13 ja 12) sekä muunnettavan arvon 12-bitin tarkkuudella (bitit 11 - 0). Muunnin käynnistettiin main-moduulista asettamalla da_enable-rekisterin arvoksi 1. Kanava valittiin asettamalla da_channel-rekisterin arvoksi 0 tai 1 käytettävän kanavan mukaan. Da_data_in-rekisteriin asetettiin muunnettava 12-bittinen arvo, joka lähetettiin DA-muuntimelle seuraavan datasanan lähetyksen yhteydessä. Daspi-moduulissa koottiin kaikki rekisterit yhdeksi 16-bittiseksi datasanaksi ennen lähetystä. Muunninta käytettiin perusmoodissa, jolloin kaikki DA-muuntimen moodibitit asetettiin automaattisesti nollaksi. Main-moduulin tilakaavio on esitetty kuvassa 23.

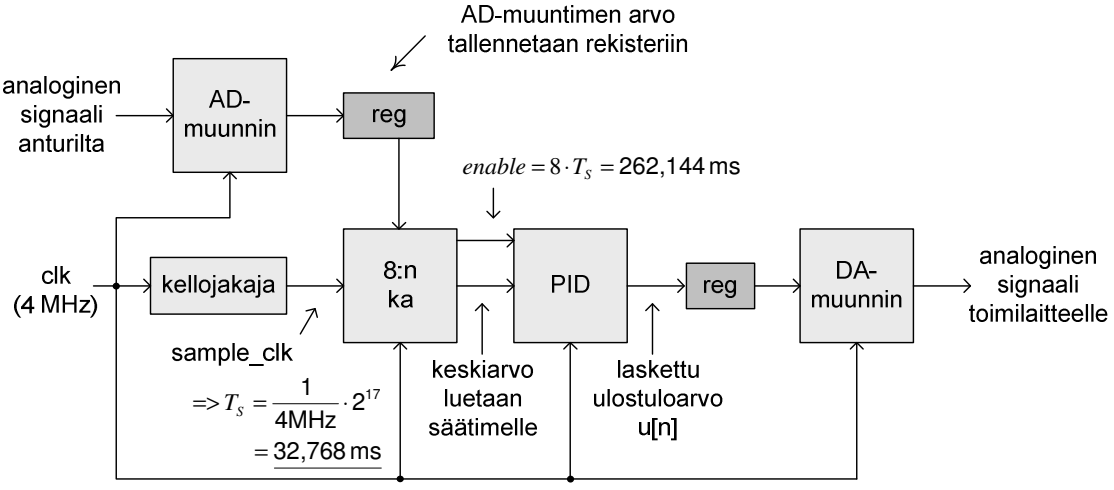


Kuva 23. Main-moduulin tilakaavio

Säätöalgoritmin toteutus

Säätimeksi toteutettiin PID-säädin teoriassa esitetyn yhtälön (9) mallin mukaisesti. Kyseisen algoritmin etuna on, että sillä voidaan helposti toteuttaa mikä tahansa PID-säätimen moodi (P, PI, PD, PID) asettamalla K_I tai K_D nollaksi. Kuten yhtälöstä havaitaan, käytöstä poistetun termin kertolaskun tuloksena saadaan nolla, jolloin se ei vaikuta säätimen ulostuloon. Integroinnin summalausekkeen päivittäminen ei ollut tässä tapauksessa ongelma, koska laskenta oli tehokasta logiikan avulla. Toisaalta mitä enemmän monimutkaisia laskutoimituksia tehdään, sitä enemmän tarvitaan piirin pinta-alaa. Pinta-alaa tarvitaan myös sitä enemmän, mitä enemmän toimintoja suoritetaan rinnakkain. Yhtälön (9) mukaisen lausekkeen tuloksena saatiin tieto, mihin suuntaan ja minkä verran ulostuloa pitää säätää. Tämän takia algoritmin tulos täytyi lisätä edelliseen ulostuloarvoon $u[n-1]$.

Säätimen toimintaa mallinnettiin lohkokaaviolla, johon kuvattiin eri osien toiminnat sekä niiden väliset yhteydet. Säätimen toiminnallinen lohkokaavio on esitetty kuvassa 24.



Kuva 24. Säätimen lohkokaavio

Logiikalla säätimen ulostulo saadaan laskettua muutaman kellojakson aikana. Hitaissa prosesseissa näin nopea säätöarvon laskeminen on kuitenkin turhaa.

Lisäksi analogisen signaalin muuntamiseen meni aikaa AD-muuntimella. Säätimen hidastamista varten tehtiin kellojakaja, jonka tahdissa otettiin näytteitä keskiarvon laskentaa varten. Keskiarvon laskemisella vähennettiin mittaussignaalissa esiintyvää kohinaa. Kellojakaja toteutettiin 17-bittisen rekisterin avulla, jonka arvoa vähennettiin yhdellä jokaisella systeemikellon nousevalla reunalla. Kun rekisterin arvo oli nolla, saatiin laskurin ulostulona yhden kellojakson mittainen pulssi näytteenottoa varten. Kun kahdeksan näytettä oli saatu, aloitettiin säätimen ulostuloarvon laskeminen. 17-bittisellä rekisterillä ja 4 MHz:n kellolla saatiin kellojakajasta ulos noin 33 ms näytteenottotaajuus. Kun näytteitä otettiin kahdeksan kappaletta 33 ms näytteenottotaajuudella, käynnistettiin PID-säädin noin 260 ms välein.

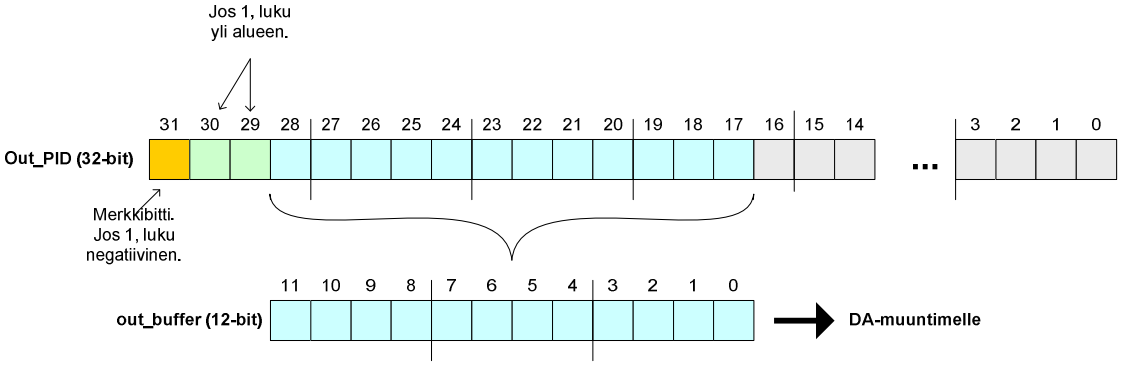
Aluksi säätimellä laskettiin erosuure, joka saatiin asetusarvon ja mittaustuloksista muodostetun keskiarvon erotuksesta. Seuraavalla kellojaksolla erosuuretta integroitiin ja derivoitiin. Integroinnin approksimaatiossa käytettiin teoriassa esitetyn yhtälön (17) mukaista trapetsoidisääntöä, eli Tustinin approksimaatiota (kuva 11). Yhtälön (17) mukaisesti laskettiin edellisen ($e[n-1]$) ja nykyisen erosuureen ($e[n]$) keskiarvo ja lisättiin se Int_sum-rekisteriin. Derivaatan approksimoinnissa käytettiin teoriassa esitetyn yhtälön (14) mukaista taaksepäin derivointia. Koska näytteet olivat peräkkäisiä, vähennettiin nykyisestä erosuureesta edellisen erosuureen arvo.

Seuraavaksi tutkittiin, oliko edellä saaduissa tuloksissa negatiivisia lukuja. Tätä varten tarkkailtiin tuloksien eniten merkitsevää bittiä (MSB). Jos MSB-bitti oli 1, luku oli negatiivinen kahden komplementin luku. Jokaisen kohdalla ylimmän bitin arvo tallennettiin rekisteriin myöhempää tarvetta varten. Samalla negatiivisista kahden komplementin luvuista muodostettiin positiiviset komplementoimalla rekisterin bitit ja lisäämällä tulokseen 1. Kaikista negatiivisista tuloksista komplementoidut arvot tallennettiin omiin rekistereihinsä, etteivät alkuperäiset tulokset muuttuneet.

