Onderzoek visualisatie SystemC

Onderzoek naar het visualiseren van een SystemC model

14 april 2011

Dennis van Leeuwen 07004222

Inhoudsopgave

1. Inleiding....3
2. Voorbeeld van het grafisch weergeven van een SystemC model 4
3. Formaten....6
4. Tekenprogramma’s....8
5. Verdere formaten en programma's....12
6. Conclusie....13
7. Ontwikkeling van het conversie programma....15
8. Resultaat....18
9. Bronnen....20
10. Bijlages....21

1. Inleiding

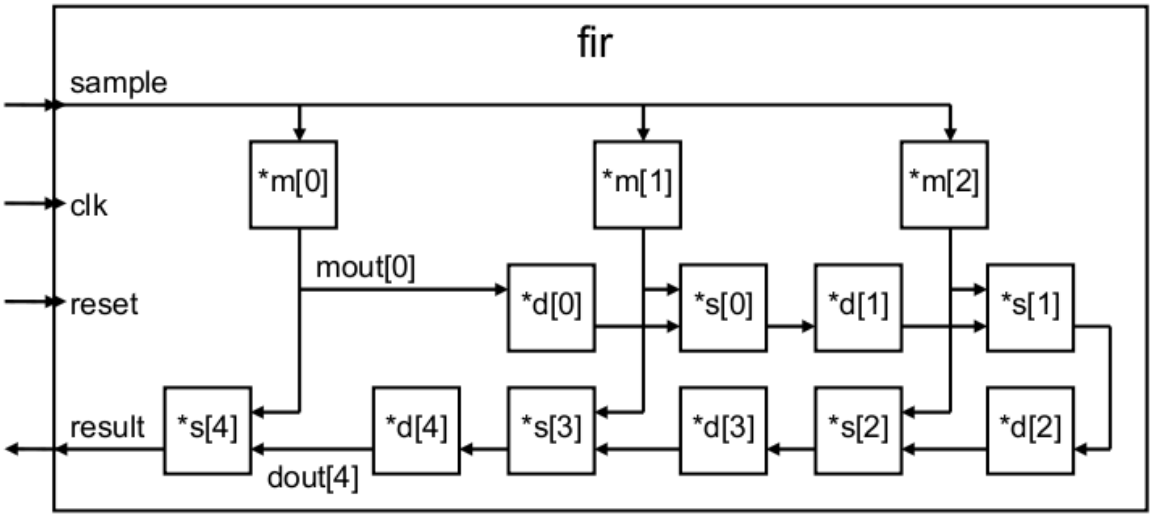
Er wordt een onderzoek gedaan naar een manier om de gegevens die er uit de extractie van een SystemC model worden gehaald te visualiseren. Het SystemC model wordt op dit moment beschreven in een formaat genaamd SCMDL(SystemC Model Description Language). Dit formaat is gebaseerd op het XML formaat. SCMDL wordt alleen gebruikt voor het project SHaBE en kan dus niet zomaar in een tekenprogramma worden geladen om te worden gevisualiseerd. Er wordt daarom een onderzoek gedaan naar een tekenprogramma waarin een SystemC model gegenereerd door SHaBE kan worden gevisualiseerd of naar andere manier om een SystemC model te visualiseren.

In het eerste hoofdstuk wordt een voorbeeld gegeven hoe de grafisch weergave van het SystemC model eruit moet komen te zien en worden de diverse eisen van de te visualiseren onderdelen beschreven. Het voorbeeld in dit hoofdstuk wordt als richtlijn genomen als grafisch weergave van een SystemC model beschreven in het SCMDL formaat.

In de daaropvolgende hoofdstukken worden de verschillende programma’s en formaten die zijn onderzocht besproken. Bij elk formaat en programma wordt een toelichting gegeven tot in hoeverre dit formaat en dit programma geschikt is voor het visualiseren van de hiërarchie van een SystemC model.

2. Voorbeeld van het grafisch weergeven van een SystemC model

Om te kunnen bepalen hoe de grafische weergave van een SystemC model eruit moet komen te zien, wordt het voorbeeld uit de scriptie van Harry Broeders gebruikt. In afbeelding 1 is deze afbeelding weergegeven. Deze afbeelding is een grafische weergave van het SystemC model dat wordt gegenereerd bij het uitvoeren van SHaBE op het SystemC model in bijlage A.



Afbeelding 1: Schematische weergave FIR filter SystemC model

In de tekening zijn diverse aspecten van belang:

Tonen van input en output poorten
Om onderscheid te maken tussen in- en outputpoorten kan er gebruik worden gemaakt van bijvoorbeeld een driehoek. De richting van de driehoek bepaalt of het een in- of output poort is. In de bovenstaande afbeelding is 'sample' een input poort en 'result' een output poort.
Specificaties van het weergeven van een poort:

- Tonen van de poort. (SPEC1)
- Tonen van het type van een poort. (SPEC2)
- Tonen van de naam van de poort. (SPEC3)

Tonen van signalen en de richting van de signalen
Een signaal is een verbinding tussen verschillende poorten. Ieder signaal heeft een richting. Via een signaal kan data worden verstuurd tussen twee poorten.
Specificaties van het weergeven van een signaal:

- Tonen van het signaal (SPEC4)
- Tonen van de richting van een signaal. (SPEC5)
- Tonen van de naam van een signaal. (SPEC6)

Tonen van de modules en de hiërarchie van de modules
Zoals in bovenstaande afbeelding is te zien bevat de module genaamd 'fir' diverse submodules zoals *m[0], *d[0] en *s[0]
Specificaties van het weergeven van een module:

