Tampereen yliopisto

Antti Peltotalo

TEHOELEKTRONIIKAN OHJAUSKORTIN FPGA-PIIRIN TOIMINTA JA NOPEIDEN SARJAVÄYLIEN SUORITUSKYVYN ARVIOINTI

Informaatioteknologian ja viestinnän tiedekunta
Diplomityö
Maaliskuu 2022

TIIVISTELMÄ

Antti Peltotalo: Tehoelektroniikan ohjauskortin FPGA-piirin toiminta ja nopeiden sarjaväylien suorituskyvyn arviointi
Diplomityö
Tampereen yliopisto
Sähkötekniikka
Maaliskuu 2022

Tehoelektroniikkajärjestelmässä digitaalisen säätimen toteuttamat tehtävät voidaan jakaa usean eri integroidun piirin kesken. Mikroprosessorilla suoritetaan esimerkiksi käyttöliittymän toimintoja sekä kommunikaatiorajapintoja käyttäjän suuntaan. Digitaaliselle signaaliprosessorille annetaan raskaat ja paljon prosessointitehoa vaativat säätöalgoritmien laskennat. FPGA:n vastuulla on suorittaa nopeita ja aikakriittisiä tehtäviä. Eri vastuualueiden piirien tulee myös voida keskustella viiveettä ja vaihtaa usein suuria määriä dataa, jolloin kommunikaatioväylän tulee toimia tehokkaasti.

Työssä käytiin läpi nopeiden sarjaväylien toimintaperiaatteita ja vaihtoehtoja FPGA:n ja DSP:n väliseen kommunikaatioon, sekä toteutettiin tehoelektroniikkajärjestelmän digitaalisen säätimen FPGA-piirin toiminnallisuudet Merus Power Dynamics Oy:n suunnittelemaan säätöjärjestelmän prototyyppiin. FPGA-moduuliksi valittiin Trenz Electronicsin TE0714 Artix-7 piirillä. Säätöalgoritmien laskentaan käytettiin 66AK2G12 System-on-Chip piirikortin DSP-ydintä, jonka toiminta ei kuulu tämän työn aihealueeseen. Piirien välisen tiedonsiirron väyläksi valikoitui PCI Express tarjolla olevien valmiiden lohkojen ja hyvän tuen perusteella. Toteutettuihin FPGA:n ominaisuuksiin kuului AD-muuntimien ajurien kirjoittaminen, ylätason säätösilmukan toteutus ja PCIe-lohkon käyttöönotto. Työn tavoitteena oli verifioida digitaalisen säätimen prototyypin toimintaa loistehokompensoinnin virtareferenssin säädössä. Prototyypin kolmivaiheinen virran ja jännitteen mittaus toteutettiin erillisillä virta- ja jännitemittauskorteilla, joissa käytettiin ADS8866 16-bittisiä daisy chainiin kytkettyjä AD-muuntimia. 3x16-bit mittausdata luettiin molemmilta korteilta FPGA:lle kerran säätösilmukan jaksossa. Säätösilmukan jakson pituudeksi valittiin 50 mikrosekuntia, jonka aikana luetaan virran ja jännitteen arvot FPGA:lle, lähetetään data PCIe-väylän yli DSP:lle säätöalgoritmin laskentaa varten, ja lopuksi lähetetään DSP:n laskema virtareferenssi valokuitulähettimelle. Suurimmaksi pullonkaulaksi säätösilmukan suorituksessa osoittautui AD-muuntimien näytteenottotaajuus. ADS8866-piiri daisy chain -moodissa aloittaa uuden näytteistyksen kerran 10 mikrosekunnissa, joka on 20 % koko säätösilmukan suoritusajasta. PCI Express -väylän käyttöönotto osoittautui haastavaksi tehtäväksi, mikä aiheutti lopulta tehottoman 14,3 % hyötykuorman siirron. Toisaalta 4,48 mikrosekunnin lukuoperaatio oli silti riittävä prototyypin toiminnan verifiointiin.

Merus Power Dynamics Oy:n säädinkortin prototyypillä tehtiin kaksi testiä. Säätimen toiminta todettiin suoriutuvan onnistuneesti loisvirtareferenssin laskennasta- ja lähettämisestä aktiivisuodatinmoduulille, jota käytettiin nimellisellä teholla. Tämän jälkeen aktiivisuodatinta käytettiin tyhjäkäynnillä ilman kuormaa. Prototyypin toiminta todettiin, kun kuormamoduuli kytkettiin verkkoon. Järjestelmän vasteaika loistehon kompensointiin oli vähän yli puoli verkkojaksoa eli noin 10 millisekuntia. Prototyyppi siis suoriutui sille asetetuista testeistä ja säädinkehitys voi edetä seuraavaan vaiheeseen.

Avainsanat: tehoelektroniikkajärjestelmä, digitaalinen säätöjärjestelmä, FPGA, PCI Express, korkean suorituskyvyn sarjaväylä

ABSTRACT

Antti Peltotalo: FPGA functions of Power electronic system control platform and evaluation of high-speed serial buses performance
Thesis type
Tampere University
Degree Programme
March 2022

Digital controller functionalities in a power electronic system can be divided between several different integrated circuits. A Microprocessor is used for user interface and communication functionalities. Digital signal processor is responsible for heavy tasks such as, control algorithm calculation, which requires a lot of computational power. FPGA is used for fast time critical tasks. Integrated circuits for special responsibilities must also transfer great amounts of data with minimal latency. That is one of the reasons why the used communication bus must be efficient.

In the thesis, high-speed serial bus functionalities and options for FPGA and DSP communication are reviewed. Power electronic system digital controller FPGA implementation was also created for Merus Power Dynamics Oy control platform prototype. Trenz Electronic TE0714 FPGA module with Artix-7 chip was chosen as FPGA module. Digital signal processor unit on the 66AK2G12 System-on-Chip platform was used for control algorithm calculation. Functionalities of the DSP are not in the scope of this thesis. PCI Express was chosen as serial bus for the prototype, since it had intellectual properties available along with good support. Implemented FPGA features were A/D converter control logic, main control loop logic and PCIe core integration. Goal of the thesis was to verify usage of digital controller prototype as a current reference controller for active harmonic filter acting as a reactive power compensator. Separate PCBs were used for three-phase current and voltage measurement. Each PCB had daisy chained ADS8866 16-bit A/D converters. 3x16-bit measurement data was sent to the FPGA from both measurement PCBs once in a control cycle. Length of the control cycle was selected to 50 microseconds. During the control loop, currents and voltages are read, measurement data is sent to DSP via PCIe bus, control algorithm is ran, and finally current reference values are sent to the fiber optic transmitter. The biggest bottle neck in the system turned out to be sampling frequency of A/D converters. In daisy-chain mode, ADS8866 chip starts new conversion once every 10 microseconds which is 20% of the complete control cycle. Integrating PCIe bus to the system turned out to be most challenging and time-consuming task, which resulted to a slightly inefficient 14,3% payload efficiency for data transfer. On the other hand, memory read request with the duration of 4,48 microsecond was still enough for the verification of the prototype.

Two verification tests were made for Merus Power Dynamics Oy control platform prototype. Correct behaviour of the prototype was verified successfully in a setup where the prototype provided current reference signal to a active harmonic filter acting as reactive power compensator. Second test included running the active harmonic filter module without load and checking the response of control platform prototype when a load was connected to the network. Response of the platform was half of the network cycle, which is 10 milliseconds. The conclusion is, that control platform prototype performed well in both tests required. Successful verification provided Merus Power with necessary information for proceeding to the next phase with the control platform development.

Keywords: power electronic system, digital control platform, FPGA, PCI Express, high-speed serial bus

ALKUSANAT

Tämä diplomityö on kymmenen vuoden opintojeni viimeinen etappi. Urakka ei ole ollut helppo, sillä kirjoitusprosessiin on saanut etsiä motivaatiota enemmän kuin olisi osannut kuvitella. Päällimmäisenä mielessä on helpotus. Haluan kiittää perhettäni siitä, että he kannustivat minut yliopisto-opintojen pariin.

Olen kiitollinen siitä, että sain opiskella juuri Tampereen Teknillisessä Yliopistossa, ja tutustua TTY:n yhteisön kiltoihin ja kerhoihin, jotka tekivät opiskeluajastani muistamisen arvoisen. Ilman tätä yhteisöä ja sen ihmisiä en olisi saanut opintojeni ohella yhtä erityisiä kokemuksia ja taitoja. Iso kiitos Sähkökillalle, TiTe:lle, killoille, joiden kautta löysin paljon kavereita ja hyviä tuttuja. Erityiskiitos kuuluu kuitenkin EESTEC LC Tampereelle, jonka avulla kasvoin itsevarmemmaksi, opin elämästä, sekä sain paljon kansainvälisiä ystäviä. Kiitos myös Tampereen Akateemiselle Kyykkäseuralle sekä kyykkää pelaavalle väelle.

Kiitos Merukselle ja sen työyhteisölle. Sain hyvän ensimmäisen oman alan työpaikan ja mielenkiintoisen diplomityöaiheen.

Tampereella, 28. maaliskuuta 2022

Antti Peltotalo

SISÄLLYSLUETTELO

KUVALUETTELO

TAULUKKOLUETTELO

OHJELMA- JA ALGORITMILUETTELO

LYHENTEET JA MERKINNÄT

8b/10b-koodaus	Tietoliikenteessä käytettävä linjakoodaus
A/D-muunnin	Analogia-digitaalimuunnin (engl. analog-to-digital converter, ADC)
ASIC	Tiettyyn käyttötarkoitukseen suunniteltu mikropiiri (engl. application-specific integrated circuit)
Aurora protokolla	Xilinxin kehittämä linkkitason tiedonsiirtoprotokolla
CMOS	Kanavatransistoreihin perustuva mikropiiritekniikka (engl. Complementary Metal Oxide Semiconductor)
D/A-muunnin	Digitaali-analogiamuunnin (engl. DAC, Digital to Analog Converter)
DFT	Fourier-muunnoksen diskreettiaikainen yleistys (engl. discrete fourier transform)
DSP	Digitaalista signaalinkäsittelyä varten rakennettu suoritin
EMIF	(engl. External Memory Interface)
ESS	Järjestelmä energian varastointiin ja energiatuotannon tasapainottamiseen (engl. Energy Storage System)
FPGA	Ohjelmoitava porttimatriisi, uudelleenohjelmoitava logiikkapiiri (engl. field-programmable gate array)
Gbps	Gigabittiä sekunnissa (engl. Gigabits per second)
HDL	Laitteiston kuvauskieli, esimerkiksi kytkennän kuvaus FPGA-piirille (engl. hardware description language)
HSS	(engl. High-Speed Serial)
I/O	Siirräntä, tiedon siirtäminen esim. suorittimesta sisään tai siitä ulos (engl. input/output)
IGBT	Kytkinkäyttöön tarkoitettu suuritehoinen bipolaaritransistori (engl. Insulated-Gate Bipolar Transistor)
ISO	International Standardization Organization
LV	Alajännitepuoli järjestelmässä (engl. low voltage)
MOSFET	Digitaalisissa sekä analogisissa piireissä käytettävä eristehilatransistori (engl. Metal-oxide-semiconductor field-effect transistor)
MPU	Mikroprosessori, mikropiirille ingetroitu suoritin (engl. microprocessor unit)
MV	Keskijännitepuoli järjestelmässä (engl. medium voltage)

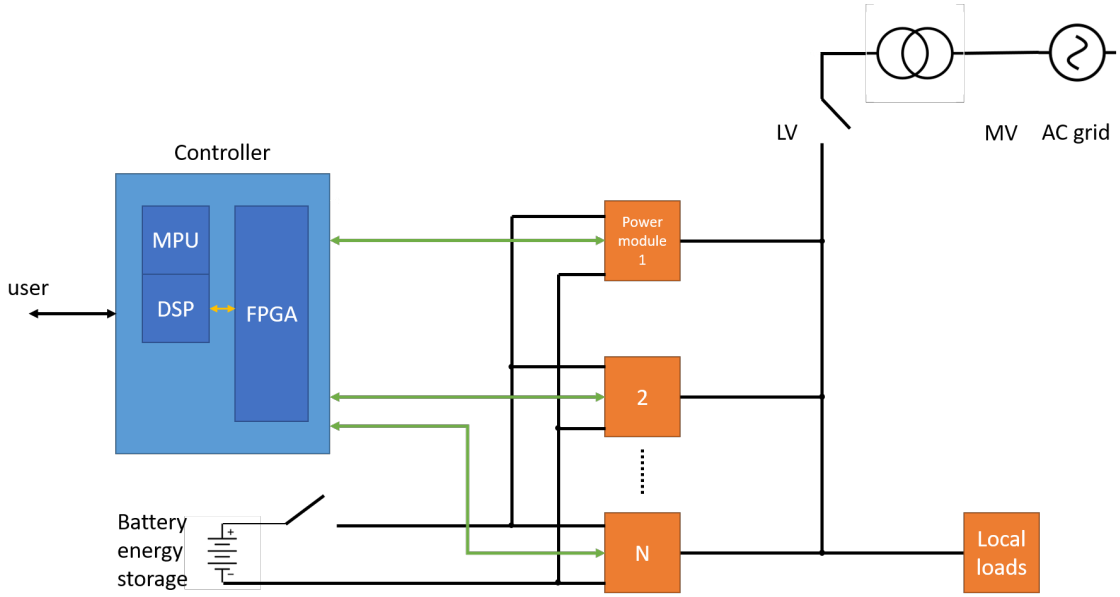
OSI-malli	Open Systems Interconnection, malli, jonka puitteissa tietoliikennejärjestelmät tulisi suunnitella
PCI	Tietokoneväylä (engl. Peripheral Component Interconnect)
PCI Express	Peripheral Component Interconnect Express - suurinopeuksinen standardoitu väylätyyppi
PLC	Programmable Logic Controller, Teollisuuden automaatio-ohjauksessa käytettävä pieni tietokone
PLL	säätöjärjestelmä, jonka ulostulo riippuu sisäänmenon vaihekulmasta (engl. phase-locked loop)
PWM	Pulssinleveysmodulaatio, pulssisuhteella säädettävä jännitteen modulointitapa (engl. pulse-width modulation)
SDRAM	(engl. Universal Parallel Port)
SerDes	(engl. Serializer/Deserializer)
UPP	(engl. Universal Parallel Port)

1 JOHDANTO

Mikroelektroniikan laajamittainen kehitys mikroprosessoreiden, digitaalisten signaaliprosessoreiden (engl. Digital Signal Processor, DSP), muistipiirien, A/D- ja D/A-muuntimien, CMOS-piirien sekä tehotransistorien (MOSFET, IGBT) alueella on laajentanut huomattavasti tehoelektroniikan käyttöä viimeisen kolmenkymmenen vuoden aikana. Erityisesti tehotransistorien kehitys on kasvattanut sovellusten käyttövirran skaalaa muutamista ampeereista tuhansiin ampeereihin ja käyttöjännitteen skaalaa sadoista volteista kilovoltteihin jopa miljoonien hertsien kytkentätaajuudella [1]. Energian hinnan kasvun ja ympäristövaatimusten muuttuessa ankaremmiksi tehoelektroniikan sovellukset leviävät yhä laajemmalle, sillä tehoelektroniikka on avainteknologioita uusiutuvien energianlähteiden hyödyntämisessä ja energian säästössä [2]. Energiatilanteen muuttuessa monimutkaisemmaksi sähkönlaadulle asetetut vaatimukset, sähköverkon vakaus, uusiutuvan energian integrointi verkkoon ja energianvarastointijärjestelmät (ESS) tuovat omat haasteensa mukaan yhtälöön, mikä tulee ottaa huomioon [3] [2].

Tehoelektroniikkajärjestelmää ohjataan ja valvotaan nykyään useimmiten digitaalisella säätimellä, joka mittaa systeemin ulostuloja ja säätää haluttuja suureita vertaamalla systeemin ulostuloja referenssiarvoihin, mikä minimoi erosuureen [4]. Digitaalisella säätimellä toteutettuja toimintoja voidaan vaivattomasti muokata ja päivittää laitteiston asentamisen jälkeenkin. Mikroprosessorien potentiaalin ansiosta saadaan toteutettua vaativia ohjausalgoritmeja reaaliajassa [1]. Säätimen toiminnot voidaan edelleen jakaa järjestelmäohjaukseen ja tehomuuntimen ohjaukseen. Järjestelmäohjauksen toimintoihin voi kuulua esimerkiksi hitaiden ohjausalgoritmien, kuten tehomuuntimien säätöreferenssien tuottamisen sekä järjestelmäparametrien säätö. Tehomuuntimen ohjaus puolestaan sisältää nopean ja aikakriittisen säädön sekä pulssinleveysmoduloinnin transistorien hilaohjaussignaalien tuottamiseksi, jotka yleensä toteutetaan laitteen kytkentätaajuudella. Kuvassa 1.1 esitelty säädin on toteutettu keskitetyllä säätöarkkitehtuurilla, eli yksi pääsäädin ohjaa koko järjestelmää. Kuvassa 1.1 on esitetty esimerkki akkuvarastoon kytketyn tehomuunninjärjestelmän lohkokaaviosta. Tehomuuntimien DC-puoli on kytketty akkuvarastoon ja AC-puoli on kytketty muuntajan kautta sähköverkkoon ja suoraan paikallisiin kuormalaitteisiin. Pääsäätimestä lähtee tyypillisesti keskijännitetasolla yhteys paikallisille tehomuunninmoduulien säätimille sekä mittaussignaaleille. Lisäksi säätimelle tuodaan mittaussignaaleita järjestelmätasolta. Sillä voidaan myös toteuttaa yhteys muihin

ulkoisiin järjestelmiin ja käyttöliittymiin.



Kuva 1.1. *Akkuvarastoon kytketyn tehomuunninjärjestelmän lohkokaavio*

Järjestelmätasolla mitataan esimerkiksi seuraavia suureita:

- verkkojännite
- kuormavirta
- kokonaisinvertterivirta
- DC-linkin jännite
- Akkujärjestelmän jännite
- kuormapuolen AC-jännite

Tehomuuntajien puolella tarvitaan seuraavia mittauksia:

- tehomuuntajamoduulien virta
- DC-jännite
- IGBT-moduulien lämpötila

Field-programmable gate array, eli FPGA toteuttaa tässä tapauksessa tehomuuntimien nopeiden ja aikakriittisten ohjaussignaalien tuottamisen sekä AD-muuntimien ohjauksen. FPGA on uudelleenohjelmoitava logiikkapiiri. Se toimii tehokkaasti nopeissa ja aikakriittisissä ohjaus- ja suojaustehtävissä sekä rinnakkaisuutta vaativissa toiminnoissa [5]. Suuren volyymin tuotannossa FPGA:n tilalla käytetään myös ASIC-piirejä (Application-specific integrated circuit), jotka voivat olla kustannustehokkaita, mutta evät ole uudelleenohjelmoitavissa [6]. Kuvan 1.1 MPU eli mikroprosessoriyksikkö voi toteuttaa järjestelmän vähemmän aikakriittisiä toimintoja ja tehtäviä, jotka vaativat vähemmän prosessointikykyä. MPU:lla toteutetaan myös käyttöliittymät ja kommunikaatiorajapinnat. DSP eli digitaalinen signaaliprosessori suorittaa parempaa laskentatehoa vaativat toiminnot, kuten tehomoduulin

säätöalgoritmit ja signaalinkäsittelyn sähkönlaadun analysoimiseksi. DSP:n käsittelemä data saadaan FPGA:lta kommunikaatiolinkin kautta.

Tehomuuntimien toiminta perustuu vielä useimmiten IGBT-bipolaaritransistoreilla toteutettuihin kytkinmoduuleihin [7], joiden maksimikytkentätaajuus liikkuu noin 20 kHz:n rajalla [8]. Mutta yhä useammin sovelluksissa käytetään jo Silicon Carbide (SiC) transistoreilla toteutettuja kytkinmoduuleita, joilla saadaan pienennettyä tehohäviöitä huomattavasti perinteisiin piitä käyttäviin tehotransistoreihin verrattuna [7][9]. SiC-transistoreilla voidaan saavuttaa myös yli 20kHz:n kytkentätaajuuksia [8], joten aikaa säädön suorittamiselle ja viiveille ei juuri ole. Tehomuuntimissa luotettavuus on tärkein ominaisuus, joka toteutuu muun muassa vikadiagnostiikalla, joihin reagoidaan sopivilla toimilla. Muita tärkeitä vaatimuksia ovat esimerkiksi järjestelmän moduulien välinen synkronointi, kommunikaation nopeus ja matala slave-säätimien prosessointiviive [10]. Esimerkiksi mitattavien analogiasuureiden AD-muunnokset ja datan siirto laskentaa suorittavien piirien välillä vievät laskenta-aikaa ohjausalgoritmia suorittavalta piiriltä, jolloin ylimääräiset viiveet halutaan minimoida [11]. Solmukohtien määrä järjestelmässä määrittää [10] mukaan masterin I/O pinnien määrän, vaadittavan rinnakkaisen prosessointikyvyn ja jokaisen säätösyklin prosessointi- ja etenemisviiveen. Kommunikaatiokanavan vaatima kapasiteetti, eli säädön kaistanleveys, voidaan määritellä [10] ja [12] mukaan kaavalla 1.1,

$$Kaistanleveys = N_{val/node} * N_n * N_b * f_{SW} * (1 + k_{oh}) \tag{1.1}$$

missä N_n on järjestelmän slave-solmukohtien lukumäärä, F_{SW} on muuntimen kytkentätaajuus, $N_{val/node}$ on solmuihin lähetettävien lukuarvojen lukumäärä, $N_{b/val}$ on bittien lukumäärä per lukuarvo ja P_{ovb} on ei-informaatiobittien (overheadbittien) lukumäärä prosentteina. Norjalainen tutkimus [12] antaa esimerkin yksinkertaisesta kolmivaiheisesta järjestelmästä, joka koostuu neljästä solmupisteestä (1 mastersolmu ja 3 slavesolmua), joiden välillä kulkee jännite-, virta-, status- ja PWM pulssisuhdedataa. Jos PWM-signaalin resoluutio bitteinä on 10, tehomuuntimen kytkentätaajuus 10 KHz, ja datansiirron hyötykuorma 50 % saadaan järjestelmän kaistanleveydeksi:

$$\begin{aligned} Kaistanleveys &= 4 * 3 * 10 * 10000 * (1 + 0.5) \\ Kaistanleveys &= 1.8Mb/s \end{aligned} \tag{1.2}$$

Luonnollisesti monimutkaisempi järjestelmä vaatii paljon suuremman kaistanleveyden, sillä kaistanleveys riippuu kaikista järjestelmän muuttujista. [12] asettaa järjestelmälle halutuksi kaistanleveydeksi 100 Mb/s, jotta järjestelmän suunnitteluun saataisiin enemmän joustavuutta. Myös mahdollisuus tulevaisuuden laajennuksille tulee ottaa huomioon.

Aiemmin mainituista tehomuuntimien ohjausjärjestelmien vaatimuksista tämä diplomityö keskittyy säätöjärjestelmän FPGA:n ja DSP:n välisen pienilatenssisen datansiirtoväylän

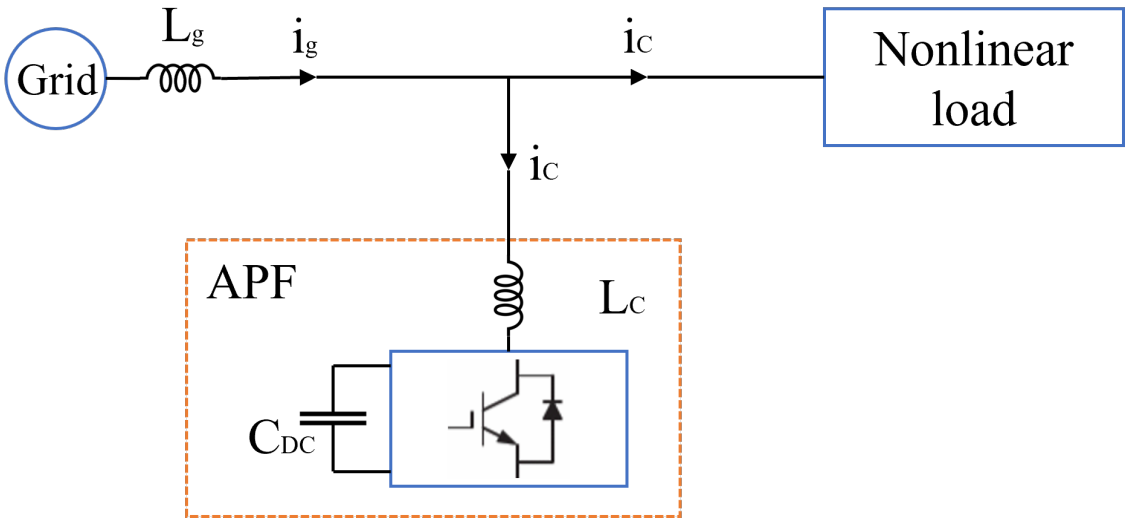
valintaan ja FPGA:n HDL-toteutukseen. Piirien välinen linkki on kriittinen osa säätimen toimintaa, sillä datan määrä kyseisessä linkissä voi olla suuri, ja vaatimus latenssille tiukka. Säädin voi toimia monien erilaisten tehoelektroniikkajärjestelmien päätason säätimenä. Tässä työssä pääsäätimen prototyyppiä testataan aktiivisuodattimen kanssa loistehon kompensoinnissa. Tämän kappaleen jälkeen käydään nopeasti läpi aktiivisuodattimen toiminta. Kappaleet kolme ja neljä keskittyvät tiedonsiirtomenetelmiin, ja lopuksi esitellään prototyyppisäätimen FPGA:n toiminta ja piirien välisen kommunikaation toteutusta. Viimeisissä kappaleissa käydään läpi prototyypin toiminnan testausta aktiivisuodatinmoduulin ohjausyksikkönä sekä arvioidaan kommunikaation suorituskykyä.

2 AKTIIVISUODATTIMET JA SÄÄTÖ

Sähköjärjestelmissä loisteho ja harmoniset virrat alentavat sähkön laatua, pienentävät verkon kapasiteettia ja aiheuttavat ylimääräisiä tehohäviöitä [13]. Harmoniset virrat ovat sinimuotoisia virtoja, joiden taajuus on sähköjärjestelmän perustaajuuden monikertoja [14]. Perinteinen harmonisten virtojen vaimennus ja loistehon kompensointi on toteutettu keloista, kondensaattoreista ja vastuksista koostuvilla LC-suotimilla, jotka kytketään epälineaarisen kuorman kanssa rinnakkain. LC-suodin on yksinkertainen ja edullinen ratkaisu, mutta harmonisten virtojen kompensoinnin ominaisuudet riippuvat verkon ominaisuuksista ja parametreista, kuten impedanssista ja taajuudesta. [15]

Aktiiviset tehosuodattimet ovat tehomuuntimia, jotka pystyvät vähentämään harmonisten virtojen osuutta sähköverkossa. Aktiivisuodattimen etuja ovat sopeutuminen verkkotaajuuden, jänniteamplitudin ja verkon impedanssin muutoksiin. Kuvan 2.1 kytkennässä on kuvattu aktiivisuodattimen kytkeytyminen verkkoon. Perusteoria voidaan tiivistää seuraavasti:

- epälineaarisen kuorman harmoninen virta i_L havaitaan
- tehomuunnin tuottaa kompensaatiovirran i_C, jolla on sama amplitudi ja vastakkainen polariteetti kuin harmonisella virralla
- kompensaatiovirta i_C sammuttaa harmonisen virran i_L, jolloin verkon virta i_g säilyttää sinimuotoisen peruskomponenttinsa.



Kuva 2.1. *Aktiivisuodattimen käyttö harmonisten virtojen suodatuksessa*

Aktiivisuodattimet voidaan luokitella kytkennän mukaan. Esimerkiksi kuvan 2.1 aktiivisuodatin on rinnankytketty epälineaarisen kuorman kanssa. Rinnankytketty aktiivisuodatin soveltuu kompensoimaan harmonisten yliaaltojen lähteitä, jotka havaitaan virtalähteenä. Sarjaankytketty aktiivisuodatin kytkeytyy verkon ja epälineaarisen kuorman väliin muuntajakytkennällä. Tällaista kytkentää käytetään kompensoimaan harmonisten yliaaltojen lähde, joka havaitaan jännitelähteenä. Sarjaankytketty aktiivisuodatin voi kompensoida harmonisten jännitekomponenttien lisäksi myös jännitteen epätasapainon verkossa tai kuormassa. Hybridimuotoinen aktiivisuodatin koostuu sarjaankytketystä aktiivisuodattimesta ja rinnankytketystä passiivisesta LC-suotimesta. LC-suodin huolehtii pääosin harmonisten yliaaltojen kompensoinnista ja aktiivisuodattimen rooli on parantaa LC-suotimen ominaisuuksia, sillä se muun muassa estää verkon ja rinnankytketyn LC-suotimen resonanssin. [15]

2.1 Rinnankytketyn aktiivisuodattimen säätö

Tässä kappaleessa esitellään myöhemmin prototyyppilaitteistolla toteutettavan loisvirtakompensoinnin referenssin laskenta. [15] mainitsee kolme metodia, joilla epälineaarisen kuorman harmonista sisältöä arvioidaan: Fourier-analyysi, Fryze power theory ja instantaneous reactive power theory. [15] kuvaa instantaneous reactive power -teorian kokonaisuudessaan, ja johtaa kolmivaiheisista verkkovirroista pätötehon virtankomponentin $i_p(t)$ ja loistehon virtakomponentin $i_q(t)$. [15]

Verkon jännitteet $u_{ga}(t)$,$u_{gb}(t)$,$u_{gc}(t)$ ja virrat $i_{ga}(t)$,$i_{gb}(t)$,$i_{gc}(t)$ voidaan muuntaa abc-koordinaatistosta $\alpha\beta$-koordinaatistoon, jos oletetaan, että järjestelmän jännitteet ja virrat eivät sisällä nollakomponentteja. Näin kolmivaiheisen verkon laskentaa saadaan yksinkertaistettua.

$$\begin{bmatrix} u_{g\alpha}(t) \\ u_{g\beta}(t) \end{bmatrix} = C_{32} \begin{bmatrix} u_{ga}(t) \\ u_{gb}(t) \\ u_{gc}(t) \end{bmatrix} \tag{2.1}$$

$$\begin{bmatrix} i_{g\alpha}(t) \\ i_{g\beta}(t) \end{bmatrix} = C_{32} \begin{bmatrix} i_{ga}(t) \\ i_{gb}(t) \\ i_{gc}(t) \end{bmatrix} \tag{2.2}$$

missä

$$C_{32} = \sqrt{2/3} \begin{bmatrix} 1 & -1/2 & -1/2 \\ 0 & \sqrt{3}/2 & -\sqrt{3}/2 \end{bmatrix} \tag{2.3}$$

Hetkellinen pätöteho saadaan $\alpha\beta$-koordinaatistoon muunnettujen jännitteiden ja virtojen tulojen summana. Hetkellinen loisteho on vastaavasti $\alpha\beta$-koordinaatistoon muunnettujen jännitteiden ja virtojen tulojen erotus. Matriisimuodossa esitettynä tehot lasketaan kaavan 2.4 mukaisesti.

$$\begin{bmatrix} p(t) \\ q(t) \end{bmatrix} = \begin{bmatrix} u_{g\alpha}(t) & u_{g\beta}(t) \\ u_{g\beta}(t) & -u_{g\alpha}(t) \end{bmatrix} \begin{bmatrix} i_{g\alpha}(t) \\ i_{g\beta}(t) \end{bmatrix} \tag{2.4}$$

Pätö- ja loistehon virtakomponentit saadaan kertomalla $\alpha\beta$-virtavektori sini-kosinimatriisilla C.

$$\begin{bmatrix} i_p(t) \\ i_q(t) \end{bmatrix} = C \begin{bmatrix} i_{g\alpha}(t) \\ i_{g\beta}(t) \end{bmatrix} \tag{2.5}$$

missä

$$C = \begin{bmatrix} \sin(\omega t) & -\cos(\omega t) \\ -\cos(\omega t) & -\sin(\omega t) \end{bmatrix} \tag{2.6}$$

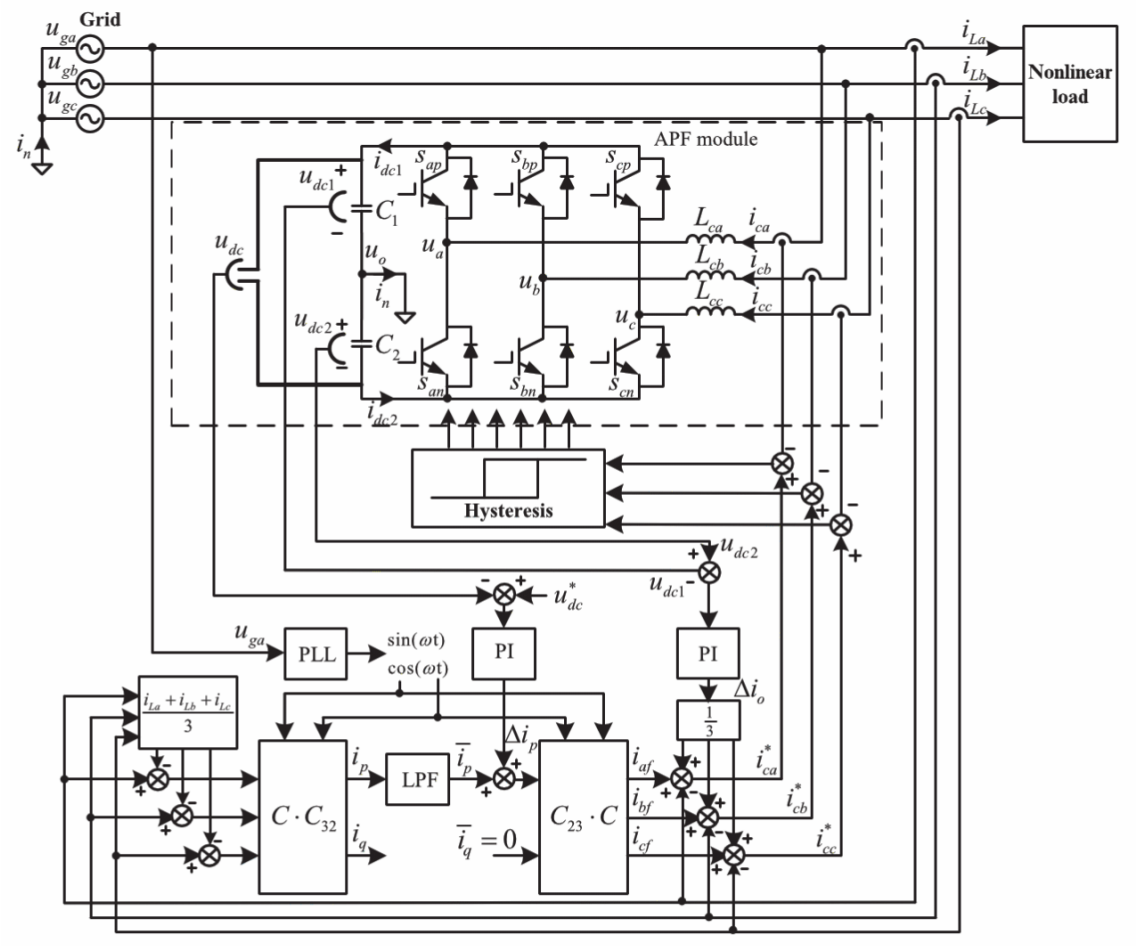
Yhdistämällä yhtälöt 2.2 ja 2.5 pätö- ja loistehon virtakomponentit voidaan laskea suoraan kolmivaiheisista virroista.

$$\begin{bmatrix} i_p(t) \\ i_q(t) \end{bmatrix} = CC_{32} \begin{bmatrix} i_{ga}(t) \\ i_{gb}(t) \\ i_{gc}(t) \end{bmatrix} \tag{2.7}$$

Kuvassa 2.2 on [15] esittämä lohkokaavio eräästä rinnankytketyn aktiivisuodattimen ohjausjärjestelmästä ja signaaleista. Ohjausjärjestelmä toteuttaa seuraavat toiminnallisuudet

- epälineaarisen kuorman virrat $i_{La}(t)$,$i_{Lb}(t)$,$i_{Lc}(t)$ mitataan

- PLL, eli vaihelukittu silmukka -yksikkö havaitsee verkkojännitteen $u_{ga}(t)$ vaihekulman
- järjestelmä laskee referenssivirrat $i^*_{ca}(t)$,$i^*_{cb}(t)$,$i^*_{cc}(t)$ kompensoinnille
- aktiivisuodattimen ulostulovirrat $i_{ca}(t)$,$i_{cb}(t)$,$i_{cc}(t)$ havaitaan ja ohjataan ohjauspiirille
- virtahystereesisilmukan ohjauspiiri tuottaa ulostuloonsa tehomuuntimen ajurin ulostulon vertaamalla kompensointivirtoja $i_{ca}(t)$,$i_{cb}(t)$,$i_{cc}(t)$ referenssivirtoihin $i^*_{ca}(t)$,$i^*_{cb}(t)$,$i^*_{cc}(t)$
- päävirtapiirin kompensointivirta seuraa referenssivirtaa, jolloin harmonisten yliaaltojen kompensointivirta ja loistehon virtakomponentti toteutuvat
- DC-puolen jännitetasoa $u_{dc}(t)$ säädetään PI-säätimellä, joka toteuttaa säätöä DC-jännitteen $u_{dc}(t)$ ja jännitereferenssin u^*_{Dc} erosuuretta säätämällä. Myös tehomuuntimen DC-tasapainoa säädetään PI-säätimellä, joka säätää DC-jännitteen ylä- ja alapuolen $u_{dc1}(t)$ ja $u_{dc2}(t)$ erosuuretta.