Aiemmin laskettu erosuure, integroinnin summa ja derivaatta tai niiden negatiivisista arvoista muodostettu arvo kerrottiin kukin omalla vahvistuksellaan K_P,

K_I ja K_D. Case-rakenteen avulla tutkittiin luvun negatiivisuutta 1-bittisillä rekistereillä, jotka oli muodostettu lukujen MSB-biteistä. Jokaisen termin (P, I ja D) kohdalla tutkittiin, oliko luku negatiivinen. Jos oli, vahvistus kerrottiin sen positiivisella arvolla, joka oli tallennettu aiemmin eri rekisteriin kuin itse tulos. Muussa tapauksessa käytettiin lukujen alkuperäisiä arvoja. Tulokset lisättiin tai vähennettiin, etumerkistä riippuen, edelliseen laskettuun säätimen ulostuloarvoon $u[n-1]$.

Lopuksi tutkittiin, oliko saatu tulos määritettyjen rajojen sisässä. Kertolaskujen tuloksena saadusta 32-bittisestä ulostulorekisteristä voitiin käyttää vain 12 bittiä DA-muuntimelle. Tätä varten määriteltiin 12-bittinen out_buffer-rekisteri, jonka arvo päivitettiin aina, kun uusi ulostulon arvo oli laskettu. Alimmat bitit karsittiin pois ja ylempien bittien avulla tutkittiin, oliko luku alueen sisäpuolella (kuva 25).



Kuva 25. Laskettu arvo skaalataan DA-muuntimelle

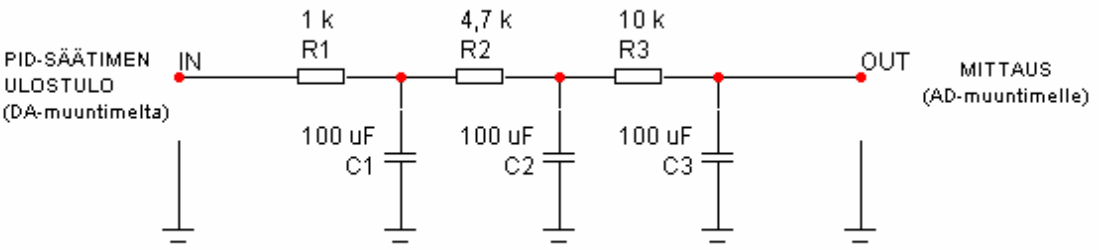
Jos MSB-bitti (bitti 31) oli 1, saatu tulos oli negatiivinen ja ulostulon arvoksi (out_buffer) täytyi antaa 000_H. Samalla 32-bittisen Out_PID-rekisterin arvoksi annettiin 00000000_H. Jos biteistä 29 ja 30 toinen tai molemmat olivat ykkösiä, luku oli negatiivinen ja ulostulorekisterin arvoksi annettiin FFF_H ja OutPID-rekisterin arvoksi $1FFE0000_H$. Luvun ollessa alueen sisäpuolella out_buffer-rekisteri sai Out_PID-rekisterin bittien 28…17 arvon.

Kun ulostulon arvo oli laskettu ja päivitetty out_buffer-rekisteriin, jäätiin odottamaan, kunnes uudet 8 näytettä oli saatu. Näytteitä otettiin jatkuvasti myös las-

kennan aikana, ja uusi ulostuloarvon laskeminen aloitettiin, kunhan keskiarvo oli saatu laskettua. PID-moduulin tilakaavio on esitetty liitteessä B. Testausta varten säätimeen lisättiin ominaisuus, jolla asetusarvoa voitiin pienentää tai suurentaa askelittain. Tämä tapahtui helposti lisäämällä kaksi kytkintä main-moduulin sisääntuloportteihin ja määrittelemällä PID-moduulissa, että toinen kytkin vähensi asetusarvoa ja toinen lisäsi sitä (ei ole esitetty PID-moduulin tilakaaviossa).

5.4 Testaus

Prosessia simuloitiin kytkennällä, jossa oli kolme peräkkäistä RC-piiriä. Kytkentä reagoi hitaasti vasteen muutoksiin ja näin ollen simuloi hyvin paineen käyttäytymistä hitaissa prosesseissa. Kytkentä on esitetty kuvassa 26.

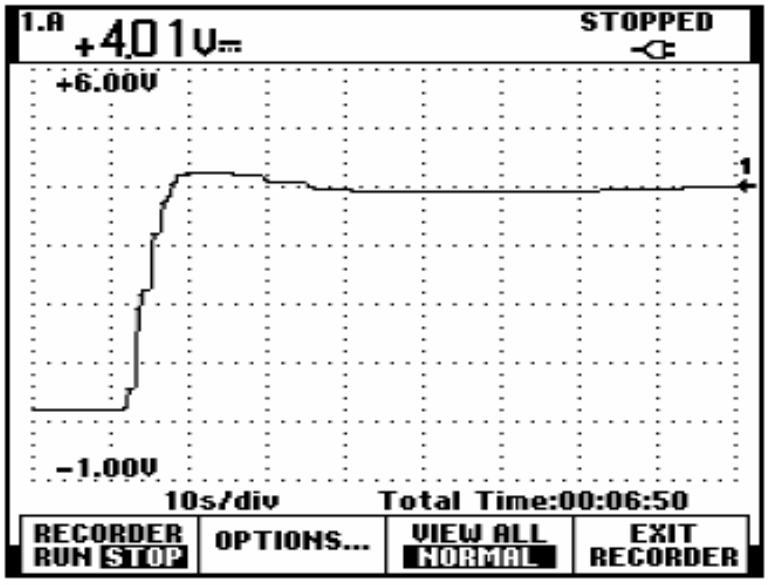


Kuva 26. Prosessin simuloiminen kolmella RC-piirillä

Simulaatioprosessi kytkettiin DA-muuntimen kanavaan 0 ja prosessin suuretta mitattiin AD-muuntimen kanavalla 0. Oskilloskooppi kytkettiin säätimen ulostuloon (DA-muuntimen kanava 0), jolla voitiin tutkia säätimen askelvastetta kyseisessä ”prosessissa”.

Säätimen virittäminen

PID-säätimen parametrit viritettiin kokeellisesti. Viritys aloitettiin muuttamalla K_P:n arvoa K_I:n ja K_D:n ollessa nolla. K_P:n arvo säädettiin sellaiseksi, että säädin selvästi värähteli askelmaiseen säätöarvon muutokseen, mutta kuitenkin vaimeni sen jälkeen. Seuraavaksi viritettiin parametri K_I, joka tasoitti säätimen värähtelyä ja säädin asettui säätöarvoonsa edellistä nopeammin. Viimeiseksi viritettiin parametri K_D. Derivointitermin käyttöönotolla parannettiin säätimen vastetta. Viritetyn säätimen parametrit olivat K_P=11FF$_H$, K_I=0100$_H$ ja K_D.=2F00$_H$. Viritetyn säätimen askelvaste on esitetty kuvassa 27. Vaikka parametrit asetettiin kokeilemalla, saatiin tulokseksi aika lähelle teoriassa esitetyn PID-säätimen askelvasteen (kuva 7) mukainen säätökäyrä.



Kuva 27. Viritetyn PID-säätimen askelvaste

Säätimen parametrien arvot annettiin, kun järjestelmä alustettiin reset-kytkimellä. Parametreja muutettaessa synteesi sekä logiikan kääntäminen piirin ohjelmointitiedostoksi jouduttiin tekemään uudelleen, jolloin se oli aikaa vievää. Tämän takia parametrien viritys lopetettiin, kun säätimen todettiin toimivan oikein.

6 SÄÄTÖJÄRJESTELMÄN TOTEUTUS PROSESSISOLMUUN

Työssä oleva säädin kehitettiin prosessisolmuun (lähetin L3-2), joka ohjasi kolmannen kerroksen venttiiliä CV-302 (kuva 19). Mitattava paineanturi (PIC-302) oli myös kytketty samaan prosessisolmuun, jolloin mittaus ja säädin olivat saman laitteen alla.

Työssä käytettiin samaa AD- ja DA-muuntimet sisältävää levyä kuin kehitysalustallakin. Levy oli suunniteltu käytettäväksi nimenomaan prosessisolmussa, joten FPSLIC-levy vain painettiin muunninlevyn päälle. Levyt olivat yhteydessä levyjen välillä kiinteästi liittimien kautta, joten oli tärkeää, että signaalit määritettiin oikein logiikan kääntämisvaiheessa.

6.1 Logiikan suunnittelu

Säätimen jatkokehitysvaiheessa säädintä muokattiin siten, että aiemmin kehitetty säädin voitiin lisätä valmiin prosessisolmun FPGA-lohkoon. Samalla mittaustulosten välittäminen prosessisolmulta tietokantaan piti toimi normaalisti. Tarkoituksena oli saada parametrit K_P, K_I ja K_D sekä asetusarvo ja säätimen ohjaukseen tarvittavat bitit langattoman verkon kautta säätimelle. Parametrit välitettiin prosessorilta FPGA-lohkolle dataväylän avulla.

