# Komponentin paketointi

## Elementtien paketointijärjestys

Jotta paketointivuossa olisi selkeämpi järjestys on elementtien järjestyksessä ohjattava siihen että ensin paketoidaan sellaiset elementit joissa ei viitata muihin elementteihin ja tämän jälkeen käsitellään muut elementit jolloin niitä tehtäessä voidaan verrata jo olemassa oleviin elementteihin. Tarvittaessa on kuitenkin pystyttävä palaamaan vuossa takaisinpäin, jotta voidaan esim. lisätä unohtunut portti.

Järjestys:

1. VLNV ja description kenttä

* VLNV määrittelee komponentille yksiselitteisen tunnisteen kirjaston sisällä.
* Description sisältää tekstuaalisen kuvauksen komponentista.
* Pakollinen aina

1. FileSetit (spirit:fileSets)

* Komponenteissa voi olla useita fileSetteja (ja kannattaa olla esim. sourcet eri fileSetissä kuin dokumentaatiotiedostot ja projektitiedostoille oma fileSet)
* Pakollinen kenttä

1. Tiedostot (spirit:file fileSetin sisällä)

* Määrittelee yksittäisen tiedoston joka kuuluu kyseiselle komponentille
* Pakollinen kenttä

1. Choices (spirit:choices)

* Tällä voidaan määritellä jollekin nimelle rajattu arvojoukko (vrt. c++ enum).
* Tätä voidaan käyttää modelParametreissa ja parametreissa tai missä tahansa elementissä käyttämällä choiceRef-elementtiä.
* Ei pakollinen kenttä

1. modelParametrit (spirit:modelParameters)

* sisältää parametrit joilla voidaan muokata komponentin asetuksia ylemmältä hierarkiatasolta.
* Esim VHDL-kielen genericit tulee määritellä näillä elementeillä
* Voi käyttää spirit:choice-elementtiä
* Pakollinen mikäli genericeja on olemassa

1. Parametrit (spirit:parameters)

* Määrittelee komponentille konfiguroitavan arvon jota voidaan käyttää muualla komponentin IP-Xact spesifikaatiossa.
* Eroaa modelParametrista siten että näitä arvoja ei voi johtaa hierarkiassa ylemmältä tasolta vaan ne määritellään komponentille itselleen eli ne ovat tavallaan vakioita joita voi käyttää komponentin sisällä.
* Voi käyttää spirit:choice-elementtiä
* Ei pakollinen kenttä

1. Muistikartat (spirit:memoryMaps)

* Voidaan määritellä jokaiselle slave-interfacelle luomalla kyseisestä interfacesta viite tällaiseen elementtiin.
* Mahdollisesti funbasessa pakollinen kenttä?

1. OsoiteAvaruudet (spirit:addressSpaces)

* Määrittelee master-interfacen näkemän osoiteavaruuden siten että master-interfacesta voidaan viitata tällaiseen elementtiin.
* Mahdollisesti funbasessa pakollinen kenttä?

1. remapStates (spirit:remapStates)

* Määrittelee remap-ehdot (mitä varten?)
* Ei sisällä osoitteiden remappausta vaan se tehdään memoryRemap-elementissä
* Ei pakollinen kenttä

1. Näkymät (spirit:views spiri:modelin sisällä)

* Voi sisältää viitteen designiin tai designConfigurationiin jolloin komponentti on hierarkinen ja sisältää alikomponentteja
* Voi sisältää viitteen filesettiin jossa on määritelty komponentin toteutuksen lähdetiedostot.
* Pakollinen kenttä

1. Portit (spirit:ports spirit:modelin sisällä)

* Sisältää komponentin ulospäin näkyvät portit
* Portti on joko wire- tai transactional-tyyppiä
* Voi sisältää määrittelyn portin tyypistä jossa on määritelty mihin komponentin näkymiin kyseinen määrittely pätee. Eli siis saattaa sisältää viitteen view-elementtiin.
* Pakollinen kenttä

1. busInterfacet (spirit:busInterfaces)

* Määrittelee interfacen johon voidaan yhdistää toisen komponentin samantyyppinen interface
* Voi olla tyyppiä master, slave, system, monitor, mirroredMaster, mirroredSlave, mirroredSystem tai monitor
* Master-tyyppi voi sisältää viitteen addressSpace-elementtiin
* Slave-tyyppi voi sisältää viitteen memoryMap-elementtiin
* Slave-tyyppi voi sisältää bridge-elementin jolla voidaan yhdistää master-interface kyseiseen slave-interfaceen komponentin sisällä (toimii bridgenä kahden channelin välissä)
* Pakollinen kenttä

1. Kanavat (spirit:channels)

* Sisältää listan kaikista mirroredBusInterfaceista jotka kuuluvat samaan kanavaan.
* Sisältää viitteen vähintään kahteen busInterfaceen.
* Pakollinen kenttä jos kyseessä on väyläkomponentti

1. Prosessorit (spirit:cpus)

* Määrittelee ohjelmoitavan suorittimen jolla on jonkinlainen osoiteavaruus.
* Sisältää viitteen addressSpace-elementtiin.
* Ei pakollinen kenttä

1. otherClockDrivers (spirit:otherClockDrivers)

* Sisältää määrittelyt kellosignaaleista jotka eivät tule suoraan komponentin top-level porteista (esim generoidut kellot tai virtuaaliset kellot)
* Tällä voidaan määritellä kellosignaalin muoto
* Ei pakollinen kenttä

1. Komponenttigeneraattorit (spirit:componentGenerators)

* Määrittelee generaattorin joka voidaan ajaa tälle komponentille
* Sisältää polun ajettavaan generaattoritiedostoon

Ei pakollinen kenttä