Kuva 2.2. *Rinnankytketyn aktiivisuodatinjärjestelmän lohkokaavio [15]*

[15]

3 TIEDONSIIRTOMENETELMÄT

Digitaalista dataa voidaan siirtää kahden pisteen välillä rinnakkaismuotoisesti tai sarjamuotoisesti. Rinnakkaismuotoisessa siirrossa kaikki bitit siirtyvät yhtä aikaa lähteestä määränpäähän, kun taas sarjasiirrossa bitit siirtyvät yksi kerrallaan. Rinnakkaismuotoinen siirto on siis nopeampaa, kun käytetään samoja kellonopeuksia, mutta vaatii myös oman kaistan jokaiselle siirrettävälle bitille. Siirtomatka täytyy myös säilyttää lyhyenä, muutamien metrien pituisena, jotta bittien saapumisaikaerot eivät ole merkittäviä. Rinnakkaissiirron vaatiman laitteiston takia rinnakkaissiirto onkin kalliimpaa sarjasiirtoon verrattuna. Rinnakkaismuotoiseen siirtoon vaadittava siirtotie voidaan toteuttaa joko moninapaisella kaapelilla tai vaihtoehtoisesti johtimille voidaan reitittää paljon pinta-alaa piirikortilta. Sarjasiirto puolestaan vaatii vähemmän piirikytkentöjä, sillä siihen tarvitaan vain yksi polku tai kaapeli, joten myös hinta on alhaisempi. [16]

Esimerkiksi [17] mukaan tiedonsiirtoväylät voidaan jakaa kategorioihin useiden ominaisuuksien perusteella. Väylät voidaan jakaa arkkitehtuurin mukaan synkronisiin ja asynkronisiin (synchronous vs. asynchronous), sekä multipleksattuihin ja ei-multipleksattuihin (multiplexed vs. non-multiplexed). Synkronisessa väylässä kaikki operaatiot tapahtuvat määritellyllä kellosignaalin reunalla. Asynkronisessa väylässä operaatiot tapahtuvat määritellyillä ohjaussignaalien reunoilla. Multipleksatussa väylässä data ja osoite jakavat saman linjan, jolloin väylä tarvitsee vähemmän johtimia. [17]

Sulautetuissa järjestelmissä tärkein ominaisuus on I/O:n suorituskyky, joka voi vaikuttaa negatiivisesti koko järjestelmän suorituskykyyn. I/O:n suorituskyvyn parantamiseksi piirikortin suunnittelija käy läpi useita I/O:n ja prosessorien kommunikaatiojärjestelmän vaihtoehtoja varmistaakseen jokaisen oheislaitteen toiminnan. Esimerkiksi hitaampien I/O-laitteiden synkronointi masterprosessorin kanssa voidaan toteuttaa tuomalla keskeytykset sekä statusliput kaikkien järjestelmän integroitujen piirien saataville. [18]

Yleisimmät I/O:n suorituskykyä mittaavat ominaisuudet ovat:

- Suorituskyky: Maksimimäärä dataa per aikayksikkö, joka pystytään käsittelemään. Esimerkiksi tavua per sekunti. Järjestelmän komponentti, jolla on alhaisin suorituskyky määrää koko järjestelmän suorituskyvyn.
- Suoritusaika: Kokonaisaika, joka I/O komponentilta kuluu kaiken datan käsittelyyn.
- Vasteaika: I/O komponentilta kuluva aika, joka kuluu datankäsittelypyynnön hetkestä prosessoinnin aloittamiseen.

[18]

Itse väylän suorituskykyä mitataan tyypillisesti sen kaistanleveyden avulla. Kaistanleveys kuvaa datamäärää, jonka väylä pystyy siirtämään tietyssä ajassa. Väylän fyysiset ominaisuudet sekä väylän kommunikaation protokollat vaikuttavat suorituskykyyn. Esimerkiksi mitä yksinkertaisempaa kättelymekanismia protokolla käyttää, sitä suurempi on väylän kaistanleveys. Fyysiset ominaisuudet, kuten pituus, rinnakkaisten kaistojen määrä ja yhdistettyjen laitteiden määrä joko rajoittavat tai parantavat väylän suorituskykyä. [18] Väylän pituus on verrannollinen siirtonopeuteen. Siirtonopeuksien ollessa vain muutaman megahertsin suuruisia, väylät pystyvät olemaan useiden metrien pituisia. Suuremmilla siirtonopeuksilla puolestaan väylien täytyy olla lyhyempiä, jotta etenemisviive ei vaikuta suorituskykyyn. Esimerkiksi PCI-väylä rajoittaa ajettavien väylien tehoa, ja siten rajoittaa liitettävien laitteiden määrää. [17] Edellä mainitut multipleksatut väylät puolestaan aiheuttavat viiveitä, sillä yhteen väylään yhdistetyt signaalit vaativat logiikkaa molemmissa tiedonsiirtotien päissä [18]. Vielä yksi kaistanleveyteen vaikuttava tekijä on se, kuinka monta databittiä väylä pystyy siirtämään yhden syklin aikana. Väylän kaistanleveys ilmoitetaan kahden potensseina. Esimerkiksi väylä, jonka leveys on 1 (2^0) siirtää 32 bitin paketin yksi bitti kerrallaan sarjamuotoisesti, kun taas väylä, jonka leveys on 8 (2^3) siirtää 32 bitin paketin neljässä erillisessä osassa, ja niin edelleen. Jokainen siirto-operaatio tuottaa viivettä, sillä jokaisen siirron kohdalla käydään läpi esimerkiksi kättelyoperaatio. [18]

3.1 Rinnakkaissiirto ja sen haasteet

Rinnakkaisväylää käyttävä datansiirto tarvitsee oman polun, eli piirilevyn johtimen tai kaapelin jokaista siirrettävää bittiä varten. Jos käytetään eromuotoista siirtoa, yksi bitti tarvitsee kaksi kulkureittiä. Vaikka rinnakkaismuotoinen siirto onkin nopea tiedonsiirtomenetelmä, se on kallis ja laitteistointensiivinen, sillä jokainen datapolku vaatii oman lähettimen sekä vastaanottimen. Siirtoetäisyyksien kasvaessa piirikortteja yhdistävistä johtimista tulee pitkiä ja kalliita. [19]

Pääasiassa rinnakkaisväylän käyttöä rajoittaa pitkät etäisyydet liitäntöjen välillä. Datan siirtonopeuksien kasvaessa siirtotien ominaisuudet määrittävät ylärajan siirtonopeuksille. Digitaalisten piirien nopeuden kasvaessa pienempien ja tehokkaampien prosessoreiden myötä rinnakkaissiirto useiden väylien yli on [16] mukaan saavuttanut ylärajansa. Siirtotien induktanssi, kapasitanssi sekä johtimien ylikuuluminen on kasvanut niin suureksi, ettei korkeampia siirtonopeuksia voida saavuttaa. [16]

Kirjassa [20] esitetään kaksi rinnakkaisväylän merkittävää ongelmaa edellä mainittujen lisäksi. Integroitujen piirien kehittymisen myötä yhdelle piirille mahdutettujen komponenttien määrä on kasvanut Mooren lain mukaan, mutta I/O pinnien tiheys piirien koteloinnissa ei ole kasvanut samassa suhteessa. Toinen [20] mainitsema ongelma liittyy ajoitusvaatimuksiin. Jos tiedonsiirto kahden piirin välillä toteutetaan synkronisesti

samalla kellosignaalilla, täytyy saapuvan datan täyttää signaalin asetus- ja pitoajan vaatimukset, mitkä määräytyvät piirin kellonsignaalin sisääntulon perusteella. Eli saapuvan datan tulee olla stabiili kellosignaalin näytteistyksen reunalla. Edellä mainituilla signaalien asetus- ja pitoajoilla tulee olla laskettuna riittävät toleranssit, jotta kellopiirin jakautumisen tuottavat viiveet voidaan ottaa huomioon. Viiveisiin vaikuttaa piirin tuotantoprosessi, jännite sekä lämpötila. Suuremmilla kellotaajuuksilla käytetään vaihelukittua silmukkaa (PLL) kellosignaalin vaiheen tasaamiseksi. PLL:n avulla voidaan mukautua muun muassa jännitteen ja lämpötilan aiheuttamiin olosuhteisiin. Mikäli kellotaajuus nostetaan riittävän korkealle, ei välttämättä ole mahdollista toteuttaa ollenkaan rinnakkaissiirtoon perustuvaa järjestelmää. [21]

Nämä kaksi mainittua rinnakkaisväylään liittyvää ongelmaa voidaan selättää muokkaamalla rinnakkaisväylän arkkitehtuuria. Näitä arkkitehtuuria muokkaavia muutoksia [21] kutsuu rinnakkaisväylän laajennuksiksi. Esimerkiksi väylän I/O-pinnejä voidaan multipleksata kapeampaan yhdyskäytävään ja näin vähentää siirtoon vaadittavaa pinta-alaa. Tästä aiheutuu toisaalta vaatimus korkeammasta referenssikellon taajuudesta. Yleisesti ottaen suuritaajuinen järjestelmäkello, joka jaetaan koko järjestelmälle, voi aiheuttaa kohinaa, sähkömagneettisia häiriöitä sekä tehohäviöitä. Kyseisiä ongelmia voidaan kiertää käyttämällä jälleen PLL-lohkoja, joilla referenssikello monistetaan muualle järjestelmään käyttökelpoiselle taajuudelle. [21]

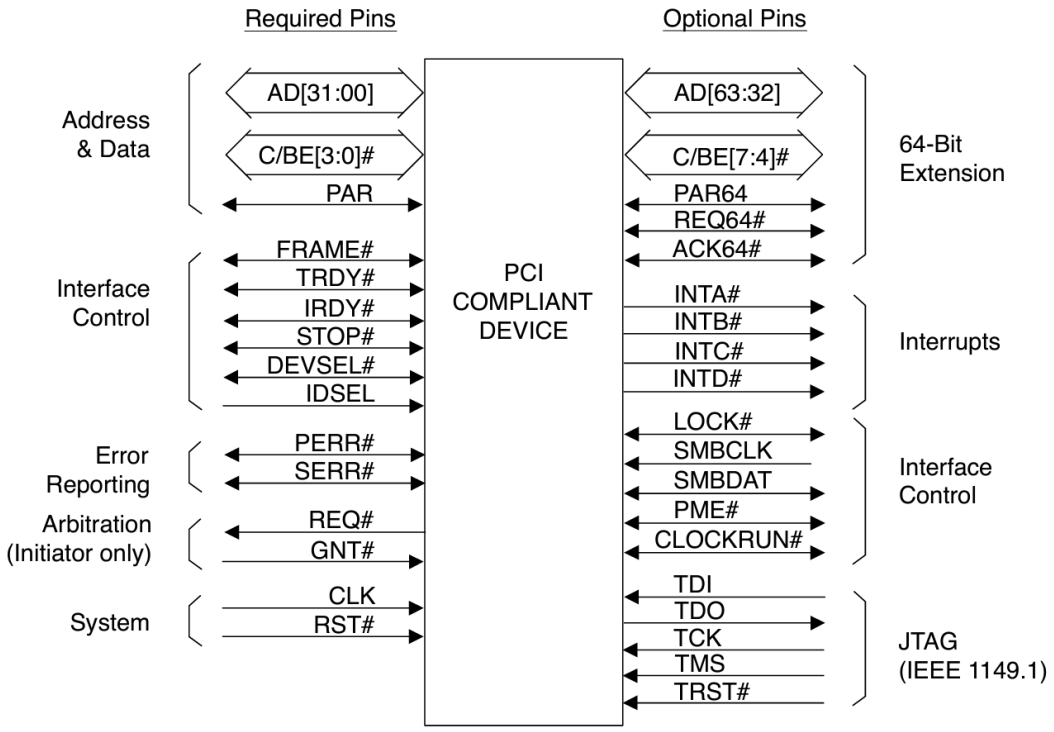
Siirtonopeuksien kasvaessa 2,5 Gbps:iin ja korkeammalle Serializer/Deserializer (HSS/SerDes) -laitteet ovat nykyään selkeästi hallitsevia I/O-toteutuksia nopeassa tiedonsiirrossa. SerDes-laitteet eroavat lähdesynkronisista käyttöliittymistä siten että dataa vastaanottava piiri sisältää kellon- ja datan palautuspiirin, joka määrittelee optimaalisen näytteistyksen kellosignaalin erottamiseen datan joukosta. SerDes käsitellään perusteellisemmin kappaleessa 4.3. [21]

Seuraavissa alakappaleissa esitellään rinnakkaisväylistä Peripheral Component Interconnect (PCI) sekä Texas Instrumentsin External Memory Interface (EMIF) ja Universal Parallel Port (UPP).

3.1.1 PCI

Intel kehitti alkuperäisen PCI-spesifikaation tuodakseen selkeyttä sekasortoisille markkinoille. [17] ilmoittaa pohjakonfiguraation siirtonopeuden teoreettiseksi maksimiksi 132 Mbytes/sec. Laajennukset PCI-pohjakonfiguraatioon voivat nostaa teoreettisen maksimin nelinkertaiseksi, eli 528 Mbytes/sec. PCI-X lisäys kasvattaa mahdollista siirtonopeutta yli neljään gigatavuun sekunnissa. [17]

Kuvassa 3.1 on esitelty PCI-standardin määrittelemät signaalit. PCI-liitäntä vaatii minimissään 47 pinniä kohdelaitteelle ja 49 pinniä masterlaitteelle. Pinnimäärä on riittävä 32 bittiseen 33 MHz:n dataväylään. Kuvan 3.1 64 bitin laajennus vaihtoehtoisissa pinneissä mahdollistaa 64 bitin datasiirron. [17]

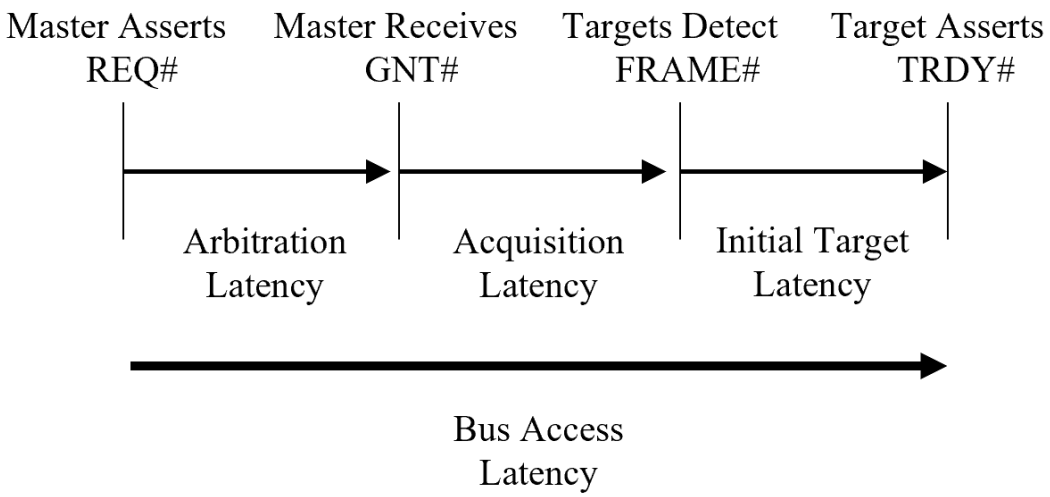


Kuva 3.1. *PCI standardin määrittelemät signaalit [17]*

PCI-väylän suorituskykyä parantaa ominaisuus, joka mahdollistaa useiden mastereiden samanaikaisen toiminnan. Väylässä jokainen master voi pyytää väylän käyttöönsä milloin vain. Ennen kuin master voi toteuttaa PCI-siirron, sen täytyy pyytää väylän käyttöoikeus. Tämä käyttöoikeus tarvitsee myös myöntää. Tätä tarkoitusta varten jokaisella väylän masterilla on request- ja grant-signaalit, joilla ne ovat suoraan yhteydessä välittäjälohkoon. Kun väylän master asettaa request-signaalin, kuluu äärellinen määrä aikaa, kunnes ensimmäinen dataelementti on siirretty. Tähän viitataan termillä bus access latency. PCI:n bus access latency koostuu kuvan 3.2 mukaisista komponenteista. [17]

PCI-spesifikaatio antaa suorituskyvyn optimoinnin työkaluksi viivelaskurin (spesifikaatiossa: latency timer), joka vaaditaan pursketta lähettävältä masterilta. Viivelaskurin tarkoitus on estää masteria käyttämästä väylää jatkuvasti itse, jos muilla mastereilla on tarve käyttää väylää. Viivelaskuriin ohjelmoitu arvo esittää kellosyklien minimiarvoa, jonka master saa aloittaessaan tiedonsiirron. Kun master asettaa frame-lipun, viivelaskuri alustetaan. Tämän jälkeen jokaisella kellopulssilla laskurin arvoa vähennetään. Jos laskuri saavuttaa arvon nolla, ennen kuin tiedonsiirto valmistuu ja masteri ei ole saanut grant-signaalia, toinen masteri tarvitsee väylää käyttöönsä ja nykyisen masterin tiedonsiirto keskeytyy. [17]

PCI-väylässä alhainen viive sekä korkean välityskyvyn tavoittelu on toteutettu kompromissina. Korkea välityskyky saavutetaan sallimalla liittyneille laitteille long burst transfer -moodi. Vastaavasti alhainen viive saadaan aikaiseksi vähentämällä purskeen maksimipituutta. [17] mukaan viiveitä analysoidessa hyödyllisintä on ottaa huomioon



Kuva 3.2. *PCI bus access latency*

Taulukko 3.1. *PCI viiveiden vertailu kaistanleveyden suhteen.*

Data Phases	Bytes Transferred	Total Clocks	Bandwidth Mb/s	Latency μs
8	32	16	60	0,48
16	64	24	80	0,72
32	128	40	96	1,20
64	256	72	107	2,16

siirrot, jotka ilmentävät väylän tyypillistä käyttäytymistä. Tyypillinen viive kellosykleinä voidaan ilmaista kaavan 3.1 avulla. Edellä mainitun viivelaskurin vaikutuksia ei seuraavassa kaavassa oteta huomioon. [17]

$$Latency_{typ} = 8 + (n - 1) + 1(clocks) \tag{3.1}$$

Kaavassa 3.1 8 kellosykliä on tyypillinen kesto kohdelaitteen *target ready* -lipun asettamiselle, *n* on siirto-operaatioiden lukumäärä, viimeinen yksittäinen kellojakso on toimeton sykli useimpien siirtojen välillä. [17]

Viivelaskuri ohjaa suuren välityskyvyn ja alhaisen viiveen välistä kompromissia. Taulukko kuvaa tätä kompromissia eri pituisten purskeiden välillä. Taulukon 3.1 tulokset perustuvat tyypillisen viiveen kaavaan 3.1. [17]

Total Clocks kertoo tiedonsiirron valmistumiseen vaadittavien kellosyklien määrä. Eli kyseessä on sama arvo kuin $Latency_{typ}$. Siirrettävän datan määrä per siirto kaksinkertaistuu, kun taulukkoa luetaan ylhäältä alas, mutta viive ei kaksinkertaistu. Datan määrä kahdeksankertaistuu alimmalle riville edetessä, mutta viive kasvaa lopulta vain 4,5 kertaiseksi. Tämä kertoo siitä että jokaiseen PCI-siirtoon liittyy myös yleisrasitetta *overhead*, ja mitä pidempi siirto on, sitä tehokkaampi väylä on. [17]

3.1.2 EMIF

External Memory Interface, EMIF, on Texas Instrumentsin asynkroninen rinnakkaisväylä. EMIF on suunniteltu käytettäväksi TI:n digitaalisilla signaaliprosessoreilla tai Digital Media System-on-Chip -järjestelmissä (DMSoC). EMIF-väylää käytettiin ensimmäisenä TI:n 32-bittisen C2000-sarjan mikrokontrollerilaitteissa, sillä osa C2000-sovelluksista vaativat suurempaa muistikapasiteettia kuin mitä piirin sisäinen muisti pystyi tarjoamaan. Väylän tarkoitus on siis tarjota yhteys ulkoisiin muistilaitteisiin kuten Flash- ja SDRAM-muistiin. Ulkopuolista lisämuistia voidaan käyttää esimerkiksi datavarastona tai käskymuistina. [22]

EMIF on yhteensopiva teollisuusstandardin muistilaitteiden kanssa ja tukee siten sekä synkronisia että asynkronisia muistikäyttöliittymiä. Asynkroninen muistikäyttöliittymä ei käytä signaalin kohdistamiseen yhteistä kellosignaalia, vaan yhdistelmälogiikalla toteutettuja ohjaussignaaleita. Esimerkiksi Static RAM (SRAM) ja pysyväismuisti kuten NOR flash ovat yleisiä asynkronista käyttöliittymää hyödyntäviä muistilaitteita. Myös esimerkiksi FPGA:t sekä digitaaliset lähetinvastaanottimet voivat käyttää EMIF-väylää chip-to-chip datansiirtoon. Ainoa EMIF-väylän tukema synkroninen muistikäyttöliittymä löytyy Synchronous Dynamic RAM:ille (SDRAM). Synkroninen käyttöliittymä käyttää yhteistä kellosignaalia ennalta määrätyillä ajoituksilla, jotka kertovat, milloin ohjaus- ja datasignaalit ovat käypiä. [22]

Väylän suorituskykyyn vaikuttavat useat rautaan liittyvät ja ohjelmalliset tekijät. Selkeät suorituskykyä rajoittavat tekijät, kuten siirtonopeus ja siirrettävän sanan pituus riippuvat väylään liitetystä muistilaitteesta. TI:n C2000 EMIF -käyttöohjeen [22] mukaan EMIF-väylään kytketyistä CPU-, DMA- ja CLA-laitteista jokainen kykenee käyttämään sekä 16-bittistä että myös 32-bittistä käyttötilaa dataoperaatioille. DMA eli Direct Memory Access on ominaisuus, joka sallii datansiirron oheislaitteen ja muistin välillä, ilman että prosessorin tarvitsee osallistua siirtoon. CLA eli Control Law Accelerator on TI:n C2000-prosessoreilta löytyvä itsenäinen raskaaseen laskentaan suunniteltu liukulukurautakiihdytin. 16-bittinen ja 32-bittinen käyttötila vaativat molemmat yhtä monta kellosykliä siirto-operaation toteutukseen, jolloin 32-bittinen käyttötila mahdollistaa suoraan paremman datanläpäisykyvyn. [22]

CPU-, DMA- ja CLA-yhteydet EMIF:een viedään normaalisti synkronointilohkon läpi, sillä kyseiset endpointit voivat toimia jokainen omalla kellotaajuudellaan. Tämä tuo väylän läpäisykykyyn viivettä, sillä jokainen EMIF-muistialueen kysely käy läpi kolme synkronointiviivettä tuottavaa järjestelmäkellon sykliä. Kyseistä viivettä pystytään minimoimaan, pitämällä synkronointilohko jatkuvasti täydessä kuormassa. Läpäisykykyä voidaan maksimoida käyttämällä muistilohkosiirtoja yksittäisten datasanojen siirron sijaan. TI:n suunnittelu- ja käyttöohjeet C2000 EMIF-käyttöliittymälle [22] vertailee väylän suorituskykyä eri käyttölaitteiden välillä. Vertailussa, esimerkiksi DMA:n ja CLA:n asynkroninen kirjoitus EMIF-väylään tapahtuu siirrettävän datalohkon koosta riippumatta lähes 115 MB/s nopeudella, mutta CPU:n kirjoituksen suorituskyky kärsii pienemmillä

datalohkoilla. Asynkronisessa lukuoperaatiossa puolestaan CPU suoriutuu parhaiten lähes 100 MB/s siirtonopeudella datalohkon koosta riippumatta CLA:n suoriutuessa lähes yhtä hyvin. [22]

3.1.3 UPP

Universal parallel port (uPP) on Texas Instrumentsin digitaalisille signaaliprosessoreille kehitetty monikanavainen suurinopeuksinen rinnakkaisväylä. uPP on suunniteltu erityisesti suurinopeuksisille ADC- tai DAC-muuntimille. Väylä pystyy [23] mukaan 75 MHz siirtonopeuteen 8-16 bitin väylänleveydellä per kaista. Siirtonopeutta rajoittaa DSP:n maksimikellotaajuus. TI:n uPP-käyttöoppaan [23] mukaan väylä voidaan yhdistää FPGA-piireihin tai muihin uPP-laitteisiin. Ominaisuuksissa mainitaan myös duplex-moodi, eli datan lähetys ja vastaanottaminen voivat tapahtua rinnakkain, jos lähetykselle ja vastaanotolle on käytössä omat kanavat. uPP käyttää yksinkertaista 2-4 ohjauspinnin säätöprotokollaa. Läpäisykyvyn maksimoimiseksi KeyStone-arkkitehtuurin piirit käyttävät [23] mukaan sisäistä DMA:ta datan lähettämisen tai vastaanottamiseen I/O-kanavista. Laitteesta löytyy kaksi erillistä kelloa, joista toinen on moduulin kello, joka ohjaa sisäistä logiikkaa, toinen kello on datansiirtoon käytettävä kello, joka pyörii käyttöliittymäkanavassa lähetysmoodissa. uPP hyödyntää Single Data Rate (SDR) ja Double Data Rate (DDR) kellottamisjärjestelmiä. SDR-järjestelmä hyväksyy datapinnien datan kellosignaalin nousevalla tai laskevalla reunalla. DDR-moodissa data hyväksytään sekä nousevalla- että laskevalla reunalla. [23]

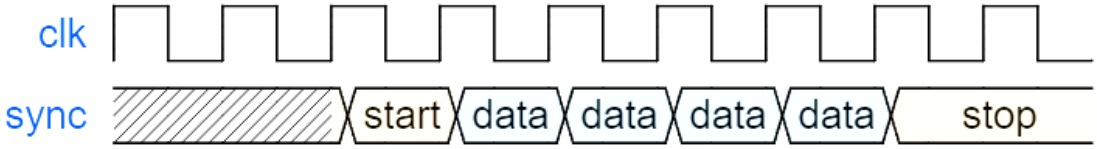
3.2 Sarjamuotoinen datansiirto

Digitaalipiirien nopeuksien kasvaessa pienempien ja nopeampien integroitujen piirien myötä rinnakkaissiirto useiden väylien yli on saavuttanut [16] mukaan suorituskykyrajansa. Suurimmat rajoitukset korkeammille siirtonopeuksille johtuvat suurten piirikorttien väyläjohtimien ylikuulumisesta sekä hajainduktanssista ja hajakapasitanssista. Nopeampien mikropiirien myötä sarjasiirrosta on tullut käytännöllisempää. Myös piirit ja kaapelit ovat halvempia ja yksinkertaisempia, kun käytetään sarjasiirtoa. [16]

Tiedonsiirtomenetelmät voidaan jakaa asynkroniseen ja synkroniseen siirtoon [16]. Asynkronisessa siirrossa lähettäjä ja vastaanotin on viritetty samalle nimelliselle bittitaajuudelle. Mikäli lähettäjän ja vastaanottimen bittitaajuudet eivät pysy täysin samana, täytyy bittiaikojen olla suuremmat, eli kovin pitkää datajonoa ei voida siirtää, sillä näytteistyskohta ajautuu väärään paikkaan. Esimerkiksi Universal Asynchronous Receiver Transmitterin eli sarjaliikennepiiri UART:n vastaanottimen kellon tulee vastata samaa baudinopeutta eli elektronisen signaalin muutosnopeutta per sekunti.[24] Asynkronisessa siirrossa jokaista siirrettävää merkkiä edeltää yksi aloitusbitti ja synkronointiin käytetään yhtä tai kahta lopetusbittiä [16]. Vaihtoehtoisesti voidaan käyttää vielä pariteettibittiä, jota varsinkin vanhoissa kohinaisissa järjestelmissä

käytettiin virheellisen siirtojen havaitsemiseen [24]. Synkronoinnin tarve johtuu satunnaisesta intervallista merkkien välillä. Eli hetken toimettomana olleen vastaanottimen pitää tietää, milloin uusi data otetaan vastaan. Kuvassa 3.3 on esitelty asynkroninen tiedonsiirto ja kuvassa 3.4 on esitelty synkroninen tiedonsiirto. [25]

Kuva 3.3. *Asynkroninen tiedonsiirto*



Kuva 3.4. *Synkroninen tiedonsiirto*

Synkronisessa siirrossa bitit puolestaan näytteistetään erillisen kellosignaalin avulla, joka synkronoidaan dataan. Tämä mahdollistaa suurempien datajonojen siirtämisen kerralla, kun väylällä ei ole tarvetta tahdistaa siirtoa aina, kun kellot ajautuvat erilleen. [25]

Tiedonsiirto datalähteen ja vastaanottajan välillä voidaan jaotella myös tiedonsiirron suunnan perusteella. Siirtometodeja on kolmea tyyppiä:

- simplex
- half duplex
- full duplex

Simplex-käsitteellä tarkoitetaan tiedonsiirtomenetelmää, jossa data liikkuu vain yhteen suuntaan lähettäjän ja vastaanottajan välillä. Half-duplex-käsitteellä tarkoitetaan tiedonsiirtomenetelmää, jossa data liikkuu molempiin suuntiin, mutta ei ikinä samanaikaisesti. Tässä menetelmässä lähettäjän ja vastaanottajan roolit ovat vaihdettavissa siten, että data voi liikkua kumpaankin suuntaan. Full-duplex-käsitteellä tarkoitetaan tiedonsiirtomenetelmää, jossa data liikkuu molempiin suuntiin samanaikaisesti. Full-duplex vaatii omat linjat lähettäjän ja vastaanottajan välille. [25]

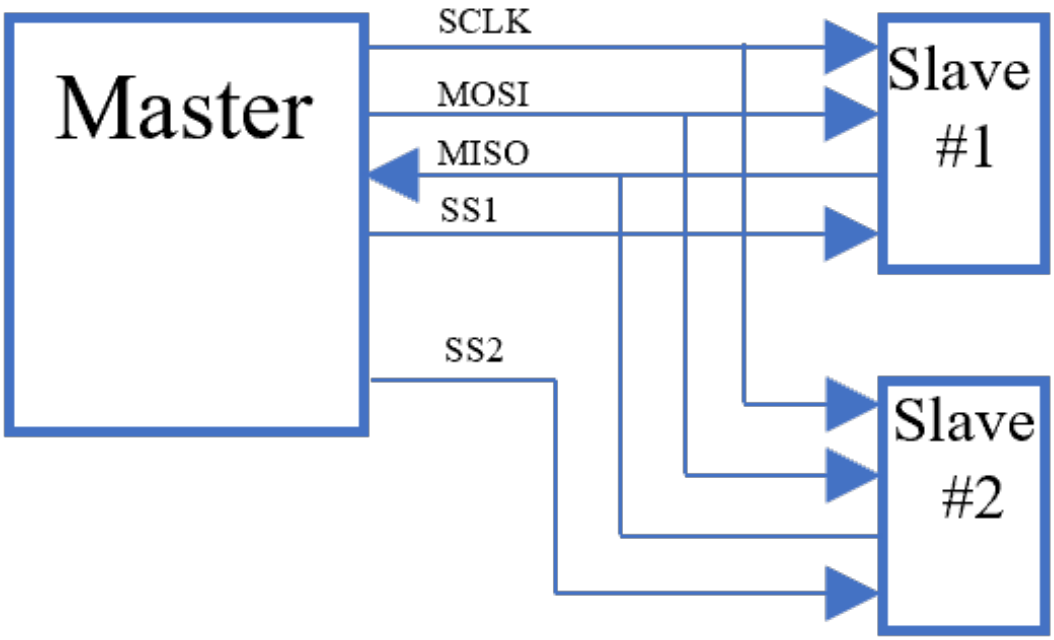
Seuraavat kappaleet esittelevät Serial Peripheral Interface -väylän (SPI) sekä I2C-väylän.