Toteutuksessa käytettiin taas valmista pohjaa, joka erosi hieman kehitysalustan versiosta. Prosessisolmun FPGA-lohkolle oli määritelty SPI-väylien tiedonsiirto, mukaan lukien Nanonet-radion SPI-väylä. Mittaustieto lähetettiin AD-muuntimen SPI-väylän kautta FPGA:lle, ja se tallennettiin FPGA-lohkon ja prosessorin yhteismuistiin. Prosessorilla mittaustulokset luettiin muistista, ja ne skaalattiin vastaamaan mitattavan suureen arvoa. Prosessorilta mittaukset lähetettiin langattoman verkon kautta järjestelmäsolmulle ja sieltä tietokantaan. Toimilaitteiden ohjaustieto välitettiin käyttäjältä tietokantaan, joka lähetti sen järjestelmäsolmulle ja siitä langattomasti prosessisolmulle. Prosessisolmulla ohjausdata välitettiin prosessorilta FPGA-lohkolle dataväylän kautta. Muuntimet oli toteutettu samalla

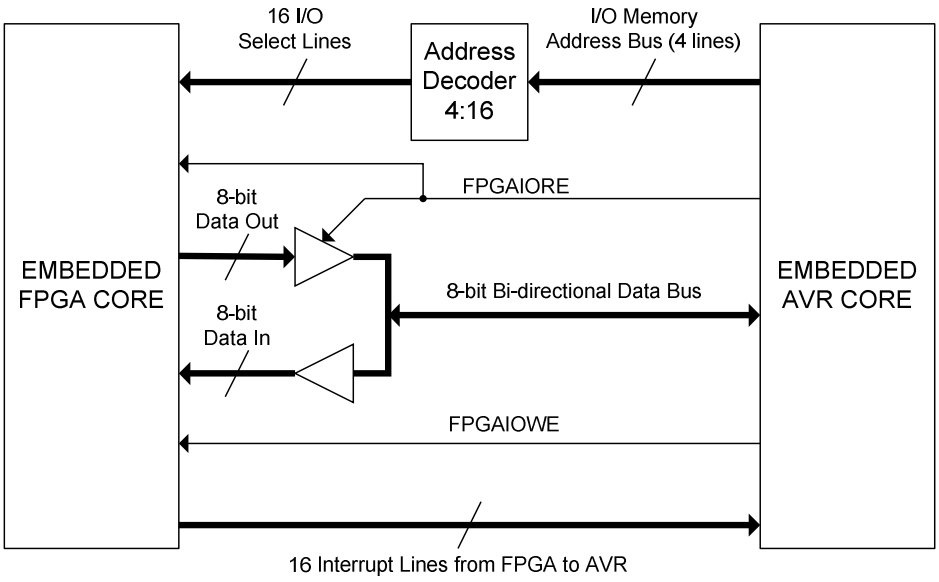
tapaa kuin kehitysalustalla. Ainoastaan AD-muuntimen moduuli erosi siten, että mittaustuloksista muodostettiin keskiarvo jo AD-muuntimen moduulissa.

Säädintä muokattiin siten, että AD-muuntimelta otettiin ainoastaan kahdeksan ylintä bittiä käsiteltäväksi. Tarkoituksena oli pienentää rekisterien kokoa ja näin ollen säästää myös piirin pinta-alaa. Keskiarvon laskeminen poistettiin, koska se oli jo hoidettu AD-muuntimen ohjausmoduulissa. Prosessisolmun kiteen taajuus oli 8 MHz. Säätimen näytteenottoaika lyheni järjestelmän kellotaajuuden kaksinkertaistumisesta ja koska keskiarvoa ei tarvinnut enää laskea. Tämän vuoksi PID-säätimen näytteenottotaajuuden muodostavan laskurin rekisteriä jouduttiin leventämään. Tällä kertaa kello toteutettiin 21-bittisellä rekisterillä, jolloin saatiin näytteenottotaajuudeksi sama 260 ms kuin aiemmin.

Prosessiteollisuuden toimilaitteita ohjataan 4 mA - 20 mA virtaviestillä. Esimerkiksi venttiili on kiinni virran arvolla 4 mA tai vähemmän. Venttiili avautuu, kun virta on suurempi kuin 4 mA ja on sitä enemmän auki, mitä suurempi virta on. Kun ohjausvirran arvo saavuttaa 20 mA rajan, on venttiili täysin auki. Prosessisolmun DA-muuntimelta saatiin 0 - 5 V jännite. Järjestelmän ohjausjännite muutetaan jännite-virta-muuntimella virraksi, jolloin 0 V vastaa 4 mA virtaa ja 5 V 20 mA virtaa. Venttiili siis aukeaa, kun DA-muuntimen ulostulojännitettä kasvatetaan. Prosessissa paine käyttäytyy päinvastoin venttiilin aukioloasentoon nähden. Näin ollen venttiiliä pitää sulkea, että painetta saadaan kasvatettua. Kehitysalustalla kehitetyssä säätimen versiossa ulostulon arvoa eli jännitettä kasvatettiin, kun mitattu jännite oli alle asetusarvon. Tämän takia prosessisolmulle kehitetyssä säätimessä erosuureesta tehtiin sen käänteisarvo komplementoimalla bitit ja lisäämällä tulokseen 1. Tällöin toimilaite säätyi oikeaan suuntaan.

Parametrien välittäminen piirin AVR-mikro-ohjaimelta toteutettiin FPGA-lohkon ja mikro-ohjaimen yhdistävän dataväylän avulla. Lisäksi tarvittiin I/O Select -linjoja, joiden avulla dataväylä kytkettiin FPGA-lohkolla haluttuun sisäiseen porttiin. FPSLIC-piirissä FPGA-lohko ja AVR-mikro-ohjain on yhdistetty dataväylän ja I/O Select -linjojen avulla toisiinsa kuvan 28 mukaisesti. Edellisten lin-

jojen lisäksi käytössä on 16 keskeytyslinjaa FPGA-lohkolta mikro-ohjaimelle sekä kaksi ohjauslinjaa dataväylän suunnan valitsemiseen.



Kuva 28. FPGA-lohkon ja AVR-mikro-ohjaimen välinen liityntä

Ohjauslinjojen FPGAIORE ja FPGAIOWE avulla valitaan dataväylän suunta FPGA-lohkolta. Molempia linjoja ohjataan prosessorilta päin. Kun luetaan FPGA-lohkolta, FPGAIORE-linja menee aktiiviseksi ja 8-bittinen Data Out -linja kytketään prosessorin datalinjalle. Kirjoitettaessa FPGA-lohkolle asetetaan FPGAIOWE-linja aktiiviseksi ja nyt Data In -linja kytkeytyy dataväylään. Ohjauslinjojen käsittely toimii automaattisesti prosessorilta kirjoitettaessa ja luettaessa.

I/O Select -linjat dekoodataan neljästä prosessorilta tulevasta I/O-valintalinjasta. Dekoodauksen jälkeen saadaan käyttöön 16 I/O-valintalinjaa FPGA-lohkon ohjaamiseen. I/O Select -linjat valitaan prosessorilta asettamalla FISCR-rekisterin kahden alimman bitin arvot (XFIS1 ja XFIS0) taulukon 2 mukaisesti.

Bitit XFIS1 ja XFIS0 valitaan prosessorilla kirjoittamalla FISCR-rekisteriin ennen dataväylän lukemista tai siihen kirjoittamista. Tämän lisäksi prosessorin ohjelmoijan tulee tietää, mihin ulostulorekisteriin tieto pitää kirjoittaa.

Esimerkiksi kirjoitetaan FPGA-lohkolle I/O Select -linjan 9 osoittamaan I/O-porttiin. Portin tulee olla määriteltynä FPGA-lohkolla 8-bittiseksi sisääntulolin-

jaksi, mikäli kaikki bitit halutaan käyttää. Aluksi asetetaan ohjauslinjojen arvoiksi XFIS1=0 ja XFIS0=1. Tämä tapahtuu kirjoittamalla prosessorin ohjelmassa FISCR-rekisteriin arvo 01_H. Rekisteriin kirjoittamisen jälkeen I/O Select -linjat 1, 5, 9 ja 13 ovat aktiivisia.