- Tonen van de module. (SPEC7)
- Tonen van de hiërarchie van een module. (SPEC8)
- Tonen van de naam van een module. (SPEC9)

Overige
Naast bovenstaande specificaties is er een extra wens:

- Bij voorkeur is het gebruikte programma open-source. (SPEC10)

3. Formaten

3.1. GML

GML is een formaat ontwikkeld als standaard voor het opslaan van een graaf. In het formaat zijn verschillende tags voor groepen, knopen, grafen en de lijnen tussen de knopen gedefinieerd. Het GML formaat is opgezet als standaard voor het tonen van grafen, dit omdat er verschillende tools ontstonden die ieder hun eigen formaat gebruikte. Om onderscheid te maken tussen de verschillende gegevens in een GML bestand wordt er gebruik gemaakt van blokhaken. Een voorbeeld hiervan is te zien in bijlage B.

Voordelen:

1. Veel specificaties kunnen worden gerealiseerd, namelijk: SPEC1 en SPEC3 t/m SPEC9.

Nadelen:

1. Formaat is opgebouwd met blokhaken. Dit maakt het formaat snel onoverzichtelijk.
2. Poorten kunnen niet worden beschreven.

3.2. GraphML

GraphML is een formaat gebaseerd op het XML en op het GML formaat. Net zoals de oprichters van het GML formaat vonden de oprichters van het GraphML formaat het nodig dat er een standaard formaat kwam waarmee grafen getoond kunnen worden. GraphML is een opvolger van het GML formaat. In GraphML is er rekening gehouden met eventuele uitbreidingen die nodig zijn om extra gegevens in het formaat op te slaan. Hiervoor kunnen extra tags kunnen worden gebruikt. Dit kan door zelf een uitbreiding van het formaat te definiëren in een DTD.

Voordelen:

1. Goed uitbreidbaar formaat
2. Bijna alle specificaties kunnen worden gerealiseerd, namelijk SPEC1 en SPEC3 t/m SPEC9. In combinaties met de yFiles extentie kan ook SPEC2 worden gerealiseerd.

Nadelen:

1. Gebaseerd op XML. Dit kan bij grote grafen meer overhead opleveren dan bij bijvoorbeeld GML.

3.3. GXL(Graph eXchange Language)

Zover er informatie te vinden was over GXL, lijkt het dat GXL vooral bedoeld is om klassenmodellen mee te tekenen. In het GXL formaat kunnen geen poorten worden gedefinieerd. Het tonen van poorten zal dus op een andere manier moeten worden gedaan als dit formaat wordt gebruikt.

Voordelen:

1. Veel gewenste punten kunnen worden gerealiseerd, namelijk SPEC3, SPEC4, SPEC5, SPEC6, SPEC7, SPEC8, SPEC9.

Nadelen:

1. Het formaat is vooral bedoeld om klassenmodellen in te beschrijven.

2. Poorten kunnen niet worden beschreven.

3.4. Graph6

Graph6 is een formaat waarin grafen kunnen worden opgeslagen. Het formaat maakt geen onderscheid tussen de graaf waar de pijl begint en de graaf waar de pijl eindigt, waardoor het erg moeilijk wordt om de afhankelijkheden in de SystemC hiërarchie te visualiseren.
Voordelen:

1. Veel compacter formaat dan een op XML gebaseerd formaat.

Nadelen:

1. De richting van de pijl is niet te definiëren
2. Alleen de specificaties SPEC4 en SPEC6 kunnen worden gerealiseerd
3. Niet leesbaar formaat voor mensen.

3.5. IP-XACT

IP-XACT is een formaat waarmee hardware modules kunnen worden beschreven. IP-XACT is een open standaard formaat gebaseerd op XML. In dit formaat kunnen de diverse componenten beschreven worden die er in een systeem worden gebruikt. Dit formaat is dus niet een formaat waarin een graaf wordt beschreven, maar een formaat waarin de eigenschappen van een hardware module wordt beschreven. Omdat deze eigenschappen allemaal in dit formaat worden beschreven is het mogelijk om dit formaat te gebruiken bij het genereren van modellen voor een simulatie van hardware.
Voordelen:

1. Gestandariseerd formaat(volgens IEEE)
2. Gebaseerd op XML, waardoor het geparsed kan worden.
3. Alle specificaties kunnen worden gerealiseerd in het IP-XACT formaat.

Nadelen:

1. IP-XACT wordt door maar (heel) weinig programma's gebruikt. Het enige gevonden programma is Magillem.

4. Tekenprogramma’s

4.1. OGDF(Open Graph Drawing Framework)

Met gebruik van de OGDF library kan er een applicatie worden geschreven waarin alle wensen vervuld kunnen worden. In het OGDF zijn diverse algoritmes aanwezig om de route van de pijlen en de plaats van de modules te bepalen, zodat er geen overlappende pijlen en modules in de tekening worden geplaatst. Hierdoor het gebruik van deze algoritmes blijft de tekening overzichtelijk.
In OGDF is ook een utility beschikbaar genaamd gml2pic. Door het gebruik van deze utility kan de graaf die met OGDF is gemaakt worden omgezet naar een afbeelding.
Voordelen van OGDF:

1. Aanwezigheid van lay-out algoritmes
2. Vrijheid van implementatie qua vormen en dikte van lijnen.
3. Open source library

Nadelen:

1. Gemaakte graaf moet worden geëxporteerd als afbeelding of GraphML bestand. Een SystemC model geëxporteerd als afbeelding wordt snel onoverzichtelijk naarmate er meer modules in de afbeelding worden gebruikt, omdat de hiërarchie niet in of uitklapbaar is in een afbeelding. Er zal dus alsnog een ander programma nodig zijn waarmee een GraphML bestand geopend kan worden.

