OULUN YLIOPISTO
UNIVERSITY of OULU

KANDIDAATINTYÖ

Laitteistoemulointi järjestelmäsuunnittelussa

Ville Kangas

Ohjaaja: Jukka Lahti

SÄHKÖTEKNIIKAN KOULUTUSOHJELMA

2015

Kangas, V. (2015) Laitteistoemulointi järjestelmäsuunnittelussa. Oulun yliopisto, sähkötekniikan osasto, sähkötekniikan koulutusohjelma. Kandidaatintyö, 21 s.

TIIVISTELMÄ

Tässä kandidaatintyössä tarkastellaan laitteistoemulaattorin käyttöä järjestelmäsuunnittelussa. Työssä esiltellään emulointi teknologiana ja verrataan sitä muihin käytettyihin mallinnusmenetelmiin. Emuloinnissa käytetyt menetelmät kuvataan ja tarkastellaan emuloinnin paikkaa suunnitteluvuossa. Esitellään eräitä laitteistovaihtoehtoja ja vertaillaan niiden ominaisuuksia.

Avainsanat: laitteistoemulointi, järjestelmäsuunnittelu, In-circuit-emulointi, simulointikiihdytys

Kangas, V. (2015) Hardware assisted system verification. University of Oulu, Department of Electrical Engineering, Degree Programme in Electrical Engineering. Bachelor's thesis, 21 p.

ABSTRACT

This thesis work studies usage of hardware emulation in system development. Emulation as a technology is introduced and compared to other modeling methodologies. Different emulation methodologies are described and emulation's place in design flow is studied. Some emulation hardware options are introduced and their features are compared.

Key words: hardware emulation, system development, in-circuit emulation, simulation acceleration

SISÄLLYSLUETTELO

LYHENTEET

ASIC	Application Specific Integrated Circuit, sovelluskohtainen mikropiiri
CPF	Common Power Format, Silicon Integration Initiativen (Si2) standardi suunnittelun tehonkulutuksen kuvaamiseen
DPI	Direct Programming Interface, SystemVerilogin ohjelmointi-rajapinta
DUT	Design Under Test, testattava suunnittelu
EDA	Electronics Design Automation, elektroniikan suunnitteluautomaatio
FPGA	Field Programmable Gate Array, laitteistoteknologia kovonkuvauskielisen suunnittelun toteuttamiseen
FSDB	Fast Signal Database, formaatti signaaliaktiivisuuden kuvaamiseen
HDL	Hardware Description Language, kovonkuvauskieli
IC	Integrated Circuit, mikropiiri
ICE	In-Circuit Emulation, in-circuit testausmenetelmä emuloinnissa
IP	Intellectual Property, tässä työssä immateriaalioikeuden suojaama aineeton omaisuus
ISS	Instruction Set Simulator, käskykantasimulaattori
JTAG	Joint Test Action Group, IEEE-1149.1 standardiin perustuva in-circuit testausmenetelmä
MPU	MicroProcessor Unit, mikroprosessori
RTL	Register Transfer Level, rekisterisiirtotaso
SAIF	Switching Activity Interface Format, formaatti signaali-aktiivisuuden kuvaamiseen
SCE-MI	Standard Co-Emulation Modeling Interface, emuloinnissa käytetty transaktiotason rajapintamäärittely, Accelleran standardi
SoC	System-on-Chip, järjestelmäpiiri
SVA	SystemVerilog Assertion, SystemVerilog väittämä
UPF	Universal Power Format, IEEE-1801 standardi suunnittelun tehonkulutuksen kuvaamiseen
UVM	Universal Verification Methodology, SystemVerilog-kieleen pohjautuva varmennusmentelmä, Accelleran standardi
VIP	Verification Intellectual Property, suunnittelun varmentamiseen tarkoitettu immateriaalioikeuden suojaama aineeton omaisuus

1. JOHDANTO

Tässä kandidaatintyössä tarkastellaan laitteistoemulaattorin käyttöä järjestelmäsuunnittelussa. Työssä kuvataan emulointi teknologiana ja verrataan sitä muihin käytettyihin mallinnusmenetelmiin. Emuloinnin käyttötarkoitus ja menetelmät järjestelmäsuunnittelussa käydään myös läpi. Lopuksi esitellään merkittävimmät markkinoilla olevat laitteistovaihtoehdot ja kerrotaan myös kirjoittajan omista käytännön kokemuksista aiheen piirissä.

Työn tekijä toimi vuosina 2001 – 2007 teknisissä tehtävissä Nokia (Siemens) Networksin emulaattoripalvelussa ja vuosina 2013 – 2014 Broadcomin Oulun yksikön emulaattoritiimin esimiehenä.

2. EMULOINTI JÄRJESTELMÄSUUNNITTELUN OSANA

2.1 Mitä laitteistoemulointi on?

Laitteistoemulointi on teknologia, jonka avulla digitaalisesta mikropiiristä (IC, Integrated Circuit) voidaan tehdä varhainen prototyyppi erikseen tätä tarkoitusta varten kehitetyssä laitteessa eli emulaattorissa. Suunnitelman toiminnallisuus voidaan mallintaa emulaattorissa täydellisesti.

Emulointiteknologian tarjoamat hyödyt:

- Kaikki suunnittelun digtaalinen logiikka voidaan mallintaa, muistit mukaanlukien
- Emuloinnilla voidaan toteuttaa riittävän suorituskykyinen ympäristö IC-suunnitelman toiminnallista varmentamista varten
- Laiteläheisen ohjelmiston kehittäminen ja järjestelmän integrointi voidaan aloittaa ennen kuin varsinaiset piirit on saatu tehtaalta
- Emulointi tarjoaa täydellisen näkyvyyden suunnittelun sisäisiin signaaleihin kovonkuvauskieltä (HDL, Hardware Description Language) tukevan simulaattorin tapaan. Tätä ominaisuutta voidaan käyttää suunnitelman varmentamisen lisäksi myös tukemaan piirin herätysvaihetta (wake-up) ensimmäisen piirierän saavuttua.
- Emulointiympäristön avulla voidaan tehdä varhaisia tuote-esittelyitä eri yhteistyötahoille, kuten asiakkaille ja rahoittajille.