Taulukko 2. I/O select -linjojen valitseminen FISCR-rekisterin kahdella alimmalla bitillä XFIS0 ja XFIS1

Read or Write I/O Address	FISCR Register		FPGA I/O Select Lines			
	XFIS1	XFIS0	15..12	11..8	7…4	3…0
FISUA	0	0	0000	0000	0000	0001
	0	1	0000	0000	0000	0010
	1	0	0000	0000	0000	0100
	1	1	0000	0000	0000	1000
FISUB	0	0	0000	0000	0001	0000
	0	1	0000	0000	0010	0000
	1	0	0000	0000	0100	0000
	1	1	0000	0000	1000	0000
FISUC	0	0	0000	0001	0000	0000
	0	1	0000	0010	0000	0000
	1	0	0000	0100	0000	0000
	1	1	0000	1000	0000	0000
FISUD	0	0	0001	0000	0000	0000
	0	1	0010	0000	0000	0000
	1	0	0100	0000	0000	0000
	1	1	1000	0000	0000	0000

Taulukon 2 avulla nähdään, mihin I/O-osoitteeseen arvo pitää milloinkin kirjoittaa. Kuten taulukosta nähdään, kaikilla ohjausbittien kombinaatioilla on aina jokainen prosessorin I/O-osoite kytkettynä (FISUA..D) yhtä aikaa. Esimerkissä haluttiin kirjoittaa yhdeksännen I/O Select -linjan osoittamaan osoitteeseen, jolloin haluttu arvo pitää kirjoittaa I/O-osoitteeseen FISUC.

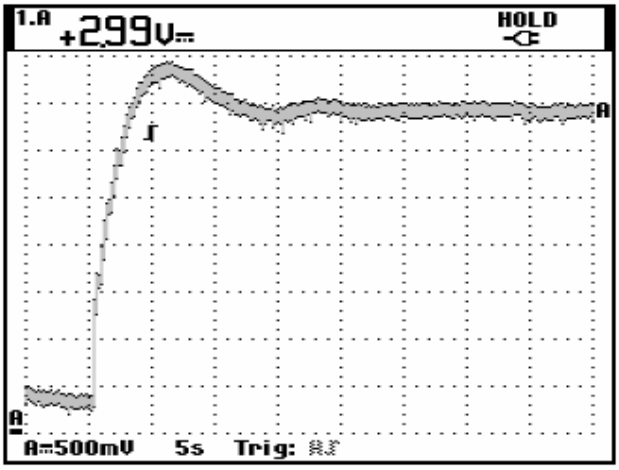
Asetusarvo ja parametrit muutettiin prosessisolmun toteutuksessa 8-bittisiksi. Tämä helpotti dataväylästä lukemista, koska tarvittiin vain yksi I/O-valintalinja parametria kohden. Osa I/O-valintalinjoista oli jo käytössä. PID-säätimen asetusarvo luettiin I/O-osoitteesta 9, kontrollibitit osoitteesta 10 ja parametri Kp, Ki ja Kd osoitteista 12, 13 ja 14. Kontrollirekisterissä sijaitsi neljä bittiä PID-

säätimen ohjaamista varten. Kahdella alimmalla bitillä (bitit 0 ja 1) valittiin AD-muuntimen kanava, josta säädin luki mittaustuloksen. Kolmas bitti (bitti 2) määräsi DA-muuntimen kanavan, johon säätimen toimilaite oli kytketty. Neljännellä bitillä (bitti 3) ohjattiin PID-säädin päälle tai pois päältä. Jos PID-säädin oli pois päältä, DA-muuntimiin kirjoitus toimi normaalisti. PID-säätimen ollessa päällä kirjoitettiin toiseen muuntimen kanavaan PID-säätimen ulostulo ja toiseen kanavaan DA-muuntimelle asetettu arvo. Prosessisolmun main-moduulin tilakaavio on esitetty liitteessä C.

Muutoksia jouduttiin tekemään myös AVR:n ja järjestelmäsolmun ohjelmiin. Kun ohjausdata saapui prosessisolmulle, kirjoitettiin DA-muuntimen arvot ja PID-säätimen parametrit FPGA-lohkolle. Parametrit ja ohjausdata asetettiin web-sivuilla, jotka lähettivät sen Jabber-palvelimen kautta järjestelmäsolmuille ja siitä edelleen prosessisolmuille.

6.2 Testaus

Testaamisessa käytettiin samaa kytkentää kuin kehitysalustalle kehitetyn säätimen testaamisessa. Parametreja muutettiin web-sivuilta ja säädin viritettiin kohdalleen. Prosessisolmun säätimen askelvaste on esitetty kuvassa 29. Parametrien arvot olivat K_P=50, K_I=1 ja K_D=120.



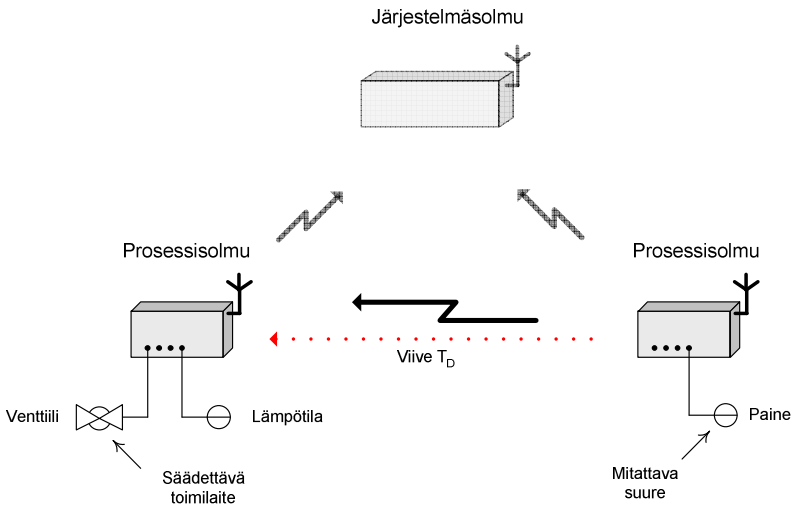
Kuva 29. Prosessisolmun säätimen askelvaste

7 MITTAUSTIEDON VÄLITTÄMINEN LANGATTOMAN VERKON KAUTTA

Työssä tutkittiin myös, miten mittaustieto voitaisiin lähettää langattomasti prosessisolmulta toiselle, säätävälle prosessisolmulle. Eri toteutusvaihtoehdoissa merkitsevimpänä asiana oli viive, joka kului mittaushetkestä siihen, kun se saapui PID-säätimelle. Tutkimuksessa vertailtiin myös, miten suuria muutoksia toteutukset vaatisivat nykyisessä langattomassa mittaus- ja ohjausjärjestelmässä.

Mittaustuloksen lähettäminen suoraan prosessisolmulta toiselle

Eräs ratkaisuista olisi lähettää mittaustulos suoraan prosessisolmulta toiselle kuvan 30 mukaisesti.



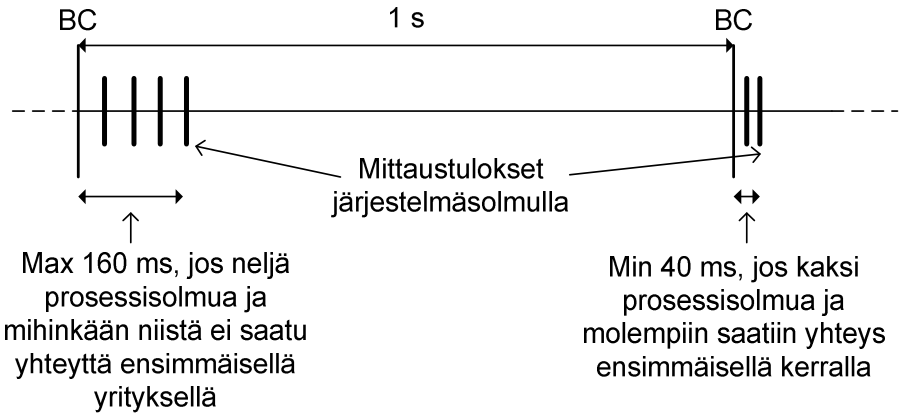
Kuva 30. Mittaustiedon lähettäminen suoraan prosessisolmulta toiselle

Mittaavalta prosessisolmulta mittaustulos välittyy radion kautta suoraan säätävälle prosessisolmulle. Langattomassa mittaus- ja ohjausjärjestelmässä eri kerrosten radioliikenne on estetty, etteivät eri kerrosten radioliikenteet häiritsisi toisiaan. Tämän vuoksi tässä toteutuksessa molempien prosessisolmujen tulisi sijaita samassa kerroksessa tai niiden radioliikenne tulisi olla toteutettu samalla tapaa.

Toteutuksen hyvänä puolena on, että tällä tavalla toteutetussa järjestelmässä päästään todella lyhyeen viiveeseen. Prosessisolmujen välinen radioliikenne on nopeaa, jolloin mittaustulos olisi lähes välittömästi säätävän prosessisolmun käytettävissä. Toteutus nykyiseen järjestelmään on kuitenkin hankalaa. Käyttäjän pitäisi voida valita, miltä prosessisolmulta mittaus lähetetään, ja mikä prosessisolmu sen ottaa vastaan. Toteutus vaatisi suuria muutoksia prosessisolmujen ohjelmassa.

