

Mikael Arvaja

**MALLINTAMINEN AJURIKEHITYKSEN  
APUNA JÄRJESTELMÄPIIRIN  
VALMISKOMPONENTTITASOLLA**

Informaatioteknologia ja viestintä  
Kandidaatintyö  
Huhtikuu 2020

# TIIVISTELMÄ

Mikael Arvaja : Mallintaminen ajurikehityksen apuna järjestelmäpiirin valmiskomponenttitasolla  
Kandidaatintyö  
Tampereen yliopisto  
Ohjelmistotekniikka  
Huhtikuu 2020

---

Järjestelmäpiirin mallintaminen on toimintatapa, jolla pyritään nopeuttamaan piirin tuotantoprosessia. Ajuri on järjestelmäpiirin ohjelmiston matalin taso, joka yhdistää fyysisen laitteiston ja ohjelmiston toisiinsa. Tässä työssä tutustutaan järjestelmäpiirin valmiskomponenttikohtaisten ajurien kehitysprosesseihin ja tutkitaan, minkälaisia vaikutuksia mallinnuksella on ajurien kehittämiseen. Tätä tutkittiin haastatteleamalla järjestelmäpiirikehittäjiä ja vertailemalla kahden eri järjestelmäpiirituotteen tuotantoprosesseja, joista toisessa oli mukana mallinnus ja toisessa ei.

Työn alussa kerrotaan järjestelmäpiirin tuotantoprosessista sekä siitä, miten prosessin pitkä kesto tuo monia taloudellisia sekä teknisiä vaikeuksia. Näiden vaikeuksien taklaamiseksi ehdotetaan järjestelmäpiirin mallinnusta sekä valmiskomponenttien uudelleenkäyttöä. Laitteiston mallintaminen mahdollistaa korkean tason ohjelmiston tekemisen aloittamisen ennen kuin varsinainen laitteisto on saatavilla. Mallinnus myös edistää laitteistokehitystä. Komponenttien ja niiden ajurien uudelleenkäyttö taas vähentää turhaa työtä, ja näin puolestaan edistää koko tuotantoprosessia.

Tutkimuksessa mallinnuksen vaikutuksia ajurikehitykseen tarkastellaan ajurien valmistusajan, ohjelmoinnin, testauksen ja uudelleenkäytön näkökulmista. Keskeisimmät positiiviset vaikutukset liittyvät ajurikehittäjien saamaan osaamiseen heidän ollessaan mukana mallinnustyössä. Mallintaessaan laitteistoa kehittäjät oppivat sen toimintaa ja osaavat tästä syystä tehdä valmiskomponenteille parempia ajureita sekä paikantaa koko systeemin testiajoissa löytyneet viat paremmin. Negatiiviset vaikutukset liittyvät mallinnustyön vaatimaan resurssien varaamiseen sekä mallin mahdolliseen hyödyttömyyteen ajurikehityksen näkökulmasta. Mallinnuksella huomattiin olevan myös paljon potentiaalisia mahdollisuuksia, jotka tässä työssä tutkitun tuotteen kohdalla eivät vielä olleet realismia. Tällaisia mahdollisuuksia ovat esimerkiksi ajurien testaaminen mallilla emulaattorin sijaan ja vianselvitys mallin välitulosten avulla.

Avainsanat: Järjestelmäpiiri, valmiskomponentti, ajuri, mallinnus

# SISÄLLYSLUETTELO

1. JOHDANTO .....	1
2. JÄRJESTELMÄPIIRIN KEHITYSPROSESSI .....	3
2.1 Valmiskomponentit.....	4
2.2 Valmiskomponenttien ajurit .....	5
2.2.1 Ajurien kehitysprosessi .....	6
2.2.2 Ajurien testaaminen .....	6
2.3 Järjestelmäpiirin mallinnus .....	7
3. MALLINNUKSEN VAIKUTUKSIEN SELVITYS .....	9
3.1 Mallinnuksen vaikutukset ajurien valmistusaikaan.....	9
3.2 Mallinnuksen vaikutukset ajurien ohjelmointiin .....	10
3.3 Mallinnuksen vaikutukset ajurien testaukseen.....	12
3.4 Mallinnuksen vaikutukset uudelleenkäytettävyyteen.....	14
4. YHTEENVETO.....	15
LÄHTEET .....	17

## LYHENTEET JA MERKINNÄT

ASIC	<i>Application specific integrated circuit</i> , piiri, joka on suunniteltu johonkin tiettyyn käyttötarkoitukseen
FPGA	<i>Field-programmable gate array</i> , piiri, jonka sisältämä logiikka voidaan ohjelmoida uudelleen
IP	<i>Intellectual property</i> , valmiskomponentti; järjestelmän osa, joka suorittaa tietyn tehtävän, ja jota voidaan käyttää uudelleen
SoC	<i>System-on-Chip</i> , järjestelmäpiiri

# 1. JOHDANTO

SoCien (System-on-Chip) eli järjestelmäpiirien tuotannossa on vaikeaa pysyä teknologian kehityksen ja muiden valmistajien kanssa kilpailussa mukana. ASIC-järjestelmäpiirien (Application specific integrated circuit) kehityksessä on aina ollut suurena ongelmana niiden pitkä markkinoille saattamiseen tarvittava aika. Kehitysprosessia on pitkittänyt erityisesti laitteiston ja ohjelmiston samanaikaisen kehittämistyön puute. Vasta valmiin fyysisen laitteiston päälle on voitu alkaa rakentaa ohjelmistoa. Yksittäisen ASIC-järjestelmäpiirituotteen kehittämisen aloittamisesta tuotteen markkinoille saamiseen voi mennä useita vuosia. [1]

Tämän tuotantoprosessin nopeuttamiseksi tarvitaan tehokkaita toimintatapoja. Yksi tehokas tapa nopeuttaa järjestelmäpiirin tuotantoprosessia on mahdollistaa sen ohjelmiston rakentaminen jo ennen sen laitteiston valmistumista. Järjestelmäpiirien mallintaminen on tuotantomenetelmä, jolla pyritään juuri tähän. Toinen merkittävä tapa lyhentää tuotantoaikaa on valmiiksi tehdyn työn uudelleenkäyttö. Mahdollisimman tehokas uudelleenkäytettävien IP-lohkojen (Intellectual property) eli valmiskomponenttien hyödyntäminen tekee piirien rakentamisesta vaivattomampaa ja vähentää turhaa työtä.

Jotta fyysisen valmiskomponenttitason saa yhdistettyä järjestelmäpiiriä käsittelevään ohjelmaan, tarvitaan näiden väliin ajuriohjelmat. Ajuri on ohjelmiston hierarkian matalin taso, jonka tehtävä on mahdollistaa laitteiston käsitteleminen ohjelmistolla. Piirin ohjelmisto rakennetaan siis ajurien päälle. Se, miten ja mitä ohjelmisto pystyy laitteistolla tekemään, riippuu suurelta osin ajureista.

