



TAMPEREEN TEKNILLINEN YLIOPISTO

HANNU RANTA
IP-XACT-JÄRJESTELMIEN SUUNNITTELU KACTUS 2-
TYÖKALULLA
Kandidaatintyö

Tarkastaja: Erno Salminen

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO

Tietotekniikan koulutusohjelma

RANTA, HANNU: IP-XACT-järjestelmien suunnittelu Kactus 2-työkalulla,
Designing IP-XACT-systems with Kactus 2 software

Kandidaatintyö, 22 sivua, 21 liitesivua

Toukokuu 2012

Pääaine: Digitaali- ja tietokonetekniikka

Tarkastaja: Erno Salminen

Avainsanat: Sulautettu järjestelmä, IP-XACT, IEEE1685, Kactus 2, System on chip, SoC

Sulautettujen SoC-järjestelmien monimutkaisuuden ja kompleksisuuden kasvaessa on pyritty kehittämään lukuisia keinoja helpottamaan suunnittelua. IP-lohkojen uudelleenkäyttö on eräs niistä ja käytännössä ainoa mahdollisuus suunnitella nykyaikaisia monimutkaisia piirejä. IP-XACT standardi pyrkii helpottamaan uudelleenkäyttöä tarjoamalla alustariippumattoman lähestymistavan. IP-XACTin idea perustuu IP-lohkojen paketointiin XML-metadatan sisään. Komponentin todellinen toteutus, esimerkiksi VHDL-tiedosto, vain linkitetään osaksi IP-XACT komponenttia. Menetelmä mahdollistaa tehokkaan uudelleenkäytön.

Tässä kandidaatintyössä keskitytään IP-XACT standardiin ja sitä hyödyntävään Kactus 2 ohjelmistoon. Kactus 2 tukee kaikkia IP-XACT standardin määrittämiä ominaisuuksia ja lisäksi laajentaa ajattelutavan koskemaan myös ohjelmistokomponentteja.

Työn osana toteutettiin kaksi perustason tutoriaalia, joiden tarkoituksena on opettaa IP-XACT standardin ja Kactus 2 järjestelmän peusteet. Tutoriaalit testattiin viiden henkilön suuruisella testiryhmällä ja niiden toimivuus analysoitiin.

SISÄLLYS

1	Johdanto.....	1
2	IP-XACTin käyttö	4
2.1	Uudelleenkäyttö.....	4
2.2	IP-XACT standardi yleisesti.....	4
2.2.1	Tekninen toteutus.....	5
2.2.2	Käyttö.....	7
2.2.3	Tulevaisuus.....	9
2.3	Muut samankaltaiset tekniikat.....	9
2.3.1	SystemC.....	9
2.3.2	OpenAccess API.....	10
2.3.3	UML Profile for Modeling and Analysis of Real-Time and Embedded systems (MARTE).....	11
3	Kactus 2.....	12
3.1	IP-XACTin suunnitteluvuo ja Kactus 2.....	13
4	Tulokset.....	15
4.1	Tutoriaalien käyttäjätestit.....	17
5	Yhteenveto.....	19
	Lähteet.....	21
	Liite 1: counter.vhdl.....	23
	Liite 2: counter.vhdl:n IP-XACT-kuvaus puumuodossa.....	25
	Liite 3: Toteutetut tutoriaalit.....	27

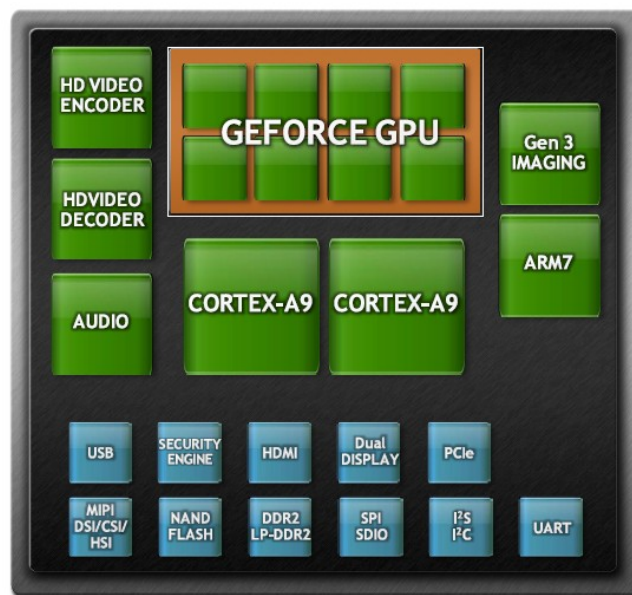
TERMIT JA NIIDEN MÄÄRITELMÄT

Termi	Määritelmä
Cyclone II	Alteran FPGA-piiri
EDA	engl. Electronic Desing Automation. Yleisnimitys suunnitteluohjelmille.
FPGA-piiri	engl. Field-Programmable Gate Array. Uudelleenohjelmoitava logiikkapiiri
GPL2	engl. General Public License. Vapaiden ohjelmien jakelemiseen tarkoitettu lisenssi
IEEE	engl. Institute of Electrical and Electronics Engineers. Kansainvälinen tieto- ja sähkötekniikan järjestö
IP	engl. Intellectual Property. Aineeton omaisuus
IP-lohko	Uudelleenkäytettävä lohko
IP-XACT	XML-formaatti, jonka avulla kuvataan uudelleenkäytettäviä lohkoja
Kactus 2	Tampereen teknillisessä yliopistossa kehitetty ohjelmisto IP-XACT-järjestelmien toteuttamiseen
MARTE	engl. Modeling and Analysis of Real Time and Embedded systems. Standardi sulautettujen sovellusten mallintamiseen UML 2-kuvauksella
Mooren laki	Intelin perustajan Gordon E. Mooren vuonna 1965 tekemä havainto, jonka mukaan transistorien lukumäärä kaksinkertaistuu mikropiireissä joka toinen vuosi.
MP-SoC	engl. Multiprocessor System-on-Chip. Useamman suorittimen sisältävä SoC-järjestelmä
Object Management Group, OMG	UML:n ja siihen perustuvien standardien kehityksestä vastaava organisaatio
OpenAccess	EDA-tietokanta ja siihen liittyvä rajapinta, joiden tarkoituksena on mahdollistaa suunnittelutyökalujen välinen interaktio
RTL	engl. Register-transfer level. rekisterisiirtotason laitteistonkuvaus, josta voidaan lsyntesoida porttitason kuvaus
SI2	engl. Silicon Integration Initiative. OpenAccess API:n kehitystä koordinoiva organisaatio
SoC	engl. System-on-Chip. Yhdelle piirille integroitu useista osista koostuva järjestelmä

SPIRIT Consortium	Elektroniikkavalmistajien yhteenliittymä, joka vastaa IP-XACT standardin suunnittelusta
System C	C++:n laajennus, joka mahdollistaa tarkan järjestelmätason laitteistokuvauksen tekemisen
TGI	engl. Tight generator interface. IP-XACT generaattorin rajapintamäärittely
UML	engl. Unified Modeling Language. Object Management Groupin (OMG) vuonna 1997 standardoima graafinen mallinnuskieli.
VHSIC	engl. Very-high-speed integrated circuit. Integroitu digitaalinen piiri
VHDL	engl. VHSIC hardware description language. Laitteistonkuvauskieli
VLNV	engl. Vendor, Library, Name, Version. IP-XACTin tapa yksilöidä komponentit
XML	engl. eXtensible Markup Language. Standardi, jolla tiedon merkitys on kuvattavissa tiedon sekaan.
XSLT	XSL Transform. Muunnosprosessi jolla XML-tiedosto voidaan muuttaa toiseksi

1 JOHDANTO

Sulautettujen järjestelmien koko ja kompleksisuus kasvaa jatkuvasti nopealla tahdilla teknologioiden kehittyessä. Tuotekehityksen suurimpina ongelmina on olemassa olevan toiminnallisuuden siirtäminen käytössä olevalle alustalle, joka saattaa merkittävästi erota alkuperäisestä kehitysalustasta. Modernit Soc-järjestelmät (engl. System-on-Chip) sisältävät suuria määriä erilaisia IP-lohkoja (engl. Intellectual Property), kuten esimerkiksi muisteja ja prosessoreita. [1]. *Kuva 1.1* havainnollistaa modernin SOC-järjestelmän rakennetta ja kompleksisuutta (esimerkkinä Nvidia Tegra 2 mobiilipiiri). Kuvasta on nähtävissä, että piirillä sijaitsee esimerkiksi kolme ARM Cortex-A9 prosessoria ja GeForce grafiikkayksikkö. Lisäksi voidaan havaita erityyppisiä liityntöjä ja muisteja.

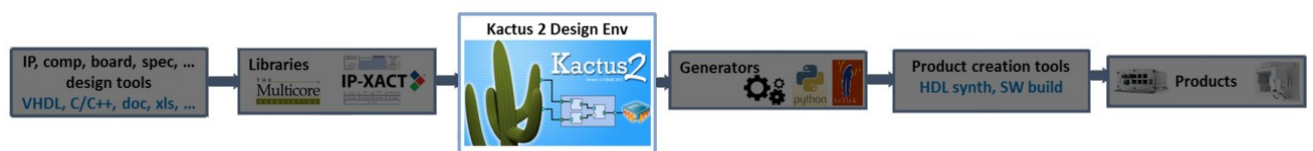


Kuva 1.1: Nvidia Tegra 2-piirin lohkokaavio [2]

Kuvassa 1.1 esitetyn kaltaisten monimutkaisten piirien tapauksessa suunnitteluaikaa pystytään huomattavasti lyhentämään uudelleenkäyttämällä IP-lohkoja. Uudelleenkäytön helpottamiseen on kehitetty metadatapohjaisia kuvausjärjestelmiä, joista tässä kandidaatintyössä keskitytään IP-XACT-järjestelmiin ja niiden toteuttamiseen Kactus 2-työkalulla.

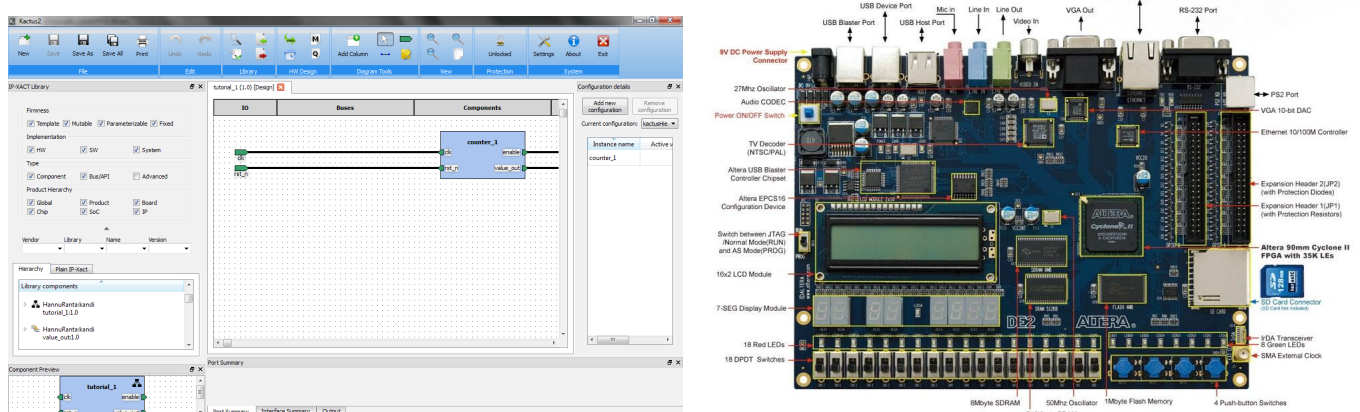
IP-XACT on IEEE:n (Institute of Electrical and Electronics Engineers) vuonna 2009 standardoima tekniikka IP-lohkojen kuvaamiseen metadatan avulla [3]. Kactus 2 on Tampereen teknillisen yliopiston Tietokonetekniikan laitoksella kehitetty avoimen lähdekoodin ohjelmisto, joka on suunnattu erityisesti FPGA-pohjaisten (Field Programmable Gate Array) System-on-Chip-järjestelmien suunnitteluun [4].

Tässä kandidaatintyössä tutustutaan Kactus 2-ohjelmistoon ja toteutetaan yksinkertaisista IP-lohkoista koostuva järjestelmä, joka syntesoidaan Alteran FPGA-piirille. Käytetty suunnitteluvuoro on kuvattu *kuvassa 1.2*. Tämän kandidaatintyön kannalta oleelliset osat on esitetty kirkkaammalla värillä. Kuva esittää suunnitteluvuoron etenemisen vasemmalta oikealle. Ensimmäisessä vaiheessa tehdään suunnittelu- ja määrittelydokumentit ja toteutetaan tarvittavat ohjelma- ja laitteistokomponentit. Sitten valmiit komponentit viedään IP-XACT-kirjastoon ja yhdistetään suunnitteluohjelmassa toimivaksi toteutukseksi. Generaattori tuottaa IP-XACT-kuvauksista valmiin syntesoitavan piirin.



Kuva 1.2: Suunnitteluvuoro ja Kactus 2 [4]

Työssä käytetyt resurssit on esitetty *kuvassa 1.3*. Työn pääosassa on Kactus 2-ohjelmisto. Laitteistona toimii Alteran DE2 Developer and Education Board-kerhityskortti, jolle suunnittelut syntesoidaan.



Kuva 1.3: Käytettävät resurssit. Vasemmalla Kactus 2-ohjelmisto ja oikealla Altera DE2 FPGA-kehityslauta [5]

Dokumentin rakenne on seuraava: *Luvussa 2* kuvataan IP-XACT standardin taustalla oleva teoria ja tutustutaan sen käyttöön ja tulevaisuudennäkymiin. Lisäksi tutustutaan muihin vastaaviin tekniikoihin ja verrataan IP-XACTia niihin. *Luvussa 3* kuvataan

Kactus 2-ohjelmisto ja kerrotaan sen käytöstä IP-XACT järjestelmien kehittämiseen.

Luvussa 4 esitellään muita IP-XACT standardia käyttäviä ohjelmia ja vertaillaan niitä Kactus 2-ohjelmistoon. *Viides luku* sisältää yhteenvedon. Työhön liittyvien IP-lohkojen VHDL-koodit (Very High Speed Integrated Circuit Hardware Description Language) ja esimerkki IP-XACT metadatatiedostosta on liitetty dokumentin loppuun.

2 IP-XACTIN KÄYTTÖ

Tässä luvussa kuvataan IP-XACT standardin perusteet, tekninen toteutus ja käyttökohteet. Lisäksi perehdytään standardin tulevaisuudennäkymiin ja muihin vastaaviin tekniikoihin.