3.3 Serial Peripheral Interface

Serial Peripheral Interface, eli SPI-väylä on tavanomaisesti Chip-to-chip liitännöissä käytetty synkroninen full duplex -siirtoon kykenevä sarjaväylä. Tyypillisiä käyttökohteita SPI-väylälle ovat muun muassa mikrokontrollerille liittyvät laitteet, kuten ADC- ja DAC-muuntimet, näytöt ja sensorit. [19] arvioi SPI-väylän tyypilliseksi tiedonsiirtonopeudeksi 20-100 Mb/s.

SPI-väylään liitettyjä laitteita kutsutaan nimillä SPI master ja SPI slave. Laite, joka tuottaa kellosignaalin on väylän master. Väylän data synkronoidaan kellon nousevalla tai

laskevalla reunalla. SPI-liitännässä voi olla kolme tai neljä johdinta. Näistä suositumpi on nelijohtoinen liitäntä. SPI-väylän signaalit on esitetty kuvassa 3.5. Nelijohtimisesta väylästä löytyy seuraavat signaalit:

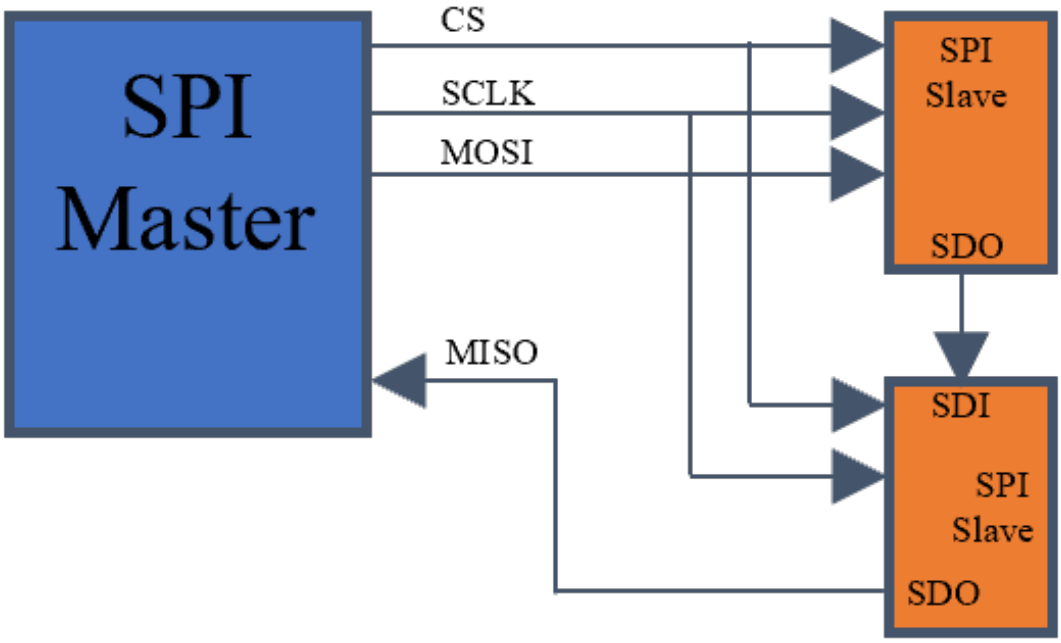


Kuva 3.5. *SPI väylän signaalit*

- Clock (SPI CLK, SCLK)
- Chip select (CS), Slave select (SS)
- Master out, slave in (MOSI)
- Master in, slave out (MISO)

Chip select -signaali valitsee usean slaven käyttötapauksessa laitteen. MOSI- ja MISO-signaalit ovat datalinjoja, joiden välityksellä siirrettävä data liikkuu laitteiden välillä. SPI-kommunikaatio alkaa masterin lähettäessä kellosignaalin ja asettaessa Chip select -signaalin. SPI-väylä toimii full-duplex käyttöliittymänä, eli master ja slave voivat molemmat lähettää dataa samaan aikaan omia datalinjojaan pitkin. Kellosignaali synkronoi siirrettävän ja vastaanotettavan datan. [26]

Perinteisessä SPI moodissa, kuten kuvassa 3.5, jokainen väylään yhdistetty slave vaatii oman chip select -signaalin masterilta. Koska slave-laitteet käyttävät yhteisiä datalinjoja, vain yksi slave voi olla kerrallaan aktiivisena, jotta datan korruptoitumiselta voidaan välttyä. Useat slave-laitteet lisäävät masterin vaatimia sisäänmenoja ja ulostuloja. Kuvan 3.6 daisy chain -moodissa chip select -signaali on kytketty kaikkiin slave-laitteisiin ja MISO-linja on ketjutettu kaikkien slave-laitteiden läpi. Toisaalta tällä metodilla siirtoon vaadittavien kellosyklien määrä on verrannollinen slave-laitteen positioon laiteketjussa. [26]



Kuva 3.6. *SPI daisy chain*

3.4 I2C

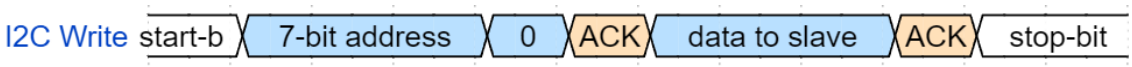
I2C, toiselta nimeltään IIC eli Inter-Integrated Circuit tai TWI (Two-Wire Interface), on alun perin Phillips Semiconductorin kehittämä väyläprotokolla. I2C on kätevä rautatason järjestelmissä, sillä se vaatii vain kaksi johdinta kommunikaation toteutukseen. Kommunikaatiossa käytettävät signaalit ovat:

- System Clock (SCL)
- System Data (SDA)

Jokainen väylää käyttävä laite voi tuottaa datasignaalin vetämällä erillisen jännitelinjan alas. Ylösvetovastus vetää jännitelinjan jännitteen takaisin ylös väylän ollessa joutilaana. Kuten kappaleen 3.3 SPI-väylä, I2C protokolla voi käyttää yhtä master-laitetta ja useita slave-laitteita. Väylän toiminnan aloittaa aina master ajamalla kellosignaalia ja slavelaitteet vastaavat vain silloin kun niille on tehty kysely. I2C eroaa SPI-väylästä 7-bittisen osoitejärjestelmän muodossa. Jokaisella slavelaitteella on uniikki 7-bittinen osoite, jolla liikenne erotellaan. I2C:n siirto-operaatioiden toiminta perustuu SCL ja SDA linjojen tiloihin. I2C-siirto alkaa SCL-linjan ollessa ylhäällä, jonka jälkeen SDA-linjan tila muuttuu ykkösestä nollaksi. Kun linja on levossa, riittää että SDA:n tila vaihtuu ylhäältä alas. Repeated start -ehto pitää väylän samalla slave-laitteella, jotta voidaan siirtää pidempiä purskeita. Datalinja SDA:n tila luetaan SCL-linjan laskevalla reunalla. [27]

3.4.1 I2C siirto-operaatiot

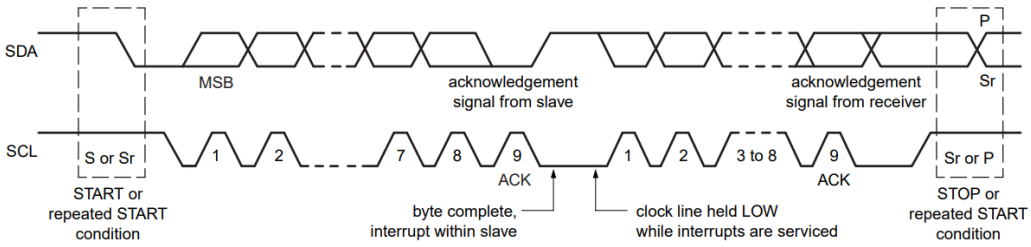
I2C käyttää 7-bittisen osoitejärjestelmän lisäksi kirjoitus- ja lukubittiä, joka kertoo slavelaitteelle onko kyseessä kirjoitus- vai lukuoperaatio. Näin muodostuu 8-bittisen tavun pituinen osoite, jossa LSB:n paikalla sijaitsee R/W-bitti. Kuvat 3.7 ja 3.8 havainnollistavat kirjoitus- ja lukuoperaatioiden toiminnan. Siirto alkaa start-bitillä, jota seuraa 7-bittinen osoite ja kirjoitus- tai lukubitti. Protokollan mukaan slavelaitteen tulee hyväksyä pyyntö vetämällä datalinja alas. Tämä tulkitaan ACK- eli acknowledge-bittinä. Datalinja jää ylös, jos slave ei vastaa. Tämä tulkitaan NAK-bittinä eli negative acknowledge-bittinä. [27]



Kuva 3.7. *I2C:n kirjoitusoperaatio*

Kuva 3.8. *I2C:n lukuoperaatio*

Kuva 3.9 esittää kokonaisuutena siirto-operaation vaiheet kello- ja datalinja rinnakkain. Start- tai repeated start -ehto aloittaa siirron. Uusi databitti siirtyy SCL-kellolinjan laskevalla reunalla, ja kirjoitus- ja lukupyynnön toteutuminen ilmoitetaan ACK/NACK-bitillä. Siirto-operaatio päättyy stop tai repeated start -ehtoon. [27]



Kuva 3.9. *I2C:n datasiirto [28]*

3.4.2 I2C-väylän suorituskyky

I2C-väylän speksin 6. versio [28] jakaa väylän eri moodeihin suorituskyvyn mukaan. Väylän siirtokyky ja toimintamoodit on esitetty taulukossa 3.2. Alun perin väylän datansiirtokyky rajoittui nopeuteen 100 kbit/s, mutta ajan myötä speksiin on tehty lisäyksiä, jotka mahdollistavat nopeamman ja monipuolisemman toiminnan. [28]

Fast mode -I2C-väylään on teknologian kehityksen myötä tehty seuraavia oleellisia lisäyksiä:

- Fast mode -yhteensopivien laitteiden sisäänmenoihin on sisällytetty piikkien vaimennus sekä Schmitt trigger -ominaisuus, joka sisäänmenon tilanvaihdon

Taulukko 3.2. *I2C-moodien datansiirtokyky*

I2C-moodi	siirtonopeus
Standard-mode	100 kbit/s
Fast-mode	400 kbit/s
Fast-mode Plus	1 Mbit/s
High-speed mode	3.4 Mbit/s
Ultra Fastmode	5 Mbit/s

jälkeen pyrkii säilyttämään tilansa.

- Ulostulobuffereihin on sisällytetty kaltevuuden ohjaus SDA- ja SCL-signaalien laskevalle reunalle.
- Ulkopuolisten väylään liitettyjen pull-up-laitteiden tulee pystyä sopeutumaan lyhyempään signaalin nousuaikaan.

Fast mode plus sekä high-speed mode käyttävät komponentteja, jotka kykenevät korkeaan virranajoon sekä pystyvät vastaamaan pienempiin signaalin nousu- ja laskuajan vaatimuksiin saavuttaen jälleen nopeampia datansiirtonopeuksia vanhempiin standardeihin verrattuna. [28]

Ultra Fast mode I2C (UFm I2C) toimii yksisuuntaisesti ja on [28] mukaan hyödyllisin ajamaan LED-ajureita ja muita laitteita, jotka eivät tarvitse palautesignaalia. UFm I2C perustuu standardiprotokollaan, mutta R/W-ohjausbitti on muutettu vain kirjoitusbitiksi ja ajoituskaavion 3.9 yhdeksäs bitti on ajettu jatkuvasti ylös, jolloin ACK-sykli ohitetaan väylän yksisuuntaisen luonteen takia. UFm-laitteet eivät ole yksisuuntaisen siirron takia taaksepäin yhteensopivia. [28]

4 KORKEAN SUORITUSKYVYN SARJAMUOTOISET VÄYLÄT

Tiukemmat vaatimukset datansiirron nopeuden kasvuun ovat pakottaneet kommunikaatioväyliä kehittymään tehokkaammiksi ja vähemmän tehoa kuluttaviksi. Merkittävä siirtymä rinnakkaisväylistä sarjaväyliin on vähentänyt laitteiden vaatimaa pinta-alaa. Kappaleessa käsitellään lähetinvastaanottimien ominaisuuksia, jotka mahdollistavat suurinopeuksisten sarjamuotoisten kommunikaatiolinkkien toiminnan. Luvussa käydään myös läpi merkittäviä korkean suorituskyvyn väyliä, joita harkittiin vaihtoehdoiksi Merus Powerin säätöjärjestelmän prototyypin tiedonsiirtoväyläksi.

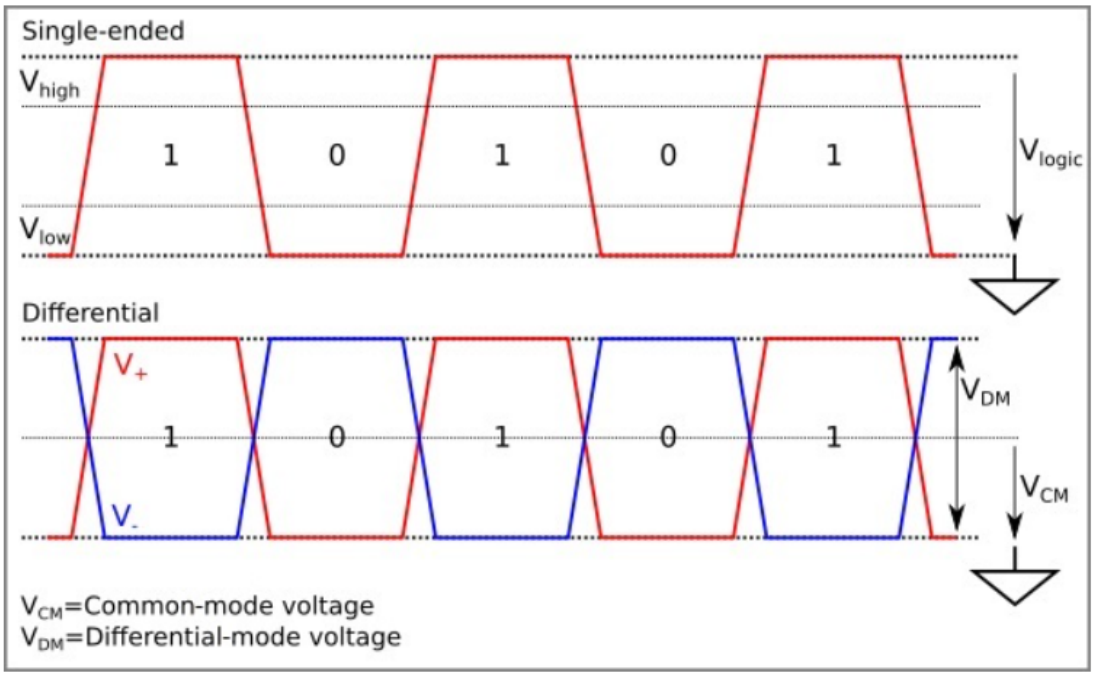
4.1 Lähetinvastaanotinteknologiat

[29] mukaan vuosituhannen alun järjestelmät käyttivät lähetinvastaanottimia, jotka pystyivät tukemaan I/O-nopeuksia 3,125 Gbps:iin asti. Näillä nopeuksilla lähetinvastaanottimen lohkon tulee sisältää teknologiaa, jolla voidaan varmistaa tarkka datansiirto useiden medioiden läpi. [29] esittelee lähetinvastaanottimien kehitystä sekä niiden soveltamista eri käyttötarkoituksissa.

4.1.1 Yksipuoliset ja eromuotoiset signaalit

Eromuotoisia signaaleita on perinteisesti käytetty pitkille siirtomatkoille chip-to-chip-kommunikaation sijasta. Integroitujen piirien nopeuksien kasvaessa, syntyi tarve suurista nopeuksista selviävälle signalointimenetelmälle. Differentiaalinen signalointi on esimerkiksi vähemmän altis kohinalle, sekä erityisesti vähemmän altis yhteysmuotoiselle häiriölle [30] verrattuna yksipäiseen single-ended-signalointiin. Menetelmä vertaa kahta signaalia toisiinsa. Eli jos positiivisen solmun jännite on suurempi kuin negatiivisen solmun signaali, on signaali silloin ylhäällä tai yksi. Negatiivisen solmun signaalin ollessa positiivisempi kuin positiivisen solmun signaali, signaali on alhaalla tai nolla. Differentiaalisen signaalin toimintaa havainnollistaa kuva 4.1. [30]

Teollisuuden vaatimukset kaistanleveydelle ja nopeudelle ovat vaatineet jatkuvaa I/O-suunnittelun kehitystä. Gigabit Serial I/O:lla tarkoitetaan vähintään yhden gigabitin nopeuksilla toimivaa I/O:ta. Suurinopeuksinen sarjalinkki mahdollistaa tehokkaan datan liikuttelun piirikorttien välillä. Etuja tuovat muun muassa myös pienempi pinnimäärä,



Kuva 4.1. *Yksipuolisen ja differentiaalisen signaalin ajoituskaavio [31]*

alhainen EMI ja pienempi hinta. Multigigabit lähettimet (MGT) pystyvät liikuttamaan dataa nopeasti muuntamalla rinnakkaismuotoisen datan sarjamuotoiseksi ja siirtämällä sen sarjalinkin ylitse. [30]

CMOS-logiikkaan perustuva Low Voltage Differential Signaling (LVDS) I/O-standardi tarjoaa korkean datansiirtonopeuden matalalla kohinalla, EMI-suojauksella ja matalalla tehontarpeella. Käyttökohteita LVDS:llä on muun muassa suurta kaistanleveyttä vaativissa tiedonsiirtosovelluksissa, kuten taustalevyjen lähetinvastaanottimissa tai kellosignaalin jakelusovelluksissa. Syy LVDS:n valintaan sovelluksissa on usein sen matala 350 mV:n jännite-ero johtimien välillä. LVDS on myös energiatehokas standardi. Sen AC-teho on alhainen, sillä signaalin kytkentäjännite on pieni. Tämä johtaa pieniin häviöihin per signaalin siirtymä. LVDS luo ympärilleen myös matalampaa sähkömagneettista säteilyä (EMI), sillä laitteen luoma EMI riippuu taajuudesta, ulostulojännitteen vaihtelusta sekä muuttumisnopeudesta. [29]

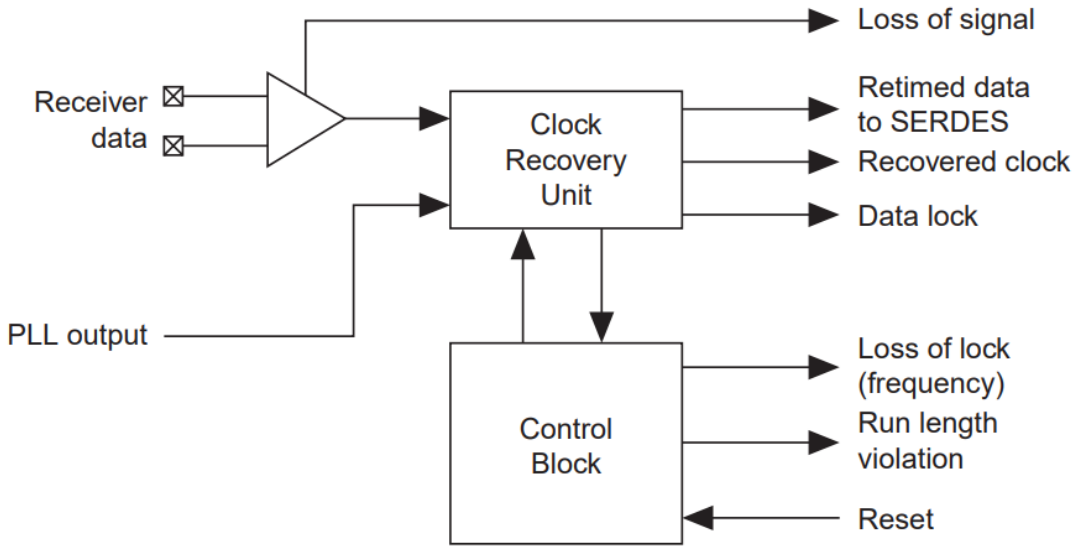
[29] mukaan LVDS:n määrittelee kaksi samankaltaista teollisuusstandardia. IEEE 1596.3 tukee datanopeuksia 250 Mbps:n asti. ANSI/TIA/EIA-644 puolestaan suosittelee korkeampia datanopeuksia 655 Mbps asti ja esittää teoreettiseksi maksimiksi 1.923 Gbps. [29]

Muita eromuotoisia signalointiteknologioita ovat esimerkiksi Low Voltage Positive Emitter Coupled Logic (LVPECL), jota on käytetty videografiikka-, datakommunikaatio- ja telekommunikaatiosovelluksissa. Pseudo Current Mode Logic (PCML) on käytössä esimerkiksi datakommunikaatio- ja verkkosovelluksissa. HyperTransport Technology I/O puolestaan kehitettiin chip-to-chip käyttöliittymien käyttöön tietokoneisiin perustuvissa

laitteissa. [29]

4.1.2 Kellosignaalin palautus

Kellosignaalin palautus eli Clock Data Recovery (CDR) tarjoaa datan eheyden varmistamiseksi tekniikan, jolla kellosignaali saadaan sulautettua datan joukkoon sekä eroteltua datavirrasta vastaanottimessa. Lähetyspiiri koostuu rinnakkais-sarjamuuntimesta ja synkronointilohkosta. Synkronointilohko ottaa kellosignaalin ja käyttää sitä rinnakkais-sarjamuuntamaan datan. Tämä kellosignaali sulautetaan datasignaaliin ennen lähetystä. Vastaanotin puolestaan koostuu Clock Recovery Unitista (CRU) ja sarjarinnakkaismuuntimesta. Vastaanotettu data syötetään CRU-yksikköön, joka laskee kellon ja sen vaiheen datan muutoksista. Lohkokaavio prosessista on esitetty kuvassa 4.2. [29]



Kuva 4.2. *Clock Data Recovery Block Diagram [29]*

Kellonopeus määritellään CRU:n toimesta datan siirtymätilojen avulla. Erillisen kellosignaalin puuttuessa siirtymät ykkösestä nollaan ja nollasta ykköseen datassa itsessään täytyy päätellä, jotta kello voidaan palauttaa. CDR-kommunikaatio paljastaa yleensä datasyklin jakson pituuden maksimin tai siirtotiheyden minimin. CRU jatkaa kellosignaalin ajamista edellisen datasiirron nopeudella, kun datansiirrossa tapahtuu pysähdys. Tämä voi aiheuttaa sen, ettei data ole enää välttämättä synkronoitu kelloon. Ongelma voidaan välttää käyttämällä esimerkiksi 8b10b-enkoodausta. Vaikka nämä enkoodaustavat tuovat jossain määrin lisää yleisrasitetta datasignaaliin, rasite on yleensä kuitenkin vähäistä verrattuna CDR:n tuomaan suorituskyvyn lisäykseen. [29]

Datan enkoodausta käytetään datansiirrossa, sillä se helpottaa paljon kellosignaalin synkronointia, DC-balansointia, ja virheenkorjausta. Käytetyimpiä enkoodausjärjestelmiä ovat 2000-luvun alussa olleet esimerkiksi 5b6b, 8b10b ja sen monikerrat, mukaan lukien 64b66b [29]. Protokollat, kuten 10 gigabit Ethernet, Fibre Channel ja Infiniband käyttävät

Taulukko 4.1. *Esimerkki 8b10b-symboleista*

Name	Hex	8 Bits	RD-	RD+
D10.7	EA	11101010	0101011110	0101010001
D31.7	FF	11111111	1010110001	0101001110
D4.5	A4	10100100	1101011010	0010101010
D0.0	00	00000000	1001110100	0110001011
D23.0	17	00010111	1110100100	0001011011

8b10b enkoodausta, mikä tekee enkoodauslohkon liittämisen suoraan lähetinvastaanottimeen mahdolliseksi vaihtoehdoksi monille lähetinvastaanottimien valmistajille. Toiminnon integrointi lohkoon mahdollistaa jatkuvan nopean siirron, sillä enkoodausta ei tarvitse käsitellä ohjelmallisesti. [29]

4.2 Linjakoodaus

Linjakoodausmenetelmät muokkaavat dataa, jotta siirtolinkissä ei olisi DC-komponenttia. Linjakoodausmenetelmät varmistavat myös sen, että kellosignaalin palautuspiiri saa riittävästi siirtymistiloja toimiakseen. Lisäksi menetelmät voivat myös mahdollistaa clock correction -toiminnon, lohkosynkronoinnin, kanavien yhdistämisen sekä kaistan jakamisen osakanaviin. [32] jakaa linjakoodausmenetelmät kahteen kategoriaan, value lookup -menetelmään ja self-modifying-virtaan tai datamuokkaimiin (scramblers). [32]

4.2.1 8b10b

IBM kehitti laajasti hyödynnetyn 8b/10b enkoodausmenetelmän. 8b10b-enkoodausta käyttää muun muassa Infiniband, Gigabit Ethernet, FiberChannel sekä 10 Gigabit Ethernet -käyttöliittymä XAUI. Menetelmä toimii siten, että 8 bittiset sanat käännetään 10 bittisiksi symboleiksi, jotka varmistavat kellosignaalin palautusyksikölle riittävän määrän siirtymätiloja. [32]

DC-balansointi saavutetaan 8b10b-linjakoodauksessa metodilla, jonka nimi on juokseva poikkeama. 8b10b käyttää kahta erilaista symbolia jokaiselle datan arvolle. Helpoin keino olisi sallia vain symbolit, joissa on sama määrä ykkösiä ja nollia, mutta se rajoittaisi liikaa käytettävien symbolien määrää. Useimmissa tapauksissa yhdessä symbolissa on kuusi nollaa ja neljä ykköstä, ja toisessa neljä nollaa ja kuusi ykköstä. Ykkösten ja nollien kokonaismäärästä pidetään kirjaa, ja seuraava symboli määräytyy sen mukaan, mitä DC tasapainon saavuttamiseksi tarvitaan. Näihin symboleihin viitataan normaalisti symboleilla + ja -. Taulukko 4.1 esittelee muutaman muunnosesimerkin. [32]

Myös vastaanotin voi monitoroida juoksevaa poikkeamaa ja havaita saapuvasta datavirrasta, onko poikkeamasääntöä rikottu.