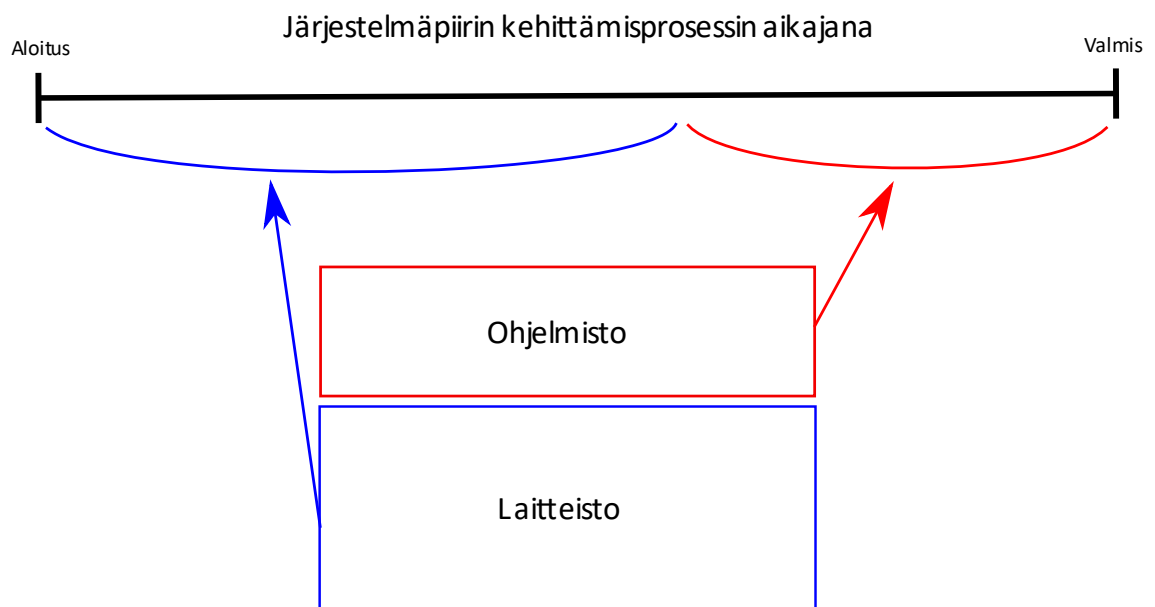
Tässä työssä tutkitaan sitä, miten valmiskomponenttitason mallintaminen vaikuttaa ajurien kehittämiseen. Työssä tutustutaan järjestelmäpiirin ja sen ajurien kehitysprosessiin ja tutkitaan, mitä hyötyjä tai haittoja ajurien kehittämiseen järjestelmäpiirin mallintamisella on. Tätä tutkitaan tutustumalla kahden eri tuotteen kehitysprosesseihin sekä haastatteleamalla järjestelmäpiirikehittäjiä, joilla on kokemusta näiden sekä muiden tuotteiden kehitysprosesseista. Tutkimuksen kohteena olevista tuotteista toisen kehittämisessä on hyödynnetty mallinnusta ja toisen ei. Mallintamisen tai ajurien ohjelmoinnin teknisiin yksityiskohtiin ei paneuduta sen syvemmin, vaan keskitytään kehitysprosessiin suurempana kokonaisuutena.

Järjestelmäpiirin, valmiskomponenttien ja ajurien kehitysprosessit esitellään tämän työn luvussa 2. Luvussa kerrotaan myös, mitä mallintamisella tässä asiayhteydessä tarkoitetaan ja mihin sillä pyritään. Luvussa 3 käydään läpi työn puitteissa suoritettu tutkimus mallinnuksen vaikutuksista sekä kerrotaan, minkälaisia havaintoja tutkimuksessa tehtiin. Lopuksi luvussa 4 on koko työn yhteenveto.

## 2. JÄRJESTELMÄPIIRIN KEHITYSPROSESSI

Järjestelmäpiirejä valmistetaan pääasiassa kahta eri tyyppiä: ASIC eli *Application specific integrated circuit* sekä FPGA eli *Field-programmable gate array*. Näiden kahden oleellisin ero on siinä, että ASIC suunnitellaan ja tuotetaan alusta asti yhtä tiettyä käyttötarkoitusta varten eikä sen sisältämää logiikkaa voida enää jälkikäteen muokata. FPGA:n sisältämän logiikan voi taas puolestaan ohjelmoida uudelleen. ASIC-piirien hyvä puoli on siinä, että ne ovat tehokkaampia kuin FPGA-piirit, ja niiden valmistaminen suurissa tuotantoerissä on FPGA-piirejä edullisempaa [1]. Tässä työssä tarkastellaan ASIC-järjestelmäpiirejä, koska työn käsittelemät keskeiset ongelmat liittyvät pääasiassa näiden piirien valmistusprosesseihin.

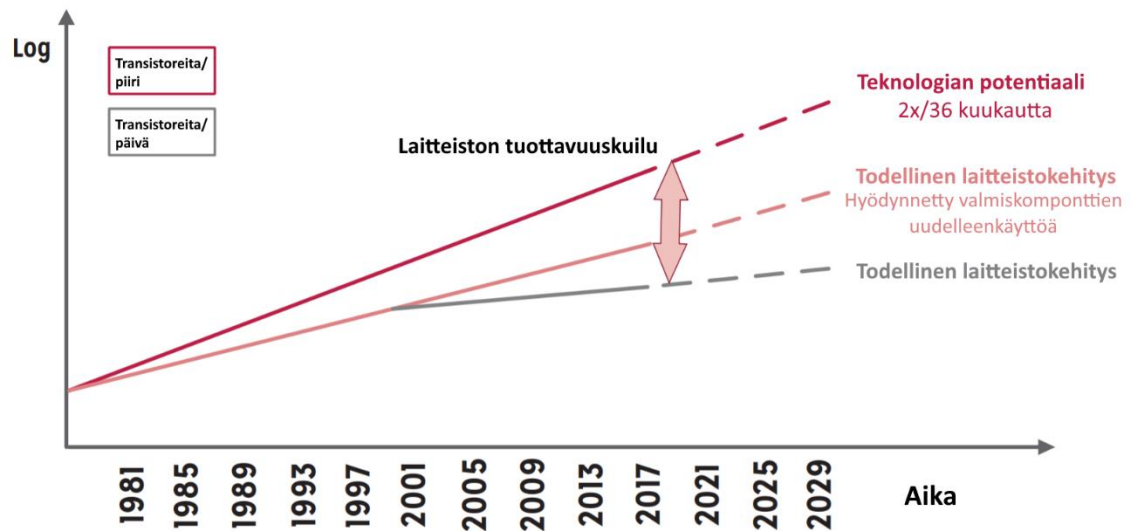
Tyypillisesti ASIC-järjestelmäpiirin kehitysprosessissa on ensin tehty laitteisto (hardware) ja vasta valmiin laitteiston päälle on alettu kehittämään ohjelmistoa (software) [1]. Tällaisen prosessin aikajana on kuvattu pelkistetyksi kuvassa 1.



**Kuva 1.** Järjestelmäpiirin kehitysprosessi.

Yllä kuvattu prosessimalli on erittäin aikaa vievä. Laitteiston kehittämiseen voi mennä vuosia, minkä jälkeen aloitettu ohjelmistokehitys venyttää tuotteen valmistumista vielä vuodella tai useammalla. Tuotteiden pitkä valmistusaika tuo paljon taloudellisia sekä teknisiä vaikeuksia. Prosessin pitkittyminen kuluttaa paljon resursseja, ja kilpaileminen muiden valmistajien kanssa hankaloituu. Teknologian kehittyessä nopeasti tuottavuuskuilu kasvaa, eli tuotannon kyvyt eivät pysy teknologisten mahdollisuuksien

mukana (kuva 2). Pahimmillaan pitkä valmistusaika voi tarkoittaa tuotteen vaatimusten vanhentumista valmistuspäivään mennessä. [1]



**Kuva 2.** Tuottavuuskuilu [2] (suomennettu).

Kuvassa 2 havainnollistetaan tuottavuuskuilua (design productivity gap) järjestelmäpiirin transistorien määränä ajan funktiona. Kuvan punainen viiva kuvaa teknologian teoreettista potentiaalia. Harmaa viiva kuvaa todellista laitteistokehitystä, ja oranssi viiva kuvaa sellaista laitteistokehitystä, jossa on hyödynnetty valmistuskomponenttien uudelleenkäyttöä. Teoreettisen potentiaalin ja todellisen laitteistokehityksen väliin jää tuottavuuskuilu.