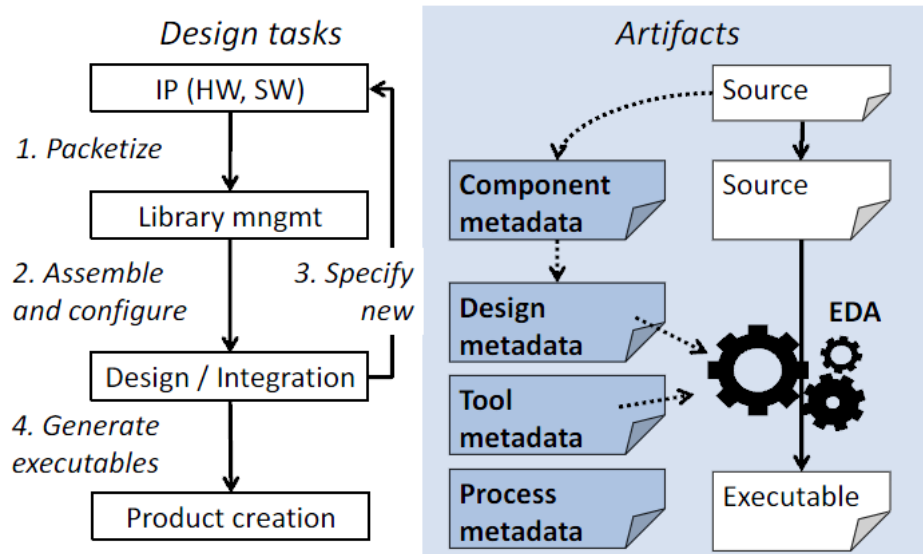
2.1 Uudelleenkäyttö

Uudelleenkäyttö on nykyaikaisten SoC-järjestelmien suunnittelussa keskeisessä asemassa. [6] Transistorimäärien kasvaessa ja piirien kompleksisuuden lisääntyessä suunnitteluun käytetty aika kasvaisi radikaalisti ilman IP-lohkojen uudelleenkäyttöä. Uudelleenkäytön avulla transistorien määrää ja piirien kompleksisuutta saadaan edelleen jatkuvasti kasvatettua ja Mooren laki pidettyä voimassa. EDA-työkalujen valmistajat ovat huomanneet uudelleenkäytön merkityksen teollisuudelle ja kehittäneet sitä tukevia ratkaisuja. Ne ovat kuitenkin vaatineet pidättäytymistä valmistajakohtaisesti tietyssä tekniikassa. IP-XACT standardi pyrkii vastaamaan tähän haasteeseen tarjoten universaalin tuen IP-lohkojen uudelleenkäytölle.

2.2 IP-XACT standardi yleisesti

IP-XACT on SPIRIT Consortiumin kehittämä ja IEEE:n vuonna 2009 standardoima XML-pohjainen (Extensible Markup Language) tekniikka IP-lohkojen kuvaamiseen SoC-järjestelmissä. Tekniikka mahdollistaa IP-lohkojen rakenteen, väylien, signaalien, muistikarttojen ja osoiteavaruuden abstraktin kuvauksen. Lisäksi standardin avulla on mahdollista kuvata useista alijärjestelmistä koostuvia järjestelmiä. Tämä mahdollistaa IP-lohkojen helpon käyttöönoton ja integraation projektiin kohdealustasta riippumatta [7].

Kuvassa 2.1 näkyy IP-XACT pohjaisen suunnitteluvuon tärkeimmät vaiheet. Kuvan vasemmalla puolella on esitetty suunnitteluvaihe ja oikealla on siihen liittyvän IP-XACT-objekti. Kuva etenee ylhäältä alaspäin. Alussa IP-lohkot paketoidaan ja lisätään komponenttikirjastoon. Tämän jälkeen ne integroidaan ja lopullinen tuote luodaan EDA-ohjelmistolla.



Kuva 2.1: IP-XACT pohjainen suunnitteluvuon [4]

2.2.1 Tekninen toteutus

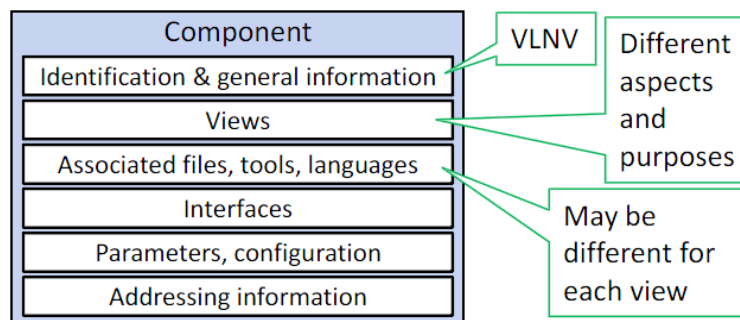
IP-XACT standardi määrittelee IP-lohkojen XML-metadatan kuvaustavan ja suunnitteluohjelmistojen käyttämät rajapinnat. Standardi määrittelee seitsemän erilaista päätason objektia [3] [8]. Niitä ovat *IP-XACT component*, *bus definition*, *abstraction definition*, *design*, *design configuration*, *abstractor* ja *generator chain*.

Näistä IP-XACT *component* sisältää yksittäisen lohkon metadatan. Tähän kuuluu esimerkiksi fyysisen rajapinnan kuvaus, lähdekooditiedostot ja konfigurointi-arvot. Bus definition ja abstraction definition sisältävät tiedon komponenttien välisistä rajapinnoista. Design sisältää tiedon kaikkien komponenttien ja niiden välisten väylien suhteista toisiinsa. Se voi kuvata myös hierarkkisen komponentin sisäisen rakenteen.

Liitteessä 2 on esitetty Kactus 2 ohjelman luoman IP-XACT-komponentin rakenne. Komponentti on toiminnaltaan yksinkertainen binäärilaskuri ja sen toiminnallisuuden määrittelevä VHDL-koodi löytyy liitteestä 1. Liitteen 2 rakenteesta ilmenee hyvin XML-dokumenteille tyypillinen hierarkisuus, jota myös IP-XACT hyödyntää. IP-XACT suunnitteluympäristö käsittelee liitteen 2 mukaisia metadatatiedostoja suorien lähdekooditiedostojen sijaan. Nämä metadatatiedostot yksilöidään valmistajan,

kirjaston, nimen ja version mukaan (VLNV eli vendor, library, name ja version). VLNv-tietojen tulee olla kaikille komponenteille uniikkeja [4].

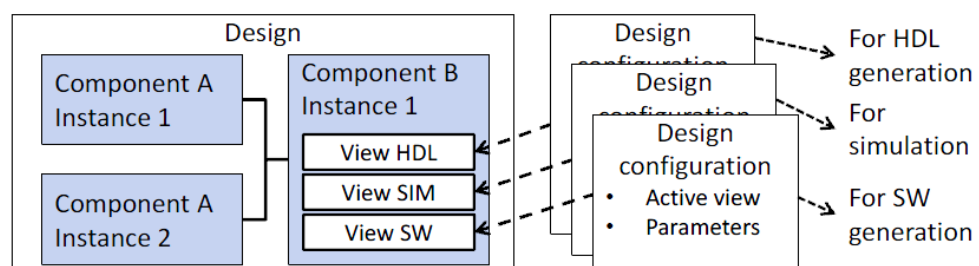
IP-XACT objekti voi sisältää kaikkien abstraktiotasojen kuvaukset samassa komponentissa. Myös eri näkymien (view), konfiguraatioiden (configuration) ja versioiden (version) sisällyttäminen samasta komponentista on mahdollista. *Kuvassa 2.2* on esitetty IP-XACT komponentin rakenne. Komponentti rakentuu yleisestä informaatiosta, näkymistä, rajapinnoista, parametreista ja osoiteinformaatiosta.



Kuva 2.2: IP-XACT-komponentin rakenne [4]

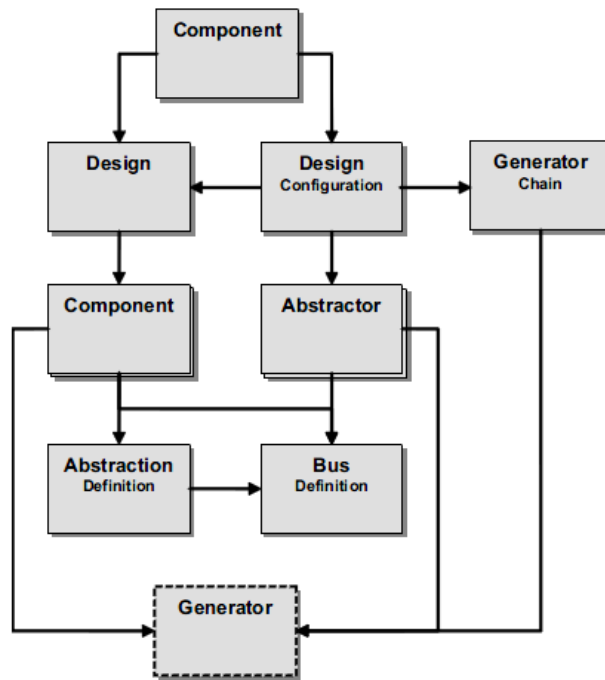
Esimerkkinä erityyppisistä näkymistä (view) ovat esimerkiksi RTL (register-transfer level), simulaatio ja dokumentaatio. Erilaisiin näkymiin voi liittyä näkymäkohtaisia tiedostoja (associated files). Näin on mahdollista sisällyttää yhteen IP-XACT komponenttiin esimerkiksi erillinen simulaatiomalli ja syntesoitavissa oleva toteutus.

Komponentit ja väylät muodostavat suunnitteluja (design), jotka toimivat kuten perinteisten suunnitteluohjelmien avulla toteutetut ylimmän tason toteutukset. Suunnittelut voivat sisältää erilaisia konfiguraatioita eri tarkoituksiin kuten simulointiin ja varsinaiseen toteutukseen. *Kuvassa 2.3* näkyy esimerkki suunnittelun rakenteesta. Suunnittelussa käytetään kahta komponenttia, A ja B. Komponentista A on kaksi eri instanssia. Komponentista B on kolme eri näkymää syntesointia, simulointia ja ohjelmistokehitystä varten ja jokaiselle näkymälle on oma konfiguraatio.



Kuva 2.3: IP-XACT suunnittelun rakenne ja konfiguraatiot [4]

Suunnittelu ei voi koskaan viitata toiseen suunnitteluun vaan se on aina toteutuksen ylin taso. Suunnittelu voidaan kuitenkin sitoa komponentin sisään. Tämä mahdollistaa alihierarkioiden luomisen. IP-XACT objektien välisiä suhteita ja interaktiota on selvennetty *kuvan 2.4* avulla. Kuvassa näkyvät generaattoriketjut ovat apuohjelmia tai -skriptejä, joilla suunnittelua voi helpottaa. Niiden avulla voi esimerkiksi asettaa parametrejä automaattisesti, ajaa tarkistuksia tai luoda otsikkotiedostoja ohjelmoijia varten.



Kuva 2.4: IP-XACT objektien välinen interaktio [3]

IP-XACTia tukevat suunnitteluohjelmistot pitävät pääsääntöisesti huolen objektien oikeaoppisesta linkityksestä. Sen tunteminen saattaa kuitenkin auttaa järjestelmän suunnitteluvaiheessa välttämään pahimpia suunnitteluvirheitä. Standardi sallii valmistajakohtaisten lisäysten (engl. Vendor extension) tekemisen XML-tiedostoihin. Ne helpottavat tiettyjen asioiden toteuttamista mutta aiheuttavat ongelmia yhteensopivuudessa.

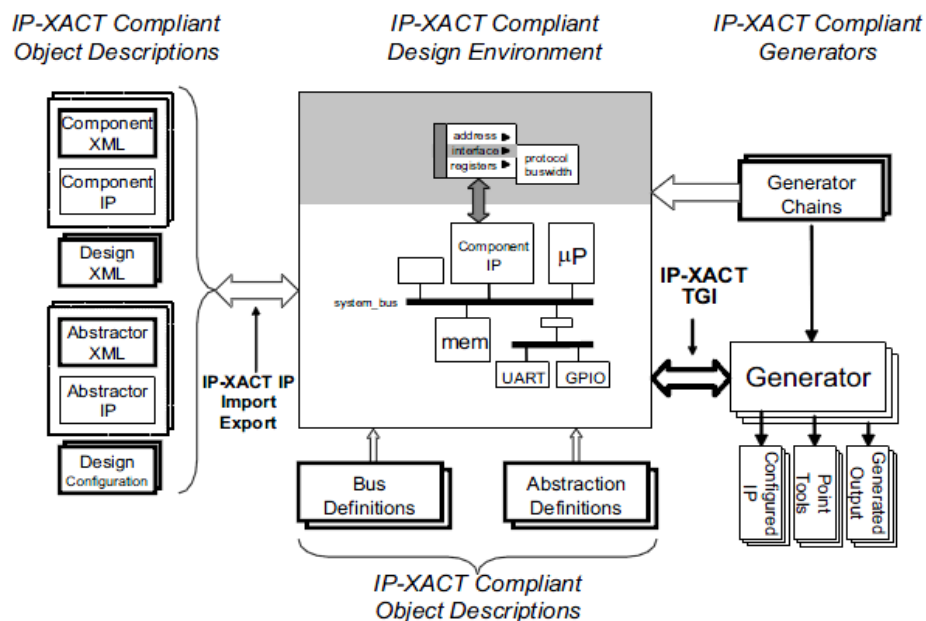
2.2.2 Käyttö

IP-XACT järjestelmien suunnitteluvuo on hyvin samankaltainen kuin perinteinen IP-lohkojen suunnitteluvuo [3]. Suunnitteluvuon seurannasta ja toteutuksesta pitää pääsääntöisesti huolen IP-XACT suunnitteluympäristö (IP-XACT Design Environment). Suunnitteluympäristö koordinoi työkalujoukkoa ja IP-lohkoja luomalla ja muokkaamalla metadatatiedostoja. Suunnitteluympäristö mahdollistaa IP-XACT muotoisten IP-lohkojen toteuttamisen koordinoitun front-end ohjelmiston ja IP-tietokannan avulla. Se mahdollistaa metadatatiedostojen luomisen ja hallitsemisen sekä

tunnistaa yleensä siihen tuotujen IP-lohkojen hierarkian ja konfiguraation. IP-XACT standardi ei aseta rajoituksia suunnitteluympäristön arkkitehtuuriin tai toteutukseen.