8b10b käyttää myös 12 erikoismerkkiä, jotka tulkitaan 12:sta ohjausmerkiksi, joita

Taulukko 4.2. *Pätevät ohjauksen K-merkit*

Name	Hex	8 Bits	RD-	RD+
K28.0	1C	00011100	0011110100	1100001011
K28.1	3C	00111100	0011111001	1100000110
K28.2	5C	01011100	0011110101	1100001010
K28.3	7C	01111100	0011110011	1100001100
K28.4	9C	10011100	0011110010	1100001101
K28.5	BC	10111100	0011111010	1100000101
K28.6	DC	11011100	0011110110	1100001001
K28.7	FC	11111100	0011111000	1100000111
K23.7	F7	11110111	1110101000	0001010111
K27.7	FB	11111011	1101101000	0010010111
K29.7	FD	11111101	1011101000	0100010111
K30.7	FE	11111110	0111101000	1000010111

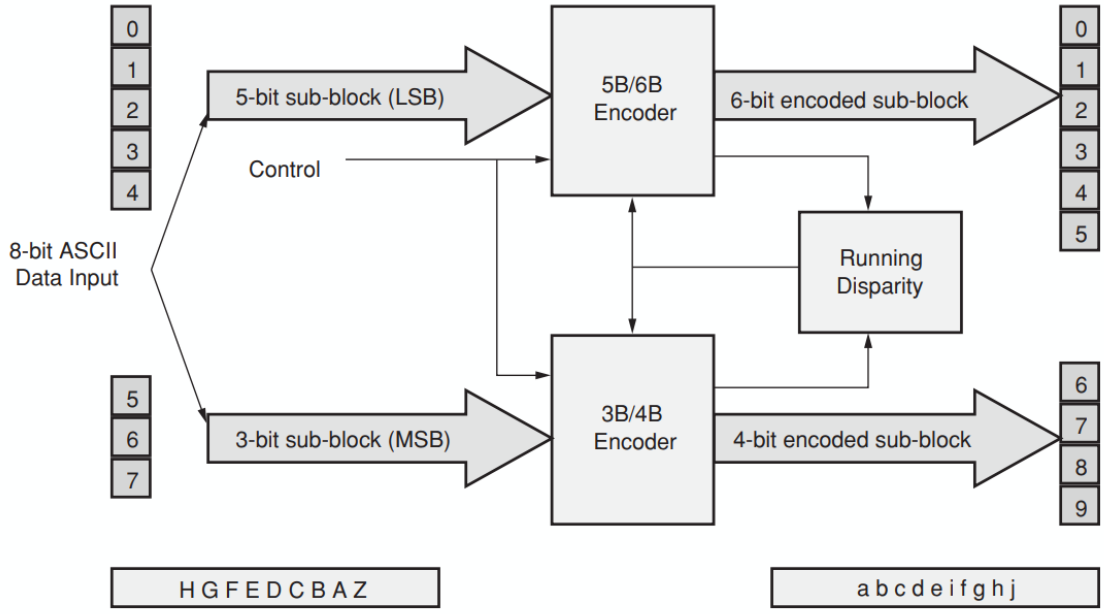
kutsutaan K-merkeiksi (K-character). Näitä merkkejä käytetään datan ryhmittelyyn, ohjaukseen ja kaistan jakamiseen alakanaviin. Taulukko 4.2 sisältää kaikki pätevät K-merkit. [32]

Sarja-rinnakkaismuunnin selvittää vastaanotettujen symbolien rajat comma detection toiminnon avulla. Jokainen 10-bittinen merkkijono on rajattu comma-merkillä. 8b10b:n tapauksessa comma on yksi tai kaksi symbolia, jotka on määritelty comma -merkiksi tai ryhmitelty jonoksi. Vastaanotin skannaa saapuneen datavirran ja tarkistaa, löytyykö siitä ennalta määrätty bittijono. Bittijonon löytyessä sanan rajat resetoidaan vastaamaan havaittua comma-sarjaa. Comma-merkin protokolla tulee olla ennalta määrätty sillä määrätty comma-merkki ei saa esiintyä datajoukossa. Usein käytetään yhtä tai useampaa K-merkin joukkoa comma-merkkinä. Nämä merkit ovat taulukon 4.2 K28.1, K28.5 ja K28.7, sillä ne kaikki sisältävät bittikuvion 1100000 ensimmäisenä seitsemänä bittinä. Samaa bittikuviota ei löydy muista merkeistä, joten se on uniikki bittisarja merkkijärjestelmässä. Edellä mainittujen K-merkkien käyttö on turvallinen vaihtoehto, kun ollaan rakentamassa kustomoitua protokollaa. Esimerkiksi Gigabit Ethernet käyttää K28.5-merkkiä comma-merkkinä. Kuva 4.3 esittää 8b10b enkoodauksen lohkokaaviona. [32]

8 sisääntulevaa bittiä jaetaan 5:n ja 3:n bitin väyliin. Molempien väylien data enkoodataan omiksi alalohkoikseen. Yksi 8b10b järjestelmän huonoista puolista on ylimääräinen laskenta-aika (overhead). Jotta saavutettaisiin 2.5 Gigabitin kaistanleveys, tarvitaan 3.125 Gbps nopeus johtimessa. [32]

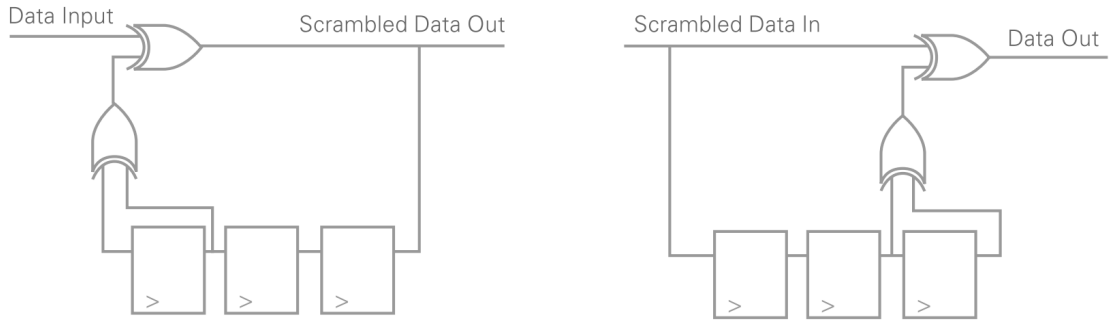
4.2.2 64b66b

Uudempaa teknologiaa edustava 64b66b-enkoodausjärjestelmä tarjoaa ratkaisun uudempien ja nopeampien väylien vaatimuksiin. Esimerkiksi



Kuva 4.3. *Enkooderi/Dekooderi-lohkokaavio [32]*

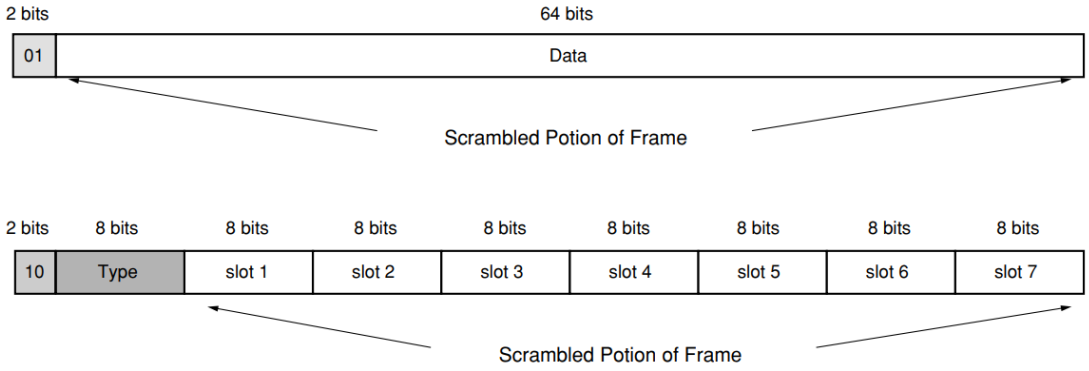
telekommunikaatiostandardi Synchronous Optical Network (SONET) kykeni yli 10 Gb siirtoon, jolloin tarvittiin tehokkaampaa 64b66b-linjakoodausta. 64b66b-linjakoodausjärjestelmä käyttää hakutaulukoiden sijaan salausmenetelmää (scrambling) yhdistettynä ei-salattuun synkronisointimalliin ja valvontatyyppiin. Salausmenetelmän tarkoituksena on rikkoa pitkät ykkösten ja nollien jonot sekä muut haitalliset bittikuviot, jotka vaikeuttavat vastaanottimen dekoodauskykyä [32]. Kuva 4.4 esittää salauspiirin rakenteen. Salauspiiri vaatiin logiikkapiirin, joka tarjoaa korkean todennäköisyyden tasaisesti jakautuville nollille ja ykkösille. Piiri sallii sen, että koodattu data näyttää ulospäin satunnaiselta tiheillä siirtymillä, mutta samalla sen voi kääntää helposti vastaanottimen päässä [33]. Kuvassa 4.4 salauslohko koostuu sarjasta kiikkuja, jotka siirtävät datavirtaa. Useimmiten kiikut syöttävät seuraavan bitin, mutta toisinaan kiikun ulostulo syötetään OR- tai AND-portin läpi vanhemman bitin kanssa. [32]



Kuva 4.4. *Salauspiiri [32]*

64b66b-järjestelmälle on kaksi kehystyyppiä. Yksinkertaisempi näistä koostuu 2 bitin synkronointikuviosta "01", jonka jälkeen tulee 64 databittiä. Data sekoitetaan

synkronointibittejä lukuun ottamatta. Toinen kehystyyppi sallii myös ohjausinformaation datan oheen. Ohjauskehys alkaa 2-bittisellä kuviolla "10". Kahdeksan seuraavaa bittiä tyyppikentässä määrittää 56 bitin muodon. Esimerkiksi jos tyyppikentässä on arvo 0xCC, loppudata sisältää neljä tavua dataa ja kolme tavua valvontatietoa. 4.5 esittää 64b66b-linjakoodauksen bittien järjestyksen. [32]



Kuva 4.5. *64b66b-kaavio [32]*

64b66b-järjestelmä hoitaa datan kohdistuksen synkronointibittien 01 ja 10 avulla, jotka esiintyvät datassa vähintään joka 66 bitin välein. Prosessi valitsee satunnaisen aloituskohdan ja tarkistaa, esiintyykö siinä validi synkronointibittien kombinaatio. Kun oikea kombinaatio löytyy, positio 66 bittiä myöhemmin tarkistetaan. Kun riittävät määrä synkronointibittejä löytyy, kohdistusprosessi katsotaan toteutuneeksi. Välistä unohtuneet synkronoinnit nostavat virhelaskuria, ja tietyn ajan kuluessa riittävä määrä virheitä aloittaa synkronointiprosessin uudelleen. [32]

64b66b-enkoodausjärjestelmällä saadaan pienennettyä järjestelmän yleisrasitteita (overhead), mutta metodin käyttö tuo vastaavasti pidempiä kohdistusaikoja ja mahdollisuuden lievään DC-epätasapainoon. Myös enkooderi ja dekooderi ovat monimutkaisempia. Esimerkiksi salauslohkojen käynnistäminen ja sammuttaminen hyötykuorman ja synkronointidatan välillä tekee 64b66b-piiristä monimutkaisen. [32] Esimerkiksi Xilinx tarjoaa myös 64b66b-linjakoodauksella toteutettua Aurora-protokollaa 8b10b-linjakoodaustoteutuksen lisäksi [34].

4.2.3 128b130b

PCI Express -versiot 1.0 ja 2.0 käyttävät molemmat 8b10b-koodausta, mutta modernimmat versiot 3.0 ja 4.0 pystyvät hyödyntämään 128b130b-koodausjärjestelmää saavuttaakseen suuremman läpimenotiheyden paremmin hallittavalla juovataajuudella (line rate). Juovataajuus kertoo nopeuden, jolla bitit lähetetään johtimeen. 128b130b:n voidaan ajatella olevan kuten 64b66b-järjestelmä kaksinkertaisella hyötykuormalla. Käytössä on edelleen vain kaksi alustusbittiä. Taulukko 4.3 näyttää PCI Express -versiot ja niiden käyttämät koodausmenetelmät. [33]

Taulukko 4.3. *PCI Express -versioiden koodausmenetelmät*

PCI Express Version	Encoding	Line Rate	Throughput (x16)
1.0	8b/10b	2.5 Gbps	4 GB/s
2.0	8b/10b	5 Gbps	8 GB/s
3.0	128b/130b	8 Gbps	15.8 GB/s
4.0	128b/130b	16 Gbps	31.5 GB/s

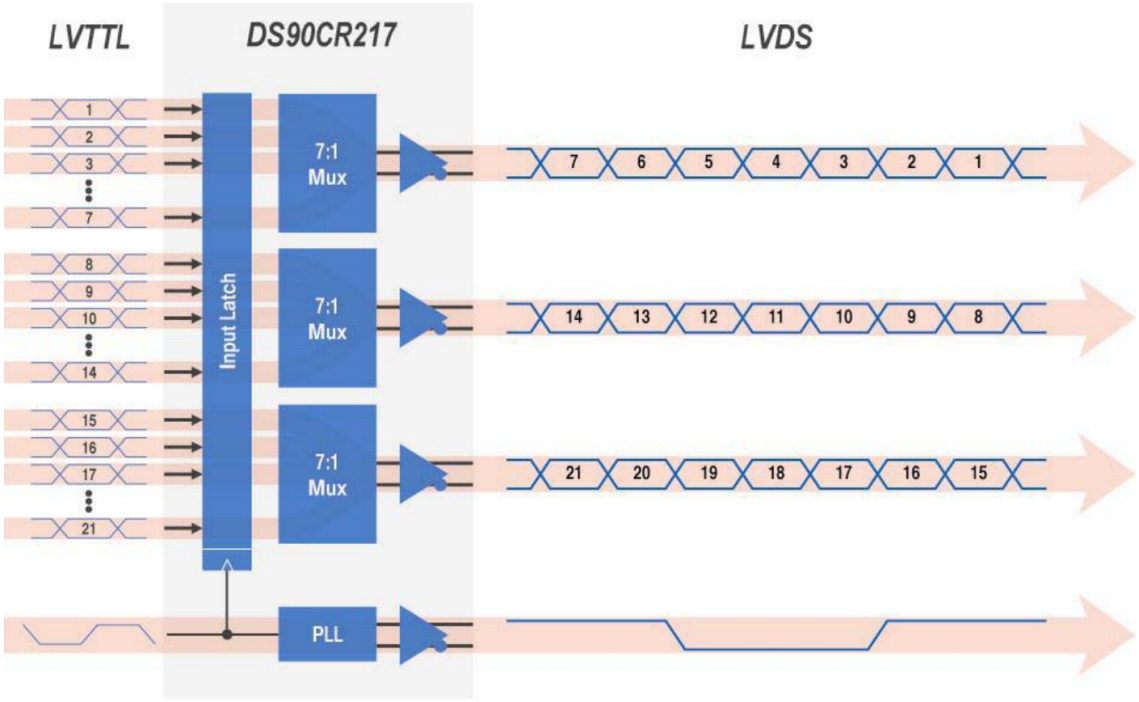
PCI Express siirtyi kolmannen sukupolven kohdalla 128b130b-koodaukseen nostaakseen datan läpimenotiheyttä pitäen samalla juovataajuuden hallittavissa olevassa nopeudessa. [33]

4.3 SerDes

SerDes, eli Serializer/Deserializer on kahden pisteen välinen suoran yhteyden muodostava sarjamuotoinen kommunikaatioteknologia [35]. Konseptin tarkoituksena on muuntaa rinnakkaismuotoinen hidas signaali sarjamuotoiseksi ja siirtää se optisen kaapelin tai kuparijohtimen läpi vastaanottimeen, jonka jälkeen signaali muunnetaan takaisin rinnakkaismuotoiseksi [35]. Näin saadaan hyödynnettyä tiedonsiirtokanavan kapasiteetti täysin, vähentäen siirtoon vaadittavien kanavien määrää ja kommunikaatiomenetelmän kustannuksia [35]. [36] jakaa SerDes-laitteet muutamaan arkkitehtuuriin, jotka on räätälöity tiettyjä vaatimuksia varten. Arkkitehtuurien ymmärtäminen auttaa suunnittelijaa tekemään oikean valinnan sopivan SerDes arkkitehtuurin suhteen. Tässä kappaleessa esitetyt SerDes laitteistot ovat normaalisti sisäänrakennettuna System on Chip -piirikorteille ja FPGA-piireille. [36]

Parallel clock SerDes -arkkitehtuuria käytetään sarjamuuntamaan leveitä rinnakkaisia data address control -väyliä, kuten PCI:tä, prosessoriväyliä ja ohjausväyliä. Yhden multiplexerin sijasta kyseisessä arkkitehtuurissa käytetään useita N to 1 -multipleksereitä, joista jokainen sarjamuuntaa oman osuutensa väylästä. 4.6 havainnollistaa 21 bittiä leveän väylän sarjamuuntamisen kolmen 7 to 1 multiplekserin avulla. [36]

Jaetut datavirrat etenevät rinnakkain vastaanottimeen oman kellosignaalin kanssa, jota vastaanotin käyttää ottaakseen datan sisään. Kellodataparien poikkeamat tulee minimoida rinnakkaistamisen onnistumiseksi vastaanottimessa. Vaikka parallel clock SerDes käyttääkin useita eromuotoisia signaalipareja, kyseinen arkkitehtuuri käyttää silti vähemmän johtimia rinnakkaissiirtoon verrattuna. Muita hyötyjä ovat muun muassa alhaisempi teho, parempi virranajokyky pitkässä kaapelissa, pienempi kohina/EMI sekä alhaisemmat kaapeli- ja liitinkulut. Parallel clock serializer tuo myös joustavuutta, sillä eromuotoisten signaaliparien määrä voi olla vapaasti määrättävä. Näin voidaan välttyä myös erittäin suurien datanopeuksien tuottamilta suunnitteluongelmilta. Kyseinen SerDes tarjoaa myös erinomaisen hinta-suorituskykysuhteen ja se on usein ainoa käytännöllinen keino perinteisen leveän rinnakkaisväylän siirtoon useiden metrien



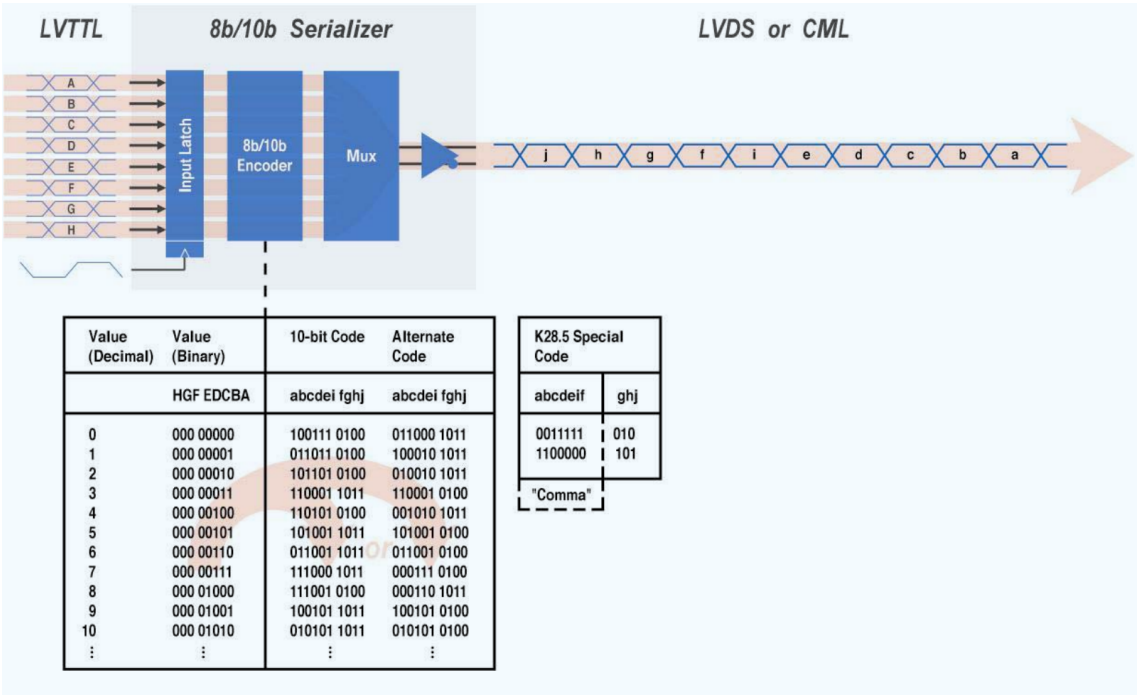
Kuva 4.6. *Parallel clock serializer [36]*

pituisen kaapelin yli. Tavanomaisia sovelluksia, jotka hyödyntävät parallel clock serdes arkkitehtuuria, ovat esimerkiksi pinottavat Ethernet-kytkimien laajennukset, rack-to-rack-telekommunikaatioliittimet ja video- tai kameralinkit. [36]

8b/10b:n rinnakkais-sarjamuunnin koodaa jokaisen rinnakkaisen tavun 10-bittiseksi koodiksi ja sarjoittaa sen eromuotoiseen johtimeen. Tämän arkkitehtuurin rakenne on esitelty kuvassa 4.7. [36]

Löytääkseen 10-bittiseksi koodatun sanan rajat sarjamuotoisesta virrasta lähettäjä merkitsee nämä rajat lähettämällä *commamerkiksi* kutsutun symbolin. Uniikki bittisarja commamerkissä ei esiinny koskaan normaalissa dataliikenteessä, mikä tarjoaa siten luotettavan tuntomerkin vastaanottimelle koodin kohdistamiselle. 8b/10b-SerDes sopii sarjamuuntamaan tavuorientoitunutta dataa, kuten pakettiliikennettä kokoajapiirikortin, kaapelin tai valokuidun yli. Esimerkiksi Ethernet, Fibre Channel ja InfiniBand -standardit käyttävät 8b/10b-koodausta 1,0625; 1,25; 2,5 ja 3,125 Gbps:n bittinopeuksila, joilla monet SerDes:t toimivat. 8b/10b-datavirta on DC-balansoitu. Lähetettävien bittien juokseva poikkeama eli ykkösten ja nollien lukumäärä on keskiarvoltaan nolla. Tämä mahdollistaa luotettavan ohjauksen AC-kytketyissä ympäristöissä ja valokuitumoduuleissa. DC-balansointi 8b/10b-enkoodauksessa auttaa myös nostamaan johtimien virranajokykyä. [36]

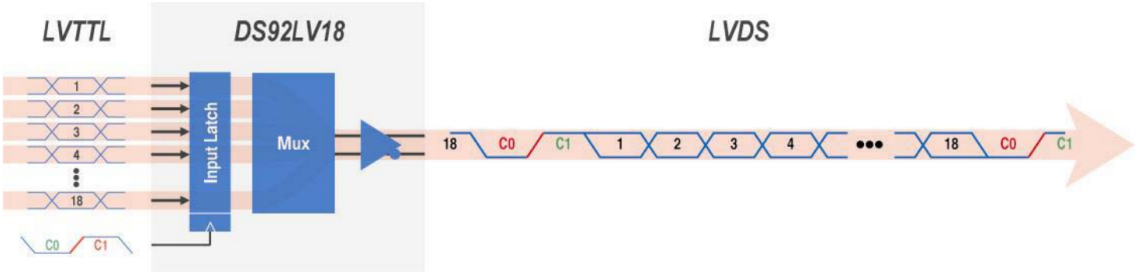
8b/10b-enkoodaus mahdollistaa myös keinon virheentarkasteluun sekä säätöinformaation lähettämiseen. Useimmat enkoodauksen 10-bittisen koodin permutaatiot eivät ole valideja 8b/10b-enkoodauksen sanoja. Tästä syystä sarja-rinnakkaismuunnin pystyy liputtamaan kelpaamattomat koodaukset. Tällainen virhetarkastelu antaa myös mahdollisuuden sarjalinkin suorituskyvyn monitorointiin.



Kuva 4.7. *8b/10b serializer [36]*

Monet standardit määrittelevät ohjaussanoja, kuten kohdistuslippuja, jotka auttavat järjestelmää rakentamaan ja purkamaan paketteja. Tämä on tehnyt 8b/10b-enkoodauksesta suositun rinnakkais-sarjamuuntimen arkkitehtuurin tietoliikennetietoja käsittelevissä järjestelmissä. [36]

Embedded clock bits -arkkitehtuurilla toteutettu lähetin sarjamuuntaa dataväylän ja kellon yhteen sarjamuotoiseen signaalipariin. Jokaisella syklillä on kaksi kellobittiä, nolla ja yksi, joilla jokaisen sarjamuunnetun sanan alku ja loppu paketoidaan kehykseksi. Tästä syystä arkkitehtuurilla on myös nimi *start-stop bit* SerDes. Datan hyötykuorman paketoiminen aloitus- ja lopetusbitin ympärille tuottaa datavirtaan jaksollisen nousevan reunan. Arkkitehtuurin toimintaa kuvaa 4.8. [36]



Kuva 4.8. *18-bit Embedded clock bits serializer [36]*

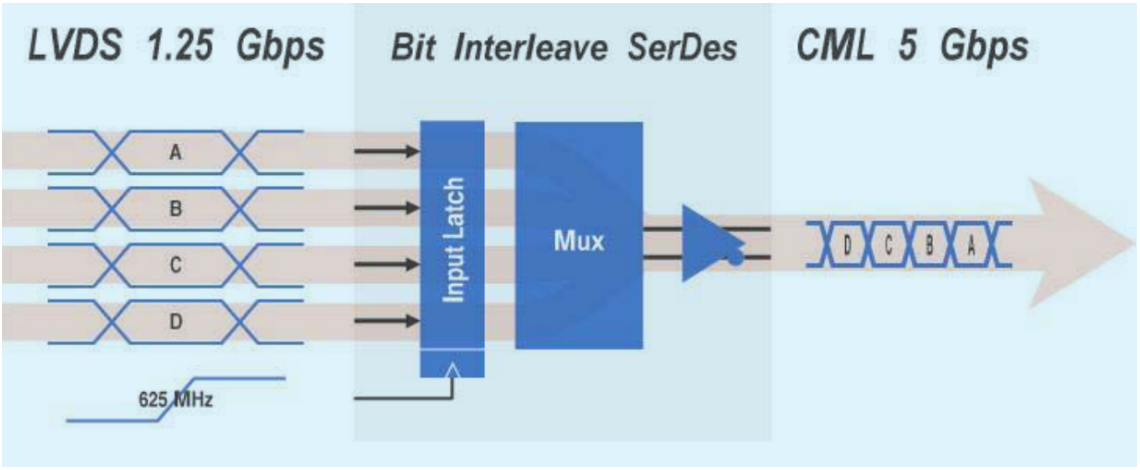
Vastaanotin etsii sulautetun kellon säännöllisen nousevan reunan ja synkronoituu siihen. Embedded clock bits SerDesiä voidaan käyttää sovelluksiin, jotka kuljettavat raakadatan ohessa muita signaaleita, kuten ohjausbittiä, pariteettibittiä, kehysbittejä, synkronisointibittejä, statusta ynnä muuta. 18 bitin lähetin sarjamuuntaa lähetettävän datan lisäksi kaksi ylimääräistä bittiä, kuten pariteettibitin ja kehyksen bitin. Tällä

lähettimellä vältytään ylimääräiseltä logiikalta ja datan puskuroinnilta, sillä kaikki lisätyt bitit sarjamuunnetaan datan ohessa normaalilla A/D-muuntimen näytteenottotaajuudella. [36]

Toinen embedded clock serdes -ominaisuus on vastaanottimen automaattinen lukittautuminen satunnaiseen dataan, mikä on hyödyllinen ominaisuus järjestelmissä, joissa vastaanotin on erillinen moduuli ilman suoraa ohjausta. On myös tavanomaista, että järjestelmässä yksi lähetin lähettää usealle vastaanottimelle. Tällaisessa tilanteessa järjestelmän väylään liitetty uusi vastaanotin lukittautuu satunnaiseen dataan ilman dataliikenteen keskeytymistä. Vastaanottimen sarja-rinnakkaismuunnin jäljittää saapuvasta liikenteestä sulautetun kellon nousevan reunan ja lukittautuu siihen. [36]

Embedded clock bits SerDes sopii hyvin esimerkiksi prosessoimattoman raakadatan ja ohjaussignaalin siirtoon keskittyviin sovelluksiin. Tällaisia sovelluksia voivat olla esimerkiksi signaalinkäsittelyjärjestelmät, kuten tukiasemat tai kuvantamis- ja sensorijärjestelmät, joissa A/D-muunnin, kamera tai näyttö käsittelee raakadataa linkin toisessa päässä olevan prosessointiyksikön avulla. [36]

Bit interleaving SerDes multipleksaa useita hitaampia sarjamuotoisia datavirtoja yhteen nopeampaan virtaan lomittamalla bitit. Vastaanotin demultipleksaa bitit takaisin alkuperäiseen hitaampaan virtaan. Arkkitehtuuria havainnollistaa kuva 4.9. [36]



Kuva 4.9. *Bit Interleaving serializer [36]*

Bit interleaving SerDes:ia käytetään tavanomaisesti telekommunikaatiolähettimien laitteistossa esimerkiksi SONET/SDH-datavirtojen yhteenkokoamisessa siirtoa varten. Synchronous optical networking (SONET) ja synchronous digital hierarchy (SDH) ovat protokollia, joilla voidaan siirtää useita bittivirtoja synkronisesti valokuidun ylitse.

Tässä luvussa esitellyillä SerDes-arkkitehtuureilla on useita hyvin erilaisia etuja ja haittoja, ja ne soveltuvat moniin eri käyttötarkoituksiin. Haittoja voivat olla esimerkiksi tiukat ajoitusvaatimukset, korkea hinta sekä haasteet suunnittelussa. Tietyt arkkitehtuurit muun muassa mahdollistavat esimerkiksi väljemmät ajoitusvaatimukset tai ovat esimerkiksi synkronoitavissa edullisesti. Oikean arkkitehtuurityypin hyödyntäminen käyttökohteesta riippuen jää järjestelmän suunnittelijan vastuulle. [36]

4.4 PCI Express

PCI Express on kolmannen sukupolven korkean suorituskyvyn tiedonsiirtoväylä, jotka käytetään oheislaitteiden yhdistämiseen laskenta- ja kommunikaatioalustoilla. PCIe:n arkkitehdit ovat säilyttäneet edellisistä PCIe-sukupolvista hyödyllisimmät piirteet ja hyödyntäneet tietokonearkkitehtuurin uudemmat kehitysaskeleet. PCIe hyödyntää esimerkiksi samoja käyttömalleja ja load-store-arkkitehtuuria kuin PCI ja PCI-X. PCIe on siten ohjelmistollisesti taaksepäin yhteensopiva PCI ja PCI-X -järjestelmien kanssa. [37]

PCI Express on point-to-point-tyyppinen sarjamuotoinen liitäntä kahden laitteen välille. Useat PCIe-laitteet puolestaan liittyvät toisiinsa kytkimien avulla. Sarjamuotoinen liitäntä laitteiden välillä vähentää myös pinnimäärää, mikä puolestaan vähentää PCIe-piirikorttien kustannuksia ja järjestelmän monimutkaisuutta. Sarjaväylänä PCIe:llä on paljon yhteistä InfiniBandin tai Fibre Channelin kaltaisten aiempien sarjamuotoisten väylien kanssa, mutta samalla se pysyy ohjelmallisesti taaksepäin yhteensopivana PCI:n kanssa. [37]

4.4.1 PCIe-arkkitehtuuri

Kuten monet suurinopeuksiset sarjaväylät, myös PCIe käyttää kaksisuuntaista dual-simplex-liitäntää, joka kykenee lähettämään ja vastaanottamaan dataa samaan aikaan. Polkua kahden PCIe-laitteen välillä kutsutaan linkiksi, ja se koostuu yhdestä tai useammasta lähettävästä ja vastaanottavasta parista. Yhtä paria kutsutaan kaistaksi. PCIe:n speksi sallii linkinmuodostuksen 1, 2, 4, 8, 12, 16, tai 32 kaistan avulla. Kaistojen lukumäärää kutsutaan linkin leveydeksi ja se esitetään seuraavalla tavalla: x1, x2, x4, x8, x16 ja x32. Useampi kaista kasvattaa linkin kaistanleveyttä, mutta kasvattaa samalla sen hintaa, tehonkulutusta ja vie enemmän tilaa. [37]

Taulukko 6.2 kertoo bittinopeudesta ja väylän ominaisuuksista johdetut, linkin leveydestä riippuvat kaistanleveydet kolmen ensimmäisen PCIe-sukupolven välillä. Yksi väylän ominaisuuksista on kappaleessa 4.2 kuvattu linjakoodausprosessi. Kaksi ensimmäistä PCIe-sukupolvea käyttäät 8b/10b-koodausta, joka luo 10-bittisen ulostulon 8-bittisestä sisäänmenosta. Eli yhden tavun siirtäminen vaatii 10 bitin lähettämisen. PCIe Gen1 -bittinopeus on 2.5 GT/s (gigatransfers per second). Kun tämä bittinopeus jaetaan kymmenellä saadaan selville että yksi kaista pystyy lähettämään 0.25 GB/s:n nopeudella. Koska kyseessä on double simplex -väylä, lähettäminen ja vastaanottaminen on mahdollista samaan aikaan. Näin 0.5 GB/s:n kaistanleveys per kaista on mahdollista. Eli

$$Gen1PCIeBandwidth = (2.5Gb/s * 2suuntaa)/10bittipersymboli = 0.5GB/s \quad (4.1)$$

PCIe Gen2 puolestaan kaksinkertaistaa taajuuden, jolloin myös kaistanleveys kaksinkertaistuu. PCIe Gen3 vaihtoi linjakoodauksen 128b/130b:hen, mutta nosti nopeuden vain 8GT/s:iin silti kaksinkertaistaen jälleen kaistanleveyden

Taulukko 4.4. *PCIe-sukupolvien kaistanleveys [37]*

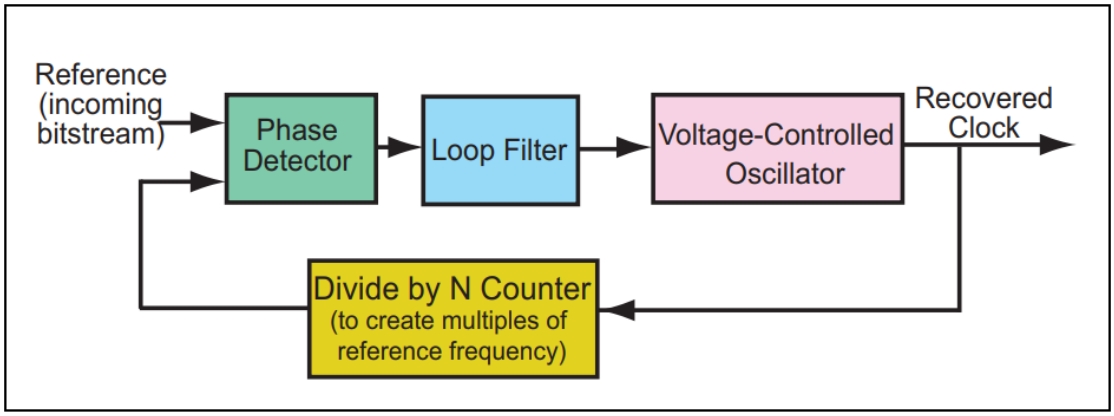
Link Width	GT/s	x1	x2	x4	x8	x12	x16	x32
Gen1 Bandwidth (GB/s)	2.5	0.5	1	2	4	6	8	16
Gen2 Bandwidth (GB/s)	5	1	2	4	8	12	16	32
Gen3 Bandwidth (GB/s)	8	2	4	8	16	24	32	64

128b/130b-linjakoodauksen luonteen ansiosta. 128b130b sisällyttää 2 bittiä yleisrasitteeksi jokaista 128 bittiä kohden, mutta yleisrasite ei ole tarpeeksi merkittävä, jotta se otettaisiin huomioon esitetyissä taulukoissa. [37] laskee Gen3 PCIe:n kaistanleveyden seuraavalla tavalla:

$$Kaistanleveys = (8.0Gb/s * 2suuntaa)/8bittipertavu = 2.0GB/s. \tag{4.2}$$

[37]