## 2.1 Valmiskomponentit

Tässä työssä käsitellään sellaisia järjestelmäpiirejä, joiden toiminnallisuus on toteutettu suurelta osin valmiskomponenteilla. Valmiskomponentti eli *Intellectual property* -lohko tarkoittaa piirin osaa, joka suorittaa jonkin sille määrätyn toiminnon. Esimerkkinä tällaisesta toiminnosta voisi olla vaikka Fourier-muunnoksen suorittaminen tai datan puskurointi. Toisiinsa yhdistetyt valmiskomponentit muodostavat piirin toiminnallisen kokonaisuuden.

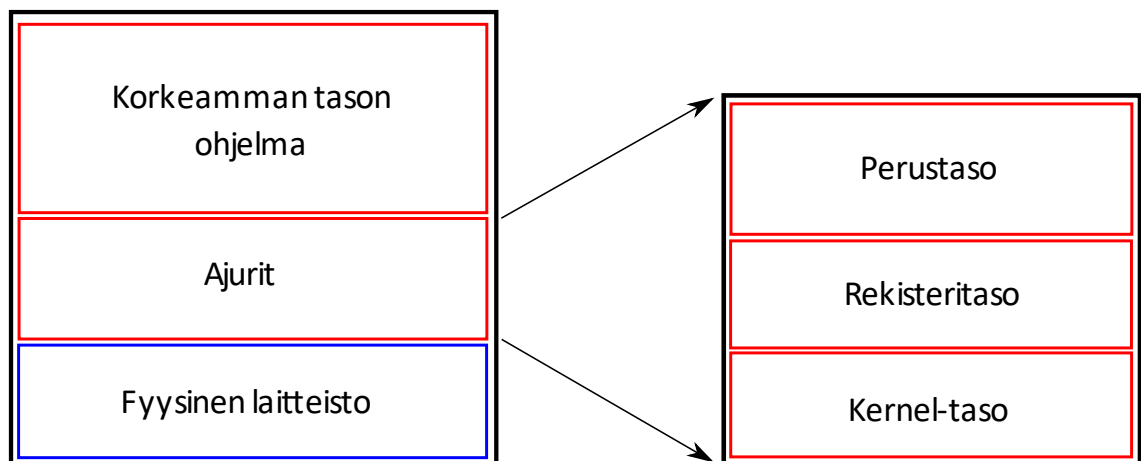
Valmiskomponenttien vahvuus on niiden uudelleenkäytettävyys. Kerran tehtyä toiminnallisuutta voidaan hyödyntää toisessa järjestelmäpiirituotteessa tai useamman kerran samassa tuotteessa. Yu-Jung Huangin et al. mukaan valmiskomponenttien uudelleenkäyttö on avainasemassa tuottavuuskuilun pienentämisessä [3, s. 62]. Tämä nähdään myös kuvasta 2. Kuitenkaan todellisuudessa komponenttien uudelleenkäyttö ei ole täysin saumatonta. Järjestelmäpiirin tehtävä on optimoida valmiskomponenteista kootun kokonaisuuden suorituskyky, ja monesti komponenttia joudutaan erikseen

muokkaamaan, jotta se olisi parempi käyttötarkoitukseensa [4, s. 2]. Tutkimuksen mukaan [5, s. 3] komponenttien uudelleenkäyttö voi säästää keskimäärin 62 % työstä, jota tehtäisiin ilman uudelleenkäyttöä.

Valmiskomponentin määritelmä on myös tavallaan häilyvä. Uudelleenkäytettävä kokonaisuus, joka suorittaa tietyn tehtävän, voi koostua muista kokonaisuuksista, jotka suorittavat tiettyjä tehtäviä. Toisin sanoen valmiskomponentti voi koostua muista valmiskomponenteista. [6]

## 2.2 Valmiskomponenttien ajurit

Jotta korkeamman tason ohjelma voisi käsitellä valmiskomponenttien laitteistoa, täytyy jokaiselle komponentille kirjoittaa oma ajuriohjelma. Ajuri on siis järjestelmäpiirin ohjelmiston matalin taso, joka tarjoaa rajapinnan korkeamman tason ohjelman ja fyysisen laitteiston välille.



**Kuva 3.** Järjestelmäpiirin ja ajurin hierarkia.

Valmiskomponenttien ajurit koostuvat yleensä kolmesta tasosta kuvan 3 mukaisesti. Kernel- eli ydintaso on ajurin matalin taso. Sen tehtävä on yhdistää komponentin laitteisto ja ohjelmisto toisiinsa. Rekisteritaso on abstraktio komponentin rekisterirajapinnasta ja tarjoaa funktioita rekistereistä lukemista ja niihin kirjoittamista varten. Perustaso sisältää komponentin toiminnan logiikkaa ja se antaa korkeamman tason ohjelmalle mahdollisimman hyödyllisen ja käyttöystävällisen rajapinnan komponentin asetusten muokkaamista varten.

Kernel- ja rekisteritasot ovat usein yksinkertaista koodia hyvin matalalla tasolla. Tästä syystä ne voidaan tuottaa automaattisesti. Pelkän rekisterirajapinnan avulla on helppoa tuottaa jokaiselle rekisterille luku- ja kirjoitusfunktiot. Koodituotannon automatisointia on

järkevä hyödyntää mahdollisimman paljon, sillä se vähentää käsin tehtävää työtä ja nopeuttaa muutosten tekemistä.

Ajurien raja ohjelmiston hierarkiassa on häilyvä. On tapauskohtaista, mihin korkeamman tason ohjelmiston ja ajurin korkeimman tason raja vedetään [6]. Yksinkertaisimmillaan ajurin ohjelmoija huolehtii vain rekisterirajapinnasta eikä valmiskomponentin sisäisellä toiminnalla tai tehtävällä ole ohjelmoijalle mitään merkitystä. Tällaisessa tapauksessa perustaso on vain korkeamman tason versio rekisteritasosta tai sitä ei tehdä lainkaan. Monesti kuitenkin ajuriin sisällytetään komponentin tai koko järjestelmäpiirin vaatimaa logiikkaa ja ohjelmoijan on tunnettava komponentin tarkoitus ja toiminta hyvinkin perinpohjaisesti [6].

### **2.2.1 Ajurien kehitysprosessi**

Ajurit eroavat korkeamman tason ohjelmistosta niin, että niiden tekeminen voidaan aloittaa jo ennen kuin laitteiston kehitys on valmis. Ajurin ohjelmoija saa valmiskomponentin suunnittelijalta komponentin määrittelyn ja rekisterirajapinnan. Nämä muuttuvat komponentin suunnittelun edetessä, mutta alustavaa ajuria voidaan alkaa rakentaa heti, kun ensimmäinen versio rekisterirajapinnasta on selvillä. [6] Tässä toimintatavassa on sekä hyvät että huonot puolensa. Vaikka komponentin vanhat versiot päivittyvätkin usein ja muutoksia ajureihin täytyy tehdä jatkuvasti, on jo valmiiksi tehtyä työtä nopeampi muokata kuin tehdä koko ajuri alusta loppuun vasta laitteiston valmistuessa. Kuitenkin tällaisessa prosessissa tehdään paljon turhaa työtä, sillä valmiskomponenttien rakenne voi muuttua radikaalisti kesken suunnittelun, eikä tällöin enää vanhalla ajuritoteutuksella ole välttämättä mitään arvoa.