Emulointiteknologialla on seuraavia rajoitteita:

- Analogisia komponentteja ei voida mallintaa.
- Emulaattorit toimivat kellojaksopohjaisesti, joten ne eivät tue ajoitussimulointeja.
- Emulaattorit ovat varsin kalliita

[1, 2]

2.2 Emulointi ja muut mallinnusmenetelmät

Digitaalista suunnittelua voidaan mallintaa useilla eri menetelmillä ennen piirien valmistumista. Taulukko 1 esittelee erilaisia mallinnusmenetelmiä ja vertaa niitä emulaattorimalliin.

Mallinnusmenetelmistä HDL-simulaattori ja korkean tason malli toimivat työasemaympäristössä ja tuovat hyvin esille sen rajoitteet. Korkean tason malli on suorituskyvyltään hyvä, mutta malli on epätarkka eikä sisällä aikakonseptia. Näiden puutteiden vuoksi se ei sovellu rekisterisiirtotason (RTL, Register Transfer Layer) kuvauksen varmennukseen eikä laiteläheisen ohjelmiston kehittämiseen. HDL-simulaattori puolestaan tarjoaa tarkan mallin, mutta juuri yksityiskohtaisuuden vuoksi malli on raskas ja työasemaympäristössä etenkin suunnitelman koon kasvaessa suorituskyvyltään heikko. Myös simulaattorin vahvuus lyhyt suunnittelusykli (simuloi –> etsi vika [debuggaa] –> korjaa suunnittelua –> käännä –

> simuloi uudelleen) on erityisen tärkeä ominaisuus varhaisessa suunnitteluvaiheessa, kun suunnitelman kypsyysaste on vielä matala. HDL-simulaattori onkin parhaimmillaan lohkotasolla toimittaessa.

Työasemaympäristölle vaihtoehdon tarjoavat emulaattori ja FPGA-prototyyppi (Field Programmable Gate Array), jotka mallintavat suunnittelun erillisessä laitteistossa. Emulaattorimallin hyviä puolia FPGA-toteutukseen verrattuna ovat suhteessa nopeampi ympäristön rakennus- ja päivitysaika, sekä hyvät vianetsintäominaisuudet. Suorituskyky on FPGA-ratkaisua matalampi, mutta silti riittävä laitteistoläheisen ohjelmiston kehitykseen. Emulaattoritoteutuksen suurin ongelma on sen huomattavasti muita menetelmiä korkeampi hinta. Hinnan vuoksi emulaattoriympäristön kopioita ei tavallisesti ole mahdollista saada käyttöön siinä määrin, kuin niille olisi projektissa tarvetta.

Esitellyt mallinnusmenetelmät täydentävät pitkälti toisiaan ja samassa projektissa voidaan käyttää useampia niistä tai vaikkapa niitä kaikkia.

[3, 4]

Taulukko 1. Digitaalisuunnittelun mallinnusmenetelmien vertailua

	HDL-simulaattori	Emulaattori	FPGA-prototyyppi	Korkean tason malli (esim. C)
Valmistusaika	Päiviä	Päiviä	Viikkoja	Päiviä
Päivitysaika	Minuutteja	Tunteja	Päiviä	Tunteja
Mallin tarkkuus	Korkea	Korkea	Korkea	Matala
Aikakonsepti	Kyllä	Kyllä	Kyllä	Ei
Vian etsinnän sujuvuus	Hyvä	Keskitaso	Matala	Hyvä
RTL-varmennus	Kyllä	Kyllä	Kyllä	Ei
Suorituskyky	Matala	Keskitaso	Korkea	Korkea
Käyttökohde	Lohkotaso	Järjestelmätaso	Järjestelmätaso	Järjestelmätaso
Kopion hinta	Matala	Korkea	Matala	Matala

3. EMULOINNIN KÄYTTÖ SUUNNITTELUSSA

3.1 Laitteiston ja ohjelmiston rinnakkainen kehittäminen

Monissa nykyajan elektroniikkatuotteissa on laitteiston lisäksi merkittävä ohjelmistosisältö. Tuotteen markkinoille saamiseksi sekä laitteiston että ohjelmiston on oltava valmiina ja niistä koostuvan kokonaisuuden on toimittava. Aiemmin laiteläheisen ohjelmiston kehitys lähti toden teolla käyntiin vasta kun elektroniikka oli saatu valmiiksi. Kehitettäessä uutta järjestelmäpiiriä (SoC, System-on-Chip), ensimmäisen piirisarjan odottaminen tehtaalta aiheuttaisi kuukausien viiveen, vaikka suunnitelman toiminnallisuus on jo valmis. Ongelma voidaan kiertää mallintamalla suunnittelu tässä vaiheessa laitteistoemulaattorilla tai FPGA-ratkaisulla. Nykyisin laitteiston ja ohjelmiston kehitys pyritäänkin tekemään mahdollisimman pitkälle rinnakkain.

Piirisuunnittelua simulointiympäristössä varmennettaessa testitapaukset perustuvat ne tehneen insinöörin tulkintaan suunnitelman spesifikaatiosta. Kun suunnittelua varmennetaan ohjelmistolla, herätteet vastaavat paremmin reaalimaailman käyttötilannetta.

Laitteistoemulaattoriympäristössä voidaan sen paremman suorituskyvyn avulla ajaa myös huomattavasti pidempiä testitapauksia kuin simulaattoriympäristössä. Mikropiirisuunnittelussa on erittäin tärkeää saada suunnitelma kattavasti varmennettua ennen sen lähettämistä tehtaalle. Mitä aiemmin suunnittelussa oleva merkittävä virhe löytyy, sitä vähäisemmät ovat sen korjaamisen kustannukset. Virhe, joka RTL-suunnitteluvaiheessa voidaan vielä korjata hetkessä, voi ilmetessään juuri ennen suunnitelman lähettämistä tehtaalle aiheuttaa useiden päivien tai jopa viikkojen viivästymisen. Myöhemmässä vaiheessa viiveet vielä kasvavat tästä ja syntyy myös merkittäviä kustannuksia. Pahimmillaan merkittävä ongelma selviää vasta loppukäyttäjien toimesta tuotteiden jo ollessa markkinoilla. Tästä voi seurata huomattavia taloudellisia tappioita ja vahinkoa yrityksen maineelle.