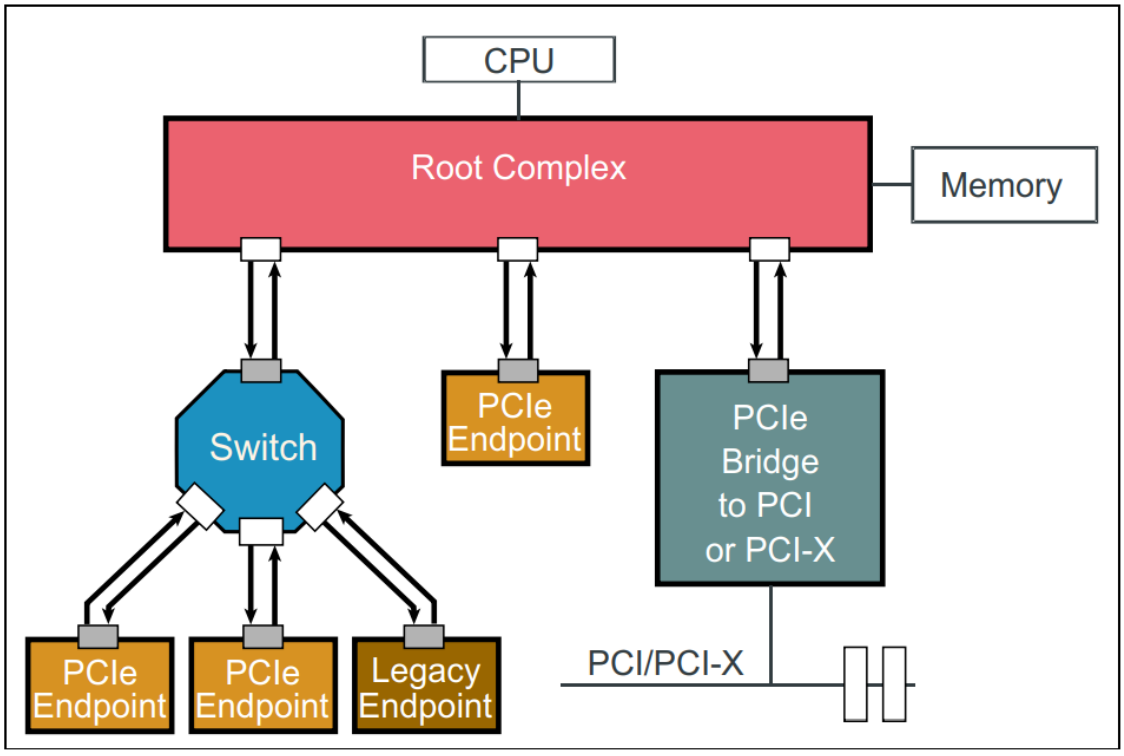
PCIe käyttää kappaleessa 4.3 esiteltyä kellon sulauttamista lähetettävään dataan 8b10b-koodauksen avulla. Vastaanotin palauttaa kellopulssin datavirrasta ja käyttää sitä saapuvan datan salpaamiseen. Kuvan 4.10 PLL-piirin lohkokaavio ottaa sisään bittivirran referenssiksi ja vertaa sen ajoitusta tai vaihetta määrätyllä kellotaajuudella luotuun kellosignaaliin. Vertailun tuloksen perusteella ulostulon kellosignaalin taajuutta joko nopeutetaan tai hidastetaan, kunnes vertaillut signaalit vastaavat toisiaan. Tässä vaiheessa PLL piirin sanotaan olevan lukittu. Piiri tarkistaa kellosignaalien vastaavuuden toistuvasti, jolloin saadaan kompensoitua esimerkiksi lämpötilan tai jännitteen vaikutusta kellosignaaliin. PLL tarvitsee sisäänmenoon siirtymätiloja voidakseen tehdä vaiheen vertailun. Jos väylässä ei kulje dataa hetkeen, PLL alkaa ajautua pois oikeasta taajuudesta. 8b10b-koodaus estää ongelman sallimalla vain viisi peräkkäistä nollaa tai ykköstä bittivirrassa. [37]



Kuva 4.10. *PLL-piirin lohkokaavio [37]*

4.4.2 PCIe-topologia

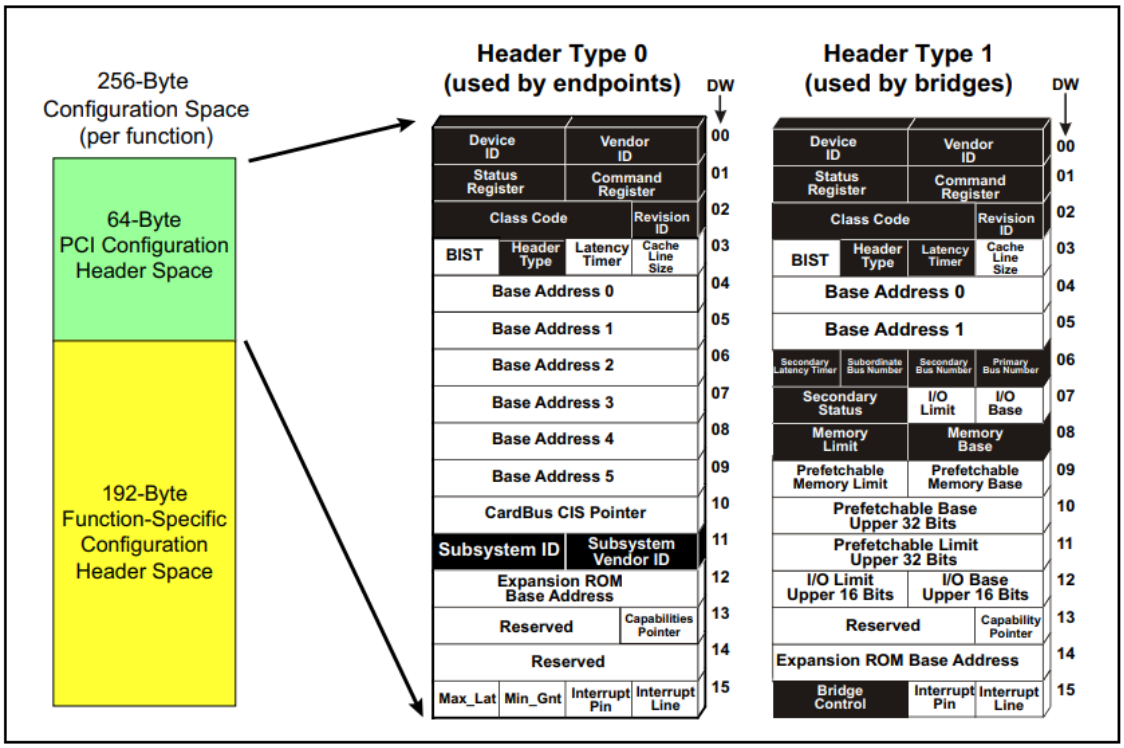
PCIe käyttää kytkimiä ja siltoja joustavan järjestelmän rakentamiseen, sillä yksittäiset linkit voivat yhdistää vain kaksi käyttöliittymää. Kuva 4.11 esittelee yksinkertaisen PCIe-topologian. Taaksepäin yhteensopivuus PCI-standardin kanssa halutaan säilyttää, jolloin monimutkaiset rakenteet jäävät pois ja jäljellä on yksinkertainen puurakenne. Root Complex yhdistää CPU:n ja itse PCIe-väylät toisiinsa, ja lohko voi sisältää useitakin komponentteja, kuten esimerkiksi prosessorin ja DRAM lohkon käyttöliittymän. Kytkimet sallivat useamman laitteen liittymisen yhteen PCIe-porttiin, ja ne toimivat kuten pakettien reititin, joka ohjaa paketin osoitteen perusteella oikeaan polkuun. Sillat puolestaan tarjoavat liityntäkohdan muille väylille, kuten PCI:lle ja PCI-X:lle. Endpointit ovat PCIe-topologiassa laitteita, jotka eivät ole kytkimiä tai siltoja, ja toimivat toimeksiantajina ja transaktioiden täydentäjinä väylässä. Endpointilla on vain yksi paluukanavan portti Root Complexin suuntaan. Laitteet, jotka on suunniteltu käyttämään vanhempaa väylää, kuten PCI-X:ää, mutta ovat nyt liitettynä PCIe-käyttöliittymään, ovat legacy endpointteja. Natiivit PCIe-endpointit ovat memory mapped -laitteita (MMIO devices). [37]



Kuva 4.11. *Esimerkki PCIe-topologiasta [37]*

Ohjelmallinen yhteensopivuus pidetään yllä pitämällä konfiguraation header-osa muuttumattomana PCI:n kanssa. Kuva 4.12 esittää tyypin 0 ja 1 headerit PCIe endpointille ja PCIe sillalle. PCIe root complex näyttäytyy ohjelmallisesti kokoelmana sisäisiä väyliä ja siltoja, jotka levittäytyvät moniin portteihin. Sisäinen väylä näkyvät konfiguraatio-ohjelmalle PCI-väylänä, jonka järjestysnumero on nolla. PCIe-portit näkyvät PCI-to-PCI-siltana. Kytkin puolestaan näyttäytyy ohjelmalle kokoelmana siltoja,

joita yhdistää yhteinen väylä. [37]



Kuva 4.12. *PCIe Configuration Headers [37]*

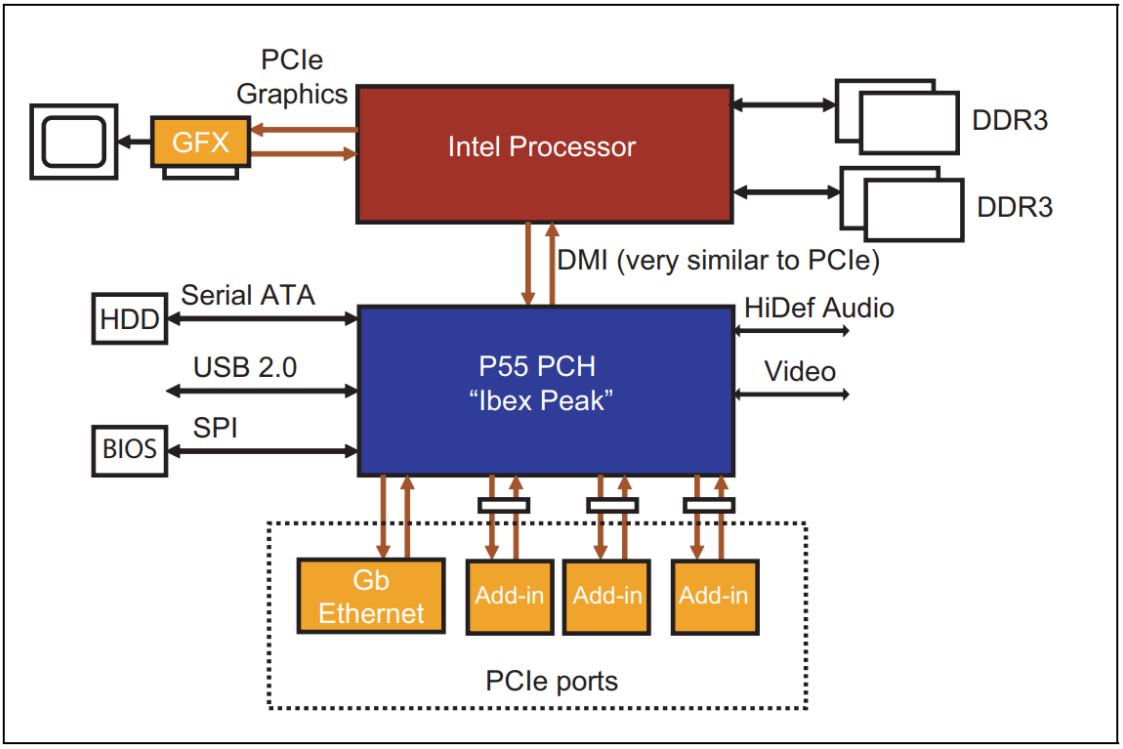
[37] esittelee PCIe:hen perustuvan edullisen järjestelmän, joka on esitetty kuvassa 4.13. Järjestelmäesimerkki ei eroa paljoakaan perinteisestä PCI-järjestelmästä, mutta siitä löytyy muutama PCIe-portti ja korttilohkoja. Kuvan 4.13 Intel Processor -lohko on järjestelmän Root Complex ja sisältää luonnollisesti useita komponentteja. Lohkoon sisältyy esimerkiksi x16 PCIe-portti grafiikkakorttia varten sekä kaksi DRAM-kanavaa, mikä tarkoittaa, että lohkosta tulee löytyä muistiajuri sekä reitityslogiikkaa integroituna CPU:hun. [37]

4.4.3 Laitekerrokset

PCIe on määritelty kuvan 4.14 mukaisesti laitekerroksiin. Kerrokset voidaan jakaa kahteen itsenäisesti toimivaan osaan, sillä jokaisella kerroksella on lähettävä puoli ulosmenevälle liikenteelle ja vastaanottava puoli saapuvalle liikenteelle. [37]

Device Core

Device Core toteuttaa laitteen päätoiminnallisuuden. Jos laite on endpoint-tyyppinen, Core voi koostua enimmillään kahdeksasta toiminnallisuudesta, joista jokainen toteuttaa oman konfiguraatioavaruutensa. Jos laite on kytkintyyppinen, Core koostuu paketinvälityslogiikasta ja sisäisestä väylästä. Jos laite on Root Complex -tyyppiä, se toteuttaa virtuaalisen PCI-väylä nollan ja siellä sijaitsevat kaikki piirisarjaan upotetut Endpointit ja virtuaaliset sillat. Device Coren voidaan ajatella olevan kaikkien



Kuva 4.13. *Esimerkki PCIe-järjestelmästä [37]*

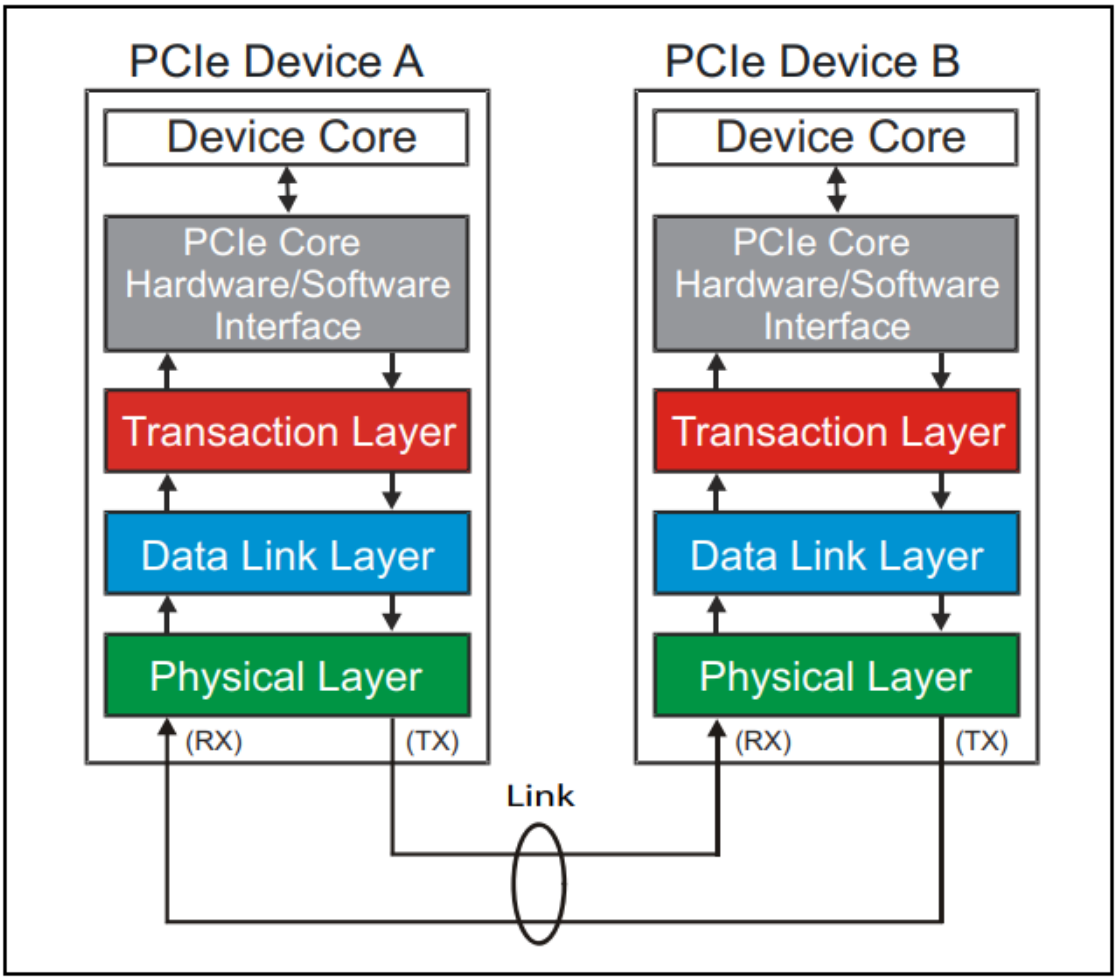
tiedonsiirtopyyntöjen lähde tai määränpää. [37]

Link Training

Yksi olennaisista Physical Layer -laitekerroksen velvollisuuksista on lähettimen ja vastaanottimen yhteyden eli linkin käynnistäminen ja virittäminen. PCIe:n termistö käyttää tästä virittäytymisestä termiä *training*. PCIe:n tapauksessa kyseinen prosessi on automatisoitu ja se käsittää useiden tilatietojen selvittämisen. Esimerkiksi linkin leveys voi vaihdella yhdestä kaistasta 32 kaistaan. Myös useita nopeuksia voi olla vaihtoehtona. Viritysprosessi selvittää nämä asetukset saavuttaakseen sopivan yhdistelmän. PCIe Link Training tarkistaa ja etsii seuraavat asetukset:

- Linkin leveys - width
- Linkin siirtonopeus - data rate
- Kaistan kytkentä päinvastoin - Lane reversal
- Vastakkainen polariteetti - Polarity inversion
- Kellosignaalin palautus - Bit lock per Lane
- Etsitään bitstreamista tunnistettava datapiste - Symbol lock per Lane
- Kaistojenvälisen poikkeaman poisto - Lane-to-Lane de-skew within a Multi-Lane Link

[37]



Kuva 4.14. *PCIe Laitekerrokset [37]*

Transaction Layer

Transaction Layer vastaa TLP-pakettien (Transaction Layer Packet) luomisesta lähetyspuolella ja dekoodauksesta vastaanottavassa päässä. Tällä kerroksella on myös muita ominaisuuksia, kuten Quality of Service -toiminnosta, datavuon ohjauksesta ja tapahtumien järjestämisestä. Transacion Layer tukee jaettua protokollaa non-posted tyyppisille tiedonsiirtotapahtumille ja liittää sisääntulevan täydentävän tiedonsiirron ulosmenevään non-posted pyyntöön. TLP:t voidaan luokitella neljään pyyntöjen kategoriaan: Memory, IO, Configuration ja Messages. Tiedonsiirto määritellään Request- ja Completion-paketin yhdistelmäksi. Request paketti kuljettaa kohdelaitteelle käskyn ja Completion paketti lähettää takaisin vastauksen. PCIe Request -tyypit on listattu taulukossa 4.5. [37]

Completerin tulee vastata Requesterin lähettämiin Non-Posted-pyyntöihin Completion-paketilla. Kuten taulukosta 4.5 voidaan nähdä, Memory Write ja Message -tyyppiset pyynnöt ovat Posted-pyyntöjä, jolloin kohdelaite ei vastaanota Completion TLP:tä Requesterille. Posted-siirrot parantavat väylän suorituskykyä sillä Requesterin ei tarvitse odottaa vastausta. Toisaalta Requester ei saa palautetta siitä, onko kirjoitus

Taulukko 4.5. *PCI Express Request tyypit [37]*

Request Type	Non-Posted or Posted
Memory Read	Non-Posted
Memory Write	Posted
Memory Read Lock	Non-Posted
IO Read	Non-Posted
IO Write	Non-Posted
Configuration Read (Type 0 and Type 1)	Non-Posted
Configuration Write (Type 0 and Type 1)	Non-Posted
Messages	Posted

mennyt virheettä läpi. Tämä PCI:stä periytynyt käytäntö on silti erittäin hyvä ominaisuus, sillä virheiden todennäköisyys on erittäin pieni ja saavutettu suorituskyvyn lisäys on merkittävä. [37]

Data Link Layer luo Data Link Layer -paketteja (DLLP) lähettävällä puolella. Kerros myös hoitaa DLLP:n dekoodauksen vastaanottavalla puolella. Data Link Layer vastaa myös linkin virhetarkastelusta ja virheen korjauksesta. [37]

Physical Layer vastaa Ordered-Set-pakettien luomisesta ja dekoodauksesta. Kerros vastaa myös kaikkien pakettien (TLP, DLLP ja Ordered-Set) lähettämisestä linkkiin ja kaikkien vastaanotettujen pakettien käsittelystä. Paketit prosessoidaan lähettimen puolella järjestelemällä ne logiikan, salauslaitteiston, enkooderien ja pakettien sarjamuuntimien mukaan. Paketti lähetetään lopulta ulos jokaista kaistaa hyväksi käyttäen. Vastaanottava fyysinen kerros käsittelee eromuotoisesti johtimeen koodatun datan ja muuttaa sen jälleen digitaaliseen muotoon, minkä jälkeen saapuva bittivirta sarja-rinnakkaismuunnetaan. Sarja-rinnakkaismuunnos toteutetaan Clock and Data Recovery -piirin (CDR) palauttaman kellosignaalin avulla. Vastaanotetut paketit viedään 8b10b/128b130b-dekoodereiden, salauksenpurkajien ja tavujen lomituksenpurkajien läpi. Lopulta fyysisen kerroksen Link Training ja Status State Machine (LTSSM) toteuttavat linkin käynnistyksen ja ylläpidon. [37]

4.4.4 Osoiteavaruus

Jotta laitteen sisäisiin rekistereihin voidaan tehdä luku- ja kirjoitusoperaatioita, täytyy muistien sijainteihin voida määrätä osoitteet järjestelmän tukemista osoiteavaruuksista. PCI Express tukee PCI-väylän tavoin kolmea osoiteavaruustyyppiä taaksepäin yhteensopivuuden vuoksi. Nämä ovat Configuration-, Memory- sekä IO-osoiteavaruus. Näistä IO- sekä Configuration-osoiteavaruus ovat käytössä PCI-yhteensopivuuden takia. PCIe:n Configuration-osoiteavaruuteen kuvataan usein laitekohtaisia rekistereitä, jotka liittyvät ohjaukseen, tilatietoihin tai osoitinrekistereihin. IO-osoiteavaruus puolestaan ei ole moderneissa järjestelmissä enää käytössä, jolloin modernimmat järjestelmät alkoivat kuvaamaan IO-laitteiden sisäisiä rekistereitä Memory-osoiteavaruuteen. Yleisesti tähän

osoiteavaruuteen viitataan termillä Memory-Mapped IO, eli MMIO. PCIe:n spesifikaatio kehottaa [37] mukaan välttämään IO-osoiteavaruuden käyttöä, sillä sen tarkoitus on ainoastaan legacy-käytössä. PCIe tukee maksimissaan 64 bittiä pitkiä osoitteita. Edellä mainittua MMIO-osoiteavaruustyyppiä voi olla PCIe-laitteiden käytössä kahta erilaista. Ensimmäinen näistä on Prefetchable MMIO, joka voidaan määritellä osoitealueen datan olevan etukäteen noudettavissa välimuistiin esimerkiksi ennakoitavissa olevaa lukuoperaatiota varten. Toinen tyyppi on Non-Prefetchable MMIO, jossa osoitealueen data ei ole etukäteen noudettavissa välimuistiin. [37]

Järjestelmään liittyvien PCI- ja PCIe-laitteiden tulee tarjota järjestelmälle keino, jolla määrittää laitteen tarve muistiosoitteille. Mikäli laitteen vaatimukset osoiteavaruuden suhteen ovat tiedossa, laitteelle varataan oikeaa muistityyppiä sisältävä alue käytettävissä olevasta osoiteavaruudesta. Tarve osoiteavaruuden käytölle ilmaistaan Configuration-osoiteavaruuden headerissa Base Address Rekistereiden (BAR) avulla. (paljon englanninkielistä termistöä, mikä on oikea ratkaisu?) Kuten kappaleessa 4.4 käsiteltiin, endpoint-laitteet käyttävät tyypin 0 headereita, kun taas bridge-tyypin-laitteet, kuten kytkimet (switch) ja root complexit, käyttävät tyypin 1 headereita. Tyypin 1 headerit tarjoavat kaksi BAR:ia käyttöön ja tyypin 0 header puolestaan kuusi BAR:ia. BAR:ien sijoittuminen ja määrä kahden tyyppisissä Configuration headereissa voidaan nähdä kuvasta 4.12. Asettamalla BAR:ien alimmat bitit halutulla tavalla voidaan määrittää haluttu osoiteavaruustyyppi ja osoiteavaruuden koko. BAR:ien ylempiä bittejä voidaan kirjoittaa ohjelmallisesti. Järjestelmän kerneli, eli käyttöjärjestelmän ydin tai BIOS, joka etsii ja lataa käyttöjärjestelmän keskusmuistin, lukee BAR:iin kovakoodatut alimmat bitit ja kirjoittaa ylempiin bitteihin referenssiosoitteen (Base Address) laitteelle allokoitua osoitealuetta varten. Esimerkiksi yksittäinen endpoint voi tarvittaessa pyytää käyttöönsä kuutta eri osoiteavaruutta. Toisaalta useimmat laitteet pyytävät käyttöönsä [37] mukaan 1-3 eri osoitealuetta. BARien alustamisen jälkeen kyseisten BAR:ien osoitealueeseen osuvat luku- ja kirjoituspyynnöt hyväksytään, sillä pyynnöt osuvat kohdealueelle. BAR:ien sisältö tulkitaan ohjelmallisesti peräkkäin. Käyttämättömät BAR:it voidaan alustaa nollilla kovakoodaamalla, jolloin ohjelma osaa tulkita, etteivät kyseiset BAR:it tule käyttöön. BAR:it voidaan myös ottaa käyttöön järjestyksestä huolimatta, eli Endpoint voi käyttää ainoastaan esimerkiksi BAR1:ä, jolloin loput BAR:it alustetaan nollilla. [37]

4.4.5 Tiedonsiirto

PCIe:tä käyttävän funktion BARien ollessa alustettuina funktio pystyy nyt vaatimaan itselleen minkä tahansa tiedonsiirtotapahtuman, joka on kohdennettu funktion omistamalle osoitealueelle. Toisaalta funktio voi havaita tiedonsiirtotapahtumat vain, mikäli ne ohjataan funktiosta katsottuna sen datavirran saapumissuunnassa sijaitsevaan linkkiin, eli esimerkiksi root portista sen alapuolella sijaitsevaan endpointiin. Toinen esimerkki tapahtumien siirtymisestä voidaan esittää, kun tarkastellaan tiedon kulkua root portin näkökulmasta. Root portin tulee tietää, mitä osoitealueita sijaitsee sen alapuolella, jotta root port kykene välittämään oikean kirjoitus- ja lukupyynnön eteenpäin

sen ensisijaisesta käyttöliittymästä toissijaiseen. Näitä termejä kutsutaan yleisesti vastaavassa järjestyksessä upstream ja downstream. [37]

TLP-pakettien reititys

PCIe-laitteiden muodostamat point-to-point-linkit vaativat paljon tiedon reititystä laitteiden välillä. Saapuva liikenne tarkastetaan ensin virheiden varalta, kun paketti saapuu linkin sisääntuloporttiin. Virheentarkistuksen jälkeen paketti voidaan hyväksyä ja käyttää sisäisesti, ohjata sopivaan lähtöporttiin tai hylätä kokonaan, mikäli paketti ei ole oikeassa kohteessa eikä ohjattavissa eteenpäin. Jokaisen laitteen sisääntuloportti havaitsee PCIe-linkin ollessa täysin toiminnassa kolmen tyyppistä liikennettä. Pakettien tyypit ovat: Ordered Sets, Data Link Layer Packets (DLLP) ja Transaction Layer Packets (TLP). Ordered Sets ja DLLP:t ovat paikallisia datapaketteja, eikä niitä ohjata toiseen linkkiin. TLP-paketit liikkuvat linkistä linkkiin pakettien headerin sisältävän reititystiedon perusteella. Tyypillisesti root complexin liikenne sisältää paketteja, joita vain ohjataan Endpointilta toiselle. Endpointit puolestaan näkevät tyypillisesti vain liikennettä, mikä on ohjattu niille itselleen, eli joko hyväksyvät tai hylkäävät TLP-paketteja. [37]

PCIe käyttää useiden muiden sarjaprotokollien tapaan Split Transaction -protokollaa, joka sallii kohdelaitteen vastaanottavan yhden tai useamman pyynnön, joihin vastataan erillisellä täydentävällä vastauksella (Completion). Edellä mainittua protokollaa käyttämällä voidaan välttyä esimerkiksi odotustiloista tai viivästetyistä siirroista. Sen sijaan, että siirron kohteelle tehdään useita kyselyitä siirtovalmiuden selvittämiseksi, kohde voi vastata Completion-paketilla ollessaan valmis. Näin syntyy vähintään kaksi erillistä TLP-pakettia jokaista tiedonsiirto-operaatiota kohden. Yksi lukupyyntö voi johtaa esimerkiksi useaan Completion TLP:hen. TLP-pakettien Request-tyypit käytiin läpi aiemmin taulukossa 4.5. Taulukosta nähtiin ovatko operaatiot Posted vai Non-Posted -tyyppiä, eli odottavatko operaatiot vastausta Completion-paketin muodossa. Jotta Request-Completion siirtojen tuomalta viiveeltä voitaisiin välttyä, esimerkiksi Memory Write -paketit ovat Posted-tyyppiä, eli Requesterin näkökulmasta tiedonsiirto katsotaan valmiiksi, kun paketit ovat poistuneet Requesterista. Tämä tuo PCIe-protokollan tiedonsiirtoon pienen määrän epävarmuutta, mutta näin voidaan saavuttaa etua väylän suorituskyvyn näkökulmasta. [37]

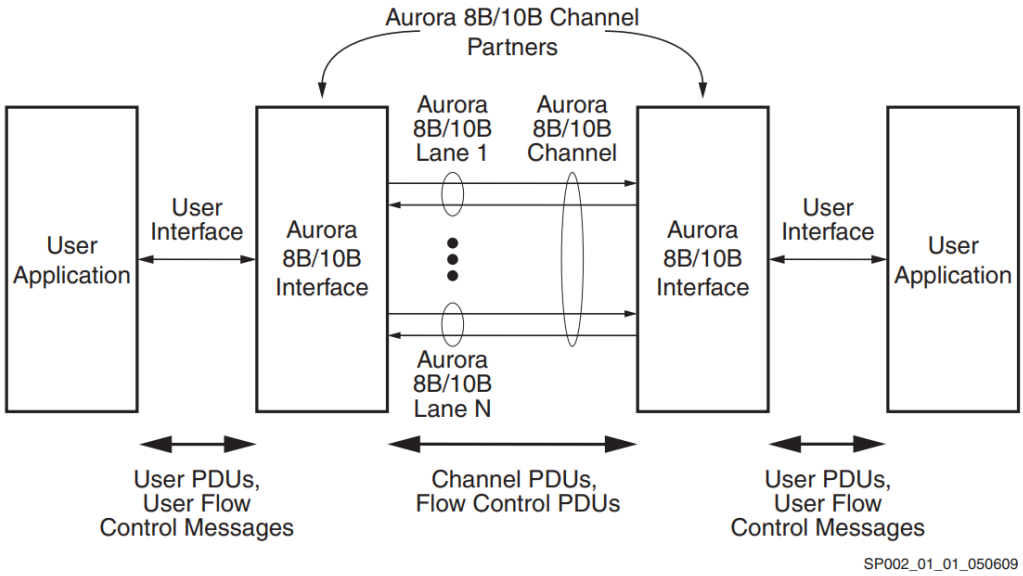
MSI-keskeytykset

PCIe:n toiminnan kannalta ylimääräistä kaistaa käyttävät signaalit eivät ole haluttuja, joten vanhan PCI-mallin mukaiset pinnit keskeytyksille on korvattu Message-Signaled keskeytyksillä (MSI). Tämä tarkoittaa sitä, että keskeytykset on sulautettu samalle kaistalle muun datan joukkoon. Nimestään huolimatta MSI-keskeytykset ovat Posted Memory -tyyppisiä kirjoitusoperaatioita. MSI-kirjoitusoperaatiot poikkeavat muista kirjoitusoperaatioista vain keskeytyksille kohdistetun osoitealueen perusteella. PCIe:n toiminnot ilmaisevat MSI-tukensa MSI Capability -rekisterin avulla, jonka jokainen

toiminto toteuttaa. MSI Capability -rekisterit sisältävät keskeytyksen kohteen muistiosoitteen, lukuarvon, joka tullaan kirjoittamaan kyseiseen osoitteeseen sekä ainutlaatuisten dataankoodattavien viestien lukumäärän. Natiivit PCIe-laitteet tukevat aina vähintään 64-bittisiä osoitteita. MSI-keskeytykset eivät luonnollisesti tapahdu välittömästi, vaan keskeytysviive mitataan hetkestä, jolloin keskeytyssignaali lähtee liikkeelle hetkeen, jolloin ohjelma palvelee keskeytyksen tehnyttä laitetta. MSI-keskeytykset voivat aiheuttaa myös odottamattomia virheitä, sillä virhetiloja MSI-kirjoitusoperaatioissa kohdellaan samaan tapaan kuten, muita Memory Write -virhetiloja. Mikäli MSI-pakettia ei havaita, prosessori ei havaitse myöskään keskeytystä. [37]

4.5 Aurora 8b/10b

Aurora 8b/10b on Xilinx:n kehittämä avoin linkkitason protokolla, jota käytetään siirtämään dataa kahden pisteen välillä. Protokolla käyttää sarjamuotoisia kaistoja (lanes) siirtoon. Ensisijaisia käyttökohteita ovat chip-to-chip ja board-to-board -sovellukset. Aurorasta on tarjolla myös Aurora 64b/66b-versio, joka käyttää nimensä mukaisesti 64b/66b-merkistökoodausta 8b/10b sijaan [34]. Aurora 8b10b-kanava koostuu yhdestä tai useammasta kaistasta, joista jokainen on full-duplex -sarjaliitäntä. Aurora-speksissä laitteita, jotka kommunikoivat kanavan yli kutsutaan termillä channel partners. Kuva 4.15 esittää yleiskuvan kahden Aurora 8b10b channel partnersin liitännästä. [38]



Kuva 4.15. *Aurora 8b10b-kanavan yleiskuvaus [38]*

Tietovuo koostuu käyttäjäsovelluksen ja Aurora-käyttöliittymän välisistä käyttäjän Protocol Data Unitien (PDU) ja vuonohjausviestien siirrosta, sekä Aurora-kanavan läpi liikkuvista Channel-PDU ja vuonohjaus-PDU -siirroista. [38]

Taulukko 4.6. *Tiedonsiirron prioriteetit*

Data Type	Priority
Clock Compensation Sequences	Highest
Initialization Sequences	
Native Flow Control PDUs	
User Flow Control PDUs	
Channel PDUs	
Idle Sequences	Lowest

4.5.1 Tiedonsiirto ja vastaanotto

Auroran 8b10b-tiedonsiirrossa pienin tiedonsiirron yksikkö, joka siirtyy kanavan yli, on merkkipari, eli pienin siirrettävä määrä dataa on kaksi merkkiä. Aurora-kanavan sisältämä informaatio muodostuu aina useista merkkipareista. Käyttäjäsovellus syöttää Aurora-kanavan käyttöliittymään virran kahdeksanbittisiä tavuja eli oktetteja. Oktettien virta siirtyy kanavan yli merkkipareina. [38]

Aurora 8b10b-kanavan yli siirtyy kuuden eri tyypin dataa.