4.2. yFiles

yFiles is net als het OGDF een library waarmee de structuur van een graaf kan worden beschreven. De yWorks library kan net zoals het OGDF de onderdelen uit de graaf met een algoritme positioneren zodat er weinig/geen overlap is te zien in de visualisatie van de graaf.
Het gebruik van yFiles vereist echter een licentie. Deze licentie kost tussen de 1800 en de 19200 euro. yFiles is tevens ook geen open source project. De source code kan wel gekocht worden, maar dit kost ook tussen de 12000 en 36000 euro. Waarbij er per jaar nog eens een bedrag moet worden betaald om de updates te ontvangen.
Voordelen:

1. Aanwezigheid lay-out algoritmes

Nadelen:

1. Gemaakte graaf moet worden geexporteerd als afbeelding of GraphML bestand. Afbeelding is niet in of uitklapbaar, waardoor er GraphML gebruikt moet worden. Er moet dan nog een programma worden gemaakt/gevonden waarmee de GraphML kan worden gevisualiseerd.
2. De kosten van de licentie voor het gebruik van het programma zijn erg hoog.

4.3. yEd

Met het programma yEd kan een GraphML bestand geopend worden. Het programma yEd is ontwikkeld in Java en zou dus op ieder platform moeten kunnen worden uitgevoerd.
In yEd is een functie beschikbaar waarmee gebruikers zelf de graaf kunnen positioneren. yEd is gratis te gebruiken, maar het programma is niet open-source. Een voorbeeld van yEd is te zien in bijlage E.
Voordelen:

1. Gratis te gebruiken.
2. Beschikbaar op veel platformen.
3. Grafisch mooie omgeving.
4. Biedt de mogelijkheid aan de gebruik om de graaf volgens een bepaalde lay-out algoritme te tonen.

Nadelen:

1. Niet open source.
2. Programma kan de port tags uit het GraphML formaat (nog) niet visualiseren.

4.4. Graphlet

Graphlet is een tool waarmee het GXL formaat kan worden gevisualiseerd. Het ontwikkelteam achter deze tool lijkt echter gestopt te zijn met het ontwikkelen van de tool. De tool biedt niet echt een mooie interface. Het programma is niet open-source, maar het is tot op heden wel gratis te gebruiken.
Nadelen:

1. Ontwikkeling is gestopt.
2. Grafische weergave van een graaf is minder mooi als in de andere onderzochte programma's.
3. Programma tekent volgens het GXL formaat en dit formaat is vooral bedoeld om klassendiagrammen te tekenen, waardoor de tekening er niet uit komt te zien, zoals de opdrachtgever wenst.
4. Poorten kunnen niet worden getekend.

4.5. VGJ

VGJ is een tool waarmee de GML grafen getoond kunnen worden. Deze tool biedt echter niet de mogelijkheid om de hiërarchie van de modules goed te tonen, waardoor het bijna onmogelijk wordt om deze tool voor het tonen van de SystemC modules te gebruiken.
Voordelen:

1. Java applicatie dus op ieder platform te gebruiken.
2. Open source

Nadelen:

1. Grafisch minst mooie visualisatie van een graaf tov andere programma's.
2. Biedt geen mogelijkheid om poorten te tonen.

4.6. Magillem

Voor het openen van het IP-XACT formaat is er op dit moment maar 1 programma beschikbaar en dat is Magillem. Een licentie voor dit programma kost tussen de €2000 en €65000. Om IP-XACT te kunnen bewerken kan een simpele tekstverwerker worden gebruikt. Voor het visualiseren van IP-XACT zal echter het Magillem programma moeten worden aangeschaft.
Voordelen:

1. Programma maakt gebruik van een standaard, IP-XACT
2. Grafisch mooie weergave en biedt de mogelijkheid om de modules uit de IP-XACT direct te gebruiken in een ander project.

Nadelen:

1. Licenties voor dit programma zijn voor veel mensen onbetaalbaar. Hierdoor wordt het moeilijk om de open source wereld van dit programma gebruik te laten maken om IP-XACT te openen. Dit programma is zover bekend het enige programma wat het IP-XACT formaat kan openen.

4.7. SpiceVision

SpiceVision is een programma waarmee RTL blok diagrammen gemaakt kunnen worden. Ook is het mogelijk om op een hoger RTL niveau het blok diagram te bekijken. Dit is een van de wensen van de opdrachtgever. Om SpiceVision te gebruiken is er wel een licentie nodig en het programma is niet open source.
De kosten van deze licentie liggen rond de 3.200 euro per jaar. Voor academische doeleinden wordt er een korting gegeven van 50%
Voordelen:

1. Goede visualisatie van de hiërarchie.
2. Maakt standaard gebruik van lay-out algoritmes.

Nadelen:

1. Programma is niet open source
2. Erg dure licentie

4.8. Block Diagram Editor

Block Diagram Editor is net als SpiceVision een tool waarmee blokdiagrammen getoond kunnen worden. Helaas wordt als snel duidelijk dat het onmogelijk is om een SystemC model in dit programma te tekenen zonder dat er zelf een lay-out algoritme moet worden toegevoegd. In Block Diagram moet je tot in tegenstelling tot SpiceVision zelf bepalen op welk coördinaat een signaal begint, een module komt te staan of waar een poort wordt geplaatst.
Voordelen:

1. Open source programma.

Nadelen:

1. Het programma bevat geen lay-out algoritme om de modules en signalen mee te positioneren.
2. Hiërarchie is niet inklapbaar.