Ollakseen kuitenkin IP-XACT version 1.5 mukainen on suunnitteluympäristön mahdollistettava IP-XACT standardin mukaisten IP-lohkojen tuominen ja vienti sovelluksesta niin komponenteille ja suunnitteluille. Lisäksi ympäristön on tuettava TGI-rajapintaa (engl. tight generator interface) pystyäkseen kommunikoimaan ulkoisten generaattoreiden kanssa. Eri IP-XACT versioiden välinen yhteensopivuus mahdollistetaan kuvausten välisellä XSLT-muunnoksella (engl. XSL Transform). Kuvassa 2.5 on esitetty standardin mukaisen suunnitteluympäristön lohkokkaavio. IP-XACT-spesifiset rajapinnat ja formaatit on kuvassa lihavoitu.

Kuvassa vasemmalla näkyy IP-XACT yhteensopivat objektit, eli komponenttien XML-kuvaukset, suunnittelun XML-kuvaus ja konfiguraatiodiedot sekä abstraktorin XML-kuvaus. Komponentin ja abstraktorin XML-kuvaus paketoit sisäänsä varsinaisen toteutuksen kyseisestä objektista. Näitä elementtejä voidaan tuoda ja viedä kirjastoon suunnitteluympäristöstä (engl. Design environment). Suunnitteluympäristö käyttää IP-XACT TGI rajapintaa keskustellessaan generaattoreiden kanssa. Generaattorit puolestaan hoitavat suunnittelun sovittamisen kohdepiirille ja alustalle.



Kuva 2.5: IP-XACT suunnitteluympäristö [3]

Esimerkkinä määrittelyn täyttävästä suunnitteluympäristöstä toimii Kactus 2, jota tässä kandidaatintyössä käsitellään. Vaatimusten mukaiset työkalut luokitellaan *IP-XACT Enabled*-luokkaan.

2.2.3 Tulevaisuus

IP-XACT on menetelmänä suhteellisen uusi ja sitä tukevia työkaluja on tarjolla kohtalaisen vähän. Työkalujen puute ja toisaalta standardin käyttöönottamisen haastavuus [9] ovat hidastaneet menetelmän leviämistä. Standardin versio 1.5 keskittyi kuitenkin korjaamaan pahimmat ongelmakohdat [3] ja suurin osa alan yrityksistä tähtäsikin sen käyttöönottoon.

Menetelmän hyödyt ja sen tuoma ajansäästö on todistettu monessa projektissa. *Taulukko 2.1* kuvaa erään monimutkaisen järjestelmän suunnittelutyöhön kulunutta aikaa ilman IP-XACTin hyödyntämistä ja menetelmän käyttöönoton jälkeen. IP-XACTista johtuva hyöty on selvästi nähtävissä [10]. Suunnitteluun käytetty aika putosi huomattavasti ja tämän myötä projektin kustannukset tippuivat huomattavasti.

Taulukko 2.1: Suunnittelutyön ajankäyttö IP-XACTin kanssa ja ilman [10]

Tehtävä	Käytetty aika [vrk]	Käytetty aika kun IP-XACT on otettu käyttöön [vrk]
Väylämäärittelyt ja IP-kuvaus	30	1
Datan valmistelu	150	7
Kokoaminen	15	0
Yhteensä	195	8

Järjestelmien monimutkaistuessa IP-XACTin avulla saavutettu ajansäästö kasvaa merkittäväksi ja sen yleistymisen teollisuudessa on erittäin todennäköistä.

2.3 Muut samankaltaiset tekniikat

Tässä aliluvussa esitellään lyhyesti muita suunnittelun abstraktointiin tähtääviä tekniikoita. Tekniikat eivät suoraan kilpaile IP-XACT standardin kanssa vaan ne kaikki on suunniteltu hieman erilaisiin tarkoituksiin. Parhaimmillaan tekniikat täydentävät saumattomasti toisiaan.

2.3.1 SystemC

SystemC on C++-kielen luokkakirjasto, jonka avulla on mahdollista luoda tarkkoja malleja ohjelmisto- ja rautatason algoritmeista [11]. SystemC:n avulla on mahdollista luoda nopeasti järjestelmätason malleja joita käytetään validointiin ja optimointiin.

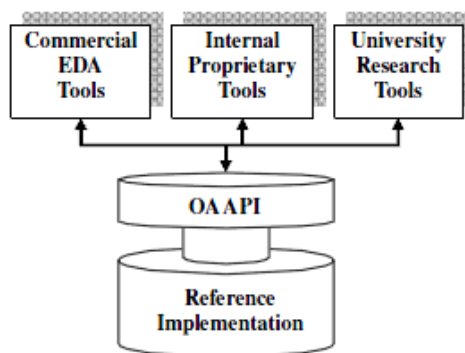
Järjestelmätason malli on käytännössä C++ ohjelma, joka vastaa toiminnaltaan laitteistolohkoa.

SystemC mahdollistaa IP-XACTin tavoin rauta- ja ohjelmistotason yhtäaikaisen suunnittelun ja tukee useita abstraktiotasoja. SystemC yhdistää korkean tason ohjelmointikielen ilmaisuvoiman ja oliopohjaisuuden perinteisten laitteistokuvauskielten tarkkaan laitteiston mallinnukseen. Kielellä tuotettu koodi ei kuitenkaan syntesoidu kovin tehokkaasti, joten sen suurin käyttökohde on simuloinnissa. Ohjelmisto- ja laitteistototeutus on helppo simuloida yhdessä SystemC:n avulla.

2.3.2 OpenAccess API

Open Access on EDA-tietokanta (engl. electronic design automation), joka on suunniteltu mahdollistamaan suunnitteluohjelmistojen välinen yhteensopivuus Open Access API:n kautta [12]. Open Access tähtää samanlaiseen datamallien väliseen yhteensopivuuteen kuin IP-XACT. Open Access tietokanta tallentaa datan omassa muodossaan. Tietokantaan pääsee käsiksi tarkasti määritellyn rajapinnan kautta.

OpenAccess pyrkii tehostamaan tietojen siirtämistä suunnitteluohjelmistosta toiseen juuri tämän rajapinnan avulla. Suunnitteludatan lukeminen tietokannasta on tehokkaampaa kuin tiedostomuodon muuttaminen suunnitteluohjelman tukemaan muotoon. Kuva 2.7 esittää OpenAccess tietokannan sisäisen rakenteen ja sen käyttämisen rajapinnan läpi. Tietokantaan on tallennettu referenssitoteutus lohkoista ja sitä on mahdollista käyttää OpenAccessAPI:n kautta.



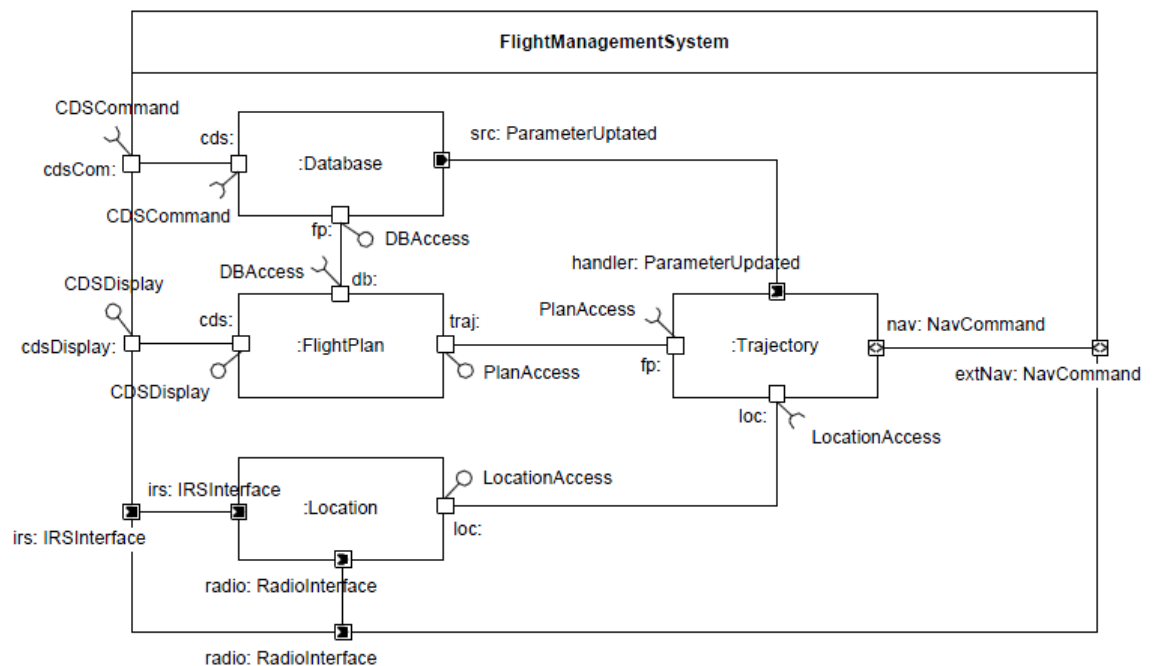
Kuva 2.7: OpenAccess tietokannan toiminta rajapinnan läpi [12]

Open Access on yli 30 yrityksen yhteisprojekti jota koordinoi SI2 eli Silicon Integration Initiative. Se on myös laajasti käytössä yrityksissä ja sitä kehitetään vauhdilla.

2.3.3 UML Profile for Modeling and Analysis of Real-Time and Embedded systems (MARTE)

Modeling and Analysis of Real-Time and Embedded systems eli MARTE on UML-pohjainen (engl. Unified Modelling Language) standardi laitteistotason järjestelmien kuvaamiseen [13]. MARTE tukee spesifikaatio, suunnittelu ja verifiointivaiheiden kuvausta. Se mahdollistaa aiemmin esiteltyjen tekniikoiden tavoin suunnittelutyökalujen välisen interaktion tallettamalla suunnitteludatan UML-muotoon.

MARTEn avulla on mahdollista mallintaa reaaliaikaisten ja sulautettujen järjestelmien toimintaa ja kuvata järjestelmiä alustariippumattomasti. Pääpiirteissään idea on samankaltainen kuin IP-XACT standardissa. Myös MARTEn avulla onnistuu ohjelmiston ja rautalohkojen samanaikainen suunnittelu ja toteutus. Standardi koostuu neljästä pääpaketista, joita ovat *design model package*, *analysis model package*, *MARTE foundations package* ja *annexes profile package*. Esimerkki MARTElla luodusta komponentista on esitetty kuvassa 2.8. Kuvasta näkyy hyvin MARTEn hierarkisuus.



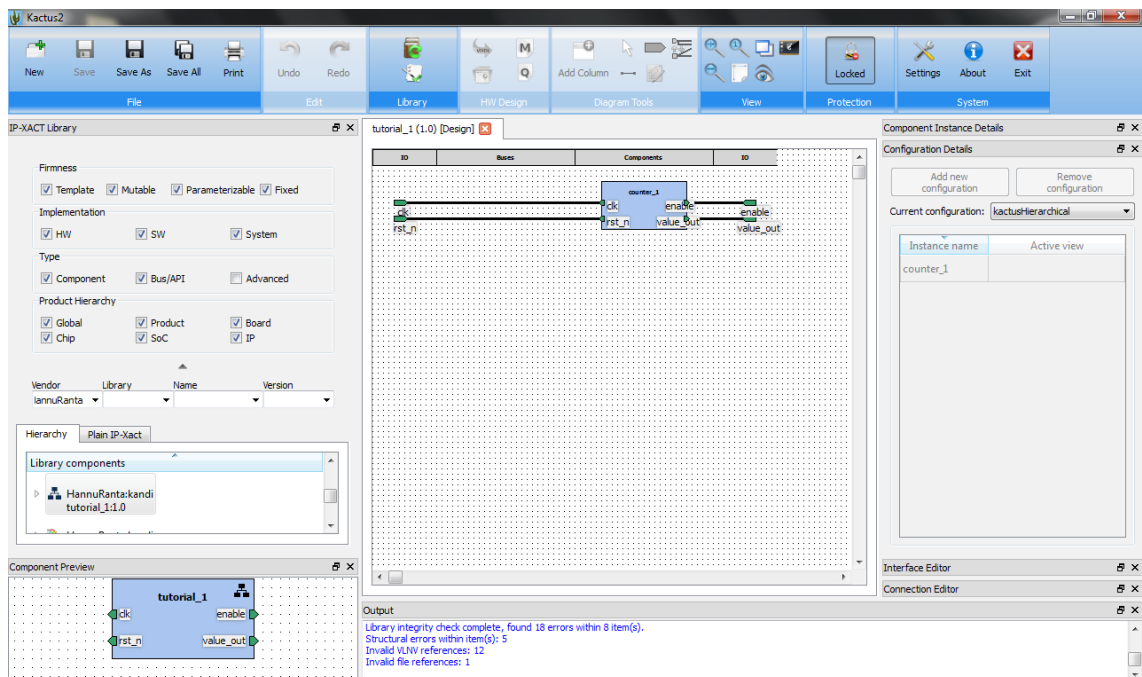
Kuva 2.8: Esimerkki MARTEn avulla luodusta lohkokaaviosta [13]

MARTE on monipuolinen standardi, joka mahdollistaa IP-XACTia monipuolisemman toiminnallisuuden. Standardit kuitenkin täydentävät luontevasti toisiaan ja niitä on mahdollista hyödyntää rinnakkain.

3 KACTUS 2

Kactus 2 on Tampereen teknillisen yliopiston Tietokonetekniikan laitoksella kehitetty IP-XACT standardia hyödyntävä suunnitteluohjelma, joka on ensisijaisesti suunnattu FPGA-pohjaisten MP-SoC järjestelmien suunnitteluun [1] [4]. Kactus 2 laajentaa IP-XACTin XML-paketoinnin myös ohjelmistokomponentteihin, mutta tässä kandidaatintyössä tarkastellaan vain alkuperäisessä standardissa määriteltyä, laitteistokomponentteihin keskittyä osaa.

Kuvassa 3.1 on esitetty Kactus 2 ohjelman päänäkökulma kun suunnittelu on avattuna. Kuvankaappaus on otettu Kactusin versiosta 1.2. Kuvassa vasemmalla näkyy kirjaston hallintaan liittyvät toiminnot ja keskellä näkyy auki oleva suunnittelu graafisena esityksenä. Alhaalla on komentotulkki ja oikealla näkymien hallintaan liittyvät toiminnot. Ylinä näkyy ribbon-tyyppinen valikkopalkki.