Kehittämällä laitteistoa ja ohjelmistoa rinnakkain, saadaan myös tietoa järjestelmän toiminnasta. Kun tämä tieto tulee riittävän varhaisessa vaiheessa, pystytään vielä tekemään muutoksia järjestelmätasolla. Voidaan esimerkiksi muuttaa tehtävien jakoa laitteiston ja ohjelmiston välillä. Näin pystytään vaikuttamaan muunmuassa tuotteen suorituskykyyn ja tehonkulutukseen.

[2, 3]

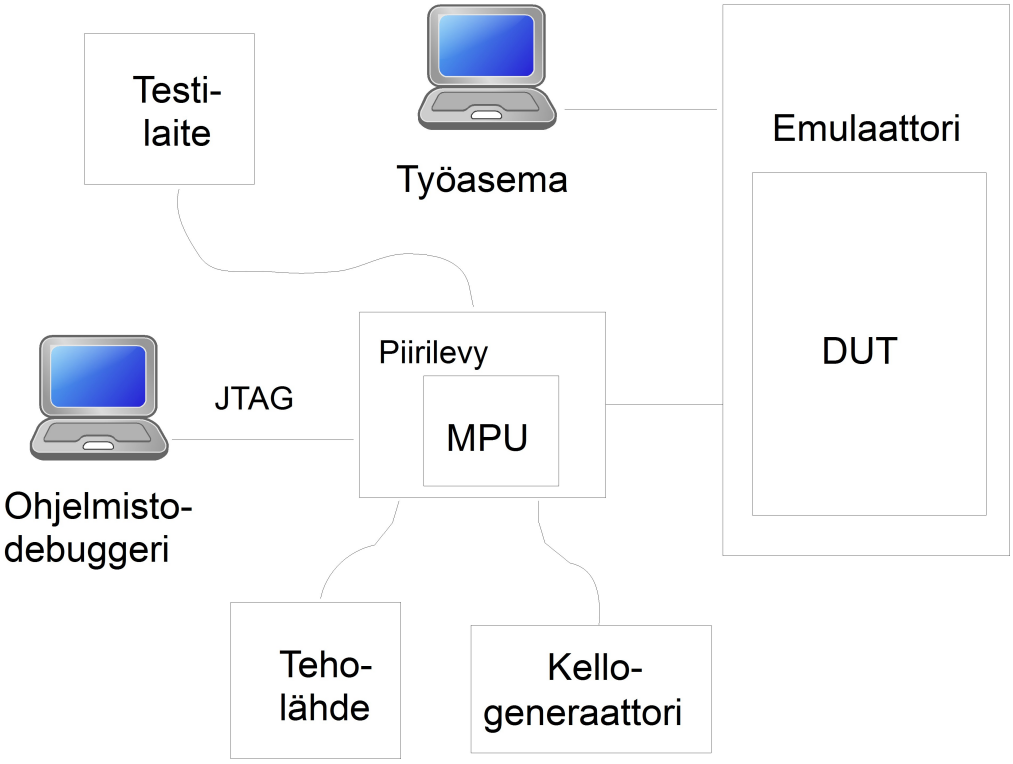
3.2 In-circuit emulointi

In-circuit-emulointi (ICE, In-Circuit Emulation) on laitteistoemuloinnin menetelmistä perinteisimpiä. Siinä ajatuksena on kytkeä emulaattorimalli todelliseen kohdejärjestelmään. Kuvassa 1 on esimerkki tyypillisestä ICE-ratkaisusta. Emulointi-istuntoa kontrolloidaan isäntätyöaseman (host workstation) avulla. Testattava suunnittelu (DUT, Device Under Test) mallinnetaan emulaattorissa. Ympäristössä käytettävät testilaitteistot kytketään emulaattoriin siinä tätä tarkoitusta varten olevien liityntöjen avulla. Ympäristön olennainen osa on mikroprosessori (MPU, MicroProcessor Unit), joka on kuvan tapauksessa toteutettu fyysisellä komponentilla

ja tarkoitusta varten erikseen kehitetyllä piirilevyllä (PWB, Printed Wiring Board). Piirilevyyn liitetään vielä ulkoinen teholähde ja kellogeneraattori. Ohjelmistodebuggeri pyörii PC:llä (Personal Computer) ja yhdistetään mikroprosessoriin JTAG-yhteyden (Joined Test Action Group, IEEE-1149.1 standardiin perustuva in-circuit testausmenetelmä) avulla.

Mikroprosessori ja emulaattorissa mallinnettu suunnitelma toimivat huomattavasti reaalimaailmaa hitaammin, käytännössä yleensä enintään noin 1 MHz nopeudella. Ulkopuoliset laitteet taas toimivat usein normaalinopeudella. Jos ympäristössä tarvitaan esimerkiksi ethernet-yhteys mikroprosessorin ja ulkoisen testi-PC:n välille, yhteyden toteuttamista varten on liikenne reititettävä erillisen nopeussillan kautta. Nopeussillan tehtävänä on puskuroida ulkomaailmasta tuleva liikenne ja synkronoida se emulaattorin kelloon ja toisinpäin.

[2, 3, 5]



Kuva 1. In-circuit-emulointiympäristön kokoonpano.

3.3 Prosessorien mallintaminen emulaattoriympäristössä

Emuloitaviin ympäristöihin kuuluu usein olennaisena osana prosessori. Prosessori saadaan mallinnettua täydellisesti tuomalla se ympäristöön fyysisenä komponenttina tai mallintamalla se emulaattorin sisällä niin sanottuna Soft IP-mallina (Intellectual Property). Prosessorista ja sen valmistajasta riippuen molemmat tavat eivät välttämättä ole aina mahdollisia.