4.9. gSysC

gSysC is een programma wat net als SHaBE gegevens uit een SystemC model kan extraheren. gSysC biedt echter niet de mogelijkheid om dynamisch de gegevens uit een SystemC model te extraheren, vandaar dat SHaBE is ontwikkeld. gSysC kan tevens ook de hiërarchie van het SystemC model tonen. Dit deel van gSysC zou kunnen worden hergebruikt om de gegevens die SHaBE uit het SystemC model haalt te tonen. Een voorbeeld van een gSysC model is te zien in bijlage D.
gSysC maakt gebruik van Qt3 om de hiërarchie te tekenen. In Qt3 zijn de algoritmes aanwezig om lijnen en blokken te tonen. In Qt3 is echter geen lay-out algoritme aanwezig, daarom heeft de auteur van het programma voor een alternatieve manier van het tonen van modules en signalen gekozen en dat is om de modules even groot weer te geven en in rijen evenwijdig aan elkaar. Vervolgens worden de lijnen die tussen de modules lopen gebundeld wanneer deze naar een andere module toe gaan. Zoals in bijlage D is te zien, kan dit een erg onoverzichtelijke lay-out opleveren.
Door op een module te klikken wordt de hiërarchie van de module in een nieuw venster weergegeven.
Voordelen:

1. Programma is open source
2. Hiërarchie kan goed worden weergegeven.

Nadelen:

1. Signalen worden gebundeld. Hierdoor is het onoverzichtelijk waar het signaal vandaan komt en waar het signaal naartoe gaat.

5. Verdere formaten en programma's

Uit de bovenstaande formaten en tekenprogramma's blijkt al snel dat veel van de formaten en programma's gemeenschappelijk eigenschappen hebben. Bovenstaande selectie van programma's en formaten is slechts een deel van alle bekeken formaten en programma's. Er zijn eigenlijk teveel programma's om op te noemen die grafen kunnen visualiseren. Ieder programma waarin UML wordt weergegeven is al zo'n programma. In UML zijn ook diverse diagrammen gedefinieerd. Het diagram wat nog het meeste lijkt op wat er gewenst wordt is het component diagram. Na divers onderzoek zijn er ook diverse programma die vanuit XML een diagram kunnen tekenen, zoals het programma hypermodel. Echter kan in dit geval geen component diagram worden getekend maar alleen een klassendiagram.

6. Conclusie

Het programma waarmee de hiërarchie van het SystemC model gevisualiseerd gaat worden, is bij voorkeur open source. Magillem is dit niet en valt daarmee eigenlijk al af, omdat er daarnaast betere oplossingen zijn. Dit programma is echter een prima alternatief en biedt ook het voordeel om de modules die uit het SystemC model worden geëxtraheerd direct te gebruiken in een hard- en software simulatie. Dit is een erg handige functie voor hard- en software ontwikkelaars. Omdat het programma niet gratis te downloaden is, wordt ervoor gekozen om het IP-XACT formaat en daarmee het enige programma wat het IP-XACT formaat kan openen, Magillem, niet te gebruiken.

Graph6 is een erg onduidelijk formaat omdat het niet uit human-readable tekst bestaat. Verder kan dit formaat alleen undirected graphs opslaan, waardoor het onmogelijk wordt om aan te geven in de tekening welke kant het signaal opgaat.

Het formaat GXL biedt niet de mogelijkheid om poorten aan te maken. Het formaat is hierdoor niet geschikt om het SystemC model in op te slaan. Het formaat is tevens ook niet uitbreidbaar met een zelfgemaakte DTD.
GML biedt dezelfde mogelijkheden als GXL en is daardoor ook ongeschikt, omdat poorten niet kunnen worden gedefinieerd.

GraphML heeft al deze nadelen niet. In GraphML kan alles worden gedefinieerd wat nodig is om de hiërarchie van een SystemC model te visualiseren. Het voordeel wat GraphML daarnaast biedt is dat het goed uitbreidbaar is, zodat extra eisen die in de toekomst aan het formaat worden gesteld in het formaat toegevoegd kunnen worden. Dit kan door zelf een extensie van het GraphML formaat in het GraphML bestand toe te voegen.

Om GraphML weer te geven zijn er verschillende tools beschikbaar. Een van de tools die zowel de hiërarchie volgens in een in- en uitklap mechanisme kan weergeven en alle namen bij de juiste items kan zetten is yEd.
yEd is gratis te gebruiken en biedt het voordeel dat het te gebruiken is op alle platformen die JAVA ondersteunen.
yEd kan met verschillende formaten werken, waaronder GraphML. Omdat er tijdens het omzetten van de SCMDL naar GraphML geen coördinaten worden toegevoegd aan de GraphML is het van belang om een editor te gebruiken, die de lay-out van de graaf zelf kan opmaken. Dit algoritme voor het bepalen van de lay-out is in yEd aanwezig. Het enige nadeel van yEd is dat dit programma de poorten die in de GraphML gedefinieerd zijn niet toont. Het tonen van poorten wordt wel opgegeven als feature. Dus dit zal in de toekomst wel mogelijk worden.