- Clock Compensation Sequence: Jono valvontamerkkejä, joita käytetään estämään Channel Partnerien kellotaajuuksien eroavaisuuksien aiheuttamaa vastaanottimen yliajoa.
- Initialization Sequences: Neljä järjestettyä sarjaa, joita käytetään idle sequencen kanssa Aurorakanavan alustamisessa.
- Native Flow Control PDUs: Linkkitason virran valvonnan PDU.
- User Flow Control PDUs: Käyttäjäsovelluksen luomat datavuon valvonnan viestit.
- Channel PDUs: Tiedonsiirtoa varten paketoitu käyttäjän PDU.
- Idle Sequences: Valvontamerkkien sarja, joka siirretään silloin, kun muuta dataa ei siirretä.

Aurora 8b10b-datatyyppien prioriteettijärjestys on esitetty taulukossa 4.6. Flow Control -pyyntöä käsitellessä lähettäjän prioriteettijärjestys vaihtuu siten, että Idle Sequences saa korkeamman prioriteetin kuin Channel PDU. Näin estetään datakehyksien siirto, kunnes Flow Control -laskuri on valmis. [38]

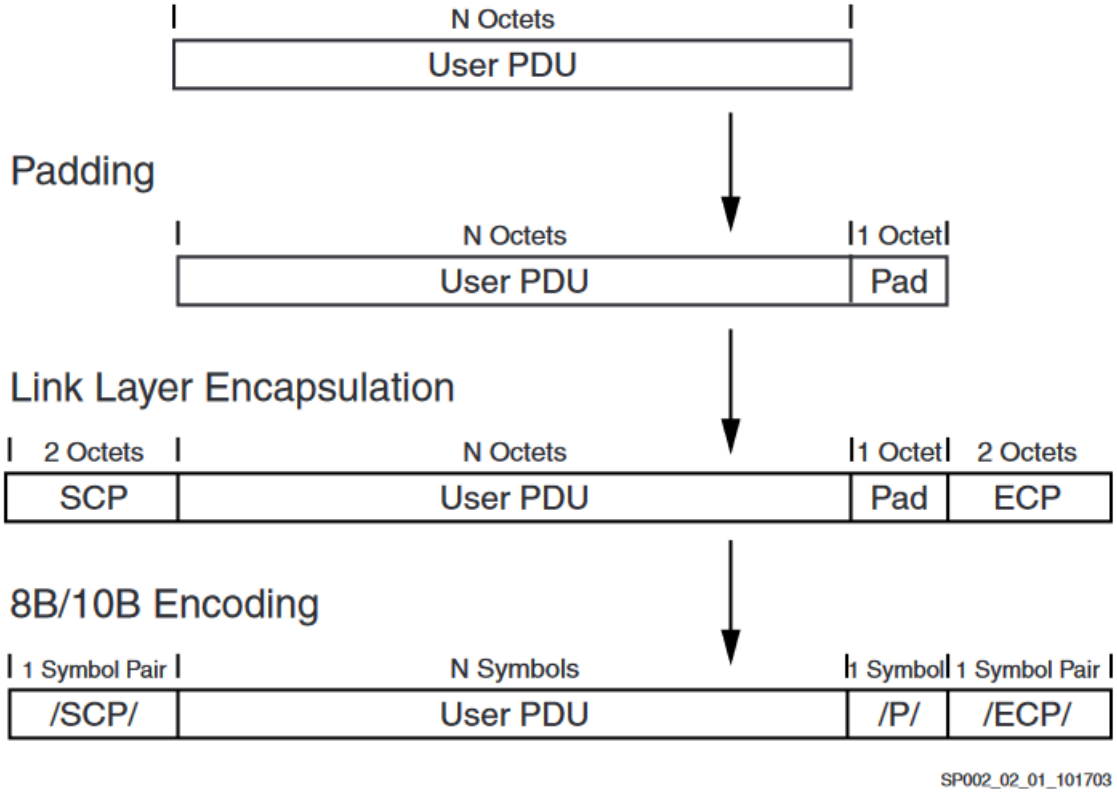
Aurora 8b10b protokolla tukee kahta vuonhallintamekanismia. Ensimmäinen näistä on Native Flow Control, joka on linkkitason ohjausmekanismi. Native Flow Control PDU:t luo ja tulkitsee Aurora 8b10b käyttöliittymä. Nämä PDU:t tuotetaan tyypillisesti silloin, kun vastaanottajapuolen datavarasto on tyhjentynyt. [38]

4.5.2 Käyttäjä-PDU-pakettien siirto

Käyttäjä-PDU-pakettien siirto tapahtuu neljässä vaiheessa:

- Padding, parillisuuden tarkistus ja mahdollinen 8 bitin lisäys varsinaisen viestin perään.
- Channel PDU:n paketointi viestin ympärille.
- 8B/10B-linjakoodaus.
- Sarjamuunto ja kellon sulautus linjakoodaukseen.

Aivan aluksi Padding-vaiheessa tarkistetaan, muodostuuko lähetettävä viesti parillisesta määrästä symboleita. Vaatimus johtuu Auroran 8b/10b linjakoodauksen luonteesta. Koko edellä mainittu paketointiprosessi on esitetty kuvassa 4.16.



Kuva 4.16. *PDU-paketointiprosessi lähetyksessä [38]*

Parillisuuden tarkistuksen jälkeen käyttäjä-PDU paketoidaan ohjaussymboleilla, jotka merkitsevät kanava-PDU:n alkukohdan ja loppukohdan. Ennen siirtoa kokonaisuus linjakoodataan 8B/10B-menetelmällä. Viimeisenä linjakoodattu paketti muunnetaan sarjamuotoon ja lähetetään eromuotoisella Non Return to Zero (NRZ) -formaatilla. [38]

4.6 Serial RapidIO

RapidIO-liitännän arkkitehtuuri on avoin standardi, joka tarjoaa monipuolisia ratkaisuja sulautetuille sovelluksille. Nämä sovellukset voivat olla esimerkiksi prosessoreiden, muistien tai muistia käyttävien I/O-laitteiden liittämistä toisiinsa. RapidIO on tarkoitettu ensisijaisesti järjestelmän sisäiseksi käyttöliittymäksi, joka sallii piirien tai piirikorttien

välisen kommunikaation. Liitännän speksi määrittelee rinnakkaistyyppisen- sekä sarjamuotoisen RapidIO-liitännän. Molemmat liitäntätyypit jakavat samat ohjelmointimallit sekä siirto- ja muistimekanismit. Sarjamuotoista RapidIO:ta käytetään tyypillisesti esimerkiksi backplane eli kokoajapiirikorttityyppisissä sovelluksissa, sekä DSP sovelluksissa. RapidIO-liitäntä on PCIe:n tavoin määritelty kerroksittaiseksi arkkitehtuuriksi, joka sallii tulevaisuuden laajennukset säilyttäen samalla yhteensopivuuden vanhoihin standardeihin. [39]

RapidIO:n operaatiot perustuvat pyyntö- ja vastaustyyppisiin siirtoihin. Järjestelmä käyttää pakettityyppisiä elementtejä endpointtien väliseen kommunikaatioon. [39] käyttää termejä master tai aloitteentekijä -initiaattori. Mainitun tyyppiset endpointit rakentavat pyyntöjä, jotka siirretään kohteeseen. Kohde puolestaan lähettää vastaustyyppisen paketin takaisin täydentäen siirron. Tiedonsiirron kulkua ohjataan tehtävään tarkoitetuilla symboleilla. Näitä ohjaussymboleita käytetään esimerkiksi pakettien hyväksymiseen, datavuon ohjausinformaation säilyttämiseen ja ylläpitotoimintoihin.

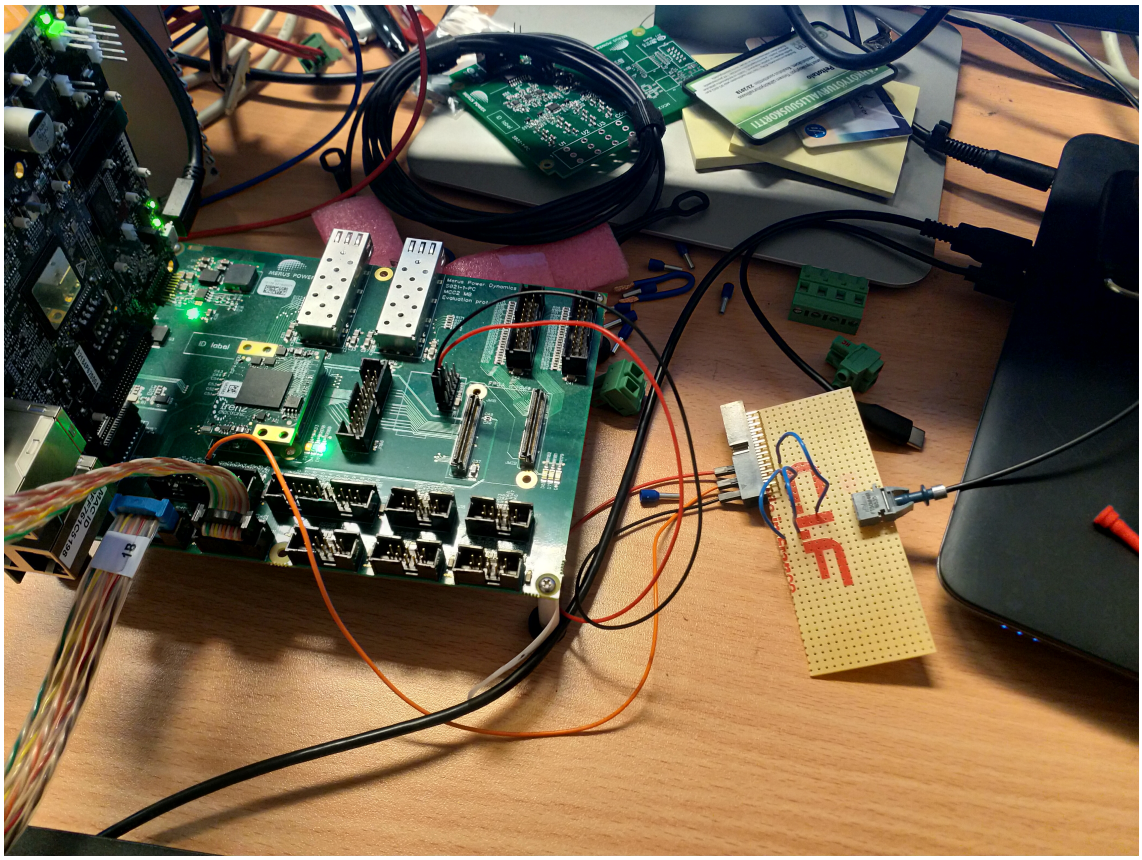
5 MCP-PROTOTYYPPI

Merus Control Platform (MCP) -prototyypin evaluointivaiheessa haluttiin toteuttaa ja todeta toimivaksi vaihtosähkömuuttajan ohjauksen toimintaketju. Kappaleet 5 ja 6 keskittyvät prototyypin FPGA-moduulin toiminnallisuuksien toteuttamiseen, prototyypin verifiointiin ja sarjaväylien suorituskyvyn arviointiin. DSP-piirin toteuttama säätöalgoritmien laskenta ja muu piirille suunniteltu toiminnallisuus ei kuulu tämän työn aihepiiriin.

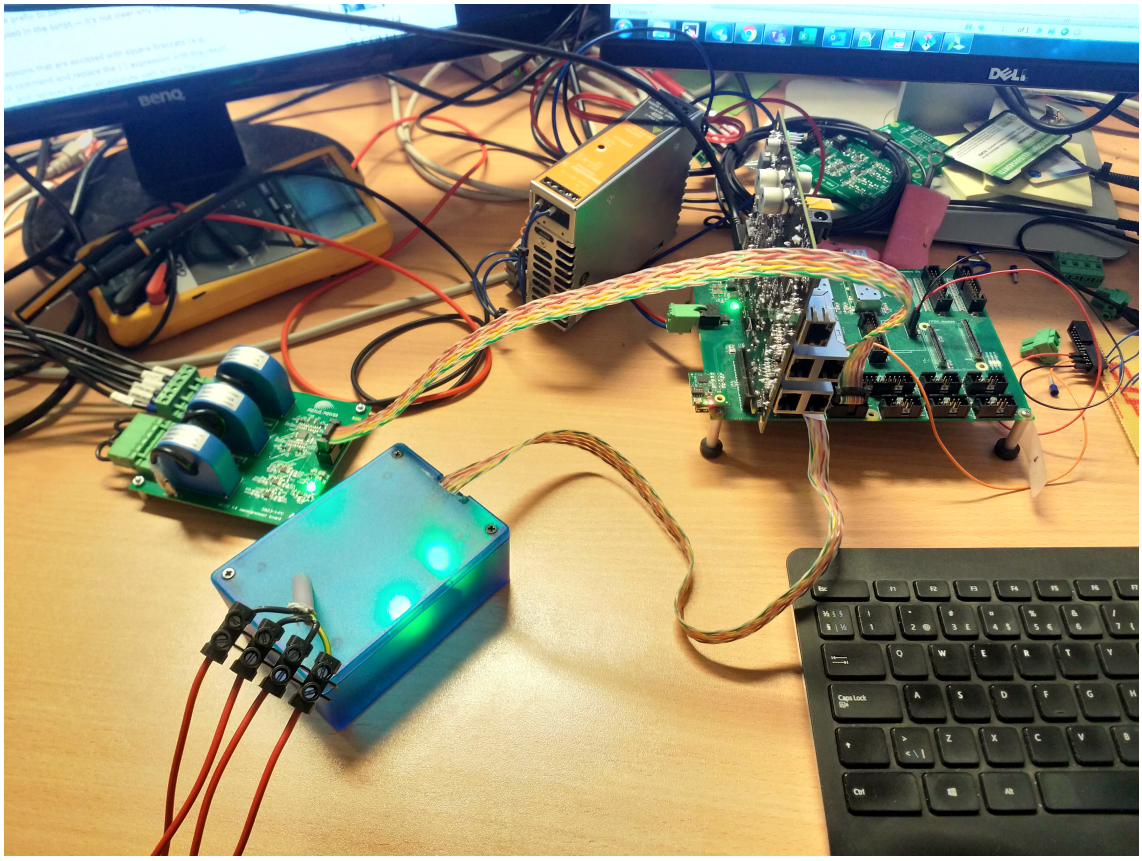
Testijärjestelmä koostuu jännite- ja virtamittauspiirikorteista, jotka mittaavat kuorman virtaa ja jännitettä. Mittauspiirikorttien signaalit tulevat sisään MCP-kokoajapiirikortille, ja eteenpäin FPGA-moduulille, joka kokoaa mittaussignaalit yhteen. FPGA-moduuli kirjoittaa mitatut suureet Block RAM -muistilohkoon jokaisella ohjausjaksolla. DSP-piiri omalla SoC-piirillä lukee Block RAM -lohkon PCI Express -väylän ylitse ja laskee virtareferenssin mitattujen virta- ja jännitesuureiden perusteella. DSP päivittää lasketun virtareferenssin Block RAM -lohkoon, josta FPGA lähettää virtareferenssit eteenpäin S/PDIF-koodattuna valokuitukortille. Valokuitukortti lähettää virtareferenssin Merus Powerin A2-sarjan aktiivisuodatinmoduulille kuitulinkin yli. Edellä mainittu aktiivisuodatinratkaisu perustuu kolmitasotopologiaan ja se voi toimia sekä reaaliaikaisena yliaaltosuodattimena että loistehon kompensoijana [40].

MCP-prototyyppi koostuu Merus Powerin -kokoajapiirikortista, Trenz Electronicsin TE0714 FPGA -moduulista Xilinxin Artix-7 FPGA-piirillä, Texas Instrumentsin K2G-ICE-evaluointipiirikortista sekä Merus Powerin jännite- ja virtamittauspiirikorteista. TI:n K2G-ICE-kortti sisältää 66AK2G12 System on Chip -piirin, joka on varustettu yhdellä C66x DSP:llä sekä ARM A15 -piirillä. MCP-prototyyppi löytyy kuvasta 5.1. Pohjakortilta löytyy TE0714 SoM -kortin liittimien lisäksi PCIe-väylän liitin, johon TI:n K2G-ICE kiinnittyy. Lisäksi kortissa on useita yleiskäyttöisiä liittimiä sekä liittimet virta- ja jännitemittauskorteille. Pohjakortilta löytyy myös 100 MHz:n referenssikello PCIe:lle kahdella lähdöllä. Toinen lähtö on syötetty FPGA:lle JM1-liittimeen (MGT CLK0 kuvassa 5.3) ja toinen PCIe-liittimeen.

Seuraavassa kuvassa 5.2 näkyvät myös virta- ja jännitemittauskortit, joiden ulostulot viedään lattakaapeleilla kokoajapiirikortin liittimiin.



Kuva 5.1. *MCP-prototyyppi*



Kuva 5.2. *MCP-evaluointipiirikortin prototyyppi ja mittauskortit*

Taulukko 5.1. *Tiedonsiirtoväylien vertailu*

SoC / FPGA	66AK2G1x	TMS320F28388D	TMS320C6657	Xilinx Artix-7	ilmainen/ maksullinen
SRIO	-	-	4 Lane SRIO 2.1	1x, 2x, 4x Lane Gen 2	maksullinen
EMIF	32-bit sisäinen	Dual core 16-bit (1 core 32-bit)	16-bit EMIF	Custom Logic	ilmainen
Aurora 8B/10B	-	-	-	1x, 2x, 4x 0,5-6,25 Gbps	ilmainen
PCIe	Single Lane Gen 2	-	1-2 Lane Gen 2	x4 Gen2	ilmainen

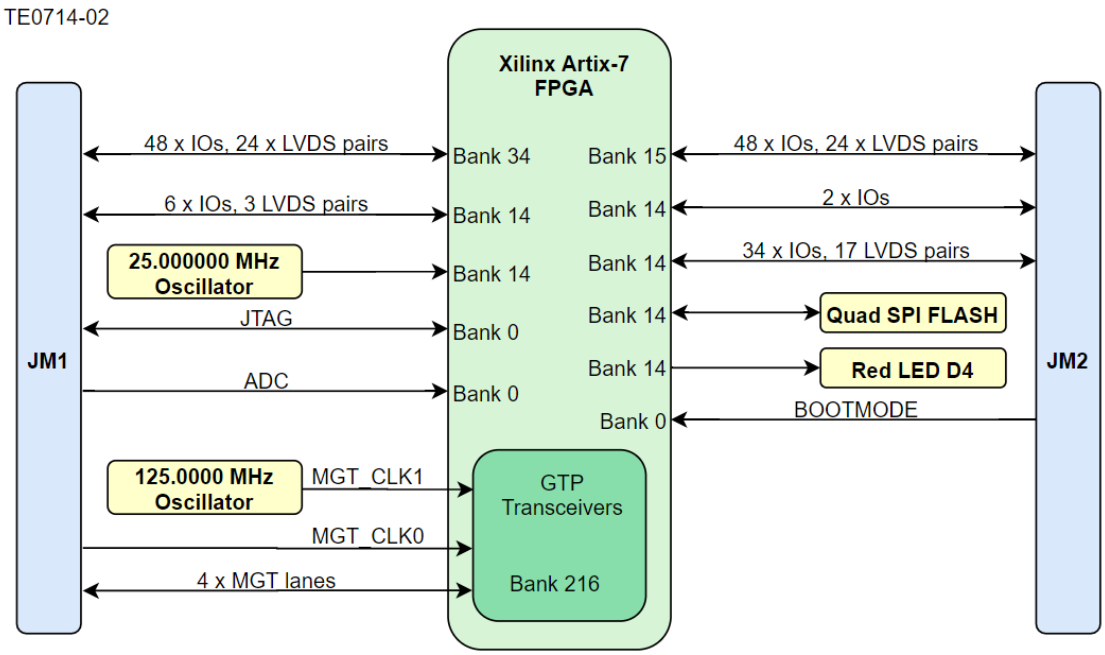
5.1 Tiedonsiirtoväylän valinta

MCP-prototyypin tiedonsiirtoväylän valinnassa haluttiin preferoida sarjaväylää rinnakkaisväylän sijasta, jotta vältyttäisiin monimutkaisesta ja suuren pinta-alan käyttävästä piirikytkennästä. Esimerkiksi pelkästään kolmivaiheiset virta- ja jännitemittaukset tuottavat 16 bitin tarkkuudella kuudesta eri sisäänmenosta mittausdataa kerran säätösilmukan suorituksen aikana. Varteenotettavia vaihtoehtoja nopealle sarjaväylälle olivat Xilinxin Auroraprotokolla, Serial RapidIO sekä PCI Express. Myös Texas Instrumentsin External Memory Interface eli EMIF on mainitsemisen arvoinen, sillä monet TI:n DSP- ja SoC-alustat käyttävät väylää ulkoisiin laitteisiin yhdistämiseen. SRIO-väylälle löytyi vähän tukea eri prosessoreilta. Lisäksi SRIO:n maksulliset IP:t vaikuttivat valintaan. Aurora on kevyt kahden pisteen välinen kommunikaatioprotokolla, jota käytetään FPGA-piirien väliseen siirtoon. Auroralta ei löydy prosessoreille tukea, joten vaihtoehto karsiutuu pois, mutta tulevaisuudessa FPGA-piirien välinen siirto Auroralla voi olla käyttökelpoinen konsepti. Auroran siirtonopeus on myös joustava verrattuna esimerkiksi kiinteätä siirtonopeutta käyttäviin SRIO- ja PCIe-väyliin [41]. Taulukkoon 5.1 on koottu, mitä tiedonsiirtoväyliä eri prosessorialustat tukevat. TMS320F28388D on TI:n C2000-sarjan mikrokontrollerialusta, TMS320C6657 on TI:n dual-core DSP-alusta, 66AK2G1x on TI:n multicore DSP+Arm System-on-Chip -alusta ja Artix-7 on Xilinxin 7-sarjan keskikastin suorituskyvyn FPGA.

Väyläksi MCP-prototyyppiin valittiin PCI Express, sillä väylälle on tarjolla hyvä dokumentaatio. PCIe-liitäntästandardia pitää yllä usean tietotekniikka-alan yrityksen ryhmä, PCI Special Interest Group [42]. Tämän lisäksi Xilinxin kirjastot tarjoavat 7-sarjan FPGA-piireille tarkoitetun PCIe-logiikkalohkon ja kattavan dokumentaation lohkon käyttöön. Lohkon mukana Xilinx tarjoaa esimerkkitoteutuksen Verilog- ja VHDL-laitteistonkuvauskielillä. [43]

5.2 FPGA-moduulin ominaisuudet

Prototyypin FPGA-ratkaisuksi valittiin Trenz Electronicsin TE0714 System on Module, josta löytyy suuri määrä I/O-pinnejä. SoM:sta löytyy muun muassa neljä korkean suorituskyvyn lähetinvastaanotinkaistaa. TE0714-lohkokaaviosta 5.3 nähdään miten FPGA-piirin I/O:t on ryhmitelty SoM-moduulin liittimiin. SoM-moduuli tarjoaa FPGA-piirin käyttöön 25 megahertsin oskillaattorin, josta projektin kellosignaalit johdetaan. Lohkokaavion mukaan GTP-lähetinvastaanottimen käyttöön löytyy 125 MHz:n oskillaattori, mutta MGT-käyttöön soveltuvia kellosignaaleita voidaan tuoda myös moduulin ulkopuolelta. [44]



Kuva 5.3. *TE0714 SoM -lohkokaavio [44]*

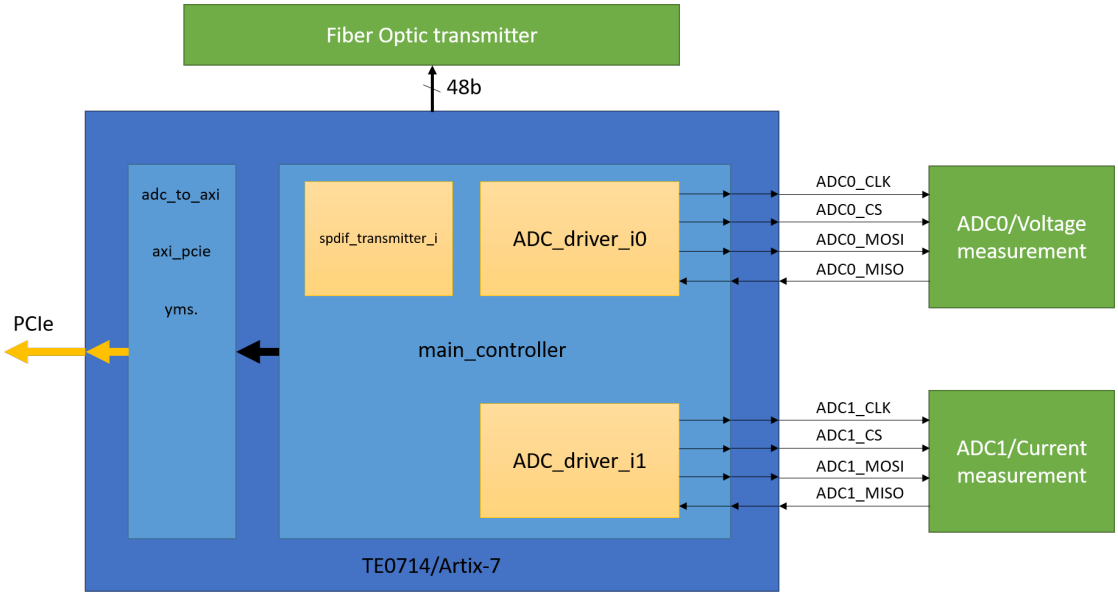
TE0714 SoM:n FPGA-piiriksi valittiin Xilinxin Artix-7. Kyseiset FPGA-piirit on optimoitu matalan tehon sovelluksille, jotka vaativat lähetinvastaanottimia. Artix-7 FPGA -perhe on jaettu alempiin luokkiin laitteen ominaisuuksien mukaan. MCP-prototyypissä käytetyn Artix-7:n laitteeksi valikoitui keskikastin XC7A50T, jonka ominaisuuksista löytyy vaadittu integroitu PCI Express -lohko sekä neljä GTP-lähetinvastaanotinkaistaa. [45]

Artix-7:n lisäksi Xilinxin 7-sarjan FPGA-tuotepakettiin kuuluu kolme muuta FPGA-perhettä. Nämä ovat Spartan-7, Kintex-7 ja Virtex-7. Kaikkiin 7-sarjan piirien ominaisuuksiin sisältyy edellämainittu sulautettu PCI Express -lohko, joka mahdollistaa x8 Gen3 endpoint- ja root port -suunnittelun. Xilinx:n mukaan 7-sarja pystyy sisäänrakennettujen lähetinvastaanottimien avulla vähintään 600 Mbps:n sarjasiirtoon ja maksimissaan 6,6 ja 28,05 Gbps:n bittinopeuksiin. Seuraavassa taulukossa 5.2 on vertailtu 7-sarjan ominaisuuksia lähetinvastaanottimien ja PCIe:n osalta. [45]

MCP-prototyypissä FPGA:n toiminnallisuuksiksi määriteltiin tehoelektroniikan sekä mittauskorttimoduulien käyttöliittymän toteuttaminen. MCP:n mittauskortit kytkeytyvät

Taulukko 5.2. *7-sarjan FPGA-perheiden lähetinvastaanotinvertailu*

Max. Capability	Spartan-7	Artix-7	Kintex-7	Virtex-7
Transceivers	-	16	32	96
Transceiver Speed	-	6.6 Gb/s	12.5 Gb/s	28.05 Gb/s
PCIe Interface	-	x4 Gen2	x8 Gen2	x8 Gen3



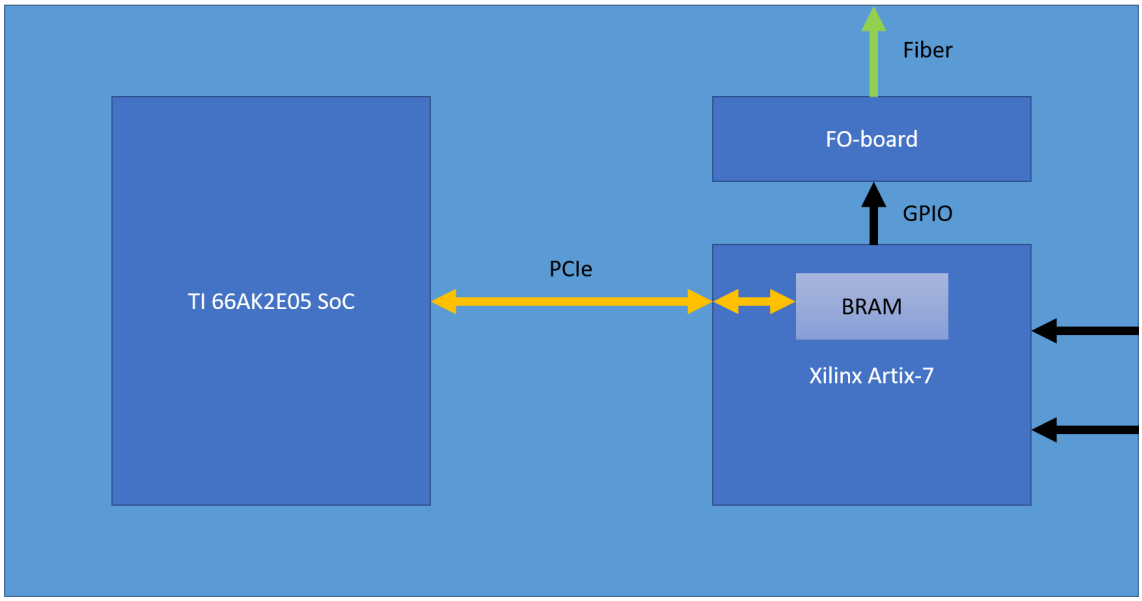
Kuva 5.4. *FPGA:n RTL-toteutuksen moduulit ja SPI-väylän signaalit*

FPGA-piirille Serial Peripheral Interface (SPI) -väylän avulla. Lohkokaavio mittauskorttien kytkeytymisestä on esitetty kuvassa 5.4.