Perinteinen tapa on käyttää fyysistä prosessoria. Tarkoitusta varten on kehitettävä erillinen piirilevy tai hankittava se emulaattorivalmistajalta. Tähän menee aikaa ja levyn kehittäminen vaatii usein myös aika tavalla asiantuntemusta. Emulaattoriympäristössä prosessoria kellotetaan varsin matalalla kellotaajuudella, koska prosessorin kellotaajuuden suhteen emuloidun suunnittelun kellotaajuuteen on oltava oikea eli sama kuin lopullisessa järjestelmässä. Matalasta kellotaajuudesta voi aiheutua erilaisia ongelmia, jotka ympäristöä rakennettaessa on ratkaistava. Toisaalta fyysistä prosessoria käytettäessä emulaattorin kapasiteettia ei kulu prosessorin mallintamiseen. Prosessorilla ei myöskään ole vaikutusta emulaattoriympäristön suorituskykyyn (suurimpaan mahdolliseen kellotaajuuteen). Fyysistä prosessoria voidaan käyttää myös silloin, kun lopullisessa järjestelmässä prosessori sijaitsee emuloidun suunnitelman sisällä. Tässä tapauksessa puhutaan niin sanotusta Bonded Out Coresta.

Soft IP-toteutustavassa prosessori mallinnetaan emulaattorissa. Valmistaja toimittaa tätä varten kovonkuvauskielisen version prosessorista. Soft IP-malli on prosessorivalmistajan kannalta yrityssalaisuus ja sen toimittamisen edellytyksenä on hyvä luottamus asiakkaaseen. Lisäksi prosessorimalli on tyypillisesti salattu. Soft IP-ratkaisun huonona puolena on se, että osa emulaattorin kapasiteetista kuluu prosessorin mallintamiseen. Prosessorin mallintaminen saattaa myös laskea sitä kellotaajuutta, jolla emuloitu suunnittelu voi enintään toimia. Hyvänä puolena on toisaalta se, että ympäristöstä tulee yksinkertaisempi ja siten luotettavampi, eikä erillistä prosessorilevyä tarvita. Soft IP-mallin toimittamiseen liittyvän luottamuksellisuusriskin vastapainoksi lähestymistapa sisältää prosessorivalmistajan kannalta myös merkittäviä hyötyjä. Markkinoille tulevaa uutta prosessorimallia käyttävä emulaattoriympäristö voidaan rakentaa jo ennen kuin kyseistä prosessoria on fyysisenä komponenttina edes saatavilla. Prosessorivalmistajan kannalta näin saadaan uusi prosessori tuottamaan liikevaihtoa mahdollisimman nopeasti ja saadaan myös enemmän suunnitteluvoittoja eli oma prosessori useampaan eri asiakkaiden tuotteeseen.

Edellä mainittujen toteutustapojen lisäksi prosessori voidaan mallintaa emulaattoriympäristössä myös käskykantasimulaattorilla (ISS, Instruction Set Simulator). Prosessorista on tätä varten oltava saatavilla erikseen käskykantasimulaattoria varten kehitetty malli. Käskykantasimulaattori toimii työasemaympäristössä ja suorituskyvyn takia prosessorimalli on yksinkertaistettu. Tästä huolimatta käskykantasimulaattorimallin avulla toteutetun emulaattoriympäristön suorituskyky jää merkittävästi alhaisemmaksi kuin aiemmin mainituilla muilla toteutustavoilla.

[2]

3.4 Transaktiotason mallinnus emuloinnissa

3.4.1 Standard Co-Emulation Modeling Interface-rajapinta

Useissa emulaattoriympäristöissä osa toiminnallisuudesta mallinnetaan emulaattorin ulkopuolella. Accelleran Standard Co-Emulation Modeling Interface 2 (SCE-MI 2) standardi määrittelee monikanavaisen kommunikaatiorajapinnan emulaattorin ja

ulkomaailman välille. SCE-MI:n avulla on pyritty ratkaisemaan useita rajapintaan liittyviä ongelmia.

Jotta emulaattorin hyvä suorituskyky pystytään hyödyntämään, ympäristön kaikkien osien nopeuden on oltava riittävä. Emulaattorin ulkopuolella mallinnettava järjestelmän osa voi periaatteessa olla mallinnettu millä abstraktiotasolla tahansa. Käytännössä useimmiten käytössä on ajasta riippumaton (un-timed) käyttäytymistason malli. Laitteistoemulaattorin puolella oleva järjestelmän osa puolestaan on mallinnettu kellojakson tarkkuudella (cycle-accurate). Tämä ei ole ongelma, koska emulaattorin suorituskyky on hyvin korkea. SCE-MI-rajapinta toimii transaktiotasolla. Näin myös rajapinnan suorituskyky saadaan maksimoitua. Transaktioihin perustuvan rajapinnan avulla yksittäinen emulaattorin ulkopuolelta tuleva viesti (message) eli transaktio voi aiheuttaa kymmeniä tai satoja kellotettuja tapahtumia (event) emulaattorin puolella. Samalla tavalla emulaattorin puolella yksittäiseen ulkopuolelle suunnattuun transaktioon voidaan pakata jopa tuhansien kellotettujen tapahtumien luoma informaatio. SCE-MI-rajapinta toimii siten myös abstraktiosiltana järjestelmän eri osien välillä.

Standardoidun rajanpinnan on toivottu myös herättävän riippumattomien suunnitelman varmentamiseen tarkoitettuja IP-malleja (VIP, Verification Intellectual Property) kehittävien yritysten mielenkiinnon. Aiemmin laitteistoemulaattoreiden ohjelmointirajapinnat (API, Application Programming Interface) olivat valmistajakohtaisia. Tästä johtuen VIP-malleja oli saatavissa lähinnä emulaattoritoimittajilta ja niiden tarjonta oli rajallista. Emulaattoreita käyttävien yritysten haluttomuus investoida toimittajakohtaisten mallien kehittämiseen oli ylimääräinen hidastava tekijä uusia emulointihankkeita suunniteltaessa.

Rajapinta on pyritty mallintamaan siten, että transaktorimalleja olisi mahdollisimman helppoa muuntaa simulaattorimaailmasta tukemaan myös emulaattorikäyttöä. Käytännössä tämä on toteutettu hyödyntämällä SCE-MI:ä kehitettäessä soveltuvin osin SystemVerilogin Direct Programming Interface-ohjelmointirajapintaa (DPI).