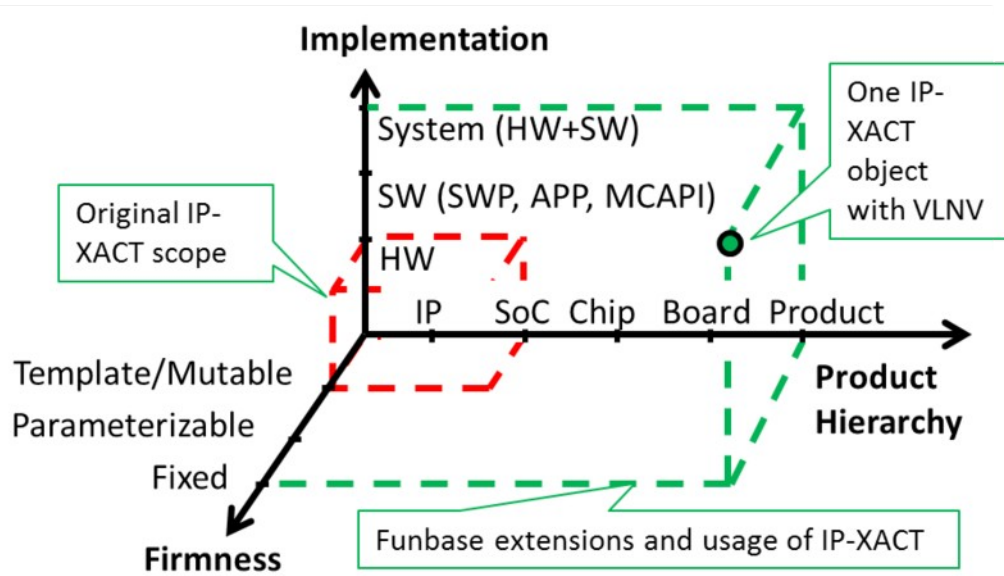
Kuva 3.1: Kactus 2:n päänäkökulma

Kuten *kuvasta 3.1* näkyy on Kactusin ulkoasu pyritty pitämään mahdollisimman selkeänä ja funktionaalisena. Ohjelmiston suunnittelulähtökohtana on tarjota mahdollisimman helppo sisäänpääsy metadata pohjaiseen suunnitteluun ja IP-lohkojen

uudelleenkäyttöön [4]. Lisäksi tarkoituksena on tarjota ohjelmistokehittäjille aiempaa helpompi lähestymistapa FPGA-pohjaiseen työskentelyyn. Ohjelma on toteutettu C++:lla ja Qt:lla. Se on julkaistu GPL2-lisenssin alla ja ladattavissa Sourceforgesta [14].

3.1 IP-XACTin suunnitteluvuo ja Kactus 2

Suunnittelussa hyödynnetään *kuvassa 2.1* esiteltyä suunnitteluvuota. Kactus 2 laajentaa alkuperäistä IP-XACTin metadataformaattia *kuvan 3.2* esittämällä tavalla.

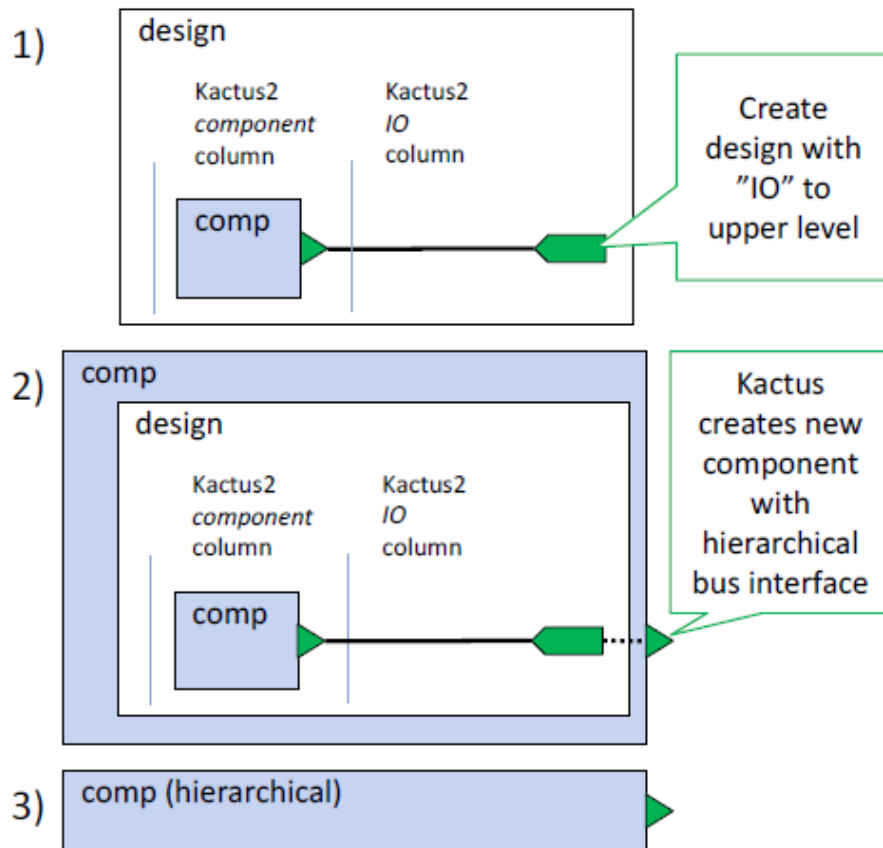


Kuva 3.2: Kactus 2:n metadatatamalli [4]

Laajennus on merkitty kuvaan vihreällä. Tässä kandidaatintyössä keskitytään kuitenkin vain alkuperäisen standardin mukaiseen, kuvassa punaisella merkittyyn osaan. *Kuvassa 1.2* nähtiin, miten Kactus 2 sijoittuu suunnitteluvuohon suhteessa muihin työkaluihin.

Hardware-lohkojen osalta Kactus noudattaa varsin tarkasti *luvussa 2* kuvattua IP-XACT standardin mukaista tapaa käsitellä metadataa. Esimerkki Kaktuksen tuottamasta metadatatiedostosta on *liitteessä 2*. Liite on puumuotoinen kuvaus XML-tiedostosta ja siitä on selkeästi erotettavissa VLNV-tiedot, rajapintatiedot ja muut IP-XACT standardissa määritellyt asiat.

IP-XACT kirjaston hallinta ja lohkojen kapseloiminen Kactusin avulla on varsin helppoa ja suoraviivaista. Kactus automatisoi suunnitteluprosessia monin tavoin. Jos esimerkiksi väylä yhdistetään komponenttiin, luodaan väylän pohjalta automaattisesti määrittely portille komponentin IP-XACT tietoihin. Tämä esimerkki on kuvattu tarkemmin *kuvassa 3.3*.

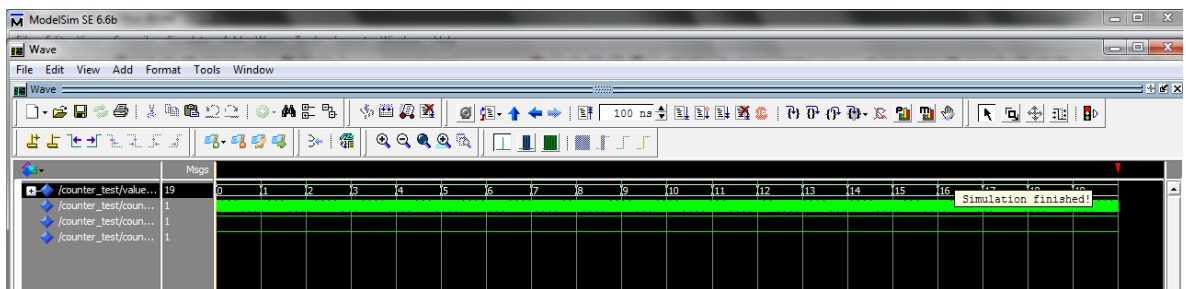


Kuva 3.3: Esimerkki Kactusin suunnittelua helpottavasta automaatiosta [4]

Kactus tukee sekä bottom-up että top-down tyylisiä suunnittelua ja sen automaattiset generointitoiminnot auttavat molemmissa suunnittelutyyleissä.

4 TULOKSET

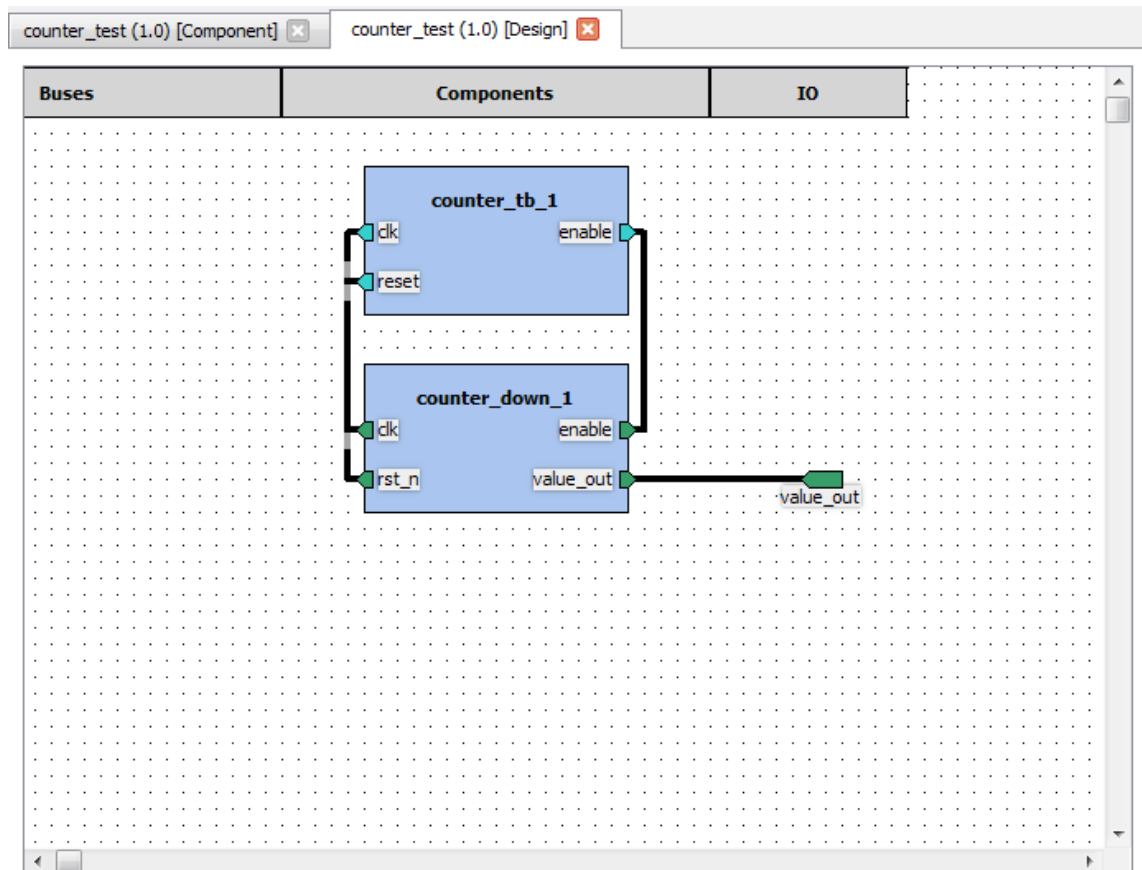
Kandidaatintyön tuloksena toteutettiin kaksi tutoriaalia, jotka kaikki löytyvät *liitteestä 3*. Ensimmäinen tutoriaali kertoo IP-lohkon käyttöönotosta Kactus 2 ohjelman avulla ja opettaa ohjelman käytön perusteet. Tutoriaali on jaettu kahteen osaan, joista ensimmäisessä otetaan lohko käyttöön ja syntesoidaan se IP-laudalle ja toisessa testataan lohko modelsimin avulla. *Kuvassa 4.1* on esitetty kuvankaappaus Modelsimin aaltomuotoikkunasta.



Kuva 4.1: Tutoriaalin lohko simuloituna Modelsimissa

Tutoriaalia varten toteutettiin testipenkki jolla lohkon simulointi on suoraviivaista. Samaa testipenkkiä voi käyttää myös toisessa tutoriaalissa käyttöön otettavan lohkon simulointiin.

Kuvassa 4.2 on nähtävissä ensimmäisen tutoriaalin lohko testipenkkiin yhdistettynä. Kyseisen lohkon VHDL-kuvaus on esitetty liitteessä 1 ja IP-XACT metadatatiedoston puunäkymä liitteessä 2. Liitteen 2 puunäkymästä on helppo havaita esimerkiksi nimitiedot (*name*, *vendor*, *library*, *version*), väylän hierarkia ja rakenne (*bus interfaces*) sekä tiedostojen (*file sets*) tiedot.



Kuva 4.2: Ensimmäisen tutoriaalin laskuri ja testipenkki Kactusessa

Toisessa tutoriaalissa opastetaan oman IP-XACT lohkon luominen olemassa olevien VHDL-tiedostojen pohjalta. Tutoriaalin tarkoituksena on opastaa Kactusin monimutkaisempien toimintojen pariin ja kertoa IP-XACT standardin perusteet. Tutoriaalia varten annetaan valmiina alaspäin laskevan laskurin VHDL-tiedosto, joka tuodaan Kactuseseen ja lisätään IP-XACT komponenttiin. Tätä lohkoa on mahdollista simuloida samalla valmiilla testipenkillä kuin ensimmäisen tutoriaalin lohkoa. Tutoriaalin pääasiallinen opittava asia on IP-XACT komponentin ja sen väylämäärittelyiden luominen Kactus 2:ssa ja IP-XACT standardin perusteet.

4.1 Tutoriaalien käyttäjätestit

Tutoriaalit testattiin viidellä koehenkilöllä, joilla oletettiin olevan perustiedot digitaalitekniikasta. Koehenkilöiden löytäminen osoittautui lopputuloksena suhteellisen vaikeaksi, mutta kovan etsinnän jälkeen henkilöt löytyivät. Testit dokumentoitiin observoimalla testihenkilöiden suoriutumista tutoriaalien teon aikana ja lopuksi kysymällä koehenkilöiden tuntemuksia. Testihenkilöt käyttivät testeihin samaa tietokonetta, johon oli asennettu kaikki tarvittavat ohjelmistot. Kaikki koehenkilöt suoriutuivat tutoriaaleista, mutta osan kohdalla aika-arvio meni suhteellisen paljon pieleen ja varsinkin toisen tutoriaalin suorittamiseen kului huomattavasti arvioitua enemmän aikaa. Tutoriaalit olivat laadultaan ja selkeydeltään testihenkilöiden mielestä hyviä. Suoritusajalla oli suora korrelaatio henkilöiden digitaalitekniikan tietämykseen. Eniten digitaalitekniikan kursseja lukenut testihenkilö suoriutui tutoriaaleista nopeiten ja ymmärsi myös parhaiten IP-XACT standardin ajatuksen tutoriaalien jälkeen.

Taulukkoon 4.1 on koottu tutoriaalien suorittamiseen kulunut aika jokaisen testihenkilön kohdalla. Lisäksi taulukkoon on merkitty joitain observoinnin aikana syntyneitä huomioita.