Siinä missä valmiskomponentteja voidaan käyttää uudelleen, niin voidaan niiden ajureitakin. Mutta kuten aliluvussa 2.1 mainittiin, komponentit saattavat usein uudelleenkäytön yhteydessä muuttua, ja tällöin olemassa olevaa ajuria joudutaan enemmän tai vähemmän muokkaamaan. [6]

### **2.2.2 Ajurien testaaminen**

Ajurin ohjelmoija kirjoittaa ajurin perustasolle tavanomaisesti yksikkötestit. Ennen kuin laitteisto tai emulaattori on saatavilla, yksikkötestit voidaan ajaa natiivi- eli paikallisympäristössä. Paikallisympäristössä ajatut testit eivät takaa ajurien toimivuutta oikealla laitteistolla, sillä kaikki lukeminen ja kirjoittaminen tehdään rekisteritason kautta testiä ajavan tietokoneen omaan muistiin. Laitteiston toimintaa ja reagointia syötteisiin ei oteta näissä testeissä huomioon lainkaan. Paikallistesteillä yleensä vain todennetaan,

että ajurin koodissa ei ole ilmeisiä ohjelmointivirheitä, ja että ajurin perustaso käyttää rekisteritasoa oikein.

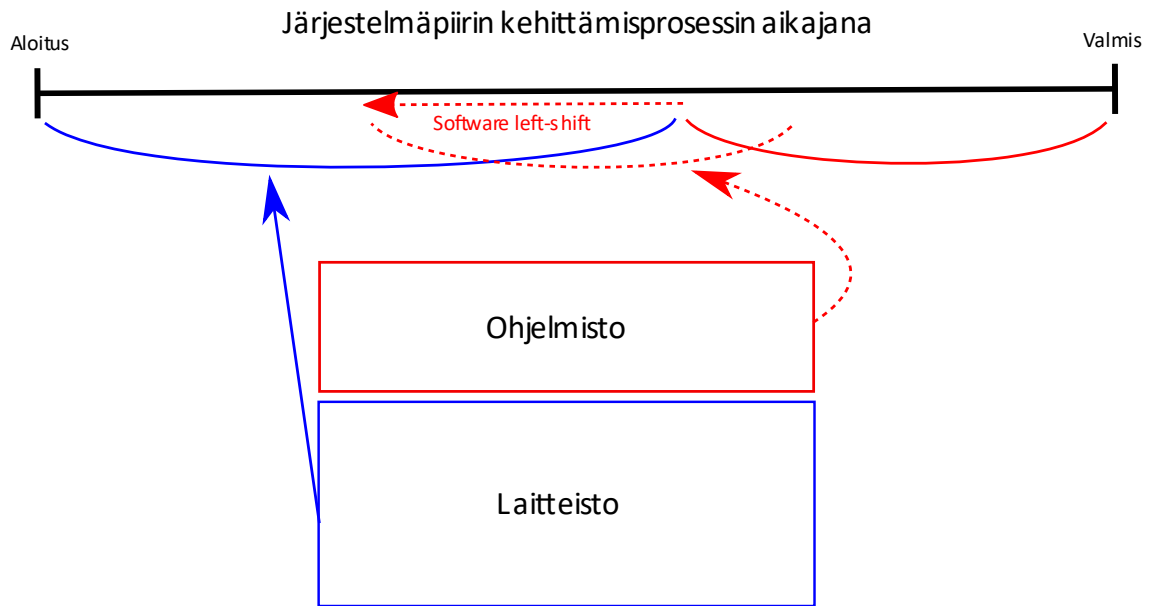
Ennen kuin fyysinen laitteisto on saatavilla, ajureita voidaan testata emulaattorilla. Emulaattori tarkoittaa laitteistoa tai ohjelmistoa, joka yhdistettynä tietokoneeseen saa tämän käyttäytymään toisenlaisen tietokoneen tavoin [7]. Järjestelmäpiirin testauksessa emulaattori on siis ohjelma tai laitteisto, joka käyttäytyy kehitettävän laitteiston tavoin. Emulaattori ja piirin laitteisto eivät eroa ajureita testattaessa lähes millään tavalla toisistaan, joten ajurien eheys voidaan hyvin pitkälti todentaa jo emulaattoritestauksella. Emuloidut rekisterit käyttäytyvät oikeiden rekisterien tavoin, ja emuloitua valmiskomponenttia voidaan konfiguroida ja käyttää oikean komponentin tavoin. Merkittävin ero laitteistolla ja emulaattorilla testattaessa on suoritusnopeus. Lopulliset suoritusajat ja ajureiden tehokkuus saadaan selville vasta kun fyysinen laitteisto on saatavilla. [8]

### 2.3 Järjestelmäpiirin mallinnus

Mallinnuksella tarkoitetaan tässä yhteydessä fyysisen järjestelmäpiirin tai jonkin sen osan toiminnallisuuden mallintamista ohjelmistolla. Malli ei todellisuudessa toimi niin kuin valmis laitteisto, eikä se suorita niitä toimintoja, joita laitteisto suorittaa. Sen tehtävä on tarjota rajapinta, joka käyttäytyy kuin valmis tuote. Rajapinta pyytää ohjelmalta samat parametrit ja antaa ohjelmalle samat paluuarvot kuin oikea laitteisto. [1]

Ihanteellisessa tilanteessa malli näkyy järjestelmäpiirin ohjelmistolle juuri niin kuin oikea laitteisto näkyisi. Tästä on ohjelmiston kehityksen kannalta monia etuja. Ohjelmistoa voidaan kunnolla alkaa rakentamaan vasta, kun valmis laitteisto ja sen rajapinta on saatavilla. Tyypillisesti malli on nopeammin valmis kuin laitteisto, ja mallin kuvatessa täydellisesti laitteistoa ohjelmistoa voidaan alkaa rakentaa mallin päälle jo ennen kuin laitteisto on valmis. Toisin kuin laitteisto, ohjelmalla tehty malli voidaan helposti monistaa, ja kaikki ohjelmiston kehittäjät saavat sen käyttöön yhtäaikaaisesti. Mallin on myös laitteistoa helpommin muutettavissa, joten muutospäätöksen jälkeen muutos saadaan ohjelmistolle näkyviin todella nopeasti, ja näin ohjelmistokehitys jatkuu nopeammin. [9]

Mallin mahdollistamalle ohjelmistokehityksen aikaistumiselle on olemassa termi *Software left-shift* [10, s. 2], jonka aiheuttamaa kehitysprosessin muutosta havainnollistetaan kuvassa 4.



**Kuva 4.** *Software left-shift.*

Aikaisempi aloitus luonnollisesti johtaa siihen, että ohjelmisto – ja näin myös koko tuote – valmistuu aikaisemmin. Mm. autovalmistaja Toyota kertoo, että mallinnus nopeuttaa kehitysprosesseja ja mahdollistaa tuotteiden toimittamisen markkinoille ennen kilpailijoita [11].

Ohjelmistokehityksen lisäksi malli hyödyttää laitteistokehitystä. Malli pyrkii kuvaamaan laitteiston toimintaa jo ennen kuin varsinaista laitteiston kehittämistä on edes aloitettu. Näin laitteiston toiminnallisuutta voidaan testata jo hyvin aikaisessa vaiheessa simuloimalla sitä mallin avulla. [9] Simuloinnin ansiosta laitteiston suunnitelmaan voidaan esimerkiksi tehdä tärkeitä muutoksia ennen kuin laitteistoa aletaan rakentamaan.