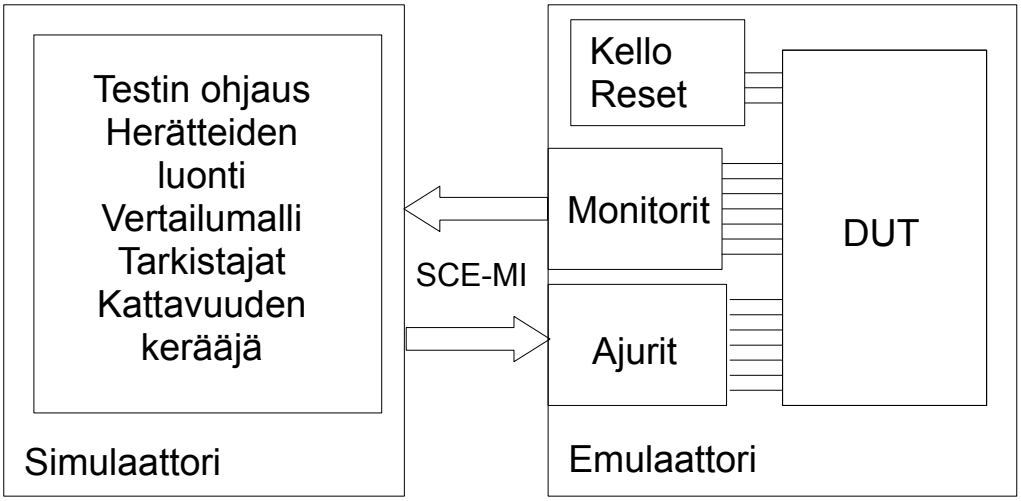
[6, 7]

3.4.2 SystemVerilog-testipenkin kiihdyttäminen emulaattorilla

SCE-MI-rajapinnan avulla on mahdollista rakentaa testipenkkirakenne, jota voidaan käyttää simulointiympäristössä ja kiihdyttää myös laitteistoemulaattorilla. Testipenkin käyttäytymistason osia ei voida syntesoida. Ne mallinnetaan työaseman puolella simulaattorissa. Loput testipenkistä syntesoidaan ja mallinnetaan emulaattorin puolella yhdessä suunnittelun kanssa. Kuvassa 2 on esitetty miten tämä jako voidaan tehdä käytännössä SystemVerilog UVM:n perustuvassa testipenkissä. Simulaattorin puolella mallinnettavia osia ovat testin ohjaus, testiherätteiden luonti ja vertailumalli. Vertailumalli on korkeamman tason kuvaus suunnitelmasta. Se tuottaa testiherätteisiin oikean vasteen, jota sitten verrataan suunnittelun tuottamaan vasteeseen. Simulaattorin puolella toteutetaan myös erilaiset tarkistajat (checkers), tulostaulukot (scoreboard) ja kattavuustiedon kerääjät. Emulaattorissa puolestaan mallinnetaan ajurit ja monitorit. Ajurien tehtävänä on vastaanottaa simulaattorin suunnalta tuleva informaatio (testin ohjaus, testiherätteet, jne.) ja välittää se edelleen. Monitorit puolestaan keräävät informaatiota (testivasteet, kattavuustieto, jne.) ja

lähettävät sen rajapinnan yli simulaattorille. Emulaattorin puolella myös luodaan tarvittavat kellot ja reset-signaali. SCE-MI-transaktorit ylittävät simulaattorin ja emulaattorin rajapinnan. Transaktorien toiminnallisuudesta osa toteutetaan simulaattorin ja osa emulaattorin puolella.

[7]

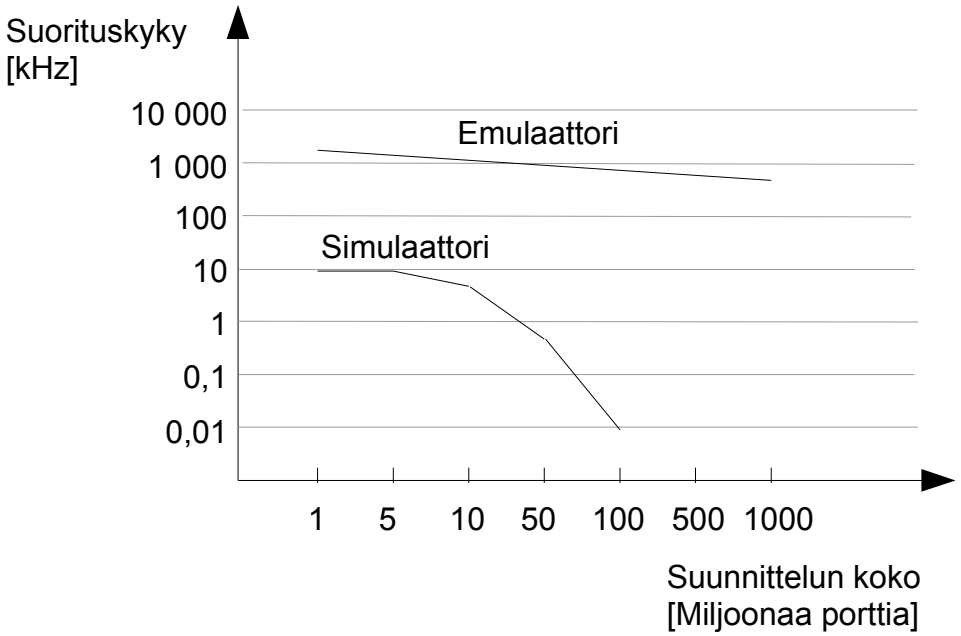


Kuva 2. Emulaattorilla kiihdytetty UVM-testiympäristö.