Taulukko 4.1: Testihenkilöiden käyttämä aika

Henkilö	Käytetty aika [minuuttia]	Huomioita
1	60	-
2	40	Eniten taustatietoa
3	65	-
4	55	-
5	50	Kehittyi nopeasti

Testihenkilöille tehtyjen loppukysymysten perusteella Kactuksen perusominaisuuksien hallinta oli kaikilla testatuilla hyvin hallussa. IP-XACT standardin perusidean sen sijaan ymmärsi vain kaksi viidestä testatusta. Tämä osoittaa Kactuksen kyvyn piilottaa varsinaisen standardin monimutkaisuus käyttäjältä. Myös standardin vaikeus ja monimutkaisuus varmasti vaikeuttaa sen ymmärtämistä. Jos tutoriaalisarjaa kuitenkin laajennetaan tulevaisuudessa on sen avulla helppo perehdyttää IP-XACTin pariin. Asian oppii kaikkien testihenkilöiden mielestä selkeästi paremmin käytännön kautta kun IP-XACTin yhteyden esimerkiksi annettuihin VHDL-tiedostoihin ymmärtää.

Kaiken kaikkiaan tutoriaalit toimivat erittäin hyvin ja testatut henkilöt uskoivat oppineensa Kaktuksen käytön perusteet niiden avulla. Jatkokehityksenä tutoriaaleissa olisi voinut painottaa enemmän IP-XACT metadatatiedoston ja VHDL-tiedoston yhtäläisyyksiä. Varsinkin valmiiksi generoidun VHDL-tiedoston muokkaaminen käsin saattaa joissain tapauksissa tulla kyseeseen ja tällöin olisi hyvä ymmärtää yhteys Kaktuksen näyttämään visuaaliseen esitykseen.

5 YHTEENVETO

IP-XACTin tulevaisuus vaikuttaa lupaavalta. Monimutkaisia SoC-järjestelmiä ei käytännössä pysty nykyään toteuttamaan ilman IP-lohkojen uudelleenkäyttöä. Uudelleenkäyttöä on kuitenkin aiemmin vaikeuttanut laitteistorajoitteet. IP-XACT mahdollistaa laitteistoriippumattoman toteutustavan.

Standardi on vielä suhteellisen uusi, eikä sitä ole toistaiseksi laajemmin hyödynnetty teollisuudessa. Standardista ja sen käyttöönotosta kiinnostuneita yrityksiä [9] on kuitenkin useita, joten tulevaisuudessa IP-XACT pohjainen suunnittelu tulee varmasti yleistymään.

Kactus 2 hyödyntää lähes kaikkia IP-XACTin ominaisuuksia ja laajentaa samaa ajatusta lisäksi ohjelmistopuolelle. Toistaiseksi Kactuksella on varsin vähän kilpailijoita markkinoilla ja sen avoimuus tekee siitä kiinnostavan vaihtoehdon. IP-lohkojen suunnittelu Kactusen avulla on suoraviivaista ja alkuvaikeuksien jälkeen suhteellisen helppoa.

IP-XACTin esittelemä suunnittelumalli ja sen toteuttaminen Kactusen avulla antaa hyvän käsityksen uudelleenkäytön eduista ja erityisesti sen helppoudesta IP-XACTin tarjoamien ominaisuuksien avulla. Teoria standardin taustalla on suhteellisen monimutkainen, mutta järjestelmän toiminnan ymmärtäminen auttaa sen käyttämistä suunnittelussa. Suurimmaksi IP-XACTin levinneisyyttä rajoittaneeksi tekijäksi onkin nähty juuri monimutkaisuus [9]. Toisaalta esimerkiksi Kactusista käytettäessä ei IP-XACTin toimintaa taustalla juuri huomaa.

Toteutettujen tutoriaalien avulla IP-XACT standardiin ja Kactus 2:n käyttöön pääsi nopeasti sisään, mutta IP-XACTin idea ei auennut perusteita pidemmälle, sillä kyseessä on erittäin monimutkainen standardi jonka opetteluun kuluu aikaa. Tutoriaalit opettavat kuitenkin perusajatuksat ja kaikki koehenkilöt joilla tutoriaalit testattiin osasivat selittää ne. Myös Kactus 2 ohjelman käyttäminen luonnistui tutoriaalien jälkeen loistavasti.

Tutoriaalien jatkokehityksessä on hyvä painottaa enemmän IP-XACT metadatatiedoston yhteyttä VHDL-tiedostoihin, sillä jos valmiiksi generoituja VHDL-tiedostoja joutuu editoimaan käsin on yhteys hyvä tuntee. Nämä tutoriaalit opettavat

kuitenkin perusteen Kactus 2:n käytöstä ja IP-XACTista joten niiden päälle on hyvä rakentaa uusia ja pidemmälle meneviä tutoriaaleja.

LÄHTEET

- [1] A. Kamppi, L. Matilainen, J-M. Määttä, E. Salminen, T D. Hämäläinen, M. Hännikäinen, ”Kactus2: Environment for Embedded Product Development Using IP-XACT and MCAP1” 14th Euromicro Conference on Digital System Design, Oulu, 31.8-2.9.2011, IEEE Computer Society s. 262-265
- [2] NVidia. ”Bringing High-End Graphics to Handheld Devices” Whitepaper, 2011, 28 s. Saatavilla: http://www.nvidia.com/content/PDF/tegra_white_papers/Bringing_High-End_Graphics_to_Handheld_Devices.pdf
- [3] IEEE-1685, ”IEEE Standard for IP-XACT, Standard Structure for Packaging, Integrating, and Reusing IP within Tool Flows” USA 2009, IEEE 374 s.
- [4] A. Kamppi, L. Matilainen, J-M. Määttä, E. Salminen, T D. Hämäläinen, ”Kactus 2: IP-XACT/IEEE1685 Compatible Design Environment for Embedded Multiprocessor System-on-Chip Products”, Tampere 2011, Report 37, 48s.
- [5] Tampereen teknillinen yliopisto, ”FPGA-kehitysalustan lainaaminen” [Viitattu 1.12.2011] [Päivitetty 1/2011] Saatavilla: http://www.tkt.cs.tut.fi/Opetus/Fpga_board/
- [6] M. Keating, P Bricaud, ”Reuse Methodology Manual for System-on-a-Chip Designs” Third Edition, USA 2002, Kluwer Academic Publishers, 293s.
- [7] C. Kwanghyun, K. Jaebeom, J. Euibong, K. Sik, L. Zhenmin, C. Young-Rae, M. Byeong, C. Kyu-Myung, D. Nongseo, G. Giheung, S. Yongin, D. Gyeonggi, ”Reusable Platform Design Methodology For SoC Integration And Verification” International SoC Design Conference 2008, Korea 24-25.11.2008, IEEE 2008 s. 78-81
- [8] T. Arpinen, T. Koskinen, E. Salminen, T. Hämäläinen, M. Hännikäinen, ”Evaluating UML2 Modeling of IP-XACT Objects for Automatic MP-SoC Integration onto FPGA” Tampere 2009, 6s.

- [9] D. Murray, "IP-XACT Usage Survey Results" 2010, Survey, 11s
Saatavilla: [http://integrationinsights.typepad.com/pdf/IP-XACT Usage Survey.pdf](http://integrationinsights.typepad.com/pdf/IP-XACT%20Usage%20Survey.pdf)
- [10] A. Berna, A. Sparsh, S. Aygin, S. Topcu, E. Armagan,
"An IP-XACT Deployment Case: IZARN IP", Verkkoartikkeli
[Viitattu 16.1.2012] Saatavilla:
[http://www.design-reuse.com/articles/26224/ip-xact-deployment-
case-izarn-ip.html](http://www.design-reuse.com/articles/26224/ip-xact-deployment-case-izarn-ip.html)
- [11] Synopsys, Inc. "System C version 2.0 User's Guide"
USA 2002, 212 s.
- [12] M. Guiney, E. Leavitt, "An Introduction to OpenAccess:
An Open Source Data Model and API for IC Design"
2006 Asia and South Pacific Design Automation Conference,
IEEE Press, USA 2006, 434 – 436 s.
- [13] Object Management Group, "UML Profile for MARTE: Modeling
and Analysis of Real-Time Embedded Systems"
USA 2011, Object Management Group Inc, 754 s.
- [14] Sourceforge: Kactus 2. Verkkosivu [Viitattu 1.5.2012]
Saatavilla: <http://sourceforge.net/projects/kactus2/>

LIITE 1: COUNTER.VHDL

```

-----
-- Title       : Counter
-- Project     : Kandidaatintyö
-----
-- File        : counter.vhd
-- Author      : Hannu Ranta
-- Company     :
-- Created     : 2011-11-19
-- Last update : 2011-11-24
-- Platform    :
-- Standard    : VHDL'87
-----
-- Description: up-counter
-----
-- Copyright (c) 2011
-----
-- Revisions  :
-- Date       Version  Author  Description
-- 2011-11-19  1.0      Hannu   Created
-----

-- Kirjastot
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

-----
-- ENTITY
-----
entity counter is

    generic (
        width_g : integer := 20);           -- counter width

    port (
        clk      : in  std_logic;           -- clk
        rst_n    : in  std_logic;           -- reset_n
        enable    : in  std_logic;           -- enable
        value_out : out std_logic_vector(width_g - 1 downto 0));
        --value out

end counter;

```

```
-----  
-- ARCHITECHTURE  
-----
```

```
architecture rtl of counter is
```

```
    signal internal_count_r : signed(width_g - 1 downto 0);  
    -- internal signal for result
```

```
    signal clk_count : integer;  
    -- internal signal for clk
```

```
    constant clk_cycle_c : integer := 50000;  
    -- clk-cycle
```

```
begin -- rtl
```

```
    -- purpose: counter process  
    -- type   : sequential  
    -- inputs : clk, rst_n  
    -- outputs: internal_count
```

```
    counter_pros : process (clk, rst_n)  
    begin -- process counter_pros
```

```
        -- asynchronous reset (active low)
```

```
        if rst_n = '0' then
```

```
            internal_count_r <= (others => '0');  
            clk_count <= 0;
```

```
        elsif clk'event and clk = '1' then -- rising clock edge
```

```
            if clk_count = clk_cycle_c then  
                clk_count <= 0;
```

```
                if enable = '1' then  
                    internal_count_r <= internal_count_r + 1;  
                else  
                    internal_count_r <= internal_count_r;  
                end if;
```

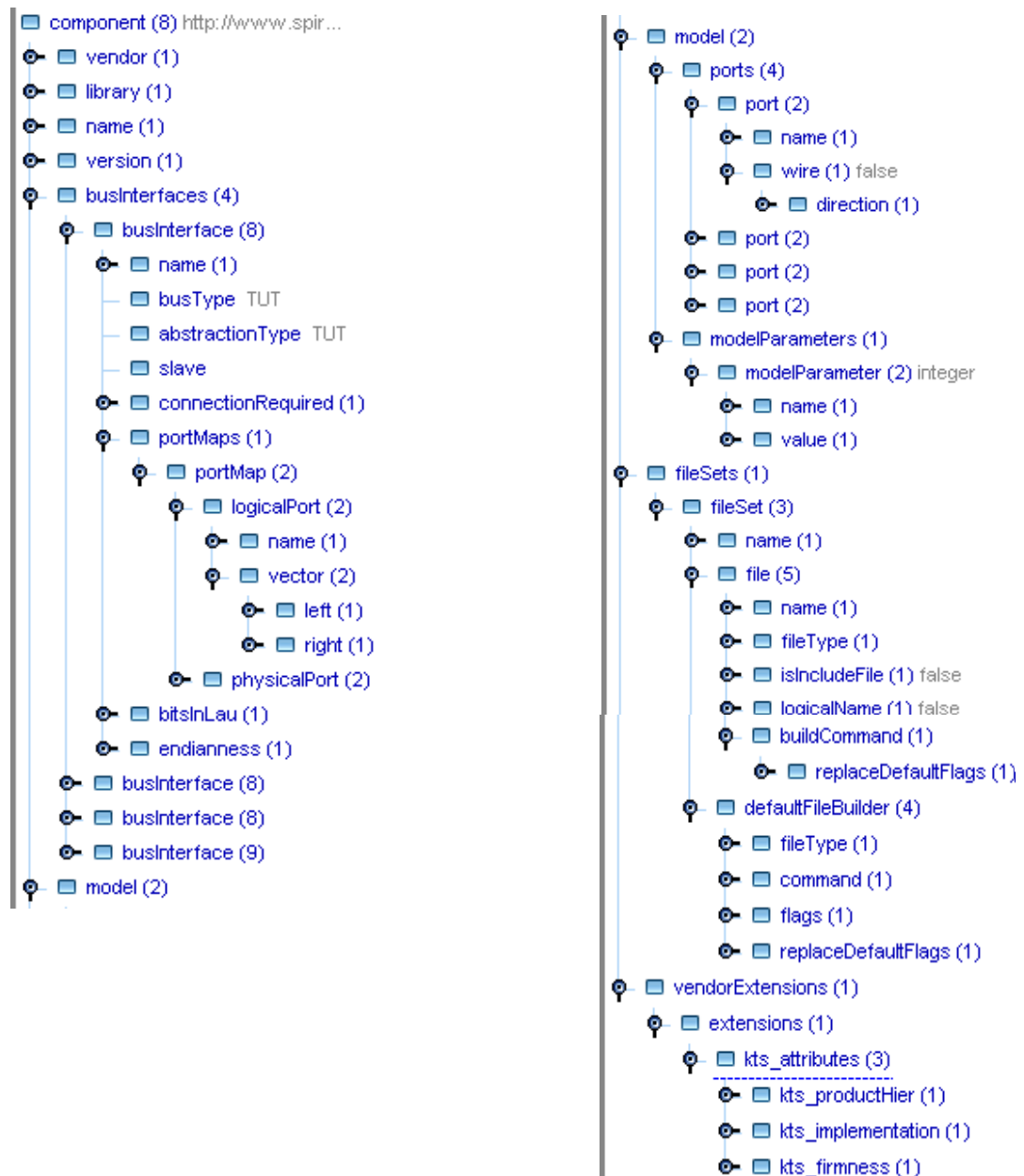
```
            else  
                clk_count <= clk_count + 1;  
            end if;
```

```
        end if;  
    end process counter_pros;
```

```
    value_out <= std_logic_vector(internal_count_r);
```

```
end rtl;
```

LIITE 2: COUNTER.VHDL:N IP-XACT-KUVAUS PUUMUODOSSA



Yllä näkyvässä XML-puussa on kuvattu liitteestä 1 löytyvän laskurin IP-XACT-metadata puumuodossa. Ylimpänä vasemmalla näkyy lohkon VLVN-tiedot. Lohkon malli (model) jakautuu kahteen osaan eli fyysisiin portteihin ja parametreihin. Lohkoon liittyy yksi VHDL-tiedosto jolle on määritetty käännöskomento simulaatiota varten.