Laitteistoa voidaan mallintaa monella eri tapaa. Se, mitä mallinnetaan ja miten, riippuu tuotantoprosessikohtaisista tavoitteista ja siitä, mihin mallinnuksessa halutaan panostaa. Ihanteellisesti malli kuvaa laitteiston sisäisenkin toteutuksen täydellisesti. Tämä on kuitenkin monesti liian työläs vaihtoehto, johon voi kulua aikaa jopa enemmän kuin itse laitteiston tekemiseen. Järjestelmäpiirin mallintaminen voikin usein olla järkevämpää valmiskomponenttitasolla, jolloin komponenttien sisäänmeno ja ulostulo mallinnetaan tarkasti, mutta niiden sisäiseen toimintaan ei oteta kantaa. [9]

### 3. MALLINNUKSEN VAIKUTUKSIEN SELVITYS

Tässä työssä mallinnuksen vaikutuksia käsitellään tarkastelemalla kahden Nokian ASIC-järjestelmäpiirituotteen kehitysprosesseja, joista toisessa on hyödynnetty mallinnusta. Kummankin tuotteen kehittäminen on aloitettu viimeisen viiden vuoden aikana [12][13]. Voidaan siis todeta molempien kehitysprosessien olevan moderneja. Tuotteiden teknisiin yksityiskohtiin ei keskitytä, vaan tutustutaan tuotteiden kehitysprosesseihin ja siihen, miten ajurien kehittäminen eroaa tuotteiden välillä. Huomiotta jätetään eroavaisuudet, jotka riippuvat selvästi mallinnuksesta riippumattomista syistä, kuten esimerkiksi tuotteiden eri vaatimuksista.

Vaikutuksia on tutkittu pääsääntöisesti haastattelemalla järjestelmäpiirikehittäjiä heidän kokemuksistaan ajurien kehittämiseen ja mallintamiseen liittyen. Myös tuotteiden ajurien lähdekoodeihin, testeihin, kehitysprosessien aikatauluihin sekä muuhun tuotteisiin liittyvään materiaaliin on tutustuttu.

Tässä tutkimuksessa on kuitenkin otettava huomioon, että kyseiselle tuotantotiimille mallinnusprosessi on ensimmäinen laatuaan. Mallinnuksen hyödyt ja haitat saataisiin varmasti paremmin selville, jos kokemusta mallintamisesta olisi enemmän, ja jos sen mahdollisuudet tunnettaisiin paremmin. Tämän järjestelmäpiirin kohdalla on siis todennäköisesti myös kärsitty jonkinlaisista kasvukivuista kehitettäessä mallia ja sitä ympäröivää tuotantoprosessia.

#### 3.1 Mallinnuksen vaikutukset ajurien valmistusaikaan

Niin kuin aliluvussa 2.3 mainittiin, ajureita voidaan alkaa tekemään jo valmiskomponentin määrittelyn ensimmäisen version ollessa valmis. Ajurien kehittämisen aloitusajankohta valittiin molempiin tarkasteltaviin tuotteisiin samoin kriteerein. Molempien tuotteiden kohdalla ajureita alettiin tekemään silloin, kun ajuritiimin resurssit olivat vapautuneet ja uudelle työlle oli tarvetta. Myös laitteiston kehittämistyön täytyi olla siinä pisteessä, että ajurien tekeminen oli järkevää aloittaa [8][14]. Mallinnuksen vaikutukset aloitusajankohtaan nähtiin joko täysin merkityksettömänä tai jopa viivästyttävänä tekijänä. Mallin kehitys oli hieman myöhässä, ja jos se olisi ollut ajallaan, haastateltavan mukaan ajureita oltaisiin mahdollisesti voitu alkaa kehittää vielä aikaisemmin, sillä mallinnuksen resurssit ovat pois muista resursseista [8]. Tältä pohjalta voidaan siis todeta, että ajurien kehittämisen aloittaminen on pääosin kiinni resursseista, ja että mallinnustyön vaikutuksen aloittamisen ajankohtaan liittyvät resurssien varaamiseen.

Tuotteen mallinnustiimiin otettiin mukaan kehittäjiä ajuritiimistä. Syynä tähän oli mallinnuksen resurssipula sekä projektin johdon ajatus siitä, että osallistuminen mallinnustyöhön saattaisi pitkässä juoksussa auttaa ajurien kehittämisessä. [8] On selvää, että työvoiman lainaaminen hidastaa ajurien kehittämisprosessia ainakin aluksi. Näin kehittäjät myös kokivat. Heidän mielestään mallinnus hidasti ajurien valmistumista, mutta he myönsivät myös, että mallinnustyöstä saatu tietotaito voi sujuvoittaa ajurikehitystä myöhemmissä vaiheissa. Esimerkiksi ajurien ylläpitovaiheessa huomattujen ongelmien syyt löydetään todennäköisesti nopeammin, kun ymmärretään mallinnuksen kautta laitteiston toimintaa. [8]

Mallinnetun tuotteen ajureita on nyt tehty noin puoli vuotta, kun taas mallintamattoman tuotteen ajureita on tehty useita vuosia. Voidaan sanoa, että mallintamattoman tuotteen ajurikehitys on tällä hetkellä pidemmällä, mutta on vaikea arvioida, kuinka paljon pidemmällä se tarkalleen on. Mallinnetun tuotteen kehitysprosessi on nuorempi, ja tästä syystä se on luonnollisesti jäljessä. Kuitenkin mallinnetun tuotteen kehittäjät olivat toiveikkaita ja optimistisia sen suhteen, että tuotetta ei jouduttaisi tekemään yhtä pitkään kuin mallintamatonta tuotetta on tehty.

Tuotteiden valmistusaikojen eroavaisuuteen vaikuttaa moni asia, kuten ajurityön resurssit, laitteiston vaatimukset ja moni muu tekninen seikka, jolla ei ole tekemistä mallinnuksen kanssa. [8][14] Näiden tilastojen pohjalta on mahdotonta tehdä selkeitä johtopäätöksiä mallinnuksen vaikutuksista ajurien valmistusaikaan. Jotta parempia johtopäätöksiä voitaisiin tehdä, pitäisi tarkastella laajempaa tuotantoprosessien otantaa.

Mallin tarkoitus ei kuitenkaan ole ajurien, vaan koko järjestelmäpiirin valmistumisen nopeuttaminen. Sillä, kuinka nopeasti ajurit saadaan tehtyä, on kehittämisprosessin kannalta merkitystä vain silloin, kun se tarkoittaa koko prosessin valmistumisen nopeutumista. Monesti valmiskomponenttien ja niiden ajurien kehittäminen menee käsi kädessä, ja kun mallinnustyö edistää laitteiston valmistumista, niin se edistää myös ajurien valmistumista.

### **3.2 Mallinnuksen vaikutukset ajurien ohjelmointiin**

Ajuritimiin työvoiman lainaamisessa mallinnustyöhön on hyviä puolia myös itse ajurikehityksen kannalta. Yksi suurimmista hyödyistä, mitä ajurien ohjelmoija saa mallinnustyöstä, on laitteiston toiminnan oppiminen ja ymmärtäminen. Mallinnustyö on hyvin lähellä laitteiston kehittämistyötä. Haastateltava kehittäjä väitti, että valmiskomponentin mallintaja tuntee komponentin mallinnettavan osan toiminnan todennäköisesti yhtä hyvin kuin sen suunnittelija [8]. Mallintaessa valmiskomponenttia

voi siis oppia sen toiminnan erittäin perusteellisesti. Tästä on paljon hyötyä ohjelmoidessa komponentille ajuria. Komponentin suunnittelijan tekemään dokumentaatioon ei tarvitse enää ajurintekovaiheessa tutustua sen tarkemmin, kun ymmärtää komponentin toiminnan jo valmiiksi. Ajurin ohjelmoiminen sujuu näin helpommin, ja ajurista saa todennäköisesti tehtyä käyttötarkoitukseensa paremman.