MCP:n mittauskortit sisältävät kumpikin kolme Texas Instrumentsin ADS8866-analogia-digitaalimuunninta, yksi jokaiselle vaiheelle. ADS8866 on 16-bittinen 100 kHz:n näytteenottotaajuuden piiri. Laite tarjoaa SPI-yhteensopivan daisy chain -toimintoa tukevan sarjakäyttöliittymän, jota käytettiin prototyypissä rinnakkaisväylällä toteutettujen AD-muuntimien sijaan. [46]

FPGA:n ja DSP:n välinen kommunikaatio toteutettiin kuvan 5.5 mukaisesti PCI Express -väylän avulla. MCP-prototyypin toiminnan voi yksinkertaistaa viiteen vaihteeseen.

1. ADC-muunnos valmis, talleta ADC-data rekistereihin.
2. ADC-data siirretty BRAMiin, lähetä MSI-keskeytys DSP:lle.
3. DSP laskee mittausdatasta virtareferenssin ja tallettaa sen BRAMiin.
4. FPGA muuntaa virtareferenssin S/PDIF-protokollan mukaiseen muotoon ja eteenpäin valokuitukortille.
5. Ohjaussignaali saapuu valokuidusta vaihtosähkömuuttajan ohjauskortille.



Kuva 5.5. *MCP-prototyypin laitteiston rakenne*

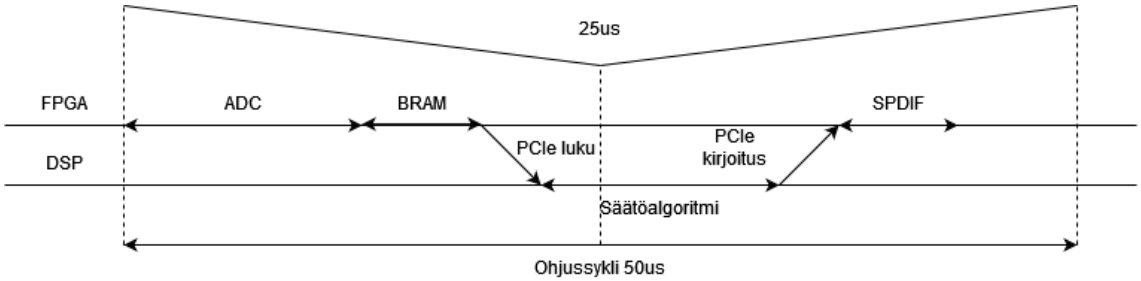
5.3 FPGA-piirin toiminta

Yksinkertaisempien FPGA:n toiminnallisuuksien toteutuksessa käytettiin VHDL eli VHSIC (Very High Speed Integrated Circuit) Hardware Description Language -laitteistonkuvauskieltä. Edellä mainittuja yksinkertaisempia toiminnallisuuksia on esimerkiksi kuvassa 5.4 esitetty main_controller -lohko, jonka tehtävänä on toteuttaa FPGA:n ylätason ohjaussykliä, sekä hallita ja järjestää dataa. Myös mittakorttien mastereina toimivat ADC_driver_ix-moduulit toteutettiin VHDL laitteistonkuvauskielellä.

FPGA-projektin hallintaan käytettiin Xilinxin Vivado IDE:tä, joka tarjosi perinteisten ominaisuuksien, kuten simulaatiotyökalujen, synteesin, implementaation ja bitstreamin kääntämisen lisäksi graafisia työkaluja valmiiden Intellectual Propertyjen (IP) integroimiseksi projektiin. Vivadon IP-keskeinen design flow mahdollisti moduulien lisäämisen projektiin useista lähteistä, kuten Register Transfer Level (RTL) -lähdetiedostoista ja Vivado IP-paketeista [47]. RTL-lähdetiedostot ovat esimerkiksi VHDL- ja Verilog-tyyppisiä tiedostoja. FPGA-projektin Vivado Block Design -lohkokaavio löytyy kokonaisuudessaan liitteestä A. FPGA:n ylätason ohjaussykliä toteuttava main_controller on lisätty projektiin RTL-moduulina ja loput toiminnallisuudet Xilinxin IP-moduuleina. AXI Memory Mapped to PCI Express -lohko tarjoaa rajapinnan AXI4-väylää käyttävän sulautetun järjestelmän ja PCI Express -järjestelmän välille. Moduuli kääntää AXI4-muistiinluku- ja AXI4-muistiinkirjoitusoperaatiot PCIe Transaction Layer Packets (TLP) -paketeiksi ja vastaavasti toisinpäin [48]. AXI on protokolla, joka tukee master-slave-komponenttien pursketyyppistä korkeataajuista kommunikaatiota [49]. Adc_to_axi-moduuli kirjoittaa jännite- ja virtamittausdatan sekä virtareferenssidatan AXI4-Lite-väylään. AXI4-Lite toteuttaa yksinkertaisempaa ohjausrekisterityyppistä käyttöliittymää, joka ei vaadi kaikkia AXI4-toiminnallisuuksia [49].

5.3.1 MCP-prototyypin ajoitukset

Järjestelmän ajoituksia ohjaa FPGA, ja kaikki kriittinen laskenta tapahtuu yhdessä ohjaussyklissä. Prototyypin ohjaussykli on muokattava, ja evaluointivaiheeseen valikoitui 50μs:n eli 20 kHz:n taajuus, joka on jaettu kahteen 25μs:n ramppiin. Kaikki datansiirto sekä säätöalgoritmit tulee ehtiä suorittaa jaksollisesti ennen seuraavan ohjaussyklin alkua. Ohjaussykli toimi prototyypissä hieman jäykästi, ja esimerkiksi PCIe-väylän MSI-keskeytykset ilmoittivat datan siirrosta ainoastaan DSP-piirin suuntaan, joten jatkokehityksen kannalta datanpäivitys myös FPGA:n suuntaan tarvitsee tunnistuksen. Prototyypin ajoituskaavio on esitetty kuvassa 5.6.



Kuva 5.6. *MCP-prototyypin ajoituskaavio*

Ohjaussyklin alussa FPGA lähettää Start Of Conversion -signaalin mittakorttien ADC-muuntimille. ADC-muunnoksen valmistuttua ADC-datarekisterien sisältö kirjoitetaan FPGA:n Block RAM -lohkoon ja DSP-kortille lähetetään PCIe-väylän yli MSI-keskeytys. DSP lukee uuden mittausdatan FPGA:n Block RAM:sta ja laskee mittausdatasta vaihtosähkömuuttajalle virtareferenssin. Tämä virtareferenssi kirjoitetaan Block RAM:iin. Lopuksi virtareferenssi muunnetaan valokuitukorttia varten S/PDIF-protokollan mukaiseksi 48-bittiseksi jonoksi.

5.3.2 Jännite- ja virtamittauksen ADC-muuntimet

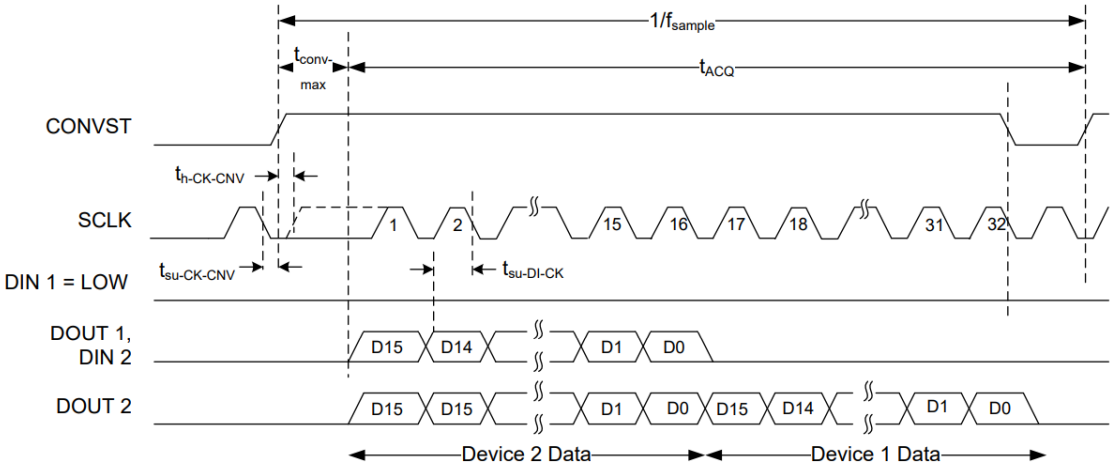
MCP-prototyypin jännite- ja virranmittauskortit käyttävät kumpikin kolmea Texas Instrumentsin ADS8866 ADC -muunninta kytkettynä daisy chain -kaskadiin. 16 bitin tarkkuuteen pystyvät rinnakkaisväylällä toteutetut muuntimet olisivat vaatineet yhteensä 96 rinnakkaista kaistaa toteuttaakseen kolmivaiheisen virta- ja jännitemittauksen. AD-muuntimien toimintaa rajoittavat erityyppiset viiveet, kuten muunnosviive ja datansiirtoviive. Esimerkiksi ADS8866-piiri käyttää sisäistä kelloa näytteistykseen. Muunnoksen kesto voi vaihdella, mutta on kuitenkin rajoitettu [46] ilmoittamiin raja-arvoihin. Kyseinen piiri käyttää ulkoista kellosignaalia ainoastaan sarjadatan lukuoperaatioon. Mittausdatan noutaminen prosessoitavaksi toimii ketjuttamalla muuntimet daisy chainiin, mutta prosessia saa tehokkaammaksi käyttämällä esimerkiksi nopeampaa näytteistystä tukevaa piiriä sekä omaa datasignaalia jokaiselle mittaukselle daisy chainin sijaan. [46]

Kuvassa 5.7 on esitetty ADS8866-piirin ajoituskaavio daisy chain -moodissa. Piirin

Taulukko 5.3. *ADS8866 daisy chain -ajoitusvaatimukset*

		MIN	TYP	MAX	UNIT
t_{ACQ}	Acquisition time	1200			ns
t_{conv}	Conversion time	500		8800	ns
$1/f_{sample}$	Time between conversions	10000			ns

toiminta daisy chain -moodissa vaatii SCLK-kellosignaalin tahdistamaan ADC-muunnoksen. CONVST-signaali toimii aloitusbittinä muunnokselle ja datalinjat sisältävät näytteistetyn datan. [46] taulukoi ADS8866-piirin daisy chain -moodin ajoitusvaatimukset. Taulukkoon 5.3 on koottu ADC-ajurin kannalta oleelliset ajoitusvaatimukset. Lisäksi [46] asettaa vaatimuksia signaalien nousu- ja laskuajoille.



Kuva 5.7. *ADS8866-ajoituskaavio [46]*

VHDL-laitteistokuvauskielellä kirjoitetun ADC-ajurin toimintaa kuvaa yksinkertaistettu tilakone 5.8. Tilojen toiminta selitetään seuraavaksi tarkemmin:

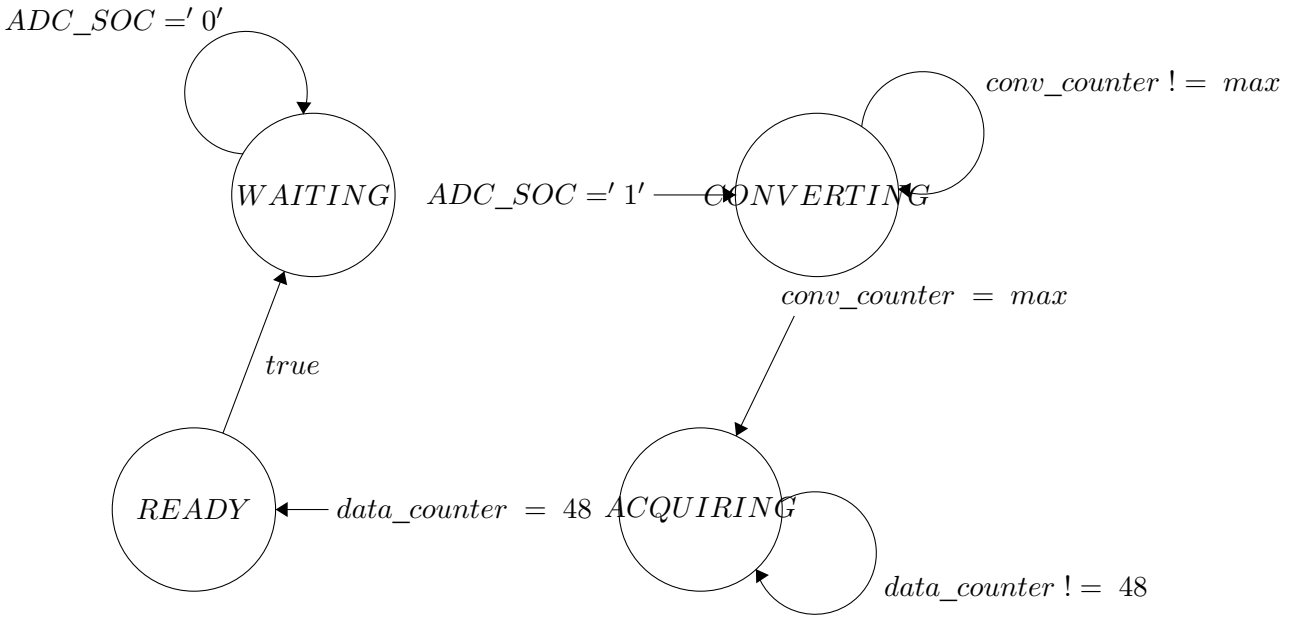
- WAITING
  - ADC-ajurin alkutilassa *WAITING*, ajuri odottaa ylemmän tason *main_controller*-moduulilta Start Of Conversion -signaalia *ADC_SOC*
  - SOC-signaalin saavuttua alustetaan conversion time -laskuri ja ADC-datarekisteri, nostetaan *CONVST* signaali ylös
- CONVERTING
  - conversion-laskuri nostaa laskurin arvoa jokaisella kellon nousevalla reunalla
  - kun laskurin maksimiarvo eli taulukon 5.3 conversion-ajan maksimiarvo 8800 ns on täyttynyt, siirrytään *ACQUIRING* tilaan
- ACQUIRING
  - siirretään ADC-datarekisterin bittejä ja otetaan uusi databitti MISO-kanavasta sisään datarekisterin LSB-paikalle

Taulukko 5.4. *ADC-ajurin logiikkasimuloinnin ajoitukset*

tapahtuma	loppuhetki	alkuhetki	erotus/kesto	yksikkö
Conversion	33827.500	25027.500	8800.000	ns
Acquisition	75027.500	33907.500	41120.000	ns
Time between conversions	75027.500	25027.500	50000.000	ns

  - kun datarekisteri on täynnä, asetetaan conversion-signaali alas ja siirrytään tilaan *READY*

- READY
  - nollataan datarekisteri ja siirrytään alkutilaan *WAITING*



Kuva 5.8. *ADC-ajurin yksinkertaistettu tilakone*

ADC-ajurin toiminta todennettiin kirjoittamalla ADC-ajurille testipenkki, jonka jälkeen logiikan toiminta simuloitiin Vivadon logiikkasimulaattorilla. ADC-ajurin simulaation aaltomuotoikkuna on esitetty liitteessä B. Simuloinnin oleellisimmat ajoitukset on esitetty taulukossa 5.4. Voidaan todeta, että adc-muunnoksen maksimiaika 8800 ns täyttyy. Myös acquisition-ajan minimiaika täyttyy, sillä kyseinen aika määritellään ensimmäisen bitin siirtymisen alusta uuden ADC-muunnoksen alkuun 5.7 kaavion mukaan. Time between conversions täyttyy myös, sillä uusi adc-muunnos aloitetaan vain joka 50. mikrosekunti ohjaussyklin alussa.

5.3.3 Ylätason ohjauslohkon toiminta

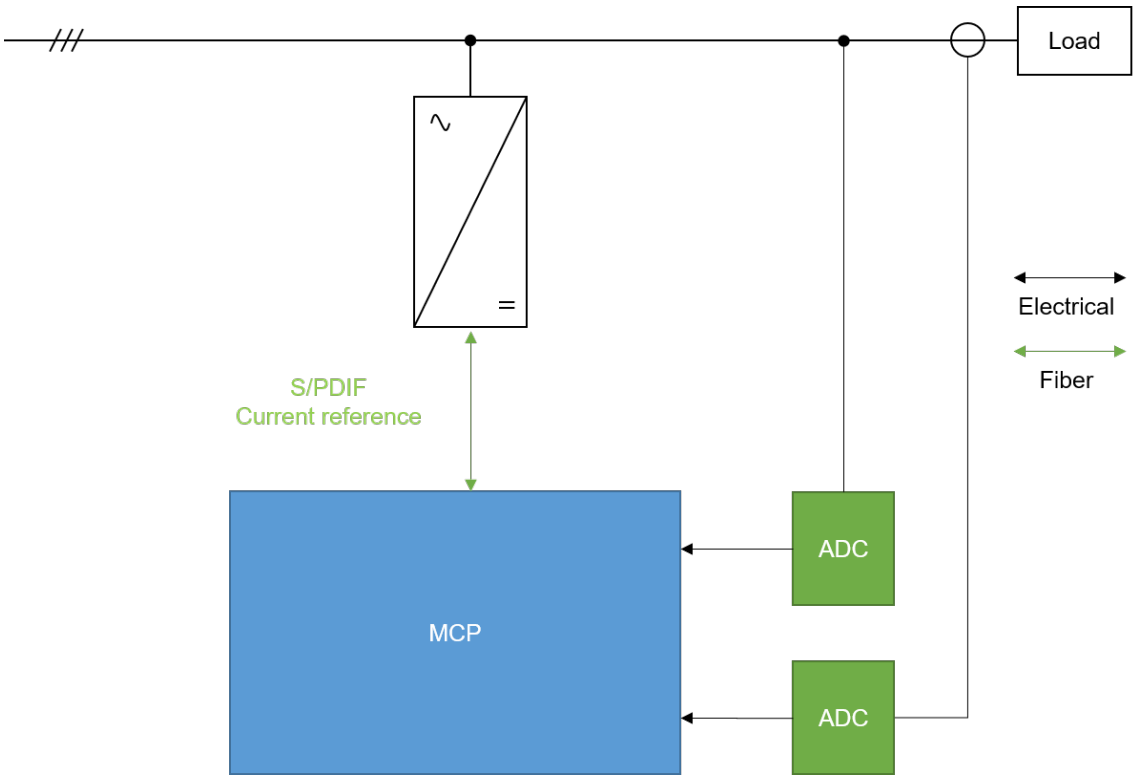
FPGA-piirin ylätason ohjauslohko ohjaa MCP-prototyypin toimintaa ja noudattaa 5.6 mukaista ajoituskaaviota. Moduuli odottaa käyttöjännitteen ilmaantumisen jälkeen PCIe-linkin muodostumista. PCIe-linkin oikea tila sallii varsinainen ohjaussyklin

käynnistymisen. FPGA-piiri johtaa kellosignaalit TE0714-moduulilta 5.3 saapuvasta 25 MHz:n oskillaattorin signaalista. PCIe-linkin muodostamisen jälkeen piiri odottaa 25μs-laskurin kasvamista huippuarvoonsa ja asettaa sitten ADC-muuntimille lähetettävät Start Of Conversion -bitit. ADC-ajureilta saapuvat ready-signaalit antavat herätteen ADC_to_AXI-moduulille, joka kirjoittaa datarekisterien sisällön FPGA:n Block RAMiin. Kun kirjoitusoperaatio on valmis, DSP-kortti saa MSI-keskeytyksen, jolloin BRAM luetaan PCIe-väylän yli ja virtareferenssien laskenta aloitetaan. Ohjaussyklin nousevan rampin puolivälissä virtareferenssit luetaan BRAMista ja rekisterin bitit järjestetään uudelleen S/PDIF-siirtoa varten. Lopulta S/PDIF-moduulissa linjakoodattu data siirretään ulostuloon ja edelleen valokuitulähettimeen. Ylätason ohjauslohkon logiikan toiminta todennettiin kirjoittamalla sille testipenkki ja simuloimalla osaprojekti Vivadon logiikkasimulaattorilla.

6 MCP-PROTOTYYPIN VERIFIOINTI

Edellisessä kappaleessa esitetyn prototyypin toimintaa testattiin Merus Powerin A2-aktiivisuodattimen virtareferenssiohjauksella. Testijärjestelmässä aktiivisuodatinmoduuli toimi ulkoisella virtareferenssillä, pitäen samalla yllä DC-linkkiä. Prototyypin toiminta koostuu virta- ja jännitemittausdatan kokoamisesta, virtareferenssin laskennasta sekä virtareferenssin lähettämisestä ohjaussuureena aktiivisuodatinmoduulille.

MCP-prototyypin testijärjestelmän toimintaketjun lohkokaavio on esitetty kuvassa 6.1.



Kuva 6.1. *MCP-prototyypin testijärjestelmän lohkokaavio*

Kokonaiskuva prototyypin testijärjestelyistä on esitetty kuvassa 6.2. Jännitemittauspiirikortti on kiinnitetty Merus Power:n A2-aktiivisuodatinmoduulin kytkentäpisteeseen. Kuormamoduulin virtamuuntajan ulostulo on kiinnitetty virtamittauspiirikorttiin. Valokuitupiirikortin kuitukaapeli puolestaan on liitetty A2-moduulin ohjauspiirikortin kuitusisäänmenoon.

Kuvasta 6.3 nähdään tarkemmin prototyypin moduulit. Erillisten virta- ja

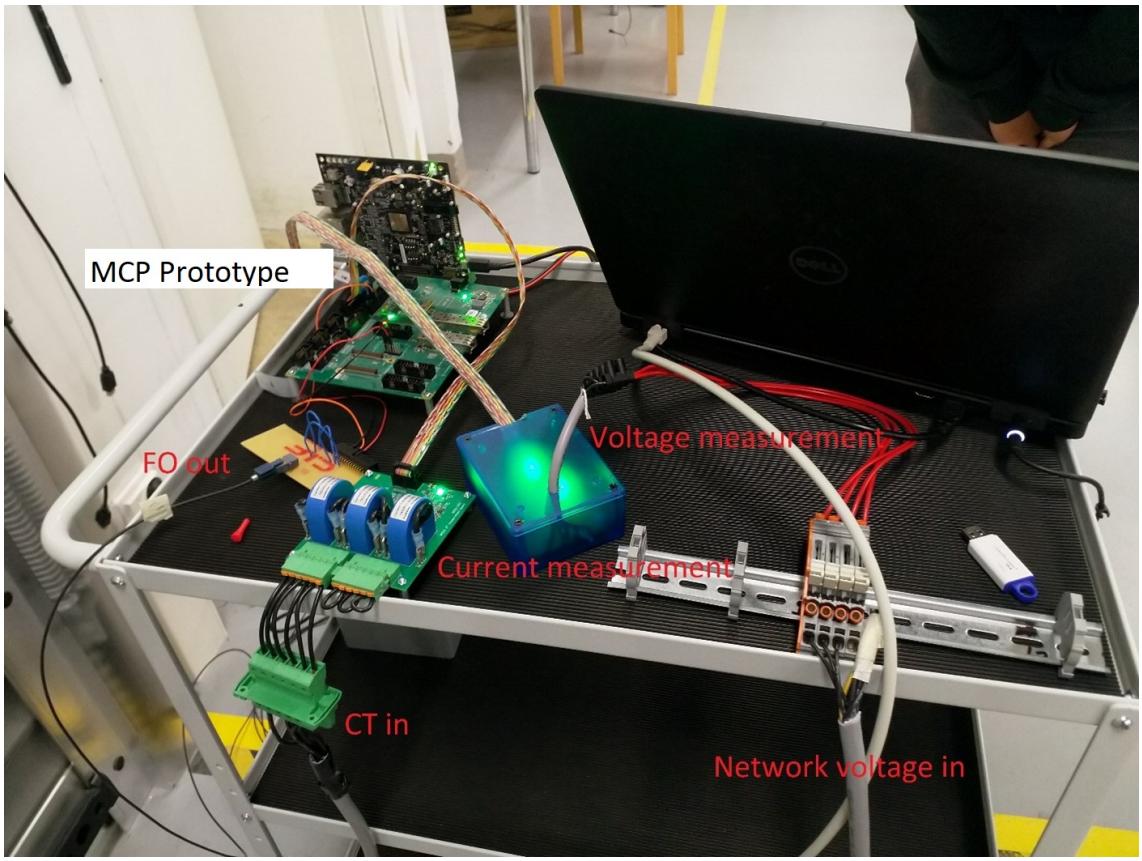
Kuva 6.2. *MCP-prototyypin testijärjestelmä testausmontussa*

jännitemittauspiirikorttien signaalit kytkettiin lattakaapeleilla prototyypin kokoajapiirikortin GPIO-liittimiin. Vastaavasti ulostuleva virtareferenssi liitettiin yksittäisestä GPIO-pinnistä valokuitupiirikorttiin.

6.1 Testitulokset

MCP-prototyypille tehtiin kaksi testiä. Ensin A2-aktiivisuodatinmoduulia käytettiin nimellisellä teholla ja todettiin, suoriutuuko MCP prototyyppi loisvirtareferenssin laskennasta ja lähettämisestä. Ohjausalgoritmin laskenta esiteltiin kappaleessa 2. Toisena testinä suoritettiin yksinkertainen askelkoe, jossa A2-moduulia käytettiin tyhjäkäynnillä ilman kuormaa ja todettiin MCP prototyypin toiminta, kun kuormamoduuli kytkettiin verkkoon. MCP prototyypin toiminta todettiin mittaamalla oskilloskoopilla verkkovirtaa ja verkon jännitettä yhdessä vaiheessa. Edellä mainittujen suureiden lisäksi mitattiin A2-moduulin tuottamaa kompensointivirtaa sekä verkkovirran vaiheiden virran summaa. Mitatut signaalit ovat nähtävillä taulukossa 6.1.

Skooppikuva verkon tilasta ennen testiä on esitetty kuvassa 6.4. Kuvasta nähdään, että kanavan 2 kuormavirran ja verkon jännitteen välinen vaihekulma on 90 astetta, eli kuormavirta on loisvirtaa. Testin lähtötilanteessa kompensointivirtaa ei vielä tuoteta, eli myös math-kanavan verkkovirta on loisvirtaa.



Kuva 6.3. *MCP-prototyypin moduulit*

Taulukko 6.1. *MCP-prototyyppitestin mittaussuureet*

Oskilloskoopin kanava	Mitattu signaali
CH1	Verkkojännite, L1
CH2	Kuormavirta, L1
CH3	Kompensointivirta, L1
Math	Verkkovirran summa

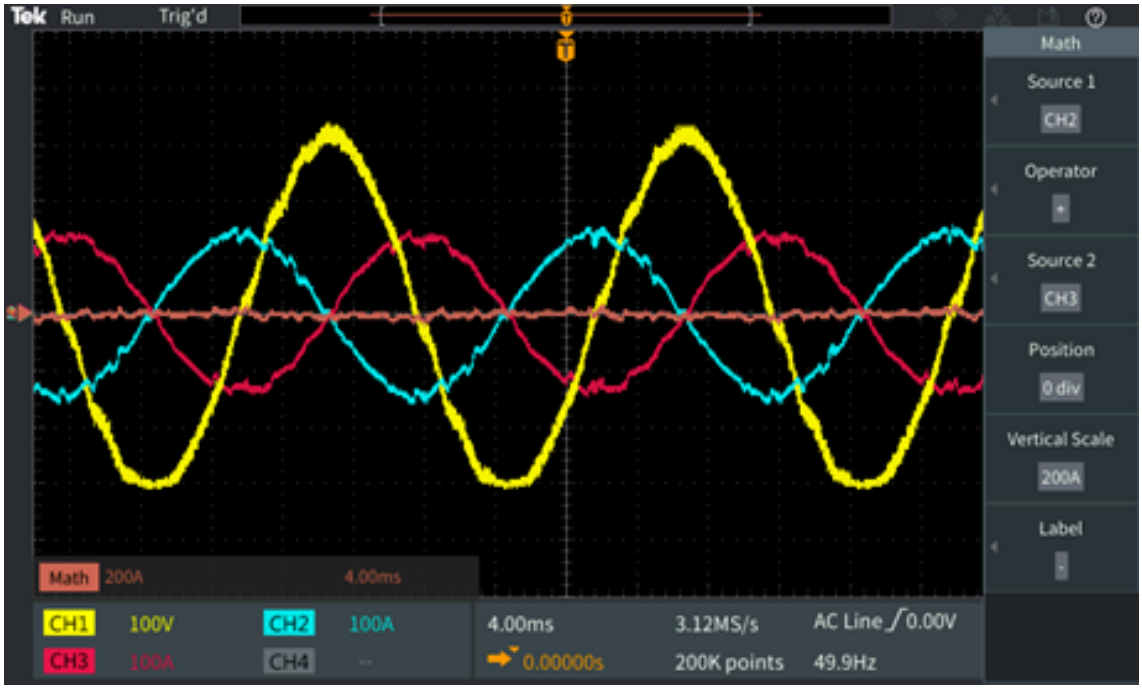
Kuvassa 6.5 kompensointi on kytketty päälle ja kompensointivirta on onnistuneesti kuormavirtaan verrattuna vastakkaisvaiheista loisvirtaa, jolloin verkon loisvirtarasitus poistuu.

Kuvassa 6.6 kuormavirta laitetaan päälle askelmaisesti. Testissä järjestelmän vasteaika on noin puoli verkkojaksoa, kunnes loisvirta on kokonaan kompensoitu.

6.2 Kommunikaation viiveet

Prototyypin toteutustavasta johtuen DSP suoritti sekä luku- että kirjoitusoperaatiot PCIe-väylän yli FPGA:n BRAMiin. Suurin osa merkittävistä operaatioista suoritettiin DSP:llä, joten myös viiveet selvitettiin DSP:n loggerin avulla. Itse logi ajoituksineen on esitetty kuvassa 6.7. Kuvasta nähdään, että selkeästi suurin osuus eli 66,6 % suoritusajasta kuluu memcopy-operaatioon, eli mittausdatan noutoon FPGA:n BRAM-lohkosta PCIe-väylän ylitse. Taulukossa 6.2 on eritelty DSP:n operaatioiden kesto nanosekunteina. Memcopy-

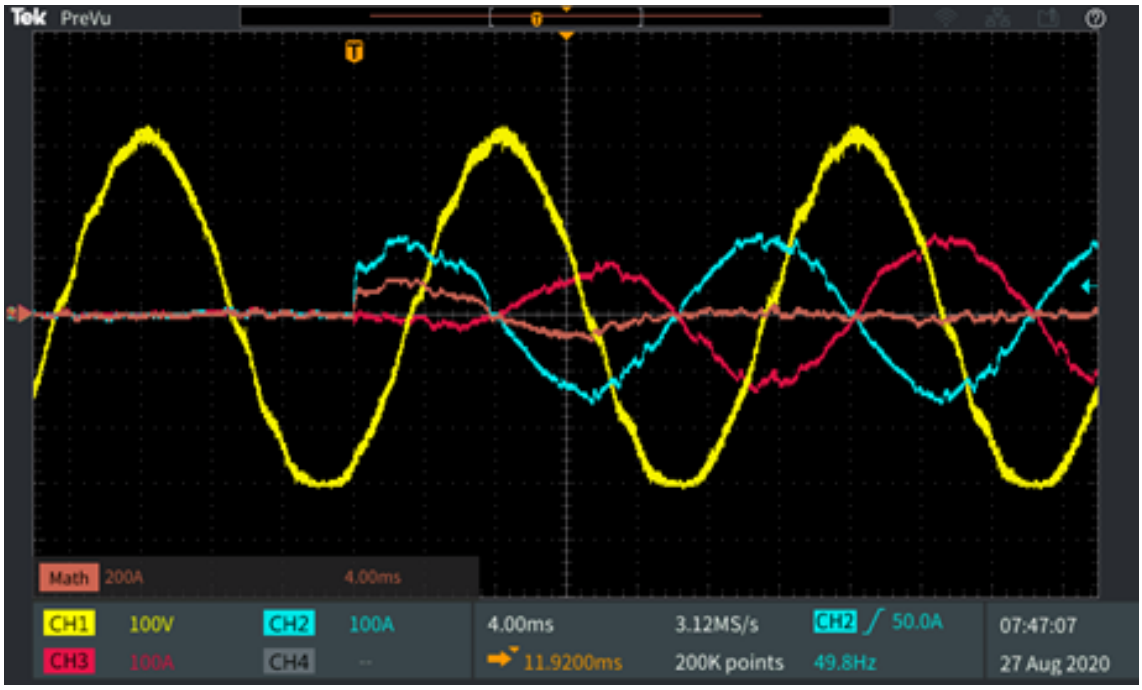
Kuva 6.4. *MCP-prototyyppitestin verkko ennen testiä*



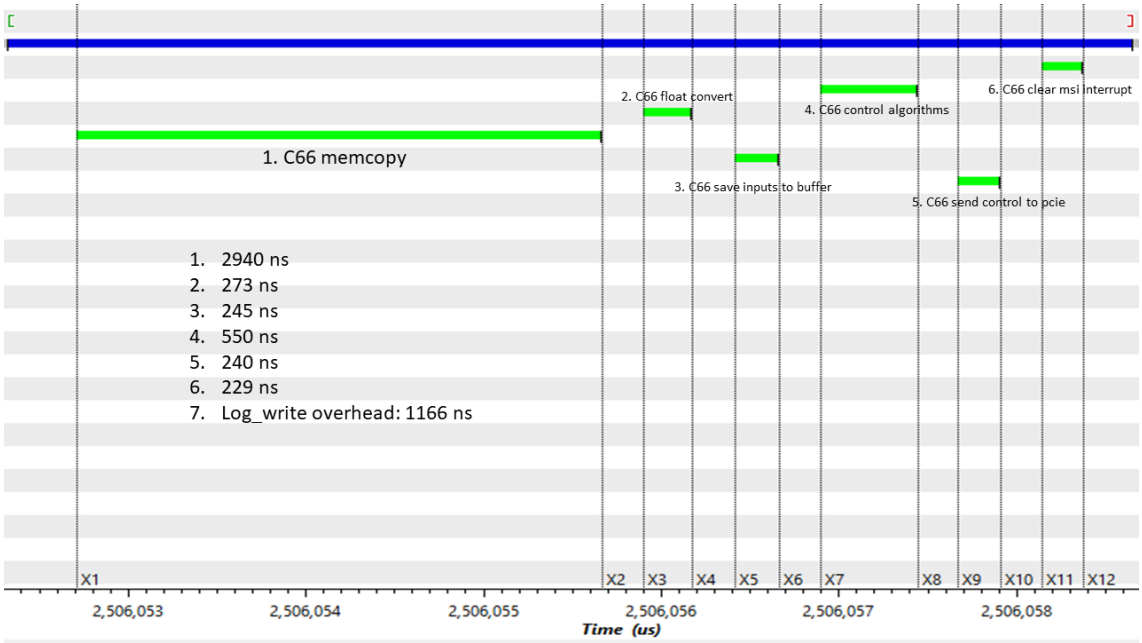
Kuva 6.5. *MCP-prototyyppitestin verkko A2-moduulin nimellisellä teholla*

opraatio kesti kokonaisuudessaan 2940 nanosekuntia.