3.5 Simulointien kiihdyttäminen emulaattorilla

HDL-simulaattori on välttämätön työkalu SoC-suunnitelman varmentamisessa. Suunnittelun koon kasvaessa simulaattorin suorituskyky kuitenkin hidastuu merkittävästi. Simulaattorilla ei ole myöskään mahdollista ajaa kovin pitkiä testitapauksia kohtuullisessa ajassa. Suuri määrä lyhyitä testitapauksia voidaan ajaa nopeasti rinnakkain, jos saatavilla on serverifarmi ja riittävästi simulaattorilisenssejä. Yksittäisen pitkän testitapauksen simulointia ei kuitenkaan voida purkaa pieniin paloihin ja ajaa samalla tavalla rinnakkain. Simulointien suorituskykyä voidaan merkittävästi lisätä käyttämällä laitteistoemulaattoria niiden kiihdyttämiseen. Kuva 3 vertailee simulaattorin ja emulaattorin suorituskykyä suhteessa suunnitelman kokoon. Suorituskykyero on huomattava jo pienillä suunnitteluilla. suunnitelman koon kasvaessa simulaattorin nopeus pienenee nopeasti ja hyvin suurilla suunnitteluilla simulaattori hidastuu käytännössä käyttökelvottomaksi. Emulaattorin suorituskyky puolestaan säilyy varsin hyvin suunnilleen samalla tasolla, vaikka suunnitelman koko kasvaisi hyvinkin suureksi.

[4]



Kuva 3. Suunnittelun koon vaikutus emulaattorin ja simulaattorin suorituskykyyn. [4]

4. KÄYTÄNNÖN EMULOINTIRATKAISUT

4.1 Emuloinnin laitteistovaihtoehdot

Laitteistoemulaattoreita on saatavilla useilta eri toimittajilta. Taulukossa 2 on vertailtu suurimpien EDA-toimittajien tarjoamia vaihtoehtoja: Cadence Palladium XP 2 GLX, Mentor Veloce 2 ja Synopsys EVE Zebu Server 3. Nimissä numerot viittaavat laitteiden mallisukupolviin.

Laitteiden tekninen toteutus eroaa toisistaan. Palladium ja Veloce on toteutettu valmistajan erikseen tätä tarkoitusta varten kehittämillä piireillä, kun taas Zebussa käytetään Xilinxin FPGA-piirejä. Palladiumissa piirit ovat Boolean-prosessoreita ja Velocessa FPGA-piirejä. Cadence ilmoittaa käännösnopeuden ylärajaksi 75 miljoonaa ASIC-porttia (Application Specific Integrated Circuit) tunnissa ja Mentor 40 miljoonaa porttia tunnissa. Synopsys ei ilmoita käännösnopeutta. Puuttuva tieto ja kirjoittajan oma kokemus kaupallisiin FPGA-piireihin perustuvista emulaattoreista viittaavat siihen, että käännösnopeus tuskin on Zebun vahvuusalue. Emulaattorikabinetteja yhteen liittämällä kaikkien valmistajien laitteista saa rakennettua kokonaiskapasiteetiltaan valtavia 2-3 miljardin ASIC-portin kokonaisuuksia. Kaikkia laitteita voivat käyttää useat käyttäjät yhtä aikaa. Samanaikaisten käyttäjien enimmäismäärät vaihtelevat noin viidestäkymmenestä useisiin satoihin. Käytännössä tämä toteutetaan siten, että laitteiston kokonaiskapasiteetti jaetaan useamman käyttäjän kesken itse kunkin tarpeiden mukaisesti.

Kaikki vertailtavat laitteet tukevat monipuolisesti erilaisia varmennusmenetelmiä. Alhaiseen tehonkulutukseen tähtäävää (power-aware) suunnittelua varten Cadence tukee Common Power Format- (CPF) ja muut Universal Power Format-standardia (UPF). Tehonkulutuksen arviointia (power estimation) varten kaikista laitteista saadaan informaatio signaalien aktiivisuudesta Switching Activity Interface Format (SAIF) ja Fast Signal Database (FSDB) muodoissa. Kaikki laitteet tukevat SystemVerilog väittämiä (SVA, SystemVerilog Assertion) ja UVM:a. Mentorin ja Cadencen laitteista löytyy myös tuki toiminnalliselle kattavuudelle. Zebun tiedoissa tästä ei löydy mainintaa, joten tuki todennäköisesti puuttuu. Kaikki laitteistot tukevat aiemmin mainittua SCE-MI-standardia liitynnöissään ulkomaailmaan.

[8, 9, 10]

Taulukko 2. Laitteistoemulaattoreiden vertailua

	Cadence Palladium XP 2 GLX	Mentor Veloce 2	Synopsys EVE Zebu Server 3
Käytetty piiriteknologia	Custom Boolean-prosessorit	Custom FPGA-piirit	Xilinx Virtex 7 XC7V2000T FPGA-piirit
Kapasiteetti enintään	2.3 miljardia ASIC-porttia	2 miljardia ASIC-porttia	3 miljardia ASIC-porttia
Samanaikaisia käyttäjiä enintään	512	128	49
Käännösnopeus, yläraja	75 miljoonaa porttia tunnissa	40 miljoonaa porttia tunnissa	Valmistaja ei ilmoita
Low Power-tuki	CPF	UPF	UPF
Tehoestimoinnin tuki	SAIF & FSDB	SAIF & FSDB	SAIF & FSDB
SystemVerilog-väittämien tuki	Kyllä	Kyllä	Kyllä
UVM-tuki	Kyllä	Kyllä	Kyllä
Toiminnallisen kattavuuden tuki	Kyllä	Kyllä	Ei
SCE-MI-tuki	Kyllä	Kyllä	Kyllä

4.2 Emulointia ennen ja nyt

Laitteistoemulointi on kehittynyt merkittävästi niiden noin viidentoista vuoden aikana, joina työn kirjoittaja on työssään alaa seurannut.

Vielä vuosituhannen vaihteessa toimivan emulaattorimallin aikaansaaminen oli huomattavan työlästä. Sama malli ei myöskään välttämättä toiminut toisessa samanlaisessa emulaattorissa, jos se vaikkapa sijaitsi eri laboratoriossa, jonka lämpötila ja kosteusprosentti poikkesivat alkuperäisestä.