Valmiskomponentin rekisterirajapinnan tarkka tunteminen selkeyttää ja sujuvoittaa ajurin ohjelmointia. Komponentti saattaa sisältää ns. turhia rekistereitä, joita tuotteen ohjelmisto ei tarvitse lainkaan. Tällaisille rekistereille ei tarvitsisi myöskään siis tarjota ajureissa funktioita, sillä ajureita käyttävä ohjelmisto ei tule käyttämään näitä funktioita. Päinvastoin rajapinnassa voi olla jotain rekisterejä, jotka ovat ohjelmiston kannalta erityisen tärkeitä. Näille rekistereille voitaisiin tehdä useampia funktioita, jotta niitä voitaisiin tarvittaessa käsitellä monipuolisemmin. Tällaisten rekisterien funktioiden tekemistä voitaisiin myöskin priorisoida kehittämissaikataulussa, ja ajureista voitaisiin tarvittaessa toimittaa eteenpäin suppeammat versiot ennen kuin koko valmiskomponentti ja sen ajurin suunnittelu- ja kehittämisprosessit ovat valmiita. Laitteistoläheinen mallinnustyö auttaa ajurikehittäjiä ymmärtämään rekisterien merkitystä ja luo yllämainittuja mahdollisuuksia optimoida ajureita. [8]

Valmiskomponenttien lisäksi mallinnustyössä kehittäjä voi oppia tuntemaan myös koko piirin toimintaa. Mallilla suoritetaan koko systeemin testiajoja. Haastateltava ajurikehittäjä kertoi oppineensa paljon tutustuessaan näihin testeihin. Testiajoissa mallille annetaan syöte samalla tavalla, kuin valmiille tuotteelle annettaisiin, ja mallilta saadaan ulostulo, jonka pitäisi vastata tuotteen ulostuloa samalla syötteellä. Tällaisiin testeihin tutustuminen auttaa ymmärtämään, miten tieto liikkuu systeemin sisällä, ja miten valmiskomponentit ovat yhdistettynä toisiinsa ja koko muuhun systeemiin. [8]

Kun mallintamattoman tuotteen ajurikehittäjiltä kysyttiin haastattelussa, että kuinka hyvin he tuntevat tuotteensa laitteiston, ensimmäinen vastaus oli ”ei tarpeeksi hyvin”. Kehittäjät kertoivat opetelleensa valmiskomponenttien toimintaa pääasiassa näiden suunnittelijoiden kirjoittamista dokumentaatioista. Monesti kuitenkin nämä dokumentaatiot olivat osittain puutteellisia, eikä niistä löytynyt kaikkea sitä tietoa, mitä ajurin kehittäjän olisi hyvä tietää. Paljon oltiin jouduttu lähettämään laitteiston toimintaan liittyviä kysymyksiä valmiskomponenttien suunnittelijoille ja testaajille. Lopulta kuitenkin omista osa-alueista oltiin saatu tarpeeksi hyvä ymmärrys ajurien tekemistä varten, vaikka ymmärryksessä olikin jotain aukkoja. Kehittäjät kertoivat suuremman ongelman olevan systeemin kokonaisuuden ymmärtäminen. Tästä eivät ajurien kehittäjät olleet oppineet paljoa, ja he pelkäsivät tuotekokonaisuuden ymmärtämisen puutteen

aiheuttavan ongelmia myöhemmin korjattaessa tuotteen mahdollisia toiminnallisia virheitä. [14]

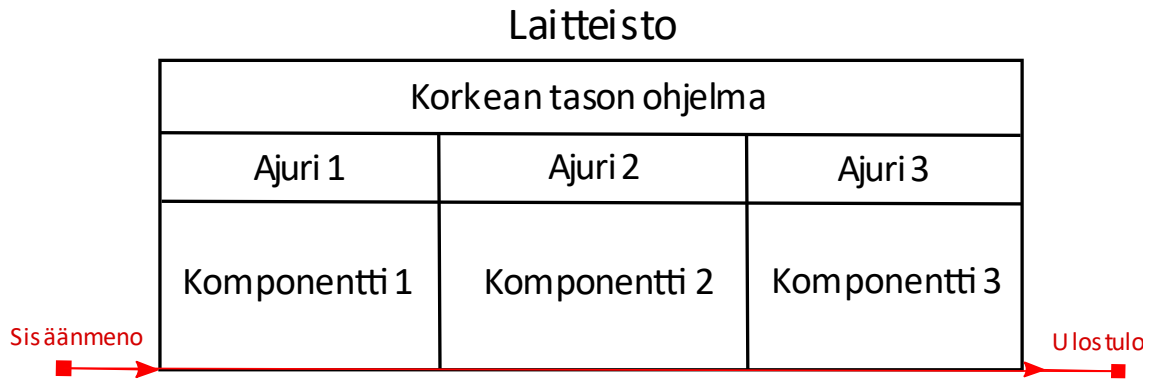
Aina ei kuitenkaan ajuria tehdessä tarvitsekaan tietää erityisen tarkasti, miten laitteisto toimii. Molempien tuotteiden ajurikehittäjät kertoivat, että monet ajurit voidaan tehdä ilman minkäänlaista ymmärrystä valmiskomponentin toiminnasta [8][14]. Niin kuin aliluvussa 2.3 kerrottiin, yksinkertaisimmillaan ajuri onkin vain komponentin rekisterirajapinnan abstraktio, joka sisältää luku- ja kirjoitusfunktiot jokaiseen rekisteriin. Joskus ajurien yksinkertaisuus on jopa vaatimus. Haastateltava kehittäjä kertoi, että välillä järjestelmäpiirin korkeatasoisemman ohjelmiston kehittäjät haluavat mahdollisimman paljon vastuuta itselleen, eivätkä halua ajurien sisältävän liikaa sisäänrakennettua logiikkaa. [14]

### **3.3 Mallinnuksen vaikutukset ajurien testaukseen**

Mallinnustyöstä saatu osaaminen auttaa ajurikehittäjiä luonnollisesti myös testaus- ja vianselvitysvaiheessa. Mallinnetun tuotteen kehittäjät uskoivat, että tulevaisuudessa, jos ajureita käytettäessä huomataan jokin vika, he osaavat paremmin paikantaa vian, sillä he tuntevat ajurien toiminnan lisäksi valmiskomponenttien ja koko laitteiston toimintaa [8]. Vastaavasti mallintamattoman tuotteen kehittäjät kertoivat törmänneensä ongelmiin vianselvityksen kanssa. He kertoivat testauksen yhteydessä löytäneensä vikoja, joiden syiden paikantaminen oli haastavaa, ja he toivoivat ymmärtävänsä laitteistoa paremmin selvittääkseen näistä ongelmista. [14]

Mallin hyödyt testauksessa eivät kuitenkaan rajoitu osaamiseen. Ajureita voidaan testata mallilla emulaattorin tapaan. Jos mallin rajapinnat vastaavat täydellisesti laitteiston rajapintoja, laitteistotason testejä saadaan ajettua paikallisympäristössä ja ajurien testaamisesta tulee huomattavasti vaivattomampaa. Tarkasteltavan mallinnetun tuotteen kohdalla kuitenkin ei ole näin. Ajurikehittäjät kokivat, ettei mallista ole toistaiseksi suurta hyötyä ajurien testaamisessa, sillä mallin rajapinnat eivät tällä hetkellä vastaa tarpeeksi hyvin laitteiston rajapintoja. He kertoivat emulaattorin olevan ikään kuin mallin kanssa kilpaileva järjestelmä, joka on vielä toistaiseksi parempi vaihtoehto testaamiselle, ja jolla kaikki tarvittava ajuritestaus saadaan tällä hetkellä suoritettua. Heidän mielestään ideaalitapaus olisi kuitenkin se, että testausta voitaisiin suorittaa mallilla, ja he toivoivat, että tämä olisi tulevaisuudessa mahdollista. [8]