PCIe:n Memory Read Request -lukuoperaatiot toimivat pyyntö-vastaus-periaatteella, eli DSP toimii PCIe:n termein Requester-roolissa ja FPGA Completer-roolissa. Requester lähettää TLP:n (Transaction Layer Packet), joka pyytää Completeria tekemään lukuoperaation. Completer lähettää takaisin TLP:n, joka sisältää dataa. Prototyypissä BRAMista luetaan 3x16 bittiä jännitteenmittausdataa sekä 3x16 bittiä virranmittausdataa



Kuva 6.6. *MCP-prototyyppitestin verkko askeltestin aikana*



Kuva 6.7. *DSP:n ajoitukset*

eli yhteensä 3x32 bittiä mittausdataa. Siirron luonteen takia DSP tekee siis kolme lukupyyntöä FPGA:n BRAMista siirtääkseen kolmivaiheisen jännite- ja virtamittausdatan säätöalgoritmin laskentaa varten. Yksi lukupyyntö on PCIe:n yhteydessä kolmen kaksinkertaisen sanan (Double Word) eli 3x32 bitin mittainen [37]. Completion TLP puolestaan on neljän kaksinkertaisen sanan eli 4x32 bitin mittainen, kun 3x32 bitin mittaisen otsikon (Header) päälle lisätään 32 bitin kokoinen varsinainen kuorma, joka sisältää BRAMista luetun mittausdatan. Eli kokonaisuudessaan kolmeen 32 bitin pakettiin ryhmitelty mittausdata voitiin siirtää FPGA:lta DSP:lle kolmessa Request

Taulukko 6.2. *DSP-funktioiden suoritusajat*

Nro.	tehtävä	kesto	yksikkö
1	memcopy	2940	ns
2	float conversion	273	ns
3	save inputs to buffer	245	ns
4	control algorithms	550	ns
5	send control to pcie	240	ns
6	MSI interrupt clear	229	ns
7	Logwrite overhead	1166	ns
8	operaatioiden kokonaiskesto	4477	ns
9	operaatiot - memcopy	1537	ns

Completion -lukuoperaatiossa. Tämä tarkoittaa 3x3x32 bitin Request TLP:tä ja 3x4x32 bitin Completion TLP:tä 3x32 bitin hyötykuormalla. Hyötykuorman osuus siirrettävästä datasta on tässä operaatiossa

$$3 * 32bit/(3 * 3 * 32bit + 3 * 4 * 32bit) * 100\% = 14,3\% \tag{6.1}$$

Taulukosta 6.2 voidaan huomata, että säätöalgoritmin referenssien siirto-operaatio *send control to pcie* kesti vain 240 nanosekuntia. Siirtoon kulunut aika on siis huomattavasti nopeampi kuin mittausdatan lukuoperaatio. Säätöalgoritmin referenssien hyötykuorma on toisaalta vain 2x16 bittiä eli yhteensä 32 bittiä, joka voidaan kirjoittaa FPGA:lle yhdellä Memory Write -tyyppisellä TLP:llä. Näin ollen FPGA:lle on sovittu ennalta määrätty alue BRAMin osoitetilasta, mihin DSP kirjoittaa datan välttäen näin Memory Read Requestin request completion -tyyppiset operaatiot. Nyt koko kirjoitusoperaatio voidaan hoitaa 32-bittisillä muistiosoitteilla 4x32 bitin pituisen TLP:n avulla. Hyötykuorman osuus siirrettävästä datasta on

$$1x32bit/4 * 32bit * 100\% = 25\% \tag{6.2}$$

Operaatiot eivät ole tietenkään kovin vertailukelpoisia keskenään, mutta nyt syy operaatioiden hyvin eripituiselle kestolle on selkeä.

Otetaan seuraavaksi vielä esimerkki siitä, mikäli FPGA kirjoittaisi mittausdatan suoraan DSP:lle käyttäen Memory Write -tyyppisiä TLP-paketteja, siirron hyötykuorma nousisi 14,3 %:sta 25 %:iin. Jos oletetaan, että myös DSP:n suuntaan siirto tapahtuu yhtä nopeasti kuin FPGA:n suuntaan, koko mittausdata saataisiin siirrettyä DSP:lle noin 3x240 nanosekunnissa eli yhteensä 720 nanosekunnissa, mikä on 24,5 % alkuperäisestä Memory Read Request -kestosta.

Prototyypin tavoitteisiin nähden nykyinen ratkaisu oli erittäin riittävä, sillä DSP:n operaatioihin kului kokonaisuudessaan aikaa 4.48µs, mikä on mittakorttien adc-piirien

ajoitusvaatimuksiin verrattuna melko lyhyt aika. ADS8866-piirien daisy chain -moodin ajoitusvaatimukset on listattu taulukossa 5.3. ADS8866:n näytteistystaajuus on rajoitettu minimissään yhteen muunnokseen per 10µs, mikä on jo yksinään 20 % koko ohjaussyklin suoritusajasta.

7 YHTEENVETO

Tehoelektroniikkajärjestelmää ohjaava ja valvova digitaalinen säädin ohjaa säätöparametrien arvoja vertaamalla systeemin ulostuloja referenssiarvoihin, mikä minimoi erosuureen. Digitaalinen säädin mahdollistaa toimintojen muokkauksen ja päivittämisen laitteiston asentamisen jälkeen, jolloin järjestelmän ylläpito on vaivattomampaa. Mikroprosessorien käyttö mahdollistaa ohjausalgoritmien toteuttamisen reaaliajassa. Säätimen toiminnot voitiin jakaa järjestelmäohjaukseen, johon kuuluu ohjausalgoritmien, kuten tehomuuntimen säätöreferenssien tuottaminen, ja järjestelmäparametrien säätö. Tehomuuntimen ohjaus sisältää nopean ja aikakriittisen säädön, joka toteutetaan laitteen kytkentätaajuudella, joka voi olla esimerkiksi 20 kHz. Mikroprosessorin tukena voidaan käyttää Field-programmable gate array -piiriä, eli FPGA-piiriä, joka toimii tehokkaasti nopeissa ja aikakriittisissä ohjaus- ja suojaustehtävissä sekä rinnakkaisuutta vaativissa toiminnallisuuksissa. Parempaa laskentatehoa vaativat tehtävät, kuten säätöalgoritmit ja signaalinkäsittely, toteutetaan usein digitaalisella signaaliprosessorilla.

Työssä toteutettiin Merus Power Dynamics Oy:lle tehoelektroniikkajärjestelmän pääsäädinprototyypin FPGA-piirin toiminnallisuus. Prototyypin ytimenä toimi FPGA- ja DSP-piirien välisen kommunikaation toteuttava PCI Express -väylä, jonka soveltuvuutta pääsäätimen käyttötarkoitukseen arvioitiin. Ensin työssä esiteltiin perinteisiä tiedonsiirtomenetelmiä sekä muutama suosittu väyläprotokolla. Tämän jälkeen käytiin läpi Serializer/Deserializer-menetelmiä sekä linjakoodausmenetelmiä. Viimeisenä teoriaosuudessa käsiteltiin korkean suorituskyvyn sarjamuotoisia väyliä, kuten PCI-Express -väylää, Aurora-protokollaa sekä Serial RapidIO:ta. Lopuksi työssä esiteltiin pääsäädinprototyypin FPGA-toteutus, minkä jälkeen käytiin läpi prototyypin verifiointi, testitulokset ja kommunikaatioviiveet.

Itse verifiointivaiheessa prototyypin toimintaa testattiin aktiivisuodatinmoduulin loisvirrankompensointikäytössä. Testijärjestelmässä prototyyppi tuotti ulkoisen virtareferenssin aktiivisuodatinmoduulille. Ensimmäisessä testissä todettiin prototyypin toiminta loisvirtareferenssin toteuttamisessa, kun aktiivisuodatinmoduulia käytettiin nimellisellä teholla. Toisessa testissä aktiivisuodatinta käytettiin tyhjäkäynnillä, ja todettiin prototyypin toiminta, kun kuormamoduuli kytkettiin verkkoon. Ensimmäisessä testissä prototyyppi onnistui tuottamaan loisvirtareferenssin mittausdatan perusteella, jolloin verkon loisvirtarasitus saatiin onnistuneesti poistettua. Askeltestissä järjestelmän vasteaika oli noin 10 millisekuntia, eli prototyyppi suoriutui hyvin toisestakin testistä.

DSP-funktioiden loggaus paljasti, että suoritusajasta 66,6 % kului mittausdatan noutoon FPGA:n BRAM-lohkosta. Tehoton DSP:n käyttö johtui haastavan PCIe-väylän epäoptimaalisesta käytöstä. 3x32 bitin hyötykuorman siirtäminen Memory Read Request -operaatiolla 32-bittisellä osoitteistuksella tuottaa hyötykuorman osuudeksi siirrettävässä datamäärässä vain 14,3 %. Prototyypin toiminnan verifiointiin PCIe-väylän toiminta oli silti erittäin riittävä, sillä kokonaisuudessaan aikaa kului DSP:n operaatioihin yhteensä 4.48µs. Suurin rajoittava tekijä prototyypin toiminnassa oli ADS8866-piirin daisy chain -moodin näytteistystaajuus, joka on rajoitettu yhteen muutokseen per 10µs, mikä on 20 % ohjaussyklin suoritusajasta.

Järeähkö PCI Express saattaa vaikuttaa toisaalta varsin ylimitoitetulta ratkaisulta työssä esitellyn mittausdatan siirtoon. Eivätkä suoritetut testit ja viiveiden mittaukset välttämättä antaneet hyvää kokonaiskuvaa siitä mihin prototyyppitestauksella pyrittiin. Mittausdataa olisi esimerkiksi voinut generoida enemmän, jotta suoritetut testit olisivat tukeneet paremmin jatkokehitysprojektia. Yksi jatkokehityksen tavoitteista on saavuttaa mahdollisimman pieni viive datansiirrolle, mikä puolestaan mahdollistaa pienen säätöviiveen. Näin varaudutaan kytkentätaajuuksien nousuun esimerkiksi SiC puolijohteiden kanssa.

Jatkokehitystä ajatellen PCIe-kommunikaation kehittäminen on kriittinen osa-alue, jotta DSP:n laskenta-aika voidaan hyödyntää järkevämmissä toiminnoissa. Mittausdata tulee siis voida kirjoittaa Memory Write -operaatiolla PCIe:n yli suoraan DSP:lle. Myös virta- ja jännitemittaus tulee toteuttaa tulevaisuudessa tehokkaammin. Näytteistys prototyypin ADC-piireillä tapahtuu turhan hitaasti daisy chain -moodissa, joten jatkossa kolmivaiheiset virta- ja jännitemittaukset on hyvä siirtää suoraan rinnakkaismuotoisena omiin FPGA-pinneihin. Myös FPGA-piirin kokoa on syytä harkita uudelleen, sillä prototyypissä on käytössä XC7A50T-laitteella varustettu keskikastin Artix-7 piiri. On todennäköistä, että järjestelmätason säädintä halutaan jollakin aikajänteellä myös laajentaa lopullisen suunnittelun jälkeenkin.

LÄHTEET

[1] Sozański, K. *Digital Signal Processing in Power Electronics Control Circuits*. Springer London, 2017.

[2] Rong, Y., Wang, J., Shen, Z., Burgos, R., Boroyevich, D. ja Zhou, S. Distributed Control and Communication System for PEBB-based Modular Power Converters. *2019 IEEE Electric Ship Technologies Symposium (ESTS)*. 2019, 627–633. DOI: 10.1109/ESTS.2019.8847807.

[3] Al-Haddad, K. *Power electronics for renewable energy systems, transportation, and industrial applications*. eng. Chichester, West Sussex, United Kingdom ;

[4] Simes M. G. Farret, F. A. *Modeling Power Electronics and Interfacing Energy Conversion System*. Wiley-IEEE Press, 2017.

[5] Naouar, M.-W., Monmasson, E., Naassani, A., Slama-Belkhodja, I. ja Patin, N. FPGA-Based Current Controllers for AC Machine Drives-A Review. eng. *IEEE transactions on industrial electronics (1982)* 54.4 (2007), 1907–1925. ISSN: 0278-0046.

[6] Deschamps, J.-P., Bioul, G. J. A. ja Sutter, G. D. *Synthesis of arithmetic circuits: FPGA, ASIC and embedded systems*. eng. Hoboken: WILEY, 2006. ISBN: 0471741418.

[7] Adan, A. O., Tanaka, D., Burgyan, L. ja Kakizaki, Y. The Current Status and Trends of 1,200-V Commercial Silicon-Carbide MOSFETs: Deep Physical Analysis of Power Transistors From a Designer's Perspective. *IEEE Power Electronics Magazine* 6.2 (2019), 36–47. DOI: 10.1109/MPEL.2019.2909592.

[8] Sasagawa, M., Nakamura, T., Inoue, H. ja Funaki, T. A study on the high frequency operation of DC-DC converter with SiC DMOSFET. *The 2010 International Power Electronics Conference - ECCE ASIA -*. 2010, 1946–1949. DOI: 10.1109/IPEC.2010.5542089.

[9] Davis, S. *SiC Transistor Basics: FAQs*. 2013. URL: https://www.powerelectronics.com/technologies/discrete-power-semis/article/21860696/sic-transistor-basics-faqs (viitattu 10. 12. 2020).

[10] Vaidya, D., Mukherjee, S., Zagrodnik, M. A. ja Wang, P. *A review of communication protocols and topologies for power converters*. 2017, 2233–2238.

[11] I., M. Power Electronics System Communications. 1999. URL: http://hdl.handle.net/10919/31218.

[12] Toh, C. L. ja Norum, L. E. A performance analysis of three potential control network for monitoring and control in Power Electronics converter. *2012 IEEE International Conference on Industrial Technology*. 2012, 224–229. DOI: 10.1109/ICIT.2012.6209942.

[13] Singh, B. *Power quality problems and mitigation techniques*. eng. Chichester, England: Wiley. ISBN: 1-118-92206-9.

[14] Fuchs, E. F. *Power quality in power systems and electrical machines*. eng. Amsterdam ;

[15] *Control of power electronic converters and systems. Volume 2*. eng. London, United Kingdom: Academic Press. ISBN: 978-0-12-816168-5.

[16] Frenzel, L. E. *Handbook of serial communications interfaces : a comprehensive compendium of serial digital input/output (I/O) standards*. Elsevier Science, 2016.

[17] Abbott, D. *PCI Bus Demystified, 2nd Edition*. 2. painos. 2004.

[18] *Embedded hardware*. eng. Amsterdam ; 2008.

[19] L., F. *Serial I/O Interfaces Dominate Data Communications*. 2015. URL: https://www.electronicdesign.com/technologies/communications/article/21800967/serial-io-interfaces-dominate-data-communications (viitattu 17. 09. 2020).

[20] Rockrohr, J. D., Dramstad, K., Mohammad, A., Ogilvie, C. R. ja Sorna, M. A. *High Speed Serdes Devices and Applications*. eng. 1. Aufl. New York, NY: Springer-Verlag, 2009. ISBN: 9780387798332.

[21] Chu, P. P. *RTL hardware design using VHDL : coding for efficiency, portability, and scalability*. eng. Hoboken, New Jersey: Wiley-Interscience, 2006. ISBN: 1-280-44810-5.

[22] Instruments, T. *Design and Usage Guidlines for the C2000 External Memory Interface (EMIF)*. 2017. URL: https://www.ti.com/lit/an/sprac96a/sprac96a.pdf (viitattu 04. 01. 2021).

[23] Instruments, T. *KeyStone Architecture Universal Parallel Port (uPP) - User Guide*. 2012. URL: https://www.ti.com/lit/ug/spruhg9/spruhg9.pdf (viitattu 11. 01. 2021).

[24] Gay, W. *Advanced Raspberry Pi Raspbian Linux and GPIO Integration*. eng. Berkeley, CA, 2018.

[25] Ibe, O. C. (C. *Fundamentals of data communication networks*. Wiley, 2018.

[26] Dhaker, P. *Introduction to SPI Interface*. 2018. URL: https://www.analog.com/media/en/analog-dialogue/volume-52/number-3/introduction-to-spi-interface.pdf (viitattu 27. 11. 2020).

[27] Gay, W. I2C. eng. *Beginning STM32*. Berkeley, CA: Apress, 2018, 195–221. ISBN: 1484236238.

[28] Semiconductors, N. *I2C-bus specification and user manual*. 2014. URL: https://www.nxp.com/docs/en/user-guide/UM10204.pdf (viitattu 27. 11. 2020).

[29] Altera. *The Evolution of High-Speed Transceiver Technology*. 2002. URL: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/wp/wp_hs_transceiver.pdf (viitattu 22. 10. 2020).

[30] solutions, C. P. *Exploring the Difference Between Single-Ended and Differential Signals*. 2021. URL: https://resources.pcb.cadence.com/blog/msa2021-exploring-the-

difference-between-single-ended-and-differential-signals (viitattu 05. 11. 2020).

[31] C., P. *The Why and How of Differential Signalin*. 2016. URL: https://www.allaboutcircuits.com/technical-articles/the-why-and-how-of-differential-signaling/ (viitattu 20. 10. 2020).

[32] Athavale A., C. C. *High-Speed Serial I/O Made Simple*. Xilinx Connectivity Solutions, 2005. URL: https://www.xilinx.com/publications/archives/books/serialio.pdf (viitattu 15. 10. 2020).

[33] Instruments, N. *High-Speed Serial Explained*. 2016. URL: ftp://ftp.ni.com/evaluation/HighSpeedSerial_WP_Final.pdf (viitattu 11. 11. 2020).

[34] Xilinx. *Aurora 64b/66b Protocol Specification*. 2014. URL: https://www.xilinx.com/support/documentation/ip_documentation/aurora_64b66b_protocol_spec_sp011.pdf (viitattu 16. 10. 2021).

[35] Electronic, U. *What is SERDES?* 2021. URL: https://www.utmel.com/blog/categories/integrated%20circuit/what-is-serdes (viitattu 13. 01. 2022).

[36] D., L. *SerDes Architectures and Applications*. 2004. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.173.4371&rep=rep1&type=pdf (viitattu 21. 10. 2020).

[37] Jackson M., B. R. *PCI Express Technology 3.0*. First Edition. MindShare Press, 2012.

[38] Xilinx. *Aurora 8b/10b Protocol Specification*. 2014. URL: https://www.xilinx.com/support/documentation/ip_documentation/aurora_8b10b_protocol_spec_sp002.pdf (viitattu 16. 11. 2020).

[39] Fuller, S. H. *RapidIO the embedded system interconnect*. eng. Chichester, England ; 2005.

[40] Oy, M. P. D. *Merus Power active harmonic filters - A2 series*. 2019. URL: https://www.meruspower.fi/wp-content/uploads/2020/11/Merus-A2-series_EN_DIGI.pdf (viitattu 16. 10. 2021).

[41] F., Z. eng. Springer, 2020. ISBN: 978-981-15-1870-6.

[42] PCI-SIG. *PCI-SIG*. 2021. URL: https://pcisig.com/specifications/ (viitattu 26. 10. 2021).

[43] Xilinx. *7 Series FPGAs Data Sheet: Overview*. 2021. URL: https://www.xilinx.com/support/documentation/ip_documentation/pcie_7x/v3_0/pg054-7series-pcie.pdf (viitattu 26. 10. 2021).

[44] Electronic, T. *TE0714 Technical Reference Manual*. 2019. URL: https://www.mouser.com/datasheet/2/989/TRM-TE0714-02-1628529.pdf (viitattu 20. 11. 2020).

[45] Xilinx. *7 Series FPGAs Data Sheet: Overview*. 2020. URL: https://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf (viitattu 03. 11. 2020).

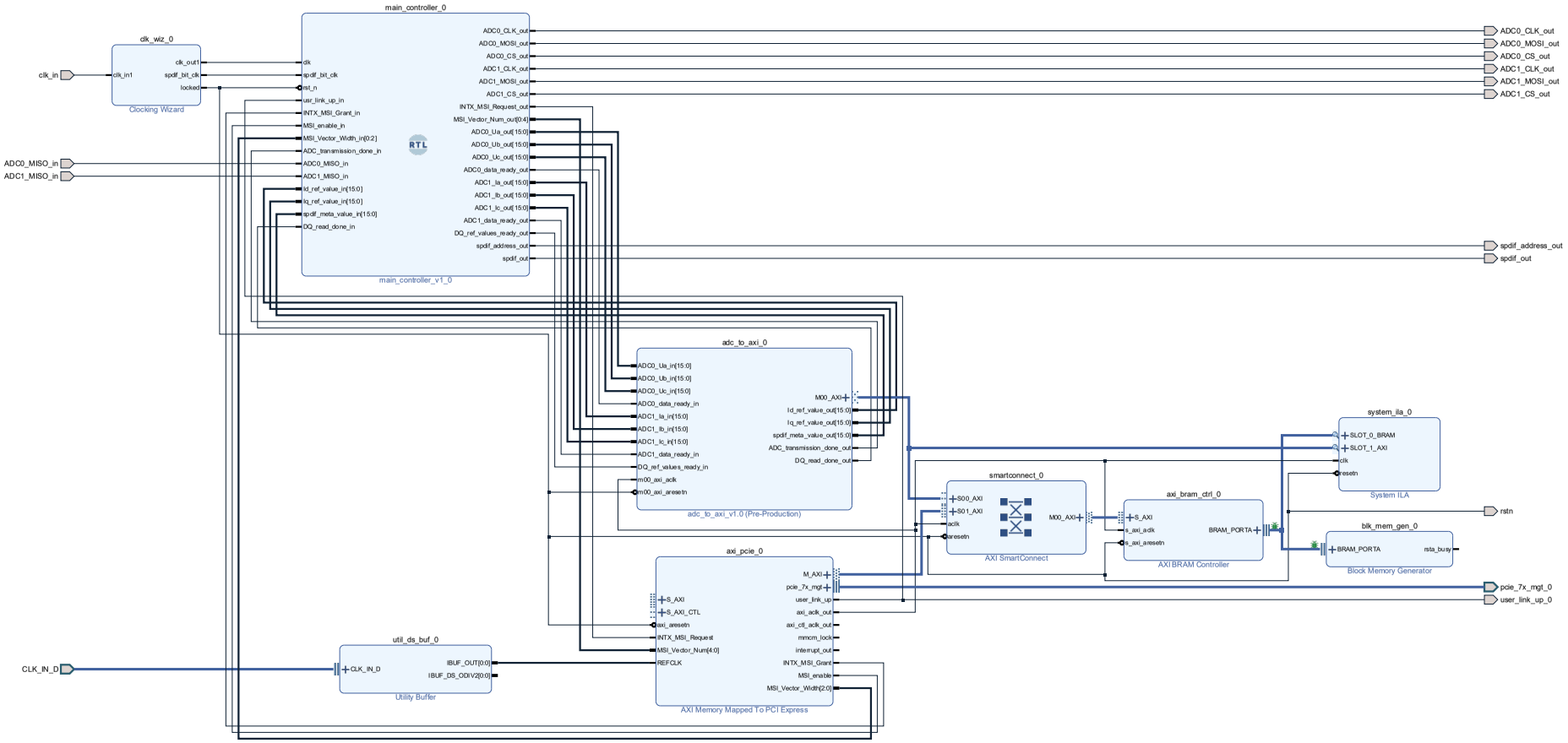
[46] Instruments, T. *ADS8866 Datasheet*. 2019. URL: https://www.ti.com/lit/ds/symlink/ads8866.pdf?ts=1605880617116 (viitattu 20. 11. 2020).

[47] Xilinx. *Designing with IP*. 2020. URL: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2020_1/ug896-vivado-ip.pdf (viitattu 25. 11. 2020).

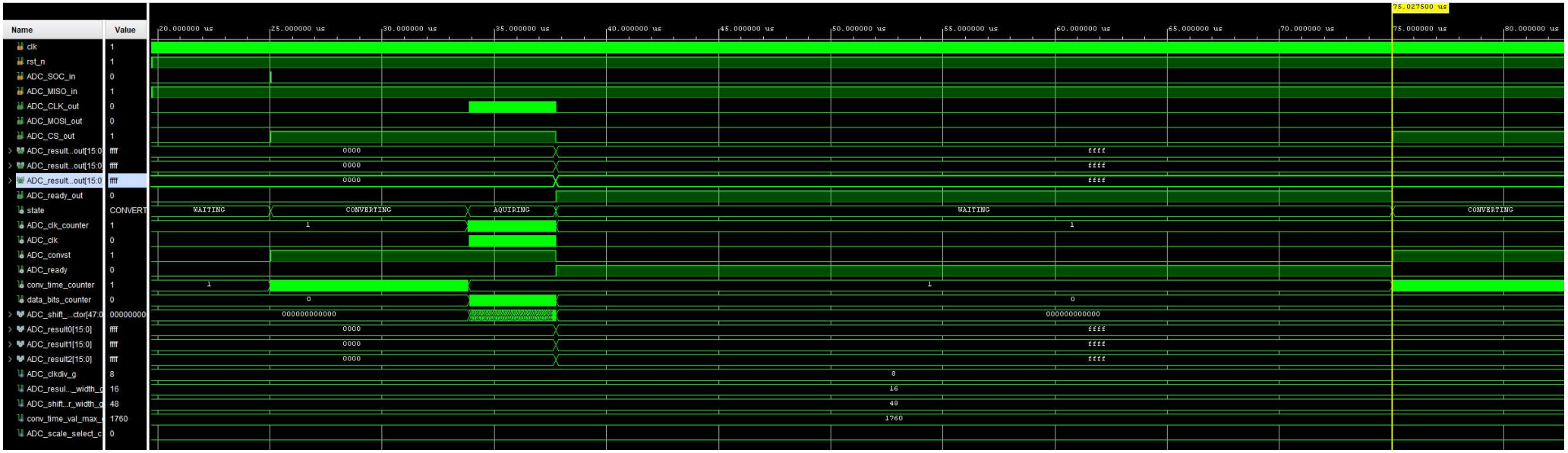
[48] Xilinx. *AXI Memory Mapped To PCIe Gen2 v2.8*. 2020. URL: https://www.xilinx.com/support/documentation/ip_documentation/axi_pcie/v2_8/pg055-axi-bridge-pcie.pdf (viitattu 25. 11. 2020).

[49] arm. *AMBA AXI and ACE Protocol Specification*. 2020. URL: https://developer.arm.com/documentation/ihi0022/latest.

A VIVADO-LOHKOKAAVIO


clk_in
clk_wiz_0
clk_in1
clk_out1
spdif_bit_clk
locked
Clocking Wizard
main_controller_0
clk
spdif_bit_clk
rst_n
usr_link_up_in
INTX_MSI_Grant_in
MSI_enable_in
MSI_Vector_Width_in[0:2]
ADC_transmission_done_in
ADC0_MISO_in
ADC1_MISO_in
Id_ref_value_in[15:0]
Iq_ref_value_in[15:0]
spdif_meta_value_in[15:0]
DQ_read_done_in
RTL
ADC0_CLK_out
ADC0_MOSI_out
ADC0_CS_out
ADC1_CLK_out
ADC1_MOSI_out
ADC1_CS_out
INTX_MSI_Request_out
MSI_Vector_Num_out[0:4]
ADC0_Ua_out[15:0]
ADC0_Ub_out[15:0]
ADC0_Uc_out[15:0]
ADC0_data_ready_out
ADC1_Ia_out[15:0]
ADC1_Ib_out[15:0]
ADC1_Ic_out[15:0]
ADC1_data_ready_out
DQ_ref_values_ready_out
spdif_address_out
spdif_out
main_controller_v1_0
ADC0_MISO_in
ADC1_MISO_in
ADC0_CLK_out
ADC0_MOSI_out
ADC0_CS_out
ADC1_CLK_out
ADC1_MOSI_out
ADC1_CS_out
spdif_address_out
spdif_out
adc_to_axi_0
ADC0_Ua_in[15:0]
ADC0_Ub_in[15:0]
ADC0_Uc_in[15:0]
ADC0_data_ready_in
ADC1_Ia_in[15:0]
ADC1_Ib_in[15:0]
ADC1_Ic_in[15:0]
ADC1_data_ready_in
DQ_ref_values_ready_in
m00_axi_aclk
m00_axi_aresetn
M00_AXI
Id_ref_value_out[15:0]
Iq_ref_value_out[15:0]
spdif_meta_value_out[15:0]
ADC_transmission_done_out
DQ_read_done_out
adc_to_axi_v1.0 (Pre-Production)
smartconnect_0
S00_AXI
S01_AXI
aclk
aresetn
M00_AXI
AXI SmartConnect
axi_bram_ctrl_0
S_AXI
s_axi_aclk
s_axi_aresetn
BRAM_PORTA
AXI BRAM Controller
system_ila_0
SLOT_0_BRAM
SLOT_1_AXI
clk
resetn
System ILA
rstn
blk_mem_gen_0
BRAM_PORTA
rsta_busy
Block Memory Generator
axi_pcie_0
S_AXI
S_AXI_CTL
axi_aresetn
INTX_MSI_Request
MSI_Vector_Num[4:0]
REFCLK
M_AXI
pcie_7x_mgt
user_link_up
axi_aclk_out
axi_ctl_aclk_out
mmcm_lock
interrupt_out
INTX_MSI_Grant
MSI_enable
MSI_Vector_Width[2:0]
AXI Memory Mapped To PCI Express
pcie_7x_mgt_0
user_link_up_0
CLK_IN_D
util_ds_buf_0
CLK_IN_D
IBUF_OUT[0:0]
IBUF_DS_ODIV2[0:0]
Utility Buffer

B ADC-AJURIN LOGIIKKASIMULOINTI


Name
Value
clk
rst_n
ADC_SOC_in
ADC_MISO_in
ADC_CLK_out
ADC_MOSI_out
ADC_CS_out
ADC_result...out[15:0]
ADC_result...out[15:0]
ADC_result...out[15:0]
ADC_ready_out
state
ADC_clk_counter
ADC_clk
ADC_convst
ADC_ready
conv_time_counter
data_bits_counter
ADC_shift_...ctor[47:0]
ADC_result0[15:0]
ADC_result1[15:0]
ADC_result2[15:0]
ADC_clkdiv_g
ADC_resul..._width_g
ADC_shift...r_width_g
conv_time_val_max_i
ADC_scale_select_c
20.000000 us
25.000000 us
30.000000 us
35.000000 us
40.000000 us
45.000000 us
50.000000 us
55.000000 us
60.000000 us
65.000000 us
70.000000 us
75.000000 us
75.027500 us
80.000000 us
WAITING
CONVERTING
AQUIRING
WAITING
CONVERTING
0000
ffff
000000000000
8
16
48
1760