Kaktuksen omat laajennokset näkyvät viimeisinä ja niissä on lähinnä lohkon luokittelutietoja ja graafisen näkymän koordinaattitietoja.


LIITE 3: TOTEUTETUT TUTORIAALIT

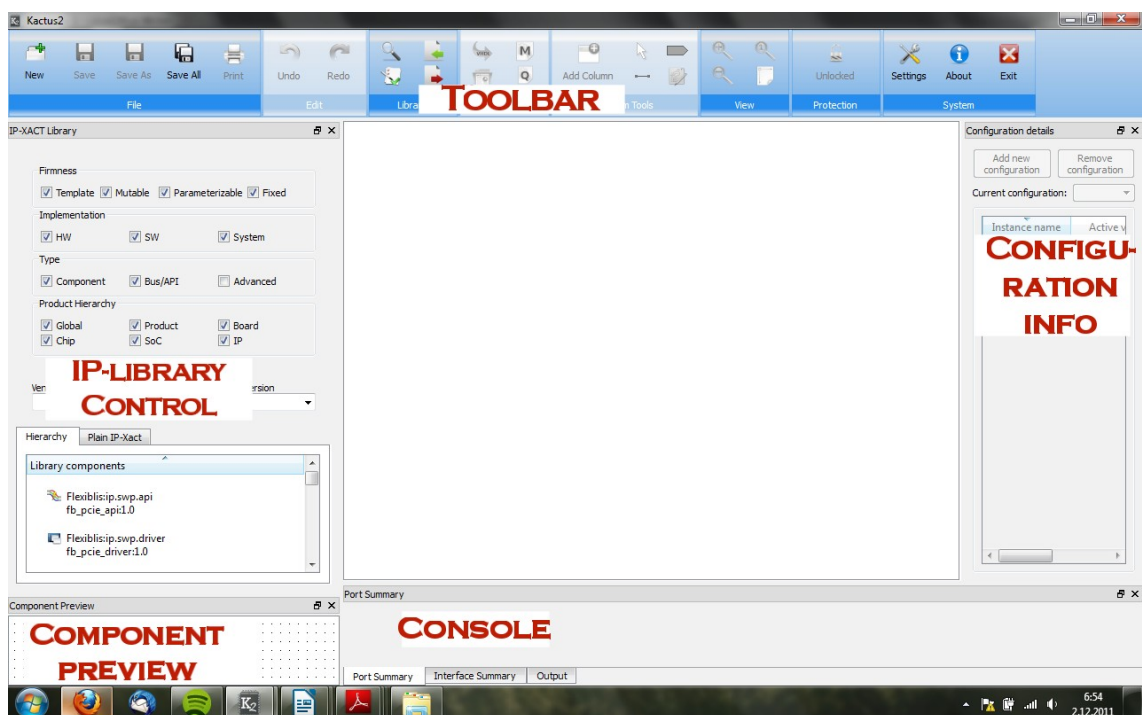
Tutorial 1a: Introduction to Kactus 2 software By Hannu Ranta

In this tutorial you will learn the basic usage of Kactus 2 software and implement a simple IP-block to Altera DE2 development board. To complete this tutorial you need basic knowledge in a field of digital logic. This tutorial will take approximately 10 minutes.

This tutorial has been made for Kactus version 1.1 but it is usable in newer versions also.

1. Overview

To start Kactus 2 click the icon  from Windows Start-menu. Kactus will start and open in a start view shown in *picture 1*.

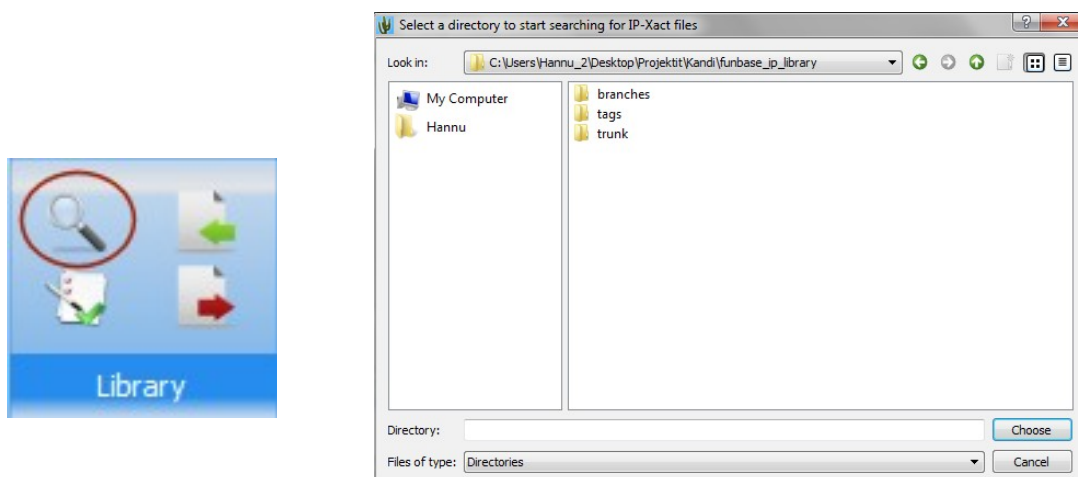


Picture 1: Basic view of Kactus 2

Software is mainly controlled using the toolbar at the top of the screen. IP-library can be managed using the fields in left border. Top of the left you will see the search tools for IP-components and found components are listed in the middle. When you activate component by pressing it the preview will be shown in the preview window down left corner. In the middle of the program is the area where design will be done and at the bottom is the console window for status messages and information. Right border will show information about ports, instances and connections when the design is under construction.

2. Adding components to the library

At first you have to add the IP-XACT files from the disk to program. Choose the magnifier glass icon (Search IP-XACT files) shown in *picture 2*. the file dialog will open. Choose the right folder and press **Choose-button**.

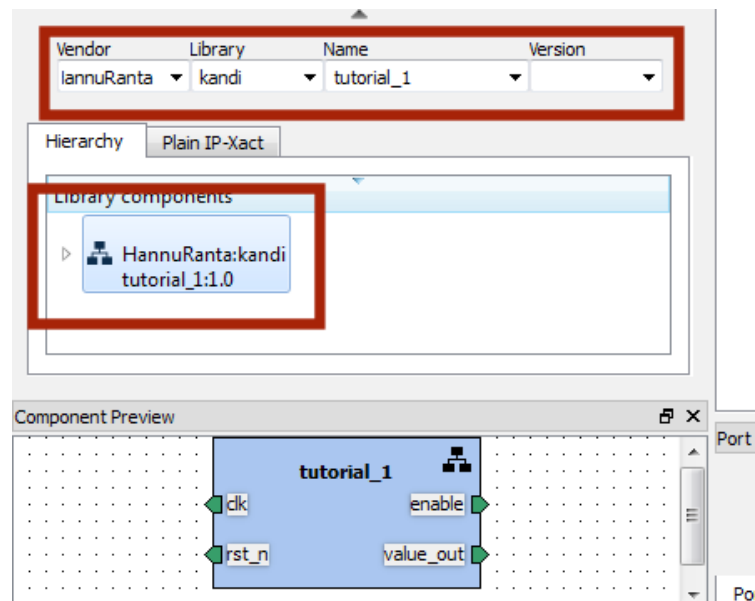


Picture 2: Searching the IP-XACT files

Now you have successfully added the IP-XACT files to the Kactus 2 library and they will be shown in the library manager shown in the left side. You can search IP-blocks using different filters of library manager. This operation is shown in next paragraph.

3. Finding and opening the design

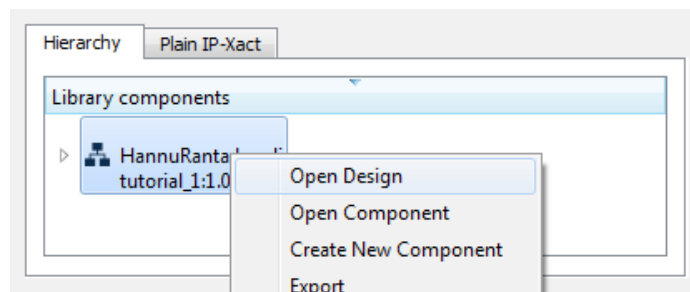
Insert the data shown in *picture 3* to the filtering options of library manager. After that you will find the design used in this tutorial.



Picture 3: Searching the right design

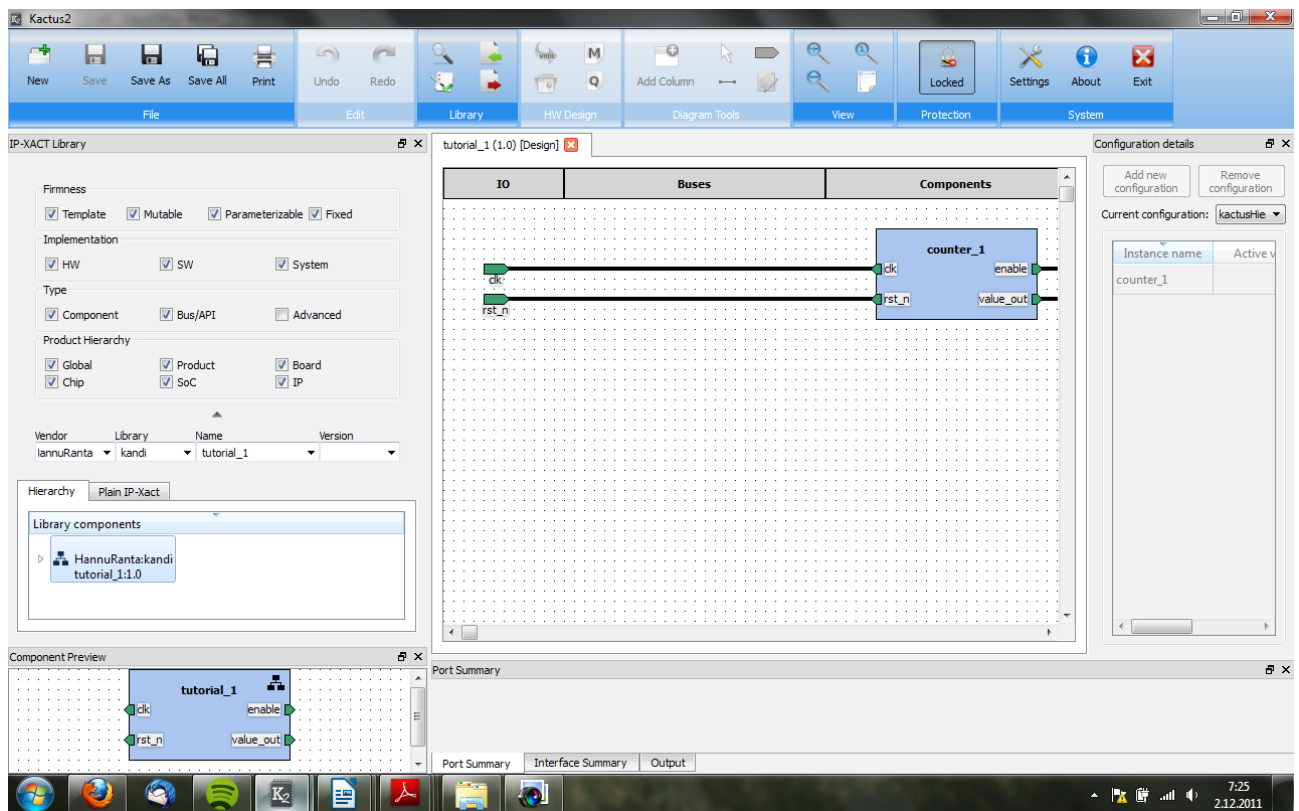
Double-click the design in **Library Components** window to open it in **Component Preview** window.

Right-click on the top of the component and press **Open Design** button as shown in *picture 4*.



Picture 4: Opening the design

Design will be open in the middle of the program as shown in *picture 5*.

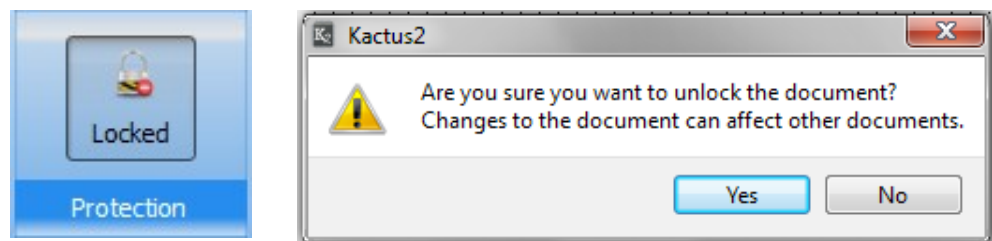


Picture 5: Design opened

Ports, connections and components are shown in their own areas. This will keep the view simple and clean also with complex designs. In this tutorial the design is a simple counter.

4. Generate top-level VHDL and unlock design

Open the component editing lock by pressing the lock icon in a toolbar. Press **YES** when the pop-up window opens. This process is shown in *picture 6*.



Picture 6: Opening the lock

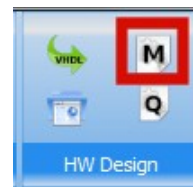
Now you are able to make changes in design. At first generate the top level VHDL-description by clicking **Generate Top-VHDL**-icon (marked green in *picture 7*) and save it. Kactus will ask if you would like to save generated file to IP-XACT metadata. Click **YES**.



Picture 7: Generating top-level VHDL (green square)

5. Generating Modelsim-macro and Quartus-files

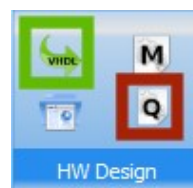
Generating Modelsim-macros directly from Kactus 2 allows designers an easy way to simulate and test their designs. To generate a macro press **Generate ModelSim Makefile** – icon from toolbar. This icon is marked red in *picture 8*. Kactus will ask you a folder where the macro will be saved. Select a folder and press **Save**. After this step Kactus will ask if you want to add the generated macro to IP-XACT metadata. Press **Yes**.



Picture 8: Generating Modelsim-macro

There is a separate testbench to test this counter-block. Full testing process is covered in *tutorial 1b*.