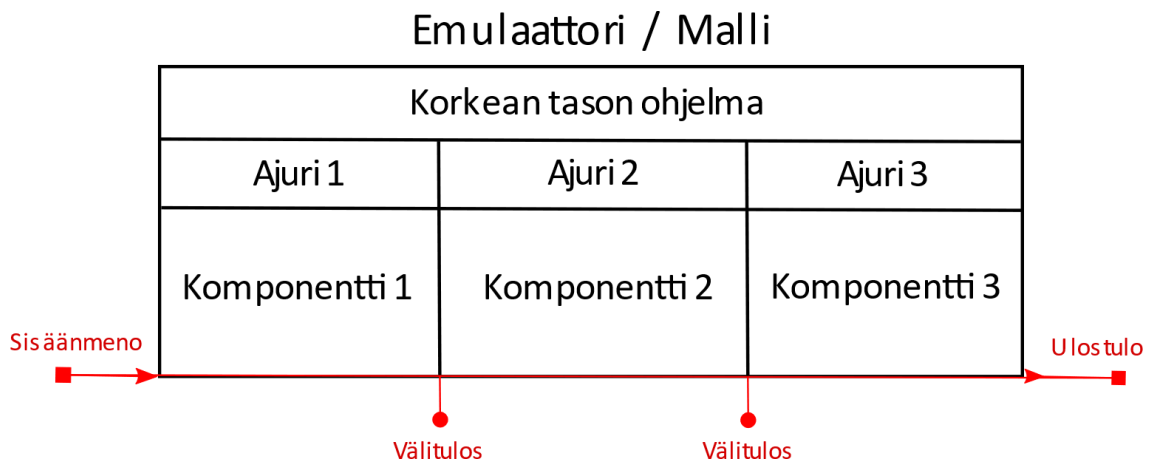
Ihanteellisessa tilanteessa mallista on konkreettista hyötyä myös vianselvityksessä. Kuvassa 5 on yksinkertaistettu esitys järjestelmäpiirilaitteistosta ja siitä, miten tieto kulkee tämän läpi.



**Kuva 5.** Laitteiston rakenne ja tiedonkulku.

Valmis laitteisto on hyvin nopea ja tehokas. Sisään menevä tieto käsitellään halutulla tapaa piirin sisällä, minkä jälkeen se annetaan eteenpäin. Tiedonkulkua laitteiston sisällä ei voida tarkastella, eikä komponenttien välisiä välituloksia nähdä. Jos testausvaiheessa huomataan ulostulon olevan virheellinen, vikaa on lähes mahdotonta paikallistaa pelkällä laitteistolla. [1]

Kuva 6 taas esittää emulaattorin ja mallin yksinkertaistettuna. Näiden sisäistä tiedonkulkua on mahdollista seurata hyvinkin tarkasti. Mallin tarkoitus on toimia niin kuin laitteiston pitäisi toimia. Emulaattori taas luodaan laitteistokehityksen pohjalta, ja se todennäköisesti sisältää samat viat kuin laitteisto. Vertailemalla siis emulaattorin ja mallin välituloksia, voidaan nähdä, missä kohtaa ne eroavat toisistaan, ja tätä kautta vika saadaan paikannettua johonkin komponenttiin. [1]



**Kuva 6.** Mallin ja emulaattorin rakenne sekä tiedonkulku.

Tällainen mallin ja emulaattorin hyödyntäminen ei kuitenkaan aina ole mahdollista. Välitulosten tarkastelun toteuttaminen vaatii ylimääräistä työtä, jota tarkasteltavan mallinnetun tuotteen kohdalla ei ole ainakaan toistaiseksi tehty.

### 3.4 Mallinnuksen vaikutukset uudelleenkäytettävyyteen

Työssä käsitelty mallinnusprosessi on tuotantotiimille ensimmäinen. Tästä syystä kyseisen tuotteen mallien uudelleenkäyttöä ei voida vielä tarkastella. Kuitenkin oikein tehdyllä mallinnuksella on mahdollisuus edistää uudelleenkäytettävyyttä. Työssä käsiteltävän tuotteen mallinnuksen johtaja kertoi haastattelussa, että mallinnuksen yksi tavoite on tehdä valmiskomponenttien malleista sellaisia, että niitä voitaisiin käyttää uudelleen [9]. Jos tulevaisuuden tuotteissa käytettäisiin mallinnetun tuotteen valmiskomponentteja, olisivat oikein tehdyt komponenttimallit myös käytettävissä uuden tuotteen mallinnuksessa. Tämä luonnollisesti edistäisi tuotteen ohjelmiston kehitystä, sillä jos olemassa olevat ajurit olisivat testattu valmiilla mallilla, korkeamman tason ohjelmaa voitaisiin alkaa rakentaa niiden päälle. Niin kuin aliluvussa 2.1 kerrottiin, kehitysprosessi lyhenee huomattavasti kun työtä on tehty valmiiksi.

Haasteeksi mallien uudelleenkäytössä tuleekin käytetty mallinnusmenetelmä. Malli voidaan sitoa tarkasti tiettyyn arkkitehtuuriin, jolloin uudelleenkäytettävyys kärsii. Haastateltava mallinnuksen ammattilainen kertoi, että valmiskomponentin ydintoiminnan ja muihin komponentteihin liitettävyyden erottaminen auttaisi uudelleenkäytettävyyttä, sillä liitettävyyttä olisi tällöin helpompi muokata. [9]

Mallin ylläpidettävyys ja muokattavuus on muutenkin suuressa roolissa uudelleenkäytön kannalta. Uudelleenkäytettäviin valmiskomponentteihin voi tulla myös muutoksia, mistä aliluvussa 2.1 mainittiinkin. Näiden muutosten vieminen mallille voi olla joko helppoa tai hankalaa riippuen siitä, miten malli on tehty.

## 4. YHTEENVETO

Tässä työssä tutustuttiin järjestelmäpiirin ja erityisesti sen ajurien tuotantoprosessiin sekä tutkittiin, minkälaisia vaikutuksia järjestelmäpiirin valmiskomponenttien mallintamisella on ajurien kehittämiseen. Tutkimusta suoritettiin pääasiallisesti haastattelemalla järjestelmäpiirikehittäjiä, joilla on kokemusta ajurikehityksestä tai mallinnuksesta.

Valmiskomponenttien mallinnuksella on selvästi paljon positiivisia vaikutuksia ajurikehitykseen. Kaikista positiivisimmat vaikutukset liittyvät ajurikehittäjien saamaan osaamiseen heidän ollessaan mukana mallinnustyössä. Valmiskomponenttia mallintanut ajurikehittäjä oppii komponentin mallinnettavan osan toiminnan perusteellisesti ja osaa tehdä ajurista käyttötarkoitukseensa paremman sekä välttää turhan työn tekemistä. Mallin ja tämän testien ymmärtäminen auttaa ymmärtämään myös järjestelmäpiirilaitteiston kokonaisuutta, mistä voi olla suuri apu esimerkiksi vianselvitysvaiheessa.