Tot slot is er nog de SystemC front-end genaamd gSysC die ook de geëxtraheerde data uit een SystemC beschrijving grafisch kan weergeven. Het grote nadeel van gSysC is dat deze de signalen bundelt, waardoor het niet precies is wat de opdrachtgever wilt hebben. Om gSysC aan passen om ook de

signalen volgens een lay-out algoritme te tonen is hoogstwaarschijnlijk teveel werk, omdat de gehele implementatie van gSysC daardoor moet worden aangepast.

Het uiteindelijke formaat wat gekozen wordt is GraphML. Dit formaat wordt gekozen omdat het alles biedt wat nodig is en daarbij ook uitbreidbaar is. Dit is erg handig als er in de toekomst extra wensen ontstaan, waardoor het formaat uitgebreid moet worden.

Als tekenprogramma wordt gekozen voor yEd. Dit wordt ten eerste gedaan omdat de opdrachtgever niet wenst zelf een tekenprogramma te ontwikkelen, ook al is dat in dit geval de beste optie, omdat geen van alle tekenprogramma's perfect passen bij de aangegeven eisen. Een van de redenen van het niet ontwikkelen van een tekenprogramma is dat dit hoogstwaarschijnlijk niet binnen de tijdsperiode van de afstudeerstage kan gebeuren. Mocht dit tekenprogramma in de toekomst worden ontwikkeld dan kan dit programma gebruik maken van onderdelen uit de volgende programma's: OGDF, Block diagram Editor en gSysC. Deze library en programma's zijn open source en bieden allemaal onderdelen die genodigd zijn in een tekenprogramma die aan de eisen van de opdrachtgever moet gaan voldoen.

De voornaamste reden dat er voor yEd wordt gekozen en niet voor andere tekenprogramma's zijn:

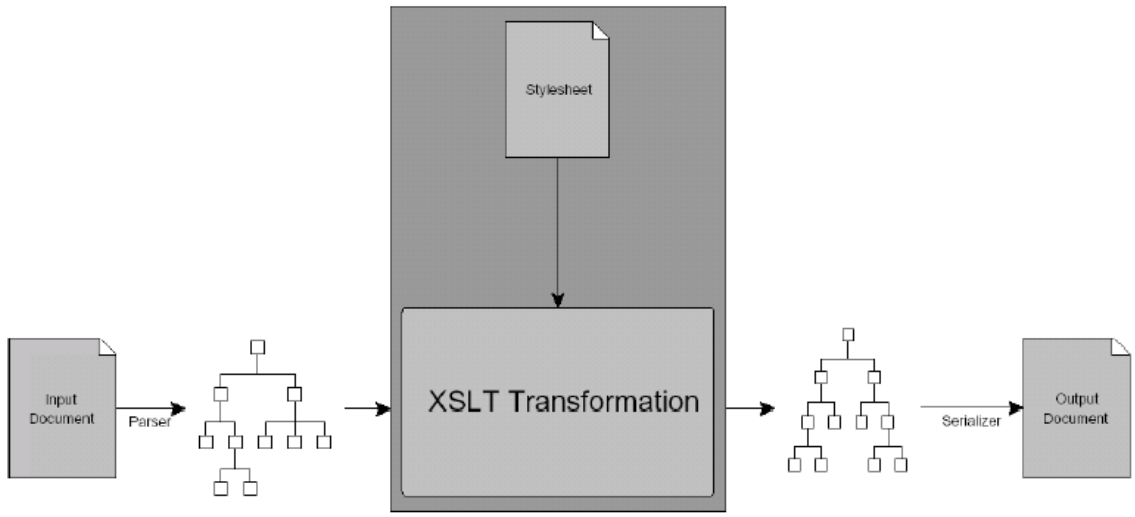
- yEd is gratis te gebruiken.
- yEd biedt in tegenstelling tot de andere programma's de mogelijkheid om alle eisen van de opdrachtgever in te voeren. Op dit moment ondersteund yEd het visualiseren van poorten nog niet. Dit is wel een van de punten die de ontwikkelaars van yEd op het programma hebben staan. Wanneer de poorten daadwerkelijk worden geïmplementeerd is onduidelijk.
- yEd biedt is een visueel mooie omgeving die in JAVA is geïmplementeerd, waardoor het een platform onafhankelijk programma is.

7. Ontwikkeling van het conversie programma

Nu is vastgesteld dat de SCMDL, gegenereerd door SHaBE, moet worden omgezet naar GraphML, voor het tonen van de afbeelding, wordt er een programma genaamd SCMDL2GraphML ontwikkeld.

7.1. Transformatie SCMDL

SCMDL staat voor System-C Model Description Language. SCMDL is een formaat gebaseerd op XML. In dit formaat is zowel de hiërarchie als het gedrag van een System-C model opgeslagen. De gehele definitie van het formaat is te vinden op http://shabe.sourceforge.net/systemc-model/systemc-model.html
Uit het onderzoek welke voorafgaand aan dit ontwerp is gedaan, is gekomen dat het formaat moet worden omgezet naar het GraphML formaat. Dit wordt gedaan door gebruik te maken van een XSLT transformatie. Dit wordt gedaan zodat de applicatie die ontwikkeld wordt onafhankelijk wordt van het formaat waarnaar de SCMDL getransformeerd gaat worden.