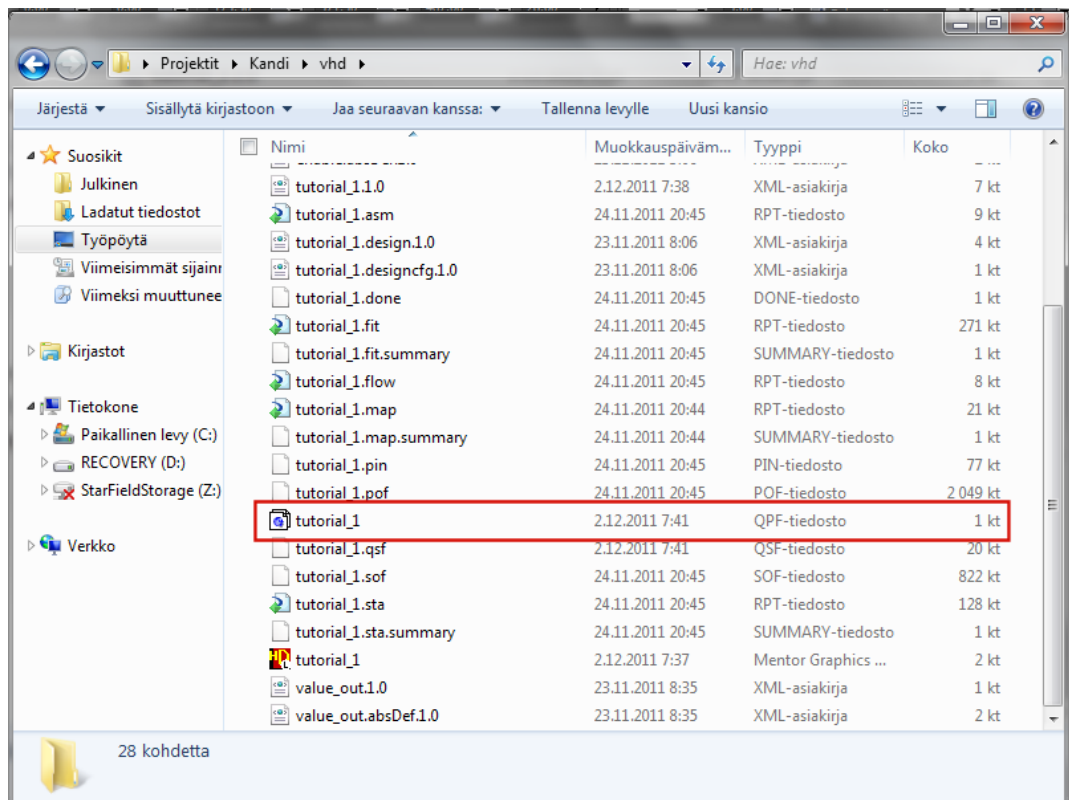
Next you will generate Quartus project files for FPGA-synthezation. To generate Quartus project-file click **Generate Quartus Project**-icon (marked red in *picture 9*). Kactus will ask you a folder where the project is created.



Picture 9: Generating Quartus project file (red square)

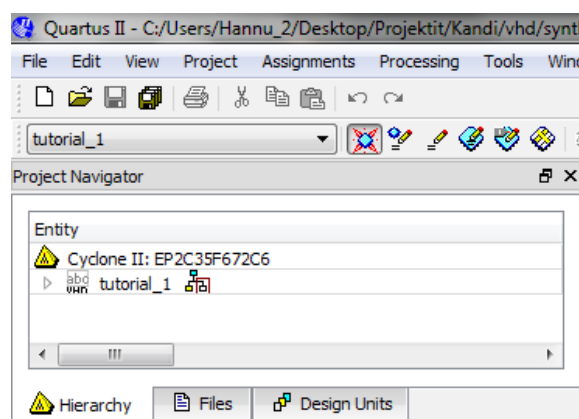
5. Compiling and synthesizing the design

Quartus project based on your design is now generated and you have simulation makro ready for Modelsim. Close Kactus and open a folder where you saved your Quartus project. Open project by double-clicking a project file (QPF-file). Correct file is shown in *picture 10*.



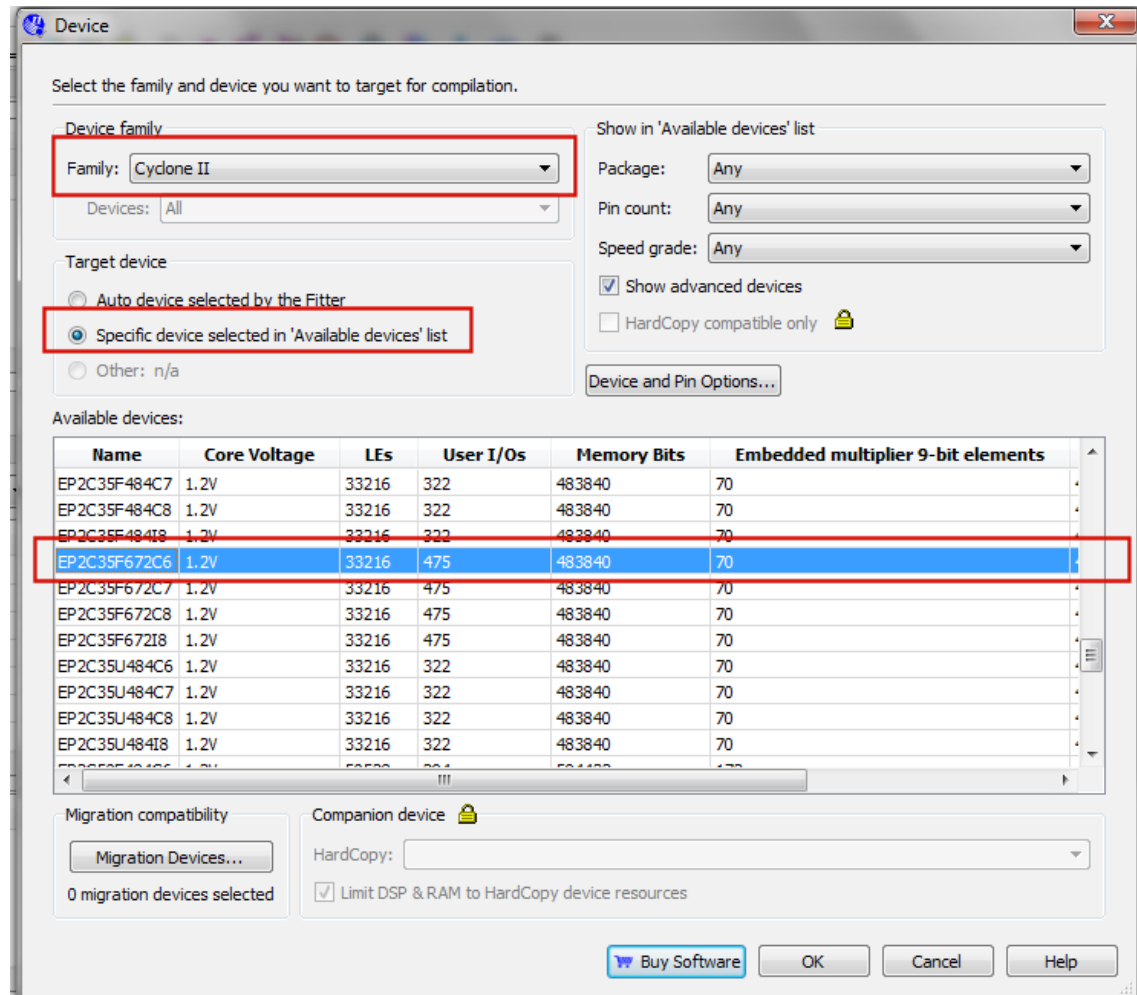
Picture 10: Quartus project file

After this Quartus will open. It might take a while. When Quartus is ready you will see project in a **Project Navigator** window as shown in in *picture 11*.



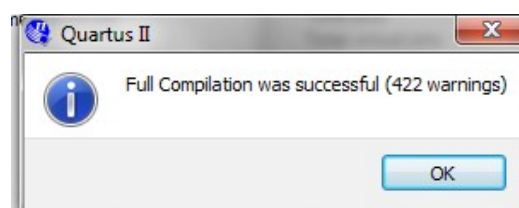
Picture 11: Project Navigator-view

Check that you have a correct circuit (Cyclone II) selected in Quartus. You will find this information from the right side of a yellow pyramid in **Project Navigator** window shown in *picture 11*. If circuit is not correct change settings to match settings in *picture 12*.



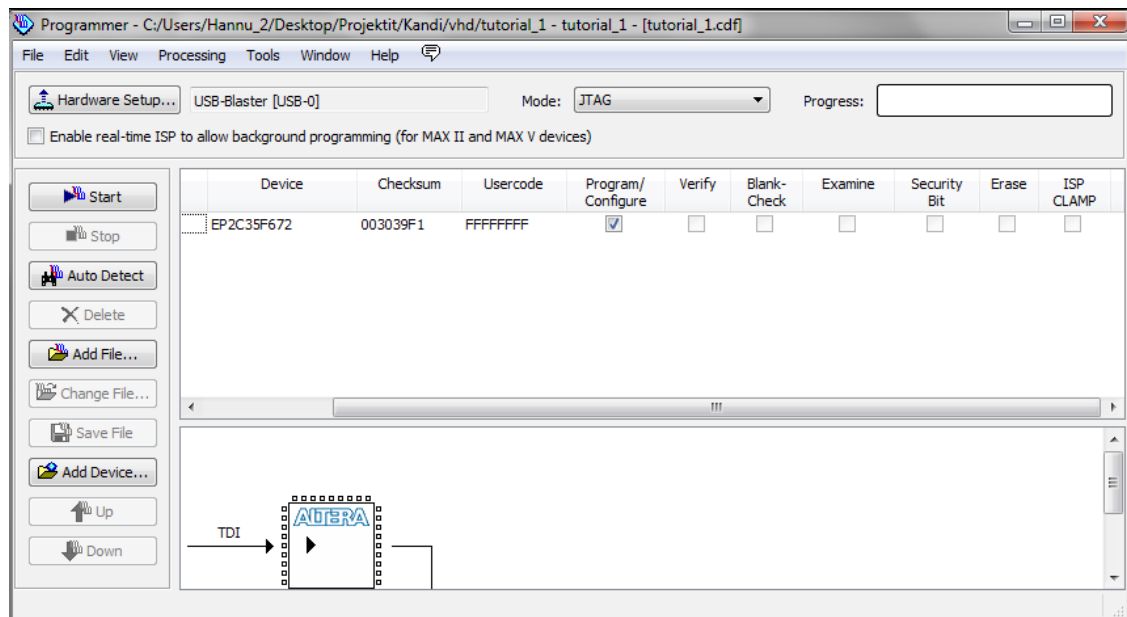
Picture 12: Device-settings of Quartus

Project is now ready for compilation. Start compiling process from menu (**Processing->Start Compilation**). Compiling will take couple of minutes. When compilation is ready Quartus will notify you with dialog shown in *picture 13*. Click **OK** to close dialog. Do not mind about the warnings.



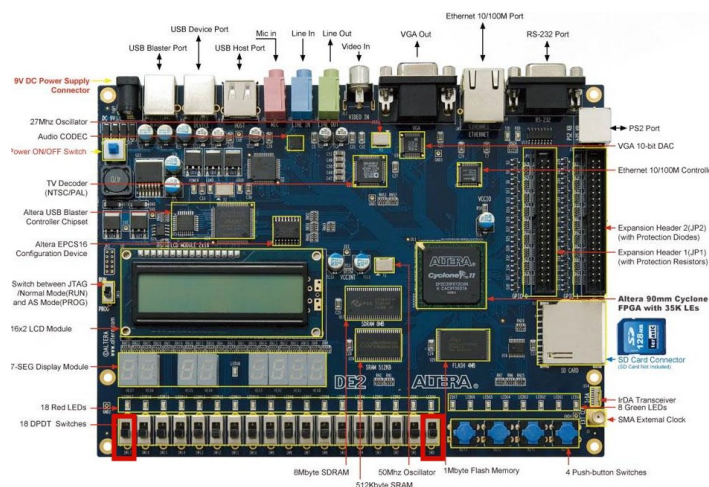
Picture 13: Compilation finished

Project is now ready for FPGA synthezation. Power up development platform and check USB-connection. When everything is ready start compiling software from Quartus (**Tools->Programmer**). Programmer will start. Find correct file using **Add file** button if it is not automatically added (.sof file). Check *Hardware Setup* option. It should be *USB-Blaster*. If there is something else change it by clicking **Hardware Setup** button. Check that *Program/Configure* is selected and that everythig is similar than in *picture 14*.



Picture 14: Programmer settings

When everything is ready click **Start**-button and wait the programming process. When it is ready rise reset-switch (SW17) and enable-switch (SW0) from development platform. You should see red and green LEDs flashing. Correct switches are marked red in *picture 15*.



Picture 15: Reset- and enable-switches (SW17 and SW0)

6. The end

You have successfully completed this tutorial.

Hannu Ranta 2012

Tutorial 1b: Simulation

Hannu Ranta

In this tutorial you will simulate counter block used in tutorial 1a using a specific testbench. This tutorial will take about 15 minutes to complete.

1. Open testbench and generate Modelsim-macro

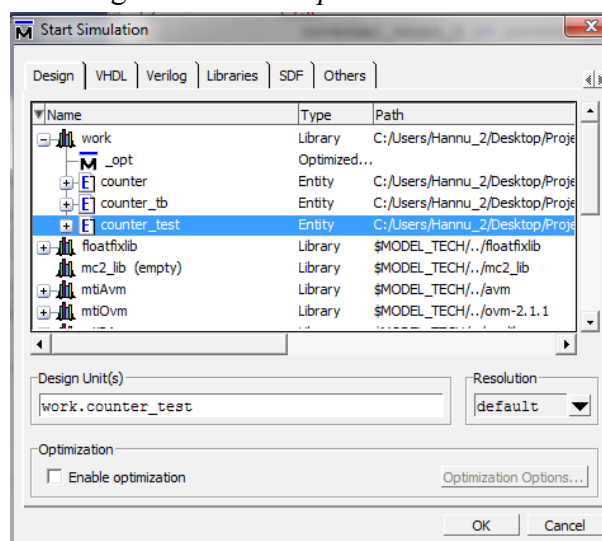
Open design named *counter_test* in Kactus 2 and generate top level VHDL and Modelsim-macros. This process is described in tutorial 1a. In windows environment you need to modify Modelsim-macro a bit. Open macro file in notepad or similar program and comment out lines shown in listing 1. (To comment out a line add # character in front of the line).