Mallinnuksella on potentiaalia olla suureksi konkreettiseksi avuksi ajurikehitykselle. Ajurien kannalta ihanteellinen malli näkyisi ajurille täydellisesti laitteiston tavoin, ja sillä voitaisiin suorittaa samoja ajuritestejä, kuin mitä laitteistolla tai emulaattorilla suoritetaan. Välitulosten tarkastelun toteuttaminen auttaisi konkreettisesti laitteistotason testaamista ja erityisesti vianselvitystä. Näitä ominaisuuksia ei kuitenkaan työssä tarkastellun tuotteen malliin ole toteutettu, tai ne eivät ole niin pitkälle vietyjä, että niistä olisi toistaiseksi konkreettista hyötyä.

Mallin uudelleenkäytettävyydestä ei voida tämän työn puitteissa sanoa paljoa, sillä mallinnusprosessi on tuotantotiimin ensimmäinen. On kuitenkin mahdollista, että uudelleenkäytettävän valmiskomponentin malli edistäisi myös komponentin ajurin uudelleenkäytettävyyttä, sillä ajuri voisi olla jo valmiiksi testattu valmiilla mallilla.

Negatiivisia vaikutuksia tuotteen mallinnuksella oli ajurikehityksen resursseihin. Kehittäjiä otettiin ajuritiimistä mallinnukseen mukaan, mikä tietenkin hidastaa ajurien kehittämistä. Iso osa ajurien kehittämistyöstä on myös niin yksinkertaista, että ylimääräisellä osaamisella ja ymmärryksellä ei ole merkittävää arvoa. Ajurien testaamisessa emulaattori toimii laitteiston korvikkeena, eikä mallilla testaamisesta ole sen suurempaa hyötyä, jos malli ei ole oikeanlainen.

Tutkimuksessa havaitut vaikutukset ovat vielä kootusti taulukossa 1. Taulukossa vaikutukset ovat eritelty plussiin, mahdollisuuksiin sekä miinuksiin.

**Taulukko 1. Mallinnuksen vaikutukset ajurikehitykseen.**

PLUSSAT	<ul style="list-style-type: none"> <li>+ Valmiskomponentin toiminnan oppiminen</li> <li>+ Kehittäjä osaa tehdä paremman ajurin</li> <li>+ Kehittäjä osaa välttää turhaa työtä</li> <li>+ Järjestelmäkokonaisuuden oppiminen</li> <li>+ Kehittäjä osaa paikantaa vian paremmin</li> <li>+ Koko tuotantoprosessin nopeutuminen</li> </ul>
MAHDOLLISUUDET	<ul style="list-style-type: none"> <li>• Ajurien testaaminen mallilla</li> <li>• Vianselvitys mallin välitulosten avulla</li> <li>• Ajurien uudelleenkäytettävyyden edistäminen</li> </ul>
MIINUKSET	<ul style="list-style-type: none"> <li>- Pois ajurikehityksen resursseista</li> <li>- Viivästyttää ajurien kehittämistä</li> <li>- Mallinnustyöstä saatu osaaminen voi olla turhaa</li> <li>- Mallista ei välttämättä ole konkreettista hyötyä ajurien testaamisessa</li> </ul>

Mallinnuksen mahdollisuudet ovat tavallaan rajattomat, mutta kaikkea ei voida mallintaa. On tuotantoprosessikohtaista, mitkä mallinnuksen päätavoitteet ovat, ja mihin mallinnuksessa panostetaan. Mallinnustyö on aina pois resursseista, mutta oikein ohjattuna mallinnukseen käytetyt resurssit voivat maksaa itsensä moninkertaisena takaisin.

Tästä tutkimuksesta ei voida kuitenkaan tehdä liian suuria johtopäätöksiä, eikä mainittuja vaikutuksia voida pitää yleispätevinä totuuksina. Tutkimuksessa tarkasteltiin vain kahta tuotantoprosessia, joista toisessa oli mukana mallinnus. Tutkimus keskittyi myöskin vain yhden yrityksen ja yhden tuotantotiimin prosesseihin, ja kyseiselle tuotantotiimille mallinnusprosessi oli ensimmäinen laatuaan. Otanta on siis kapea, mutta tutkimus osoittaa silti hyvin, mihin asioihin mallinnus vaikuttaa ensimmäisenä.

# LÄHTEET

- [1] Seppälä K., insinööri, tiiminvetäjä. Nokia Oyj, Tampere. Yksityinen opetussessio 29.1.2020
- [2] Bahar R., Jones A., Katkooori S., Madden P., Marculescu D. & Markov I. Workshops on Extreme Scale Design Automation (ESDA) Challenges and Opportunities for 2025 and Beyond. 2014. Saatavissa: <https://pdfs.semanticscholar.org/edb4/6e8223d7084a4e94388d4e9ed9df35187116.pdf?ga=2.77517660.875219240.1585737572-1565175929.1584516579>
- [3] Huang Y., Liu C., Chang S., Chuang F. & Chen C. Design of LCD driver IP for SOC applications. *Proceedings of 2004 IEEE Asia-Pacific Conference on Advanced System Integrated Circuits*, s. 62–65. Saatavissa: <https://doi.org/10.1109/APASIC.2004.1349405>
- [4] Saha D., & Sur-Kolay S. SoC: A Real Platform for IP Reuse, IP Infringement, and IP Protection. *VLSI Design*, 2011. Saatavissa: <https://doi.org/10.1155/2011/731957>
- [5] Mathaikutty D., Shukla S. *Metamodeling-Driven IP Reuse for SoC Integration and Microprocessor Design*. Boston: Artech House, Inc; 2009. Saatavissa: <http://search.ebscohost.com/login.aspx?direct=true&AuthType=cookie.ip.uid&db=nlebk&AN=305426&site=ehost-live&scope=site>
- [6] Leppäniemi P., diplomi-insinööri, tekninen johtaja. Nokia Oyj, Tampere. Haastattelu 6.2.2020
- [7] Butterfield A. & Szymanski J. "Emulator." *A Dictionary of Electronics and Electrical Engineering*, Oxford University Press, 21.6.2018. Saatavissa: <https://www-oxfordreference-com.libproxy.tuni.fi/view/10.1093/acref/9780198725725.001.0001/acref-9780198725725-e-1569>
- [8] Nieminen H., insinööri, kehittäjä & Harjunpää M., insinööri, kehittäjä. Nokia Oyj, Tampere. Haastattelu 3.3.2020
- [9] Palin A., diplomi-insinööri, tekninen johtaja, SoC Architecture. Nokia Oyj, Tampere. Sähköpostihaastattelu 24.3.2020
- [10] Mounika S. & Renuka B. S. An Approach for Early Software Development to Reduce Time to Market in today's Mobile Market. *IJSET - International Journal of Innovative Science, Engineering & Technology, Vol. 1 Issue 5, July 2014*. Saatavissa: [http://www.ijiset.com/v1s5/IJSET\\_V1\\_I5\\_18.pdf](http://www.ijiset.com/v1s5/IJSET_V1_I5_18.pdf)
- [11] Ito H. Reducing Time to Market Using Model-Based Design: Q&A with Toyota. Saatavissa: [https://se.mathworks.com/content/dam/mathworks/tagteam/Objects/t/83941\\_92285v00\\_Toyota\\_QandA\\_final.pdf](https://se.mathworks.com/content/dam/mathworks/tagteam/Objects/t/83941_92285v00_Toyota_QandA_final.pdf)
- [12] Nokian sisäinen dokumentti. Mallinnetun tuotteen Sharepoint-sivu, 2020

- [13] Nokian sisäinen dokumentti. Mallintamattoman tuotteen Sharepoint-sivu, 2020
- [14] Lahtinen V., diplomi-insinööri, kehittäjä & Palojärvi A., insinööri, kehittäjä. Nokia Oyj, Tampere. Haastattelu 11.3.2020