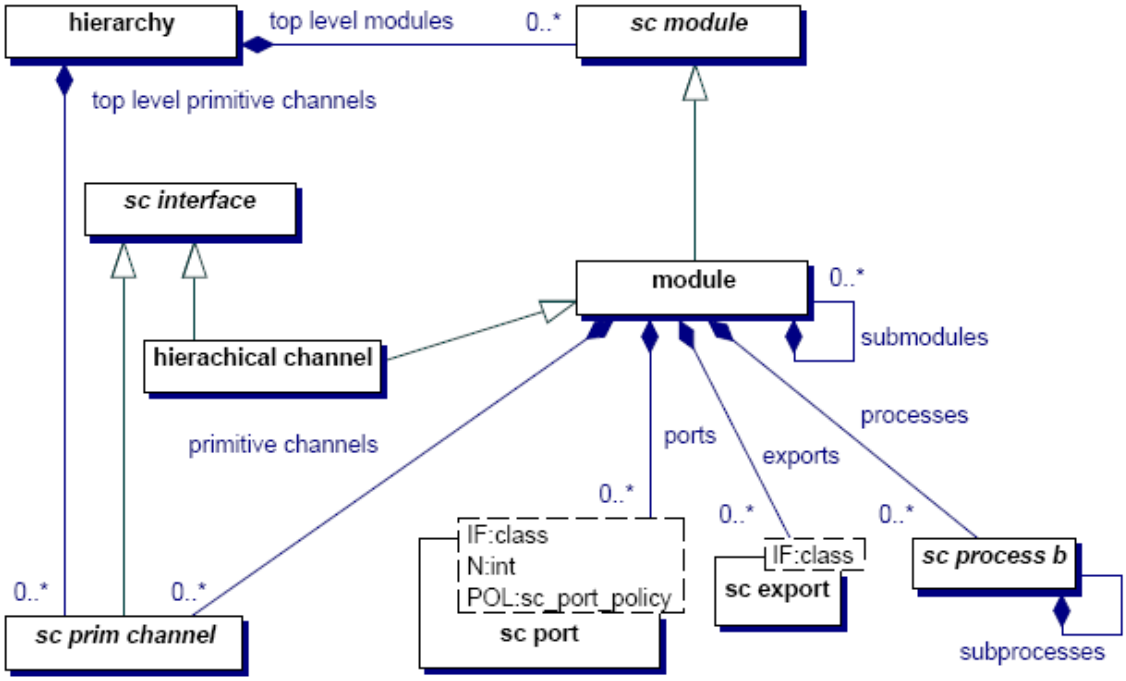


Afbeelding 2: Schematische overzicht XSLT transformatie

Voor de ontwikkeling worden de libraries Xerces-C en Xalan-C gebruikt. Deze 2 libraries bevatten functionaliteit waarmee XML kan worden geparsed en XML kan worden getransformeerd volgens een XSLT transformatie. De Xerces-C en Xalan-C zijn beide open-source library en daarom gratis te gebruiken in de implementatie van de XSLT tranformator.

De structuur waarnaar de SCMDL moet worden getransformeerd bestaat uit diverse tags. Hieronder een overzicht van de structuur van de elementen.

7.2. SCMDL



Afbeelding 3: SCMDL XML hierarchie

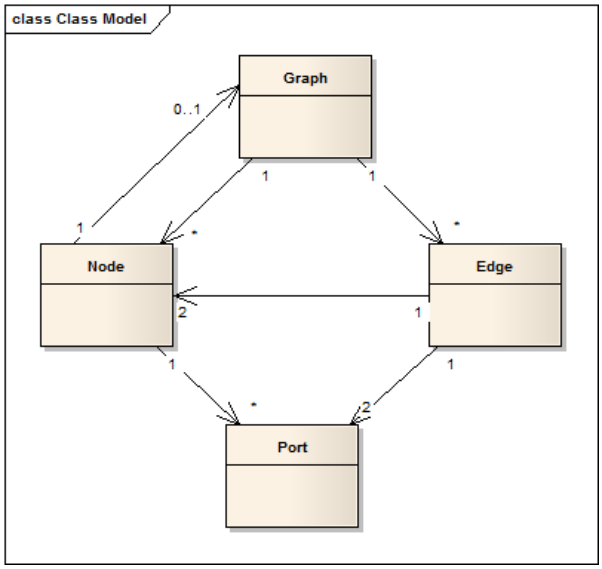
In bovenstaande afbeelding zijn de afgeleide klasses allemaal tags in de SCMDL. Tijdens het omzetten van de SCMDL naar de GraphML wordt er alleen gefocust op het omzetten van de volgende SCMDL tags:

- hierarchy
- module
- sc_port
- sc_export
- sc_prim_channel
- hierarchical channel

Dit wordt gedaan, omdat er tijdens het onderzoek ook alleen gefocust is op deze aspecten van het formaat.

7.3. GraphML

Zoals te zien in de bovenstaande afbeeldingen, bestaat de SCMDL uit veel meer tags dan de GraphML. GraphML bestaat namelijk alleen uit een graph, node, edge en port tag. De relatie tussen deze tags is in de rechter afbeelding te zien.



Afbeelding 4: GraphML XML hierarchie

7.4. Transformatie

Tijdens de transformatie moet er op de volgende punten worden gelet om de juiste output te krijgen.

- Modules zijn gekoppeld aan modules m.b.v. een sc_prim_channel tussen sc_port tags van de modules
- Poorten van modules worden aan poorten van submodules gekoppeld zonder het gebruik van een sc_prim_channel, maar door het gebruik van een hierarchical channel.
- Vanwege het gemis van de visualisatie van poorten wordt alleen de poort van de diepste module in de module hiërarchie gekoppeld aan een edge. Hierdoor ontstaat er één lijn tussen de juiste poorten, ook als de submodules in yEd zijn ingeklapt.