Emulaattoreissa oli myös usein pahoja luotettavuusongelmia. Pahimmillaan osia piti vaihtaa lähes viikottain. Useita tunteja kestäviä emulaattorin diagnostiikka-ajoja piti ajaa säännöllisesti ongelmien etsimiseksi. Testitapauksia ajettaessa ei virheen tapahtuessa aina voitu olla varmoja, onko virhe suunnitelussa vai emulaattorilaitteessa. Ajoittain vakiokäytäntönä virhetilanteessa ensimmäiseksi käytiin laboratoriossa resetoimassa emulaattori ja ajettiin testitapaus uudestaan, jotta nähtiin vieläkö virhe toistuu.

Emulaattoriympäristöt olivat varsin monimutkaisia erillisine prosessorilevyineen ja muine ulkoisine laitteineen. Monet kaapelit, liittimet ja hyppylangat lisäsivät ympäristön vikaherkkyyttä.

Ympäristöt eivät aina olleet täysin etäkäytettäviä, eikä meluisa emulaattori-laboratorio ollut erityisen miellyttävä työympäristö.

Nykyisin tilanne on monessa suhteessa muuttunut. Emulaattorit ovat varsin luotettavia ja Soft IP-prosessorimalleja käyttämällä ympäristöt saadaan yksinkertaisemmiksi ja vikasietoisemmiksi.

Emulaattoriympäristöt ovat täysin etäkäytettäviä. Emulaattorit ovat edelleen hyvin kalliita ja testilaitteistojen tarve nostaa kustannuksia entisestään. Suurten yritysten kannalta mahdollisuus käyttää laitteita etänä mistä päin maailmaa tahansa helpottaa investoinnin tehokasta hyödyntämistä. Jos yrityksellä on toimintaa useammalla mantereella, emulaattorikapasiteettia voidaan käyttää “seuraa aurinkoa” (follow the sun) menetelmällä, jolloin eurooppalaisten käyttäjien lähtiessä kotiin vuoroon tulevat amerikkalaiset ja sen jälkeen aasialaiset käyttäjät.

Aikoinaan käyttäjät tunsivat toisensa ja sopivat emulaattoreiden käytöstä keskenään. Käyttäjäkunnan laajentuessa useammalle paikkakunnalle syntyy tarve erilliselle varausjärjestelmälle tai “emulointia tarvittaessa” (emulation on demand) tyyppiselle ratkaisulle. Jälkimmäisessä mallissa käyttäjä kytketään automaattisesti vaatimuksia vastaavaan emulointipaikkaan (emulation seat). Järjestely tarvitsee toimiakseen suuren määrän samanlaisia emulointipaikkoja. Tässä ongelmaksi tulevat erilaiset ulkoiset laitteet, joita tarvitaan erilaisia eri tilanteissa. Nykyisen sukupolven emulaattoreissa liitynnät ulkomaailmaan kytkeytyvät kiinteästi tiettyyn osaan emulaattorin logiikkaa, eikä niihin ole pääsyä emulaattorin muista osista. Näin kaikki laitteet eivät ole käytettävissä kaikista emulointipaikoista. Jos nämä liitynnät saataisiin joustavammiksi, sillä olisi merkittävä käytön tehokkuutta lisäävä vaikutus.

Automaation lisääminen on toinen osa-alue, jolla emuloinnin käyttöä voidaan tehostaa. Usein käytetyllä interaktiivisella käyttömallilla syntyy paljon hetkiä, jolloin emulaattori on varattu tietylle käyttäjälle, mutta hän ei aktiivisesti käytä sitä. Automaation avulla seuraava testi alkaa heti edellisen loputtua ja kallis laitteisto saadaan hyödynnettyä tehokkaasti. Käyttäjän kannalta on kätevää, että hän voi laittaa testinsä jonoon ja ne tulevat automaattisesti ajetuksi. Useissa tilanteissa automaatio voisi parantaa myös käyttäjien elämän laatua. Eräässäkin projektissa suunnittelijan piti aloittaa testaus viideltä sunnuntaiaamuna, kun muita vapaita aikoja kiireelliselle testaukselle ei löytynyt koko viikolla. Emuloinnin hinnasta johtuen laitteita ei tavallisesti ole riittävästi käytettävissä kiireisinä aikoina.

5. POHDINTA

5.1 Emuloinnin menetelmien merkitys käytännössä

Kirjoittajan kokemuksen mukaan laitteistoemulaattorin käyttötarkoituksista tällä hetkellä merkittävin on sen mahdollistama laitteiston ja ohjelmiston rinnakkainen kehittäminen. Tässä tarkoituksessa emuloinnista on huomattavaa hyötyä ja tämä on näkynyt myös projektien aikatauluissa. FPGA-pohjaiset ratkaisut pyrkivät haastamaan emulaattorin asemaa tällä osa-alueella. Kompastuskivenä on tähän saakka ollut FPGA-mallin rakentamisen huomattava työläys emulaattorimalliin verrattuna. Ensimmäisen mallin valmiiksi saamisen jälkeen tämä näkyy myös aina suunnittelua päivitettäessä.

Emulaattorin käyttö simulointien kiihdyttämiseen vaatii jo tapauskohtaisemman tarkastelun. Kuten aiemmin jo todettiin, kiihdyttämisestä on hyötyä lähinnä pitkien testitapausten ajamisessa, koska lyhyitä voi ajaa tehokkaasti rinnakkain serverifarmissa. Testipenkit eivät myöskään automaattisesti sovellu simulointikiihdytykseen, vaan ne on erikseen suunniteltava sitä varten. Tämä johtuu siitä, että ainoastaan syntesoituvia osia voidaan kiihdyttää emulaattorissa ja myöskin emulaattorin ja simulaattorin rajapinnan tulisi olla mahdollisimman yksinkertainen. Testipenkin soveltuvuutta simulointikiihdytykseen on helppo tutkia simulaattorin profilointityökalulla. Koska suorituskyvyn määrää simulaattorin puolelle jäävän laskennan osuus, sen on oltava mahdollisimman pieni. Jos simulaattorin puolelle jää 5 % prosessointiajasta, emulaattorilla kiihdyttämällä voidaan teoriassa saavuttaa 20 kertainen suorituskyky. Tässä ei vielä huomioida rajapinnan aiheuttamaa viivettä. Hyvän suorituskyvyn saavuttamiseksi joudutaan helposti tekemään kompromisseja esimerkiksi testipenkin uudelleenkäytettävyyden suhteen.