Listing 1: Corrected makefile

```
# rm -f Makefile
# vmake Makefile
echo " Script has been executed "
```

2. Simulating design

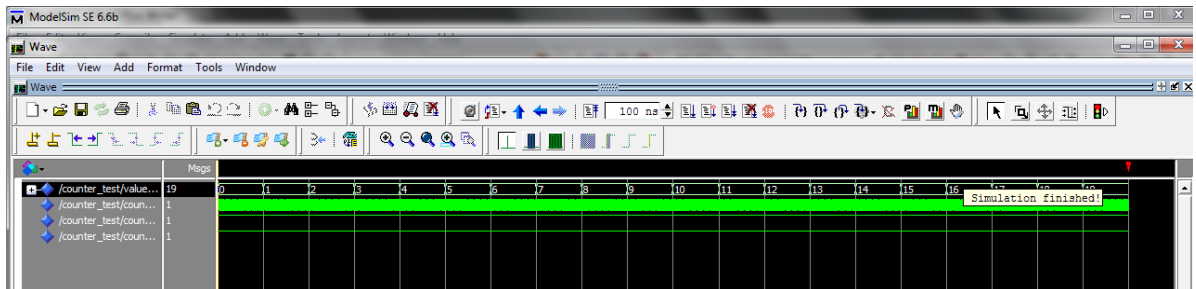
Open Modelsim and type command *do ../create_makefile*. Modelsim will compile files. You are ready to start simulation. Choose **Simulate->Start simulation** from menu. New window will open. Select *counter_test* and check that optimize design is NOT selected. Correct settings are shown in *picture 1*.



Picture 1: Simulation settings

Select all waves shown in blue background and right click on top of them. Choose **Add->To Wave->Selected Signal**.

Start simulation selecting **Run -All** from toolbar in wave window. Simulation will run and end up after few seconds. Your wave window should look like one in *picture 2*.



Picture 2: Wave-window

3. The end

You are now finished simulation.

Hannu Ranta 2012

Tutorial 2: Implementing IP-XACT block using Kactus 2

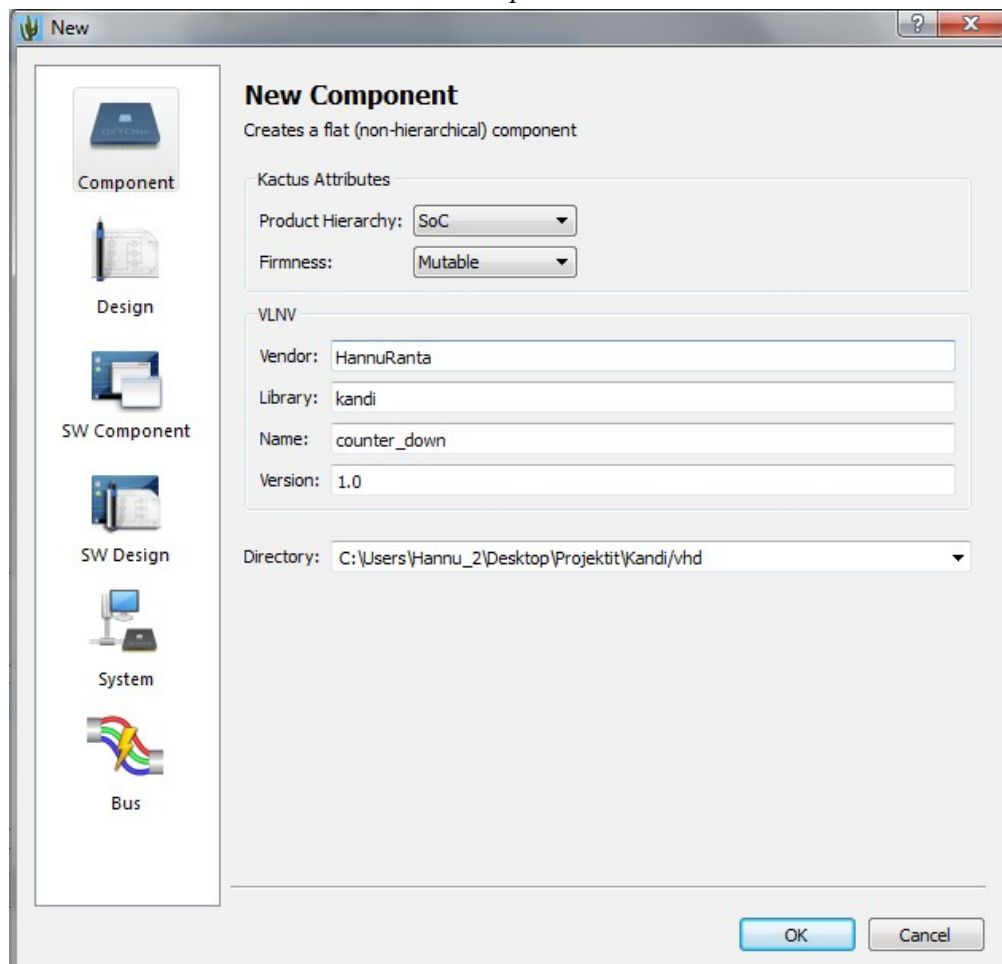
Hannu Ranta

In this tutorial you will create IP-XACT component and add it to the library. This tutorial requires the basic knowledge of digital logic and you should also complete tutorial 1 before this one. This tutorial will take about 10 minutes.

This tutorial is written for Kactus 1.2 but you can also use newer version of software.

1. Creating component

To create new component press **New** from toolbar. **New component** window will open. Enter VLN information as shown in *picture 1*.



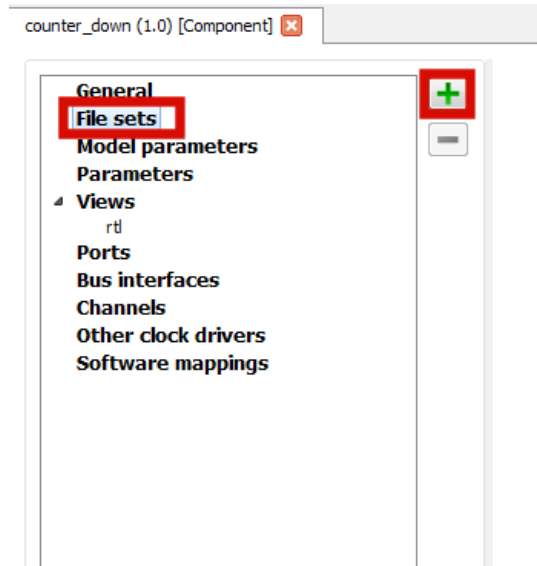
The screenshot shows the 'New Component' dialog box. On the left is a sidebar with icons and labels: 'Component' (blue chip), 'Design' (blue document), 'SW Component' (blue document with chip), 'SW Design' (blue document with chip), 'System' (computer monitor), and 'Bus' (rainbow arrow). The main window has a title bar 'New' and standard window controls. The title is 'New Component' with a subtitle 'Creates a flat (non-hierarchical) component'. Under 'Kactus Attributes', 'Product Hierarchy' is a dropdown set to 'SoC' and 'Firmness' is a dropdown set to 'Mutable'. Under 'VLNV', 'Vendor' is 'HannuRanta', 'Library' is 'kandi', 'Name' is 'counter_down', and 'Version' is '1.0'. The 'Directory' is a dropdown set to 'C:\Users\Hannu_2\Desktop\Projektit\Kandi\vhd'. At the bottom right are 'OK' and 'Cancel' buttons.

Picture 1: New component window

Press **OK** when form is filled. Component is now created.

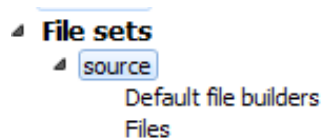
2. Adding source file

Next we need to add VHDL source file to design. To do so press **File Sets** tab active and press + sign shown in *picture 2*.



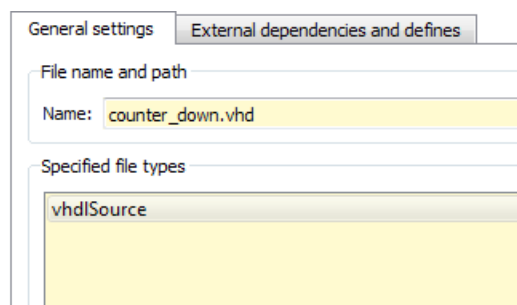
Picture 2: Add file sets

New tab will open. Enter a name for fileset in name field. Press a small arrow to expand file set list as shown in *picture 3*.



Picture 3: File sets list expanded

Select Files from the list shown above and press green + sign. File dialog will open. Find correct VHDL file (in this tutorial counter_down.vhd) and press **Open**. Set parameters as shown in *picture 4*.



Picture 4: Parameters for VHDL

Activate *Default file builders* tab and set compilation parameters as shown in *picture 5*.

File type	Command	Flags	Replace default flags
vhdlSource	vcom	-quiet -check_synthesis	false

Picture 5: File builders

3. Adding ports

Next we will add ports to IP-XACT file. Open *Ports* tab and add parameters as shown in *picture 6*.

General

File sets

- source
 - Default file builders
 - Files
 - counter_down.vhd

Model parameters

Parameters

Views

- rtl

Ports

Bus interfaces

Channels

Other clock drivers

Software mappings

	Name	Direction	Width	Left (higher) bound	Right (lower) bound	Type
1	clk	in	1	0	0	std_logic
2	rst_n	in	1	0	0	std_logic
3	enable	in	1	0	0	std_logic
4	value_out	out	20	19	0	std_logic_vector

Picture 6: Port settings

Ports are now defined. Next we will add *Bus interfaces*.

4. Adding Bus interfaces

Select Bus interfaces tab. At first we will define clock signal. Enter values as shown in *picture 7*. You can also drag these from library.

General Interface mode Port maps Parameters

Name and description

Name:

Display Name:

Description:

Bus definition

Vendor:

Library:

Name:

Version:

Abstraction definition

Vendor:

Library:

Name:

Version:

Picture 7: Clock interface settings

Select *interface mode* tab and set **Select interface mode** option to **slave**. Select *Port maps* tab and activate **CLK** from logical port list and **clk** from physical port list. Press **Connect**. After that you should have similar view than in *picture 8*. This step will map ports from IP-XACT component to ports in your VHDL.

General Interface mode Port maps Parameters

1 to 1 1 to many Clean up Connect

Logical ports

Physical ports

enable
rst_n
value_out

Logical left	Logical right	Logical name	Physical name
0	0	CLK	clk

Picture 8: Port maps

Add reset, enable and value_out ports similar way. General settings for each bus are shown in *picture 9*.

The image displays three screenshots of a software interface, likely a hardware description language (HDL) editor, showing the 'General' tab for defining signals. Each screenshot has tabs for 'General', 'Interface mode', 'Port maps', and 'Parameters'.

Top Left Screenshot (rst_n):

- Name and description:** Name: `rst_n`, Display Name: (empty), Description: (empty).
- Bus definition:** Vendor: `TUT`, Library: `ip.hwp.interface`, Name: `reset.busdef`, Version: `1.0`.
- Abstraction definition:** Vendor: `TUT`, Library: `ip.hwp.interface`, Name: `reset.absDef`, Version: `1.0`.

Top Right Screenshot (enable):

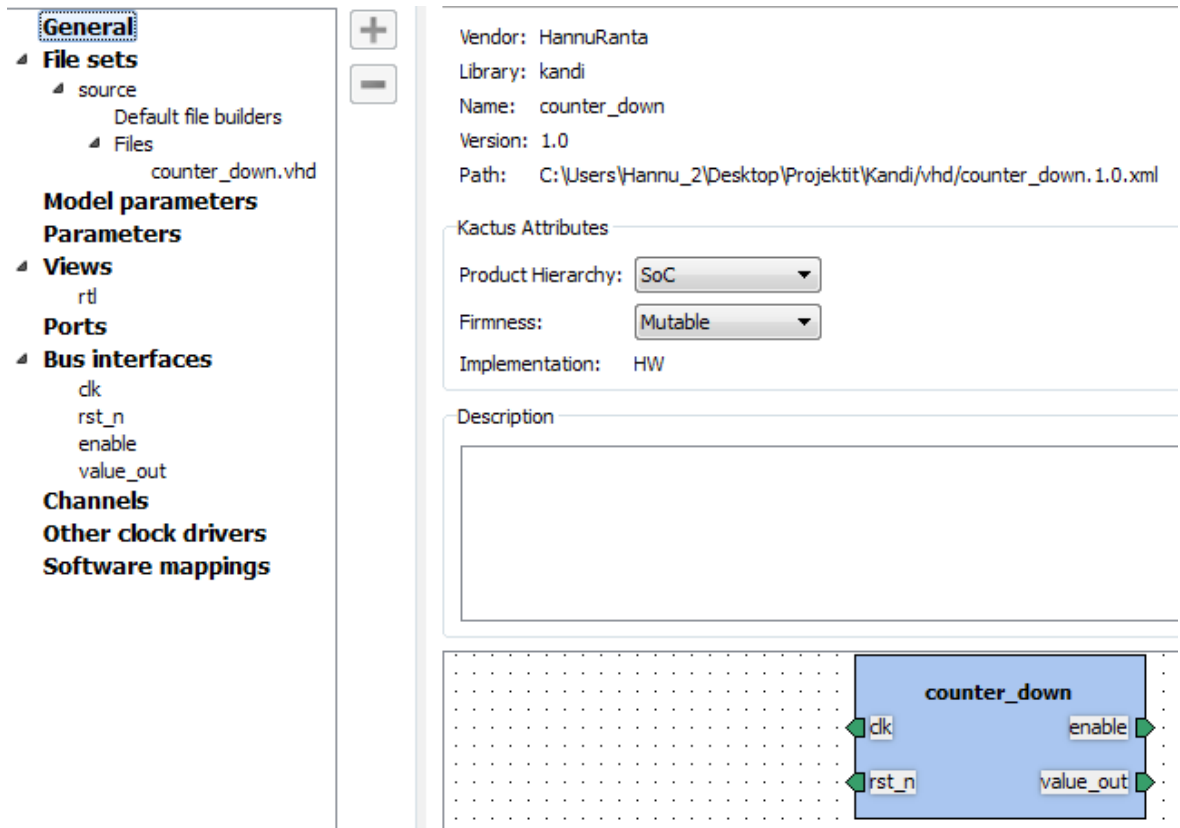
- Name and description:** Name: `enable`, Display Name: (empty), Description: (empty).
- Bus definition:** Vendor: `HannuRanta`, Library: `kandi`, Name: `enable`, Version: `1.0`.
- Abstraction definition:** Vendor: `HannuRanta`, Library: `kandi`, Name: `enable.absDef`, Version: `1.0`.

Bottom Screenshot (value_out):

- Name and description:** Name: `value_out`, Display Name: (empty), Description: (empty).
- Bus definition:** Vendor: `HannuRanta`, Library: `kandi`, Name: `value_out`, Version: `1.0`.
- Abstraction definition:** Vendor: `HannuRanta`, Library: `kandi`, Name: `value_out.absDef`, Version: `1.0`.

Picture 9: General settings for reset, enable and value_out signals

All connections are now added and should be seen in *general* tab as shown in *picture 10*.



Picture 10: General tab

As you can see, ports are now also marked in preview image.

5. The end

You are now successfully completed this tutorial. You can replace original counter with the newly created *counter_down* and return to *tutorial 1*.

Hannu Ranta 2012