Modules

Modules worden omgezet naar nodes. Wanneer een module een submodule is wordt deze ook omgezet naar een node, maar dan wordt de parentmodule wel omgezet naar een groupnode. Een groupnode is een node met daarin een nieuw graph tag. Alle submodules in een module zijn dus in een graaf binnen de node gedefinieerd.

Porten

Op dit moment ondersteund yEd nog geen porten, daarom wordt op dit moment een module gebruikt als beginpunt/eindpunt van een channel ipv een port.

Channels

Channels worden omgezet naar edges. Edges zijn lijnen tussen twee Nodes. Een edge heeft altijd een begin en een eindpunt. Net zoals een channel tussen twee porten is verbonden.

8. Resultaat

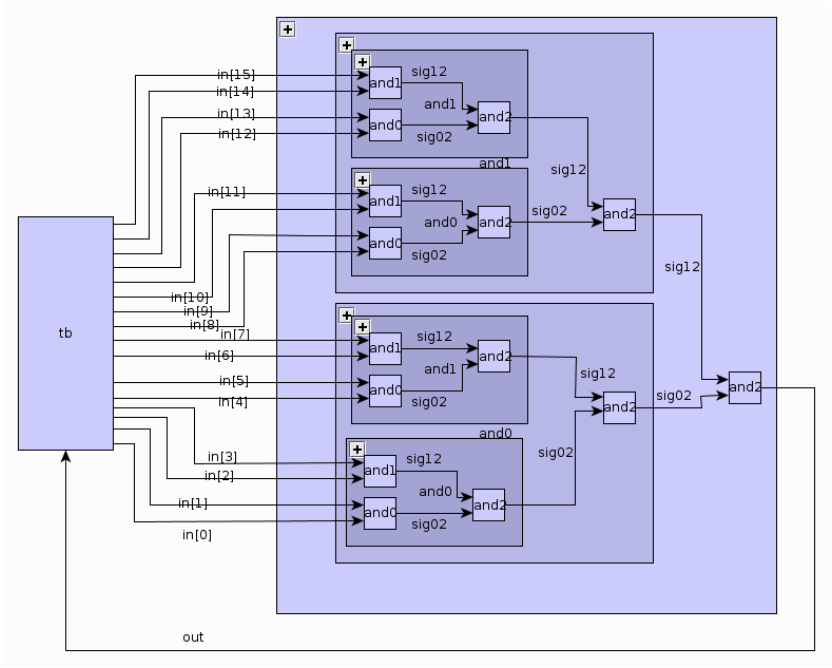
Het voorbeeld dat in hoofdstuk 2 is gebruikt om de eisen mee te bepalen kan helaas niet worden gebruikt voor de visualisatie, omdat het geen eis was om multi ports te visualiseren. In dit voorbeeld wordt hier wel gebruik van gemaakt. Daarom wordt er nieuw voorbeeld gebruikt om het resultaat toe te lichten.

8.1. SystemC model

```
template <unsigned int N>
struct Two_pow_N {
          static const unsigned int value = 2 * Two_pow_N<N-1>::value;
};

template <>
struct Two_pow_N<0> {
          static const unsigned int value = 1;
};

SC_MODULE(And2) {
          sc_in<bool> in[2];
          sc_out<bool> out;
          SC_CTOR(And2) {}
};

template <unsigned int N>
SC_MODULE(And2_N) {
          sc_in<bool> in[Two_pow_N<N>::value];
          sc_out<bool> out;
          SC_CTOR(And2_N): and0("and0"), and1("and1"), and2("and2") {
                    for (unsigned int i(0); i<Two_pow_N<N-1>::value; ++i) {
                              and0.in[i](in[i]);
                              and1.in[i](in[Two_pow_N<N-1>::value + i]);
                    }
                    and0.out(sig02);
                    and1.out(sig12);
                    and2.in[0](sig02);
                    and2.in[1](sig12);
                    and2.out(out);
          }
private:
          And2_N<N-1> and0, and1;
          And2 and2;
          sc_signal<bool> sig02, sig12;
};

template <>
struct And2_N<1>: public And2 {
          And2_N<1>(sc_module_name name): And2(name) {}
};

template <unsigned int N>
SC_MODULE(TB) {
public:
          sc_out<bool> stimulus[Two_pow_N<N>::value];
          sc_in<bool> response;
          SC_CTOR(TB){}
};

int sc_main(int argc, char* argv[]) {
          const int N = 4;
          sc_signal<bool> in[Two_pow_N<N>::value], out;

          TB<N> tb("tb");
          And2_N<N> dut("dut");

          for (unsigned int i(0); i<Two_pow_N<N>::value; ++i) {
                    tb.stimulus[i](in[i]);
                    dut.in[i](in[i]);
          }
          dut.out(out);
          tb.response(out);

          sc_start(1000, SC_NS);
          return 0;
}
```

8.2. Visualisatie

De visualisatie van de SCMDL die wordt gegenereerd wanneer het bovenstaande SystemC voorbeeld wordt geëxtraheerd, is te zien in afbeelding 5.