Mittauksen lähettäminen saman järjestelmäsolmun kautta

Mittaustulos voidaan lähettää prosessisolmulta säätävälle prosessisolmulle myös järjestelmäsolmun välityksellä. Nykyisessä järjestelmässä lähetetään broadcast-viesti sekunnin välein prosessisolmuille, jolloin viimeisimmät mittaustulokset otetaan talteen prosessisolmulla (kuva 31). Seuraavaksi aloitetaan tiedonsiirto prosessisolmujen ja järjestelmäsolmujen välillä. Aluksi mittaustulokset haetaan lähettimeltä 1. Jos yhteys lähettimeen onnistui, kysytään seuraavan lähettimen mittaustuloksia 20 ms:n jälkeen. Jos yhteys lähettimeen 1 ei ollut onnistunut, odotetaan vielä seuraavat 20 ms, ja jos vieläkään yhteyttä ei saada, siirrytään seuraavaan lähettimeen ja merkitään lähetin poissaolevaksi.



Kuva 31. Järjestelmäsolmun ja prosessisolmujen radioliikenne ajallisesti esitettynä

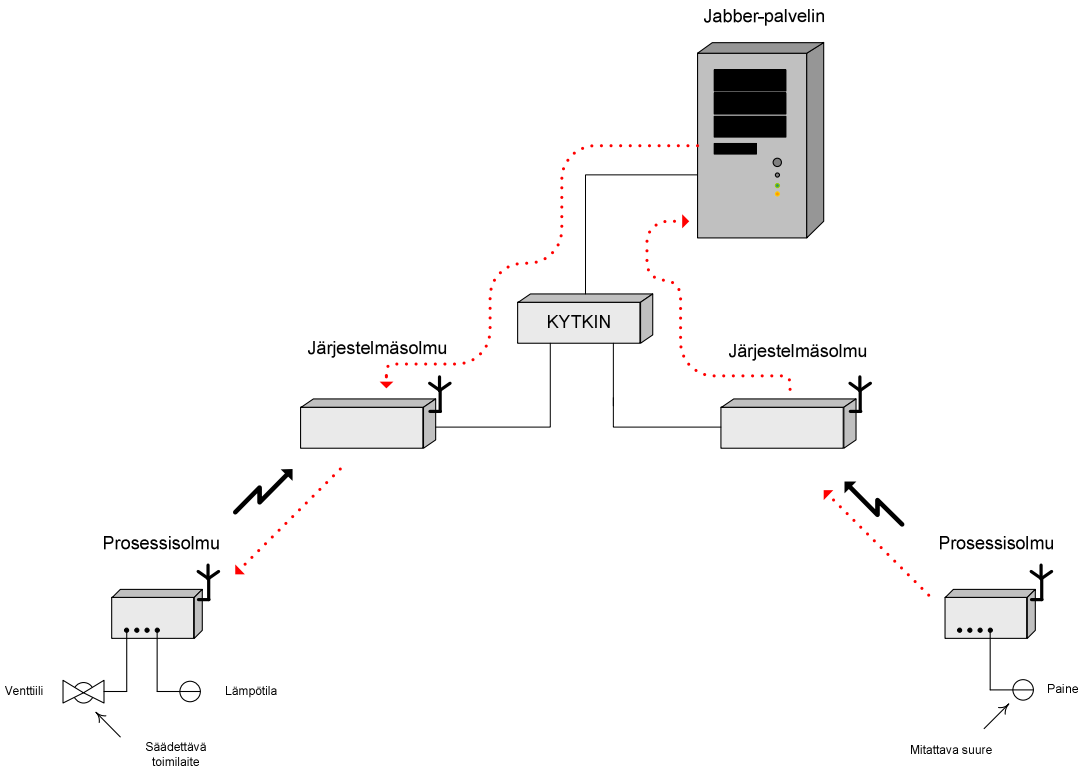
Mittaustuloksella menee maksimissaan 160 ms, kun prosessisolmun mittaustulos on järjestelmäsolmulla, jos käytössä on neljä prosessisolmua. Minimissään päästään 80 ms viiveeseen, jos käytössä on kaksi prosessisolmua ja molempiin saadaan yhteys ensimmäisellä yrityksellä. Järjestelmäsolmulta mittaustulos lähetetään prosessisolmulle heti sen saavuttua. Viivettä voitaisiin vielä lyhentää, jos mitattava suure olisi lähettimellä 1 ja säädin lähettimellä 2, jolloin mittaustulos voitaisiin lähettää lähettimelle 2 heti, kun se on luettu lähettimeltä 1. Nykyisessä järjestelmässä uudet mittaukset luetaan sekunnin välein. Muuttamatta tätä aikaa voitaisiin säädin käynnistää sekunnin välein.

Tämä toteutus vaatisi paljon muutoksia järjestelmäsolmun ohjelmassa. Lisäksi käyttöalue rajautuu yhteen kerrokseen.

Mittauksen lähettäminen Jabber-palvelimen kautta

Jabber-palvelin on standardin XMPP-protokollan mukaisen Jabber-asiakkaiden välisen viestiliikenteen reitittäjä. Jabber-palvelin pitää yllä tietoa siihen kytkeytyneiden asiakkaiden tilasta. Koska järjestelmäsolmut ovat Jabber-asiakkaita, myös niiden kesken voidaan lähettää viestejä. Jabber-palvelimen käyttö mahdollistaa sen, että myös kerrosten välinen liikenne on mahdollista.

Toteutuksessa mittaustulos lähetetään edellisen toteutuksen tapaan prosessisolmulta järjestelmäsolmulle, jossa siitä muodostetaan XMPP-viesti. Mittaava ja säätävä prosessisolmu voidaan valita minkä tahansa järjestelmäsolmun alta (kuva 32). Säätimelle lähetettävän mittaustuloksen vastaanottajaksi tulee valita järjestelmäsolmu, jonka radioverkossa säätävä prosessisolmu sijaitsee. Viestin pitäisi myös sisältää tieto, mikä säätävän prosessisolmun tunnus on.



Kuva 32. Jabber-palvelin viestin välittäjänä

Tämän toteutuksen viive riippuu edellisen toteutuksen tapaan siitä, miten monta prosessisolmua järjestelmäsolmujen alla on ja miten hyvin yhteys niihin on onnistunut. Järjestelmäsolmut lähettävät ja vastaanottavat Jabber-viestejä silloin, kun mittaustulokset on vastaanotettu prosessisolmuilta. Jos vastaanottava prosessisolmu on juuri lähettänyt Broadcast-viestin, kuluu maksimissaan toiset 160 ms, ennen kuin tieto lähetetään prosessisolmulle. Kokonaisviive jää kuitenkin maksimissaankin alle 400 ms, joten prosessisolmun säädintä voitaisiin ohjata yhden sekunnin välein, kuten edellisessä toteutuksessa.

Jabber-palvelimen käyttö mahdollistaisi tiedonsiirron eri kerrosten välillä. Toteutus vaatisi, että järjestelmäsolmun Jabber-toiminnoista tehtäisiin monipuolisempia.

8 TULOSTEN TARKASTELU

Säätimen toteuttaminen FPGA-piirille on monimutkainen ja vaativa prosessi. Syinä tähän ovat laskuoperaatioiden suorittaminen logiikalla ja suunnitteluohjelmien monimutkaisuus. Logiikalla toteutetuissa järjestelmissä joudutaan ottamaan huomioon, missä lukujärjestelmässä laskutoimituksien suoritus pitää tehdä. Ongelmia esiintyi varsinkin kertolaskujen suorittamisessa, jos toinen tai molemmat kerrottavista olivat negatiivisia.

FPGA-toteutuksen etuna oli, että laskennasta saatiin todella tehokasta. Lisäksi tällainen säädin vei vain vähän prosessorin suoritusaikaa. Työssä käytetty algoritmi laskettiin FPGA-lohkolla noin 1 µs aikana 8 MHz:n kellotaajuudella. Paperitehtaiden prosessit ovat yleensä hitaita, jolloin näin nopealle laskentateholle ei ole tarvetta. Toisaalta uusi ohjausarvo toimilaitteelle saadaan nopeasti mittauksen valmistumisen jälkeen. Nopeasta laskentatehosta saadaan suurin hyöty silloin, kun säädettävän prosessin viive ja vasteaika ovat erittäin pieniä. Tällaisia ovat esimerkiksi eräät moottorinohjaukseen liittyvät sovellukset.