5.2 Emulaattorin valinta

Emulaattoreita on saatavilla useamalta eri toimittajalta. Hankintaa suunniteltaessa on järkevää tehdä tekninen evaluointi eri vaihtoehdoilla ja mahdollisimman läheisesti suunniteltua käyttötapaa muistuttavalla esimerkillä (suunnitelma, käytetty menetelmä ja ympäristön yksityiskohdat). Eri valmistajien laitteiden välillä saattaa olla eroa siinä, miten hyvin ne suoriutuvat eri emuloinnin menetelmiä käytettäessä. Myöskään valmistajien ilmoittamia laitteiden nimelliskellotaajuuksia ei kannata vertailla sellaisenaan. Ainoastaan testaamalla asia käytännössä selviää, millaista suorituskykyä eri vaihtoehdot voivat tarjota erityyppisille suunnitteluille. Emulaattoriympäristöjä suunniteltaessa on hyvä pyrkiä tekemään ratkaisuja, jotka eivät sido tiettyyn laitetoimittajaan. Tämä on usein varsin haasteellista. Asiakkaan kannalta riippumattomuus tietystä laitetoimittajasta on eduksi kaupallisissa neuvotteluissa.

5.3 Emulaattori-insinöörin työstä

Emulaattori-insinöörin työssä pääsee näköalapaikalle seuraamaan laitteiston ja ohjelmiston kehittämistä. Työssä tarvitaan monipuolista osaamista. On hyvä ymmärtää IC-suunnittelua ja testattavan suunnitelman toimintaa. Usein emulaattoriympäristöjä varten on kehitettävä erikseen erilaisia testauksessa tarvittavia testipenkin osia. Emulaattorikäännökset ovat varsin monivaiheisia ja työkalujen verrattain pienen käyttäjämäärän vuoksi työkaluongelmat eivät ole harvinaisia. Yleisestä EDA-työkaluosaamisesta onkin paljon hyötyä. Suuri osa emulaattorien käyttäjistä on ohjelmistokehittäjiä. On hyvä ymmärtää ohjelmistosuunnittelijoiden toimintatapoja ja heidän käyttämäänsä ammattikieltä. Emulaattori-insinööri toimii palveluammatissa, joten asiakaspalveluhenkinen suhtautumistapa käyttäjäkuntaan on syytä omaksua. Emulaattorit ovat varsin kapea osaamisen alue ja emuloinnin kustannusten vuoksi laitteita on lähinnä hyvin suurilla yrityksillä. Tällä hetkellä avoimia työpaikkoja emulaattori-insinööreille näyttää olevan lähinnä Yhdysvalloissa.

6. YHTEENVETO

Laitteistoemuloinnin avulla digitaalisesta mikropiiristä saadaan tehtyä kellojakso-tarkka ja hyvän suorituskyvyn omaava malli. Emulaattorin avulla laitteistoa ja ohjelmistoa voidaan kehittää rinnakkain. Emulaattori, HDL-simulaattori ja FPGA-prototyyppi täydentävät menetelminä toisiaan ja samassa projektissa saatetaan käyttää jopa niitä kaikkia.

In-circuit-emuloinnin avulla emulaattorissa mallinnettua suunnittelua päästään testaamaan sen lopullisessa kohdeympäristössä. Prosessorien mallintaminen onnistuu liittämällä järjestelmään fyysinen prosessori tai käskykantasimulaattori. Soft IP-prosessorimallin tapauksessa prosessorin valmistaja toimittaa kovonkuvauskielisen version, joka voidaan mallintaa emulaattorin sisällä.

Emulaattori voidaan liittää sen ulkopuolella oleviin järjestelmän osiin käyttämällä SCE-MI-standardin mukaista rajapintaa. SCE-MI-rajapinta on toteutettu transaktio-tasolla korkean suorituskyvyn mahdollistamiseksi.

Emulaattoria voidaan käyttää myös kiihdyttämään simulaatioita. Simulaattori kytketään emulaattoriin ja mahdollisimman suuri osa logiikasta mallinnetaan emulaattorin puolella. Tästä on hyötyä erityisesti pitkien testitapausten ja suurten suunnittelujen tapauksessa.

Emulaattoreita on saatavilla useilta eri laitetoimittajilta. Esimerkkinä voidaan mainita Cadence, Mentor ja Synopsys. Emulointi on kehittynyt menetelmänä merkittävästi viimeisten 15 vuoden aikana. Erityisesti laitteiden luotettavuudessa ja käytettävyydessä on tapahtunut edistystä. Nykyisin emulaattoriympäristöjä voidaan käyttää etänä vaikka toiselta mantereelta käsin.

7. VIITTEET

Internet julkaisu
[1] Rizzatti L. (2014) Hardware Emulation Goes Mainstream. www.chipdesignmag.com

White paper
[2] Turner R. (2005) Accelerated Hardware/Software Co-Verification Speeds ”First Silicon and First Software”. Cadence Design Systems, Inc.

Konferenssijulkaisu
[3] Tutliani S. & Kumar J. (2007) Speeding-up HW/SW Co-Development Using HW Emulation. Sun Microsystems, CadenceLive 2007

Internet julkaisu
[4] Rizzatti L. (2014) When to use simulation, when to use emulation. www.electronicproducts.com

White paper
[5] Wouters M. (2007) The Target Platform Methodology for HW/SW Debugging Before Silicion. IMEC

Standardi
[6] (2014) Standard Co-Emulation Modeling Interface (SCE-MI) Reference Manual Version 2.2. Accellera

White paper
[7] Van Der Schoot H., Saha A., Garg A. & Suresh K. (2010) Off To The Races With Your Accelerated SystemVerilog Testbench. Mentor Graphics

Esite
[8] (2013) Cadence Palladium XP II Verification Computing Platform. Cadence Design Systems, Inc.

Esite
[9] (2013) Veloce2 brochure, Mentor Graphics

Internet sivu www.synopsys.com
[10] Zebu Server 3 informaatio (luettu 18.5.2015)