Afbeelding 5: Visualisatie SystemC model

9. Bronnen

OGDF
OGDF - Open Graph Drawing Framework
http://www.ogdf.net/forum/showthread.php?tid=174 <= extentie

yFiles
http://live.yworks.com/yfiles-ajax/html/Grouped_Graph.html
http://www.gupro.de/GXL/GraphModel/graphModel.html
http://docs.yworks.com/yfiles/doc/developers-guide/gml.html#ex_gml

yEd
http://www.yworks.com/en/products_yed_about.html

VGJ
http://www.eng.auburn.edu/department/cse/research/graph_drawing/vgj.html

Blod
https://sourceforge.net/projects/blod/

Graphlet
http://www.fim.uni-passau.de/en/fim/faculty/chairs/theoretische-informatik/projects.html

IP XACT
http://www.accellera.org/tech/community_support
TGI
http://www.accellera.org/XMLSchema/SPIRIT/1685-2009/TGI/TGI.html

10. Bijlages

10.1. Bijlage A

```
template <typename T , unsigned int ORDER>
SC_MODULE ( FIR ) {
   sc_in_clk clk ;
   sc_in<bool> reset ;
   sc_in<T> sample ;
   sc_out<T> result ;
   FIR ( sc_module_name name , const T coeff [ ORDER ] ) : sc_module ( name ) {
       for ( unsigned int i ( 0 ) ; i<MULT ; ++i ) {
           m [ i ] = new M<T>("" , coeff [ i ] ) ;
           m [ i]−>in ( sample ) ;
           m [ i]−>out ( mout [ i ] ) ;
       }
       for ( unsigned int i ( 0 ) ; i<ORDER ; ++i ) {
           s [ i ] = new S<T>("" ) ;
           s [ i]−>in1 ( dout [ i ] ) ;
           s [ i]−>in2 ( i<MULT−1 ? mout [ i+1] : mout [ ORDER −(i+1) ] ) ;
           if ( i==ORDER −1)
               s [ i]−>out ( result ) ;
           else
               s [ i]−>out ( sout [ i ] ) ;
           d [ i ] = new D<T>("" ) ;
           d [ i]−>clk ( clk ) ;
           d [ i]−>reset ( reset ) ;
           d [ i]−>in ( i==0 ? mout [ 0 ] : sout [ i −1]) ;
           d [ i]−>out ( dout [ i ] ) ;
       }
   }
   ˜ FIR ( ) {
       for ( unsigned int i ( 0 ) ; i<MULT ; ++i ) {
           delete m [ i ] ;
       }
       for ( unsigned int i ( 0 ) ; i<ORDER ; ++i ) {
           delete s [ i ] ;
           delete d [ i ] ;
       }
   }
private :
   static const unsigned int MULT = ORDER /2 + 1 ;
   M<T> ∗m [ MULT ] ;
   S<T> ∗s [ ORDER ] ;
   D<T> ∗d [ ORDER ] ;
   sc_signal<T> mout [ MULT ] , sout [ ORDER −1] , dout [ ORDER ] ;
};
```

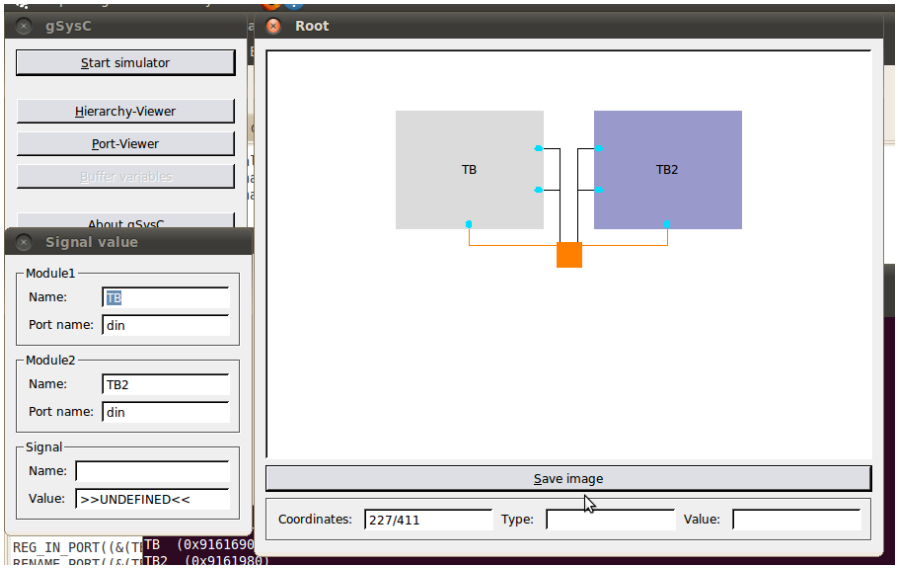
main functie:

```
sc_fixed <32 , 2> coeff [ 3 ];
read_from_file ( coeff , 3 );
FIR<sc_fixed <32 , 2>, 5> fir ( "fir" , coeff );
```

10.2. Bijlage B

```
graph [
        comment "This is a sample graph"
        directed 1
        id 42
        label "Hello, I am a graph"
        node [
                id 1
                label "Node 1"
                thisIsASampleAttribute 42
        ]
        node [
                id 2
                label "node 2"
                thisIsASampleAttribute 43
        ]
        node [
                id 3
                label "node 3"
                thisIsASampleAttribute 44
        ]
        edge [
                source 1
                target 2
                label "Edge from node 1 to node 2"
        ]
        edge [
                source 2
                target 3
                label "Edge from node 2 to node 3"
        ]
        edge [
                source 3
                target 1
                label "Edge from node 3 to node 1"
        ]
]
```

10.3. Bijlage D



10.4. Bijlage E