Työssä käytetyssä FPGA-lohkossa oli 40 000 porttia. Näinkin suurella määrällä portteja voidaan toteuttaa melko monimutkaisia järjestelmiä, puhumattakaan suurista, yli sadantuhannen portin sisältävistä piireistä. Esimerkiksi tässä työssä piirin FPGA-lohkolle oli toteutettu säädin ja kaikki laitteen tiedonsiirtorajapinnat. Toteutus vei piirin pinta-alaa jo noin 75 - 80 %, mutta toisaalta logiikkaa ei ollut optimoitu mahdollisimman vähän tilaa vieväksi. Säätimen toteutuksessakin olisi luultavasti päästy pienempään pinta-alan käyttöön, jos rinnakkaisia toimintoja olisi suoritettu enemmän peräkkäin ja käytetty esimerkiksi samaa kertolaskumoduulia kaikissa kertolaskuissa. Tällöin myös säätimen laskenta-aika olisi pidentynyt huomattavasti.

Säätimen virittämisellä on tärkeä vaikutus säätimen toimintaan. Huonosti viritetty säädin ei toimi oikein, ja pahimmassa tapauksessa väärin viritetyn säätimen ulostulo voi vain värähdellä laidasta toiseen. Työssä tämä asia huomattiin, kun säädintä viritettiin. Teollisuuden kaupalliset säätimet on viritetty valmiiksi tietynlaisiin prosesseihin, ja usein ne toimivatkin jollain tapaa ilman virittämistä. Usein

tällaiset säätimet jätetäänkin virittämättä ja säätimen optimaalista toimintaa ei välttämättä saavuteta. Syynä virittämisen vähyyteen voi olla, että säätimen parametrien virittäminen prosessin ja toimintatavan mukaan on vaikeaa. Työssä parametrit viritettiin kokeilemalla eri arvoja, kunnes saatiin halutunlainen askelvaste. Tämä ei ole paras keino säätimen virittämiseen. Säätimen askelvasteesta saatiin teoriassa esitetyn kaltainen, mutta luultavasti tuloksesta olisi saatu parempi käyttämällä jotain muuta viritysmenetelmää, jonka jälkeen parametreja olisi vielä hienosäädetty.

Työssä verrattiin kolmea eri toteutustapaa, miten mittaustulos voitaisiin reitittää toiselle prosessisolmulle. Toteutustapoja on olemassa useita. Lyhyin viive saavutettaisiin lähettämällä mittaustulos suoraan prosessisolmulta toiselle. Toteutus on kuitenkin hankalaa nykyisessä mittaus- ja ohjausjärjestelmässä ja käyttöalue rajautuu yhteen kerrokseen, koska radioliikenne on erilaista eri kerroksissa. Tämä toteutustapa voisi olla toteutuskelpoisin jossain muussa järjestelmässä, jossa radioliikenne toimii jo valmiiksi prosessisolmujen kesken.

Toisessa toteutuksessa mittaustulos lähetetään järjestelmäsolmun kautta toiselle prosessisolmulle. Viive hieman kasvaa, ja uudet mittaustulokset saadaan järjestelmäsolmulle sekunnin välein. Sekunnin mittausväliä lyhentämällä mittaustulos voitaisiin päivittää nopeasti säätävälle prosessisolmulle. Viivettä voitaisiin myös pienentää, jos mittaustulos lähetettäisiin heti, kun se on saapunut mittaavalta prosessisolmulta, eikä odotettaisi kaikkien lähettimien mittaustuloksia.

Kolmas toteutustapa liittyi XMPP-viestin lähettämiseen toiselle järjestelmäsolmulle. Tämän toteutuksen etuna on, että mittaustulos voidaan lähettää minkä tahansa järjestelmäsolmun alle ja silti mittaustulos viipyy matkalla pahimmassakin tapauksessa alle 400 ms. Lisäksi tämä olisi helpoin tapa toteuttaa mittaustiedon lähettäminen prosessisolmulta toiselle, joten tämä olisi varteenotettavin toteutusvaihtoehto.

9 YHTEENVETO

Insinöörityössä toteutettiin PID-säädin järjestelmäpiirillä. Säädintä ohjattiin langattoman verkon kautta. Säädettävän suureen mittaus sekä säätöalgoritmi olivat samassa prosessisolmussa. PID-säädin toteutettiin piirin FPGA-lohkolla ja sitä ohjattiin samalle piirille integroidun prosessorin avulla.

Aluksi PID-säädin suunniteltiin ja toteutettiin kehitysalustalle, jossa sen toiminta testattiin. Seuraavassa vaiheessa säädin toteutettiin prosessisolmulle, jossa käytettiin hyväksi aiemmin kehitettyä säädintä. Säätimen toteutuksessa prosessisolmulle rekisterien kokoja jouduttiin pienentämään tilansäästämisen vuoksi. Prosessisolmun toteutuksessa suurimmat muutokset jouduttiin tekemään tiedon välitykseen eri moduulien välillä. Säätimestä toteutettiin sellainen, että mittaustuloksien tallentaminen tietokantaan sekä ohjaustietojen välittäminen muille kuin säätimen toimilaitteelle toimi normaalisti. Lisäksi säädintä voidaan käyttää kaikissa PID-säätimen moodeissa.

Työn loppuvaiheessa tutkittiin toteutustapoja, miten mittausarvo voitaisiin lähettää toiselta prosessisolmulta säätävälle prosessisolmulle langattoman verkon kautta. Toteutustapojen välisinä eroina tutkimuksessa olivat mittaustuloksella matkaan kuluva viive sekä se, miten helposti se olisi toteutettavissa nykyiseen järjestelmään.

Työssä eniten vaikeuksia tuottivat laskutoimitusten suorittaminen logiikalla sekä suunnitteluohjelmiston ongelmat. System Designer -ohjelmassa todettiin olevan paljon virheitä, vaikka uusiakin päivityksiä oli asennettu. Yksi ongelmista oli esimerkiksi se, että piirin ohjelmointiin tarkoitettu ohjelma kaatui useita kertoja, ennen kuin suunniteltu logiikka saatiin ohjelmoitua piirille. Ohjelmistossa esiintyi myös muita ongelmia, kuten HDL-editorin kaatuilu sekä synteesiohjelman huono toiminta aika ajoin.

Työssä opittiin säätötekniikkaa sekä logiikan suunnittelua FPGA-piirille, joista molemmista oli erittäin vähän kokemusta ennen työn suoritusta. Tämä insinöörityö selvensi paljon molempia tekniikan osa-alueita.

LÄHTEET

1 Dorf, R. & Bishop, R. Modern Control Systems. Kymmenes painos. 881 s. ISBN 0-13-127765-0.

2 Lappeenrannan teknillinen yliopisto, sähkötekniikan osasto. Mikroprosessori - elektroniikkalaitteen digitaalinen äly. Luettu 24.2.2005. [WWW-dokumentti]. http://www.ee.lut.fi/mikroprosessori.html

3 Teknillinen korkeakoulu, systeemitekniikan laboratorio. Digitaalisen säädön verkkokurssi. Päivitetty 21.1.2005. [WWW-dokumentti]. http://www.control.hut.fi/Kurssit/AS-74.2112/verkkokurssi/

4 Paltemaa, I., Raipala, J., Uusitalo, P., Ala-Rantala, K., Jussila, T. & Yli-Fossi, T. Säädön suunnittelu. Päivitetty 25.8.2004. 14 s. [PDF-dokumentti]. http://www.ac.tut.fi/aci/courses/ACI-20030/akt/sslabra04.pdf

5 Nihtilä, M., Oksanen, T., Eriksson, L. P-, PI- ja PID-säätö. Teknillisen korkeakoulun systeemitekniikan laboratorio, 2002. 21 s. [PDF-dokumentti]. http://www.control.hut.fi/Kurssit/AS-0.2230/tyo11/Ohje.pdf

6 Zhao, W., Kim, B.H., Larson A.C. & Voyles, R.M. FPGA Implementation of Closed-Loop Control System for Small-Scale Robot. IEEE Advanced Robotics, 2005. ICAR ’05 julkaisut. Sivut 70-77. [PDF-dokumentti]. ieeexplore.ieee.org/iel5/10070/32295/01507393.pdf?arnumber=1507393

7 Mellon, C. CTM: PID Tutorial. [WWW-dokumentti] Päivitetty 26.8.1997.

http://www.engin.umich.edu/group/ctm/PID/PID.html

8 Ogata, K. Discrete-Time Control Systems. Toinen painos. Englewood Cliffs: Prentice Hall, 1995. 745 s. ISBN: 0-13-034281-5

9 Harju, T. PID-säätimen käytännön viritysmenetelmät. Teknillinen korkeakoulu. 31 s. Luettu 13.3.2006. [PDF-dokumentti]. http://www.control.hut.fi/Kurssit/AS-74.3176/TuneUp_Luento.pdf

10 Teknillinen korkeakoulu, systeemitekniikan laboratorio. PID-säätimen virittäminen. 10 s. [PDF-dokumentti]. http://www.control.hut.fi/Kurssit/AS-0.2230/tyo8/Ohje.pdf

11 Haltsonen, S, Levomäki, J. & Rautanen, E.T. Digitaalitekniikka. Helsinki: Edita Prima, 2004. 400 s. ISBN: 951-37-3886-8.

12 Lahti, J. Digitaalitekniikka III. Oulun yliopisto, elektroniikan laboratorio, 1999. Luentomoniste.

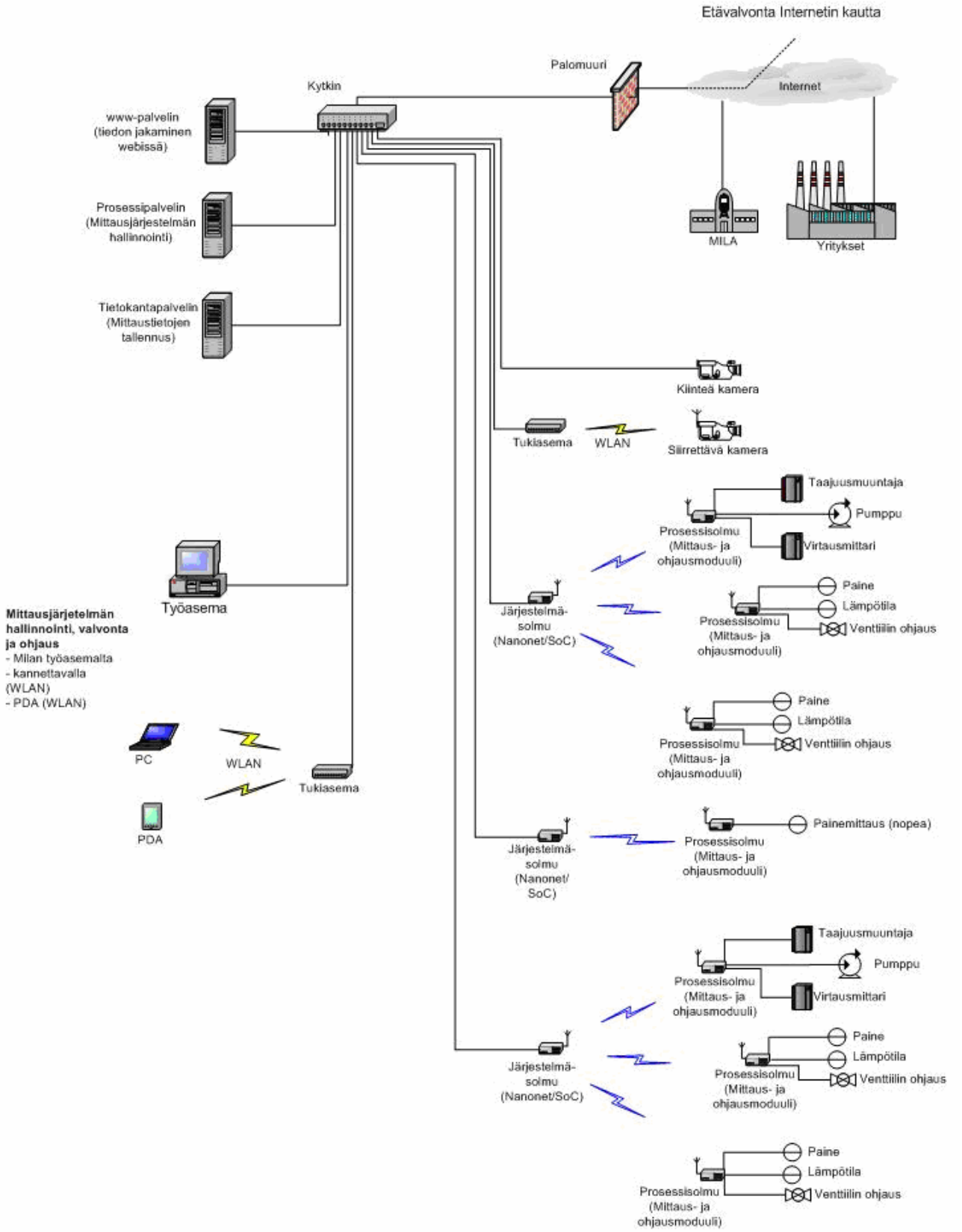
13 Hänninen, V. ”Ensimmäinen kenttäohjelmoitava järjestelmäpiiri”. Prosessori-lehden uutiset, 1999.

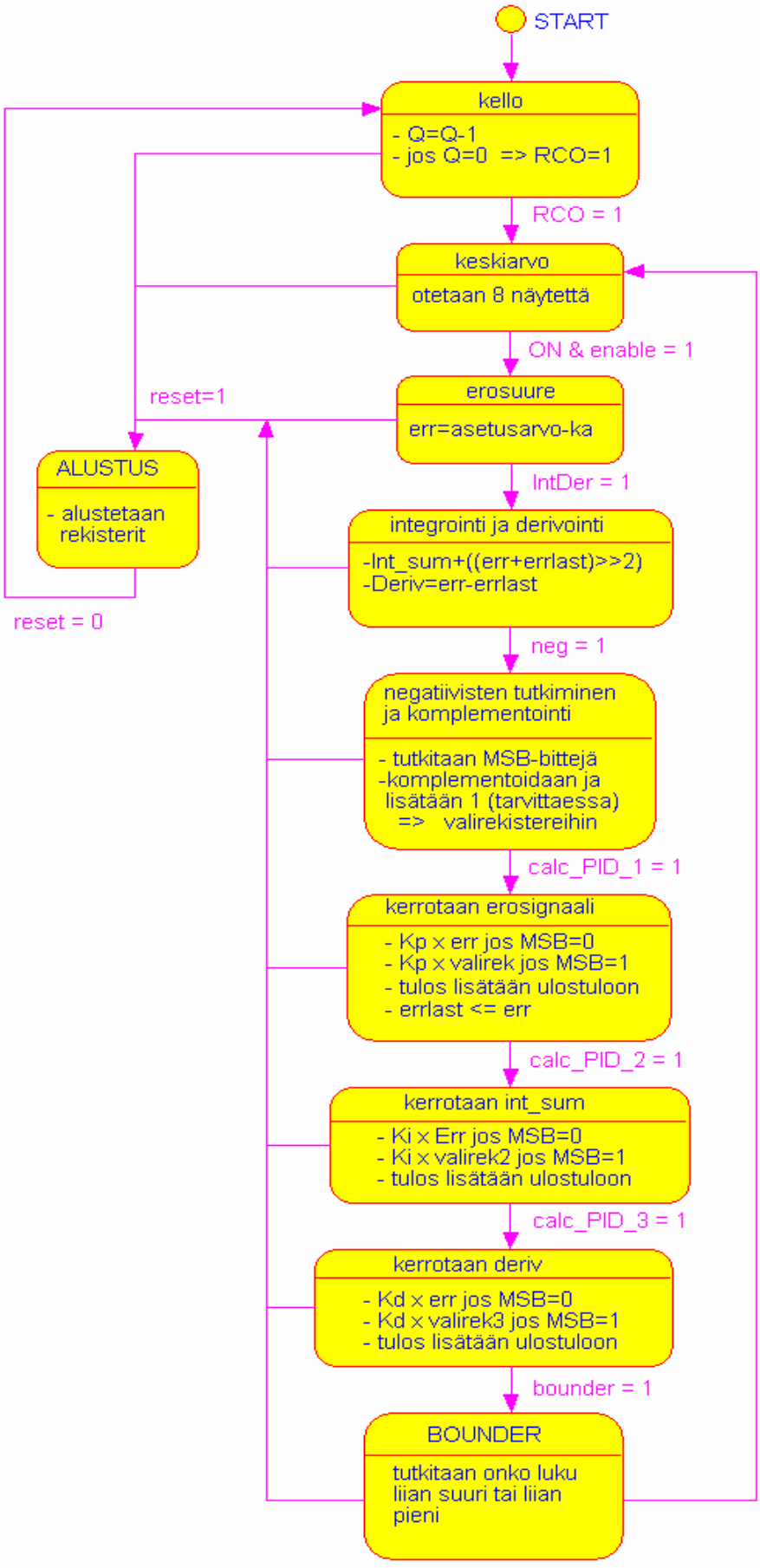
14 Atmel Corporation. FPSLIC (AVR with FPGA) from Atmel. Luettu 17.3.2006. [WWW-dokumentti]. http://www.atmel.com/products/FPSLIC/

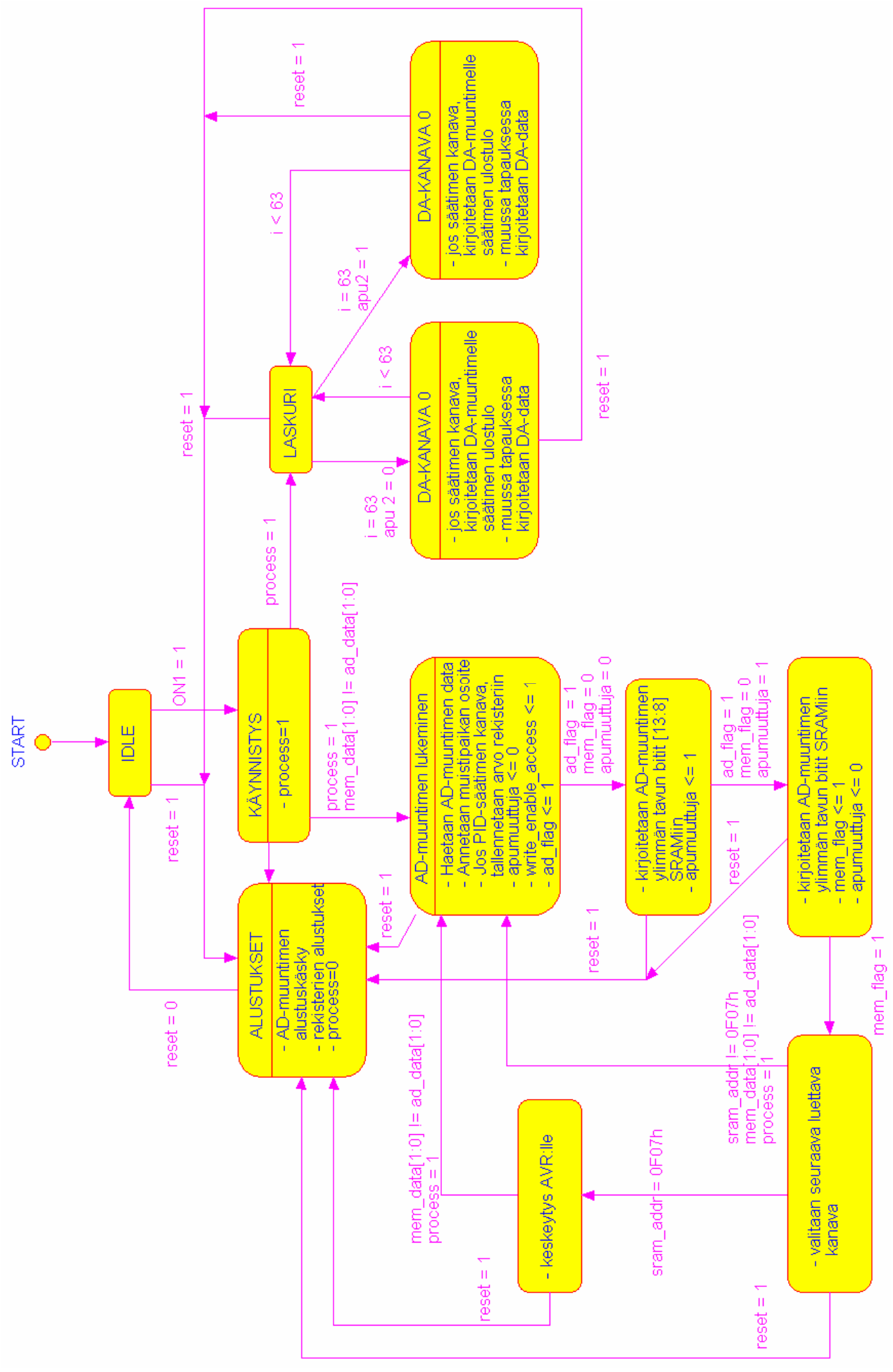
15 Wikström, K. “Ohjelmoitavilla nopeasti markkinoille”, Prosessori 11/1998.

16 Hyde, D.C. CSCI 320 Computer Architecture Handbook on Verilog HDL. Päivitetty 23.8.1997. [PDF-dokumentti] http://www.eg.bucknell.edu/~cs320/1995-fall/manual.pdf

LIITTEET


Etävalvonta Internetin kautta
Palomuuri
Kytkin
Internet
www-palvelin
(tiedon jakaminen
webissä)
Prosessipalvelin
(Mittausjärjestelmän
hallinnointi)
Tietokantapalvelin
(Mittaustietojen
tallennus)
MILA
Yritykset
Kiinteä kamera
Tukiasema
WLAN
Siirrettävä kamera
Taajuusmuuntaja
Pumppu
Prosessisolmu
(Mittaus- ja
ohjausmoduuli)
Virtausmittari
Työasema
Järjestelmä-
solmu
(Nanonet/SoC)
Paine
Lämpötila
Prosessisolmu
(Mittaus- ja
ohjausmoduuli)
Venttiilin ohjaus
Mittausjärjetelmän
hallinnointi, valvonta
ja ohjaus
- Milan työasemalta
- kannettavalla
(WLAN)
- PDA (WLAN)
PC
WLAN
Tukiasema
PDA
Painemittaus (nopea)
Järjestelmä-
solmu
(Nanonet/
SoC)
Prosessisolmu
(Mittaus- ja
ohjausmoduuli)
Taajuusmuuntaja
Pumppu
Prosessisolmu
(Mittaus- ja
ohjausmoduuli)
Virtausmittari
Järjestelmä-
solmu
(Nanonet/SoC)
Paine
Lämpötila
Prosessisolmu
(Mittaus- ja
ohjausmoduuli)
Venttiilin ohjaus
Paine
Lämpötila
Prosessisolmu
(Mittaus- ja
ohjausmoduuli)
Venttiilin ohjaus


START
kello
- Q=Q-1
- jos Q=0 => RCO=1
RCO = 1
keskiarvo
otetaan 8 näytettä
ON & enable = 1
erosuure
err=asetusarvo-ka
reset=1
ALUSTUS
- alustetaan rekisterit
reset = 0
IntDer = 1
integrointi ja derivointi
-Int_sum+((err+errlast)>>2)
-Deriv=err-errlast
neg = 1
negatiivisten tutkiminen ja komplementointi
- tutkitaan MSB-bittejä
-komplementoidaan ja lisätään 1 (tarvittaessa)
=> valirekistereihin
calc_PID_1 = 1
kerrotaan erosignaali
- Kp x err jos MSB=0
- Kp x valirek jos MSB=1
- tulos lisätään ulostuloon
- errlast <= err
calc_PID_2 = 1
kerrotaan int_sum
- Ki x Err jos MSB=0
- Ki x valirek2 jos MSB=1
- tulos lisätään ulostuloon
calc_PID_3 = 1
kerrotaan deriv
- Kd x err jos MSB=0
- Kd x valirek3 jos MSB=1
- tulos lisätään ulostuloon
bounder = 1
BOUNDER
tutkitaan onko luku liian suuri tai liian pieni


START
IDLE
ON1 = 1
reset = 1
reset = 0
KÄYNNISTYS
- process=1
process = 1
ALUSTUKSET
- AD-muuntimen alustuskäsky
- rekisterien alustukset
- process=0
reset = 1
process = 1
mem_data[1:0] != ad_data[1:0]
AD-muuntimen lukeminen
- Haetaan AD-muuntimen data
- Annetaan muistipaikan osoite
- Jos PID-säätimen kanava, tallennetaan arvo rekisteriin
- apumuuttuja <= 0
- write_enable_access <= 1
- ad_flag <= 1
ad_flag = 1
mem_flag = 0
apumuuttuja = 0
- kirjoitetaan AD-muuntimen ylimmän tavun bitit [13:8] SRAMiin
- apumuuttuja <= 1
ad_flag = 1
mem_flag = 0
apumuuttuja = 1
- kirjoitetaan AD-muuntimen ylimmän tavun bitit SRAMiin
- mem_flag <= 1
- apumuuttuja <= 0
mem_flag = 1
- valitaan seuraava luettava kanava
sram_addr != 0F07h
mem_data[1:0] != ad_data[1:0]
process = 1
sram_addr = 0F07h
- keskeytys AVR:lle
mem_data[1:0] != ad_data[1:0]
process = 1
reset = 1
reset = 1
reset = 1
reset = 1
reset = 1
LASKURI
i < 63
i = 63
apu2 = 1
i < 63
i = 63
apu 2 = 0
DA-KANAVA 0
- jos säätimen kanava, kirjoitetaan DA-muuntimelle säätimen ulostulo
- muussa tapauksessa kirjoitetaan DA-data
DA-KANAVA 0
- jos säätimen kanava, kirjoitetaan DA-muuntimelle säätimen ulostulo
- muussa tapauksessa kirjoitetaan DA-data
reset = 1
reset = 1